ALIBABA CLOUD

Alibaba Cloud

MaxCompute

Development

Document Version: 20220711

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Data types

1.1. Data type editions

MaxCompute V2.0 supports two data type editions: MaxCompute V2.0 data type edition and Hive-compatible data type edition. These editions are compatible with mainstream open source products. In addition to these editions, MaxCompute still supports the MaxCompute V1.0 data type edition.

Data type editions supported by MaxCompute

MaxCompute allows you to configure data type editions by using the following parameters:

- odps.sql.type.system.odps2: specifies whether to enable MaxCompute V2.0 data types. Valid values: true or false.
- odps.sql.decimal.odps2: specifies whether to enable the DECIMAL type in MaxCompute V2.0. Valid values: true or false.
- odps.sql.hive.compatible: specifies whether to enable Hive-compatible data types. Valid values: true or false. In the Hive-compatible data type edition, some data types and SQL statements are compatible with Hive.

When you add a project, MaxCompute allows you to choose a proper data type edition. Default settings for each data type edition:

- MaxCompute V1.0 data type edition

  ```
  setproject odps.sql.type.system.odps2=false; -- Disable MaxCompute V2.0 data types.
  setproject odps.sql.decimal.odps2=false; -- Disable the DECIMAL type in MaxCompute V2.0.
  setproject odps.sql.hive.compatible=false; -- Disable Hive-compatible data types.
  ```

  The MaxCompute V1.0 data type edition is suitable for early MaxCompute projects whose dependent components do not support the MaxCompute V2.0 data type edition.

- MaxCompute V2.0 data type edition

  ```
  setproject odps.sql.type.system.odps2=true; -- Enable MaxCompute V2.0 data types.
  setproject odps.sql.decimal.odps2=true; -- Enable the DECIMAL data type in MaxCompute V2.
  0.
  setproject odps.sql.hive.compatible=false; -- Disable Hive-compatible data types.
  ```

  The MaxCompute V2.0 data type edition is suitable for MaxCompute projects that do not contain baseline data before April 2020 and whose dependent components support the MaxCompute V2.0 data type edition.

- Hive-compatible data type edition

  ```
  setproject odps.sql.type.system.odps2=true; -- Enable MaxCompute V2.0 data types.
  setproject odps.sql.decimal.odps2=true; -- Enable the DECIMAL type in MaxCompute V2.0.
  setproject odps.sql.hive.compatible=true; -- Enable Hive-compatible data types.
  ```

  The Hive-compatible data type edition is suitable for MaxCompute projects that are migrated from Hadoop and whose dependent components support the MaxCompute V2.0 data type edition.

> **Note** For more information about data types that are supported by other Alibaba Cloud services or components, see Compatibility between components and data types.

Select a data type edition

The data type edition affects the following items:

- Data types of a table
- Execution of DML statements and built-in functions
- Development components that are closely related to MaxCompute

Before you select a data type edition for a project, we recommend that you read and fully understand the descriptions and differences of data type editions. For more information, see Differences between the MaxCompute V2.0 data type edition and other data type editions.

View the data type edition of a project

You can run the following command on the MaxCompute client, Query editor, MaxCompute Studio, or DataWorks to view the properties of an existing project.

```
setproject;
```

Check the values of odps.sql.type.system.odps2, odps.sql.decimal.odps2, and odps.sql.hive.compatible to determine the data type edition of the project.

Change the data type edition of a project

If the selected data type edition cannot meet your requirements, you can change the data type edition.

If you are the project owner or assigned the Super_Administrator role, you can run the following commands on the MaxCompute client, Query editor, MaxCompute Studio, or DataWorks to change the data type edition of the project.

```
-- View the data type edition of a project.
setproject;
-- Enable or disable MaxCompute V2.0 data types.
setproject odps.sql.type.system.odps2=true/false;
-- Enable or disable the DECIMAL type in MaxCompute V2.0.
setproject odps.sql.decimal.odps2=true/false;
-- Enable or disable Hive-compatible data types.
setproject odps.sql.hive.compatible=true/false;
```

We recommend that you change the data type edition of a project based on the following rules:

- The project uses the MaxCompute V2.0 data type edition, but some dependent components do not support MaxCompute V2.0 data types. You can use one of the following methods to address this issue:
  - Change the data type edition of the project to the MaxCompute V1.0 data type edition.

- Set the session-level data type edition to the MaxCompute V1.0 data type edition for the components that do not support the MaxCompute V2.0 data type edition.

  > **Note** Commit the following command with the commands in a session to set the data type edition of the session to the MaxCompute V1.0 data type edition. The command must be in lowercase.

  ```
  set odps.sql.type.system.odps2=false;
  ```

- The project uses the MaxCompute V2.0 data type edition. However, some SQL statements must use MaxCompute V1.0 data types, and some features, such as the current_timestamp function, use the MaxCompute V2.0 data types. You can use one of the following methods to address this issue:
  - Split SQL statements that require MaxCompute V1.0 data types into multiple sessions and set the data type edition for these sessions to the MaxCompute V1.0 data type edition.
  - Rewrite SQL statements.
- The project uses the MaxCompute V2.0 data type edition at the early stage. However, the project needs to use the MaxCompute V1.0 data type edition at later stages. You can use one of the following methods to address this issue:
  - To read the data of a table that uses MaxCompute V2.0 data types, convert data of the TINYINT, SMALLINT, or INT type to the BIGINT type, and data of the CHAR or VARCHAR type to the STRING type.
  - Create a table that uses MaxCompute V1.0 data types and then use the CAST function to import the data of a table that uses MaxCompute V2.0 data types into the created table.

1.2. MaxCompute V1.0 data type edition

The MaxCompute V1.0 data type edition is one of the three data type editions of MaxCompute. The MaxCompute V1.0 data type edition supports only MaxCompute V1.0 data types. This topic describes the MaxCompute V1.0 data type edition in terms of its definition, supported data types, and differences between this edition and other data type editions.

Definition

If the MaxCompute V1.0 data type edition is used in your project, the data types are defined based on the following code:

```
setproject odps.sql.type.system.odps2=false; -- Disable MaxCompute V2.0 data types.
setproject odps.sql.decimal.odps2=false; -- Disable the DECIMAL type in MaxCompute V2.0.
setproject odps.sql.hive.compatible=false; -- Disable Hive-compatible data types.
```

Scenarios

The MaxCompute V1.0 data type edition is suitable for scenarios in which your project is of an early MaxCompute version and depends on components that do not support the MaxCompute V2.0 data type edition.

Data types

Data type	Constant example	Description
BIGINT	100000000000L and -1L	The 64-bit signed integer type. Valid values: -2^{63} + 1 to 2^{63} -1.
DOUBLE	3.14159261E+7	The 64-bit binary floating point type.
DECIMAL	3.5BD and 99999999999.9999999BD	The exact numeric type based on the decimal system. Valid values of the integer part: -10^{36} + 1 to 10^{36} -1. The decimal part is accurate to 10^{-18}.
STRING	"abc", 'bcd', "alibaba", and 'inc'	The string type. The maximum length is 8 MB.
DATETIME	DATETIME'2017-11-11 00:00:00'	The DATETIME type. Valid values: 0000-01-01 to 9999-12-31.
BOOLEAN	True and False	The BOOLEAN type. Valid values: True and False.

When you use the MaxCompute V1.0 data type edition, take note of the following points:

- All the preceding data types support NULL values.
- By default, an integer constant is processed as the BIGINT type. If a constant exceeds the value range of the BIGINT type, such as 1,000,000,000,000,000,000,000,000, the constant is processed as the DOUBLE type. For example, the integer constant 1 in `SELECT 1 + a;` is processed as the BIGINT type.
- If the MaxCompute V1.0 data type edition is used, built-in functions that contain parameters of the MaxCompute V2.0 data types cannot be used.
- Partition key columns of partitioned tables must be of the STRING type.
- Constants of the STRING type can be concatenated. For example, abc and xyz can be concatenated as abcxyz.
- If a constant is inserted into a field of the DECIMAL type, the expression of the constant must conform to the format in the constant definition. Example: `3.5BD` in the following sample code:

  ```
  INSERT INTO test_tb(a) VALUES (3.5BD);
  ```

- Values of the DATETIME type do not include the millisecond component. You can add `-dfp` in Tunnel commands to specify the time format that is accurate to the millisecond, such as `tunnel upload -dfp 'yyyy-MM-dd HH:mm:ss.SSS'`. For more information about Tunnel commands, see Tunnel commands.

Differences between the MaxCompute V1.0 data type edition and other data type editions

- The LIMIT, ORDER BY, DISTRIBUTE BY, SORT BY, and CLUSTER BY operations are different between the MaxCompute V1.0 data type edition and the MaxCompute V2.0 data type edition.

  In this example, the `SELECT * FROM t1 UNION ALL SELECT * FROM t2 LIMIT10;` statement is used.

  - If the MaxCompute V1.0 data type edition is used, the statement is expressed as `SELECT * FROM t1 UNION ALL SELECT * FROM ( SELECT * FROM t2 LIMIT 10) t2;` .
  - If the MaxCompute V2.0 data type edition is used, the statement is expressed as `SELECT * FROM ( SELECT * FROM t1 UNION ALL SELECT * FROM t2 ) t LIMIT 10;` .

- The IN expressions are different between the MaxCompute V1.0 data type edition and the MaxCompute V2.0 data type edition.

  In this example, the `a IN (1, 2, 3)` expression is used.

  - If the MaxCompute V1.0 data type edition is used, all values enclosed in parentheses () must be of the same type.
  - If the MaxCompute V2.0 data type edition is used, all values enclosed in parentheses () can be implicitly converted into the same type.

Complex data types

Data type	Definition	Constructor
ARRAY	• `ARRAY<BIGINT>` • `ARRAY<STRUCT<a:BIGINT, b:STRING>>`	• `ARRAY(1, 2, 3)` • `ARRAY(NAMED_STRUCT('a', 1, 'b', '2'), NAMED_STRUCT('a', 3, 'b', '4'))`
MAP	• `MAP<STRING, STRING>` • `MAP<BIGINT, ARRAY<STRING>>`	• `MAP("k1", "v1", "k2", "v2")` • `MAP(1L, ARRAY('a', 'b'), 2L, ARRAY('x', 'y'))`
STRUCT	• `STRUCT<'x', BIGINT, 'y', BIGINT>` • `STRUCT<'field1', BIGINT, 'field2', ARRAY<BIGINT>, 'field3', MAP<BIGINT>>`	• `NAMED_STRUCT('x', 1, 'y', 2)` • `NAMED_STRUCT('field1', 100L, 'field2', ARRAY(1, 2), 'field3', MAP(1, 100, 2, 200))`

Note In MaxCompute, complex data types can be nested. For more information about the related built-in functions, see Other functions, Other functions, or Other functions.

1.3. MaxCompute V2.0 data type

edition

The MaxCompute V2.0 data type edition is one of the three data type editions of MaxCompute. This topic describes the MaxCompute V2.0 data type edition in terms of its definition, supported data types, and differences between this edition and other data type editions.

Definition

If the MaxCompute V2.0 data type edition is used in your project, the data types are defined based on the following code:

```
setproject odps.sql.type.system.odps2=true; -- Enable MaxCompute V2.0 data types.
setproject odps.sql.decimal.odps2=true; -- Enable the DECIMAL data type in MaxCompute V2.0.

setproject odps.sql.hive.compatible=false; -- Disable Hive-compatible data types.
```

Scenarios

The MaxCompute V2.0 data type edition is suitable for scenarios in which your project does not contain data generated before April 2020 and depends on components that support the MaxCompute V2.0 data type edition.

Basic data types

Data type	Constant example	Description
TINYINT	1Y and -127Y	The 8-bit signed integer type. Valid values: -128 to 127.
SMALLINT	32767S and -100S	The 16-bit signed integer type. Valid values: -32768 to 32767.
INT	1000 and -15645787	The 32-bit signed integer type. Valid values: -2^{31} to 2^{31} -1.
BIGINT	100000000000L and -1L	The 64-bit signed integer type. Valid values: -2^{63} + 1 to 2^{63} -1.
BINARY	unhex('FA34E10293CB42848573A4E39937F479')	A binary number. The maximum length is 8 MB.
FLOAT	cast(3.14159261E+7 as float)	The 32-bit binary floating point type.
DOUBLE	3.14159261E+7	The 64-bit binary floating point type.
		The precise numeric type based on the decimal system. • precision: indicates the number of digits in a value. Valid values: `1 t`

Data type	Constant example	Description
DECIMAL(precision,scale)	3.5BD and 99999999999.9999999BD	`o 38` . • scale: indicates the number of digits to the right of the decimal point in a value. Valid values: `0 to 18` . If the two parameters are not specified, the default expression of this data type is `decimal(38,18)` . **Note** • DECIMAL of the old and new editions cannot exist in the same table. • The Hive-compatible mode is enabled by running the `setproject odps.sql.hive.compatible=true;` command. In this case, if the number of digits to the right of the decimal point in a value of the `Decimal(precision,scale)` type exceeds the value of the scale parameter during data uploads by using the Tunnel Upload command or SQL operations, the value is rounded. If the integer part exceeds the limit, an error is reported.

Data type	Constant example	Description
VARCHAR(n)	None	The variable-length character type, in which n specifies the length. Valid values: 1 to 65535.
CHAR(n)	None	The fixed-length character type, in which n specifies the length. The maximum value is 255. If the length does not reach the specified value, extra spaces are automatically filled but are not involved in the comparison.
STRING	"abc", 'bcd', "alibaba", and 'inc'	The string type. The maximum length is 8 MB.
DATE	DATE'2017-11-11'	The DATE type, in the format of `yyyy-mm-dd`. Valid values: 0000-01-01 to 9999-12-31.
DATETIME	DATETIME'2017-11-11 00:00:00'	The DATETIME type. Valid values: 0000-01-01 00:00:00.000 to 9999-12-31 23:59:59.999, accurate to the millisecond.
TIMESTAMP	TIMESTAMP'2017-11-11 00:00:00.123456789'	The TIMESTAMP type. Valid values: 0000-01-01 00:00:00.000000000 to 9999-12-31 23:59:59.999999999, accurate to the nanosecond. **Note** The timestamp is independent of time zones. In any time zone, the timestamp stores a date offset value from Epoch (UTC 1970-01-01 00:00:00). You can use built-in functions to calculate data of the TIMESTAMP type based on time zones. For example, you can use the `cast(<a timestamp> as string)` function to convert data of the TIMESTAMP type into the STRING type based on the current time zone.

Data type	Constant example	Description
BOOLEAN	True and False	The BOOLEAN type. Valid values: True and False.

When you use the MaxCompute V2.0 data type edition, take note of the following points:

- All the preceding data types support NULL values.
- The INT keyword in an SQL statement refers to the 32-bit integer type.

```
-- Convert a into a 32-bit integer.
cast(a as INT)
```

- By default, an integer constant is processed as the INT type. For example, integer constant 1 in `SELECT 1 + a;` is processed as the INT type. If a constant exceeds the value range of the INT type but does not exceed the value range of the BIGINT type, the constant is processed as the BIGINT type. If the constant exceeds the value range of the BIGINT type, the constant is processed as the DOUBLE type.
- Implicit conversions
  - Some implicit conversions are disabled. If the data type is converted from STRING to BIGINT, from STRING to DATETIME, from DOUBLE to BIGINT, from DECIMAL to DOUBLE, or from DECIMAL to BIGINT, precision may be reduced, or errors may occur. You can use the CAST function to force the data type conversions.
  - VARCHAR constants can be implicitly converted into STRING constants.
- Tables, built-in functions, and user-defined functions (UDFs)
  - Built-in functions that require the MaxCompute V2.0 data type edition can be run.
  - The data types defined in UDFs are parsed and overloaded based on the MaxCompute V2.0 data type edition.
  - The data type of a partition key column can be STRING, VARCHAR, CHAR, TINYINT, SMALLINT, INT, or BIGINT.
- Constants of the STRING type can be concatenated. For example, abc and xyz can be concatenated as abcxyz.
- If a constant is inserted into a field of the DECIMAL type, the expression of the constant must conform to the format in the constant definition. Example: `3.5BD` in the following sample code:

```
insert into test_tb(a) values (3.5BD)
```

- Values of the DATETIME type do not include the millisecond component. You can add `-dfp` to Tunnel commands to specify the time format that is accurate to the millisecond, such as `tunnel upload -dfp 'yyyy-MM-dd HH:mm:ss.SSS'`. For more information about Tunnel commands, see Tunnel commands.

Complex data types

Data type	Definition	Constructor

Data type	Definition	Constructor
ARRAY	• `array<int>` • `array<struct<a:int, b:string>>`	• `array(1, 2, 3)` • `array(array(1, 2), array(3, 4))`
MAP	• `map<string, string>` • `map<smallint, array<string>>`	• `map("k1", "v1", "k2", "v2")` • `map(1S, array('a', 'b'), 2S, array('x', 'y'))`
STRUCT	• `struct<x:int, y:int>` • `struct<field1:bigint, field2:array<int>, field3:map<int, int>>`	• `named_struct('x', 1, 'y', 2)` • `named_struct('field1', 100L, 'field2', array(1, 2), 'field3', map(1, 100, 2, 200))`

> **Note**
>
> - Complex data types of MaxCompute can be nested. For more information about the related built-in functions, see ARRAY, MAP, or STRUCT.
> - We recommend that the size of complex data types of MaxCompute not exceed 1 MB. If the size of complex data types of MaxCompute exceeds 1 MB, an out of memory (OOM) error may occur during calculation.

Differences between the MaxCompute V2.0 data type edition and other data type editions

- DML execution rules are different.
  - The execution rules of LIMIT statements in set operations are different.

    In this example, the `SELECT * FROM t1 UNION ALL SELECT * FROM t2 limit 10;` statement is used.
    - If the MaxCompute V1.0 data type edition is used, the statement is expressed as `SELECT * FROM t1 UNION ALL SELECT * FROM ( SELECT * FROM t2 limit 10) t2;`.
    - If the MaxCompute V2.0 data type edition is used, the statement is expressed as `SELECT * FROM (SELECT * FROM t1 UNION ALL SELECT * FROM t2 ) t limit 10;`.

    This difference also applies to the ORDER BY, DISTRIBUTE BY, SORT BY, and CLUSTER BY clauses.
  - The parsing of data types in IN expressions is different.

    In this example, the `a in (1, 2, 3)` expression is used.
    - If the MaxCompute V1.0 data type edition is used, all values enclosed in parentheses () must be of the same type.
    - If the MaxCompute V2.0 data type edition is used, all values enclosed in parentheses () can be implicitly converted into the same type.

- The conversion rules for INSERT statements are different.
  - Hive-compatible data type edition: If a source data type can be explicitly converted into the data type of a table, MaxCompute automatically inserts a conversion function and runs it.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A source data type must be implicitly converted into the data type of a table. Otherwise, an error is returned.

    ```
    -- The following operations succeed in Hive-compatible mode but fail in other modes:
    create table t (a bigint);
    insert into table select 1.5;
    ```

- Function behavior is different.
  - `+`, `-`, `*`, `/`, and POW function
    - Hive-compatible data type edition: If the data exceeds the value range of a data type, the initial value is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the data exceeds the value range of a data type, an error is returned. In other modes, the value null is returned.
  - `>`, `>=`, `=`, `<`, and `<=`
    - Hive-compatible data type edition: The values of the DOUBLE type are directly compared.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the values of the DOUBLE type are compared, they are considered the same if the first 15 digits to the right of the decimal point are the same. Other digits after the decimal point are not compared.
  - Bitwise operators: `&`, `|`, and `^`
    - Hive-compatible data type edition: A value of the same data type as the input parameter is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A value of the BIGINT type is returned.
  - LENGTH, LENGTHB, FIND_IN_SET, INSTR, SIZE, HASH, and SIGN functions
    - Hive-compatible data type edition: A value of the INT type is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A value of the BIGINT type is returned.
  - FLOOR and CEIL
    - Hive-compatible data type edition: If the input parameter is of the DECIMAL type, a value of the DECIMAL type is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the input parameter is of the DECIMAL type, a value of the BIGINT type is returned.
  - ROUND
    - Hive-compatible data type edition: If the input parameter is of the TINYINT, SMALLINT, INT, or BIGINT type, a value of the same data type as the input parameter is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the input parameter is of the TINYINT, SMALLINT, INT, or BIGINT type, a value of the DATETIME type is returned.

- FROM_UNIXTIME
  - Hive-compatible data type edition: A value of the STRING type is returned.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A value of the DATETIME type is returned.
- CONCAT_WS
  - Hive-compatible data type edition: If a connected input string is NULL, the string is ignored.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If a connected input string is NULL, NULL is returned.
- FIND_IN_SET
  - Hive-compatible data type edition: An empty string is considered the matching of the tail of the string.

    ```
    -- Hive-compatible mode
    find_in_set("","")  1 is returned.
    find_in_set("", "a,") 2 is returned.
    ```

  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: An empty string is considered unmatched, and 0 is returned.
- REGEXP_(EXTRACT/REPLACE)
  - Hive-compatible data type edition: The REGEXP schema complies with the specifications of Java regular expressions.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: The REGEXP schema complies with MaxCompute specifications.
- SUBSTR

  ```
  ```

  start_position: required. A value of the BIGINT type. The default value is 1.

1.4. Hive-compatible data type edition

This topic describes the Hive-compatible data type edition in terms of its definition, supported data types, and differences with other data type editions. The Hive-compatible data type edition is one of the three data type editions of MaxCompute.

Description

If you want to use the Hive-compatible data type edition in your project, you must run the following code to enable the required data types:

```
setproject odps.sql.type.system.odps2=true; -- Enable MaxCompute V2.0 data types.
setproject odps.sql.decimal.odps2=true; -- Enable the DECIMAL data type in MaxCompute V2.0.

setproject odps.sql.hive.compatible=true; -- Enable Hive-compatible data types.
```

Scenarios

The Hive-compatible data type edition is suitable for MaxCompute projects that are migrated from Hadoop and whose dependent components support the MaxCompute V2.0 data type edition.

Basic data types

The basic data types in the Hive-compatible data type edition is similar to those defined in the MaxCompute V2.0 data type edition. The difference between these two editions lies only in the DECIMAL type.

Data type	Constant	Description
TINYINT	1Y and -127Y	The 8-bit signed integer type. Valid values: -128 to 127.
SMALLINT	32767S and -100S	The 16-bit signed integer type. Valid values: -32768 to 32767.
INT	1000 and -15645787	The 32-bit signed integer type. Valid values: -2^{31} to $2^{31} - 1$.
BIGINT	100000000000L and -1L	The 64-bit signed integer type. Valid values: $-2^{63} + 1$ to $2^{63} - 1$.
BINARY	unhex('FA34E10293CB42848573A4E39937F479')	A binary number. The maximum length is 8 MB.
FLOAT	cast(3.14159261E+7 as float)	The 32-bit binary floating point type.
DOUBLE	3.14159261E+7	The 64-bit binary floating point type.
DECIMAL(precision,scale)	3.5BD and 99999999999.9999999BD	The exact numeric type based on the decimal system. • precision: indicates the number of digits in a value. Valid values: `1 to 38`. • scale: indicates the number of digits to the right of the decimal point in a value. Valid values: `0 to 38`. If the two parameters are not specified, the default expression of this data type is `decimal(10,0)`.
VARCHAR(n)	None	The variable-length character type, in which n specifies the length. Valid values: 1 to 65535.

Data type	Constant	Description
CHAR(n)	None	The fixed-length character type, in which n specifies the length. The maximum value is 255. If the length does not reach the specified value, extra spaces are automatically filled but are not involved in the comparison.
STRING	"abc", 'bcd', "alibaba", and 'inc'	The STRING type. The maximum length is 8 MB.
DATE	DATE'2017-11-11'	The DATE type. The value is in the format of `yyyy-mm-dd`. Valid values: 0000-01-01 to 9999-12-31.
DATETIME	DATETIME'2017-11-11 00:00:00'	The DATETIME type. Valid values: 0000-01-01 00:00:00.000 to 9999-12-31 23.59:59.999.The value is accurate to the millisecond.
TIMESTAMP	TIMESTAMP'2017-11-11 00:00:00.123456789'	The TIMESTAMP type. The timestamp is independent of time zones. Valid values: 0000-01-01 00:00:00.000000000 to 9999-12-31 23.59:59.999999999. The value is accurate to the nanosecond. **Note** For some time zone-related functions, such as `cast(<a timestamp> as string)`, data of the TIMESTAMP type that is independent of time zones must be displayed based on the current time zone.
BOOLEAN	True and False	The BOOLEAN type. Valid values: True and False.

This section describes the data types:

- All the preceding data types can contain NULL values.
- The INT keyword in an SQL statement refers to the 32-bit integer type.

```
-- Convert the value a into a 32-bit integer.
cast(a as INT)
```

- By default, an integer constant is processed as the INT type. For example, integer constant 1 in `SELECT 1 + a;` is processed as the INT type. If a constant exceeds the value range of the INT type but does not exceed the value range of the BIGINT type, the constant is processed as the BIGINT type. If the constant exceeds the value range of the BIGINT type, the constant is processed as the DOUBLE type.
- Implicit conversions
  - Specific implicit conversions are disabled. For example, if the data type is converted from STRING to BIGINT, from STRING to DATETIME, from DOUBLE to BIGINT, from DECIMAL to DOUBLE, or from DECIMAL to BIGINT, the precision may be reduced, or errors may occur. You can use the CAST function to force the data type conversions.
  - VARCHAR constants can be implicitly converted into STRING constants.
- Tables, built-in functions, and user-defined functions (UDFs)
  - Built-in functions that require the MaxCompute V2.0 data type edition can be used.
  - The data types that are defined in UDFs are parsed and overloaded based on Hive-compatible data types.
  - The data type of a partition key column can be STRING, VARCHAR, CHAR, TINYINT, SMALLINT, INT, or BIGINT.
  - Specific functions do not support partition pruning in Hive-compatible mode. For more information, see Mappings between built-in functions of MaxCompute and built-in functions of Hive, MySQL, and Oracle.
- STRING constants can be combined. For example, abc and xyz can be combined as abcxyz.
- If a constant is inserted into a field of the DECIMAL type, the expression of the constant must conform to the format in the constant definition. For example, `3.5BD` is used in the following sample code.

```
insert into test_tb(a) values (3.5BD)
```

- Time values of the DATETIME type do not include the millisecond component. You can add `-dfp` to Tunnel commands to display milliseconds in the time values, such as `tunnel upload -dfp 'yyyy-MM-dd HH:mm:ss.SSS'`. For more information about Tunnel commands, see Tunnel commands.

Complex data types

Data type	Definition	Constructor
ARRAY	• `array<int>` • `array<struct<a:int, b:string>>`	• `array(1, 2, 3)` • `array(array(1, 2), array(3, 4))`
MAP	• `map<string, string>` • `map<smallint, array<string>>`	• `map("k1", "v1", "k2", "v2")` • `map(1S, array('a', 'b'), 2S, array('x', 'y'))`

Data type	Definition	Constructor
STRUCT	• `struct<x:int, y:int>` • `struct<field1:bigint, field2:array<int>, field3:map<int, int>>`	• `named_struct('x', 1, 'y', 2)` • `named_struct('field1', 100L, 'field2', array(1, 2), 'field3', map(1, 100, 2, 200))`

Note Complex data types of MaxCompute can be nested. For more information about the related built-in functions, see ARRAY, MAP, or STRUCT.

Differences between the Hive-compatible data type edition and other data type editions

- The conversion rules for INSERT statements are different.
  - Hive-compatible data type edition: If a source data type can be explicitly converted into the data type of a table, MaxCompute automatically inserts a conversion function and runs it.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A source data type must be implicitly converted into the data type of a table. Otherwise, an error is returned.

```
-- The following operations succeed in Hive-compatible mode but fail in other modes:
create table t (a bigint);
insert into table select 1.5;
```

- Function behavior is different.
  - `+`, `-`, `*`, `/`, and POW function
    - Hive-compatible data type edition: If the data exceeds the value range of a data type, the initial value is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the data exceeds the value range of a data type, an error is returned. In other modes, the value null is returned.
  - `>`, `>=`, `=`, `<`, and `<=`
    - Hive-compatible data type edition: The values of the DOUBLE type are directly compared.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the values of the DOUBLE type are compared, they are considered the same if the first 15 digits to the right of the decimal point are the same. Other digits after the decimal point are not compared.
  - Bitwise operators: `&`, `|`, and `^`
    - Hive-compatible data type edition: A value of the same data type as the input parameter is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A value of the BIGINT type is returned.
  - LENGTH, LENGTHB, FIND_IN_SET, INSTR, SIZE, HASH, and SIGN functions
    - Hive-compatible data type edition: A value of the INT type is returned.
    - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A value of the BIGINT type is returned.

- FLOOR and CEIL
  - Hive-compatible data type edition: If the input parameter is of the DECIMAL type, a value of the DECIMAL type is returned.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the input parameter is of the DECIMAL type, a value of the BIGINT type is returned.
- ROUND
  - Hive-compatible data type edition: If the input parameter is of the TINYINT, SMALLINT, INT, or BIGINT type, a value of the same data type as the input parameter is returned.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If the input parameter is of the TINYINT, SMALLINT, INT, or BIGINT type, a value of the DATETIME type is returned.
- FROM_UNIXTIME
  - Hive-compatible data type edition: A value of the STRING type is returned.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: A value of the DATETIME type is returned.
- CONCAT_WS
  - Hive-compatible data type edition: If a connected input string is NULL, the string is ignored.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: If a connected input string is NULL, NULL is returned.
- FIND_IN_SET
  - Hive-compatible data type edition: An empty string is considered the matching of the tail of the string.

    ```
    -- Hive-compatible mode
    find_in_set("","")  1 is returned.
    find_in_set("", "a,") 2 is returned.
    ```

  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: An empty string is considered unmatched, and 0 is returned.
- REGEXP_(EXTRACT/REPLACE)
  - Hive-compatible data type edition: The REGEXP schema complies with the specifications of Java regular expressions.
  - MaxCompute V1.0 data type edition and MaxCompute V2.0 data type edition: The REGEXP schema complies with MaxCompute specifications.
- SUBSTR

  ```
  ```

  start_position: required. A value of the BIGINT type. The default value is 1.

1.5. Compatibility between components and data types

MaxCompute can be integrated with multiple Alibaba Cloud components. Data types supported by these components are different. Some components support only MaxCompute V1.0 data types.

Component	MaxCompute V1.0 data type	MaxCompute V2.0 data type	Complex data type	Hive-compatible data type	DECIMAL type in MaxCompute V2.0
Storage	Supported.	Supported.	Supported.	Supported.	Supported.
SDK	Supported.	Supported.	Supported.	Supported.	Supported.
SQL	Supported.	Supported.	Supported.	Supported.	Supported.
MapReduce	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
OpenMapReduce	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
PAI	PAI supports data types other than DECIMAL.	PAI supports only the INT type of the newly added data types.	Some features support the MAP type.	Not supported.	Not supported.
PyODPS	Supported.	• DataFrame does not support the MaxCompute V2.0 data type edition. • Other features support the MaxCompute V2.0 data type edition.	Supported.	• DataFrame does not support the MaxCompute V2.0 data type edition. • Other features do not support the DATE type of the MaxCompute V2.0 data type edition.	Not supported.
JDBC	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
Graph	Supported.	Supported.	Supported.	Supported.	Not supported.
Tunnel	Supported.	Supported.	Supported.	Supported.	Supported.
DataWorks (Data Integration)	Supported.	Supported.	Supported.	Supported.	Supported.
DataX	Supported.	Not supported.	Not supported.	Not supported.	Not supported.

Component	MaxCompute V1.0 data type	MaxCompute V2.0 data type	Complex data type	Hive-compatible data type	DECIMAL type in MaxCompute V2.0
DataWorks (Data Quality)	Supported.	Supported.	Supported.	Supported.	Supported.
DataHub	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
Lightning	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
DataWorks (Data Map/Metadata)	Supported.	Supported.	Supported.	Supported.	Supported.
DataWorks (IDE)	Supported.	Supported.	Supported.	Supported.	Supported.
DataWorks (WebExcel)	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
DataWorks (Security Center)	Supported.	Supported.	Supported.	Supported.	Supported.
DataWorks (Data Security Guard)	Supported.	Supported.	Supported.	Supported.	Supported.

2.Common commands

2.1. Common SQL statements

This topic describes common SQL statements in MaxCompute.

MaxCompute allows you to perform operations on objects, such as projects, tables, resources, and instances. You can use tools or the MaxCompute SDK to perform operations on these objects.

> **Note** For more information about the MaxCompute SDK, see MaxCompute SDK.

Common SQL statements

The following table describes common SQL statements in MaxCompute.

Operation category	Common statement
Project operations	• Enter a project (use) • View project properties (setproject) • Configure project properties (setproject) • View account systems of a project (list accountproviders) • Add the RAM account system (add) • Remove the RAM account system (remove)
SET operations	• SET • SHOW FLAGS
Security operations	• Enable project data protection (ProjectProtection) • Add a trusted project (add trustedproject) • Remove a trusted project (remove trustedproject) • View trusted projects (list trustedprojects) • View the security configurations of a project (show SecurityConfiguration)
User and role operations	• Add a user (add user) • Remove a user (remove user) • View users (list users) • Create a role (create role) • View roles (list roles) • Grant a role to a user (grant) • Revoke a user from a role (revoke) • Delete a role (drop role)
Authorization operations	• Grant a role or user (grant) • Revoke role or user permissions (revoke)

Operation category	Common statement
Table operations	• Create a table (create table) • Change the owner of a table (alter table) • Delete a table (drop table) • View the information of a table or view (desc) • View the information of a partition (desc partition) • List tables (show tables) • List partitions (show partitions)
Partition and column operations	• Add partitions (alter table add) • Delete partitions (alter table drop) • Add columns or comments (alter table add columns) • Change the name and comment of a column (alter table change column)
Instance operations	• View instance information (show) • Query instance status (status) • View information of running instances (top instance) • Stop an instance (kill) • Obtain job information of an instance (desc) • Obtain the operational log of jobs in an instance (wait)
Resource operations	• Add resources (add) • View information of all resources (list) • Create an alias for a resource (alias) • Download a resource (get) • Delete a resource (drop)
Function operations	• Create a function (create function) • Delete a function (drop function) • View a function (desc function) • View the function list (list functions)
Tunnel operations	• Upload data (Tunnel upload) • Download data (Tunnel download)
Time zone configuration operations	Time zone configuration operations
Other operations	Cost estimation (cost sql)

Usage notes

Keywords, project names, table names, and column names in MaxCompute SQL statements are not case sensitive.

2.2. Project operations

After you create a MaxCompute project, you must go to the project to perform subsequent operations, such as development, analysis, and O&M. This topic describes common operations on projects, such as going to a project, viewing a project, configuring project-level properties for a project, or viewing the project-level properties of a project.

The following table describes common project operations.

Operation	Description	Role	Platform
Go to a project	Goes to a project on which you have access permissions.	Users who have project access permissions	Execute the statements described in this topic on the MaxCompute client.
View the properties of a project	Views the properties of a project.	Project owner	
Set project properties	Sets the properties of a project.	Project owner	
View the account systems of a project	Views the information of the account systems supported by a project.	Project owner	
Add the RAM account system	Adds the Resource Access Management (RAM) account system for a project.	Project owner	
Remove the RAM account system	Removes the RAM account system of a project.	Project owner	

Precautions

- A project is not a workspace. To obtain a project name, log on to the MaxCompute console and view the name in the **MaxCompute Project Name** column on the **Project management** tab.
- You cannot execute statements to create or delete projects in MaxCompute. For more information about how to create a project, see Create a MaxCompute project.

Go to a project

Goes to a specific project on which you have access permissions. After you go to the project, you can manage all objects in the project.

- Syntax

```
-- Go to a project.
use <project_name>;
```

- Parameter

  project_name: the name of the project to which you want to go. If the project does not exist or you do not have access permissions on the project, an error is returned.

- Examples
  - Example 1: Go to a specific project and access the objects in the project.

    ```
    -- The current project is my_project, and the project that you want to go to is my_proj
    ect_test. You have access permissions on the my_project_test project.
    odps@ my_project>use my_project_test;
    -- After you go to the my_project_test project, you can perform other operations. For e
    xample, query the test_src table in the my_project_test project.
    odps@ my_project_test>select * from test_src;
    ```

    MaxCompute searches for the test_src table in the my_project_test project. If the table exists, data in the table is returned. If the table does not exist, an error is returned.
  - Example 2: To access an object in a project from another project, grant permissions on the object and specify the name of the project to which the object belongs.

    ```
    -- Access the test_src table in the my_project2 project from the my_project_test projec
    t.
    odps@ my_project_test>select * from my_project2.test_src;
    ```

View the properties of a project

Views the project-level properties of a project. MaxCompute also allows you to view the session-level properties of a project. For more information, see SHOW FLAGS.

- Syntax

  ```
  setproject;
  ```

- Properties

  The following table describes common project properties.

Property (KEY)	Description	Valid value (VALUE)
odps.sql.allow.fullscan	Specifies whether to enable a full table scan on a project. A full table scan occupies a large number of resources, which reduces data processing efficiency. Therefore, we recommend that you do not enable this feature.	◦ true: A full table scan is enabled. ◦ false: A full table scan is disabled.

Property (KEY)	Description	Valid value (VALUE)
odps.table.lifecycle	Specifies whether to configure a lifecycle for tables in a project.	◦ optional: The lifecycle clause is optional in a table creation statement. If you do not configure a lifecycle for a table, the table does not expire. ◦ mandatory: The lifecycle clause is required in a table creation statement. ◦ inherit: If you do not configure a lifecycle for a table when you create the table, the value of odps.table.lifecycle.value is used by default.
odps.table.lifecycle.value	The lifecycle of a table. Unit: days.	1 to 37231. Default value: 37231.
odps.security.ip.whitelist	A whitelist of IP addresses that are authorized to access the project over the classic network. For more information, see Manage IP address whitelists.	A list of IP addresses that are separated by commas (,).
odps.security.vpc.whitelist	A whitelist of IP addresses that are authorized to access the project over a specific virtual private cloud (VPC). For more information, see Manage IP address whitelists.	RegionID_VPCID[IP Address].
READ_TABLE_MAX_ROW	The maximum number of data records that can be returned by a SELECT statement.	1 to 10000. Default value: 10000.
odps.sql.type.system.odps2	Specifies whether to enable the MaxCompute V2.0 data type edition. For more information about the MaxCompute V2.0 data type edition, see MaxCompute V2.0 data type edition.	◦ true: The MaxCompute V2.0 data type edition is enabled. ◦ false: The MaxCompute V2.0 data type edition is disabled.

Property (KEY)	Description	Valid value (VALUE)
odps.sql.hive.compatible	Specifies whether to enable the Hive-compatible data type edition. MaxCompute supports Hive syntax, such as `inputRecordReader`, `outputRecordReader`, and `SerDe`, only after the Hive-compatible data type edition is enabled. For more information about the Hive-compatible data type edition, see Hive-compatible data type edition.	◦ true: The Hive-compatible data type edition is enabled. ◦ false: The Hive-compatible data type edition is disabled.
odps.sql.decimal.odps2	Specifies whether to enable `DECIMAL(precision,scale)` in the MaxCompute V2.0 data type edition. For more information, see MaxCompute V2.0 data type edition.	◦ true: DECIMAL(precision,scale) in the MaxCompute V2.0 data type edition is enabled. ◦ false: DECIMAL(precision,scale) in the MaxCompute V2.0 data type edition is disabled.
odps.sql.metering.value.max	The upper limit on resources consumed by an SQL statement. For more information, see 消费监控告警Consumption control.	N/A.
odps.sql.timezone	The time zone of the MaxCompute project that you accessed. For more information about time zones, see Time zone configuration operations.	N/A.
odps.sql.unstructured.oss.commit.mode	Specifies whether to enable the multipart upload feature of Object Storage Service (OSS) to write data to OSS external tables. For more information, see Write data to OSS.	◦ true: The multipart upload feature is enabled. ◦ false: The multipart upload feature is disabled.

Property (KEY)	Description	Valid value (VALUE)
odps.sql.groupby.orderby.position.alias	Specifies whether to use integer constants in the `GROUP BY` and `ORDER BY` clauses as column IDs in SELECT statements. **Note** If this property is set to True for an existing project, data parsing or other operations may fail to be performed. Make sure that the original logic can be correctly executed for the existing project when you set this property to True. Otherwise, set the session-level property.	○ true: Integer constants in the GROUP BY and ORDER BY clauses can be used as column IDs in SELECT statements. ○ false: Integer constants in the GROUP BY and ORDER BY clauses cannot be used as column IDs in SELECT statements.
odps.forbid.fetch.result.by.bearertoken	Specifies whether to display the results of jobs on the **Result** tab of Logview. This parameter is used to protect data security.	○ true: The results of jobs are not displayed on the Result tab of Logview. ○ false: The results of jobs are displayed on the Result tab of Logview.

Set project properties

Sets the project-level properties of a project. This statement takes effect within 5 minutes after executed. You can check the result 5 minutes after the statement is successfully executed. MaxCompute also allows you to set session-level properties for a project. For more information, see SET.

- Syntax

```
setproject <KEY>=<VALUE>;
```

- Parameters
  - KEY: the name of the property.
  - VALUE: the value of the property. For more information about properties, see View project properties.
- Examples

  Enable a full table scan for a project.

```
setproject odps.sql.allow.fullscan=true;
```

View the account systems of a project

Views the information of the account systems supported by a project. The account systems include the Alibaba Cloud account system and RAM account system. Syntax:

```
list accountproviders;
```

> **Note** By default, MaxCompute identifies only the Alibaba Cloud account system. It cannot identify the RAM account system.

Add the RAM account system

Adds the RAM account system for a project. Syntax:

```
add accountprovider ram;
```

Remove the RAM account system

Removes the RAM account system of a project. Syntax:

```
remove accountprovider ram;
```

2.3. SET operations

MaxCompute allows you to set system variables for sessions. This topic describes how to set and view the system variables, which can affect MaxCompute operations.

The following table describes the statements that are used for SET operations.

Operation	Description	Role	Operation platform
SET	Sets MaxCompute system variables for the current session.	Users who have operation permissions on projects	You can execute the statements that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
SHOW FLAGS	Displays the properties that you configured by using the SET statement.		

SET

You can execute the SET statement to set MaxCompute system variables for the current session. MaxCompute also allows you to set properties for projects. For more information, see Set project properties.

- Syntax

  ```
  set <KEY>=<VALUE>
  ```

- Parameters

- KEY: the name of the property.
- VALUE: the value of the property.

The following table describes common properties at the session level.

Property	Description	Valid value
console.sql.result.instancetunnel	Specifies whether to enable InstanceTunnel. For more information, see Usage notes.	■ True: InstanceTunnel is enabled. ■ False: InstanceTunnel is disabled.
odps.stage.mapper.mem	Sets the memory size of each Map worker.	Valid values: 256 to 12288. Default value: 1024. Unit: MB.
odps.stage.reducer.mem	Sets the memory size of each Reduce worker.	Valid values: 256 to 12288. Default value: 1024. Unit: MB.
odps.stage.joiner.mem	Sets the memory size of each Join worker.	Valid values: 256 to 12288. Default value: 1024. Unit: MB.
odps.stage.mem	Sets the total memory size of all workers in a specified MaxCompute job. This property has a lower priority than the odps.stage.mapper.mem, odps.stage.reducer.mem, and odps.stage.joiner.mem properties.	Valid values: 256 to 12288. No default value.
odps.stage.mapper.split.size	Changes the input data amount of each Map worker, which is the split size of the input file. You can use this property to indirectly control the number of workers at each Map stage. **Note** If the LIMIT keyword is used in SQL statements, the number of concurrent workers is limited to 1. This way, only a single worker is allowed to run at a time. In this case, when you configure this parameter, we recommend that you do not add the LIMIT keyword to SQL statements.	Default value: 256. Unit: MB.

Property	Description	Valid value
odps.stage.reducer.num	Changes the number of workers at each Reduce stage. **Note** If the LIMIT keyword is used in SQL statements, the number of concurrent workers is limited to 1. This way, only a single worker is allowed to run at a time. In this case, when you configure this parameter, we recommend that you do not add the LIMIT keyword to SQL statements.	No default value.
odps.stage.joiner.num	Changes the number of workers at each Join stage. **Note** If the LIMIT keyword is used in SQL statements, the number of concurrent workers is limited to 1. This way, only a single worker is allowed to run at a time. In this case, when you configure this parameter, we recommend that you do not add the LIMIT keyword to SQL statements.	No default value.

Property	Description	Valid value
odps.stage.num	Changes the total number of concurrent workers in a specified MaxCompute job. This property has a lower priority than the odps.stage.mapper.split.size, odps.stage.reducer.mem, and odps.stage.joiner.num properties. **Note** If the LIMIT keyword is used in SQL statements, the number of concurrent workers is limited to 1. This way, only a single worker is allowed to run at a time. In this case, when you configure this parameter, we recommend that you do not add the LIMIT keyword to SQL statements.	No default value.
odps.sql.reshuffle.dynamicpt	Specifies whether to enable dynamic partitioning. Dynamic partitioning prevents the generation of a large number of small files.	■ True: Dynamic partitioning is enabled. ■ False: Dynamic partitioning is disabled. **Note** If a small number of dynamic partitions are generated, we recommend that you set this parameter to False to prevent data skew.
odps.sql.type.system.odps2	Specifies whether to enable the MaxCompute V2.0 data type edition. For more information about the MaxCompute V2.0 data type edition, see MaxCompute V2.0 data type edition.	■ True: The MaxCompute V2.0 data type edition is enabled. ■ False: The MaxCompute V2.0 data type edition is disabled.

Property	Description	Valid value
odps.sql.hive.compatible	Specifies whether to enable the Hive-compatible data type edition. MaxCompute supports Hive syntax, such as `inputRecordReader`, `outputRecordReader`, and `Serde`, only after the Hive-compatible data type edition is enabled. For more information about the Hive-compatible data type edition, see Hive-compatible data type edition.	■ True: The Hive-compatible data type edition is enabled. ■ False: The Hive-compatible data type edition is disabled.
odps.sql.executionengine.coldata.deep.buffer.size.max	Sets the cache size requested to store a column that contains complex data types when MaxCompute writes data to a table. This improves the writing performance. You can set this parameter if the output table contains excessive complex data types or the size of a variable of a specified complex data type in the output table is too large. ■ If the schemas of three columns in the output table are of complex data types, such as STRING, MAP, STRUCT, ARRAY, and BINARY, MaxCompute reserves 192 MB of memory (64 × 3) for table write operations by default. The cache requested for each column is used to store data in the row that is specified by `batch row count`. ■ If you estimate that the size of the variables of complex data types in the table is small, we recommend that you set this parameter to a small value. For example, if the size of each variable of a specified complex data type does not exceed 1,024 bytes and the default value (1024) of `batch row count` is used, you can set the value to 1048576 (1024 × 1024). `set odps.sql.executionengine.coldata.deep.buffer.size.max=1048576;` ■ If you estimate that the size of each variable of a complex data type is between 7 MB and 8 MB, and `batch row count` is set to 32, you can adjust this parameter to a value that equals 256 MB (8 × 32).	Unit: bytes. Default value: 67108864.

Property	Description	Valid value
	■ If the output table contains variables of complex data types or the small table involved in the `MAPJOIN` operation contains complex data types, the memory used for job running may be affected when you adjust the value of this parameter. If you use the preceding calculation method, a large value of this parameter may cause an out of memory (OOM) error.	
odps.sql.udf.getjsonobj.new	Specifies whether the GET_JSON_OBJECT function retains the original string when it returns a value. As of April 20, 2021, the GET_JSON_OBJECT function automatically retains the original strings for the returned results. For MaxCompute projects created before April 20, 2021, the GET_JSON_OBJECT function automatically returns JSON-formatted reserved characters by using escape characters. This prevents the impact on existing jobs. For more information about the GET_JSON_OBJECT function, see GET_JSON_OBJECT.	■ True: The original string is retained. ■ False: The original string is not retained.
odps.sql.udf.jvm.memory	Specifies the maximum memory size of the Java Virtual Machine (JVM) heap for a user-defined function (UDF). If a large amount of data is calculated and sorted in the memory by using a UDF, an out of memory (OMM) error may occur. In this case, you can increase the value of this parameter. However, you must optimize UDF code based on your business requirements to prevent OMM errors.	Valid values: 256 to 12288. Default value: 1024. Unit: MB.
odps.sql.udf.timeout	Specifies the timeout period of UDFs.	Valid values: 0 to 3600. Default value: 600. Unit: seconds.

Property	Description	Valid value
LabelSecurity	Specifies whether to enable LabelSecurity. For more information about LabelSecurity, see Label-based access control.	■ True: LabelSecurity is enabled. ■ False: LabelSecurity is disabled.
odps.sql.session.resources	Specifies resources that are referenced by user-defined types (UDTs). You can specify multiple resources and separate them with commas (,). For more information about how to reference resources, see Overview.	The uploaded resources.
odps.sql.udt.display.tostring	Specifies whether to enable the `java.util.Objects.toString(...)` mechanism on the Wrap for all the columns whose outputs are UDTs.	■ True: The java.util.Objects.toString(...) mechanism is enabled. ■ False: The java.util.Objects.toString(...) mechanism is disabled.
odps.sql.session.java.imports	Specifies Java packages that are referenced by UDTs. You can specify multiple Java packages and separate them with commas (,). For more information about how to reference Java packages, see Overview.	The uploaded Java package.
CheckPermissionUsingACL	Specifies whether to enable ACL-based access control. For more information about ACL-based access control, see Permissions.	■ True: ACL-based access control is enabled. ■ False: ACL-based access control is disabled.
CheckPermissionUsingPolicy	Specifies whether to enable policy-based access control. For more information about policy-based access control, see Policy-based access control.	■ True: Policy-based access control is enabled. ■ False: Policy-based access control is disabled.

Property	Description	Valid value
ObjectCreatorHasAccessPermission	Specifies whether creators of objects can perform operations on objects that are created by themselves.	■ True: By default, creators of objects can perform operations on objects that are created by themselves. ■ False: By default, creators of objects cannot perform operations on objects created by themselves.
ObjectCreatorHasGrantPermission	Specifies whether creators of objects can authorize other users to perform operations on objects.	■ True: By default, creators of objects can authorize other users to perform operations on objects. ■ False: By default, creators of objects cannot authorize other users to perform operations on objects.
ProjectProtection	Specifies whether to enable project data protection. For more information about project data protection, see Project data protection.	■ True: Project data protection is enabled. ■ False: Project data protection is disabled.
odps.sql.skewjoin	Specifies whether to enable the SKEWJOIN feature to prevent long tails.	■ True: The SKEWJOIN feature is enabled. ■ False: The SKEWJOIN feature is disabled.
odps.sql.skewinfo	Specifies the key and value on which SKEWJOIN is executed. For more information, see Optimize JOIN long tails	N/A.
odps.sql.reducer.instances	Specifies the number of hash buckets. For more information about hash buckets, see Table operations.	Valid values: 0 to 4000.
odps.sql.udf.ppr.deterministic	Specifies whether to enable partition pruning for UDFs. For more information about partition pruning, see WHERE clause (where_condition).	■ True: Partition pruning for UDFs is enabled. ■ False: Partition pruning for UDFs is disabled.

Property	Description	Valid value
odps.sql.udf.ppr.to.subquery	Specifies whether to ignore the error that occurs during result backfilling when partition pruning is being performed. For more information about partition pruning, see WHERE clause (where_condition).	■ True: The error is ignored. ■ False: The error is not ignored.
odps.optimizer.enable.range.partial.repartitioning	Specifies whether to enable the Shuffle Remove feature for range-clustered tables.	■ True: The Shuffle Remove feature is enabled. ■ False: The Shuffle Remove feature is disabled.
odps.optimizer.skew.join.topk.num	Specifies the number of hot key values obtained by the optimizer when it performs an Aggregate operation. For more information, see SKEWJOIN HINT.	N/A.
odps.optimizer.stat.collect.auto	Specifies whether to enable the Freeride feature. After the Freeride feature is enabled, the column stats metrics of tables are automatically collected. For more information, see Collect information for the optimizer of MaxCompute.	■ True: The Freeride feature is enabled. ■ False: The Freeride feature is disabled.
odps.optimizer.stat.collect.plan	Sets a collection plan to collect specific column stats metrics of specific columns. For more information, see Collect information for the optimizer of MaxCompute.	N/A.
odps.sql.executionengine.batch.rowcount	Specifies the number of data rows that can be processed by the SQL engine at a time. Default value: 1024. If an OOM error occurs because a large field exists in a row or if a timeout occurs due to poor UDF performance, you must set this parameter to a smaller value. We recommend that you do not set this parameter to a value too small because a small value may affect the performance of the SQL engine.	Valid values: 1 to 1024.
odps.sql.external.net.vpc	Specifies whether to enable virtual private cloud (VPC) for external tables. For more information, see Hologres external tables.	■ True: VPC is enabled for external tables. ■ False: VPC is disabled for external tables.

Property	Description	Valid value
odps.sql.groupby.position.alias	Specifies whether to process integer constants in the `GROUP BY` clause as column numbers in a SELECT statement.	■ True: Integer constants are processed as column numbers. ■ False: Integer constants are not processed as column numbers.
odps.sql.groupby.skewindata	Specifies whether to enable the anti-skew feature for the `GROUP BY` clause.	■ True: The anti-skew feature for the GROUP BY clause is enabled. ■ False: The anti-skew feature for the GROUP BY clause is disabled.
odps.sql.orderby.position.alias	Specifies whether to process integer constants in the `ORDER BY` clause as column numbers in a SELECT statement.	■ True: Integer constants are processed as column numbers. ■ False: Integer constants are not processed as column numbers.
odps.sql.groupby.orderby.position.alias	Specifies whether to process integer constants in the `GROUP BY` and `ORDER BY` clauses as column numbers in a SELECT statement. We **recommend** that you specify this parameter. **Note** If you set this parameter to true for projects in which tasks are running, the task parsing and running may be affected. Therefore, you must make sure that existing tasks can still run properly based on the original logic when you set the parameter to true. Otherwise, specify this parameter for sessions.	■ True: Integer constants are processed as column numbers. ■ False: Integer constants are not processed as column numbers.
odps.sql.jobconf.odps2	Specifies whether to enable jobConf2 to generate SQL execution plans.	■ True: JobConf2 is enabled. ■ False: JobConf2 is disabled.
odps.sql.mapjoin.memory.max	Specifies the size of small tables that are read to the memory when the MAPJOIN statement is executed. Unit: MB.	Valid values: 0 to 8192.

Property	Description	Valid value
odps.sql.metering.value.max	Specifies the upper limit on resources consumed by an SQL statement. For more information, see 消费监控告警 Consumption control.	N/A.
odps.sql.python.version	Specifies the Python version on which SQL statements are executed.	■ cp27 ■ cp37
odps.sql.select.output.format	Specifies whether to display the table headers in the returned results of the MaxCompute client. For more information, see MaxCompute client.	■ `{""needHeader"":false,""fieldDelim"":""""}` : The table headers are not displayed. ■ `{""needHeader"":true,""fieldDelim"":""""}` : The table headers are displayed.
odps.sql.timezone	Specifies the time zone of the MaxCompute project. For more information about time zones, see Time zone configuration operations.	N/A.
odps.sql.unstructured.data.oss.use.https	Specifies whether to obtain data by using HTTPS at the underlying layer when you use a built-in extractor to create external tables. For more information, see Create an OSS external table	■ True: This property is enabled. ■ False: This property is disabled.
odps.sql.unstructured.oss.commit.mode	Specifies whether to enable the multipart upload feature of Object Storage Service (OSS) to write data to external tables. For more information, see Write data to OSS.	■ True: Multipart upload is enabled. ■ False: Multipart upload is disabled.

- Example

```
-- Set the size of data read by each Mapper to 256 MB.
set odps.stage.mapper.split.size=256;
```

SHOW FLAGS

The SHOW FLAGS statement displays the properties that you configured by using the SET statement.
Syntax:

```
show flags;
```

2.4. Security operations

MaxCompute provides project data protection to ensure data security. This topic describes common statements that are used for security operations.

The following table describes common statements that are used for security operations. For more information , see Project security configurations.

Operation	Description	Role	Operation platform
Enable project data protection	Enables project data protection.	A project owner or a user assigned the Super_Administrator role	You must execute this statement on the MaxCompute client.
Add a trusted project	Adds a trusted project to the current project.		You can execute the statements described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Remove a trusted project	Removes an existing trusted project from the current project.		
View trusted projects	Views the trusted projects that have been added to the current project.	A project owner or a user assigned the Super_Administrator or Admin role	
View the security configurations of a project	Views the security configuration properties of the current project.		

Enable project data protection

Enables project data protection. You are allowed to access data only in projects. Data can be transferred only between trusted projects. This enhances data security.

Syntax:

```
ProjectProtection={True|False};
```

Add a trusted project

Adds a trusted project to the current project. This ensures smooth data transfer between trusted projects.

- Syntax

  ```
  add trustedproject <project_name>;
  ```

- Parameter

  project_name: required. The name of the trusted project that you want to add.

Remove a trusted project

Removes an existing trusted project from the current project.

- Syntax

  ```
  remove trustedproject <project_name>;
  ```

- Parameter

  project_name: required. The name of the trusted project that you want to remove.

View trusted projects

Views the trusted projects that have been added to the current project.

Syntax:

```
list trustedprojects;
```

View the security configurations of a project

Views the security configuration properties of the current project.

Syntax:

```
show SecurityConfiguration;
```

2.5. User and role operations

If you want to develop a MaxCompute project with another user, you can add this user to the MaxCompute project. If you want to add multiple users to a project and grant them the same permissions, you can create a role and grant the role permissions to these users at a time. This topic describes common operations on users and roles, such as add, remove, and view users or roles.

The following table describes common statements that are used for user and role operations.

Operation	Description	Role	Operation platform
Add a user	Adds an Alibaba Cloud account or a Resource Access Management (RAM) user to a MaxCompute project.	A project owner or a user assigned the Super_Administrator role	You can execute the statements described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • MaxCompute Studio
Remove a user	Removes an Alibaba Cloud account or a RAM user from a MaxCompute project.		
View users	Views user information in a MaxCompute project.		
Create a role	Creates a role in a MaxCompute project.		• MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
View roles	Views the information of all roles in a MaxCompute project.		
Grant a role to a user	Grants a role to a user.		
Revoke a user from a role	Revokes a user from a specific role.		

Operation	Description	Role	Operation platform
Delete a role	Deletes an existing role from a MaxCompute project.		

Add a user

Adds an Alibaba Cloud account or a RAM user to a MaxCompute project.

- Limits

  MaxCompute allows you to add only the RAM users that belong to your Alibaba Cloud account to a project. It does not allow you to add RAM users that belong to other Alibaba Cloud accounts.

- Syntax

```
add user <user_name>;
```

- Parameter

  user_name: required. The name of the Alibaba Cloud account or RAM user that you want to add. The format of an Alibaba Cloud account is `ALIYUN$****@aliyun.com;` . The format of a RAM user is `RAM$****` .

- Examples
  - Example 1: Add the Alibaba Cloud account test_user@aliyun.com to a MaxCompute project.

```
add user ALIYUN$test_user@aliyun.com;
```

  - Example 2: Add the RAM user ram_test_user to a MaxCompute project.

```
add user RAM$ram_test_user;
```

Remove a user

If a user leaves the MaxCompute project team, the user must be removed from the project. After the user is removed, the user is no longer authorized to access resources in the project.

- Limits
  - Before you remove a user who is assigned a role, you must revoke the role.
  - MaxCompute does not support the complete removal of a user and the relevant authorization data. After a user is removed from a project, permissions related to the user are retained. If the user is added to the project again, the historical access permissions of the user will be activated again.

- Syntax

```
remove user <user_name>;
```

- Parameter

  user_name: required. The name of the Alibaba Cloud account or RAM user that you want to remove. The format of an Alibaba Cloud account is `ALIYUN$****@aliyun.com;` . The format of a RAM user is `RAM$****` .

- Examples

- Example 1: Remove the Alibaba Cloud account test_user@aliyun.com.

```
remove user ALIYUN$test_user@aliyun.com;
```

- Example 2: Remove the RAM user ram_test_user.

```
remove user RAM$ram_test_user;
```

View users

Views user information in a MaxCompute project. Syntax:

```
list users;
```

Create a role

Creates a role in a MaxCompute project.

- Syntax

```
create role <role_name>;
```

- Parameter

  role_name: required. The name of the role that you want to add.

- Example

```
-- Create the player role.
create role player;
```

View roles

Views the information of all roles in a MaxCompute project.

Syntax:

```
list roles;
```

Grant a role to a user

Grants a role to a user so that the user has all the permissions of the role.

- Limits

  Before you grant a role to a user, you must grant the role the permissions on project objects. For more information, see Grant a role or user.

- Syntax

```
grant <role_name> to <user_name>;
```

- Parameters
  - role_name: required. The name of the role that you want to grant to a user.
  - user_name: required. The name of the Alibaba Cloud account or RAM user to which you grant the role. The format of an Alibaba Cloud account is `ALIYUN$****@aliyun.com;`. The format of a RAM user is `RAM$****`.

- Example

```
-- Grant the player role to the Alibaba Cloud account test_user@aliyun.com.
grant player to ALIYUN$test_user@aliyun.com;
```

Revoke a user from a role

Revokes a user from a specific role.

- Syntax

```
revoke <role_name> from <user_name>;
```

- Parameters
  - role_name: required. The name of the role from which you want to revoke a user.
  - user_name: required. The name of the Alibaba Cloud account or RAM user that you want to revoke. The format of an Alibaba Cloud account is `ALIYUN$****@aliyun.com;` . The format of a RAM user is `RAM$****` .
- Example

```
-- Revoke the Alibaba Cloud account alice@aliyun.com from the player role.
revoke player from ALIYUN$alice@aliyun.com;
```

Delete a role

Deletes an existing role from a MaxCompute project.

- Limits

  Before you delete a role, you must make sure that no users are assigned this role.
- Syntax

```
drop role <role_name>;
```

- Parameter

  role_name: required. The name of the role that you want to delete.
- Example

```
-- Delete the player role.
drop role player;
```

2.6. Authorization operations

If you add a user to a project, you must authorize the user to perform a specific operation on an object in the project. If you add a role to a project, you must grant operation permissions to the role and then grant the role to the user. The user has operation permissions on the object only after authorization. This topic describes how to grant and revoke the permissions of a user or role.

For more information about the objects and related operation types, see Permissions. The following table describes common statements that are used for authorization operations.

Operation	Description	Role	Operation platform
Grant a role or user	Grants a role or user a specific operation permission on a specific type of object.	A project owner or a user assigned the Super_Administrator or Admin role	You can execute the statements described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Revoke role or user permissions	Revokes a specific operation permission on a specific type of object from a role or user.		

Grant a role or user

Grants a role or user a specific operation permission on a specific type of object.

- Syntax

```
grant <Action> [, <Action>] on <Object> <object_name> to {user|role} {<user_name|role_name>};
```

- Parameters
  - Action: required. The name of the operation permission that you want to grant. For more information, see Permissions.
  - Object: required. The object type of the operation permission that you want to grant to the user or role. For more information, see Permissions.
  - object_name: required. The name of the object.
  - user|role: Select user or role. user indicates that the operation permission is granted to the user. role indicates that the operation permission is granted to the role.
  - user_name|role_name: the name of the user or role to which you grant the operation permission.
- Examples
  - Example 1: Grant the user test_user@aliyun.com the CREATE TABLE permission on the project prj1.

```
grant CreateTable on Project prj1 to user ALIYUN$test_user@aliyun.com;
```

  - Example 2: Grant the player role the CREATE INSTANCE permission on the project prj1.

```
grant CreateInstance on Project prj1 to role player;
```

Revoke role or user permissions

Revokes a specific operation permission on a specific type of object from a role or user.

- Syntax

```
revoke <Action> [, <Action>] on <Object> <object_name> from {user|role} {<user_name|role_name>};
```

- Parameters

- Action: required. The name of the operation permission that you want to revoke. For more information, see Permissions.
- Object: required. The object type of the operation permission that you want to revoke from the user or role. For more information, see Permissions.
- object_name: required. The name of the object.
- user|role: Select user or role for this parameter. user indicates that the operation permission is revoked from the user. role indicates that the operation permission is revoked from the role.
- user_name|role_name: the name of the user or role from which you revoke the operation permission.

- Examples
  - Example 1: Revoke the CREATE TABLE permission on the project prj1 from the user test_user@aliyun.com.

    ```
    revoke CreateTable on Project prj1 from user ALIYUN$test_user@aliyun.com;
    ```

  - Example 2: Revoke the CREATE INSTANCE permission on the project prj1 from the player role.

    ```
    revoke CreateInstance on Project prj1 from role player;
    ```

2.7. Table operations

In MaxCompute, tables are used to store data. When you develop, analyze, and maintain a data warehouse, you must process table data. This topic describes common table operations, such as creating, dropping, and viewing tables.

The following table describes the common statements that are used for the operations on tables. For more information about table operations, see Table operations.

Operation	Description	Role	Operation platform
Create a table.	Creates a non-partitioned table or partitioned table.	Users who have the CREATE TABLE permission on a project	You can execute the statements
Change the owner of a table	Changes the owner of a table.	Project owner	
Drop a table	Drops a non-partitioned table or partitioned table.	Users who have the DROP permission on tables	

Operation	Description	Role	Operation platform that are described in this topic on the following platforms:
View the information about tables or views	Views the information about MaxCompute views, internal tables, or external tables.	Users who have the DESCRIBE permission to read the metadata of a table	• MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
View the partition information	Views the partition information about a partitioned table.	Users who have the DESCRIBE permission to read the metadata of a table	
Display tables and views in a project	Displays all the tables and views or the tables and views that meet specific rules, such as regular expressions, in a project.	Users who have the LIST permission on objects in a project	
Display partitions	Displays all the partitions of a table.	Users who have the LIST permission on objects in a project	

For more information about the operations on partitions and columns, see Partition and column operations.

For more information about the operations on the lifecycle of tables, see Lifecycle management operations.

Create a table

Creates a non-partitioned table, a partitioned table, an external table, or a clustered table.

- Limits
  - A partitioned table can have a maximum of six levels of partitions. For example, if a table uses date columns as partition key columns, the six levels of the partitions are `year/month/week/day/hour/minute`.
  - By default, a table can have a maximum of 60,000 partitions. You can adjust the maximum number of partitions in a table based on your business requirements.

  For more information about the limits on tables, see MaxCompute SQL limits.
- Syntax

```
-- Create a table.
 create [external] table [if not exists] <table_name>
 [(<col_name> <data_type> [not null] [default <default_value>] [comment <col_comment>], .
..)]
 [comment <table_comment>]
 [partitioned by (<col_name> <data_type> [comment <col_comment>], ...)]
 -- Configure the shuffle and sort properties of a clustered table that you want to creat
e.
 [clustered by | range clustered by (<col_name> [, <col_name>, ...]) [sorted by (<col_nam
e> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> buckets]
 -- Used only for external tables.
 [stored by StorageHandler]
 -- Used only for external tables.
 [with serdeproperties (options)]
 -- Used only for external tables.
 [location <osslocation>]
 -- Set the table to a transactional table. You can later modify or delete the data of th
e transactional table. Transactional tables have specific limits. Create a transactional
table base on your business requirements.
 [tblproperties("transactional"="true")]
 [lifecycle <days>];
-- Create a table based on an existing table and replicate data from the existing table t
o the new table. Partition properties are not replicated.
create table [if not exists] <table_name> [lifecycle <days>] as <select_statement>;
-- Create a table that has the same schema as an existing table. Data in the existing tab
le is not replicated.
create table [if not exists] <table_name> like <existing_table_name> [lifecycle <days>];
```

- Parameters
  - external: optional. This parameter specifies that the table you want to create is an external table.
  - if not exists: optional. If you create a table by using the name of an existing table but do not specify the if not exists parameter, an error is returned. If you create a table by using the name of an existing table and specify the if not exists parameter, a success message is returned even if the schema of the existing table is different from the schema of the table that you want to create. If you create a table by using the name of an existing table, the table is not created and the metadata of the existing table is not changed.
  - table_name: required. The name of the table. The name must be 1 to 128 bytes in length, and can contain letters, digits, and underscores (_). The name must start with a letter and cannot contain special characters. The name is not case-sensitive. If the value of this parameter does not meet the requirements, an error is returned.
  - col_name: optional. The name of a table column. The name must be 1 to 128 bytes in length, and can contain letters, digits, and underscores (_). The name must start with a letter and cannot contain special characters. The name is not case-sensitive. If the value of this parameter does not meet the requirements, an error is returned.
  - col_comment: optional. The comment of a column. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.
  - data_type: optional. The data type of a column. The following data types are supported: BIGINT, DOUBLE, BOOLEAN, DATETIME, DECIMAL, and STRING. For more information about data types, see Data type editions.

- not null: optional. If you specify this parameter for a column, the values of the column cannot be NULL. For more information about how to modify the parameter, see Change the non-nullable property of a non-partition key column in a table.
- default_value: optional. The default value of the specified column. If a column is not specified in an `INSERT` operation, the default value is used for the column.
- table_comment: optional. The comment of a table. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.
- partitioned by (<col_name> <data_type> [comment <col_comment>], ...: optional. The partition fields of a partitioned table.
  - col_name: the name of a partition key column. The name must be 1 to 128 bytes in length, and can contain letters, digits, and underscores (_). The name must start with a letter and cannot contain special characters. The name is not case-sensitive. If the value of this parameter does not meet the requirements, an error is returned.
  - data_type: the data type of a partition key column. In the MaxCompute V1.0 data type edition, partition key columns must be of the STRING type. In the MaxCompute V2.0 data type edition, partition key columns can be of the TINYINT, SMALLINT, INT, BIGINT, VARCHAR, or STRING type. For more information, see Data type editions. If you use a partition field to partition a table, a full table scan is not required when you add partitions, update partition data, or read partition data. This improves the efficiency of data processing.
  - col_comment: the comment of a partition key column. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.

    > **Note** The value of a partition key column cannot contain double-byte characters, such as Chinese characters. It must start with a letter and can contain letters, digits, and supported special characters. It must be 1 to 128 bytes in length. The following special characters are supported: spaces, colons (:), underscores (_), dollar signs ($), number signs (#), periods (.), exclamation points (!), and at signs (@). The behavior of other characters is not defined, such as escaped characters `\t`, `\n`, and `/`.

- clustered by | range clustered by (<col_name> [, <col_name>, ...]) [sorted by (<col_name> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> buckets: optional. The shuffle and sort properties of the clustered table that you want to create.

  Clustered tables are classified into hash-clustered tables and range-clustered tables.

- Hash-clustered tables
  - CLUSTERED BY: the hash key. MaxCompute performs a hash operation on specified columns and distributes data to each bucket based on the hash values. To prevent data skew and hot spots and to improve the efficiency of concurrent executions, we recommend that you specify columns with a large value range and a small number of duplicate key values in `CLUSTERED BY`. To optimize the performance of `JOIN` operations, we recommend that you select commonly used join or aggregate keys. Join and aggregate keys are similar to primary keys in conventional databases.
  - SORTED BY: the sequence of fields in a bucket. To improve performance, we recommend that you keep the configuration of the SORTED BY clause consistent with that of the CLUSTERED BY clause. After you specify fields in the SORTED BY clause, MaxCompute automatically generates indexes, which can be used to accelerate data queries.
  - number_of_buckets: the number of hash buckets. This parameter is required and the value of this parameter varies based on the amount of data. By default, MaxCompute supports a maximum of 1,111 reducers. This means that MaxCompute supports a maximum of 1,111 hash buckets. You can run the `set odps.sql.reducer.instances=xxx;` command to increase the maximum number of hash buckets. However, the maximum number of hash buckets cannot exceed 4,000. Otherwise, performance may be affected.

    To maintain optimal performance, we recommend that you take note of the following rules when you specify the number of hash buckets:

    - Keep the size of each hash bucket around 500 MB. For example, if you want to add 1,000 hash buckets to a partition whose size is 500 GB, the size of each hash bucket is 500 MB on average. If a table contains a large amount of data, you can increase the size of each hash bucket from 500 MB to a size in the range of 2 GB to 3 GB. You can also run the `set odps.sql.reducer.instances=xxx;` command to set the maximum number of hash buckets to a value greater than 1111.
    - To optimize the performance of `JOIN` operations, we recommend that you do not configure the shuffle and sort properties for hash-clustered tables. The number of hash buckets of a table must be a multiple of the number of hash buckets of the other table. For example, one table has 256 hash buckets and the other table has 512 hash buckets. We recommend that you set the number of hash buckets to 2^n, such as 512, 1024, 2048, and 4096. This way, MaxCompute can automatically split and merge hash buckets. To improve execution efficiency, you can also skip the step of configuring the shuffle and sort properties for hash-clustered tables.

- Range-clustered tables
  - RANGE CLUSTERED BY: the range-clustered columns. MaxCompute performs the bucket operation on the specified columns and distributes data to each bucket based on the bucket ID.
  - SORTED BY: the sequence of fields in a bucket. You can use this parameter in the same way as you use it for a hash-clustered table.
  - number_of_buckets: the number of hash buckets. Compared with hash-clustered tables, range-clustered tables have no limits on the number of buckets when data is evenly distributed. If you do not specify the number of buckets in a range-clustered table, MaxCompute automatically determines the optimal number based on the amount of data.
  - If JOIN and AGGREGATE operations are performed on range-clustered tables and the join key or group key is the range-clustered key or the prefix of the range-clustered key, you can manage flags to disable shuffling. This improves execution efficiency. To control shuffling, you can set `odps.optimizer.enable.range.partial.repartitioning` to true or false. By default, this parameter is set to false, which indicates that shuffling is disabled.

    > **Note**
    > - Clustered tables help optimize the following aspects:
    >   - Bucket pruning
    >   - Aggregation
    >   - Storage
    > - Limits on clustered tables:
    >   - The `INSERT INTO` statement is not supported. You can add data only by using the `INSERT OVERWRITE` statement.
    >   - The data that is imported by using Tunnel commands is not arranged in order. Therefore, you cannot import data into a range-clustered table by using Tunnel commands.

- stored by StorageHandler: optional. This parameter specifies StorageHandler based on the data format of the external table.
- with serdeproperties (options): optional. The parameters related to the authorization, compression, and character parsing of the external table.
- osslocation: optional. The Object Storage Service (OSS) bucket where the data of the external table is stored. For more information, see Create an OSS external table and Use a custom extractor to access OSS.

- tblproperties("transactional"="true"): optional. -- Set the table to a transactional table. You can later perform the `UPDATE` or `DELETE` operation on the transactional table to update or delete data by rows. For more information, see UPDATE and DELETE.

  A transactional table has the following limits:

  - MaxCompute allows you to set a table to a transactional table only when you create the table. If you execute the `ALTER TABLE` statement to change an existing table to a transactional table, an error is returned.

    ```
    alter table not_txn_tbl set tblproperties("transactional"="true");
    -- The following error is returned:
    FAILED: Catalog Service Failed, ErrorCode: 151, Error Message: Set transactional is n
    ot supported
    ```

  - When you create a clustered table or an external table, you cannot set the clustered table or external table to a transactional table.
  - You cannot convert between transactional tables and MaxCompute internal tables, external tables, or clustered tables.
  - Jobs from other systems, such as MaxCompute Spark, Machine Learning Platform for AI, and Graph, cannot access transactional tables.
  - `CLONE TABLE` and `MERGE PARTITION` operations are not supported.
  - Before you execute the `UPDATE`, `DELETE`, or `INSERT OVERWRITE` statement for important data in transactional tables, you must execute the `SELECT` and `INSERT` statements to back up the data to other tables.

- lifecycle: optional. The lifecycle of the table. The value must be a positive integer. Unit: days.
  - Non-partitioned tables: If data in a non-partitioned table remains unchanged for the number of days specified by days after the last data update, MaxCompute executes a statement, such as DROP TABLE, to reclaim the table.
  - Partitioned tables: MaxCompute determines whether to reclaim a partition based on the value of LastDataModifiedTime. Unlike non-partitioned tables, a partitioned table is not deleted even if all of its partitions have been reclaimed. You can configure lifecycles for tables, but not for partitions.

- You can use the `create table [if not exists] <table_name> [lifecycle <days>] as <select_statement>;` statement to create a table and replicate data to the table. However, partition properties and the lifecycle attribute of the source table are not replicated to the created table. The partition key columns of the source table are considered common columns in the created table. You can also configure the lifecycle parameter to reclaim the table.
- You can use the `create table [if not exists] <table_name> like <existing_table_name> [lifecycle <days>];` statement to create a table that has the same schema as the source table. However, tables that are created by using this statement do not replicate data or replicate the lifecycle attribute of the source table. You can also configure the lifecycle parameter to reclaim the table.

- Examples
  - Example 1: Create a non-partitioned table named test1.

    ```
    create table test1 (key STRING);
    ```

- Example 2: Create a partitioned table named sale_detail.

```
create table if not exists sale_detail(
 shop_name     STRING,
 customer_id   STRING,
 total_price   DOUBLE)
partitioned by (sale_date STRING, region STRING);
```

- Example 3: Create a table named sale_detail_ctas1, replicate data from the sale_detail table to the sale_detail_ctas1 table, and then configure the lifecycle for the sale_detail_ctas1 table.

```
create table sale_detail_ctas1 lifecycle 10 as select * from sale_detail;
```

You can run the `desc extended sale_detail_ctas1;` command to view table details, such as the schema and lifecycle of a table.

The `sale_detail` table is a partitioned table, but the `sale_detail_ctas1` table that is created by using `create table ... as select_statement ...` does not replicate partition properties. The partition key columns of the source table are considered common columns in the created table. The `sale_detail_ctas1` is a non-partitioned table that has five columns.

- Example 4: Create the sale_detail_ctas2 table and use constants as column values in the `SELECT` clause.

```
-- Column names are specified.
create table sale_detail_ctas2
as
select shop_name, customer_id, total_price, '2013' as sale_date, 'China' as region
from sale_detail;
-- Column names are not specified.
create table sale_detail_ctas3
as
select shop_name, customer_id, total_price, '2013', 'China'
from sale_detail;
```

> **Note** If you use constants as column values in the `SELECT` clause, we recommend that you specify column names. In this example, the names of the fourth and fifth columns in the sale_detail_ctas3 table contain suffixes that are similar to `_c4` and `_c5`.

- Example 5: Create a table named sale_detail_like that uses the same schema as the sale_detail table and configure the lifecycle for the sale_detail_like table.

```
create table sale_detail_like like sale_detail lifecycle 10;
```

You can run the `desc extended sale_detail_like;` command to view table details, such as the schema and lifecycle of a table.

The schema of the sale_detail_like table is the same as that of the sale_detail table. The two tables have the same properties, such as column names, column comments, and table comments, aside from the lifecycle. However, data in the sale_detail table is not replicated to the sale_detail_like table.

- Example 6: Create a table named test_newtype that uses new data types.

```
set odps.sql.type.system.odps2=true;
CREATE TABLE test_newtype (
    c1 TINYINT
    ,c2 SMALLINT
    ,c3 INT
    ,c4 BIGINT
    ,c5 FLOAT
    ,c6 DOUBLE
    ,c7 DECIMAL
    ,c8 BINARY
    ,c9 TIMESTAMP
    ,c10 ARRAY<MAP<BIGINT,BIGINT>>
    ,c11 MAP<STRING,ARRAY<BIGINT>>
    ,c12 STRUCT<s1:STRING,s2:BIGINT>
    ,c13 VARCHAR(20))
LIFECYCLE 1
;
```

- Example 7: Create a hash-clustered table named t1. This table is a non-partitioned table.

```
create table t1 (a STRING, b STRING, c BIGINT) clustered by (c) sorted by (c) into 1024 buckets;
```

- Example 8: Create a hash-clustered table named t2. This table is a partitioned table.

```
create table t2 (a STRING, b STRING, c BIGINT) partitioned by (dt STRING) clustered by (c) sorted by (c) into 1024 buckets;
```

- Example 9: Create a hash-clustered table named t3. This table is a non-partitioned table.

```
create table t3 (a STRING, b STRING, c BIGINT) range clustered by (c) sorted by (c) into 1024 buckets;
```

- Example 10: Create a range-clustered table named t4. This table is a partitioned table.

```
create table t4 (a STRING, b STRING, c BIGINT) partitioned by (dt STRING) range clustered by (c) sorted by (c);
```

- Example 11: Create a transactional table named t5. This table is a non-partitioned table.

```
create table t5(id bigint) tblproperties("transactional"="true");
```

- Example 12: Create a transactional table t6. This table is a partitioned table.

```
create table if not exists t6(id bigint) partitioned by(ds string) tblproperties ("transactional"="true");
```

Change the owner of a table

Changes the owner of a table.

- Syntax

```
alter table <table_name> changeowner to <new_owner>;
```

- Parameters
  - table_name: required. The name of the table whose owner you want to change.
  - new_owner: required. The new owner of the table.
- Examples

```
-- Change the owner of the test1 table to ALIYUN$xxx@aliyun.com.
alter table test1 changeowner to 'ALIYUN$xxx@aliyun.com';
```

Drop a table

Drops a non-partitioned table or partitioned table.

- Usage notes
  - Before you drop a table, confirm that the table can be dropped. Proceed with caution. If you accidentally drop a table, you can restore the table if the backup and restoration feature is enabled for the project and the table is dropped within the backup data retention period specified for the project. For more information about the backup and restoration feature, see Backup and restoration.
  - After you drop a table, the volume of stored data in a MaxCompute project decreases.
- Syntax

```
drop table [if exists] <table_name>;
```

- Parameters
  - if exists: optional. If you do not specify the if exists parameter and the table that you want to drop does not exist, an error is returned. If you specify the if exists parameter, a success message is returned regardless of whether the table exists.
  - table_name: required. The name of the table that you want to drop.
- Example

```
-- Drop the sale_detail table. A success message is returned regardless of whether the sa
le_detail table exists.
drop table if exists sale_detail;
```

View the information about tables or views

Views the information about MaxCompute internal tables, views, external tables, clustered tables, or transactional tables. For more information about how to view detailed table information, see SELECT syntax.

- Syntax

```
-- View the information about a table or view.
desc <table_name|view_name> [partition (<pt_spec>)];
-- View the information about an external table, a clustered table, or a transactional ta
ble. You can also execute this statement to view extended information about an internal t
able.
desc extended <table_name>;
```

- Parameters
  -

-
- pt_spec: optional. The partition in the partitioned table that you want to view. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
- extended: This parameter is required if the table is an external table, a clustered table, or a transactional table. This parameter is used to query extended information about a table. You can also use this parameter to view extended information about an internal table, such as whether a column of the internal table can contain NULL values.

- Examples
  - Example 1: View the information about the test1 table.

    ```
    desc test1;
    ```

    The following result is returned:

    ```
    +------------------------------------------------------------------------------------+
    | Owner: ALIYUN$maoXXX@alibaba-inc.com | Project: $project_name                      |
    | TableComment:                                                                      |
    +------------------------------------------------------------------------------------+
    | CreateTime:               2020-11-16 17:47:48                                      |
    | LastDDLTime:              2020-11-16 17:47:48                                      |
    | LastModifiedTime:         2020-11-16 17:47:48                                      |
    +------------------------------------------------------------------------------------+
    | InternalTable: YES      | Size: 0                                                  |
    +------------------------------------------------------------------------------------+
    | Native Columns:                                                                    |
    +------------------------------------------------------------------------------------+
    | Field           | Type        | Label | Comment                                    |
    +------------------------------------------------------------------------------------+
    | key             | string      |       |                                            |
    +------------------------------------------------------------------------------------+
    ```

- Example 2: View the information about the sale_detail table.

```
desc sale_detail;
```

The following result is returned:

```
+------------------------------------------------------------------------+
| Owner: ALIYUN$maoXXX@alibaba-inc.com | Project: $project_name          |
| TableComment:                                                          |
+------------------------------------------------------------------------+
| CreateTime:               2017-06-28 15:05:17                          |
| LastDDLTime:              2017-06-28 15:05:17                          |
| LastModifiedTime:         2017-06-28 15:05:17                          |
+------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                      |
+------------------------------------------------------------------------+
| Native Columns:                                                        |
+------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                         |
+------------------------------------------------------------------------+
| shop_name       | string     |       |                                 |
| customer_id     | string     |       |                                 |
| total_price     | double     |       |                                 |
+------------------------------------------------------------------------+
| Partition Columns:                                                     |
+------------------------------------------------------------------------+
| sale_date       | string     |                                         |
| region          | string     |                                         |
+------------------------------------------------------------------------+
```

- Example 3: View the detailed information about the sale_detail_ctas1 table.

```
desc extended sale_detail_ctas1;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$maoXXX@alibaba-inc.com | Project: $project_name                      |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2021-07-07 15:29:53                                      |
| LastDDLTime:              2021-07-07 15:29:53                                      |
| LastModifiedTime:         2021-07-07 15:29:53                                      |
| Lifecycle:                10                                                       |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| shop_name | string |       |               | true     | NULL         |              |
| customer_id | string |       |               | true     | NULL         |
|
| total_price | double |       |               | true     | NULL         |
|
| sale_date | string |       |               | true     | NULL         |              |
| region    | string |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  98cb8a38733c49eabed4735173818147                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
+------------------------------------------------------------------------------------+
```

The sale_date and region columns are considered as common columns. They are not partition key columns.

- Example 4: View the information about the sale_detail_ctas2 table.

```
desc sale_detail_ctas2;
```

The following result is returned:

```
+------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name           |
| TableComment:                                                          |
+------------------------------------------------------------------------+
| CreateTime:               2017-06-28 15:42:17                          |
| LastDDLTime:              2017-06-28 15:42:17                          |
| LastModifiedTime:         2017-06-28 15:42:17                          |
+------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                      |
+------------------------------------------------------------------------+
| Native Columns:                                                        |
+------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                         |
+------------------------------------------------------------------------+
| shop_name       | string     |       |                                 |
| customer_id     | string     |       |                                 |
| total_price     | double     |       |                                 |
| sale_date       | string     |       |                                 |
| region          | string     |       |                                 |
+------------------------------------------------------------------------+
```

- Example 5: View the details about the sale_detail_like table.

```
desc extended sale_detail_like;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2021-07-07 15:40:38                                      |
| LastDDLTime:              2021-07-07 15:40:38                                      |
| LastModifiedTime:         2021-07-07 15:40:38                                      |
| Lifecycle:                10                                                       |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| shop_name | string |       |               | true     | NULL         |              |
| customer_id | string |       |               | true     | NULL         |
|
| total_price | double |       |               | true     | NULL         |
|
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| sale_date       | string     |                                                     |
| region          | string     |                                                     |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  61782ff7713f426e9d6f91d5deeac99a                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
+------------------------------------------------------------------------------------+
```

Aside from the lifecycle configuration, the properties, such as field types and partition types, of the sale_detail_like table are the same as those of the sale_detail table.

Note The data size in the output of the `DESC table_name` command includes the data size of the recycle bin. If you want to clear the recycle bin, execute the `PURGE TABLE table_name` statement. Then, execute the `DESC table_name` statement to view the size of data that excludes the size of data in the recycle bin. You can also execute the `SHOW RECYCLE BIN` statement to view the details about data in the recycle bin for the current project.

- Example 6: View the information about the test_newtype table.

```
desc test_newtype;
```

The following result is returned:

```
| Native Columns:                                                                                |
+------------------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                                 |
+------------------------------------------------------------------------------------------------+
| c1              | tinyint    |       |                                                         |
| c2              | smallint   |       |                                                         |
| c3              | int        |       |                                                         |
| c4              | bigint     |       |                                                         |
| c5              | float      |       |                                                         |
| c6              | double     |       |                                                         |
| c7              | decimal    |       |                                                         |
| c8              | binary     |       |                                                         |
| c9              | timestamp  |       |                                                         |
| c10             | array<map<bigint,bigint>> |       |                                                |
| c11             | map<string,array<bigint>> |       |                                                |
| c12             | struct<s1:string,s2:bigint> |       |                                              |
| c13             | varchar(20) |       |                                                       |
+------------------------------------------------------------------------------------------------+
OK
```

- Example 7: View the information of the t1 hash-clustered table. This table is a non-partitioned table. The clustering attribute is displayed in Extended Info.

```
desc extended t1;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2020-11-16 18:00:56                                      |
| LastDDLTime:              2020-11-16 18:00:56                                      |
| LastModifiedTime:         2020-11-16 18:00:56                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| a        | string |       |               | true     | NULL         |              |
| b        | string |       |               | true     | NULL         |              |
| c        | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  e6b06f705dc34a36a5b72e5af486cab7                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
| ClusterType:              hash                                                     |
| BucketNum:                1024                                                     |
| ClusterColumns:           [c]                                                      |
| SortColumns:              [c ASC]                                                  |
+------------------------------------------------------------------------------------+
OK
```

- Example 8: View the information about the t2 hash-clustered table. This table is a partitioned table. The clustering attribute is displayed in Extended Info.

  ```
  desc extended t2;
  ```

  The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime: 2017-12-25 11:18:26                                                    |
| LastDDLTime: 2017-12-25 11:18:26                                                   |
| LastModifiedTime: 2017-12-25 11:18:26                                              |
| Lifecycle: 2                                                                       |
+------------------------------------------------------------------------------------+
| InternalTable: YES | Size: 0                                                       |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field | Type   | Label | Comment                                                   |
+------------------------------------------------------------------------------------+
| a     | string |       |                                                           |
| b     | string |       |                                                           |
| c     | bigint |       |                                                           |
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| dt    | string |                                                                   |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID: 91a3395d3ef64b4d9ee1d2852755                                              |
| IsArchived: false                                                                  |
| PhysicalSize: 0                                                                    |
| FileNum: 0                                                                         |
| ClusterType: hash                                                                  |
| BucketNum: 1024                                                                    |
| ClusterColumns: [c]                                                                |
| SortColumns: [c ASC]                                                               |
+------------------------------------------------------------------------------------+
OK
```

- Example 9: View the information about the t3 range-clustered table. This table is a non-partitioned table. The clustering attribute is displayed in Extended Info.

```
desc extended t3;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:                2020-11-16 18:01:05                                     |
| LastDDLTime:               2020-11-16 18:01:05                                     |
| LastModifiedTime:          2020-11-16 18:01:05                                     |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| a        | string |       |               | true     | NULL         |              |
| b        | string |       |               | true     | NULL         |              |
| c        | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                   38d170aca2684f4baadbbe1931a6ae1f                        |
| IsArchived:                false                                                   |
| PhysicalSize:              0                                                       |
| FileNum:                   0                                                       |
| StoredAs:                  AliOrc                                                  |
| CompressionStrategy:       normal                                                  |
| ClusterType:               range                                                   |
| BucketNum:                 1024                                                    |
| ClusterColumns:            [c]                                                     |
| SortColumns:               [c ASC]                                                 |
+------------------------------------------------------------------------------------+
OK
```

- Example 10: View the information about the t4 range-clustered table. This table is a partitioned table. The clustering attribute is displayed in Extended Info.

```
desc extended t4;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2020-11-16 19:17:48                                      |
| LastDDLTime:              2020-11-16 19:17:48                                      |
| LastModifiedTime:         2020-11-16 19:17:48                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| a        | string |       |               | true     | NULL         |              |
| b        | string |       |               | true     | NULL         |              |
| c        | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| dt              | string     |                                                     |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  6ebc3432e283449188c861427bcd6ee4                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
| ClusterType:              range                                                    |
| BucketNum:                0                                                        |
| ClusterColumns:           [c]                                                      |
| SortColumns:              [c ASC]                                                  |
+------------------------------------------------------------------------------------+
OK
```

- Example 11: Check whether the t5 non-partitioned table is a transactional table.

  > **Note** We recommend that you use the MaxCompute client to check whether a table is a transactional table. The version of the MaxCompute client must be V0.35.4 or later. For more information about how to download and use the MaxCompute client, see MaxCompute client. Other tools may not be updated to display transactional information.

```
desc extended t5;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@aliyun.com | Project: $project_name                            |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2021-02-18 10:56:27                                      |
| LastDDLTime:              2021-02-18 10:56:27                                      |
| LastModifiedTime:         2021-02-18 10:56:27                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| id       | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
...
| Transactional:            true                                                     |
+------------------------------------------------------------------------------------+
```

- Example 12: Check whether the t6 partitioned table is a transactional table.

  > **Note** We recommend that you use the MaxCompute client to check whether a table is a transactional table. The version of the MaxCompute client must be V0.35.4 or later. For more information about how to download and use the MaxCompute client, see MaxCompute client. Other tools may not be updated to display transactional information.

  ```
  desc extended t6;
  ```

  The following result is returned:

  ```
  +------------------------------------------------------------------------------------+
  | Owner: ALIYUN$xxxxx@test.aliyunid.com | Project: $project_name                     |
  | TableComment:                                                                      |
  +------------------------------------------------------------------------------------+
  | CreateTime:               2021-02-18 15:34:54                                      |
  | LastDDLTime:              2021-02-18 15:34:54                                      |
  | LastModifiedTime:         2021-02-18 15:34:54                                      |
  +------------------------------------------------------------------------------------+
  | InternalTable: YES      | Size: 0                                                  |
  +------------------------------------------------------------------------------------+
  | Native Columns:                                                                    |
  +------------------------------------------------------------------------------------+
  | Field           | Type        | Label | Comment                                    |
  +------------------------------------------------------------------------------------+
  | id              | bigint      |       |                                            |
  +------------------------------------------------------------------------------------+
  | Partition Columns:                                                                 |
  +------------------------------------------------------------------------------------+
  | ds              | string      |                                                    |
  +------------------------------------------------------------------------------------+
  | Extended Info:                                                                     |
  +------------------------------------------------------------------------------------+
  ...
  | Transactional:            true                                                     |
  +------------------------------------------------------------------------------------+
  ```

View partition information

Views the partition information about a partitioned table.

- Syntax

  ```
  desc <table_name> partition (<pt_spec>);
  ```

- Parameters
  - table_name: required. The name of the partitioned table whose partition information you want to view.
  - pt_spec: required. The information about the partition that you want to view. The value of this parameter is in the `partition_col1=col1_value1, partition_col2=col2_value1...` format. If a table has multi-level partitions, you must specify the values of all the partition key columns.

- Example

```
-- Query partition information about the partitioned table sale_detail.
desc sale_detail partition (sale_date='201310',region='beijing');
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| PartitionSize: 2109112                                                             |
+------------------------------------------------------------------------------------+
| CreateTime:               2015-10-10 08:48:48                                      |
| LastDDLTime:              2015-10-10 08:48:48                                      |
| LastModifiedTime:         2015-10-11 01:33:35                                      |
+------------------------------------------------------------------------------------+
OK
```

Display tables and views in a project

Displays all the tables and views or the tables and views that meet specific rules in a project.

- Syntax

```
-- Display all the tables and views in a project.
show tables;
-- Display the tables or views whose names contain the chart keyword in a project.
show tables like '<chart>';
```

- Example

```
-- Display the tables whose names contain the sale* keyword in a project. The asterisk (*
) indicates any character.
show tables like 'sale*';
```

The following result is returned:

```
ALIYUN$account_name:sale_detail
......
-- ALIYUN is a system prompt, which indicates that the table is created by using an Aliba
ba Cloud account. If the table was created by a RAM user, the system prompt is RAM.
```

Display partitions

Displays all the partitions of a table. If the table does not exist or the table is a non-partitioned table, an error is returned.

- Syntax

```
show partitions <table_name>;
```

- Parameters

  table_name: required. The name of the partitioned table whose partition information you want to view.

- Example

```
-- Display all the partitions of the sale_detail table.
show partitions sale_detail;
```

The following result is returned:

```
sale_date=201310/region=beijing
sale_date=201312/region=shenzhen
sale_date=201312/region=xian
sale_date=2014/region=shenzhen
OK
```

2.8. Partition and column operations

This topic describes common operations on partitions or columns in a MaxCompute table, such as adding or deleting a partition and adding or changing the name and comment of a column. You can perform these operations based on your business requirements.

For more information about partition and column operations, see Partition and column operations. The following table describes common statements that are used for the operations on partitions and columns in a MaxCompute table.

Operation	Description	Role	Operation platform
Add partitions	Adds partitions to an existing partitioned table.	Users who have the ALTER permission on tables	You can execute the statements that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Delete partitions	Deletes partitions from an existing partitioned table.		
Add columns or column comments	Adds columns or column comments to an existing non-partitioned table or partitioned table.		
Change the name or comment of a column	Changes the name or comment of a column in a non-partitioned table or partitioned table.		

2.9. Instance operations

SQL, Spark, and MapReduce jobs submitted in the MaxCompute console are converted into MaxCompute instances. This topic describes common operations that can be performed on instances, such as querying the information and status of a specified instance, stopping an instance, and obtaining the operational logs of an instance.

Each MaxCompute instance has a unique ID. The instance ID is permanently valid. The following table describes common statements that are used for instance operations.

Operation	Description	Role	Platform

Operation	Description	Role	Platform
View instance information	Views instance information.	Users who have read permissions on instances or the LIST permission on the objects in a project	You can execute the statements that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Query the instance status	Queries the status of a specified instance.	Users who have read permissions on instances	
View information about the running instances	Views the information about instances that are running in a project.	Users who have read permissions on instances or the LIST permission on the objects in a project	
Stop an instance	Stops a specified instance that is in the Running state.	Users who have write permissions on instances	
Obtain job information about an instance	Obtains job information based on a specified instance ID.	Users who have read permissions on instances	
Obtain the operational log of jobs in an instance	Obtains the operational log of jobs based on a specified instance ID.		

View instance information

Views instance information. The information includes StartTime (in seconds), RunTime (in seconds), Status, InstanceID, Owner, and Query statements.

- Syntax

```
show p|proc|processlist|instances [from <startdate>] [to <enddate>] [-p <project_name>] [
-limit <number> | <number>] [-all];
ls|list instances [from <startdate>] [to <enddate>] [-p <project_name>] [-limit <number>
| <number>] [-all];
```

  The following statements are equivalent: `show p`, `show proc`, `show processlist`, `show instances`, `ls instances`, and `list instances`.

- Parameters
  - startdate and enddate: optional. The instance information that is submitted by a user within the period from startdate to enddate is returned. The date specified by the startdate parameter must be earlier than the date specified by the enddate parameter. The instance information that is submitted on the day that is specified by enddate is not included. The values of the two parameters must be in the `yyyy-mm-dd` format and are accurate to the day. If you do not configure the parameters, the instance information that is submitted on the current day is returned.
  - project_name: optional. The name of the MaxCompute project to which the instance that you want to query belongs. You must have the permissions to view the instance of the MaxCompute project. If you do not configure this parameter, the instance of the current MaxCompute project is queried.

- number: optional. The number of instances that you want to return.
- If you configure this parameter, information about N instances that is submitted at the time nearest to the current time is returned in chronological order. N is specified by the number parameter. If you do not configure this parameter, information about the instances that meet specific requirements is returned. `-limit <number>` and `number` are equivalent.
- -all: optional. Information about all instances that are run by the members of the MaxCompute project is returned. If you do not configure this parameter, information about the instances that are run by the current user in the MaxCompute project is returned.

  If the number parameter is not specified, information about up to 50 instances is returned by default. If the number parameter is configured, information about N instances is returned. N is specified by the number parameter.

- Examples
  - Example 1: View the information about all instances that are run by the current user in the current MaxCompute project on the current day. Sample statement:

    ```
    show p;
    ```

    The following result is returned:

    ```
    StartTime             RunTime  Status   InstanceID                    Owner
    Query
    2021-09-14 11:43:04   0s       Success  20210914**************3rw2    ALIYUN$****@test.
    aliyunid.com
    2021-09-14 11:43:05   1s       Success  20210914**************5t32    ALIYUN$****@test.
    aliyunid.com select date_sub(datetime '2005-03-01 00:00:00', 1);
    2021-09-14 11:58:13   0s       Success  20210914**************5pr2    ALIYUN$****@test.
    aliyunid.com
    2021-09-14 11:58:15   1s       Success  20210914**************5qgr    ALIYUN$****@test.
    aliyunid.com select date_sub(date '2005-02-28', -1);
    2021-09-14 12:02:15   1s       Success  20210914**************h8o7    ALIYUN$****@test.
    aliyunid.com select date_sub('2008-03-01 00:00:00', 2);
    2021-09-14 12:02:15   0s       Success  20210914**************5t32    ALIYUN$****@test.
    aliyunid.com
    2021-09-14 12:02:31   0s       Success  20210914**************5pr2    ALIYUN$****@test.
    aliyunid.com
    2021-09-14 12:02:32   0s       Success  20210914**************euq2    ALIYUN$****@test.
    aliyunid.com select date_sub('2005-03-01 00:00:00', 2);
    2021-09-14 13:35:42   0s       Success  20210914**************1ms2    ALIYUN$****@test.
    aliyunid.com
    2021-09-14 13:35:43   0s       Success  20210914**************j8o7    ALIYUN$****@test.
    aliyunid.com select date_sub(getdate(),1);
    2021-09-14 13:40:40   1s       Success  20210914**************h3wz    ALIYUN$****@test.
    aliyunid.com select date_sub(getdate(),0);
    2021-09-14 13:40:40   0s       Success  20210914**************9nm7    ALIYUN$****@test.
    aliyunid.com
    12 instances
    ```

- Example 2: View the information about the instances that are run by the current user in the current MaxCompute project within a specified period of time, and specify the number of instances whose information you want to query. Sample statement:

```
show instances from 2021-09-14 to 2021-09-15 -limit 10;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 11:58:13   0s       Success  20210914**************5pr2   ALIYUN$****@test.
aliyunid.com
2021-09-14 11:58:15   1s       Success  20210914**************5qgr   ALIYUN$****@test.
aliyunid.com    select date_sub(date '2005-02-28', -1);
2021-09-14 12:02:15   1s       Success  20210914**************h8o7   ALIYUN$****@test.
aliyunid.com    select date_sub('2008-03-01 00:00:00', 2);
2021-09-14 12:02:15   0s       Success  20210914**************5t32   ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:31   0s       Success  20210914**************5pr2   ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:32   0s       Success  20210914**************euq2   ALIYUN$****@test.
aliyunid.com    select date_sub('2005-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2   ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************j8o7   ALIYUN$****@test.
aliyunid.com    select date_sub(getdate(),1);
2021-09-14 13:40:40   1s       Success  20210914**************h3wz   ALIYUN$****@test.
aliyunid.com    select date_sub(getdate(),0);
2021-09-14 13:40:40   0s       Success  20210914**************9nm7   ALIYUN$****@test.
aliyunid.com
10 instances
```

- Example 3: View the information about the instances that are run by all users in another MaxCompute project within a specified period of time and specify the number of instances whose information you want to query. Sample statement:

```
ls instances from 2021-09-14 to 2021-09-15 -p doc_test_dev -all -limit 10;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 11:59:16   0s       Success  20210914**************6pr3    ALIYUN$****@test.
aliyunid.com
2021-09-14 11:59:20   1s       Success  20210914**************6qgr    ALIYUN$****@test.
aliyunid.com    select date_sub(date '2007-02-26', -1);
2021-09-14 12:02:19   1s       Success  20210914**************h8o7    ALIYUN$****@test.
aliyunid.com    select date_sub('2009-03-01 00:00:00', 2);
2021-09-14 12:02:25   0s       Success  20210914**************7t42    ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:37   0s       Success  20210914**************7pr2    ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:40   0s       Success  20210914**************emq2    ALIYUN$****@test.
aliyunid.com    select date_sub('2015-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2    ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************68o7    ALIYUN$kiki
select date_sub(getdate(),1);
2021-09-14 13:45:40   1s       Success  20210914**************73wz    ALIYUN$kiki
select date_sub(getdate(),0);
2021-09-14 13:45:45   0s       Success  20210914**************9nm7    ALIYUN$dreak
10 instances
```

- Example 4: View the information about the instances that are run by all users in another MaxCompute project on the current day, and specify the number of instances whose information you want to query. Sample statement:

```
show p -p doc_test_dev -all 5;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 12:02:40   0s       Success  20210914**************emq2    ALIYUN$****@test.
aliyunid.com    select date_sub('2015-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2    ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************68o7    ALIYUN$kiki
select date_sub(getdate(),1);
2021-09-14 13:45:40   1s       Success  20210914**************73wz    ALIYUN$kiki
select date_sub(getdate(),0);
2021-09-14 13:45:45   0s       Success  20210914**************9nm7    ALIYUN$dreak
5 instances
```

Query the instance status

Queries the status of a specified instance. If the instance is created by a different user, an error is returned when you query the status of this instance.

- Syntax

```
status <instance_id>;
```

- Parameter

  instance_id: required. The ID of an instance whose status you want to query. Each instance ID is unique.

- Output

  An instance can be in one of the following states:

  - Running: The instance is running.
  - Success: The instance operation succeeds.
  - Waiting: The instance is waiting to run.
  - Failed: The job fails, but the data in the destination table is not modified.
  - Cancelled: The instance is stopped.
  - Terminated: The job is completed.

- Example

```
-- Query the status of the instance whose ID is 20131225123xxxxxxxxxxxxxxx.
status 20131225123xxxxxxxxxxxxxxx;
```

  The following result is returned:

```
Success
```

View information about the running instances

Views the information about the instances that are running in a project. The information includes InstanceID, Owner, Type, StartTime, Progress, Status, Priority, RuntimeUsage (CPU/MEM), totalUsage (CPU/MEM), and QueueingInfo (POS/LEN).

- Syntax

```
-- View the information about the running instances that are submitted by the current acc
ount in the current project.
top instance;
-- Return the information about all or a specified number of running instances in the cur
rent project.
top instance [-all][-limit <number>];
```

- Parameters
  - -all: optional. Information about all jobs that are running in the current project is returned. By default, information about up to 50 jobs can be returned.
  - -limit number: optional. Information about a specified number of jobs that are running in the current project is returned.

- Example

```
-- Return the information about the first five instances that are running in the current
project.
top instance -limit 5;
```

The following result is returned:

```
QueueingInstances: 0 total.
```

Stop an instance

Stops a specified instance that is in the Running state.

> **Note** The execution of this statement is an asynchronous process. After the system accepts the request and returns a result, it does not mean that the distributed job has been stopped. Instead, it only means that the system has received the request. You must execute the `STATUS` statement to check the status of the instance.

- Syntax

```
kill <instance_id>;
```

- Parameter

  instance_id: the ID of an instance. Each instance ID is unique. Only an instance in the Running state can be specified. Otherwise, an error is returned.

- Example

```
-- Stop the instance whose ID is 20131225123xxxxxxxxxxxxxx.
kill 20131225123xxxxxxxxxxxxxx;
```

Obtain job information about an instance

Obtains job information based on a specified instance ID. The information includes SQL, Owner, StartTime, EndTime, and Status.

- Syntax

```
desc instance <instance_id>;
```

- Parameter

  instance_id: the ID of an instance. Each instance ID is unique.

- Example

```
-- Query job information about the instance whose ID is 20150715xxxxxxxxxxxxxx.
desc instance 20150715xxxxxxxxxxxxxx;
```

The following result is returned:

```
ID                                      20150715xxxxxxxxxxxxxxx
Owner                                   ALIYUN$XXXXXX@alibaba-inc.com
StartTime                               2015-07-15 18:34:41
EndTime                                 2015-07-15 18:34:42
Status                                  Terminated
console_select_query_task_1436956481295 Success
Query                                   select * from mj_test;
```

Obtain the operational log of jobs in an instance

Obtains the operational log of jobs based on a specified instance ID. The log includes the URL of Logview. You can view the log details of jobs in Logview.

> **Note** You can open the Logview only for the instances that are created in the last three days. For instances that are created for more than three days, you can obtain the Logview URLs from their logs.

- Syntax

```
wait <instance_id>;
```

- Parameter

  instance_id: the ID of an instance. Each instance ID is unique.

- Example

```
-- Query the operational log of jobs for the instance whose ID is 20170925161122379gxxxxx
x.
wait 20170925161122379gxxxxxx;
```

The following result is returned:

```
ID = 20170925161122379g3xxxxxx
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&p=alian&i=20
1709251611223xxxxxxdqp&token=XXXXXXvbiI6IjEifQ==
Job Queueing...
Summary:
resource cost: cpu 0.05 Core * Min, memory 0.05 GB * Min
inputs:
        alian.bank_data: 41187 (588232 bytes)
outputs:
        alian.result_table: 8 (640 bytes)
Job run time: 2.000
Job run mode: service job
Job run engine: execution engine
M1:
        instance count: 1
        run time: 1.000
        instance time:
                min: 1.000, max: 1.000, avg: 1.000
        input records:
                TableScan_REL5213301: 41187  (min: 41187, max: 41187, avg: 41187
)
        output records:
                StreamLineWrite_REL5213305: 8  (min: 8, max: 8, avg: 8)
R2_1:
        instance count: 1
        run time: 2.000
        instance time:
                min: 2.000, max: 2.000, avg: 2.000
        input records:
                StreamLineRead_REL5213306: 8  (min: 8, max: 8, avg: 8)
        output records:
                TableSink_REL5213309: 8  (min: 8, max: 8, avg: 8)
```

2.10. Resource operations

Resource is a concept specific to MaxCompute. To accomplish jobs by using user-defined functions (UDFs) or MapReduce, you must upload files as MaxCompute resources. This topic describes common operations on resources, such as adding, viewing, downloading, and deleting resources.

The following table describes common resource operations.

Operation	Description	Performed by	Operation platform

Operation	Description	Performed by	Operation platform
Add resources	Adds resources to a MaxCompute project.	Users who have the Write permission on resources.	You can run the commands that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
View resource information	Views the detailed information of a resource.	Users who have the Read permission on resources.	
View a resource list	Views the information of resources in the project.	Users who have the List permission on objects in a project	
Create an alias for a resource	Creates an alias for a resource.	Users who have the Write permission on resources.	
Download resources	Downloads resources in a MaxCompute project to your on-premises machine.	Users who have the Write permission on resources.	
Delete resources	Deletes existing resources from a MaxCompute project.	Users who have the Delete permission on resources.	

Add resources

Adds resources to a MaxCompute project.

- Limits
  - MaxCompute does not allow you to add external tables as resources.
  - The maximum size of a resource file is 500 MB. The size of resources referenced by a single SQL or MapReduce job cannot exceed 2,048 MB.
- Syntax

```
add file <local_file> [as <alias>] [comment '<comment>'][-f];
add archive <local_file> [as <alias>] [comment '<comment>'][-f];
add table <table_name> [partition (<spec>)] [as <alias>] [comment '<comment>'][-f];
add py|jar <localfile> [comment '<comment>'][-f];
```

- Parameters
  - file|archive|table|py|jar: required. The type of the resource. For more information about resource types, see Resource.
  - local_file: required. The path of the file that you want to add. The file name is used as the resource name, which uniquely identifies a resource.
  - table_name: required. The name of a table in MaxCompute.
  - spec: required. If the resource that you want to add is a partitioned table, MaxCompute takes only a partition in the table as a resource, instead of taking the whole table as a resource.
  - alias: optional. The name of the resource. If this parameter is not specified, the file name is used as the resource name. JAR packages or Python script files that are used as resources do not support this parameter.

- comment: optional. The comment of the resource.
- -f: optional. If a duplicate resource name exists, the existing resource is replaced. If you do not specify this option and a duplicate resource name exists, the resource fails to be added.

- Examples
  - Example 1: Add a file as a resource to a MaxCompute project.

    ```
    add file banking.txt;
    ```

    The following result is returned:

    ```
    OK: Resource 'banking.txt' have been created.
    ```

  - Example 2: Add a partitioned table whose alias is sale.res as a resource to MaxCompute.

    ```
    add table sale_detail partition (ds='20150602') as sale.res comment 'sale detail on 201
    50602' -f;
    ```

    The following result is returned:

    ```
    OK: Resource 'sale.res' have been updated.
    ```

  - Example 3: Add a Python file as a resource to MaxCompute.

    ```
    add py python.py [comment '<comment>'][-f];
    ```

    The following result is returned:

    ```
    OK: Resource 'python.py' have been created.
    ```

View resource information

Views the information of a specified resource in the project. The information includes the resource name, owner, resource type, resource size, creation time, last modification time, and MD5 value of the resource file.

- Syntax

  ```
  desc resource <resource_name>;
  ```

- Parameters

  resource_name: required. The name of an existing resource.

- Return value
  - name: the name of the resource.
  - Owner: the account that owns the resource.
  - Type: the type of the resource.
  - Comment: the comment of the resource.
  - CreatedTime: the time when the resource was created.
  - LastModifiedTime: the time when the resource was last updated.
  - LastUpdator: the account that performed the last update operation.
  - Size: the size of the resource file. Unit: MB.
  - Md5sum: the MD5 value of the resource file.

> **Note** The name of a MaxCompute resource is not case-sensitive. resource_A and resource_a represent the same resource.

- Examples

```
-- View information of resource topn_new.jar.
desc resource topn_new.jar;
```

The following result is returned:

```
Name                                topn_new.jar
Owner                               ALIYUN$****@test.aliyunid.com
Type                                JAR
Comment                             cloudopenapi
CreatedTime                         2020-12-29 13:55:11
LastModifiedTime                    2020-12-29 13:55:11
LastUpdator
Size                                11438795
Md5sum                              8bcf6aabf****56c0
```

View a resource list

Views the information of resources in a project. The information includes the resource name, owner, creation time, last modification time, and resource type.

- Syntax

```
list resources;
```

- Examples

```
list resources;
```

The following result is returned:

```
Resource Name         Owner                 Creation Time         Last Modified Time    T
ype          Last Updator Resource Size  Source         comment
getaddr.jar           ALIYUN$****           2020-06-18 15:47:28   2020-06-18 15:47:28   j
ar           1353716                             cloudopenapi
ip.dat                ALIYUN$****           2020-06-18 15:49:46   2020-06-18 15:49:46   f
ile          8525962                             cloudopenapi
2 resources
```

Create an alias for a resource

Creates an alias for a resource. The alias command can be used in MapReduce or UDF code to read different resources based on a specified resource name. During this process, you do not need to modify the code.

- Syntax

```
alias <alias>=<real>;
```

- Parameters
  - alias: the alias of the resource.

- real: the name of the resource.

- Examples

```
-- Add resources res_20121208 and res_20121209.
add table sale_detail partition (ds='20121208') as res_20121208;
add table sale_detail partition (ds='20121209') as res_20121209;
-- Set the alias of the resource res_20121208 to resName and call this resource.
alias resName=res_20121208;
jar -resources resName -libjars work.jar  -classpath ./work.jar com.company.MainClass arg
s ...;
-- Set the alias of the resource res_20121209 to resName and call this resource.
alias resName=res_20121209;
jar -resources resName -libjars work.jar  -classpath ./work.jar com.company.MainClass arg
s ...;
```

In this example, the `resName` alias references different resource tables in two jobs. You can read different copies of data without the need to modify the code.

Download resources

Downloads resources in a MaxCompute project to your on-premises machine. The resource type must be FILE, JAR, ARCHIVE, or PY. The TABLE type is not supported.

- Syntax

```
get resource <resource_name> <path>;
```

- Parameters
  - resource_name: required. The name of the resource that you want to download.
  - path: required. The path where the resource is saved on your on-premises machine.

- Examples

```
get resource getaddr.jar D:\;
```

Delete resources

Deletes existing resources from a MaxCompute project.

- Syntax

```
drop resource <resource_name>;
```

- Parameters

  resource_name: the name of the resource that you want to delete.

- Examples

```
drop resource getaddr.jar;
```

  The following result is returned:

```
Confirm to "drop resource getaddr.jar" (yes/no)? y
OK
```

2.11. Function operations

You can use built-in functions or user-defined functions (UDFs) of MaxCompute for data computing. Built-in functions can be called directly. UDFs can be called only after being customized. This topic describes common operations on functions, such as creating, deleting, and viewing functions.

The following table describes common operations on functions.

Operation	Description	Performed by	Operation platform
Create a UDF	Creates a UDF in a MaxCompute project.	Users who have the Write permission on functions	You can execute the statements described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Delete a UDF	Deletes an existing UDF from a MaxCompute project.	Users who have the Delete permission on functions	
View a UDF	Views the information of a specified UDF in a MaxCompute project.	Users who have the Read permission on functions	
View the UDF list	Views the information of all UDFs in a MaxCompute project.	Users who have the List permission on objects in a project	
View the built-in function list	Views the information of all built-in functions in the current MaxCompute project.		

Create a UDF

Creates a UDF in a MaxCompute project.

- Limits
  - Function names must be unique in a project. You cannot create a function that has the same name as an existing function in the project.
  - UDFs cannot overwrite built-in functions of MaxCompute. Only the project owner can use UDFs to overwrite built-in functions. If you use a UDF that overwrites a built-in function, warning information is displayed in Summary of the Logview of your job after the SQL statement is executed.
- Syntax

```
create function <function_name> as <'package_to_class'> using <'resource_list'>;
```

- Parameters
  - function_name: required. The name of the UDF that you want to create.

- package_to_class: required. The class of the UDF. This parameter is case-sensitive and must be enclosed in single quotation marks (').
  - For a Java UDF, specify this name as a fully qualified class name from the top-level package name to the UDF class name.
  - For a Python UDF, specify this name in the *Python script name.Class name* format.

    > **Note** The Python script name refers to the underlying resource name that uniquely identifies a resource. The resource name of MaxCompute is not case-sensitive. For example, the resource name is pyudf_test.py the first time you upload a resource. If you rename the resource to PYUDF_TEST.py in DataStudio or use PYUDF_TEST.py to overwrite pyudf_test.py on the MaxCompute client, the underlying resource name that uniquely identifies the resource is still pyudf_test.py. In this case, when you create a UDF based on the resource, the class name must be pyudf_test.SampleUDF. You can run the `list resource;` command to view the underlying resource names that uniquely identify all resources.

- resource_list: required. The list of resources used by the UDF.
  - The resource list must include the resource that contains UDF code. Make sure that the resource is uploaded to MaxCompute.
  - If the code calls the Distributed Cache API to read resource files, this resource list must also contain the list of resource files that are read by the UDF.
  - The resource list consists of multiple resource names and must be enclosed in single quotation marks ('). The resource names are separated by commas (,).
  - To specify the project that contains the resource, write the parameter in the `<project_name>/resources/<resource_name>` format.

- Examples
  - Example 1: Create the `my_lower` function. In this example, the Java UDF class `org.alidata.odps.udf.examples.Lower` is in *my_lower.jar*.

    ```
    create function my_lower as 'org.alidata.odps.udf.examples.Lower' using 'my_lower.jar';
    ```

  - Example 2: Create the `my_lower` function. In this example, the Python UDF class MyLower is in the *pyudf_test.py* script of the `test_project` project.

    ```
    create function my_lower as 'pyudf_test.MyLower' using 'test_project/resources/pyudf_test.py';
    ```

  - Example 3: Create the `test_udtf` function. In this example, the Java UDF class `com.aliyun.odps.examples.udf.UDTFResource` is in *udtfexample1.jar*. The function depends on the file resource *file_resource.txt*, the table resource *table_resource1*, and the archive resource *test_archive.zip*.

    ```
    create function test_udtf as 'com.aliyun.odps.examples.udf.UDTFResource' using 'udtfexample1.jar, file_resource.txt, table_resource1, test_archive.zip';
    ```

Delete a UDF

Deletes an existing UDF from a MaxCompute project.

- Syntax

  ```
  drop function <function_name>;
  ```

- Parameters

  function_name: required. The name of an existing UDF.

- Example

```
-- Delete the my_lower function.
drop function my_lower;
```

View a UDF

Views the information of a specified UDF in a MaxCompute project. The information includes the name, owner, creation time, class name, and resource list of the UDF.

- Syntax

```
desc function <function_name>;
```

- Parameters

  function_name: required. The name of an existing function.

- Return value
  - Name: the name of the UDF.
  - Owner: the account that owns the UDF.
  - Created Time: the time when the UDF is created.
  - Class: the class of the UDF, which is case-sensitive.
  - Resources: the list of resources that are used by the UDF.

- Example

```
-- View the information of the my_lower function.
desc function my_lower;
```

  The following result is returned:

```
Name                                    my_lower
Owner                                   ALIYUN$****
Created Time                            2020-06-18 15:50:19
Class                                   org.alidata.odps.udf.examples.Lower
Resources                               project_name/my_lower.jar
```

View the UDF list

Views the information of all UDFs in a MaxCompute project.

- Syntax

```
list functions [-p <project_name>];
```

- Parameters

  project_name: optional. The name of a MaxCompute project.

- Example

```
list functions;
```

The following result is returned:

```
Name              Owner                                      Create Time           Clas
s                              Resources
ipv4_ipv6_aton    ALIYUN$****@aliyun.com 2021-11-15 13:42:14   com.aliyun.odps.udf.udfFun
ction.IpLocation ipv4.txt,ipv6.txt,udf-1.0-SNAPSHOT.jar
Lower_test        ALIYUN$****@aliyun.com 2021-08-25 15:51:22   com.aliyun.odps.udf.exampl
e.Lower udf-1.0-SNAPSHOT.jar
my_add            ALIYUN$****@aliyun.com 2021-05-08 11:26:02
my_index          ALIYUN$****@aliyun.com 2021-08-25 12:01:05   com.aliyun.odps.examples.u
df.UdfArray udf-1.0-SNAPSHOT.jar
my_sum            ALIYUN$****@aliyun.com 2021-05-08 10:24:58
my_udtf           ALIYUN$****@aliyun.com 2021-02-23 11:37:30   com.aliyun.odps.examples.u
df.UDTFResource udf-1.0-SNAPSHOT.jar
numpy             ALIYUN$****@aliyun.com 2020-11-11 14:12:50   numpy.TryImport
numpy.py,numpy-1.19.4-cp37-cp37m-manylinux1_x86_64.zip
ST_Aggr_ConvexHull ALIYUN$****@aliyun.com 2021-03-18 17:06:29   com.esri.hadoop.hive.ST_A
ggr_ConvexHull esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Aggr_Intersection ALIYUN$****@aliyun.com 2021-03-18 17:06:29   com.esri.hadoop.hive.ST
_Aggr_Intersection esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Aggr_Union     ALIYUN$****@aliyun.com 2021-03-18 17:06:30   com.esri.hadoop.hive.ST_Ag
gr_Union esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Area           ALIYUN$****@aliyun.com 2021-03-18 17:06:30   com.esri.hadoop.hive.ST_Ar
ea      esri-geometry-api.jar,spatial-sdk-hive.jar
ST_AsBinary       ALIYUN$****@aliyun.com 2021-03-18 17:06:30   com.esri.hadoop.hive.ST_As
Binary  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_AsGeoJson      ALIYUN$****@aliyun.com 2021-03-18 17:06:49   com.esri.hadoop.hive.ST_As
GeoJson esri-geometry-api.jar,spatial-sdk-hive.jar
ST_AsJson         ALIYUN$****@aliyun.com 2021-03-18 17:06:50   com.esri.hadoop.hive.ST_As
Json    esri-geometry-api.jar,spatial-sdk-hive.jar
ST_AsShape        ALIYUN$****@aliyun.com 2021-03-18 17:06:50   com.esri.hadoop.hive.ST_As
Shape   esri-geometry-api.jar,spatial-sdk-hive.jar
ST_AsText         ALIYUN$****@aliyun.com 2021-03-18 17:06:50   com.esri.hadoop.hive.ST_As
Text    esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Bin            ALIYUN$****@aliyun.com 2021-03-18 17:06:50   com.esri.hadoop.hive.ST_Bi
n       esri-geometry-api.jar,spatial-sdk-hive.jar
ST_BinEnvelope    ALIYUN$****@aliyun.com 2021-03-18 17:07:01   com.esri.hadoop.hive.ST_Bi
nEnvelope esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Boundary       ALIYUN$****@aliyun.com 2021-03-18 17:07:01   com.esri.hadoop.hive.ST_Bo
undary  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Buffer         ALIYUN$****@aliyun.com 2021-03-18 17:07:01   com.esri.hadoop.hive.ST_Bu
ffer    esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Centroid       ALIYUN$****@aliyun.com 2021-03-18 17:07:01   com.esri.hadoop.hive.ST_Ce
ntroid  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Contains       ALIYUN$****@aliyun.com 2021-03-18 17:07:01   com.esri.hadoop.hive.ST_Co
ntains  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_ConvexHull     ALIYUN$****@aliyun.com 2021-03-18 17:07:13   com.esri.hadoop.hive.ST_Co
nvexHull esri-geometry-api.jar,spatial-sdk-hive.jar
ST_CoordDim       ALIYUN$****@aliyun.com 2021-03-18 17:07:14   com.esri.hadoop.hive.ST_Co
ordDim  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Crosses        ALIYUN$****@aliyun.com 2021-03-18 17:07:14   com.esri.hadoop.hive.ST_Cr
osses   esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Difference     ALIYUN$****@aliyun.com 2021-03-18 17:07:14   com.esri.hadoop.hive.ST_Di
fference esri-geometry-api.jar,spatial-sdk-hive.jar
```

```
ST_Dimension      ALIYUN$****@aliyun.com 2021-03-18 17:07:14   com.esri.hadoop.hive.ST_Di
mension esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Disjoint       ALIYUN$****@aliyun.com 2021-03-18 17:07:31   com.esri.hadoop.hive.ST_Di
sjoint  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Distance       ALIYUN$****@aliyun.com 2021-03-18 17:07:31   com.esri.hadoop.hive.ST_Di
stance  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_EndPoint       ALIYUN$****@aliyun.com 2021-03-18 17:07:31   com.esri.hadoop.hive.ST_En
dPoint  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Envelope       ALIYUN$****@aliyun.com 2021-03-18 17:07:32   com.esri.hadoop.hive.ST_En
velope  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_EnvIntersects  ALIYUN$****@aliyun.com 2021-03-18 17:07:32   com.esri.hadoop.hive.ST_En
vIntersects esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Equals         ALIYUN$****@aliyun.com 2021-03-18 17:07:44   com.esri.hadoop.hive.ST_Eq
uals    esri-geometry-api.jar,spatial-sdk-hive.jar
ST_ExteriorRing   ALIYUN$****@aliyun.com 2021-03-18 17:07:44   com.esri.hadoop.hive.ST_Ex
teriorRing esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeodesicLengthWGS84 ALIYUN$****@aliyun.com 2021-03-18 17:07:44   com.esri.hadoop.hive.
ST_GeodesicLengthWGS84 esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeomCollection ALIYUN$****@aliyun.com 2021-03-18 17:07:44   com.esri.hadoop.hive.ST_Ge
omCollection esri-geometry-api.jar,spatial-sdk-hive.jar
ST_Geometry       ALIYUN$****@aliyun.com 2021-03-18 17:07:44   com.esri.hadoop.hive.ST_Ge
ometry  esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeometryN      ALIYUN$****@aliyun.com 2021-03-18 17:07:55   com.esri.hadoop.hive.ST_Ge
ometryN esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeometryType   ALIYUN$****@aliyun.com 2021-03-18 17:07:55   com.esri.hadoop.hive.ST_Ge
ometryType esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeomFromGeoJson ALIYUN$****@aliyun.com 2021-03-18 17:07:55   com.esri.hadoop.hive.ST_G
eomFromGeoJson esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeomFromJson   ALIYUN$****@aliyun.com 2021-03-18 17:07:55   com.esri.hadoop.hive.ST_Ge
omFromJson esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeomFromShape  ALIYUN$****@aliyun.com 2021-03-18 17:07:56   com.esri.hadoop.hive.ST_Ge
omFromShape esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeomFromText   ALIYUN$****@aliyun.com 2021-03-18 17:08:10   com.esri.hadoop.hive.ST_Ge
omFromText esri-geometry-api.jar,spatial-sdk-hive.jar
ST_GeomFromWKB    ALIYUN$****@aliyun.com 2021-03-18 17:08:10   com.esri.hadoop.hive.ST_Ge
omFromWKB esri-geometry-api.jar,spatial-sdk-hive.jar
```

View the built-in function list

Views the information of all built-in functions in the current MaxCompute project.

- Syntax

```
show functions;
```

- Example

```
show functions;
```

The following result is returned:

```
ID = 20211214091641326gg0g****
::::ABS SCALAR  INT ABS(INT arg0),DOUBLE ABS(DOUBLE arg0),DECIMAL(?,?) ABS(DECIMAL(?,?) a
rg0),BIGINT ABS(BIGINT arg0)
::::ACOS        SCALAR  DOUBLE ACOS(DOUBLE arg0),DOUBLE ACOS(DECIMAL(?,?) arg0)
::::ADD_MONTHS  SCALAR  STRING ADD_MONTHS(TIMESTAMP arg0, BIGINT arg1),STRING ADD_MONTHS(
STRING arg0, BIGINT arg1),STRING ADD_MONTHS(DATE arg0, BIGINT arg1)
::::ALL_MATCH   SCALAR  BOOLEAN ALL_MATCH(ARRAY<T> arg0, java.util.function.Function<T, j
ava.lang.Boolean> arg1)
::::ANY_MATCH   SCALAR  BOOLEAN ANY_MATCH(ARRAY<T> arg0, java.util.function.Function<T, j
ava.lang.Boolean> arg1)
::::ANY_VALUE   AGGREGATOR      T ANY_VALUE([DISTINCT] T arg1)
::::APPROX_DISTINCT     AGGREGATOR      BIGINT APPROX_DISTINCT([DISTINCT] P arg1, DOUBLE
arg2),BIGINT APPROX_DISTINCT([DISTINCT] P arg1)
::::ARG_MAX     AGGREGATOR      R ARG_MAX([DISTINCT] T arg1, R arg2)
::::ARG_MIN     AGGREGATOR      R ARG_MIN([DISTINCT] T arg1, R arg2)
::::ARRAY       SCALAR  ARRAY<STRING> ARRAY(),ARRAY<T> ARRAY(T arg0...)
::::ARRAYS_OVERLAP      SCALAR  BOOLEAN ARRAYS_OVERLAP(ARRAY<T> arg0, ARRAY<T> arg1)
::::ARRAYS_ZIP  SCALAR  null
::::ARRAY_CONTAINS      SCALAR  BOOLEAN ARRAY_CONTAINS(ARRAY<T> arg0, T arg1)
::::ARRAY_DISTINCT      SCALAR  ARRAY<T> ARRAY_DISTINCT(ARRAY<T> arg0)
::::ARRAY_EXCEPT        SCALAR  ARRAY<T> ARRAY_EXCEPT(ARRAY<T> arg0, ARRAY<T> arg1)
::::ARRAY_INTERSECT     SCALAR  ARRAY<T> ARRAY_INTERSECT(ARRAY<T> arg0, ARRAY<T> arg1)
::::ARRAY_JOIN  SCALAR  STRING ARRAY_JOIN(ARRAY<STRING> arg0, STRING arg1),STRING ARRAY_J
OIN(ARRAY<STRING> arg0, STRING arg1, STRING arg2)
::::ARRAY_MAX   SCALAR  T ARRAY_MAX(ARRAY<T> arg0)
::::ARRAY_MIN   SCALAR  T ARRAY_MIN(ARRAY<T> arg0)
::::ARRAY_POSITION      SCALAR  BIGINT ARRAY_POSITION(ARRAY<T> arg0, T arg1)
::::ARRAY_REDUCE        SCALAR  OUT ARRAY_REDUCE(ARRAY<IN> arg0, BUF arg1, java.util.func
tion.BiFunction<BUF, IN, BUF> arg2, java.util.function.Function<BUF, OUT> arg3)
::::ARRAY_REMOVE        SCALAR  ARRAY<T> ARRAY_REMOVE(ARRAY<T> arg0, T arg1)
::::ARRAY_REPEAT        SCALAR  ARRAY<T> ARRAY_REPEAT(T arg0, BIGINT arg1)
::::ARRAY_SORT  SCALAR  ARRAY<T> ARRAY_SORT(ARRAY<T> arg0, java.util.function.BiFunction<
T, T, java.lang.Long> arg1)
::::ARRAY_UNION SCALAR  ARRAY<T> ARRAY_UNION(ARRAY<T> arg0, ARRAY<T> arg1)
::::ASCII       SCALAR  BIGINT ASCII(STRING arg0)
::::ASIN        SCALAR  DOUBLE ASIN(DOUBLE arg0),DOUBLE ASIN(DECIMAL(?,?) arg0)
::::ATAN        SCALAR  DOUBLE ATAN(DOUBLE arg0),DOUBLE ATAN(DECIMAL(?,?) arg0)
::::AT_MOST_ONE_ROW     AGGREGATOR      T AT_MOST_ONE_ROW([DISTINCT] T arg1)
::::AVG AGGREGATOR      DECIMAL(?,?) AVG([DISTINCT] DECIMAL(?,?) arg1),DOUBLE AVG([DISTIN
CT] DOUBLE arg1)
::::AVG WINDOW  DOUBLE AVG([DISTINCT] DOUBLE arg0),DECIMAL(?,?) AVG([DISTINCT] DECIMAL(?,
?) arg0)
::::BASE64      SCALAR  STRING BASE64(BINARY arg0)
::::BIN SCALAR  STRING BIN(BIGINT arg0)
::::BITAND      SCALAR  BIGINT BITAND(BIGINT arg0, BIGINT arg1)
::::BITNOT      SCALAR  BIGINT BITNOT(BIGINT arg0)
::::BITOR       SCALAR  BIGINT BITOR(BIGINT arg0, BIGINT arg1)
::::BITXOR      SCALAR  BIGINT BITXOR(BIGINT arg0, BIGINT arg1)
::::BROUND      SCALAR  DOUBLE BROUND(DOUBLE arg0, BIGINT arg1),DOUBLE BROUND(DOUBLE arg0
)
::::CASE        SCALAR  null
......
```

2.12. Tunnel operations

MaxCompute allows you to run Tunnel commands to upload and download data. This topic describes how to upload and download data by using Tunnel commands.

For more information about Tunnel operations, see Tunnel commands. The following table describes common commands that are used for Tunnel operations.

Operation	Description	Role	Operation platform
Upload data	Uploads local data to a MaxCompute table in append mode.	Users who have the ALTER permission on tables	You must run the commands described in this topic on the MaxCompute client.
Download data	Downloads MaxCompute table data or the execution results of a specified instance to a directory on your on-premises machine.	Users who have the SELECT permissions to read table data	

Upload data

Uploads local data to a MaxCompute table in append mode. You are not charged when you upload data to MaxCompute.

- Limits
  - You can upload files or level-1 directories to only one table or one partition in a table each time.
  - For a partitioned table, you must specify the partition to which you want to upload data. If the table has multiple levels of partitions, you must specify a lowest-level partition.
- Syntax

```
Tunnel upload <path> [<project_name>.]<table_name>[/<pt_spc>];
```

- Parameters
  - path: required. The directory and name of the data file that you want to upload from your on-premises machine. The default format of data files is TXT.

    You can save data files in the `bin` directory of the MaxCompute client. In this case, you must set path to a value in the `File name.File name extension` format. You can also save data files to another directory, such as the test folder in drive D. In this case, you must set path to a value in the `D:\test\File name.File name extension` format.
  - project_name: optional. The name of the project to which the destination table belongs. This parameter is required when you access tables across projects.
  - table_name: required. The name of the destination table.
  - pt_spc: optional. The partitions that are specified in a partitioned table. You must specify a lowest-level partition. The value of this parameter is in the `partition_col1=col1_value1, partition_col2=col2_value1...` format.
- Examples

- Example 1: Upload data in the log.txt file to the test_table table of the current project. The log.txt file is saved in the `bin` folder of the MaxCompute client. Sample statements:

```
Tunnel upload log.txt test_table;
```

- Example 2: Upload data in the log.txt file to the `p1="b1",p2="b2"` partitions of the test_table table in the test_project project. The table has two levels of partitions. The log.txt file is saved in the test folder on drive D. Sample statements:

```
Tunnel upload D:\test\log.txt test_project.test_table/p1="b1",p2="b2";
```

Download data

Downloads MaxCompute table data or the execution results of a specified instance to a directory on your on-premises machine. You can download MaxCompute data only from the Internet. You are charged based on the size of data that is downloaded. The fee is calculated by using the following formula: Fee per download = Amount of data downloaded (GB) × Unit price of download (USD 0.1166/GB).

- Limits
  - You can download data from only one table or one partition to a single local file each time.
  - For a partitioned table, you must specify the partition from which you want to download data. If the table has multiple levels of partitions, you must specify a lowest-level partition.
- Syntax

```
Tunnel download [<project_name>.]<table_name>[/<pt_spc>] <path>;
```

- Parameters
  - project_name: optional. The name of the project to which the destination table belongs. This parameter is required when you access tables across projects.
  - table_name: required. The name of the destination table.
  - pt_spc: optional. The partitions that are specified in a partitioned table. You must specify a lowest-level partition. The value of this parameter is in the `partition_col1=col1_value1, partition_col2=col2_value1...` format.
  - path: required. The directory and name of the downloaded data file. The default format of the downloaded data file is TXT.

    You can save data files in the `bin` directory of the MaxCompute client. In this case, you must set path to a value in the `File name.File name extension` format. You can also save data files to another directory, such as the test folder in drive D. In this case, you must set path to a value in the `D:\test\File name.File name extension` format.
- Examples
  - Example 1: Download data from the second-level partitioned table test_project.test_table to the test_table.txt file. The test_table.txt file is saved in the `bin` folder of the MaxCompute client. Sample statements:

```
Tunnel download test_project.test_table/p1="b1",p2="b2" test_table.txt;
```

- Example 2: Download data from the second-level partitioned table test_project.test_table to the test_table.txt file. The test_table.txt file is saved in the test folder on drive D. Sample statements:

```
Tunnel download test_project.test_table/p1="b1",p2="b2" D:\test\test_table.txt;
```

2.13. Time zone configuration operations

This topic describes how to use a SET command to configure the time zone for a MaxCompute project.

The following types of jobs support the time zone configuration feature:

- MapReduce
- Spark on MaxCompute
  - If tasks are submitted to the MaxCompute computing cluster, the time zone of the project is automatically obtained.
  - If tasks are submitted from spark-shell, spark-sql, or pyspark in yarn-client mode, you must configure parameters in the *spark-defaults.conf* file of the driver and add `spark.driver.extraJavaOptions -Duser.timezone=America/Los_Angeles`. The timezone parameter indicates the time zone that you want to use.
- Machine Learning Platform for AI (PAI)
- Graph

Configuration methods

By default, the time zone of a MaxCompute project is UTC+8. The DATETIME, TIMESTAMP, and DATE fields and the related built-in time functions are all calculated based on UTC+8. You can use one of the following methods to configure the time zone:

- Session level: Submit the `set odps.sql.timezone=<timezoneid>;` statement along with a computing statement for execution.

```
--Set the time zone to Asia/Tokyo.
SET odps.sql.timezone=Asia/Tokyo;
--Query the current time zone.
SELECT getdate();
output:
+------------+
| _c0        |
+------------+
| 2018-10-30 23:49:50 |
+------------+
```

- Project level: Execute the `setProject odps.sql.timezone=<timezoneid>;` statement. Only the project owner has the permission to execute this statement.

> **Notice** After the time zone of a project is configured, it is used for all time computing, and the data of existing jobs is affected. Therefore, configure the time zone only when it is necessary. We recommend that you configure time zones only for new projects.

Limits and usage notes

- SQL built-in date functions, user-defined functions (UDFs), user-defined types (UDTs), user-defined joins (UDJs), and the `SELECT TRANSFORM` statement allow you to obtain the timezone attribute of a project to configure the time zone.
- A time zone must be configured in the format such as `Asia/Shanghai`, which supports daylight saving time. Do not configure the time zone in the GMT+9 format.
- If the time zone in SDK differs from that of the project, you must configure the GMT time zone to convert the data type from DATETIME to STRING.
- After the time zone is configured, differences exist between the real time and the output time of related SQL statements that you run in DataWorks. Between the years of 1900 and 1928, the time difference is 352 seconds. Before the year of 1900, the time difference is 9 seconds.
- MaxCompute, SDK for Java, and the related client are updated to ensure that DATETIME data stored in MaxCompute is correct across multiple time zones. The versions of the required SDK for Java and related client have the *-oversea* suffix. The update may affect the display of DATETIME data that is earlier than January 1, 1928 in MaxCompute.
- If the local time zone is not UTC+8 when you update MaxCompute, we recommend that you update SDK for Java and the related client. This ensures that the SQL-based computing results and data transferred by using Tunnel commands that are later than January 1, 1900 are accurate and consistent. After the update, for the DATETIME data that is earlier than January 1, 1900, the SQL-based computing results and data transferred by using Tunnel commands may still have a difference of 343 seconds. For DATETIME data that is earlier than January 1, 1928 and uploaded before SDK for Java and the related client are updated, the time in the new versions is 352 seconds earlier.
- If you do not update SDK for Java and the client to versions with the *-oversea* suffix, the SQL-based computing results differ from data that is transferred by using Tunnel commands. For data that is earlier than January 1, 1900, the time difference is 9 seconds. For data that is between January 1, 1900 and January 1, 1928, the time difference is 352 seconds.

  > **Note** The modification of the time zone configuration in SDK for Java or on the related client does not affect the time zone configuration in DataWorks. Therefore, the time zones are different. You must evaluate how the scheduled jobs in DataWorks are affected. The time zone of a DataWorks server in the Japan (Tokyo) region is GMT+9, and that in the Singapore (Singapore) region is GMT+8.

- If you use a third-party client that is connected to MaxCompute by using Java Database Connectivity (JDBC), you must configure the time zone on the client to ensure time consistency between the client and the server.

2.14. SHOW commands

This topic describes how to use SHOW commands in different operations.

The following table describes the usage of SHOW commands in different operations.

Operation type	Description
SET operations	View the properties that you configured by using the SET command.
Tunnel operations	View the commands that are used to upload or download data or the logs of the upload or download operations.

Operation type	Description
Security operations	View the security configuration properties of a project.
Package operations	View the packages that are created or installed in a project.
Backup operations	• View the information about tables in a project and tables in the backup state. You can view information such as table names, table IDs, creation time, and deletion time. • View the backup data of a specified table and obtain information about data versions within the retention period. • View the backup data of a deleted table and obtain information about data versions within the retention period. • View the backup data of a specified partition and obtain information about data versions within the retention period. • View the backup data of a deleted partition and obtain information about data versions within the retention period.
Table operations	• View all the partitions of a table. • View all the tables and views in a project or the tables and views that meet specific rules in a project.
Function operations	View the information about all built-in functions in a project.
Instance operations	View the information about an instance. The information includes StartTime (accurate to the second), RunTime (in seconds), Status, InstanceID, Owner, and Query (query statement).
Permission query operations	Query the permissions of users or roles and permissions on objects.
Operations that can be performed to collect information about the optimizer of MaxCompute	Test collection results of column stats metrics.

SET operations

The SHOW FLAGS statement displays the properties that you configured by using the SET statement. Syntax:

```
show flags;
```

> **Note** For more information about SET operations, see SET operations.

Tunnel operations

- Displays historical records.

- Syntax

```
tunnel show history [-n <number>];
```

-n <number>: specifies the number of times that the command is executed.

- Examples

Example 1: Display history records. By default, 500 data records are displayed.

```
tunnel show history;
```

Example 2: Display the commands used in the last five data uploads or downloads.

```
tunnel show history -n 5;
```

- Displays the logs of the last data upload or download.

```
tunnel show log;
```

> **Note** For more information about Tunnel commands, see Tunnel commands.

Security operations

Views the security configuration properties of the current project.

> **Note** For more information about security operations, see Security operations.

Package operations

A package creator views the packages that are created or installed in a MaxCompute project.

- Syntax

```
show packages;
```

- Examples

View the information about the packages that are created or installed in the test_project_a project. Sample command:

```
-- View packages.
show packages;
```

The returned result contains the following information:

- PackageName: the name of the package that is created or installed.
- CreateTime: the time when the package is created.
- SourceProject: the name of the MaxCompute project to which the package belongs.
- InstallTime: the time when the package is installed.
- Status: the status of the package.

```
+-------------+--------------------------+
| PackageName | CreateTime               |
+-------------+--------------------------+
| datashare   | 2021-12-28T18:10:39+0800 |
+-------------+--------------------------+
+-------------+-------------------+--------------------------+--------+
| PackageName | SourceProject     | InstallTime              | Status |
+-------------+-------------------+--------------------------+--------+
| systables   | information_schema | 2020-11-24T14:11:23+0800 | OK     |
+-------------+-------------------+--------------------------+--------+
```

Note For more information about package operations, see Cross-project resource access based on packages.

Backup operations

- View the information about tables in a project and tables in the backup state. You can view information such as table names, table IDs, creation time, and deletion time.

```
show history for tables;
```

- View the backup data of a specified table and obtain information about data versions within the retention period.

```
show history for table <table_name>;
```

- View the backup data of a deleted table and obtain information about data versions within the retention period.

```
show history for table table_name ('id'='xxxx');
```

- View the backup data of a specified partition and obtain information about data versions within the retention period.

```
show history for table table_name partition_spec;
```

- View the backup data of a deleted partition and obtain information about data versions within the retention period. You can obtain the value of id from the ObjectId field in the output of the `show history for table;` command.

```
show history for table table_name PARTITION('id'='xxxx');
```

Note For more information about backup and restoration commands, see Backup and restoration.

Table operations

- View the CREATE TABLE statement.
  - Syntax

```
show create table <table_name>;
```

- Parameters

  table_name: required. The name of the table for which you want to view the CREATE TABLE statement.

- Example

```
-- View the CREATE TABLE statement that is used to create the sale_detail table.
show create table sale_detail;
```

The following result is returned:

```
CREATE TABLE IF NOT EXISTS doc_test_dev.sale_detail(shop_name STRING,customer_id STRING
,total_price DOUBLE) PARTITIONED BY (sale_date STRING,region STRING) STORED AS ALIORC;
```

- View all the tables and views or the tables and views that meet specific rules in a project.
  - Syntax

```
-- Display all the tables and views in a project.
show tables;
-- Display the tables or views whose names contain the chart keyword in a project.
show tables like '<chart>';
```

  - Example

```
-- Display the tables whose names contain the sale* keyword in a project. The asterisk
(*) indicates any character.
show tables like 'sale*';
```

The following result is returned:

```
ALIYUN$account_name:sale_detail
......
-- ALIYUN is a system prompt, which indicates that the table is created by using an Ali
baba Cloud account. If the table was created by a RAM user, the system prompt is RAM.
```

- View all the partitions of a table. If the table does not exist or the table is a non-partitioned table, an error is returned.
  - Syntax

```
show partitions <table_name>;
```

  - Parameters

    table_name: required. The name of the partitioned table whose partition information you want to view.

- Example

```
-- Display all the partitions of the sale_detail table.
show partitions sale_detail;
```

The following result is returned:

```
sale_date=201310/region=beijing
sale_date=201312/region=shenzhen
sale_date=201312/region=xian
sale_date=2014/region=shenzhen
OK
```

> **Note** For more information about table operations, see Table operations.

Function operations

View the information about all built-in functions in a MaxCompute project.

- Syntax

```
show functions;
```

- Example

```
show functions;
```

The following result is returned:

```
ID = 20211214091641326gg0g****
::::ABS SCALAR  INT ABS(INT arg0),DOUBLE ABS(DOUBLE arg0),DECIMAL(?,?) ABS(DECIMAL(?,?) a
rg0),BIGINT ABS(BIGINT arg0)
::::ACOS        SCALAR  DOUBLE ACOS(DOUBLE arg0),DOUBLE ACOS(DECIMAL(?,?) arg0)
::::ADD_MONTHS  SCALAR  STRING ADD_MONTHS(TIMESTAMP arg0, BIGINT arg1),STRING ADD_MONTHS(
STRING arg0, BIGINT arg1),STRING ADD_MONTHS(DATE arg0, BIGINT arg1)
::::ALL_MATCH   SCALAR  BOOLEAN ALL_MATCH(ARRAY<T> arg0, java.util.function.Function<T, j
ava.lang.Boolean> arg1)
::::ANY_MATCH   SCALAR  BOOLEAN ANY_MATCH(ARRAY<T> arg0, java.util.function.Function<T, j
ava.lang.Boolean> arg1)
::::ANY_VALUE   AGGREGATOR      T ANY_VALUE([DISTINCT] T arg1)
::::APPROX_DISTINCT     AGGREGATOR      BIGINT APPROX_DISTINCT([DISTINCT] P arg1, DOUBLE
arg2),BIGINT APPROX_DISTINCT([DISTINCT] P arg1)
::::ARG_MAX     AGGREGATOR      R ARG_MAX([DISTINCT] T arg1, R arg2)
::::ARG_MIN     AGGREGATOR      R ARG_MIN([DISTINCT] T arg1, R arg2)
::::ARRAY       SCALAR  ARRAY<STRING> ARRAY(),ARRAY<T> ARRAY(T arg0...)
::::ARRAYS_OVERLAP      SCALAR  BOOLEAN ARRAYS_OVERLAP(ARRAY<T> arg0, ARRAY<T> arg1)
::::ARRAYS_ZIP  SCALAR  null
::::ARRAY_CONTAINS      SCALAR  BOOLEAN ARRAY_CONTAINS(ARRAY<T> arg0, T arg1)
::::ARRAY_DISTINCT      SCALAR  ARRAY<T> ARRAY_DISTINCT(ARRAY<T> arg0)
::::ARRAY_EXCEPT        SCALAR  ARRAY<T> ARRAY_EXCEPT(ARRAY<T> arg0, ARRAY<T> arg1)
::::ARRAY_INTERSECT     SCALAR  ARRAY<T> ARRAY_INTERSECT(ARRAY<T> arg0, ARRAY<T> arg1)
::::ARRAY_JOIN  SCALAR  STRING ARRAY_JOIN(ARRAY<STRING> arg0, STRING arg1),STRING ARRAY_J
OIN(ARRAY<STRING> arg0, STRING arg1, STRING arg2)
::::ARRAY_MAX   SCALAR  T ARRAY_MAX(ARRAY<T> arg0)
::::ARRAY_MIN   SCALAR  T ARRAY_MIN(ARRAY<T> arg0)
::::ARRAY_POSITION      SCALAR  BIGINT ARRAY_POSITION(ARRAY<T> arg0, T arg1)
::::ARRAY_REDUCE        SCALAR  OUT ARRAY_REDUCE(ARRAY<IN> arg0, BUF arg1, java.util.func
tion.BiFunction<BUF, IN, BUF> arg2, java.util.function.Function<BUF, OUT> arg3)
::::ARRAY_REMOVE        SCALAR  ARRAY<T> ARRAY_REMOVE(ARRAY<T> arg0, T arg1)
::::ARRAY_REPEAT        SCALAR  ARRAY<T> ARRAY_REPEAT(T arg0, BIGINT arg1)
::::ARRAY_SORT  SCALAR  ARRAY<T> ARRAY_SORT(ARRAY<T> arg0, java.util.function.BiFunction<
T, T, java.lang.Long> arg1)
::::ARRAY_UNION SCALAR  ARRAY<T> ARRAY_UNION(ARRAY<T> arg0, ARRAY<T> arg1)
::::ASCII       SCALAR  BIGINT ASCII(STRING arg0)
::::ASIN        SCALAR  DOUBLE ASIN(DOUBLE arg0),DOUBLE ASIN(DECIMAL(?,?) arg0)
::::ATAN        SCALAR  DOUBLE ATAN(DOUBLE arg0),DOUBLE ATAN(DECIMAL(?,?) arg0)
::::AT_MOST_ONE_ROW     AGGREGATOR      T AT_MOST_ONE_ROW([DISTINCT] T arg1)
::::AVG AGGREGATOR      DECIMAL(?,?) AVG([DISTINCT] DECIMAL(?,?) arg1),DOUBLE AVG([DISTIN
CT] DOUBLE arg1)
::::AVG WINDOW  DOUBLE AVG([DISTINCT] DOUBLE arg0),DECIMAL(?,?) AVG([DISTINCT] DECIMAL(?,
?) arg0)
::::BASE64      SCALAR  STRING BASE64(BINARY arg0)
::::BIN SCALAR  STRING BIN(BIGINT arg0)
::::BITAND      SCALAR  BIGINT BITAND(BIGINT arg0, BIGINT arg1)
::::BITNOT      SCALAR  BIGINT BITNOT(BIGINT arg0)
::::BITOR       SCALAR  BIGINT BITOR(BIGINT arg0, BIGINT arg1)
::::BITXOR      SCALAR  BIGINT BITXOR(BIGINT arg0, BIGINT arg1)
::::BROUND      SCALAR  DOUBLE BROUND(DOUBLE arg0, BIGINT arg1),DOUBLE BROUND(DOUBLE arg0
)
::::CASE        SCALAR  null
......
```

> **Note** For more information about function operations, see Function operations.

Instance operations

Views instance information. The information includes StartTime (in seconds), RunTime (in seconds), Status, InstanceID, Owner, and Query statements.

- Syntax

```
show p|proc|processlist|instances [from <startdate>] [to <enddate>] [-p <project_name>] [
-limit <number> | <number>] [-all];
ls|list instances [from <startdate>] [to <enddate>] [-p <project_name>] [-limit <number>
| <number>] [-all];
```

  The following statements are equivalent: `show p`, `show proc`, `show processlist`, `show instances`, `ls instances`, and `list instances`.

- Parameters
  - startdate and enddate: optional. The instance information that is submitted by a user within the period from startdate to enddate is returned. The date specified by the startdate parameter must be earlier than the date specified by the enddate parameter. The instance information that is submitted on the day that is specified by enddate is not included. The values of the two parameters must be in the `yyyy-mm-dd` format and are accurate to the day. If you do not configure the parameters, the instance information that is submitted on the current day is returned.
  - project_name: optional. The name of the MaxCompute project to which the instance that you want to query belongs. You must have the permissions to view the instance of the MaxCompute project. If you do not configure this parameter, the instance of the current MaxCompute project is queried.
  - number: optional. The number of instances that you want to return.
  - If you configure this parameter, information about N instances that is submitted at the time nearest to the current time is returned in chronological order. N is specified by the number parameter. If you do not configure this parameter, information about the instances that meet specific requirements is returned. `-limit <number>` and `number` are equivalent.
  - -all: optional. Information about all instances that are run by the members of the MaxCompute project is returned. If you do not configure this parameter, information about the instances that are run by the current user in the MaxCompute project is returned.

    If the number parameter is not specified, information about up to 50 instances is returned by default. If the number parameter is configured, information about N instances is returned. N is specified by the number parameter.

- Examples

- Example 1: View the information about all instances that are run by the current user in the current MaxCompute project on the current day. Sample statement:

```
show p;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 11:43:04   0s       Success  20210914**************3rw2   ALIYUN$****@test.
aliyunid.com
2021-09-14 11:43:05   1s       Success  20210914**************5t32   ALIYUN$****@test.
aliyunid.com select date_sub(datetime '2005-03-01 00:00:00', 1);
2021-09-14 11:58:13   0s       Success  20210914**************5pr2   ALIYUN$****@test.
aliyunid.com
2021-09-14 11:58:15   1s       Success  20210914**************5qgr   ALIYUN$****@test.
aliyunid.com select date_sub(date '2005-02-28', -1);
2021-09-14 12:02:15   1s       Success  20210914**************h8o7   ALIYUN$****@test.
aliyunid.com select date_sub('2008-03-01 00:00:00', 2);
2021-09-14 12:02:15   0s       Success  20210914**************5t32   ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:31   0s       Success  20210914**************5pr2   ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:32   0s       Success  20210914**************euq2   ALIYUN$****@test.
aliyunid.com select date_sub('2005-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2   ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************j8o7   ALIYUN$****@test.
aliyunid.com select date_sub(getdate(),1);
2021-09-14 13:40:40   1s       Success  20210914**************h3wz   ALIYUN$****@test.
aliyunid.com select date_sub(getdate(),0);
2021-09-14 13:40:40   0s       Success  20210914**************9nm7   ALIYUN$****@test.
aliyunid.com
12 instances
```

- Example 2: View the information about the instances that are run by the current user in the current MaxCompute project within a specified period of time, and specify the number of instances whose information you want to query. Sample statement:

```
show instances from 2021-09-14 to 2021-09-15 -limit 10;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 11:58:13   0s       Success  20210914**************5pr2   ALIYUN$****@test.
aliyunid.com
2021-09-14 11:58:15   1s       Success  20210914**************5qgr   ALIYUN$****@test.
aliyunid.com    select date_sub(date '2005-02-28', -1);
2021-09-14 12:02:15   1s       Success  20210914**************h8o7   ALIYUN$****@test.
aliyunid.com    select date_sub('2008-03-01 00:00:00', 2);
2021-09-14 12:02:15   0s       Success  20210914**************5t32   ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:31   0s       Success  20210914**************5pr2   ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:32   0s       Success  20210914**************euq2   ALIYUN$****@test.
aliyunid.com    select date_sub('2005-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2   ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************j8o7   ALIYUN$****@test.
aliyunid.com    select date_sub(getdate(),1);
2021-09-14 13:40:40   1s       Success  20210914**************h3wz   ALIYUN$****@test.
aliyunid.com    select date_sub(getdate(),0);
2021-09-14 13:40:40   0s       Success  20210914**************9nm7   ALIYUN$****@test.
aliyunid.com
10 instances
```

- Example 3: View the information about the instances that are run by all users in another MaxCompute project within a specified period of time and specify the number of instances whose information you want to query. Sample statement:

```
ls instances from 2021-09-14 to 2021-09-15 -p doc_test_dev -all -limit 10;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 11:59:16   0s       Success  20210914**************6pr3    ALIYUN$****@test.
aliyunid.com
2021-09-14 11:59:20   1s       Success  20210914**************6qgr    ALIYUN$****@test.
aliyunid.com    select date_sub(date '2007-02-26', -1);
2021-09-14 12:02:19   1s       Success  20210914**************h8o7    ALIYUN$****@test.
aliyunid.com    select date_sub('2009-03-01 00:00:00', 2);
2021-09-14 12:02:25   0s       Success  20210914**************7t42    ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:37   0s       Success  20210914**************7pr2    ALIYUN$****@test.
aliyunid.com
2021-09-14 12:02:40   0s       Success  20210914**************emq2    ALIYUN$****@test.
aliyunid.com    select date_sub('2015-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2    ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************68o7    ALIYUN$kiki
select date_sub(getdate(),1);
2021-09-14 13:45:40   1s       Success  20210914**************73wz    ALIYUN$kiki
select date_sub(getdate(),0);
2021-09-14 13:45:45   0s       Success  20210914**************9nm7    ALIYUN$dreak
10 instances
```

- Example 4: View the information about the instances that are run by all users in another MaxCompute project on the current day, and specify the number of instances whose information you want to query. Sample statement:

```
show p -p doc_test_dev -all 5;
```

The following result is returned:

```
StartTime             RunTime  Status   InstanceID                  Owner
Query
2021-09-14 12:02:40   0s       Success  20210914**************emq2    ALIYUN$****@test.
aliyunid.com    select date_sub('2015-03-01 00:00:00', 2);
2021-09-14 13:35:42   0s       Success  20210914**************1ms2    ALIYUN$****@test.
aliyunid.com
2021-09-14 13:35:43   0s       Success  20210914**************68o7    ALIYUN$kiki
select date_sub(getdate(),1);
2021-09-14 13:45:40   1s       Success  20210914**************73wz    ALIYUN$kiki
select date_sub(getdate(),0);
2021-09-14 13:45:45   0s       Success  20210914**************9nm7    ALIYUN$dreak
5 instances
```

> **Note** For more information about instance operations, see Instance operations.

Permission query operations

MaxCompute allows you to query the permissions of a user or a role, or the permissions on an object by using SHOW commands. For more information, see Query permissions by using MaxCompute SQL or Permission-related command set.

Operations that can be performed to collect information about the optimizer of MaxCompute

You can run the `show statistic` command to test collection results of column stats metrics.

- Examples

```
-- Test the collection result of the tinyint1 column.
show statistic analyze2_test columns (tinyint1);
-- Test the collection results of the smallint1, string1, boolean1, and timestamp1 column
s.
show statistic analyze2_test columns (smallint1, string1, boolean1, timestamp1);
-- Test the collection results of all columns.
show statistic analyze2_test columns;
```

- Output

```
-- Collection result of the tinyint1 column:
ID = 20201126085225150gnqo****
tinyint1:MaxValue:      20                  -- The value of max.
tinyint1:DistinctNum:   4.0                 -- The value of ndv.
tinyint1:MinValue:      1                   -- The value of min.
tinyint1:NullNum:       1.0                 -- The value of nNulls.
tinyint1:TopK:  {1=1.0, 10=1.0, 20=1.0}     -- The value of topK. 10=1.0 indicates that
the occurrence frequency of column value 10 is 1. A maximum of 20 values with the highest
frequency of occurrence can be returned.
-- Collection results of the smallint1, string1, boolean1, and timestamp1 columns:
ID = 20201126091636149gxgf****
smallint1:MaxValue:     20
smallint1:DistinctNum:  4.0
smallint1:MinValue:     2
smallint1:NullNum:      1.0
smallint1:TopK:         {2=1.0, 7=1.0, 20=1.0}
string1:MaxLength       6.0                 -- The value of maxColLen.
string1:AvgLength:      3.0                 -- The value of avgColLen.
string1:DistinctNum:    4.0
string1:NullNum:        1.0
string1:TopK:   {str1=1.0, str12=1.0, str123=1.0}
boolean1:DistinctNum:   3.0
boolean1:NullNum:       1.0
boolean1:TopK:  {false=2.0, true=1.0}
timestamp1:DistinctNum:         3.0
timestamp1:NullNum:     1.0
timestamp1:TopK:        {2018-09-17 00:00:00.0=2.0, 2018-09-18 00:00:00.0=1.0}
-- Collection results of all columns:
ID = 20201126092022636gzm1****
tinyint1:MaxValue:      20
tinyint1:DistinctNum:   4.0
tinyint1:MinValue:      1
```

```
tinyint1:NullNum:       1.0
tinyint1:TopK:  {1=1.0, 10=1.0, 20=1.0}
smallint1:MaxValue:     20
smallint1:DistinctNum:  4.0
smallint1:MinValue:     2
smallint1:NullNum:      1.0
smallint1:TopK:         {2=1.0, 7=1.0, 20=1.0}
int1:MaxValue:  7
int1:DistinctNum:       3.0
int1:MinValue:  4
int1:NullNum:   1.0
int1:TopK:      {4=2.0, 7=1.0}
bigint1:MaxValue:       11111118
bigint1:DistinctNum:    4.0
bigint1:MinValue:       8
bigint1:NullNum:        1.0
bigint1:TopK:   {8=1.0, 2222228=1.0, 11111118=1.0}
double1:MaxValue:       123452.3
double1:DistinctNum:    4.0
double1:MinValue:       12.3
double1:NullNum:        1.0
double1:TopK:   {12.3=1.0, 67892.3=1.0, 123452.3=1.0}
decimal1:MaxValue:      22.4
decimal1:DistinctNum:   4.0
decimal1:MinValue:      2.4
decimal1:NullNum:       1.0
decimal1:TopK:  {2.4=1.0, 12.4=1.0, 22.4=1.0}
decimal2:MaxValue:      52.5
decimal2:DistinctNum:   4.0
decimal2:MinValue:      2.57
decimal2:NullNum:       1.0
decimal2:TopK:  {2.57=1.0, 42.5=1.0, 52.5=1.0}
string1:MaxLength       6.0
string1:AvgLength:      3.0
string1:DistinctNum:    4.0
string1:NullNum:        1.0
string1:TopK:   {str1=1.0, str12=1.0, str123=1.0}
varchar1:MaxLength      6.0
varchar1:AvgLength:     3.0
varchar1:DistinctNum:   4.0
varchar1:NullNum:       1.0
varchar1:TopK:  {str2=1.0, str200=1.0, str21=1.0}
boolean1:DistinctNum:   3.0
boolean1:NullNum:       1.0
boolean1:TopK:  {false=2.0, true=1.0}
timestamp1:DistinctNum:         3.0
timestamp1:NullNum:     1.0
timestamp1:TopK:        {2018-09-17 00:00:00.0=2.0, 2018-09-18 00:00:00.0=1.0}
datetime1:DistinctNum:  3.0
datetime1:NullNum:      1.0
datetime1:TopK:         {1537117199000=2.0, 1537030799000=1.0}
```

> **Note** For more information about how to collect metadata for the optimizer of MaxCompute, see Collect information for the optimizer of MaxCompute.

2.15. Other operations

This topic describes other common statements that are used when you use MaxCompute SQL for your business development.

Cost estimation

Estimates the cost of executing an SQL job based on the amount of input data, number of user-defined functions (UDFs), and complexity of the job. If partition pruning is enabled for the UDFs, you cannot estimate the expense in this case because you cannot determine the number of partitions that are scanned.

> **Note** The result is for reference only. You can view your bill for the actual costs.

- Syntax

```
cost sql <sql_sentence>;
```

- Parameter

  sql_sentence: the SQL statement for which you want to estimate the execution cost.

- Example

```
cost sql select * from sale_detail;
```

  Returned results:

```
ID = 20150715113033121xxxxxxx
UDF:0
Complexity:1.0
Input:0 Bytes
```

3.SQL

3.1. Overview of MaxCompute SQL

The MaxCompute SQL syntax is similar to the SQL syntax. The MaxCompute SQL syntax is a subset of the standard ANSI SQL-92 syntax and extends the standard syntax. This topic describes the scenarios of MaxCompute SQL and the tools that are supported by MaxCompute SQL. This topic also provides instructions on how to use MaxCompute SQL.

Scenarios

MaxCompute SQL is suitable for scenarios in which batch jobs are run to compute gigabytes, terabytes, or exabytes of data. After you submit a MaxCompute SQL job, queue scheduling may occur and lasts for tens of seconds to several minutes. In this scenario, you can submit a batch job to process large amounts of data at the same time. We recommend that you do not connect MaxCompute to a foreground business system that needs to handle thousands to tens of thousands of transactions per second.

Instructions

Operation	References	Description
Learn MaxCompute SQL	Differences in the support for SQL statements	Describes the syntax differences between MaxCompute SQL and mainstream databases.
	Reserved words and keywords	Describes reserved words and keywords in MaxCompute SQL statements.
	Data types	Describes the data types that are supported by MaxCompute SQL, data type editions, and differences between the data type editions.
	Type conversions	Describes the type conversions that are supported by MaxCompute SQL.
	Data type mappings	Describes the data type differences between MaxCompute SQL and mainstream databases.
	Escape character	Describes the escape characters in MaxCompute SQL.
	LIKE usage	Describes the characters that are supported by the LIKE operator for character matching in MaxCompute SQL.

Operation	References	Description
	Regular expressions	Describes the regular expression rules that are supported by MaxCompute SQL.
	Operators	Describes the relational operators, arithmetic operators, bitwise operators, and logical operators in MaxCompute.
Use MaxCompute SQL	Limits	Describes the limits of MaxCompute SQL to help you write scripts that meet specific rules.
	DDL statements	Describes the syntax of the DDL statements that are supported by MaxCompute SQL, such as the DDL statements that are used to manage tables, lifecycles, partitions, and columns.
	DML statements	Describes the syntax of the DML statements that are supported by MaxCompute SQL, such as `INSERT INTO` and `INSERT OVERWRITE`.
	DQL statements	Describes the syntax of the `SELECT` statements that are supported by MaxCompute SQL. The SELECT statements are used to query data.
	Enhanced SQL syntax	Describes the syntax of the statements that can be used to improve the readability and execution efficiency of MaxCompute SQL, such as `LOAD` and `UNLOAD`.
	MaxCompute UDF	Describes user-defined functions that are supported by MaxCompute, including user-defined scalar functions (UDFs), user-defined table-valued functions (UDTFs), and user-defined aggregate functions (UDAFs), and provides instructions on how to create these functions.

Operation	References	Description
	MaxCompute UDT	Describes how to directly call the classes and methods of third-party programming languages in SQL, or how to directly use third-party objects to obtain data.
	MaxCompute UDJ	Describes how to perform custom operations across tables or on multiple tables.
	Run MaxCompute SQL in script mode	Describes how to compile SQL scripts in script mode.

Related tools

You can develop MaxCompute SQL jobs by using related tools based on the complexity of the jobs.

- If you want to develop simple jobs, we recommend that you use the MaxCompute client or MaxCompute console (query editor).
- If you want to develop complex jobs, we recommend that you use MaxCompute Studio or the DataWorks console.

3.2. Differences in the support for SQL statements

This topic describes the differences in the support for SQL statements between MaxCompute and Hive, MySQL, Oracle, and SQL Server. In this topic, you can view the SQL statements that MaxCompute does not support.

Support for DDL statements

Statement	MaxCompute	Hive	MySQL	Oracle	SQL Server
CREATE TABLE-PRIMARY KEY	N	N	Y	Y	Y
CREATE TABLE-NOT NULL	Y	N	Y	Y	Y
CREATE TABLE-CLUSTER BY	Y	Y	N	Y	Y
CREATE TABLE-EXTERNAL TABLE	Y (OSS, OTS, TDDL)	Y	N	Y	N
CREATE TABLE-TEMPORARY TABLE	N	Y	Y	Y	Y (with the # prefix)
INDEX-CREATE INDEX	N	Y	Y	Y	Y

Statement	MaxCompute	Hive	MySQL	Oracle	SQL Server
VIRTUAL COLUMN	N	N	N	Y	Y

Support for DML statements

Statement	MaxCompute	Hive	MySQL	Oracle	SQL Server
CTE	Y	Y	Y	Y	Y
SELECT - recursive CTE	N	N	N	Y	Y
SELECT - GROUP BY ROLL UP	Y	Y	Y	Y	Y
SELECT - GROUP BY CUBE	Y	Y	N	Y	Y
SELECT - GROUPING SET	Y	Y	N	Y	Y
SELECT - IMPLICIT JOIN	Y	Y	N	Y	Y
SELECT - PIVOT	N	N	N	Y	Y
SEMI JOIN	Y	Y	Y	N	N
SELEC TRANSFR OM	Y	Y	N	N	N
SELECT - correlated subquery	Y	Y	Y	Y	Y
ORDER BY NULLS FIRST / LAST	N	Y	Y	Y	Y
LATERAL VIEW	Y	Y	N	Y	Y (CROSS APPLY keyword)

Statement	MaxCompute	Hive	MySQL	Oracle	SQL Server
SET OPERATOR-UNION (distinct)	Y	Y	Y	Y	Y
SET OPERATOR-INTERSECT	Y	N	N	Y	Y
SET OPERATOR-MINUS/EXCEPT	Y	N	N	Y	Y (keyword EXCEPT)
INSERT INTO ... VALUES	Y	Y	Y	Y	Y
INSERT INTO (ColumnList)	Y	Y	Y	Y	Y
UPDATE ... WHERE	Y	Y	Y	Y	Y
DELETE ... WHERE	Y	Y	Y	Y	Y
MERGE INTO	Y	Y	N	Y	Y
ANALYTIC-reusable WINDOWING CLAUSE	Y	Y	N	N	N
ANALYTIC-CURRENT ROW	Y	Y	N	Y	Y
ANALYTIC-UNBOUNDED	Y	N	Y	Y	Y
ANALYTIC-RANGE ...	N	Y	N	Y	Y

Statement	MaxCompute	Hive	MySQL	Oracle	SQL Server
WHILE DO	N	N	Y	Y	Y

Support for scripting statements

Statement	MaxCompute	Hive	MySQL	Oracle	SQL Server
TABLE VARIABLE	Y	Y	Y	Y	Y
SCALER VARIABLE	Y	Y	Y	Y	Y
ERROR HANDLING-RAISE ERROR	N	N	Y	Y	Y
ERROR HANDLING-TRY CATCH	N	N	N	Y	Y
FLOW CONTROL-LOOP	N	N	Y	Y	Y
CURSOR	N	N	Y	Y	Y

3.3. MaxCompute SQL limits

This topic describes the limits of MaxCompute SQL statements.

Item	Maximum value/Limit	Category	Description
Table name length	128 bytes	Length	A table or column name can contain only letters, digits, and underscores (_). It must start with a letter. Special characters are not supported.
Comment length	1,024 bytes	Length	A comment is a valid string that cannot exceed 1,024 bytes in length.
Column definitions in a table	1,200	Quantity	A table can contain a maximum of 1,200 column definitions.
Partitions in a table	60,000	Quantity	A table can contain a maximum of 60,000 partitions.

Item	Maximum value/Limit	Category	Description
Partition levels of a table	6	Quantity	A table can contain a maximum of six levels of partitions.
Output display	10,000 rows	Quantity	A SELECT statement can return a maximum of 10,000 rows.
Number of destination tables for `INSERT` operations	256	Quantity	The `MULTI-INSERT` statement allows you to insert data into a maximum of 256 tables at the same time.
`UNION ALL`	256	Quantity	The `UNION ALL` statement allows you to combine a maximum of 256 tables.
`MAPJOIN`	128	Quantity	A `MAPJOIN` hint allows you to join a maximum of 128 small tables.
`MAPJOIN` memory	512 MB	Size	The memory size for all small tables cannot exceed 512 MB when you specify a `MAPJOIN` hint in SQL statements.
`ptinsubq`	1,000 rows	Quantity	A PT IN SUBQUERY statement can generate a maximum of 1,000 rows.
Length of an SQL statement	2 MB	Length	An SQL statement cannot exceed 2 MB in length. This limit is suitable for the scenarios in which you use an SDK to call SQL statements.
Conditions of a `WHERE` clause	256	Quantity	A `WHERE` clause can contain a maximum of 256 conditions.
Length of a column record	8 MB	Length	The maximum length of a column record in a table is 8 MB.
Parameters in an IN clause	1,024	Quantity	This item specifies the maximum number of parameters in an IN clause, such as `IN (1,2,3….,1024)`. If the number of parameters in an `IN` clause is too large, the compilation performance is affected. We recommend that you use a maximum of 1,024 parameters, but this is not a fixed upper limit.

Item	Maximum value/Limit	Category	Description
`jobconf.json`	1 MB	Size	The maximum size of the `jobconf.json` file is 1 MB. If a table contains a large number of partitions, the size of the `jobconf.json` file may exceed 1 MB.
View	Not writable	Operation	A view is not writable and does not support the `INSERT` statements.
Data type and position of a column	Unmodifiable	Operation	The data type and position of a column cannot be modified.
Java user-defined functions (UDFs)	Not allowed to be `abstract` or `static`	Operation	Java UDFs cannot be `abstract` or `static`.
Partitions that can be queried	10,000	Quantity	A maximum of 10,000 partitions can be queried.
SQL execution plans	1 MB	Size	The size of an execution plan that is generated by using MaxCompute SQL statements cannot exceed 1 MB. Otherwise, the error message `FAILED: ODPS-0010000:System internal error - The Size of Plan is too large` is reported.

Note The preceding limits cannot be manually modified.

3.4. DDL SQL

3.4.1. Table operations

In MaxCompute, tables are used to store data. When you develop, analyze, and maintain a data warehouse, you must process table data. This topic describes the operations that you can perform on tables.

The following table describes the statements that are used for operations on tables.

Operation	Description	Role	Operation platform

Operation	Description	Role	Operation platform
Create a table	Creates a non-partitioned table, a partitioned table, an external table, or a clustered table.	Users who have the CREATE TABLE permission on a project	You can perform the operations by using the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Change the owner of a table	Changes the owner of a table.	Project owner	
Modify the comment of a table	Modifies the comment of a table.	Users who have the ALTER permission on tables	
Change the value of LastDataModifiedTime	Changes the value of `LastDataModifiedTime` for a table to the current time.		
Modify the clustering attribute of a table	Adds or removes the clustering attribute to or from a table.		
Rename a table	Renames a table.		
Clear partition data from a non-partitioned table	Clears data from a specified non-partitioned table.		
Drop a table	Drops a partitioned table or a non-partitioned table.	Users who have the DROP permission on tables	
View the information about tables or views	Views the information about MaxCompute internal tables, views, external tables, clustered tables, or transactional tables.	Users who have the DESCRIBE permission to read the metadata of a table	
View partition information	Views the details of partitions in a table.		
View the CREATE TABLE statement	Views the SQL DDL statement that is used to create a specified table.		

Operation	Description	Role	Operation platform
Display tables and views in a project	Displays all the tables and views or the tables and views that meet specific rules, such as regular expressions, in a project.	Users who have the LIST permission on objects in a project	
Display partitions	Displays all the partitions of a table. If the table does not exist or the table is a non-partitioned table, an error is returned.		

Create a table

Creates a non-partitioned table, a partitioned table, an external table, or a clustered table.

- Limits
  - A partitioned table can have a maximum of six levels of partitions. For example, if a table uses date columns as partition key columns, the six levels of the partitions are `year/month/week/day/hour/minute`.
  - By default, a table can have a maximum of 60,000 partitions. You can adjust the maximum number of partitions in a table based on your business requirements.

  For more information about the limits on tables, see MaxCompute SQL limits.
- Syntax

```
-- Create a table.
 create [external] table [if not exists] <table_name>
 [(<col_name> <data_type> [not null] [default <default_value>] [comment <col_comment>], .
..)]
 [comment <table_comment>]
 [partitioned by (<col_name> <data_type> [comment <col_comment>], ...)]
 -- Configure the shuffle and sort properties of a clustered table that you want to creat
e.
 [clustered by | range clustered by (<col_name> [, <col_name>, ...]) [sorted by (<col_nam
e> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> buckets]
 -- Used only for external tables.
 [stored by StorageHandler]
 -- Used only for external tables.
 [with serdeproperties (options)]
 -- Used only for external tables.
 [location <osslocation>]
 -- Set the table to a transactional table. You can later modify or delete the data of th
e transactional table. Transactional tables have specific limits. Create a transactional
table base on your business requirements.
 [tblproperties("transactional"="true")]
 [lifecycle <days>];
-- Create a table based on an existing table and replicate data from the existing table t
o the new table. Partition properties are not replicated.
create table [if not exists] <table_name> [lifecycle <days>] as <select_statement>;
-- Create a table that has the same schema as an existing table. Data in the existing tab
le is not replicated.
create table [if not exists] <table_name> like <existing_table_name> [lifecycle <days>];
```

- Parameters
  - external: optional. This parameter specifies that the table you want to create is an external table.
  - if not exists: optional. If you create a table by using the name of an existing table but do not specify the if not exists parameter, an error is returned. If you create a table by using the name of an existing table and specify the if not exists parameter, a success message is returned even if the schema of the existing table is different from the schema of the table that you want to create. If you create a table by using the name of an existing table, the table is not created and the metadata of the existing table is not changed.
  - table_name: required. The name of the table. The name must be 1 to 128 bytes in length, and can contain letters, digits, and underscores (_). The name must start with a letter and cannot contain special characters. The name is not case-sensitive. If the value of this parameter does not meet the requirements, an error is returned.
  - col_name: optional. The name of a table column. The name must be 1 to 128 bytes in length, and can contain letters, digits, and underscores (_). The name must start with a letter and cannot contain special characters. The name is not case-sensitive. If the value of this parameter does not meet the requirements, an error is returned.
  - col_comment: optional. The comment of a column. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.
  - data_type: optional. The data type of a column. The following data types are supported: BIGINT, DOUBLE, BOOLEAN, DATETIME, DECIMAL, and STRING. For more information about data types, see Data type editions.

- not null: optional. If you specify this parameter for a column, the values of the column cannot be NULL. For more information about how to modify the parameter, see Change the non-nullable property of a non-partition key column in a table.
- default_value: optional. The default value of the specified column. If a column is not specified in an `INSERT` operation, the default value is used for the column.
- table_comment: optional. The comment of a table. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.
- partitioned by (<col_name> <data_type> [comment <col_comment>], ...: optional. The partition fields of a partitioned table.
  - col_name: the name of a partition key column. The name must be 1 to 128 bytes in length, and can contain letters, digits, and underscores (_). The name must start with a letter and cannot contain special characters. The name is not case-sensitive. If the value of this parameter does not meet the requirements, an error is returned.
  - data_type: the data type of a partition key column. In the MaxCompute V1.0 data type edition, partition key columns must be of the STRING type. In the MaxCompute V2.0 data type edition, partition key columns can be of the TINYINT, SMALLINT, INT, BIGINT, VARCHAR, or STRING type. For more information, see Data type editions. If you use a partition field to partition a table, a full table scan is not required when you add partitions, update partition data, or read partition data. This improves the efficiency of data processing.
  - col_comment: the comment of a partition key column. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.

    > **Note** The value of a partition key column cannot contain double-byte characters, such as Chinese characters. It must start with a letter and can contain letters, digits, and supported special characters. It must be 1 to 128 bytes in length. The following special characters are supported: spaces, colons (:), underscores (_), dollar signs ($), number signs (#), periods (.), exclamation points (!), and at signs (@). The behavior of other characters is not defined, such as escaped characters `\t`, `\n`, and `/`.

- clustered by | range clustered by (<col_name> [, <col_name>, ...]) [sorted by (<col_name> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> buckets: optional. The shuffle and sort properties of the clustered table that you want to create.

  Clustered tables are classified into hash-clustered tables and range-clustered tables.

- Hash-clustered tables
  - CLUSTERED BY: the hash key. MaxCompute performs a hash operation on specified columns and distributes data to each bucket based on the hash values. To prevent data skew and hot spots and to improve the efficiency of concurrent executions, we recommend that you specify columns with a large value range and a small number of duplicate key values in `CLUSTERED BY`. To optimize the performance of `JOIN` operations, we recommend that you select commonly used join or aggregate keys. Join and aggregate keys are similar to primary keys in conventional databases.
  - SORTED BY: the sequence of fields in a bucket. To improve performance, we recommend that you keep the configuration of the SORTED BY clause consistent with that of the CLUSTERED BY clause. After you specify fields in the SORTED BY clause, MaxCompute automatically generates indexes, which can be used to accelerate data queries.
  - number_of_buckets: the number of hash buckets. This parameter is required and the value of this parameter varies based on the amount of data. By default, MaxCompute supports a maximum of 1,111 reducers. This means that MaxCompute supports a maximum of 1,111 hash buckets. You can run the `set odps.sql.reducer.instances=xxx;` command to increase the maximum number of hash buckets. However, the maximum number of hash buckets cannot exceed 4,000. Otherwise, performance may be affected.

    To maintain optimal performance, we recommend that you take note of the following rules when you specify the number of hash buckets:

    - Keep the size of each hash bucket around 500 MB. For example, if you want to add 1,000 hash buckets to a partition whose size is 500 GB, the size of each hash bucket is 500 MB on average. If a table contains a large amount of data, you can increase the size of each hash bucket from 500 MB to a size in the range of 2 GB to 3 GB. You can also run the `set odps.sql.reducer.instances=xxx;` command to set the maximum number of hash buckets to a value greater than 1111.
    - To optimize the performance of `JOIN` operations, we recommend that you do not configure the shuffle and sort properties for hash-clustered tables. The number of hash buckets of a table must be a multiple of the number of hash buckets of the other table. For example, one table has 256 hash buckets and the other table has 512 hash buckets. We recommend that you set the number of hash buckets to 2^n, such as 512, 1024, 2048, and 4096. This way, MaxCompute can automatically split and merge hash buckets. To improve execution efficiency, you can also skip the step of configuring the shuffle and sort properties for hash-clustered tables.

- Range-clustered tables
  - RANGE CLUSTERED BY: the range-clustered columns. MaxCompute performs the bucket operation on the specified columns and distributes data to each bucket based on the bucket ID.
  - SORTED BY: the sequence of fields in a bucket. You can use this parameter in the same way as you use it for a hash-clustered table.
  - number_of_buckets: the number of hash buckets. Compared with hash-clustered tables, range-clustered tables have no limits on the number of buckets when data is evenly distributed. If you do not specify the number of buckets in a range-clustered table, MaxCompute automatically determines the optimal number based on the amount of data.
  - If JOIN and AGGREGATE operations are performed on range-clustered tables and the join key or group key is the range-clustered key or the prefix of the range-clustered key, you can manage flags to disable shuffling. This improves execution efficiency. To control shuffling, you can set `odps.optimizer.enable.range.partial.repartitioning` to true or false. By default, this parameter is set to false, which indicates that shuffling is disabled.

    > **Note**
    > - Clustered tables help optimize the following aspects:
    >   - Bucket pruning
    >   - Aggregation
    >   - Storage
    > - Limits on clustered tables:
    >   - The `INSERT INTO` statement is not supported. You can add data only by using the `INSERT OVERWRITE` statement.
    >   - The data that is imported by using Tunnel commands is not arranged in order. Therefore, you cannot import data into a range-clustered table by using Tunnel commands.

- stored by StorageHandler: optional. This parameter specifies StorageHandler based on the data format of the external table.
- with serdeproperties (options): optional. The parameters related to the authorization, compression, and character parsing of the external table.
- osslocation: optional. The Object Storage Service (OSS) bucket where the data of the external table is stored. For more information, see Create an OSS external table and Use a custom extractor to access OSS.

- tblproperties("transactional"="true"): optional. -- Set the table to a transactional table. You can later perform the `UPDATE` or `DELETE` operation on the transactional table to update or delete data by rows. For more information, see UPDATE and DELETE.

  A transactional table has the following limits:

  - MaxCompute allows you to set a table to a transactional table only when you create the table. If you execute the `ALTER TABLE` statement to change an existing table to a transactional table, an error is returned.

    ```
    alter table not_txn_tbl set tblproperties("transactional"="true");
    -- The following error is returned:
    FAILED: Catalog Service Failed, ErrorCode: 151, Error Message: Set transactional is n
    ot supported
    ```

  - When you create a clustered table or an external table, you cannot set the clustered table or external table to a transactional table.
  - You cannot convert between transactional tables and MaxCompute internal tables, external tables, or clustered tables.
  - Jobs from other systems, such as MaxCompute Spark, Machine Learning Platform for AI, and Graph, cannot access transactional tables.
  - `CLONE TABLE` and `MERGE PARTITION` operations are not supported.
  - Before you execute the `UPDATE`, `DELETE`, or `INSERT OVERWRITE` statement for important data in transactional tables, you must execute the `SELECT` and `INSERT` statements to back up the data to other tables.

- lifecycle: optional. The lifecycle of the table. The value must be a positive integer. Unit: days.
  - Non-partitioned tables: If data in a non-partitioned table remains unchanged for the number of days specified by days after the last data update, MaxCompute executes a statement, such as DROP TABLE, to reclaim the table.
  - Partitioned tables: MaxCompute determines whether to reclaim a partition based on the value of LastDataModifiedTime. Unlike non-partitioned tables, a partitioned table is not deleted even if all of its partitions have been reclaimed. You can configure lifecycles for tables, but not for partitions.

- You can use the `create table [if not exists] <table_name> [lifecycle <days>] as <select_statement>;` statement to create a table and replicate data to the table. However, partition properties and the lifecycle attribute of the source table are not replicated to the created table. The partition key columns of the source table are considered common columns in the created table. You can also configure the lifecycle parameter to reclaim the table.
- You can use the `create table [if not exists] <table_name> like <existing_table_name> [lifecycle <days>];` statement to create a table that has the same schema as the source table. However, tables that are created by using this statement do not replicate data or replicate the lifecycle attribute of the source table. You can also configure the lifecycle parameter to reclaim the table.

- Examples
  - Example 1: Create a non-partitioned table named test1.

    ```
    create table test1 (key STRING);
    ```

- Example 2: Create a partitioned table named sale_detail.

```
create table if not exists sale_detail(
 shop_name     STRING,
 customer_id   STRING,
 total_price   DOUBLE)
partitioned by (sale_date STRING, region STRING);
```

- Example 3: Create a table named sale_detail_ctas1, replicate data from the sale_detail table to the sale_detail_ctas1 table, and then configure the lifecycle for the sale_detail_ctas1 table.

```
create table sale_detail_ctas1 lifecycle 10 as select * from sale_detail;
```

You can run the `desc extended sale_detail_ctas1;` command to view table details, such as the schema and lifecycle of a table.

The `sale_detail` table is a partitioned table, but the `sale_detail_ctas1` table that is created by using `create table ... as select_statement ...` does not replicate partition properties. The partition key columns of the source table are considered common columns in the created table. The `sale_detail_ctas1` is a non-partitioned table that has five columns.

- Example 4: Create the sale_detail_ctas2 table and use constants as column values in the `SELECT` clause.

```
-- Column names are specified.
create table sale_detail_ctas2
as
select shop_name, customer_id, total_price, '2013' as sale_date, 'China' as region
from sale_detail;
-- Column names are not specified.
create table sale_detail_ctas3
as
select shop_name, customer_id, total_price, '2013', 'China'
from sale_detail;
```

> **Note** If you use constants as column values in the `SELECT` clause, we recommend that you specify column names. In this example, the names of the fourth and fifth columns in the sale_detail_ctas3 table contain suffixes that are similar to `_c4` and `_c5`.

- Example 5: Create a table named sale_detail_like that uses the same schema as the sale_detail table and configure the lifecycle for the sale_detail_like table.

```
create table sale_detail_like like sale_detail lifecycle 10;
```

You can run the `desc extended sale_detail_like;` command to view table details, such as the schema and lifecycle of a table.

The schema of the sale_detail_like table is the same as that of the sale_detail table. The two tables have the same properties, such as column names, column comments, and table comments, aside from the lifecycle. However, data in the sale_detail table is not replicated to the sale_detail_like table.

- Example 6: Create a table named test_newtype that uses new data types.

```
set odps.sql.type.system.odps2=true;
CREATE TABLE test_newtype (
    c1 TINYINT
    ,c2 SMALLINT
    ,c3 INT
    ,c4 BIGINT
    ,c5 FLOAT
    ,c6 DOUBLE
    ,c7 DECIMAL
    ,c8 BINARY
    ,c9 TIMESTAMP
    ,c10 ARRAY<MAP<BIGINT,BIGINT>>
    ,c11 MAP<STRING,ARRAY<BIGINT>>
    ,c12 STRUCT<s1:STRING,s2:BIGINT>
    ,c13 VARCHAR(20))
LIFECYCLE 1
;
```

- Example 7: Create a hash-clustered table named t1. This table is a non-partitioned table.

```
create table t1 (a STRING, b STRING, c BIGINT) clustered by (c) sorted by (c) into 1024
buckets;
```

- Example 8: Create a hash-clustered table named t2. This table is a partitioned table.

```
create table t2 (a STRING, b STRING, c BIGINT) partitioned by (dt STRING) clustered by
(c) sorted by (c) into 1024 buckets;
```

- Example 9: Create a hash-clustered table named t3. This table is a non-partitioned table.

```
create table t3 (a STRING, b STRING, c BIGINT) range clustered by (c) sorted by (c) int
o 1024 buckets;
```

- Example 10: Create a range-clustered table named t4. This table is a partitioned table.

```
create table t4 (a STRING, b STRING, c BIGINT) partitioned by (dt STRING) range cluster
ed by (c) sorted by (c);
```

- Example 11: Create a transactional table named t5. This table is a non-partitioned table.

```
create table t5(id bigint) tblproperties("transactional"="true");
```

- Example 12: Create a transactional table t6. This table is a partitioned table.

```
create table if not exists t6(id bigint) partitioned by(ds string) tblproperties ("tran
sactional"="true");
```

Change the owner of a table

Changes the owner of a table.

- Syntax

```
alter table <table_name> changeowner to <new_owner>;
```

- Parameters
  - table_name: required. The name of the table whose owner you want to change.
  - new_owner: required. The new owner of the table.
- Examples

```
-- Change the owner of the test1 table to ALIYUN$xxx@aliyun.com.
alter table test1 changeowner to 'ALIYUN$xxx@aliyun.com';
```

Modify the comment of a table

Modifies the comment of a table.

- Syntax

```
alter table <table_name> set comment '<new_comment>';
```

- Parameters
  - table_name: required. The name of the table whose comment you want to modify.
  - new_comment: required. The new comment of the table.
- Example

```
alter table sale_detail set comment 'new coments for table sale_detail';
```

You can execute the `DESC table_name` statement of MaxCompute to view the modification result of the `comment` in the table.

Change the value of LastDataModifiedTime

MaxCompute SQL allows you to execute the `TOUCH` statement to change the value of `LastDataModifiedTime`. You can change the value of `LastDataModifiedTime` to the current time. After you execute this statement to change the value of `LastDataModifiedTime`, MaxCompute determines that the table data has changed, and restarts the lifecycle of the table from the time that is specified by LastDataModifiedTime.

- Syntax

```
alter table <table_name> touch;
```

- Parameters

  table_name: required. The name of the table whose LastDataModifiedTime you want to modify.
- Example

```
alter table sale_detail touch;
```

Modify the clustering attribute of a table

MaxCompute allows you to add or remove the clustering attribute to or from a table by executing the `ALTER TABLE` statement.

- Syntax

- Syntax of the statement that is used to add the hash clustering attribute for a table:

```
alter table <table_name> [clustered by (<col_name> [, <col_name>, ...]) [sorted by (<co
l_name> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> buckets
];
```

- Syntax of the statement that is used to remove the hash clustering attribute from a table:

```
alter table <table_name> not clustered;
```

- If you do not specify the number of buckets in a range-clustered table, MaxCompute automatically determines the optimal number based on the amount of data. Syntax:

```
alter table <table_name> [range clustered by (<col_name> [, <col_name>, ...]) [sorted b
y (<col_name> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> b
uckets];
```

- Syntax of the statement that is used to remove the range clustering attribute from a table or partition:

```
alter table <table_name> not clustered;
alter table <table_name> <pt_spec> not clustered;
```

> **Note**
>
> - The `ALTER TABLE` statement can modify the clustering attribute only for a partitioned table. The clustering attribute of a non-partitioned table cannot be modified after the table is created. The `ALTER TABLE` statement is suitable for existing tables. After you specify the clustering attribute, new partitions are stored based on the clustering attribute that you specified.
> - `ALTER TABLE` takes effect only on the new partitions in a partitioned table. The new partitions include those generated by using `INSERT OVERWRITE` and are stored based on the new clustering attribute. The clustering attribute and storage method remain unchanged for the original partitions. After you specify the clustering properties for a table, you can remove the clustering properties and add clustering properties for the table again. You can specify different clustering columns, sort columns, and numbers of buckets for new partitions.
> - `ALTER TABLE` takes effect only on the new partitions. Therefore, this statement cannot be used to specify partitions.

- Parameters

  For more information, see Create a table.

Rename a table

Renames a table. After you rename a table, only the name of the table is changed. Data in the table is not changed.

- Syntax

```
alter table <table_name> rename to <new_table_name>;
```

- Parameters

- table_name: required. The name of the table that you want to rename.
- new_table_name: required. The new name of the table. If the name specified by the new_table_name parameter exists, an error is returned.

- Example

```
alter table sale_detail rename to sale_detail_rename;
```

Clear partition data from a non-partitioned table

Clears partition data from a specified non-partitioned table. For more information about how to clear data from one or more partitions in a partitioned table, see Clear data from a partition.

- Syntax

```
truncate table <table_name>;
```

- Parameters

  table_name: required. The name of the non-partitioned table whose partition data you want to clear.

Drop a table

Drops a non-partitioned table or partitioned table.

- Usage notes
  - Before you drop a table, confirm that the table can be dropped. Proceed with caution. If you accidentally drop a table, you can restore the table if the backup and restoration feature is enabled for the project and the table is dropped within the backup data retention period specified for the project. For more information about the backup and restoration feature, see Backup and restoration.
  - After you drop a table, the volume of stored data in a MaxCompute project decreases.

- Syntax

```
drop table [if exists] <table_name>;
```

- Parameters
  - if exists: optional. If you do not specify the if exists parameter and the table that you want to drop does not exist, an error is returned. If you specify the if exists parameter, a success message is returned regardless of whether the table exists.
  - table_name: required. The name of the table that you want to drop.

- Example

```
-- Drop the sale_detail table. A success message is returned regardless of whether the sa
le_detail table exists.
drop table if exists sale_detail;
```

View the information about tables or views

Views the information about MaxCompute internal tables, views, external tables, clustered tables, or transactional tables. For more information about how to view detailed table information, see SELECT syntax.

- Syntax

```
-- View the information about a table or view.
desc <table_name|view_name> [partition (<pt_spec>)];
-- View the information about an external table, a clustered table, or a transactional ta
ble. You can also execute this statement to view extended information about an internal t
able.
desc extended <table_name>;
```

- Parameters
  -
  -
  - pt_spec: optional. The partition in the partitioned table that you want to view. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
  - extended: This parameter is required if the table is an external table, a clustered table, or a transactional table. This parameter is used to query extended information about a table. You can also use this parameter to view extended information about an internal table, such as whether a column of the internal table can contain NULL values.

- Examples
  - Example 1: View the information about the test1 table.

```
desc test1;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$maoXXX@alibaba-inc.com | Project: $project_name                      |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2020-11-16 17:47:48                                      |
| LastDDLTime:              2020-11-16 17:47:48                                      |
| LastModifiedTime:         2020-11-16 17:47:48                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                     |
+------------------------------------------------------------------------------------+
| key             | string     |       |                                             |
+------------------------------------------------------------------------------------+
```

- Example 2: View the information about the sale_detail table.

```
desc sale_detail;
```

The following result is returned:

```
+------------------------------------------------------------------------+
| Owner: ALIYUN$maoXXX@alibaba-inc.com | Project: $project_name          |
| TableComment:                                                          |
+------------------------------------------------------------------------+
| CreateTime:                2017-06-28 15:05:17                         |
| LastDDLTime:               2017-06-28 15:05:17                         |
| LastModifiedTime:          2017-06-28 15:05:17                         |
+------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                      |
+------------------------------------------------------------------------+
| Native Columns:                                                        |
+------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                         |
+------------------------------------------------------------------------+
| shop_name       | string     |       |                                 |
| customer_id     | string     |       |                                 |
| total_price     | double     |       |                                 |
+------------------------------------------------------------------------+
| Partition Columns:                                                     |
+------------------------------------------------------------------------+
| sale_date       | string     |                                         |
| region          | string     |                                         |
+------------------------------------------------------------------------+
```

- Example 3: View the detailed information about the sale_detail_ctas1 table.

```
desc extended sale_detail_ctas1;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$maoXXX@alibaba-inc.com | Project: $project_name                      |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2021-07-07 15:29:53                                      |
| LastDDLTime:              2021-07-07 15:29:53                                      |
| LastModifiedTime:         2021-07-07 15:29:53                                      |
| Lifecycle:                10                                                       |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| shop_name | string |       |               | true     | NULL         |              |
| customer_id | string |       |               | true     | NULL         |
|
| total_price | double |       |               | true     | NULL         |
|
| sale_date | string |       |               | true     | NULL         |              |
| region    | string |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  98cb8a38733c49eabed4735173818147                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
+------------------------------------------------------------------------------------+
```

The sale_date and region columns are considered as common columns. They are not partition key columns.

- Example 4: View the information about the sale_detail_ctas2 table.

```
desc sale_detail_ctas2;
```

The following result is returned:

```
+------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name           |
| TableComment:                                                          |
+------------------------------------------------------------------------+
| CreateTime:               2017-06-28 15:42:17                          |
| LastDDLTime:              2017-06-28 15:42:17                          |
| LastModifiedTime:         2017-06-28 15:42:17                          |
+------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                      |
+------------------------------------------------------------------------+
| Native Columns:                                                        |
+------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                         |
+------------------------------------------------------------------------+
| shop_name       | string     |       |                                 |
| customer_id     | string     |       |                                 |
| total_price     | double     |       |                                 |
| sale_date       | string     |       |                                 |
| region          | string     |       |                                 |
+------------------------------------------------------------------------+
```

- Example 5: View the details about the sale_detail_like table.

```
desc extended sale_detail_like;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2021-07-07 15:40:38                                      |
| LastDDLTime:              2021-07-07 15:40:38                                      |
| LastModifiedTime:         2021-07-07 15:40:38                                      |
| Lifecycle:                10                                                       |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| shop_name | string |       |               | true     | NULL         |              |
| customer_id | string |       |               | true     | NULL         |
|
| total_price | double |       |               | true     | NULL         |
|
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| sale_date       | string     |                                                     |
| region          | string     |                                                     |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  61782ff7713f426e9d6f91d5deeac99a                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
+------------------------------------------------------------------------------------+
```

Aside from the lifecycle configuration, the properties, such as field types and partition types, of the sale_detail_like table are the same as those of the sale_detail table.

Note The data size in the output of the `DESC table_name` command includes the data size of the recycle bin. If you want to clear the recycle bin, execute the `PURGE TABLE table_name` statement. Then, execute the `DESC table_name` statement to view the size of data that excludes the size of data in the recycle bin. You can also execute the `SHOW RECYCLE BIN` statement to view the details about data in the recycle bin for the current project.

- Example 6: View the information about the test_newtype table.

```
desc test_newtype;
```

The following result is returned:

```
| Native Columns:                                                                          |
+------------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                          |
+------------------------------------------------------------------------------------------+
| c1              | tinyint    |       |                                                  |
| c2              | smallint   |       |                                                  |
| c3              | int        |       |                                                  |
| c4              | bigint     |       |                                                  |
| c5              | float      |       |                                                  |
| c6              | double     |       |                                                  |
| c7              | decimal    |       |                                                  |
| c8              | binary     |       |                                                  |
| c9              | timestamp  |       |                                                  |
| c10             | array<map<bigint,bigint>> |       |                                  |
| c11             | map<string,array<bigint>> |       |                                  |
| c12             | struct<s1:string,s2:bigint> |       |                                |
| c13             | varchar(20) |       |                                                |
+------------------------------------------------------------------------------------------+
OK
```

- Example 7: View the information of the t1 hash-clustered table. This table is a non-partitioned table. The clustering attribute is displayed in Extended Info.

```
desc extended t1;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2020-11-16 18:00:56                                      |
| LastDDLTime:              2020-11-16 18:00:56                                      |
| LastModifiedTime:         2020-11-16 18:00:56                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| a        | string |       |               | true     | NULL         |              |
| b        | string |       |               | true     | NULL         |              |
| c        | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  e6b06f705dc34a36a5b72e5af486cab7                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
| ClusterType:              hash                                                     |
| BucketNum:                1024                                                     |
| ClusterColumns:           [c]                                                      |
| SortColumns:              [c ASC]                                                  |
+------------------------------------------------------------------------------------+
OK
```

- Example 8: View the information about the t2 hash-clustered table. This table is a partitioned table. The clustering attribute is displayed in Extended Info.

  ```
  desc extended t2;
  ```

  The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime: 2017-12-25 11:18:26                                                    |
| LastDDLTime: 2017-12-25 11:18:26                                                   |
| LastModifiedTime: 2017-12-25 11:18:26                                              |
| Lifecycle: 2                                                                       |
+------------------------------------------------------------------------------------+
| InternalTable: YES | Size: 0                                                       |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field | Type   | Label | Comment                                                   |
+------------------------------------------------------------------------------------+
| a     | string |       |                                                           |
| b     | string |       |                                                           |
| c     | bigint |       |                                                           |
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| dt    | string |                                                                   |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID: 91a3395d3ef64b4d9ee1d2852755                                              |
| IsArchived: false                                                                  |
| PhysicalSize: 0                                                                    |
| FileNum: 0                                                                         |
| ClusterType: hash                                                                  |
| BucketNum: 1024                                                                    |
| ClusterColumns: [c]                                                                |
| SortColumns: [c ASC]                                                               |
+------------------------------------------------------------------------------------+
OK
```

- Example 9: View the information about the t3 range-clustered table. This table is a non-partitioned table. The clustering attribute is displayed in Extended Info.

```
desc extended t3;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2020-11-16 18:01:05                                      |
| LastDDLTime:              2020-11-16 18:01:05                                      |
| LastModifiedTime:         2020-11-16 18:01:05                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| a        | string |       |               | true     | NULL         |              |
| b        | string |       |               | true     | NULL         |              |
| c        | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  38d170aca2684f4baadbbe1931a6ae1f                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
| ClusterType:              range                                                    |
| BucketNum:                1024                                                     |
| ClusterColumns:           [c]                                                      |
| SortColumns:              [c ASC]                                                  |
+------------------------------------------------------------------------------------+
OK
```

- Example 10: View the information about the t4 range-clustered table. This table is a partitioned table. The clustering attribute is displayed in Extended Info.

```
desc extended t4;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@alibaba-inc.com | Project: $project_name                       |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2020-11-16 19:17:48                                      |
| LastDDLTime:              2020-11-16 19:17:48                                      |
| LastModifiedTime:         2020-11-16 19:17:48                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| a        | string |       |               | true     | NULL         |              |
| b        | string |       |               | true     | NULL         |              |
| c        | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| dt              | string     |                                                     |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  6ebc3432e283449188c861427bcd6ee4                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
| StoredAs:                 AliOrc                                                   |
| CompressionStrategy:      normal                                                   |
| ClusterType:              range                                                    |
| BucketNum:                0                                                        |
| ClusterColumns:           [c]                                                      |
| SortColumns:              [c ASC]                                                  |
+------------------------------------------------------------------------------------+
OK
```

- Example 11: Check whether the t5 non-partitioned table is a transactional table.

  > **Note** We recommend that you use the MaxCompute client to check whether a table is a transactional table. The version of the MaxCompute client must be V0.35.4 or later. For more information about how to download and use the MaxCompute client, see MaxCompute client. Other tools may not be updated to display transactional information.

```
desc extended t5;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$xxxxx@aliyun.com | Project: $project_name                            |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:                2021-02-18 10:56:27                                     |
| LastDDLTime:               2021-02-18 10:56:27                                     |
| LastModifiedTime:          2021-02-18 10:56:27                                     |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| id       | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
...
| Transactional:             true                                                    |
+------------------------------------------------------------------------------------+
```

- Example 12: Check whether the t6 partitioned table is a transactional table.

  > **Note** We recommend that you use the MaxCompute client to check whether a table is a transactional table. The version of the MaxCompute client must be V0.35.4 or later. For more information about how to download and use the MaxCompute client, see MaxCompute client. Other tools may not be updated to display transactional information.

  ```
  desc extended t6;
  ```

  The following result is returned:

  ```
  +------------------------------------------------------------------------------------+
  | Owner: ALIYUN$xxxxx@test.aliyunid.com | Project: $project_name                     |
  | TableComment:                                                                      |
  +------------------------------------------------------------------------------------+
  | CreateTime:               2021-02-18 15:34:54                                      |
  | LastDDLTime:              2021-02-18 15:34:54                                      |
  | LastModifiedTime:         2021-02-18 15:34:54                                      |
  +------------------------------------------------------------------------------------+
  | InternalTable: YES      | Size: 0                                                  |
  +------------------------------------------------------------------------------------+
  | Native Columns:                                                                    |
  +------------------------------------------------------------------------------------+
  | Field           | Type        | Label | Comment                                    |
  +------------------------------------------------------------------------------------+
  | id              | bigint      |       |                                            |
  +------------------------------------------------------------------------------------+
  | Partition Columns:                                                                 |
  +------------------------------------------------------------------------------------+
  | ds              | string      |                                                    |
  +------------------------------------------------------------------------------------+
  | Extended Info:                                                                     |
  +------------------------------------------------------------------------------------+
  ...
  | Transactional:            true                                                     |
  +------------------------------------------------------------------------------------+
  ```

View partition information

Views the partition information about a partitioned table.

- Syntax

  ```
  desc <table_name> partition (<pt_spec>);
  ```

- Parameters
  - table_name: required. The name of the partitioned table whose partition information you want to view.
  - pt_spec: required. The information about the partition that you want to view. The value of this parameter is in the `partition_col1=col1_value1, partition_col2=col2_value1...` format. If a table has multi-level partitions, you must specify the values of all the partition key columns.

- Example

```
-- Query partition information about the partitioned table sale_detail.
desc sale_detail partition (sale_date='201310',region='beijing');
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| PartitionSize: 2109112                                                             |
+------------------------------------------------------------------------------------+
| CreateTime:               2015-10-10 08:48:48                                      |
| LastDDLTime:              2015-10-10 08:48:48                                      |
| LastModifiedTime:         2015-10-11 01:33:35                                      |
+------------------------------------------------------------------------------------+
OK
```

View the CREATE TABLE statement

Views the CREATE TABLE statement that is used to create a table. This helps you recreate a schema of the table by using SQL statements.

- Syntax

```
show create table <table_name>;
```

- Parameters

  table_name: required. The name of the table for which you want to view the CREATE TABLE statement.

- Example

```
-- View the CREATE TABLE statement that is used to create the sale_detail table.
show create table sale_detail;
```

The following result is returned:

```
CREATE TABLE IF NOT EXISTS doc_test_dev.sale_detail(shop_name STRING,customer_id STRING,t
otal_price DOUBLE) PARTITIONED BY (sale_date STRING,region STRING) STORED AS ALIORC;
```

Display tables and views in a project

Displays all the tables and views or the tables and views that meet specific rules in a project.

- Syntax

```
-- Display all the tables and views in a project.
show tables;
-- Display the tables or views whose names contain the chart keyword in a project.
show tables like '<chart>';
```

- Example

```
-- Display the tables whose names contain the sale* keyword in a project. The asterisk (*
) indicates any character.
show tables like 'sale*';
```

The following result is returned:

```
ALIYUN$account_name:sale_detail
......
-- ALIYUN is a system prompt, which indicates that the table is created by using an Aliba
ba Cloud account. If the table was created by a RAM user, the system prompt is RAM.
```

Display partitions

Displays all the partitions of a table. If the table does not exist or the table is a non-partitioned table, an error is returned.

- Syntax

```
show partitions <table_name>;
```

- Parameters

  table_name: required. The name of the partitioned table whose partition information you want to view.

- Example

```
-- Display all the partitions of the sale_detail table.
show partitions sale_detail;
```

The following result is returned:

```
sale_date=201310/region=beijing
sale_date=201312/region=shenzhen
sale_date=201312/region=xian
sale_date=2014/region=shenzhen
OK
```

3.4.2. Partition and column operations

This topic describes the operations that you can perform on partitions or columns in a MaxCompute table. You can perform these operations based on your business requirements.

The following table describes the statements that are used for operations on partitions or columns in a MaxCompute table.

Type	Operation	Description	Role	Operation platform
	Add partitions	Adds partitions to an existing partitioned table.		
	Drop partitions	Drops partitions from an existing partitioned table.		
	Change LastDataModifiedTime of a partition	Changes `LastDataModifiedTime` of a partition in a partitioned table.		

Type	Operation	Description	Role	Operation platform
Partition operations	Change a value in a partition key column	Changes a value in a partition key column of a partitioned table.	Users who have the Alter permission on tables	You can execute the statements that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
	Merge partitions	Merges multiple partitions of a partitioned table into one partition. This operation deletes the dimension information about the partitions that are merged and transfers the partition data to a specified partition.		
	Clear data from a partition	Clears data from a specified partition.		
Column operations	Add columns or column comments	Adds columns or column comments to an existing non-partitioned table or partitioned table.		
	Drop columns	Drops columns from an existing non-partitioned table or partitioned table.		
	Change the order of a column	Changes the order of a specified column in a table.		
	Modify the name of a column	Modifies the name of a column in an existing non-partitioned table or partitioned table.		
	Modify the comment of a column	Modifies the comment of a column in an existing non-partitioned table or partitioned table.		
	Change the name and comment of a column	Modifies the name and comment of a column in an existing non-partitioned table or partitioned table at the same time.		

Type	Operation	Description	Role	Operation platform
	Change the non-nullable property of a non-partition key column in a table	Changes the non-nullable property of a non-partition key column in a table.		

Add partitions

Adds partitions to an existing partitioned table.

- Limits
  - A MaxCompute table can have a maximum of 60,000 partitions.
  - To add the values of partition key columns to a table that has multi-level partitions, you must specify all the partitions.
  - This operation can add only the values of partition key columns. The names of partition key columns cannot be added.
- Syntax

```
alter table <table_name> add [if not exists] partition <pt_spec> [partition <pt_spec> par
tition <pt_spec>...];
```

- Parameters
  - table_name: required. The name of the partitioned table to which you want to add partitions.
  - if not exists: optional. If you do not specify the if not exists parameter and a partition with the same name already exists, this operation fails and an error is returned.
  - pt_spec: required. The partitions that you want to add. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col indicates the column name. partition_col_value indicates the column value. The names of partition key columns are not case-sensitive, but their values are case-sensitive.
- Examples
  - Example 1: Add a partition to the sale_detail table. The partition stores the sales records in the China (Hangzhou) region in December 2013.

```
alter table sale_detail add if not exists partition (sale_date='201312', region='hangzh
ou');
```

  - Example 2: Add two partitions to the sale_detail table. The partitions store the sales records in the China (Beijing) and China (Shanghai) regions in December 2013.

```
alter table sale_detail add if not exists partition (sale_date='201312', region='beijin
g') partition (sale_date='201312', region='shanghai');
```

- Example 3: Add one or more partitions to the sale_detail table and specify only the partition key column sale_date for the partitions. An error is returned because you must specify the two partition key columns sale_date and region for the partitions.

```
alter table sale_detail add if not exists partition (sale_date='20111011');
```

Drop partitions

Drops partitions from an existing partitioned table.

MaxCompute allows you to drop partitions based on a specified filter condition. If you want to drop multiple partitions that meet a specified filter condition at a time, you can use an expression to specify the filter condition, use the filter condition to match partitions, and drop the partitions at a time.

- Limits
  - You can specify the information of only one partition key column in a PARTITION (<partition_filtercondition>) clause.
  - If you use an expression to specify PARTITION (<partition_filtercondition>), the function used in the expression must be a built-in scalar function.
- Precautions
  - After you drop a partition from a table in your MaxCompute project, the volume of stored data in your MaxCompute project decreases.
  - You can specify a lifecycle for a partitioned table. This way, MaxCompute automatically reclaims partitions whose data is not updated within the time specified by the lifecycle. For more information about the lifecycle, see Lifecycle.
- Syntax
  - The filter condition is not specified.

```
-- Drop one partition at a time.
alter table <table_name> drop [if exists] partition <pt_spec>;
-- Drop multiple partitions at a time.
alter table <table_name> drop [if exists] partition <pt_spec>,partition <pt_spec>[,part
ition <pt_spec>....];
```

  - The filter condition is specified.

```
alter table <table_name> drop [if exists] partition <partition_filtercondition>;
```

- Parameters
  - table_name: required. The name of the partitioned table from which you want to drop partitions.
  - if exists: optional. If you do not specify the if exists parameter and the partition that you want to drop does not exist, an error is returned.
  - pec: required. The partitions that you want to drop. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col indicates the column name. partition_col_value indicates the column value. The names of partition key columns are not case-sensitive, but their values are case-sensitive.

- partition_filtercondition: the filter condition. This parameter is required when you specify the filter condition. This parameter is not case-sensitive. The parameter is in the following format:

```
partition_filtercondition
    : partition (<partition_col> <relational_operators> <partition_col_value>)
    | partition (scalar(<partition_col>) <relational_operators> <partition_col_value>)
    | partition (<partition_filtercondition1> AND|OR <partition_filtercondition2>)
    | partition (NOT <partition_filtercondition>)
    | partition (<partition_filtercondition1>)[,partition (<partition_filtercondition2>
), ...]
```

  - partition_col: the name of a partition key column.
  - relational_operators: the relational operator. For more information, see Operators.
  - partition_col_value: a value in the partition key column. The value may be a comparison value or regular expression. The data type of this value must be the same as the data type of the partition key column.
  - scalar(): the scalar function. The scalar function generates a scalar based on the input value, processes the values in the column specified by partition_col, and uses relational_operators to compare the processed values with partition_col_value.
  - The filter conditions that are used to drop partitions support the logical operators NOT, AND, and OR. You can use PARTITION (<NOT partition_filtercondition>) to obtain the complementary set of the filter conditions that you specified. You can use PARTITION (<partition_filtercondition1 ANDOR partition_filtercondition2>) to obtain the condition that is used to match the partitions you want to drop.
  - Multiple PARTITION (<partition_filtercondition>) clauses are supported. If these clauses are separated by commas (,), the logical relationship between the clauses is OR. The filter condition is obtained based on the OR logical relationship and used to match the partitions that you want to drop.

- Examples
  - The filter condition is not specified.

```
-- Drop a partition from the sale_detail table. The partition stores the sales record i
n the China (Hangzhou) region in December 2013.
alter table sale_detail drop if exists partition(sale_date='201312',region='hangzhou');

-- Drop two partitions from the sale_detail table. The partitions store the sales recor
ds in the China (Hangzhou) and China (Shanghai) regions in December 2013.
alter table sale_detail drop if exists partition(sale_date='201312',region='hangzhou'),
partition(sale_date='201312',region='shanghai');
```

  - The filter condition is specified.

```
-- Create a partitioned table named sale_detail.
create table if not exists sale_detail(
shop_name     STRING,
customer_id   STRING,
total_price   DOUBLE)
partitioned by (sale_date STRING);
-- Add partitions to the table.
alter table sale_detail add if not exists
partition (sale_date= '201910')
partition (sale_date= '201911')
```

```
partition (sale_date= '201912')
partition (sale_date= '202001')
partition (sale_date= '202002')
partition (sale_date= '202003')
partition (sale_date= '202004')
partition (sale_date= '202005')
partition (sale_date= '202006')
partition (sale_date= '202007');
-- Drop partitions from the table at a time.
alter table sale_detail drop if exists partition(sale_date < '201911');
alter table sale_detail drop if exists partition(sale_date >= '202007');
alter table sale_detail drop if exists partition(sale_date LIKE '20191%');
alter table sale_detail drop if exists partition(sale_date IN ('202002','202004','20200
6'));
alter table sale_detail drop if exists partition(sale_date BETWEEN '202001' AND '202007
');
alter table sale_detail drop if exists partition(substr(sale_date, 1, 4) = '2020');
alter table sale_detail drop if exists partition(sale_date < '201912' OR sale_date >= '
202006');
alter table sale_detail drop if exists partition(sale_date > '201912' AND sale_date <=
'202004');
alter table sale_detail drop if exists partition(NOT sale_date > '202004');
-- Drop partitions by using multiple PARTITION (<partition_filtercondition>) clauses. T
he logical relationship between these clauses is OR.
alter table sale_detail drop if exists partition(sale_date < '201911'), partition(sale_
date >= '202007');
-- Add partitions in other formats.
alter table sale_detail add IF NOT EXISTS
partition (sale_date= '2019-10-05')
partition (sale_date= '2019-10-06')
partition (sale_date= '2019-10-07');
-- Use regular expressions to match the partitions that you want to drop and drop these
partitions at a time.
alter table sale_detail drop if exists partition(sale_date RLIKE '2019-\\d+-\\d+');
-- Create a table named region_sale_detail table. The table contains multi-level partit
ions.
create table if not exists region_sale_detail(
shop_name     STRING,
customer_id   STRING,
total_price   DOUBLE)
partitioned by (sale_date STRING , region STRING );
-- Add partitions to the table.
alter table region_sale_detail add IF NOT EXISTS
partition (sale_date= '201910',region = 'shanghai')
partition (sale_date= '201911',region = 'shanghai')
partition (sale_date= '201912',region = 'shanghai')
partition (sale_date= '202001',region = 'shanghai')
partition (sale_date= '202002',region = 'shanghai')
partition (sale_date= '201910',region = 'beijing')
partition (sale_date= '201911',region = 'beijing')
partition (sale_date= '201912',region = 'beijing')
partition (sale_date= '202001',region = 'beijing')
partition (sale_date= '202002',region = 'beijing');
-- Execute the following statement to drop multi-level partitions at a time. The logica
```

```
l relationship between the two PARTITION (<partition_filtercondition>) clauses is OR. A
fter the statement is executed, the partitions in which the value of the sale_date colu
mn is earlier than 201911 and the partitions in which the value of the region column is
beijing are dropped.
alter table region_sale_detail drop if exists partition(sale_date < '201911'),partition
(region = 'beijing');
-- Execute the following statement to drop a partition in which the value of the sale_d
ate column is earlier than 201911 and the value of the region column is beijing.
alter table region_sale_detail drop if exists partition(sale_date < '201911', region =
'beijing');
```

When you drop multi-level partitions at a time, you cannot specify a filter condition that is based on multiple partition key columns in one `PARTITION partition_filtercondition` clause. Otherwise, the following error is returned: `FAILED: ODPS-0130071:[1,82] Semantic analysis exception - invalid column reference region, partition expression must have one and only one column reference` .

```
-- If you specify the information of multiple partition key columns in a PARTITION (<pa
rtition_filtercondition>) clause, an error is returned. Example of incorrect usage:
alter table region_sale_detail drop if exists partition(sale_date < '201911' AND region
= 'beijing');
```

Change LastDataModifiedTime of a partition

Changes the LastDataModifiedTime of a partition in a partitioned table. MaxCompute SQL allows you to execute the `TOUCH` statement to change `LastDataModifiedTime` of a partition in a partitioned table. This operation changes `LastDataModifiedTime` to the current time. In this case, MaxCompute considers that data is updated, and the new lifecycle of the table or partition starts from the time specified by LastDataModifiedTime.

- Limits

  If a table contains multi-level partitions, you must specify all partition levels when you change LastDataModifiedTime of a partition in the table.

- Syntax

```
alter table <table_name> touch partition (<pt_spec>);
```

- Parameters
  - table_name: required. The name of the partitioned table whose LastDataModifiedTime you want to change. If the table does not exist, an error is returned.
  - pt_spec: required. The partition whose LastDataModifiedTime you want to change. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col indicates the column name. partition_col_value indicates the column value. If the specified column name or column value does not exist, an error is returned.

- Examples

```
-- Change LastDataModifiedTime of the partition in which the value of the sale_date colum
n is 201312 and the value of the region column is shanghai in the sale_detail table.
alter table sale_detail touch partition (sale_date='201312', region='shanghai');
```

Change a value in a partition key column

Changes a value in a partition key column. MaxCompute SQL allows you to execute the `RENAME` statement to change values in partition key columns.

- Limits
  - The RENAME statement can change values in partition key columns but cannot change the column names.
  - If a table contains multi-level partitions, you must specify all partition levels when you change values in partition key columns.
- Syntax

```
alter table <table_name> partition (<pt_spec>) rename to partition (<new_pt_spec>);
```

- Parameters
  - table_name: required. The name of the table in which you want to change column values in a partition.
  - pt_spec: required. The partition in which you want to change column values. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col indicates the column name. partition_col_value indicates the column value. If the specified column name or column value does not exist, an error is returned.
  - new_pt_spec: required. The new partition information. The value of this parameter is in the `(partition_col1 = new_partition_col_value1, partition_col2 = new_partition_col_value2, ...)` format. partition_col indicates the column name. new_partition_col_value indicates the new column value.
- Examples

```
-- Change column values in a partition of the sale_detail table.
alter table sale_detail partition (sale_date = '201312', region = 'hangzhou') rename to p
artition (sale_date = '201310', region = 'beijing');
```

Merge partitions

Merges multiple partitions of a partitioned table into one partition. MaxCompute SQL allows you to execute the `MERGE PARTITION` statement to merge multiple partitions of a table into one partition. This operation deletes the dimension information about the merged partitions and migrates the partition data to a specified partition.

- Limits
  - Partitions of external tables cannot be merged. After partitions of a clustered table are merged into one partition, the partition does not have the clustering attribute.
  - You can merge a maximum of 4,000 partitions at a time.
- Syntax

```
alter table <table_name> merge [if exists] partition (<predicate>) [, partition(<predicat
e2>) ...] overwrite partition (<fullpartitionSpec>) [purge];
```

- Parameters
  - table_name: required. The name of the partitioned table whose partitions you want to merge.

- if exists: optional. If if exists is not specified and the partitions that you want to merge do not exist, an error is returned when you merge partitions. If if exists is specified but no partitions meet the `merge` condition, a new partition cannot be generated after you merge partitions. If you merge partitions and modify source data by using a statement at the same time, an error is returned even if you specify if exists. The statement may be `INSERT`, `RENAME`, or `DROP`.
- predicate: required. The condition that is used to match the partitions that you want to merge.
- fullpartitionSpec: required. The partition that is generated after you merge partitions.
- purge: optional. If you specify this parameter, logs in the session directory are deleted. By default, logs that are generated in the last three days are deleted. For more information, see Purge.

- Examples
  - Example 1: Merge the partitions that meet a specified condition into the destination partition.

```
-- View the partitions in a partitioned table.
show partitions intpstringstringstring;
ds=20181101/hh=00/mm=00
ds=20181101/hh=00/mm=10
ds=20181101/hh=10/mm=00
ds=20181101/hh=10/mm=10
-- Merge all partitions that meet the hh='00' condition into the ds='20181101', hh='00'
, mm='00' partition.
alter table intpstringstringstring merge partition(hh='00') overwrite partition(ds='201
81101', hh='00', mm='00');
-- View the partition that is generated after you merge partitions.
show partitions intpstringstringstring;
ds=20181101/hh=00/mm=00
ds=20181101/hh=10/mm=00
ds=20181101/hh=10/mm=10
```

  - Example 2: Merge specified partitions into the destination partition.

```
-- Merge specified partitions into the destination partition.
alter table intpstringstringstring merge if exists partition(ds='20181101', hh='00', mm
='00'), partition(ds='20181101', hh='10', mm='00'),  partition(ds='20181101', hh='10',
mm='10') overwrite partition(ds='20181101', hh='00', mm='00') purge;
-- View the partition that is generated in the partitioned table after you merge partit
ions.
show partitions intpstringstringstring;
ds=20181101/hh=00/mm=00
```

Clear data from a partition

Clears the data from a specified partition of a partitioned table.

MaxCompute allows you to clear the data in partitions that meet a specified filter condition. If you want to drop one or more partitions that meet a filter condition at a time, you can use an expression to specify the condition, use the condition to match partitions, and then clear the partition data.

- Syntax
  - The filter condition is not specified.

```
truncate table <table_name> partition <pt_spec>[, partition <pt_spec>....];
```

- The filter condition is specified.

```
truncate table <table_name> partition <partition_filtercondition>;
```

- Parameters
  - table_name: required. The name of the partitioned table of which you want to clear partition data.
  - pt_spec: required. The partition in which you want to clear data. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col indicates the column name. partition_col_value indicates the column value. The names of partition key columns are not case-sensitive, but their values are case-sensitive.
  - partition_filtercondition: the filter condition. This parameter is required when you specify the filter condition. This parameter is not case-sensitive. The parameter is in the following format:

```
partition_filtercondition
    : partition (<partition_col> <relational_operators> <partition_col_value>)
    | partition (scalar(<partition_col>) <relational_operators> <partition_col_value>)
    | partition (<partition_filtercondition1> AND|OR <partition_filtercondition2>)
    | partition (NOT <partition_filtercondition>)
    | partition (<partition_filtercondition1>)[,partition (<partition_filtercondition2>
), ...]
```

    - partition_col: the name of a partition key column.
    - relational_operators: the relational operator. For more information, see Operators.
    - partition_col_value: a value in the partition key column. The value may be a comparison value or regular expression. The data type of this value must be the same as the data type of the partition key column.
    - scalar(): the scalar function. The scalar function generates a scalar based on the input value, processes the values in the column specified by partition_col, and uses relational_operators to compare the processed values with partition_col_value.
    - The filter conditions that are used to drop partitions support the logical operators NOT, AND, and OR. You can use PARTITION (<NOT partition_filtercondition>) to obtain the complementary set of the filter conditions that you specified. You can use PARTITION (<partition_filtercondition1 ANDOR partition_filtercondition2>) to obtain the condition that is used to match the partitions you want to drop.
    - Multiple PARTITION (<partition_filtercondition>) clauses are supported. If these clauses are separated by commas (,), the logical relationship between the clauses is OR. The filter condition is obtained based on the OR logical relationship and used to match the partitions that you want to drop.

- Examples
  - The filter condition is not specified.

```
-- Clear the data from a partition of the sale_detail table. The partition stores the s
ales record in the China (Hangzhou) region in December 2013.
truncate table sale_detail partition(sale_date='201312',region='hangzhou');
-- Clear the data from two partitions of the sale_detail table. The partitions store th
e sales records in the China (Hangzhou) and China (Shanghai) regions in December 2013.
truncate table sale_detail partition(sale_date='201312',region='hangzhou'),  partition(
sale_date='201312',region='shanghai');
```

- The filter condition is specified.

```
-- Clear the data from multiple partitions of the sale_detail table. The partitions sto
re the sales records in the China (Hangzhou) region and sale_date of the sales records
starts with 2013.
truncate table sale_detail partition(sale_date like '2013%' and region='hangzhou');
```

Add columns or column comments

Adds columns or column comments to an existing non-partitioned table or partitioned table.

> **Note** MaxCompute allows you to add columns of the STRUCT type, such as `struct<x: string, y: bigint>` and `map<string, struct<x: double, y: double>>`. This feature is in trial use.

- Syntax

```
alter table <table_name> add columns (<col_name1> <type1> comment ['<col_comment>'][, <co
l_name2> <type2> comment '<col_comment>'...]);
```

- Parameters
  - table_name: required. The name of the table to which you want to add columns. You cannot specify the placement of a new column in the table. By default, the new column is added as the last column.
  - col_name: required. The name of the column that you want to add to the table.
  - type: required. The data type of the column that you want to add to the table.
  - col_comment: optional. The comment of the column that you want to add to the table.
- Examples
  - Example 1: Add two columns to the sale_detail table.

```
alter table sale_detail add columns (customer_name STRING, education BIGINT);
```

  - Example 2: Add two columns and their comments to the sale_detail table.

```
alter table sale_detail add columns (customer_name STRING comment 'Customer', education
BIGINT comment 'Education' );
```

  - Example 3: Add a column of a complex data type to the sale_detail table.

```
alter table sale_detail add columns (region struct<province:string, area:string>);
```

Drop columns

Drops one or more specified columns from an existing non-partitioned table or partitioned table.

> **Note** This feature is in trial use.
>
> In the following scenarios, if you drop a column from a table, the read and write operations on the table are affected.
>
> - If your job is a MapReduce job and the version of MapReduce is V1.0, you cannot use Graph jobs to read data from or write data to the table whose column is dropped.
> - If your job is a Spark job that runs on the Cupid console and one of the following Spark versions is used, data can be read from the table whose column is dropped but cannot be written to the table.
>   - Spark-2.3.0-odps0.34.0
>   - Spark-3.1.1-odps0.34.0
> - If your job is a Machine Learning Platform for AI (PAI) job, data can be read from the table whose column is dropped but cannot be written to the table.
> - If your job is a Hologres job and the version of Hologres is earlier than Hologres V1.3, data cannot be read from and written to the table whose column is dropped when Hologres references the table as an external table.
>
> When you use MaxCompute Streaming Tunnel to write data to the table whose column is dropped, you cannot change the table schema.

- Syntax

```
alter table <table_name> drop columns <col_name1>[, <col_name2>...];
```

- Parameters
  - table_name: required. The name of the table from which you want to drop columns.
  - col_name: required. The name of the column that you want to drop.
- Examples

```
-- Drop the customer_id column from the sale_detail table. Enter yes on the MaxCompute cl
ient to drop the column.
alter table sale_detail drop columns customer_id;
-- Drop the shop_name and customer_id columns from the sale_detail table. Enter yes on th
e MaxCompute client to drop the columns.
alter table sale_detail drop columns shop_name, customer_id;
```

Change the order of a column

Changes the order of a column in an existing non-partitioned table or partitioned table.

> **Note** This feature is in trial use.
>
> In the following scenarios, if you change the order of a column in a table, add a column to a table, and change the order of the new column in the table, the read and write operations on the table are affected.
>
> - If your job is a MapReduce job and the version of MapReduce is V1.0, you cannot use Graph jobs to read data from or write data to the table whose column is dropped.
> - If your job is a Spark job that runs on the Cupid console and one of the following Spark versions is used, data can be read from the table whose column is dropped but cannot be written to the table.
>   - Spark-2.3.0-odps0.34.0
>   - Spark-3.1.1-odps0.34.0
> - If your job is a Machine Learning Platform for AI (PAI) job, data can be read from the table whose column is dropped but cannot be written to the table.
> - If your job is a Hologres job and the version of Hologres is earlier than Hologres V1.3, data cannot be read from and written to the table whose column is dropped when Hologres references the table as an external table.
>
> When you use MaxCompute Streaming Tunnel to write data to the table whose column is dropped, you cannot change the table schema.

- Syntax

```
alter table <table_name> change <old_column_name> <new_column_name> <column_type> after <
column_name>;
```

- Parameters
  - table_name: required. The name of the table in which you want to change the column order.
  - old_column_name: required. The original name of the column whose order you want to change.
  - new_col_name: required. The new name of the column whose order you want to change. The value of new_col_name can be the same as the value of old_column_name. If they are the same, the name of the column is not changed. However, the name specified by the new_col_name parameter cannot be the same as the name of a column except the name specified by the old_column_name parameter.
  - column_type: required. The original data type of the column whose order you want to change. The value of this parameter cannot be changed.
  - column_name: required. The column after which you want to place the specified column.

- Examples

```
-- Change the name of the customer_id column to customer and place the customer column af
ter the total_price column in the sale_detail table.
alter table sale_detail change customer_id customer string after total_price;
-- Place the customer_id column after the total_price column in the sale_detail table. Th
e name of the customer_id column is not changed.
alter table sale_detail change customer_id customer_id string after total_price;
```

Modify the name of a column

Modifies the name of a column in an existing non-partitioned table or partitioned table.

- Syntax

```
alter table <table_name> change column <old_col_name> rename to <new_col_name>;
```

- Parameters
  - table_name: required. The name of the table in which you want to modify the name of a column.
  - old_col_name: required. The name of the column that you want to modify. The column specified by old_col_name must already exist in the table.
  - new_col_name: required. The new name of the column. The table does not contain a column named new_col_name.
- Examples

```
-- Change the name of the customer_name column to customer in the sale_detail table.
alter table sale_detail change column customer_name rename to customer;
```

Modify the comment of a column

Modifies the comment of a column in an existing non-partitioned table or partitioned table.

- Syntax

```
alter table <table_name> change column <col_name> comment '<col_comment>';
```

- Parameters
  - table_name: required. The name of the table in which you want to modify the column comment.
  - col_name: required. The name of the column whose comment you want to modify. The column specified by col_name must already exist in the table.
  - col_comment: required. The new comment of the column. The comment must be a valid string that is 1 to 1,024 bytes in length. If the value of this parameter does not meet the requirements, an error is returned.
- Examples

```
-- Modify the comment of the customer column in the sale_detail table.
alter table sale_detail change column customer comment 'customer';
```

Change the name and comment of a column

Changes the name or comment of a column in a non-partitioned table or partitioned table.

- Syntax

```
alter table <table_name> change column <old_col_name> <new_col_name> <column_type> commen
t '<col_comment>';
```

- Parameters
  - table_name: required. The name of the table in which you want to change the name and comment of a column.
  - old_col_name: required. The original name of the column whose name and comment need to be changed. The column specified by `old_col_name` must already exist in the table.

- new_col_name: required. The new name of the column. The table does not contain a column named `new_col_name`.
- column_type: required. The data type of the column.
- col_comment: optional. The new comment of the column. The maximum size of a comment is 1,024 bytes.

- Examples

```
-- Change the name of the customer_name column in the sale_detail table to customer_newna
me and the comment of the column to customer.
alter table sale_detail change column customer_name customer_newname STRING comment 'cust
omer';
```

Change the non-nullable property of a non-partition key column in a table

Changes the non-nullable property of a non-partition key column in a table. If the values of a non-partition key column in a table cannot be null, you can execute the statement to allow the column values to be null.

You can execute the `DESC EXTENDED table_name;` statement to view the value of the `Nullable` property. This property determines whether the value of a non-partition key column can be null. If the value of the `Nullable` property is `true`, the column value can be null. If the value of the `Nullable` property is `false`, the column value must not be null.

- Limits

  After you change the value of the Nullable property to true, you cannot restore the property setting. Proceed with caution.

- Syntax

```
alter table <table_name> change column <old_col_name> null;
```

- Parameters
  - table_name: required. The name of the table in which you want to change the value of the Nullable property.
  - old_col_name: required. The name of the non-partition key column whose Nullable property you want to change. The column specified by old_col_name must already exist in the table.

- Examples

```
-- Create a partitioned table. The values in the id column must not be null.
create table null_test(id int not null, name string) partitioned by (ds string);
-- Change the values in the id column to null.
alter table null_test change column id null;
```

3.4.3. Lifecycle management operations

MaxCompute provides the table lifecycle management feature. This helps you release the storage space and reclaim a partition of this table.

Lifecycle

When you create a table, you can use the `lifecycle` keyword to specify the lifecycle of this table.

The `LastDataModifiedTime` value of a table in MaxCompute is updated every time the data of the table is modified. MaxCompute determines whether to reclaim this table based on its `LastDataModifiedTime` value and lifecycle.

- If the table is a non-partitioned table and its data remains unchanged for a specified number of `days` after data is last modified, MaxCompute automatically executes a statement, such as `DROP TABLE`, to reclaim the table.
- If the table is a partitioned table, MaxCompute determines whether to reclaim a partition of this table based on the `LastDataModifiedTime` value of the table. The table is not deleted even after its last partition is reclaimed.

The following table describes lifecycle-related operations.

Operation	Description	Role	Operation platform
Configure the lifecycle for a table	Configures the lifecycle for a table when you create the table.	Users who have the ALTER permission on tables	You can run these commands on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Change the lifecycle configured for a table	Changes the lifecycle configured for an existing partitioned table or a non-partitioned table.		
Disable or restore the lifecycle	Disables or restores the lifecycle configured for a specified table or partition.		

Limits

- The lifecycle can be configured only for tables instead of partitions. After the lifecycle is configured for a partitioned table, the lifecycle takes effect for all partitions in the table.
- You cannot cancel the lifecycles of non-partitioned tables. You can only modify the lifecycles of these tables.
- You can cancel the lifecycle of a specified partition and modify the lifecycles only for tables.

Configure the lifecycle for a table

Configures the lifecycle for a table when you create the table.

- Syntax

```
-- Create a table.
 create [external] table [if not exists] <table_name>
 [(<col_name> <data_type> [default <default_value>] [comment <col_comment>], ...)]
 [comment <table_comment>]
 [partitioned by (<col_name> <data_type> [comment <col_comment>], ...)]
 -- Configure the shuffle and sort properties of a clustered table when you create the ta
ble.
 [clustered by | range clustered by (<col_name> [, <col_name>, ...]) [sorted by (<col_nam
e> [asc | desc] [, <col_name> [asc | desc] ...])] into <number_of_buckets> buckets]
 -- Used only for external tables.
 [stored by StorageHandler]
 -- Used only for external tables.
 [with serdeproperties (options)]
 -- Used only for external tables.
 [location <osslocation>]
 lifecycle <days>;
-- Create a table based on an existing table.
create table [if not exists] <table_name> [as <select_statement> | like <existing_table_n
ame>];
```

- Parameters
  - table_name: required. The name of the table for which you want to configure a lifecycle.
  - days: required. The lifecycle after modification. The value must be a positive integer. Unit: days.

> **Note** For more information about the parameters required for creating a table, see Create a table.

- Example

```
-- Create the test_lifecycle table with a lifecycle of 100 days.
create table test_lifecycle (key string) lifecycle 100;
```

Change the lifecycle configured for a table

Changes the lifecycle configured for an existing partitioned table or a non-partitioned table.

- Syntax

```
alter table <table_name> set lifecycle <days>;
```

- Parameters
  - table_name: required. The name of the table whose lifecycle you want to change.
  - days: required. The lifecycle after modification. The value must be a positive integer. Unit: days.
- Example

```
-- Change the lifecycle configured for the test_lifecycle table to 50 days.
alter table test_lifecycle set lifecycle 50;
```

Disable or restore the lifecycle

Disables or restores the lifecycle configured for a specified table or partition.

- Syntax

```
alter table <table_name> [<pt_spec>] {enable|disable} lifecycle;
```

- Parameters
  - table_name: required. The name of the table whose lifecycle you want to disable or restore.
  - pt_spec: optional. The partition of the table whose lifecycle you want to disable or restore. The parameter value is in the format of `partition_col1=col1_value1, partition_col2=col2_value1...`. If a table has multi-level partitions, you must specify column values of all partitions.
  - enable: restores the lifecycle of a table or a specified partition of a table.
    - After the table lifecycle management feature is enabled again, a table and its partitions can be reclaimed. By default, the lifecycle settings of the current table and its partitions are used.
    - Before you enable the table lifecycle management feature for a table, you can modify the lifecycle settings of the table and its partitions. This prevents data from being mistakenly reclaimed due to the use of previous settings.
  - disable: disables the table lifecycle management feature for a table or a specified partition.
    - Disabling the lifecycle of a table and its partitions takes precedence over restoring the lifecycle of a table and its partitions. If you specify `table disable lifecycle`, `pt_spec enable lifecycle` becomes invalid.
    - After you disable the table lifecycle management feature for a table, the lifecycle settings for the table, and the enable or disable flag for table partitions are retained.
    - After the table lifecycle management feature is disabled for a table, you can still change the lifecycle settings of the table and its partitions.
- Examples
  - Example 1: Disable the table lifecycle management feature for the trans table.

```
alter table trans disable lifecycle;
```

  - Example 2: Disable the table lifecycle management feature for the dt='20141111' partition in the trans table.

```
alter table trans partition (dt='20141111') disable lifecycle;
```

3.4.4. View-related operations

A view is a virtual table that is created based on existing tables. Its structure and content are derived from these tables. A view corresponds to one or more tables. You can use views if you want to retain query results without creating additional tables.

The following table describes view-related operations.

Operation	Description	Role	Operation platform

Operation	Description	Role	Operation platform
Create or update a view	Creates a view or updates an existing view based on a query statement.	Users who have the CREATE TABLE permission on a project	You can execute the statements that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Rename a view	Renames an existing view.	Users who have the ALTER permission on tables	
Query a view	Views the information of an existing view.	Users who have the DESCRIBE permission on the metadata of a table	
Drop a view	Drops an existing view.	Users who have the DROP permission on tables	
Change the owner of a view	Changes the owner of an existing view.	Users who have the ALTER permission on tables	

Create or update a view

Creates a view or updates an existing view based on a query statement.

- Limits
  - A view can reference other views but cannot reference itself. Circular reference is not supported.
  - You are not allowed to write data to a view. For example, `INSERT INTO` or `INSERT OVERWRITE` cannot be executed on a view.
- Syntax

```
create [or replace] view [if not exists] <view_name>
    [(<col_name> [comment <col_comment>], ...)]
    [comment <view_comment>]
    as <select_statement>;
```

- Parameters
  - or replace: optional. This parameter is required when you want to update a view.

- if not exists: optional. If the `CREATE VIEW` statement is executed without if not exists and the view that you want to create already exists, an error is returned. In this case, you can execute the `CREATE OR REPLACE VIEW` statement to recreate the view. The permissions on the view remain unchanged after the view is recreated.
- view_name: required. The name of the view that you want to create or update.
- col_name: required. The names of the columns in the view that you want to create.
- col_comment: optional. The column comments of the view that you want to create.
- view_comment: optional. The comment of the view that you want to create.
- select_statement: required. The `SELECT` clause that provides the data source of the view. You must have read permissions on the table that the view references. When you create or update a view, you can use only one valid `SELECT` clause.

> **Note** After a view is created or updated, the view may be inaccessible if its referenced table is modified. For example, the referenced table is deleted. You must maintain the mappings between referenced tables and views.

- Examples
  - Example 1: Create the sale_detail_view view based on the sale_detail table.

```
create view if not exists sale_detail_view
(store_name, customer_id, price, sale_date, region)
comment 'a view for table sale_detail'
as select * from sale_detail;
```

  - Example 2: Update the sale_detail_view view based on the sale_detail table.

```
create or replace view if not exists sale_detail_view
(store_name, customer_id, price)
comment 'a view for table sale_detail'
as select shop_name, customer_id, total_price from sale_detail;
```

Rename a view

Renames an existing view.

- Syntax

```
alter view <view_name> rename to <new_view_name>;
```

- Parameters
  - view_name: required. The name of the view that you want to rename.
  - new_view_name: required. The new name of the view. If a view with the same name already exists, an error is returned.
- Example

```
-- Rename the sale_detail_view view as market.
alter view sale_detail_view rename to market;
```

Query a view

For more information about this command, see View the information about tables or views.

Drop a view

Drops an existing view.

- Syntax

```
drop view [if exists] <view_name>;
```

- Parameters
  - if exists: optional. If you run the command without specifying the if exists option and the view does not exist, an error is returned.
  - view_name: required. The name of the view that you want to drop.
- Example

```
-- Drop the sale_detail_view view.
drop view if exists sale_detail_view;
```

Change the owner of a view

Changes the owner of an existing view.

- Syntax

```
alter view <view_name> changeowner to <new_owner>;
```

- Parameters
  - view_name: required. The name of the view whose owner you want to change.
  - new_owner: required. The new owner of the view.
- Example

```
-- Change the owner of the sale_detail_view view to ALIYUN$xxx@aliyun.com.
alter view sale_detail_view changeowner to 'ALIYUN$xxx@aliyun.com';
```

3.4.5. Materialized view operations

A materialized view is a database object that stores the pre-calculation results of a time-consuming query operation, such as JOIN or AGGREGATE. You can reuse the pre-calculation results when you want to perform the same query. This way, the speed of queries is accelerated. This topic describes the statements and syntax that are related to materialized views. This topic also provides examples on how to use materialized views.

Background information

A view is a stored query accessible as a virtual table. Each time you query a view, the query statement is converted into the SQL statement that is used to define the view. A materialized view is a special physical table that occupies storage resources to store real data. For more information about the billing rules for materialized views, see Billing rules.

Materialized views are suitable for the following queries:

- Queries that are in a fixed mode and are frequently executed

- Queries that involve time-consuming operations, such as JOIN or AGGREGATE
- Queries that involve only a small portion of data in a table

The following table compares traditional queries and queries based on materialized views.

Item	Traditional query	Query based on a materialized view

Query statements	SQL statements are used to query data.
	`select empid, deptname`
	`from emps join depts`
	`on emps.deptno=depts.deptno`
	`where hire_date >= '2018-01-01';`
		Data is queried based on the materialized view that you created.
		Statement that is used to create a materialized view:
		`create materialized view mv`
		`as select empid, deptname,`
		`hire_date`
		`from emps join depts`
		`on emps.deptno=depts.deptno`
		`where hire_date >= '2016-01-01';`
		Statement that is used to query data based on the materialized view that you created:
		`select empid, deptname from mv`
		`where hire_date >= '2018-01-01';`
		If the query rewrite feature is enabled for the materialized view, data is directly obtained from the query result that is contained in the materialized view when you execute the first SQL statement in the following code:
		`select empid, deptname`
		`from emps join depts`
		`on emps.deptno=depts.deptno`
		`where hire_date >= '2018-01-01';`
		`-- The preceding statement is equivalent to the following statement:`
		`select empid, deptname from mv`
		`where hire_date >= '2018-01-01';`

Item	Traditional query	Query based on a materialized view
Query characteristics	Queries involve table reading, JOIN operations, and filter operations that are performed by using WHERE clauses. If the source table contains a large amount of data, the query speed is slow. The operation complexity is high and the operation efficiency is low.	Queries involve table reading and filter operations. JOIN operations are not involved. MaxCompute matches the optimal materialized view and reads data from the optimal materialized view. This greatly improves query efficiency.

The following table describes the operations that you can perform on materialized views by executing statements.

Operation	Description	Role	Platform
Create a materialized view that supports clustering or partitioning	Creates a materialized view based on a query statement.	Users who have the CREATE TABLE permission on a project	You can execute the statements that are described in this topic on the following platforms: • MaxCompute client • Query editor of the MaxCompute console • DataWorks console • MaxCompute Studio
Update a materialized view	Updates an existing materialized view.	Users who have the ALTER permission on tables	
Change the lifecycle of an existing materialized view	Changes the lifecycle of an existing materialized view.	Users who have the ALTER permission on tables	
Enable or disable the lifecycle feature for an existing materialized view	Enables or disables the lifecycle feature for an existing materialized view.	Users who have the ALTER permission on tables	
Query the information about a materialized view	Queries the basic information about a materialized view.	Users who have the DESCRIBE permission on the metadata of a table	

Operation	Description	Role	Platform
Query the status of a materialized view	Queries the status of a materialized view.	Users who have the DESCRIBE permission on the metadata of a table	
Drop an existing materialized view	Drops an existing materialized view.	Users who have the DROP permission on tables	
Drop partitions from an existing materialized view	Drops partitions from an existing materialized view.	Users who have the DROP permission on tables	
Perform a penetration query based on a materialized view	Queries data from the source partitioned table if the partition data that you want to query is not contained in the materialized view.	Users who have write permissions and the CREATE TABLE permission on a project	

Limits

Before you use materialized views, take note of the following limits:

- Window functions are not supported.
- User-defined table-valued functions (UDTFs) are not supported.
- By default, non-deterministic functions, such as user-defined scalar functions (UDFs) and user-defined aggregate functions (UDAFs) are not supported. If you must use non-deterministic functions, run the `set odps.sql.materialized.view.support.nondeterministic.function=true;` command at the session level.

Create a materialized view that supports clustering or partitioning

You can create a materialized view that supports clustering or partitioning based on the data for materialized view scenarios.

- Limits
  - The name of the materialized view that you want to create cannot be the same as the name of an existing table, view, or materialized view. You can execute the `show tables;` statement to view the names of all tables and materialized views in a MaxCompute project.
  - You cannot create a materialized view based on an existing materialized view.
  - You cannot create a materialized view based on an external table.
- Precautions

- If the query statement based on which you create a materialized view fails to be executed, you cannot create the materialized view.
- Partition key columns in a materialized view must be derived from a source table. The sequence and number of the columns in the materialized view must be the same as the sequence and number of the columns in the source table. Column names can be different.
- You must specify comments for all columns, including partition key columns. If you specify comments only for some columns, an error is returned.
- You can specify both the partitioning and clustering attributes for a materialized view. In this case, the data in each partition has the specified clustering attribute.
- If the query statement based on which you create a materialized view contains operators that are not supported by the materialized view, an error is returned. For more information about the operators that are supported by materialized views, see Perform a query rewrite operation based on a materialized view.
- By default, MaxCompute does not allow you to create materialized views by using non-deterministic functions, such as UDFs or UDAFs. If you must use non-deterministic functions, run the `set odps.sql.materialized.view.support.nondeterministic.function=true;` command at the session level.

- Syntax

```
set odps.sql.materialized.view.initial.partition={"Partition key column": "Column value"}
;
-- Optional. This command is used to specify the initial partition to prevent excessive u
nnecessary partitions from being created.
create materialized view [if not exists] [<project_name>.]<mv_name>
[lifecycle <days>]
[(<col_name> [comment <col_comment>], ...)]
[{disable|enable rewrite]
[partitioned on (<col_name> [, <col_name>, ...])
[clustered by|range clustered by (<col_name> [, <col_name>, ...])
     [sorted by (<col_name> [asc | desc] [, <col_name> [asc | desc] ...])]
           into <number_of_buckets> buckets]
[tblproperties("compressionstrategy"="normal/high/extreme","enable_auto_substitute"="true
")]
as <select_statement>;
```

- Parameters
  - if not exists: optional. If you do not specify if not exists and the materialized view that you want to create already exists, an error is returned.
  - project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.
  - mv_name: required. The name of the materialized view that you want to create.
  - days: optional. The lifecycle of the materialized view that you want to create. Unit: days.
  - col_name: optional. The name of a column in the materialized view that you want to create.
  - col_comment: optional. The comment on a column in the materialized view that you want to create.

- disable|enable rewrite: optional. Specifies whether to disable or enable the query rewrite feature for the materialized view. If you do not configure this parameter, the query rewrite feature is enabled for the materialized view. You can execute the `alter materialized view [project_name.]<mv_name> disable rewrite;` statement to disable this feature for the materialized view. You can also execute the `alter materialized view [project_name.]<mv_name> enable rewrite;` statement to enable this feature for the materialized view.
- partitioned on: optional. The partition key columns in the materialized view that you want to create. If you want to create a partitioned materialized view, you must configure this parameter.
- clustered by|range clustered by: optional. The shuffle attribute of the materialized view that you want to create. If you want to create a clustered materialized view, you must configure the clustered by or range clustered by parameter.
- sorted by: optional. The sort attribute of the materialized view that you want to create. If you want to create a clustered materialized view, you must configure this parameter.
- number_of_buckets: optional. The number of buckets in the materialized view that you want to create. If you want to create a clustered materialized view, you must configure this parameter.
- tblproperties: optional. compressionstrategy specifies the data storage and compression policy of the materialized view that you want to create. Valid values: normal, high, and extreme. enable_auto_substitute specifies whether to automatically query data from the source table if the materialized view does not contain the partition from which you want to query data. For more information, see Perform a penetration query based on a materialized view.
- select_statement: required. The SELECT statement. For more information about the syntax of the SELECT statement, see SELECT syntax.

- Examples
  - Example 1: Create a materialized view that contains a partition key column named ds. Sample statement:

```
create materialized view mv lifecycle 7
(
  key comment 'unique id',
  ds comment 'date'
)
partitioned on (ds)
as select t1.id, t1.value, t1.ds as ds from t1 join t2 on t1.id = t2.id;
```

  - Example 2: Create a non-partitioned materialized view that is clustered. Sample statement:

```
create materialized view mv lifecycle 7
clustered by id sorted by value into 1024 buckets
as select t1.id, t1.value, t1.ds as ds from t1 join t2 on t1.id = t2.id;
```

  - Example 3: Create a partitioned materialized view that is clustered. Sample statement:

```
create materialized view mv lifecycle 7
partitioned on (ds)
clustered by id sorted by value into 1024 buckets
as select t1.id, t1.value, t1.ds as ds from t1 join t2 on t1.id = t2.id;
```

Update a materialized view

If you perform operations, such as insert, overwrite, update, and delete operations on the table or partition that corresponds to a materialized view, the materialized view becomes invalid and cannot be used for query rewrite operations. You can check the status of a materialized view. If the materialized view is invalid, you must update the materialized view. For more information about how to check the status of a materialized view, see Query the status of a materialized view.

- Precautions
  - You can perform only full updates on a materialized. You cannot perform incremental updates on a materialized view.
  - You can trigger scheduled updates on a materialized view in the DataWorks console. For more information about how to trigger scheduling operations in the DataWorks console, see Scheduling configuration.
- Syntax

```
alter materialized view [<project_name>.]<mv_name> rebuild [partition(<expression1>, <exp
ressio2>...)];
```

- Parameters
  - project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.
  - mv_name: required. The name of the materialized view that you want to update.
  - expression: optional. The expression that is used to specify the partitions that you want to update. If you want to update a partitioned materialized view, you must configure this parameter.
- Examples
  - Example 1: Update a non-partitioned materialized view. Sample statement:

```
alter materialized view count_mv rebuild;
```

  - Example 2: Update a partition of a partitioned materialized view. Sample statement:

```
alter materialized view mv rebuild partition (ds='20210101');
```

  - Example 3: Update the partitions of a partitioned materialized view that meet the specific conditions. Sample statement:

```
alter materialized view mv rebuild partition(ds>='20210101', ds<='20210105');
```

Change the lifecycle of an existing materialized view

You can change the lifecycle of an existing materialized view.

- Syntax

```
alter materialized view [<project_name>.]<mv_name> set lifecycle <days>;
```

- Parameters
  - project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.

- mv_name: required. The name of the materialized view that you want to update.
- days: required. The new lifecycle of the materialized view. Unit: days.

- Examples

```
-- Change the lifecycle of an existing materialized view to 10 days.
alter materialized view mv set lifecycle 10;
```

Enable or disable the lifecycle feature for an existing materialized view

You can enable or disable the lifecycle feature for an existing materialized view.

- Syntax

```
alter materialized view [<project_name>.]<mv_name> [<pt_spec>] enable|disable lifecycle;
```

- Parameters
  - project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.
  - mv_name: required. The name of the materialized view for which you want to enable or disable the lifecycle feature.
  - pt_spec: optional. The partition information of the materialized view for which you want to enable or disable the lifecycle feature. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col indicates the column name, and partition_col_value indicates the column value.
  - enable|disable: required. The `enable` option specifies that the lifecycle feature is enabled for a materialized view or a partition of a materialized view. The `disable` option specifies that the lifecycle feature is disabled for a materialized view or a partition of a materialized view. If you disable the lifecycle feature, lifecycle management is not required for the materialized view or partition.

- Examples
  - Example 1: Enable the lifecycle feature for a materialized view. Sample statement:

```
alter materialized view mv partition (ds='20210101') enable lifecycle;
```

  - Example 2: Disable the lifecycle feature for a materialized view. Sample statement:

```
alter materialized view mv partition (ds='20210101') disable lifecycle;
```

Query the information about a materialized view

You can query the information about a materialized view, including the schema and modification time of the materialized view.

- Syntax

```
desc extended [<project_name>.]<mv_name>;
```

- Parameters

- project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.
- mv_name: required. The name of the materialized view that you want to query.

- Examples

```
desc extended mv;
```

The following code shows a sample query result:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$****@aliyunid.com | Project: ****                                    |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2021-08-01 17:50:15                                      |
| LastDDLTime:              2021-08-01 17:50:15                                      |
| LastModifiedTime:         2021-08-01 17:50:15                                      |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 0                                                  |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field    | Type    | Label | ExtendedLabel | Nullable | DefaultValue | Comment      |
+------------------------------------------------------------------------------------+
| page_id  | string |       |               | true     | NULL         |              |
| _c1      | bigint |       |               | true     | NULL         |              |
+------------------------------------------------------------------------------------+
| Extended Info:                                                                     |
+------------------------------------------------------------------------------------+
| TableID:                  e4a7f1169588400ab39bc3076426****                         |
| IsArchived:               false                                                    |
| PhysicalSize:             0                                                        |
| FileNum:                  0                                                        |
+------------------------------------------------------------------------------------+
```

Query the status of a materialized view

You can query the status of a materialized view. This operation allows you to view changes to the source table and determines whether the materialized view is valid. A materialized view can be in one of the following states:

- Valid

  When you execute a query statement, MaxCompute queries data from the materialized view instead of querying data from the source table.

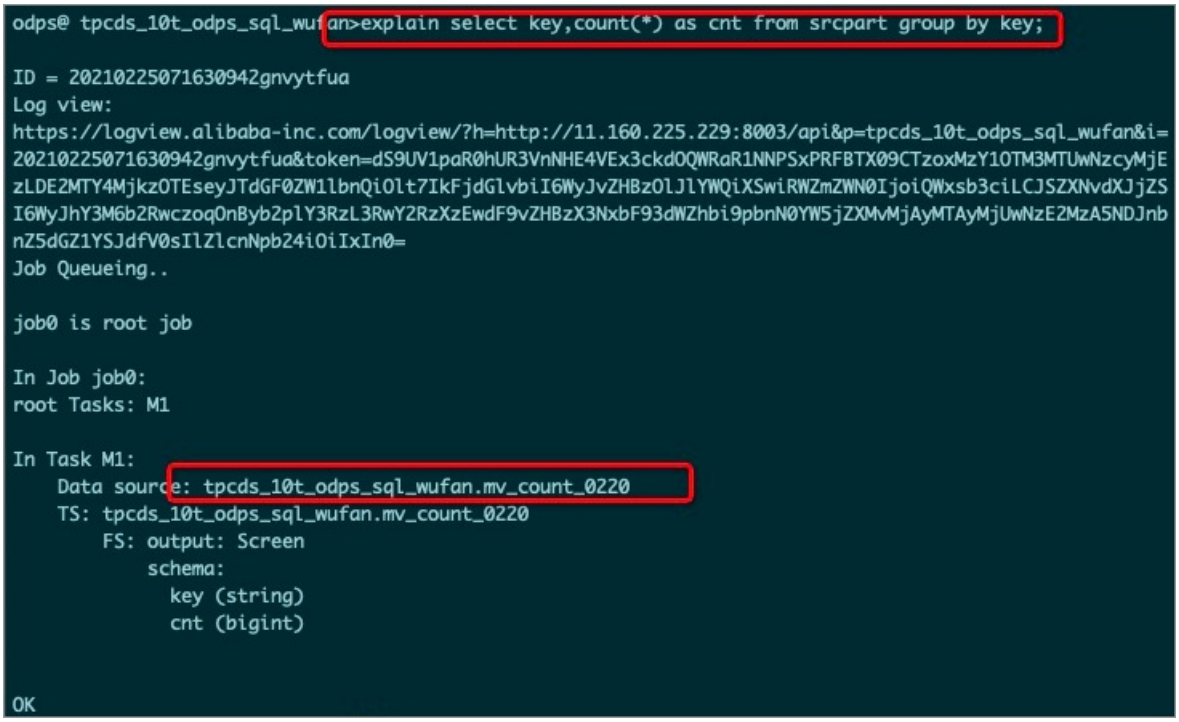
- Invalid

  When you execute a query statement, MaxCompute cannot directly query data from the materialized view. In this case, MaxCompute queries data from the source table. As a result, the query speed is not accelerated.

MaxCompute does not allow you to query the status of a materialized view by running commands. To determine whether a materialized view is valid, you can use one of the following methods:

- Method 1: Execute the EXPLAIN statement to check whether a materialized view is valid.

  In the following figure, the name of a materialized view appears as the data source in the output. This indicates that the query uses the materialized view.



- Method 2: View the job execution diagram on the Logview UI or view information on the **Json Summary** tab of the Logview UI.

  If the name of a materialized view can be queried on the job execution diagram or **Json Summary** tab, the materialized view is valid. Otherwise, the materialized view is invalid.

Drop an existing materialized view

You can drop an existing materialized view.

- Syntax

```
drop materialized view [if exists] [<project_name>.]<mv_name>;
```

- Parameters
  - if exists: optional. If you do not specify if exists and the materialized view that you want to drop does not exist, an error is returned.
  - project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.
  - mv_name: required. The name of the materialized view that you want to drop.

- Examples

```
drop materialized view mv;
```

Drop partitions from an existing materialized view

You can drop one or more partitions from an existing materialized view.

- Syntax

```
alter materialized view [<project_name>.]<mv_name> drop [if exists] partition <pt_spec> [
partition <pt_spec>, partition <pt_spec>....];
```

- Parameters
  - project_name: optional. The name of the MaxCompute project to which the materialized view belongs. If you do not configure this parameter, the current MaxCompute project is used. You can log on to the MaxCompute console. In the top navigation bar, select a region and view the name of the MaxCompute project on the **Project management** tab.
  - mv_name: required. The name of the partitioned materialized view from which you want to drop partitions.
  - if exists: optional. If you do not specify if exists and the materialized view that you want to drop does not exist, an error is returned.
  - pt_spec: The partitions that you want to drop. You must specify at least one partition. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partitio n_col_value2, ...)` format. partition_col indicates the column name, and partition_col_value indicates the column value.

- Examples
  - Example 1: Drop a partition from a partitioned materialized view. Sample statement:

```
alter materialized view mv drop partition (ds='20210101');
```

  - Example 2: Drop the partitions that meet the specific conditions from a partitioned materialized view. Sample statement:

```
alter materialized view mv drop partition (ds>='20210101', ds<='20210105');
```

Perform a query rewrite operation based on a materialized view

The most important feature of materialized views is to perform query rewrite operations on query statements. To perform query rewrite operations on a query statement based on a materialized view, you must add `set odps.sql.materialized.view.enable.auto.rewriting=true;` before the query statement. If a materialized view is invalid, the materialized view cannot be used for query rewrite operations. In this case, data is queried from the source table, and the query speed is not accelerated.

> **Note** By default, a MaxCompute project can use only its materialized views for query rewrite operations. If you want to perform query rewrite operations on query statements based on the materialized views of other MaxCompute projects, you must add `set odps.sql.materialized.view.source.project.white.list=<project_name1>,<project_name2>, <project_name3>;` before the query statements to specify the MaxCompute projects.

The following table describes the types of operators that are supported by query rewrite operations of materialized views in MaxCompute and other services.

Operator	Category	MaxCompute	BigQuery	Amazon RedShift	Hive

Operator	Category	MaxCompute	BigQuery	Amazon RedShift	Hive
FILTER	Exact match of expressions	Supported	Supported	Supported	Supported
	Partial match of expressions	Supported	Supported	Supported	Supported
AGGREGATE	Single AGGREGATE operation	Supported	Supported	Supported	Supported
	Multiple AGGREGATE operations	Not supported	Not supported	Not supported	Not supported
JOIN	JOIN operation	INNER JOIN	Not supported	INNER JOIN	INNER JOIN
	Single JOIN operation	Supported	Not supported	Supported	Supported
	Multiple JOIN operations	Supported	Not supported	Supported	Supported
AGGREGATE+JOIN	-	Supported	Not supported	Supported	Supported

The query rewrite operations based on a materialized view require that the data in a query statement be obtained from the materialized view. The data includes output columns, the columns required by filter operations, the columns required by aggregate functions, and the columns required by JOIN operations. If the columns that are required in the query statement are not included in the materialized view or are not supported by the aggregate functions, you cannot perform query rewrite operations based on the materialized view. Examples:

- Rewrite a query statement that contains filter conditions. The following statement creates a materialized view.

  ```
  create materialized view mv as select a,b,c from src where a>5;
  ```

  Execute a query statement based on the created materialized view. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select a,b from src where a>5;`	`select a,b from mv;`
`select a, b from src where a=10;`	`select a,b from mv where a=10;`
`select a, b from src where a=10 and b=3;`	`select a,b from mv where a=10 and b=3;`

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select a, b from src where a>3;`	`(select a,b from src where a>3 and a<=5) union (select a,b from mv);`
`select a, b from src where a=10 and d=4;`	The query rewrite operation fails because Column d does not exist in mv.
`select d, e from src where a=10;`	The query rewrite operation fails because Column d and Column e do not exist in mv.
`select a, b from src where a=1;`	The query rewrite operation fails because data that meets the expression of a=1 does not exist in mv.

- Rewrite a query statement with aggregate functions
  - If the SQL statement of a materialized view has the same aggregate key as the query statement, all aggregate functions can be rewritten. If different aggregate keys are used, only aggregate functions SUM, MIN, and MAX are supported.

    The following statement creates a materialized view.

    ```
    create materialized view mv as
    select a, b, sum(c) as sum, count(d) as cnt from src group by a, b;
    ```

    Execute a query statement based on the created materialized view. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select a, sum(c) from src group by a;`	`select a, sum(sum) from mv group by a;`
`select a, count(d) from src group by a, b;`	`select a, cnt from mv;`
`select a, count(b) from (select a, b from src group by a, b) group by a;`	`select a,count(b) from mv group by a;`
`select a, count(b) from src group by a;`	The query rewrite operation fails because Column a and Column b have been aggregated in the materialized view. As a result, Column b cannot be re-aggregated.
`select a, count(c) from src group by a;`	The query rewrite operation fails because the COUNT function does not support re-aggregation.

- If an aggregate function includes `DISTINCT`, the query rewrite operation is supported when the SQL statement of the materialized view has the same aggregate key as the query statement. Otherwise, the query rewrite operation is not supported.

  The following statement creates a materialized view.

  ```
  create materialized view mv as
  select a, b, sum(distinct c) as sum, count(distinct d) as cnt from src group by a, b;
  ```

  Execute a query statement based on the created materialized view. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select a, count(distinct d) from src group by a, b;`	`select a, cnt from mv;`
`select a, count(c) from src group by a, b;`	The query rewrite operation fails because the COUNT function does not support re-aggregation.
`select a, count(distinct c) from src group by a;`	The query rewrite operation fails because data in Column a needs to be re-aggregated.

- Rewrite a query statement that contains a `JOIN` clause
  - Data input by using `JOIN` clauses

    The following statements create materialized views.

    ```
    create materialized view mv1 as select a, b from j1 where b > 10;
    create materialized view mv2 as select a, b from j2 where b > 10;
    ```

    Execute a query statement based on the created materialized views. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select j1.a,j1.b,j2.a from (select a,b from j1 where b > 10) j1 join j2 on j1.a=j2.a;`	`select mv1.a, mv1.b, j2.a from mv1 join j2 on mv1.a=j2.a;`
`select j1.a,j1.b,j2.a from(select a,b from j1 where b > 10) j1join(select a,b from j2 where b > 10) j2on j1.a=j2.a;`	`select mv1.a,mv1.b,mv2.a from mv1 join mv2 on mv1.a=mv2.a;`

- `JOIN` clause with filter conditions

  The following statements create materialized views.

```
-- Create non-partitioned materialized views.
create materialized  view mv1 as select j1.a, j1.b from j1 join j2 on j1.a=j2.a;
create materialized  view mv2 as select j1.a, j1.b from j1 join j2 on j1.a=j2.a where j
1.a > 10;
-- Create a partitioned materialized view.
create materialized  view mv lifecycle 7 partitioned on (ds) as select t1.id, t1.ds as
ds from t1 join t2 on t1.id = t2.id;
```

  Execute a query statement based on the created materialized views. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select j1.a,j1.b from j1 join j2 on j1.a =j2.a where j1.a=4;`	`select a, b from mv1 where a=4;`
`select j1.a,j1.b from j1 join j2 on j1.a =j2.a where j1.a > 20;`	`select a,b from mv2 where a>20;`
`select j1.a,j1.b from j1 join j2 on j1.a =j2.a where j1.a > 5;`	`(select j1.a,j1.b from j1 join j2 on j1. a=j2.a where j1.a > 5 and j1.a <= 10) uni on select * from mv2;`
`select key from t1 join t2 on t1.id= t2. id where t1.ds='20210306';`	`select key from mv where ds='20210306';`
`select key from t1 join t2 on t1.id= t2. id where t1.ds>='20210306';`	`select key from mv where ds>='20210306';`
`select j1.a,j1.b from j1 join j2 on j1.a =j2.a where j2.a=4;`	The query rewrite operation fails because the materialized view does not have Column j2.a.

- `JOIN` clause for joining tables

  The following statement creates a materialized view.

```
create materialized  view mv as select j1.a, j1.b from j1 join j2 on j1.a=j2.a;
```

  Execute a query statement based on the created materialized view. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select j1.a, j1.b from j1 join j2 join j 3 on j1.a=j2.a and j1.a=j3.a;`	`select mv.a, mv.b from mv join j3 on mv. a=j3.a;`
`select j1.a, j1.b from j1 join j2 join j 3 on j1.a=j2.a and j2.a=j3.a;`	`select mv.a,mv.b from mv join j3 on mv.a =j3.a;`

> **Note**
>
> - The preceding types of statements can be used together. You can rewrite a query statement that meets rewrite conditions.
> - MaxCompute selects the optimal rewrite rule to execute query statements. If operations are added after a query statement is rewritten, MaxCompute does not select the query statement after the rewrite because this query statement is not optimal.

- Rewrite a query statement that contains a `LEFT JOIN` clause

  The following statement creates a materialized view.

```
create materialized view mv lifecycle 7(
        user_id,
        job,
        total_amount
) as select t1.user_id, t1.job, sum(t2.order_amout) as total amount
      from user_info as t1 left join sale_order as t2 on t1.user_id=t2.user_id group by t
1.user_id;
```

  Execute a query statement based on the created materialized view. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select t1.user_id, sum(t2.order_amout) as total amount from user_info as t1 left join sale_order as t2 on t1.user_id=t2.user_id group by t1.user_id;`	`select user_id, total_amount from mv;`

- Rewrite query a statement with `UNION ALL`

  The following statement creates a materialized view.

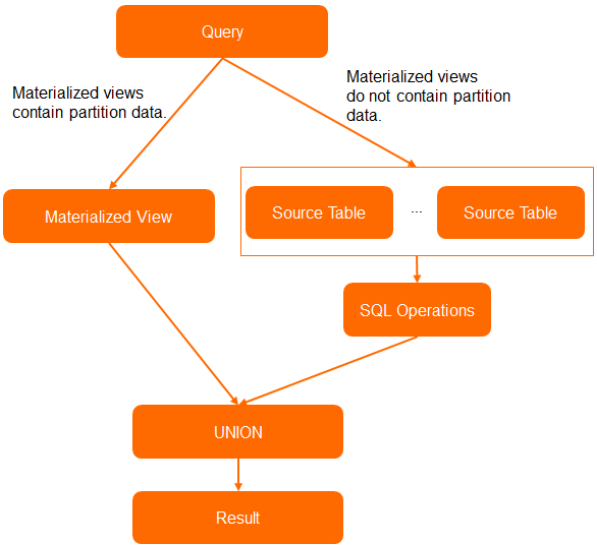
```
create materialized view mv lifecycle 7(
        user_id,
        tran_amount,
        tran_date
) as (select user_id, tran_amount, tran_date from alipay_tran union all
 select user_id, tran_amount, tran_date from unionpay_tran);
```

  Execute a query statement based on the created materialized view. The following table lists the query statements before and after the query rewrite operation.

Query statement before the query rewrite operation	Query statement after the query rewrite operation
`select user_id, tran_amount from alipay_tran union all select user_id, tran_amount from unionpay_tran;`	`select user_id, total_amount from mv;`

Perform a penetration query based on a materialized view

A partitioned materialized view does not contain the data of all partitions in a source partitioned table if only the latest partition data is updated to the partitioned materialized view. If the partition data that you want to query does not exist in a partitioned materialized view, the system performs a penetration query to query data from the source partitioned table. The following figure shows how to perform a penetration query.



To allow a materialized view to support penetration query, you must configure the following parameters:

- Set `odps.optimizer.cbo.rule.filter.black` to re at the session level to enable constant folding for partition key columns.
- When you create a materialized view, add `"enable_auto_substitute"="true"` to tblproperties.

The following example shows how to perform a penetration query based on a materialized view.

1. Create a partitioned materialized view that supports penetration query. Sample statement:

```
-- Enable constant folding for partition key columns in partitions to ensure that parti
tion data is dynamically changed.
set odps.optimizer.cbo.rule.filter.black=re;
-- Initialize the data of the 20210101 partition.
set odps.sql.materialized.view.initial.partition={"dt": "20210101"};
-- Create a partitioned materialized view that supports penetration query.
create materialized view if not exists mv lifecycle 7
tblproperties("enable_auto_substitute"="true")
partitioned on (dt)
as select id, name, dt from src;
```

2. Query the data of the 20210101 partition in the src table from the mv materialized view. Sample statement:

```
select * from mv where dt='20210101';
```

3. Query the data of the 20210102 partition in the src table from the mv materialized view. The mv materialized view does not contain the partition data. Therefore, a penetration query is performed to query the partition data from the src table. Sample statement:

```
select * from mv where dt = '20210102';
-- The preceding statement is equivalent to the following statement because the mv mate
rialized view does not contain the data of the 20210102 partition and the partition dat
a needs to be queried from the src table.
select * from (select id, name, dt from src where dt='20210102') t;
```

4. Query the data of partitions 20201230 to 20210102 from the mv materialized view. The mv materialized view does not contain all partition data. Therefore, a penetration query is performed to obtain the data that is not contained in the mv materialized view, and a UNION operation is performed on the obtained data and the data that is queried from the mv materialized view to return the final result.

```
select * from mv where dt >= '20201230' and dt<='20210102' and id='5';
-- The preceding statement is equivalent to the following statement because the mv mate
rialized view does not contain the data of the 20210102 partition and a penetration que
ry is performed to query the data of the 20210102 partition from the source table.
select * from
(select id, name, dt from src where dt='20211231' or dt='20210102';
 union all
 select * from mv where dt='20210101'
) t where id = '5';
```

Example

Scenario: A page visit table named visit_records contains the page ID, user ID, and visit time of each user. The numbers of visits to different pages need to be queried and analyzed. The following code shows the structure of the visit_records table.

```
+------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                     |
+------------------------------------------------------------------------------------+
| page_id         | string     |       |                                             |
| user_id         | string     |       |                                             |
| visit_time      | string     |       |                                             |
+------------------------------------------------------------------------------------+
```

You can create a materialized view for the visit_records table to collect the number of visits to each page based on the page ID. Then, you can perform subsequent query operations based on the materialized view.

1. Execute the following statement to create a materialized view:

```
create materialized view count_mv as select page_id, count(*) from visit_records group
by page_id;
```

2. Execute the following query statement:

```
set odps.sql.materialized.view.enable.auto.rewriting=true;
select page_id, count(*) from visit_records group by page_id;
```

When you execute the query statement, MaxCompute can match the materialized view count_mv and read the aggregated data from count_mv.

3. Execute the following statement to check whether the query statement matches the materialized view:

```
explain select page_id, count(*) from visit_records group by page_id;
```

The following result is returned:

```
job0 is root job
In Job job0:
root Tasks: M1
In Task M1:
    Data source: doc_test_dev.count_mv
    TS: doc_test_dev.count_mv
        FS: output: Screen
            schema:
              page_id (string)
              _c1 (bigint)
OK
```

In the returned result, the value of Data source contains count_mv. This indicates that the materialized view is valid and the query is rewritten.

Billing rules

When you use materialized views, you are charged for the following items:

- Storage

  Materialized views occupy physical storage space. You are charged for the physical storage space that is occupied by materialized views. For more information about the billing rules, see Storage pricing (pay-as-you-go).

- Computing

  Data is queried when you create, update, and query materialized views, and rewrite queries (if materialized views are valid). These operations consume computing resources and generate computing costs.

  - If the billing method of your MaxCompute project is subscription, you are not charged for data computations.
  - If the billing method of your MaxCompute project is pay-as-you-go, you are charged for data computations based on the complexity of the SQL statement and the amount of input data. For more information about the billing rules, see Billing for standard SQL jobs. Take note of the following points:
    - If you execute an SQL statement to create or update a materialized view, the computing fee depends on the amount of input data and complexity of the SQL statement.
    - If the materialized view is valid, data is read from the materialized view when the query rewrite operation is performed. The amount of input data that is read from the materialized view of the query statement is related to the materialized view and is irrelevant to the source table of the materialized view. If the materialized view is invalid, the query rewrite operation cannot be performed, and data is queried from the source table. The amount of input data for the query statement is related to the source table. For more information about how to query the status of a materialized view, see Query the status of a materialized view.
    - Data bloat may occur if a materialized view is generated based on the association of multiple tables. Therefore, the amount of data read from a materialized view may not be absolutely less than the amount of data in the source table. MaxCompute cannot guarantee that data reading from a materialized view costs less than data reading from the source table.

3.5. Insert Operation

3.5.1. Insert or update data into a table or a static partition (INSERT INTO and INSERT OVERWRITE)

MaxCompute allows you to execute the `INSERT INTO` or `INSERT OVERWRITE` statement to insert or update data into a table or a static partition.

You can execute the INSERT INTO or INSERT OVERWRITE statement on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Prerequisites

Before you execute these statements, make sure that you are granted the ALTER permission on the destination table and the DESCRIBE permission on metadata of the source table. For more information, see Permissions.

Description

When you use MaxCompute SQL to process data, you can execute the `INSERT INTO` or `INSERT OVERWRITE` statement to save the execution results of the `SELECT` statements to the destination table. Differences between the two statements:

- `INSERT INTO`: inserts data into a table or a static partition of a table. You can specify the values of partition key columns in this statement to insert data into a specified partition. If you want to insert a small amount of test data, you can use this statement with VALUES.
- `INSERT OVERWRITE`: clears a specified table and inserts data into the table or its static partitions.

  > **Note**
  >
  > - The `INSERT` syntax in MaxCompute is different from that in MySQL or Oracle. You must add the `TABLE` keyword and `table_name` to `INSERT OVERWRITE`. You do not need to add the `TABLE` keyword to `INSERT INTO`.
  > - If you execute the `INSERT OVERWRITE` statement on the same partition several times, the size of the partition into which data is inserted may be different every time you run the `DESC` command. This is because the logic to split files changes after you sequentially execute the `SELECT` and `INSERT OVERWRITE` statements for the same partition in a table. The total length of data remains the same after you execute `INSERT OVERWRITE`. You do not need to worry about the storage fees.

For more information about how to insert data into a dynamic partition, see Insert or overwrite data into dynamic partitions (DYNAMIC PARTITION).

Limits

When you execute `INSERT INTO` or `INSERT OVERWRITE` to insert or update data into a table or a static partition of a table, take note of the following limits:

- `INSERT INTO`: This statement cannot be used to insert data into a clustered table.
- `INSERT OVERWRITE`: This statement does not allow you to specify the columns into which you want to insert data. You can execute the `INSERT INTO` statement to specify the columns. For example, If you execute `create table t(a string, b string); insert into t(a) values ('1');`, 1 is inserted into Column a, and NULL or the default value is inserted into Column b.
- MaxCompute does not provide the locking mechanism for the tables for which INSERT operations are being performed. We recommend that you do not execute the `INSERT INTO` and `INSERT OVERWRITE` statements for a table at the same time.

Syntax

```
insert {into|overwrite} table <table_name> [partition (<pt_spec>)] [(<col_name> [,<col_name
> ...)]
<select_statement>
from <from_statement>
[zorder by <zcol_name> [, <zcol_name> ...]];
```

- table_name: required. The name of the table into which you want to insert data.
- pt_spec: optional. The partition into which you want to insert data. Only constants are allowed. Expressions, such as functions, are not allowed. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
- col_name: optional. The name of the column in the table into which you want to insert data. `INSERT OVERWRITE` does not allow you to specify `[(<col_name> [,<col_name> ...)]`.
- select_statement: required. The `SELECT` clause that is used to query the data that you want insert into the destination table from the source table. For more information about `SELECT` clauses, see SELECT syntax.

  > **Note**
  > - The mappings between the source and destination tables are based on the column sequence in `select_statement`, instead of the mappings between column names in the tables.
  > - If the destination table has static partitions and you want to insert data into a static partition, partition key columns cannot be included in `select_statement`.

- from_statement: required. The `FROM` clause that indicates the data source, such as a source table name.
- zorder by <zcol_name> [, <zcol_name> ...]: optional. If you write data to a table or partition, you can use this clause to co-locate rows with similar data records based on the columns specified in select_statement. This improves filtering performance for queries and reduces storage costs. The `ORDER BY x, y` clause sorts data records based on the ordering of x coming before y. The `ZORDER BY x, y` clause co-locates rows with similar x values and rows with similar y values. In terms of column-based data filtering and sorting in an SQL SELECT statement, the `ORDER BY` clause filters

and sorts data based on x, whereas the `ZORDER BY` clause filters and sorts data based on x or on both x and y. This increases the column compression ratio.

When you use `ZORDER BY`, take note of the following **limits**:

- If the destination table is a clustered table, the `ZORDER BY` clause is not supported.
- `ZORDER BY` can be used with `DISTRIBUTE BY` but cannot be used with `ORDER BY`, `CLUSTER BY`, or `SORT BY`.

> **Note** If you use the `ZORDER BY` clause to write data, more resources and time are consumed.

Examples

- Example 1: Execute the `INSERT INTO` statement to append data to a partitioned table named `sale_detail`. Sample statement:

```
-- Create a partitioned table named sale_detail.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add a partition to the sale_detail table.
alter table sale_detail add partition (sale_date='2013', region='china');
-- Append data to the sale_detail table. The abbreviated form of INSERT INTO TABLE table_
name is INSERT INTO table_name. However, the TABLE keyword in INSERT OVERWRITE TABLE tabl
e_name cannot be omitted.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',10
0.1),('s2','c2',100.2),('s3','c3',100.3);
-- Enable a full table scan only for the current session. Execute the SELECT statement to
view data in the sale_detail table.
set odps.sql.allow.fullscan=true;
select * from sale_detail;
-- Returned result:
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 2: Execute the `INSERT OVERWRITE` statement to update the data in the `sale_detail_insert` table. Sample statement:

```
-- Create the sale_detail_insert table that has the same schema as the sale_detail table.

create table sale_detail_insert like sale_detail;
-- Add a partition to the sale_detail_insert table.
alter table sale_detail_insert add partition (sale_date='2013', region='china');
-- Extract data from the sale_detail table and insert the data into the sale_detail_inser
t table. Names of partition key columns in the sale_detail_insert table do not need to be
declared and cannot be rearranged.
-- If the sale_detail_insert table contains static partitions, the values of partition ke
y columns are declared in PARTITION(). These values do not need to be included in select_
statement. You need only to search for column names based on the sequence of common colum
ns in the sale_detail_insert table and sequentially map the declared column values to the
columns in the sale_detail_insert table. If the sale_detail_insert table contains dynamic
partitions, the names of partition key columns must be included in select_statement. For
more information, see Insert or overwrite data into dynamic partitions (DYNAMIC PARTITION
).
insert overwrite table sale_detail_insert partition (sale_date='2013', region='china')
  select
  shop_name,
  customer_id,
  total_price
  from sale_detail
  zorder by customer_id, total_price;
-- Enable a full table scan only for the current session. Execute the SELECT statement to
view the data in the sale_detail_insert table.
set odps.sql.allow.fullscan=true;
select * from sale_detail_insert;
-- Returned result:
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 3: Execute the `INSERT OVERWRITE` statement to update the data in the `sale_detail_insert` table and adjust the sequence of columns in `select_statement`. The mappings between the source and destination tables are based on the sequence of columns in `select_statement`, instead of the mappings between column names in the two tables. Sample statement:

  ```
  insert overwrite table sale_detail_insert partition (sale_date='2013', region='china')
      select customer_id, shop_name, total_price from sale_detail;
  select * from sale_detail_insert;
  ```

  Returned result:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| c1         | s1          | 100.1       | 2013       | china      |
| c2         | s2          | 100.2       | 2013       | china      |
| c3         | s3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

When you create the `sale_detail_insert` table, the column sequence is defined as `shop_name string, customer_id string, and then total_price bigint`. However, you insert the data from the `sale_detail` table to the `sale_detail_insert` table based on the sequence of `customer_id, shop_name, and then total_price`. As a result, the data in the `sale_detail.customer_id` column is inserted into the `sale_detail_insert.shop_name` column, and the data in the `sale_detail.shop_name` column is inserted into the `sale_detail_insert.customer_id` column.

- Example 4: If you insert data into a partition, its partition key columns cannot be included in `select_statement`. After the following statement is executed, an error is returned. This is because `sale_date` and `region` are partition key columns. These columns cannot be included in `select_statement` if INSERT OVERWRITE or INSERT INTO statement is used to insert or update data into a static partition. Sample statement of incorrect usage:

```
insert overwrite table sale_detail_insert partition (sale_date='2013', region='china')
   select shop_name, customer_id, total_price, sale_date, region from sale_detail;
```

- Example 5: pt_spec in `PARTITION()` must be constants instead of expressions. Sample statement of incorrect usage:

```
insert overwrite table sale_detail_insert partition (sale_date=datepart('2016-09-18 01:10
:00', 'yyyy') , region='china')
   select shop_name, customer_id, total_price from sale_detail;
```

3.5.2. Insert or overwrite data into dynamic partitions (DYNAMIC PARTITION)

MaxCompute allows you to insert data into a dynamic partition by using `INSERT INTO` or `INSERT OVERWRITE`.

You can execute the INSERT INTO or INSERT OVERWRITE statement on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Prerequisites

Before you execute these statements, make sure that you are granted the ALTER permission on the destination table and the DESCRIBE permission on metadata of the source table. For more information, see Permissions.

Description

When you use MaxCompute SQL to process data, you need only to specify the names of partition key columns in `INSERT INTO` or `INSERT OVERWRITE`, instead of the values of the partition key columns. After you specify the values of the partition key columns in `select_statement`, MaxCompute automatically inserts data into the destination partitions based on the column values.

For more information about how to insert data into a static partition, see Insert or update data into a table or a static partition (INSERT INTO and INSERT OVERWRITE).

Limits

If you insert data into a dynamic partition by using `INSERT INTO` or `INSERT OVERWRITE`, take note of the following limits:

- A maximum of 10,000 dynamic partitions can be generated after `INSERT INTO` is executed. A maximum of 60,000 dynamic partitions can be generated after `INSERT OVERWRITE` is executed.
- In a distributed environment, an SQL statement used to insert or update data into dynamic partitions can generate a maximum of 512 dynamic partitions. If the number of dynamic partitions exceeds this limit, an exception occurs.
- The values of the dynamic partitions cannot be NULL, and cannot contain special characters. Otherwise, the following error is reported: `FAILED: ODPS-0123031:Partition exception - invalid dynamic partition value: province=xxx`.
- Clustered tables do not support dynamic partitions.

Usage notes

If you want to update table data into a dynamic partition, take note of the following points:

- If you want to use INSERT INTO or INSERT OVERWRITE to `insert data into a partition` that does not exist, MaxCompute automatically creates a partition.
- If you want to run multiple jobs at the same time to `insert data into partitions` that do not exist, MaxCompute automatically creates partitions for the first job that is successfully executed. However, only one partition is created for this job.
- If you cannot control `job concurrency`, we recommend that you run the `ALTER TABLE` command to create partitions in advance. For more information, see Partition and column operations.
- If a destination table has multiple levels of partitions, you can specify some partitions as static partitions in an `INSERT` statement. However, the static partitions must be high-level partitions.
- To insert data into a dynamic partition, you must specify partition key columns in `select_statement`. Otherwise, the data fails to be inserted.

Syntax

```
insert {into|overwrite} table <table_name> partition (<ptcol_name>[, <ptcol_name> ...])
<select_statement> from <from_statement>;
```

- table_name: required. The name of the destination table into which you want to insert data.
- ptcol_name: required. The name of the partition key column in the destination table.
- select_statement: required. The `SELECT` clause that is used to query the data that you want insert into the destination table from the source table.

If the destination table has only level-1 dynamic partitions, the value of the last field in `select_statement` indicates the value of the dynamic partition in the destination table. The mappings between the column values in the source table in `select_statement` and the column values in the destination table are determined by the column sequence, instead of column names. If the sequence of columns in the source table is different from that in the destination table, we recommend that you specify columns in the select_statement based on the column sequence in the destination table.

- from_statement: required. The `FROM` clause that indicates the data source, such as a source table name.

Examples

- Example 1: Insert data from a source table into a destination table. You can obtain the partitions generated based on the region column only after the statement is executed. The following statements show an example:

```
-- Create a destination table named total_revenues.
create table total_revenues (revenue double) partitioned by (region string);
-- Insert the data from the sale_detail table into the total_revenues table. For more inf
ormation about the sale_detail table, see Insert or update data into a table or a static
partition (INSERT INTO and INSERT OVERWRITE).
insert overwrite table total_revenues partition(region)
select total_price as revenue, region from sale_detail;
-- Execute the SHOW PARTITIONS statement to view the partitions in the total_revenues tab
le.
show partitions total_revenues;
-- The following result is returned:
region=china
-- Enable a full table scan only for the current session. Execute the SELECT statement to
query data from the total_revenues table.
set odps.sql.allow.fullscan=true;
select * from total_revenues;
-- The following result is returned:
+------------+------------+
| revenue    | region     |
+------------+------------+
| 100.1      | china      |
| 100.2      | china      |
| 100.3      | china      |
+------------+------------+
```

- Example 2: Insert data from a source table into a destination table. If the destination table has multiple levels of partitions, the level-1 partition sale_date must be specified. The following statements show an example:

```
-- Create a destination table named sale_detail_dypart.
create table sale_detail_dypart like sale_detail;
-- Specify a level-1 partition and insert data into the destination table.
insert overwrite table sale_detail_dypart partition (sale_date='2013', region)
select shop_name,customer_id,total_price,region from sale_detail;
-- Enable a full table scan only for the current session. Execute the SELECT statement to
query data from the sale_detail_dypart table.
set odps.sql.allow.fullscan=true;
select * from sale_detail_dypart;
-- The following result is returned:
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 3: The mappings between the columns in select_statement and the columns in dynamic partitions in a destination table are determined by the column sequence, instead of the column names. The following statements show an example:

```
-- Insert data from the sale_detail table into the sale_detail_dypart table.
insert overwrite table sale_detail_dypart partition (sale_date, region)
select shop_name,customer_id,total_price,sale_date,region from sale_detail;
-- Enable a full table scan only for the current session. Execute the SELECT statement to
query data from the sale_detail_dypart table.
set odps.sql.allow.fullscan=true;
select * from sale_detail_dypart;
-- The following result is returned: The sale_date column in the dynamic partition of the
sale_detail_dypart table is determined by the sale_date column in the sale_detail table.
The region column in the dynamic partition of the sale_detail_dypart table is determined
by the region column in the sale_detail table.
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
-- Insert data from the sale_detail table into the sale_detail_dypart table and change th
e sequence of columns in select_statement.
insert overwrite table sale_detail_dypart partition (sale_date, region)
select shop_name,customer_id,total_price,region,sale_date from sale_detail;
-- Enable a full table scan only for the current session. Execute the SELECT statement to
query data from the sale_detail_dypart table.
set odps.sql.allow.fullscan=true;
select * from sale_detail_dypart;
-- The following result is returned: The sale_date column in the dynamic partition of the
sale_detail_dypart table is determined by the region column in the sale_detail table. The
region column in the dynamic partition of the sale_detail_dypart table is determined by t
he sale_date column in the sale_detail table.
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | china      | 2013       |
| s2         | c2          | 100.2       | china      | 2013       |
| s3         | c3          | 100.3       | china      | 2013       |
+------------+-------------+-------------+------------+------------+
```

- Example 4: If you insert data into a dynamic partition, you must specify the columns in the dynamic partition in `select_statement`. Otherwise, the data fails to be inserted. Example of incorrect usage:

```
insert overwrite table sale_detail_dypart partition (sale_date='2013', region)
select shop_name,customer_id,total_price from sale_detail;
```

- Example 5: If you specify only low-level sub-partitions when you insert data into dynamic partitions, you may fail to insert data into high-level partitions. Example of incorrect usage:

```
insert overwrite table sale_detail_dypart partition (region='china', sale_date)
select shop_name,customer_id,total_price,sale_date from sale_detail;
```

- Example 6: If the data type of a partition key column does not exactly match the data type of the column in `select_statement`, an implicit conversion is performed when MaxCompute inserts data into a dynamic partition. The following statements show an example:

```
-- Create a source table named src.
create table src (c int, d string) partitioned by (e int);
-- Add a partition to the src table.
alter table src add if not exists partition (e=201312);
-- Append data to the src table.
insert into src partition (e=201312) values (1,100.1),(2,100.2),(3,100.3);
-- Create a destination table named parttable.
create table parttable(a int, b double) partitioned by (p string);
-- Insert data from the src table into the parttable table.
insert into parttable partition (p) select c, d, current_timestamp() from src;
-- Query data in the parttable table.
select * from parttable;
-- The following result is returned:
+------------+------------+------------+
| a          | b          | p          |
+------------+------------+------------+
| 1          | 100.1      | 2020-11-25 15:13:28.686 |
| 2          | 100.2      | 2020-11-25 15:13:28.686 |
| 3          | 100.3      | 2020-11-25 15:13:28.686 |
+------------+------------+------------+
```

Note If your data is ordered, it is randomly scattered when the data is inserted into a dynamic partition. This reduces the data compression ratio. In this case, we recommend that you use Tunnel commands to upload the data to dynamic partitions to increase the data compression ratio. For more information about how to use Tunnel commands, see Migrate data from ApsaraDB RDS to MaxCompute based on dynamic partitioning.

3.5.3. UPDATE and DELETE

MaxCompute allows you to execute the `DELETE` or `UPDATE` statement to delete or update data of specific rows in transactional tables.

You can execute the INSERT INTO or INSERT OVERWRITE statement on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Prerequisites

You are granted the `SELECT` and `UPDATE` permissions to read data from and update data in a transactional table. For more information, see Permissions.

Description

Similar to traditional SQL statements, the `DELETE` and `UPDATE` statements in MaxCompute can be used to delete or update the data of specific rows in tables.

Each time you execute the `DELETE` or `UPDATE` statement, a delta file is automatically generated to store information about the delete or update operation. This file is invisible to users. The following section describes how the delta files are generated.

- `DELETE`: A delta file contains the `txnid` and `rowid` fields, both of which have a value of the BIGINT type. The rowid field indicates the deleted row in the base files of the transactional table. The txnid field indicates the delete operation that is performed on the row.

  For example, a base file of the t1 table is f1 and the content of the base file is `a, b, c, a, b`. When you execute the `delete from t1 where c1='a';` statement, a delta file named `f1.delta` is generated. If the value of the `txnid` field is `t0`, the content of the `f1.delta` file is `((0, t0), (3, t0))`. This indicates that the rows whose IDs are 0 and 3 are deleted from the t0 transaction. If you execute another `DELETE` statement on the t1 table again, another delta file named `f2.delta` is generated. The file name is generated based on the f1 base file. When you query the data in the t1 table, the system filters the deleted data based on the f1, f1.delta, and f2.delta files and returns the data that is not deleted.

- `UPDATE`: The logic of an `UPDATE` statement is converted into the logic of executing the `DELETE` and `INSERT INTO` statements.

The `DELETE` and `UPDATE` statements have the following benefits:

- Reduce the amount of data to be written

  Before the DELETE and UPDATE statements are provided, MaxCompute allows you to execute only the `INSERT INTO` or `INSERT OVERWRITE` statement to update or delete data in tables. For more information, see Insert or update data into a table or a static partition (INSERT INTO and INSERT OVERWRITE). If you want to execute an `INSERT` statement when you update a small amount of data in a table or a partition of the table, you must first execute a `SELECT` statement to read all data from the table and update the data. Then, you can execute the `INSERT` statement to insert all data into the table. This method is inefficient. However, if you use the `DELETE` or `UPDATE` statement in the preceding scenario, the system does not need to write all data in the table. This reduces the amount of data to be written.

  > **Note**
  > - If you use the pay-as-you-go billing method, you are not charged for the write operations when you execute the `DELETE`, `UPDATE`, or `INSERT OVERWRITE` statement. However, when you execute the `DELETE` or `UPDATE` statement, MaxCompute must filter data by partition and read the data that you want to delete or update. You are charged for the read operations based on the pay-as-you-go billing method of SQL jobs. Therefore, compared with the `INSERT OVERWRITE` statement, the `DELETE` or `UPDATE` statement does not help you reduce costs.
  > - If you use the subscription billing method, fewer resources are consumed to write data when you execute the `DELETE` or `UPDATE` statement. Compared with the `INSERT OVERWRITE` statement, the DELETE or UPDATE statement allows you to run more jobs when the same amount of resources is used.

- Read the table with the latest data

Before the DELETE and UPDATE statements are provided, MaxCompute allows you to use history tables to update multiple data entries in a table. If you use a history table, you must add auxiliary columns such as `start_date` and `end_date` in the table. These columns indicate the lifecycle of a data entry. To query the latest data of a table, the system must identify the latest data from large amounts of data based on the timestamps. This is a time-consuming process. However, you can execute the `DELETE` or `UPDATE` statement to delete or update data. When you query the data in a table, the system reads the latest data of the table based on the base files and all delta files.

> **Notice** After you execute the `DELETE` and `UPDATE` statements multiple times on a transactional table, the transactional table occupies a larger storage space. In this case, the costs of the storage and subsequent queries on the table increase. In addition, the efficiency of subsequent queries is reduced. To resolve these issues, we recommend that you execute the ALTER TABLE COMPACT statement to merge the base files with all delta files on a regular basis. For more information, see ALTER TABLE COMPACT.

If multiple jobs are run in parallel on a table, conflicts may occur. For more information, see ACID semantics.

Scenarios

You can execute the `DELETE` or `UPDATE` statement to delete or update a small amount of data in tables or partitions of the tables at a low frequency. For example, delete or update less than 5% of the data in a table or a partition of the table on the next day after the data is generated.

The `DELETE` or `UPDATE` statement is not applicable if you want to delete or update data at a high frequency, or if you want to write data to tables in real time.

Limits

The `DELETE` statement, `UPDATE` statement, and transactional tables on which the DELETE or UPDATE statement is executed have the following limits:

- You can execute the DELETE and UPDATE statements only on transactional tables. For more information about transactional tables, see Table operations.
- When you create a clustered table or an external table, you cannot set the clustered table or external table to a transactional table.
- You cannot convert between transactional tables and MaxCompute internal tables, external tables, or clustered tables.
- Jobs from other systems, such as MaxCompute Spark, Machine Learning Platform for AI, and Graph, cannot access transactional tables.
- `CLONE TABLE` and `MERGE PARTITION` operations are not supported.
- Before you execute the `UPDATE`, `DELETE`, or `INSERT OVERWRITE` statement for important data in transactional tables, you must execute the `SELECT` and `INSERT` statements to back up the data to other tables.

Usage notes

When you execute the `DELETE` or `UPDATE` statement to delete or update data in tables or partitions of the tables, take note of the following items:

- In specific scenarios, you may want to execute the DELETE or UPDATE statement for a small amount of data in a table and infrequently perform read and other operations in subsequent procedures. To

reduce the storage space that is occupied by the table, we recommend that you merge the base files with all delta files after you execute the `DELETE` or `UPDATE` statement for the table several times. For more information, see ALTER TABLE COMPACT.

- In specific scenarios, you may want to delete or update more than 5% of the data in a table or a partition of the table at a low frequency and perform frequent subsequent queries. We recommend that you execute the `INSERT OVERWRITE` or `INSERT INTO` statement in such scenarios. For more information, see Insert or update data into a table or a static partition (INSERT INTO and INSERT OVERWRITE).

  For example, you want to perform delete or update operations 10 times each day and delete or update 10% of the data in a table each time. In this case, we recommend that you estimate the total cost and the consumption of the subsequent read performance if you execute the `DELETE` or `UPDATE` statement on the table. Then, compare the estimated result with that of executing the `INSERT OVERWRITE` or `INSERT INTO` statement. This helps you choose an efficient method.

- Each time you execute the DELETE statement on a table, a delta file is automatically generated. As a result, the occupied storage space may not be reduced. If you want to execute the `DELETE` statement to delete data to reduce storage usage, you can merge the base files with all delta files. For more information, see ALTER TABLE COMPACT.
- MaxCompute executes multiple `DELETE` and `UPDATE` statements in jobs at a time. Each statement consumes resources and incurs fees. We recommend that you delete or update a batch of data at a time. For example, if you run a Python script to generate and submit a large number of row-level update jobs, and each statement executes on only one row or a small number of rows of data, each statement incurs fees that correspond to the amount of input data scanned by the SQL statement and consumes the related computing resources. When multiple statements are accumulated, the costs are significantly increased and the system efficiency is reduced. Examples:

```
-- We recommend that you execute the following statement:
update table1 set col1= (select value1 from table2 where table1.id = table2.id and table1
.region = table2.region);
-- We recommend that you do not execute the following statements:
update table1 set col1=1 where id='2021063001'and region='beijing';
update table1 set col1=2 where id='2021063002'and region='beijing';
```

DELETE

You can execute the `DELETE` statement to delete one or more rows that meet the specified conditions in partitioned transactional tables or non-partitioned transactional tables.

- Syntax

```
delete from <table_name> [where <where_condition>];
```

- Parameters
  - table_name: required. The name of the transactional table on which you want to execute the `DELETE` statement.
  - where_condition: optional. A WHERE clause that is used to filter data based on conditions. For more information, see WHERE clause (where_condition). If you execute the DELETE statement on a table without a WHERE clause, all data in the table is deleted.
- Examples

- Example 1: Create a non-partitioned transactional table named acid_delete and insert data into the table. Then, execute the `DELETE` statement to delete the rows that meet the specified conditions from the table. The following statements show an example:

```
-- Create a non-partitioned transactional table named acid_delete.
create table if not exists acid_delete(id bigint) tblproperties ("transactional"="true"
);
-- Insert data into the table.
insert overwrite table acid_delete values(1),(2),(3),(2);
-- Query the table to check whether data is inserted.
select * from acid_delete;
+------------+
| id         |
+------------+
| 1          |
| 2          |
| 3          |
| 2          |
+------------+
-- Delete the rows whose value of the id column is 2. If you want to execute the statem
ent on the MaxCompute client (odpscmd), you must enter yes or no to confirm the deletio
n.
delete from acid_delete where id = 2;
-- Query the table to check whether the table contains only the rows whose values of th
e id column are 1 and 3.
select * from acid_delete;
+------------+
| id         |
+------------+
| 1          |
| 3          |
+------------+
```

- Example 2: Create a partitioned transactional table named acid_delete_pt and insert data into the table. Then, execute the `DELETE` statement to delete the rows that meet the specified conditions from the table. The following statements show an example:

```
-- Create a partitioned transactional table named acid_delete_pt.
create table if not exists acid_delete_pt(id bigint) partitioned by(ds string) tblprope
rties ("transactional"="true");
-- Add partitions to the table.
alter table acid_delete_pt add if not exists partition (ds= '2019');
alter table acid_delete_pt add if not exists partition (ds= '2018');
-- Insert data into the table.
insert overwrite table acid_delete_pt partition (ds='2019') values(1),(2),(3);
insert overwrite table acid_delete_pt partition (ds='2018') values(1),(2),(3);
-- Query the table to check whether data is inserted.
select * from acid_delete_pt;
+------------+------------+
| id         | ds         |
+------------+------------+
| 1          | 2018       |
| 2          | 2018       |
| 3          | 2018       |
| 1          | 2019       |
| 2          | 2019       |
| 3          | 2019       |
+------------+------------+
-- Delete the rows whose values of the id and ds columns are 2 and 2019. If you want to
execute the statement on the MaxCompute client (odpscmd), you must enter yes or no to c
onfirm the deletion.
delete from acid_delete_pt where ds='2019' and id = 2;
-- Query the table to check whether the rows whose values of the id and ds columns are
2 and 2019 are deleted.
select * from acid_delete_pt;
+------------+------------+
| id         | ds         |
+------------+------------+
| 1          | 2018       |
| 2          | 2018       |
| 3          | 2018       |
| 1          | 2019       |
| 3          | 2019       |
+------------+------------+
```

- Example 3: Create a destination table named acid_delete_t and an associated table named acid_delete_s. Then, delete the rows that meet the specified conditions from the destination table based on the associated table. The following statements show an example:

```
-- Create a destination table named acid_delete_t and an associated table named acid_de
lete_s.
create table if not exists acid_delete_t(id int,value1 int,value2 int) tblproperties ("
transactional"="true");
create table if not exists acid_delete_s(id int,value1 int,value2 int);
-- Insert data into the tables.
insert overwrite table acid_delete_t values(2,20,21),(3,30,31),(4,40,41);
insert overwrite table acid_delete_s values(1,100,101),(2,200,201),(3,300,301);
-- Delete the rows in the acid_delete_t table whose value of the id column does not mat
ch that of the rows in the acid_delete_s table. If you want to execute the statement on
the MaxCompute client (odpscmd), you must enter yes or no to confirm the deletion.
delete from acid_delete_t where not exists (select * from acid_delete_s where acid_dele
te_t.id=acid_delete_s.id);
-- Query the acid_delete_t table to check whether the table contains only the rows whos
e values of the id column are 2 and 3.
select * from acid_delete_t;
+------------+------------+------------+
| id         | value1     | value2     |
+------------+------------+------------+
| 2          | 20         | 21         |
| 3          | 30         | 31         |
+------------+------------+------------+
```

UPDATE

You can execute the `UPDATE` statement to update the values of one or more columns of the rows that meet the specified conditions in partitioned transactional tables or non-partitioned transactional tables.

- Syntax

```
update <table_name> set <col1_name> = <value1> [, <col2_name> = <value2> ...] [where <whe
re_condition>];
update <table_name> set (<col1_name> [, <col2_name> ...]) = (<value1> [, <value2> ...])[w
here <where_condition>];
```

- Parameters
  - table_name: required. The name of the transactional table on which you want to execute the `UPDATE` statement.
  - col1_name and col2_name: the columns that you want to update. You must specify at least one column.
  - value1 and value2: the new values that you want to assign to the specified columns. You must update the value of at least one column.
  - where_condition: optional. A WHERE clause that is used to filter data based on conditions. For more information, see WHERE clause (where_condition). If you execute the UPDATE statement on a table without a WHERE clause, all data in the table is updated.
- Examples

- Example 1: Create a non-partitioned table named acid_update and insert data into the table. Then, execute the `UPDATE` statement to update the columns of the rows that meet the specified conditions in the table. The following statements show an example:

```
-- Create a non-partitioned table named acid_update.
create table if not exists acid_update(id bigint) tblproperties ("transactional"="true"
);
-- Insert data into the table.
insert overwrite table acid_update values(1),(2),(3),(2);
-- Query the table to check whether data is inserted.
select * from acid_update;
+------------+
| id         |
+------------+
| 1          |
| 2          |
| 3          |
| 2          |
+------------+
-- Update the value 2 in the id column to 4.
update acid_update set id = 4 where id = 2;
-- Query the table to check whether the value 2 is changed to 4 in the id column.
select * from acid_update;
+------------+
| id         |
+------------+
| 1          |
| 3          |
| 4          |
| 4          |
+------------+
```

- Example 2: Create a partitioned table named acid_update and insert data into the table. Then, execute the `UPDATE` statement to update the columns of the rows that meet the specified conditions in the table. The following statements show an example:

```
-- Create a partitioned table named acid_update_pt.
create table if not exists acid_update_pt(id bigint) partitioned by(ds string) tblprope
rties ("transactional"="true");
-- Add a partition to the table.
alter table acid_update_pt add if not exists partition (ds= '2019');
-- Insert data into the added partition.
insert overwrite table acid_update_pt partition (ds='2019') values(1),(2),(3);
-- Query the partition to check whether data is inserted.
select * from acid_update_pt where ds = '2019';
+------------+------------+
| id         | ds         |
+------------+------------+
| 1          | 2019       |
| 2          | 2019       |
| 3          | 2019       |
+------------+------------+
-- Update the value 2 of the id column in the 2019 partition to 4.
update acid_update_pt set id = 4 where ds = '2019' and id = 2;
-- Query the partition to check whether the value 2 is changed to 4 in the id column.
select * from acid_update_pt where ds = '2019';
+------------+------------+
| id         | ds         |
+------------+------------+
| 4          | 2019       |
| 1          | 2019       |
| 3          | 2019       |
+------------+------------+
```

- Example 3: Create a transactional table named acid_update_t that you want to update and an associated table named acid_update_s. Then, update the values of multiple columns at a time in the acid_update_t table. The following statements show an example:

```
-- Create a transactional table named acid_update_t that you want to update and an asso
ciated table named acid_update_s.
create table if not exists acid_update_t(id int,value1 int,value2 int) tblproperties ("
transactional"="true");
create table if not exists acid_update_s(id int,value1 int,value2 int);
-- Insert data into the tables.
insert overwrite table acid_update_t values(2,20,21),(3,30,31),(4,40,41);
insert overwrite table acid_update_s values(1,100,101),(2,200,201),(3,300,301);
--Method 1: Update the values of specific columns with constants.
update acid_update_t set (value1, value2) = (60,61);
-- Query the acid_update_t table to check whether data is updated as expected.
select * from acid_update_t;
+------------+------------+------------+
| id         | value1     | value2     |
+------------+------------+------------+
| 2          | 60         | 61         |
| 3          | 60         | 61         |
| 4          | 60         | 61         |
+------------+------------+------------+
```

```
--Method 2: Use the data in the acid_update_s table to update all rows in the acid_upda
te_t table. If specific rows in the acid_update_t table cannot be matched, null values
are assigned to the rows.
update acid_update_t set (value1, value2) = (select value1, value2 from acid_update_s w
here acid_update_t.id = acid_update_s.id);
-- Query the acid_update_t table to check whether data is updated as expected.
select * from acid_update_t;
+------------+------------+------------+
| id         | value1     | value2     |
+------------+------------+------------+
| 2          | 200        | 201        |
| 3          | 300        | 301        |
| 4          | NULL       | NULL       |
+------------+------------+------------+
-- Method 3: Use the data in the acid_update_s table to update the acid_update_t table.
Add filter conditions for the acid_update_t table to update only the rows that intersec
t with those in the acid_update_s table.
update acid_update_t set (value1, value2) = (select value1, value2 from acid_update_s w
here acid_update_t.id = acid_update_s.id) where acid_update_t.id in (select id from aci
d_update_s);
-- Query the acid_update_t table to check whether data is updated as expected.
select * from acid_update_t;
+------------+------------+------------+
| id         | value1     | value2     |
+------------+------------+------------+
| 4          | 40         | 41         |
| 2          | 200        | 201        |
| 3          | 300        | 301        |
+------------+------------+------------+
-- Method 4: Use the aggregate results of the acid_update_t and acid_update_s tables to
update the acid_update_t table.
update acid_update_t set (id, value1, value2) = (select id, max(value1),max(value2) fro
m acid_update_s where acid_update_t.id = acid_update_s.id group by acid_update_s.id) wh
ere acid_update_t.id in (select id from acid_update_s);
-- Query the acid_update_t table to check whether data is updated as expected.
select * from acid_update_t;
+------------+------------+------------+
| id         | value1     | value2     |
+------------+------------+------------+
| 4          | 40         | 41         |
| 2          | 200        | 201        |
| 3          | 300        | 301        |
+------------+------------+------------+
```

ALTER TABLE COMPACT

The base files and delta files of a transactional table occupy the physical storage. You cannot directly read such files. If you execute the `UPDATE` or `DELETE` statement on a transactional table, the data in the base files is not updated, but a delta file is generated for each operation. In this case, the more delete or update operations are performed, the more storage space the table occupies. The number of delta files increases. As a result, you are charged more for storage usage and subsequent queries.

If you execute the `UPDATE` or `DELETE` statement on a table or a partition of the table multiple times, a large number of delta files are generated. When the system reads data from the table, the system loads the delta files to identify the deleted and updated rows. A large number of delta files reduce the efficiency of reading data. In this case, you can merge the base files with the delta files to reduce storage usage and improve the read efficiency.

- Syntax

```
alter table <table_name> [partition (<partition_key> = '<partition_value>' [, ...])] comp
act {minor|major};
```

- Parameters
  - table_name: required. The name of the transactional table for which you want to merge the base files and delta files.
  - partition_key: optional. The name of a partition key column in the partitioned transactional table.
  - partition_value: optional. The value of a partition key column in the partitioned transactional table.
  - major|minor: One of them must be specified. Differences between minor compaction and major compaction:
    - `minor`: merges each base file with all delta files that are generated based on the base file and deletes the delta files.
    - `major`: merges each base file with all delta files that are generated based on the base file, deletes the delta files, and merges small base files. If the size of a base file is less than 32 MB or a delta file is generated, the effect of merging files is equivalent to the effect of executing the `INSERT OVERWRITE` statement. However, if the size of a base file is greater than or equal to 32 MB and no delta files are generated, the data of the table is not overwritten.
- Examples
  - Example 1: Merge files of the acid_delete table. The following statement shows an example:

```
alter table acid_delete compact minor;
```

    The following information is returned:

```
Summary:
Nothing found to merge, set odps.merge.cross.paths=true if cross path merge is permitte
d.
OK
```

  - Example 2: Merge files of the acid_update_pt table. The following statement shows an example:

```
alter table acid_update_pt partition (ds = '2019') compact major;
```

    The following information is returned:

```
Summary:
table name: acid_update_pt /ds=2019  instance count: 2  run time: 6
  before merge, file count:        8  file size: 2613  file physical size: 7839
   after merge, file count:        2  file size: 679  file physical size: 2037
OK
```

FAQ

- Issue 1:
  - Problem description: When I execute the `UPDATE` statement, the following error message is returned: `ODPS-0010000:System internal error - fuxi job failed, caused by: Data Set should contain exactly one row`.
  - Cause: The rows that you want to update do not match the rows that are queried by subqueries. In this case, the system cannot determine which rows need to be updated. The following statement shows an example:

    ```
    update store set (s_county, s_manager) = (select d_country, d_manager from store_delta sd where sd.s_store_sk = store.s_store_sk) where s_store_sk in (select s_store_sk from store_delta);
    ```

    The `select d_country, d_manager from store_delta sd where sd.s_store_sk = store.s_store_sk` subquery is used to filter the data in the store_delta table. Then, the data that meets the specified condition is used to update the data of the store table. For example, three rows that have the s_store_sk column in the store table are `[1, 2, 3]`. If the rows that have the s_store_sk column in the store_delta table are `[1, 1]` and do not match the rows in the store table, the preceding error message is returned.
  - Solution: Make sure that the rows that you want to update exactly match the rows that are queried by subqueries.
- Issue 2:
  - Problem description: When I run the `compact` command in DataWorks DataStudio, the following error message is returned: `ODPS-0130161:[1,39] Parse exception - invalid token 'minor', expect one of 'StringLiteral','DoubleQuoteStringLiteral'`.
  - Cause: The MaxCompute client version that corresponds to the exclusive resource group of DataWorks does not support the `compact` command.
  - Solution: Submit a ticket to contact the DataWorks technical support team to update the MaxCompute client version that corresponds to the exclusive resource group.

3.5.4. MERGE INTO

If you want to perform the `INSERT`, `UPDATE`, and `DELETE` operations on a transactional table, you can encapsulate the operations in one `MERGE INTO` statement. Then, you can execute this statement to perform multiple operations on the table based on the join condition with a source table. This feature improves efficiency. The `MERGE INTO` operation is in public preview. You are not charged for executing the `MERGE INTO` statement on transactional tables. However, you are charged for executing other query statements on transactional tables. In public preview, Alibaba Cloud does not ensure the usage of this feature in production environments. We recommend that you back up your data in advance.

The Select and Update permissions to read data from and update data in a transactional table are granted. For more information, see Permissions.

Feature description

MaxCompute allows you to execute the `DELETE` and `UPDATE` statements on a table to delete data from or update data in the table. If you want to perform multiple operations including `INSERT`, `UPDATE`, and `DELETE` on the table at a time, you must write and execute a statement for each operation. In this case, multiple full table scans are required. To increase efficiency, MaxCompute allows you to execute a `MERGE INTO` statement to perform multiple operations on a table at a time. In this case, only one full table scan is required. This method is more efficient than separately executing the `INSERT`, `UPDATE`, and `DELETE` statements.

The `MERGE INTO` statement ensures the atomicity of multiple operations. A job is successful only after all the `INSERT`, `UPDATE`, and `DELETE` operations in the job are successful. If an operation fails, the job also fails.

If you separately execute the `INSERT`, `UPDATE`, and `DELETE` statements, specific operations may fail. The data on which successful operations are performed cannot be restored However, the `MERGE INTO` statement helps you prevent this issue.

Limits

The `MERGE INTO` statement has the following limits:

- You can execute the MERGE INTO statement only on transactional tables. For more information about transactional tables, see Table operations.
- You cannot perform multiple `INSERT` or `UPDATE` operations on the same rows in a table by using a single `MERGE INTO` statement.

Syntax

```
merge into <target_table> as <alias_name_t> using <source expression|table_name> as <alias_
name_s>
-- The ON clause specifies the JOIN conditions of the source table and the destination tabl
e.
on <boolean expression1>
-- The WHEN MATCHED...THEN clause specifies the operation to be performed when the result o
f the ON clause is True. The operations of multiple WHEN MATCHED...THEN clauses cannot be p
erformed on the same data.
when matched [and <boolean expression2>] then update set <set_clause_list>
when matched [and <boolean expression3>] then delete
-- The WHEN MATCHED...THEN clause specifies the operation to be performed when the result o
f the ON clause is False.
when not matched [and <boolean expression4>] then insert values <value_list>
```

- target_table: required. Specify the name of the destination table, which must be an existing table.
- alias_name_t: required. Specify the alias of the destination table.
- source expression|table_name: required. Specify the name of the source table, view, or subquery that you want to join with the destination table.
- alias_name_s: required. Specify the alias of the source table, view, or subquery that you want to join with the destination table.
- boolean expression1: required. Specify a condition that returns a value of the BOOLEAN type. The value must be True or False.
- boolean expression2, boolean expression3, and boolean expression4: optional. You can specify a condition for each of the `UPDATE`, `DELETE`, and `INSERT` operations that you want to perform.

The condition must return a value of the BOOLEAN type. Take note of the following items:

- If a MERGE INTO statement has three WHEN clauses, each of the `UPDATE`, `DELETE`, and `INSERT` operations can be used only once.
- If both the `UPDATE` and `DELETE` operations are included in a MERGE INTO statement, you must specify a `[and <boolean expression>]` condition for the operation that needs to be performed first.
- The `WHEN NOT MATCHED` clause can be used only as the last WHEN clause and supports only the `INSERT` operation.

- set_clause_list: required when you use the `UPDATE` operation. Specify the data that you want to update. For more information about the `UPDATE` operation, see UPDATE.
- value_list: required when you use the `INSERT` operation. Specify the data that you want to insert. For more information about the `VALUES` statement, see VALUES.

Examples

Create a destination table named acid_address_book_base1 and a source table named exists tmp_table1, and insert data into the tables. Execute the `MERGE INTO` statement on the destination table. The data entries in the source table that meet the specified `ON` joint condition are used to update the joined data entries in the destination table. The data entries that do not meet the specified `ON` join condition and whose value of the event_type column is I are inserted into the destination table. The following statements show an example:

```
-- Create a destination table named acid_address_book_base1.
create table if not exists acid_address_book_base1
(id bigint,first_name string,last_name string,phone string)
partitioned by(year string, month string, day string, hour string)
tblproperties ("transactional"="true");
-- Create a source table named exists tmp_table1.
create table if not exists tmp_table1
(id bigint, first_name string, last_name string, phone string, _event_type_ string);
-- Insert data into the acid_address_book_base1 table.
insert overwrite table acid_address_book_base1
partition(year='2020', month='08', day='20', hour='16')
values (4, 'nihaho', 'li', '222'), (5, 'tahao', 'ha', '333'),
(7, 'djh', 'hahh', '555');
-- Insert data into the exists tmp_table1 table.
insert overwrite table tmp_table1 values
(1, 'hh', 'liu', '999', 'I'), (2, 'cc', 'zhang', '888', 'I'),
(3, 'cy', 'zhang', '666', 'I'),(4, 'hh', 'liu', '999', 'U'),
(5, 'cc', 'zhang', '888', 'U'),(6, 'cy', 'zhang', '666', 'U');
-- Execute the MERGE INTO statement.
merge into acid_address_book_base1 as t using tmp_table1 as s
on s.id = t.id and t.year='2020' and t.month='08' and t.day='20' and t.hour='16'
when matched then update set t.first_name = s.first_name, t.last_name = s.last_name, t.phon
e = s.phone
when not matched and (s._event_type_='I') then insert values(s.id, s.first_name, s.last_nam
e,s.phone,'2020','08','20','16');
-- Query the acid_address_book_base1 table to check the result of the MERGE INTO operation.
select * from acid_address_book_base1;
+------------+------------+------------+------------+------------+------------+------------
+------------+
| id         | first_name | last_name  | phone      | year       | month      | day
| hour       |
+------------+------------+------------+------------+------------+------------+------------
+------------+
| 4          | hh         | liu        | 999        | 2020       | 08         | 20
| 16         |
| 5          | cc         | zhang      | 888        | 2020       | 08         | 20
| 16         |
| 7          | djh        | hahh       | 555        | 2020       | 08         | 20
| 16         |
| 1          | hh         | liu        | 999        | 2020       | 08         | 20
| 16         |
| 2          | cc         | zhang      | 888        | 2020       | 08         | 20
| 16         |
| 3          | cy         | zhang      | 666        | 2020       | 08         | 20
| 16         |
+------------+------------+------------+------------+------------+------------+------------
+------------+
```

3.5.5. MULTI INSERT

MaxCompute SQL allows you to insert data into different destination tables or partitions at the same time by using a MULTI INSERT statement. If you execute a MULTI INSERT statement, multiple `INSERT INTO` or `INSERT OVERWRITE` operations are performed at the same time.

You can execute the INSERT INTO or INSERT OVERWRITE statement on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Prerequisites

You have the ALTER permission on the destination tables and the DESCRIBE permission on metadata in the source table. For more information about how to grant the permissions, see Permissions.

Description

To use MaxCompute SQL to process data, you can execute a `MULTI INSERT` statement to insert data into different destination tables or partitions at the same time.

Limits

When you execute a `MULTI INSERT` statement, take note of the following limits:

- A single `MULTI INSERT` statement can contain up to 255 INSERT operations. If the number of INSERT operations exceeds 255, a syntax error is returned.
- In a `MULTI INSERT` statement, you cannot specify the same destination partition in a partitioned table for multiple INSERT operations.
- In a `MULTI INSERT` statement, you cannot specify the same non-partitioned table for multiple INSERT operations.

Syntax

```
from <from_statement>
insert overwrite | into table <table_name1> [partition (<pt_spec1>)]
<select_statement1>
insert overwrite | into table <table_name2> [partition (<pt_spec2>)]
<select_statement2>
...;
```

- from_statement: required. The `FROM` clause that specifies the data source. For example, you can specify the name of a source table in this clause.
- table_name: required. The names of the tables into which you want to insert data.
- pt_spec: optional. The partitions into which you want to insert data. Only constants are allowed. Expressions, such as functions, are not allowed. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. If you want to insert data into two partitions that are specified by pt_spec1 and pt_spec2, the partition information of pt_spec1 and pt_spec2 must be different.
- select_statement: required. The `SELECT` clause that is used to query the data that you want to insert into the destination tables or partitions from the source table.

Examples

- Example 1: Insert data from the *sale_detail* table into the specified partitions of the *sale_detail_multi* table. The partitions store the sales records for the years 2010 and 2011 in the Chinese mainland. Sample statements:

```
-- Create a destination table named sale_detail_multi.
create table sale_detail_multi like sale_detail;
-- Enable a full table scan only for the current session. -- Insert data from the sale_de
tail table into the sale_detail_multi table.
set odps.sql.allow.fullscan=true;
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price
insert overwrite table sale_detail_multi partition (sale_date='2011', region='china' )
select shop_name, customer_id, total_price;
-- Enable a full table scan only for the current session. Execute the SELECT statement to
query the data in the sale_detail_multi table.
set odps.sql.allow.fullscan=true;
select * from sale_detail_multi;
-- The following result is returned:
+------------+------------+------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+------------+------------+------------+------------+
| s1         | c1          | 100.1       | 2010       | china      |
| s2         | c2          | 100.2       | 2010       | china      |
| s3         | c3          | 100.3       | 2010       | china      |
| s1         | c1          | 100.1       | 2011       | china      |
| s2         | c2          | 100.2       | 2011       | china      |
| s3         | c3          | 100.3       | 2011       | china      |
+------------+------------+------------+------------+------------+
```

- Example 2: If the same partition is specified for multiple INSERT operations in a single MULTI INSERT statement, an error is returned. **Sample statement of incorrect usage:**

```
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price;
```

3.5.6. VALUES

If you want to insert a small amount of data into a table that has limited data space, you can use `INSERT … VALUES` or `VALUES TABLE` to perform this operation.

Make sure that you have the ALTER permission on the destination table and the DESCRIBE permission on metadata in the source table before you perform the `INSERT INTO` operation. For more information, see Permissions.

You can execute the INSERT INTO or INSERT OVERWRITE statement on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console

- DataWorks console
- MaxCompute Studio

Operations

MaxCompute allows you to use the `INSERT … VALUES` or `VALUES TABLE` operation to insert a small amount of data into a table.

Operation	Description
`INSERT... VALUES`	In the business testing phase, you can use `INSERT … VALUES` to insert data into a table to perform a simple test. • To insert a few or a dozen of data records into the test table, you can use `INSERT … VALUES`. • To insert dozens of data records into the test table, you can use Tunnel commands to upload a *TXT* or *CSV* file that contains the data records. For more information, see Import data to tables.
`values table`	To perform simple computing operations on the inserted data, we recommend that you use `VALUES TABLE` of MaxCompute. `VALUES TABLE` can be used in `INSERT` statements and data manipulation language (DML) statements. Features: • If no physical tables are available, create a table that contains multiple rows of arbitrary data and perform computing operations on the table. • Create a constant table by using VALUES TABLE, instead of the combination of `SELECT * FROM` and `UNION ALL`. • `VALUES TABLE` allows you to execute the `SELECT` statement without the `FROM` clause. Data of other tables cannot be contained in the expressions of `SELECT`. The underlying implementation principle is to select data from an anonymous VALUES table that contains only one row and no columns. This way, you can test user-defined functions (UDFs) or other functions without the need to manually create a DUAL table.

Limits

When you use the `INSERT ... VALUES` or `VALUES TABLE` operation to insert data into tables, you cannot use `INSERT OVERWRITE` to specify the columns to which you want to insert data. Instead, you can use only `INSERT INTO`.

Syntax

```
--INSERT ... VALUES
insert into table <table_name>
[partition (<pt_spec>)][(<col1_name> ,<col2_name>,...)]
values (<col1_value>,<col2_value>,...),(<col1_value>,<col2_value>,...),...
--values table
values (<col1_value>,<col2_value>,...),(<col1_value>,<col2_value>,...),<table_name> (<col1_
name> ,<col2_name>,...)...
```

- table_name: required. The name of the table into which you want to insert data. The table must be an existing table.
- pt_spec: optional. The destination partition into which you want to insert data. The value is in the format of `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)`. If you want to update data into a partitioned table, you must specify this parameter.
- col_name: optional. The name of the destination column into which you want to insert data.
- col_name: optional. The value of the column in the destination table. If you specify multiple column values, separate them with commas (,). The column values can be constant and non-constant expressions, such as custom function expressions or built-in function expressions. If column values are not specified, the default value is NULL.

> **Note**
> - Complex data types, except ARRAY, cannot be used to construct constants. For more information about how to use the ARRAY data type in `VALUES`, see Example 3.
> - If you want to use `VALUES` to insert data of the DATETIME or TIMESTAMP type, specify the data type in `VALUES`. For more information, see Example 4.

Examples

- Example 1: Use `INSERT ... VALUES` to insert data into a specified partition. Sample statements:

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value bigint) partitioned by (p string);
-- Add the abc partition to the srcp table.
alter table srcp add if not exists partition (p='abc');
-- Insert data into the abc partition in the srcp table.
insert into table srcp partition (p='abc') values ('a',1),('b',2),('c',3);
-- Query data from the srcp table.
select * from srcp where p='abc';
-- Return result
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| a          | 1          | abc        |
| b          | 2          | abc        |
| c          | 3          | abc        |
+------------+------------+------------+
```

- Example 2: Use `INSERT ... VALUES` to insert data into a random partition. Sample statements:

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value bigint) partitioned by (p string);
-- Insert data into the srcp table without specifying a partition.
insert into table srcp partition (p)(key,p) values ('d','20170101'),('e','20170101'),('f'
,'20170101');
-- Query data from the srcp table.
select * from srcp where p='20170101';
-- Return result
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| d          | NULL       | 20170101   |
| e          | NULL       | 20170101   |
| f          | NULL       | 20170101   |
+------------+------------+------------+
```

- Example 3: Use complex data types to construct constants and import data by using `INSERT` operations. Sample statements:

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value array<int>) partitioned by (p string);
-- Add a partition to the srcp table.
alter table srcp add if not exists partition (p='abc');
-- Insert data into the abc partition of the srcp table.
insert into table srcp partition (p='abc') select 'a', array(1, 2, 3);
-- Query data from the srcp table.
select * from srcp where p='abc';
-- Return result
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| a          | [1,2,3]    | abc        |
+------------+------------+------------+
```

- Example 4: Perform the `INSERT … VALUES` operation to insert data of the DATETIME or TIMESTAMP type to the table and specify the data type in `values`. Sample statements:

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string, value timestamp) partitioned by (p string);
-- Add a partition to the srcp table.
alter table srcp add if not exists partition (p='abc');
-- Insert data into the abc partition in the srcp table.
insert into table srcp partition (p='abc') values (datetime'2017-11-11 00:00:00',timestam
p'2017-11-11 00:00:00.123456789');
-- Query data from the srcp table.
select * from srcp where p='abc';
-- Return result
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| 2017-11-11 00:00:00 | 2017-11-11 00:00:00.123 | abc        |
+------------+------------+------------+
```

- Example 5: Use `VALUES TABLE` to insert data into a partitioned table. Sample statements:

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value bigint) partitioned by (p string);
-- Insert data into the srcp table.
insert into table srcp partition (p) select concat(a,b), length(a)+length(b),'20170102' f
rom values ('d',4),('e',5),('f',6) t(a,b);
-- Query data from the srcp table.
select * from srcp where p='20170102';
-- Return result
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| d4         | 2          | 20170102   |
| e5         | 2          | 20170102   |
| f6         | 2          | 20170102   |
+------------+------------+------------+
```

`VALUES (…), (…) t(a, b)` defines that a table named `t` contains `a` and `b` columns. The data type of the a column is STRING and that of the b column is BIGINT. Data types of the columns must be derived from the `VALUES` list.

- Example 6: Construct a constant table by using VALUES TABLE, instead of the combination of `SELECT * FROM` and `UNION ALL`. Sample statements:

```
select 1 c union all select 2 c;
-- The preceding statement is equivalent to the following statement:
select * from values (1), (2) t(c);
-- Return result
+------------+
| c          |
+------------+
| 1          |
| 2          |
+------------+
```

- Example 7: Use `VALUES TABLE` without the `FROM` clause to insert data. Sample statements:

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value bigint) partitioned by (p string);
-- Insert data into the srcp table.
insert into table srcp partition (p) select abs(-1), length('abc'), getdate();
-- Query data from the srcp table.
select * from srcp;
-- Return result
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| 1          | 3          | 2020-11-25 18:39:48 |
+------------+------------+------------+
```

- Example 8: Use non-constant expressions. Sample statements:

```
select * from values ('a'),(to_date('20190101', 'yyyyMMdd')),(getdate()) t(d);
```

The following result is returned:

```
+------------+
| d          |
+------------+
| 2021-02-01 18:01:38 |
| 2019-01-01 00:00:00 |
| a          |
+------------+
```

3.6. Select Operation

3.6.1. SELECT syntax

MaxCompute allows you to query data by using `SELECT` statements. This topic describes the syntax of `SELECT` statements in MaxCompute. It also describes how to use SELECT statements to perform operations, such as nested queries, sorting, and queries by group.

Before you execute `SELECT` statements, make sure that you have been granted the SELECT permission on the destination table. For more information, see Permissions.

You can execute the INSERT INTO or INSERT OVERWRITE statement on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Overview

`SELECT` statements are used to query data that meets the specified conditions from a table. The following table describes the query operations that can be performed in different scenarios.

Type	Description
Subqueries	Allows you to perform further queries based on the result of a query.
INTERSECT, UNION, EXCEPT, and MINUS	Allows you to obtain the intersection, union, or supplementary set of two datasets.
JOIN	Allows you to perform `JOIN` operations to join tables and obtain the data that meets the join condition and query condition.
SEMI JOIN	Allows you to filter data in the left table by using the right table and obtain the data that appears only in the left table.
MAPJOIN hints	Allows you to explicitly specify `MAPJOIN` hints in `SELECT` statements when you perform `JOIN` operations on one large table and one or more small tables. This improves query performance.
SKEWJOIN HINT	If two tables that you want to join contain hot key values, a long tail issue may occur. You can extract hot key values from the two tables, separately calculate the join result of hot key values and the join result of non-hot key values, and then joint the calculated data.

Type	Description
Lateral View	Allows you to use LATERAL VIEW with a user-defined table-valued function (UDTF) to split one row of data into multiple rows.
GROUPING SETS	Allows you to aggregate and analyze data from multiple dimensions.
SELECT TRANSFORM	Allows you to start a specified child process and use standard input to enter data in the required format. Then, you can parse the standard output of the child process to obtain the output data.

Limits

- After a `SELECT` statement is executed, a maximum of 10,000 rows of results can be displayed. This limit does not apply to `SELECT` clauses. `SELECT` clauses return all results in response to the query from the upper layer.
- If you execute a `SELECT` statement to query data from a partitioned table, you cannot perform a full table scan on the table.

  If your project was created after 20:00:00 on January 10, 2018, you cannot perform a full table scan on a partitioned table in the project. This limit applies when you execute a SELECT statement to query data from the table. To query data from a partitioned table, you must specify the partitions that you want to scan. This reduces unnecessary I/O and conserves computing resources. This also reduces your computing costs if you use the pay-as-you-go billing method.

  To perform a full table scan on a partitioned table, add the `set odps.sql.allow.fullscan=true;` command before the SQL statement that is used for the full table scan. Then, commit and run the added command with the SQL statement. For example, if you want to perform a full table scan on the `sale_detail` partitioned table, execute the following statement:

```
set odps.sql.allow.fullscan=true;
select * from sale_detail;
```

Syntax

```
[with <cte>[, ...] ]
select [all | distinct] <select_expr>[, <except_expr>)][, <replace_expr>] ...
       from <table_reference>
       [where <where_condition>]
       [group by {<col_list>|rollup(<col_list>)}]
           [having <having_condition>]
       [order by <order_condition>]
       [distribute by <distribute_condition> [sort by <sort_condition>]|[ cluster by <clust
er_condition>] ]
       [limit <number>]
       [window <window_clause>]
```

For more information about the sequence to execute clauses in a SELECT statement, see Sequence for executing clauses in a SELECT statement.

Sample data

This topic provides sample source data and sample statements to demonstrate how to prepare source data. The following sample statements show how to create the sale_detail table and insert data into this table.

```
-- Create a partitioned table named sale_detail.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add partitions to the sale_detail table.
alter table sale_detail add partition (sale_date='2013', region='china');
-- Insert data into the sale_detail table.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
```

WITH clause (CTE)

The WITH clause is optional. The WITH clause contains one or more common table expressions (CTEs). A CTE is used as a temporary table in the runtime environment. You can reference the temporary table in subsequent queries. When you use a CTE, you must comply with the following rules:

- The name of a CTE must be unique in a WITH clause.
- A CTE that is defined in a WITH clause can be referenced only by other CTEs that are defined later in the same WITH clause.

  For example, A is the first CTE in a WITH clause and B is the second CTE in the same WITH clause.

  - If A references A, the reference is invalid. **Incorrect usage of CTEs**:

    ```
    with
    A as (select 1 from A)
    select * from A;
    ```

  - If A references B, the reference is invalid. **Incorrect usage of CTEs**:

    ```
    with
    A as (select * from B )
    B as (select 1 as C)
    select * from B;
    ```

  - If A references B and B references A, the references are invalid. Circular reference is not supported. **Incorrect usage of CTEs**:

    ```
    with
    A as (select * from B ),
    B as (select * from A )
    select * from B;
    ```

Sample statement of correct usage:

```
with
A as (select 1 as C),
B as (select * from A)
select * from B;
```

The following result is returned:

```
+---+
| c |
+---+
| 1 |
+---+
```

Column expression (select_expr)

select_expr is required. `select_expr` is in the format of `col1_name, col2_name, column expression,...`. This format indicates a common column or partition key column that you want to query, or a regular expression that you use to query data. When you use select_expr, you must comply with the following rules:

- Specify the names of the columns from which you want to read data.

  The following statement reads data of the `shop_name` column from the `sale_detail` table. Sample statement:

  ```
  select shop_name from sale_detail;
  ```

  The following result is returned:

  ```
  +------------+
  | shop_name  |
  +------------+
  | s1         |
  | s2         |
  | s3         |
  +------------+
  ```

- Use an asterisk (`*`) to represent all columns. You can also use an asterisk (*) with `where_condition` to specify filter conditions.

- The following statement queries data of all columns from the `sale_detail` table. Sample statement:

```
-- Enable a full table scan only for the current session.
set odps.sql.allow.fullscan=true;
select * from sale_detail;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- The following statement uses an asterisk (*) with `where_condition` to specify filter conditions. Sample statement:

```
select * from sale_detail where shop_name='s1';
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Use a regular expression.
  - The following statement queries data of all columns whose names start with `sh` from the `sale_detail` table. Sample statement:

```
select `sh.*` from sale_detail;
```

The following result is returned:

```
+------------+
| shop_name  |
+------------+
| s1         |
| s2         |
| s3         |
+------------+
```

- The following statement queries data of all columns whose names are not `shop_name` from the `sale_detail` table. Sample statement:

```
select `(shop_name)?+.+` from sale_detail;
```

The following result is returned:

```
+-------------+-------------+------------+------------+
| customer_id | total_price | sale_date  | region     |
+-------------+-------------+------------+------------+
| c1          | 100.1       | 2013       | china      |
| c2          | 100.2       | 2013       | china      |
| c3          | 100.3       | 2013       | china      |
+-------------+-------------+------------+------------+
```

- The following statement queries data of all columns except the columns whose names are `shop_name` and `customer_id` from the `sale_detail` table. Sample statement:

```
select `(shop_name|customer_id)?+.+` from sale_detail;
```

The following result is returned:

```
+-------------+------------+------------+
| total_price | sale_date  | region     |
+-------------+------------+------------+
| 100.1       | 2013       | china      |
| 100.2       | 2013       | china      |
| 100.3       | 2013       | china      |
+-------------+------------+------------+
```

- The following statement queries data of all columns except the columns whose names start with `t` from the `sale_detail` table. Sample statement:

```
select `(t.*)?+.+` from sale_detail;
```

The following result is returned:

```
+------------+-------------+------------+------------+
| shop_name  | customer_id | sale_date  | region     |
+------------+-------------+------------+------------+
| s1         | c1          | 2013       | china      |
| s2         | c2          | 2013       | china      |
| s3         | c3          | 2013       | china      |
+------------+-------------+------------+------------+
```

> **Note**
>
> If the name of col2 is the prefix of the name of col1 and you want to exclude multiple columns, make sure that the name of col1 is placed before that of col2. The longer column name is placed before the shorter column name. For example, two partitions of a partitioned table do not need to be queried. One partition is named `ds` and the other is named `dshh`. The name of the ds partition is the prefix for the name of the dshh partition. Therefore, the `select `(dshhds)?+.+` from t;` expression is correct, but the `select `(ds dshh)?+.+` from t;` expression is incorrect.

- Use `DISTINCT` before the name of a column to filter out duplicate values from the column and return only distinct values. If you use `ALL` before the name of a column, all values of the column, including duplicate values, are returned. If DISTINCT is not used, `ALL` is used.
  - The following statement queries data of the region column from the sale_detail table and returns only one distinct value. Sample statement:

```
select distinct region from sale_detail;
```

The following result is returned:

```
+------------+
| region     |
+------------+
| china      |
+------------+
```

  - The following statement specifies multiple columns after the `DISTINCT` option. The `DISTINCT` option takes effect on all the specified columns instead of a single column. Sample statement:

```
select distinct region, sale_date from sale_detail;
```

The following result is returned:

```
+------------+------------+
| region     | sale_date  |
+------------+------------+
| china      | 2013       |
+------------+------------+
```

Column exclusion expression (except_expr)

except_expr is optional. `except_expr` is in the `except(col1_name, col2_name, ...)` format. You can use except_expr to read data from most columns in a table and exclude data from a small number of columns in the table. For example, you can execute the `select * except(col1_name, col2_name, ...) from ...;` statement to read data from all columns except the col1 and col2 columns.

Sample statement:

```
-- Read data from all columns, except the region column, in the sale_detail table.
select * except(region) from sale_detail;
```

The following result is returned:

```
+-----------+-------------+-------------+-----------+
| shop_name | customer_id | total_price | sale_date |
+-----------+-------------+-------------+-----------+
| s1        | c1          | 100.1       | 2013      |
| s2        | c2          | 100.2       | 2013      |
| s3        | c3          | 100.3       | 2013      |
+-----------+-------------+-------------+-----------+
```

Column modification expression (replace_expr)

replace_expr is optional. `replace_expr` is in the `replace(exp1 [as] col1_name, exp2 [as] col2_name, ...)` format. You can use replace_expr to read data from most columns in a table and modify data of a small number of columns in the table. For example, you can execute the `select * replace(exp1 as col1_name, exp2 as col2_name, ...) from ...;` statement to replace the data of the col1 column with the calculation result of exp1, and replace the data of the col2 column with the calculation result of exp2 when you read data from a table.

Sample statement:

```
-- Read data from the sale_detail table and modify the data in the total_price and region c
olumns.
select * replace(total_price+100 as total_price, 'shanghai' as region) from sale_detail;
```

The following result is returned:

```
+-----------+-------------+-------------+-----------+--------+
| shop_name | customer_id | total_price | sale_date | region |
+-----------+-------------+-------------+-----------+--------+
| s1        | c1          | 200.1       | 2013      | shanghai |
| s2        | c2          | 200.2       | 2013      | shanghai |
| s3        | c3          | 200.3       | 2013      | shanghai |
+-----------+-------------+-------------+-----------+--------+
```

Destination table information (table_reference)

table_reference is required. `table_reference` specifies the table that you want to query. When you use table_reference, you must comply with the following rules:

- Specify the name of a destination table. Sample statement:

```
select customer_id from sale_detail;
```

The following result is returned:

```
+-------------+
| customer_id |
+-------------+
| c1          |
| c2          |
| c3          |
+-------------+
```

- Use a nested subquery. Sample statement:

```
select * from (select region,sale_date from sale_detail) t where region = 'china';
```

The following result is returned:

```
+------------+------------+
| region     | sale_date  |
+------------+------------+
| china      | 2013       |
| china      | 2013       |
| china      | 2013       |
+------------+------------+
```

WHERE clause (where_condition)

where_condition is optional. `where_condition` specifies a filter condition. If where_condition is used for a partitioned table, column pruning can be performed. When you use where_condition, you must comply with the following rules:

- Use where_condition with relational operators to obtain the data that meets the specified conditions. Relational operators include:
  - `>` , `<` , `=` , `>=` , `<=` , and `<>`
  - `LIKE` and `RLIKE`
  - `IN` and `NOT IN`
  - `BETWEEN…AND`

  For more information, see Relational operators.

  The following statement specifies the partitions that you want to scan in `where_condition` . This avoids a full table scan. Sample statement:

```
select *
from sale_detail
where sale_date >= '2008' and sale_date <= '2014';
-- The preceding statement is equivalent to the following statement.
select *
from sale_detail
where sale_date between '2008' and '2014';
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

> **Note** You can execute the EXPLAIN statement to check whether partition pruning takes effect. A common user-defined function (UDF) or the partition condition settings of JOIN may cause partition pruning to fail. For more information, see Check whether partition pruning is effective.

- Use UDF-based partition pruning. If you use UDFs, MaxCompute executes the UDFs as small jobs and backfills partitions with the results of these jobs. You can enable UDF-based partition pruning by using one of the following methods:
  - Add an annotation to a UDF class when you write a UDF.

    ```
    @com.aliyun.odps.udf.annotation.UdfProperty(isDeterministic=true)
    ```

    > **Note** The UDF annotation `com.aliyun.odps.udf.annotation.UdfProperty` is defined in the *odps-sdk-udf.jar* file. To use this annotation, you must update the version of the referenced *odps-sdk-udf* to 0.30.X or later.

  - Add `set odps.sql.udf.ppr.deterministic = true;` before the SQL statement that you want to execute. Then, all UDFs in the SQL statement are considered `deterministic` UDFs. The preceding SET command backfills partitions with the results of jobs. A maximum of 1,000 partitions can be backfilled with the results of jobs. If you add an annotation to a UDF class, an error that indicates more than 1,000 partitions are backfilled may be returned. To ignore this error, you can run the `set odps.sql.udf.ppr.to.subquery = false;` command. After you run this command, UDF-based partition pruning is no longer in effect.

- In a column expression (select_expr), if the column that is renamed a column alias uses a function, the column alias cannot be referenced in the `WHERE` clause. **Incorrect sample statement**:

```
select  task_name
        ,inst_id
        ,settings
        ,GET_JSON_OBJECT(settings, '$.SKYNET_ID') as skynet_id
        ,GET_JSON_OBJECT(settings, '$.SKYNET_NODENAME') as user_agent
from    Information_Schema.TASKS_HISTORY
where   ds = '20211215' and skynet_id is not null
limit 10;
```

GROUP BY (col_list)

GROUP BY is optional. In most cases, `GROUP BY` is used with aggregate functions to group columns based on the specified common columns, partition key columns, or regular expressions. When you use `GROUP BY`, you must comply with the following rules:

- `GROUP BY` takes precedence over `SELECT`. Therefore, columns in `GROUP BY` can be specified

by column names of the input table of `SELECT` or an expression that is formed by columns of the input table of SELECT. When you use GROUP BY, take note of the following points:

- If columns in `GROUP BY` are specified by a regular expression, the complete expression must be used.
- The columns that do not use aggregate functions in a `SELECT` statement must be specified in `GROUP BY`.

Example:

- The following statement groups table data by the column name region. In this case, data is grouped based on the values of the region column. Sample statement:

```
select region from sale_detail group by region;
```

The following result is returned:

```
+------------+
| region     |
+------------+
| china      |
+------------+
```

- The following statement groups table data based on the values of the region column and returns the total sales of each group. Sample statement:

```
select sum(total_price) from sale_detail group by region;
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| 300.6      |
+------------+
```

- The following statement groups table data based on the values of the region column and returns distinct values and total sales of each group. Sample statement:

```
select region, sum (total_price) from sale_detail group by region;
```

The following result is returned:

```
+------------+------------+
| region     | _c1        |
+------------+------------+
| china      | 300.6      |
+------------+------------+
```

- The following statement groups table data based on the alias of an output column in a `SELECT` statement. Sample statement:

```
select region as r from sale_detail group by r;
-- The preceding statement is equivalent to the following statement:
select region as r from sale_detail group by region;
```

The following result is returned:

```
+------------+
| r          |
+------------+
| china      |
+------------+
```

- The following statement groups data by select_expr. Note that select_expr must be a complete regular expression. Sample statement:

```
select 2 + total_price as r from sale_detail group by 2 + total_price;
```

The following result is returned:

```
+------------+
| r          |
+------------+
| 102.1      |
| 102.2      |
| 102.3      |
+------------+
```

- If some columns in a `SELECT` statement do not use aggregate functions, these columns must be specified in `GROUP BY`. Otherwise, an error is returned. **Incorrect sample statement**:

```
select region, total_price from sale_detail group by region;
```

Sample statement of correct usage:

```
select region, total_price from sale_detail group by region, total_price;
```

The following result is returned:

```
+------------+-------------+
| region     | total_price |
+------------+-------------+
| china      | 100.1       |
| china      | 100.2       |
| china      | 100.3       |
+------------+-------------+
```

- If you add the `set hive.groupby.position.alias=true;` command before a SELECT statement, integer constants in the `GROUP BY` clause are considered column numbers in a `SELECT` statement. Sample statement:

```
-- Run this command with the following SELECT statement.
set odps.sql.groupby.position.alias=true;
-- 1 indicates the region column, which is the first column read by the following SELECT
statement. This statement groups table data based on the values of the region column and
returns distinct values of the region column and total sales of each group.
select region, sum(total_price) from sale_detail group by 1;
```

The following result is returned:

```
+------------+------------+
| region     | _c1        |
+------------+------------+
| china      | 300.6      |
+------------+------------+
```

HAVING clause (having_condition)

having_condition is optional. In most cases, `having_condition` is used with aggregate functions to filter data. Sample statement:

```
-- Insert data into the sale_detail table to display the data rendering effect.
insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null','c5'
,null),('s6','c6',100.4),('s7','c7',100.5);
-- Use having_condition with aggregate functions to filter data.
select region,sum(total_price) from sale_detail
group by region
having sum(total_price)<305;
```

The following result is returned:

```
+------------+------------+
| region     | _c1        |
+------------+------------+
| china      | 300.6      |
| shanghai   | 200.9      |
+------------+------------+
```

ORDER BY (order_condition)

order_condition is optional. `ORDER BY` is used to sort all data records based on a specified common column or partition key column. ORDER BY can also be used to sort all data records based on a specified constant. When you use `ORDER BY`, you must comply with the following rules:

- By default, data is sorted in ascending order. If you want to sort data in descending order, the `DESC` keyword is required.
- By default, `ORDER BY` is followed by `LIMIT <number>` to limit the number of data rows that are displayed in the output. If ORDER BY is not followed by `LIMIT <number>`, an error is returned. You can also work around this `limit`. For more information, see LIMIT <number>.

- The following statement queries data from the sale_detail table, sorts data records in ascending order based on the values of the total_price column, and then displays the first two records. Sample statement:

```
select * from sale_detail order by total_price limit 2;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- The following statement queries data from the sale_detail table, sorts data records in descending order based on the values of the total_price column, and then displays the first two records. Sample statement:

```
select * from sale_detail order by total_price desc limit 2;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s3         | c3          | 100.3       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- NULL is the smallest value when you use `ORDER BY` to sort data. This is also the case in MySQL. However, this is not the case in Oracle.

  The following statement queries data from the sale_detail table, sorts data records in ascending order based on the values of the total_price column, and then displays the first two records. Sample statement:

```
select * from sale_detail order by total_price limit 2;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- `ORDER BY` is followed by the alias of an output column of a `SELECT` statement. If you do not specify the alias of an output column of a `SELECT` statement, the name of this column is used as the alias of this column.

  The following statement adds the alias of an output column after `ORDER BY`. Sample statement:

```
select total_price as t from sale_detail order by total_price limit 3;
-- The preceding statement is equivalent to the following statement:
select total_price as t from sale_detail order by t limit 3;
```

The following result is returned:

```
+------------+
| t          |
+------------+
| 100.1      |
| 100.2      |
| 100.3      |
+------------+
```

- If you add the `set hive.orderby.position.alias=true;` command before a SELECT statement, integer constants in the `ORDER BY` clause are considered column numbers in the `SELECT` statement. Sample statement:

  ```
  -- Run this command with the following SELECT statement.
  set odps.sql.orderby.position.alias=true;
  select * from sale_detail order by 3 limit 3;
  ```

  The following result is returned:

  ```
  +------------+-------------+-------------+------------+------------+
  | shop_name  | customer_id | total_price | sale_date  | region     |
  +------------+-------------+-------------+------------+------------+
  | s1         | c1          | 100.1       | 2013       | china      |
  | s2         | c2          | 100.2       | 2013       | china      |
  | s3         | c3          | 100.3       | 2013       | china      |
  +------------+-------------+-------------+------------+------------+
  ```

- An `OFFSET` clause can be used with an `ORDER BY...LIMIT` clause to specify the number of rows to skip. The format is `ORDER BY...LIMIT m OFFSET n`, which can be abbreviated as `ORDER BY...LIMIT n, m`. `LIMIT m` specifies that m rows of data are returned. `OFFSET n` specifies that n rows are skipped before data is returned. If you do not want to skip rows, you can use `OFFSET 0` in the statement that you want to execute. You can also execute the statement without specifying an OFFSET clause.

  The following statement sorts the data of the sale_detail table in ascending order based on the values of the total_price column and displays three rows of data starting from the third row. Sample statement:

  ```
  select customer_id,total_price from sale_detail order by total_price limit 3 offset 2;
  -- The preceding statement is equivalent to the following statement:
  select customer_id,total_price from sale_detail order by total_price limit 2, 3;
  ```

  The following result is returned:

  ```
  +-------------+-------------+
  | customer_id | total_price |
  +-------------+-------------+
  | c3          | 100.3       |
  +-------------+-------------+
  ```

The queried data contains only three rows of data. In this case, only the third row is returned.

DISTRIBUTE BY hash partition (distribute_condition)

DISTRIBUTE BY is optional. `DISTRIBUTE BY` is used to perform hash partitioning on data based on the values of specific columns.

`DISTRIBUTE BY` controls how the output of a mapper is distributed among reducers. To prevent the same data from being distributed to different reducers, you can use `DISTRIBUTE BY`. This ensures that the same group of data is distributed to the same reducer.

The alias of an output column of a `SELECT` statement must be specified. If you execute a `SELECT` statement to query data of a column and the alias of this column is not specified, the column name is used as the alias. Sample statement:

```
-- The following statement queries the values of the region column from the sale_detail tab
le and performs hash partitioning on data based on the values of the region column.
select region from sale_detail distribute by region;
-- The preceding statement is equivalent to the following statements:
select region as r from sale_detail distribute by region;
select region as r from sale_detail distribute by r;
```

SORT BY (sort_condition)

SORT BY is optional. In most cases, SORT BY is used with `DISTRIBUTE BY`. When you use `SORT BY`, you must comply with the following rules:

- By default, data is sorted in ascending order. If you want to sort data in descending order, the `DESC` keyword is required.
- If `SORT BY` is preceded by `DISTRIBUTE BY`, `SORT BY` sorts the result of `DISTRIBUTE BY` based on the values of a specified column.
  - The following statements query the values of the region and total_price columns from the sale_detail table, perform hash partitioning on the query results based on the values of the region column, and then sort the partitioning results in ascending order based on the values of the total_price column. Sample statement:

    ```
    -- Insert data into the sale_detail table to display the data rendering effect.
    insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null',
    'c5',null),('s6','c6',100.4),('s7','c7',100.5);
    select region,total_price from sale_detail distribute by region sort by total_price;
    ```

    The following result is returned:

    ```
    +------------+-------------+
    | region     | total_price |
    +------------+-------------+
    | shanghai   | NULL        |
    | china      | 100.1       |
    | china      | 100.2       |
    | china      | 100.3       |
    | shanghai   | 100.4       |
    | shanghai   | 100.5       |
    +------------+-------------+
    ```

- The following statement queries the values of the region and total_price columns from the sale_detail table, performs hash partitioning on the query results based on the values of the region column, and then sorts the partitioning results in descending order based on the values of the total_price column. Sample statement:

  ```
  select region,total_price from sale_detail distribute by region sort by total_price des
  c;
  ```

  The following result is returned:

  ```
  +------------+-------------+
  | region     | total_price |
  +------------+-------------+
  | shanghai   | 100.5       |
  | shanghai   | 100.4       |
  | china      | 100.3       |
  | china      | 100.2       |
  | china      | 100.1       |
  | shanghai   | NULL        |
  +------------+-------------+
  ```

- If `SORT BY` is not preceded by `DISTRIBUTE BY`, `SORT BY` sorts the data that is distributed to each reducer.

  This ensures that the output data of each reducer is sorted in order and increases the storage compression ratio. If data is filtered during data reading, this method reduces the amount of data that is read from disks and improves the efficiency of subsequent global sorting. Sample statement:

  ```
  select region,total_price from sale_detail sort by total_price desc;
  ```

  The following result is returned:

  ```
  +------------+-------------+
  | region     | total_price |
  +------------+-------------+
  | china      | 100.3       |
  | china      | 100.2       |
  | china      | 100.1       |
  | shanghai   | 100.5       |
  | shanghai   | 100.4       |
  | shanghai   | NULL        |
  +------------+-------------+
  ```

> **Note**
>
> - Columns in an `ORDER BY, DISTRIBUTE BY, or SORT BY` clause must be specified by the aliases of the output columns in a `SELECT` statement. Column aliases can be Chinese.
> - In MaxCompute, an `ORDER BY, DISTRIBUTE BY, or SORT BY` clause is executed after a `SELECT` statement. Therefore, columns in ORDER BY, DISTRIBUTE BY, or SORT BY must be specified by the aliases of the output columns in the `SELECT` statement.
> - `ORDER BY` cannot be used at the same time with `DISTRIBUTE BY` or `SORT BY`. Similarly, `GROUP BY` cannot be used at the same time with `DISTRIBUTE BY` or `SORT BY`.

LIMIT <number>

LIMIT <number> is optional. In `LIMIT <number>`, `number` is a constant that restricts the number of rows that can be displayed.

> **Note** `LIMIT <number>` is used to scan and filter data for a distributed query system. When you use `LIMIT <number>`, the amount of data returned is not reduced. Therefore, computing costs are not reduced.

The following sections describe the limits of LIMIT <number> and how to work around these limits.

- `ORDER BY` must be used with `LIMIT <number>`.

  `ORDER BY` sorts all data of a single node. By default, ORDER BY is used with `LIMIT <number>` to prevent a single node from processing large amounts of data. You can work around this `limit` by using the following methods:

  - To work around the `limit` for a project, run the `setproject odps.sql.validate.orderby.limit=false;` command.
  - To remove the limit for a session, commit and run the `set.odps.sql.validate.orderby.limit=false;` command with the SQL statement that you want to execute.

    > **Note** After you work around this `limit`, if a single node has large amounts of data to sort, more resources and time are consumed.

- Limited rows are displayed.

  If you execute a `SELECT` statement without a `LIMIT <number>` clause or the number specified in the `LIMIT <number>` clause exceeds the maximum `number (n)` of rows that can be displayed, a maximum of n rows can be displayed.

  The maximum number of rows that can be displayed varies based on projects. You can use one of the following methods to control the maximum number:

- If project data protection is disabled, modify the odpscmd config.ini file.

  Set `use_instance_tunnel` to true in the odpscmd config.ini file. If the `instance_tunnel_max_record` parameter is not configured, the number of rows that can be displayed is not limited. Otherwise, the number of rows that can be displayed is limited by the `instance_tunnel_max_record` parameter. The maximum value of the `instance_tunnel_max_record` parameter is 10000. For more information about InstanceTunnel, see Usage notes.

- If project data protection is enabled, the number of rows that can be displayed is limited by the `READ_TABLE_MAX_ROW` parameter. The maximum value of this parameter is 10000.

> **Note** You can run the `SHOW SecurityConfiguration;` command to view the value of `ProjectProtection`. If `ProjectProtection` is set to true, you can determine whether to disable project data protection based on your business requirements. You can run the `set ProjectProtection=false;` command to disable project data protection. By default, `ProjectProtection` is set to false. For more information about project data protection, see Project data protection.

Window clause (window_clause)

For more information about the window clause, see Syntax.

3.6.2. Sequence for executing clauses in a SELECT statement

Clauses in a `SELECT` statement that is written in compliance with the `SELECT` syntax of MaxCompute are executed in a different sequence from the clauses in a standard SELECT statement. This topic describes the sequence for executing clauses in a `SELECT` statement of MaxCompute and provides examples for reference.

Sequence for executing clauses in a SELECT statement

The SELECT syntax includes the following clauses:

- `SELECT`
- `FROM`
- `WHERE`
- `GROUP BY`
- `HAVING`
- `ORDER BY`
- `DISTRIBUTE BY`
- `SORT BY`
- `LIMIT`

The following clauses cannot be used with `DISTRIBUTE BY` or `SORT BY`: `ORDER BY` and `GROUP BY`. To address this issue, you can execute clauses in a `SELECT` statement in one of the following sequences:

- Sequence 1: `FROM` > `WHERE` > `GROUP BY` > `HAVING` > `SELECT` > `ORDER BY` > `LIMIT`

- Sequence 2: `FROM` > `WHERE` > `SELECT` > `DISTRIBUTE BY` > `SORT BY`

To avoid confusion, MaxCompute allows you to write a SELECT statement in the preceding sequences. The syntax of a SELECT statement can be changed to the following form:

```
from <table_reference>
[where <where_condition>]
[group by <col_list>]
[having <having_condition>]
select [all | distinct] <select_expr>, <select_expr>, ...
[order by <order_condition>]
[distribute by <distribute_condition> [sort by <sort_condition>] ]
[limit <number>]
```

Sample data

This topic provides source data and sample statements for generating source data. This helps you understand how to prepare source data. Sample statements:

```
-- Create a partitioned table named sale_detail.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add partitions to the sale_detail table.
alter table sale_detail add partition (sale_date='2013', region='china') partition (sale_da
te='2014', region='shanghai');
-- Insert data into the sale_detail table.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null','c5'
,null),('s6','c6',100.4),('s7','c7',100.5);
```

Examples

- Example 1: Clauses in a SELECT statement are executed in Sequence 1.

```
-- Write a SELECT statement based on the SELECT syntax.
select region,max(total_price)
from sale_detail
where total_price > 100
group by region
having sum(total_price)>305
order by region
limit 5;
-- Write a SELECT statement based on Sequence 1. The following statement is equivalent to
the preceding statement.
from sale_detail
where total_price > 100
group by region
having sum(total_price)>305
select region,max(total_price)
order by region
limit 5;
```

The following result is returned:

```
+------------+------------+
| region     | _c1        |
+------------+------------+
| china      | 100.3      |
+------------+------------+
```

Logic to execute clauses in the SELECT statement:

i. Retrieves the data that meets the condition (`WHERE total_price > 100`) from the sale_detail table (`FROM sale_detail`).

ii. Groups the data obtained from Step a based on the values of the region column (`GROUP BY`).

iii. Retrieves the data whose total_price is greater than 305 from the data obtained from Step b (`HAVING sum(total_price)>305`).

iv. Obtains the maximum value of the total_price column in each region (`SELECT region,max(total_price)`) from the data obtained from Step c.

v. Sorts the data obtained from Step d based on the values of the region column (`ORDER BY region`).

vi. Displays the first five data records (`LIMIT 5`) of the data obtained from Step e.

- Example 2: Clauses in a SELECT statement are executed in Sequence 2.

```
-- Write a SELECT statement based on the SELECT syntax.
select shop_name
       ,total_price
       ,region
from   sale_detail
where  total_price > 100.2
distribute by region
sort by total_price;
-- Write a SELECT statement based on Sequence 2. The following statement is equivalent to
the preceding statement.
from   sale_detail
where  total_price > 100.2
select shop_name
       ,total_price
       ,region
distribute by region
sort by total_price;
```

The following result is returned:

```
+------------+-------------+------------+
| shop_name  | total_price | region     |
+------------+-------------+------------+
| s3         | 100.3       | china      |
| s6         | 100.4       | shanghai   |
| s7         | 100.5       | shanghai   |
+------------+-------------+------------+
```

Logic to execute clauses in the SELECT statement:

i. Retrieves the data that meets the condition (`WHERE total_price > 100.2`) from the sale_detail table (`FROM sale_detail`).

ii. Retrieves data from the data obtained from Step a based on the values of the shop name, total price, and region columns (`SELECT shop_name, total_price, region`).

iii. Performs hash partitioning on the data obtained from Step b based on the values of the region column (`DISTRIBUTE BY region`).

iv. Sorts the data obtained from Step c based on the values of the total price column (`SORT BY total_price`).

3.6.3. Subqueries

You can use a subquery if you want to perform a further query based on the query results of a SELECT statement. This topic describes the definition and usage of subqueries that are supported in MaxCompute.

Description

Subqueries are nested inside a SELECT statement to perform complex data queries. MaxCompute supports the following types of subqueries:

- Basic subquery
- IN SUBQUERY

- NOT IN SUBQUERY
- EXISTS SUBQUERY
- NOT EXISTS SUBQUERY
- SCALAR SUBQUERY

Sample data

This topic provides source data and sample statements for generating source data. This helps you understand how to prepare source data. Sample statements:

```
-- Create a partitioned table named sale_detail.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add partitions to the sale_detail table.
alter table sale_detail add partition (sale_date='2013', region='china') partition (sale_da
te='2014', region='shanghai');
-- Insert data into the sale_detail table.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null','c5'
,null),('s6','c6',100.4),('s7','c7',100.5);
```

Basic subquery

Tables are objects for common queries. You can also use a `SELECT` statement that is nested inside another SELECT statement to specify an object for a subquery. A subquery in a `FROM` clause can be used as a table. You can `join` the subquery with other tables or subqueries. For more information about `JOIN` operations, see JOIN.

- Syntax
  - Syntax 1

    ```
    select <select_expr> from (<select_statement>) [<sq_alias_name>];
    ```

  - Syntax 2

    ```
    select (<select_statement>) from <table_name>;
    ```

- Parameters
  - select_expr: required. The value of this parameter is in the format of `col1_name, col2_name, Regular expression,...`. This format indicates common columns or partition key columns that you want to query or regular expressions that are used for a query.
  - select_statement: required. This parameter specifies a subquery clause. If you use Syntax 2, the subquery result can have only one row. For more information about the syntax, see SELECT syntax.
  - sq_alias_name: optional. This parameter specifies the alias of a subquery.
  - table_name: required. This parameter specifies the name of the table that you want to query.

- Examples
  - Example 1: Use Syntax 1. The following statement shows an example:

```
select * from (select shop_name from sale_detail) a;
```

The following result is returned:

```
+------------+
| shop_name  |
+------------+
| s1         |
| s2         |
| s3         |
| null       |
| s6         |
| s7         |
+------------+
```

  - Example 2: Use Syntax 2. The following statement shows an example:

```
select (select * from sale_detail where shop_name='s1') from sale_detail;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s1         | c1          | 100.1       | 2013       | china      |
| s1         | c1          | 100.1       | 2013       | china      |
| s1         | c1          | 100.1       | 2013       | china      |
| s1         | c1          | 100.1       | 2013       | china      |
| s1         | c1          | 100.1       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 3: Use Syntax 1. In this example, a subquery in a `FROM` clause is used as a table, and the subquery is `joined` with other tables or subqueries. The following statements show an example:

```
-- Create a table and join the table with a subquery.
create table shop as select shop_name,customer_id,total_price from sale_detail;
select a.shop_name, a.customer_id, a.total_price from
(select * from shop) a join sale_detail on a.shop_name = sale_detail.shop_name;
```

The following result is returned:

```
+------------+-------------+-------------+
| shop_name  | customer_id | total_price |
+------------+-------------+-------------+
| null       | c5          | NULL        |
| s6         | c6          | 100.4       |
| s7         | c7          | 100.5       |
| s1         | c1          | 100.1       |
| s2         | c2          | 100.2       |
| s3         | c3          | 100.3       |
+------------+-------------+-------------+
```

IN SUBQUERY

`IN SUBQUERY` is used in a similar manner to `LEFT SEMI JOIN`.

- Syntax
  - Syntax 1

```
select <select_expr1> from <table_name1> where <select_expr2> in (select <select_expr2>
from <table_name2>);
-- The preceding statement is equivalent to the following statement with LEFT SEMI JOIN
:
select <select_expr1> from <table_name1> <alias_name1> left semi join <table_name2> <al
ias_name2> on <alias_name1>.<select_expr1> = <alias_name2>.<select_expr2>;
```

> **Note** If `select_expr2` specifies partition key columns, `select <select_expr2> from <table_name2>` is not converted into `LEFT SEMI JOIN`. A separate job is started to run a subquery. MaxCompute compares the subquery results with the columns that you specify in `select_expr2` in sequence. If the partitions of the table specified by `table_name1` contain the columns in `select_expr2` and these columns are not included in the results, MaxCompute does not read data from these partitions. This ensures that partition pruning is still valid.

- Syntax 2

  MaxCompute supports `IN SUBQUERY` and correlated conditions. `where <table_name2_colname> = <table_name1>.<colname>` is a correlated condition. MaxCompute V1.0 does not support expressions that reference source tables from both subqueries and outer queries. MaxCompute V2.0 supports such expressions. These expressions are part of the `ON` condition in `SEMI JOIN` operations.

  ```
  select <select_expr1> from <table_name1> where <select_expr2> in (select <select_expr2> from <table_name2> where <table_name2_colname> = <table_name1>.<colname>);
  ```

  > **Note** MaxCompute supports `IN SUBQUERY` that does not serve as a `JOIN` condition. For example, a non-`WHERE` clause uses IN SUBQUERY, or a `WHERE` clause uses IN SUBQUERY that cannot be converted into a `JOIN` condition. In this case, IN SUBQUERY cannot be converted into `SEMI JOIN`. A separate job must be started to run a subquery. Correlated conditions are not supported.

- Syntax 3

  IN SUBQUERY supports multi-column subqueries based on the preceding capabilities and limits. This rule also applies to PostgreSQL. If you use Syntax 3 for IN SUBQUERY, you do not need to split a query into subqueries. Multi-column subqueries reduce one JOIN operation and save computing resources. You can use multi-column subqueries in the following ways:

  - Use a simple SELECT statement in which you specify multiple columns for the `IN SUBQUERY` expression.
  - Use aggregate functions for the `IN SUBQUERY` expression. For more information about aggregate functions, see Aggregate functions.
  - Use constants for the `IN SUBQUERY` expression.

- Parameters
  - select_expr1: required. The value of this parameter is in the format of `col1_name, col2_name, Regular expression,...`. This format indicates common columns or partition key columns that you want to query or regular expressions that are used for a query.
  - table_name1 and table_name2: required. These parameters specify the names of tables.
  - select_expr2: required. This parameter specifies the names of the columns in the tables specified by table_name1 and table_name2. Columns in the two tables are mapped to each other.
  - col_name: required. This parameter specifies the name of a column in the table.
- Examples

- Example 1: Use Syntax 1. The following statement shows an example:

```
select * from sale_detail where shop_name in (select shop_name from shop);
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 2: Use Syntax 2. The following statement shows an example:

```
select * from sale_detail where shop_name in (select shop_name from shop where customer
_id = shop.customer_id);
```

The following result is returned:

```
+-----------+-------------+-------------+-----------+--------+
| shop_name | customer_id | total_price | sale_date | region |
+-----------+-------------+-------------+-----------+--------+
| null      | c5          | NULL        | 2014      | shanghai |
| s6        | c6          | 100.4       | 2014      | shanghai |
| s7        | c7          | 100.5       | 2014      | shanghai |
| s1        | c1          | 100.1       | 2013      | china  |
| s2        | c2          | 100.2       | 2013      | china  |
| s3        | c3          | 100.3       | 2013      | china  |
+-----------+-------------+-------------+-----------+--------+
```

- Example 3: `IN SUBQUERY` does not serve as a `JOIN` condition. The following statement shows an example:

```
select * from sale_detail where shop_name in (select shop_name from shop) and total_pri
ce > 100.3;
```

IN SUBQUERY cannot be converted into `SEMI JOIN`. This is because the `WHERE` clause includes an `AND` operator. A separate job is started to run the subquery.

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

- Example 4: Multiple columns are specified in a SELECT statement for a subquery. The following statements show an example:

```
-- Sample data is reconstructed to help you understand this example.
create table if not exists t1(a bigint,b bigint,c bigint,d bigint,e bigint);
create table if not exists t2(a bigint,b bigint,c bigint,d bigint,e bigint);
insert into table t1 values (1,3,2,1,1),(2,2,1,3,1),(3,1,1,1,1),(2,1,1,0,1),(1,1,1,0,1)
;
insert into table t2 values (1,3,5,0,1),(2,2,3,1,1),(3,1,1,0,1),(2,1,1,0,1),(1,1,1,0,1)
;
-- Scenario 1: The IN SUBQUERY expression is a simple SELECT statement in which you spe
cify multiple columns.
select a, b from t1 where (c, d) in (select a, b from t2 where e = t1.e);
-- The following result is returned:
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 3          |
| 2          | 2          |
| 3          | 1          |
+------------+------------+
-- Scenario 2: The IN SUBQUERY expression uses aggregate functions.
select a, b from t1 where (c, d) in (select max(a), b from t2 where e = t1.e group by b
having max(a) > 0);
-- The following result is returned:
+------------+------------+
| a          | b          |
+------------+------------+
| 2          | 2          |
+------------+------------+
-- Scenario 3: The IN SUBQUERY expression uses constants.
select a, b from t1 where (c, d) in ((1, 3), (1, 1));
-- The following result is returned:
+------------+------------+
| a          | b          |
+------------+------------+
| 2          | 2          |
| 3          | 1          |
+------------+------------+
```

NOT IN SUBQUERY

`NOT IN SUBQUERY` is used in a similar manner to `LEFT ANTI JOIN`. However, if the values of a row are NULL for a specified column in the table that you want to query, the value of the expression in `NOT IN SUBQUERY` is NULL. In this case, the `WHERE` condition is invalid, and no data is returned. This processing logic is different from that of `LEFT ANTI JOIN`.

- Syntax

- Syntax 1

```
select <select_expr1> from <table_name1> where <select_expr2> not in (select <select_ex
pr2> from <table_name2>);
-- The preceding statement is equivalent to the following statement with LEFT ANTI JOIN
.
select <select_expr1> from <table_name1> <alias_name1> left anti join <table_name2> <al
ias_name2> on <alias_name1>.<select_expr1> = <alias_name2>.<select_expr2>;
```

> **Note** If `select_expr2` specifies partition key columns, `select <select_expr2> from <table_name2>` is not converted into `LEFT ANTI JOIN`. A separate job is started to run a subquery. MaxCompute compares the subquery results with the columns specified in `select_expr2` in sequence. If the partitions of the table specified by `table_name1` contain the columns in `select_expr2` and these columns are not included in the results, MaxCompute does not read data from these partitions. This ensures that partition pruning is still valid.

- Syntax 2

  MaxCompute supports `NOT IN SUBQUERY` and correlated conditions. `where <table_name2_colname> = <table_name1>.<colname>` in a subquery is a correlated condition. MaxCompute V1.0 does not support expressions that reference source tables from both subqueries and outer queries. MaxCompute V2.0 supports such expressions. These expressions are part of the `ON` condition in `ANTI JOIN` operations.

```
select <select_expr1> from <table_name1> where <select_expr2> not in (select <select_ex
pr2> from <table_name2> where <table_name2_colname> = <table_name1>.<colname>);
```

> **Note** MaxCompute supports `NOT IN SUBQUERY` that does not serve as a `JOIN` condition. For example, a non-`WHERE` clause uses NOT IN SUBQUERY, or a `WHERE` clause uses NOT IN SUBQUERY that cannot be converted into a `JOIN` condition. In this case, NOT IN SUBQUERY cannot be converted into `ANTI JOIN`. A separate job must be started to run a subquery. Correlated conditions are not supported.

- Syntax 3

  NOT IN SUBQUERY supports multi-column subqueries based on the preceding capabilities and limits. This rule also applies to PostgreSQL. If you use Syntax 3 for NOT IN SUBQUERY, you do not need to split a query into multiple subqueries. Multi-column subqueries reduce one JOIN operation and save computing resources. You can use multi-column subqueries in the following ways:

  - Use a simple SELECT statement in which you specify multiple columns for the `NOT IN SUBQUERY` expression.
  - Use aggregate functions for the `NOT IN SUBQUERY` expression. For more information about aggregate functions, see Aggregate functions.
  - Use constants for the `NOT IN SUBQUERY` expression.

- Parameters
  - select_expr1: required. The value of this parameter is in the format of `col1_name, col2_name, Regular expression,...`. This format indicates common columns or partition key columns that you want to query or regular expressions that are used for a query.

- table_name1 and table_name2: required. These parameters specify the names of tables.
- select_expr2: required. This parameter specifies the names of the columns in the tables specified by table_name1 and table_name2. Columns in the two tables are mapped to each other.
- col_name: required. This parameter specifies the name of a column in the table.

- Examples
  - Example 1: Use Syntax 1. The following statements show an example:

```
-- Create a table named shop1 and insert data into the table.
create table shop1 as select shop_name,customer_id,total_price from sale_detail;
insert into shop1 values ('s8','c1',100.1);
select * from shop1 where shop_name not in (select shop_name from sale_detail);
```

    The following result is returned:

```
+------------+-------------+-------------+
| shop_name  | customer_id | total_price |
+------------+-------------+-------------+
| s8         | c1          | 100.1       |
+------------+-------------+-------------+
```

  - Example 2: Use Syntax 2. The following statement shows an example:

```
select * from shop1 where shop_name not in (select shop_name from sale_detail where cus
tomer_id = shop1.customer_id);
```

    The following result is returned:

```
+------------+-------------+-------------+
| shop_name  | customer_id | total_price |
+------------+-------------+-------------+
| s8         | c1          | 100.1       |
+------------+-------------+-------------+
```

  - Example 3: `NOT IN SUBQUERY` does not serve as a `JOIN` condition. The following statement shows an example:

```
select * from shop1 where shop_name not in (select shop_name from sale_detail) and tota
l_price < 100.3;
```

    NOT IN SUBQUERY cannot be converted into `ANTI JOIN`. This is because the `WHERE` clause includes an `AND` operator. A separate job is started to run a subquery.

    The following result is returned:

```
+------------+-------------+-------------+
| shop_name  | customer_id | total_price |
+------------+-------------+-------------+
| s8         | c1          | 100.1       |
+------------+-------------+-------------+
```

- Example 4: If the values of a row in the table from which you want to query data are NULL, no data is returned. The following statements show an example:

```
-- Create a table named sale and insert data into the table.
create table if not exists sale
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
alter table sale add partition (sale_date='2013', region='china');
insert into sale partition (sale_date='2013', region='china') values ('null','null',nul
l),('s2','c2',100.2),('s3','c3',100.3);
select * from sale where shop_name not in (select shop_name from sale_detail);
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
+------------+-------------+-------------+------------+------------+
```

- Example 5: Multiple columns are specified in a SELECT statement for a subquery. The following statements show an example:

```
-- Sample data is reconstructed to help you understand this example. The sample data is
the same as that of IN SUBQUERY.
create table if not exists t1(a bigint,b bigint,c bigint,d bigint,e bigint);
create table if not exists t2(a bigint,b bigint,c bigint,d bigint,e bigint);
insert into table t1 values (1,3,2,1,1),(2,2,1,3,1),(3,1,1,1,1),(2,1,1,0,1),(1,1,1,0,1)
;
insert into table t2 values (1,3,5,0,1),(2,2,3,1,1),(3,1,1,0,1),(2,1,1,0,1),(1,1,1,0,1)
;
-- Scenario 1: The NOT IN SUBQUERY expression is a simple SELECT statement in which you
specify multiple columns.
select a, b from t1 where (c, d) not in (select a, b from t2 where e = t1.e);
-- The following result is returned:
+------------+------------+
| a          | b          |
+------------+------------+
| 2          | 1          |
| 1          | 1          |
+------------+------------+
-- Scenario 2: The NOT IN SUBQUERY expression uses aggregate functions.
select a, b from t1 where (c, d) not in (select max(a), b from t2 where e = t1.e group
by b having max(a) > 0);
-- The following result is returned:
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 3          |
| 3          | 1          |
| 2          | 1          |
| 1          | 1          |
+------------+------------+
-- Scenario 3: The NOT IN SUBQUERY expression uses constants.
select a, b from t1 where (c, d) not in ((1, 3), (1, 1));
-- The following result is returned:
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 3          |
| 2          | 1          |
| 1          | 1          |
+------------+------------+
```

EXISTS SUBQUERY

When you use an `EXISTS SUBQUERY` clause, if the subquery returns at least one row of data, True is returned. If the subquery does not return data, False is returned.

MaxCompute supports only the `WHERE` subqueries that have correlated conditions. To use an `EXISTS SUBQUERY` clause, you must convert this clause into `LEFT SEMI JOIN` .

- Syntax

```
select <select_expr> from <table_name1> where exists (select <select_expr> from <table_na
me2> where <table_name2_colname> = <table_name1>.<colname>);
```

- Parameters
  - select_expr: required. The value of this parameter is in the format of `col1_name, col2_name, Regular expression,...`. This format indicates common columns or partition key columns that you want to query or regular expressions that are used for a query.
  - table_name1 and table_name2: required. These parameters specify the names of tables.
  - col_name: required. This parameter specifies the name of a column in the table.
- Example

```
select * from sale_detail where exists (select * from shop where customer_id = sale_detai
l.customer_id);
-- The preceding statement is equivalent to the following statement:
select * from sale_detail a left semi join shop b on a.customer_id = b.customer_id;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

NOT EXISTS SUBQUERY

When you use a `NOT EXISTS SUBQUERY` clause, if no data is returned, True is returned. Otherwise, False is returned.

MaxCompute supports only the `WHERE` subqueries that have correlated conditions. To use a `NOT EXISTS SUBQUERY` clause, you must convert this clause into `LEFT ANTI JOIN`.

- Syntax

```
select <select_expr> from <table_name1> where not exists (select <select_expr> from <tabl
e_name2> where <table_name2_colname> = <table_name1>.<colname>);
```

- Parameters
  - select_expr: required. The value of this parameter is in the format of `col1_name, col2_name, Regular expression,...`. This format indicates common columns or partition key columns that you want to query or regular expressions that are used for a query.
  - table_name1 and table_name2: required. These parameters specify the names of tables.
  - col_name: required. This parameter specifies the name of a column in the table.
- Example

```
select * from sale_detail where not exists (select * from shop where shop_name = sale_det
ail.shop_name);
-- The preceding statement is equivalent to the following statement:
select * from sale_detail a left anti join shop b on a.shop_name = b.shop_name;
```

The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
+------------+-------------+-------------+------------+------------+
```

SCALAR SUBQUERY

If the output result of a SCALAR SUBQUERY clause contains only one row and one column of data, the result can be used as a scalar for data computations.

All SCALAR SUBQUERY clauses whose output result contains only one row and one column of data can be rewritten based on the following syntax. If the output result of SCALAR SUBQUERY contains only one row of data and one `MAX` or `MIN` operator is nested outside SCALAR SUBQUERY, the result does not change.

- Syntax

```
select <select_expr> from <table_name1> where (<select count(*) from <table_name2> where
<table_name2_colname> = <table_name1>.<colname>) <Scalar operator> <scalar_value>;
-- The preceding statement is equivalent to the following statement:
select <table_name1>.<select_expr> from <table_name1> left semi join (select <colname>, c
ount(*) from <table_name2> group by <colname> having count(*) <Scalar operator> <scalar_v
alue>) <table_name2> on <table_name1>.<colname> = <table_name2>.<colname>;
```

> **Note**
>
> - The output result of `select count(*) from <table_name2> where <table_name2_colname> = <table_name1>.<colname>` is a row set. The output contains only one row and one column of data. In this case, the result can be used as a scalar. In practical application, SCALAR SUBQUERY is converted into `JOIN` as much as possible.
> - The output result of SCALAR SUBQUERY can be used as a scalar only if you can confirm in the compilation phase that SCALAR SUBQUERY returns only one row and one column of data. If you cannot make this confirmation until the running phase, the compiler reports an error. The compiler can compile the statements that meet the following requirements:
>   - The `SELECT` list of SCALAR SUBQUERY uses aggregate functions that are not included in the parameters of a specified user-defined table-valued function (UDTF).
>   - SCALAR SUBQUERY that uses aggregate functions does not include a `GROUP BY` clause.

SCALAR SUBQUERY also supports multi-column subqueries based on the preceding capabilities and limits.

- A SELECT list is a SCALAR SUBQUERY expression in which you specify multiple columns. The expression must be an equality expression.
- Columns in a SELECT list can be an expression of the BOOLEAN type. Only equivalent comparison is supported.
- A `WHERE` clause supports multi-column comparison. Only equivalent comparison is supported.

- Parameters

- select_expr: required. The value of this parameter is in the format of `col1_name, col2_name, Regular expression,...`. This format indicates common columns or partition key columns that you want to query or regular expressions that are used for a query.
- table_name1 and table_name2: required. These parameters specify the names of tables.
- col_name: required. This parameter specifies the name of a column in the table.
- Scalar operator: required. The operators include greater than (>), less than (<), equal to (=), greater than or equal to (>=), and less than or equal to (<=).
- scalar_value: required. This parameter specifies a scalar value.

- Limits
  - `SCALAR SUBQUERY` can reference columns from outer queries. If `SCALAR SUBQUERY` uses multiple-level nesting, only the outermost column can be referenced.

```
-- Sample statement that you can execute:
select * from t1 where (select count(*) from t2 where t1.a = t2.a) = 3;
-- Sample statement that you cannot execute. This is because columns from outer queries cannot be referenced in a SELECT statement for a subquery.
select * from t1 where (select count(*) from t2 where (select count(*) from t3 where t3.a = t1.a) = 2) = 3;
```

  - `SCALAR SUBQUERY` can be used only in a `WHERE` clause.

```
-- SCALAR SUBQUERY cannot be referenced in a SELECT statement for a subquery.
select * from t1 where (select t1.b + count(*) from t2) = 3;
-- SCALAR SUBQUERY cannot be referenced in a SELECT statement for an outer query.
select (select count(*) from t2 where t2.a = t1.a) from t1;
```

- Examples
  - Example 1: Common usage. The following statement shows an example:

```
select * from shop where (select count(*) from sale_detail where sale_detail.shop_name = shop.shop_name) >= 1;
```

    The following result is returned:

```
+------------+-------------+-------------+
| shop_name  | customer_id | total_price |
+------------+-------------+-------------+
| s1         | c1          | 100.1       |
| s2         | c2          | 100.2       |
| s3         | c3          | 100.3       |
| null       | c5          | NULL        |
| s6         | c6          | 100.4       |
| s7         | c7          | 100.5       |
+------------+-------------+-------------+
```

- Example 2: Multiple columns are specified in a SELECT statement for a subquery. The following statements show an example:

```
-- Sample data is reconstructed to help you understand this example.
create table if not exists ts(a bigint,b bigint,c double);
create table if not exists t(a bigint,b bigint,c double);
insert into table ts values (1,3,4.0),(1,3,3.0);
insert into table t values (1,3,4.0),(1,3,5.0);
-- Scenario 1: Columns in a SELECT list is a SCALAR SUBQUERY expression in which you sp
ecify multiple columns. The expression must be an equality expression. Sample statement
s that you cannot execute: select (select a, b from t where c > ts.c) as (a, b), a from
ts;
select (select a, b from t where c = ts.c) as (a, b), a from ts;
-- The following result is returned:
+------------+------------+------------+
| a          | b          | a2         |
+------------+------------+------------+
| 1          | 3          | 1          |
| NULL       | NULL       | 1          |
+------------+------------+------------+
-- Scenario 2: Columns in a SELECT list is an expression of the BOOLEAN type. Only equi
valent comparison is supported. Sample statements that you cannot execute: select (a,b)
> (select a,b from ts where c = t.c) from t;
select (a,b) = (select a,b from ts where c = t.c) from t;
-- The following result is returned:
+------+
| _c0  |
+------+
| true |
| false |
+------+
-- Scenario 3: A WHERE clause supports multi-column comparison. Only equivalent compari
son is supported. Sample statements that you cannot execute: select * from t where (a,b
) > (select a,b from ts where c = t.c);
select * from t where c > 3.0 and (a,b) = (select a,b from ts where c = t.c);
-- The following result is returned:
+------------+------------+------------+
| a          | b          | c          |
+------------+------------+------------+
| 1          | 3          | 4.0        |
+------------+------------+------------+
select * from t where c > 3.0 or (a,b) = (select a,b from ts where c = t.c);
-- The following result is returned:
+------------+------------+------------+
| a          | b          | c          |
+------------+------------+------------+
| 1          | 3          | 4.0        |
| 1          | 3          | 5.0        |
+------------+------------+------------+
```

3.6.4. INTERSECT, UNION, EXCEPT, and MINUS

This topic describes how to perform the following operations on datasets of query results in MaxCompute: `INTERSECT`, `INTERSECT ALL`, `INTERSECT DISTINCT`, `UNION`, `UNION ALL`, `UNION DISTINCT`, `EXCEPT`, `EXCEPT ALL`, `EXCEPT DISTINCT`, `MINUS`, `MINUS ALL`, and `MINUS DISTINCT`.

Description

MaxCompute supports the following operations on datasets:

- INTERSECT: returns the intersection of two datasets. The intersection includes the values that are contained in both datasets.
- UNION: returns the union of two datasets. The union is a dataset that is obtained by combining the two datasets.
- EXCEPT and MINUS: returns distinct values from one of two datasets. These values are not contained in the other dataset.

Limits

The following limits are imposed on INTERSECT, UNION, EXCEPT, and MINUS:

- MaxCompute allows you to perform INTERSECT, UNION, EXCEPT, or MINUS operations on a maximum of 256 datasets at the same time. If the number of datasets exceeds 256, an error is returned.
- The number of columns in the left and right tables must be the same.

Usage notes

When you perform INTERSECT, UNION, EXCEPT, or MINUS operations on two datasets, take note of the following items:

- The results of INTERSECT, UNION, EXCEPT, or MINUS operations may not be sorted in a specific order.
- If data types of the datasets are not consistent, MaxCompute implicitly converts the data types before you perform INTERSECT, UNION, EXCEPT, or MINUS operations. For more information about implicit conversions, see Data types. To prevent compatibility issues, MaxCompute disables implicit conversions between data of the STRING type and data of other types for INTERSECT, UNION, EXCEPT, or MINUS operations.

INTERSECT

- Syntax

```
-- Obtain an intersection that contains duplicate values.
<select_statement1> intersect all <select_statement2>;
-- Obtain an intersection that does not contain duplicate values. The usage of INTERSECT
is equivalent to that of INTERSECT DISTINCT.
<select_statement1> intersect [distinct] <select_statement2>;
```

- Parameters
  - select_statement1 and select_statement2: required. These parameters specify the `SELECT` clauses. For more information about the syntax of the clauses, see SELECT syntax.
  - distinct: optional. This parameter is used to remove duplicate values from the intersection of two datasets.
- Examples

- Example 1: Obtain the intersection of two datasets. The intersection contains duplicate values. Sample statement:

```
select * from values (1, 2), (1, 2), (3, 4), (5, 6) t(a, b)
intersect all
select * from values (1, 2), (1, 2), (3, 4), (5, 7) t(a, b);
```

The following result is returned:

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 1          | 2          |
| 3          | 4          |
+------------+------------+
```

- Example 2: Obtain the intersection of two datasets. The intersection does not contain duplicate values. Sample statements:

```
select * from values (1, 2), (1, 2), (3, 4), (5, 6) t(a, b)
intersect distinct
select * from values (1, 2), (1, 2), (3, 4), (5, 7) t(a, b);
-- The preceding statement is equivalent to the following statement:
select distinct * from
(select * from values (1, 2), (1, 2), (3, 4), (5, 6) t(a, b)
intersect all
select * from values (1, 2), (1, 2), (3, 4), (5, 7) t(a, b)) t;
```

The following result is returned:

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 3          | 4          |
+------------+------------+
```

UNION

- Syntax

```
-- Obtain a union that contains duplicate values.
<select_statement1> union all <select_statement2>;
-- Obtain a union that does not contain duplicate values.
<select_statement1> union [distinct] <select_statement2>;
```

- Usage notes
  - If multiple `UNION ALL` operations exist, use parentheses () to specify the priorities of the `UNION ALL` operations.

- If `UNION` is followed by a `CLUSTER BY`, `DISTRIBUTE BY`, `SORT BY`, `ORDER BY`, or `LIMIT` clause and `odps.sql.type.system.odps2` is set to false, the clause works only on the last `select_statement` of `UNION`. If `odps.sql.type.system.odps2` is set to true, the clause works on the results of all `UNION` operations.

- Parameters
  - select_statement1 and select_statement2: required. These parameters specify the `SELECT` clauses. For more information about the syntax of the clauses, see SELECT syntax.
  - distinct: optional. This parameter is used to remove duplicate values from the union of two datasets.

- Examples
  - Example 1: Obtain the union of two datasets. The union contains duplicate values. Sample statement:

    ```
    select * from values (1, 2), (1, 2), (3, 4) t(a, b)
    union all
    select * from values (1, 2), (1, 4) t(a, b);
    ```

    The following result is returned:

    ```
    +------------+------------+
    | a          | b          |
    +------------+------------+
    | 1          | 2          |
    | 1          | 2          |
    | 3          | 4          |
    | 1          | 2          |
    | 1          | 4          |
    +------------+------------+
    ```

  - Example 2: Obtain the union of two datasets. The union does not contain duplicate values. Sample statements:

    ```
    select * from values (1, 2), (1, 2), (3, 4) t(a, b)
    union distinct
    select * from values (1, 2), (1, 4) t(a, b);
    -- The preceding statement is equivalent to the following statement:
    select distinct * from (
    select * from values (1, 2), (1, 2), (3, 4) t(a, b)
    union all
    select * from values (1, 2), (1, 4) t(a, b));
    ```

    The following result is returned:

    ```
    +------------+------------+
    | a          | b          |
    +------------+------------+
    | 1          | 2          |
    | 1          | 4          |
    | 3          | 4          |
    +------------+------------+
    ```

- Example 3: Use parentheses () to specify the priorities of `UNION ALL` operations. Sample statement:

```
select * from values (1, 2), (1, 2), (5, 6) t(a, b)
union all
(select * from values (1, 2), (1, 2), (3, 4) t(a, b)
union all
select * from values (1, 2), (1, 4) t(a, b));
```

The following result is returned:

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 1          | 2          |
| 5          | 6          |
| 1          | 2          |
| 1          | 2          |
| 3          | 4          |
| 1          | 2          |
| 1          | 4          |
+------------+------------+
```

- Example 4: Use `UNION` that is followed by a `CLUSTER BY`, `DISTRIBUTE BY`, `SORT BY`, `ORDER BY`, or `LIMIT` clause in a SELECT statement and set `odps.sql.type.system.odps2` to true. Sample statements:

```
set odps.sql.type.system.odps2=true;
select explode(array(3, 1)) as (a) union all select explode(array(0, 4, 2)) as (a) order by a limit 3;
```

The following result is returned:

```
+------------+
| a          |
+------------+
| 0          |
| 1          |
| 2          |
+------------+
```

- Example 5: Use `UNION` that is followed by a `CLUSTER BY`, `DISTRIBUTE BY`, `SORT BY`, `ORDER BY`, or `LIMIT` clause in a SELECT statement and set `odps.sql.type.system.odps2` to false. Sample statements:

```
set odps.sql.type.system.odps2=false;
select explode(array(3, 1)) as (a) union all select explode(array(0, 4, 2)) as (a) order by a limit 3;
```

The following result is returned:

```
+------------+
| a          |
+------------+
| 3          |
| 1          |
| 0          |
| 2          |
| 4          |
+------------+
```

EXCEPT and MINUS

- Syntax

```
-- Obtain the supplementary set of two datasets. The supplementary set contains duplicate values.
<select_statement1> except all <select_statement2>;
<select_statement1> minus all <select_statement2>;
-- Obtain the supplementary set of two datasets. The supplementary set does not contain duplicate values.
<select_statement1> except [distinct] <select_statement2>;
<select_statement1> minus [distinct] <select_statement2>;
```

> **Note** The usage of `EXCEPT` is equivalent to that of `MINUS`.

- Parameters
  - select_statement1 and select_statement2: required. These parameters specify the `SELECT` clauses. For more information about the syntax of the clauses, see SELECT syntax.
  - distinct: optional. This parameter is used to remove duplicate values from the supplementary set of two datasets.
- Examples

- Example 1: Obtain the supplementary set of two datasets. The supplementary set contains duplicate values. Sample statements:

```
select * from values (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b)
except all
select * from values (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
-- The preceding statement is equivalent to the following statement:
select * from values (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b)
minus all
select * from values (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
```

The following result is returned:

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 1          | 2          |
| 3          | 4          |
| 7          | 8          |
+------------+------------+
```

- Example 2: Obtain the supplementary set of two datasets. The supplementary set does not contain duplicate values. Sample statements:

```
select * from values (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b)
except distinct
select * from values (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
-- The preceding statement is equivalent to the following statements:
select * from values (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b)
minus distinct
select * from values (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
-- The preceding statements are equivalent to the following statement:
select distinct * from values (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b) ex
cept all select * from values (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
```

The following result is returned:

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 7          | 8          |
+------------+------------+
```

3.6.5. JOIN

MaxCompute allows you to use `JOIN` operations to join tables and return the data that meets join and query conditions. This topic describes the following JOIN operations: LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN, INNER JOIN, NATURAL JOIN, implicit JOIN, and multiple JOIN operations.

Description

MaxCompute supports the following types of `JOIN` operations:

- `LEFT OUTER JOIN`

  It is also called `LEFT JOIN`. LEFT OUTER JOIN returns all rows in the left table, including the rows that do not match any rows in the right table.

  > **Note** In most cases, the left table is a large table, and the right table is a small table. If the values in some rows of the right table are duplicate, we recommend that you do not perform multiple consecutive `LEFT JOIN` operations. If you perform multiple consecutive LEFT JOIN operations, data expansion may occur and interrupt your jobs.

- `RIGHT OUTER JOIN`

  It is also called `RIGHT JOIN`. RIGHT OUTER JOIN returns all rows in the right table, including the rows that do not match any rows in the left table.

- `FULL OUTER JOIN`

  It is also called `FULL JOIN`. FULL OUTER JOIN returns all rows in both the left and right tables.

- `INNER JOIN`

  The `INNER` keyword can be omitted. `INNER JOIN` returns data rows if a match exists between the left and right tables. If no rows match, no result is returned.

- `NATURAL JOIN`

  In a `NATURAL JOIN` operation, the conditions that are used to join two tables are determined based on the common fields between the two tables. MaxCompute supports `OUTER NATURAL JOIN`. If you use the `USING` clause, the `JOIN` operation returns common fields only once.

- Implicit JOIN operation

  You can perform an implicit JOIN operation without the need to specify the `JOIN` keyword.

- Multiple JOIN operations

  MaxCompute supports multiple `JOIN` operations. You can use parentheses () to specify the priorities of `JOIN` operations. A `JOIN` operation that is enclosed in parentheses () has a higher priority.

> **Note** If an SQL statement contains the `WHERE` clause and you use the `JOIN` clause before the `WHERE` clause, the `JOIN` operation is performed first. Then, the results obtained from the `JOIN` operation are filtered based on the conditions specified by the `WHERE` clause. The final result is the intersection of two tables, not all rows in the left table.

Limits

Limits on the use of `JOIN`:

- MaxCompute does not support `CROSS JOIN`. A CROSS JOIN operation joins two tables without requiring you to specify conditions in the `ON` clause.
- You must use equi-joins and combine conditions by using `AND`. You can use non-equi joins or combine multiple conditions by using `OR` in a `MAPJOIN` operation. For more information, see MAPJOIN.

Syntax

```
<table_reference> join <table_factor> [<join_condition>]
| <table_reference> {left outer|right outer|full outer|inner|natural} join <table_reference
> <join_condition>
```

- table_reference: required. The query statement for the left table on which the `JOIN` operation is performed. The value of this parameter is in the `table_name [alias] | table_query [alias] |...` format.
- table_factor: required. The query statement for the right table or a table on which the `JOIN` operation is performed. The value of this parameter is in the `table_name [alias] | table_subquery [alias] |...` format.
- join_condition: optional. A `JOIN` condition is a combination of one or more equality expressions. The value of this parameter is in the `on equality_expression [and equality_expression]...` format. `equality_expression` is an equality expression.

> **Note** If partition pruning conditions are specified in the `WHERE` clause, partition pruning takes effect on both tables. If partition pruning conditions are specified in the `ON` clause, partition pruning takes effect only on the secondary table. As a result, a full table scan is run for the primary table. For more information, see Check whether partition pruning is effective.

Sample data

The following sample source data is provided to help you understand the examples in this topic. The following statements show how to create the sale_detail and sale_detail_jt tables and insert data into the tables.

```
-- Create two partitioned tables named sale_detail and sale_detail_jt.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
create table if not exists sale_detail_jt
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add partitions to the partitioned tables sale_detail and sale_detail_jt.
alter table sale_detail add partition (sale_date='2013', region='china') partition (sale_da
te='2014', region='shanghai');
alter table sale_detail_jt add partition (sale_date='2013', region='china');
-- Insert data into the partitioned tables sale_detail and sale_detail_jt.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null','c5'
,null),('s6','c6',100.4),('s7','c7',100.5);
insert into sale_detail_jt partition (sale_date='2013', region='china') values ('s1','c1',1
00.1),('s2','c2',100.2),('s5','c2',100.2);
-- Create a table that you want to join.
create table shop as select shop_name, customer_id, total_price from sale_detail;
```

Examples

- Example 1: LEFT OUTER JOIN. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Both the sale_detail_jt and sale_detail tables have the shop_name column. You must use
aliases to distinguish between the columns in the SELECT statement.
select a.shop_name as ashop, b.shop_name as bshop from sale_detail_jt a
       left outer join sale_detail b on a.shop_name=b.shop_name;
```

  Returned result:

```
+------------+------------+
| ashop      | bshop      |
+------------+------------+
| s2         | s2         |
| s1         | s1         |
| s5         | NULL       |
+------------+------------+
```

- Example 2: RIGHT OUTER JOIN. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Both the sale_detail_jt and sale_detail tables have the shop_name column. You must use
aliases to distinguish between the columns in the SELECT statement.
select a.shop_name as ashop, b.shop_name as bshop from sale_detail_jt a
       right outer join sale_detail b on a.shop_name=b.shop_name;
```

Returned result:

```
+------------+------------+
| ashop      | bshop      |
+------------+------------+
| NULL       | s3         |
| NULL       | s6         |
| NULL       | null       |
| s2         | s2         |
| NULL       | s7         |
| s1         | s1         |
+------------+------------+
```

- Example 3: FULL OUTER JOIN. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Both the sale_detail_jt and sale_detail tables have the shop_name column. You must use
aliases to distinguish between the columns in the SELECT statement.
select a.shop_name as ashop, b.shop_name as bshop from sale_detail_jt a
       full outer join sale_detail b on a.shop_name=b.shop_name;
```

Returned result:

```
+------------+------------+
| ashop      | bshop      |
+------------+------------+
| NULL       | s3         |
| NULL       | s6         |
| s2         | s2         |
| NULL       | null       |
| NULL       | s7         |
| s1         | s1         |
| s5         | NULL       |
+------------+------------+
```

- Example 4: INNER JOIN. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Both the sale_detail_jt and sale_detail tables have the shop_name column. You must use
aliases to distinguish between the columns in the SELECT statement.
select a.shop_name as ashop, b.shop_name as bshop from sale_detail_jt a
       inner join sale_detail b on a.shop_name=b.shop_name;
```

Returned result:

```
+------------+------------+
| ashop      | bshop      |
+------------+------------+
| s2         | s2         |
| s1         | s1         |
+------------+------------+
```

- Example 5: NATURAL JOIN. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Perform a NATURAL JOIN operation.
select * from sale_detail_jt natural join sale_detail;
-- The preceding statement is equivalent to the following statements.
select sale_detail_jt.shop_name as shop_name, sale_detail_jt.customer_id as customer_id,
sale_detail_jt.total_price as total_price, sale_detail_jt.sale_date as sale_date, sale_de
tail_jt.region as region from sale_detail_jt
inner join sale_detail
on sale_detail_jt.shop_name=sale_detail.shop_name and sale_detail_jt.customer_id=sale_det
ail.customer_id and sale_detail_jt.total_price=sale_detail.total_price and sale_detail_jt
.sale_date=sale_detail.sale_date and sale_detail_jt.region=sale_detail.region;
```

Returned result:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 6: implicit JOIN. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Perform an implicit JOIN operation.
select * from sale_detail_jt, sale_detail where sale_detail_jt.shop_name = sale_detail.sh
op_name;
-- The preceding statement is equivalent to the following statement.
select * from sale_detail_jt join sale_detail on sale_detail_jt.shop_name = sale_detail.s
hop_name;
```

Returned result:

```
+------------+------------+------------+------------+------------+------------+--------
------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     | shop_name2 | custome
r_id2 | total_price2 | sale_date2 | region2    |
+------------+------------+------------+------------+------------+------------+--------
------+-------------+------------+------------+
| s2         | c2          | 100.2       | 2013       | china      | s2         | c2
| 100.2        | 2013       | china      |
| s1         | c1          | 100.1       | 2013       | china      | s1         | c1
| 100.1        | 2013       | china      |
+------------+------------+------------+------------+------------+------------+--------
------+-------------+------------+------------+
```

- Example 7: Multiple JOIN operations. No priority is specified. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Both the sale_detail_jt and sale_detail tables have the shop_name column. You must use
aliases to distinguish between the columns in the SELECT statement.
select a.* from sale_detail_jt a full outer join sale_detail b on a.shop_name=b.shop_name
        full outer join sale_detail c on a.shop_name=c.shop_name;
```

Returned result:

```
+------------+------------+------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+------------+------------+------------+------------+
| s5         | c2          | 100.2       | 2013       | china      |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| s1         | c1          | 100.1       | 2013       | china      |
| NULL       | NULL        | NULL        | NULL       | NULL       |
| s2         | c2          | 100.2       | 2013       | china      |
| NULL       | NULL        | NULL        | NULL       | NULL       |
+------------+------------+------------+------------+------------+
```

- Example 7: Multiple JOIN operations. Use parentheses () to specify the priorities of JOIN operations. Sample statements:

```
-- The full table scan feature must be enabled for partition tables. Otherwise, the JOIN
operation fails.
set odps.sql.allow.fullscan=true;
-- Perform multiple JOIN operations. Use parentheses () to specify the priority.
select * from shop join (sale_detail_jt join sale_detail on sale_detail_jt.shop_name = sa
le_detail.shop_name) on shop.shop_name=sale_detail_jt.shop_name;
```

Returned result:

```
+------------+------------+------------+------------+------------+------------+--------
------+-------------+------------+------------+------------+-------------+------------
-+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     | shop_name2 | custome
r_id2 | total_price2 | sale_date2 | region2    | shop_name3 | customer_id3 | total_price3
| sale_date3 | region3    |
+------------+------------+------------+------------+------------+------------+--------
------+-------------+------------+------------+------------+-------------+------------
-+------------+------------+
| s2         | c2          | 100.2       | 2013       | china      | s2         | c2
| 100.2        | 2013       | china      | s2         | c2           | 100.2        | 201
3       | china      |
| s1         | c1          | 100.1       | 2013       | china      | s1         | c1
| 100.1        | 2013       | china      | s1         | c1           | 100.1        | 201
3       | china      |
+------------+------------+------------+------------+------------+------------+--------
------+-------------+------------+------------+------------+-------------+------------
-+------------+------------+
```

- Example 8: Use `JOIN` and `WHERE` to query the number of records whose region is china and whose shop_name field has the same value in the two tables. All records in the sale_detail table are retained. Sample statements:

```
-- The full table scan feature must be enabled for partitioned tables. Otherwise, the JOI
N operation fails.
set odps.sql.allow.fullscan=true;
-- Execute the following SQL statement.
select a.shop_name
        ,a.customer_id
        ,a.total_price
        ,b.total_price
from    (select * from sale_detail where region = "china") a
left join  (select * from sale_detail_jt where region = "china") b
on      a.shop_name = b.shop_name;
```

Returned result:

```
+------------+------------+------------+--------------+
| shop_name  | customer_id | total_price | total_price2 |
+------------+------------+------------+--------------+
| s1         | c1          | 100.1       | 100.1        |
| s2         | c2          | 100.2       | 100.2        |
| s3         | c3          | 100.3       | NULL         |
+------------+------------+------------+--------------+
```

Incorrect usage:

```
select a.shop_name
        ,a.customer_id
        ,a.total_price
        ,b.total_price
from    sale_detail a
left join  sale_detail_jt b
on      a.shop_name = b.shop_name
where   a.region = "china" and b.region = "china";
```

Returned result:

```
+------------+-------------+-------------+--------------+
| shop_name  | customer_id | total_price | total_price2 |
+------------+-------------+-------------+--------------+
| s1         | c1          | 100.1       | 100.1        |
| s2         | c2          | 100.2       | 100.2        |
+------------+-------------+-------------+--------------+
```

The returned result is the intersection of the two tables, not all rows in the sale_detail table.

3.6.6. SEMI JOIN

MaxCompute supports two types of SEMI JOIN operations: LEFT SEMI JOIN and LEFT ANTI JOIN. In SEMI JOIN, data in the right table does not appear in the result set and is only used to filter data in the left table. This topic describes how to use `LEFT SEMI JOIN` and `LEFT ANTI JOIN`.

Description

MaxCompute supports the following types of `SEMI JOIN` operations:

- `LEFT SEMI JOIN`

  A `LEFT SEMI JOIN` operation returns rows from the left table that has matching rows in the right table. For example, if one row of data in the left table meets a specified condition and the data appears in the right table, the data is included in the result set.

  In MaxCompute, the usage of `LEFT SEMI JOIN` is similar to that of `IN SUBQUERY`. For more information about IN SUBQUERY, see IN SUBQUERY. You can choose one of the two operations.

- `LEFT ANTI JOIN`

  A `LEFT ANTI JOIN` operation returns rows from the left table that does not have matching rows in the right table. For example, if one row of data in the left table meets a specified condition and the data does not appear in the right table, the data is included in the result set.

  In MaxCompute, the usage of `LEFT ANTI JOIN` is similar to that of `NOT IN SUBQUERY`. For more information about NOT IN SUBQUERY, see NOT IN SUBQUERY.

`SEMI JOIN` supports `MAPJOIN hints`. These hints improve the performance of `LEFT SEMI JOIN` and `LEFT ANTI JOIN`. For more information about `MAPJOIN hints`, see MAPJOIN hints.

Sample data

This topic provides sample source data and sample statements that are related to the data. This helps you understand how to prepare source data. The following statements describe how to create the sale_detail and sale_detail_sj tables and insert data into the tables.

```
-- Create the sale_detail and sale_detail_sj tables. The two tables are partitioned tables.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
create table if not exists sale_detail_sj
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add one partition to each table.
alter table sale_detail add partition (sale_date='2013', region='china');
alter table sale_detail_sj add partition (sale_date='2013', region='china');
-- Insert data into the two tables.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail_sj partition (sale_date='2013', region='china') values ('s1','c1',1
00.1),('s2','c2',100.2),('s5','c2',100.2),('s2','c2',100.2);
```

Examples

- Example 1: Query data of the total_price column from the sale_detail table and obtain the data that appears in the total_price column of the sale_detail_sj table. Sample statement:

  ```
  select * from sale_detail a left semi join sale_detail_sj b on a.total_price=b.total_pric
  e;
  ```

  The following result is returned:

  ```
  +------------+-------------+-------------+------------+------------+
  | shop_name  | customer_id | total_price | sale_date  | region     |
  +------------+-------------+-------------+------------+------------+
  | s2         | c2          | 100.2       | 2013       | china      |
  | s1         | c1          | 100.1       | 2013       | china      |
  +------------+-------------+-------------+------------+------------+
  ```

  Some data of the total_price column in the sale_detail table **appears** in the total_price column of the sale_detail_sj table and such data is returned.

- Example 2: Query data of the total_price column from the sale_detail table and obtain the data that does not appear in the total_price column of the sale_detail_sj table. Sample statement:

  ```
  select * from sale_detail a left anti join sale_detail_sj b on a.total_price=b.total_pric
  e;
  ```

  The following result is returned:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

Some data of the total_price column in the sale_detail table **does not appear** in the total_price column of the sale_detail_sj table and such data is returned.

3.6.7. MAPJOIN hints

This topic describes how to explicitly specify a `MAPJOIN` hint in a `SELECT` statement to `join` a large table with one or more small tables. The `MAPJOIN hint` speeds up your data queries.

Description

A JOIN operation involves three stages: map, shuffle, and reduce. In most cases, tables are `joined` in the reduce stage.

`MAPJOIN` joins tables in the map stage instead of the reduce stage. This accelerates data transmission, reduces system resource consumption, and optimizes the performance of jobs.

In the map stage, `MAPJOIN` loads all data in the specified small tables into the memory of the program that performs the `JOIN` operation. This improves the efficiency of the `JOIN` operation.

In MaxCompute SQL, you cannot use non-equi `joins` or the `OR` logic in the `ON` condition. However, you can do this in `MAPJOIN` .

Limits

Limits on the use of `MAPJOIN` :

- In the map stage, `MAPJOIN` loads all data in the specified tables into the memory of the program that performs the JOIN operation. The tables specified for MAPJOIN must be small tables, and the total memory occupied by the table data cannot exceed 512 MB. MaxCompute compresses your data before storage. After the tables are loaded into the memory, the data volume of small tables sharply increases. 512 MB indicates the maximum data volume after small tables are loaded into the memory.
- Limits on `JOIN` operations in `MAPJOIN` :
  - The left table in a `LEFT OUTER JOIN` operation must be a large table.
  - The right table in a `RIGHT OUTER JOIN` operation must be a large table.
  - MAPJOIN cannot be used in a `FULL OUTER JOIN` operation.
  - The left or right table in an `INNER JOIN` operation can be a large table.
- MaxCompute allows you to specify a maximum of 128 small tables for `MAPJOIN` . If you specify more than 128 small tables, a syntax error is returned.

Usage notes

You can execute `MAPJOIN` only after you add the `/*+ mapjoin(<table_name>) */` hint to a `SELECT` statement. Take note of the following items:

- When you reference a small table or a subquery, you must reference the alias of the table or

subquery.

- In `MAPJOIN`, you can use subqueries as small tables.
- In `MAPJOIN`, you can use non-equi joins or combine conditions by using `OR`. You can calculate the Cartesian product by using `MAPJOIN ON 1 = 1` without specifying the `ON` condition, for example, `select /*+ mapjoin(a) */ a.id from shop a join table_name b on 1=1;`. However, this calculation method may increase the data volume.
- Separate multiple small tables in `MAPJOIN` with commas (,), for example, `/*+ mapjoin(a,b,c)*/`.

Sample data

This topic provides sample source data and sample statements that are related to the data. This helps you understand how to prepare source data. The following statements describe how to create the sale_detail and sale_detail_sj tables and insert data into the tables.

```
-- Create the sale_detail and sale_detail_sj tables. The two tables are partitioned tables.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
create table if not exists sale_detail_sj
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add one partition to each table.
alter table sale_detail add partition (sale_date='2013', region='china');
alter table sale_detail_sj add partition (sale_date='2013', region='china');
-- Insert data into the two tables.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail_sj partition (sale_date='2013', region='china') values ('s1','c1',1
00.1),('s2','c2',100.2),('s5','c2',100.2),('s2','c2',100.2);
```

Example

Perform `JOIN` operations on the sale_detail and the sale_detail_sj tables. The operations must meet one of the following conditions: 1. The total values of the total_price column in the sale_detail_sj table are less than the total values of the total_price column in the sale_detail table. 2. The sum of values of the total_price column in the sale_detail_sj table and values of the total_price column in the sale_detail table is less than 500. Sample statement:

```
select /*+ mapjoin(a) */
        a.shop_name,
        a.total_price,
        b.total_price
from sale_detail_sj a join sale_detail b
on a.total_price < b.total_price or a.total_price + b.total_price < 500;
```

The following result is returned:

```
+-----------+-------------+--------------+
| shop_name | total_price | total_price2 |
+-----------+-------------+--------------+
| s1        | 100.1       | 100.1        |
| s2        | 100.2       | 100.1        |
| s5        | 100.2       | 100.1        |
| s2        | 100.2       | 100.1        |
| s1        | 100.1       | 100.2        |
| s2        | 100.2       | 100.2        |
| s5        | 100.2       | 100.2        |
| s2        | 100.2       | 100.2        |
| s1        | 100.1       | 100.3        |
| s2        | 100.2       | 100.3        |
| s5        | 100.2       | 100.3        |
| s2        | 100.2       | 100.3        |
+-----------+-------------+--------------+
```

3.6.8. DISTRIBUTED MAPJOIN

DISTRIBUTED MAPJOIN is an optimized version of MAPJOIN. You can use DISTRIBUTED MAPJOIN when you join a small table with a large table. You can use DISTRIBUTED MAPJOIN and MAPJOIN to reduce shuffling and sorting on the large table.

Precautions

- The sizes of the tables that you want to join must be different. The size of the large table must be greater than 10 TB, and the size of the small table must be within the range of [1 GB, 100 GB].
- Data in the small table must be evenly distributed. If a small table contains long tails, excessive data is generated in a single shard of the table. As a result, an out of memory (OOM) error and the remote procedure call (RPC) timeout issue may occur.
- If an SQL task runs more than 20 minutes, we recommend that you use DISTRIBUTED MAPJOIN for optimization.
- Excessive resources are occupied when a task is running. Therefore, we recommend that you do not run a task in a small quota group.

> **Note** On the **Quotas** page, you can change the quota group. For more information, see Configure quota groups.

Use DISTRIBUTED MAPJOIN

To use `DISTRIBUTED MAPJOIN`, you must add the hint `/*+distmapjoin(<table_name>(shard_count=<n>,replica_count=<m>))*/` to a `SELECT` statement. Both the shard_count and replica_count parameters are used to determine the parallelism of tasks. Formula: `Parallelism of tasks = shard_count × replica_count`.

- Parameters
  - table_name: the name of the small table that you want to join.
  - shard_count=<n>: the number of data shards of the small table that you want to join. The data shards of the small table are distributed on each compute node for data processing. n: the number of shards. In most cases, this parameter is set to an odd number.

    > **Note**
    > - We recommend that you manually specify the shard_count parameter. You can estimate the value of the shard_count parameter based on the size of the small table. The estimated data amount that is processed by a single shard node is within the range of [200 MB, 500 MB].
    > - If you set the shard_count parameter to an excessively large value, the data processing performance and stability are affected. If you set the shard_count parameter to an excessively small value, an error may occur due to the excessive use of memory.

  - replica_count=<m>: the number of replicas of the small table. m indicates the number of replicas. Default value: 1.

    > **Note** To reduce access pressure and prevent the failure of the entire task caused by the failure of a single node, you can create multiple replicas of data in the same shard. If a node frequently restarts because the parallelism of tasks is high or the environment is unstable, you can increase the value of the replica_count parameter. You can set this parameter to 2 or 3.

- Syntax

```
-- Recommended. Specify the shard_count parameter and retain the default value 1 for the replica_count parameter.
/*+distmapjoin(a(shard_count=5))*/
-- Recommended. Specify the shard_count and replica_count parameters.
/*+distmapjoin(a(shard_count=5,replica_count=2))*/
-- Use DISTRIBUTED MAPJOIN to join multiple small tables.
/*+distmapjoin(a(shard_count=5,replica_count=2),b(shard_count=5,replica_count=2)) */
-- Use DISTRIBUTED MAPJOIN and MAPJOIN together.
/*+distmapjoin(a(shard_count=5,replica_count=2)),mapjoin(b)*/
```

Example

This example describes how to use DISTRIBUTED MAPJOIN when you insert data into the partitioned table tmall_dump_lasttable.

- Standard syntax

```
insert OVERWRITE table tmall_dump_lasttable partition(ds='20211130')
select t1.*
from
(
    select nid, doc,type
    from search_ods.dump_lasttable where ds='20211203'
)t1
join
(
    select distinct item_id
    from tbcdm.dim_tb_itm
    where ds='20211130'
    and bc_type='B'
    and is_online='Y'
)t2
on t1.nid=t2.item_id;
```

- Syntax after optimization

```
insert OVERWRITE table tmall_dump_lasttable partition (ds='20211130')
select /*+ distmapjoin(t2(shard_count=35)) */ t1.*
from
(
    select nid, doc, type
    from search_ods.dump_lasttable where ds='20211203'
)t1
join
(
    select distinct item_id
    from tbcdm.dim_tb_itm
    where ds='20211130'
    and bc_type='B'
    and is_online='Y'
)t2
on t1.nid=t2.item_id;
```

3.6.9. SKEWJOIN HINT

If two tables you want to join have hot key values, a long tail may occur. In this case, you can use SKEWJOIN HINT to automatically or manually extract hot key values from the two tables, separately calculate the join result of hot key values and the join result of non-hot key values, and then join the calculated data. This accelerates the JOIN operation.

Usage

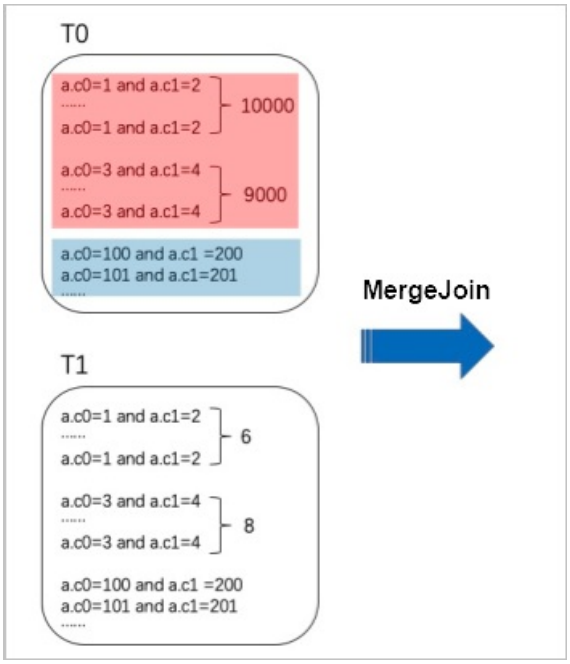
MAP JOIN can be performed only after you add the SKEWJOIN HINT `/*+ skewJoin(<table_name>[(<column1_name>[,<column2_name>,...])][((<value11>,<value12>)[,(<value21>,<value22>)...])] */` to the SELECT statement. In this hint, table_name indicates the name of a skewed table, column_name indicates the name of a skewed column, and value indicates a skewed key value.

```
-- Method 1: Include the alias of the table in SKEWJOIN HINT.
select /*+ skewjoin(a) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1;
-- Method 2: Include the table name and possibly skewed columns in SKEWJOIN HINT. In the fo
llowing statement, the c0 and c1 columns of table a are skewed columns.
select /*+ skewjoin(a(c0, c1)) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1 and
a.c2 = b.c2;
-- Method 3: Include the table name, columns, and skewed hot key values in SKEWJOIN HINT. I
f skewed key values are of the STRING type, enclose each value with double quotation marks.
In the following statement, (a.c0=1 and a.c1="2") and (a.c0=3 and a.c1="4") contain skewed
hot key values.
select /*+ skewjoin(a(c0, c1)((1, "2"), (3, "4"))) */ * from T0 a join T1 b on a.c0 = b.c0
and a.c1 = b.c1 and a.c2 = b.c2;
```

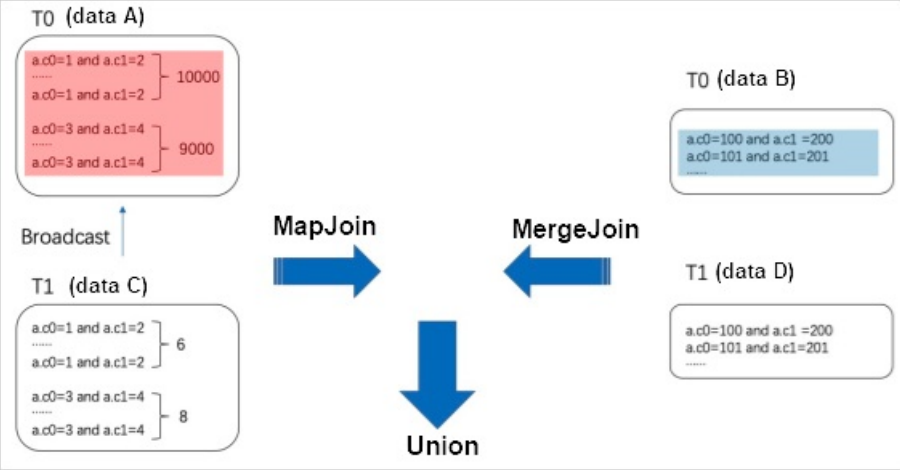
Note Method 3 is more efficient than Method 1 and Method 2.

Implementation

Hot key values are the key values that appear multiple times in a table. In the following figure, the red part has 10,000 records of `a.c0=1 and a.c1=2` and 9,000 records of `a.c0=3 and a.c1 = 4`.



If SKEWJOIN HINT is not used and the T0 and T1 tables contain large amounts of data, only the MERGE JOIN statement can be executed on the two tables. In this case, the same hot key values are shuffled to a single node. As a result, data skew occurs. After SKEWJOIN HINT is used, the optimizer runs an Aggregate to dynamically obtain top 20 hot key values based on the number of their records. The optimizer separately extracts a hot key value (data A) and a non-hot key value (data B) from the T0 table, and separately extracts a value (data C) that can be joined with data A and a value (data D) that cannot be joined with data A from the T1 table. Then, the MAP JOIN statement is executed on data A and data C (with a small data volume) and the MERGE JOIN statement is executed on data B and data D. The UNION statement is executed on the results of the MAP JOIN and MERGE JOIN statements to generate the final result, as shown in the following figure.



Precautions

- When you use SKEWJOIN HINT for JOIN statements, take note of the following limits:
  - INNER JOIN: SKEWJOIN HINT can be specified for the left or right table in the INNER JOIN statement.
  - LEFT JOIN, SEMI JOIN, and ANTI JOIN: SKEWJOIN HINT can be specified only for the left table.
  - RIGHT JOIN: SKEWJOIN HINT can be specified only for the right table.
  - FULL JOIN: SKEWJOIN HINT is not supported.
- We recommend that you add SKEWJOIN HINT only to the JOIN statements that will cause data skew. This is because an Aggregate is run after SKEWJOIN HINT is added, which slows down the JOIN operation.
- The data type of Left Side Join Key must be the same as that of Right Side Join Key for the JOIN statement to which SKEWJOIN HINT is added. If the data types are different, SKEWJOIN HINT becomes ineffective. In the preceding example, the data types of a.c0 and b.c0 must be the same and the data types of a.c1 and b.c1 must be the same. To ensure data type consistency, you can use the CAST function to convert the join keys in subqueries. Examples:

```
create table T0(c0 int, c1 int, c2 int, c3 int);
create table T1(c0 string, c1 int, c2 int);
-- Method 1:
select /*+ skewjoin(a) */ * from T0 a join T1 b on cast(a.c0 as string) = cast(b.c0 as st
ring) and a.c1 = b.c1;
-- Method 2:
select /*+ skewjoin(b) */ * from (select cast(a.c0 as string) as c00 from T0 a) b join T1
c on b.c00 = c.c0;
```

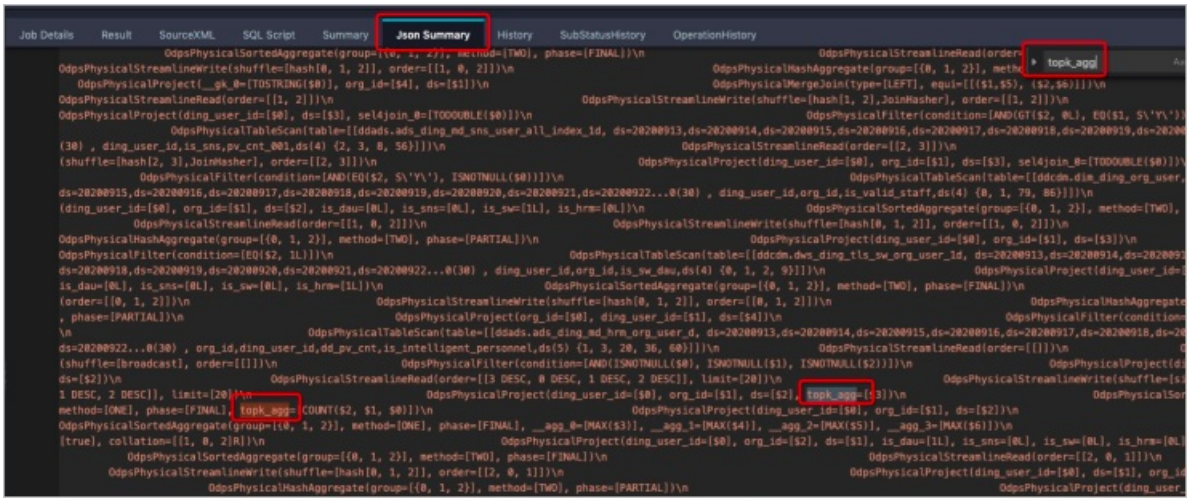
- After SKEWJOIN HINT is added, the optimizer runs an Aggregate to obtain top 20 hot key values, by

default. You can run the `set odps.optimizer.skew.join.topk.num = xx;` command to specify the number of hot key values that the optimizer can obtain.

- SKEWJOIN HINT allows you to add a hint only for the left or right table involved in the JOIN statement.
- In the JOIN statement to which SKEWJOIN HINT is added, `left key = right key` must be included. SKEWJOIN HINT cannot be added to the CARTESIAN JOIN statement.
- The following statement shows how to use SKEWJOIN HINT with other hints. Note that SKEWJOIN HINT cannot be added to the JOIN statement to which MAPJOIN HINT is added.

```
select /*+ mapjoin(c), skewjoin(a) */ * from T0 a join T1 b on a.c0 = b.c3 join T2 c on a
.c0 = c.c7;
```

- On the **Json Summary** tab of Logview, you can search for the topk_agg field. If such a field exists, as shown in the following figure, SKEWJOIN HINT has taken effect.



3.6.10. Lateral View

MaxCompute allows you to use a LATERAL VIEW clause and a user-defined table-valued function (UDTF) to split one row of data into multiple rows of data. This topic describes how to use the LITERAL VIEW clause to split a row of data and aggregate the split data.

Introduction

If you directly use a UDTF in a `SELECT` statement, issues may occur. To address these issues, you can use a LATERAL VIEW clause with a UDTF to split a row of data into multiple rows and aggregate the split data.

If the UDTF that you defined does not generate rows, the related input rows remain in the output of the LATERAL VIEW clause. In addition, the values of all columns that are generated by the UDTF are NULL.

Syntax

```
lateralView: lateral view [outer] <udtf_name>(<expression>) <table_alias> as <columnAlias>
(',' <columnAlias>)
fromClause: from <baseTable> (lateralView) [(lateralView) ...]
```

- udtf_name: required. This parameter specifies the name of the UDTF that splits a row of data into

multiple rows and aggregates the split data. For more information, see Other functions.

- expression: required. This parameter specifies the name of the column to which the row data that you want to split belongs.
- table_alias: required. This parameter specifies the alias of the result set of the UDTF.
- columnAlias: required. This parameter specifies the alias of the column that is obtained after data splitting.
- baseTable: required. This parameter specifies the name of the source table.

> **Note**
>
> Multiple LATERAL VIEW clauses may follow a `FROM` clause. The LATERAL VIEW clauses can reference the aliases of the tables and columns that are listed before the FROM clause. This aims to split row data in different columns.

Sample data

For example, the pageAds table contains three columns. The first column is pageid string. The second column is col1 array<int>. The third column is col2 array<string>. The following table provides the data in this table.

pageid	col1	col2
front_page	[1, 2, 3]	["a", "b", "c"]
contact_page	[3, 4, 5]	["d", "e", "f"]

Examples

- Use one LATERAL VIEW clause.
  - Example 1: Use a LATERAL VIEW clause to split col1. Sample code:

```
select pageid, col1_new, col2 from pageAds lateral view explode(col1) adTable as col1_n
ew;
```

Returned result:

```
+------------+------------+------------+
 pageid      col1_new    col2
+------------+------------+------------+
 front_page  1           ["a","b","c"]
 front_page  2           ["a","b","c"]
 front_page  3           ["a","b","c"]
 contact_page  3           ["d","e","f"]
 contact_page  4           ["d","e","f"]
 contact_page  5           ["d","e","f"]
+------------+------------+------------+
```

- Example 2: Use a LATERAL VIEW clause to split col1 and aggregate the split data. Sample code:

```
select col1_new, count(1) as count from pageAds lateral view explode(col1) adTable as c
ol1_new group by col1_new;
```

Returned result:

```
+------------+------------+
 col1_new    count
+------------+------------+
 1           1
 2           1
 3           2
 4           1
 5           1
+------------+------------+
```

- Use multiple LATERAL VIEW clauses.

  Use multiple LATERAL VIEW clauses to split col1 and col2. Sample code:

```
select pageid,mycol1, mycol2 from pageAds
    lateral view explode(col1) myTable1 as mycol1
    lateral view explode(col2) myTable2 as mycol2;
```

Returned result:

```
+------------+------------+------------+
 pageid      mycol1      mycol2
+------------+------------+------------+
 front_page  1           a
 front_page  1           b
 front_page  1           c
 front_page  2           a
 front_page  2           b
 front_page  2           c
 front_page  3           a
 front_page  3           b
 front_page  3           c
 contact_page  3           d
 contact_page  3           e
 contact_page  3           f
 contact_page  4           d
 contact_page  4           e
 contact_page  4           f
 contact_page  5           d
 contact_page  5           e
 contact_page  5           f
+------------+------------+------------+
```

References

To transpose data from rows to columns or from columns to rows during business development, you can also view the examples in Transpose rows to columns or columns to rows.

3.6.11. HAVING clause

In MaxCompute SQL, the WHERE keyword cannot be used together with aggregate functions. In this case, you can use the HAVING clause.

Write an SQL SELECT statement that contains a HAVING clause in the following syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

For example, a table named `Orders` contains the following fields: `Customer`, `OrderPrice`, `Order_date`, and `Order_id`. If you want to search for customers whose total order amount is less than 2,000, you can execute the following SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

3.6.12. GROUPING SETS

In some cases, you need to execute a `UNION ALL` clause multiple times to aggregate and analyze data from multiple dimensions. For example, if you want to aggregate Column a, aggregate Column b, and then aggregate Column a and Column b, you can use `GROUPING SETS` to perform these operations. This topic describes how to use `GROUPING SETS` for multidimensional aggregation.

Description

`GROUPING SETS` is an extension of a `GROUP BY` clause in a `SELECT` statement. The GROUPING SETS clause allows you to group your results in multiple ways, without the need to execute multiple `SELECT` statements and `UNION ALL` in sequence. This allows MaxCompute to generate more efficient execution plans with higher performance.

The following table describes the syntax that is associated with `GROUPING SETS`.

Type	Description

Type	Description

`CUBE`	A special form of `GROUPING SETS`, CUBE lists all the possible combinations of specified columns as `grouping sets`. You can also use CUBE with `GROUPING SETS`.
	`group by cube (a, b, c)`
	`-- Equivalent to the following clauses:`
	`grouping sets ((a,b,c),(a,b),(a,c),(b,c),(a),(b),(c),())`
	`group by cube ( (a, b), (c, d) )`
	`-- Equivalent to the following clauses:`
	`grouping sets (`
	`( a, b, c, d ),`
	`( a, b ),`
	`( c, d ),`
	`( )`
	`)`
	`group by a, cube (b, c), grouping sets ((d), (e))`
	`-- Equivalent to the following clause:`
	`group by grouping sets (`
	`(a, b, c, d), (a, b, c, e),`
	`(a, b, d), (a, b, e),`
	`(a, c, d), (a, c, e),`
	`(a, d), (a, e)`
	`)`

`ROLLUP`	A special form of `GROUPING SETS`. ROLLUP aggregates specified columns and generates `grouping sets` in a hierarchical manner. You can also use ROLLUP with `GROUPING SETS`.
	`group by rollup (a, b, c)`
	`-- Equivalent to the following clauses:`
	`grouping sets ((a,b,c),(a,b),(a), ())`
	`group by rollup ( a, (b, c), d )`
	`-- Equivalent to the following clauses:`
	`grouping sets (`
	`( a, b, c, d ),`
	`( a, b, c ),`
	`( a ),`
	`( )`
	`)`
	`group by grouping sets((b), (c), rollup(a,b,c))`
	`-- Equivalent to the following clause:`
	`group by grouping sets (`
	`(b), (c),`
	`(a,b,c), (a,b), (a), ()`
	`)`

Type	Description
`GROUPING`	NULL is used as placeholders in the results of `GROUPING SETS`. As a result, NULL placeholders cannot be distinguished from NULL values. To address this issue, MaxCompute provides `GROUPING`. `GROUPING` allows you to use the name of a column as a parameter. If specific rows are aggregated based on a column, 0 is returned. In this case, NULL is used as a value. If specific rows are not aggregated based on a column, 1 is returned. In this case, NULL is used as a placeholder.
`GROUPING_ID`	GROUPING_ID allows you to use the names of one or more columns as parameters. The `grouping` results of columns are used to bitmap integer values.
`GROUPING__ID`	`GROUPING__ID` does not require parameters. It is used for Hive-compatible queries. GROUPING__ID is equivalent to `GROUPING__ID(GROUP BY Parameter list)` in MaxCompute. The parameters of GROUPING__ID are in the same order as `GROUP BY`. **Note** If you use Hive 2.3.0 or later, we recommend that you use this function in MaxCompute. If you use a Hive version earlier than 2.3.0, we recommend that you do not use this function in MaxCompute.

Examples

Example for using `GROUPING SETS`:

1. Prepare data.

```
create table requests lifecycle 20 as
select * from values
    (1, 'windows', 'PC', 'Beijing'),
    (2, 'windows', 'PC', 'Shijiazhuang'),
    (3, 'linux', 'Phone', 'Beijing'),
    (4, 'windows', 'PC', 'Beijing'),
    (5, 'ios', 'Phone', 'Shijiazhuang'),
    (6, 'linux', 'PC', 'Beijing'),
    (7, 'windows', 'Phone', 'Shijiazhuang')
as t(id, os, device, city);
```

2. Use one of the following methods to group data:
   - Execute multiple `SELECT` statements to group data.

```
select NULL, NULL, NULL, count(*)
from requests
union all
select os, device, NULL, count(*)
from requests group by os, device
union all
select null, null, city, count(*)
from requests group by city;
```

- Use `GROUPING SETS` to group data.

```
select os,device, city ,count(*)
from requests
group by grouping sets((os, device), (city), ());
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | _c3        |
+------------+------------+------------+------------+
| NULL       | NULL       | NULL       | 7          |
| NULL       | NULL       | Beijing    | 4          |
| NULL       | NULL       | Shijiazhuang | 3          |
| ios        | Phone      | NULL       | 1          |
| linux      | PC         | NULL       | 1          |
| linux      | Phone      | NULL       | 1          |
| windows    | PC         | NULL       | 3          |
| windows    | Phone      | NULL       | 1          |
+------------+------------+------------+------------+
```

Note If some expressions are not used in GROUPING SETS, NULL is used as placeholders for these expressions, for example, NULL of the city column in the fourth row to the eighth row. This way, you can perform operations on the result sets.

Examples for using CUBE or ROLLUP

Examples for using `CUBE` or `ROLLUP` based on the syntax of `GROUPING SETS`:

- Example 1: Use `CUBE` to list all the possible columns os, device, and city as `grouping sets`. Sample statement:

```
select os,device, city, count(*)
from requests
group by cube (os, device, city);
-- The preceding statement is equivalent to the following statement:
select os,device, city, count(*)
from requests
group by grouping sets ((os, device, city),(os, device),(os, city),(device,city),(os),(de
vice),(city),());
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | _c3        |
+------------+------------+------------+------------+
| NULL       | NULL       | NULL       | 7          |
| NULL       | NULL       | Beijing    | 4          |
| NULL       | NULL       | Shijiazhuang | 3          |
| NULL       | PC         | NULL       | 4          |
| NULL       | PC         | Beijing    | 3          |
| NULL       | PC         | Shijiazhuang | 1          |
| NULL       | Phone      | NULL       | 3          |
| NULL       | Phone      | Beijing    | 1          |
| NULL       | Phone      | Shijiazhuang | 2          |
| ios        | NULL       | NULL       | 1          |
| ios        | NULL       | Shijiazhuang | 1          |
| ios        | Phone      | NULL       | 1          |
| ios        | Phone      | Shijiazhuang | 1          |
| linux      | NULL       | NULL       | 2          |
| linux      | NULL       | Beijing    | 2          |
| linux      | PC         | NULL       | 1          |
| linux      | PC         | Beijing    | 1          |
| linux      | Phone      | NULL       | 1          |
| linux      | Phone      | Beijing    | 1          |
| windows    | NULL       | NULL       | 4          |
| windows    | NULL       | Beijing    | 2          |
| windows    | NULL       | Shijiazhuang | 2          |
| windows    | PC         | NULL       | 3          |
| windows    | PC         | Beijing    | 2          |
| windows    | PC         | Shijiazhuang | 1          |
| windows    | Phone      | NULL       | 1          |
| windows    | Phone      | Shijiazhuang | 1          |
+------------+------------+------------+------------+
```

- Example 2: Use `CUBE` to list all the possible combinations of columns, `(os, device),(device, city)`, as `grouping sets`. Sample statement:

```
select os,device, city, count(*)
from requests
group by cube ((os, device), (device, city));
-- The preceding statement is equivalent to the following statement:
select os,device, city, count(*)
from requests
group by grouping sets ((os, device, city),(os, device),(device,city),());
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | _c3        |
+------------+------------+------------+------------+
| NULL       | NULL       | NULL       | 7          |
| NULL       | PC         | Beijing    | 3          |
| NULL       | PC         | Shijiazhuang | 1          |
| NULL       | Phone      | Beijing    | 1          |
| NULL       | Phone      | Shijiazhuang | 2          |
| ios        | Phone      | NULL       | 1          |
| ios        | Phone      | Shijiazhuang | 1          |
| linux      | PC         | NULL       | 1          |
| linux      | PC         | Beijing    | 1          |
| linux      | Phone      | NULL       | 1          |
| linux      | Phone      | Beijing    | 1          |
| windows    | PC         | NULL       | 3          |
| windows    | PC         | Beijing    | 2          |
| windows    | PC         | Shijiazhuang | 1          |
| windows    | Phone      | NULL       | 1          |
| windows    | Phone      | Shijiazhuang | 1          |
+------------+------------+------------+------------+
```

- Example 3: Use `ROLLUP` to aggregate the os, device, and city columns in a hierarchical manner to generate multiple `grouping sets`. Sample statement:

```
select os,device, city, count(*)
from requests
group by rollup (os, device, city);
-- The preceding statement is equivalent to the following statement:
select os,device, city, count(*)
from requests
group by grouping sets ((os, device, city),(os, device),(os),());
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | _c3        |
+------------+------------+------------+------------+
| NULL       | NULL       | NULL       | 7          |
| ios        | NULL       | NULL       | 1          |
| ios        | Phone      | NULL       | 1          |
| ios        | Phone      | Shijiazhuang | 1          |
| linux      | NULL       | NULL       | 2          |
| linux      | PC         | NULL       | 1          |
| linux      | PC         | Beijing    | 1          |
| linux      | Phone      | NULL       | 1          |
| linux      | Phone      | Beijing    | 1          |
| windows    | NULL       | NULL       | 4          |
| windows    | PC         | NULL       | 3          |
| windows    | PC         | Beijing    | 2          |
| windows    | PC         | Shijiazhuang | 1          |
| windows    | Phone      | NULL       | 1          |
| windows    | Phone      | Shijiazhuang | 1          |
+------------+------------+------------+------------+
```

- Example 4: Use `ROLLUP` to aggregate `os, (os,device), city` in a hierarchical manner to generate multiple `grouping sets`. Sample statement:

```
select os,device, city, count(*)
from requests
group by rollup (os, (os,device), city);
-- The preceding statement is equivalent to the following statement:
select os,device, city, count(*)
from requests
group by grouping sets ((os, device, city),(os, device),(os),());
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | _c3        |
+------------+------------+------------+------------+
| NULL       | NULL       | NULL       | 7          |
| ios        | NULL       | NULL       | 1          |
| ios        | Phone      | NULL       | 1          |
| ios        | Phone      | Shijiazhuang | 1          |
| linux      | NULL       | NULL       | 2          |
| linux      | PC         | NULL       | 1          |
| linux      | PC         | Beijing    | 1          |
| linux      | Phone      | NULL       | 1          |
| linux      | Phone      | Beijing    | 1          |
| windows    | NULL       | NULL       | 4          |
| windows    | PC         | NULL       | 3          |
| windows    | PC         | Beijing    | 2          |
| windows    | PC         | Shijiazhuang | 1          |
| windows    | Phone      | NULL       | 1          |
| windows    | Phone      | Shijiazhuang | 1          |
+------------+------------+------------+------------+
```

- Example 5: Use `GROUP BY`, `CUBE`, and `GROUPING SETS` to generate multiple `grouping sets`. Sample statement:

```
select os,device, city, count(*)
from requests
group by os, cube(os,device), grouping sets(city);
-- The preceding statement is equivalent to the following statement:
select os,device, city, count(*)
from requests
group by grouping sets((os,device,city),(os,city),(os,device,city));
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | _c3        |
+------------+------------+------------+------------+
| ios        | NULL       | Shijiazhuang | 1          |
| ios        | Phone      | Shijiazhuang | 1          |
| linux      | NULL       | Beijing    | 2          |
| linux      | PC         | Beijing    | 1          |
| linux      | Phone      | Beijing    | 1          |
| windows    | NULL       | Beijing    | 2          |
| windows    | NULL       | Shijiazhuang | 2          |
| windows    | PC         | Beijing    | 2          |
| windows    | PC         | Shijiazhuang | 1          |
| windows    | Phone      | Shijiazhuang | 1          |
+------------+------------+------------+------------+
```

Example for using GROUPING and GROUPING_ID

Sample statement:

```
select a,b,c,count(*),
grouping(a) ga, grouping(b) gb, grouping(c) gc, grouping_id(a,b,c) groupingid
from values (1,2,3) as t(a,b,c)
group by cube(a,b,c);
```

The following result is returned:

```
+------------+------------+------------+------------+------------+------------+------------
+------------+
| a          | b          | c          | _c3        | ga         | gb         | gc
| groupingid |
+------------+------------+------------+------------+------------+------------+------------
+------------+
| NULL       | NULL       | NULL       | 1          | 1          | 1          | 1
| 7          |
| NULL       | NULL       | 3          | 1          | 1          | 1          | 0
| 6          |
| NULL       | 2          | NULL       | 1          | 1          | 0          | 1
| 5          |
| NULL       | 2          | 3          | 1          | 1          | 0          | 0
| 4          |
| 1          | NULL       | NULL       | 1          | 0          | 1          | 1
| 3          |
| 1          | NULL       | 3          | 1          | 0          | 1          | 0
| 2          |
| 1          | 2          | NULL       | 1          | 0          | 0          | 1
| 1          |
| 1          | 2          | 3          | 1          | 0          | 0          | 0
| 0          |
+------------+------------+------------+------------+------------+------------+------------
+------------+
```

By default, the columns that are not specified in `GROUP BY` are filled with NULL. You can use `GROUPING` to specify the values that you require. Sample statement that is based on the syntax of `GROUPING SETS`:

```
select
  if(grouping(os) == 0, os, 'ALL') as os,
  if(grouping(device) == 0, device, 'ALL') as device,
  if(grouping(city) == 0, city, 'ALL') as city,
  count(*) as count
from requests
group by os, device, city grouping sets((os, device), (city), ());
```

The following result is returned:

```
+------------+------------+------------+------------+
| os         | device     | city       | count      |
+------------+------------+------------+------------+
| ALL        | ALL        | ALL        | 7          |
| ALL        | ALL        | Beijing    | 4          |
| ALL        | ALL        | Shijiazhuang | 3          |
| ios        | Phone      | ALL        | 1          |
| linux      | PC         | ALL        | 1          |
| linux      | Phone      | ALL        | 1          |
| windows    | PC         | ALL        | 3          |
| windows    | Phone      | ALL        | 1          |
+------------+------------+------------+------------+
```

Example for using GROUPING__ID:

Example for using `GROUPING__ID` without parameters specified:

```
select
a, b, c, count(*), grouping__id
from values (1,2,3) as t(a,b,c)
group by a, b, c grouping sets ((a,b,c), (a));
-- The preceding statement is equivalent to the following statement:
select
a, b, c, count(*), grouping_id(a,b,c)
from values (1,2,3) as t(a,b,c)
group by a, b, c grouping sets ((a,b,c), (a));
```

The following result is returned:

```
+------------+------------+------------+------------+------------+
| a          | b          | c          | _c3        | _c4        |
+------------+------------+------------+------------+------------+
| 1          | NULL       | NULL       | 1          | 3          |
| 1          | 2          | 3          | 1          | 0          |
+------------+------------+------------+------------+------------+
```

3.6.13. SELECT TRANSFORM

The `SELECT TRANSFORM` statement allows you to start a specified child process and enter data in the required format into the child process by using standard input. Then, you can parse the standard output of the child process to obtain the output data. The `SELECT TRANSFORM` statement allows you to run scripts in other programming languages without the need to write user-defined table-valued functions (UDTFs).

Description

The performance of `SELECT TRANSFORM` and UDTFs varies based on scenarios. The comparative test results show that `SELECT TRANSFORM` outperforms UDTFs when you query small amounts of data. However, UDTFs outperform SELECT TRANSFORM when you query large amounts of data. `SELECT TRANSFORM` is easy to develop and more suitable for ad hoc queries.

`SELECT TRANSFORM` is compatible with a variety of programming languages. This statement allows you to write scripts in commands to implement simple features. Programming languages, such as AWK, Python, Perl, and Shell, support this operation. This way, you do not need to perform additional operations, such as write script files and upload resources. This simplifies the development process. To implement complex features, you can upload script files. For more information, see Example of calling Python scripts and Example of calling Java scripts.

The following table provides the comparison results of UDTFs and `SELECT TRANSFORM` in different dimensions.

Category	`SELECT TRANSFORM`	UDTF
Data type	A child process uses standard input and output to transmit data. All data is processed as strings. If you use `SELECT TRANSFORM`, a type conversion is required.	The output results and input parameters of UDTFs support multiple data types.
Data transmission	Data transmission is based on the pipeline of an operating system. However, the cache size of the pipeline is only 4 KB and cannot be changed. If the pipeline is empty or fully occupied, `SELECT TRANSFORM` cannot be performed. SELECT TRANSFORM calls underlying systems to read and write data during data transmission. This statement provides higher performance in data transmission than Java programs.	No limit is imposed on the pipeline cache.
Transmission of constant parameters	Constant parameters need to be transmitted.	Constant parameters are optionally transmitted.

Category	SELECT TRANSFORM	UDTF
Process	SELECT TRANSFORM supports two processes: parent and child processes. If the usage of computing resources is high and the data throughput is low, SELECT TRANSFORM can use the multi-core feature of a server.	A single process is used.
Performance	SELECT TRANSFORM supports the native code of tools such as AWK. This allows SELECT TRANSFORM to deliver higher performance than Java programs.	The performance is low.

Limits

PHP and Ruby are not deployed on MaxCompute compute clusters. Therefore, you cannot call PHP or Ruby scripts in MaxCompute.

Syntax

```
select transform(<arg1>, <arg2> ...)
[(row format delimited (fields terminated by <field_delimiter> (escaped by <character_escap
e>)) (lines separated by <line_separator>) (null defined as <null_value>))]
using '<unix_command_line>'
(resources '<res_name>' (',' '<res_name>')*)
[(as <col1>, <col2> ...)]
(row format delimited (fields terminated by <field_delimiter> (escaped by <character_escape
>))
(lines separated by <line_separator>)
(null defined as <null_value>)
```

- SELECT TRANSFORM keyword: required. You can replace this keyword with the keyword map or reduce that has the same semantics. To make the syntax clearer, we recommend that you use SELECT TRANSFORM.
- arg1,arg2...: required. These arguments specify the input data. The formats of these arguments are the same as those in SELECT statements. If you use the default formats, the results of expressions for each argument are implicitly converted into values of the STRING type. Then, the arguments are combined with \t and passed to the specified child process.
- ROW FORMAT clause: optional. This clause allows you to customize the format of input and output data.

  Two ROW FORMAT clauses are used in the syntax. The first clause specifies the format of input data, whereas the second clause specifies the format of output data. By default, \t is used as a column delimiter, \n is used as a row delimiter, and \N is used to represent NULL values.

> **Note**
>
> - Only one character is allowed for field_delimiter, character_escape, or line_separator. If you specify a string for these parameters, the first character in the string is used.
> - MaxCompute supports syntaxes in the formats that are specified by Apache Hive, for example, `inputRecordReader`, `outputRecordReader`, and `SerDe`. You must enable the Hive compatibility mode to use these syntaxes. To enable the Hive compatibility mode, add `set odps.sql.hive.compatible=true;` before SQL statements. For more information about the syntaxes that are supported by Apache Hive, see Hive documentation.
> - If you specify a syntax that is supported by Apache Hive, such as `inputRecordReader` or `outputRecordReader`, SQL statements may be executed at a lower speed.

- USING clause: required. This clause specifies the command that is used to start a child process.
  - In most MaxCompute SQL statements, the USING clause specifies resources. However, the USING clause in the SELECT TRANSFORM statement specifies the command to start a child process. The USING clause is used to ensure the compatibility with the syntax of Apache Hive.
  - The syntax of the USING clause is similar to the syntax of a shell script. However, instead of running a shell script, the USING clause creates a child process based on the command that you specify. Therefore, some shell features, such as input and output redirection, pipeline, and loop, cannot be used. A shell script can be used as the command to start a child process if necessary.
- RESOURCES clause: optional. This clause specifies the resources that a child process can access. You can use one of the following methods to specify the resources that a child process can access:
  - Use the RESOURCES clause to specify resources, for example, `using 'sh foo.sh bar.txt' resources 'foo.sh','bar.txt'`.
  - Use set odps.sql.session.resources to specify resources. For example, you can add `set odps.sql.session.resources=foo.sh,bar.txt;` before SQL statements to specify resources.

    After this global configuration applies, all SELECT TRANSFORM statements can access the resources. Separate multiple resource files with commas (,).
- AS clause: optional. This clause specifies the output columns and their data types, for example, `as(col1:bigint, col2:boolean)`.
  - If you do not specify the data types for output columns, the default data type STRING is used. For example, `AS(col1, col2)` indicates that the output columns are of the STRING type.
  - The output data is obtained by parsing the standard output of the child process. If the specified data is not of the STRING type, MaxCompute implicitly calls the `CAST` function to convert the data type to STRING. Runtime exceptions may occur during the conversion.
  - You cannot specify data types for only some of the specified columns, for example, `as(col1, col2:bigint)`.
  - If you omit the `AS` clause, the field before the first `\t` in the standard output data is the key and all the following parts are the value. This is equivalent to `AS(key, value)`.

Example of calling shell commands

Run a shell command to generate 50 rows of data starting from 1 to 50. The output is the `data` field. Use the output of the shell command as the input of `SELECT TRANSFORM`. Sample statement:

```
select transform(script) using 'sh' as (data)
from (
        select  'for i in `seq 1 50`; do echo $i; done' as script
      ) t
;
-- The preceding statement is equivalent to the following statement:
select transform('for i in `seq 1 50`; do echo $i; done') using 'sh' as (data);
```

The following result is returned:

```
+------------+
| data       |
+------------+
| 1          |
| 2          |
| 3          |
| 4          |
| 5          |
| 6          |
| 7          |
| 8          |
| 9          |
| 10         |
| 11         |
| 12         |
| 13         |
| 14         |
| 15         |
| 16         |
| 17         |
| 18         |
| 19         |
| 20         |
| 21         |
| 22         |
| 23         |
| 24         |
| 25         |
| 26         |
| 27         |
| 28         |
| 29         |
| 30         |
| 31         |
| 32         |
| 33         |
| 34         |
| 35         |
| 36         |
| 37         |
| 38         |
| 39         |
| 40         |
| 41         |
```

```
| 42          |
| 43          |
| 44          |
| 45          |
| 46          |
| 47          |
| 48          |
| 49          |
| 50          |
+-------------+
```

Example of calling Python commands

Use a Python command to generate 50 rows of data starting from 1 to 50. The output is the `data` field. Use the output of the Python command as the input of `SELECT TRANSFORM`. Sample statement:

```
select transform(script) using 'python' as (data)
from (
        select  'for i in xrange(1, 51):  print i;' as script
      ) t
;
-- The preceding statement is equivalent to the following statement:
select transform('for i in xrange(1, 51):  print i;') using 'python' as (data);
```

The following result is returned:

```
+-------------+
| data        |
+-------------+
| 1           |
| 2           |
| 3           |
| 4           |
| 5           |
| 6           |
| 7           |
| 8           |
| 9           |
| 10          |
| 11          |
| 12          |
| 13          |
| 14          |
| 15          |
| 16          |
| 17          |
| 18          |
| 19          |
| 20          |
| 21          |
| 22          |
| 23          |
| 24          |
| 25          |
```

```
| 25          |
| 26          |
| 27          |
| 28          |
| 29          |
| 30          |
| 31          |
| 32          |
| 33          |
| 34          |
| 35          |
| 36          |
| 37          |
| 38          |
| 39          |
| 40          |
| 41          |
| 42          |
| 43          |
| 44          |
| 45          |
| 46          |
| 47          |
| 48          |
| 49          |
| 50          |
+-------------+
```

Example of calling AWK commands

Create a test table. Run an AWK command to provide the second column of the test table as the output. The output data is the `data` field. Use the output of the AWK command as the input of `SELECT TRANSFORM`. Sample statements:

```
-- Create a test table.
create table testdata(c1 bigint,c2 bigint);
-- Insert test data into the test table.
insert into table testdata values (1,4),(2,5),(3,6);
-- Execute the SELECT TRANSFORM statement.
select transform(*) using "awk '//{print $2}'" as (data) from testdata;
```

The following result is returned:

```
+-------------+
| data        |
+-------------+
| 4           |
| 5           |
| 6           |
+-------------+
```

Example of calling Perl commands

Create a test table. Run a Perl command to provide data of the test table as the output. The output data is the `data` field. Use the output of the Perl command as the input of `SELECT TRANSFORM`. Sample statements:

```
-- Create a test table.
create table testdata(c1 bigint,c2 bigint);
-- Insert test data into the test table.
insert into table testdata values (1,4),(2,5),(3,6);
-- Execute the SELECT TRANSFORM statement.
select transform(testdata.c1, testdata.c2) using "perl -e 'while($input = <STDIN>){print $i
nput;}'" from testdata;
```

The following result is returned:

```
+------------+------------+
| key        | value      |
+------------+------------+
| 1          | 4          |
| 2          | 5          |
| 3          | 6          |
+------------+------------+
```

Example of calling Python scripts

1. Create the *myplus.py* file. Sample statements:

   ```
   #!/usr/bin/env python
   import sys
   line = sys.stdin.readline()
   while line:
       token = line.split('\t')
       if (token[0] == '\\N') or (token[1] == '\\N'):
           print '\\N'
       else:
           print str(token[0]) +'\t' + str(token[1])
       line = sys.stdin.readline()
   ```

2. Add the Python script file as a resource to MaxCompute.

   ```
   add py ./myplus.py -f;
   ```

   > **Note** You can also use the DataWorks console to add the Python file as a resource. For more information, see Create a MaxCompute resource.

3. Execute the `SELECT TRANSFORM` statement to call this file.

```
-- Create a test table.
create table testdata(c1 bigint,c2 bigint);
-- Insert test data into the test table.
insert into table testdata values (1,4),(2,5),(3,6);
-- Execute the SELECT TRANSFORM statement.
select
transform (testdata.c1, testdata.c2)
using 'python myplus.py'resources 'myplus.py'
as (result1,result2)
from testdata;
-- The preceding statement is equivalent to the following statements:
set odps.sql.session.resources=myplus.py;
select transform (testdata.c1, testdata.c2)
using 'python myplus.py'
as (result1,result2)
from testdata;
```

The following result is returned:

```
+------------+------------+
| result1    | result2    |
+------------+------------+
| 1          | 4          |
|            | NULL       |
| 2          | 5          |
|            | NULL       |
| 3          | 6          |
|            | NULL       |
+------------+------------+
```

Example of calling Java scripts

1. Write a Java script and export it as the *Sum.jar* file. Sample Java code:

```
package com.aliyun.odps.test;
import java.util.Scanner;
public class Sum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            String s = sc.nextLine();
            String[] tokens = s.split("\t");
            if (tokens.length < 2) {
                throw new RuntimeException("illegal input");
            }
            if (tokens[0].equals("\\N") || tokens[1].equals("\\N")) {
                System.out.println("\\N");
            }
            System.out.println(Long.parseLong(tokens[0]) + Long.parseLong(tokens[1]));
        }
    }
}
```

2. Add the file as a resource to MaxCompute.

```
add jar ./Sum.jar -f;
```

3. Execute the `SELECT TRANSFORM` statement to call the file.

```
-- Create a test table.
create table testdata(c1 bigint,c2 bigint);
-- Insert test data into the test table.
insert into table testdata values (1,4),(2,5),(3,6);
-- Execute the SELECT TRANSFORM statement.
select transform(testdata.c1, testdata.c2) using 'java -cp Sum.jar com.aliyun.odps.test
.Sum' resources 'Sum.jar' from testdata;
-- The preceding statement is equivalent to the following statements:
set odps.sql.session.resources=Sum.jar;
select transform(testdata.c1, testdata.c2) using 'java -cp Sum.jar com.aliyun.odps.test
.Sum' from testdata;
```

The following result is returned:

```
+-----+
| cnt |
+-----+
| 5   |
| 7   |
| 9   |
+-----+
```

Note Although Java and Python have a ready-made UDTF framework, `SELECT TRANSFORM` makes it easier for you to write scripts. SELECT TRANSFORM does not require additional dependencies and has no format requirements, and even can directly use offline scripts. The directory for saving offline Java scripts can be obtained from the `JAVA_HOME` environment variable. The directory for saving offline Python scripts can be obtained from the `PYTHON_HOME` environment variable.

Example of calling scripts in series

You can execute the `SELECT TRANSFORM` statements in series. To perform this operation, you can use the `DISTRIBUTE BY` and `SORT BY` clauses to preprocess input data. Sample statements:

```
select transform(key, value) using 'cmd2' from
(
    select transform(*) using 'cmd1' from
    (
        select * from testdata distribute by c2 sort by c1
    ) t distribute by key sort by value
) t2;
```

You can also use the `map` and `reduce` keywords to execute the SELECT TRANSFORM statements in series.

```
@a := select * from data distribute by col2 sort by col1;
@b := map * using 'cmd1' distribute by col1 sort by col2 from @a;
reduce * using 'cmd2' from @b;
```

3.7. SQL enhancement operations

3.7.1. LOAD

If you want to import data from an external data store into a MaxCompute table or a partition of the table, you can use LOAD statements. This topic describes how to use LOAD statements to import data in the CSV format or another open source format from an external data store into a MaxCompute table.

Before you execute `LOAD` statements, make sure that you have the CREATE TABLE and ALTER permissions on MaxCompute projects. For more information about authorization, see Permissions.

You can execute the statements that are described in this topic on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Description

MaxCompute allows you to execute the `LOAD OVERWRITE` or `LOAD INTO` statement to import data from an external data store into a MaxCompute table or a partition of the table. The data that you want to import must be in the CSV format or another open source format. The external data store can be Object Storage Service (OSS), Amazon Redshift, or BigQuery. If you want to import data from Amazon Redshift or BigQuery into a MaxCompute table, you must first import the data into OSS.

MaxCompute also allows you to import data into a partition of a partitioned table in dynamic partition mode.

The `LOAD INTO` statement directly appends data to a table or a partition of the table. The `LOAD OVERWRITE` statement clears a table or a partition of the table and then inserts data into the table or partition.

Prerequisites

MaxCompute must be granted the related permissions before you import data into MaxCompute. The authorization methods for the `LOAD OVERWRITE` and `LOAD INTO` statements are the same as those for MaxCompute external tables. You can use one-click authorization to ensure security. For more information, see STS authorization.

After the authorization is complete, you need to select an appropriate import method based on the format of the data that you want to import.

- Use a built-in extractor to import data
- Import data in another open source format

Limits

- Data in an external data store can be imported into only the MaxCompute projects that are in the

same region as the external data store.

- If you want to import data into a partitioned table, make sure that the schema of the table in the external data store does not include partition key columns.

Use a built-in extractor to import data

- Syntax

```
{load overwrite|into} table <table_name> [partition (<pt_spec>)]
from location <external_location>
stored by <StorageHandler>
[with serdeproperties (<Options>)];
```

- Parameters
  - table_name: required. The name of the table into which you want to insert data. You must create the table before you insert data into it. The schema of the table, excluding partition key columns, must be consistent with the layout of the data in the external data store.
  - pt_spec: optional. The partition information of the table into which you want to insert data. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
  - external_location: required. The OSS directory that stores the data. The value of this parameter is in the `'oss://<oss_endpoint>/<object>'` format. For more information about OSS endpoints, see OSS domain names. By default, MaxCompute reads all files in this directory.
  - StorageHandler: required. The name of a storage handler that is considered a built-in extractor. `com.aliyun.odps.CsvStorageHandler` is a storage handler that is used to process CSV files. It defines how to read data from and write data to CSV files. You need to specify only this parameter based on your business requirements. The related logic is implemented by the system. You can use this parameter in the same way as you use it for a MaxCompute external table. For more information, see Create an OSS external table.
  - Options: optional. The properties related to the external table in WITH SERDEPROPERTIES. The properties are the same as those for creating the MaxCompute external table. For more information about the properties, see Create an OSS external table.

- Example

  The owners of MaxCompute and OSS resources use the same Alibaba Cloud account. Use a built-in extractor to import data from the vehicle.csv file into a MaxCompute table over a virtual private cloud (VPC).

  i. Perform one-click authorization.

  ii. Save the vehicle.csv file to the `mc-test/data_location/` directory of an OSS bucket in the `oss-cn-hangzhou` region and organize the path to the OSS directory. For more information about how to create an OSS bucket, see Create buckets.

  The vehicle.csv file contains the following data:

```
1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S
1,6,9,1,46.81006,-92.08174,9/14/2014 0:00,S
1,7,53,1,46.81006,-92.08174,9/14/2014 0:00,N
1,8,63,1,46.81006,-92.08174,9/14/2014 0:00,SW
1,9,4,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,10,31,1,46.81006,-92.08174,9/14/2014 0:00,N
```

Organize the following path to the OSS directory based on the bucket, region, and endpoint:

```
oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/
```

iii. Log on to the MaxCompute client and create a table named ambulance_data_csv_load. Sample statement:

```
create table ambulance_data_csv_load (
vehicleId INT,
recordId INT,
patientId INT,
calls INT,
locationLatitute DOUBLE,
locationLongtitue DOUBLE,
recordTime STRING,
direction STRING );
```

iv. Execute the `LOAD OVERWRITE` statement to import the vehicle.csv file from OSS into the destination table. Sample statement:

```
load overwrite table ambulance_data_csv_load
from
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/'
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole',   -- The Aliba
ba Cloud Resource Name (ARN) of the AliyunODPSDefaultRole role. You can view the ARN
on the RAM Roles page.
'odps.text.option.delimiter'=','
);
```

v. View the import result of the ambulance_data_csv_load table. Sample statements:

```
-- Enable a full table scan. This operation takes effect only for the current session
.
set odps.sql.allow.fullscan=true;
select * from ambulance_data_csv_load;
```

The following result is returned:

```
+------------+------------+------------+------------+-----------------+-------------
------+------------+------------+
| vehicleid  | recordid   | patientid  | calls      | locationlatitute | locationlong
titue | recordtime | direction  |
+------------+------------+------------+------------+-----------------+-------------
------+------------+------------+
| 1          | 1          | 51         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | S          |
| 1          | 2          | 13         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | NE         |
| 1          | 3          | 48         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | NE         |
| 1          | 4          | 30         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | W          |
| 1          | 5          | 47         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | S          |
| 1          | 6          | 9          | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | S          |
| 1          | 7          | 53         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | N          |
| 1          | 8          | 63         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | SW         |
| 1          | 9          | 4          | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | NE         |
| 1          | 10         | 31         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | N          |
+------------+------------+------------+------------+-----------------+-------------
------+------------+------------+
```

Import data in another open source format

- Syntax

```
{load overwrite|into} table <table_name> [partition (<pt_spec>)]
from location <external_location>
[row format serde '<serde_class>'
  [with serdeproperties (<Options>)]
]
stored as <file_format>;
```

- Parameters
  - table_name: required. The name of the table into which you want to insert data. You must create the table before you insert data into it. The schema of the table, excluding partition key columns, must be consistent with the layout of the data in the external data store.
  - pt_spec: optional. The partition information of the table into which you want to insert data. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
  - external_location: required. The OSS directory that stores the data. The value of this parameter is in the `'oss://<oss_endpoint>/<object>'` format. For more information about OSS endpoints, see OSS domain names. By default, MaxCompute reads all files in this directory.
  - serde_class: optional. You can use this parameter in the same way as you use it for a MaxCompute external table. For more information, see Create an OSS external table.

- Options: required. The properties related to the external table in WITH SERDEPROPERTIES. The properties are the same as those for creating the MaxCompute external table. For more information about the properties, see Create an OSS external table.
- file_format: required. The format of the data that you want to import, such as ORC, PARQUET, RCFILE, SEQUENCEFILE, or TEXTFILE. You can use this parameter in the same way as you use it for a MaxCompute external table. For more information, see Create an OSS external table.

> **Note**
> - If you want to use the default values of `serde_class` and `Options`, you do not need to specify these parameters.
> - The size of a file that you want to import cannot exceed 3 GB. If the size exceeds 3 GB, split the file.

- Examples
  - Example 1: Import data in another open source format. The owners of MaxCompute and OSS use the same Alibaba Cloud account. Import data from a text file named vehicle into a MaxCompute table over a VPC.
    a. Perform one-click authorization.
    b. Save the vehicle text file to the `mc-test/data_location/` directory of an OSS bucket in the `oss-cn-hangzhou` region and organize the path to OSS directory. For more information about how to create an OSS bucket, see Create buckets.

    The vehicle text file contains the following data:

```
1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S
1,6,9,1,46.81006,-92.08174,9/14/2014 0:00,S
1,7,53,1,46.81006,-92.08174,9/14/2014 0:00,N
1,8,63,1,46.81006,-92.08174,9/14/2014 0:00,SW
1,9,4,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,10,31,1,46.81006,-92.08174,9/14/2014 0:00,N
```

    Organize the following path to the OSS directory based on the bucket, region, and endpoint:

```
oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/
```

c. Log on to the MaxCompute client and create a destination table named ambulance_data_textfile_load_pt. Sample statement:

```
create table ambulance_data_textfile_load_pt (
vehicleId STRING,
recordId STRING,
patientId STRING,
calls STRING,
locationLatitute STRING,
locationLongtitue STRING,
recordTime STRING,
direction STRING)
partitioned by (ds STRING);
```

d. Execute the `LOAD OVERWRITE` statement to import the vehicle text file from OSS to the destination table. Sample statement:

```
load overwrite table ambulance_data_textfile_load_pt partition(ds='20200910')
from
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/'
row format serde 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
stored as textfile;
```

e. View the import result of the ambulance_data_textfile_load_pt table. Sample statements:

```
-- Enable a full table scan. This operation takes effect only for the current sessi
on.
set odps.sql.allow.fullscan=true;
select * from ambulance_data_textfile_load_pt;
```

The following result is returned:

```
+------------+------------+------------+------------+-----------------+-----------
--------+------------+------------+------------+
| vehicleid  | recordid   | patientid  | calls      | locationlatitute | locationlo
ngtitue | recordtime | direction  | ds         |
+------------+------------+------------+------------+-----------------+-----------
--------+------------+------------+------------+
| 1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,6,9,1,46.81006,-92.08174,9/14/2014 0:00,S | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,7,53,1,46.81006,-92.08174,9/14/2014 0:00,N | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,8,63,1,46.81006,-92.08174,9/14/2014 0:00,SW | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,9,4,1,46.81006,-92.08174,9/14/2014 0:00,NE | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
| 1,10,31,1,46.81006,-92.08174,9/14/2014 0:00,N | NULL       | NULL       | NULL
| NULL             | NULL               | NULL       | NULL       | 20200910   |
+------------+------------+------------+------------+-----------------+-----------
--------+------------+------------+------------+
```

- Example 2: Import data into a destination table in dynamic partition mode.

  > **Note** If the subdirectories under an OSS directory are mapped to partition names, you can import data into a partitioned table in dynamic partition mode.

  a. Perform one-click authorization.

b. Save the vehicle1.csv file to the `mc-test/data_location/ds=20200909/` directory of the OSS bucket and the vehicle2.csv file to the `mc-test/data_location/ds=20200910/` directory of the OSS bucket in the `oss-cn-hangzhou` region and organize the path to the OSS directory. For more information about how to create an OSS bucket, see Create buckets.

The vehicle1.csv and vehicle2.csv files contain the following data:

```
--vehicle1.csv
1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S
1,6,9,1,46.81006,-92.08174,9/14/2014 0:00,S
--vehicle2.csv
1,7,53,1,46.81006,-92.08174,9/14/2014 0:00,N
1,8,63,1,46.81006,-92.08174,9/14/2014 0:00,SW
1,9,4,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,10,31,1,46.81006,-92.08174,9/14/2014 0:00,N
```

Organize the following paths to the OSS directories based on the bucket, region, and endpoint:

```
oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/ds=20200909/'
oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/ds=20200910/'
```

c. Log on to the MaxCompute client and create a destination table named ambulance_data_csv_load_dynpt. Sample statement:

```
create table ambulance_data_csv_load_dynpt (
vehicleId STRING,
recordId STRING,
patientId STRING,
calls STRING,
locationLatitute STRING,
locationLongtitue STRING,
recordTime STRING,
direction STRING)
partitioned by (ds STRING);
```

d. Execute the `LOAD OVERWRITE` statement to import the files from OSS into the destination table. Sample statement:

```
load overwrite table ambulance_data_csv_load_dynpt partition(ds)
from
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-test/data_location/'
row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
stored as textfile;
```

e. -- View the import result of the ambulance_data_csv_load_dynpt table. Sample statements:

```
-- Enable a full table scan. This operation takes effect only for the current sessi
on.
set odps.sql.allow.fullscan=true;
select * from ambulance_data_csv_load_dynpt;
```

The following result is returned:

```
+------------+------------+------------+------------+------------------+-----------
--------+------------+------------+------------+
| vehicleid  | recordid   | patientid  | calls      | locationlatitute | locationlo
ngtitue | recordtime | direction  | ds         |
+------------+------------+------------+------------+------------------+-----------
--------+------------+------------+------------+
| 1          | 7          | 53         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | N          | 20200909   |
| 1          | 8          | 63         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | SW         | 20200909   |
| 1          | 9          | 4          | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | NE         | 20200909   |
| 1          | 10         | 31         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | N          | 20200909   |
| 1          | 1          | 51         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | S          | 20200910   |
| 1          | 2          | 13         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | NE         | 20200910   |
| 1          | 3          | 48         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | NE         | 20200910   |
| 1          | 4          | 30         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | W          | 20200910   |
| 1          | 5          | 47         | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | S          | 20200910   |
| 1          | 6          | 9          | 1          | 46.81006         | -92.08174
| 9/14/2014 0:00 | S          | 20200910   |
+------------+------------+------------+------------+------------------+-----------
--------+------------+------------+------------+
```

3.7.2. UNLOAD

MaxCompute allows you to export data from MaxCompute projects to Object Storage Service (OSS). This provides an easy way to store structured data in OSS. This also allows other computing engines to use the data that is exported from MaxCompute to OSS. This topic describes how to use UNLOAD statements to export data in the CSV format or another open source format from MaxCompute to OSS.

You can execute the statements that are described in this topic on the following platforms:

- MaxCompute client
- Query editor of the MaxCompute console
- DataWorks console
- MaxCompute Studio

Prerequisites

- OSS is activated.

  For more information about how to activate OSS, see Activate OSS.

- You have the SELECT permission on the table that you want to export from a MaxCompute project.

  For more information about authorization, see Permissions.

Limits

The use of UNLOAD statements has the following limits:

- MaxCompute automatically splits the file that is exported to OSS into multiple parts and generates a name for the file. You cannot customize the name or file name extension for the exported file.
- File name extensions cannot be added to the exported files in an open source format.
- If you repeatedly export data, the previously exported file is not overwritten. Instead, a new file is generated.

Usage notes

- You are not charged for UNLOAD statements. The subquery clauses in the UNLOAD statements need to scan data and use computing resources to calculate the results. Therefore, the subquery clauses are charged as common SQL jobs.
- In some scenarios, you can store structured data in OSS to reduce storage costs. However, you must estimate the costs in advance.

  The MaxCompute storage fee is USD 0.018 per GB per month. For more information about storage fees, see Storage pricing (pay-as-you-go). The data compression ratio is about 5:1 for the data that is imported into MaxCompute. You are charged based on the size of data after compression.

  If you use the Standard storage class of OSS to store your data, the unit price is USD 0.018 per GB per month. For more information about the fees for the Infrequent Access (IA), Archive, and Cold Archive storage classes, see Storage fees.

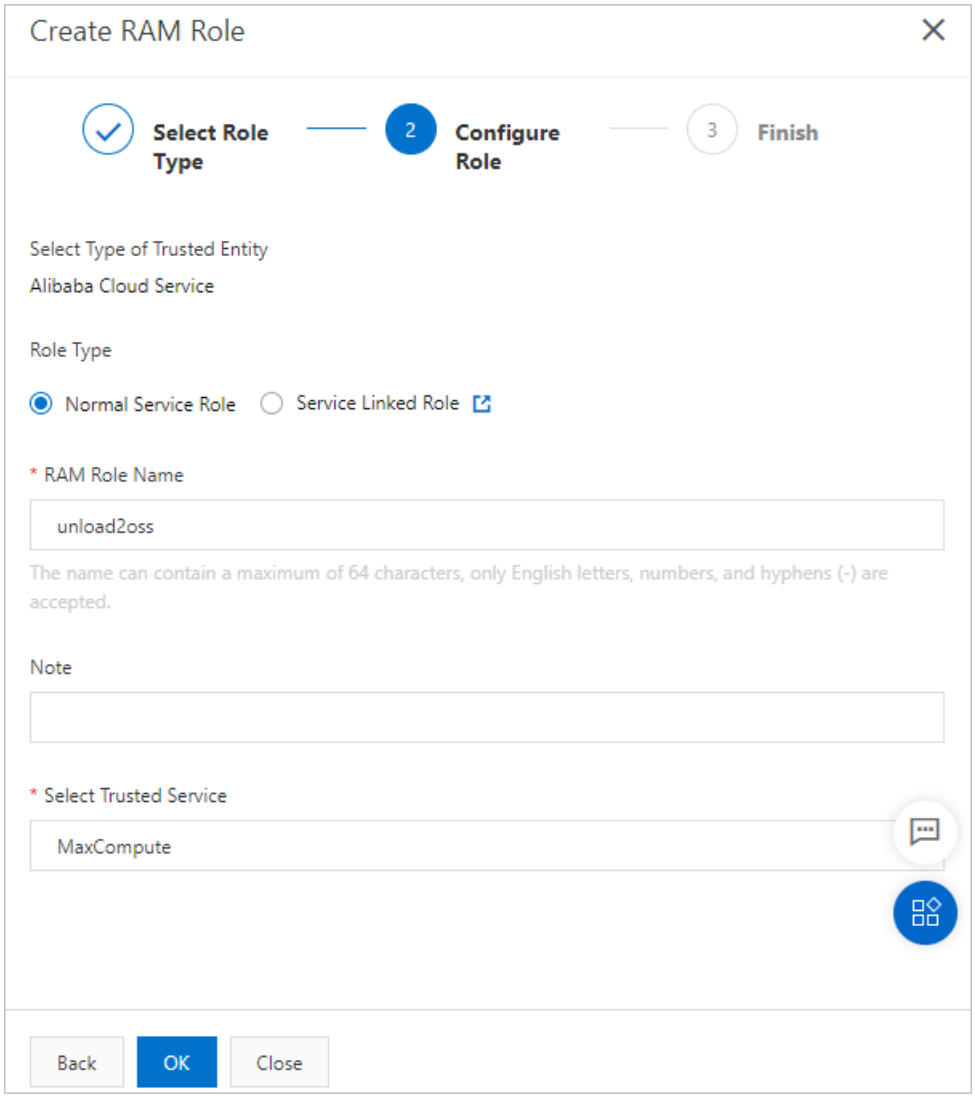
  If you want to export data to reduce storage costs, we recommend that you: (1) Evaluate the data compression ratio based on the data feature test. (2) Estimate the costs of using UNLOAD statements based on the query statement used when you export data. (3) Evaluate the method for accessing the exported data to avoid extra costs caused by unnecessary data migration.

Authorization for access to OSS

If you want to export data to OSS, you must grant MaxCompute the permissions to access OSS. The authorization methods for UNLOAD statements are the same as those for MaxCompute external tables. You need to configure the properties specified by `WITH SERDEPROPERTIES` in UNLOAD statements and then use RAM roles to complete OSS authorization. Authorization procedure:

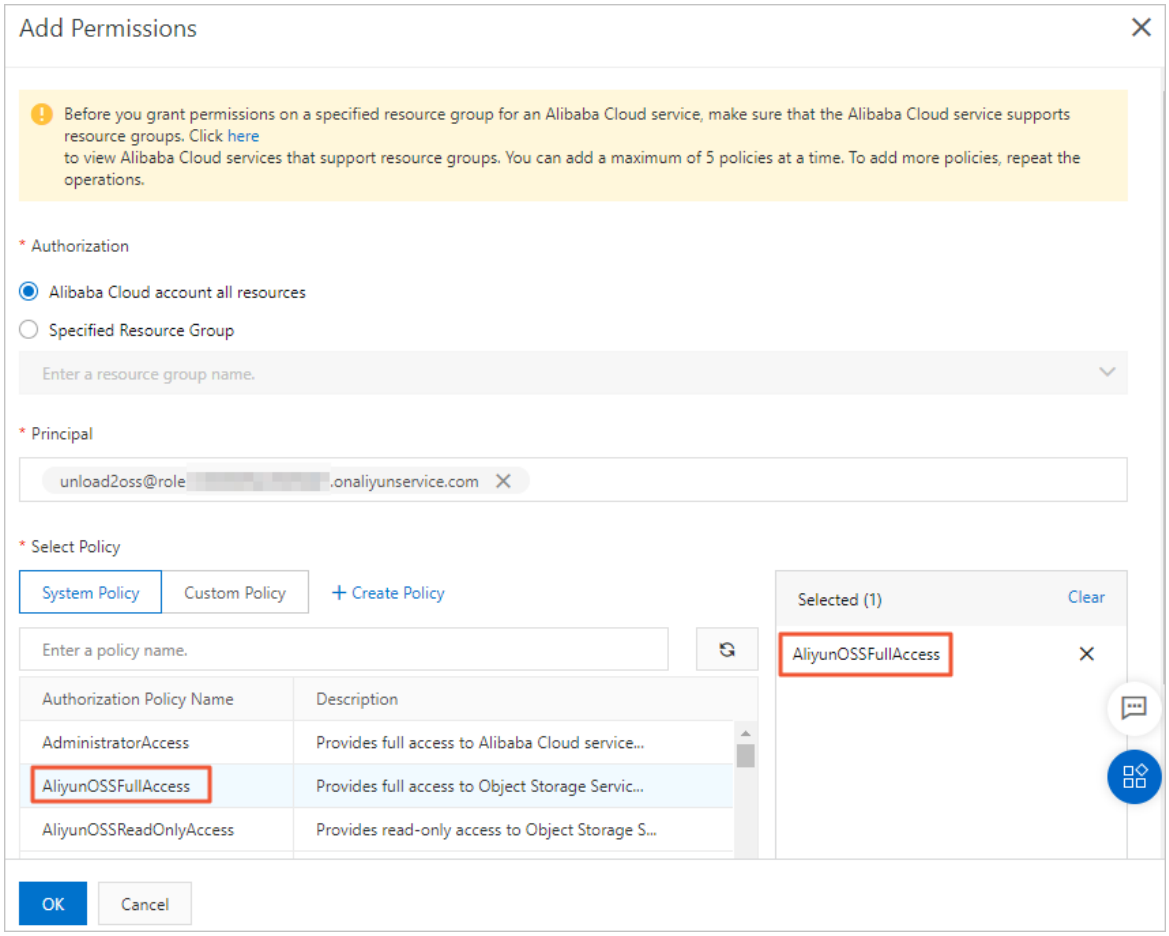
1. Log on to the RAM console. In the left-side navigation pane, choose RAM Roles. On the page that appears, click Create RAM Role. In the Create RAM Role panel, specify **Alibaba Cloud Service** for Trusted entity type in the Select Role Type step. In the Configure Role step, specify **Normal Service Role** for **Role Type**, specify RAM Role Name, such as unload2oss, and then select **MaxCompute** from the Select Trusted Service drop-down list.

   For more information, see Create a RAM role for a trusted Alibaba Cloud service.



2. After the role is created, attach the system policy AliyunOSSFullAccess to the role.

   For more information, see Grant permissions to a RAM role.



After the authorization is complete, you must select one of the following export methods based on the format of the data that you want to export:

- Use a built-in extractor to export data
- Export data in another open source format

Use a built-in extractor to export data

- Syntax

```
unload from {<select_statement>|<table_name> [partition (<pt_spec>)]}
into
location <external_location>
[stored by <StorageHandler>]
[with serdeproperties ('<property_name>'='<property_value>',...)];
```

- Parameters
  - select_statement: a `SELECT` clause. This clause is used to query the data that needs to be inserted into a table in the destination OSS directory from the source table. The source table can be a partitioned table or a non-partitioned table. For more information about `SELECT` clauses, see SELECT syntax.

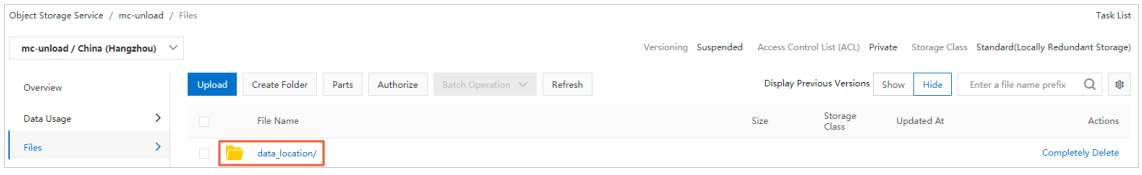
- table_name and pt_spec: You can use the table name or the combination of the table and partition names to specify the data that you want to export. This export method does not automatically generate query statements. Therefore, no fees are incurred. The value of pt_spec is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
- external_location: required. The destination OSS directory to which you want to export data. The value of this parameter is in the `'oss://<oss_endpoint>/<object>'` format . For more information about OSS directories, see OSS domain names.
- StorageHandler: required. The name of the storage handler that is considered a built-in extractor. Set this parameter to `com.aliyun.odps.CsvStorageHandler` or `com.aliyun.odps.TsvStorageHandler`. The storage handler is used to process CSV and TSV files and defines how to read data from or write data to these files. You need to specify only this parameter based on your business requirements. The related logic is implemented by the system. If you use a built-in extractor to export data, the file name extension `.csv` or `.tsv` is automatically added to the files that are exported. You can use this parameter in the same way as you use it for MaxCompute external tables. For more information, see Create an OSS external table.
- <property_name>'='<property_value>': optional. property_name specifies the name of a property and property_value specifies the value of a property. You can use this clause in the same way as you use it for MaxCompute external tables. For more information about the properties, see Create an OSS external table.

- Examples

  This section demonstrates how to export data of the sale_detail table from a MaxCompute project to OSS. The sale_detail table contains the following data:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

  i. Log on to the OSS console and create the directory `mc-unload/data_location/` in the OSS bucket in the `oss-cn-hangzhou` region and organize the OSS directory. For more information about how to create an OSS bucket, see Create buckets.



  The following OSS directory is organized based on the bucket, region, and endpoint.

```
oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location
```

  ii. Log on to the MaxCompute client and execute the UNLOAD statement to export data of the sale_detail table to OSS. The following examples are provided:

- Example 1: Export the data of the sale_detail table as a CSV file and package the file into a GZIP file. Sample statements:

```
-- Control the number of exported files: Set the size of data of the MaxCompute tab
le read by a single worker. Unit: megabytes. The MaxCompute table is compressed bef
ore you export it. The size of the exported data is about four times the data size
before the export.
set odps.stage.mapper.split.size=256;
-- Export data.
unload from
(select * from sale_detail)
into
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location'
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties ('odps.properties.rolearn'='acs:ram::139699392458****:role/unl
oad2oss', 'odps.text.option.gzip.output.enabled'='true');
-- The preceding statements are equivalent to the following statements:
set odps.stage.mapper.split.size=256;
unload from sale_detail
into
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location'
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties ('odps.properties.rolearn'='acs:ram::139699392458****:role/unl
oad2oss', 'odps.text.option.gzip.output.enabled'='true');
```

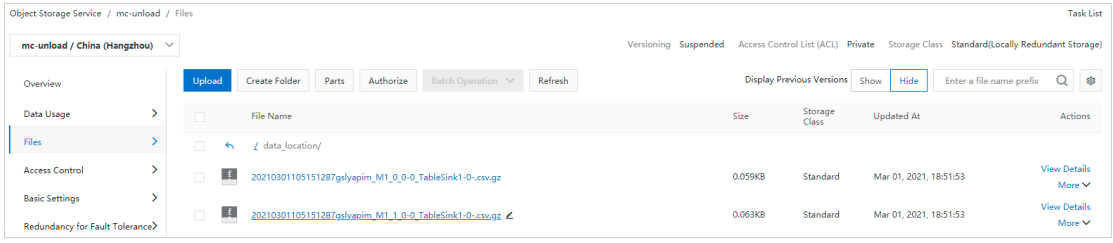
- Example 2: Export data from the partition (sale_date='2013', region='china') in the sale_detail table as a TSV file to OSS and package the file into a GZIP file.

```
-- Control the number of exported files: Set the size of data of the MaxCompute tab
le read by a single worker. Unit: megabytes. The MaxCompute table is compressed bef
ore you export it. The size of the exported data is about four times the data size
before the export.
set odps.stage.mapper.split.size=256;
-- Export data.
unload from sale_detail partition (sale_date='2013',region='china')
into
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location'
stored by 'com.aliyun.odps.TsvStorageHandler'
with serdeproperties ('odps.properties.rolearn'='acs:ram::139699392458****:role/unl
oad2oss', 'odps.text.option.gzip.output.enabled'='true');
```

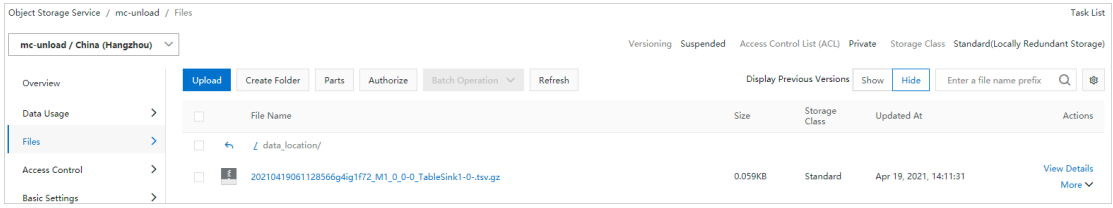
`'odps.text.option.gzip.output.enabled'='true'` specifies that the exported file is compressed in a GZIP file. Only the GZIP format is supported.

iii. Log on to the OSS console to view the import result in the destination OSS directory.

- Import result of Example 1



- Import result of Example 2



Export data in another open source format

- Syntax

```
unload from {<select_statement>|<table_name> [partition (<pt_spec>)]}
into
location <external_location>
[row format serde '<serde_class>'
  [with serdeproperties ('<property_name>'='<property_value>',...)]
]
storeds as <file_format>
[properties('<tbproperty_name>'='<tbproperty_value>')];
```

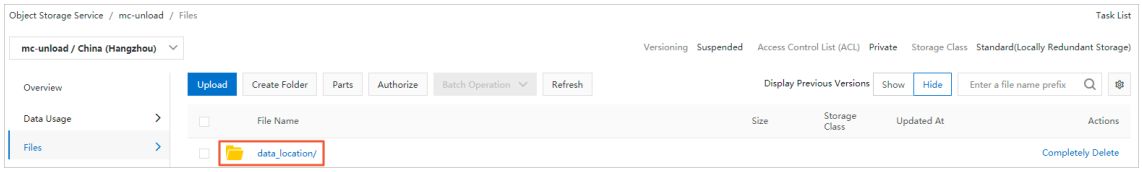
- Parameters
  - select_statement: a `SELECT` clause. This clause is used to query the data that needs to be inserted into a table in the destination OSS directory from the source table. The source table can be a partitioned table or a non-partitioned table. For more information about `SELECT` clauses, see SELECT syntax.
  - table_name and pt_spec: You can use the table name or the combination of the table and partition names to specify the data that you want to export. This export method does not automatically generate query statements. Therefore, no fees are incurred. The value of pt_spec is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format.
  - external_location: required. The destination OSS directory to which you want to export data. The value of this parameter is in the `'oss://<oss_endpoint>/<object>'` format . For more information about OSS directories, see OSS domain names.
  - serde_class: optional. You can use this clause in the same way as you use it for MaxCompute external tables. For more information, see Open source data formats supported by OSS external tables.
  - '<property_name>'='<property_value>': optional. property_name specifies the name of a property and property_value specifies the value of a property. The supported properties are the same as those supported by MaxCompute external tables. For more information about the properties, see Open source data formats supported by OSS external tables.
  - file_format: required. The format of the exported file, such as ORC, PARQUET, RCFILE, SEQUENCEFILE, or TEXTFILE. You can use this parameter in the same way as you use it for MaxCompute external tables. For more information, see Open source data formats supported by OSS external tables.
  - '<tbproperty_name>'='<tbproperty_value>': optional. tbproperty_name specifies the name of a property in the extended information of the external table. tbproperty_value specifies the value of a property in the extended information of the external table. For example, if you want to export data in an open source format as a file compressed by using Snappy or LZO, you can set the mcfed.parquet.compression property to `SNAPPY` or `LZO` .

- Examples

  This section demonstrates how to export data of the sale_detail table from a MaxCompute project to OSS. The sale_detail table contains the following data:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

  i. Log on to the OSS console and create the directory `mc-unload/data_location/` in the OSS bucket in the `oss-cn-hangzhou` region and organize the OSS directory. For more information about how to create an OSS bucket, see Create buckets.



  The following OSS directory is organized based on the bucket, region, and endpoint.

```
oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location
```

  ii. Log on to the MaxCompute client and execute the UNLOAD statement to export data of the sale_detail table to OSS. The following examples are provided:

  - Example 1: Export data of the sale_detail table as a file in the PARQUET format and compressed by using Snappy. Sample statements:

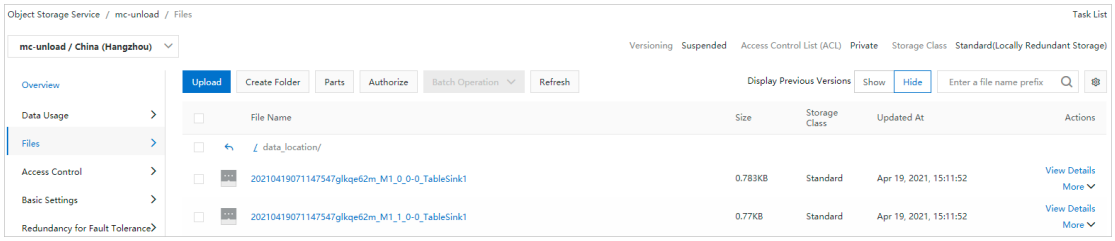
```
-- Control the number of exported files: Set the size of data of the MaxCompute tab
le read by a single worker. Unit: megabytes. The MaxCompute table is compressed bef
ore you export it. The size of the exported data is about four times the data size
before the export.
set odps.stage.mapper.split.size=256;
-- Export data.
unload from
(select * from sale_detail)
into
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location'
row format serde 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
with serdeproperties ('odps.properties.rolearn'='acs:ram::139699392458****:role/unl
oad2oss')
stored as parquet
properties('mcfed.parquet.compression'='SNAPPY');
```

- Example 2: Export data from the partition (sale_date='2013', region='china') in the sale_detail table as a PARQUET file to OSS and compress the file by using Snappy. Sample statements:

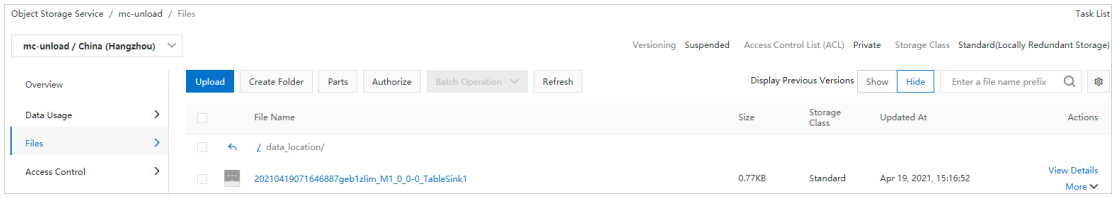
```
-- Control the number of exported files: Set the size of data of the MaxCompute tab
le read by a single worker. Unit: megabytes. The MaxCompute table is compressed bef
ore you export it. The size of the exported data is about four times the data size
before the export.
set odps.stage.mapper.split.size=256;
-- Export data.
unload from sale_detail partition (sale_date='2013',region='china')
into
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/mc-unload/data_location'
row format serde 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
with serdeproperties ('odps.properties.rolearn'='acs:ram::139699392458****:role/unl
oad2oss')
stored as parquet
properties('mcfed.parquet.compression'='SNAPPY');
```

iii. Log on to the OSS console to view the import result in the destination OSS directory.

- Import result of Example 1



- Import result of Example 2



> **Note** If the exported data is compressed by using Snappy or LZO, the file name extension .snappy or .lzo of the exported file cannot be displayed.

3.7.3. EXPLAIN

In most cases, you need to analyze query statements or table schemas to find performance bottlenecks during development. MaxCompute SQL provides the `EXPLAIN` statement to help you analyze query statements. This topic describes the features and syntax of the `EXPLAIN` statement. This topic also provides examples on using the EXPLAIN statement.

Description

The `EXPLAIN` statement can display the execution plan structure of a DML statement in MaxCompute SQL. This statement helps you understand how an SQL statement is executed and provides guidance for optimizing SQL statements. One query statement corresponds to multiple jobs, and one job corresponds to multiple tasks.

> **Note** If a query statement is complex, the number of rows in the output results of the `EXPLAIN` statement may exceed the threshold specified by the API of the upper-layer application. As a result, the output results cannot be completely displayed. To address this issue, you can split the query statement into multiple subqueries and execute the `EXPLAIN` statement on each subquery to obtain the structure of the job.

Syntax

```
explain <dml query>;
```

dml query: required. The `SELECT` statement. For more information, see SELECT syntax.

Output

The output results of the `EXPLAIN` statement includes the following information:

- Dependencies between jobs

  For example, `job0 is a root job`. If a query requires only `job0`, only one row of data is displayed.

- Dependencies between tasks

  ```
  In Job job0:
  root Tasks: M1_Stg1, M2_Stg1
  J3_1_2_Stg1 depends on: M1_Stg1, M2_Stg1
  ```

  `job0` contains three tasks, `M1_Stg1`, `M2_Stg1`, and `J3_1_2_Stg1`. MaxCompute runs the `J3_1_2_Stg1` task after the `M1_Stg1` and `M2_Stg1` tasks are run.

  Naming conventions of tasks:

  - MaxCompute provides four task types: map, reduce, join, and local work. The first letter in a task name indicates the type of the task. For example, `M2Stg1` is a map task.
  - The digit that follows the first letter indicates the task ID. This ID is unique among all tasks that correspond to a specific query.
  - Digits separated by underscores (_) represent the direct dependency of a task. For example, `J3_1_2_Stg1` indicates that the task with the ID of 3 depends on the M1_Stg1 and M2_Stg1 tasks.

- Dependencies between all operators in a task

  The operator string describes the execution semantics of a task. Operator string structure:

```
In Task M2:
    Data source: mf_mc_bj.sale_detail_jt/sale_date=2013/region=china # Data source descri
bes the input of the task.
    TS: mf_mc_bj.sale_detail_jt/sale_date=2013/region=china           # TableScanOperator
        FIL: ISNOTNULL(customer_id)                                   # FilterOperator
            RS: order: +                                              # ReduceSinkOperato
r
                nullDirection: *
                optimizeOrderBy: False
                valueDestLimit: 0
                dist: HASH
                keys:
                      customer_id
                values:
                      customer_id (string)
                      total_price (double)
                partitions:
                      customer_id
In Task J3_1_2:
    JOIN:                                                          # JoinOperator
         StreamLineRead1 INNERJOIN StreamLineRead2
         keys:
             0:customer_id
             1:customer_id
        AGGREGATE: group by:customer_id                           # GroupByOperator
         UDAF: SUM(total_price) (__agg_0_sum)[Complete],SUM(total_price) (__agg_1_sum)[Co
mplete]
            RS: order: +
                nullDirection: *
                optimizeOrderBy: True
                valueDestLimit: 10
                dist: HASH
                keys:
                      customer_id
                values:
                      customer_id (string)
                      __agg_0 (double)
                      __agg_1 (double)
                partitions:
In Task R4_3:
    SEL: customer_id,__agg_0,__agg_1                              # SelectOperator
        LIM:limit 10                                              # LimitOperator
            FS: output: Screen                                    # FileSinkOperator
                schema:
                  customer_id (string) AS ashop
                  __agg_0 (double) AS ap
                  __agg_1 (double) AS bp
```

Operator descriptions:

- TableScanOperator (TS): describes the logic of `FROM` statement blocks in a query statement. The alias of the input table is displayed in the output results of the `EXPLAIN` statement.

- SelectOperator (SEL): describes the logic of `SELECT` statement blocks in a query statement. The columns that are passed to the next operator are displayed in the execution results of the `EXPLAIN` statement. Multiple columns are separated by commas (,).
  - If a column is specified, the value is displayed in the `<alias>.<column_name>` format.
  - If an expression is specified, the value is displayed as a list of functions, such as `func1(arg1_1, arg1_2, func2(arg2_1, arg2_2))`.
  - If a constant is specified, the constant value is displayed.
- FilterOperator (FIL): describes the logic of `WHERE` statement blocks in a query statement. The output results of the `EXPLAIN` statement include a `WHERE` expression, which is in a form that is similar to that of SelectOperator.
- JoinOperator (JOIN): describes the logic of `JOIN` statement blocks in a query statement. The output results of the `EXPLAIN` statement show which tables are joined in which way.
- GroupByOperator (AGGREGATE): describes the logic of aggregate operations. This operator is displayed if an aggregate function is used in a query statement. The content of the aggregate function is displayed in the execution results of the `EXPLAIN` statement.
- ReduceSinkOperator (RS): describes the logic of data distribution between tasks. If the result of a task is transferred to another task, ReduceSinkOperator must be used to distribute data at the last stage of the task. The result sorting method, distributed keys, distributed values, and columns that are used to calculate the hash value are displayed in the output results of the `EXPLAIN` statement.
- FileSinkOperator (FS): describes the logic of storage operations on final data records. If `INSERT` statement blocks are included in a query statement, the name of the table into which you want to insert data is displayed in the output results of the `EXPLAIN` statement.
- LimitOperator (LIM): describes the logic of `LIMIT` statement blocks in a query statement. The number of returned rows that are specified in a `LIMIT` statement block is displayed in the execution results of the `EXPLAIN` statement.
- MapjoinOperator (HASHJOIN): describes `JOIN` operations on large tables. This operator is similar to JoinOperator.

Sample data

The following sample source data is provided to help you understand the examples in this topic. The following statements show how to create the sale_detail and sale_detail_jt tables and insert data into the tables.

```
-- Create two partitioned tables named sale_detail and sale_detail_jt.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
create table if not exists sale_detail_jt
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add partitions to the partitioned tables sale_detail and sale_detail_jt.
alter table sale_detail add partition (sale_date='2013', region='china') partition (sale_da
te='2014', region='shanghai');
alter table sale_detail_jt add partition (sale_date='2013', region='china');
-- Insert data into the partitioned tables sale_detail and sale_detail_jt.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',100.
1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null','c5'
,null),('s6','c6',100.4),('s7','c7',100.5);
insert into sale_detail_jt partition (sale_date='2013', region='china') values ('s1','c1',1
00.1),('s2','c2',100.2),('s5','c2',100.2);
-- Create a table that you want to join.
create table shop as select shop_name, customer_id, total_price from sale_detail;
```

Examples

Execute the following statements based on the sample data:

```
-- Execute the query statement.
select a.customer_id as ashop, sum(a.total_price) as ap,count(b.total_price) as bp
from (select * from sale_detail_jt where sale_date='2013' and region='china') a
inner join (select * from sale_detail where sale_date='2013' and region='china') b
on a.customer_id=b.customer_id
group by a.customer_id
order by a.customer_id
limit 10;
-- Obtain the execution plan of the query statement.
explain
select a.customer_id as ashop, sum(a.total_price) as ap,count(b.total_price) as bp
from (select * from sale_detail_jt where sale_date='2013' and region='china') a
inner join (select * from sale_detail where sale_date='2013' and region='china') b
on a.customer_id=b.customer_id
group by a.customer_id
order by a.customer_id
limit 10;
```

The following result is returned:

```
job0 is root job
In Job job0:
root Tasks: M1
In Task M1_U0:
    TS: doc_test_dev.sale_detail_jt/sale_date=2013/region=china
        FIL: ISNOTNULL(customer_id)
            HASHJOIN:
                    Filter1 INNERJOIN Filter2
                    keys:
                        0:customer_id
                        1:customer_id
                    non-equals:
                        0:
                        1:
                    bigTable: Filter1
                LocalSortBy: order: +
                             nullDirection: *
                             keys:customer_id
                   AGGREGATE: group by:customer_id
                    UDAF: SUM(total_price) (__agg_0_sum)[Complete],COUNT(total_price) (__a
gg_1_count)[Complete]
                       LIM:limit 10
                           FS: output: Screen
                               schema:
                                 customer_id (string) AS ashop
                                 __agg_0 (double) AS ap
                                 __agg_1 (bigint) AS bp
In Task M1_U1:
    TS: doc_test_dev.sale_detail/sale_date=2013/region=china
        FIL: ISNOTNULL(customer_id)
            HASHJOIN:
                    Filter1 INNERJOIN Filter2
                    keys:
                        0:customer_id
                        1:customer_id
                    non-equals:
                        0:
                        1:
                    bigTable: Filter1
                LocalSortBy: order: +
                             nullDirection: *
                             keys:customer_id
                   AGGREGATE: group by:customer_id
                    UDAF: SUM(total_price) (__agg_0_sum)[Complete],COUNT(total_price) (__a
gg_1_count)[Complete]
                       LIM:limit 10
                           FS: output: Screen
                               schema:
                                 customer_id (string) AS ashop
                                 __agg_0 (double) AS ap
                                 __agg_1 (bigint) AS bp
```

Execute the following statements based on the sample data:

```
-- Execute the query statement.
select /*+ mapjoin(a) */
       a.customer_id as ashop, sum(a.total_price) as ap,count(b.total_price) as bp
 from (select * from sale_detail_jt
where sale_date='2013' and region='china') a
inner join (select * from sale_detail where sale_date='2013' and region='china') b
on a.total_price<b.total_price
group by a.customer_id
order by a.customer_id
limit 10;
-- Obtain the execution plan of the query statement.
explain
select /*+ mapjoin(a) */
       a.customer_id as ashop, sum(a.total_price) as ap,count(b.total_price) as bp
 from (select * from sale_detail_jt
where sale_date='2013' and region='china') a
inner join (select * from sale_detail where sale_date='2013' and region='china') b
on a.total_price<b.total_price
group by a.customer_id
order by a.customer_id
limit 10;
```

The following result is returned:

```
job0 is root job
In Job job0:
root Tasks: M1
In Task M1_U0:
    TS: doc_test_dev.sale_detail_jt/sale_date=2013/region=china
        HASHJOIN:
                 TableScan1 INNERJOIN TableScan2
                 keys:
                     0:
                     1:
                 non-equals:
                     0:
                     1:
                 bigTable: TableScan2
            FIL: LT(total_price,total_price)
                LocalSortBy: order: +
                             nullDirection: *
                             keys:customer_id
                   AGGREGATE: group by:customer_id
                    UDAF: SUM(total_price) (__agg_0_sum)[Complete],COUNT(total_price) (__a
gg_1_count)[Complete]
                       LIM:limit 10
                           FS: output: Screen
                               schema:
                                 customer_id (string) AS ashop
                                 __agg_0 (double) AS ap
                                 __agg_1 (bigint) AS bp
In Task M1_U1:
    TS: doc_test_dev.sale_detail/sale_date=2013/region=china
        HASHJOIN:
                 TableScan1 INNERJOIN TableScan2
                 keys:
                     0:
                     1:
                 non-equals:
                     0:
                     1:
                 bigTable: TableScan2
            FIL: LT(total_price,total_price)
                LocalSortBy: order: +
                             nullDirection: *
                             keys:customer_id
                   AGGREGATE: group by:customer_id
                    UDAF: SUM(total_price) (__agg_0_sum)[Complete],COUNT(total_price) (__a
gg_1_count)[Complete]
                       LIM:limit 10
                           FS: output: Screen
                               schema:
                                 customer_id (string) AS ashop
                                 __agg_0 (double) AS ap
                                 __agg_1 (bigint) AS bp
```

3.7.4. Common table expressions

A common table expression (CTE) is a temporary named result set that is used to simplify SQL queries. MaxCompute allows you to use SQL-compliant CTEs to improve the readability and execution efficiency of SQL statements. This topic describes the features, command syntax, and use examples of CTEs.

Description

A CTE can be considered a temporary result set that is defined within the execution scope of a DML statement. Similar to a derived table, a CTE is not stored as an object. It is used only during queries. CTEs improve the readability of SQL statements and simplify complex queries.

Syntax

```
with
     <cte_name> as
    (
        <cte_query>
    )
    [,<cte_name2>  as
     (
     <cte_query2>
     )
    ,……]
```

- cte_name: required. The name of a CTE. The name must be unique within a `WITH` clause. After you define a CTE, you can use the value of this parameter to indicate the CTE in the query.
- cte_query: required. A `SELECT` statement. The result set of the `SELECT` statement is filled in the specified CTE.

Example

Sample code without CTEs:

```
insert overwrite table srcp partition (p='abc')
select * from (
    select a.key, b.value
    from (
        select * from src where key is not null    ) a
    join (
        select * from src2 where value > 0    ) b
    on a.key = b.key
) c
union all
select * from (
    select a.key, b.value
    from (
        select * from src where key is not null    ) a
    left outer join (
        select * from src3 where value > 0    ) b
    on a.key = b.key and b.key is not null
)d;
```

In the preceding code, the two `JOIN` clauses on both sides of `UNION` use the output of the same subquery statement as their left tables. Therefore, the subquery statement is repeated in the code.

You can use CTEs to prevent statement repetition. Sample code with CTEs:

```
with
  a as (select * from src where key is not null),
  b as (select  * from src2 where value > 0),
  c as (select * from src3 where value > 0),
  d as (select a.key, b.value from a join b on a.key=b.key),
  e as (select a.key,c.value from a left outer join c on a.key=c.key and c.key is not null)
insert overwrite table srcp partition (p='abc')
select * from d union all select * from e;
```

In the preceding code, the subquery that corresponds to `a` is written only once and is subsequently reused as a CTE. You can specify multiple subqueries as CTEs in the same `WITH` clause. This way, you can repeatedly use them in the statement the same way as you use variables. CTEs also eliminate the need to repeatedly nest subqueries.

3.7.5. CLONE TABLE

If you want to clone data from one table to another, you can use the `CLONE TABLE` statement of MaxCompute to improve the efficiency of data cloning. This topic describes the features, limits, and syntax of the `CLONE TABLE` statement. This topic also provides examples on how to use the CLONE TABLE statement.

Description

The CLONE TABLE statement is used to clone data from a source table to a destination table. After you clone the data to the destination table, we recommend that you verify the cloned data to ensure data accuracy. For example, you can execute the `SELECT` statement to query the data in the destination table and execute the `DESC` statement to query the size of the destination table.

Limits

- The schema of the destination table must be compatible with the schema of the source table.
- You can execute the `CLONE TABLE` statement for partitioned tables and non-partitioned tables. You cannot execute the CLONE TABLE statement for clustered tables.
- If you have created the destination table before the CLONE TABLE statement is executed, you can execute the CLONE TABLE statement to clone data from up to 10,000 partitions at the same time.
- If you have not created the destination table before the CLONE TABLE statement is executed, the number of partitions from which you can clone data at the same time is unlimited. This way, the atomicity of the cloning operation is ensured.
- You can execute the `CLONE TABLE` statement up to seven times for the same non-partitioned table or for the same partition of a partitioned table.
- You cannot execute the `CLONE TABLE` statement for projects across multiple regions.
- You cannot execute the `CLONE TABLE` statement for external tables.

Syntax

```
clone table <[<src_project_name>.]<src_table_name>> [partition(<pt_spec>), ...]
 to <[<dest_project_name>.]<dest_table_name>> [if exists [overwrite | ignore]] ;
```

- src_project_name: optional. The name of the MaxCompute project to which the source table belongs. If you do not configure this parameter, the name of the current project is used. This parameter is required if the source table and destination table do not belong to the same MaxCompute project.
- src_table_name: required. The name of the source table.
- pt_spec: optional. The partition information of the source table. The value of this parameter is in the `(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)` format. partition_col specifies the name of a partition key column, and partition_col_value specifies the value of a partition key column.
- dest_project_name: optional. The name of the MaxCompute project to which the destination table belongs. If you do not configure this parameter, the name of the current project is used. This parameter is required if the source table and destination table do not belong to the same MaxCompute project.
- desc_table_name: required. The name of the destination table.
  - If you have not created the destination table, the `CLONE TABLE` statement creates the destination table by using the syntax of the `CREATE TABLE LIKE` statement. For more information about the `CREATE TABLE LIKE` statement, see Create a table.
  - If you have created the destination table and you specify `if exists overwrite`, the `CLONE TABLE` statement overwrites the data in the destination table or the data in the specified partition of the destination table.
  - If you have created the destination table and you specify `if exists ignore`, the `CLONE TABLE` statement skips the existing partitions and does not overwrite the data in the existing partitions of the destination table.

Sample data

Sample source data is provided for you to better understand the examples in this topic. The following statements show how to create a partitioned table named sale_detail and a non-partitioned table named sale_detail_np and insert data into the tables.

- sale_detail table

```
-- Create a partitioned table named sale_detail.
create table if not exists sale_detail
(
shop_name     string,
customer_id   string,
total_price   double
)
partitioned by (sale_date string, region string);
-- Add partitions to the sale_detail table.
alter table sale_detail add partition (sale_date='2013', region='china') partition (sale_
date='2014', region='shanghai');
-- Insert data into the sale_detail table.
insert into sale_detail partition (sale_date='2013', region='china') values ('s1','c1',10
0.1),('s2','c2',100.2),('s3','c3',100.3);
insert into sale_detail partition (sale_date='2014', region='shanghai') values ('null','c
5',null),('s6','c6',100.4),('s7','c7',100.5);
```

Query data in the sale_detail table. Sample statement:

```
select * from sale_detail;
-- The following result is returned:
+------------+------------+------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+------------+------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+------------+------------+------------+------------+
```

- sale_detail_np table

```
-- Create a non-partitioned table named sale_detail_np.
create table if not exists sale_detail_np
(
shop_name     string,
customer_id   string,
total_price   double
);
-- Insert data into the sale_detail_np table.
insert into sale_detail_np values ('s4','c4',100.4);
```

Query data in the sale_detail_np table. Sample statement:

```
select * from sale_detail_np;
-- The following result is returned:
+------------+------------+-------------+
| shop_name  | customer_id | total_price |
+------------+------------+-------------+
| s4         | c4          | 100.4       |
+------------+------------+-------------+
```

Examples

This section provides examples on how to use the `CLONE TABLE` statement based on the sample data.

- Example 1: Clone full data from the sale_detail_np table to the sale_detail_np_clone table. Sample statements:

```
-- Clone full data from the sale_detail_np table to the sale_detail_np_clone table.
clone table sale_detail_np to sale_detail_np_clone;
-- Query data in the sale_detail_np_clone table to verify the accuracy of the data.
select * from sale_detail_np_clone;
-- The following result is returned:
+------------+-------------+-------------+
| shop_name  | customer_id | total_price |
+------------+-------------+-------------+
| s4         | c4          | 100.4       |
+------------+-------------+-------------+
```

- Example 2: Clone data from a specified partition of the sale_detail table to the sale_detail_clone table. Sample statements:

```
-- Clone data from a specified partition of the sale_detail table to the sale_detail_clon
e table.
clone table sale_detail partition (sale_date='2013', region='china') to sale_detail_clone
if exists overwrite;
-- Query data in the sale_detail_clone table to verify the accuracy of the data.
select * from sale_detail_clone;
-- The following result is returned:
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
+------------+-------------+-------------+------------+------------+
```

- Example 3: Clone full data from the sale_detail table to the sale_detail_clone table that is created in Example 2 and skip the existing partitions in the sale_detail_clone table. Sample statements:

```
-- Clone full data from the sale_detail table to the sale_detail_clone table.
clone table sale_detail to sale_detail_clone if exists ignore;
-- Query data in the sale_detail_clone table to verify the accuracy of the data.
select * from sale_detail_clone;
-- The following result is returned:
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

- Example 4: Clone full data from the sale_detail table to the sale_detail_clone1 table. Sample statements:

```
-- Clone full data from the sale_detail table to the sale_detail_clone1 table.
clone table sale_detail to sale_detail_clone1;
-- Query data in the sale_detail_clone1 table to verify the accuracy of the data.
select * from sale_detail_clone1;
-- The following result is returned:
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

3.7.6. Parameterized view

The parameterized view feature allows you to import tables or variables to customize views. This topic describes the parameterized view feature that is supported by the MaxCompute SQL engine.

Background information

In the traditional views of MaxCompute, complex SQL scripts are encapsulated at the underlying layer. You can call views like reading a standard table without the need to understand the implementation mechanism at the underlying layer. Traditional views are widely used because they implement code encapsulation and reuse. However, traditional views do not support parameters. For example, you cannot pass filter conditions to filter data in the underlying tables of a view or pass other parameters to the view. This makes code reuse efficiency low. The MaxCompute SQL engine supports parameterized views and allows you to import tables or variables to customize views.

Syntax

```
create [or replace] [if not exists] <view_name>( <variable_name> <variable_type> [, <variab
le_name> <variable_type> ...])
[returns <return_variable> table (<col_name> <col_type> comment <col_comment> [,<col_name>
<col_type> comment <col_comment>])]
[comment <view_comment>]
as
{<select_statement> | begin <statements> end}
```

- view_name: required. This parameter specifies the name of the view.
- variable_name: required. This parameter specifies the name of the view variable.
- variable_type: required. This parameter specifies the type of the view variable.
- return_variable: optional. This parameter specifies the name of the variable returned by the view.
- col_name: optional. This parameter specifies the name of the column returned by the view.
- col_type: optional. This parameter specifies the type of the column returned by the view.
- col_comment: optional. This parameter specifies the comment of the column returned by the view.

- view_comment: optional. This parameter specifies the comment of the view.
- select_statement: the `SELECT` clause. You must specify this clause or the statements clause.
- statements: the script of the view. You must specify this clause or the select_statement clause.

Define a parameterized view

To create a parameterized view, use the following syntax:

```
-- view with parameters
-- param @a -a table parameter
-- param @b -a string parameter
-- returns a table with schema (key string, value string)
create view if not exists pv1(@a table (k string,v bigint), @b string)
as
select srcp.key,srcp.value from srcp join @a on srcp.key=a.k and srcp.p=@b;
```

Description:

- Parameters are defined. Therefore, you must use **Script Mode SQL** to define a parameterized view.
- The created pv1 view has two parameters, table and string. You can set the table parameter to a table name and the string parameter to a basic data type.
- You can also use a subquery, such as `select * from view_name( (select 1 from src where a > 0), 1);`, as a parameter value.
- When you create a view, you can set a parameter to any, which indicates any data type. For example, use `create view paramed_view (@a any) as select * from src where case when @a is null then key1 else key2 end = key3;` to define a view. In this example, the first parameter of the view can be a value of any data type.

  However, the value any cannot be used in operations, such as `+` or `AND`, that require specific data types. A field of the any type is often used as a PassThrough column in the table specified by the table parameter. Example:

  ```
  create view paramed_view (@a table(name string, id any, age bigint)) as select * from @a
  where name = 'foo' and age < 25;
  -- Call the view.
  select * from param_view((select name, id, age from students));
  ```

  > **Note** After you execute the `CREATE VIEW` statement to create a view, you can execute the `DESC` statement to obtain the description of the view. The description contains the return type of the view.
  >
  > The return type of a view is recalculated when the view is called. The return type may be different from the return type, such as any, that you specify when you create the view.

- When you create a view, you can use an asterisk (*) in the table parameter to retrieve any columns. If an asterisk (*) is used to indicate any columns, you can specify a specific data type for the columns or use the value any. Example:

```
create view paramed_view (@a table(key string, * any), @b table(key string, * string)) as
select a.* from @a join @b ON a.key = b.key;
-- Call the view.
select name, address from param_view((select school, name, age, address from student), sc
hool) where age < 20;
```

In this example, the view uses two table parameters. In the table specified by the first table parameter, the first column is of the STRING type and can be followed by any columns of any data types. In the table specified by the second table parameter, the first column is of the STRING type and can be followed by any columns of the STRING type. Take note of the following points:

- The varied-length part must be placed at the end of the table that is specified by the table parameter. This means that the columns indicated by an asterisk (*) cannot be followed by other columns. Therefore, the table that is specified by the table parameter can contain only one varied-length list.
- The varied-length part must be placed at the end of the table that is specified by the table parameter. However, the columns of an input table may not be arranged in this sequence. In this case, the columns need to be rearranged. A subquery can be used as a parameter value and must be enclosed in a pair of parentheses ().
- No name is specified for the varied-length part of the table that is specified by the table parameter. As a result, data in the varied-length part cannot be referenced or calculated when you define the view.
- Although you cannot calculate the varied-length part, you can use the `SELECT *` statement to transfer data in the varied-length part out of the table.
- The columns of the table that is specified by the table parameter may be different from the fixed-length columns that you specify when you define the view. If the column names are different, the compiler automatically renames them. If the data types are different, the compiler performs an implicit conversion. If the implicit conversion fails, an error occurs.

Call a parameterized view

You can execute the following statement to call the pv1 view that you defined:

```
@a := select * from src where value >0;
--call view with table variable and scalar
@b := select * from pv1(@a,'20170101');
@another_day := '20170102';
--call view with table name and scalar variable
@c := select * from pv1(src2, @another_day);
@d := select * from @c union all select * from @b;
with
t as(select * from src3)
select * from @c
union all
select * from @d
union all
select * from pv1(t,@another_day);
```

> **Note** You can use different parameters to call the pv1 view.
>
> - The value of the table parameter can be the name of a physical table, view, or table variable. You can also set the table parameter to a table alias in a common table expression (CTE).
> - The values of common parameters can be variables or constants.

Usage notes

- In a parameterized view, only DML statements can be used in scripts. The SELECT statements that directly return outputs, `INSERT`, or `CREATE TABLE` cannot be used.
- A parameterized view can contain multiple SQL statements.

```
-- view with parameters
-- param @a -a table parameter
-- param @b -a string parameter
-- returns a table with schema (key string,value string)
create view if not exists pv2(@a table (k string,v bigint), @b string) as
begin
@srcp := select * from srcp where p=@b;
@pv2 := select srcp.key,srcp.value from @srcp join @a on srcp.key=a.k;
end;
```

> **Note** Content between `begin` and `end` is the script of this view. `@pv2 :=...` is similar to the `RETURN` statement in other programming languages. @pv2 is used to assign a value to an implicit table variable that has the same name as the view.

- The matching rules for actual and formal view parameters are the same as those specified in a common weakly-typed language. If a view parameter can be implicitly converted, it can match the defined parameter. For example, a value of the BIGINT type can match parameters of the DOUBLE type. For table variables, if the schema of Table a can be inserted into Table b, Table a can be used to match the table-type parameters that have the same schema as Table b.
- You can explicitly declare the return type to make the code easier to read.

```
create view if not exists pv3(@a table (k string,v bigint), @b string)
returns @ret table (x string comment 'This is the x',y string comment 'This is the y')
comment 'This is view pv3'
as
begin
    @srcp := select * from srcp where p=@b;
    @ret := select srcp.key,srcp.value from @srcp join @a on srcp.key=a.k;
end;
```

> **Note** `returns @ret table (x string,y string)` defines the following information:
>
> - The return type is `table (x string,y string)`, which indicates the type returned to the caller. You can use this parameter to customize the table schema.
> - The response parameter is `@ret`. In the view script, the parameter is assigned a value. This parameter defines the name of the response parameter.
>
> You can consider a view that contains no `begin/end` or return variables as a simplified parameterized view.

3.8. FAQ about SQL

3.8.1. FAQ about DDL operations

This topic provides answers to some frequently asked questions about DDL operations on tables, partitions, and columns in MaxCompute.

Category	FAQ
Table operation	• Does MaxCompute support virtual tables, such as DUAL tables in MySQL? • Does a MaxCompute table have indexes? • How do I manage the Hash Clustering attribute of a table? • How do I change a non-partitioned table to a partitioned table? • Can I restore a table that is accidentally deleted? • How do I query tables that are created by a specified user? • How do I check whether a specified table exists? • How do I obtain the names of all tables in my project? • How do I quickly identify partitioned tables in a project? • How do I view the time when a specified MaxCompute table is most recently accessed? • How do I view the amount of data in a table? • How do I view the number of rows in a table?
Partition operation	• What is the difference between partitions and partition key columns? • Is it recommended for a table to contain a large number of partitions? • Can I add or modify partition fields in a source table if the source table does not have partition fields? • How do I check whether a specified partition exists? • How do I obtain the number of partitions?

Category	FAQ
Column operation	• Can I add columns to or delete columns from a MaxCompute table? • How do I add columns to a MaxCompute table? • How do I configure auto-increment columns? • What is the maximum number of columns that can be stored in a MaxCompute table? • What do I do if the name of a column in the table that I want to create is the same as the keyword? • How do I change the data type of a column? • How do I delete a column from a table?
Lifecycle management operation	If I set the lifecycle of a partitioned table to 3 days and each partition of the table contains a large amount of data, how does the system delete historical data of the table?

Does MaxCompute support virtual tables, such as DUAL tables in MySQL?

No, MaxCompute does not support virtual tables. You can manually create a DUAL table.

Does a MaxCompute table have indexes?

No, a MaxCompute table does not have indexes. You can use the Hash Clustering attribute to achieve a similar effect as clustered indexes in databases. For more information, see Table operations.

How do I manage the Hash Clustering attribute of a table?

- To add the Hash Clustering attribute of a table, run the following command: `alter table table_name [clustered by (col_name [, col_name, ...]) [sorted by (col_name [asc | desc] [, col_name [asc | desc] ...])] into number_of_buckets buckets];` .
- To remove the Hash Clustering attribute of a table, run the following command: `alter table table_name not clustered;` .

How do I change a non-partitioned table to a partitioned table?

You cannot change a non-partitioned table to a partitioned table or add partition key columns. You can create a partitioned table. For more information, see Table operations.

Can I restore a table that is accidentally deleted?

Yes, you can restore the deleted table because MaxCompute supports the backup and restoration feature. If you accidentally delete a table, you can run the backup and restoration commands to restore the table free of charge within 24 hours. You can customize the backup data retention period. For more information about the backup and restoration feature, see Backup and restoration.

How do I query tables that are created by a specified user?

If you want to query tables that are created by a specified user, you can use the metadata view TABLES to filter tables based on the owner_name field. For more information about TABLES, see TABLES.

How do I check whether a specified table exists?

You can use the TABLE_EXISTS function to check whether the specified table exists. For more information, see TABLE_EXISTS.

How do I obtain the names of all tables in my project?

You can run the `show tables;` command on the MaxCompute client or use the metadata service of MaxCompute to obtain the names of all tables in your project. For more information, see Table operations or Information Schema.

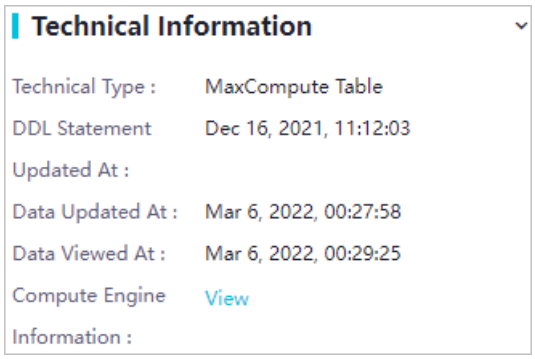
How do I quickly identify partitioned tables in a project?

You can run the following command on the MaxCompute client to view the partitioned tables in your project:

```
select table_name from information_schema.columns where is_partition_key = true group by table_name;
```

How do I view the time when a specified MaxCompute table is most recently accessed?

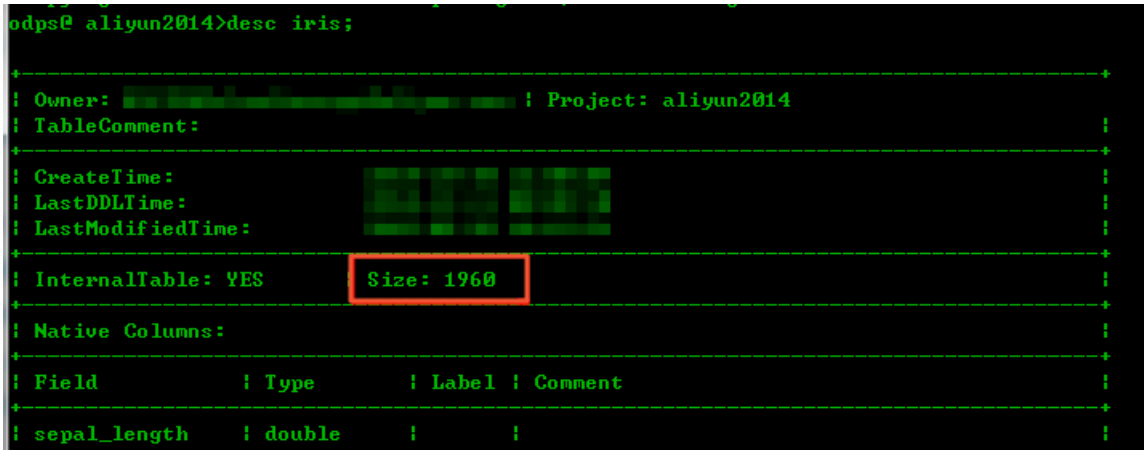
You can go to Data Map of DataWorks, click the name of the table that you want to query, and then go to the table details page to view the time when the table is most recently accessed.



How do I view the amount of data in a table?

You can view the number of data records in a table and the physical space occupied by the table.

- You can run the `desc` command to view the physical space of the full table. You can execute the SQL statement `select count() as cnt from table_name;` to view the number of data records in the table.



- You can run the `desc` command with a WHERE clause to view the physical space that is occupied by a partition in a partitioned table. You can execute the SQL statement `select count() as cnt from table_name where ...;` to view the number of data records in a partition.

How do I view the number of rows in a table?

You can execute the `select count(*) from table_name;` statement on the MaxCompute client to view the number of rows in a partitioned table or a non-partitioned table.

What is the difference between partitions and partition key columns?

Tables in MaxCompute can be partitioned. A partitioned table contains partition key columns. You can use a partition key column to create partitions.

For example, `ds` in `ds=20150101` indicates a partition key column and `ds=20150101` indicates a partition.

Is it recommended for a table to contain a large number of partitions?

A table in MaxCompute can have a maximum of 60,000 partitions, and the capacity of each partition is not limited. However, excessive partitions in a table cause inconvenience for data collection and analysis.

In MaxCompute, the number of instances is limited in a job. The number of instances in a job is closely related to the amount of input data and the number of partitions. Therefore, we recommend that you determine the number of partitions in a table based on your business requirements.

Can I add or modify partition fields in a source table if the source table does not have partition fields?

No, MaxCompute does not allow you to add or modify partition fields in a source table. After a partition field is created, it cannot be modified. You can create a partitioned table and use an SQL statement that is used for dynamic partitioning to import data from the source table to the new partitioned table. For more information, see Insert or overwrite data into dynamic partitions (DYNAMIC PARTITION).

How do I check whether a specified partition exists?

You can use the PARTITION_EXISTS function to check whether the specified partition exists. For more information about the PARTITION_EXISTS function, see PARTITION_EXISTS.

How do I obtain the number of partitions?

You can use the PARTITIONS view of Information Schema to obtain the partition names. Then, you can count the number of partitions.

Can I add columns to or delete columns from a MaxCompute table?

You can add columns to a MaxCompute table, but you cannot delete columns from a MaxCompute table.

How do I add columns to a MaxCompute table?

The following example shows how to add columns to a MaxCompute table. If data exists in the table, the values of the added column are NULL.

```
alter table table_name add columns (col_name1 type1, col_name2 type2…);
```

For more information about how to add columns, see Add columns.

How do I configure auto-increment columns?

MaxCompute does not support auto-increment columns. If you want to use auto-increment columns and the amount of data is small, we recommend that you use Window functions.

What is the maximum number of columns that can be stored in a MaxCompute table?

A MaxCompute table can store a maximum of 1,200 columns. If the number of columns exceeds the limit, perform one of the following operations:

- Perform dimension reduction on the data to reduce the number of columns to no more than 1,200.
- Change the data storage method. For example, use a device certificate, or use a sparse or dense matrix.

What do I do if the name of a column in the table that I want to create is the same as the keyword?

If you use a keyword to name a table, a column, or a partition, you must enclose the keyword in a pair of grave accents (``). Otherwise, an error is returned.

How do I change the data type of a column?

You cannot change the data types of columns. You can only add columns. You cannot delete table fields or modify table fields or partition fields in a table in the production environment. If you need to modify the data type of a column in a table, you must delete the table and recreate a table. You can also create an external table. After you delete and recreate the table, you can reload data. For more information about data types, see Data type editions.

How do I delete a column from a table?

You cannot delete a column from a MaxCompute table. If you want to delete a column, perform the following steps:

1. Run the following command to create a table:

```
create table new_table_name as select c1,c2,c3 from old_table_name;
```

2. Run the following command to delete the original table and rename the new table:

```
drop table old_table_name;
alter table new_table_name rename to old_table_name;
```

If I set the lifecycle of a partitioned table to 3 days and each partition of the table contains a large amount of data, how does the system delete historical data of the table?

If a partition in the table is not modified within the lifecycle, MaxCompute automatically reclaims the partition.

You can run the `desc table_name partition(pt_spec);` command to check whether a specified partition is modified within the lifecycle. You can run the `desc tablename;` command to query the lifecycle of the table. MaxCompute reclaims tables and partitions at 17:00:00 every day. In most cases, related data is displayed on DataWorks with a delay of one day.

3.8.2. FAQ about DML operations

This topic provides answers to some frequently asked questions about DML operations on tables in MaxCompute.

Category	FAQ

Category	FAQ
Data insertion or update	• Is the original data damaged if an error occurs during an INSERT operation? • What do I do if a field mismatch error appears when I execute the INSERT INTO or INSERT OVERWRITE statement? • What do I do if the "a single instance cannot output data to more than 10000 partitions" error message appears when I execute the INSERT INTO or INSERT OVERWRITE statement? • What do I do if the "invalid dynamic partition value" error message appears when I insert a dynamic partition into a MaxCompute table? • What do I do if an error occurs when I insert data of the FLOAT type into a MaxCompute table? • After I execute the INSERT INTO SELECT statement and the SELECT statement for the same data, the returned results are different. Why? • When I insert data into a field of the VARCHAR(10) type in the destination table, is an error returned if the data length overflows? • What do I do if the "Transaction timeout because cannot acquire exclusive lock" error message appears when I execute a MaxCompute SQL statement? • How do I update data in a MaxCompute table or partition?
Data deletion	• How do I delete data from a MaxCompute table or partition? • If a non-partitioned table contains a large amount of data, how do I delete duplicate data from the table?

Is the original data damaged if an error occurs during an INSERT operation?

No, the original data is not damaged. MaxCompute is atomic. If the INSERT operation succeeds, data is updated. If the INSERT operation fails, data is rolled back.

What do I do if a field mismatch error appears when I execute the INSERT INTO or INSERT OVERWRITE statement?

When you execute the INSERT INTO or INSERT OVERWRITE statement to insert data, you must make sure that the field sequence, field types, and the total number of fields in the returned results of the SELECT statement match those in the destination table. MaxCompute does not allow you to specify a field in an INSERT statement. If you want to insert NULL values or other default values into some columns, you can specify NULL or the default values in the SELECT statement. For example, you can use `select 'a', null, col_name from table_name;` with an INSERT statement to insert the a and null values into the first two columns and insert the return values of the col_name column into the third column.

What do I do if the "a single instance cannot output data to more than 10000 partitions" error message appears when I execute the INSERT INTO or INSERT OVERWRITE statement?

- Problem description

  When I execute the INSERT INTO or INSERT OVERWRITE statement, the following error message appears:

```
FAILED: ODPS-0123031:Partition exception - a single instance cannot output data to more t
han 10000 partitions
```

- Cause

  A MaxCompute table can contain a maximum of 60,000 partitions. However, a maximum of only 10,000 partitions are allowed in the output table for a job. In most cases, this error is caused by the incorrect configuration of partition fields. For example, partitioning based on the ID field causes excessive partitions.

- Solution

  In most cases, the number of partitions is large if the output table of a job contains thousands of dynamic partitions. If the number of partitions in the table exceeds 10,000, an error may occur in the business logic or SQL syntax. If no logic or syntax error occurs, we recommend that you modify the partition fields of the partitioned table or split the business logic into multiple jobs to avoid this error.

What do I do if the "invalid dynamic partition value" error message appears when I insert a dynamic partition into a MaxCompute table?

- Problem description

  When I insert a dynamic partition into a MaxCompute table, the following error message appears:

```
FAILED: ODPS-0123031:Partition exception - invalid dynamic partition value: province=shan
ghai
```

- Cause

  The dynamic partition is invalid. Dynamic partitioning is performed based on specified fields. Fields that contain special characters or Chinese characters cannot be used as dynamic partition fields.

- Solution

  When you insert dynamic partitions into a MaxCompute table, take note of the following rules:

  - When you insert dynamic partitions into a MaxCompute table in a distributed environment, a maximum of 512 dynamic partitions can exist in a single process.
  - An SQL statement that is used for dynamic partitioning cannot generate more than 2,000 dynamic partitions.
  - Partition values that are dynamically generated cannot be NULL.
  - If the destination table contains multi-level partitions, you can specify some partitions as static partitions in an INSERT statement. However, the static partitions must be high-level partitions.

What do I do if an error occurs when I insert data of the FLOAT type into a MaxCompute table?

For more information about the basic data types supported by the MaxCompute V2.0 data type edition, see Data type editions. Data of the FLOAT data type has no constants. To insert data of this type, you can first use the CAST function to convert data into the FLOAT data type. For example, you can use `cast( 5.1as float)` to convert `'5.1'` of the STRING type into `5.1` of the FLOAT type.

If you want to use new data types, such as TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, or BINARY, in MaxCompute SQL, you must run one of the following commands to enable the MaxCompute V2.0 data type edition for a session or project:

- To enable the MaxCompute V2.0 data type edition for a session, you must add `set odps.sql.type.`

`system.odps2=true;` before the SQL statement, and then commit and execute them together.

- To enable the MaxCompute V2.0 data type edition for a project, you must run the `setproject odps.sql.type.system.odps2=true;` command for the project. This command must be run by the project owner.

After I execute the INSERT INTO SELECT statement and the SELECT statement for the same data, the returned results are different. Why?

- Problem description

  After I separately execute the INSERT INTO SELECT statement and the SELECT statement for the same field of the STRING type, the decimal places in the returned result are different. In the result of the SELECT statement, two decimal places are reserved. In the result of the INSERT INTO SELECT statement, multiple decimal places are displayed.

- Cause

  If the field in the INSERT INTO SELECT statement is of the STRING type, the field is first converted into the DOUBLE type. Then, the ROUND operation is performed on the field of the DOUBLE type when the field is implicitly converted into the DECIMAL type. Data of the DOUBLE type is inaccurate. Therefore, multiple decimal places may still be displayed even if the ROUND operation is performed.

- Solution

  We recommend that you explicitly convert data types. You can add the following statement to use CAST to explicitly convert the data type to the DECIMAL type.

```
case when pcm.abc is null then 0
                    else round(cast(pcm.abc as decimal) ,2)
                end abc
```

When I insert data into a field of the VARCHAR(10) type in the destination table, is an error returned if the data length overflows?

When data is inserted into a field of the `VARCHAR(10)` type, the data is truncated and no error is returned if the data length overflows.

What do I do if the "Transaction timeout because cannot acquire exclusive lock" error message appears when I execute a MaxCompute SQL statement?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

```
Failed to run ddltask - Modify DDL meta encounter exception : ODPS-0121096:MetaStore tran
saction conflict - Reached maximum retry times because of OTSStorageTxnLockKeyFail(Inner
exception: Transaction timeout because cannot acquire exclusive lock.)
```

- Cause

  MaxCompute does not have a locking mechanism for the table on which you perform operations. The error is caused by metadata competition. In this case, you must check whether multiple read/write operations are performed on the table or table partitions at the same time.

- Solution

  We recommend that you do not perform multiple read/write operations on a table at the same time.

How do I update data in a MaxCompute table or partition?

MaxCompute allows you to execute the `UPDATE` statement to update data of specific rows in transactional tables.

If the table is not a transactional table, you must import data from the source partition or the source table to a new partition or table. The update operation is performed during the import. The name of the new partition or new table can be the same as the name of the source partition or source table. This way, the data is updated in the source partition or source table.

How do I delete data from a MaxCompute table or partition?

MaxCompute allows you to execute the `DELETE` statement to delete data from specific rows in transactional tables.

If the table is not a transactional table, use the following method to delete data:

- Run the `drop` command to delete the table. This way, the data is also deleted.
- If the table is a non-partitioned table, you can run the `truncate table table_name;` command to delete data from the table or run the `insert overwrite` command to perform a similar operation.
  - Example 1: Delete the data records in which the value of Col is 1 from TableA. Sample command:

    ```
    insert overwrite table TableA select a,b,c.... from TableA where Col <> 1;
    ```

  - Example 2: Delete all data. Sample command:

    ```
    insert overwrite table TableA select a,b,c.... from TableA where 1=2;
    ```

- If you want to delete data from a partitioned table, you can run the `alter table table_name drop if exists partition(Partition key column='Specific partition value')` command to delete the partition. This way, the data in the partition is deleted.

  For example, if the partition key column of the testtable table is ds, run the following command to delete the `ds='20170520'` partition:

  ```
  alter table testtable drop if exists partition (ds='20170520');
  ```

- Execute the INSERT statement and the WHERE clause to import the required data to a new partition or table. If you execute the INSERT statement, the source table and destination table can be the same.

  ```
  insert overwrite table sale_detail select * from sale_detail where name='mengyonghui';
  ```

If a non-partitioned table contains a large amount of data, how do I delete duplicate data from the table?

If each column in the non-partitioned table contains duplicate data, you can perform the GROUP BY operation on all columns. For example, if the columns of the non-partitioned table *table1* are c1, c2, and c3. You can run the following command to delete duplicate data from the table:

```
insert overwrite table table1 select c1, c2, c3 from table1 group by
 c1, c2, c3;
```

Note Before you perform this operation, we recommend that you back up data and evaluate whether the cost of this operation is lower than the cost of data import based on the amount of data.

3.8.3. FAQ about DQL operations

This topic provides answers to some frequently asked questions about data query language (DQL) operations in MaxCompute.

Category	FAQ
GROUP BY	• What do I do if the "Repeated key in GROUP BY" error message appears when I execute a MaxCompute SQL statement? • What do I do if the "Expression not in GROUP BY key" error message appears when I execute a MaxCompute SQL statement? • Table B is generated after I execute GROUP BY on Table A. The number of rows in Table B is less than the number of rows in Table A, but the physical storage capacity of Table B is ten times the physical storage capacity of Table A. Why? • Is query performance affected when I use GROUP BY to query 10 billion data records? Is the data amount limited when I use GROUP BY to query data?
ORDER BY	• After I query data in MaxCompute, how are the query results sorted? • Does MaxCompute support ORDER BY FIELD NULLS LAST? • What do I do if the "ORDER BY must be used with a LIMIT clause" error message appears when I execute a MaxCompute SQL statement?
Subquery	When I execute a MaxCompute SQL statement with NOT IN that is followed by a subquery, the subquery is expected to return tens of thousands of data records. However, if a subquery that follows IN or NOT IN returns partition data, the maximum number of data records that can be returned is 1,000. How do I ensure that the subquery returns the expected number of data records and the logic of NOT IN is implemented?
INTERSECT, UNION, EXCEPT, and MINUS	• How do I merge two tables that are not associated with each other? • What do I do if the "ValidateJsonSize error" error message appears when I perform UNION ALL operations?

Category	FAQ
JOIN	• What do I do if the "Both left and right aliases encountered in JOIN" error message appears when I perform a JOIN operation? • What do I do if the "Maximum 16 join inputs allowed" error message appears when I perform a JOIN operation? • What do I do if the number of returned data records is greater than the number of data records in one of the source tables after I perform a JOIN operation? • When I perform a JOIN operation, does partition pruning take effect if the partition pruning condition is specified in an ON clause or a WHERE clause?
MAPJOIN	• How do I use MAPJOIN to cache multiple small tables? • Can I exchange the large table and small tables that are specified in the MAPJOIN statement?
Others	• After I configure filter conditions in a MaxCompute SQL statement, an error message indicating that the size of the input data exceeds 100 GB appears. What do I do? • Does the WHERE condition of a fuzzy query in MaxCompute SQL support regular expressions? • If I want to synchronize only 100 data records, how do I use LIMIT to specify the number of data records that I want to synchronize in a WHERE clause? • How can I improve the query efficiency? Can I adjust the partition settings? • Does MaxCompute SQL support the WITH AS statement? • How do I split one row of data into multiple rows? • After I specify use_instance_tunnel=false and instance_tunnel_max_record=10 in the odps_config.ini file of the client, the SELECT statement still generates a large number of output records. Why? • How do I use a regular expression to determine whether the values of a field are in Chinese?

What do I do if the "Repeated key in GROUP BY" error message appears when I execute a MaxCompute SQL statement?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

  ```
  FAILED: ODPS-0130071:Semantic analysis exception - Repeated key in GROUP BY.
  ```

- Cause

  SELECT DISTINCT cannot be followed by a constant.

- Solution

  Split the execution logic of the SQL statement into two layers. This way, the DISTINCT logic without constants is processed at the inner layer, and constant data is added to the outer layer.

What do I do if the "Expression not in GROUP BY key" error message appears when I execute a MaxCompute SQL statement?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

  ```
  FAILED: ODPS-0130071:Semantic analysis exception - Expression not in GROUP BY key : line 1:xx 'xxx'
  ```

- Cause

  Columns that are not specified in the GROUP BY clause cannot be directly referenced. For more information, see GROUP BY (col_list).

- Solution

  Modify the SQL statement to ensure that the columns that are queried by using SELECT are the columns that are specified in the GROUP BY clause or the columns that are processed by using aggregate functions, such as SUM or COUNT.

Table B is generated after I execute GROUP BY on Table A. The number of rows in Table B is less than the number of rows in Table A, but the physical storage capacity of Table B is ten times the physical storage capacity of Table A. Why?

In MaxCompute, data is stored in columnar compression mode. If the data in the adjacent columns is similar, the data compression ratio is high. If `odps.sql.groupby.skewindata` is set to true, data is scattered when you execute an SQL statement to write data. In this case, the data compression ratio is low. If you want to obtain a high data compression ratio, you can sort specific data when you execute an SQL statement to write data.

Is query performance affected when I use GROUP BY to query 10 billion data records? Is the data amount limited when I use GROUP BY to query data?

No, query performance is not affected when you use GROUP BY to query 10 billion data records. The data amount is not limited when you use GROUP BY to query data. For more information about GROUP BY, see GROUP BY (col_list).

After I query data in MaxCompute, how are the query results sorted?

The data of MaxCompute tables is arranged in random order. If you do not configure order settings, the data is also returned in random order.

If you want to obtain sorted data, configure order settings for the data. For example, you can specify `order by xx limit n` in the SQL statement to sort data.

If you want to sort full data, set `n` after `limit` to `the value of the total number of data records + 1`.

> **Note** If you sort large amounts of full data, query performance is significantly affected and memory overflow may occur. We recommend that you do not perform this operation.

Does MaxCompute support ORDER BY FIELD NULLS LAST?

No, MaxCompute does not support ORDER BY FIELD NULLS LAST. For more information about the syntax supported by MaxCompute, see Differences in the support for SQL statements.

What do I do if the "ORDER BY must be used with a LIMIT clause" error message appears when I execute a MaxCompute SQL statement?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

  ```
  FAILED: ODPS-0130071:[1,27] Semantic analysis exception - ORDER BY must be used with a LI
  MIT clause, please set odps.sql.validate.orderby.limit=false to use it.
  ```

- Cause

  The ORDER BY clause needs to sort all data of a single node. By default, the ORDER BY clause is used with the LIMIT clause to prevent a single node from processing large amounts of data.

- Solution

  You can remove the limit on the simultaneous execution of the ORDER BY and LIMIT clauses for a project or session.

  - To remove the limit for a project, run the `setproject odps.sql.validate.orderby.limit=false;` command.
  - To remove the limit for a session, commit and run the `set.odps.sql.validate.orderby.limit=false;` command with the SQL statement that you want to execute.

    > **Note** If a single node has large amounts of data to sort after you remove the limit, a large number of resources and much time are consumed.

For more information about ORDER BY, see ORDER BY (order_condition).

When I execute a MaxCompute SQL statement with NOT IN that is followed by a subquery, the subquery is expected to return tens of thousands of data records. However, if a subquery that follows IN or NOT IN returns partition data, the maximum number of data records that can be returned is 1,000. How do I ensure that the subquery returns the expected number of data records and the logic of NOT IN is implemented?

You can use `LEFT OUTER JOIN` to query data.

```
select * from a where a.ds not in (select ds from b);
Replace the preceding statement with the following statement:
select a.* from a left outer join (select distinct ds from b) bb on a.ds=bb.ds where bb.ds
is not null;
```

How do I merge two tables that are not associated with each other?

You can perform `UNION ALL` operations to complete vertical merging. If you want to implement horizontal merging, you can use the `row_number` function to add an ID column to both tables. Then, associate the tables by using the ID columns and read the fields of the tables. For more information, see UNION or ROW_NUMBER.

What do I do if the "ValidateJsonSize error" error message appears when I perform UNION ALL operations?

- Problem description

  When I execute the SQL statement `select count(1) as co from client_table union all ...` that contains 200 UNION All operations, the following error message appears:

```
FAILED: build/release64/task/fuxiWrapper.cpp(344): ExceptionBase: Submit fuxi Job failed,
{
    "ErrCode": "RPC_FAILED_REPLY",
    "ErrMsg": "exception: ExceptionBase:build/release64/fuxi/fuximaster/fuxi_master.cpp(1
018): ExceptionBase: StartAppFail: ExceptionBase:build/release64/fuxi/fuximaster/app_mast
er_mgr.cpp(706): ExceptionBase: ValidateJsonSize error: the size of compressed plan is la
rger than 1024KB\nStack
```

- Causes
  - Cause 1: After an SQL statement is converted into an execution plan, the length of the execution plan exceeds 1024 KB, which is the maximum size allowed by the underlying architecture. As a result, an SQL execution error is returned. The length of the execution plan is not directly related to the length of the SQL statement. Therefore, the length of the execution plan cannot be estimated.
  - Cause 2: The length of the execution plan exceeds the limit due to a large number of partitions.
  - Cause 3: The SQL statement fails to run due to excessive small files.
- Solutions
  - Solution to Cause 1: If an SQL statement is excessively long, we recommend that you split the statement into multiple statements. This prevents the generated execution plan from exceeding the maximum length.
  - Solution to Cause 2: If a large number of partitions exist, adjust the number of partitions. For more information, see Partition.
  - Solution to Cause 3: If excessive small files exist, merge small files.

What do I do if the "Both left and right aliases encountered in JOIN" error message appears when I perform a JOIN operation?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

```
FAILED: ODPS-0130071:Semantic analysis exception - Both left and right aliases encountere
d in JOIN : line 3:3 'xx': . I f you really want to perform this join, try mapjoin
```

- Causes
  - Cause 1: The ON condition in the SQL statement includes a non-equi join, such as `table1.c1>table2.c3`.

- Cause 2: The data on one side of the JOIN condition in the SQL statement comes from two tables, such as `table1.col1 = concat(table1.col2,table2.col3)`.

- Solutions
  - Solution to Cause 1: Change the non-equi join in the ON condition in the SQL statement to an equi-join.
  - Solution to Cause 2: If one of the tables is small, use the MAPJOIN method.

What do I do if the "Maximum 16 join inputs allowed" error message appears when I perform a JOIN operation?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

  ```
  FAILED: ODPS-0123065:Join exception - Maximum 16 join inputs allowed
  ```

- Cause

  A MaxCompute SQL statement can perform MAPJOIN on a maximum of six small tables and can consecutively join a maximum of 16 tables.

- Solution

  Join several small tables into a temporary table as an input table to reduce the number of input tables.

What do I do if the number of returned data records is greater than the number of data records in one of the source tables after I perform a JOIN operation?

- Problem description

  After I execute the following MaxCompute SQL statement, the number of returned data records is greater than the number of data records in the *table1* table.

  ```
  select count(*) from table1 a left outer join table2 b on a.ID = b.ID;
  ```

- Cause

  In the preceding SQL statement, a left outer join is performed on the ID fields of table1 and table2. Therefore, the following situations may occur:

  - If the data that you want to join cannot be found in table2, table1 still returns a data record.
  - If the data that you want to join cannot be found in table1 but can be found in table2, no data is returned.
  - If the data that you want to join can be found in table1 and table2, the join logic is the same as inner joins. If an ID has values in both table1 and table2, the returned result is the Cartesian product of table1 and table2.

  The following table provides sample data in table1.

id	values
1	a

id	values
1	b
2	c

The following table provides sample data in table2.

id	values
1	A
1	B
3	D

The following table lists the returned results after `select count(*) from table1 a left outer join table2 b on a.ID = b.ID;` is executed.

id1	values1	id2	values2
1	b	1	B
1	b	1	A
1	a	1	B
1	a	1	A
2	c	NULL	NULL

- Both tables have data whose value of the ID field is 1. Therefore, the Cartesian product operation is performed and four data records are returned.
- Only table1 has data whose value of the ID field is 2. Therefore, one data record is returned.
- Only table2 has data whose value of the ID field is 3. Therefore, no data is returned.

- Solution

  Check whether the increase in the number of data records is caused by the data in table2. The following statement shows an example. In this statement, `limit 10` is added to prevent data from flooding your screen if table2 has a large amount of data. You need only to check the first few data records to determine the reason for the increase in the number of data records.

  ```
  select id, count() as cnt from table2 group by id having cnt>1 limit 10;
  ```

  If you do not want to perform the Cartesian product operation when duplicate data exists and you want to achieve the effect that is equivalent to IN in SQL, you can use the following statement:

  ```
  select * from table1 a left outer join (select distinct id from table2) b on a.id = b.id;
  ```

I specify the partition condition when I perform a JOIN operation, but the system prompts that full table scan is prohibited. Why?

- Problem description

  When I execute the following statement in two projects, the statement is successfully executed only in one of the projects.

  ```
  select t.stat_date
  from fddev.tmp_001 t
  left outer join (select '20180830' as ds from fddev.dual ) t1
  on t.ds = 20180830
  group by t.stat_date;
  ```

  The following error message appears:

  ```
  Table(fddev,tmp_001) is full scan with all partitions,please specify partition predicates
  .
  ```

- Cause

  To specify a partition in a SELECT statement, you must use a WHERE clause. The ON condition in your SELECT statement does not comply with the SQL standard.

  The `set odps.sql.outerjoin.supports.filters=true` command is run in the project in which the statement is successfully executed. This configuration converts the condition in the ON clause into a filter condition to allow non-standard SQL statements. This configuration is compatible with the HIVE syntax but does not comply with the SQL standard.

- Solution

  We recommend that you place the partition filter conditions in a WHERE clause.

When I perform a JOIN operation, does partition pruning take effect if the partition pruning condition is specified in an ON clause or a WHERE clause?

Partition pruning takes effect if the partition pruning condition is specified in a WHERE clause. If the partition pruning condition is specified in an ON clause, partition pruning takes effect on the secondary table. Partition pruning does not take effect on the primary table. Therefore, a full table scan is triggered. For more information about partition pruning, see Check whether partition pruning is effective.

How do I use MAPJOIN to cache multiple small tables?

You can specify the aliases of the tables that you want to cache in the MAPJOIN statement.

For example, a table named iris exists in a project. The table has the following data:

```
+------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                     |
+------------------------------------------------------------------------------------+
| sepal_length    | double     |       |                                             |
| sepal_width     | double     |       |                                             |
| petal_length    | double     |       |                                             |
| petal_width     | double     |       |                                             |
| category        | string     |       |                                             |
+------------------------------------------------------------------------------------+
```

The following sample code shows how to use MAPJOIN to cache small tables.

```
select
  /*+ mapjoin(b,c) */
  a.category,
  b.cnt as cnt_category,
  c.cnt as cnt_all
from iris a
join
(
  select count() as cnt,category from iris group by category
) b
on a.category = b.category
join
(
  select count(*) as cnt from iris
) c;
```

Can I exchange the large table and small tables that are specified in the MAPJOIN statement?

The large table and small tables in the MAPJOIN statement are distinguished based on the size of the space that is used by each table.

The system loads all data in the specified small tables to the memory of the program that performs the JOIN operation. This helps accelerate the execution of the JOIN operation. If you exchange the large table and small tables in the MAPJOIN statement, no error is returned but the processing performance decreases.

After I configure filter conditions in a MaxCompute SQL statement, an error message indicating that the size of the input data exceeds 100 GB appears. What do I do?

Filter partitions before you obtain data. After you obtain data, filter non-partition fields. The size of the input table varies based on the size of the table after the partitions are filtered and before the non-partition fields are filtered.

Does the WHERE condition of a fuzzy query in MaxCompute SQL support regular expressions?

Yes, the WHERE condition of a fuzzy query in MaxCompute SQL supports regular expressions. For example, `select * from user_info where address rlike '[0-9]{9}';` indicates that the IDs that consist of nine digits are searched.

If I want to synchronize only 100 data records, how do I use LIMIT to specify the number of data records that I want to synchronize in a WHERE clause?

LIMIT cannot be used in a WHERE clause. You can execute an SQL statement to read 100 data records before you synchronize data.

How can I improve the query efficiency? Can I adjust the partition settings?

If you use partition fields to partition a table, a full table scan is not triggered when partitions are added or when partition data is updated or read. This improves the efficiency of data processing. For more information, see Table operations and MaxCompute partition configuration and usage.

Does MaxCompute SQL support the WITH AS statement?

Yes, MaxCompute SQL supports the WITH AS statement. MaxCompute supports SQL-compliant common table expressions (CTEs) to improve the readability and execution efficiency of SQL statements. For more information, see Common table expressions.

How do I split one row of data into multiple rows?

You can use Lateral View with table generation functions, such as SPLIT and EXPLODE, to split one row of data into multiple rows of data and aggregate the split data.

After I specify use_instance_tunnel=false and instance_tunnel_max_record=10 in the odps_config.ini file of the client, the SELECT statement still generates a large number of output records. Why?

To use `instance_tunnel_max_record` to control the number of output records, you must change `use_instance_tunnel=false` to `use_instance_tunnel=true`.

How do I use a regular expression to determine whether the values of a field are in Chinese?

The following statement shows an example.

```
select 'Field name' rlike '[\\x{4e00}-\\x{9fa5}]+';
```

3.8.4. Other FAQ about SQL

This topic provides answers to some frequently asked questions about SQL statements in MaxCompute, such as questions related data types and limits on SQL.

Category	FAQ

Category	FAQ
Data types	• Does MaxCompute support time fields whose values do not include the hour, minute, and second parts? • Why does the result of equivalent comparison between values of the DOUBLE type during the execution of MaxCompute SQL statements fail to meet the expectation? • How do I resolve the precision overflow issue of values of the DECIMAL data type? • What do I do if the newly created MaxCompute project does not support implicit conversions between data types? • What do I do if the "XXXtypeisnotenabled incurrentmode" error message appears when I query data?
Limits on SQL	• What do I do if the size of a field exceeds 8 MB? • What do I do if the "partitions exceeds the specified limit" error message appears when I execute a MaxCompute SQL statement?
Operations to run SQL jobs	• How do I execute MaxCompute SQL in non-interactive mode? • Can I transfer MaxCompute configurations from an Alibaba Cloud account to another Alibaba Cloud account by using SQL statements? • How do I synchronize data from a table in the development environment to a table in the production environment? • How do I determine whether a field is empty? • Can I use Shell nodes of DataWorks to call MaxCompute SQL? • How do I run loops in SQL? • How do I call assignment nodes in SQL?
Operations to view SQL jobs	• How do I view all SQL statements that are executed on a daily basis in a MaxCompute project? • How do I view the amount of resources consumed by an SQL job? • How do I estimate the cost of executing an SQL job?
Operations to tune SQL jobs	How do I improve the efficiency of job operation when SQL jobs run slowly?

Does MaxCompute support time fields whose values do not include the hour, minute, and second parts?

Yes, MaxCompute supports time fields whose values do not include the hour, minute, and second parts. The time fields can be of the DATE data type. To use the DATE data type, you must enable the MaxCompute V2.0 data type edition. For more information about the MaxCompute V2.0 data type edition, see MaxCompute V2.0 data type edition.

Why does the result of equivalent comparison between values of the DOUBLE type during the execution of MaxCompute SQL statements fail to meet the expectation?

The values of the DOUBLE type in MaxCompute have different precisions. We recommend that you do not use an equal sign (=) to compare two values of the DOUBLE type.

To resolve this issue, subtract the two values of the DOUBLE type and take the absolute value. If the absolute value is extremely small, the two values of the DOUBLE type are considered equal.

How do I resolve the precision overflow issue of values of the DECIMAL data type?

You can run the `set odps.sql.decimal.odps2=true;` command at the session level to enable the MaxCompute V2.0 data type edition. The maximum length supported by MaxCompute for the DECIMAL data type is 38 bits. However, if data is stored based on the maximum length during business processing, data overflow may occur. To avoid data overflow, we recommend that you decrease the maximum length of the DECIMAL data type.

What do I do if the newly created MaxCompute project does not support implicit conversions between data types?

To support implicit data type conversions, you must disable the MaxCompute V2.0 data type edition. For more information about data type conversions, see Type conversions.

What do I do if the "XXXtypeisnotenabled incurrentmode" error message appears when I query data?

You can run the `set odps.sql.decimal.odps2=true;` command at the session level to enable the MaxCompute V2.0 data type edition.

What do I do if the size of a field exceeds 8 MB?

The size of a single field in a MaxCompute table cannot exceed 8 MB due to the storage limit. If the size of a field is greater than 8 MB, we recommend that you split the field into multiple fields. You can design the specific splitting logic based on the characteristics of your business to ensure that the size of each field does not exceed 8 MB.

Large fields with complex structures significantly affect the computing performance during data development and analysis. We recommend that you design your data architecture based on the following data warehouse specifications to avoid large fields:

- When you archive raw data with a complex structure at an operational data store (ODS) layer, compress the data.
- Perform data cleansing on the incremental data at the ODS layer on a regular basis, such as every day. Split complex fields into multiple simple fields and store the fields in tables at the common data model (CDM) layer for easy data collection and analysis of data.

What do I do if the "partitions exceeds the specified limit" error message appears when I execute a MaxCompute SQL statement?

- Problem description

  When I execute a MaxCompute SQL statement, the following error message appears:

  ```
  FAILED: ODPS-0010000:System internal error - OTS filtering exception - Ots read range par
  titions exceeds the specified limit:10000:  tableName:xxxx , please check hive conf key
  ```

- Cause

A MaxCompute table can contain up to 60,000 partitions. However, you can query only up to 10,000 partitions at the same time. Common causes:

- Cause 1: No partition condition is specified.
- Cause 2: A field similar to user ID is used as the partition field. As a result, excessive partitions are generated.

- Solutions
  - Solution to Cause 1: Specify a partition condition.
  - Solution to Cause 2: Change the partition field.

How do I execute MaxCompute SQL in non-interactive mode?

In the operating system, you can execute MaxCompute SQL in non-interactive mode by using shell commands.

- Use `odps -f filename` to read and process SQL files.

  The first row of the file specified by the filename parameter is SQL, which indicates that the `SQL` mode is enabled. Sample statement:

  ```
  SQL
    select ... from table_name where xxx;
  ```

- If you execute only one SQL statement, you can use the `sqltext` method in MaxCompute SQL. Sample statement:

  ```
  ./odpscmd -e "select * from dual;"
  ```

  You can use `odps -help` to obtain more information.

Can I transfer MaxCompute configurations from an Alibaba Cloud account to another Alibaba Cloud account by using SQL statements?

No, you cannot transfer MaxCompute configurations from an Alibaba Cloud account to another Alibaba Cloud account by using SQL statements. If you want to transfer MaxCompute configurations from an Alibaba Cloud account to another Alibaba Cloud account, you can use package authorization. For more information about package authorization, see Best practices of MaxCompute multi-team data development project management.

How do I synchronize data from a table in the development environment to a table in the production environment?

Run the following command on the MaxCompute client:

```
insert into project.table select * from project_dev.table;
```

If you do not have read and write permissions on tables in the production environment, you must complete account authorization. For more information about authorization, see Permissions.

How do I determine whether a field is empty?

You can use the operators of MaxCompute SQL to determine whether a field is empty. For more information about the operators of MaxCompute SQL, see Operators.

Can I use Shell nodes of DataWorks to call MaxCompute SQL?

No, you cannot use Shell nodes of DataWorks to call MaxCompute SQL. Shell nodes support only standard shell syntax. Shell nodes do not support interactive syntax. If a large number of jobs exist, you can use ODPS SQL nodes to run jobs. For more information about ODPS SQL nodes, see Create an ODPS SQL node.

How do I run loops in SQL?

You can use do-while nodes of DataWorks to run loops in SQL.

How do I call assignment nodes in SQL?

You can use for-each nodes of DataWorks to call assignment nodes in SQL.

How do I view all SQL statements that are executed on a daily basis in a MaxCompute project?

You can run the `show p -all -limit <number>;` command to view the historical SQL information of all members in the MaxCompute project. For more information about how to view instance information, see View instance information.

How do I view the amount of resources consumed by an SQL job?

You can view the amount of resources consumed by SQL jobs in your bills. For more information about how to analyze MaxCompute bills, see Analyze MaxCompute bills.

How do I estimate the cost of executing an SQL job?

You can run the `cost sql` command to estimate the cost of executing an SQL job. For more information about the cost sql command, see Cost estimation.

How do I improve the efficiency of job operation when SQL jobs run slowly?

You can use Logview to diagnose SQL jobs that run slowly. For more information, see Use Logview to diagnose jobs that run slowly.

For more information about how to optimize SQL jobs, see Best practices for computing optimization.

3.9. Built-in functions

3.9.1. Overview

MaxCompute provides a large number of built-in functions to meet data processing requirements in most business scenarios. This topic describes the types of built-in functions that are provided by MaxCompute. This topic also describes how to use the built-in functions.

Background information

The following table describes the types of built-in functions that are provided by MaxCompute.

Type	Description
Date functions	Used to process data of a date type, such as DATE, DATETIME, or TIMESTAMP. For example, you can use these functions to add and subtract date values, calculate date value differences, extract date fields, obtain the current time, and convert date formats.
Mathematical functions	Used to process data of a numeric type, such as BIGINT, DOUBLE, DECIMAL, or FLOAT. For example, you can use these functions to convert numeral systems, perform mathematical operations, round values, and obtain random numbers.
Window functions	Used to process the data of columns in a window. For example, you can use these functions to calculate the sum, maximum value, minimum value, average value, and median value of column data, sort column data, obtain the data of columns at a given offset, and sample column data.
Aggregate functions	Used to aggregate multiple input records into an output value. For example, you can use these functions to calculate the sum, maximum value, minimum value, and average value of data, aggregate parameters, and concatenate strings.
String functions	Used to process data of the STRING type. For example, you can use these functions to truncate strings, replace strings, search for strings, convert uppercase and lowercase letters, and convert string formats.
Complex type functions	Used to process data of the MAP, ARRAY, STRUCT, or JSON type. For example, you can use these functions to deduplicate, aggregate, sort, and merge elements.
Other functions	Used to process data in other business scenarios.

For more information about the mappings between the built-in functions of MaxCompute and the built-in functions of open source systems, see Mappings between built-in functions of MaxCompute and built-in functions of Hive, MySQL, and Oracle.

Usage notes

When you use built-in functions that are provided by MaxCompute, take note of the following items:

- For a built-in function, the types and number of input parameters and function format must meet the function syntax requirements. If the function syntax requirements are not met, MaxCompute cannot parse the built-in function and an error may occur when you execute the SQL statement in which the built-in function is called.
- If the input parameters of a built-in function are of a type that is supported by the MaxCompute V2.0 data type edition, you must enable the MaxCompute V2.0 data type edition. The data types supported by the MaxCompute V2.0 data type edition include TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY. If you do not enable the MaxCompute V2.0 data type edition, an error may occur when you execute the SQL statement in which the built-in function is called. You can enable the MaxCompute V2.0 data type edition at the session or project level.
  - Session level: Add `set odps.sql.type.system.odps2=true;` before the SQL statement in which a built-in function is called. Then, commit and execute them together. This configuration is valid for only the current SQL statement.

  - Project level: The owner of a project can enable the MaxCompute V2.0 data type edition for the project based on the project requirements. The configuration takes effect after 10 to 15 minutes. This configuration is valid for all the subsequent SQL statements.

    ```
    setproject odps.sql.type.system.odps2=true;
    ```

- If you enable the MaxCompute V2.0 data type edition for a project, some implicit conversions are disabled, such as the conversions from STRING to BIGINT, STRING to DATETIME, DOUBLE to BIGINT, DECIMAL to DOUBLE, and DECIMAL to BIGINT. This may cause a loss of precision or errors. In this case, you can use the CAST function to forcefully convert the data types to resolve these issues. You can also disable the MaxCompute V2.0 data type edition.
- If the name of a UDF is the same as that of a built-in function, the UDF is preferentially called. For example, if UDF CONCAT and built-in function CONCAT both exist in MaxCompute, the system automatically calls UDF CONCAT instead of the built-in function CONCAT. If you want to call the built-in function, you must add the symbol `::` before the built-in function, for example, `select ::concat('ab', 'c');` .
- If the settings of global properties of MaxCompute projects are different, the execution results of built-in functions may be different. You can run the `setproject;` command to configure the global properties of a MaxCompute project.

3.9.2. Mappings between built-in functions of MaxCompute and built-in functions of Hive, MySQL, and Oracle

This topic describes the mappings between the built-in functions of MaxCompute and the built-in functions of Hive, MySQL, and Oracle. This way, you can find the built-in functions of MaxCompute that match specific built-in functions of Hive, MySQL, and Oracle.

Date functions

MaxCompute	Hive	MySQL	Oracle
DATEADD	N/A	N/A	N/A
DATE_ADD	DATE_ADD	DATE_ADD	N/A
DATE_SUB	DATE_SUB	DATE_SUB	N/A
DATEDIFF	DATEDIFF	DATEDIFF	MONTHS_BETWEEN
DATEPART	N/A	DATE_FORMAT	EXTRACT (DATETIME)
DATETRUNC	TRUNC	DATE_FORMAT	EXTRACT (DATETIME)
FROM_UNIXTIME	FROM_UNIXTIME	FROM_UNIXTIME	N/A
GETDATE	CURRENT_DATE	NOW	CURRENT_DATE

MaxCompute	Hive	MySQL	Oracle
ISDATE	N/A	STR_TO_DATE (The return value FALSE indicates that a string cannot be converted into a date value.)	N/A
LASTDAY	LAST_DAY	LAST_DAY	LAST_DAY
TO_DATE	TO_DATE	STR_TO_DATE	DATE
TO_CHAR	N/A	DATE_FORMAT	TO_CHAR (DATETIME)
UNIX_TIMESTAMP	UNIX_TIMESTAMP	UNIX_TIMESTAMP	N/A
WEEKDAY	N/A	WEEKDAY	N/A
WEEKOFYEAR	WEEKOFYEAR	WEEKOFYEAR	N/A
ADD_MONTHS	ADD_MONTHS	ADDDATE	ADD_MONTHS
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	CURRENT_TIMESTAMP
DAY	DAY	DAY	DAY
DAYOFMONTH	DAYOFMONTH	DAYOFMONTH	N/A
EXTRACT	EXTRACT	EXTRACT	EXTRACT
FROM_UTC_TIMESTAMP	FROM_UTC_TIMESTAMP	N/A	N/A
HOUR	HOUR	HOUR	HOUR
LAST_DAY	LAST_DAY	LAST_DAY	N/A
MINUTE	MINUTE	MINUTE	MINUTE
MONTH	MONTH	MONTH	MONTH
MONTHS_BETWEEN	MONTHS_BETWEEN	TIMESTAMPDIFF	MONTHS_BETWEEN
NEXT_DAY	NEXT_DAY	N/A	NEXT_DAY
QUARTER	QUARTER	QUARTER	QUARTER
SECOND	SECOND	SECOND	N/A
TO_MILLIS	N/A	N/A	N/A
YEAR	YEAR	YEAR	N/A

Note The MaxCompute mode is enabled by default. To use the Hive-compatible mode, run one of the following commands:

```
-- Switch to the Hive-compatible mode at the project level.
setproject odps.sql.hive.compatible=True;
-- Switch to the Hive-compatible mode at the session level.
set odps.sql.hive.compatible=True;
```

Mathematical functions

MaxCompute	Hive	MySQL	Oracle
ABS	ABS	ABS	ABS
ACOS	ACOS	ACOS	ACOS
ASIN	ASIN	ASIN	ASIN
ATAN	ATAN	ATAN	ATAN
CEIL	CEIL	CEIL	CEIL
CONV	CONV	CONV	N/A
COS	COS	COS	COS
COSH	COSH	N/A	COSH
COT	COT	COT	COT
EXP	EXP	EXP	EXP
FLOOR	FLOOR	FLOOR	FLOOR
LN	LN	LN	LN
LOG	LOG	LOG	LOG
POW	POW	POW	POWER
RAND	RAND	RAND	N/A
ROUND	ROUND	ROUND	ROUND
SIN	SIN	SIN	SIN
SINH	SINH	N/A	SINH
SQRT	SQRT	SQRT	SQRT
TAN	TAN	TAN	TAN

MaxCompute	Hive	MySQL	Oracle
TANH	TANH	N/A	TANH
TRUNC	TRUNC	TRUNCATE	TRUNC
BIN	BIN	BIN	BITAND
CBRT	CBRT	N/A	N/A
CORR	CORR	CORR	CORR
DEGREES	DEGREES	DEGREES	DEGREES
E	E	N/A	N/A
FACTORIAL	FACTORIAL	N/A	N/A
FORMAT_NUMBER	FORMAT_NUMBER	FORMAT	N/A
HEX	HEX	HEX	RAWTOHEX
LOG2	LOG2	LOG2	LOG
LOG10	LOG10	LOG10	LOG
PI	PI	PI	PI
RADIANS	RADIANS	RADIANS	RADIANS
SIGN	SIGN	SIGN	SIGN
SHIFTLEFT	SHIFTLEFT	<<	N/A
SHIFTRIGHT	SHIFTRIGHT	>>	N/A
SHIFTRIGHTUNSIGNED	SHIFTRIGHTUNSIGNED	>>>	N/A
UNHEX	UNHEX	UNHEX	HEXTORAW
WIDTH_BUCKET	WIDTH_BUCKET	N/A	WIDTH_BUCKET

Note The MaxCompute mode is enabled by default. To use the Hive-compatible mode, run one of the following commands:

```
-- Switch to the Hive-compatible mode at the project level.
setproject odps.sql.hive.compatible=True;
-- Switch to the Hive-compatible mode at the session level.
set odps.sql.hive.compatible=True;
```

Window functions

MaxCompute	Hive	MySQL	Oracle
COUNT	COUNT	COUNT	COUNT
Window functions	AVG	AVG	AVG
MAX	MAX	MAX	MAX
MIN	MIN	MIN	MIN
Window functions	N/A	N/A	MEDIAN
STDDEV	N/A	STDDEV	STDDEV
STDDEV_SAMP	N/A	STDDEV_SAMP	STDDEV_SAMP
SUM	SUM	SUM	SUM
Window functions	DENSE_RANK	DENSE_RANK	DENSE_RANK
Window functions	RANK	RANK	RANK
Window functions	LAG	LAG	LAG
Window functions	LEAD	LEAD	LEAD
Window functions	PERCENT_RANK	PERCENT_RANK	PERCENT_RANK
Window functions	ROW_NUMBER	ROW_NUMBER	ROW_NUMBER
Window functions	N/A	N/A	N/A
Window functions	CUME_DIST	CUME_DIST	CUME_DIST
Window functions	NTILE	NTILE	NTILE

Aggregate functions

MaxCompute	Hive	MySQL	Oracle
AVG	AVG	AVG	AVG
COUNT	COUNT	COUNT	COUNT
COUNT_IF	N/A	N/A	N/A
MAX	MAX	MAX	MAX
MIN	MIN	MIN	MIN
MEDIAN	N/A	N/A	MEDIAN
STDDEV	STDDEV	STDDEV	STDDEV

MaxCompute	Hive	MySQL	Oracle
STDDEV_SAMP	STDDEV_SAMP	STDDEV_SAMP	STDDEV_SAMP
SUM	SUM	SUM	SUM
WM_CONCAT	N/A	GROUP_CONCAT	WM_CONCAT
ANY_VALUE	N/A	N/A	N/A
APPROX_DISTINCT	N/A	N/A	N/A
ARG_MAX	N/A	N/A	N/A
ARG_MIN	N/A	N/A	N/A
COLLECT_LIST	COLLECT LIST	N/A	COLLECT
COLLECT_SET	COLLECT SET	N/A	COLLECT
COVAR_POP	COVAR_POP	N/A	COVAR_POP
COVAR_SAMP	COVAR_SAMP	N/A	COVAR_SAMP
NUMERIC_HISTOGRAM	NUMERIC_HISTOGRAM	N/A	N/A
PERCENTILE	PERCENTILE	N/A	N/A
PERCENTILE_APPROX	PERCENTILE_APPROX	N/A	N/A
VARIANCE/VAR_POP	VARIANCE/VAR_POP	VAR_POP	VARIANCE/VAR_POP
VAR_SAMP	VAR_SAMP	VAR_SAMP	VAR_SAMP

Note The MaxCompute mode is enabled by default. To use the Hive-compatible mode, run one of the following commands:

```
-- Switch to the Hive-compatible mode at the project level.
setproject odps.sql.hive.compatible=True;
-- Switch to the Hive-compatible mode at the session level.
set odps.sql.hive.compatible=True;
```

String functions

MaxCompute	Hive	MySQL	Oracle
ASCII	ASCII	ASCII	ASCII
CHAR_MATCHCOUNT	N/A	N/A	N/A
CHR	CHR	CHAR	CHR

MaxCompute	Hive	MySQL	Oracle
CONCAT	CONCAT	CONCAT	CONCAT
ENCODE	ENCODE	N/A	N/A
FIND_IN_SET	FIND_IN_SET	FIND_IN_SET	N/A
FORMAT_NUMBER	FORMAT_NUMBER	FORMAT	N/A
FROM_JSON	N/A	N/A	N/A
GET_JSON_OBJECT	GET_JSON_OBJECT	JSON_EXTRACT	N/A
INSTR	INSTR	INSTR	INSTR
IS_ENCODING	N/A	N/A	N/A
KEYVALUE	N/A	N/A	N/A
LENGTH	LENGTH	LENGTH	LENGTH
LENGTHB	LENGTHB	LENGTHB	LENGTHB
LOCATE	LOCATE	LOCATE	N/A
LTRIM	LTRIM	LTRIM	LTRIM
MD5	MD5	MD5	N/A
PARSE_URL	PARSE_URL	N/A	N/A
PARSE_URL_TUPLE	PARSE_URL_TUPLE	N/A	N/A
REGEXP_COUNT	N/A	N/A	REGEXP_COUNT
REGEXP_EXTRACT	REGEXP_EXTRACT	N/A	N/A
REGEXP_INSTR	N/A	REGEXP_INSTR	REGEXP_INSTR
REGEXP_REPLACE	REGEXP_REPLACE	REGEXP_REPLACE	REGEXP_REPLACE
REGEXP_SUBSTR	N/A	REGEXP_SUBSTR	REGEXP_SUBSTR
REPEAT	REPEAT	REPEAT	REPEAT
REVERSE	REVERSE	REVERSE	REVERSE
RTRIM	RTRIM	RTRIM	RTRIM
SPACE	SPACE	SPACE	SPACE
SPLIT_PART	N/A	N/A	N/A

MaxCompute	Hive	MySQL	Oracle
SUBSTR	SUBSTR	SUBSTR	SUBSTR
SUBSTRING	SUBSTRING	SUBSTRING	SUBSTR
TO_CHAR	N/A	N/A	N/A
TO_JSON	N/A	N/A	N/A
TOLOWER	LOWER	LOWER	LOWER
TOUPPER	UPPER	UPPER	UPPER
TRIM	TRIM	TRIM	TRIM
URL_DECODE	N/A	N/A	N/A
URL_ENCODE	N/A	N/A	PERCENTILE_CONT
CONCAT_WS	CONCAT_WS	CONCAT_WS	N/A
JSON_TUPLE	JSON_TUPLE	N/A	N/A
LPAD	LPAD	LPAD	LPAD
RPAD	RPAD	RPAD	RPAD
REPLACE	REPLACE	REPLACE	REPLACE
SOUNDEX	SOUNDEX	SOUNDEX	SOUNDEX
SUBSTRING_INDEX	SUBSTRING_INDEX	SUBSTRING_INDEX	N/A
TRANSLATE	TRANSLATE	N/A	TRANSLATE

Note The MaxCompute mode is enabled by default. To use the Hive-compatible mode, run one of the following commands:

```
-- Switch to the Hive-compatible mode at the project level.
setproject odps.sql.hive.compatible=True;
-- Switch to the Hive-compatible mode at the session level.
set odps.sql.hive.compatible=True;
```

Other functions

MaxCompute	Hive	MySQL	Oracle
BASE64	BASE64	TO_BASE64	UTL_ENCODE.BASE64_ENCODE

MaxCompute	Hive	MySQL	Oracle
BETWEEN AND expression	BETWEEN AND	BETWEEN AND	BETWEEN AND
CASE WHEN expression	CASE WHEN	CASE WHEN	CASE WHEN
CAST	CAST	CAST	CAST
COALESCE	COALESCE	COALESCE	COALESCE
COMPRESS	N/A	COMPRESS	UTL_COMPRESS.LZ_COMPRESS
CRC32	CRC32	CRC32	N/A
DECODE	DECODE	N/A	DECODE
DECOMPRESS	N/A	UNCOMPRESS	UTL_COMPRESS.LZ_UNCOMPRESS
GET_IDCARD_AGE	N/A	N/A	N/A
GET_IDCARD_BIRTHDAY	N/A	N/A	N/A
GET_IDCARD_SEX	N/A	N/A	N/A
GET_USER_ID	CURRENT_USER	CURRENT_USER	UID
GREATEST	GREATEST	GREATEST	N/A
HASH	HASH	N/A	ORA_HASH
IF	IF	IF	IF
LEAST	LEAST	LEAST	LEAST
MAX_PT	N/A	N/A	N/A
NULLIF	NULLIF	NULLIF	NULLIF
NVL	NVL	IFNULL	N/A
ORDINAL	N/A	N/A	N/A
PARTITION_EXISTS	N/A	N/A	N/A
SAMPLE	N/A	N/A	N/A
SHA	SHA	SHA	N/A
SHA1	SHA1	SHA1	N/A
SHA2	SHA2	SHA2	N/A

MaxCompute	Hive	MySQL	Oracle
SIGN	SIGN	SIGN	SIGN
SPLIT	SPLIT	SPLIT	N/A
STACK	STACK	N/A	N/A
STR_TO_MAP	STR_TO_MAP	N/A	N/A
TABLE_EXISTS	N/A	N/A	N/A
TRANS_ARRAY	N/A	N/A	N/A
TRANS_COLS	N/A	N/A	N/A
UNBASE64	UNBASE64	FROM_BASE64	UTL_ENCODE.BASE64_DECODE
UNIQUE_ID	N/A	N/A	N/A
UUID	N/A	UUID	UID

Note The MaxCompute mode is enabled by default. To use the Hive-compatible mode, run one of the following commands:

```
-- Switch to the Hive-compatible mode at the project level.
setproject odps.sql.hive.compatible=True;
-- Switch to the Hive-compatible mode at the session level.
set odps.sql.hive.compatible=True;
```

Complex type functions

Function type	MaxCompute	Hive	MySQL	Oracle
	ALL_MATCH	N/A	N/A	N/A
	ANY_MATCH	N/A	N/A	N/A
	ARRAY	ARRAY	N/A	N/A
	ARRAY_CONTAINS	ARRAY_CONTAINS	N/A	N/A
	ARRAY_DISTINCT	N/A	N/A	N/A
	ARRAY_EXCEPT	N/A	N/A	N/A
	ARRAY_INTERSECT	N/A	N/A	N/A
	ARRAY_JOIN	N/A	N/A	N/A

Function type	MaxCompute	Hive	MySQL	Oracle
ARRAY	ARRAY_MAX	N/A	N/A	N/A
	ARRAY_MIN	N/A	N/A	N/A
	ARRAY_POSITION	N/A	N/A	N/A
	ARRAY_REMOVE	N/A	N/A	N/A
	ARRAY_REDUCE	N/A	N/A	N/A
	ARRAY_REPEAT	N/A	N/A	N/A
	ARRAY_SORT	N/A	N/A	N/A
	ARRAY_UNION	N/A	N/A	N/A
	ARRAYS_OVERLAP	N/A	N/A	N/A
	ARRAYS_ZIP	N/A	N/A	N/A
	CONCAT	CONCAT	N/A	N/A
	EXPLODE	EXPLODE	N/A	N/A
	FILTER	N/A	N/A	N/A
	INDEX	[] operator	N/A	N/A
	POSEXPLODE	POSEXPLODE	N/A	N/A
	SIZE	SIZE	N/A	N/A
	SLICE	N/A	N/A	N/A
	SORT_ARRAY	SORT_ARRAY	N/A	N/A
	TRANSFORM	N/A	N/A	N/A
	ZIP_WITH	N/A	N/A	N/A
	EXPLODE	EXPLODE	N/A	N/A
	INDEX	[] operator	N/A	N/A
	MAP	MAP	N/A	N/A
	MAP_CONCAT	N/A	N/A	N/A
	MAP_ENTRIES	N/A	N/A	N/A
	MAP_FILTER	N/A	N/A	N/A

Function type	MaxCompute	Hive	MySQL	Oracle
MAP	MAP_FROM_ARRAYS	N/A	N/A	N/A
	MAP_FROM_ENTRIES	N/A	N/A	N/A
	MAP_KEYS	MAP_KEYS	N/A	N/A
	MAP_VALUES	MAP_VALUES	N/A	N/A
	MAP_ZIP_WITH	N/A	N/A	N/A
	SIZE	SIZE	N/A	N/A
	TRANSFORM_KEYS	N/A	N/A	N/A
	TRANSFORM_VALUES	N/A	N/A	N/A
STRUCT	FIELD	. operator	N/A	N/A
	INLINE	INLINE	N/A	N/A
	STRUCT	STRUCT	N/A	N/A
	NAMED_STRUCT	N/A	N/A	N/A
JSON	FROM_JSON	N/A	N/A	N/A
	GET_JSON_OBJECT	GET_JSON_OBJECT	JSON_EXTRACT	N/A
	JSON_TUPLE	JSON_TUPLE	N/A	N/A
	TO_JSON	N/A	N/A	N/A

Note The MaxCompute mode is enabled by default. To use the Hive-compatible mode, run one of the following commands:

```
-- Switch to the Hive-compatible mode at the project level.
setproject odps.sql.hive.compatible=True;
-- Switch to the Hive-compatible mode at the session level.
set odps.sql.hive.compatible=True;
```

3.9.3. Date functions

MaxCompute SQL provides common date functions. You can select an appropriate date function based on your business requirements to complete date calculations and conversions. This topic describes the syntax and parameters of the date functions that are supported by MaxCompute SQL. This topic also provides examples on how to use date functions to develop data.

Function	Description
DATEADD	Changes a date value based on the time unit specified by datepart and the interval specified by delta.
DATE_ADD	Adds or subtracts a number of days to or from a date value based on the interval specified by delta. The DATE_ADD function is the inverse of the DATE_SUB function.
DATE_FORMAT	Converts a date value into a string in a specified format.
DATE_SUB	Adds or subtracts a number of days to or from a date value based on the interval specified by delta. The DATE_SUB function is the inverse of the `DATE_ADD` function.
DATEDIFF	Calculates the difference between two date values based on the time unit specified by datepart.
DATEPART	Returns a specified component of a date value based on the time unit specified by datepart.
DATETRUNC	Truncates a date value based on the time unit specified by datepart.
FROM_UNIXTIME	Converts a UNIX timestamp of the BIGINT type into a date value of the DATETIME type.
GETDATE	Returns the current system time as a date value.
ISDATE	Determines whether a date string can be converted into a date value in a specified format.
LASTDAY	Returns the last day of the month in which a date value falls.
TO_DATE	Converts a string into a date value in a specified format.
TO_CHAR	Converts a date value into a string in a specified format.
UNIX_TIMESTAMP	Converts a date value into a UNIX timestamp that is an integer.
WEEKDAY	Returns a number that represents the day of the week in which a date value falls.
WEEKOFYEAR	Returns a number that represents the week of the year in which a date value falls.
ADD_MONTHS	Returns a date value that is obtained after a number of months are added to a specified date.
CURRENT_TIMESTAMP	Returns the current timestamp.
DAY	Returns the day in which a date value falls.
DAYOFMONTH	Returns the day component of a date value.
EXTRACT	Returns a specified component of a timestamp.

Function	Description
FROM_UTC_TIMESTAMP	Converts a UTC timestamp into a timestamp for a specified time zone.
HOUR	Returns the hour component of a date value.
LAST_DAY	Returns the last day of the month in which a date value falls.
MINUTE	Returns the minute component of a date value.
MONTH	Returns the month in which a date value falls.
MONTHS_BETWEEN	Returns the number of months between specified date values.
NEXT_DAY	Returns the date of the first weekday that is later than a date value.
QUARTER	Returns the quarter in which a date value falls.
SECOND	Returns the second component of a date value.
TO_MILLIS	Converts a date value into a UNIX timestamp that is accurate to the millisecond.
YEAR	Returns the year in which a date value falls.

Usage notes

MaxCompute V2.0 provides additional date functions. If the functions that you use involve new data types, you must run one of the following SET commands to enable the MaxCompute V2.0 data type edition. The new data types include TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY.

- Session level: To use the MaxCompute V2.0 data type edition, you must add `set odps.sql.type.system.odps2=true;` before the SQL statement that you want to execute, and commit and execute them together.
- Project level: The project owner can run the following command to enable the MaxCompute V2.0 data type edition for the project based on the project requirements. The configuration takes effect after 10 to 15 minutes.

  ```
  setproject odps.sql.type.system.odps2=true;
  ```

  For more information about `setproject`, see Project operations. For more information about the precautions that you must take when you enable the MaxCompute V2.0 data type edition at the project level, see Data type editions.

Sample data

This section provides sample source data for you to understand how to use date functions. In this topic, a table named mf_date_fun_t is created and data is inserted into the table. Sample statements:

```
create table if not exists mf_date_fun_t(
    id      int,
    date1   date,
    datetime1   datetime,
    timestamp1 timestamp,
    date2   date,
    datetime2   datetime,
    timestamp2 timestamp,
    date3 string,
    date4 bigint);
insert into mf_date_fun_t values
(1,DATE'2021-11-29',DATETIME'2021-11-29 00:01:00',TIMESTAMP'2021-01-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-20',123456780),
(2,DATE'2021-11-28',DATETIME'2021-11-28 00:02:00',TIMESTAMP'2021-02-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-21',123456781),
(3,DATE'2021-11-27',DATETIME'2021-11-27 00:03:00',TIMESTAMP'2021-03-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-22',123456782),
(4,DATE'2021-11-26',DATETIME'2021-11-26 00:04:00',TIMESTAMP'2021-04-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-23',123456783),
(5,DATE'2021-11-25',DATETIME'2021-11-25 00:05:00',TIMESTAMP'2021-05-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-24',123456784),
(6,DATE'2021-11-24',DATETIME'2021-11-24 00:06:00',TIMESTAMP'2021-06-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-25',123456785),
(7,DATE'2021-11-23',DATETIME'2021-11-23 00:07:00',TIMESTAMP'2021-07-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-26',123456786),
(8,DATE'2021-11-22',DATETIME'2021-11-22 00:08:00',TIMESTAMP'2021-08-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-27',123456787),
(9,DATE'2021-11-21',DATETIME'2021-11-21 00:09:00',TIMESTAMP'2021-09-11 00:00:00.123456789',
DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','20
21-11-28',123456788),
(10,DATE'2021-11-20',DATETIME'2021-11-20 00:10:00',TIMESTAMP'2021-10-11 00:00:00.123456789'
,DATE'2021-10-29',DATETIME'2021-10-29 00:00:00',TIMESTAMP'2021-10-11 00:00:00.123456789','2
021-11-29',123456789);
```

DATEADD

- Syntax

```
date|datetime dateadd(date|datetime|timestamp <date>, bigint <delta>, string <datepart>)
```

- Description

  Changes a date value based on the time unit specified by datepart and the interval specified by delta. To add or subtract an interval to or from the current time, you can use this function with the GETDATE function.

- Parameters
  - date: required. A date value of the DATE, DATETIME, or TIMESTAMP type.

    A value of the STRING type is implicitly converted into a value of the DATETIME type before calculation if the value format conforms to the DATETIME-type format `yyyy-mm-dd hh:mi:ss`, such as `2021-08-28 00:00:00`, and the MaxCompute V1.0 data type edition is used.
  - delta: required. The interval that you want to add to or subtract from the specified component of a date value. The value of this parameter must be of the BIGINT type. If the value of delta is greater than 0, the date value is incremented. Otherwise, the date value is decremented.

    If the input value is of the STRING or DOUBLE type, the value is implicitly converted into a value of the BIGINT type before calculation.

    > **Note**
    > - If you add or subtract the interval specified by delta based on the time unit specified by datepart, a carry or return at more significant date components may occur. The year, month, hour, minute, and second components are calculated by using different numeral systems. The year component uses the base-10 numeral system. The month component uses the base-12 numeral system. The hour component uses the base-24 numeral system. The minute and second components use the base-60 numeral system.
    > - If the DATEADD function adds the interval specified by delta to the month component of a date value and this operation does not cause an overflow on the day component, you can retain the value of the day component. Otherwise, you must set the value of the day component to the last day of the specified month.
  - datepart: required. The date part that you want to modify in the date value. The value is a constant of the STRING type. If the format of the input value is invalid or the input value is not a constant of the STRING type, an error is returned.

    The value of this parameter is specified in compliance with the rules of conversions between the STRING and DATETIME types. The value `yyyy` indicates that the DATEADD function adds an interval to the year component of the date value. The value `mm` indicates that the DATEADD function adds an interval to the month component of the date value. The value `dd` indicates that the DATEADD function adds an interval to the day component of the date value. For more information about the rules for type conversions, see Type conversions. The Extended Date/Time Format (EDTF) is also supported, such as `-year`, `-month`, `-mon`, `-day`, or `-hour`.
- Return value

  A value of the DATE or DATETIME type is returned. The return value is in the `yyyy-mm-dd` or `yyyy-mm-dd hh:mi:ss` format. The return value varies based on the following rules:
  - If the value of date is not of the DATE, DATETIME, or TIMESTAMP type, an error is returned.
  - If the value of date is null, an error is returned.
  - If the value of delta or datepart is null, null is returned.
- Examples
  - Examples of static data

- Example 1: common use

```
-- The return value is 2005-03-01 00:00:00. After one day is added, the result is bey
ond the last day of February. The first day of March is returned.
select dateadd(datetime '2005-02-28 00:00:00', 1, 'dd');
-- The return value is 2005-02-27 00:00:00. One day is subtracted.
select dateadd(datetime '2005-02-28 00:00:00', -1, 'dd');
-- The return value is 2006-10-28 00:00:00. After 20 months are added, the month over
flows, and the year value increases by 1.
select dateadd(datetime '2005-02-28 00:00:00', 20, 'mm');
-- The return value is 2005-03-28 00:00:00.
select dateadd(datetime '2005-02-28 00:00:00', 1, 'mm');
-- The return value is 2005-02-28 00:00:00. February in 2005 has only 28 days. Theref
ore, the last day of February is returned.
select dateadd(datetime '2005-01-29 00:00:00', 1, 'mm');
-- The return value is 2005-02-28 00:00:00.
select dateadd(datetime '2005-03-30 00:00:00', -1, 'mm');
-- The return value is 2005-03-18.
select dateadd(date '2005-02-18', 1, 'mm');
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command tog
ether with the SQL statement. The return value is 2005-03-18 00:00:00.0.
set odps.sql.type.system.odps2=true;
select dateadd(timestamp '2005-02-18 00:00:00', 1, 'mm');
-- If the current time is 2020-11-17 16:31:44, the return value is 2020-11-16 16:31:4
4.
select dateadd(getdate(),-1,'dd');
-- The value null is returned.
select dateadd(date '2005-02-18', 1, null);
```

- Example 2: use of DATEADD in which a value of the DATETIME type is expressed as a constant

  In MaxCompute SQL statements, a value of the DATETIME type cannot be directly expressed as a constant. The following statement uses an **invalid expression** of a value of the DATETIME type:

```
select dateadd(2005-03-30 00:00:00, -1, 'mm');
```

  To describe a constant of the DATETIME type, use a **valid expression** of a value of the DATETIME type in the following statement:

```
-- Explicitly convert a constant of the STRING type into the DATETIME type. The retur
n value is 2005-02-28 00:00:00.
select dateadd(cast("2005-03-30 00:00:00" as datetime), -1, 'mm');
```

- Example 3: The input value is of the STRING type.

```
-- The input value is of the STRING type but does not conform to the DATETIME-type fo
rmat. As a result, an error is returned.
select dateadd('2021-08-27',1,'dd');
-- The input value is of the STRING type and conforms to the DATETIME-type format, an
d the MaxCompute V1.0 data type edition is used in your project. The return value is
2005-03-01 00:00:00.
set odps.sql.type.system.odps2=false;
select dateadd('2005-02-28 00:00:00', 1, 'dd');
```

- Examples of table data

  Change date values in the date1, datetime1, and timestamp1 columns based on the time unit specified by datepart and the interval specified by delta. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, dateadd(date1,1,'dd') as date1_dateadd, datetime1, dateadd(datetime1,1,'m
m') as datetime1_dateadd, timestamp1, dateadd(timestamp1,-1,'yyyy') as timestamp1_datea
dd from mf_date_fun_t;
```

  The following result is returned:

```
+------------+---------------+---------------------+---------------------+-------------
-----------------+--------------------------------+
| date1      | date1_dateadd | datetime1           | datetime1_dateadd   | timestamp1
| timestamp1_dateadd            |
+------------+---------------+---------------------+---------------------+-------------
-----------------+--------------------------------+
| 2021-11-29 | 2021-11-30    | 2021-11-29 00:01:00 | 2021-12-29 00:01:00 | 2021-01-11 0
0:00:00.123456789 | 2020-01-11 00:00:00.123456789 |
| 2021-11-28 | 2021-11-29    | 2021-11-28 00:02:00 | 2021-12-28 00:02:00 | 2021-02-11 0
0:00:00.123456789 | 2020-02-11 00:00:00.123456789 |
| 2021-11-27 | 2021-11-28    | 2021-11-27 00:03:00 | 2021-12-27 00:03:00 | 2021-03-11 0
0:00:00.123456789 | 2020-03-11 00:00:00.123456789 |
| 2021-11-26 | 2021-11-27    | 2021-11-26 00:04:00 | 2021-12-26 00:04:00 | 2021-04-11 0
0:00:00.123456789 | 2020-04-11 00:00:00.123456789 |
| 2021-11-25 | 2021-11-26    | 2021-11-25 00:05:00 | 2021-12-25 00:05:00 | 2021-05-11 0
0:00:00.123456789 | 2020-05-11 00:00:00.123456789 |
| 2021-11-24 | 2021-11-25    | 2021-11-24 00:06:00 | 2021-12-24 00:06:00 | 2021-06-11 0
0:00:00.123456789 | 2020-06-11 00:00:00.123456789 |
| 2021-11-23 | 2021-11-24    | 2021-11-23 00:07:00 | 2021-12-23 00:07:00 | 2021-07-11 0
0:00:00.123456789 | 2020-07-11 00:00:00.123456789 |
| 2021-11-22 | 2021-11-23    | 2021-11-22 00:08:00 | 2021-12-22 00:08:00 | 2021-08-11 0
0:00:00.123456789 | 2020-08-11 00:00:00.123456789 |
| 2021-11-21 | 2021-11-22    | 2021-11-21 00:09:00 | 2021-12-21 00:09:00 | 2021-09-11 0
0:00:00.123456789 | 2020-09-11 00:00:00.123456789 |
| 2021-11-20 | 2021-11-21    | 2021-11-20 00:10:00 | 2021-12-20 00:10:00 | 2021-10-11 0
0:00:00.123456789 | 2020-10-11 00:00:00.123456789 |
+------------+---------------+---------------------+---------------------+-------------
-----------------+--------------------------------+
```

DATE_ADD

- Syntax

```
date date_add(date|timestamp|string <startdate>, bigint <delta>)
```

- Description

  Adds or subtracts an interval that is specified by delta to or from a date value that is specified by startdate. To add or subtract an interval to or from the current time, you can use this function with the GETDATE function.

The logic of this function is opposite to that of the DATE_SUB function.

- Parameters
  - startdate: required. The start date. A value of the DATE, DATETIME, or STRING type is supported.

    If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATE type before calculation. The input value of the STRING type must include at least the `'yyyy-mm-dd'` part, such as `'2019-12-27'`.
  - delta: required. The interval that you want to add or subtract. The value of this parameter must be of the BIGINT type. If the value of delta is greater than 0, an interval is **added** to the start date. If the value of delta is less than 0, an interval is **subtracted** from the start date. If the value of delta is 0, the date value remains unchanged.
- Return value

  A value of the DATE type is returned. The return value is in the `yyyy-mm-dd` format. The return value varies based on the following rules:
  - If the value of startdate is not of the DATE, DATETIME, or STRING type, an error is returned.
  - If the value of startdate is null, an error is returned.
  - If the value of delta is null, null is returned.
- Examples
  - Examples of static data

```
-- The return value is 2005-03-01. After one day is added, the result is beyond the las
t day of February. The first day of March is returned.
select date_add(datetime '2005-02-28 00:00:00', 1);
-- The return value is 2005-02-27. One day is subtracted.
select date_add(date '2005-02-28', -1);
-- The return value is 2005-03-20.
set odps.sql.type.system.odps2=false;
select date_add('2005-02-28 00:00:00', 20);
-- If the current time is 2020-11-17 16:31:44, the return value is 2020-11-16.
select date_add(getdate(),-1);
-- The value null is returned.
select date_add('2005-02-28 00:00:00', null);
```

- Examples of table data

  Change date values in the date1, datetime1, and timestamp1 columns based on the interval specified by delta. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, date_add(date1,1) as date1_date_add, datetime1, date_add(datetime1,-1) as
datetime1_date_add, timestamp1, date_add(timestamp1,0) as timestamp1_date_add from mf_d
ate_fun_t;
```

  The following result is returned:

```
+------------+----------------+---------------------+-------------------+--------------
-----------------+--------------------+
| date1      | date1_date_add | datetime1           | datetime1_date_add| timestamp1
| timestamp1_date_add|
+------------+----------------+---------------------+-------------------+--------------
-----------------+--------------------+
| 2021-11-29 | 2021-11-30     | 2021-11-29 00:01:00 | 2021-11-28        | 2021-01-11 00
:00:00.123456789 | 2021-01-11         |
| 2021-11-28 | 2021-11-29     | 2021-11-28 00:02:00 | 2021-11-27        | 2021-02-11 00
:00:00.123456789 | 2021-02-11         |
| 2021-11-27 | 2021-11-28     | 2021-11-27 00:03:00 | 2021-11-26        | 2021-03-11 00
:00:00.123456789 | 2021-03-11         |
| 2021-11-26 | 2021-11-27     | 2021-11-26 00:04:00 | 2021-11-25        | 2021-04-11 00
:00:00.123456789 | 2021-04-11         |
| 2021-11-25 | 2021-11-26     | 2021-11-25 00:05:00 | 2021-11-24        | 2021-05-11 00
:00:00.123456789 | 2021-05-11         |
| 2021-11-24 | 2021-11-25     | 2021-11-24 00:06:00 | 2021-11-23        | 2021-06-11 00
:00:00.123456789 | 2021-06-11         |
| 2021-11-23 | 2021-11-24     | 2021-11-23 00:07:00 | 2021-11-22        | 2021-07-11 00
:00:00.123456789 | 2021-07-11         |
| 2021-11-22 | 2021-11-23     | 2021-11-22 00:08:00 | 2021-11-21        | 2021-08-11 00
:00:00.123456789 | 2021-08-11         |
| 2021-11-21 | 2021-11-22     | 2021-11-21 00:09:00 | 2021-11-20        | 2021-09-11 00
:00:00.123456789 | 2021-09-11         |
| 2021-11-20 | 2021-11-21     | 2021-11-20 00:10:00 | 2021-11-19        | 2021-10-11 00
:00:00.123456789 | 2021-10-11         |
+------------+----------------+---------------------+-------------------+--------------
-----------------+--------------------+
```

DATE_FORMAT

- Syntax

```
string date_format(date|datetime|timestamp|string <date>, string <format>)
```

- Description

  Converts a date value into a string in a specified format.

- Parameters

- date: required. The date value that you want to convert. The date value can be of the DATE, DATETIME, TIMESTAMP, or STRING type.
  - Date values of the DATE, DATETIME, or STRING type are supported only when the Hive-compatible data type edition is enabled. You can run the `set odps.sql.hive.compatible=true;` command to enable the Hive-compatible data type edition.
  - Date values of the STRING type must include at least the `'yyyy-MM-dd'` part, such as `'2019-12-27'`.
- format: required. A constant of the STRING type. This parameter specifies the date format. Examples: `yyyy-MM-dd hh:mm:ss:SSS` and `yyyy-MM-dd hh:mi:ss:SSS`. format consists of the following components:
  - `YYYY` or `yyyy`: the year
  - `MM`: the month
  - `mm`: the minute
  - `dd`: the day
  - `HH`: the hour that is expressed in the base-24-numeral system
  - `hh`: the hour that is expressed in the base-12-numeral system
  - `mi`: the minute
  - `ss`: the second
  - `SSS`: the millisecond

  > **Note**
  > - If the Hive-compatible data type edition is disabled, the date format must be `yyyy-MM-dd hh:mi:ss`. If `yyyy-MM-dd hh:mm:ss` is used, the `mm` component takes the same value as the MM component.
  > - If the Hive-compatible data type edition is enabled, the date format must be `yyyy-MM-dd hh:mm:ss`. If `yyyy-MM-dd hh:mi:ss` is used, null is returned.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of date is not of the DATE, DATETIME, or TIMESTAMP type, null is returned.
  - If the value of date is null, an error is returned.
  - If the value of format is null, null is returned.
- Examples

- Examples of static data

```
-- Enable the Hive-compatible data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.hive.compatible=true;
-- If the current time is 2022-04-24 15:49, the return value is 2022-49-24 03:49:01.902
.
select date_format(from_utc_timestamp(current_timestamp(), 'UTC'),'yyyy-mm-dd hh:mm:ss.
SSS');
-- The return value is 2022-04-24.
select date_format('2022-04-24','yyyy-MM-dd');
```

- Examples of table data

  Convert date values in the datetime1 and timestamp1 columns into strings in a specified format. Data in Sample data is used in this example. Sample statement:

```
-- Enable the Hive-compatible data type edition and the MaxCompute V2.0 data type editi
on. Commit the following SET commands together with the SQL statement.
set odps.sql.hive.compatible=true;
set odps.sql.type.system.odps2=true;
select datetime1, timestamp1, date_format(datetime1,'yyyy/mm/dd'),
       date_format(datetime1,'yyyy/mm/dd HH:mm:ss'),
       date_format(timestamp1,'yyyy/mm/dd HH:mm:ss')
from mf_date_fun_t;
```

  The following result is returned:

```
+------------+------------+-----+-----+-----+
| datetime1  | timestamp1 | _c2 | _c3 | _c4 |
+------------+------------+-----+-----+-----+
| 2021-11-29 00:01:00 | 2021-01-11 00:00:00.123456789 | 2021/01/29 | 2021/01/29 00:01:0
0 | 2021/00/11 00:00:00 |
| 2021-11-28 00:02:00 | 2021-02-11 00:00:00.123456789 | 2021/02/28 | 2021/02/28 00:02:0
0 | 2021/00/11 00:00:00 |
| 2021-11-27 00:03:00 | 2021-03-11 00:00:00.123456789 | 2021/03/27 | 2021/03/27 00:03:0
0 | 2021/00/11 00:00:00 |
| 2021-11-26 00:04:00 | 2021-04-11 00:00:00.123456789 | 2021/04/26 | 2021/04/26 00:04:0
0 | 2021/00/11 00:00:00 |
| 2021-11-25 00:05:00 | 2021-05-11 00:00:00.123456789 | 2021/05/25 | 2021/05/25 00:05:0
0 | 2021/00/11 00:00:00 |
| 2021-11-24 00:06:00 | 2021-06-11 00:00:00.123456789 | 2021/06/24 | 2021/06/24 00:06:0
0 | 2021/00/11 00:00:00 |
| 2021-11-23 00:07:00 | 2021-07-11 00:00:00.123456789 | 2021/07/23 | 2021/07/23 00:07:0
0 | 2021/00/11 00:00:00 |
| 2021-11-22 00:08:00 | 2021-08-11 00:00:00.123456789 | 2021/08/22 | 2021/08/22 00:08:0
0 | 2021/00/11 00:00:00 |
| 2021-11-21 00:09:00 | 2021-09-11 00:00:00.123456789 | 2021/09/21 | 2021/09/21 00:09:0
0 | 2021/00/11 00:00:00 |
| 2021-11-20 00:10:00 | 2021-10-11 00:00:00.123456789 | 2021/10/20 | 2021/10/20 00:10:0
0 | 2021/00/11 00:00:00 |
+------------+------------+-----+-----+-----+
```

DATE_SUB

- Syntax

```
date date_sub(date|timestamp|string <startdate>, bigint <delta>)
```

- Description

  Adds or subtracts an interval that is specified by delta to or from a date value that is specified by startdate. To add or subtract an interval to or from the current time, you can use this function with the GETDATE function.

  The logic of this function is opposite to that of the DATE_ADD function.

- Parameters
  - startdate: required. The start date. A value of the DATE, DATETIME, or STRING type is supported.

    If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATE type before calculation. The input value of the STRING type must include at least the `'yyyy-mm-dd'` part, such as `'2019-12-27'`.
  - delta: required. The interval that you want to add or subtract. The value of this parameter must be of the BIGINT type. If the value of delta is greater than 0, an interval is **subtracted** from the start date. If the value of delta is less than 0, an interval is **added** to the start date. If the value of delta is 0, the date value remains unchanged.

- Return value

  A value of the DATE type is returned. The return value is in the `yyyy-mm-dd` format. The return value varies based on the following rules:
  - If the value of startdate is not of the DATE, DATETIME, or STRING type, an error is returned.
  - If the value of startdate is null, an error is returned.
  - If the value of delta is null, null is returned.

- Examples
  - Examples of static data

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
-- The return value is 2005-02-28. One day is subtracted. The last day of February is r
eturned.
select date_sub(datetime '2005-03-01 00:00:00', 1);
-- The return value is 2005-03-01. One day is added.
select date_sub(date '2005-02-28', -1);
-- The return value is 2005-02-27. Two days are subtracted.
set odps.sql.type.system.odps2=false;
select date_sub('2005-03-01 00:00:00', 2);
-- If the current time is 2021-09-10 16:31:44, the return value is 2020-09-09.
select date_sub(getdate(),1);
-- The return value is null.
select date_sub('2005-03-01 00:00:00', null);
```

- Examples of table data

  Change date values in the date1, datetime1, and timestamp1 columns based on the interval specified by delta. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, date_sub(date1,1) as date1_date_sub, datetime1, date_sub(datetime1,-1) as
datetime1_date_sub, timestamp1, date_sub(timestamp1,0) as timestamp1_date_sub from mf_d
ate_fun_t;
```

  The following result is returned:

```
+------------+----------------+---------------------+--------------------+-------------
------------------+----------------------+
| date1      | date1_date_sub | datetime1           | datetime1_date_sub | timestamp1
| timestamp1_date_sub |
+------------+----------------+---------------------+--------------------+-------------
------------------+----------------------+
| 2021-11-29 | 2021-11-28     | 2021-11-29 00:01:00 | 2021-11-30         | 2021-01-11 0
0:00:00.123456789 | 2021-01-11           |
| 2021-11-28 | 2021-11-27     | 2021-11-28 00:02:00 | 2021-11-29         | 2021-02-11 0
0:00:00.123456789 | 2021-02-11           |
| 2021-11-27 | 2021-11-26     | 2021-11-27 00:03:00 | 2021-11-28         | 2021-03-11 0
0:00:00.123456789 | 2021-03-11           |
| 2021-11-26 | 2021-11-25     | 2021-11-26 00:04:00 | 2021-11-27         | 2021-04-11 0
0:00:00.123456789 | 2021-04-11           |
| 2021-11-25 | 2021-11-24     | 2021-11-25 00:05:00 | 2021-11-26         | 2021-05-11 0
0:00:00.123456789 | 2021-05-11           |
| 2021-11-24 | 2021-11-23     | 2021-11-24 00:06:00 | 2021-11-25         | 2021-06-11 0
0:00:00.123456789 | 2021-06-11           |
| 2021-11-23 | 2021-11-22     | 2021-11-23 00:07:00 | 2021-11-24         | 2021-07-11 0
0:00:00.123456789 | 2021-07-11           |
| 2021-11-22 | 2021-11-21     | 2021-11-22 00:08:00 | 2021-11-23         | 2021-08-11 0
0:00:00.123456789 | 2021-08-11           |
| 2021-11-21 | 2021-11-20     | 2021-11-21 00:09:00 | 2021-11-22         | 2021-09-11 0
0:00:00.123456789 | 2021-09-11           |
| 2021-11-20 | 2021-11-19     | 2021-11-20 00:10:00 | 2021-11-21         | 2021-10-11 0
0:00:00.123456789 | 2021-10-11           |
+------------+----------------+---------------------+--------------------+-------------
------------------+----------------------+
```

DATEDIFF

- Syntax

```
bigint datediff(date|datetime|timestamp <date1>, date|datetime|timestamp <date2>, string
<datepart>)
```

- Description

  Calculates the difference between date1 and date2. The difference is measured in the time unit specified by datepart.

- Parameters

- date1 and date2: required. The minuend and subtrahend, which are of the DATE, DATETIME, or TIMESTAMP type. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.
- datepart: optional. The time unit, which is a constant of the STRING type.

  If you enable the MaxCompute V2.0 data type edition, you can leave datepart empty. Default value: day. For more information about the MaxCompute V2.0 data type edition, see Data type editions. The EDTF is also supported, such as `-year`, `-month`, `-mon`, `-day`, or `-hour`.

  > **Note** This function omits the lower unit based on the time unit specified by datepart and then calculates the result.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of date1 or date2 is not of the DATE, DATETIME, or TIMESTAMP type, an error is returned.
  - If date1 is earlier than date2, a negative value is returned.
  - If the value of date1 or date2 is null, null is returned.
  - If the value of datepart is null, null is returned.

- Examples
  - Examples of static data

```
-- The start time is 2005-12-31 23:59:59 and the end time is 2006-01-01 00:00:00.
    -- The return value is 1.
    select datediff(end, start, 'dd');
    -- The return value is 1.
    select datediff(end, start, 'mm');
    -- The return value is 1.
    select datediff(end, start, 'yyyy');
    -- The return value is 1.
    select datediff(end, start, 'hh');
    -- The return value is 1.
    select datediff(end, start, 'mi');
    -- The return value is 1.
    select datediff(end, start, 'ss');
-- The return value is 1800.
select datediff(datetime'2013-05-31 13:00:00', datetime'2013-05-31 12:30:00', 'ss');
-- The return value is 30.
set odps.sql.type.system.odps2=false;
select datediff('2013-05-31 13:00:00', '2013-05-31 12:30:00', 'mi');
-- The return value is 11.
select datediff(date '2013-05-21', date '2013-05-10', 'dd');
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement. The return value is 1800.
set odps.sql.type.system.odps2=true;
select datediff(timestamp '2013-05-31 13:00:00', timestamp '2013-05-31 12:30:00', 'ss')
;
-- The start time is 2018-06-04 19:33:23.234 and the end time is 2018-06-04 19:33:23.25
0. Date values that are accurate to the millisecond do not adopt the standard DATETIME
type and cannot be implicitly converted into the DATETIME type. In this case, an explic
it conversion is required. The return value is 16.
select datediff(to_date('2018-06-04 19:33:23.250', 'yyyy-mm-dd hh:mi:ss.ff3'), to_date(
'2018-06-04 19:33:23.234', 'yyyy-mm-dd hh:mi:ss.ff3') , 'ff3');
-- The return value is null.
select datediff(date '2013-05-21', date '2013-05-10', null);
-- The return value is null.
select datediff(date '2013-05-21', null, 'dd');
```

- Examples of table data

  Calculate the differences between the values in the date1 and date2 columns, between the values in the datetime1 and datetime2 columns, and between the values in the timestamp1 and timestamp2 columns. The differences are measured in the specified time unit. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, date2, datediff(date1,date2,'dd') as date1_date2_datediff, datetime1, dat
etime2, datediff(datetime1, datetime2,'dd') as datetime1_datetime2_datediff, timestamp1
, timestamp2, datediff(timestamp1, timestamp2,'mm') as timestamp1_timestamp2_datediff f
rom mf_date_fun_t;
```

  The following result is returned:

```
+------------+------------+---------------------+--------------------+--------------
------+-----------------------------+-------------------------------+----------------
--------------+---------------------------------+
| date1      | date2      | date1_date2_datediff | datetime1          | datetime2
| datetime1_datetime2_datediff | timestamp1                  | timestamp2
| timestamp1_timestamp2_datediff |
+------------+------------+---------------------+--------------------+--------------
------+-----------------------------+-------------------------------+----------------
--------------+---------------------------------+
| 2021-11-29 | 2021-10-29 | 31                  | 2021-11-29 00:01:00 | 2021-10-29 00:
00:00 | 31                          | 2021-01-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -9                              |
| 2021-11-28 | 2021-10-29 | 30                  | 2021-11-28 00:02:00 | 2021-10-29 00:
00:00 | 30                          | 2021-02-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -8                              |
| 2021-11-27 | 2021-10-29 | 29                  | 2021-11-27 00:03:00 | 2021-10-29 00:
00:00 | 29                          | 2021-03-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -7                              |
| 2021-11-26 | 2021-10-29 | 28                  | 2021-11-26 00:04:00 | 2021-10-29 00:
00:00 | 28                          | 2021-04-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -6                              |
| 2021-11-25 | 2021-10-29 | 27                  | 2021-11-25 00:05:00 | 2021-10-29 00:
00:00 | 27                          | 2021-05-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -5                              |
| 2021-11-24 | 2021-10-29 | 26                  | 2021-11-24 00:06:00 | 2021-10-29 00:
00:00 | 26                          | 2021-06-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -4                              |
| 2021-11-23 | 2021-10-29 | 25                  | 2021-11-23 00:07:00 | 2021-10-29 00:
00:00 | 25                          | 2021-07-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -3                              |
| 2021-11-22 | 2021-10-29 | 24                  | 2021-11-22 00:08:00 | 2021-10-29 00:
00:00 | 24                          | 2021-08-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -2                              |
| 2021-11-21 | 2021-10-29 | 23                  | 2021-11-21 00:09:00 | 2021-10-29 00:
00:00 | 23                          | 2021-09-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | -1                              |
| 2021-11-20 | 2021-10-29 | 22                  | 2021-11-20 00:10:00 | 2021-10-29 00:
00:00 | 22                          | 2021-10-11 00:00:00.123456789 | 2021-10-11 00:00
:00.123456789 | 0                               |
+------------+------------+---------------------+--------------------+--------------
------+-----------------------------+-------------------------------+----------------
--------------+---------------------------------+
```

DATEPART

- Syntax

```
bigint datepart(date|datetime|timestamp <date>, string <datepart>)
```

- Description

  Returns a specified component of a date value based on the time unit specified by datepart.

- Parameters

- date: required. A value of the DATE, DATETIME, or TIMESTAMP type. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.
- datepart: required. A constant of the STRING type. This parameter supports EDTF.

  The value of this parameter is specified in compliance with the rules of conversions between the STRING and DATETIME types. The value `yyyy` indicates that the DATEADD function adds an interval to the year component of the date value. The value `mm` indicates that the DATEADD function adds an interval to the month component of the date value. The value `dd` indicates that the DATEADD function adds an interval to the day component of the date value. For more information about the rules for type conversions, see Type conversions. The Extended Date/Time Format (EDTF) is also supported, such as `-year`, `-month`, `-mon`, `-day`, or `-hour`.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATE, DATETIME, or TIMESTAMP type, an error is returned.
  - If the value of date is null, an error is returned.
  - If the value of datepart is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 2013.
select datepart(datetime'2013-06-08 01:10:00', 'yyyy');
-- The return value is 6.
select datepart(datetime'2013-06-08 01:10:00', 'mm');
-- The return value is 2013.
select datepart(date '2013-06-08', 'yyyy');
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement. The return value is 2013.
set odps.sql.type.system.odps2=true;
select datepart(timestamp '2013-06-08 01:10:00', 'yyyy');
-- The return value is 2013.
set odps.sql.type.system.odps2=false;
select datepart('2013-06-08 01:10:00', 'yyyy');
-- The return value is null.
select datepart(date '2013-06-08', null);
```

- Examples of table data

  Extract date values from the date1, datetime1, and timestamp1 columns based on the time unit specified by datepart. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, datepart(date1,'yyyy') as date1_datepart, datetime1, datepart(datetime1,'
dd') as datetime1_datepart, timestamp1, datepart(timestamp1,'mm') as timestamp1_datepar
t from mf_date_fun_t;
```

  The following result is returned:

```
+------------+----------------+---------------------+-------------------------+--------
----------------------+---------------------+
| date1      | date1_datepart | datetime1           | datetime1_datepart      | timesta
mp1                   | timestamp1_datepart |
+------------+----------------+---------------------+-------------------------+--------
----------------------+---------------------+
| 2021-11-29 | 2021           | 2021-11-29 00:01:00 | 29                      | 2021-01
-11 00:00:00.123456789 | 1                   |
| 2021-11-28 | 2021           | 2021-11-28 00:02:00 | 28                      | 2021-02
-11 00:00:00.123456789 | 2                   |
| 2021-11-27 | 2021           | 2021-11-27 00:03:00 | 27                      | 2021-03
-11 00:00:00.123456789 | 3                   |
| 2021-11-26 | 2021           | 2021-11-26 00:04:00 | 26                      | 2021-04
-11 00:00:00.123456789 | 4                   |
| 2021-11-25 | 2021           | 2021-11-25 00:05:00 | 25                      | 2021-05
-11 00:00:00.123456789 | 5                   |
| 2021-11-24 | 2021           | 2021-11-24 00:06:00 | 24                      | 2021-06
-11 00:00:00.123456789 | 6                   |
| 2021-11-23 | 2021           | 2021-11-23 00:07:00 | 23                      | 2021-07
-11 00:00:00.123456789 | 7                   |
| 2021-11-22 | 2021           | 2021-11-22 00:08:00 | 22                      | 2021-08
-11 00:00:00.123456789 | 8                   |
| 2021-11-21 | 2021           | 2021-11-21 00:09:00 | 21                      | 2021-09
-11 00:00:00.123456789 | 9                   |
| 2021-11-20 | 2021           | 2021-11-20 00:10:00 | 20                      | 2021-10
-11 00:00:00.123456789 | 10                  |
+------------+----------------+---------------------+-------------------------+--------
----------------------+---------------------+
```

DATETRUNC

- Syntax

```
date|datetime datetrunc (date|datetime|timestamp <date>, string <datepart>)
```

- Description

  Truncates a date value based on the time unit specified by datepart.

- Parameters

- date: required. A value of the DATE, DATETIME, or TIMESTAMP type. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.
- datepart: required. A constant of the STRING type. This parameter supports EDTF.

  The value of this parameter is specified in compliance with the rules of conversions between the STRING and DATETIME types. The value `yyyy` indicates that the DATEADD function adds an interval to the year component of the date value. The value `mm` indicates that the DATEADD function adds an interval to the month component of the date value. The value `dd` indicates that the DATEADD function adds an interval to the day component of the date value. For more information about the rules for type conversions, see Type conversions. The Extended Date/Time Format (EDTF) is also supported, such as `-year`, `-month`, `-mon`, `-day`, or `-hour`.

- Return value

  A value of the DATE or DATETIME type is returned. The return value is in the `yyyy-mm-dd` or `yyyy-mm-dd hh:mi:ss` format. The return value varies based on the following rules:

  - If the value of date is not of the DATE, DATETIME, or TIMESTAMP type, an error is returned.
  - If the value of date is null, an error is returned.
  - If the value of datepart is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 2011-01-01 00:00:00.
select datetrunc(datetime'2011-12-07 16:28:46', 'yyyy');
-- The return value is 2011-12-01 00:00:00.
select datetrunc(datetime'2011-12-07 16:28:46', 'month');
-- The return value is 2011-12-07 00:00:00.
select datetrunc(datetime'2011-12-07 16:28:46', 'DD');
-- The return value is 2011-01-01.
select datetrunc(date '2011-12-07', 'yyyy');
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement. The return value is 2011-01-01 00:00:00.0.
set odps.sql.type.system.odps2=true;
select datetrunc(timestamp '2011-12-07 16:28:46', 'yyyy');
-- The return value is 2011-01-01 00:00:00.0.
set odps.sql.type.system.odps2=false;
select datetrunc('2011-12-07 16:28:46', 'yyyy');
-- The return value is null.
select datetrunc(date '2011-12-07', null);
```

- Examples of table data

  Truncate date values in the date1, datetime1, and timestamp1 columns based on the time unit specified by datepart. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, datetrunc(date1,'yyyy') as date1_datetrunc, datetime1, datetrunc(datetime
1,'dd') as datetime1_datetrunc, timestamp1, datetrunc(timestamp1,'mm') as timestamp1_da
tetrunc from mf_date_fun_t;
```

  The following result is returned:

```
+------------+-----------------+---------------------+---------------------+-----------
-------------------+----------------------+
| date1      | date1_datetrunc | datetime1           | datetime1_datetrunc | timestamp1
| timestamp1_datetrunc |
+------------+-----------------+---------------------+---------------------+-----------
-------------------+----------------------+
| 2021-11-29 | 2021-01-01      | 2021-11-29 00:01:00 | 2021-11-29 00:00:00 | 2021-01-11
00:00:00.123456789 | 2021-01-01 00:00:00  |
| 2021-11-28 | 2021-01-01      | 2021-11-28 00:02:00 | 2021-11-28 00:00:00 | 2021-02-11
00:00:00.123456789 | 2021-02-01 00:00:00  |
| 2021-11-27 | 2021-01-01      | 2021-11-27 00:03:00 | 2021-11-27 00:00:00 | 2021-03-11
00:00:00.123456789 | 2021-03-01 00:00:00  |
| 2021-11-26 | 2021-01-01      | 2021-11-26 00:04:00 | 2021-11-26 00:00:00 | 2021-04-11
00:00:00.123456789 | 2021-04-01 00:00:00  |
| 2021-11-25 | 2021-01-01      | 2021-11-25 00:05:00 | 2021-11-25 00:00:00 | 2021-05-11
00:00:00.123456789 | 2021-05-01 00:00:00  |
| 2021-11-24 | 2021-01-01      | 2021-11-24 00:06:00 | 2021-11-24 00:00:00 | 2021-06-11
00:00:00.123456789 | 2021-06-01 00:00:00  |
| 2021-11-23 | 2021-01-01      | 2021-11-23 00:07:00 | 2021-11-23 00:00:00 | 2021-07-11
00:00:00.123456789 | 2021-07-01 00:00:00  |
| 2021-11-22 | 2021-01-01      | 2021-11-22 00:08:00 | 2021-11-22 00:00:00 | 2021-08-11
00:00:00.123456789 | 2021-08-01 00:00:00  |
| 2021-11-21 | 2021-01-01      | 2021-11-21 00:09:00 | 2021-11-21 00:00:00 | 2021-09-11
00:00:00.123456789 | 2021-09-01 00:00:00  |
| 2021-11-20 | 2021-01-01      | 2021-11-20 00:10:00 | 2021-11-20 00:00:00 | 2021-10-11
00:00:00.123456789 | 2021-10-01 00:00:00  |
+------------+-----------------+---------------------+---------------------+-----------
-------------------+----------------------+
```

FROM_UNIXTIME

- Syntax

```
datetime from_unixtime(bigint <unixtime>)
```

- Description

  Converts unixtime of the BIGINT type into a date value of the DATETIME type.

- Parameters

unixtime: required. A date value of the BIGINT type in the UNIX format. The value of this parameter is accurate to the second. If the input value is of the STRING, DOUBLE, or DECIMAL type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the BIGINT type before calculation.

- Return value

  A value of the DATETIME type is returned. The return value is in the `yyyy-mm-dd hh:mi:ss` format. If the value of unixtime is null, null is returned.

  > **Note** In the Hive-compatible data type edition where `set odps.sql.hive.compatible=true;` is run, if the input value is of the STRING type, a date value of the STRING type is returned.

- Examples
  - Examples of static data

    ```
    -- The return value is 1973-11-30 05:33:09.
    select from_unixtime(123456789);
    -- The return value is 1973-11-30 05:33:09.
    set odps.sql.type.system.odps2=false;
    select from_unixtime('123456789');
    -- The return value is null.
    select from_unixtime(null);
    ```

  - Examples of table data

    Convert values in the date4 column into date values. Data in Sample data is used in this example. Sample statement:

    ```
    select date4, from_unixtime(date4) as date4_from_unixtime from mf_date_fun_t;
    ```

    The following result is returned:

    ```
    +------------+---------------------+
    | date4      | date4_from_unixtime |
    +------------+---------------------+
    | 123456780  | 1973-11-30 05:33:00 |
    | 123456781  | 1973-11-30 05:33:01 |
    | 123456782  | 1973-11-30 05:33:02 |
    | 123456783  | 1973-11-30 05:33:03 |
    | 123456784  | 1973-11-30 05:33:04 |
    | 123456785  | 1973-11-30 05:33:05 |
    | 123456786  | 1973-11-30 05:33:06 |
    | 123456787  | 1973-11-30 05:33:07 |
    | 123456788  | 1973-11-30 05:33:08 |
    | 123456789  | 1973-11-30 05:33:09 |
    +------------+---------------------+
    ```

GETDATE

- Syntax

  ```
  datetime getdate()
  ```

- Description

Returns the current system time as a date value. MaxCompute uses UTC+8 as the standard time zone.

- Return value

  The current date and time are returned, which are of the DATETIME type.

  > **Note** In MaxCompute SQL, `GETDATE` always returns a fixed value. The return value is an arbitrary time during the execution of the MaxCompute SQL job. The time is accurate to the second. If you enable the MaxCompute V2.0 data type edition, the time is accurate to the millisecond.

ISDATE

- Syntax

```
boolean isdate(string <date>, string <format>)
```

- Description

  Determines whether a date string can be converted into a date value in a specified format. If the date string can be converted into a date value in the specified format, true is returned. Otherwise, false is returned.

- Parameters
  - date: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.
  - format: required. A constant of the STRING type. This parameter does not support EDTF. If redundant format strings exist in format, this function converts the date string that corresponds to the first format string into a date value. The rest strings are considered delimiters. For example, `isdate("1234-yyyy", "yyyy-yyyy")` returns true.

- Return value

  A value of the BOOLEAN type is returned. If the value of date or format is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is true.
select isdate('2021-10-11','yyyy-mm-dd');
-- The return value is false.
set odps.sql.type.system.odps2=false;
select isdate(1678952314,'yyyy-mm-dd');
```

- Examples of table data

  Determine whether date strings in the date3 column can be converted into date values in a specified format. Data in Sample data is used in this example. Sample statement:

```
select date3, isdate(date3,'yyyy-mm-dd') as date3_isdate from mf_date_fun_t;
```

  The following result is returned:

```
+------------+--------------+
| date3      | date3_isdate |
+------------+--------------+
| 2021-11-20 | true         |
| 2021-11-21 | true         |
| 2021-11-22 | true         |
| 2021-11-23 | true         |
| 2021-11-24 | true         |
| 2021-11-25 | true         |
| 2021-11-26 | true         |
| 2021-11-27 | true         |
| 2021-11-28 | true         |
| 2021-11-29 | true         |
+------------+--------------+
```

LASTDAY

- Syntax

```
datetime lastday(datetime <date>)
```

- Description

  Returns the last day of the month in which a date value falls. Only the day component is truncated. The hour, minute, and second components are expressed as `00:00:00`.

- Parameters

  date: a date value of the DATETIME type. The date value is in the `yyyy-mm-dd hh:mi:ss` format. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.

- Return value

  A value of the DATETIME type is returned. The return value is in the `yyyy-mm-dd hh:mi:ss` format. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME or STRING type or the format does not meet the requirements, an error is returned.
  - If the value of date is null, null is returned.

- Examples

- Examples of static data

```
-- The return value is 2013-06-30 00:00:00.
select lastday (datetime '2013-06-08 01:10:00');
-- The return value is 2013-06-30 00:00:00.
set odps.sql.type.system.odps2=false;
select lastday ('2013-06-08 01:10:00');
-- The return value is null.
select lastday (null);
```

- Examples of table data

  Obtain the last day of the month in which each date value in the datetime1 column falls. Data in Sample data is used in this example. Sample statement:

```
select datetime1, lastday(datetime1) as datetime1_lastday from mf_date_fun_t;
```

  The following result is returned:

```
+---------------------+---------------------+
| datetime1           | datetime1_lastday   |
+---------------------+---------------------+
| 2021-11-29 00:01:00 | 2021-11-30 00:00:00 |
| 2021-11-28 00:02:00 | 2021-11-30 00:00:00 |
| 2021-11-27 00:03:00 | 2021-11-30 00:00:00 |
| 2021-11-26 00:04:00 | 2021-11-30 00:00:00 |
| 2021-11-25 00:05:00 | 2021-11-30 00:00:00 |
| 2021-11-24 00:06:00 | 2021-11-30 00:00:00 |
| 2021-11-23 00:07:00 | 2021-11-30 00:00:00 |
| 2021-11-22 00:08:00 | 2021-11-30 00:00:00 |
| 2021-11-21 00:09:00 | 2021-11-30 00:00:00 |
| 2021-11-20 00:10:00 | 2021-11-30 00:00:00 |
+---------------------+---------------------+
```

TO_DATE

- Syntax

```
datetime to_date(string <date>, string <format>)
```

- Description

  Converts a string into a date value in a specified format.

- Parameters
  - date: required. A date value of the STRING type. This parameter specifies the date string that you want to convert. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- format: required. A constant of the STRING type. This parameter specifies the date format. format does not support EDTF. Other characters are omitted as invalid characters during parsing.

  The value of format must contain `yyyy`. Otherwise, null is returned. If redundant format strings exist in format, this function converts the date string that corresponds to the first format string into a date value. The rest strings are considered delimiters. For example, `to_date("1234-2234", "yyyy-yyyy")` returns `1234-01-01 00:00:00`.

  The format consists of different date components. `yyyy` indicates a 4-digit year. `mm` indicates a 2-digit month. `dd` indicates a 2-digit day. `hh` indicates an hour based on the base-24 numeral system. `mi` indicates a 2-digit minute. `ss` indicates a 2-digit second. `ff3` indicates a 3-digit millisecond.

- Return value

  A value of the DATETIME type is returned. The return value is in the `yyyy-mm-dd hh:mi:ss` format. If the value of date or format is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 2010-12-03 00:00:00.
select to_date('Alibaba 2010-12*03', 'Alibaba yyyy-mm*dd');
-- The return value is 2008-07-18 00:00:00.
select to_date('20080718', 'yyyymmdd');
-- The return value is 2008-07-18 20:30:00.
select to_date('200807182030','yyyymmddhhmi');
-- '2008718' cannot be converted into a standard date value, and an error is returned.
The value must be written as '20080718'.
select to_date('2008718', 'yyyymmdd');
'Alibaba 2010-12*3' cannot be converted into a standard date value, and an error is ret
urned. The value must be written as 'Alibaba 2010-12*03'.
select to_date('Alibaba 2010-12*3', 'Alibaba yyyy-mm*dd');
'2010-24-01' cannot be converted into a standard date value, and an error is returned.
The value must be written as '2010-01-24'.
select to_date('2010-24-01', 'yyyy');
-- The return value is 2018-10-30 15:13:12.
select to_date('20181030 15-13-12.345','yyyymmdd hh-mi-ss.ff3');
-- The return value is null.
select to_date(null,'yyyymmdd hh-mi-ss.ff3');
-- The return value is null.
select to_date('20181030 15-13-12.345',null);
```

- Examples of table data

  Convert date values in the date3 column into date values in a specified format. Data in Sample data is used in this example. Sample statement:

```
select date3, to_date(date3, 'yyyy-mm-dd') as date3_to_date from mf_date_fun_t;
```

  The following result is returned:

```
+------------+---------------------+
| date3      | date3_to_date       |
+------------+---------------------+
| 2021-11-20 | 2021-11-20 00:00:00 |
| 2021-11-21 | 2021-11-21 00:00:00 |
| 2021-11-22 | 2021-11-22 00:00:00 |
| 2021-11-23 | 2021-11-23 00:00:00 |
| 2021-11-24 | 2021-11-24 00:00:00 |
| 2021-11-25 | 2021-11-25 00:00:00 |
| 2021-11-26 | 2021-11-26 00:00:00 |
| 2021-11-27 | 2021-11-27 00:00:00 |
| 2021-11-28 | 2021-11-28 00:00:00 |
| 2021-11-29 | 2021-11-29 00:00:00 |
+------------+---------------------+
```

TO_CHAR

- Syntax

```
string to_char(datetime <date>, string <format>)
```

- Description

  Converts a date value of the DATETIME type into a string in a specified format.

- Parameters
  - date: required. A date value of the DATETIME type. The date value is in the `yyyy-mm-dd hh:mi:ss` format. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.
  - format: required. A constant of the STRING type. In the format parameter, the date format part is replaced by the related data and the other characters remain unchanged in the output.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of date is not of the DATETIME or STRING type, an error is returned.
  - If the value of date is null, an error is returned.
  - If the value of format is null, null is returned.

- Examples

- Examples of static data

```
-- The return value is 'Alibaba Cloud Financial Services 2010-12*03'. If the MaxCompute
client runs in Windows, Chinese characters may not be properly displayed in the returne
d result.
select to_char(datetime'2010-12-03 00:00:00', 'Alibaba Cloud Financial Services yyyy-mm
*dd');
-- The return value is 20080718.
select to_char(datetime'2008-07-18 00:00:00', 'yyyymmdd');
-- The return value is 20080718.
set odps.sql.type.system.odps2=false;
select to_char('2008-07-18 00:00:00', 'yyyymmdd');
-- 'Alibaba 2010-12*3' cannot be converted into a standard date value, and an error is
returned. The value must be written as 'Alibaba 2010-12*03'.
select to_char(datetime'Alibaba 2010-12*3', 'Alibaba yyyy-mm*dd');
-- '2010-24-01' cannot be converted into a standard date value, and an error is returne
d. The value must be written as '2010-01-24'.
select to_char(datetime'2010-24-01', 'yyyy');
-- '2008718' cannot be converted into a standard date value, and an error is returned.
The value must be written as '20080718'.
select to_char(datetime'2008718', 'yyyymmdd');
-- The return value is null.
select to_char(datetime'2010-12-03 00:00:00', null);
```

- Examples of table data

  Convert date values in the datetime1 column into strings in a specified format. Data in Sample data is used in this example. Sample statement:

```
select datetime1, to_char(datetime1, 'yyyy-mm-dd') as datetime1_to_char from mf_date_fu
n_t;
```

  The following result is returned:

```
+---------------------+-------------------+
| datetime1           | datetime1_to_char |
+---------------------+-------------------+
| 2021-11-29 00:01:00 | 2021-11-29        |
| 2021-11-28 00:02:00 | 2021-11-28        |
| 2021-11-27 00:03:00 | 2021-11-27        |
| 2021-11-26 00:04:00 | 2021-11-26        |
| 2021-11-25 00:05:00 | 2021-11-25        |
| 2021-11-24 00:06:00 | 2021-11-24        |
| 2021-11-23 00:07:00 | 2021-11-23        |
| 2021-11-22 00:08:00 | 2021-11-22        |
| 2021-11-21 00:09:00 | 2021-11-21        |
| 2021-11-20 00:10:00 | 2021-11-20        |
+---------------------+-------------------+
```

UNIX_TIMESTAMP

- Syntax

```
bigint unix_timestamp(datetime|date|timestamp|string <date>)
```

- Description

  Converts a date value into a UNIX timestamp that is an integer.

- Parameters

  date: required. A date value of the DATETIME, DATE, TIMESTAMP, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation. If you enable the MaxCompute V2.0 data type edition, the implicit conversion fails. In this case, you must use the `CAST` function, such as `unix_timestamp(cast(... as datetime))`, to convert data types. You can also disable the MaxCompute V2.0 data type edition.

- Return value

  A UNIX timestamp of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, DATE, TIMESTAMP, or STRING type or the format does not meet the requirements, an error is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 1237518660.
select unix_timestamp(datetime'2009-03-20 11:11:00');
-- The return value is 1237518660.
set odps.sql.type.system.odps2=false;
select unix_timestamp('2009-03-20 11:11:00');
-- The return value is null.
select unix_timestamp(null);
```

- Examples of table data

  Convert date values in the date1, datetime1, and timestamp1 columns into UNIX timestamps that are integers. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, unix_timestamp(date1) as date1_unix_timestamp, datetime1, unix_timestamp(
datetime1) as datetime1_unix_timestamp, timestamp1, unix_timestamp(timestamp1) as times
tamp1_unix_timestamp from mf_date_fun_t;
```

  The following result is returned:

```
+------------+---------------------+---------------------+--------------------------+-
-----------------------------+---------------------------+
| date1      | date1_unix_timestamp | datetime1           | datetime1_unix_timestamp |
timestamp1                   | timestamp1_unix_timestamp |
+------------+---------------------+---------------------+--------------------------+-
-----------------------------+---------------------------+
| 2021-11-29 | 1638115200           | 2021-11-29 00:01:00 | 1638115260               |
2021-01-11 00:00:00.123456789 | 1610294400                |
| 2021-11-28 | 1638028800           | 2021-11-28 00:02:00 | 1638028920               |
2021-02-11 00:00:00.123456789 | 1612972800                |
| 2021-11-27 | 1637942400           | 2021-11-27 00:03:00 | 1637942580               |
2021-03-11 00:00:00.123456789 | 1615392000                |
| 2021-11-26 | 1637856000           | 2021-11-26 00:04:00 | 1637856240               |
2021-04-11 00:00:00.123456789 | 1618070400                |
| 2021-11-25 | 1637769600           | 2021-11-25 00:05:00 | 1637769900               |
2021-05-11 00:00:00.123456789 | 1620662400                |
| 2021-11-24 | 1637683200           | 2021-11-24 00:06:00 | 1637683560               |
2021-06-11 00:00:00.123456789 | 1623340800                |
| 2021-11-23 | 1637596800           | 2021-11-23 00:07:00 | 1637597220               |
2021-07-11 00:00:00.123456789 | 1625932800                |
| 2021-11-22 | 1637510400           | 2021-11-22 00:08:00 | 1637510880               |
2021-08-11 00:00:00.123456789 | 1628611200                |
| 2021-11-21 | 1637424000           | 2021-11-21 00:09:00 | 1637424540               |
2021-09-11 00:00:00.123456789 | 1631289600                |
| 2021-11-20 | 1637337600           | 2021-11-20 00:10:00 | 1637338200               |
2021-10-11 00:00:00.123456789 | 1633881600                |
+------------+---------------------+---------------------+--------------------------+-
-----------------------------+---------------------------+
```

WEEKDAY

- Syntax

```
bigint weekday (datetime <date>)
```

- Description

  Returns a number that represents the day of the week in which a date value falls.

- Parameters

date: required. A value of the DATETIME type. The format is `yyyy-mm-dd hh:mi:ss`. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - Monday is considered the first day of the week and its return value is 0. Days of the week are numbered in ascending order starting from 0. The return value of Sunday is 6.
  - If the value of date is not of the DATETIME or STRING type or the format does not meet the requirements, an error is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 4.
select weekday (datetime '2009-03-20 11:11:00');
-- The return value is 4.
set odps.sql.type.system.odps2=false;
select weekday ('2009-03-20 11:11:00');
-- The return value is null.
select weekday (null);
```

  - Examples of table data

    Calculate the day of the week in which each date value in the datetime1 column falls. Data in Sample data is used in this example. Sample statement:

```
select datetime1, weekday(datetime1) as datetime1_weekday from mf_date_fun_t;
```

    The following result is returned:

```
+---------------------+-------------------+
| datetime1           | datetime1_weekday |
+---------------------+-------------------+
| 2021-11-29 00:01:00 | 0                 |
| 2021-11-28 00:02:00 | 6                 |
| 2021-11-27 00:03:00 | 5                 |
| 2021-11-26 00:04:00 | 4                 |
| 2021-11-25 00:05:00 | 3                 |
| 2021-11-24 00:06:00 | 2                 |
| 2021-11-23 00:07:00 | 1                 |
| 2021-11-22 00:08:00 | 0                 |
| 2021-11-21 00:09:00 | 6                 |
| 2021-11-20 00:10:00 | 5                 |
+---------------------+-------------------+
```

WEEKOFYEAR

- Syntax

```
bigint weekofyear (datetime <date>)
```

- Description

Returns a number that represents the week of the year in which a date value falls. Monday is considered the first day of the week.

> **Note** To determine whether a week belongs to the current year or to the next year, find the year in which more than four days of the week fall. If the week belongs to the current year, it is considered the last week of the year. If the week belongs to the next year, it is considered the first week of the next year.

- Parameters

  date: required. A value of the DATETIME type. The format is `yyyy-mm-dd hh:mi:ss`. If the input value is of the STRING type and the MaxCompute V1.0 data type edition is used in your project, the input value is implicitly converted into the DATETIME type before calculation.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 1. 20141229 is in year 2014, but most days of the week fall in y
ear 2015. Therefore, the return value 1 indicates the first week of year 2015.
select weekofyear(to_date("20141229", "yyyymmdd"));
-- The return value is 1.
select weekofyear(to_date("20141231", "yyyymmdd"));
-- The return value is 53.
select weekofyear(to_date("20151229", "yyyymmdd"));
-- The return value is 48.
set odps.sql.type.system.odps2=false;
select weekofyear('2021-11-29 00:01:00');
-- The return value is null.
select weekofyear('20141231');
-- The return value is null.
select weekofyear(null);
```

- Examples of table data

  Calculate the day of the week in which each date value in the datetime1 column falls. Data in Sample data is used in this example. Sample statement:

```
select datetime1, weekofyear(datetime1) as datetime1_weekofyear from mf_date_fun_t;
```

  The following result is returned:

```
+---------------------+----------------------+
| datetime1           | datetime1_weekofyear |
+---------------------+----------------------+
| 2021-11-29 00:01:00 | 48                   |
| 2021-11-28 00:02:00 | 47                   |
| 2021-11-27 00:03:00 | 47                   |
| 2021-11-26 00:04:00 | 47                   |
| 2021-11-25 00:05:00 | 47                   |
| 2021-11-24 00:06:00 | 47                   |
| 2021-11-23 00:07:00 | 47                   |
| 2021-11-22 00:08:00 | 47                   |
| 2021-11-21 00:09:00 | 46                   |
| 2021-11-20 00:10:00 | 46                   |
+---------------------+----------------------+
```

ADD_MONTHS

- Syntax

```
string add_months(date|datetime|timestamp|string <startdate>, int <num_months>)
```

- Description

  Returns a date value that is obtained after the number of months specified by num_months is added to startdate. This function is an additional function of MaxCompute V2.0.

- Parameters
  - startdate: required. A value of the DATE, DATETIME, TIMESTAMP, or STRING type. The value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.
  - num_months: required. A value of the INT type.

- Return value

  A value of the STRING type is returned. The return value is in the `yyyy-mm-dd` format. The return value varies based on the following rules:
  - If the value of startdate is not of the DATE, DATETIME, TIMESTAMP, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of startdate is null, an error is returned.
  - If the value of num_months is null, null is returned.

- Examples

- Examples of static data

```
-- The return value is 2017-05-14.
select add_months('2017-02-14',3);
-- The return value is 0017-05-14.
select add_months('17-2-14',3);
-- The return value is 2017-05-14.
select add_months('2017-02-14 21:30:00',3);
-- The return value is null.
select add_months('20170214',3);
-- The return value is null.
select add_months('2017-02-14 21:30:00',null);
```

- Examples of table data

  Convert date values in the date1, datetime1, timestamp1, and date3 columns into UNIX timestamps that are integers. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, add_months(date1,1) as date1_add_months, datetime1, add_months(datetime1,
2) as datetime1_add_months, timestamp1, add_months(timestamp1,3) as timestamp1_add_mont
hs, date3, add_months(date3,4) as date3_add_months from mf_date_fun_t;
```

  The following result is returned:

```
+------------+------------------+---------------------+----------------------+---------
---------------------+----------------------+------------+------------------+
| date1      | date1_add_months | datetime1           | datetime1_add_months | timestam
p1                   | timestamp1_add_months | date3      | date3_add_months |
+------------+------------------+---------------------+----------------------+---------
---------------------+----------------------+------------+------------------+
| 2021-11-29 | 2021-12-29       | 2021-11-29 00:01:00 | 2022-01-29           | 2021-01-
11 00:00:00.123456789 | 2021-04-11           | 2021-11-20 | 2022-03-20       |
| 2021-11-28 | 2021-12-28       | 2021-11-28 00:02:00 | 2022-01-28           | 2021-02-
11 00:00:00.123456789 | 2021-05-11           | 2021-11-21 | 2022-03-21       |
| 2021-11-27 | 2021-12-27       | 2021-11-27 00:03:00 | 2022-01-27           | 2021-03-
11 00:00:00.123456789 | 2021-06-11           | 2021-11-22 | 2022-03-22       |
| 2021-11-26 | 2021-12-26       | 2021-11-26 00:04:00 | 2022-01-26           | 2021-04-
11 00:00:00.123456789 | 2021-07-11           | 2021-11-23 | 2022-03-23       |
| 2021-11-25 | 2021-12-25       | 2021-11-25 00:05:00 | 2022-01-25           | 2021-05-
11 00:00:00.123456789 | 2021-08-11           | 2021-11-24 | 2022-03-24       |
| 2021-11-24 | 2021-12-24       | 2021-11-24 00:06:00 | 2022-01-24           | 2021-06-
11 00:00:00.123456789 | 2021-09-11           | 2021-11-25 | 2022-03-25       |
| 2021-11-23 | 2021-12-23       | 2021-11-23 00:07:00 | 2022-01-23           | 2021-07-
11 00:00:00.123456789 | 2021-10-11           | 2021-11-26 | 2022-03-26       |
| 2021-11-22 | 2021-12-22       | 2021-11-22 00:08:00 | 2022-01-22           | 2021-08-
11 00:00:00.123456789 | 2021-11-11           | 2021-11-27 | 2022-03-27       |
| 2021-11-21 | 2021-12-21       | 2021-11-21 00:09:00 | 2022-01-21           | 2021-09-
11 00:00:00.123456789 | 2021-12-11           | 2021-11-28 | 2022-03-28       |
| 2021-11-20 | 2021-12-20       | 2021-11-20 00:10:00 | 2022-01-20           | 2021-10-
11 00:00:00.123456789 | 2022-01-11           | 2021-11-29 | 2022-03-29       |
+------------+------------------+---------------------+----------------------+---------
---------------------+----------------------+------------+------------------+
```

CURRENT_TIMESTAMP

- Syntax

```
timestamp current_timestamp()
```

- Description

  Returns the current timestamp. The return value is not fixed. This function is an additional function of MaxCompute V2.0.

- Return value

A value of the TIMESTAMP type is returned.

- Examples

```
-- The return value is '2017-08-03 11:50:30.661'.
set odps.sql.type.system.odps2=true;
select current_timestamp();
```

DAY

- Syntax

```
int day(datetime|timestamp|date|string <date>)
```

- Description

  Returns the day in which a date value falls. This function is an additional function of MaxCompute V2.0.

- Parameters

  date: required. A date value of the DATETIME, TIMESTAMP, DATE, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, DATE, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 1.
select day('2014-09-01');
-- The return value is null.
select day('20140901');
-- The return value is null.
select day(null);
```

- Examples of table data

  Obtain the day in which each date value in the date1, datetime1, timestamp1, and date3 columns falls. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, day(date1) as date1_day, datetime1, day(datetime1) as datetime1_day, time
stamp1, day(timestamp1) as timestamp1_day, date3, day(date3) as date3_day from mf_date_
fun_t;
```

  The following result is returned:

```
+------------+-----------+---------------------+---------------+-----------------------
--------+----------------+-----------+-----------+
| date1      | date1_day | datetime1           | datetime1_day | timestamp1
| timestamp1_day | date3      | date3_day |
+------------+-----------+---------------------+---------------+-----------------------
--------+----------------+-----------+-----------+
| 2021-11-29 | 29        | 2021-11-29 00:01:00 | 29            | 2021-01-11 00:00:00.12
3456789 | 11             | 2021-11-20 | 20        |
| 2021-11-28 | 28        | 2021-11-28 00:02:00 | 28            | 2021-02-11 00:00:00.12
3456789 | 11             | 2021-11-21 | 21        |
| 2021-11-27 | 27        | 2021-11-27 00:03:00 | 27            | 2021-03-11 00:00:00.12
3456789 | 11             | 2021-11-22 | 22        |
| 2021-11-26 | 26        | 2021-11-26 00:04:00 | 26            | 2021-04-11 00:00:00.12
3456789 | 11             | 2021-11-23 | 23        |
| 2021-11-25 | 25        | 2021-11-25 00:05:00 | 25            | 2021-05-11 00:00:00.12
3456789 | 11             | 2021-11-24 | 24        |
| 2021-11-24 | 24        | 2021-11-24 00:06:00 | 24            | 2021-06-11 00:00:00.12
3456789 | 11             | 2021-11-25 | 25        |
| 2021-11-23 | 23        | 2021-11-23 00:07:00 | 23            | 2021-07-11 00:00:00.12
3456789 | 11             | 2021-11-26 | 26        |
| 2021-11-22 | 22        | 2021-11-22 00:08:00 | 22            | 2021-08-11 00:00:00.12
3456789 | 11             | 2021-11-27 | 27        |
| 2021-11-21 | 21        | 2021-11-21 00:09:00 | 21            | 2021-09-11 00:00:00.12
3456789 | 11             | 2021-11-28 | 28        |
| 2021-11-20 | 20        | 2021-11-20 00:10:00 | 20            | 2021-10-11 00:00:00.12
3456789 | 11             | 2021-11-29 | 29        |
+------------+-----------+---------------------+---------------+-----------------------
--------+----------------+-----------+-----------+
```

DAYOFMONTH

- Syntax

```
int dayofmonth(datetime|timestamp|date|string <date>)
```

- Description

  Returns the day component of a date value. This function is an additional function of MaxCompute V2.0.

- Parameters

date: required. A date value of the DATETIME, TIMESTAMP, DATE, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, DATE, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The return value is 1.
    select dayofmonth('2014-09-01');
    -- The return value is null.
    select dayofmonth('20140901');
    -- The return value is null.
    select dayofmonth(null);
    ```

  - Examples of table data

    Obtain the day components of date values in the date1, datetime1, timestamp1, and date3 columns. Data in Sample data is used in this example. Sample statement:

    ```
    -- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
    her with the SQL statement.
    set odps.sql.type.system.odps2=true;
    select date1, dayofmonth(date1) as date1_dayofmonth, datetime1, dayofmonth(datetime1) a
    s datetime1_dayofmonth, timestamp1, dayofmonth(timestamp1) as timestamp1_dayofmonth, da
    te3, dayofmonth(date3) as date3_dayofmonth from mf_date_fun_t;
    ```

    The following result is returned:

```
+------------+------------------+---------------------+----------------------+---------
---------------------+-----------------------+------------+------------------+
| date1      | date1_dayofmonth | datetime1           | datetime1_dayofmonth | timestam
p1                    | timestamp1_dayofmonth | date3      | date3_dayofmonth |
+------------+------------------+---------------------+----------------------+---------
---------------------+-----------------------+------------+------------------+
| 2021-11-29 | 29               | 2021-11-29 00:01:00 | 29                   | 2021-01-
11 00:00:00.123456789 | 11                    | 2021-11-20 | 20                |
| 2021-11-28 | 28               | 2021-11-28 00:02:00 | 28                   | 2021-02-
11 00:00:00.123456789 | 11                    | 2021-11-21 | 21                |
| 2021-11-27 | 27               | 2021-11-27 00:03:00 | 27                   | 2021-03-
11 00:00:00.123456789 | 11                    | 2021-11-22 | 22                |
| 2021-11-26 | 26               | 2021-11-26 00:04:00 | 26                   | 2021-04-
11 00:00:00.123456789 | 11                    | 2021-11-23 | 23                |
| 2021-11-25 | 25               | 2021-11-25 00:05:00 | 25                   | 2021-05-
11 00:00:00.123456789 | 11                    | 2021-11-24 | 24                |
| 2021-11-24 | 24               | 2021-11-24 00:06:00 | 24                   | 2021-06-
11 00:00:00.123456789 | 11                    | 2021-11-25 | 25                |
| 2021-11-23 | 23               | 2021-11-23 00:07:00 | 23                   | 2021-07-
11 00:00:00.123456789 | 11                    | 2021-11-26 | 26                |
| 2021-11-22 | 22               | 2021-11-22 00:08:00 | 22                   | 2021-08-
11 00:00:00.123456789 | 11                    | 2021-11-27 | 27                |
| 2021-11-21 | 21               | 2021-11-21 00:09:00 | 21                   | 2021-09-
11 00:00:00.123456789 | 11                    | 2021-11-28 | 28                |
| 2021-11-20 | 20               | 2021-11-20 00:10:00 | 20                   | 2021-10-
11 00:00:00.123456789 | 11                    | 2021-11-29 | 29                |
+------------+------------------+---------------------+----------------------+---------
---------------------+-----------------------+------------+------------------+
```

EXTRACT

- Syntax

```
int extract(<datepart> from date|datetime|timestamp <date>)
```

- Description

  Extracts the date component specified by datepart from a date value specified by date. This function is an additional function of MaxCompute V2.0.

- Parameters
  - datepart: required. A value that can be set to YEAR, MONTH, DAY, HOUR, or MINUTE.
  - date: required. A date value of the DATE, DATETIME, TIMESTAMP, or STRING type. The value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:
  - If the value of datepart is not YEAR, MONTH, DAY, HOUR, or MINUTE, an error is returned.
  - If the value of datepart is null, an error is returned.

- If the value of date is not of the DATE, DATETIME, TIMESTAMP, or STRING type or is null, null is returned.

- Examples
  - Examples of static data

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select  extract(year from '2019-05-01 11:21:00') year
         ,extract(month from '2019-05-01 11:21:00') month
         ,extract(day from '2019-05-01 11:21:00') day
         ,extract(hour from '2019-05-01 11:21:00') hour
         ,extract(minute from '2019-05-01 11:21:00') minute;
-- The following result is returned:
+------+-------+------+------+--------+
| year | month | day  | hour | minute |
+------+-------+------+------+--------+
| 2019 | 5     | 1    | 11   | 21     |
+------+-------+------+------+--------+
-- The return value is null.
select  extract(year from null);
```

  - Examples of table data

    Extract the specified date components from date values in the timestamp1 column. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select timestamp1, extract(year from timestamp1) year, timestamp2, extract(month from t
imestamp2) month from mf_date_fun_t;
```

    The following result is returned:

```
+-------------------------------+------+-------------------------------+-------+
| timestamp1                    | year | timestamp2                    | month |
+-------------------------------+------+-------------------------------+-------+
| 2021-01-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-02-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-03-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-04-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-05-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-06-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-07-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-08-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-09-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
| 2021-10-11 00:00:00.123456789 | 2021 | 2021-10-11 00:00:00.123456789 | 10    |
+-------------------------------+------+-------------------------------+-------+
```

FROM_UTC_TIMESTAMP

- Syntax

```
datetime|timestamp from_utc_timestamp({any primitive type}*, string <timezone>)
```

- Description

  Converts a UTC timestamp into a timestamp in a specified time zone. This function is an additional function of MaxCompute V2.0.

- Parameters
  - {any primitive type}*: required. A timestamp of the TIMESTAMP, DATETIME, TINYINT, SMALLINT, INT, or BIGINT type. If the value is of the TINYINT, SMALLINT, INT, or BIGINT type, the time unit is accurate to the millisecond.
  - timezone: required. The new time zone.

    > **Note** You can search for the time zone list by using a search engine.

- Return value

  A value of the DATETIME or TIMESTAMP type is returned. The return value is in the `yyyy-mm-dd hh:mi:ss` or `yyyy-mm-dd hh:mi:ss.ff3` format. The return value varies based on the following rules:
  - If the value of {any primitive type}* is not of the TIMESTAMP, DATETIME, TINYINT, SMALLINT, INT, or BIGINT type, an error is returned.
  - If the value of {any primitive type}* is null, an error is returned.
  - If the value of timezone is null, null is returned.

- Examples
  - Examples of static data

```
-- The time unit of the input value is accurate to the millisecond and the return value
is 2017-08-01 04:24:00.0.
select from_utc_timestamp(1501557840000, 'PST');
-- The return value is 1970-01-30 08:00:00.0.
select from_utc_timestamp('1970-01-30 16:00:00','PST');
-- The return value is 1970-01-29 16:00:00.0.
select from_utc_timestamp('1970-01-30','PST');
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement. The return value is 2011-12-25 17:00:00:00.123.
set odps.sql.type.system.odps2=true;
select from_utc_timestamp(timestamp '2011-12-25 09:00:00.123456', 'Asia/Shanghai');
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement. The return value is 2011-12-25 01:55:00.0.
set odps.sql.type.system.odps2=true;
select from_utc_timestamp(timestamp '2011-12-25 06:55:00', 'America/Toronto');
-- The return value is null.
select from_utc_timestamp('1970-01-30',null);
```

- Examples of table data

  Convert date values in the datetime1 and timestamp1 columns into timestamps in a specified time zone. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select datetime1, from_utc_timestamp(datetime1,'PST') pst, timestamp1, from_utc_timesta
mp(timestamp1,'Asia/Shanghai') asia from mf_date_fun_t;
```

  The following result is returned:

```
+---------------------+---------------------+-------------------------------+----------
---------------------+
| datetime1           | pst                 | timestamp1                    | asia
|
+---------------------+---------------------+-------------------------------+----------
---------------------+
| 2021-11-29 00:01:00 | 2021-11-28 16:01:00 | 2021-01-11 00:00:00.123456789 | 2021-01-1
1 08:00:00.123456789 |
| 2021-11-28 00:02:00 | 2021-11-27 16:02:00 | 2021-02-11 00:00:00.123456789 | 2021-02-1
1 08:00:00.123456789 |
| 2021-11-27 00:03:00 | 2021-11-26 16:03:00 | 2021-03-11 00:00:00.123456789 | 2021-03-1
1 08:00:00.123456789 |
| 2021-11-26 00:04:00 | 2021-11-25 16:04:00 | 2021-04-11 00:00:00.123456789 | 2021-04-1
1 08:00:00.123456789 |
| 2021-11-25 00:05:00 | 2021-11-24 16:05:00 | 2021-05-11 00:00:00.123456789 | 2021-05-1
1 08:00:00.123456789 |
| 2021-11-24 00:06:00 | 2021-11-23 16:06:00 | 2021-06-11 00:00:00.123456789 | 2021-06-1
1 08:00:00.123456789 |
| 2021-11-23 00:07:00 | 2021-11-22 16:07:00 | 2021-07-11 00:00:00.123456789 | 2021-07-1
1 08:00:00.123456789 |
| 2021-11-22 00:08:00 | 2021-11-21 16:08:00 | 2021-08-11 00:00:00.123456789 | 2021-08-1
1 08:00:00.123456789 |
| 2021-11-21 00:09:00 | 2021-11-20 16:09:00 | 2021-09-11 00:00:00.123456789 | 2021-09-1
1 08:00:00.123456789 |
| 2021-11-20 00:10:00 | 2021-11-19 16:10:00 | 2021-10-11 00:00:00.123456789 | 2021-10-1
1 08:00:00.123456789 |
+---------------------+---------------------+-------------------------------+----------
---------------------+
```

HOUR

- Syntax

```
int hour(datetime|timestamp|string <date>)
```

- Description

  Returns the hour component of a date value.

- Parameters

date: required. A date value of the DATETIME, TIMESTAMP, or STRING type. The date value is in the `yyyy-mm-dd hh:mi:ss` or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings. This function is an additional function of MaxCompute V2.0.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The return value is 12.
    select hour('2014-09-01 12:00:00');
    -- The return value is 12.
    select hour('12:00:00');
    -- The return value is null.
    select hour('20140901120000');
    -- The return value is null.
    select hour(null);
    ```

  - Examples of table data

    Obtain the hour components of date values in the datetime1 and timestamp1 columns. Data in Sample data is used in this example. Sample statement:

    ```
    -- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
    her with the SQL statement.
    set odps.sql.type.system.odps2=true;
    select datetime1, hour(datetime1) as datetime1_hour, timestamp1, hour(timestamp1) as ti
    mestamp1_hour from mf_date_fun_t;
    ```

    The following result is returned:

```
+---------------------+---------------+------------------------------+---------------
--+
| datetime1           | datetime1_hour | timestamp1                   | timestamp1_hou
r |
+---------------------+---------------+------------------------------+---------------
--+
| 2021-11-29 00:01:00 | 0              | 2021-01-11 00:00:00.123456789 | 0
|
| 2021-11-28 00:02:00 | 0              | 2021-02-11 00:00:00.123456789 | 0
|
| 2021-11-27 00:03:00 | 0              | 2021-03-11 00:00:00.123456789 | 0
|
| 2021-11-26 00:04:00 | 0              | 2021-04-11 00:00:00.123456789 | 0
|
| 2021-11-25 00:05:00 | 0              | 2021-05-11 00:00:00.123456789 | 0
|
| 2021-11-24 00:06:00 | 0              | 2021-06-11 00:00:00.123456789 | 0
|
| 2021-11-23 00:07:00 | 0              | 2021-07-11 00:00:00.123456789 | 0
|
| 2021-11-22 00:08:00 | 0              | 2021-08-11 00:00:00.123456789 | 0
|
| 2021-11-21 00:09:00 | 0              | 2021-09-11 00:00:00.123456789 | 0
|
| 2021-11-20 00:10:00 | 0              | 2021-10-11 00:00:00.123456789 | 0
|
+---------------------+---------------+------------------------------+---------------
--+
```

LAST_DAY

- Syntax

```
string last_day(date|datetime|timestamp|string <date>)
```

- Description

  Returns the last day of the month in which a date value falls. This function is an additional function of MaxCompute V2.0.

- Parameters

  date: required. A date value of the DATE, DATETIME, TIMESTAMP, or STRING type. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the STRING type is returned. The return value is in the `yyyy-mm-dd` format. The return value varies based on the following rules:

  - If the value of date is not of the DATE, DATETIME, TIMESTAMP, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, an error is returned.

- Examples

- Examples of static data

```
-- The return value is 2017-03-31.
select last_day('2017-03-04');
-- The return value is 2017-07-31.
select last_day('2017-07-04 11:40:00');
-- The return value is null.
select last_day('20170304');
```

- Examples of table data

  Obtain the last day of the month in which each date value in the date1, datetime1, timestamp1, and date3 columns falls. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, last_day(date1) as date1_lastday, datetime1, last_day(datetime1) as datet
ime1_lastday, timestamp1, last_day(timestamp1) as timestamp1_lastday, date3, last_day(d
ate3) as date3_lastday from mf_date_fun_t;
```

  The following result is returned:

```
+------------+---------------+---------------------+-------------------+---------------
----------------+-------------------+------------+---------------+
| date1      | date1_lastday | datetime1           | datetime1_lastday | timestamp1
| timestamp1_lastday | date3      | date3_lastday |
+------------+---------------+---------------------+-------------------+---------------
----------------+-------------------+------------+---------------+
| 2021-11-29 | 2021-11-30    | 2021-11-29 00:01:00 | 2021-11-30        | 2021-01-11 00:
00:00.123456789 | 2021-01-31         | 2021-11-20 | 2021-11-30    |
| 2021-11-28 | 2021-11-30    | 2021-11-28 00:02:00 | 2021-11-30        | 2021-02-11 00:
00:00.123456789 | 2021-02-28         | 2021-11-21 | 2021-11-30    |
| 2021-11-27 | 2021-11-30    | 2021-11-27 00:03:00 | 2021-11-30        | 2021-03-11 00:
00:00.123456789 | 2021-03-31         | 2021-11-22 | 2021-11-30    |
| 2021-11-26 | 2021-11-30    | 2021-11-26 00:04:00 | 2021-11-30        | 2021-04-11 00:
00:00.123456789 | 2021-04-30         | 2021-11-23 | 2021-11-30    |
| 2021-11-25 | 2021-11-30    | 2021-11-25 00:05:00 | 2021-11-30        | 2021-05-11 00:
00:00.123456789 | 2021-05-31         | 2021-11-24 | 2021-11-30    |
| 2021-11-24 | 2021-11-30    | 2021-11-24 00:06:00 | 2021-11-30        | 2021-06-11 00:
00:00.123456789 | 2021-06-30         | 2021-11-25 | 2021-11-30    |
| 2021-11-23 | 2021-11-30    | 2021-11-23 00:07:00 | 2021-11-30        | 2021-07-11 00:
00:00.123456789 | 2021-07-31         | 2021-11-26 | 2021-11-30    |
| 2021-11-22 | 2021-11-30    | 2021-11-22 00:08:00 | 2021-11-30        | 2021-08-11 00:
00:00.123456789 | 2021-08-31         | 2021-11-27 | 2021-11-30    |
| 2021-11-21 | 2021-11-30    | 2021-11-21 00:09:00 | 2021-11-30        | 2021-09-11 00:
00:00.123456789 | 2021-09-30         | 2021-11-28 | 2021-11-30    |
| 2021-11-20 | 2021-11-30    | 2021-11-20 00:10:00 | 2021-11-30        | 2021-10-11 00:
00:00.123456789 | 2021-10-31         | 2021-11-29 | 2021-11-30    |
+------------+---------------+---------------------+-------------------+---------------
----------------+-------------------+------------+---------------+
```

MINUTE

- Syntax

```
int minute(datetime|timestamp|string <date>)
```

- Description

  Returns the minute component of a date value. This function is an additional function of MaxCompute V2.0.

- Parameters

  date: required. A date value of the DATETIME, TIMESTAMP, or STRING type. The date value is in the `yyyy-mm-dd hh:mi:ss` or `yyyy-mm-dd hh:mi:ss.ff3` format.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 30.
select minute('2014-09-01 12:30:00');
-- The return value is 30.
select minute('12:30:00');
-- The return value is null.
select minute('20140901120000');
-- The return value is null.
select minute(null);
```

- Examples of table data

  Obtain the minute components of date values in the datetime1 and timestamp1 columns. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select datetime1, minute(datetime1) as datetime1_minute, timestamp1, minute(timestamp1)
as timestamp1_minute from mf_date_fun_t;
```

  The following result is returned:

```
+---------------------+-----------------+-------------------------------+-------------
------+
| datetime1           | datetime1_minute | timestamp1                    | timestamp1_m
inute |
+---------------------+-----------------+-------------------------------+-------------
------+
| 2021-11-29 00:01:00 | 1                | 2021-01-11 00:00:00.123456789 | 0
|
| 2021-11-28 00:02:00 | 2                | 2021-02-11 00:00:00.123456789 | 0
|
| 2021-11-27 00:03:00 | 3                | 2021-03-11 00:00:00.123456789 | 0
|
| 2021-11-26 00:04:00 | 4                | 2021-04-11 00:00:00.123456789 | 0
|
| 2021-11-25 00:05:00 | 5                | 2021-05-11 00:00:00.123456789 | 0
|
| 2021-11-24 00:06:00 | 6                | 2021-06-11 00:00:00.123456789 | 0
|
| 2021-11-23 00:07:00 | 7                | 2021-07-11 00:00:00.123456789 | 0
|
| 2021-11-22 00:08:00 | 8                | 2021-08-11 00:00:00.123456789 | 0
|
| 2021-11-21 00:09:00 | 9                | 2021-09-11 00:00:00.123456789 | 0
|
| 2021-11-20 00:10:00 | 10               | 2021-10-11 00:00:00.123456789 | 0
|
+---------------------+-----------------+-------------------------------+-------------
------+
```

MONTH

- Syntax

```
int month(datetime|timestamp|date|string <date>)
```

- Description

  Returns the month in which a date value falls. This function is an additional function of MaxCompute V2.0.

- Parameters

date: required. A date value of the DATETIME, TIMESTAMP, DATE, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, DATE, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The return value is 9.
    select month('2014-09-01');
    -- The return value is null.
    select month('20140901');
    -- The return value is null.
    select month(null);
    ```

  - Examples of table data

    Obtain the month in which each date value in the date1, datetime1, timestamp1, and date3 columns falls. Data in Sample data is used in this example. Sample statement:

    ```
    -- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
    her with the SQL statement.
    set odps.sql.type.system.odps2=true;
    select date1, month(date1) as date1_month, datetime1, month(datetime1) as datetime1_mon
    th, timestamp1, month(timestamp1) as timestamp1_month, date3, month(date3) as date3_mon
    th from mf_date_fun_t;
    ```

    The following result is returned:

```
+------------+-------------+---------------------+-----------------+-------------------
------------+------------------+------------+-------------+
| date1      | date1_month | datetime1           | datetime1_month | timestamp1
| timestamp1_month | date3      | date3_month |
+------------+-------------+---------------------+-----------------+-------------------
------------+------------------+------------+-------------+
| 2021-11-29 | 11          | 2021-11-29 00:01:00 | 11              | 2021-01-11 00:00:0
0.123456789 | 1                | 2021-11-20 | 11          |
| 2021-11-28 | 11          | 2021-11-28 00:02:00 | 11              | 2021-02-11 00:00:0
0.123456789 | 2                | 2021-11-21 | 11          |
| 2021-11-27 | 11          | 2021-11-27 00:03:00 | 11              | 2021-03-11 00:00:0
0.123456789 | 3                | 2021-11-22 | 11          |
| 2021-11-26 | 11          | 2021-11-26 00:04:00 | 11              | 2021-04-11 00:00:0
0.123456789 | 4                | 2021-11-23 | 11          |
| 2021-11-25 | 11          | 2021-11-25 00:05:00 | 11              | 2021-05-11 00:00:0
0.123456789 | 5                | 2021-11-24 | 11          |
| 2021-11-24 | 11          | 2021-11-24 00:06:00 | 11              | 2021-06-11 00:00:0
0.123456789 | 6                | 2021-11-25 | 11          |
| 2021-11-23 | 11          | 2021-11-23 00:07:00 | 11              | 2021-07-11 00:00:0
0.123456789 | 7                | 2021-11-26 | 11          |
| 2021-11-22 | 11          | 2021-11-22 00:08:00 | 11              | 2021-08-11 00:00:0
0.123456789 | 8                | 2021-11-27 | 11          |
| 2021-11-21 | 11          | 2021-11-21 00:09:00 | 11              | 2021-09-11 00:00:0
0.123456789 | 9                | 2021-11-28 | 11          |
| 2021-11-20 | 11          | 2021-11-20 00:10:00 | 11              | 2021-10-11 00:00:0
0.123456789 | 10               | 2021-11-29 | 11          |
+------------+-------------+---------------------+-----------------+-------------------
------------+------------------+------------+-------------+
```

MONTHS_BETWEEN

- Syntax

```
double months_between(datetime|timestamp|date|string <date1>, datetime|timestamp|date|str
ing <date2>)
```

- Description

  Returns the number of months between date1 and date2. This function is an additional function of MaxCompute V2.0.

- Parameters

  date1 and date2: required. Values of the DATETIME, TIMESTAMP, DATE, or STRING type. The input values are in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the input values are of the STRING type, the values must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the DOUBLE type is returned. The return value varies based on the following rules:

  - If date1 is later than date2, a positive value is returned. If date2 is later than date1, a negative value is returned.

- If date1 and date2 separately correspond to the last days of two months, the return value is an integer that represents the number of months. Otherwise, the return value is calculated by using the following formula: (date1 - date2)/31.
- If the value of date1 or date2 is null, null is returned.

- Examples
  - Examples of static data

```
-- The return value is 3.9495967741935485.
select months_between('1997-02-28 10:30:00', '1996-10-30');
-- The return value is -3.9495967741935485.
select months_between('1996-10-30','1997-02-28 10:30:00' );
-- The return value is -3.0.
select months_between('1996-09-30','1996-12-31');
-- The return value is null.
select months_between('1996-09-30',null);
```

  - Examples of table data

    Calculate the number of months between date values in the timestamp1 and timestamp2 columns. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select timestamp1, timestamp2, months_between(timestamp1, timestamp2) from mf_date_fun_
t;
```

    The following result is returned:

```
+-------------------------------+-------------------------------+------------+
| timestamp1                    | timestamp2                    | _c2        |
+-------------------------------+-------------------------------+------------+
| 2021-01-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -9.0       |
| 2021-02-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -8.0       |
| 2021-03-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -7.0       |
| 2021-04-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -6.0       |
| 2021-05-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -5.0       |
| 2021-06-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -4.0       |
| 2021-07-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -3.0       |
| 2021-08-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -2.0       |
| 2021-09-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | -1.0       |
| 2021-10-11 00:00:00.123456789 | 2021-10-11 00:00:00.123456789 | 0.0        |
+------------+------------+------------+
```

NEXT_DAY

- Syntax

```
string next_day(timestamp|date|datetime|string <startdate>, string <week>)
```

- Description

Returns the date of the first weekday that is later than startdate and matches the week value. The date of the specified weekday in the next week is returned. This function is an additional function of MaxCompute V2.0.

- Parameters
  - startdate: required. A value of the TIMESTAMP, DATE, DATETIME, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.
  - week: required. A value of the STRING type. The value of this parameter can be the first two or three letters of a weekday or the full name of a weekday, such as MO, TUE, or FRIDAY.
- Return value

  A value of the STRING type is returned. The return value is in the `yyyy-mm-dd` format. The return value varies based on the following rules:
  - If the value of date is not of the TIMESTAMP, DATE, DATETIME, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, an error is returned.
  - If the value of week is null, null is returned.
- Examples
  - Examples of static data

```
-- The return value is 2017-08-08.
select next_day('2017-08-01','TU');
-- The return value is 2017-08-08.
select next_day('2017-08-01 23:34:00','TU');
-- The return value is null.
select next_day('20170801','TU');
-- The return value is null.
select next_day('2017-08-01 23:34:00',null);
```

- Examples of table data

  Obtain the date of the day in the next week to which each date value in the date1, datetime1, timestamp1, and date3 columns corresponds. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, next_day(date1,'MON') as date1_next_day, datetime1, next_day(datetime1,'T
UE') as datetime1_next_day, timestamp1, next_day(timestamp1,'WED') as timestamp1_next_d
ay, date3, next_day(date3,'THU') as date3_next_day from mf_date_fun_t;
```

  The following result is returned:

```
+------------+----------------+---------------------+--------------------+-------------
------------------+---------------------+------------+----------------+
| date1      | date1_next_day | datetime1           | datetime1_next_day | timestamp1
| timestamp1_next_day | date3      | date3_next_day |
+------------+----------------+---------------------+--------------------+-------------
------------------+---------------------+------------+----------------+
| 2021-11-29 | 2021-12-06     | 2021-11-29 00:01:00 | 2021-11-30         | 2021-01-11 0
0:00:00.123456789 | 2021-01-13          | 2021-11-20 | 2021-11-25     |
| 2021-11-28 | 2021-11-29     | 2021-11-28 00:02:00 | 2021-11-30         | 2021-02-11 0
0:00:00.123456789 | 2021-02-17          | 2021-11-21 | 2021-11-25     |
| 2021-11-27 | 2021-11-29     | 2021-11-27 00:03:00 | 2021-11-30         | 2021-03-11 0
0:00:00.123456789 | 2021-03-17          | 2021-11-22 | 2021-11-25     |
| 2021-11-26 | 2021-11-29     | 2021-11-26 00:04:00 | 2021-11-30         | 2021-04-11 0
0:00:00.123456789 | 2021-04-14          | 2021-11-23 | 2021-11-25     |
| 2021-11-25 | 2021-11-29     | 2021-11-25 00:05:00 | 2021-11-30         | 2021-05-11 0
0:00:00.123456789 | 2021-05-12          | 2021-11-24 | 2021-11-25     |
| 2021-11-24 | 2021-11-29     | 2021-11-24 00:06:00 | 2021-11-30         | 2021-06-11 0
0:00:00.123456789 | 2021-06-16          | 2021-11-25 | 2021-12-02     |
| 2021-11-23 | 2021-11-29     | 2021-11-23 00:07:00 | 2021-11-30         | 2021-07-11 0
0:00:00.123456789 | 2021-07-14          | 2021-11-26 | 2021-12-02     |
| 2021-11-22 | 2021-11-29     | 2021-11-22 00:08:00 | 2021-11-23         | 2021-08-11 0
0:00:00.123456789 | 2021-08-18          | 2021-11-27 | 2021-12-02     |
| 2021-11-21 | 2021-11-22     | 2021-11-21 00:09:00 | 2021-11-23         | 2021-09-11 0
0:00:00.123456789 | 2021-09-15          | 2021-11-28 | 2021-12-02     |
| 2021-11-20 | 2021-11-22     | 2021-11-20 00:10:00 | 2021-11-23         | 2021-10-11 0
0:00:00.123456789 | 2021-10-13          | 2021-11-29 | 2021-12-02     |
+------------+----------------+---------------------+--------------------+-------------
------------------+---------------------+------------+----------------+
```

QUARTER

- Syntax

```
int quarter (datetime|timestamp|date|string <date>)
```

- Description

  Returns the quarter in which a date value falls. Valid values: 1 to 4. This function is an additional function of MaxCompute V2.0.

- Parameters

date: required. A date value of the DATETIME, TIMESTAMP, DATE, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, DATE, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The return value is 4.
    select quarter('1970-11-12 10:00:00');
    -- The return value is 4.
    select quarter('1970-11-12');
    -- The return value is null.
    select quarter(null);
    ```

  - Examples of table data

    Obtain the quarter in which each date value in the date1, datetime1, timestamp1, and date3 columns falls. Data in Sample data is used in this example. Sample statement:

    ```
    -- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
    her with the SQL statement.
    set odps.sql.type.system.odps2=true;
    select date1, quarter(date1) as date1_quarter, datetime1, quarter(datetime1) as datetim
    e1_quarter, timestamp1, quarter(timestamp1) as timestamp1_quarter, date3, quarter(date3
    ) as date3_quarter from mf_date_fun_t;
    ```

    The following result is returned:

```
+------------+---------------+---------------------+-------------------+---------------
---------------+-------------------+------------+---------------+
| date1      | date1_quarter | datetime1           | datetime1_quarter | timestamp1
| timestamp1_quarter | date3      | date3_quarter |
+------------+---------------+---------------------+-------------------+---------------
---------------+-------------------+------------+---------------+
| 2021-11-29 | 4             | 2021-11-29 00:01:00 | 4                 | 2021-01-11 00:
00:00.123456789 | 1                 | 2021-11-20 | 4             |
| 2021-11-28 | 4             | 2021-11-28 00:02:00 | 4                 | 2021-02-11 00:
00:00.123456789 | 1                 | 2021-11-21 | 4             |
| 2021-11-27 | 4             | 2021-11-27 00:03:00 | 4                 | 2021-03-11 00:
00:00.123456789 | 1                 | 2021-11-22 | 4             |
| 2021-11-26 | 4             | 2021-11-26 00:04:00 | 4                 | 2021-04-11 00:
00:00.123456789 | 2                 | 2021-11-23 | 4             |
| 2021-11-25 | 4             | 2021-11-25 00:05:00 | 4                 | 2021-05-11 00:
00:00.123456789 | 2                 | 2021-11-24 | 4             |
| 2021-11-24 | 4             | 2021-11-24 00:06:00 | 4                 | 2021-06-11 00:
00:00.123456789 | 2                 | 2021-11-25 | 4             |
| 2021-11-23 | 4             | 2021-11-23 00:07:00 | 4                 | 2021-07-11 00:
00:00.123456789 | 3                 | 2021-11-26 | 4             |
| 2021-11-22 | 4             | 2021-11-22 00:08:00 | 4                 | 2021-08-11 00:
00:00.123456789 | 3                 | 2021-11-27 | 4             |
| 2021-11-21 | 4             | 2021-11-21 00:09:00 | 4                 | 2021-09-11 00:
00:00.123456789 | 3                 | 2021-11-28 | 4             |
| 2021-11-20 | 4             | 2021-11-20 00:10:00 | 4                 | 2021-10-11 00:
00:00.123456789 | 4                 | 2021-11-29 | 4             |
+------------+---------------+---------------------+-------------------+---------------
---------------+-------------------+------------+---------------+
```

SECOND

- Syntax

```
int second(datetime|timestamp|string <date>)
```

- Description

  Returns the second component of a date value. This function is an additional function of MaxCompute V2.0.

- Parameters

  date: required. A date value of the DATETIME, TIMESTAMP, or STRING type. The date value is in the `yyyy-mm-dd hh:mi:ss` or `yyyy-mm-dd hh:mi:ss.ff3` format.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples

- Examples of static data

```
-- The return value is 45.
select second('2014-09-01 12:30:45');
-- The return value is 45.
select second('12:30:45');
-- The return value is null.
select second('20140901123045');
-- The return value is null.
select second(null);
```

- Examples of table data

  Obtain the second components of date values in the datetime1 and timestamp1 columns. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
elect datetime1, second(datetime1) as datetime1_second, timestamp1, second(timestamp1)
as timestamp1_second from mf_date_fun_t;
```

  The following result is returned:

```
+---------------------+-----------------+-------------------------------+-------------
------+
| datetime1           | datetime1_second | timestamp1                    | timestamp1_s
econd |
+---------------------+-----------------+-------------------------------+-------------
------+
| 2021-11-29 00:01:00 | 0                | 2021-01-11 00:00:00.123456789 | 0
|
| 2021-11-28 00:02:00 | 0                | 2021-02-11 00:00:00.123456789 | 0
|
| 2021-11-27 00:03:00 | 0                | 2021-03-11 00:00:00.123456789 | 0
|
| 2021-11-26 00:04:00 | 0                | 2021-04-11 00:00:00.123456789 | 0
|
| 2021-11-25 00:05:00 | 0                | 2021-05-11 00:00:00.123456789 | 0
|
| 2021-11-24 00:06:00 | 0                | 2021-06-11 00:00:00.123456789 | 0
|
| 2021-11-23 00:07:00 | 0                | 2021-07-11 00:00:00.123456789 | 0
|
| 2021-11-22 00:08:00 | 0                | 2021-08-11 00:00:00.123456789 | 0
|
| 2021-11-21 00:09:00 | 0                | 2021-09-11 00:00:00.123456789 | 0
|
| 2021-11-20 00:10:00 | 0                | 2021-10-11 00:00:00.123456789 | 0
|
+---------------------+-----------------+-------------------------------+-------------
------+
```

TO_MILLIS

- Syntax

```
bigint to_millis(datetime|timestamp <date>);
```

- Description

  Converts a date value into a UNIX timestamp that is accurate to the millisecond. This function is an additional function of MaxCompute V2.0.

- Parameters

  date: required. A date value of the DATETIME or TIMESTAMP type. The date value is in the `yyyy-mm-dd hh:mi:ss` or `yyyy-mm-dd hh:mi:ss.ff3` format.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME or TIMESTAMP type, an error is returned.
  - If the value of date is null, an error is returned.

- Examples
  - Examples of static data

```
-- The return value is 1617174900000.
select to_millis(datetime '2021-03-31 15:15:00');
-- The return value is 1617174900000.
set odps.sql.type.system.odps2=true;
select to_millis(timestamp '2021-03-31 15:15:00');
```

  - Examples of table data

    Convert date values in the datetime1 and timestamp1 columns into UNIX timestamps that are accurate to the millisecond. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select datetime1, to_millis(datetime1) as datetime1_to_millis, timestamp1, to_millis(ti
mestamp1) as timestamp1_to_millis from mf_date_fun_t;
```

    The following result is returned:

```
+---------------------+---------------------+-------------------------------+----------
------------+
| datetime1           | datetime1_to_millis | timestamp1                    | timestamp
1_to_millis |
+---------------------+---------------------+-------------------------------+----------
------------+
| 2021-11-29 00:01:00 | 1638115260000       | 2021-01-11 00:00:00.123456789 | 161029440
0123        |
| 2021-11-28 00:02:00 | 1638028920000       | 2021-02-11 00:00:00.123456789 | 161297280
0123        |
| 2021-11-27 00:03:00 | 1637942580000       | 2021-03-11 00:00:00.123456789 | 161539200
0123        |
| 2021-11-26 00:04:00 | 1637856240000       | 2021-04-11 00:00:00.123456789 | 161807040
0123        |
| 2021-11-25 00:05:00 | 1637769900000       | 2021-05-11 00:00:00.123456789 | 162066240
0123        |
| 2021-11-24 00:06:00 | 1637683560000       | 2021-06-11 00:00:00.123456789 | 162334080
0123        |
| 2021-11-23 00:07:00 | 1637597220000       | 2021-07-11 00:00:00.123456789 | 162593280
0123        |
| 2021-11-22 00:08:00 | 1637510880000       | 2021-08-11 00:00:00.123456789 | 162861120
0123        |
| 2021-11-21 00:09:00 | 1637424540000       | 2021-09-11 00:00:00.123456789 | 163128960
0123        |
| 2021-11-20 00:10:00 | 1637338200000       | 2021-10-11 00:00:00.123456789 | 163388160
0123        |
+---------------------+---------------------+-------------------------------+----------
------------+
```

YEAR

- Syntax

```
int year(datetime|timestamp|date|string <date>)
```

- Description

  Returns the year in which a date value falls. This function is an additional function of MaxCompute V2.0.

- Parameters

  date: required. A date value of the DATETIME, TIMESTAMP, DATE, or STRING type. The input value is in the `yyyy-mm-dd`, `yyyy-mm-dd hh:mi:ss`, or `yyyy-mm-dd hh:mi:ss.ff3` format. If the value is of the STRING type, the value must include at least the `yyyy-mm-dd` part and must not contain extra strings.

- Return value

  A value of the INT type is returned. The return value varies based on the following rules:

  - If the value of date is not of the DATETIME, TIMESTAMP, DATE, or STRING type or the format does not meet the requirements, null is returned.
  - If the value of date is null, null is returned.

- Examples

- Examples of static data

```
-- The return value is 1970.
select year('1970-01-01 12:30:00');
-- The return value is 1970.
select year('1970-01-01');
-- The return value is 70.
select year('70-01-01');
-- The return value is null.
select year('1970/03/09');
-- The return value is null.
select year(null);
```

- Examples of table data

  Obtain the year in which each date value in the date1, datetime1, timestamp1, and date3 columns falls. Data in Sample data is used in this example. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit the following SET command toget
her with the SQL statement.
set odps.sql.type.system.odps2=true;
select date1, year(date1) as date1_year, datetime1, year(datetime1) as datetime1_year,
timestamp1, year(timestamp1) as timestamp1_year, date3, year(date3) as date3_year from
mf_date_fun_t;
```

  The following result is returned:

```
+------------+------------+---------------------+----------------+---------------------
----------+----------------+------------+------------+
| date1      | date1_year | datetime1           | datetime1_year | timestamp1
| timestamp1_year | date3      | date3_year |
+------------+------------+---------------------+----------------+---------------------
----------+----------------+------------+------------+
| 2021-11-29 | 2021       | 2021-11-29 00:01:00 | 2021           | 2021-01-11 00:00:00.
123456789 | 2021            | 2021-11-20 | 2021       |
| 2021-11-28 | 2021       | 2021-11-28 00:02:00 | 2021           | 2021-02-11 00:00:00.
123456789 | 2021            | 2021-11-21 | 2021       |
| 2021-11-27 | 2021       | 2021-11-27 00:03:00 | 2021           | 2021-03-11 00:00:00.
123456789 | 2021            | 2021-11-22 | 2021       |
| 2021-11-26 | 2021       | 2021-11-26 00:04:00 | 2021           | 2021-04-11 00:00:00.
123456789 | 2021            | 2021-11-23 | 2021       |
| 2021-11-25 | 2021       | 2021-11-25 00:05:00 | 2021           | 2021-05-11 00:00:00.
123456789 | 2021            | 2021-11-24 | 2021       |
| 2021-11-24 | 2021       | 2021-11-24 00:06:00 | 2021           | 2021-06-11 00:00:00.
123456789 | 2021            | 2021-11-25 | 2021       |
| 2021-11-23 | 2021       | 2021-11-23 00:07:00 | 2021           | 2021-07-11 00:00:00.
123456789 | 2021            | 2021-11-26 | 2021       |
| 2021-11-22 | 2021       | 2021-11-22 00:08:00 | 2021           | 2021-08-11 00:00:00.
123456789 | 2021            | 2021-11-27 | 2021       |
| 2021-11-21 | 2021       | 2021-11-21 00:09:00 | 2021           | 2021-09-11 00:00:00.
123456789 | 2021            | 2021-11-28 | 2021       |
| 2021-11-20 | 2021       | 2021-11-20 00:10:00 | 2021           | 2021-10-11 00:00:00.
123456789 | 2021            | 2021-11-29 | 2021       |
+------------+------------+---------------------+----------------+---------------------
----------+----------------+------------+------------+
```

3.9.4. Mathematical functions

MaxCompute SQL provides common mathematical functions that you can use for development. You can select mathematical functions based on your business requirements to compute data or convert data types. This topic describes the syntax, parameters, and examples of mathematical functions supported by MaxCompute SQL.

The following table lists mathematical functions supported by MaxCompute SQL.

Function	Description
ABS	Calculates the absolute value.
ACOS	Calculates the arccosine.
ASIN	Calculates the arcsine.
ATAN	Calculates the arctangent.
CEIL	Rounds up a number and returns the nearest integer.
CONV	Converts a number from one number system to another.
COS	Calculates the cosine.
COSH	Calculates the hyperbolic cosine.
COT	Calculates the cotangent.
EXP	Calculates the exponential value.
FLOOR	Rounds down a number and returns the nearest integer.
LN	Calculates the natural logarithm.
LOG	Calculates the logarithm.
POW	Calculates the nth power of a value.
RAND	Returns a random number.
ROUND	Returns a value rounded to the specified decimal place.
SIN	Calculates the sine.
SINH	Calculates the hyperbolic sine.
SQRT	Calculates the square root.
TAN	Calculates the tangent.
TANH	Calculates the hyperbolic tangent.
TRUNC	Truncates the input value to the specified decimal place.
BIN	Calculates the binary code.
CBRT	Calculates the cube root.
CORR	Calculates the Pearson correlation coefficient.
DEGREES	Converts a radian value into a degree.

Function	Description
E	Calculates the value of e.
FACTORIAL	Calculates the factorial.
FORMAT_NUMBER	Converts a number into a string in the specified format.
HEX	Converts an integer or a string into a hexadecimal number.
LOG2	Calculates the logarithm of a number with the base number of 2.
LOG10	Calculates the logarithm of a number with the base number of 10.
PI	Calculates the value of π.
RADIANS	Converts a degree into a radian value.
SIGN	Returns the sign of the input value.
SHIFTLEFT	Shifts a value left by a specific number of places.
SHIFTRIGHT	Shifts a value right by a specific number of places.
SHIFTRIGHTUNSIGNED	Shifts an unsigned value right by a specific number of places.
UNHEX	Converts a hexadecimal string into a string.
WIDTH_BUCKET	Returns the ID of the bucket into which the value of a specific expression falls.

Usage notes

MaxCompute V2.0 provides additional date functions. If the functions that you use involve new data types, you must run one of the following SET commands to enable the MaxCompute V2.0 data type edition. The new data types include TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY.

- Session level: To use the MaxCompute V2.0 data type edition, you must add `set odps.sql.type.system.odps2=true;` before the SQL statement that you want to execute, and commit and execute them together.
- Project level: The project owner can run the following command to enable the MaxCompute V2.0 data type edition for the project based on the project requirements. The configuration takes effect after 10 to 15 minutes.

  ```
  setproject odps.sql.type.system.odps2=true;
  ```

  For more information about `setproject`, see Project operations. For more information about the precautions that you must take when you enable the MaxCompute V2.0 data type edition at the project level, see Data type editions.

Sample data

This section provides sample source data for you to understand how to use mathematical functions. In this topic, a table named mf_math_fun_t is created and data is inserted into the table. Sample statements:

```
create table if not exists mf_math_fun_t(
     int_data     int,
     bigint_data  bigint,
     double_data  double,
     decimal_data decimal,
     float_data   float,
     string_data  string
    );
insert into mf_math_fun_t values
(null, -10, 0.525, 0.525BD, cast(0.525 as float), '10'),
(-20, null, -0.1, -0.1BD, cast(-0.1 as float), '-10'),
(0, -1, null, 20.45BD, cast(-1 as float), '30'),
(-40, 4, 0.89, null, cast(0.89 as float), '-30'),
(5, -50, -1, -1BD, null, '50'),
(-60, 6, 1.5, 1.5BD, cast(1.5 as float), '-50'),
(-1, -70, -7.5, -7.5BD, cast(-7.5 as float),null ),
(-80, 1, -10.2, -10.2BD, cast(-10.2 as float), '-1' ),
(9, -90, 2.58, 2.58BD, cast(2.58 as float), '0'),
(-100, 10, -5.8, -5.8BD, cast(-5.8 as float), '-90');
```

ABS

- Syntax

  ```
  bigint|double|decimal abs(<number>)
  ```

- Description

  This function calculates the absolute value of number.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, or DECIMAL type. If the input value is of the STRING type, it is implicitly converted into a value of the DOUBLE type before calculation.

  > **Note** If the input value is of the BIGINT type and is greater than the maximum value of the BIGINT type, a value of the DOUBLE type is returned. However, the precision may be lost.

- Return value

  The type of the return value depends on the type of the input parameter. The return value varies based on the following rules:

  - If number is of the DOUBLE, BIGINT, or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING type, a value of the DOUBLE type is returned.
  - If number is set to null, null is returned.

- Examples

- Examples of static data

```
-- The value null is returned.
select abs(null);
-- The value 1 is returned.
select abs(-1);
-- The value 1.2 is returned.
select abs(-1.2);
-- The value 2.0 is returned.
select abs("-2");
-- The value 1.2232083745629837E32 is returned.
select abs(122320837456298376592387456923748);
-- Calculate the absolute value of the id field in tbl1. The following example shows th
e usage of an ABS function in SQL statements. Other built-in functions, except window f
unctions and aggregate functions, are used in a similar way.
select abs(id) from tbl1;
```

- Example of table data

  Calculate the absolute value based on the Sample data. Sample statement:

```
select abs(bigint_data) as bigint_new, abs(double_data) as double_new, abs(decimal_data
) as decimal_new, abs(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+------------+------------+-------------+------------+
| bigint_new | double_new | decimal_new | string_new |
+------------+------------+-------------+------------+
| 10         | 0.525      | 0.525       | 10.0       |
| NULL       | 0.1        | 0.1         | 10.0       |
| 1          | NULL       | 20.45       | 30.0       |
| 4          | 0.89       | NULL        | 30.0       |
| 50         | 1.0        | 1           | 50.0       |
| 6          | 1.5        | 1.5         | 50.0       |
| 70         | 7.5        | 7.5         | NULL       |
| 1          | 10.2       | 10.2        | 1.0        |
| 90         | 2.58       | 2.58        | 0.0        |
| 10         | 5.8        | 5.8         | 90.0       |
+------------+------------+-------------+------------+
```

ACOS

- Syntax

```
double|decimal acos(<number>)
```

- Description

  This function calculates the arccosine of number.

- Parameters

  number: required. The value ranges from -1 to 1. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  The type of the return value depends on the type of the input parameter. The value ranges from 0 to π. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number does not fall into the range from -1 to 1, null is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 0.5155940062460905 is returned.
select acos("0.87");
-- The value 1.5707963267948966 is returned.
select acos(0);
-- The value null is returned.
select acos(null);
```

  - Example of table data

    Calculate the arccosine based on the Sample data. Sample statement:

```
select acos(bigint_data) as bigint_new, acos(double_data) as double_new, acos(decimal_d
ata) as decimal_new, acos(string_data) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+-------------------+--------------------+--------------------+----------------------+
| bigint_new        | double_new         | decimal_new        | string_new           |
+-------------------+--------------------+--------------------+----------------------+
| NULL              | 1.0180812136981134 | 1.0180812136981134 | NULL                 |
| NULL              | 1.6709637479564565 | 1.6709637479564565 | NULL                 |
| 3.141592653589793 | NULL               | NULL               | NULL                 |
| NULL              | 0.4734511572720662 | NULL               | NULL                 |
| NULL              | 3.141592653589793  | 3.141592653589793  | NULL                 |
| NULL              | NULL               | NULL               | NULL                 |
| NULL              | NULL               | NULL               | NULL                 |
| 0.0               | NULL               | NULL               | 3.141592653589793    |
| NULL              | NULL               | NULL               | 1.5707963267948966   |
| NULL              | NULL               | NULL               | NULL                 |
+-------------------+--------------------+--------------------+----------------------+
```

ASIN

- Syntax

```
double|decimal asin(<number>)
```

- Description

  This function calculates the arcsine of number.

- Parameters

number: required. The value ranges from -1 to 1. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  The type of the return value depends on the type of the input parameter. The value ranges from -π/2 to π/2. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number does not fall into the range from -1 to 1, null is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value 1.5707963267948966 is returned.
    select asin(1);
    -- The value -1.5707963267948966 is returned.
    select asin(-1);
    -- The value null is returned.
    select asin(null);
    ```

  - Example of table data

    Calculate the arcsine based on the Sample data. Sample statement:

    ```
    select asin(bigint_data) as bigint_new, asin(double_data) as double_new, asin(decimal_d
    ata) as decimal_new, asin(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+--------------------+--------------------+--------------------+--------------------
+
| bigint_new         | double_new         | decimal_new        | string_new
|
+--------------------+--------------------+--------------------+--------------------
+
| NULL               | 0.5527151130967832 | 0.5527151130967832 | NULL
|
| NULL               | -0.1001674211615598| -0.1001674211615598| NULL
|
| -1.5707963267948966| NULL               | NULL               | NULL
|
| NULL               | 1.0973451695228305 | NULL               | NULL
|
| NULL               | -1.5707963267948966| -1.5707963267948966| NULL
|
| NULL               | NULL               | NULL               | NULL
|
| NULL               | NULL               | NULL               | NULL
|
| 1.5707963267948966 | NULL               | NULL               | -1.5707963267948966
|
| NULL               | NULL               | NULL               | 0.0
|
| NULL               | NULL               | NULL               | NULL
|
+--------------------+--------------------+--------------------+--------------------
+
```

ATAN

- Syntax

```
double atan(<number>)
```

- Description

  This function calculates the arctangent of number.

- Parameters

  number: required. The value is of the DOUBLE type. If the input value is of the STRING, BIGINT, or DECIMAL type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE type is returned. The value ranges from -π/2 to π/2. If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value 0.7853981633974483 is returned.
select atan(1);
-- The value -0.7853981633974483 is returned.
select atan(-1);
-- The value null is returned.
select atan(null);
```

- Example of table data

  Calculate the arctangent based on the Sample data. Sample statement:

```
select atan(bigint_data) as bigint_new, atan(double_data) as double_new, atan(decimal_d
ata) as decimal_new, atan(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+--------------------+--------------------+--------------------+------------------
---+
| bigint_new         | double_new         | decimal_new        | string_new
|
+--------------------+--------------------+--------------------+------------------
---+
| -1.4711276743037347 | 0.483447001567199  | 0.483447001567199  | 1.471127674303734
7  |
| NULL               | -0.09966865249116204 | -0.09966865249116204 | -1.47112767430373
47 |
| -0.7853981633974483 | NULL               | 1.521935491607842  | 1.537475330916649
3  |
| 1.3258176636680326 | 0.7272626879966904 | NULL               | -1.53747533091664
93 |
| -1.550798992821746 | -0.7853981633974483 | -0.7853981633974483 | 1.550798992821746
|
| 1.4056476493802699 | 0.982793723247329  | 0.982793723247329  | -1.55079899282174
6  |
| -1.5565115842075   | -1.4382447944982226 | -1.4382447944982226 | NULL
|
| 0.7853981633974483 | -1.473069419436178 | -1.473069419436178 | -0.78539816339744
83 |
| -1.5596856728972892 | 1.2010277920014796 | 1.2010277920014796 | 0.0
|
| 1.4711276743037347 | -1.4000611153196139 | -1.4000611153196139 | -1.55968567289728
92 |
+--------------------+--------------------+--------------------+------------------
---+
```

CEIL

- Syntax

```
bigint ceil(<value>)
```

- Description

  This function rounds up value and returns the nearest integer.

- Parameters

  value: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the BIGINT type is returned. If the value of number is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 2 is returned.
select ceil(1.1);
-- The value -1 is returned.
select ceil(-1.1);
-- The value null is returned.
select ceil(null);
```

  - Example of table data

    Round up a number based on the Sample data. Sample statement:

```
select ceil(bigint_data) as bigint_new, ceil(double_data) as double_new, ceil(decimal_d
ata) as decimal_new, ceil(string_data) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+------------+------------+-------------+------------+
| bigint_new | double_new | decimal_new | string_new |
+------------+------------+-------------+------------+
| -10        | 1          | 1           | 10         |
| NULL       | 0          | 0           | -10        |
| -1         | NULL       | 21          | 30         |
| 4          | 1          | NULL        | -30        |
| -50        | -1         | -1          | 50         |
| 6          | 2          | 2           | -50        |
| -70        | -7         | -7          | NULL       |
| 1          | -10        | -10         | -1         |
| -90        | 3          | 3           | 0          |
| 10         | -5         | -5          | -90        |
+------------+------------+-------------+------------+
```

CONV

- Syntax

```
string conv(<input>, bigint <from_base>, bigint <to_base>)
```

- Description

  This function converts a number from one number system to another.

- Parameters
  - input: required. The value is the integer you want to convert, which is of the STRING type. If the input value is of the BIGINT or DOUBLE type, it is implicitly converted into a value of the STRING type before calculation.

- from_base and to_base: required. The values of these parameters are decimal numbers. The values can be 2, 8, 10, or 16. If the input value is of the STRING or DOUBLE type, it is implicitly converted into a value of the BIGINT type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of input, from_base, or to_base is null, null is returned.
  - The conversion process runs at 64-bit precision. If an overflow occurs, null is returned.
  - If the value of input is a negative value, null is returned. If the value of input is a decimal, it is converted into an integer before the conversion of number systems. The decimal part is left out.

- Examples
  - Examples of static data

```
-- The value 12 is returned.
select conv('1100', 2, 10);
-- The value C is returned.
select conv('1100', 2, 16);
-- The value 171 is returned.
select conv('ab', 16, 10);
-- The value AB is returned.
select conv('ab', 16, 16);
-- The value null is returned.
select conv('1100', null, 10);
```

  - Example of table data

    Convert a number into a binary value based on the Sample data. Sample statement:

```
select conv(bigint_data,10,2) as bigint_new, conv(double_data,10,2) as double_new, conv
(decimal_data,10,2) as decimal_new, conv(string_data,10,2) as string_new from mf_math_f
un_t;
```

    The following result is returned:

```
+------------+------------+-------------+------------+
| bigint_new | double_new | decimal_new | string_new |
+------------+------------+-------------+------------+
| NULL       | 0          | 0           | 1010       |
| NULL       | NULL       | NULL        | NULL       |
| NULL       | NULL       | 10100       | 11110      |
| 100        | 0          | NULL        | NULL       |
| NULL       | NULL       | NULL        | 110010     |
| 110        | 1          | 1           | NULL       |
| NULL       | NULL       | NULL        | NULL       |
| 1          | NULL       | NULL        | NULL       |
| NULL       | 10         | 10          | 0          |
| 1010       | NULL       | NULL        | NULL       |
+------------+------------+-------------+------------+
```

COS

- Syntax

```
double|decimal cos(<number>)
```

- Description

  This function calculates the cosine of number, which is a radian value.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value 2.6794896585028633e-8 is retuned.
    select cos(3.1415926/2);
    -- The value -0.9999999999999986 is returned.
    select cos(3.1415926);
    -- The value null is returned.
    select cos(null);
    ```

  - Example of table data

    Calculate the cosine based on the Sample data. Sample statement:

    ```
    select cos(bigint_data) as bigint_new, cos(double_data) as double_new, cos(decimal_data
    ) as decimal_new, cos(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+---------------------+--------------------+---------------------+--------------------
-+
| bigint_new          | double_new         | decimal_new         | string_new
|
+---------------------+--------------------+---------------------+--------------------
-+
| -0.8390715290764524 | 0.8653239416229412 | 0.8653239416229412  | -0.8390715290764524
|
| NULL                | 0.9950041652780258 | 0.9950041652780258  | -0.8390715290764524
|
| 0.5403023058681398  | NULL               | -0.02964340851507803 | 0.15425144988758405
|
| -0.6536436208636119 | 0.6294120265736969 | NULL                | 0.15425144988758405
|
| 0.9649660284921133  | 0.5403023058681398 | 0.5403023058681398  | 0.9649660284921133
|
| 0.960170286650366   | 0.0707372016677029 | 0.0707372016677029  | 0.9649660284921133
|
| 0.6333192030862999  | 0.3466353178350258 | 0.3466353178350258  | NULL
|
| 0.5403023058681398  | -0.7142656520272003| -0.7142656520272003 | 0.5403023058681398
|
| -0.4480736161291701 | -0.8464080412157756| -0.8464080412157756 | 1.0
|
| -0.8390715290764524 | 0.8855195169413189 | 0.8855195169413189  | -0.4480736161291701
|
+---------------------+--------------------+---------------------+--------------------
-+
```

COSH

- Syntax

```
double|decimal cosh(<number>)
```

- Description

  This function calculates the hyperbolic cosine of number.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value 2.5091784169949913 is returned.
select cosh(3.1415926/2);
-- The value null is returned.
select cosh(null);
```

- Example of table data

  Calculate the hyperbolic cosine based on the Sample data. Sample statement:

```
select cosh(bigint_data) as bigint_new, cosh(double_data) as double_new, cosh(decimal_d
ata) as decimal_new, cosh(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+----------------------+--------------------+--------------------+--------------------
--+
| bigint_new           | double_new         | decimal_new        | string_new
|
+----------------------+--------------------+--------------------+--------------------
--+
| 11013.232920103324   | 1.1410071063729532 | 1.1410071063729532 | 11013.232920103324
|
| NULL                 | 1.0050041680558035 | 1.0050041680558035 | 11013.232920103324
|
| 1.5430806348152437   | NULL               | 380445243.96844625 | 5343237290762.231
|
| 27.308232836016487   | 1.42289270202111   | NULL               | 5343237290762.231
|
| 2.592352764293536e21 | 1.5430806348152437 | 1.5430806348152437 | 2.592352764293536e2
1 |
| 201.7156361224559    | 2.352409615243247  | 2.352409615243247  | 2.592352764293536e2
1 |
| 1.2577193354595834e30 | 904.0214837702166  | 904.0214837702166  | NULL
|
| 1.5430806348152437   | 13451.593055733929 | 13451.593055733929 | 1.5430806348152437
|
| 6.102016471589204e38 | 6.636456081840602  | 6.636456081840602  | 1.0
|
| 11013.232920103324   | 165.151293732197   | 165.151293732197   | 6.102016471589204e3
8 |
+----------------------+--------------------+--------------------+--------------------
--+
```

COT

- Syntax

```
double|decimal cot(<number>)
```

- Description

  This function calculates the cotangent of number, which is a radian value.

- Parameters

number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value 2.6794896585028643E-8 is returned.
    select cot(3.1415926/2);
    -- The value null is returned.
    select cot(null);
    ```

  - Example of table data

    Calculate the cotangent based on the Sample data. Sample statement:

    ```
    select cosh(bigint_data) as bigint_new, cosh(double_data) as double_new, cosh(decimal_d
    ata) as decimal_new, cosh(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+----------------------+--------------------+--------------------+--------------------
--+
| bigint_new           | double_new         | decimal_new        | string_new
|
+----------------------+--------------------+--------------------+--------------------
--+
| 11013.232920103324   | 1.1410071063729532 | 1.1410071063729532 | 11013.232920103324
|
| NULL                 | 1.0050041680558035 | 1.0050041680558035 | 11013.232920103324
|
| 1.5430806348152437   | NULL               | 380445243.96844625 | 5343237290762.231
|
| 27.308232836016487   | 1.42289270202111   | NULL               | 5343237290762.231
|
| 2.592352764293536e21 | 1.5430806348152437 | 1.5430806348152437 | 2.592352764293536e2
1 |
| 201.7156361224559    | 2.352409615243247  | 2.352409615243247  | 2.592352764293536e2
1 |
| 1.2577193354595834e30 | 904.0214837702166  | 904.0214837702166  | NULL
|
| 1.5430806348152437   | 13451.593055733929 | 13451.593055733929 | 1.5430806348152437
|
| 6.102016471589204e38 | 6.636456081840602  | 6.636456081840602  | 1.0
|
| 11013.232920103324   | 165.151293732197   | 165.151293732197   | 6.102016471589204e3
8 |
+----------------------+--------------------+--------------------+--------------------
--+
```

EXP

- Syntax

```
double|decimal exp(<number>)
```

- Description

  This function calculates the exponential value of number.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value 4.810477252069109 is returned.
select exp(3.1415926/2);
-- The value null is returned.
select exp(null);
```

- Example of table data

  Calculate the exponential value based on the Sample data. Sample statement:

```
select exp(bigint_data) as bigint_new, exp(double_data) as double_new, exp(decimal_data
) as decimal_new, exp(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+------------------------+------------------------+------------------------+--------
-----------------+
| bigint_new             | double_new             | decimal_new            | string_
new              |
+------------------------+------------------------+------------------------+--------
-----------------+
| 0.000045399929762484854 | 1.6904588483790914     | 1.6904588483790914     | 22026.4
65794806718      |
| NULL                   | 0.9048374180359595     | 0.9048374180359595     | 0.00004
5399929762484854 |
| 0.36787944117144233    | NULL                   | 760890487.9368925      | 1068647
4581524.463      |
| 54.598150033144236     | 2.4351296512898744     | NULL                   | 9.35762
2968840175e-14   |
| 1.9287498479639178e-22 | 0.36787944117144233    | 0.36787944117144233    | 5.18470
5528587072e21    |
| 403.4287934927351      | 4.4816890703380645     | 4.4816890703380645     | 1.92874
98479639178e-22  |
| 3.975449735908647e-31  | 0.0005530843701478336  | 0.0005530843701478336  | NULL
|
| 2.718281828459045      | 0.000037170318684126734 | 0.000037170318684126734 | 0.36787
944117144233     |
| 8.194012623990515e-40  | 13.197138159658358     | 13.197138159658358     | 1.0
|
| 22026.465794806718     | 0.0030275547453758153  | 0.0030275547453758153  | 8.19401
2623990515e-40   |
+------------------------+------------------------+------------------------+--------
-----------------+
```

FLOOR

- Syntax

```
bigint floor(<number>)
```

- Description

  This function rounds down number and returns the nearest integer that is no greater than the value of number.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the BIGINT type is returned. If the value of number is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 1 is returned.
select floor(1.2);
-- The value 0 is returned.
select floor(0.1);
-- The value -2 is returned.
select floor(-1.2);
-- The value -1 is returned.
select floor(-0.1);
-- The value 0 is returned.
select floor(0.0);
-- The value 0 is returned.
select floor(-0.0);
-- The value null is returned.
select floor(null);
```

  - Example of table data

    Round down a number based on the Sample data. Sample statement:

```
select floor(bigint_data) as bigint_new, floor(double_data) as double_new, floor(decima
l_data) as decimal_new, floor(string_data) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+------------+------------+-------------+------------+
| bigint_new | double_new | decimal_new | string_new |
+------------+------------+-------------+------------+
| -10        | 0          | 0           | 10         |
| NULL       | -1         | -1          | -10        |
| -1         | NULL       | 20          | 30         |
| 4          | 0          | NULL        | -30        |
| -50        | -1         | -1          | 50         |
| 6          | 1          | 1           | -50        |
| -70        | -8         | -8          | NULL       |
| 1          | -11        | -11         | -1         |
| -90        | 2          | 2           | 0          |
| 10         | -6         | -6          | -90        |
+------------+------------+-------------+------------+
```

LN

- Syntax

```
double|decimal ln(<number>)
```

- Description

  This function calculates the natural logarithm of number.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is a negative value or 0, null is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value 1.144729868791239 is returned.
    select ln(3.1415926);
    -- The value null is returned.
    select ln(null);
    ```

  - Example of table data

    Calculate the natural logarithm value based on the Sample data. Sample statement:

    ```
    select ln(bigint_data) as bigint_new, ln(double_data) as double_new, ln(decimal_data) a
    s decimal_new, ln(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+--------------------+----------------------+--------------------+--------------------
-+
| bigint_new         | double_new           | decimal_new        | string_new
|
+--------------------+----------------------+--------------------+--------------------
-+
| NULL               | -0.6443570163905132  | -0.6443570163905132 | 2.302585092994046
|
| NULL               | NULL                 | NULL               | NULL
|
| NULL               | NULL                 | 3.017982882488811  | 3.4011973816621555
|
| 1.3862943611198906 | -0.11653381625595151 | NULL               | NULL
|
| NULL               | NULL                 | NULL               | 3.912023005428146
|
| 1.791759469228055  | 0.4054651081081644   | 0.4054651081081644 | NULL
|
| NULL               | NULL                 | NULL               | NULL
|
| 0.0                | NULL                 | NULL               | NULL
|
| NULL               | 0.9477893989335261   | 0.9477893989335261 | NULL
|
| 2.302585092994046  | NULL                 | NULL               | NULL
|
+--------------------+----------------------+--------------------+--------------------
-+
```

LOG

- Syntax

```
double log(<base>, <x>)
```

- Description

  This function calculates the logarithm of x whose base number is base.

- Parameters
  - base: required. The base number. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.
  - x: required. The value for which the logarithm is calculated. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE type is returned. The return value varies based on the following rules:
  - If the value of base or x is null, null is returned.
  - If the value of base or x is a negative value or 0, null is returned.
  - If the value of base is 1, null is returned. The value 1 causes division by zero.

- Examples
  - Examples of static data

```
-- The value 4.0 is returned.
select log(2, 16);
-- The value null is returned.
select log(2, null);
```

  - Example of table data

    Calculate the logarithm value of a column whose base number is 2 based on the Sample data. Sample statement:

```
select log(2,bigint_data) as bigint_new, log(2,double_data) as double_new, log(2,decima
l_data) as decimal_new, log(2,string_data) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+--------------------+----------------------+--------------------+--------------------+
| bigint_new         | double_new           | decimal_new        | string_new         |
+--------------------+----------------------+--------------------+--------------------+
| NULL               | -0.929610672108602   | -0.929610672108602 | 3.3219280948873626 |
| NULL               | NULL                 | NULL               | NULL               |
| NULL               | NULL                 | 4.354028938054387  | 4.906890595608519  |
| 2.0                | -0.16812275880832692 | NULL               | NULL               |
| NULL               | NULL                 | NULL               | 5.643856189774724  |
| 2.584962500721156  | 0.5849625007211562   | 0.5849625007211562 | NULL               |
| NULL               | NULL                 | NULL               | NULL               |
| 0.0                | NULL                 | NULL               | NULL               |
| NULL               | 1.3673710656485296   | 1.3673710656485296 | NULL               |
| 3.3219280948873626 | NULL                 | NULL               | NULL               |
+--------------------+----------------------+--------------------+--------------------+
```

POW

- Syntax

```
double|decimal pow(<x>, <y>)
```

- Description

  This function calculates the yth power of x, namely, `x^y`.

- Parameters
  - x: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.
  - y: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.
- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:
  - If x or y is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If x or y is of the STRING or BIGINT type, a value of the DOUBLE type is returned.

- If the value of x or y is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 65536.0 is returned.
select pow(2, 16);
-- The value null is returned.
select pow(2, null);
```

  - Example of table data

    Calculate the second power of values in a column based on the Sample data. Sample statement:

```
select pow(bigint_data, 2) as bigint_new, pow(double_data, 2) as double_new, pow(decima
l_data, 2) as decimal_new, pow(string_data, 2) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+------------+----------------------+----------------------+------------+
| bigint_new | double_new           | decimal_new          | string_new |
+------------+----------------------+----------------------+------------+
| 100.0      | 0.275625             | 0.275625             | 100.0      |
| NULL       | 0.010000000000000002 | 0.010000000000000002 | 100.0      |
| 1.0        | NULL                 | 418.2025             | 900.0      |
| 16.0       | 0.7921               | NULL                 | 900.0      |
| 2500.0     | 1.0                  | 1.0                  | 2500.0     |
| 36.0       | 2.25                 | 2.25                 | 2500.0     |
| 4900.0     | 56.25                | 56.25                | NULL       |
| 1.0        | 104.03999999999999   | 104.03999999999999   | 1.0        |
| 8100.0     | 6.6564000000000005   | 6.6564000000000005   | 0.0        |
| 100.0      | 33.64                | 33.64                | 8100.0     |
+------------+----------------------+----------------------+------------+
```

RAND

- Syntax

```
double rand(bigint <seed>)
```

- Description

  This function returns a random number of the DOUBLE type. The value ranges from 0 to 1.

- Parameters

  seed: optional. The value is of the BIGINT type. This parameter specifies the random seed that determines the starting point in generating random numbers.

  > **Note** You can use seed to determine the random number sequence. After seed is determined, the return value of this function is fixed. If the execution environment is the same and the seed value remains unchanged, the return value is the same. If you need to return different results, you must modify the seed value.

- Return value

  A value of the DOUBLE type is returned.

- Examples

```
-- The value 4.7147460303803655E-4 is returned.
select rand();
-- The value 0.7308781907032909 is returned.
select rand(1);
```

ROUND

- Syntax

```
double|decimal round(<number>[, bigint <decimal_places>])
```

- Description

  This function returns a number rounded to the specified decimal place.

- Parameters
  - number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.
  - decimal_places: optional. The value is a constant of the BIGINT type. This parameter specifies the decimal place to which the number is rounded. If this parameter is not specified, the number is rounded to the ones place. The default value is 0.

    > **Note** The value of decimal_places can be negative. A negative value indicates counting from the decimal point to the left, and the decimal part is excluded. If decimal_places exceeds the length of the integer part, 0 is returned.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:
  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If decimal_places is not of the BIGINT type, an error is returned.
  - If the value of number or decimal_places is null, null is returned.

- Examples

- Examples of static data

```
-- The value 125.0 is returned.
select round(125.315);
-- The value 125.3 is returned.
select round(125.315, 1);
-- The value 125.32 is returned.
select round(125.315, 2);
-- The value 125.315 is returned.
select round(125.315, 3);
-- The value -125.32 is returned.
select round(-125.315, 2);
-- The value 100.0 is returned.
select round(123.345, -2);
-- The value null is returned.
select round(null);
-- The value 123.345 is returned.
select round(123.345, 4);
-- The value 0.0 is returned.
select round(123.345, -4);
```

- Example of table data

  Return numbers that are rounded to the specified decimal place in a column based on the Sample data. Sample statement:

```
select round(bigint_data, 1) as bigint_new, round(double_data, 2) as double_new, round(
decimal_data, 1) as decimal_new, round(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+------------+------------+-------------+------------+
| bigint_new | double_new | decimal_new | string_new |
+------------+------------+-------------+------------+
| -10.0      | 0.53       | 0.5         | 10.0       |
| NULL       | -0.1       | -0.1        | -10.0      |
| -1.0       | NULL       | 20.5        | 30.0       |
| 4.0        | 0.89       | NULL        | -30.0      |
| -50.0      | -1.0       | -1          | 50.0       |
| 6.0        | 1.5        | 1.5         | -50.0      |
| -70.0      | -7.5       | -7.5        | NULL       |
| 1.0        | -10.2      | -10.2       | -1.0       |
| -90.0      | 2.58       | 2.6         | 0.0        |
| 10.0       | -5.8       | -5.8        | -90.0      |
+------------+------------+-------------+------------+
```

SIN

- Syntax

```
double|decimal sin(<number>)
```

- Description

  This function calculates the sine of number, which is a radian value.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value -0.3048106211022167 is returned.
    select sin(60);
    -- The value null is returned.
    select sin(null);
    ```

  - Example of table data

    Calculate the sine of values in a column based on the Sample data. Sample statement:

    ```
    select sin(bigint_data) as bigint_new, sin(double_data) as double_new, sin(decimal_data
    ) as decimal_new, sin(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+----------------------+----------------------+----------------------+----------------
-----+
| bigint_new           | double_new           | decimal_new          | string_new
|
+----------------------+----------------------+----------------------+----------------
-----+
| 0.5440211108893698   | 0.5012130046737979   | 0.5012130046737979   | -0.5440211108893
698  |
| NULL                 | -0.09983341664682815 | -0.09983341664682815 | 0.54402111088936
98   |
| -0.8414709848078965  | NULL                 | 0.9995605376022045   | -0.9880316240928
618  |
| -0.7568024953079282  | 0.7770717475268238   | NULL                 | 0.98803162409286
18   |
| 0.26237485370392877  | -0.8414709848078965  | -0.8414709848078965  | -0.2623748537039
2877 |
| -0.27941549819892586 | 0.9974949866040544   | 0.9974949866040544   | 0.26237485370392
877  |
| -0.7738906815578891  | -0.9379999767747389  | -0.9379999767747389  | NULL
|
| 0.8414709848078965   | 0.6998746875935423   | 0.6998746875935423   | -0.8414709848078
965  |
| -0.8939966636005579  | 0.5325349075556212   | 0.5325349075556212   | 0.0
|
| -0.5440211108893698  | 0.46460217941375737  | 0.46460217941375737  | -0.8939966636005
579  |
+----------------------+----------------------+----------------------+----------------
-----+
```

SINH

- Syntax

```
double|decimal sinh(<number>)
```

- Description

  This function calculates the hyperbolic sine of number.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value 5.343237290762231E12 is returned.
select sinh(30);
-- The value null is returned.
select sinh(null);
```

- Example of table data

  Calculate the hyperbolic sine of values in a column based on the Sample data. Sample statement:

```
select sinh(bigint_data) as bigint_new, sinh(double_data) as double_new, sinh(decimal_d
ata) as decimal_new, sinh(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+-----------------------+---------------------+---------------------+---------------
--------+
| bigint_new            | double_new          | decimal_new         | string_new
|
+-----------------------+---------------------+---------------------+---------------
--------+
| -11013.232874703393   | 0.5494517420061382  | 0.5494517420061382  | 11013.23287470
3393    |
| NULL                  | -0.10016675001984403 | -0.10016675001984403 | -11013.2328747
03393   |
| -1.1752011936438014   | NULL                | 380445243.96844625  | 5343237290762.
231     |
| 27.28991719712775     | 1.0122369492687646  | NULL                | -5343237290762
.231    |
| -2.592352764293536e21 | -1.1752011936438014 | -1.1752011936438014 | 2.592352764293
536e21  |
| 201.71315737027922    | 2.1292794550948173  | 2.1292794550948173  | -2.59235276429
3536e21 |
| -1.2577193354595834e30 | -904.0209306858466  | -904.0209306858466  | NULL
|
| 1.1752011936438014    | -13451.593018563612 | -13451.593018563612 | -1.17520119364
38014   |
| -6.102016471589204e38 | 6.560682077817757   | 6.560682077817757   | 0.0
|
| 11013.232874703393    | -165.1482661774516  | -165.1482661774516  | -6.10201647158
9204e38 |
+-----------------------+---------------------+---------------------+---------------
--------+
```

SQRT

- Syntax

```
double|decimal sqrt(<number>)
```

- Description

  This function calculates the square root of number.

- Parameters

number: required. The value is of the DOUBLE or DECIMAL type. The value must be greater than 0. If the value is less than 0, null is returned. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 2.0 is returned.
select sqrt(4);
-- The value null is returned.
select sqrt(null);
```

  - Example of table data

    Calculate the square root of values in a column based on the Sample data. Sample statement:

```
select sqrt(bigint_data) as bigint_new, sqrt(double_data) as double_new, sqrt(decimal_d
ata) as decimal_new, sqrt(string_data) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+--------------------+--------------------+--------------------+--------------------+
| bigint_new         | double_new         | decimal_new        | string_new         |
+--------------------+--------------------+--------------------+--------------------+
| NULL               | 0.724568837309472  | 0.724568837309472  | 3.1622776601683795 |
| NULL               | NULL               | NULL               | NULL               |
| NULL               | NULL               | 4.522167621838006  | 5.477225575051661  |
| 2.0                | 0.9433981132056604 | NULL               | NULL               |
| NULL               | NULL               | NULL               | 7.0710678118654755 |
| 2.449489742783178  | 1.224744871391589  | 1.224744871391589  | NULL               |
| NULL               | NULL               | NULL               | NULL               |
| 1.0                | NULL               | NULL               | NULL               |
| NULL               | 1.606237840420901  | 1.606237840420901  | 0.0                |
| 3.1622776601683795 | NULL               | NULL               | NULL               |
+--------------------+--------------------+--------------------+--------------------+
```

TAN

- Syntax

```
double|decimal tan(<number>)
```

- Description

  This function calculates the tangent of number, which is a radian value.

- Parameters

number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value -6.405331196646276 is returned.
    select tan(30);
    -- The value null is returned.
    select tan(null);
    ```

  - Example of table data

    Calculate the tangent of values in a column based on the Sample data. Sample statement:

    ```
    select tan(bigint_data) as bigint_new, tan(double_data) as double_new, tan(decimal_data
    ) as decimal_new, tan(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+----------------------+----------------------+----------------------+----------------
-----+
| bigint_new           | double_new           | decimal_new          | string_new
|
+----------------------+----------------------+----------------------+----------------
-----+
| -0.6483608274590866  | 0.5792200822893652   | 0.5792200822893652   | 0.64836082745908
66   |
| NULL                 | -0.10033467208545055 | -0.10033467208545055 | -0.6483608274590
866  |
| -1.5574077246549023  | NULL                 | -33.71948732190433   | -6.4053311966462
76   |
| 1.1578212823495775   | 1.2345994590490046   | NULL                 | 6.40533119664627
6    |
| 0.27190061199763077  | -1.5574077246549023  | -1.5574077246549023  | -0.2719006119976
3077 |
| -0.29100619138474915 | 14.101419947171719   | 14.101419947171719   | 0.27190061199763
077  |
| -1.2219599181369434  | -2.706013866772691   | -2.706013866772691   | NULL
|
| 1.5574077246549023   | -0.979852083895097   | -0.979852083895097   | -1.5574077246549
023  |
| 1.995200412208242    | -0.6291704256385503  | -0.6291704256385503  | 0.0
|
| 0.6483608274590866   | 0.5246662219468002   | 0.5246662219468002   | 1.99520041220824
2    |
+----------------------+----------------------+----------------------+----------------
-----+
```

TANH

- Syntax

```
double|decimal tanh(<number>)
```

- Description

  This function calculates the hyperbolic tangent of number.

- Parameters

  number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:

  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value 1.0 is returned.
select tanh(30);
-- The value null is returned.
select tanh(null);
```

- Example of table data

  Calculate the hyperbolic tangent of values in a column based on the Sample data. Sample statement:

```
select tanh(bigint_data) as bigint_new, tanh(double_data) as double_new, tanh(decimal_d
ata) as decimal_new, tanh(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+--------------------+--------------------+--------------------+------------------
---+
| bigint_new         | double_new         | decimal_new        | string_new
|
+--------------------+--------------------+--------------------+------------------
---+
| -0.9999999958776927 | 0.48154979836430806  | 0.48154979836430806  | 0.999999995877692
7  |
| NULL               | -0.09966799462495582 | -0.09966799462495582 | -0.99999999587769
27 |
| -0.7615941559557649 | NULL               | 1.0                | 1.0
|
| 0.999329299739067  | 0.7113937318189625   | NULL               | -1.0
|
| -1.0               | -0.7615941559557649  | -0.7615941559557649  | 1.0
|
| 0.9999877116507956  | 0.9051482536448664   | 0.9051482536448664   | -1.0
|
| -1.0               | -0.9999993881955461  | -0.9999993881955461  | NULL
|
| 0.7615941559557649  | -0.9999999972367348  | -0.9999999972367348  | -0.76159415595576
49 |
| -1.0               | 0.9885821584459533   | 0.9885821584459533   | 0.0
|
| 0.9999999958776927  | -0.9999816679925603  | -0.9999816679925603  | -1.0
|
+--------------------+--------------------+--------------------+------------------
---+
```

TRUNC

- Syntax

```
double|decimal trunc(<number>[, bigint <decimal_places>])
```

- Description

  This function truncates the input value of number to the specified decimal place.

> **Note** If the Hive-compatible data type edition is used, this function is not a mathematical function. It is used to convert a date value. For more information about the related date function, see DATETRUNC. You must set the data type edition of your MaxCompute project based on your business requirements. For more information about data type editions, see Data type editions.

- Parameters
  - number: required. The value is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation.
  - decimal_places: optional. The value is a constant of the BIGINT type. This parameter specifies the position where the number is truncated. If this parameter is not specified, the number is truncated to the ones place. decimal_places can be a negative value, which indicates that the number is truncated from the decimal point to the left and the decimal part is left out. If decimal_places exceeds the length of the integer part, 0 is returned.
- Return value

  A value of the DOUBLE or DECIMAL type is returned. The return value varies based on the following rules:
  - If number is of the DOUBLE or DECIMAL type, a value of the same type is returned.
  - If number is of the STRING or BIGINT type, a value of the DOUBLE type is returned.
  - If decimal_places is not of the BIGINT type, an error is returned.
  - If the value of number or decimal_places is null, null is returned.

  > **Note**
  > - If a value of the DOUBLE type is returned, the return value may not be properly displayed. This issue exists in all systems. For more information, see `trunc(125.815,1)` in the following examples.
  > - The number is filled with zeros from the specified position.
- Examples

- Examples of static data

```
-- The value 125.0 is returned.
select trunc(125.815,0);
-- The value 125.80000000000001 is returned.
select trunc(125.815,1);
-- The value 125.81 is returned.
select trunc(125.815,2);
-- The value 125.815 is returned.
select trunc(125.815,3);
-- The value -125.81 is returned.
select trunc(-125.815,2);
-- The value 120.0 is returned.
select trunc(125.815,-1);
-- The value 100.0 is returned.
select trunc(125.815,-2);
-- The value 0.0 is returned.
select trunc(125.815,-3);
-- The value 123.345 is returned.
select trunc(123.345,4);
-- The value 0.0 is returned.
select trunc(123.345,-4);
-- The value null is returned.
select trunc(123.345,null);
```

- Example of table data

  Truncate numbers in a column to the specified decimal place based on the Sample data. Sample statement:

```
select trunc(bigint_data, 1) as bigint_new, trunc(double_data,1) as double_new, trunc(d
ecimal_data, 1) as decimal_new, trunc(string_data, 1) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+------------+--------------------+-------------+------------+
| bigint_new | double_new         | decimal_new | string_new |
+------------+--------------------+-------------+------------+
| -10.0      | 0.5                | 0.5         | 10.0       |
| NULL       | -0.1               | -0.1        | -10.0      |
| -1.0       | NULL               | 20.4        | 30.0       |
| 4.0        | 0.8                | NULL        | -30.0      |
| -50.0      | -1.0               | -1          | 50.0       |
| 6.0        | 1.5                | 1.5         | -50.0      |
| -70.0      | -7.5               | -7.5        | NULL       |
| 1.0        | -10.200000000000001 | -10.2       | -1.0       |
| -90.0      | 2.5                | 2.5         | 0.0        |
| 10.0       | -5.800000000000001 | -5.8        | -90.0      |
+------------+--------------------+-------------+------------+
```

BIN

- Syntax

```
string bin(<number>)
```

- Description

  This function calculates the binary code of number. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the BIGINT, INT, SMALLINT, or TINYINT type.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If number is not of the BIGINT, INT, SMALLINT, or TINYINT type, an error is returned.
  - If the value of number is 0, 0 is returned.
  - If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value 0 is returned.
    select bin(0);
    -- The value null is returned.
    select bin(null);
    -- The value 1100 is returned.
    select bin(12);
    ```

  - Example of table data

    Calculate the binary code of values in the int_data and bigint_data columns based on the Sample data. Sample statement:

    ```
    -- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
    tements.
    set odps.sql.type.system.odps2=true;
    select bin(int_data) as int_new, bin(bigint_data) as bigint_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+------------------------------------------------------------------------------+---------
----------------------------------------------------------+
| int_new                                                                      | bigint_n
ew                                                         |
+------------------------------------------------------------------------------+---------
----------------------------------------------------------+
| NULL                                                                         | 11111111
111111111111111111111111111111111111111111111111110110 |
| 11111111111111111111111111111111111111111111111111101100                    | NULL
|
| 0                                                                            | 11111111
111111111111111111111111111111111111111111111111111111 |
| 11111111111111111111111111111111111111111111111111011000                    | 100
|
| 101                                                                          | 11111111
111111111111111111111111111111111111111111111111001110 |
| 11111111111111111111111111111111111111111111111111000100                    | 110
|
| 11111111111111111111111111111111111111111111111111111111                    | 11111111
111111111111111111111111111111111111111111111110111010 |
| 11111111111111111111111111111111111111111111111111110110000                 | 1
|
| 1001                                                                         | 11111111
111111111111111111111111111111111111111111111110100110 |
| 11111111111111111111111111111111111111111111111111110011100                 | 1010
|
+------------------------------------------------------------------------------+---------
----------------------------------------------------------+
```

CBRT

- Syntax

```
double cbrt(<number>)
```

- Description

  This function calculates the cube root of number. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the BIGINT, INT, SMALLINT, TINYINT, DOUBLE, FLOAT, or STRING type.

- Return value

  A value of the DOUBLE type is returned. The return value varies based on the following rules:

  - If number is not of the BIGINT, INT, SMALLINT, TINYINT, DOUBLE, FLOAT, or STRING type, an error is returned.
  - If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value 2.0 is returned.
select cbrt(8);
-- The value null is returned.
select cbrt(null);
```

- Example of table data

  Calculate the cube root of values in columns except the decimal_data column based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select cbrt(int_data) as int_new, cbrt(bigint_data) as bigint_new, cbrt(double_data) as
double_new, cbrt(float_data) as float_new, cbrt(string_data) as string_new from mf_math
_fun_t;
```

  The following result is returned:

```
+--------------------+--------------------+---------------------+-------------------
--+---------------------+
| int_new            | bigint_new         | double_new          | float_new
| string_new         |
+--------------------+--------------------+---------------------+-------------------
--+---------------------+
| NULL               | -2.1544346900318834 | 0.806714323012272   | 0.8067143108004823
| 2.1544346900318834 |
| -2.7144176165949063 | NULL               | -0.46415888336127786 | -0.46415888566678
| -2.1544346900318834 |
| 0.0                | -1.0               | NULL                | -1.0
| 3.107232505953859  |
| -3.4199518933533937 | 1.5874010519681996 | 0.9619001716077046  | 0.961900166454112
| -3.107232505953859 |
| 1.7099759466766968 | -3.6840314986403864 | -1.0               | NULL
| 3.6840314986403864 |
| -3.9148676411688634 | 1.8171205928321394 | 1.1447142425533317  | 1.1447142425533317
| -3.6840314986403864 |
| -1.0               | -4.121285299808557 | -1.9574338205844317 | -1.957433820584431
7 | NULL               |
| -4.308869380063767 | 1.0                | -2.168702885250197  | -2.168702871732312
7 | -1.0               |
| 2.080083823051904  | -4.481404746557165 | 1.3715339700741747  | 1.3715339565548288
| 0.0                |
| -4.641588833612778 | 2.1544346900318834 | -1.7967017791430528 | -1.796701798838090
7 | -4.481404746557165 |
+--------------------+--------------------+---------------------+-------------------
--+---------------------+
```

CORR

- Syntax

```
double corr(<col1>, <col2>)
```

- Description

  This function calculates the Pearson correlation coefficient for two columns of data. This function is an extension function of MaxCompute V2.0.

- Parameters

  col1 and col2: required. The names of the two columns for which the Pearson correlation coefficient is calculated. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, or DECIMAL type. Data in the col1 and col2 columns can be of different data types.

- Return value

  A value of the DOUBLE type is returned. If an input column has a null value in a row, the row is not involved in the calculation.

- Examples

  Calculate the Pearson correlation coefficient for the double_data and float_data columns of data based on the Sample data. Sample statement:

  ```
  select corr(double_data,float_data) from mf_math_fun_t;
  ```

  The value 1.0 is returned.

DEGREES

- Syntax

  ```
  double degrees(<number>)
  ```

- Description

  This function converts a radian value into a degree.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type. This function is an extension function of MaxCompute V2.0.

- Return value

  A value of the DOUBLE type is returned. If the value of number is null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value 90.0 is returned.
    select degrees(1.5707963267948966);
    -- The value 0.0 is returned.
    select degrees(0);
    -- The value null is returned.
    select degrees(null);
    ```

- Example of table data

  Obtain all degrees that correspond to all columns based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select degrees(int_data) as int_new, degrees(bigint_data) as bigint_new, degrees(double
_data) as double_new, degrees(decimal_data) as decimal_new, degrees(float_data) as floa
t_new, degrees(string_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+--------------------+--------------------+--------------------+--------------------
-+--------------------+--------------------+
| int_new            | bigint_new         | double_new         | decimal_new
| float_new          | string_new         |
+--------------------+--------------------+--------------------+--------------------
-+--------------------+--------------------+
| NULL               | -572.9577951308232 | 30.08028424436822  | 30.08028424436822
| 30.080282878330387 | 572.9577951308232  |
| -1145.9155902616465 | NULL              | -5.729577951308232 | -5.729577951308232
| -5.729578036685597 | -572.9577951308232 |
| 0.0                | -57.29577951308232 | NULL               | 1171.6986910425335
| -57.29577951308232 | 1718.8733853924698 |
| -2291.831180523293 | 229.1831180523293  | 50.99324376664326  | NULL
| 50.99324294702057  | -1718.8733853924698 |
| 286.4788975654116  | -2864.7889756541163 | -57.29577951308232 | -57.29577951308232
| NULL               | 2864.7889756541163 |
| -3437.7467707849396 | 343.77467707849394 | 85.94366926962348  | 85.94366926962348
| 85.94366926962348  | -2864.7889756541163 |
| -57.29577951308232 | -4010.7045659157625 | -429.71834634811745 | -429.71834634811745
| -429.71834634811745 | NULL              |
| -4583.662361046586 | 57.29577951308232  | -584.4169510334397 | -584.4169510334397
| -584.416940105137  | -57.29577951308232 |
| 515.662015617741   | -5156.620156177409 | 147.8231111437524  | 147.8231111437524
| 147.82310677243132 | 0.0                |
| -5729.5779513082325 | 572.9577951308232 | -332.31552117587745 | -332.31552117587745
| -332.31553210418014 | -5156.620156177409 |
+--------------------+--------------------+--------------------+--------------------
-+--------------------+--------------------+
```

E

- Syntax

```
double e()
```

- Description

  This function calculates the value of `e`. This function is an extension function of MaxCompute V2.0.

- Return value

  A value of the DOUBLE type is returned.

- Examples

```
-- The value 2.718281828459045 is returned.
select e();
```

FACTORIAL

- Syntax

```
bigint factorial(<number>)
```

- Description

  This function calculates the factorial of number. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the BIGINT, INT, SMALLINT, or TINYINT type and ranges from 0 to 20.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of number is 0, 1 is returned.
  - If the value of number is null or a value that does not fall into the range from 0 to 20, null is returned.

- Examples
  - Examples of static data

```
-- The value 120 is returned. 5!=5*4*3*2*1=120
select factorial(5);
-- The value 1 is returned.
select factorial(0);
-- The value null is returned.
select factorial(null);
```

- Example of table data

  Calculate the factorial of values in the int_data and bigint_data columns based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select factorial(int_data) as int_new, factorial(bigint_data) as bigint_new from mf_mat
h_fun_t;
```

  The following result is returned:

```
+------------+------------+
| int_new    | bigint_new |
+------------+------------+
| NULL       | NULL       |
| NULL       | NULL       |
| 1          | NULL       |
| NULL       | 24         |
| 120        | NULL       |
| NULL       | 720        |
| NULL       | NULL       |
| NULL       | 1          |
| 362880     | NULL       |
| NULL       | 3628800    |
+------------+------------+
```

FORMAT_NUMBER

- Syntax

```
string format_number(float|double|decimal <expr1>, <expr2>)
```

- Description

  This function converts a number into a string in the specified format. This function is an extension function of MaxCompute V2.0.

- Parameters
  - expr1: required. This parameter specifies the expression that you want to format. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type.
  - expr2: required. This parameter specifies the format of the expression after the conversion. It can specify the number of decimal places that you want to retain. It can also be expressed in a format similar to `#,###,###.##`.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If expr2 is greater than 0, the value is rounded to the specified place after the decimal point.
  - If expr2 is equal to 0, the value has no decimal point or decimal part.
  - If expr2 is less than 0 or greater than 340, an error is returned.
  - If the value of expr1 or expr2 is null, null is returned.

- Examples

- Examples of static data

```
-- The value 5.230 is returned.
select format_number(5.230134523424545456,3);
-- The value 12,332.123 is returned.
select format_number(12332.123456, '#,###,###,###.###');
-- The value null is returned.
select format_number(null,3);
```

- Example of table data

  Retain values in all columns to the specified number of decimal places based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select format_number(int_data, 1) as int_new, format_number(bigint_data, 1) as bigint_n
ew, format_number(double_data, 2) as double_new, format_number(decimal_data, 1) as deci
mal_new, format_number(float_data, 0) as float_new, format_number(string_data, 1) as st
ring_new from mf_math_fun_t;
```

  The following result is returned:

```
+---------+------------+------------+-------------+-----------+------------+
| int_new | bigint_new | double_new | decimal_new | float_new | string_new |
+---------+------------+------------+-------------+-----------+------------+
| NULL    | -10.0      | 0.53       | 0.5         | 1         | 10.0       |
| -20.0   | NULL       | -0.10      | -0.1        | -0        | -10.0      |
| 0.0     | -1.0       | NULL       | 20.5        | -1        | 30.0       |
| -40.0   | 4.0        | 0.89       | NULL        | 1         | -30.0      |
| 5.0     | -50.0      | -1.00      | -1.0        | NULL      | 50.0       |
| -60.0   | 6.0        | 1.50       | 1.5         | 2         | -50.0      |
| -1.0    | -70.0      | -7.50      | -7.5        | -8        | NULL       |
| -80.0   | 1.0        | -10.20     | -10.2       | -10       | -1.0       |
| 9.0     | -90.0      | 2.58       | 2.6         | 3         | 0.0        |
| -100.0  | 10.0       | -5.80      | -5.8        | -6        | -90.0      |
+---------+------------+------------+-------------+-----------+------------+
```

HEX

- Syntax

```
string hex(<number>)
```

- Description

  This function converts a number or a string into a hexadecimal number. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

- If the value of number is not 0 or null, a value of the STRING type is returned.
- If the value of number is 0, 0 is returned.
- If the value of number is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 0 is returned.
select hex(0);
-- The value 616263 is returned.
select hex('abc');
-- The value 11 is returned.
select hex(17);
-- The value 3137 is returned.
select hex('17');
-- An error is returned.
select hex(null);
```

  - Example of table data

    Convert numbers in all columns into hexadecimal numbers based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select hex(int_data) as int_new, hex(bigint_data) as bigint_new, hex(double_data) as do
uble_new, hex(decimal_data) as decimal_new, hex(float_data) as float_new, hex(string_da
ta) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+------------------+------------------+------------+-------------+------------+--------
----+
| int_new          | bigint_new       | double_new | decimal_new | float_new  | string_
new |
+------------------+------------------+------------+-------------+------------+--------
----+
| NULL             | FFFFFFFFFFFFFFF6 | 302E353235 | 302E353235  | 302E353235 | 3130
|
| FFFFFFFFFFFFFFEC | NULL             | 2D302E31   | 2D302E31    | 2D302E31   | 2D3130
|
| 0                | FFFFFFFFFFFFFFFF | NULL       | 32302E3435  | 2D31       | 3330
|
| FFFFFFFFFFFFFFD8 | 4                | 302E3839   | NULL        | 302E3839   | 2D3330
|
| 5                | FFFFFFFFFFFFFFCE | 2D312E30   | 2D31        | NULL       | 3530
|
| FFFFFFFFFFFFFFC4 | 6                | 312E35     | 312E35      | 312E35     | 2D3530
|
| FFFFFFFFFFFFFFFF | FFFFFFFFFFFFFFBA | 2D372E35   | 2D372E35    | 2D372E35   | NULL
|
| FFFFFFFFFFFFFFB0 | 1                | 2D31302E32 | 2D31302E32  | 2D31302E32 | 2D31
|
| 9                | FFFFFFFFFFFFFFA6 | 322E3538   | 322E3538    | 322E3538   | 30
|
| FFFFFFFFFFFFFF9C | A                | 2D352E38   | 2D352E38    | 2D352E38   | 2D3930
|
+------------------+------------------+------------+-------------+------------+--------
----+
```

LOG2

- Syntax

```
double log2(<number>)
```

- Description

  This function calculates the logarithm of number with the base number of 2. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type.

- Return value

  A value of the DOUBLE type is returned. If the value of number is 0, a negative value, or null, null is returned.

- Examples

- Examples of static data

```
-- The value null is returned.
select log2(null);
-- The value null is returned.
select log2(0);
-- The value 3.0 is returned.
select log2(8);
```

- Example of table data

  Calculate the logarithm of all columns with the base number of 2 based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select log2(int_data) as int_new, log2(bigint_data) as bigint_new, log2(double_data) as
double_new, log2(decimal_data) as decimal_new, log2(float_data) as float_new, log2(stri
ng_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+--------------------+--------------------+----------------------+--------------------+
----------------------+--------------------+
| int_new            | bigint_new         | double_new           | decimal_new        |
float_new            | string_new         |
+--------------------+--------------------+----------------------+--------------------+
----------------------+--------------------+
| NULL               | NULL               | -0.929610672108602   | -0.929610672108602 |
-0.9296107376258038  | 3.3219280948873626 |
| NULL               | NULL               | NULL                 | NULL               |
NULL                 | NULL               |
| NULL               | NULL               | NULL                 | 4.354028938054387  |
NULL                 | 4.906890595608519  |
| NULL               | 2.0                | -0.16812275880832692 | NULL               |
-0.16812278199699915 | NULL               |
| 2.321928094887362  | NULL               | NULL                 | NULL               |
NULL                 | 5.643856189774724  |
| NULL               | 2.584962500721156  | 0.5849625007211562   | 0.5849625007211562 |
0.5849625007211562   | NULL               |
| NULL               | NULL               | NULL                 | NULL               |
NULL                 | NULL               |
| NULL               | 0.0                | NULL                 | NULL               |
NULL                 | NULL               |
| 3.1699250014423126 | NULL               | 1.3673710656485296   | 1.3673710656485296 |
1.367371022986166    | NULL               |
| NULL               | 3.3219280948873626 | NULL                 | NULL               |
NULL                 | NULL               |
+--------------------+--------------------+----------------------+--------------------+
----------------------+--------------------+
```

LOG10

- Syntax

```
double log10(<number>)
```

- Description

  This function calculates the logarithm of number with the base number of 10. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type.

- Return value

  A value of the DOUBLE type is returned. If the value of number is 0, a negative value, or null, null is returned.

- Examples
  - Examples of static data

    ```
    -- The value null is returned.
    select log10(null);
    -- The value null is returned.
    select log10(0);
    -- The value 0.9030899869919435 is returned.
    select log10(8);
    ```

  - Example of table data

    Calculate the logarithm of values in all columns with the base number of 10 based on the Sample data. Sample statement:

    ```
    -- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
    tements.
    set odps.sql.type.system.odps2=true;
    select log10(int_data) as int_new, log10(bigint_data) as bigint_new, log10(double_data)
    as double_new, log10(decimal_data) as decimal_new, log10(float_data) as float_new, log1
    0(string_data) as string_new from mf_math_fun_t;
    ```

    The following result is returned:

```
+--------------------+--------------------+--------------------+--------------------+
----------------------+--------------------+
| int_new            | bigint_new         | double_new         | decimal_new        |
float_new             | string_new         |
+--------------------+--------------------+--------------------+--------------------+
----------------------+--------------------+
| NULL               | NULL               | -0.2798406965940431 | -0.2798406965940431 |
-0.27984071631668606  | 1.0                |
| NULL               | NULL               | NULL               | NULL               |
NULL                  | NULL               |
| NULL               | NULL               | NULL               | 1.3106933123433606  |
NULL                  | 1.4771212547196624 |
| NULL               | 0.6020599913279623 | -0.0506099933550872 | NULL               |
-0.050610000335573106 | NULL               |
| 0.6989700043360187 | NULL               | NULL               | NULL               |
NULL                  | 1.6989700043360185 |
| NULL               | 0.7781512503836435 | 0.17609125905568124 | 0.17609125905568124 |
0.17609125905568124   | NULL               |
| NULL               | NULL               | NULL               | NULL               |
NULL                  | NULL               |
| NULL               | 0.0                | NULL               | NULL               |
NULL                  | NULL               |
| 0.9542425094393249 | NULL               | 0.4116197059632301  | 0.4116197059632301  |
0.411619693120579     | NULL               |
| NULL               | 1.0                | NULL               | NULL               |
NULL                  | NULL               |
+--------------------+--------------------+--------------------+--------------------+
----------------------+--------------------+
```

PI

- Syntax

```
double pi()
```

- Description

  This function calculates the value of π. This function is an extension function of MaxCompute V2.0.

- Return value

  A value of the DOUBLE type is returned.

- Examples

```
-- The value 3.141592653589793 is returned.
select pi();
```

RADIANS

- Syntax

```
double radians(<number>)
```

- Description

This function converts a degree into a radian value. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type.

- Return value

  A value of the DOUBLE type is returned. If the value of number is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 1.5707963267948966 is returned.
select radians(90);
-- The value 0.0 is returned.
select radians(0);
-- The value null is returned.
select radians(null);
```

  - Example of table data

    Convert values in all columns into a radian value based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select radians(int_data) as int_new, radians(bigint_data) as bigint_new, radians(double
_data) as double_new, radians(decimal_data) as decimal_new, radians(float_data) as floa
t_new, radians(string_data) as string_new from mf_math_fun_t;
```

    The following result is returned:

```
+----------------------+----------------------+-----------------------+-------------
-----------+-----------------------+----------------------+
| int_new               | bigint_new            | double_new             | decimal_new
| float_new              | string_new            |
+----------------------+----------------------+-----------------------+-------------
-----------+-----------------------+----------------------+
| NULL                  | -0.17453292519943295  | 0.00916297857297023    | 0.0091629785
7297023    | 0.009162978156851308   | 0.17453292519943295   |
| -0.3490658503988659   | NULL                  | -0.0017453292519943296 | -0.001745329
2519943296 | -0.0017453292780017621 | -0.17453292519943295  |
| 0.0                   | -0.017453292519943295 | NULL                   | 0.3569198320
328404     | -0.017453292519943295  | 0.5235987755982988    |
| -0.6981317007977318   | 0.06981317007977318   | 0.015533430342749534   | NULL
| 0.015533430093078181   | -0.5235987755982988   |
| 0.08726646259971647   | -0.8726646259971648   | -0.017453292519943295  | -0.017453292
519943295  | NULL                  | 0.8726646259971648    |
| -1.0471975511965976   | 0.10471975511965977   | 0.02617993877991494    | 0.0261799387
7991494    | 0.02617993877991494    | -0.8726646259971648   |
| -0.017453292519943295 | -1.2217304763960306   | -0.1308996938995747    | -0.130899693
8995747    | -0.1308996938995747    | NULL                  |
| -1.3962634015954636   | 0.017453292519943295  | -0.17802358370342158   | -0.178023583
70342158   | -0.17802358037447025   | -0.017453292519943295 |
| 0.15707963267948966   | -1.5707963267948966   | 0.045029494701453704   | 0.0450294947
01453704   | 0.04502949336987316    | 0.0                   |
| -1.7453292519943295   | 0.17453292519943295   | -0.10122909661567112   | -0.101229096
61567112   | -0.10122909994462247   | -1.5707963267948966   |
+----------------------+----------------------+-----------------------+-------------
-----------+-----------------------+----------------------+
```

SIGN

- Syntax

```
double sign(<number>)
```

- Description

  This function returns the sign of the input value. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is of the DOUBLE, BIGINT, INT, SMALLINT, TINYINT, FLOAT, DECIMAL, or STRING type.

- Return value

  A value of the DOUBLE type is returned. The return value varies based on the following rules:

  - If the value of number is a positive value, 1.0 is returned.
  - If the value of number is a negative value, -1.0 is returned.
  - If the value of number is 0, 0.0 is returned.
  - If the value of number is null, null is returned.

- Examples

- Examples of static data

```
-- The value -1.0 is returned.
select sign(-2.5);
-- The value 1.0 is returned.
select sign(2.5);
-- The value 0.0 is returned.
select sign(0);
-- The value null is returned.
select sign(null);
```

- Example of table data

  Obtain the sign of numbers in all columns based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select sign(int_data) as int_new, sign(bigint_data) as bigint_new, sign(double_data) as
double_new, sign(decimal_data) as decimal_new, sign(float_data) as float_new, sign(stri
ng_data) as string_new from mf_math_fun_t;
```

  The following result is returned:

```
+------------+------------+------------+-------------+------------+------------+
| int_new    | bigint_new | double_new | decimal_new | float_new  | string_new |
+------------+------------+------------+-------------+------------+------------+
| NULL       | -1.0       | 1.0        | 1           | 1.0        | 1.0        |
| -1.0       | NULL       | -1.0       | -1          | -1.0       | -1.0       |
| 0.0        | -1.0       | NULL       | 1           | -1.0       | 1.0        |
| -1.0       | 1.0        | 1.0        | NULL        | 1.0        | -1.0       |
| 1.0        | -1.0       | -1.0       | -1          | NULL       | 1.0        |
| -1.0       | 1.0        | 1.0        | 1           | 1.0        | -1.0       |
| -1.0       | -1.0       | -1.0       | -1          | -1.0       | NULL       |
| -1.0       | 1.0        | -1.0       | -1          | -1.0       | -1.0       |
| 1.0        | -1.0       | 1.0        | 1           | 1.0        | 0.0        |
| -1.0       | 1.0        | -1.0       | -1          | -1.0       | -1.0       |
+------------+------------+------------+-------------+------------+------------+
```

SHIFTLEFT

- Syntax

```
int shiftleft(tinyint|smallint|int <number1>, int <number2>)
bigint shiftleft(bigint <number1>, int <number2>)
```

- Description

  This function shifts a value left by a specific number of places (<<). This function is an extension function of MaxCompute V2.0.

- Parameters
  - number1: required. The value is of the TINYINT, SMALLINT, INT, or BIGINT type.
  - number2: required. The value is of the INT type.

- Return value

A value of the INT or BIGINT type is returned. The return value varies based on the following rules:

- If number1 is not of the TINYINT, SMALLINT, INT, or BIGINT type, an error is returned.
- If number2 is not of the INT type, an error is returned.
- If the value of number1 or number2 is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 4 is returned. The following statement shifts the binary value of 1 two pl
aces to the left (1<<2,0001 shifted to be 0100).
select shiftleft(1,2);
-- The value 32 is returned. The following statement shifts the binary value of 4 three
places to the left (4<<3,0100 shifted to be 100000).
select shiftleft(4,3);
-- The value null is returned.
select shiftleft(null,2);
```

  - Example of table data

    Shift numbers in the int_data and bigint_data columns left by a specific number of places based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select shiftleft(int_data, 1) as int_new, shiftleft(bigint_data, 1) as bigint_new from
mf_math_fun_t;
```

    The following result is returned:

```
+---------+------------+
| int_new | bigint_new |
+---------+------------+
| NULL    | -20        |
| -40     | NULL       |
| 0       | -2         |
| -80     | 8          |
| 10      | -100       |
| -120    | 12         |
| -2      | -140       |
| -160    | 2          |
| 18      | -180       |
| -200    | 20         |
+---------+------------+
```

SHIFTRIGHT

- Syntax

```
int shiftright(tinyint|smallint|int <number1>, int <number2>)
bigint shiftright(bigint <number1>, int <number2>)
```

- Description

This function shifts a value right by a specific number of places (>>). This function is an extension function of MaxCompute V2.0.

- Parameters
  - number1: required. The value is of the TINYINT, SMALLINT, INT, or BIGINT type.
  - number2: required. The value is of the INT type.
- Return value

  A value of the INT or BIGINT type is returned. The return value varies based on the following rules:
  - If number1 is not of the TINYINT, SMALLINT, INT, or BIGINT type, an error is returned.
  - If number2 is not of the INT type, an error is returned.
  - If the value of number1 or number2 is null, null is returned.
- Examples
  - Examples of static data

```
-- The value 1 is returned. The following statement shifts the binary value of 4 two pl
aces to the right (4>>2,0100 shifted to be 0001).
select shiftright(4,2);
-- The value 4 is returned. The following statement shifts the binary value of 32 three
places to the right (32>>3,100000 shifted to be 0100).
select shiftright(32,3);
-- The value null is returned.
select shiftright(null,3);
```

  - Example of table data

    Shift numbers in the int_data and bigint_data columns right by a specific number of places based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select shiftright(int_data, 1) as int_new, shiftright(bigint_data, 1) as bigint_new fro
m mf_math_fun_t;
```

    The following result is returned:

```
+---------+------------+
| int_new | bigint_new |
+---------+------------+
| NULL    | -5         |
| -10     | NULL       |
| 0       | -1         |
| -20     | 2          |
| 2       | -25        |
| -30     | 3          |
| -1      | -35        |
| -40     | 0          |
| 4       | -45        |
| -50     | 5          |
+---------+------------+
```

SHIFTRIGHTUNSIGNED

- Syntax

```
int shiftrightunsigned(tinyint|smallint|int <number1>, int <number2>)
bigint shiftrightunsigned(bigint <number1>, int <number2>)
```

- Description

  This function shifts an unsigned value right by a specific number of places (>>>). This function is an extension function of MaxCompute V2.0.

- Parameters
  - number1: required. The value is an integer of the TINYINT, SMALLINT, INT, or BIGINT type.
  - number2: required. The value is an integer of the INT type.

- Return value

  A value of the INT or BIGINT type is returned. The return value varies based on the following rules:
  - If number1 is not of the TINYINT, SMALLINT, INT, or BIGINT type, an error is returned.
  - If number2 is not of the INT type, an error is returned.
  - If the value of number1 or number2 is null, null is returned.

- Examples
  - Examples of static data

```
-- The value 2 is returned. The following statement shifts the binary unsigned value of
8 two places to the right (8>>>2,1000 shifted to be 0010).
select shiftrightunsigned(8,2);
-- The value 1073741820 is returned. The following statement shifts the binary value of
-14 two places to the right (-14>>>2, 11111111 11111111 11111111 11110010 shifted to be
00111111 11111111 11111111 11111100).
select shiftrightunsigned(-14,2);
-- The value null is returned.
select shiftrightunsigned(-14,null);
```

- Example of table data

  Shift unsigned values in the int_data and bigint_data columns right by a specific number of places based on the Sample data. Sample statement:

```
-- Enable the MaxCompute V2.0 data type edition. Commit this command along with SQL sta
tements.
set odps.sql.type.system.odps2=true;
select shiftrightunsigned(int_data, 1) as int_new, shiftrightunsigned(bigint_data, 1) a
s bigint_new from mf_math_fun_t;
```

  The following result is returned:

```
+------------+---------------------+
| int_new    | bigint_new          |
+------------+---------------------+
| NULL       | 9223372036854775803 |
| 2147483638 | NULL                |
| 0          | 9223372036854775807 |
| 2147483628 | 2                   |
| 2          | 9223372036854775783 |
| 2147483618 | 3                   |
| 2147483647 | 9223372036854775773 |
| 2147483608 | 0                   |
| 4          | 9223372036854775763 |
| 2147483598 | 5                   |
+------------+---------------------+
```

UNHEX

- Syntax

```
binary unhex(string <number>)
```

- Description

  This function converts a hexadecimal string into a string. This function is an extension function of MaxCompute V2.0.

- Parameters

  number: required. The value is a hexadecimal string.

- Return value

  A value of the BINARY type is returned. The return value varies based on the following rules:

  - If the value of number is 0, an error is returned.
  - If the value of number is null, null is returned.

- Examples

```
-- The value abc is returned.
select unhex('616263');
-- The value abc is returned.
select unhex(616263);
-- The value null is returned.
select unhex(null);
```

WIDTH_BUCKET

- Syntax

```
width_bucket(numeric <expr>, numeric <min_value>, numeric <max_value>, int <num_buckets>)
```

- Description

  This function specifies the number of buckets and the minimum and maximum values of the acceptable range for a bucket. It allows you to construct equi-width buckets, in which the bucket range is divided into intervals that have an identical size. It returns the ID of the bucket into which the value of a specific expression falls. This function supports the following data types: DECIMAL(precision,scale) in the MaxCompute V2.0 data type edition, BIGINT, INT, FLOAT, DOUBLE, and DECIMAL. For more information, see MaxCompute V2.0 data type edition. This function is an extension function of MaxCompute V2.0.

- Parameters
  - expr: required. This parameter specifies the expression for which you want to identify the matching bucket ID.
  - min_value: required. This parameter specifies the minimum value of the acceptable range for the bucket.
  - max_value: required. This parameter specifies the maximum value of the acceptable range for the bucket. The value must be greater than min_value.
  - num_buckets: required. This parameter specifies the number of buckets. The value must be greater than 0.

- Return value

  A value of the BIGINT type is returned. The value ranges from 0 to num_buckets plus 1. The return value varies based on the following rules:
  - If the value of expr is less than that of min_value, 0 is returned.
  - If the value of expr is greater than that of max_value, the value of num_buckets plus 1 is returned.
  - If the value of expr is null, null is returned. In other cases, the ID of the bucket into which the value falls is returned. The bucket ID is calculated based on the following formula: `Bucket ID = floor( num_buckets × (expr - min_value)/(max_value - min_value) + 1)` .
  - If the value of min_value, max_value, or num_buckets is null, null is returned.

- Examples

- Example 1: The values of all input parameters are not null. Sample statement:

```
select key,value,width_bucket(value,100,500,5) as value_group
from values
    (1,99),
    (2,100),
    (3,199),
    (4,200),
    (5,499),
    (6,500),
    (7,501),
    (8,NULL)
as t(key,value);
```

The following result is returned:

```
+-------+--------+-------------+
| key   | value  | value_group |
+-------+--------+-------------+
| 1     | 99     | 0           |
| 2     | 100    | 1           |
| 3     | 199    | 2           |
| 4     | 200    | 2           |
| 5     | 499    | 5           |
| 6     | 500    | 6           |
| 7     | 501    | 6           |
| 8     | \N     | \N          |
+-------+--------+-------------+
```

- Example 2: The value of an input parameter is null. Sample statement:

```
select key,value,width_bucket(value,100,500,null) as value_group
from values
    (1,99),
    (2,100),
    (3,199),
    (4,200),
    (5,499),
    (6,500),
    (7,501),
    (8,NULL)
as t(key,value);
```

The following result is returned:

```
+------+-------+-------------+
| key  | value | value_group |
+------+-------+-------------+
| 1    | 99    | NULL        |
| 2    | 100   | NULL        |
| 3    | 199   | NULL        |
| 4    | 200   | NULL        |
| 5    | 499   | NULL        |
| 6    | 500   | NULL        |
| 7    | 501   | NULL        |
| 8    | NULL  | NULL        |
+------+-------+-------------+
```

3.9.5. Window functions

You can use window functions in MaxCompute SQL to flexibly process and analyze data of columns in a window. This topic describes the syntax and parameters of window functions that are supported by MaxCompute SQL. This topic also provides examples on how to use window functions. It guides you through data development by using window functions.

The following table lists the window functions that are supported by MaxCompute SQL.

Function	Description
AVG	Calculates the average value of data in a window.
CLUSTER_SAMPLE	Samples random rows of data. If true is returned, the specified row of data is sampled.
COUNT	Calculates the number of rows in a window.
CUME_DIST	Calculates the cumulative distribution of data in a partition.
DENSE_RANK	Calculates the rank of a row in an ordered group of rows. The ranks are consecutive integers.
FIRST_VALUE	Obtains the calculated result of the first row of data in the window to which the current row belongs.

Function	Description
LAG	Obtains the calculated result of the Nth row of data that precedes the current row at a given offset in a window.
LAST_VALUE	Obtains the calculated result of the last row of data in the window to which the current row belongs.
LEAD	Obtains the calculated result of the Nth row of data that follows the current row at a given offset in a window.
MAX	Calculates the maximum value in a window.
MEDIAN	Calculates the median in a window.
MIN	Calculates the minimum value in a window.
NTH_VALUE	Obtains the calculated result of the Nth row of data in a window to which the current row belongs.
NTILE	Splits rows of data in a partition into N groups of equal size and returns the number of the group to which the current row belongs. The group number ranges from 1 to N.
PERCENT_RANK	Calculates the percentile rank of a row in a group of rows.
RANK	Calculates the rank of a row in a group of rows. The ranks may not be consecutive integers.
ROW_NUMBER	Calculates the sequence number of a row. The row number starts from 1.
STDDEV	Returns the population standard deviation of all input values. This function is also called STDDEV_POP.
STDDEV_SAMP	Returns the sample standard deviation of all input values.
SUM	Calculates the sum of data in a window.

Limits

Before you use window functions, take note of the following limits:

- Window functions are supported only in `SELECT` statements.
- A window function cannot contain nested window functions or aggregate functions.
- You cannot use window functions together with aggregate functions of the same level.

Syntax

Syntax of window functions:

```
<function_name>([distinct][<expression> [, ...]]) over (<window_definition>)
<function_name>([distinct][<expression> [, ...]]) over <window_name>
```

- function_name: a built-in window function, built-in aggregate function, or user-defined aggregate

function (UDAF). For more information about built-in aggregate functions, see Aggregate functions. For more information about UDAFs, see Overview.

> **Note** If you use a built-in aggregate function as a window function, you must run the `set odps.sql.window.function.newimpl=true;` command at the session level.

- expression: the format of a window function. The format is subject to the function syntax.
- windowing_definition: the definition of a window. For more information about the syntax of windowing_definition, see windowing_definition.
- window_name: the name of a window. You can use the `window` keyword to configure a window and use windowing_definition to specify the name of the window. Syntax of named_window_def:

```
window <window_name> as (<window_definition>)
```

Position of named_window_def in an SQL statement:

```
select ... from ... [where ...] [group by ...] [having ...] named_window_def [order by ..
.] [limit ...]
```

windowing_definition

Syntax

```
--partition_clause:
[partition by <expression> [, ...]]
--orderby_clause:
[order by <expression> [asc|desc][nulls {first|last}] [, ...]]
[<frame_clause>]
```

If you use a window function in a SELECT statement, data is partitioned and sorted based on `partition by` and `order by` in windowing_definition when the window function is executed. If the SELECT statement does not include `partition by`, only one partition exists. If the SELECT statement does not include `order by`, data in a partition is arranged in a random order, and a data stream is generated. After the data stream is generated, a group of rows are extracted from the data stream based on `frame_clause` in windowing_definition to create a window for the current row. The window function calculates the data included in the window to which the current row belongs.

- partition by <expression> [, ...]: optional. This parameter specifies the partition information. If the values of partition key columns are the same for a group of rows, these rows are included in the same window. For more information about the format of partition by, see Table operations.
- order by <expression> [asc|desc][nulls {first|last}] [, ...]: optional. This parameter specifies how to sort rows of data in a window.

> **Note** If the values of the column that is specified in `order by` are the same, the sorting result may not be accurate. To reduce the random ordering of data, make sure that the values of the column that is specified in `order by` are unique.

- frame_clause: optional. This parameter is used to determine the data boundaries of a window. For more information about frame_clause, see frame_clause.

frame_clause

Syntax

```
-- Syntax 1
{ROWS|RANGE|GROUPS} <frame_start> [<frame_exclusion>]
-- Syntax 2
{ROWS|RANGE|GROUPS} between <frame_start> and <frame_end> [<frame_exclusion>]
```

frame_clause is a closed interval that is used to determine the data boundaries of a window. The data boundaries are determined based on the rows that are specified by frame_start and frame_end.

- ROWS|RANGE|GROUPS: required. ROWS, RANGE, and GROUPS indicate the types of frame_clause. The implementation rules of frame_start and frame_end vary based on the type of frame_clause.
  - ROWS: The data boundaries of a window are determined based on the number of rows.
  - RANGE: The data boundaries of a window are determined based on the comparison results of the values of the column that is specified in `order by`. In most cases, `order by` is specified in windowing_definition. If `order by` is not specified in windowing_definition, the values of the column that is specified in `order by` are the same for all rows in a partition. NULL values are considered equivalent.
  - GROUPS: In a partition, rows that have the same value of the column specified in `order by` form a group. If `order by` is not specified, all rows in the partition form a group. NULL values are considered equivalent.
- frame_start and frame_end: the start and end rows of a window. frame_start is required. frame_end is optional. If frame_end is not specified, the default value CURRENT ROW is used.

  The row specified by frame_start must precede or be the same as the row specified by frame_end. Compared with the row specified by frame_end, the row specified by frame_start is closer to the first row in a window after all data in the window is sorted based on the column that is specified in `order by` of windowing_definition. The following table describes the valid values and logic of frame_start and frame_end when the type of frame_clause is ROWS, RANGE, or GROUPS.

frame_clause type	frame_start or frame_end value	Description
ROWS, RANGE, and GROUPS	UNBOUNDED PRECEDING	Indicates the first row of a partition. Rows are counted from 1.
	UNBOUNDED FOLLOWING	Indicates the last row of a partition.
ROWS	CURRENT ROW	Indicates the current row. Each row of data corresponds to a result calculated by a window function. The current row indicates the row whose data is calculated by using a window function.
	offset PRECEDING	Indicates the Nth row that precedes the current row at a given `offset`. For example, if `0 PRECEDING` indicates the current row, `1 PRECEDING` indicates the previous row. `offset` must be a non-negative integer.

frame_clause type	frame_start or frame_end value	Description
	offset FOLLOWING	Indicates the Nth row that follows the current row at a given `offset`. For example, if `0 FOLLOWING` indicates the current row, `1 FOLLOWING` indicates the next row. `offset` must be a non-negative integer.
	CURRENT ROW	◦ If frame_start is set to CURRENT ROW, it indicates the first row that has the same value of the column specified in `order by` as the current row. ◦ If frame_end is set to CURRENT ROW, it indicates the last row that has the same value of the column specified in `order by` as the current row.

frame_clause type	frame_start or frame_end value	Description
RANGE	offset PRECEDING	The rows that are specified by frame_start and frame_end are determined based on the sorting order that is specified by `order by`. For example, if data in a window is sorted by X, Xi indicates the X value that corresponds to the ith row, and Xc indicates the X value that corresponds to the current row. ○ `order by` is set to asc: ■ frame_start indicates the first row that meets the following requirement: `Xc - Xi ≤ offset`. ■ frame_end indicates the last row that meets the following requirement: `Xc - Xi ≥ offset`. ○ `order by` is set to desc: ■ frame_start indicates the first row that meets the following requirement: `Xi - Xc ≤ offset`. ■ frame_end indicates the last row that meets the following requirement: `Xi - Xc ≥ offset`. The column that is specified in `order by` can be of the following data types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, DATETIME, DATE, and TIMESTAMP. Syntax for `offset` of a date type: ○ `N`: indicates N days or N seconds. It must be a non-negative integer. For an offset of the DATETIME or TIMESTAMP type, it indicates N seconds. For an offset of the DATE type, it indicates N days. ○ `interval 'N' {YEAR\MONTH\DAY\HOUR\MINUTE\SECOND}`: indicates N years, months, days, hours, minutes, or seconds. For example, `INTERVAL '3' YEAR` indicates 3 years. ○ `INTERVAL 'N-M' YEAR TO MONTH`: indicates N years and M months. For example, `INTERVAL '1-3' YEAR TO MONTH` indicates 1 year and 3 months. ○ `INTERVAL 'D[ H[:M[:S[:N]]]]' DAY TO SECOND`: indicates D days, H hours, M minutes, S seconds, and N nanoseconds. For example, `INTERVAL '1 2:3:4:5' DAY TO SECOND` indicates 1 day, 2 hours, 3 minutes, 4 seconds, and 5 nanoseconds.

frame_clause type	frame_start or frame_end value	Description
	offset FOLLOWING	The rows that are specified by frame_start and frame_end are determined based on the sorting order that is specified by `order by`. For example, if data in a window is sorted by X, Xi indicates the X value that corresponds to the ith row, and Xc indicates the X value that corresponds to the current row. ○ `order by` is set to asc: ■ frame_start indicates the first row that meets the following requirement: `Xi - Xc ≥ offset`. ■ frame_end indicates the last row that meets the following requirement: `Xi - Xc ≤ offset`. ○ `order by` is set to desc: ■ frame_start indicates the first row that meets the following requirement: `Xc - Xi ≥ offset`. ■ frame_end indicates the last row that meets the following requirement: `Xc - Xi ≤ offset`.
GROUPS	CURRENT ROW	○ If frame_start is set to CURRENT ROW, it indicates the first row of the group to which the current row belongs. ○ If frame_end is set to CURRENT ROW, it indicates the last row of the group to which the current row belongs.
	offset PRECEDING	○ If frame_start is set to offset PRECEDING, it indicates the first row of the Nth group that precedes the group of the current row at a given `offset`. ○ If frame_end is set to offset PRECEDING, it indicates the last row of the Nth group that precedes the group of the current row at a given `offset`. **Note** You cannot set frame_start to UNBOUNDED FOLLOWING, and you cannot set frame_end to UNBOUNED PRECEDING.

frame_clause type	frame_start or frame_end value	Description
	offset FOLLOWING	◦ If frame_start is set to offset FOLLOWING, it indicates the first row of the Nth group that follows the group of the current row at a given `offset`. ◦ If frame_end is set to offset FOLLOWING, it indicates the last row of the Nth group that follows the group of the current row at a given `offset`. **Note** You cannot set frame_start to UNBOUNDED FOLLOWING, and you cannot set frame_end to UNBOUNED PRECEDING.

- frame_exclusion: optional. This parameter is used to remove specific rows from a window. Valid values:
  - EXCLUDE NO OTHERS: indicates that no rows are excluded from the window.
  - EXCLUDE CURRENT ROW: indicates that the current row is excluded from the window.
  - EXCLUDE GROUP: indicates that an entire group of rows in a partition is excluded from the window. In the group, all rows have the same value of the column that is specified in `order by` as the current row.
  - EXCLUDE TIES: indicates that an entire group of rows, except for the current row, are excluded from the window.

Default frame_clause

If you do not specify frame_clause, MaxCompute uses the default frame_clause to determine the data boundaries of a window.

- If `odps.sql.hive.compatible` is set to true, the following default frame_clause is used. This rule applies to most SQL systems.

```
RANGE between UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE NO OTHERS
```

- If `odps.sql.hive.compatible` is set to false, `order by` is specified, and one of the following window functions is used, the default frame_clause in ROWS mode is used: AVG, COUNT, MAX, MIN, STDDEV, STEDEV_POP, STDDEV_SAMP, and SUM.

```
ROWS between UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE NO OTHERS
```

Example of data boundaries of a window

In this example, a table named tbl contains three columns that are of the BIGINT type: `pid, oid, and rid`. The tbl table contains the following data:

```
+------------+------------+------------+
| pid        | oid        | rid        |
+------------+------------+------------+
| 1          | NULL       | 1          |
| 1          | NULL       | 2          |
| 1          | 1          | 3          |
| 1          | 1          | 4          |
| 1          | 2          | 5          |
| 1          | 4          | 6          |
| 1          | 7          | 7          |
| 1          | 11         | 8          |
| 2          | NULL       | 9          |
| 2          | NULL       | 10         |
+------------+------------+------------+
```

You can replace ellipses (`...`) in SQL statements with windowing_definition to display the data in the windows in which each row of data is included.

> **Note** If a value in the window column in the returned result is NULL, no data is contained in the window.

- Windows in ROWS mode
  - windowing_definition 1

    ```
    partition by pid order by oid ROWS between UNBOUNDED PRECEDING and CURRENT ROW
    -- Sample SQL statement:
    set odps.sql.window.function.newimpl=true;
    select pid, oid, rid, collect_list(rid) over(partition by pid order by oid ROWS between
    UNBOUNDED PRECEDING and CURRENT ROW) as window from tbl;
    ```

    The following result is returned:

    ```
    +------------+------------+------------+--------+
    | pid        | oid        | rid        | window |
    +------------+------------+------------+--------+
    | 1          | NULL       | 1          | [1]    |
    | 1          | NULL       | 2          | [1, 2] |
    | 1          | 1          | 3          | [1, 2, 3] |
    | 1          | 1          | 4          | [1, 2, 3, 4] |
    | 1          | 2          | 5          | [1, 2, 3, 4, 5] |
    | 1          | 4          | 6          | [1, 2, 3, 4, 5, 6] |
    | 1          | 7          | 7          | [1, 2, 3, 4, 5, 6, 7] |
    | 1          | 11         | 8          | [1, 2, 3, 4, 5, 6, 7, 8] |
    | 2          | NULL       | 9          | [9]    |
    | 2          | NULL       | 10         | [9, 10] |
    +------------+------------+------------+--------+
    ```

- windowing_definition 2

```
partition by pid order by oid ROWS between UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid ROWS between
UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | NULL       | 2          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 1          | 3          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 1          | 4          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 2          | 5          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 4          | 6          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 7          | 7          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 11         | 8          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 2          | NULL       | 9          | [9, 10] |
| 2          | NULL       | 10         | [9, 10] |
+------------+------------+------------+--------+
```

- windowing_definition 3

```
partition by pid order by oid ROWS between 1 FOLLOWING and 3 FOLLOWING
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid ROWS between
1 FOLLOWING and 3 FOLLOWING) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [2, 3, 4] |
| 1          | NULL       | 2          | [3, 4, 5] |
| 1          | 1          | 3          | [4, 5, 6] |
| 1          | 1          | 4          | [5, 6, 7] |
| 1          | 2          | 5          | [6, 7, 8] |
| 1          | 4          | 6          | [7, 8] |
| 1          | 7          | 7          | [8]    |
| 1          | 11         | 8          | NULL   |
| 2          | NULL       | 9          | [10]   |
| 2          | NULL       | 10         | NULL   |
+------------+------------+------------+--------+
```

- windowing_definition 4

```
partition by pid order by oid ROWS between UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE
CURRENT ROW
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid ROWS between
UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE CURRENT ROW) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | NULL   |
| 1          | NULL       | 2          | [1]    |
| 1          | 1          | 3          | [1, 2] |
| 1          | 1          | 4          | [1, 2, 3] |
| 1          | 2          | 5          | [1, 2, 3, 4] |
| 1          | 4          | 6          | [1, 2, 3, 4, 5] |
| 1          | 7          | 7          | [1, 2, 3, 4, 5, 6] |
| 1          | 11         | 8          | [1, 2, 3, 4, 5, 6, 7] |
| 2          | NULL       | 9          | NULL   |
| 2          | NULL       | 10         | [9]    |
+------------+------------+------------+--------+
```

- windowing_definition 5

```
partition by pid order by oid ROWS between UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE
GROUP
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid ROWS between
UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE GROUP) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | NULL   |
| 1          | NULL       | 2          | NULL   |
| 1          | 1          | 3          | [1, 2] |
| 1          | 1          | 4          | [1, 2] |
| 1          | 2          | 5          | [1, 2, 3, 4] |
| 1          | 4          | 6          | [1, 2, 3, 4, 5] |
| 1          | 7          | 7          | [1, 2, 3, 4, 5, 6] |
| 1          | 11         | 8          | [1, 2, 3, 4, 5, 6, 7] |
| 2          | NULL       | 9          | NULL   |
| 2          | NULL       | 10         | NULL   |
+------------+------------+------------+--------+
```

- windowing_definition 6

```
partition by pid order by oid ROWS between UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE
TIES
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid ROWS between
UNBOUNDED PRECEDING and CURRENT ROW EXCLUDE TIES) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [1]    |
| 1          | NULL       | 2          | [2]    |
| 1          | 1          | 3          | [1, 2, 3] |
| 1          | 1          | 4          | [1, 2, 4] |
| 1          | 2          | 5          | [1, 2, 3, 4, 5] |
| 1          | 4          | 6          | [1, 2, 3, 4, 5, 6] |
| 1          | 7          | 7          | [1, 2, 3, 4, 5, 6, 7] |
| 1          | 11         | 8          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 2          | NULL       | 9          | [9]    |
| 2          | NULL       | 10         | [10]   |
+------------+------------+------------+--------+
```

The differences between EXCLUDE CURRENT ROW and EXCLUDE GROUP can be obtained based on the comparison between the `window` column values of rows with the `rid` column values of 2, 4, and 10 in Syntax 5 and Syntax 6. If frame_exclusion is set to EXCLUDE GROUP, the rows that have the same `pid` column value in a partition are extracted when the rows have the same `oid` column value as the current row.

- Windows in RANGE mode

- windowing_definition 1

```
partition by pid order by oid RANGE between UNBOUNDED PRECEDING and CURRENT ROW
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid RANGE betwee
n UNBOUNDED PRECEDING and CURRENT ROW) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [1, 2] |
| 1          | NULL       | 2          | [1, 2] |
| 1          | 1          | 3          | [1, 2, 3, 4] |
| 1          | 1          | 4          | [1, 2, 3, 4] |
| 1          | 2          | 5          | [1, 2, 3, 4, 5] |
| 1          | 4          | 6          | [1, 2, 3, 4, 5, 6] |
| 1          | 7          | 7          | [1, 2, 3, 4, 5, 6, 7] |
| 1          | 11         | 8          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 2          | NULL       | 9          | [9, 10] |
| 2          | NULL       | 10         | [9, 10] |
+------------+------------+------------+--------+
```

If frame_end is set to CURRENT ROW, the last row that has the same value of the `oid` column in `order by` as the current row is obtained. Therefore, the `window` column value of the row whose `rid` column value is 1 is [1, 2].

- windowing_definition 2

```
partition by pid order by oid RANGE between CURRENT ROW and UNBOUNDED FOLLOWING
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid RANGE betwee
n CURRENT ROW and UNBOUNDED FOLLOWING) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | NULL       | 2          | [1, 2, 3, 4, 5, 6, 7, 8] |
| 1          | 1          | 3          | [3, 4, 5, 6, 7, 8] |
| 1          | 1          | 4          | [3, 4, 5, 6, 7, 8] |
| 1          | 2          | 5          | [5, 6, 7, 8] |
| 1          | 4          | 6          | [6, 7, 8] |
| 1          | 7          | 7          | [7, 8] |
| 1          | 11         | 8          | [8]    |
| 2          | NULL       | 9          | [9, 10] |
| 2          | NULL       | 10         | [9, 10] |
+------------+------------+------------+--------+
```

- windowing_definition 3

```
partition by pid order by oid RANGE between 3 PRECEDING and 1 PRECEDING
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid RANGE betwee
n 3 PRECEDING and 1 PRECEDING) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [1, 2] |
| 1          | NULL       | 2          | [1, 2] |
| 1          | 1          | 3          | NULL   |
| 1          | 1          | 4          | NULL   |
| 1          | 2          | 5          | [3, 4] |
| 1          | 4          | 6          | [3, 4, 5] |
| 1          | 7          | 7          | [6]    |
| 1          | 11         | 8          | NULL   |
| 2          | NULL       | 9          | [9, 10] |
| 2          | NULL       | 10         | [9, 10] |
+------------+------------+------------+--------+
```

For the row whose value of `oid` in `order by` is NULL, if frame_start is set to `offset PRECEDING` or offset FOLLOWING, the row is the first row whose value of oid in order by is NULL. If frame_end is set to offset PRECEDING or offset FOLLOWING, the row is the last row whose value of oid in `order by` is NULL.

- Windows in GROUPS mode

  windowing_definition

```
partition by pid order by oid GROUPS between 2 PRECEDING and CURRENT ROW
-- Sample SQL statement:
set odps.sql.window.function.newimpl=true;
select pid, oid, rid, collect_list(rid) over(partition by pid order by oid GROUPS between
2 PRECEDING and CURRENT ROW) as window from tbl;
```

The following result is returned:

```
+------------+------------+------------+--------+
| pid        | oid        | rid        | window |
+------------+------------+------------+--------+
| 1          | NULL       | 1          | [1, 2] |
| 1          | NULL       | 2          | [1, 2] |
| 1          | 1          | 3          | [1, 2, 3, 4] |
| 1          | 1          | 4          | [1, 2, 3, 4] |
| 1          | 2          | 5          | [1, 2, 3, 4, 5] |
| 1          | 4          | 6          | [3, 4, 5, 6] |
| 1          | 7          | 7          | [5, 6, 7] |
| 1          | 11         | 8          | [6, 7, 8] |
| 2          | NULL       | 9          | [9, 10] |
| 2          | NULL       | 10         | [9, 10] |
+------------+------------+------------+--------+
```

Sample data

This section provides sample source data and examples for you to understand how to use the functions. Create a table named emp and insert the sample data into the table. Sample commands:

```
create table if not exists emp
   (empno bigint,
    ename string,
    job string,
    mgr bigint,
    hiredate datetime,
    sal bigint,
    comm bigint,
    deptno bigint);
tunnel upload emp.txt emp;
```

The emp.txt file contains the following sample data:

```
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

AVG

- Syntax

```
double avg([distinct] double <expr>) over ([partition_clause] [orderby_clause] [frame_cla
use])
decimal avg([distinct] decimal <expr>) over ([partition_clause] [orderby_clause] [frame_c
lause])
```

- Description

  This function returns the average value of expr in a window.

- Parameters
  - expr: required. The expression that is used to calculate the returned result. A value of the DOUBLE or DECIMAL type.
    - If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If it is of another data type, an error is returned.
    - If the input value is NULL, this row is not used for calculation.
    - If the distinct keyword is specified, the average value of distinct values is calculated.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

  If the input value of expr is of the DECIMAL type, a value of the DECIMAL type is returned. If the input value is of another data type, a value of the DOUBLE type is returned. If the input value of expr is NULL, NULL is returned.

- Examples

- Example 1: Use the deptno column to define a window and calculate the average value of the sal column. The order by clause is not specified. This function returns the cumulative average value of the values from the first row to the last row in the current window. The current window includes the rows that have the same deptno column value. Sample statements:

```
select deptno, sal, avg(sal) over (partition by deptno) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 2916.6666666666665 |   -- This row is the first row of this
window. The return value is the cumulative average value of the values from the first r
ow to the sixth row.
| 10         | 2450       | 2916.6666666666665 |   -- The return value is the cumulativ
e average value of the values from the first row to the sixth row.
| 10         | 5000       | 2916.6666666666665 |   -- The return value is the cumulativ
e average value of the values from the first row to the sixth row.
| 10         | 1300       | 2916.6666666666665 |
| 10         | 5000       | 2916.6666666666665 |
| 10         | 2450       | 2916.6666666666665 |
| 20         | 3000       | 2175.0     |
| 20         | 3000       | 2175.0     |
| 20         | 800        | 2175.0     |
| 20         | 1100       | 2175.0     |
| 20         | 2975       | 2175.0     |
| 30         | 1500       | 1566.6666666666667 |
| 30         | 950        | 1566.6666666666667 |
| 30         | 1600       | 1566.6666666666667 |
| 30         | 1250       | 1566.6666666666667 |
| 30         | 1250       | 1566.6666666666667 |
| 30         | 2850       | 1566.6666666666667 |
+------------+------------+------------+
```

- Example 2: In non-Hive-compatible mode, use the deptno column to define a window and calculate the average value of the sal column. The order by clause is specified. This function returns the cumulative average value of the values from the first row to the current row in the current window. The current window includes the rows that have the same deptno value. Sample statements:

```
-- Disable the Hive-compatible mode.
set odps.sql.hive.compatible=false;
-- Execute the following statement:
select deptno, sal, avg(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1300.0     |           -- This row is the first row of this
window.
| 10         | 1300       | 1300.0     |           -- The return value is the cumulativ
e average value of the values in the first and second rows.
| 10         | 2450       | 1683.3333333333333 |   -- The return value is the cumulativ
e average value of the values from the first row to the third row.
| 10         | 2450       | 1875.0     |           -- The return value is the cumulativ
e average value of the values from the first row to the fourth row.
| 10         | 5000       | 2500.0     |           -- The return value is the cumulativ
e average value of the values from the first row to the fifth row.
| 10         | 5000       | 2916.6666666666665 |   -- The return value is the cumulativ
e average value of the values from the first row to the sixth row.
| 20         | 800        | 800.0      |
| 20         | 1100       | 950.0      |
| 20         | 2975       | 1625.0     |
| 20         | 3000       | 1968.75    |
| 20         | 3000       | 2175.0     |
| 30         | 950        | 950.0      |
| 30         | 1250       | 1100.0     |
| 30         | 1250       | 1150.0     |
| 30         | 1500       | 1237.5     |
| 30         | 1600       | 1310.0     |
| 30         | 2850       | 1566.6666666666667 |
+------------+------------+------------+
```

- Example 3: In Hive-compatible mode, use the deptno column to define a window and calculate the average value of the sal column. The order by clause is specified. This function returns the cumulative average value of the values from the first row to the row that has the same sal value as the current row in the current window. The average values for the rows that have the same sal value are the same. The current window includes the rows that have the same deptno value. Sample statements:

```
-- Enable the Hive-compatible mode.
set odps.sql.hive.compatible=true;
-- Execute the following statement:
select deptno, sal, avg(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1300.0     |          -- This row is the first row of this
window. The average value for the first row is the cumulative average value of the valu
es in the first and second rows because the two rows have the same sal value.
| 10         | 1300       | 1300.0     |          -- The return value is the cumulative
average value of the values in the first and second rows.
| 10         | 2450       | 1875.0     |          -- The average value for the third ro
w is the cumulative average value of the values from the first row to the fourth row be
cause the third and fourth rows have the same sal value.
| 10         | 2450       | 1875.0     |          -- The return value is the cumulative
average value of the values from the first row to the fourth row.
| 10         | 5000       | 2916.6666666666665 |
| 10         | 5000       | 2916.6666666666665 |
| 20         | 800        | 800.0      |
| 20         | 1100       | 950.0      |
| 20         | 2975       | 1625.0     |
| 20         | 3000       | 2175.0     |
| 20         | 3000       | 2175.0     |
| 30         | 950        | 950.0      |
| 30         | 1250       | 1150.0     |
| 30         | 1250       | 1150.0     |
| 30         | 1500       | 1237.5     |
| 30         | 1600       | 1310.0     |
| 30         | 2850       | 1566.6666666666667 |
+------------+------------+------------+
```

CLUSTER_SAMPLE

- Syntax

```
boolean cluster_sample(bigint <N>) OVER ([partition_clause])
boolean cluster_sample(bigint <N>, bigint <M>) OVER ([partition_clause])
```

- Description
  - `cluster_sample(bigint <N>)`: indicates that random N rows are sampled.

- `cluster_sample(bigint <N>, bigint <M>)`: indicates that rows are sampled based on a specified ratio (M/N). The number of rows that are sampled is calculated by using the following formula: `partition_row_count × M/N`. `partition_row_count` indicates the number of rows in a partition.

- Parameters
  - N: required. A constant of the BIGINT type. If N is set to NULL, NULL is returned.
  - M: required. A constant of the BIGINT type. If M is set to NULL, NULL is returned.
  - partition_clause: optional. For more information, see windowing_definition.

- Return value

  A value of the BOOLEAN type is returned.

- Examples

  If you want to sample about 20% of data entries in each group, execute the following statement:

```
select deptno, sal
    from (
        select deptno, sal, cluster_sample(5, 1) over (partition by deptno) as flag
        from emp
        ) sub
    where flag = true;
```

  The following result is returned:

```
+------------+------------+
| deptno     | sal        |
+------------+------------+
| 10         | 1300       |
| 20         | 3000       |
| 30         | 950        |
+------------+------------+
```

COUNT

- Syntax

```
bigint count(*) over ([partition_clause] [orderby_clause] [frame_clause])
bigint count([distinct] <expr>[,...]) over ([partition_clause] [orderby_clause] [frame_cl
ause])
```

- Description
  - `count(*)`: the total number of rows to return.
  - `count([distinct] <expr>[,...])`: When you calculate the number of rows, the rows whose expr is NULL are ignored. If multiple expr parameters exist, any rows whose expr is NULL are ignored. In addition, if the distinct keyword is specified, the number of rows after deduplication is calculated. Any rows whose expr is NULL are ignored.

- Parameters
  - expr: required. This parameter specifies the column whose values you want to count. All data types are supported. If the input value is NULL, the row that contains this value is not used for the calculation. If the DISTINCT keyword is specified, the count value of distinct values is obtained.

- partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the BIGINT type is returned.

- Examples
  - Example 1: Use the sal column to define a window. The order by clause is not specified. This function returns the cumulative count from the first row to the last row in the current window. The current window includes the rows that have the same sal value. Sample statement:

    ```
    select sal, count(sal) over (partition by sal) as count from emp;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | sal        | count      |
    +------------+------------+
    | 800        | 1          |
    | 950        | 1          |
    | 1100       | 1          |
    | 1250       | 2          |   -- This row is the first row of this window. The value of
    COUNT for the first row is the cumulative count for the second row because the two rows
    have the same sal value.
    | 1250       | 2          |   -- The return value is the cumulative count from the firs
    t row to the second row in the current window.
    | 1300       | 2          |
    | 1300       | 2          |
    | 1500       | 1          |
    | 1600       | 1          |
    | 2450       | 2          |
    | 2450       | 2          |
    | 2850       | 1          |
    | 2975       | 1          |
    | 3000       | 2          |
    | 3000       | 2          |
    | 5000       | 2          |
    | 5000       | 2          |
    +------------+------------+
    ```

- Example 2: In non-Hive-compatible mode, use the sal column to define a window. The order by clause is specified. This function returns the cumulative count from the first row to the current row in the current window. The current window includes the rows that have the same sal value. Sample statements:

```
-- Disable the Hive-compatible mode.
set odps.sql.hive.compatible=false;
-- Execute the following statement:
select sal, count(sal) over (partition by sal order by sal) as count from emp;
```

The following result is returned:

```
+------------+------------+
| sal        | count      |
+------------+------------+
| 800        | 1          |
| 950        | 1          |
| 1100       | 1          |
| 1250       | 1          |   -- This row is the first row of this window. The cumulati
ve count for the first row is 1.
| 1250       | 2          |   -- The cumulative count for the second row is 2.
| 1300       | 1          |
| 1300       | 2          |
| 1500       | 1          |
| 1600       | 1          |
| 2450       | 1          |
| 2450       | 2          |
| 2850       | 1          |
| 2975       | 1          |
| 3000       | 1          |
| 3000       | 2          |
| 5000       | 1          |
| 5000       | 2          |
+------------+------------+
```

- Example 3: In Hive-compatible mode, use the sal column to define a window. The order by clause is specified. This function returns the cumulative count from the first row to the last row in the current window. The current window includes the rows that have the same sal value. Sample statements:

```
-- Enable the Hive-compatible mode.
set odps.sql.hive.compatible=true;
-- Execute the following statement:
select sal, count(sal) over (partition by sal order by sal) as count from emp;
```

The following result is returned:

```
+------------+------------+
| sal        | count      |
+------------+------------+
| 800        | 1          |
| 950        | 1          |
| 1100       | 1          |
| 1250       | 2          |   -- This row is the first row of this window. The value of
COUNT for the first row is the cumulative count for the second row because the two rows
have the same sal value.
| 1250       | 2          |   -- The return value is the cumulative count from the firs
t row to the second row in the current window.
| 1300       | 2          |
| 1300       | 2          |
| 1500       | 1          |
| 1600       | 1          |
| 2450       | 2          |
| 2450       | 2          |
| 2850       | 1          |
| 2975       | 1          |
| 3000       | 2          |
| 3000       | 2          |
| 5000       | 2          |
| 5000       | 2          |
+------------+------------+
```

CUME_DIST

- Syntax

```
double cume_dist() over([partition_clause] [orderby_clause])
```

- Description

  This function calculates the cumulative distribution. The cumulative distribution indicates the ratio of rows whose values are greater than or equal to the values of the current row to all rows in a partition. The ratio is determined based on orderby_clause.

- Parameters

  partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

A value of the DOUBLE type is returned. The return value is calculated by using the following formula: `row_number_of_last_peer/partition_row_count`. `row_number_of_last_peer` indicates the value returned by ROW_NUMBER that corresponds to the last row of the group to which the current row belongs. `partition_row_count` indicates the number of rows in a partition to which the current row belongs.

- Examples

  Group all employees based on the deptno column and calculate the cumulative distribution of employees in each group by salary. Sample statement:

```
select deptno, ename, sal, concat(round(cume_dist() over (partition by deptno order by sa
l desc)*100,2),'%') as cume_dist from emp;
```

  The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | cume_dist  |
+------------+------------+------------+------------+
| 10         | JACCKA     | 5000       | 33.33%     |
| 10         | KING       | 5000       | 33.33%     |
| 10         | CLARK      | 2450       | 66.67%     |
| 10         | WELAN      | 2450       | 66.67%     |
| 10         | TEBAGE     | 1300       | 100.0%     |
| 10         | MILLER     | 1300       | 100.0%     |
| 20         | SCOTT      | 3000       | 40.0%      |
| 20         | FORD       | 3000       | 40.0%      |
| 20         | JONES      | 2975       | 60.0%      |
| 20         | ADAMS      | 1100       | 80.0%      |
| 20         | SMITH      | 800        | 100.0%     |
| 30         | BLAKE      | 2850       | 16.67%     |
| 30         | ALLEN      | 1600       | 33.33%     |
| 30         | TURNER     | 1500       | 50.0%      |
| 30         | MARTIN     | 1250       | 83.33%     |
| 30         | WARD       | 1250       | 83.33%     |
| 30         | JAMES      | 950        | 100.0%     |
+------------+------------+------------+------------+
```

DENSE_RANK

- Syntax

```
bigint dense_rank() over ([partition_clause] [orderby_clause])
```

- Description

  This function calculates the rank of a row in an ordered group of rows that are sorted based on orderby_clause. The rank counts from 1. In a partition, rows with the same value of the column that is specified in `order by` have the same rank. The rank increases by 1 each time the value of the column that is specified in `order by` changes.

- Parameters

  partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

A value of the BIGINT type is returned. If orderby_clause is not specified, the values in the returned results are all 1.

- Examples

  Group all employees based on the deptno column. In each group, sort the employees in descending order based on the sal value to obtain the sequence numbers of the employees in their respective groups. Sample statement:

```
select deptno, ename, sal, dense_rank() over (partition by deptno order by sal desc) as n
ums from emp;
```

  The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | nums       |
+------------+------------+------------+------------+
| 10         | JACCKA     | 5000       | 1          |
| 10         | KING       | 5000       | 1          |
| 10         | CLARK      | 2450       | 2          |
| 10         | WELAN      | 2450       | 2          |
| 10         | TEBAGE     | 1300       | 3          |
| 10         | MILLER     | 1300       | 3          |
| 20         | SCOTT      | 3000       | 1          |
| 20         | FORD       | 3000       | 1          |
| 20         | JONES      | 2975       | 2          |
| 20         | ADAMS      | 1100       | 3          |
| 20         | SMITH      | 800        | 4          |
| 30         | BLAKE      | 2850       | 1          |
| 30         | ALLEN      | 1600       | 2          |
| 30         | TURNER     | 1500       | 3          |
| 30         | MARTIN     | 1250       | 4          |
| 30         | WARD       | 1250       | 4          |
| 30         | JAMES      | 950        | 5          |
+------------+------------+------------+------------+
```

FIRST_VALUE

- Syntax

```
first_value(<expr>[, <ignore_nulls>]) over ([partition_clause] [orderby_clause] [frame_cl
ause])
```

- Description

  This function returns the value of expr that corresponds to the first row of a window.

- Parameters
  - expr: required. The expression that is used to calculate the returned result.
  - ignore_nulls: optional. A value of the BOOLEAN type. This parameter specifies whether to ignore NULL values. Default value: False. If this parameter is set to True, a non-NULL value of expr that corresponds to the first row of a window is returned.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

A value of the same data type as expr is returned.

- Examples

  Group all employees by department and return the first row of data in each group. Sample statement:

  - order by is not specified.

```
select deptno, ename, sal, first_value(sal) over (partition by deptno) as first_value f
rom emp;
```

The following result is returned:

```
+------------+------------+------------+-------------+
| deptno     | ename      | sal        | first_value |
+------------+------------+------------+-------------+
| 10         | TEBAGE     | 1300       | 1300        |   -- This row is the first row o
f the current window.
| 10         | CLARK      | 2450       | 1300        |
| 10         | KING       | 5000       | 1300        |
| 10         | MILLER     | 1300       | 1300        |
| 10         | JACCKA     | 5000       | 1300        |
| 10         | WELAN      | 2450       | 1300        |
| 20         | FORD       | 3000       | 3000        |   -- This row is the first row o
f the current window.
| 20         | SCOTT      | 3000       | 3000        |
| 20         | SMITH      | 800        | 3000        |
| 20         | ADAMS      | 1100       | 3000        |
| 20         | JONES      | 2975       | 3000        |
| 30         | TURNER     | 1500       | 1500        |   -- This row is the first row o
f the current window.
| 30         | JAMES      | 950        | 1500        |
| 30         | ALLEN      | 1600       | 1500        |
| 30         | WARD       | 1250       | 1500        |
| 30         | MARTIN     | 1250       | 1500        |
| 30         | BLAKE      | 2850       | 1500        |
+------------+------------+------------+-------------+
```

- order by is specified.

```
select deptno, ename, sal, first_value(sal) over (partition by deptno order by sal desc
) as first_value from emp;
```

The following result is returned:

```
+------------+------------+------------+-------------+
| deptno     | ename      | sal        | first_value |
+------------+------------+------------+-------------+
| 10         | JACCKA     | 5000       | 5000        |   -- This row is the first row o
f the current window.
| 10         | KING       | 5000       | 5000        |
| 10         | CLARK      | 2450       | 5000        |
| 10         | WELAN      | 2450       | 5000        |
| 10         | TEBAGE     | 1300       | 5000        |
| 10         | MILLER     | 1300       | 5000        |
| 20         | SCOTT      | 3000       | 3000        |   -- This row is the first row o
f the current window.
| 20         | FORD       | 3000       | 3000        |
| 20         | JONES      | 2975       | 3000        |
| 20         | ADAMS      | 1100       | 3000        |
| 20         | SMITH      | 800        | 3000        |
| 30         | BLAKE      | 2850       | 2850        |   -- This row is the first row o
f the current window.
| 30         | ALLEN      | 1600       | 2850        |
| 30         | TURNER     | 1500       | 2850        |
| 30         | MARTIN     | 1250       | 2850        |
| 30         | WARD       | 1250       | 2850        |
| 30         | JAMES      | 950        | 2850        |
+------------+------------+------------+-------------+
```

LAG

- Syntax

```
lag(<expr>[, bigint <offset>[, <default>]]) over([partition_clause] orderby_clause)
```

- Description

  This function returns the value of expr that precedes the current row at a given offset.

- Parameters
  - expr: required. The expression that is used to calculate the returned result.
  - offset: optional. The value is a constant of the BIGINT type and must be greater than or equal to 0. The value 0 indicates the current row, and the value 1 indicates the previous row. Default value: 1. If the input value is of the STRING or DOUBLE type, it is implicitly converted into a value of the BIGINT type before calculation.
  - default: optional. The default value when the value of offset is out of the valid range. The value of this parameter must be a constant. The default value of this parameter is NULL. The value of this parameter must be of the same data type as the value of expr. If the value of expr is not a constant, the parameter value is determined based on the current row.
  - partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the same data type as expr is returned.

- Examples

  Group all employees based on the deptno column and calculate the value of sal for each employee at a given offset. Sample statement:

```
select deptno, ename, sal, lag(sal, 1) over (partition by deptno order by sal) as sal_new
from emp;
```

  The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | sal_new    |
+------------+------------+------------+------------+
| 10         | TEBAGE     | 1300       | NULL       |
| 10         | MILLER     | 1300       | 1300       |
| 10         | CLARK      | 2450       | 1300       |
| 10         | WELAN      | 2450       | 2450       |
| 10         | KING       | 5000       | 2450       |
| 10         | JACCKA     | 5000       | 5000       |
| 20         | SMITH      | 800        | NULL       |
| 20         | ADAMS      | 1100       | 800        |
| 20         | JONES      | 2975       | 1100       |
| 20         | SCOTT      | 3000       | 2975       |
| 20         | FORD       | 3000       | 3000       |
| 30         | JAMES      | 950        | NULL       |
| 30         | MARTIN     | 1250       | 950        |
| 30         | WARD       | 1250       | 1250       |
| 30         | TURNER     | 1500       | 1250       |
| 30         | ALLEN      | 1600       | 1500       |
| 30         | BLAKE      | 2850       | 1600       |
+------------+------------+------------+------------+
```

LAST_VALUE

- Syntax

```
last_value(<expr>[, <ignore_nulls>]) over([partition_clause] [orderby_clause] [frame_clau
se])
```

- Description

  This function returns the value of expr that corresponds to the last row of a window.

- Parameters
  - expr: required. The expression that is used to calculate the returned result.
  - ignore_nulls: optional. A value of the BOOLEAN type. This parameter specifies whether to ignore NULL values. Default value: False. If this parameter is set to True, a non-NULL value of expr that corresponds to the last row of a window is returned.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

A value of the same data type as expr is returned.

- Examples

  Group all employees by department and return the last row of data in each group. Sample statement:

  - If order by is not specified, the rows from the first row to the last row belong to the current window. The value of the last row in the current window is returned.

    ```
    select deptno, ename, sal, last_value(sal) over (partition by deptno) as last_value fro
    m emp;
    ```

    The following result is returned:

    ```
    +------------+------------+------------+-------------+
    | deptno     | ename      | sal        | last_value |
    +------------+------------+------------+-------------+
    | 10         | TEBAGE     | 1300       | 2450        |
    | 10         | CLARK      | 2450       | 2450        |
    | 10         | KING       | 5000       | 2450        |
    | 10         | MILLER     | 1300       | 2450        |
    | 10         | JACCKA     | 5000       | 2450        |
    | 10         | WELAN      | 2450       | 2450        |   -- This row is the last row of
    the current window.
    | 20         | FORD       | 3000       | 2975        |
    | 20         | SCOTT      | 3000       | 2975        |
    | 20         | SMITH      | 800        | 2975        |
    | 20         | ADAMS      | 1100       | 2975        |
    | 20         | JONES      | 2975       | 2975        |   -- This row is the last row of
    the current window.
    | 30         | TURNER     | 1500       | 2850        |
    | 30         | JAMES      | 950        | 2850        |
    | 30         | ALLEN      | 1600       | 2850        |
    | 30         | WARD       | 1250       | 2850        |
    | 30         | MARTIN     | 1250       | 2850        |
    | 30         | BLAKE      | 2850       | 2850        |   -- This row is the last row of
    the current window.
    +------------+------------+------------+-------------+
    ```

  - If order by is specified, the rows from the first row to the current row belong to the current window. The value of the current row in the current window is returned.

    ```
    select deptno, ename, sal, last_value(sal) over (partition by deptno order by sal desc)
    as last_value from emp;
    ```

    The following result is returned:

```
+------------+------------+------------+-------------+
| deptno     | ename      | sal        | last_value |
+------------+------------+------------+-------------+
| 10         | JACCKA     | 5000       | 5000        |   -- This row is the current row
of the current window.
| 10         | KING       | 5000       | 5000        |   -- This row is the current row
of the current window.
| 10         | CLARK      | 2450       | 2450        |   -- This row is the current row
of the current window.
| 10         | WELAN      | 2450       | 2450        |   -- This row is the current row
of the current window.
| 10         | TEBAGE     | 1300       | 1300        |   -- This row is the current row
of the current window.
| 10         | MILLER     | 1300       | 1300        |   -- This row is the current row
of the current window.
| 20         | SCOTT      | 3000       | 3000        |   -- This row is the current row
of the current window.
| 20         | FORD       | 3000       | 3000        |   -- This row is the current row
of the current window.
| 20         | JONES      | 2975       | 2975        |   -- This row is the current row
of the current window.
| 20         | ADAMS      | 1100       | 1100        |   -- This row is the current row
of the current window.
| 20         | SMITH      | 800        | 800         |   -- This row is the current row
of the current window.
| 30         | BLAKE      | 2850       | 2850        |   -- This row is the current row
of the current window.
| 30         | ALLEN      | 1600       | 1600        |   -- This row is the current row
of the current window.
| 30         | TURNER     | 1500       | 1500        |   -- This row is the current row
of the current window.
| 30         | MARTIN     | 1250       | 1250        |   -- This row is the current row
of the current window.
| 30         | WARD       | 1250       | 1250        |   -- This row is the current row
of the current window.
| 30         | JAMES      | 950        | 950         |   -- This row is the current row
of the current window.
+------------+------------+------------+-------------+
```

LEAD

- Syntax

```
lead(<expr>[, bigint <offset>[, <default>]]) over([partition_clause] orderby_clause)
```

- Description

  This function returns the value of expr that corresponds to the Nth row following the current row at a given offset.

- Parameters
  - expr: required. The expression that is used to calculate the returned result.

- offset: optional. The value is a constant of the BIGINT type and must be greater than or equal to 0. The value 0 indicates the current row, and the value 1 indicates the next row. Default value: 1. If the input value is of the STRING or DOUBLE type, it is implicitly converted into a value of the BIGINT type before calculation.
- default: optional. The default value when the value of offset is out of the valid range. The value of this parameter must be a constant. The default value of this parameter is NULL. The value of this parameter must be of the same data type as the value of expr. If the value of expr is not a constant, the parameter value is determined based on the current row.
- partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the same data type as expr is returned.

- Examples

  Group all employees based on the deptno column and calculate the value of sal for each employee at a given offset. Sample statement:

```
select deptno, ename, sal, lead(sal, 1) over (partition by deptno order by sal) as sal_ne
w from emp;
```

  The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | sal_new    |
+------------+------------+------------+------------+
| 10         | TEBAGE     | 1300       | 1300       |
| 10         | MILLER     | 1300       | 2450       |
| 10         | CLARK      | 2450       | 2450       |
| 10         | WELAN      | 2450       | 5000       |
| 10         | KING       | 5000       | 5000       |
| 10         | JACCKA     | 5000       | NULL       |
| 20         | SMITH      | 800        | 1100       |
| 20         | ADAMS      | 1100       | 2975       |
| 20         | JONES      | 2975       | 3000       |
| 20         | SCOTT      | 3000       | 3000       |
| 20         | FORD       | 3000       | NULL       |
| 30         | JAMES      | 950        | 1250       |
| 30         | MARTIN     | 1250       | 1250       |
| 30         | WARD       | 1250       | 1500       |
| 30         | TURNER     | 1500       | 1600       |
| 30         | ALLEN      | 1600       | 2850       |
| 30         | BLAKE      | 2850       | NULL       |
+------------+------------+------------+------------+
```

MAX

- Syntax

```
max(<expr>) over([partition_clause] [orderby_clause] [frame_clause])
```

- Description

  This function returns the maximum value of expr in a window.

- Parameters
  - expr: required. The expression that is used to calculate the maximum value. The input value can be of any data type other than BOOLEAN. If the input value for a row is NULL, this row is not used for calculation.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.
- Return value

  A value of the same type as expr is returned.
- Examples
  - Example 1: Use the deptno column to define a window and obtain the maximum value of the sal column. The order by clause is not specified. This function returns the maximum value of the current window. The current window includes the rows that have the same deptno value. Sample statement:

    ```
    select deptno, sal, max(sal) over (partition by deptno) from emp;
    ```

    The following result is returned:

    ```
    +------------+------------+------------+
    | deptno     | sal        | _c2        |
    +------------+------------+------------+
    | 10         | 1300       | 5000       |   -- This row is the first row of this window.
    The return value is the maximum value among the values from the first row to the sixth
    row.
    | 10         | 2450       | 5000       |   -- The return value is the maximum value amo
    ng the values from the first row to the sixth row.
    | 10         | 5000       | 5000       |   -- The return value is the maximum value amo
    ng the values from the first row to the sixth row.
    | 10         | 1300       | 5000       |
    | 10         | 5000       | 5000       |
    | 10         | 2450       | 5000       |
    | 20         | 3000       | 3000       |
    | 20         | 3000       | 3000       |
    | 20         | 800        | 3000       |
    | 20         | 1100       | 3000       |
    | 20         | 2975       | 3000       |
    | 30         | 1500       | 2850       |
    | 30         | 950        | 2850       |
    | 30         | 1600       | 2850       |
    | 30         | 1250       | 2850       |
    | 30         | 1250       | 2850       |
    | 30         | 2850       | 2850       |
    +------------+------------+------------+
    ```

- Example 2: Use the deptno column to define a window and obtain the maximum value of the sal column. The order by clause is specified. This function returns the maximum value among the values from the first row to the current row in the current window. The current window includes the rows that have the same deptno value. Sample statement:

```
select deptno, sal, max(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1300       |   -- This row is the first row of this window.

| 10         | 1300       | 1300       |   -- The return value is the maximum value amo
ng the values in the first and second rows.
| 10         | 2450       | 2450       |   -- The return value is the maximum value amo
ng the values from the first row to the third row.
| 10         | 2450       | 2450       |   -- The return value is the maximum value amo
ng the values from the first row to the fourth row.
| 10         | 5000       | 5000       |
| 10         | 5000       | 5000       |
| 20         | 800        | 800        |
| 20         | 1100       | 1100       |
| 20         | 2975       | 2975       |
| 20         | 3000       | 3000       |
| 20         | 3000       | 3000       |
| 30         | 950        | 950        |
| 30         | 1250       | 1250       |
| 30         | 1250       | 1250       |
| 30         | 1500       | 1500       |
| 30         | 1600       | 1600       |
| 30         | 2850       | 2850       |
+------------+------------+------------+
```

MEDIAN

- Syntax

```
median(<expr>) over ([partition_clause] [orderby_clause] [frame_clause])
```

- Description

  This function returns the median of expr in a window.

- Parameters
  - expr: required. The expression that is used to calculate the median. A value of the DOUBLE or DECIMAL type. The value of this parameter must be 1 to 255 digits in length.
    - If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If it is of another data type, an error is returned.
    - If the input value is NULL, NULL is returned.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the DOUBLE or DECIMAL type is returned. If the values of all expressions that are specified byexpr are NULL, NULL is returned.

- Examples

  Use the deptno column to define a window and calculate the median value of the sal column. This function returns the median value of the current window. The current window includes the rows that have the same deptno value. Sample statement:

```
select deptno, sal, median(sal) over (partition by deptno) from emp;
```

  The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 2450.0     |   -- This row is the first row of this window. T
he return value is the median value of the values from the first row to the sixth row.
| 10         | 2450       | 2450.0     |
| 10         | 5000       | 2450.0     |
| 10         | 1300       | 2450.0     |
| 10         | 5000       | 2450.0     |
| 10         | 2450       | 2450.0     |
| 20         | 3000       | 2975.0     |
| 20         | 3000       | 2975.0     |
| 20         | 800        | 2975.0     |
| 20         | 1100       | 2975.0     |
| 20         | 2975       | 2975.0     |
| 30         | 1500       | 1375.0     |
| 30         | 950        | 1375.0     |
| 30         | 1600       | 1375.0     |
| 30         | 1250       | 1375.0     |
| 30         | 1250       | 1375.0     |
| 30         | 2850       | 1375.0     |
+------------+------------+------------+
```

MIN

- Syntax

```
min(<expr>) over([partition_clause] [orderby_clause] [frame_clause])
```

- Description

  This function returns the minimum value of expr in a window.

- Parameters
  - expr: required. The expression that is used to calculate the minimum value. The input value can be of any data type other than BOOLEAN. If the value for a row is NULL, this row is not used for calculation.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

A value of the same data type as expr is returned.

- Examples
  - Example 1: Use the deptno column to define a window and obtain the minimum value of the sal column. The order by clause is not specified. This function returns the minimum value of the current window. The current window includes the rows that have the same deptno value. Sample statement:

```
select deptno, sal, min(sal) over (partition by deptno) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1300       |   -- This row is the first row of this window.
The return value is the minimum value among the values from the first row to the sixth
row.
| 10         | 2450       | 1300       |   -- The return value is the minimum value amo
ng the values from the first row to the sixth row.
| 10         | 5000       | 1300       |   -- The return value is the minimum value amo
ng the values from the first row to the sixth row.
| 10         | 1300       | 1300       |
| 10         | 5000       | 1300       |
| 10         | 2450       | 1300       |
| 20         | 3000       | 800        |
| 20         | 3000       | 800        |
| 20         | 800        | 800        |
| 20         | 1100       | 800        |
| 20         | 2975       | 800        |
| 30         | 1500       | 950        |
| 30         | 950        | 950        |
| 30         | 1600       | 950        |
| 30         | 1250       | 950        |
| 30         | 1250       | 950        |
| 30         | 2850       | 950        |
+------------+------------+------------+
```

- Example 2: Use the deptno column to define a window and obtain the minimum value of the sal column. The order by clause is specified. This function returns the minimum value among the values from the first row to the current row in the current window. The current window includes the rows that have the same deptno value. Sample statement:

```
select deptno, sal, min(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1300       |   -- This row is the first row of this window.

| 10         | 1300       | 1300       |   -- The return value is the minimum value amo
ng the values in the first and second rows.
| 10         | 2450       | 1300       |   -- The return value is the minimum value amo
ng the values from the first row to the third row.
| 10         | 2450       | 1300       |
| 10         | 5000       | 1300       |
| 10         | 5000       | 1300       |
| 20         | 800        | 800        |
| 20         | 1100       | 800        |
| 20         | 2975       | 800        |
| 20         | 3000       | 800        |
| 20         | 3000       | 800        |
| 30         | 950        | 950        |
| 30         | 1250       | 950        |
| 30         | 1250       | 950        |
| 30         | 1500       | 950        |
| 30         | 1600       | 950        |
| 30         | 2850       | 950        |
+------------+------------+------------+
```

NTH_VALUE

- Syntax

```
nth_value(<expr>, <number> [, <ignore_nulls>]) over ([partition_clause] [orderby_clause]
[frame_clause])
```

- Description

  This function returns the value of expr that corresponds to the Nth row in a window.

- Parameters
  - expr: required. The expression that is used to calculate the returned result.
  - number: required. The value is of the BIGINT type and must be an integer greater than or equal to 1. If the input value is 1, this function is equivalent to FIRST_VALUE.
  - ignore_nulls: optional. A value of the BOOLEAN type. This parameter specifies whether to ignore NULL values. Default value: False. If this parameter is set to True, a non-NULL value of expr that corresponds to the Nth row of a window is returned.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the same data type as expr is returned.

- Examples

  Group all employees by department and return the sixth row of data in each group. Sample statement:

  - If order by is not specified, the rows from the first row to the last row belong to the current window. The value of the sixth row in the current window is returned.

    ```
    select deptno, ename, sal, nth_value(sal,6) over (partition by deptno) as nth_value fro
    m emp;
    ```

    The following result is returned:

    ```
    +------------+------------+------------+------------+
    | deptno     | ename      | sal        | nth_value  |
    +------------+------------+------------+------------+
    | 10         | TEBAGE     | 1300       | 2450       |
    | 10         | CLARK      | 2450       | 2450       |
    | 10         | KING       | 5000       | 2450       |
    | 10         | MILLER     | 1300       | 2450       |
    | 10         | JACCKA     | 5000       | 2450       |
    | 10         | WELAN      | 2450       | 2450       |   -- This row is the sixth row of
    the current window.
    | 20         | FORD       | 3000       | NULL       |
    | 20         | SCOTT      | 3000       | NULL       |
    | 20         | SMITH      | 800        | NULL       |
    | 20         | ADAMS      | 1100       | NULL       |
    | 20         | JONES      | 2975       | NULL       |   -- This current window has less
    than six rows, and NULL is returned.
    | 30         | TURNER     | 1500       | 2850       |
    | 30         | JAMES      | 950        | 2850       |
    | 30         | ALLEN      | 1600       | 2850       |
    | 30         | WARD       | 1250       | 2850       |
    | 30         | MARTIN     | 1250       | 2850       |
    | 30         | BLAKE      | 2850       | 2850       |   -- This row is the sixth row of
    the current window.
    +------------+------------+------------+------------+
    ```

- If order by is specified, the rows from the first row to the current row belong to the current window. The value of the sixth row in the current window is returned.

```
select deptno, ename, sal, nth_value(sal,6) over (partition by deptno order by sal) as
nth_value from emp;
```

The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | nth_value  |
+------------+------------+------------+------------+
| 10         | TEBAGE     | 1300       | NULL       |
| 10         | MILLER     | 1300       | NULL       |   -- This window has only two row
s, and NULL is returned.
| 10         | CLARK      | 2450       | NULL       |
| 10         | WELAN      | 2450       | NULL       |
| 10         | KING       | 5000       | 5000       |
| 10         | JACCKA     | 5000       | 5000       |
| 20         | SMITH      | 800        | NULL       |
| 20         | ADAMS      | 1100       | NULL       |
| 20         | JONES      | 2975       | NULL       |
| 20         | SCOTT      | 3000       | NULL       |
| 20         | FORD       | 3000       | NULL       |
| 30         | JAMES      | 950        | NULL       |
| 30         | MARTIN     | 1250       | NULL       |
| 30         | WARD       | 1250       | NULL       |
| 30         | TURNER     | 1500       | NULL       |
| 30         | ALLEN      | 1600       | NULL       |
| 30         | BLAKE      | 2850       | 2850       |
+------------+------------+------------+------------+
```

NTILE

- Syntax

```
bigint ntile(bigint <N>) over ([partition_clause] [orderby_clause])
```

- Description

  This function splits rows of data in a partition into N groups of equal size and returns the number of the group to which the specified row belongs. If data in the partition cannot be split into N groups of equal size, one more row is preferentially allocated to the first M groups.

- Parameters
  - N: required. This parameter specifies the number of splits. The input value is of the BIGINT type.
  - partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the BIGINT type is returned.

- Examples

  Divide all employees into three groups based on the sal column in descending order and obtain the number of the group to which each employee belongs. Sample statement:

```
select deptno, ename, sal, ntile(3) over (partition by deptno order by sal desc) as nt3 f
rom emp;
```

The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | nt3        |
+------------+------------+------------+------------+
| 10         | JACCKA     | 5000       | 1          |
| 10         | KING       | 5000       | 1          |
| 10         | CLARK      | 2450       | 2          |
| 10         | WELAN      | 2450       | 2          |
| 10         | TEBAGE     | 1300       | 3          |
| 10         | MILLER     | 1300       | 3          |
| 20         | SCOTT      | 3000       | 1          |
| 20         | FORD       | 3000       | 1          |
| 20         | JONES      | 2975       | 2          |
| 20         | ADAMS      | 1100       | 2          |
| 20         | SMITH      | 800        | 3          |
| 30         | BLAKE      | 2850       | 1          |
| 30         | ALLEN      | 1600       | 1          |
| 30         | TURNER     | 1500       | 2          |
| 30         | MARTIN     | 1250       | 2          |
| 30         | WARD       | 1250       | 3          |
| 30         | JAMES      | 950        | 3          |
+------------+------------+------------+------------+
```

PERCENT_RANK

- Syntax

```
double percent_rank() over([partition_clause] [orderby_clause])
```

- Description

  This function calculates the percentile rank of the current row in a partition based on orderby_clause.

- Parameters

  partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the DOUBLE type is returned. The valid value range is [0.0, 1.0]. The return value is calculated by using the following formula: `(rank - 1)/(partition_row_count - 1)`. `rank` indicates the return value of the RANK function that corresponds to the current row. `partition_row_count` indicates the number of rows in the partition to which the current row belongs. If the partition contains only one row of data, 0.0 is returned.

- Examples

  Calculate the percentile rank of each employee in a group based on the sal column. Sample statement:

```
select deptno, ename, sal, percent_rank(sal) over (partition by deptno order by sal desc)
as sal_new from emp;
```

The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | sal_new    |
+------------+------------+------------+------------+
| 10         | JACCKA     | 5000       | 0.0        |
| 10         | KING       | 5000       | 0.0        |
| 10         | CLARK      | 2450       | 0.4        |
| 10         | WELAN      | 2450       | 0.4        |
| 10         | TEBAGE     | 1300       | 0.8        |
| 10         | MILLER     | 1300       | 0.8        |
| 20         | SCOTT      | 3000       | 0.0        |
| 20         | FORD       | 3000       | 0.0        |
| 20         | JONES      | 2975       | 0.5        |
| 20         | ADAMS      | 1100       | 0.75       |
| 20         | SMITH      | 800        | 1.0        |
| 30         | BLAKE      | 2850       | 0.0        |
| 30         | ALLEN      | 1600       | 0.2        |
| 30         | TURNER     | 1500       | 0.4        |
| 30         | MARTIN     | 1250       | 0.6        |
| 30         | WARD       | 1250       | 0.6        |
| 30         | JAMES      | 950        | 1.0        |
+------------+------------+------------+------------+
```

RANK

- Syntax

```
bigint rank() over ([partition_clause] [orderby_clause])
```

- Description

  This function returns the rank of the current row in a partition based on the order specified by orderby_clause. The rank counts from 1.

- Parameters

  partition_clause and orderby_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the BIGINT type is returned. The return value may be duplicate and inconsecutive. The return value is the sequence number of the first row in the group to which the current row belongs. The sequence number of the first row is calculated by using the ROW_NUMBER() function. If orderby_clause is not specified, the values in the returned results are all 1.

- Examples

  Group all employees based on the deptno column. In each group, sort the employees in descending order based on the sal value to obtain the sequence numbers of the employees in their respective groups. Sample statement:

```
select deptno, ename, sal, rank() over (partition by deptno order by sal desc) as nums fr
om emp;
```

  The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | nums       |
+------------+------------+------------+------------+
| 10         | JACCKA     | 5000       | 1          |
| 10         | KING       | 5000       | 1          |
| 10         | CLARK      | 2450       | 3          |
| 10         | WELAN      | 2450       | 3          |
| 10         | TEBAGE     | 1300       | 5          |
| 10         | MILLER     | 1300       | 5          |
| 20         | SCOTT      | 3000       | 1          |
| 20         | FORD       | 3000       | 1          |
| 20         | JONES      | 2975       | 3          |
| 20         | ADAMS      | 1100       | 4          |
| 20         | SMITH      | 800        | 5          |
| 30         | BLAKE      | 2850       | 1          |
| 30         | ALLEN      | 1600       | 2          |
| 30         | TURNER     | 1500       | 3          |
| 30         | MARTIN     | 1250       | 4          |
| 30         | WARD       | 1250       | 4          |
| 30         | JAMES      | 950        | 6          |
+------------+------------+------------+------------+
```

ROW_NUMBER

- Syntax

```
row_number() over([partition_clause] [orderby_clause])
```

- Description

  This function returns the sequence number of the current row in a partition. The sequence number counts from 1.

- Parameters

  For more information, see windowing_definition. frame_clause is not supported.

- Return value

  A value of the BIGINT type is returned.

- Examples

  Group all employees based on the deptno column. In each group, sort the employees in descending order based on the sal value to obtain the sequence numbers of the employees in their respective groups. Sample statement:

```
select deptno, ename, sal, row_number() over (partition by deptno order by sal desc) as n
ums from emp;
```

  The following result is returned:

```
+------------+------------+------------+------------+
| deptno     | ename      | sal        | nums       |
+------------+------------+------------+------------+
| 10         | JACCKA     | 5000       | 1          |
| 10         | KING       | 5000       | 2          |
| 10         | CLARK      | 2450       | 3          |
| 10         | WELAN      | 2450       | 4          |
| 10         | TEBAGE     | 1300       | 5          |
| 10         | MILLER     | 1300       | 6          |
| 20         | SCOTT      | 3000       | 1          |
| 20         | FORD       | 3000       | 2          |
| 20         | JONES      | 2975       | 3          |
| 20         | ADAMS      | 1100       | 4          |
| 20         | SMITH      | 800        | 5          |
| 30         | BLAKE      | 2850       | 1          |
| 30         | ALLEN      | 1600       | 2          |
| 30         | TURNER     | 1500       | 3          |
| 30         | MARTIN     | 1250       | 4          |
| 30         | WARD       | 1250       | 5          |
| 30         | JAMES      | 950        | 6          |
+------------+------------+------------+------------+
```

STDDEV

- Syntax

```
double stddev|stddev_pop([distinct] <expr>) over ([partition_clause] [orderby_clause] [fr
ame_clause])
decimal stddev|stddev_pop([distinct] <expr>) over ([partition_clause] [orderby_clause] [f
rame_clause])
```

- Description

  This function returns the population standard deviation of all input values.

- Parameters
  - expr: required. The expression that is used to calculate the population standard deviation. A value of the DOUBLE or DECIMAL type.
    - If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If it is of another data type, an error is returned.
    - If the value for a row is NULL, this row is not used for calculation.
    - If the distinct keyword is specified, the population standard deviation of distinct values is calculated.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value

  A value of the same data type as expr is returned. If the values of all expressions that are specified by expr are NULL, NULL is returned.

- Examples

- Example 1: Use the deptno column to define a window and calculate the population standard deviation of the sal column. The order by clause is not specified. This function returns the cumulative population standard deviation of the current window. The current window includes the rows that have the same deptno value. Sample statement:

```
select deptno, sal, stddev(sal) over (partition by deptno) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1546.1421524412158 |   -- This row is the first row of this
window. The return value is the cumulative population standard deviation of the values
from the first row to the sixth row.
| 10         | 2450       | 1546.1421524412158 |   -- The return value is the cumulativ
e population standard deviation of the values from the first row to the sixth row.
| 10         | 5000       | 1546.1421524412158 |
| 10         | 1300       | 1546.1421524412158 |
| 10         | 5000       | 1546.1421524412158 |
| 10         | 2450       | 1546.1421524412158 |
| 20         | 3000       | 1004.7387720198718 |
| 20         | 3000       | 1004.7387720198718 |
| 20         | 800        | 1004.7387720198718 |
| 20         | 1100       | 1004.7387720198718 |
| 20         | 2975       | 1004.7387720198718 |
| 30         | 1500       | 610.1001739241042 |
| 30         | 950        | 610.1001739241042 |
| 30         | 1600       | 610.1001739241042 |
| 30         | 1250       | 610.1001739241042 |
| 30         | 1250       | 610.1001739241042 |
| 30         | 2850       | 610.1001739241042 |
+------------+------------+------------+
```

- Example 2: In non-Hive-compatible mode, use the deptno column to define a window and calculate the population standard deviation of the sal column. The order by clause is specified. This function returns the cumulative population standard deviation of the values from the first row to the current row in the current window. The current window includes the rows that have the same deptno value. Sample statements:

```
-- Disable the Hive-compatible mode.
set odps.sql.hive.compatible=false;
-- Execute the following statement:
select deptno, sal, stddev(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 0.0        |           -- This row is the first row of this
window.
| 10         | 1300       | 0.0        |           -- The return value is the cumulativ
e population standard deviation of the values in the first and second rows.
| 10         | 2450       | 542.1151989096865 |    -- The return value is the cumulativ
e population standard deviation of the values from the first row to the third row.
| 10         | 2450       | 575.0      |           -- The return value is the cumulativ
e population standard deviation of the values from the first row to the fourth row.
| 10         | 5000       | 1351.6656391282572 |
| 10         | 5000       | 1546.1421524412158 |
| 20         | 800        | 0.0        |
| 20         | 1100       | 150.0      |
| 20         | 2975       | 962.4188277460079 |
| 20         | 3000       | 1024.2947268730811 |
| 20         | 3000       | 1004.7387720198718 |
| 30         | 950        | 0.0        |
| 30         | 1250       | 150.0      |
| 30         | 1250       | 141.4213562373095 |
| 30         | 1500       | 194.8557158514987 |
| 30         | 1600       | 226.71568097509268 |
| 30         | 2850       | 610.1001739241042 |
+------------+------------+------------+
```

- Example 3: In Hive-compatible mode, use the deptno column to define a window and calculate the population standard deviation of the sal column. The order by clause is specified. This function returns the cumulative population standard deviation of the values from the first row to the row that has the same sal value as the current row in the current window. The population standard deviations for the rows that have the same sal value are the same. The current window includes the rows that have the same deptno value. Sample statements:

```
-- Enable the Hive-compatible mode.
set odps.sql.hive.compatible=true;
-- Execute the following statement:
select deptno, sal, stddev(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 0.0        |           -- This row is the first row of this
window. The population standard deviation for the first row is the cumulative populatio
n standard deviation of the values in the first and second rows because the two rows ha
ve the same sal value.
| 10         | 1300       | 0.0        |           -- The return value is the cumulativ
e population standard deviation of the values in the first and second rows.
| 10         | 2450       | 575.0      |           -- The population standard deviation
for the third row is the cumulative population standard deviation of the values from th
e first row to the fourth row because the third and fourth rows have the same sal value
.
| 10         | 2450       | 575.0      |           -- The return value is the cumulativ
e population standard deviation of the values from the first row to the fourth row.
| 10         | 5000       | 1546.1421524412158 |
| 10         | 5000       | 1546.1421524412158 |
| 20         | 800        | 0.0        |
| 20         | 1100       | 150.0      |
| 20         | 2975       | 962.4188277460079 |
| 20         | 3000       | 1004.7387720198718 |
| 20         | 3000       | 1004.7387720198718 |
| 30         | 950        | 0.0        |
| 30         | 1250       | 141.4213562373095 |
| 30         | 1250       | 141.4213562373095 |
| 30         | 1500       | 194.8557158514987 |
| 30         | 1600       | 226.71568097509268 |
| 30         | 2850       | 610.1001739241042 |
+------------+------------+------------+
```

STDDEV_SAMP

- Syntax

```
double stddev_samp([distinct] <expr>) over([partition_clause] [orderby_clause] [frame_cla
use])
decimal stddev_samp([distinct] <expr>) over([partition_clause] [orderby_clause] [frame_cl
ause])
```

- Description

This function returns the sample standard deviation of all input values.

- Parameters
  - expr: required. This parameter specifies the expression that is used to calculate the sample standard deviation. The value is of the DOUBLE or DECIMAL type.
    - If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If it is of another data type, an error is returned.
    - If the value for a row is NULL, this row is not used for the calculation.
    - If the distinct keyword is specified, the sample standard deviation of distinct values is calculated.
  - partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.
- Return value

  A value of the same data type as expr is returned. If the values of all expressions specified by expr are NULL, NULL is returned. If the window has only one row of data whose expr value is not NULL, 0 is returned.
- Examples
  - Example 1: Use the deptno column to define a window and calculate the sample standard deviation of the sal column. The order by clause is not specified. This function returns the cumulative sample standard deviation of the current window. The current window includes the rows that have the same deptno value. Sample statement:

    ```
    select deptno, sal, stddev_samp(sal) over (partition by deptno) from emp;
    ```

    The following result is returned:

    ```
    +------------+------------+------------+
    | deptno     | sal        | _c2        |
    +------------+------------+------------+
    | 10         | 1300       | 1693.7138680032904 |   -- This row is the first row of this
    window. The return value is the cumulative sample standard deviation of the values from
    the first row to the sixth row.
    | 10         | 2450       | 1693.7138680032904 |   -- The return value is the cumulativ
    e sample standard deviation of the values from the first row to the sixth row.
    | 10         | 5000       | 1693.7138680032904 |   -- The return value is the cumulativ
    e sample standard deviation of the values from the first row to the sixth row.
    | 10         | 1300       | 1693.7138680032904 |
    | 10         | 5000       | 1693.7138680032904 |
    | 10         | 2450       | 1693.7138680032904 |
    | 20         | 3000       | 1123.3320969330487 |
    | 20         | 3000       | 1123.3320969330487 |
    | 20         | 800        | 1123.3320969330487 |
    | 20         | 1100       | 1123.3320969330487 |
    | 20         | 2975       | 1123.3320969330487 |
    | 30         | 1500       | 668.331255192114 |
    | 30         | 950        | 668.331255192114 |
    | 30         | 1600       | 668.331255192114 |
    | 30         | 1250       | 668.331255192114 |
    | 30         | 1250       | 668.331255192114 |
    | 30         | 2850       | 668.331255192114 |
    +------------+------------+------------+
    ```

- Example 2: Use the deptno column to define a window and calculate the sample standard deviation of the sal column. The order by clause is specified. This function returns the cumulative sample standard deviation of the values from the first row to the current row in the current window. The current window includes the rows that have the same deptno value. Sample statement:

  ```
  select deptno, sal, stddev_samp(sal) over (partition by deptno order by sal) from emp;
  ```

  The following result is returned:

  ```
  +------------+------------+------------+
  | deptno     | sal        | _c2        |
  +------------+------------+------------+
  | 10         | 1300       | 0.0        |          -- This row is the first row of this
  window.
  | 10         | 1300       | 0.0        |          -- The return value is the cumulative
  sample standard deviation of the values in the first and second rows.
  | 10         | 2450       | 663.9528095680697 |   -- The return value is the cumulative
  sample standard deviation of the values from the first row to the third row.
  | 10         | 2450       | 663.9528095680696 |
  | 10         | 5000       | 1511.2081259707413 |
  | 10         | 5000       | 1693.7138680032904 |
  | 20         | 800        | 0.0        |
  | 20         | 1100       | 212.13203435596427 |
  | 20         | 2975       | 1178.7175234126282 |
  | 20         | 3000       | 1182.7536725793752 |
  | 20         | 3000       | 1123.3320969330487 |
  | 30         | 950        | 0.0        |
  | 30         | 1250       | 212.13203435596427 |
  | 30         | 1250       | 173.20508075688772 |
  | 30         | 1500       | 225.0      |
  | 30         | 1600       | 253.4758371127315 |
  | 30         | 2850       | 668.331255192114 |
  +------------+------------+------------+
  ```

SUM

- Syntax

  ```
  sum([distinct] <expr>) over ([partition_clause] [orderby_clause] [frame_clause])
  ```

- Description

  This function returns the sum of expr in a window.

- Parameters
  - expr: required. This parameter specifies the column whose sum you want to calculate. The column is of the DOUBLE, DECIMAL, or BIGINT type.
    - If an input value is of the STRING type, it is implicitly converted into a value of the DOUBLE type before calculation. If it is of another data type, an error is returned.
    - If the value for a row is NULL, this row is not used for calculation.
    - If the distinct keyword is specified, the sum of distinct values is calculated.

- partition_clause, orderby_clause, and frame_clause: For more information about these parameters, see windowing_definition.

- Return value
  - If input values are of the BIGINT type, a value of the BIGINT type is returned.
  - If input values are of the DECIMAL type, a value of the DECIMAL type is returned.
  - If input values are of the DOUBLE or STRING type, a value of the DOUBLE type is returned.
  - If input values are NULL, NULL is returned.

- Examples
  - Example 1: Use the deptno column to define a window and calculate the sum of the sal column. The order by clause is not specified. This function returns the cumulative sum of the current window. The current window includes the rows that have the same deptno value. Sample statement:

    ```
    select deptno, sal, sum(sal) over (partition by deptno) from emp;
    ```

    The following result is returned:

    ```
    +------------+------------+------------+
    | deptno     | sal        | _c2        |
    +------------+------------+------------+
    | 10         | 1300       | 17500      |   -- This row is the first row of this window.
    The return value is the cumulative sum of the values from the first row to the sixth ro
    w.
    | 10         | 2450       | 17500      |   -- The return value is the cumulative sum of
    the values from the first row to the sixth row.
    | 10         | 5000       | 17500      |   -- The return value is the cumulative sum of
    the values from the first row to the sixth row.
    | 10         | 1300       | 17500      |
    | 10         | 5000       | 17500      |
    | 10         | 2450       | 17500      |
    | 20         | 3000       | 10875      |
    | 20         | 3000       | 10875      |
    | 20         | 800        | 10875      |
    | 20         | 1100       | 10875      |
    | 20         | 2975       | 10875      |
    | 30         | 1500       | 9400       |
    | 30         | 950        | 9400       |
    | 30         | 1600       | 9400       |
    | 30         | 1250       | 9400       |
    | 30         | 1250       | 9400       |
    | 30         | 2850       | 9400       |
    +------------+------------+------------+
    ```

- Example 2: In non-Hive-compatible mode, use the deptno column to define a window and calculate the sum of the sal column. The order by clause is specified. This function returns the cumulative sum of the values from the first row to the current row in the current window. The current window includes the rows that have the same deptno value. Sample statements:

```
-- Disable the Hive-compatible mode.
set odps.sql.hive.compatible=false;
-- Execute the following statement:
select deptno, sal, sum(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 1300       |   -- This row is the first row of this window.

| 10         | 1300       | 2600       |   -- The return value is the cumulative sum of
the values in the first and second rows.
| 10         | 2450       | 5050       |   -- The return value is the cumulative sum of
the values from the first row to the third row.
| 10         | 2450       | 7500       |
| 10         | 5000       | 12500      |
| 10         | 5000       | 17500      |
| 20         | 800        | 800        |
| 20         | 1100       | 1900       |
| 20         | 2975       | 4875       |
| 20         | 3000       | 7875       |
| 20         | 3000       | 10875      |
| 30         | 950        | 950        |
| 30         | 1250       | 2200       |
| 30         | 1250       | 3450       |
| 30         | 1500       | 4950       |
| 30         | 1600       | 6550       |
| 30         | 2850       | 9400       |
+------------+------------+------------+
```

- Example 3: In Hive-compatible mode, use the deptno column to define a window and calculate the sum of the sal column. The order by clause is specified. This function returns the cumulative sum of the values from the first row to the row that has the same sal value as the current row in the current window. The sum values for the rows that have the same sal value are the same. The current window includes the rows that have the same deptno value. Sample statements:

```
-- Enable the Hive-compatible mode.
set odps.sql.hive.compatible=true;
-- Execute the following statement:
select deptno, sal, sum(sal) over (partition by deptno order by sal) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| deptno     | sal        | _c2        |
+------------+------------+------------+
| 10         | 1300       | 2600       |   -- This row is the first row of this window.
The sum for the first row is the cumulative sum of the values in the first and second r
ows because the two rows have the same sal value.
| 10         | 1300       | 2600       |   -- The return value is the cumulative sum of
the values in the first and second rows.
| 10         | 2450       | 7500       |   -- The sum for the third row is the cumulati
ve sum of the values from the first row to the fourth row because the third and fourth
rows have the same sal value.
| 10         | 2450       | 7500       |   -- The return value is the cumulative sum of
the values from the first row to the fourth row.
| 10         | 5000       | 17500      |
| 10         | 5000       | 17500      |
| 20         | 800        | 800        |
| 20         | 1100       | 1900       |
| 20         | 2975       | 4875       |
| 20         | 3000       | 10875      |
| 20         | 3000       | 10875      |
| 30         | 950        | 950        |
| 30         | 1250       | 3450       |
| 30         | 1250       | 3450       |
| 30         | 1500       | 4950       |
| 30         | 1600       | 6550       |
| 30         | 2850       | 9400       |
+------------+------------+------------+
```

3.9.6. Aggregate functions

Aggregate functions group multiple input records to form a single output record. You can use an aggregate function with a `GROUP BY` clause of MaxCompute SQL. This topic describes the syntax, parameters, and examples of aggregate functions that are supported by MaxCompute SQL. It guides you through data development by using aggregate functions.

The following table describes the aggregate functions that are supported by MaxCompute SQL.

Function	Description
AVG	Returns the average value of a column.

Function	Description
COUNT	Returns the number of records that match the specified criteria.
COUNT_IF	Returns the number of records whose expr value is True.
MAX	Returns the maximum value of a column.
MIN	Returns the minimum value of a column.
MEDIAN	Returns the median value of a column.
STDDEV	Returns the population standard deviation of all the input values.
STDDEV_SAMP	Returns the sample standard deviation of all the input values.
SUM	Returns the sum of a column.
WM_CONCAT	Concatenates strings with a specified delimiter.
ANY_VALUE	Returns a non-deterministic value from a specified column.
APPROX_DISTINCT	Returns the approximate number of distinct input values in a specified column.
ARG_MAX	Returns the column value of the row that corresponds to the maximum value of a specified column.
ARG_MIN	Returns the column value of the row that corresponds to the minimum value of a specified column.
COLLECT_LIST	Aggregates values from a specified column into an array.
COLLECT_SET	Aggregates only distinct values from a specified column into an array.
NUMERIC_HISTOGRAM	Returns the approximate histogram of a specified column.
PERCENTILE_APPROX	Returns approximate percentiles. This function applies to scenarios in which a large amount of data is calculated.

Usage notes

MaxCompute V2.0 provides additional date functions. If the functions that you use involve new data types, you must run one of the following SET commands to enable the MaxCompute V2.0 data type edition. The new data types include TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY.

- Session level: To use the MaxCompute V2.0 data type edition, you must add `set odps.sql.type.system.odps2=true;` before the SQL statement that you want to execute, and commit and execute them together.
- Project level: The project owner can run the following command to enable the MaxCompute V2.0 data type edition for the project based on the project requirements. The configuration takes effect after 10 to 15 minutes.

```
setproject odps.sql.type.system.odps2=true;
```

For more information about `setproject`, see Project operations. For more information about the precautions that you must take when you enable the MaxCompute V2.0 data type edition at the project level, see Data type editions.

If you use an SQL statement that includes multiple aggregate functions and the project resources are insufficient, memory overflow may occur. We recommend that you optimize the SQL statement or purchase computing resources based on your business requirements.

Syntax

Syntax of an aggregate function:

```
<aggregate_name>(<expression>[,..]) [within group (order by <col1>[,<col2>…])] [filter (where <where_condition>)]
```

- `<aggregate_name>(<expression>[,...])`: a built-in aggregate function or a user-defined aggregate function (UDAF). The format of an aggregate function is based on its syntax.
- `within group (order by <col1>[,<col2>…])`: If the syntax of an aggregate function includes this expression, the system automatically sorts the input data of `<col1>[,<col2>…]` in ascending order. To sort the input data in descending order, use the expression `within group (order by <col1>[,<col2>…] [desc])`.

  Before you use this expression, take note of the following points:

  - You can use this expression only for WM_CONCAT, COLLECT_LIST, COLLECT_SET, and UDAFs.
  - If multiple aggregate functions in a SELECT statement include the expression `within group (order by <col1>[,<col2>…])`, `order by <col1>[,<col2>…]` must be the same for these functions.
  - If the parameters of an aggregate function include the DISTINCT keyword, columns with distinct values must be specified in the expression `order by <col1>[,<col2>…]`.

  Examples:

```
-- Example 1: Sort the input data in ascending order and return the output data.
select
  x,
  wm_concat(',', y) within group (order by y)
from values('k', 1),('k', 3),('k', 2) as t(x, y)
group by x;
-- The following result is returned:
+------------+------------+
| x          | _c1        |
+------------+------------+
| k          | 1,2,3      |
+------------+------------+
-- Example 2: Sort the input data in descending order and return the output data.
select
  x,
  wm_concat(',', y) within group (order by y desc)
from values('k', 1),('k', 3),('k', 2) as t(x, y)
group by x;
-- The following result is returned:
+------------+------------+
| x          | _c1        |
+------------+------------+
| k          | 3,2,1      |
+------------+------------+
```

- `[filter (where <where_condition>)]`: If an aggregate function includes this expression, the aggregate function processes only the data that meets the condition specified by `<where_condition>`. For more information about `<where_condition>`, see WHERE clause (where_condition).

  Before you use this expression, take note of the following points:

  - Only built-in aggregate functions support this expression. UDAFs do not support this expression.
  - `count(*)` does not support this expression. To add filter conditions to `count(*)`, you can use the COUNT_IF function.
  - The COUNT_IF function does not support this expression.

  Examples:

```
-- Example 1: Filter and aggregate data.
select
  sum(x),
  sum(x) filter (where y > 1),
  sum(x) filter (where y > 2)
  from values(null, 1),(1, 2),(2, 3),(3, null) as t(x, y);
-- The following result is returned:
+------------+------------+------------+
| _c0        | _c1        | _c2        |
+------------+------------+------------+
| 6          | 3          | 2          |
+------------+------------+------------+
-- Example 2: Use multiple aggregate functions to filter and aggregate data.
select
  count_if(x > 2),
  sum(x) filter (where y > 1),
  sum(x) filter (where y > 2)
  from values(null, 1),(1, 2),(2, 3),(3, null) as t(x, y);
-- The following result is returned:
+------------+------------+------------+
| _c0        | _c1        | _c2        |
+------------+------------+------------+
| 1          | 3          | 2          |
+------------+------------+------------+
```

Filter expressions

- Limits
  - Only built-in aggregate functions of MaxCompute support filter expressions. UDAFs do not support filter expressions.
  - `COUNT(*)` cannot be used with filter expressions. Use the COUNT_IF function instead if you want to specify a filter expression.
- Syntax

```
<aggregate_name>(<expression>[,...]) [filter (where <where_condition>)]
```

- Description

  All aggregate functions support filter expressions. If you specify a filter condition, only the row data that meets the filter condition can be passed to the related aggregate function for data processing.
- Parameters
  - aggregate_name: required. The name of the aggregate function. Select an aggregate function that is described in this topic based on your business requirements.
  - expression: required. The parameters of the aggregate function that you select. Specify this parameter based on the description of the aggregate function that you select.
  - where_condition: optional. The filter condition. For more information about where_condition, see WHERE clause (where_condition).
- Return value

  For more information, see the description of the return value for each aggregate function.

- Example:

```
select sum(sal) filter (where deptno=10), sum(sal) filter (where deptno=20), sum(sal) fil
ter (where deptno=30) from emp;
```

The following result is returned:

```
+------------+------------+------------+
| _c0        | _c1        | _c2        |
+------------+------------+------------+
| 17500      | 10875      | 9400       |
+------------+------------+------------+
```

AVG

- Syntax

```
double avg(double <colname>)
decimal avg(decimal <colname>)
```

- Description

  Returns the average value of a column.

- Parameters

  colname: required. The name of a column, which is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation.

- Return value

  A value of the DECIMAL or DOUBLE type is returned. The return value varies based on the following rules:

  - If the value of colname is of the DECIMAL type, a value of the DECIMAL type is returned.
  - If the value of colname is of another valid data type, a value of the DOUBLE type is returned.
  - If the value of colname is of the BOOLEAN type, the value is not used for calculation.
  - If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Calculate the average salary (sal) values of all employees. Sample statement:

```
select avg(sal) from emp;
```

The following result is returned:

```
+-------------+
| _c0         |
+-------------+
| 2222.0588235294117 |
+-------------+
```

- Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the average salary (sal) values of employees in each department. Sample statement:

```
select deptno, avg(sal) from emp group by deptno;
```

The following result is returned:

```
+------------+------------+
| deptno     | _c1        |
+------------+------------+
| 10         | 2916.6666666666665 |
| 20         | 2175.0     |
| 30         | 1566.6666666666667 |
+------------+------------+
```

COUNT

- Syntax

```
bigint count([distinct|all] <colname>)
```

- Description

  Returns the number of records that match the specified criteria.

- Parameters
  - distinctall: optional. This parameter specifies whether to remove duplicates during the counting. The default value is all, which indicates that all records are counted. If this parameter is set to distinct, only records with distinct values are counted.
  - colname: required. The name of a column, which can be of any data type. The value of colname can be an asterisk (`*`). `count(*)` indicates that the number of all rows is returned.

- Return value

  A value of the BIGINT type is returned. If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Calculate the total number of employees in all departments. Sample statement:

```
select count(*) from emp;
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| 17         |
+------------+
```

- Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the number of employees in each department. Sample statement:

  ```
  select deptno, count(*) from emp group by deptno;
  ```

  The following result is returned:

  ```
  +------------+------------+
  | deptno     | _c1        |
  +------------+------------+
  | 10         | 6          |
  | 20         | 5          |
  | 30         | 6          |
  +------------+------------+
  ```

- Example 3: Remove duplicates when you calculate the number of departments. Sample statement:

  ```
  select count(distinct deptno) from emp;
  ```

  The following result is returned:

  ```
  +------------+
  | _c0        |
  +------------+
  | 3          |
  +------------+
  ```

COUNT_IF

- Syntax

  ```
  bigint count_if(boolean <expr>)
  ```

- Description

  Returns the number of records whose expr value is True.

- Parameters

  expr: required. A BOOLEAN expression.

- Return value

  A value of the BIGINT type is returned. If the value of the expr parameter is False or the value of a specified column in expr is null, the row that contains this value is not used for calculation.

- Sample statement:

  ```
  select count_if(sal > 1000), count_if(sal <=1000) from emp;
  ```

  The following result is returned:

  ```
  +------------+------------+
  | _c0        | _c1        |
  +------------+------------+
  | 15         | 2          |
  +------------+------------+
  ```

MAX

- Syntax

```
max(<colname>)
```

- Description

  Returns the maximum value of a column.

- Parameters

  colname: required. The name of a column, which can be of any data type other than BOOLEAN.

- Return value

  The type of the return value is the same as the type of the colname parameter. The return value varies based on the following rules:

  - If the value of colname is null, the row that contains this value is not used for calculation.
  - If the value of colname is of the BOOLEAN type, the value is not used for calculation.

- Examples
  - Example 1: Calculate the highest salary (sal) of all employees. Sample statement:

    ```
    select max(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 5000       |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the highest salary of employees in each department. Sample statement:

    ```
    select deptno, max(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 5000       |
    | 20         | 3000       |
    | 30         | 2850       |
    +------------+------------+
    ```

MIN

- Syntax

```
min(<colname>)
```

- Description

  Returns the minimum value of a column.

- Parameters

  colname: required. The name of a column, which can be of any data type other than BOOLEAN.

- Return value

  The type of the return value is the same as the type of the colname parameter. The return value varies based on the following rules:

  - If the value of colname is null, the row that contains this value is not used for calculation.
  - If the value of colname is of the BOOLEAN type, the value is not used for calculation.

- Examples
  - Example 1: Calculate the lowest salary (sal) of all employees. Sample statement:

    ```
    select min(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 800        |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the lowest salary of employees in each department. Sample statement:

    ```
    select deptno, min(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 1300       |
    | 20         | 800        |
    | 30         | 950        |
    +------------+------------+
    ```

MEDIAN

- Syntax

  ```
  double median(double <colname>)
  decimal median(decimal <colname>)
  ```

- Description

  Returns the median value of a column.

- Parameters

  colname: required. The name of a column, which is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, the value is implicitly converted into the DOUBLE type before calculation.

- Return value

A value of the DECIMAL or DOUBLE type is returned. The return value varies based on the following rules:

- If the value of colname is of the DECIMAL type, a value of the DECIMAL type is returned.
- If the value of colname is of another valid data type, a value of the DOUBLE type is returned.
- If the value of colname is of the BOOLEAN type, the value is not used for calculation.
- If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Calculate the median salary (sal) values of all employees. Sample statement:

    ```
    select median(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 1600.0     |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the median salary values of employees in each department. Sample statement:

    ```
    select deptno, median(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 2450.0     |
    | 20         | 2975.0     |
    | 30         | 1375.0     |
    +------------+------------+
    ```

STDDEV

- Syntax

  ```
  double stddev(double <colname>)
  decimal stddev(decimal <colname>)
  ```

- Description

  Returns the population standard deviation of all the input values.

- Parameters

  colname: required. The name of a column, which is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, the value is implicitly converted into the DOUBLE type before calculation.

- Return value

  A value of the DECIMAL or DOUBLE type is returned. The return value varies based on the following rules:

- If the value of colname is of the DECIMAL type, a value of the DECIMAL type is returned.
- If the value of colname is of another valid data type, a value of the DOUBLE type is returned.
- If the value of colname is of the BOOLEAN type, the value is not used for calculation.
- If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Calculate the population standard deviation of salary (sal) values of all employees. Sample statement:

    ```
    select stddev(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 1262.7549932628976 |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the population standard deviation of salary values of employees in each department. Sample statement:

    ```
    select deptno, stddev(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 1546.1421524412158 |
    | 20         | 1004.7387720198718 |
    | 30         | 610.1001739241043 |
    +------------+------------+
    ```

STDDEV_SAMP

- Syntax

  ```
  double stddev_samp(double <colname>)
  decimal stddev_samp(decimal <colname>)
  ```

- Description

  Returns the sample standard deviation of all the input values.

- Parameters

  colname: required. The name of a column, which is of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, the value is implicitly converted into the DOUBLE type before calculation.

- Return value

  A value of the DECIMAL or DOUBLE type is returned. The return value varies based on the following rules:

- If the value of colname is of the DECIMAL type, a value of the DECIMAL type is returned.
- If the value of colname is of another valid data type, a value of the DOUBLE type is returned.
- If the value of colname is of the BOOLEAN type, the value is not used for calculation.
- If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Calculate the sample standard deviation of salary (sal) values of all employees. Sample statement:

    ```
    select stddev_samp(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 1301.6180541247609 |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the sample standard deviation of salary values of employees in each department. Sample statement:

    ```
    select deptno, stddev_samp(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 1693.7138680032901 |
    | 20         | 1123.3320969330487 |
    | 30         | 668.3312551921141 |
    +------------+------------+
    ```

SUM

- Syntax

  ```
  sum(<colname>)
  ```

- Description

  Returns the sum of a column.

- Parameters

  colname: required. The name of a column, which is of the DOUBLE, DECIMAL, or BIGINT type. If the input value is of the STRING type, the value is implicitly converted into the DOUBLE type before calculation.

- Return value

  A value of the DECIMAL or DOUBLE type is returned. The return value varies based on the following rules:

  - If the value of colname is of the DECIMAL type, a value of the DECIMAL type is returned.

- If the value of colname is of another valid data type, a value of the DOUBLE type is returned.
- If the value of colname is of the BOOLEAN type, the value is not used for calculation.
- If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Calculate the sum of salary (sal) values of all employees. Sample statement:

    ```
    select sum(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 37775      |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the sum of salary values of employees in each department. Sample statement:

    ```
    select deptno, sum(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 17500      |
    | 20         | 10875      |
    | 30         | 9400       |
    +------------+------------+
    ```

WM_CONCAT

- Syntax

  ```
  string wm_concat(string <separator>, string <colname>)
  ```

- Description

  Concatenates values in colname with a delimiter that is specified by separator.

- Parameters
  - separator: required. The delimiter, which is a constant of the STRING type.
  - colname: required. The name of a column, which is of the STRING type. If the input value is of the BIGINT, DOUBLE, or DATETIME type, the value is implicitly converted into the STRING type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of separator is not a constant of the STRING type, an error is returned.
  - If the value of colname is not of the STRING, BIGINT, DOUBLE, or DATETIME type, an error is returned.
  - If the value of colname is null, the row that contains this value is not used for calculation.

> **Note** If `table_name` in the `select wm_concat(',', name) from table_name;` statement is an empty set, null is returned.

- Examples
  - Example 1: Concatenate the names (ename) of all employees. Sample statement:

    ```
    select wm_concat(',', ename) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | SMITH,ALLEN,WARD,JONES,MARTIN,BLAKE,CLARK,SCOTT,KING,TURNER,ADAMS,JAMES,FORD,MILLER,J
    ACCKA,WELAN,TEBAGE |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and concatenate the names (ename) of employees in each department. Sample statement:

    ```
    select deptno, wm_concat(',', ename) from emp group by deptno order by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | CLARK,KING,MILLER,JACCKA,WELAN,TEBAGE |
    | 20         | SMITH,JONES,SCOTT,ADAMS,FORD |
    | 30         | ALLEN,WARD,MARTIN,BLAKE,TURNER,JAMES |
    +------------+------------+
    ```

  - Example 3: Use this function with `GROUP BY` to group all employees by department (deptno) and concatenate the salary values of the employees in each department after duplicates are removed. Sample statement:

    ```
    select deptno, wm_concat(distinct ',', sal) from emp group by deptno order by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 1300,2450,5000 |
    | 20         | 1100,2975,3000,800 |
    | 30         | 1250,1500,1600,2850,950 |
    +------------+------------+
    ```

ANY_VALUE

- Syntax

  ```
  any_value(<colname>)
  ```

- Description

Returns a non-deterministic value from a specified column. This function is an additional function of MaxCompute V2.0.

- Parameters

  colname: required. The name of a column, which can be of any data type.

- Return value

  The type of the return value is the same as the type of the colname parameter. If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Select one of the employees. Sample statement:

    ```
    select any_value(ename) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | SMITH      |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and select one employee from each group. Sample statement:

    ```
    select deptno, any_value(ename) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | CLARK      |
    | 20         | SMITH      |
    | 30         | ALLEN      |
    +------------+------------+
    ```

ARG_MAX

- Syntax

  ```
  arg_max(<valueToMaximize>, <valueToReturn>)
  ```

- Description

  Finds the row in which the value of valueToMaximize is included and returns the value of valueToReturn in the row. This function is an additional function of MaxCompute V2.0.

- Parameters
  - valueToMaximize: required. The name of a column, which can be of any data type.
  - valueToReturn: required. The name of a column, which can be of any data type.

- Return value

The type of the return value is the same as the type of the valueToReturn parameter. If the value of valueToMaximize is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Return the name of the employee with the highest salary. Sample statement:

    ```
    select arg_max(sal, ename) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | KING       |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and return the name of the employee with the highest salary in each department. Sample statement:

    ```
    select deptno, arg_max(sal, ename) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | KING       |
    | 20         | SCOTT      |
    | 30         | BLAKE      |
    +------------+------------+
    ```

ARG_MIN

- Syntax

  ```
  arg_min(<valueToMinimize>, <valueToReturn>)
  ```

- Description

  Finds the row in which the value of valueToMinimize is included and returns the value of valueToReturn in the row. This function is an additional function of MaxCompute V2.0.

- Parameters
  - valueToMinimize: required. The name of a column, which can be of any data type.
  - valueToReturn: required. The name of a column, which can be of any data type.

- Return value

  The type of the return value is the same as the type of the valueToReturn parameter. If the value of valueToMinimize is null, the row that contains this value is not used for calculation.

- Examples

- Example 1: Return the name of the employee with the lowest salary. Sample statement:

```
select arg_min(sal, ename) from emp;
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| SMITH      |
+------------+
```

- Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and return the name of the employee with the lowest salary in each department. Sample statement:

```
select deptno, arg_min(sal, ename) from emp group by deptno;
```

The following result is returned:

```
+------------+------------+
| deptno     | _c1        |
+------------+------------+
| 10         | MILLER     |
| 20         | SMITH      |
| 30         | JAMES      |
+------------+------------+
```

APPROX_DISTINCT

- Syntax

```
approx_distinct(<colname>)
```

- Description

  Returns the approximate number of distinct input values in a specified column. This function is an additional function of MaxCompute V2.0.

- Parameters

  colname: required. The name of the column from which duplicates need to be removed.

- Return value

  A value of the BIGINT type is returned. This function produces a standard error of 5%. If the value of colname is null, the row that contains this value is not used for calculation.

- Examples

- Example 1: Calculate the approximate number of distinct values in the sal column. Sample statement:

```
select approx_distinct(sal) from emp;
```

The following result is returned:

```
+-------------------+
| numdistinctvalues |
+-------------------+
| 12                |
+-------------------+
```

- Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the approximate number of distinct values in the sal column. Sample statement:

```
select deptno, approx_distinct(sal) from emp group by deptno;
```

The following result is returned:

```
+------------+-------------------+
| deptno     | numdistinctvalues |
+------------+-------------------+
| 10         | 3                 |
| 20         | 4                 |
| 30         | 5                 |
+------------+-------------------+
```

COLLECT_LIST

- Syntax

```
array collect_list(<colname>)
```

- Description

  Aggregates the values that are specified by colname into an array. This function is an additional function of MaxCompute V2.0.

- Parameters

  colname: required. The name of a column, which can be of any data type.

- Return value

  A value of the ARRAY type is returned. If the value of colname is null, the row that contains this value is not used for calculation.

- Examples

- Example 1: Aggregate the salary (sal) values of all employees into an array. Sample statement:

```
select collect_list(sal) from emp;
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| [800,1600,1250,2975,1250,2850,2450,3000,5000,1500,1100,950,3000,1300,5000,2450,1300]
|
+------------+
```

- Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and aggregate the salary values of the employees in each department. Sample statement:

```
select deptno, collect_list(sal) from emp group by deptno;
```

The following result is returned:

```
+------------+------------+
| deptno     | _c1        |
+------------+------------+
| 10         | [2450,5000,1300,5000,2450,1300] |
| 20         | [800,2975,3000,1100,3000] |
| 30         | [1600,1250,1250,2850,1500,950] |
+------------+------------+
```

- Example 3: Use this function with `GROUP BY` to group all employees by department (deptno) and aggregate the salary values of the employees in each department after duplicates are removed. Sample statement:

```
select deptno, collect_list(distinct sal) from emp group by deptno;
```

The following result is returned:

```
+------------+------------+
| deptno     | _c1        |
+------------+------------+
| 10         | [1300,2450,5000] |
| 20         | [800,1100,2975,3000] |
| 30         | [950,1250,1500,1600,2850] |
+------------+------------+
```

COLLECT_SET

- Syntax

```
array collect_set(<colname>)
```

- Description

  Aggregates the values that are specified by colname into an array with only distinct values. This function is an additional function of MaxCompute V2.0.

- Parameters

colname: required. The name of a column, which can be of any data type.

- Return value

  A value of the ARRAY type is returned. If the value of colname is null, the row that contains this value is not used for calculation.

- Examples
  - Example 1: Aggregate the salary (sal) values of all employees into an array with only distinct values. Sample statement:

    ```
    select collect_set(sal) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | [800,950,1100,1250,1300,1500,1600,2450,2850,2975,3000,5000] |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and aggregate the salary values of the employees in each department into an array with only distinct values. Sample statement:

    ```
    select deptno, collect_set(sal) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | [1300,2450,5000] |
    | 20         | [800,1100,2975,3000] |
    | 30         | [950,1250,1500,1600,2850] |
    +------------+------------+
    ```

NUMERIC_HISTOGRAM

- Syntax

  ```
  map<double key, double value> numeric_histogram(bigint <buckets>, double <colname>)
  ```

- Description

  Returns the approximate histogram of a specified column. This function is an additional function of MaxCompute V2.0.

- Parameters
  - buckets: required. A value of the BIGINT type. This parameter specifies the maximum number of buckets in the column whose approximate histogram is returned.
  - colname: required. A value of the DOUBLE type. This parameter specifies the columns whose approximate histograms need to be calculated.

- Return value

A value of the `map<double key, double value>` type is returned. In the return value, key indicates the X-axis of the approximate histogram, and value indicates the approximate height of the Y-axis of the approximate histogram. The return value varies based on the following rules:

- If the value of buckets is null, null is returned.
- If the value of colname is null, the row that contains this value is not used for calculation.

- Example

  Return the approximate histogram of the sal column. Sample statement:

```
select numeric_histogram(5, sal) from emp;
```

  The following result is returned:

```
+------------+
| _c0        |
+------------+
| {"1328.5714285714287":7.0,"2450.0":2.0,"5000.0":2.0,"875.0":2.0,"2956.25":4.0} |
+------------+
```

PERCENTILE_APPROX

- Syntax

```
double percentile_approx (double <colname>, <p> [, <B>]))
-- Returns multiple approximate percentiles as an array.
array<double> percentile_approx (double <colname>, array(<p1> [, <p2>...]) [, <B>])
```

- Description

  Returns approximate percentiles. This function applies to scenarios in which a large amount of data is calculated. This function sorts the values in a specified column in ascending order and then returns the approximate percentile of the column at the p th percentage. The value of `percentile_approx` starts from 1. For example, if a column contains values of 100, 200, and 300, the sequence numbers of these values are 1, 2, and 3. If you calculate the 0.6th percentile of the column, the `PERCENTILE_APPROX` function returns 180, which is calculated by using the following formula: `100 + (200 - 100) × 0.8`. This value corresponds to 1.8, which is calculated by using the formula 3 × 0.6 and is between sequence numbers 1 and 2. This function is an additional function of MaxCompute V2.0.

  > **Note** `PERCENTILE_APPROX` differs from `PERCENTILE` in the following aspects:
  >
  > - `PERCENTILE_APPROX` is used to calculate the approximate percentile, and `PERCENTILE` is used to calculate the exact percentile. If the amount of data is large, `PERCENTILE` may fail to run due to memory limits, but `PERCENTILE_APPROX` does not have this issue.
  > - The implementation of `PERCENTILE_APPROX` is consistent with that of `PERCENTILE_APPROX` in Hive, but the calculation algorithm of PERCENTILE_APPROX is different from that of `percentile`. If the amount of data is small, the execution result of PERCENTILE_APPROX is different from that of `PERCENTILE`.

- Parameters
  - colname: required. The name of a column, which is of the DOUBLE type.
  - p: required. The approximate percentile. Valid values: `[0.0, 1.0]`.

- B: the accuracy of the return value. A higher accuracy indicates a more accurate value. If you do not specify this parameter, 10000 is used.

- Return value

  A value of the DOUBLE or ARRAY type is returned. The return value varies based on the following rules:

  - If the value of colname is null, the row that contains this value is not used for calculation.
  - If the value of p or B is null, an error is returned.

- Examples
  - Example 1: Calculate the 0.3rd percentile in the sal column. Sample statement:

    ```
    select percentile_approx(sal, 0.3) from emp;
    ```

    The following result is returned:

    ```
    +------------+
    | _c0        |
    +------------+
    | 1252.5     |
    +------------+
    ```

  - Example 2: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the 0.3rd percentile of employees in each department in the sal column. Sample statement:

    ```
    select deptno, percentile_approx(sal, 0.3) from emp group by deptno;
    ```

    The following result is returned:

    ```
    +------------+------------+
    | deptno     | _c1        |
    +------------+------------+
    | 10         | 1300.0     |
    | 20         | 950.0      |
    | 30         | 1070.0     |
    +------------+------------+
    ```

- Example 3: Use this function with `GROUP BY` to group all employees by department (deptno) and calculate the 0.3rd, 0.5th, and 0.8th percentiles of employees in each department in the sal column. Sample statement:

```
set odps.sql.type.system.odps2=true;
select deptno, percentile_approx(sal, array(0.3, 0.5, 0.8), 1000) from emp group by dep
tno;
```

The following result is returned:

```
+------------+------------+
| deptno     | _c1        |
+------------+------------+
| 10         | [1300.0,1875.0,3470.000000000001] |
| 20         | [950.0,2037.5,2987.5] |
| 30         | [1070.0,1250.0,1580.0] |
+------------+------------+
```

3.9.7. String functions

You can use string functions to process specified strings in MaxCompute SQL. This topic describes the syntax and parameters of string functions that are supported by MaxCompute SQL, and provides examples on how to use string functions. This topic guides you through data development by using string functions.

The following table describes the string functions that are supported by MaxCompute SQL.

Function	Description
ASCII	Returns the ASCII code of the first character in a specified string.
CHAR_MATCHCOUNT	Calculates the number of characters of String A that appear in String B.
CHR	Converts an ASCII code into a character.
CONCAT	Concatenates all the specified strings and returns the final string.
ENCODE	Encodes a string in the specified encoding format.
FIND_IN_SET	Returns the position of the specified string among multiple strings that are separated by commas (,).
FORMAT_NUMBER	Converts a number into a string in the specified format.
FROM_JSON	Returns data of the ARRAY, MAP, or STRUCT type based on a given JSON string and a given output format.
GET_JSON_OBJECT	Extracts a single string from a standard JSON string based on a specified method.
INSTR	Returns the position of String A in String B.
IS_ENCODING	Determines whether a string can be converted from one character set to another character set.

Function	Description
KEYVALUE	Splits a string into key-value pairs, separates the key-value pairs, and then returns the value that corresponds to a specified key.
LENGTH	Returns the length of a string.
LENGTHB	Calculates the length of a string in bytes.
LOCATE	Returns the position of a specified string in another string.
LTRIM	Removes the characters from the left side of a string.
MD5	Returns the MD5 value of a string.
PARSE_URL	Parses a URL and returns the specified part of the URL.
PARSE_URL_TUPLE	Parses a URL and returns multiple parts of the URL.
REGEXP_COUNT	Returns the number of substrings that match a specified pattern from a specified position.
REGEXP_EXTRACT	Splits a string into groups based on a specified pattern and returns the string in a specified group.
REGEXP_INSTR	Returns the start or end position of a substring that starts at a specified position and matches a specified pattern for a specified number of times.
REGEXP_REPLACE	Uses a string to replace a substring of another string if the substring matches a specified pattern for a specified number of times.
REGEXP_SUBSTR	Returns a substring in a string that matches a specified pattern for a specified number of times from a specified position.
REPEAT	Returns a string that repeats a specified string for a specified number of times.
REVERSE	Returns the characters of a string in reverse order.
RTRIM	Removes the characters from the right side of a string.
SPACE	Generates a space string.
SPLIT_PART	Uses a delimiter to split a string into substrings and returns a substring of the specified part of the string.
SUBSTR	Returns a substring that has a specified length from a specified position of a string. The string is of the STRING type.
SUBSTRING	Returns a substring that has a specified length from a specified position of a string. The string is of the STRING or BINARY type.
TO_CHAR	Converts data of the BOOLEAN, BIGINT, DECIMAL, or DOUBLE type into the STRING type.

Function	Description
TO_JSON	Converts data of a complex data type into a JSON string.
TOLOWER	Converts uppercase letters in a string into lowercase letters.
TOUPPER	Converts lowercase letters in a string into uppercase letters.
TRIM	Removes the characters from both of the left and right sides of a string.
URL_DECODE	Encodes the input string in the `application/x-www-form-urlencoded MIME` format and returns the encoded string.
URL_ENCODE	Converts an input string that is in the `application/x-www-form-urlencoded MIME` format into a standard string.
CONCAT_WS	Concatenates all input strings in an array by using a specified delimiter.
JSON_TUPLE	Extracts strings from a standard JSON string based on a set of input keys.
LPAD	Left pads a string to a specified length.
RPAD	Right pads a string to a specified length.
REPLACE	Replaces a substring in String A that matches String B with another substring.
SOUNDEX	Converts a string into a string of the SOUNDEX type.
SUBSTRING_INDEX	Truncates a string from a specified delimiter.
TRANSLATE	Replaces the part of String A that appears in String B with String C.

Usage notes

MaxCompute V2.0 provides additional date functions. If the functions that you use involve new data types, you must run one of the following SET commands to enable the MaxCompute V2.0 data type edition. The new data types include TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY.

- Session level: To use the MaxCompute V2.0 data type edition, you must add `set odps.sql.type.system.odps2=true;` before the SQL statement that you want to execute, and commit and execute them together.
- Project level: The project owner can run the following command to enable the MaxCompute V2.0 data type edition for the project based on the project requirements. The configuration takes effect after 10 to 15 minutes.

  ```
  setproject odps.sql.type.system.odps2=true;
  ```

  For more information about `setproject`, see Project operations. For more information about the precautions that you must take when you enable the MaxCompute V2.0 data type edition at the project level, see Data type editions.

ASCII

- Syntax

```
bigint ascii(string <str>)
```

- Description

  Returns the ASCII code of the first character in a string specified by str.

- Parameters

  str: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str is null, null is returned.

- Examples
  - Example 1: Return the ASCII code of the first character in the string `abcde`. Sample statement:

    ```
    -- The return value is 97.
    select ascii('abcde');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select ascii(null);
    ```

CHAR_MATCHCOUNT

- Syntax

  ```
  bigint char_matchcount(string <str1>, string <str2>)
  ```

- Description

  Returns the number of characters that belong to str1 and appear in str2.

- Parameters

  str1 and str2: required. Values of the STRING type. The values must be valid UTF-8 strings. If invalid characters (non-Unicode-encoded characters) are found during the comparison of the two strings, this function returns a negative value.

- Return value

  A value of the BIGINT type is returned. If the value of str1 or str2 is null, null is returned.

- Examples
  - Example 1: Calculate the number of characters that belong to the string `aabc` and appear in the string `abcde`. Sample statement:

    ```
    -- The return value is 4.
    select char_matchcount('aabc','abcde');
    ```

- Example 2: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select char_matchcount(null,'abcde');
```

CHR

- Syntax

```
string chr(bigint <ascii>)
```

- Description

  Converts a specified ASCII code into characters.

- Parameters

  ascii: required. A value of the BIGINT type. This value specifies the ASCII code. Valid values: 0 to 128. If the input value is of the STRING, DOUBLE, or DECIMAL type, the value is implicitly converted into a value of the BIGINT type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of ascii is not within the valid range, an error is returned.
  - If the value of ascii is not of the BIGINT, STRING, DOUBLE, or DECIMAL type, an error is returned.
  - If the value of ascii is null, null is returned.

- Examples
  - Example 1: Convert the ASCII code `100` into characters. Sample statement:

```
-- The return value is d.
select chr(100);
```

  - Example 2: The input parameter is set to null. Sample statement:

```
-- The return value is null.
select chr(null);
```

  - Example 3: The input value is a character of the STRING type. Sample statement:

```
-- The input value is implicitly converted into a value of the BIGINT type before calcu
lation. The return value is d.
select chr('100');
```

CONCAT

- Syntax

```
array<T> concat(array<T> <a>, array<T> <b>[,...])
string concat(string <str1>, string <str2>[,...])
```

- Description
  - Arrays as inputs: Concatenates all elements of multiple arrays and returns a new array.
  - Strings as inputs: Concatenates multiple strings and returns a new string.

- Parameters
  - a and b: required. These parameters specify arrays that `T` in `array<T>` specifies the data type of the elements in the arrays. The elements can be of any data type. The elements in Array a and the elements in Array b must be of the same data type. The null elements are also involved in the operation.
  - str1 and str2: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- Return value
  - A value of the ARRAY type is returned. If one of the input arrays is null, NULL is returned.
  - A value of the STRING type is returned. If no input parameters are configured or an input parameter is set to null, NULL is returned.
- Examples
  - Example 1: Concatenate all elements of `array(10, 20)` and `array(20, -20)`. Sample statement:

    ```
    -- [10, 20, 20, -20] is returned.
    select concat(array(10, 20), array(20, -20));
    ```

  - Example 2: One of the input arrays contains a null element. Sample statement:

    ```
    -- [10, null, 20, -20] is returned.
    select concat(array(10, null), array(20, -20));
    ```

  - Example 3: One of the input arrays is null. Sample statement:

    ```
    -- NULL is returned.
    select concat(array(10, 20), null);
    ```

  - Example 4: Concatenate strings `aabc` and `abcde`. Sample statement:

    ```
    -- aabcabcde is returned.
    select concat('aabc','abcde');
    ```

  - Example 5: The input is empty. Sample statement:

    ```
    -- NULL is returned.
    select concat();
    ```

  - Example 6: One of the input strings is null. Sample statement:

    ```
    -- NULL is returned.
    select concat('aabc', 'abcde', null);
    ```

ENCODE

- Syntax

  ```
  binary encode(string <str>, string <charset>)
  ```

- Description

  Encodes string str in the format specified by charset.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string that you want to encode.
  - charset: required. A value of the STRING type. This parameter specifies the encoding format. Valid values: UTF-8, UTF-16, UTF-16LE, UTF-16BE, ISO-8859-1, and US-ASCII.
- Return value

  A value of the BINARY type is returned. If the value of str or charset is null, null is returned.
- Examples
  - Example 1: Encode string `abc` in the `UTF-8` format. Sample statement:

    ```
    -- The return value is abc.
    select encode("abc", "UTF-8");
    ```
  - Example 2: Encode string `abc` in the `UTF-16BE` format. Sample statement:

    ```
    -- The return value is =00a=00b=00c.
    select encode("abc", "UTF-16BE");
    ```
  - Example 3: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select encode("abc", null);
    ```

FIND_IN_SET

- Syntax

  ```
  bigint find_in_set(string <str1>, string <str2>)
  ```
- Description

  Returns the position of substring str1 in string str2. The substrings in string str2 are separated by commas (,). The first position is 1.
- Parameters
  - str1: required. A value of the STRING type. This parameter specifies the substring that you want to search for.
  - str2: required. A value of the STRING type. This parameter specifies a string in which substrings are separated by commas (,).
- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:
  - If str2 does not contain str1 or str1 contains commas (,), 0 is returned.
  - If the value of str1 or str2 is null, null is returned.
- Examples
  - Example 1: Return the position of substring `ab` in `abc,hello,ab,c`. Sample statement:

    ```
    -- The return value is 3.
    select find_in_set('ab', 'abc,hello,ab,c');
    ```

- Example 2: Return the position of substring `hi` in `abc,hello,ab,c`. Sample statement:

```
-- The return value is 0.
select find_in_set('hi', 'abc,hello,ab,c');
```

- Example 3: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select find_in_set(null, 'abc,hello,ab,c');
```

FORMAT_NUMBER

- Syntax

```
string format_number(float|double|decimal <expr1>, int <expr2>)
```

- Description

  Converts the value of expr1 into a string that meets the format specified by expr2.

- Parameters
  - expr1: required. A value of the FLOAT, DOUBLE, or DECIMAL type. This parameter specifies the expression that you want to format.
  - expr2: required. A value of the INT type. Valid values: 0 to 340. This parameter specifies the number of decimal places that you want to retain. It can also be expressed in a format similar to `#,###,###.##`. The number of decimal places returned varies based on the value of this parameter.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of `expr2` is greater than 0 and is less than or equal to 340, a value that is rounded to the specified number of decimal places is returned.
  - If the value of `expr2` is 0, the return value contains only the integral part and does not contain the decimal point or decimal part.
  - If the value of `expr2` is less than 0 or greater than `340`, an error is returned.
  - If expr1 or expr2 is empty or is set to null, null is returned.

- Examples
  - Example 1: Return a value in the specified format based on a given number. Sample statements:

```
-- The return value is 5.230.
select format_number(5.230134523424545456,3);
-- The return value is 12,332.123.
select format_number(12332.123456, '#,###,###,###.###');
```

  - Example 2: An input parameter is empty or set to null. Sample statements:

```
-- The return value is null.
select format_number('',3);
-- The return value is null.
select format_number(null,3);
```

FROM_JSON

- Syntax

```
from_json(<jsonStr>, <schema>)
```

- Description

  Returns data of the ARRAY, MAP, or STRUCT type based on JSON string jsonStr and output format schema.

- Parameters
  - jsonStr: required. The JSON string that you entered.
  - schema: required. The schema of the JSON string. The value of this parameter must be in the same format as that in the statement for creating a table, such as `array<bigint>`, `map<string, array<string>>`, or `struct<a:int, b:double, `C`:map<string,string>>`.

    > **Note** Keys in a struct are case-sensitive. You can also specify a struct in the format of `a BIGINT, b DOUBLE`, which is equivalent to `STRUCT<a:BIGINT, b:DOUBLE>`.

    The following table describes the mappings between JSON data types and MaxCompute data types.

JSON data type	MaxCompute data type
OBJECT	STRUCT, MAP, and STRING
ARRAY	ARRAY and STRING
NUMBER	TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, and STRING
BOOLEAN	BOOLEAN and STRING
STRING	STRING, CHAR, VARCHAR, BINARY, DATE, and DATETIME
NULL	All types

    > **Note** The JSON string of the OBJECT and ARRAY types are parsed as much as possible. If the data type of the JSON string is not mapped to any MaxCompute data type, the JSON string is omitted. For ease of use, all JSON data types can be converted into the STRING data type supported by MaxCompute. When you convert a JSON string of the NUMBER type to a value of the FLOAT, DOUBLE, or DECIMAL type, the precision of the value cannot be ensured. We recommend you convert the JSON string to a value of the STRING type and then convert the obtained value to a value of the FLOAT, DOUBLE, or DECIMAL type.

- Return value

  A value of the ARRAY, MAP, or STRUCT type is returned.

- Examples

- Example 1: Convert a specific JSON string into a value of a specific data type. Sample statements:

```
-- {"a":1,"b":0.8} is returned.
select from_json('{"a":1, "b":0.8}', 'a int, b double');
-- {"time":"26/08/2015"} is returned.
select from_json('{"time":"26/08/2015"}', 'time string');
-- {"a":1,"b":0.8} is returned.
select from_json('{"a":1, "b":0.8}', 'a int, b double, c string');
-- [1,2,3] is returned.
select from_json('[1, 2, 3, "a"]', 'array<bigint>');
-- {"d":"v","a":"1","b":"[1,2,3]","c":"{}"} is returned.
select from_json('{"a":1,"b":[1,2,3],"c":{},"d":"v"}', 'map<string, string>');
```

- Example 2: Use the `map_keys` and `from_json` functions to obtain all keys in the JSON string. You can also use JSON_KEYS for the same purpose. Sample statement:

```
-- ["a","b"] is returned.
select map_keys(from_json('{"a":1,"b":2}','map<string,string>'));
```

GET_JSON_OBJECT

- Syntax

```
string get_json_object(string <json>, string <path>)
```

- Description

  Extracts a single string from a standard JSON string based on path. The original data is read each time this function is called. Therefore, repeated calls may affect system performance and increase costs. To prevent repeated calls, you can use the `GET_JSON_OBJECT` function with UDTFs. For more information, see Convert JSON log data by using MaxCompute built-in functions and UDTFs.

- Parameters
  - json: required. A value of the STRING type. This parameter specifies a standard JSON object in the format of `{Key:Value, Key:Value,...}`. If the string contains a double quotation mark ("), use two backslashes (\\) to escape the double quotation mark (") before extraction. If the string contains a single quotation mark ('), use a single backslash (\) to escape the single quotation mark (') before extraction.
  - path: required. A value of the STRING type. This parameter specifies the path in the value of the json parameter and starts with `$`. For more information about the path parameter, see LanguageManual UDF. For more information about best practices, see Migrate JSON data from OSS to MaxCompute. Different characters have the following meanings:
    - `$`: indicates the root node.
    - `.` or `['']`: indicates a child node. MaxCompute parses JSON objects by using `.` or `['']`. If a key in a JSON object contains a period (.), ['] can be used.
    - `[]` (`[number]`): indicates an array subscript, which starts from 0.
    - `*`: indicates the wildcard for `[]`, which returns an entire array. An asterisk (`*`) cannot be escaped.

- Return value
  - If the json parameter is empty or invalid, NULL is returned.
  - If the format of json is valid and path exists, the related string is returned.

- To determine how this function returns a value, you can specify `odps.sql.udf.getjsonobj.new` for a session.
  - If you run the `set odps.sql.udf.getjsonobj.new=true;` command, this function retains the original strings when it returns a value.

    We recommend that you use this configuration because it results in more standard function return behavior. This facilitates data processing and improves data processing performance. **If a MaxCompute project has jobs in which this function escapes JSON reserved characters, we recommend that you retain the original escaping operation to prevent errors caused by lack of verification.** The function complies with the following rules when it returns a value:

    - The return value is still a JSON string, which can be parsed as JSON data. You do not need to use the `REPLACE` or `REGEXP_REPLACE` function to replace backslashes (\).
    - Duplicate keys are allowed in a JSON object. If duplicate keys exist, the data can be parsed.

      ```
      -- 1 is returned.
      select get_json_object('{"a":"1","a":"2"}', '$.a');
      ```

    - The encoded strings that correspond to emojis are supported. However, DataWorks does not allow you to enter emojis. DataWorks allows you to enter only the encoded strings that correspond to emojis to MaxCompute by using a tool, such as Data Integration. DataWorks uses the `GET_JSON_OBJECT` function to process the data.

      ```
      -- An emoji is returned.
      select get_json_object('{"a":"<Emoji>"}', '$.a');
      ```

    - The output results are displayed in alphabetical order.

      ```
      -- {"b":"1","a":"2"} is returned.
      select get_json_object('{"b":"1","a":"2"}', '$');
      ```

  - If you run the `set odps.sql.udf.getjsonobj.new=false;` command, this function escapes JSON reserved characters when it returns a value. The function complies with the following rules when it returns a value:
    - JSON reserved characters, such as line feeds (\n) and double quotation marks (") are displayed as `'\n'` and `'\"'`.
    - Each key in a JSON object must be unique. If duplicate keys exist, the data may fail to be parsed.

      ```
      -- NULL is returned.
      select get_json_object('{"a":"1","a":"2"}', '$.a');
      ```

    - The encoded strings that correspond to emojis cannot be parsed.

      ```
      -- NULL is returned.
      select get_json_object('{"a":"<Emoji>"}', '$.a');
      ```

    - The output results are displayed in alphabetical order.

      ```
      -- {"a":"2","b":"1"} is returned.
      select get_json_object('{"b":"1","a":"2"}', '$');
      ```

> **Note** For MaxCompute projects that are created on or after January 21, 2021, the `GET_JSON_OBJECT` function retains the original strings when it returns a value. For MaxCompute projects that are created before January 21, 2021, the `GET_JSON_OBJECT` function escapes JSON reserved characters when it returns a value. The following example helps you determine how the `GET_JSON_OBJECT` function returns a value in a MaxCompute project.
>
> ```
> select get_json_object('{"a":"[\\"1\\"]"}', '$.a');
> -- Return JSON reserved characters by using escape characters.
> [\"1\"]
> -- Return the original strings.
> ["1"]
> ```
>
> You can submit a ticket to request the MaxCompute technical support team to enable the `GET_JSON_OBJECT` function. This way, you do not need to frequently specify flagodps.sql.udf.getjsonobj.new for a session. This function retains the original strings when it returns a value.

- Examples
  - Example 1: Extract information from the JSON object `src_json.json`. Sample statement:

```
-- The JSON string src_json.json contains the following content:
+----+
json
+----+
{"store":
{"fruit":[{"weight":8,"type":"apple"},{"weight":9,"type":"pear"}],
"bicycle":{"price":19.95,"color":"red"}
},
"email":"amy@only_for_json_udf_test.net",
"owner":"amy"
}
-- Extract the information of the owner field and return amy.
select get_json_object(src_json.json, '$.owner') from src_json;
-- Extract the information of the first array in the store.fruit field and return {"wei
ght":8,"type":"apple"}.
select get_json_object(src_json.json, '$.store.fruit[0]') from src_json;
-- Extract the information of the non-existent field and return NULL.
select get_json_object(src_json.json, '$.non_exist_key') from src_json;
```

- Example 2: Extract information from a JSON object of the ARRAY type. Sample statement:

```
-- 2222 is returned.
select get_json_object('{"array":[["aaaa",1111],["bbbb",2222],["cccc",3333]]}','$.array
[1][1]');
-- ["h0","h1","h2"] is returned.
set odps.sql.udf.getjsonobj.new=true;
select get_json_object('{"aaa":"bbb","ccc":{"ddd":"eee","fff":"ggg","hhh":["h0","h1","h
2"]},"iii":"jjj"}','$.ccc.hhh[*]');
-- ["h0","h1","h2"] is returned.
set odps.sql.udf.getjsonobj.new=false;
select get_json_object('{"aaa":"bbb","ccc":{"ddd":"eee","fff":"ggg","hhh":["h0","h1","h
2"]},"iii":"jjj"}','$.ccc.hhh');
-- h1 is returned.
select get_json_object('{"aaa":"bbb","ccc":{"ddd":"eee","fff":"ggg","hhh":["h0","h1","h
2"]},"iii":"jjj"}','$.ccc.hhh[1]');
```

- Example 3: Extract information from a JSON object that contains a period `(.)`. Sample statement:

```
-- Create a table.
create table mf_json (id string, json string);
-- Insert data into the table. The key in the data contains a period (.).
insert into table mf_json (id, json) values ("1", "{
\"China.beijing\":{\"school\":{\"id\":0,\"book\":[{\"title\": \"A\",
\"price\": 8.95},{\"title\": \"B\",\"price\": 10.2}]}}}");
-- Insert data into the table. The key in the data does not contain a period (.).
insert into table mf_json (id, json) values ("2", "{
\"China_beijing\":{\"school\":{\"id\":0,\"book\":[{\"title\": \"A\",
\"price\": 8.95},{\"title\": \"B\",\"price\": 10.2}]}}}");
-- Query the value of id in the JSON object whose key is China.beijing. 0 is returned.
Only [''] can be used to specify the key because the key contains a period (.). This wa
y, MaxCompute can parse the key.
select get_json_object(json, "$['China.beijing'].school['id']") from mf_json where id =
1;
-- -- Query the value of id in the JSON object whose key is China_beijing. 0 is returne
d. You can use one of the following statements:
select get_json_object(json, "$['China_beijing'].school['id']") from mf_json where id =
2;
select get_json_object(json, "$.China_beijing.school['id']") from mf_json where id =2;
```

- Example 4: The json parameter is empty or invalid. Sample statement:

```
-- NULL is returned.
select get_json_object('','$.array[1][1]');
-- NULL is returned.
select get_json_object('"array":["aaaa",1111],"bbbb":["cccc",3333]','$.array[1][1]');
```

- Example 5: Escape a JSON string. Sample statement:

```
set odps.sql.udf.getjsonobj.new=true;
-- "1" is returned.
select get_json_object('{"a":"\\"1\\"","b":"2"}', '$.a');
-- '1' is returned.
select get_json_object('{"a":"\'1\'","b":"2"}', '$.a');
```

INSTR

- Syntax

```
bigint instr(string <str1>, string <str2>[, bigint <start_position>[, bigint <nth_appearance>]])
```

- Description

  Returns the position of substring str2 in string str1.

- Parameters
  - str1: required. A value of the STRING type. This parameter specifies the string that contains the substring you want to search for. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
  - str2: required. A value of the STRING type. This parameter specifies the substring that you want to search for. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
  - start_position: optional. A value of the BIGINT type. If the input value is of another data type, an error is returned. This parameter specifies the character in str1 from which the search starts. The default start position is the first character, marked as 1. If start_position is a negative value, the search starts from the end to the start of the string and the last character is -1.
  - nth_appearance: optional. A value of the BIGINT type, which is greater than 0. This parameter specifies the nth time that substring str2 appears in string str1. If the value of nth_appearance is of another data type or is less than or equal to 0, an error is returned.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:
  - If str2 is not found in str1, 0 is returned.
  - If str2 is an empty string, the matching always succeeds. For example, 1 is returned for `select instr('abc','');`.
  - If the value of str1, str2, start_position, or nth_appearance is null, null is returned.

- Examples
  - Example 1: Return the position of the substring `e` in the string `Tech on the net`. Sample statement:

    ```
    -- The return value is 2.
    select instr('Tech on the net', 'e');
    ```

  - Example 2: Return the position of substring `on` in string `Tech on the net`. Sample statement:

    ```
    -- The return value is 6.
    select instr('Tech on the net', 'on');
    ```

  - Example 3: Return the position of the second occurrence in which the substring `e` appears in the string `Tech on the net` from the third character. Sample statement:

    ```
    -- The return value is 14.
    select instr('Tech on the net', 'e', 3, 2);
    ```

- Example 4: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select instr('Tech on the net', null);
```

IS_ENCODING

- Syntax

```
boolean is_encoding(string <str>, string <from_encoding>, string <to_encoding>)
```

- Description

  Determines whether the input string str can be converted from the character set specified by from_encoding into the character set specified by to_encoding. This function can also be used to determine whether the input string is garbled. In most cases, from_encoding is set to UTF-8 and to_encoding is set to GBK.

- Parameters
  - str: required. A value of the STRING type. An empty string can belong to any character set.
  - from_encoding and to_encoding: required. Values of the STRING type. from_encoding specifies the source character set and to_encoding specifies the destination character set.

- Return value

  A value of the BOOLEAN type is returned. The return value varies based on the following rules:
  - If str can be converted, True is returned. Otherwise, False is returned.
  - If the value of str, from_encoding, or to_encoding is null, null is returned.

KEYVALUE

- Syntax

```
keyvalue(string <str>,[string <split1>,string <split2>,] string <key>)
keyvalue(string <str>,string <key>)
```

- Description

  Splits the string str into key-value pairs by split1, separates the key-value pairs by split2, and then returns the value of the specified key.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string that you want to split.
  - split1 and split2: optional. Values of the STRING type. These parameters specify the strings that are used as delimiters to split the source string. If you do not specify the two parameters, the default value of split1 is a semicolon (`;`) and the default value of split2 is a colon (`:`). If a key-value pair that is obtained after the source string is split by split1 contains multiple delimiters specified by split2, the returned result is undefined.
  - key: required. A value of the STRING type. After the source string is split by split1 and split2 in sequence, the value that corresponds to the key is returned.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of split1 or split2 is null, null is returned.

- If the value of str or key is null or no key matches, null is returned.
- If multiple key-value pairs match, the value that corresponds to the first matched key is returned.

- Examples
  - Example 1: Split the string `0:1\;1:2` into key-value pairs and return the value that corresponds to the key `1`. Sample statement:

    ```
    -- The return value is 2.
    select keyvalue('0:1\;1:2', 1);
    ```

    The split1 and split2 parameters are not specified. The default value of split1 is a semicolon (`;`) and the default value of split2 is a colon (`:`).

    After the source string is split by using split1, the key-value pairs `0:1\,1:2` are returned. After the key-value pairs are split by using split2, the following keys and values are returned:

    ```
    0 1/
    1 2
    ```

    The value 2 that corresponds to key 1 is returned.

  - Example 2: Split the string `\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:0\;market:shoes\;instPayAmount:0\;` into key-value pairs by using a backslash and a semicolon (`\;`), separate the values from keys by using a colon (`:`), and then return the value that corresponds to the key `tf`. Sample statement:

    ```
    -- The return value is 21910.
    select keyvalue("\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:
    0\;market:shoes\;instPayAmount:0\;","\;",":","tf");
    ```

    After the source string `\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:0\;market:shoes\;instPayAmount:0\;` is split by using a backslash and a semicolon (`\;`), the following key-value pairs are generated:

    ```
    decreaseStore:1, xcard:1, isB2C:1, tf:21910, cart:1, shipping:2, pf:0, market:shoes, an
    d instPayAmount:0.
    ```

    After the key-value pairs are separated by using a colon (`:`), the following keys and values are generated:

    ```
    decreaseStore 1
    xcard 1
    isB2C 1
    tf 21910
    cart 1
    shipping 2
    pf 0
    market shoes
    instPayAmount 0
    ```

    The value 21910 that corresponds to the key `tf` is returned.

LENGTH

- Syntax

```
bigint length(string <str>)
```

- Description

  Returns the length of the string str.

- Parameters

  str: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str is null, null is returned.
  - If the value of str is not UTF-8 encoded, -1 is returned.

- Examples
  - Example 1: Return the length of the string `Tech on the net`. Sample statement:

    ```
    -- The return value is 15.
    select length('Tech on the net');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select length(null);
    ```

LENGTHB

- Syntax

  ```
  bigint lengthb(string <str>)
  ```

- Description

  Returns the length of a string specified by str in bytes.

- Parameters

  str: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str is null, null is returned.

- Examples
  - Example 1: Return the length of string `Tech on the net` in bytes. Sample statement:

    ```
    -- The return value is 15.
    select lengthb('Tech on the net');
    ```

- Example 2: The input parameter is set to null. Sample statement:

```
-- The return value is null.
select lengthb(null);
```

LOCATE

- Syntax

```
bigint locate(string <substr>, string <str>[, bigint <start_pos>])
```

- Description

  Returns the position of substring substr in string str. You can use start_pos to specify the position from which the search starts. The value starts from 1.

- Parameters
  - substr: required. A value of the STRING type. This parameter specifies the substring that you want to search for.
  - str: required. A value of the STRING type. This parameter specifies the string in which you want to search for the substring.
  - start_pos: optional. A value of the BIGINT type. This parameter specifies the position from which the search starts.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:
  - If substr cannot be found in str, 0 is returned.
  - If the value of str or substr is null, null is returned.
  - If the value of start_pos is null, 0 is returned.

- Examples
  - Example 1: Return the position of string `ab` in string `abchelloabc`. Sample statement:

```
-- The return value is 1.
select locate('ab', 'abchelloabc');
```

  - Example 2: Return the position of string `hi` in string `abchelloabc`. Sample statement:

```
-- The return value is 0.
select locate('hi', 'abc,hello,ab,c');
```

  - Example 3: The start_pos parameter is set to null. Sample statement:

```
-- The return value is 0.
select locate('ab', 'abhelloabc', null);
```

LTRIM

- Syntax

```
string ltrim(string <str>[, <trimChars>])
string trim(leading [<trimChars>] from <str>)
```

- Description

Removes the characters from the left side of a string that is specified by str.

- If you do not specify trimChars, the spaces on the left side are removed by default.
- If you specify trimChars, the substring that consists of one or more characters specified by trimChars is removed from the left side of the string that is specified by str.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string from the left side of which the characters are removed. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.
  - trimChars: optional. A value of the STRING type. This parameter specifies the characters to be removed.
- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str or trimChars is null, null is returned.
- Examples
  - Example 1: Remove the space from the left side of string `yxTxyomxx`. Sample statement:

    ```
    -- The return value is yxTxyomxx .
    select ltrim(' yxTxyomxx ');
    -- The preceding statement is equivalent to the following statement:
    select trim(leading from ' yxTxyomxx ');
    ```
  - Example 2: Remove the substring that consists of one or more characters in the `xy` collection from the left side of the string `yxTxyomxx`.

    ```
    -- The return value is Txyomxx. If x or y appears on the left side, it is removed.
    select ltrim('yxTxyomxx', 'xy');
    -- The preceding statement is equivalent to the following statement:
    select trim(leading 'xy' from 'yxTxyomxx');
    ```
  - Example 3: The input parameter is set to null. Sample statements:

    ```
    -- The return value is null.
    select ltrim(null);
    select ltrim('yxTxyomxx', null);
    ```

MD5

- Syntax

  ```
  string md5(string <str>)
  ```
- Description

  Returns the MD5 value of a string specified by str.
- Parameters

str: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str is null, null is returned.

- Examples
  - Example 1: Return the MD5 value of string `Tech on the net`. Sample statement:

    ```
    -- The return value is ddc4c4796880633333d77a60fcda9af6.
    select md5('Tech on the net');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select md5(null);
    ```

PARSE_URL

- Syntax

  ```
  string parse_url(string <url>, string <part>[, string <key>])
  ```

- Description

  Parses url and extracts information based on the value specified by part.

- Parameters
  - url: required. A value of the STRING type. This parameter specifies a URL. If the URL is invalid, an error is returned.
  - part: required. A value of the STRING type. Valid values: HOST, PATH, QUERY, REF, PROTOCOL, AUTHORITY, FILE, and USERINFO. The value of this parameter is not case-sensitive.
  - key: optional. If part is set to QUERY, this function returns the value that corresponds to key.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of url, part, or key is null, null is returned.
  - If the value of part is invalid, an error is returned.

- Examples

```
-- The return value is example.com.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'HOST');
-- The return value is /over/there/index.dtb.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'PATH');
-- The return value is animal.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'QUERY', 'type');
-- The return value is nose.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'REF');
-- The return value is file.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'PROTOCOL');
-- The return value is username:password@example.com:8042.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'AUTHORITY');
-- The return value is username:password.
select parse_url('file://username:password@example.com:8042/over/there/index.dtb?type=ani
mal&name=narwhal#nose', 'USERINFO');
```

PARSE_URL_TUPLE

- Syntax

```
string parse_url_tuple(string <url>, string <key1>, string <key2>,...)
```

- Description

  Parses url and extracts the strings that are specified by a group of input keys, such as key1 and key2. The PARSE_URL_TUPLE function is similar to the PARSE_URL function. However, the PARSE_URL_TUPLE function provides better performance because it can extract the strings that correspond to multiple keys at the same time.

- Parameters
  - url: required. A value of the STRING type. This parameter specifies a URL. If the URL is invalid, an error is returned.

- key1 and key2: required. Values of the STRING type. These parameters specify the keys that correspond to the strings you want to extract. Valid values:
  - HOST: indicates the host address, which can be a domain name or an IP address.
  - PATH: indicates the path of network resources in the server.
  - QUERY: indicates the string that you want to query.
  - REF: indicates the URL annotation, which is displayed when you move your pointer over the URL.
  - PROTOCOL: indicates the protocol type.
  - AUTHORITY: indicates the domain name or IP address, port number, and user authentication information (such as username and password) of the server.
  - FILE: indicates the path of network resources in the server and the content that you want to query. FILE consists of PATH and QUERY.
  - USERINFO: indicates user authentication information.
  - QUERY:<KEY>: indicates the value of the specified key in the query string.

  The value of this parameter is not case-sensitive. If you specify a value other than the preceding values, an error is returned.

- Return value

  A value of the STRING type is returned. If the value of url or key is null, an error is returned.

- Examples

  Extract the strings that correspond to keys from `file://username:password@example.com:8042/over/there/index.dtb?type=animal&name=narwhal#nose`. Sample statement:

```
select parse_url_tuple('file://username:password@example.com:8042/over/there/index.dtb?type=animal&name=narwhal#nose', 'HOST', 'PATH', 'QUERY', 'REF', 'PROTOCOL', 'AUTHORITY', 'FILE', 'USERINFO', 'QUERY:type', 'QUERY:name') as (item0, item1, item2, item3, item4, item5, item6, item7, item8, item9);
```

  The following result is returned:

```
+------------+------------+------------+------------+------------+------------+------------+------------+------------+------------+
| item0      | item1      | item2      | item3      | item4      | item5      | item6      | item7      | item8      | item9      |
+------------+------------+------------+------------+------------+------------+------------+------------+------------+------------+
| example.com | /over/there/index.dtb | type=animal&name=narwhal | nose       | file       | username:password@example.com:8042 | /over/there/index.dtb?type=animal&name=narwhal | username:password | animal     | narwhal    |
+------------+------------+------------+------------+------------+------------+------------+------------+------------+------------+
```

REGEXP_COUNT

- Syntax

```
bigint regexp_count(string <source>, string <pattern>[, bigint <start_position>])
```

- Description

Returns the number of substrings that match a specified pattern in the source string from the start position specified by start_position.

- Parameters
  - source: required. A value of the STRING type. This parameter specifies the string that contains the substring you want to search for. If the value is not a string, an error is returned.
  - pattern: required. A constant of the STRING type or a regular expression. This parameter specifies the pattern that a specified substring must match. For more information about regular expressions, see Regular expressions. If pattern is an empty string or is of another data type, an error is returned.
  - start_position: optional. A constant of the BIGINT type. The value of this parameter must be greater than 0. If the value is of another data type or is less than or equal to 0, an error is returned. If you do not specify this parameter, the default value is 1. This value indicates that the search starts from the first character of the source string.
- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:
  - If no substring is matched, 0 is returned.
  - If the value of source, pattern, or start_position is null, null is returned.
- Examples
  - Example 1: Calculate the number of substrings in `abababc` that match a specified pattern from the specified position. Sample statements:

    ```
    -- The return value is 1.
    select regexp_count('abababc', 'a.c');
    -- The return value is 2.
    select regexp_count('abababc', '[[:alpha:]]{2}', 3);
    ```
  - Example 2: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select regexp_count('abababc', null);
    ```
  - Example 3: Calculate the number of colons (`:`) in the JSON string `{"account_id":123456789,"account_name":"allen","location":"hangzhou","bill":100}`. Sample statement:

    ```
    -- The return value is 4.
    select regexp_count('{"account_id":123456789,"account_name":"allen","location":"hangzhou","bill":100}',':');
    ```

REGEXP_EXTRACT

- Syntax

  ```
  string regexp_extract(string <source>, string <pattern>[, bigint <occurrence>])
  ```
- Description

  Splits the source string into groups based on a given pattern and returns the string in the nth group specified by occurrence.
- Parameters

- source: required. A value of the STRING type. This parameter specifies the string that you want to split.
- pattern: required. A constant of the STRING type or a regular expression. This parameter specifies the pattern that a specified substring must match. For more information about regular expressions, see Regular expressions.
- occurrence: optional. A constant of the BIGINT type. The value of this parameter must be greater than or equal to 0.

> **Note** Data is stored in the UTF-8 format. Chinese characters can be represented in hexadecimal. They are encoded in the range of **[\\x{4e00},\\x{9fa5}]**.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If pattern is an empty string or no group is specified in pattern, an error is returned.
  - If the value of occurrence is not of the BIGINT type or is less than 0, an error is returned. If you do not specify this parameter, the default value is 1. This value indicates that the string in the first group is returned. If occurrence is set to 0, all substrings that match pattern are returned.
  - If the value of source, pattern, or occurrence is null, null is returned.

REGEXP_INSTR

- Syntax

```
bigint regexp_instr(string <source>, string <pattern>[,bigint <start_position>[, bigint <
occurrence>[, bigint <return_option>]]])
```

- Description

  Returns the start or end position of the substring that matches pattern at the nth occurrence specified by occurrence, in the source string from the start position specified by start_position.

- Parameters
  - source: required. A value of the STRING type. This parameter specifies the source string.
  - pattern: required. A constant of the STRING type or a regular expression. This parameter specifies the pattern that a specified substring must match. For more information about regular expressions, see Regular expressions. If pattern is an empty string, an error is returned.
  - start_position: optional. A constant of the BIGINT type. This parameter specifies the position from which the search starts. If you do not specify this parameter, the default value 1 is used.
  - occurrence: optional. A constant of the BIGINT type. If you do not specify this parameter, the default value 1 is used. This value indicates the position where a substring matches the pattern in the search for the first time.
  - return_option: optional. A constant of the BIGINT type. This parameter specifies whether the start or end position of the substring that matches a specified pattern is returned. Valid values: 0 and 1. If this parameter is not specified, the default value 0 is used. If this parameter is set to an invalid number or a value of another data type, an error is returned. The value 0 indicates that the start position of the substring that matches a specified pattern is returned. The value 1 indicates that the end position of the substring that matches a specified pattern is returned.

- Return value

A value of the BIGINT type is returned. This value indicates the start or end position of the matched substring in the source string based on the type specified by return_option. The return value varies based on the following rules:

- If pattern is an empty string, an error is returned.
- If the value of start_position or occurrence is not of the BIGINT type or is less than or equal to 0, an error is returned.
- If the value of source, pattern, start_position, occurrence, or return_option is null, null is returned.

- Examples
  - Example 1: Return the start position of the substring that matches `o[[:alpha:]]{1}` at the `second` occurrence in the `i love www.taobao.com` string from the `third` character. Sample statement:

    ```
    -- The return value is 14.
    select regexp_instr('i love www.taobao.com', 'o[[:alpha:]]{1}', 3, 2);
    ```

  - Example 2: Return the end position of the substring that matches `o[[:alpha:]]{1}` at the `second` occurrence in the `i love www.taobao.com` string from the `third` character. Sample statement:

    ```
    -- The return value is 16.
    select regexp_instr('i love www.taobao.com', 'o[[:alpha:]]{1}', 3, 2, 1);
    ```

  - Example 3: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select regexp_instr('i love www.taobao.com', null, 3, 2);
    ```

REGEXP_REPLACE

- Syntax

  ```
  string regexp_replace(string <source>, string <pattern>, string <replace_string>[, bigint <occurrence>])
  ```

- Description

  Uses a string specified by replace_string to replace the substring that matches a given pattern at the nth occurrence specified by occurrence in the source string and returns a result.

- Parameters
  - source: required. A value of the STRING type. This parameter specifies three strings that you want to replace.
  - pattern: required. A constant of the STRING type or a regular expression. This parameter specifies the pattern that a specified substring must match. For more information about regular expressions, see Regular expressions. If pattern is an empty string, an error is returned.
  - replace_string: required. A value of the STRING type. The value is used to replace the string that matches the pattern.

- occurrence: optional. A constant of the BIGINT type, which must be greater than or equal to 0. The value of this parameter indicates that the string that matches the specified pattern at the nth occurrence specified by occurrence is replaced with replace_string. If this parameter is set to 0, all the substrings that match the specified pattern are replaced. If this parameter is of another data type or is less than 0, an error is returned. Default value: 0.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the group that you want to reference does not exist, no substring is replaced.
  - If the value of replace_string is null and a substring matches the given pattern, null is returned.
  - If the value of replace_string is null but no substring matches the given pattern, the original string is returned.
  - If the value of source, pattern, or occurrence is null, null is returned.

- Examples
  - Example 1: Use `(\\1)\\2-\\3` to replace all the substrings that match `([[:digit:]]{3})\\.([[:digit:]]{3})\\.([[:digit:]]{4})` in the `123.456.7890` string. Sample statement:

    ```
    -- The return value is (123)456-7890.
    select regexp_replace('123.456.7890', '([[:digit:]]{3})\\.([[:digit:]]{3})\\.([[:digit:]]{4})',
    '(\\1)\\2-\\3', 0);
    ```

  - Example 2: Replace the substring that matches the specified pattern in the `abcd` string. Sample statements:

    ```
    -- The return value is a b c d.
    select regexp_replace('abcd', '(.)', '\\1 ', 0);
    -- The return value is a bcd.
    select regexp_replace('abcd', '(.)', '\\1 ', 1);
    -- The return value is abcd.
    select regexp_replace('abcd', '(.)', '\\2', 1);
    ```

  - Example 3: The data of the url column in the table url_set is in the `www.simple@xxx.com` format and `xxx` in each row is different. Replace all the content after `www` in the url column. Sample statement:

    ```
    -- The return value is wwwtest.
    select regexp_replace(url,'(www)(.*)','wwwtest',0) from url_set;
    ```

  - Example 4: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select regexp_replace('abcd', '(.)', null, 0);
    ```

- Example 5: The group that you want to reference does not exist. Sample statements:

```
-- Only one group is defined in the pattern and the group that you want to reference do
es not exist.
-- We recommend that you do not use this function in this way. The result of referencin
g a nonexistent group is not defined.
regexp_replace("abcd", "(.*)(.)$", "\\2", 0) = "d"
-- No group is defined in the pattern. Therefore, "\1" references a nonexistent group.
-- We recommend that you do not use this function in this way. The result of referencin
g a nonexistent group is not defined.
regexp_replace("abcd", "a", "\\1", 0) = "bcd"
```

REGEXP_SUBSTR

- Syntax

```
string regexp_substr(string <source>, string <pattern>[, bigint <start_position>[, bigint
<occurrence>]])
```

- Description

  Returns a string that matches a given pattern at the nth occurrence specified by occurrence, in the source string from the start position specified by start_position.

- Parameters
  - source: required. A value of the STRING type. This parameter specifies the string in which the substring you want to search for.
  - pattern: required. A constant of the STRING type or a regular expression. This parameter specifies the pattern that a specified substring must match. For more information about regular expressions, see Regular expressions.
  - start_position: optional. A constant of the BIGINT type. The value of this parameter must be greater than 0. If you do not specify this parameter, the default value is 1, which indicates that the search starts from the first character of the source string.
  - occurrence: optional. A constant of the BIGINT type. The value of this parameter must be greater than 0. If you do not specify this parameter, the default value is 1. This value indicates that the first matched substring is returned.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If pattern is an empty string, an error is returned.
  - If no substring matches the specified pattern, null is returned.
  - If the value of start_position or occurrence is not of the BIGINT type or is less than or equal to 0, an error is returned.
  - If the value of source, pattern, start_position, occurrence, or return_option is null, null is returned.

- Examples

- Example 1: Return the substring in the `I love aliyun very much` string that matches a specified pattern. Sample statements:

```
-- The return value is aliyun.
select regexp_substr('I love aliyun very much', 'a[[:alpha:]]{5}');
-- The return value is have.
select regexp_substr('I have 2 apples and 100 bucks!', '[[:blank:]][[:alnum:]]*', 1, 1)
;
-- The return value is 2.
select regexp_substr('I have 2 apples and 100 bucks!', '[[:blank:]][[:alnum:]]*', 1, 2)
;
```

- Example 2: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select regexp_substr('I love aliyun very much', null);
```

REPEAT

- Syntax

```
string repeat(string <str>, bigint <n>)
```

- Description

  Returns a string that repeats the string specified by str for n times.

- Parameters
  - str: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.
  - n: required. A value of the BIGINT type. The returned string cannot exceed 2 MB in length.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If n is left empty, an error is returned.
  - If the value of str or n is null, null is returned.

- Examples
  - Example 1: Return a string that repeats the string `abc` for `5` times. Sample statement:

```
-- The return value is abcabcabcabcabc.
select repeat('abc', 5);
```

  - Example 2: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select repeat('abc', null);
```

REVERSE

- Syntax

```
string reverse(string <str>)
```

- Description

  Returns the characters of a string in reverse order.

- Parameters

  str: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str is null, null is returned.

- Examples

  - Example 1: Return a string whose characters are in reverse order of the string `I love aliyun very much`. Sample statement:

    ```
    -- The return value is hcum yrev nuyila evol I.
    select reverse('I love aliyun very much');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select reverse(null);
    ```

RTRIM

- Syntax

  ```
  string rtrim(string <str>[, <trimChars>])
  string trim(trailing [<trimChars>] from <str>)
  ```

- Description

  Removes the characters from the right side of a string that is specified by str.

  - If you do not specify trimChars, the spaces on the right side are removed by default.
  - If you specify trimChars, the substring that consists of one or more characters specified by trimChars is removed from the right side of the string that is specified by str.

- Parameters

  - str: required. A value of the STRING type. This parameter specifies the string from the right side of which the characters are removed. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.
  - trimChars: optional. A value of the STRING type. This parameter specifies the characters to be removed.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str or trimChars is null, null is returned.

- Examples
  - Example 1: Remove the space from the right side of string `yxTxyomxx`. Sample statements:

```
-- The return value is  yxTxyomxx.
select rtrim(' yxTxyomxx ');
-- The preceding statement is equivalent to the following statement:
select trim(trailing from ' yxTxyomxx ');
```

  - Example 2: Remove the substring that consists of one or more characters in the `xy` collection from the right side of the string `yxTxyomxx`.

```
-- The return value is yxTxyom. If x or y appears on the right side of a string, it is removed.
select ltrim('yxTxyomxx', 'xy');
-- The preceding statement is equivalent to the following statement:
select trim(trailing 'xy' from 'yxTxyomxx');
```

  - Example 3: The input parameter is set to null. Sample statements:

```
-- The return value is null.
select rtrim(null);
select ltrim('yxTxyomxx', 'null');
```

SPACE

- Syntax

```
string space(bigint <n>)
```

- Description

  Generates a space string with a length of n.

- Parameters

  n: required. A value of the BIGINT type. The returned string cannot exceed 2 MB in length.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If n is left empty, an error is returned.
  - If the value of n is null, null is returned.

- Examples

```
-- The return value is 10.
select length(space(10));
```

SPLIT_PART

- Syntax

```
string split_part(string <str>, string <separator>, bigint <start>[, bigint <end>])
```

- Description

  Uses a delimiter specified by separator to split a string specified by str, and returns a substring that starts from the character specified by start and ends with the character specified by end.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string that you want to split. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.
  - separator: required. A constant of the STRING type. This parameter specifies the delimiter that is used to split the string. The delimiter can be a character or a string.
  - start: required. A constant of the BIGINT type. The value of this parameter must be greater than 0. This parameter specifies the start position of the substring to be returned. The position starts from 1.
  - end: a constant of the BIGINT type. The value of this parameter must be greater than or equal to start. This parameter specifies the end position of the substring to be returned. If this parameter is not specified, the value of this parameter is equal to start and the substring that starts from the character specified by start is returned.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If start is set to a value greater than the number of substrings, for example, the string has 6 substrings but the start value is greater than 6, an empty string is returned.
  - If separator is absent from a string specified by str and start is set to 1, the entire string specified by str is returned. If str is an empty string, an empty string is returned.
  - If separator is an empty string, the original string specified by str is returned.
  - If the value of end is greater than the number of substrings, the substring that starts from the character specified by start is returned.
  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of separator is not a constant of the STRING type, an error is returned.
  - If the value of start or end is not a constant of the BIGINT type, an error is returned.
  - If the value of an input parameter except separator is null, null is returned.

- Examples
  - Example 1: Use commas (`,`) as a delimiter to split the string `a,b,c,d` and return the substring that matches the specified rule. Sample statements:

    ```
    -- The return value is a.
    select split_part('a,b,c,d', ',', 1);
    -- The return value is a,b.
    select split_part('a,b,c,d', ',', 1, 2);
    ```

- Example 2: The value of start is greater than the number of substrings after the specified string is split into the substrings. Sample statement:

```
-- The return value is an empty string.
select split_part('a,b,c,d', ',', 10);
```

- Example 3: separator does not exist in the string specified by str. Sample statements:

```
-- The return value is a,b,c,d.
select split_part('a,b,c,d', ':', 1);
-- The return value is an empty string.
select split_part('a,b,c,d', ':', 2);
```

- Example 4: separator is an empty string. Sample statement:

```
-- The return value is a,b,c,d.
select split_part('a,b,c,d', '', 1);
```

- Example 5: The value of end is greater than the number of substrings after the specified string is split into the substrings. Sample statement:

```
-- The return value is b,c,d.
select split_part('a,b,c,d', ',', 2, 6);
```

- Example 6: An input parameter except separator is set to null. Sample statement:

```
-- The return value is null.
select split_part('a,b,c,d', ',', null);
```

SUBSTR

- Syntax

```
string substr(string <str>, bigint <start_position>[, bigint <length>])
```

- Description

  Returns a substring that starts from start_position in a string specified by str and has a length specified by length.

- Parameters
  - str: required. A value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, the value is implicitly converted into values of the STRING type before calculation.
  - start_position: required. A value of the BIGINT type. Default value: 1.
    - Hive-compatible data type edition: If start_position is set to 0, the return value is the same as that when this parameter is set to 1.
    - MaxCompute V1.0 and MaxCompute V2.0 data type editions: If start_position is set to 0, null is returned.
  - length: optional. A value of the BIGINT type. This parameter specifies the length of a substring. The value of this parameter must be greater than 0.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

- If the value of str is not of the STRING, BIGINT, DECIMAL, DOUBLE, or DATETIME type, an error is returned.
- If the value of length is not of the BIGINT type or is less than or equal to 0, an error is returned.
- If length is not specified, the substring from the start position to the end of str is returned.
- If the value of str, start_position, or length is null, null is returned.

- Examples
  - Example 1: Return a substring with the specified length that starts from the specified position in the string `abc`. Sample statements:

```
-- The return value is bc.
select substr('abc', 2);
-- The return value is b.
select substr('abc', 2, 1);
-- The return value is bc.
select substr('abc',-2 , 2);
```

  - Example 2: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select substr('abc', null);
```

SUBSTRING

- Syntax

```
string substring(string|binary <str>, int <start_position>[, int <length>])
```

- Description

  Returns a substring that starts from start_position in a string specified by str and has a length specified by length.

- Parameters
  - str: required. A value of the STRING or BINARY type.
  - start_position: required. A value of the INT type. The start position starts from 1. If start_position is set to 0, an empty string is returned. If start_position is a negative value, the search starts from the end to the start of the string and the last character is -1.
  - length: optional. A value of the BIGINT type. This parameter specifies the length of a substring. The value of this parameter must be greater than 0.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If str is not of the STRING or BINARY type, an error is returned.
  - If the value of length is not of the BIGINT type or is less than or equal to 0, an error is returned.
  - If length is not specified, the substring from the start position to the end of str is returned.
  - If the value of str, start_position, or length is null, null is returned.

- Examples

- Example 1: Return a substring with the specified length that starts from the specified position in the string `abc`. Sample statements:

```
-- The return value is bc.
select substring('abc', 2);
-- The return value is b.
select substring('abc', 2, 1);
-- The return value is bc.
select substring('abc',-2,2);
-- The return value is ab.
select substring('abc',-3,2);
-- The return value is 001.
substring(bin(2345), 2, 3);
```

- Example 2: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select substring('abc', null, null);
```

TO_CHAR

- Syntax

```
string to_char(boolean <value>)
string to_char(bigint <value>)
string to_char(double <value>)
string to_char(decimal <value>)
```

- Description

  Converts data of the BOOLEAN, BIGINT, DECIMAL, or DOUBLE type into the STRING type.

- Parameters

  value: required. A value of the BOOLEAN, BIGINT, DECIMAL, or DOUBLE type.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If value is not of the BOOLEAN, BIGINT, DECIMAL, or DOUBLE type, an error is returned.
  - If value is set to null, null is returned.

- Examples
  - Example 1: Convert values into the STRING type. Sample statements:

```
-- The return value is 123.
select to_char(123);
-- The return value is TRUE.
select to_char(true);
-- The return value is 1.23.
select to_char(1.23);
```

  - Example 2: The input parameter is set to null. Sample statement:

```
-- The return value is null.
select to_char(null);
```

TO_JSON

- Syntax

```
to_json(<expr>)
```

- Description

  Converts an expression of a given complex data type into a JSON string.

- Parameters

  expr: required. The expression of the ARRAY, MAP, or STRUCT type.

  > **Note** If the input expression is of the STRUCT type (`struct<key1:value1, key2:value2>`), take note of the following points:
  >
  > - All keys are converted into lowercase letters when you convert the expression into a JSON string.
  > - If a `value` is null, the key-value pair to which the `value` belongs is not included in the JSON string that is returned. For example, if `value2` is null, `key2:value2` is not included in the JSON string that is returned.

- Examples
  - Example 1: Convert an expression of a given data type into a JSON string. Sample statement:

```
-- {"a":1,"b":2} is returned.
select to_json(named_struct('a', 1, 'b', 2));
-- {"time":"26/08/2015"} is returned.
select to_json(named_struct('time', "26/08/2015"));
-- [{"a":1,"b":2}] is returned.
select to_json(array(named_struct('a', 1, 'b', 2)));
-- {"a":{"b":1}} is returned.
select to_json(map('a', named_struct('b', 1)));
-- {"a":1} is returned.
select to_json(map('a', 1));
-- [{"a":1}] is returned.
select to_json(array((map('a', 1))));
```

  - Example 2: The input expression is of the STRUCT type. Sample statement:

```
-- {"a":"B"} is returned. If the expression of the STRUCT type is converted into a JSON string, all keys are converted into lowercase letters.
select to_json(named_struct("A", "B"));
-- {"k2":"v2"} is returned. The key-value pair to which null belongs is not included in the JSON string that is returned.
select to_json(named_struct("k1", cast(null as string), "k2", "v2"));
```

TOLOWER

- Syntax

```
string tolower(string <source>)
```

- Description

Converts uppercase letters in a string specified by source into lowercase letters.

- Parameters

  source: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of source is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of source is null, null is returned.

- Examples
  - Example 1: Convert uppercase letters in a string into lowercase letters. Sample statements:

    ```
    -- The return value is abcd.
    select tolower('aBcd');
    -- The return value is china fighting.
    Select tolower ('China Fighting');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select tolower(null);
    ```

TOUPPER

- Syntax

  ```
  string toupper(string <source>)
  ```

- Description

  Converts lowercase letters in a string specified by source into uppercase letters.

- Parameters

  source: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of source is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of source is null, null is returned.

- Examples
  - Example 1: Convert lowercase letters in a string into uppercase letters. Sample statements:

    ```
    -- The return value is ABCD.
    select toupper('aBcd');
    -- The return value is CHINA FIGHTING.
    select toupper('China Fighting');
    ```

- Example 2: The input parameter is set to null. Sample statement:

```
-- The return value is null.
select toupper(null);
```

TRIM

- Syntax

```
string trim(string <str>[,<trimChars>])
string trim([BOTH] [<trimChars>] from <str>)
```

- Description

  Removes the characters from both of the left and right sides of a string that is specified by str.

  - If you do not specify trimChars, the spaces on both of the left and right sides are removed by default.
  - If you specify trimChars, the substrings that consist of one or more characters specified by trimChars are removed from both of the left and right sides of the string that is specified by str.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string from both of the left and right sides of which the characters are removed. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, the value is implicitly converted into a value of the STRING type before calculation.
  - trimChars: optional. A value of the STRING type. This parameter specifies the characters to be removed.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:

  - If the value of str is not of the STRING, BIGINT, DOUBLE, DECIMAL, or DATETIME type, an error is returned.
  - If the value of str or trimChars is null, null is returned.

- Examples
  - Example 1: Remove the space from both of the left and right sides of the string `yxTxyomxx`. Sample statements:

```
-- The return value is yxTxyomxx.
select trim(' yxTxyomxx ');
-- The preceding statement is equivalent to the following statement:
select trim(both from ' yxTxyomxx ');
select trim(from ' yxTxyomxx ');
```

- Example 2: Remove the substrings that consist of one or more characters in the `xy` collection from both of the left and right sides of the string `yxTxyomxx`.

```
-- The return value is yxTxyom. If x or y appears on the left or right side, it is remo
ved.
select trim('yxTxyomxx', 'xy');
-- The preceding statement is equivalent to the following statement:
select trim(both 'xy' from 'yxTxyomxx');
select trim('xy' from 'yxTxyomxx');
```

- Example 3: The input parameter is set to null. Sample statements:

```
-- The return value is null.
select trim(null);
select trim('yxTxyomxx', null);
```

URL_DECODE

- Syntax

```
string url_decode(string <input>[, string <encoding>])
```

- Description

  Converts an input string from the `application/x-www-form-urlencoded MIME` format into a normal string. This is the inverse function of `url_encoding`. The encoding format must comply with the following rules:

  - All letters remain unchanged.
  - Periods (.), hyphens (-), asterisks (*), and underscores (_) remain unchanged.
  - Plus signs (+) are converted into spaces.
  - The `%xy` formatted sequence is converted into byte values. Consecutive byte values are decoded to the related strings based on the value of encoding.
  - Other characters remain unchanged.

- Parameters
  - input: required. A value of the STRING type. This parameter specifies the string that you want to enter.
  - encoding: optional. This parameter specifies the encoding format, which can be GBK or UTF-8. If you do not specify this parameter, the default value is UTF-8.

- Return value

  Returns a UTF-8 encoded string of the STRING type. If the value of input or encoding is null, null is returned.

- Examples

```
-- The return value is for url_decode:// (fdsf).
select url_decode('%E7%A4%BA%E4%BE%8Bfor+url_decode%3A%2F%2F+%28fdsf%29');
-- The return value is Example for URL_DECODE:// dsf(fasfs).
select url_decode('Example+for+url_decode+%3A%2F%2F+dsf%28fasfs%29', 'GBK');
```

URL_ENCODE

- Syntax

```
string url_encode(string <input>[, string <encoding>])
```

- Description

  Encodes the input string in the `application/x-www-form-urlencoded MIME` format and returns the encoded string. The encoding format must comply with the following rules:

  - All letters remain unchanged.
  - Periods (.), hyphens (-), asterisks (*), and underscores (_) remain unchanged.
  - Spaces are converted into plus signs (+).
  - Other characters are converted into byte values based on the specified encoding format. Each byte value is then represented in the `%xy` format, where `xy` is the hexadecimal representation of the character value.

- Parameters
  - input: required. A value of the STRING type. This parameter specifies the string that you want to enter.
  - encoding: optional. This parameter specifies the encoding format, which can be GBK or UTF-8. If you do not specify this parameter, the default value is UTF-8.

- Return value

  A value of the STRING type is returned. If the value of input or encoding is null, null is returned.

- Examples

```
-- The return value is %E7%A4%BA%E4%BE%8Bfor+url_encode%3A%2F%2F+%28fdsf%29.
select url_encode('Example for url_encode:// (fdsf)');
-- The return value is Example+for+url_encode+%3A%2F%2F+dsf%28fasfs%29.
select url_encode('Example for url_encode:// dsf(fasfs)', 'GBK');
```

CONCAT_WS

- Syntax

```
string concat_ws(string <separator>, string <str1>, string <str2>[,...])
string concat_ws(string <separator>, array<string> <a>)
```

- Description

  Concatenates all strings in a parameter or elements in an array by using the specified delimiter and returns the result. This function is an extension function of MaxCompute V2.0.

- Parameters
  - separator: required. A delimiter of the STRING type.
  - str1 and str2: At least two strings must be specified. Values of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, the value is implicitly converted into values of the STRING type before calculation.
  - a: required. This parameter specifies an array. The elements in the array are of the STRING type.

- Return value

  A value of the STRING or STRUCT type is returned. The return value varies based on the following rules:

- If the value of str1 or str2 is not of the STRING, BIGINT, DECIMAL, DOUBLE, or DATETIME type, an error is returned.
- If no input parameters are present or an input parameter is set to null, null is returned.

- Examples
  - Example 1: Concatenate the strings `name` and `hanmeimei` by using a colon (`:`). Sample statement:

    ```
    -- The return value is name:hanmeimei.
    select concat_ws(':','name','hanmeimei');
    ```

    -
  - Example 2: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select concat_ws(':','avg',null,'34');
    ```

  - Example 3: Concatenate the elements in the `array('name', 'hanmeimei')` array by using colons (`:`). Sample statement:

    ```
    -- The return value is name:hanmeimei.
    select concat_ws(':',array('name', 'hanmeimei'));
    ```

JSON_TUPLE

- Syntax

  ```
  string json_tuple(string <json>, string <key1>, string <key2>,...)
  ```

- Description

  Extracts strings from a standard JSON string based on a set of input keys, such as `(key1,key2,...)` .

- Parameters
  - json: required. A value of the STRING type. This parameter specifies a standard JSON string.
  - key: required. A value of the STRING type. This parameter is used to describe the `path` of a JSON object in the JSON string. The value cannot start with a dollar sign ($). You can enter multiple keys at a time. MaxCompute parses JSON objects by using `.` or `['']` . If a key in a JSON object contains a period (.), ['] can be used.

- Return value

  A value of the STRING type is returned.

Note

- If json is empty or invalid, NULL is returned.
- If key is empty, invalid, or does not exist in the JSON string, NULL is returned.
- If json is valid and key exists, the related string is returned.
- This function can parse JSON data that contains Chinese characters.
- This function can parse nested JSON data.
- This function can parse JSON data that contains nested arrays.
- The parsing action is equivalent to the execution of GET_JSON_OBJECT along with `set od ps.sql.udf.getjsonobj.new=true;`. To obtain multiple objects from a JSON string, you must call the GET_JSON_OBJECT function multiple times. As a result, the JSON string is parsed multiple times. The JSON_TUPLE function allows you to enter multiple keys at a time and the JSON string is parsed only once. JSON_TUPLE is more efficient than GET_JSON_OBJECT.
- JSON_TUPLE is a UDTF and is used with Lateral View when other columns need to be selected.

LPAD

- Syntax

```
string lpad(string <str1>, int <length>, string <str2>)
```

- Description

  Left pads str1 with str2 to a specified length. This function is an extension function of MaxCompute V2.0.

- Parameters
  - str1: required. A value of the STRING type. This parameter specifies the string that you want to left pad.
  - length: required. A value of the INT type. This parameter specifies the number of characters that are used for left padding.
  - str2: required. This parameter specifies the string that you use to left pad another string.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of length is less than the number of characters in str1, this function truncates str1 from the left to obtain a string with the number of characters specified by length.
  - If length is set to 0, an empty string is returned.
  - If no input parameters are present or an input parameter is set to null, null is returned.

- Examples
  - Example 1: Left pad the string `abcdefgh` with the string `12` to obtain a string with 10 characters in length. Sample statement:

```
-- The return value is 12abcdefgh.
select lpad('abcdefgh', 10, '12');
```

- Example 2: Left pad the string `abcdefgh` with the string `12` to obtain a string with 5 characters in length. Sample statement:

```
-- The return value is abcde.
select lpad('abcdefgh', 5, '12');
```

- Example 3: The value of length is 0. Sample statement:

```
-- The return value is an empty string.
select lpad('abcdefgh' ,0, '12');
```

- Example 4: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select lpad(null ,0, '12');
```

RPAD

- Syntax

```
string rpad(string <str1>, int <length>, string <str2>)
```

- Description

  Right pads str1 with str2 to a specified length. This function is an extension function of MaxCompute V2.0.

- Parameters
  - str1: required. A value of the STRING type. This parameter specifies the string that you want to right pad.
  - length: required. A value of the INT type. This parameter specifies the number of characters that are used for right padding.
  - str2: required. This parameter specifies the string that you use to left pad another string.

- Return value

  A value of the STRING type is returned. The return value varies based on the following rules:
  - If the value of length is less than the number of characters in str1, this function truncates str1 from the left to obtain a string with the number of characters specified by length.
  - If length is set to 0, an empty string is returned.
  - If no input parameters are present or an input parameter is set to null, null is returned.

- Examples
  - Example 1: Right pad the string `abcdefgh` with the string `12` to obtain a string with 10 characters in length. Sample statement:

```
-- The return value is abcdefgh12.
select rpad('abcdefgh', 10, '12');
```

  - Example 2: Right pad the string `abcdefgh` with the string `12` to obtain a string with 5 characters in length. Sample statement:

```
-- The return value is abcde.
select rpad('abcdefgh', 5, '12');
```

- Example 3: The value of length is 0. Sample statement:

```
-- The return value is an empty string.
select rpad('abcdefgh' ,0, '12');
```

- Example 4: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select rpad(null ,0, '12');
```

REPLACE

- Syntax

```
string replace(string <str>, string <old>, string <new>)
```

- Description

  Uses a string specified by new to replace the part of a string specified by str that is exactly the same as a string specified by old, and returns the obtained string. If no part of the string specified by str is exactly the same as the string specified by old, the original string is returned. This function is an extension function of MaxCompute V2.0.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string that you want to replace.
  - old: required. This parameter specifies the string that you use for comparison.
  - new: required. This parameter specifies the string that you use to replace the original string.
- Return value

  A value of the STRING type is returned. If an input parameter is set to null, null is returned.

- Examples
  - Example 1: Replace the part of the string `ababab` that is exactly the same as the string `abab` with the string `12`. Sample statement:

```
-- The return value is 12ab.
select replace('ababab','abab','12');
```

  - Example 2: An input parameter is set to null. Sample statement:

```
-- The return value is null.
select replace('123abab456ab',null,'abab');
```

SOUNDEX

- Syntax

```
string soundex(string <str>)
```

- Description

  Converts a normal string into a string of the SOUNDEX type.

- Parameters

str: required. A value of the STRING type. This parameter specifies the string that you want to convert. This function is an extension function of MaxCompute V2.0.

- Return value

  A value of the STRING type is returned. If the value of str is null, null is returned.

- Examples
  - Example 1: Convert the string `hello` into a string of the SOUNDEX type. Sample statement:

    ```
    -- The return value is H400.
    select soundex('hello');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select soundex(null);
    ```

SUBSTRING_INDEX

- Syntax

  ```
  string substring_index(string <str>, string <separator>, int <count>)
  ```

- Description

  Truncates the string str to a substring from the first character to the nth delimiter. n is specified by count. If count is a positive value, the string is truncated from left to right. If count is a negative value, the string is truncated from right to left. This function is an extension function of MaxCompute V2.0.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string that you want to truncate.
  - separator: required. A delimiter of the STRING type.
  - count: required. A value of the INT type. This parameter specifies the position of the delimiter.

- Return value

  A value of the STRING type is returned. If an input parameter is set to null, null is returned.

- Examples
  - Example 1: Truncate the string `https://www.alibabacloud.com`. Sample statements:

    ```
    -- The return value is https://www.alibabacloud.
    select substring_index('https://www.alibabacloud.com', '.', 2);
    -- The return value is alibabacloud.com.
    select substring_index('https://www.alibabacloud.com', '.', -2);
    ```

  - Example 2: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select substring_index('https://www.alibabacloud.com', null, 2);
    ```

TRANSLATE

- Syntax

```
string translate(string|varchar <str1>, string|varchar <str2>, string|varchar <str3>)
```

- Description

  Uses the specified characters in str3 to replace the characters that are included in str1 from str2. If characters that are included in str1 do not appear in str2, no character is replaced. This function is an extension function of MaxCompute V2.0.

- Return value

  A value of the STRING type is returned. If an input parameter is set to null, null is returned.

- Examples
  - Example 1: Use the specified characters in the string `cd` to replace the characters that are included in `abababab` from `abab`. Sample statement:

    ```
    -- The return value is cdcdcd.
    select translate('abababab','abab','cd');
    ```

  - Example 2: Use the specified characters in the string `cdefg` to replace the characters that are included in `abababab` from `abab`. Sample statement:

    ```
    -- The return value is cdcdcd.
    select translate('abababab','abab','cdefg');
    ```

  - Example 3: An input parameter is set to null. Sample statement:

    ```
    -- The return value is null.
    select translate('abababab','cd',null);
    ```

3.9.8. Complex type functions

You can use complex type functions in MaxCompute SQL to process data of complex data types, such as ARRAY, MAP, STRUCT, and JSON. This topic describes the syntax and parameters of complex type functions that are supported by MaxCompute SQL, and provides examples on how to use complex type functions. This topic guides you through data development by using complex type functions.

The following table describes the complex type functions that are supported by MaxCompute SQL.

Function type	Function	Description
	ALL_MATCH	Checks whether all the elements in an array meet a specific condition.
	ANY_MATCH	Checks whether an element in an array meets a specific condition.
	ARRAY	Creates an array based on given values.
	ARRAY_CONTAINS	Checks whether an array contains a given value.
	ARRAY_DISTINCT	Removes duplicate elements from an array.
	ARRAY_EXCEPT	Finds the elements that exist in Array A but do not exist in Array B and returns the elements as a new array without duplicates.

Function type	Function	Description
ARRAY functions	ARRAY_INTERSECT	Calculates the intersection of two arrays.
	ARRAY_JOIN	Concatenates the elements in an array by using a delimiter.
	ARRAY_MAX	Returns the largest element in an array.
	ARRAY_MIN	Returns the smallest element in an array.
	ARRAY_POSITION	Returns the position of the first occurrence of a given element in an array.
	ARRAY_REDUCE	Aggregates the elements in an array.
	ARRAY_REMOVE	Removes a given element from an array.
	ARRAY_REPEAT	Returns a new array in which a given element is repeated several times.
	ARRAY_SORT	Sorts the elements in an array based on a comparator.
	ARRAY_UNION	Calculates the union of two arrays and returns the union as a new array without duplicates.
	ARRAYS_OVERLAP	Checks whether two arrays contain the same element.
	ARRAYS_ZIP	Merges multiple arrays.
	CONCAT	Concatenates multiple arrays or strings.
	EXPLODE	Transposes one row of data into multiple rows. This function is a user-defined table-valued function (UDTF).
	FILTER	Filters the elements in an array.
	INDEX	Returns the element at a specific position in an array.
	POSEXPLODE	Converts an array into a table that has two columns. The first column lists the position of each element in the array, starting from 0. The second column lists the elements.
	SIZE	Returns the number of elements in an array.
	SLICE	Copies the elements in an array from a specific position based on a specific length and returns the elements as a new array.
	SORT_ARRAY	Sorts the elements in an array.
	TRANSFORM	Transforms the elements in an array.

Function type	Function	Description
	ZIP_WITH	Merges two arrays at the element level based on element positions and returns a new array.
MAP functions	EXPLODE	Transposes one row of data into multiple rows. This function is a UDTF.
	INDEX	Returns the value that meets a specific condition in a map.
	MAP	Creates a map based on given key-value pairs.
	MAP_CONCAT	Returns the union of multiple maps.
	MAP_ENTRIES	Converts key-value pairs in a map into a struct array.
	MAP_FILTER	Filters the elements in a map.
	MAP_FROM_ARRAYS	Creates a map based on given arrays.
	MAP_FROM_ENTRIES	Creates a map based on given struct arrays.
	MAP_KEYS	Returns all keys in a map as an array.
	MAP_VALUES	Returns all values in a map as an array.
	MAP_ZIP_WITH	Merges two given maps into a single map.
	SIZE	Returns the number of key-value pairs in a map.
	TRANSFORM_KEYS	Transforms the keys in a map by using a given function. The values in the map are not changed.
	TRANSFORM_VALUES	Transforms the values in a map by using a given function. The keys in the map are not changed.
STRUCT functions	FIELD	Obtains the value of a member variable in a struct.
	INLINE	Expands a given struct array. Each array element corresponds to a row and each struct element corresponds to a column in each row.
	STRUCT	Creates a struct based on a given value list.
	NAMED_STRUCT	Creates a struct based on given name-value pairs.
JSON functions	FROM_JSON	Returns data of the ARRAY, MAP, or STRUCT type based on a given JSON string and a given output format.
	GET_JSON_OBJECT	Extracts a single string from a standard JSON string by using a specific method.

Function type	Function	Description
	JSON_TUPLE	Extracts strings from a standard JSON string based on a set of input keys.
	TO_JSON	Converts data of a complex data type into a JSON string.

ALL_MATCH

- Syntax

```
boolean all_match(array<T> <a>, function<T, boolean> <predicate>)
```

- Description

  Checks whether all the elements in Array a meet the predicate condition.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - predicate: required. This parameter specifies the built-in function, user-defined function, or expression that is used to determine whether all the elements in Array a meet requirements. The data type of the input parameter must be the same as the data type of the elements in Array a.

- Return value

  A value of the BOOLEAN type is returned. The return value varies based on the following rules:
  - If all the elements in Array a meet the predicate condition or the array is empty, True is returned.
  - If one or more elements in Array a do not meet the predicate condition, False is returned.
  - If an element in Array a is null and other elements meet the predicate condition, NULL is returned.

- Examples
  - Example 1: Check whether all the elements in `array(4, 5, 6)` meet the `x-> x > 3` condition. This condition specifies that all elements are greater than 3. Sample statement:

```
-- True is returned.
select all_match(array(4, 5, 6), x -> x>3);
```

  - Example 2: The array is empty. Sample statement:

```
-- True is returned.
select all_match(array(), x -> x>3);
```

  - Example 3: Check whether all the elements in `array(1, 2, -10, 100, -30)` meet the `x-> x > 3` condition. Sample statement:

```
-- False is returned.
select all_match(array(1, 2, -10, 100, -30), x -> x>3);
```

  - Example 4: Check whether all the elements in `array(10, 100, 30, null)` meet the `x-> x > 3` condition. Sample statement:

```
-- NULL is returned.
select all_match(array(10, 100, 30, null), x -> x>3);
```

ANY_MATCH

- Syntax

```
boolean any_match(array<T> <a>, function<T, boolean> <predicate>)
```

- Description

  Checks whether an element in Array a meets the predicate condition.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - predicate: required. This parameter specifies the built-in function, user-defined function, or expression that is used to determine whether all the elements in Array a meet requirements. The data type of the input parameter must be the same as the data type of the elements in Array a.

- Return value

  A value of the BOOLEAN type is returned. The return value varies based on the following rules:
  - If one or more elements in Array a meet the predicate condition, True is returned.
  - If no elements in Array a meet the predicate condition or the array is empty, False is returned.
  - If an element in Array a is null and other elements do not meet the predicate condition, NULL is returned.

- Examples
  - Example 1: Check whether an element in `array(1, 2, -10, 100, -30)` meets the `x-> x > 3` condition. Sample statement:

```
-- True is returned.
select any_match(array(1, 2, -10, 100, -30), x-> x > 3);
```

  - Example 2: The array is empty. Sample statement:

```
-- False is returned.
select any_match(array(), x-> x > 3);
```

  - Example 3: Check whether an element in `array(1, 2, -10, -20, -30)` meets the `x-> x > 3` condition. Sample statement:

```
-- False is returned.
select any_match(array(1, 2, -10, -20, -30), x-> x > 3);
```

  - Example 4: Check whether an element in `array(1, 2, null, -10)` meets the `x-> x > 3` condition. Sample statement:

```
-- NULL is returned.
select any_match(array(1, 2, null, -10), x-> x > 3);
```

ARRAY

- Syntax

```
array array(<value>,<value>[, ...])
```

- Description

  Creates an array based on given values.

- Parameters

  value: required. All data types are supported. All values must be of the same data type.

- Return value

  A value of the ARRAY type is returned.

- Examples

  The `t_table` table contains the `c1 (BIGINT), c2 (STRING), c3 (STRING), c4 (BIGINT), and c5 (BIGINT)` fields. Data in the table:

```
+------------+----+----+------------+------------+
| c1         | c2 | c3 | c4         | c5         |
+------------+----+----+------------+------------+
| 1000       | k11 | k21 | 86         | 15         |
| 1001       | k12 | k22 | 97         | 2          |
| 1002       | k13 | k23 | 99         | 1          |
+------------+----+----+------------+------------+
```

  Sample statement:

```
-- Create an array based on the data in the c2, c4, c3, and c5 columns.
select array(c2,c4,c3,c5) from t_table;
-- The following result is returned:
+------+
| _c0  |
+------+
| [k11, 86, k21, 15] |
| [k12, 97, k22, 2] |
| [k13, 99, k23, 1] |
+------+
```

ARRAY_CONTAINS

- Syntax

```
boolean array_contains(array<T> <a>, value <v>)
```

- Description

  Checks whether Array a contains Element v.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - v: required. This parameter specifies the element that you want to check. The value of v must be of the same data type as the elements in Array a.

- Return value

  A value of the BOOLEAN type is returned.

- Examples

The `t_table_array` table contains the `c1 (BIGINT) and t_array (ARRAY<STRING>)` fields. Data in the table:

```
+------------+---------+
| c1         | t_array |
+------------+---------+
| 1000       | [k11, 86, k21, 15] |
| 1001       | [k12, 97, k22, 2] |
| 1002       | [k13, 99, k23, 1] |
+------------+---------+
```

Sample statement:

```
-- Check whether the t_array column contains the value 1.
select c1, array_contains(t_array,'1') from t_table_array;
-- The following result is returned:
+------------+------+
| c1         | _c1  |
+------------+------+
| 1000       | false |
| 1001       | false |
| 1002       | true |
+------------+------+
```

ARRAY_DISTINCT

- Syntax

```
array<T> array_distinct(array<T> <a>)
```

- Description

  Removes duplicate elements from Array a.

- Parameters

  a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.

- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:

  - The returned array has no duplicate elements, and the elements are sorted in the same order as Array a.
  - If an element in Array a is null, the null element is involved in the operation.
  - If the input array is empty, an empty array is returned.

- Examples

  - Example 1: Remove duplicate elements from `array(10, 20, 30, 30, 20, 10)`. Sample statement:

    ```
    -- [10,20,30] is returned.
    select array_distinct(array(10, 20, 30, 30, 20, 10));
    ```

- Example 2: Remove duplicate elements from `array(10, 20, 20, null, null, 30, 20, null)`. Sample statement:

```
-- [10,20,null,30] is returned.
select array_distinct(array(10, 20, 20, null, null, 30, 20, null));
```

- Example 3: The array is empty. Sample statement:

```
-- [] is returned.
select array_distinct(array());
```

ARRAY_EXCEPT

- Syntax

```
array<T> array_except(array<T> <a>, array<T> <b>)
```

- Description

  Finds the elements that exist in Array a but do not exist in Array b and returns the elements as a new array without duplicates.

- Parameters

  a and b: required. These parameters specify arrays that `T` in `array<T>` specifies the data type of the elements in the arrays. The elements can be of any data type. The elements in Array a and the elements in Array b must be of the same data type.

- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:

  - The returned array has no duplicate elements, and the elements are sorted in the same order as Array a.
  - If an element in an array is null, the null element is involved in the operation.
  - If one of the input arrays is empty, a new array is returned based on the non-empty array without duplicates.
  - If the two input arrays are empty, an empty array is returned.

- Examples
  - Example 1: Find the elements that exist in `array(1, 1, 3, 3, 5, 5)` but do not exist in `array(1, 1, 2, 2, 3, 3)` and return the elements as a new array without duplicates. Sample statement:

```
-- [5] is returned.
select array_except(array(1, 1, 3, 3, 5, 5), array(1, 1, 2, 2, 3, 3));
```

  - Example 2: Find the elements that exist in `array(1, 1, 3, 3, 5, 5, null, null)` but do not exist in `array(1, 1, 2, 2, 3, 3)` and return the elements as a new array without duplicates. Sample statement:

```
-- [5,null] is returned.
select array_except(array(1, 1, 3, 3, 5, 5, null, null), array(1, 1, 2, 2, 3, 3));
```

- Example 3: One of the input arrays is empty. Sample statement:

```
-- [2,1] is returned.
select array_except(array(2, 1, 1, 2), cast(array() as array<int>));
```

- Example 4: The two input arrays are empty. Sample statement:

```
-- [] is returned.
select array_except(cast(array() as array<int>), cast(array() as array<int>));
```

ARRAY_INTERSECT

- Syntax

```
array<T> array_intersect(array<T> <a>, array<T> <b>)
```

- Description

  Calculates the intersection of Array a and Array b and removes duplicate elements.

- Parameters

  a and b: required. These parameters specify arrays that `T` in `array<T>` specifies the data type of the elements in the arrays. The elements can be of any data type. The elements in Array a and the elements in Array b must be of the same data type.

- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:

  - If an element in an array is null, the null element is involved in the operation.
  - The returned array has no duplicate elements, and the elements are sorted in the same order as Array a.
  - If Array a or Array b is null, NULL is returned.

- Examples
  - Example 1: Calculate the intersection of `array(1, 2, 3)` and `array(1, 3, 5)`, and remove duplicate elements. Sample statement:

```
-- [1,3] is returned.
select array_intersect(array(1, 2, 3), array(1, 3, 5));
```

  - Example 2: Calculate the intersection of `array(10, 20, 20, 30, 30, null, null)` and `array(30, 30, 20, 20, 40, null, null)`, and remove duplicate elements. Sample statement:

```
-- [20,30,null] is returned.
select array_intersect(array(10, 20, 20, 30, 30, null, null), array(30, 30, 20, 20, 40,
null, null));
```

ARRAY_JOIN

- Syntax

```
array_join(array<T> <a>, <delimiter>[, <nullreplacement>])
```

- Description

Concatenates the elements in Array a by using a delimiter. If the array contains a null element, use nullreplacement to specify the string that you want to use to replace the null element in the results. If you do not configure nullreplacement, the null element is ignored.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array.

    > **Note** If the elements in the array are not of the STRING type, MaxCompute converts the data type of the elements into STRING.

  - delimiter: required. A value of the STRING type. This parameter specifies the string that is used to separate the concatenated elements in Array a.
  - nullreplacement: optional. This parameter specifies the string that is used to replace null elements.
- Return value

  A value of the STRING type is returned.
- Examples

```
-- 10,20,20,30 is returned.
select array_join(array(10, 20, 20, null, null, 30), ",");
-- 10##20##20##null##null##30 is returned.
select array_join(array(10, 20, 20, null, null, 30), "##", "null");
```

ARRAY_MAX

- Syntax

```
T array_max(array<T> <a>)
```

- Description

  Returns the largest element in Array a.
- Parameters

  a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array.

  The following data types are supported:
  - TINYINT, SMALLINT, INT, and BIGINT
  - FLOAT and DOUBLE
  - BOOLEAN
  - DECIMAL and DECIMALVAL
  - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
  - STRING, BINARY, VARCHAR, and CHAR
  - ARRAY, STRUCT, and MAP
- Return value

  The largest element in Array a is returned. The return value varies based on the following rules:
  - If Array a is null, NULL is returned.

- If an element in Array a is null, the null element is not involved in the operation.

- Examples

```
-- 20 is returned.
select array_max(array(1, 20, null, 3));
```

ARRAY_MIN

- Syntax

```
T array_min(array<T> <a>)
```

- Description

  Returns the smallest element in Array a.

- Parameters

  a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array.

  The following data types are supported:

  - TINYINT, SMALLINT, INT, and BIGINT
  - FLOAT and DOUBLE
  - BOOLEAN
  - DECIMAL and DECIMALVAL
  - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
  - STRING, BINARY, VARCHAR, and CHAR
  - ARRAY, STRUCT, and MAP

- Return value

  The smallest element in Array a is returned. The return value varies based on the following rules:

  - If Array a is null, NULL is returned.
  - If an element in Array a is null, the null element is not involved in the operation.

- Examples

```
-- 1 is returned.
select array_min(array(1, 20, null, 3));
```

ARRAY_POSITION

- Syntax

```
bigint array_position(array<T> <a>, T <element>)
```

- Description

  Returns the position of the first occurrence of a given element in Array a. The position numbers of elements are counted from left to right and start from 1.

- Parameters

- a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The following data types are supported:
  - TINYINT, SMALLINT, INT, and BIGINT
  - FLOAT and DOUBLE
  - BOOLEAN
  - DECIMAL and DECIMALVAL
  - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
  - STRING, BINARY, VARCHAR, and CHAR
  - ARRAY, STRUCT, and MAP
- element: required. The element whose position you want to query. The data type of this parameter must be the same as the data type of the elements in Array a.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the following rules:
  - If Array a or element is null, NULL is returned.
  - If the specific element is not found, 0 is returned.
- Examples
  - Example 1: Return the position of the first occurrence of `1` in `array(3, 2, 1)`. Sample statement:

    ```
    -- 3 is returned.
    select array_position(array(3, 2, 1), 1);
    ```
  - Example 2: element is null. Sample statement:

    ```
    -- NULL is returned.
    select array_position(array(3, 1, null), null);
    ```

ARRAY_REMOVE

- Syntax

  ```
  array<T> array_remove(array<T> <a>, T <element>)
  ```
- Description

  Removes a given element from Array a.
- Parameters

  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The following data types are supported:
    - TINYINT, SMALLINT, INT, and BIGINT
    - FLOAT and DOUBLE
    - BOOLEAN
    - DECIMAL and DECIMALVAL
    - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
    - STRING, BINARY, VARCHAR, and CHAR
    - ARRAY, STRUCT, and MAP
  - element: required. The element that you want to remove. The data type of this parameter must be the same as the data type of the elements in Array a.
- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:
  - If an element in Array a is null, the null element is not involved in the operation.
  - If Array a or element is null, NULL is returned.
  - If Array a does not contain the specific element, Array a is returned.
- Examples
  - Example 1: Remove `1` from `array(3, 2, 1)`. Sample statement:

```
-- [3,2] is returned.
select array_remove(array(3, 2, 1), 1);
```

  - Example 2: element is null. Sample statement:

```
-- NULL is returned.
select array_remove(array(3, 1, null), null);
```

  - Example 3: Remove `2` from `array(3, 1, null)`. Sample statement:

```
-- [3,1,null] is returned.
select array_remove(array(3, 1, null), 2);
```

ARRAY_REDUCE

- Syntax

```
R array_reduce(array<T> <a>, buf <init>, function<buf, T, buf> <merge>, function<buf, R>
<final>)
```

- Description

  Aggregates the elements in Array a.
- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - init: required. The initial value of the intermediate result that is used to aggregate elements.

- merge: required. A built-in function, user-defined function, or expression that is used to perform an operation on each element in Array a and the intermediate result. This function or expression uses the elements of Array a and the init parameter as input parameters.
- final: required. A built-in function, user-defined function, or expression that is used to convert the intermediate result into the final result. This function or expression uses the result of merge as the input parameter. R specifies the data type of the output.

- Return value

  The data type of the return value is the same as the data type that is specified for final.

- Examples

```
-- 6 is returned.
select array_reduce(array(1, 2, 3), 0, (buf, e)->buf + e, buf->buf);
-- 2.5 is returned.
select array_reduce(array(1, 2, 3, 4), named_struct('sum', 0, 'count', 0), (buf, e)->name
d_struct('sum', buf.sum + e, 'count', buf.count + 1), buf -> buf.sum / buf.count);
```

ARRAY_REPEAT

- Syntax

```
array<T> array_repeat(T <element>, int <count>)
```

- Description

  Returns a new array in which element t is repeated count times.

- Parameters
  - t: required. This parameter specifies the element that you want to repeat. The following data types are supported:
    - TINYINT, SMALLINT, INT, and BIGINT
    - FLOAT and DOUBLE
    - BOOLEAN
    - DECIMAL and DECIMALVAL
    - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
    - STRING, BINARY, VARCHAR, and CHAR
    - ARRAY, STRUCT, and MAP
  - count: required. This parameter specifies the number of repetitions. A value of the INT type is required. The value must be greater than or equal to 0.

- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:
  - If the value of count is null, NULL is returned.
  - If the value of count is less than 0, an empty array is returned.

- Examples

- Example 1: Repeat `123` `twice` and return the new array. Sample statement:

```
-- [123, 123] is returned.
select array_repeat('123', 2);
```

- Example 2: The value of count is null. Sample statement:

```
-- NULL is returned.
select array_repeat('123', null);
```

- Example 3: The value of count is less than 0. Sample statement:

```
-- [] is returned.
select array_repeat('123', -1);
```

ARRAY_SORT

- Syntax

```
array<T> array_sort(array<T> <a>, function<T, T, bigint> <comparator>)
```

- Description

  Sorts the elements in Array a based on a comparator.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - comparator: required. A built-in function, user-defined function, or expression that is used to compare two elements in the array.

    Processing logic of `comparator(a, b)`: If a is equal to b, 0 is returned. If a is less than b, a negative integer is returned. If a is greater than b, a positive integer is returned. If `comparator(a, b)` returns NULL, an error is returned.

- Return value

  A value of the ARRAY type is returned.

- Examples

```
-- [{"a":1,"b":10},{"a":2,"b":12},{"a":3,"b":11}] is returned.
select array_sort(a, (a,b)->case when a.a> b.a then 1L when a.a=b.a then 0L else -1L end)
from values (
  array(named_struct('a', 1, 'b', 10),
        named_struct('a', 3, 'b', 11),
        named_struct('a', 2, 'b', 12)))
  as t(a);
```

ARRAY_UNION

- Syntax

```
array<T> array_union(array<T> <a>,  array<T> <b>)
```

- Description

Calculates the union of Array a and Array b, and removes duplicate elements.

- Parameters

  a and b: required. These parameters specify arrays that `T` in `array<T>` specifies the data type of the elements in the arrays. The elements can be of any data type. The elements in Array a and the elements in Array b must be of the same data type.

  The following data types are supported:

  - TINYINT, SMALLINT, INT, and BIGINT
  - FLOAT and DOUBLE
  - BOOLEAN
  - DECIMAL and DECIMALVAL
  - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
  - STRING, BINARY, VARCHAR, and CHAR
  - ARRAY, STRUCT, and MAP

- Return value

  A value of the ARRAY type is returned. If Array a or Array b is null, NULL is returned.

- Examples
  - Example 1: Calculate the union of `array(1, 2, 3)` and `array(1, 3, 5)`, and remove duplicate elements. Sample statement:

    ```
    -- [1,2,3,5] is returned.
    select array_union(array(1, 2, 3), array(1, 3, 5));
    ```

  - Example 2: One of the arrays is null. Sample statement:

    ```
    -- NULL is returned.
    select array_union(array(1, 2, 3), null);
    ```

ARRAYS_OVERLAP

- Syntax

  ```
  boolean arrays_overlap(array<T> <a>,  array<T> <b>)
  ```

- Description

  Checks whether Array a and Array b contain the same element.

- Parameters

  a and b: required. These parameters specify arrays that `T` in `array<T>` specifies the data type of the elements in the arrays. The elements can be of any data type. The elements in Array a and the elements in Array b must be of the same data type.

  The following data types are supported:

  - TINYINT, SMALLINT, INT, and BIGINT
  - FLOAT and DOUBLE
  - BOOLEAN
  - DECIMAL and DECIMALVAL

- DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth
- STRING, BINARY, VARCHAR, and CHAR
- ARRAY, STRUCT, and MAP

- Return value

  A value of the BOOLEAN type is returned. The return value varies based on the following rules:

  - If Array a contains at least one element that is in Array b and is not null, True is returned.
  - If Array a and Array b do not contain the same element, both of the arrays are not empty, and one or both of the arrays contain a null element, NULL is returned.
  - If Array a and Array b do not contain the same element, and both of the arrays are not empty and do not contain a null element, False is returned.

- Examples
  - Example 1: Check whether `array(1, 2, 3)` and `array(3, 4, 5)` contain the same element. Sample statement:

    ```
    -- True is returned.
    select arrays_overlap(array(1, 2, 3), array(3, 4, 5));
    ```

  - Example 2: Check whether `array(1, 2, 3)` and `array(6, 4, 5)` contain the same element. Sample statement:

    ```
    -- False is returned.
    select arrays_overlap(array(1, 2, 3), array(6, 4, 5));
    ```

  - Example 3: One of the arrays contains a null element. Sample statement:

    ```
    -- NULL is returned.
    select arrays_overlap(array(1, 2, 3), array(5, 4, null));
    ```

ARRAYS_ZIP

- Syntax

  ```
  array<struct<T, U, ...>> arrays_zip(array<T> <a>, array<U> <b>[, ...])
  ```

- Description

  Merges multiple given arrays and returns a struct array, in which the Nth struct contains all the Nth elements of the input arrays.

- Parameters

  a and b: required. These parameters specify arrays that `T` in `array<T>` and `U` in `array<U>` specify the data types of the elements in the arrays. The elements can be of any data type.

  The following data types are supported:

  - TINYINT, SMALLINT, INT, and BIGINT
  - FLOAT and DOUBLE
  - BOOLEAN
  - DECIMAL and DECIMALVAL
  - DATE, DATETIME, TIMESTAMP, IntervalDayTime, and IntervalYearMonth

- STRING, BINARY, VARCHAR, and CHAR
- ARRAY, STRUCT, and MAP

- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:

  - The Nth struct in the generated struct array contains all the Nth elements of the input arrays. If an array contains less than N elements, null is used as the Nth element of the array.
  - If one or more input arrays are null, NULL is returned.

- Examples
  - Example 1: Merge `array(1, 2, 3)` and `array(2, 3, 4)` into a struct array. Sample statement:

    ```
    -- [{0:1, 1:2}, {0:2, 1:3}, {0:3, 1:4}] is returned.
    select arrays_zip(array(1, 2, 3), array(2, 3, 4));
    ```

  - Example 2: Merge `array(1, 2, 3)` and `array(4, 5)` into a struct array. Sample statement:

    ```
    -- [{0:1, 1:4}, {0:2, 1:5}, {0:3, 1:null}] is returned.
    select arrays_zip(array(1, 2, 3), array(4, 5));
    ```

CONCAT

- Syntax

  ```
  array<T> concat(array<T> <a>, array<T> <b>[,...])
  string concat(string <str1>, string <str2>[,...])
  ```

- Description
  - Arrays as inputs: Concatenates all elements of multiple arrays and returns a new array.
  - Strings as inputs: Concatenates multiple strings and returns a new string.

- Parameters
  - a and b: required. These parameters specify arrays that `T` in `array<T>` specifies the data type of the elements in the arrays. The elements can be of any data type. The elements in Array a and the elements in Array b must be of the same data type. The null elements are also involved in the operation.
  - str1 and str2: required. A value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

- Return value
  - A value of the ARRAY type is returned. If one of the input arrays is null, NULL is returned.
  - A value of the STRING type is returned. If no input parameters are configured or an input parameter is set to null, NULL is returned.

- Examples

- Example 1: Concatenate all elements of `array(10, 20)` and `array(20, -20)`. Sample statement:

```
-- [10, 20, 20, -20] is returned.
select concat(array(10, 20), array(20, -20));
```

- Example 2: One of the input arrays contains a null element. Sample statement:

```
-- [10, null, 20, -20] is returned.
select concat(array(10, null), array(20, -20));
```

- Example 3: One of the input arrays is null. Sample statement:

```
-- NULL is returned.
select concat(array(10, 20), null);
```

- Example 4: Concatenate strings `aabc` and `abcde`. Sample statement:

```
-- aabcabcde is returned.
select concat('aabc','abcde');
```

- Example 5: The input is empty. Sample statement:

```
-- NULL is returned.
select concat();
```

- Example 6: One of the input strings is null. Sample statement:

```
-- NULL is returned.
select concat('aabc', 'abcde', null);
```

EXPLODE

- Limits
  - A `SELECT` statement can contain only one `EXPLODE` function, and no other columns of a table are allowed.
  - This function cannot be used with the `GROUP BY`, `CLUSTER BY`, `DISTRIBUTE BY`, or `SORT BY` clause.
- Syntax

```
explode (<var>)
```

- Description

  Transposes one row of data into multiple rows. This function is a UDTF.
  - If the parameter value is of the `array<T>` type, the array stored in the column is transposed into multiple rows.
  - If the parameter value is of the `map<K, V>` type, each key-value pair of the map stored in the column is transposed into one row with two columns. One column is used to store keys, and the other column is used to store values.
- Parameters

  var: required. The value must be of the `array<T>` or `map<K, V>` type.

- Return value

  Rows after transposition are returned.

- Examples

  The `t_table_map` table contains the `c1 (BIGINT) and t_map (MAP<STRING,BIGINT>)` fields. Data in the table:

```
  +------------+-------+
| c1         | t_map |
+------------+-------+
| 1000       | {k11:86, k21:15} |
| 1001       | {k12:97, k22:2} |
| 1002       | {k13:99, k23:1} |
+------------+-------+
```

  Sample statement:

```
select explode(t_map) from t_table_map;
-- The following result is returned:
+-----+------------+
| key | value      |
+-----+------------+
| k11 | 86         |
| k21 | 15         |
| k12 | 97         |
| k22 | 2          |
| k13 | 99         |
| k23 | 1          |
+-----+------------+
```

FILTER

- Syntax

```
array<T> filter(array<T> <a>, function<T,boolean> <func>)
```

- Description

  Filters the elements in Array a by using func and returns a new array.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - func: required. This parameter specifies the built-in function, user-defined function, or expression that is used to filter the elements in Array a. The value must be of the same data type as the elements in Array a. The output result of the function or expression is of the BOOLEAN type.

- Return value

  A value of the ARRAY type is returned.

- Examples

```
-- [2, 3] is returned.
select filter(array(1, 2, 3), x -> x > 1);
```

INDEX

- Syntax

```
index(<var1>[<var2>])
```

- Description
  - If var1 is of the `map<K, V>` type, this function obtains the element that is at position var2 in var1. The position numbers of elements are counted from left to right and start from 0.
  - If var1 is of the `map<K, V>` type, this function obtains the value whose key is var2 in var1.

> **Note** When you use this function, you must remove `index` and directly execute `<var1>[<var2>]`. Otherwise, an error is returned.

- Parameters
  - var1: required. The value must be of the `array<T>` or `map<K, V>` type. `T` in `array<T>` specifies the data type of the elements in an array. The elements can be of any data type. `K` and `V` in `map<K, V>` specify the keys and values of a map.
  - var2: required.
    - If var1 is of the `array<T>` type, var2 must be of the BIGINT type and greater than or equal to 0.
    - If var1 is of the `map<K, V>` type, var2 must be of the same data type as K.
- Return value
  - If var1 is of the `array<T>` type, a value of the data type that is specified by T is returned. The return value varies based on the following rules:
    - If the number of elements in var1 is less than var2, NULL is returned.
    - If var1 is null, NULL is returned.
  - If var1 is of the `map<K, V>` type, a value that is of the same data type as V is returned. The return value varies based on the following rules:
    - If `map<K, V>` does not contain a key whose value is var2, NULL is returned.
    - If var1 is null, NULL is returned.
- Examples
  - Example 1: var1 is of the `array<T>` type. Sample statement:

```
-- c is returned.
select array('a','b','c')[2];
```

  - Example 2: var1 is of the `map<K, V>` type. Sample statement:

```
-- 1 is returned.
select str_to_map("test1=1,test2=2")["test1"];
```

POSEXPLODE

- Syntax

```
posexplode(array<T> <a>)
```

- Description

  Converts Array a into a table that has two columns. The first column lists the position of each element in the array, starting from 0. The second column lists the elements.

- Parameters

  a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.

- Return value

  A table is returned.

- Examples

```
select posexplode(array('a','c','f','b'));
-- The following result is returned:
+------------+------------+
| pos        | val        |
+------------+------------+
| 0          | a          |
| 1          | c          |
| 2          | f          |
| 3          | b          |
+------------+------------+
```

SIZE

- Syntax

```
int size(array<T> <a>)
int size(map<K, V> <b> )
```

- Description
  - An array as the input: Calculates the number of elements in Array a.
  - A map as the input: Calculates the number of key-value pairs in Map b.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - b: required. This parameter specifies a map. `K` and `V` in `map<K, V>` specify the keys and values of a map.

- Return value

  A value of the INT type is returned.

- Examples
  - Example 1: Calculate the number of elements in `array('a','b')`. Sample statement:

```
-- 2 is returned.
select size(array('a','b'));
```

- Example 2: Calculate the number of key-value pairs in `map('a',123,'b',456)`.

```
-- 2 is returned.
select size(map('a',123,'b',456));
```

SLICE

- Syntax

```
array<T> slice(array<T> <a>, <start>, <length>)
```

- Description

  Copies the elements in an array from the start position based on a specific length and returns the elements as a new array.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - start: required. This parameter specifies the position at which the function starts to copy elements from left to right. The minimum positive value of this parameter is 1. You can also set the start parameter to a negative value. In this case, the start position is counted from the end of the array, but the elements are still copied from left to right.
  - length: required. The number of elements in the returned array. The value must be greater than or equal to 0. If the value is greater than the length of the input array, this function returns a new array that consists of the elements from the start position to the end of the input array.

- Return value

  A value of the ARRAY type is returned.

- Examples
  - Example 1: Copy the elements in `array(10, 20, 20, null, null, 30)` from position `1` based on a length of `3`. Sample statement:

```
-- [10, 20, 20] is returned.
select slice(array(10, 20, 20, null, null, 30), 1, 3);
```

  - Example 2: Copy the elements in `array(10, 20, 20, null, null, 30)` from position `-2` based on a length of `2`. Sample statement:

```
-- [null, 30] is returned.
select slice(array(10, 20, 20, null, null, 30), -2, 2);
```

  - Example 3: Copy the elements in `array(10, 20, 20, null, null, 30)` from position `3` based on a length of `10`. Sample statement:

```
-- [20, null, null, 30] is returned.
select slice(array(10, 20, 20, null, null, 30), 3, 10);
```

  - Example 4: Copy the elements in `array(10, 20, 20, null, null, 30)` from position `3` based on a length of `0`. Sample statement:

```
-- [] is returned.
select slice(array(10, 20, 20, null, null, 30), 3, 0);
```

SORT_ARRAY

- Syntax

```
array<T> sort_array(array<T> <a>[, <isasc>])
```

- Description

  Sorts the elements in an array.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.
  - isasc: optional. This parameter specifies whether to sort elements in the array in ascending or descending order. Valid values: true and false. The value true indicates the ascending order and false indicates the descending order. If you do not specify this parameter, the elements are sorted in ascending order.

- Return value

  A value of the ARRAY type is returned. NULL is interpreted as the minimum value.

- Examples
  - Example 1: The `t_array` table contains the `c1 ARRAY<STRING>, c2 ARRAY<INT>, and c3 ARRAY<STRING>` fields. Data in the table:

    ```
    +------------+---------+--------------+
    | c1         | c2      | c3           |
    +------------+---------+--------------+
     [a, c, f, b]    [4, 5, 7, 2, 5, 8]    [You, Me, Him]
    +------------+---------+--------------+
    ```

    Sort data in each column of the table. Sample statement:

    ```
    [a, b, c, f] [2, 4, 5, 5, 7, 8] [Him, You, Me] is returned.
    select sort_array(c1),sort_array(c2),sort_array(c3) from t_array;
    ```

  - Example 2: Sort the elements in `array(10, 20, 40, 30, 30, null, 50)` in descending order. Sample statement:

    ```
    -- [50, 40, 30, 30, 20, 10, null] is returned.
    select sort_array(array(10, 20, 40, 30, 30, null, 50), false);
    ```

TRANSFORM

- Syntax

```
array<R> transform(array<T> <a>, function<T, R> <func>)
```

- Description

  Transforms the elements in Array a by using func and returns a new array.

- Parameters
  - a: required. This parameter specifies an array that `T` in `array<T>` specifies the data type of the elements in the array. The elements can be of any data type.

- func: required. This parameter specifies the built-in function, user-defined function, or expression that is used to transform the elements in Array a. The value must be of the same data type as the elements in Array a. `R` specifies the data type of the output results.

- Return value

  A value of the ARRAY type is returned.

- Examples

```
-- [2, 3, 4] is returned.
select transform(array(1, 2, 3), x -> x + 1);
```

ZIP_WITH

- Syntax

```
array<R> array_sort(array<T> <a>, array<S> <b>, function<T, S, R> <combiner>)
```

- Description

  Merges Array a and Array b at the element level based on element positions and a combiner and returns a new array.

- Parameters
  - a and b: required. These parameters specify arrays that `T` in `array<T>` and `S` in `array<S>` specify the data types of the elements in the arrays. The elements can be of any data type.
  - combiner: required. This parameter specifies the built-in function, user-defined function, or expression that is used to merge Array a and Array b at the element level. The combiner consists of two input parameters. One input parameter must be of the same data type as the elements in Array a. The other input parameter must be of the same data type as the elements in Array b.

- Return value

  A value of the ARRAY type is returned. The return value varies based on the following rules:
  - The elements in the returned array are at the same positions as the elements in Array a and Array b.
  - If Array a and Array b have different lengths, the null elements are added to the shorter array before the arrays are merged.

- Examples

```
-- [2, 4, 6, null] is returned.
select zip_with(array(1,2,3), array(1,2,3,4), (x,y) -> x + y);
```

MAP

- Syntax

```
map(K, V) map(K <key1>, V <value1>, K <key2>, V <value2>[, ...])
```

- Description

  Creates a map based on given key-value pairs.

- Parameters
  - key: required. All keys must be of the same data type after implicit conversions. Only basic data

types are supported.

- value: required. All values must be of the same data type after implicit conversions. All data types are supported.

- Return value

  Data of the MAP type is returned.

  > **Note** You can specify the `odps.sql.map.key.dedup.policy` parameter at the session level to configure the method that is used to process duplicate keys. Valid values:
  >
  > - exception: An error is returned.
  > - last_win: The latter key overwrites the former key.
  >
  > If you do not specify the parameter, the default value last_win is used.

- Examples
  - Example 1: No duplicate keys exist. The `t_table` table contains the `c1 (BIGINT), c2 (STRING), c3 (STRING), c4 (BIGINT), and c5 (BIGINT)` fields. Data in the table:

    ```
    +------------+----+----+------------+------------+
    | c1         | c2 | c3 | c4         | c5         |
    +------------+----+----+------------+------------+
    | 1000       | k11 | k21 | 86         | 15         |
    | 1001       | k12 | k22 | 97         | 2          |
    | 1002       | k13 | k23 | 99         | 1          |
    +------------+----+----+------------+------------+
    ```

    Sample statement:

    ```
    -- Define a map based on the key-value pairs between the c2 and c4 columns, and between the c3 and c5 columns.
    select map(c2,c4,c3,c5) from t_table;
    -- The following result is returned:
    +------+
    | _c0  |
    +------+
    | {k11:86, k21:15} |
    | {k12:97, k22:2} |
    | {k13:99, k23:1} |
    +------+
    ```

- Example 2: Duplicate keys exist. The `t_table` table contains the `c1 (BIGINT), c2 (STRING), c3 (STRING), c4 (BIGINT), and c5 (BIGINT)` fields. Data in the table:

```
1000,'k11','k11',86,15
1001,'k12','k22',97,2
1002,'k13','k23',99,1
1003,'k13','k24',100,1
1004,'k12','k25',95,1
```

Sample statement:

```
-- Define a map based on the key-value pairs between the c2 and c4 columns, and between the c3 and c5 columns.
select map(c2,c4,c3,c5) from t_table;
-- The following result is returned:
+------+
| _c0  |
+------+
| {'k11':15} |
| {'k12':97, 'k22':2} |
| {'k13':99, 'k23':1} |
| {'k13':100, 'k24':1} |
| {'k12':95, 'k25':1} |
+------+
```

MAP_CONCAT

- Syntax

```
map<K, V> map_concat([string <mapDupKeyPolicy>,] map<K, V> <a>, map<K, V> <b>[,...])
```

- Description

  Calculates the union of multiple maps.

- Parameters
  - mapDupKeyPolicy: optional. A value of the STRING type. This parameter specifies the method that is used to process duplicate keys. Valid values:
    - exception: An error is returned.
    - last_win: The latter key overwrites the former key.

    You can also specify the `odps.sql.map.key.dedup.policy` parameter at the session level to configure the method that is used to process duplicate keys. For example, you can set `odps.sql.map.key.dedup.policy` to exception. If you do not specify this parameter, the default value last_win is used.

    > **Note** The behavior implementation of MaxCompute is determined based on mapDupKeyPolicy. If you do not specify mapDupKeyPolicy, the value of `odps.sql.map.key.dedup.policy` prevails.

  - a and b: required. These parameters specify maps. The keys of the maps must be of the same data type, and the values of the maps must be of the same data type. `K` and `V` in `map<K, V>` specify the keys and values of a map.

- Return value

  Data of the MAP type is returned. An error is returned in the following scenarios:

  - A map is null or the key of a map is null.
  - The data types of multiple maps are different.

- Examples

```
-- {1:a, 2:b, 3:c} is returned.
select map_concat(map(1, 'a', 2, 'b'), map(3, 'c'));
-- {1:a, 2:d, 3:c} is returned.
select map_concat('last_win', map(1, 'a', 2, 'b'), map(3, 'c'), map(2, 'd'));
```

MAP_ENTRIES

- Syntax

```
array<struct<K, V>> map_entries(map<K, V> <a>):
```

- Description

  Converts key-value pairs in Map a into a struct array.

- Parameters

  a: required. This parameter specifies a map. `K` and `V` in `map<K, V>` specify the keys and values of a map.

- Return value

  A struct array is returned. If the input is null, NULL is returned.

- Examples

```
-- [{key:1, value:a}, {key:2, value:b}] is returned.
select map_entries(map(1,  'a',  2,  'b'));
```

MAP_FILTER

- Syntax

```
map<K, V> map_filter(map<K, V> <input>, function <K, V, boolean> <predicate>)
```

- Description

  Filters the elements in Map input and retains only the elements that meet the predicate condition.

- Parameters
  - input: required. A value of the MAP type. `K` and `V` in `map<K, V>` specify the keys and values of a map.
  - predicate: required. This parameter specifies the built-in function, user-defined function, or expression that is used to filter the elements in the map. The predicate condition consists of two input parameters that correspond to the keys and values in input. The output result is of the BOOLEAN type.

- Return value

  Data of the MAP type is returned.

- Examples

```
-- {-30:100, 20:50} is returned.
select map_filter(map(10, -20, 20, 50, -30, 100, 21, null), (k, v) -> (k+v) > 10);
```

MAP_FROM_ARRAYS

- Syntax

```
map<K, V> map_from_arrays([string <mapDupKeyPolicy>,] array<K> <a>, array<V> <b>))
```

- Description

  Creates a map based on Array a and Array b.

- Parameters
  - mapDupKeyPolicy: optional. A value of the STRING type. This parameter specifies the method that is used to process duplicate keys. Valid values:
    - exception: An error is returned.
    - last_win: The latter key overwrites the former key.

    You can also specify the `odps.sql.map.key.dedup.policy` parameter at the session level to configure the method that is used to process duplicate keys. For example, you can set `odps.sql.map.key.dedup.policy` to exception. If you do not specify this parameter, the default value last_win is used.

    > **Note** The behavior implementation of MaxCompute is determined based on mapDupKeyPolicy. If you do not specify mapDupKeyPolicy, the value of `odps.sql.map.key.dedup.policy` prevails.

  - a: required. This parameter specifies an array. This parameter corresponds to the key in the generated map. `K` in `array<K>` specifies the data type of the elements in the array. The elements can be of any data type.
  - b: required. This parameter specifies an array. This parameter corresponds to the value in the generated map. `V` in `array<V>` specifies the data type of the elements in the array. The elements can be of any data type.

- Return value

  Data of the MAP type is returned. The return value varies based on the following rules:
  - If Array a or Array b is null, NULL is returned.
  - If Array a contains a null element or the two arrays are of different lengths, an error is returned.

- Examples

```
-- {1:2, 3:4} is returned.
select map_from_arrays(array(1.0, 3.0), array('2', '4'));
-- {1:2, 3:6} is returned.
select map_from_arrays('last_win', array(1.0, 3.0, 3), array('2', '4', '6'));
```

MAP_FROM_ENTRIES

- Syntax

```
map<K, V> map_from_entries([string <mapDupKeyPolicy>,] array <struct<K, V> , struct<K, V>
[,...]>)
```

- Description

  Creates a map based on given struct arrays.

- Parameters
  - mapDupKeyPolicy: optional. A value of the STRING type. This parameter specifies the method that is used to process duplicate keys. Valid values:
    - exception: An error is returned.
    - last_win: The latter key overwrites the former key.

    You can also specify the `odps.sql.map.key.dedup.policy` parameter at the session level to configure the method that is used to process duplicate keys. For example, you can set `odps.sql.map.key.dedup.policy` to exception. If you do not specify this parameter, the default value last_win is used.

    > **Note** The behavior implementation of MaxCompute is determined based on mapDupKeyPolicy. If you do not specify mapDupKeyPolicy, the value of `odps.sql.map.key.dedup.policy` prevails.

  - Data of the STRUCT type is required. K corresponds to the keys in the generated map. V corresponds to the values in the generated map. `K` and `V` in `struct<K, V>` specify the keys and values of a struct array.

- Return value

  Data of the MAP type is returned. The return value varies based on the following rules:
  - If a struct array is null, NULL is returned.
  - If the number of fields in a struct array is not 2 or the key of a struct array is null, an error is returned.

- Examples

```
-- {1:a, 2:b} is returned.
select map_from_entries(array(struct(1, 'a'), struct(2, 'b')));
-- {1:a, 2:c} is returned.
select map_from_entries(array(struct(1, 'a'), struct(2, 'b'), struct(2, 'c')));
```

MAP_KEYS

- Syntax

```
array<K> map_keys(map<K, V> <a>)
```

- Description

  Returns all keys in Map a as an array.

- Parameters

  a: required. This parameter specifies a map. `K` and `V` in `map<K, V>` specify the keys and values of a map.

- Return value

  A value of the ARRAY type is returned. If the input map is null, NULL is returned.

- Examples

  The `t_table_map` table contains the `c1 (BIGINT) and t_map (MAP<STRING,BIGINT>)` fields. Data in the table:

```
+------------+-------+
| c1         | t_map |
+------------+-------+
| 1000       | {k11:86, k21:15} |
| 1001       | {k12:97, k22:2} |
| 1002       | {k13:99, k23:1} |
+------------+-------+
```

  Sample statement:

```
-- Return keys in the t_map column as an array.
select c1, map_keys(t_map) from t_table_map;
-- The following result is returned:
+------------+------+
| c1         | _c1  |
+------------+------+
| 1000       | [k11, k21] |
| 1001       | [k12, k22] |
| 1002       | [k13, k23] |
+------------+------+
```

MAP_VALUES

- Syntax

```
array<V> map_values(map<K, V> <a>)
```

- Description

  Returns all values in Map a as an array.

- Parameters

  a: required. This parameter specifies a map. `K` and `V` in `map<K, V>` specify the keys and values of a map.

- Return value

  A value of the ARRAY type is returned. If the input map is null, NULL is returned.

- Examples

  The `t_table_map` table contains the `c1 (BIGINT) and t_map (MAP<STRING,BIGINT>)` fields. Data in the table:

```
+------------+-------+
| c1         | t_map |
+------------+-------+
| 1000       | {k11:86, k21:15} |
| 1001       | {k12:97, k22:2} |
| 1002       | {k13:99, k23:1} |
+------------+-------+
```

Sample statement:

```
-- Return keys in the t_map column as an array.
select c1,map_values(t_map) from t_table_map;
-- The following result is returned:
+------------+------+
| c1         | _c1  |
+------------+------+
| 1000       | [86, 15] |
| 1001       | [97, 2] |
| 1002       | [99, 1] |
+------------+------+
```

MAP_ZIP_WITH

- Syntax

```
<K, V1, V2, V3> map<K, V3> map_zip_wuth(map<K, V1> <input1>, map<K, V2> <input2>, functio
n<K, V1, V2, V3> <func>)
```

- Description

  Merges Map input1 and Map input2 into a single map. The keys of the new map are the union of the keys of the two input maps. The value of each key of the new map is calculated by using func.

- Parameters
  - input1 and input2: required. These parameters specify maps. `K` and `V` in `map<K, V>` specify the keys and values of a map.
  - func: required. func consists of three input parameters, which correspond to a key, the value that corresponds to the key in input1, and the value that corresponds to the key in input2. If the key does not exist in input1 or input2, null is used to replace the value that corresponds to the key in func.

- Return value

  Data of the data type defined by func is returned.

- Examples

```
-- {1:[1, 1, 4], 2:[2, 2, 5], 3:[3, null, null], 4:[4, null, 7]} is returned.
select map_zip_with(map(1, 1, 2, 2, 3, null), map(1, 4, 2, 5, 4, 7), (k, v1, v2) -> array
(k, v1, v2));
```

TRANSFORM_KEYS

- Syntax

```
map<K2, V> transform_keys([string <mapDupKeyPolicy>,] map<K1, V> <input>, function<K1, V, K2> <func>)
```

- Description

  Transforms the keys in Map input by using func. The values in the map are not changed.

- Parameters
  - mapDupKeyPolicy: optional. A value of the STRING type. This parameter specifies the method that is used to process duplicate keys. Valid values:
    - exception: An error is returned.
    - last_win: The latter key overwrites the former key.

    You can also specify the `odps.sql.map.key.dedup.policy` parameter at the session level to configure the method that is used to process duplicate keys. For example, you can set `odps.sql.map.key.dedup.policy` to exception. If you do not specify this parameter, the default value last_win is used.

    > **Note** The behavior implementation of MaxCompute is determined based on mapDupKeyPolicy. If you do not specify mapDupKeyPolicy, the value of `odps.sql.map.key.dedup.policy` prevails.

  - input: required. This parameter specifies a map. `K1` and `V` in `map<K1, V>` specify the keys and values of a map.
  - func: required. This parameter specifies the built-in function, user-defined function, or expression that is used to transform the keys. The function or expression consists of two input parameters that correspond to the keys and values in input. `K2` specifies the data type of keys in the returned map.

- Return value

  Data of the MAP type is returned. If one of the new keys is null, an error is returned.

- Examples

```
-- {-10:-20, 70:50, 71:101} is returned.
select transform_keys(map(10, -20, 20, 50, -30, 101), (k, v) -> k + v);
-- No error is returned. The returned result depends on the order of the elements in the input map.
select transform_keys("last_win", map(10, -20, 20, 50, -30, 100), (k, v) -> k + v);
-- An error is returned because duplicate keys exist.
select transform_keys("exception", map(10, -20, 20, 50, -30, 100), (k, v) -> k + v);
```

TRANSFORM_VALUES

- Syntax

```
map<K, V2> transform_values(map<K, V1> <input>, function<K, V1, V2> <func>)
```

- Description

  Transforms the values in Map input by using func. The keys in the map are not changed.

- Parameters

- input: required. This parameter specifies a map. `K` and `V1` in `map<K, V1>` specify the keys and values of the map.
- func: required. This parameter specifies the built-in function, user-defined function, or expression that is used to transform the keys. The function or expression consists of two input parameters that correspond to the keys and values in input. `V2` specifies the data type of values in the returned map.

- Return value

  Data of the MAP type is returned.

- Examples

```
-- {-30:71, 10:-10, 20:null} is returned.
select transform_values(map(10, -20, 20, null, -30, 101), (k, v) -> k + v);
```

FIELD

- Syntax

```
T field(struct <s>, string <fieldName>)
```

- Description

  Obtains the value of a member variable in a struct.

- Parameters
  - s: required. This parameter specifies a struct. The struct is in the format of `{f1:T1, f2:T2[, ...]}`. `f1` and `f2` specify member variables, `T1` specifies the value of `f1`, and `T2` specifies the value of `f2`.
  - fieldName: required. A value of the STRING type. The value must be one of the member variables in the struct.

- Return value

  The value of the specific member variable in the struct is returned.

- Examples

```
-- hello is returned.
select field(named_struct('f1', 'hello', 'f2', 3), 'f1');
```

INLINE

- Syntax

```
inline(array<struct<f1:T1, f2:T2[, ...]>>)
```

- Description

  Expands a given struct array. Each array element corresponds to a row and each struct element corresponds to a column in each row.

- Parameters

  f1:T1 and f2:T2: required. A value of any data type. `f1` and `f2` specify member variables, `T1` specifies the value of `f1`, and `T2` specifies the value of `f2`.

- Return value

  The expanded data of the struct array is returned.

- Examples

  The `t_table` table contains the `t_struct (STRUCT<user_id:BIGINT,user_name:STRING,married:STRING,weight:DOUBLE>)` field. Data in the table:

```
+-----------+
| t_struct |
+-----------+
| {user_id:10001, user_name:LiLei, married:N, weight:63.5} |
| {user_id:10002, user_name:HanMeiMei, married:Y, weight:43.5} |
+-----------+
```

  Sample statement:

```
-- Expand the t_struct column.
select inline(array(t_struct)) from t_table;
-- The following result is returned:
+------------+-----------+---------+------------+
| user_id    | user_name | married | weight     |
+------------+-----------+---------+------------+
| 10001      | LiLei     | N       | 63.5       |
| 10002      | HanMeiMei | Y       | 43.5       |
+------------+-----------+---------+------------+
```

STRUCT

- Syntax

```
struct struct(<value1>,<value2>[, ...])
```

- Description

  Creates a struct based on a given value list.

- Parameters

  value: required. Data in the array can be of any data type.

- Return value

  A value of the STRUCT type is returned. Column names are sequentially named as `col1, col2, ...`

- Examples

```
-- {col1:a, col2:123, col3:true, col4:56.9} is returned.
select struct('a',123,'true',56.90);
```

NAMED_STRUCT

- Syntax

```
struct named_struct(string <name1>, T1 <value1>, string <name2>, T2 <value2>[, ...])
```

- Description

  Creates a struct based on given name-value pairs.

- Parameters
  - value: required. All data types are supported.
  - name: required. The column name of the STRING type. This parameter is a constant.
- Return value

  A value of the STRUCT type is returned. Column names are sequentially named as `name1, name2, ...`.
- Examples

```
-- {user_id:10001, user_name:LiLei, married:F, weight:63.5} is returned.
select named_struct('user_id',10001,'user_name','LiLei','married','F','weight',63.50);
```

FROM_JSON

- Syntax

```
from_json(<jsonStr>, <schema>)
```

- Description

  Returns data of the ARRAY, MAP, or STRUCT type based on JSON string jsonStr and output format schema.
- Parameters
  - jsonStr: required. The JSON string that you entered.

- schema: required. The schema of the JSON string. The value of this parameter must be in the same format as that in the statement for creating a table, such as `array<bigint>`, `map<string, array<string>>`, or `struct<a:int, b:double, `C`:map<string,string>>`.

  > **Note** Keys in a struct are case-sensitive. You can also specify a struct in the format of `a BIGINT, b DOUBLE`, which is equivalent to `STRUCT<a:BIGINT, b:DOUBLE>`.

  The following table describes the mappings between JSON data types and MaxCompute data types.

JSON data type	MaxCompute data type
OBJECT	STRUCT, MAP, and STRING
ARRAY	ARRAY and STRING
NUMBER	TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, and STRING
BOOLEAN	BOOLEAN and STRING
STRING	STRING, CHAR, VARCHAR, BINARY, DATE, and DATETIME
NULL	All types

  > **Note** The JSON string of the OBJECT and ARRAY types are parsed as much as possible. If the data type of the JSON string is not mapped to any MaxCompute data type, the JSON string is omitted. For ease of use, all JSON data types can be converted into the STRING data type supported by MaxCompute. When you convert a JSON string of the NUMBER type to a value of the FLOAT, DOUBLE, or DECIMAL type, the precision of the value cannot be ensured. We recommend you convert the JSON string to a value of the STRING type and then convert the obtained value to a value of the FLOAT, DOUBLE, or DECIMAL type.

- Return value

  A value of the ARRAY, MAP, or STRUCT type is returned.

- Examples
  - Example 1: Convert a specific JSON string into a value of a specific data type. Sample statements:

```
-- {"a":1,"b":0.8} is returned.
select from_json('{"a":1, "b":0.8}', 'a int, b double');
-- {"time":"26/08/2015"} is returned.
select from_json('{"time":"26/08/2015"}', 'time string');
-- {"a":1,"b":0.8} is returned.
select from_json('{"a":1, "b":0.8}', 'a int, b double, c string');
-- [1,2,3] is returned.
select from_json('[1, 2, 3, "a"]', 'array<bigint>');
-- {"d":"v","a":"1","b":"[1,2,3]","c":"{}"} is returned.
select from_json('{"a":1,"b":[1,2,3],"c":{},"d":"v"}', 'map<string, string>');
```

- Example 2: Use the `map_keys` and `from_json` functions to obtain all keys in the JSON string. You can also use JSON_KEYS for the same purpose. Sample statement:

```
-- ["a","b"] is returned.
select map_keys(from_json('{"a":1,"b":2}','map<string,string>'));
```

GET_JSON_OBJECT

- Syntax

```
string get_json_object(string <json>, string <path>)
```

- Description

  Extracts a single string from a standard JSON string based on path. The original data is read each time this function is called. Therefore, repeated calls may affect system performance and increase costs. To prevent repeated calls, you can use the `GET_JSON_OBJECT` function with UDTFs. For more information, see Convert JSON log data by using MaxCompute built-in functions and UDTFs.

- Parameters
  - json: required. A value of the STRING type. This parameter specifies a standard JSON object in the format of `{Key:Value, Key:Value,...}`. If the string contains a double quotation mark ("), use two backslashes (\\) to escape the double quotation mark (") before extraction. If the string contains a single quotation mark ('), use a single backslash (\) to escape the single quotation mark (') before extraction.
  - path: required. A value of the STRING type. This parameter specifies the path in the value of the json parameter and starts with `$`. For more information about the path parameter, see LanguageManual UDF. For more information about best practices, see Migrate JSON data from OSS to MaxCompute. Different characters have the following meanings:
    - `$`: indicates the root node.
    - `.` or `['']`: indicates a child node. MaxCompute parses JSON objects by using `.` or `['']`. If a key in a JSON object contains a period (.), ['] can be used.
    - `[]` (`[number]`): indicates an array subscript, which starts from 0.
    - `*`: indicates the wildcard for `[]`, which returns an entire array. An asterisk (`*`) cannot be escaped.

- Return value
  - If the json parameter is empty or invalid, NULL is returned.
  - If the format of json is valid and path exists, the related string is returned.
  - To determine how this function returns a value, you can specify `odps.sql.udf.getjsonobj.new` for a session.

- If you run the `set odps.sql.udf.getjsonobj.new=true;` command, this function retains the original strings when it returns a value.

  We recommend that you use this configuration because it results in more standard function return behavior. This facilitates data processing and improves data processing performance. **If a MaxCompute project has jobs in which this function escapes JSON reserved characters, we recommend that you retain the original escaping operation to prevent errors caused by lack of verification.** The function complies with the following rules when it returns a value:

  - The return value is still a JSON string, which can be parsed as JSON data. You do not need to use the `REPLACE` or `REGEXP_REPLACE` function to replace backslashes (\).
  - Duplicate keys are allowed in a JSON object. If duplicate keys exist, the data can be parsed.

    ```
    -- 1 is returned.
    select get_json_object('{"a":"1","a":"2"}', '$.a');
    ```

  - The encoded strings that correspond to emojis are supported. However, DataWorks does not allow you to enter emojis. DataWorks allows you to enter only the encoded strings that correspond to emojis to MaxCompute by using a tool, such as Data Integration. DataWorks uses the `GET_JSON_OBJECT` function to process the data.

    ```
    -- An emoji is returned.
    select get_json_object('{"a":"<Emoji>"}', '$.a');
    ```

  - The output results are displayed in alphabetical order.

    ```
    -- {"b":"1","a":"2"} is returned.
    select get_json_object('{"b":"1","a":"2"}', '$');
    ```

- If you run the `set odps.sql.udf.getjsonobj.new=false;` command, this function escapes JSON reserved characters when it returns a value. The function complies with the following rules when it returns a value:

  - JSON reserved characters, such as line feeds (\n) and double quotation marks (") are displayed as `'\n'` and `'\"'`.
  - Each key in a JSON object must be unique. If duplicate keys exist, the data may fail to be parsed.

    ```
    -- NULL is returned.
    select get_json_object('{"a":"1","a":"2"}', '$.a');
    ```

  - The encoded strings that correspond to emojis cannot be parsed.

    ```
    -- NULL is returned.
    select get_json_object('{"a":"<Emoji>"}', '$.a');
    ```

  - The output results are displayed in alphabetical order.

    ```
    -- {"a":"2","b":"1"} is returned.
    select get_json_object('{"b":"1","a":"2"}', '$');
    ```

> **Note** For MaxCompute projects that are created on or after January 21, 2021, the `GET_JSON_OBJECT` function retains the original strings when it returns a value. For MaxCompute projects that are created before January 21, 2021, the `GET_JSON_OBJECT` function escapes JSON reserved characters when it returns a value. The following example helps you determine how the `GET_JSON_OBJECT` function returns a value in a MaxCompute project.
>
> ```
> select get_json_object('{"a":"[\\"1\\"]"}', '$.a');
> -- Return JSON reserved characters by using escape characters.
> [\"1\"]
> -- Return the original strings.
> ["1"]
> ```
>
> You can submit a ticket to request the MaxCompute technical support team to enable the `GET_JSON_OBJECT` function. This way, you do not need to frequently specify flagodps.sql.udf.getjsonobj.new for a session. This function retains the original strings when it returns a value.

- Examples
  - Example 1: Extract information from the JSON object `src_json.json`. Sample statement:

```
-- The JSON string src_json.json contains the following content:
+----+
json
+----+
{"store":
{"fruit":[{"weight":8,"type":"apple"},{"weight":9,"type":"pear"}],
"bicycle":{"price":19.95,"color":"red"}
},
"email":"amy@only_for_json_udf_test.net",
"owner":"amy"
}
-- Extract the information of the owner field and return amy.
select get_json_object(src_json.json, '$.owner') from src_json;
-- Extract the information of the first array in the store.fruit field and return {"wei
ght":8,"type":"apple"}.
select get_json_object(src_json.json, '$.store.fruit[0]') from src_json;
-- Extract the information of the non-existent field and return NULL.
select get_json_object(src_json.json, '$.non_exist_key') from src_json;
```

- Example 2: Extract information from a JSON object of the ARRAY type. Sample statement:

```
-- 2222 is returned.
select get_json_object('{"array":[["aaaa",1111],["bbbb",2222],["cccc",3333]]}','$.array
[1][1]');
-- ["h0","h1","h2"] is returned.
set odps.sql.udf.getjsonobj.new=true;
select get_json_object('{"aaa":"bbb","ccc":{"ddd":"eee","fff":"ggg","hhh":["h0","h1","h
2"]},"iii":"jjj"}','$.ccc.hhh[*]');
-- ["h0","h1","h2"] is returned.
set odps.sql.udf.getjsonobj.new=false;
select get_json_object('{"aaa":"bbb","ccc":{"ddd":"eee","fff":"ggg","hhh":["h0","h1","h
2"]},"iii":"jjj"}','$.ccc.hhh');
-- h1 is returned.
select get_json_object('{"aaa":"bbb","ccc":{"ddd":"eee","fff":"ggg","hhh":["h0","h1","h
2"]},"iii":"jjj"}','$.ccc.hhh[1]');
```

- Example 3: Extract information from a JSON object that contains a period `(.)`. Sample statement:

```
-- Create a table.
create table mf_json (id string, json string);
-- Insert data into the table. The key in the data contains a period (.).
insert into table mf_json (id, json) values ("1", "{
\"China.beijing\":{\"school\":{\"id\":0,\"book\":[{\"title\": \"A\",
\"price\": 8.95},{\"title\": \"B\",\"price\": 10.2}]}}}");
-- Insert data into the table. The key in the data does not contain a period (.).
insert into table mf_json (id, json) values ("2", "{
\"China_beijing\":{\"school\":{\"id\":0,\"book\":[{\"title\": \"A\",
\"price\": 8.95},{\"title\": \"B\",\"price\": 10.2}]}}}");
-- Query the value of id in the JSON object whose key is China.beijing. 0 is returned.
Only [''] can be used to specify the key because the key contains a period (.). This wa
y, MaxCompute can parse the key.
select get_json_object(json, "$['China.beijing'].school['id']") from mf_json where id =
1;
-- -- Query the value of id in the JSON object whose key is China_beijing. 0 is returne
d. You can use one of the following statements:
select get_json_object(json, "$['China_beijing'].school['id']") from mf_json where id =
2;
select get_json_object(json, "$.China_beijing.school['id']") from mf_json where id =2;
```

- Example 4: The json parameter is empty or invalid. Sample statement:

```
-- NULL is returned.
select get_json_object('','$.array[1][1]');
-- NULL is returned.
select get_json_object('"array":["aaaa",1111],"bbbb":["cccc",3333]','$.array[1][1]');
```

- Example 5: Escape a JSON string. Sample statement:

```
set odps.sql.udf.getjsonobj.new=true;
-- "1" is returned.
select get_json_object('{"a":"\\"1\\"","b":"2"}', '$.a');
-- '1' is returned.
select get_json_object('{"a":"\'1\'","b":"2"}', '$.a');
```

JSON_TUPLE

- Syntax

```
string json_tuple(string <json>, string <key1>, string <key2>,...)
```

- Description

  Extracts strings from a standard JSON string based on a set of input keys, such as `(key1,key2,...)`.

- Parameters
  - json: required. A value of the STRING type. This parameter specifies a standard JSON string.
  - key: required. A value of the STRING type. This parameter is used to describe the `path` of a JSON object in the JSON string. The value cannot start with a dollar sign ($). You can enter multiple keys at a time. MaxCompute parses JSON objects by using `.` or `['']`. If a key in a JSON object contains a period (.), ['] can be used.

- Return value

  A value of the STRING type is returned.

  > **Note**
  > - If json is empty or invalid, NULL is returned.
  > - If key is empty, invalid, or does not exist in the JSON string, NULL is returned.
  > - If json is valid and key exists, the related string is returned.
  > - This function can parse JSON data that contains Chinese characters.
  > - This function can parse nested JSON data.
  > - This function can parse JSON data that contains nested arrays.
  > - The parsing action is equivalent to the execution of GET_JSON_OBJECT along with `set od ps.sql.udf.getjsonobj.new=true;`. To obtain multiple objects from a JSON string, you must call the GET_JSON_OBJECT function multiple times. As a result, the JSON string is parsed multiple times. The JSON_TUPLE function allows you to enter multiple keys at a time and the JSON string is parsed only once. JSON_TUPLE is more efficient than GET_JSON_OBJECT.
  > - JSON_TUPLE is a UDTF and is used with Lateral View when other columns need to be selected.

TO_JSON

- Syntax

```
to_json(<expr>)
```

- Description

  Converts an expression of a given complex data type into a JSON string.

- Parameters

  expr: required. The expression of the ARRAY, MAP, or STRUCT type.

> **Note** If the input expression is of the STRUCT type (`struct<key1:value1, key2:value2>`), take note of the following points:
>
> - All keys are converted into lowercase letters when you convert the expression into a JSON string.
> - If a `value` is null, the key-value pair to which the `value` belongs is not included in the JSON string that is returned. For example, if `value2` is null, `key2:value2` is not included in the JSON string that is returned.

- Examples
  - Example 1: Convert an expression of a given data type into a JSON string. Sample statement:

```
-- {"a":1,"b":2} is returned.
select to_json(named_struct('a', 1, 'b', 2));
-- {"time":"26/08/2015"} is returned.
select to_json(named_struct('time', "26/08/2015"));
-- [{"a":1,"b":2}] is returned.
select to_json(array(named_struct('a', 1, 'b', 2)));
-- {"a":{"b":1}} is returned.
select to_json(map('a', named_struct('b', 1)));
-- {"a":1} is returned.
select to_json(map('a', 1));
-- [{"a":1}] is returned.
select to_json(array((map('a', 1))));
```

  - Example 2: The input expression is of the STRUCT type. Sample statement:

```
-- {"a":"B"} is returned. If the expression of the STRUCT type is converted into a JSON string, all keys are converted into lowercase letters.
select to_json(named_struct("A", "B"));
-- {"k2":"v2"} is returned. The key-value pair to which null belongs is not included in the JSON string that is returned.
select to_json(named_struct("k1", cast(null as string), "k2", "v2"));
```

3.9.9. Other functions

MaxCompute SQL provides other functions that are commonly used in the development process. You can use these functions based on your business requirements. This topic describes the command syntax and parameters of these functions, such as CAST, DECODE, LEAST, and SPLIT. This topic also provides examples on how to use these functions.

Function	Description
BASE64	Converts a binary value into a Base64-encoded string.
BETWEEN AND expression	Returns the values that fall in or fall out of the specified range.
CASE WHEN expression	Returns values based on the computing result of an expression.

Function	Description
CAST	Converts the result of an expression into the specified data type.
COALESCE	Returns the first non-null value in the parameter list.
COMPRESS	Uses the GZIP algorithm to compress input parameters of the STRING or BINARY type.
CRC32	Calculates the cyclic redundancy check (CRC) value of a value that is of the STRING or BINARY type.
DECODE	Implements the `IF-THEN-ELSE` logic.
DECOMPRESS	Uses the GZIP algorithm to decompress input parameters of the BINARY type.
GET_IDCARD_AGE	Returns an age in years based on the ID card number.
GET_IDCARD_BIRTHDAY	Returns the date of birth based on the ID card number.
GET_IDCARD_SEX	Returns the gender based on the ID card number.
GET_USER_ID	Obtains the ID of the current account.
GREATEST	Returns the maximum value of the input parameters.
HASH	Calculates a hash value based on the input parameters.
IF	Checks whether a specified condition is true.
LEAST	Returns the minimum value of the input parameters.
MAX_PT	Returns the name of the largest level-1 partition in a partitioned table.
NULLIF	Checks whether the values of two input parameters are the same.
NVL	Specifies the return values of the parameters whose values are null.
ORDINAL	Sorts the values of the input variables in ascending order and returns the value that is ranked at a specified position.
PARTITION_EXISTS	Checks whether a specified partition exists in a table.
SAMPLE	Samples all column values that are read and filters out the rows that do not meet sampling conditions.
SHA	Calculates the SHA-1 hash value of a value that is of the STRING or BINARY type.
SHA1	Calculates the SHA-1 hash value of a value that is of the STRING or BINARY type.
SHA2	Calculates the SHA-2 hash value of a value that is of the STRING or BINARY type.

Function	Description
SIGN	Determines the sign of a value. The sign indicates whether a value is positive or negative.
SPLIT	Splits a string with a specified delimiter and returns an array.
STACK	Splits a specified parameter group into a specified number of rows.
STR_TO_MAP	Splits a string with a specified delimiter and returns a key-value pair.
TABLE_EXISTS	Checks whether a specified table exists.
TRANS_ARRAY	Transposes one row of data into multiple rows. This function is a user-defined table-valued function (UDTF) that transposes an array separated by fixed delimiters in a column into multiple rows.
TRANS_COLS	Transposes one row of data into multiple rows. This function is a UDTF that transposes columns into rows.
UNBASE64	Converts a Base64-encoded string into a binary value.
UNIQUE_ID	Returns a unique ID. This function is more efficient than the UUID function.
UUID	Returns a random ID.

BASE64

- Syntax

```
string base64(binary <value>)
```

- Description

  Converts a binary value into a Base64-encoded string.

- Parameters

  value: required. A value of the BINARY type, which is the value that you want to convert.

- Return value

  A value of the STRING type is returned. If the input parameter is set to null, null is returned.

- Examples
  - Example 1: Convert the binary result of `cast ('alibaba' as binary)` into a Base64-encoded string. Sample statement:

```
-- YWxpYmFiYQ== is returned.
select base64(cast ('alibaba' as binary));
```

  - Example 2: The input parameter is set to null. Sample statement:

```
-- The value null is returned.
select base64(null);
```

BETWEEN AND expression

- Syntax

```
<a> [NOT] between <b> and <c>
```

- Description

  Returns the values of field a. The values must fall in or fall out of the range between b and c.

- Parameters
  - a: required. The field whose values you want to obtain.
  - b and c: required. The two parameters specify a value range. The data types of the two parameters must be the same as the data type of the a parameter.

- Return value

  The values that fall in or fall out of the specified range are returned.

  If the a, b, or c parameter is empty, null is returned.

- Examples

  The `emp` table contains the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

  Query the data whose `sal` is from 1000 to 1500. Sample statement:

```
select * from emp where sal between 1000 and 1500;
```

  The following result is returned:

```
+-------+-------+-----+------------+------------+------------+------------+------------+
| empno | ename | job | mgr        | hiredate   | sal        | comm       | deptno     |
+-------+-------+-----+------------+------------+------------+------------+------------+
| 7521  | WARD  | SALESMAN | 7698  | 1981-02-22 00:00:00 | 1250.0     | 500.0      | 30
|
| 7654  | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.0     | 1400.0     | 30 |
| 7844  | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00 | 1500.0     | 0.0        | 30 |
| 7876  | ADAMS | CLERK | 7788  | 1987-05-23 00:00:00 | 1100.0     | NULL     | 20   |
| 7934  | MILLER | CLERK | 7782  | 1982-01-23 00:00:00 | 1300.0     | NULL      | 10  |
| 7956  | TEBAGE | CLERK | 7748  | 1982-12-30 00:00:00 | 1300.0     | NULL      | 10  |
+-------+-------+-----+------------+------------+------------+------------+------------+
```

CASE WHEN expression

- Syntax

  MaxCompute provides the following `CASE WHEN` syntax:

  ```
  case <value>
  when <value1> then <result1>
  when <value2> then <result2>
  ...
  else <resultn>
  end
  ```

  ```
  case
  when (<_condition1>) then <result1>
  when (<_condition2>) then <result2>
  when (<_condition3>) then <result3>
  ...
  else <resultn>
  end
  ```

- Description

  Returns the value of result based on the calculation result of value or _condition.

- Parameters
  - value: required. The value that is used for comparison.
  - _condition: required. The condition that is used for comparison.
  - result: required. The return value.

- Return value
  - If the data types of all result values are only BIGINT and DOUBLE, the values are returned after their data types are converted into the DOUBLE type.
  - If result values of the STRING type exist, the values are converted into the STRING type before they are returned. If a data type conversion is not supported, an error is returned. For example, data of the BOOLEAN type cannot be converted into the STRING type.
  - Conversions between other data types are not allowed.

- Examples

The `sale_detail` table contains the `shop_name (STRING), customer_id (STRING), and total_price (DOUBLE)` fields. This table contains the following data:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

Sample statement:

```
select
case
when region='china' then 'default_region'
when region like 'shang%' then 'sh_region'
end as region
from sale_detail;
```

The following result is returned:

```
+-------------+
| region      |
+-------------+
| default_region |
| default_region |
| default_region |
| sh_region   |
| sh_region   |
| sh_region   |
+-------------+
```

CAST

- Syntax

```
cast(<expr> as <type>)
```

- Description

  Converts the data type of the data source specified by expr into the data type specified by type.

- Parameters
  - expr: required. The data source whose data type you want to convert.

- type: required. The data type into which you want to convert the data. Usage:
  - `cast(double as bigint)`: converts a value of the DOUBLE type into the BIGINT type.
  - `cast(string as bigint)`: converts a value of the STRING type into the BIGINT type. If the string consists of numerals expressed in the INTEGER form, the string is converted into the BIGINT type. If the string consists of numerals expressed in the FLOAT or EXPONENTIAL form, the string is converted into the DOUBLE type and then into the BIGINT type.
  - The default date format, `yyyy-mm-dd hh:mi:ss`, is used for `cast(string as datetime)` and `cast(datetime as string)`.

- Return value

  A value of the specified data type is returned.

- Examples
  - Example 1: common usage. Sample statement:

    ```
    -- The value 1 is returned.
    select cast('1' as bigint);
    ```

  - Example 2: incorrect usage. If the type conversion fails or an unsupported type conversion occurs, an error is returned. **Incorrect sample statement**:

    ```
    select cast('abc' as bigint);
    ```

COALESCE

- Syntax

  ```
  coalesce(<expr1>, <expr2>, ...)
  ```

- Description

  Returns the first non-null value in `<expr1>, <expr2>, ...`.

- Parameters

  expr: required. The values that you want to check.

- Return value

  The data type of the return value is the same as the data type of the input parameter.

- Examples
  - Example 1: common usage. Sample statement:

    ```
    -- The value 1 is returned.
    select coalesce(null,null,1,null,3,5,7);
    ```

- Example 2: If the data types of parameter values are not defined, an error is returned.
  - **Incorrect sample statement:**

```
-- The value abc cannot be identified because the data type of the value abc is not d
efined. An error is returned.
select coalesce(null,null,1,null,abc,5,7);
```

  - **Correct sample statement:**

```
select coalesce(null,null,1,null,'abc',5,7);
```

- Example 3: If data is not read from a table and all the values of the input parameters are null, an error is returned. **Incorrect sample statement:**

```
-- An error is returned because non-null values do not exist.
select coalesce(null,null,null,null);
```

- Example 4: If data is read from a table and all the values of the input parameters are null, null is returned.

  Original data table:

```
+-----------+-------------+------------+
| shop_name | customer_id | toal_price |
+-----------+-------------+------------+
| ad        | 10001       | 100.0      |
| jk        | 10002       | 300.0      |
| ad        | 10003       | 500.0      |
| tt        | NULL        | NULL       |
+-----------+-------------+------------+
```

  The field values for the tt shop in the original table are all null. After the following statement is executed, the value null is returned:

```
select coalesce(customer_id,total_price) from sale_detail where shop_name='tt';
```

COMPRESS

- Syntax

```
binary compress(string <str>)
binary compress(binary <bin>)
```

- Description

  Uses the GZIP algorithm to compress str or bin.

- Parameters
  - str: required. A value of the STRING type.
  - bin: required. A value of the BINARY type.
- Return value

  A value of the BINARY type is returned. If the input parameter is set to null, null is returned.

- Examples

- Example 1: Use the GZIP algorithm to compress string `hello`. Sample statement:

```
-- =1F=8B=08=00=00=00=00=00=00=03=CBH=CD=C9=C9=07=00=86=A6=106=05=00=00=00 is returned.

select compress('hello');
```

- Example 2: The input parameter is empty. Sample statement:

```
-- =1F=8B=08=00=00=00=00=00=00=03=03=00=00=00=00=00=00=00=00=00 is returned.
select compress('');
```

- Example 3: The input parameter is set to null. Sample statement:

```
-- The value null is returned.
select compress(null);
```

CRC32

- Syntax

```
bigint crc32(string|binary <expr>)
```

- Description

  Calculates the CRC value of a value specified by expr. The value is of the STRING or BINARY type.

- Parameters

  expr: required. A value of the STRING or BINARY type.

- Return value

  A value of the BIGINT type is returned. The return value varies based on the values of the input parameters.

  - If the input parameter is set to null, null is returned.
  - If an input parameter is left empty, 0 is returned.

- Examples

  - Example 1: Calculate the CRC value of string `ABC`. Sample statement:

```
-- 2743272264 is returned.
select crc32('ABC');
```

  - Example 2: An input parameter is set to null. Sample statement:

```
-- The value null is returned.
select crc32(null);
```

DECODE

- Syntax

```
decode(<expression>, <search>, <result>[, <search>, <result>]...[, <default>])
```

- Description

  Implements the `IF-THEN-ELSE` logic.

- Parameters
  - expression: required. The expression that you want to compare.
  - search: required. The search item that is used to compare with expression.
  - result: required. The value that is returned when the value of search is the same as the value of expression.
  - default: optional. If no search items match the expression, the value of default is returned. If no value is specified for this parameter, null is returned.

  > **Note**
  > - Except for the null values, all other values of the result parameter must be of the same data type. If the values are of different data types, an error is returned.
  > - The values of search and expression must be of the same data type. Otherwise, an error is returned.

- Return value
  - If a search item matches the expression, result is returned.
  - If no search item matches the expression, default is returned.
  - If no value is specified for the default parameter, null is returned.
  - If duplicate search items match the expression, the value of the first search item is returned.
  - In most cases, null is returned when MaxCompute SQL calculates `NULL=NULL`. However, the DECODE function considers that the two null values are the same.

- Examples

  The `sale_detail` table contains the `shop_name (STRING), customer_id (STRING), and total_price (DOUBLE)` fields. This table contains the following data:

```
+------------+-------------+-------------+------------+------------+
| shop_name  | customer_id | total_price | sale_date  | region     |
+------------+-------------+-------------+------------+------------+
| s1         | c1          | 100.1       | 2013       | china      |
| s2         | c2          | 100.2       | 2013       | china      |
| s3         | c3          | 100.3       | 2013       | china      |
| null       | c5          | NULL        | 2014       | shanghai   |
| s6         | c6          | 100.4       | 2014       | shanghai   |
| s7         | c7          | 100.5       | 2014       | shanghai   |
+------------+-------------+-------------+------------+------------+
```

  Sample statement:

```
-- If the value of customer_id is c1, Taobao is returned. If the value is c2, Alipay is r
eturned. If the value is c3, Aliyun is returned. If the value is null, N/A is returned. I
n other cases, Others is returned.
select
decode(customer_id,
'c1', 'Taobao',
'c2', 'Alipay',
'c3', 'Aliyun',
Null, 'N/A',
'Others') as result
from sale_detail;
-- The preceding statement is equivalent to the following statement:
if customer_id = c1 then
result := 'Taobao';
elsif customer_id = c2 then
result := 'Alipay';
elsif customer_id = c3 then
result := 'Aliyun';
...
else
result := 'Others';
end if;
```

The following result is returned:

```
+------------+
| result     |
+------------+
| Others     |
| Others     |
| Others     |
| Taobao     |
| Alipay     |
| Aliyun     |
+------------+
```

DECOMPRESS

- Syntax

```
binary decompress(binary <bin>)
```

- Description

  Uses the GZIP algorithm to decompress bin.

- Parameters

  bin: required. A value of the BINARY type.

- Return value

  A value of the BINARY type is returned. If the input parameter is set to null, null is returned.

- Examples

- Example 1: Decompress the compression result of string `hello, world`, and convert the decompression result into a string. Sample statement:

```
-- hello, world is returned.
select cast(decompress(compress('hello, world')) as string);
```

- Example 2: The input parameter is set to null. Sample statement:

```
-- The value null is returned.
select decompress(null);
```

GET_IDCARD_AGE

- Syntax

```
get_idcard_age(<idcardno>)
```

- Description

  Returns the current age based on the ID card number. The current age is the current year minus the birth year in the ID card number.

- Parameters

  idcardno: required. A 15-digit or 18-digit ID card number of the STRING type. During calculation, the validity of the ID card is checked based on the province code and the last digit of the ID card number. If the check fails, null is returned.

- Return value

  A value of the BIGINT type is returned. If the input parameter is set to null, null is returned.

GET_IDCARD_BIRTHDAY

- Syntax

```
get_idcard_birthday(<idcardno>)
```

- Description

  Returns the date of birth based on the ID card number.

- Parameters

  idcardno: required. A 15-digit or 18-digit ID card number of the STRING type. During calculation, the validity of the ID card is checked based on the province code and the last digit of the ID card number. If the check fails, null is returned.

- Return value

  A value of the DATETIME type is returned. If the input parameter is set to null, null is returned.

GET_IDCARD_SEX

- Syntax

```
get_idcard_sex(<idcardno>)
```

- Description

Returns the gender based on the ID card number. Valid values: `M` and `F`. M indicates male and F indicates female.

- Parameters

  idcardno: required. The 15-digit or 18-digit ID card number of the STRING type. During calculation, the validity of the ID card is checked based on the province code and the last digit of the ID card number. If the check fails, null is returned.

- Return value

  A value of the STRING type is returned. If the input parameter is set to null, null is returned.

GET_USER_ID

- Syntax

```
get_user_id()
```

- Description

  Obtains the ID of the current account, which is the user ID or user identifier (UID).

- Parameters

  No parameters are required.

- Return value

  The ID of the current account is returned.

- Examples

```
select get_user_id();
-- The following result is returned:
+------------+
| _c0        |
+------------+
| 1117xxxxxxxx8519 |
+------------+
```

GREATEST

- Syntax

```
greatest(<var1>, <var2>[,...])
```

- Description

  Returns the maximum value of the input parameters.

- Parameters

  var1 and var2: required. The parameters are of the BIGINT, DOUBLE, DECIMAL, DATETIME, or STRING type.

- Return value
  - The maximum value of the input parameters is returned. If implicit conversions are not performed, the return value is of the same data type as the input parameters.
  - The value null is interpreted as the minimum value.

- If the input parameters are of different data types, the input parameters of the DOUBLE, BIGINT, DECIMAL, and STRING types are converted into the DOUBLE type for comparison, and the input parameters of the STRING and DATETIME types are converted into the DATETIME type for comparison. Implicit conversions of other data types are not allowed.
- If `odps.sql.hive.compatible` is set to true and an input parameter is set to null, null is returned.

HASH

- Syntax
  - Syntax for MaxCompute projects that use the Hive-compatible data type edition:

    ```
    int hash(<value1>, <value2>[, ...]);
    ```

  - Syntax for MaxCompute projects that do not use the Hive-compatible data type edition:

    ```
    bigint hash(<value1>, <value2>[, ...]);
    ```

- Description

  Returns a hash value that is obtained after a hash operation is performed on value1 and value2.

- Parameters

  value1 and value2: required. These parameters specify the parameters on which you want to perform a hash operation. The parameters can be of different data types. The data types supported in the Hive-compatible data type edition and non-Hive-compatible data type editions are different.

  - Data types supported in the Hive-compatible data type edition: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, BOOLEAN, STRING, CHAR, VARCHAR, DATETIME, and DATE.
  - Data types supported in non-Hive-compatible data type editions: BIGINT, DOUBLE, BOOLEAN, STRING, and DATETIME.

  > **Note** If the two input parameters are set to the same value, the returned hash values are the same. However, if the two returned hash values are the same, the value of the two input parameters may not be the same and a hash collision may occur.

- Return value

  A value of the INT or BIGINT type is returned. If an input parameter is left empty or set to null, 0 is returned.

- Examples
  - Example 1: Calculate the hash value of the input parameters that are of the same data type. Sample statement:

    ```
    -- The value 66 is returned.
    select hash(0, 2, 4);
    ```

  - Example 2: Calculate the hash value of the input parameters that are of different data types. Sample statement:

    ```
    -- The value 97 is returned.
    select hash(0, 'a');
    ```

- Example 3: An input parameter is left empty or is set to null. Sample statements:

```
-- The value 0 is returned.
select hash(0, null);
-- The value 0 is returned.
select hash(0, '');
```

IF

- Syntax

```
if(<testCondition>, <valueTrue>, <valueFalseOrNull>)
```

- Description

  Checks whether testCondition is true. If testCondition is true, the value of valueTrue is returned. Otherwise, the value of valueFalseOrNull is returned.

- Parameters
  - testCondition: required. The expression that you want to evaluate. The value is of the BOOLEAN type.
  - valueTrue: required. The value that is returned when testCondition is true.
  - valueFalseOrNull: the value that is returned when testCondition is false. You can set this parameter to null.

- Return value

  The data type of the return value is the same as the data type of valueTrue or valueFalseOrNull.

- Examples

```
-- The value 200 is returned.
select if(1=2, 100, 200);
```

LEAST

- Syntax

```
least(<var1>, <var2>[,...])
```

- Description

  Returns the minimum value of the input parameters.

- Parameters

  var: required. The values of the input parameters. The parameters are of the BIGINT, DOUBLE, DECIMAL, DATETIME, or STRING type.

- Return value
  - The minimum value of the input parameters is returned. If implicit conversions are not performed, the return value is of the same data type as the input parameters.
  - If a data type conversion is performed between the DOUBLE, BIGINT, and STRING types, a value of the DOUBLE type is returned. If a data type conversion is performed between the STRING and DATETIME types, a value of the DATETIME type is returned. If a data type conversion is performed between the DECIMAL, DOUBLE, BIGINT, and STRING types, a value of the DECIMAL type is returned. Implicit conversions of other data types are not allowed.

- The value null is interpreted as the minimum value.
- If the values of all input parameters are null, null is returned.

- Examples

```
-- The value 2 is returned.
select least(5, 2, 7);
```

MAX_PT

- Syntax

```
max_pt(<table_full_name>)
```

- Description

  Returns the name of the largest level-1 partition that contains data in a partitioned table and reads the data of this partition. This function determines the largest partition by sorting partitions in alphabetical order.

  You can also use a standard SQL statement instead of the statement in which the `MAX_PT` function is used. For example, you can use `select * from table where pt = (select max(pt) from table);` instead of `select * from table where pt = max_pt("table");` .

  > **Note** MaxCompute does not provide the `MIN_PT` function. If you need to obtain the smallest partition in which data is stored in a partitioned table, you cannot use the SQL statement `select * from table where pt = min_pt("table");` . Instead, you can use the standard SQL statement `select * from table where pt = (select min(pt) from table);` to achieve the same effect as the `MAX_PT` function.

- Parameters

  table_full_name: required. A value of the STRING type. This parameter specifies the name of the table. You must have read permissions on the table.

- Return value

  The name of the largest partition is returned.

  > **Note**
  >
  > If a partition is added by using the `ALTER TABLE` statement and the partition contains no data, the name of this partition is not returned.

- Examples
  - Example 1: The tbl table is a partitioned table. The partitions in the table are 20120901 and 20120902, both of which contain data. If you execute the following statement, this `function` returns `'20120902'` , and the MaxCompute SQL statement reads data from the `20120902` partition. Sample statement:

    ```
    select * from tbl where pt=max_pt('myproject.tbl');
    -- The preceding statement is equivalent to the following statement:
    select * from tbl where pt = (select max(pt) from myproject.tbl);
    ```

- Example 2: If a partitioned table contains multiple levels of partitions, use the standard SQL statement to obtain data from the largest partition. Sample statement:

```
select * from table where pt1 = (select max(pt1) from table) and pt2 = (select max(pt2)
from table where pt1 = (select max(pt1) from table));
```

NULLIF

- Syntax

```
T nullif(T <expr1>, T <expr2>)
```

- Description

  Compares the values of expr1 and expr2. If the values are the same, null is returned. If the values are not the same, the value of expr1 is returned.

- Parameters

  expr1 and expr2: required. Expressions of any data type. `T` specifies the type of input data. The type can be a data type supported by MaxCompute.

- Return value

  The value of expr1 or null is returned.

- Examples

```
-- The value 2 is returned.
select nullif(2, 3);
-- The value null is returned.
select nullif(2, 2);
-- The value 3 is returned.
select nullif(3, null);
```

NVL

- Syntax

```
nvl(T <value>, T <default_value>)
```

- Description

  Returns default_value if value is null. Otherwise, value is returned. The value and default_value parameters must be of the same data type.

- Parameters
  - value: required. The input parameter. `T` specifies the type of input data. The type can be a data type supported by MaxCompute.
  - default_value: required. The value that is used to replace null. The data type of default_value must be the same as the data type of value.

- Examples

  A table named `t_data` contains three columns: `c1`, `c2`, and `c3`. The c1 column is of the STRING type. The c2 column is of the BIGINT type. The c3 column is of the DATETIME type. This table contains the following data:

```
+----+------------+------------+
| c1 | c2 | c3 |
+----+------------+------------+
| NULL | 20 | 2017-11-13 05:00:00 |
| ddd | 25 | NULL |
| bbb | NULL | 2017-11-12 08:00:00 |
| aaa | 23 | 2017-11-11 00:00:00 |
+----+------------+------------+
```

After the `NVL` function is called, the null value in `c1` is returned as 00000, the null value in `c2` is returned as 0, and the null value in `c3` is returned as a hyphen (`-`). Sample statement:

```
select nvl(c1,'00000'),nvl(c2,0),nvl(c3,'-') from nvl_test;
-- The following result is returned:
+-----+------------+-----+
| _c0 | _c1 | _c2 |
+-----+------------+-----+
| 00000 | 20 | 2017-11-13 05:00:00 |
| ddd | 25 | - |
| bbb | 0 | 2017-11-12 08:00:00 |
| aaa | 23 | 2017-11-11 00:00:00 |
+-----+------------+-----+
```

ORDINAL

- Syntax

```
ordinal(bigint <nth>, <var1>, <var2>[,...])
```

- Description

  Sorts the values of the input variables in ascending order and returns the value that is ranked nth.

- Parameters
  - nth: required. A value of the BIGINT type. This parameter specifies the position of the value that you want to return. If the value is null, null is returned.
  - var: required. A value of the BIGINT, DOUBLE, DATETIME, or STRING type. This parameter specifies the values that you want to sort.

- Return value
  - The value that is ranked nth is returned. If implicit conversions are not performed, the return value is of the same data type as the input parameter.
  - If a data type conversion is performed between the DOUBLE, BIGINT, and STRING types, a value of the DOUBLE type is returned. If a data type conversion is performed between the STRING and DATETIME types, a value of the DATETIME type is returned. Implicit conversions of other data types are not allowed.
  - The value null is interpreted as the minimum value.

- Examples

```
-- The value 2 is returned.
select ordinal(3, 1, 3, 2, 5, 2, 4, 6);
```

PARTITION_EXISTS

- Syntax

```
boolean partition_exists(string <table_name>, string... <partitions>)
```

- Description

  Checks whether a specified partition exists in a table.

- Parameters
  - table_name: required. The table name, which is of the STRING type. You can specify a project name in the table name. The name of a table can be `my_proj.my_table`. If you do not specify a project name, the current project name is used.
  - partitions: required. The names of partitions, which are of the STRING type. In this parameter, you must specify the values of partition key columns in a table based on the sequence of the columns. The number of values must be the same as the number of partition key columns.

- Return value

  A value of the BOOLEAN type is returned. If the specified partition exists, True is returned. Otherwise, False is returned.

- Examples

```
-- Create a partitioned table named foo.
create table foo (id bigint) partitioned by (ds string, hr string);
-- Add partitions to the partitioned table named foo.
alter table foo add partition (ds='20190101', hr='1');
-- Check whether partitions 20190101 and 1 exist. True is returned.
select partition_exists('foo', '20190101', '1');
```

SAMPLE

- Syntax

```
boolean sample(<x>, <y>, [<column_name1>, <column_name2>[,...]])
```

- Description

  Samples all values that are read from column_name based on x and y, and filters out the rows that do not meet sampling conditions.

- Parameters
  - x and y: x is required. x and y are integer constants that are greater than 0. Their values are of the BIGINT type. The two parameters indicate that the values fall into x portions based on the hash function and the yth portion is used.

    y is optional. If no value is specified for the y parameter, the first option is used, and you do not need to specify column_name.

    If x or y is of another data type, the value of x or y is less than or equal to 0, or y is greater than x, an error is returned. If the value of x or y is null, null is returned.

- column_name: optional. The name of the column on which sampling is performed. If no value is specified for this parameter, random sampling is performed based on the values of x and y. The name of the column can be of any data type, and the column value can be null. Implicit conversions are not performed. If column_name is set to null, an error is returned.

  > **Note** To avoid data skew due to the null value, uniform hashing is performed on the null values in column_name in x portions. If no value is specified for the column_name parameter and the amount of data is small, the output is not necessarily uniform. In this case, we recommend that you specify a value for column_name to obtain a uniform output.

- Return value

  A value of the BOOLEAN type is returned.

- Examples

  The `tbla` table contains the `cola` column.

```
-- The values in the cola column fall into four portions based on the hash function, and the first portion is used. True is returned.
select * from tbla where sample (4, 1 , cola);
-- The values in each row are randomly hashed to four portions, and the second portion is used. True is returned.
select * from tbla where sample (4, 2);
```

SHA

- Syntax

```
string sha(string|binary <expr>)
```

- Description

  Calculates the SHA-1 hash value of expr that is of the STRING or BINARY type and returns the SHA-1 hash value in hexadecimal string format.

- Parameters

  expr: required. A value of the STRING or BINARY type.

- Return value

  A value of the STRING type is returned. If the input parameter is set to null, null is returned.

- Examples
  - Example 1: Calculate the SHA hash value of string `ABC`. Sample statement:

```
-- 3c01bdbb26f358bab27f267924aa2c9a03fcfdb8 is returned.
select sha('ABC');
```

  - Example 2: The input parameter is set to null. Sample statement:

```
-- The value null is returned.
select sha(null);
```

SHA1

- Syntax

```
string sha1(string|binary <expr>)
```

- Description

  Calculates the SHA-1 hash value of expr that is of the STRING or BINARY type and returns the SHA-1 hash value in hexadecimal string format.

- Parameters

  expr: required. A value of the STRING or BINARY type.

- Return value

  A value of the STRING type is returned. If the input parameter is set to null, null is returned.

- Examples
  - Example 1: Calculate the SHA-1 hash value of string `ABC`. Sample statement:

    ```
    -- 3c01bdbb26f358bab27f267924aa2c9a03fcfdb8 is returned.
    select sha1('ABC');
    ```

  - Example 2: The input parameter is set to null. Sample statement:

    ```
    -- The value null is returned.
    select sha1(null);
    ```

SHA2

- Syntax

```
string sha2(string|binary <expr>, bigint <number>)
```

- Description

  Calculates the SHA-2 hash value of expr that is of the STRING or BINARY type and returns the SHA-2 hash value in the format specified by number.

- Parameters
  - expr: required. A value of the STRING or BINARY type.
  - number: required. A value of the BIGINT type. This parameter specifies the hash bit length, which must be 224, 256, 384, 512, or 0. The value that is returned when this parameter is set to 256 is the same as the value that is returned when this parameter is set to 0.

- Return value

  A value of the STRING type is returned. The value null is returned in the following scenarios:
  - An input parameter is set to null.
  - The value of number does not fall in the valid value range.

- Examples
  - Example 1: Calculate the SHA-2 hash value of string `ABC`. Sample statement:

    ```
    -- b5d4045c3f466fa91fe2cc6abe79232a1a57cdf104f7a26e716e0a1e2789df78 is returned.
    select sha2('ABC', 256);
    ```

- Example 2: An input parameter is set to null. Sample statement:

```
-- The value null is returned.
select sha2('ABC', null);
```

SIGN

- Syntax

```
sign(<x>)
```

- Description

  Determines the sign of x. The sign indicates whether a value is positive or negative.

- Parameters

  x: required. A value of the DOUBLE or DECIMAL type. This parameter specifies a constant, a function, or an expression.

- Return value
  - If the value of x is positive, 1.0 is returned.
  - If the value of x is negative, -1.0 is returned.
  - If the value of x is 0, 0.0 is returned.
  - If the value of x is empty, an error is returned.

- Examples

```
-- The value -1.0 is returned.
select sign(5-13);
```

SPLIT

- Syntax

```
split(<str>, <pat>)
```

- Description

  Returns an array after str is split with pat.

- Parameters
  - str: required. A value of the STRING type. This parameter specifies the string that you want to split.
  - pat: required. A delimiter of the STRING type. Regular expressions are supported. For more information about regular expressions, see Regular expressions.

- Return value

  An array is returned. The elements in the array are of the STRING type.

- Examples

```
-- The array [a,  b,  c] is returned.
select split("a, b, c", ",");
```

STACK

- Syntax

```
stack(n, expr1, ..., exprk)
```

- Description

  Splits `expr1, ..., exprk` into n rows. Unless otherwise specified, the output result uses the default column names `col0, col1...`.

- Parameters
  - n: required. The number of rows obtained after splitting.
  - expr: required. The parameter that you want to split. `expr1,... exprk` must be of the INTEGER type, and the number of parameters must be an integer multiple of n. The parameter must be able to be split into n complete rows. Otherwise, an error is returned.

- Return value

  n rows with a specific number of columns are returned. The number of columns is equal to the number of parameters divided by n.

- Examples

```
-- Split the parameter group of 1, 2, 3, 4, 5, 6 into three rows.
select stack(3, 1, 2, 3, 4, 5, 6);
-- The following result is returned:
+------+------+
| col0 | col1 |
+------+------+
| 1    | 2    |
| 3    | 4    |
| 5    | 6    |
+------+------+
-- Split 'A',10,date '2015-01-01','B',20,date '2016-01-01' into two rows.
select stack(2,'A',10,date '2015-01-01','B',20,date '2016-01-01') as (col0,col1,col2);
-- The following result is returned:
+------+------+------+
| col0 | col1 | col2 |
+------+------+------+
| A    | 10   | 2015-01-01 |
| B    | 20   | 2016-01-01 |
+------+------+------+
-- Split the parameter group of a, b, c, and d into two rows. If the source table contain
s multiple rows, this function is called for each row.
select stack(2,a,b,c,d) as (col,value)
from values
    (1,1,2,3,4),
    (2,5,6,7,8),
    (3,9,10,11,12),
    (4,13,14,15,null)
as t(key,a,b,c,d);
-- The following result is returned:
+------+-------+
| col  | value |
+------+-------+
| 1    | 2     |
| 3    | 4     |
| 5    | 6     |
| 7    | 8     |
| 9    | 10    |
| 11   | 12    |
| 13   | 14    |
| 15   | NULL  |
+------+-------+
-- Use this function with the LATERAL VIEW clause.
select tf.* from (select 0) t lateral view stack(2,'A',10,date '2015-01-01','B',20, date
'2016-01-01') tf as col0,col1,col2;
-- The following result is returned:
+------+------+------+
| col0 | col1 | col2 |
+------+------+------+
| A    | 10   | 2015-01-01 |
| B    | 20   | 2016-01-01 |
+------+------+------+
```

STR_TO_MAP

- Syntax

```
str_to_map([string <mapDupKeyPolicy>,] <text> [, <delimiter1> [, <delimiter2>]])
```

- Description

  Splits text into key-value pairs by using delimiter1 and then separates keys from values in the key-value pairs by using delimiter2.

- Parameters
  - mapDupKeyPolicy: optional. A value of the STRING type. This parameter specifies the method that is used to process duplicate keys. Valid values:
    - exception: An error is returned.
    - last_win: The latter key overwrites the former key.

    You can also specify the `odps.sql.map.key.dedup.policy` parameter at the session level to configure the method that is used to process duplicate keys. For example, you can set `odps.sql.map.key.dedup.policy` to exception. If you do not specify this parameter, the default value last_win is used.

    > **Note** The behavior implementation of MaxCompute is determined based on mapDupKeyPolicy. If you do not specify mapDupKeyPolicy, the value of `odps.sql.map.key.dedup.policy` prevails.

  - text: required. The string that you want to split, which is of the STRING type.
  - delimiter1: optional. The delimiter of the STRING type. If no value is specified for this parameter, commas (`,`) are used.
  - delimiter2: optional. The delimiter of the STRING type. If no value is specified for this parameter, equal signs (`=`) are used.

    > **Note** If the delimiter is a regular expression or special character, you must add two backslashes (\\) before the delimiter for escaping. The following special characters can be used as a delimiter: . ? + * :.

- Return value

  A value of the `MAP<STRING, STRING>` type is returned. The return value indicates that the string specified by text is split by using delimiter1 and delimiter2.

- Examples

```
-- {test1:1, test2:2} is returned.
select str_to_map('test1&1-test2&2','-','&');
-- {test1:1, test2:2} is returned.
select str_to_map("test1.1,test2.2", ",", "\\.");
-- {test1:1, test2:3} is returned.
select str_to_map("test1.1,test2.2,test2.3", ",", "\\.");
```

TABLE_EXISTS

- Syntax

```
boolean table_exists(string <table_name>)
```

- Description

  Checks whether a specified table exists.

- Parameters

  table_name: required. The name of the table. A value of the STRING type. You can specify a project name in the table name. The name of a table can be `my_proj.my_table`. If you do not specify a project name, the current project name is used.

- Return value

  A value of the BOOLEAN type is returned. If the specified table exists, True is returned. Otherwise, False is returned.

- Examples

```
-- Use this function for the list in the SELECT statement.
select if(table_exists('abd'), col1, col2) from src;
```

TRANS_ARRAY

- Limits
  - All columns that are used as `keys` must be placed before the columns that are to be transposed.
  - Only one UDTF is allowed in a `SELECT` statement.
  - This function cannot be used with the `GROUP BY`, `CLUSTER BY`, `DISTRIBUTE BY`, or `SORT BY` clause.

- Syntax

```
trans_array (<num_keys>, <separator>, <key1>,<key2>,…,<col1>,<col2>,<col3>) as (<key1>,<key2>,...,<col1>, <col2>)
```

- Description

  Transposes one row of data into multiple rows. This function is a UDTF that transposes an array separated by fixed delimiters in a column into multiple rows.

- Parameters
  - num_keys: required. The value is a constant of the BIGINT type and must be greater than or equal to `0`. This parameter specifies the number of columns that can be used as `keys` when you transpose one row into multiple rows.
  - separator: required. The value is a constant of the STRING type. This parameter is used to split a string into multiple elements. If this parameter is left empty, an error is returned.
  - keys: required. The columns that are used as `keys` when you transpose one row into multiple rows. The number of keys is specified by num_keys. If all columns are used as `keys` and num_keys is equal to the total number of all columns, only one row is returned.
  - cols: required. The array that you want to transpose into rows. All columns that follow `keys` are considered arrays to be transposed. The parameter value must be of the STRING type to store arrays in the STRING format, such as `Hangzhou;Beijing;shanghai`. The values in this array are separated by semicolons (`;`).

- Return value

  Transposed rows are returned. The new column name is specified by `as`. The data types of columns that are used as `keys` remain unchanged. All other columns are of the STRING type. The number of transposed rows is based on the array with the maximum number of elements. If the number of rows is insufficient, the value null is added.

- Examples
  - Example 1: The `t_table` table contains the following data:

```
+----------+----------+------------+
| login_id | login_ip | login_time |
+----------+----------+------------+
| wangwangA | 192.168.0.1,192.168.0.2 | 20120101010000,20120102010000 |
| wangwangB | 192.168.45.10,192.168.67.22,192.168.6.3 | 20120111010000,20120112010000,2
0120223080000 |
+----------+----------+------------+
-- Execute the following SQL statement:
select trans_array(1, ",", login_id, login_ip, login_time) as (login_id,login_ip,login_
time) from t_table;
-- The following result is returned:
+----------+----------+------------+
| login_id | login_ip | login_time |
+----------+----------+------------+
| wangwangB | 192.168.45.10 | 20120111010000 |
| wangwangB | 192.168.67.22 | 20120112010000 |
| wangwangB | 192.168.6.3 | 20120223080000 |
| wangwangA | 192.168.0.1 | 20120101010000 |
| wangwangA | 192.168.0.2 | 20120102010000 |
+----------+----------+------------+
-- The table contains the following data:
Login_id LOGIN_IP LOGIN_TIME
wangwangA 192.168.0.1,192.168.0.2 20120101010000
-- The value null is added to supplement the array in which data is insufficient.
Login_id Login_ip Login_time
wangwangA 192.168.0.1 20120101010000
wangwangA 192.168.0.2 NULL
```

- Example 2: The mf_fun_array_test_t table contains the following data:

```
+------------+------------+------------+------------+
| id         | name       | login_ip   | login_time |
+------------+------------+------------+------------+
| 1          | Tom        | 192.168.100.1,192.168.100.2 | 20211101010101,20211101010102
|
| 2          | Jerry      | 192.168.100.3,192.168.100.4 | 20211101010103,20211101010104
|
+------------+------------+------------+------------+
-- Use the id and name columns as keys to transpose data in the table. Execute the foll
owing SQL statement:
select trans_array(2, ",", Id,Name, login_ip, login_time) as (Id,Name,login_ip,login_ti
me) from mf_fun_array_test_t;
-- The following result is returned:
+------------+------------+------------+------------+
| id         | name       | login_ip   | login_time |
+------------+------------+------------+------------+
| 1          | Tom        | 192.168.100.1 | 20211101010101 |
| 1          | Tom        | 192.168.100.2 | 20211101010102 |
| 2          | Jerry      | 192.168.100.3 | 20211101010103 |
| 2          | Jerry      | 192.168.100.4 | 20211101010104 |
+------------+------------+------------+------------+
```

TRANS_COLS

- Limits
  - All columns that are used as `keys` must be placed before the columns that are to be transposed.
  - Only one UDTF is allowed in a `SELECT` statement.
- Syntax

```
trans_cols (<num_keys>, <key1>,<key2>,…,<col1>, <col2>,<col3>) as (<idx>, <key1>,<key2>,…
,<col1>, <col2>)
```

- Description

  Transposes one row of data into multiple rows. This function is a UDTF that transposes columns into rows.
- Parameters
  - num_keys: required. The value is a constant of the BIGINT type and must be greater than or equal to `0`. This parameter specifies the number of columns that can be used as keys when you transpose one row into multiple rows.
  - keys: required. The columns that are used as keys when you transpose one row into multiple rows. The number of keys is specified by num_keys. If all columns are used as keys and num_keys is equal to the total number of all columns, only one row is returned.
  - idx: required. The ID of a row after the row is transposed.
  - cols: required. The columns that you want to transpose into rows.
- Return value

Transposed rows are returned. The new column name is specified by `as`. The first output column is the transposed subscript, which starts from 1. The data types of the columns that are used as keys remain unchanged, and the data types of other columns remain unchanged.

- Examples

  The `t_table` table contains the following data:

```
+----------+----------+------------+
| Login_id | Login_ip1 | Login_ip2 |
+----------+----------+------------+
| wangwangA | 192.168.0.1 | 192.168.0.2 |
+----------+----------+------------+
-- Execute the following SQL statement:
select trans_cols(1, login_id, login_ip1, login_ip2) as (idx, login_id, login_ip) from t_
table;
-- The following result is returned:
idx    login_id    login_ip
1    wangwangA    192.168.0.1
2    wangwangA    192.168.0.2
```

UNBASE64

- Syntax

```
binary unbase64(string <str>)
```

- Description

  Converts a Base64-encoded string specified by str into a binary value.

- Parameters

  str: required. A value of the STRING type. It is a Base64-encoded string that you want to convert.

- Return value

  A value of the BINARY type is returned. If the input parameter is set to null, null is returned.

- Examples
  - Example 1: Convert the string `YWxpYmFiYQ==` into a binary value. Sample statement:

```
-- alibaba is returned.
select unbase64('YWxpYmFiYQ==');
```

  - Example 2: The input parameter is set to null. Sample statement:

```
-- The value null is returned.
select unbase64(null);
```

UNIQUE_ID

- Syntax

```
string unique_id()
```

- Description

Returns a unique ID, such as `29347a88-1e57-41ae-bb68-a9edbdd9****_1`. This function is more efficient than the UUID function, and the returned ID is longer. Compared with the UUID function, this function returns a unique ID that ends with the suffix "_Digit", such as `_1`.

UUID

- Syntax

```
string uuid()
```

- Description

  Returns a random ID, such as `29347a88-1e57-41ae-bb68-a9edbdd9****`.

  > **Note** The return value is a random global ID, which is unique in most cases.

3.9.10. Use cases of built-in functions

3.9.10.1. Fix the precision issue of the ROUND function

The ROUND function rounds input values based on the number of decimal places that you specify. This topic describes the precision issue of the ROUND function and provides a solution and an example for reference.

Example

The returned result of the ROUND function in SQL code does not meet the expectation. Sample statement:

```
select round(0.25375,4);
```

The returned result is expected to be 0.2538 based on the logic of the ROUND function. However, the input value is not rounded up, and 0.2537 is returned.

Cause

MaxCompute converts the input value of the function to the DOUBLE type before the function is executed. During the data type conversion, loss of precision occurs. As a result, the returned result of the ROUND function is inaccurate. In this example, 0.25375 is converted to 0.2536999999... As a result, 0.2537 is returned.

When you use the ROUND function to process data in a table, the actual data types of the table fields are read. In this case, the precision issue cannot occur.

Solution

To ensure the precision of input values, you can use the CAST function to convert the input values that you specified to the DECIMAL type. The precision issue does not occur on data of the DECIMAL type. Sample statement:

```
select round(cast(0.25375 as decimal), 4);
```

The returned result is 0.2538.

3.9.10.2. Implement capabilities provided by the GROUP_CONCAT function

GROUP_CONCAT is a function provided by MySQL. This function is used to concatenate values of specified fields in each group that is generated by using GROUP BY. This topic describes how to use a built-in function of MaxCompute to implement the capabilities that are provided by the GROUP_CONCAT function.

Example

A table named price_total contains the following columns: name, price, and saleid. The table contains the following data:

```
+--------+------------+------------+
| name   | price      | saleid     |
+--------+------------+------------+
| bag    | 50         | 1          |
| sugar  | 20         | 3          |
| noodle | 2          | 4          |
| potato | 5          | 6          |
| bag    | 100        | 2          |
| sugar  | 10         | 4          |
| potato | 4          | 3          |
| sugar  | 50         | 7          |
| noodle | 2          | 5          |
| noodle | 5          | 1          |
+--------+------------+------------+
```

In this example, all products are grouped by product name, which is the name column in the table. You can use the GROUP_CONCAT function in MySQL to meet the following requirements:

- Requirement 1: Merge the values in the price column of the same group. The returned results contain duplicate values. Sample statement:

  ```
  select name, group_concat(price) from price_total group by name;
  ```

- Requirement 2: Merge the values in the price column of the same group. The returned results do not contain duplicate values. Sample statement:

  ```
  select name, group_concat(distinct price) from price_total group by name;
  ```

- Requirement 3: Merge and sort the values in the price column of the same group. The returned results contain duplicate values. Sample statement:

  ```
  select name, group_concat(price order by price desc) from price_total group by name;
  ```

- Requirement 4: Merge and sort the values in the price column of the same group. The returned results do not contain duplicate values. Sample statement:

  ```
  select name, group_concat(distinct price order by price desc) from price_total group by n
  ame;
  ```

- Requirement 5: Merge the values in the price column and the values in the saleid column of the same

group. Sample statement:

```
select name, group_concat(concat_ws(':', price, saleid)) from price_total group by name;
```

Solution

MaxCompute does not support the GROUP_CONCAT function. You can use the WM_CONCAT function to meet the requirements in this example.

Note that the WM_CONCAT function is not equivalent to the GROUP_CONCAT function. The following table describes the capabilities of the functions.

Capability	WM_CONCAT	GROUP_CONCAT	Description
Deduplication by using DISTINCT	✓	✓	None.
Delimiter	✓	✓	A delimiter can be used to concatenate the values that you want to merge. Take note of the following points when you include delimiters in the WM_CONCAT and GROUP_CONCAT functions: • Delimiters cannot be omitted when you use the WM_CONCAT function. • Delimiters can be omitted when you use the GROUP_CONCAT function. If delimiters are omitted, commas (,) are used as delimiters.

Capability	WM_CONCAT	GROUP_CONCAT	Description
Sorting	✗	✓	The WM_CONCAT function does not support sorting for merged values. If you want to sort values, use the following method: • Sort the fields that you want to merge and then merge the fields. • Use user-defined functions (UDFs). For more information about UDFs, see MaxCompute UDF.
Merging of multiple columns	✓	✓	The WM_CONCAT and GROUP_CONCAT functions are functions that merge rows. The two functions can be used with the CONCAT function or the CONCAT_WS function that is used to merge columns to merge values of multiple columns in different groups.

You can use the WM_CONCAT function to meet the following requirements based on the function capabilities that are described in the preceding table:

- Requirement 1: Merge the values in the price column of the same group. The returned results contain duplicate values.

  ```
  select name, wm_concat(',', price) as price_new from price_total group by name;
  ```

  The following results are returned:

  ```
  +--------+-----------+
  | name   | price_new |
  +--------+-----------+
  | bag    | 50,100    |
  | noodle | 2,2,5     |
  | potato | 5,4       |
  | sugar  | 20,10,50  |
  +--------+-----------+
  ```

- Requirement 2: Merge the values in the price column of the same group. The returned results do not contain duplicate values.

```
select name, wm_concat(distinct ',', price) as price_new from price_total group by name;
```

The following results are returned:

```
+--------+-----------+
| name   | price_new |
+--------+-----------+
| bag    | 100,50    |
| noodle | 2,5       |
| potato | 4,5       |
| sugar  | 10,20,50  |
+--------+-----------+
```

- Requirement 3: Merge and sort the values in the price column of the same group. The returned results contain duplicate values.

```
-- Remove the constraint on simultaneous execution of ORDER BY and LIMIT.
set odps.sql.validate.orderby.limit=false;
select name, wm_concat(',', price) as price_new from (select name, price from price_total
order by name, price desc) group by name;
```

The following results are returned:

```
+--------+-----------+
| name   | price_new |
+--------+-----------+
| bag    | 100,50    |
| noodle | 5,2,2     |
| potato | 5,4       |
| sugar  | 50,20,10  |
+--------+-----------+
```

- Requirement 4: Merge and sort the values in the price column of the same group. The returned results do not contain duplicate values.

```
-- Remove the constraint on simultaneous execution of ORDER BY and LIMIT.
set odps.sql.validate.orderby.limit=false;
select name, wm_concat(',', price) as price_new from (select distinct name, price from pr
ice_total order by name, price asc) group by name;
```

The following results are returned:

```
+--------+-----------+
| name   | price_new |
+--------+-----------+
| bag    | 50,100    |
| noodle | 2,5       |
| potato | 4,5       |
| sugar  | 10,20,50  |
+--------+-----------+
```

- Requirement 5: Merge the values in the price column and the values in the saleid column of the same group.

```
select name, wm_concat(',', concat_ws(':',price,saleid)) as price_new from price_total gr
oup by name;
-- The preceding statement is equivalent to the following statement:
select name, wm_concat(',', concat(price,':',saleid)) from price_total group by name;
```

The following results are returned:

```
+--------+-----------------+
| name   | price_new       |
+--------+-----------------+
| bag    | 50:1,100:2      |
| noodle | 2:4,2:5,5:1     |
| potato | 5:6,4:3         |
| sugar  | 20:3,10:4,50:7  |
+--------+-----------------+
```

3.9.11. Common errors for built-in functions

This topic describes common errors that may occur when you use MaxCompute built-in functions. This topic helps you know the causes and solutions of the errors.

The following common errors may occur when you use MaxCompute built-in functions:

- Error message: Semantic analysis exception - X type is not enabled in current mode
- Error message: Invalid number of arguments - function Y needs m parameters, actually have n
- Error message: Invalid argument type- invalid type X of argument m for function Y
- Error message: Semantic analysis exception - function or view Y cannot be resolved

Error message: Semantic analysis exception - X type is not enabled in current mode

- Problem description

  When a built-in function is called, the following error message appears:

  ```
  FAILED: ODPS-0130071:[1,27] Semantic analysis exception - TIMESTAMP type is not enabled i
  n current mode. Please set odps.sql.type.system.odps2=true to use it.
  ```

- Cause

  The built-in function that you use in the SQL statement involves new data types of the MaxCompute V2.0 data type edition, such as TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, or BINARY. However, the MaxCompute V2.0 data type edition is not enabled. As a result, MaxCompute fails to process the data.

- Solution

  You can use one of the following methods to enable the MaxCompute V2.0 data type edition on the MaxCompute client:

  - Session level: To enable the MaxCompute V2.0 data type edition at the session level, you must insert `set odps.sql.type.system.odps2=true;` before the SQL statement. Then, commit and execute them together.

- Project level: The project owner can enable the MaxCompute V2.0 data type edition at the project level based on the business requirements. The configuration takes effect after 10 to 15 minutes. To enable the MaxCompute V2.0 data type edition at the project level, run the following command:

  ```
  setproject odps.sql.type.system.odps2=true;
  ```

  After the command takes effect, execute the SQL statement again.

  For more information about `setproject`, see Project operations. For more information about the precautions that you must take when you enable the MaxCompute V2.0 data type edition at the project level, see Data type editions.

Error message: Invalid number of arguments - function Y needs m parameters, actually have n

- Problem description

  When a built-in function is called, the following error message appears:

  ```
  FAILED: ODPS-0130221:[1,8] Invalid number of arguments - function from_utc_timestamp need
  s 2 parameters, actually have 1
  ```

- Cause

  The actual number of input parameters of the built-in function that is used in the SQL statement does not meet the function syntax requirements. As a result, MaxCompute fails to process the data.

- Solution

  Check the number of input parameters of the built-in function, and add or reduce parameters based on the check result to meet the function syntax requirements. Then, execute the SQL statement again.

Error message: Invalid argument type- invalid type X of argument m for function Y

- Problem description

  When a built-in function is called, the following error message appears:

  ```
  FAILED: ODPS-0130121:[1,18] Invalid argument type - invalid type STRING of argument 1 for
  function all_match, expect ARRAY<T>
  ```

- Cause

  The actual data types of input parameters of the built-in function that is used in the SQL statement does not meet the function syntax requirements. As a result, MaxCompute fails to process the data.

- Solution

  Modify the input parameters to ensure that the data types meet the function syntax requirements. Then, execute the SQL statement again.

Error message: Semantic analysis exception - function or view Y cannot be resolved

- Problem description

When a built-in function is called, the following error message appears:

```
FAILED: ODPS-0130071:[1,8] Semantic analysis exception - function or view 'row_number' ca
nnot be resolved
```

- Cause

  The name of the built-in function that is used in the SQL statement is invalid or the built-in function does not have input parameters. As a result, MaxCompute fails to process the data.

- Solution

  Check the function name and input parameter names and add input parameters based on the function syntax requirements. Make sure that the names and syntax format are valid, and then execute the SQL statement again.

3.9.12. FAQ about built-in functions

This topic provides answers to some frequently asked questions about built-in functions of MaxCompute.

Category	FAQ
Date functions	• Does a built-in function of MaxCompute support the conversion from 2010/1/3 to 2010-01-03? • How do I convert a UNIX timestamp of the BIGINT type to a date value of the DATETIME type? • How do I obtain the current system time? • What do I do if the "cannot be resolved" error message appears when I use the YEAR, QUARTER, MONTH, or DAY function? • When I use the TO_DATE function, an error message appears, which indicates that the date value does not contain the minute part. What do I do?
Mathematical functions	Why is an unexpected result returned when I use the ROUND function to round data of the DOUBLE type?
Window functions	What are the functions that I can use to configure an auto-increment sequence in MaxCompute?
Aggregate functions	How do I concatenate values in a field?
String functions	• Does MaxCompute support the MD5 function? • How do I left pad a string with 0s to increase the string length to the specified length? • Does MaxCompute support the SUBSTRING_INDEX function of MySQL? • Does the pattern parameter of the REGEXP_COUNT function support nested query statements? • Does MaxCompute support the to_char (Data, FM9999.00) function of Oracle?

Category	FAQ
Complex type functions	• How do I concatenate all fields that match a specified condition in a JSON string? • How do I use each key in a JSON string as a field? • How do I convert a JSON string into an array?
Other functions	• Which function of MaxCompute is similar to the IFNULL function of MySQL? • How do I convert one row of data into multiple rows of data? • What do I do if the "Expression not in GROUP BY key" error message appears when I use the COALESCE function? • What do I do if the "Invalid function" error message appears when I use the IFNULL function?
Implicit conversions	What do I do if an implicit data type conversion error appears when I use a built-in function of MaxCompute?

Does a built-in function of MaxCompute support the conversion from 2010/1/3 to 2010-01-03?

No, the built-in functions of MaxCompute do not support the conversion from 2010/1/3 to 2010-01-03. You can use `to_char(to_date('2010/01/03', 'yyyy/mm/dd'), 'yyyy-mm-dd')` to convert the date value 2010/01/03 to 2010-01-03. For more information for the built-in functions, see TO_DATE and TO_CHAR.

To convert 2010/1/3 to 2010-01-03, you must write a user-defined function (UDF). For more information about how to write a UDF, see Overview.

How do I convert a UNIX timestamp of the BIGINT type to a date value of the DATETIME type?

You can use the FROM_UNIXTIME function to convert the UNIX timestamp of the BIGINT type to a date value of the DATETIME type. For more information, see FROM_UNIXTIME.

How do I obtain the current system time?

You can use the GETDATE function to obtain the current system time. For more information, see GETDATE.

What do I do if the "cannot be resolved" error message appears when I use the YEAR, QUARTER, MONTH, or DAY function?

- Problem description

  The following error message appears when I use the YEAR, QUARTER, MONTH, or DAY function:

  ```
  FAILED: ODPS-0130071:[1,8] Semantic analysis exception - function or view 'year' cannot b
  e resolved
  ```

- Cause

The YEAR, QUARTER, MONTH, and DAY functions are the extension functions of MaxCompute V2.0. To use these date functions, you must enable the MaxCompute V2.0 data type edition.

- Solution

  Add `set odps.sql.type.system.odps2 = true;` before the SQL statement that you use to enable the MaxCompute V2.0 data type edition.

When I use the TO_DATE function, an error message appears, which indicates that the date value does not contain the minute part. What do I do?

- Problem description

  The following error message appears when an SQL statement that contains `to_date('2016-07-18 18:18:18', 'yyyy-MM-dd HH:mm:ss')` is executed:

  ```
  FAILED: ODPS-0121095:Invalid arguments - format string has second part, but doesn't have minute part : yyyy-MM-dd HH:mm:ss
  ```

- Cause

  The format of the second parameter that is included in the TO_DATE function is invalid. Both `mm` and `MM` indicate the month. `mi` indicates the minute.

- Solution

  Modify the function in the SQL statement to `to_date('2016-07-18 18:18:18', 'yyyy-MM-dd HH:mi:ss')`.

Why is an unexpected result returned when I use the ROUND function to round data of the DOUBLE type?

When I use the `ROUND` function to round a value of the DOUBLE type, the value 4.515 is rounded to 4.51. Sample statement:

```
select round(4.515, 2),round(125.315, 2);
```

A value of the DOUBLE type is an 8-byte double-precision floating point number, which has a difference in precision. The value of the DOUBLE type for `4.515` is `4.514999999...`. As a result, the rounding result is 4.51.

What are the functions that I can use to configure an auto-increment sequence in MaxCompute?

You can use the ROW_NUMBER function to configure an auto-increment sequence. For more information, see ROW_NUMBER.

How do I concatenate values in a field?

You can use the WM_CONCAT function to concatenate values in a field. For more information, see WM_CONCAT.

Does MaxCompute support the MD5 function?

Yes, MaxCompute supports the MD5 function. For more information, see MD5.

How do I left pad a string with 0s to increase the string length to the specified length?

You can use the LPAD function to left pad a string with 0s to increase the string length to the specified length. For more information, see LPAD.

Does MaxCompute support the SUBSTRING_INDEX function of MySQL?

Yes, MaxCompute supports the SUBSTRING_INDEX function. For more information, see SUBSTRING_INDEX.

Does the pattern parameter of the REGEXP_COUNT function support nested query statements?

No, the pattern parameter of the REGEXP_COUNT function does not support nested query statements. For more information about how to use the REGEXP_COUNT function, see REGEXP_COUNT.

Does MaxCompute support the `to_char (Data, FM9999.00)` function of Oracle?

No, MaxCompute does not support this function. If you want to change only the display format of numbers, you can use the FORMAT_NUMBER function. Sample statement:

```
-- The return value is 12,332.123.
select format_number(12332.123456, '#,###,###,###.###');
```

What do I do if the "Invalid function" error message appears when I use the IFNULL function?

- Problem description

  The IFNULL function is used in the following SQL statement:

  ```
  select a.id as id > , ifnull(concat('phs\xxx', a.insy, '\xxxb\xxx', ifnull()))
  ```

  The following error message appears:

  ```
  Semantic analysis exception - Invalid function : line 1:41 'ifnull'
  ```

- Cause

  MaxCompute does not support the `IFNULL` function.

- Solution

  Use the `CASE WHEN` expression or the `COALESCE` function. For more information, see CASE WHEN expression or COALESCE.

How do I concatenate all fields that match a specified condition in a JSON string?

MaxCompute SQL queries data that matches the specified conditions by using filter conditions, such as `like`. After MaxCompute SQL obtains the query results, MaxCompute SQL uses the ARRAY or MAP function to construct the data as a complex data type such as MAP or ARRAY, and then uses the TO_JSON function to aggregate the data.

How do I use each key in a JSON string as a field?

You can use the GET_JSON_OBJECT function to extract fields from a JSON string.

How do I convert a JSON string into an array?

You can use the FROM_JSON function in an SQL statement to perform a data type conversion, such as `select from_json(<col_name>, "array<bigint>");`.

Which function of MaxCompute is similar to the IFNULL function of MySQL?

The IFNULL function of MySQL is similar to the NVL function of MaxCompute. For more information about the mappings between MaxCompute built-in functions and Hive, MySQL, and Oracle functions, see Mappings between built-in functions of MaxCompute and built-in functions of Hive, MySQL, and Oracle.

How do I convert one row of data into multiple rows of data?

You can use the TRANS_COLS function to convert one row of data into multiple rows of data.

What do I do if the "Expression not in GROUP BY key" error message appears when I use the COALESCE function?

- Problem description

  The following error message appears when more than one expression is used in the COALESCE function:

  ```
  FAILED: ODPS-0130071:Semantic analysis exception - Expression not in GROUP BY key : line
  8:9 "$.table"
  ```

  Sample statement:

  ```
  select
  md5(concat(aid,bid)) as id
  ,aid
  , bid
  , sum(amountdue) as amountdue
  , coalesce(
  sum(regexp_count(get_json_object(extended_x, '$.table.tableParties'), '{')),
  decode(get_json_object(extended_x, '$.table'), null, 0, 1)
  ) as tableparty
  , decode(sum(headcount),null,0,sum(headcount) ) as headcount
  , 'a' as pt
  from e_orders
  where pt='20170425'
  group by aid, bid;
  ```

- Cause

  A required grouping field is not specified for GROUP BY.

- Solution

  Specify the following expression after GROUP BY. The expression returns a field.

```
coalesce(
sum(regexp_count(get_json_object(extended_x, '$.table.tableParties'), '{')),
decode(get_json_object(extended_x, '$.table'), null, 0, 1)
) as tableparty
, decode(sum(headcount),null,0,sum(headcount) ) as headcount
```

What do I do if an implicit data type conversion error appears when I use a built-in function of MaxCompute?

If you specify `odps.sql.type.system.odps2=true` to enable the MaxCompute V2.0 data type edition for a MaxCompute project, implicit conversions between the following data types are disabled and a loss of precision or an error may occur:

- STRING->BIGINT
- STRING->DATETIME
- DOUBLE->BIGINT
- DECIMAL->DOUBLE
- DECIMAL->BIGINT

In this case, you can use the CAST function to perform a forced conversion or specify `odps.sql.type.system.odps2=false` to disable the MaxCompute V2.0 data type edition.

3.10. UDF

3.10.1. Overview

MaxCompute offers a variety of built-in functions to meet your business requirements. If the built-in functions do not meet your business requirements, you can write code to create user-defined functions (UDFs). This topic describes the types, scenarios, development process, and usage notes of UDFs that are supported in MaxCompute.

Background information

In the broad sense, UDFs include user-defined scalar functions, user-defined aggregation functions (UDAFs), and user-defined table-valued functions (UDTFs). In the narrow sense, UDFs refer only to user-defined scalar functions. The following table describes the types of MaxCompute UDFs.

UDF type	Scenario
UDF	A one-to-one mapping is established between the input and output data of a UDF. Each time a UDF reads a row of data, it returns an output value.
UDTF	A one-to-many mapping is established between the input and output data of a UDTF. Each time a UDTF reads a row of data, it returns multiple values, which are considered a table.
UDAF	A many-to-one mapping is established between the input and output data of a UDAF. Multiple input records are aggregated to generate one output value.

In addition to the preceding UDFs, MaxCompute offers the following UDFs for special scenarios.

UDF type	Scenario
Code-embedded UDFs	If you want to simplify the development process of MaxCompute UDFs and view the code logic, you can embed Java or Python code into SQL scripts.
SQL functions	If your code contains duplicate code, you can use SQL UDFs to improve the code reuse rate and simplify the development process.
Open source geospatial UDFs	You can use Hive geospatial functions to analyze spatial data in MaxCompute.

Limits

You cannot access the Internet by using UDFs. If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.

Usage notes

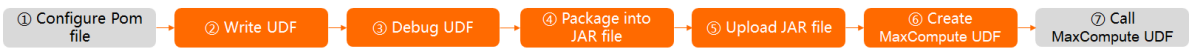
Before you use UDFs, take note of the following items:

- UDFs cannot compete with built-in functions in performance. We recommend that you preferentially use built-in functions to implement your business logic.
- If you use a UDF in SQL statements, the memory usage of a computing job may exceed the default allocated memory size if a large amount of data is computed and data skew occurs. In this case, you can run the `set odps.sql.udf.joiner.jvm.memory=xxxx;` command at the session level to resolve the issue. For more information about the MaxCompute UDF FAQ, see FAQ about MaxCompute UDFs.
- If the name of a UDF is the same as that of a built-in function, the UDF is preferentially called. For example, if UDF CONCAT and built-in function CONCAT both exist in MaxCompute, the system automatically calls UDF CONCAT instead of the built-in function CONCAT. If you want to call the built-in function, you must add the symbol `::` before the built-in function, for example, `select ::concat('ab', 'c');`.

Development process

This section describes how to develop a UDF, UDTF, or UDAF. The development process of Code-embedded UDFs, SQL functions, and Open source geospatial UDFs is different. For more information, see the related documentation.

- The following figure demonstrates how to write a MaxCompute UDF in Java.



No.	Required	Description	Platform	References

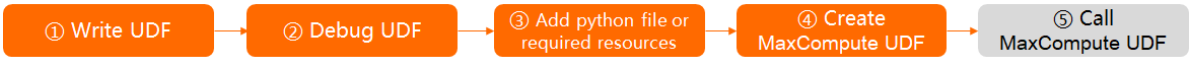
No.	Required	Description	Platform	References

1	No	Before you can use Maven to write code, you must add the related SDK dependencies to the POM file. This ensures that the code can be compiled. The following SDK dependency shows an example:
		`<dependency>`
		`<groupId>com.aliyun.odps</groupId>`
		`<artifactId>odps-sdk-udf</artifactId>`
		`<version>0.29.10-public</version>`
		`</dependency>`
		You can search for `odps-sdk-udf` from Maven repositories to obtain the version of the SDK dependency.
			IntelliJ IDEA (Maven)	None

2	Yes	Write a UDF based on your business requirements.	IntelliJ IDEA (Maven) and MaxCompute Studio	Develop a UDF in Java
3	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		
4	Yes	Debug the UDF code to ensure that the code is packaged into a JAR file after it is successfully run on your on-premises machine.		

No.	Required	Description	Platform	References
5	Yes	Upload the JAR file as a resource to your MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Package a Java program, upload the package, and create a MaxCompute UDF ◦ DataWorks ▪ Create a JAR resource ▪ Register a MaxCompute function
6	Yes	Create a UDF based on the JAR file that you uploaded.		
7	No	Call the UDF in the query data code.		None

- The following figure demonstrates how to write a MaxCompute UDF in Python.



No.	Required	Description	Platform	References
1	Yes	Write a UDF based on your business requirements.	MaxCompute Studio	Develop a Python UDF
2	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		

No.	Required	Description	Platform	References
3	Yes	Upload Python files or required resources, such as file resources, table resources, and third-party packages, to a MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Upload a Python program and create a MaxCompute UDF ◦ DataWorks Create a Python resource and register a UDF
4	Yes	Create a UDF based on the uploaded Python files or required resources.		
5	No	Call the UDF in the query data code.		None

Instructions

Use the following methods to call UDFs:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

MaxCompute SDK

The following table describes the SDKs provided by MaxCompute. For more information about the packages included in each SDK and the classes in the packages, see MaxCompute SDK.

SDK name	Description
odps-sdk-core	Provides classes for managing basic resources of MaxCompute.
odps-sdk-commons	Provides common Utils for Java.
odps-sdk-udf	Provides UDFs.
odps-sdk-mapred	Provides the MapReduce API.
odps-sdk-graph	Provides the Graph API.

3.10.2. UDF

3.10.2.1. Overview

MaxCompute allows you to write user-defined functions (UDFs) in Java or Python to extend the capabilities of MaxCompute functions and accommodate your business requirements. This topic describes the types, limits, usage notes, and development process of UDFs. This topic also describes how to use UDFs.

Background information

UDFs are suitable for scenarios in which the input and output have a one-to-one mapping. If a UDF is called, the UDF reads one row of data and returns a value. MaxCompute supports two types of UDFs based on the programming language: Java UDFs and Python UDFs. The following table describes the two types of UDFs.

UDF type	Description
Java UDF	This type of UDF is written in Java to implement the function logic. For more information, see Java UDFs.
Python UDF	This type of UDF is written in Python to implement the function logic. Python UDFs are classified into Python 2 UDFs and Python 3 UDFs. • Python 2 UDFs: Python 2.7 is used. For more information, see Python 2 UDFs. • Python 3 UDFs: CPython 3.7.3 is used. For more information, see Python 3 UDFs.

Limits

You cannot access the Internet by using UDFs. If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.

Usage notes

Before you use UDFs, take note of the following items:

- UDFs cannot compete with built-in functions in performance. We recommend that you preferentially use built-in functions to implement your business logic.
- If you use a UDF in SQL statements, the memory usage of a computing job may exceed the default allocated memory size if a large amount of data is computed and data skew occurs. In this case, you can run the `set odps.sql.udf.joiner.jvm.memory=xxxx;` command at the session level to resolve the issue. For more information about the MaxCompute UDF FAQ, see FAQ about MaxCompute UDFs.
- If the name of a UDF is the same as that of a built-in function, the UDF is preferentially called. For example, if UDF CONCAT and built-in function CONCAT both exist in MaxCompute, the system automatically calls UDF CONCAT instead of the built-in function CONCAT. If you want to call the built-in function, you must add the symbol `::` before the built-in function, for example, `select ::concat('ab', 'c');`.

Development process

This section describes the development process of a UDF.

- The following figure demonstrates how to write a MaxCompute UDF in Java.

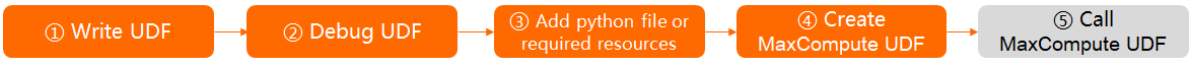
No.	Required	Description	Platform	References

1	No	Before you can use Maven to write code, you must add the related SDK dependencies to the POM file. This ensures that the code can be compiled. The following SDK dependency shows an example:
		`<dependency>`
		`<groupId>com.aliyun.odps</groupId>`
		`<artifactId>odps-sdk-udf</artifactId>`
		`<version>0.29.10-public</version>`
		`</dependency>`
		You can search for `odps-sdk-udf` from Maven repositories to obtain the version of the SDK dependency.
			IntelliJ IDEA (Maven)	None

2	Yes	Write a UDF based on your business requirements.	IntelliJ IDEA (Maven) and MaxCompute Studio	Develop a UDF in Java
3	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		
4	Yes	Debug the UDF code to ensure that the code is packaged into a JAR file after it is successfully run on your on-premises machine.		

No.	Required	Description	Platform	References
5	Yes	Upload the JAR file as a resource to your MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Package a Java program, upload the package, and create a MaxCompute UDF ◦ DataWorks ▪ Create a JAR resource ▪ Register a MaxCompute function
6	Yes	Create a UDF based on the JAR file that you uploaded.		
7	No	Call the UDF in the query data code.		None

- The following figure demonstrates how to write a MaxCompute UDF in Python.



No.	Required	Description	Platform	References
1	Yes	Write a UDF based on your business requirements.	MaxCompute Studio	Develop a Python UDF
2	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		

No.	Required	Description	Platform	References
3	Yes	Upload Python files or required resources, such as file resources, table resources, and third-party packages, to a MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Upload a Python program and create a MaxCompute UDF ◦ DataWorks Create a Python resource and register a UDF
4	Yes	Create a UDF based on the uploaded Python files or required resources.		
5	No	Call the UDF in the query data code.		None

Instructions

Use the following methods to call UDFs:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

3.10.2.2. Java UDFs

This topic describes how to write a user-defined function (UDF) in Java.

UDF code structure

You can use IntelliJ IDEA (Maven) or MaxCompute Studio to write a UDF in Java. The UDF code can contain the following information:

- Java package: optional.

  You can package Java classes that are defined into a JAR file. This helps you find and use these classes.

- Base UDF class: required.

The required UDF class is `com.aliyun.odps.udf.UDF`. If you want to use other UDF classes or complex data types, follow the instructions provided in MaxCompute SDK to add the required classes. For example, the UDF class that corresponds to the STRUCT data type is `com.aliyun.odps.data.Struct`.

- `@Resolve` annotation: optional.

  The annotation is in the `@Resolve(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return value of a UDF. If you want to use the STRUCT data type for a UDF, you cannot use the reflection feature for the `com.aliyun.odps.data.Struct` class to obtain the names and types of fields. In this case, you must add the `@Resolve` annotation to the com.aliyun.odps.data.Struct class. This `annotation` affects only the overloading of the UDF whose input parameters or return value contain the com.aliyun.odps.data.Struct class. Example: `@Resolve("struct<a:string>,string->string")`. For more information about how to use complex data types in Java UDFs, see Use complex data types in Java UDFs.

- Custom Java class: required.

  A custom Python class is the organizational unit of UDF code. This class defines the variables and methods that are used to meet your business requirements.

- `evaluate` method: required.

  The evaluate method is a non-static public method and is contained in a custom Java class. The data types of the input parameters and return value of the `evaluate` method are used as the function signature of a UDF in SQL statements. The function signature defines the data types of the input parameters and return value of the UDF.

  You can implement multiple `evaluate` methods in a UDF. When you call a UDF, MaxCompute matches an `evaluate` method based on the data types of the UDF input parameters.

  When you write a Java UDF, you can use Java data types or Java writable data types. For more information about the mappings between the data types supported in MaxCompute projects, Java data types, and Java writable data types, see Data types.

- UDF initialization or termination code: optional. You can use the `void setup(ExecutionContext ctx)` method to initialize a UDF and use the `void close()` method to terminate a UDF. The `void setup(ExecutionContext ctx)` method is called before the `evaluate` method is called. The void setup(ExecutionContext ctx) method is called only once and is used to initialize the resources that are required for data computing or initialize the members of a class. The `void close()` method is called after the `evaluate` method is called. The void close() method is used to clean up data, such as closing files.

Sample UDF code:

- Use Java data types

```
// Package the defined Java classes into a file named org.alidata.odps.udf.examples.
package org.alidata.odps.udf.examples;
// Inherit the UDF class.
import com.aliyun.odps.udf.UDF;
// The custom Java class.
public final class Lower extends UDF {
// The evaluate method. String indicates the data types of the input parameters and retur
n indicates the return value.
    public String evaluate(String s) {
        if (s == null) {
        return null;
    }
        return s.toLowerCase();
  }
}
```

- Use Java writable data types

```
// Package the defined Java classes into a file named com.aliyun.odps.udf.example.
package com.aliyun.odps.udf.example;
// Add the class that corresponds to a Java writable data type.
import com.aliyun.odps.io.Text;
// Inherit the UDF class.
import com.aliyun.odps.udf.UDF;
// The custom Java class.
public class MyConcat extends UDF {
  private Text ret = new Text();
// The evaluate method. Text indicates the data types of the input parameters and return
indicates the return value.
  public Text evaluate(Text a, Text b) {
      if (a == null || b == null) {
      return null;
    }
      ret.clear();
      ret.append(a.getBytes(), 0, a.getLength());
      ret.append(b.getBytes(), 0, b.getLength());
      return ret;
  }
}
```

MaxCompute also allows you to use Hive UDFs whose Hive version is compatible with MaxCompute. For more information, see Hive UDFs.

Limits

You cannot access the Internet by using UDFs. If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.

Usage notes

Before you write a Java UDF, take note of the following points:

- We recommend that the JAR files of different UDFs do not contain the classes that have the same name but different logic. For example, the JAR file of UDF 1 is named udf1.jar and the JAR file of UDF 2 is named udf2.jar. Both files contain a class named `com.aliyun.UserFunction.class`, but the class has different logic in the files. If UDF 1 and UDF 2 are called in the same SQL statement, MaxCompute loads the com.aliyun.UserFunction.class from one of the two files. As result, the UDFs cannot run as expected and a compilation error may occur.
- The data types of the input parameters or return value of a Java UDF are objects. The first letters of these data types must be capitalized, such as String.
- NULL values in MaxCompute SQL are represented by NULL in Java. NULL values in MaxCompute SQL cannot be represented by values of primitive data types in Java and cannot be used.

Data types

In MaxCompute, different data type editions support different data types. In MaxCompute V2.0 and later, more data types and complex data types, such as ARRAY, MAP, and STRUCT, are supported. For more information about MaxCompute data type editions, see Data type editions.

The following table describes the mappings between the data types supported in MaxCompute projects, Java data types, and Java writable data types. You must write Java UDFs based on the mappings to ensure the consistency of the data types.

MaxCompute data type	Java data type	Java writable data type
TINYINT	java.lang.Byte	ByteWritable
SMALLINT	java.lang.Short	ShortWritable
INT	java.lang.Integer	IntWritable
BIGINT	java.lang.Long	LongWritable
FLOAT	java.lang.Float	FloatWritable
DOUBLE	java.lang.Double	DoubleWritable
DECIMAL	java.math.BigDecimal	BigDecimalWritable
BOOLEAN	java.lang.Boolean	BooleanWritable
STRING	java.lang.String	Text
VARCHAR	com.aliyun.odps.data.Varchar	VarcharWritable
BINARY	com.aliyun.odps.data.Binary	BytesWritable
DATETIME	java.util.Date	DatetimeWritable
TIMESTAMP	java.sql.Timestamp	TimestampWritable
INTERVAL_YEAR_MONTH	N/A	IntervalYearMonthWritable
INTERVAL_DAY_TIME	N/A	IntervalDayTimeWritable
ARRAY	java.util.List	N/A

MaxCompute data type	Java data type	Java writable data type
MAP	java.util.Map	N/A
STRUCT	com.aliyun.odps.data.Struct	N/A

For more information about how to use complex data types in Java UDFs, see Use complex data types in Java UDFs.

> **Note** The input parameters or return value of a UDF can be of Java writable data types only if your MaxCompute project uses the MaxCompute V2.0 data type edition.

Instructions

After you develop a Java UDF, you can use MaxCompute SQL to call this UDF. For more information about how to develop a Java UDF, see Development process. The following steps describe how to call the UDF:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute Studio to develop and call a Java UDF, see Example.

Hive UDFs

If your MaxCompute project uses the MaxCompute V2.0 data type edition and supports Hive UDFs, you can directly use Hive UDFs whose Hive version is compatible with MaxCompute.

The Hive version that is compatible with MaxCompute is 2.1.0, which corresponds to Hadoop 2.7.2. If a UDF is developed on another Hive or Hadoop version, you must use Hive 2.1.0 or Hadoop 2.7.2 to recompile the JAR file of the UDF.

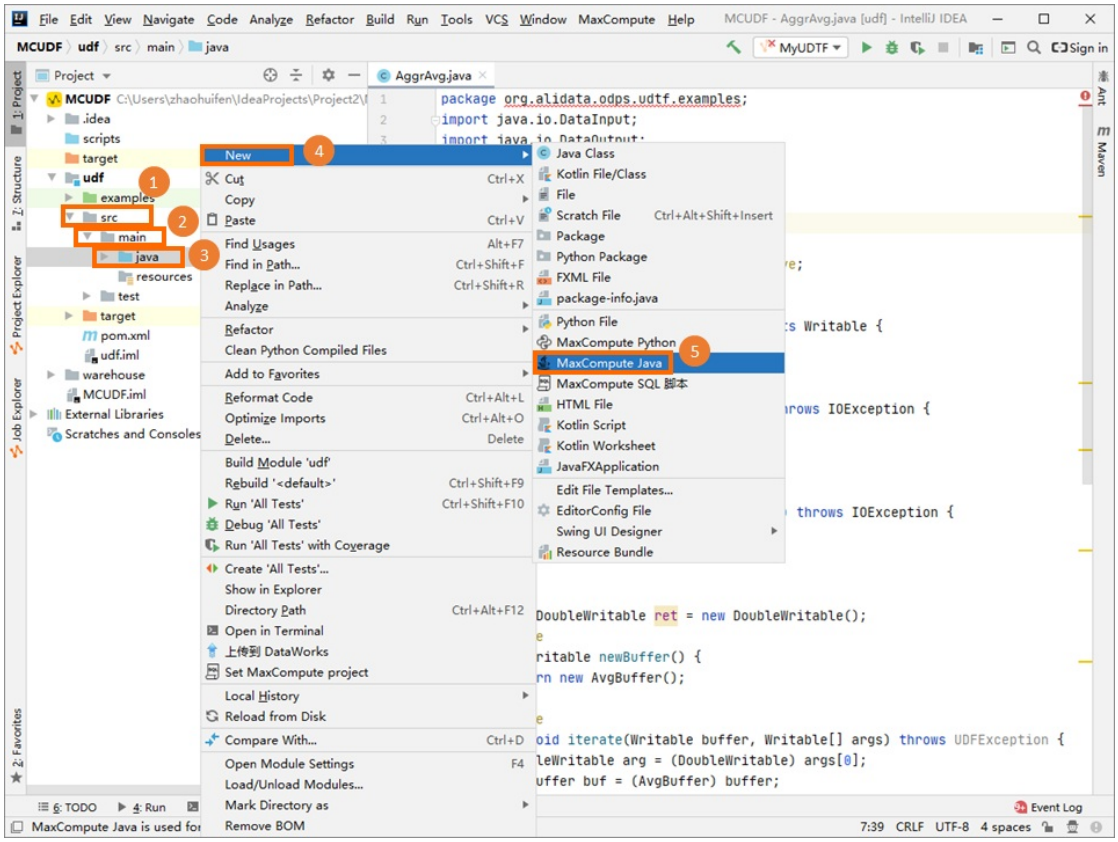
For more information about how to use Hive UDFs in MaxCompute, see Write a Hive UDF in Java.

Example

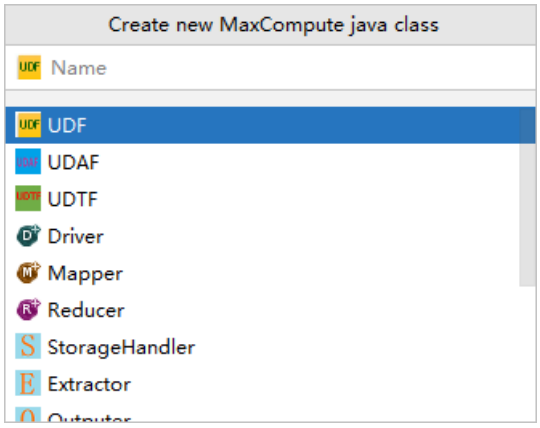
This example demonstrates how to use MaxCompute Studio to develop and call a Java UDF that is used to convert all letters to lowercase letters.

1. Make the following preparations on IntelliJ IDEA:
   i. Install MaxCompute Studio.
   ii. Establish a connection to a MaxCompute project.
   iii. Create a MaxCompute Java module.
2. Write UDF code.

i. In the left-side navigation pane of the **Project** tab, choose **src > main > java**, right-click java, and then choose **New > MaxCompute Java**.

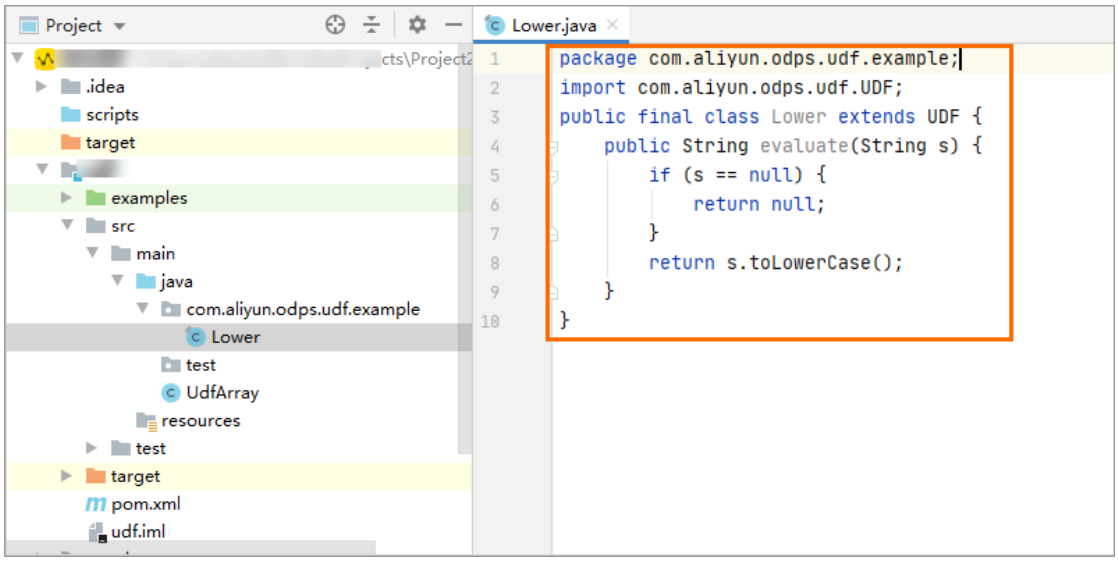


ii. In the **Create new MaxCompute java class** dialog box, click **UDF**, enter a class name in the **Name** field, and then press Enter. The class is named Lower in this example.



Name: the name of the MaxCompute Java class. If you have not created a package, enter packagename.classname. The system automatically generates a package.

iii. Write code in the code editor.



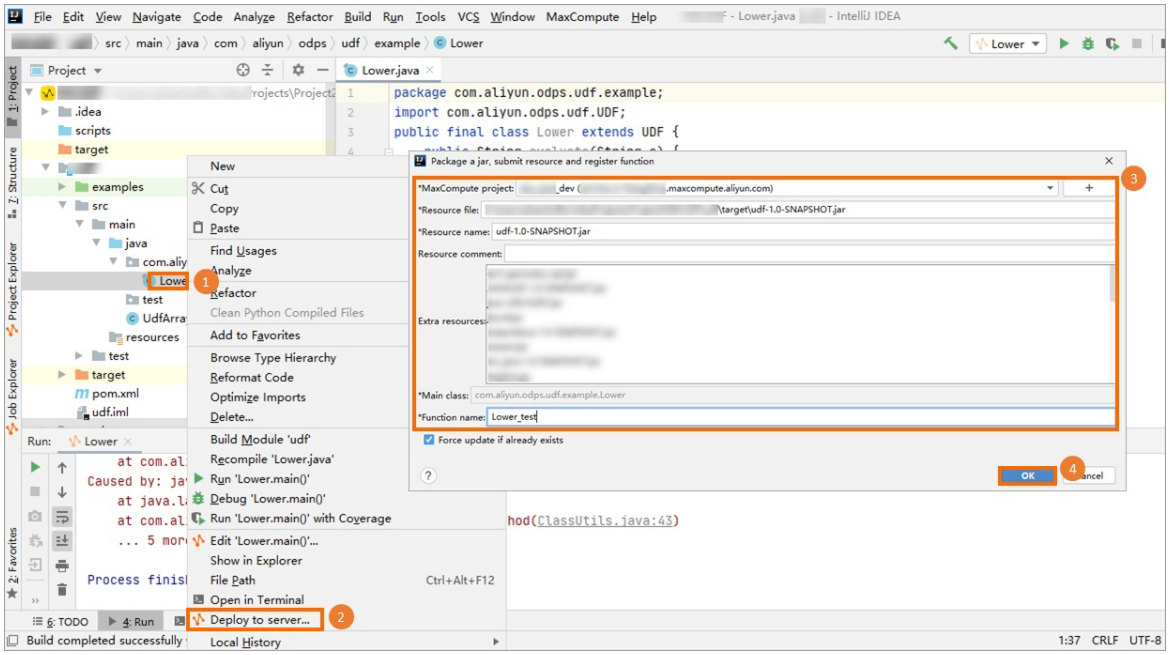
Sample UDF code:

```
package <packagename>;
import com.aliyun.odps.udf.UDF;
public final class Lower extends UDF {
    public String evaluate(String s) {
        if (s == null) {
           return null;
        }
           return s.toLowerCase();
    }
}
```

> **Note** Debug the Java UDF on your on-premises machine if necessary. For more information, see Develop and debug UDFs.

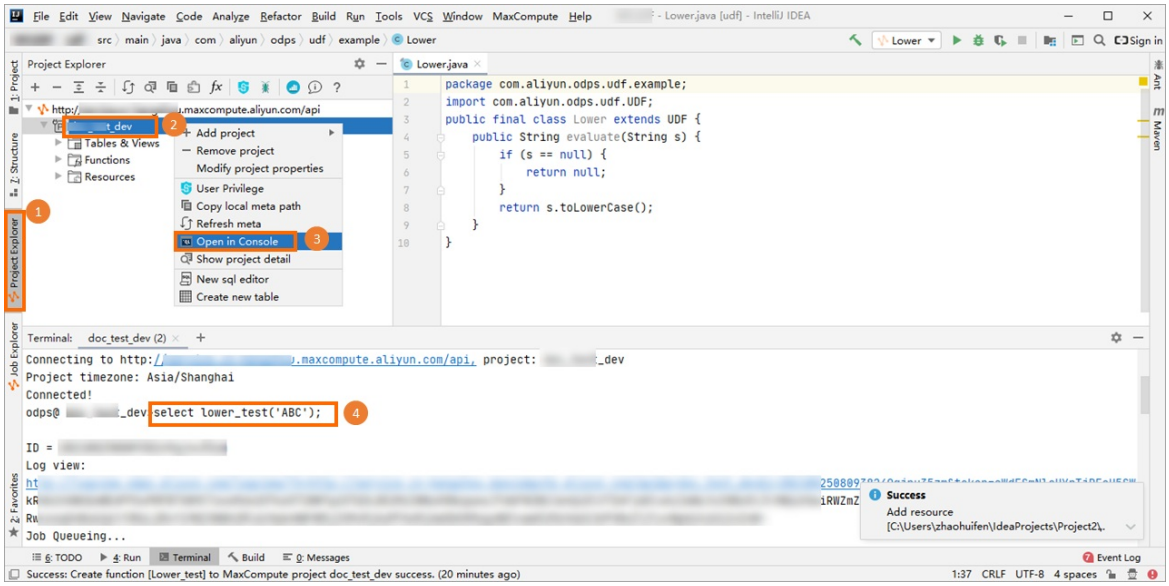
3. Create the Java UDF.

Right-click the JAR file of the UDF and select **Deploy to server...**. In the **Package a jar, submit resource and register function** dialog box, configure the parameters and click **OK**.



- **MaxCompute project**: the name of the MaxCompute project to which the UDF belongs. Retain the default value, which indicates that the connection to the MaxCompute project is established when you write the UDF.
- **Resource file**: the path of the resource file on which the UDF depends. Retain the default value.
- **Resource name**: the name of the resource on which the UDF depends. Retain the default value.
- **Function name**: the name of the UDF that you want to create. This name is used in the SQL statements that are used to call this UDF. Example: Lower_test.

4. In the left-side navigation pane, click the **Project Explore** tab. Right-click the MaxCompute project to which the UDF belongs, select **Open in Console**, enter the SQL statement for calling the UDF, and then press Enter to execute the SQL statement.



Sample statement:

```
select lower_test('ABC');
```

Returned results:

```
+------+
| _c0 |
+------+
| abc |
+------+
```

3.10.2.3. Python UDF

3.10.2.3.1. Python 2 UDFs

The Python 2 version that is used by MaxCompute is Python 2.7. This topic describes how to write a user-defined function (UDF) in Python 2.

UDF code structure

You can use MaxCompute Studio to write UDF code in Python 2. The UDF code can contain the following information:

- Encoding declaration: optional.

  The declaration format is `#coding:utf-8` or `# -*- coding: utf-8 -*-`. The two formats are equivalent. If Chinese characters appear in UDF code that is written in Python 2, an error is returned when you run the UDF. To address this issue, you must add an encoding declaration to the header of the code.

- Module import: required.

  UDF code must include `from odps.udf import annotate`, which is used to import the function signature module. This way, MaxCompute can identify the function signature that is defined in the code. If you want to reference files or tables in UDF code, the UDF code must include `from odps.distcache import get_cache_file` or `from odps.distcache import get_cache_table`.

- Function signature: required.

  The function signature is in the `@annotate(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return value of the UDF. For more information about function signatures, see Function signatures and data types.

- Custom Python class: required.

  A custom Python class is the organizational unit of UDF code. This class defines the variables and methods that are used to meet your business requirements. In UDF code, you can also reference third-party libraries that are installed in MaxCompute or reference files or tables. For more information, see Third-party libraries or Reference resources.

- `evaluate` method: required.

  The evaluate method is contained in the custom Python class. The `evaluate` method defines the input parameters and return value of the UDF. Each Python class can contain only one `evaluate` method.

Sample code:

```
#coding:utf-8
# Import the function signature module.
from odps.udf import annotate
# The function signature.
@annotate("bigint,bigint->bigint")
# The custom Python class.
class MyPlus(object):
# The evaluate method.
   def evaluate(self, arg0, arg1):
       if None in (arg0, arg1):
           return None
       return arg0 + arg1
```

Limits

MaxCompute allows you to write Python 2 UDFs in Python 2.7 and run the UDF code in a sandbox environment. In this environment, the following operations are prohibited:

- Read data from and write data to local files.
- Start subprocesses.
- Start threads.
- Enable socket communication.
- Use other systems to call Python 2 UDFs.

Due to these limits, the code that you upload must be written by using Python standard libraries. If modules or C extension modules in Python standard libraries are involved in the preceding operations, these modules cannot be used. Take note of the following points about modules in Python standard libraries:

- All the modules that are implemented based on Python standard libraries and do not depend on extension modules are available.
- The following C extension modules are available:
  - array and audioop
  - binascii and bisect
  - cmath, _codecs_cn, _codecs_hk, _codecs_iso2022, _codecs_jp, _codecs_kr, _codecs_tw, _collections, and cStringIO
  - datetime
  - _functools and future_builtins
  - _heapq and _hashlib
  - itertools
  - _json
  - _locale and _lsprof
  - math, _md5, and _multibytecodec
  - operator
  - _random
  - _sha256, _sha512, _sha, _struct, and strop

- time
- unicodedata
- _weakref
- cPickle

- When you run UDF code in a sandbox environment, the maximum size of data that can be written to the standard output (sys.stdout) or standard error output (`sys.stderr`) is 20 KB. If the size exceeds 20 KB, extra characters are ignored.

Third-party libraries

Third-party libraries, such as NumPy, are installed in the Python 2 environment of MaxCompute as supplements to standard libraries.

> **Note** The use of third-party libraries is subject to some limits. For example, when you use a third-party library, you are not allowed to access local data and you can use only limited network I/O resources. The related APIs in the third-party libraries are disabled.

Function signatures and data types

Format of function signatures:

```
@annotate(<signature>)
```

The `signature` parameter is a string that specifies the data types of input parameters and return value. When you run a UDF, the data types of the input parameters and return value of the UDF must be consistent with the data types specified in the function signature. The data type consistency is checked during semantic parsing. If the data types are inconsistent, an error is returned. Format of a signature:

```
'arg_type_list -> type'
```

Parameter description:

- `arg_type_list` : specifies the data types of input parameters. If multiple input parameters are used, their data types are separated by commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, and VARCHAR. Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

  `arg_type_list` can be represented by an asterisk (*) or left empty ('').

  - If `arg_type_list` is represented by an asterisk (*), a random number of input parameters are allowed.
  - If `arg_type_list` is left empty (''), no input parameters are used.
- `type` : specifies the data types of return value. For a UDF, only one column of values is returned. The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, and DECIMAL(precision,scale). Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

> **Note** When you write UDF code, you can select a data type based on the data type edition used by your MaxCompute project. For more information about MaxCompute data type editions and the data types supported in each edition, see Data type editions.

The following table provides examples of valid function signatures.

Function signature	Description
`'bigint,double->string'`	The data types of the input parameters are BIGINT and DOUBLE and the data type of the return value is STRING.
`'*->string'`	A random number of input parameters are used and the data type of the return value is STRING.
`'->double'`	No input parameters are used and the data type of the return value is DOUBLE.
`'array<bigint>->struct<x:string, y:int>'`	The data type of the input parameters is ARRAY<BIGINT> and the data type of the return value is STRUCT<x:STRING, y:INT>.
`'->map<bigint, string>'`	No input parameters are used and the data type of the return value is MAP<BIGINT, STRING>.

The following table describes the mappings between the data types that are supported in MaxCompute SQL and the Python 2 data types. You must write Python UDFs based on the mappings to ensure the consistency of data types.

MaxCompute SQL data type	Python 2 data type
BIGINT	INT
STRING	STR
DOUBLE	FLOAT
BOOLEAN	BOOL
DATETIME	INT
FLOAT	FLOAT
CHAR	STR
VARCHAR	STR
BINARY	BYTEARRAY
DATE	INT
DECIMAL	DECIMAL.DECIMAL

MaxCompute SQL data type	Python 2 data type
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

> **Note**
>
> - The DATETIME type supported in MaxCompute SQL is mapped to the Python data type INT. A value of the INT type follows the UNIX format, which is the number of milliseconds that have elapsed since 00:00:00 Thursday, January 1, 1970. You can process data of the DATETIME type by using the DATETIME module in Python standard libraries.
> - The `silent` parameter is added to `odps.udf.int(value)`. If the `silent` parameter is set to True and the data type of `value` cannot be converted into the INT type, None is returned, and no error is returned.
> - NULL in MaxCompute SQL is mapped to None in Python 2.

Reference resources

You can reference files or tables in Python 2 UDF code by using the `odps.distcache` module.

- `odps.distcache.get_cache_file(resource_name)`: returns the content of a specified file.
  - `resource_name` is a string that specifies the name of an existing file in your MaxCompute project. If the file name is invalid or the file does not exist, an error is returned.

    > **Note** To reference a file in UDF code, you must declare the file when you create the UDF. Otherwise, an error is returned when you call the UDF.

  - The return value is a file-like object. If this object is no longer used, you must call the `close` method to release the open file.

  The following code shows how to reference a file.

```
from odps.udf import annotate
from odps.distcache import get_cache_file
@annotate('bigint->string')
class DistCacheExample(object):
def __init__(self):
    cache_file = get_cache_file('test_distcache.txt')
    kv = {}
    for line in cache_file:
        line = line.strip()
        if not line:
            continue
        k, v = line.split()
        kv[int(k)] = v
    cache_file.close()
    self.kv = kv
def evaluate(self, arg):
    return self.kv.get(arg)
```

- `odps.distcache.get_cache_table(resource_name)`: returns the content of a specified table.
  - `resource_name` is a string that specifies the name of an existing table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned.
  - The return value is of the GENERATOR data type. The caller traverses the table to obtain the table content. A record of the ARRAY type is obtained each time the caller traverses the table.

  The following code shows how to reference a table.

```
from odps.udf import annotate
from odps.distcache import get_cache_table
@annotate('->string')
class DistCacheTableExample(object):
    def __init__(self):
        self.records = list(get_cache_table('udf_test'))
        self.counter = 0
        self.ln = len(self.records)
    def evaluate(self):
        if self.counter > self.ln - 1:
            return None
        ret = self.records[self.counter]
        self.counter += 1
        return str(ret)
```

Usage notes

After you develop a Python 2 UDF, you can use MaxCompute SQL to call this UDF. For more information about how to develop a Python 2 UDF, see Development process. You can call a Python 2 UDF by using one of the following methods:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute client to develop and call a Python 2 UDF, see Develop a Python UDF.

3.10.2.3.2. Python 3 UDFs

Python Software Foundation announced the End of Life (EOL) for Python 2. Due to this reason, MaxCompute supports Python 3 and uses CPython 3.7.3. This topic describes how to write a user-defined function (UDF) in Python 3.

UDF code structure

You can use MaxCompute Studio to write UDF code in Python 3. The UDF code can contain the following information:

- Module import: required.

  UDF code must include `from odps.udf import annotate`, which is used to import the function signature module. This way, MaxCompute can identify the function signature that is defined in the code. If you want to reference files or tables in UDF code, the UDF code must include `from odps.distcache import get_cache_file` or `from odps.distcache import get_cache_table`.

- Function signature: required.

  The function signature is in the `@annotate(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return value of the UDF. For more information about function signatures, see Function signatures and data types.

- Custom Python class: required.

  A custom Python class is the organizational unit of UDF code. This class defines the variables and methods that are used to meet your business requirements. In UDF code, you can also reference third-party libraries that are installed in MaxCompute or reference files or tables. For more information, see Third-party libraries or Reference resources.

- `evaluate` method: required.

  The evaluate method is contained in the custom Python class. The `evaluate` method defines the input parameters and return value of the UDF. Each Python class can contain only one `evaluate` method.

Sample code:

```
# Import the function signature module.
from odps.udf import annotate
# The function signature.
@annotate("bigint,bigint->bigint")
# The custom Python class.
class MyPlus(object):
# The evaluate method.
   def evaluate(self, arg0, arg1):
       if None in (arg0, arg1):
           return None
       return arg0 + arg1
```

> **Note** Python 2 UDFs and Python 3 UDFs differ in terms of the underlying Python version. You can write a UDF based on the capability of the Python version that you use.

Limits

Python 3 is incompatible with Python 2. Due to this reason, you cannot use Python 2 code and Python 3 code in a single SQL statement at the same time.

Port Python 2 UDFs

Python Software Foundation announced the EOL for Python 2. Therefore, we recommend that you port Python 2 UDFs. The method used to port Python 2 UDFs varies based on the types of MaxCompute projects.

- In a new project or an existing project for which you write UDFs in Python for the first time, we recommend that you use Python 3 to write all Python UDFs.
- In an existing project where a large number of Python 2 UDFs exist, proceed with caution when you enable Python 3. If you want to replace Python 2 UDFs with Python 3 UDFs, use the following methods:
  - Use Python 3 to write new UDFs and enable Python 3 for new jobs at the session level. For more information about how to enable Python 3, see Enable Python 3.
  - Rewrite Python 2 UDFs in a manner in which the UDFs are compatible with Python 2 and Python 3. For more information about how to rewrite UDFs, see Porting Python 2 Code to Python 3.

    > **Note** If you want to write a public UDF that is shared among multiple projects, we recommend that you use a UDF that is compatible with Python 2 and Python 3.

Enable Python 3

By default, Python 2 is used to write UDFs in a MaxCompute project. If you want to write UDFs in Python 3, add the following command before the SQL statement that you want to execute. Then, commit and execute the statement.

```
set odps.sql.python.version=cp37;
```

Third-party libraries

NumPy is not installed in the Python 3 runtime environment in MaxCompute. To use a NumPy UDF, you must manually upload a NumPy wheel package. If you obtain this package from Python Package Index (PyPI) or an image, the package is named *numpy-<Version>-cp37-cp37m-manylinux1_x86_64.whl*. For more information about how to upload a file, see Resource operations or Reference third-party packages in Python UDFs.

For more information about standard libraries that are supported by Python 3, see The Python Standard Library.

Function signatures and data types

Format of function signatures:

```
@annotate(<signature>)
```

The `signature` parameter is a string that specifies the data types of input parameters and return value. When you run a UDF, the data types of the input parameters and return value of the UDF must be consistent with the data types specified in the function signature. The data type consistency is checked during semantic parsing. If the data types are inconsistent, an error is returned. Format of a signature:

```
'arg_type_list -> type'
```

Parameter description:

- `arg_type_list`: specifies the data types of input parameters. If multiple input parameters are used, their data types are separated by commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, and VARCHAR. Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

  `arg_type_list` can be represented by an asterisk (*) or left empty ('').

  - If `arg_type_list` is represented by an asterisk (*), a random number of input parameters are allowed.
  - If `arg_type_list` is left empty (''), no input parameters are used.

- `type`: specifies the data types of return value. For a UDF, only one column of values is returned. The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, and DECIMAL(precision,scale). Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

> **Note** When you write UDF code, you can select a data type based on the data type edition used by your MaxCompute project. For more information about MaxCompute data type editions and the data types supported in each edition, see Data type editions.

The following table provides examples of valid function signatures.

Function signature	Description
`'bigint,double->string'`	The data types of the input parameters are BIGINT and DOUBLE and the data type of the return value is STRING.
`'*->string'`	A random number of input parameters are used and the data type of the return value is STRING.
`'->double'`	No input parameters are used and the data type of the return value is DOUBLE.
`'array<bigint>->struct<x:string, y:int>'`	The data type of the input parameters is ARRAY<BIGINT> and the data type of the return value is STRUCT<x:STRING, y:INT>.
`'->map<bigint, string>'`	No input parameters are used and the data type of the return value is MAP<BIGINT, STRING>.

The following table describes the mappings between the data types that are supported in MaxCompute SQL and the Python 2 data types. You must write Python UDFs based on the mappings to ensure the consistency of data types.

MaxCompute SQL data type	Python 3 data type
BIGINT	INT
STRING	UNICODE
DOUBLE	FLOAT
BOOLEAN	BOOL
DATETIME	DATETIME.DATETIME
FLOAT	FLOAT
CHAR	UNICODE
VARCHAR	UNICODE
BINARY	BYTES
DATE	DATETIME.DATE
DECIMAL	DECIMAL.DECIMAL
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

Reference resources

You can reference files or tables in Python 2 UDF code by using the `odps.distcache` module.

- `odps.distcache.get_cache_file(resource_name, mode)`: returns the content of a specified file based on the value of `mode` that you specified.
  - `resource_name` is a string that specifies the name of an existing table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned.
  - The value of `mode` is of the STRING type. Default value: `'t'`. If the value of `mode` is `'t'`, the file is displayed in text mode. If the value of `mode` is `'b'`, the file is displayed in binary mode.
  - The return value is a file-like object. If this object is no longer used, you must call the `close` method to release the open file.

  The following code shows how to reference a file.

```
from odps.udf import annotate
from odps.distcache import get_cache_file
@annotate('bigint->string')
class DistCacheExample(object):
def __init__(self):
    cache_file = get_cache_file('test_distcache.txt')
    kv = {}
    for line in cache_file:
        line = line.strip()
        if not line:
            continue
        k, v = line.split()
        kv[int(k)] = v
    cache_file.close()
    self.kv = kv
def evaluate(self, arg):
    return self.kv.get(arg)
```

- `odps.distcache.get_cache_table(resource_name)`: returns the content of a specified table.
  - `resource_name` specifies the name of the table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned. Data of the following types in the table can be read: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, FLOAT, CHAR, VARCHAR, BINARY, DATE, DECIMAL, ARRAY, MAP, and STRUCT.
  - The return value is of the GENERATOR data type. The caller traverses the table to obtain the table content. A record of the ARRAY type is obtained each time the caller traverses the table.

The following code shows how to reference a table.

```
from odps.udf import annotate
from odps.distcache import get_cache_table
@annotate('->string')
class DistCacheTableExample(object):
    def __init__(self):
        self.records = list(get_cache_table('udf_test'))
        self.counter = 0
        self.ln = len(self.records)
    def evaluate(self):
        if self.counter > self.ln - 1:
            return None
        ret = self.records[self.counter]
        self.counter += 1
        return str(ret)
```

Instructions

After you develop a Python 3 UDF, you can use MaxCompute SQL to call the UDF. For more information about how to call a Python 3 UDF, see Development process. You can call a Python 3 UDF by using one of the following methods:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more

information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute client to develop and call a Python 3 UDF, see Develop a Python UDF.

3.10.2.4. Use UDFs to access resources in VPCs

By default, MaxCompute does not allow you to access resources in virtual private clouds (VPCs) by using user-defined functions (UDFs). To use UDFs to access resources in a VPC, you must establish a network connection between MaxCompute and the VPC. This topic describes how to use a UDF to access resources in a VPC.

Prerequisites

Make sure that the following requirements are met:

- A UDF is written and created.

  For more information about how to write and create a UDF, see Java UDFs or Python UDFs.

- The MaxCompute client is installed.

  For more information about how to install and configure the MaxCompute client, see Install and configure the MaxCompute client.

Context

Before you can use a UDF to access resources in a VPC, you must establish a network connection between MaxCompute and the VPC. For more information, see VPC connection scheme. After you establish the network connection, add `set odps.session.networklink=<networklink_name>;` before the SQL statement that you want to execute. Then, commit the SET command together with the SQL statement to call the UDF. `networklink_name` specifies the name of the network connection that you established.

Limits

The VPC connection scheme is supported only in the China (Beijing), China (Shanghai), China (Zhangjiakou), China (Hangzhou), and China (Shenzhen) regions. Therefore, you can use UDFs to access only the resources that are deployed in VPCs in these regions.

Step 1: Establish a network connection

Log on to the MaxCompute console and establish a network connection between MaxCompute and a VPC. For more information, see VPC connection scheme.

Step 2: Call the UDF to access resources in the VPC

1. Start the MaxCompute client.
2. Execute the SQL statement to call the UDF.

   In this example, a UDF named my_ping is created. The my_ping UDF is used to determine whether an IP address can be pinged. Sample statements:

```
-- Configure the name of the network connection that you established based on the VPC c
onnection scheme. This setting is valid only for the current session.
set odps.session.networklink=testLink;
-- Execute the following SQL statement to call the UDF to access resources in the VPC:
select my_ping('123.56.xxx.xxx');
```

If True is returned, the network connection is established and the UDF can be called. If an error is returned, check whether the network connection information is correctly configured.

3.10.2.5. Java UDF examples

3.10.2.5.1. Write a Hive UDF in Java

This topic describes how to use the MaxCompute client to write a Hive user-defined function (UDF) in Java. A Hive version that is compatible with MaxCompute is used.

Prerequisites

The MaxCompute client is installed. For more information, see Install and configure the MaxCompute client.

Usage notes

Before you use a Hive UDF, take note of the following points:

- When you run the `ADD JAR` command to add resource packages for a Hive UDF on the MaxCompute client, you must specify all JAR packages that you want to add. This is because MaxCompute cannot automatically add all JAR packages to the classpath.
- To call a Hive UDF, you must add the `set odps.sql.hive.compatible=true;` command before the SQL statement that you want to execute. Then, commit and execute the command and SQL statement together.
- If a Java UDF is running in a distributed environment, the UDF is limited by the Java sandbox for MaxCompute. For more information, see Java sandbox.

Sample code

Sample code:

```
package com.aliyun.odps.compiler.hive;
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspectorFactory;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
public class Collect extends GenericUDF {
  @Override
  public ObjectInspector initialize(ObjectInspector[] objectInspectors) throws UDFArgumentE
xception {
    if (objectInspectors.length == 0) {
      throw new UDFArgumentException("Collect: input args should >= 1");
    }
    for (int i = 1; i < objectInspectors.length; i++) {
      if (objectInspectors[i] != objectInspectors[0]) {
        throw new UDFArgumentException("Collect: input oi should be the same for all args")
;
      }
    }
    return ObjectInspectorFactory.getStandardListObjectInspector(objectInspectors[0]);
  }
  @Override
  public Object evaluate(DeferredObject[] deferredObjects) throws HiveException {
    List<Object> objectList = new ArrayList<>(deferredObjects.length);
    for (DeferredObject deferredObject : deferredObjects) {
      objectList.add(deferredObject.get());
    }
    return objectList;
  }
  @Override
  public String getDisplayString(String[] strings) {
    return "Collect";
  }
}
```

The preceding code packages a random number of parameters that are of any data type into an array. The JAR file of the Hive UDF is named test.jar in this example.

Procedure

1. Start the MaxCompute client.
2. Package the code in Sample code into a JAR file by using the Hive platform. Then, run the following command to add the JAR file as a MaxCompute resource.

   ```
   -- Add the JAR file as a MaxCompute resource.
   add jar test.jar;
   ```

   For more information about how to add resources, see Add resources.
3. Run the following command to register the UDF:

```
-- Register the UDF.
create function hive_collect as 'com.aliyun.odps.compiler.hive.Collect' using 'test.jar
';
```

For more information about how to register UDFs, see Create a UDF.

4. Run the following command and SQL statement to call the UDF:

```
-- Set odps.sql.hive.compatible to true to ensure the compatibility with Hive.
set odps.sql.hive.compatible=true;
-- Call the UDF.
select hive_collect(4y, 5y, 6y);
```

Returned result:

```
+------+
| _c0  |
+------+
| [4, 5, 6] |
+------+
```

3.10.2.5.2. Use complex data types in Java UDFs

This topic describes how to use complex data types in Java user-defined functions (UDFs) by using MaxCompute Studio.

Prerequisites

MaxCompute Studio is installed and connected to a MaxCompute project. A MaxCompute Java module is created.

For more information, see Install MaxCompute Studio, Manage project connections, and Create a MaxCompute Java module.

Sample code

In the following code, three evaluate methods are defined for function overloading.

- Method 1: Use an array as a parameter. The array corresponds to the java.util.List class.
- Method 2: Use a map as a parameter. The map corresponds to the java.util.Map class.
- Method 3: Use a struct as a parameter. The struct corresponds to the com.aliyun.odps.data.Struct class.

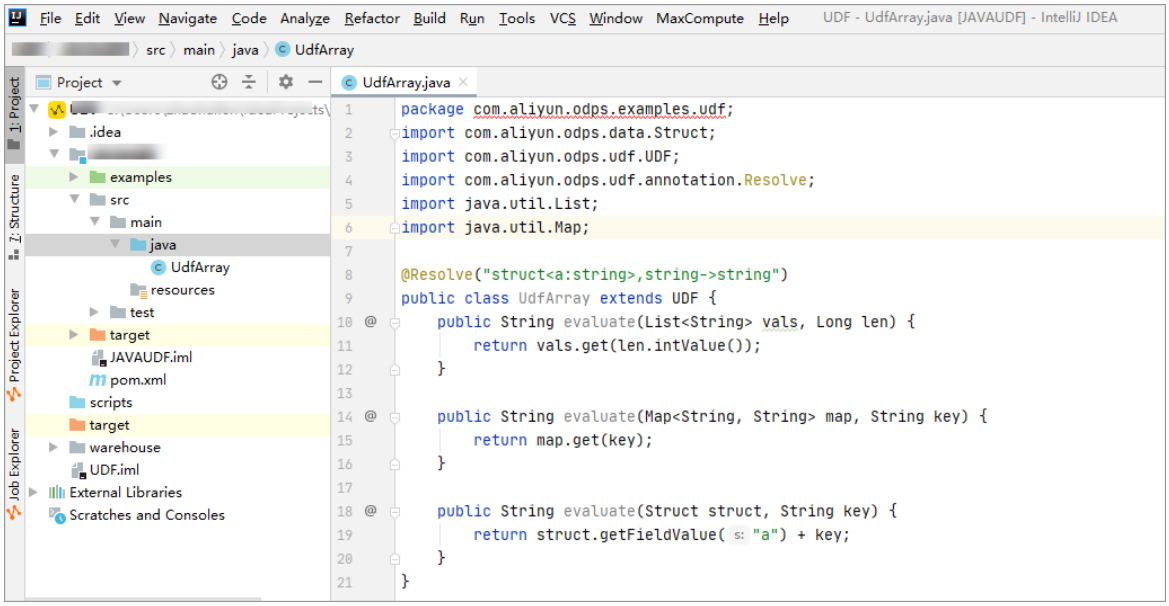
> **Note** You cannot use the reflection feature for the com.aliyun.odps.data.Struct class to obtain the names and types of fields. If you want to use the STRUCT data type for a UDF, you must add the `@Resolve annotation` to the com.aliyun.odps.data.Struct class. This annotation affects only the overloading of a UDF whose input parameters or return value contains the com.aliyun.odps.data.Struct class.

```
import com.aliyun.odps.data.Struct;
import com.aliyun.odps.udf.UDF;
import com.aliyun.odps.udf.annotation.Resolve;
import java.util.List;
import java.util.Map;
@Resolve("struct<a:string>,string->string")
public class UdfArray extends UDF {
    // Receive two parameters. The first parameter corresponds to the ARRAY data type, and the second parameter corresponds to the index of the element that you want to obtain. The code segment is used to obtain the element at the index position.
    public String evaluate(List<String> vals, Long index) {
        return vals.get(index.intValue());
    }
    // Receive two parameters. The first parameter corresponds to the MAP data type, and the second parameter corresponds to the key that you want to obtain. The code segment is used to obtain the value that corresponds to the key.
    public String evaluate(Map<String, String> map, String key) {
        return map.get(key);
    }
    // Receive two parameters. The first parameter corresponds to the STRUCT data type, and the second parameter is a key value. The code segment is used to obtain the value that corresponds to member variable a in the data of the STRUCT data type, add the key value to the obtained value, and then return the value of the STRING type.
    public String evaluate(Struct struct, String key) {
        return struct.getFieldValue("a") + key;
    }
}
```

Procedure

1. Write a Java UDF in MaxCompute Studio. In this example, the name of the Java class is `UdfArray`, and the code in Sample code is used.

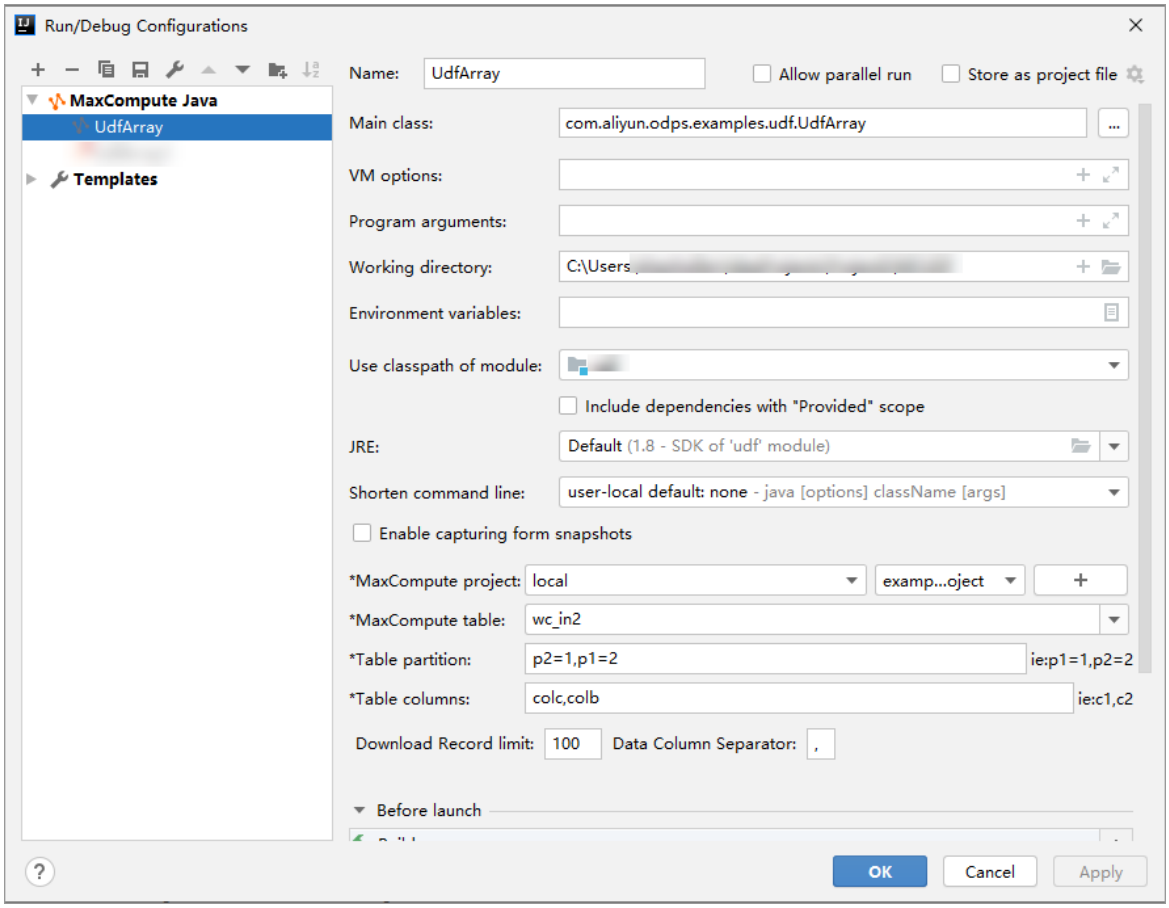
   For more information about how to write a UDF, see Write a UDF.



2. Run and debug the UDF on your on-premises machine to check whether the UDF code is run as

expected.

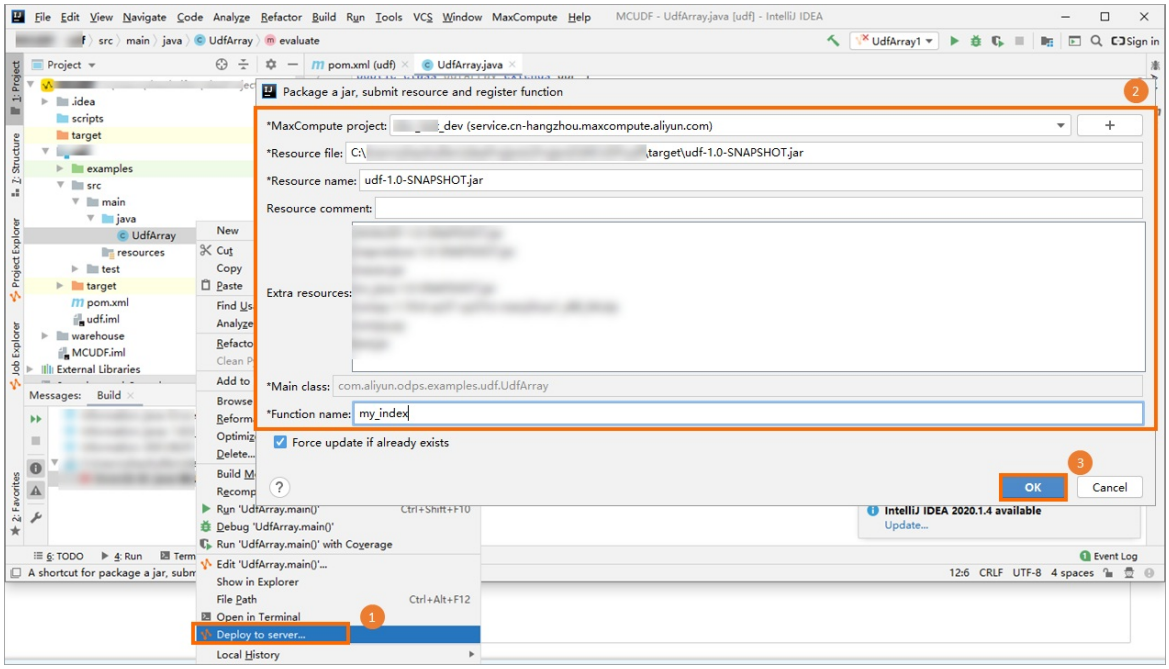
For more information about how to debug UDFs, see Perform a local run to debug the UDF.



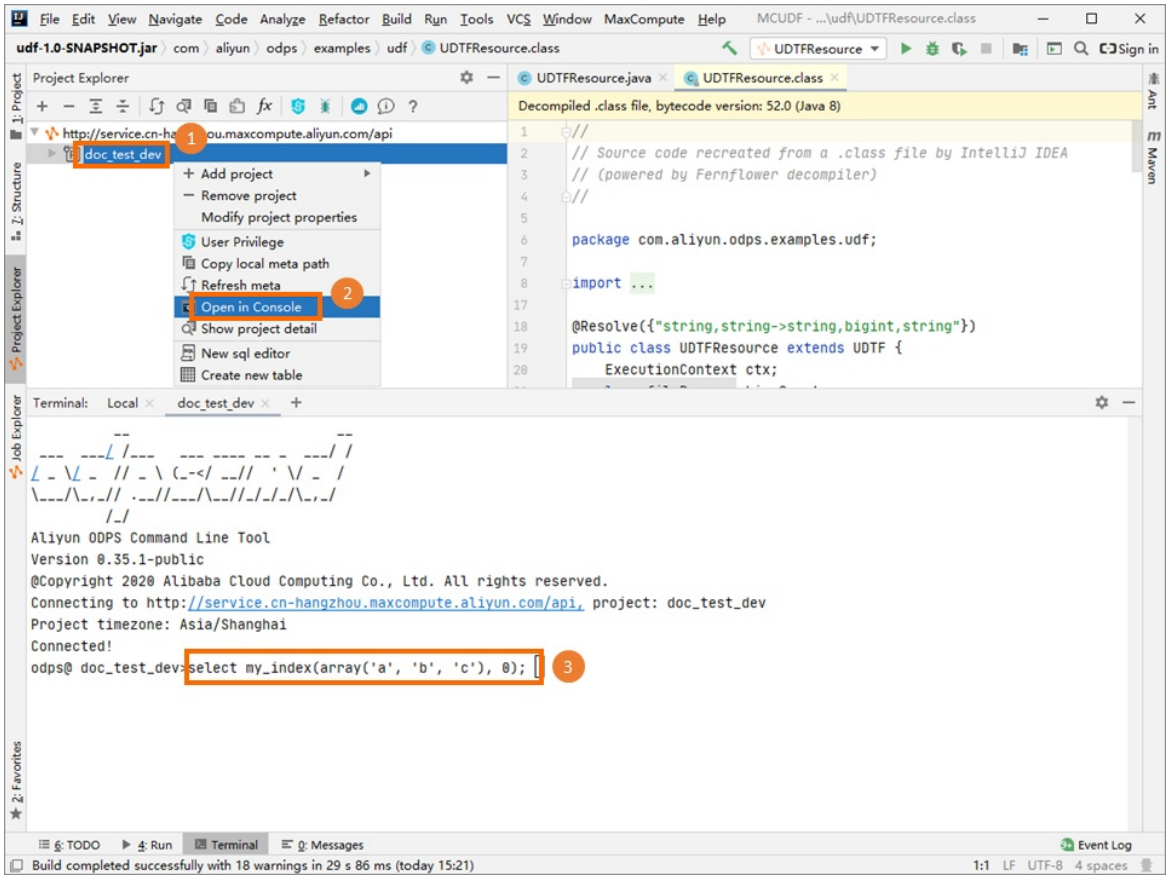
> **Note** The parameter settings in the preceding figure are provided for reference.

3. Package the created UDF, such as `my_index`, into a JAR file, upload the file to the MaxCompute project, and then register the UDF.

   For more information about how to package UDFs, see Procedure.



4. In the left-side navigation pane of MaxCompute Studio, click the **Project Explorer** tab. Right-click your MaxCompute project and select Open in Console to start the MaxCompute client. Then, execute an SQL statement to call the UDF that you created.



Sample statements:

```
select my_index(array('a', 'b', 'c'), 0); -- The return value is a.
select my_index(map('key_a','val_a', 'key_b', 'val_b'), 'key_b'); -- The return value i
s val_b.
select my_index(named_struct('a', 'hello'), 'world'); -- The return value is hello worl
d.
```

3.10.2.5.3. Obtain a JSON string

This topic describes how to use a Java user-defined function (UDF) to obtain a JSON string based on the specified path.

UDF used to obtain a JSON string

```
UDFGetJsonObject(String jsonString, String pathString)
```

- Description

  This UDF parses a JSON string specified by the jsonString parameter and returns the JSON object at the path specified by the pathString parameter. If the input JSON string is invalid, null is returned.

- Parameters
  - jsonString: the JSON string.
  - pathString: the path. A dollar sign `$` indicates a root node, a period `.` indicates a child node, and a pair of brackets `[]` indicates an array subscript. The value can only contain letters, digits, and underscores (_).

UDF example

- Upload resources

  Due to sandbox limits, to run the UDF in Alibaba Cloud MaxCompute, you must upload the *org.json* package and the UDF JAR package to MaxCompute as resources. Reference the two packages when you create the UDF in DataWorks. You can visit MVN to download the *org.json* package.

- Register the UDF

  After *UDFGetJsonObject.java* is tested, register the UDF. In this example, *json4.jar* is the JAR package of the UDF, and `json-20180813.jar` is the `org.json` package.

- Example

  After the UDF is registered, execute the following statement:

  ```
  select UDFGetJsonObject('{"grade":"One","persons":[{"age":"a1","name":"n1"},{"age":"a2","
  name":"n2"}]}', '$.persons[0].age');
  ```

  The result is `a1`.

UDF code example

```
package com.aliyun.odps.examples.udf; // The package name, which can be customized.
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
```

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
// Alibaba Cloud UDF.
import com.aliyun.odps.udf.UDF;
public class UDFGetJsonObject extends UDF {
  // Declare a private immutable pattern.
  private final Pattern patternKey = Pattern.compile("^([a-zA-Z0-9_\\-]+).*");
  private final Pattern patternIndex = Pattern.compile("\\[([0-9]+|\\*)\\]");
  // Use LinkedHashMap to implement a least recently used (LRU) cache.
  // Define the inner static subclasses of HashCache, which are inherited from the LinkedHa
shMap class.
  static class HashCache<K, V> extends LinkedHashMap<K, V> {
    private static final int CACHE_SIZE = 16;
    private static final int INIT_SIZE = 32;
    // Declare LOAD_FACTOR of the private constant HashMap.
    private static final float LOAD_FACTOR = 0.6f;
    HashCache() {
      super(INIT_SIZE, LOAD_FACTOR);
    }
    private static final long serialVersionUID = 1;
    // Override the LinkedHashMap method to determine whether the cache size exceeds the li
mit and clear the cache if necessary.
    @Override
    protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
      return size() > CACHE_SIZE;
    }
  }
  // Use HashCache to create a cache area of the Map type.
  static Map<String, Object> extractObjectCache = new HashCache<String, Object>();
  static Map<String, String[]> pathExprCache = new HashCache<String, String[]>();
  static Map<String, ArrayList<String>> indexListCache = new HashCache<String, ArrayList<St
ring>>();
  static Map<String, String> mKeyGroup1Cache = new HashCache<String, String>();
  static Map<String, Boolean> mKeyMatchesCache = new HashCache<String, Boolean>();
  String result = new String();
  public UDFGetJsonObject() {
  }
  /**
   * Extract json object from a json string based on json path specified, and
   * return json string of the extracted json object. It will return null if the
   * input json string is invalid.
   *
   * A limited version of JSONPath supported: $ : Root object . : Child operator
   * [] : Subscript operator for array * : Wildcard for []
   *
   * Syntax not supported that's worth noticing: '' : Zero length string as key
   * .. : Recursive descent &amp;#064; : Current object/element () : Script
   * expression ?() : Filter (script) expression. [,] : Union operator
   * [start:end:step] : array slice operator
   *
   * @param jsonString
   *
```

```
   *          the json string.
   * @param pathString
   *          the json path expression.
   * @return json string or null when an error happens.
   */
  public String evaluate(String jsonString, String pathString) {
    // Create the evaluate method. Like in Hive, UDFs in MaxCompute use the evaluate method
, with jsonString as the input JSON string and pathString as the input path.
    // If the input JSON string or path is empty, null is returned.
    if (jsonString == null || jsonString.equals("") || pathString == null
        || pathString.equals("")) {
      return null;
    }
    try {
      // Cache pathExpr
      // Obtain the pathExpr parameter of the String Array type, which indicates an array o
f paths.
      String[] pathExpr = pathExprCache.get(pathString);
      if (pathExpr == null) {
        // Use a period (.) as a separator to obtain the pathExpr parameter of the String A
rray type. A value of -1 indicates that separators in the last few characters are also spli
t.
        pathExpr = pathString.split("\\.", -1);
        // Use the put method to cache pathString and pathExpr to pathExprCache.
        pathExprCache.put(pathString, pathExpr);
      }
      // If the path does not start with a dollar sign ($), null is returned.
      if (! pathExpr[0].equalsIgnoreCase("$")) {
        return null;
      }
      // Cache extractObject
      Object extractObject = extractObjectCache.get(jsonString);
      if (extractObject == null) {
        extractObject = new JSONObject(jsonString);
        // Use the put method to cache the content of the JSON string.
        extractObjectCache.put(jsonString, extractObject);
      }
      for (int i = 1; i < pathExpr.length; i++) {
        // Use the extract method to extract a JSON string based on pathExpr.
        extractObject = extract(extractObject, pathExpr[i]);
      }
      // Provide the obtained JSON string of the Object type as a String-type string.
      return extractObject.toString();
    } catch (Exception e) {
      // If an exception occurs, null is returned.
      return null;
    }
  }
  private Object extract(Object json, String path) throws JSONException {
    // Cache patternkey.matcher(path).matches()
    Matcher mKey = null;
    Boolean mKeyMatches = mKeyMatchesCache.get(path);
    if (mKeyMatches == null) {
      // The path does not exist in the cache.
      mKey = patternKey.matcher(path);
```

```
      mKey = patternKey.matcher(path);
      // If the path matches a JSON string, True is returned. If the path fails to match a
JSON string, False is returned.
      mKeyMatches = mKey.matches() ? Boolean.TRUE : Boolean.FALSE;
      mKeyMatchesCache.put(path, mKeyMatches);
    }
    if (! mKeyMatches.booleanValue()) {
      return null;
    }
    // Cache mkey.group(1)
    // Obtain the JSON string that corresponds to the path.
    String mKeyGroup1 = mKeyGroup1Cache.get(path);
    if (mKeyGroup1 == null) {
      // Check whether the JSON string exists in the cache.
      if (mKey == null) {
        mKey = patternKey.matcher(path);
      }
      mKeyGroup1 = mKey.group(1);
      // Cache the JSON string based on the path.
      mKeyGroup1Cache.put(path, mKeyGroup1);
    }
    // Obtain the JSON string.
    json = extract_json_withkey(json, mKeyGroup1);
    // Cache indexList
    // Obtain the JSON string of the Array type.
    ArrayList<String> indexList = indexListCache.get(path);
    if (indexList == null) {
      // Check whether the JSON string of the Array type exists in the cache.
      Matcher mIndex = patternIndex.matcher(path);
      indexList = new ArrayList<String>();
      while (mIndex.find()) {
        indexList.add(mIndex.group(1));
      }
      indexListCache.put(path, indexList);
    }
    if (indexList.size() > 0) {
      // Extract the JSON string of the Array type based on the path.
      json = extract_json_withindex(json, indexList);
    }
    return json;
  }
  // Create a JSON object of the Array type.
  ArrayList<Object> jsonList = new ArrayList<Object>();
  // Create a method to obtain a JSON string of the Array type.
  private Object extract_json_withindex(Object json, ArrayList<String> indexList)
      throws JSONException {
    jsonList.clear();
    jsonList.add(json);
    Iterator<String> itr = indexList.iterator();
    while (itr.hasNext()) {
      String index = itr.next();
      ArrayList<Object> tmp_jsonList = new ArrayList<Object>();
      // If the path contains a wildcard, all JSON strings that match the wildcard are retu
rned.
      if (index.equalsIgnoreCase("*")) {
```

```
        for (int i = 0; i < (jsonList).size(); i++) {
          try {
            JSONArray array = (JSONArray) (jsonList).get(i);
            for (int j = 0; j < array.length(); j++) {
              tmp_jsonList.add(array.get(j));
            }
          } catch (Exception e) {
            continue;
          }
        }
        jsonList = tmp_jsonList;
      } else {
        // If no wildcard exists, all JSON data is traversed.
        for (int i = 0; i < (jsonList).size(); i++) {
          try {
            tmp_jsonList.add(((JSONArray) (jsonList).get(i)).get(Integer
                .parseInt(index)));
          } catch (ClassCastException e) {
            continue;
          } catch (JSONException e) {
            return null;
          }
          jsonList = tmp_jsonList;
        }
      }
    }
    return (jsonList.size() > 1) ? new JSONArray(jsonList) : jsonList.get(0);
  }
  // Create a method to obtain the JSON string of common types.
  private Object extract_json_withkey(Object json, String path)
      throws JSONException {
    if (json.getClass() == org.json.JSONArray.class) {
      JSONArray jsonArray = new JSONArray();
      for (int i = 0; i < ((JSONArray) json).length(); i++) {
        Object josn_elem = ((JSONArray) json).get(i);
        try {
          Object json_obj = ((JSONObject) josn_elem).get(path);
          if (json_obj.getClass() == org.json.JSONArray.class) {
            for (int j = 0; j < ((JSONArray) json_obj).length(); j++) {
              jsonArray.put(((JSONArray) json_obj).get(j));
            }
          } else {
            jsonArray.put(json_obj);
          }
        } catch (Exception e) {
          continue;
        }
      }
      return (jsonArray.length() == 0) ? null : jsonArray;
    } else {
      return ((JSONObject) json).get(path);
    }
  }
}
```

3.10.2.5.4. Check the value of a JSON string

This topic describes how to use a Java UDF to obtain a JSON string based on its path and check whether it has a specific value.

UDF used to check the value of a JSON string

```
UDFGetJsonObject(String jsonString, String pathString)
```

- Description

  This UDF is used to parse jsonString and obtain the value of pathString. It can also be used to determine whether the obtained value contains a specific value.

- Parameters
  - jsonString: the JSON string.
  - pathString: the path. A dollar sign (`$`) indicates a root node, a period (`.`) indicates a child node, and a bracket (`[]`) indicates an array subscript. pathString can only contain letters, digits, and underscores (_).

UDF example

- Resource upload

  Due to sandbox limits, to run this function in Alibaba Cloud MaxCompute, you need to upload the *org.json* and UDF JAR packages to MaxCompute as resources and reference the two packages when you create a function in DataWorks. You can visit MVN to download the *org.json* package.

- Function registration

  TestJson.jar is the JAR package generated by packaging the example code of the UDF used to check the value of a JSON string. *json-20180813.jar* is the `org.json` package.

- Examples

  After the UDF is registered, execute the following statements:

  ```
  select TestJson('{"LogA\":[{"id":"11562,20508"}, {"id":"11563,20509"}]}', "LogA[0]");
  ```

  The result is `{"id":"11562,20508"}` .

  ```
  select bi_udf:bi_get_value('{"LogA\":[{"id":"11562,20508"}, {"id":"11563,20509"}]}', "LogA[0].id", "20508");
  ```

  The result is `true` .

  ```
  select bi_udf:bi_get_value('{"LogA\":[{"id":"11562,20508"}, {"id":"11563,20509"}]}', "LogA[0].id", "20508", "&");
  ```

  The result is `true` .

UDF code example

```
package com.aliyun.odps.examples.udf; // The package name, which can be defined as needed.
```

```
import com.aliyun.odps.io.Text;
import com.aliyun.odps.udf.UDF; // Alibaba Cloud UDF.
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class UDFGetValue extends UDF {
    // Declare the mode of a regular expression with the private final pattern.
    private final Pattern patternKey = Pattern.compile("^([a-zA-Z0-9_\\-]+).*");
    private final Pattern patternIndex = Pattern.compile("\\[([0-9]+|\\*)\\]");
    // Use HashCache to create a cache region of the Map type.
    private static Map<String, Object> extractObjectCache = new HashCache<String, O
bject>();
    private static Map<String, String[]> pathExprCache = new HashCache<String, Stri
ng[]>();
    private static Map<String, ArrayList<String>> indexListCache = new HashCache<Stri
ng, ArrayList<String>>();
    private static Map<String, String> mKeyGroup1Cache = new HashCache<String, Stri
ng>();
    private static Map<String, Boolean> mKeyMatchesCache = new HashCache<String, Bo
olean>();
    private Text result = new Text();
    private ArrayList<Object> jsonList = new ArrayList<Object>();
    // An LRU cache using a linked hash map.
    // Define the inner static subclasses of HashCache, which are inherited from the Linked
HashMap class.
    private static class HashCache<K, V> extends LinkedHashMap<K, V> {
        private static final int CACHE_SIZE = 16;
        private static final int INIT_SIZE = 32;
        // Declare LOAD_FACTOR of the private constant HashMap.
        private static final float LOAD_FACTOR = 0.6f;
        private static final long serialVersionUID = 1;
        HashCache() {
            super(INIT_SIZE, LOAD_FACTOR);
        }
        // LinkedHashMap overload is used to determine whether the cache size exceeds the l
imit and clear the cache if necessary.
        @Override
        protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
            return size() > CACHE_SIZE;
        }
    }
    public UDFGetValue() {
    }
    /**
     * @param jsonStr indicates the input JSON string.
     * @param path indicates the JSON path.
     * @return indicates the return value.
     */
```

```
    public String evaluate(String jsonStr, String path) {
        // The way to create the EVALUATE method, which is the same as HIVE. MaxCompute UDF
s generally use the EVALUATE method, with jsonString as the input JSON string and pathStrin
g as the input path.
        // If the path to the input JSON string is empty, null is returned.
        if (jsonStr == null || jsonStr == "" || path == null || path == "")
{
            return null;
        }
        String jResult = evaluateJson(jsonStr, path);
        if (jResult == null || "".equals(jResult))
            return null;
        result.set(jResult);
        return result.toString();
    }
    /**
     * @param jsonStr indicates the input JSON string.
     * @param path indicates the JSON path.
     * @param value indicates the return value.
     * @return checks whether a specific path in the JSON string contains all content in va
lue and then returns "true" or "false" based on the result.
     */
    public String evaluate(String jsonStr, String path, String value) {
        if (jsonStr == null || jsonStr == "" || path == null
                || path == "" || value == null || "".equals(value)) {
            return null;
        }
        String jResult = evaluateJson(jsonStr, path);
        if (jResult == null || "".equals(jResult)) {
            result.set("false");
            return result.toString();
        }
        String[] sV = value.split(",");
        String[] dV = jResult.split(",");
        int sameCount = 0;
        for (int i = 0; i < sV.length; i++) {
            for (int j = 0; j < dV.length; j++) {
                if (sV[i].equals(dV[j])) {
                    sameCount++;
                    break;
                }
            }
        }
        if (sameCount == sV.length) {
            result.set("true");
        } else {
            result.set("false");
        }
        return result.toString();
    }
    /**
     * @param jsonStr indicates the input JSON string.
     * @param path indicates the JSON path.
     * @param value indicates the return value.
     * @param flag can be set to a vertical bar (|) which indicates that any content in val
```

```
     * @param flag can be set to a vertical bar (|) which indicates that any content in val
ue is contained or an ampersand (&) which indicates that all content in value must be conta
ined.
     * @return checks whether the JSON string contains value and then returns "true" or "fa
lse" based on the result.
     */
    public String evaluate(String jsonStr, String path, String value, String flag) {
        if ("&".equals(flag))
            return evaluate(jsonStr, path, value);
        if (jsonStr == null || jsonStr.equals("") || path == null || path.equals("")
                || value == null || "".equals(value) || flag == null
                || !"|".equals(flag)) {
            result.set("false");
            return result.toString();
        }
        String jResult = evaluateJson(jsonStr, path);
        if (jResult == null || "".equals(jResult)) {
            result.set("false");
            return result.toString();
        }
        String[] sV = value.split(",");
        String[] dV = jResult.split(",");
        for (int i = 0; i < sV.length; i++) {
            for (int j = 0; j < dV.length; j++) {
                if (sV[i].equals(dV[j])) {
                    result.set("true");
                    return result.toString();
                }
            }
        }
        result.set("false");
        return result.toString();
    }
    /**
     * Extract json object from a json string based on json path specified, and
     * return json string of the extracted json object. It will return null if
     * the input json string is invalid.
     *
     * A limited version of JSONPath supported: $ : Root object . : Child
     * operator [] : Subscript operator for array * : Wildcard for []
     *
     * Syntax not supported that's worth noticing: '' : Zero length string as
     * key .. : Recursive descent &amp;#064; : Current object/element () :
     * Script expression ?() : Filter (script) expression. [,] : Union operator
     * [start:end:step] : array slice operator
     *
     * @param jsonString the json string.
     * @param pathString the json path expression.
     * @return json string or null when an error happens.
     */
    private String evaluateJson(String jsonString, String pathString) {
        try {
            pathString = "$." + pathString;
            // Cache pathExpr
            // Obtain the input path of the array of the STRING type.
```

```
            // Obtain the input path of the array of the STRING type.
            String[] pathExpr = pathExprCache.get(pathString);
            if (pathExpr == null) {
                // Use a period (.) as a separator to obtain the pathExpr array of the STRI
NG type of the input path. -1 indicates that separators in the last few digits are also spl
it.
                pathExpr = pathString.split("\\.", -1);
                // Use the PUT method to cache pathString and pathExpr to pathExprCache.
                pathExprCache.put(pathString, pathExpr);
            }
            // If the path does not start with a dollar sign ($), null is returned.
            if (! pathExpr[0].equalsIgnoreCase("$")) {
                return null;
            }
            // Cache extractObject
            Object extractObject = extractObjectCache.get(jsonString);
            if (extractObject == null) {
                extractObject = new JSONObject(jsonString);
                Object put = extractObjectCache.put(jsonString, extractObject);
            }
            for (int i = 1; i < pathExpr.length; i++) {
                // Use the EXTRACT method to extract a JSON string based on pathExpr.
                extractObject = extract(extractObject, pathExpr[i]);
            }
            // Provide the obtained JSON string of the OBJECT type as a STRING-type string.
            return extractObject.toString();
        } catch (Exception e) {
            // If an exception occurs, null is returned.
            return null;
        }
    }
    private Object extract(Object json, String path) throws JSONException {
        // Cache patternkey.matcher(path).matches()
        Matcher mKey = null;
        Boolean mKeyMatches = mKeyMatchesCache.get(path);
        // The path does not exist in the cache.
        if (mKeyMatches == null) {
            mKey = patternKey.matcher(path);
            // If the path matches a JSON string, True is returned. If the path fails to ma
tch a JSON string, False is returned.
            mKeyMatches = mKey.matches() ? Boolean.TRUE : Boolean.FALSE;
            mKeyMatchesCache.put(path, mKeyMatches);
        }
        if (! mKeyMatches.booleanValue()) {
            return null;
        }
        // Cache mkey.group(1)
        // Obtain the JSON string that corresponds to the path.
        String mKeyGroup1 = mKeyGroup1Cache.get(path);
        // Check whether the JSON string exists in the cache.
        if (mKeyGroup1 == null) {
            if (mKey == null) {
                mKey = patternKey.matcher(path);
            }
            mKeyGroup1 = mKey.group(1);
```

```
            // Cache the JSON string based on the path.
            mKeyGroup1Cache.put(path, mKeyGroup1);
        }
        // Obtain the JSON string.
        json = extract_json_withkey(json, mKeyGroup1);
        // Cache indexList
        // Obtain the JSON string of the ARRAY type.
        ArrayList<String> indexList = indexListCache.get(path);
        // Check whether the JSON string of the ARRAY type exists in the cache.
        if (indexList == null) {
            Matcher mIndex = patternIndex.matcher(path);
            indexList = new ArrayList<String>();
            while (mIndex.find()) {
                indexList.add(mIndex.group(1));
            }
            indexListCache.put(path, indexList);
        }
        if (indexList.size() > 0) {
            // Extract the JSON string of the ARRAY type based on the path.
            json = extract_json_withindex(json, indexList);
        }
        return json;
    }
    private Object extract_json_withindex(Object json, ArrayList<String> indexList) throws
JSONException {
        jsonList.clear();
        jsonList.add(json);
        Iterator<String> itr = indexList.iterator();
        while (itr.hasNext()) {
            String index = itr.next();
            // Create a JSON object of the ARRAY type.
            ArrayList<Object> tmp_jsonList = new ArrayList<Object>();
            // Create a method to obtain a JSON string of the ARRAY type.
            if (index.equalsIgnoreCase("*")) {
                for (int i = 0; i < (jsonList).size(); i++) {
                    try {
                        JSONArray array = (JSONArray) (jsonList).get(i);
                        for (int j = 0; j < array.length(); j++) {
                            tmp_jsonList.add(array.get(j));
                        }
                    } catch (Exception e) {
                        continue;
                    }
                }
                jsonList = tmp_jsonList;
              // If no wildcard exists, all JSON data is traversed.
            } else {
                for (int i = 0; i < (jsonList).size(); i++) {
                    try {
                        tmp_jsonList.add(((JSONArray) (jsonList).get(i))
                                .get(Integer.parseInt(index)));
                    } catch (ClassCastException e) {
                        continue;
                    } catch (JSONException e) {
```

```
                        return null;
                    }
                    jsonList = tmp_jsonList;
                }
            }
        }
        return (jsonList.size() > 1) ? new JSONArray(jsonList) : jsonList
                .get(0);
    }
    // Create a method to obtain the JSON string of common types.
    private Object extract_json_withkey(Object json, String path) throws JSONException {
        if (json.getClass() == JSONArray.class) {
            JSONArray jsonArray = new JSONArray();
            for (int i = 0; i < ((JSONArray) json).length(); i++) {
                Object josn_elem = ((JSONArray) json).get(i);
                try {
                    Object json_obj = ((JSONObject) josn_elem).get(path);
                    if (json_obj.getClass() == JSONArray.class) {
                        for (int j = 0; j < ((JSONArray) json_obj).length(); j++) {
                            jsonArray.put(((JSONArray) json_obj).get(j));
                        }
                    } else {
                        jsonArray.put(json_obj);
                    }
                } catch (Exception e) {
                    continue;
                }
            }
            return (jsonArray.length() == 0) ? null : jsonArray;
        } else {
            return ((JSONObject) json).get(path);
        }
    }
}
```

3.10.2.5.5. Convert data types to JSON STRING

This topic describes how to use a Java UDF to convert data types to JSON STRING.

During data usage, you often need to convert data types to JSON STRING. For example, if the data type is ARRAY, you may need to convert it to JSON STRING for data import and download because Tunnel does not support uploading and downloading data of complex types.

UDF used to convert data types to JSON STRING

```
toJSONString()
```

- Description: used to convert a MaxCompute object (field) to a JSON string, along with the necessary string escape. This UDF can work with `PutJsonValues`.
- Parameters: The types of input parameters cannot be DATETIME.

UDF example

- Resource upload

  Due to sandbox limits, to run this function in Alibaba Cloud MaxCompute, you need to upload the *fastjson.JSON* and UDF JAR packages to MaxCompute as resources and reference the two packages when you create a function in DataWorks. You can visit MVN to download the *fastjson.JSON* package.

- Function registration

  After *ToJsonStrin.java* passes the test, register it as a function.

  In this example, *ODPSUDF-1.0-SNAPSHOT2.jar* is the JAR package generated by packaging the example code of the UDF used to convert data types to JSON STRING. *fastjson-1.2.28.odps.jar* is the *fastjson.JSON* package.

- Example

  ```
  SELECT tojson(array(2.0,3.0,4.0));
  ```

  > **Note** `tojson` is the function you register by using DataWorks or IntelliJ IDEA.

  The result is as follows:

  ```
  [2.0,3.0,4.0]
  ```

UDF code example

```
import com.aliyun.odps.udf.UDF; // Alibaba Cloud UDF.
import java.math.BigDecimal;
import com.alibaba.fastjson.JSON;  // The fastjson.JSON class is introduced.
import java.util.Arrays;
import java.util.List;
public class ToJsonString extends UDF {
// The following code calls the toJSONString method of fastjson.JSON to convert various dat
a types to JSON STRING. You can add more data types as needed.
    public String evaluate(String obj) {
        return JSON.toJSONString(obj);
    }
    public String evaluate(Long obj) {
        return JSON.toJSONString(obj);
    }
    public String evaluate(Double obj) {
        return JSON.toJSONString(obj);
    }
    public String evaluate(Boolean obj) {
        return JSON.toJSONString(obj);
    }
    public String evaluate(BigDecimal obj) {
        return JSON.toJSONString(obj);
    }
// In MaxCompute, the ARRAY type corresponds to the LIST type in Java UDFs. For example, th
e Array <double> type matches the List <Double> type in a UDF.
    public String evaluate(List<Double> obj) {
        return JSON.toJSONString(obj);
    }
    public String evaluate(double[] obj) {
        return JSON.toJSONString(obj);
    }
    public String evaluate(int[] obj) {
        return JSON.toJSONString(obj);
    }
}
```

UDF unit testing

```
// The following code is used to check whether the data type conversion takes effect. You c
an comment out the code when you use the function.
    public static void main(String[] args) {
        //System.out.println(new ToJsonString().evaluate(null));//MaxCompute can call this
method but Java cannot.
        System.out.println(new ToJsonString().evaluate("Chinese\t"));
        System.out.println(new ToJsonString().evaluate(123L));
        System.out.println(new ToJsonString().evaluate(1.123));
        System.out.println(new ToJsonString().evaluate(true));
        //System.out.println(new ToJsonString().evaluate(new Date()));// UDFs and UDAFs do
not support the DATETIME type.
        System.out.println(new ToJsonString().evaluate(BigDecimal.TEN));
        System.out.println(new ToJsonString().evaluate(Arrays.asList(1.0,2.0,3.0,4.0)));
        double[] args1 = {1,2,3,4};
        System.out.println(new ToJsonString().evaluate(args1));
       }
    }
```

3.10.2.5.6. Add key-value pairs to a JSON string

This topic describes how to use a Java UDF to add one or more key-value pairs to a basic JSON string or overwrite the pairs. Nested JSON strings support this function.

UDF used to add key-value pairs to a JSON string

```
PutJsonValues(String jsonBase,String Key1,String Value1,String Key2,String Value2...)
```

- Description: This function is used to add one or more key-value pairs to a basic JSON string or overwrite the pairs. Nested JSON strings support this function. The number of parameters can be an odd value. This UDF can work with the `ToJsonObeject` function in Convert data types to JSON STRING. You must first register the `ToJsonObeject` function.
- Parameters:
  - jsonBase: the basic JSON string to which you want to add key-value pairs.
  - Key1: Key1 that you want to add. It is of the STRING type.
  - Value1: Value1 that corresponds to Key1. It is of the STRING type.
  - Key2: Key2 that you want to add. It is of the STRING type.
  - Value2: Value2 that corresponds to Key2. It is of the STRING type.

UDF examples

- Resource upload

  Due to sandbox limits, to run this function in Alibaba Cloud MaxCompute, you must upload the *fastjson.JSON* and UDF JAR packages to MaxCompute as resources. When you create a function in DataWorks, you must reference the two packages. You can visit MVN to download the *fastjson.JSON* package.

- Function registration

  After *PutJsonValues.java* passes the test, register it as a function.

In this example, *PutJsonValues.jar* is the JAR package that is generated by packaging the sample code of the UDF used to add key-value pairs to a JSON string. *ODPSUDF-1.0-SNAPSHOT2.jar* is the JAR package generated by packaging `ToJsonString` in the previous topic. *fastjson-1.2.28.odps.jar* is a *fastjson.JSON* package.

> **Note**
> - For more information about the procedure in this example, see Create a MaxCompute resource.
> - To publish a UDF to a server for production use, you must package, upload, and then register the UDF. You can use the one-click publish feature to complete these steps. MaxCompute Studio allows you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence.
> - You can run common commands on the client to upload resources. For more information, see Resource operations.

- Examples

  After the UDF is registered, execute the following statement:

```
SELECT tojson(NULL)
,  tojson(concat('123', chr(3)))
,  tojson(123)
,  tojson(12.3)
,  tojson(true)
,  tojson(CAST(123  AS  DECIMAL))
,  putJsonValuesTest('{}', 'a', '123', 'b', tojson('abc'), 'c', '{"c1":567}');
```

  The following result is returned:

```
null
"123\u0003"
123
12.3
true
123
{"a":123,"b":"abc","c":{"c1":567}}
```

> **Note**
> - `null` indicates the NULL value. `a` is a numeric value without quotation marks. `b` is a string which needs to be converted to a JSON string by using `tojson`. `{"c1":567}` is a valid JSON element and can be directly passed.
> - `putJsonValues` and `tojson` are functions that you registered by using DataWorks or IntelliJ IDEA.

UDF code example

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject; // The fastjson.JSON class is introduced.
import com.aliyun.odps.udf.UDF;// Alibaba Cloud UDF.
public class PutJsonValues extends UDF {
// Data of the STRING type in MaxCompute is consistent with Java UDFs. The String... keyVal
ues field is a formal parameter that is changeable, which indicates that the number of para
meters is uncertain.
    public  PutJsonValues(){
    }
    public  String evaluate(String jsonBase, String... keyValues) {
        if (jsonBase == null) {
            return null;
        }
        JSONObject outputObj = JSON.parseObject(jsonBase);
        for (int i = 1; i < keyValues.length; i = i + 2) {
            String key = keyValues[i - 1];
            String value = keyValues[i];
            if (key == null || value == null) {
                continue;
            }
            Object valueObj = JSON.parse(value);
            outputObj.put(key, valueObj);
        }
        return JSON.toJSONString(outputObj);
    }
}
```

UDF unit testing

```
// The following code is used to check whether the data type conversion takes effect. You c
an comment out the code when you use this function.
public static void main(String[] args) {
    PutJsonValues putJsonValues=new PutJsonValues();
    System.out.println(putJsonValues.hashCode());
    String js=putJsonValues.evaluate("{}", "k", "null");
    System.out.println(js);
    System.out.println(new PutJsonValues().evaluate("{}", "a", "123", "b", "234", "c", "345
"));
    System.out.println(new PutJsonValues().evaluate("{}", "k", "\"abc\""));
    System.out.println(new PutJsonValues().evaluate("{}", "k", "{\"kk\":123}"));
}
```

3.10.2.5.7. Replace strings by using regular expressions

This topic describes how to use a Java UDF to replace strings by using regular expressions.

UDF used to replace strings by using regular expressions

```
String UDFRegxpReplace(String s, String regex, String replacement)
```

- Description: used to replace strings that match the regular expression. Compared with the REGEXP_REPLACE built-in function of MaxCompute, the regular expression in this UDF supports variables.
- Parameters:
  - s: the string you want to replace, which is of the STRING type
  - regex: the regular expression of the STRING type
  - replacement: the string you use to replace the original string, which is of the STRING type

UDF example

- Function registration

  After *UDFRegxpReplace.java* passes the test, register it as a function.

  > **Note** To publish a UDF to a server for production use, the UDF needs to go through packaging, uploading, and registration. You can use the one-click publish function to complete these steps. MaxCompute Studio allows you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence. For more information, see Package a Java program, upload the package, and create a MaxCompute UDF.

- Example

  After the UDF is registered, execute the following statement:

```
select UDFRegxpReplace("foobar", "oo|ar", "") from dual;
```

  The result is as follows:

```
+-----+
| _c0 |
+-----+
| fb  |
+-----+
```

UDF code example

```
// The package name, which can be defined as needed.
package com.aliyun.odps.examples.udf;
import com.aliyun.odps.udf.annotation.UdfProperty;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
@UdfProperty(isDeterministic=true)
public class UDFRegxpRplace extends UDF {
    private String lastRegex = new String();
    private Pattern p = null;
    private String lastReplacement = new String();
    private String replacementString = "";
    public String evaluate(String s, String regex, String replacement) {
        if (s == null || regex == null || replacement == null) {
            return null;
        }
        // If the regular expression is changed, compile it again.
        if (! regex.equals(lastRegex) || p == null) {
            lastRegex = regex;
            p = Pattern.compile(regex.toString());
        }
        Matcher m = p.matcher(s.toString());
        // If the string used to replace the original string is changed, run toString again
.
        if (! replacement.equals(lastReplacement)) {
            lastReplacement = replacement;
            replacementString = replacement.toString();
        }
        StringBuffer sb = new StringBuffer();
        while (m.find()) {
            m.appendReplacement(sb, replacementString);
        }
        m.appendTail(sb);
        return sb.toString();
    }
}
```

3.10.2.5.8. Obtain the values of strings that have no delimiters

This topic describes how to obtain the value that corresponds to a specific key in a string that has no delimiters.

UDF used to obtain the value of a string that has no delimiters

```
UDFKeyValue(String str, String splitor1, String splitor2, String key)
```

- Description: used to obtain the value that corresponds to a specific key in a string.

> **Note** This UDF is not suitable for a string that has delimiters. To obtain the value that corresponds to a specific key in a string that has delimiters, see UDFKeyValueEx.

- Parameters:
  - str: the target string.
  - splitor1: the delimiter used to split the string to obtain `key-value` pairs. The default value of splitor1 is a semicolon (`;`).
  - splitor2: the delimiter used to split the obtained `key-value` pairs. The default value of splitor2 is a colon (`:`).

UDF example

1. Function registration

   After *UDFKeyValue.java* passes the test, register it as a function.

   > **Note** To publish a UDF to a server for production use, the UDF needs to go through packaging, uploading, and registration. You can use the one-click publish function to complete these steps. MaxCompute Studio allows you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence. For more information, see Package a Java program, upload the package, and create a MaxCompute UDF.

2. Examples

   After the UDF is registered, execute one of the following statements:

   - Example 1

     ```
     select UDFKeyValue('a:1;b:2;','a');
     ```

     The result is as follows:

     ```
     +------+
     | _c0 |
     +------+
     | 1    |
     +------+
     ```

   - Example 2

     ```
     select UDFKeyValue('a:1;b:2;','\;',':','a');
     ```

     The result is as follows:

     ```
     +------+
     | _c0 |
     +------+
     | 1    |
     +------+
     ```

UDF code example

```
// The package name, which can be defined as needed.
package com.aliyun.odps.examples.udf;
```

```
import com.aliyun.odps.io.Text;
import com.aliyun.odps.udf.UDF;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
public class UDFKeyValue extends UDF{
        // Save all key-value pairs.
    private Map<String, String> mKeyValueache = new ConcurrentHashMap<String, String>();
    private Text result = new Text();
    public UDFKeyValue() {
    }
    public String evaluate(String str, String keyname){
        // The default value of split1 is a semicolon (;), and that of split2 is a colon (:
).
        return evaluate(str, ";", ":", keyname);
    }
    public String evaluate(String str, String split1, String split2, String keyname) {
        try {
                        // Use splitor1 to split the string to obtain key-value pairs.
            if (str == null || "".equals(str))
                return null;
            String[] values1 = str.split(split1);
            mKeyValueache.clear();
            int i = 0;
                        // Use splitor2 to split the obtained key-value pairs.
            while (i < values1.length) {
                storeKeyValue(values1[i], split2);
                i++;
            }
            String resultValue = getKeyValue(keyname);
            if (resultValue == null)
                return null;
            result.set(new Text(resultValue));
                        // Return the result.
            return result.toString();
        } catch (Exception e) {
            return null;
        }
    }
    private boolean storeKeyValue(String keyValues, String split) {
        if (keyValues == null || "".equals(keyValues))
            return false;
        if (mKeyValueache == null)
            mKeyValueache = new ConcurrentHashMap<String, String>();
        String[] keyValueArr = keyValues.split(split);
        if (keyValueArr.length == 2) {
            mKeyValueache.put(keyValueArr[0], keyValueArr[1]);
            return true;
        }
        return false;
    }
    private String getKeyValue(String keyName) {
        if (keyName == null ||
            "".equals(keyName) ||
            mKeyValueache == null ||
```

```
            mKeyValueache.size() == 0)
            return null;
        return mKeyValueache.get(keyName);
    }
}
```

3.10.2.5.9. Obtain the values of strings that have delimiters

This topic describes how to use a Java UDF to obtain the value that corresponds to a specific key in a string that has key/value delimiters.

UDF used to obtain the value of a string that has delimiters

```
UDFKeyValueEx(String str, String split1, String split2, String keyname)
```

- Description: used to obtain the value that corresponds to a specific key in a string. Use string delimiters and key/value delimiters to split a string in sequence and then obtain the value that corresponds to a specific key. Different from UDFKeyValue, this UDF is ideal for values that have key/value delimiters.
- Parameters:
  - `str`: the target string.
  - `split1`: the string delimiter, which is a regular expression. The default value is a semicolon (`;`).
  - `split2`: the key/value delimiter, which is not a regular expression. The default value is a colon (`:`).
  - `keyname`: the key whose value you want to obtain.

UDF example

1. Function registration

   After *UDFKeyValueEx.java* passes the test, register it as a function.

   > **Note**
   > - MaxCompute Studio provides the one-click publish function for you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence. You can also upload a JAR package by using DataWorks.
   > - You can upload resources by running common commands on the client. For more information, see MaxCompute resources.

2. Examples

   After the UDF is registered, execute one of the following statements:

   - Example 1

```
select UDFKeyValueEx('a:b;c:d', ';', ':', 'a');
```

The result is as follows:

```
+------+
| _c0 |
+------+
| b   |
+------+
```

- Example 2

```
select UDFKeyValueEx('a:b:c;c:d', ';', ':', 'a:b');
```

The result is as follows:

```
+------+
| _c0 |
+------+
| c   |
+------+
```

UDF code example

```
// The package name, which can be defined as needed.
package com.aliyun.odps.examples.udf;
import com.taobao.bi.odps.udf.UDF;
public class UDFKeyValueEx extends UDF {
    public String evaluate(String str, String keyname) {
                // The default value of split1 is a semicolon (;), and that of split2 is a
colon (:).
        return evaluate(str, ";", ":", keyname);
    }
    public String evaluate(String str, String split1, String split2, String keyname) {
        try {
            if (str==null || split1==null || split2==null || keyname==null){
                return null;
            }
                        // Use a key/value delimiter.
            String keySplit = keyname+split2;
                        // Traverse the string.
            for (String subStr : str.split(split1)){
                if (subStr.startsWith(keySplit)){
                    if (keySplit.length() < subStr.length()){
                                                // Return the result.
                        return subStr.substring(keySplit.length());
                    }else{
                        return null;
                    }
                }
            }
            // There is no value that corresponds to the key.
            return null;
        } catch (Exception e) {
            return null;
        }
    }
}
```

3.10.2.5.10. Obtain time in a specific format

This topic describes how to use the Evaluate() method to obtain time in a specific format.

UDF used to obtain time in a specific format

```
String UDFGetDate(String date, Long days)
String UDFGetDate(String date, Long months)
```

- Description: used to obtain time at specific intervals. The return value is in the format of yyyyMMdd or yyyy-MM-dd.
- Parameters:
  - date: the time, which is in the format of yyyyMMddHHmmss, yyyyMMdd, yyyy-MM-dd, or yyyy-MM-dd HH:mm:ss.
  - days: the number of days between two time points. The value can be positive or negative.

- months: the number of months between two time points. If the value is 2, add two months to the specified month. If the value is -2, subtract two months from the specified month.

UDF example

- Function registration

  After *UDFGetDate.java* passes the test, register it as a function.

  > **Note** To publish a UDF to a server for production use, the UDF needs to go through packaging, uploading, and registration. You can use the one-click publish function to complete these steps. MaxCompute Studio allows you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence. For more information, see Package a Java program, upload the package, and create a MaxCompute UDF.

- Examples

  After the UDF is registered, execute one of the following statements:

  - Example 1

    ```
    select getDateUDF(20140503,-2);
    ```

    The result is as follows:

    ```
    +------+
    | _c0 |
    +------+
    | 20140501 |
    +------+
    ```

  - Example 2

    ```
    select getDateUDF("20100405",2,10);
    ```

    The result is as follows:

    ```
    +------+
    +------+
    | _c0 |
    +------+
    | 2010-06-10 |
    +------+
    ```

  - Example 3

    ```
    select getDateUDF("20100405",2,'first');
    ```

    The result is as follows:

    ```
    +------+
    | _c0 |
    +------+
    | 2010-06-01 |
    +------+
    ---+
    ```

- Example 4

```
select getDateUDF("20100405",-2,"10");
```

The result is as follows:

```
+-----+
| _c0 |
+-----+
| 2010-02-10 |
+-----+---+
```

UDF code example

```
package com.aliyun.odps.examples.udf; // The package name, which can be defined as needed.
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.udf.UDF;
import com.aliyun.odps.udf.annotation.UdfProperty;
@UdfProperty(isDeterministic=true)
public class UDFGetDate extends UDF{
  public String evaluate(String riqi, Long days) {
        try {
                        // Determine the date.
            String year = riqi.substring(0, 2);
            if (!" 18".equals(year) && !" 19".equals(year) && !" 20".equals(year)
                    && !" 21".equals(year)) {
                return "err";
            }
            String date = "";
            if (riqi.endsWith(".0")) {
                riqi = riqi.substring(0, riqi.length() - 2);
            }
            if (riqi.indexOf("-") >= 0) {
                riqi = riqi.substring(0, 10);
                riqi = riqi.replaceAll("-", "");
            } else {
                riqi = riqi.substring(0, 8);
            }
            try {
                                // Process a specific date by adding or subtracting the num
ber of days between two time points.
                date = getDay(riqi, days.intValue());
            } catch (Exception e) {
                return "err";
            }
            return date;
        } catch (Exception e) {
            return "err";
        }
    }
```

```
    }
    public String evaluate(String riqi, Long n, String flag) {
        try {
                        // Determine the date.
            String year = riqi.substring(0, 2);
            if (!" 18".equals(year) && !" 19".equals(year) && !" 20".equals(year)
                    && !" 21".equals(year)) {
                return "err";
            }
        } catch (Exception e) {
            return "err";
        }
        try {
            String date = "";
            if (riqi.endsWith(".0")) {
                riqi = riqi.substring(0, riqi.length() - 2);
            }
            if (riqi.indexOf("-") >= 0) {
                riqi = riqi.substring(0, 10);
                riqi = riqi.replaceAll("-", "");
            } else {
                riqi = riqi.substring(0, 8);
            }
            try {
                                // If flag is set to first, the first day of that month is
returned.
                if ("first".equals(flag)) {
                    date = getMonthStartDate(riqi, n.intValue());
                } else if ("last".equals(flag)) {
                                // If flag is set to last, the last day of that month is ret
urned.
                    date = getMonthEndDate(riqi, n.intValue());
                } else {
                    date = getMonthDate(riqi, n.intValue(), flag);
                }
            } catch (Exception e) {
                return "err";
            }
            return date;
        } catch (Exception e) {
            return "err";
        }
    }
    public String evaluate(Long n) {
        if (null == n)
            return "err";
        return getNmonthAgo(n.intValue());
    }
    private String getMonthDate(String riqi, int n, String flag) {
        Calendar ca = Calendar.getInstance();
        Date dtBegin = new Date();
        try {
            dtBegin = new SimpleDateFormat("yyyyMMdd").parse(riqi);
        } catch (ParseException e1) {
            e1.printStackTrace();
```

```
        }
        int day = Integer.parseInt(flag);
        ca.setTime(dtBegin);
        ca.add(Calendar.MONTH, n);
        ca.set(Calendar.HOUR_OF_DAY, 0);
        ca.set(Calendar.MINUTE, 0);
        ca.set(Calendar.SECOND, 0);
        ca.set(Calendar.DAY_OF_MONTH, day);
        Date firstDate = ca.getTime();
        return ymdFormat(firstDate);
    }
    private String getNmonthAgo(int n) {
        Calendar ca = Calendar.getInstance();
        Date dtBegin = new Date();
        ca.setTime(dtBegin);
        ca.add(Calendar.MONTH, n);
        Date firstDate = ca.getTime();
        return ymdhmsFormat(firstDate);
    }
        // Process a specific date by adding or subtracting the number of days between two
time points.
    private String getDay(String strBeginDate, int n) {
        Date dtBegin = new Date();
        try {
            dtBegin = new SimpleDateFormat("yyyyMMdd").parse(strBeginDate);
        } catch (ParseException e1) {
            e1.printStackTrace();
        }
        Calendar cld = Calendar.getInstance();
        cld.setTime(dtBegin);
        int day = cld.get(Calendar.DAY_OF_YEAR);
        String strYear = strBeginDate.substring(0, 4);
        String strInputPath = "";
        String strDate = "";
        try {
            int nDays = 1;
            cld.setTime(dtBegin);
            cld.set(Calendar.DAY_OF_YEAR, day + n);
            Date dt = cld.getTime();
            strDate = new SimpleDateFormat("yyyyMMdd").format(dt);
            strYear = strDate.substring(0, 4);
            strInputPath = strDate;
        } catch (NumberFormatException e) {
            e.printStackTrace();
        }
        return strInputPath;
    }
        // Apply the flag parameter to a specific date.
    private String getMonthStartDate(String riqi, int n) {
        Calendar ca = Calendar.getInstance();
        Date dtBegin = new Date();
        try {
            dtBegin = new SimpleDateFormat("yyyyMMdd").parse(riqi);
        } catch (ParseException e1) {
```

```
            e1.printStackTrace();
        }
        ca.setTime(dtBegin);
        ca.add(Calendar.MONTH, n);
        ca.set(Calendar.HOUR_OF_DAY, 0);
        ca.set(Calendar.MINUTE, 0);
        ca.set(Calendar.SECOND, 0);
        ca.set(Calendar.DAY_OF_MONTH, 1);
        Date firstDate = ca.getTime();
        return ymdFormat(firstDate);
    }
    private String getMonthEndDate(String riqi, int n) {
        Calendar ca = Calendar.getInstance();
        Date dtBegin = new Date();
        try {
            dtBegin = new SimpleDateFormat("yyyyMMdd").parse(riqi);
        } catch (ParseException e1) {
            e1.printStackTrace();
        }
        ca.setTime(dtBegin);
        ca.add(Calendar.MONTH, n);
        ca.set(Calendar.HOUR_OF_DAY, 23);
        ca.set(Calendar.MINUTE, 59);
        ca.set(Calendar.SECOND, 59);
        ca.set(Calendar.DAY_OF_MONTH, 1);
        ca.add(Calendar.MONTH, 1);
        ca.add(Calendar.DAY_OF_MONTH, -1);
        Date lastDate = new Date(ca.getTime().getTime());
        return ymdFormat(lastDate);
    }
    private String ymdFormat(Date date) {
        if (date == null) {
            return "";
        }
                // Obtain the specified date format.
        DateFormat ymdFormat = new SimpleDateFormat("yyyy-MM-dd");
        return ymdFormat.format(date);
    }
    private String ymdhmsFormat(Date date) {
        if (date == null) {
            return "";
        }
                // Obtain the specified date format.
        DateFormat ymdFormat = new SimpleDateFormat("yyyyMMddHHmmss");
        return ymdFormat.format(date);
    }
}
```

3.10.2.5.11. Obtain a remainder

This topic describes how to use Java UDFs to obtain the remainders after values of different types are divided.

UDFs used to obtain remainders

```
Integer UDFPMod(INTEGER a, INTEGER b)
Long UDFPMod(Long a, Long b)
Double UDFPMod(Double a, Double b)
```

- Description: used to return the remainder by dividing a by b.
- Parameters:
  - a: the dividend, which can be of the INTEGER, LONG, or DOUBLE type
  - b: the divisor, which can be of the INTEGER, LONG, or DOUBLE type

UDF example

- Function registration

  After *UDFPMod.java* passes the test, register it as a function.

  > **Note** To publish a UDF to a server for production use, the UDF needs to go through packaging, uploading, and registration. You can use the one-click publish function to complete these steps. MaxCompute Studio allows you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence. For more information, see Package a Java program, upload the package, and create a MaxCompute UDF.

- Examples

  After the UDF is registered, execute one of the following statements:

  - Example 1

    ```
    select pmodTest(-9L,-4L);
    ```

    The result is as follows:

    ```
    +-----+
    | _c0 |
    +-----+
    | -1  |
    +-----+
    ```

  - Example 2

    ```
    select pmodTest(9.0,-4.0);
    ```

    The result is as follows:

    ```
    +-----+
    | _c0 |
    +-----+
    | -3.0 |
    +-----+
    ```

- Example 3

```
select pmodTest(9,4);
```

The result is as follows:

```
+-----+
| _c0 |
+-----+
| 1   |
+-----+
```

UDF code example

```
package com.aliyun.odps.examples.udf// The package name, which can be defined as needed.
import com.aliyun.odps.udf.UDF;
public class UDFPMod extends UDF {
    // Use data of the LONG type.
    public String evaluate(Long  a,  Long  b)  {
        if  ((a  == null) || (b == null) || b==0L)  {
            return  null;
        }
        Long d=((a % b) + b) % b;
        return d.toString();
    }
    // Use data of the DOUBLE type.
    public String evaluate(Double a, Double b)  {
        if  ((a == null) || (b == null) || b==0.0)  {
            return  null;
        }
        Double d=((a % b) + b) % b;
        return d.toString() ;
    }
    // Use data of the INTEGER type.
    public String evaluate(Integer a, Integer b)  {
        if  ((a == null) || (b == null) || b==0)  {
            return  null;
        }
        Integer d=((a % b) + b) % b;
        return d.toString();
    }
}
```

Note In this example, `toString` is performed on the type of the return value, but the test result is not affected.

UDF unit testing

```
package com.aliyun.odps.examples.udf// The package name, which can be defined as needed.
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestUDFPMod {
    private UDFPMod udf = new UDFPMod();
    @Test
    public void test_null(){
        assertEquals(null, udf.evaluate(null, 1L));
        assertEquals(null, udf.evaluate(1L, null));
        assertEquals(null, udf.evaluate((Long)null, null));
        assertEquals(null, udf.evaluate(null, 1.0));
        assertEquals(null, udf.evaluate(1.0, null));
        assertEquals(null, udf.evaluate((Double)null, null));
    }
    @Test
    public void test_Normal cases(){
        assertEquals(1L, udf.evaluate(9L, 4L));
        assertEquals(3L, udf.evaluate(-9L, 4L));
        assertEquals(-1L, udf.evaluate(-9L, -4L));
        assertEquals(-3L, udf.evaluate(9L, -4L));
        // An error is returned if the code is not added. null is returned if the code is a
dded.
        assertEquals(null, udf.evaluate(9L, 0L));
        assertEquals(1.0, udf.evaluate(9.0, 4.0));
        assertEquals(3.0, udf.evaluate(-9.0, 4.0));
        assertEquals(-1.0, udf.evaluate(-9.0, -4.0));
        assertEquals(-3.0, udf.evaluate(9.0, -4.0));
        // Nan is returned if the code is not added. null is returned if the code is added.
        assertEquals(null, udf.evaluate(9.0, 0.0));
    }
}
```

3.10.2.5.12. Obtain the character at a specific position in a URL

This topic describes how to obtain the character at a specific position in a URL by using the indexOf(), substring(), or lastIndexOf() method.

UDF used to obtain the character at a specific position in a URL

```
String UDFGetHtmQianN(String url, Long n)
```

- Description: used to obtain the character at a specific position in a URL.
- Parameters:
  - url: the URL you want to query. The value must be of the STRING type.
  - n: the length of the character at a specific position. The value must be of the LONG type.

UDF example

- Function registration

  After *UDFGetHtmQianN.java* passes the test, register it as a function.

  > **Note** To publish a UDF to a server for production use, the UDF needs to go through packaging, uploading, and registration. You can use the one-click publish function to complete these steps. MaxCompute Studio allows you to run the `mvn clean package` command, upload a JAR package, and register the UDF in sequence. For more information, see Package a Java program, upload the package, and create a MaxCompute UDF.

- Examples

  After the UDF is registered, execute one of the following statements:

  - Example 1

    ```
    select udfGetHtmQianNTest("http://www.taobao.com", 1);
    ```

    The result is as follows:

    ```
    +-----+
    | _c0 |
    +-----+
    |     |
    +-----+
    ```

    > **Note** The return result is not NULL.

  - Example 2

    ```
    select udfGetHtmQianNTest("http://www.taobao.com/a.htm", 1);
    ```

    The result is as follows:

    ```
    +-----+
    | _c0 |
    +-----+
    | a   |
    +-----+
    ```

  - Example 3

    ```
    select udfGetHtmQianNTest("http://www.taobao.com/a-b-c-d.htm", 3);
    ```

    The result is as follows:

    ```
    +-----+
    | _c0 |
    +-----+
    | b   |
    +-----+
    ```

UDF code example

```
package com.aliyun.odps.examples.udf// The package name, which can be defined as needed.
import com.aliyun.odps.udf.UDF;
public class UDFGetHtmQianN extends UDF {
    public String evaluate(String url, Long n) {
        try {
            // The position of the first occurrence of .htm in the URL string. The value is
of the INT type.
            int index = url.indexOf(".htm");
            if (index < 0) {
                return "";
            }
            // Begin the extraction at position 0, but exclude the character at the index p
osition.
            String a = url.substring(0, index);
            // Return the position of the last occurrence of the forward slash (/) in the U
RL string.
            index = a.lastIndexOf("/");
            // The length of the obtained string is calculated by using the following formu
la: a.length() - (index + 1).
            String b = a.substring(index + 1, a.length());
            // Use an en dash (-) to split string b and obtain a string array.
            String[] c = b.split("-");
            // A value is returned only if c.length is greater than or equal to n.
            if (c.length < n) {
                return "";
            }
            if (n == 0) {
                return "";
            }
            // Return the character that corresponds to a specific subscript in the string
array.
            String d = c[c.length - n.intValue()];
            return d;
        } catch (Exception e) {
            return "err";
        }
    }
}
```

UDF unit testing

```
package com.aliyun.odps.examples.udf// The package name, which can be defined as needed.
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestUDFGetHtmQianN {
    private UDFGetHtmQianN udf = new UDFGetHtmQianN();
    @Test
    public void test_url(){
        Long n = 1L;
        assertEquals("err", udf.evaluate(null, n));
        // An error is returned if the code is not added. null is returned if the code is a
dded.
        assertEquals("", udf.evaluate("", n));
        assertEquals("", udf.evaluate("http://www.taobao.com", n));
        assertEquals("a", udf.evaluate("http://www.taobao.com/a.htm", n));
        assertEquals("b", udf.evaluate("http://www.taobao.com/a-b.htm", n));
    }
    @Test
    public void test_index(){
        String url = "http://www.taobao.com/a-b-c.htm";
        assertEquals("err", udf.evaluate(url, null));
        // An error is returned if the code is not added. null is returned if the code is a
dded.
        assertEquals("err", udf.evaluate(url, -1L));
        assertEquals("", udf.evaluate(url, 0L));
        assertEquals("c", udf.evaluate(url, 1L));
        // An en dash (-) is returned if the code is not added. null is returned if the cod
e is added.
        assertEquals("", udf.evaluate(url, 100L));
    }
}
```

3.10.2.6. Python UDF examples

3.10.2.6.1. Use a Python UDF to process data of a complex data type

This topic describes how to use a Python user-defined function (UDF) to process data of the complex data type ARRAY, MAP, or STRUCT. The operations described in this topic are performed on the MaxCompute client.

Prerequisites

The MaxCompute client is installed and configured. For more information, see Install and configure the MaxCompute client.

Example: Use a Python UDF to convert data from the `array<bigint>` type to the `array<datetime>` type

1. Save the following code as a Python script file and store the file in the `bin` directory of the MaxCompute client. In this example, the following code is saved as a Python script file named

array.py.

```
from odps.udf import annotate
import datetime
@annotate('array<bigint>->array<datetime>')
class ArrayExample:
    def evaluate(self, intput_list):
        output_list = list()
        for item in intput_list:
            t = datetime.datetime.fromtimestamp(item)
            output_list.append(t)
        return output_list
```

2. Log on to the MaxCompute client.
3. Run the following command to add the Python script file as a resource to MaxCompute:

```
add py array.py;
```

The following result is returned:

```
OK: Resource 'array.py' have been created.
```

For more information about commands that can be used to add resources to MaxCompute, see Add resources.

4. Run the following command to register a Python UDF:

```
create function array_udf as 'array.ArrayExample' using 'array.py';
```

Description of the fields in the preceding command:

- `array_udf` is the name of the Python UDF that you registered. The name is subsequently called in an SQL statement.
- In `array.ArrayExample`, `array` represents array.py that is the name of the Python script file, and `ArrayExample` is the name of the class that is defined in the Python script file array.py.

The following result is returned:

```
Success: Function 'array_udf' have been created.
```

For more information about how to register a function, see Create a UDF.

5. Prepare test data and call the Python UDF in an SQL statement. Make sure that the Python UDF can process data as expected.

```
-- Create a test table.
create table array_test(col1 array<bigint>);
-- Insert data into the table.
insert into array_test values (array(1554047999)), (array(1554047989));
-- Call the Python UDF in the following SELECT statement to process data in the table.
set odps.sql.python.version=cp37;
select array_udf(col1) from array_test;
```

The following result is returned:

```
+------+
 _c0
+------+
 [2019-03-31 23:59:49]
 [2019-03-31 23:59:59]
+------+
```

Example: Use a Python UDF to convert data from the `map<string,bigint>` type to the `map<string,datetime>` type

1. Save the following code as a Python script file and store the file in the `bin` directory of the MaxCompute client. In this example, the following code is saved as a Python script file named map.py.

```
from odps.udf import annotate
import datetime
@annotate('map<string,bigint>->map<string,datetime>')
class MapExample:
    def evaluate(self, intput_dict):
        output_dict = dict()
        for key in intput_dict:
            value = intput_dict[key]
            t = datetime.datetime.fromtimestamp(value)
            output_dict[key] = t
        return output_dict
```

2. Log on to the MaxCompute client.
3. Run the following command to add the Python script file as a resource to MaxCompute:

```
add py map.py;
```

The following result is returned:

```
OK: Resource 'map.py' have been created.
```

For more information about commands that can be used to add resources to MaxCompute, see Add resources.

4. Run the following command to register a Python UDF:

```
create function map_udf as 'map.MapExample' using 'map.py';
```

Description of the fields in the preceding command:

- `map_udf` is the name of the Python UDF that you registered. The name is subsequently called in an SQL statement.
- In `map.MapExample`, `map` represents map.py that is the name of the Python script file, and `MapExample` is the name of the class that is defined in the Python script file map.py.

The following result is returned:

```
Success: Function 'map_udf' have been created.
```

For more information about how to register a function, see Create a UDF.

5. Prepare test data and call the Python UDF in an SQL statement. Make sure that the Python UDF can process data as expected.

```
-- Create a test table.
create table map_test(col1 map<string,bigint>);
-- Insert data into the table.
insert into map_test values (map('a', 1554047999)), (map('b',1554047989));
-- Call the Python UDF in the following SELECT statement to process data in the table.
set odps.sql.python.version=cp37;
select map_udf(col1) from map_test;
```

The following result is returned:

```
+------+
 _c0
+------+
 {b:2019-03-31 23:59:49}
 {a:2019-03-31 23:59:59}
+------+
```

Example: Use a Python UDF to convert data from the `struct<input_name:string,input_timestamp:bigint>` type to the `struct<output_name:string,output_time:datetime>` type

1. Save the following code as a Python script file and store the file in the `bin` directory of the MaxCompute client. In this example, the following code is saved as a Python script file named struct.py.

```
from odps.udf import annotate
import datetime, collections
@annotate('struct<input_name:string,input_timestamp:bigint>->struct<output_name:string,
output_time:datetime>')
class StructExample:
    def evaluate(self, intput_namedtuple):
        OutputNamedTuple = collections.namedtuple('output_namedtuple', ['output_name',
'output_time'])
        name_val = intput_namedtuple.input_name
        time_val = datetime.datetime.fromtimestamp(intput_namedtuple.input_timestamp)
        output_namedtuple = OutputNamedTuple(name_val, time_val)
        return output_namedtuple
```

2. Log on to the MaxCompute client.
3. Run the following command to add the Python script file as a resource to MaxCompute:

```
add py struct.py;
```

The following result is returned:

```
OK: Resource 'struct.py' have been created.
```

For more information about commands that can be used to add resources to MaxCompute, see Add resources.

4. Run the following command to register a Python UDF:

```
create function struct_udf as 'struct.StructExample' using 'struct.py';
```

Description of the fields in the preceding command:

- `struct_udf` is the name of the Python UDF that you registered. The name is subsequently called in an SQL statement.
- In `struct.StructExample`, `struct` represents struct.py that is the name of the Python script file, and `StructExample` is the name of the class that is defined in the Python script file struct.py.

The following result is returned:

```
Success: Function 'struct_udf' have been created.
```

For more information about how to register a function, see Create a UDF.

5. Prepare test data and call the Python UDF in an SQL statement. Make sure that the Python UDF can process data as expected.

```
-- Create a test table.
create table struct_test(col1 struct<a:string, b:bigint>);
-- Insert data into the table.
insert into struct_test values (struct('date1',1554047999)), (struct('date2',1554047989
));
-- Call the Python UDF in the following SELECT statement to process data in the table.
set odps.sql.python.version=cp37;
select struct_udf(col1) from struct_test;
```

The following result is returned:

```
+------+
 _c0
+------+
 {output_name:date2, output_time:2019-03-31 23:59:49}
 {output_name:date1, output_time:2019-03-31 23:59:59}
+------+
```

3.10.2.6.2. Reference a file resource

This topic describes how to reference a file resource by using Python user-defined functions (UDFs) on the MaxCompute client.

Prerequisites

Make sure that the following requirements are met:

- The MaxCompute client is installed and configured.

  For more information about how to install and configure the MaxCompute client, see Install and configure the MaxCompute client.

- The file that you want to reference is added to your MaxCompute project as a resource.

  In this example, the test_distcache.txt file is added to a MaxCompute project as a resource. The file contains the following data:

```
1 a
2 b
3 c
4 d
```

For more information about how to add resources, see Add resources.

Sample code for a Python UDF

The following sample code shows how to use a Python file to obtain the data that meets the specific requirements from the test_distcache.txt file.

```
from odps.udf import annotate
from odps.distcache import get_cache_file
@annotate('bigint->string')
class DistCacheExample(object):
    def __init__(self):
        cache_file = get_cache_file('test_distcache.txt')
        kv = {}
        for line in cache_file:
            line = line.strip()
            if not line:
                continue
            k, v = line.split()
            kv[int(k)] = v
        cache_file.close()
        self.kv = kv
    def evaluate(self, arg):
        return self.kv.get(arg)
```

Procedure

1. Save the Sample code for a Python UDF as a Python script and store the Python script in the `bin` directory of the MaxCompute client. In this example, the Python script is named file.py.
2. Start the MaxCompute client.
3. Run the following command to add the Python script file as a MaxCompute resource:

   ```
   add py file.py;
   ```

   The following command output is returned:

   ```
   OK: Resource 'file.py' have been created.
   ```

   For more information about the commands that you can run to add resources, see Add resources.
4. Run the following command to register a Python UDF:

   ```
   create function file_udf as 'file.DistCacheExample' using 'file.py, test_distcache.txt'
   ;
   ```

   Parameter description:

   - `file_udf`: the name of the Python UDF that you want to register. The function is subsequently called in an SQL statement.

- `file.DistCacheExample`: `file` is the name of the file.py script. `DistCacheExample` is the class defined in the file.py script.

The following command output is returned:

```
Success: Function 'file_udf' have been created.
```

For more information about how to register UDFs, see Create a UDF.

5. Construct test data and call the registered function.

```
-- Create a test table.
create table file_table (arg bigint);
-- Insert data into the table.
insert into file_table values (1), (4), (15), (123), (7995);
-- Call the registered function in the SQL statement to obtain the data that meets the
specific requirements from the test_distcache.txt file.
select file_udf(arg) from file_table;
```

The following result is returned:

```
+-----+
 _c0
+-----+
 a
 d
 NULL
 NULL
 NULL
+-----+
```

3.10.2.6.3. Reference a table resource

This topic describes how to reference a table resource by using Python user-defined functions (UDFs) on the MaxCompute client.

Prerequisites

Make sure that the following requirements are met:

- The MaxCompute client is installed and configured.

  For more information about how to install and configure the MaxCompute client, see Install and configure the MaxCompute client.

- The table that you want to reference is added to your MaxCompute project as a resource.

  In the following example, the udf_test table is added to a MaxCompute project as a resource. The table contains the following data:

```
+------------+------+
 col1        col2
+------------+------+
 1           a
 2           b
 4           c
 5           d
+------------+------+
```

For more information about how to add resources, see Add resources.

Sample code for a Python UDF

The following sample code shows how to use a Python UDF to traverse data in the udf_test table to obtain an array.

```
from odps.udf import annotate
from odps.distcache import get_cache_table
@annotate('->string')
class DistCacheTableExample(object):
    def __init__(self):
        self.records = list(get_cache_table('udf_test'))
        self.counter = 0
        self.ln = len(self.records)
    def evaluate(self):
        if self.counter > self.ln - 1:
            return None
        ret = self.records[self.counter]
        self.counter += 1
        return str(ret)
```

Procedure

1. Save the Sample code for a Python UDF as a Python script and store the Python script in the `bin` directory of the MaxCompute client. In this example, the Python script is named table.py.
2. Start the MaxCompute client.
3. Run the following command to add the Python script file as a MaxCompute resource:

   ```
   add py table.py;
   ```

   The following command output is returned:

   ```
   OK: Resource 'table.py' have been created.
   ```

   For more information about the commands that you can run to add resources, see Add resources.
4. Run the following command to register a Python UDF:

   ```
   create function table_udf as 'table.DistCacheTableExample' using 'table.py,udf_test';
   ```

   Parameter description:

   - `table_udf`: the name of the Python UDF that you want to register. The function is subsequently called in an SQL statement.

- `table.DistCacheTableExample`: `table` is the name of the table.py script. `DistCacheTableExample` is the class defined in the table.py script.

The following command output is returned:

```
Success: Function 'table_udf' have been created.
```

For more information about how to register UDFs, see Create a UDF.

5. Construct test data and call the registered function.

```
-- Create a test table.
create table table_test (arg bigint);
-- Insert data into the table.
insert into table_test values (1), (4), (15), (123), (7995);
-- Call the registered function in the SQL statement.
select table_udf() from table_test;
```

The following result is returned:

```
+------+
 _c0
+------+
 (4, 'c')
 (5, 'd')
 (1, 'a')
 (2, 'b')
 NULL
+------+
```

3.10.2.6.4. Reference third-party packages in Python UDFs

MaxCompute allows you to reference third-party packages in Python user-defined functions (UDFs), such as NumPy packages, third-party packages that need to be compiled, and third-party packages that are dependent on dynamic-link libraries (DLLs). This topic describes how to reference third-party packages in Python UDFs.

Context

You can use one of the following methods to reference third-party packages in a Python UDF:

- Reference NumPy packages in Python 3 UDFs

  You must change the file name extension of the NumPy package, use the MaxCompute client to upload the NumPy package, and then register the package. After you register the package, a UDF is created. You can call the UDF after you create the UDF.

- Reference third-party packages that need to be compiled

You must compile the setup.py script in a third-party package, generate a wheel package, and then change the file name extension of the wheel package in an environment that is compatible with MaxCompute. Then, use the MaxCompute client to upload the wheel package and register the package. After you register the package, a UDF is created. You can call the UDF after you create the UDF. We recommend that you use a Linux operating system. If you use a Windows operating system, we recommend that you use Docker.

- Reference third-party packages that are dependent on DLLs

  You must compile the .so library file based on the source code of a third-party package, generate a wheel package, and then change the file name extension of the wheel package. Then, use the MaxCompute client to upload the wheel package and the .so library file and register the package and the file. After you register the package and the file, a UDF is created. You can call the UDF after you create the UDF.

Prerequisites

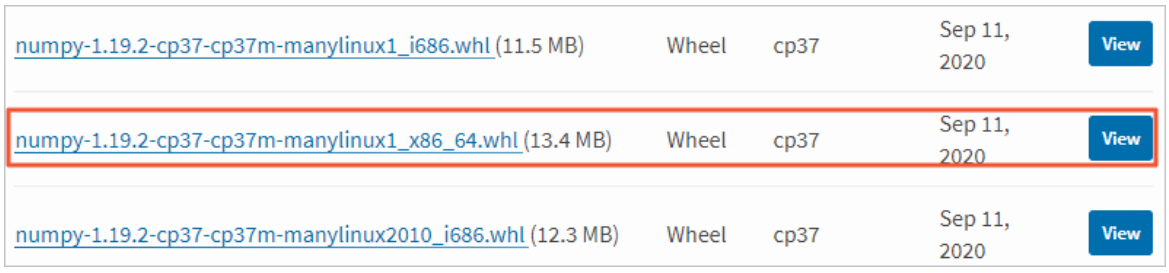
Make sure that the following prerequisites are met:

- Python is installed. We recommend that you install Python 3.
- The MaxCompute client is installed and configured. For more information, see Install and configure the MaxCompute client.
- pip, setuptools, and wheel are installed if you want to use Python UDFs to reference third-party packages that need to be compiled. You can run the `pip install setuptools` command to install setuptools and run the `pip install wheel` command to install wheel.
- PROJ 6 is installed if you use a third-party package of GDAL 3.0 or later.
- Docker is installed if you use Docker to compile third-party packages. For more information, see Docker documentation.

Reference NumPy packages in Python 3 UDFs

You can use Python 3 in MaxCompute to reference NumPy packages. By default, the NumPy library is installed in Python 2 in MaxCompute. You do not need to manually upload NumPy packages in Python 2. To reference a NumPy package in Python 3 UDFs, perform the following steps:

1. In the **Download files** section of the PyPI page, click the package whose name ends with cp37-cp37m-manylinux1_x86_64.whl to download the package. In this example, NumPy 1.19.2 is used.



   > **Note** If you download a package whose name ends with other characters, the operation may fail. If you need to select another version of the NumPy package, click **Release history** in the **Navigation** section in the upper-left corner of the PyPI page to view the historical versions.

2. Change the file name extension of the downloaded NumPy package to .zip.

   Example: *numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip*.

3. Use the MaxCompute client to upload the NumPy package to your MaxCompute project. For more information about how to upload the package, see Resource operations.

   Sample command:

   ```
   ADD ARCHIVE D:\Downloads\numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip -f;
   ```

4. Write a Python UDF script and save it as a PY file.

   In this example, the saved file is named import_numpy.py. The following code shows the Python UDF script:

   ```
   from odps.udf import annotate
   @annotate("->string")
   class TryImport(object): # The class name is TryImport.
       def __init__(self):
           import sys
           sys.path.insert(0, 'work/numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip') # The
   NumPy package. You need only to change the package name after work/.
       def evaluate(self):
           import numpy
           return "import succeed"
   ```

5. Use the MaxCompute client to upload the import_numpy.py script to your MaxCompute project as a resource.

   Sample command:

   ```
   ADD PY D:\Desktop\import_numpy.py -f;
   ```

6. Use the uploaded import_numpy.py script and NumPy package to create a UDF on the MaxCompute client. For more information about how to create a UDF, see Function operations.

   In this example, the created UDF is named numpy. Sample command:

   ```
   CREATE FUNCTION numpy AS 'import_numpy.TryImport' USING 'doc_test_dev/resources/import_
   numpy.py,numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip';
   ```

   > **Note** When you create a UDF, you must add a NumPy package, such as *numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip*, to the resource list.

7. After you register the UDF, you can call the UDF in SQL statements. Make sure that Python 3 is enabled to execute SQL statements. For more information, see Python 3 UDFs.

Reference third-party packages that need to be compiled

If a third-party package is a TAR.GZ package that is downloaded from PyPI or a source code package that is downloaded from GitHub, the setup.py file may be stored in the root directory of the decompressed third-party package. To use this type of third-party package, you must compile the setup.py file and generate a wheel package in an environment that is compatible with MaxCompute. Then, upload the package as a resource and register the package as a UDF. After the UDF is created, you can call third-party packages in Python UDFs. For more information about how to upload a resource and create a UDF, see Reference NumPy packages in Python 3 UDFs.

> **Notice**
>
> - Third-party packages run in a Linux operating system. We recommend that you compile third-party packages in a Linux operating system. If you compile third-party packages in a Windows operating system, compatibility issues may occur.
> - If you use a Windows operating system, we recommend that you use Python of the required version to compile the setup.py file and generate a wheel package in the Docker container created from the quay.io/pypa/manylinux2010_x86_64 image. Python of the required version is stored in `/opt/python/cp27-cp27m/bin/python` or `/opt/python/cp37-cp37m/bin/python3`.

If you use a Linux operating system, make sure that the following requirements are met:

- A Python version that is compatible with MaxCompute is used. You can run the following command in the command-line interface (CLI) of your system to check the Python version:

  ```
  python -c "import wheel.pep425tags; print(wheel.pep425tags.get_abi_tag())"
  ```

  - If `cp27m` or `cp37m` is returned, the version meets the compatibility requirements.
  - If `cp27mu` or `cp37mu` is returned, the version does not meet the compatibility requirements. In this case, you must run the `./configure --enable-unicode=ucs2` command to change the Python encoding format to UCS-2.
- If code in C or C++ is required, your Linux operating system must be compatible with the GNU Compiler Collection (GCC) version in use.

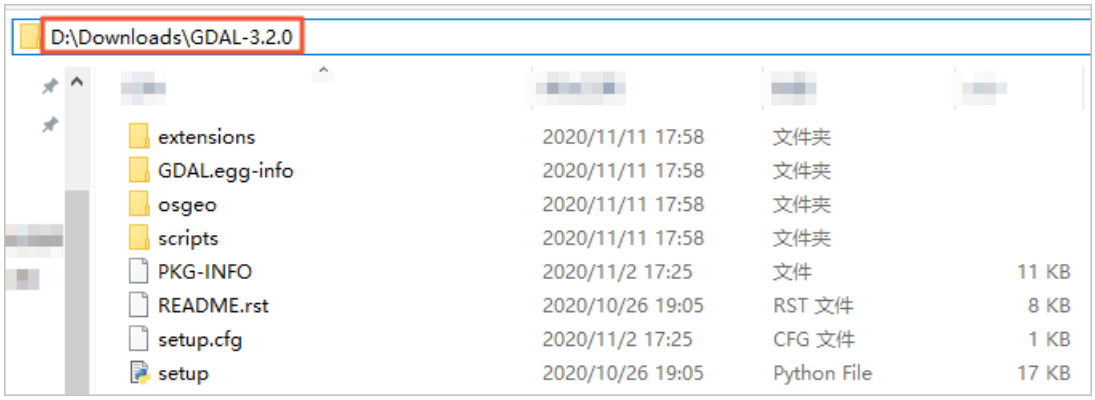
  > **Note** We recommend that you use GCC 4.9.2 or earlier. If the GCC version is later than 4.9.2, the .so file in the generated wheel package may be incompatible with MaxCompute.

If all requirements are met, perform the following steps to compile the setup.py file and generate a wheel package:

1. Decompress a third-party package to your on-premises machine and run the required command in the CLI to go to the path where the setup.py file is stored.

   For example, the *GDAL-3.2.0.zip* package is downloaded. After you decompress the package, the setup.py file is stored in *D:\Downloads\GDAL-3.2.0*. Sample command:

   ```
   cd D:\Downloads\GDAL-3.2.0
   ```

2. Run the following command in the CLI to check whether bdist_wheel is returned:

Sample command:

```
python setup.py --help-command
```

- If yes, go to Step 3.
- If no, change `from distutils.core import setup` to `from setuptools import setup` in the setup.py file. Then, go to Step 3.

3. Run the following command in the CLI to compile the setup.py file and generate a wheel package:

```
python setup.py bdist_wheel
```

> **Note** The wheel package is stored in the dist folder.

Reference third-party packages that are dependent on DLLs

Some third-party packages for Python depend on Python libraries and other DLLs. This section describes how to use the Docker container to compile the .so library file and generate a wheel package that can be used in MaxCompute. The container is created from the quay.io/pypa/manylinux2010_x86_64 image. GDAL 3.0.4 is used in this example. You must upload the generated .so library file, wheel package, or NumPy package as resources and register the file and the packages as a UDF. After the UDF is created, you can call third-party packages in Python UDFs. For more information about how to upload a resource and create a UDF, see Reference NumPy packages in Python 3 UDFs.

> **Note** Make sure that Docker is installed before you can reference third-party packages that are dependent on DLLs in Python UDFs. For more information, see Docker documentation.

To reference third-party packages that are dependent on DLLs in Python UDFs, perform the following steps:

1. View the dependencies in the **Dependencies** section of the PyPI page.

   The following figure shows the dependencies of GDAL 3.0.4.

   Dependencies

   - *libgdal (3.0.4 or greater) and header files (gdal-devel)*
   - *numpy (1.0.0 or greater) and header files (numpy-devel) (not explicitly required, but many examples and utilities will not work without it)*

   > **Note** In the preceding figure, the dependencies include *libgdal* and *numpy*. To obtain *libgdal*, compile the GDAL source code in the Docker container. To obtain *numpy*, obtain the NumPy package on the PyPI page or from the Docker container.

2. Obtain the NumPy package.

   You can use one of the following methods to obtain the NumPy package:

   - In the **Download files** section of the PyPI page, click the package whose name ends with *cp37-cp37m-manylinux1_x86_64.whl* to download the package.

> **Note** If Python 2 is used, perform the following operations to download the NumPy package: In the **Navigation** section of the PyPI page, click **Release history**, select 1.16.6 or an earlier version, and then click the package whose name ends with *cp27-cp27m-manylinux1_x86_64.whl*.

- Run the `/opt/python/cp37-cp37m/bin/pip download numpy -d ./` command to download the Numpy package to the current directory.

3. Compile the .so library file.
   i. Download the GDAL 3.0.4 source code file and decompress it to your on-premises machine.
   ii. Download the Docker container created from the quay.io/pypa/manylinux2010_x86_64 image and enter the input mode of the Docker client.

   Sample commands:

   ```
   docker pull quay.io/pypa/manylinux2010_x86_64
   docker run -it quay.io/pypa/manylinux1_x86_64 /bin/bash
   ```

   iii. Upload the GDAL 3.0.4 source code to the Docker container.

   Sample command:

   ```
   docker cp ./gdal-3.0.4 <CONTAINER ID>:/opt/source/
   ```

   For more information about how to obtain CONTAINER ID, see docker ps.
4. Compile GDAL 3.0.4 source code in the container. For more information, see BuildingOnUnix.

   Sample commands:

   ```
   # Specify the directory to install PROJ 6 in the configure field.
   ./configure --prefix=/path/to/install/prefix --with-proj=/path/to/install/proj6/prefix
   make
   make install
   export PATH=/path/to/install/prefix/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/install/prefix/lib:$LD_LIBRARY_PATH
   export GDAL_DATA=/path/to/install/prefix/share/gdal
   # Test
   gdalinfo --version
   ```

   The following errors may occur during compilation:

   - `configure: error: PROJ 6 symbols not found`: If this error occurs, install PROJ 6 to support GDAL 3.0 or later.
   - `fatal error: zlib.h: No such file or directory`: If this error occurs, use the `yum install zlib-devel` command instead.
5. Run the Docker download commands to download two .so library files (not symbolic links) to your on-premises machine. Obtain *libgdal.so* from the lib folder in the installation directory of GDAL and *libproj.so* from the lib folder in the installation directory of PROJ 6.
6. Generate a GDAL wheel package in the Docker container. For more information, see BuildingOnUnix.

   Sample commands:

```
# If NumPy is required, install NumPy first.
/opt/python/cp37-cp37m/bin/pip install numpy
# Switch to the directory in which GDAL source code is saved.
cd swig/python
# Generate a wheel package and save it in the dist folder. Example: GDAL-3.0.4-cp37-cp3
7m-linux_x86_64.whl
/opt/python/cp37-cp37m/bin/python setup.py bdist_wheel
```

7. Upload the generated .so library file, wheel package, or NumPy package as resources and register the file and the packages as a UDF. After the UDF is created, you can call third-party packages in Python UDFs. For more information about how to upload a resource and create a UDF, see Reference NumPy packages in Python 3 UDFs.

   Take note of the following items when you upload a resource and create a UDF:

   - When you upload resources, you must upload *libgdal.so* and *libproj.so* as file resources and *numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip* and *GDAL-3.0.4-cp37-cp37m-linux_x86_64.zip* as archive resources.
   - When you create functions, you must add *libgdal.so*, *libproj.so*, *numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip*, and *GDAL-3.0.4-cp37-cp37m-linux_x86_64.zip* to the resource list of the functions.

   Sample code for a Python UDF:

```
# coding: utf-8
from odps.udf import annotate
from odps.distcache import get_cache_file
def include_file(file_name):
    import os, sys
    so_file = get_cache_file(file_name, 'b')
    with open(so_file.name, 'rb') as fp:
        content=fp.read()
        so = open(file_name, "wb")
        so.write(content)
        so.flush()
        so.close()
@annotate("->string")
class TryImport(object):
    def __init__(self):
        import sys
        include_file('libgdal.so.26')
        include_file('libproj.so.15')
        sys.path.insert(0, 'work/GDAL-3.0.4-cp37-cp37m-linux_x86_64.zip') # The GDAL pa
ckage after compilation. You need only to change the package name that follows work/.
        sys.path.insert(0, 'work/numpy-1.19.2-cp37-cp37m-manylinux1_x86_64.zip') # The
NumPy package. You need only to change the package name after work/.
    def evaluate(self):
        from osgeo import gdal
        from osgeo import ogr
        from osgeo import osr
        from osgeo import gdal_array
        from osgeo import gdalconst
        return "import succeed"
```

> **Note** If an error that indicates *libgdal.so.26* or *libproj.so.15* cannot be found occurs, you must change *libgdal.so* to *libgdal.so.26* or *libproj.so* to *libproj.so.15*.

3.10.3. UDTF

3.10.3.1. Overview

MaxCompute allows you to write user-defined table-valued functions (UDTFs) in Java or Python to extend the capabilities of MaxCompute functions and accommodate your business requirements. This topic describes the types, limits, usage notes, and development process of UDTFs. This topic also describes how to use UDTFs.

Background information

You can use UDTFs to return multiple values for a single data input. The input and output data of a UDTF have a one-to-many relationship. Each time a UDTF reads a row of data, the UDTF returns multiple values. The returned values are considered a table. MaxCompute allows you to write UDTFs in Java or Python.

UDTF type	Description
Java UDTF	MaxCompute allows you to write UDTFs in Java. For more information, see Java UDTFs.
Python UDTF	MaxCompute allows you to write UDTFs in Python 2 and Python 3. • Python 2 UDTFs: The Python version is 2.7. For more information, see Python 2 UDTF. • Python 3 UDTFs: The Python version is CPython-3.7.3. For more information, see Python 3 UDTFs.

The built-in functions of MaxCompute include some UDTFs, such as EXPLODE. For more information about built-in UDTFs, see Other functions or Complex type functions.

Limits

- You cannot access the Internet by using user-defined functions (UDFs). If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.
- If you use a UDTF in a `SELECT` statement, you cannot specify other columns or use other expressions in this statement. The following **sample code** shows an incorrect SQL statement.

```
-- The statement contains a UDTF and another column.
select value, user_udtf(key) as mycol ...
```

- UDTFs cannot be nested. The following **sample code** shows an incorrect SQL statement.

```
-- A UDTF named user_udtf2 is nested in a UDTF named user_udtf1.
select user_udtf1(user_udtf2(key)) as mycol...;
```

- A UDTF cannot be used with a `GROUP BY`, `DISTRIBUTE BY`, or `SORT BY` clause in the same `SELECT` statement. The following **sample code** shows an incorrect SQL statement.

```
-- A UDTF is used together with a GROUP BY clause.
select user_udtf(key) as mycol ... group by mycol;
```

Usage notes

Before you use UDFs, take note of the following items:

- UDFs cannot compete with built-in functions in performance. We recommend that you preferentially use built-in functions to implement your business logic.
- If you use a UDF in SQL statements, the memory usage of a computing job may exceed the default allocated memory size if a large amount of data is computed and data skew occurs. In this case, you can run the `set odps.sql.udf.joiner.jvm.memory=xxxx;` command at the session level to resolve the issue. For more information about the MaxCompute UDF FAQ, see FAQ about MaxCompute UDFs.
- If the name of a UDF is the same as that of a built-in function, the UDF is preferentially called. For example, if UDF CONCAT and built-in function CONCAT both exist in MaxCompute, the system automatically calls UDF CONCAT instead of the built-in function CONCAT. If you want to call the built-in function, you must add the symbol `::` before the built-in function, for example, `select ::concat('ab', 'c');`.

Development process

The following figure shows how to write a MaxCompute UDTF in Java and Python.

- The following figure demonstrates how to write a MaxCompute UDF in Java.

No.	Required	Description	Platform	References

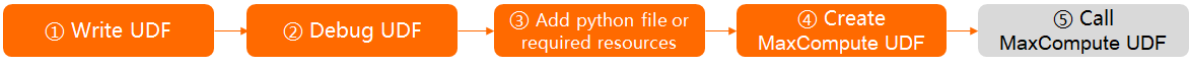
No.	Required	Description	Platform	References

1	No	Before you can use Maven to write code, you must add the related SDK dependencies to the POM file. This ensures that the code can be compiled. The following SDK dependency shows an example:
		`<dependency>`
		`<groupId>com.aliyun.odps</groupId>`
		`<artifactId>odps-sdk-udf</artifactId>`
		`<version>0.29.10-public</version>`
		`</dependency>`
		You can search for `odps-sdk-udf` from Maven repositories to obtain the version of the SDK dependency.
			IntelliJ IDEA (Maven)	None

2	Yes	Write a UDF based on your business requirements.	IntelliJ IDEA (Maven) and MaxCompute Studio	Develop a UDF in Java
3	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		
4	Yes	Debug the UDF code to ensure that the code is packaged into a JAR file after it is successfully run on your on-premises machine.		

No.	Required	Description	Platform	References
5	Yes	Upload the JAR file as a resource to your MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Package a Java program, upload the package, and create a MaxCompute UDF ◦ DataWorks ▪ Create a JAR resource ▪ Register a MaxCompute function
6	Yes	Create a UDF based on the JAR file that you uploaded.		
7	No	Call the UDF in the query data code.		None

- The following figure demonstrates how to write a MaxCompute UDF in Python.



No.	Required	Description	Platform	References
1	Yes	Write a UDF based on your business requirements.	MaxCompute Studio	Develop a Python UDF
2	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		

No.	Required	Description	Platform	References
3	Yes	Upload Python files or required resources, such as file resources, table resources, and third-party packages, to a MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Upload a Python program and create a MaxCompute UDF ◦ DataWorks Create a Python resource and register a UDF
4	Yes	Create a UDF based on the uploaded Python files or required resources.		
5	No	Call the UDF in the query data code.		None

Instructions

Use the following methods to call UDFs:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

3.10.3.2. Java UDTFs

This topic describes how to write a user-defined table-valued function (UDTF) in Java.

UDTF code structure

You can use Maven in IntelliJ IDEA or MaxCompute Studio to write UDTF code in Java. The UDTF code can contain the following information:

- Java package: optional.

  You can package defined Java classes into a file for future use.
- Base UDTF classes: required.

The following base UDTF classes must be included: `com.aliyun.odps.udf.UDTF`, `com.aliyun.odps.udf.annotation.Resolve`, and `com.aliyun.odps.udf.UDFException`. com.aliyun.odps.udf.annotation.Resolve specifies the `@Resolve` annotation, and com.aliyun.odps.udf.UDFException specifies the method that is used to implement Java classes. If you need to use other UDTF classes or complex data types, add the required classes by following the instructions provided in MaxCompute SDK.

- Custom Java class: required.

  A custom Java class is the organizational unit of UDTF code. This class defines the variables and methods that are used to meet your business requirements.

- `@Resolve` annotation: required.

  The annotation is in the `@Resolve(<signature>)` format. The `signature` is a function signature that defines the data types of input parameters and return values of a UDTF. You cannot obtain function signatures for UDTFs by using the reflection feature. You can obtain a function signature only by using a `@Resolve` annotation, such as `@Resolve("smallint->varchar(10)")`. For more information about `@Resolve` annotations, see @Resolve annotations.

- Methods to implement Java classes: required.

  The following table describes the methods that can be used to implement Java classes. You can select one of the methods based on your business requirements.

Method	Description
`public void setup(ExecutionContext ctx) throws UDFException`	The initialization method. Before a UDTF processes the input data, MaxCompute calls the user-defined initialization behavior. `setup` is called once for each worker.
`public void process(Object[] args) throws UDFException`	`process` is called once for each SQL record. The parameters of `process` are the input parameters of the UDTF that is specified in SQL statements. The input parameters are passed in the process function as `Object[]`, and the results are returned by using the `forward` function. You must call the `forward` function in the `process` function to determine the output data.
`public void close() throws UDFException`	The method to terminate a UDTF. This method is called only once. It is called only after the last record is processed.
`public void forward(Object …o) throws UDFException`	You can call the `forward` method to return data. One record is generated each time the `forward` function is called. When you call a UDTF in an SQL query statement, you can use the `AS` clause to rename the output of the `forward` method.

  You can use Java data types or Java writable types to write a Java UDTF. For more information about the mappings among the data types that are supported by MaxCompute projects, Java data types, and Java writable types, see Data types.

The following example shows the UDTF code.

```
// Package Java classes into a JAR file named org.alidata.odps.udtf.examples.
package org.alidata.odps.udtf.examples;
// The base UDTF classes.
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDTFCollector;
import com.aliyun.odps.udf.annotation.Resolve;
import com.aliyun.odps.udf.UDFException;
// The custom Java class.
// The @Resolve annotation.
@Resolve("string,bigint->string,bigint")
public class MyUDTF extends UDTF {
     // The method that is used to implement the Java classes.
     @Override
     public void process(Object[] args) throws UDFException {
         String a = (String) args[0];
         Long b = (Long) args[1];
         for (String t: a.split("\\s+")) {
         forward(t, b);
       }
     }
   }
```

Limits

- You cannot access the Internet by using user-defined functions (UDFs). If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.
- If you use a UDTF in a `SELECT` statement, you cannot specify other columns or use other expressions in this statement. The following **sample code** shows an incorrect SQL statement.

  ```
  -- The statement contains a UDTF and another column.
  select value, user_udtf(key) as mycol ...
  ```

- UDTFs cannot be nested. The following **sample code** shows an incorrect SQL statement.

  ```
  -- A UDTF named user_udtf2 is nested in a UDTF named user_udtf1.
  select user_udtf1(user_udtf2(key)) as mycol...;
  ```

- A UDTF cannot be used with a `GROUP BY` , `DISTRIBUTE BY` , or `SORT BY` clause in the same `SELECT` statement. The following **sample code** shows an incorrect SQL statement.

  ```
  -- A UDTF is used together with a GROUP BY clause.
  select user_udtf(key) as mycol ... group by mycol;
  ```

Usage notes

When you write a Java UDTF, take note of the following points:

- We recommend that you do not package classes that have the same name but different logic into the JAR files of different UDTFs. For example, the JAR file of UDTF 1 is named udtf1.jar and the JAR file

of UDTF 2 is named udtf2.jar. Both packages contain a class named `com.aliyun.UserFunction.class`, but the class has different logic. If UDTF 1 and UDTF 2 are called in the same SQL statement, MaxCompute loads the com.aliyun.UserFunction.class from one of the two files. As a result, the UDTFs cannot run as expected and a compilation error may occur.

- The data type of an input parameter or a return value in a Java UDTF is an object. The first letter of the data types that you specify in the Java UDTF code must be in uppercase, such as String.
- NULL values in SQL are represented by NULL in Java. Primitive data types in Java cannot represent NULL values in SQL. Therefore, these data types cannot be used.

@Resolve annotations

`@Resolve` annotation format:

```
@Resolve(<signature>)
```

`signature` is a function signature string. This parameter is used to identify the data types of the input parameters and return values. When a UDTF is run, the input parameters and return values of the UDTF must be of the same data type as those specified in the function signature. The system checks whether the UDTF complies with the definition of the function signature during semantics parsing. If the data types of the UDTF are inconsistent with the data types specified in the function signature, an error is returned. The signature is in the following format:

```
'arg_type_list -> type_list'
```

Parameter description:

- `type_list`: indicates the data types of return values. A UDTF can return multiple columns. The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.
- `arg_type_list`: indicates the data types of input parameters. If multiple input parameters are used, specify multiple data types and separate them with commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, VARCHAR, complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

  `arg_type_list` can also be set to an asterisk (*) or left empty.

  - If `arg_type_list` is set to an asterisk (*), a random number of input parameters are used.
  - If `arg_type_list` is left empty, no input parameters are used.

The following table provides examples of `@Resolve` annotations.

@Resolve annotation	Description
`@Resolve('bigint,boolean->string,datetime')`	The data types of input parameters are BIGINT and BOOLEAN, and the data types of return values are STRING and DATETIME.
`@Resolve('*->string, datetime')`	A random number of input parameters are used. The data types of return values are STRING and DATETIME.

@Resolve annotation	Description
`@Resolve('->double, bigint, string')`	No input parameters are used. The data types of return values are DOUBLE, BIGINT, and STRING.
`@Resolve("array<string>,struct<a1:bigint,b1:string>,string->map<string,bigint>,struct<b1:bigint>")`	The data types of input parameters are ARRAY, STRUCT, and MAP. The data types of return values are MAP and STRUCT.

Data types

In MaxCompute, different data type editions support different data types. In MaxCompute V2.0 and later, more data types and complex data types, such as ARRAY, MAP, and STRUCT, are supported. For more information about MaxCompute data type editions, see Data type editions.

The following table describes the mappings among the data types that are supported by MaxCompute projects, Java data types, and Java writable types. You must write Java UDTFs based on the mappings to ensure data type consistency.

MaxCompute data type	Java data type	Java writable data type
TINYINT	java.lang.Byte	ByteWritable
SMALLINT	java.lang.Short	ShortWritable
INT	java.lang.Integer	IntWritable
BIGINT	java.lang.Long	LongWritable
FLOAT	java.lang.Float	FloatWritable
DOUBLE	java.lang.Double	DoubleWritable
DECIMAL	java.math.BigDecimal	BigDecimalWritable
BOOLEAN	java.lang.Boolean	BooleanWritable
STRING	java.lang.String	Text
VARCHAR	com.aliyun.odps.data.Varchar	VarcharWritable
BINARY	com.aliyun.odps.data.Binary	BytesWritable
DATETIME	java.util.Date	DatetimeWritable
TIMESTAMP	java.sql.Timestamp	TimestampWritable
INTERVAL_YEAR_MONTH	N/A	IntervalYearMonthWritable
INTERVAL_DAY_TIME	N/A	IntervalDayTimeWritable
ARRAY	java.util.List	N/A
MAP	java.util.Map	N/A

MaxCompute data type	Java data type	Java writable data type
STRUCT	com.aliyun.odps.data.Struct	N/A

Note You can use Java writable types for the input parameters or return values of UDTFs only when your MaxCompute project uses the MaxCompute V2.0 data type edition.

Instructions

After you develop a Java UDTF by following the instructions in Development process, you can use MaxCompute SQL to call the Java UDTF. You can use one of the following methods to call the Java UDTF:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

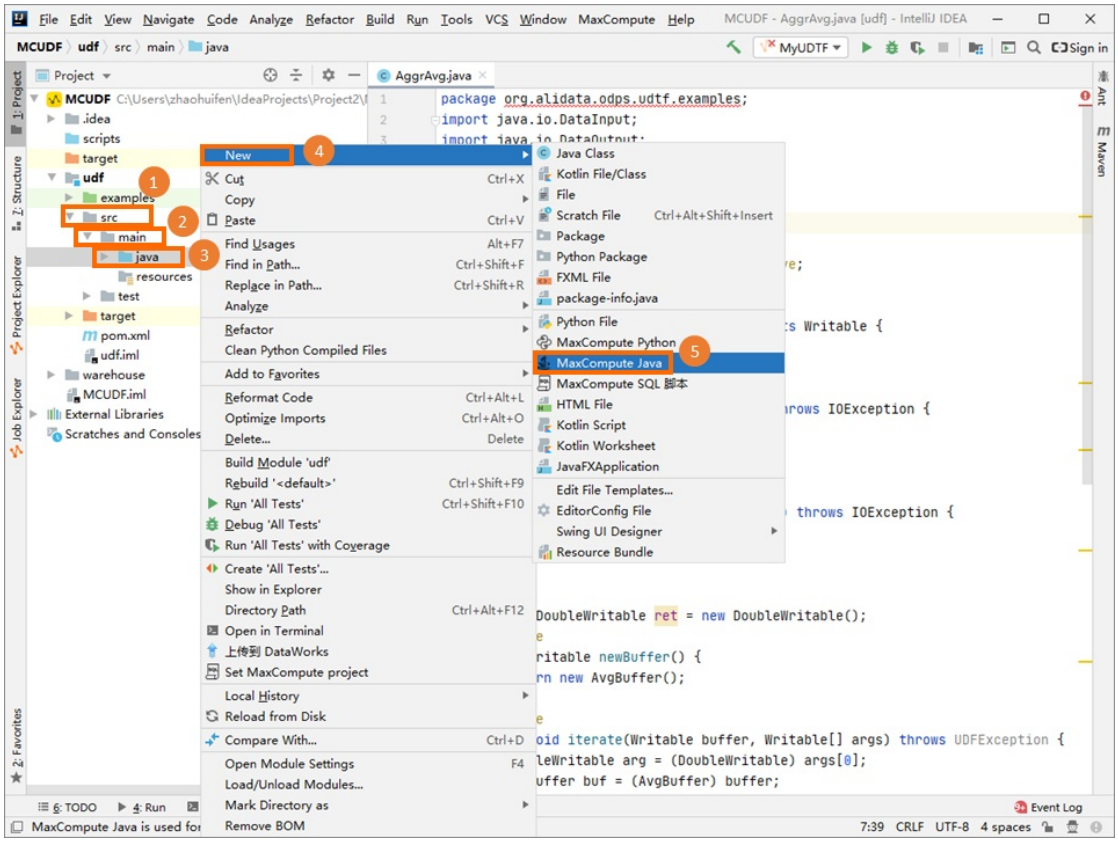
For more information about how to use MaxCompute Studio to develop and call a Java UDTF, see Example.

Example

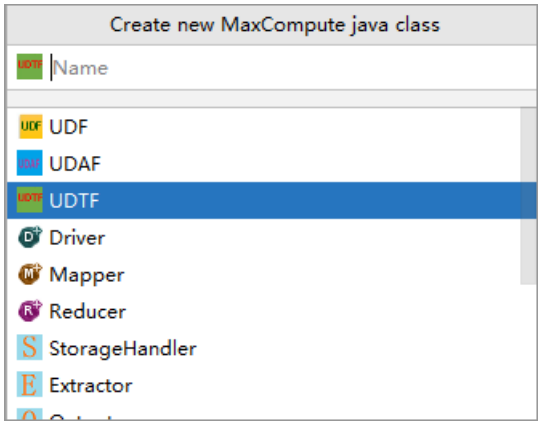
This example describes how to use MaxCompute Studio to develop and call a Java UDTF.

1. Make the following preparations on IntelliJ IDEA:
   i. Install MaxCompute Studio.
   ii. Establish a connection to a MaxCompute project.
   iii. Create a MaxCompute Java module.
2. Write UDTF code.

i. In the left-side navigation pane of the **Project** tab, choose **src > main > java**, right-click java, and then choose **New > MaxCompute Java**.

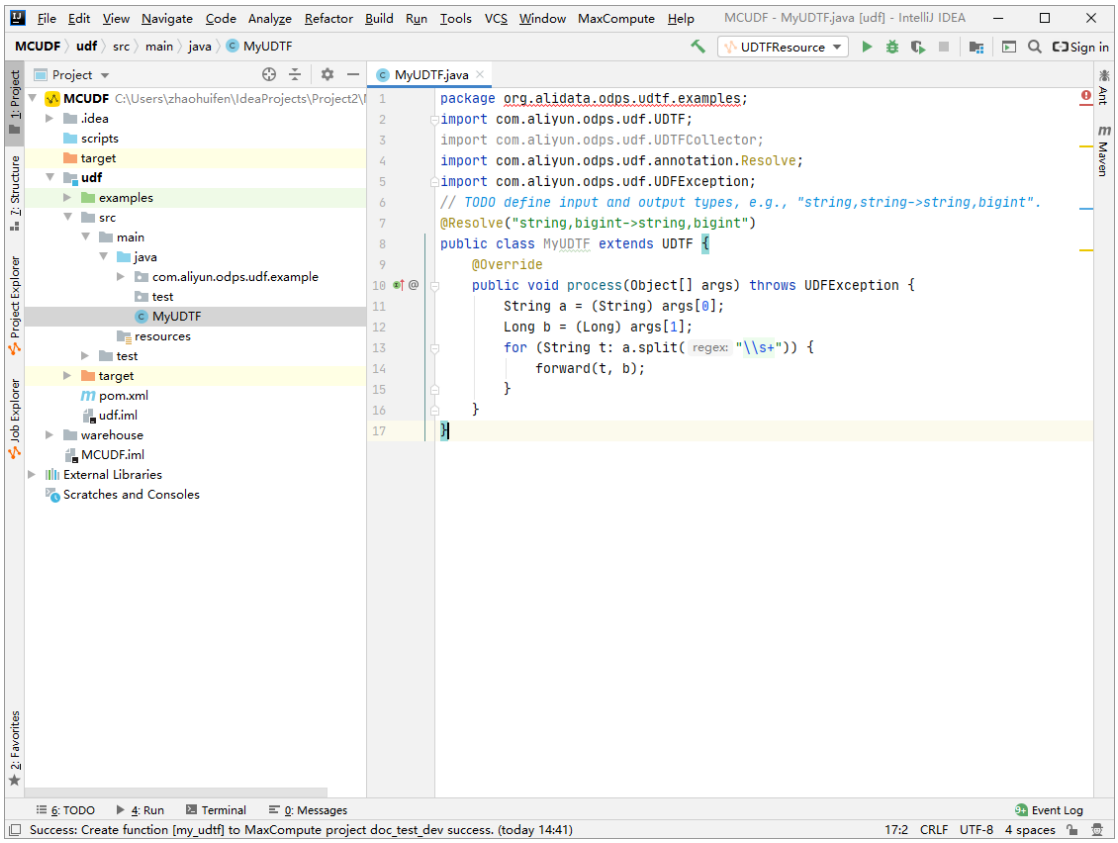


ii. In the **Create new MaxCompute java class** dialog box, click **UDTF**, enter a name in the **Name** field, and then press Enter. In this example, the Java class is named MyUDTF.



Name is the name of the MaxCompute Java class. If no package is created, enter packagename.classname. The system automatically creates a package.

iii. Write code in the code editor.

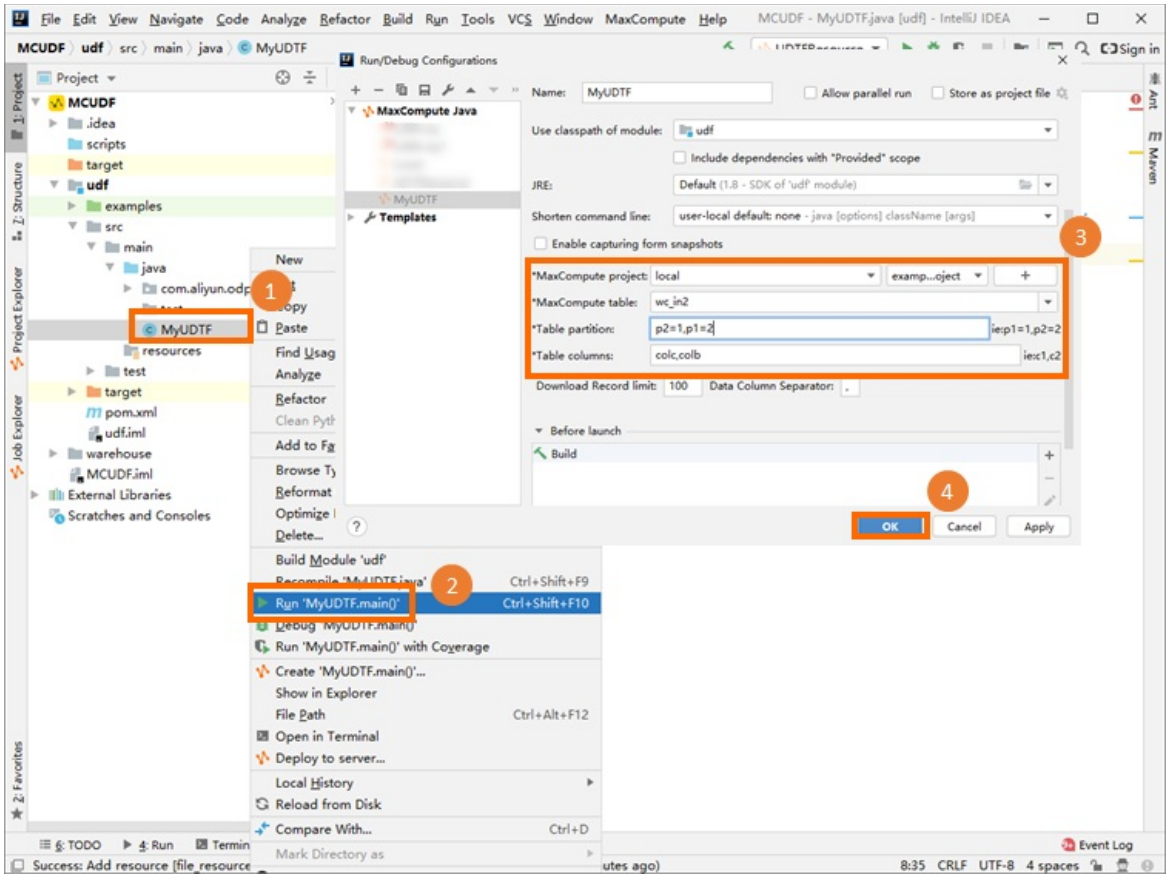


The following example shows the UDTF code.

```
package org.alidata.odps.udtf.examples;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDTFCollector;
import com.aliyun.odps.udf.annotation.Resolve;
import com.aliyun.odps.udf.UDFException;
// TODO define input and output types, e.g., "string,string->string,bigint".
   @Resolve("string,bigint->string,bigint")
   public class MyUDTF extends UDTF {
     @Override
     public void process(Object[] args) throws UDFException {
       String a = (String) args[0];
       Long b = (Long) args[1];
       for (String t: a.split("\\s+")) {
         forward(t, b);
       }
     }
   }
```

3. Debug the UDTF on your on-premises machine to ensure that the code can run successfully.

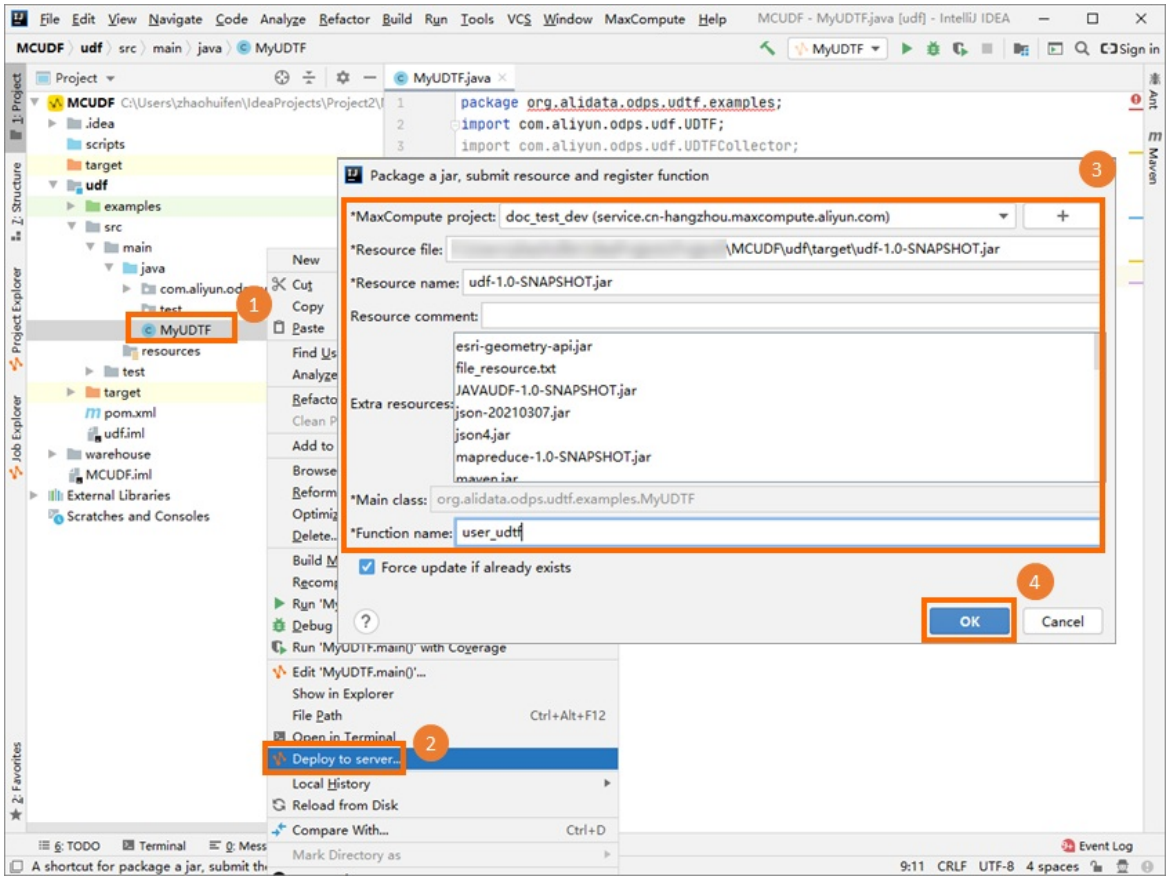
   For more information about debugging operations, see Perform a local run to debug the UDF.



Note You can configure parameters based on the values in the preceding figure.

4. Package the created UDTF into a JAR file, upload the file to your MaxCompute project, and then register the UDTF. In this example, the function name is `user_udtf`.

   For more information about how to package a UDTF, see Procedure.



5. In the left-side navigation pane of MaxCompute Studio, click **Project Explorer**. Right-click your MaxCompute project, select Open in Console from the drop-down list to start the MaxCompute client, and then execute the SQL statement to call the new UDTF.

   The following example shows the data structure of the my_table table that you want to query.

```
+------------+------------+
| col0       | col1       |
+------------+------------+
| A B        | 1          |
| C D        | 2          |
+------------+------------+
```

   Execute the following SQL statement to call the UDTF:

```
select user_udtf(col0, col1) as (c0, c1) from my_table;
```

   The following result is returned:

```
+----+------------+
| c0 | c1         |
+----+------------+
| A  | 1          |
| B  | 1          |
| C  | 2          |
| D  | 2          |
+----+------------+
```

3.10.3.3. Examples of Java UDTFs

3.10.3.3.1. Use a Java UDTF to read resources from MaxCompute

This topic describes how to use a Java user-defined table-valued function (UDTF) to read resources from MaxCompute on MaxCompute Studio.

Prerequisites

MaxCompute Studio is installed and connected to a MaxCompute project. A MaxCompute Java module is created.

For more information about related operations, see Install MaxCompute Studio, Manage project connections, and Create a MaxCompute Java module.

For more information about MaxCompute resources, see Resource.

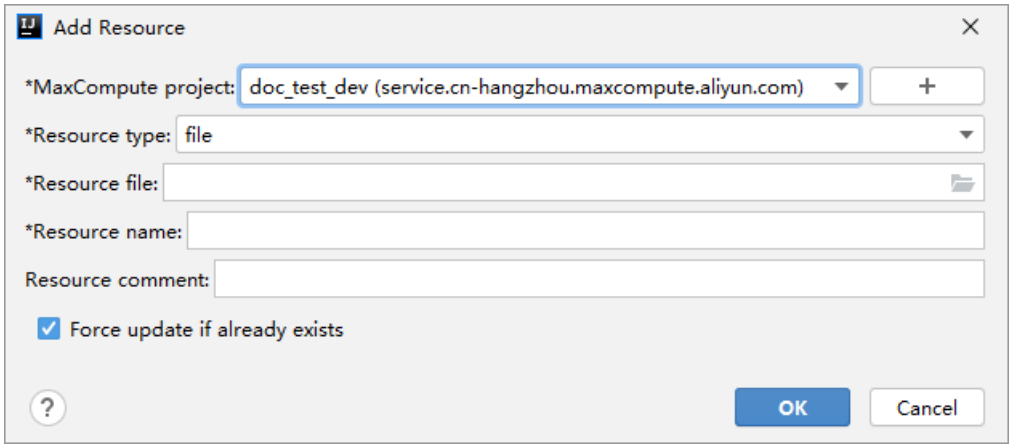
Sample code

```
package com.aliyun.odps.examples.udf;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Iterator;
import com.aliyun.odps.udf.ExecutionContext;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.annotation.Resolve;
/**
 * project: example_project
 * table: wc_in2
 * partitions: p2=1,p1=2
 * columns: colc,colb
 */
@Resolve("string,string->string,bigint,string")
public class UDTFResource extends UDTF {
  ExecutionContext ctx;
  long fileResourceLineCount;
  long tableResource1RecordCount;
  long tableResource2RecordCount;
  @Override
  public void setup(ExecutionContext ctx) throws UDFException {
  this.ctx = ctx;
  try {
   InputStream in = ctx.readResourceFileAsStream("file_resource.txt");
   BufferedReader br = new BufferedReader(new InputStreamReader(in));
   String line;
   fileResourceLineCount = 0;
   while ((line = br.readLine()) != null) {
     fileResourceLineCount++;
```

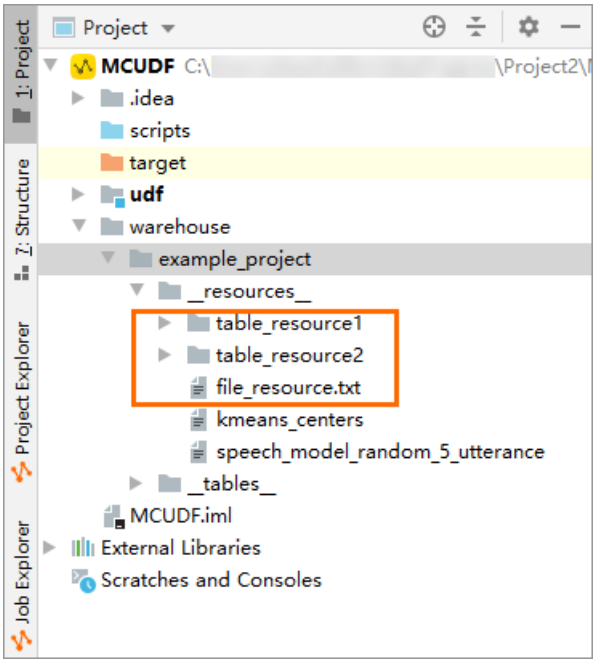
```
    }
    br.close();
    Iterator<Object[]> iterator = ctx.readResourceTable("table_resource1").iterator();
    tableResource1RecordCount = 0;
    while (iterator.hasNext()) {
      tableResource1RecordCount++;
      iterator.next();
    }
    iterator = ctx.readResourceTable("table_resource2").iterator();
    tableResource2RecordCount = 0;
    while (iterator.hasNext()) {
      tableResource2RecordCount++;
      iterator.next();
    }
  } catch (IOException e) {
    throw new UDFException(e);
  }
}
    @Override
    public void process(Object[] args) throws UDFException {
      String a = (String) args[0];
      long b = args[1] == null ? 0 : ((String) args[1]).length();
      forward(a, b, "fileResourceLineCount=" + fileResourceLineCount + "|tableResource1Recor
dCount="
      + tableResource1RecordCount + "|tableResource2RecordCount=" + tableResource2RecordCoun
t);
    }
}
```

Procedure

1. In the top navigation bar of MaxCompute Studio, choose **MaxCompute > Add Resource**. In the Add Resource dialog box, add the resource files listed in the following table.

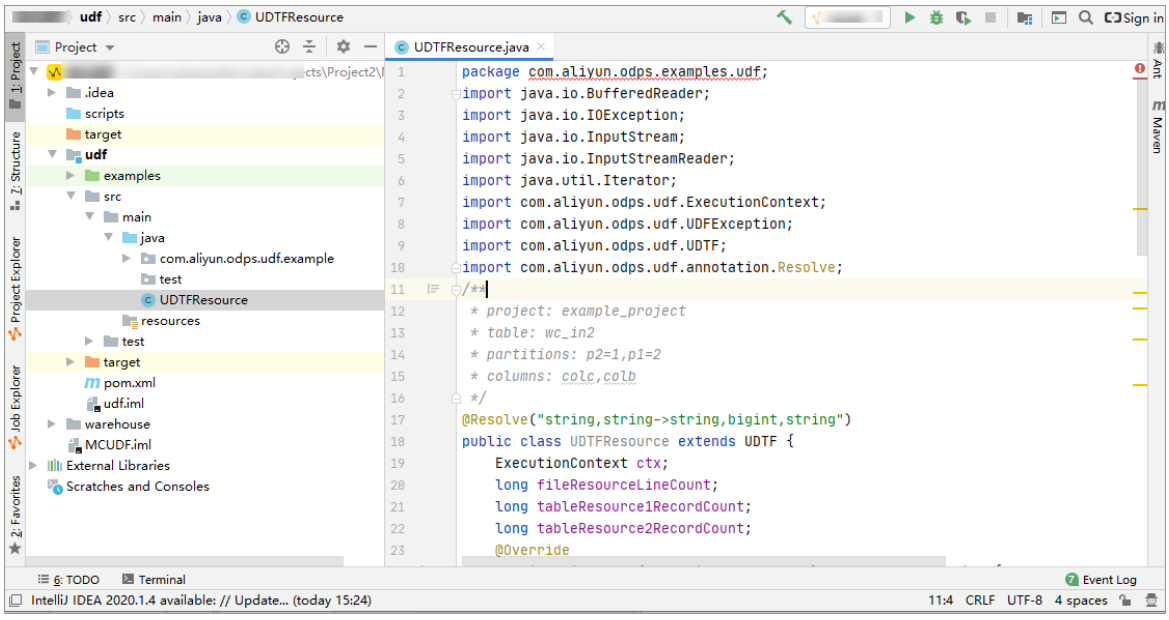


Resource file	Resource type	Example

Resource file	Resource type	Example
file_resource.txt	file	You can obtain sample resource files from the location shown in the following figure on the MaxCompute client. 
table_resource1	table	
table_resource2	table	

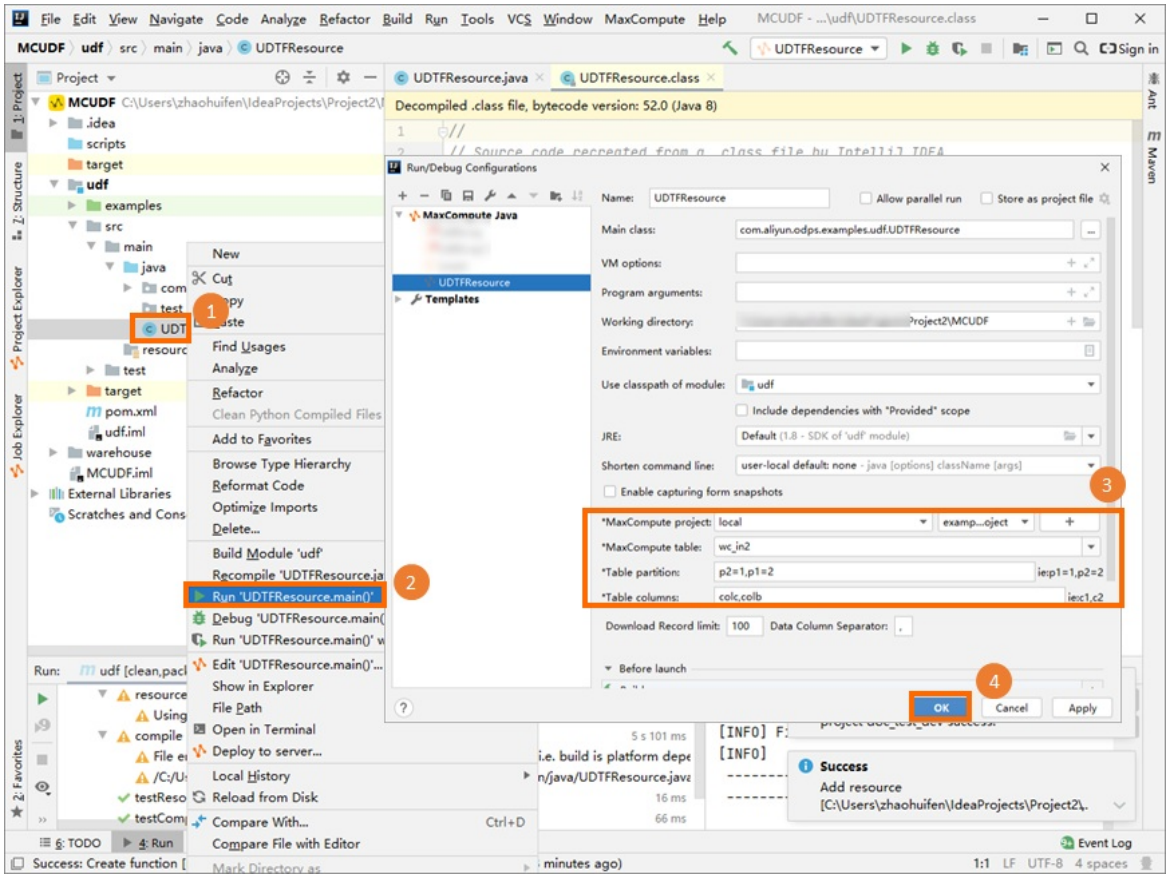
2. Create a Java UDTF program on MaxCompute Studio. For example, the name of the MaxCompute Java class is `UDTFResource` and the program code is the code in Sample code.

   For more information about how to create a UDTF, see Write a UDF.



3. Debug the UDTF on your on-premises machine and make sure that the UDTF code can normally run.

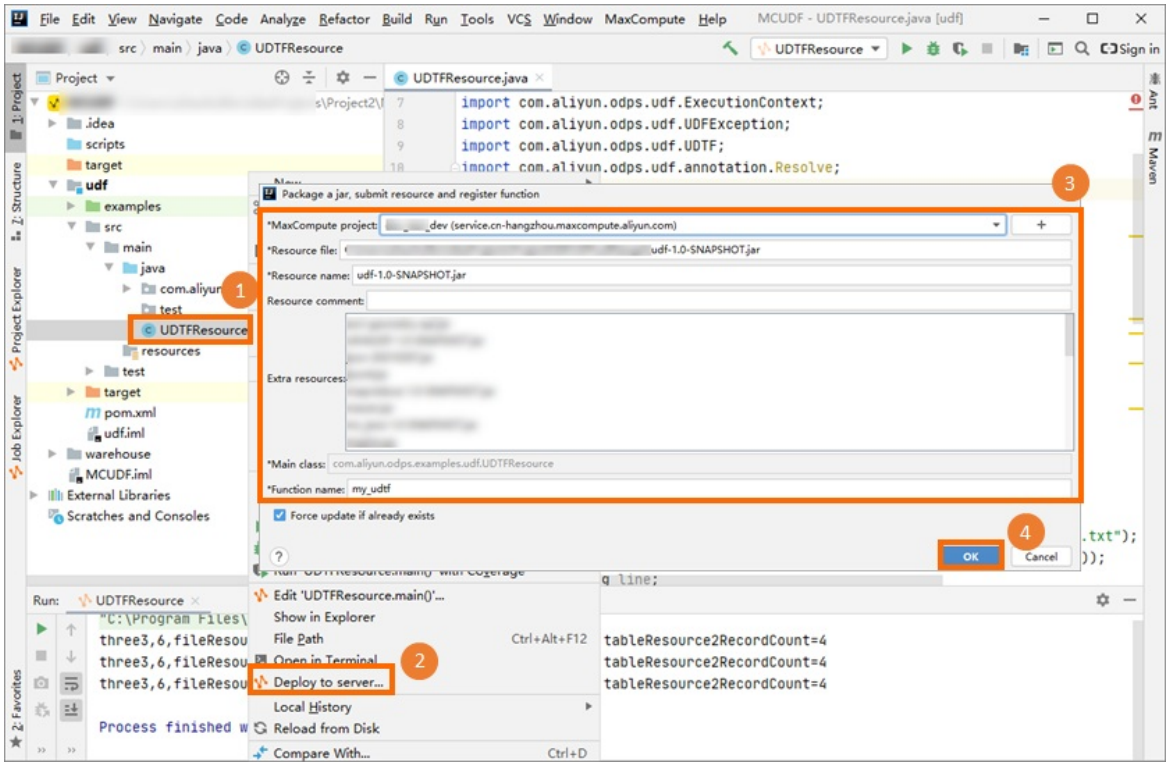
   For more information about how to debug a UDTF, see Perform a local run to debug the UDF.



Note You can configure the runtime parameters based on the example shown in the preceding figure.

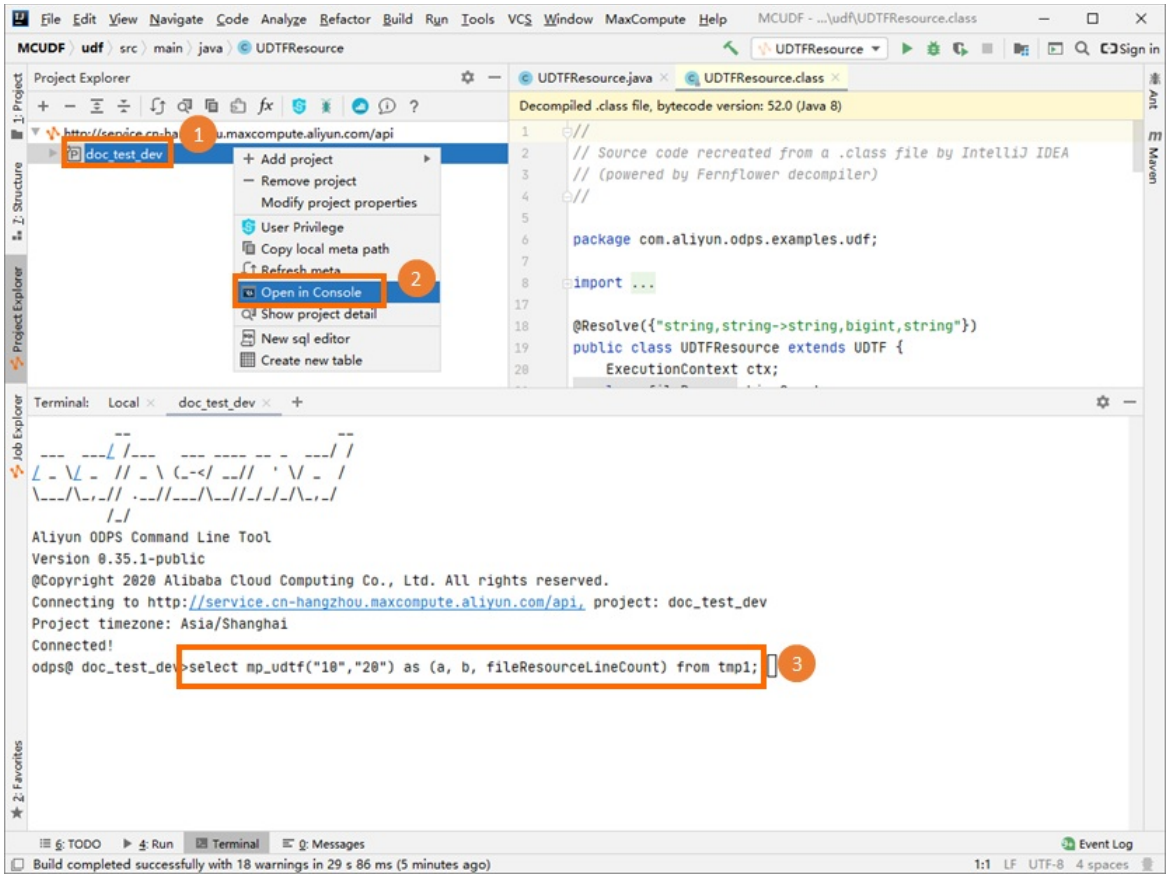
4. Package the created UDTF as a JAR file, submit the file to your MaxCompute project, and then register the UDTF, such as `my_udtf`.

   For more information about how to package a UDTF, see Procedure.



You must select the three resource files that are added in Step 1 for Extra resources.

5. In the left-side navigation pane of MaxCompute Studio, click **Project Explorer**. On the page that appears, right-click the MaxCompute project for which the UDTF is created and select Open in Console to start the MaxCompute client. Then, execute SQL statements in the code editor to call the created UDTF.



SQL sample code:

```
select my_udtf("10","20") as (a, b, fileResourceLineCount);
```

The following result is returned:

```
+-------+------------+--------+
| a | b      | fileResourceLineCount |
+-------+------------+--------+
| 10    | 2          | fileResourceLineCount=3|tableResource1RecordCount=0|tableResourc
e2RecordCount=0 |
| 10    | 2          | fileResourceLineCount=3|tableResource1RecordCount=0|tableResourc
e2RecordCount=0 |
+-------+------------+--------+
```

3.10.3.4. Python UDTF

3.10.3.4.1. Python 2 UDTF

The Python 2 version that is used by MaxCompute is Python 2.7. This topic describes how to write a user-defined table-valued function (UDTF) in Python 2.

UDTF code structure

You can use MaxCompute Studio to write UDTF code in Python 2. The UDTF code must contain the following information:

- Encoding declaration: optional.

  The declaration format is `#coding:utf-8` or `# -*- coding: utf-8 -*-`. The two formats are equivalent. If Chinese characters appear in UDTF code that is written in Python 2, an error is returned when you run the UDTF. To resolve this issue, you must add an encoding declaration to the header of the code.

- Module import: required.

  UDTF code must include `from odps.udf import annotate` and `from odps.udf import BaseUDTF`. `from odps.udf import annotate` is used to import the function signature module. This way, MaxCompute can identify the function signature that is defined in the code. `from odps.udf import BaseUDTF` is the base class for Python UDTFs. You must use this class to implement methods such as `process` and `close` in derived classes.

  If you want to reference files or tables in UDTF code, UDTF code must include `from odps.distcache import get_cache_file` or `from odps.distcache import get_cache_table`.

- Function signature: optional.

  The function signature is in the `@annotate(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return value of a UDTF. If you do not specify a function signature, input parameters of any data type can be matched when you call a UDTF in SQL statements. As a result, the data types of the return values cannot be inferred and all output parameters are of the STRING type. For more information about function signatures, see Function signatures and data types.

- Custom Python class (derived class): required.

  A custom Python class is the organizational unit of UDTF code. This class defines the variables and methods that are used to meet your business requirements. In UDTF code, you can reference third-party libraries that are built in MaxCompute or reference files or tables. For more information, see Third-party libraries or Resource reference.

- Methods to implement Python classes: required.

  Four methods can be used to implement Python classes. The following table describes these methods.

Method	Description
`BaseUDTF.init()`	The initialization method. To implement this method for a derived class, you must call the `super(BaseUDTF, self).init()` initialization method of the base class when you start to run code. The `INIT` method is called only once throughout the lifecycle of a UDTF. This method is called only before the first record is processed. If a UDTF needs to save internal states, all states can be initialized by using this method.
`BaseUDTF.process([args, ...])`	The `process` function is called once for each SQL record. The parameters of the `process` function are the input parameters of the UDTF that is specified in SQL statements.

Method	Description
`BaseUDTF.forward([args, ...])`	The output method of a UDTF. This method is called by user code. One output record is generated each time the `forward` method is called. The parameters in the `forward` method are the UDTF output parameters that are specified in SQL statements. If no function signature is specified in the Python code, all output values must be converted into the STRING type when the `forward` method is called.
`BaseUDTF.close()`	The method to terminate a UDTF. This method is called only once. It is called only before the last record is processed.

The following example shows the UDTF code.

```
#coding:utf-8
# Import the function signature module and the base class.
from odps.udf import annotate
from odps.udf import BaseUDTF
# The function signature.
@annotate('string -> string')
# The custom Python class.
class Explode(BaseUDTF):
# Methods used to implement Python classes.
   def process(self, arg):
       props = arg.split(',')
       for p in props:
           self.forward(p)
```

Limits

MaxCompute allows you to write Python 2 UDTFs in Python 2.7 and run the UDTF code in a sandbox environment. In this environment, the following operations are prohibited:

- Read data from and write data to local files.
- Start subprocesses.
- Start threads.
- Enable socket communication.
- Use other systems to call Python 2 UDFs.

Due to these limits, the code that you upload must be written by using Python standard libraries. If modules or C extension modules in Python standard libraries are involved in the preceding operations, these modules cannot be used. Take note of the following points about modules in Python standard libraries:

- All the modules that are implemented based on Python standard libraries and do not depend on extension modules are available.
- The following C extension modules are available:
  - array and audioop
  - binascii and bisect

- cmath, _codecs_cn, _codecs_hk, _codecs_iso2022, _codecs_jp, _codecs_kr, _codecs_tw, _collections, and cStringIO
- datetime
- _functools and future_builtins
- _heapq and _hashlib
- itertools
- _json
- _locale and _lsprof
- math, _md5, and _multibytecodec
- operator
- _random
- _sha256, _sha512, _sha, _struct, and strop
- time
- unicodedata
- _weakref
- cPickle

- When you run UDF code in a sandbox environment, the maximum size of data that can be written to the standard output (sys.stdout) or standard error output (`sys.stderr`) is 20 KB. If the size exceeds 20 KB, extra characters are ignored.

Third-party libraries

Third-party libraries, such as NumPy, are installed in the Python 2 environment of MaxCompute as supplements to standard libraries.

> **Note** The use of third-party libraries is subject to some limits. For example, when you use a third-party library, you are not allowed to access local data and you can use only limited network I/O resources. The related APIs in the third-party libraries are disabled.

Function signatures and data types

Format of function signature:

```
@annotate(<signature>)
```

`signature` is a function signature string. This parameter is used to identify the data types of the input parameters and return values. When a UDTF is run, the input parameters and return values of the UDTF must be of the same data type as those specified in the function signature. The system checks whether the UDTF complies with the definition of the function signature during semantics parsing. If the data types of the UDTF are inconsistent with the data types specified in the function signature, an error is returned. The signature is in the following format:

```
'arg_type_list -> type_list'
```

Parameter description:

- `type_list` : indicates the data types of return values. A UDTF can return multiple columns. The

following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

- `arg_type_list`: indicates the data types of input parameters. If multiple input parameters are used, specify multiple data types and separate them with commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, VARCHAR, complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

  `arg_type_list` can also be set to an asterisk (*) or left empty.

  - If `arg_type_list` is set to an asterisk (*), a random number of input parameters are used.
  - If `arg_type_list` is left empty, no input parameters are used.

> **Note** When you write UDTF code, you can select a data type based on the data type edition of your MaxCompute project. For more information about data type editions and the data types supported by each edition, see Data type editions.

The following table provides examples of valid function signatures.

Function signature	Description
`@annotate('bigint,boolean->string,datetime')`	The data types of the input parameters are BIGINT and BOOLEAN. The data types of the return values are STRING and DATETIME.
`@annotate('*->string, datetime')`	A random number of input parameters are used and the data types of the return values are STRING and DATETIME.
`@annotate('->double, bigint, string')`	No input parameters are used, and the data types of the return values are DOUBLE, BIGINT, and STRING.
`@annotate("array<string>,struct<a1:bigint,b1:string>,string->map<string,bigint>,struct<b1:bigint>")`	The data types of the input parameters are ARRAY, STRUCT, and MAP. The data types of the return values are MAP and STRUCT.

The following table describes the mappings between the data types that are supported in MaxCompute SQL and the Python 2 data types. You must write Python UDTFs based on the mappings to ensure data type consistency.

MaxCompute SQL data type	Python 2 data type
BIGINT	INT
STRING	STR
DOUBLE	FLOAT
BOOLEAN	BOOL
DATETIME	INT

MaxCompute SQL data type	Python 2 data type
FLOAT	FLOAT
CHAR	STR
VARCHAR	STR
BINARY	BYTEARRAY
DATE	INT
DECIMAL	DECIMAL.DECIMAL
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

> **Note**
>
> - The DATETIME type supported in MaxCompute SQL is mapped to the Python data type INT. A value of the INT type follows the UNIX format, which is the number of milliseconds that have elapsed since 00:00:00 Thursday, January 1, 1970. You can process data of the DATETIME type by using the DATETIME module in Python standard libraries.
> - The `silent` parameter is added to `odps.udf.int(value)`. If the `silent` parameter is set to True and the data type of `value` cannot be converted into the INT type, None is returned, and no error is returned.
> - NULL in MaxCompute SQL is mapped to None in Python 2.

Resource reference

You can reference files and tables in Python UDTFs by using the `odps.distcache` module.

- `odps.distcache.get_cache_file(resource_name)`: returns the content of a specific file.
  - `resource_name` is a string that specifies the name of an existing file in your MaxCompute project. If the file name is invalid or the file does not exist, an error is returned.

    > **Note** To reference a file in a UDTF, you must declare the file resource when you create the UDTF. Otherwise, an error is returned when the UDTF is called.

  - The return value is a file-like object. If this object is no longer used, you must call the `close` method to release the open file.
- `odps.distcache.get_cache_table(resource_name)`: returns the content of a specified table.
  - `resource_name` is a string that specifies the name of an existing table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned.
  - The return value is of the generator type. The caller traverses the table to obtain the content. Each time the caller traverses the table, a record of the ARRAY type is generated.

The following sample code shows how to reference files and tables.

```
# -*- coding: utf-8 -*-
from odps.udf import annotate
from odps.udf import BaseUDTF
from odps.distcache import get_cache_file
from odps.distcache import get_cache_table
@annotate('string -> string, bigint')
class UDTFExample(BaseUDTF):
    """Read pageid and adid in the file and table to generate dict.
    """
    def __init__(self):
        import json
        cache_file = get_cache_file('test_json.txt')
        self.my_dict = json.load(cache_file)
        cache_file.close()
        records = list(get_cache_table('table_resource1'))
        for record in records:
            self.my_dict[record[0]] = [record[1]]
    """Enter pageid and generate pageid and all adid values.
    """
    def process(self, pageid):
        for adid in self.my_dict[pageid]:
            self.forward(pageid, adid)
```

Usage notes

After you develop a Python 2 UDTF by following the instructions in Development process, you can use MaxCompute SQL to call the Python 2 UDTF. The following steps describe how to call a Python 2 UDTF:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute Studio to develop and call a Python 2 UDTF, see Develop a Python UDF.

3.10.3.4.2. Python 3 UDTFs

Python Software Foundation announced the End of Life (EOL) for Python 2. Due to this reason, MaxCompute supports Python 3 and uses CPython 3.7.3. This topic describes how to write user-defined table-valued function (UDTF) code in Python 3.

UDTF code structure

You can use MaxCompute Studio to write UDTF code in Python 3. The code can contain the following information:

- Module import: required.

UDTF code must include `from odps.udf import annotate` and `from odps.udf import BaseUDTF`. `from odps.udf import annotate` is used to import the function signature module. This way, MaxCompute can identify the function signature that is defined in the code. `from odps.udf import BaseUDTF` is the base class for Python UDTFs. You must use this class to implement methods such as `process` and `close` in derived classes.

If you want to reference files or tables in UDTF code, UDTF code must include `from odps.distcache import get_cache_file` or `from odps.distcache import get_cache_table`.

- Function signature: optional.

  The function signature is in the `@annotate(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return values of the UDTF. If you do not specify a function signature, input parameters of any data type can be matched when you call a UDTF in SQL statements. As a result, the data types of the return values cannot be inferred and all output parameters are of the STRING type. For more information about function signatures, see Function signatures and data types.

- Custom Python class (derived class): required.

  A custom Python class is the organizational unit of UDTF code. This class defines the variables and methods that are used to meet your business requirements. In UDTF code, you can reference third-party libraries that are built in MaxCompute or reference files or tables. For more information, see Third-party libraries or Reference resources.

- Methods to implement Python classes: required.

  Four methods can be used to implement Python classes. The following table describes these methods.

Method	Description
`BaseUDTF.init()`	The initialization method. To implement this method for a derived class, you must call the `super(BaseUDTF, self).init()` initialization method of the base class when you start to run code. The `INIT` method is called only once throughout the lifecycle of a UDTF. This method is called only before the first record is processed. If a UDTF needs to save internal states, all states can be initialized by using this method.
`BaseUDTF.process([args, ...])`	The `process` function is called once for each SQL record. The parameters of the `process` function are the input parameters of the UDTF that is specified in SQL statements.
`BaseUDTF.forward([args, ...])`	The output method of a UDTF. This method is called by user code. One output record is generated each time the `forward` method is called. The parameters in the `forward` method are the UDTF output parameters that are specified in SQL statements. If no function signature is specified in the Python code, all output values must be converted into the STRING type when the `forward` method is called.
`BaseUDTF.close()`	The method to terminate a UDTF. This method is called only once. It is called only before the last record is processed.

The following example shows the UDTF code.

```
# Import the function signature module and the base class.
from odps.udf import annotate
from odps.udf import BaseUDTF
# The function signature.
@annotate('string -> string')
# The custom Python class.
class Explode(BaseUDTF):
# Methods used to implement the custom Python class.
   def process(self, arg):
       props = arg.split(',')
       for p in props:
           self.forward(p)
```

> **Note** The underlying Python versions of Python 2 UDTFs and Python 3 UDTFs are different. You must write a UDTF based on the capabilities of the Python version that you use.

Limits

Python 3 is incompatible with Python 2. Due to this reason, you cannot use Python 2 code and Python 3 code in a single SQL statement at the same time.

Port Python 2 UDTFs

Python Software Foundation announced the EOL for Python 2. Therefore, we recommend that you port Python 2 UDTFs. The method used to port Python 2 UDTFs varies based on the types of MaxCompute projects.

- New project: If your project is a new MaxCompute project or you use Python to write a UDTF in the project for the first time, we recommend that you use Python 3 to write all Python UDTFs.
- Existing project: If your project is an existing project in which a large number of Python 2 UDTFs are created, exercise caution when you enable Python 3. If you plan to gradually replace Python 2 UDTFs with Python 3 UDTFs, use one of the following methods:
  - Use Python 3 to write new UDTFs and enable Python 3 for new jobs at the session level. For more information about how to enable Python 3, see Enable Python 3.
  - Rewrite Python 2 UDTFs to make them compatible with both Python 2 and Python 3. For more information about how to rewrite UDTFs, see Porting Python 2 Code to Python 3.

    > **Note** If you need to write a public UDTF and grant multiple MaxCompute projects the permissions on the UDTF, we recommend that you make sure that the UDTF is compatible with both Python 2 and Python 3.

Enable Python 3

By default, Python 2 is used to write UDFs in a MaxCompute project. If you want to write UDFs in Python 3, add the following command before the SQL statement that you want to execute. Then, commit and execute the statement.

```
set odps.sql.python.version=cp37;
```

Third-party libraries

NumPy is not installed in the Python 3 runtime environment in MaxCompute. To use a NumPy UDTF, you must manually upload a NumPy wheel package. If you obtain a NumPy wheel package from Python Package Index (PyPI) or obtain this package from an image, the file name is *numpy-<Version>-cp37-cp37m-manylinux1_x86_64.whl*. For more information about how to upload a file, see Resource operations or Reference third-party packages in Python UDFs.

Function signatures and data types

Format of function signatures:

```
@annotate(<signature>)
```

`signature` is a function signature string. This parameter is used to identify the data types of the input parameters and return values. When a UDTF is run, the input parameters and return values of the UDTF must be of the same data type as those specified in the function signature. The system checks whether the UDTF complies with the definition of the function signature during semantics parsing. If the data types of the UDTF are inconsistent with the data types specified in the function signature, an error is returned. The signature is in the following format:

```
'arg_type_list -> type_list'
```

Parameter description:

- `type_list`: indicates the data types of return values. A UDTF can return multiple columns. The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.
- `arg_type_list`: indicates the data types of input parameters. If multiple input parameters are used, specify multiple data types and separate them with commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, VARCHAR, complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

  `arg_type_list` can also be set to an asterisk (*) or left empty.

  - If `arg_type_list` is set to an asterisk (*), a random number of input parameters are used.
  - If `arg_type_list` is left empty, no input parameters are used.

> **Note** When you write UDTF code, you can select a data type based on the data type edition of your MaxCompute project. For more information about data type editions and the data types supported by each edition, see Data type editions.

The following table provides examples of valid function signatures.

Function signature	Description
`@annotate('bigint,boolean->string,datetime')`	The data types of the input parameters are BIGINT and BOOLEAN. The data types of the return values are STRING and DATETIME.

Function signature	Description
`@annotate('*->string, datetime')`	A random number of input parameters are used and the data types of the return values are STRING and DATETIME.
`@annotate('->double, bigint, string')`	No input parameters are used, and the data types of the return values are DOUBLE, BIGINT, and STRING.
`@annotate("array<string>,struct<a1:bigint,b1:string>,string->map<string,bigint>,struct<b1:bigint>")`	The data types of the input parameters are ARRAY, STRUCT, and MAP. The data types of the return values are MAP and STRUCT.

The following table describes the mappings between the data types that are supported by MaxCompute projects and the Python data types. You must write Python UDTFs based on the mappings to ensure data type consistency. The following table describes the data type mappings.

MaxCompute SQL data type	Python 3 data type
BIGINT	INT
STRING	UNICODE
DOUBLE	FLOAT
BOOLEAN	BOOL
DATETIME	DATETIME.DATETIME
FLOAT	FLOAT
CHAR	UNICODE
VARCHAR	UNICODE
BINARY	BYTES
DATE	DATETIME.DATE
DECIMAL	DECIMAL.DECIMAL
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

Reference resources

You can reference files and tables in Python UDTFs by using the `odps.distcache` module.

- `odps.distcache.get_cache_file(resource_name)`: returns the content of a specific file.

- `resource_name` is a string that specifies the name of an existing file in your MaxCompute project. If the file name is invalid or the file does not exist, an error is returned.

  > **Note** To reference a file in a UDTF, you must declare the file resource when you create the UDTF. Otherwise, an error is returned when the UDTF is called.

- The return value is a file-like object. If this object is no longer used, you must call the `close` method to release the open file.

- `odps.distcache.get_cache_table(resource_name)`: returns the content of a specified table.
  - `resource_name` is a string that specifies the name of an existing table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned.
  - The return value is of the generator type. The caller traverses the table to obtain the content. Each time the caller traverses the table, a record of the ARRAY type is generated.

The following sample code shows how to reference files and tables.

```
from odps.udf import annotate
from odps.udf import BaseUDTF
from odps.distcache import get_cache_file
from odps.distcache import get_cache_table
@annotate('string -> string, bigint')
class UDTFExample(BaseUDTF):
    """Read pageid and adid_list from the file and table to generate dict.
    """
    def __init__(self):
        import json
        cache_file = get_cache_file('test_json.txt')
        self.my_dict = json.load(cache_file)
        cache_file.close()
        records = list(get_cache_table('table_resource1'))
        for record in records:
            self.my_dict[record[0]] = record[1]
    """Enter pageid and generate pageid and all adid values.
    """
    def process(self, pageid):
        for adid in self.my_dict[pageid]:
            self.forward(pageid, adid)
```

Instructions

After you develop a Python 3 UDTF by following the instructions in Development process, you can use MaxCompute SQL to call the Python 3 UDTF. You can use one of the following methods to call the Python 3 UDTF:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute Studio to develop and call a Python 3 UDTF, see Develop a Python UDF.

3.10.3.5. Examples of Python UDTFs

3.10.3.5.1. Use a Python 2 UDTF to read resources from MaxCompute

This topic describes how to use a Python 2 user-defined table-valued function (UDTF) to read resources from MaxCompute on the MaxCompute client.

Prerequisites

The MaxCompute client is installed. For more information, see Install and configure the MaxCompute client.

Sample code

```
# -*- coding: utf-8 -*-
from odps.udf import annotate
from odps.udf import BaseUDTF
from odps.distcache import get_cache_file
from odps.distcache import get_cache_table
@annotate('string -> string, bigint')
class UDTFExample(BaseUDTF):
    """Read pageid and adid from resource files and resource tables and generate dict.
    """
    def __init__(self):
        import json
        cache_file = get_cache_file('test_json.txt')
        self.my_dict = json.load(cache_file)
        cache_file.close()
        records = list(get_cache_table('table_resource1'))
        for record in records:
            self.my_dict[record[0]] = [record[1]]
    """Enter pageid and generate pageid and all the adid.
    """
    def process(self, pageid):
        for adid in self.my_dict[pageid]:
            self.forward(pageid, adid)
```

Procedure

1. Save the sample code as a py_udtf_example.py file and place the file in the bin folder of the MaxCompute client.
2. Log on to the MaxCompute client to create a resource table named table_resource1 and an internal table named tmp1 and insert data into the tables. Prepare the resource file named test_json.txt and place the file in the bin folder of the MaxCompute client. The tmp1 table is the table on which you execute DML statements to write data in subsequent operations.

   For more information about how to log on to the MaxCompute client, see Start the MaxCompute client. The following sample code shows the operations that you can perform in this step:

- Create a resource table named table_resource1 and insert data into the table.

```
create table if not exists table_resource1 (pageid string, adid int);
insert into table table_resource1 values("contact_page2",2),("contact_page3",5);
```

- Create an internal table named tmp1 and insert data into the table.

```
create table if not exists tmp1 (pageid string);
insert into table tmp1 values ("front_page"),("contact_page1"),("contact_page3");
```

- View the content of the resource file test_json.txt.

```
{"front_page":[1, 2, 3], "contact_page1":[3, 4, 5]}
```

3. Add the py_udtf_example.py and test_json.txt files and the table_resource1 table as MaxCompute resources on the MaxCompute client.

   For more information about how to add resources, see Add resources. Sample statements:

```
add py py_udtf_example.py;
add file test_json.txt;
add table table_resource1 as table_resource1;
```

4. Create a UDTF named my_udtf on the MaxCompute client.

   For more information about how to create a UDTF, see Create a UDF. Sample statement:

```
create function my_udtf as 'py_udtf_example.UDTFExample' using 'py_udtf_example.py, tes
t_json.txt, table_resource1';
```

5. Execute SQL statements on the MaxCompute client to call the created UDTF.

   Sample statements:

   - Example 1: Use only a UDTF to execute an SQL statement.

```
select my_udtf(pageid) as (pageid, adid) from tmp1;
```

     The following result is returned:

```
+------------+------------+
| pageid     | adid       |
+------------+------------+
| front_page | 1          |
| front_page | 2          |
| front_page | 3          |
| contact_page1 | 3          |
| contact_page1 | 4          |
| contact_page1 | 5          |
| contact_page3 | 5          |
+------------+------------+
```

   - Example 2: Rewrite the statement in Example 1 and use the LATERAL VIEW clause and a UDTF to execute an SQL statement.

```
select pageid, adid from tmp1 lateral view my_udtf(pageid) adTable as udtf_pageid, ad
id;
```

     The following result is returned:

```
+------------+------------+
| pageid     | adid       |
+------------+------------+
| front_page | 1          |
| front_page | 2          |
| front_page | 3          |
| contact_page1 | 3          |
| contact_page1 | 4          |
| contact_page1 | 5          |
| contact_page3 | 5          |
+------------+------------+
```

The output result contains two columns: pageid and adid. pageid is obtained from tmp1 and adid is generated by the UDTF.

- Example 3: Use an aggregate function, the LATERAL VIEW clause, and a UDTF to execute an SQL statement.

```
-- Calculate the number of times each adid occurs.
select adid, count(1) as cnt
    from tmp1 lateral view my_udtf(pageid) adTable as udtf_pageid, adid
group by adid;
```

The following result is returned:

```
+------------+------------+
| adid       | cnt        |
+------------+------------+
| 1          | 1          |
| 2          | 1          |
| 3          | 2          |
| 4          | 1          |
| 5          | 2          |
+------------+------------+
```

3.10.3.5.2. Use a Python 3 UDTF to read resources from MaxCompute

This topic describes how to use a Python 3 user-defined table-valued function (UDTF) to read resources from MaxCompute on the MaxCompute client.

Prerequisites

The MaxCompute client is installed. For more information, see Install and configure the MaxCompute client.

Sample code

```
from odps.udf import annotate
from odps.udf import BaseUDTF
from odps.distcache import get_cache_file
from odps.distcache import get_cache_table
@annotate('string -> string, bigint')
class UDTFExample(BaseUDTF):
    """Read pageid and adid from resource files and resource tables and generate dict.
    """
    def __init__(self):
        import json
        cache_file = get_cache_file('test_json.txt')
        self.my_dict = json.load(cache_file)
        cache_file.close()
        records = list(get_cache_table('table_resource1'))
        for record in records:
            self.my_dict[record[0]] = record[1]
    """Enter pageid and generate pageid and all the adid.
    """
    def process(self, pageid):
        for adid in self.my_dict[pageid]:
            self.forward(pageid, adid)
```

Procedure

1. Save the sample code as a py_udtf_example.py file and place the file in the bin folder of the MaxCompute client.
2. Log on to the MaxCompute client to create a resource table named table_resource1 and an internal table named tmp1 and insert data into the tables. Prepare the resource file test_json.txt and place the file in the bin folder of the MaxCompute client. The tmp1 table is the table on which you execute DML statements to write data in subsequent operations.

   For more information about how to log on to the MaxCompute client, see Start the MaxCompute client. The following sample code shows the operations that you can perform in this step:

   - Create a resource table named table_resource1 and insert data into the table.

     ```
     create table if not exists table_resource1 (pageid string, adid_list array<int>);
     insert into table table_resource1 values("contact_page2",array(2,3,4)),("contact_page
     3",array(5,6,7));
     ```

     > **Note** The adid_list field in table_resource1 is of the ARRAY type. Execute the `set od ps.sql.python.version=cp37;` statement at the session level to enable Python 3 to read data of the ARRAY type.

   - Create an internal table named tmp1 and insert data into the table.

     ```
     create table if not exists tmp1 (pageid string);
     insert into table tmp1 values ("front_page"),("contact_page1"),("contact_page3");
     ```

   - View the content of the resource file test_json.txt.

     ```
     {"front_page":[1, 2, 3], "contact_page1":[3, 4, 5]}
     ```

3. Add the py_udtf_example.py and test_json.txt files and the table_resource1 table as MaxCompute resources on the MaxCompute client.

   For more information about how to add resources, see Add resources. Sample statements:

   ```
   add py py_udtf_example.py;
   add file test_json.txt;
   add table table_resource1 as table_resource1;
   ```

4. Create a UDTF named my_udtf on the MaxCompute client.

   For more information about how to create a UDTF, see Create a UDF. Sample statement:

   ```
   create function my_udtf as 'py_udtf_example.UDTFExample' using 'py_udtf_example.py, tes
   t_json.txt, table_resource1';
   ```

5. Execute SQL statements on the MaxCompute client to call the created UDTF.

   Sample statements:

   - Example 1: Use only a UDTF to execute an SQL statement.

     ```
     select my_udtf(pageid) as (pageid, adid) from tmp1;
     ```

     The following result is returned:

     ```
     +------------+------------+
     | pageid     | adid       |
     +------------+------------+
     | front_page | 1          |
     | front_page | 2          |
     | front_page | 3          |
     | contact_page1 | 3          |
     | contact_page1 | 4          |
     | contact_page1 | 5          |
     | contact_page3 | 5          |
     | contact_page3 | 6          |
     | contact_page3 | 7          |
     +------------+------------+
     ```

   - Example 2: Rewrite the statement in Example 1 and use the LATERAL VIEW clause and a UDTF to execute an SQL statement.

     ```
     select pageid, adid from tmp1 lateral view my_udtf(pageid) adTable as udtf_pageid, ad
     id;
     ```

     The following result is returned:

```
+--------+------------+
| pageid | adid       |
+--------+------------+
| front_page | 1          |
| front_page | 2          |
| front_page | 3          |
| contact_page1 | 3          |
| contact_page1 | 4          |
| contact_page1 | 5          |
| contact_page3 | 5          |
| contact_page3 | 6          |
| contact_page3 | 7          |
+--------+------------+
```

- Example 3: Use an aggregate function, the LATERAL VIEW clause, and a UDTF to execute an SQL statement.

```
select adid, count(1) as cnt
    from tmp1 lateral view my_udtf(pageid) adTable as udtf_pageid, adid
group by adid;
```

The following result is returned:

```
+------------+------------+
| adid       | cnt        |
+------------+------------+
| 1          | 1          |
| 2          | 1          |
| 3          | 2          |
| 4          | 1          |
| 5          | 2          |
| 6          | 1          |
| 7          | 1          |
+------------+------------+
```

3.10.4. UDAF

3.10.4.1. Overview

MaxCompute allows you to write user-defined aggregate functions (UDAFs) in Java or Python to extend the capabilities of MaxCompute functions and accommodate your business requirements. This topic describes the types, limits, usage notes, and development process of UDAFs. This topic also describes how to use UDAFs.

Background information

A many-to-one mapping is established between the input and output data of a UDAF. Multiple input records are aggregated to generate one output value. MaxCompute allows you to write UDAFs in Java or Python. The following table describes the two types of UDAFs.

UDAF type	Description
Java UDAF	This type of UDAF is written in Java to implement the function logic. For more information, see Java UDAFs.
Python UDAF	This type of UDAF is written in Python to implement the function logic. Python UDAFs are classified into Python 2 UDAFs and Python 3 UDAFs. • Python 2 UDAFs: Python 2.7 is used. For more information, see Python 2 UDAF. • Python 3 UDAFs: CPython 3.7.3 is used. For more information, see Python 3 UDAF.

Limits

You cannot access the Internet by using UDFs. If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.

Usage notes

Before you use UDFs, take note of the following items:

- UDFs cannot compete with built-in functions in performance. We recommend that you preferentially use built-in functions to implement your business logic.
- If you use a UDF in SQL statements, the memory usage of a computing job may exceed the default allocated memory size if a large amount of data is computed and data skew occurs. In this case, you can run the `set odps.sql.udf.joiner.jvm.memory=xxxx;` command at the session level to resolve the issue. For more information about the MaxCompute UDF FAQ, see FAQ about MaxCompute UDFs.
- If the name of a UDF is the same as that of a built-in function, the UDF is preferentially called. For example, if UDF CONCAT and built-in function CONCAT both exist in MaxCompute, the system automatically calls UDF CONCAT instead of the built-in function CONCAT. If you want to call the built-in function, you must add the symbol `::` before the built-in function, for example, `select ::concat('ab', 'c');` .

Development process

This section describes the development process of a UDAF.

- The following figure demonstrates how to write a MaxCompute UDF in Java.

No.	Required	Description	Platform	References

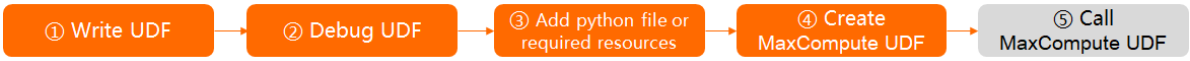
No.	Required	Description	Platform	References

1	No	Before you can use Maven to write code, you must add the related SDK dependencies to the POM file. This ensures that the code can be compiled. The following SDK dependency shows an example:
		`<dependency>`
		`<groupId>com.aliyun.odps</groupId>`
		`<artifactId>odps-sdk-udf</artifactId>`
		`<version>0.29.10-public</version>`
		`</dependency>`
		You can search for `odps-sdk-udf` from Maven repositories to obtain the version of the SDK dependency.
			IntelliJ IDEA (Maven)	None

2	Yes	Write a UDF based on your business requirements.	IntelliJ IDEA (Maven) and MaxCompute Studio	Develop a UDF in Java
3	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		
4	Yes	Debug the UDF code to ensure that the code is packaged into a JAR file after it is successfully run on your on-premises machine.		

No.	Required	Description	Platform	References
5	Yes	Upload the JAR file as a resource to your MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ▪ Add resources ▪ Create a UDF ◦ MaxCompute Studio Package a Java program, upload the package, and create a MaxCompute UDF ◦ DataWorks ▪ Create a JAR resource ▪ Register a MaxCompute function
6	Yes	Create a UDF based on the JAR file that you uploaded.		
7	No	Call the UDF in the query data code.		None

- The following figure demonstrates how to write a MaxCompute UDF in Python.



No.	Required	Description	Platform	References
1	Yes	Write a UDF based on your business requirements.	MaxCompute Studio	Develop a Python UDF
2	Yes	Debug the UDF by running it on your on-premises machine or by performing unit testing to check whether the result meets expectations.		

No.	Required	Description	Platform	References
3	Yes	Upload Python files or required resources, such as file resources, table resources, and third-party packages, to a MaxCompute project.	MaxCompute client, MaxCompute Studio, and DataWorks	◦ MaxCompute client ■ Add resources ■ Create a UDF ◦ MaxCompute Studio Upload a Python program and create a MaxCompute UDF ◦ DataWorks Create a Python resource and register a UDF
4	Yes	Create a UDF based on the uploaded Python files or required resources.		
5	No	Call the UDF in the query data code.		None

Instructions

Use the following methods to call UDFs:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

3.10.4.2. Java UDAFs

This topic describes how to write a user-defined aggregate function (UDAF) in Java.

UDAF code structure

You can use Maven in IntelliJ IDEA or MaxCompute Studio to write a UDAF in Java. The UDAF code can contain the following information:

- Java package: optional.

  You can package Java classes that are defined into a JAR file for future use.
- Base UDAF classes: required.

The required UDAF classes are `com.aliyun.odps.udf.Aggregator` and `com.aliyun.odps.udf.annotation.Resolve`. The com.aliyun.odps.udf.annotation.Resolve class corresponds to the `@Resolve` annotation. `com.aliyun.odps.udf.UDFException`: optional. This class corresponds to the methods that are used to initialize and terminate Java classes. If you want to use other UDAF classes or complex data types, add the required classes by following the instructions provided in MaxCompute SDK.

- `@Resolve` annotation: required.

  The annotation is in the `@Resolve(<signature>)` format. The `signature` is a function signature that defines the data types of input parameters and the return values of a UDAF. UDAFs cannot obtain function signatures by using the reflection feature. You can obtain a function signature only by using a `@Resolve` annotation, such as `@Resolve("smallint->varchar(10)")`. For more information about the `@Resolve` annotation, see @Resolve annotations.

- Custom Java class: required.

  A custom Java class is the organizational unit of UDAF code. This class defines the variables and methods that are used to meet your business requirements.

- Methods to implement the custom Java class: required.

  Java UDAFs must inherit the `com.aliyun.odps.udf.Aggregator` class and implement the following methods:

```
import com.aliyun.odps.udf.ContextFunction;
import com.aliyun.odps.udf.ExecutionContext;
import com.aliyun.odps.udf.UDFException;
public abstract class Aggregator implements ContextFunction {
    // The initialization method.
    @Override
    public void setup(ExecutionContext ctx) throws UDFException {
    }
    // The terminate method.
    @Override
    public void close() throws UDFException {
    }
    // Create an aggregation buffer.
    abstract public Writable newBuffer();
    // The iterate method.
    // The buffer is an aggregation buffer, which stores the data that is aggregated in a specific phase. The aggregated data refers to the dataset that is obtained after GROUP BY is performed for different Map tasks. One buffer is created for each row of data that is aggregated.
    // Writable[] indicates a row of data, which specifies the passed column in the code. For example, writable[0] indicates the first column, and writable[1] indicates the second column.
    // args specifies the parameters that are used to call a UDAF in SQL. It cannot be NULL, but the values in args can be NULL, which indicates that the input data is NULL.
    abstract public void iterate(Writable buffer, Writable[] args) throws UDFException;
    // The terminate method.
    abstract public Writable terminate(Writable buffer) throws UDFException;
    // The merge method.
    abstract public void merge(Writable buffer, Writable partial) throws UDFException;
}
```

The most important methods are `iterate`, `merge`, and `terminate`, which are used to implement the main logic of the UDAF. In addition, you must implement a user-defined writable buffer.

A user-defined writable buffer converts the objects in memory into byte sequences (or other data transmission protocols) for persistent storage in disks and network transmission. MaxCompute uses distributed computing to process UDAFs. Therefore, MaxCompute must serialize or deserialize data before the data can be transmitted between different devices.

When you write a Java UDAF, you can use Java data types or Java writable data types. For more information about the mappings between the data types that are supported by MaxCompute projects, Java data types, and Java writable data types, see Data types.

Sample code:

```
// Package the defined Java classes into a file named org.alidata.odps.udaf.examples.
package org.alidata.odps.udaf.examples;
// The base UDAF classes.
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.udf.Aggregator;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.annotation.Resolve;
// The custom Java class.
// The @Resolve annotation.
@Resolve("double->double")
public class AggrAvg extends Aggregator {
// The methods that are used to implement the custom Java class.
  private static class AvgBuffer implements Writable {
    private double sum = 0;
    private long count = 0;
    @Override
    public void write(DataOutput out) throws IOException {
      out.writeDouble(sum);
      out.writeLong(count);
    }
    @Override
    public void readFields(DataInput in) throws IOException {
      sum = in.readDouble();
      count = in.readLong();
    }
  }
  private DoubleWritable ret = new DoubleWritable();
  @Override
  public Writable newBuffer() {
    return new AvgBuffer();
  }
  @Override
  public void iterate(Writable buffer, Writable[] args) throws UDFException {
    DoubleWritable arg = (DoubleWritable) args[0];
    AvgBuffer buf = (AvgBuffer) buffer;
    if (arg != null) {
```

```
      buf.count += 1;
      buf.sum += arg.get();
    }
  }
  @Override
  public Writable terminate(Writable buffer) throws UDFException {
    AvgBuffer buf = (AvgBuffer) buffer;
    if (buf.count == 0) {
      ret.set(0);
    } else {
      ret.set(buf.sum / buf.count);
    }
    return ret;
  }
  @Override
  public void merge(Writable buffer, Writable partial) throws UDFException {
    AvgBuffer buf = (AvgBuffer) buffer;
    AvgBuffer p = (AvgBuffer) partial;
    buf.sum += p.sum;
    buf.count += p.count;
  }
}
```

Limits

You cannot access the Internet by using UDFs. If you want to access the Internet by using UDFs, fill in the network connection application form based on your business requirements and submit the application. After the application is approved, the MaxCompute technical support team will contact you and help you establish network connections. For more information about how to fill in the network connection application form, see Network connection process.

Usage notes

Before you write a Java UDAF, take note of the following points:

- We recommend that you do not package classes that have the same name but different logic into the JAR files of different UDAFs. For example, the JAR file of UDAF 1 is named udaf1.jar and the JAR file of UDAF 2 is named udaf2.jar. Both packages contain a class named `com.aliyun.UserFunction.class`, but the class has different logic in the packages. If UDAF 1 and UDAF 2 are called in the same SQL statement, MaxCompute loads the com.aliyun.UserFunction.class from one of the two packages. As result, the UDAFs cannot run as expected and a compilation error may occur.
- The data type of an input parameter or a return value in a Java UDAF is an object. The first letter of the data types that you specify in the Java UDAF code must be in uppercase, such as String.
- NULL values in MaxCompute SQL are represented by NULL in Java. Primitive data types in Java cannot represent NULL values in MaxCompute SQL. Therefore, these data types cannot be used.

@Resolve annotations

`@Resolve` annotation format:

```
@Resolve(<signature>)
```

The `signature` parameter is a string that specifies the data types of input parameters and return values. When you run a UDAF, the data types of input parameters and return values of the UDAF must be consistent with the data types specified in the function signature. The data type consistency is checked during semantic parsing. If the data types are inconsistent, an error is returned. Format of a signature:

```
'arg_type_list -> type'
```

- `arg_type_list`: indicates the data types of input parameters. If multiple input parameters are used, specify multiple data types and separate them with commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, VARCHAR, complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

  `arg_type_list` can also be set to an asterisk (*) or left empty.

  - If `arg_type_list` is set to an asterisk (*), a random number of input parameters are used.
  - If `arg_type_list` is left empty, no input parameters are used.
- `type`: specifies the data type of return values. For a UDAF, only one column of values is returned. The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, and DECIMAL(precision, scale). Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

> **Note** When you write UDAF code, you can select a data type based on the data type edition used by your MaxCompute project. For more information about data type editions and data types supported by each edition, see Data type editions.

The following table provides examples of `@Resolve` annotations.

@Resolve annotation	Description
`@Resolve('bigint,double->string')`	The data types of input parameters are BIGINT and DOUBLE, and the data type of return values is STRING.
`@Resolve('*->string')`	A random number of input parameters are used and the data type of return values is STRING.
`@Resolve('->double')`	No input parameters are used and the data type of return values is DOUBLE.
`@Resolve('array<bigint>->struct<x:string, y:int>')`	The data type of input parameters is ARRAY<BIGINT> and the data type of return values is STRUCT<x:STRING, y:INT>.

Data types

In MaxCompute, different data type editions support different data types. In MaxCompute V2.0 and later, more data types and complex data types, such as ARRAY, MAP, and STRUCT, are supported. For more information about MaxCompute data type editions, see Data type editions.

The following table describes the mappings among the data types that are supported by MaxCompute projects, Java data types, and Java writable data types. You must write Java UDAFs based on the mappings to ensure data type consistency.

MaxCompute data type	Java data type	Java writable data type
TINYINT	java.lang.Byte	ByteWritable
SMALLINT	java.lang.Short	ShortWritable
INT	java.lang.Integer	IntWritable
BIGINT	java.lang.Long	LongWritable
FLOAT	java.lang.Float	FloatWritable
DOUBLE	java.lang.Double	DoubleWritable
DECIMAL	java.math.BigDecimal	BigDecimalWritable
BOOLEAN	java.lang.Boolean	BooleanWritable
STRING	java.lang.String	Text
VARCHAR	com.aliyun.odps.data.Varchar	VarcharWritable
BINARY	com.aliyun.odps.data.Binary	BytesWritable
DATETIME	java.util.Date	DatetimeWritable
TIMESTAMP	java.sql.Timestamp	TimestampWritable
INTERVAL_YEAR_MONTH	N/A	IntervalYearMonthWritable
INTERVAL_DAY_TIME	N/A	IntervalDayTimeWritable
ARRAY	java.util.List	N/A
MAP	java.util.Map	N/A
STRUCT	com.aliyun.odps.data.Struct	N/A

Note The input parameters or return values of a UDAF can be of Java writable data types only if your MaxCompute project uses the MaxCompute V2.0 data type edition.

Instructions

After you develop a Java UDAF, you can use MaxCompute SQL to call this UDAF. For more information about how to develop a Java UDAF, see Development process. You can use one of the following methods to call the Java UDAF:

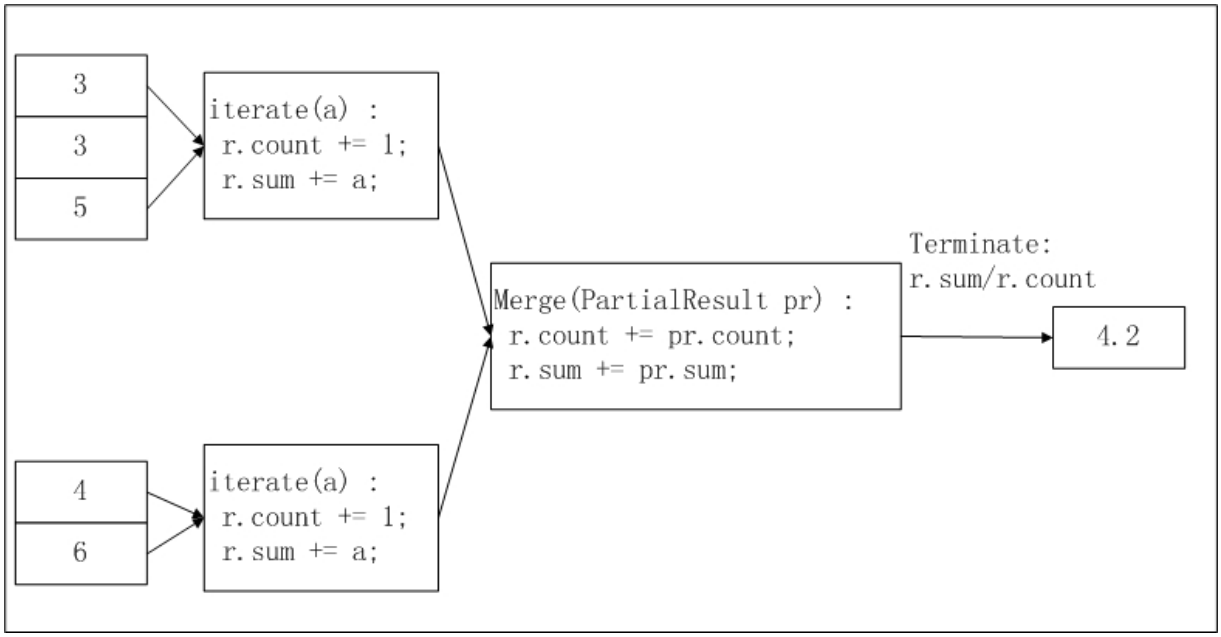
- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an

example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute Studio to develop and call a Java UDAF, see Example.

Example

This example describes how to develop a UDAF named `AggrAvg` by using MaxCompute Studio. The AggrAvg UDAF is used to calculate average values. The following figure shows the logic of the AggrAvg UDAF.



1. Slice the input data. MaxCompute slices the input data into the specified size based on the MapReduce processing workflow. The size of each slice is suitable for a worker to finish the calculation within a specified period of time.

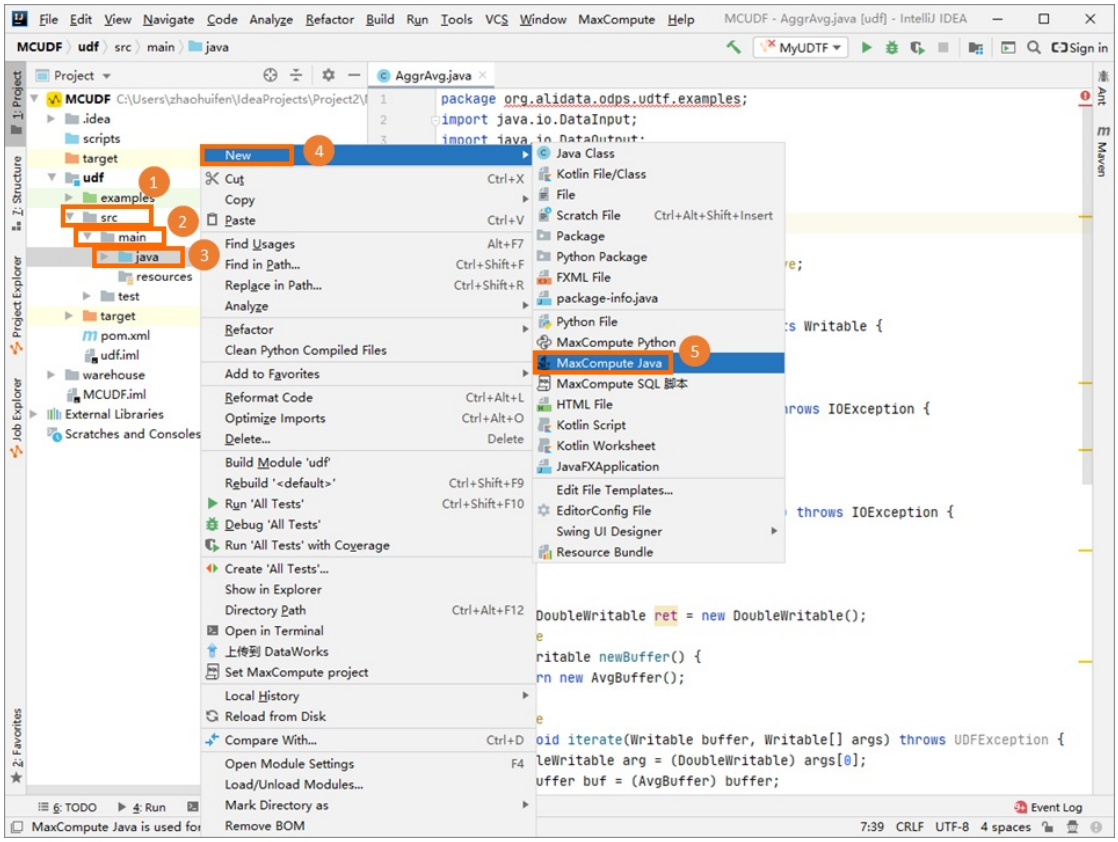
   You can configure the `odps.stage.mapper.split.size` parameter to adjust the size of the slices. For more information about the logic of data slicing, see Process.

2. Each worker counts the number of data records and total data volume in a slice. You can use the number of data records and total data volume in each slice as an intermediate result.
3. Each worker collects the information of each slice generated in Step 2.
4. In the final output, `r.sum/r.count` is the average value of all input data.

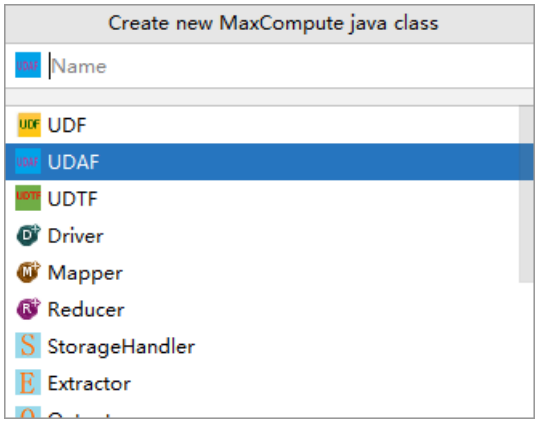
To use MaxCompute Studio to develop and call a Java UDAF, perform the following steps:

1. Make the following preparations on IntelliJ IDEA:
   i. Install MaxCompute Studio.
   ii. Establish a connection to a MaxCompute project.
   iii. Create a MaxCompute Java module.
2. Write UDAF code.

i. In the left-side navigation pane of the **Project** tab, choose **src > main > java**, right-click java, and then choose **New > MaxCompute Java**.

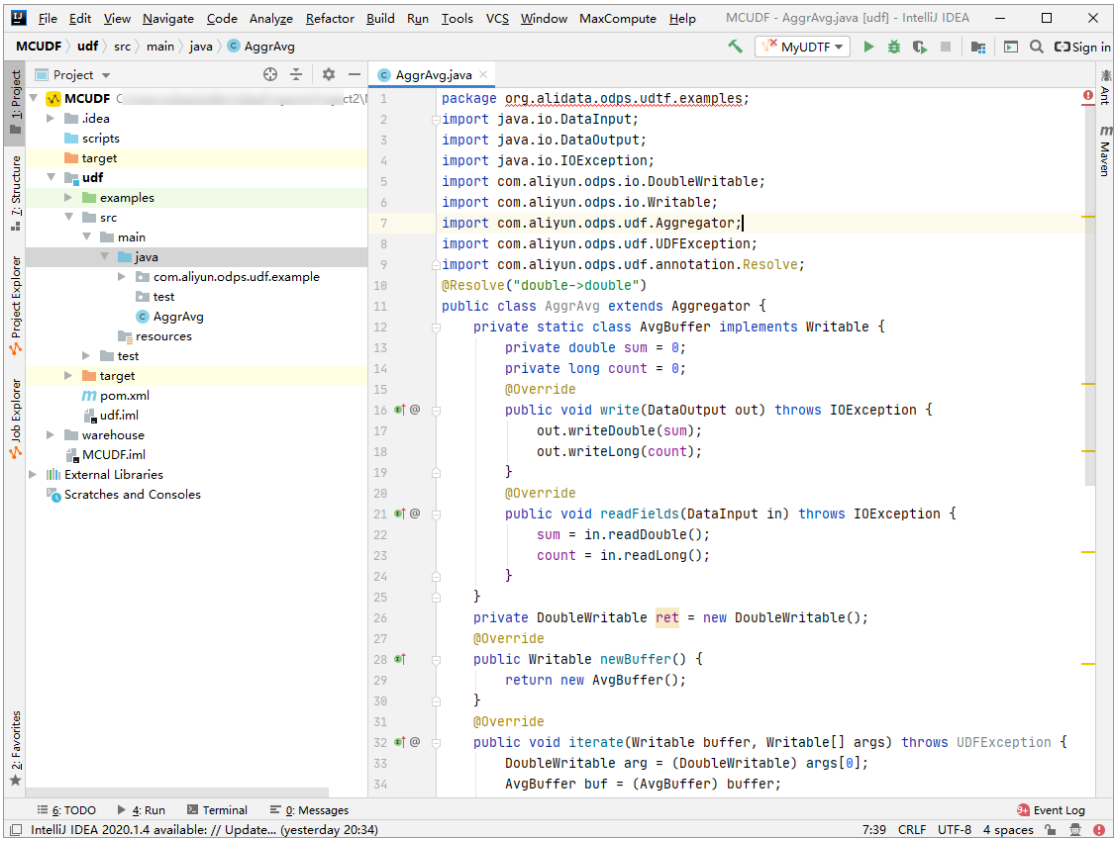


ii. In the **Create new MaxCompute java class** dialog box, click **UDAF**, enter a class name in the **Name** field, and then press Enter. In this example, the Java class is named AggrAvg.



Name: the name of the MaxCompute Java class. If no package is created, configure this parameter in the Package name.Class name format. The system automatically creates a package.

iii. Write code in the code editor.



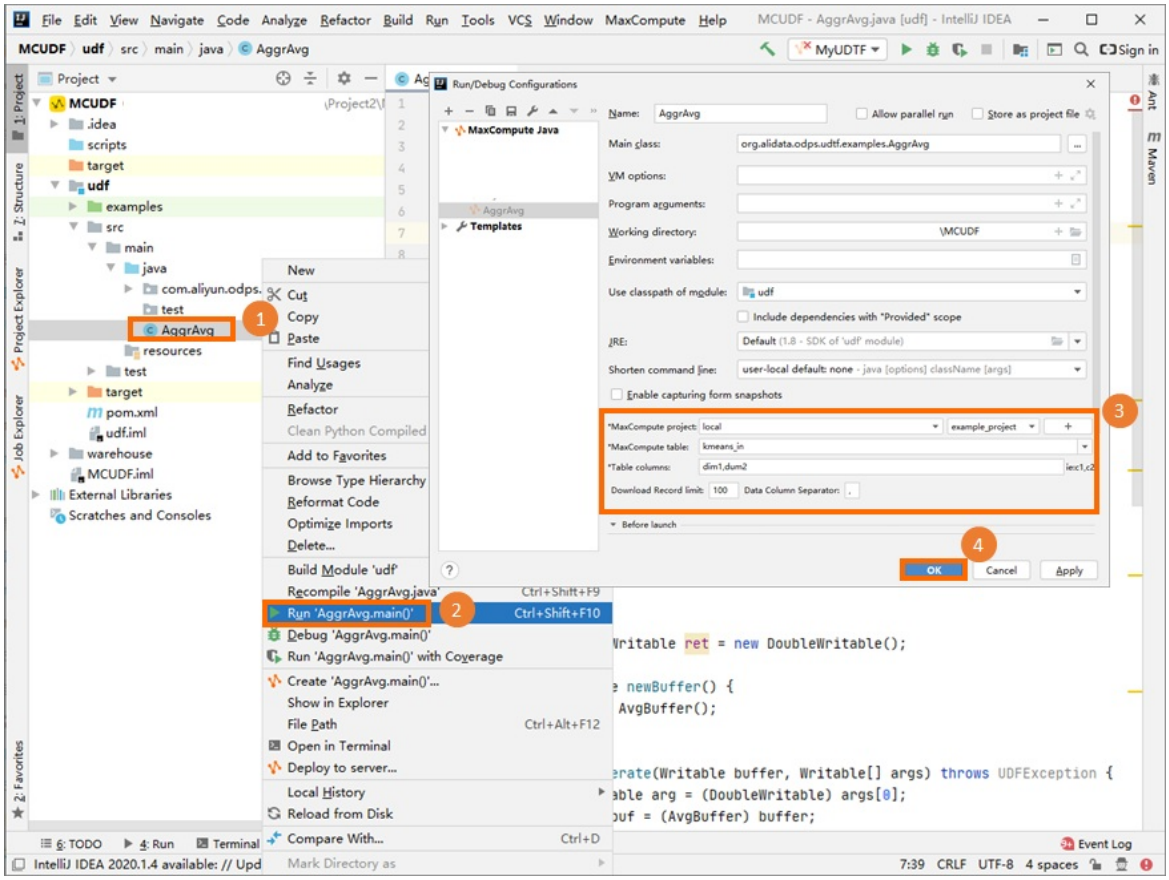
Sample UDAF code:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.udf.Aggregator;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.annotation.Resolve;
@Resolve("double->double")
public class AggrAvg extends Aggregator {
  private static class AvgBuffer implements Writable {
    private double sum = 0;
    private long count = 0;
    @Override
    public void write(DataOutput out) throws IOException {
      out.writeDouble(sum);
      out.writeLong(count);
    }
    @Override
    public void readFields(DataInput in) throws IOException {
      sum = in.readDouble();
      count = in.readLong();
    }
  }
  private DoubleWritable ret = new DoubleWritable();
  @Override
  public Writable newBuffer() {
```

```
    return new AvgBuffer();
  }
  @Override
  public void iterate(Writable buffer, Writable[] args) throws UDFException {
    DoubleWritable arg = (DoubleWritable) args[0];
    AvgBuffer buf = (AvgBuffer) buffer;
    if (arg != null) {
      buf.count += 1;
      buf.sum += arg.get();
    }
  }
  @Override
  public Writable terminate(Writable buffer) throws UDFException {
    AvgBuffer buf = (AvgBuffer) buffer;
    if (buf.count == 0) {
      ret.set(0);
    } else {
      ret.set(buf.sum / buf.count);
    }
    return ret;
  }
  @Override
  public void merge(Writable buffer, Writable partial) throws UDFException {
    AvgBuffer buf = (AvgBuffer) buffer;
    AvgBuffer p = (AvgBuffer) partial;
    buf.sum += p.sum;
    buf.count += p.count;
  }
}
```

3. Debug the UDAF on your on-premises machine and verify that the code runs as expected.

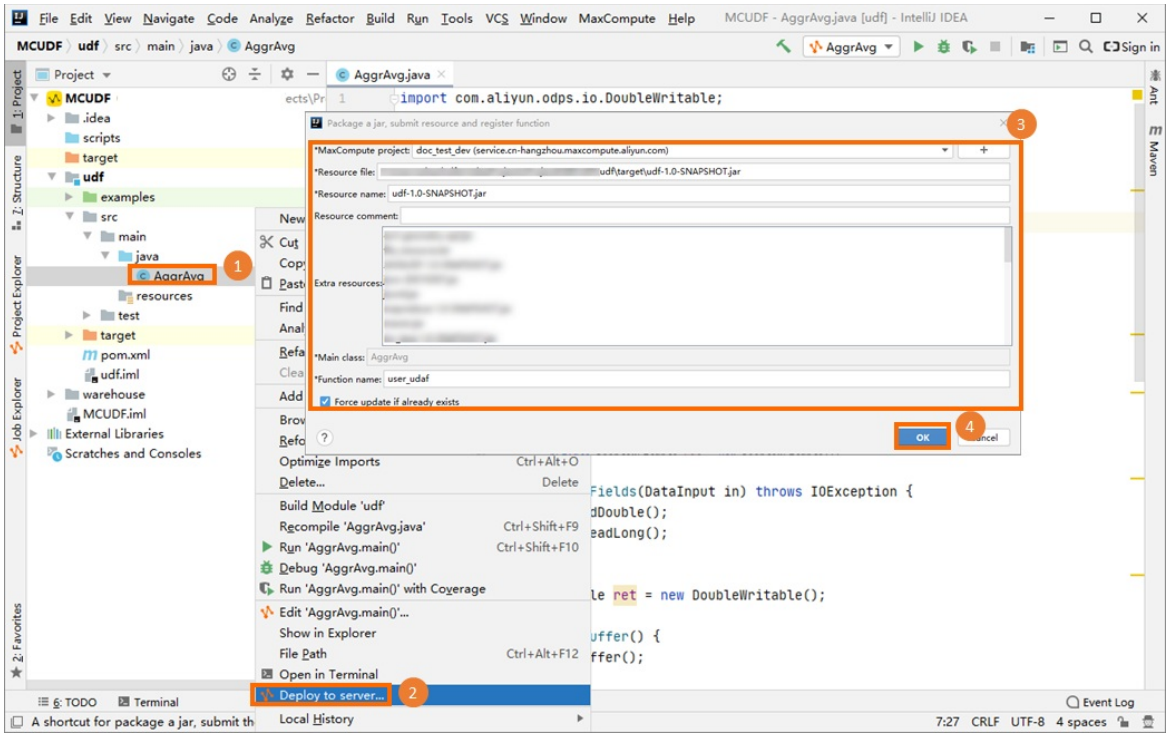
   For more information about how to debug a UDAF, see Perform a local run to debug the UDF.



> **Note** The parameter settings in the preceding figure are for reference.

4. Package the UDAF code into a JAR file, upload the file to your MaxCompute project, and then create the UDAF. In this example, a UDAF named `user_udaf` is created.

   For more information about how to package a UDAF, see Procedure.



5. In the left-side navigation pane of MaxCompute Studio, click **Project Explorer**. Right-click your MaxCompute project to start the MaxCompute client, and execute an SQL statement to call the created UDAF.

   The following example shows the data structure of the my_table table that you want to query.

```
+------------+------------+
| col0       | col1       |
+------------+------------+
| 1.2        | 2.0        |
| 1.6        | 2.1        |
+------------+------------+
```

   Execute the following statement to call the UDAF:

```
select user_udaf(col0) as c0 from my_table;
```

   The following result is returned:

```
+----+
| c0 |
+----+
| 1.4|
+----+
```

3.10.4.3. Python UDAF

3.10.4.3.1. Python 2 UDAF

The Python 2 version that is used by MaxCompute is Python 2.7. This topic describes how to write a user-defined aggregate function (UDAF) in Python 2.

UDAF code structure

You can use MaxCompute Studio to write UDAF code in Python 2. The UDAF code can contain the following information:

- Encoding declaration: optional.

  The declaration format is `#coding:utf-8` or `# -*- coding: utf-8 -*-`. The two formats are equivalent. If Chinese characters appear in UDAF code that is written in Python 2, an error is returned when you run the UDAF. To address this issue, you must add an encoding declaration to the header of the code.

- Module import: required.

  UDAF code must include at least `from odps.udf import annotate` and `from odps.udf import BaseUDAF`. `from odps.udf import annotate` is used to import the function signature module. This way, MaxCompute can identify the function signature that is defined in the code. `from odps.udf import BaseUDAF` is a base class for Python UDAFs. You must use this class to implement methods such as `iterate`, `merge`, or `terminate` in derived classes.

  If you want to reference file or table resources in UDAF code, UDAF code must include `from odps.distcache import get_cache_file` or `from odps.distcache import get_cache_table`.

- Function signature: required.

  The function signature is in the `@annotate(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return value of the UDAF. For more information about function signatures, see Function signatures and data types.

- Custom Python class (derived class): required.

  A custom Python class is the organizational unit of UDAF code. This class defines the variables and methods that are used to meet your business requirements. In UDAF code, you can also reference third-party libraries that are installed in MaxCompute or reference files or tables. For more information, see Third-party libraries or Reference resources.

- Methods to implement Python classes: required.

  The following table describes the four methods that can be used to implement Python classes. You can select a method based on your business requirements.

Method	Description
`BaseUDAF.new_buffer()`	Returns the intermediate value buffer of a UDAF. `buffer` must be a marshallable object, such as LIST or DICT, and the `buffer` size cannot increase with the amount of data. In extreme cases, the `buffer` size cannot exceed 2 MB after the marshaling operation.
`BaseUDAF.iterate(buffer[, args, ...])`	Aggregates `args` into the intermediate value `buffer`.
`BaseUDAF.merge(buffer, pbuffer)`	Stores the merged results of `pbuffer` and the intermediate value `buffer` in the `buffer`.
`BaseUDAF.terminate(buffer)`	Converts `buffer` into a value of a basic data type in MaxCompute SQL.

Sample code:

```
#coding:utf-8
# Import the function signature module and base classes.
from odps.udf import annotate
from odps.udf import BaseUDAF
# The function signature.
@annotate('double->double')
# The custom Python class.
class Average(BaseUDAF):
# Methods to implement Python classes.
    def new_buffer(self):
        return [0, 0]
    def iterate(self, buffer, number):
        if number is not None:
            buffer[0] += number
            buffer[1] += 1
    def merge(self, buffer, pbuffer):
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def terminate(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] / buffer[1]
```

Limits

MaxCompute allows you to write Python 2 UDAFs in Python 2.7 and run the UDAF code in a sandbox environment. In this environment, the following operations are prohibited:

- Read data from and write data to local files.
- Start subprocesses.
- Start threads.
- Enable socket communication.
- Use other systems to call Python 2 UDFs.

Due to these limits, the code that you upload must be written by using Python standard libraries. If modules or C extension modules in Python standard libraries are involved in the preceding operations, these modules cannot be used. Take note of the following points about modules in Python standard libraries:

- All the modules that are implemented based on Python standard libraries and do not depend on extension modules are available.
- The following C extension modules are available:
  - array and audioop
  - binascii and bisect
  - cmath, _codecs_cn, _codecs_hk, _codecs_iso2022, _codecs_jp, _codecs_kr, _codecs_tw, _collections, and cStringIO
  - datetime
  - _functools and future_builtins
  - _heapq and _hashlib

- itertools
- _json
- _locale and _lsprof
- math, _md5, and _multibytecodec
- operator
- _random
- _sha256, _sha512, _sha, _struct, and strop
- time
- unicodedata
- _weakref
- cPickle

- When you run UDF code in a sandbox environment, the maximum size of data that can be written to the standard output (sys.stdout) or standard error output (`sys.stderr`) is 20 KB. If the size exceeds 20 KB, extra characters are ignored.

Third-party libraries

Third-party libraries, such as NumPy, are installed in the Python 2 environment of MaxCompute as supplements to standard libraries.

> **Note** The use of third-party libraries is subject to some limits. For example, when you use a third-party library, you are not allowed to access local data and you can use only limited network I/O resources. The related APIs in the third-party libraries are disabled.

Function signatures and data types

Format of function signatures:

```
@annotate(<signature>)
```

The `signature` parameter is a string that specifies the data types of the input parameters and return value. When you run a UDAF, the data types of input parameters and the return value of the UDAF must be consistent with the data types specified in the function signature. Data type consistency is checked during semantic parsing. If the data types are inconsistent, an error is returned. Format of function signature:

```
'arg_type_list -> type'
```

- `arg_type_list` : indicates the data types of input parameters. If multiple input parameters are used, specify multiple data types and separate them with commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, VARCHAR, complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

  `arg_type_list` can also be set to an asterisk (*) or left empty.

  - If `arg_type_list` is set to an asterisk (*), a random number of input parameters are used.
  - If `arg_type_list` is left empty, no input parameters are used.

- `type`: specifies the data type of return values. For a UDAF, only one column of values is returned. The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, and DECIMAL(precision, scale). Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

> **Note** When you write UDAF code, you can select a data type based on the data type edition used by your MaxCompute project. For more information about data type editions and data types supported by each edition, see Data type editions.

The following table provides examples of valid function signatures.

Function signature	Description
`@annotate('bigint,double->string')`	The data types of input parameters are BIGINT and DOUBLE and the data type of the return values is STRING.
`@annotate('*->string')`	A random number of input parameters are used and the data type of the return values is STRING.
`@annotate('->double')`	No input parameters are used and the data type of the return values is DOUBLE.
`@annotate('array<bigint>->struct<x:string, y:int>')`	The data type of input parameters is ARRAY<BIGINT> and the data type of the return value is STRUCT<x:STRING, y:INT>.

The following table describes the mappings between the data types that are supported in MaxCompute SQL and the Python 2 data types. You must write Python UDAFs based on the mappings to ensure data type consistency. The following table describes the data type mappings.

MaxCompute SQL data type	Python 2 data type
BIGINT	INT
STRING	STR
DOUBLE	FLOAT
BOOLEAN	BOOL
DATETIME	INT
FLOAT	FLOAT
CHAR	STR
VARCHAR	STR
BINARY	BYTEARRAY
DATE	INT

MaxCompute SQL data type	Python 2 data type
DECIMAL	DECIMAL.DECIMAL
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

> **Note**
> - The DATETIME type supported in MaxCompute SQL is mapped to the Python data type INT. A value of the INT type follows the UNIX format, which is the number of milliseconds that have elapsed since 00:00:00 Thursday, January 1, 1970. You can process data of the DATETIME type by using the DATETIME module in Python standard libraries.
> - The `silent` parameter is added to `odps.udf.int(value)`. If the `silent` parameter is set to True and the data type of `value` cannot be converted into the INT type, None is returned, and no error is returned.
> - NULL in MaxCompute SQL is mapped to None in Python 2.

Reference resources

You can reference files and tables in Python 2 UDAF code by using the `odps.distcache` module.

- `odps.distcache.get_cache_file(resource_name)`: returns the content of a specific file.
  - `resource_name` is a string that specifies the name of an existing file in your MaxCompute project. If the file name is invalid or the file does not exist, an error is returned.

    > **Note** To reference a file in the UDAF code, you must declare the file when you create the UDAF. Otherwise, an error is returned when you call the UDAF.

  - The return value is a file-like object. If this object is no longer used, you must call the `close` method to release the file.
- `odps.distcache.get_cache_table(resource_name)`: returns the content of a specific table.
  - `resource_name` is a string that specifies the name of an existing table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned.
  - The return value is of the GENERATOR type. The caller traverses the table to obtain the table content. A record of the ARRAY type is obtained each time the caller traverses the table.

For more information, see Reference resources (Python 2 UDFs) and Reference resources (Python 2 UDTFs).

Usage notes

After you develop a Python 2 UDAF by following the instructions in Development process, you can use MaxCompute SQL to call this UDAF. The following steps describe how to call a Python 2 UDAF:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an

example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute Studio to develop and call a Python 2 UDAF, see Develop a Python UDF.

3.10.4.3.2. Python 3 UDAF

Python Software Foundation announced the End of Life (EOL) for Python 2. Due to this reason, MaxCompute supports Python 3 and uses CPython 3.7.3. This topic describes how to write a user-defined aggregate function (UDAF) in Python 3.

UDAF code structure

You can use MaxCompute Studio to write UDAF code in Python 3. The UDAF code must contain the following information:

- Module import: required.

  UDAF code must include at least `from odps.udf import annotate` and `from odps.udf import BaseUDAF`. `from odps.udf import annotate` is used to import the function signature module. This way, MaxCompute can identify the function signature that is defined in the code. `from odps.udf import BaseUDAF` is a base class for Python UDAFs. You must use this class to implement methods such as `iterate`, `merge`, or `terminate` in derived classes.

  If you want to reference file or table resources in UDAF code, UDAF code must include `from odps.distcache import get_cache_file` or `from odps.distcache import get_cache_table`.

- Function signature: required.

  The function signature is in the `@annotate(<signature>)` format. The `signature` parameter is used to define the data types of the input parameters and return value of the UDAF. For more information about function signatures, see Function signatures and data types.

- Custom Python class (derived class): required.

  A custom Python class is the organizational unit of UDAF code. This class defines the variables and methods that are used to meet your business requirements. In UDAF code, you can also reference third-party libraries that are installed in MaxCompute or reference files or tables. For more information, see Third-party libraries or Reference resources.

- Methods to implement Python classes: required.

  The following table describes the four methods that can be used to implement Python classes. You can select a method based on your business requirements.

Method	Description
`BaseUDAF.new_buffer()`	Returns the intermediate value buffer of a UDAF. `buffer` must be a marshallable object, such as LIST or DICT, and the `buffer` size cannot increase with the amount of data. In extreme cases, the `buffer` size cannot exceed 2 MB after the marshaling operation.
`BaseUDAF.iterate(buffer[, args, ...])`	Aggregates `args` into the intermediate value `buffer`.

Method	Description
`BaseUDAF.merge(buffer, pbuffer)`	Stores the merged results of `pbuffer` and the intermediate value `buffer` in the `buffer`.
`BaseUDAF.terminate(buffer)`	Converts `buffer` into a value of a basic data type in MaxCompute SQL.

Sample code:

```
# Import the function signature module and base classes.
from odps.udf import annotate
from odps.udf import BaseUDAF
# The function signature.
@annotate('double->double')
# The custom Python class.
class Average(BaseUDAF):
# Methods used to implement Python classes.
    def new_buffer(self):
        return [0, 0]
    def iterate(self, buffer, number):
        if number is not None:
            buffer[0] += number
            buffer[1] += 1
    def merge(self, buffer, pbuffer):
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def terminate(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] / buffer[1]
```

Note Python 2 UDAFs and Python 3 UDAFs differ in terms of the underlying Python version. You can write a UDAF based on the capability of the Python version that you use.

Limits

Python 3 is incompatible with Python 2. Due to this reason, you cannot use Python 2 code and Python 3 code in a single SQL statement at the same time.

Port Python 2 UDAFs

Python Software Foundation announced the EOL for Python 2. Therefore, we recommend that you port Python 2 UDAFs. The method used to port Python 2 UDAF varies based on the types of MaxCompute projects.

- New project: If your project is a new MaxCompute project or if it is the first time that you use Python to write a UDAF for a MaxCompute project, we recommend that you use Python 3 to write all Python UDAFs.
- Existing project: If your project is an existing project for which a large number of Python 2 UDAFs are created, proceed with caution when you enable Python 3. If you plan to gradually replace Python 2

UDAFs with Python 3 UDAFs, use one of the following methods:

- Use Python 3 to write new UDAFs and enable Python 3 for new jobs at the session level. For more information about how to enable Python 3, see Enable Python 3.
- Rewrite Python 2 UDAFs to make them compatible with both Python 2 and Python 3. For more information about how to rewrite Python 2 UDAFs, see Porting Python 2 Code to Python 3.

> **Note** If you want to write a public UDAF that can be shared among multiple projects, we recommend that the UDAF be compatible with both Python 2 and Python 3.

Enable Python 3

By default, Python 2 is used to write UDFs in a MaxCompute project. If you want to write UDFs in Python 3, add the following command before the SQL statement that you want to execute. Then, commit and execute the statement.

```
set odps.sql.python.version=cp37;
```

Third-party libraries

The third-party library NumPy is not installed in the Python 3 environment of MaxCompute. To use a NumPy UDAF, you must manually upload a NumPy wheel package. If you download a NumPy wheel package from Python Package Index (PyPI) or obtain this package from an image, the package is named in the following format *numpy-<Version>-cp37-cp37m-manylinux1_x86_64.whl*. For more information about how to upload a package, see Resource operations or Reference third-party packages in Python UDFs.

Function signatures and data types

Format of function signatures:

```
@annotate(<signature>)
```

The `signature` parameter is a string that specifies the data types of the input parameters and return value. When you run a UDAF, the data types of input parameters and the return value of the UDAF must be consistent with the data types specified in the function signature. Data type consistency is checked during semantic parsing. If the data types are inconsistent, an error is returned. Format of function signature:

```
'arg_type_list -> type'
```

- `arg_type_list`: indicates the data types of input parameters. If multiple input parameters are used, specify multiple data types and separate them with commas (,). The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, DECIMAL(precision,scale), CHAR, VARCHAR, complex data types (ARRAY, MAP, and STRUCT), and nested complex data types.

  `arg_type_list` can also be set to an asterisk (*) or left empty.

  - If `arg_type_list` is set to an asterisk (*), a random number of input parameters are used.
  - If `arg_type_list` is left empty, no input parameters are used.
- `type`: specifies the data type of return values. For a UDAF, only one column of values is returned.

The following data types are supported: BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, FLOAT, BINARY, DATE, and DECIMAL(precision, scale). Complex data types, such as ARRAY, MAP, and STRUCT, and nested complex data types are also supported.

> **Note** When you write UDAF code, you can select a data type based on the data type edition used by your MaxCompute project. For more information about data type editions and data types supported by each edition, see Data type editions.

The following table provides examples of valid function signatures.

Function signature	Description
`@annotate('bigint,double->string')`	The data types of input parameters are BIGINT and DOUBLE and the data type of the return values is STRING.
`@annotate('*->string')`	A random number of input parameters are used and the data type of the return values is STRING.
`@annotate('->double')`	No input parameters are used and the data type of the return values is DOUBLE.
`@annotate('array<bigint>->struct<x:string, y:int>')`	The data type of input parameters is ARRAY<BIGINT> and the data type of the return value is STRUCT<x:STRING, y:INT>.

The following table describes the mappings between the data types that are supported in MaxCompute SQL and the Python 2 data types. You must write Python UDAFs based on the mappings to ensure data type consistency. The following table describes the data type mappings.

MaxCompute SQL data type	Python 3 data type
BIGINT	INT
STRING	UNICODE
DOUBLE	FLOAT
BOOLEAN	BOOL
DATETIME	DATETIME.DATETIME
FLOAT	FLOAT
CHAR	UNICODE
VARCHAR	UNICODE
BINARY	BYTES
DATE	DATETIME.DATE
DECIMAL	DECIMAL.DECIMAL

MaxCompute SQL data type	Python 3 data type
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

Reference resources

You can reference files and tables in Python 2 UDAF code by using the `odps.distcache` module.

- `odps.distcache.get_cache_file(resource_name)`: returns the content of a specific file.
  - `resource_name` is a string that specifies the name of an existing file in your MaxCompute project. If the file name is invalid or the file does not exist, an error is returned.

    > **Note** To reference a file in the UDAF code, you must declare the file when you create the UDAF. Otherwise, an error is returned when you call the UDAF.

  - The return value is a file-like object. If this object is no longer used, you must call the `close` method to release the file.
- `odps.distcache.get_cache_table(resource_name)`: returns the content of a specific table.
  - `resource_name` is a string that specifies the name of an existing table in your MaxCompute project. If the table name is invalid or the table does not exist, an error is returned.
  - The return value is of the GENERATOR type. The caller traverses the table to obtain the table content. A record of the ARRAY type is obtained each time the caller traverses the table.

For more information, see Reference resources (Python 3 UDFs) and Reference resources (Python 3 UDTFs).

Usage notes

After you develop a Python 3 UDAF by following the instructions in Development process, you can use MaxCompute SQL to call this UDAF. The following steps describe how to call a Python 3 UDAF:

- Use a UDF in a MaxCompute project: The method is similar to that of using built-in functions.
- Use a UDF across projects: Use a UDF of Project B in Project A. The following statement shows an example: `select B:udf_in_other_project(arg0, arg1) as res from table_t;`. For more information about resource sharing across projects, see Cross-project resource access based on packages.

For more information about how to use MaxCompute Studio to develop and call a Python 3 UDAF, see Develop a Python UDF.

3.10.5. Code-embedded UDFs

This topic describes how to use code-embedded user-defined functions (UDFs) to embed Java or Python code into SQL scripts.

Background information

Code-embedded UDFs of MaxCompute resolve the following issues in code implementation and maintenance:

- Complicated code implementation: After you create UDFs and develop code, you must compile the code in Java and create resources and functions.
- Inconvenient code maintenance: You cannot directly view the implementation logic of UDFs that are referenced in SQL scripts or obtain the source code of JAR packages.
- Poor code readability: To implement Java library functions by using user-defined type (UDT), you must write Java code as expressions in long code lines. In addition, you may fail to write some Java code as expressions. Example:

```
Foo f = new Foo();
f.execute();
f.getResult();
```

Features

Code-embedded UDFs allow you to embed Java or Python code into SQL scripts. When you compile a script, Janino-compiler identifies and extracts the embedded code, compiles the code in Java, and then dynamically generates resources and creates temporary functions.

You can place SQL scripts and third-party code lines in the same source code file. This simplifies the usage of UDT or UDFs and facilitates daily development and maintenance.

Limits

You can use only Janino-compiler to compile embedded Java code, and the syntax of the embedded Java code must be a subset of the standard JDK syntax. Embedded Java code has the following limits:

- It does not support lambda expressions.
- You cannot specify multiple types of exceptions in a single catch block. For example, `catch(Exception1 | Exception2 e)` is not allowed.
- It cannot automatically infer generic arguments. For example, `Map map = new HashMap<>();` is not supported.
- Expressions for type argument inference are ignored. You must use cast expressions to specify the argument type, for example, `(String) myMap.get(key)`.
- Assertions are forcibly enabled, even if the -ea option of the Java virtual machine (JVM) is used.
- Code that is programmed in Java 8 or later versions is not supported.

Reference embedded code in UDT

You must submit and execute the script in Script Mode SQL. For more information, see Script Mode SQL. Sample code:

```
SELECT
  s,
  com.mypackage.Foo.extractNumber(s)
FROM VALUES ('abc123def'),('apple') AS t(s);
#CODE ('lang'='JAVA')
package com.mypackage;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class Foo {
  final static Pattern compile = Pattern.compile(". *?([ 0-9]+).*");
  public static String extractNumber(String input) {
    final Matcher m = compile.matcher(input);
    if (m.find()) {
      return m.group(1);
    }
    return null;
  }
}
#END CODE;
```

- `#CODE` and `#END CODE` indicate the beginning and end of the embedded code block, respectively. In this example, the embedded code block is placed at the end of the script, and applies to the whose script.
- `'lang'='JAVA'` indicates that the embedded code is in Java. JAVA can be replaced with `PYTHON`.
- You can use UDT syntax in the SQL script to call `Foo.extractNumber`.

Define and call a Java code-embedded UDF

Sample code:

```
CREATE TEMPORARY FUNCTION foo AS 'com.mypackage.Reverse' USING
#CODE ('lang'='JAVA')
package com.mypackage;
import com.aliyun.odps.udf.UDF;
public class Reverse extends UDF {
  public String evaluate(String input) {
    if (input == null) return null;
    StringBuilder ret = new StringBuilder();
    for (int i = input.toCharArray().length - 1; i >= 0; i--) {
      ret.append(input.toCharArray()[i]);
    }
    return ret.toString();
  }
}
#END CODE;
SELECT foo('abdc');
```

- You can place the embedded code block next to `USING` or at the end of the script. If you place it next to `USING`, it applies only to the `CREATE TEMPORARY FUNCTION` statement.
- The function created by `CREATE TEMPORARY FUNCTION` is a temporary function. This temporary function is executed only during the current execution process and is not stored in the MaxCompute

meta system.

Define and call a Java code-embedded UDTF

Sample code:

```
CREATE TEMPORARY FUNCTION foo AS 'com.mypackage.Reverse' USING
#CODE ('lang'='JAVA', 'filename'='embedded.jar')
package com.mypackage;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.annotation.Resolve;
@Resolve({"string->string,string"})
public class Reverse extends UDTF {
  @Override
  public void process(Object[] objects) throws UDFException {
    String str = (String) objects[0];
    String[] split = str.split(",");
    forward(split[0], split[1]);
  }
}
#END CODE;
SELECT foo('ab,dc') AS (a,b);
```

The return value of `@Resolve` must be of the `string[]` type. However, Janino-compiler cannot identify `"string->string,string"` in embedded code as `string[]`. To enable Janino-compiler to identify "string->string,string" as string[], "string->string,string" must be enclosed in braces `{}`. When you create a Java UDTF by using a common method, braces `{}` are not required.

Define and call a Python code-embedded UDF

Sample code:

```
CREATE TEMPORARY FUNCTION foo AS 'embedded.UDFTest' USING
#CODE ('lang'='PYTHON', 'filename'='embedded')
from odps.udf import annotate
@annotate("bigint->bigint")
class UDFTest(object):
  def evaluate(self, a):
    return a * a
#END CODE;
SELECT foo(4);
```

- The indentation of Python code must comply with Python language specifications.
- When you register a Python UDF, the class name that follows the `AS` clause must contain the file name of the Python source code. You can use `'filename'='embedded'` to specify a virtual file name.

3.10.6. SQL functions

SQL functions allow you to reference SQL user-defined functions (UDFs) in SQL scripts. This topic describes how to manage SQL functions.

Background

MaxCompute allows you to create UDFs by using SQL scripts in addition to the Java or Python code. SQL functions support more function type parameters and anonymous function parameters. SQL functions allow you to define business logic that suits your business requirements. SQL functions can be used to implement simple features and improve the code reuse rate. SQL functions provide the following features:

- Allows you to reference and call SQL UDFs in SQL scripts.
- Allows you to pass parameters of the function type to built-in functions, UDFs, or SQL functions when you call SQL functions.
- Allows you to pass parameters of the anonymous function type to anonymous functions when you call SQL functions.

You can use SQL functions of MaxCompute to solve the following issues:

- In most cases, a large amount of similar code exists, which is inconvenient to maintain and prone to errors. If you want to use UDFs, you must develop code, compile the code in Java, and then create resources and functions. The process is complex. In addition, UDFs cannot catch up with built-in functions in terms of performance. The following code provides an example:

```
select
    nvl(str_to_map(get_json_object(col, '$.key1')), 'default') as key1,
    nvl(str_to_map(get_json_object(col, '$.key2')), 'default') as key2,
    ...
    nvl(str_to_map(get_json_object(col, '$.keyN')), 'default') as keyN
from t;
```

- One function can be passed to another function as a parameter by using code that is similar to lambda expressions in Java.

Remarks

When you call an SQL function, make sure that the type of the read data meets the requirements of SQL functions for the data type. Otherwise, the function may fail to be parsed.

When you create, query, call, and delete SQL UDFs, make sure that you have permissions to perform operations on the UDFs. For more information about function permissions and authorization, see Permissions.

Create permanent SQL functions

After you create permanent SQL functions and store them in the metadata system, you can call these functions for all queries. For more information about calling these functions, see Call an SQL function. Command syntax:

```
create sql function <function_name>(@<parameter_in1> <datatype>[, @<parameter_in2> <datatyp
e>...])
[returns @<parameter_out> <datatype>]
as [begin]
<function_expression>
[end];
```

- function_name: required. The name of the SQL function that you want to create. Each function name must be unique and can be registered only once. The SQL function names cannot be the same as

those of built-in functions. You can run the `LIST FUNCTIONS;` command to view all the functions in a project.

- parameter_in: required. The input parameters of the SQL function that you want to create. Function type parameters and anonymous function parameters are supported. For information about function type parameters, see Function type parameters. For information about anonymous function parameters, see Anonymous function parameters.
- datatype: required. The data type of the parameters.
- returns: required. The return value of the function. It is a variable. If you do not specify the returns parameter, the value of function_name is returned by default.
- parameter_out: required. The response parameter of the function.
- function_expression: required. The function expression.

The following statement provides an example:

```
create sql function my_add(@a BIGINT) as @a + 1;
```

In the preceding example, `@a + 1` indicates the logic of the SQL function. You can write it as an expression. The expression can be a built-in operator, built-in function, or UDF.

If the function logic is complex, you can write multiple SQL statements and enclose them with BEGIN and END. returns specifies the return value of the function. If you do not specify this parameter, the value of the function_name parameter is returned by default. The following statement provides an example:

```
create sql function my_sum(@a BIGINT, @b BIGINT, @c BIGINT) returns @my_sum BIGINT
as begin
    @temp := @a + @b;
    @my_sum := @temp + @c;
end;
```

> **Note** To write multiple SQL statements, you must use SQL in script mode. For more information, see Script Mode SQL.

Create temporary SQL functions

If you do not need to store SQL functions in the metadata system of MaxCompute, you can create temporary SQL functions. These functions apply only to the current SQL script. The following statement provides an example:

```
function <function_name>(@<parameter_in1> <datatype>[, @<parameter_in2> <datatype>...])
[returns @<parameter_out> <datatype>]
as [begin]
<function_expression>
[end];
```

For more information about parameters in this statement, see Create permanent SQL functions.

> **Note** To write multiple SQL statements, you must use SQL in script mode. For more information, see Script Mode SQL.

The following statement provides an example:

```
function my_add(@a BIGINT) as @a + 1;
```

Query the information of an SQL function

You can query an SQL function in the same way that you query a Java or Python UDF. The following statement provides an example:

```
desc function <function_name>;
```

function_name: the name of the created SQL function.

The following statement provides an example:

```
desc function my_add;
```

The following result is returned:

```
Name                                    my_add
Owner                                   ALIYUN$santie_doctest@test.aliyunid.com
Created Time                            2021-05-08 11:26:02
SQL Definition Text                     CREATE SQL FUNCTION MY_ADD(@a BIGINT) AS @a + 1
```

Note If you want to query an SQL function on the MaxCompute client, make sure that the client version is later than 0.34.0. For more information about how to view the version of the MaxCompute client and perform operations on the client, see MaxCompute client.

Delete an SQL function

You can delete an SQL function in the same way that you delete a Java or Python UDF. Syntax:

```
drop function <function_name>;
```

function_name: the name of the created SQL function.

The following statement provides an example:

```
drop function my_add;
```

Call an SQL function

You can call an SQL function in the same way that you call a built-in function. The following statement provides an example:

```
select <function_name>(<column_name>[,...]) from <table_name>;
```

- function_name: the name of the created SQL function.
- column_name: the column name of the table from which you want to query data. The data type of the column must be the same as the data type defined by the SQL function.

- table_name: the name of the table from which you want to query data.

The following statement provides an example:

```
-- Create a table named src.
create table src (c bigint, d string);
insert into table src values (1,100.1),(2,100.2),(3,100.3);
-- Call the my_add function.
select my_add(c) from src;
-- The following result is returned.
+------------+
| _c0        |
+------------+
| 2          |
| 3          |
| 4          |
+------------+
```

Function type parameters

When you call an SQL function, you can pass in built-in functions, UDFs, or other SQL functions. The following statement provides an example:

```
function add(@a BIGINT) as @a + 1;
function op(@a BIGINT, @fun function (BIGINT) returns BIGINT) as @fun(@a);
select op(key, add), op(key, abs) from values (1),(2) as t (key);
-- The following result is returned.
+------------+------------+
| _c0        | _c1        |
+------------+------------+
| 2          | 1          |
| 3          | 2          |
+------------+------------+
```

In the example, two input parameters are specified for the `OP` function. `@a` specifies a value of the BIGINT type. `@fun` specifies a function whose input and output parameters are of the BIGINT type. The `OP` function passes `@a` to the function that is specified by `@fun` and then to the ADD and ABS functions for processing `@a`. For more information about the ABS function, see Mathematical functions.

Anonymous function parameters

You can pass parameters of the anonymous function type to anonymous functions when you call SQL functions. The following statement provides an example:

```
function op(@a BIGINT, @fun function (BIGINT) returns BIGINT) as @fun(@a);
select op(key, function (@a) as @a + 1) from values (1),(2) as t (key);
```

In the example, `function (@a) as @a + 1` is an anonymous function. You do not need to specify a data type for the `@a` parameter. The compiler infers the data type of `@a` based on the parameter definition of the OP function.

Examples

Scenario: Convert the format of a date from `yyyy-mm-dd` to `yyyymmdd`.

For example, the formats of the following dates need to be converted: 2020-11-21, 2020-1-01, 2019-5-1, and 19-12-1.

Solutions:

- Solution 1: Use an SQL function. We **recommend** that you use this solution. The following statement shows an example:

```
create sql function y_m_d2yyyymmdd(@y_m_d string) returns @yyyymmdd string
as begin
    @yyyymmdd := concat(lpad(split_part(@y_m_d, '-', 1), 4, '0'), lpad(split_part(@y_m_d, '-', 2), 2, '0'), lpad(split_part(@y_m_d, '-', 3), 2, '0')) ;
end;
select y_m_d2yyyymmdd(d) from values('2020-11-21'),('2020-1-01'), ('2019-5-1'), ('19-12-1') t (d);
```

  The following result is returned:

```
+------------+
| _c0        |
+------------+
| 20201121   |
| 20200101   |
| 20190501   |
| 00191201   |
+------------+
```

- Solution 2: Repeatedly call a function, which decreases the code reuse rate. Therefore, we **recommend that you do not use** this solution. The following statement shows an example:

```
select concat(lpad(split_part(d, '-', 1), 4, '0'), lpad(split_part(d, '-', 2), 2, '0'), lpad(split_part(d, '-', 3), 2, '0')) from values('2020-11-21'),('2020-1-01'), ('2019-5-1'), ('19-12-1') t (d);
```

> **Note** To write multiple SQL statements, you must use SQL in script mode. For more information, see Script Mode SQL.

3.10.7. Open source geospatial UDFs

This topic describes how to use open source geospatial user-defined functions (UDFs) to analyze spatial data.

Prerequisites

- Git is installed.
- Maven is installed and environment variables are configured.
- The MaxCompute client is installed.

  For more information about how to install the MaxCompute client, see Install and configure the MaxCompute client.

Context

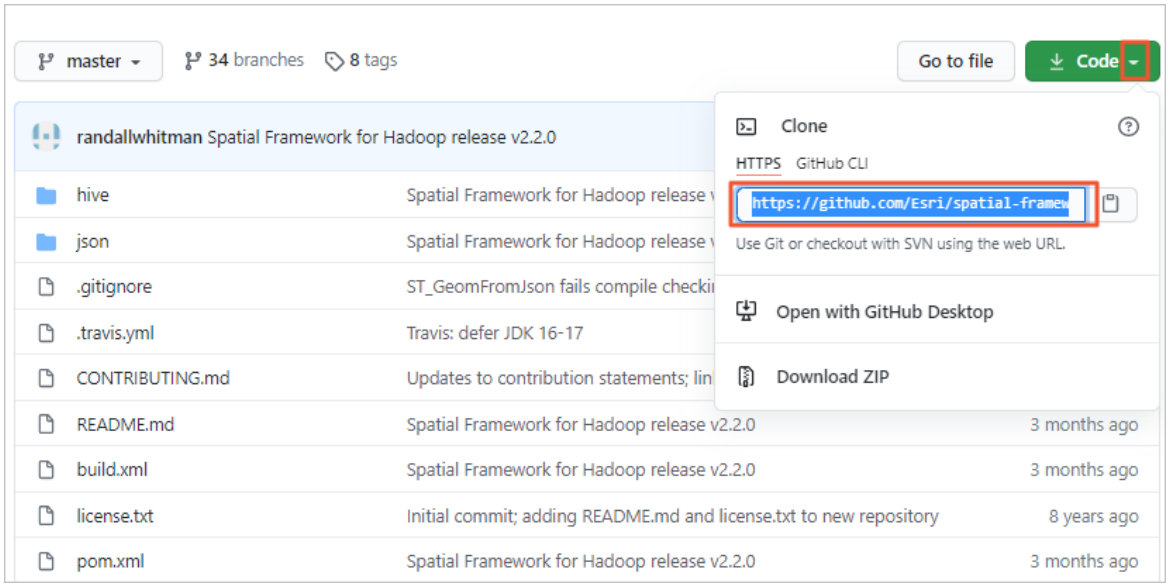
Apache Hive provides a set of open source geospatial UDFs. For more information, visit GitHub. MaxCompute allows you to directly use Hive UDFs, including Hive geospatial functions, in MaxCompute.

For more information about how to use Hive UDFs in MaxCompute, see Hive UDFs.

> **Note** If you encounter an issue when you use UDFs, submit your issue on GitHub for help.

Step 1: Prepare a local UDF

1. Obtain the URL to download the code package of geospatial UDFs.



2. Open the Git command-line tool, and run one of the following commands to download the geospatial UDF code package for Hive 2.1.0 to your on-premises machine. Hive 2.1.0 corresponds to Hadoop 2.7.2.

   Example:

   ```
   git clone https://github.com/Esri/spatial-framework-for-hadoop.git
   ```

   Or

   ```
   git clone -b "v2.1.0" --single-branch git@github.com:Esri/spatial-framework-for-hadoop.
   git
   ```

3. Run the following commands to build a project by using Maven:

   Example:

   ```
   cd spatial-framework-for-hadoop
   mvn clean package -DskipTests -P java-8,hadoop-2.7,hive-2.1
   ```

4. Run the following command to copy the created JAR package. This JAR package contains all methods of the open source geospatial UDFs.

   Example:

   ```
   cp hive/target/spatial-sdk-hive-2.1.1-SNAPSHOT.jar .../spatial-sdk-hive.jar
   ```

5. Run one of the following commands to download the JAR packages that the project depends on:

```
wget https://repo1.maven.org/maven2/com/esri/geometry/esri-geometry-api/2.2.0/esri-geometry-api-2.2.0.jar .../esri-geometry-api.jar
```

Or

```
cp ~/.m2/repository/com/esri/geometry/esri-geometry-api/2.2.0/esri-geometry-api-2.2.0.jar .../esri-geometry-api.jar
```

Step 2: Register UDFs with MaxCompute

1. Run the following commands on the MaxCompute client to upload the two JAR packages that the project depends on to the project:

   For more information about how to add resources, see Resource operations.

```
add jar esri-geometry-api.jar;
add jar spatial-sdk-hive.jar;
```

2. Run the following commands to register the UDFs:

```
CREATE FUNCTION ST_Aggr_ConvexHull  AS  'com.esri.hadoop.hive.ST_Aggr_ConvexHull' USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Aggr_Intersection    AS  'com.esri.hadoop.hive.ST_Aggr_Intersection' USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Aggr_Union   AS  'com.esri.hadoop.hive.ST_Aggr_Union'  USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Area AS  'com.esri.hadoop.hive.ST_Area'    USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_AsBinary AS  'com.esri.hadoop.hive.ST_AsBinary'    USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_AsGeoJson    AS  'com.esri.hadoop.hive.ST_AsGeoJson'   USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_AsJson   AS  'com.esri.hadoop.hive.ST_AsJson'  USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_AsShape  AS  'com.esri.hadoop.hive.ST_AsShape' USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_AsText   AS  'com.esri.hadoop.hive.ST_AsText'  USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Bin  AS  'com.esri.hadoop.hive.ST_Bin' USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_BinEnvelope  AS  'com.esri.hadoop.hive.ST_BinEnvelope' USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Boundary AS  'com.esri.hadoop.hive.ST_Boundary'    USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Buffer   AS  'com.esri.hadoop.hive.ST_Buffer'  USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Centroid AS  'com.esri.hadoop.hive.ST_Centroid'    USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Contains AS  'com.esri.hadoop.hive.ST_Contains'    USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_ConvexHull   AS  'com.esri.hadoop.hive.ST_ConvexHull'  USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_CoordDim AS  'com.esri.hadoop.hive.ST_CoordDim'    USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
```

```
CREATE FUNCTION ST_Crosses  AS  'com.esri.hadoop.hive.ST_Crosses' USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Difference   AS  'com.esri.hadoop.hive.ST_Difference'  USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Dimension    AS  'com.esri.hadoop.hive.ST_Dimension'   USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Disjoint AS  'com.esri.hadoop.hive.ST_Disjoint'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Distance AS  'com.esri.hadoop.hive.ST_Distance'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_EndPoint AS  'com.esri.hadoop.hive.ST_EndPoint'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Envelope AS  'com.esri.hadoop.hive.ST_Envelope'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_EnvIntersects    AS  'com.esri.hadoop.hive.ST_EnvIntersects'   USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Equals   AS  'com.esri.hadoop.hive.ST_Equals'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_ExteriorRing AS  'com.esri.hadoop.hive.ST_ExteriorRing'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeodesicLengthWGS84  AS  'com.esri.hadoop.hive.ST_GeodesicLengthWGS8
4' USING 'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeomCollection   AS  'com.esri.hadoop.hive.ST_GeomCollection'  USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Geometry AS  'com.esri.hadoop.hive.ST_Geometry'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeometryN    AS  'com.esri.hadoop.hive.ST_GeometryN'   USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeometryType AS  'com.esri.hadoop.hive.ST_GeometryType'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeomFromGeoJson  AS  'com.esri.hadoop.hive.ST_GeomFromGeoJson' USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeomFromJson AS  'com.esri.hadoop.hive.ST_GeomFromJson'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeomFromShape    AS  'com.esri.hadoop.hive.ST_GeomFromShape'   USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeomFromText AS  'com.esri.hadoop.hive.ST_GeomFromText'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_GeomFromWKB  AS  'com.esri.hadoop.hive.ST_GeomFromWKB' USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_InteriorRingN    AS  'com.esri.hadoop.hive.ST_InteriorRingN'   USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Intersection AS  'com.esri.hadoop.hive.ST_Intersection'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Intersects   AS  'com.esri.hadoop.hive.ST_Intersects'  USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Is3D AS  'com.esri.hadoop.hive.ST_Is3D'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_IsClosed AS  'com.esri.hadoop.hive.ST_IsClosed'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_IsEmpty  AS  'com.esri.hadoop.hive.ST_IsEmpty' USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_IsMeasured   AS  'com.esri.hadoop.hive.ST_IsMeasured'  USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
```

```
CREATE FUNCTION ST_IsRing   AS  'com.esri.hadoop.hive.ST_IsRing'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_IsSimple AS  'com.esri.hadoop.hive.ST_IsSimple'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Length   AS  'com.esri.hadoop.hive.ST_Length'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_LineFromWKB  AS  'com.esri.hadoop.hive.ST_LineFromWKB' USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_LineString   AS  'com.esri.hadoop.hive.ST_LineString'  USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_M    AS  'com.esri.hadoop.hive.ST_M'   USING 'spatial-sdk-hive.jar,e
sri-geometry-api.jar';
CREATE FUNCTION ST_MaxM AS  'com.esri.hadoop.hive.ST_MaxM'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MaxX AS  'com.esri.hadoop.hive.ST_MaxX'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MaxY AS  'com.esri.hadoop.hive.ST_MaxY'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MaxZ AS  'com.esri.hadoop.hive.ST_MaxZ'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MinM AS  'com.esri.hadoop.hive.ST_MinM'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MinX AS  'com.esri.hadoop.hive.ST_MinX'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MinY AS  'com.esri.hadoop.hive.ST_MinY'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MinZ AS  'com.esri.hadoop.hive.ST_MinZ'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_MLineFromWKB AS  'com.esri.hadoop.hive.ST_MLineFromWKB'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_MPointFromWKB    AS  'com.esri.hadoop.hive.ST_MPointFromWKB'   USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_MPolyFromWKB AS  'com.esri.hadoop.hive.ST_MPolyFromWKB'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_MultiLineString  AS  'com.esri.hadoop.hive.ST_MultiLineString' USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_MultiPoint   AS  'com.esri.hadoop.hive.ST_MultiPoint'  USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_MultiPolygon AS  'com.esri.hadoop.hive.ST_MultiPolygon'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_NumGeometries    AS  'com.esri.hadoop.hive.ST_NumGeometries'   USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_NumInteriorRing  AS  'com.esri.hadoop.hive.ST_NumInteriorRing' USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_NumPoints    AS  'com.esri.hadoop.hive.ST_NumPoints'   USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Overlaps AS  'com.esri.hadoop.hive.ST_Overlaps'    USING 'spatial-sd
k-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Point    AS  'com.esri.hadoop.hive.ST_Point'   USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_PointFromWKB AS  'com.esri.hadoop.hive.ST_PointFromWKB'    USING 'sp
atial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_PointN   AS  'com.esri.hadoop.hive.ST_PointN'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
```

```
CREATE FUNCTION ST_PointZ   AS  'com.esri.hadoop.hive.ST_PointZ'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_PolyFromWKB  AS  'com.esri.hadoop.hive.ST_PolyFromWKB' USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Polygon  AS  'com.esri.hadoop.hive.ST_Polygon' USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Relate   AS  'com.esri.hadoop.hive.ST_Relate'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_SetSRID  AS  'com.esri.hadoop.hive.ST_SetSRID' USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_SRID AS  'com.esri.hadoop.hive.ST_SRID'    USING 'spatial-sdk-hive.j
ar,esri-geometry-api.jar';
CREATE FUNCTION ST_StartPoint   AS  'com.esri.hadoop.hive.ST_StartPoint'  USING 'spatia
l-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_SymmetricDiff    AS  'com.esri.hadoop.hive.ST_SymmetricDiff'   USING
'spatial-sdk-hive.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Touches  AS  'com.esri.hadoop.hive.ST_Touches' USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Union    AS  'com.esri.hadoop.hive.ST_Union'   USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_Within   AS  'com.esri.hadoop.hive.ST_Within'  USING 'spatial-sdk-hi
ve.jar,esri-geometry-api.jar';
CREATE FUNCTION ST_X    AS  'com.esri.hadoop.hive.ST_X'   USING 'spatial-sdk-hive.jar,e
sri-geometry-api.jar';
CREATE FUNCTION ST_Y    AS  'com.esri.hadoop.hive.ST_Y'   USING 'spatial-sdk-hive.jar,e
sri-geometry-api.jar';
CREATE FUNCTION ST_Z    AS  'com.esri.hadoop.hive.ST_Z'   USING 'spatial-sdk-hive.jar,e
sri-geometry-api.jar';
```

Step 3: Test a UDF

Submit an SQL statement on the MaxCompute client to test whether a UDF can be normally used. Example:

```
-- Enable Hive compatibility and submit a UDF for testing.
set odps.sql.hive.compatible=true;
select ST_AsText(ST_Point(1, 2));
```

The following command output indicates that the UDF is registered with MaxCompute.

```
+-----+
| _c0 |
+-----+
| POINT (1 2) |
+-----+
```

3.10.8. FAQ about MaxCompute UDFs

3.10.8.1. FAQ about MaxCompute Java UDFs

This topic provides answers to some frequently asked questions about MaxCompute user-defined functions (UDFs) that are written in Java.

Class- or dependency-related issues

When you call a MaxCompute UDF, the following issues related to classes or dependencies may occur:

- Issue 1: The error message `ClassNotFoundException` or `Some dependencies are missing` appears.
  - Causes:
    - Cause 1: The JAR file that is specified when you create the UDF is invalid.
    - Cause 2: One or more JAR files on which the UDF depends are not uploaded to MaxCompute. For example, the required third-party package is not uploaded.
    - Cause 3: The UDF is not called in the MaxCompute project in which the UDF is created. For example, a MaxCompute UDF is created in a development project but is called in a production project.
    - Cause 4: The required file does not exist or the resource type is invalid. For example, the type of the uploaded file is PY, but the file type specified in `get_cache_file` of the UDF code is FILE.
  - Solutions:
    - Solution to Cause 1: Check the content of the JAR file and confirm that the JAR file contains all the required classes. Then, repackage resources into a JAR file, and upload the file to your MaxCompute project. For more information about how to package resources into files and upload the files to your MaxCompute project, see Package a Java program, upload the package, and create a MaxCompute UDF.
    - Solution to Cause 2: Upload the required third-party package as a resource to the MaxCompute project. Then, add this package to the resource list when you create the UDF. For more information about how to upload resources and create UDFs, see Add resources and Create a UDF.
    - Solution to Cause 3: On the MaxCompute client, run the `list functions;` command for the project in which the MaxCompute UDF is called. Then, confirm that the MaxCompute UDF is displayed in the command output and the classes and required resources of the MaxCompute UDF are valid.
    - Solution to Cause 4: On the MaxCompute client, run the `desc function <function_name>;` command. Then, confirm that all the required files are displayed in the resource list in the command output. If the type of the uploaded file is inconsistent with the file type specified in get_cache_file, you can run the `add <file_type> <file_name>;` command to add the required file.
- Issue 2: The error message `NoClassDefFoundError` or `NoSuchMethodError` appears.
  - Causes:
    - Cause 1: The version of the third-party library in the uploaded JAR file is inconsistent with the version of the built-in third-party library in MaxCompute.
    - Cause 2: This issue occurs due to sandbox limits. The detailed error information `java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "createClassLoader")` appears in the standard error of the job instance. The information indicates that this issue occurs due to sandbox limits. When the UDF runs in a distributed environment, the UDF is subject to sandbox limits. For more information about the sandbox limits, see Java sandbox.

- Solutions:
  - Solution to Cause 1: Use `maven-shade-plugin` to resolve the version inconsistency issue, change the path for storing JAR files of the UDF, repackage resources into a JAR file, and then upload the file to your MaxCompute project. For more information about how to package resources into files and upload the files to your MaxCompute project, see Package a Java program, upload the package, and create a MaxCompute UDF.
  - Solution to Cause 2: Follow the instructions provided in Issues related to Java sandbox limits.

Issues related to Java sandbox limits

- Issue 1: **An error occurs when a MaxCompute UDF is called to access local files, access the Internet, access a distributed file system, or create a Java thread.**
- Causes:
  - Cause 1: The error occurs due to sandbox limits when the MaxCompute UDF runs in a distributed environment. For more information about the sandbox limits, see Java sandbox.
  - Cause 2: The MaxCompute UDF does not support access to the Internet.
- Solutions:
  - Solution to Cause 1: Run the `set odps.isolation.session.enable=true;` command to enable a sandbox at the session level. If the issue persists after the sandbox is enabled, fill in an application form to join the related DingTalk group and feed back the issue to the MaxCompute technical support team.
  - Solution to Cause 2: Fill in and submit a network connection application form based on your business requirements. After the MaxCompute technical support team receives the application, the team contacts you for establishing network connections. For more information about how to fill in application forms, see Network connection process.

Performance-related issues

When you call a MaxCompute UDF, the following performance issues may occur:

- Issue 1: **The error message `kInstanceMonitorTimeout` appears.**
  - Cause: Data processing of the MaxCompute UDF times out. By default, the duration in which the UDF processes data is limited. In most cases, a UDF must process 1024 rows of data at a time within 1800s. This duration is not the total duration in which a worker runs but the duration in which the UDF processes a small batch of data records at a time. In most cases, MaxCompute SQL can process more than 10,000 rows of data per second. This limit aims only to prevent infinite loops in a MaxCompute UDF. If an infinite loop occurs, CPU resources are occupied for a long period of time.

- Solution:
  - If MaxCompute needs to process a large amount of data, you can call `ExecutionContext.claimAlive` in the Java class method of the MaxCompute UDF to reset the timer.
  - Optimize the logic of the MaxCompute UDF code. After the optimization, you can configure the following parameters at the session level to adjust the operation of the MaxCompute UDF before you call the MaxCompute UDF. This way, the data processing is accelerated.

Parameter	Description
`set odps.function.timeout=xxx;`	The timeout period for running a UDF. Default value: 1800. Unit: second. You can increase the value of this parameter based on your business requirements. Valid values: 1 to 3600.
`set odps.stage.mapper.split.size=xxx;`	The input data amount of a Map worker. Default value: 256. Unit: MB. You can decrease the value of this parameter based on your business requirements.
`set odps.sql.executionengine.batch.rowcount=xxx;`	The number of rows that MaxCompute can process at a time. Default value: 1024. You can decrease the value of this parameter based on your business requirements.

- Issue 2: The error message `errMsg:SigKill(OOM)` or `OutOfMemoryError` appears.
  - Cause: MaxCompute runs jobs in three stages: Map, Reduce, and Join. If MaxCompute processes a large amount of data, the data processing of each instance at each stage is time-consuming.
  - Solution:

- If the error is reported for the `fuxi` or `runtime` code, you can configure the following resource parameters to accelerate the data processing.

Parameter	Description
`set odps.stage.mapper.mem=xxx;`	The memory size of a Map worker. Default value: 1024. Unit: MB. You can increase the value of this parameter based on your business requirements.
`set odps.stage.reducer.mem=xxx;`	The memory size of a Reduce worker. Default value: 1024. Unit: MB. You can increase the value of this parameter based on your business requirements.
`set odps.stage.joiner.mem=xxx;`	The memory size of a Join worker. Default value: 1024. Unit: MB. You can increase the value of this parameter based on your business requirements.
`set odps.stage.mapper.split.size=xxx;`	The input data amount of a Map worker. Default value: 256. Unit: MB. You can increase the value of this parameter based on your business requirements.
`set odps.stage.reducer.num=xxx;`	The number of Reduce workers. You can increase the value of this parameter based on your business requirements.
`set odps.stage.joiner.num=xxx;`	The number of Join workers. You can increase the value of this parameter based on your business requirements.

- If this error is reported for Java code, you can adjust the preceding parameters and run the `set odps.sql.udf.jvm.memory=xxx;` command to increase the Java virtual machine (JVM) memory size.

For more information about the parameters, see SET operations.

UDTF-related issues

When you call a Java UDTF, the following issues may occur:

- Issue 1: The error message `Semantic analysis exception - only a single expression in the SELECT clause is supported with UDTF's` appears when you call a UDTF.
  - Cause: Columns or expressions are specified in the SELECT statement in which the UDTF is called. The following sample code shows an incorrect SQL statement:

```
select b.*, 'x', udtffunction_name(v) from table lateral view udtffunction_name(v) b as f1, f2;
```

  - Solution: You can use the Java UDTF with Lateral View in the SELECT statement. Sample statement:

```
select b.*, 'x' from table lateral view udtffunction_name(v) b as f1, f2;
```

- Issue 2: **The error message `Semantic analysis exception - expect 2 aliases but have 0` appears.**
  - Cause: The aliases of the output columns are not specified in the SELECT statement in which the UDTF is called.
  - Solution: You can use the `AS` clause to specify the aliases of output columns in the SELECT statement in which the Java UDTF is called. Sample statement:

    ```
    select udtffunction_name(paramname) as (col1, col2);
    ```

3.10.8.2. FAQ about MaxCompute Python UDFs

This topic provides answers to some frequently asked questions about MaxCompute user-defined functions (UDFs) that are written in Python.

Classes and resources

The following class or resource issues may occur:

- Problem 1: **When I call a MaxCompute UDF, the `function 'xxx' cannot be resolved` error is reported.**
  - Possible causes:
    - Cause 1: The project that you use is not the project in which the MaxCompute UDF is registered. For example, if you register a MaxCompute UDF in a development project but call the MaxCompute UDF in a production project, the error is reported.
    - Cause 2: The classes or resources of the MaxCompute UDF are invalid.
    - Cause 3: The type of a resource on which the MaxCompute UDF depends is invalid. For example, you upload a Python file, which is of the PY type. However, `get_cache_file` in the MaxCompute UDF code requires a resource of the FILE type.
    - Cause 4: The data of a resource on which the MaxCompute UDF depends is not the latest. If you upload a resource to MaxCompute by using DataWorks, a delay may occur. In this case, the data of the resource on which the MaxCompute UDF depends is not the latest.
    - Cause 5: The Python version is not supported. By default, MaxCompute uses Python 2 to run jobs. If non-ASCII characters exist in the Python code, an error is reported when you run the code.

- Solutions:
  - Solution 1: On the MaxCompute client, run the `list functions;` command in the project where the error is reported. If the MaxCompute UDF is not contained in the command output, the project is not the project in which the MaxCompute UDF is registered. Go to the project in which the MaxCompute UDF is registered and call the MaxCompute UDF again.
  - Solution 2: On the MaxCompute client, run the `desc function <function_name>;` command and check the values of Class and Resources in the command output.

    If the value of the Class or Resources parameter is invalid, run the `create function <function_name> as <'package_to_class'> using <'resource_list'>;` command to register the UDF again. Configure the package_to_class parameter in the Python script name. Class name format. Use the resource_list parameter to specify all the resources that you want to reference in MaxCompute, such as files, tables, compressed packages, and third-party packages.

    For more information about how to register a function, see Create a UDF.
  - Solution 3: On the MaxCompute client, run the `desc resource <resource_name>;` command. Check whether the resource type that is specified by the Type parameter in the command output is valid. If the resource type is invalid, run the `add <file_type> <file_name>;` command to upload the resource again.
    - If the resource reference method that is used in the MaxCompute UDF code is `get_cache_file`, a file is referenced and the resource type must be FILE.
    - If the resource reference method that is used in the MaxCompute UDF code is `get_cache_table`, a table is referenced and the resource type must be TABLE.
    - If the resource reference method that is used in the MaxCompute UDF code is `get_cache_archive`, a compressed package is referenced and the resource type must be ARCHIVE.

    For more information about how to upload resources, see Add resources.
  - Solution 4: On the MaxCompute client, run the `desc resource <resource_name>;` command. Check whether the resource data is the latest based on the value of the LastModifiedTime parameter in the command output. You must use the latest resource in the MaxCompute UDF.
  - Solution 5: Add `#coding:utf-8` or `# -*- coding: utf-8 -*-` to the Python code header to declare the encoding format. You can also add the `set odps.sql.python.version=cp37;` command before the SQL statement that is used to call the MaxCompute UDF and submit the command together with the SQL statement. This way, Python 3 is used to run jobs.

- Problem 2: **When I call a MaxCompute UDF in which `get_cache_archive('xxx.zip')` is configured, one of the following errors is reported: `IOError: Download resource: xxx.zip failed`, `odps.distcache.DistributedCacheError`, and `fuxi job failed: Download resource failed: xxx.zip`.**

- Possible causes:
  - Cause 1: The compressed package does not exist. When you register the MaxCompute UDF, you do not specify a compressed package in the command.
  - Cause 2: The resource type of the compressed package is not ARCHIVE.
  - Cause 3: The package name or file name extension in the code is inconsistent with the actual package name or file name extension. For example, xxx.zip is specified in the code but the uploaded file is xxx.tar.gz. In this case, the package is decompressed in the ZIP format. As a result, the decompression fails and an error is reported.
  - Cause 4: Two UDFs that are in the same job depend on resources that have the same name. However, the resources exist in different projects.
- Solutions:
  - Solution 1: On the MaxCompute client, run the `desc function <function_name>;` command. Check whether the value of the Resources parameter in the command output contains the name of the compressed package in the error message.

    If the name of the compressed package does not exist, run the `create function <function_name> as <'package_to_class'> using <'resource_list'>;` command to register the MaxCompute UDF again. Add the name of the compressed package to the resource_list parameter in the command.

    For more information about how to register a function, see Create a UDF.
  - Solution 2: On the MaxCompute client, run the `desc resource <resource_name>;` command. Check whether the value of the Type parameter in the command output is ARCHIVE.

    If the value of the Type parameter is not ARCHIVE, run the `add archive <file_name>;` command to upload the compressed package again.

    For more information about how to upload resources, see Add resources.
  - Solution 3: On the MaxCompute client, run the `desc function <function_name>;` command. Check whether the package name and file name extension in the value of the Resources parameter in the command output are consistent with the actual package name and file name extension.

    If the package name or file name extension in the command output is inconsistent with the actual package name or file name extension, run the `add archive <file_name>;` command to upload the compressed package again. Set the file_name parameter to a value that is consistent with the actual package name and file name extension.
  - Solution 4: Troubleshoot all the UDFs on which the job depends, including the UDFs on which a view depends. Check the projects to which the UDFs belong and the resources that correspond to the UDFs. If the resources that have the same name exist in different projects, we recommend that you modify the UDFs on which the job depends or modify the resource name.

- Problem 3: **When I call a MaxCompute UDF in which `get_cache_table(table_name)` is configured, the following error is reported: `odps.distcache.DistributedCacheError: Table resource "xxx_table_name" not found`.**
  - Possible causes:
    - Cause 1: The table does not exist. When you register the MaxCompute UDF, you do not specify a table in the command.
    - Cause 2: The resource type of the table is not TABLE.

- Solutions:
  - Solution 1: On the MaxCompute client, run the `desc function <function_name>;` command. Check whether the value of the Resources parameter in the command output contains the name of the table in the error message.

    If the name of the table does not exist, run the `create function <function_name> as <'package_to_class'> using <'resource_list'>;` command to register the MaxCompute UDF again. Add the name of the table to the resource_list parameter in the command.

    For more information about how to register a function, see Create a UDF.
  - Solution 2: On the MaxCompute client, run the `desc resource <resource_name>;` command. Check whether the value of the Type parameter in the command output is TABLE.

    If the value of the Type parameter is not TABLE, run the `add table <table_name>;` command to upload the table again.

    For more information about how to upload resources, see Add resources.

- Problem 4: **When I call a MaxCompute UDF that references a third-party package, the following error is reported: `ImportError: No module named 'xxx'`.**
  - Possible causes:
    - Cause 1: The resource type of the third-party package is not ARCHIVE.
    - Cause 2: When you register the MaxCompute UDF, you do not specify a third-party package in the command.
    - Cause 3: You do not specify the path of the third-party package in the MaxCompute UDF code.
    - Cause 4: The third-party package is a WHEEL file but its file name extension is invalid. You must download a WHEEL file based on the Python version.
    - Cause 5: The third-party package is not a WHEEL file or a pure Python package, but the package contains the setup.py file.
    - Cause 6: The name of the Python file for the MaxCompute UDF conflicts with the name of the third-party module that is referenced by the MaxCompute UDF. For example, if the Python file for the MaxCompute UDF is A.py, "import A" in the code imports the A.py file rather than the module in the third-party package.

- Solutions:
  - Solution 1: On the MaxCompute client, run the `desc resource <resource_name>;` command. Check whether the value of the Type parameter in the command output is ARCHIVE.

    If the value of the Type parameter is not ARCHIVE, run the `add archive <file_name>;` command to upload the compressed package again.

    For more information about how to upload resources, see Add resources.
  - Solution 2: On the MaxCompute client, run the `desc function <function_name>;` command. Check whether the value of the Resources parameter in the command output contains the name of the third-party package in the error message.

    If the name of the third-party package does not exist, run the `create function <function_name> as <'package_to_class'> using <'resource_list'>;` command to register the MaxCompute UDF again. Add the name of the third-party package to the resource_list parameter in the command.

    For more information about how to register a function, see Create a UDF.
  - Solution 3: Check whether the path of the third-party package is configured in `sys.path.insert(0, 'work/third-party package path')` in the MaxCompute UDF code. For example, the module name is A and the Python file is A.py. The following examples show you how to determine the path that you need to configure in the code:
    - Example 1: The Python file is located in the resource_dir folder, and this folder is compressed into the resource-of-A.zip package. The path that you need to configure in `sys.path.insert` is `work/resource-of-A.zip/resource_dir/`.
    - Example 2: The Python file is located in the resource_dir folder, and all files in this folder are compressed into the resource-of-A.zip package. The path that you need to configure in `sys.path.insert` is `work/resource-of-A.zip/`.
    - Example 3: The Python file is located in the resource_dir/path1/path2 folder and all files in the resource_dir folder are compressed into the resource-of-A.zip package. The path that you need to configure in `sys.path.insert` is `work/resource-of-A.zip/path1/path2/`.

    > **Note** By default, resources of the ARCHIVE type are placed in the relative path `./work/` of the execution path of the MaxCompute UDF.
  - Solution 4: The WHEEL file varies based on the Python version. If you use Python 2, you must download the WHEEL file whose name contains `cp27-cp27m-manylinux1_x86_64`. If you use Python 3, you must download the WHEEL file whose name contains `cp37-cp37m-manylinux1_x86_64`. You can change the file name extension of the downloaded WHEEL file to .zip. This way, you do not need to package the WHEEL file into a ZIP file.
  - Solution 5: In a Python environment that is compatible with MaxCompute, compile the setup.py file into a WHEEL file. Then, upload resources and register the MaxCompute UDF. For more information about how to compile a third-party package, see Reference third-party packages that need to be compiled.
  - Solution 6: Change the name of the Python file for the MaxCompute UDF.
- Problem 5: **When I call a MaxCompute UDF that references a standard Python 3 library, the following error is reported: `ImportError: No module named enum`.**

- Cause: Python 3 is not enabled for MaxCompute projects. By default, Python 2 is used to call MaxCompute UDFs. Therefore, the standard Python 3 library cannot be identified.
- Solution: Add the `set odps.sql.python.version=cp37;` command before the SQL statement that is used to call the MaxCompute UDF and submit the command together with the SQL statement.

- Problem 6: **When I call a MaxCompute UDF, the `failed to get Udf info from xxx.py` error is reported.**
  - Cause: In the code of the user-defined table-valued function (UDTF) or user-defined aggregate function (UDAF), the statement used for importing the base class is invalid. For example, if you use `import odps.udf.BaseUDTF` or `import odps.udf.BaseUDAF`, the error is reported.
  - Solution: Change the statement to `from odps.udf import BaseUDTF` or `from odps.udf import BaseUDAF`.

Performance

- Problem: **When I call a MaxCompute UDF, the `kInstanceMonitorTimeout` error is reported.**
- Cause: Data processing of the MaxCompute UDF times out. By default, the duration in which the UDF processes data is limited. In most cases, a UDF must process 1024 rows of data at a time within 1800s. This duration is not the total duration in which a worker runs but the duration in which the UDF processes a small batch of data records at a time. In most cases, MaxCompute SQL can process more than 10,000 rows of data per second. This limit aims only to prevent infinite loops in a MaxCompute UDF. If an infinite loop occurs, CPU resources are occupied for a long period of time.
- Solutions:
  - Add information about logging to the MaxCompute UDF code. This way, you can view the logs to check whether an infinite loop occurs. You can also check whether the duration of processing a single record by the MaxCompute UDF meets your expectation based on the time information recorded in the logs. Add the following information about logging to the code. After your job runs, you can view the logs in StdOut of the Logview UI.
    - Python 2

      ```
      sys.stdout.write('your log')
      sys.stdout.flush()
      ```

    - Python 3

      ```
      print('your log', flush=True)
      ```

- If the amount of data is large, the MaxCompute UDF may run for a long period of time. You can adjust the following parameters to prevent the timeout error.

Parameter	Description
`set odps.function.timeout=xxx;`	The timeout period of the MaxCompute UDF. Default value: 1800s. You can set this parameter to a larger value based on your business requirements. Valid values: 1s to 3600s.
`set odps.sql.executionengine.batch.rowcount=xxx;`	The number of data rows that the MaxCompute UDF processes at a time. Default value: 1024. You can set this parameter to a smaller value based on your business requirements.

Network

- Problem: **When I call a MaxCompute UDF to access the Internet, an error is reported.**
- Cause: MaxCompute UDFs do not support access to the Internet.
- Solution: Fill out and submit the network connection application form based on your business requirements. The MaxCompute technical support team then contacts you to establish a network connection. For more information about how to fill out the application form, see Network connection process.

Sandbox

- Problem: **When I call a MaxCompute UDF, the following error is reported:** `RuntimeError: xxx has been blocked by sandbox` .
- Cause: Some calls to Python UDFs are blocked by sandboxes.
- Solutions:
  - Add the `set odps.isolation.session.enable=true;` command before the SQL statement that is used to call a Python UDF and submit the command together with the SQL statement.
  - By default, `odps.isolation.session.enable` is set to true for Python 3 UDFs.

Encoding

The following encoding problems may occur:

- Problem 1: **When I call a MaxCompute UDF, the following error is reported:** `SyntaxError: Non-ASCII charactor '\xe8' in file xxx. on line yyy` .
  - Cause: The Python file for the MaxCompute UDF contains non-ASCII characters and runs in Python 2.

- Solutions:
  - Add the `set odps.sql.python.version=cp37;` command before the SQL statement that is used to call the MaxCompute UDF and submit the command together with the SQL statement. This way, Python 3 is used to run jobs.
  - Add the following information to the beginning of the Python file. This way, the default encoding format of Python 2 is changed to UTF-8.

```
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
```

- Problem 2: **When I call a MaxCompute UDF that is written in Python 2, the following error is reported: `UnicodeEncodeError: 'ascii' code can't encode characters in position x-y: ordinal not in range(128)`.**
  - Cause: The return value type that is specified in the function signature is STRING. However, the MaxCompute UDF returns a Python object of the UNICODE type. For example, if the Python object is named ret, MaxCompute uses `str(ret)` to convert the return value ret to a value of the STR type. If ret is within the ASCII encoding range, it can be converted to the STR type. However, if ret is not within the ASCII encoding range, the conversion fails and an error is reported.
  - Solution: Add the following statement to the `evaluate` method in the Python code:

```
return ret.encode('utf-8')
```

- Problem 3: **When I call a MaxCompute UDF that is written in Python 3, the following error is reported: `UnicodeDecodeError: 'utf-8' codec can't decode byte xxx in position xxx: invalid continuation byte`.**
  - Cause: The input parameter type that is specified in the function signature is STRING. However, the string that is entered when you call the Python 3 UDF cannot be decoded in the UTF-8 format into a Python object of the STR type.
  - Solutions:
    - Do not write strings that are encoded in a format other than UTF-8 to MaxCompute tables.

      A Python 2 UDF returns a GBK-based Python object that is of the STR type. The Python object can be normally written to a MaxCompute table but cannot be read by a Python 3 UDF. Therefore, we recommend that the Python object be encoded in the UTF-8 format before it is returned by the Python 2 UDF. For example, `ret.decode('gbk').encode('utf-8')` can be used to encode the Python object ret.
    - Use the built-in function `is_encoding` in an SQL statement to filter out the data that is encoded in a format other than UTF-8. Example:

```
select py_udf(input_col) from example_table where is_encoding(input_col, 'utf-8', 'utf-8') = true;
```

    - Change the input parameter type that is specified in the function signature of the Python code to BINARY. Then, convert the data type of the columns that are of the STRING type to the BINARY type in the SQL statement and use the columns as the input parameters of the Python 3 UDF. Example:

```
select py_udf(cast(input_col as binary)) from example_table;
```

Function signature

The following function signature problems may occur:

- Problem 1: **When I call a MaxCompute UDF, the following error is reported:** `resolve annotation of class xxx for UDTF/UDF/UDAF yyy contains invalid content '<EOF>'` .
  - Cause: The input or output parameters of a MaxCompute UDF are of a complex data type, and the related information in the function signature is invalid.
  - Solution: Modify the information about the complex data type in the function signature to make sure that the function signature is valid. For more information about function signatures, see Function signatures and data types.
- Problem 2: **When I call a MaxCompute UDF, the following error is reported:** `TypeError: expected <class 'xxx'> but <class 'yyy'> found, value:zzz` .
  - Cause: The return value type that is specified in the function signature is inconsistent with the data type of the data that is returned by the MaxCompute UDF.
  - Solution: Confirm the expected data type and modify the function signature or MaxCompute UDF code to ensure consistency.
- Problem 3: **When I call a MaxCompute UDF, the following error is reported:** `Semantic analysis exception - evaluate function in class xxx.yyy for user defined function zz does not match annotation ***->***` .
  - Cause: The number of input parameters that are specified in the function signature is inconsistent with the number of input parameters for the related method in the MaxCompute UDF code.
  - Solution: Confirm the actual number of input parameters and modify the function signature or MaxCompute UDF code to ensure consistency.

Third-party packages

- Problem: **When I call a MaxCompute UDF, the following error is reported:** `GLIBCXX_x.x.x not found` .
- Cause: The GLIBCXX version on which the .so library file depends is later than the version that is supported by MaxCompute. The same issue may occur on glibc and CXXABI versions.
- Solution: Use a compatible wheel package or recompile the .so library file in an environment that is compatible with MaxCompute. MaxCompute supports the following latest versions on which binary executable files or .so library files depend:

```
GLIBC <= 2.17
CXXABI <= 1.3.8
GLIBCXX <= 3.4.19
GCC <= 4.2.0
```

UDTF

- Problem: **When I call a MaxCompute UDTF, the following error is reported:** `Semantic analysis exception - expect 2 aliases but have 0` .
- Cause: No output column name is specified in the Python UDTF code.
- Solution: Use the `AS` clause to specify column names in the SELECT statement that is used to call the Python UDTF. Example:

```
select my_udtf(col0, col1) as (ret_col0, ret_col1, ret_col2) from tmp1;
```

UDAFs

- Problem 1: **When I call a MaxCompute UDAF, the following error is reported: `Script exception - ValueError: unmarshallable object`.**
  - Cause: The value of the `buffer` parameter in the Python UDAF code is an unmarshallable object. For more information, see Marshal.
  - Solution: When you assign a value to `buffer`, make sure that the value is a marshallable object. If you want to use two `buffers` that are of the LIST and DICT types in the Python UDAF, you must use `return [list(), dict()]` in the `new_buffer` method. When you use `buffer or pbuffer` in the `iterate, merge, or terminate` method, the `buffer` of the LIST type corresponds to `buffer[0] or pbuffer[0]`, and the `buffer` of the DICT type corresponds to `buffer[1] or pbuffer[1]`. If the elements of a `buffer` are of the LIST or DICT type, the elements must also be marshallable objects.
- Problem 2: **When I call a MaxCompute UDAF, the following error is reported: `Python UDAF buffer size overflowed: 2821486749`.**
  - Cause: The `buffer` size in the Python UDAF code exceeds 2 GB after the `marshaling` operation. The `buffer` size is processed in an incorrect manner. As a result, the `buffer` size increases with the data amount.
  - Solution: Rewrite the logic of the Python UDAF code. This way, the `buffer` size does not increase with the data amount. For example, if a `buffer` is a `list`, data cannot be added to the `buffer` in the `iteration` and `merging` phases. For more information about Python UDAFs, see Overview.

3.11. UDT

3.11.1. Overview

MaxCompute introduces user-defined types (UDTs) based on the new-generation SQL engine. UDTs allow you to reference classes or objects of third-party programming languages in SQL statements to call methods or obtain data.

Introduction

UDTs supported by MaxCompute and other SQL engines are different. UDTs supported by other SQL engines are similar to the STRUCT type in MaxCompute. UDTs supported by MaxCompute are similar to the CREATE TYPE statement. A UDT contains both fields and methods. You do not need to use Data Definition Language (DDL) statements to define type mappings in MaxCompute. Instead, MaxCompute allows you to reference types directly in SQL statements. The following examples show how to use UDTs.

For example, to call the *java.lang* package in SQL statements, you can use one of the following methods:

- Use UDTs to call *java.lang*

```
-- Enable new data types. The following example uses a new type of INTEGER (INT).
set odps.sql.type.system.odps2=true;
SELECT java.lang.Integer.MAX_VALUE;
```

Similar to Java, the *java.lang* package can be omitted. Therefore, the preceding statement is equivalent to the following statement:

```
set odps.sql.type.system.odps2=true;
SELECT Integer.MAX_VALUE;
```

The following result is returned:

```
+------------+
| max_value |
+------------+
| 2147483647 |
+------------+
```

- Use user-defined functions (UDFs) to call *java.lang*
  i. Define a UDF class.

  ```
  package com.aliyun.odps.test;
  public class IntegerMaxValue extends com.aliyun.odps.udf.UDF {
    public Integer evaluate() {
       return Integer.MAX_VALUE;
    }
  }
  ```

  ii. Compile the UDF into a JAR package, upload the package, and create a function.

  ```
  add jar odps-test.jar;
  create function integer_max_value as 'com.aliyun.odps.test.IntegerMaxValue' using 'od
  ps-test.jar';
  ```

  iii. Call the function in the SQL statement.

  ```
  select integer_max_value();
  ```

In this example, UDTs simplify the procedure for you to use other programming languages to extend SQL features.

Scenarios

UDTs are suitable for the following scenarios:

- You want to use some features that are not provided by MaxCompute but can be implemented in other programming languages.

  For example, to implement some features, you only need to call built-in Java classes once. However, MaxCompute does not provide methods to implement these features. If you use UDFs to execute these tasks, the procedure is complex.
- You want to call a third-party library in SQL statements to implement the related features. In this scenario, UDFs allow you to directly use a function provided by a third-party library in a SQL statement, instead of wrapping the function inside a UDF.
- You want to directly call the source code of a third-party programming language in SQL statements.

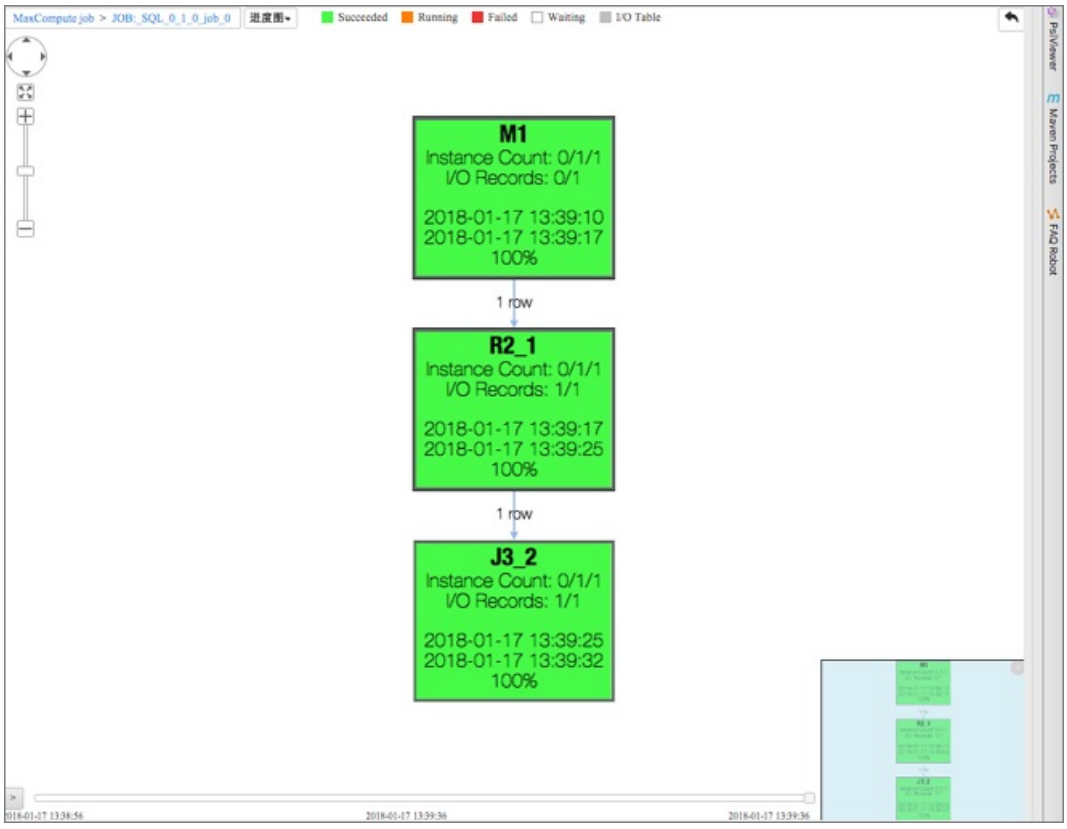
SELECT TRANSFORM allows you to include scripts in SQL statements to make these SQL statements easier to read and maintain. For some programming languages, such as Java, the source code can be executed only after it is compiled. You can use UDTs to reference objects and classes of these languages in SQL statements.

Procedure

The following example shows how to run a UDT:

```
-- Sample data.
@table1 := select * from values ('100000000000000000000') as t(x);
@table2 := select * from values (100L) as t(y);
-- Code logic.
-- Create an object by using the new method.
@a := select new java.math.BigInteger(x) x from @table1;
-- Call a static method.
@b := select java.math.BigInteger.valueOf(y) y from @table2;
-- Call an instance method.
select /*+mapjoin(b)*/ x.add(y).toString() from @a a join @b b;
-- The following result is displayed:
100000000000000000100
```

The following figure shows the process.



This UDT has three stages: M1, R2, and J3. If a `JOIN` operation is used in MapReduce, data must be reshuffled. As a result, data is processed at multiple stages. The processes and physical machines that process data vary based on the stages.

Only the `new java.math.BigInteger(x)` method is called at the M1 stage.

The `java.math.BigInteger.valueOf(y)` and `x.add(y).toString()` methods are separately called at the J3 stage. These methods are called at different stages and executed in different processes and on different physical machines. The UDT encapsulates these stages to achieve an effect like all the stages are implemented on the same JVM.

The preceding example shows that the result of a subquery supports UDT columns. The x column retrieved by variable a is of the `java.math.BigInteger` type rather than a built-in type. You can transfer the UDT data to another operator and then call its method. You can also use the UDT data in a data shuffle.

Features

- UDTs support only Java. By default, all classes of SDK for Java can be referenced by UDTs.

  > **Note** JDK 1.8 is used. A version later than JDK 1.8 may not be supported.

- UDTs also allow you to upload JAR packages and directly reference these packages. Some flags are provided for UDTs.
  - `set odps.sql.session.resources` : specifies the resource that you want to reference. You can specify multiple resources and separate them with commas (,). Example: `set odps.sql.session.resources=foo.sh,bar.txt;` .

    > **Note** This flag functions the same as the flag used to specify resources in the `SELECT TRANSFORM` statement. Therefore, this flag controls two features.

    ```
    set odps.sql.type.system.odps2=true;
    set odps.sql.session.resources=odps-test.jar;
    -- Specify the JAR package to reference. This package must be uploaded to the requi
    red project.
    select new com.aliyun.odps.test.IntegerMaxValue().evaluate();
    ```

  - `odps.sql.session.java.imports` : specifies the default Java package. You can specify multiple packages and separate them with commas (,). This flag is similar to the `IMPORT` statement in Java. You can specify a classpath, such as `java.math.BigInteger` , or use `*` . `Static import` is not supported.

    ```
    set odps.sql.type.system.odps2=true;
    set odps.sql.session.resources=odps-test.jar;
    set odps.sql.session.java.imports=com.aliyun.odps.test. *;
    -- Specify the default JAR package.
    select new IntegerMaxValue().evaluate();
    ```

- UDTs support resource access. In MaxCompute SQL, you can call the static method `com.aliyun.odps.udf.impl.UDTExecutionContext.get()` to obtain the `ExecutionContext` object. Then, you can use this object to access the current `ExecutionContext` class and then access resources, such as files and tables.
- UDTs support the following operations:
  - Create objects by using the `new` method.

- Create arrays by using the `new` method. Initializer lists can be used. Example: `new Integer[] { 1, 2, 3 }`.
- Call methods, including static methods.
- Access fields, including static fields.

> **Note**
> - Only public methods and public fields are supported.
> - Identifiers in UDTs contain the names of packages, classes, methods, and fields. All identifiers are case-sensitive.
> - Anonymous classes and lambda expressions are not supported.
> - UDTs are used in expressions. Functions that do not return values cannot be called in expressions. This issue will be resolved in later versions.

- UDTs support the following data types:
  - UDTs support SQL type conversions, such as `cast(1 as java.lang.Object)`. UDTs do not support Java type conversions, such as `(Object)1`.
  - Java data types are mapped to built-in data types. The mapping can be applied to UDTs. For more information, see the data type mapping table in Java UDFs.
    - You can directly call the method of the Java type to which the built-in type is mapped. Example: `'123'.length() , 1L.hashCode()`.
    - UDTs can be used in built-in functions and UDFs. For example, in `chr(Long.valueOf('100'))`, `Long.valueOf` returns a value of the `java.lang.Long` type. The `CHR` built-in function supports the built-in BIGINT type.
    - The data of a Java primitive type is automatically converted to the boxing type and the preceding two rules apply.

> **Note** For some new built-in data types, you must use `set odps.sql.type.system.odps2=true;` to declare these types. Otherwise, an error occurs.

- The following type conversion rules apply in UDTs:
  - UDT objects can be implicitly converted to the objects of their base classes.
  - UDT objects can be forcibly converted to the objects of their base classes or subclasses.
  - The data type conversion between two objects without inheritance follows native conversion rules. However, such conversions may result in changes to the data. For example, data of the `java.lang.Long` type can be forcibly converted to the `java.lang.Integer` type. This conversion uses the rules that are used to convert the built-in BIGINT type to the INT type. This process may result in changes to the data and even loss of data precision.

  > **Note** UDT objects cannot be saved to disks. This means that UDT objects cannot be `inserted` into tables because DDL statements do not support UDTs. However, if the data type can be implicitly converted to one of the built-in types, you can create tables that contain UDT objects. BINARY is a built-in type and supports automatic serialization. You can save the byte[] arrays to disks. The saved byte[] arrays can be deserialized to the BINARY type. To save UDTs, you must call serialization and deserialization methods to convert the data type to BINARY.

  The output cannot be a UDT. However, you can call the `toString()` method to convert the data type to the `java.lang.String` type because the toString() method supports all Java classes. You can use this method to check UDT data during debugging.

  You can also add the `set odps.sql.udt.display.tostring=true;` flag to enable MaxCompute to convert all output UDT data to strings by using the `java.util.Objects.toString(...)` method. This facilitates debugging. This flag is typically used for debugging because it can be applied only to PRINT statements. It cannot be applied to `INSERT` statements.
- UDTs support Java generics. For example, based on the parameter type, the compiler can determine that the value returned by `java.util.Arrays.asList(new java.math.BigInteger('1'))` is of the `java.util.List<java.math.BigInteger>` type.

  > **Note** You must specify the type parameter in a constructor function or use `java.lang.Object`. This is the same as Java.
  >
  > For example, the result of `new java.util.ArrayList(java.util.Arrays.asList('1', '2'))` is of the `java.util.ArrayList<Object>` type. The result of `new java.util.ArrayList<String>(java.util.Arrays.asList('1', '2'))` is of the `java.util.ArrayList<String>` type.

- All operators use the semantics of MaxCompute SQL.
  - Combination of strings: The result of `String.valueOf(1) + String.valueOf(2)` is 3. The two strings are implicitly converted to DOUBLE-type values and summed. If you use Java string concatenation to combine the strings, the result is 12.
  - `=` operations: The `=` operator in SQL statements is used as a comparison operator. It is used to compare one expression with another expression. You must call the equals method in Java to check whether two objects are equivalent. The `=` operator cannot be used to verify the equivalence of two objects.
- UDTs do not have a clear definition of object equality. This is caused by data reshuffling. Objects may be transmitted between different processes or physical machines. During transmission, an object may be referenced as two different objects. For example, an object may be shuffled to two machines and

then reshuffled. Therefore, when you use UDTs, you must use the `equals` method instead of the `=` operator to verify the equivalence of two objects.

Objects in the same row or column are correlated in some way. However, a correlation between objects in different rows or columns cannot be ensured.

- UDTs cannot be used as shuffle keys in clauses, such as `JOIN`, `GROUP BY`, `DISTRIBUTE BY`, `SORT BY`, `ORDER BY`, or `CLUSTER BY.`

  UDTs can be used at the stages in expressions, but cannot be used as outputs. For example, you cannot call the `group by new java.math.BigInteger('123')` method. However, you can call the `group by new java.math.BigInteger('123').hashCode()` method. This is because the value returned by `hashCode` is of the `int.class` type, which can be used as the built-in INT type. This applies the rules of both the built-in types and specific Java types.

- You can use UDTs to implement the feature provided by the SCALAR function. You can use the COLLECT_SET and Other functions built-in functions with UDTs to implement the features provided by aggregate and table-valued functions.

Benefits

UDTs have the following features:

- UDTs are easy to use. You do not need to define any functions.
- UDTs support all JDK features. This improves SQL flexibility.
- UDT code can be stored in the same file as the SQL code. This facilitates management.
- You can directly reference the libraries of other programming languages and reuse code that you have written in other languages.
- You can create object-oriented features.

Features to be improved:

- Call functions that do not return values and functions that directly use transferred data. For functions that directly use transferred data, their return values are ignored. For example, if you call the `add` method provided by the List interface, this method returns the list that you have transferred.
- Use anonymous classes and lambda expressions.
- Use UDTs as shuffle keys.
- Support more programming languages, such as Python.

Performance

UDTs run in a similar way to UDFs. Therefore, the performance of UDTs is almost the same as that of UDFs. The optimized computing engine improves the performance of UDTs in specific scenarios.

- If a UDT object is used in different processes, it must be serialized and deserialized. If you use UDTs to perform operations that do not require data reshuffling, such as `JOIN` or `AGGREGATE`, the overheads of serialization and deserialization are avoided.
- The runtime of UDTs is based on Codegen rather than reflection. This causes no performance loss. Multiple UDTs can be executed in a single function call. In the preceding example, `values[x].add(values[y]).divide(java.math.BigInteger.valueOf(2))` is called only once. Therefore, even though the operational units of UDTs are small, no additional interface overheads are caused.

Security

Similar to UDFs, UDTs are limited by the Java sandbox model. To perform the operations that are limited, you must enable sandbox isolation or apply to join a sandbox whitelist.

3.11.2. Usage examples

This topic describes common usage examples of user-defined types (UDTs). You can use Java arrays, JSON, complex data types, and table-valued functions in UDTs. You can also aggregate data, overload functions, and reference embedded code in UDTs.

> **Note** Run the following code in script mode. For more information about the script mode, see Script Mode SQL.

Use Java arrays

```
set odps.sql.type.system.odps2=true;
set odps.sql.udt.display.tostring=true;
SELECT
    new Integer[10],    -- Create an array that contains 10 elements.
    new Integer[] {c1, c2, c3},  -- Create an array that contains three elements by initial
izing an ArrayList.
    new Integer[][] { new Integer[] {c1, c2}, new Integer[] {c3, c4} },  -- Create a multid
imensional array.
    new Integer[] {c1, c2, c3} [2], -- Access the elements in the array by using indexes.
    java.util.Arrays.asList(c1, c2, c3);    -- Create a list of the List<Integer> type, whi
ch can be used as an array of the Array<Int> type. This is another way to create a built-in
array.
FROM VALUES (1,2,3,4) AS t(c1, c2, c3, c4);
```

Use JSON

The runtime of a UDT carries a JSON dependency (2.2.4), which can be directly used in JSON.

> **Note** In addition to JSON dependencies, MaxCompute runtime also carries other dependencies, including `commons-logging (1.1.1)`, `commons-lang (2.5)`, `commons-io (2.4)`, and `protobuf-java (2.4.1)`.

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.java.imports=java.util.*,java,com.google.gson. *; -- To import multipl
e packages at a time, separate the packages with commas (,).
@a := select new Gson() gson;   -- Create a Gson object.
select
gson.toJson(new ArrayList<Integer>(Arrays.asList(1, 2, 3))), -- Convert an object to a JSON
string.
cast(gson.fromJson('["a","b","c"]', List.class) as List<String>) -- Deserialize the JSON st
ring. Gson forcibly converts the deserialization result from the List<Object> type to the L
ist<String> type for future use.
from @a;
```

Compared with the built-in function String functions, this UDT-based method is simpler. In addition, this method deserializes the content that is extracted from the JSON string to a supported data type. The deserialization improves efficiency.

Use complex data types

The built-in data type ARRAY maps the `java.util.List` method and the built-in data type MAP maps the `java.util.Map` method.

- Java objects in classes that implement `java.util.List` or `java.util.Map` can be used to process complex-type data in MaxCompute SQL.
- MaxCompute can directly call java.util.List or java.util.Map to process data of the ARRAY or MAP type.

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.java.imports=java.util.*;
select
    size(new ArrayList<Integer>()),        -- Call the built-in function size() to obtain t
he size of the ArrayList.
    array(1,2,3).size(),                   -- Call the built-in function size() of the List
method for the built-in data type ARRAY.
    sort_array(new ArrayList<Integer>()),  -- Sort the data in the ArrayList.
    al[1],                                 -- java.util.List does not support indexing but
can process data of the ARRAY type that supports indexing.
    Objects.toString(a),        -- Convert data from the ARRAY type to the STRING type.
    array(1,2,3).subList(1, 2)             -- Obtain a sublist.
from (select new ArrayList<Integer>(array(1,2,3)) as al, array(1,2,3) as a) t;
```

Aggregate data

If you use a UDT to aggregate data, you must first use the built-in function COLLECT_SET or COLLECT_LIST to aggregate the data to a list, and then call the scalar method of the UDT to calculate the aggregate value.

The following example shows how to calculate the median of BigInteger data. You cannot directly call the built-in function MEDIAN because the data is of the `java.math.BigInteger` type.

```
set odps.sql.session.java.imports=java.math.*;
@test_data := select * from values (1),(2),(3),(5) as t(value);
@a := select collect_list(new BigInteger(value)) values from @test_data;  -- Aggregate the
data to a list.
@b := select sort_array(values) as values, values.size() cnt from @a;  -- Sort the data.
@c := select if(cnt % 2 == 1, new BigDecimal(values[cnt div 2]), new BigDecimal(values[cnt
div 2 - 1].add(values[cnt div 2])).divide(new BigDecimal(2))) med from @b;
-- The final output.
select med.toString() from @c;
```

The `collect_list` function cannot be used to aggregate partial data because it can only aggregate all data in a specific group. It is less efficient than built-in aggregate functions of MaxCompute or user-defined aggregate functions (UDAFs). We recommend that you use built-in aggregate functions if possible. If all data in a group is aggregated, data skew may occur.

The UDT-based method produces a higher efficiency than a built-in aggregate function such as `WM_CONCAT` in aggregating all data in a group.

Implement table-valued functions

Table-valued functions allow you to specify multiple input rows and columns, and can generate multiple output rows and columns. To implement a table-valued function, perform the following steps:

- Specify multiple input rows or columns. For more information, see Aggregate data.
- Call the java.util.List or java.util.Map method to generate a data collection, and then call the explode function to split the collection into multiple rows.
- Call different getter methods to obtain data from different fields in a UDT. The obtained data is returned in multiple columns.

The following example shows how to split a JSON string and return the splitting result in multiple columns:

```
@a := select '[{"a":"1","b":"2"},{"a":"1","b":"2"}]' str; -- The sample data.
@b := select new com.google.gson.Gson().fromJson(str, java.util.List.class) l from @a; -- D
eserialize the JSON string.
@c := select cast(e as java.util.Map<Object,Object>) m from @b lateral view explode(l) t as
e;  -- Call the explode function to split the string.
@d := select m.get('a') as a, m.get('b') as b from @c; -- Return the splitting result in mu
ltiple columns.
select a.toString() a, b.toString() b from @d; -- The final output. Columns a and b in the
variable d are of the Object type.
```

Overload functions

In MaxCompute, user-defined functions (UDFs) overload functions by overloading the `evaluate` method. Generics are not supported in this mode. To define a function, you must write an `evaluate` method for each supported data type. Functions with some input types such as ARRAY cannot be overloaded in this mode. If the Resolve annotation is unavailable, Python UDFs or user-defined table-valued functions (UDTFs) determine input parameters based on the number of parameters and support variable-length parameters. However, if this mechanism is used, the compiler does not detect specific errors in static mode.

To address this issue, you can use a UDT to overload functions. UDTs support generics, class inheritance, and variable-length parameters, and offer flexible methods to define functions. Example:

```
public class UDTClass {
    // This function supports a numeric-type parameter and returns a value of the DOUBLE ty
pe. The parameter can be of the TINYINT, SMALLINT, INT, BIGINT, FLOAT, or DOUBLE type. You
can also specify a user-defined data type that is defined based on a numeric type in the pa
rameter.
    public static Double doubleValue(Number input) {
        return input.doubleValue();
    }
    // This function supports a numeric-type parameter and a parameter of any data type. Th
e return value is of the same type as the second parameter.
    public static <T extends Number, R> R nullOrValue(T a, R b) {
        return a.doubleValue() > 0 ? b : null;
    }
    // This function supports a parameter of the ARRAY or LIST type. The parameter value ca
n contain any type of elements. The function returns a value of the BIGINT type.
    public static Long length(java.util.List<? extends Object> input) {
        return input.size();
    }
    // If the code does not forcibly convert the data type, you must specify the default ma
p object java.util.Map<Object, Object> of the UDT. To call another map object, for example,
map<bigint,bigint>, you can use the following method:
    // 1. To define the function, use java.util.Map<? extends Object, ? extends Object>.
    // 2. If you call the function, implement a cast function to forcibly convert the type,
for example, UDTClass.mapSize(cast(mapObj as java.util.Map<Object, Object>)).
    public static Long mapSize(java.util.Map<Object, Object> input) {
        return input.size();
    }
}
```

In specific scenarios, a UDF can use `com.aliyun.odps.udf.ExecutionContext` that is called by the `setup` method to obtain the context. The UDT can call the `com.aliyun.odps.udt.UDTExecutionContext.get()` method to obtain the `ExecutionContext` object.

Reference embedded code in a UDT

Code-embedded UDFs allow you to place SQL scripts and third-party code lines in the same source code file. This simplifies the usage of UDTs and facilitates daily development and maintenance. For more information, see Code-embedded UDFs.

3.12. Script Mode SQL

The SQL engine of MaxCompute supports the script mode. If you use the script mode to compile an SQL script file, all statements in the file are compiled at the same time. You do not need to compile each statement. After the script file is compiled, it is submitted to MaxCompute, and an execution plan is generated. This way, the statements in the script file are scheduled in one queue and are executed once. This allows you to fully utilize the resources in MaxCompute.

> **Note** In script mode, you cannot estimate the expense of SQL statements by using the Cost SQL command described in Cost estimation. You can view your bill for the accurate expense in the Alibaba Cloud Management Console. For more information, see View billing details.

The script mode allows you to compile SQL statements similar to the way you compile SQL statements in a common programming language. You do not need to consider how to organize statements.

Syntax

```
--set
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=xxx;]
[...]
--ddl
create table table1 xxx;
[create table table2 xxx;]
[...]
--dml
@var1 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM table3
        [WHERE where_condition];
@var2 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM table4
        [WHERE where_condition];
@var3 := SELECT [ALL | DISTINCT] var1.select_expr, var2.select_expr, ...
        FROM @var1 join @var2 on ...;
INSERT OVERWRITE|INTO TABLE [PARTITION (partcol1=val1, partcol2=val2 ...)]
        SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM @var3;
[@var4 := SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var1
        UNION ALL | UNION
        SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var2;
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
        AS
        SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM @var4;]
[...]
```

Syntax description

- The script mode supports three types of statements: SET statements, DML statements, and some DDL statements. The DDL statements that are used to display results, such as DESC and SHOW, are not supported.
- A complete script file must be a list of SET, DDL, and DML statements in sequence. A script file can contain multiple statements of each type, but statements of the same type must be placed together. You must place these three types of statements in the preceding sequence.
- The at sign (`@`) declares a variable.
- A script file can contain a maximum of one statement that is used to display results, such as a SELECT statement. If a script file contains more than one such a statement, an error is returned. We recommend that you do not include SELECT statements that are used to display results in the script file.
- A script file can contain a maximum of one CREATE TABLE AS statement and must end with this statement. We recommend that you do not include the statements used to create tables and INSERT statements in the same script command.
- In script mode, if a statement in the script file fails to be executed, all the other statements in the

script file also fail.

- In script mode, a job can be generated to process data only after all the input data is prepared and inserted.
- In script mode, if a table is written and then read, the following error is returned:

```
INSERT OVERWRITE table src2 SELECT * FROM src WHERE key > 0;
@a := SELECT * FROM src2;
SELECT * FROM @a;
```

To prevent this error, you can modify your script file based on the following sample script:

```
@a := SELECT * FROM src WHERE key > 0;
INSERT OVERWRITE table src2 SELECT * FROM @a;
SELECT * FROM @a;
```

- You can use the script mode in the following scenarios:
  - A single statement with nested subqueries needs to be rewritten, or a complex script must be split into multiple statements.
  - Data from different data sources are prepared at different time points, and the time gap is large. For example, the data from one data source is prepared at 01:00, and that from another data source is prepared at 07:00. In this case, the table variables are not suitable for packaging the statements of different types in a script file.
  - You need to assign a constant value to a variable and execute the `SELECT * FROM Variable` statement to convert the variable into a scalar value. This scalar value is used with other columns for calculation. The constant value can also be stored in a table that has only one row. The following statements show an example. For more information about the syntax of the SELECT * FROM Variable statement, see Subqueries.

```
@a := SELECT 10; -- Assign the constant value 10 to @a. You can also execute the SELECT
col1 FROM t1 statement to store the constant value to table t1 that has only one row.
@b := SELECT key,value+(SELECT * FROM @a) FROM t2 WHERE key >10000; -- Use the value of
@a with value in table t2 for calculation.
SELECT * FROM @b;
```

Example

The following code shows how to run MaxCompute SQL in script mode:

```
CREATE TABLE IF NOT EXISTS dest(key STRING, value BIGINT) PARTITIONED BY (d STRING);
CREATE TABLE IF NOT EXISTS dest2(key STRING, value BIGINT) PARTITIONED  BY (d STRING);
@a := SELECT * FROM src WHERE value >0;
@b := SELECT * FROM src2 WHERE key is not null;
@c := SELECT * FROM src3 WHERE value is not null;
@d := SELECT a.key,b.value FROM @a LEFT OUTER JOIN @b ON a.key=b.key AND b.value>0;
@e := SELECT a.key,c.value FROM @a INNER JOIN @c ON a.key=c.key;
@f := SELECT * FROM @d UNION SELECT * FROM @e UNION SELECT * FROM @a;
INSERT OVERWRITE table dest PARTITION (d='20171111') SELECT * FROM @f;
@g := SELECT e.key,c.value FROM @e JOIN @c ON e.key=c.key;
INSERT OVERWRITE TABLE dest2 PARTITION (d='20171111') SELECT * FROM @g;
```

Required tools

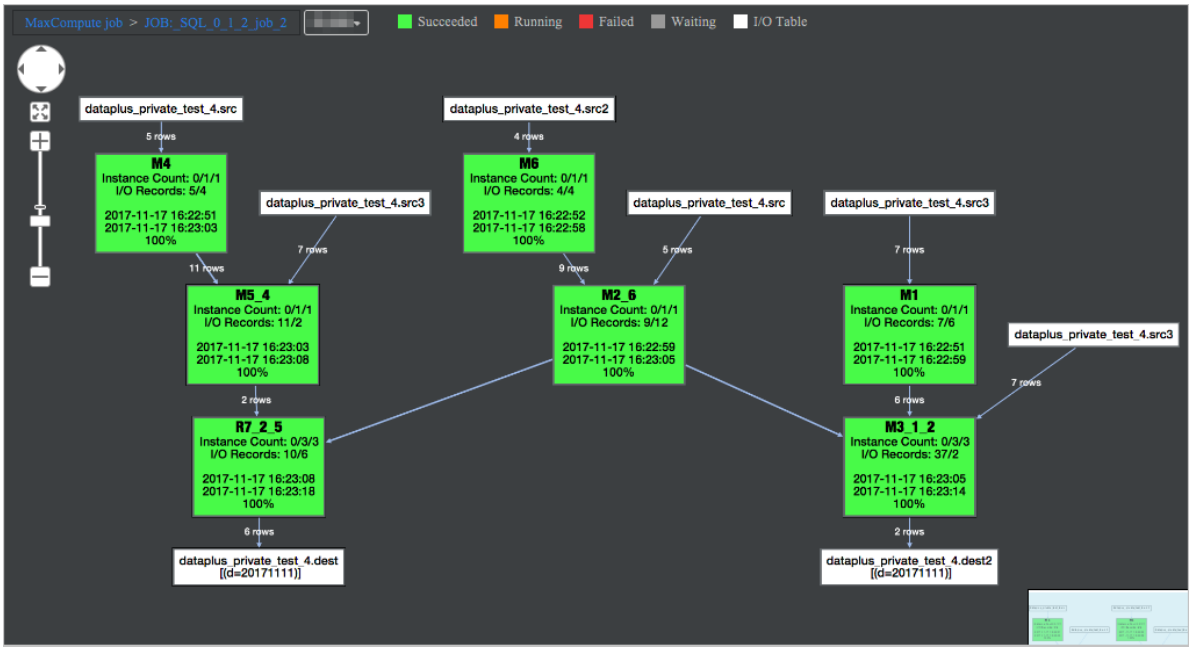
- MaxCompute Studio. For more information, see What is MaxCompute Studio?.
- MaxCompute client (odpscmd). For more information, see Install and configure the MaxCompute client.
- DataWorks. For more information, see Create an ODPS Script node.

Methods to use the script mode

- **Use MaxCompute Studio**

  Before you use MaxCompute Studio, make sure that MaxCompute Studio is installed, the project link is added, and a MaxCompute SQL script file is created. For more information, see Install IntelliJ IDEA, Manage project connections, and Create a MaxCompute script module.

  After you compile and submit the script file, you can view the graphic execution plan. Although the script file contains multiple statements, the graphic execution plan is shown in a directed acyclic graph (DAG).



- **Use the MaxCompute client**

  You must use the MaxCompute client of a version later than 0.27 to submit the script file. We recommend that you install the latest version of the MaxCompute client installation package. After you install the latest version of the package, use the `-s` option to submit the script file.

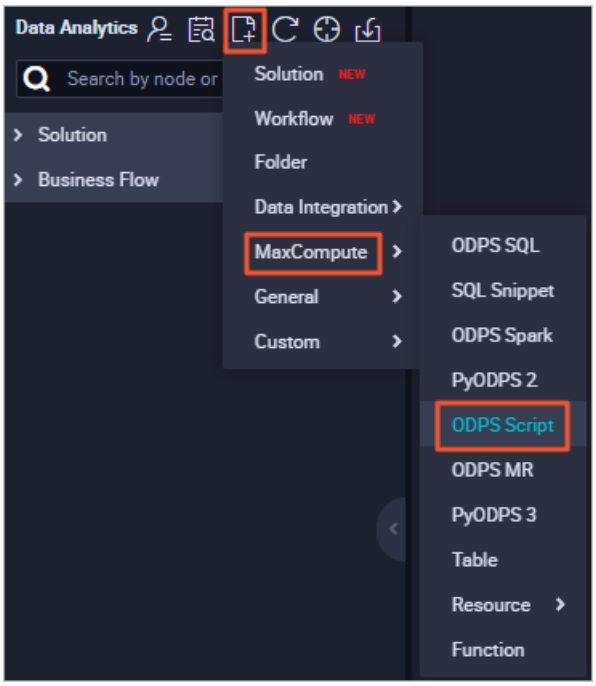
  Edit the source code file *myscript.sql* in script code and call odpscmd in the system command line to run the following command. For more information about how to run the MaxCompute client by using the system command line, see Run the MaxCompute client.

```
..\bin>odpscmd -s myscript.sql
```

  > **Note** Similar to the `-f` and `-e` options, the `-s` option is a command line option for the MaxCompute client. The -s option is not a command in an interactive environment. If the MaxCompute client (odpscmd) is used in an interactive environment, the script mode and table variables are not supported.

- **Use DataWorks**

In the DataWorks console, you can create an ODPS Script node, as shown in the following figure.



After the node is created, use the script mode to edit the script file and click the **Run** icon in the toolbar to submit the script file to MaxCompute. This way, you can use the Logview URL in the output to view the graphic execution plan and results.

3.13. Appendix

3.13.1. Operators

Operators are used to perform program code operations. This topic describes the operators that are supported by MaxCompute.

The following table describes the operator types that are supported by MaxCompute.

Operator type	Description
Relational operators	Used for comparison operations.
Arithmetic operators	Used for arithmetic operations.
Bitwise operators	Used for operations on binary digits.
Logical operators	Used to connect multiple conditions. In most cases, logical operators are used to connect expressions or values of the BOOLEAN type.

Relational operators

Operator	Description

Operator	Description
A=B	• If A or B is NULL, NULL is returned. • If A is equal to B, TRUE is returned. Otherwise, FALSE is returned.
A<>B	• If A or B is NULL, NULL is returned. • If A is not equal to B, TRUE is returned. Otherwise, FALSE is returned.
A<B	• If A or B is NULL, NULL is returned. • If A is less than B, TRUE is returned. Otherwise, FALSE is returned.
A<=B	• If A or B is NULL, NULL is returned. • If A is less than or equal to B, TRUE is returned. Otherwise, FALSE is returned.
A>B	• If A or B is NULL, NULL is returned. • If A is greater than B, TRUE is returned. Otherwise, FALSE is returned.
A>=B	• If A or B is NULL, NULL is returned. • If A is greater than or equal to B, TRUE is returned. Otherwise, FALSE is returned.
A IS NULL	If A is NULL, TRUE is returned. Otherwise, FALSE is returned.
A IS NOT NULL	If A is not NULL, TRUE is returned. Otherwise, FALSE is returned.

A LIKE B	If A or B is NULL, NULL is returned. If String A matches Pattern B, TRUE is returned. Otherwise, FALSE is returned.
	• The percent sign (%) matches an arbitrary number of characters.
	• The underscore (_) matches a single character.
	• To match percent signs (%) or underscores (_), you must escape percent signs (%) or underscores (_) with single quotation marks ('). After the percent signs (%) or underscores (_) are escaped, `'%'` or `'_'` is used for the match.
	`'aaa' like 'a__'= TRUE`
	`'aaa' like 'a%' = TRUE`
	`'aaa' like 'aab'= FALSE`
	`'a%b' like 'a\\%b'= TRUE`
	`'axb' like 'a\\%b'= FALSE`

A RLIKE B	If String A matches String Constant or Regular Expression B, TRUE is returned. Otherwise, FALSE is returned. If B is an empty string, an error is returned. If A or B is NULL, NULL is returned.

Operator	Description
A IN B	• If A is included in Set B, TRUE is returned. Otherwise, FALSE is returned. • If A is NULL, NULL is returned. • If B contains only one NULL element, the operator is A IN (NULL). In this case, NULL is returned. • B must be a constant set and contain at least one element. All the elements in B must be of the same data type. **Note** If Set B contains the NULL element and other elements, the data type of NULL is considered the same as the data types of other elements in Set B.
BETWEEN AND	The expression is `A [NOT] BETWEEN B AND C`. • If A, B, or C is NULL, NULL is returned. • If A is greater than or equal to B and less than or equal to C, TRUE is returned. Otherwise, FALSE is returned.
IS [NOT] DISTINCT FROM	The expression is `A IS [NOT] DISTINCT FROM B`. For more information, see IS DISTINCT FROM and IS NOT DISTINCT FROM.

Common use of relational operators in statements:

```
SELECT * FROM user WHERE user_id = '0001';
SELECT * FROM user WHERE user_name <> 'maggie';
SELECT * FROM user WHERE age > '50';
SELECT * FROM user WHERE birth_day >= '1980-01-01 00:00:00';
SELECT * FROM user WHERE is_female is null;
SELECT * FROM user WHERE is_female is not null;
SELECT * FROM user WHERE user_id in (0001,0010);
SELECT * FROM user WHERE user_name like 'M%';
```

Before you perform some relational operations, you must convert the data type. Otherwise, NULL may be returned. For more information about type conversions, see Type conversions.

Values of the DOUBLE type in MaxCompute are different in precision. Due to this reason, we recommend that you do not use an equal sign (=) for comparison between two values of the DOUBLE type. You can subtract a value of the DOUBLE type from another value of the DOUBLE type, and obtain the absolute value for comparison. If the absolute value is negligible, the two values of the DOUBLE type are considered equal. Example:

```
ABS(0.9999999999 - 1.0000000000) < 0.000000001
 -- 0.9999999999 and 1.0000000000 have a precision of 10 decimal digits, whereas 0.00000000
1 has a precision of 9 decimal digits.
 -- 0.9999999999 is considered equal to 1.0000000000.
```

Note

- ABS is a built-in function provided by MaxCompute. This function is used to obtain the absolute value of its input. For more information, see ABS.
- In most cases, a value of the DOUBLE type in MaxCompute can provide a precision of 14 decimal digits.
- If you compare a value of the STRING type with a value of the BIGINT type, the data types of the values are automatically converted into DOUBLE. The precision loss may occur during the comparison. To address this issue, you can use **CAST STRING AS BIGINT** to convert the STRING type into BIGINT.

Arithmetic operators

Operator	Description
A+B	If A or B is NULL, NULL is returned. Otherwise, the result of A + B is returned.
A-B	If A or B is NULL, NULL is returned. Otherwise, the result of A - B is returned.
A*B	If A or B is NULL, NULL is returned. Otherwise, the result of A × B is returned.
A/B	If A or B is NULL, NULL is returned. Otherwise, the result of A/B is returned. Note If A and B are of the BIGINT type, the return value is of the DOUBLE type.
A%B	If A or B is NULL, NULL is returned. Otherwise, the remainder of A/B is returned.
+A	A is returned.
-A	If A is NULL, NULL is returned. Otherwise, -A is returned.
A DIV B	If A or B is NULL, NULL is returned. Otherwise, the result of A DIV B is returned.

Common use of arithmetic operators in statements:

```
SELECT age+10, age-10, age%10, -age, age*age, age/10, age div 10 FROM user;
```

Note

- You can use only the values of the STRING, BIGINT, or DOUBLE type to perform arithmetic operations. You cannot use the values of the DATATIME or BOOLEAN type to perform arithmetic operations.
- Values of the STRING type are implicitly converted into the DOUBLE type before arithmetic operations.
- If you use values of the BIGINT and DOUBLE types to perform arithmetic operations, the value of the BIGINT type is implicitly converted into the DOUBLE type before the operations. The return value is of the DOUBLE type.
- If A and B are of the BIGINT type, the return value is of the DOUBLE type after you perform the A/B operation. For other arithmetic operations, the return value is of the BIGINT type.

Bitwise operators

Operator	Description
A&B	The bitwise AND result of A and B is returned. For example, the result of 1&2 is 0, the result of 1&3 is 1, and the bitwise AND result of NULL and any value is NULL. A and B must be of the BIGINT type.
A|B	The bitwise OR result of A and B is returned. For example, the result of 1|2 is 3, the result of 1|3 is 3, and the bitwise OR result of NULL and any value is NULL. A and B must be of the BIGINT type.
A||B	This operator is used to join strings. For example, `a||b||c` is equivalent to `CONCAT(a, b, c)`.

Note Bitwise operators do not support implicit type conversions. You can use only values of the BIGINT type in bitwise operations.

Logical operators

Operator	Description
A and B	TRUE and TRUE=TRUE
	TRUE and FALSE=FALSE
	FALSE and TRUE=FALSE
	FALSE and FALSE=FALSE
	FALSE and NULL=FALSE
	NULL and FALSE=FALSE
	TRUE and NULL=NULL

Operator	Description
	NULL and TRUE=NULL
	NULL and NULL=NULL
A or B	TRUE or TRUE=TRUE
	TRUE or FALSE=TRUE
	FALSE or TRUE=TRUE
	FALSE or FALSE=FALSE
	FALSE or NULL=NULL
	NULL or FALSE=NULL
	TRUE or NULL=TRUE
	NULL or TRUE=TRUE
	NULL or NULL=NULL
NOT A	If A is NULL, NULL is returned.
	If A is TRUE, FALSE is returned.
	If A is FALSE, TRUE is returned.

Note Logical operators do not support implicit type conversions. You can use only the values of the BOOLEAN type in logical operations.

Operator precedence

The following table lists the precedence of operators in descending order. If operators have the same precedence, the operations are performed from left to right by default.

Operator	Precedence
IS (NOT) NULL	1
^	2
*, /, %, DIV	3
+, -	4
||	5
&	6

Operator	Precedence
|	7
(NOT) LIKE, (NOT) RLIKE, =, ==, IS (NOT) DISTINCT FROM, <>, !=, <=, <, >=, >	8
(NOT) IN, (NOT) BETWEEN AND	9
NOT	10
AND	11
OR	12

If you want to preferentially process some operations, you must enclose the operations in parentheses (). Example:

```
a=1 and b=1 or c=1        -- a and b are calculated before c.
a=1 and (b=1 or c=1)      -- b and c are calculated before a.
```

3.13.2. Escape character

In MaxCompute SQL, the backslash character (`\`) is an escape character that invokes an alternative representation on the following characters in a character sequence or indicates a literal interpretation of these characters.

If the backslash character (\) in a string constant is followed by three valid octal digits in the range from 001 to 177, the combination of the backslash and the three digits is interpreted into an ASCII character.

Escape sequence

MaxCompute SQL uses the backlash character (\) as an escape character for the escape sequences in the following table.

Escape sequence	Represented character
\b	backspace
\t	tab
\n	newline
\r	carriage-return
\'	Single quotation mark
\"	Double quotation mark
\ \	Backslash
\;	Semicolon

Escape sequence	Represented character
\Z	control-Z
\0 or \00	Terminator

Example

- Use an escape sequence to represent a string constant.

  In MaxCompute SQL, you can represent a string constant by using single quotation marks or double quotation marks. A string constant that contains a double quotation mark can be enclosed in a pair of single quotation marks. A string constant that contains a single quotation mark can be enclosed in a pair of double quotation marks. You can also use the backslash (\) as an escape character to represent a string constant. The following example shows two ways to represent a string constant:

  ```
  "I'm a happy manong."
  'I\'m a happy manong.'
  ```

- Use an escape sequence to represent a special character. For example, run the following command:

  ```
  select length('a\tb');
  ```

  The preceding command returns 3, which indicates that the string contains three characters and that `\t` is treated as one character.

  ```
  select 'a\ab',length('a\ab');
  ```

  The preceding command returns aab, 3, which indicates that `\a` is treated as letter a.

3.13.3. LIKE usage

This topic describes how to use the LIKE operator for character matching.

In LIKE character matching, `%` is used to match a string that consists of zero or more characters, and `_` is used to match a single character.

To match the `%` or `_` character itself, you must escape the character. Use `\\%` or `\\_` to match the `%` or `_` character.

```
'abcd' like 'ab%' -- true
'abcd' like 'ab_' -- false
'ab_cde' like 'ab\\_c%'; -- true
```

Notice MaxCompute SQL statements support only UTF-8 character sets. If data is encoded in another format, the calculation result may be incorrect.

3.13.4. Regular expressions

Regular expressions in MaxCompute SQL use the Perl Compatible Regular Expressions (PCRE) standards to match strings by character.

RLIKE

To perform queries, you can use the **RLIKE** operator together with regular expressions.

Metacharacter	Description
^	Matches the beginning of a string.
$	Matches the end of a string.
.	Matches any single character.
*	Matches zero or more instances of the preceding character or character pattern.
+	Matches one or more instances of the preceding character or character pattern.
?	Matches zero or one instance of the preceding character or character pattern.
?	Matches a modifier. If this character follows one of other characters (*, +, ?, {n}, {n,}, or {n,m}), the match pattern is non-greedy. In the non-greedy algorithm, a character string matches as few characters as possible. In the greedy algorithm, a character string matches as many characters as possible. By default, the greedy algorithm is used.
A|B	Matches A or B.
(abc)*	Matches zero or more instances of the abc sequence.
{n} or {m,n}	The number of matches.
[ab]	Matches any character in the brackets. The character can be a or b. Fuzzy match is used.
[a-d]	Matches one of the following characters: a, b, c, and d.
[^ab]	Matches any character except those in the square brackets.
[::]	For more information, see the following table.
\	The escape character.
\n	n is a digit from 1 to 9 and is backward referenced.
\d	Digits.
\D	Non-digit characters.

Character group

Category	Character group	Description	Valid value
	[[:alnum:]]	Letters and digits	[a-zA-Z0-9]
	[[:alpha:]]	Letters	[a-zA-Z]

Category	Character group	Description	Valid value
POSIX	[[:ascii:]]	ASCII characters	[\x00-\x7F]
	[[:blank:]]	Spaces and tab characters	[\t]
	[[:cntrl:]]	Control characters	[\x00-\x1F\x7F]
	[[:digit:]]	Digits	[0-9]
	[[:graph:]]	Characters other than whitespace characters	[\x21-\x7E]
	[[:lower:]]	Lowercase letters	[a-z]
	[[:print:]]	[:graph:] and whitespace characters	[\x20-\x7E]
	[[:punct:]]	Punctuations	[][!"#$%&'()*+,./:;<=>? @\^_`{|}~-]
	[[:space:]]	Whitespace characters	[\t\r\n\v\f]
	[[:upper:]]	Uppercase letters	[A-Z]
	[[:xdigit:]]	Hexadecimal characters	[A-Fa-f0-9]
Chinese	N/A	Double-byte characters such as Chinese characters.	[^\\x{00}-\\x{ff}]
	N/A	Chinese characters	[\\x{4e00}-\\x{9fa5}]
	N/A	Chinese punctuations	Chinese punctuations do not have unified Unicode range. You can search for **the Unicode of a Chinese punctuation** in the search engine and use operators to match Chinese punctuations.

Escape characters

The system uses a backslash (`\`) as the escape character. Backslashes (`\`) in regular expressions must also be escaped. For example, a regular expression is used to match the `a+b` string. The plus sign (`+`) is a special character in the expression and must be escaped. In the regular expression engine, the string is expressed as `a\\+b` . The system must perform another escape in the expression. Therefore, the expression that can match the string is `a\\\+b` .

The following example shows how to use escape characters in a regular expression. The test_dual table is used.

```
select 'a+b' rlike 'a\\\+b';
+------+
| _c1  |
+------+
| true |
+------+
```

The `\` character is a special character in the regular expression engine and must be expressed as `\\` in the engine. Then, the system performs another escape in the expression. As a result, the `\` character is expressed as `\\\\`.

```
select 'a\\b', 'a\\b' rlike 'a\\\b';
+-----+------+
| _c0 | _c1  |
+-----+------+
| a\b | false|
+-----+------+
select 'a\\b', 'a\\b' rlike 'a\\\\b';
+-----+------+
| _c0 | _c1  |
+-----+------+
| a\b | true |
+-----+------+
```

> **Note** If a MaxCompute SQL statement contains `a\\b`, the system returns `a\b` because MaxCompute escapes the expression.

If a string contains tab characters, the system stores `\t` as one character when the system reads this expression. In this case, \t is a common character in regular expressions.

```
select 'a\tb', 'a\tb' rlike 'a\tb';
+---------+------+
| _c0     | _c1  |
+---------+------+
| a     b | true |
+---------+------+
```

3.13.5. Reserved words and keywords

This topic introduces all reserved words and keywords in MaxCompute SQL.

> **Notice**
>
> - When you name a table, a column, or a partition, do not use reserved words or keywords. Otherwise, an error may occur.
> - Reserved words are not case-sensitive.
> - If you use keywords to name a table, a column, or a partition, you must escape the keywords by using double quotation marks (`"`). Otherwise, an error is returned.

```
    %    &    &&    (    )    *    +
    -    .    /    ;    <    <=    <>
    =    >    >=    ?    ADD    AFTER    ALL
    ALTER    ANALYZE    AND    ARCHIVE    ARRAY    AS    ASC
    BEFORE    BETWEEN    BIGINT    BINARY    BLOB    BOOLEAN    BOTH    DECIMAL
    BUCKET    BUCKETS    BY    CASCADE    CASE    CAST    CFILE
    CHANGE    CLUSTER    CLUSTERED    CLUSTERSTATUS    COLLECTION    COLUMN    COLUMNS
    COMMENT    COMPUTE    CONCATENATE    CONTINUE    CREATE    CROSS    CURRENT
    CURSOR    DATA    DATABASE    DATABASES    DATE    DATETIME    DBPROPERTIES
    DEFERRED    DELETE    DELIMITED    DESC    DESCRIBE    DIRECTORY    DISABLE
    DISTINCT    DISTRIBUTE    DOUBLE    DROP    ELSE    ENABLE    END    EXCEPT
    ESCAPED    EXCLUSIVE    EXISTS    EXPLAIN    EXPORT    EXTENDED    EXTERNAL
    FALSE    FETCH    FIELDS    FILEFORMAT    FIRST    FLOAT    FOLLOWING
    FORMAT    FORMATTED    FROM    FULL    FUNCTION    FUNCTIONS    GRANT
    GROUP    HAVING    HOLD_DDLTIME    IDXPROPERTIES    IF    IMPORT    IN
    INDEX    INDEXES    INPATH    INPUTDRIVER    INPUTFORMAT    INSERT    INT
    INTERSECT    INTO    IS    ITEMS    JOIN    KEYS    LATERAL
    LEFT    LIFECYCLE    LIKE    LIMIT    LINES    LOAD    LOCAL
    LOCATION    LOCK    LOCKS    LONG    MAP    MAPJOIN    MATERIALIZED
    MINUS    MSCK    NOT    NO_DROP    NULL    OF    OFFLINE    OFFSET
    ON    OPTION    OR    ORDER    OUT    OUTER    OUTPUTDRIVER
    OUTPUTFORMAT    OVER    OVERWRITE    PARTITION    PARTITIONED    PARTITIONPROPERTIES
PARTITIONS
    PERCENT    PLUS    PRECEDING    PRESERVE    PROCEDURE    PURGE    RANGE
    RCFILE    READ    READONLY    READS    REBUILD    RECORDREADER    RECORDWRITER
    REDUCE    REGEXP    RENAME    REPAIR    REPLACE    RESTRICT    REVOKE
    RIGHT    RLIKE    ROW    ROWS    SCHEMA    SCHEMAS    SELECT
    SEMI    SEQUENCEFILE    SERDE    SERDEPROPERTIES    SET    SHARED    SHOW
    SHOW_DATABASE    SMALLINT    SORT    SORTED    SSL    STATISTICS    STATUS    STORED
    STREAMTABLE    STRING    STRUCT    TABLE    TABLES    TABLESAMPLE    TBLPROPERTIES
    TEMPORARY    TERMINATED    TEXTFILE    THEN    TIMESTAMP    TINYINT    TO
    TOUCH    TRANSFORM    TRIGGER    TRUE    TYPE    UNARCHIVE    UNBOUNDED    UNDO
    UNION    UNIONTYPE    UNIQUEJOIN    UNLOCK    UNSIGNED    UPDATE    USE
    USING    UTC    UTC_TMESTAMP    VIEW    WHEN    WHERE    WHILE    DIV
```

3.13.6. Data type mappings

This topic describes the data type mappings between MaxCompute and the following components: Hive, Oracle, and MySQL.

The following table lists the data type mappings.

MaxCompute data type	Hive data type	Oracle data type	MySQL data type

MaxCompute data type	Hive data type	Oracle data type	MySQL data type
BOOLEAN	BOOLEAN	None **Note** CHAR(1), INTEGER, or NUMBER(1) is used instead. The value 1 indicates true, and the value 0 indicates false.	None **Note** TINYINT(1) is used instead.
TINYINT	TINYINT	NUMBER(3,0)	TINYINT
SMALLINT	SMALLINT	NUMBER(5,0)	SMALLINT
INT	INT	NUMBER(7,0)	MEDIUMINT
INT	INT	NUMBER(10,0)	INT
BIGINT	BIGINT	NUMBER(20,0)	BIGINT
FLOAT	FLOAT	BINARY_FLOAT **Note** This data type is supported in Oracle Database 10g and later.	FLOAT
DOUBLE	DOUBLE	BINARY_DOUBLE **Note** This data type is supported in Oracle Database 10g and later.	DOUBLE
DECIMAL	DECIMAL	NUMBER(P,S)	• DECIMAL • NUMERIC
STRING	STRING	• VARCHAR • VARCHAR2 • CHAR • NCHAR • NVARCHAR3	• VARCHAR • CHAR

MaxCompute data type	Hive data type	Oracle data type	MySQL data type
VARCHAR	VARCHAR	• VARCHAR • VARCHAR2 • CHAR • NCHAR • NVARCHAR3	VARCHAR
STRING	CHAR	CHAR	CHAR
BINARY	BINARY	RAW	• BINARY • VARBINARY
TIMESTAMP	TIMESTAMP	TIMESTAMP(N)	TIMESTAMP
DATETIME	DATE	DATE	DATETIME
ARRAY	ARRAY	Not supported	Not supported
MAP	`MAP<key,value>`	Not supported	Not supported
STRUCT	STRUCT	Not supported	Not supported
Not supported	UNION	Not supported	Not supported

3.13.7. Type conversions

MaxCompute SQL allows you to convert data types. Two conversion methods are supported: explicit conversion and implicit conversion.

Explicit conversion

An explicit conversion uses the CAST function to convert the data type of a value. The following table describes the rules of explicit conversions supported by MaxCompute SQL. For more information about the CAST function, see CAST.

From/To	BIGINT	DOUBLE	STRING	DATETIME	BOOLEAN	DECIMAL	FLOAT
BIGINT	N/A	Y	Y	N	Y	Y	Y
DOUBLE	Y	N/A	Y	N	Y	Y	Y
STRING	Y	Y	N/A	Y	Y	Y	Y
DATETIME	N	N	Y	N/A	N	N	N
BOOLEAN	Y	Y	Y	N	N/A	Y	Y

From/To	BIGINT	DOUBLE	STRING	DATETIME	BOOLEAN	DECIMAL	FLOAT
DECIMAL	Y	Y	Y	N	Y	N/A	Y
FLOAT	Y	Y	Y	N	Y	Y	N/A

Y indicates that the conversion is supported. N indicates that the conversion is not supported. N/A indicates that the conversion is not required. An error is returned if an unsupported explicit conversion is performed.

Examples

```
SELECT CAST(user_id AS DOUBLE) AS new_id;
SELECT CAST('2015-10-01 00:00:00' AS DATETIME) AS new_date;
SELECT CAST(ARRAY(1,2,3) AS ARRAY<STRING>);
SELECT CONCAT_WS(',', CAST(ARRAY(1, 2) AS ARRAY<STRING>));
```

Instructions and limits

- If a value of the DOUBLE type is converted into that of the BIGINT type, digits after the decimal point are removed. For example, you can execute `CAST(1.6 AS BIGINT) = 1` to convert 1.6 of the DOUBLE Type to 1 of the BIGINT type.
- If a value of the STRING type that meets the format of the DOUBLE type is converted into the BIGINT type, the STRING type is first converted into the DOUBLE type and then to the BIGINT type. During the conversion, the digits after the decimal point are removed. For example, you can execute `CAST("1.6" AS BIGINT) = 1` to convert 1.6 of the STRING type to 1 of the BIGINT type.
- If a value of the STRING type that meets the format of the BIGINT type is converted into the DOUBLE type, one digit is retained after the decimal point. For example, you can execute `CAST("1" AS DOUBLE) = 1.0` to convert 1 of the STRING type to 1.0 of the DOUBLE type.
- The default format `yyyy-mm-dd hh:mi:ss` is used during the conversion that involves the DATETIME type.
- Some data types cannot be explicitly converted, but can be converted by using SQL built-in functions. For example, you can use the TO_CHAR function to convert the BOOLEAN type into the STRING type. For more information, see TO_CHAR. You can also use the TO_DATE function to convert the STRING type into the DATETIME type. For more information, see TO_DATE.
- If the value of the DECIMAL type exceeds its value range, the `CAST STRING TO DECIMAL` operation may cause errors, such as overflow of the most significant bit or removal of the least significant bit.
- A conversion from the DECIMAL type to the DOUBLE or FLOAT type results in a loss of precision. In scenarios where high precision is required, for example, when you calculate the bill amount or premium rate, we recommend that you retain the DECIMAL type.
- MaxCompute allows you to convert complex data types. Implicit conversions between complex data types can be implemented only when their subtypes support implicit conversions. Explicit conversions between complex data types can be implemented only when their subtypes support explicit conversions. If you convert the STRUCT type, field names can be inconsistent, but the number of fields must be consistent and these fields must support implicit or explicit conversions. Examples:
  - `ARRAY<BIGINT>` can be implicitly or explicitly converted to `ARRAY<STRING>`.
  - `ARRAY<BIGINT>` can be explicitly converted to `ARRAY<INT>`. Implicit conversions are not supported.

- `ARRAY<BIGINT>` cannot be implicitly or explicitly converted to `ARRAY<DATETIME>`.
- `STRUCT<a:BIGINT,b:INT>` can be implicitly converted to `STRUCT<col1:STRING,col2:BIGINT>`. Implicit or explicit conversions to `STRUCT<a:STRING>` are not supported.

Implicit conversion and its application scope

An implicit conversion allows MaxCompute to automatically convert data types based on the context and predefined rules. The following table describes the rules of implicit conversions supported by MaxCompute.

From/To	BOOLEAN	TINYINT	SMALLINT	INT	BIGINT	FLOAT
BOOLEAN	Y	N	N	N	N	N
TINYINT	N	Y	Y	Y	Y	Y
SMALLINT	N	N	Y	Y	Y	Y
INT	N	N	Y	Y	Y	Y
BIGINT	N	N	N	N	Y	Y
FLOAT	N	N	N	N	Y	Y

From/To	DOUBLE	DECIMAL	STRING	VARCHAR	TIMESTAMP	BINARY
DOUBLE	Y	Y	Y	Y	N	N
DECIMAL	N	Y	Y	Y	N	N
STRING	Y	Y	Y	Y	N	N
VARCHAR	Y	Y	N	N	N/A	N/A
TIMESTAMP	N	N	Y	Y	Y	N
BINARY	N	N	N	N	N	Y

Y indicates that the conversion is supported. N indicates that the conversion is not supported. N/A indicates that the conversion is not required. An error is returned if an unsupported implicit conversion is performed or if a conversion fails.

> **Note**
>
> - MaxCompute V2.0 introduces the methods to define constants of the DECIMAL and DATETIME types. For example, 100BD indicates value 100 of the DECIMAL type. `2017-11-11 00:00:00` indicates a constant of the DATETIME type. Constants can be directly defined in the VALUES clauses and tables.
> - If an implicit conversion is used, MaxCompute automatically converts data types based on context. If the types do not match, you can use the CAST function to explicitly convert data types.
> - Implicit conversion rules apply to specific scopes. In specific scenarios, only part of the rules take effect.

Example

```
SELECT user_id+age+'12345', CONCAT(user_name,user_id,age) FROM user;
```

Implicit conversions with different operators:

- **Implicit conversions with relational operators**

  Relational operators include `=, <>, <, ≤, >, ≥, IS NULL, IS NOT NULL, LIKE, RLIKE, and IN`. The rules that use `LIKE, RLIKE, and IN` are different from those that use other relational operators. The rules described in this section do not apply to the three operators.

  The following table describes the rules of implicit conversions when data of different types is used for relational operations.

From/To	BIGINT	DOUBLE	STRING	DATETIME	BOOLEAN	DECIMAL
BIGINT	N/A	DOUBLE	DOUBLE	N	N	DECIMAL
DOUBLE	DOUBLE	N/A	DOUBLE	N	N	DECIMAL
STRING	DOUBLE	DOUBLE	N/A	DATETIME	N	DECIMAL
DATETIME	N	N	DATETIME	N/A	N	N
BOOLEAN	N	N	N	N	N/A	N
DECIMAL	DECIMAL	DECIMAL	DECIMAL	N	N	N/A

  > **Note**
  >
  > - If two values you want to compare do not support implicit conversions, the relational operation cannot be completed and an error is returned.
  > - For more information about relational operators, see Operators.

- **Implicit conversions with special relational operators**

  Special relational operators are `LIKE, RLIKE, and IN`.

- Syntax of LIKE and RLIKE

```
source LIKE pattern;
source RLIKE pattern;
```

> **Note**
> - The source and pattern parameters of LIKE and RLIKE must be of the STRING type.
> - Other types can neither be involved in the operation nor be implicitly converted into the STRING type.

- Syntax of IN

```
key IN (value1, value2, ...)
```

> **Note**
> - The data types in the value list next to IN must be consistent.
> - If the data types of values include BIGINT, DOUBLE, and STRING, convert the values of the BIGINT and STRING types into the DOUBLE type. If the data types of values include DATETIME and STRING, convert the values of the STRING type into the DATETIME type. Conversions between other data types are not allowed.

- **Implicit conversions with arithmetic operators**

  Arithmetic operators include: `+, -, *, /, and %`. The following rules apply to implicit conversions with these operators.

  - Only the values of the STRING, BIGINT, DOUBLE, and DECIMAL types can be used for arithmetic operations.
  - Values of the STRING type are implicitly converted into the DOUBLE type before the arithmetic operations.
  - If values of the BIGINT and DOUBLE types are used for the arithmetic operations, the value of the BIGINT type is implicitly converted into the DOUBLE type.
  - The values of the DATETIME and BOOLEAN types cannot be used for arithmetic operations.

- **Implicit conversions with logical operators**

  Logical operators include `AND, OR, and NOT`. The following rules apply to implicit conversions with these operators:

  - Only the values of the BOOLEAN type can be used for logical operations.
  - Value of other types cannot be used for logical operations or implicitly converted.

Implicit conversions with built-in functions

MaxCompute SQL provides a variety of built-in functions. These functions can be used to calculate one or more columns of a specific row and provide data of a specific type. The following rules apply to implicit conversions with these functions:

- If you call a built-in function and the data type of an input parameter is different from that defined in the function, the data type of the input parameter is converted into the function-defined data type.

- The parameters of each built-in function in MaxCompute SQL have different requirements for implicit conversions. For more information, see Built-in functions.

Implicit conversions with CASE WHEN

For more information about CASE WHEN, see CASE WHEN expression. The following rules apply to implicit conversions with CASE WHEN:

- If the return values are of the BIGINT and DOUBLE types, all the values are converted into the DOUBLE type.
- If the return values include those of the STRING type, all the values are converted into the STRING type. If a conversion, such as a conversion from BOOLEAN to STRING fails, an error is returned.
- Conversions between other data types are not allowed.

Conversions between the STRING and DATETIME types

MaxCompute supports conversions between the STRING and DATETIME types. The format `yyyy-mm-dd hh:mi:ss` is used during the conversions.

Time unit	String (not case-sensitive)	Valid value
Year	yyyy	0001-9999
Month	mm	01-12
Day	dd	01-28|29|30|31
Hour	hh	00-23
Minute	mi	00-59
Second	ss	00-59

Note

- If the first digit of the value range of each time unit is 0, 0 cannot be omitted. For example, `2014-1-9 12:12:12` is an invalid DATETIME format and it cannot be converted from the STRING type to the DATETIME type. It must be written as `2014-01-09 12:12:12`.
- Only the STRING type that meets the preceding format requirements can be converted into the DATETIME type. For example, `CAST("2013-12-31 02:34:34" AS DATETIME)` converts `2013-12-31 02:34:34` of the STRING type to the DATETIME type. Similarly, if the DATETIME type is converted into the STRING type, the `yyyy-mm-dd hh:mi:ss` format is automatically used after the conversion.

MaxCompute provides the TO_DATE function to convert the STRING type that does not meet the format of the DATETIME type to the DATETIME type. For more information, see TO_DATE.

3.13.8. Dynamic parameters

This topic describes the dynamic parameters that are supported by MaxCompute user-defined aggregate functions (UDAFs) and user-defined table-valued functions (UDTFs) when the Resolve annotation is used.

Extension of Resolve annotation syntax

Both MaxCompute UDAFs and UDTFs use the Resolve annotation to generate the signature of a function. However, this method cannot be overloaded, because the input and output parameters are fixed.

```
@com.aliyun.odps.udf.annotation.Resolve("BIGINT->DOUBLE")
public class UDTFClass extends UDTF {
    ...
}
```

> **Note** This example defines a UDTF that accepts parameters of the BIGINT type and returns values of the DOUBLE type.

The syntax of the Resolve annotation in MaxCompute is extended.

- You can use an asterisk (`*`) in a parameter list to indicate that an input parameter can be of any length and type. For example, `@Resolve('double,*->String')` indicates a parameter list in which the first parameter is of the DOUBLE type, followed by parameters of any length and type. In this case, you must compile code to calculate the number and types of input parameters, and manage them based on the `printf` function in the C programming language.

  > **Note** Asterisks (*) in return values indicate different meanings.

- You can use `any` in a parameter list to indicate that parameters of all types are valid. For example, `@Resolve('double,any->string')` indicates a parameter list in which the first parameter is of the DOUBLE type, followed by parameters of any type.

  > **Note** `any` cannot be used in return values or subtypes of complex data types, such as ARRAY.

- Asterisks can be used in return values of UDTFs to indicate that any number of values of the STRING type can be returned. The number of return values is based on the number of aliases that are configured when a function is called. For example, the call method of `@Resolve("ANY,ANY->DOUBLE,*")` is `UDTF(x, y) as (a, b, c)`. In this example, three aliases a, b, and c are configured for `as`. The editor identifies that a is of the DOUBLE type and b and c are of the STRING type. The type of values in the first column returned in the Resolve annotation is given. Three values are returned. Therefore, the `forward` method called by a UDTF must `forward` an array of three elements. Otherwise, an error is returned.

  > **Note** However, the error is not returned during compilation. Therefore, the UDTF caller must follow the rule defined by the UDTF to set the number of aliases in SQL. The number of return values of an aggregate function is fixed to 1. Therefore, this rule has no effect on UDAFs.

UDTF examples

```
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.annotation.Resolve;
import org.json.JSONException;
import org.json.JSONObject;
@Resolve("STRING,*->STRING,*")
public class JsonTuple extends UDTF {
  private Object[] result = null;
  @Override
  public void process(Object[] input) throws UDFException {
    if (result == null) {
      result = new Object[input.length];
    }
    try {
      JSONObject obj = new JSONObject((String)input[0]);
      for (int i = 1; i < input.length; i++) {
        // The variable-length part of a return value must be of the STRING type.
        result[i] = String.valueOf(obj.get((String)(input[i])));
      }
      result[0] = null;
    } catch (JSONException ex) {
      for (int i = 1; i < result.length; i++) {
        result[i] = null;
      }
      result[0] = ex.getMessage();
    }
    forward(result);
  }
}
```

In this example, the number of return values is based on the number of input parameters. The first output parameter indicates a JSON file, and the other output parameters are the keys parsed from the JSON file. The first return value is an error message during JSON file parsing. If no error occurs, the content parsed from the JSON file is provided based on the sequence of input keys. Example:

```
-- Set the number of output aliases based on that of input parameters.
SELECT my_json_tuple(json, 'a', 'b') as exceptions, a, b FROM jsons;
-- The variable-length part can have no columns.
SELECT my_json_tuple(json) as exceptions, a, b FROM jsons;
-- An error occurs when the following SQL statement is executed because the number of alias
es does not match the actual number.
-- This error is not returned during compilation.
SELECT my_json_tuple(json, 'a', 'b') as exceptions, a, b, c FROM jsons;
```

If the extensions described in this topic fail to meet your business requirements, we recommend that you use UDTs to implement the features of aggregate functions and UDTFs. For more information, see Overview.

3.14. MaxCompute SQLML

3.14.1. Overview

SQLML is an SQL portal provided by MaxCompute for you to use the capabilities of Machine Learning Platform for AI (PAI). At the underlying layer, MaxCompute SQLML relies on PAI to create, predict, and evaluate models. This topic describes SQLML features. It also describes the machine learning models, model prediction functions, and model evaluation functions supported by SQLML.

Description

MaxCompute SQLML depends on Alibaba Cloud services: MaxCompute and PAI. You can develop MaxCompute SQLML jobs in DataWorks, learn MaxCompute data by using PAI, and use machine learning models to make predictions. Then, use these results to guide your business planning.

- MaxCompute: provides SQLML, an SQL portal for you to use PAI.
- Client: SQL operation platform. You can choose DataWorks (recommended), MaxCompute SDK (Java SDK or Python SDK), MaxCompute odpscmd, or MaxCompute Studio.
- PAI: provides machine learning models.

MaxCompute SQLML helps data developers, analysts, and data scientists use SQL to create, train, and apply machine learning models. It also helps SQL practitioners use their SQL skills to implement PAI capabilities without the need to migrate data.

Usage notes

Procedure on how to use MaxCompute SQLML:

1. Activate MaxCompute, DataWorks Basic, and Pay-As-You-Go (PAI Studio, DSW, and EAS).
2. Prepare a dataset.

   The dataset is used to train and predict models.
3. Create or configure a DataWorks workspace. Set Compute Engines to **MaxCompute** and Machine Learning Services to **PAI Studio** for the workspace.
4. Use DataWorks to create a table and import the data in the dataset into the table.
5. Process the imported data based on the requirements of the specific model and create training datasets and test datasets. Training datasets are used to train models. Test datasets are used to predict models.
6. Create a machine learning model and make predictions by using the model prediction functions provided by MaxCompute.
7. Evaluate the accuracy of the prediction results by using the model evaluation functions provided by MaxCompute.

For usage examples, see Quick start.

Supported machine learning models

MaxCompute SQLML supports the following machine learning models:

- Logistic regression for binary classification: The model name is logisticregression_binary. For more information, see Linear SVM.
- Logistic regression for multiclass classification: The model name is logisticregression_multi. For more information, see PS-SMART Regression.
- Linear regression: The model name is linearregression. For more information, see GBDT Regression.

Supported model prediction functions

MaxCompute SQLML supports the `ml_predict` model prediction function. Syntax:

```
ml_predict(model <model_name>, table <data_source>[, map<string, string> <parameters>])
```

- model_name: required. This parameter specifies the name of the model that you want to create.
- data_source: required. This parameter specifies the data source used for the prediction, which can be a table or a SELECT statement.
- parameters: optional. This parameter specifies the parameters used for the prediction. The parameters are the same as those in PAI. For more information about the parameters, see Linear SVM, PS-SMART Regression, or GBDT Regression.

Supported model evaluation functions

MaxCompute SQLML supports the following model evaluation functions to evaluate the accuracy of the prediction results:

- Binary classification evaluation: implemented by using the built-in function `ml_evaluate`. You can evaluate the model by using indexes such as area under curve (AUC), Kolmogorov-Smirnov (KS), and F1 score. Syntax:

  ```
  ml_evaluate(table <data_source>[, map<string, string> <parameters>])
  ```

- Multiclass classification evaluation: implemented by using the built-in function `ml_multiclass_evaluate`. You can evaluate a multiclass classification model based on its prediction and actual results. The evaluation indexes include accuracy, kappa, and F1 score. Syntax:

  ```
  ml_multiclass_evaluate(table <data_source>[, map<string, string> <parameters>])
  ```

- Linear regression evaluation: implemented by using the built-in function `ml_regression_evaluate`. You can evaluate a linear algorithm model based on its prediction and actual results such as the indexes and residual histogram. The evaluation indexes include SST, SSE, SSR, R2, R, MSE, RMSE, MAE, MAD, MAPE, count, yMean, and predictMean. Syntax:

  ```
  ml_regression_evaluate(table <data_source>[, map<string, string> <parameters>])
  ```

In the preceding syntax:

- data_source: required. This parameter specifies the data to be evaluated. The label results and prediction results must be included. The value can be a table or a SELECT statement.
- parameters: optional. This parameter specifies the parameters used for the prediction. The parameters are the same as those in PAI. For more information about the parameters, see Linear SVM, PS-SMART Regression, or GBDT Regression.

3.14.2. Quick start

This topic provides an example to describe how to use MaxCompute SQLML and the logistic regression for binary classification model of Machine Learning Platform for AI (PAI). In this example, Mushroom Data Set, an open source dataset is used to make predications about whether mushrooms are toxic.

Prerequisites

An Alibaba Cloud account is created and passes the real-name verification. For more information, see Create an Alibaba Cloud account.

If you want to perform operations as a RAM user, make sure that the RAM user is available and authorized. For more information, see Create a RAM user.

Procedure

1. (Optional)Activate a MaxCompute pay-as-you-go resource package, DataWorks Basic, and Pay-As-You-Go (PAI Studio, DSW, and EAS). The services must be in the same region.
   i. Go to the product page of Alibaba Cloud MaxCompute and click **Buy Now**.

   For more information, see Activate MaxCompute and DataWorks.

   > **Note**
   > - If you have not activated MaxCompute but use this method to activate MaxCompute, DataWorks Basic and a MaxCompute pay-as-you-go resource package are activated by default.
   > - If you have activated a MaxCompute pay-as-you-go resource package, skip this step.

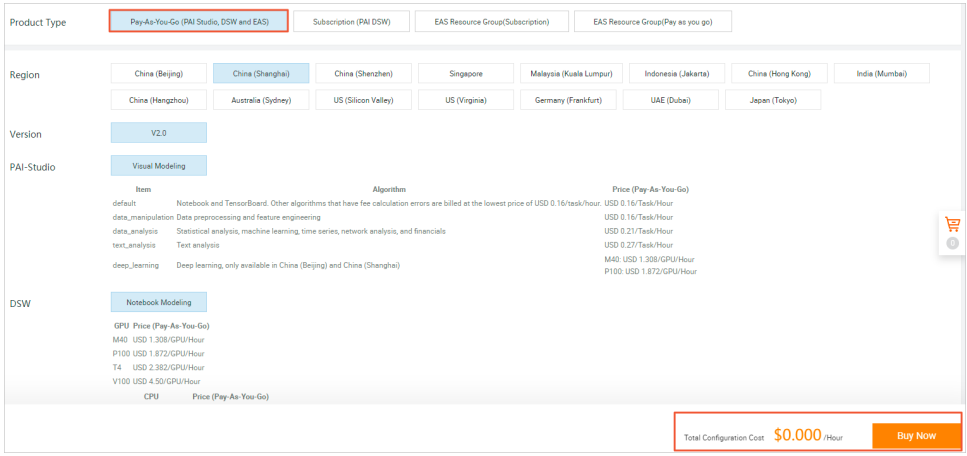
   ii. Go to the buy page of DataWorks and activate **DataWorks Basic**.

   For more information, see Activate DataWorks.

   > **Note** If you have activated DataWorks Basic, skip this step.

   iii. Go to the buy page of PAI and activate Pay-As-You-Go (PAI Studio, DSW, and EAS).

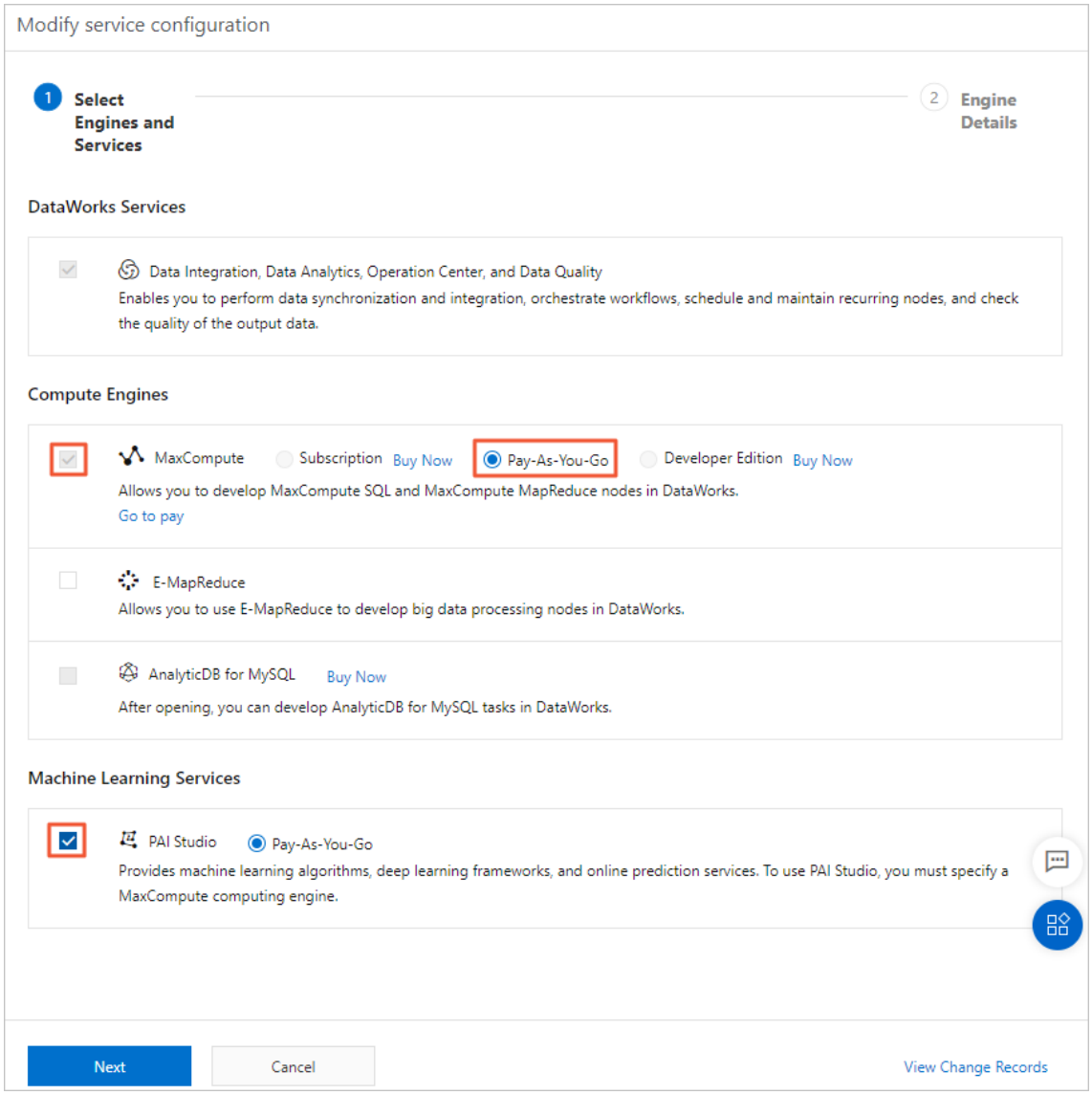
   For more information, see Activate PAI.



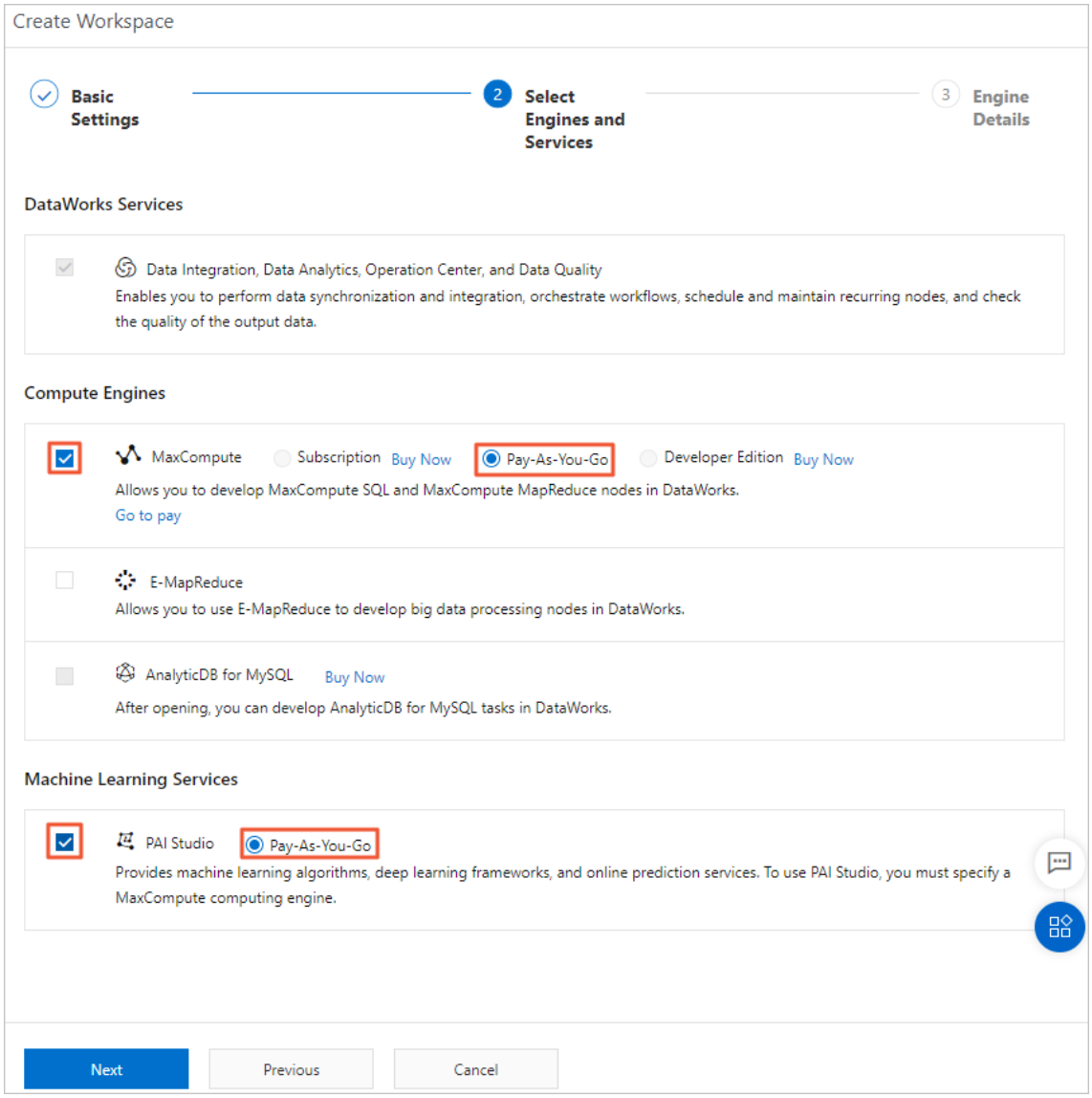
   > **Note** If you have activated Pay-As-You-Go (PAI Studio, DSW, and EAS), skip this step.

2. Download the Mushroom Data Set file named agaricus-lepiota.data and save it as a TXT, CSV, or LOG file. Example: agaricus-lepiota.data.txt.
3. Log on to the DataWorks console and create or configure a DataWorks workspace.
   - If a DataWorks workspace is available, go to the Workspaces page. Find the workspace that you want to manage and click **Modify service configuration** in the Actions column. In the panel that appears, set Compute Engines to **MaxCompute** (pay-as-you-go) and Machine Learning

Services to **PAI Studio** (pay-as-you-go).



- If no DataWorks workspaces are available, create one. In the Create Workspace panel, set Compute Engines to **MaxCompute** (pay-as-you-go) and Machine Learning Services to **PAI Studio** (pay-as-you-go). For more information, see Create a workspace.



4. In the DataWorks console, create a table named mushroom_classification and import the prepared dataset.
   i. Find the workspace that you want to manage and click **Data Analytics** in the **Actions** column. On the page that appears, create a table named mushroom_classification.

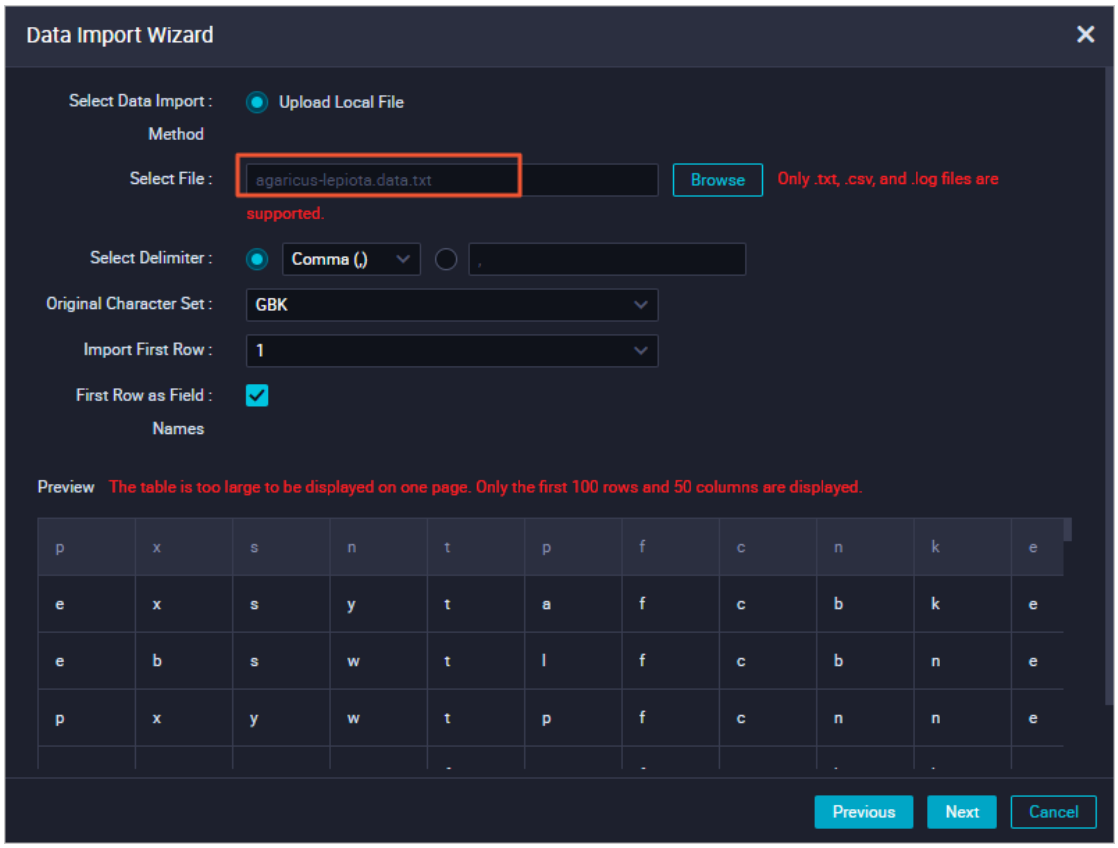
      For more information, see Create a MaxCompute table.

      The following DDL statement is used to create a table:

```
create table mushroom_classification (
    label      string               comment 'poisonous=p,edible=e',
    cap_shape  string               comment 'bell=b,conical=c,convex=x,flat=f,knobb
ed=k,sunken=s',
    cap_surface string              comment 'fibrous=f,grooves=g,scaly=y,smooth=s',
    cap_color string                comment 'brown=n,buff=b,cinnamon=c,gray=g,green
=r,pink=p,purple=u,red=e,white=w,yellow=y',
    bruises string                  comment 'bruises=t,no=f',
    odor string                     comment 'almond=a,anise=l,creosote=c,fishy=y,fo
ul=f,musty=m,none=n,pungent=p,spicy=s',
    gill_attachment string          comment 'attached=a,descending=d,free=f,notched
=n',
    gill_spacing string             comment 'close=c,crowded=w,distant=d',
    gill_size string                comment 'broad=b,narrow=n',
    gill_color string               comment 'black=k,brown=n,buff=b,chocolate=h,gra
y=g,green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y',
    stalk_shape string              comment 'enlarging=e,tapering=t',
    stalk_root string               comment 'bulbous=b,club=c,cup=u,equal=e,rhizomo
rphs=z,rooted=r,missing=?',
    stalk_surface_above_ring string comment 'fibrous=f,scaly=y,silky=k,smooth=s',
    stalk_surface_below_ring string comment 'fibrous=f,scaly=y,silky=k,smooth=s',
    stalk_color_above_ring string   comment 'brown=n,buff=b,cinnamon=c,gray=g,orang
e=o,pink=p,red=e,white=w,yellow=y',
    stalk_color_below_ring string   comment 'brown=n,buff=b,cinnamon=c,gray=g,orang
e=o,pink=p,red=e,white=w,yellow=y',
    veil_type string                comment 'partial=p,universal=u',
    veil_color string               comment 'brown=n,orange=o,white=w,yellow=y',
    ring_number string              comment 'none=n,one=o,two=t',
    ring_type string                comment 'cobwebby=c,evanescent=e,flaring=f,larg
e=l,none=n,pendant=p,sheathing=s,zone=z',
    spore_print_color string        comment 'black=k,brown=n,buff=b,chocolate=h,gre
en=r,orange=o,purple=u,white=w,yellow=y',
    population string               comment 'abundant=a,clustered=c,numerous=n,scat
tered=s,several=v,solitary=y',
    habitat string                  comment 'grasses=g,leaves=l,meadows=m,paths=p,u
rban=u,waste=w,woods=d'
);
```

ii. Import the data in the agaricus-lepiota.data.txt file to the mushroom_classification table. Set Select a method for matching to **By Location**.

For more information, see Create tables and import data.



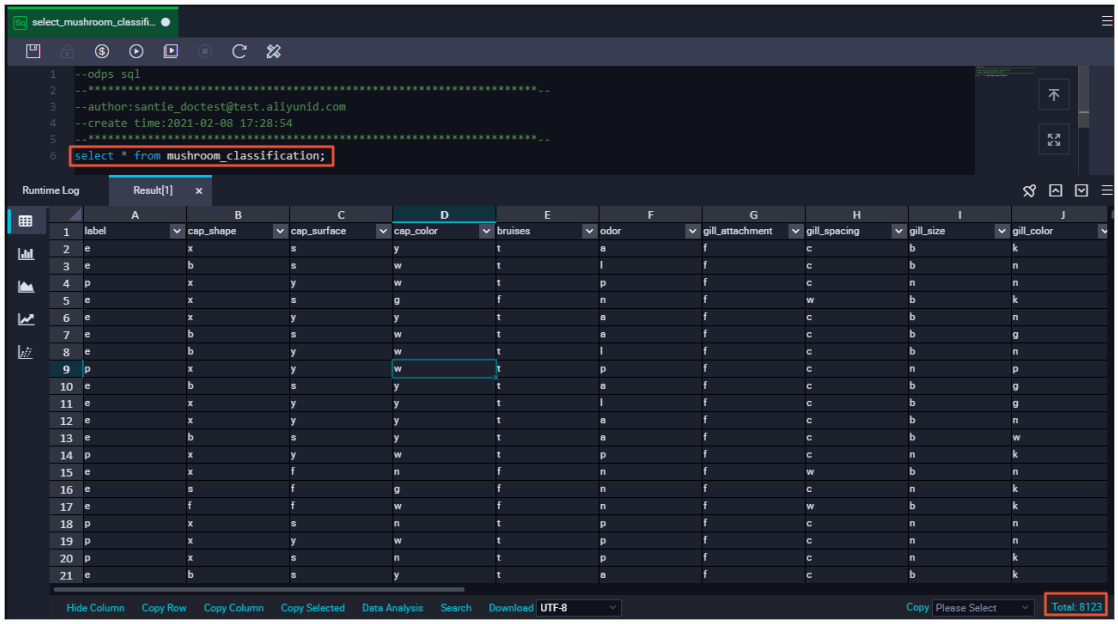
iii. Use the ad hoc query feature of DataWorks, create an ODPS SQL node, and then execute SQL statements to verify the data import results.

For more information, see Use the ad-hoc query feature to execute SQL statements (optional).

Sample statement:

```
select * from mushroom_classification;
```

The following results are returned.



5. Use one-hot encoding to process data in the mushroom_classification table.

The fields must be numeric for the logistic regression for binary classification model. Therefore, one-hot encoding is used to convert the enumerated values into numeric values. For example, valid values for cap_shape are `b`, `c`, `x`, `f`, `k`, and `s`. One-hot encoding converts the six enumerated values into six columns. Each column corresponds to an enumerated value. If the value of cap_shape is equal to the enumerated value of the column to which it corresponds, enter 1. Otherwise, enter 0.

i. (Optional)Create a workflow. Example: mc_test.

For more information, see Create a workflow.

> **Note** If you have created a workflow, skip this step.

ii. Create an ODPS Script node, write code, use one-hot encoding to process the imported data, and then write the processed data into the new table named mushroom_classification_one_hot.

For more information, see Create an ODPS Script node.

Sample statement:

```
create temporary function one_hot as 'onehot.OneHotEncoding' using
#CODE ('lang'='JAVA')
package onehot;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.UDTF;
```

```
import com.aliyun.odps.udf.annotation.Resolve;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
@Resolve({"string,string,string,string,string,string,string,string,string,string,"
+
        "string,string,string,string,string,string,string,string,string,string,stri
ng,string" +
        "->" +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,"+
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint,bigint," +
        "bigint,bigint,bigint,bigint,bigint,bigint"})
public class OneHotEncoding extends UDTF {
  private static char[][] features = {
          { 'b','c','x','f','k','s'}, //cap-shape
          { 'f','g','y','s'}, //cap-surface
          { 'n','b','c','g','r','p','u','e','w','y'}, //cap-color
          { 't','f'}, //bruises
          { 'a','l','c','y','f','m','n','p','s'}, //odor
          { 'a','d','f','n'}, //gill-attachment
          { 'c','w','d'}, //gill-spacing
          { 'b','n'}, //gill-size
          { 'k','n','b','h','g','r','o','p','u','e','w','y'}, //gill-color
          { 'e','t'}, //stalk-shape
          { 'b','c','u','e','z','r','?'}, //stalk-root
          { 'f','y','k','s'}, //stalk-surface-above-ring
          { 'f','y','k','s'}, //stalk-surface-below-ring
          { 'n','b','c','g','o','p','e','w','y'}, //stalk-color-above-ring
          { 'n','b','c','g','o','p','e','w','y'}, //stalk-color-below-ring
          { 'p','u'}, //veil-type
          { 'n','o','w','y'}, //veil-color
          { 'n','o','t'}, //ring-number
          { 'c','e','f','l','n','p','s','z'}, //ring-type
          { 'k','n','b','h','r','o','u','w','y'}, //spore-print-color
          { 'a','c','n','s','v','y'}, //population
          { 'g','l','m','p','u','w','d'}, //habitat
  };
  @Override
  public void process(Object[] objects) throws UDFException, IOException {
    List<Long> featuresEncoding = new ArrayList<>(126);
    for (int i = 0; i < objects.length; i++) {
      String value = (String)objects[i];
      char[] feature = features[i];
      for (char c : feature) {
```

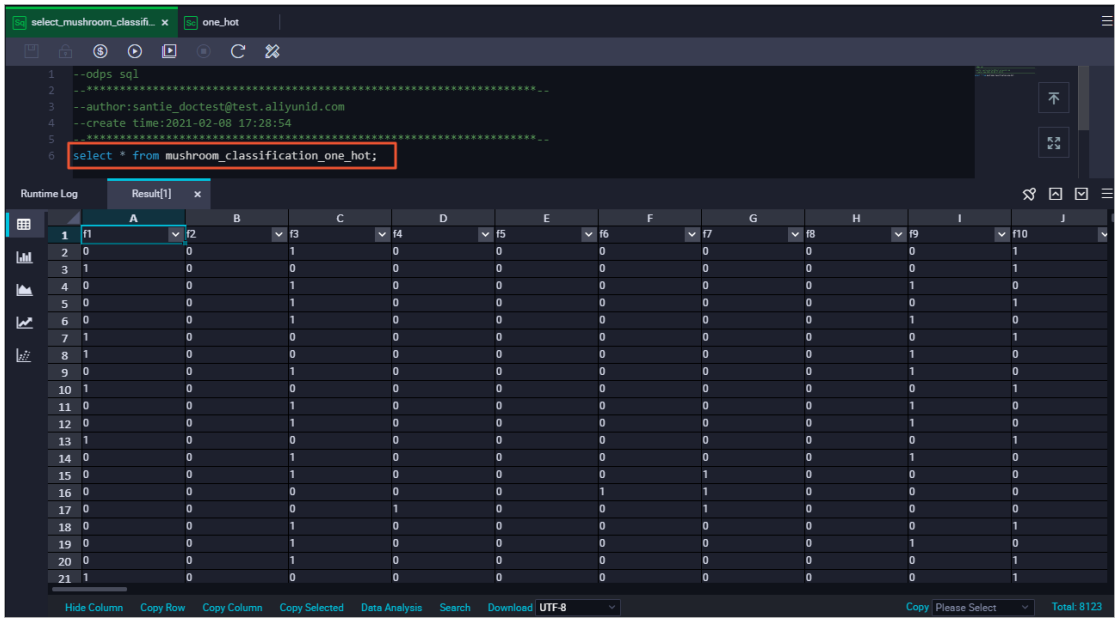
```
        featuresEncoding.add(value.charAt(0) == c ? 1L : 0L);
      }
    }
    forward(featuresEncoding.toArray());
  }
}
#END CODE;
create table mushroom_classification_one_hot as
select t.*, label
from mushroom_classification
lateral view
one_hot(cap_shape,cap_surface,cap_color,bruises,odor,gill_attachment,
        gill_spacing, gill_size, gill_color, stalk_shape,stalk_root ,
        stalk_surface_above_ring,stalk_surface_below_ring,stalk_color_above_ring,
        stalk_color_below_ring,veil_type,veil_color,ring_number,ring_type,spore_pri
nt_color,
        population,habitat) t
AS f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15,f16,f17,f18,f19,f20,
f21,f22,f23,f24,f25,f26,f27,f28,f29,f30,f31,f32,f33,f34,f35,f36,f37,f38,f39,f40,
f41,f42,f43,f44,f45,f46,f47,f48,f49,f50,f51,f52,f53,f54,f55,f56,f57,f58,f59,f60,
f61,f62,f63,f64,f65,f66,f67,f68,f69,f70,f71,f72,f73,f74,f75,f76,f77,f78,f79,f80,
f81,f82,f83,f84,f85,f86,f87,f88,f89,f90,f91,f92,f93,f94,f95,f96,f97,f98,f99,f100,
f101,f102,f103,f104,f105,f106,f107,f108,f109,f110,f111,f112,f113,f114,f115,f116,
f117,f118,f119,f120,f121,f122,f123,f124,f125,f126;
```

iii. Use the ad hoc query feature of DataWorks, create an ODPS SQL node, and then execute SQL statements to verify the results of one-hot encoding-based processing.

Sample statement:

```
select * from mushroom_classification_one_hot;
```

The following results are returned.



iv. Use the ad hoc query feature of DataWorks, create an ODPS SQL node, and then create a training dataset and a test dataset based on the data in mushroom_classification_one_hot.

Sample statement:

```
-- Training dataset. About 25% of the data is used to train models.
create table mushroom_training as
select * from mushroom_classification_one_hot where sample(4,1);
-- Test dataset. About the rest 75% of the data is used to predict and evaluate mod
els.
create table mushroom_predict as
select * from mushroom_classification_one_hot except all select * from mushroom_tra
ining;
```

6. Create a machine learning model and make predictions.

i. Use the ad hoc query feature of DataWorks, create an ODPS SQL node, and then create a logistic regression for binary classification model based on the training dataset. The model is named lr_test_model.

Sample statement:

```
create model lr_test_model
with properties('model_type'='logisticregression_binary', 'goodValue'='p','maxIter'
='1000')
as select * from mushroom_training;
```

> **Note**
>
> - You can specify more parameters in `properties`. The parameters are the same as those in PAI. For more information, see Linear SVM.
> - The SQL engine extracts and executes the statement after `as` and stores the results in a temporary table. You can view the results in Summary in the Logview of a job. The lifecycle of the temporary table is one day. If the table expires, it is automatically deleted.
> - If you want to delete the model, execute the `drop offlinemodel lr_test_model` statement.

ii. Use the ad hoc query feature of DataWorks, create an ODPS SQL node, and then use `ml_predict` to predict the data in the test dataset based on lr_test_model.

Sample statement:

```
create table mushroom_predict_result as
select * from ml_predict(
    lr_test_model,
    (select * from mushroom_predict)
);
```

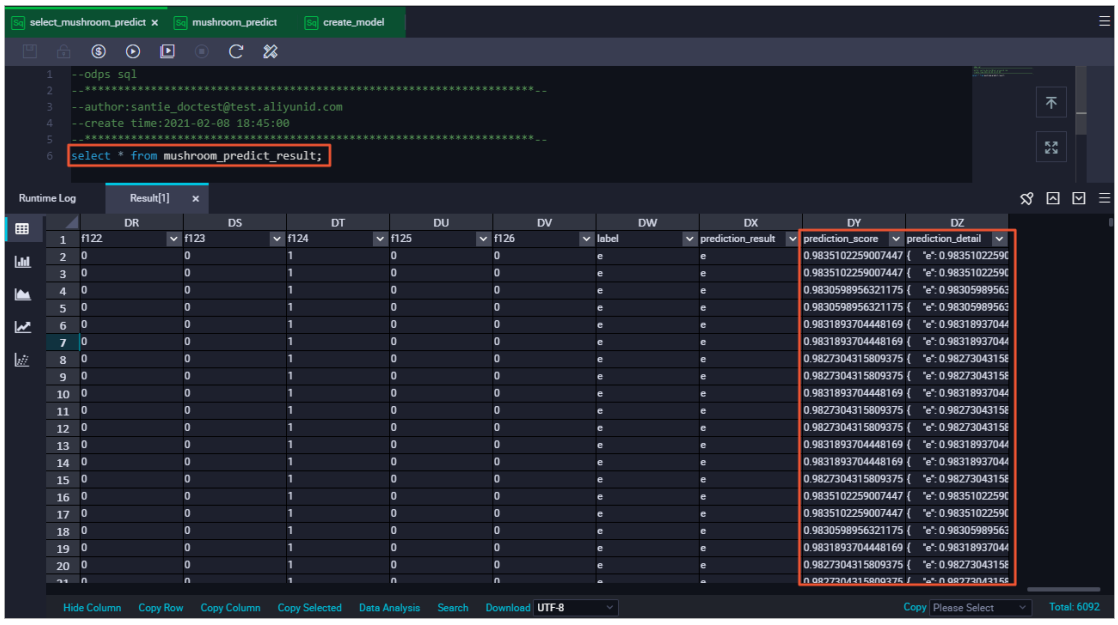
> **Note**
>
> - The SQL engine saves the subquery results of `ml_predict` to a temporary table. The lifecycle of the temporary table is one day. If the table expires, it is automatically deleted.
> - The results of `ml_predict` can be placed in the `FROM` clause of the query statement. You can also execute the `INSERT` or `CREATE TABLE AS` statement to save the results to another table. For more information about `ml_predict`, see Supported model prediction functions.

iii. Use the ad hoc query feature of DataWorks, create an ODPS SQL node, and then execute the required statement to view the prediction results for the mushroom_predict_result table.

Sample statement:

```
select * from mushroom_predict_result;
```

The following results are returned.



7. Use `ml_evaluate` to evaluate the prediction accuracy of the model.

   For more information about `ml_evaluate`, see Supported model evaluation functions.

3.14.3. Billing

This topic describes how to estimate the cost of a MaxCompute SQLML job.

Context

In most cases, each algorithm component consists of multiple subtasks. To calculate the cost, you must first calculate the costs of the subtasks in each algorithm component and sum up the costs of all the components. MaxCompute and Machine Learning Platform for AI (PAI) are used to run a MaxCompute SQLML job. Therefore, the cost of the MaxCompute SQLML job consists of the costs generated for both MaxCompute and PAI.

Billing rules

The underlying machine learning capabilities of MaxCompute SQLML are independently charged based on the billing rules for PAI. Therefore, the cost of a MaxCompute SQLML job is calculated based on the following parts:

- SQL job: The computing and storage costs of an SQL job are calculated based on the billing rules for MaxCompute SQL. For more information, see Billing method.
- Machine learning model: The cost of a machine learning model is calculated based on the billing rules for Machine Learning Studio. The following table describes the rules. For more information about PAI billing, see Billing of Machine Learning Studio.

Machine learning model	Unit price (USD/compute hour)	Region
Logistic regression for binary classification	0.21	○ China (Beijing) ○ China (Shanghai) ○ China (Hangzhou) ○ China (Shenzhen) ○ China (Hong Kong)
Logistic regression for multiclass classification		
Linear regression		

Billing example

You must obtain the usage of CPU cores and memory to estimate the cost based on the billing rules for MaxCompute SQLML.

In this example, an existing prediction dataset in Quick start is used to estimate the cost.

1. Use the ad hoc query feature of DataWorks, create a logistic regression for binary classification model based on the prediction dataset mushroom_predict, and then obtain the Logview URL. The model is named lr_test_model_1.

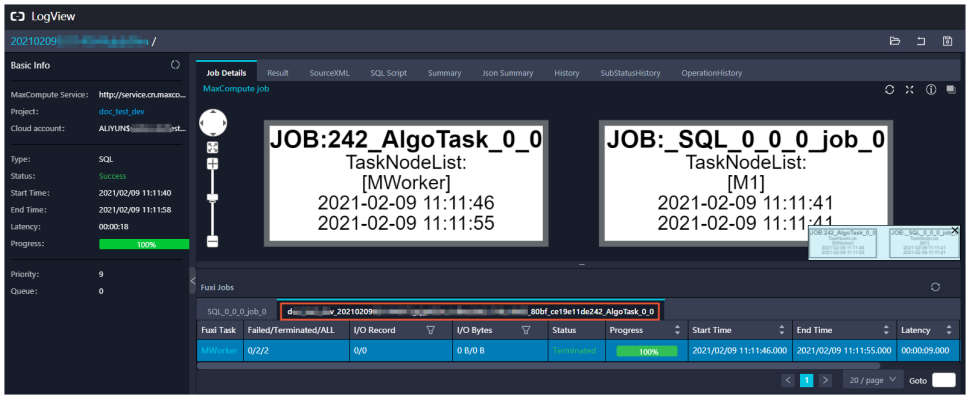
   Sample statement:

   ```
   create model lr_test_model_1
   with properties('model_type'='logisticregression_binary', 'goodValue'='p','maxIter'='10
   00')
   as select * from mushroom_predict;
   ```

   The following results are returned.

2. Log on to Logview based on the Logview URL and obtain the instance ID of the AlgoTask job in Logview.

   A PAI job is defined as an AlgoTask job in MaxCompute. The naming convention for an AlgoTask job is `Projectname+'_'+InstanceID+"AlgoTask_x_x"`. You can obtain the instance ID from the job name.



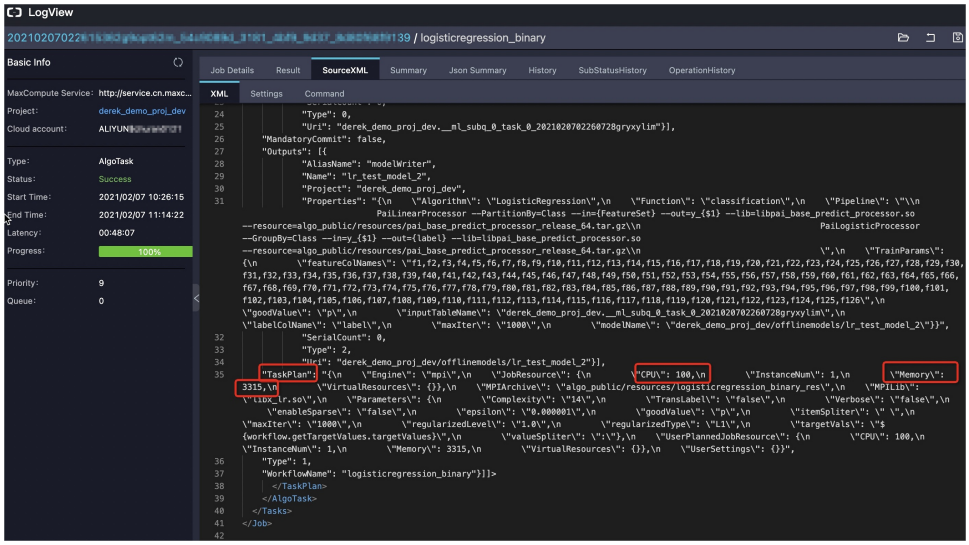
3. Use the ad hoc query feature of DataWorks and obtain the Logview URL of the PAI job based on the instance ID.

   Sample statement:

```
wait 20210209031144617gcjpt62m_1dbea3a8_17de_40d5_80bf_ce19e11d****;
```

   The following results are returned.

4. Log on to Logview based on the Logview URL of the PAI job. Go to the **SourceXML** tab and query the usage of CPU cores and memory for TaskPlan.



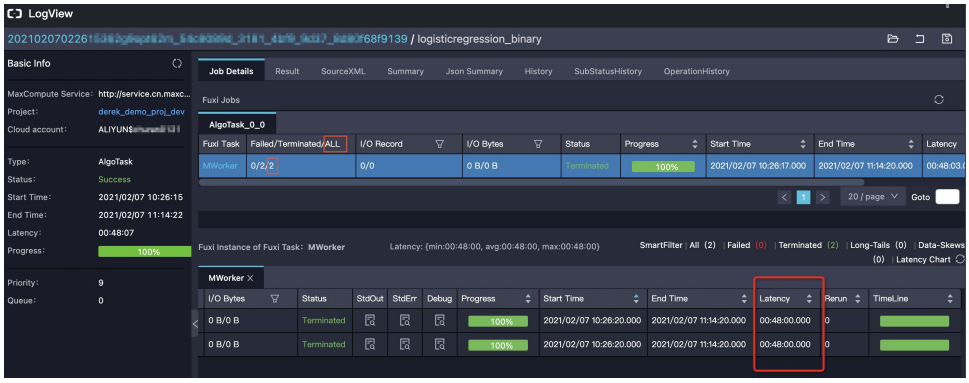
In the preceding rules:

- The number of used CPU cores is calculated by dividing the value of CPU by 100. In this example, one CPU core is used to run the job.
- The unit of Memory is MB. In this example, 3315 MB (3.2 GB) of memory is used to run the job.

> **Notice** The preceding results show the CPU cores and memory that are used by a subtask. A job may have multiple subtasks. To calculate the number of subtasks for a job, perform the next step.

5. Log on to Logview based on the Logview URL of the PAI job and obtain the number and duration of subtasks on the **Job Details** tab.

   The following figure shows that two subtasks exist and the duration for each subtask is 48 minutes.



In this example, the number of used CPU cores is 2 (2 x 1), the used memory size is 6.4 GB (2 x 3.2 GB), and the duration is 48 minutes.

6. Estimate the total cost.

   i. Calculate the number of billable hours.

```
Number of billable hours = Max(Number of CPU cores x Duration,Used memory size x Du
ration/4) = Max[2 x 1x (48 x 60/3600),2 × 3.2 × (48 x 60/3600)/4] ≈ 1.6
```

ii. Calculate the cost of the PAI job.

```
Cost of a PAI job = Number of billable hours for subtasks × Unit price ≈ 1.6 × 0.21
= 0.336 USD
```

The cost of the PAI job in this example is 0.336 USD.

iii. Calculate the cost of the SQL job. For more information, see Computing pricing.

iv. The sum of the costs for the PAI job and the SQL job equals the cost of a MaxCompute SQLML job.

View your bills

In your bill, the cost of a MaxCompute SQLML job is the sum of the costs generated for both MaxCompute and PAI.

3.15. External table

3.15.1. Overview

MaxCompute is the core computing component of the Alibaba Cloud big data platform and provides powerful computing capabilities. MaxCompute schedules a large number of nodes to run computing jobs in parallel. It also provides mechanisms to systematically process and manage distributed computing features, such as failovers and retries.

Background information

MaxCompute SQL provides an entry point for distributed data processing. This allows you to process and store exabytes of offline data. The computing framework of MaxCompute continues to evolve to meet the requirements that arise from expanded big data business and new use scenarios. In early versions, MaxCompute provides powerful computing capabilities to process internal data in special formats. It is now available to process external data.

MaxCompute SQL is now used to process structured data that is stored in MaxCompute internal tables in the **CFile** column store format. You must use different tools to import external user data to MaxCompute tables for data computations. The user data includes texts and unstructured data. For example, to process Object Storage Service (OSS) data in MaxCompute, you can use one of the following methods:

- Use OSS SDK or other tools to download data from OSS. Then, use MaxCompute Tunnel to import the downloaded data to a MaxCompute table.
- Write a user-defined function (UDF) to call OSS SDK and access OSS data.

However, the two methods have deficiencies.

- The first method requires data transfer operations outside the MaxCompute system. If a large amount of OSS data needs to be processed, parallel operations are required to accelerate the process. As a result, you cannot fully utilize the large-scale computing capabilities of MaxCompute.
- The second method requires UDF-based access permissions. It also requires that developers control the number of parallel jobs and handle issues related to data partitioning.

MaxCompute provides external tables to address these issues. External tables are used to process data that is stored outside MaxCompute internal tables. You can execute a simple Data Definition Language (DDL) statement to create an external table in MaxCompute. Then, you can use this table to associate it with external data sources. This allows access to and output of data in various formats. In most cases, external tables can be accessed like standard MaxCompute tables. You can fully utilize the computing capabilities of MaxCompute SQL to process external data.

> **Note**
>
> - If you use an external table, the data in this table is not stored in MaxCompute, and you are not charged for the storage of the table data.
> - Full search is supported for external tables.
> - Tunnel commands and Tunnel SDK cannot be used for external tables. You can use Tunnel to upload data to MaxCompute internal tables. You can also use OSS SDK for Python to upload data to OSS and map the data to external tables in MaxCompute.
> - You can create, search, configure, and process external tables in the DataWorks console. You can also query and analyze data in external tables. For more information, see External table.

Examples

This section describes how to use MaxCompute external tables to process unstructured data:

- To access unstructured data in OSS and Tablestore, see Access OSS data and Access Tablestore data.
- To use external tables to access OSS data, you must authorize MaxCompute to access OSS. The authorization is performed in the Resource Access Management (RAM) console. For more information, see STS authorization.
- The unstructured data processing framework of MaxCompute allows you to export MaxCompute data to OSS by using the INSERT statement. For more information, see Write data to OSS.
- For more information about how to process data in various open source formats, see Open source data formats supported by OSS external tables.

3.15.2. STS authorization

This topic describes how to authorize MaxCompute to access Object Storage Service (OSS) and Tablestore by using one-click authorization or by customizing a Resource Access Management (RAM) role. MaxCompute uses RAM and Security Token Service (STS) of Alibaba Cloud to secure data access.

STS authorization for OSS

To access OSS data by using MaxCompute external tables, you must grant OSS access permissions to the Alibaba Cloud account that is used to run MaxCompute jobs. You can use one of the following methods to grant OSS access permissions:

- Method 1: If MaxCompute and OSS are within the same Alibaba Cloud account, log on to the RAM console and perform one-click authorization. We recommend that you use this method.
- Method 2: If MaxCompute and OSS are not within the same Alibaba Cloud account, you can customize a role and grant permissions to the role.
  i. Create a RAM role.

Log on to the RAM console by using your OSS account and create a role on the RAM Roles page in the RAM console. For example, you can create the oss-admin role.

For more information about how to create a RAM role, see Create a RAM role for a trusted Alibaba Cloud account.

ii. Modify the policy of the RAM role.

On the RAM Roles page in the RAM console, click the name of the created RAM role in the RAM Role Name column. On the page that appears, click the **Trust Policy Management** tab and click Edit Trust Policy to modify the policy.

For more information about how to modify a policy, see Edit the trust policy of a RAM role. The following sample code shows the content of a policy.

```
-- If MaxCompute and OSS are not within the same Alibaba Cloud account, execute the f
ollowing statements to allow MaxCompute to access OSS.
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "ID of the Alibaba Cloud account that owns the MaxCompute project@odps.aliyuncs.
com"
   ]
 }
}
],
"Version": "1"
}
```

`ID of the Alibaba Cloud account that owns the MaxCompute project` is the account that is used to access OSS.

iii. Create a policy.

On the Policies page in the RAM console, create a policy, such as AliyunODPSRolePolicy. The following sample code describes the policy content. You can also customize policies.

```
{
"Version": "1",
"Statement": [
{
 "Action": [
   "oss:ListBuckets",
   "oss:GetObject",
   "oss:ListObjects",
   "oss:PutObject",
   "oss:DeleteObject",
   "oss:AbortMultipartUpload",
   "oss:ListParts"
 ],
 "Resource": "*",
 "Effect": "Allow"
}
]
}
```

iv. Attach the created policy AliyunODPSRolePolicy to the RAM role.

For more information about authorization, see Grant permissions to a RAM role.

STS authorization for Tablestore

To access Tablestore data by using MaxCompute external tables, you must grant Tablestore access permissions to the Alibaba Cloud account that is used to run MaxCompute jobs. You can use one of the following methods to grant permissions to the account:

- Method 1: If MaxCompute and OSS are within the same Alibaba Cloud account, log on to the RAM console and perform one-click authorization. We recommend that you use this method.
- Method 2: If MaxCompute and Tablestore are not within the same Alibaba Cloud account, you can customize a role and grant permissions to the role.
  i. Create a RAM role.

  Log on to the RAM console by using your Tablestore account and create a role on the RAM Roles page in the RAM console. For example, you can create the oss-adminots role.

  For more information how to create a RAM role, see Create a RAM role for a trusted Alibaba Cloud account.

  ii. Modify the policy of the RAM role.

  On the RAM Roles page in the RAM console, click the name of the created RAM role in the RAM Role Name column. On the page that appears, click the **Trust Policy Management** tab and click Edit Trust Policy to modify the policy.

```
-- If MaxCompute and Tablestore are not within the same Alibaba Cloud account, execut
e the following statements to allow MaxCompute to access Tablestore.
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "UID of the Alibaba Cloud account that owns the MaxCompute project@odps.aliyuncs
.com"
   ]
 }
}
],
"Version": "1"
}
```

`ID of the Alibaba Cloud account that owns the MaxCompute project` is the account that is used to access Tablestore.

iii. Create a policy.

On the Policies page in the RAM console, create a policy, such as AliyunODPSRolePolicy. The following sample code shows the policy content. You can also customize policies.

```
{
"Version": "1",
"Statement": [
{
 "Action": [
   "ots:ListTable",
   "ots:DescribeTable",
   "ots:GetRow",
   "ots:PutRow",
   "ots:UpdateRow",
   "ots:DeleteRow",
   "ots:GetRange",
   "ots:BatchGetRow",
   "ots:BatchWriteRow",
   "ots:ComputeSplitPointsBySize"
 ],
 "Resource": "*",
 "Effect": "Allow"
}
]
}
```

iv. Attach the created policy AliyunODPSRolePolicy to the RAM role.

For more information about authorization, see Grant permissions to a RAM role.

3.15.3. Network connection process

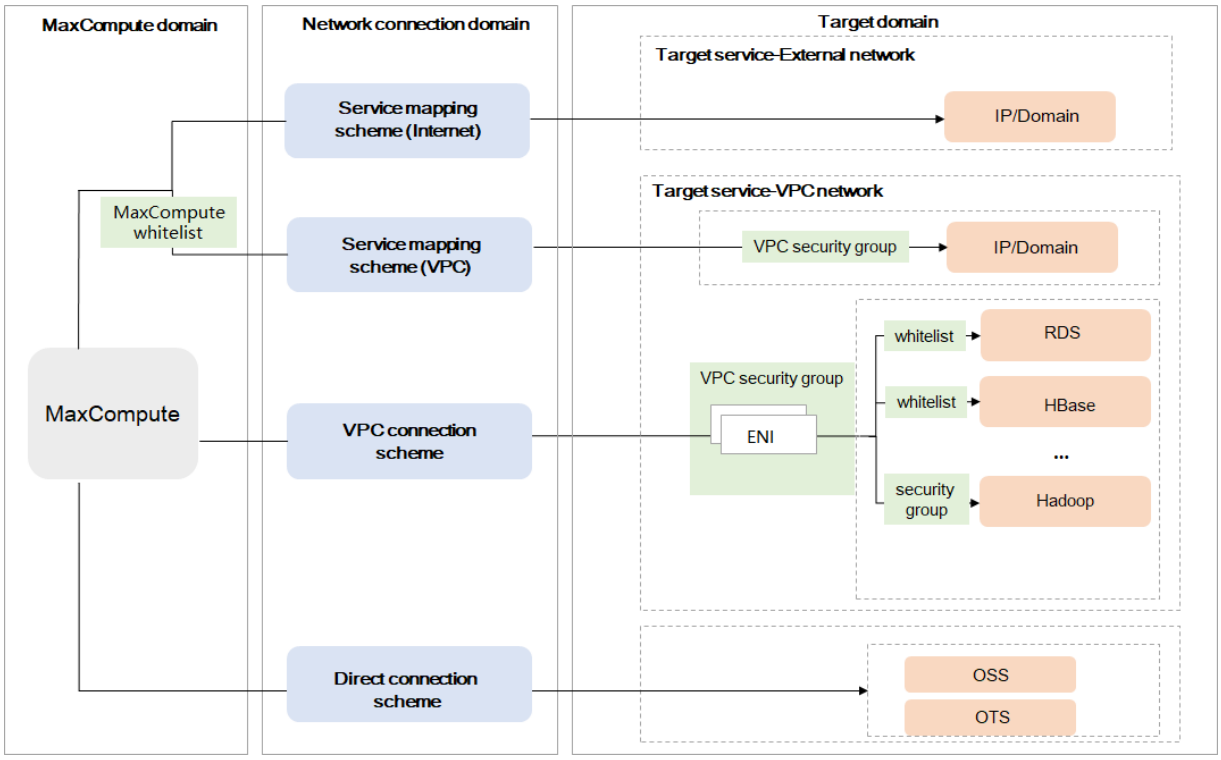
By default, MaxCompute cannot access a service over the Internet or over a virtual private cloud (VPC). To allow the access, you must establish a network connection between MaxCompute and the specified object, such as an IP address, an endpoint, an ApsaraDB RDS instance, ApsaraDB for HBase, or Hadoop cluster. This topic describes the network architecture between MaxCompute and the object that you want to access and the supported network connection schemes.

Background information

If you want to access an object over the Internet or over a VPC by using one of the following methods, you must apply for a network connection between MaxCompute and the object.

- Call a user-defined function (UDF) that you created to access an object over the Internet or over a VPC.
- Use an external table that you created to access an object over the Internet or over a VPC.
- Use the lakehouse solution of MaxCompute to access an object in a VPC.

The following figure shows the network architecture between MaxCompute and the object that you want to access and the supported network connection schemes.



You can use one of the following schemes to allow MaxCompute to access an object:

- Service mapping scheme

  The service mapping scheme varies based on the type of the network in which the object you want to access resides.

- Service mapping scheme (Internet)

  You can use this scheme if you want to access a public IP address or a public endpoint by calling a UDF or by using an external table. If you use this scheme, you must fill in the Network connection application form. If no security limits are imposed on the destination IP address or endpoint, you can access the IP address or endpoint after the application is approved.

  > **Note** If security limits are imposed on the destination IP address or endpoint, contact the MaxCompute technical support team to handle this issue based on the security limits.

- Service mapping scheme (VPC)

  You can use this scheme if you want to access an IP address or endpoint in a VPC by calling a UDF or by using an external table when MaxCompute is connected to the VPC. You need only to add the Classless Inter-Domain Routing (CIDR) blocks of the region in which your MaxCompute project resides to the security group of the VPC and add the VPC in which the destination IP address or endpoint resides to your MaxCompute project. After you grant mutual access between MaxCompute and the VPC, MaxCompute can access the destination IP address or endpoint in the VPC.

- VPC connection scheme

  You can use this scheme if you want to access an ApsaraDB RDS instance, an ApsaraDB for HBase cluster, or a Hadoop cluster that resides in a VPC by calling a UDF or by using the lakehouse solution. If you use this scheme, you must complete authorization and create and configure a security group for MaxCompute in the VPC console, fill in the Network connection application form, and then configure the security group for the service that you want to access to establish a connection between MaxCompute and the service.

- Direct connection scheme

  You can use this scheme if you want to access Alibaba Cloud Object Storage Service (OSS) or Tablestore by calling a UDF or by using an external table. In this scenario, the network connection between MaxCompute and OSS or Tablestore is established. You do not need to apply for a network connection.

  > **Note**
  > - If you have created an OSS or Tablestore external table, you can access OSS or Tablestore by using the internal or public endpoint of OSS or Tablestore.
  > - If you call a UDF to access OSS or Tablestore, you can access OSS or Tablestore by using only the public endpoint of OSS or Tablestore. For more information about the endpoints of OSS, see Regions and endpoints. For more information about the endpoints of Tablestore, see Endpoint.

Prerequisites

Before you use the service mapping scheme (Internet) or VPC connection scheme to apply for a network connection between MaxCompute and an object, make sure that the following conditions are met:

- A MaxCompute project is created.

If a MaxCompute project already exists, you can directly use the project without the need to create another project. If you use the lakehouse solution, we recommend that you set the data type edition for your MaxCompute project to the Hive-compatible data type edition. For more information about how to create a MaxCompute project, see Create a MaxCompute project.

- The Alibaba Cloud account of the VPC owner is obtained. The Alibaba Cloud account that is used to access the MaxCompute project and the administrator account of the destination service or cluster are also obtained.

Limits

- If you use the lakehouse solution, only the Alibaba Cloud account of the VPC owner can create MaxCompute projects.
- When you use MaxCompute to access an ApsaraDB RDS instance, an ApsaraDB for HBase cluster, or a Hadoop cluster that resides in a VPC, make sure that the MaxCompute project is in the same region as this VPC.

Supported regions

The following table describes the regions in which you can use the service mapping scheme or VPC connection scheme to establish a network connection between MaxCompute and the specified object.

Scheme	Region	Connected object
Service mapping scheme (Internet)	All regions at the China site (aliyun.com)	Public IP address or endpoint
Service mapping scheme (VPC)	• China (Beijing) • China (Shanghai)	IP address or endpoint of a VPC
VPC connection scheme	• China (Beijing) • China (Shanghai) • China (Zhangjiakou) • China (Hangzhou) • China (Shenzhen)	• IP address or endpoint of a VPC • ApsaraDB RDS instance • ApsaraDB for HBase cluster • Hadoop cluster

Service mapping scheme (Internet)

To allow MaxCompute to access a public IP address or a public endpoint, perform the following operations:

1. **Submit an application.**
   i. Open the MaxCompute network connection application form and configure the parameters shown in the following figure.

Service mapping application

ii. After the MaxCompute technical support team receives the application, the team reviews and completes the network configuration. If you receive a text message indicating that the application is approved, you can proceed with the subsequent steps.

> **Note** If the text message indicates that the review failed, fill in the form again, verify that the information is correct, and then submit the application.

2. **Test network connectivity.**

   Log on to the MaxCompute client, configure the `odps.internet.access.list` property, and execute the SQL statement that calls a UDF to access the destination IP address or endpoint. Syntax:

```
-- Set the odps.internet.access.list property to the public IP address or endpoint and
port number that are specified in the network connection application form.
set odps.internet.access.list=<ip_address:port | realm_name:port>;
-- Execute the following SQL statement to call a UDF.
select <UDF_name>("<http://ip_address | realm_name>");
```

   - ip_address:port | realm_name:port: required. The public IP address or endpoint and port number that you want to access.
   - UDF_name: the name of the UDF that you use to access the public IP address or endpoint.

     The following example shows the UDF code.

```
package com.aliyun.odps.test.udf;
import com.aliyun.odps.udf.UDF;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
public class <UDF_name> extends UDF {
  public String evaluate(String urlStr) throws IOException {
    URL url = new URL(urlStr);
    StringBuilder sb = new StringBuilder();
    try (BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStr
eam()))) {
      String line;
      while ((line = reader.readLine()) != null) {
        sb.append(line).append('\n');
      }
    }
    return sb.toString();
  }
}
```

   The built-in UDF that is created based on the sample UDF code is named url_fetch. After the network connection application is approved, execute the following statements:

```
set odps.internet.access.list=www.aliyun.com:80;
select url_fetch("http://www.aliyun.com");
```

Service mapping scheme (VPC)

To allow MaxCompute to access a service in a VPC, perform the following operations:

1. **Configure a whitelist.**
   i. Add MaxCompute CIDR blocks to a security group of the VPC that you want to access.

   You must add the MaxCompute CIDR blocks of the region in which you want to establish a network connection to the security group of the VPC to allow the IP addresses in the MaxCompute CIDR blocks to access the services in the VPC. The following table lists the CIDR blocks. For more information about how to configure VPC security groups, see VPC security groups.

Region	MaxCompute CIDR block
China (Shanghai)	100.104.49.64/26, 100.104.212.192/26, 100.104.244.0/26, and 100.104.94.0/26
China (Beijing)	100.104.218.0/26, 100.104.120.0/26, 100.104.156.192/26, 100.104.149.0/26, 100.104.49.64/26, 100.104.212.192/26, 100.104.244.0/26, 100.104.94.0/26

   ii. Log on to the MaxCompute client. Then, add the VPC to the whitelist of MaxCompute to allow the services in the VPC to access MaxCompute. Syntax:

```
-- Add the destination VPC to the whitelist of MaxCompute.
setproject odps.security.outbound.destination=<RegionID>_<VPC ID>[*];
```

   - RegionID: required. The ID of the region to which the VPC belongs. For more information about how to obtain region IDs, see Obtain the ID of the region to which the VPC belongs and the ID of the VPC.
   - VPC ID: required. The ID of the VPC that you want to access. You can log on to the VPC console and obtain the ID of the destination VPC from the **Instance ID/Name** column on the VPCs page.
   - [*]: required. A wildcard, which indicates that all IP addresses and port numbers in the VPC are added to the whitelist of MaxCompute. The brackets (`[]`) cannot be omitted.

   For example, if the ID of the destination VPC is `vpc-bp1e4p7feyvwt103j****` and the region is `China (Shanghai)`, you can run the following command to add all IP addresses and port numbers in the VPC to the whitelist of MaxCompute.

```
setproject odps.security.outbound.destination=cn-shanghai_vpc-bp1e4p7feyvwt103j****
[*];
```

2. **Test network connectivity.**

   Use the MaxCompute client to configure the following properties and submit the SQL statement that is used to call a UDF to access the destination IP address in the VPC.

   For example, if the IP address of the VPC that you want to access is `192.0.2.0` and the port number of the VPC is 80, you can run the following commands to test network connectivity.

```
-- Specify the ID of the VPC that you want to access.
set odps.vpc.id=vpc-bp1e4p7feyvwt103j****;
-- Specify the IP address and port number of the VPC that you want to access.
set odps.vpc.access.ips=192.0.2.0:80;
-- Call the UDF to access the destination IP address and port number.
select url_fetch("http://192.0.2.0:80");
```

VPC connection scheme

You can perform authorization and create and configure a security group for MaxCompute in the VPC console, and then configure the security group for the service that you want to access to allow MaxCompute to access the service. The VPC connection scheme allows MaxCompute to access multiple IP addresses or CIDR blocks in the destination VPC.

To use the VPC connection scheme, perform the following operations:

1. **Perform authorization.**

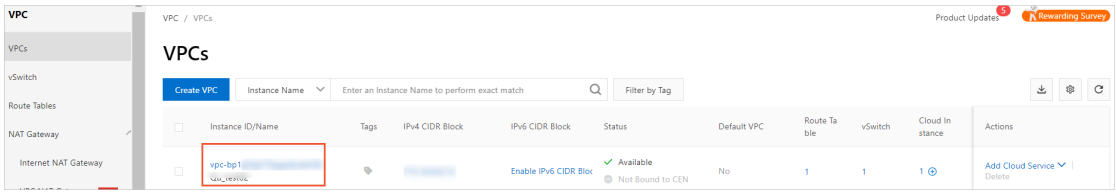
   Log on to the Alibaba Cloud Management Console by using the Alibaba Cloud account of the VPC owner, click Authorization, and then click **Confirm Authorization Policy** to complete the authorization.

   Grant the Alibaba Cloud account the permission to allow MaxCompute to create and bind elastic network interfaces (ENIs) in a VPC security group. After the authorization is complete, MaxCompute automatically creates ENIs in the VPC.

2. **Configure a security group.**

   You can create a security group for MaxCompute to manage the access of MaxCompute to various resources in the VPC.

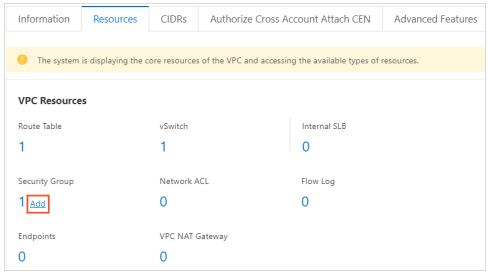
   i. Log on to the VPC console. On the VPCs page, click the ID of the destination VPC. On the page that appears, click the **Resources** tab.

   ii. In the **VPC Resources** section of the **Resources** tab, move the pointer over the value of Security Group and click **Add**. On the Security Groups page, click Create Security Group to create a security group for MaxCompute and record the security group ID.

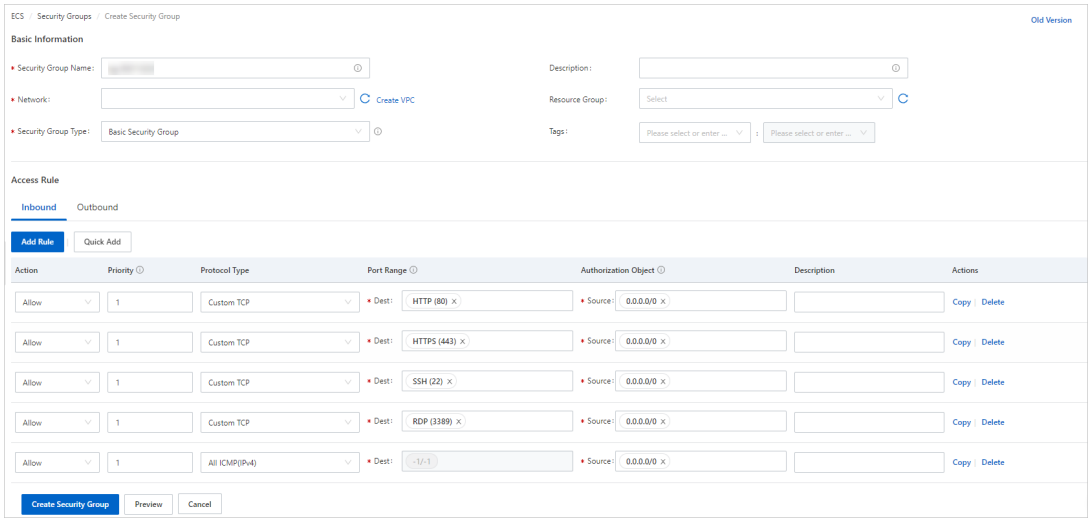
   The security group that you created must be a basic security group. Therefore, you must select the same VPC as the service that MaxCompute needs to access. For more information about how to create a security group, see Create a security group.

   - **Create a security group**

- **Configure a security group**

  You can add inbound and outbound rules to the security group based on the default configurations.

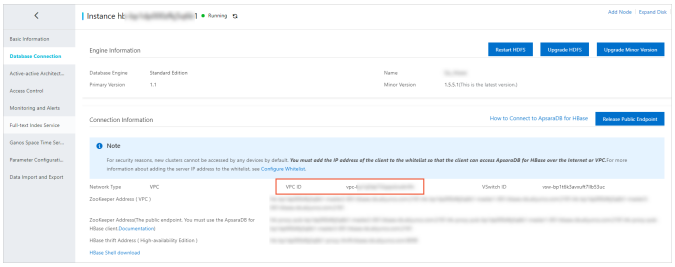
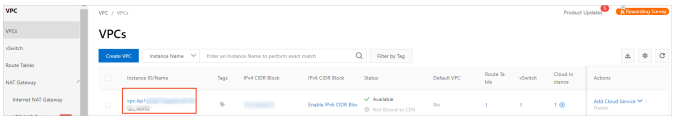


> **Note**
>
> - You must create a basic security group, instead of an advanced security group. By default, basic security groups allow outbound traffic. By default, advanced security groups do not allow outbound traffic. If you use an advanced security group, no services in the VPC can be accessed.
> - By default, MaxCompute automatically creates two ENIs based on the bandwidth requirements. You can use the two ENIs free of charge. The ENIs that are created by MaxCompute belong to the security group that you created. If you want to establish a connection between MaxCompute and an ApsaraDB for HBase cluster but the security group of MaxCompute cannot be added to the security group of the ApsaraDB for HBase cluster, you can add the IP addresses of the ENIs that are created by MaxCompute to a whitelist of the ApsaraDB for HBase cluster. The IP addresses of the ENIs may change. Therefore, we recommend that you add the CIDR block of the vSwitch to which the VPC belongs to the whitelist.
>
>   To obtain the IP addresses of the ENIs, perform the following operations: Log on to the Elastic Compute Service (ECS) console. In the left-side navigation pane, click **ENIs** in the Network & Security section to view the IP addresses of the ENIs.

3. **Submit an application.**

i. Open the MaxCompute network connection application form and configure the parameters shown in the following figure.

Network connection application form

	Obtain the ID of the region to which the VPC belongs and the ID of the VPC
	■  ■ 
	■ ■

ii. After the MaxCompute technical support team receives the application, the team reviews and completes the network configuration. If you receive a text message indicating that the application is approved, you can proceed with the subsequent steps.

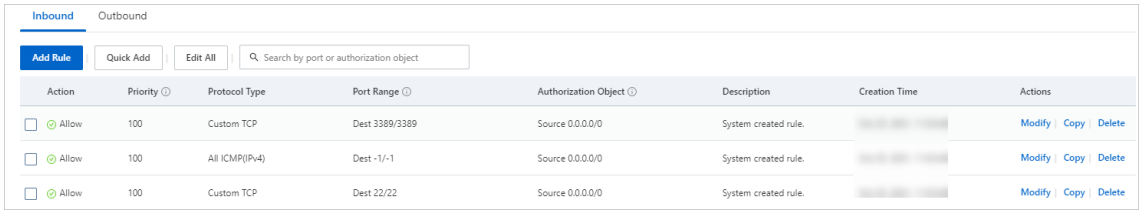
> **Note** If the text message indicates that the review failed, fill in the form again, verify that the information is correct, and then submit the application.

4. Configure the security group of the service that MaxCompute needs to access.
   - Configure the security group of the Hadoop cluster that MaxCompute needs to access.

     To ensure that MaxCompute can access a Hadoop cluster, configure the following information for the security group of the Hadoop cluster:

- Add inbound rules to the security group of the Hadoop cluster.
- Set the authorization object to the security group to which the ENIs belong. The security group is the one you created in Step 2.
- Set the port number for the Hive metastore service to 9083.
- Set the port number for NameNode of Hadoop Distributed File System (HDFS) to 8020.
- Set the port number for DataNode of HDFS to 50010.

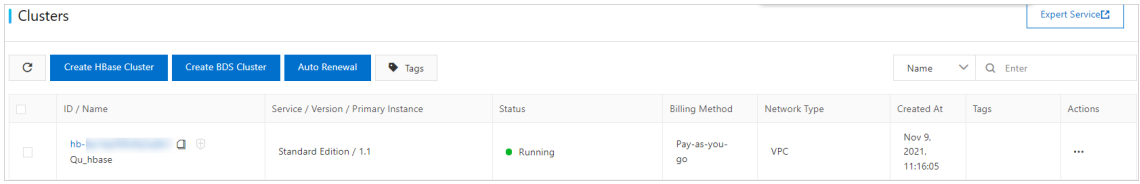
For example, if you want to allow MaxCompute to access a Hadoop cluster that is deployed on Alibaba Cloud E-MapReduce (EMR), you must configure the security group rules that are shown in the following figure. For more information about how to configure security group rules, see Add a security group rule.



- Configure the security group of an ApsaraDB for HBase cluster.

  Add the security group that is created for MaxCompute to the security group of the ApsaraDB for HBase cluster or add the IP addresses of the ENIs that are created by MaxCompute to a whitelist of the ApsaraDB for HBase cluster.

  For example, if you want to allow MaxCompute to access an ApsaraDB for HBase cluster, perform the following operations: Log on to the ApsaraDB for HBase console. On the **Clusters** page, click the name of the ApsaraDB for HBase cluster that you want to access. In the left-side navigation pane, click **Access Control**. Then, add the security group of MaxCompute on the **Security Group** tab or add the IP addresses of the ENIs created by MaxCompute to a whitelist of the ApsaraDB for HBase cluster on the **Whitelist Setting** tab. For more information about how to add a security group or add an IP address to a whitelist, see Configure IP address whitelists and security groups.



> **Note** If the security group of MaxCompute cannot be added, you can add the IP addresses of the ENIs that are created by MaxCompute to a whitelist of the ApsaraDB for HBase cluster on the **Whitelist Setting** tab. If the MaxCompute configuration is changed, the IP addresses of the ENIs may change. Therefore, we recommend that you add the CIDR block of the vSwitch to which the VPC belongs to the whitelist.

- Configure the security group of an ApsaraDB RDS instance.

  Add the security group that is created for MaxCompute to the security group of the ApsaraDB RDS instance or add the IP addresses of the ENIs that are created by MaxCompute to a whitelist of the ApsaraDB RDS instance.

For example, if you want to allow MaxCompute to access an ApsaraDB RDS instance, perform the following operations: Log on to the ApsaraDB RDS console. On the **Instances** page, click the name of the ApsaraDB RDS instance that you want to access in the Instance ID/Name column. In the left-side navigation pane, click **Data Security**. Then, add a security group on the **Security Group** tab or an IP whitelist on the **Whitelist Settings** tab. For more information about how to add a security group or add an IP address to a whitelist, see Configure a security group for an ApsaraDB RDS for MySQL instance or Configure an IP address whitelist for an ApsaraDB RDS for MySQL instance.

> **Note** If the MaxCompute configuration is changed, the IP addresses of the ENIs may change. We recommend that you add the CIDR block of the vSwitch to which the VPC belongs to the whitelist.

5. **Test network connectivity.**

   You can test the network connectivity by following the operations in Lakehouse of MaxCompute.

3.15.4. Access external tables

3.15.4.1. Create an OSS external table

MaxCompute allows you to create an Object Storage Service (OSS) external table in your MaxCompute project. The OSS external table maps to an OSS directory. You can use the OSS external table to access unstructured data in data files in the OSS directory or write data from your MaxCompute project to the OSS directory. This topic describes the syntax and parameters that are used to create an OSS external table. This topic also provides examples on how to create an OSS external table.

Context

OSS is a secure, cost-effective, and highly reliable cloud storage service that can store a large number of data files in any format. If you want to use MaxCompute to read data from an OSS directory or write data from a MaxCompute project to an OSS directory, you can create an OSS external table in your MaxCompute project. The OSS external table maps to the OSS directory.

You can create a partitioned table or a non-partitioned table as an OSS external table. The type of the table that you need to create varies based on the format of the OSS directory in which data files are stored. If data files are stored in a partitioned directory, you must create a partitioned table. Otherwise, you must create a non-partitioned table. For more information about how to read partition data, see Read OSS data stored in partitions.

Prerequisites

Before you create an OSS external table, make sure that the Alibaba Cloud account or RAM user that you use meets the following requirements:

- The Alibaba Cloud account or RAM user is granted access permissions on OSS.

  For more information about authorization, see STS authorization for OSS.

- The Alibaba Cloud account or RAM user is granted the CreateTable permission on your project.

  For more information about operation permissions on tables in a MaxCompute project, see Permissions.

Usage notes

When you use OSS external tables, take note of the following items:

- For data files in different formats, the statements that are used to create OSS external tables are different only in some parameters. We recommend that you carefully read the syntax that is used to create an OSS external table and the parameter descriptions. This helps you create an OSS external table that meets your business requirements. If an invalid parameter is configured when you create an OSS external table, you may fail to read OSS data or write data to OSS by using the OSS external table.
- OSS external tables record only the mappings between these tables and OSS directories. If you delete an OSS external table, data files in the OSS directory that is mapped to the table are not deleted.
- If a data file stored in OSS is an object of the Archive storage class, you must first restore the file. For more information about the restoration operation, see Restore objects.

Operation platforms

The following table lists platforms in which you can create OSS external tables.

Creation method	Platform
MaxCompute SQL statements	MaxCompute client
	Query editor
	ODPS SQL nodes in the DataWorks console
	SQL scripts in MaxCompute Studio
Web UI	SQL scripts in MaxCompute Studio
	DataWorks console

Syntax used to create an OSS external table

The following table describes the syntax that is used to create OSS external tables in different scenarios. For more information about syntax parameters and properties, see Reference: Parameters in the syntax, Reference: WITH SERDEPROPERTIES properties, and Reference: TBLPROPERTIES properties.

Scenario	Syntax	Supported format of OSS data files

Scenario	Syntax	Supported format of OSS data files

Create an external table by using a built-in text extractor	`create external table [if not exists]`
	`<mc_oss_extable_name>`
	`(`
	`<col_name> <data_type>,`
	`...`
	`)`
	`[partitioned by (<col_name> <data_type>, ...)]`
	`stored by '<StorageHandler>'`
	`with serdeproperties (`
	`['<property_name>'='<property_value>',...]`
	`)`
	`location '<oss_location>';`
		• CSV data files
		• CSV data files compressed in the GZIP format
		• TSV data files
		• TSV data files compressed in the GZIP format

Create an external table by using a built-in open source data extractor	`create external table [if not exists]`
	`<mc_oss_extable_name>`
	`(`
	`<col_name> <data_type>,`
	`...`
	`)`
	`[partitioned by (<col_name> <data_type>, ...)]`
	`[row format serde '<serde_class>'`
	`[with serdeproperties (`
	`['<property_name>'='<property_value>',...])`
	`]`
	`stored as <file_format>`
	`location '<oss_location>'`
	`[using '<resource_name>']`
	`[tblproperties`
	`('<tbproperty_name>'='<tbproperty_value>',...)];`
		• PARQUET data files
		• PARQUET data files compressed in the SNAPPY, GZIP, or LZO format
		• TEXTFILE (JSON or TEXT) data files
		• TEXTFILE data files compressed in the SNAPPY, LZO, BZ2, GZ, or DEFLATE format
		• ORC data files
		• ORC data files compressed in the SNAPPY or ZLIB format
		• AVRO data files
		• SEQUENCEFILE data files

Scenario	Syntax	Supported format of OSS data files
Create an external table by using a custom extractor	create external table [if not exists] <mc_oss_extable_name> (<col_name> <date_type>, ...) [partitioned by (<col_name> <data_type>, ...)] stored by '<StorageHandler>' with serdeproperties (['<property_name>'='<property_value>',...]) location '<oss_location>' using '<jar_name>';	Data files except the preceding formats

Syntax used to add partition data to OSS external tables

If the OSS external table that you created is a partitioned table, partition data needs to be added to the OSS external table. MaxCompute automatically adds partition data to the OSS external table based on the OSS directory that is specified when you create the OSS external table. Syntax:

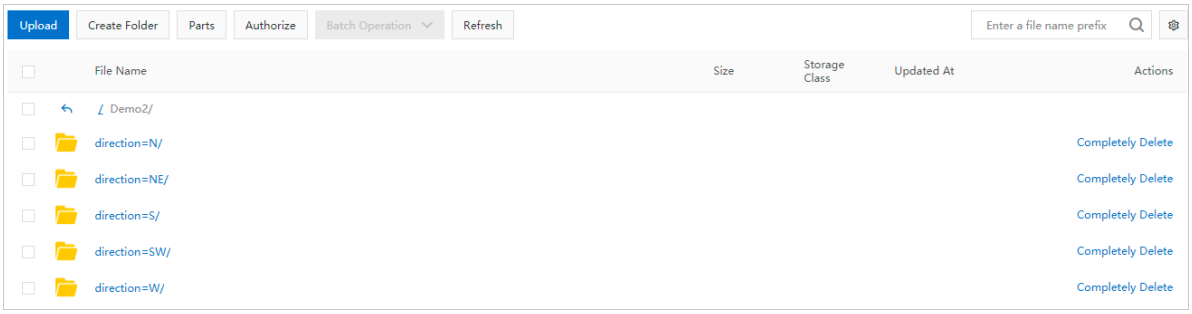
- Syntax 1 (recommended): After you execute the following statement, MaxCompute automatically parses the OSS directory structure, identifies partitions, and adds partition data to the OSS external table.

  ```
  msck repair table <mc_oss_extable_name> add partitions;
  ```

- Syntax 2: Execute the following statement to manually add partition data to the OSS external table.

  ```
  alter table <mc_oss_extable_name> add partition (<col_name>=<col_value>) [add partition (
  <col_name>=<col_value>) ...];
  ```

  The values of col_name and col_value must be consistent with the names of the OSS subdirectories where the partition data file is stored. If the OSS directory structure of the partition data file shown in the following figure is used, the value of col_name is `direction`, and the values of col_value are `N, NE, S, SW, and W`. One `add partition` clause in the ALTER TABLE statement is used to add a partition that is mapped to an OSS subdirectory. The number of `add partition` clauses must be the same as that of OSS subdirectories.



Example: Prepare data

Sample data is provided for you to better understand the examples in this topic.

- oss_endpoint: `oss-cn-hangzhou-internal.aliyuncs.com`, which indicates that OSS is deployed in the China (Hangzhou) region.
- Bucket name: `oss-mc-test`.
- Directory name: `Demo1/`, `Demo2/`, `Demo3/`, and `SampleData/`.

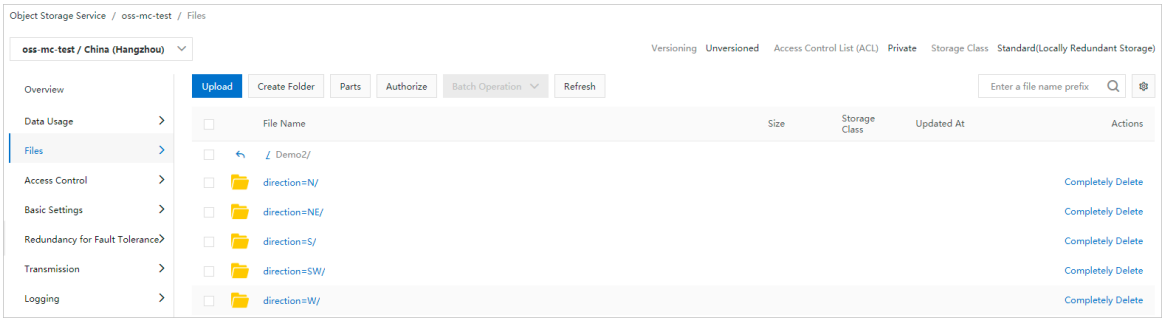
  The following data files are uploaded to the preceding directories:

  - The vehicle.csv file is uploaded to the `Demo1/` directory. The vehicle.csv file is used to create a mapping between the Demo1/ directory and the non-partitioned external table that is created by using a built-in text extractor. The file contains the following data:

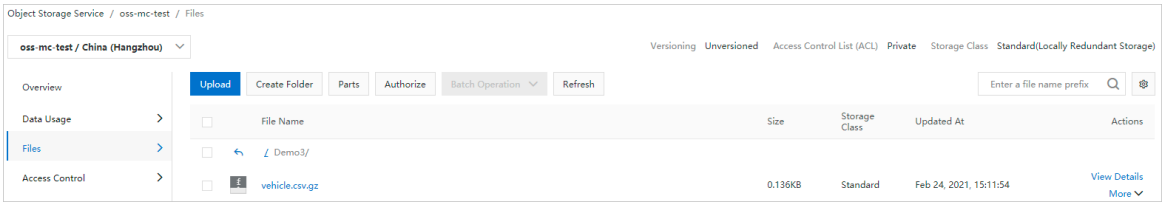
```
1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S
1,6,9,1,46.81006,-92.08174,9/15/2014 0:00,S
1,7,53,1,46.81006,-92.08174,9/15/2014 0:00,N
1,8,63,1,46.81006,-92.08174,9/15/2014 0:00,SW
1,9,4,1,46.81006,-92.08174,9/15/2014 0:00,NE
1,10,31,1,46.81006,-92.08174,9/15/2014 0:00,N
```

- The `Demo2/` directory contains the following subdirectories: `direction=N/`, `direction=NE/`, `direction=S/`, `direction=SW/`, and `direction=W/`. The vehicle1.csv, vehicle2.csv, vehicle3.csv, vehicle4.csv, and vehicle5.csv files are separately uploaded to the five subdirectories. The files are used to create mappings between the Demo2/ directory and the partitioned external table that is created by using a built-in text extractor. The files contain the following data:

```
--vehicle1.csv
1,7,53,1,46.81006,-92.08174,9/15/2014 0:00
1,10,31,1,46.81006,-92.08174,9/15/2014 0:00
--vehicle2.csv
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00
1,9,4,1,46.81006,-92.08174,9/15/2014 0:00
--vehicle3.csv
1,6,9,1,46.81006,-92.08174,9/15/2014 0:00
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00
1,6,9,1,46.81006,-92.08174,9/15/2014 0:00
--vehicle4.csv
1,8,63,1,46.81006,-92.08174,9/15/2014 0:00
--vehicle5.csv
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00
```



- The vehicle.csv.gz file is uploaded to the `Demo3/` directory. The vehicle.csv file is contained in the vehicle.csv.gz file and has the same content as the vehicle.csv file in the `Demo1/` directory. The vehicle.csv file is used to create a mapping between the Demo3/ directory and the OSS external table whose data is compressed.



- The vehicle6.csv file is uploaded to the `SampleData/` directory. The vehicle6.csv file is used to create a mapping between the SampleData/ directory and the OSS external table that is created by using a built-in open source data extractor. The file contains the following data:

```
1|1|51|1|46.81006|-92.08174|9/14/2014 0:00|S
1|2|13|1|46.81006|-92.08174|9/14/2014 0:00|NE
1|3|48|1|46.81006|-92.08174|9/14/2014 0:00|NE
1|4|30|1|46.81006|-92.08174|9/14/2014 0:00|W
1|5|47|1|46.81006|-92.08174|9/14/2014 0:00|S
1|6|9|1|46.81006|-92.08174|9/14/2014 0:00|S
1|7|53|1|46.81006|-92.08174|9/14/2014 0:00|N
1|8|63|1|46.81006|-92.08174|9/14/2014 0:00|SW
1|9|4|1|46.81006|-92.08174|9/14/2014 0:00|NE
1|10|31|1|46.81006|-92.08174|9/14/2014 0:00|N
```

Example: Create a non-partitioned table as an OSS external table by using a built-in text extractor

Create a mapping between the non-partitioned external table and the `Demo1/` directory in Example: Prepare data. Sample statement used to create an OSS external table:

```
create external table if not exists mc_oss_csv_external1
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
 'odps.properties.rolearn'='acs:ram::xxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo1/';
```

You can execute the `DESC EXTENDED mc_oss_csv_external1;` statement to view the schema of the OSS external table that you created.

Example: Create a partitioned table as an OSS external table by using a built-in text extractor

Create a mapping between the partitioned external table and the `Demo2/` directory in Example: Prepare data. Sample statements used to create an OSS external table and add partition data to the OSS external table:

```
create external table if not exists mc_oss_csv_external2
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string
)
partitioned by (
direction string
)
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
 'odps.properties.rolearn'='acs:ram::xxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo2/';
-- Add partition data.
msck repair table mc_oss_csv_external2 add partitions;
-- The preceding statement is equivalent to the following statement:
alter table mc_oss_csv_external2 add partition (direction = 'N') partition (direction = 'NE
') partition (direction = 'S') partition (direction = 'SW') partition (direction = 'W');
```

You can execute the `DESC EXTENDED mc_oss_csv_external2;` statement to view the schema of the OSS external table that you created.

Example: Create an OSS external table that is used to read or write compressed data by using a built-in text extractor

Create a mapping between the OSS external table and the `Demo3/` directory in Example: Prepare data. Sample statement used to create an OSS external table:

```
create external table if not exists mc_oss_csv_external3
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
 'odps.properties.rolearn'='acs:ram::xxxxxx:role/aliyunodpsdefaultrole',
 'odps.text.option.gzip.input.enabled'='true',
 'odps.text.option.gzip.output.enabled'='true'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo3/';
```

You can execute the `DESC EXTENDED mc_oss_csv_external3;` statement to view the schema of the OSS external table that you created.

Example: Create an OSS external table by using a built-in open source data extractor

- PARQUET data files are mapped to the OSS external table.

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as parquet
location '<oss_location>';
select ... from <mc_oss_extable_name> ...;
```

- TEXTFILE data files are mapped to the OSS external table.
  - **Example of creating an OSS external table that maps to TEXT data files**

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
row format serde 'org.apache.hive.hcatalog.data.JsonSerDe'
stored as textfile
location '<oss_location>';
select ... from <mc_oss_extable_name> ...;
```

CREATE EXTERNAL TABLE statements do not support custom `row format` settings. Default `row format` settings:

```
FIELDS TERMINATED BY: '\001'
ESCAPED BY: '\'
COLLECTION ITEMS TERMINATED BY: '\002'
MAP KEYS TERMINATED BY: '\003'
LINES TERMINATED BY: '\n'
NULL DEFINED AS: '\N'
```

- **Example of creating an OSS external table that maps to CSV data files**

```
-- Disable the native text reader.
set odps.ext.hive.lazy.simple.serde.native=false;
-- Create an OSS external table.
create external table <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
with serdeproperties (
"separatorChar" = ",",
"quoteChar"= '"',
"escapeChar"= "\\"
)
stored as textfile
location '<oss_location>'
tblproperties (
"skip.header.line.count"="1",
"skip.footer.line.count"="1"
);
select ... from <mc_oss_extable_name> ...;
```

> **Note** OpenCSVSerde supports only data of the STRING type. When you execute a Data Manipulation Language (DML) statement, you must add the `set odps.sql.hive.compatible=true;` command before the DML statement and submit this command with the statement.

- **Example of creating an OSS external table that maps to JSON data files**

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
row format serde 'org.apache.hive.hcatalog.data.JsonSerDe'
stored as textfile
location '<oss_location>';
select ... from <mc_oss_extable_name> ...;
```

- ORC data files are mapped to the OSS external table.

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as orc
location '<oss_location>';
select ... from <mc_oss_extable_name> ...;
```

- AVRO data files are mapped to the OSS external table.

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as avro
location '<oss_location>';
select ... from <mc_oss_extable_name> ...;
```

- SEQUENCEFILE data files are mapped to the OSS external table.

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as sequencefile
location '<oss_location>';
select ... from <mc_oss_extable_name> ...;
```

Example: Create an OSS external table by using a custom extractor

Create a mapping between the OSS external table and the `SampleData/` directory in Example: Prepare data.

Perform the following steps:

1. Use the Java programs TextExtractor.java, TextOutputer.java, SplitReader.java, and TextStorageHandler.java that are developed by using MaxCompute Studio.

   For more information about how to develop a Java program, see Develop a UDF.

2. MaxCompute Studio allows you to package the TextStorageHandler.java program into a JAR file and upload the file to your MaxCompute project as a resource.

   In this example, the JAR file is named `javatest-1.0-SNAPSHOT.jar`. For more information about how to package resources into files and upload the files to your MaxCompute project, see Package a Java program, upload the package, and create a MaxCompute UDF.

3. Execute the following statement to create an OSS external table:

```
create external table if not exists ambulance_data_txt_external
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.udf.example.text.TextStorageHandler'
  with serdeproperties (
'delimiter'='|',
'odps.properties.rolearn'='acs:ram::xxxxxxxxxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/SampleData/'
using 'javatest-1.0-SNAPSHOT.jar';
```

delimiter indicates the delimiter of OSS data.

You can execute the `DESC EXTENDED ambulance_data_txt_external;` statement to view the schema of the OSS external table that you created.

Reference: Parameters in the syntax

This section describes the parameters in the syntax.

Parameter	Required	Description
mc_oss_extable_name	Yes	The name of the OSS external table that you want to create. Table names are not case-sensitive. When you query external tables, the table names are not case-sensitive, and forced uppercase and lowercase conversions are not supported.
col_name	Yes	The name of a column in the OSS external table. Before you read OSS data, make sure that the schema of the OSS external table is the same as the schema of the OSS data files. Otherwise, OSS data cannot be read.
data_type	Yes	The data type of a column in the OSS external table. Before you read OSS data, make sure that the data types of columns in the OSS external table are the same as the data types of columns in the OSS data files. Otherwise, OSS data cannot be read.

Parameter	Required	Description
partitioned by (<col_name> <data_type>, ...)	Required if specific conditions are met	If data files in OSS are stored in partitioned directories, this parameter is required to create a partitioned external table. • col_name: The name of a partition key column. • data_type: The data type of a partition key column.

StorageHandler	Required if specific conditions are met	If you use a built-in text extractor or a custom extractor to create an OSS external table, this parameter is required. MaxCompute extractors are categorized into the following types:
		• Built-in text extractors:
		  ○ `com.aliyun.odps.CsvStorageHandler`: defines how to read data from and write data to CSV files or CSV.GZ files.
		  ○ `com.aliyun.odps.TsvStorageHandler`: defines how to read data from and write data to TSV files or TSV.GZ files.
		• Custom extractors: You can write MaxCompute user defined functions (UDFs) to create a custom extractor. For more information about how to write MaxCompute UDFs, see Develop a UDF.

'<property_name>'='<property_value>'	Yes	The extended property of the external table. For more information the properties, see Reference: WITH SERDEPROPERTIES properties.

Parameter	Required	Description
oss_location	Yes	The OSS directory where data files are stored. The OSS directory is in the `oss://<oss_endpoint>/<Bucket name>/<Directory name>/` format. MaxCompute automatically reads data from all files in the OSS directory that you specified. • oss_endpoint: the OSS endpoint. We recommend that you use an internal endpoint of OSS to prevent extra fees that are incurred by Internet traffic. For more information about the internal endpoints of OSS, see Regions and endpoints. **Note** We recommend that OSS for storing data files is deployed in the same region as your MaxCompute project. MaxCompute can be deployed only in some regions. Therefore, cross-region data connectivity issues may occur. • Bucket name: the name of the OSS bucket. For more information about how to view bucket names, see List buckets. • Directory name: the name of the OSS directory. You do not need to include file names in directory names. `-- Example of a valid OSS directory: oss://oss-cn-shanghai-internal.aliyuncs.com/oss-mc-test/Demo1/ -- Examples of invalid OSS directories: http://oss-cn-shanghai-internal.aliyuncs.com/oss-mc-test/Demo1/ -- HTTP connections are not supported. https://oss-cn-shanghai-internal.aliyuncs.com/oss-mc-test/Demo1/ -- HTTPS connections are not supported. oss://oss-cn-shanghai-internal.aliyuncs.com/Demo1 -- The endpoint is invalid. oss://oss-cn-shanghai-internal.aliyuncs.com/oss-mc-test/Demo1/vehicle.csv -- The file name does not need to be included in the directory name.`

Parameter	Required	Description
jar_name	Required if specific conditions are met	If you use a custom extractor to create an OSS external table, this parameter is required. This parameter specifies the JAR package that corresponds to the custom extractor code. The JAR package must be added to your MaxCompute project as a resource. For more information about how to add resources, see Add resources.
serde_class	Required if specific conditions are met	This parameter specifies the built-in open source data extractor of MaxCompute. If the data file is in the TEXTFILE format, this parameter is required. You do not need to configure this parameter in other scenarios. Mappings between open source data formats that are supported by MaxCompute and SerDe classes: • PARQUET: `org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`. • TEXTFILE: `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`, `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`, and `org.apache.hadoop.hive.serde2.OpenCSVSerde`. • ORC: `org.apache.hadoop.hive.ql.io.orc.OrcSerde`. • AVRO: `org.apache.hadoop.hive.serde2.avro.AvroSerDe`. • SEQUENCEFILE: `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`. • RCFILE: `org.apache.hadoop.hive.serde2.columnar.LazyBinaryColumnarSerDe`. • ORCFILE: `org.apache.hadoop.hive.ql.io.orc.OrcSerde`.
file_format	Required if specific conditions are met	If OSS data files are in an open source format, this parameter is required. This parameter specifies the format of the OSS data files. **Note** The size of a single file cannot exceed 3 GB. If the file size exceeds 3 GB, we recommend that you split the file into multiple files.

Parameter	Required	Description
resource_name	Required if specific conditions are met	If you use a custom SerDe class, you must use this parameter to specify the resource that contains the custom SerDe class. JAR packages that are related to the SerDe class must be added to your MaxCompute project as resources. For more information about how to add resources, see Add resources.
'<tbproperty_name>'='<tbproperty_value>'	Required if specific conditions are met	The extended property of the external table. For more information the properties, see Reference: TBLPROPERTIES properties.

Reference: WITH SERDEPROPERTIES properties

property_name	Scenario	Description	property_value	Default value
odps.properties.rolearn	If you use Security Token Service (STS) authorization, add this property.	Specifies the Alibaba Cloud Resource Name (ARN) of AliyunODPSDefaultRole in Resource Access Management (RAM).	You can obtain the ARN from the role details page in the RAM console.	None
delimiter	If you need to specify column delimiters for a CSV or TSV data file, add this property.	MaxCompute normally reads data from each column in an OSS data file based on the specified delimiters.	A single character	Comma (,)
odps.text.option.gzip.input.enabled	If you need to read data from a CSV.GZ or TSV.GZ data file, add this property.	**CSV or TSV compression properties**. If you set this parameter to True, MaxCompute normally reads data from CSV.GZ or TSV.GZ data files. Otherwise, the data read operation fails.	• True • False	False
odps.text.option.gzip.output.enabled	If you need to write data to OSS by using the GZIP compression algorithm, add this property.	**CSV or TSV compression properties**. If you set this parameter to True, MaxCompute normally writes data to OSS by using the GZIP compression algorithm. Otherwise, the data is not compressed.	• True • False	False

property_name	Scenario	Description	property_value	Default value
odps.text.option.header.lines.count	If the OSS data file is in the CSV or TSV format and you need to ignore the first N rows in the OSS data file, add this property.	Specifies the number of rows that are ignored when you use MaxCompute to read OSS data files.	Non-negative integer	0
odps.text.option.null.indicator	If the OSS data file is in the CSV or TSV format and you need to define parsing rules of NULL in the OSS data file, add this property.	Specifies the string that is parsed as NULL in an SQL statement. For example, if you use `\N` to represent NULL, `a,\N,b` is parsed as `a, NULL, b`.	String	Empty string
odps.text.option.ignore.empty.lines	If the OSS data file is in the CSV or TSV format and you need to define processing rules of empty rows in the OSS data file, add this property.	If you set this parameter to True, MaxCompute reads all rows in the data file. Otherwise, MaxCompute reads the empty rows.	• True • False	True
odps.text.option.encoding	If the OSS data file is in the CSV or TSV format and the encoding format of the OSS data file is not the default encoding format, add this property.	Make sure that the encoding format that is configured in this scenario is the same as the encoding format of the OSS data file. Otherwise, MaxCompute cannot read data successfully.	• UTF-8 • UTF-16 • US-ASCII • GBK	UTF-8
odps.text.option.delimiter	If you need to specify column delimiters for a CSV or TSV data file, add this property.	Make sure that the column delimiters specified in this scenario can be used to normally read each column of the OSS data file. Otherwise, the data read by MaxCompute is misaligned.	A single character	Comma (,)

property_name	Scenario	Description	property_value	Default value
odps.text.option.use.quote	If fields in a CSV or TSV data file contain carriage return, line feed (CRLF) pairs, double quotation marks ("), or commas (,), add this property.	Specifies whether to recognize column delimiters in a CSV file if it uses double quotation marks (") as column delimiters. If fields in the CSV file contain specified symbols that are used to separate multiple values, these fields must be enclosed in double quotation marks (`"`). The symbols include CRLF pairs, double quotation marks ("), and commas (,). If a field contains a double quotation mark (`"`), replace the double quotation mark (") with two double quotation marks ("") for escaping.	• True • False	False
mcfed.parquet.compression	If you need to write data in a PARQUET data file to OSS by using a compression algorithm, add this property.	**PARQUET compression property**. By default, data in the PARQUET data file is not compressed. If you specify a compression algorithm, you must add the `set odps.sql.hive.compatible=true;` command before the statement that is used to create an OSS external table and submit this command with the statement.	• SNAPPY • GZIP	None
parquet.file.cache.size	If you need to improve the performance of reading data in a PARQUET data file, add this property.	Specifies the maximum amount of data that can be cached when you read OSS data files. Unit: KB.	1024	None
parquet.io.buffer.size	If you need to improve the performance of reading data in a PARQUET data file, add this property.	Specifies the maximum amount of data that can be cached when the size of the OSS data file exceeds 1,024 KB. Unit: KB.	4096	None
separatorChar	If you need to specify column delimiters for a CSV data file in the TEXTFILE format, add this property.	Specifies the column delimiters for a CSV data file.	A single string	Comma (,)

property_name	Scenario	Description	property_value	Default value
quoteChar	If fields in a CSV data file in the TEXTFILE format contain CRLF pairs, double quotation marks ("), or commas (,), add this property.	Specifies the quote for a CSV data file.	A single string	None
escapeChar	If you need to specify the rules that are used to escape a CSV data file in the TEXTFILE format, add this property.	Specifies escape characters for a CSV data file.	A single string	None

Reference: TBLPROPERTIES properties

property_name	Scenario	Description	property_value	Default value
skip.header.line.count	If you need to ignore the first N rows in a CSV data file in the TEXTFILE format, add this property.	If you use MaxCompute to read OSS data, the data in the specified number of rows that start from the first row is ignored.	Non-negative integer	None
skip.footer.line.count	If you need to ignore the last N rows in a CSV data file in the TEXTFILE format, add this property.	If you use MaxCompute to read OSS data, the data in the specified number of rows that start from the last row is ignored.	Non-negative integer	None
mcfed.orc.compress	If you need to write data in an ORC data file to OSS by using a compression algorithm, add this property.	**ORC compression property**. Specifies the compression algorithm of ORC data files. If you specify a compression algorithm, you must add the `set odps.sql.hive.compatible=true;` command before the statement that is used to create an OSS external table and submit this command with the statement.	• SNAPPY • ZLIB	None

property_name	Scenario	Description	property_value	Default value
mcfed.mapreduce.output.fileoutputformat.compress	If you need to write data in a TEXTFILE data file to OSS by using a compression algorithm, add this property.	**TEXTFILE compression property**. If you set this parameter to True, MaxCompute writes data in the TEXTFILE data files to OSS by using a compression algorithm. Otherwise, data is not compressed. If you specify a compression algorithm, you must add the `set odps.sql.hive.compatible=true;` command before the statement that is used to create an OSS external table and submit this command with the statement.	• True • False	False
mcfed.mapreduce.output.fileoutputformat.compress.codec	If you need to write data in a TEXTFILE data file to OSS by using a compression algorithm, add this property.	**TEXTFILE compression property**. Specifies the compression algorithm of the TEXTFILE data files. If you specify a compression algorithm, you must add the `set odps.sql.hive.compatible=true;` command before the statement that is used to create an OSS external table and submit this command with the statement.	• com.hadoop.compression.lzo.LzoCodec • com.hadoop.compression.lzo.LzopCodec • org.apache.hadoop.io.compress.SnappyCodec • com.aliyun.odps.io.compress.SnappyRawCodec	None
io.compression.codecs	If the OSS data file is in the raw Snappy format, add this property.	If you set this parameter to True, MaxCompute normally reads data from compressed files. Otherwise, the data read operation fails.	com.aliyun.odps.io.compress.SnappyRawCodec	None

3.15.4.2. Read OSS data

After you create an Object Storage Service (OSS) external table, you can use the external table to read the data that is stored in an OSS directory.

Context

After you create an OSS external table, you can use one of the following methods to read OSS data based on your business requirements:

- Method 1 (recommended): Import OSS data in an open source format from the OSS external table to

a MaxCompute internal table before you read the data.

If you directly read data from the OSS external table, each read operation involves OSS I/O operations and high-performance optimizations on MaxCompute storage cannot be implemented. As a result, the data reading performance is not high. If you require repeated data computing or high computing efficiency, we recommend that you perform the following operations in sequence to improve the computing performance of MaxCompute: 1. Create a MaxCompute internal table that has the same schema as the OSS external table. 2. Import OSS data from the OSS external table to the MaxCompute internal table. 3. Execute complex query statements for the MaxCompute internal table.

Sample statements:

```
create table <table_internal> like <mc_oss_extable_name>;
insert overwrite table <table_internal> select * from <mc_oss_extable_name>;
```

- Method 2: Directly read OSS data from the OSS external table. The operations are the same as those for reading data from a MaxCompute internal table.

  If you use this method, the OSS external table is used in the same way as a MaxCompute internal table, but data is directly read from an OSS directory.

Usage notes

When you use an external table to read OSS data, take note of the following items:

- You must create an OSS external table before you read OSS data from the external table. For more information about the formats of the OSS data files that are supported by MaxCompute and the commands that are used to create OSS external tables, see Syntax used to create an OSS external table.
- If complex data types are involved in the SQL statement that you want to execute, you must add the `set odps.sql.type.system.odps2=true;` command before the SQL statement and submit this command with the statement. For more information about data types, see Data type editions.
- If you use an OSS external table that maps to open source data, you must set `odps.sql.hive.compatible` to true at the session level before you use MaxCompute to read data from the OSS external table. Otherwise, an error is returned.

Example: Read OSS data from a non-partitioned OSS external table that is created by using a built-in text extractor

Read OSS data from the OSS external table mc_oss_csv_external1 that is created in Example: Create a non-partitioned table as an OSS external table by using a built-in text extractor.

Use the MaxCompute client to read data from the OSS external table. Sample statement:

```
select recordId, patientId, direction from mc_oss_csv_external1 where patientId > 25;
```

The following result is returned:

```
+------------+------------+------------+
 recordid    patientid   direction
+------------+------------+------------+
 1           51          S
 3           48          NE
 4           30          W
 5           47          S
 7           53          N
 8           63          SW
 10          31          N
+------------+------------+------------+
```

Example: Read OSS data from a partitioned OSS external table that is created by using a built-in text extractor

Read OSS data from the OSS external table mc_oss_csv_external2 that is created in Example: Create a partitioned table as an OSS external table by using a built-in text extractor.

Use the MaxCompute client to read data from the OSS external table. Sample statement:

```
select recordId, patientId, direction from mc_oss_csv_external2 where direction = 'NE';
```

The following result is returned:

```
+------------+------------+------------+
 recordid    patientid   direction
+------------+------------+------------+
 2           13          NE
 3           48          NE
 9           4           NE
+------------+------------+------------+
```

Example: Read OSS compressed data from an OSS external table that is created by using a built-in text extractor

Read OSS data from the OSS external table mc_oss_csv_external3 that is created in Example: Create an OSS external table that is used to read or write compressed data by using a built-in text extractor.

Use the MaxCompute client to read data from the OSS external table. Sample statement:

```
select recordId, patientId, direction from mc_oss_csv_external3 where patientId > 25;
```

Note If OSS compressed data is in an open source format, you must add the `set odps.sql.hive.compatible=true;` command before the SQL statement and submit this command with the statement.

The following result is returned:

```
+------------+------------+------------+
 recordid    patientid   direction
+------------+------------+------------+
 1           51          S
 3           48          NE
 4           30          W
 5           47          S
 7           53          N
 8           63          SW
 10          31          N
+------------+------------+------------+
```

Example: Read OSS text data from an OSS external table that is created by using a custom extractor

Read OSS data from the OSS external table ambulance_data_txt_external that is created in Example: Create an OSS external table by using a custom extractor.

Use the MaxCompute client to read data from the OSS external table. Sample statement:

```
select recordId, patientId, direction from ambulance_data_txt_external where patientId > 25
;
```

The following result is returned:

```
+----------+----------+----------+
 recordid  patientid  direction
+----------+----------+----------+
 1         51         S
 3         48         NE
 4         30         W
 5         47         S
 7         53         N
 8         63         SW
 10        31         N
+----------+----------+----------+
```

3.15.4.3. Use a custom extractor to access OSS

If the data format in Object Storage Service (OSS) is complex and the built-in extractor cannot meet your business requirements, you can use a custom extractor to read unstructured data, such as the text data and non-text data, from OSS. This topic describes how to define a custom extractor and use the custom extractor to create an OSS external table to read data from OSS.

Prerequisites

Limits

- A custom extractor that is defined by using the method described in this topic cannot access data of the DATETIME type in OSS text files. For more information about how to define a custom extractor to access data of the DATETIME type in OSS text files, see Access data of the DATETIME type in OSS text files by using a MaxCompute custom extractor.

-
-

Syntax used to create an external table

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <date_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
-- Specify a custom extractor.
stored by '<StorageHandler>'
-- Specify the parameters that are related to the OSS external table.
with serdeproperties (
 'delimiter'='<delimiter>',
 'odps.properties.rolearn'='<ram_arn>'
-- Set odps.text.option.gzip.input.enabled to true if OSS files are in the GZIP format.
 [,'odps.text.option.gzip.input.enabled'='true']
-- Specify the properties that are related to the OSS external table based on your business
requirements.
 [,'<property_name>'='<property_value>'[,'<property_name>'='<property_value>'...]]
)
location '<oss_location>'
using '<jar_name>';
```

-
-
-
-
-
- StorageHandler: required. Specifies the class name of the custom storage handler.
-
-
-
- <property_name>'='<property_value>': optional. property_name specifies the name of a property and property_value specifies the value of a property. For more information about properties, see Create an OSS external table.
- USING: specifies the JAR package that contains the class definition of the custom extractor.

Use a custom extractor to access text data in OSS

The OSS text file, vehicle.csv, contains the following data. The columns in the file are separated by vertical bars (`|`). The vehicle.csv file is stored in the `/demo/SampleData/CustomTxt/AmbulanceData/` directory.

```
1|1|51|1|46.81006|-92.08174|9/14/2014 0:00|S
1|2|13|1|46.81006|-92.08174|9/14/2014 0:00|NE
1|3|48|1|46.81006|-92.08174|9/14/2014 0:00|NE
1|4|30|1|46.81006|-92.08174|9/14/2014 0:00|W
1|5|47|1|46.81006|-92.08174|9/14/2014 0:00|S
1|6|9|1|46.81006|-92.08174|9/14/2014 0:00|S
1|7|53|1|46.81006|-92.08174|9/14/2014 0:00|N
1|8|63|1|46.81006|-92.08174|9/14/2014 0:00|SW
1|9|4|1|46.81006|-92.08174|9/14/2014 0:00|NE
1|10|31|1|46.81006|-92.08174|9/14/2014 0:00|N
```

Procedure:

1. Use IntelliJ IDEA to define the class of an extractor.

   Define a common extractor and use a delimiter as a parameter. The extractor can process all text files that have a similar format. Sample code of the extractor:

```
/**
 * Text extractor that extract schematized records from formatted plain-text(csv, tsv e
tc.)
 **/
public class TextExtractor extends Extractor {
  private InputStreamSet inputs;
  private String columnDelimiter;
  private DataAttributes attributes;
  private BufferedReader currentReader;
  private boolean firstRead = true;
  public TextExtractor() {
    // default to ",", this can be overwritten if a specific delimiter is provided (via
DataAttributes)
    this.columnDelimiter = ",";
  }
  // no particular usage for execution context in this example
  @Override
  public void setup(ExecutionContext ctx, InputStreamSet inputs, DataAttributes attribu
tes) {
    this.inputs = inputs; // inputs specifies an InputStreamSet. An InputStream is retu
rned every time the next() function is called. This InputStream can read all the data o
f an OSS file.
    this.attributes = attributes;
    // check if "delimiter" attribute is supplied via SQL query
    String columnDelimiter = this.attributes.getValueByKey("delimiter"); // The delimit
er can be used as a parameter in Data Definition Language (DDL) statements.
    if ( columnDelimiter != null)
    {
      this.columnDelimiter = columnDelimiter;
    }
    // note: more properties can be inited from attributes if needed
  }
  @Override
  public Record extract() throws IOException {// The extract() function returns a recor
d that is extracted from the external table.
    String line = readNextLine();
    if (line == null) {
      return null; // If null is returned, all records in the OSS external table have b
een read.
    }
    return textLineToRecord(line); // textLineToRecord splits a row into multiple colum
ns by using the delimiter.
  }
  @Override
  public void close(){
    // no-op
  }
}
```

Note For more information about how textLineToRecord splits a row into multiple columns, see TextExtractor.java.

2. Use IntelliJ IDEA to define the class of a storage handler.

   A storage handler is a unified entry point that can be used to run custom logic for processing external tables. Sample code:

```
package com.aliyun.odps.udf.example.text;
public class TextStorageHandler extends OdpsStorageHandler {
  @Override
  public Class<? extends Extractor> getExtractorClass() {
    return TextExtractor.class;
  }
  @Override
  public Class<? extends Outputer> getOutputerClass() {
    return TextOutputer.class;
  }
}
```

3. Compile the code for the preceding classes of the extractor and storage handler by using IntelliJ IDEA, package the code into a JAR file, and then add the file as a MaxCompute resource on the MaxCompute client.

   Sample command used to add the JAR file:

```
add jar odps-udf-example.jar;
```

4. Create an external table on the MaxCompute client.

   In the following sample command, delimiter indicates the delimiter of OSS data.

```
create external table if not exists ambulance_data_txt_external
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.udf.example.text.TextStorageHandler'
  with serdeproperties (
'delimiter'='\\|',
'odps.properties.rolearn'='acs:ram::xxxxxxxxxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/Demo/SampleData/Cus
tomTxt/AmbulanceData/'
using 'odps-udf-example.jar';
```

5. On the MaxCompute client, execute the following SQL statement to query data from the external table:

```
select recordId, patientId, direction from ambulance_data_txt_external where patientId
> 25;
```

Use a custom extractor to access non-text data in OSS

This section describes how to use a custom extractor to access and process a WAV file that contains audio data.

Procedure:

1. Define the classes of the custom extractor and storage handler, compile the code for these classes, and then package the code into a JAR file. For more information about the logic implementation, see SpeechSentenceSnrExtractor.
2. On the MaxCompute client, execute the following SQL statement to create an external table:

```
create external table if not exists speech_sentence_snr_external
(
sentence_snr double,
id string
)
stored by 'com.aliyun.odps.udf.example.speech.SpeechStorageHandler'
with serdeproperties (
    'mlfFileName'='sm_random_5_utterance.text.label' ,
    'speechSampleRateInKHz' = '16'
)
location 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/dev/SpeechSentenceT
est/'
using 'odps-udf-example.jar,sm_random_5_utterance.text.label';
```

   - com.aliyun.odps.udf.example.speech.SpeechStorageHandler: encapsulates an extractor to calculate the average signal-to-noise ratio (SNR) of valid sentences in each WAV file, and runs SQL jobs to process the structured data that is extracted.
   - location: specifies the directory in which WAV files are stored. For example, `oss://oss-cn-hangzhou-zmf.aliyuncs.com/oss-odps-test/dev/SpeechSentenceTest/` stores multiple WAV files. MaxCompute reads all the files in this directory, splits files based on the file level, and then allocates these files to multiple compute nodes for data processing.
3. On the MaxCompute client, execute the following SQL statement to query the results:

```
select sentence_snr, id from speech_sentence_snr_external where sentence_snr > 10.0;
```

   The following result is returned:

```
--------------------------------------------------------------
| sentence_snr |                     id                      |
--------------------------------------------------------------
|   34.4703    |          J310209090013_H02_K03_042          |
--------------------------------------------------------------
|   31.3905    | tsh148_seg_2_3013_3_6_48_80bd359827e24dd7_0 |
--------------------------------------------------------------
|   35.4774    | tsh148_seg_3013_1_31_11_9d7c87aef9f3e559_0  |
--------------------------------------------------------------
|   16.0462    | tsh148_seg_3013_2_29_49_f4cb0990a6b4060c_0  |
--------------------------------------------------------------
|   14.5568    |   tsh_148_3013_5_13_47_3d5008d792408f81_0   |
--------------------------------------------------------------
```

A custom extractor allows you to execute SQL statements to process multiple audio files in OSS in a distributed manner. You can also process other unstructured data, such as images and videos, in the same way to make full use of the powerful computing capabilities of MaxCompute.

3.15.4.4. Open source data formats supported by OSS external tables

If Object Storage Service (OSS) data is in an open source format, you cannot use a built-in extractor or custom extractor of MaxCompute to access the OSS data. You must specify an open source data format that is supported by an OSS external table when you create this table. This topic describes how to create an OSS external table and use the table to associate, read, and process OSS data in an open source format.

Prerequisites

Context

MaxCompute uses an unstructured data framework to associate OSS external tables with OSS data in an open source format. The unstructured framework calls the implementation method that is provided by an open source community to parse OSS data in an open source format. The method can be seamlessly integrated with MaxCompute.

This method is similar to the method that uses a built-in extractor to access OSS data. However, the two methods use different syntax to create OSS external tables. This topic describes the syntax for creating OSS external tables. For more information about the limits, precautions, procedures, and examples of creating OSS external tables, see Create an OSS external table.

The open source data formats that are supported by MaxCompute are PARQUET, TEXT, CSV, JSON, ORC, AVRO, and SEQUENCEFILE.

> Note

Syntax used to create an external table

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
[row format serde '<serde_class>'
  [with serdeproperties (
    'odps.properties.rolearn'='<ram_arn>'
    [,'<property_name>'='<property_value>'[,'<property_name>'='<property_value>'...]])
]
-- Configurations that are required for reading OSS data in an open source format.
stored as <file_format>
location '<oss_location>'
[using '<resource_name>'];
```

-

-
-
-
-
- serde_class: optional. This parameter is required only when OSS data is stored in a special format, such as a TEXTFILE format. The TEXTFILE format can be CSV, JSON, or TEXT. Mappings between open source data formats and SerDe classes:
  - PARQUET: `org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`
  - TEXTFILE: `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`
  - ORC: `org.apache.hadoop.hive.ql.io.orc.OrcSerde`
  - AVRO: `org.apache.hadoop.hive.serde2.avro.AvroSerDe`
  - SEQUENCEFILE: `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`
  - RCFILE: `org.apache.hadoop.hive.serde2.columnar.LazyBinaryColumnarSerDe`
  - ORCFILE: `org.apache.hadoop.hive.ql.io.orc.OrcSerde`
- odps.properties.rolearn'='<ram_arn>: optional. The Alibaba Cloud Resource Name (ARN) of AliyunODPSDefaultRole in Resource Access Management (RAM). You can obtain the ARN from the role details page in the RAM console.
- '<property_name>'='<property_value>': optional. property_name specifies the name of a property. property_value specifies the value of a property. For more information about external table properties, see the examples that are provided in this topic.
- file_format: required. The open source format of OSS data, such as ORC or PARQUET.

  > **Note** The size of a single file cannot exceed 3 GB. If the file size exceeds 3 GB, we recommend that you split the file into multiple small files.

-
- using: optional. If you use a custom SerDe class, you need to use this parameter to specify the dependent resource. The resource information includes the custom SerDe class. For more information about how to upload resources, see Resource operations.

Example of creating an OSS external table that is associated with OSS data in the PARQUET format

Syntax:

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
stored as parquet
location '<oss_location>';
```

By default, OSS data in the PARQUET format is not compressed. To compress the data in the PARQUET format in MaxCompute, you must add the `set odps.sql.hive.compatible=true;` command before the CREATE EXTERNAL TABLE statement, enable data compression, and then submit this command with the CREATE EXTERNAL TABLE statement. Supported compression formats:

- `Snappy`: To use the Snappy compression format, add `'mcfed.parquet.compression'='SNAPPY'` to `WITH SERDEPROPERTIES` in the CREATE EXTERNAL TABLE statement.
- `GZIP`: To use the GZIP compression format, add `'mcfed.parquet.compression'='GZIP'` to `WITH SERDEPROPERTIES` in the CREATE EXTERNAL TABLE statement.

Example of creating an OSS external table that is associated with OSS data in a TEXTFILE format

If OSS data is in a TEXTFILE format, such as CSV, JSON, or TEXT, and you want to use OSS external tables to read, write, and compress the data, you can use the OSS external tables that are provided by MaxCompute. This allows you to read data from and write data to the files whose compression format is SNAPPY or LZO. For more information, see Compression data formats supported by OSS external tables.

- **Example of creating an OSS external table that is associated with OSS data in the TEXT format**
  - If OSS data is in the JSON format and stored as text files in multiple OSS directories and the file storage and naming convention comply with the specified rules, you can create a partitioned OSS external table and associate the table with the OSS data. Syntax used to create a partitioned OSS external table:

    ```
    create external table [if not exists] <mc_oss_extable_name>
    (
    <col_name> <data_type>,
    ...
    )
    partitioned by (<col_name> <data_type>, ...)
    row format serde 'org.apache.hive.hcatalog.data.JsonSerDe'
    stored as textfile
    location '<oss_location>';
    ```

  - If the OSS subdirectories that are mapped to the partitions of the partitioned OSS external table are named in the format of partition names, you must add partition data to the table. For more information about how to add partition data to OSS external tables, see Create an OSS external table.

- If the OSS subdirectories that are mapped to the partitions of the partitioned OSS external table are not named in the format of partition names or are not under the directory when you create the table, you can execute the ALTER TABLE statement to add partition data. Sample OSS directories:

  ```
  oss://oss-cn-hangzhou-internal.aliyuncs.com/bucket/text_data_20170102/;
  oss://oss-cn-hangzhou-internal.aliyuncs.com/bucket/text_data_20170103/;
  ...
  ```

  You can execute the following statements to add partition data to the preceding directories:

  ```
  alter table tpch_lineitem_textfile add partition(ds="20170102")
  location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/bucket/text_data_20170102/';
  alter table tpch_lineitem_textfile add partition(ds="20170103")
  location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/bucket/text_data_20170103/';
  ...
  ```

- CREATE EXTERNAL TABLE statements do not support custom `row format` settings. Default `row format` settings:

  ```
  FIELDS TERMINATED BY: '\001'
  ESCAPED BY: '\'
  COLLECTION ITEMS TERMINATED BY: '\002'
  MAP KEYS TERMINATED BY: '\003'
  LINES TERMINATED BY: '\n'
  NULL DEFINED AS: '\N'
  ```

- **Example of creating an OSS external table that is associated with OSS data in the CSV format**

  Syntax:

  ```
  create external table [if not exists] <mc_oss_extable_name>
  (
  <col_name> <data_type>,
  ...
  )
  [partitioned by (<col_name> <data_type>, ...)]
  row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    with serdeproperties
      ('separatorChar'=',', 'quoteChar'='"', 'escapeChar'='\\')
  stored as textfile
  location '<oss_location>';
  ```

  Note OpenCSVSerde supports only data of the STRING type. When you execute a data manipulation language (DML) statement, you must add the `set odps.sql.hive.compatible=true;` command before the DML statement and commit this command with the statement.

- If you use a CREATE EXTERNAL TABLE statement to create an OSS external table that is associated with OSS data in the CSV format, you can specify the following properties in `WITH SERDEPROPERTIES` of the statement:

  ```
  'separatorChar'=','
  'quoteChar'='"'
  'escapeChar'='\\'
  ```

- MaxCompute allows you to use the `skip.header.line.count` and `skip.footer.line.count` properties to skip the first and last rows of the CSV file during data processing. The properties can also be used if your CSV file is compressed in the .gz, .bz2, or .lzo format. Example:

  ```
  -- Enable the Hive-compatible data type edition. MaxCompute supports Hive syntax only a
  fter you enable the Hive-compatible data type edition.
  set odps.sql.hive.compatible=true;
  -- Enable the query of table data that is compatible with the data formats of other dat
  a platforms.
  set odps.sql.split.hive.bridge=true;
  -- Disable the native text reader.
  set odps.ext.hive.lazy.simple.serde.native=false;
  create external table oss_csv_compressed
  (
    col0 string,
    col1 string,
    col2 string
  )
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"= '"',
  "escapeChar"= "\\"
  )
  STORED AS TEXTFILE
  LOCATION 'oss://xxx:xxx=@oss-test.aliyun-inc.com/odps-oss-test/csv/compressed/'
  tblproperties (
  "skip.header.line.count"="1",
  "skip.footer.line.count"="1"
  );
  ```

- **Example of creating an OSS external table that is associated with OSS data in the JSON format**

  Syntax:

  ```
  create external table [if not exists] <mc_oss_extable_name>
  (
  <col_name> <data_type>,
  ...
  )
  [partitioned by (<col_name> <data_type>, ...)]
  row format serde 'org.apache.hive.hcatalog.data.JsonSerDe'
  stored as textfile
  location '<oss_location>';
  ```

Example of creating an OSS external table that is associated with OSS data in the ORC format

Syntax:

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as orc
location '<oss_location>';
```

Example of creating an OSS external table that is associated with OSS data in the AVRO format

Syntax:

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as avro
location '<oss_location>';
```

Example of creating an OSS external table that is associated with OSS data in the SEQUENCEFILE format

Syntax:

```
create external table [if not exists] <mc_oss_extable_name>
(
<col_name> <data_type>,
...
)
[partitioned by (<col_name> <data_type>, ...)]
stored as sequencefile
location '<oss_location>';
```

Read and process OSS data in an open source format

The preceding examples show that you need only to change file_format in the STORED AS clause when you create OSS external tables that are associated with OSS data in different open source formats.

After you create an OSS external table, use one of the following methods to read and process OSS data in an open source format based on your business requirements:

- Method 1 (recommended): Import OSS data in an open source format from the OSS external table into a MaxCompute internal table, and then read and process the data.

If you directly read data from the OSS external table, each read operation involves OSS I/O operations and high-performance optimizations on MaxCompute storage cannot be implemented. As a result, the performance of MaxCompute deteriorates. If you require repeated data computing or high computing efficiency, we recommend that you perform the following operations in sequence to improve the computing performance of MaxCompute: 1. Create a MaxCompute internal table that has the same schema as the OSS external table. 2. Import OSS data in an open source format from the OSS external table into the MaxCompute internal table. 3. Execute complex query statements for the MaxCompute internal table.

> **Note** If complex data types are involved in the SQL statement that you want to execute, you must add the `set odps.sql.type.system.odps2=true;` command before the SQL statement and submit this command with the statement. For more information about data types, see Data type editions.

Sample statements:

```
create table table_internal like table_extable;
insert overwrite table table_internal select * from table_extable;
```

- Method 2: Directly read and process OSS data in an open source format from the OSS external table. The operations are the same as those for reading and processing data in a MaxCompute internal table.

  If you use this method, the OSS external table is used as a MaxCompute internal table. The difference is that MaxCompute directly reads data from the OSS external table and processes the data.

  > **Note** If an OSS external table is associated with OSS data in a TEXTFILE format, you must set `odps.sql.hive.compatible` to true for a session before you use MaxCompute to read data from the OSS external table. Otherwise, an error is returned.

3.15.4.5. Read OSS data stored in partitions

MaxCompute allows you to create Object Storage Service (OSS) external tables as partitioned tables to access OSS data that is stored in partitions. This method reduces the volume of data that must be read and improves the data processing efficiency. This topic describes the standard and custom formats of OSS partition directories that are supported by MaxCompute.

Background information

After you create an OSS external table, MaxCompute performs a full table scan on all data in the OSS directory, including data files in subdirectories. If the volume of data in the OSS directory is large, a full table scan causes extra I/O operations and prolongs data processing. You can address this issue by using one of the following methods:

- Method 1 (recommended): Store data in a standard or custom partition directory in OSS. When you create an OSS external table by using MaxCompute, you must specify the partitions and oss_location information in the statement that is used to create the table. We recommend that you use standard partition directories. For more information, see Standard partition directories and Custom partition directories.
- Method 2: Plan multiple directories to store data. You can create multiple OSS external tables to read the data that is stored in each OSS directory. This way, each OSS external table points to a subset of OSS data. This method is complex and has poor data management performance. We

recommend that you do not use this method.

Standard partition directories

The following example shows the format of a standard partition directory.

```
oss://<oss_endpoint>/<Bucket name>/<Directory name>/<partitionKey1=value1>/<partitionKey2=value2>/...
```

> **Note** Data is uploaded to OSS in the OSS console or by using other OSS tools. Therefore, the format of the directory where the data is saved varies based on the format of the data that is uploaded.

Parameter	Required	Description
oss_endpoint	Yes	The OSS endpoint. We recommend that you use an internal endpoint of OSS to prevent extra fees that are incurred by Internet traffic. For more information about the internal endpoints of OSS, see Regions and endpoints. **Note** We recommend that the OSS bucket for storing data is deployed in the same region as the MaxCompute project. MaxCompute can be deployed only in some regions. Therefore, cross-region data connectivity issues may occur.
Bucket Name	Yes	The name of the OSS bucket. For more information about how to view the bucket name, see List buckets.
Directory Name	Yes	The name of the OSS directory. You do not need to include file names in directory names.
partitionKey	Yes	The name of the partition key column in the OSS external table.
value	Yes	The value of the partition key column in the OSS external table.

For example, a company stores log files in the CSV format in OSS and uses MaxCompute to process data in these log files on a daily basis. Specify the following standard partition directories for OSS data:

```
oss://oss-odps-test/log_data/year=2016/month=06/day=01/logfile
oss://oss-odps-test/log_data/year=2016/month=06/day=02/logfile
oss://oss-odps-test/log_data/year=2016/month=07/day=10/logfile
oss://oss-odps-test/log_data/year=2016/month=08/day=08/logfile
...
```

The following statements show how to create an OSS external table and import and analyze partition data based on the preceding partition directories:

```
-- Create an OSS external table.
create external table log_table_external (
    click STRING,
    ip STRING,
    url STRING
  )
partitioned by (
    year STRING,
    month STRING,
    day STRING
  )
stored by 'com.aliyun.odps.CsvStorageHandler'
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/log_data/';
-- Import OSS partition data.
alter table log_table_external add partition (year = '2016', month = '06', day = '01');
alter table log_table_external add partition (year = '2016', month = '06', day = '02');
alter table log_table_external add partition (year = '2016', month = '07', day = '10');
alter table log_table_external add partition (year = '2016', month = '08', day = '08');
...
-- Analyze the data. MaxCompute accesses only the file (logfile) in the log_data/year=2016/
month=06/day=01 subdirectory. MaxCompute does not scan all data in the log_data directory.
select count(distinct(ip)) from log_table_external where year = '2016' and month = '06' and
day = '01';
```

For more information about how to read data in a standard partition directory, see Example: Create a partitioned table as an OSS external table by using a built-in text extractor.

Custom partition directories

If data in OSS is stored in partitions in a non-standard partition directory, MaxCompute allows you to associate different subdirectories with different partitions.

The following example shows the format of custom partition directories. A custom partition directory contains only values of partition key columns and does not contain names of partition key columns.

```
oss://oss-odps-test/log_data_customized/2016/06/01/logfile
oss://oss-odps-test/log_data_customized/2016/06/02/logfile
oss://oss-odps-test/log_data_customized/2016/07/10/logfile
oss://oss-odps-test/log_data_customized/2016/08/08/logfile
...
```

After you create an OSS external table, execute the `alter table ... add partition ... location ...` statement to specify subdirectories and associate these subdirectories with different partitions. Sample statements:

```
alter table log_table_external add partition (year = '2016', month = '06', day = '01')
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-odps-test/log_data_customized/201
6/06/01/';
alter table log_table_external add partition (year = '2016', month = '06', day = '02')
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-odps-test/log_data_customized/201
6/06/02/';
alter table log_table_external add partition (year = '2016', month = '07', day = '10')
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-odps-test/log_data_customized/201
6/07/10/';
alter table log_table_external add partition (year = '2016', month = '08', day = '08')
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-odps-test/log_data_customized/201
6/08/08/';
```

After you specify custom partition directories, you can access data in the subdirectories regardless of whether the data is saved in standard partition directories.

3.15.4.6. Write data to OSS

MaxCompute allows you to execute INSERT statements to write data from a MaxCompute project to an Object Storage Service (OSS) directory by using OSS external tables. This topic describes how to write data to OSS and provides examples of data write operations.

Background information

You can write data in an internal table of MaxCompute to OSS. After you read data from an OSS external table and process the data, you can write the processed data back to OSS.

- Write data to OSS by using a built-in text extractor or a built-in open source data extractor.

  If you need to write data in the CSV or TSV format, an open source format, or a compression format that is supported by MaxCompute to OSS, you can create an OSS external table by using a built-in text extractor or a built-in open source data extractor of MaxCompute and execute an INSERT statement to write data to OSS by using the OSS external table.

  Examples:

  - Example: Write data to a non-partitioned OSS directory by using a built-in text extractor
  - Example: Write data to a partitioned OSS directory by using a built-in text extractor
  - Example: Write data in a compression format to OSS by using a built-in text extractor
  - Example: Write data to OSS by using a built-in open source data extractor

- Write data to OSS by using a custom extractor.

  If you need to write data in a custom format to OSS, you can create an OSS external table by using a custom extractor and execute an INSERT statement to write data to OSS by using the OSS external table.

  For more information, see Example: Write data to OSS by using a custom extractor.

- Write data to OSS by using the multipart upload feature.

If you need to write data in an open source format to OSS, you can create an OSS external table by using a built-in open source data extractor, enable or disable the multipart upload feature, and execute an INSERT statement to write data to OSS by using the OSS external table. To enable or disable the multipart upload feature, you must specify the `odps.sql.unstructured.oss.commit.mode` parameter at the session or project level. By default, this feature is disabled. For more information about the multipart upload feature, see Multipart upload.

Before you use this feature, make sure that jobConf2 is enabled for your project to generate a job execution plan. To enable jobConf2, set the `odps.sql.jobconf.odps2` parameter to True.

> **Note** The default value of `odps.sql.jobconf.odps2` is True. If it is not set to True, run the `set odps.sql.jobconf.odps2=true;` command to enable jobConf2 at the session level.

Valid values of the `odps.sql.unstructured.oss.commit.mode` parameter:

- False: Data that is written from MaxCompute to an OSS external table is stored in the .odps folder in the directory specified by LOCATION. A .meta file is included in the .odps folder to ensure that the data written to OSS is consistent with the data in MaxCompute. Only MaxCompute can correctly process the data in the .odps folder. If another data processing engine parses the data in this folder, an error is returned.
- True: MaxCompute uses the multipart upload feature to ensure that the data written to OSS is consistent with the data in MaxCompute in `two-phase commit` mode. In this case, the .odps folder and the .meta file are not generated. Other data processing engines can normally parse the data that is written from MaxCompute to OSS.

> **Notice** In extreme cases, if a job that executes the `INSERT OVERWRITE` statement fails, historical data is deleted but new data is not written to the destination table.
>
> Cause: A hardware failure occurs or the metadata fails to be updated. This issue rarely occurs. If this issue occurs, new data cannot be written to the destination table, and the deleted historical data cannot be restored because the delete operation in OSS cannot be rolled back.
>
> Solution:
>
> - If you want to use historical data to overwrite the data of an OSS external table, you must back up the OSS data. This way, you can use the backup data to overwrite the OSS external table when the job fails. For example, you can execute the `insert overwrite table T select * from table T;` statement to overwrite table T by using the historical data of table T.
> - If a job that executes the `INSERT OVERWRITE` statement can be repeatedly submitted, you can submit the job again when the job fails.

Prerequisites

Before you write data to OSS, make sure that the following requirements are met:

-
- (Optional) An OSS bucket, OSS directories, and OSS data files are prepared.

> **Note** MaxCompute can automatically create an OSS directory. You can also manually create an OSS directory.

For more information about how to create an OSS bucket, how to create an OSS directory, and how to upload data files, see Create buckets, Create directories, and Upload files.

Example: Write data to a non-partitioned OSS directory by using a built-in text extractor

Read data from the OSS external table mc_oss_csv_external1 that is created in Example: Create a non-partitioned table as an OSS external table by using a built-in text extractor and write the data in the CSV format to the OSS directory `oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo1/output/` .

Perform the following steps:

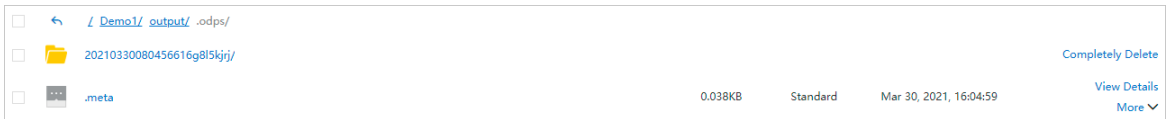
1. Log on to the MaxCompute client and execute the following statement to create an OSS external table. The OSS external table maps to the OSS directory to which you want to write data.

```
create external table if not exists mc_oss_csv_external4
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
 'odps.properties.rolearn'='acs:ram::xxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo1/output/';
```

2. Execute the `INSERT OVERWRITE` or `INSERT INTO` statement on the MaxCompute client to write data to the created OSS external table.

```
insert into table mc_oss_csv_external4 select * from mc_oss_csv_external1;
```

If the statement is successfully executed, you can view the exported files in the OSS directory. A folder named .odps is generated in the output folder. The .odps folder contains a .meta file and a folder where .csv files are saved. Example of a file in the OSS directory: `output/.odps/20210330******/R2_1_0_0-0_TableSink1-0-.csv` .



> **Note**
>
> - The .meta file in the .odps folder is an extra macro data file that is generated by MaxCompute. This file records valid data in the .odps folder. If the INSERT statement is successfully executed, all data in the .odps folder is valid. MaxCompute parses the .meta file only when the job that executes the INSERT statement fails. If the INSERT OVERWRITE statement is used and the job that executes the statement fails or is terminated, execute the INSERT OVERWRITE statement again.
> - If you use a built-in extractor of MaxCompute to process TSV or CSV files, the number of files generated in the OSS directory is the same as the number of concurrent SQL jobs.
> - If you execute the `insert overwrite ... select ... from ...;` statement and 1,000 mappers are allocated on the source table specified by from_tablename, 1,000 TSV or CSV files will be generated.
> - You can use flexible semantics and configurations of MaxCompute to limit the number of files that can be generated. If an outputer runs in a mapper, you can adjust the number of generated files by changing the value of `odps.stage.mapper.split.size` to adjust the number of concurrent mappers. If an outputer runs in a reducer, you can adjust the number of generated files by changing the value of `odps.stage.reducer.num`. If an outputer runs in a joiner, you can adjust the number of generated files by changing the value of `odps.stage.joiner.num`.

Example: Write data to a partitioned OSS directory by using a built-in text extractor

Read data from the OSS external table mc_oss_csv_external2 that is created in Example: Create a partitioned table as an OSS external table by using a built-in text extractor and write the data in the CSV format to the OSS directory `oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo2/output/`.

Perform the following steps:

1. Log on to the MaxCompute client and execute the following statement to create an OSS external table. The OSS external table maps to the OSS directory to which you want to write data.

```
create external table if not exists mc_oss_csv_external5
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string
)
partitioned by (
direction string
)
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
 'odps.properties.rolearn'='acs:ram::xxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo2/output/';
```

2. Execute the `INSERT OVERWRITE` or `INSERT INTO` statement on the MaxCompute client to write data to the created OSS external table.

```
insert into table mc_oss_csv_external5 partition (direction) select * from mc_oss_csv_e
xternal2;
```

If the statement is successfully executed, you can view the exported files in the OSS directory. Subdirectories that are mapped to the partitions specified in the INSERT statement are generated in the output folder. A folder named .odps is generated in each subdirectory. The .odps folder contains the .meta file and a folder where .csv files are saved. Example of a file in the OSS directory: `output/direction=N/.odps/20210330*********/R2_1_0_0-0_TableSink1-0-.csv`.

> **Note**
>
> - The .meta file in the .odps folder is an extra macro data file that is generated by MaxCompute. This file records valid data in the .odps folder. If the INSERT statement is successfully executed, all data in the .odps folder is valid. MaxCompute parses the .meta file only when the job that executes the INSERT statement fails. If the INSERT OVERWRITE statement is used and the job that executes the statement fails or is terminated, execute the INSERT OVERWRITE statement again.
> - If you use a built-in extractor of MaxCompute to process TSV or CSV files, the number of files generated in the OSS directory is the same as the number of concurrent SQL jobs.
> - If you execute the `insert overwrite ... select ... from ...;` statement and 1,000 mappers are allocated on the source table specified by from_tablename, 1,000 TSV or CSV files will be generated.
> - You can use flexible semantics and configurations of MaxCompute to limit the number of files that can be generated. If an outputer runs in a mapper, you can adjust the number of generated files by changing the value of `odps.stage.mapper.split.size` to adjust the number of concurrent mappers. If an outputer runs in a reducer, you can adjust the number of generated files by changing the value of `odps.stage.reducer.num`. If an outputer runs in a joiner, you can adjust the number of generated files by changing the value of `odps.stage.joiner.num`.

Example: Write data in a compression format to OSS by using a built-in text extractor

Read data from the OSS external table mc_oss_csv_external1 that is created in Example: Create a non-partitioned table as an OSS external table by using a built-in text extractor and write the data in the GZIP format to the OSS directory `oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo3/output/`.

Perform the following steps:

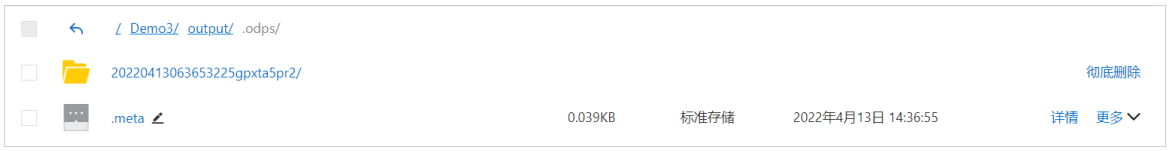
1. Log on to the MaxCompute client and execute the following statement to create an OSS external table. The OSS external table maps to the OSS directory to which you want to write data.

```
create external table if not exists mc_oss_csv_external6
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.CsvStorageHandler'
with serdeproperties (
 'odps.properties.rolearn'='acs:ram::xxxxxx:role/aliyunodpsdefaultrole',
 'odps.text.option.gzip.input.enabled'='true',
 'odps.text.option.gzip.output.enabled'='true'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo3/output/';
```

2. Execute the `INSERT OVERWRITE` or `INSERT INTO` statement on the MaxCompute client to write data to the created OSS external table.

```
insert into table mc_oss_csv_external6 select * from mc_oss_csv_external1;
```

If the statement is successfully executed, you can view the exported files in the OSS directory. A folder named .odps is generated in the output folder. The .odps folder contains the .meta file and a folder where .gz files are saved. Example of a file in the OSS directory: `output/.odps/20220413********/M1_0_0-0_TableSink1-0-.csv.gz`.



> **Note**
>
> - The .meta file in the .odps folder is an extra macro data file that is generated by MaxCompute. This file records valid data in the .odps folder. If the INSERT statement is successfully executed, all data in the .odps folder is valid. MaxCompute parses the .meta file only when the job that executes the INSERT statement fails. If the INSERT OVERWRITE statement is used and the job that executes the statement fails or is terminated, execute the INSERT OVERWRITE statement again.
> - If you use a built-in extractor of MaxCompute to process TSV or CSV files, the number of files generated in the OSS directory is the same as the number of concurrent SQL jobs.
> - If you execute the `insert overwrite ... select ... from ...;` statement and 1,000 mappers are allocated on the source table specified by from_tablename, 1,000 TSV or CSV files will be generated.
> - You can use flexible semantics and configurations of MaxCompute to limit the number of files that can be generated. If an outputer runs in a mapper, you can adjust the number of generated files by changing the value of `odps.stage.mapper.split.size` to adjust the number of concurrent mappers. If an outputer runs in a reducer, you can adjust the number of generated files by changing the value of `odps.stage.reducer.num`. If an outputer runs in a joiner, you can adjust the number of generated files by changing the value of `odps.stage.joiner.num`.

Example: Write data to OSS by using a built-in open source data extractor

Read data from the OSS external table mc_oss_csv_external1 that is created in Example: Create a non-partitioned table as an OSS external table by using a built-in text extractor and write the data in the ORC format to the OSS directory `oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo4/output/`.

Perform the following steps:

1. Log on to the MaxCompute client and execute the following statement to create an OSS external table. The OSS external table maps to the OSS directory to which you want to write data.

```
create external table if not exists mc_oss_orc_external
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored as orc
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/Demo4/output/';
```

2. Execute the `INSERT OVERWRITE` or `INSERT INTO` statement on the MaxCompute client to write data to the created OSS external table.

```
insert into table mc_oss_orc_external select * from mc_oss_csv_external1;
```

If the statement is successfully executed, you can view the exported files in the OSS directory. A folder named .odps is generated in the output folder. The .odps folder contains the .meta file and a folder where ORC files are saved. Example of a file in the OSS directory: `output/.odps/20220413********/M1_0_0-0_TableSink1`.

Note

- The .meta file in the .odps folder is an extra macro data file that is generated by MaxCompute. This file records valid data in the .odps folder. If the INSERT statement is successfully executed, all data in the .odps folder is valid. MaxCompute parses the .meta file only when the job that executes the INSERT statement fails. If the INSERT OVERWRITE statement is used and the job that executes the statement fails or is terminated, execute the INSERT OVERWRITE statement again.
- If you use a built-in extractor of MaxCompute to process TSV or CSV files, the number of files generated in the OSS directory is the same as the number of concurrent SQL jobs.
- If you execute the `insert overwrite ... select ... from ...;` statement and 1,000 mappers are allocated on the source table specified by from_tablename, 1,000 TSV or CSV files will be generated.
- You can use flexible semantics and configurations of MaxCompute to limit the number of files that can be generated. If an outputer runs in a mapper, you can adjust the number of generated files by changing the value of `odps.stage.mapper.split.size` to adjust the number of concurrent mappers. If an outputer runs in a reducer, you can adjust the number of generated files by changing the value of `odps.stage.reducer.num`. If an outputer runs in a joiner, you can adjust the number of generated files by changing the value of `odps.stage.joiner.num`.

Example: Write data to OSS by using a custom extractor

This example shows how to write data from MaxCompute to an OSS directory by using the custom extractor that is developed in Example: Create an OSS external table by using a custom extractor.

Perform the following steps:

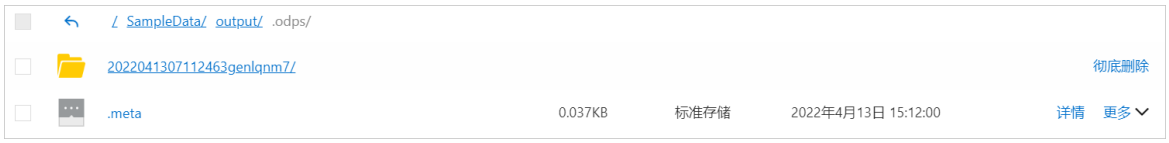
1. Log on to the MaxCompute client and execute the following statement to create an OSS external table. The OSS external table maps to the OSS directory to which you want to write data.

```
create external table if not exists mc_oss_external
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
stored by 'com.aliyun.odps.udf.example.text.TextStorageHandler'
  with serdeproperties (
'delimiter'='|',
'odps.properties.rolearn'='acs:ram::xxxxxxxxxxxxx:role/aliyunodpsdefaultrole'
)
location 'oss://oss-cn-hangzhou-internal.aliyuncs.com/oss-mc-test/SampleData/output'
using 'javatest-1.0-SNAPSHOT.jar';
```

2. Execute the `INSERT OVERWRITE` or `INSERT INTO` statement on the MaxCompute client to write data to the created OSS external table.

```
insert into table mc_oss_external select * from mc_oss_csv_external1;
```

If the statement is successfully executed, you can view the exported files in the OSS directory. A folder named .odps is generated in the output folder. The .odps folder contains the .meta file and a folder where TEXT files are saved. Example of a file in the OSS directory: `output/.odps/20220413*********/M1_0_0-0_TableSink1-0`.



Note

- The .meta file in the .odps folder is an extra macro data file that is generated by MaxCompute. This file records valid data in the .odps folder. If the INSERT statement is successfully executed, all data in the .odps folder is valid. MaxCompute parses the .meta file only when the job that executes the INSERT statement fails. If the INSERT OVERWRITE statement is used and the job that executes the statement fails or is terminated, execute the INSERT OVERWRITE statement again.
- If you use a built-in extractor of MaxCompute to process TSV or CSV files, the number of files generated in the OSS directory is the same as the number of concurrent SQL jobs.
- If you execute the `insert overwrite ... select ... from ...;` statement and 1,000 mappers are allocated on the source table specified by from_tablename, 1,000 TSV or CSV files will be generated.
- You can use flexible semantics and configurations of MaxCompute to limit the number of files that can be generated. If an outputer runs in a mapper, you can adjust the number of generated files by changing the value of `odps.stage.mapper.split.size` to adjust the number of concurrent mappers. If an outputer runs in a reducer, you can adjust the number of generated files by changing the value of `odps.stage.reducer.num`. If an outputer runs in a joiner, you can adjust the number of generated files by changing the value of `odps.stage.joiner.num`.

3.15.5. Access Tablestore data

This topic describes how to import data from Tablestore to MaxCompute. This allows you to create seamless connections between multiple data sources.

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. It allows you to store and access large amounts of structured data in real time. For more information, see Tablestore documentation.

You can create, search for, configure, and process external tables in the DataWorks console. You can also query and analyze data in external tables. For more information, see External table.

You must ensure the connectivity between MaxCompute and Tablestore. If you use Alibaba Cloud MaxCompute to access a Tablestore instance, we recommend that you use the internal endpoint of the Tablestore instance. The endpoint ends with *ots-internal.aliyuncs.com*. Example: *tablestore://odps-ots-dev.cn-shanghai.ots-internal.aliyuncs.com*.

Both Tablestore and MaxCompute have their own data type systems. The following table lists the mapping between data types supported by Tablestore and MaxCompute.

MaxCompute	Tablestore
STRING	STRING
BIGINT	INTEGER
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN

MaxCompute	Tablestore
BINARY	BINARY

STS authorization

A secure authorization channel is required for MaxCompute to access Tablestore data. MaxCompute uses Alibaba Cloud Resource Access Management (RAM) and Security Token Service (STS) to ensure the security of data access.

You can use one of the following methods to grant permissions:

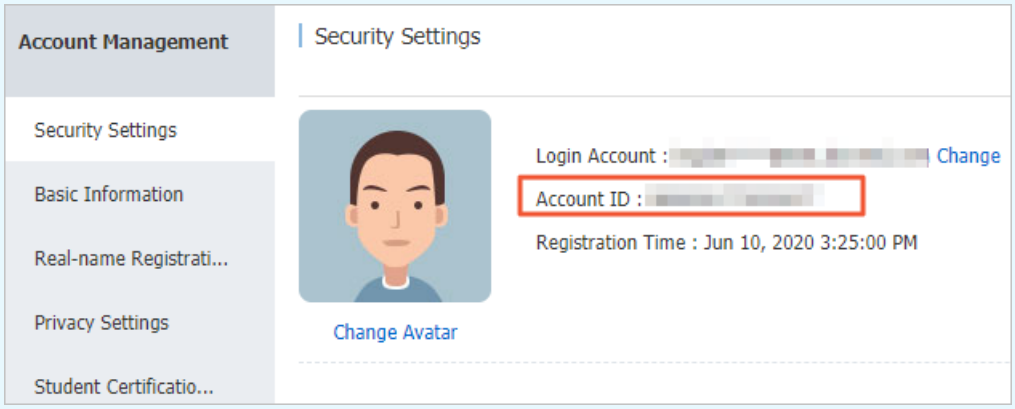
- If MaxCompute and Tablestore are under the same Alibaba Cloud account, log on to the Alibaba Cloud Management Console and click here to perform one-click authorization.
- Customize authorization.
  i. Authorize access from MaxCompute to Tablestore in the RAM console.

     Log on to the RAM console and create a role such as AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser. If MaxCompute and Tablestore are not under the same account, log on to the RAM console by using the Tablestore account.

  ii. Modify policy settings.

```
-- If MaxCompute and Tablestore are under the same account, use the following policy
configuration:
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "odps.aliyuncs.com"
   ]
 }
}
],
"Version": "1"
}
-- If MaxCompute and Tablestore are not under the same account, use the following pol
icy configuration:
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "ID of the Alibaba Cloud account of MaxCompute@odps.aliyuncs.com"
   ]
 }
}
],
"Version": "1"
}
```

Note You can click your profile picture in the upper-right corner of the page that appears to view the ID.



iii. Edit the AliyunODPSRolePolicy authorization policy for the role.

```
{
"Version": "1",
"Statement": [
{
 "Action": [
   "ots:ListTable",
   "ots:DescribeTable",
   "ots:GetRow",
   "ots:PutRow",
   "ots:UpdateRow",
   "ots:DeleteRow",
   "ots:GetRange",
   "ots:BatchGetRow",
   "ots:BatchWriteRow",
   "ots:ComputeSplitPointsBySize"
 ],
 "Resource": "*",
 "Effect": "Allow"
}
]
}
-- You can also customize other permissions.
```

iv. Grant the AliyunODPSRolePolicy permission to the role.

Create an external table

In MaxCompute, you can use external tables to import Tablestore data to the meta system of MaxCompute for processing. This section describes the related concepts and how to connect MaxCompute to Tablestore.

Execute the following statement to create an external table:

```
DROP TABLE IF EXISTS ots_table_external;
CREATE EXTERNAL TABLE IF NOT EXISTS ots_table_external
(
odps_orderkey bigint,
odps_orderdate string,
odps_custkey bigint,
odps_orderstatus string,
odps_totalprice double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler' -- (1)
WITH SERDEPROPERTIES ( -- (2)
'tablestore.columns.mapping'=':o_orderkey,:o_orderdate,o_custkey, o_orderstatus,o_totalpric
e', -- 1
'tablestore.table.name'='ots_tpch_orders', -- 2
'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole' --3
)
LOCATION 'tablestore://odps-ots-dev.cn-shanghai.ots-internal.aliyuncs.com'; -- (3)
```

Description:

- `com.aliyun.odps.TableStoreStorageHandler` is a built-in MaxCompute storage handler that is used

to process Tablestore data. It defines the interaction between MaxCompute and Tablestore. The related logic is implemented by MaxCompute.

- `SERDEPROPERITES` provides parameter options. If you use TableStoreStorageHandler, you must specify tablestore.columns.mapping, tablestore.table.name, and odps.properties.rolearn.
  i. tablestore.columns.mapping: the columns of the Tablestore table that MaxCompute needs to access. The columns include primary key columns and attribute columns.
     - The column whose name starts with a colon (:) is a primary key column, such as `:o_orderkey` and `:o_orderdate`. Other columns are attribute columns.
     - A Tablestore table supports a maximum of four primary key columns. The data types of these columns must be STRING, INTEGER, or BINARY. The first primary key column is the partition key.
     - If you specify column mappings, you must specify all primary key columns of the Tablestore table. You only need to specify the attribute columns that MaxCompute needs to access instead of specifying all attribute columns. The specified attribute columns must be in the Tablestore table. Otherwise, errors are returned when you query the new external table.
  ii. tablestore.table.name: the name of the Tablestore table that MaxCompute needs to access. If the table name does not exist in Tablestore, an error is returned. MaxCompute does not create a Tablestore table.
  iii. odps.properties.rolearn: the ARN of AliyunODPSDefaultRole in RAM. You can obtain the ID on the **RAM Roles** page of the RAM console.
- LOCATION specifies the name and endpoint of the Tablestore instance. You must complete RAM or STS authorization to ensure secure access to Tablestore data.

You can execute the following statement to view the structure of the new external table:

```
desc extended <table_name>;
```

In the execution result, Extended Info includes external table information such as the storage handler and location in addition to basic information.

Query an external table

After you create an external table, Tablestore data is imported to MaxCompute. Then, you can use MaxCompute SQL syntax to access Tablestore data. Example:

```
SELECT odps_orderkey, odps_orderdate, SUM(odps_totalprice) AS sum_total
FROM ots_table_external
WHERE odps_orderkey > 5000 AND odps_orderkey < 7000 AND odps_orderdate >= '1996-05-03' AND
odps_orderdate < '1997-05-01'
GROUP BY odps_orderkey, odps_orderdate
HAVING sum_total> 400000.0;
```

If you access Tablestore data by using MaxCompute SQL statements, all operations, such as the selection of column names, are completed in MaxCompute. In the preceding example, the column names are odps_orderkey and odps_totalprice rather than the name of the primary key column (o_orderkey) or attribute column (o_totalprice) in the original Tablestore table. Mapping is defined in the DDL statement that is used to create the external table. You can also retain the names of the primary key columns and attribute columns in the original Tablestore table as needed.

If you want to compute one piece of data multiple times, you can import the data from Tablestore to an internal table of MaxCompute. This way, you do not need to read the data from Tablestore every time.

```
CREATE TABLE internal_orders AS
SELECT odps_orderkey, odps_orderdate, odps_custkey, odps_totalprice
FROM ots_table_external
WHERE odps_orderkey > 5000 ;
```

internal_orders is a MaxCompute table that supports all the features of a MaxCompute internal table. The internal_orders table uses the efficiently compressed column store and includes complete internal macro data and statistical information. Tables are stored in MaxCompute, which delivers faster data access than Tablestore. This method is suitable for data that needs to be computed multiple times.

Export data from MaxCompute to Tablestore

Note MaxCompute does not create external tables in Tablestore. Before you export data to Tablestore table, make sure that the table exists. Otherwise, an error is reported.

An external table named ots_table_external is created to allow MaxCompute to access the ots_tpch_orders table in Tablestore. The data is also stored in the internal_orders table of MaxCompute. If you want to process data in the internal_orders table and export the processed data to Tablestore, execute the `INSERT OVERWRITE TABLE` statement. Example:

```
INSERT OVERWRITE TABLE ots_table_external
SELECT odps_orderkey, odps_orderdate, odps_custkey, CONCAT(odps_custkey, 'SHIPPED'), CEIL(o
dps_totalprice)
FROM internal_orders;
```

Note If the data in a MaxCompute table is sorted based on primary keys, data is written to a single partition of the Tablestore table. In this case, you cannot fully utilize distributed write operations. We recommend that you use `DISTRIBUTE BY rand()` to first scatter the data. Example:

```
INSERT OVERWRITE TABLE ots_table_external
SELECT odps_orderkey, odps_orderdate, odps_custkey, CONCAT(odps_custkey, 'SHIPPED'), CE
IL(odps_totalprice)
FROM (SELECT * FROM internal_orders DISTRIBUTE BY rand()) t;
```

Tablestore is a NoSQL data storage service that stores data in the key-value pair format. Data outputs from MaxCompute affect only the rows that include the primary keys of the Tablestore table. In this example, only the rows that include odps_orderkey and odps_orderdate are affected. Only the attribute columns that are specified when you create the ots_table_external table are updated. The columns that are not included in the external table are not modified.

Note

- If the size of data that you write from MaxCompute to Tablestore is greater than 4 MB at a time, you must remove the excess data and then write data to Tablestore again. In this case, an error may occur.

```
ODPS-0010000:System internal error - Output to TableStore failed with exception:
TableStore BatchWrite request id XXXXX failed with error code OTSParameterInvalid
and message:The total data size of BatchWriteRow request exceeds the limit
```

- It is considered a single operation to write multiple data entries at a time or by row. For more information, see BatchWriteRow. If you want to write large amounts of data at a time, you can write the data by row.
- If you write multiple data entries at a time, make sure that you do not write duplicate rows. If duplicate rows exist, the following error may occur:

```
ErrorCode: OTSParameterInvalid, ErrorMessage: The input parameter is invalid
```

For more information, see What do I do if OTSParameterInvalid is reported when I use BatchWriteRow to submit 100 data entries at a time?

3.15.6. Hologres external tables

You can use the external table feature of MaxCompute to access data of Hologres data sources by using the Java Database Connectivity (JDBC) driver. This topic describes how to execute the CREATE EXTERNAL TABLE statement to create a Hologres external table. In the statement, you must specify a Hologres data source, Security Token Service (STS) authentication information, a Hologres source table, and a JDBC driver.

Context

Hologres is a real-time interactive analytics data warehouse. It is compatible with PostgreSQL and seamlessly integrates with MaxCompute.

You can create a Hologres external table in MaxCompute to query the data of a Hologres data source based on STS authentication information and a PostgreSQL JDBC driver. This method prevents redundant data storage and allows you to obtain query results at a fast speed without the need to import or export data.

Prerequisites

Before you create a Hologres external table, make sure that the following conditions are met:

- A Hologres database and a Hologres source table are created.

  For more information about how to create a Hologres database, see Create a database.

  For more information about how to create a Hologres table, see CREATE TABLE.

  Information of the sample Hologres instance in this topic:

  - Name of the Hologres database: `mc_test`.
  - Schema of the Hologres database: `public`.

- Endpoint of the Hologres instance in the classic network: `hgprecn-cn-oew210ut****-cn-hangzhou-internal.hologres.aliyuncs.com:80`.
- Name of the Hologres table: `holo`. The following figure shows the data of the table.

A	B
id	name
1	kate
2	mary
3	bob
4	tom
5	lulu
6	mark
7	haward
8	lilei
9	hanmeimei
10	lily
11	lucy

- The MaxCompute project for which you want to create a Hologres external table is created.

  For more information about how to create a MaxCompute project, see Create a MaxCompute project.
- The MaxCompute client is installed.

  For more information about how to install the MaxCompute client, see Install and configure the MaxCompute client.

Limits

When you use Hologres external tables, take note of the following limits:

- MaxCompute does not support the update or delete operation on Hologres external tables.
- Partitioned tables of Hologres are not mapped to partitioned tables of MaxCompute. Hologres external tables cannot be partitioned.
- If you use multiple processes to concurrently write a large amount of data to a Hologres external table, a process rewrites data to the Hologres external table in rare cases. As a result, duplicate data exists.
- In a Hologres external table that you create in MaxCompute, data of the DECIMAL type must have a precision of 38 and a scale of 18. If the number of decimal places of data in a column is less than 18, you can set the data type of the column to string when you create a Hologres external table in MaxCompute. When you use the data in the external table, you can use the `CAST` function to forcefully convert the data of the STRING type to the DECIMAL type.
- Complex data types, such as ARRAY, MAP, and STRUCT, are not supported in the Hologres external table that you create in MaxCompute.
- The JSON, JSONB, and MONEY data types that are supported in Hologres do not match data types that are supported in MaxCompute. When you create a Hologres external table in MaxCompute, you cannot configure columns that match data of these types in the Hologres source table.

Usage notes

When you use Hologres external tables, take note of the following items:

- The names of the parent and child tables in a Hologres database are specified in Hologres external tables. SQL statements can be executed on the Hologres external tables. Parent and child tables can be mapped to Hologres external tables. However, parent tables can only be read.

- If the error `FAILED: Generating job conf failed, gen jobconf failed: External table location scheme "jdbc:postgresql" is not supported` is reported when you query data from a Hologres external table or insert data into a Hologres external table, the `jobconf2` feature that is used to optimize and update SQL execution plans is not enabled. By default, the jobconf2 feature is enabled for most MaxCompute projects. If the jobconf2 feature is disabled for your project, you can add the following properties at the session level to enable the feature:

```
set odps.sql.jobconf.odps2=true;
set odps.sql.jobconf.odps2.enforce=true;
set odps.sql.split.hive.bridge=true;
set odps.sql.hive.compatible=true;
```

- You cannot use the INSERT ON CONFLICT statement to write data to Hologres external tables. For more information about the INSERT ON CONFLICT statement, see INSERT ON CONFLICT. If the Hologres source table contains a primary key, you must ensure that the primary key of the data that you want to write to the Hologres external table is not the same as the primary key of the Hologres source table.
- When you create an external table, the table name and field names are not case-sensitive. When you query external tables or fields, the table names and field names are not case-sensitive, and forcible uppercase and lowercase conversions are not supported.

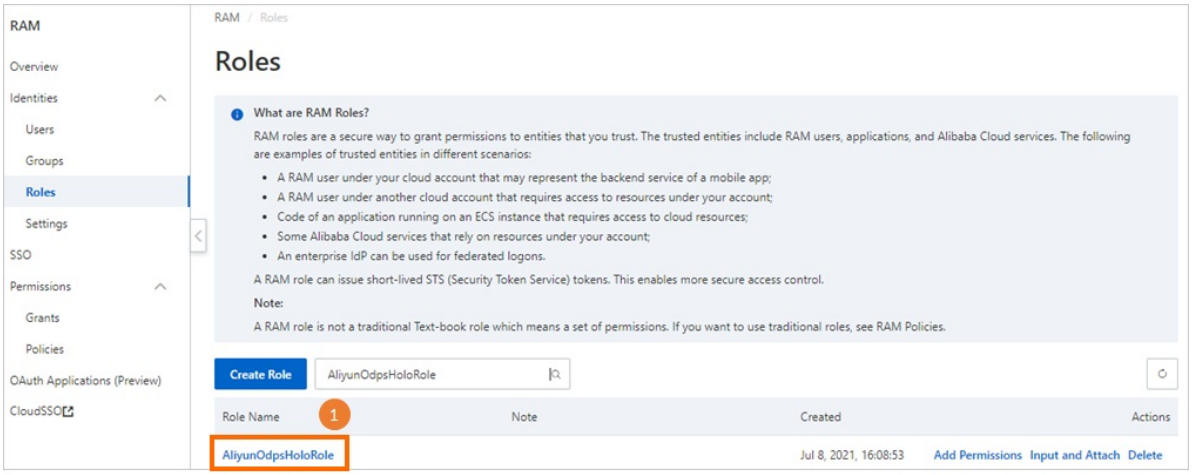
Syntax

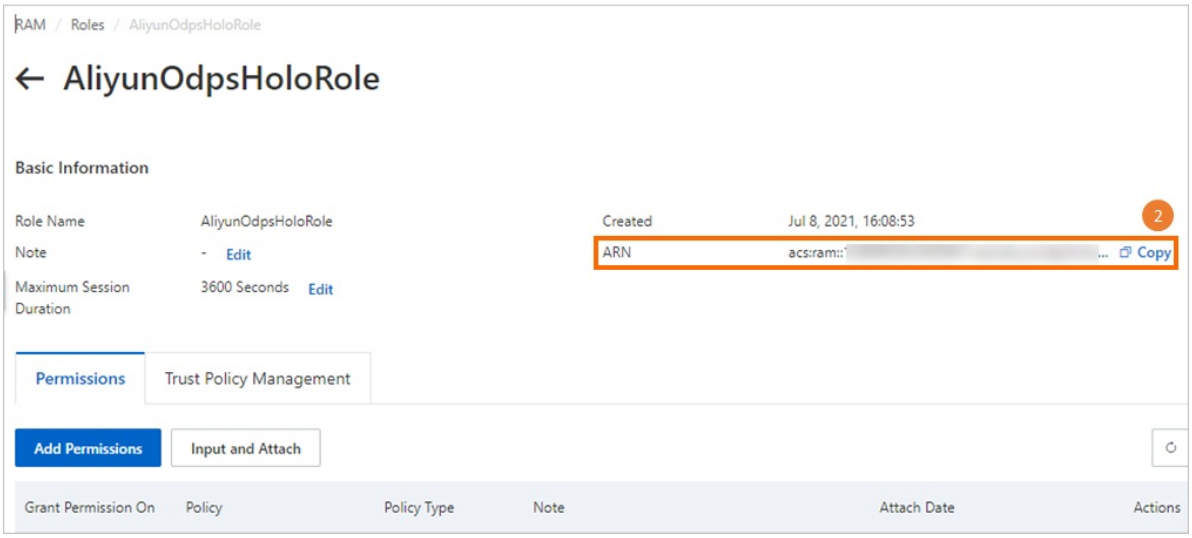
When you create a Hologres external table, you must specify a storage handler, STS authentication information, and the JDBC URL of a Hologres data source in the CREATE EXTERNAL TABLE statement. Syntax for creating a Hologres external table:

```
create external table [if not exists] <table_name>(
  <col1_name> <data_type>,
  <col2_name> <data_type>,
  ......
)
stored by '<com.aliyun.odps.jdbc.JdbcStorageHandler>'
with serdeproperties (
  'odps.properties.rolearn'='<ram_arn>')
location '<jdbc:postgresql://<endpoint>:<port>/<database>?ApplicationName=MaxCompute&[curre
ntSchema=<schema>&][useSSL={true|false}&]table=<holo_table_name>/>'
tblproperties (
  'mcfed.mapreduce.jdbc.driver.class'='org.postgresql.Driver',
  'odps.federation.jdbc.target.db.type'='holo',
  ['odps.federation.jdbc.colmapping'='<col1:column1,col2:column2,col3:column3,...>']
);
```

- if not exists: optional. If you create a table by using the name of an existing table but do not specify the if not exists parameter, an error is returned. If you specify the if not exists parameter, a success message is returned no matter whether a table with the same name already exists. The success message is returned even if the schema of the existing table is different from that of the table you want to create. If you create a table by using the name of an existing table, the table is not created and the metadata of the existing table is not changed.
- table_name: required. The name of the Hologres external table that you want to create.
- col_name: required. The name of a column in the Hologres external table.
- data_type: required. The data type of a column in the Hologres external table.

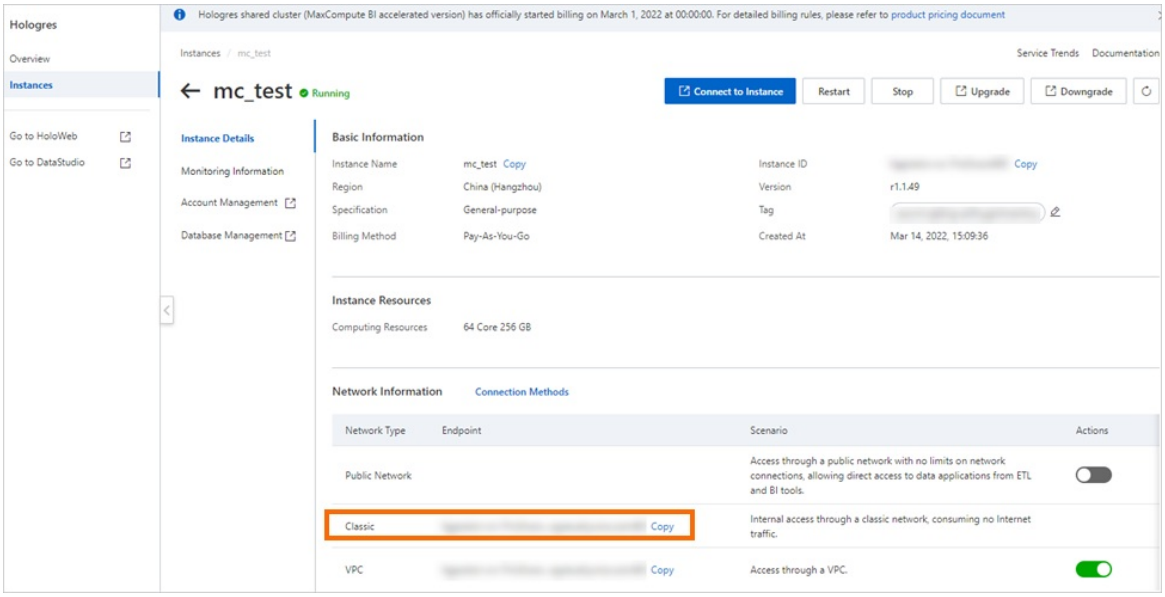
- stored by: required. A storage handler, which defines the method that you want to use to access the Hologres external table. Set the value to `com.aliyun.odps.jdbc.JdbcStorageHandler`. The value indicates that the Hologres external table is accessed by using JdbcStorageHandler.
- ram_arn: required. The Alibaba Cloud Resource Name (ARN) of the specified RAM role. The ARN is used as the STS authentication information. To obtain the ARN of the specified RAM role, you can perform the following steps: Log on to the Resource Access Management (RAM) console. On the Roles page, click the name of the RAM role whose ARN you want to query in the Role Name column. On the page that appears, view the ARN in the **Basic Information** section.



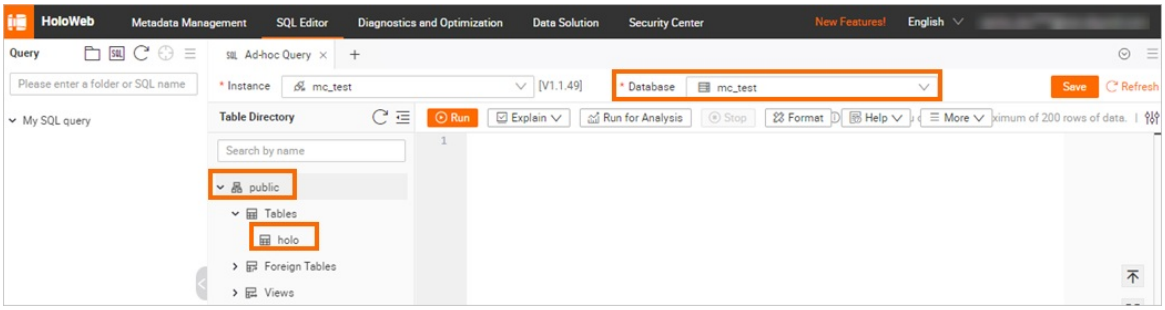


- location: required. The JDBC URL of the Hologres instance. Description of the fields in this parameter:
  - endpoint: required. The endpoint of the Hologres instance in the **classic network**. For more information about how to obtain endpoints, see Instance configurations.

- port: required. The port number of the Hologres instance. For more information about how to obtain port numbers, see Instance configurations.



- database: required. The name of the Hologres database that you want to access. For more information about Hologres databases, see CREATE DATABASE.
- ApplicationName: required. The default value is MaxCompute and no modification is required.
- schema: optional. If the name of the source table is unique in the Hologres database or the source table is a table in the default schema, you do not need to specify this parameter. For more information about schemas, see CREATE SCHEMA.
- holo_table_name: required. The name of the Hologres source table. For more information about Hologres source tables, see CREATE TABLE.



- tblproperties:
  - mcfed.mapreduce.jdbc.driver.class: required. The JDBC driver that is used to access the Hologres database. Set the value to `org.postgresql.Driver`.
  - odps.federation.jdbc.target.db.type: required. The type of the Hologres database that you want to access. Set the value to `holo`.

- odps.federation.jdbc.colmapping: optional. If you want to map some columns of the Hologres source table to the Hologres external table, you must configure this parameter. This parameter specifies the mappings between the fields of the Hologres source table and the fields of the Hologres external table. If you do not configure this parameter, the fields in the source table are mapped to the fields of the Hologres external table based on the sequence of the fields in the source table. The value of this parameter is in the `Column name 1 of the Hologres external table:Column name 1 of the Hologres source table,Column name 2 of the Hologres external table:Column name 2 of the Hologres source table,...` format.

Procedure

To create a Hologres external table, perform the following steps:

1. Step 1: Create a RAM role

   Create a RAM role and obtain the ARN of the RAM role. The ARN is used to specify the STS authentication information when you create an external table.

2. Step 2: Add the RAM role to a Hologres instance and grant permissions to the RAM role

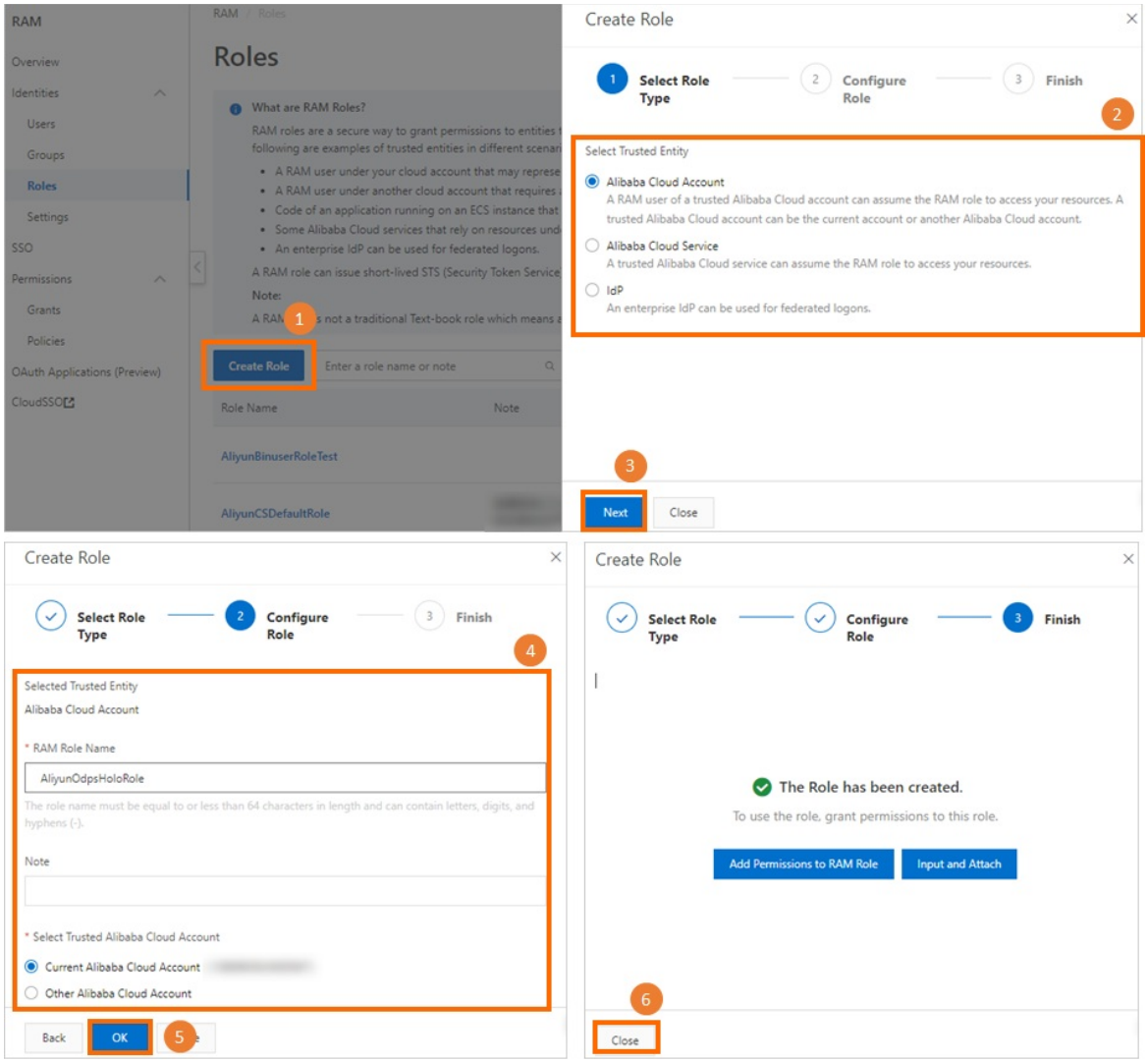
   Add the RAM role that you created to the Hologres instance and grant the RAM role the permissions to access the Hologres instance.

3. Step 3: Create a Hologres external table

   Create a Hologres external table that is mapped to the Hologres source table.

Step 1: Create a RAM role

1. Log on to the RAM console and create a RAM role.



In the Create Role panel, select Alibaba Cloud Account or IdP for Select Trusted Entity.

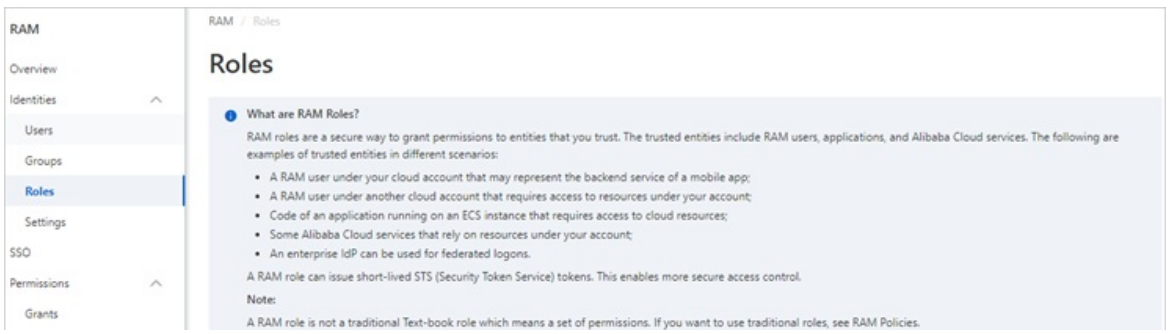
- Alibaba Cloud Account:

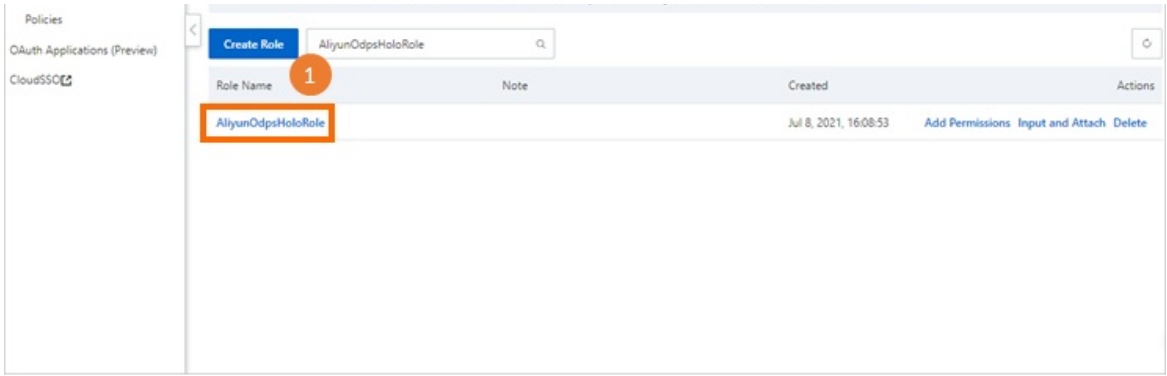
  A RAM user of your Alibaba Cloud account can access cloud resources by assuming a RAM role. For more information, see Create a RAM role for a trusted Alibaba Cloud account.

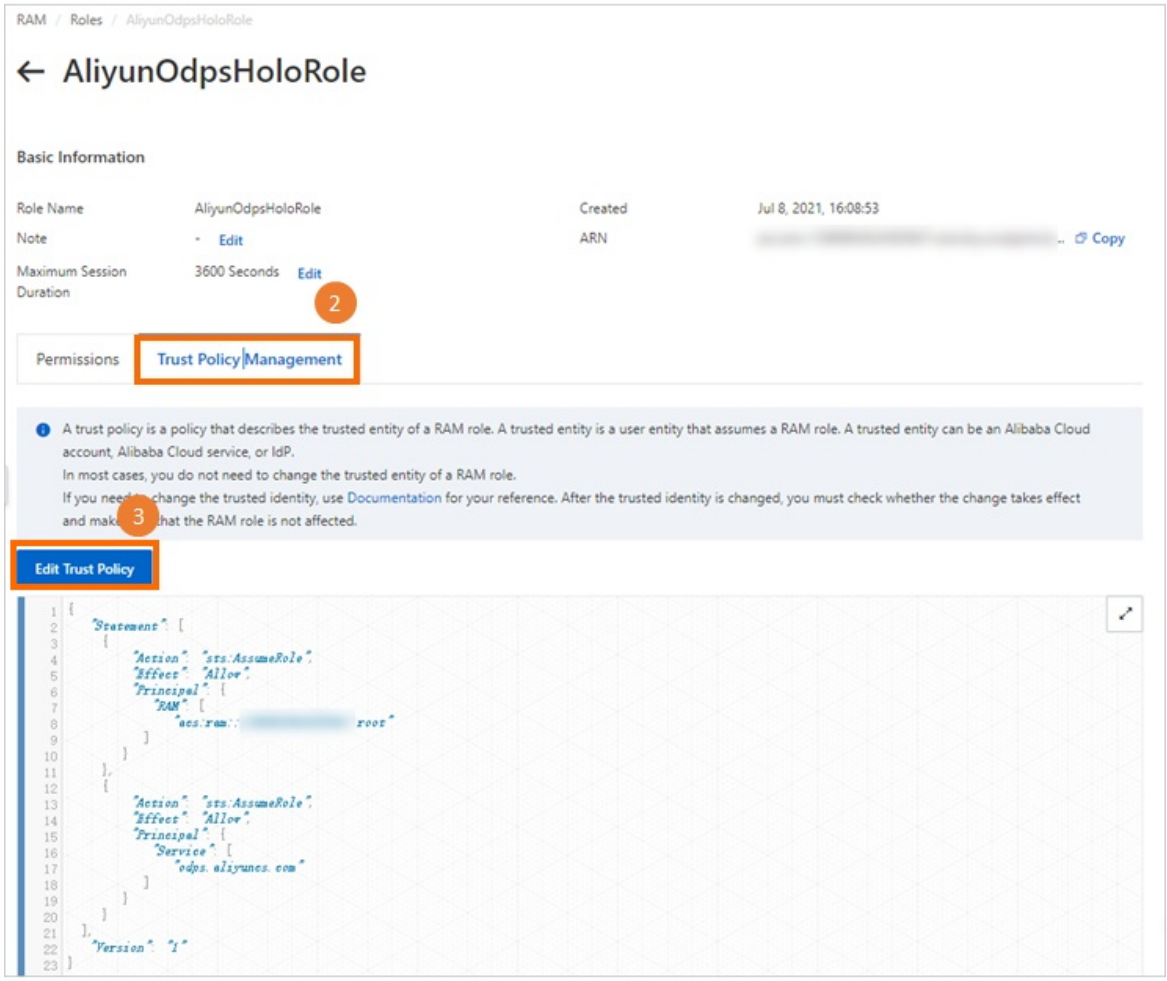
- IdP:

  You can log on to the Alibaba Cloud Management Console by using role-based single sign-on (SSO). You can log on without the need to provide a username and password. For more information about DataWorks, see Create a RAM role for a trusted IdP.

2. Edit the trust policy.




Policies
OAuth Applications (Preview)
CloudSSO
Create Role
AliyunOdpsHoloRole
1
Role Name
Note
Created
Actions
AliyunOdpsHoloRole
Jul 8, 2021, 16:08:53
Add Permissions Input and Attach Delete


RAM / Roles / AliyunOdpsHoloRole
← AliyunOdpsHoloRole
Basic Information
Role Name
AliyunOdpsHoloRole
Note
Edit
Maximum Session Duration
3600 Seconds
Edit
Created
Jul 8, 2021, 16:08:53
ARN
Copy
2
Permissions
Trust Policy Management
A trust policy is a policy that describes the trusted entity of a RAM role. A trusted entity is a user entity that assumes a RAM role. A trusted entity can be an Alibaba Cloud account, Alibaba Cloud service, or IdP.
In most cases, you do not need to change the trusted entity of a RAM role.
If you need to change the trusted identity, use Documentation for your reference. After the trusted identity is changed, you must check whether the change takes effect and make sure that the RAM role is not affected.
3
Edit Trust Policy
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"RAM": [
"acs:ram:[redacted]root"
]
}
},
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"odps.aliyuncs.com"
]
}
}
],
"Version": "1"
}


Edit Trust Policy
Role Name
AliyunOdpsHoloRole
4
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"RAM": [
"acs:ram:[redacted]root"
]
}
},
{
"Action": "sts:AssumeRole",
"Effect": "Allow",

i. On the **Roles** page, click the name of the RAM role that you create.

ii. Click the **Trust Policy Management** tab.

iii. On the **Trust Policy Management** tab, click **Edit Trust Policy**.

iv. In the **Edit Trust Policy** panel, modify the trust policy configuration based on the following content.

The configuration of the trust policy varies based on the type of the trusted entity that you selected.

- Alibaba Cloud Account is selected for Select Trusted Entity:

```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "RAM": [
          "acs:ram::<ID of your Alibaba Cloud account>:root"
        ]
      }
    },
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "odps.aliyuncs.com"
        ]
      }
    }
  ],
  "Version": "1"
}
```

- IdP is selected for Select Trusted Entity:

```
"Statement": [
        {
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "saml:recipient": "https://signin.aliyun.com/saml-role/sso"
                }
            },
            "Effect": "Allow",
            "Principal": {
                "Federated": [
                    "acs:ram::<ID of your Alibaba Cloud account>:saml-provider/ID
P"
                ]
            }
        },
        {
            "Action": "sts:AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "odps.aliyuncs.com"
                ]
            }
        }
    ],
    "Version": "1"
}
}
```

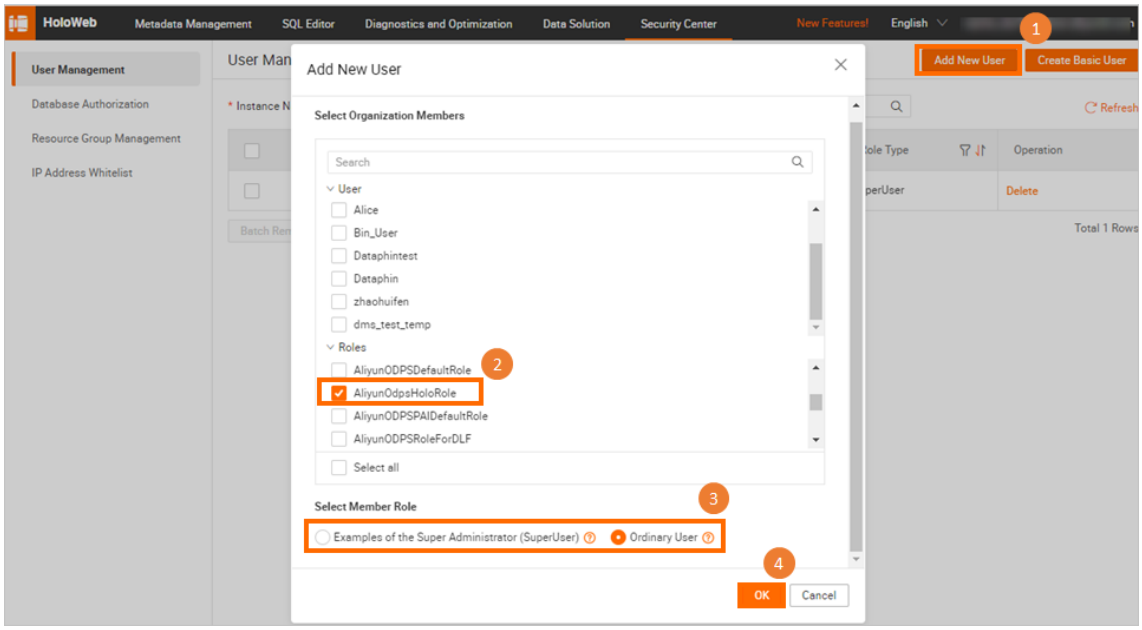
> **Note** You can obtain the ID of your Alibaba Cloud account on the Security Settings page of Account Center.

v. Click **OK**.

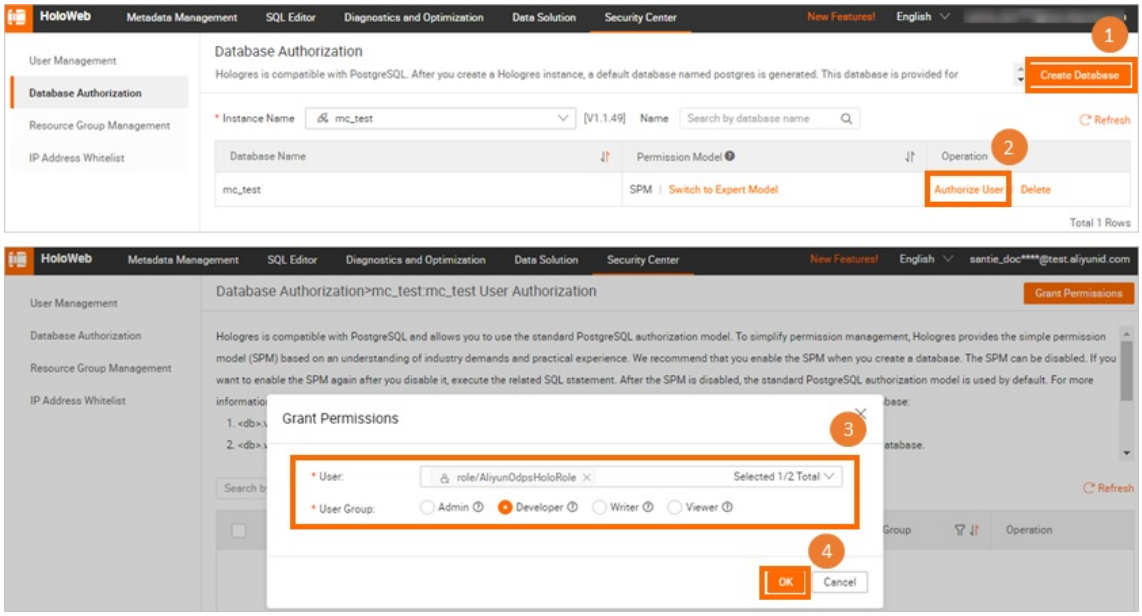
Step 2: Add the RAM role to a Hologres instance and grant permissions to the RAM role

Before the RAM role can use the Hologres instance, the role must obtain the required development permissions on the Hologres instance. By default, the RAM role is not granted the permissions to view or manage instances in the Hologres console. You must grant the permissions to the RAM role by using your Alibaba Cloud account. After you add the RAM role to a Hologres instance, you can use one of the following methods to grant the permissions to the RAM role:

- Use the Hologres console to grant the required permissions to the RAM role.
  i. Log on to the Hologres console.
  ii. In the left-side navigation pane, click **Instances**. On the Hologres Instances page, click the name of the Hologres instance to which you want to add the RAM role.
  iii. On the instance details page, click **Accounts**.
  iv. On the **User Management** page, click **Add New User** to add the RAM role to the Hologres instance.



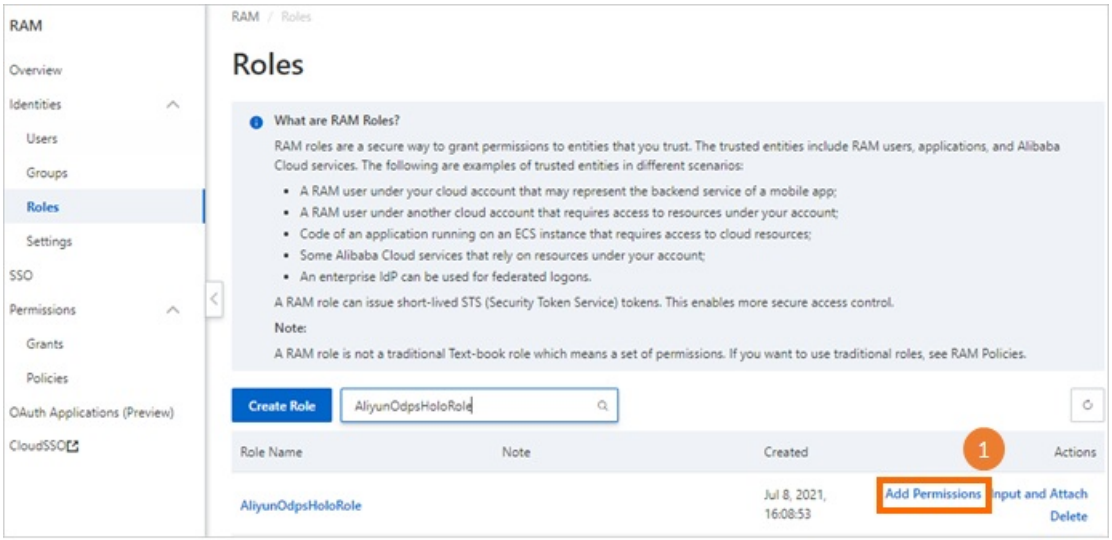
v. On the **Database Authorization** page, grant the development permissions on the instance to the RAM role.

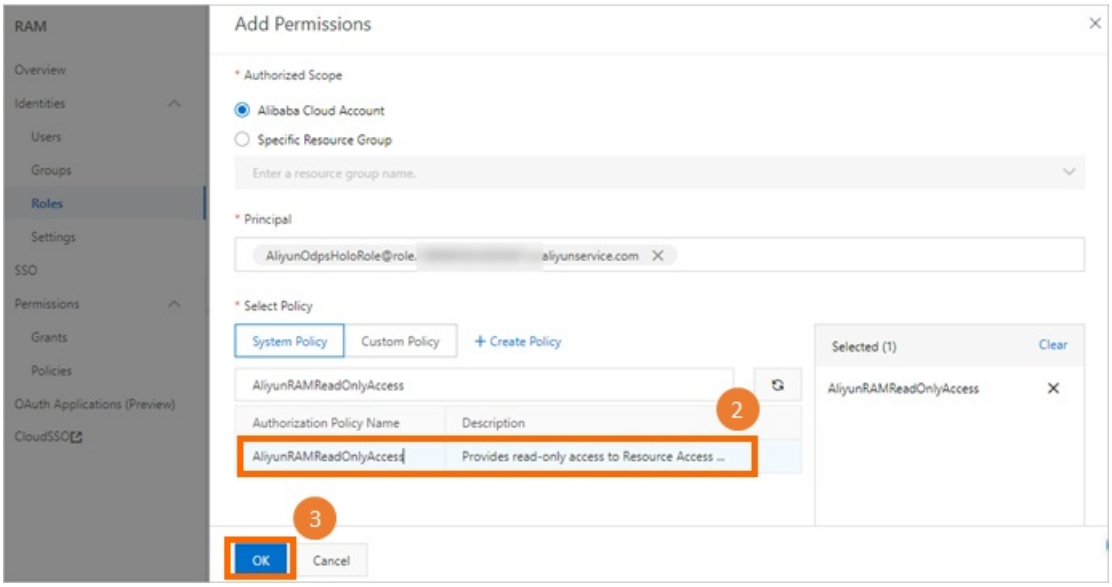


- Use an SQL statement to grant the required permissions to the RAM role.

  For more information about how to use an SQL statement to grant the required permissions to a RAM role, see Overview.

- By default, a RAM user is not granted the permissions to perform operations in the Hologres console. If you want a RAM user to assume the RAM role, you must attach the **AliyunRAMReadOnlyAccess** policy to the RAM user by using your Alibaba Cloud account. Otherwise, the RAM user cannot perform operations in the Hologres console. For more information, see Grant permissions on Hologres to RAM users.





Step 3: Create a Hologres external table

Log on to the MaxCompute client and create a Hologres external table based on the prepared data. For more information about the syntax, see Syntax.

1. Start the MaxCompute client and go to the MaxCompute project for which you want to create a Hologres external table.

   For more information about the commands that are used to go to a project, see Project operations.

2. Execute the following statement to create a Hologres external table:

```
create external table if not exists my_table_holo_jdbc
(
 id bigint,
 name string
)
stored by 'com.aliyun.odps.jdbc.JdbcStorageHandler'
with serdeproperties (
  'odps.properties.rolearn'='acs:ram::139699392458****:role/aliyunodpsholorole')
location 'jdbc:postgresql://hgprecn-cn-oew210ut****-cn-hangzhou-internal.hologres.aliyu
ncs.com:80/mc_test?ApplicationName=MaxCompute&currentSchema=public&useSSL=false&table=h
olo/'
tblproperties (
  'mcfed.mapreduce.jdbc.driver.class'='org.postgresql.Driver',
  'odps.federation.jdbc.target.db.type'='holo',
  'odps.federation.jdbc.colmapping'='id:id,name:name'
);
```

3. Execute the following statements to query the data of the Hologres source table by using the created Hologres external table:

```
-- Add the following properties to access the Hologres external table.
set odps.sql.split.hive.bridge=true;
set odps.sql.hive.compatible=true;
-- Query the data of the Hologres external table.
select * from my_table_holo_jdbc limit 10;
```

The following results are returned:

```
+------------+------------+
| id         | name       |
+------------+------------+
| 1          | kate       |
| 2          | mary       |
| 3          | bob        |
| 4          | tom        |
| 5          | lulu       |
| 6          | mark       |
| 7          | haward     |
| 8          | lilei      |
| 9          | hanmeimei  |
| 10         | lily       |
+------------+------------+
```

4. (Optional)Exchange data between MaxCompute and Hologres based on the Hologres external table and perform joint data analysis.

   For example, you can write data that is processed by MaxCompute to a Hologres database by using the Hologres external table. This accelerates data analysis and implements online services. Sample statement:

```
-- Add the following properties to access the Hologres external table.
set odps.sql.split.hive.bridge=true;
set odps.sql.hive.compatible=true;
-- Insert data into the Hologres external table.
insert into my_table_holo_jdbc values (12,'alice');
-- Query the data of the Hologres external table.
select * from my_table_holo_jdbc;
```

The following result is returned:

```
+------------+------------+
| id         | name       |
+------------+------------+
| 12          | alice      |
| 1          | kate       |
| 2          | mary       |
| 3          | bob        |
| 4          | tom        |
| 5          | lulu       |
| 6          | mark       |
| 7          | haward     |
| 8          | lilei      |
| 9          | hanmeimei  |
| 10         | lily       |
| 11         | lucy       |
+------------+------------+
```

Dimension tables that are frequently updated are stored in Hologres databases. This meets real-time data update requirements. MaxCompute uses external tables to access the dimension tables in Hologres databases. Then, association analysis is performed on the data in the dimension tables and the fact tables in MaxCompute. Sample statements:

```
-- Add the following properties to access the Hologres external table.
set odps.sql.split.hive.bridge=true;
set odps.sql.hive.compatible=true;
-- Create a MaxCompute internal table.
create table holo_test as select * from my_table_holo_jdbc;
--- Perform association analysis on the data in the MaxCompute internal table and Holog
res external table.
select * from my_table_holo_jdbc t1 inner join holo_test t2 on t1.id=t2.id;
```

The following result is returned:

```
+------------+------------+------------+------------+
| id         | name       | id2        | name2      |
+------------+------------+------------+------------+
| 1          | kate       | 1          | kate       |
| 2          | mary       | 2          | mary       |
| 3          | bob        | 3          | bob        |
| 4          | tom        | 4          | tom        |
| 5          | lulu       | 5          | lulu       |
| 6          | mark       | 6          | mark       |
| 7          | harward    | 7          | harward    |
| 8          | lilei      | 8          | lilei      |
| 9          | hanmeimei  | 9          | hanmeimei  |
| 10         | lily       | 10         | lily       |
| 11         | lucy       | 11         | lucy       |
| 12         | alice      | 12         | alice      |
+------------+------------+------------+------------+
```

3.16. MaxCompute Query Acceleration

3.16.1. Overview

This topic describes the system architecture, benefits, scenarios, and limits of the MaxCompute Query Acceleration (MCQA) feature.

Feature description

The MCQA feature of MaxCompute accelerates the execution of small- and medium-sized query jobs and reduces the execution time from minutes to seconds. This feature is compatible with other query features of MaxCompute.

MCQA allows you to perform ad hoc queries or business intelligence (BI) analysis by connecting mainstream BI tools or SQL clients to MaxCompute projects.

MCQA uses an independent resource pool that does not use quota groups. MCQA automatically identifies query jobs and shortens the job queue length, which improves user experience.

The billing method of MCQA query jobs is pay-as-you-go, which is the same as SQL query jobs. For more information, see Pay-as-you-go billing for MCQA jobs.

MaxCompute provides a free trial of MCQA so that you can run SQL jobs that use subscription resources. For more information, see Free trial of MCQA.

MaxCompute allows you to write the query results of MCQA jobs to a temporary cache. If you run the same query job later, MaxCompute preferentially returns the results in the cache to speed up the execution of the job. For more information about how to cache query results, see Enhancement: Query result caching mechanism of MCQA jobs.

> **Note** If an MCQA query job is rolled back to an SQL query job, you are charged based on the billing method of the SQL query job.

Benefits

MCQA has the following benefits:

- Low-latency resource scheduling

  Uses an efficient and low-latency resource scheduling policy and an independent resource pool.

- Automatic identification

  Automatically identifies the size of query jobs. MaxCompute can use MCQA to accelerate the queries or process multiple query jobs at the same time and then return the query results. This allows you to analyze query jobs of different sizes or complexities.

- Syntax compatibility

  Uses the same SQL syntax as MaxCompute.

- Selection of query methods

  Allows the MaxCompute client to use MCQA or the offline mode to run query jobs. You can also configure MaxCompute to forcibly use MCQA in latency-sensitive scenarios.

Scenarios

The following table describes the scenarios for which MCQA is suitable.

Scenario	Description	Applicable scope
Ad hoc query	You can use MCQA to optimize the query performance of small- and medium-sized datasets (less than 100 GB) and perform low-latency queries on MaxCompute tables. This helps develop and analyze data.	You can specify query criteria based on your requirements, obtain query results, and adjust the query logic. In this scenario, the query latency must be within dozens of seconds. Users are data developers or data analysts who have mastered SQL skills and prefer to use the client tools that they are familiar with to analyze queries.
BI analysis	If you use MaxCompute to build an enterprise-class data warehouse, MaxCompute performs extract, transform, load (ETL) operations to process data into business-oriented and consumable aggregate data. MCQA features low latency and supports elastic concurrency and data caching. You can use MCQA with the partitions and buckets in MaxCompute tables to run concurrent jobs, generate reports, analyze statistics, and analyze fixed reports at a low cost.	In most cases, the query object is aggregate data. This scenario is suitable for multidimensional queries, fixed queries, or high-frequency queries that contain small amounts of data. In this scenario, queries are latency-sensitive, and the results are returned in seconds. For example, the latency for most queries is less than 5 seconds. The time elapsed for each query varies based on the data size and query complexity.

Scenario	Description	Applicable scope
Detailed queries and analysis of large amounts of data	MCQA automatically identifies the size of query jobs. MCQA can respond to and process small-sized jobs in a timely manner, and can allocate the resources required for large-sized jobs. This helps analysts run query jobs of different sizes and complexities.	In this scenario, large amounts of historical data are queried. The size of the valid data that is queried is small, and the requirement for latency is low. Users are business analysts who need to gain business insights from data, explore potential business opportunities, and validate business assumptions.

Limits

MCQA supports only SELECT statements and does not support user-defined functions (UDFs). If you submit a statement or function that MCQA does not support from the MaxCompute client of V0.35.1 or later, the client automatically rolls back to the common offline mode to execute the statement or function. If you use the MaxCompute JDBC driver (V3.2.3 or later) or MaxCompute SDK (V0.35.1 or later), you can configure related parameters to allow the MaxCompute client to automatically roll back to the offline mode. Other tools do not support such a rollback.

The following table describes the limits of MCQA.

Item	Description
Feature	• The MCQA feature is available only in MaxCompute Standard Edition that uses the pay-as-you-go billing method. • The MCQA feature supports the subscription billing method. MaxCompute provides a free trial of the MCQA feature that uses subscription resources. For more information about the free trial, see Free trial of MCQA. • The MCQA feature is unavailable in MaxCompute Developer Edition. You must upgrade MaxCompute to Standard Edition.

Item	Description
Query	• If more than 1,000 instances are used by an MCQA query job, the system automatically rolls back this MCQA query job to an SQL query job. • If you submit an MCQA query job from the MaxCompute client, the default timeout period is 30 seconds. If you submit an MCQA query job from the ad hoc query module of DataWorks, the default timeout period is 20 seconds. If the MCQA query job times out, the system automatically rolls back the MCQA query job to an SQL query job. • MCQA can cache only the data that is stored in ALIORC tables into memory to accelerate queries. • You cannot use MCQA to query data from external tables.
Query concurrency	You can run a maximum of 120 parallel MCQA query jobs in each MaxCompute project.

3.16.2. Usage notes

MaxCompute Query Acceleration (MCQA) is a built-in feature of MaxCompute. MCQA uses the native MaxCompute SQL language and supports the built-in functions and permission systems of MaxCompute. This topic describes how to use the MCQA feature.

Context

You can use one of the following methods to enable the MCQA feature:

- Use the MaxCompute client. For more information, see Enable MCQA on the MaxCompute client.
- Use the Ad-Hoc Query or Manually Triggered Workflow feature of DataWorks. By default, MCQA is enabled for the Ad-Hoc Query or Manually Triggered Workflow feature. For more information, see Enable MCQA on the Ad-Hoc Query or Manually Triggered Workflow page.
- Use the MaxCompute Java Database Connectivity (JDBC) driver. For more information, see JDBC.
- Use a MaxCompute SDK. This method requires dependencies in a pom.xml file. For more information, see Enable MCQA by using MaxCompute SDK for Java.
- Use MaxCompute Studio. If you use MaxCompute Studio to enable the MCQA feature, you must configure the SQL execution mode. For more information, see Enable MCQA by using MaxCompute Studio
- Use PyODPS. If you use PyODPS to enable the MCQA feature, you must call the run_sql_interactive() method. For more information, see Enable MCQA by using PyODPS.
- Use SQLAlchemy based on PyODPS. For more information, see Enable MCQA by using SQLAlchemy of PyODPS or by using a third-party tool that supports SQLAlchemy interface.

Enable MCQA on the MaxCompute client

To enable the MCQA feature on the MaxCompute client, perform the following steps:

1. Download the latest version of the MaxCompute client (odpscmd).

> **Note** The client version must be V0.35.1 or later.

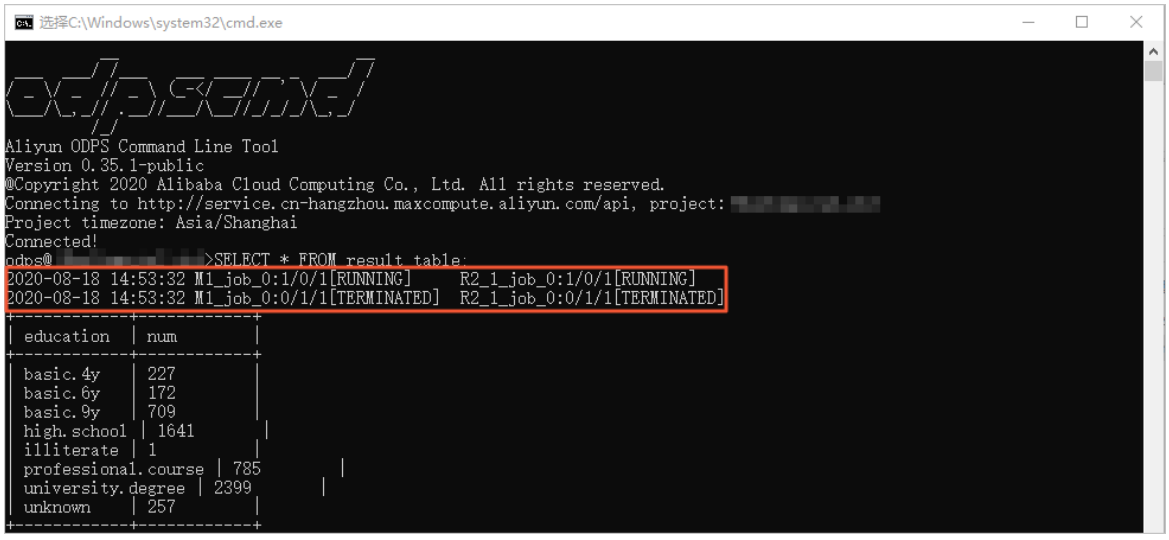
2. Install and configure the client. For more information, see Install and configure the MaxCompute client.
3. Modify the odps_config.ini configuration file in the conf folder, and add the following command at the end of the configuration file:

```
enable_interactive_mode=true
```

4. Run the MaxCompute client in the bin folder. Run ./bin/odpscmd in Linux, and run ./bin/odpscmd.bat in Windows. If the following information appears, the MaxCompute client is running properly.

5. Run a query job to verify that the MCQA feature is enabled.

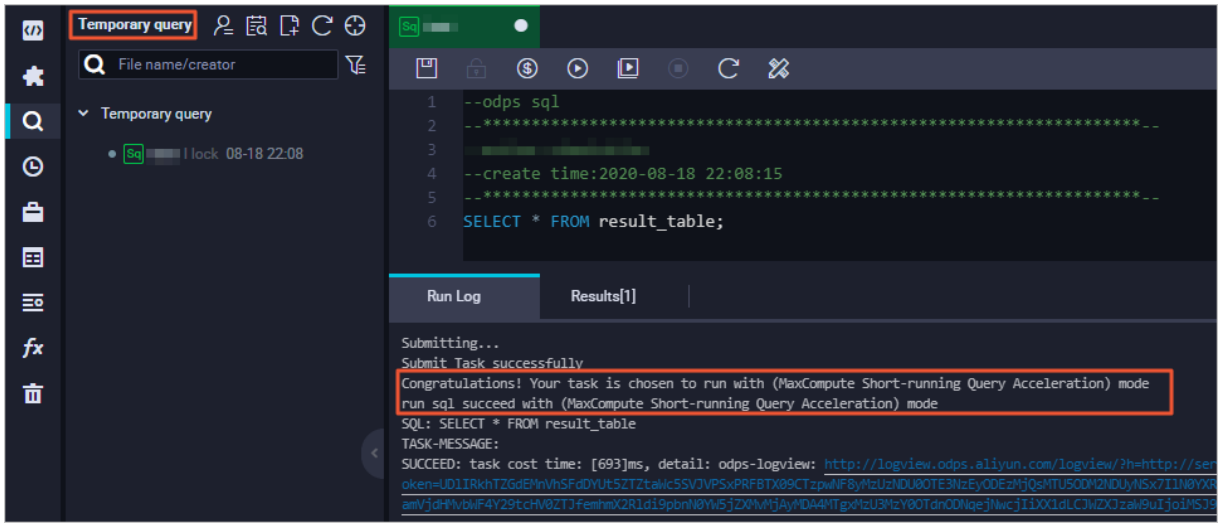
   If the returned results contain the following information after you run the query job, the MCQA feature is enabled.



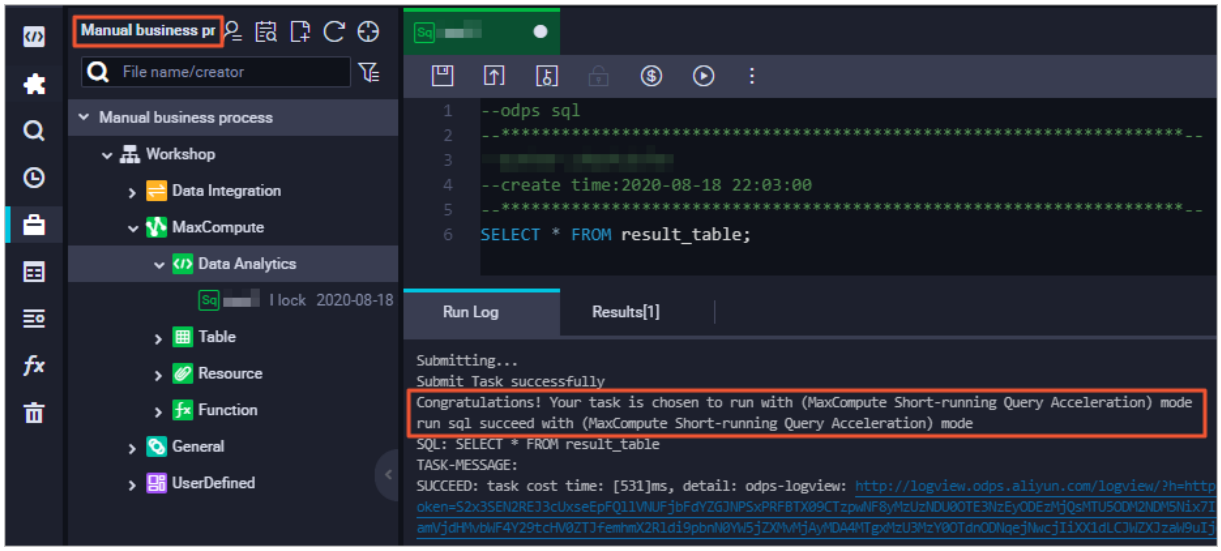
Enable MCQA on the Ad-Hoc Query or Manually Triggered Workflow page

By default, the MCQA feature is enabled on the **Ad-Hoc Query** or **Manually Triggered Workflow** page in the DataWorks console. Manual operations are not required. If you want to disable the MCQA feature, submit a ticket.

Run a query job on the **Ad-Hoc Query** page. If the returned results contain the following information, the MCQA feature is enabled. For more information about **Ad-Hoc Query**, see Create an ad hoc query.



Run a query job on the **Manually Triggered Workflow** page. If the returned results contain the following information, the MCQA feature is enabled. For more information about **Manually Triggered Workflow**, see Manage manually triggered workflows.



JDBC

You can use the MaxCompute JDBC driver to enable the MCQA feature in the following scenarios:

- Use the MaxCompute JDBC driver to connect to MaxCompute. For more information, see Usage notes of JDBC. To enable the MCQA feature, you must modify the relevant configurations. For more information, see Enable MCQA by using the MaxCompute JDBC driver.
- Use the MaxCompute JDBC driver to connect to Tableau. Then, you can use Tableau to analyze MaxCompute data in a visualized manner. For more information, see Configure MaxCompute JDBC on Tableau. To enable the MCQA feature, you must modify the relevant configurations. For more information, see Enable MCQA on Tableau Server by using the MaxCompute JDBC driver.
- Use the MaxCompute JDBC driver to connect to SQL Workbench/J. Then, you can use SQL Workbench/J to execute SQL statements on the MaxCompute data. For more information, see Configure MaxCompute JDBC on SQL Workbench/J. To enable the MCQA feature, you must modify the relevant

configurations. For more information, see Enable MCQA on SQL Workbench/J by using the MaxCompute JDBC driver.

Enable MCQA by using the MaxCompute JDBC driver

If you use the MaxCompute JDBC driver to connect to MaxCompute, perform the following steps to enable the MCQA feature. For more information, see Usage notes of JDBC.

1. Download the JDBC JAR file that supports the MCQA feature or download the source code that can be compiled.
2. Add the following dependency to the pom.xml file in the Maven repository:

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-jdbc</artifactId>
  <version>3.2.8</version>
  <classifier>jar-with-dependencies</classifier>
</dependency>
```

> **Note** The version of the MaxCompute JDBC driver must be V3.2.0 or later.

3. Create a Java program based on the source code and configure the required information. For more information, see ODPS JDBC.

   The source code contains the following information:

```
String accessId = "your_access_id";
String accessKey = "your_access_key";
String conn = "jdbc:odps:http://service.<regionid>.maxcompute.aliyun.com/api?project=<y
our_project_name>"&accessId&accessKey&charset=UTF-8&interactiveMode=true&autoSelectLimi
t=1000000000";
Statement stmt = conn.createStatement();
-- Replace your_access_id with the AccessKey ID of your Alibaba Cloud account and your_
access_key with the AccessKey secret of your Alibaba Cloud account. Replace your_projec
t_name with the name of the project for which the MCQA feature is enabled.
Connection conn = DriverManager.getConnection(conn, accessId, accessKey);
Statement stmt = conn.createStatement();
String tableName = "testOdpsDriverTable";
stmt.execute("drop table if exists " + tableName);
stmt.execute("create table " + tableName + " (key int, value string)");
```

   To enable the MCQA feature by using the MaxCompute JDBC driver, you need only to modify `String conn` or the code.

   - `String conn` : Add `interactiveMode=true` to enable the MCQA feature. If the number of data records exceeds 1,000,000, add `autoSelectLimit=1000000000"` .
   - Code: Add `interactiveMode=true` . If the number of data records exceeds 1,000,000, add `autoSelectLimit=1000000000"` .

4. (Optional)Configure the following parameters in String conn to optimize the processing logic.
   - `enableOdpsLogger` : is used to display logs. If you do not configure Simple Logging Facade for Java (SLF4J), we recommend that you set this parameter to True.
   - `fallbackForUnknownError` : The default value is False. If this parameter is set to True, the

system rolls back to the offline mode when an unknown error occurs.

- `fallbackForResourceNotEnough`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when resources are insufficient.
- `fallbackForUpgrading`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode during an upgrade.
- `fallbackForRunningTimeout`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when an operation times out.
- `fallbackForUnsupportedFeature`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when the MCQA feature is not supported.
- `alwaysFallback`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode in any of the preceding scenarios. This parameter is supported only for MaxCompute JDBC driver V3.2.3 and later.

Enable MCQA on Tableau Server by using the MaxCompute JDBC driver

Add `interactiveMode=true` to the URL of **Tableau Server**. We recommend that you also add `enableOdpsLogger=true` to display logs. For more information, see Configure MaxCompute JDBC on Tableau.

Sample URL:

```
http://service.cn-beijing.maxcompute.aliyun.com/api?project=****_beijing&interactiveMode=true&enableOdpsLogger=true&autoSelectLimit=1000000000"
```

To enable MCQA for some tables in a MaxCompute project, add the `table_list=table_name1,table_name2` property to the URL of Tableau **Server**. Then, use this property to specify the tables for which you want to enable MCQA. Separate table names with commas (,). If you specify an excessive number of tables, access to the URL of Tableau Server becomes time-consuming. We recommend that you specify only the required tables in the URL of Tableau Server. Sample URL:

```
http://service.cn-beijing.maxcompute.aliyun.com/api?project=****_beijing&interactiveMode=true&enableOdpsLogger=true&autoSelectLimit=1000000000"&table_list=orders,customers
```

If a table contains a large number of partitions, we recommend that you do not use data from all the partitions as the data source. You can filter the required partitions or run custom SQL queries to obtain the required data.

Enable MCQA on SQL Workbench/J by using the MaxCompute JDBC driver

After you configure the MaxCompute JDBC driver, you can use the MCQA feature on SQL Workbench/J by modifying the JDBC URL that you specified on the profile configuration page. For more information about how to configure the profile, see Configure MaxCompute JDBC on SQL Workbench/J.

Specify the JDBC URL in the following format: `jdbc:odps:<MaxCompute_endpoint>?project=<MaxCompute_project_name>&accessId=<AccessKey ID>&accessKey=<AccessKey Secret>&charset=UTF-8&interactiveMode=true&autoSelectLimit=1000000000"`. Parameter description:

- maxcompute_endpoint: the endpoint of the region in which your MaxCompute project resides. For more information, see Endpoints.

- maxcompute_project_name: the name of your MaxCompute project.
- AccessKey ID: the AccessKey ID that is used to access your MaxCompute project.
- AccessKey secret: the AccessKey secret that corresponds to the AccessKey ID.
- charset=UTF-8: the character set encoding format.
- interactiveMode: specifies whether to enable the MCQA feature. To enable the MCQA feature, set this parameter to `true`.
- autoSelectLimit: This parameter is required only if the number of data records exceeds 1,000,000.

Enable MCQA by using MaxCompute SDK for Java

For more information about Alibaba Cloud MaxCompute SDK for Java, see SDK for Java. You must add a specified dependency to the pom.xml file in the Maven repository. Sample dependency:

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-sdk-core</artifactId>
  <version>0.35.5-public</version>
</dependency>
```

> **Note** The version of MaxCompute SDK for Java must be V0.35.1 or later.

The following sample code shows how to create a Java program:

```
import com.aliyun.odps.Odps;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.ResultSet;
import com.aliyun.odps.sqa.*;
import java.io.IOException;
import java.util.*;
public class SQLExecutorExample {
    public static void SimpleExample() {
        // Specify the Alibaba Cloud account and project information.
        Account account = new AliyunAccount("<your_access_id>", "<your_access_key>");
        Odps odps = new Odps(account);
        odps.setDefaultProject("<your_project_name>");
        odps.setEndpoint("http://service.<regionid>.maxcompute.aliyun.com/api");
        // Prepare to build an SQLExecutor.
        SQLExecutorBuilder builder = SQLExecutorBuilder.builder();
        SQLExecutor sqlExecutor = null;
        try {
            // run in offline mode or run in interactive mode
            if (false) {
                // Create an SQLExecutor that runs offline SQL queries by default.
                sqlExecutor = builder.odps(odps).executeMode(ExecuteMode.OFFLINE).build();
            } else {
                // Create an SQLExecutor that runs SQL queries with MCQA enabled by default
. Make sure that the system automatically rolls back to the offline query mode if an SQL qu
```

```
ery with MCQA enabled fails.
                sqlExecutor = builder.odps(odps).executeMode(ExecuteMode.INTERACTIVE).fallb
ackPolicy(FallbackPolicy.alwaysFallbackPolicy()).build();
            }
            // Pass special query settings if required.
            Map<String, String> queryHint = new HashMap<>();
            queryHint.put("odps.sql.mapper.split.size", "128");
            // Submit a query job. You can pass hints.
            sqlExecutor.run("select count(1) from test_table;", queryHint);
            // List the System.out.println() interfaces that can be used to query common in
formation.
            // UUID
            System.out.println("ExecutorId:" + sqlExecutor.getId());
            // Query the Logview URL of the current query job.
            System.out.println("Logview:" + sqlExecutor.getLogView());
            // Query the instance on which the current query job is run. In interactive mod
e, multiple query jobs may be run on the same instance.
            System.out.println("InstanceId:" + sqlExecutor.getInstance().getId());
            // Query the progress of the current query job. You can check the progress bar
in the console.
            System.out.println("QueryStageProgress:" + sqlExecutor.getProgress());
            // Query the changelogs about the status of the current query job, such as roll
back messages.
            System.out.println("QueryExecutionLog:" + sqlExecutor.getExecutionLog());
            // Obtain results of query jobs by calling one of the following API operations:

            if(false) {
                // Query the results of all query jobs. The API operation that you called i
s a synchronous operation and may occupy a thread until the query succeeds or fails.
                // Write the results of all query jobs to the memory at the same time. To p
revent memory issues, we recommend that you do not perform this operation if the amount of
data is large.
                List<Record> records = sqlExecutor.getResult();
                printRecords(records);
            } else {
                // Query the ResultSet iterator of the query results. The API operation tha
t you called is a synchronous operation and may occupy a thread until the query succeeds or
fails.
                // Read the results of query jobs in several batches. We recommend that you
perform this operation if the amount of data is large.
                ResultSet resultSet = sqlExecutor.getResultSet();
                while (resultSet.hasNext()) {
                    printRecord(resultSet.next());
                }
            }
            // run another query
            sqlExecutor.run("select * from test_table;", new HashMap<>());
            if(false) {
                // Query the results of all query jobs. The API operation that you called i
s a synchronous operation and may occupy a thread until the query succeeds or fails.
                // Write the results of all query jobs to the memory at the same time. To p
revent memory issues, we recommend that you do not perform this operation if the amount of
data is large.
                List<Record> records = sqlExecutor.getResult();
                printRecords(records);
```

```
                printRecords(records);
            } else {
                // Query the ResultSet iterator of the query results. The API operation tha
t you called is a synchronous operation and may occupy a thread until the query succeeds or
fails.
                // Read the results of query jobs in several batches. We recommend that you
perform this operation if the amount of data is large.
                ResultSet resultSet = sqlExecutor.getResultSet();
                while (resultSet.hasNext()) {
                    printRecord(resultSet.next());
                }
            }
        } catch (OdpsException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (sqlExecutor != null) {
                // Close the SQLExecutor and release related resources.
                sqlExecutor.close();
            }
        }
    }
    // SQLExecutor can be reused by pool mode
    public static void ExampleWithPool() {
        // Specify the Alibaba Cloud account and project information.
        Account account = new AliyunAccount("your_access_id", "your_access_key");
        Odps odps = new Odps(account);
        odps.setDefaultProject("your_project_name");
        odps.setEndpoint("http://service.<regionid>.maxcompute.aliyun.com/api");
        // Run query jobs by using a connection pool.
        SQLExecutorPool sqlExecutorPool = null;
        SQLExecutor sqlExecutor = null;
        try {
            // Create a connection pool. Specify the connection pool size and the default e
xecution mode.
            SQLExecutorPoolBuilder builder = SQLExecutorPoolBuilder.builder();
            builder.odps(odps)
                    .initPoolSize(1) // init pool executor number
                    .maxPoolSize(5)  // max executors in pool
                    .executeMode(ExecuteMode.INTERACTIVE); // run in interactive mode
            sqlExecutorPool = builder.build();
            // Obtain an SQLExecutor from the connection pool. If no SQLExecutor is obtaine
d from the connection pool and the total number of SQLExecutors does not exceed the upper l
imit, an SQLExecutor is automatically added.
            sqlExecutor = sqlExecutorPool.getExecutor();
            // Use the SQLExecutor the same way as you use it in the preceding example.
            sqlExecutor.run("select count(1) from test_table;", new HashMap<>());
            System.out.println("InstanceId:" + sqlExecutor.getId());
            System.out.println("Logview:" + sqlExecutor.getLogView());
            List<Record> records = sqlExecutor.getResult();
            printRecords(records);
        } catch (OdpsException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        } finally {
            sqlExecutor.close();
        }
        sqlExecutorPool.close();
    }
    private static void printRecord(Record record) {
        for (int k = 0; k < record.getColumnCount(); k++) {
            if (k != 0) {
                System.out.print("\t");
            }
            if (record.getColumns()[k].getType().equals(OdpsType.STRING)) {
                System.out.print(record.getString(k));
            } else if (record.getColumns()[k].getType().equals(OdpsType.BIGINT)) {
                System.out.print(record.getBigint(k));
            } else {
                System.out.print(record.get(k));
            }
        }
    }
    private static void printRecords(List<Record> records) {
        for (Record record : records) {
            printRecord(record);
            System.out.println();
        }
    }
    public static void main(String args[]) {
        SimpleExample();
        ExampleWithPool();
    }
}
```

Enable MCQA by using MaxCompute Studio

MCQA is supported in MaxCompute Studio V3.5.0 or later. We recommend that you install the latest version of MaxCompute Studio. For more information about how to install MaxCompute Studio, see Install MaxCompute Studio.

In the SQL editor of MaxCompute Studio, set the SQL execution mode to **UseShortQueryAcceleration** or **FallbackWhenFailed** and execute query statements to enable MCQA.

- **UseShortQueryAcceleration**: Use MCQA to execute query statements.
- **FallbackWhenFailed**: Use MCQA to roll back the system to the default offline mode and execute SQL query statements again if the SQL query statements fail to be executed.

Enable MCQA by using PyODPS

MCQA is supported in PyODPS V0.10.7 or later. When you execute SQL statements, you can call the `run_sql_interactive()` method to enable MCQA. The following code shows an example:

```
odps = ODPS(...)
instance = odps.run_sql_interactive('select count(*) from test_table')
# logview
print(instance.get_logview_address())
# Optional. If you use Tunnel to download query results, the MaxCompute client does not nee
d to wait for the completion of the task instance.
# Call the wait_for_success method to wait for the completion of the task instance. You can
also call the wait_for_completion method to perform this operation. The difference between
the two methods is that an error is returned if the task instance fails when you call the w
ait_for_success method.
instance.wait_for_success()
# Download the results.
# Display alerts returned by PyODPS.
print(instance.get_warnings())
# Display results. An error is returned if the task instance fails.
print(instance.get_printable_result())
# Download results from the frontend.
for each_record in instance.open_reader():
    pass # The data type of each record in MaxCompute.
# Use Tunnel to download the results.
for each_record in instance.open_reader(tunnel=True):
    pass
```

The `run_sql_interactive()` method can contain the following parameters:

- `service_name`: optional. The name of the MCQA session.
- `service_startup_timeout`: optional. The timeout period for waiting for a session attach.
- `force_reattach`: optional. `run_sql_interactive()` automatically reuses sessions. The force_reattach parameter is used to forcibly reattach to a session.

Enable MCQA by using SQLAlchemy of PyODPS or by using a third-party tool that supports SQLAlchemy interface

PyODPS is integrated with SQLAlchemy that is used to query data in MaxCompute. To enable MCQA, you must specify the following parameters in the connection string:

- `interactive_mode`: required. This parameter specifies whether to enable the MCQA feature. Set this parameter to true.
- `reuse_odps`: optional. This parameter specifies whether to forcibly reuse connections. Set this parameter to true. This value indicates that connections are forcibly reused. Connection reuse improves the performance of some third-party tools, such as Apache SuperSet.

You can configure `fallback_policy=<policy1>,<policy2>,...` in the connection string to optimize the processing logic. The rollback policies specified by this parameter are similar to the parameters that are configured when you enable MCQA by using the MaxCompute JDBC driver. These policies are used to specify the rollback operation that is performed when query acceleration fails.

- `generic`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when an unknown error occurs.
- `noresource`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when resources are insufficient.
- `upgrading`: The default value is False. If this parameter is set to True, the system rolls back to the

offline mode during an upgrade.

- `timeout`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when a connection times out.
- `unsupported`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode when the MCQA feature is not supported.
- `default`: If you specify this parameter, the system performs a rollback based on the settings of the unsupported, upgrading, noresource, or timeout parameters. If you do not specify `fallback_policy` in the connection string, the default rollback policy is used.
- `all`: The default value is False. If this parameter is set to True, the system rolls back to the offline mode in any of the preceding scenarios. This parameter is supported only for MaxCompute JDBC driver V3.2.3 and later.

You can use the following connection string if you want to enable MCQA and forcible connection reuse, and roll back the system to the offline mode in the scenario of an upgrade, MCQA not supported, or insufficient resources:

```
odps://<access_id>:<access_key>@<project>/?endpoint=<endpoint>&interactive_mode=true&reuse_
odps=true&fallback_policy=unsupported,upgrading,noresource
```

4.CUPID references

4.1. PyODPS

4.1.1. Quick start

PyODPS is MaxCompute SDK for Python. It provides easy-to-use Python programming interfaces. Similar to pandas, PyODPS provides fast, flexible, and expressive data structures. You can use the data processing feature of PyODPS, which is similar to that of pandas, by calling the DataFrame API provided by PyODPS. This topic describes how to use PyODPS in your projects.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute and DataWorks.
- DataWorks is activated, and a workspace is created. For more information, see Create a MaxCompute project.

Context

You can develop most programs in MaxCompute by using SQL statements. However, you must use Python to develop complex business logic and user defined functions (UDFs). For example, you must use Python in the following scenarios:

- Interface connection

  An external service must provide the required information to complete authentication before the service can access a data record in MaxCompute tables by using HTTP interfaces.

- Asynchronous calls

  In most cases, the system creates a node for each of thousands of tasks with similar data processing logic. These nodes are difficult to manage and occupy excessive resources at the same time. PyODPS allows you to use queues to asynchronously run SQL tasks at a high concurrency. Queues also help you manage all nodes in a unified manner.

- UDF development

  If the built-in functions of MaxCompute cannot meet your requirement, you can develop UDFs. For example, you can develop a UDF to format data of the DECIMAL type with thousands separators.

- SQL performance improvement

  To check whether database transactions follow the first in, first out (FIFO) rule, you must compare each new record with historical records in the only transaction table. This transaction table generally contains a large number of records. If you use SQL statements to complete the comparison, the time complexity is $O(N^2)$. In addition, the SQL statements may fail to return the expected result. To resolve this issue, you can use the cache method in Python code to traverse records in the transaction table only once. In this case, the time complexity is O(N), and the efficiency is improved significantly.

- Other scenarios

  You can use Python to develop an amount allocation model. For example, you can develop a model to allocate USD 10 to three persons. In this model, you must define the logic for handling the cash over and short.

For more information about the application scenarios of PyODPS, see the following topics:

- Quick start
- Use a PyODPS node to read data from a partitioned table
- Use a PyODPS node to pass parameters
- Reference a third-party package in a PyODPS node
- Use a PyODPS node to read data from the level-1 partition of the specified table
- Use a PyODPS node to query data based on specific criteria
- Use a PyODPS node to perform sequence operations

Procedure

1. Create a PyODPS node.

   This section describes how to create a PyODPS 2 node in the DataWorks console. For more information, see Create a PyODPS 2 node.

   > **Note**
   > - The Python version of a PyODPS node is 2.7.
   > - Each PyODPS node can process a maximum of 50 MB of data and can occupy a maximum of 1 GB of memory. Otherwise, DataWorks terminates the PyODPS node. Do not write Python code that will process a large amount of data in PyODPS nodes.
   > - Writing and debugging code in DataWorks is inefficient. We recommend that you install IntelliJ IDEA locally to write code.

   i. Create a workflow.

   Log on to the DataWorks console and go to the DataStudio page. On the left side of the page, right-click **Business Flow** and click **Create Workflow**.

   ii. Create a PyODPS 2 node.

   Find the newly created workflow, right-click the workflow name, and then choose **Create > MaxCompute > PyODPS 2**. In the Create Node dialog box, specify Node Name and click **Commit**.

2. Configure and run the PyODPS 2 node.

i. Write the code of the PyODPS 2 node.

Write the test code in the code editor of the PyODPS 2 node. In this example, write the following code in the code editor, which covers a full range of table operations. For more information about table operations and SQL operations, see Tables and SQL.

```
from odps import ODPS
import sys
reload (sys)
# Set UTF-8 as the default encoding format. You must execute this statement if the
data contains Chinese characters.
sys.setdefaultencoding('utf8')
# Create a non-partitioned table named my_new_table, which contains the fields with
the specified names and of the specified data types.
# Each PyODPS node in DataWorks contains the global variable odps or o, which is th
e MaxCompute entry. You can use the entry without the need to define it. For more i
nformation, see Use PyODPS in DataWorks.
table = o.create_table('my_new_table', 'num bigint, id string', if_not_exists=True)
# Write data to the my_new_table table.
records = [[111, 'aaa'],
          [222, 'bbb'],
          [333, 'ccc'],
          [444, 'Chinese']]
o.write_table(table, records)
# Read data from the my_new_table table.
for record in o.read_table(table):
    print record[0],record[1]
# Read data from the my_new_table table by executing an SQL statement.
result = o.execute_sql('select * from my_new_table;',hints={'odps.sql.allow.fullsca
n': 'true'})
# Obtain the execution result of the SQL statement.
with result.open_reader() as reader:
    for record in reader:
        print record[0],record[1]
# Delete the table.
table.drop()
```

ii. Run the code.

After you write the code, click the icon in the top toolbar. After the code is run, you can view the running result of the PyODPS 2 node on the **Runtime Log** tab. The result in the following figure indicates that the running of the code is successful.

4.1.2. Installation guide and limits

PyODPS is MaxCompute SDK for Python. It provides the DataFrame framework and basic operations on MaxCompute objects for you to analyze data in MaxCompute by using Python. This topic describes how to install PyODPS and its limits.

Prerequisites

The following requirements are met before you install PyODPS:

- The setuptools version is 3.0 or later.
- The Requests version is 2.4.0 or later.

Procedure

1. Install pip. We recommend that you use pip, a Python package manager, to install PyODPS. For more information, see Installation. The following installation commands are for your reference:

   ```
   pip install setuptools>=3.0
   pip install requests>=2.4.0
   pip install greenlet>=0.4.10  # Optional. It accelerates Tunnel-based data upload.
   pip install cython>=0.19.0  # Optional. We recommend that you do not install Cython if you use a Windows operating system.
   ```

   > **Note**
   > - We recommend that you use Alibaba Cloud images to accelerate data download. For more information about Alibaba Cloud images, see Index of /pypi/.
   > - If you use a Windows operating system, make sure that you have installed Visual C++ and Cython of correct versions. Otherwise, you cannot accelerate Tunnel-based data upload. For more information about the versions of Visual C++ and Cython, see WindowsCompilers.

2. Run the following command to install PyODPS:

   ```
   pip install pyodps
   ```

3. Run the following command to check whether the installation is successful:

   ```
   python -c "from odps import ODPS"
   ```

4. If the Python version is not the default, you can run the following command to switch to the default version after you have installed pip:

   ```
   /home/tops/bin/python2.7 -m pip install setuptools>=3.0
   ```

Limits

- MaxCompute SQL statements have limits. For more information, see MaxCompute SQL limits.
- PyODPS nodes in DataWorks have limits. For more information, see Use PyODPS in DataWorks.
- Restricted by the sandbox, some programs that are debugged locally by using the pandas computing backend cannot be debugged in MaxCompute.

What to do next

We recommend that you install the following tools to accelerate Tunnel-based data upload:

- Greenlet 0.4.10 or later
- Cython 0.19.0 or later

4.1.3. Platform instructions

4.1.3.1. Overview

PyODPS can be called as a data development node on a data development platform such as DataWorks. The platform provides a PyODPS running environment and can schedule and run nodes. You do not need to manually create a MaxCompute entry object. To migrate PyODPS nodes from a data development platform to a manually deployed PyODPS environment, read the instructions in the following topics:

- Migrate PyODPS nodes from a data development platform to a local PyODPS environment
- Use PyODPS in DataWorks

4.1.3.2. Migrate PyODPS nodes from a data development platform to a local PyODPS environment

If you need to debug PyODPS locally or the resources on the data development platform where PyODPS is deployed cannot meet your requirements, you can migrate PyODPS nodes from the platform to a local PyODPS environment.

To migrate PyODPS nodes from a data development platform to a local PyODPS environment, perform the following steps:

1. Install PyODPS in a local environment. For more information, see Installation guide and limits.
2. Create a MaxCompute entry object in the local environment.

   You can execute the following statement on the data development platform to generate a template of the statement that is required to create a MaxCompute entry object. Then, you can modify the template to obtain the required statement.

   ```
   print("\nfrom odps import ODPS\no = ODPS(%r, '<access-key>', %r, '<endpoint>')\n" % (o.account.access_id, o.project))
   ```

   Parameter description:

   - <access-key>: the AccessKey pair of your Alibaba Cloud account.
   - <endpoint>: the endpoint of MaxCompute. For more information, see Endpoints.
   - Project name: You do not need to specify the project name in the statement for creating a MaxCompute entry object if the statement is generated by using the preceding statement.

When you migrate PyODPS nodes from DataWorks to a local environment, you can manually specify a project name in the statement for creating a MaxCompute entry object. Make sure that you understand the difference between workspaces in DataWorks and projects in MaxCompute. A DataWorks workspace in basic mode maps a MaxCompute project named in the <Project name> format, and provides a production environment for jobs in the MaxCompute project. A DataWorks workspace in standard mode maps a MaxCompute project named in the <Project name>_dev format and a MaxCompute project named in the <Project name> format. This workspace provides a development environment for jobs in the MaxCompute project named in the <Project name>_dev format and a production environment for jobs in the MaxCompute project named in the <Project name> format. For more information, see 简单模式和标准模式的区别.

3. Place the obtained statement at the beginning of all code.

4.1.3.3. Use PyODPS in DataWorks

This topic describes how to use PyODPS in DataWorks.

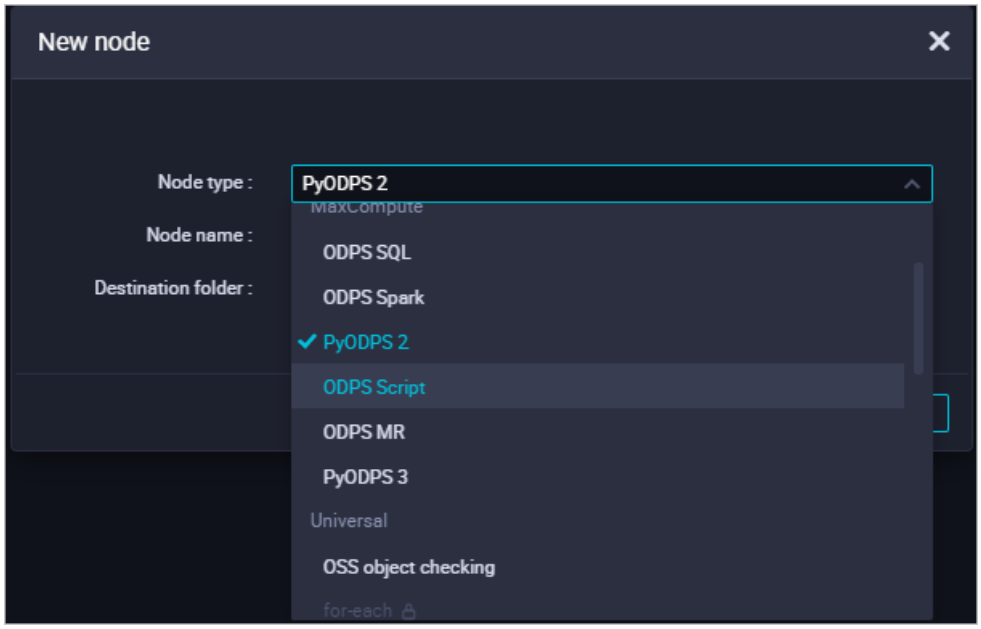
Go to the Data Analytics page of DataWorks

You can go to the **Data Analytics** page of DataWorks to create a PyODPS node.

PyODPS nodes are classified into two types: PyODPS 2 and PyODPS 3. The two types of PyODPS nodes use different Python versions at the underlying layer. PyODPS 2 nodes use Python 2 and PyODPS 3 nodes use Python 3. You can create a PyODPS node based on the Python version in use.

For more information about how to create a PyODPS node, see Create a PyODPS 2 node and Create a PyODPS 3 node.

For more examples, see Use a PyODPS node to segment Chinese text based on Jieba.



Limits

- Each PyODPS node can process a maximum of 50 MB of data and can occupy a maximum of 1 GB of memory. Otherwise, DataWorks terminates the PyODPS node. Do not write unnecessary Python data processing code in PyODPS tasks.
- The efficiency of writing and debugging code in DataWorks is low. We recommend that you install an

integrated development environment (IDE) on your machine to write code.

- To avoid excess pressure on the gateway of DataWorks, DataWorks limits the CPU utilization and memory usage. If the system displays **Got killed**, the memory usage exceeds the limit and the system terminates the related processes. Therefore, we recommend that you do not perform local data operations. However, the limits on the memory usage and CPU utilization do not apply to SQL or DataFrame nodes, except to_pandas, that are initiated by PyODPS.
- Functions may be limited in the following aspects due to the lack of packages such as matplotlib:
  - The use of the plot function of DataFrame is affected.
  - DataFrame user-defined functions (UDFs) can be used only after they are submitted to MaxCompute. As required by the Python sandbox, you can use only pure Python libraries and the NumPy library to run UDFs. Other third-party libraries such as pandas cannot be used.
  - However, you can use the NumPy and pandas libraries that are pre-installed in DataWorks to run non-UDFs. Third-party packages that contain binary code are not supported.
- For compatibility reasons, options.tunnel.use_instance_tunnel is set to False in DataWorks by default. If you want to enable InstanceTunnel globally, you must set this parameter to True.
- For implementation reasons, the Python atexit package is not supported. You must use try-finally to implement relevant features.

MaxCompute entry

Each PyODPS node in DataWorks contains the global variable `odps` or `o`, which is the MaxCompute entry. You do not need to specify the MaxCompute entry.

```
print(o.exist_table('pyodps_iris'))
```

Execute SQL statements

You can execute SQL statements in the PyODPS node. For more information, see SQL.

By default, InstanceTunnel is disabled in DataWorks. In this case, instance.open_reader is run by using the Result interface, and a maximum of 10,000 data records can be read. You can use reader.count to obtain the number of data records. If you need to iteratively obtain all data, you must remove the limit on the amount of data. You can execute the following statements to enable InstanceTunnel and remove the limit.

```
options.tunnel.use_instance_tunnel = True
options.tunnel.limit_instance_tunnel = False  # Remove the limit on the amount of data.
with instance.open_reader() as reader:
    # Use InstanceTunnel to read all data.
```

You can also add `tunnel=True` to open_reader to enable InstanceTunnel for the current open_reader operation. You can add `limit=False` to open_reader to remove the limit on the amount of data for the current open_reader operation.

```
with instance.open_reader(tunnel=True, limit=False) as reader:
# The current open_reader operation is performed by using InstanceTunnel, and all data can
be read.
```

DataFrame

- Execution method

  DataFrame API operations are not automatically performed. These operations can be performed only when you explicitly call an automatically executed method.

```
from odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
for record in iris[iris.sepal_width < 3].execute():  # Call an automatically executed met
hod to process each data record.
```

  To call an automatically executed method for data display, set `options.interactive` to True.

```
from odps import options
from odps.df import DataFrame
options.interactive = True  # Set options.interactive to True at the beginning of the cod
e.
iris = DataFrame(o.get_table('pyodps_iris'))
print(iris.sepal_width.sum())  # The method is executed immediately after the system disp
lays information.
```

- Display details

  To display details, you must set `options.verbose` to True. By default, this parameter is set to True in DataWorks. The system displays details such as the Logview URL during the running process.

Obtain scheduling parameters

Different from SQL nodes in DataWorks, a PyODPS node does not replace strings such as ${param_name} in code. Instead, the PyODPS node adds a dictionary named `args` as a global variable before it runs the code. You can obtain the scheduling parameters from the dictionary. This way, Python code is not affected. For example, if you set `ds=${yyyymmdd}` under **Schedule > Parameter** in DataWorks, you can run the following commands to obtain the parameter value:

```
print('ds=' + args['ds'])
ds=20161116
```

> **Note** You can run the following command to obtain the partition named `ds=${yyyymmdd}`:
>
> `o.get_table('table_name').get_partition('ds=' + args['ds'])`

4.1.4. Basic operations

4.1.4.1. Overview

PyODPS supports basic operations on MaxCompute objects. You can use Python-compliant programming methods to perform operations on MaxCompute.

PyODPS allows you to perform basic operations on the following MaxCompute objects:

- Projects
- Tables

- SQL
- Task instances
- Resources
- Functions

4.1.4.2. Projects

This topic describes how to perform basic operations on projects by using PyODPS.

A project is the basic organizational unit of MaxCompute. For more information, see Project.

You can perform the following basic operations on a project:

- Obtain a project: You can call the `get_project()` method of a MaxCompute entry object to obtain a specific project.

  ```
  project = o.get_project('my_project')  # Obtain the specified project.
  project = o.get_project()              # Obtain the current project.
  ```

  To obtain a specific project, you must set the project_name parameter to the name of the project for the get_project() method. If you do not specify a project name for the method, the method returns the current project.
- Check whether a project exists: You can call the `exist_project()` method to check whether a specific project exists.

4.1.4.3. Tables

This topic describes the basic operations as well as other operations that you can perform on a MaxCompute table. For example, you can create a table, create a table schema, synchronize table updates, obtain table data, delete a table, manage table partitions, and convert a table to a DataFrame.

Basic operations

A table is the data storage unit in MaxCompute. For more information, see Table. You can perform the following basic operations on tables:

- Obtain all tables in the current project by calling the `list_tables()` method of a MaxCompute entry object.

  ```
  # Obtain all tables in the current project.
  for table in o.list_tables():
  ```
- Check whether a specific table exists by calling the `exist_table()` method of a MaxCompute entry object.
- Obtain a specific table by calling the `get_table()` method of a MaxCompute entry object.

```
t = o.get_table('table_name')
t.schema
odps.Schema {
  c_int_a                 bigint
  c_int_b                 bigint
  c_double_a              double
  c_double_b              double
  c_string_a              string
  c_string_b              string
  c_bool_a                boolean
  c_bool_b                boolean
  c_datetime_a            datetime
  c_datetime_b            datetime
}
t.lifecycle
-1
print(t.creation_time)
2014-05-15 14:58:43
t.is_virtual_view
False
t.size
1408
t.comment
'table_name Table Comment'
t.schema.columns
[<column c_int_a, type bigint>,
 <column c_int_b, type bigint>,
 <column c_double_a, type double>,
 <column c_double_b, type double>,
 <column c_string_a, type string>,
 <column c_string_b, type string>,
 <column c_bool_a, type boolean>,
 <column c_bool_b, type boolean>,
 <column c_datetime_a, type datetime>,
 <column c_datetime_b, type datetime>]
t.schema['c_int_a']
<column c_int_a, type bigint>
t.schema['c_int_a'].comment
'Comment of column c_int_a'
```

You can obtain tables across projects by specifying the `project` parameter.

```
t = o.get_table('table_name', project='other_project')
```

Create a table schema

You can use one of the following methods to create a table schema:

- Create a schema based on table columns and optional partitions.

```
from odps.models import Schema, Column, Partition
columns = [Column(name='num', type='bigint', comment='the column'),
           Column(name='num2', type='double', comment='the column2')]
partitions = [Partition(name='pt', type='string', comment='the partition')]
schema = Schema(columns=columns, partitions=partitions)
schema.columns
[<column num, type bigint>,
 <column num2, type double>,
 <partition pt, type string>]
schema.partitions
[<partition pt, type string>]
schema.names  # Obtain the names of non-partitioned fields.
['num', 'num2']
schema.types  # Obtain the data types of non-partitioned fields.
[bigint, double]
```

- Create a schema by calling the `Schema.from_lists()` method. This method is more convenient, but you cannot directly set comments for columns and partitions.

```
schema = Schema.from_lists(['num', 'num2'], ['bigint', 'double'], ['pt'], ['string'])
schema.columns
[<column num, type bigint>,
 <column num2, type double>,
 <partition pt, type string>]
```

Create a table

You can call the `create_table()` method to create a table by using one of the following methods:

- Use a table schema to create a table.

```
table = o.create_table('my_new_table', schema)
table = o.create_table('my_new_table', schema, if_not_exists=True)  # Create the table on
ly if no table with the same name exists.
table = o.create_table('my_new_table', schema, lifecycle=7)  # Set the lifecycle of the t
able.
```

- Create a table by specifying the names and data types of the fields to be contained in the table.

```
# Create a non-partitioned table.
table = o.create_table('my_new_table', 'num bigint, num2 double', if_not_exists=True)
# Create a partitioned table with the specified table columns and partition fields.
table = o.create_table('my_new_table', ('num bigint, num2 double', 'pt string'), if_not_e
xists=True)
```

By default, when you create a table, you can only use the BIGINT, DOUBLE, DECIMAL, STRING, DATETIME, BOOLEAN, MAP, and ARRAY data types. If you need to use other data types such as TINYINT and STRUCT, you must set `options.sql.use_odps2_extension` to True. Example:

```
from odps import options
options.sql.use_odps2_extension = True
table = o.create_table('my_new_table', 'cat smallint, content struct<title:varchar(100),
body:string>')
```

Synchronize table updates

After another program updates a table, for example, the table schema, you can call the `reload()` method to synchronize the update.

```
table.reload()
```

Insert a record to a table

A record refers to a single row in a table. You can call the `new_record()` method to insert a record to a specific table.

```
t = o.get_table('mytable')
r = t.new_record(['val0', 'val1'])  # The number of values must be equal to the number of f
ields in the table schema.
r2 = t.new_record()     # You can leave the value empty.
r2[0] = 'val0' # Set a value based on an offset.
r2['field1'] = 'val1'  # Set a value based on the field name.
r2.field1 = 'val1'  # Set a value based on an attribute.
print(record[0])  # Obtain the value at position 0.
print(record['c_double_a'])  # Obtain a value based on a field.
print(record.c_double_a)  # Obtain a value based on an attribute.
print(record[0: 3])  # Slice data.
print(record[0, 2, 3])  # Obtain values at multiple positions.
print(record['c_int_a', 'c_double_a'])  # Obtain values based on multiple fields.
```

Write data to a table

- Call the `write_table()` method of a MaxCompute entry object to write data to a table.

  ```
  records = [[111, 'aaa', True],                # A list can be specified.
             [222, 'bbb', False],
             [333, 'ccc', True],
             [444, 'Chinese', False]]
  o.write_table('test_table', records, partition='pt=test', create_partition=True)
  ```

  > **Note**
  > - Each time you call the `write_table()` method, MaxCompute generates a file on the server. This operation is time-consuming. If a large number of files are concurrently generated, subsequent queries will be less efficient. We recommend that you write multiple records at a time or provide a generator object if you use the write_table() method.
  > - If you call the `write_table()` method to write data to a table, new data will be appended to existing data. PyODPS does not provide options to overwrite existing data. You must manually delete the data that you want to overwrite. For a non-partitioned table, you must call the `table.truncate()` method to delete data. For a partitioned table, you must delete partitions first and then create partitions again.

- Call the `open_writer()` method to write data to a table.

```
with t.open_writer(partition='pt=test') as writer:
records = [[111, 'aaa', True],                # A list can be specified.
          [222, 'bbb', False],
          [333, 'ccc', True],
          [444, 'Chinese', False]]
writer.write(records)  # records can be iterable objects.
with t.open_writer(partition='pt1=test1,pt2=test2') as writer: # Write data in multi-leve
l partitioning mode.
records = [t.new_record([111, 'aaa', True]),   # Record objects can be used.
           t.new_record([222, 'bbb', False]),
           t.new_record([333, 'ccc', True]),
           t.new_record([444, 'Chinese', False])]
writer.write(records)
```

If the specified partition does not exist, set the `create_partition` parameter to True to create a partition. Example:

```
with t.open_writer(partition='pt=test', create_partition=True) as writer:
     records = [[111, 'aaa', True],                # A list can be specified.
                [222, 'bbb', False],
                [333, 'ccc', True],
                [444, 'Chinese', False]]
   writer.write(records)  # records can be iterable objects.
```

- Use multiple processes to concurrently write data to a table.

  If multiple processes concurrently write data to a table, each process shares the same session ID but writes data in a separate block. Each block maps a file on the server. After all the processes finish writing data, the main process submits the data.

```
import random
from multiprocessing import Pool
from odps.tunnel import TableTunnel
def write_records(session_id, block_id):
    # Create a session with the specified session ID.
    local_session = tunnel.create_upload_session(table.name, upload_id=session_id)
    # Create a writer with the specified block ID.
    with local_session.open_record_writer(block_id) as writer:
        for i in range(5):
            # Generate data and write the data to the corresponding block.
            record = table.new_record([random.randint(1, 100), random.random()])
            writer.write(record)
if __name__ == '__main__':
    N_WORKERS = 3
    table = o.create_table('my_new_table', 'num bigint, num2 double', if_not_exists=True)
    tunnel = TableTunnel(o)
    upload_session = tunnel.create_upload_session(table.name)
    # Each process uses the same session ID.
    session_id = upload_session.id
    pool = Pool(processes=N_WORKERS)
    futures = []
    block_ids = []
    for i in range(N_WORKERS):
        futures.append(pool.apply_async(write_records, (session_id, i)))
        block_ids.append(i)
    [f.get() for f in futures]
    # Submit the data in all the blocks.
    upload_session.commit(block_ids)
```

Obtain table data

You can use one of the following methods to obtain data from a table:

- Call the `read_table()` method of a MaxCompute entry object.

```
for record in o.read_table('test_table', partition='pt=test'):
# Process a record.
```

- Call the `head()` method to obtain less than 10,000 data records from the beginning of a table.

```
t = o.get_table('table_name')
# Process each record.
for record in t.head(3):
```

- Call the `open_reader()` method.
  - Open the reader with a WITH clause.

```
with t.open_reader(partition='pt=test') as reader:
count = reader.count
for record in reader[5:10]  # You can execute the statement multiple times until all re
cords are read. The number of records is specified by count. You can change it to paral
lel-operation code.
# Process a record.
```

- Open the reader without a WITH clause.

```
reader = t.open_reader(partition='pt=test')
count = reader.count
for record in reader[5:10]  # You can execute the statement multiple times until all re
cords are read. The number of records is specified by count. You can change it to paral
lel-operation code.
# Process a record.
```

Delete a table

You can call the `delete_table()` method to delete an existing table.

```
o.delete_table('my_table_name', if_exists=True)  # Delete a table only if the table exists.
 t.drop()  # Call the drop() method to delete a table if the table exists.
```

Create a DataFrame

PyODPS provides the DataFrame framework, which allows you to conveniently query and manage MaxCompute data. For more information, see DataFrame. You can call the `to_df()` method to convert a table to a DataFrame.

```
table = o.get_table('my_table_name')
df = table.to_df()
```

Manage table partitions

- Check whether a table is partitioned.

```
if table.schema.partitions:
 print('Table %s is partitioned.' % table.name)
```

- Iterate over all the partitions in a table.

```
for partition in table.partitions:
     print(partition.name)
for partition in table.iterate_partitions(spec='pt=test'):
# Iterate over level-2 partitions.
```

- Check whether a partition exists.

```
table.exist_partition('pt=test,sub=2015')
```

- Obtain a partition.

```
partition = table.get_partition('pt=test')
 print(partition.creation_time)
2015-11-18 22:22:27
 partition.size
0
```

- Create a partition.

```
t.create_partition('pt=test', if_not_exists=True)  # Create a partition only if no partit
ion with the same name exists.
```

- Delete a partition.

```
t.delete_partition('pt=test', if_exists=True)  # Delete a partition only if the partition exists.
partition.drop()  # Call the drop() method to delete a partition if the partition exists.
```

Upload and download data by using MaxCompute Tunnel

MaxCompute Tunnel is the data tunnel of MaxCompute. You can use Tunnel to upload data to or download data from MaxCompute.

- Example of data upload

```
from odps.tunnel import TableTunnel
table = o.get_table('my_table')
tunnel = TableTunnel(odps)
upload_session = tunnel.create_upload_session(table.name, partition_spec='pt=test')
with upload_session.open_record_writer(0) as writer:
    record = table.new_record()
    record[0] = 'test1'
    record[1] = 'id1'
    writer.write(record)
    record = table.new_record(['test2', 'id2'])
    writer.write(record)
upload_session.commit([0])
```

- Example of data download

```
from odps.tunnel import TableTunnel
tunnel = TableTunnel(odps)
download_session = tunnel.create_download_session('my_table', partition_spec='pt=test')
with download_session.open_record_reader(0, download_session.count) as reader:
    for record in reader:
# Process each record.
```

> **Note**
>
> - PyODPS does not allow you to upload data by using foreign tables. For example, you cannot upload data from Object Storage Service (OSS) or Tablestore by using foreign tables.
> - We recommend that you use the write and read interfaces of tables instead of Tunnel.
> - In a CPython environment, PyODPS compiles C programs during installation to accelerate Tunnel-based upload and download.

4.1.4.4. SQL

PyODPS supports MaxCompute SQL queries and provides methods to obtain execution results.

Execute SQL statements

You can call the `execute_sql()` and `run_sql()` methods of a MaxCompute entry object to execute SQL statements for creating task instances. For more information, see Task instances.

Parameters

- statement: the SQL statement to execute.

  > **Note** Not all SQL statements can be executed by using methods of the MaxCompute entry object. Some SQL statements that are executable on the MaxCompute client may fail to be executed by using the `execute_sql()` and `run_sql()` methods. You must use other methods to execute non-DDL or non-DML statements. For example, you must use the `run_security_query` method to execute GRANT or REVOKE statements, and use the `run_xflow` or `execute_xflow` method to call API operations.

- hints: the runtime parameters. The hints parameter is of the DICT type.

Examples

- Execute SQL statements.

```
o.execute_sql('select * from table_name')  # Execute the statement in synchronous mode. O
ther instances are blocked until the execution of the SQL statement is complete.
instance = o.run_sql('select * from table_name')  # Execute the statement in asynchronous
mode.
print(instance.get_logview_address())  # Obtain the Logview URL of an instance.
instance.wait_for_success()  # Other instances are blocked until the execution of the SQL
statement is complete.
```

- Set runtime parameters for SQL statements.

```
o.execute_sql('select * from pyodps_iris', hints={'odps.sql.mapper.split.size': 16})
```

  If you set the `sql.settings` parameter globally, the runtime parameters are automatically added each time you execute the statement.

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from pyodps_iris')  # The hints parameter is automatically set ba
sed on global settings.
```

Obtain the execution results of SQL statements

You can call the `open_reader` method to obtain the execution results of SQL statements. When the query results are being read, the following situations may occur:

- The SQL statements return structured data.

```
with o.execute_sql('select * from table_name').open_reader() as reader:
    for record in reader:
    # Process each record.
```

- If the `DESC` command is executed, you can use `reader.raw` to obtain the raw SQL query results.

```
with o.execute_sql('desc table_name').open_reader() as reader:
    print(reader.raw)
```

If you specify `options.tunnel.use_instance_tunnel == True` when you call the `open_reader` method, PyODPS automatically calls InstanceTunnel. However, if you use an earlier version of MaxCompute or an error occurs when PyODPS calls InstanceTunnel, PyODPS generates an alert and automatically downgrades the call object to the old Result interface. You can identify the cause of the downgrade based on the alert information. If the result of InstanceTunnel does not meet your expectation, set the options.tunnel.use_instance_tunnel parameter to False. When you call the `open_reader` method, you can specify whether to call InstanceTunnel or the Result interface by setting the tunnel parameter.

```
# Call InstanceTunnel.
with o.execute_sql('select * from table_name').open_reader(tunnel=True) as reader:
    for record in reader:
    # Process each record.
# Call the Result interface.
with o.execute_sql('select * from table_name').open_reader(tunnel=False) as reader:
    for record in reader:
    # Process each record.
```

By default, PyODPS does not limit the amount of data that can be read from an instance. However, for a protected project, the amount of data that can be downloaded by using Tunnel is limited. If you do not specify `options.tunnel.limit_instance_tunnel`, the limit is automatically enabled, and the number of data records that can be downloaded is limited based on the project configurations. In most cases, a maximum of 10,000 data records can be downloaded. If you want to manually limit the number of data records that can be downloaded, you can add a limit option to the `open_reader` method. For example, you can specify `open_reader(tunnel = True, limit=1000)` to limit the number of data records that can be downloaded to 1,000 rows. You can also set `options.tunnel.limit_instance_tunnel` to True in the Python script.

If your MaxCompute version supports only the old Result interface and you want to read all the execution results of SQL statements, you can write the execution results to another table and then use open_reader to read data from the table. This operation is subject to the security configurations of your MaxCompute project.

In PyODPS V0.7.7.1 or later, you can use the `open_reader` method to call InstanceTunnel to obtain all data. For more information, see InstanceTunnel.

```
instance = o.execute_sql('select * from movielens_ratings limit 20000')
with instance.open_reader(tunnel=True) as reader:
    print(reader.count)
    # for record in reader: Traverse the 20,000 data records. In this example, only 10 data
records are obtained based on data slicing.
    for record in reader[:10]:
        print(record)
```

Configure resource aliases

If the resources referenced by a user-defined function (UDF) dynamically change, you can configure an alias for the old resource and use the alias as the name of the new resource. This way, you do not need to delete the UDF or create another UDF.

```
from odps.models import Schema
myfunc = '''\
from odps.udf import annotate
from odps.distcache import get_cache_file
@annotate('bigint->bigint')
class Example(object):
    def __init__(self):
        self.n = int(get_cache_file('test_alias_res1').read())
    def evaluate(self, arg):
        return arg + self.n
'''
res1 = o.create_resource('test_alias_res1', 'file', file_obj='1')
o.create_resource('test_alias.py', 'py', file_obj=myfunc)
o.create_function('test_alias_func',
                  class_type='test_alias.Example',
                  resources=['test_alias.py', 'test_alias_res1'])
table = o.create_table(
    'test_table',
    schema=Schema.from_lists(['size'], ['bigint']),
    if_not_exists=True
)
data = [[1, ], ]
# Write a row of data that contains only one value: 1.
o.write_table(table, 0, [table.new_record(it) for it in data])
with o.execute_sql(
    'select test_alias_func(size) from test_table').open_reader() as reader:
    print(reader[0][0])
res2 = o.create_resource('test_alias_res2', 'file', file_obj='2')
# Set the alias of resource res1 as the name of resource res2. Without the need to modify t
he UDF or resource.
with o.execute_sql(
    'select test_alias_func(size) from test_table',
    aliases={'test_alias_res1': 'test_alias_res2'}).open_reader() as reader:
    print(reader[0][0])
```

Execute SQL statements in an interactive environment

You can use SQL plug-ins to execute SQL statements or run parameterized queries in IPython and Jupyter. For more information, see IPython and Jupyter Notebook.

Set biz_id

In some situations, you must specify `biz_id` for the SQL statement to execute. Otherwise, an error occurs. In this case, you can set `biz_id` in the global `options` to fix the error.

```
from odps import options
options.biz_id = 'my_biz_id'
o.execute_sql('select * from pyodps_iris')
```

4.1.4.5. Task instances

A task, such as an SQL task, is the basic compute unit in MaxCompute. This topic describes the basic operations on task instances in PyODPS.

Basic operations

When a task is triggered, a MaxCompute instance is created for the task. For more information, see Instance. You can perform the following operations on MaxCompute instances:

- `list_instances()` : obtains all instances in a specific project.
- `exist_instance()` : checks whether a specific instance exists.
- `get_instance()` : obtains a specific instance.
- `stop_instance()` : stops a specific instance.

> **Note** You can also call the `stop()` method to stop a specific instance.

Example

```
for instance in o.list_instances():
    print(instance.id)
    o.exist_instance('my_instance_id')
```

Obtain the Logview URL of an instance

- For an SQL task instance, you can call the `get_logview_address` method to obtain its Logview URL.

```
# Execute an SQL statement to obtain a specific instance. Then, call the get_logview_addr
ess method to obtain the Logview URL of the instance.
instance = o.run_sql('desc pyodps_iris')
print(instance.get_logview_address())
# Obtain the Logview URL of an instance with the specified instance ID.
instance = o.get_instance('2016042605520945g9k5pvyi2')
print(instance.get_logview_address())
```

- For an XFlow task instance, you must enumerate its sub-instances and then obtain the Logview URL of each sub-instance.

```
instance = o.run_xflow('AppendID', 'algo_public', {'inputTableName': 'input_table', 'outp
utTableName': 'output_table'})
for sub_inst_name, sub_inst in o.get_xflow_sub_instances(instance).items():
    print('%s: %s' % (sub_inst_name, sub_inst.get_logview_address()))
```

Obtain the status of an instance

An instance can be in the `Running` , `Suspended` , or `Terminated` state. You can use the following methods to obtain the status of an instance:

- `status` : obtains the status of an instance.
- `is_terminated` : checks whether the execution of the current instance is completed.
- `is_successful` : checks whether the execution of the current instance is successful. If the instance is running or fails, False is returned.

```
instance = o.get_instance('2016042605520945g9k5pvyi2')
instance.status
<Status.TERMINATED: 'Terminated'>
    from odps.models import Instance
    instance.status == Instance.Status.TERMINATED
    True
    instance.status.value
    'Terminated'
```

Note You can call the `wait_for_completion` method to ask the system to return the status until the execution of the instance is completed. The `wait_for_success` method functions similarly. The difference is that if the instance fails, an exception is thrown.

Perform operations on sub-instances

A running instance may contain one or more sub-instances. You can call the following methods to perform specific operations on sub-instances:

- `get_task_names`: obtains all sub-instances. This method returns the names of all sub-instances.

```
instance.get_task_names()
['SQLDropTableTask']
```

- `get_task_result`: obtains the execution result of a specific sub-instance. This method returns the execution result of each sub-instance in the form of a dictionary.

```
instance = o.execute_sql('select * from pyodps_iris limit 1')
instance.get_task_names()
['AnonymousSQLTask']
instance.get_task_result('AnonymousSQLTask')
'"sepallength","sepalwidth","petallength","petalwidth","name"\n5.1,3.5,1.4,0.2,"Iris-seto
sa"\n'
instance.get_task_results()
OrderedDict([('AnonymousSQLTask', "sepallength","sepalwidth","petallength","petalwidth","
name"\n5.1,3.5,1.4,0.2,"Iris-setosa"\n')])
```

- `get_task_progress`: obtains the current running progress of a specific sub-instance when the instance is running.

```
while not instance.is_terminated():
    for task_name in instance.get_task_names():
        print(instance.id, instance.get_task_progress(task_name).get_stage_progress_forma
tted_string())
        time.sleep(10)
20160519101349613gzbzufck2 2016-05-19 18:14:03 M1_Stg1_job0:0/1/1[100%]
```

4.1.4.6. Resources

PyODPS supports file and table resources. This topic describes the common operations on resources in PyODPS.

Basic operations

- `list_resources()`: lists all resources in a specific project.
- `exist_resource()`: checks whether a specific resource exists.
- `delete_resource()`: deletes a specific resource. You can also call the `drop()` method to delete a resource.
- `create_resource()`: creates a resource.
- `open_resource()`: reads a resource.

Manage file resources

Files of basic file types and the .py, .jar, and .archive types are supported.

- Create a file resource

  To create a file resource, you can call the `create_resource()` method with a resource name, a file type, and a file-like object or string specified.

```
resource = o.create_resource('test_file_resource', 'file', file_obj=open('/to/path/file')
)  # Create a file resource with a file-like object.
resource = o.create_resource('test_py_resource', 'py', file_obj='import this')  # Create
a file resource with a string.
```

- Read and modify a file resource

  You can use one of the following methods to open a file resource:

  - Call the `open()` method for the file resource.
  - Call the `open_resource()` method of a MaxCompute entry object.

  An opened file resource is a file-like object. Similar to the `open()` method that is built in Python, the preceding two methods also allow you to specify the opening mode of a file resource.

```
with resource.open('r') as fp:  # Open the file in read mode.
     content = fp.read()  # Read all content of the file.
     fp.seek(0)  # Move the file pointer to the beginning of the file.
     lines = fp.readlines()  # Read multiple lines.
     fp.write('Hello World')  # Error. Data cannot be written to a file in read mode.
 with o.open_resource('test_file_resource', mode='r+') as fp:  # Open the file in read/wr
ite mode.
     fp.read()
     fp.tell()  # Locate the current position of the file pointer.
     fp.seek(10)
     fp.truncate()  # Truncate the file to the specified length.
     fp.writelines(['Hello\n', 'World\n'])  # Write multiple lines to the file.
     fp.write('Hello World')
     fp.flush()  # Manually call the method to submit the update to MaxCompute.
```

  PyODPS supports the following file opening modes:

  - `r`: read mode. The file can be opened but data cannot be written to it.
  - `w`: write mode. Data can be written to the file, but data in the file cannot be read. If a file is opened in write mode, the file content is cleared first.
  - `a`: append mode. Data can be added to the end of the file.
  - `r+`: read/write mode. You can read data from and write data to the file.

- `w+`: This mode is similar to `r+`, but the file content is cleared first when the file is opened.
- `a+`: This mode is similar to `r+`, but data can be written only to the end of the file.

In PyODPS, you can also open a file resource in binary mode. For example, some compressed files must be opened in binary mode.

- `rb`: binary read mode
- `r+b`: binary read/write mode

Table resources

- Create a table resource

```
o.create_resource('test_table_resource', 'table', table_name='my_table', partition='pt=te
st')
```

- Update a table resource

```
table_resource = o.get_resource('test_table_resource')
table_resource.update(partition='pt=test2', project_name='my_project2')
```

- Obtain a table and a partition of the table

```
table_resource = o.get_resource('test_table_resource')
table = table_resource.table
print(table.name)
partition = table_resource.partition
print(partition.spec)
```

- Read data from and write data to a table

```
table_resource = o.get_resource('test_table_resource')
with table_resource.open_writer() as writer:
    writer.write([0, 'aaaa'])
    writer.write([1, 'bbbbb'])
    with table_resource.open_reader() as reader:
        for rec in reader:
            print(rec)
```

4.1.4.7. Functions

You can create user defined functions (UDFs) and use them in MaxCompute SQL.

Basic operations

- `list_functions()`: obtains all functions in a specific project.
- `exist_function()`: checks whether a specific function exists.
- `get_function()`: obtains a specific function.
- `create_function()`: creates a function.
- `delete_function()`: deletes a specific function.

Create a function

You can call the `create_function()` method of a MaxCompute entry object to create a function. The following code shows examples of how to create a function by using a resource in the current project and in another project:

```
# Reference a resource in the current project.
resource = o.get_resource('my_udf.py')
function = o.create_function('test_function', class_type='my_udf.Test', resources=[resource
])
# Reference a resource in another project.
resource2 = o.get_resource('my_udf.py', project='another_project')
function2 = o.create_function('test_function2', class_type='my_udf.Test', resources=[resour
ce2])
```

Delete a function

You can call the `delete_function()` method of a MaxCompute entry object to delete a function. You can also call the `drop()` method to delete a function. The following code shows examples of how to delete a function by using the delete_function() and drop() methods:

```
o.delete_function('test_function')
function.drop()  # Call the drop() method if the function exists.
```

Update a function

You can call the `update()` method to update a function. The following code shows an example:

```
function = o.get_function('test_function')
new_resource = o.get_resource('my_udf2.py')
function.class_type = 'my_udf2.Test'
function.resources = [new_resource, ]
function.update()  # Update the function.
```

4.1.4.8. SQLAlchemy

PyODPS is integrated with SQLAlchemy. You can use SQLAlchemy to query data in MaxCompute. This topic describes how to connect SQLAlchemy to a MaxCompute project and call the SQLAlchemy interface.

Connect SQLAlchemy to a MaxCompute project

Syntax

```
from sqlalchemy import create_engine
engine = create_engine('odps://<access_id>:<access_key>@<project>/?endpoint=<endpoint>')
conn = engine.connect()
```

- access_id: the AccessKey ID that is used to access the MaxCompute project.

  You can obtain the AccessKey ID from the AccessKey Management page.
- access_key: the AccessKey secret that corresponds to the AccessKey ID.

You can obtain the AccessKey secret from the AccessKey Management page.

- project: the name of the MaxCompute project that you want to access.

  This parameter specifies the name of your MaxCompute project instead of the DataWorks workspace to which the MaxCompute project corresponds. Log on to the MaxCompute console. In the top navigation bar, select a region and view the name of your MaxCompute project on the **Project management** tab.

- endpoint: the endpoint of the region where your MaxCompute project resides.

  For more information about the endpoints of MaxCompute in different regions, see Endpoints.

For the existing ODPS object o, if you call `o.to_global()` to configure the object as a global object, you do not need to specify the preceding parameters in the connection string. Sample statements:

```
from sqlalchemy import create_engine
o.to_global()  # set ODPS object as global one
engine = create_engine('odps://')
```

Call the SQLAlchemy interface

After you connect SQLAlchemy to a MaxCompute project, you can call the SQLAlchemy interface. The following statements show how to create tables, insert data into tables, and query data from tables.

- Create a table

```
from sqlalchemy import Table, Column, Integer, String, MetaData
metadata = MetaData()
users = Table('users', metadata,
    Column('id', Integer),
    Column('name', String),
    Column('fullname', String),
)
metadata.create_all(engine)
```

- Insert data

```
ins = users.insert().values(id=1, name='jack', fullname='Jack Jones')
conn.execute(ins)
```

- Query data

```
from sqlalchemy.sql import select
s = select([users])
result = conn.execute(s)
for row in result:
print(row)
(1, 'jack', 'Jack Jones')
```

4.1.5. Configurations

This topic describes the configuration options provided by PyODPS.

You can use `odps.options` to obtain the configuration options provided by PyODPS.

```
from odps import options
# Set the lifecycle option to specify the lifecycle of all output tables.
options.lifecycle = 30
# Set the tunnel.string_as_binary option to True to use bytes instead of Unicode to downloa
d data of the STRING type.
options.tunnel.string_as_binary = True
# When you execute PyODPS DataFrames in MaxCompute, you can refer to the following configur
ation to set the limit to a relatively large value during a sort operation.
options.df.odps.sort.limit = 100000000
```

General configurations

Option	Description	Default value
end_point	The endpoint of MaxCompute.	None
default_project	The default project.	None
log_view_host	The hostname of Logview.	None
log_view_hours	The retention time of Logview. Unit: hours.	24
local_timezone	The time zone that is used. True indicates local time, and False indicates UTC. The time zone of `pytz` can also be used.	None
lifecycle	The lifecycle of all tables.	None
temp_lifecycle	The lifecycle of temporary tables.	1
biz_id	The user ID.	None
verbose	Specifies whether to display logs.	False
verbose_log	The log receiver.	None
chunk_size	The size of the write buffer.	1496
retry_times	The number of request retries.	4
pool_connections	The number of cached connections in the connection pool.	10
pool_maxsize	The maximum capacity of the connection pool.	10
connect_timeout	The connection timeout period.	5
read_timeout	The read timeout period.	120

Option	Description	Default value
api_proxy	The API proxy server.	None
data_proxy	The data proxy server.	None
completion_size	The limit on the number of object completion listing items.	10
notebook_repr_widget	Specifies whether to use interactive graphs.	True
sql.settings	Global hints for MaxCompute SQL.	None
sql.use_odps2_extension	Specifies whether to enable MaxCompute 2.0 language extension.	False

Data upload and download configurations

Option	Description	Default value
tunnel.endpoint	The endpoint of MaxCompute Tunnel.	None
tunnel.use_instance_tunnel	Specifies whether to use InstanceTunnel to obtain execution results.	True
tunnel.limit_instance_tunnel	Specifies whether to limit the number of data records obtained by using InstanceTunnel.	None
tunnel.string_as_binary	Specifies whether to use bytes instead of Unicode for data of the STRING type.	False

DataFrame configurations

Option	Description	Default value
interactive	Specifies whether DataFrames are used in an interactive environment.	Depends on the detection value.
df.analyze	Specifies whether to enable functions that are not built in MaxCompute.	True
df.optimize	Specifies whether to enable full DataFrame optimization.	True

Option	Description	Default value
df.optimizes.pp	Specifies whether to enable DataFrame predicate pushdown optimization.	True
df.optimizes.cp	Specifies whether to enable DataFrame column pruning optimization.	True
df.optimizes.tunnel	Specifies whether to enable DataFrame tunnel optimization.	True
df.quote	Specifies whether to use a pair of grave accents (``) to mark field and table names in the backend of MaxCompute SQL.	True
df.libraries	The resource name of the third-party library that is used for DataFrame operations.	None
df.supersede_libraries	Specifies whether to use the self-uploaded NumPy to replace the version in the service.	False
df.odps.sort.limit	The default limit on the number of items that are added during a sort operation of DataFrames.	10000

Machine learning configurations

Option	Description	Default value
ml.xflow_settings	The XFlow execution configuration.	None
ml.xflow_project	The default XFlow project name.	algo_public
ml.use_model_transfer	Specifies whether to use ModelTransfer to obtain the Predictive Model Markup Language (PMML) files of models.	False
ml.model_volume	The name of the volume used by ModelTransfer.	pyodps_volume

4.1.6. DataFrame

4.1.6.1. Overview

PyODPS provides a pandas-like API, PyODPS DataFrame, which can make full use of the computing power of MaxCompute. You can also change the data source from MaxCompute tables to pandas DataFrame, so that the same code can be executed on pandas.

- Quick start: describes how to create and manage a DataFrame object and how to use DataFrame to process data.
- Create a DataFrame object: describes how to create a DataFrame project to reference a data source.
- Sequence: introduces sequence objects in DataFrame. SequenceExpr represents a column in a two-dimensional dataset. You are not allowed to manually create SequenceExpr objects. You can only retrieve one from a collection object.
- Collection: introduces collection objects in DataFrame. CollectionExpr supports various operations on two-dimensional datasets, such as column operations, data filtering, and data transformation.
- Execution: introduces the execution methods that you can call to perform operations in DataFrame.
- MapReduce API: describes how to use the MapReduce API in DataFrame.
- Column operations: describes the column operations supported by DataFrame.
- Aggregation: describes the aggregation operations supported by DataFrame. It also describes how to implement group aggregation and write aggregate functions.
- Sort, deduplicate, sample, and transform data: describes how to perform sorting, deduplication, sampling, and data transformation on DataFrame objects.
- Data merging: describes the data merge operations supported by DataFrame. These operations include the JOIN and UNION operations.
- Window functions: describes the window functions supported by DataFrame.
- Plotting: describes the plotting methods provided by DataFrame.
- Debugging: describes how to perform DataFrame debugging. DataFrame can optimize and display the entire execution. You can visualize the execution.

For more information about pandas, see Get started with pandas.

4.1.6.2. Quick start

This topic describes how to create and manage a DataFrame object. It also provides information about how to use DataFrame to process data.

Create and manage a DataFrame object

The MovieLens 100K dataset is used in this example. The following tables exist: `pyodps_ml_100k_movies` (movie-related data), `pyodps_ml_100k_users` (user-related data), and `pyodps_ml_100k_ratings` (rating-related data).

1. If no MaxCompute objects are available in the runtime environment, create a MaxCompute object.

```
from odps import ODPS
o = ODPS('**your-access-id**', '**your-secret-access-key**',
         project='**your-project**', endpoint='**your-end-point**')
```

2. Create a DataFrame object by specifying a table.

```
from odps.df import DataFrame
users = DataFrame(o.get_table('pyodps_ml_100k_users'))
```

3. Use the `dtypes` attribute to view the fields of the DataFrame object and the data types of the

fields.

```
users.dtypes
odps.Schema {
  user_id             int64
  age                 int64
  sex                 string
  occupation          string
  zip_code            string
}
```

4. Use the `head` method to have a quick preview on the first N data records.

```
users.head(10)
   user_id  age  sex     occupation  zip_code
0        1   24    M     technician     85711
1        2   53    F          other     94043
2        3   23    M         writer     32067
3        4   24    M     technician     43537
4        5   33    F          other     15213
5        6   42    M      executive     98101
6        7   57    M  administrator     91344
7        8   36    M  administrator     05201
8        9   29    M        student     01002
9       10   53    M         lawyer     90703
```

5. If you do not want to view all the fields, perform the following operations as needed:
   - Specify the fields that you want to query.

     ```
     users[['user_id', 'age']].head(5)
        user_id  age
     0        1   24
     1        2   53
     2        3   23
     3        4   24
     4        5   33
     ```

   - Exclude the fields that you do not want to query.

     ```
     users.exclude('zip_code', 'age').head(5)
        user_id  sex  occupation
     0        1    M  technician
     1        2    F       other
     2        3    M      writer
     3        4    M  technician
     4        5    F       other
     ```

   - Exclude some fields and add new fields based on computation. For example, add the `sex_bool` attribute and set it to True if `sex` is `M`. Otherwise, set it to False.

```
users.select(users.exclude('zip_code', 'sex'), sex_bool=users.sex == 'M').head(5)
   user_id  age  occupation  sex_bool
0        1   24  technician      True
1        2   53       other     False
2        3   23      writer      True
3        4   24  technician      True
4        5   33       other     False
```

6. Obtain the numbers of male and female users.

```
users.groupby(users.sex).agg(count=users.count())
   sex  count
0    F    273
1    M    670
```

7. Divide users by occupation, obtain the first 10 occupations that have the largest population, and sort the occupations in descending order of population.

```
df = users.groupby('occupation').agg(count=users['occupation'].count())
df.sort(df['count'], ascending=False)[:10]
      occupation  count
0        student    196
1          other    105
2       educator     95
3  administrator     79
4       engineer     67
5     programmer     66
6      librarian     51
7         writer     45
8      executive     32
9      scientist     31
```

Alternatively, use the `value_counts` method. Note that the number of records returned by this method is limited by the `options.df.odps.sort.limit` parameter. For more information, see Configurations.

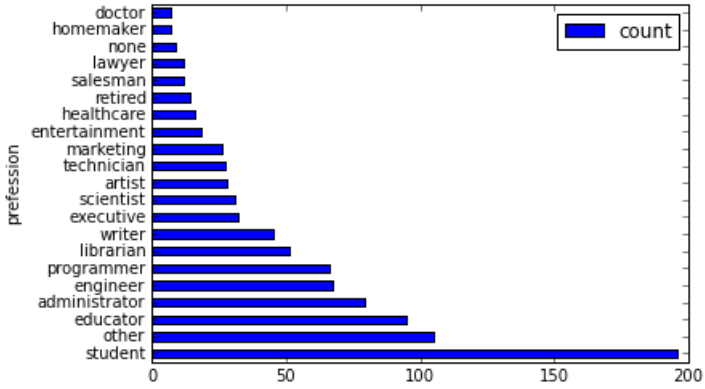
```
users.occupation.value_counts()[:10]
      occupation  count
0        student    196
1          other    105
2       educator     95
3  administrator     79
4       engineer     67
5     programmer     66
6      librarian     51
7         writer     45
8      executive     32
9      scientist     31
```

8. Show data in an intuitive graph.

```
%matplotlib inline
```

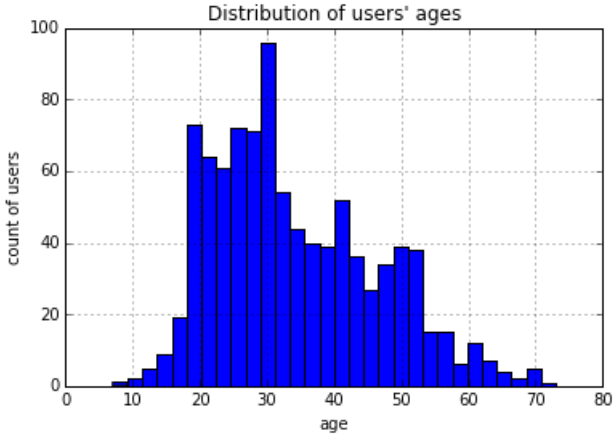
9. Use a horizontal column chart to visualize the data.

```
users['occupation'].value_counts().plot(kind='barh', x='occupation', ylabel='prefession
')
<matplotlib.axes._subplots.AxesSubplot at 0x10653cfd0>
```



10. Divide users into 30 groups by age and view the histogram of age distribution.

```
users.age.hist(bins=30, title="Distribution of users' ages", xlabel='age', ylabel='coun
t of users')
<matplotlib.axes._subplots.AxesSubplot at 0x10667a510>
```



11. Join the three tables and save them as a new table.

```
movies = DataFrame(o.get_table('pyodps_ml_100k_movies'))
ratings = DataFrame(o.get_table('pyodps_ml_100k_ratings'))
o.delete_table('pyodps_ml_100k_lens', if_exists=True)
lens = movies.join(ratings).join(users).persist('pyodps_ml_100k_lens')
lens.dtypes
odps.Schema {
  movie_id                              int64
  title                                 string
  release_date                          string
  video_release_date                    string
  imdb_url                              string
  user_id                               int64
  rating                                int64
  unix_timestamp                        int64
  age                                   int64
  sex                                   string
  occupation                            string
  zip_code                              string
}
```

12. Divide users aged 0 to 80 into eight age groups.

```
labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
 cut_lens = lens[lens, lens.age.cut(range(0, 81, 10), right=False, labels=labels).renam
e('age_group')]
```

13. View the first 10 ages each with only one user, as well as their age groups.

```
cut_lens['age_group', 'age'].distinct()[:10]
   age_group  age
0       0-9    7
1     10-19   10
2     10-19   11
3     10-19   13
4     10-19   14
5     10-19   15
6     10-19   16
7     10-19   17
8     10-19   18
9     10-19   19
```

14. View the total rating and average rating of users in each age group.

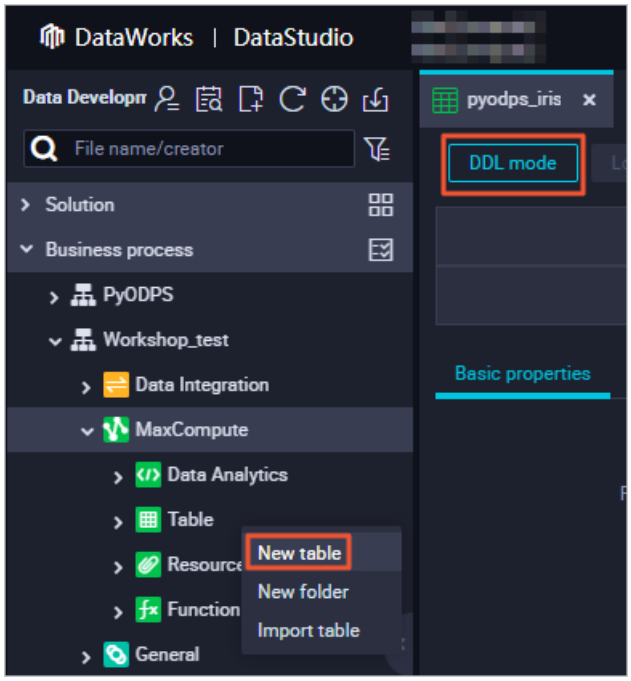
```
cut_lens.groupby('age_group').agg(cut_lens.rating.count().rename('total_rating'), cut_l
ens.rating.mean().rename('avg_rating'))
     age_group  avg_rating  total_rating
0       0-9  3.767442        43
1     10-19  3.486126      8181
2     20-29  3.467333     39535
3     30-39  3.554444     25696
4     40-49  3.591772     15021
5     50-59  3.635800      8704
6     60-69  3.648875      2623
7     70-79  3.649746       197
```

Use DataFrame to process data

Before you perform the following steps, download the Iris dataset and save the file to your computer as sample data. A DataWorks PyODPS node is used in this example. For more information, see Create a PyODPS 2 node.

1. Create a test data table.

   Use the table management feature of DataWorks to create a table. Then, click **DDL mode**.

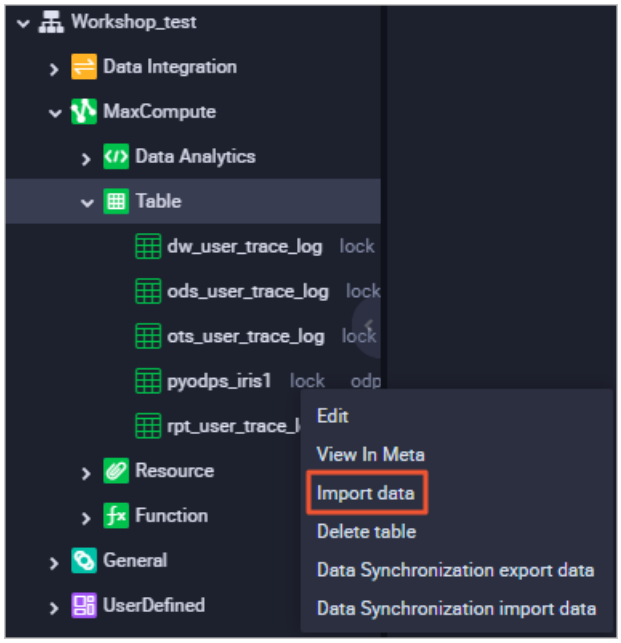
   

   Execute the following statement to generate a table structure:

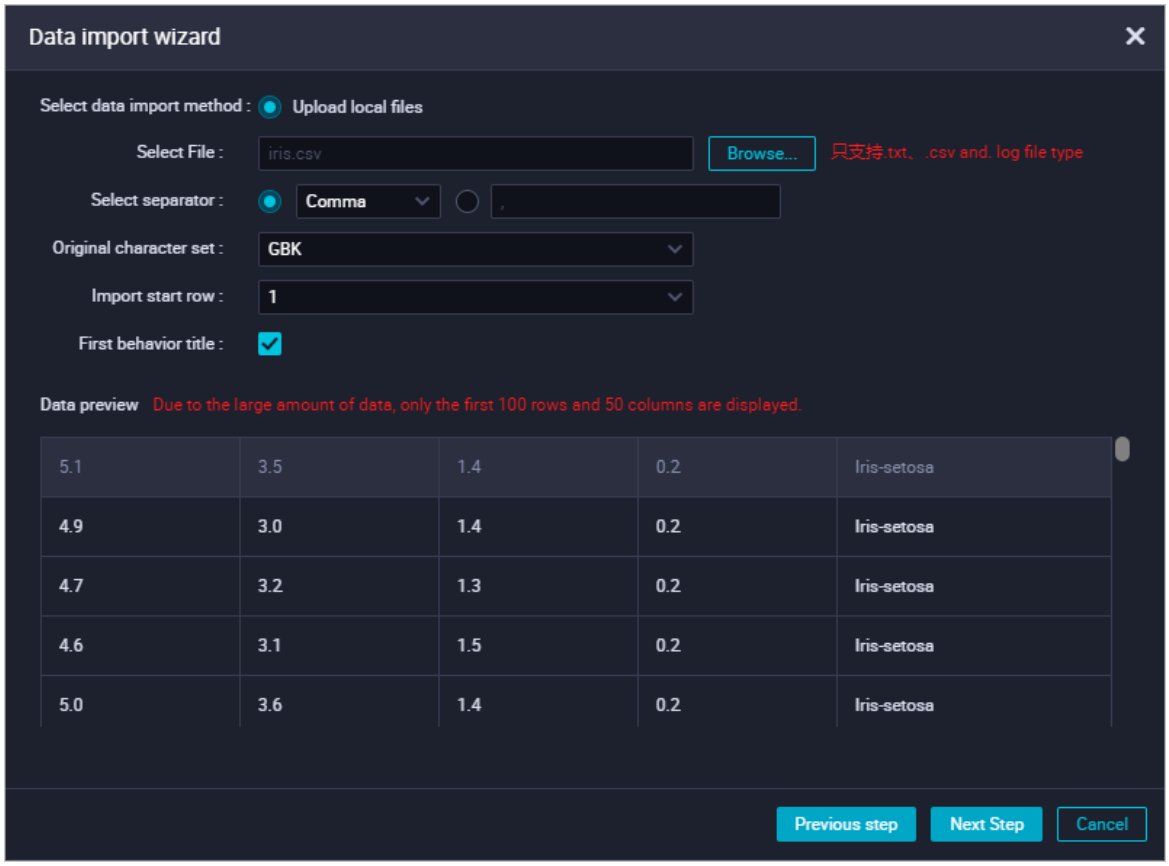
   ```
   CREATE TABLE pyodps_iris (
       sepallength double COMMENT 'sepallength(cm)',
       sepalwidth double COMMENT 'sepalwidth(cm)',
       petallength double COMMENT 'petallength(cm)',
       petalwidth double COMMENT 'petalwidth(cm)',
       name string COMMENT 'name'
   ) ;
   ```

2. Upload test data.

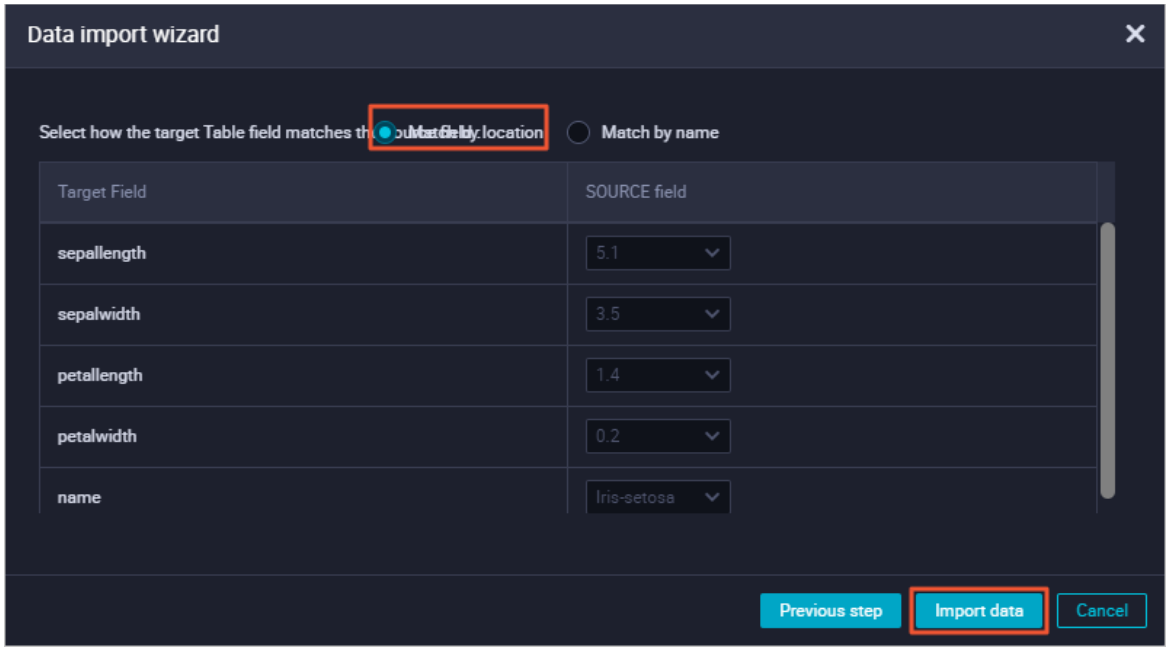
   Click **Import data**.



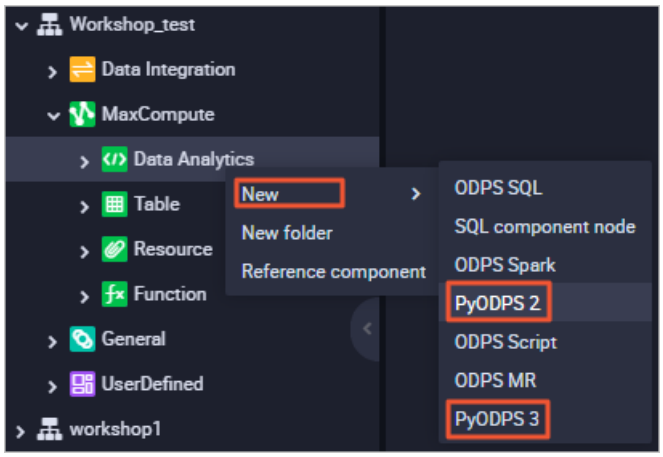
Enter the table name, upload the downloaded dataset, and then click Next.



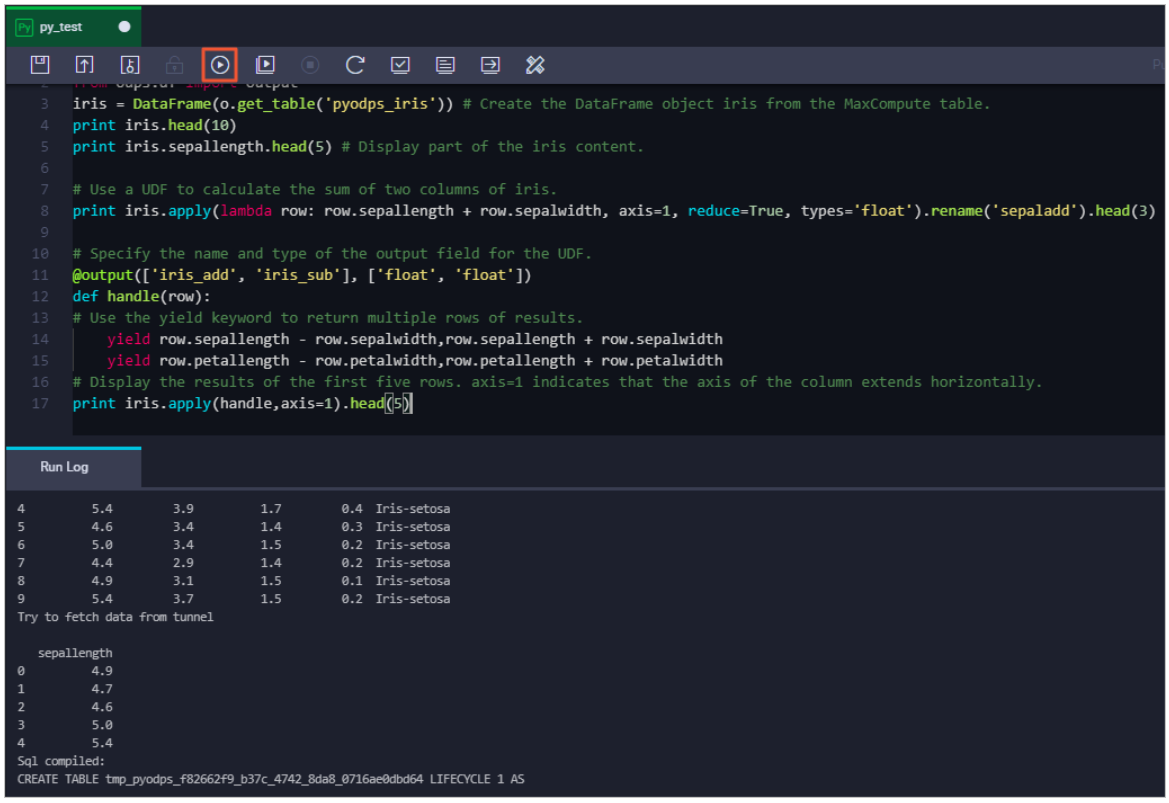
Select **Match by location** and click Import Data.



3. Create a PyODPS node to store and run code.



4. Enter the code and click the Run icon. You can view the result in the **Run Log** section in the lower pane.



Code details:

```
from odps.df import DataFrame
from odps.df import output
iris = DataFrame(o.get_table('pyodps_iris')) # Create the DataFrame object iris from th
e MaxCompute table.
print iris.head(10)
print iris.sepallength.head(5) # Display part of the iris content.
# Use a UDF to calculate the sum of two columns of iris.
print iris.apply(lambda row: row.sepallength + row.sepalwidth, axis=1, reduce=True, typ
es='float').rename('sepaladd').head(3)
# Specify the name and type of the output field for the UDF.
@output(['iris_add', 'iris_sub'], ['float', 'float'])
def handle(row):
# Use the yield keyword to return multiple rows of results.
    yield row.sepallength - row.sepalwidth,row.sepallength + row.sepalwidth
    yield row.petallength - row.petalwidth,row.petallength + row.petalwidth
# Display the results of the first five rows. axis=1 indicates that the axis of the col
umn extends horizontally.
print iris.apply(handle,axis=1).head(5)
```

4.1.6.3. Create a DataFrame object

This topic describes how to create a DataFrame object. After this object is created, it can be used to reference a data source.

Background information

To use a DataFrame project, you must familiarize yourself with the operations on the `Collection` (`DataFrame`), `Sequence` , and `Scalar` objects. Collection indicates a tabular data structure (two-dimensional structure). Sequence indicates a column (one-dimensional structure). Scalar indicates a scalar object.

If you use Pandas data to create a DataFrame object, the object contains the actual data. If you use a MaxCompute table to create a DataFrame object, the object does not contain the actual data. The object contains only data operations. After a DataFrame object is created, MaxCompute can store and compute data.

Procedure

DataFrame object is the only Collection object that you must create. A DataFrame object can be used to reference MaxCompute tables, MaxCompute partitions, Pandas DataFrame objects, and SQLAlchemy tables (database tables). The reference operations for these data sources are the same. You can process data without the need to modify code. You only need to change the input and output pointers. This way, you can migrate small amounts of test code that is running locally to MaxCompute. The accuracy of the migration is ensured by PyODPS DataFrame.

To create a DataFrame object, you only need to call the required MaxCompute table, Pandas DataFrame object, or SQLAlchemy table.

```
from odps.df import DataFrame
# Create a DataFrame object by using a MaxCompute table.
 iris = DataFrame(o.get_table('pyodps_iris'))
 iris2 = o.get_table('pyodps_iris').to_df()  # Use the to_df method for the table.
# Create a DataFrame object by using a MaxCompute partition.
 pt_df = DataFrame(o.get_table('partitioned_table').get_partition('pt=20171111'))
 pt_df2 = o.get_table('partitioned_table').get_partition('pt=20171111').to_df()  # Use the
to_df method for the partition.
# Create a DataFrame object by using a Pandas DataFrame object.
 import pandas as pd
 import numpy as np
 df = DataFrame(pd.DataFrame(np.arange(9).reshape(3, 3), columns=list('abc')))
# Create a DataFrame object by using an SQLAlchemy table.
 engine = sqlalchemy.create_engine('mysql://root:123456@localhost/movielens')
 metadata = sqlalchemy.MetaData(bind = engine) # Bind the metadata to a database engine.
 table = sqlalchemy.Table('top_users', metadata, extend_existing=True, autoload=True)
 users = DataFrame(table)
```

If you create a DataFrame object by using a Pandas DataFrame object, note the following points during the initialization of the DataFrame object:

- PyODPS DataFrame attempts to infer the NUMPY OBJECT or STRING data type. If a column is empty, an error is returned. To avoid these errors, set `unknown_as_string` to True, and convert the data type of this column to STRING.
- You can use the `as_type` parameter to forcibly convert the data type. If a basic data type is used, this type is forcibly converted when you create a PyODPS DataFrame object. If a Pandas DataFrame object contains a LIST or DICT column, the system does not infer the data type of this column. You must manually set `as_type` to DICT.

```
df2 = DataFrame(df, unknown_as_string=True, as_type={'null_col2': 'float'})
df2.dtypes
odps.Schema {
  sepallength           float64
  sepalwidth            float64
  petallength           float64
  petalwidth            float64
  name                  string
  null_col1             string   # The data type cannot be identified. You can set unknown_
as_string to True to convert the data type to STRING.
  null_col2             float64  # The data type is forcibly converted to FLOAT.
}
 df4 = DataFrame(df3, as_type={'list_col': 'list<int64>'})
 df4.dtypes
odps.Schema {
  id        int64
  list_col  list<int64>  # The data type cannot be identified or automatically converted. Y
ou must specify as_type.
}
```

Note PyODPS DataFrame does not allow you to upload Object Storage Service (OSS) or Tablestore (OTS) external tables to MaxCompute.

4.1.6.4. Sequence

SequenceExpr represents a column in a two-dimensional dataset. You cannot manually create a SequenceExpr object. Instead, you can retrieve one from a Collection object.

Column retrieval

You can use `collection.column_name` to retrieve a column.

```
iris.sepallength.head(5)
   sepallength
0          5.1
1          4.9
2          4.7
3          4.6
4          5.0
```

If the column name is stored in a string variable, you can use `df[column_name]` to retrieve a column.

```
iris['sepallength'].head(5)
   sepallength
0          5.1
1          4.9
2          4.7
3          4.6
4          5.0
```

Column types

DataFrame has its own type system. When a table is initialized, the MaxCompute data types are cast. This design provides support for more types of execution backends. The execution backend of DataFrame can be MaxCompute SQL, Pandas, or a MySQL or PostgreSQL database.

The following table lists the mappings between MaxCompute and DataFrame data types.

MaxCompute type	DataFrame type
BIGINT	INT64
DOUBLE	FLOAT64
STRING	STRING
DATETIME	DATETIME
BOOLEAN	BOOLEAN
DECIMAL	DECIMAL
ARRAY<VALUE_TYPE>	LIST<VALUE_TYPE>
MAP<KEY_TYPE, VALUE_TYPE>	DICT<KEY_TYPE, VALUE_TYPE>
When `options.sql.use_odps2_extension=True`, the following data types are also supported.	
TINYINT	INT8
SMALLINT	INT16
INT	INT32
FLOAT	FLOAT32

The following items describe DataFrame data types:

- PyODPS DataFrame supports the following data types: INT8, INT16, INT32, INT64, FLOAT32, FLOAT64, BOOLEAN, STRING, DECIMAL, DATETIME, LIST, and DICT.
- For LIST and DICT types, you must specify the types of the values they contain. Otherwise, an error occurs.
- DataFrame does not support the TIMESTAMP and STRUCT types introduced in MaxCompute 2.0. The new types will be supported in future releases.
- You can use `sequence.dtype` to retrieve the data type of a Sequence object.

```
iris.sepallength.dtype
float64
```

- You can use the `astype` method to change the type of a column. This method requires a type as input and returns the converted Sequence object as output.

```
iris.sepallength.astype('int')
   sepallength
0            5
1            4
2            4
3            4
4            5
```

Column name

In DataFrame computing, a Sequence object must have a column name. Typically, DataFrame automatically creates a name for each Sequence object.

```
iris.groupby('name').sepalwidth.max()
   sepalwidth_max
0             4.4
1             3.4
2             3.8
```

In the preceding example, `sepalwidth` is named `sepalwidth_max` after the max() method is applied to take the maximum value. In some operations, such as adding a Scalar object to a Sequence object, the resulting column is automatically assigned the name of the Sequence object. In other cases, you need to manually name a Sequence object.

You can use the `rename` method to rename a Sequence object.

```
iris.sepalwidth.rename('sepal_width').head(5)
   sepal_width
0          3.5
1          3.0
2          3.2
3          3.1
4          3.6
```

Simple column transformations

You can perform operations on a Sequence object to obtain a new Sequence object. This is similar to performing operations on a simple Python variable.

```
(iris.sepallength + 5).head(5)
   sepallength
0         10.1
1          9.9
2          9.7
3          9.6
4         10.0
```

Note that when two columns are involved in operations, you need to manually name the resulting column. For more information about column transformations, see Column operations.

```
(iris.sepallength + iris.sepalwidth).rename('sum_sepal').head(5)
   sum_sepal
0        8.6
1        7.9
2        7.9
3        7.7
4        8.6
```

4.1.6.5. Collection

CollectionExpr supports all operations on DataFrame two-dimensional datasets. It can be seen as a MaxCompute table or a spreadsheet. DataFrame objects are also CollectionExpr objects. CollectionExpr supports various operations on two-dimensional datasets such as column operations, filter operations, and transformation operations.

Retrieve types

You can use the `dtypes` method to retrieve the types of all columns in a CollectionExpr object. The `dtypes` method returns a Schema type.

```
iris.dtypes
odps.Schema {
  sepallength           float64
  sepalwidth            float64
  petallength           float64
  petalwidth            float64
  name                  string
}
```

Select, add, and remove columns

- Select columns

  You can use the `expr[columns]` syntax to select certain columns from a CollectionExpr object to form a new dataset.

  ```
  iris['name', 'sepallength'].head(5)
           name  sepallength
  0  Iris-setosa          5.1
  1  Iris-setosa          4.9
  2  Iris-setosa          4.7
  3  Iris-setosa          4.6
  4  Iris-setosa          5.0
  ```

  > **Note**
  >
  > If only one column is needed, you need to add a comma (,) after the column name or explicitly mark the column as a list, such as `iris[iris.sepal_length,]` or `iris[[iris.sepal_length]]`. Otherwise, a Sequence object, instead of a Collection object, is returned.

- Delete columns

  You can use the `exclude` method to exclude certain columns of the original dataset from the new dataset.

```
iris.exclude('sepallength', 'petallength')[:5]
   sepalwidth  petalwidth         name
0         3.5         0.2  Iris-setosa
1         3.0         0.2  Iris-setosa
2         3.2         0.2  Iris-setosa
3         3.1         0.2  Iris-setosa
4         3.6         0.2  Iris-setosa
```

  In PyODPS version 0.7.2 and later, you can use a new method to directly exclude certain columns from the dataset.

```
del iris['sepallength']
del iris['petallength']
iris[:5]
   sepalwidth  petalwidth         name
0         3.5         0.2  Iris-setosa
1         3.0         0.2  Iris-setosa
2         3.2         0.2  Iris-setosa
3         3.1         0.2  Iris-setosa
4         3.6         0.2  Iris-setosa
```

- Add columns

  You can use the `expr[expr, new_sequence]` syntax to add a transformed column to an existing Collection object. The new column is part of the new Collection.

  In the following example, a new column is created by adding one to each element in the `sepalwidth` column of `iris`, renamed to `sepalwidthplus1`, and appended to the dataset to form a new dataset.

```
iris[iris, (iris.sepalwidth + 1).rename('sepalwidthplus1')].head(5)
   sepallength  sepalwidth  petallength  petalwidth         name  \
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
   sepalwidthplus1
0              4.5
1              4.0
2              4.2
3              4.1
4              4.6
```

  When you use the `expr[expr, new_sequence]` syntax, note that the transformed column may have the same name as the original column. Rename the new column if you want to merge it with the original Collection.

  In PyODPS version 0.7.2 and later, you can directly append a column to the current dataset.

```
iris['sepalwidthplus1'] = iris.sepalwidth + 1
iris.head(5)
   sepallength  sepalwidth  petallength  petalwidth         name  \
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
   sepalwidthplus1
0              4.5
1              4.0
2              4.2
3              4.1
4              4.6
```

- Add and delete columns simultaneously

  You can also use the `exclude` method to exclude the original column and append the transformed column to the dataset, without the need to rename the new column.

```
iris[iris.exclude('sepalwidth'), iris.sepalwidth * 2].head(5)
   sepallength  petallength  petalwidth         name  sepalwidth
0          5.1          1.4         0.2  Iris-setosa         7.0
1          4.9          1.4         0.2  Iris-setosa         6.0
2          4.7          1.3         0.2  Iris-setosa         6.4
3          4.6          1.5         0.2  Iris-setosa         6.2
4          5.0          1.4         0.2  Iris-setosa         7.2
```

  In PyODPS version 0.7.2 and later, you can directly overwrite an existing column on the current dataset. The following example illustrates the syntax.

```
iris['sepalwidth'] = iris.sepalwidth * 2
iris.head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1         7.0          1.4         0.2  Iris-setosa
1          4.9         6.0          1.4         0.2  Iris-setosa
2          4.7         6.4          1.3         0.2  Iris-setosa
3          4.6         6.2          1.5         0.2  Iris-setosa
4          5.0         7.2          1.4         0.2  Iris-setosa
```

  The `select` method provides another way to create a new Collection. When you use this method, you need to pass in the selected columns as input parameters. You can use the `keyword` argument to rename a column to the given keyword.

```
iris.select('name', sepalwidthminus1=iris.sepalwidth - 1).head(5)
          name  sepalwidthminus1
0  Iris-setosa               2.5
1  Iris-setosa               2.0
2  Iris-setosa               2.2
3  Iris-setosa               2.1
4  Iris-setosa               2.6
```

You can also pass in a lambda expression, which takes the result from the previous operation as a parameter. During the execution, PyODPS checks the lambda expression and passes in the Collection object generated from the previous operation and replaces it with the correct columns.

```
iris['name', 'petallength'][[lambda x: x.name]].head(5)
          name
0  Iris-setosa
1  Iris-setosa
2  Iris-setosa
3  Iris-setosa
4  Iris-setosa
```

In PyODPS version 0.7.2 and later, conditional assignments are supported.

```
iris[iris.sepallength > 5.0, 'sepalwidth'] = iris.sepalwidth * 2
iris.head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1        14.0          1.4         0.2  Iris-setosa
1          4.9         6.0          1.4         0.2  Iris-setosa
2          4.7         6.4          1.3         0.2  Iris-setosa
3          4.6         6.2          1.5         0.2  Iris-setosa
4          5.0         7.2          1.4         0.2  Iris-setosa
```

Introduce constants and random numbers

- DataFrame allows you to append a column of constants to a Collection object. Scalar is required for this operation, and you need to manually specify the column name.

```
from odps.df import Scalar
iris[iris, Scalar(1).rename('id')][:5]
   sepallength  sepalwidth  petallength  petalwidth         name  id
0          5.1         3.5          1.4         0.2  Iris-setosa   1
1          4.9         3.0          1.4         0.2  Iris-setosa   1
2          4.7         3.2          1.3         0.2  Iris-setosa   1
3          4.6         3.1          1.5         0.2  Iris-setosa   1
4          5.0         3.6          1.4         0.2  Iris-setosa   1
```

You can use NullScalar to specify a null column. In this case, you need to specify the field type.

```
from odps.df import NullScalar
iris[iris, NullScalar('float').rename('fid')][:5]
   sepal_length  sepal_width  petal_length  petal_width     category   fid
0           5.1          3.5           1.4          0.2  Iris-setosa  None
1           4.9          3.0           1.4          0.2  Iris-setosa  None
2           4.7          3.2           1.3          0.2  Iris-setosa  None
3           4.6          3.1           1.5          0.2  Iris-setosa  None
4           5.0          3.6           1.4          0.2  Iris-setosa  None
```

In PyODPS version 0.7.12 and later, a simpler method is provided.

```
iris['id'] = 1
iris
   sepallength  sepalwidth  petallength  petalwidth         name  id
0          5.1         3.5          1.4         0.2  Iris-setosa   1
1          4.9         3.0          1.4         0.2  Iris-setosa   1
2          4.7         3.2          1.3         0.2  Iris-setosa   1
3          4.6         3.1          1.5         0.2  Iris-setosa   1
4          5.0         3.6          1.4         0.2  Iris-setosa   1
```

Note that this method cannot automatically recognize the type of null values. Therefore, we recommend that you use the following syntax to add null columns.

```
iris['null_col'] = NullScalar('float')
iris
   sepallength  sepalwidth  petallength  petalwidth         name  null_col
0          5.1         3.5          1.4         0.2  Iris-setosa      None
1          4.9         3.0          1.4         0.2  Iris-setosa      None
2          4.7         3.2          1.3         0.2  Iris-setosa      None
3          4.6         3.1          1.5         0.2  Iris-setosa      None
4          5.0         3.6          1.4         0.2  Iris-setosa      None
```

- DataFrame also allows you to append a column of random numbers to a Collection object. The column type is FLOAT and the value range is 0-1. Each number has a different value. RandomScalar is required for this operation, and its parameter is an optional random seed.

```
from odps.df import RandomScalar
iris[iris, RandomScalar().rename('rand_val')][:5]
   sepallength  sepalwidth  petallength  petalwidth         name  rand_val
0          5.1         3.5          1.4         0.2  Iris-setosa  0.000471
1          4.9         3.0          1.4         0.2  Iris-setosa  0.799520
2          4.7         3.2          1.3         0.2  Iris-setosa  0.834609
3          4.6         3.1          1.5         0.2  Iris-setosa  0.106921
4          5.0         3.6          1.4         0.2  Iris-setosa  0.763442
```

Filter data

A Collection object allows you to filter data. In the following example, the records where `sepallength` is greater than 5 are retrieved.

```
iris[iris.sepallength > 5].head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          5.4         3.9          1.7         0.4  Iris-setosa
2          5.4         3.7          1.5         0.2  Iris-setosa
3          5.8         4.0          1.2         0.2  Iris-setosa
4          5.7         4.4          1.5         0.4  Iris-setosa
```

The following example shows how to query records by using multiple filter conditions:

- AND (&) operator

```
iris[(iris.sepallength < 5) & (iris['petallength'] > 1.5)].head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          4.8         3.4          1.6         0.2     Iris-setosa
1          4.8         3.4          1.9         0.2     Iris-setosa
2          4.7         3.2          1.6         0.2     Iris-setosa
3          4.8         3.1          1.6         0.2     Iris-setosa
4          4.9         2.4          3.3         1.0  Iris-versicolor
```

- OR (|) operator

```
iris[(iris.sepalwidth < 2.5) | (iris.sepalwidth > 4)].head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.7         4.4          1.5         0.4     Iris-setosa
1          5.2         4.1          1.5         0.1     Iris-setosa
2          5.5         4.2          1.4         0.2     Iris-setosa
3          4.5         2.3          1.3         0.3     Iris-setosa
4          5.5         2.3          4.0         1.3  Iris-versicolor
```

> **Note** You must use an ampersand `&` to reprent the AND operator and use a vertical bar `|` to represent the OR operator. `and` and `or` are not allowed.

- NOT (~) operator

```
iris[~(iris.sepalwidth > 3)].head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          4.9         3.0          1.4         0.2  Iris-setosa
1          4.4         2.9          1.4         0.2  Iris-setosa
2          4.8         3.0          1.4         0.1  Iris-setosa
3          4.3         3.0          1.1         0.1  Iris-setosa
4          5.0         3.0          1.6         0.2  Iris-setosa
```

- You can explicitly call the `filter` method to specify multiple conditions by using AND (&) operators.

```
iris.filter(iris.sepalwidth > 3.5, iris.sepalwidth < 4).head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.0         3.6          1.4         0.2  Iris-setosa
1          5.4         3.9          1.7         0.4  Iris-setosa
2          5.4         3.7          1.5         0.2  Iris-setosa
3          5.4         3.9          1.3         0.4  Iris-setosa
4          5.7         3.8          1.7         0.3  Iris-setosa
```

- You can use a lambda expression to perform sequential operations.

```
iris[iris.sepalwidth > 3.8]['name', lambda x: x.sepallength + 1]
          name  sepallength
0  Iris-setosa          6.4
1  Iris-setosa          6.8
2  Iris-setosa          6.7
3  Iris-setosa          6.4
4  Iris-setosa          6.2
5  Iris-setosa          6.5
```

- If a Collection object contains a BOOLEAN column, you can use the column as a filter condition.

```
df.dtypes
odps.Schema {
  a boolean
  b int64
}
df[df.a]
      a  b
0  True  1
1  True  3
```

Therefore, when you retrieve a single Sequence from a Collection object, only the BOOLEAN column can be used as a valid filter condition.

```
df[df.a, ] # retrieve a one-column collection.
df[[df.a]] # retrieve a one-column collection.
df.select(df.a) # retrieve a one-column collection explicitly.
df[df.a] # column 'a' is a boolean column. Filter data on the boolean column.
df.a # retrieve a column from a collection.
df['a'] # retrieve a column from a collection.
```

- You can also use the `query` method in Pandas to filter data, and use column names such as `sepallength` in an expression. In query statements, both the ampersand `&` and `and` represent the AND operator, and both the vertical bar `|` and `or` represent the OR operator.

```
iris.query("(sepallength < 5) and (petallength > 1.5)").head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          4.8         3.4          1.6         0.2     Iris-setosa
1          4.8         3.4          1.9         0.2     Iris-setosa
2          4.7         3.2          1.6         0.2     Iris-setosa
3          4.8         3.1          1.6         0.2     Iris-setosa
4          4.9         2.4          3.3         1.0  Iris-versicolor
```

When a local variable is required in an expression, add an at sign `@` before the variable name.

```
var = 4
iris.query("(iris.sepalwidth < 2.5) | (sepalwidth > @var)").head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.7         4.4          1.5         0.4     Iris-setosa
1          5.2         4.1          1.5         0.1     Iris-setosa
2          5.5         4.2          1.4         0.2     Iris-setosa
3          4.5         2.3          1.3         0.3     Iris-setosa
4          5.5         2.3          4.0         1.3  Iris-versicolor
```

The `query` method supports the following syntax.

Syntax	Description
name	Names without the at sign `@` are processed as column names. Names with the at sign (@) are processed as local variables.

Syntax	Description
operator	The following operators are supported: `+`, `-`, `*`, `/`, `//`, `%`, `**`, `==`, `!=`, `<`, `<=`, `>`, `>=`, `in`, `not in`.
bool	An ampersand `&` and `and` represent the AND operator. A vertical bar `|` and `or` represent the OR operator.
attribute	This syntax is used to retrieve the attribute of the object.
index, slice, subscript	Slice operations.

Lateral view

You can use the `explode` method to convert the LIST and MAP columns into multiple rows for output. You can also use the `apply` method to generate a lateral view. You often need to merge the output with the columns in the original table for operations such as aggregation. In this case, you can use the lateral view function of DataFrame. That is, map the collection generated by the lateral view function with the column names in the original collection.

The following example shows how to generate a lateral view.

```
df
   id         a             b
0   1  [a1, b1]  [a2, b2, c2]
1   2      [c1]      [d2, e2]
df[df.id, df.a.explode(), df.b]
   id   a             b
0   1  a1  [a2, b2, c2]
1   1  b1  [a2, b2, c2]
2   2  c1      [d2, e2]
df[df.id, df.a.explode(), df.b.explode()]
   id   a   b
0   1  a1  a2
1   1  a1  b2
2   1  a1  c2
3   1  b1  a2
4   1  b1  b2
5   1  b1  c2
6   2  c1  d2
7   2  c1  e2
```

If the lateral view method does not produce output for a row, the row does not appear in the output by default. To display the row in the output, you can set keep_nulls to True. In this case, the column value for the row is None in the output.

```
df
   id         a
0   1  [a1, b1]
1   2        []
df[df.id, df.a.explode()]
   id   a
0   1  a1
1   1  b1
df[df.id, df.a.explode(keep_nulls=True)]
   id     a
0   1    a1
1   1    b1
2   2  None
```

For more information about how to generate a lateral view by using the `explode` method, see Collection-related operations.

Limit output records

Output the first three records only.

```
iris[:3]
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
```

For MaxCompute SQL backend, `start` and `step` methods are not supported in slice operations. But you can use the `limit` method.

```
iris.limit(3)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
```

Note You can perform slice operations on Collection objects only, not Sequence objects.

4.1.6.6. Execution

This topic describes the execution methods that are supported in DataFrame API operations.

Deferred execution

DataFrame API operations are not automatically executed. These operations are executed only when you explicitly call the `execute` method or when you call the methods that internally call `execute`. The following table lists the methods that internally call the execute method.

Method	Description	Return value
persist	Saves the execution results to MaxCompute tables.	PyODPS DataFrame
execute	Executes the operations and returns all the results.	ResultFrame
head	Executes the operations and returns the first N rows of result data.	ResultFrame
tail	Executes the operations and returns the last N rows of result data.	ResultFrame
to_pandas	Converts a Collection object to a pandas DataFrame object or a Sequence object to a Series object. If you set the wrap parameter to True, an Alibaba Cloud MaxCompute SDK for python (PyODPS) DataFrame object is returned.	• If the wrap parameter is set to True, a PyODPS DataFrame object is returned. • If the wrap parameter is set to False, a pandas DataFrame object is returned. The default value of the wrap parameter is False.
plot, hist, and boxplot	Plotting methods.	N/A

Note In an interactive environment, PyODPS DataFrame automatically calls the `execute` method when it displays result data or when it calls the `repr` method. You do not need to manually call the `execute` method.

Example

```
>>> iris[iris.sepallength < 5][:5]
   sepallength  sepalwidth  petallength  petalwidth         name
0          4.9         3.0          1.4         0.2  Iris-setosa
1          4.7         3.2          1.3         0.2  Iris-setosa
2          4.6         3.1          1.5         0.2  Iris-setosa
3          4.6         3.4          1.4         0.3  Iris-setosa
4          4.4         2.9          1.4         0.2  Iris-setosa
```

To disable automatic execution, run the following code:

```
>>> from odps import options
>>> options.interactive = False
>>>
>>> iris[iris.sepallength < 5][:5]
Collection: ref_0
  odps.Table
    name: odps_test_sqltask_finance.`pyodps_iris`
    schema:
      sepallength           : double
      sepalwidth            : double
      petallength           : double
      petalwidth            : double
      name                  : string
Collection: ref_1
  Filter[collection]
    collection: ref_0
    predicate:
      Less[sequence(boolean)]
        sepallength = Column[sequence(float64)] 'sepallength' from collection ref_0
        Scalar[int8]
          5
Slice[collection]
  collection: ref_1
  stop:
    Scalar[int8]
      5
```

If you call `repr` to return a printable representation of the object, the entire abstract syntax tree is displayed.

> **Note** The ResultFrame is a result set and cannot be used in subsequent calculations.

To retrieve all records from the ResultFrame, use an iterative method.

```
>>> result = iris.head(3)
>>> for r in result:
>>>     print(list(r))
[5.0999999999999996, 3.5, 1.3999999999999999, 0.20000000000000001, u'Iris-setosa']
[4.9000000000000004, 3.0, 1.3999999999999999, 0.20000000000000001, u'Iris-setosa']
[4.7000000000000002, 3.2000000000000002, 1.3, 0.20000000000000001, u'Iris-setosa']
```

If pandas is installed, a ResultFrame can be converted into a pandas DataFrame or a PyODPS DataFrame that uses the pandas backend.

```
pd_df = iris.head(3).to_pandas()  # Return a pandas DataFrame.
 wrapped_df = iris.head(3).to_pandas(wrap=True)  # Return a PyODPS DataFrame that uses the
pandas backend.
```

Save results to MaxCompute tables

You can call the `persist` method to return a new DataFrame object for a Collection object. The persist method uses the table name as the parameter.

```
>>> iris2 = iris[iris.sepalwidth < 2.5].persist('pyodps_iris2')
>>> iris2.head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          4.5         2.3          1.3         0.3     Iris-setosa
1          5.5         2.3          4.0         1.3 Iris-versicolor
2          4.9         2.4          3.3         1.0 Iris-versicolor
3          5.0         2.0          3.5         1.0 Iris-versicolor
4          6.0         2.2          4.0         1.0 Iris-versicolor
```

You can pass the `partitions` parameter to the `persist` method to create a partitioned table. The table is partitioned based on the columns that are specified by `partitions`.

```
iris3 = iris[iris.sepalwidth < 2.5].persist('pyodps_iris3', partitions=['name'])
 iris3.data
odps.Table
  name: odps_test_sqltask_finance.`pyodps_iris3`
  schema:
    sepallength           : double
    sepalwidth            : double
    petallength           : double
    petalwidth            : double
  partitions:
    name                  : string
```

To write data to a partition of an existing table, you can pass the `partition` parameter to the `persist` method. The partition parameter specifies the destination partition. For example, set the partition parameter to `ds=******`. The table must contain all columns of the DataFrame object and the columns must be of the same type. The `drop_partition` and `create_partition` parameters are valid only if the partition parameter is specified. The drop_partition parameter specifies whether to delete the specified partition if the partition exists. The create_partition parameter specifies whether to create the specified partition if the partition does not exist.

```
iris[iris.sepalwidth < 2.5].persist('pyodps_iris4', partition='ds=test', drop_partition=Tru
e, create_partition=True)
```

When you write data to a table, you can specify the time-to-live (TTL) of the table. For example, the following statement sets the TTL of the table to 10 days.

```
iris[iris.sepalwidth < 2.5].persist('pyodps_iris5', lifecycle=10)
```

If the data source does not contain MaxCompute objects, you must manually specify the MaxCompute object or mark the entrance object as global when you call the `persist` method.

```
# Assume that the entrance object of PyODPS is o.
# Specify the entrance object.
df.persist('table_name', odps=o)
# Alternatively, mark the entrance object as global.
o.to_global()
df.persist('table_name')
```

Save results to pandas DataFrame

You can call the `to_pandas` method to save results to pandas DataFrame objects. If the `wrap` parameter is set to True, a PyODPS DataFrame object is returned.

```
type(iris[iris.sepalwidth < 2.5].to_pandas())
pandas.core.frame.DataFrame
type(iris[iris.sepalwidth < 2.5].to_pandas(wrap=True))
odps.df.core.DataFrame
```

After data is read from Tables, PyODPS can execute the `open_reader` method and use `reader.to_pandas()` to convert Collection objects to pandas DataFrame objects.

Set runtime parameters

You can set the runtime parameters for methods such as `execute`, `persist`, and `to_pandas`. This setting is valid only for the MaxCompute SQL backend.

- Set global parameters. For more information, see SQL.
- You can specify the `hints` parameter in these methods. This ensures that the specified runtime parameters are valid only for the current calculation.

```
iris[iris.sepallength < 5].to_pandas(hints={'odps.sql.mapper.split.size': 16})
```

Display details at runtime

To view the logview of an instance at runtime, you must modify the global configurations.

```
from odps import options
 options.verbose = True
 iris[iris.sepallength < 5].exclude('sepallength')[:5].execute()
Sql compiled:
SELECT t1.`sepalwidth`, t1.`petallength`, t1.`petalwidth`, t1.`name`
FROM odps_test_sqltask_finance.`pyodps_iris` t1
WHERE t1.`sepallength` < 5
LIMIT 5
logview:
http://logview
   sepalwidth  petallength  petalwidth         name
0         3.0          1.4         0.2  Iris-setosa
1         3.2          1.3         0.2  Iris-setosa
2         3.1          1.5         0.2  Iris-setosa
3         3.4          1.4         0.3  Iris-setosa
4         2.9          1.4         0.2  Iris-setosa
```

You can specify a logging function.

```
my_logs = []
 def my_logger(x):
     my_logs.append(x)
 options.verbose_log = my_logger
 iris[iris.sepallength < 5].exclude('sepallength')[:5].execute()
   sepalwidth  petallength  petalwidth         name
0         3.0          1.4         0.2  Iris-setosa
1         3.2          1.3         0.2  Iris-setosa
2         3.1          1.5         0.2  Iris-setosa
3         3.4          1.4         0.3  Iris-setosa
4         2.9          1.4         0.2  Iris-setosa
 print(my_logs)
['Sql compiled:', 'SELECT t1.`sepalwidth`, t1.`petallength`, t1.`petalwidth`, t1.`name` \nF
ROM odps_test_sqltask_finance.`pyodps_iris` t1 \nWHERE t1.`sepallength` < 5 \nLIMIT 5', 'lo
gview:', u'http://logview']
```

Cache intermediate results

When PyODPS DataFrame calculates data, some Collection objects are used multiple times. To view the execution results of an intermediate process, you can call the `cache` method to mark a Collection object that you want to calculate first.

> **Note** The execution of the `cache` method is deferred. Automatic calculation is not immediately triggered after the `cache` method is called.

```
cached = iris[iris.sepalwidth < 3.5].cache()
 df = cached['sepallength', 'name'].head(3)
 df
   sepallength         name
0          4.9  Iris-setosa
1          4.7  Iris-setosa
2          4.6  Iris-setosa
 cached.head(3)  # The result can be immediately retrieved because the cached object is cal
culated.
   sepallength         name
0          4.9  Iris-setosa
1          4.7  Iris-setosa
2          4.6  Iris-setosa
```

Asynchronous and parallel execution

PyODPS DataFrame supports asynchronous operations. For the `execute`, `persist`, `head`, `tail`, and `to_pandas` methods, you can pass the `async` parameter to enable asynchronous execution. The `timeout` parameter specifies the time-out period. Asynchronous operations return Future objects.

```
from odps.df import Delay
 delay = Delay()  # Create the Delay object.
 df = iris[iris.sepal_width < 5].cache()  # Common dependency of subsequent expressions.
 future1 = df.sepal_width.sum().execute(delay=delay)  # Return the Future object. The execu
tion is not started.
 future2 = df.sepal_width.mean().execute(delay=delay)
 future3 = df.sepal_length.max().execute(delay=delay)
 delay.execute(n_parallel=3)  # The execution starts with three concurrent threads.
|========================================|   1 /  1  (100.00%)        21s
 future1.result()
 458.10000000000014
 future2.result()
3.0540000000000007
```

In the preceding example, PyODPS DataFrame first executes the object of the shared dependency. Then, PyODPS DataFrame sets the degree of parallelism to 3 and executes objects from `future1` to `future3`. If `n_parallel` is set to 1, the execution time reaches 37 seconds.

You can pass the `async` parameter to `delay.execute` to specify whether to enable asynchronous execution. If asynchronous execution is enabled, you can also use the `timeout` parameter to specify the time-out period.

4.1.6.7. Column operations

This topic describes how to call the DataFrame API to perform column operations.

Column operations

```
from odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
lens = DataFrame(o.get_table('pyodps_ml_100k_lens'))
```

If you add a constant or execute a sin function to a sequence object, you add the constant or execute the sin function to all the elements of the sequence object.

Null-related functions

The DataFrame API provides null-related built-in functions, including isnull, notnull, and fillna. You can use the `isnull` function to check whether the value of a field is null. You can use the `notnull` function to check whether the value of a field is not null. You can use the `fillna` function to replace null values with specified values.

```
>>> iris.sepallength.isnull().head(5)
   sepallength
0        False
1        False
2        False
3        False
4        False
```

Logic functions

The DataFrame API provides the logic functions ifelse and switch. You can use the `ifelse` function to check whether a BOOLEAN field is true. If the field is true, the parameter numbered as 0 is returned. If the field is false, the parameter numbered as 1 is returned.

```
>>> (iris.sepallength > 5).ifelse('gt5', 'lte5').rename('cmp5').head(5)
   cmp5
0   gt5
1  lte5
2  lte5
3  lte5
4  lte5
```

You can use the `switch` function to handle multiple conditions.

```
>>> iris.sepallength.switch(4.9, 'eq4.9', 5.0, 'eq5.0', default='noeq').rename('equalness')
.head(5)
   equalness
0       noeq
1      eq4.9
2       noeq
3       noeq
4      eq5.0
```

```
>>> from odps.df import switch
>>> switch(iris.sepallength == 4.9, 'eq4.9', iris.sepallength == 5.0, 'eq5.0', default='noe
q').rename('equalness').head(5)
   equalness
0       noeq
1      eq4.9
2       noeq
3       noeq
4      eq5.0
```

In Python on MaxCompute (PyODPS) V0.7.8 and later, you can modify the values of a column in a dataset based on the specified conditions.

```
>>> iris[iris.sepallength > 5, 'cmp5'] = 'gt5'
>>> iris[iris.sepallength <= 5, 'cmp5'] = 'lte5'
>>> iris.head(5)
   cmp5
0   gt5
1  lte5
2  lte5
3  lte5
4  lte5
```

Mathematical operations

In numerical fields, a sequence object supports mathematical operations such as addition (+), subtraction (-), multiplication (*), and division (/). The log and sin functions are also supported.

```
>>> (iris.sepallength * 10).log().head(5)
   sepallength
0     3.931826
1     3.891820
2     3.850148
3     3.828641
4     3.912023
```

```
>>> fields = [iris.sepallength,
>>>           (iris.sepallength / 2).rename('sepallength divided by 2'),
>>>           (iris.sepallength ** 2).rename('sepallength squared')]
>>> iris[fields].head(5)
   sepallength  sepallength divided by 2  sepallength squared
0          5.1                      2.55                26.01
1          4.9                      2.45                24.01
2          4.7                      2.35                22.09
3          4.6                      2.30                21.16
4          5.0                      2.50                25.00
```

The following table describes the supported arithmetic operations.

Arithmetic operation	Description
abs	Returns the absolute value of the given number.
sqrt	Returns the square root of the given number.
sin	N/A
sinh	N/A
cos	N/A
cosh	N/A
tan	N/A
tanh	N/A
arccos	N/A
arccosh	N/A
arcsin	N/A
arcsinh	N/A
arctan	N/A
arctanh	N/A
exp	Returns e raised to the power of the given number.

Arithmetic operation	Description
expm1	Returns e raised to the power of the given number, minus 1.
log	Returns the logarithm of the given number by using a supplied base.
log2	N/A
log10	N/A
log1p	log(1+x)
radians	Converts the values in radians to degrees.
degrees	Converts the values in degrees to radians.
ceil	Returns the smallest integer that is no less than the given number.
floor	Returns the largest integer that is no greater than the given number.
trunc	Returns a number truncated to the specified decimal place.

You can compare a sequence object with another sequence or scalar objects.

```
>>> (iris.sepallength < 5).head(5)
   sepallength
0        False
1         True
2         True
3         True
4        False
```

The DataFrame API does not support sequential comparison, such as `3 <= iris.sepallength <= 5`. However, you can use the `between` function to check whether the value of `iris.sepallength` is within a specified interval.

```
>>> (iris.sepallength.between(3, 5)).head(5)
   sepallength
0        False
1         True
2         True
3         True
4         True
```

By default, the `between` function specifies an interval that includes endpoints. To specify an open interval, set the `inclusive` parameter to False.

```
>>> (iris.sepallength.between(3, 5, inclusive=False)).head(5)
   sepallength
0        False
1         True
2         True
3         True
4        False
```

String-related operations

The DataFrame API provides a number of string-related operations for sequence or scalar objects.

```
>>> fields = [
>>>     iris.name.upper().rename('upper_name'),
>>>     iris.name.extract('Iris(.*)', group=1)
>>> ]
>>> iris[fields].head(5)
    upper_name     name
0  IRIS-SETOSA  -setosa
1  IRIS-SETOSA  -setosa
2  IRIS-SETOSA  -setosa
3  IRIS-SETOSA  -setosa
4  IRIS-SETOSA  -setosa
```

The following table describes the string-related operations.

Operation	Description
capitalize	N/A
contains	Returns whether the given string contains a substring. The substring is a regular expression if the `regex` parameter is set to True. The regex parameter is set to True by default.
count	Returns the number of occurrences of the specified string.
endswith	Ends the given string with the specified string.
startswith	Starts the given string with the specified string.
extract	Extracts a regular expression. If the group is not specified, substrings that satisfy the pattern of a regular expression are returned. If the group is specified, the specified group is returned.
find	Searches from left to right and returns the position of the first occurrence of the specified substring. If no matching substring exists, the value -1 is returned.

Operation	Description
rfind	Searches from right to left and returns the position of the first occurrence of the specified substring. If no matching substring exists, the value -1 is returned.
replace	Replaces the substrings that satisfy the pattern of a regular expression with another substring. If you specify the n parameter, the substrings are replaced n times.
get	Returns the string at the specified position.
len	Returns the length of the given string.
ljust	Pads the given string with the character specified by `fillchar` on the right until the string reaches the length specified by `width`. The default pad character is a space.
rjust	Pads the given string with the character specified by `fillchar` on the left until the string reaches the length specified by `width`. The default pad character is a space.
lower	Converts the given string to lowercase.
upper	Converts the given string to uppercase.
lstrip	Removes spaces on the left side of the given string, including blank lines.
rstrip	Removes spaces on the right side of the given string, including blank lines.
strip	Removes spaces on both sides of the given string, including blank lines.
split	Splits the given string at the specified delimiter and returns a value of LIST<STRING> type.
pad	Pads the given string with the character specified by `fillchar` at the specified position, which can be on the left side, right side, or both sides of the string. The default pad character is a space.
repeat	Repeats n times.
slice	Performs slice operations.
swapcase	Converts all the uppercase characters to lowercase and all the lowercase characters to uppercase in the given string.

Operation	Description
title	Returns a titlecased version of the given string where words start with an uppercase character and the remaining characters are lowercase. This operation is the same as the `str.title` operation.
zfill	Pads the given string with the character 0 on the left side of the string until the string reaches the length specified by `width`.
isalnum	Returns True if all characters in the given string are alphanumeric. Otherwise, False is returned. This operation is the same as the `str.isalnum` operation.
isalpha	Returns True if all characters in the given string are alphabetic. Otherwise, False is returned. This operation is the same as the `str.isalpha` operation.
isdigit	Returns True if all the characters in the given string are digits. Otherwise, False is returned. This operation is the same as the `str.isdigit` operation.
isspace	Returns True if all the characters in the given string are spaces. Otherwise, False is returned. This operation is the same as the `str.isspace` operation.
islower	Returns True if all the cased characters in the given string are lowercase. Otherwise, False is returned. This operation is the same as the `str.islower` operation.
isupper	Returns True if all the cased characters in the given string are uppercase. Otherwise, False is returned. This operation is the same as the `str.isupper` operation.
istitle	Returns True if the given string is a titlecased string. Otherwise, False is returned. This operation is the same as the `str.istitle` operation.
isnumeric	Returns True if all characters in the given string are numeric. Otherwise, False is returned. This operation is the same as the `str.isnumeric` operation.
isdecimal	Returns True if all characters in the given string are decimal characters. Otherwise, False is returned. This operation is the same as the `str.isdecimal` operation.

Operation	Description
todict	Splits the given string at the specified delimiter into a dict and returns a value of DICT<STRING, STRING> type. The two input parameters are the project delimiter and key-value delimiter.
strptime	Converts the given string representing a time to the specified format. The time format is the same as the time format in the standard Python library. For more information about the time formats in Python, see Basic date and time types.

Time-related operations

You can call time-related built-in functions to manage sequence or scalar objects of DATETIME type.

```
>>> df = lens[[lens.unix_timestamp.astype('datetime').rename('dt')]]
>>> df[df.dt,
>>>    df.dt.year.rename('year'),
>>>    df.dt.month.rename('month'),
>>>    df.dt.day.rename('day'),
>>>    df.dt.hour.rename('hour')].head(5)
                   dt  year  month  day  hour
0  1998-04-08 11:02:00  1998      4    8    11
1  1998-04-08 10:57:55  1998      4    8    10
2  1998-04-08 10:45:26  1998      4    8    10
3  1998-04-08 10:25:52  1998      4    8    10
4  1998-04-08 10:44:19  1998      4    8    10
```

The following table describes the time-related attributes.

Time-related attribute	Description
year	N/A
month	N/A
day	N/A
hour	N/A
minute	N/A
second	N/A
weekofyear	Returns a number representing the week of the year where the provided date falls. Monday is taken as the first day of a week.
weekday	Returns a number representing the day of the week where the provided date falls.

Time-related attribute	Description
dayofweek	Returns a number representing the day of the week where the provided date falls.
strftime	Converts the given string representing a time to the specified format. The time format is the same as the time in the standard Python library. For more information about the time formats in Python, see Basic date and time types.

PyODPS also supports the addition and subtraction of time. For example, you can retrieve the date three days before the current date. You can subtract one date column from another to obtain the difference in the number of milliseconds.

```
>>> df
                           a                          b
0 2016-12-06 16:43:12.460001 2016-12-06 17:43:12.460018
1 2016-12-06 16:43:12.460012 2016-12-06 17:43:12.460021
2 2016-12-06 16:43:12.460015 2016-12-06 17:43:12.460022
>>> from odps.df import day
>>> df.a - day(3)
                           a
0 2016-12-03 16:43:12.460001
1 2016-12-03 16:43:12.460012
2 2016-12-03 16:43:12.460015
>>> (df.b - df.a).dtype
int64
>>> (df.b - df.a).rename('a')
         a
0  3600000
1  3600000
2  3600000
```

The following table describes the supported DATETIME types.

Type	Description
year	N/A
month	N/A
day	N/A
hour	N/A
minute	N/A
second	N/A
millisecond	N/A

Collection-related operations

PyODPS supports collections of LIST and DICT types. You can use subscripts to retrieve an item from both types. You can also use the `len` method to retrieve the number of items in each collection.

In addition, collections of LIST and DICT types support the `explode` method. You can use the explode method to display the content of a collection. For collections of LIST type, the `explode` method returns one column by default. If you set the `pos` parameter to True, the explode method returns two columns. One of the columns indicates the serial number of each value in the array. The explode method is similar to the `enumerate` method in Python. For collections of DICT type, the `explode` method returns two columns. The two columns indicate the keys and values, respectively. You can pass in column names to the `explode` method as the names of the generated columns.

The following examples show how to use the explode method:

```
>>> df
   id         a                             b
0   1  [a1, b1]  {'a2': 0, 'b2': 1, 'c2': 2}
1   2      [c1]           {'d2': 3, 'e2': 4}
>>> df[df.id, df.a[0], df.b['b2']]
   id   a    b
0   1  a1    1
1   2  c1  NaN
>>> df[df.id, df.a.len(), df.b.len()]
   id  a  b
0   1  2  3
1   2  1  2
>>> df.a.explode()
    a
0  a1
1  b1
2  c1
>>> df.a.explode(pos=True)
   a_pos   a
0      0  a1
1      1  b1
2      0  c1
>>> # Specify column names.
>>> df.a.explode(['pos', 'value'], pos=True)
   pos value
0    0    a1
1    1    b1
2    0    c1
>>> df.b.explode()
  b_key  b_value
0    a2        0
1    b2        1
2    c2        2
3    d2        3
4    e2        4
>>> # Specify column names.
>>> df.b.explode(['key', 'value'])
  key  value
0  a2      0
1  b2      1
2  c2      2
3  d2      3
4  e2      4
```

You can use the `explode` method with Collection together. In this way, the columns generated by the `explode` method are combined with the original columns.

```
>>> df[df.id, df.a.explode()]
   id   a
0   1  a1
1   1  b1
2   2  c1
>>> df[df.id, df.a.explode(), df.b.explode()]
   id   a b_key  b_value
0   1  a1    a2        0
1   1  a1    b2        1
2   1  a1    c2        2
3   1  b1    a2        0
4   1  b1    b2        1
5   1  b1    c2        2
6   2  c1    d2        3
7   2  c1    e2        4
```

In addition to the `len` and `explode` methods, collections of LIST type support the following two methods.

Method	Description
contains(v)	Checks whether the given list contains a specified element.
sort	Sorts the given list and returns a value of LIST type.

Collections of DICT type also support the following two methods.

Method	Description
keys	Retrieves DICT keys and returns a value of LIST type.
values	Retrieves DICT values and returns a value of LIST type.

Other operations

You can use the `isin` operation to check whether the elements of a sequence object exist in a specified collection. You can use the `notin` operation to check whether the elements of a sequence object do not exist in a specified collection.

```
>>> iris.sepallength.isin([4.9, 5.1]).rename('sepallength').head(5)
   sepallength
0         True
1         True
2        False
3        False
4        False
```

You can use the `cut` operation to divide data in a sequence object into several segments.

```
>>> iris.sepallength.cut(range(6), labels=['0-1', '1-2', '2-3', '3-4', '4-5']).rename('sepa
llength_cut').head(5)
   sepallength_cut
0             None
1              4-5
2              4-5
3              4-5
4              4-5
```

You can use the `include_under` and `include_over` operations to specify the maximum and minimum values, respectively.

```
>>> labels = ['0-1', '1-2', '2-3', '3-4', '4-5', '5-']
>>> iris.sepallength.cut(range(6), labels=labels, include_over=True).rename('sepallength_cu
t').head(5)
   sepallength_cut
0              5-
1             4-5
2             4-5
3             4-5
4             4-5
```

Call built-in functions or UDFs in MaxCompute

To call built-in functions or user defined functions (UDFs) in MaxCompute to create columns, you can use the `func` function. The returned value of the func function is of STRING type by default. You can use the `rtype` parameter to specify the type of the returned value.

```
>>> from odps.df import func
>>>
>>> iris[iris.name, func.rand(rtype='float').rename('rand')][:4]
>>> iris[iris.name, func.rand(10, rtype='float').rename('rand')][:4]
>>> # Call UDFs defined in MaxCompute. You must specify the column name if the column name
cannot be automatically determined.
>>> iris[iris.name, func.your_udf(iris.sepalwidth, iris.sepallength, rtype='float').rename(
'new_col')]
>>> # Call UDFs from other projects. You can specify the column name by using the name para
meter.
>>> iris[iris.name, func.your_udf(iris.sepalwidth, iris.sepallength, rtype='float', project
='udf_project', name='new_col')]
```

Note The Pandas backend does not allow you to execute expressions that contain the `func` function.

4.1.6.8. Aggregation

This topic describes the aggregation operations that are supported by Alibaba Cloud MaxCompute SDK for Python (PyODPS) DataFrame and describes how to group and aggregate data and write custom aggregations. PyODPS DataFrame also provides a HyperLogLog operation for you to count the number of distinct values in columns.

```
from odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
```

You can perform the following common aggregation operations on the preceding DataFrame:

- Call the `describe` function to view the quantity, maximum, minimum, mean, and standard deviation of numerical columns in the DataFrame.

  ```
  >>> print(iris.describe())
  ```

  The following result is returned:

  ```
      type  sepal_length  sepal_width  petal_length  petal_width
  0  count    150.000000   150.000000    150.000000   150.000000
  1   mean      5.843333     3.054000      3.758667     1.198667
  2    std      0.828066     0.433594      1.764420     0.763161
  3    min      4.300000     2.000000      1.000000     0.100000
  4    max      7.900000     4.400000      6.900000     2.500000
  ```

- Perform aggregation operations in a single column.

  ```
  >>> iris.sepallength.max()
  ```

  The following result is returned:

  ```
  7.9
  ```

- To aggregate over a distinct sequence of data records, call the `unique` function before you call the corresponding aggregate function.

  ```
  >>> iris.name.unique().cat(sep=',')
  ```

  The following result is returned:

  ```
  u'Iris-setosa,Iris-versicolor,Iris-virginica'
  ```

- If all columns support the same aggregation operation, perform this aggregation operation on the entire DataFrame.

  ```
  >>> iris.exclude('category').mean()
  ```

  The following result is returned:

  ```
     sepal_length  sepal_width  petal_length  petal_width
  1      5.843333     3.054000      3.758667     1.198667
  ```

- Call the `count` function to calculate the total number of rows in the DataFrame.

  ```
  >>> iris.count()
  ```

  The following result is returned:

```
150
```

The following table lists the aggregation operations that Alibaba Cloud MaxCompute SDK for Python (PyODPS) supports.

Aggregation	Description
count or size	Calculates the number of rows.
unique	Calculates the number of distinct values.
min	Calculates the minimum value.
max	Calculates the maximum value.
sum	Calculates the total sum of specified values.
mean	Calculates the mean value.
median	Calculates the median value.
quantile(p)	Calculates the p-quantile. This function returns accurate results only if integers are calculated.
var	Calculates the variance.
std	Calculates the standard deviation.
moment	Calculates the Nth central moment or the Nth moment.
skew	Calculates the sample skewness. This function returns unbiased estimation results.
kurtosis	Calculates the sample kurtosis. This function returns unbiased estimation results.
cat	Concatenates character strings with a separator.
tolist	Aggregates a column into a list.

Note PyODPS DataFrame ignores null values of aggregation operations on columns in MaxCompute and the pandas backend. This is different from pandas DataFrame but is similar to the logic of SQL.

Group and aggregate data

You can use the following methods to group and aggregate data:

- PyODPS DataFrame provides the `groupby` function to group data. After the data is grouped, call the `agg` or `aggregate` function to aggregate the data. The result columns include the grouped column and the aggregated column.

```
>>> iris.groupby('name').agg(iris.sepallength.max(), smin=iris.sepallength.min())
```

The following result is returned:

```
             name  sepallength_max  smin
0     Iris-setosa              5.8   4.3
1  Iris-versicolor             7.0   4.9
2   Iris-virginica             7.9   4.9
```

- PyODPS DataFrame provides the `value_counts` function. After the data is grouped based on a specified column, you can sort the groups in descending order based on the number of distinct values in each group.
  - Call the `groupby` function.

    ```
    >>> iris.groupby('name').agg(count=iris.name.count()).sort('count', ascending=False).he
    ad(5)
    ```

    The following result is returned:

    ```
                 name  count
    0   Iris-virginica     50
    1  Iris-versicolor     50
    2      Iris-setosa     50
    ```

  - Call the `value_counts` function.

    ```
    >>> iris['name'].value_counts().head(5)
    ```

    The following result is returned:

    ```
                 name  count
    0   Iris-virginica     50
    1  Iris-versicolor     50
    2      Iris-setosa     50
    ```

- You can retrieve the column name of a single aggregated column. However, this operation restricts you to use only aggregate functions to manage the values in the aggregated column.

  ```
  >>> iris.groupby('name').petallength.sum()
  ```

  The following result is returned:

  ```
     petallength_sum
  0            73.2
  1           213.0
  2           277.6
  ```

  ```
  >>> iris.groupby('name').agg(iris.petallength.notnull().sum())
  ```

  The following result is returned:

  ```
               name  petallength_sum
  0     Iris-setosa               50
  1  Iris-versicolor              50
  2   Iris-virginica              50
  ```

- You can also group data by constant value. This operation requires Scalar initialization.

```
>>> from odps.df import Scalar
>>> iris.groupby(Scalar(1)).petallength.sum()
```

The following result is returned:

```
   petallength_sum
0            563.8
```

Write custom aggregations

Use the `agg` or `aggregate` function to call custom aggregations on columns. A custom aggregation requires a class to provide the following methods:

- `buffer()`: Returns a mutable object such as LIST or DICT. The `buffer` size must not increase with the amount of data.
- `__call__(buffer, *val)`: Aggregates values to `buffer`.
- `merge(buffer, pbuffer)`: Aggregates `pbuffer` to `buffer`.
- `getvalue(buffer)`: Returns the final value.

The following sample code provides an example to show how to calculate the mean value:

```
class Agg(object):
    def buffer(self):
        return [0.0, 0]
    def __call__(self, buffer, val):
        buffer[0] += val
        buffer[1] += 1
    def merge(self, buffer, pbuffer):
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def getvalue(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] / buffer[1]
```

```
>>> iris.sepalwidth.agg(Agg)
```

The following result is returned:

```
3.0540000000000007
```

When you write custom aggregations, note the following considerations:

- If the data type of the output is different from that of the input, you must specify the data type for the output.

```
>>> iris.sepalwidth.agg(Agg, 'float')
```

- You can use custom aggregations to group and aggregate data.

```
>>> iris.groupby('name').sepalwidth.agg(Agg)
```

The following result is returned:

```
   petallength_aggregation
0                    3.418
1                    2.770
2                    2.974
```

- You can use the `agg` function to call custom aggregations on multiple columns.

```
class Agg(object):
    def buffer(self):
        return [0.0, 0.0]
    def __call__(self, buffer, val1, val2):
        buffer[0] += val1
        buffer[1] += val2
    def merge(self, buffer, pbuffer):
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def getvalue(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] / buffer[1]
```

```
>>> from odps.df import agg
>>> to_agg = agg([iris.sepalwidth, iris.sepallength], Agg, rtype='float')  # Call custom
aggregations on two columns.
>>> iris.groupby('name').agg(val=to_agg)
```

The following result is returned:

```
              name       val
0      Iris-setosa  0.682781
1  Iris-versicolor  0.466644
2   Iris-virginica  0.451427
```

- To call an existing user-defined aggregate function (UDAF) in MaxCompute, you only need to specify the function name.

```
>>> iris.groupby('name').agg(iris.sepalwidth.agg('your_func'))  # Aggregate the values in
a single column.
>>> to_agg = agg([iris.sepalwidth, iris.sepallength], 'your_func', rtype='float')
>>> iris.groupby('name').agg(to_agg.rename('val'))  # Aggregate the values in multiple co
lumns.
```

> **Note** Due to limits of Python user-defined functions (UDFs), you cannot specify the LIST or DICT type as the input or output data type for custom aggregations.

HyperLogLog counting

PyODPS DataFrame provides the `hll_count` operation. You can call this HyperLogLog operation to count the number of distinct values in a column. This operation returns an estimated number. If large amounts of data is calculated, you can call this operation to estimate the number of distinct values.

For example, you can call this operation to calculate the number of unique visitors (UVs) and obtain an estimated number in a short period of time.

```
>>> df = DataFrame(pd.DataFrame({'a': np.random.randint(100000, size=100000)}))
>>> df.a.hll_count()
```

The following result is returned:

```
63270
```

```
>>> df.a.nunique()
```

The following result is returned:

```
63250
```

Note The `splitter` parameter is used to split columns and calculate the number of distinct values.

4.1.6.9. Sort, deduplicate, sample, and transform data

This topic describes the operations you can perform on a DataFrame. You can sort, deduplicate, sample, and scale data. You can also process null value.

```
from odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
```

Sort data

You can sort data only in a collection.

- Call the `sort` method or `sort_values` method to sort data.

```
>>> iris.sort('sepalwidth').head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.2         2.2          4.5         1.5  Iris-versicolor
2          6.0         2.2          5.0         1.5   Iris-virginica
3          6.0         2.2          4.0         1.0  Iris-versicolor
4          5.5         2.3          4.0         1.3  Iris-versicolor
```

- To sort data in descending order, set the `ascending` parameter to False.

```
>>> iris.sort('sepalwidth', ascending=False).head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.7         4.4          1.5         0.4  Iris-setosa
1          5.5         4.2          1.4         0.2  Iris-setosa
2          5.2         4.1          1.5         0.1  Iris-setosa
3          5.8         4.0          1.2         0.2  Iris-setosa
4          5.4         3.9          1.3         0.4  Iris-setosa
```

- You can also set `-` to sort data in descending order.

```
>>> iris.sort(-iris.sepalwidth).head(5)
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.7         4.4          1.5         0.4  Iris-setosa
1          5.5         4.2          1.4         0.2  Iris-setosa
2          5.2         4.1          1.5         0.1  Iris-setosa
3          5.8         4.0          1.2         0.2  Iris-setosa
4          5.4         3.9          1.3         0.4  Iris-setosa
```

- You can sort multiple fields simultaneously in a DataFrame.

```
>>> iris.sort(['sepalwidth', 'petallength']).head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          4.0         1.0  Iris-versicolor
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          5.0         1.5   Iris-virginica
4          4.5         2.3          1.3         0.3      Iris-setosa
```

- If you need to sort multiple fields in different orders, add a list of BOOLEAN type values for the `ascending` parameter. The length of the list must be equal to the length of the sorted fields.

```
>>> iris.sort(['sepalwidth', 'petallength'], ascending=[True, False]).head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          5.0         1.5   Iris-virginica
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          4.0         1.0  Iris-versicolor
4          6.3         2.3          4.4         1.3  Iris-versicolor
```

  You can also sort multiple fields in different orders in the following way:

```
>>> iris.sort(['sepalwidth', -iris.petallength]).head(5)
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          5.0         1.5   Iris-virginica
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          4.0         1.0  Iris-versicolor
4          6.3         2.3          4.4         1.3  Iris-versicolor
```

> **Note** MaxCompute requires that you specify the number of records you need to sort. The default value is 10000. You can use the `options.df.odps.sort.limit` parameter to modify the number of records. You can set the parameter to a value greater than 10000. However, this setting can cause an out-of-memory (OOM) error.

Deduplicate data

- To deduplicate a collection, you can call the `distinct` method in the following ways:

```
>>> iris[['name']].distinct()
             name
0      Iris-setosa
1  Iris-versicolor
2   Iris-virginica
```

```
>>> iris.distinct('name')
             name
0      Iris-setosa
1  Iris-versicolor
2   Iris-virginica
```

```
>>> iris.distinct('name', 'sepallength').head(3)
          name  sepallength
0  Iris-setosa          4.3
1  Iris-setosa          4.4
2  Iris-setosa          4.5
```

- To deduplicate a sequence, you can call the `unique` method. However, the sequence that is deduplicated by using the `unique` method cannot be selected in a collection.

```
>>> iris.name.unique()
             name
0      Iris-setosa
1  Iris-versicolor
2   Iris-virginica
```

An error occurs when you use the following code:

```
>>> iris[iris.name, iris.name.unique()]
```

Sample data

To sample data from a collection, call the `sample` method. Python on MaxCompute (PyODPS) supports the following four sampling methods:

> **Note** MaxCompute DataFrame must support XFlow to execute the following sampling methods, except for sampling by parts. If MaxCompute DataFrame does not support XFlow, you can only execute the sampling methods at the backend of Pandas DataFrame.

- Sampling by parts

  Data is divided into `parts` by using this sampling method. You can select the part by number.

```
>>> iris.sample(parts=10)  # Split data into 10 parts and take the part numbered as 0 by default.
>>> iris.sample(parts=10, i=0)  # Split data into 10 parts and take the part numbered as 0.
>>> iris.sample(parts=10, i=[2, 5])   # Split data into 10 parts and take the parts numbered as 2 and 5.
>>> iris.sample(parts=10, columns=['name', 'sepalwidth'])  # Sample the data by the values of the name and sepalwidth columns.
```

- Sampling by proportion or the number of records

  You must specify the number of records or the proportion of data that you want to sample when you use this method. To enable sampling with replacement, set the replace parameter to True.

```
>>> iris.sample(n=100)  # Sample 100 records.
>>> iris.sample(frac=0.3)  # Sample 30% of all records.
```

- Sampling by weight

  You can specify the weight column, the number of records, and the proportion of records that you want to sample when you use this method. To enable sampling with replacement, set the `replace` parameter to True.

```
>>> iris.sample(n=100, weights='sepal_length')
>>> iris.sample(n=100, weights='sepal_width', replace=True)
```

- Sampling by stratification

  You can specify the label column that you want to stratify when you use this method. You can also specify the sampling proportion (the `frac` parameter) or the number of records (the `n` parameter) for each label. This sampling method does not support sampling with replacement.

```
>>> iris.sample(strata='category', n={'Iris Setosa': 10, 'Iris Versicolour': 10})
>>> iris.sample(strata='category', frac={'Iris Setosa': 0.5, 'Iris Versicolour': 0.4})
```

Scale data

A DataFrame supports data scaling based on the maximum and minimum values, average value, and standard deviation. The following data is an example for data scaling:

```
name  id  fid
0  name1   4  5.3
1  name2   2  3.5
2  name2   3  1.5
3  name1   4  4.2
4  name1   3  2.2
5  name1   3  4.1
```

- You can use the `min_max_scale` method to normalize data.

```
df.min_max_scale(columns=['fid'])
    name  id       fid
0  name1   4  1.000000
1  name2   2  0.526316
2  name2   3  0.000000
3  name1   4  0.710526
4  name1   3  0.184211
5  name1   3  0.684211
```

- You can use the `min_max_scale` method with the `feature_range` parameter to specify the output value range. The following example shows how to keep the output values in the range of (-1, 1):

```
df.min_max_scale(columns=['fid'], feature_range=(-1, 1))
    name  id       fid
0  name1   4  1.000000
1  name2   2  0.052632
2  name2   3 -1.000000
3  name1   4  0.421053
4  name1   3 -0.631579
5  name1   3  0.368421
```

- If you need to keep the original values, you must use the `preserve` parameter. The scaled data is added in a new column. By default, the new column is named by adding the *_scaled* suffix to the original column name. You can use the `suffix` parameter to change the suffix name.

```
df.min_max_scale(columns=['fid'], preserve=True)
    name  id  fid  fid_scaled
0  name1   4  5.3    1.000000
1  name2   2  3.5    0.526316
2  name2   3  1.5    0.000000
3  name1   4  4.2    0.710526
4  name1   3  2.2    0.184211
5  name1   3  4.1    0.684211
```

- You can also use the `min_max_scale` method with the `group` parameter to specify one or more group columns and to retrieve the minimum and maximum values from the specified column to scale data.

```
df.min_max_scale(columns=['fid'], group=['name'])
    name  id       fid
0  name1   4  1.000000
1  name1   4  0.645161
2  name1   3  0.000000
3  name1   3  0.612903
4  name2   2  1.000000
5  name2   3  0.000000
```

  The preceding example shows that data in both `name1` and `name2` is scaled based on the minimum and maximum values of the two groups.

- You can use the `std_scale` method to scale data based on standard normal distribution. You can also use the `std_scale` method with the `preserve` parameter to keep the original column, and use this method with the `group` parameter to group data.

```
df.std_scale(columns=['fid'])
    name  id       fid
0  name1   4  1.436467
1  name2   2  0.026118
2  name2   3 -1.540938
3  name1   4  0.574587
4  name1   3 -0.992468
5  name1   3  0.496234
```

Process null value

A DataFrame supports the feature of deleting rows with null value and the feature of filling null value. The following data is an example for processing null value:

```
id   name   f1   f2   f3   f4
0   0  name1  1.0  NaN  3.0  4.0
1   1  name1  2.0  NaN  NaN  1.0
2   2  name1  3.0  4.0  1.0  NaN
3   3  name1  NaN  1.0  2.0  3.0
4   4  name1  1.0  NaN  3.0  4.0
5   5  name1  1.0  2.0  3.0  4.0
6   6  name1  NaN  NaN  NaN  NaN
```

- You can use the `dropna` method to delete the rows that contain null value in the `subset` object.

```
df.dropna(subset=['f1', 'f2', 'f3', 'f4'])
   id   name   f1   f2   f3   f4
0   5  name1  1.0  2.0  3.0  4.0
```

- To keep the rows that contain non-null values, set the `how` parameter to all.

```
df.dropna(how='all', subset=['f1', 'f2', 'f3', 'f4'])
   id   name   f1   f2   f3   f4
0   0  name1  1.0  NaN  3.0  4.0
1   1  name1  2.0  NaN  NaN  1.0
2   2  name1  3.0  4.0  1.0  NaN
3   3  name1  NaN  1.0  2.0  3.0
4   4  name1  1.0  NaN  3.0  4.0
5   5  name1  1.0  2.0  3.0  4.0
```

- You can use the `thresh` parameter to specify the minimum number of non-null values in a row.

```
df.dropna(thresh=3, subset=['f1', 'f2', 'f3', 'f4'])
   id   name   f1   f2   f3   f4
0   0  name1  1.0  NaN  3.0  4.0
2   2  name1  3.0  4.0  1.0  NaN
3   3  name1  NaN  1.0  2.0  3.0
4   4  name1  1.0  NaN  3.0  4.0
5   5  name1  1.0  2.0  3.0  4.0
```

- You can call the `fillna` method to replace null value with a specified constant or values in an existing column.
  - The following example shows how to replace null value with a constant:

```
df.fillna(100, subset=['f1', 'f2', 'f3', 'f4'])
   id   name     f1     f2     f3     f4
0   0  name1    1.0  100.0    3.0    4.0
1   1  name1    2.0  100.0  100.0    1.0
2   2  name1    3.0    4.0    1.0  100.0
3   3  name1  100.0    1.0    2.0    3.0
4   4  name1    1.0  100.0    3.0    4.0
5   5  name1    1.0    2.0    3.0    4.0
6   6  name1  100.0  100.0  100.0  100.0
```

- The following example shows how to replace null value with values in an existing column:

```
df.fillna(df.f2, subset=['f1', 'f2', 'f3', 'f4'])
   id   name   f1   f2   f3   f4
0   0  name1  1.0  NaN  3.0  4.0
1   1  name1  2.0  NaN  NaN  1.0
2   2  name1  3.0  4.0  1.0  4.0
3   3  name1  1.0  1.0  2.0  3.0
4   4  name1  1.0  NaN  3.0  4.0
5   5  name1  1.0  2.0  3.0  4.0
6   6  name1  NaN  NaN  NaN  NaN
```

- A DataFrame supports backward filling and forward filling to fill null value. The following table defines the valid values of the `method` parameter.

Valid value	Definition
bfill or backfill	Backward filling
ffill or pad	Forward filling

Examples

```
df.fillna(method='bfill', subset=['f1', 'f2', 'f3', 'f4'])
   id   name   f1   f2   f3   f4
0   0  name1  1.0  3.0  3.0  4.0
1   1  name1  2.0  1.0  1.0  1.0
2   2  name1  3.0  4.0  1.0  NaN
3   3  name1  1.0  1.0  2.0  3.0
4   4  name1  1.0  3.0  3.0  4.0
5   5  name1  1.0  2.0  3.0  4.0
6   6  name1  NaN  NaN  NaN  NaN
df.fillna(method='ffill', subset=['f1', 'f2', 'f3', 'f4'])
   id   name   f1   f2   f3   f4
0   0  name1  1.0  1.0  3.0  4.0
1   1  name1  2.0  2.0  2.0  1.0
2   2  name1  3.0  4.0  1.0  1.0
3   3  name1  NaN  1.0  2.0  3.0
4   4  name1  1.0  1.0  3.0  4.0
5   5  name1  1.0  2.0  3.0  4.0
6   6  name1  NaN  NaN  NaN  NaN
```

You can use the `ffill` or `bfill` function in the preceding example to simplify the code. The `ffill` function equals `fillna(method='ffill')`, and the `bfill` function equals `fillna(method='bfill')`.

4.1.6.10. Use UDFs and the third-party Python libraries

This topic describes how to use user-defined functions (UDFs) and the third-party Python libraries.

Use UDFs

DataFrame allows you to use the `map` method for a Sequence object to call UDFs on all of its elements.

```
>>> iris.sepallength.map(lambda x: x + 1).head(5)
   sepallength
0          6.1
1          5.9
2          5.7
3          5.6
4          6.0
```

Note When you use UDFs, you cannot use List or Dict types as input or output.

If the type of the Sequence is changed after the `map` method is executed, you need to explicitly specify the new type of the Sequence.

```
>>> iris.sepallength.map(lambda x: 't'+str(x), 'string').head(5)
   sepallength
0         t5.1
1         t4.9
2         t4.7
3         t4.6
4         t5.0
```

If a function contains a closure, the change of a closure variable value outside the function causes the variable value within the function to change.

```
>>> dfs = []
>>> for i in range(10):
>>>     dfs.append(df.sepal_length.map(lambda x: x + i))
```

As a result, each `SequenceExpr` object in `dfs` is `df.sepal_length + 9`. To solve this problem, you can use the function as the return value of another function, or use `partial`. See the following two examples.

```
>>> dfs = []
>>> def get_mapper(i):
>>>     return lambda x: x + i
>>> for i in range(10):
>>>     dfs.append(df.sepal_length.map(get_mapper(i)))
```

```
>>> import functools
>>> dfs = []
>>> for i in range(10):
>>>     dfs.append(df.sepal_length.map(functools.partial(lambda v, x: x + v, i)))
```

The `map` method also supports existing UDFs. You can pass in str type parameters, which represent function names, or Function objects. For more information, see Functions.

When you implement the `map` method for Python functions, you need to use MaxCompute Python UDF. If your project does not support Python UDFs, you cannot use the `map` method. In addition, all Python UDF limits apply.

The only available third-party library (contains the code in C language) is NumPy. For more information about how to use the third-party Python libraries, see Use the third-party Python libraries.

In addition to UDFs, DataFrame provides many built-in functions, some of which are implemented by using the `map` function. Therefore, if your project does not support Python UDF, you cannot use these functions. Note that Alibaba Cloud public services do not support Python UDF.

> **Note** Because of the difference in byte code definitions, if you use the new features supported by Python 3 (such as `yield from`), errors may occur when the code is executed on MaxCompute Worker of Python 2.7. Therefore, we recommend that you make sure your code runs normally before you write production code by using MapReduce API in Python 3.

Example of counter usage:

```
from odps.udf import get_execution_context
def h(x):
    ctx = get_execution_context()
    counters = ctx.get_counters()
    counters.get_counter('df', 'add_one').increment(1)
    return x + 1
df.field.map(h)
```

You can find the counter value in JSONSummary of LogView.

Use UDFs for one row

To use UDFs for one row, you can use the `apply` method. The `axis` parameter must be set to 1 to indicate that the operation is performed on the row. The UDF of the `apply` method takes a parameter, which is a row of data from the preceding Collection object. You can retrieve the data in a field by using the attribute or offset.

- If `the reduce parameter is set to True, a Sequence object is returned. Otherwise, a Collection object is returned. You can use the names` and `types` parameters to specify the field names and types of the returned Sequence or Collection object. By default, if you do not specify a type, STRING type is used.

  ```
  >>> iris.apply(lambda row: row.sepallength + row.sepalwidth, axis=1, reduce=True, types='
  float').rename('sepaladd').head(3)
     sepaladd
  0       8.6
  1       7.9
  2       7.9
  ```

- If `reduce` is False in the UDF of the `apply` method, you can use the `yield` keyword to return multiple rows of results.

```
>>> iris.count()
150
>>>
>>> def handle(row):
>>>     yield row.sepallength - row.sepalwidth, row.sepallength + row.sepalwidth
>>>     yield row.petallength - row.petalwidth, row.petallength + row.petalwidth
>>>
>>> iris.apply(handle, axis=1, names=['iris_add', 'iris_sub'], types=['float', 'float']).
count()
300
```

- You can also annotate the returned field names and types in the function. You do not need to specify them when you call the the function.

```
>>> from odps.df import output
>>>
>>> @output(['iris_add', 'iris_sub'], ['float', 'float'])
>>> def handle(row):
>>>     yield row.sepallength - row.sepalwidth, row.sepallength + row.sepalwidth
>>>     yield row.petallength - row.petalwidth, row.petallength + row.petalwidth
>>>
>>> iris.apply(handle, axis=1).count()
300
```

- You can also use `map_reduce` of `map-only`, which is equivalent to the `apply` method with `axis=1`.

```
>>> iris.map_reduce(mapper=handle).count()
300
```

- To use an existing user-defined table function (UDTF) in MaxCompute, specify the name of the UDTF.

```
>>> iris['name', 'sepallength'].apply('your_func', axis=1, names=['name2', 'sepallength2'
], types=['string', 'float'])
```

- When you use the `apply` method for rows and `reduce` is False, you can use the lateral view with the existing rows for purposes such as aggregation.

```
>>> from odps.df import output
>>>
>>> @output(['iris_add', 'iris_sub'], ['float', 'float'])
>>> def handle(row):
>>>     yield row.sepallength - row.sepalwidth, row.sepallength + row.sepalwidth
>>>     yield row.petallength - row.petalwidth, row.petallength + row.petalwidth
>>>
>>> iris[iris.category, iris.apply(handle, axis=1)]
```

Use custom aggregations for all columns

When you use the `apply` method, if `axis` is not specified or the value of `axis` is 0, you can pass in a custom aggregation class to aggregate all Sequence objects.

```
class Agg(object):
    def buffer(self):
        return [0.0, 0]
    def __call__(self, buffer, val):
        buffer[0] += val
        buffer[1] += 1
    def merge(self, buffer, pbuffer):
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def getvalue(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] / buffer[1]
```

```
>>> iris.exclude('name').apply(Agg)
   sepallength_aggregation  sepalwidth_aggregation  petallength_aggregation  petalwidth_agg
regation
0                 5.843333                   3.054                 3.758667
1.198667
```

Note When you use UDFs, you cannot use the LIST or DICT types as input or output.

Reference resources

UDFs can also read MaxCompute resources (such as table and file resources) or reference a Collection object as resources. To do this, you need to write the UDFs as a closure or callable class. See the following two examples.

```
>>> file_resource = o.create_resource('pyodps_iris_file', 'file', file_obj='Iris-setosa')
>>>
>>> iris_names_collection = iris.distinct('name')[:2]
>>> iris_names_collection
       sepallength
0      Iris-setosa
1  Iris-versicolor
```

```
>>> def myfunc(resources):  # resources are passed in by calling order
>>>     names = set()
>>>     fileobj = resources[0] # file resources are represented by a file-like object
>>>     for l in fileobj:
>>>         names.add(l)
>>>     collection = resources[1]
>>>     for r in collection:
>>>         names.add(r.name)  # retrieve the values by using the field name or offset
>>>     def h(x):
>>>         if x in names:
>>>             return True
>>>         else:
>>>             return False
>>>     return h
>>>
>>> df = iris.distinct('name')
>>> df = df[df.name,
>>>         df.name.map(myfunc, resources=[file_resource, iris_names_collection], rtype='bo
olean').rename('isin')]
>>>
>>> df
              name   isin
0      Iris-setosa   True
1  Iris-versicolor   True
2   Iris-virginica  False
```

Note When the partitioned tables are read, partition fields are not included.

If `axis` is 1 in a row operation, you need to write a function closure or callable class. For aggregation operations on columns, you only need to use the `__init__` function to read resources.

```
>>> words_df
                    sentence
0                Hello World
1               Hello Python
2  Life is short I use Python
>>>
>>> import pandas as pd
>>> stop_words = DataFrame(pd.DataFrame({'stops': ['is', 'a', 'I']}))
>>>
>>> @output(['sentence'], ['string'])
>>> def filter_stops(resources):
>>>     stop_words = set([r[0] for r in resources[0]])
>>>     def h(row):
>>>         return ' '.join(w for w in row[0].split() if w not in stop_words),
>>>     return h
>>>
>>> words_df.apply(filter_stops, axis=1, resources=[stop_words])
               sentence
0           Hello World
1          Hello Python
2  Life short use Python
```

Note In this example, `stop_words` is a local variable, which is referenced as a resource in MaxCompute during execution.

Use the third-party Python libraries

You can upload the third-party Python packages as resources to MaxCompute. Supported formats include *whl*, *egg*, *zip*, and *tar.gz*. When you use a global method or an immediately-invoked method, you need to specify the packages you need before you can use the third-party libraries in UDFs. You also need to specify the dependencies of the third-party libraries. Otherwise, errors may occur when you import the files.

- You can upload resources by calling the PyODPS resource upload API `create_resource` operation.

  Take a *python-dateutil* package as an example.

  i. You can run the `pip download` command to download the package and its dependencies to a specific path. Two packages are downloaded: *six-1.10.0-py2.py3-none-any.whl* and *python_dateutil-2.5.3-py2.py3-none-any.whl*. Note that the downloaded packages must support the Linux environment.

  ```
  $ pip download python-dateutil -d /to/path/
  ```

  ii. Then upload the two files to MaxCompute as resources.

  ```
  >>> # make sure that file extensions are correct.
  >>> odps.create_resource('six.whl', 'file', file_obj=open('six-1.10.0-py2.py3-none-any.whl', 'rb'))
  >>> odps.create_resource('python_dateutil.whl', 'file', file_obj=open('python_dateutil-2.5.3-py2.py3-none-any.whl', 'rb'))
  ```

  iii. Now you have a DataFrame object that only contains a STRING field.

```
>>> df
               datestr
0  2016-08-26 14:03:29
1  2015-08-26 14:03:29
```

iv. Use the following third-party libraries in global configurations.

```
>>> from odps import options
>>>
>>> def get_year(t):
>>>     from dateutil.parser import parse
>>>     return parse(t).strftime('%Y')
>>>
>>> options.df.libraries = ['six.whl', 'python_dateutil.whl']
>>> df.datestr.map(get_year)
   datestr
0     2016
1     2015
```

Alternatively, use the `libraries` parameter of a method to specify the third-party libraries.

```
>>> def get_year(t):
>>>     from dateutil.parser import parse
>>>     return parse(t).strftime('%Y')
>>>
>>> df.datestr.map(get_year).execute(libraries=['six.whl', 'python_dateutil.whl'])
   datestr
0     2016
1     2015
```

By default, PyODPS supports the third-party libraries that contain pure Python code but no file operations. In later versions of MaxCompute, PyODPS also supports Python libraries that contain binary code or file operations. These libraries must be suffixed with cp27-cp27m-manylinux1_x86_64 and uploaded as archives. The *.whl* packages must be renamed to *.zip* files. You also need to set odps.isolation.session.enable to True or enable `isolation` in your project. The following example shows how to upload and use the special functions in `scipy`.

```
>>> # packages containing binary code must be uploaded by using the archive method, and t
he file extension .whl must be replaced with .zip.
>>> odps.create_resource('scipy.zip', 'archive', file_obj=open('scipy-0.19.0-cp27-cp27m-m
anylinux1_x86_64.whl', 'rb'))
>>>
>>> # if isolation is enabled for your project, the following option is optional.
>>> options.sql.settings = { 'odps.isolation.session.enable': True }
>>>
>>> def psi(value):
>>>     # we recommend that you import the third-party libraries inside the function to a
void the errors caused by the varying structure of a binary package between different ope
rating systems.
>>>     from scipy.special import psi
>>>     return float(psi(value))
>>>
>>> df.float_col.map(psi).execute(libraries=['scipy.zip'])
```

Binary packages that only contain source code can be packed into Wheel files and uploaded in Linux Shell. The Wheel files generated in Mac and Windows cannot be used in MaxCompute.

```
python setup.py bdist_wheel
```

- You can also use MaxCompute Console to upload resources.
  i. Most of the Python packages are supplied with .whl packages, including the packages containing binary files on various platforms. Therefore, you need to first find the packages that can run on MaxCompute.
  ii. In addition, if you want to use a third-party package in MaxCompute, all dependencies must be included, which is complex. The following table lists the dependencies of the packages.

Package name	Dependencies
pandas	numpy, python-dateutil, pytz, six
scipy	numpy
scikit-learn	numpy, scipy

> **Note** The numpy package is already provided. You only need to upload python-dateutil, pytz, pandas, SciPy, sklearn, and six packages to run the pandas, scipy, and scikit-learn packages.

  iii. You can find python-dateutil-2.6.0.zip in python-dateutils and download it.

  iv. Rename the downloaded file to python-dateutil.zip, and upload the file as a resource in the MaxCompute console.

```
add archive python-dateutil.zip;
```

> **Note** Find, download, and upload pytz-2017.2.zip and six-1.11.0.tar.gz files in the same way as you do for the python-dateutil file.

  v. To ensure that the packages that contain code in C language, such as Pandas, run on MaxCompute normally, you need to find the .whl package whose name contains cp27-cp27m-manylinux1_x86_64. Therefore, you need to find and download pandas-0.20.2-cp27-cp27m-manylinux1_x86_64.whl, change its extension to .zip, and run `add archive pandas.zip;` on MaxCompute Console to upload it. Upload scipy and scikit-learn packages to MaxCompute in the same way as you do in the preceding procedure.

The following table lists the resources to be downloaded for all packages.

Package name	File name	Name of resource for upload
python-dateutil	python-dateutil-2.6.0.zip	python-dateutil.zip
pytz	pytz-2017.2.zip	pytz.zip
six	six-1.11.0.tar.gz	six.tar.gz
pandas	pandas-0.20.2-cp27-cp27m-manylinux1_x86_64.zip	pandas.zip
scipy	scipy-0.19.0-cp27-cp27m-manylinux1_x86_64.zip	scipy.zip
scikit-learn	scikit_learn-0.18.1-cp27-cp27m-manylinux1_x86_64.zip	sklearn.zip

Specify the third-party Python libraries

- Globally specify the libraries to be used.

```
>>> from odps import options
>>> options.df.libraries = ['six.whl', 'python_dateutil.whl']
```

- In an immediately-invoked method, locally specify the libraries to be used.

```
>>> df.apply(my_func, axis=1).to_pandas(libraries=['six.whl', 'python_dateutil.whl'])
```

4.1.6.11. MapReduce API

This topic describes the MapReduce API.

PyODPS DataFrame supports the MapReduce API. You can separately write the `map` and `reduce` functions, because a `map_reduce` process can contain only `mappers` or `reducers`.

The following example shows how to execute the `WordCount` program:

```
>>> #encoding=utf-8
>>> from odps import ODPS
>>> from odps import options
>>> options.verbose = True
>>> o = ODPS('your-access-id', 'your-secret-access-key',project='DMP_UC_dev', endpoint='htt
p://service-corp.odps.aliyun-inc.com/api')
>>> from odps.df import DataFrame
>>> def mapper(row):
>>>     for word in row[0].split():
>>>         yield word.lower(), 1
>>>
>>> def reducer(keys):
>>>     # A list rather than cnt=0 is used. If you use cnt=0, cnt in the h function is cons
idered a local variable, whose value is not included in the output.
>>>     cnt = [0]
>>>     def h(row, done):  # done indicates that the rows with this key are iterated.
>>>         cnt[0] += row[1]
>>>         if done:
>>>             yield keys[0], cnt[0]
>>>     return h
>>> # The zx_word_count table has only one column, which is of the STRING type.
>>> word_count = DataFrame(o.get_table('zx_word_count'))
>>> table = word_count.map_reduce(mapper, reducer, group=['word', ],
                        mapper_output_names=['word', 'cnt'],
                        mapper_output_types=['string', 'int'],
                        reducer_output_names=['word', 'cnt'],
                        reducer_output_types=['string', 'int'])
         word  cnt
0         are    1
1         day    1
2      doing?    1
3   everybody    1
4       first    1
5       hello    2
6         how    1
7          is    1
8          so    1
9         the    1
10       this    1
11      world    1
12        you    1
```

Use the `group` parameter to specify the field by which you want to use the `reduce` function to group data. If you do not specify this parameter, the data is grouped by all the fields. `Reducers` receive and initialize aggregated `keys`, and process the rows that are aggregated based on the `keys`. `done` indicates that all rows related to these `keys` are iterated.

For ease of understanding, the function is written as a closure in this example. You can also write the function as a `callable` class.

```
class reducer(object):
    def __init__(self, keys):
        self.cnt = 0
    def __call__(self, row, done):  # done indicates that the rows with this key are iterat
ed.
        self.cnt += row.cnt
        if done:
            yield row.word, self.cnt
```

The code is simplified if you use `output` for commenting.

```
>>> from odps.df import output
>>>
>>> @output(['word', 'cnt'], ['string', 'int'])
>>> def mapper(row):
>>>     for word in row[0].split():
>>>         yield word.lower(), 1
>>>
>>> @output(['word', 'cnt'], ['string', 'int'])
>>> def reducer(keys):
>>>     # A list rather than cnt=0 is used. If you use cnt=0, cnt in the h function is cons
idered a local variable, whose value is not included in the output.
>>>     cnt = [0]
>>>     def h(row, done):  # done indicates that the rows with this key are iterated.
>>>         cnt[0] += row.cnt
>>>         if done:
>>>             yield keys.word, cnt[0]
>>>     return h
>>>
>>> word_count = DataFrame(o.get_table('zx_word_count'))
>>> table = word_count.map_reduce(mapper, reducer, group='word')
         word  cnt
0         are    1
1         day    1
2      doing?    1
3   everybody    1
4       first    1
5       hello    2
6         how    1
7          is    1
8          so    1
9         the    1
10       this    1
11      world    1
12        you    1
```

During iteration, you can use the `sort` parameter to sort the data by column and the `ascending` parameter to specify the sorting order. The `ascending` parameter can be a BOOL value, which indicates that all the fields specified by the `sort` parameter are arranged in the same order. The ascending parameter can also be a list. The number of strings in the list must be the same as the number of fields specified by the `sort` parameter.

Specify a combiner

In the `MapReduce` API, a `combiner` is used to aggregate data in a `mapper`. A combiner is used in the same way as a `reducer`, but a combiner cannot reference resources. The names and data types of the fields generated from a `combiner` must be the same as those from the `mapper` that corresponds to the combiner.

In this example, you can use the `reducer` as the `combiner` to aggregate data in the `mapper` to reduce shuffled data.

```
>>> words_df.map_reduce(mapper, reducer, combiner=reducer, group='word')
```

Reference resources

In the MapReduce API, you can separately specify the resources referenced by `mappers` and `reducers`.

In the following example, deprecated word filtering is performed in the `mapper`, and the number of words in a whitelist in the `reducer` is plus 5.

```
>>> white_list_file = o.create_resource('pyodps_white_list_words', 'file', file_obj='Python
\nWorld')
>>>
>>> @output(['word', 'cnt'], ['string', 'int'])
>>> def mapper(resources):
>>>     stop_words = set(r[0].strip() for r in resources[0])
>>>     def h(row):
>>>         for word in row[0].split():
>>>             if word not in stop_words:
>>>                 yield word, 1
>>>     return h
>>>
>>> @output(['word', 'cnt'], ['string', 'int'])
>>> def reducer(resources):
>>>     d = dict()
>>>     d['white_list'] = set(word.strip() for word in resources[0])
>>>     d['cnt'] = 0
>>>     def inner(keys):
>>>         d['cnt'] = 0
>>>         def h(row, done):
>>>             d['cnt'] += row.cnt
>>>             if done:
>>>                 if row.word in d['white_list']:
>>>                     d['cnt'] += 5
>>>                 yield keys.word, d['cnt']
>>>         return h
>>>     return inner
>>>
>>> words_df.map_reduce(mapper, reducer, group='word',
>>>                     mapper_resources=[stop_words], reducer_resources=[white_list_file])
     word  cnt
0   hello    2
1    life    1
2  python    7
3   world    6
4   short    1
5     use    1
```

Use third-party Python libraries

The usage is similar to that of a third-party Python library in map stages.

- Specify the library globally:

```
>>> from odps import options
>>> options.df.libraries = ['six.whl', 'python_dateutil.whl']
```

- If you use an immediately-invoked method, specify the library locally:

```
>>> df.map_reduce(mapper=my_mapper, reducer=my_reducer, group='key').execute(libraries=['
six.whl', 'python_dateutil.whl'])
```

> **Note** Due to the difference in bytecode definitions, if you write code based on new features supported by Python 3, such as `yield from`, an error is reported when the code is executed on a MaxCompute worker of Python 2.7. Before you implement production operations by using the MapReduce API of Python 3, we recommend that you make sure that the code runs normally.

Reshuffle data

If data is unevenly distributed in a cluster, you can call the `reshuffle` method to reshuffle data.

```
>>> df1 = df.reshuffle()
```

By default, data is hashed as random numbers. You can also distribute data by column and sort the reshuffled data in a specific order.

```
>>> df1.reshuffle('name', sort='id', ascending=False)
```

Bloom filter

PyODPS DataFrame provides the `bloom_filter` interface to calculate data with a Bloom filter.

If you specify a collection object and its `sequence1` for column calculation, you can apply the Bloom filter to `sequence2`. This way, the data that exists in `sequence1` rather than `sequence2` is filtered out, but the data that is not in sequence2 may not be completely filtered out. This is an approximate method. This method quickly filters out useless data from the collection object. The Bloom filter is suitable for `JOIN` operations in the scenario where data volumes are significantly different. Most data is not `joined` in this scenario. For example, most user visits do not create transactions. If you want to `join` visit data with transaction data, you can apply the Bloom filter to visit data before you `join` visit data with transaction data. This enhances operation performance.

```
>>> df1 = DataFrame(pd.DataFrame({'a': ['name1', 'name2', 'name3', 'name1'], 'b': [1, 2, 3,
4]}))
>>> df1
       a  b
0  name1  1
1  name2  2
2  name3  3
3  name1  4
>>> df2 = DataFrame(pd.DataFrame({'a': ['name1']}))
>>> df2
       a
0  name1
>>> df1.bloom_filter('a', df2.a) # The row 0 can be a computation expression, for example,
df1.a + '1'.
       a  b
0  name1  1
1  name1  4
```

Description:

- Small amounts of data are processed. Therefore, the rows that contain `name2` and `name3` in

column `a` of `df1` are filtered out. However, if the system processes large amounts of data, the system may not filter out all the data that meets the specified conditions.

- In the preceding `JOIN` operation, the data that is not filtered out does not affect the data accuracy. However, data filtering greatly enhances the `JOIN` performance.
- You can specify the `capacity` and `error_rate` parameters to configure the data volume and error rate. The default values are `3000` and `0.01`.

> **Note** If you increase the value of the `capacity` parameter or decrease the value of the `error_rate` parameter, the usage of memory is increased. Therefore, set the parameters to proper values based on your requirements.

For more information about collection objects, see DataFrame execution.

Pivot table

PyODPS DataFrame provides the pivot table feature. The following code shows the sample table data:

```
>>> df
     A    B      C  D  E
0  foo  one  small  1  3
1  foo  one  large  2  4
2  foo  one  large  2  5
3  foo  two  small  3  6
4  foo  two  small  3  4
5  bar  one  large  4  5
6  bar  one  small  5  3
7  bar  two  small  6  2
8  bar  two  large  7  1
```

- If you use the pivot table feature, the `rows` parameter is required to obtain the average value based on one or more fields.

```
>>> df['A', 'D', 'E'].pivot_table(rows='A')
     A  D_mean  E_mean
0  bar     5.5    2.75
1  foo     2.2    4.40
```

- You can specify multiple fields in `rows` to aggregate data based on the fields.

```
>>> df.pivot_table(rows=['A', 'B', 'C'])
     A    B      C  D_mean  E_mean
0  bar  one  large     4.0     5.0
1  bar  one  small     5.0     3.0
2  bar  two  large     7.0     1.0
3  bar  two  small     6.0     2.0
4  foo  one  large     2.0     4.5
5  foo  one  small     1.0     3.0
6  foo  two  small     3.0     5.0
```

- You can use the `values` parameter to specify the column that you want to calculate.

```
>>> df.pivot_table(rows=['A', 'B'], values='D')
     A    B    D_mean
0  bar  one  4.500000
1  bar  two  6.500000
2  foo  one  1.666667
3  foo  two  3.000000
```

- By default, the average value is calculated. You can use the `aggfunc` parameter to specify one or more aggregate functions.

```
>>> df.pivot_table(rows=['A', 'B'], values=['D'], aggfunc=['mean', 'count', 'sum'])
     A    B    D_mean  D_count  D_sum
0  bar  one  4.500000        2      9
1  bar  two  6.500000        2     13
2  foo  one  1.666667        3      5
3  foo  two  3.000000        2      6
```

- You can use the values of an original column as the values of a column in the new collection object.

```
>>> df.pivot_table(rows=['A', 'B'], values='D', columns='C')
     A    B  large_D_mean  small_D_mean
0  bar  one           4.0           5.0
1  bar  two           7.0           6.0
2  foo  one           2.0           1.0
3  foo  two           NaN           3.0
```

- You can use `fill_value` to replace empty values.

```
>>> df.pivot_table(rows=['A', 'B'], values='D', columns='C', fill_value=0)
     A    B  large_D_mean  small_D_mean
0  bar  one             4             5
1  bar  two             7             6
2  foo  one             2             1
3  foo  two             0             3
```

Key-value string conversion

DataFrame can extract key-value pairs into columns, and convert standard columns to key-value pairs. For more information about how to create and manage a DataFrame object, see Create a DataFrame object.

- Extract key-value pairs into columns. Example:

```
>>> df
    name               kv
0  name1  k1=1,k2=3,k5=10
1  name1    k1=7.1,k7=8.2
2  name2    k2=1.2,k3=1.5
3  name2      k9=1.1,k2=1
```

Use the `extract_kv` method to extract key-value fields:

```
>>> df.extract_kv(columns=['kv'], kv_delim='=', item_delim=',')
   name   kv_k1  kv_k2  kv_k3  kv_k5  kv_k7  kv_k9
0  name1    1.0    3.0    NaN   10.0    NaN    NaN
1  name1    7.0    NaN    NaN    NaN    8.2    NaN
2  name2    NaN    1.2    1.5    NaN    NaN    NaN
3  name2    NaN    1.0    NaN    NaN    NaN    1.1
```

The `columns` parameter specifies the fields that you want to extract. The `kv_delim` parameter specifies the delimiter between the keys and values. The `item_delim` parameter specifies the delimiter between key-value pairs. If you do not specify the parameters, keys and values are separated with colons (:), and key-value pairs are separated with commas (,). The name of the output field is the combination of the original field name and key value. The name and key value are connected by using an underscore (`_`). The default value for a missing column is NONE. You can use `fill_value` to fill the values for missing columns.

```
>>> df.extract_kv(columns=['kv'], kv_delim='=', fill_value=0)
   name   kv_k1  kv_k2  kv_k3  kv_k5  kv_k7  kv_k9
0  name1    1.0    3.0    0.0   10.0    0.0    0.0
1  name1    7.0    0.0    0.0    0.0    8.2    0.0
2  name2    0.0    1.2    1.5    0.0    0.0    0.0
3  name2    0.0    1.0    0.0    0.0    0.0    1.1
```

- Convert multiple columns to key-value pairs. Example:

```
>>> df
   name    k1   k2   k3    k5   k7   k9
0  name1  1.0  3.0  NaN  10.0  NaN  NaN
1  name1  7.0  NaN  NaN   NaN  8.2  NaN
2  name2  NaN  1.2  1.5   NaN  NaN  NaN
3  name2  NaN  1.0  NaN   NaN  NaN  1.1
```

Use the `to_kv` method to convert data to the key-value format:

```
>>> df.to_kv(columns=['k1', 'k2', 'k3', 'k5', 'k7', 'k9'], kv_delim='=')
    name               kv
0  name1  k1=1,k2=3,k5=10
1  name1    k1=7.1,k7=8.2
2  name2    k2=1.2,k3=1.5
3  name2      k9=1.1,k2=1
```

4.1.6.12. Data merging

This topic describes the join and union operations that are supported by Alibaba Cloud MaxCompute SDK for Python (PyODPS) DataFrame to merge data in tables.

Import data from MaxCompute tables and create DataFrame objects. The following sample code provides an example to show how to create DataFrame objects:

```
>>> from odps.df import DataFrame
>>> movies = DataFrame(o.get_table('pyodps_ml_100k_movies'))
>>> ratings = DataFrame(o.get_table('pyodps_ml_100k_ratings'))
>>> movies.dtypes
odps.Schema {
  movie_id                       int64
  title                          string
  release_date                   string
  video_release_date             string
  imdb_url                       string
}
>>> ratings.dtypes
odps.Schema {
  user_id                int64
  movie_id               int64
  rating                 int64
  unix_timestamp         int64
}
```

Join operation

PyODPS DataFrame allows you to `join` two Collection objects.

- If you do not specify `join` conditions, the DataFrame API uses the columns of the same name to `join` the Collection objects.

```
>>> movies.join(ratings).head(3)
   movie_id              title  release_date  video_release_date
imdb_url  user_id  rating  unix_timestamp
0         3  Four Rooms (1995)   01-Jan-1995                    http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...       49       3       888068877
1         3  Four Rooms (1995)   01-Jan-1995                    http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      621       5       881444887
2         3  Four Rooms (1995)   01-Jan-1995                    http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      291       3       874833936
```

- You can specify explicit `join` conditions. For a `join` operation, if the column names specified in the `on` condition for the two Collection objects are the same, the system uses one of the specified columns for the new table.

```
>>> movies.join(ratings, on='movie_id').head(3)
   movie_id              title  release_date  video_release_date
imdb_url  user_id  rating  unix_timestamp
0         3  Four Rooms (1995)   01-Jan-1995                    http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...       49       3       888068877
1         3  Four Rooms (1995)   01-Jan-1995                    http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      621       5       881444887
2         3  Four Rooms (1995)   01-Jan-1995                    http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      291       3       874833936
```

- For other types of `join` operations such as `left_join`, if the column names specified in the on condition for the two Collection objects are the same, the system renames the specified columns for the new table.

```
>>> movies.left_join(ratings, on='movie_id').head(3)
   movie_id_x              title  release_date  video_release_date
imdb_url  user_id  movie_id_y  rating  unix_timestamp
0           3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/
title-exact?Four%20Rooms%...       49           3       3       888068877
1           3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/
title-exact?Four%20Rooms%...      621           3       5       881444887
2           3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/
title-exact?Four%20Rooms%...      291           3       3       874833936
```

In the preceding sample code, the two `movie_id` columns are renamed as `movie_id_x` and `movie_id_y`. The renaming rule depends on the `suffixes` parameter. By default, the value of the suffixes parameter is `('_x', '_y')`. When columns of the same name are found, the system renames the columns by using the specified suffixes.

```
>>> ratings2 = ratings[ratings.exclude('movie_id'), ratings.movie_id.rename('movie_id2')]
>>> ratings2.dtypes
odps.Schema {
  user_id                      int64
  rating                       int64
  unix_timestamp               int64
  movie_id2                    int64
}
>>> movies.join(ratings2, on=[('movie_id', 'movie_id2')]).head(3)
   movie_id              title  release_date  video_release_date
imdb_url  user_id  rating  unix_timestamp  movie_id2
0         3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...       49       3       888068877          3
1         3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      621       5       881444887          3
2         3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      291       3       874833936          3
```

- You can specify an expression that uses the equality operator in the `on` condition to rename columns for the new table.

```
>>> movies.join(ratings2, on=[movies.movie_id == ratings2.movie_id2]).head(3)
   movie_id              title  release_date  video_release_date
imdb_url  user_id  rating  unix_timestamp  movie_id2
0         3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...       49       3       888068877          3
1         3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      621       5       881444887          3
2         3  Four Rooms (1995)   01-Jan-1995                     http://us.imdb.com/M/ti
tle-exact?Four%20Rooms%...      291       3       874833936          3
```

- If you perform a `self-join` operation, you can use the `view` method to retrieve columns from the left and right Collection objects.

```
>>> movies2 = movies.view()
>>> movies.join(movies2, movies.movie_id == movies2.movie_id)[movies, movies2.movie_id.re
name('movie_id2')].head(3)
   movie_id            title_x release_date_x video_release_date_x  \
0         2   GoldenEye (1995)    01-Jan-1995                 True
1         3  Four Rooms (1995)    01-Jan-1995                 True
2         4  Get Shorty (1995)    01-Jan-1995                 True
                                        imdb_url_x  movie_id2
0  http://us.imdb.com/M/title-exact?GoldenEye%20(...          2
1  http://us.imdb.com/M/title-exact?Four%20Rooms%...          3
2  http://us.imdb.com/M/title-exact?Get%20Shorty%...          4
```

PyODPS DataFrame supports the `left_join`, `right_join`, and `outer_join` operations in addition to the `join` operation. In the left_join, right_join, and outer_join operations, renamed columns are suffixed with `_x` and `_y` by default. You can use a 2-`tuple` to define the suffixes in the `suffixes` parameter.

- When you perform the left_join, right_join, or outer_join operation, to avoid duplicate columns in the new table, set the `merge_columns` option to True. The system then selects non-null values from the duplicate columns as the values in the new column.

```
>>> movies.left_join(ratings, on='movie_id', merge_columns=True)
```

To use the `mapjoin` operation, set `mapjoin` to True. The system then performs the `mapjoin` operation on the right Collection object. You can `join` a Collection object from MaxCompute to a Collection object from pandas, or `join` a Collection object from MaxCompute to a Collection object from a database. In both cases, MaxCompute executes the computation.

Union operation

If two tables have the same columns of which the data types are the same, you can perform the `union` or `concat` operation to combine the tables into a single table regardless of how the columns are ordered.

```
>>> mov1 = movies[movies.movie_id < 3]['movie_id', 'title']
>>> mov2 = movies[(movies.movie_id > 3) & (movies.movie_id < 6)]['title', 'movie_id']
>>> mov1.union(mov2)
   movie_id             title
0         1   Toy Story (1995)
1         2   GoldenEye (1995)
2         4  Get Shorty (1995)
3         5     Copycat (1995)
```

You can `union` a Collection object from MaxCompute to a Collection object from pandas, or `union` a Collection object from MaxCompute to a Collection object from a database. In both cases, MaxCompute executes the computation.

4.1.6.13. Window functions

This topic describes window functions that are supported by the DataFrame API.

```
grouped = iris.groupby('name')
grouped.mutate(grouped.sepallength.cumsum(), grouped.sort('sepallength').row_number()).head
(10)
          name  sepallength_sum  row_number
0  Iris-setosa            250.3           1
1  Iris-setosa            250.3           2
2  Iris-setosa            250.3           3
3  Iris-setosa            250.3           4
4  Iris-setosa            250.3           5
5  Iris-setosa            250.3           6
6  Iris-setosa            250.3           7
7  Iris-setosa            250.3           8
8  Iris-setosa            250.3           9
9  Iris-setosa            250.3          10
```

- Use window functions to select columns.

```
iris['name', 'sepallength', iris.groupby('name').sort('sepallength').sepallength.cumcount
()].head(5)
          name  sepallength  sepallength_count
0  Iris-setosa          4.3                  1
1  Iris-setosa          4.4                  2
2  Iris-setosa          4.4                  3
3  Iris-setosa          4.4                  4
4  Iris-setosa          4.5                  5
```

- Use window functions to aggregate data by scalar. The processing method is a combination of grouping and aggregation.

```
from odps.df import Scalar
iris.groupby(Scalar(1)).sort('sepallength').sepallength.cumcount()
```

The following table lists the window functions that are supported by the DataFrame API.

Window function	Description
cumsum	Calculates the cumulative sum.
cummean	Calculates the cumulative mean.
cummedian	Calculates the cumulative median.
cumstd	Calculates the cumulative standard deviation.
cummax	Calculates the cumulative maximum.
cummin	Calculates the cumulative minimum.
cumcount	Calculates the cumulative sum.

Window function	Description
lag	Retrieves the value of a row at a given offset that precedes the current row. The system determines the number of the row to retrieve the value based on the following formula: Number of the current row - Offset value.
lead	Retrieves the value of a row at a given offset that follows the current row. The system determines the number of the row to retrieve the value based on the following formula: Number of the current row + Offset value.
rank	Calculates the rank of a row in an ordered group.
dense_rank	Calculates the dense rank of a row in an ordered group.
percent_rank	Calculates the relative rank of a row in an ordered group.
row_number	Calculates the row number. Row numbers start from 1.
qcut	Divides a group of data into N bins based on the data sequence and returns the number of the bin that contains the current data. If data is not evenly distributed in bins, more data is distributed to the first bin by default.
nth_value	Retrieves the Nth value in the group.
cume_dist	Calculates the proportion of rows whose values are no greater than the current value to all rows in a group.

The following tables list parameters that are supported by different window functions.

- The `rank`, `dense_rank`, `percent_rank`, and `row_number` window functions support the following parameters.

Parameter	Description
sort	The keyword that is used to sort rows. This parameter is an empty string by default.
ascending	Specifies whether to sort rows in ascending order. This parameter is set to True by default.

- The `lag` and `lead` window functions support the following parameters in addition to the parameters that are supported by the `rank` function.

Parameter	Description
offset	The number of rows that are between the current row and the row where you want to retrieve data.
default	The value to return if the row at the specified offset does not exist.

- The `cumsum`, `cummax`, `cummin`, `cummean`, `cummedian`, `cumcount`, and `cumstd` window functions support the following parameters in addition to the parameters that are supported by the `rank` function.

Parameter	Description
unique	Specifies whether to enable data deduplication. This parameter is set to False by default.
preceding	The start of a window.
following	The end of a window.

4.1.6.14. Plotting

This topic describes the plotting feature that is provided by Alibaba Cloud MaxCompute SDK for Python (PyODPS) DataFrame.

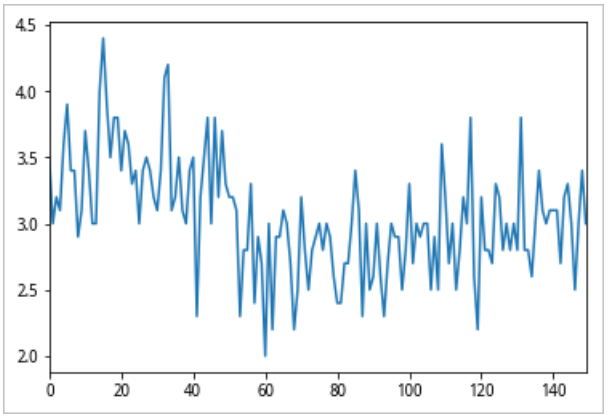
To enable the plotting feature, install the pandas and Matplotlib libraries.

Run the `pip install matplotlib` command to install the Matplotlib library, and run the following sample code in Jupyter to create plots.

Sample code for plotting

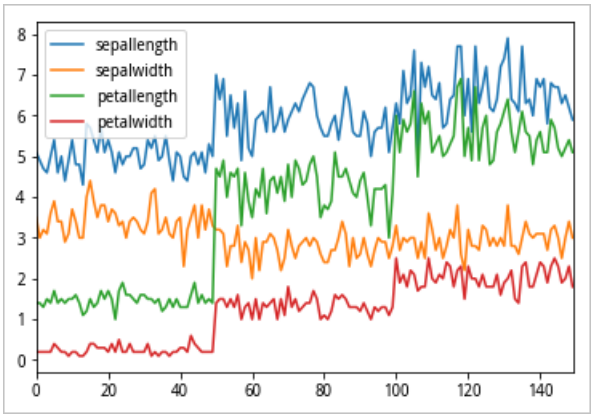
- Single-line plot

```
>>> from odps.df import DataFrame
>>> iris = DataFrame(o.get_table('pyodps_iris'))
>>> %matplotlib inline
>>> iris.sepalwidth.plot()
<matplotlib.axes._subplots.AxesSubplot at 0x10c2b3510>
```



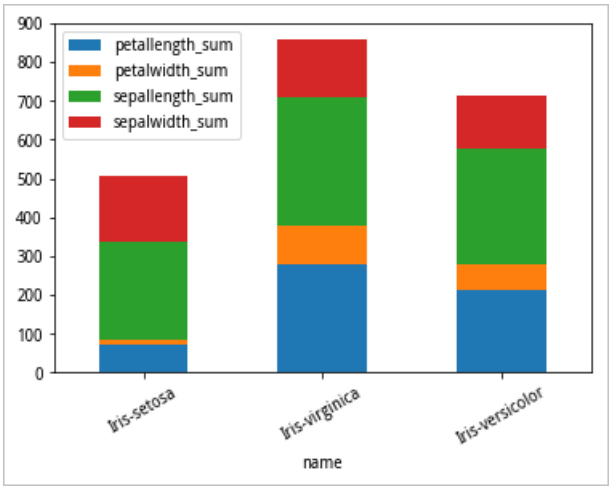
- Multi-line plot

```
>>> iris.plot()
<matplotlib.axes._subplots.AxesSubplot at 0x10db7e690>
```



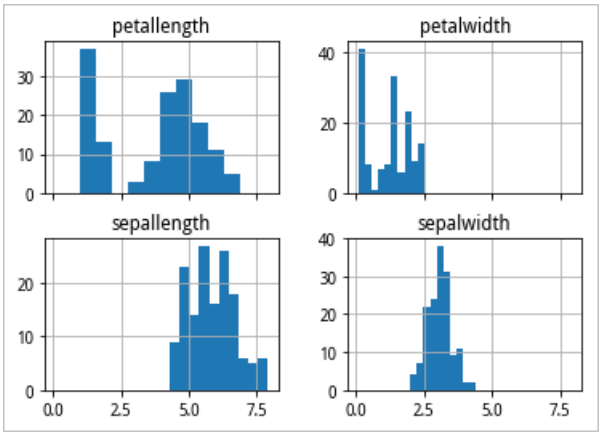
- Vertical bar plot

```
>>> iris.groupby('name').sum().plot(kind='bar', x='name', stacked=True, rot=30)
<matplotlib.axes._subplots.AxesSubplot at 0x10c5f2090>
```



- Histogram

```
>>> iris.hist(sharex=True)
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x10e013f90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x10e2d1c10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x10e353f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x10e3c4410>]], dtype=object)
```



The `kind` parameter specifies the plot type. The following table lists the plot types that are supported by PyODPS DataFrame. For more information, see pandas.DataFrame.plot.

kind	Description
line	The line plot
bar	The vertical bar plot
barh	The horizontal bar plot
hist	The histogram
box	The box plot
kde	The kernel density estimation plot
density	The kernel density estimation plot
area	The area plot
pie	The pie plot
scatter	The scatter plot
hexbin	The hexagonal bin plot

The `plot` function also supports the following parameters.

Parameter	Description
xlabel	Specifies the label for the x-axis.
ylabel	Specifies the label for the y-axis.
xlabelsize	Specifies the size of label for the x-axis.
ylabelsize	Specifies the size of the label for the y-axis.
labelsize	Specifies the size of the label for the axis.

Parameter	Description
title	Specifies the name of the title.
titlesize	Specifies the size of the title.
annotate	Specifies whether to add an annotation.

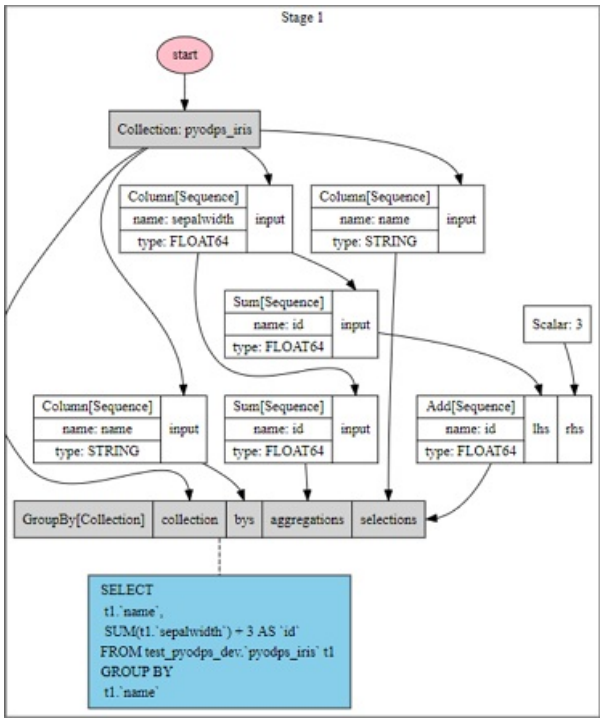
4.1.6.15. Debugging

Alibaba Cloud MaxCompute SDK for Python (PyODPS) DataFrame optimizes the execution process of each operation. You can use the visualization feature to display the entire computation process.

DataFrame visualization

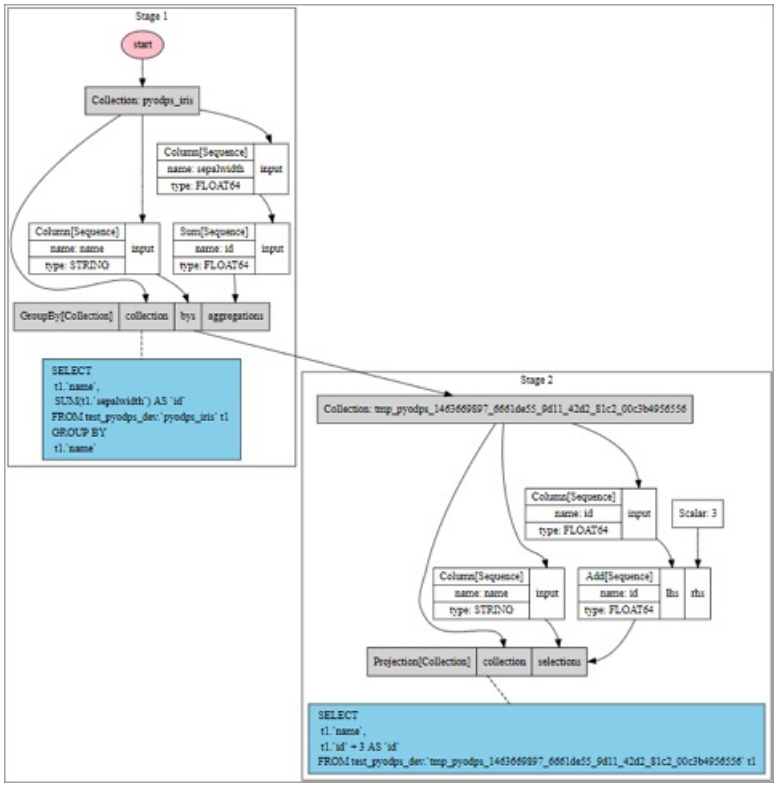
DataFrame visualization depends on graphviz and the *graphviz* package for Python.

```
>>> df = iris.groupby('name').agg(id=iris.sepalwidth.sum())
>>> df = df[df.name, df.id + 3]
>>> df.visualize()
```



The computation process shows that PyODPS DataFrame combines the `groupby` operation and the column filtering operation.

```
>>> df = iris.groupby('name').agg(id=iris.sepalwidth.sum()).cache()
>>> df2 = df[df.name, df.id + 3]
>>> df2.visualize()
```



The entire execution process runs in two steps because a cache operation is performed.

View compiled results at the MaxCompute SQL backend

Call the `compile` method to view compiled results of SQL statements at the MaxCompute SQL backend.

```
>>> df = iris.groupby('name').agg(sepalwidth=iris.sepalwidth.max())
>>> df.compile()
Stage 1:
SQL compiled:
SELECT
  t1.`name`,
  MAX(t1.`sepalwidth`) AS `sepalwidth`
FROM test_pyodps_dev.`pyodps_iris` t1
GROUP BY
  t1.`name`
```

Perform local debugging by using the pandas computation backend

If you use MaxCompute tables as the sources for DataFrame objects, the system does not compile or perform some operations at the MaxCompute SQL backend. Instead, the system uses the MaxCompute Tunnel service to download data. This way, the system does not need to wait for the MaxCompute SQL backend to schedule download tasks. This allows you to download small amounts of data from MaxCompute to a local directory and use the pandas computation backend to compile and debug code. You can perform the following debugging operations:

- Select all or some rows of data from a non-partitioned table, or filter the column data, and then calculate the number of specified rows. The column filtering operation does not compute column values.

- Select all or some rows of data from all or the first several partition columns that are specified in a partitioned table, or filter the column data, and then calculate the number of the specified rows.

Assume that the `iris` DataFrame object uses a non-partitioned MaxCompute table as the source. The following operation uses the MaxCompute Tunnel service to download data:

```
>>> iris.count()
>>> iris['name', 'sepalwidth'][:10]
```

Assume that the DataFrame object uses a partitioned table that includes the `ds`, `hh`, and `mm` partition fields. The following operation uses the MaxCompute Tunnel service to download data:

```
>>> df[:10]
>>> df[df.ds == '20160808']['f0', 'f1']
>>> df[(df.ds == '20160808') & (df.hh == 3)][:10]
>>> df[(df.ds == '20160808') & (df.hh == 3) & (df.mm == 15)]
```

In this case, you can use the `to_pandas` method to download the data to a local directory for debugging.

```
>>> DEBUG = True
>>> if DEBUG:
>>> df = iris[:100].to_pandas(wrap=True)
>>> else:
>>> df = iris
```

At the end of compiling, set `DEBUG` to False to complete computation on MaxCompute.

> **Note** Some programs that pass local debugging may fail to run on MaxCompute because of the limits of the sandbox.

4.1.7. User experience enhancement

4.1.7.1. Command line

PyODPS provides an enhanced command line tool.

After you configure an account, you do not need to enter the account information again. You can perform the following steps to configure an account and call objects:

1. Import the PyODPS enhancement tool.

   ```
   >>> from odps.inter import setup, enter, teardown
   ```

2. Configure your account.

   ```
   >>> setup('**your-access_id**', '**your-access-key**', '**your-project**', endpoint='**
   your-endpoint**')
   ```

   > **Note** If you do not specify the `room` parameter, the `default`, `room` is used.

3. Call the enter method to create a room object on any Python interactive interface.

```
>>> room = enter()
>>> o = room.odps
>>> o.get_table('table_name')
odps.Table
  name: odps_test_sqltask_finance.`table_name`
  schema:
    c_int_a                : bigint
    c_int_b                : bigint
    c_double_a             : double
    c_double_b             : double
    c_string_a             : string
    c_string_b             : string
    c_bool_a               : boolean
    c_bool_b               : boolean
    c_datetime_a           : datetime
    c_datetime_b           : datetime
```

> **Note** The MaxCompute object is not automatically updated when you change the setup of the `room`. You must call the `enter` method again to retrieve the new `room` object.

After you configure an account and call objects, you can store, retrieve, or delete objects in the `room` or delete the entire `room`.

- You can store commonly used MaxCompute tables or resources in the `room`.

```
>>> room.store('stored-table', o.get_table('table_name'), desc='Simple stored table examp
le')
```

You can call the `display` method to display the stored objects as a table.

```
>>> room.display()
default name         desc
stored-table     Simple stored table example
iris          Iris dataset
```

- You can run the `room['stored-table']` or `room.iris` command to retrieve the stored objects.

```
>>> room['stored-table']
odps.Table
  name: odps_test_sqltask_finance.`table_name`
  schema:
    c_int_a                : bigint
    c_int_b                : bigint
    c_double_a             : double
    c_double_b             : double
    c_string_a             : string
    c_string_b             : string
    c_bool_a               : boolean
    c_bool_b               : boolean
    c_datetime_a           : datetime
    c_datetime_b           : datetime
```

- You can call the `drop` method to delete objects from the `room`.

```
>>> room.drop('stored-table')
>>> room.display()
default name         desc
iris          Iris dataset
```

- You can call the `teardown` method to delete a `room`. If no parameters are specified, the default `room` is deleted.

```
teardown()
```

4.1.7.2. IPython

PyODPS provides an IPython plug-in to simplify the operations on MaxCompute.

IPython enhancement

Some commands are provided for command line enhancement.

- Run the following commands to load the plug-in:

```
%load_ext odps
%enter
```

The following result is returned:

```
<odps.inter.Room at 0x11341df10>
```

- In this case, the `o` and `odps` global variables can be retrieved. You can run the `o.get_table` or `odps.get_table` command to call a table.

```
o.get_table('table_name')
odps.get_table('table_name')
```

The following result is returned:

```
odps.Table
  name: odps_test_sqltask_finance.`table_name`
  schema:
    c_int_a                : bigint
    c_int_b                : bigint
    c_double_a             : double
    c_double_b             : double
    c_string_a             : string
    c_string_b             : string
    c_bool_a               : boolean
    c_bool_b               : boolean
    c_datetime_a           : datetime
    c_datetime_b           : datetime
```

- Run the following command to display the stored objects as a table:

```
%stores
```

default name	desc
iris	Iris dataset

Object name completion

PyODPS enhances the code completion feature that is provided by IPython. When you write a statement such as `o.get_xxx`, the object name is automatically completed. In the following examples, <tab> is used to denote pressing the Tab key. When you enter the statement and encounter <tab>, press the Tab key.

- You can use the Tab key to complete the object name.

```
o.get_table(<tab>
```

- You can enter the first few characters of an object name and press the Tab key to complete it.

```
o.get_table('tabl<tab>
```

  IPython auto-completes the table name that starts with `tabl`.

- This feature also completes the names of objects in different projects. Syntax:

```
o.get_table(project='project_name', name='tabl<tab>
o.get_table('tabl<tab>', project='project_name')
```

- If multiple matching objects exist, IPython provides a list. `options.completion_size` specifies the maximum number of objects in the list. The default value is 10.

SQL statements

PyODPS provides an SQL plug-in to execute MaxCompute SQL statements.

- You can use %sql to execute a single-line SQL statement.

```
In [37]: %sql select * from pyodps_iris limit 5
|==========================================|   1 /  1  (100.00%)         3s
Out[37]:
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
```

- You can use `%%sql` to execute a multiple-line SQL statement.

```
In [38]: %%sql
   ....: select * from pyodps_iris
   ....: where sepallength < 5
   ....: limit 5
   ....:
|========================================|   1 /  1  (100.00%)        15s
Out[38]:
   sepallength  sepalwidth  petallength  petalwidth         name
0          4.9         3.0          1.4         0.2  Iris-setosa
1          4.7         3.2          1.3         0.2  Iris-setosa
2          4.6         3.1          1.5         0.2  Iris-setosa
3          4.6         3.4          1.4         0.3  Iris-setosa
4          4.4         2.9          1.4         0.2  Iris-setosa
```

- To execute parameterized SQL statements, you can use `:parameter` to specify the parameter.

```
In [1]: %load_ext odps
In [2]: mytable = 'table_name'
In [3]: %sql select * from :mytable
|========================================|   1 /  1  (100.00%)         2s
Out[3]:
   c_int_a  c_int_b  c_double_a  c_double_b  c_string_a  c_string_b c_bool_a  \
0        0        0       -1203           0           0       -1203     True
  c_bool_b         c_datetime_a         c_datetime_b
0    False  2012-03-30 23:59:58  2012-03-30 23:59:59
```

- For SQL runtime parameters, you can use `%set` to set a global parameter or use SET within an SQLCell to set a local parameter. The following example sets a local parameter, which does not affect the global settings.

```
In [17]: %%sql
         set odps.sql.mapper.split.size = 16;
         select * from pyodps_iris;
```

The following example sets a global parameter. Global settings are applied to all subsequent SQL statements.

```
In [18]: %set odps.sql.mapper.split.size = 16
```

Upload pandas DataFrame objects to MaxCompute tables

PyODPS provides the commands to upload pandas DataFrame objects to MaxCompute tables. You only need to run the `%persist` command. The first parameter `df` is the variable name. The second parameter `pyodps_pandas_df` is the MaxCompute table name.

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.arange(9).reshape(3, 3), columns=list('abc'))
%persist df pyodps_pandas_df
```

4.1.7.3. Jupyter Notebook

PyODPS enhances the result exploration and progress display features of Jupyter Notebook.

Result exploration

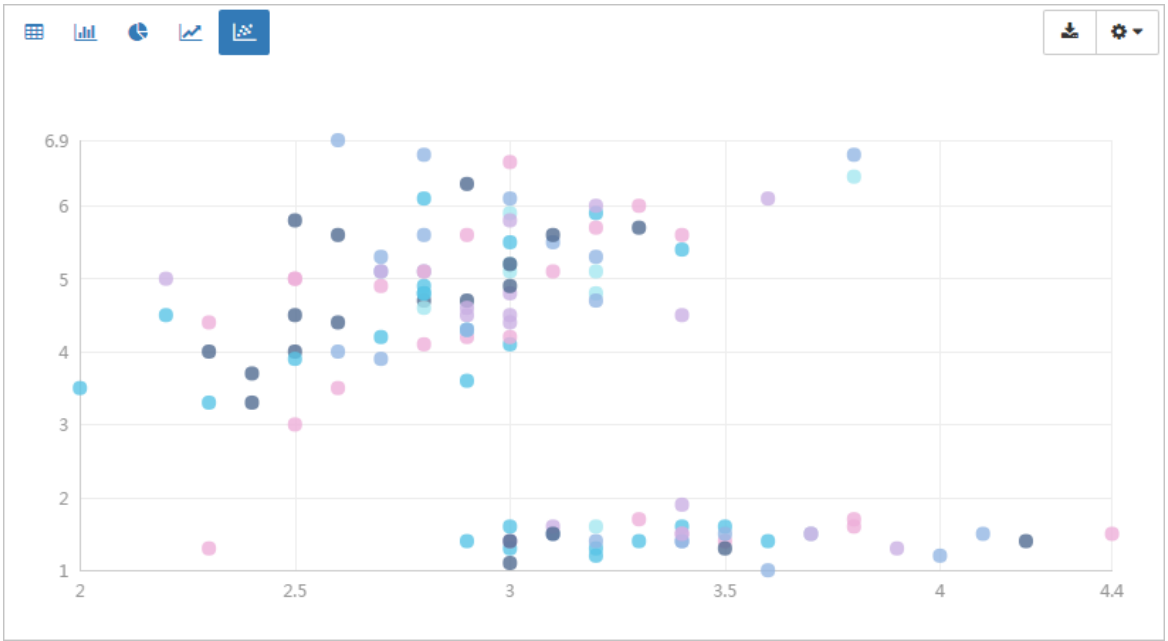
PyODPS provides a data exploration feature in Jupyter Notebook for SQLCell and DataFrame. You can use interactive data exploration tools to browse local data and create graphs.

1. If the execution result is a DataFrame object, PyODPS reads the result and displays it in a paged table. You can click a page number, the Previous button, or the Next button to browse the data.

sepallength	sepalwidth	petallength	petalwidth	name
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

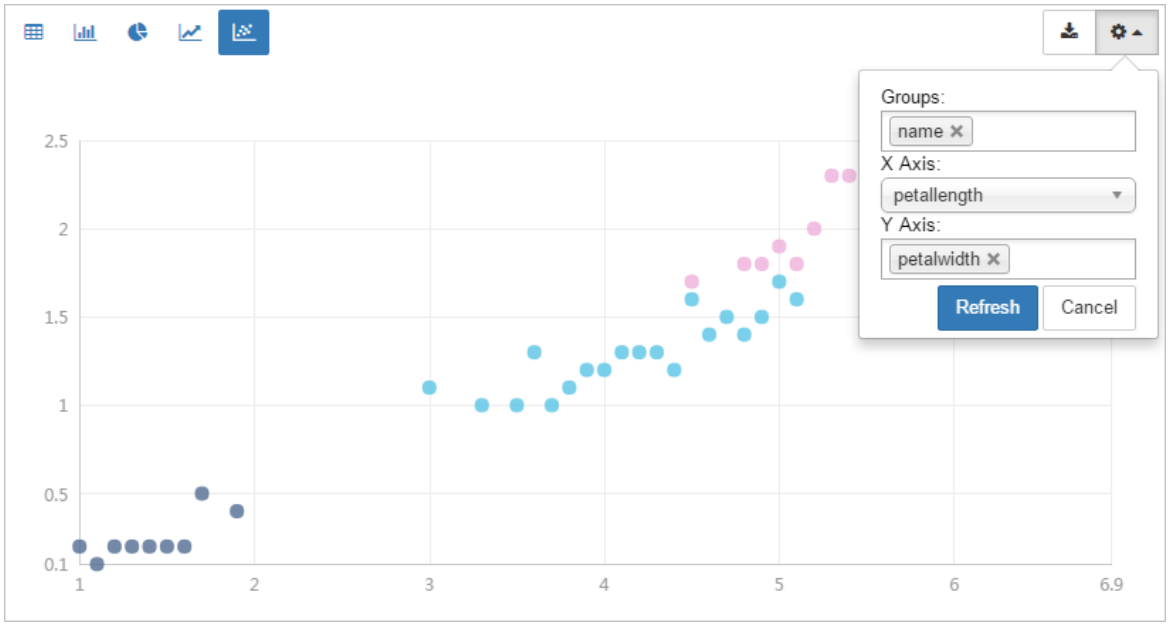
« 1 2 3 4 5 6 7 ... 15 »

2. You can select other display modes on the top of the table to display the result in a column chart, pie chart, line chart, or scatter chart. The following figure shows a scatter chart created based on the default fields, which are the first three fields.



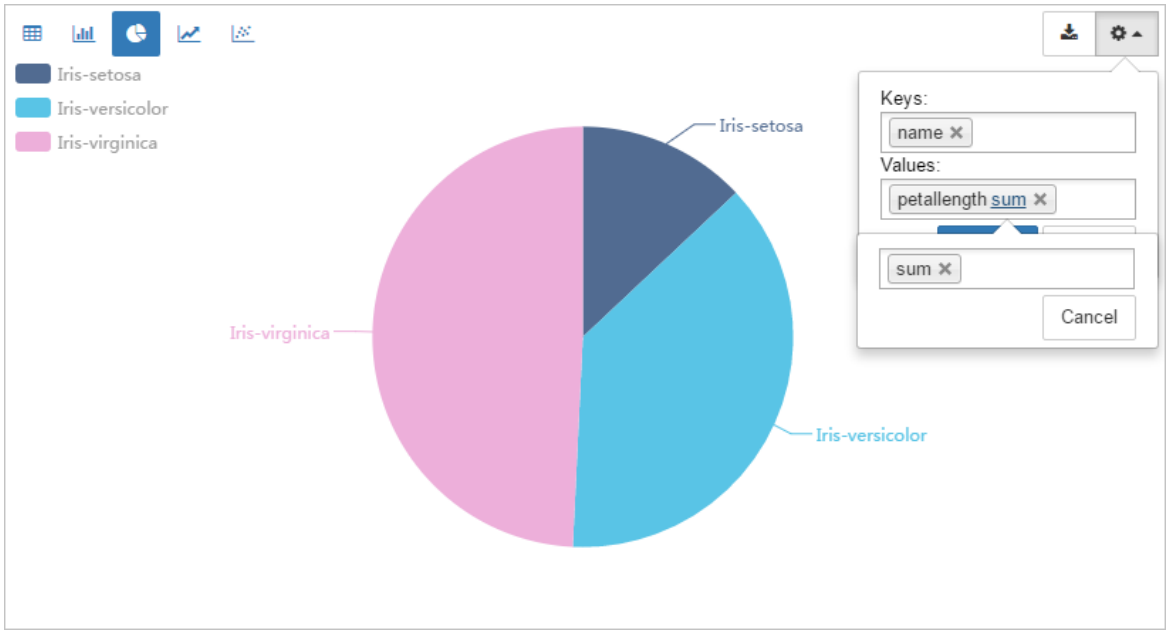
3. You can click the Settings icon in the upper-right corner of a graph to modify the settings. For example, set Groups to name, X Axis to petallength, and Y Axis to petalwidth. The result is shown

in the following graph. The petallength-petalwidth settings display the data in a manner that is easy to understand.



For column charts and pie charts, you can select an aggregate function for the value fields. The default aggregate function for column charts is `sum`, and that for pie charts is `count`. You can click the function name next to the name of the value field to select another function.

For line charts, the values on the x-axis cannot be null. If any values are null, the graph may not be correctly displayed.

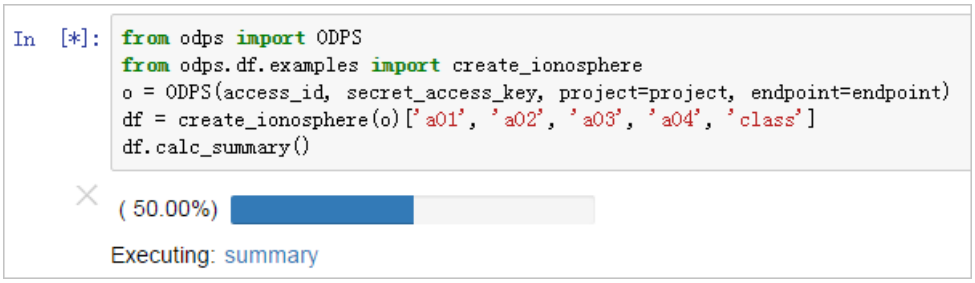


4. After you make the graph, click the **Download** icon to save it.

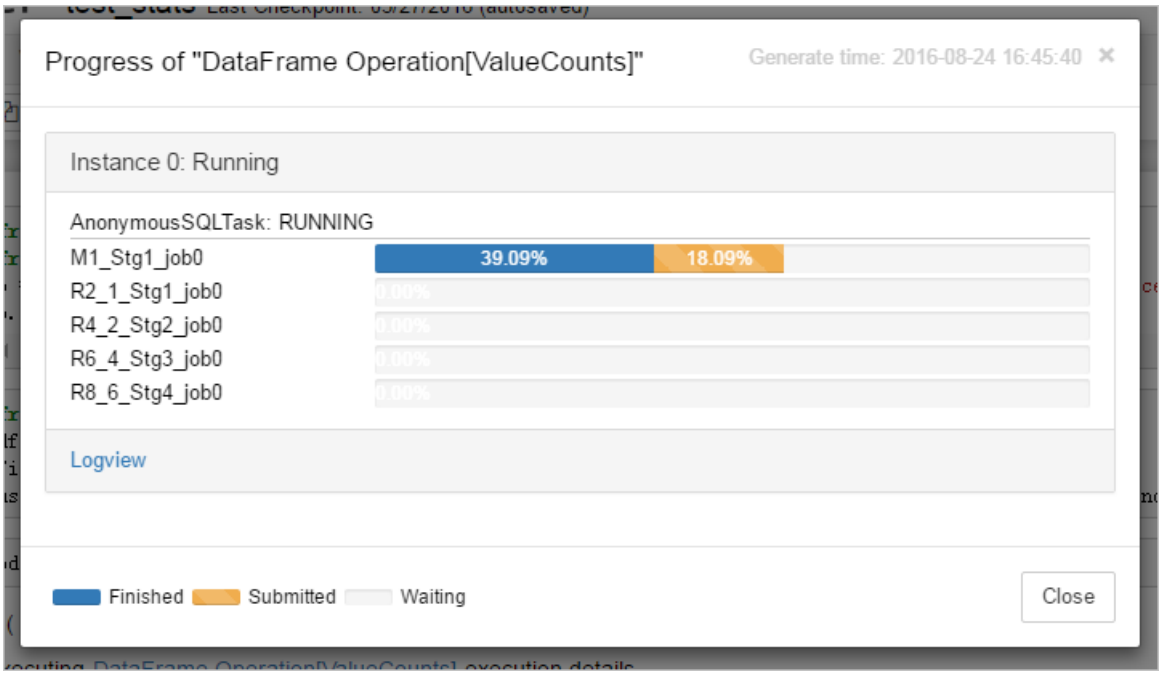
> **Note** To use this feature, you must install pandas and ipywidgets.

Progress display

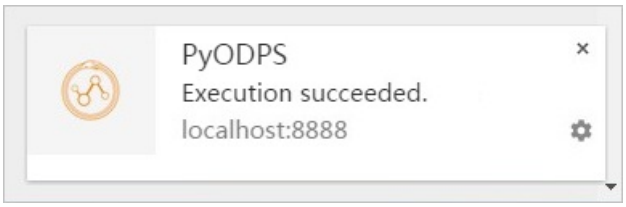
The execution of large jobs takes extended periods of time. PyODPS provides progress bars to show the execution progress. When DataFrame jobs, machine learning jobs, or SQL statements that start with `%sql` are executed in Jupyter Notebook, a list of these jobs and their overall progress are displayed.



If you click a job name, a dialog box that shows the progress of each task in the job appears.



After the execution is completed, a message that indicates whether the job succeeded appears.



4.1.8. API overview

This topic provides the links to PyODPS API documentation, which provides parameter descriptions and examples of each function:

- Definitions
- DataFrame Reference

4.1.9. Examples

4.1.9.1. Reference a third-party package in a PyODPS node

This topic describes how to reference a third-party package in a PyODPS node.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.

Procedure

1. Download the packages that are listed in the following table.

Package	File	Resource file
python-dateutil	python-dateutil-2.6.0.zip	python-dateutil.zip
pytz	pytz-2017.2.zip	pytz.zip
six	six-1.11.0.tar.gz	six.tar.gz
pandas	pandas-0.20.2-cp27-cp27m-manylinux1_x86_64.whl	pandas.zip
scipy	scipy-0.19.0-cp27-cp27m-manylinux1_x86_64.whl	scipy.zip
scikit-learn	scikit_learn-0.18.1-cp27-cp27m-manylinux1_x86_64.whl	sklearn.zip

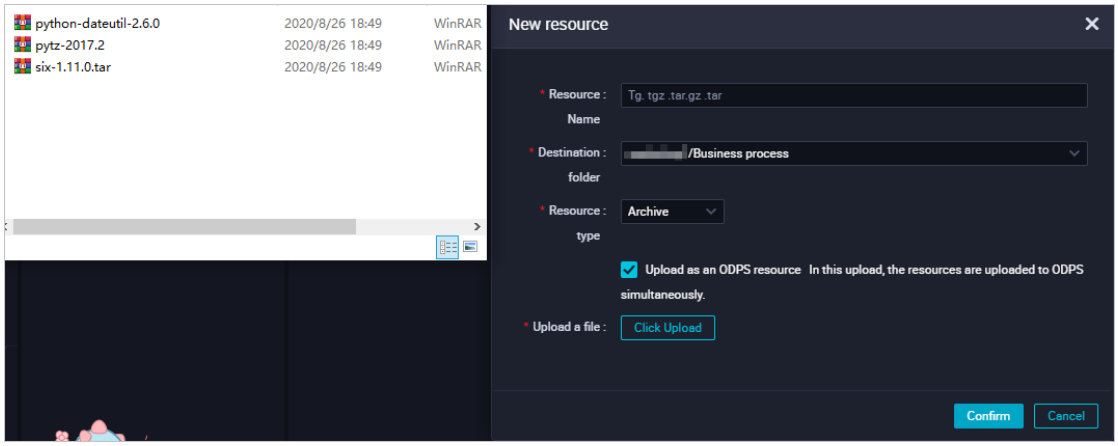
> **Note**
>
> - You need to manually change the file name extensions of the pandas, scipy, and scikit-learn packages from .whl to .zip.
> - Change the names of the resource files that you downloaded to be the same as those listed in the **Resource file** column of the preceding table.
> - Upload the preceding resource files as the resource files of the Archive storage class.

2. Log on to the DataWorks console.
3. Create a workflow.
   i. On the **DataStudio** page, right-click **Business Flow** and select **Create Workflow**.
   ii. In the **Create Workflow** dialog box, specify **Workflow Name** and click **Create**.
4. Create and commit resources.

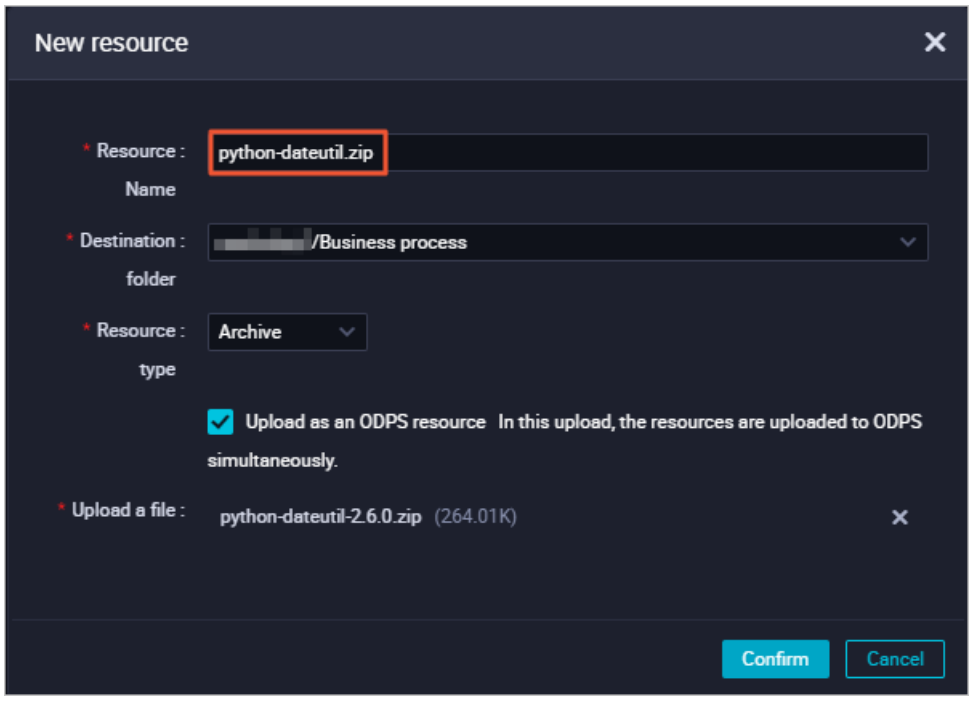
i. On the **DataStudio** page, move the pointer over the icon and choose **MaxCompute > Resources > Archive**.

You can also unfold Business Flow, right-click a workflow, and then choose **Create > MaxCompute > Resource > Archive**.

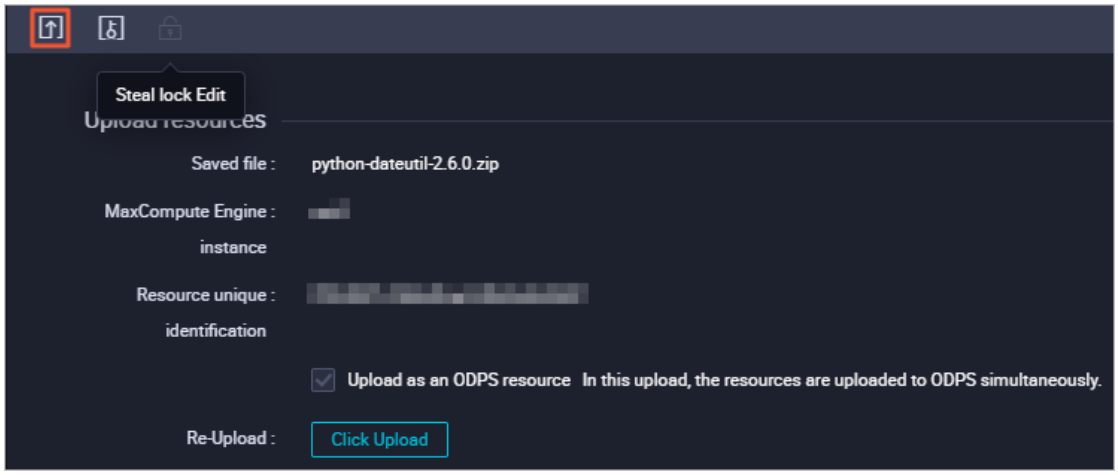
ii. In the **Create Resource** dialog box, click **Upload** and select the *python-dateutil-2.6.0.zip* file.



iii. Enter *python.dateutil.zip* in the Resource Name field and click **OK**.



iv. Click the icon to upload the resource file.



v. Repeat the preceding steps to create and commit the resource files named *pytz.zip*, *six.tar.gz*, *pandas.zip*, *sklearn.zip*, and *scipy.zip*.

5. Create a PyODPS node.

i. Right-click the workflow that you created and choose **Create > MaxCompute > PyODPS 2**.

ii. In the Create Node dialog box, specify **Node Name** and click **Commit**.

iii. On the tab of the created node, enter the code of the node in the code editor.

Sample code:

```
def test(x):
    from sklearn import datasets, svm
    from scipy import misc
    import numpy as np
    iris = datasets.load_iris()
    assert iris.data.shape == (150, 4)
    assert np.array_equal(np.unique(iris.target),  [0, 1, 2])
    clf = svm.LinearSVC()
    clf.fit(iris.data, iris.target)
    pred = clf.predict([[5.0, 3.6, 1.3, 0.25]])
    assert pred[0] == 0
    assert misc.face().shape is not None
    return x
from odps import options
hints = {
    'odps.isolation.session.enable': True
}
libraries = ['python-dateutil.zip', 'pytz.zip', 'six.tar.gz', 'pandas.zip', 'scipy.
zip', 'sklearn.zip']
iris = o.get_table('pyodps_iris').to_df()
print iris[:1].sepallength.map(test).execute(hints=hints, libraries=libraries)
```

6. Click the icon.

7. View the running result of the node on the **Run Log** tab.

```
Sql compiled:
CREATE TABLE tmp_pyodps_a3172c30_a0d7_4c88_bc39_434168263897 LIFECYCLE 1 AS
SELECT pyodps_udf_1576485276_94d9d978_af66_4e27_a874_e787022dfb3d(t1.`sepallength`) AS
`sepallength`
FROM WB_BestPractice_dev.`pyodps_iris` t1
LIMIT 1
Instance ID: 20191216083438175gcv6n4pr2
  Log view: http://logview.odps.aliyun.com/logview/?h=xxxxxx
   sepallength
0          5.1
```

Note For more information about best practices, see Use a PyODPS node to segment Chinese text based on Jieba.

4.1.9.2. Use a PyODPS node to query data based on specific criteria

This topic describes how to use a PyODPS node to query data based on specific criteria.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.
   ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

   In this example, execute the following statement to create the pyodps_iris table:

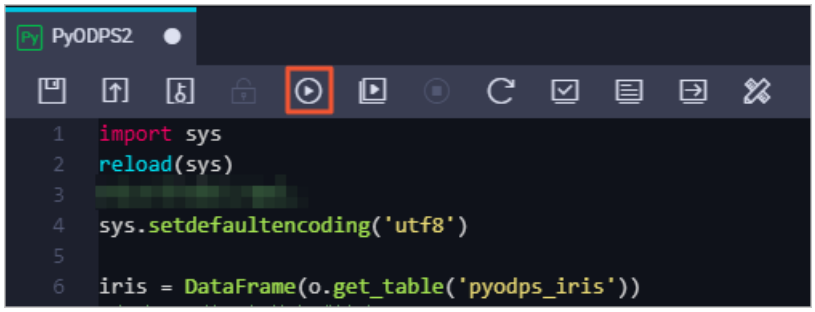
```
CREATE TABLE if not exists pyodps_iris
(
sepallength  DOUBLE comment 'sepal length (cm)',
sepalwidth   DOUBLE comment 'sepal width (cm)',
petallength  DOUBLE comment ''petal length (cm)',
petalwidth   DOUBLE comment 'petal width (cm)',
name         STRING comment 'type'
);
```

2.
3. In the left-side navigation pane, click **Workspaces**.
4. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.
5. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.

6. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
7. On the configuration tab, enter sample code in the code editor.

```
import sys
reload(sys)
# Specify the encoding format.
sys.setdefaultencoding('utf8')
iris = DataFrame(o.get_table('pyodps_iris'))
# Method 1: Return the result based on a conditional query statement.
with o.execute_sql('select * from pyodps_iris WHERE sepallength > 5 ').open_reader() as
reader4:
    print reader4.raw
    for record in reader4:
        print record["sepallength"],record["sepalwidth"],record["petallength"],record["
petalwidth"],record["name"]
# Method 2: Return the result based on the filter conditions of DataFrame.
print iris[iris.sepallength > 5].head(5)
# Method 3: Return the result by using the query method of PyODPS DataFrame.
print iris.query("(sepallength < 5) and (petallength > 1.5)").head(5)
```

8. Click the icon.



9. View the running result of the PyODPS 2 node on the **Run Log** tab.

```
Executing user script with PyODPS 0.8.0
"sepallength","sepalwidth","petallength","petalwidth","name"
5.1,3.5,1.4,0.2,"Iris-setosa"
5.4,3.9,1.7,0.4,"Iris-setosa"
5.4,3.7,1.5,0.2,"Iris-setosa"
5.8,4.0,1.2,0.2,"Iris-setosa"
5.7,4.4,1.5,0.4,"Iris-setosa"
5.4,3.9,1.3,0.4,"Iris-setosa"
5.1,3.5,1.4,0.3,"Iris-setosa"
5.7,3.8,1.7,0.3,"Iris-setosa"
5.1,3.8,1.5,0.3,"Iris-setosa"
5.4,3.4,1.7,0.2,"Iris-setosa"
5.1,3.7,1.5,0.4,"Iris-setosa"
5.1,3.3,1.7,0.5,"Iris-setosa"
5.2,3.5,1.5,0.2,"Iris-setosa"
5.2,3.4,1.4,0.2,"Iris-setosa"
5.4,3.4,1.5,0.4,"Iris-setosa"
5.2,4.1,1.5,0.1,"Iris-setosa"
5.5,4.2,1.4,0.2,"Iris-setosa"
5.5,3.5,1.3,0.2,"Iris-setosa"
5.1,3.4,1.5,0.2,"Iris-setosa"
```

```
5.1,3.8,1.9,0.4,"Iris-setosa"
5.1,3.8,1.6,0.2,"Iris-setosa"
5.3,3.7,1.5,0.2,"Iris-setosa"
7.0,3.2,4.7,1.4,"Iris-versicolor"
6.4,3.2,4.5,1.5,"Iris-versicolor"
6.9,3.1,4.9,1.5,"Iris-versicolor"
5.5,2.3,4.0,1.3,"Iris-versicolor"
6.5,2.8,4.6,1.5,"Iris-versicolor"
5.7,2.8,4.5,1.3,"Iris-versicolor"
6.3,3.3,4.7,1.6,"Iris-versicolor"
6.6,2.9,4.6,1.3,"Iris-versicolor"
5.2,2.7,3.9,1.4,"Iris-versicolor"
5.9,3.0,4.2,1.5,"Iris-versicolor"
6.0,2.2,4.0,1.0,"Iris-versicolor"
6.1,2.9,4.7,1.4,"Iris-versicolor"
5.6,2.9,3.6,1.3,"Iris-versicolor"
6.7,3.1,4.4,1.4,"Iris-versicolor"
5.6,3.0,4.5,1.5,"Iris-versicolor"
5.8,2.7,4.1,1.0,"Iris-versicolor"
6.2,2.2,4.5,1.5,"Iris-versicolor"
5.6,2.5,3.9,1.1,"Iris-versicolor"
5.9,3.2,4.8,1.8,"Iris-versicolor"
6.1,2.8,4.0,1.3,"Iris-versicolor"
6.3,2.5,4.9,1.5,"Iris-versicolor"
6.1,2.8,4.7,1.2,"Iris-versicolor"
6.4,2.9,4.3,1.3,"Iris-versicolor"
6.6,3.0,4.4,1.4,"Iris-versicolor"
6.8,2.8,4.8,1.4,"Iris-versicolor"
6.7,3.0,5.0,1.7,"Iris-versicolor"
6.0,2.9,4.5,1.5,"Iris-versicolor"
5.7,2.6,3.5,1.0,"Iris-versicolor"
5.5,2.4,3.8,1.1,"Iris-versicolor"
5.5,2.4,3.7,1.0,"Iris-versicolor"
5.8,2.7,3.9,1.2,"Iris-versicolor"
6.0,2.7,5.1,1.6,"Iris-versicolor"
5.4,3.0,4.5,1.5,"Iris-versicolor"
6.0,3.4,4.5,1.6,"Iris-versicolor"
6.7,3.1,4.7,1.5,"Iris-versicolor"
6.3,2.3,4.4,1.3,"Iris-versicolor"
5.6,3.0,4.1,1.3,"Iris-versicolor"
5.5,2.5,4.0,1.3,"Iris-versicolor"
5.5,2.6,4.4,1.2,"Iris-versicolor"
6.1,3.0,4.6,1.4,"Iris-versicolor"
5.8,2.6,4.0,1.2,"Iris-versicolor"
5.6,2.7,4.2,1.3,"Iris-versicolor"
5.7,3.0,4.2,1.2,"Iris-versicolor"
5.7,2.9,4.2,1.3,"Iris-versicolor"
6.2,2.9,4.3,1.3,"Iris-versicolor"
5.1,2.5,3.0,1.1,"Iris-versicolor"
5.7,2.8,4.1,1.3,"Iris-versicolor"
6.3,3.3,6.0,2.5,"Iris-virginica"
5.8,2.7,5.1,1.9,"Iris-virginica"
7.1,3.0,5.9,2.1,"Iris-virginica"
6.3,2.9,5.6,1.8,"Iris-virginica"
```

```
6.5,3.0,5.8,2.2,"Iris-virginica"
7.6,3.0,6.6,2.1,"Iris-virginica"
7.3,2.9,6.3,1.8,"Iris-virginica"
6.7,2.5,5.8,1.8,"Iris-virginica"
7.2,3.6,6.1,2.5,"Iris-virginica"
6.5,3.2,5.1,2.0,"Iris-virginica"
6.4,2.7,5.3,1.9,"Iris-virginica"
6.8,3.0,5.5,2.1,"Iris-virginica"
5.7,2.5,5.0,2.0,"Iris-virginica"
5.8,2.8,5.1,2.4,"Iris-virginica"
6.4,3.2,5.3,2.3,"Iris-virginica"
6.5,3.0,5.5,1.8,"Iris-virginica"
7.7,3.8,6.7,2.2,"Iris-virginica"
7.7,2.6,6.9,2.3,"Iris-virginica"
6.0,2.2,5.0,1.5,"Iris-virginica"
6.9,3.2,5.7,2.3,"Iris-virginica"
5.6,2.8,4.9,2.0,"Iris-virginica"
7.7,2.8,6.7,2.0,"Iris-virginica"
6.3,2.7,4.9,1.8,"Iris-virginica"
6.7,3.3,5.7,2.1,"Iris-virginica"
7.2,3.2,6.0,1.8,"Iris-virginica"
6.2,2.8,4.8,1.8,"Iris-virginica"
6.1,3.0,4.9,1.8,"Iris-virginica"
6.4,2.8,5.6,2.1,"Iris-virginica"
7.2,3.0,5.8,1.6,"Iris-virginica"
7.4,2.8,6.1,1.9,"Iris-virginica"
7.9,3.8,6.4,2.0,"Iris-virginica"
6.4,2.8,5.6,2.2,"Iris-virginica"
6.3,2.8,5.1,1.5,"Iris-virginica"
6.1,2.6,5.6,1.4,"Iris-virginica"
7.7,3.0,6.1,2.3,"Iris-virginica"
6.3,3.4,5.6,2.4,"Iris-virginica"
6.4,3.1,5.5,1.8,"Iris-virginica"
6.0,3.0,4.8,1.8,"Iris-virginica"
6.9,3.1,5.4,2.1,"Iris-virginica"
6.7,3.1,5.6,2.4,"Iris-virginica"
6.9,3.1,5.1,2.3,"Iris-virginica"
5.8,2.7,5.1,1.9,"Iris-virginica"
6.8,3.2,5.9,2.3,"Iris-virginica"
6.7,3.3,5.7,2.5,"Iris-virginica"
6.7,3.0,5.2,2.3,"Iris-virginica"
6.3,2.5,5.0,1.9,"Iris-virginica"
6.5,3.0,5.2,2.0,"Iris-virginica"
6.2,3.4,5.4,2.3,"Iris-virginica"
5.9,3.0,5.1,1.8,"Iris-virginica"
5.1 3.5 1.4 0.2 Iris-setosa
5.4 3.9 1.7 0.4 Iris-setosa
5.4 3.7 1.5 0.2 Iris-setosa
5.8 4.0 1.2 0.2 Iris-setosa
5.7 4.4 1.5 0.4 Iris-setosa
5.4 3.9 1.3 0.4 Iris-setosa
5.1 3.5 1.4 0.3 Iris-setosa
5.7 3.8 1.7 0.3 Iris-setosa
5.1 3.8 1.5 0.3 Iris-setosa
```

```
5.4 3.4 1.7 0.2 Iris-setosa
5.1 3.7 1.5 0.4 Iris-setosa
5.1 3.3 1.7 0.5 Iris-setosa
5.2 3.5 1.5 0.2 Iris-setosa
5.2 3.4 1.4 0.2 Iris-setosa
5.4 3.4 1.5 0.4 Iris-setosa
5.2 4.1 1.5 0.1 Iris-setosa
5.5 4.2 1.4 0.2 Iris-setosa
5.5 3.5 1.3 0.2 Iris-setosa
5.1 3.4 1.5 0.2 Iris-setosa
5.1 3.8 1.9 0.4 Iris-setosa
5.1 3.8 1.6 0.2 Iris-setosa
5.3 3.7 1.5 0.2 Iris-setosa
7.0 3.2 4.7 1.4 Iris-versicolor
6.4 3.2 4.5 1.5 Iris-versicolor
6.9 3.1 4.9 1.5 Iris-versicolor
5.5 2.3 4.0 1.3 Iris-versicolor
6.5 2.8 4.6 1.5 Iris-versicolor
5.7 2.8 4.5 1.3 Iris-versicolor
6.3 3.3 4.7 1.6 Iris-versicolor
6.6 2.9 4.6 1.3 Iris-versicolor
5.2 2.7 3.9 1.4 Iris-versicolor
5.9 3.0 4.2 1.5 Iris-versicolor
6.0 2.2 4.0 1.0 Iris-versicolor
6.1 2.9 4.7 1.4 Iris-versicolor
5.6 2.9 3.6 1.3 Iris-versicolor
6.7 3.1 4.4 1.4 Iris-versicolor
5.6 3.0 4.5 1.5 Iris-versicolor
5.8 2.7 4.1 1.0 Iris-versicolor
6.2 2.2 4.5 1.5 Iris-versicolor
5.6 2.5 3.9 1.1 Iris-versicolor
5.9 3.2 4.8 1.8 Iris-versicolor
6.1 2.8 4.0 1.3 Iris-versicolor
6.3 2.5 4.9 1.5 Iris-versicolor
6.1 2.8 4.7 1.2 Iris-versicolor
6.4 2.9 4.3 1.3 Iris-versicolor
6.6 3.0 4.4 1.4 Iris-versicolor
6.8 2.8 4.8 1.4 Iris-versicolor
6.7 3.0 5.0 1.7 Iris-versicolor
6.0 2.9 4.5 1.5 Iris-versicolor
5.7 2.6 3.5 1.0 Iris-versicolor
5.5 2.4 3.8 1.1 Iris-versicolor
5.5 2.4 3.7 1.0 Iris-versicolor
5.8 2.7 3.9 1.2 Iris-versicolor
6.0 2.7 5.1 1.6 Iris-versicolor
5.4 3.0 4.5 1.5 Iris-versicolor
6.0 3.4 4.5 1.6 Iris-versicolor
6.7 3.1 4.7 1.5 Iris-versicolor
6.3 2.3 4.4 1.3 Iris-versicolor
5.6 3.0 4.1 1.3 Iris-versicolor
5.5 2.5 4.0 1.3 Iris-versicolor
5.5 2.6 4.4 1.2 Iris-versicolor
6.1 3.0 4.6 1.4 Iris-versicolor
5.8 2.6 4.0 1.2 Iris-versicolor
```

```
5.6 2.7 4.2 1.3 Iris-versicolor
5.7 3.0 4.2 1.2 Iris-versicolor
5.7 2.9 4.2 1.3 Iris-versicolor
6.2 2.9 4.3 1.3 Iris-versicolor
5.1 2.5 3.0 1.1 Iris-versicolor
5.7 2.8 4.1 1.3 Iris-versicolor
6.3 3.3 6.0 2.5 Iris-virginica
5.8 2.7 5.1 1.9 Iris-virginica
7.1 3.0 5.9 2.1 Iris-virginica
6.3 2.9 5.6 1.8 Iris-virginica
6.5 3.0 5.8 2.2 Iris-virginica
7.6 3.0 6.6 2.1 Iris-virginica
7.3 2.9 6.3 1.8 Iris-virginica
6.7 2.5 5.8 1.8 Iris-virginica
7.2 3.6 6.1 2.5 Iris-virginica
6.5 3.2 5.1 2.0 Iris-virginica
6.4 2.7 5.3 1.9 Iris-virginica
6.8 3.0 5.5 2.1 Iris-virginica
5.7 2.5 5.0 2.0 Iris-virginica
5.8 2.8 5.1 2.4 Iris-virginica
6.4 3.2 5.3 2.3 Iris-virginica
6.5 3.0 5.5 1.8 Iris-virginica
7.7 3.8 6.7 2.2 Iris-virginica
7.7 2.6 6.9 2.3 Iris-virginica
6.0 2.2 5.0 1.5 Iris-virginica
6.9 3.2 5.7 2.3 Iris-virginica
5.6 2.8 4.9 2.0 Iris-virginica
7.7 2.8 6.7 2.0 Iris-virginica
6.3 2.7 4.9 1.8 Iris-virginica
6.7 3.3 5.7 2.1 Iris-virginica
7.2 3.2 6.0 1.8 Iris-virginica
6.2 2.8 4.8 1.8 Iris-virginica
6.1 3.0 4.9 1.8 Iris-virginica
6.4 2.8 5.6 2.1 Iris-virginica
7.2 3.0 5.8 1.6 Iris-virginica
7.4 2.8 6.1 1.9 Iris-virginica
7.9 3.8 6.4 2.0 Iris-virginica
6.4 2.8 5.6 2.2 Iris-virginica
6.3 2.8 5.1 1.5 Iris-virginica
6.1 2.6 5.6 1.4 Iris-virginica
7.7 3.0 6.1 2.3 Iris-virginica
6.3 3.4 5.6 2.4 Iris-virginica
6.4 3.1 5.5 1.8 Iris-virginica
6.0 3.0 4.8 1.8 Iris-virginica
6.9 3.1 5.4 2.1 Iris-virginica
6.7 3.1 5.6 2.4 Iris-virginica
6.9 3.1 5.1 2.3 Iris-virginica
5.8 2.7 5.1 1.9 Iris-virginica
6.8 3.2 5.9 2.3 Iris-virginica
6.7 3.3 5.7 2.5 Iris-virginica
6.7 3.0 5.2 2.3 Iris-virginica
6.3 2.5 5.0 1.9 Iris-virginica
6.5 3.0 5.2 2.0 Iris-virginica
6.2 3.4 5.4 2.3 Iris-virginica
5.9 3.0 5.1 1.8 Iris-virginica
```

```
5.9 3.0 5.1 1.8 Iris-virginica
Sql compiled:
CREATE TABLE tmp_pyodps_xxxx LIFECYCLE 1 AS
SELECT *
FROM xxx.`pyodps_iris` t1
WHERE t1.`sepallength` > 5
Instance ID: 2019xxxx
  Log view: http://logview.odps.aliyun.com/logview/?h=http://service.cn.maxcompute.aliyun.com/api&p=xxxx&i=2019xxxx&token=xxxx
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          5.4         3.9          1.7         0.4  Iris-setosa
2          5.4         3.7          1.5         0.2  Iris-setosa
3          5.8         4.0          1.2         0.2  Iris-setosa
4          5.7         4.4          1.5         0.4  Iris-setosa
Sql compiled:
CREATE TABLE tmp_pyodps_xxxx LIFECYCLE 1 AS
SELECT *
FROM xxxx.`pyodps_iris` t1
WHERE (t1.`sepallength` < 5) AND (t1.`petallength` > 1.5)
Instance ID: 2019xxxx
  Log view: http://logview.odps.aliyun.com/logview/?h=http://service.cn.maxcompute.aliyun.com/api&p=xxxx&i=2019xxxx&token=xxxx
   sepallength  sepalwidth  petallength  petalwidth            name
0          4.8         3.4          1.6         0.2     Iris-setosa
1          4.8         3.4          1.9         0.2     Iris-setosa
2          4.7         3.2          1.6         0.2     Iris-setosa
3          4.8         3.1          1.6         0.2     Iris-setosa
4          4.9         2.4          3.3         1.0  Iris-versicolor
```

4.1.9.3. Use a PyODPS node to pass parameters

This topic describes how to use a PyODPS node to pass parameters.

Prerequisites

The following operations are completed:

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Prepare test data.

i. Create a partitioned table and a source table, and import data to the source table. For more information, see Create tables and import data.

   In this example, use the following table creation statements and source data.

   - Execute the following statement to create a partitioned table named *user_detail*:

```
create table if not exists user_detail
(
userid    BIGINT comment 'user ID',
job       STRING comment 'job type',
education STRING comment 'education level'
) comment 'user information table'
partitioned by (dt STRING comment 'date',region STRING comment 'region');
```

   - Execute the following statement to create a source table named *user_detail_ods*:

```
create table if not exists user_detail_ods
(
  userid    BIGINT comment 'user ID',
  job       STRING comment 'job type',
  education STRING comment 'education level',
  dt STRING comment 'date',
  region STRING comment 'region'
);
```

   - Create a source data file named *user_detail.txt* and save the following data to the file. Import the data to the *user_detail_ods* table.

```
0001,Internet,bachelor,20190715,beijing
0002,education,junior college,20190716,beijing
0003,finance,master,20190715,shandong
0004,Internet,master,20190715,beijing
```

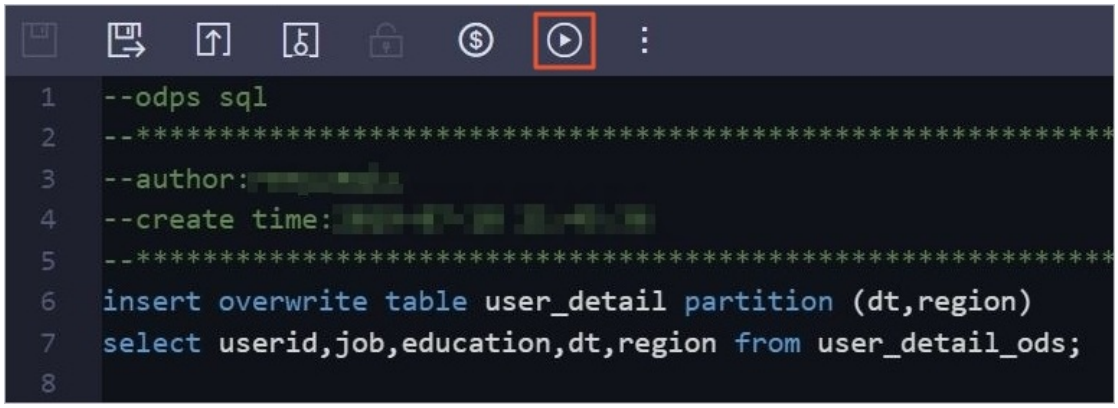
ii. Log on to the DataWorks console and click Workspaces in the left-side navigation pane. On the Workspaces page, find the target workspace and click Data Analytics in the Actions column. On the Data Development tab, right-click the target workflow in the Business process section and choose **New > MaxCompute > ODPS SQL**.

iii. In the New node dialog box, set the **Node name** parameter and click **Submit**.

iv. On the configuration tab of the ODPS SQL node, enter the following code in the code editor:

```
insert overwrite table user_detail partition (dt,region)
select userid,job,education,dt,region from user_detail_ods;
```

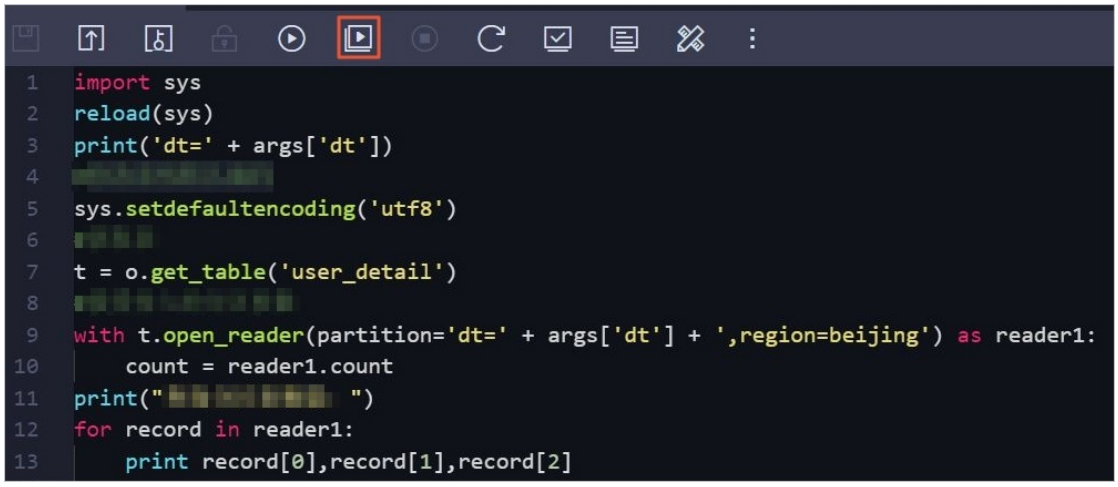
v. Click the Run icon in the toolbar to insert the data from the **user_detail_ods** table to the *user_detail* table.



2. Use a PyODPS node to pass parameters.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
   iv. On the Data Development tab, right-click the target workflow in the Business process section and choose **New > MaxCompute > PyODPS 2**.
   v. In the New node dialog box, set the **Node name** parameter and click **Submit**.
   vi. On the configuration tab of the PyODPS 2 node, enter the following code in the code editor:

```
import sys
reload(sys)
print('dt=' + args['dt'])
# Set UTF-8 as the default encoding format.
sys.setdefaultencoding('utf8')
# Obtain the user_detail table.
t = o.get_table('user_detail')
# Receive the partition field that is passed.
with t.open_reader(partition='dt=' + args['dt'] + ',region=beijing') as reader1:
    count = reader1.count
print("Query data in the partitioned table:")
for record in reader1:
    print record[0],record[1],record[2]
```

vii. Click **Advanced run (run with parameters)** in the toolbar.

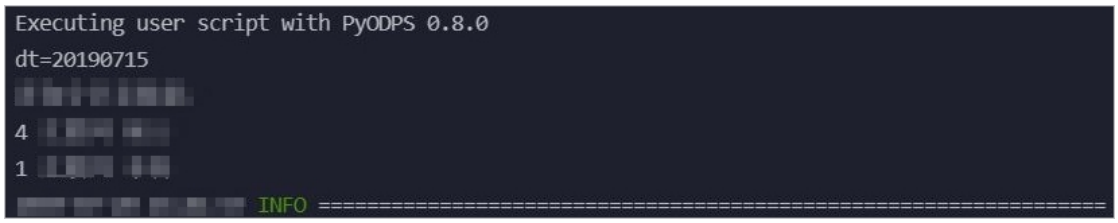


viii. In the **Parameters** dialog box, set the parameters and click **Confirm**.

Parameter description:

- **Scheduling Resource Group**: Set this parameter to **Common scheduler resource group**.
- **dt**: Set this parameter to dt=20190715.

ix. View the running result of the PyODPS 2 node on the **Run Log** tab.



4.1.9.4. Use a PyODPS node to read data from a partitioned table

This topic describes how to use a PyODPS node to read data from a partitioned table.

Prerequisites

The following operations are completed:

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Prepare test data.

i. Create a partitioned table and a source table, and import data to the source table. For more information, see Create tables and import data.

   In this example, use the following table creation statements and source data.

   - Execute the following statement to create a partitioned table named *user_detail*:

```
create table if not exists user_detail
(
userid    BIGINT comment 'user ID',
job       STRING comment 'job type',
education STRING comment 'education level'
) comment 'user information table'
partitioned by (dt STRING comment 'date',region STRING comment 'region');
```

   - Execute the following statement to create a source table named *user_detail_ods*:

```
create table if not exists user_detail_ods
(
  userid    BIGINT comment 'user ID',
  job       STRING comment 'job type',
  education STRING comment 'education level',
  dt STRING comment 'date',
  region STRING comment 'region'
);
```

   - Create a source data file named *user_detail.txt* and save the following data to the file. Import the data to the *user_detail_ods* table.

```
0001,Internet,bachelor,20190715,beijing
0002,education,junior college,20190716,beijing
0003,finance,master,20190715,shandong
0004,Internet,master,20190715,beijing
```

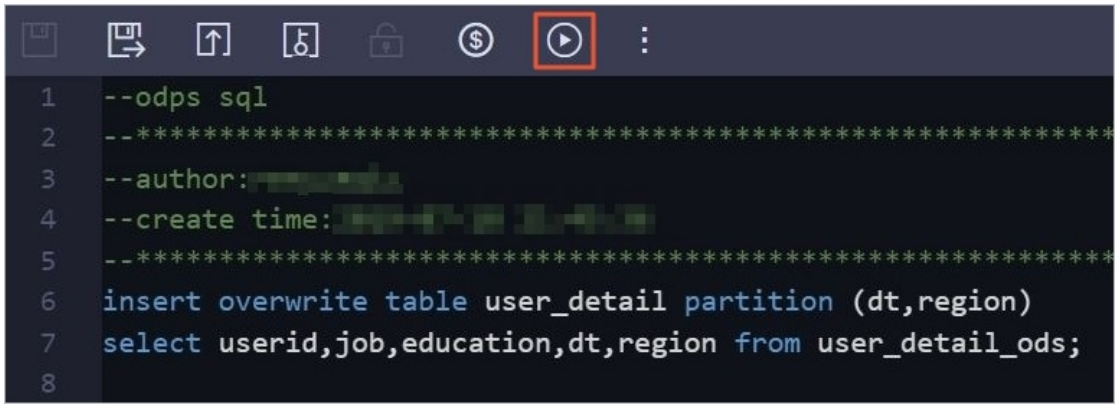
ii. Log on to the DataWorks console and click Workspaces in the left-side navigation pane. On the Workspaces page, find the target workspace and click Data Analytics in the Actions column. On the Data Development tab, right-click the target workflow in the Business process section and choose **New > MaxCompute > ODPS SQL**.

iii. In the New node dialog box, set the **Node name** parameter and click **Submit**.

iv. On the configuration tab of the ODPS SQL node, enter the following code in the code editor:

```
insert overwrite table user_detail partition (dt,region)
select userid,job,education,dt,region from user_detail_ods;
```

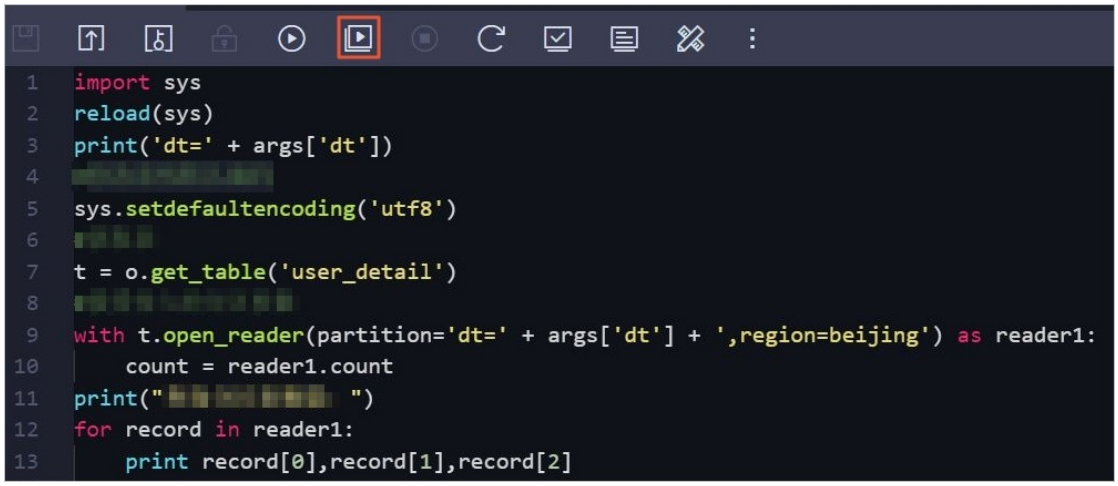
v. Click the Run icon in the toolbar to insert the data from the **user_detail_ods** table to the *user_detail* table.



2. Use a PyODPS node to read data from a partitioned table.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
   iv. On the Data Development tab, right-click the target workflow in the Business process section and choose **New > MaxCompute > PyODPS 2**.
   v. In the New node dialog box, set the **Node name** parameter and click **Submit**.
   vi. On the configuration tab of the PyODPS 2 node, enter the following code in the code editor:

```
import sys
from odps import ODPS
reload(sys)
print('dt=' + args['dt'])
# Set UTF-8 as the default encoding format.
sys.setdefaultencoding('utf8')
# Obtain the partitioned table.
t = o.get_table('user_detail')
# Check whether the specified partition exists.
print t.exist_partition('dt=20190715,region=beijing')
# View all partitions in the partitioned table.
for partition in t.partitions:
    print partition.name
# You can use one of the following methods to query data in the partitioned table:
# Method 1
with t.open_reader(partition='dt=20190715,region=beijing') as reader1:
    count = reader1.count
print("Query data in the partitioned table by using Method 1:")
for record in reader1:
    print record[0],record[1],record[2]
# Method 2
print("Query data in the partitioned table by using Method 2:")
reader2 = t.open_reader(partition='dt=20190715,region=beijing')
for record in reader2:
    print record["userid"],record["job"],record["education"]
# Method 3
print("Query data in the partitioned table by using Method 3:")
for record in o.read_table('user_detail', partition='dt=20190715,region=beijing'):
    print record["userid"],record["job"],record["education"]
```

vii. Click **Advanced run (run with parameters)** in the toolbar.

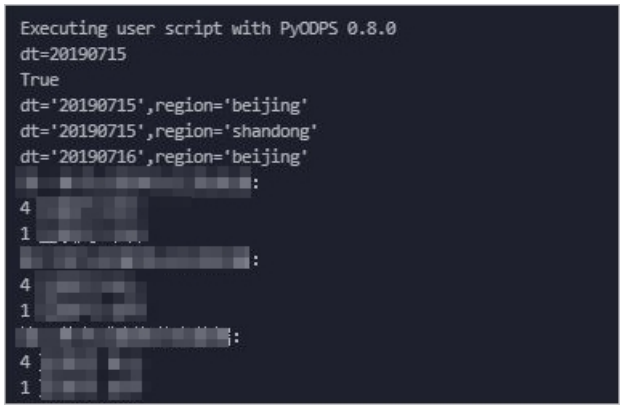


viii. In the **Parameters** dialog box, set the parameters and click **Confirm**.

Parameter description:

- **Scheduling Resource Group**: Set this parameter to **Common scheduler resource group**.
- **dt**: Set this parameter to dt=20190715.

ix. View the running result of the PyODPS 2 node on the **Run Log** tab.



4.1.9.5. Use a PyODPS node to read data from the level-1 partition of the specified table

This topic describes how to use a PyODPS node to read data from the level-1 partition of the specified table.

Prerequisites

The following operations are completed:

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Prepare test data.

i. Create a partitioned table and a source table, and import data to the source table. For more information, see Create tables and import data.

In this example, use the following table creation statements and source data.

- Execute the following statement to create a partitioned table named *user_detail*:

```
create table if not exists user_detail
(
userid    BIGINT comment 'user ID',
job       STRING comment 'job type',
education STRING comment 'education level'
) comment 'user information table'
partitioned by (dt STRING comment 'date',region STRING comment 'region');
```

- Execute the following statement to create a source table named *user_detail_ods*:

```
create table if not exists user_detail_ods
(
  userid    BIGINT comment 'user ID',
  job       STRING comment 'job type',
  education STRING comment 'education level',
  dt STRING comment 'date',
  region STRING comment 'region'
);
```

- Create a source data file named *user_detail.txt* and save the following data to the file. Import the data to the *user_detail_ods* table.

```
0001,Internet,bachelor,20190715,beijing
0002,education,junior college,20190716,beijing
0003,finance,master,20190715,shandong
0004,Internet,master,20190715,beijing
```

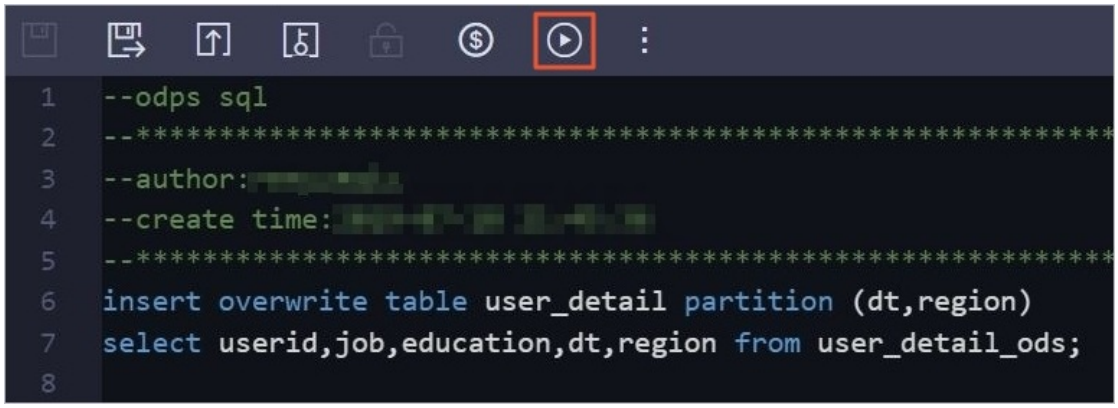
ii. Log on to the DataWorks console and click Workspaces in the left-side navigation pane. On the Workspaces page, find the target workspace and click Data Analytics in the Actions column. On the Data Development tab, right-click the target workflow in the Business process section and choose **New > MaxCompute > ODPS SQL**.

iii. In the New node dialog box, set the **Node name** parameter and click **Submit**.

iv. On the configuration tab of the ODPS SQL node, enter the following code in the code editor:

```
insert overwrite table user_detail partition (dt,region)
select userid,job,education,dt,region from user_detail_ods;
```

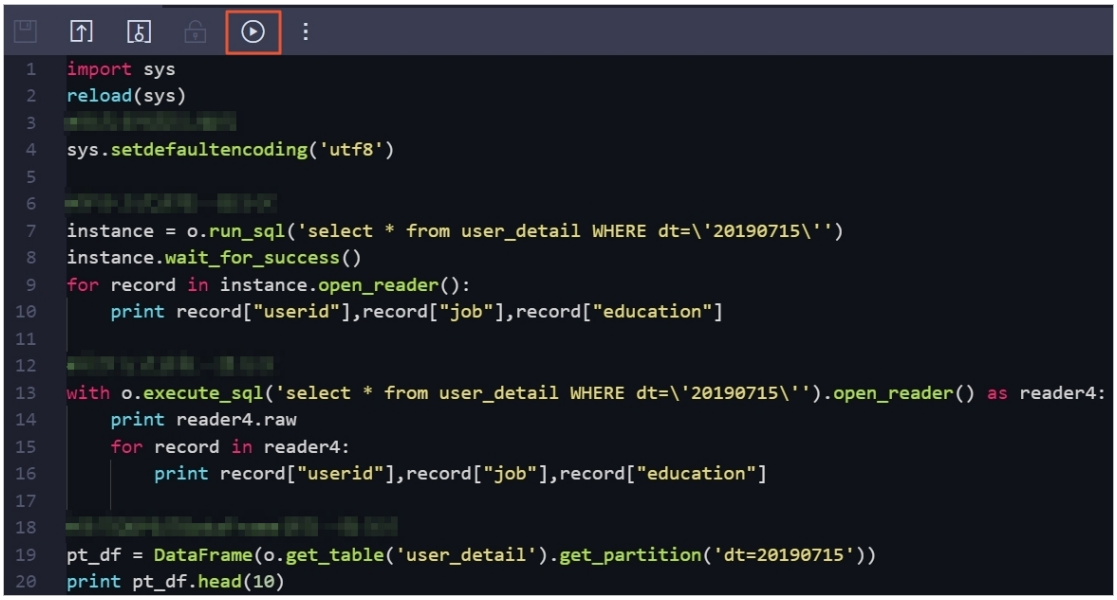
v. Click the Run icon in the toolbar to insert the data from the **user_detail_ods** table to the *user_detail* table.



2. Use a PyODPS node to read data from the level-1 partition of the user_detail table.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
   iv. On the Data Development tab, right-click the target workflow in the Business process section and choose **New > MaxCompute > PyODPS 2**.
   v. In the New node dialog box, set the **Node name** parameter and click **Submit**.
   vi. On the configuration tab of the PyODPS 2 node, enter the following code in the code editor:

```
import sys
reload(sys)
# Set UTF-8 as the default encoding format.
sys.setdefaultencoding('utf8')
# Read data from the level-1 partition in asynchronous mode.
instance = o.run_sql('select * from user_detail WHERE dt=\'20190715\'')
instance.wait_for_success()
for record in instance.open_reader():
    print record["userid"],record["job"],record["education"]
# Read data from the level-1 partition in synchronous mode.
with o.execute_sql('select * from user_detail WHERE dt=\'20190715\'').open_reader()
as reader4:
    print reader4.raw
    for record in reader4:
        print record["userid"],record["job"],record["education"]
# Use the PyODPS DataFrame to read data from the level-1 partition.
pt_df = DataFrame(o.get_table('user_detail').get_partition('dt=20190715'))
print pt_df.head(10)
```

vii. Click the Run icon in the toolbar.



```
import sys
reload(sys)
[redacted]
sys.setdefaultencoding('utf8')

[redacted]
instance = o.run_sql('select * from user_detail WHERE dt=\'20190715\'')
instance.wait_for_success()
for record in instance.open_reader():
    print record["userid"],record["job"],record["education"]

[redacted]
with o.execute_sql('select * from user_detail WHERE dt=\'20190715\'').open_reader() as reader4:
    print reader4.raw
    for record in reader4:
        print record["userid"],record["job"],record["education"]

[redacted]
pt_df = DataFrame(o.get_table('user_detail').get_partition('dt=20190715'))
print pt_df.head(10)
```

viii. View the running result of the PyODPS 2 node on the **Run Log** tab.

```
4 [redacted]
1 [redacted]
3 [redacted]

"userid","job","education","dt","region"
4,"[redacted]","[redacted]","20190715","beijing"
1,"[redacted]","[redacted]","20190715","beijing"
3,"[redacted]","[redacted]","20190715","shandong"
4 [redacted]
1 [redacted]
3 [redacted]
Try to fetch data from tunnel

   userid  job education
0       4 [redacted]
1       1 [redacted]
2       3 [redacted]
```

4.1.9.6. Use a PyODPS node to perform sequence operations

This topic describes how to use a PyODPS node to perform sequence operations.

Prerequisites

The following operations are completed:

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.
   ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

   In this example, execute the following statement to create the pyodps_iris table:

   ```
   CREATE TABLE if not exists pyodps_iris
   (
   sepallength  DOUBLE comment 'sepal length (cm)',
   sepalwidth   DOUBLE comment 'sepal width (cm)',
   petallength  DOUBLE comment ''petal length (cm)',
   petalwidth   DOUBLE comment 'petal width (cm)',
   name         STRING comment 'type'
   );
   ```

2. Log on to the DataWorks console.
3. In the left-side navigation pane, click **Workspaces**.
4. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.
5. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
6. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
7. On the configuration tab of the PyODPS 2 node, enter the code of the node in the code editor. In this example, enter the following code:

   ```
   from odps import DataFrame
   iris = DataFrame(o.get_table('pyodps_iris'))
   # Obtain data from the specified column.
   print iris.sepallength.head(5)
   print iris['sepallength'].head(5)
   # View the data type of the column.
   print iris.sepallength.dtype
   # Change the data type of the column.
   iris.sepallength.astype('int')
   # Group values in the specified column and obtain the maximum value in each group.
   print iris.groupby('name').sepallength.max().head(5)
   print iris.sepallength.max()
   # Rename the specified column.
   print iris.sepalwidth.rename('speal_width').head(5)
   # Obtain the sum of the values of the specified columns, place the sum in a new column,
   and rename the new column.
   print (iris.sepallength + iris.sepalwidth).rename('sum_sepal').head(5)
   ```

8. Click the Run icon in the toolbar.

9. View the running result of the PyODPS 2 node on the **Run Log** tab.

   In this example, the following information appears on the Run Log tab:

```
Executing user script with PyODPS 0.8.0
Try to fetch data from tunnel
   sepallength
0          4.9
1          4.7
2          4.6
3          5.0
4          5.4
Try to fetch data from tunnel
   sepallength
0          4.9
1          4.7
2          4.6
3          5.0
4          5.4
FLOAT64
Sql compiled:
CREATE TABLE tmp_pyodps_ed78e3ba_f13c_4a49_812d_2790d57c25dd LIFECYCLE 1 AS
SELECT MAX(t1.`sepallength`) AS `sepallength_max`
FROM data_service_fr.`pyodps_iris` t1
GROUP BY t1.`name`
   sepallength_max
0              5.8
1              7.0
2              7.9
Collection: ref_0
  odps.Table
    name: data_service_fr.`pyodps_iris`
    schema:
      sepallength           : double      # Sepal length (cm)
      sepalwidth            : double      # Sepal width (cm)
      petallength           : double      # Petal length (cm)
      petalwidth            : double      # Petal width (cm)
      name                  : string      # Type
max = Max[float64]
  sepallength = Column[sequence(float64)] 'sepallength' from collection ref_0
Try to fetch data from tunnel
   speal_width
0          3.0
1          3.2
2          3.1
3          3.6
4          3.9
Sql compiled:
CREATE TABLE tmp_pyodps_28120275_8d0f_4683_8318_302fa21459ac LIFECYCLE 1 AS
SELECT t1.`sepallength` + t1.`sepalwidth` AS `sum_sepal`
FROM data_service_fr.`pyodps_iris` t1
   sum_sepal
0        7.9
1        7.9
2        7.7
```

```
3        8.6
4        9.3
2019-08-13 10:48:13 INFO ================================================================
===
2019-08-13 10:48:13 INFO Exit code of the Shell command 0
2019-08-13 10:48:13 INFO --- Invocation of Shell command completed ---
2019-08-13 10:48:13 INFO Shell run successfully!
```

10. Use the same method to create and run another PyODPS node named PyExecute.

    On the configuration tab of the PyExecute node, enter the following code in the code editor:

```
from odps import options
from odps import DataFrame
# View the logview of the running instance.
options.verbose = True
iris = DataFrame(o.get_table('pyodps_iris'))
iris[iris.sepallength < 5].exclude('sepallength')[:5].execute()
my_logs = []
def my_loggers(x):
    my_logs.append(x)
options.verbose_log = my_loggers
iris[iris.sepallength < 5].exclude('sepallength')[:5].execute()
print(my_logs)
# Cache the intermediate collection.
cached = iris[iris.sepalwidth < 3.5].cache()
print cached.head(3)
# Concurrently execute methods in asynchronous mode.
from odps.df import Delay
delay = Delay() # Create a Delay object.
df = iris[iris.sepalwidth < 5].cache()  # The sum(), mean(), and max() methods depend o
n cache().
future1 = df.sepalwidth.sum().execute(delay=delay) # This method immediately returns a
future object without execution after it is called.
future2 = df.sepalwidth.mean().execute(delay=delay)
future3 = df.sepalwidth.max().execute(delay=delay)
delay.execute(n_parallel=3)
print future1.result()
print future2.result()
print future3.result()
```

    In this example, the following logs are recorded after the PyExecute node is run:

```
Executing user script with PyODPS 0.8.0
Sql compiled:
CREATE TABLE tmp_pyodps_4a204590_0510_4e9c_823b_5b837a437840 LIFECYCLE 1 AS
SELECT t1.`sepalwidth`, t1.`petallength`, t1.`petalwidth`, t1.`name`
FROM data_service_fr.`pyodps_iris` t1
WHERE t1.`sepallength` < 5
LIMIT 5
Instance ID: 20190813025233386g04djssa
  Log view: http://logview.odps.aliyun.com/logview/XXX
['Sql compiled:', 'CREATE TABLE tmp_pyodps_03b92c55_8442_4e61_8978_656495487b8a LIFECYC
LE 1 AS \nSELECT t1.`sepalwidth`, t1.`petallength`, t1.`petalwidth`, t1.`name` \nFROM d
ata_service_fr.`pyodps_iris` t1 \nWHERE t1.`sepallength` < 5 \nLIMIT 5', 'Instance ID:
20190813025236282gcsna5pr2', u'
Log view: http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&
XXX
   sepallength  sepalwidth  petallength  petalwidth         name
0          4.9         3.0          1.4         0.2  Iris-setosa
1          4.7         3.2          1.3         0.2  Iris-setosa
2          4.6         3.1          1.5         0.2  Iris-setosa
454.6
3.05100671141
4.4
2019-08-13 10:52:48 INFO ============================================================
===
2019-08-13 10:52:48 INFO Exit code of the Shell command 0
2019-08-13 10:52:48 INFO --- Invocation of Shell command completed ---
2019-08-13 10:52:48 INFO Shell run successfully!
2019-08-13 10:52:48 INFO Current task status: FINISH
```

4.1.9.7. Use a PyODPS node to aggregate data

This topic describes how to use a PyODPS node to aggregate data.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.

ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

In this example, execute the following statement to create the pyodps_iris table:

```
CREATE TABLE if not exists pyodps_iris
(
sepallength  DOUBLE comment 'sepal length (cm)',
sepalwidth   DOUBLE comment 'sepal width (cm)',
petallength  DOUBLE comment ''petal length (cm)',
petalwidth   DOUBLE comment 'petal width (cm)',
name         STRING comment 'type'
);
```

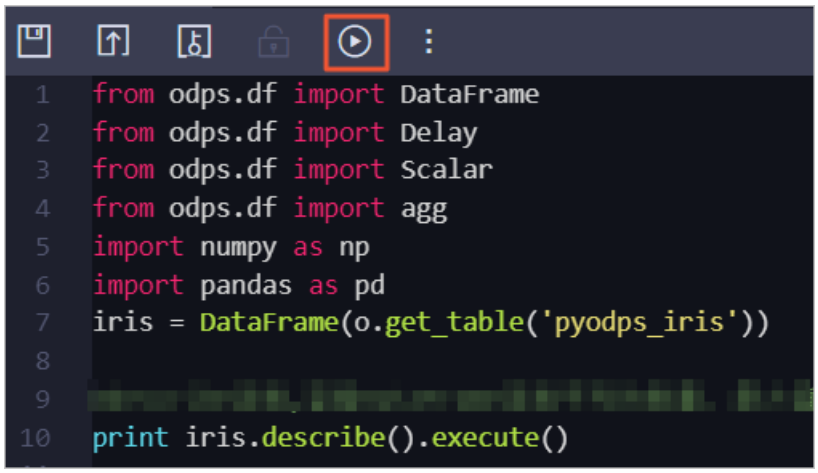
2. Log on to the DataWorks console.
3. In the left-side navigation pane, click **Workspaces**.
4. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
5. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
6. On the configuration tab of the PyODPS 2 node, enter the code of the node in the code editor.

In this example, enter the following code:

```
from odps.df import DataFrame
from odps.df import Delay
from odps.df import Scalar
from odps.df import agg
import numpy as np
import pandas as pd
iris = DataFrame(o.get_table('pyodps_iris'))
# Call the describe method to return the number, maximum value, minimum value, average
value, and standard deviation of the numeric columns in the PyODPS DataFrame.
print iris.describe().execute()
# Obtain the maximum value from the specified column.
print iris.sepallength.max().execute()
# Remove duplicate columns from the PyODPS DataFrame.
print iris.name.unique().cat(sep=',').execute()
# Calculate the total number of rows in the PyODPS DataFrame.
print iris.count().execute()
# Aggregate data by group.
print iris.groupby('name').agg(iris.sepallength.max(),smin=iris.sepallength.min()).head
()
print iris.groupby('name').agg(count=iris.name.count()).sort('count', ascending=False).
head(5)
print iris.groupby('name').petallength.sum().head()
print iris.groupby('name').agg(iris.petallength.notnull().sum()).head()
print iris.groupby(Scalar(1)).petallength.sum().head()
# Aggregate data in a single column.
class Agg(object):
    def buffer(self):
        return [0.0, 0]
    def __call__(self, buffer, val):
        buffer[0] += val
        buffer[1] += 1
    def merge(self, buffer, pbuffer):
```

```
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def getvalue(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] / buffer[1]
print iris.sepalwidth.agg(Agg).execute()
# Aggregate data in two columns.
class Agg2(object):
    def buffer(self):
        return [0.0, 0.0]
    def __call__(self, buffer, val1, val2):
        buffer[0] += val1
        buffer[1] += val2
    def merge(self, buffer, pbuffer):
        buffer[0] += pbuffer[0]
        buffer[1] += pbuffer[1]
    def getvalue(self, buffer):
        if buffer[1] == 0:
            return 0.0
        return buffer[0] - buffer[1]
to_agg = agg([iris.sepalwidth, iris.sepallength], Agg2, rtype='float')  # Call a user-d
efined aggregate function to aggregate data in two columns.
print iris.groupby('name').agg(val=to_agg).execute()
# Use HyperLogLog to approximate the number of distinct values.
df5 = DataFrame(pd.DataFrame({'a': np.random.randint(100000, size=100000)}))
print df5.a.hll_count().execute()
```

7. Click the Run icon in the toolbar.



8. View the running result of the PyODPS 2 node on the **Run Log** tab.

   In this example, the following information appears on the Run Log tab:

```
Executing user script with PyODPS 0.8.0
Sql compiled:
CREATE TABLE tmp_pyodps_28b83337_87f9_4b11_a084_f1896758d53b LIFECYCLE 1 AS
SELECT pyodps_udf_1567394092_9467b7ba_3121_4105_81dc_431e8e68be39(t3.`sepallength_count
`, t3.`sepallength_mean`, t3.`sepallength_std`, t3.`sepallength_min`, t3.`sepallength_q
uantile_25`, t3.`sepallength_quantile_50`, t3.`sepallength_quantile_75`, t3.`sepallengt
h_max`, t3.`sepalwidth_count`, t3.`sepalwidth_mean`, t3.`sepalwidth_std`, t3.`sepalwidt
h_min`, t3.`sepalwidth_quantile_25`, t3.`sepalwidth_quantile_50`, t3.`sepalwidth_quanti
```

```
x_max`, t3.`sepalwidth_quantile_25`, t3.`sepalwidth_quantile_50`, t3.`sepalwidth_quantil
le_75`, t3.`sepalwidth_max`, t3.`petallength_count`, t3.`petallength_mean`, t3.`petalle
ngth_std`, t3.`petallength_min`, t3.`petallength_quantile_25`, t3.`petallength_quantile
_50`, t3.`petallength_quantile_75`, t3.`petallength_max`, t3.`petalwidth_count`, t3.`pe
talwidth_mean`, t3.`petalwidth_std`, t3.`petalwidth_min`, t3.`petalwidth_quantile_25`,
t3.`petalwidth_quantile_50`, t3.`petalwidth_quantile_75`, t3.`petalwidth_max`) AS (`typ
e`, `sepallength`, `sepalwidth`, `petallength`, `petalwidth`)
FROM (
  SELECT COUNT(t2.`sepallength`) AS `sepallength_count`, AVG(t2.`sepallength`) AS `sepa
llength_mean`, STDDEV_SAMP(t2.`sepallength`) AS `sepallength_std`, MIN(t2.`sepallength`
) AS `sepallength_min`, PERCENTILE(t2.`sepallength`, 0.25) AS `sepallength_quantile_25`
, PERCENTILE(t2.`sepallength`, 0.5) AS `sepallength_quantile_50`, PERCENTILE(t2.`sepall
ength`, 0.75) AS `sepallength_quantile_75`, MAX(t2.`sepallength`) AS `sepallength_max`,
COUNT(t2.`sepalwidth`) AS `sepalwidth_count`, AVG(t2.`sepalwidth`) AS `sepalwidth_mean`
, STDDEV_SAMP(t2.`sepalwidth`) AS `sepalwidth_std`, MIN(t2.`sepalwidth`) AS `sepalwidth
_min`, PERCENTILE(t2.`sepalwidth`, 0.25) AS `sepalwidth_quantile_25`, PERCENTILE(t2.`se
palwidth`, 0.5) AS `sepalwidth_quantile_50`, PERCENTILE(t2.`sepalwidth`, 0.75) AS `sepa
lwidth_quantile_75`, MAX(t2.`sepalwidth`) AS `sepalwidth_max`, COUNT(t2.`petallength`)
AS `petallength_count`, AVG(t2.`petallength`) AS`petallength_mean`, STDDEV_SAMP(t2.`pet
allength`) AS `petallength_std`, MIN(t2.`petallength`) AS `petallength_min`, PERCENTILE
(t2.`petallength`, 0.25) AS `petallength_quantile_25`, PERCENTILE(t2.`petallength`, 0.5
) AS `petallength_quantile_50`, PERCENTILE(t2.`petallength`, 0.75) AS `petallength_quan
tile_75`, MAX(t2.`petallength`) AS `petallength_max`, COUNT(t2.`petalwidth`) AS `petalw
idth_count`, AVG(t2.`petalwidth`) AS `petalwidth_mean`, STDDEV_SAMP(t2.`petalwidth`) AS
`petalwidth_std`, MIN(t2.`petalwidth`) AS `petalwidth_min`, PERCENTILE(t2.`petalwidth`,
0.25) AS `petalwidth_quantile_25`, PERCENTILE(t2.`petalwidth`, 0.5) AS `petalwidth_quan
tile_50`, PERCENTILE(t2.`petalwidth`, 0.75) AS `petalwidth_quantile_75`, MAX(t2.`petalw
idth`) AS `petalwidth_max`
  FROM (
    SELECT t1.`sepallength`, t1.`sepalwidth`, t1.`petallength`, t1.`petalwidth`
    FROM DQC_0221_dev.`pyodps_iris` t1
  ) t2
) t3
          type  sepallength  sepalwidth  petallength  petalwidth
0        count   149.000000  149.000000   149.000000  149.000000
1         mean     5.848322    3.051007     3.774497    1.205369
2          std     0.828594    0.433499     1.759651    0.761292
3          min     4.300000    2.000000     1.000000    0.100000
4  quantile_25     5.000000    2.000000     1.000000    0.000000
5  quantile_50     5.000000    3.000000     4.000000    1.000000
6  quantile_75     6.000000    3.000000     5.000000    1.000000
7          max     7.900000    4.400000     6.900000    2.500000
Sql compiled:
SELECT MAX(t2.`sepallength`) AS `sepallength_max`
FROM (
  SELECT t1.`sepallength`
  FROM DQC_0221_dev.`pyodps_iris` t1
) t2
7.9
Sql compiled:
SELECT WM_CONCAT(DISTINCT ',', t2.`name`) AS `name_cat`
FROM (
  SELECT t1.`name`
  FROM DQC_0221_dev.`pyodps_iris` t1
) t2
```

```
Iris-setosa,Iris-versicolor,Iris-virginica
149
Sql compiled:
CREATE TABLE tmp_pyodps_4b00d671_82b1_4cd8_8af4_f11dbdac4570 LIFECYCLE 1 AS
SELECT t1.`name`, MAX(t1.`sepallength`) AS `sepallength_max`, MIN(t1.`sepallength`) AS
`smin`
FROM DQC_0221_dev.`pyodps_iris` t1
GROUP BY t1.`name`
              name  sepallength_max  smin
0      Iris-setosa              5.8   4.3
1  Iris-versicolor              7.0   4.9
2   Iris-virginica              7.9   4.9
Sql compiled:
CREATE TABLE tmp_pyodps_276c901f_7cf0_48ee_aaae_0d3092822b34 LIFECYCLE 1 AS
SELECT t1.`name`, COUNT(t1.`name`) AS `count`
FROM DQC_0221_dev.`pyodps_iris` t1
GROUP BY t1.`name`
ORDER BY count DESC
LIMIT 10000
              name  count
0  Iris-versicolor     50
1   Iris-virginica     50
2      Iris-setosa     49
Sql compiled:
CREATE TABLE tmp_pyodps_440d12ec_8496_4b87_b33f_b9624941e5bd LIFECYCLE 1 AS
SELECT SUM(t1.`petallength`) AS `petallength_sum`
FROM DQC_0221_dev.`pyodps_iris` t1
GROUP BY t1.`name`
Instance ID: 20190902031604255go6c472m
   petallength_sum
0             71.8
1            213.0
2            277.6
Sql compiled:
CREATE TABLE tmp_pyodps_61e62f5f_7b4f_4971_8c39_dcaccba37764 LIFECYCLE 1 AS
SELECT t1.`name`, SUM(IF(t1.`petallength` IS NOT NULL, 1, 0)) AS `petallength_sum`
FROM DQC_0221_dev.`pyodps_iris` t1
GROUP BY t1.`name`
Instance ID: 20190902031607441ghzyr392
              name  petallength_sum
0      Iris-setosa               49
1  Iris-versicolor               50
2   Iris-virginica               50
Sql compiled:
CREATE TABLE tmp_pyodps_daeffd50_88fe_4b4d_9db9_57810cfe71c3 LIFECYCLE 1 AS
SELECT SUM(t1.`petallength`) AS `petallength_sum`
FROM DQC_0221_dev.`pyodps_iris` t1
GROUP BY 1
   petallength_sum
0            562.4
Sql compiled:
SELECT pyodps_udf_1567394173_ef78183a_a4f5_4372_91dd_18d6ecc6f651(t2.`sepalwidth`) AS `
sepalwidth_aggregation`
FROM (
```

```
  SELECT t1.`sepalwidth`
  FROM DQC_0221_dev.`pyodps_iris` t1
) t2
Sql compiled:
CREATE TABLE tmp_pyodps_e312f110_64e7_465e_adae_fd053f125a90 LIFECYCLE 1 AS
SELECT t1.`name`, pyodps_udf_1567394220_49231d7a_faeb_4f2f_9de5_36c4aefaa29b(t1.`sepalw
idth`, t1.`sepallength`) AS `val`
FROM DQC_0221_dev.`pyodps_iris` t1
GROUP BY t1.`name`
              name    val
0      Iris-setosa  -77.8
1  Iris-versicolor -158.3
2   Iris-virginica -180.7
63021
```

4.1.9.8. Use a PyODPS node to perform column operations

This topic describes how to use a PyODPS node to perform column operations.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.
   ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

   In this example, execute the following statement to create the pyodps_iris table:

   ```
   CREATE TABLE if not exists pyodps_iris
   (
   sepallength  DOUBLE comment 'sepal length (cm)',
   sepalwidth   DOUBLE comment 'sepal width (cm)',
   petallength  DOUBLE comment ''petal length (cm)',
   petalwidth   DOUBLE comment 'petal width (cm)',
   name         STRING comment 'type'
   );
   ```

2. Log on to the DataWorks console.
3. In the left-side navigation pane, click **Workspaces**.
4. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.
5. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.

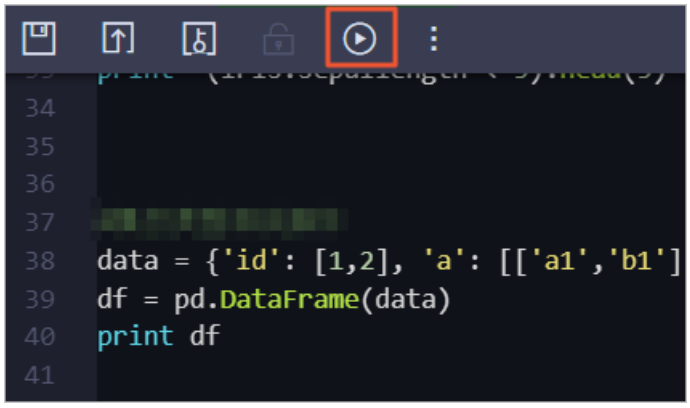
6. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
7. On the configuration tab of the PyODPS 2 node, enter the code of the node in the code editor.

   In this example, enter the following code:

```
from odps import DataFrame
import numpy as np
import pandas as pd
iris = DataFrame(o.get_table('pyodps_iris'))
# Check for NULL values.
print iris.sepallength.isnull().head(5)
# Perform logical judgment.
print (iris.sepallength > 5).ifelse('gt5','lte5').rename('cmp5').head(5)
# Perform judgment based on multiple conditions.
print iris.sepallength.switch(4.9,'eq4.9',5.0,'eq5.0',default='noeq').rename('equalness
').head(5)
from odps.df import switch
print switch(iris.sepallength == 4.9,'eq4.9',iris.sepallength == 5.0,'eq5.0',default='n
oeq').rename('equalness').head(5)
# Change some values in the specified column.
iris[iris.sepallength > 5,'cmp5'] = 'gt5'
iris[iris.sepallength <=5,'cmp5'] = 'lte5'
print iris.head(5)
# Perform mathematical calculation.
print (iris.sepallength * 10).log().head(5)
fields = [iris.sepallength,(iris.sepallength /2).rename('sepallength/2'),(iris.sepallen
gth ** 2).rename('Square of sepallength')]
print iris[fields].head(5)
print  (iris.sepallength < 5).head(5)
# Perform operations on collections.
data = {'id': [1,2], 'a': [['a1','b1'],['c1']], 'b': [{'a2': 0, 'b2': 1, 'c2': 2},{'d2'
: 3, 'e2': 4}]}
df = pd.DataFrame(data)
print df
df1 = DataFrame(df, unknown_as_string=True, as_type={'a': 'list<string>','b' : 'dict<st
ring,int64>'})
print df1.dtypes
print df1.head()
print df1[df1.id,df1.a[0],df1.b.len()].head()
print df1.a.explode().head()
print df1.a.explode(pos=True).head()
print df1.b.explode().head()
print df1.b.explode(['key','value']).head()
# Return the execution result of the explode method together with the specified columns
.
print df1[df1.id,df1.a.explode()].head()
#isin,notin,cut.
# Call the isin method to check whether elements in the specified sequence are in a spe
cific collection. The notin method can check whether elements in the specified sequence
is not in a specific collection.
print iris.sepallength.isin([4.9,5.1]).rename('sepallength').head()
# Call the cut method to specify ranges and return the range to which each element in t
he specified sequence belongs.
print iris.sepallength.cut(range(6),labels=['0-1','1-2','2-3','3-4','4-5']).rename('sep
allength_cut').head(5)
```

```
# You can use the include_under parameter to specify whether to return values that are
less than the specified upper limit. You can use the include_over parameter to specify
whether to return values that are greater than the specified lower limit.
labels = ['0-1', '1-2', '2-3', '3-4', '4-5', '5-']
iris.sepallength.cut(range(6), labels=labels, include_over=True).rename('sepallength_cu
t').head(5)
```

8. Click the Run icon in the toolbar.



9. View the running result of the PyODPS 2 node on the **Run Log** tab.

   In this example, the following information appears on the Run Log tab:

```
Executing user script with PyODPS 0.8.0
Sql compiled:
CREATE TABLE tmp_pyodps_32e39a9a_8ae6_4815_b465_4b9dfa3bf1b9 LIFECYCLE 1 AS
SELECT t1.`sepallength` IS NULL AS `sepallength`
FROM DQC_0221_dev.`pyodps_iris` t1
  sepallength
0       False
1       False
2       False
3       False
4       False
Sql compiled:
CREATE TABLE tmp_pyodps_d684aded_8ad8_4119_8558_1fddd774dd3f LIFECYCLE 1 AS
SELECT IF(t1.`sepallength` > 5, 'gt5', 'lte5') AS `cmp5`
FROM DQC_0221_dev.`pyodps_iris` t1
   cmp5
0  lte5
1  lte5
2  lte5
3  lte5
4   gt5
Sql compiled:
CREATE TABLE tmp_pyodps_cfc5bdb9_6d3e_4711_8acc_913b29b96c9a LIFECYCLE 1 AS
SELECT CASE t1.`sepallength` WHEN 4.9 THEN 'eq4.9' WHEN 5.0 THEN 'eq5.0' ELSE 'noeq' EN
D AS `equalness`
FROM DQC_0221_dev.`pyodps_iris` t1
  equalness
0     eq4.9
1      noeq
2      noeq
```

```
3     eq5.0
4      noeq
Sql compiled:
CREATE TABLE tmp_pyodps_d1fd70b4_2ad8_4dc9_838e_f566f7f5841e LIFECYCLE 1 AS
SELECT CASE WHEN t1.`sepallength` == 4.9 THEN 'eq4.9' WHEN t1.`sepallength` == 5.0 THEN
'eq5.0' ELSE 'noeq' END AS `equalness`
FROM DQC_0221_dev.`pyodps_iris` t1
  equalness
0     eq4.9
1      noeq
2      noeq
3     eq5.0
4      noeq
Sql compiled:
CREATE TABLE tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553 LIFECYCLE 1 AS
SELECT t1.`sepallength`, t1.`sepalwidth`, t1.`petallength`, t1.`petalwidth`, t1.`name`,
IF(t1.`sepallength` <= 5, 'lte5', IF(t1.`sepallength` > 5, 'gt5', CAST(NULL AS STRING))
) AS `cmp5`
FROM DQC_0221_dev.`pyodps_iris` t1
   sepallength  sepalwidth  petallength  petalwidth         name  cmp5
0          4.9         3.0          1.4         0.2  Iris-setosa  lte5
1          4.7         3.2          1.3         0.2  Iris-setosa  lte5
2          4.6         3.1          1.5         0.2  Iris-setosa  lte5
3          5.0         3.6          1.4         0.2  Iris-setosa  lte5
4          5.4         3.9          1.7         0.4  Iris-setosa   gt5
Sql compiled:
CREATE TABLE tmp_pyodps_4bfb0aa0_7ec5_4f35_9480_d36baf2b8079 LIFECYCLE 1 AS
SELECT LN(t1.`sepallength` * 10) AS `sepallength`
FROM DQC_0221_dev.`tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553` t1
   sepallength
0     3.891820
1     3.850148
2     3.828641
3     3.912023
4     3.988984
Sql compiled:
CREATE TABLE tmp_pyodps_deb6fa45_fbb1_41d2_8174_76d35eebea9c LIFECYCLE 1 AS
SELECT t1.`sepallength`, t1.`sepallength` / 2 AS `sepallength/2`, POW(t1.`sepallength`,
2) AS `Square of sepallength`
FROM DQC_0221_dev.`tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553` t1
   sepallength  sepallength/2  Square of sepallength
0          4.9           2.45                  24.01
1          4.7           2.35                  22.09
2          4.6           2.30                  21.16
3          5.0           2.50                  25.00
4          5.4           2.70                  29.16
Sql compiled:
CREATE TABLE tmp_pyodps_fa326546_c5e8_466f_915d_e783b7dcacdf LIFECYCLE 1 AS
SELECT t1.`sepallength` < 5 AS `sepallength`
FROM DQC_0221_dev.`tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553` t1
  sepallength
0        True
1        True
2        True
```

```
3       False
4       False
          a                                b  id
0  [a1, b1]  {u'c2': 2, u'a2': 0, u'b2': 1}   1
1      [c1]            {u'd2': 3, u'e2': 4}   2
odps.Schema {
  a   list<string>
  b   dict<string,int64>
  id  int64
}
          a                                b  id
0  [a1, b1]  {u'c2': 2, u'a2': 0, u'b2': 1}   1
1      [c1]            {u'd2': 3, u'e2': 4}   2
   id   a  b
0   1  a1  3
1   2  c1  2
    a
0  a1
1  b1
2  c1
   a_pos   a
0      0  a1
1      1  b1
2      0  c1
  b_key  b_value
0    c2        2
1    a2        0
2    b2        1
3    d2        3
4    e2        4
  key  value
0  c2      2
1  a2      0
2  b2      1
3  d2      3
4  e2      4
   id   a
0   1  a1
1   1  b1
2   2  c1
Sql compiled:
CREATE TABLE tmp_pyodps_fb28d9b9_fa45_4712_a564_dd751641286e LIFECYCLE 1 AS
SELECT t1.`sepallength` IN (4.9, 5.1) AS `sepallength`
FROM DQC_0221_dev.`tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553` t1
   sepallength
0         True
1        False
2        False
3        False
4        False
5        False
6        False
7        False
8         True
```

```
9        False
10       False
11       False
12       False
13       False
14       False
15       False
16        True
17       False
18        True
19       False
20        True
21       False
22        True
23       False
24       False
25       False
26       False
27       False
28       False
29       False
30       False
31       False
32       False
33        True
34       False
35       False
36        True
37       False
38        True
39       False
40       False
41       False
42       False
43        True
44       False
45        True
46       False
47       False
48       False
49       False
50       False
51       False
52       False
53       False
54       False
55       False
56        True
57       False
58       False
59       False
Sql compiled:
CREATE TABLE tmp_pyodps_1f85e8e1_a9f0_473f_ac0e_d526fdd80dc4 LIFECYCLE 1 AS
SELECT CASE WHEN (0 < t1.`sepallength`) AND (t1.`sepallength` <= 1) THEN '0-1' WHEN (1
< t1.`sepallength`) AND (t1.`sepallength` <= 2) THEN '1-2' WHEN (2 < t1.`sepallength`)
```

```
< t1. sepallength ) AND (t1. sepallength  <= 2) THEN  1-2  WHEN (2 < t1. sepallength )
AND (t1.`sepallength` <= 3) THEN '2-3' WHEN (3 < t1.`sepallength`) AND (t1.`sepallength
` <= 4) THEN '3-4' WHEN (4 < t1.`sepallength`) AND (t1.`sepallength` <= 5) THEN '4-5' E
ND AS `sepallength_cut`
FROM DQC_0221_dev.`tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553` t1
  sepallength_cut
0             4-5
1             4-5
2             4-5
3             4-5
4            None
Sql compiled:
CREATE TABLE tmp_pyodps_080a677e_904b_4edc_a961_160f8c664d7f LIFECYCLE 1 AS
SELECT CASE WHEN (0 < t1.`sepallength`) AND (t1.`sepallength` <= 1) THEN '0-1' WHEN (1
< t1.`sepallength`) AND (t1.`sepallength` <= 2) THEN '1-2' WHEN (2 < t1.`sepallength`)
AND (t1.`sepallength` <= 3) THEN '2-3' WHEN (3 < t1.`sepallength`) AND (t1.`sepallength
` <= 4) THEN '3-4' WHEN (4 < t1.`sepallength`) AND (t1.`sepallength` <= 5) THEN '4-5' W
HEN 5 < t1.`sepallength` THEN '5-' END AS `sepallength_cut`
FROM DQC_0221_dev.`tmp_pyodps_d9e5d64e_1f21_4ffb_993f_70d3f36b6553` t1
```

4.1.9.9. Use a PyODPS node to sort data

This topic describes how to use a PyODPS node to sort data.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.
   ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

   In this example, execute the following statement to create the pyodps_iris table:

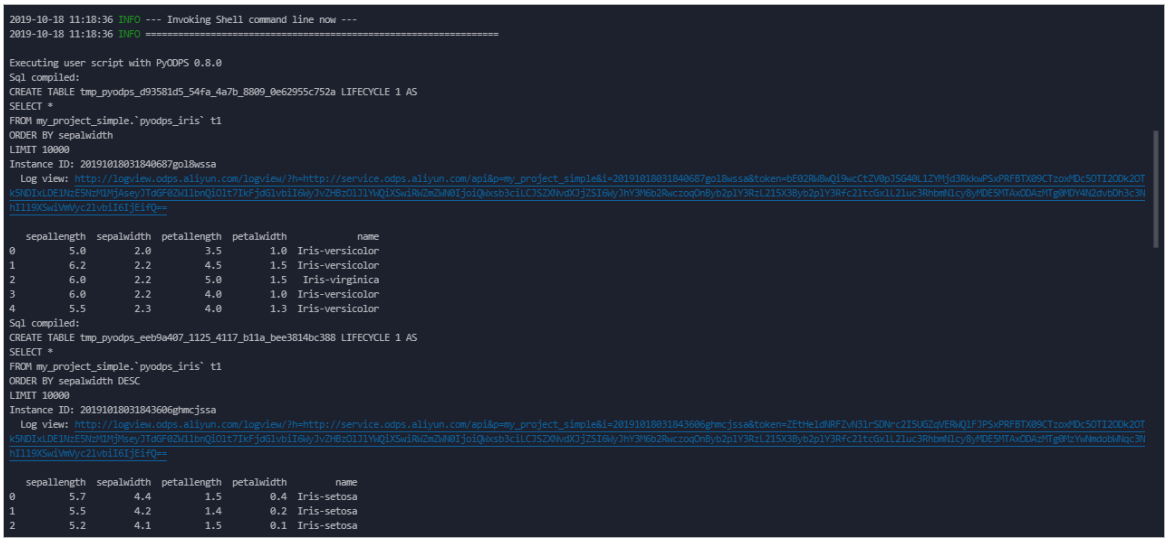
   ```
   CREATE TABLE if not exists pyodps_iris
   (
   sepallength  DOUBLE comment 'sepal length (cm)',
   sepalwidth   DOUBLE comment 'sepal width (cm)',
   petallength  DOUBLE comment ''petal length (cm)',
   petalwidth   DOUBLE comment 'petal width (cm)',
   name         STRING comment 'type'
   );
   ```

2. Log on to the DataWorks console.
3. In the left-side navigation pane, click **Workspaces**.
4. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.

5. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
6. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
7. On the configuration tab of the PyODPS 2 node, enter the following code in the code editor:

```
from  odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
# Sort data.
print iris.sort('sepalwidth').head(5)
# You can use one of the following methods to sort data in descending order:
# Set the ascending parameter to False.
print iris.sort('sepalwidth',ascending=False).head(5)
# Use a hyphen (-) as the sorting parameter.
print iris.sort(-iris.sepalwidth).head(5)
# Sort multiple fields.
print iris.sort(['sepalwidth','petallength']).head(5)
# To sort multiple fields in different orders, you can set the ascending parameter to a
list of BOOLEAN values. The number of values in the list must be the same as the number
of fields to be sorted.
print iris.sort(['sepalwidth','petallength'],ascending=[True,False]).head(5)
print iris.sort(['sepalwidth',-iris.petallength]).head(5)
```

8. Click the Run icon in the toolbar.
9. View the running result of the PyODPS 2 node on the **Run Log** tab.



In this example, the following information appears on the Run Log tab:

```
Sql compiled:
CREATE TABLE tmp_pyodps_d1b06785_dc18_4288_ad34_de860de1be08 LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pyodps_iris` t1
ORDER BY sepalwidth
LIMIT 10000
Instance ID: 20191010061554817gwml01im
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          5.0         1.5   Iris-virginica
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          4.0         1.0  Iris-versicolor
```

```
4          5.5         2.3          4.0         1.3  Iris-versicolor
Sql compiled:
CREATE TABLE tmp_pyodps_3cb90bb2_fb95_43fb_ae84_f2b5a27d72dc LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pyodps_iris` t1
ORDER BY sepalwidth DESC
LIMIT 10000
Instance ID: 20191010061601287gs086792
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.7         4.4          1.5         0.4  Iris-setosa
1          5.5         4.2          1.4         0.2  Iris-setosa
2          5.2         4.1          1.5         0.1  Iris-setosa
3          5.8         4.0          1.2         0.2  Iris-setosa
4          5.4         3.9          1.3         0.4  Iris-setosa
Sql compiled:
CREATE TABLE tmp_pyodps_97b080bb_e014_48e8_a310_4b45fcd6a2ed LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pyodps_iris` t1
ORDER BY sepalwidth DESC
LIMIT 10000
Instance ID: 20191010061606927g6emz192
   sepallength  sepalwidth  petallength  petalwidth         name
0          5.7         4.4          1.5         0.4  Iris-setosa
1          5.5         4.2          1.4         0.2  Iris-setosa
2          5.2         4.1          1.5         0.1  Iris-setosa
3          5.8         4.0          1.2         0.2  Iris-setosa
4          5.4         3.9          1.3         0.4  Iris-setosa
Sql compiled:
CREATE TABLE tmp_pyodps_6fe37b6e_6705_4052_b733_211eb9bd16ac LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pyodps_iris` t1
ORDER BY sepalwidth, petallength
LIMIT 10000
Instance ID: 20191010061611714gn586792
   sepallength  sepalwidth  petallength  petalwidth             name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          4.0         1.0  Iris-versicolor
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          5.0         1.5   Iris-virginica
4          4.5         2.3          1.3         0.3      Iris-setosa
Sql compiled:
CREATE TABLE tmp_pyodps_a52c805c_94a1_4a75_a6af_4fc9ed06ae68 LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pyodps_iris` t1
ORDER BY sepalwidth, petallength DESC
LIMIT 10000
Instance ID: 20191010061616553gw3m9592
   sepallength  sepalwidth  petallength  petalwidth             name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          5.0         1.5   Iris-virginica
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          4.0         1.0  Iris-versicolor
4          6.3         2.3          4.4         1.3  Iris-versicolor
Sql compiled:
```

```
CREATE TABLE tmp_pyodps_aac5538e_9b40_4078_b3c6_852b99c663c1 LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pyodps_iris` t1
ORDER BY sepalwidth, petallength DESC
LIMIT 10000
Instance ID: 20191010061621329gvmkc292
   sepallength  sepalwidth  petallength  petalwidth            name
0          5.0         2.0          3.5         1.0  Iris-versicolor
1          6.0         2.2          5.0         1.5   Iris-virginica
2          6.2         2.2          4.5         1.5  Iris-versicolor
3          6.0         2.2          4.0         1.0  Iris-versicolor
4          6.3         2.3          4.4         1.3  Iris-versicolor
```

4.1.9.10. Use a PyODPS node to de-duplicate data

This topic describes how to use a PyODPS node to de-duplicate data.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.
   ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

   In this example, execute the following statement to create the pyodps_iris table:

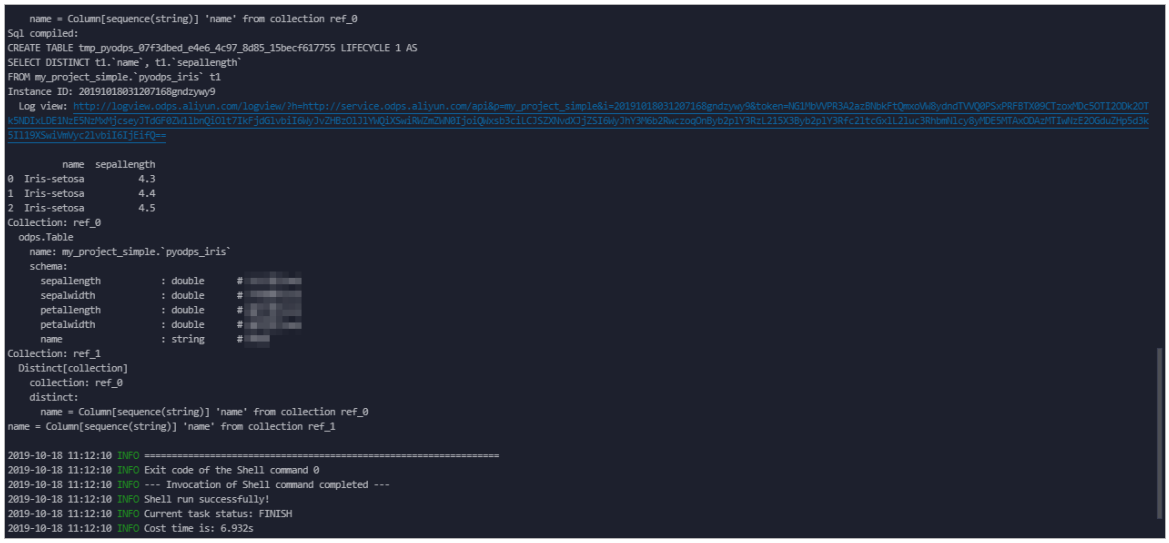
   ```
   CREATE TABLE if not exists pyodps_iris
   (
   sepallength  DOUBLE comment 'sepal length (cm)',
   sepalwidth   DOUBLE comment 'sepal width (cm)',
   petallength  DOUBLE comment ''petal length (cm)',
   petalwidth   DOUBLE comment 'petal width (cm)',
   name         STRING comment 'type'
   );
   ```

2. Log on to the DataWorks console.
3. In the left-side navigation pane, click **Workspaces**.
4. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.
5. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
6. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
7. On the configuration tab of the PyODPS 2 node, enter the code of the node in the code editor.

   In this example, enter the following code:

```
from  odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
print iris[['name']].distinct()
print iris.distinct('name')
print iris.distinct('name','sepallength').head(3)
# You can call the unique method to de-duplicate data in the specified sequence. The se
quence whose data is de-duplicated by using the unique method cannot be selected as a c
olumn.
print iris.name.unique()
```

8. Click the Run icon in the toolbar.
9. View the running result of the PyODPS 2 node on the **Run Log** tab.



In this example, the following information appears on the Run Log tab:

```
Sql compiled:
CREATE TABLE tmp_pyodps_ed85ebd5_d678_44dd_9ece_bff1822376f6 LIFECYCLE 1 AS
SELECT DISTINCT t1.`name`
FROM WB_BestPractice_dev.`pyodps_iris` t1
Instance ID: 20191010063911142g2cp5692
             name
0      Iris-setosa
1  Iris-versicolor
2   Iris-virginica
Sql compiled:
CREATE TABLE tmp_pyodps_8ce6128f_9c6f_45af_b9de_c73ce9d5ba51 LIFECYCLE 1 AS
SELECT DISTINCT t1.`name`
FROM WB_BestPractice_dev.`pyodps_iris` t1
Instance ID: 20191010063915987gmuws592
             name
0      Iris-setosa
1  Iris-versicolor
2   Iris-virginica
Sql compiled:
CREATE TABLE tmp_pyodps_a3dc338e_0fea_4d5f_847c_79fb19ec1c72 LIFECYCLE 1 AS
SELECT DISTINCT t1.`name`, t1.`sepallength`
FROM WB_BestPractice_dev.`pyodps_iris` t1
Instance ID: 20191010063922l0gj056292
          name  sepallength
0  Iris-setosa          4.3
1  Iris-setosa          4.4
2  Iris-setosa          4.5
Sql compiled:
CREATE TABLE tmp_pyodps_bc0917bb_f10c_426b_9b75_47e94478382a LIFECYCLE 1 AS
SELECT t2.`name`
FROM (
  SELECT DISTINCT t1.`name`
  FROM WB_BestPractice_dev.`pyodps_iris` t1
) t2
Instance ID: 20191010063927189g9fsz192
             name
0      Iris-setosa
1  Iris-versicolor
2   Iris-virginica
```

4.1.9.11. Use a PyODPS node to sample data

This topic describes how to use a PyODPS node to sample data.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
   i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.
   ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

   In this example, execute the following statement to create the pyodps_iris table:

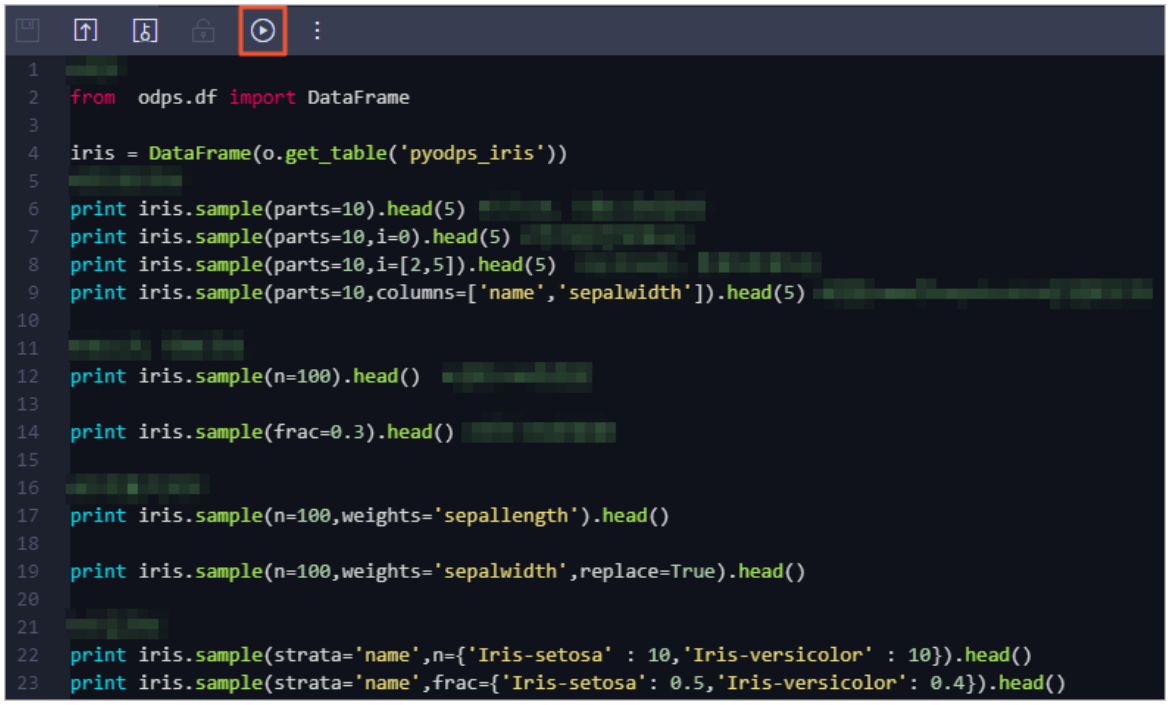
```
CREATE TABLE if not exists pyodps_iris
(
sepallength  DOUBLE comment 'sepal length (cm)',
sepalwidth   DOUBLE comment 'sepal width (cm)',
petallength  DOUBLE comment ''petal length (cm)',
petalwidth   DOUBLE comment 'petal width (cm)',
name         STRING comment 'type'
);
```

2. Log on to the DataWorks console.
3. In the left-side navigation pane, click **Workspaces**.
4. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.
5. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
6. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
7. On the configuration tab of the PyODPS node, enter the code of the node in the code editor.

   In this example, enter the following code:

```
# Sample data.
from  odps.df import DataFrame
iris = DataFrame(o.get_table('pyodps_iris'))
# Sample data by part.
print iris.sample(parts=10).head(5) # Divide the target data into 10 parts. Part 0 is o
btained by default.
print iris.sample(parts=10,i=0).head(5) # Divide the target data into 10 parts and obta
in part 0.
print iris.sample(parts=10,i=[2,5]).head(5)  # Divide the target data into 10 parts and
obtain part 2 and part 5.
print iris.sample(parts=10,columns=['name','sepalwidth']).head(5) # Divide the target d
ata into 10 parts and sample data based on the values of name and sepalwidth.
# Sample data based on the specified number of data records or proportion.
print iris.sample(n=100).head()  # Obtain 100 data records.
print iris.sample(frac=0.3).head() # Obtain 30% of the target data.
# Sample data based on the weight column.
print iris.sample(n=100,weights='sepallength').head()
print iris.sample(n=100,weights='sepalwidth',replace=True).head()
# Perform stratified sampling.
print iris.sample(strata='name',n={'Iris-setosa' : 10,'Iris-versicolor' : 10}).head()
print iris.sample(strata='name',frac={'Iris-setosa': 0.5,'Iris-versicolor': 0.4}).head(
)
```

8. Click the Run icon in the toolbar.



```
from  odps.df import DataFrame

iris = DataFrame(o.get_table('pyodps_iris'))

print iris.sample(parts=10).head(5)
print iris.sample(parts=10,i=0).head(5)
print iris.sample(parts=10,i=[2,5]).head(5)
print iris.sample(parts=10,columns=['name','sepalwidth']).head(5)

print iris.sample(n=100).head()

print iris.sample(frac=0.3).head()

print iris.sample(n=100,weights='sepallength').head()

print iris.sample(n=100,weights='sepalwidth',replace=True).head()

print iris.sample(strata='name',n={'Iris-setosa' : 10,'Iris-versicolor' : 10}).head()
print iris.sample(strata='name',frac={'Iris-setosa': 0.5,'Iris-versicolor': 0.4}).head()
```

9. View the running result of the PyODPS 2 node on the **Run Log** tab.

   In this example, the following information appears on the Run Log tab:

```
Collection: ref_0
  odps.Table
    name: WB_BestPractice_dev.`pyodps_iris`
    schema:
      sepallength           : double      # Sepal length (cm)
      sepalwidth            : double      # Sepal width (cm)
      petallength           : double      # Petal length (cm)
      petalwidth            : double      # Petal width (cm)
      name                  : string      # Type
Sample[collection]
  _input:
  _parts:
    Scalar[int8]
      10
  _i:
  _replace:
    Scalar[boolean]
      False
Collection: ref_0
  odps.Table
    name: WB_BestPractice_dev.`pyodps_iris`
    schema:
      sepallength           : double      # Sepal length (cm)
      sepalwidth            : double      # Sepal width (cm)
      petallength           : double      # Petal length (cm)
      petalwidth            : double      # Petal width (cm)
      name                  : string      # Type
Sample[collection]
  _input:
   parts:
```

```
  _
    Scalar[int8]
      10
  _i:
  _replace:
    Scalar[boolean]
      False
Collection: ref_0
  odps.Table
    name: WB_BestPractice_dev.`pyodps_iris`
    schema:
      sepallength           : double      # Sepal length (cm)
      sepalwidth            : double      # Sepal width (cm)
      petallength           : double      # Petal length (cm)
      petalwidth            : double      # Petal width (cm)
      name                  : string      # Type
Sample[collection]
  _input:
  _parts:
    Scalar[int8]
      10
  _i:
  _replace:
    Scalar[boolean]
      False
Collection: ref_0
  odps.Table
    name: WB_BestPractice_dev.`pyodps_iris`
    schema:
      sepallength           : double      # Sepal length (cm)
      sepalwidth            : double      # Sepal width (cm)
      petallength           : double      # Petal length (cm)
      petalwidth            : double      # Petal width (cm)
      name                  : string      # Type
Sample[collection]
  _input:
  _parts:
    Scalar[int8]
      10
  _i:
  _sampled_fields:
    name = Column[sequence(string)] 'name' from collection ref_0
    sepalwidth = Column[sequence(float64)] 'sepalwidth' from collection ref_0
  _replace:
    Scalar[boolean]
      False
Executing RandomSample...
Command: PAI -name RandomSample -project algo_public -Dreplace="false" -Dlifecycle="1"
-DoutputTableName="tmp_pyodps_1570690014_69f3d75d_9537_4c9c_87ea_a5f6ad8d2e07" -Dsample
Size="100" -DinputTableName="WB_BestPractice_dev.pyodps_iris";
Instance ID: 20191010064654985g6co9592
Sub Instance: create_output (20191010064700688g5cbn62m_71ee0561_bcc4_4147_b849_f7468835
3fb6)
Sub Instance: without_replacement (20191010064703694g9cbn62m_93a8a15b_ffd1_4afe_8928_19
f28455a15c)
```

```
Try to fetch data from tunnel
    sepallength  sepalwidth  petallength  petalwidth            name
0           5.1         3.5          1.4         0.2     Iris-setosa
1           4.9         3.0          1.4         0.2     Iris-setosa
2           4.7         3.2          1.3         0.2     Iris-setosa
3           4.6         3.1          1.5         0.2     Iris-setosa
4           5.0         3.6          1.4         0.2     Iris-setosa
5           4.6         3.4          1.4         0.3     Iris-setosa
6           4.4         2.9          1.4         0.2     Iris-setosa
7           4.9         3.1          1.5         0.1     Iris-setosa
8           4.8         3.4          1.6         0.2     Iris-setosa
9           4.8         3.0          1.4         0.1     Iris-setosa
10          4.3         3.0          1.1         0.1     Iris-setosa
11          5.1         3.5          1.4         0.3     Iris-setosa
12          5.7         3.8          1.7         0.3     Iris-setosa
13          5.1         3.8          1.5         0.3     Iris-setosa
14          5.1         3.7          1.5         0.4     Iris-setosa
15          4.6         3.6          1.0         0.2     Iris-setosa
16          5.1         3.3          1.7         0.5     Iris-setosa
17          4.8         3.4          1.9         0.2     Iris-setosa
18          5.0         3.4          1.6         0.4     Iris-setosa
19          5.2         3.5          1.5         0.2     Iris-setosa
20          5.2         3.4          1.4         0.2     Iris-setosa
21          4.8         3.1          1.6         0.2     Iris-setosa
22          5.2         4.1          1.5         0.1     Iris-setosa
23          5.5         4.2          1.4         0.2     Iris-setosa
24          4.9         3.1          1.5         0.1     Iris-setosa
25          5.0         3.2          1.2         0.2     Iris-setosa
26          4.4         3.0          1.3         0.2     Iris-setosa
27          5.1         3.4          1.5         0.2     Iris-setosa
28          5.0         3.5          1.3         0.3     Iris-setosa
29          4.5         2.3          1.3         0.3     Iris-setosa
..          ...         ...          ...         ...             ...
70          7.1         3.0          5.9         2.1  Iris-virginica
71          7.6         3.0          6.6         2.1  Iris-virginica
72          7.3         2.9          6.3         1.8  Iris-virginica
73          7.2         3.6          6.1         2.5  Iris-virginica
74          6.5         3.2          5.1         2.0  Iris-virginica
75          6.8         3.0          5.5         2.1  Iris-virginica
76          5.8         2.8          5.1         2.4  Iris-virginica
77          7.7         3.8          6.7         2.2  Iris-virginica
78          7.7         2.6          6.9         2.3  Iris-virginica
79          7.7         2.8          6.7         2.0  Iris-virginica
80          6.3         2.7          4.9         1.8  Iris-virginica
81          6.7         3.3          5.7         2.1  Iris-virginica
82          6.2         2.8          4.8         1.8  Iris-virginica
83          6.1         3.0          4.9         1.8  Iris-virginica
84          6.4         2.8          5.6         2.1  Iris-virginica
85          7.2         3.0          5.8         1.6  Iris-virginica
86          7.4         2.8          6.1         1.9  Iris-virginica
87          7.9         3.8          6.4         2.0  Iris-virginica
88          6.3         2.8          5.1         1.5  Iris-virginica
89          6.3         3.4          5.6         2.4  Iris-virginica
90          6.4         3.1          5.5         1.8  Iris-virginica
```

```
91          6.0         3.0          4.8         1.8  Iris-virginica
92          6.9         3.1          5.4         2.1  Iris-virginica
93          6.9         3.1          5.1         2.3  Iris-virginica
94          5.8         2.7          5.1         1.9  Iris-virginica
95          6.8         3.2          5.9         2.3  Iris-virginica
96          6.7         3.3          5.7         2.5  Iris-virginica
97          6.3         2.5          5.0         1.9  Iris-virginica
98          6.2         3.4          5.4         2.3  Iris-virginica
99          5.9         3.0          5.1         1.8  Iris-virginica
[100 rows x 5 columns]
Executing RandomSample...
Command: PAI -name RandomSample -project algo_public -Dreplace="false" -DsampleRatio="0
.3" -DoutputTableName="tmp_pyodps_1570690039_e1867332_72ea_4656_928d_3bd6e31d87c7" -Dli
fecycle="1" -DinputTableName="WB_BestPractice_dev.pyodps_iris";
Instance ID: 20191010064720117gmpms38
Sub Instance: create_output (20191010064725740grcbn62m_b338a671_6047_4360_8792_41d2e748
e41f)
Sub Instance: without_replacement (20191010064728747gtcbn62m_6c9914da_d5c3_4336_b076_16
3edb1bf48a)
Try to fetch data from tunnel
    sepallength  sepalwidth  petallength  petalwidth            name
0           5.1         3.5          1.4         0.2     Iris-setosa
1           4.7         3.2          1.3         0.2     Iris-setosa
2           4.6         3.1          1.5         0.2     Iris-setosa
3           4.8         3.4          1.6         0.2     Iris-setosa
4           5.7         4.4          1.5         0.4     Iris-setosa
5           5.1         3.5          1.4         0.3     Iris-setosa
6           5.0         3.4          1.6         0.4     Iris-setosa
7           5.2         3.4          1.4         0.2     Iris-setosa
8           4.9         3.1          1.5         0.1     Iris-setosa
9           5.5         3.5          1.3         0.2     Iris-setosa
10          4.4         3.2          1.3         0.2     Iris-setosa
11          5.0         3.3          1.4         0.2     Iris-setosa
12          5.7         2.8          4.5         1.3 Iris-versicolor
13          5.2         2.7          3.9         1.4 Iris-versicolor
14          5.0         2.0          3.5         1.0 Iris-versicolor
15          5.6         2.9          3.6         1.3 Iris-versicolor
16          5.8         2.7          4.1         1.0 Iris-versicolor
17          6.1         2.8          4.0         1.3 Iris-versicolor
18          6.6         3.0          4.4         1.4 Iris-versicolor
19          6.8         2.8          4.8         1.4 Iris-versicolor
20          6.0         2.9          4.5         1.5 Iris-versicolor
21          5.4         3.0          4.5         1.5 Iris-versicolor
22          5.7         2.9          4.2         1.3 Iris-versicolor
23          5.7         2.8          4.1         1.3 Iris-versicolor
24          6.3         2.9          5.6         1.8  Iris-virginica
25          4.9         2.5          4.5         1.7  Iris-virginica
26          6.7         2.5          5.8         1.8  Iris-virginica
27          6.4         2.7          5.3         1.9  Iris-virginica
28          6.8         3.0          5.5         2.1  Iris-virginica
29          5.7         2.5          5.0         2.0  Iris-virginica
30          5.8         2.8          5.1         2.4  Iris-virginica
31          6.5         3.0          5.5         1.8  Iris-virginica
32          6.0         2.2          5.0         1.5  Iris-virginica
```

```
33          6.3         2.7         4.9         1.8   Iris-virginica
34          7.2         3.2         6.0         1.8   Iris-virginica
35          6.2         2.8         4.8         1.8   Iris-virginica
36          6.1         3.0         4.9         1.8   Iris-virginica
37          6.4         2.8         5.6         2.1   Iris-virginica
38          7.2         3.0         5.8         1.6   Iris-virginica
39          6.3         2.8         5.1         1.5   Iris-virginica
40          6.4         3.1         5.5         1.8   Iris-virginica
41          6.0         3.0         4.8         1.8   Iris-virginica
42          6.8         3.2         5.9         2.3   Iris-virginica
43          6.3         2.5         5.0         1.9   Iris-virginica
44          6.2         3.4         5.4         2.3   Iris-virginica
Executing WeightedSample...
Command: PAI -name WeightedSample -project algo_public -DinputTableName="WB_BestPractic
e_dev.pyodps_iris" -DsampleSize="100" -DprobCol="sepallength" -Dreplace="false" -Doutpu
tTableName="tmp_pyodps_1570690063_6a62857e_8f85_4ea7_99ef_08aa259546d4" -Dlifecycle="1"
;
Instance ID: 20191010064743533gnpms38
Sub Instance: create_output (20191010064748787gkdbn62m_8d47bfb7_e470_4cce_8b69_28811f19
0083)
Sub Instance: without_replacement (20191010064751793gmdbn62m_230a1d26_5c2e_440e_a31d_fe
9e63c6f906)
Try to fetch data from tunnel
    sepallength  sepalwidth  petallength  petalwidth            name
0           4.9         3.0          1.4         0.2     Iris-setosa
1           4.7         3.2          1.3         0.2     Iris-setosa
2           5.0         3.6          1.4         0.2     Iris-setosa
3           5.4         3.9          1.7         0.4     Iris-setosa
4           5.0         3.4          1.5         0.2     Iris-setosa
5           4.4         2.9          1.4         0.2     Iris-setosa
6           4.8         3.4          1.6         0.2     Iris-setosa
7           4.8         3.0          1.4         0.1     Iris-setosa
8           5.4         3.9          1.3         0.4     Iris-setosa
9           5.1         3.5          1.4         0.3     Iris-setosa
10          5.7         3.8          1.7         0.3     Iris-setosa
11          4.6         3.6          1.0         0.2     Iris-setosa
12          5.0         3.4          1.6         0.4     Iris-setosa
13          5.2         3.5          1.5         0.2     Iris-setosa
14          5.2         3.4          1.4         0.2     Iris-setosa
15          4.7         3.2          1.6         0.2     Iris-setosa
16          4.8         3.1          1.6         0.2     Iris-setosa
17          5.5         4.2          1.4         0.2     Iris-setosa
18          4.9         3.1          1.5         0.1     Iris-setosa
19          5.0         3.2          1.2         0.2     Iris-setosa
20          5.5         3.5          1.3         0.2     Iris-setosa
21          4.9         3.1          1.5         0.1     Iris-setosa
22          5.1         3.4          1.5         0.2     Iris-setosa
23          4.5         2.3          1.3         0.3     Iris-setosa
24          4.8         3.0          1.4         0.3     Iris-setosa
25          5.1         3.8          1.6         0.2     Iris-setosa
26          4.6         3.2          1.4         0.2     Iris-setosa
27          5.3         3.7          1.5         0.2     Iris-setosa
28          5.0         3.3          1.4         0.2     Iris-setosa
29          7.0         3.2          4.7         1.4  Iris-versicolor
```

```
..          ...         ...         ...         ...             ...
70          7.2         3.6         6.1         2.5   Iris-virginica
71          6.4         2.7         5.3         1.9   Iris-virginica
72          5.7         2.5         5.0         2.0   Iris-virginica
73          5.8         2.8         5.1         2.4   Iris-virginica
74          6.4         3.2         5.3         2.3   Iris-virginica
75          7.7         3.8         6.7         2.2   Iris-virginica
76          6.9         3.2         5.7         2.3   Iris-virginica
77          5.6         2.8         4.9         2.0   Iris-virginica
78          7.7         2.8         6.7         2.0   Iris-virginica
79          6.3         2.7         4.9         1.8   Iris-virginica
80          6.7         3.3         5.7         2.1   Iris-virginica
81          7.2         3.2         6.0         1.8   Iris-virginica
82          6.2         2.8         4.8         1.8   Iris-virginica
83          6.1         3.0         4.9         1.8   Iris-virginica
84          7.2         3.0         5.8         1.6   Iris-virginica
85          7.9         3.8         6.4         2.0   Iris-virginica
86          6.4         2.8         5.6         2.2   Iris-virginica
87          6.3         2.8         5.1         1.5   Iris-virginica
88          6.1         2.6         5.6         1.4   Iris-virginica
89          6.3         3.4         5.6         2.4   Iris-virginica
90          6.4         3.1         5.5         1.8   Iris-virginica
91          6.0         3.0         4.8         1.8   Iris-virginica
92          6.9         3.1         5.1         2.3   Iris-virginica
93          5.8         2.7         5.1         1.9   Iris-virginica
94          6.8         3.2         5.9         2.3   Iris-virginica
95          6.7         3.3         5.7         2.5   Iris-virginica
96          6.7         3.0         5.2         2.3   Iris-virginica
97          6.5         3.0         5.2         2.0   Iris-virginica
98          6.2         3.4         5.4         2.3   Iris-virginica
99          5.9         3.0         5.1         1.8   Iris-virginica
[100 rows x 5 columns]
Executing WeightedSample...
Command: PAI -name WeightedSample -project algo_public -DinputTableName="WB_BestPractic
e_dev.pyodps_iris" -DsampleSize="100" -DprobCol="sepalwidth" -Dreplace="true" -DoutputT
ableName="tmp_pyodps_1570690082_f55e899c_3cb4_4eeb_ade4_b8cb79e018dc" -Dlifecycle="1";
Instance ID: 20191010064803392g9ers38
Sub Instance: create_output (20191010064808827g9ebn62m_fb70c859_913a_4830_9248_8c8eaf13
4f1d)
Sub Instance: with_replacement (20191010064811833gdebn62m_07544cc2_1d7d_4fa5_972d_196eb
6b9f537)
    sepallength  sepalwidth  petallength  petalwidth            name
0           5.1         3.5          1.4         0.2     Iris-setosa
1           4.6         3.4          1.4         0.3     Iris-setosa
2           5.0         3.4          1.5         0.2     Iris-setosa
3           5.0         3.4          1.5         0.2     Iris-setosa
4           4.4         2.9          1.4         0.2     Iris-setosa
5           4.8         3.4          1.6         0.2     Iris-setosa
6           5.8         4.0          1.2         0.2     Iris-setosa
7           5.8         4.0          1.2         0.2     Iris-setosa
8           5.1         3.5          1.4         0.3     Iris-setosa
9           5.1         3.5          1.4         0.3     Iris-setosa
10          5.1         3.5          1.4         0.3     Iris-setosa
11          5.1         3.7          1.5         0.4     Iris-setosa
12          4.6         3.6          1.0         0.2     Iris-setosa
```

```
13          4.8         3.4         1.9         0.2      Iris-setosa
14          5.0         3.0         1.6         0.2      Iris-setosa
15          5.0         3.4         1.6         0.4      Iris-setosa
16          5.2         3.4         1.4         0.2      Iris-setosa
17          4.8         3.1         1.6         0.2      Iris-setosa
18          4.8         3.1         1.6         0.2      Iris-setosa
19          5.4         3.4         1.5         0.4      Iris-setosa
20          5.4         3.4         1.5         0.4      Iris-setosa
21          5.4         3.4         1.5         0.4      Iris-setosa
22          5.2         4.1         1.5         0.1      Iris-setosa
23          5.2         4.1         1.5         0.1      Iris-setosa
24          5.5         4.2         1.4         0.2      Iris-setosa
25          5.0         3.2         1.2         0.2      Iris-setosa
26          4.9         3.1         1.5         0.1      Iris-setosa
27          4.4         3.0         1.3         0.2      Iris-setosa
28          5.1         3.4         1.5         0.2      Iris-setosa
29          5.0         3.5         1.3         0.3      Iris-setosa
..          ...         ...         ...         ...              ...
70          5.6         2.7         4.2         1.3  Iris-versicolor
71          5.7         2.9         4.2         1.3  Iris-versicolor
72          6.3         3.3         6.0         2.5   Iris-virginica
73          7.1         3.0         5.9         2.1   Iris-virginica
74          6.3         2.9         5.6         1.8   Iris-virginica
75          7.3         2.9         6.3         1.8   Iris-virginica
76          7.2         3.6         6.1         2.5   Iris-virginica
77          6.4         2.7         5.3         1.9   Iris-virginica
78          6.8         3.0         5.5         2.1   Iris-virginica
79          6.8         3.0         5.5         2.1   Iris-virginica
80          5.8         2.8         5.1         2.4   Iris-virginica
81          6.2         2.8         4.8         1.8   Iris-virginica
82          6.2         2.8         4.8         1.8   Iris-virginica
83          6.1         3.0         4.9         1.8   Iris-virginica
84          6.4         2.8         5.6         2.1   Iris-virginica
85          7.2         3.0         5.8         1.6   Iris-virginica
86          7.4         2.8         6.1         1.9   Iris-virginica
87          7.4         2.8         6.1         1.9   Iris-virginica
88          7.4         2.8         6.1         1.9   Iris-virginica
89          6.4         3.1         5.5         1.8   Iris-virginica
90          6.0         3.0         4.8         1.8   Iris-virginica
91          6.9         3.1         5.4         2.1   Iris-virginica
92          6.7         3.1         5.6         2.4   Iris-virginica
93          6.7         3.1         5.6         2.4   Iris-virginica
94          6.7         3.1         5.6         2.4   Iris-virginica
95          6.9         3.1         5.1         2.3   Iris-virginica
96          6.9         3.1         5.1         2.3   Iris-virginica
97          6.7         3.0         5.2         2.3   Iris-virginica
98          6.5         3.0         5.2         2.0   Iris-virginica
99          5.9         3.0         5.1         1.8   Iris-virginica
[100 rows x 5 columns]
Executing StratifiedSample...
Command: PAI -name StratifiedSample -project algo_public -Dlifecycle="1" -DoutputTableN
ame="tmp_pyodps_1570690104_6cc52795_2a86_4634_a905_740b3a426d3f" -DsampleSize="Iris-set
osa:10,Iris-versicolor:10" -DstrataColName="name" -DinputTableName="WB_BestPractice_dev
.pyodps_iris";
```

```
Instance ID: 20191010064824633gaco9592
Sub Instance: create_output (20191010064829870g4fbn62m_dfa297c5_23b5_43f5_bb17_83ba8d26
3630)
Sub Instance: stratified_sampling (20191010064831874g7fbn62m_fc9eddb7_42f1_49fe_8206_89
1ef451fb76)
Try to fetch data from tunnel
    sepallength  sepalwidth  petallength  petalwidth            name
0           5.4         3.9          1.7         0.4      Iris-setosa
1           4.3         3.0          1.1         0.1      Iris-setosa
2           5.4         3.9          1.3         0.4      Iris-setosa
3           5.1         3.3          1.7         0.5      Iris-setosa
4           4.7         3.2          1.6         0.2      Iris-setosa
5           4.5         2.3          1.3         0.3      Iris-setosa
6           5.0         3.5          1.6         0.6      Iris-setosa
7           5.1         3.8          1.9         0.4      Iris-setosa
8           4.8         3.0          1.4         0.3      Iris-setosa
9           5.0         3.3          1.4         0.2      Iris-setosa
10          7.0         3.2          4.7         1.4  Iris-versicolor
11          5.5         2.3          4.0         1.3  Iris-versicolor
12          6.5         2.8          4.6         1.5  Iris-versicolor
13          5.6         3.0          4.5         1.5  Iris-versicolor
14          5.7         2.6          3.5         1.0  Iris-versicolor
15          5.5         2.4          3.7         1.0  Iris-versicolor
16          5.0         2.3          3.3         1.0  Iris-versicolor
17          5.6         2.7          4.2         1.3  Iris-versicolor
18          5.7         3.0          4.2         1.2  Iris-versicolor
19          5.1         2.5          3.0         1.1  Iris-versicolor
Executing StratifiedSample...
Command: PAI -name StratifiedSample -project algo_public -DsampleRatio="Iris-setosa:0.5
,Iris-versicolor:0.4" -DoutputTableName="tmp_pyodps_1570690128_a68477cd_19e5_4fe0_bb39_
4712f76dd967" -Dlifecycle="1" -DstrataColName="name" -DinputTableName="WB_BestPractice_
dev.pyodps_iris";
Instance ID: 20191010064848733gbers38
Sub Instance: create_output (20191010064853918gwfbn62m_4eb22c22_7051_4372_8d13_05c5a417
aa87)
Sub Instance: stratified_sampling (20191010064855924g0gbn62m_b4242ac7_bd5a_47a8_a1f2_33
67a6a101a7)
Try to fetch data from tunnel
    sepallength  sepalwidth  petallength  petalwidth            name
0           4.9         3.0          1.4         0.2      Iris-setosa
1           4.7         3.2          1.3         0.2      Iris-setosa
2           5.0         3.6          1.4         0.2      Iris-setosa
3           5.4         3.9          1.7         0.4      Iris-setosa
4           5.0         3.4          1.5         0.2      Iris-setosa
5           5.4         3.7          1.5         0.2      Iris-setosa
6           4.8         3.4          1.6         0.2      Iris-setosa
7           4.8         3.0          1.4         0.1      Iris-setosa
8           5.8         4.0          1.2         0.2      Iris-setosa
9           5.4         3.4          1.7         0.2      Iris-setosa
10          5.1         3.7          1.5         0.4      Iris-setosa
11          4.8         3.4          1.9         0.2      Iris-setosa
12          5.0         3.0          1.6         0.2      Iris-setosa
13          5.0         3.4          1.6         0.4      Iris-setosa
14          5.2         3.5          1.5         0.2      Iris-setosa
```

```
15          5.2         3.4         1.4         0.2      Iris-setosa
16          4.7         3.2         1.6         0.2      Iris-setosa
17          5.2         4.1         1.5         0.1      Iris-setosa
18          5.0         3.2         1.2         0.2      Iris-setosa
19          5.1         3.4         1.5         0.2      Iris-setosa
20          4.5         2.3         1.3         0.3      Iris-setosa
21          5.0         3.5         1.6         0.6      Iris-setosa
22          5.1         3.8         1.9         0.4      Iris-setosa
23          5.1         3.8         1.6         0.2      Iris-setosa
24          5.3         3.7         1.5         0.2      Iris-setosa
25          7.0         3.2         4.7         1.4  Iris-versicolor
26          6.4         3.2         4.5         1.5  Iris-versicolor
27          6.9         3.1         4.9         1.5  Iris-versicolor
28          6.5         2.8         4.6         1.5  Iris-versicolor
29          5.7         2.8         4.5         1.3  Iris-versicolor
30          6.6         2.9         4.6         1.3  Iris-versicolor
31          5.6         2.9         3.6         1.3  Iris-versicolor
32          5.6         3.0         4.5         1.5  Iris-versicolor
33          5.6         2.5         3.9         1.1  Iris-versicolor
34          6.1         2.8         4.7         1.2  Iris-versicolor
35          6.8         2.8         4.8         1.4  Iris-versicolor
36          5.5         2.4         3.8         1.1  Iris-versicolor
37          5.5         2.4         3.7         1.0  Iris-versicolor
38          6.0         2.7         5.1         1.6  Iris-versicolor
39          5.6         3.0         4.1         1.3  Iris-versicolor
40          5.5         2.6         4.4         1.2  Iris-versicolor
41          6.1         3.0         4.6         1.4  Iris-versicolor
42          5.7         3.0         4.2         1.2  Iris-versicolor
43          5.7         2.9         4.2         1.3  Iris-versicolor
44          6.2         2.9         4.3         1.3  Iris-versicolor
```

4.1.9.12. Use a PyODPS node to scale data

This topic describes how to use a PyODPS node to scale data.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Create a table and import data to it.
    i. Download the Iris dataset named *iris.data* and rename it *iris.csv*.

ii. Create a table named pyodps_iris and upload the *iris.csv* dataset. For more information, see Create tables and import data.

In this example, execute the following statement to create the pyodps_iris table:

```
CREATE TABLE if not exists pyodps_iris
(
sepallength  DOUBLE comment 'sepal length (cm)',
sepalwidth   DOUBLE comment 'sepal width (cm)',
petallength  DOUBLE comment ''petal length (cm)',
petalwidth   DOUBLE comment 'petal width (cm)',
name         STRING comment 'type'
);
```

2.
3. In the left-side navigation pane, click **Workspaces**.
4. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
5. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
6. On the configuration tab of the PyODPS 2 node, enter the code of the node in the code editor:

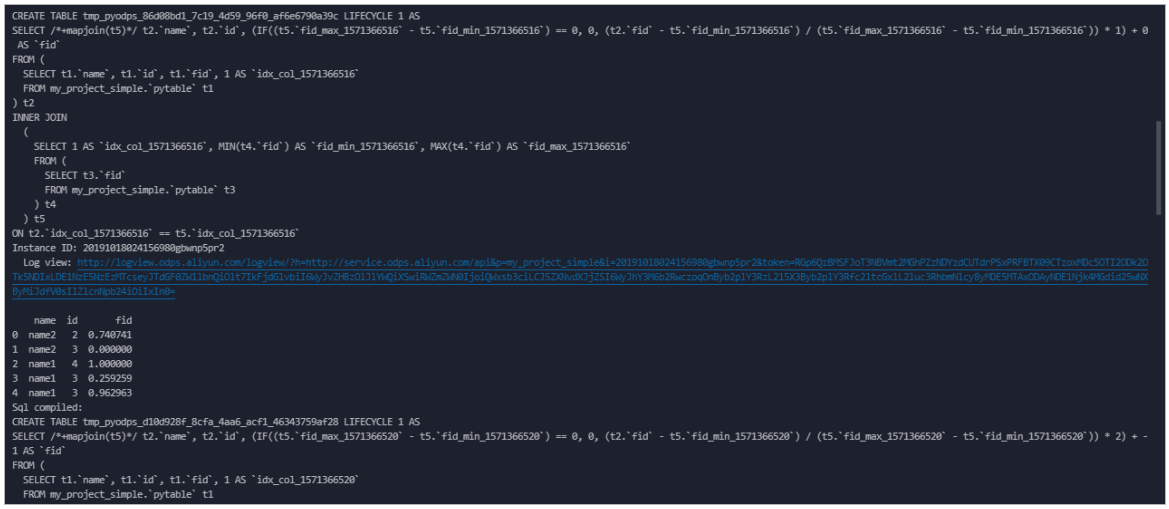
   In this example, enter the following code:

```
# Scale data.
df = DataFrame(o.get_table('pytable'))
# Call the min_max_scale method to normalize data.
print df.min_max_scale(columns=['fid']).head()
# Use the feature_range parameter to specify an output value range for the min_max_scal
e method. For example, to specify (-1,1) as the output value range, execute the followi
ng statement:
print df.min_max_scale(columns=['fid'],feature_range=(-1,1)).head()
# To keep the original values, set the preserve parameter to True. In this case, a new
column that contains the scaled data is added. The new column is named in the format Or
iginal column name_scaled. You can change the suffix _scaled by using the suffix parame
ter.
print df.min_max_scale(columns=['fid'],preserve=True).head()
# Use the group parameter to specify a group column.
print df.min_max_scale(columns=['fid'],group=['name']).head()
# You can use the group parameter to specify one or more group columns for the std_scal
e method. Similar to the min_max_scale method, the std_scale method obtains the minimum
and maximum values from the group columns to scale data.
print df.std_scale(columns=['fid']).head()
```

7. Click the Run icon in the toolbar.

```
df2 = DataFrame(o.get_table('pytable2'))

print df2.dropna(subset=['f1','f2','f3','f4']).head()

print df2.dropna(how='all', subset=['f1','f2','f3','f4']).head()

print df2.dropna(thresh=3, subset=['f1', 'f2', 'f3', 'f4']).head()

print df2.fillna(100, subset=['f1','f2','f3','f4']).head()

print df2.fillna(df2.f2, subset=['f1','f2','f3','f4']).head()

print df2.fillna(method='bfill', subset=['f1', 'f2', 'f3', 'f4']).head()

print df2.fillna(method='ffill', subset=['f1', 'f2', 'f3', 'f4']).head()
```

8. View the running result of the PyODPS 2 node on the **Run Log** tab.



In this example, the following information appears on the Run Log tab:

```
Sql compiled:
CREATE TABLE tmp_pyodps_59cb5533_6dc3_4b64_9312_b2da29d70cee LIFECYCLE 1 AS
SELECT /*+mapjoin(t5)*/ t2.`name`, t2.`id`, (IF((t5.`fid_max_1570691133` - t5.`fid_min_
1570691133`) == 0, 0, (t2.`fid` - t5.`fid_min_1570691133`) / (t5.`fid_max_1570691133` -
t5.`fid_min_1570691133`)) * 1) + 0 AS `fid`
FROM (
  SELECT t1.`name`, t1.`id`, t1.`fid`, 1 AS `idx_col_1570691133`
  FROM WB_BestPractice_dev.`pytable` t1
) t2
INNER JOIN
  (
    SELECT 1 AS `idx_col_1570691133`, MIN(t4.`fid`) AS `fid_min_1570691133`, MAX(t4.`fi
d`) AS `fid max 1570691133`
```

```
    FROM (
      SELECT t3.`fid`
      FROM WB_BestPractice_dev.`pytable` t3
    ) t4
  ) t5
ON t2.`idx_col_1570691133` == t5.`idx_col_1570691133`
Instance ID: 20191010070534951gl8by72m
    name  id       fid
0  name1   4  1.000000
1  name2   2  0.526316
2  name2   3  0.000000
3  name1   4  0.710526
4  name1   3  0.184211
5  name1   3  0.684211
Sql compiled:
CREATE TABLE tmp_pyodps_88a24967_3bdc_41ce_85d7_4bbd23e4ce01 LIFECYCLE 1 AS
SELECT /*+mapjoin(t5)*/ t2.`name`, t2.`id`, (IF((t5.`fid_max_1570691139` - t5.`fid_min_
1570691139`) == 0, 0, (t2.`fid` - t5.`fid_min_1570691139`) / (t5.`fid_max_1570691139` -
t5.`fid_min_1570691139`)) * 2) + -1 AS `fid`
FROM (
  SELECT t1.`name`, t1.`id`, t1.`fid`, 1 AS `idx_col_1570691139`
  FROM WB_BestPractice_dev.`pytable` t1
) t2
INNER JOIN
  (
    SELECT 1 AS `idx_col_1570691139`, MIN(t4.`fid`) AS `fid_min_1570691139`, MAX(t4.`fi
d`) AS `fid_max_1570691139`
    FROM (
      SELECT t3.`fid`
      FROM WB_BestPractice_dev.`pytable` t3
    ) t4
  ) t5
ON t2.`idx_col_1570691139` == t5.`idx_col_1570691139`
Instance ID: 20191010070539772gjo56292
    name  id       fid
0  name1   4  1.000000
1  name2   2  0.052632
2  name2   3 -1.000000
3  name1   4  0.421053
4  name1   3 -0.631579
5  name1   3  0.368421
Sql compiled:
CREATE TABLE tmp_pyodps_439117fc_9ef7_4086_899d_a5bf77d653e5 LIFECYCLE 1 AS
SELECT /*+mapjoin(t5)*/ t2.`name`, t2.`id`, t2.`fid`, (IF((t5.`fid_max_1570691146` - t5
.`fid_min_1570691146`) == 0, 0, (t2.`fid` - t5.`fid_min_1570691146`) / (t5.`fid_max_157
0691146` - t5.`fid_min_1570691146`)) * 1) + 0 AS `fid_scaled`
FROM (
  SELECT t1.`name`, t1.`id`, t1.`fid`, 1 AS `idx_col_1570691146`
  FROM WB_BestPractice_dev.`pytable` t1
) t2
INNER JOIN
  (
    SELECT 1 AS `idx_col_1570691146`, MIN(t4.`fid`) AS `fid_min_1570691146`, MAX(t4.`fi
d`) AS `fid_max_1570691146`
```

```
    FROM (
      SELECT t3.`fid`
      FROM WB_BestPractice_dev.`pytable` t3
    ) t4
  ) t5
ON t2.`idx_col_1570691146` == t5.`idx_col_1570691146`
Instance ID: 20191010070546769g0c14f72
    name  id  fid  fid_scaled
0  name1   4  5.3    1.000000
1  name2   2  3.5    0.526316
2  name2   3  1.5    0.000000
3  name1   4  4.2    0.710526
4  name1   3  2.2    0.184211
5  name1   3  4.1    0.684211
Sql compiled:
CREATE TABLE tmp_pyodps_d3839b4b_1087_4d52_91f5_52763b72f272 LIFECYCLE 1 AS
SELECT /*+mapjoin(t3)*/ t1.`name`, t1.`id`, (IF((t3.`fid_max_1570691151` - t3.`fid_min_
1570691151`) == 0, 0, (t1.`fid` - t3.`fid_min_1570691151`) / (t3.`fid_max_1570691151` -
t3.`fid_min_1570691151`)) * 1) + 0 AS `fid`
FROM WB_BestPractice_dev.`pytable` t1
INNER JOIN
  (
    SELECT t2.`name`, MAX(t2.`fid`) AS `fid_max_1570691151`, MIN(t2.`fid`) AS `fid_min_
1570691151`
    FROM WB_BestPractice_dev.`pytable` t2
    GROUP BY t2.`name`
  ) t3
ON t1.`name` == t3.`name`
Instance ID: 20191010070551756gtf14f72
    name  id       fid
0  name1   4  1.000000
1  name2   2  1.000000
2  name2   3  0.000000
3  name1   4  0.645161
4  name1   3  0.000000
5  name1   3  0.612903
Sql compiled:
CREATE TABLE tmp_pyodps_1ea6e5b4_129f_4d1e_a6a7_410e08d77ae6 LIFECYCLE 1 AS
SELECT /*+mapjoin(t5)*/ t2.`name`, t2.`id`, IF(t5.`fid_std_1570691157` == 0, 0, (t2.`fi
d` - t5.`fid_mean_1570691157`) / t5.`fid_std_1570691157`) AS `fid`
FROM (
  SELECT t1.`name`, t1.`id`, t1.`fid`, 1 AS `idx_col_1570691157`
  FROM WB_BestPractice_dev.`pytable` t1
) t2
INNER JOIN
  (
    SELECT 1 AS `idx_col_1570691157`, AVG(t4.`fid`) AS `fid_mean_1570691157`, STDDEV(t4
.`fid`) AS `fid_std_1570691157`
    FROM (
      SELECT t3.`fid`
      FROM WB_BestPractice_dev.`pytable` t3
    ) t4
  ) t5
ON t2.`idx_col_1570691157` == t5.`idx_col_1570691157`
```

```
Instance ID: 20191010070557788gdl14f72
    name  id       fid
0  name1   4  1.436467
1  name2   2  0.026118
2  name2   3 -1.540938
3  name1   4  0.574587
4  name1   3 -0.992468
5  name1   3  0.496234
```

4.1.9.13. Use a PyODPS node to process NULL values

This topic describes how to use a PyODPS node to process NULL values.

Prerequisites

The following operations are completed:

- MaxCompute is activated. For more information, see Activate MaxCompute.
- DataWorks is activated. For more information, see Activate DataWorks.
- A workflow is created in the DataWorks console. In this example, a workflow is created for a DataWorks workspace in basic mode. For more information, see Create a workflow.

Procedure

1. Prepare test data.
   i. Log on to the DataWorks console.
   ii. Create a table and import data to the table. For more information, see Create tables and import data.

   Execute the following statement to create a table named pytable2:

   ```
   CREATE TABLE `pytable2` (
     `id` string,
     `name` string,
     `f1` double,
     `f2` double,
     `f3` double,
     `f4` double
   ) ;
   ```

   Import the data in the pytable2.txt file to the pytable2 table. The pytable2.txt file contains the following data:

   ```
   0, name1, 1.0, NaN, 3.0, 4.0
   1, name1, 2.0, NaN, NaN, 1.0
   2, name1, 3.0, 4.0, 1.0, NaN
   3, name1, NaN, 1.0, 2.0, 3.0
   4, name1, 1.0, NaN, 3.0, 4.0
   5, name1, 1.0, 2.0, 3.0, 4.0
   6, name1, NaN, NaN, NaN, NaN
   ```

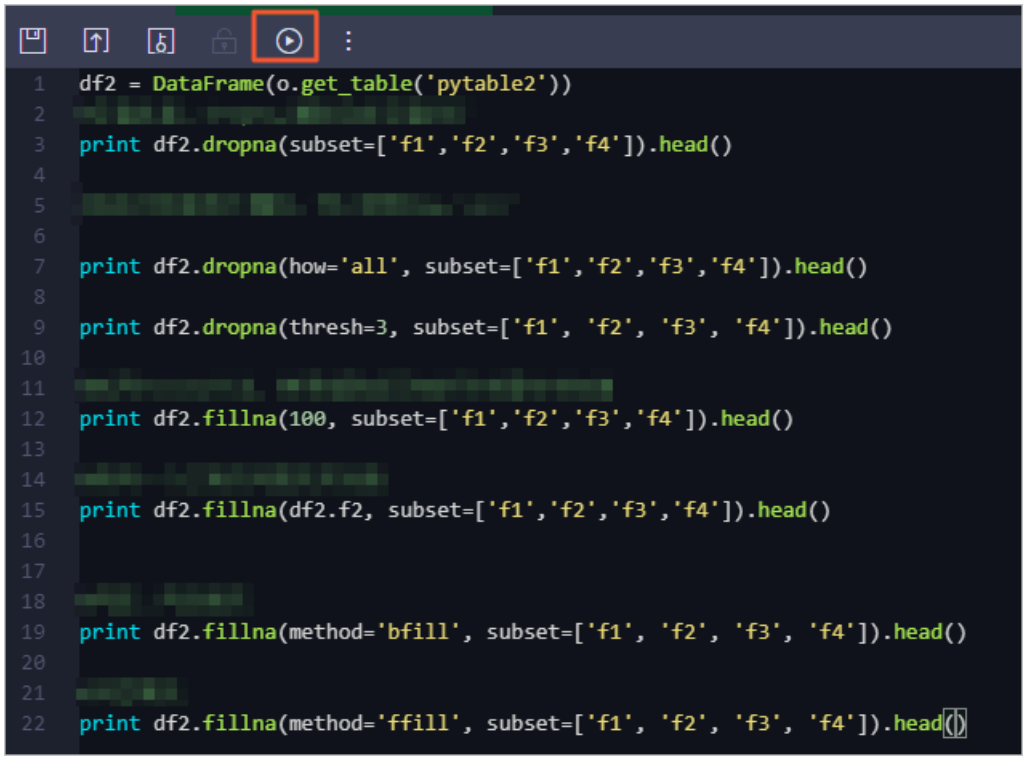
2. In the left-side navigation pane, click **Workspaces**.
3. On the page that appears, find the workspace that you want to manage and click **Data Analytics** in the Actions column.

4. On the DataStudio page, right-click a workflow and choose **New > MaxCompute > PyODPS 2**.
5. In the **New node** dialog box, specify the **Node name** parameter and click **Submit**.
6. On the configuration tab of the PyODPS 2 node, enter the code of the node in the code editor.

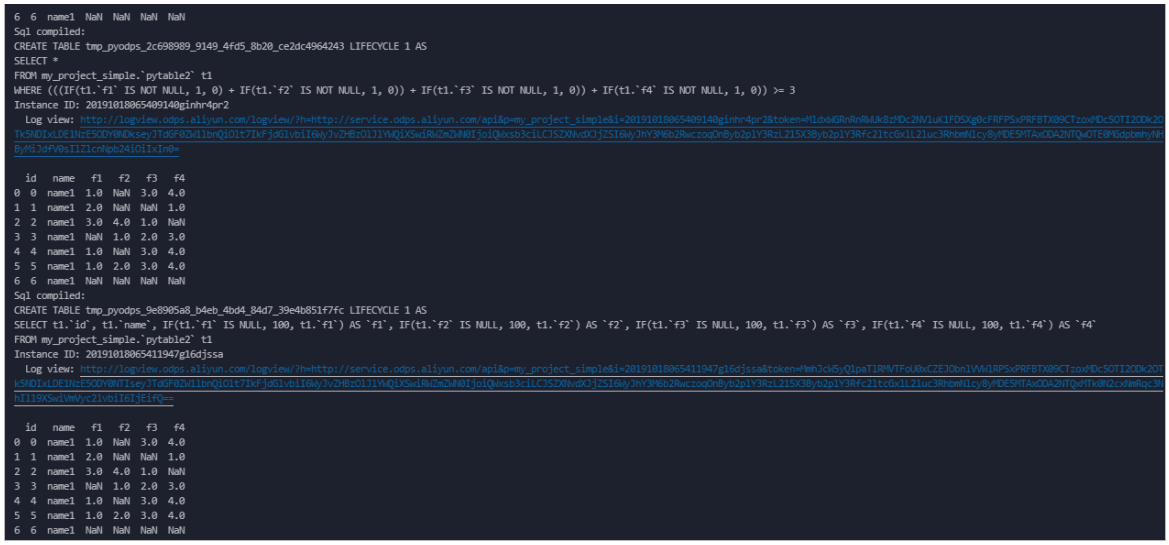
   In this example, enter the following code:

```
df2 = DataFrame(o.get_table('pytable2'))
# Call the dropna method to delete the rows that contain NULL values.
print df2.dropna(subset=['f1','f2','f3','f4']).head()
# If you do not want to delete the rows that also contain non-NULL values, specify how=
'all' for the dropna method.
print df2.dropna(how='all', subset=['f1','f2','f3','f4']).head()
print df2.dropna(thresh=3, subset=['f1', 'f2', 'f3', 'f4']).head()
# Call the fillna method to replace NULL values with a specified constant or values in
an existing column.
print df2.fillna(100, subset=['f1','f2','f3','f4']).head()
# Replace NULL values with the values in an existing column.
print df2.fillna(df2.f2, subset=['f1','f2','f3','f4']).head()
# Replace each NULL value with the value in the same column of the previous row.
print df2.fillna(method='bfill', subset=['f1', 'f2', 'f3', 'f4']).head()
# Replace each NULL value with the value in the same column of the following row.
print df2.fillna(method='ffill', subset=['f1', 'f2', 'f3', 'f4']).head()
```

7. Click the Run icon in the toolbar.



8. View the running result of the PyODPS 2 node on the **Run Log** tab.



In this example, the following information appears on the Run Log tab:

```
Sql compiled:
CREATE TABLE tmp_pyodps_d0c7d8c2_be38_4d48_b0eb_e89bae5bde01 LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pytable2` t1
WHERE (((IF(t1.`f1` IS NOT NULL, 1, 0) + IF(t1.`f2` IS NOT NULL, 1, 0)) + IF(t1.`f3` IS
NOT NULL, 1, 0)) + IF(t1.`f4` IS NOT NULL, 1, 0)) >= 4
Instance ID: 20191010071154980g2hic292
  id   name   f1   f2   f3   f4
0  0  name1  1.0  NaN  3.0  4.0
1  1  name1  2.0  NaN  NaN  1.0
2  2  name1  3.0  4.0  1.0  NaN
3  3  name1  NaN  1.0  2.0  3.0
4  4  name1  1.0  NaN  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
Sql compiled:
CREATE TABLE tmp_pyodps_49b46768_f589_48f6_be8a_b7139f31f6f2 LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pytable2` t1
WHERE (((IF(t1.`f1` IS NOT NULL, 1, 0) + IF(t1.`f2` IS NOT NULL, 1, 0)) + IF(t1.`f3` IS
NOT NULL, 1, 0)) + IF(t1.`f4` IS NOT NULL, 1, 0)) >= 1
Instance ID: 20191010071159759g0dk9592
  id   name   f1   f2   f3   f4
0  0  name1  1.0  NaN  3.0  4.0
1  1  name1  2.0  NaN  NaN  1.0
2  2  name1  3.0  4.0  1.0  NaN
3  3  name1  NaN  1.0  2.0  3.0
4  4  name1  1.0  NaN  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
Sql compiled:
CREATE TABLE tmp_pyodps_7f941800_1539_415b_9257_283ebeb893a6 LIFECYCLE 1 AS
SELECT *
FROM WB_BestPractice_dev.`pytable2` t1
WHERE (((IF(t1.`f1` IS NOT NULL, 1, 0) + IF(t1.`f2` IS NOT NULL, 1, 0)) + IF(t1.`f3` IS
NOT NULL, 1, 0)) + IF(t1.`f4` IS NOT NULL, 1, 0)) >= 3
```

```
Instance ID: 20191010071204544giyswx7
0  0  name1  1.0  NaN  3.0  4.0
1  1  name1  2.0  NaN  NaN  1.0
2  2  name1  3.0  4.0  1.0  NaN
3  3  name1  NaN  1.0  2.0  3.0
4  4  name1  1.0  NaN  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
Sql compiled:
CREATE TABLE tmp_pyodps_16d6ea6d_5195_4e4c_8346_644a395852f7 LIFECYCLE 1 AS
SELECT t1.`id`, t1.`name`, IF(t1.`f1` IS NULL, 100, t1.`f1`) AS `f1`, IF(t1.`f2` IS NUL
L, 100, t1.`f2`) AS `f2`, IF(t1.`f3` IS NULL, 100, t1.`f3`) AS `f3`, IF(t1.`f4` IS NULL
, 100, t1.`f4`) AS `f4`
FROM WB_BestPractice_dev.`pytable2` t1
Instance ID: 20191010071209190gyl56292
  id   name   f1   f2   f3   f4
0  0  name1  1.0  NaN  3.0  4.0
1  1  name1  2.0  NaN  NaN  1.0
2  2  name1  3.0  4.0  1.0  NaN
3  3  name1  NaN  1.0  2.0  3.0
4  4  name1  1.0  NaN  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
Sql compiled:
CREATE TABLE tmp_pyodps_40755ebd_2d2a_482e_b360_3f3da0d5422c LIFECYCLE 1 AS
SELECT t1.`id`, t1.`name`, IF(t1.`f1` IS NULL, t1.`f2`, t1.`f1`) AS `f1`, IF(t1.`f2` IS
NULL, t1.`f2`, t1.`f2`) AS `f2`, IF(t1.`f3` IS NULL, t1.`f2`, t1.`f3`) AS `f3`, IF(t1.`
f4` IS NULL, t1.`f2`, t1.`f4`) AS `f4`
FROM WB_BestPractice_dev.`pytable2` t1
Instance ID: 20191010071213970gbp66792
  id   name   f1   f2   f3   f4
0  0  name1  1.0  NaN  3.0  4.0
1  1  name1  2.0  NaN  NaN  1.0
2  2  name1  3.0  4.0  1.0  NaN
3  3  name1  NaN  1.0  2.0  3.0
4  4  name1  1.0  NaN  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
Sql compiled:
CREATE TABLE tmp_pyodps_d39fcce1_d8a9_4cc2_8aff_2ed1e9c6bb1b LIFECYCLE 1 AS
SELECT pyodps_udf_1570691538_d9441c59_c666_4a5d_8154_67d8bc8c24ad(t1.`id`, t1.`name`, t
1.`f1`, t1.`f2`, t1.`f3`, t1.`f4`) AS (`id`, `name`, `f1`, `f2`, `f3`, `f4`)
FROM WB_BestPractice_dev.`pytable2` t1
Instance ID: 20191010071219627goqv9292
  id   name   f1   f2   f3   f4
0  0  name1  1.0  3.0  3.0  4.0
1  1  name1  2.0  1.0  1.0  1.0
2  2  name1  3.0  4.0  1.0  NaN
3  3  name1  1.0  1.0  2.0  3.0
4  4  name1  1.0  3.0  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
Sql compiled:
CREATE TABLE tmp_pyodps_3f190cf0_f9fb_4e06_a942_ab31c0241cd3 LIFECYCLE 1 AS
```

```
SELECT pyodps_udf_1570691566_0330848b_82d3_411c_88e1_cbbcc6adb9c1(t1.`id`, t1.`name`, t
1.`f1`, t1.`f2`, t1.`f3`, t1.`f4`) AS (`id`, `name`, `f1`, `f2`, `f3`, `f4`)
FROM WB_BestPractice_dev.`pytable2` t1
Instance ID: 20191010071247729gt776792
  id   name   f1   f2   f3   f4
0  0  name1  1.0  1.0  3.0  4.0
1  1  name1  2.0  2.0  2.0  1.0
2  2  name1  3.0  4.0  1.0  1.0
3  3  name1  NaN  1.0  2.0  3.0
4  4  name1  1.0  1.0  3.0  4.0
5  5  name1  1.0  2.0  3.0  4.0
6  6  name1  NaN  NaN  NaN  NaN
```

4.1.10. FAQ about PyODPS

This topic provides answers to some frequently asked questions about PyODPS.

Category	FAQ
Install PyODPS	• What do I do if the error message "Warning: XXX not installed" appears when I install PyODPS? • What do I do if the error message "Project Not Found" appears when I install PyODPS? • What do I do if the error message "Syntax Error" appears when I install PyODPS? • What do I do if the error message "Permission Denied" appears when I install PyODPS in macOS? • What do I do if the error message "Operation Not Permitted" appears when I install PyODPS in macOS?
Import modules	• What do I do if the error message "No Module Named ODPS" appears when I run the from odps import ODPS code? • What do I do if the error message "Cannot Import Name ODPS" appears when I run the from odps import ODPS code? • What do I do if the error message "Cannot Import Module odps" appears when I run the from odps import ODPS code? • What do I do if the error message "ImportError" appears when I use PyODPS in IPython or Jupyter Notebook?
	• What does the size field represent in o.gettable('table_name').size? • How do I configure a Tunnel endpoint? • How do I use a third-party package that contains CPython in PyODPS? • What is the maximum amount of data that can be processed by a DataFrame in PyODPS? Is the size of a table limited? • How do I use max_pt in a DataFrame? • What is the difference between the open_writer() and write_table() methods when you use PyODPS to write data to a table? • Why is the amount of data that is queried on a DataWorks PyODPS node less than the amount of data that is returned in local mode?

Category	FAQ
Use PyODPS	• How do I call the count function to obtain the total number of rows in a DataFrame? • What do I do if the error message "sourceIP is not in the white list" appears when I use PyODPS? • What do I do if I fail to configure the runtime environment of MaxCompute by using from odps import options options.sql.settings? • Why does the error message "IndexError: listindexoutofrange" appears when I call the head method of a DataFrame? • What do I do if the error message "ODPSError" appears when I upload a Pandas DataFrame to MaxCompute? • What do I do if the error message "lifecycle is not specified in mandatory mode" appears when I use a DataFrame to write data to a table? • What do I do if the error message "Perhaps the datastream from server is crushed" appears when I use PyODPS to write data to a table? • What do I do if the error message "Project is protected" appears when I use PyODPS to read data from a table? • What do I do if the error message "ConnectionError: timed out try catch exception" occasionally appears when I run a PyODPS script task? • What do I do if the error message "is not defined" appears when I use PyODPS to execute the get_sql_task_cost function? • If I set options.tunnel.use_instance_tunnel to False, fields of the DATETIME type are defined in MaxCompute, but the data that is obtained by using the SELECT statements is of the STRING type. Why? • How do I use Python to achieve various purposes in PyODPS? • How do I use the Pandas DataFrame backend to debug local PyODPS programs? • What do I do if the nested loop execution is slow? • How do I prevent downloading data to a local directory? • In which scenarios can I download PyODPS data to my on-premises machine to process the data? • A maximum of 10,000 records can be obtained by using open_reader. How do I obtain more than 10,000 records? • Why am I recommended to use built-in operators instead of UDFs? • Why are the partition values of a partitioned table that is obtained by using DataFrame().schema.partitions empty? • How do I use PyODPS DataFrame to perform the Cartesian product operation? • How do I use a PyODPS node to segment Chinese text based on Jieba? • How do I use PyODPS to download full data? • Can I use execute_sql or a DataFrame to compute the percentage of null values of a field? • How do I configure data types for PyODPS?

What do I do if the error message "Warning: XXX not installed" appears when I install PyODPS?

A component is not installed when you install PyODPS. Identify the name of the component that is not installed based on XXX in the error message and run the `pip` command to install the component.

What do I do if the error message "Project Not Found" appears when I install PyODPS?

Causes

- The endpoint that you configured is invalid. You must change it to the endpoint of the project in which the PyODPS is installed. For more information about the endpoints of MaxCompute, see Endpoints.
- The positions of parameters for the MaxCompute entry object are invalid. Check the positions of the parameters for the MaxCompute entry object and make sure that you enter valid parameters. For more information about the parameters for a MaxCompute entry object, see Migrate PyODPS nodes from a data development platform to a local PyODPS environment.

What do I do if the error message "Syntax Error" appears when I install PyODPS?

The Python version is outdated. Python 2.5 or earlier is not supported. We recommend that you use the mainstream versions that are supported by PyODPS, such as Python 2.7.6 or later minor versions, Python 3.3 or later minor versions, and Python 2.6.

What do I do if the error message "Permission Denied" appears when I install PyODPS in macOS?

Run the `sudo pip install pyodps` command to install PyODPS in macOS.

What do I do if the error message "Operation Not Permitted" appears when I install PyODPS in macOS?

This issue is caused by System Integrity Protection (SIP). Restart your device and press ⌘+R during the restart. Then, run the following command in the terminal to address the issue:

```
csrutil disable
reboot
```

For more information, see Operation Not Permitted when on root - El Capitan (rootless disabled).

What do I do if the error message "No Module Named ODPS" appears when I run the from odps import ODPS code?

The PyODPS package cannot be loaded. Causes:

- Cause 1: Multiple Python versions are installed.

  Solution: In the current directory, search for the folders that are named *odps* and contain the `odps.py` or `init.py` file.

  - If an existing folder has the same name, change the folder name.

  - If you installed a Python package that is named *odps*, run the `pip uninstall odps` command to delete the package.

- Cause 2: Both Python 2 and Python 3 are installed.

  Solution: Make sure that only Python 2 or Python 3 is installed on your on-premises machine.

- Cause 3: PyODPS is not installed in the Python version that you use.

  Solution: Install PyODPS in the Python version that you use. For more information about how to install PyODPS, see Installation guide and limits.

What do I do if the error message "Cannot Import Name ODPS" appears when I run the from odps import ODPS code?

Check whether a file that is named **odps.py** exists in the current working path. If the file exists, rename the file and run the from odps import ODPS code again.

What do I do if the error message "Cannot Import Module odps" appears when I run the from odps import ODPS code?

Dependency issues occur in PyODPS. Join the DingTalk group of PyODPS technical support to fix the dependency issues. The group ID is 11701793.

What do I do if the error message "ImportError" appears when I use PyODPS in IPython or Jupyter Notebook?

Add `from odps import errors` to the header of the code.

If the error message still appears after you add `from odps import errors` to the header of the code, the IPython component is not installed. Run the `pip install -U ipython` command to install the IPython component.

What does the size field represent in o.gettable('table_name').size?

The size field specifies the physical size of a table.

How do I configure a Tunnel endpoint?

Configure the Tunnel endpoint by using the `options.tunnel.endpoint` parameter. For more information about the options.tunnel.endpoint parameter, see aliyun-odps-python-sdk.

How do I use a third-party package that contains CPython in PyODPS?

We recommend that you generate a wheel package that contains CPython. For more information, see Create a crcmod that can be used in MaxCompute.

What is the maximum amount of data that can be processed by a DataFrame in PyODPS? Is the size of a table limited?

The size of a table is not limited in PyODPS. The size of the DataFrame that is created by Pandas is limited by the size of the local memory.

How do I use max_pt in a DataFrame?

Use the `odps.df.func` module to call built-in functions of MaxCompute.

```
from odps.df import func
df = o.get_table('your_table').to_df()
df[df.ds == func.max_pt('your_project.your_table')]  # ds is a partition field.
```

What is the difference between the open_writer() and write_table() methods when you use PyODPS to write data to a table?

Each time you call the `write_table()` method, MaxCompute generates a file on the server. This operation is time-consuming. If a large number of files are generated, the efficiency of subsequent queries decreases and the memory of the server may be insufficient. We recommend that you write multiple records at the same time or provide a Generator object when you use the `write_table()` method.

By default, data is written to blocks by using the `open_writer()` method.

Why is the amount of data that is queried on a DataWorks PyODPS node less than the amount of data that is returned in local mode?

By default, Instance Tunnel is disabled in DataWorks. In this case, `instance.open_reader` is run by using the Result interface, and a maximum of 10,000 data records can be read.

After Instance Tunnel is enabled, you can execute `reader.count` to obtain the number of data records. If you need to iteratively obtain all data, you must set `options.tunnel.limit_instance_tunnel` to False to remove the limit.

How do I call the count function to obtain the total number of rows in a DataFrame?

1. After you install PyODPS, run the following command in the Python environment to create a MaxCompute table and initialize the DataFrame:

   ```
   iris = DataFrame(o.get_table('pyodps_iris'))
   ```

2. Call the count function to compute the total number of rows in the DataFrame.

   ```
   iris.count()
   ```

3. DataFrame API operations are not immediately called. These operations are called only when you explicitly call the execute method or an immediately called method. To prevent delayed execution of the count function, run the following command:

   ```
   df.count().execute()
   ```

For more information about how to obtain the total number of rows in a DataFrame, see Aggregation. For more information about the delayed execution of PyODPS methods, see Execution.

What do I do if the error message "sourceIP is not in the white list" appears when I use PyODPS?

An IP address whitelist is configured for the MaxCompute project that is accessed by PyODPS. Contact the project owner to add the IP address of your device to the IP address whitelist. For more information about how to configure IP address whitelists, see Manage IP address whitelists.

What do I do if I fail to configure the runtime environment of MaxCompute by using from odps import options options.sql.settings?

- Problem description

  When PyODPS is used to execute SQL statements, the following code is used to configure the runtime environment of MaxCompute before the request for a MaxCompute instance is sent:

  ```
  from odps import options
  options.sql.settings = {'odps.sql.mapper.split.size': 32}
  ```

  Only six mappers are enabled after the job is run. The settings do not take effect. Run the `set odps.stage.mapper.split.size=32` command on the MaxCompute client. The job is successfully run in one minute.

- Cause

  The parameters that are configured on the MaxCompute client and in PyODPS are different. The parameter on the MaxCompute client is `odps.stage.mapper.split.size`, and the parameter in PyODPS is `odps.sql.mapper.split.size`.

- Solution

  Change the parameter in PyODPS to `odps.stage.mapper.split.size`.

Why does the error message "IndexError:listindexoutofrange" appears when I call the head method of a DataFrame?

No elements exist in `list[index]` or the number of elements in `list[index]` exceeds the upper limit.

What do I do if the error message "ODPSError" appears when I upload a Pandas DataFrame to MaxCompute?

- Problem description

  The following error message appears when a Pandas DataFrame is uploaded to MaxCompute:

  ```
  ODPSError: ODPS entrance should be provided.
  ```

- Cause

  A global MaxCompute object is not found.

- Solution
  - If you use the room mechanism `%enter`, configure the global MaxCompute object.
  - Call the `to_global` method to configure the global MaxCompute object.
  - Use the `DataFrame(pd_df).persist('your_table', odps=odps)` parameter.

What do I do if the error message "lifecycle is not specified in mandatory mode" appears when I use a DataFrame to write data to a table?

- Problem description

The following error message appears when a DataFrame is used to write data to a table:

```
table lifecycle is not specified in mandatory mode
```

- Cause

  You have not configured a lifecycle for the table.

- Solution

  You must configure a lifecycle for each table in a project. Therefore, configure the following information each time you write data to a table:

```
from odps import options
options.lifecycle = 7  # Configure a lifecycle. The value must be an integer. Unit: days.
```

What do I do if the error message "Perhaps the datastream from server is crushed" appears when I use PyODPS to write data to a table?

This error is caused by dirty data. Check whether the number of columns that you specify is the same as that of the table to which you want to write data.

What do I do if the error message "Project is protected" appears when I use PyODPS to read data from a table?

The project security policy does not allow you to read data from a table. To read all data from the table, use one of the following solutions:

- Contact the project owner to add an exception policy.
- Use DataWorks or other masking tools to mask the data and export the data to a project for which data protection is not enabled before you read the data.

If you want to read some data, use one of the following solutions:

- Use `o.execute_sql('select * from <table_name>').open_reader()`.
- Use `DataFrame, o.get_table('<table_name>').to_df()`.

What do I do if the error message "ConnectionError: timed out try catch exception" occasionally appears when I run a PyODPS script task?

Causes:

- The connection timed out. The default timeout period for PyODPS is 5s. Use one of the following solutions to address the issue:
  - Add the following code to the header of the code to increase the timeout period:

```
workaround from odps import options
options.connect_timeout=30
```

  - Capture exceptions and try again.
- Some machines are not allowed to access network services due to sandbox limits. We recommend that you use exclusive resource groups for scheduling to run the script task to address this issue.

What do I do if the error message "is not defined" appears when I use PyODPS to execute the get_sql_task_cost function?

- Problem description

  The following error message appears when PyODPS is used to execute the get_sql_task_cost function:

  ```
  NameError: name 'get_task_cost' is not defined.
  ```

- Cause

  The name of the function is invalid.

- Solution

  Use the execute_sql_cost function instead of the get_sql_task_cost function.

If I set options.tunnel.use_instance_tunnel to False, fields of the DATETIME type are defined in MaxCompute, but the data that is obtained by using the SELECT statements is of the STRING type. Why?

By default, PyODPS calls the old Result interface when you call Open_Reader. In this case, the data that is obtained from the server is in the CSV format, and data of the DATETIME type is converted into the STRING type.

To address this issue, enable Instance Tunnel by setting `options.tunnel.use_instance_tunnel` to True. This way, PyODPS calls Instance Tunnel by default.

How do I use Python to achieve various purposes in PyODPS?

- Write Python functions.

  You can use multiple methods to compute the distance between two points, such as the Euclidean distance and the Manhattan distance. You can also write a series of functions and call the functions when you compute data based on your business requirements.

  ```
  def euclidean_distance(from_x, from_y, to_x, to_y):
      return ((from_x - to_x) ** 2 + (from_y - to_y) ** 2).sqrt()
  def manhattan_distance(center_x, center_y, x, y):
     return (from_x - to_x).abs() + (from_y - to_y).abs()
  ```

  The following sample code shows how to call a function that you write:

```
In [42]: df
     from_x    from_y      to_x      to_y
0  0.393094  0.427736  0.463035  0.105007
1  0.629571  0.364047  0.972390  0.081533
2  0.460626  0.530383  0.443177  0.706774
3  0.647776  0.192169  0.244621  0.447979
4  0.846044  0.153819  0.873813  0.257627
5  0.702269  0.363977  0.440960  0.639756
6  0.596976  0.978124  0.669283  0.936233
7  0.376831  0.461660  0.707208  0.216863
8  0.632239  0.519418  0.881574  0.972641
9  0.071466  0.294414  0.012949  0.368514
In [43]: euclidean_distance(df.from_x, df.from_y, df.to_x, df.to_y).rename('distance')
   distance
0  0.330221
1  0.444229
2  0.177253
3  0.477465
4  0.107458
5  0.379916
6  0.083565
7  0.411187
8  0.517280
9  0.094420
In [44]: manhattan_distance(df.from_x, df.from_y, df.to_x, df.to_y).rename('distance')
   distance
0  0.392670
1  0.625334
2  0.193841
3  0.658966
4  0.131577
5  0.537088
6  0.114198
7  0.575175
8  0.702558
9  0.132617
```

- Use conditions and loop statements in Python.

  If the tables that you want to compute are stored in a database, you must process the fields of the tables based on configurations and perform UNION or JOIN operations on the tables. If you use SQL statements to perform these operations, the process is complex. We recommend that you use a DataFrame to perform the operations.

  For example, if you want to merge 30 tables into one table, you must perform the UNION ALL operation on the 30 tables if you use SQL statements. If you use PyODPS, run the following code:

```
table_names = ['table1', ..., 'tableN']
dfs = [o.get_table(tn).to_df() for tn in table_names]
reduce(lambda x, y: x.union(y), dfs)
## The reduce statement is equivalent to the following code:
df = dfs[0]
for other_df in dfs[1:]:
    df = df.union(other_df)
```

How do I use the Pandas DataFrame backend to debug local PyODPS programs?

You can use one of the following methods to debug local PyODPS programs: The initialization methods are different in the following methods but the subsequent code is the same.

- A PyODPS DataFrame that is created by using the Pandas DataFrame can use Pandas to debug local PyODPS programs.
- A DataFrame that is created by using MaxCompute tables can be executed in MaxCompute.

Sample code:

```
df = o.get_table('movielens_ratings').to_df()
DEBUG = True
if DEBUG:
    df = df[:100].to_pandas(wrap=True)
```

If all subsequent code is written, the local debugging speed is fast. After the debugging is complete, you can change the value of the DEBUG parameter to False. Then, you can compute all data in MaxCompute.

We recommend that you use MaxCompute Studio to debug local PyODPS programs.

What do I do if the nested loop execution is slow?

We recommend that you use the Dict data structure to obtain the execution result of the loop and import the execution result to DataFrame objects in the outer loop. If you place the DataFrame object code `df=XXX` in the outer loop, a DataFrame object is generated for each loop calculation. As a result, the execution speed of the nested loop is slow.

How do I prevent downloading data to a local directory?

For more information, see Use a PyODPS node to download data to a local directory for processing or to process data online.

In which scenarios can I download PyODPS data to my on-premises machine to process the data?

You can download PyODPS data to your on-premises machine in one of the following scenarios:

- A small amount of PyODPS data needs to be processed.
- If you need to apply a Python function to a single row of data or perform operations that change one row to multiple rows, you can use the PyODPS DataFrame and make full use of the parallel computing capabilities of MaxCompute.

  For example, if you want to expand a JSON string data into one row by using key-value pairs, run the following code:

```
In [12]: df
                json
0  {"a": 1, "b": 2}
1  {"c": 4, "b": 3}
In [14]: from odps.df import output
In [16]: @output(['k', 'v'], ['string', 'int'])
    ...: def h(row):
    ...:     import json
    ...:     for k, v in json.loads(row.json).items():
    ...:         yield k, v
    ...:
In [21]: df.apply(h, axis=1)
   k  v
0  a  1
1  b  2
2  c  4
3  b  3
```

A maximum of 10,000 records can be obtained by using open_reader. How do I obtain more than 10,000 records?

Use `create table as select ...` to save the SQL execution result as a table, and use `table.open_reader` to read data.

Why am I recommended to use built-in operators instead of UDFs?

UDFs are executed slower than built-in operators during calculation. Therefore, we recommend that you use built-in operators.

If you need to process millions of rows of data and you apply a UDF to a row, the execution time is increased from 7 seconds to 27 seconds. If larger datasets or more complex operations are required, the gap in time may be larger.

Why are the partition values of a partitioned table that is obtained by using DataFrame().schema.partitions empty?

A DataFrame does not distinguish between partition fields and common fields. Therefore, partition fields are processed as common fields. You can filter out partition fields by using the following method:

```
df = o.get_table().to_df()
df[df.ds == '']
```

For more information about how to configure partitions or read data from partitions, see Tables.

How do I use PyODPS DataFrame to perform the Cartesian product operation?

For more information, see Use PyODPS DataFrame to process Cartesian products.

How do I use a PyODPS node to segment Chinese text based on Jieba?

For more information, see Use a PyODPS node to segment Chinese text based on Jieba.

How do I use PyODPS to download full data?

By default, PyODPS does not limit the amount of data that can be read from an instance. However, the amount of data that can be downloaded for a protected project by using Tunnel commands is limited. If you do not specify `options.tunnel.limit_instance_tunnel`, the limit is automatically enabled, and the number of data records that can be downloaded is limited based on the configurations of the MaxCompute project. In most cases, a maximum of 10,000 data records can be downloaded at a time. If you need to iteratively obtain all data, you must disable the `limit` on the amount of data. You can execute the following statements to enable `Instance Tunnel` and disable the `limit`:

```
options.tunnel.use_instance_tunnel = True
options.tunnel.limit_instance_tunnel = False  # Disable the limit on the amount of data and
read all data.
with instance.open_reader() as reader:
    # Use Instance Tunnel to read all data.
```

Can I use execute_sql or a DataFrame to compute the percentage of null values of a field?

We recommend that you use a DataFrame to perform aggregate operations due to its high aggregate performance.

How do I configure data types for PyODPS?

If you use PyODPS, you can use one of the following methods to enable the new data types that are supported by MaxCompute V2.0 data type edition:

- Execute `o.execute_sql('set odps.sql.type.system.odps2=true;query_sql', hints={"odps.sql.submit.mode" : "script"})` to enable the new data types that are supported by MaxCompute V2.0 data type edition.
- Enable the new data types that are supported by MaxCompute V2.0 data type edition by using a DataFrame. For example, use an executed action, such as persist, execute, or to_pandas, by configuring the hints parameter. The following configurations are valid only for a single job:

  ```
  from odps.df import DataFrame
  users - DataFrame(o.get_table('odps2_test'))
  users.persist('copy_test',hints={'odps.sql.type.system.odps2':'true'})
  ```

  If you want to enable the new data types that are supported by MaxCompute V2.0 data type edition by using a DataFrame and you want the setting to take effect globally, set `options.sql.use_odps2_extension` to True.

4.2. MapReduce

4.2.1. Summary

4.2.1.1. Overview

This topic describes the MapReduce API supported by MaxCompute and its limits.

MaxCompute provides three versions of the MapReduce API:

- MaxCompute MapReduce: the native MapReduce API. This version runs fast. It is convenient to develop a program without the need to expose file systems.
- Extended MaxCompute MapReduce (MR2): an extension of MaxCompute MapReduce. This version supports complex job scheduling logic. The implementation method is the same as that of MaxCompute MapReduce.
- Hadoop-compatible MapReduce: This version is highly compatible with Hadoop MapReduce but it is incompatible with MR2.

The preceding three versions are similar in basic terms, job submission, input and output, and resource usage. The only difference is in SDK for Java. This topic describes the basic principles of MapReduce. For more information, see Hadoop MapReduce.

> **Note** You cannot use MapReduce to read data from or write data to external tables.

Scenarios

MapReduce supports the following scenarios:

- Search: web crawl, inverted index, PageRank.
- Analysis of web access logs:
  - Analyze and summarize the characteristics of user behavior, such as web browsing and online shopping. The analysis can be used to deliver personalized recommendations.
  - Analyze user access behavior.
- Statistical analysis of texts:
  - Word count and term frequency-inverse document frequency (TFIDF) analysis of popular novels.
  - Statistical analysis of references to academic papers and patent documents.
  - Wikipedia data analysis.
- Mining of large amounts of data: mining of unstructured data, spatio-temporal data, and image data.
- Machine learning: supervised learning, unsupervised learning, and classification algorithms, such as decision trees and support vector machines (SVMs).
- Natural language processing (NLP):
  - Training and forecast based on big data.
  - Construction of a co-occurrence matrix, mining of frequent itemset data, and duplicate document detection based on existing libraries.
- Advertisement recommendations: forecast of the click-through rate (CTR) and conversion rate (CVR).

Process

A MapReduce program processes data in two stages: the map and reduce stages. It executes the map stage before the reduce stage. You can specify the processing logic of the map and reduce stages. However, the logic must comply with the conventions of the MapReduce framework. The following procedure shows how the MapReduce framework processes data:

1. The MapReduce framework partitions data and uses data in each partition as the input for a mapper. After data is partitioned, multiple mappers start to process the data at the same time.

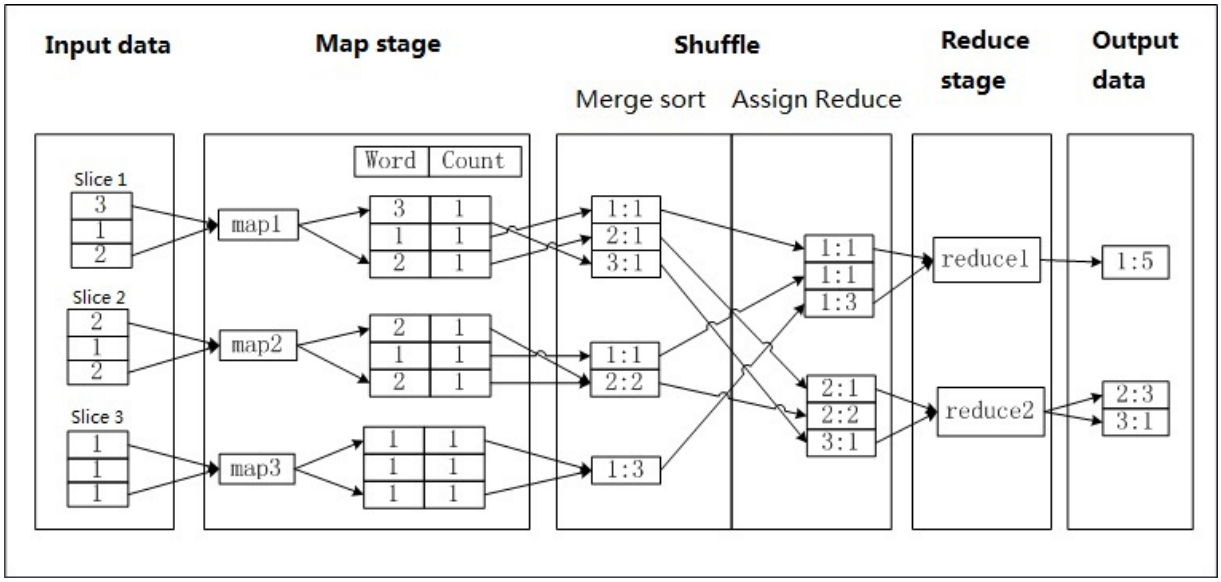
Before the map operation, the input data must be partitioned. Partitioning refers to the splitting of the input data into data blocks of the same size. Each data block is processed as the input for a single mapper. This allows you to use multiple mappers at the same time.

2. Each mapper reads its partition data, computes the data, and generates data records to a reducer. When a mapper generates data records, it must specify a key for each data record. A key specifies the reducer to which a data record is sent. Keys and reducers share a many-to-one relationship. Data records with the same key are sent to the same reducer. A reducer may receive data records with different keys.
3. Before the reduce stage, the MapReduce framework sorts data records based on keys to ensure that data records with the same key are adjacent. If you specify a **combiner**, the MapReduce framework calls the combiner to combine data records that share the same key. You can define the logic of the combiner. Different from the classic MapReduce framework protocol, MaxCompute requires that the input and output parameters of a combiner must be consistent with those of a reducer. This process is called **shuffle**.
4. At the reduce stage, data records with the same key are transferred to the same reducer. A single reducer may receive data records from multiple mappers. Each reducer performs the reduce operation on multiple data records with the same key. After the reduce operation, all data records with the same key are converted into a single value.
5. The results are generated.

> **Note** For more information about the MapReduce framework, see Features.

The following section uses WordCount as an example to explain the related concepts of MaxCompute MapReduce at different stages.

Assume that a file named a.txt exists and each line of the file contains a digit. You want to count the number of times each digit appears. Each digit is called a word, and the number of times it is used represents the count. The following figure shows how MaxCompute MapReduce counts the words.



Procedure

1. MaxCompute MapReduce partitions data in the a.txt file and uses data in each partition as the input for a mapper.
2. The mapper processes input data and records the value of the Count parameter as 1 for each

obtained digit. This way, a <Word, Count> pair is generated. The value of the Word parameter is used as the key for the newly generated pair.

3. At the early shuffle stage, the data records generated by each mapper are sorted based on keys (the value of the Word parameter). After the data records are sorted, the records are combined. This requires that you accumulate the Count values that share the same key to generate a new <Word, Count> pair. This is a merging and sorting process.
4. At the late shuffle stage, data records are transferred to reducers. The reducers sort the received data records based on the keys.
5. Each reducer uses the same logic as the combiner to process data. Each reducer accumulates the Count values with the same key (the value of the Word parameter).
6. The results are generated.

> **Note** All the MaxCompute data is stored in tables. Therefore, the input and output of MaxCompute MapReduce can be only in the table format. You cannot specify the output format, and no interfaces similar to file systems are provided.

Limits

- For more information about limits on MaxCompute MapReduce, see Limits on MaxCompute MapReduce.
- For more information about the limits on the running of MaxCompute MapReduce in local mode, see Local run.

4.2.1.2. Extended MapReduce model

The extended MapReduce model (MR2) that is provided by MaxCompute uses optimized scheduling and I/O models to reduce unnecessary I/O operations during job running.

In the extended MapReduce model, Map and Reduce functions are written in the same way as MaxCompute. The difference lies in how the jobs run. For more information, see Pipeline examples.

Background information

A MapReduce model consists of multiple MapReduce jobs. In a traditional MapReduce model, the output of each MapReduce job must be written to a disk in a distributed file system such as HDFS or to a MaxCompute table. However, subsequent Map tasks may only need to read the outputs once to prepare for the Shuffle stage. This mechanism results in redundant I/O operations.

The computing and scheduling logic of MaxCompute supports more complex programming models. A Map operation can be succeeded by any number of consecutive Reduce operations without the need for a Map operation in between, such as Map-Reduce-Reduce.

Comparison with Hadoop ChainMapper and ChainReducer

Similarly, Hadoop ChainMapper and ChainReducer also support serialized Map or Reduce operations. However, they are essentially different from the extended MapReduce model.

Hadoop ChainMapper and ChainReducer are based on the traditional MapReduce model. They support only Map operations after the original Map or Reduce operation. One benefit is that you can reuse the Mapper business logic to split a Map or Reduce operation into multiple Mapper stages. However, this does not modify the scheduling or I/O models.

4.2.1.3. Open source MapReduce

This topic describes the background of the compatibility with open source MapReduce and shows you how to use the Hadoop MapReduce plug-in.

Background

MaxCompute provides native MapReduce with a set of programming models and operations. The input and output of the operations are data in MaxCompute tables. The data is organized in the record format, which can demonstrate how the data is processed.

The operations of MaxCompute MapReduce differ from those of Hadoop MapReduce. To migrate Hadoop MapReduce jobs to MaxCompute, you must rewrite the MapReduce code, compile and debug the code by calling MaxCompute operations, compress the final code into a JAR package, and upload the package to the MaxCompute platform. This process is tedious and labor-intensive for development and testing. An ideal solution is that the Hadoop MapReduce code can be run in MaxCompute with little or no modification at all.

To achieve the ideal solution, MaxCompute provides a plug-in to adapt Hadoop MapReduce to MaxCompute MapReduce. The plug-in enables Hadoop MapReduce jobs to be compatible with MaxCompute at the binary level. You can set configurations without the need to modify the code. Then, run the original Hadoop MapReduce JAR packages on MaxCompute. The plug-in is in the test phase and does not support custom comparators or key types.

In the following example, the WordCount program is used to introduce the basic usage of the plug-in.

Download the Hadoop MapReduce plug-in

Download the Hadoop MapReduce plug-in package named *openmr_hadoop2openmr-1.0.jar*.

> **Note** The openmr_hadoop2openmr-1.0.jar package contains the dependencies of Hadoop 2.7.2. To avoid version conflicts, do not include Hadoop dependencies in the JAR packages of your jobs.

Prepare a JAR package for WordCount

Compile and export a JAR package named wordcount_test.jar with the following source code of WordCount:

```
package com.aliyun.odps.mapred.example.hadoop;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
import java.util.StringTokenizer;
public class WordCount {
    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{
```

```
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
    public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values,
            Context context
        ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Prepare test data

1. Execute the following statements to create the input table wc_in and the output table wc_out:

   ```
   create table if not exists wc_in(line string);
   create table if not exists wc_out(key string, cnt bigint);
   ```

2. Run the Tunnel upload command to import sample data to the input table wc_in.

   Import the following data of the data.txt file to wc_in:

   ```
   hello maxcompute
   hello mapreduce
   ```

Run the following command on the MaxCompute client to import the preceding data from data.txt to wc_in:

```
tunnel upload data.txt wc_in;
```

Configure the mapping between HDFS file paths and MaxCompute tables

Configure the mapping between Hadoop Distributed File System (HDFS) file paths and MaxCompute tables in the *wordcount-table-res.conf* file. The following code is an example of the file:

```
{
  "file:/foo": {
    "resolver": {
      "resolver": "com.aliyun.odps.mapred.hadoop2openmr.resolver.TextFileResolver",
      "properties": {
          "text.resolver.columns.combine.enable": "true",
          "text.resolver.seperator": "\t"
      }
    },
    "tableInfos": [
      {
        "tblName": "wc_in",
        "partSpec": {},
        "label": "__default__"
      }
    ],
    "matchMode": "exact"
  },
  "file:/bar": {
    "resolver": {
      "resolver": "com.aliyun.odps.mapred.hadoop2openmr.resolver.BinaryFileResolver",
      "properties": {
          "binary.resolver.input.key.class" : "org.apache.hadoop.io.Text",
          "binary.resolver.input.value.class" : "org.apache.hadoop.io.LongWritable"
      }
    },
    "tableInfos": [
      {
        "tblName": "wc_out",
        "partSpec": {},
        "label": "__default__"
      }
    ],
    "matchMode": "fuzzy"
  }
}
```

Set the following parameters in the preceding configuration:

The preceding configuration is a JSON file that describes the mapping between HDFS file paths and MaxCompute tables. You must configure both the input and output. Each HDFS file path matches three configuration items: resolver, tableInfos, and matchMode.

- resolver: specifies how to process data in the specified files. The following two built-in resolvers are available: com.aliyun.odps.mapred.hadoop2openmr.resolver.TextFileResolver and com.aliyun.odps.mapred.hadoop2openmr.resolver.BinaryFileResolver. After you specify the resolver, you must configure the required properties for the resolver to parse data.
  - TextFileResolver: regards the input or output as plaintext if the data is of the plaintext type. When you configure an input resolver, you must configure the text.resolver.columns.combine.enable and text.resolver.seperator properties. If you set text.resolver.columns.combine.enable to true, all columns in the input table are combined into a single string based on the delimiter specified by text.resolver.seperator. Otherwise, the first two columns in the input table are used as the key and value fields.
  - BinaryFileResolver: converts binary data into a data type that is supported by MaxCompute, such as BIGINT, BOOLEAN, or DOUBLE. When you configure an output resolver, you must configure the binary.resolver.input.key.class and binary.resolver.input.value.class properties. binary.resolver.input.key.class specifies the key type of the intermediate result, and binary.resolver.input.value.class specifies the value type.
- tableInfos: specifies the MaxCompute table that corresponds to the specified HDFS file path. You can set only the tblName parameter. The values of the partSpec and label parameters must be the same as those in the preceding sample configuration.
- matchMode: specifies the path matching mode. The valid values are exact and fuzzy. If you set this parameter to fuzzy, you can use a regular expression to match the HDFS input path in fuzzy mode.

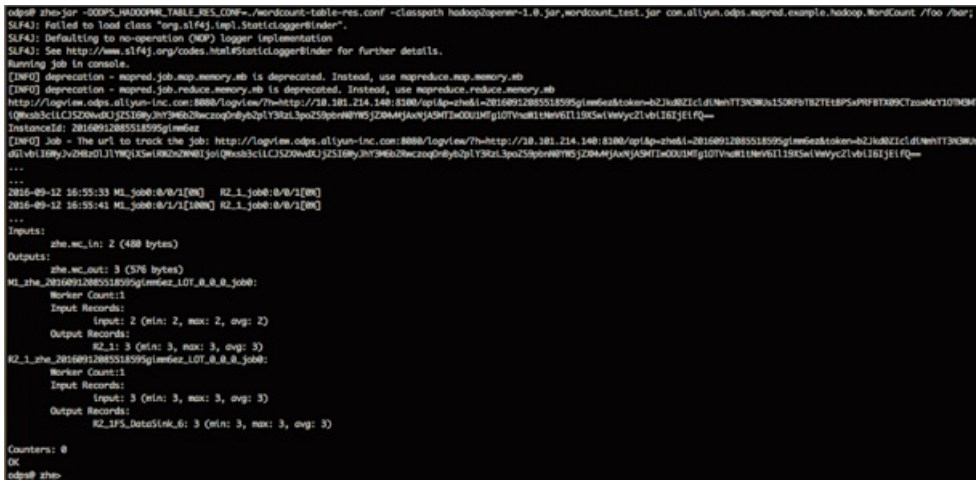
Submit a job

Run the following command on the MaxCompute client to submit a job. For more information about how to install and configure the MaxCompute client, see Client.

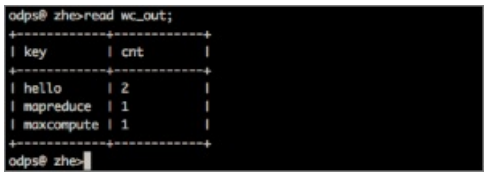
```
jar -DODPS_HADOOPMR_TABLE_RES_CONF=./wordcount-table-res.conf -classpath hadoop2openmr-1.0.
jar,wordcount_test.jar com.aliyun.odps.mapred.example.hadoop.WordCount /foo/bar;
```

> **Note**
>
> - wordcount-table-res.conf: the mapped configuration file configured with /foo/bar.
> - wordcount_test.jar: the JAR package of your Hadoop MapReduce program.
> - com.aliyun.odps.mapred.example.hadoop.WordCount: the class name of the job that you want to run.
> - /foo/bar: the path on HDFS, which is mapped to wc_in and wc_out in the JSON configuration file.
> - After you configure the mapping, you must use the Data Integration service of DataWorks to import the HDFS input file to wc_in for MapReduce computing, and export the result table wc_out to your HDFS output directory /bar.
> - Before you run the preceding command, make sure that hadoop2openmr-1.0.jar, wordcount_test.jar, and wordcount-table-res.conf have been stored in the current directory of the MaxCompute client. Otherwise, modify the configuration and the -classpath parameter in the preceding command as needed.

The following figure shows the running process.



After the job is run, you can view the result table wc_out to check whether the job is successful and whether the results meet expectations.



4.2.2. Limits

This topic describes the limits of MaxCompute MapReduce. Your business may be affected if these limits are violated.

The following table describes the limits of MaxCompute MapReduce.

Item	Value range	Classification	Configuration item	Default value	Configurable	Description
Memory occupied by an instance	[256 MB,12 GB]	Memory	`odps.stage.mapper(reducer).mem` and `odps.stage.mapper(reducer).jvm.mem`	2,048 MB and 1,024 MB	Yes	The memory occupied by a single map or reduce instance. The memory consists of two parts: the framework memory, which is 2,048 MB by default, and Java Virtual Machine (JVM) heap memory, which is 1,024 MB by default.
Number of resources	256	Quantity	-	N/A	No	Each job can reference up to 256 resources. Each table or archive is considered as one resource.

Item	Value range	Classification	Configuration item	Default value	Configurable	Description
Numbers of inputs and outputs	1,024 and 256	Quantity	-	N/A	No	The number of the inputs of a job cannot exceed 1,024, and that of the outputs of a job cannot exceed 256. A partition of a table is regarded as one input. The number of tables cannot exceed 64.
Number of counters	64	Quantity	-	N/A	No	The number of custom counters in a job cannot exceed 64. The counter group name and counter name cannot contain number signs (#). The total length of the two names cannot exceed 100 characters.
Number of map instances	[1,100000]	Quantity	odps.stage.mapper.num	N/A	Yes	The number of map instances in a job is calculated by the framework based on the split size. If no input table is specified, you can set the odps.stage.mapper.num parameter to specify the number of map instances. The value ranges from 1 to 100,000.
Number of reduce instances	[0,2000]	Quantity	odps.stage.reducer.num	N/A	Yes	By default, the number of reduce instances in a job is 25% of the number of map instances. You can set the number to a value that ranges from 0 to 2,000. Reduce instances process much more data than map instances, which may result in long processing time in the reduce stage. A job can have 2,000 reduce instances at most.
Number of retries	3	Quantity	-	N/A	No	The maximum number of retries that are allowed for a map or reduce instance is 3. Exceptions that do not allow retries may cause jobs to fail.

Item	Value range	Classification	Configuration item	Default value	Configurable	Description
Local debug mode	A maximum of 100 instances	Quantity	-	N/A	No	In local debug mode: • The number of map instances is 2 by default and cannot exceed 100. • The number of reduce instances is 1 by default and cannot exceed 100. • The number of download records for one input is 100 by default and cannot exceed 10,000.
Number of times a resource is read repeatedly	64	Quantity	-	N/A	No	The number of times that a map or reduce instance repeatedly reads a resource cannot exceed 64.
Resource bytes	2 GB	Length	-	N/A	No	The total bytes of resources that are referenced by a job cannot exceed 2 GB.
Split size	Greater than or equal to 1	Length	odps.stage.mapper.split.size	256 MB	Yes	The framework determines the number of map instances based on the split size.
Length of a string in a column	8 MB	Length	-	N/A	No	A string in a column cannot exceed 8 MB in length.
Worker timeout period	[1,3600]	Time	odps.function.timeout	600	Yes	The timeout period of a map or reduce worker when the worker does not read or write data, or stops sending heartbeats by using `context.progress()`. The default value is 600 seconds.

Item	Value range	Classification	Configuration item	Default value	Configurable	Description
Field types supported by tables that are referenced by MapReduce	BIGINT, DOUBLE, STRING, DATETIME, and BOOLEAN	Data type	-	N/A	No	When a MapReduce task references a table, an error is returned if the table has field types that are not supported.
Object Storage Service (OSS) data read	-	Feature	-	N/A	No	MapReduce cannot read OSS data.
New data types in MaxCompute V2.0	-	Feature	-	N/A	No	MapReduce does not support the new data types in MaxCompute V2.0.

4.2.3. Quick start

This topic describes how to write a MapReduce program by using MaxCompute Studio, generate a JAR file, and then run a MapReduce job on the MaxCompute client. A WordCount MapReduce job is used in this topic.

Prerequisites

Make sure that the following requirements are met:

- The MaxCompute client is installed and configured.

  For more information about how to install and configure the MaxCompute client, see Install and configure the MaxCompute client.

- MaxCompute Studio is installed and connected to the MaxCompute project that you want to use.

  For more information about how to install MaxCompute Studio and connect it to a MaxCompute project, see Install MaxCompute Studio and Manage project connections.

- The source data file is prepared and saved to your on-premises machine.

  The sample file in this topic is data.txt, whose content is `hello,odps`. You can prepare such a file and save it to the `bin` directory of the MaxCompute client.

Usage notes

If you want to use Maven to develop a MapReduce program, you can search for odps-sdk-mapred, odps-sdk-commons, and odps-sdk-core in the Maven Central Repository to obtain different versions of SDK for Java. The following dependencies must be configured in the pom.xml file:

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-mapred</artifactId>
    <version>0.36.4-public</version>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-commons</artifactId>
    <version>0.36.4-public</version>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-core</artifactId>
    <version>0.36.4-public</version>
</dependency>
```

Procedure

1. Step 1: Develop a MapReduce program

   Write, run, and debug a MapReduce program by using MaxCompute Studio.

2. Step 2: Generate and upload a MapReduce JAR file

   Package the compiled WordCount.java script into a JAR file and upload the file to the MaxCompute project.

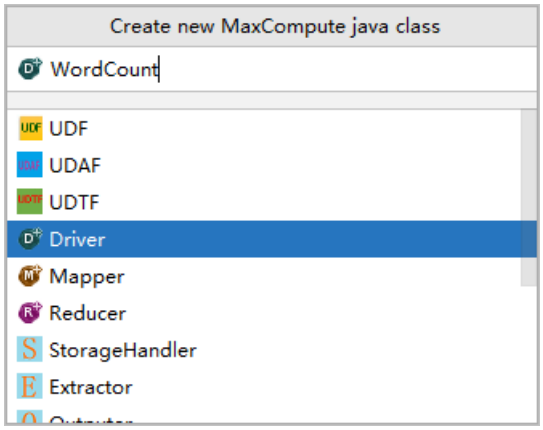
3. Step 3: Run a MapReduce job

   Run the `JAR` command based on the JAR file you uploaded to your MaxCompute project to run a MapReduce job.

Step 1: Develop a MapReduce program

1. Create a MaxCompute Java module.
   i. Start IntelliJ IDEA. In the top navigation bar, choose **File > New > Module**.
   ii. In the left-side navigation pane of the **New Module** dialog box, click **MaxCompute Java**.
   iii. Configure **Module SDK** and click **Next**.
   iv. Enter a module name, such as mapreduce, and click **Finish** in the **Module name** field.
2. Write, run, and debug a WordCount MapReduce program.
   i. In the **Project** pane, expand your MaxCompute Java module and choose **src > main > java**. Then, right-click java and choose **New > MaxCompute Java**.

ii. In the **Create new MaxCompute java class** dialog box, click **Driver**, enter the name of the MaxCompute Java class that you want to create in the **Name** field, and then press Enter. In this example, the name of the MaxCompute Java class is WordCount.

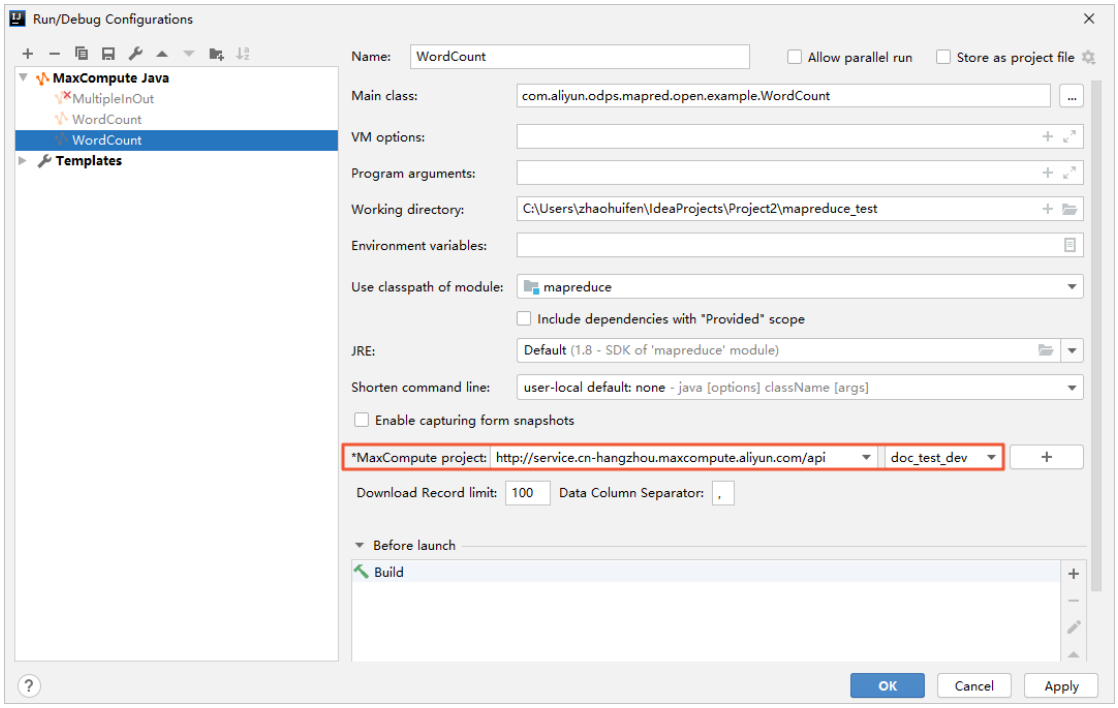


iii. In the code editor for WordCount.java, write a WordCount MapReduce program to count the number of words.

For the complete WordCount sample code, see Sample code.

iv. In the left-side navigation pane, right-click WordCount.java and select **Run**.

v. In the **Run/Debug Configurations** dialog box, set **MaxCompute project** to the MaxCompute project that you want to use.

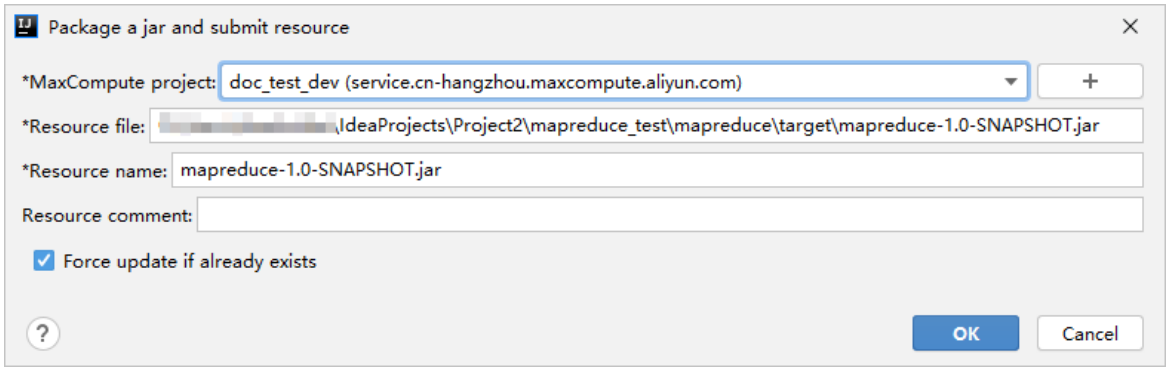


vi. Click **OK** to run and debug the WordCount.java script. Make sure that the script is compiled as expected.

Step 2: Generate and upload a MapReduce JAR file

1. In the left-side navigation pane of IntelliJ IDEA, right-click WordCount.java and select **Deploy to server**.
2. In the **Package a jar and submit resource** dialog box, configure the parameters and click **OK** to

package and upload the script.



For more information about the parameters, see Procedure.

> **Note** If you use Maven to develop the MapReduce program, you must manually upload the JAR file to your MaxCompute project from the MaxCompute client after you package the script into a JAR file. For more information about how to upload a JAR file, see Add resources. Sample statement:
>
> ```
> add jar mapreduce-1.0-SNAPSHOT.jar;
> ```

Step 3: Run a MapReduce job

1. Log on to the MaxCompute client or start the MaxCompute client in MaxCompute Studio.

   The MaxCompute client is integrated in MaxCompute Studio. You can run the MaxCompute client in MaxCompute Studio. For more information, see Integrate with MaxCompute client.

2. Create input and output tables.

   An input table contains the source data of a MapReduce job. An output table contains a processing results of the MapReduce job. Sample statement:

   ```
   --Create an input table named wc_in.
   create table wc_in (key STRING, value STRING);
   --Create an output table named wc_out.
   create table wc_out (key STRING, cnt BIGINT);
   ```

   For more information about the CREATE TABLE syntax, see Create a table.

3. Run the Tunnel Upload command to insert data into the wc_in table.

   Sample statement:

   ```
   tunnel upload data.txt wc_in;
   ```

   For more information about Tunnel commands, see Tunnel commands.

4. Run the `JAR` command to call the uploaded JAR file and run a MapReduce job.

   Sample statement:

   ```
   jar -resources mapreduce-1.0-SNAPSHOT.jar -classpath mapreduce-1.0-SNAPSHOT.jar com.ali
   yun.odps.mapred.open.example.WordCount wc_in wc_out;
   ```

   - `-resources mapreduce-1.0-SNAPSHOT.jar`: The `-resources` option specifies the name of the resource that is called by the MapReduce job. In this example, the resource is the mapreduce-

1.0-SNAPSHOT.jar file that is uploaded in Step 2.

- `-classpath mapreduce-1.0-SNAPSHOT.jar`: The `-classpath` option specifies the path of the JAR file that contains MainClass.
- `com.aliyun.odps.mapred.open.example.WordCount`: MainClass defined in the MapReduce program.
- `wc_in wc_out`: the input table and output table.

For more information about the `JAR` command, see Syntax.

5. Run the following command to view the result data that is written to the wc_out table:

```
select * from wc_out;
```

Command output:

```
+------------+------------+
| key        | cnt        |
+------------+------------+
| hello      | 1          |
| odps       | 1          |
+------------+------------+
```

4.2.4. Function Introduction

4.2.4.1. Terms

This topic introduces the basic terms of MapReduce.

Map/Reduce

When a map or reduce task runs, the setup(), map() or reduce(), and cleanup() methods are called. The setup() method is called prior to the map() or reduce() method. Each worker calls it only once.

The cleanup() method is called after the map() or reduce() method. Each worker calls it only once.

For more information about usage examples, see Example programs.

Sort

Some columns in the key records generated by a mapper can be used as sort columns. These columns do not support a custom comparator. You can select a few sort columns as group columns. These columns do not support a custom group comparator. Sort columns are used to sort your data, while group columns are used for secondary sorting.

For more information about usage examples, see Secondary sorting source code.

Partition

MaxCompute supports partition columns and custom partitioners. Partition columns take precedence over custom partitioners.

Partitioners are used to allocate the data generated by a mapper to different reducers based on the partitioning logic.

Combiner

The combiner function combines adjacent records at the shuffle stage. You can determine whether to use the combiner function based on your business logic.

The combiner function is the optimization of the MapReduce computing framework. The combiner logic is the same as the reducer logic. After a mapper generates data, the framework combines the data with the same key at the map stage.

For more information about usage examples, see Example programs.

4.2.4.2. Submit a MapReduce job

This topic describes how to run a JAR command on the MaxCompute client to submit a MapReduce job.

The MaxCompute client provides a JAR command to submit MapReduce jobs. Example:

```
jar -conf \home\admin\myconf -resources a.txt,example.jar -classpath ..\lib\example.jar:.\o
ther_lib.jar -D java.library.path=.\native;
```

Syntax

```
jar [<GENERIC_OPTIONS>] <MAIN_CLASS> [ARGS];
        -conf <configuration_file>         Specify an application configuration file
        -resources <resource_name_list>    file\table resources used in mapper or reducer,
seperate by comma
        -classpath <local_file_list>       classpaths used to run mainClass
        -D <name>=<value>                  Property value pair, which will be used to run m
ainClass
        -l                                 Run job in local mode
```

Parameters

`<GENERIC_OPTIONS>` includes the following optional parameters:

- -conf <configuration file>: specifies the JobConf configuration file. This file contains the settings of JobConf in SDK.

  The following code shows the template of a JobConf file:

  ```
  <configuration>
         <property>
            <name>import.filename</name>
            <value>resource.txt</value>
         </property>
      </configuration>
  ```

  In this example, a variable named import.filename whose value is resource.txt is specified in the JobConf configuration file.

  You can call the JobConf interface in MapReduce to obtain the value of this variable. You can also call the JobConf interface in SDK to obtain the variable value. For more information, see Resource samples.

  Example:

```
add jar data\mapreduce-examples.jar;
    jar -resources mapreduce-examples.jar -classpath data\mapreduce-examples.jar
        org.alidata.odps.mr.examples.WordCount wc_in wc_out;
    add file data\src.txt;
    add jar data\mapreduce-examples.jar;
    jar -resources src.txt,mapreduce-examples.jar -classpath data\mapreduce-examples.jar
        org.alidata.odps.mr.examples.WordCount wc_in wc_out;
    add file data\a.txt;
    add table wc_in as test_table;
    add jar data\work.jar;
    jar -conf odps-mapred.xml -resources a.txt,test_table,work.jar
        -classpath data\work.jar:otherlib.jar
        -D import.filename=resource.txt org.alidata.odps.mr.examples.WordCount args;
```

- -resources <resource_name_list>: specifies the resources that are used to run a MapReduce job. In most cases, you must specify the names of the resources that are used by the map or reduce function in the resource_name_list.

  > **Note**
  > - If the map or reduce function reads other MaxCompute resources, you must also add the names of these resources to resource_name_list.
  > - Multiple resources must be separated by commas (,). If you use cross-project resources, add `PROJECT/resources/` before resource_name_list. Example: `-resources otherproject/resources/resfile`.
  > - For more information about how to use the map or reduce function to read resources, see Resource samples.

- -classpath <local_file_list>: specifies the classpath that is used to run a MapReduce job in local mode. This parameter specifies the local paths. The paths include both the relative and absolute paths of the JAR package where the main function is located.

  Package names are separated by default file delimiters. In most cases, the Windows operating system uses semicolons (;) as the default file delimiter, and Linux uses commas (,). If you run a MapReduce job on a cloud server, separate package names with commas (,).

  > **Note** You may prefer to compile the main function in the same package as the map or reduce function. For more information, see WordCount samples. In this case, when you execute the sample program, mapreduce-examples.jar is included in both the -resources and -classpath options. However, the -resources option references the map or reduce function for distributed execution. The -classpath option references the main function for local execution. The specified JAR package must also be saved in a local directory.

- -D <prop_name>=<prop_value>: specifies the Java property of <mainClass> for local execution. You can specify multiple properties.
- -l: specifies that the MapReduce job is executed in local mode. This option is used for program debugging.

4.2.4.3. Input and output

This topic describes the input and output of MapReduce jobs in MaxCompute.

- The input and output of MapReduce jobs in MaxCompute support built-in data types of MaxCompute, including BIGINT, DOUBLE, STRING, DATETIME, and BOOLEAN. User-defined data types are not supported.
- MapReduce supports input data from multiple tables with different schemas. You can use the map function to obtain the table information corresponding to the current record.
- MapReduce supports null values as input data, but does not support views as input data.
- A reduce job can write outputs to different tables or different partitions of a table. The target tables can have different schemas. Different outputs are distinguished by labels. By default, no label is added to the default output. MapReduce does not support a function without output returned.

For more information about input and output examples, see MultipleInOut.

4.2.4.4. Resource usage

You can read MaxCompute resources at map and reduce stages. Any mapper or reducer can load resources to memory for you to use in code.

For more information, see Resource samples.

4.2.4.5. Job running in local mode

This topic describes the differences between the local mode and distributed mode in which MapReduce jobs run. It also provides examples of MapReduce jobs in local mode.

Introduction to the local mode

Before you run a job in local mode, you can specify the -local option in the JAR command to simulate the running of the job. This way, you can perform local debugging on the job.

During job running, the client downloads the metadata and data of the input table, metadata of the output table, and resources that are required for local debugging from MaxCompute. The downloaded data is saved to a local directory named `warehouse`.

After the job is completed, the computing results are saved to a file in the `warehouse` directory. If the input table and required resources are downloaded to the `warehouse` directory, MapReduce directly references the data and files in the directory next time, instead of downloading the data again.

Differences between the local mode and distributed mode

A MapReduce job that runs in local mode starts multiple map and reduce tasks to process data. These tasks run in sequence.

The simulated running process is different from an actual distributed running process in the following aspects:

- Rows in the input table: A maximum of 100 rows of data can be downloaded in local mode.
- Resource usage: In distributed mode, MaxCompute limits the size of resources that can be referenced. For more information, see MapReduce limits. However, no limits are imposed on the size of resources in local mode.
- Security: MaxCompute MapReduce and user-defined functions (UDFs) are limited by a Java sandbox in distributed mode. However, no limits are imposed in local mode.

Examples

The following code shows an example of a MapReduce job in local mode:

```
odps:my_project> jar -l com.aliyun.odps.mapred.example.WordCount wc_in wc_out
Summary:
counters: 10
    map-reduce framework
            combine_input_groups=2
            combine_output_records=2
            map_input_bytes=4
            map_input_records=1
            map_output_records=2
            map_output_[wc_out]_bytes=0
            map_output_[wc_out]_records=0
            reduce_input_groups=2
            reduce_output_[wc_out]_bytes=8
            reduce_output_[wc_out]_records=2
OK
```

For more information about the sample code of WordCount, see WordCount.

If this is the first time you run a local debugging command, a directory named *warehouse* is created in the current path after the command is executed. The following code shows the directory structure of `warehouse`.

```
<warehouse>
   |____my_project (project directory)
         |____ <__tables__>
         |       |__wc_in (table data directory)
         |       |       |____ data (file)
         |       |       |
         |       |       |____ <__schema__> (file)
         |       |__wc_out (table data directory)
         |                |____ data (file)
         |                |
         |                |____ <__schema__> (file)
         |
         |____ <__resources__>
                 |
                 |___table_resource_name (table resource)
                 |          |____<__ref__>
                 |
                 |___ file_resource_name (file resource)
```

- Directories at the same level as `my_project` indicate projects. Directories at the same level as `wc_in` and `wc_out` indicate data tables. The table data that you read or write by using the JAR command is downloaded to directories at this level.
- The *<__schema__>* file stores the metadata of a table. The following code defines the file format:

```
  project=local_project_name
  table=local_table_name
  columns=col1_name:col1_type,col2_name:col2_type
  partitions=p1:STRING,p2:BIGINT    -- In this example, you do not need to specify this f
ield.
```

Separate the name and data type of a column with a colon (:). Separate columns with commas (,). The project and table names, `project_name.table_name`, must be declared at the beginning of the *<__schema__>* file. Separate the declaration and column definition with a comma (,). Example: `project_name.table_name,col1_name:col1_type,col2_name:col2_type,……`

- The *data* file in the `tables` directory stores the table data. The number of columns and column data must match the definition in the *_schema_* file. Separate columns with commas (,).

  The *_schema_* file in the `wc_in` directory contains the following data:

  ```
  my_project.wc_in,key:STRING,value:STRING
  ```

  The *data* file contains the following data:

  ```
  0,2
  ```

  The client downloads the metadata and part of the data of a table from MaxCompute, and saves the data to the preceding files. The next time you run this example program, the client directly uses the data in the `wc_in` directory, instead of downloading it again.

  > **Note** Data can be downloaded from MaxCompute only for MapReduce jobs that run in local mode.

  The *_schema_* file in the `wc_out` directory contains the following data:

  ```
  my_project.wc_out,key:STRING,cnt:BIGINT
  ```

  The *data* file contains the following data:

  ```
  0,1
  2,1
  ```

  The client downloads the metadata of the `wc_out` table from MaxCompute, and saves the data to the *_schema_* file. After a job is completed, the results are saved to the *data* file.

  > **Note**
  > - You can also edit the *_schema_* and *data* files, and save the files in table directories.
  > - If you run a job in local mode and the client detects that the table directory exists, the client does not download the information of this table from MaxCompute. The local table directory can include a table that does not exist in MaxCompute.

4.2.5. Program Example

4.2.5.1. WordCount example

This topic describes an example of running WordCount in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
   i. Create test tables.

   ```
   create table wc_in (key string, value string);
   create table wc_out(key string, cnt bigint);
   ```

   ii. Add test resources.

   ```
   add jar data\resources\mapreduce-examples.jar -f;
   ```

2. Prepare test tables and resources for WordCount.
3. Use Tunnel to import data.

   ```
   tunnel upload data wc_in;
   ```

   The following data is imported to the wc_in table:

   ```
   hello,odps
   ```

Procedure

Run WordCount on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.WordCount wc_in wc_out
```

Expected result

The job runs normally. The following data is returned in the wc_out table:

```
+------------+------------+
| key        | cnt        |
+------------+------------+
| hello      | 1          |
| odps       | 1          |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.conf.JobConf;
```

```
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
public class WordCount {
    public static class TokenizerMapper extends MapperBase {
        private Record word;
        private Record one;
        @Override
            public void setup(TaskContext context) throws IOException {
            word = context.createMapOutputKeyRecord();
            one = context.createMapOutputValueRecord();
            one.set(new Object[] { 1L });
            System.out.println("TaskID:" + context.getTaskID().toString());
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
            for (int i = 0; i < record.getColumnCount(); i++) {
                word.set(new Object[] { record.get(i).toString() });
                context.write(word, one);
            }
        }
    }
    /**
       * A combiner class that combines map output by sum them.
       **/
    public static class SumCombiner extends ReducerBase {
        private Record count;
        @Override
            public void setup(TaskContext context) throws IOException {
            count = context.createMapOutputValueRecord();
        }
        /** The combiner implements the same interface as that of the reducer. The combiner
allows you to immediately run a local reduce job on the mapper to reduce the output of the
mapper. */
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long c = 0;
            while (values.hasNext()) {
                Record val = values.next();
                c += (Long) val.get(0);
            }
            count.set(0, c);
            context.write(key, count);
        }
    }
    /**
       * A reducer class that just emits the sum of the input values.
       **/
    public static class SumReducer extends ReducerBase {
        private Record result = null;
        @Override
            public void setup(TaskContext context) throws IOException {
```

```
            result = context.createOutputRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long count = 0;
            while (values.hasNext()) {
                Record val = values.next();
                count += (Long) val.get(0);
            }
            result.set(0, key.get(0));
            result.set(1, count);
            context.write(result);
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: WordCount <in_table> <out_table>");
            System.exit(2);
        }
        JobConf job = new JobConf();
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(SumCombiner.class);
        job.setReducerClass(SumReducer.class);
        /** Configure the schema that defines the intermediate output of the mapper as key-
value pairs. The intermediate output of the mapper exists as records. */
        job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
        /** Configure information about input and output tables. */
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
        JobClient.runJob(job);
    }
}
```

4.2.5.2. MapOnly example

For map-only jobs, a mapper directly generates key-value pairs to MaxCompute tables. You need only to specify output tables instead of the key-value metadata for the output of a mapper.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources that are used to run a map-only job.
   i. Create test tables.

```
create table wc_in (key string, value string);
create table wc_out(key string, cnt bigint);
```

ii. Add test resources.

```
add jar data\resources\mapreduce-examples.jar -f;
```

3. Use Tunnel to import data.

```
tunnel upload data wc_in;
```

The following data is imported to the wc_in table:

```
hello,odps
hello,odps
```

Procedure

Run a map-only job on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.MapOnly wc_in wc_out map
```

Expected result

The job runs normally. The following data is returned in the wc_out table:

```
+------------+------------+
| key        | cnt        |
+------------+------------+
| hello      | 1          |
| hello      | 1          |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;
public class MapOnly {
    public static class MapperClass extends MapperBase {
        @Override
            public void setup(TaskContext context) throws IOException {
            boolean is = context.getJobConf().getBoolean("option.mapper.setup", false);
            /** The main function executes the following logic only if option.mapper.setup
is set to true in the JobConf file: */
            if (is) {
                Record result = context.createOutputRecord();
                result.set(0, "setup");
```

```
                result.set(0, "setup");
                result.set(1, 1L);
                context.write(result);
            }
        }
        @Override
            public void map(long key, Record record, TaskContext context) throws IOExceptio
n {
            boolean is = context.getJobConf().getBoolean("option.mapper.map", false);
            /** The main function executes the following logic only if option.mapper.map is
set to true in the JobConf file: */
            if (is) {
                Record result = context.createOutputRecord();
                result.set(0, record.get(0));
                result.set(1, 1L);
                context.write(result);
            }
        }
        @Override
            public void cleanup(TaskContext context) throws IOException {
            boolean is = context.getJobConf().getBoolean("option.mapper.cleanup", false);
            /** The main function executes the following logic only if option.mapper.cleanu
p is set to true in the JobConf file: */
            if (is) {
                Record result = context.createOutputRecord();
                result.set(0, "cleanup");
                result.set(1, 1L);
                context.write(result);
            }
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 2 && args.length != 3) {
            System.err.println("Usage: OnlyMapper <in_table> <out_table> [setup|map|cleanup
]");
            System.exit(2);
        }
        JobConf job = new JobConf();
        job.setMapperClass(MapperClass.class);
        /** For MapOnly jobs, the number of reducers must be explicitly set to 0. */
        job.setNumReduceTasks(0);
        /** Configure information about input and output tables. */
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
        if (args.length == 3) {
            String options = new String(args[2]);
            /** You can specify key-value pairs in the JobConf file, and use getJobConf of
the context to query the configurations in a mapper. */
            if (options.contains("setup")) {
                job.setBoolean("option.mapper.setup", true);
            }
            if (options.contains("map")) {
                job.setBoolean("option.mapper.map", true);
            }
            if (options.contains("cleanup")) {
```

```
                job.setBoolean("option.mapper.cleanup", true);
            }
        }
        JobClient.runJob(job);
    }
}
```

4.2.5.3. MultipleInOut example

This topic describes an example of running MultipleInOut in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for MultipleInOut.
   i. Create test tables.

   ```
   create table wc_in1(key string, value string);
   create table wc_in2(key string, value string);
   create table mr_multiinout_out1 (key string, cnt bigint);
   create table mr_multiinout_out2 (key string, cnt bigint) partitioned by (a string,
   b string);
   alter table mr_multiinout_out2 add partition (a='1', b='1');
   alter table mr_multiinout_out2 add partition (a='2', b='2');
   ```

   ii. Add test resources.

   ```
   add jar data\resources\mapreduce-examples.jar -f;
   ```

3. Use Tunnel to import data.

   ```
   tunnel upload data1 wc_in1;
   tunnel upload data2 wc_in2;
   ```

   The following data is imported to the wc_in1 table:

   ```
   hello,odps
   ```

   The following data is imported to the wc_in2 table:

   ```
   hello,world
   ```

Procedure

Run MultipleInOut on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.MultipleInOut wc_in1,wc_in2 mr_multiinout_out1,mr_multi
inout_out2|a=1/b=1|out1,mr_multiinout_out2|a=2/b=2|out2;
```

Expected result

The job runs normally. The following data is returned in the mr_multiinout_out1 table:

```
+------------+------------+
| key        | cnt        |
+------------+------------+
| default    | 1          |
+------------+------------+
```

The following data is returned in the mr_multiinout_out2 table:

```
+--------+------------+---+---+
| key    | cnt        | a | b |
+--------+------------+---+---+
| odps   | 1          | 1 | 1 |
| world  | 1          | 1 | 1 |
| out1   | 1          | 1 | 1 |
| hello  | 2          | 2 | 2 |
| out2   | 1          | 2 | 2 |
+--------+------------+---+---+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import java.util.LinkedHashMap;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * Multi input & output example.
     **/
public class MultipleInOut {
    public static class TokenizerMapper extends MapperBase {
        Record word;
        Record one;
        @Override
            public void setup(TaskContext context) throws IOException {
            word = context.createMapOutputKeyRecord();
            one = context.createMapOutputValueRecord();
            one.set(new Object[] { 1L });
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
```

```
            throws IOException {
            for (int i = 0; i < record.getColumnCount(); i++) {
                word.set(new Object[] { record.get(i).toString() });
                context.write(word, one);
            }
        }
    }
    public static class SumReducer extends ReducerBase {
        private Record result;
        private Record result1;
        private Record result2;
        @Override
            public void setup(TaskContext context) throws IOException {
            /** Create a record for each output and add labels to distinguish outputs. */
            result = context.createOutputRecord();
            result1 = context.createOutputRecord("out1");
            result2 = context.createOutputRecord("out2");
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long count = 0;
            while (values.hasNext()) {
                Record val = values.next();
                count += (Long) val.get(0);
            }
            long mod = count % 3;
            if (mod == 0) {
                result.set(0, key.get(0));
                result.set(1, count);
                /** If you do not specify a label, the default output is used. */
                context.write(result);
            } else if (mod == 1) {
                result1.set(0, key.get(0));
                result1.set(1, count);
                context.write(result1, "out1");
            } else {
                result2.set(0, key.get(0));
                result2.set(1, count);
                context.write(result2, "out2");
            }
        }
        @Override
            public void cleanup(TaskContext context) throws IOException {
            Record result = context.createOutputRecord();
            result.set(0, "default");
            result.set(1, 1L);
            context.write(result);
            Record result1 = context.createOutputRecord("out1");
            result1.set(0, "out1");
            result1.set(1, 1L);
            context.write(result1, "out1");
            Record result2 = context.createOutputRecord("out2");
            result2.set(0, "out2");
            result2.set(1, 1L);
```

```
            context.write(result2, "out2");
        }
    }
    /** Convert partition strings such as "ds=1/pt=2" to MAP. */
    public static LinkedHashMap<String, String> convertPartSpecToMap(
        String partSpec) {
        LinkedHashMap<String, String> map = new LinkedHashMap<String, String>();
        if (partSpec != null && !partSpec.trim().isEmpty()) {
            String[] parts = partSpec.split("/");
            for (String part : parts) {
                String[] ss = part.split("=");
                if (ss.length != 2) {
                    throw new RuntimeException("ODPS-0730001: error part spec format: "
                                               + partSpec);
                }
                map.put(ss[0], ss[1]);
            }
        }
        return map;
    }
    public static void main(String[] args) throws Exception {
        String[] inputs = null;
        String[] outputs = null;
        if (args.length == 2) {
            inputs = args[0].split(",");
            outputs = args[1].split(",");
        } else {
            System.err.println("MultipleInOut in... out...");
            System.exit(1);
        }
        JobConf job = new JobConf();
        job.setMapperClass(TokenizerMapper.class);
        job.setReducerClass(SumReducer.class);
        job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
        /** Parse input table strings. */
        for (String in : inputs) {
            String[] ss = in.split("\\|");
            if (ss.length == 1) {
                InputUtils.addTable(TableInfo.builder().tableName(ss[0]).build(), job);
            } else if (ss.length == 2) {
                LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
                InputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).buil
d(), job);
            } else {
                System.err.println("Style of input: " + in + " is not right");
                System.exit(1);
            }
        }
        /** Parse output table strings. */
        for (String out : outputs) {
            String[] ss = out.split("\\|");
            if (ss.length == 1) {
                OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).build(), job);
```

```
            } else if (ss.length == 2) {
                LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
                OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).bui
ld(), job);
            } else if (ss.length == 3) {
                if (ss[1].isEmpty()) {
                    LinkedHashMap<String, String> map = convertPartSpecToMap(ss[2]);
                    OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map)
.build(), job);
                } else {
                    LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
                    OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map)
                                        .label(ss[2]).build(), job);
                }
            } else {
                System.err.println("Style of output: " + out + " is not right");
                System.exit(1);
            }
        }
        JobClient.runJob(job);
    }
}
```

4.2.5.4. MultiJobs example

This topic describes an example of running MultiJobs in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for MultiJobs.
   i. Create test tables.

   ```
   create table mr_empty (key string, value string);
   create table mr_multijobs_out (value bigint);
   ```

   ii. Add test resources.

   ```
   add table mr_multijobs_out as multijobs_res_table -f;
   add jar data\resources\mapreduce-examples.jar -f;
   ```

Procedure

Run MultiJobs on the MaxCompute client.

```
jar -resources mapreduce-examples.jar,multijobs_res_table -classpath data\resources\mapredu
ce-examples.jar
 com.aliyun.odps.mapred.open.example.MultiJobs mr_multijobs_out;
```

Expected result

The job runs normally. The following data is returned in the mr_multijobs_out table:

```
+------------+
| value      |
+------------+
| 0          |
+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * MultiJobs
     *
     * Running multiple job
     *
     **/
public class MultiJobs {
    public static class InitMapper extends MapperBase {
        @Override
            public void setup(TaskContext context) throws IOException {
            Record record = context.createOutputRecord();
            long v = context.getJobConf().getLong("multijobs.value", 2);
            record.set(0, v);
            context.write(record);
        }
    }
    public static class DecreaseMapper extends MapperBase {
        @Override
            public void cleanup(TaskContext context) throws IOException {
            /** Obtain the variable values that are defined in the main function from JobCo
nf. */
            long expect = context.getJobConf().getLong("multijobs.expect.value", -1);
            long v = -1;
            int count = 0;
            /** Read the data from the output table of the previous job. */
            Iterator<Record> iter = context.readResourceTable("multijobs_res_table");
            while (iter.hasNext()) {
                Record r = iter.next();
                v = (Long) r.get(0);
```

```
                if (expect != v) {
                    throw new IOException("expect: " + expect + ", but: " + v);
                }
                count++;
            }
            if (count != 1) {
                throw new IOException("res_table should have 1 record, but: " + count);
            }
            Record record = context.createOutputRecord();
            v--;
            record.set(0, v);
            context.write(record);
            /** Set the counter. The counter value can be obtained in the main function aft
er the job is completed. */
            context.getCounter("multijobs", "value").setValue(v);
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("Usage: TestMultiJobs <table>");
            System.exit(1);
        }
        String tbl = args[0];
        long iterCount = 2;
        System.err.println("Start to run init job.");
        JobConf initJob = new JobConf();
        initJob.setLong("multijobs.value", iterCount);
        initJob.setMapperClass(InitMapper.class);
        InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), initJob);
        OutputUtils.addTable(TableInfo.builder().tableName(tbl).build(), initJob);
        initJob.setMapOutputKeySchema(SchemaUtils.fromString("key:string"));
        initJob.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));
        /** Explicitly set the number of reducers to 0 for map-only jobs. */
        initJob.setNumReduceTasks(0);
        JobClient.runJob(initJob);
        while (true) {
            System.err.println("Start to run iter job, count: " + iterCount);
            JobConf decJob = new JobConf();
            decJob.setLong("multijobs.expect.value", iterCount);
            decJob.setMapperClass(DecreaseMapper.class);
            InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), decJob);
            OutputUtils.addTable(TableInfo.builder().tableName(tbl).build(), decJob);
            /** Explicitly set the number of reducers to 0 for map-only jobs. */
            decJob.setNumReduceTasks(0);
            RunningJob rJob = JobClient.runJob(decJob);
            iterCount--;
            /** If the specified number of iterations is reached, exit the loop. */
            if (rJob.getCounters().findCounter("multijobs", "value").getValue() == 0) {
                break;
            }
        }
        if (iterCount != 0) {
            throw new IOException("Job failed.");
        }
```

```
    }
}
```

4.2.5.5. SecondarySort example

This topic describes an example of running SecondarySort in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for SecondarySort.
   i. Create test tables.

   ```
   create table ss_in(key bigint, value bigint);
   create table ss_out(key bigint, value bigint)
   ```

   ii. Add test resources.

   ```
   add jar data\resources\mapreduce-examples.jar -f;
   ```

3. Use Tunnel to import data.

   ```
   tunnel upload data ss_in;
   ```

   The following data is imported to the ss_in table:

   ```
   1,2
   2,1
   1,1
   2,2
   ```

Procedure

Run SecondarySort on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.SecondarySort ss_in ss_out;
```

Expected result

The job runs normally. The following data is returned in the ss_out table:

```
+------------+------------+
| key        | value      |
+------------+------------+
| 1          | 1          |
| 1          | 2          |
| 2          | 1          |
| 2          | 2          |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;
/**
     *
     * This is an example ODPS Map/Reduce application. It reads the input table that
     * must contain two integers per record. The output is sorted by the first and
     * second number and grouped on the first number.
     *
     **/
public class SecondarySort {
    /**
       * Read two integers from each line and generate a key, value pair as ((left,
       * right), right).
       **/
    public static class MapClass extends MapperBase {
        private Record key;
        private Record value;
        @Override
            public void setup(TaskContext context) throws IOException {
            key = context.createMapOutputKeyRecord();
            value = context.createMapOutputValueRecord();
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
            long left = 0;
            long right = 0;
            if (record.getColumnCount() > 0) {
                left = (Long) record.get(0);
                if (record.getColumnCount() > 1) {
                    right = (Long) record.get(1);
                }
                key.set(new Object[] { (Long) left, (Long) right });
                value.set(new Object[] { (Long) right });
                context.write(key, value);
            }
        }
    }
    /**
       * A reducer class that just emits the sum of the input values
```

```
     * A reducer class that just emits the sum of the input values.
     **/
    public static class ReduceClass extends ReducerBase {
        private Record result = null;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            result.set(0, key.get(0));
            while (values.hasNext()) {
                Record value = values.next();
                result.set(1, value.get(0));
                context.write(result);
            }
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: secondarysrot <in> <out>");
            System.exit(2);
        }
        JobConf job = new JobConf();
        job.setMapperClass(MapClass.class);
        job.setReducerClass(ReduceClass.class);
        /** Set multiple columns as keys. */
        //compare first and second parts of the pair
        job.setOutputKeySortColumns(new String[] { "i1", "i2" });
        //partition based on the first part of the pair
        job.setPartitionColumns(new String[] { "i1" });
        //grouping comparator based on the first part of the pair
        job.setOutputGroupingColumns(new String[] { "i1" });
        //the map output is LongPair, Long
        job.setMapOutputKeySchema(SchemaUtils.fromString("i1:bigint,i2:bigint"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("i2x:bigint"));
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
        JobClient.runJob(job);
        System.exit(0);
    }
}
```

4.2.5.6. Resource usage example

This topic describes an example of using resources in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. On the MaxCompute client, perform the following operations to prepare test tables and resources:

i. Create test tables.

```
create table mr_upload_src(key bigint, value string);
```

ii. Add test resources.

```
add jar data\resources\mapreduce-examples.jar -f;
add file data\resources\import.txt -f;
```

iii. The *import.txt* file contains the following content:

```
1000,odps
```

Procedure

Run the following code on the MaxCompute client to upload the data in the test resources to the mr_upload_src table:

```
jar -resources mapreduce-examples.jar,import.txt -classpath data\resources\mapreduce-exampl
es.jar
com.aliyun.odps.mapred.open.example.Upload import.txt mr_upload_src;
```

Expected result

The job runs normally. The following data is returned in the mr_upload_src table:

```
+------------+------------+
| key        | value      |
+------------+------------+
| 1000       | odps       |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.BufferedInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * Upload
     *
     * Import data from text file into table
     *
     **/
```

```
public class Upload {
    public static class UploadMapper extends MapperBase {
        @Override
            public void setup(TaskContext context) throws IOException {
            Record record = context.createOutputRecord();
            StringBuilder importdata = new StringBuilder();
            BufferedInputStream bufferedInput = null;
            try {
                byte[] buffer = new byte[1024];
                int bytesRead = 0;
                String filename = context.getJobConf().get("import.filename");
                bufferedInput = context.readResourceFileAsStream(filename);
                while ((bytesRead = bufferedInput.read(buffer)) != -1) {
                    String chunk = new String(buffer, 0, bytesRead);
                    importdata.append(chunk);
                }
                String lines[] = importdata.toString().split("\n");
                for (int i = 0; i < lines.length; i++) {
                    String[] ss = lines[i].split(",");
                    record.set(0, Long.parseLong(ss[0].trim()));
                    record.set(1, ss[1].trim());
                    context.write(record);
                }
            } catch (FileNotFoundException ex) {
                throw new IOException(ex);
            } catch (IOException ex) {
                throw new IOException(ex);
            } finally {
            }
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: Upload <import_txt> <out_table>");
            System.exit(2);
        }
        JobConf job = new JobConf();
        job.setMapperClass(UploadMapper.class);
        /** Specify the resource name, which can be obtained in the map stage by using the
JobConf interface. */
        job.set("import.filename", args[0]);
        /** Explicitly set the number of reducers to 0 for map-only jobs. */
        job.setNumReduceTasks(0);
        job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));
        InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
        JobClient.runJob(job);
    }
}
```

You can use one of the following methods to set the JobConf configuration file:

- Use the JobConf interface in SDK. This method is used in the preceding example.
- Use the -conf parameter in a jar command to specify a new JobConf configuration file.

4.2.5.7. UserDefinedCounters example

This topic describes an example of running UserDefinedCounters in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for UserDefinedCounters.
   i. Create test tables.

   ```
   create table wc_in (key string, value string);
   create table wc_out(key string, cnt bigint);
   ```

   ii. Add test resources.

   ```
   add jar data\resources\mapreduce-examples.jar -f;
   ```

3. Use Tunnel to import data.

   ```
   tunnel upload data wc_in;
   ```

   The following data is imported to the wc_in table:

   ```
   hello,odps
   ```

Procedure

Run UserDefinedCounters on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.UserDefinedCounters wc_in wc_out
```

Expected result

The job runs normally. The following user-defined counters are returned:

```
Counters: 3
com.aliyun.odps.mapred.open.example.UserDefinedCounters$MyCounter
MAP_TASKS=1
REDUCE_TASKS=1
TOTAL_TASKS=2
```

The following data is returned in the wc_out table:

```
+------------+------------+
| key        | cnt        |
+------------+------------+
| hello      | 1          |
| odps       | 1          |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.counter.Counters;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;
/**
     *
     * User Defined Counters
     *
     **/
public class UserDefinedCounters {
    enum MyCounter {
        TOTAL_TASKS, MAP_TASKS, REDUCE_TASKS
    }
    public static class TokenizerMapper extends MapperBase {
        private Record word;
        private Record one;
        @Override
            public void setup(TaskContext context) throws IOException {
            super.setup(context);
            Counter map_tasks = context.getCounter(MyCounter.MAP_TASKS);
            Counter total_tasks = context.getCounter(MyCounter.TOTAL_TASKS);
            map_tasks.increment(1);
            total_tasks.increment(1);
            word = context.createMapOutputKeyRecord();
            one = context.createMapOutputValueRecord();
            one.set(new Object[] { 1L });
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
            for (int i = 0; i < record.getColumnCount(); i++) {
                word.set(new Object[] { record.get(i).toString() });
```

```
                word.set(new Object[] { record.get(1).toString() });
                context.write(word, one);
            }
        }
    }
    public static class SumReducer extends ReducerBase {
        private Record result = null;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
            Counter reduce_tasks = context.getCounter(MyCounter.REDUCE_TASKS);
            Counter total_tasks = context.getCounter(MyCounter.TOTAL_TASKS);
            reduce_tasks.increment(1);
            total_tasks.increment(1);
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long count = 0;
            while (values.hasNext()) {
                Record val = values.next();
                count += (Long) val.get(0);
            }
            result.set(0, key.get(0));
            result.set(1, count);
            context.write(result);
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err
                .println("Usage: TestUserDefinedCounters <in_table> <out_table>");
            System.exit(2);
        }
        JobConf job = new JobConf();
        job.setMapperClass(TokenizerMapper.class);
        job.setReducerClass(SumReducer.class);
        job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
        RunningJob rJob = JobClient.runJob(job);
        /** The job runs normally. The values of the custom counters in the job are returne
d. */
        Counters counters = rJob.getCounters();
        long m = counters.findCounter(MyCounter.MAP_TASKS).getValue();
        long r = counters.findCounter(MyCounter.REDUCE_TASKS).getValue();
        long total = counters.findCounter(MyCounter.TOTAL_TASKS).getValue();
        System.exit(0);
    }
}
```

4.2.5.8. Grep example

This topic describes an example of running Grep in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for Grep.
   i. Create test tables.

   ```
   create table mr_src(key string, value string);
   create table mr_grep_tmp (key string, cnt bigint);
   create table mr_grep_out (key bigint, value string);
   ```

   ii. Add test resources.

   ```
   add jar data\resources\mapreduce-examples.jar -f;
   ```

3. Use Tunnel to import data.

   ```
   tunnel upload data mr_src;
   ```

   The following data is imported to the mr_src table:

   ```
   hello,odps
   hello,world
   ```

Procedure

Run Grep on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.Grep mr_src mr_grep_tmp mr_grep_out hello;
```

Expected result

The job runs normally. The following data is returned in the mr_grep_out table:

```
+------------+------------+
| key        | value      |
+------------+------------+
| 2          | hello      |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
import java.util.regex.Pattern;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.Mapper;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     *
     * Extracts matching regexs from input files and counts them.
     *
     **/
public class Grep {
    /**
       * RegexMapper
       **/
    public class RegexMapper extends MapperBase {
        private Pattern pattern;
        private int group;
        private Record word;
        private Record one;
        @Override
            public void setup(TaskContext context) throws IOException {
            JobConf job = (JobConf) context.getJobConf();
            pattern = Pattern.compile(job.get("mapred.mapper.regex"));
            group = job.getInt("mapred.mapper.regex.group", 0);
            word = context.createMapOutputKeyRecord();
            one = context.createMapOutputValueRecord();
            one.set(new Object[] { 1L });
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context) throws IOEx
ception {
            for (int i = 0; i < record.getColumnCount(); ++i) {
                String text = record.get(i).toString();
                Matcher matcher = pattern.matcher(text);
                while (matcher.find()) {
                    word.set(new Object[] { matcher.group(group) });
                    context.write(word, one);
                }
            }
        }
    }
    /**
       * LongSumReducer
       **/
    public class LongSumReducer extends ReducerBase {
        private Record result = null;
        @Override
```

```
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context) th
rows IOException {
            long count = 0;
            while (values.hasNext()) {
                Record val = values.next();
                count += (Long) val.get(0);
            }
            result.set(0, key.get(0));
            result.set(1, count);
            context.write(result);
        }
    }
    /**
       * A {@link Mapper} that swaps keys and values.
       **/
    public class InverseMapper extends MapperBase {
        private Record word;
        private Record count;
        @Override
            public void setup(TaskContext context) throws IOException {
            word = context.createMapOutputValueRecord();
            count = context.createMapOutputKeyRecord();
        }
        /**
         * The inverse function. Input keys and values are swapped.
         **/
        @Override
            public void map(long recordNum, Record record, TaskContext context) throws IOEx
ception {
            word.set(new Object[] { record.get(0).toString() });
            count.set(new Object[] { (Long) record.get(1) });
            context.write(count, word);
        }
    }
    /**
       * IdentityReducer
       **/
    public class IdentityReducer extends ReducerBase {
        private Record result = null;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        /** Writes all keys and values directly to output. **/
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context) th
rows IOException {
            result.set(0, key.get(0));
            while (values.hasNext()) {
                Record val = values.next();
```

```
                result.set(1, val.get(0));
                context.write(result);
            }
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length < 4) {
            System.err.println("Grep <inDir> <tmpDir> <outDir> <regex> [<group>]");
            System.exit(2);
        }
        JobConf grepJob = new JobConf();
        grepJob.setMapperClass(RegexMapper.class);
        grepJob.setReducerClass(LongSumReducer.class);
        grepJob.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
        grepJob.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), grepJob);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), grepJob);
        /** Set the regular expression for grepJob. */
        grepJob.set("mapred.mapper.regex", args[3]);
        if (args.length == 5) {
            grepJob.set("mapred.mapper.regex.group", args[4]);
        }
        @SuppressWarnings("unused")
            RunningJob rjGrep = JobClient.runJob(grepJob);
        /** Specify the output of grepJob as the input of sortJob. */
        JobConf sortJob = new JobConf();
        sortJob.setMapperClass(InverseMapper.class);
        sortJob.setReducerClass(IdentityReducer.class);
        sortJob.setMapOutputKeySchema(SchemaUtils.fromString("count:bigint"));
        sortJob.setMapOutputValueSchema(SchemaUtils.fromString("word:string"));
        InputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), sortJob);
        OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), sortJob);
        sortJob.setNumReduceTasks(1); // write a single file
        sortJob.setOutputKeySortColumns(new String[] { "count" });
        @SuppressWarnings("unused")
            RunningJob rjSort = JobClient.runJob(sortJob);
    }
}
```

4.2.5.9. Join example

The MaxCompute MapReduce framework does not support Join operations. However, you can join data by using the custom map or reduce function.

Preparations

1. The JAR package of the test program is prepared. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for Join.

i. Create test tables. The mr_Join_src1 and mr_Join_src2 tables are joined in the test procedure, and the mr_join_out table is used as the output table of the Join operation.

```
create table mr_Join_src1(key bigint, value string);
create table mr_Join_src2(key bigint, value string);
create table mr_Join_out(key bigint, value1 string, value2 string);
```

ii. Add test resources.

```
add jar data\resources\mapreduce-examples.jar -f;
```

3. Use Tunnel to import data.

```
tunnel upload data1 mr_Join_src1;
tunnel upload data2 mr_Join_src2;
```

The following data is imported to the mr_Join_src1 table:

```
1,hello
2,odps
```

The following data is imported to the mr_Join_src2 table:

```
1,odps
3,hello
4,odps
```

Procedure

Perform the Join operation on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.Join mr_Join_src1 mr_Join_src2 mr_Join_out;
```

Expected result

The job runs normally. The following data is returned in the mr_Join_out table. value1 indicates the value in the mr_Join_src1 table and value2 indicates the value in the mr_Join_src2 table.

```
+------------+------------+------------+
| key        | value1     | value2     |
+------------+------------+------------+
|  1         | hello      |  odps      |
+------------+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.apache.commons.logging.Log;
```

```
import org.apache.commons.logging.LogFactory;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * Join, mr_Join_src1/mr_Join_src2(key bigint, value string), mr_Join_out(key
     * bigint, value1 string, value2 string)
     *
     */
public class Join {
    public static final Log LOG = LogFactory.getLog(Join.class);
    public static class JoinMapper extends MapperBase {
        private Record mapkey;
        private Record mapvalue;
        private long tag;
        @Override
            public void setup(TaskContext context) throws IOException {
            mapkey = context.createMapOutputKeyRecord();
            mapvalue = context.createMapOutputValueRecord();
            tag = context.getInputTableInfo().getLabel().equals("left") ? 0 : 1;
        }
        @Override
            public void map(long key, Record record, TaskContext context)
            throws IOException {
            mapkey.set(0, record.get(0));
            mapkey.set(1, tag);
            for (int i = 1; i < record.getColumnCount(); i++) {
                mapvalue.set(i - 1, record.get(i));
            }
            context.write(mapkey, mapvalue);
        }
    }
    public static class JoinReducer extends ReducerBase {
        private Record result = null;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        /** Each input of the reduce function is the records that have the same key. */
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long k = key.getBigint(0);
            List<Object[]> leftValues = new ArrayList<Object[]>();
            /** Records are sorted based on the combination of the key and tag. This ensure
s that records in the left table are passed to the reduce function first when the reduce fu
nction performs the Join operation. */
            while (values.hasNext()) {
```

```
                Record value = values.next();
                long tag = (Long) key.get(1);
                /** Data in the left table is first cached in memory. */
                if (tag == 0) {
                    leftValues.add(value.toArray().clone());
                } else {
                    /** Data in the right table is joined with all data in the left table.
*/
                    /** The sample code has poor performance and is only used as an example
. We recommend that you do not use the code in your production environment. */
                    for (Object[] leftValue : leftValues) {
                        int index = 0;
                        result.set(index++, k);
                        for (int i = 0; i < leftValue.length; i++) {
                            result.set(index++, leftValue[i]);
                        }
                        for (int i = 0; i < value.getColumnCount(); i++) {
                            result.set(index++, value.get(i));
                        }
                        context.write(result);
                    }
                }
            }
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length != 3) {
            System.err.println("Usage: Join <input table1> <input table2> <out>");
            System.exit(2);
        }
        JobConf job = new JobConf();
        job.setMapperClass(JoinMapper.class);
        job.setReducerClass(JoinReducer.class);
        job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,tag:bigint"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));
        job.setPartitionColumns(new String[]{"key"});
        job.setOutputKeySortColumns(new String[]{"key", "tag"});
        job.setOutputGroupingColumns(new String[]{"key"});
        job.setNumReduceTasks(1);
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).label("left").build(), j
ob);
        InputUtils.addTable(TableInfo.builder().tableName(args[1]).label("right").build(),
job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), job);
        JobClient.runJob(job);
    }
}
```

4.2.5.10. Sleep example

This topic describes an example of using Sleep in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test resources for SleepJob.

```
add jar data\resources\mapreduce-examples.jar -f;
```

Procedure

Run Sleep on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.Sleep 10;
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.Sleep 100;
```

Expected result

The job runs normally. The runtime of different sleep durations can be compared to determine the effect.

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.conf.JobConf;
public class Sleep {
  private static final String SLEEP_SECS = "sleep.secs";
  public static class MapperClass extends MapperBase {
    /** No data is entered, the map function is not executed, and the related logic can be
written only into setup. */
    @Override
    public void setup(TaskContext context) throws IOException {
      try {
        /** Obtain the number of sleep seconds set in JobConf. */
        Thread.sleep(context.getJobConf().getInt(SLEEP_SECS, 1) * 1000);
      } catch (InterruptedException e) {
        throw new RuntimeException(e);
      }
    }
  }
  public static void main(String[] args) throws Exception {
    if (args.length != 1) {
      System.err.println("Usage: Sleep <sleep_secs>");
      System.exit(-1);
    }
    JobConf job = new JobConf();
    job.setMapperClass(MapperClass.class);
    /** This instance is also a MapOnly job and the number of reducers must be set to 0. */
    job.setNumReduceTasks(0);
    /** The number of mappers must be specified by the user because no input table is provi
ded. */
    job.setNumMapTasks(1);
    job.set(SLEEP_SECS, args[0]);
    JobClient.runJob(job);
  }
}
```

4.2.5.11. Unique example

This topic describes an example of using Unique in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources for Unique.
   i. Create test tables.

```
create table ss_in(key bigint, value bigint);
create table ss_out(key bigint, value bigint);
```

ii. Add test resources.

```
add jar data\resources\mapreduce-examples.jar -f;
```

3. Use Tunnel to import data.

```
tunnel upload data ss_in;
```

The following data is imported to the ss_in table:

```
1,1
1,1
2,2
2,2
```

Procedure

Run Unique on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.Unique ss_in ss_out key;
```

Expected result

The job runs normally. The following data is returned in the ss_out table:

```
+------------+------------+
| key        | value      |
+------------+------------+
| 1          |   1        |
| 2          |   2        |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * Unique Remove duplicate words
     *
     **/
public class Unique {
```

```
public class Unique {
    public static class OutputSchemaMapper extends MapperBase {
        private Record key;
        private Record value;
        @Override
            public void setup(TaskContext context) throws IOException {
            key = context.createMapOutputKeyRecord();
            value = context.createMapOutputValueRecord();
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
            long left = 0;
            long right = 0;
            if (record.getColumnCount() > 0) {
                left = (Long) record.get(0);
                if (record.getColumnCount() > 1) {
                    right = (Long) record.get(1);
                }
                key.set(new Object[] { (Long) left, (Long) right });
                value.set(new Object[] { (Long) left, (Long) right });
                context.write(key, value);
            }
        }
    }
    public static class OutputSchemaReducer extends ReducerBase {
        private Record result = null;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            result.set(0, key.get(0));
            while (values.hasNext()) {
                Record value = values.next();
                result.set(1, value.get(1));
            }
            context.write(result);
        }
    }
    public static void main(String[] args) throws Exception {
        if (args.length > 3 || args.length < 2) {
            System.err.println("Usage: unique <in> <out> [key|value|all]");
            System.exit(2);
        }
        String ops = "all";
        if (args.length == 3) {
            ops = args[2];
        }
        /** The input group of Reduce is determined by the value of the setOutputGroupingCo
lumns parameter. If this parameter is not specified, the default value MapOutputKeySchema i
s used. */
        // Key Unique
```

```
        if (ops.equals("key")) {
            JobConf job = new JobConf();
            job.setMapperClass(OutputSchemaMapper.class);
            job.setReducerClass(OutputSchemaReducer.class);
            job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
            job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));
            job.setPartitionColumns(new String[] { "key" });
            job.setOutputKeySortColumns(new String[] { "key", "value" });
            job.setOutputGroupingColumns(new String[] { "key" });
            job.set("tablename2", args[1]);
            job.setNumReduceTasks(1);
            job.setInt("table.counter", 0);
            InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
            OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
            JobClient.runJob(job);
        }
        // Key&Value Unique
        if (ops.equals("all")) {
            JobConf job = new JobConf();
            job.setMapperClass(OutputSchemaMapper.class);
            job.setReducerClass(OutputSchemaReducer.class);
            job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
            job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));
            job.setPartitionColumns(new String[] { "key" });
            job.setOutputKeySortColumns(new String[] { "key", "value" });
            job.setOutputGroupingColumns(new String[] { "key", "value" });
            job.set("tablename2", args[1]);
            job.setNumReduceTasks(1);
            job.setInt("table.counter", 0);
            InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
            OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
            JobClient.runJob(job);
        }
        // Value Unique
        if (ops.equals("value")) {
            JobConf job = new JobConf();
            job.setMapperClass(OutputSchemaMapper.class);
            job.setReducerClass(OutputSchemaReducer.class);
            job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
            job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));
            job.setPartitionColumns(new String[] { "value" });
            job.setOutputKeySortColumns(new String[] { "value" });
            job.setOutputGroupingColumns(new String[] { "value" });
            job.set("tablename2", args[1]);
            job.setNumReduceTasks(1);
            job.setInt("table.counter", 0);
            InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
            OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
            JobClient.runJob(job);
        }
    }
}
```

4.2.5.12. Sort example

This topic describes an example of using Sort in MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this topic, the JAR package is named *mapreduce-examples.jar* and stored in the local path *data\resources*.
2. Prepare test tables and resources.
   i. Create test tables.

   ```
   create table ss_in(key bigint, value bigint);
   create table ss_out(key bigint, value bigint);
   ```

   ii. Add test resources.

   ```
   add jar data\resources\mapreduce-examples.jar -f;
   ```

3. Use Tunnel to import data.

   ```
   tunnel upload data ss_in;
   ```

   The following data is imported to the ss_in table:

   ```
   2,1
   1,1
   3,1
   ```

Procedure

Sort data on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.Sort ss_in ss_out;
```

Expected result

The job runs normally. The following data is returned in the ss_out table:

```
+------------+------------+
| key        | value      |
+------------+------------+
| 1          | 1          |
| 2          | 1          |
| 3          | 1          |
+------------+------------+
```

Sample code

For information about Project Object Model (POM) dependencies, see Usage notes.

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Date;
```

```
import java.util.Date;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.example.lib.IdentityReducer;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * This is the trivial map/reduce program that does absolutely nothing other
     * than use the framework to fragment and sort the input values.
     *
     **/
public class Sort {
    static int printUsage() {
        System.out.println("sort <input> <output>");
        return -1;
    }
    /**
       * Implements the identity function, mapping record's first two columns to
       * outputs.
       **/
    public static class IdentityMapper extends MapperBase {
        private Record key;
        private Record value;
        @Override
            public void setup(TaskContext context) throws IOException {
            key = context.createMapOutputKeyRecord();
            value = context.createMapOutputValueRecord();
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
            key.set(new Object[] { (Long) record.get(0) });
            value.set(new Object[] { (Long) record.get(1) });
            context.write(key, value);
        }
    }
    /**
       * The main driver for sort program. Invoke this method to submit the
       * map/reduce job.
       *
       * @throws IOException
       *           When there is communication problems with the job tracker.
       **/
    public static void main(String[] args) throws Exception {
        JobConf jobConf = new JobConf();
        jobConf.setMapperClass(IdentityMapper.class);
        jobConf.setReducerClass(IdentityReducer.class);
        /** For global sorting, the number of reducers is set to 1. All the data is transfe
rred to the same reducer. */
        /** This method applies only to the scenarios when small amounts of data are proces
```

```
sed. If large amounts of data need to be processed, use other methods, such as TeraSort. */
        jobConf.setNumReduceTasks(1);
        jobConf.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint"));
        jobConf.setMapOutputValueSchema(SchemaUtils.fromString("value:bigint"));
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), jobConf);
        OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), jobConf);
        Date startTime = new Date();
        System.out.println("Job started: " + startTime);
        JobClient.runJob(jobConf);
        Date end_time = new Date();
        System.out.println("Job ended: " + end_time);
        System.out.println("The job took " + (end_time.getTime() - startTime.getTime()) / 1
000 + " seconds.");
    }
}
```

4.2.5.13. Examples of using partitioned tables as input

The following examples use partitioned tables as the input of a MapReduce job.

- Example 1:

```
public static void main(String[] args) throws Exception {
    JobConf job = new JobConf();
    ...
        LinkedHashMap<String, String> input = new LinkedHashMap<String, String>();
    input.put("pt", "123456");
    InputUtils.addTable(TableInfo.builder().tableName("input_table").partSpec(input).buil
d(), job);
    LinkedHashMap<String, String> output = new LinkedHashMap<String, String>();
    output.put("ds", "654321");
    OutputUtils.addTable(TableInfo.builder().tableName("output_table").partSpec(output).b
uild(), job);
    JobClient.runJob(job);
}
```

- Example 2:

```
package com.aliyun.odps.mapred.open.example;
...
    public static void main(String[] args) throws Exception {
    if (args.length != 2) {
        System.err.println("Usage: WordCount <in_table> <out_table>");
        System.exit(2);
    }
    JobConf job = new JobConf();
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(SumCombiner.class);
    job.setReducerClass(SumReducer.class);
    job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
    Account account = new AliyunAccount("my_access_id", "my_access_key");
    Odps odps = new Odps(account);
    odps.setEndpoint("odps_endpoint_url");
    odps.setDefaultProject("my_project");
    Table table = odps.tables().get(tblname);
    TableInfoBuilder builder = TableInfo.builder().tableName(tblname);
    for (Partition p : table.getPartitions()) {
        if (applicable(p)) {
            LinkedHashMap<String, String> partSpec = new LinkedHashMap<String, String>();
            for (String key : p.getPartitionSpec().keys()) {
                partSpec.put(key, p.getPartitionSpec().get(key));
            }
            InputUtils.addTable(builder.partSpec(partSpec).build(), job);
        }
    }
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
    JobClient.runJob(job);
}
```

Note

- In example 2, the MaxCompute SDK and MapReduce SDK are combined to implement a MapReduce task that reads data from specific partitions.
- The preceding code cannot be compiled for execution. It is only an example of the main function.
- The applicable function is the custom code logic that determines whether the partition can be used as the input of a MapReduce job.

4.2.5.14. Pipeline examples

This topic describes pipeline examples of MapReduce.

Preparations

1. Prepare the JAR package of the test program. In this example, the JAR package is named *mapreduce-examples.jar* and saved in the *data\resources* directory.
2. Prepare test tables and resources.

i. Create tables.

```
create table wc_in (key string, value string);
create table wc_out(key string, cnt bigint);
```

ii. Add resources.

```
add jar data\resources\mapreduce-examples.jar -f;
```

3. Use Tunnel to import data.

```
tunnel upload data wc_in;
```

The following data is imported to the wc_in table:

```
hello,odps
```

Procedure

Run a WordCount pipeline on the MaxCompute client.

```
jar -resources mapreduce-examples.jar -classpath data\resources\mapreduce-examples.jar
com.aliyun.odps.mapred.open.example.WordCountPipeline wc_in wc_out;
```

Expected results

If the job succeeds, the following result is returned:

```
+------------+------------+
| key        | cnt        |
+------------+------------+
| hello      | 1          |
| odps       | 1          |
+------------+------------+
```

Sample code

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.Column;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.Job;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.pipeline.Pipeline;
public class WordCountPipelineTest {
    public static class TokenizerMapper extends MapperBase {
        Record word;
        Record one;
        @Override
            public void setup(TaskContext context) throws IOException {
```

```
            public void setup(TaskContext context) throws IOException {
            word = context.createMapOutputKeyRecord();
            one = context.createMapOutputValueRecord();
            one.setBigint(0, 1L);
        }
        @Override
            public void map(long recordNum, Record record, TaskContext context)
            throws IOException {
            for (int i = 0; i < record.getColumnCount(); i++) {
                String[] words = record.get(i).toString().split("\\s+");
                for (String w : words) {
                    word.setString(0, w);
                    context.write(word, one);
                }
            }
        }
    }
    public static class SumReducer extends ReducerBase {
        private Record value;
        @Override
            public void setup(TaskContext context) throws IOException {
            value = context.createOutputValueRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long count = 0;
            while (values.hasNext()) {
                Record val = values.next();
                count += (Long) val.get(0);
            }
            value.set(0, count);
            context.write(key, value);
        }
    }
    public static class IdentityReducer extends ReducerBase {
        private Record result;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            while (values.hasNext()) {
                result.set(0, key.get(0));
                result.set(1, values.next().get(0));
                context.write(result);
            }
        }
    }
    public static void main(String[] args) throws OdpsException {
        if (args.length != 2) {
            System.err.println("Usage: WordCountPipeline <in_table> <out_table>");
            System.exit(2);
```

```
        }
        Job job = new Job();
        /***
         * During pipeline construction, if you do not specify OutputKeySortColumns, Partit
ionColumns, and OutputGroupingColumns for a mapper, the framework uses OutputKey of the map
per as the default values of these parameters.
         ***/
        Pipeline pipeline = Pipeline.builder()
            .addMapper(TokenizerMapper.class)
            .setOutputKeySchema(
            new Column[] { new Column("word", OdpsType.STRING) })
            .setOutputValueSchema(
            new Column[] { new Column("count", OdpsType.BIGINT) })
            .setOutputKeySortColumns(new String[] { "word" })
            .setPartitionColumns(new String[] { "word" })
            .setOutputGroupingColumns(new String[] { "word" })
            .addReducer(SumReducer.class)
            .setOutputKeySchema(
            new Column[] { new Column("word", OdpsType.STRING) })
            .setOutputValueSchema(
            new Column[] { new Column("count", OdpsType.BIGINT)})
            .addReducer(IdentityReducer.class).createPipeline();
        // Add the pipeline to jobconf. If you want to configure a combiner, use jobconf.
        job.setPipeline(pipeline);
        // Configure the input and output tables.
        job.addInput(TableInfo.builder().tableName(args[0]).build());
        job.addOutput(TableInfo.builder().tableName(args[1]).build());
        // Submit the job and wait for the job to complete.
        job.submit();
        job.waitForCompletion();
        System.exit(job.isSuccessful() == true ? 0 : 1);
    }
}
```

4.2.6. Java SDK

4.2.6.1. Overview

This topic describes common MapReduce classes and methods.

If you use Maven, you can search for odps-sdk-mapred in the Maven repository to find the latest version of the SDK for Java. You can declare the SDK in your project by using the following Maven dependency:

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-mapred</artifactId>
    <version>0.26.2-public</version>
</dependency>
```

Data types

The data types that MaxCompute MapReduce supports include BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, and DECIMAL. The following table describes the mapping between MaxCompute and Java data types.

MaxCompute data type	Java data type
BIGINT	LONG
STRING	STRING
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
DATETIME	DATE
DECIMAL	BIGDECIMAL

MapReduce classes

Class	Description
MapperBase	The base class that user-defined mapper classes must inherit. A mapper converts the records in the input table to key-value pairs and passes the key-value pairs to a reducer. Alternatively, a mapper can write the key-value pairs to the result table by skipping the reduce stage. The jobs that skip the reduce stage and directly return computing results are called map-only jobs.
ReducerBase	The base class that user-defined reducer classes must inherit. A reducer reduces a set of values associated with a key.
TaskContext	Describes the context of a task. The task context is an input parameter of multiple member methods of MapperBase and ReducerBase.
JobClient	Defines a job client. A job client submits and manages jobs. A job client can submit a job in blocking or non-blocking mode. The blocking mode is a synchronous mode, whereas the non-blocking mode is an asynchronous mode.
RunningJob	Defines a running job. The objects of this class are used to track the instances of running MapReduce jobs.
JobConf	Describes the configuration of a MapReduce job. The JobConf object is defined in the main function. Then, a job client submits a job to MaxCompute based on the JobConf object.

MapperBase

The following table describes the methods of the MapperBase class.

Method	Description
void cleanup(TaskContext context)	The method that is called after the map method at the end of the map stage.

Method	Description
void map(long key, Record record, TaskContext context)	Processes records in the input table.
void setup(TaskContext context)	The method that is called before the map method at the beginning of the map stage.

ReducerBase

The following table describes the methods of the ReducerBase class.

Method	Description
void cleanup(TaskContext context)	The method that is called after the reduce method at the end of the reduce stage.
void reduce(Record key, Iterator<Record > values, TaskContext context)	Processes records in the input table.
void setup(TaskContext context)	The method that is called before the reduce method at the beginning of the reduce stage.

TaskContext

The following table describes the methods of the TaskContext class.

Method	Description
TableInfo[] getOutputTableInfo()	Obtains information about the output table.
Record createOutputRecord()	Creates records for the default output table.
Record createOutputRecord(String label)	Creates records for the output table with the specified label.
Record createMapOutputKeyRecord()	Creates records for keys in the key-value pairs that are generated at the map stage.
Record createMapOutputValueRecord()	Creates records for values in the key-value pairs that are generated at the map stage.
void write(Record record)	Writes records to the default output table. The method can be called multiple times at the reduce stage.
void write(Record record, String label)	Writes records to the output table with the specified label. The method can be called multiple times at the reduce stage.
void write(Record key, Record value)	Converts records to key-value pairs. The method can be called multiple times at the map stage.

Method	Description
BufferedInputStream readResourceFileAsStream(String resourceName)	Reads a file resource.
Iterator<Record > readResourceTable(String resourceName)	Reads a table resource.
Counter getCounter(Enum<? > name)	Obtains the counter with the specified name.
Counter getCounter(String group, String name)	Obtains the counter with the specified name in the specified group.
void progress()	Sends heartbeat information to the MapReduce framework. If your task takes an extended period of time to process data and you do not need to call the framework during this time period, you can call this method to avoid a task timeout. The default timeout period for a task is 600 seconds.

- If a worker runs for a long period of time and the framework determines that the worker times out, the framework stops the worker. In this case, you can call the progress method of the TaskContext class to prevent a worker from being stopped by MapReduce. The progress method sends heartbeat information to the framework. The progress method is not used to report the worker progress.
- The default timeout period for a worker is 10 minutes in MaxCompute MapReduce. You cannot change the timeout period. If a worker does not send heartbeat information by calling the progress method in 10 minutes, the framework terminates the worker and the map or reduce task fails. Therefore, we recommend that you periodically call the progress method in a map or reduce task to prevent the framework from terminating workers unexpectedly.

JobConf

The following table describes the methods of the JobConf class.

Method	Description
void setResources(String resourceNames)	Declares resources that are used in the current job. A mapper or reducer can read only the resources that have been declared in the TaskContext object.
void setMapOutputKeySchema(Column[] schema)	Sets the attributes of keys that are passed from the mapper to the reducer.
void setMapOutputValueSchema(Column[] schema)	Sets the attributes of values that are passed from the mapper to the reducer.
void setOutputKeySortColumns(String[] cols)	Sets the columns for sorting the keys that are passed from the mapper to the reducer.
void setOutputGroupingColumns(String[] cols)	Sets the columns for grouping the keys.
void setMapperClass(Class<? extends Mapper > theClass)	Sets a mapper for a job.

Method	Description
void setPartitionColumns(String[] cols)	Sets the partition key columns for a job. By default, the partition key columns are all columns of the keys that are generated by the mapper.
void setReducerClass(Class<? extends Reducer > theClass)	Sets a reducer for a job.
void setCombinerClass(Class<? extends Reducer > theClass)	Sets a combiner for a job. A combiner combines records with the same key. It is similar to a reducer but works at the map stage.
void setSplitSize(long size)	Sets the split size, in MB. The default split size is 256 MB.
void setNumReduceTasks(int n)	Sets the number of reduce tasks. By default, the number of reduce tasks is one-fourth of the number of map tasks.
void setMemoryForMapTask(int mem)	Sets the memory available to a worker in a map task, in MB. The default memory size is 2048 MB.
void setMemoryForReduceTask(int mem)	Sets the memory available to a worker in a reduce task, in MB. The default memory size is 2048 MB.

- The grouping columns are selected from the sort columns. The sort columns and partition key columns must exist in keys.
- At the map stage, the hash values of records from a mapper are calculated based on the specified partition key columns. The hash values help determine the reducers to which records are passed. The records are sorted based on the sort columns before the records are passed to reducers.
- At the reduce stage, input records are grouped based on the grouping columns. Then, a group of records that share the same key are passed to the reduce method as one input.

JobClient

The following table describes the methods of the JobClient class.

Method	Description
static RunningJob runJob(JobConf job)	Submits a MapReduce job in blocking mode and returns a RunningJob object.
static RunningJob submitJob(JobConf job)	Submits a MapReduce job in non-blocking mode and returns a RunningJob object.

RunningJob

The following table describes the methods of the RunningJob class.

Method	Description

Method	Description
String getInstanceID()	Obtains the ID of a job instance. You can use the job instance ID to view operational logs and manage jobs.
boolean isComplete()	Checks whether a job is completed.
boolean isSuccessful()	Checks whether a job instance is successful.
void waitForCompletion()	Waits for a job instance to end. The method is used for jobs that are submitted in asynchronous mode.
JobStatus getJobStatus()	Checks the running status of a job instance.
void killJob()	Ends the current job.
Counters getCounters()	Obtains the counter information.

InputUtils

The following table describes the methods of the InputUtils class.

Method	Description
static void addTable(TableInfo table, JobConf conf)	Adds an input table to a task. The method can be called multiple times. New tables are appended to the input queue.
static void setTables(TableInfo [] tables, JobConf conf)	Adds multiple input tables to a task.

OutputUtils

The following table describes the methods of the OutputUtils class.

Method	Description
static void addTable(TableInfo table, JobConf conf)	Adds an output table to a task. The method can be called multiple times. New tables are appended to the output queue.
static void setTables(TableInfo[] tables, JobConf conf)	Adds multiple output tables to a task.

Pipeline

Pipeline is the main class of the extended MapReduce model. You can call the Pipeline.builder method to build a pipeline. The following code shows the methods of the Pipeline class:

```
public Builder addMapper(Class<? extends Mapper> mapper)
public Builder addMapper(Class<? extends Mapper> mapper,
       Column[] keySchema, Column[] valueSchema, String[] sortCols,
       SortOrder[] order, String[] partCols,
       Class<? extends Partitioner> theClass, String[] groupCols)
public Builder addReducer(Class<? extends Reducer> reducer)
public Builder addReducer(Class<? extends Reducer> reducer,
       Column[] keySchema, Column[] valueSchema, String[] sortCols,
       SortOrder[] order, String[] partCols,
       Class<? extends Partitioner> theClass, String[] groupCols)
public Builder setOutputKeySchema(Column[] keySchema)
public Builder setOutputValueSchema(Column[] valueSchema)
public Builder setOutputKeySortColumns(String[] sortCols)
public Builder setOutputKeySortOrder(SortOrder[] order)
public Builder setPartitionColumns(String[] partCols)
public Builder setPartitionerClass(Class<? extends Partitioner> theClass)
public Builder setOutputGroupingColumns(String[] cols)
```

The following example shows how to call the Pipeline.builder method to build a pipeline:

```
Job job = new Job();
Pipeline pipeline = Pipeline.builder()
 .addMapper(TokenizerMapper.class)
 .setOutputKeySchema(
     new Column[] { new Column("word", OdpsType.STRING) })
 .setOutputValueSchema(
     new Column[] { new Column("count", OdpsType.BIGINT) })
 .addReducer(SumReducer.class)
 .setOutputKeySchema(
     new Column[] { new Column("count", OdpsType.BIGINT) })
 .setOutputValueSchema(
     new Column[] { new Column("word", OdpsType.STRING),
     new Column("count", OdpsType.BIGINT) })
 .addReducer(IdentityReducer.class).createPipeline();
job.setPipeline(pipeline);
job.addInput(...)
job.addOutput(...)
job.submit();
```

As shown in the preceding example, you can create a MapReduce job in which a mapper is followed by two reducers in the main function. If you are familiar with the basic features of MapReduce, the extended MapReduce model is easy to use.

> **Note**
>
> - Before you use the extended MapReduce model, we recommend that you learn how to use MapReduce.
> - You can create a MapReduce job in which a mapper is followed by only one reducer by using JobConf.

4.2.6.2. Compatibility with Hadoop MapReduce

This topic describes the compatibility between specific MaxCompute MapReduce interfaces and Hadoop MapReduce.

The following table describes whether specific MaxCompute MapReduce interfaces are compatible with Hadoop MapReduce.

Type	Interface	Compatible with Hadoop MapReduce
Mapper	void map(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Mapper.Context context)	Yes.
Mapper	void run(org.apache.hadoop.mapreduce.Mapper.Context context)	Yes.
Mapper	void setup(org.apache.hadoop.mapreduce.Mapper.Context context)	Yes.
Reducer	void cleanup(org.apache.hadoop.mapreduce.Reducer.Context context)	Yes.
Reducer	void reduce(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Reducer.Context context)	Yes.
Reducer	void run(org.apache.hadoop.mapreduce.Reducer.Context context)	Yes.
Reducer	void setup(org.apache.hadoop.mapreduce.Reducer.Context context)	Yes.
Partitioner	int getPartition(KEY key, VALUE value, int numPartitions)	Yes.
MapContext, which extends TaskInputOutputContext	InputSplit getInputSplit()	No. Exceptions occur.
ReduceContext	nextKey()	Yes.
ReduceContext	getValues()	Yes.
TaskInputOutputContext	getCurrentKey()	Yes.
TaskInputOutputContext	getCurrentValue()	Yes.
TaskInputOutputContext	getOutputCommitter()	No. Exceptions occur.
TaskInputOutputContext	nextKeyValue()	Yes.
TaskInputOutputContext	write(KEYOUT key, VALUEOUT value)	Yes.

Type	Interface	Compatible with Hadoop MapReduce
TaskAttemptContext	getCounter(Enum<? > counterName)	Yes.
TaskAttemptContext	getCounter(String groupName, String counterName)	Yes.
TaskAttemptContext	setStatus(String msg)	Empty implementation.
TaskAttemptContext	getStatus()	Empty implementation.
TaskAttemptContext	getTaskAttemptID()	No. Exceptions occur.
TaskAttemptContext	getProgress()	No. Exceptions occur.
TaskAttemptContext	progress()	Yes.
Job	addArchiveToClassPath(Path archive)	No.
Job	addCacheArchive(URI uri)	No.
Job	addCacheFile(URI uri)	No.
Job	addFileToClassPath(Path file)	No.
Job	cleanupProgress()	No.
Job	createSymlink()	No. Exceptions occur.
Job	failTask(TaskAttemptID taskId)	No.
Job	getCompletionPollInterval(Configuration conf)	Empty implementation.
Job	getCounters()	Yes.
Job	getFinishTime()	Yes.
Job	getHistoryUrl()	Yes.

Type	Interface	Compatible with Hadoop MapReduce
Job	getInstance()	Yes.
Job	getInstance(Cluster ignored)	Yes.
Job	getInstance(Cluster ignored, Configuration conf)	Yes.
Job	getInstance(Configuration conf)	Yes.
Job	getInstance(Configuration conf, String jobName)	Empty implementation.
Job	getInstance(JobStatus status, Configuration conf)	No. Exceptions occur.
Job	getJobFile()	No. Exceptions occur.
Job	getJobName()	Empty implementation.
Job	getJobState()	No. Exceptions occur.
Job	getPriority()	No. Exceptions occur.
Job	getProgressPollInterval(Configuration conf)	Empty implementation.
Job	getReservationId()	No. Exceptions occur.
Job	getSchedulingInfo()	No. Exceptions occur.
Job	getStartTime()	Yes.
Job	getStatus()	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
Job	getTaskCompletionEvents(int startFrom)	No. Exceptions occur.
Job	getTaskCompletionEvents(int startFrom, int numEvents)	No. Exceptions occur.
Job	getTaskDiagnostics(TaskAttemptID taskid)	No. Exceptions occur.
Job	getTaskOutputFilter(Configuration conf)	No. Exceptions occur.
Job	getTaskReports(TaskType type)	No. Exceptions occur.
Job	getTrackingURL()	Yes.
Job	isComplete()	Yes.
Job	isRetired()	No. Exceptions occur.
Job	isSuccessful()	Yes.
Job	isUber()	Empty implementation.
Job	killJob()	Yes.
Job	killTask(TaskAttemptID taskId)	No.
Job	mapProgress()	Yes.
Job	monitorAndPrintJob()	Yes.
Job	reduceProgress()	Yes.
Job	setCacheArchives(URI[] archives)	No. Exceptions occur.
Job	setCacheFiles(URI[] files)	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
Job	setCancelDelegationTokenUponJobCompletion(boolean value)	No. Exceptions occur.
Job	setCombinerClass(Class<? extends Reducer> cls)	Yes.
Job	setCombinerKeyGroupingComparatorClass(Class<? extends RawComparator> cls)	Yes.
Job	setGroupingComparatorClass(Class<? extends RawComparator> cls)	Yes.
Job	setInputFormatClass(Class<? extends InputFormat> cls)	Empty implementation.
Job	setJar(String jar)	Yes.
Job	setJarByClass(Class<? > cls)	Yes.
Job	setJobName(String name)	Empty implementation.
Job	setJobSetupCleanupNeeded(boolean needed)	Empty implementation.
Job	setMapOutputKeyClass(Class<? > theClass)	Yes.
Job	setMapOutputValueClass(Class<? > theClass)	Yes.
Job	setMapperClass(Class<? extends Mapper> cls)	Yes.
Job	setMapSpeculativeExecution(boolean speculativeExecution)	Empty implementation.
Job	setMaxMapAttempts(int n)	Empty implementation.
Job	setMaxReduceAttempts(int n)	Empty implementation.
Job	setNumReduceTasks(int tasks)	Yes.

Type	Interface	Compatible with Hadoop MapReduce
Job	setOutputFormatClass(Class<? extends OutputFormat> cls)	No. Exceptions occur.
Job	setOutputKeyClass(Class<? > theClass)	Yes.
Job	setOutputValueClass(Class<? > theClass)	Yes.
Job	setPartitionerClass(Class<? extends Partitioner> cls)	Yes.
Job	setPriority(JobPriority priority)	No. Exceptions occur.
Job	setProfileEnabled(boolean newValue)	Empty implementation.
Job	setProfileParams(String value)	Empty implementation.
Job	setProfileTaskRange(boolean isMap, String newValue)	Empty implementation.
Job	setReducerClass(Class<? extends Reducer> cls)	Yes.
Job	setReduceSpeculativeExecution(boolean speculativeExecution)	Empty implementation.
Job	setReservationId(ReservationId reservationId)	No. Exceptions occur.
Job	setSortComparatorClass(Class<? extends RawComparator> cls)	No. Exceptions occur.
Job	setSpeculativeExecution(boolean speculativeExecution)	Yes.
Job	setTaskOutputFilter(Configuration conf, org.apache.hadoop.mapreduce.Job.TaskStatusFilter newValue)	No. Exceptions occur.
Job	setupProgress()	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
Job	setUser(String user)	Empty implementation.
Job	setWorkingDirectory(Path dir)	Empty implementation.
Job	submit()	Yes.
Job	toString()	No. Exceptions occur.
Job	waitForCompletion(boolean verbose)	Yes.
Task Execution & Environment	mapreduce.map.java.opts	Empty implementation.
Task Execution & Environment	mapreduce.reduce.java.opts	Empty implementation.
Task Execution & Environment	mapreduce.map.memory.mb	Empty implementation.
Task Execution & Environment	mapreduce.reduce.memory.mb	Empty implementation.
Task Execution & Environment	mapreduce.task.io.sort.mb	Empty implementation.
Task Execution & Environment	mapreduce.map.sort.spill.percent	Empty implementation.
Task Execution & Environment	mapreduce.task.io.soft.factor	Empty implementation.
Task Execution & Environment	mapreduce.reduce.merge.inmem.thresholds	Empty implementation.
Task Execution & Environment	mapreduce.reduce.shuffle.merge.percent	Empty implementation.

Type	Interface	Compatible with Hadoop MapReduce
Task Execution & Environment	mapreduce.reduce.shuffle.input.buffer.percent	Empty implementation.
Task Execution & Environment	mapreduce.reduce.input.buffer.percent	Empty implementation.
Task Execution & Environment	mapreduce.job.id	Empty implementation.
Task Execution & Environment	mapreduce.job.jar	Empty implementation.
Task Execution & Environment	mapreduce.job.local.dir	Empty implementation.
Task Execution & Environment	mapreduce.task.id	Empty implementation.
Task Execution & Environment	mapreduce.task.attempt.id	Empty implementation.
Task Execution & Environment	mapreduce.task.is.map	Empty implementation.
Task Execution & Environment	mapreduce.task.partition	Empty implementation.
Task Execution & Environment	mapreduce.map.input.file	Empty implementation.
Task Execution & Environment	mapreduce.map.input.start	Empty implementation.
Task Execution & Environment	mapreduce.map.input.length	Empty implementation.
Task Execution & Environment	mapreduce.task.output.dir	Empty implementation.

Type	Interface	Compatible with Hadoop MapReduce
JobClient	cancelDelegationToken(Token <DelegationTokenIdentifier> token)	No. Exceptions occur.
JobClient	close()	Empty implementation.
JobClient	displayTasks(JobID jobId, String type, String state)	No. Exceptions occur.
JobClient	getAllJobs()	No. Exceptions occur.
JobClient	getCleanupTaskReports(JobID jobId)	No. Exceptions occur.
JobClient	getClusterStatus()	No. Exceptions occur.
JobClient	getClusterStatus(boolean detailed)	No. Exceptions occur.
JobClient	getDefaultMaps()	No. Exceptions occur.
JobClient	getDefaultReduces()	No. Exceptions occur.
JobClient	getDelegationToken(Text renewer)	No. Exceptions occur.
JobClient	getFs()	No. Exceptions occur.
JobClient	getJob(JobID jobid)	No. Exceptions occur.
JobClient	getJob(String jobid)	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
JobClient	getJobsFromQueue(String queueName)	No. Exceptions occur.
JobClient	getMapTaskReports(JobID jobId)	No. Exceptions occur.
JobClient	getMapTaskReports(String jobId)	No. Exceptions occur.
JobClient	getQueueAclsForCurrentUser()	No. Exceptions occur.
JobClient	getQueueInfo(String queueName)	No. Exceptions occur.
JobClient	getQueues()	No. Exceptions occur.
JobClient	getReduceTaskReports(JobID jobId)	No. Exceptions occur.
JobClient	getReduceTaskReports(String jobId)	No. Exceptions occur.
JobClient	getSetupTaskReports(JobID jobId)	No. Exceptions occur.
JobClient	getStagingAreaDir()	No. Exceptions occur.
JobClient	getSystemDir()	No. Exceptions occur.
JobClient	getTaskOutputFilter()	No. Exceptions occur.
JobClient	getTaskOutputFilter(JobConf job)	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
JobClient	init(JobConf conf)	No. Exceptions occur.
JobClient	isJobDirValid(Path jobDirPath, FileSystem fs)	No. Exceptions occur.
JobClient	jobsToComplete()	No. Exceptions occur.
JobClient	monitorAndPrintJob(JobConf conf, RunningJob job)	No. Exceptions occur.
JobClient	renewDelegationToken(Token<DelegationTokenIdentifier> token)	No. Exceptions occur.
JobClient	run(String[] argv)	No. Exceptions occur.
JobClient	runJob(JobConf job)	Yes.
JobClient	setTaskOutputFilter(JobClient.TaskStatusFilter newValue)	No. Exceptions occur.
JobClient	setTaskOutputFilter(JobConf job, JobClient.TaskStatusFilter newValue)	No. Exceptions occur.
JobClient	submitJob(JobConf job)	Yes.
JobClient	submitJob(String jobFile)	No. Exceptions occur.
JobConf	deleteLocalFiles()	No. Exceptions occur.
JobConf	deleteLocalFiles(String subdir)	No. Exceptions occur.
JobConf	normalizeMemoryConfigValue(long val)	Empty implementation.

Type	Interface	Compatible with Hadoop MapReduce
JobConf	setCombinerClass(Class<? extends Reducer> theClass)	Yes.
JobConf	setCompressMapOutput(boolean compress)	Empty implementation.
JobConf	setInputFormat(Class<? extends InputFormat> theClass)	No. Exceptions occur.
JobConf	setJar(String jar)	No. Exceptions occur.
JobConf	setJarByClass(Class cls)	No. Exceptions occur.
JobConf	setJobEndNotificationURI(String uri)	No. Exceptions occur.
JobConf	setJobName(String name)	Empty implementation.
JobConf	setJobPriority(JobPriority prio)	No. Exceptions occur.
JobConf	setKeepFailedTaskFiles(boolean keep)	No. Exceptions occur.
JobConf	setKeepTaskFilesPattern(String pattern)	No. Exceptions occur.
JobConf	setKeyFieldComparatorOptions(String keySpec)	No. Exceptions occur.
JobConf	setKeyFieldPartitionerOptions(String keySpec)	No. Exceptions occur.
JobConf	setMapDebugScript(String mDbgScript)	Empty implementation.

Type	Interface	Compatible with Hadoop MapReduce
JobConf	setMapOutputCompressorClass(Class<? extends CompressionCodec> codecClass)	Empty implementation.
JobConf	setMapOutputKeyClass(Class<? > theClass)	Yes.
JobConf	setMapOutputValueClass(Class<? > theClass)	Yes.
JobConf	setMapperClass(Class<? extends Mapper> theClass)	Yes.
JobConf	setMapRunnerClass(Class<? extends MapRunnable> theClass)	No. Exceptions occur.
JobConf	setMapSpeculativeExecution(boolean speculativeExecution)	Empty implementation.
JobConf	setMaxMapAttempts(int n)	Empty implementation.
JobConf	setMaxMapTaskFailuresPercent(int percent)	Empty implementation.
JobConf	setMaxPhysicalMemoryForTask(long mem)	Empty implementation.
JobConf	setMaxReduceAttempts(int n)	Empty implementation.
JobConf	setMaxReduceTaskFailuresPercent(int percent)	Empty implementation.
JobConf	setMaxTaskFailuresPerTracker(int noFailures)	Empty implementation.
JobConf	setMaxVirtualMemoryForTask(long vmem)	Empty implementation.
JobConf	setMemoryForMapTask(long mem)	Yes.
JobConf	setMemoryForReduceTask(long mem)	Yes.
JobConf	setNumMapTasks(int n)	Yes.

Type	Interface	Compatible with Hadoop MapReduce
JobConf	setNumReduceTasks(int n)	Yes.
JobConf	setNumTasksToExecutePerJvm(int numTasks)	Empty implementation.
JobConf	setOutputCommitter(Class<? extends OutputCommitter> theClass)	No. Exceptions occur.
JobConf	setOutputFormat(Class<? extends OutputFormat> theClass)	Empty implementation.
JobConf	setOutputKeyClass(Class<? > theClass)	Yes.
JobConf	setOutputKeyComparatorClass(Class<? extends RawComparator> theClass)	No. Exceptions occur.
JobConf	setOutputValueClass(Class<? > theClass)	Yes.
JobConf	setOutputValueGroupingComparator(Class<? extends RawComparator> theClass)	No. Exceptions occur.
JobConf	setPartitionerClass(Class<? extends Partitioner> theClass)	Yes.
JobConf	setProfileEnabled(boolean newValue)	Empty implementation.
JobConf	setProfileParams(String value)	Empty implementation.
JobConf	setProfileTaskRange(boolean isMap, String newValue)	Empty implementation.
JobConf	setQueueName(String queueName)	No. Exceptions occur.
JobConf	setReduceDebugScript(String rDbgScript)	Empty implementation.
JobConf	setReducerClass(Class<? extends Reducer> theClass)	Yes.

Type	Interface	Compatible with Hadoop MapReduce
JobConf	setReduceSpeculativeExecution(boolean speculativeExecution)	Empty implementation.
JobConf	setSessionId(String sessionId)	Empty implementation.
JobConf	setSpeculativeExecution(boolean speculativeExecution)	No. Exceptions occur.
JobConf	setUseNewMapper(boolean flag)	Yes.
JobConf	setUseNewReducer(boolean flag)	Yes.
JobConf	setUser(String user)	Empty implementation.
JobConf	setWorkingDirectory(Path dir)	Empty implementation.
FileInputFormat	N/A	No. Exceptions occur.
TextInputFormat	N/A	Yes.
InputSplit	mapred.min.split.size.	No. Exceptions occur.
FileSplit	map.input.file	No. Exceptions occur.
RecordWriter	N/A	No. Exceptions occur.
RecordReader	N/A	No. Exceptions occur.
OutputFormat	N/A	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
OutputCommitter	abortJob(JobContext jobContext, int status)	No. Exceptions occur.
OutputCommitter	abortJob(JobContext context, JobStatus.State runState)	No. Exceptions occur.
OutputCommitter	abortTask(TaskAttemptContext taskContext)	No. Exceptions occur.
OutputCommitter	abortTask(TaskAttemptContext taskContext)	No. Exceptions occur.
OutputCommitter	cleanupJob(JobContext jobContext)	No. Exceptions occur.
OutputCommitter	cleanupJob(JobContext context)	No. Exceptions occur.
OutputCommitter	commitJob(JobContext jobContext)	No. Exceptions occur.
OutputCommitter	commitJob(JobContext context)	No. Exceptions occur.
OutputCommitter	commitTask(TaskAttemptContext taskContext)	No. Exceptions occur.
OutputCommitter	needsTaskCommit(TaskAttemptContext taskContext)	No. Exceptions occur.
OutputCommitter	needsTaskCommit(TaskAttemptContext taskContext)	No. Exceptions occur.
OutputCommitter	setupJob(JobContext jobContext)	No. Exceptions occur.
OutputCommitter	setupJob(JobContext jobContext)	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
OutputCommitter	setupTask(TaskAttemptContext taskContext)	No. Exceptions occur.
OutputCommitter	setupTask(TaskAttemptContext taskContext)	No. Exceptions occur.
Counter	getDisplayName()	Yes.
Counter	getName()	Yes.
Counter	getValue()	Yes.
Counter	increment(long incr)	Yes.
Counter	setValue(long value)	Yes.
Counter	setDisplayName(String displayName)	Yes.
DistributedCache	CACHE_ARCHIVES	No. Exceptions occur.
DistributedCache	CACHE_ARCHIVES_SIZES	No. Exceptions occur.
DistributedCache	CACHE_ARCHIVES_TIMESTAMPS	No. Exceptions occur.
DistributedCache	CACHE_FILES	No. Exceptions occur.
DistributedCache	CACHE_FILES_SIZES	No. Exceptions occur.
DistributedCache	CACHE_FILES_TIMESTAMPS	No. Exceptions occur.
DistributedCache	CACHE_LOCALARCHIVES	No. Exceptions occur.
DistributedCache	CACHE_LOCALFILES	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
DistributedCache	CACHE_SYMLINK	No. Exceptions occur.
DistributedCache	addArchiveToClassPath(Path archive, Configuration conf)	No. Exceptions occur.
DistributedCache	addArchiveToClassPath(Path archive, Configuration conf, FileSystem fs)	No. Exceptions occur.
DistributedCache	addCacheArchive(URI uri, Configuration conf)	No. Exceptions occur.
DistributedCache	addCacheFile(URI uri, Configuration conf)	No. Exceptions occur.
DistributedCache	addFileToClassPath(Path file, Configuration conf)	No. Exceptions occur.
DistributedCache	addFileToClassPath(Path file, Configuration conf, FileSystem fs)	No. Exceptions occur.
DistributedCache	addLocalArchives(Configuration conf, String str)	No. Exceptions occur.
DistributedCache	addLocalFiles(Configuration conf, String str)	No. Exceptions occur.
DistributedCache	checkURIs(URI[] uriFiles, URI[] uriArchives)	No. Exceptions occur.
DistributedCache	createAllSymlink(Configuration conf, File jobCacheDir, File workDir)	No. Exceptions occur.
DistributedCache	createSymlink(Configuration conf)	No. Exceptions occur.
DistributedCache	getArchiveClassPaths(Configuration conf)	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
DistributedCache	getArchiveTimestamps(Configuration conf)	No. Exceptions occur.
DistributedCache	getCacheArchives(Configuration conf)	No. Exceptions occur.
DistributedCache	getCacheFiles(Configuration conf)	No. Exceptions occur.
DistributedCache	getFileClassPaths(Configuration conf)	No. Exceptions occur.
DistributedCache	getFileStatus(Configuration conf, URI cache)	No. Exceptions occur.
DistributedCache	getFileTimestamps(Configuration conf)	No. Exceptions occur.
DistributedCache	getLocalCacheArchives(Configuration conf)	No. Exceptions occur.
DistributedCache	getLocalCacheFiles(Configuration conf)	No. Exceptions occur.
DistributedCache	getSymlink(Configuration conf)	No. Exceptions occur.
DistributedCache	getTimestamp(Configuration conf, URI cache)	No. Exceptions occur.
DistributedCache	setArchiveTimestamps(Configuration conf, String timestamps)	No. Exceptions occur.
DistributedCache	setCacheArchives(URI[] archives, Configuration conf)	No. Exceptions occur.
DistributedCache	setCacheFiles(URI[] files, Configuration conf)	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
DistributedCache	setFileTimestamps(Configuration conf, String timestamps)	No. Exceptions occur.
DistributedCache	setLocalArchives(Configuration conf, String str)	No. Exceptions occur.
DistributedCache	setLocalFiles(Configuration conf, String str)	No. Exceptions occur.
IsolationRunner	N/A	No. Exceptions occur.
Profiling	N/A	Empty implementation.
Debugging	N/A	Empty implementation.
Data Compression	N/A	Yes.
Skipping Bad Records	N/A	No. Exceptions occur.
Job Authorization	mapred.acls.enabled	No. Exceptions occur.
Job Authorization	mapreduce.job.acl-view-job	No. Exceptions occur.
Job Authorization	mapreduce.job.acl-modify-job	No. Exceptions occur.
Job Authorization	mapreduce.cluster.administrators	No. Exceptions occur.
Job Authorization	mapred.queue.queue-name.acl-administer-jobs	No. Exceptions occur.

Type	Interface	Compatible with Hadoop MapReduce
MultipleInputs	N/A	No. Exceptions occur.
Multi{anchor:_Go Back}pleOutputs	N/A	Yes.
org.apache.hadoop.mapreduce.lib.db	N/A	No. Exceptions occur.
org.apache.hadoop.mapreduce.security	N/A	No. Exceptions occur.
org.apache.hadoop.mapreduce.lib.jobcontrol	N/A	No. Exceptions occur.
org.apache.hadoop.mapreduce.lib.chain	N/A	No. Exceptions occur.
org.apache.hadoop.mapreduce.lib.db	N/A	No. Exceptions occur.

4.2.6.3. Java sandbox

MaxCompute MapReduce and user-defined functions (UDFs) that run in distributed mode are limited by Java sandboxes. The main function of MapReduce is not subject to the limits of Java sandboxes.

Limits on the use of Java sandboxes

- Direct access to local files is not allowed. You can access files only by using interfaces provided by MaxCompute MapReduce and MaxCompute Graph. This limit applies to the following data:
  - Resources specified by the resources option, including resource files, JAR packages, and resource tables.
  - Logs generated by System.out and System.err. You can run the Log command on the MaxCompute client to view logs.
- Direct access to distributed file systems is not allowed. You can use only MaxCompute MapReduce or MaxCompute Graph to access tables.
- Java Native Interface (JNI) calls are not allowed.
- Java threads cannot be created. Linux commands cannot be executed by using sub-threads.
- Network access operations, such as acquiring local IP addresses, are not allowed.
- Limits on Java reflection: The suppressAccessChecks permission is prohibited. You cannot set a private

attribute or method accessible to read private attributes or call private methods.

Limits on access to local files

If you use the following methods to access a local file, the `access denied` exception is reported:

- java.io.File

```
public boolean delete()
public void deleteOnExit()
public boolean exists()
public boolean canRead()
public boolean isFile()
public boolean isDirectory()
public boolean isHidden()
public long lastModified()
public long length()
public String[] list()
public String[] list(FilenameFilter filter)
public File[] listFiles()
public File[] listFiles(FilenameFilter filter)
public File[] listFiles(FileFilter filter)
public boolean canWrite()
public boolean createNewFile()
public static File createTempFile(String prefix, String suffix)
public static File createTempFile(String prefix,  String suffix,File directory)
public boolean mkdir()
public boolean mkdirs()
public boolean renameTo(File dest)
public boolean setLastModified(long time)
public boolean setReadOnly()
```

- java.io.RandomAccessFile

```
RandomAccessFile(String name, String mode)
RandomAccessFile(File file, String mode)
```

- java.io.FileInputStream

```
FileInputStream(FileDescriptor fdObj)
FileInputStream(String name)
FileInputStream(File file)
```

- java.io.FileOutputStream

```
FileOutputStream(FileDescriptor fdObj)
FileOutputStream(File file)
FileOutputStream(String name)
FileOutputStream(String name, boolean append)
```

- java.lang.Class

```
public ProtectionDomain getProtectionDomain()
```

- java.lang.ClassLoader

```
ClassLoader()
ClassLoader(ClassLoader parent)
```

- java.lang.Runtime

```
public Process exec(String command)
public Process exec(String command, String envp[])
public Process exec(String cmdarray[])
public Process exec(String cmdarray[], String envp[])
public void exit(int status)
public static void runFinalizersOnExit(boolean value)
public void addShutdownHook(Thread hook)
public boolean removeShutdownHook(Thread hook)
public void load(String lib)
public void loadLibrary(String lib)
```

- java.lang.System

```
public static void exit(int status)
public static void runFinalizersOnExit(boolean value)
public static void load(String filename)
public static void loadLibrary( String libname)
public static Properties getProperties()
public static void setProperties(Properties props)
public static String getProperty(String key) // Only some keys are allowed for access to
the file.
public static String getProperty(String key, String def) // Only some keys are allowed fo
r access to the file.
public static String setProperty(String key, String value)
public static void setIn(InputStream in)
public static void setOut(PrintStream out)
public static void setErr(PrintStream err)
public static synchronized void setSecurityManager(SecurityManager s)
```

Keys supported by System.getProperty:

```
java.version
java.vendor
java.vendor.url
java.class.version
os.name
os.version
os.arch
file.separator
path.separator
line.separator
java.specification.version
java.specification.vendor
java.specification.name
java.vm.specification.version
java.vm.specification.vendor
java.vm.specification.name
java.vm.version
java.vm.vendor
java.vm.name
file.encoding
user.timezone
```

- java.lang.Thread

```
Thread()
Thread(Runnable target)
Thread(String name)
Thread(Runnable target, String name)
Thread(ThreadGroup group, ...)
public final void checkAccess()
public void interrupt()
public final void suspend()
public final void resume()
public final void setPriority (int newPriority)
public final void setName(String name)
public final void setDaemon(boolean on)
public final void stop()
public final synchronized void stop(Throwable obj)
public static int enumerate(Thread tarray[])
public void setContextClassLoader(ClassLoader cl)
```

- java.lang.ThreadGroup

```
ThreadGroup(String name)
ThreadGroup(ThreadGroup parent, String name)
public final void checkAccess()
public int enumerate(Thread list[])
public int enumerate(Thread list[], boolean recurse)
public int enumerate(ThreadGroup list[])
public int enumerate(ThreadGroup list[], boolean recurse)
public final ThreadGroup getParent()
public final void setDaemon(boolean daemon)
public final void setMaxPriority(int pri)
public final void suspend()
public final void resume()
public final void destroy()
public final void interrupt()
public final void stop()
```

- java.lang.reflect.AccessibleObject

```
public static void setAccessible(...)
public void setAccessible(...)
```

- java.net.InetAddress

```
public String getHostName()
public static InetAddress[] getAllByName(String host)
public static InetAddress getLocalHost()
```

- java.net.DatagramSocket

```
public InetAddress getLocalAddress()
```

- java.net.Socket

```
Socket(...)
```

- java.net.ServerSocket

```
ServerSocket(...)
public Socket accept()
protected final void implAccept(Socket s)
public static synchronized void setSocketFactory(...)
public static synchronized void setSocketImplFactory(...)
```

- java.net.DatagramSocket

```
DatagramSocket(...)
public synchronized void receive(DatagramPacket p)
```

- java.net.MulticastSocket

```
MulticastSocket(...)
```

- java.net.URL

```
URL(...)
public static synchronized void setURLStreamHandlerFactory(...)
java.net.URLConnection
public static synchronized void setContentHandlerFactory(...)
public static void setFileNameMap(FileNameMap map)
```

- java.net.HttpURLConnection

```
public static void setFollowRedirects(boolean set)
java.net.URLClassLoader
URLClassLoader(...)
```

- java.security.AccessControlContext

```
public AccessControlContext(AccessControlContext acc, DomainCombiner combiner)
public DomainCombiner getDomainCombiner()
```

4.2.7. FAQ about MaxCompute MapReduce

This topic provides answers to some frequently asked questions about MaxCompute MapReduce.

Category	FAQ
Features	• Can I use views as the input sources of MapReduce jobs in MaxCompute? • In which mode are the results of MapReduce jobs written to a table or partition? • Can I run a MapReduce job by calling shell files? • Can I call the setup method of a reducer to read data from input tables? • Does a mapper support data from multiple partitions of a table as input? • Can a mapper read partition fields from data records? • What is the relationship between labels and partitions? • Does MaxCompute MapReduce support map-only jobs? • Can I use a mapper to read each record from an input table by column name? • What are the differences between write(Record key, Record value) and write(Record record)? • Why do I need to use both of the -libjars and -classpath parameters to specify the JAR package of a MapReduce program in MaxCompute MapReduce? • Can I directly use the source code of Hadoop MapReduce in MaxCompute MapReduce? • How do I use MaxCompute MapReduce to sort data? • What is the purpose of backups for MapReduce jobs?

Category	FAQ
MapReduce program development	• How do I pass multiple resources on the MaxCompute client when I develop a MapReduce program? • How do I use the main method to determine whether a table is an empty table? • How do I generate logs of MapReduce jobs in MaxCompute? • Does a result table contain the duplicate data of two MapReduce jobs? • In Hadoop MapReduce, I can select multiple nodes for distributed data processing. One node represents one machine. How do I configure nodes when I perform distributed data processing in MaxCompute MapReduce? • If I do not use a combiner, the data output is normal. After I use a combiner, no input data is provided for reducers. Why? • I cannot specify the schema of an output table when I run map-only jobs in MaxCompute. Why? • How do I call the local MaxCompute server to run a MapReduce job?

Category	FAQ
Common errors	• What do I do if the error message "BufferOverflowException" appears when I run a MapReduce job in MaxCompute? • What do I do if the error message "Resource not found" appears when I run a MapReduce job in MaxCompute? • What do I do if the error message "Class Not Found" appears when I run a MapReduce job in MaxCompute? • What do I do if the error ODPS-0010000 is returned when I run a MapReduce job in MaxCompute? • What do I do if the error message "Table not found" appears when I run a MapReduce job in MaxCompute? • What do I do if the error ODPS-0123144 is returned when I run a MapReduce job in MaxCompute? • What do I do if the error message "java.security.AccessControlException" appears when I run a MapReduce job in MaxCompute? • What do I do if the error message "java.io.IOException" appears when I run a MapReduce job in MaxCompute? • What do I do if the error message "Exceed maximum read times per resource" appears when I run a MapReduce job in MaxCompute? • What do I do if an OOM error occurs before the Reduce stage when I run a MapReduce job in MaxCompute? • What do I do if an OOM error occurs when I run a MapReduce job in MaxCompute? • What do I do if the error message "java.lang.OutOfMemoryError" appears when I use 600 reducers to load a small configuration file for a MapReduce job in MaxCompute? • What do I do if the error ODPS-0420095 is returned when I run a MapReduce job in MaxCompute? • What do I do if a large number of errors occur when MaxCompute resources are referenced by a MapReduce job? • I create a JAR package of the MapReduce program that contains third-party classes by using Maven Assembly. When I run a MapReduce job, an error message, indicating that the third-party classes are not found, appears. What do I do? • When I run a Hadoop MapReduce job in MaxCompute, an error of subscript out of bounds occurs. What do I do?

Can I use views as the input sources of MapReduce jobs in MaxCompute?

No, you can use only tables as the input sources of MapReduce jobs in MaxCompute.

In which mode are the results of MapReduce jobs written to a table or partition?

The results of MapReduce jobs are written to a table or partition in overwrite mode.

Can I run a MapReduce job by calling shell files?

No, you cannot run a MapReduce job by calling shell files due to the limits of Java sandboxes. For more information about the limits of Java sandboxes, see Java sandbox.

Can I call the setup method of a reducer to read data from input tables?

No, you cannot call the setup method of a reducer to read data from input tables. However, you can call the setup method of a reducer to read data from cached tables.

Does a mapper support data from multiple partitions of a table as input?

A mapper supports data from multiple partitions of a table as input. The partitions of the same table are considered separate tables.

Can a mapper read partition fields from data records?

A mapper cannot read partition fields from data records. You can use the following code to obtain partition fields. In the code, PartitionSpec specifies the partition information.

```
PartitionSpec ps = context.getInputTableInfo().getPartitionSpec();
String area = ps.get("area");
```

What is the relationship between labels and partitions?

Labels are used to identify the partitions to which output data is written.

Does MaxCompute MapReduce support map-only jobs?

Yes, MaxCompute MapReduce supports map-only jobs. If you want to run map-only jobs in MaxCompute MapReduce, you must set the number of reducers to 0 by using `job.setNumReduceTasks(0)`.

Can I use a mapper to read each record from an input table by column name?

Yes, you can use a mapper to read each record from an input table by column name. Each record in an input table can be read by sequence number, such as `record.get(i)`, or read by column name, such as `record.get("size")`.

What are the differences between `write(Record key, Record value)` and `write(Record record)`?

- `write(Record key, Record value)` is used to generate intermediate results, such as `key.set("id", v1), value.set("size", v2)`. The intermediate results that are generated by a mapper must be transmitted to a reducer by using network connections. No associated tables are provided to infer the data types of fields. Therefore, the data types of fields must be declared for serialization. The data types of fields must be the data types that are supported by MaxCompute.

  ```
  job.setMapOutputKeySchema(SchemaUtils.fromString("id:string"));
  job.setMapOutputValueSchema(SchemaUtils.fromString("size:bigint"));
  ```

- `write(Record record)` is used to write final results to an output table. Associated tables are provided to infer the data types of fields. Therefore, the data types of fields do not need to be declared.

Why do I need to use both of the -libjars and -classpath parameters to specify the JAR package of a MapReduce program in MaxCompute MapReduce?

The local client performs operations that involve remote execution, such as job configurations. Therefore, two executors are used: a local executor and a remote executor.

The remote executor loads the package that is specified by the -libjars parameter, such as `-libjars mapreduce-examples.jar`. The local executor loads the package that is specified by the -classpath parameter, such as `-classpath lib/mapreduce-examples.jar`.

Can I directly use the source code of Hadoop MapReduce in MaxCompute MapReduce?

No, you cannot directly use the source code of Hadoop MapReduce in MaxCompute MapReduce. The APIs provided by MaxCompute MapReduce are different from the APIs provided by Hadoop MapReduce, but the overall style is similar. If you want to run the source code of Hadoop MapReduce in MaxCompute MapReduce, you must modify the source code of Hadoop MapReduce and compile the code by using MaxCompute MapReduce SDK.

How do I use MaxCompute MapReduce to sort data?

The following sample code shows how to sort data by using MaxCompute MapReduce.

```
// Specify the fields whose values you want to sort. In this example, the fields are i1 and
i2.
job.setOutputKeySortColumns(new String[] { "i1", "i2" });
// Specify how to sort the values of the two fields. In this example, the values of the i1
field are sorted in ascending order, and the values of the i2 field are sorted in descendin
g order.
job.setOutputKeySortOrder(new SortOrder[] { SortOrder.ASC, SortOrder.DESC });
```

The following sample code shows how to use the setOutputKeySortOrder method.

```
public void setOutputKeySortOrder(JobConf.SortOrder[] order)
Description: The setOutputKeySortOrder method is used to specify the order in which the val
ues of the key column are sorted.
Parameter: The order parameter specifies the order in which the values of the key column ar
e sorted. Valid values: ASC and DESC. ASC indicates the ascending order, and DESC indicates
the descending order.
```

What is the purpose of backups for MapReduce jobs?

Backups are used to accelerate data processing for MapReduce jobs. MaxCompute detects the status of your MapReduce jobs. If a MapReduce job needs to process a large amount of data, MaxCompute creates a backup job for the job. The two jobs are run in parallel to process the same data. The result of the job that is first complete is used. If a job needs to process excessively large amounts of data, backups cannot work as expected because the original job and its backup job may not be complete in a specified period of time.

How do I pass multiple resources on the MaxCompute client when I develop a MapReduce program?

You can separate multiple resources with commas (,), such as `jar -resource resource1,resource2,..` .

How do I use the main method to determine whether a table is an empty table?

You can use the following code to determine whether a table is an empty table.

```
Odps odps=SessionState.get().getOdps();
Table table=odps.tables().get('tableName');
RecordReader recordReader=table.read(1);
if(recordReader.read()==null){
//TO DO
```

How do I generate logs of MapReduce jobs in MaxCompute?

We recommend that you use one of the following methods to generate logs of MapReduce jobs in MaxCompute:

- Use `System.out.println` in the code to generate logs of MapReduce jobs in MaxCompute. The logs are exported to stdout of Logview.
- If an error occurs when you run a MapReduce job, the MaxCompute client returns the error information. You can view the error information without the need to generate logs.
- If you use common logging, logs are exported to stderr of Logview. You can view the logs in stderr.

Does a result table contain the duplicate data of two MapReduce jobs?

Yes, the result table contains the duplicate data of two MapReduce jobs. When you query data from the result table, two duplicate records are obtained.

In Hadoop MapReduce, I can select multiple nodes for distributed data processing. One node represents one machine. How do I configure nodes when I perform distributed data processing in MaxCompute MapReduce?

Compared with Hadoop MapReduce, you do not need to configure nodes in MaxCompute MapReduce.

When you run MapReduce jobs in MaxCompute, the underlying MaxCompute component determines which shards are used based on the algorithm.

If I do not use a combiner, the data output is normal. After I use a combiner, no input data is provided for reducers. Why?

This issue occurs because each record generated by a reducer cannot be mapped to the key-value pairs generated by a mapper.

I cannot specify the schema of an output table when I run map-only jobs in MaxCompute. Why?

You must specify the schema of an output table in the `CREATE TABLE` statement when you create the table. Therefore, when you run map-only jobs in MaxCompute, data is directly written to the output table without requiring you to specify the table schema.

How do I call the local MaxCompute server to run a MapReduce job?

In most cases, you must run the `jar` command on the MaxCompute client to run a MapReduce job. For more information about the syntax of the jar command, see Submit a MapReduce job.

You can perform the following steps to run a MapReduce job of a simulated MapReduce program by calling the local MaxCompute server.

1. Configure package dependencies.

   Dependency packages are required in addition to the SDK package. You can find the dependency packages in the lib folder on the MaxCompute client. The lib folder also contains the SDK JAR package. When you import JAR packages, we recommend that you import all JAR packages in the lib folder of the MaxCompute client that is of the latest version.

2. Upload the JAR package of the MapReduce program.

   Package the MapReduce program that passes local testing into a JAR file and upload the package. In this example, the JAR package is named *mr.jar*. For more information about how to upload resources, see Resource operations.

3. Specify the running mode.

   Configure JobConf. The following sample code shows how to configure JobConf.

   ```
   **Configure the connection information of MaxCompute.
   Account account = new AliyunAccount(accessid, accesskey);
   Odps odps = new Odps(account);
   odps.setEndpoint(endpoint);
   odps.setDefaultProject(project);
   **Obtain the session.
   SessionState ss = SessionState.get();
   ss.setOdps(odps);
   ss.setLocalRun(false);  **Set LocalRun to false. This value indicates that the MapReduc
   e job runs on the server. If local debugging is required, set LocalRun to true.
   **Code that includes JobConf configurations
   Job job = new Job();
   String resource = "mr.jar";
   job.setResources(resource); **This step is similar to jar -resources mr.jar.
   **Common MapReduce code
   job.setMapperClass(XXXMapper.class);
   job.setReducerClass(XXXReducer.class);
   ```

   After you configure JobConf, you can run the MapReduce job.

What do I do if the error message "BufferOverflowException" appears when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

```
FAILED: ODPS-0123131:User defined function exception - Traceback:
    java.nio.BufferOverflowException
    at java.nio.DirectByteBuffer.put(Unknown Source)
    at com.aliyun.odps.udf.impl.batch.TextBinary.put(TextBinary.java:35)
```

- Cause

  A large amount of data is written at a time. As a result, a buffer overflow occurs.

- Solution

  Make sure that the data that is written in MaxCompute meets the following limits on the data type of a single field:

```
String      8MB
Bigint      -9223372036854775807 ~ 9223372036854775807
Boolean     True/False
Double      -1.0 10308 ~ 1.0 10308
Date        0001-01-01 00:00:00 ~ 9999-12-31 23:59:59
```

What do I do if the error message "Resource not found" appears when I run a MapReduce job in MaxCompute?

When you submit a MapReduce job, you must configure the `-resources` parameter to specify the required resources. Separate multiple resources with commas (,).

What do I do if the error message "Class Not Found" appears when I run a MapReduce job in MaxCompute?

When MaxCompute MapReduce is run, the error message "Class Not Found" appears due to one of the following causes:

- The value of the `-classpath` parameter is invalid. A complete package name must be specified in the -classpath parameter.
- Incomplete source code in the src folder is packaged into the JAR file.

What do I do if the error ODPS-0010000 is returned when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

```
ODPS-0010000: System internal error - get input pangu dir meta fail.
```

- Cause

  A partition is not created or no data is inserted into the partition.

- Solution

Create a partition before you run a MapReduce job.

What do I do if the error message "Table not found" appears when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  Exception in thread "main" com.aliyun.odps.OdpsException: Table not found: project_name.table_name.
  ```

- Cause

  The name of the project in which the output table is stored is invalid or the output table does not exist.

- Solution

  TableInfo.Builder of MaxCompute MapReduce provides two parameters: ProjectName and TableName. You can configure the two parameters to specify the name of the project in which the output table is stored and the name of the output table.

What do I do if the error ODPS-0123144 is returned when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  FAILED: ODPS-0123144: Fuxi job failed - WorkerRestar
  ```

- Cause

  Data computing on the secondary node of the cluster times out. As a result, the primary node considers that the secondary node is faulty. The timeout period is fixed to 10 minutes and cannot be changed.

- Solution

  In most cases, this issue is caused by a large loop in the Reduce stage. For example, if long-tail data or Cartesian products are generated, a large loop exists. To address this issue, you must prevent large loops.

What do I do if the error message "java.security.AccessControlException" appears when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  FAILED: ODPS-0123131:User defined function exception - Traceback:
  java.lang.ExceptionInInitializerError
   ...
  Caused by: java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "getProtectionDomain")
    at java.security.AccessControlContext.checkPermission(AccessControlContext.java:472)
  ```

- Cause

  Your code violates sandbox limits. For more information about sandbox limits, see Java sandbox.

- Solution

  You must access external resources to address this issue. However, MaxCompute does not allow you to access external resources. To access external resources, you must store the processing logic and data of the external resources in MaxCompute. In this case, you must read specific configuration files. For more information, see Resource usage example.

What do I do if the error message "java.io.IOException" appears when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  Exception in thread "main" java.io.IOException: ODPS-0740001: Too many local-run maps: 10
  1, must be <= 100(specified by local-run parameter 'odps.mapred.local.map.max.tasks')
  ```

- Cause

  The default value of local-run maps is 100 and needs to be adjusted.

- Solution

  You can add the `-Dodps.mapred.local.map.max.tasks=200` configuration.

What do I do if the error message "Exceed maximum read times per resource" appears when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  ODPS-0730001: Exceed maximum read times per resource
  ```

- Cause

  The number of times resource files are read is excessively large.

- Solution

  Check the code logic that shows how resources are read. In most cases, resources are read only once when the setup method is called. Modify the code to ensure that resources are not read several times in the Map or Reduce stage.

What do I do if an OOM error occurs before the Reduce stage when I run a MapReduce job in MaxCompute?

- Cause

  A large amount of data is downloaded to memory.

- Solution

  Do not use a combiner or set `odps.mapred.map.min.split.size` to 512 for the combiner that you use.

What do I do if an OOM error occurs when I run a MapReduce job in MaxCompute?

This issue occurs due to insufficient memory. To address this issue, you can adjust the values of the following Java Virtual Machine (JVM) parameters: `odps.stage.mapper.jvm.mem and odps.stage.reducer.jvm.mem`. For example, you can set `odps.stage.mapper.jvm.mem` to 2048, which indicates 2 GB.

What do I do if the error message "java.lang.OutOfMemoryError" appears when I use 600 reducers to load a small configuration file for a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  java.lang.OutOfMemoryError: Java heap space
  ```

- Cause

  This issue occurs due to the limits of MaxCompute MapReduce. For more information, see Limits.

- Solution

  Configure memory-related parameters. For more information, see Overview.

What do I do if the error ODPS-0420095 is returned when I run a MapReduce job in MaxCompute?

- Problem description

  When a MapReduce job is run in MaxCompute, the following error message appears:

  ```
  Exception in thread "main" java.io.IOException: com.aliyun.odps.OdpsException: ODPS-04200
  95: Access Denied - The task is not in release range: LOT
  ```

- Cause

  Only PyODPS jobs and MaxCompute SQL jobs in which user-defined functions (UDFs) can be called are supported for projects of the MaxCompute developer edition. Other jobs, such as MapReduce jobs and Spark jobs, are not supported.

- Solution

  Upgrade project specifications. For more information, see Switch billing methods.

What do I do if a large number of errors occur when MaxCompute resources are referenced by a MapReduce job?

- Problem description

  When MaxCompute resources are referenced by a MapReduce job, the following error message appears:

  ```
   Caused by: com.aliyun.odps.OdpsException: java.io.FileNotFoundException: temp/mr_XXXXXX/
  resource/meta.user.group.config (Too many open files)
  ```

- Cause

The number of resources that are referenced by a single job cannot exceed 256. Otherwise, an error is returned. Each table or archive file is considered one resource. For more information about the limits on the number of resources that are referenced by a single job, see Limits.

- Solution

  Adjust the number of resources that are referenced by a single job.

I create a JAR package of the MapReduce program that contains third-party classes by using Maven Assembly. When I run a MapReduce job, an error message, indicating that the third-party classes are not found, appears. What do I do?

MaxCompute MapReduce that runs in a distributed environment is subject to the limits of Java sandboxes. The main program of MapReduce jobs is not subject to the limits. For more information about the limits of Java sandboxes, see Java sandbox.

If you want to process only JSON data, we recommend that you use Gson. This way, you do not need to include Gson classes in the JAR package. Open source Java components provide multiple classes that are used to convert strings into date values, such as SimpleDateFormat.

When I run a Hadoop MapReduce job in MaxCompute, an error of subscript out of bounds occurs. What do I do?

We recommend that you use MaxCompute MapReduce to write the job code. We also recommend that you use Spark on MaxCompute instead of MaxCompute MapReduce to write the job code.

4.3. Mars

4.3.1. Version updates

This topic describes the latest version updates of Mars, including the new and enhanced features in a specific version of Mars.

The following table describes the latest version updates of Mars. For more information, see the URL of the related version.

Version	Change type	Description
	New feature	DataFrame: • More features are added for `md.Index`. For more information, see Add more functionalities for `md.Index`(#1864). • `{DataFrame,Series}.rename_axis` is supported. For more information, see Implements `{DataFrame,Series}.rename_axis`(#1870).

Version	Change type	Description
v0.6.3	Enhanced feature	• JSON can be used for internal serialization. For more information, see Allow internal serialization to use JSON(#1882). • The performance of `{md.read_csv()` and `md.read_parquet()}.head()` is optimized. For more information, see Optimize performance of {md.read_csv(), md.read_parquet()}.head()(#1883). • The performance of `df.sort_values().head()` is optimized. For more information, see Optimize performance of `df.sort_values().head()`(#1888). • Column pruning is supported for the data sources that use `groupby().agg()`. For more information, see Support column pruning for groupby().agg() on data sources(#1889). • `named_{dataframe, series, tensor}` is optimized to obtain more metadata. For more information, see Improve `named_{dataframe, series, tensor}` that it's able to get more meta(#1897).
v0.6.2	New feature	• DataFrame `df.groupby().head()` is supported. For more information, see Implements head() on groupby objects(#1851). • Learn `mars.learn.preprocessing.{MinMaxScaler, minmax_scale}` is supported. For more information, see Implements `mars.learn.preprocessing.{MinMaxScaler, minmax_scale}'(#1858).

Version	Change type	Description
	Enhanced feature	• The `recall_by_id` computation method of Proxima is optimized. For more information, see Improve proxima `recall_by_id` computation method(#1805). • The bug that exists in reading from and writing to vineyard of tensors and DataFrame is fixed. For more information, see Revise to/from vineyard, of Tensor and DataFrame(#1806). • A memory prediction mechanism similar to `read_csv` is added for `read_parquet`. For more information, see Add memory estimation for `read_parquet` as well as `read_csv`(#1815). • Compound aggregate functions can be used in Lambda. For more information, see Support using compound agg function in lambda(#1819). • The incremental_index field is added to `reset_index`. The default value of this field is False. For more information, see Add `incremental_index` argument to `reset_index` which by default is False(#1842). • The `to_pandas` method of DataFrame and Series obtains data in batches. For more information, see Support `to_pandas` in a batch way for DataFrame and Series(#1859). • You can specify a memory scale in Kubernetes. For more information, see Support specifying memory scale in kubernetes(#1861).
v0.6.1	New feature	• Tensor When a tensor is converted into a sparse tensor, values other than 0 can be specified for missing parameters. For more information, see Support 'missing' argument for 'tensor.tosparse()' and 'fill_value' argument for 'sparse_tensor.todense()'(#1802). • DataFrame ◦ The replace API is added. For more information, see Implements {DataFrame,Series}.replace(#1765). ◦ The cartesian_chunk API is added to calculate the Cartesian product of two DataFrame objects and apply a function to each chunk. This API is an extension of the pandas API. For more information, see Add '{DataFrame, Series}.cartesian_chunk' support(#1777). ◦ String concatenation is optimized for aggregation and group aggregation. For more information, see Integrate 'str.cat' into reduction and groupby-aggregation(#1781). ◦ The level parameter is supported in aggregate functions. For more information, see Implements reduction with 'level' argument(#1784).

Version	Change type	Description
v0.6.0	Compatibility	Aggregation and group aggregation are rewritten. If the Mars client of an earlier version connects to the Mars server of this version, these features may fail.
	New feature	DataFrame: • The `num_partitions` parameter is supported during DataFrame initialization. For more information, see Support 'num_partitions' argument for DataFrame initializers(#1733). • Named aggregation is supported. For more information, see Add support for named aggregations(#1748).
	Enhanced feature	The `groupby()` and agg() functions are unified in ReductionCompiler. For more information, see Unify groupby.agg() using ReductionCompiler(#1739).
	Fixed issue	• md.read_csv is fixed to prevent the error that may occur if both names and usecols are specified. For more information, see Fix 'md.read_csv' when names and usecols specified(#1738). • The string dtype is supported for tensor reduction and PSRS balancing. For more information, see Support string dtype for tensor reductions(#1746). • XGBoost and LightGBM on DataFrame are fixed. For more information, see Fix xgboost and lightgbm on DataFrames(#1751). • The repeated running of the same code in distributed mode is fixed. For more information, see Fix repeated running of same code in distributed mode(#1753). • A column in DataFrame can be set to a tensor of the scalar type. For more information, see Support setting scalar which is a tensor for DataFrame(#1758).

4.3.2. Overview

This topic describes the features of Mars, the differences between Mars and PyODPS, and the scenarios of using Mars and the PyODPS DataFrame API.

Scenarios

Use Mars and the PyODPS DataFrame API in the following scenarios:

- Mars
  - You often call the to_pandas() method of the PyODPS DataFrame API to convert a PyODPS DataFrame to a pandas DataFrame.
  - You are familiar with the pandas API, but do not want to learn the PyODPS DataFrame API.
  - You want to use indexes.

- You want to maintain the data order after you create a DataFrame.

  The Mars DataFrame API provides the iloc method to retrieve rows and obtain data in specific rows. For example, `df.iloc[10]` is used to obtain data in the tenth row. The Mars DataFrame API also provides the `df.shift()` and `df.ffill()` methods, both of which can be used only in scenarios where the data order is maintained.

- You want to run NumPy or scikit-learn in a parallel and distributed manner, or run TensorFlow, PyTorch, and XGBoost in a distributed manner.
- You want to process data whose volume is less than 1 TB.

- PyODPS DataFrame
  - You want to use MaxCompute to schedule jobs. The PyODPS DataFrame API compiles operations on DataFrames to MaxCompute SQL statements. If you want to schedule jobs by using MaxCompute, we recommend that you use the PyODPS DataFrame API.
  - You want to schedule jobs in a more stable environment. The PyODPS DataFrame API compiles operations to MaxCompute SQL statements for execution. MaxCompute is stable, which means PyODPS is stable. Mars is new and less stable. Therefore, we recommend that you use the PyODPS DataFrame API if you require high stability.
  - If you want to process data whose volume is larger than 1 TB, we recommend that you use the PyODPS DataFrame API.

Differences between Mars and PyODPS

- API
  - Mars

    The Mars DataFrame API is fully compatible with pandas. The Mars tensor API is compatible with NumPy. The Mars learn API is compatible with scikit-learn.

  - PyODPS

    PyODPS only provides the DataFrame API, which is different from the pandas API.

- Index

- Mars

  The Mars DataFrame API supports operations based on indexes, including row indexes and column indexes. The following code shows an example:

```
In [1]: import mars.dataframe as md
In [5]: import mars.tensor as mt
In [7]: df = md.DataFrame(mt.random.rand(10, 3), index=md.date_range('2020-5-1', period
s=10))
In [9]: df.loc['2020-5'].execute()
Out[9]:
                   0         1         2
2020-05-01  0.061912  0.507101  0.372242
2020-05-02  0.833663  0.818519  0.943887
2020-05-03  0.579214  0.573056  0.319786
2020-05-04  0.476143  0.245831  0.434038
2020-05-05  0.444866  0.465851  0.445263
2020-05-06  0.654311  0.972639  0.443985
2020-05-07  0.276574  0.096421  0.264799
2020-05-08  0.106188  0.921479  0.202131
2020-05-09  0.281736  0.465473  0.003585
2020-05-10  0.400000  0.451150  0.956905
```

- PyODPS

  PyODPS does not support index-based operations.

- Data order
  - Mars

    After a Mars DataFrame is created, it maintains the data order. The Mars DataFrame API provides time series methods such as `shift`, and missing value handling methods such as `ffill` and `bfill`.

```
In [3]: df = md.DataFrame([[1, None], [None, 1]])
In [4]: df.execute()
Out[4]:
     0    1
0  1.0  NaN
1  NaN  1.0
In [5]: df.ffill().execute() # Fill the missing value with the value in the previous ro
w.
Out[5]:
     0    1
0  1.0  NaN
1  1.0  1.0
```

  - PyODPS

    PyODPS processes and stores data by using MaxCompute, which does not maintain the data order. Therefore, PyODPS does not maintain the data order or support time series methods.

- Execution

- Mars

  Mars consists of a client and a distributed execution layer. You can call the `o.create_mars_cluster` method to create a Mars cluster in MaxCompute and submit computing jobs to the Mars cluster. This process greatly saves the costs for scheduling. Mars outperforms PyODPS in processing smaller amounts of data.

- PyODPS

  PyODPS is a client and does not contain any servers. When you use the PyODPS DataFrame API, the system compiles the operations to MaxCompute SQL statements for execution. Therefore, the operations supported by the PyODPS DataFrame API depend on MaxCompute SQL. Each time you call the `execute` method, a MaxCompute job is submitted for the cluster to schedule.

Features

Mars is a unified distributed computing framework based on tensors. Mars can use parallel and distributed computing technologies to accelerate data processing for Python data science libraries such as NumPy, pandas and scikit-learn.

Mars provides the following common APIs:

- Mars tensor

  The Mars tensor API mimics the NumPy API and supports processing large multidimensional arrays, which are also called tensors. The following code shows an example of how to use the Mars tensor API:

```
import mars.tensor as mt
a = mt.random.rand(10000, 50)
b = mt.random.rand(50, 5000)
a.dot(b).execute()
```

- Mars DataFrame

  The Mars DataFrame API mimics the pandas API and supports processing and analyzing a large amount of data. The following code shows an example of how to use the Mars DataFrame API:

```
import mars.dataframe as md
ratings = md.read_csv('Downloads/ml-20m/ratings.csv')
movies = md.read_csv('Downloads/ml-20m/movies.csv')
movie_rating = ratings.groupby('movieId', as_index=False).agg({'rating': 'mean'})
result = movie_rating.merge(movies[['movieId', 'title']], on='movieId')
result.sort_values(by='rating', ascending=False).execute()
```

- Mars learn

  The Mars learn API mimics the scikit-learn API. The Mars learn API can be integrated with TensorFlow, PyTorch, and XGBoost. The following code shows an example of how to use the Mars learn API:

```
import mars.dataframe as md
from mars.learn.neighbors import NearestNeighbors
df = md.read_csv('data.csv')
nn = NearestNeighbors(n_neighbors=10)
nn.fit(df)
neighbors = nn.kneighbors(df).fetch()
```

References

- Mars on GitHub
- Mars documentation
- Mars column

Technical support

If you encounter any problems when you use Mars, join the DingTalk group whose ID is 11701793 for technical support.

4.3.3. Preparations

This topic describes how to prepare the Mars runtime environment.

To run Mars in MaxCompute, you must prepare the Mars runtime environment by using one of the following methods:

- DataWorks
  i. Create a DataWorks PyODPS 3 node, which provides features of PyODPS and Mars.

     You can run the following commands in the new PyODPS 3 node to check the versions of PyODPS and Mars. Make sure that the versions meet the requirements.

     ```
     from odps import __version__ as odps_version
     from mars import __version__ as mars_version
     print(odps_version)
     print(mars_version)
     ```

     odps_version: the version of PyODPS. Make sure that the PyODPS version is V0.9.3.1 or later.
     mars_version: the version of Mars. Make sure that the Mars version is V0.4.4 or later.

  ii. Initialize a MaxCompute entry.

     You can use the MaxCompute entry provided by the DataWorks PyODPS 3 node.

- Other environments
  i. Install pip. After pip is installed, install PyODPS and Mars by running the pip install command in a command-line interface such as the Command Prompt in Windows. The following commands show an example of how to use the pip install command:
     - (Optional)Make sure that pip is in the latest version.

       ```
       pip install -U pip
       ```

     - Install the latest version of PyODPS. In the command, https://mirrors.aliyun.com/pypi/simple/ is the URL of the Python Package Index (PyPI) mirror that Alibaba Cloud provides to accelerate package download.

       ```
       pip install pyodps -i https://mirrors.aliyun.com/pypi/simple/
       ```

     - Install the latest version of Mars.

       ```
       pip install pymars -i https://mirrors.aliyun.com/pypi/simple/
       ```

- Install the latest version of protocol buffers.

```
pip install protobuf -i https://mirrors.aliyun.com/pypi/simple/
```

- (Optional)Install the latest version of PyArrow to accelerate job execution in Mars.

```
pip install pyarrow -i https://mirrors.aliyun.com/pypi/simple/
```

> **Note** For more information about how to install pip, see Installation in the pip documentation.

ii. Initialize a MaxCompute entry.

You must use your AccessKey ID and AccessKey secret to initialize the MaxCompute entry. For more information about how to initialize a MaxCompute entry, see PyODPS: ODPS Python SDK and data analysis framework.

4.3.4. Usage notes

This topic describes how to perform operations on Mars clusters, read and write MaxCompute tables, and obtain the URLs of Mars UI, Logview, and Jupyter Notebook.

For more information about how to develop Mars jobs, see Mars.

Mars cluster operations

- Create a Mars cluster

  Run the following commands to create a Mars cluster. This process takes a while to complete.

```
from odps import options
options.verbose = True
# If the preceding commands have been configured on the DataWorks PyODPS 3 node, you do n
ot need to run the two commands.
client = o.create_mars_cluster(5, 4, 16, min_worker_num=3)
```

  where:

  - 5: the number of worker nodes in the cluster. In this example, the cluster consists of five worker nodes.
  - 4: the number of CPU cores for each worker node. In this example, each worker node has four CPU cores.
  - 16: the memory size of each worker node. In this example, each worker node has 16 GB of memory.

> **Note**
> - The memory size that you request for each worker node must be greater than 1 GB. The optimal ratio of CPU cores to the memory size is 1:4. For example, configure a worker node with 4 CPU cores and 16 GB of memory.
> - You can create a maximum of 30 worker nodes. If the number of worker nodes exceeds the upper limit, the image server may be overloaded. If you want to create more than 30 worker nodes, submit a ticket.

- min_worker_num: the minimum number of worker nodes that must be started for the system to return a client object. If this parameter is set to 3, the system returns a client object after the three worker nodes are started.

If you set `options.verbose` to True when you create a Mars cluster, the URLs of Logview, Mars UI, and Jupyter Notebook of the MaxCompute instance are displayed in the command output. You can use the Mars UI to connect to Mars clusters and query the status of clusters and jobs.

- Submit a job

  When you create a Mars cluster, the cluster creates a default session that connects to the cluster. You can call the `.execute()` method to submit a job to the cluster and run the job in the default session.

```
import mars.dataframe as md
import mars.tensor as mt
md.DataFrame(mt.random.rand(10, 3)).execute()  # Call the .execute() method to submit the job to the created cluster.
```

- Stop and release a cluster

  A Mars cluster is automatically released three days after it is created. If you no longer require a Mars cluster, you can call the `client.stop_server()` method to release the cluster.

```
client.stop_server()
```

Read and write operations on MaxCompute tables

Mars can directly read and write MaxCompute tables.

- Read MaxCompute tables

  Mars calls the `o.to_mars_dataframe` method to read a MaxCompute table and returns a Mars DataFrame.

```
In [1]: df = o.to_mars_dataframe('test_mars')
In [2]: df.head(6).execute()
Out[2]:
       col1  col2
0         0     0
1         0     1
2         0     2
3         1     0
4         1     1
5         1     2
```

- Write MaxCompute tables

  Mars calls the `o.persist_mars_dataframe(df, 'table_name')` method to save a Mars DataFrame as a MaxCompute table.

```
In [3]: df = o.to_mars_dataframe('test_mars')
In [4]: df2 = df + 1
In [5]: o.persist_mars_dataframe(df2, 'test_mars_persist')  # Save the Mars DataFrame as
a MaxCompute table.
In [6]: o.get_table('test_mars_persist').to_df().head(6)  # Call the PyODPS DataFrame API
operation to query data.
       col1  col2
0         1     1
1         1     2
2         1     3
3         2     1
4         2     2
5         2     3
```

- Use the Jupyter Notebook of a Mars cluster

  > **Note** The Jupyter Notebook can be used only if `with_notebook=True` is specified in `create_mars_cluster`.

  When you create a Jupyter Notebook document, a session is automatically created to submit jobs to the Mars cluster. Therefore, session creation does not need to be shown in the Jupyter Notebook document.

```
import mars.dataframe as md
md.DataFrame(mt.random.rand(10, 3)).sum().execute() # Call the .execute() method in the J
upyter Notebook to submit the job to the current cluster. Therefore, session creation doe
s not need to be shown in the Jupyter Notebook document.
```

  > **Note**
  > - The Jupyter Notebook document is not automatically saved. We recommend that you manually save the Jupyter Notebook document as required.
  > - You can connect your Jupyter Notebook to an existing Mars cluster. For more information, see Use an existing Mars cluster.

Other operations

- Use an existing Mars cluster
  - Recreate an existing Mars cluster based on the instance ID.

```
client = o.create_mars_cluster(instance_id=**instance-id**)
```

  - To use an existing Mars cluster, create a Mars session to visit the URL of the Mars UI.

```
from mars.session import new_session
new_session('**URL of the Mars UI**').as_default() # Set the created session as the def
ault session.
```

- Obtain the URL of the Mars UI

If you set `options.verbose` to True when you create a Mars cluster, the URL of the Mars UI is automatically displayed in the command output. You can use `client.endpoint` to obtain the URL of the Mars UI.

```
print(client.endpoint)
```

- Obtain the Logview URL of an instance

  If you set `options.verbose` to True when you create a Mars cluster, the Logview URL is automatically displayed in the command output. You can also use `client.get_logview_address()` to obtain the Logview URL.

  ```
  print(client.get_logview_address())
  ```

- Obtain the Jupyter Notebook URL

  If you set `options.verbose` to True when you create a Mars cluster, the Jupyter Notebook URL is automatically displayed in the command output. You can also use `client.get_notebook_endpoint()` to obtain the Jupyter Notebook URL.

  ```
  print(client.get_notebook_endpoint())
  ```

4.4. Graph

4.4.1. Overview

MaxCompute Graph is a processing framework designed for iterative graph computing. Graph computing jobs use graphs to build models. A graph is a collection of vertices and edges that have values.

MaxCompute Graph allows you to perform the following operations on graphs:

- Change the value of a vertex or edge.
- Add or remove a vertex.
- Add or remove an edge.

> **Note** When you edit a vertex or edge, you must use code to maintain the relationships between vertices and edges.

MaxCompute Graph performs iterations to edit graphs, enable graphs to evolve, and obtain analysis results. Typical applications include the PageRank, single source shortest path (SSSP) algorithm, and k-means algorithm. You can use SDK for Java provided by MaxCompute Graph to compile graph computing programs.

Terms

- Graph: an abstract data structure used to represent the relationships between objects. Vertices and edges are used in graphs. Vertices represent objects, and edges represent the relationships between the objects. The data described in graphs is called graph data.
- Vertex: represents an object in a graph.
- Edge: a single directed edge that represents the relationship between objects in a graph. An edge consists of a source vertex ID, a destination vertex ID, and the data associated with the edge.

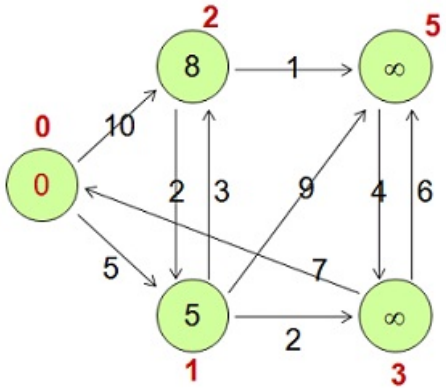
- Directed graph: a graph in which edges have directions. Two vertices on an edge represent different objects. For example, Page A is connected to Page B. In directed graphs, edges are classified into outgoing edges and incoming edges.
- Undirected graph: a graph in which edges do not have directions, such as common users in a user group.
- Outgoing edge: a directed edge on which the current vertex is the origin.
- Incoming edge: a directed edge on which the current vertex is the destination.
- Degree: the number of edges that connect to a vertex.
- Outdegree: the number of outgoing edges that connect to a vertex.
- Indegree: the number of incoming edges that connect to a vertex.
- Superstep: an iteration of a graph.

Graph data structure

MaxCompute Graph processes directed graphs. MaxCompute provides only a two-dimensional table storage structure. Therefore, you must resolve graph data into two-dimensional tables and store them in MaxCompute. You can resolve the graph data based on your business requirements.

During graph computing and analytics, use custom GraphLoader to convert two-dimensional table data into vertices and edges that are applicable to MaxCompute Graph.

- The structure of a vertex is <ID, Value, Halted, Edges>. Parameters:
  - ID: the ID of the vertex.
  - Value: the value of the vertex.
  - Halted: specifies whether the iteration of the vertex is terminated.
  - Edges: the edges that start from the vertex.
- The structure of an edge is <DestVertexID, Value>. Parameters:
  - DestVertexID: the ID of the destination vertex.
  - Value: the value of the edge.



For example, the preceding figure can be expressed in the following two-dimensional table.

Vertex	<ID, Value, Halted, Edges>
v0	<0, 0, false, [<1, 5 >, <2, 10 >]>
v1	<1, 5, false, [<2, 3>, <3, 2>, <5, 9>]>

Vertex	<ID, Value, Halted, Edges>
v2	<2, 8, false, [<1, 2>, <5, 1 >]>
v3	<3, Long.MAX_VALUE, false, [<0, 7>, <5, 6>]>
v5	<5, Long.MAX_VALUE, false, [<3, 4 >]>

Logic of a Graph program

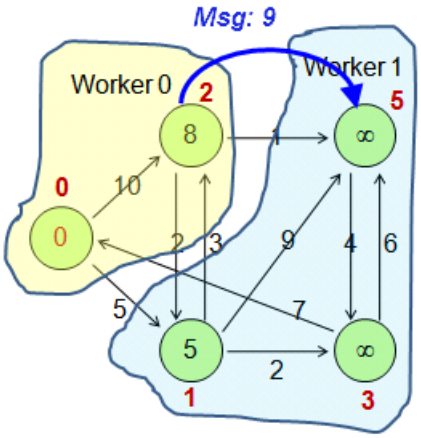
A Graph program performs the following operations: graph loading, iterative computing, and iteration termination.

1. Graph loading

   Graph loading consists of the following steps:

   i. Graph loading: MaxCompute Graph calls the custom GraphLoader to resolve the records in the input table into vertices or edges.

   ii. Partitioning: MaxCompute Graph calls the custom Partitioner to partition the vertices and distributes the partitions to the related workers. By default, MaxCompute Graph partitions the vertices based on the hash values of the vertex IDs modulo the number of workers.



   In the preceding figure, the number of workers is 2. Vertex 0 and Vertex 2 are distributed to Worker 0 because the result of vertex IDs modulo 2 is 0. Vertex 1, Vertex 3, and Vertex 5 are distributed to Worker 1 because the result of vertex IDs modulo 2 is 1.

2. Iterative computing

   An iteration is called a superstep. During an iteration, the program traverses all non-halted vertices or all vertices that receive messages, and calls the `compute(ComputeContext context, Iterable messages)` method. For non-halted vertices, the value of the Halted parameter is false. Halted vertices are automatically activated after they receive messages.

   The `compute(ComputeContext context, Iterable messages)` method can be used to perform the following operations:

   - Process the messages that are sent by the previous superstep to the current vertex.
   - Edit a graph as required:
     - Change the values of vertices or edges.
     - Send messages to some vertices.

    - Add or remove vertices or edges.
  - Use an aggregator to collect information and update global information. For more information, see Aggregator overview.
  - Set the current vertex to the halted or non-halted state.
  - Enable MaxCompute Graph to asynchronously send messages to the related workers in each superstep. The messages are then processed in the next superstep.
3. Iteration termination

   An iteration is terminated if one or more of the following conditions are met:

   - All vertices are in the halted state (the value of the Halted parameter is true), and no new messages are generated.
   - The maximum number of iterations is reached.
   - The `terminate` method of an aggregator returns true.

   Pseudocode of the Graph program:

```
// 1. load
for each record in input_table {
  GraphLoader.load();
}
// 2. setup
WorkerComputer.setup();
for each aggr in aggregators {
  aggr.createStartupValue();
}
for each v in vertices {
  v.setup();
}
// 3. superstep
for (step = 0; step < max; step ++) {
  for each aggr in aggregators {
    aggr.createInitialValue();
  }
  for each v in vertices {
     v.compute();
   }
}
// 4. cleanup
for each v in vertices {
  v.cleanup();
}
WorkerComputer.cleanup();
```

4.4.2. Aggregator overview

This topic describes the implementation mechanism and related API operations of Aggregator. It also describes how to apply Aggregator by using k-means clustering.

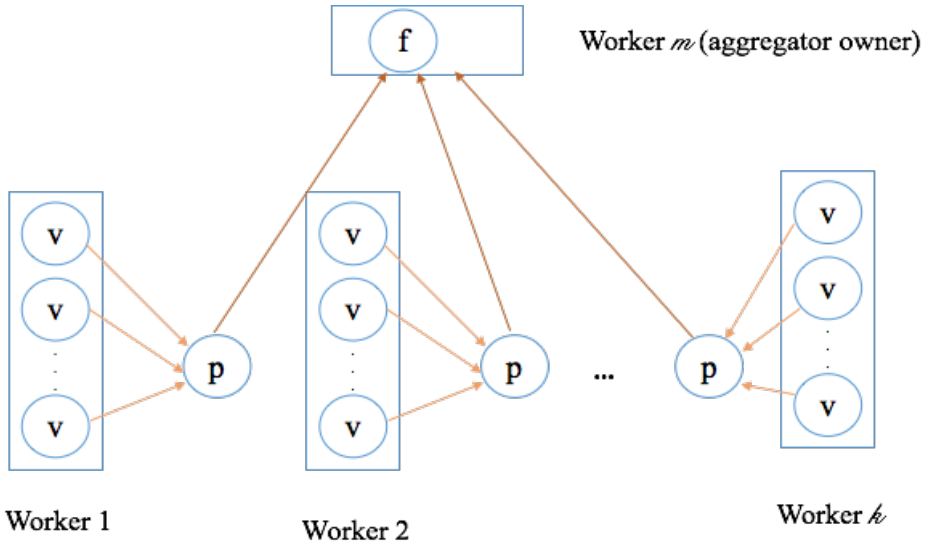
Aggregator is a common feature in MaxCompute Graph jobs. It is most suited to handle machine learning issues. In MaxCompute Graph, Aggregator is used to aggregate and process global information.

Implementation mechanism

The logic of Aggregator is divided into two parts:

- One part is implemented on all workers in distributed mode.
- The other part is implemented only on the worker where the Aggregator owner resides in single vertex mode.

Initial values are created and some of these values are aggregated on each worker. Then, the partial aggregation results of all workers are sent to the worker where the Aggregator owner resides. This worker then aggregates the received partial aggregation objects into a global aggregation result and determines whether to end the iteration. The global aggregation result is distributed to all workers in the next superstep for iteration.



1. When each worker starts, it executes createStartupValue to create AggregatorValue.
2. Before each iteration starts, each worker executes createInitialValue to initialize AggregatorValue for the iteration.
3. In an iteration, each vertex uses `context.aggregate()` to call `aggregate()` to implement partial iteration in the worker.
4. Each worker sends the partial iteration result to the worker where the Aggregator owner resides.
5. The worker where the Aggregator owner resides executes `merge` multiple times to implement global aggregation.
6. The worker where the Aggregator owner resides executes `terminate` to process the global aggregation result and determines whether to end the iteration.

API operations

Aggregator provides five API operations. This section describes when and how to use these API operations.

- createStartupValue(context)

  This API operation is performed once on all workers before each superstep starts. It is used to initialize `AggregatorValue`. In the first superstep (superstep 0), `WorkerContext.getLastAggregatedValue()` or `ComputeContext.getLastAggregatedValue()` is called to obtain the initialized `AggregatorValue` object.

- createInitialValue(context)

  This API operation is performed once on all workers when each superstep starts. It is used to initialize `AggregatorValue` for the current iteration. In most cases, `WorkerContext.getLastAggregatedValue()` is called to obtain the result of the previous iteration. Then, the partial initialization is implemented.

- aggregate(value, item)

  This API operation is also performed on all workers. It is triggered by an explicit call to `ComputeContext#aggregate(item)`, while the preceding two API operations are automatically called by the framework. This API operation is used to implement partial aggregation. The first parameter `value` indicates the aggregation result of the worker in the current superstep. The initial value is the object that is returned by `createInitialValue`. The second parameter is passed in when `ComputeContext#aggregate(item)` is called by using your code. In this API operation, `item` is used to update `value` for aggregation in most cases. After all the `aggregate` operations are performed, the obtained `value` is the partial aggregation result of the worker. The result is then sent by the framework to the worker where the Aggregator owner resides.

- merge(value, partial)

  This API operation is performed on the worker where the Aggregator owner resides. It is used to merge partial aggregation results of workers to obtain the global aggregation object. Similar to `aggregate`, `value` in this API operation indicates the aggregated results, and `partial` indicates objects that you want to aggregate. `partial` is used to update `value`.

  For example, three workers, w0, w1, and w2 generate partial aggregation results p0, p1, and p2. If p1, p0, and p2 are sent in sequence to the worker where the Aggregator owner resides, the `merge` operations are performed in the following sequence:

  i. `merge(p1, p0)` is first executed to aggregate p1 and p0 as p1.
  ii. `merge(p1, p2)` is executed to aggregate p1 and p2 as p1. p1 is the global aggregation result in this superstep.

  Therefore, if only one worker exists, the `merge` method is not required. In this case, `merge()` is not called.

- terminate(context, value)

  After the worker where the Aggregator owner resides executes `merge()`, the framework calls `terminate(context, value)` to perform the final processing. The second parameter `value` indicates the global aggregation result that is obtained by calling `merge()`. The global aggregation result can be further modified in this API operation. After `terminate()` is executed, the framework distributes the global aggregation object to all workers for the next superstep. If true is returned for `terminate()`, iteration is ended for the entire job. Otherwise, the iteration continues. If true is returned after convergence is completed, jobs are immediately ended. This applies to machine learning scenarios.

K-means clustering example

This section uses k-means clustering as an example to demonstrate how to use Aggregator.

> **Note** If you need the complete code, see Kmeans. In this section, the code is resolved and is only for reference.

- GraphLoader

  GraphLoader is used to load an input table and convert it to vertices or edges of a graph. In this example, each row of data in the input table is a sample, each sample constructs a vertex, and vertex values are used to store samples.

  A writable class `KmeansValue` is defined as the value type of a vertex.

```
public static class KmeansValue implements Writable {
    DenseVector sample;
    public KmeansValue() {
    }
    public KmeansValue(DenseVector v) {
        this.sample = v;
    }
    @Override
        public void write(DataOutput out) throws IOException {
        wirteForDenseVector(out, sample);
    }
    @Override
        public void readFields(DataInput in) throws IOException {
        sample = readFieldsForDenseVector(in);
    }
}
```

  A `DenseVector` object is encapsulated in `KmeansValue` to store a sample. The `DenseVector` type originates from matrix-toolkits-java. `wirteForDenseVector()` and `readFieldsForDenseVector()` are used for serialization and deserialization.

  Custom `KmeansReader` code:

```
public static class KmeansReader extends
    GraphLoader<LongWritable, KmeansValue, NullWritable, NullWritable> {
    @Override
        public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, KmeansValue, NullWritable, NullWritable> context)
        throws IOException {
        KmeansVertex v = new KmeansVertex();
        v.setId(recordNum);
        int n = record.size();
        DenseVector dv = new DenseVector(n);
        for (int i = 0; i < n; i++) {
            dv.set(i, ((DoubleWritable)record.get(i)).get());
        }
        v.setValue(new KmeansValue(dv));
        context.addVertexRequest(v);
    }
}
```

  In `KmeansReader`, a vertex is created when each row of data (a record) is read. `recordNum` is used as the vertex ID, and the `record` content is converted to a `DenseVector` object and encapsulated in `VertexValue`.

- Vertex

  Custom `KmeansVertex` code: The logic of the preceding code is to implement partial aggregation for samples maintained for each iteration. For more information about the code logic, see the implementation of Aggregator.

```
public static class KmeansVertex extends
    Vertex<LongWritable, KmeansValue, NullWritable, NullWritable> {
    @Override
        public void compute(
        ComputeContext<LongWritable, KmeansValue, NullWritable, NullWritable> context,
        Iterable<NullWritable> messages) throws IOException {
        context.aggregate(getValue());
    }
}
```

- Aggregator

  The main logic of Kmeans is concentrated on Aggregator. Custom `KmeansAggrValue` is used to maintain the content you want to aggregate and distribute.

```
public static class KmeansAggrValue implements Writable {
    DenseMatrix centroids;
    DenseMatrix sums; // used to recalculate new centroids
    DenseVector counts; // used to recalculate new centroids
    @Override
        public void write(DataOutput out) throws IOException {
        wirteForDenseDenseMatrix(out, centroids);
        wirteForDenseDenseMatrix(out, sums);
        wirteForDenseVector(out, counts);
    }
    @Override
        public void readFields(DataInput in) throws IOException {
        centroids = readFieldsForDenseMatrix(in);
        sums = readFieldsForDenseMatrix(in);
        counts = readFieldsForDenseVector(in);
    }
}
```

  In the preceding code, three objects are maintained in `KmeansAggrValue`:

  - `centroids`: indicates the existing K centers. If the sample is m-dimensional, `centroids` is a matrix of K × m.
  - `sums`: indicates a matrix of the same size as `centroids`. Each element records the sum of a specific dimension of the sample that is closest to a specific center. For example, `sums(i,j)` indicates the sum of dimension j of the sample that is closest to center i.
  - `counts` is a K-dimensional vector. It records the number of samples that are closest to each center. `counts` is used with `sums` to calculate a new center, which is the main content to be aggregated.

  `KmeansAggregator` is a custom Aggregator implementation class. The following section describes the implementation of the preceding API operations:

  i. Implementation of `createStartupValue()`

```
public static class KmeansAggregator extends Aggregator<KmeansAggrValue> {
    public KmeansAggrValue createStartupValue(WorkerContext context) throws IOExcepti
on {
        KmeansAggrValue av = new KmeansAggrValue();
        byte[] centers = context.readCacheFile("centers");
        String lines[] = new String(centers).split("\n");
        int rows = lines.length;
        int cols = lines[0].split(",").length; // assumption rows >= 1
        av.centroids = new DenseMatrix(rows, cols);
        av.sums = new DenseMatrix(rows, cols);
        av.sums.zero();
        av.counts = new DenseVector(rows);
        av.counts.zero();
        for (int i = 0; i < lines.length; i++) {
            String[] ss = lines[i].split(",");
            for (int j = 0; j < ss.length; j++) {
                av.centroids.set(i, j, Double.valueOf(ss[j]));
            }
        }
        return av;
    }
```

This method initializes a `KmeansAggrValue` object, reads the initial center from the `centers` file, and assigns a value to `centroids`. The initial values of `sums` and `counts` are 0.

ii. Implementation of `createInitialValue()`

```
@Override
public KmeansAggrValue createInitialValue(WorkerContext context)
    throws IOException {
    KmeansAggrValue av = (KmeansAggrValue)context.getLastAggregatedValue(0);
    // reset for next iteration
    av.sums.zero();
    av.counts.zero();
    return av;
}
```

This method first obtains `KmeansAggrValue` of the previous iteration and clears the values of `sums` and `counts`. Only the `centroids` value of the previous iteration is retained.

iii. Implementation of `aggregate()`

```
@Override
public void aggregate(KmeansAggrValue value, Object item)
    throws IOException {
    DenseVector sample = ((KmeansValue)item).sample;
    // find the nearest centroid
    int min = findNearestCentroid(value.centroids, sample);
    // update sum and count
    for (int i = 0; i < sample.size(); i ++) {
        value.sums.add(min, i, sample.get(i));
    }
    value.counts.add(min, 1.0d);
}
```

This method calls `findNearestCentroid()` to identify the index of the center that is closest to the sample `item`, uses `sums` to aggregate all dimensions, and increments the value of `counts` by 1.

The preceding three methods are executed on all workers to implement partial aggregation. The following methods can be used to implement global aggregation on the worker where the Aggregator owner resides:

i. Implementation of `merge()`

```
@Override
public void merge(KmeansAggrValue value, KmeansAggrValue partial)
    throws IOException {
    value.sums.add(partial.sums);
    value.counts.add(partial.counts);
}
```

In the preceding example, the implementation logic of `merge` is to add the values of `sums` and `counts` aggregated by each worker.

ii. Implementation of `terminate()`

```
@Override
public boolean terminate(WorkerContext context, KmeansAggrValue value)
    throws IOException {
    // Calculate the new means to be the centroids (original sums)
    DenseMatrix newCentriods = calculateNewCentroids(value.sums, value.counts, value.
centroids);
    // print old centroids and new centroids for debugging
    System.out.println("\nsuperstep: " + context.getSuperstep() +
                       "\nold centriod:\n" + value.centroids + " new centriod:\n" + n
ewCentriods);
    boolean converged = isConverged(newCentriods, value.centroids, 0.05d);
    System.out.println("superstep: " + context.getSuperstep() + "/"
                       + (context.getMaxIteration() - 1) + " converged: " + converged
);
    if (converged || context.getSuperstep() == context.getMaxIteration() - 1) {
        // converged or reach max iteration, output centriods
        for (int i = 0; i < newCentriods.numRows(); i++) {
            Writable[] centriod = new Writable[newCentriods.numColumns()];
            for (int j = 0; j < newCentriods.numColumns(); j++) {
                centriod[j] = new DoubleWritable(newCentriods.get(i, j));
            }
            context.write(centriod);
        }
        // true means to terminate iteration
        return true;
    }
    // update centriods
    value.centroids.set(newCentriods);
    // false means to continue iteration
    return false;
}
```

In the preceding example, `teminate()` calls `calculateNewCentroids()` based on `sums` and `counts` to calculate the average value and obtain a new center. Then, `isConverged()` is called to check whether the center is converged based on the Euclidean distance between the new and old centers. If the number of convergences or iterations reaches the upper limit, the new center is generated, and true is returned to end the iteration. Otherwise, the center is updated, and false is returned to continue the iteration.

- `main` method

  The `main` method is used to construct `GraphJob`, configure related settings, and submit a job.

```
public static void main(String[] args) throws IOException {
    if (args.length < 2)
        printUsage();
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(KmeansReader.class);
    job.setRuntimePartitioning(false);
    job.setVertexClass(KmeansVertex.class);
    job.setAggregatorClass(KmeansAggregator.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    // default max iteration is 30
    job.setMaxIteration(30);
    if (args.length >= 3)
        job.setMaxIteration(Integer.parseInt(args[2]));
    long start = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
                       + (System.currentTimeMillis() - start) / 1000.0 + " seconds");
}
```

> **Note** If `job.setRuntimePartitioning` is set to false, data loaded by each worker is not partitioned based on the partitioner. Data is loaded and maintained by the same worker.

4.4.3. Limits

This topic describes the limits of MaxCompute Graph.

- Each job can reference up to 256 resources. Each table or archive is considered as one unit.
- The total bytes of resources referenced by a job cannot exceed 512 MB.
- The number of the inputs of a job cannot exceed 1,024, and that of the outputs of a job cannot exceed 256. The number of input tables cannot exceed 64.
- Labels that are specified for multiple outputs cannot be null or empty strings. A label cannot exceed 256 strings in length and can contain only letters, digits, underscores (_), number signs (#), periods (.), and hyphens (-).
- The number of custom counters in a job cannot exceed 64. The counter `group name` and `counter name` cannot contain number signs (#). The total length of the two names cannot exceed 100 characters.
- The number of workers for a job is calculated by the framework. The maximum number of workers is 1,000. An exception is thrown if the number of workers exceeds this value.

- A worker consumes 200 units of CPU resources by default. The range of resources consumed is 50 to 800.
- A worker consumes 4,096 MB memory by default. The range of memory consumed is 256 MB to 12 GB.
- A worker can repeatedly read a resource up to 64 times.
- The default value of `split_size` is 64 MB. You can set the value as needed. The value of `split_size` must be greater than 0 and smaller than or equal to the result of the 9223372036854775807>>20 operation.
- GraphLoader, Vertex, and Aggregator in MaxCompute Graph are restricted by the Java sandbox when they are run in a cluster. However, the main program of Graph jobs is not restricted by the Java sandbox. For more information, see Java Sandbox.

4.4.4. Graph jobs

This topic describes how to run a MaxCompute Graph job.

Run a job

The MaxCompute client provides a JAR command for you to run MaxCompute Graph jobs. This command is used in the same way as the JAR command in MapReduce.

Syntax:

```
Usage: jar [<GENERIC_OPTIONS>] <MAIN_CLASS> [ARGS]
    -conf <configuration_file>         Specify an application configuration file
    -classpath <local_file_list>       classpaths used to run mainClass
    -D <name>=<value>                  Property value pair, which will be used to run mainC
lass
    -local                             Run job in local mode
    -resources <resource_name_list>    file/table resources used in graph, seperate by comm
a
```

<GENERIC_OPTIONS> includes the following optional parameters:

- -conf <configuration file>: the JobConf configuration file.
- -classpath <local_file_list>: the classpath used to run a job in local mode. This parameter specifies the JAR package where the main function is located.

  Users prefer to write the main function and Graph job in one package, such as SSSP. Therefore, the JAR package is used in both -resources and -classpath parameters of the sample program. However, the packages used in these two parameters have different meanings:

  - -resources references Graph jobs for distributed execution.
  - -classpath references the main function for local execution. The specified JAR package must also be saved in a local directory. Package names are separated by default file delimiters. Windows operating systems use semicolons (;) as the default file delimiter while Linux operating systems use colons (:).
- -D <prop_name>=<prop_value>: the Java property of <mainClass> for local execution. You can specify multiple properties.
- -local: specifies that the Graph jobs run in local mode. It is used for program debugging.
- -resources <resource_name_list>: the resource used to run a Graph job. You must specify the names of the resources that the Graph job uses in resource_name_list. If you read other MaxCompute

resources while you run the Graph job, you must add the names of those resources to <resource_name_list>. Separate resource names with commas (,). If you use cross-project resources, you must add PROJECT_NAME/resources/ before resource_name_list. Example: `-resources otherproject/resources/resfile`.

You can directly run the main function in the Graph job to submit the job to MaxCompute, instead of submitting the job by using the MaxCompute client. The PageRank algorithm is used in the following example:

```
public static void main(String[] args) throws Exception {
  if (args.length < 2)
    printUsage();
  Account account = new AliyunAccount(accessId, accessKey);
  Odps odps = new Odps(account);
  odps.setEndpoint(endPoint);
  odps.setDefaultProject(project);
  SessionState ss = SessionState.get();
  ss.setOdps(odps);
  ss.setLocalRun(false);
  String resource = "mapreduce-examples.jar";
  GraphJob job = new GraphJob();
  // Add the used JAR file and other files to the class cache resource. These files corresp
ond to those specified by -libjars in the JAR command.
  job.addCacheResourcesToClassPath(resource);
  job.setGraphLoaderClass(PageRankVertexReader.class);
  job.setVertexClass(PageRankVertex.class);
  job.addInput(TableInfo.builder().tableName(args[0]).build());
  job.addOutput(TableInfo.builder().tableName(args[1]).build());
  // default max iteration is 30
  job.setMaxIteration(30);
  if (args.length >= 3)
    job.setMaxIteration(Integer.parseInt(args[2]));
  long startTime = System.currentTimeMillis();
  job.run();
  System.out.println("Job Finished in "
      + (System.currentTimeMillis() - startTime) / 1000.0
      + " seconds");
}
```

Input and output

The input and output of MaxCompute Graph jobs must be tables. You cannot customize the input or output format.

- Job input definition

```
GraphJob job = new GraphJob();
job.addInput(TableInfo.builder().tableName("tblname").build());  // Use tables as the inp
ut.
job.addInput(TableInfo.builder().tableName("tblname").partSpec("pt1=a/pt2=b").build()); /
/ Use partitions as the input.
// Read only the col2 and col0 columns of the input table. Use record.get(0) to obtain co
l2 in the load() method of GraphLoader. Both are read in the same sequence.
job.addInput(TableInfo.builder().tableName("tblname").partSpec("pt1=a/pt2=b").build(), ne
w String[]{"col2", "col0"});
```

> **Note**
> - Multiple inputs are supported.
> - The addInput framework reads records from the input table and transfers the records to the user-defined GraphLoader to load graph data.
> - You cannot specify partition filter conditions. For more information about limits, see Limits of MaxCompute Graph.

- Job output definition

```
GraphJob job = new GraphJob();
// If the output table is a partitioned table, the last level of partitions must be provi
ded.
job.addOutput(TableInfo.builder().tableName("table_name").partSpec("pt1=a/pt2=b").build()
);
// true indicates that the code overwrites partitions specified by tableinfo. The value t
rue is similar to the INSERT OVERWRITE statement. The value false is similar to the INSER
T INTO statement.
job.addOutput(TableInfo.builder().tableName("table_name").partSpec("pt1=a/pt2=b").lable("
output1").build(), true);
```

> **Note**
> - Multiple outputs are supported, and each output is identified by a label.
> - A Graph job can use the write() method of WorkContext to write data to an output table during runtime. Multiple outputs must be labeled.

Read resources

- Use GraphJob to read resources

  In addition to the JAR command, the following two methods of GraphJob can be used to specify the resources read by Graph:

```
void addCacheResources(String resourceNames)
void addCacheResourcesToClassPath(String resourceNames)
```

- Use WorkerContext to read resources

  You can read resources from the WorkerContext object.

```
public byte[] readCacheFile(String resourceName) throws IOException;
public Iterable<byte[]> readCacheArchive(String resourceName) throws IOException;
public Iterable<byte[]> readCacheArchive(String resourceName, String relativePath)throws
IOException;
public Iterable<WritableRecord> readResourceTable(String resourceName);
public BufferedInputStream readCacheFileAsStream(String resourceName) throws IOException;
public Iterable<BufferedInputStream> readCacheArchiveAsStream(String resourceName) throws
IOException;
public Iterable<BufferedInputStream> readCacheArchiveAsStream(String resourceName, String
relativePath) throws IOException;
```

Note

- Resources can also be read by using the setup() method of WorkerComputer, saved in WorkerValue, and obtained by using getWorkerValue.
- If you want to read resources and process them at the same time, you can use the preceding stream APIs. This method reduces memory consumption.
- For more information about limits, see Limits of MaxCompute Graph.

4.4.5. Write a Graph job

This topic describes how to submit a Graph job. In this topic, the single source shortest path (SSSP) algorithm is used as an example.

Prerequisites

- The MaxCompute client is installed and configured.

  For more information about how to install and configure the MaxCompute client, see MaxCompute client.

- MaxCompute Studio is installed and configured.

  For more information about how to install and configure MaxCompute Studio, see Install MaxCompute Studio and Configure MaxCompute Studio.

- Java Development Kit (JDK) 1.8 or later is installed.
- A data file is prepared. The sssp.txt file is used in this example. The sssp.txt file contains the following data:

```
1 2:2,3:1,4:4
2 1:2,3:2,4:1
3 1:1,2:2,5:1
4 1:4,2:1,5:1
5 3:1,4:1
```

1. Run the MaxCompute client and execute the following statements to create an input table named sssp_in and an output table named sssp_out.

```
CREATE TABLE sssp_in (v bigint, es string);
CREATE TABLE sssp_out (vertex bigint, value bigint);
```

> **Note** For more information about how to create a table, see Table operations.

2. Run the Tunnel command to upload data in the sssp.txt file to the sssp_in table. Separate data in different columns with spaces.

```
tunnel u -fd " " sssp.txt sssp_in;
```

> **Note** In this example, the sssp.txt file is stored in the `bin` directory of the MaxCompute client. Take note of the actual directory in which the file is stored.

3. Write SSSP code.
   i. Create a **MaxCompute Java module** named odps-graph-example-sssp in IntelliJ IDEA.

   > **Note** For more information about how to create a MaxCompute Java module, see Create a MaxCompute Java module.

   ii. Create the `BaseLoadingVertexResolver` class and the `SSSP` class in **odps-graph-example-sssp**. For more information about how to create a class, see the sample code for the directed graph that is provided in SSSP.

   iii. In IntelliJ IDEA, package the Java program into a JAR file by using MaxCompute Studio. For more information about how to package a Java program into a JAR file, see Package a Java program, upload the package, and create a MaxCompute UDF.

   > **Note** In this example, the JAR package that is deployed in the MaxCompute project is named odps-graph-example-sssp.jar.

4. Run the following command on the MaxCompute client to implement the SSSP algorithm:

```
jar -libjars odps-graph-example-sssp.jar -classpath <LOCAL_JAR_PATH>/odps-graph-example
-sssp.jar SSSP 1 sssp_in sssp_out;
```

   LOCAL_JAR_PATH: indicates the local path of the odps-graph-example-sssp.jar package.

   The following result is returned:

```
vertex    value
1         0
2         2
3         1
4         3
5         2
```

   - vertex: indicates the current vertex.
   - value: indicates the value of the SSSP from the current vertex to Source Vertex 1.

> **Note** If you want to use the Graph feature, you can directly submit a Graph job.

4.4.6. SDK configuration

This topic describes the classes of MaxCompute Graph SDK for Java and how to configure the SDK.

If you use Maven, you can search for odps-sdk-graph in the Maven repository to find the latest version of the SDK for Java. You can declare the SDK in your project by using the following Maven dependency:

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-graph</artifactId>
    <version>0.20.7</version>
</dependency>
```

Class	Description
GraphJob	Defines, submits, and manages a MaxCompute Graph job. The class inherits JobConf.
Vertex	Defines a vertex in a graph. A vertex has the following properties: id, value, halted, and edges. You can specify this class by using the setVertexClass method of GraphJob.
Edge	Defines an edge in a graph. An edge has the following properties: destVertexId and value. MaxCompute Graph uses the adjacency list as the data structure. The outbound edges of a vertex are specified by the edges property of the vertex.
GraphLoader	Loads a graph. You can specify this class by using the setGraphLoaderClass method of GraphJob.
VertexResolver	Defines the custom logic for handling conflicts during graph topology modification. You can specify this class by using the setLoadingVertexResolverClass and setComputingVertexResolverClass methods of GraphJob. The setLoadingVertexResolverClass method specifies the class that is used during graph loading, whereas the setComputingVertexResolverClass method specifies the class that is used during iterative computing.
Partitioner	Specifies how to distribute vertices to workers. This way, computing can be performed by multiple workers at the same time. You can specify this class by using the setPartitionerClass method of GraphJob. By default, the HashPartitioner class is used. HashPartitioner computes the hash value of a vertex ID and divides the hash value by the number of workers to obtain the remainder of the hash value. This remainder specifies the worker to which the vertex is distributed.
WorkerComputer	Defines the custom operations to perform when a worker starts and stops. You can specify this class by using the setWorkerComputerClass method of GraphJob.
Aggregator	Processes and summarizes global information. You can specify multiple Aggregator classes by using the `setAggregatorClass(Class...)` method of GraphJob.
Combiner	Summarizes the output records with the same key. You can specify this class by using the setCombinerClass method of GraphJob.

Class	Description
Counters	Defines a counter that is used for counting workers. In the operating logic of a job, you can use the WorkerContext class to obtain the counter and count workers. The framework then automatically calculates the sum of the workers.
WorkerContext	Defines the context that encapsulates the features provided by the framework, such as the features that allow workers to modify the graph topology, send messages, write results, and read resources.

4.4.7. Development and debugging

This topic describes how to use Eclipse to develop MaxCompute Graph programs because Graph development plug-ins are unavailable in MaxCompute.

Development process:

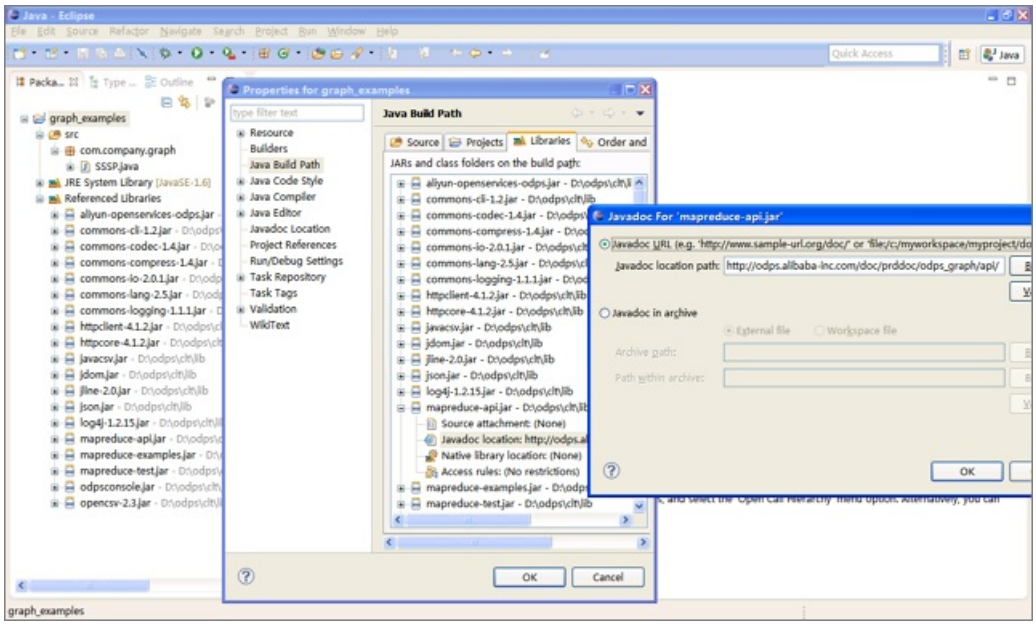
1. Compile Graph code and perform local debugging to test basic functions.
2. Perform cluster debugging to verify the result.

Development example

This topic uses the SSSP algorithm as an example to describe how to use Eclipse to develop and debug a MaxCompute Graph program.

Procedure:

1. Create a Java project named *graph_examples*.
2. Add the JAR package in the *lib* directory on the MaxCompute client to **Java Build Path** of the Eclipse project.



3. Develop a MaxCompute Graph program.

   In the actual development process, an example program, such as SSSP, is first copied and then modified. In this example, only the package path is changed to *package com.aliyun.odps.graph.example*.

4. Compile and package the code.

   In an Eclipse environment, right-click the source code directory, namely, the *src* directory, and choose **Export > Java > JAR file** to generate a JAR package. Select the path to store the JAR package, such as *D:\\odps\\clt\\odps-graph-example-sssp.jar*.

5. Run SSSP on the MaxCompute client. For more information, see (Optional) Submit Graph jobs.

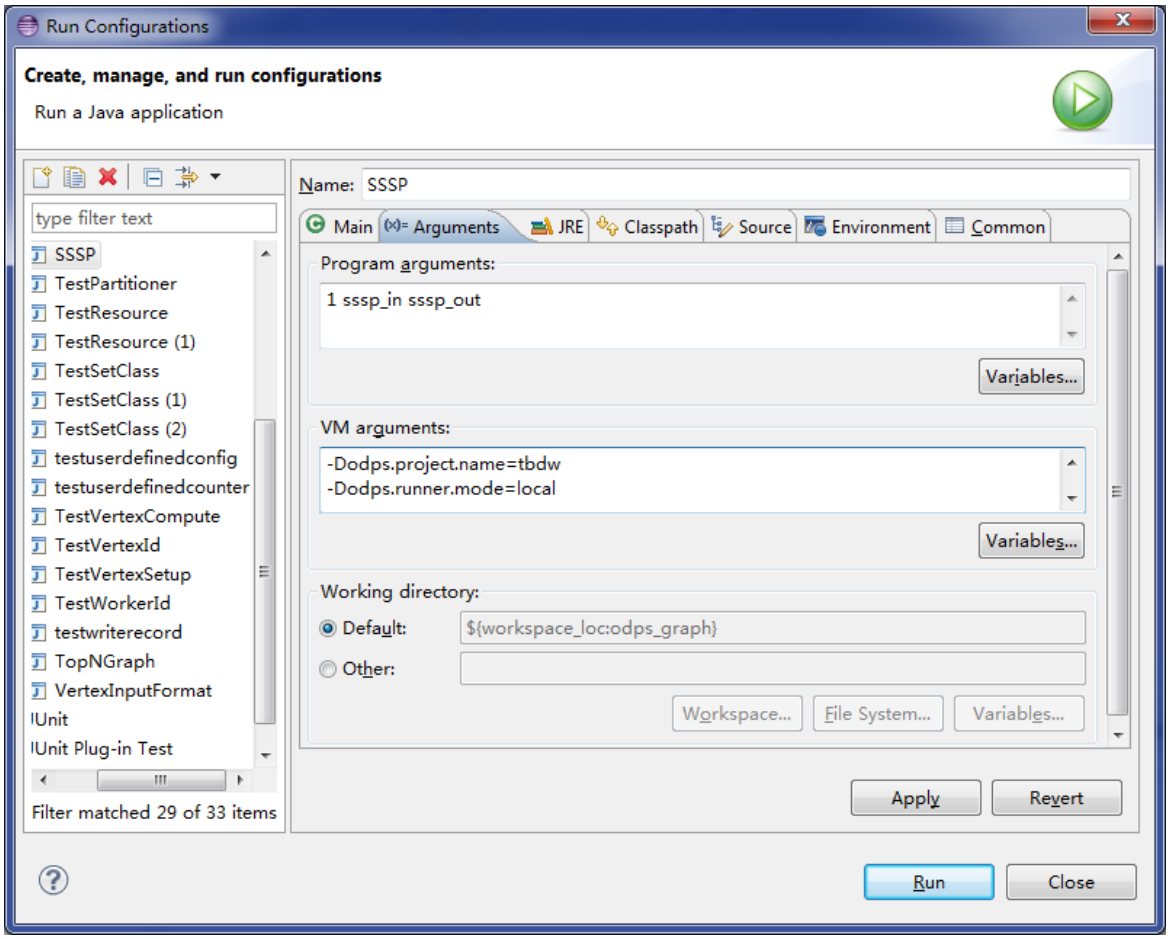
Local debugging

MaxCompute Graph supports the local debugging mode. You can use Eclipse for breakpoint debugging.

Procedure:

1. Download a Maven package named *odps-graph-local*.
2. Select the Eclipse project, right-click the main program file that contains the `main` function of a Graph job, and choose **Run As > Run Configurations** to configure parameters.
3. On the **Arguments** tab, set Program arguments to *1 sssp_in sssp_out* as the input parameter of the main program.
4. On the **Arguments** tab, set VM arguments to the following content:

```
-Dodps.runner.mode=local
-Dodps.project.name=<project.name>
-Dodps.end.point=<end.point>
-Dodps.access.id=<access.id>
-Dodps.access.key=<access.key>
```



5. In local mode in which odps.end.point is not specified, create the *sssp_in* and *sssp_out* tables in the *warehouse* directory and add the following data to the *sssp_in* table:

```
1,"2:2,3:1,4:4"
2,"1:2,3:2,4:1"
3,"1:1,2:2,5:1"
4,"1:4,2:1,5:1"
5,"3:1,4:1"
```

For more information about the *warehouse* directory, see Job running in local mode.

6. Click **Run** to run SSSP on the local client.

> **Note** Configure the parameters based on the settings in *conf/odps_config.ini* on the MaxCompute client. The preceding parameters are commonly used. Descriptions of other parameters:
>
> - odps.runner.mode: The value is *local*. This parameter is required for the local debugging feature.
> - odps.project.name: specifies the current project. This parameter is required.
> - odps.end.point: specifies the endpoint of MaxCompute. This parameter is optional. If this parameter is not specified, metadata and data of tables or resources are only read from the *warehouse* directory. An exception is reported if such data does not exist in the directory. If this parameter is specified, metadata and data are read from the *warehouse* directory first and then from the remote MaxCompute server if such data does not exist in the directory.
> - odps.access.id: specifies the AccessKey ID used to access MaxCompute. This parameter is valid only if odps.end.point is specified.
> - odps.access.key: specifies the AccessKey secret used to access MaxCompute. This parameter is valid only if odps.end.point is specified.
> - odps.cache.resources: specifies the resources you want to use. This parameter functions the same as `-resources` of the jar command.
> - odps.local.warehouse: specifies the local path to *warehouse*. The default value is *./warehouse*.

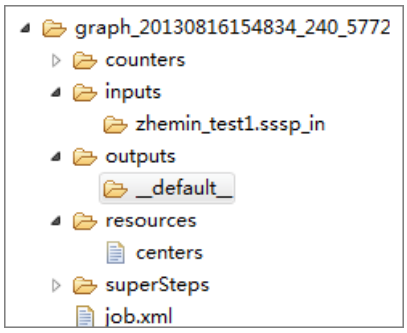
Output of the local SSSP debugging in Eclipse:

```
Counters: 3
        com.aliyun.odps.graph.local.COUNTER
                TASK_INPUT_BYTE=211
                TASK_INPUT_RECORD=5
                TASK_OUTPUT_BYTE=161
                TASK_OUTPUT_RECORD=5
 graph task finish
```

> **Note** In this example, the *sssp_in* and *sssp_out* tables must exist in the local *warehouse* directory. For more information about the *sssp_in* and *sssp_out* tables, see Write a Graph job.

Temporary directory of a local job

A temporary directory is created in the Eclipse project directory each time local debugging is performed.



The temporary directory of a local Graph job contains the following directories and files:

- *counters*: stores the counter information that is generated during the job running.
- *inputs*: stores input data of the job. Data is read from the local *warehouse* directory first. If no data is available, the MaxCompute SDK reads data from the server if odps.end.point is specified. An *input* directory reads only 10 data records by default. This threshold can be changed by using the `-Dodps.mapred.local.record.limit` parameter, of which the maximum value is 10000.
- *outputs*: stores output data of the job. If an output table exists in the local *warehouse* directory, the results in *outputs* will overwrite data in that table after the job is completed.
- *resources*: stores resources used by the job. Similar to input data, resource data is read from the local *warehouse* directory first. If no data is available, the MaxCompute SDK reads data from the server if odps.end.point is specified.
- *job.xml*: stores job configurations.
- *superstep*: stores persistent messages from each iteration.

> **Note** If detailed logs must be recorded during local debugging, you must create a *log4j* configuration file named *log4j.properties_odps_graph_cluster_debug* in the *src* directory.

Cluster debugging

After local debugging, you can submit the job to a cluster for testing.

Procedure:

1. Configure the MaxCompute client.
2. Run the `add jar /path/work.jar -f;` command to update the JAR package.
3. Run a jar command to execute the job and check the operational log and command output.

> **Note** For more information about how to run a Graph job in a cluster, see Write a Graph job.

Performance optimization

Configurations that affect the Graph performance:

- `setSplitSize(long)`: specifies the split size of an input table. The unit is MB. The value must be greater than 0. The default value is 64.
- `setNumWorkers(int)`: specifies the number of workers for a job. The value ranges from 1 to 1000. The default value is 1. The number of workers is determined by the input bytes of the job and `splitSize`.

- `setWorkerCPU(int)`: specifies the CPU resources of the Map. The value ranges from 50 to 800. The default value is 200. A one-core CPU contains 100 resources.
- `setWorkerMemory(int)`: specifies the memory resources of the Map. The unit is MB. The memory ranges from 256 to 12288. The default value is 4096.
- `setMaxIteration(int)`: specifies the maximum number of iterations. The default value is -1. If the value is less than or equal to 0, the job does not stop when the maximum number of iterations is reached.
- `setJobPriority(int)`: specifies the job priority. The value ranges from 0 to 9. The default value is 9. A greater value indicates a lower priority.

We recommend that you optimize the performance by using one or more of the following methods:

- Use `setNumWorkers` to increase the number of workers.
- Use `setSplitSize` to reduce the split size and increase the data loading speed.
- Increase the CPU or memory resources for workers.
- Set the maximum number of iterations. For applications that do not require precise results, you can reduce the number of iterations to accelerate the execution process.

`setNumWorkers` and `setSplitSize` can be used together to accelerate data loading. Assume that the value of `setNumWorkers` equals that of `workerNum`, the value of `setSplitSize` equals that of `splitSize`, and the total number of input bytes is the value of `inputSize`. The number of split data records is calculated by using the following formula: `splitNum = inputSize/splitSize`. Relationship between `workerNum` and `splitNum`:

- If the value of `splitNum` is equal to that of `workerNum`, each worker loads one split data record.
- If the value of `splitNum` is greater than that of `workerNum`, each worker loads one or more split data records.
- If the value of `splitNum` is less than that of `workerNum`, each worker uploads one or no split data record.

Therefore, you can adjust `workerNum` and `splitSize` to obtain a suitable data loading speed. In the first two cases, data is loaded faster. In the iteration phase, you only need to adjust `workerNum`. If you set `runtime partitioning` to False, we recommend that you either use `setSplitSize` to adjust the number of workers or make sure that the conditions in the first two cases are met. In the third case, some workers do not load split data records. Therefore, you can insert `set odps.graph.split.size=<m>; set odps.graph.worker.num=<n>;` before the jar command to achieve the same effect as `setNumWorkers` and `setSplitSize`.

Another common performance issue is data skew. As indicated by the counters, some workers process excessive split data records or edges than others. Data skew occurs when the number of split data records, edges, or messages that correspond to some keys is much greater than other keys. These keys are processed by a small number of workers, which results in longer runtimes. To address this issue, try one or more of the following methods:

- Use a combiner to aggregate the messages of the split data records that correspond to the keys to reduce the number of messages generated.

  Developers can define a combiner to reduce the memory and network traffic consumed by message storage, which reduces the job execution duration.

- Improve the business logic.

If the data volume is large, reading data in a disk may take up the processing time. Therefore, you can reduce the data bytes to be read to increase the overall throughput. This improves job performance. To reduce data bytes, use one of the following methods:

- Reduce data input: For some decision-making applications, processing sampled data only affects the precision of the results, not the overall accuracy. In this case, you can perform special data sampling and import the data to the input table for processing.
- Avoid reading fields that are not used: The TableInfo class of the MaxCompute Graph framework supports reading specific columns that are transferred by using column name arrays, rather than reading the entire table or partition. This reduces the input data volume and improves job performance.

Built-in JAR packages

By default, the following JAR packages are loaded on a JVM that runs Graph programs. You do not need to manually upload these resources or use `-libjars` to specify them in a command:

- *commons-codec-1.3.jar*
- *commons-io-2.0.1.jar*
- *commons-lang-2.5.jar*
- *commons-logging-1.0.4.jar*
- *commons-logging-api-1.0.4.jar*
- *guava-14.0.jar*
- *json.jar*
- *log4j-1.2.15.jar*
- *slf4j-api-1.4.3.jar*
- *slf4j-log4j12-1.4.3.jar*
- *xmlenc-0.52.jar*

> **Note** In the classpath of the JVM, the preceding built-in JAR packages are placed before your JAR packages, which may result in a version conflict. For example, your program calls a specific class function in *commons-codec-1.5.jar*, but the function is not included in *commons-codec-1.3.jar*. In this case, you can choose to call a similar function in commons-codec-1.3.jar or wait until MaxCompute is updated to the required version.

4.4.8. Examples

4.4.8.1. SSSP

Dijkstra's algorithm is a common algorithm that is used to calculate the Single Source Shortest Path (SSSP) in a directed graph.

How the Dijkstra's algorithm works:

- Initialization: The path from s to s is 0 (d[s] = 0), and the path from u to s is infinite (d[u] = ∞).
- Iteration: If an edge from u to v exists, the shortest path from s to v is updated to `d[v] = min(d[v], d[u] + weight(u, v))`. The iteration does not end until the paths from all vertices to s do not change.

> **Note** Shortest path: For a weighted directed graph `G = (V,E)`, multiple paths are available from source vertex s to sink vertex v. The path with the smallest sum of edge weights is called the shortest path from s to v.

The working mode of the algorithm shows that the algorithm is suitable for MaxCompute Graph. Each vertex maintains the current shortest path to the source vertex. If the path changes, the new path is added with the edge weight, and a message is sent to notify adjacent vertices. In the next iteration, the adjacent vertices update the shortest paths based on the received message. If the shortest path between each vertex and the source vertex does not change, the iteration ends.

Sample code

Example:

```
import java.io.IOException;
import com.aliyun.odps.io.WritableRecord;
import com.aliyun.odps.graph.Combiner;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.Edge;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.data.TableInfo;
public class SSSP {
  public static final String START_VERTEX = "sssp.start.vertex.id";
  public static class SSSPVertex extends
      Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
    private static long startVertexId = -1;
    public SSSPVertex() {
      this.setValue(new LongWritable(Long.MAX_VALUE));
    }
    public boolean isStartVertex(
        ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context) {
      if (startVertexId == -1) {
        String s = context.getConfiguration().get(START_VERTEX);
        startVertexId = Long.parseLong(s);
      }
      return getId().get() == startVertexId;
    }
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context,
        Iterable<LongWritable> messages) throws IOException {
      long minDist = isStartVertex(context) ? 0 : Integer.MAX_VALUE;
      for (LongWritable msg : messages) {
        if (msg.get() < minDist) {
          minDist = msg.get();
        }
      }
      if (minDist < this.getValue().get()) {
```

```
        this.setValue(new LongWritable(minDist));
        if (hasEdges()) {
          for (Edge<LongWritable, LongWritable> e : this.getEdges()) {
            context.sendMessage(e.getDestVertexId(), new LongWritable(minDist
                + e.getValue().get()));
          }
        }
      } else {
        voteToHalt();
      }
    }
    @Override
    public void cleanup(
        WorkerContext<LongWritable, LongWritable, LongWritable, LongWritable> context)
        throws IOException {
      context.write(getId(), getValue());
    }
  }
  public static class MinLongCombiner extends
      Combiner<LongWritable, LongWritable> {
    @Override
    public void combine(LongWritable vertexId, LongWritable combinedMessage,
        LongWritable messageToCombine) throws IOException {
      if (combinedMessage.get() > messageToCombine.get()) {
        combinedMessage.set(messageToCombine.get());
      }
    }
  }
  public static class SSSPVertexReader extends
      GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context)
        throws IOException {
      SSSPVertex vertex = new SSSPVertex();
      vertex.setId((LongWritable) record.get(0));
      String[] edges = record.get(1).toString().split(",");
      for (int i = 0; i < edges.length; i++) {
        String[] ss = edges[i].split(":");
        vertex.addEdge(new LongWritable(Long.parseLong(ss[0])),
            new LongWritable(Long.parseLong(ss[1])));
      }
      context.addVertexRequest(vertex);
    }
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2) {
      System.out.println("Usage: <startnode> <input> <output>");
      System.exit(-1);
    }
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(SSSPVertexReader.class);
    job.setVertexClass(SSSPVertex.class);
```

```
    job.setVertexClass(SSSPVertex.class);
    job.setCombinerClass(MinLongCombiner.class);
    job.set(START_VERTEX, args[0]);
    job.addInput(TableInfo.builder().tableName(args[1]).build());
    job.addOutput(TableInfo.builder().tableName(args[2]).build());
    long startTime = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
  }
}
```

Description:

- Row 19: Define `SSSPVertex`.
  - The vertex value indicates the current shortest path from this vertex to the `startVertexId` source vertex.
  - The `compute()` method uses the `d[v] = min(d[v], d[u] + weight(u, v))` iteration formula to update the vertex value.
  - The `cleanup()` method writes the vertex and its shortest path to the source vertex to the result table.
- Row 58: If the vertex value does not change, `voteToHalt()` is called to notify the framework that this vertex enters the halt state. The calculation ends after all vertices enter the halt state.
- Row 70: Define `MinLongCombiner` and combine messages that are sent to the same vertex to optimize performance and reduce memory usage.
- Row 83: Define the `SSSPVertexReader` class, load a graph, and parse each record in the table into a vertex. The first column of the record is the vertex ID, and the second column stores all edge sets that start from the vertex, such as 2:2, 3:1, and 4:4.
- Row 106: Include the `main` function, define `GraphJob`, and specify input and output tables and the implementation of `Vertex`, `GraphLoader`, and `Combiner`.

4.4.8.2. PageRank

PageRank is an algorithm used to rank web pages. The input of PageRank is a directed graph. Each vertex represents a web page. Each edge represents a link between two web pages. If the web pages are not connected, no edge exists between the vertices.

How the PageRank algorithm works:

- Initialization: A vertex value indicates the rank value of PageRank. The rank value is of the DOUBLE type. During initialization, the value of each vertex is `1/TotalNumVertices`.
- Iteration formula:

  `PageRank(i) = 0.15/TotalNumVertices + 0.85 × Value of sum`

  sum is used to add up `PageRank(j)/out_degree(j)` of each vertex that points to center i. j in the formula refers to each vertex that points to center i.

The PageRank algorithm is suitable for MaxCompute Graph programs. Each vertex j maintains its PageRank value and sends `PageRank(j)/out_degree(j)` to its adjacent vertices (for voting) in each iteration. In the next iteration, each vertex recalculates the PageRank value by using the iteration formula.

Sample code

```
import java.io.IOException;
import org.apache.log4j.Logger;
import com.aliyun.odps.io.WritableRecord;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable;
public class PageRank {
  private final static Logger LOG = Logger.getLogger(PageRank.class);
  public static class PageRankVertex extends
      Vertex<Text, DoubleWritable, NullWritable, DoubleWritable> {
    @Override
    public void compute(
        ComputeContext<Text, DoubleWritable, NullWritable, DoubleWritable> context,
        Iterable<DoubleWritable> messages) throws IOException {
      if (context.getSuperstep() == 0) {
        setValue(new DoubleWritable(1.0 / context.getTotalNumVertices()));
      } else if (context.getSuperstep() >= 1) {
        double sum = 0;
        for (DoubleWritable msg : messages) {
          sum += msg.get();
        }
        DoubleWritable vertexValue = new DoubleWritable(
            (0.15f / context.getTotalNumVertices()) + 0.85f * sum);
        setValue(vertexValue);
      }
      if (hasEdges()) {
        context.sendMessageToNeighbors(this, new DoubleWritable(getValue()
            .get() / getEdges().size()));
      }
    }
    @Override
    public void cleanup(
        WorkerContext<Text, DoubleWritable, NullWritable, DoubleWritable> context)
        throws IOException {
      context.write(getId(), getValue());
    }
  }
  public static class PageRankVertexReader extends
```

```
      GraphLoader<Text, DoubleWritable, NullWritable, DoubleWritable> {
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<Text, DoubleWritable, NullWritable, DoubleWritable> context)
        throws IOException {
      PageRankVertex vertex = new PageRankVertex();
      vertex.setValue(new DoubleWritable(0));
      vertex.setId((Text) record.get(0));
      System.out.println(record.get(0));
      for (int i = 1; i < record.size(); i++) {
        Writable edge = record.get(i);
        System.out.println(edge.toString());
        if (!( edge.equals(NullWritable.get()))) {
          vertex.addEdge(new Text(edge.toString()), NullWritable.get());
        }
      }
      LOG.info("vertex edgs size: "
          + (vertex.hasEdges() ? vertex.getEdges().size() : 0));
      context.addVertexRequest(vertex);
    }
  }
  private static void printUsage() {
    System.out.println("Usage: <in> <out> [Max iterations (default 30)]");
    System.exit(-1);
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2)
      printUsage();
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(PageRankVertexReader.class);
    job.setVertexClass(PageRankVertex.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    // default max iteration is 30
    job.setMaxIteration(30);
    if (args.length >= 3)
      job.setMaxIteration(Integer.parseInt(args[2]));
    long startTime = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
  }
}
```

Description:

- Row 23: Define the PageRankVertex class.
  - The vertex value indicates the current PageRank value of the vertex (web page).

- The compute() method uses the following iteration formula to update the vertex value: `PageRank(i) = 0.15/TotalNumVertices + 0.85 × Value of sum`
- The cleanup() method writes the vertex and its PageRank value to the result table.

- Row 55: Define the PageRankVertexReader class, load a graph, and resolve each record in the table into a vertex. The first column of the table is the source vertices and other columns are the destination vertices.
- Row 88: Include the main function, define the GraphJob class, and specify the maximum number of iterations, the input and output tables, and the implementation of Vertex and GraphLoader. By default, a maximum of 30 iterations can be performed.

4.4.8.3. K-means clustering

K-means clustering is a basic clustering algorithm that is widely used.

How k-means clustering works: Clustering is performed around k points in space, and the closest vertices are classified. The values of the clustering centers are updated in sequence by using iterations until the optimal clustering result is obtained.

Procedure to divide the sample set into k classes:

1. Select the initial centers of k classes.
2. In the ith iteration, select a sample, calculate its distance to k centers, and then classify the sample into the class of the center with the shortest distance.
3. Use the mean method to update the center value of the class.
4. For all the k centers, if the value remains unchanged or is less than a threshold after the update, the iteration ends. Otherwise, the iteration continues.

Sample code

The following example shows the code for the k-means clustering algorithm:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.log4j.Logger;
import com.aliyun.odps.io.WritableRecord;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
public class Kmeans {
  private final static Logger LOG = Logger.getLogger(Kmeans.class);
  public static class KmeansVertex extends
```

```
public static class KmeansVertex extends
    Vertex<Text, Tuple, NullWritable, NullWritable> {
  @Override
  public void compute(
      ComputeContext<Text, Tuple, NullWritable, NullWritable> context,
      Iterable<NullWritable> messages) throws IOException {
    context.aggregate(getValue());
  }
}
public static class KmeansVertexReader extends
    GraphLoader<Text, Tuple, NullWritable, NullWritable> {
  @Override
  public void load(LongWritable recordNum, WritableRecord record,
      MutationContext<Text, Tuple, NullWritable, NullWritable> context)
      throws IOException {
    KmeansVertex vertex = new KmeansVertex();
    vertex.setId(new Text(String.valueOf(recordNum.get())));
    vertex.setValue(new Tuple(record.getAll()));
    context.addVertexRequest(vertex);
  }
}
public static class KmeansAggrValue implements Writable {
  Tuple centers = new Tuple();
  Tuple sums = new Tuple();
  Tuple counts = new Tuple();
  @Override
  public void write(DataOutput out) throws IOException {
    centers.write(out);
    sums.write(out);
    counts.write(out);
  }
  @Override
  public void readFields(DataInput in) throws IOException {
    centers = new Tuple();
    centers.readFields(in);
    sums = new Tuple();
    sums.readFields(in);
    counts = new Tuple();
    counts.readFields(in);
  }
  @Override
  public String toString() {
    return "centers " + centers.toString() + ", sums " + sums.toString()
        + ", counts " + counts.toString();
  }
}
public static class KmeansAggregator extends Aggregator<KmeansAggrValue> {
  @SuppressWarnings("rawtypes")
  @Override
  public KmeansAggrValue createInitialValue(WorkerContext context)
      throws IOException {
    KmeansAggrValue aggrVal = null;
    if (context.getSuperstep() == 0) {
      aggrVal = new KmeansAggrValue();
      aggrVal.centers = new Tuple();
```

```
        aggrVal.sums = new Tuple();
        aggrVal.counts = new Tuple();
        byte[] centers = context.readCacheFile("centers");
        String lines[] = new String(centers).split("\n");
        for (int i = 0; i < lines.length; i++) {
          String[] ss = lines[i].split(",");
          Tuple center = new Tuple();
          Tuple sum = new Tuple();
          for (int j = 0; j < ss.length; ++j) {
            center.append(new DoubleWritable(Double.valueOf(ss[j].trim())));
            sum.append(new DoubleWritable(0.0));
          }
          LongWritable count = new LongWritable(0);
          aggrVal.sums.append(sum);
          aggrVal.counts.append(count);
          aggrVal.centers.append(center);
        }
      } else {
        aggrVal = (KmeansAggrValue) context.getLastAggregatedValue(0);
      }
      return aggrVal;
    }
    @Override
    public void aggregate(KmeansAggrValue value, Object item) {
      int min = 0;
      double mindist = Double.MAX_VALUE;
      Tuple point = (Tuple) item;
      for (int i = 0; i < value.centers.size(); i++) {
        Tuple center = (Tuple) value.centers.get(i);
        // use Euclidean Distance, no need to calculate sqrt
        double dist = 0.0d;
        for (int j = 0; j < center.size(); j++) {
          double v = ((DoubleWritable) point.get(j)).get()
              - ((DoubleWritable) center.get(j)).get();
          dist += v * v;
        }
        if (dist < mindist) {
          mindist = dist;
          min = i;
        }
      }
      // update sum and count
      Tuple sum = (Tuple) value.sums.get(min);
      for (int i = 0; i < point.size(); i++) {
        DoubleWritable s = (DoubleWritable) sum.get(i);
        s.set(s.get() + ((DoubleWritable) point.get(i)).get());
      }
      LongWritable count = (LongWritable) value.counts.get(min);
      count.set(count.get() + 1);
    }
    @Override
    public void merge(KmeansAggrValue value, KmeansAggrValue partial) {
      for (int i = 0; i < value.sums.size(); i++) {
        Tuple sum = (Tuple) value.sums.get(i);
```

```
        Tuple that = (Tuple) partial.sums.get(i);
        for (int j = 0; j < sum.size(); j++) {
          DoubleWritable s = (DoubleWritable) sum.get(j);
          s.set(s.get() + ((DoubleWritable) that.get(j)).get());
        }
      }
      for (int i = 0; i < value.counts.size(); i++) {
        LongWritable count = (LongWritable) value.counts.get(i);
        count.set(count.get() + ((LongWritable) partial.counts.get(i)).get());
      }
    }
    @SuppressWarnings("rawtypes")
    @Override
    public boolean terminate(WorkerContext context, KmeansAggrValue value)
        throws IOException {
      // compute new centers
      Tuple newCenters = new Tuple(value.sums.size());
      for (int i = 0; i < value.sums.size(); i++) {
        Tuple sum = (Tuple) value.sums.get(i);
        Tuple newCenter = new Tuple(sum.size());
        LongWritable c = (LongWritable) value.counts.get(i);
        for (int j = 0; j < sum.size(); j++) {
          DoubleWritable s = (DoubleWritable) sum.get(j);
          double val = s.get() / c.get();
          newCenter.set(j, new DoubleWritable(val));
          // reset sum for next iteration
          s.set(0.0d);
        }
        // reset count for next iteration
        c.set(0);
        newCenters.set(i, newCenter);
      }
      // update centers
      Tuple oldCenters = value.centers;
      value.centers = newCenters;
      LOG.info("old centers: " + oldCenters + ", new centers: " + newCenters);
      // compare new/old centers
      boolean converged = true;
      for (int i = 0; i < value.centers.size() && converged; i++) {
        Tuple oldCenter = (Tuple) oldCenters.get(i);
        Tuple newCenter = (Tuple) newCenters.get(i);
        double sum = 0.0d;
        for (int j = 0; j < newCenter.size(); j++) {
          double v = ((DoubleWritable) newCenter.get(j)).get()
              - ((DoubleWritable) oldCenter.get(j)).get();
          sum += v * v;
        }
        double dist = Math.sqrt(sum);
        LOG.info("old center: " + oldCenter + ", new center: " + newCenter
            + ", dist: " + dist);
        // converge threshold for each center: 0.05
        converged = dist < 0.05d;
      }
      if (converged || context.getSuperstep() == context.getMaxIteration() - 1) {
```

```
        // converged or reach max iteration, output centers
        for (int i = 0; i < value.centers.size(); i++) {
          context.write(((Tuple) value.centers.get(i)).toArray());
        }
        // true means to terminate iteration
        return true;
      }
      // false means to continue iteration
      return false;
    }
  }
  private static void printUsage() {
    System.out.println("Usage: <in> <out> [Max iterations (default 30)]");
    System.exit(-1);
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2)
      printUsage();
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(KmeansVertexReader.class);
    job.setRuntimePartitioning(false);
    job.setVertexClass(KmeansVertex.class);
    job.setAggregatorClass(KmeansAggregator.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    // default max iteration is 30
    job.setMaxIteration(30);
    if (args.length >= 3)
      job.setMaxIteration(Integer.parseInt(args[2]));
    long start = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - start) / 1000.0 + " seconds");
  }
}
```

Description:

- Row 26: Define the `KmeansVertex` class. The `compute()` method is simple. It calls the `aggregate()` method of the context object and pass in the value of the current vertex. The value is of the TUPLE type and expressed by vector.
- Row 38: Define the `KmeansVertexReader` class, load a graph, and parse each record in the table as a vertex. The transmitted value of `recordNum` is used as the vertex ID. The vertex value is a tuple that consists of all columns in the record.
- Row 83: Define the `KmeansAggregator` class. This class encapsulates the main logic of the k-means clustering algorithm.
  - `createInitialValue` is the initial value (the center point for each of the k classes) that is created for each iteration. In the first iteration (superstep 0), the value of this parameter is the initial center point. In other iterations, the value is the new center point when the previous iteration ends.

- The `aggregate()` method calculates the distance from each vertex to the centers of different classes, classifies the vertex into the class of the nearest center, and updates `sum` and `count` of the class.
- The `merge()` method combines `sums` and `counts` collected by each worker.
- The `terminate()` method calculates a new center point based on `sum` and `count` of each class. If the distance between the original and new center points is less than a threshold or the number of iterations reaches the upper limit, the iteration ends, and False is returned. The final center point is written to the result table.

- Row 236: Include the `main` function, define the `GraphJob` class, and specify the maximum number of iterations, the input and output tables, and the implementation of `Vertex`, `GraphLoader`, and `Aggregator`. By default, a maximum of 30 iterations can be performed.
- Row 243: Define `job.setRuntimePartitioning(false)`. For the k-means clustering algorithm, vertices do not need to be distributed for graph loading. `RuntimePartitioning` is set to False to improve the performance of graph loading.

4.4.8.4. Bipartite matching

A bipartite graph is a graph where vertices are divided into two disjoint and independent sets and each edge connects a vertex in one set to a vertex in the other set. In a bipartite graph, a matching is a set of pairwise non-adjacent edges in which no two edges share a common vertex. Bipartite matching is often used for information matching in scenarios with clear supply and demand relationships, such as online dating websites.

The following algorithm can be used to find a matching in a bipartite graph:

- From the first vertex on the left, select another unmatched vertex by following an alternating path to find an augmenting path.
- If the unmatched vertex is found, an augmenting path is found.
- Update path information, increase the number of matched edges by 1, and stop searching.
- If no augmenting path is found, begin with another vertex.

Sample code

The following sample code implements the bipartite matching algorithm:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.Random;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
```

```
public class BipartiteMatching {
  private static final Text UNMATCHED = new Text("UNMATCHED");
  public static class TextPair implements Writable {
    public Text first;
    public Text second;
    public TextPair() {
      first = new Text();
      second = new Text();
    }
    public TextPair(Text first, Text second) {
      this.first = new Text(first);
      this.second = new Text(second);
    }
    @Override
    public void write(DataOutput out) throws IOException {
      first.write(out);
      second.write(out);
    }
    @Override
    public void readFields(DataInput in) throws IOException {
      first = new Text();
      first.readFields(in);
      second = new Text();
      second.readFields(in);
    }
    @Override
    public String toString() {
      return first + ": " + second;
    }
  }
  public static class BipartiteMatchingVertexReader extends
      GraphLoader<Text, TextPair, NullWritable, Text> {
    @Override
    public void load(LongWritable recordNum, WritableRecord record,
        MutationContext<Text, TextPair, NullWritable, Text> context)
        throws IOException {
      BipartiteMatchingVertex vertex = new BipartiteMatchingVertex();
      vertex.setId((Text) record.get(0));
      vertex.setValue(new TextPair(UNMATCHED, (Text) record.get(1)));
      String[] adjs = record.get(2).toString().split(",");
      for (String adj : adjs) {
        vertex.addEdge(new Text(adj), null);
      }
      context.addVertexRequest(vertex);
    }
  }
  public static class BipartiteMatchingVertex extends
      Vertex<Text, TextPair, NullWritable, Text> {
    private static final Text LEFT = new Text("LEFT");
    private static final Text RIGHT = new Text("RIGHT");
    private static Random rand = new Random();
    @Override
    public void compute(
        ComputeContext<Text, TextPair, NullWritable, Text> context,
```

```
      Iterable<Text> messages) throws IOException {
    if (isMatched()) {
      voteToHalt();
      return;
    }
    switch ((int) context.getSuperstep() % 4) {
    case 0:
      if (isLeft()) {
        context.sendMessageToNeighbors(this, getId());
      }
      break;
    case 1:
      if (isRight()) {
        Text luckyLeft = null;
        for (Text message : messages) {
          if (luckyLeft == null) {
            luckyLeft = new Text(message);
          } else {
            if (rand.nextInt(1) == 0) {
              luckyLeft.set(message);
            }
          }
        }
        if (luckyLeft != null) {
          context.sendMessage(luckyLeft, getId());
        }
      }
      break;
    case 2:
      if (isLeft()) {
        Text luckyRight = null;
        for (Text msg : messages) {
          if (luckyRight == null) {
            luckyRight = new Text(msg);
          } else {
            if (rand.nextInt(1) == 0) {
              luckyRight.set(msg);
            }
          }
        }
        if (luckyRight != null) {
          setMatchVertex(luckyRight);
          context.sendMessage(luckyRight, getId());
        }
      }
      break;
    case 3:
      if (isRight()) {
        for (Text msg : messages) {
          setMatchVertex(msg);
        }
      }
      break;
    }
  }
```

```
    }
    @Override
    public void cleanup(
        WorkerContext<Text, TextPair, NullWritable, Text> context)
        throws IOException {
      context.write(getId(), getValue().first);
    }
    private boolean isMatched() {
      return !getValue().first.equals(UNMATCHED);
    }
    private boolean isLeft() {
      return getValue().second.equals(LEFT);
    }
    private boolean isRight() {
      return getValue().second.equals(RIGHT);
    }
    private void setMatchVertex(Text matchVertex) {
      getValue().first.set(matchVertex);
    }
  }
  private static void printUsage() {
    System.err.println("BipartiteMatching <input> <output> [maxIteration]");
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2) {
      printUsage();
    }
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(BipartiteMatchingVertexReader.class);
    job.setVertexClass(BipartiteMatchingVertex.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    int maxIteration = 30;
    if (args.length > 2) {
      maxIteration = Integer.parseInt(args[2]);
    }
    job.setMaxIteration(maxIteration);
    job.run();
  }
}
```

4.4.8.5. Strongly connected component

A directed graph is called a strongly connected graph if every vertex is reachable from every other vertex. A strongly connected sub-graph with a large number of vertices in a directed graph is called a strongly connected component.

The algorithm for strongly connected components is based on the parallel coloring algorithm. For more information, see Optimizing Graph Algorithms on Pregel-like Systems. Each vertex contains two fields:

- colorID: stores the color of Vertex v during forward traversal. At the end of computing, vertices with the same colorID belong to the same strongly connected component.
- transposeNeighbors: stores neighbor IDs of Vertex v in the transpose graph of the input graph.

The algorithm is implemented in the following steps:

- Transpose graph formation: contains two supersteps. In the first superstep, each vertex sends a message with its ID to all its outgoing neighbors. These IDs are stored in transposeNeighbors in the second superstep.
- Trimming: contains one superstep. Each vertex with only one incoming or outgoing edge sets its colorID to its own ID, and becomes inactive. Subsequent messages sent to these vertices are ignored.
- Forward traversal: contains two subphases (supersteps): Start and Rest. In the Start phase, each vertex sets its colorID to its own ID, and sends the ID to outgoing neighbors. In the Rest phase, each vertex uses the maximum colorID it received to update its own colorID, and propagates the colorID until the colorIDs converge. When the colorIDs converge, the master process sets the phase to backward traversal.
- Backward traversal: contains two subphases, Start and Rest. In the Start phase, each vertex whose ID equals its colorID propagates its ID to the vertices in transposeNeighbors and sets its status as inactive. Subsequent messages sent to these vertices are ignored. In each of the Rest phase supersteps, each vertex receives a message matching its colorID, propagates its colorID in the transpose graph, and sets its status as inactive. If any vertex remains active, the process goes back to the trimming phase.

Sample code

The following code is a sample implementation of the algorithm for strongly connected components.

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.BooleanWritable;
import com.aliyun.odps.io.IntWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
/**
 * Definition from Wikipedia:
 * In the mathematical theory of directed graphs, a graph is said
 * to be strongly connected if every vertex is reachable from every
 * other vertex. The strongly connected components of an arbitrary
 * directed graph form a partition into subgraphs that are themselves
 * strongly connected.
 *
 * Algorithms with four phases as follows.
 * 1. Transpose Graph Formation: Requires two supersteps. In the first
 * superstep, each vertex sends a message with its ID to all its outgoing
 * neighbors, which in the second superstep are stored in transposeNeighbors.
 *
```

```
 * 2. Trimming: Takes one superstep. Every vertex with only in-coming or
 *  only outgoing edges (or neither) sets its colorID to its own ID and
 *  becomes inactive. Messages subsequently sent to the vertex are ignored.
 *
 * 3. Forward-Traversal: There are two sub phases: Start and Rest. In the
 * Start phase, each vertex sets its colorID to its own ID and propagates
 * its ID to its outgoing neighbors. In the Rest phase, vertices update
 * their own colorIDs with the minimum colorID they have seen, and propagate
 * their colorIDs, if updated, until the colorIDs converge.
 * Set the phase to Backward-Traversal when the colorIDs converge.
 *
 * 4. Backward-Traversal: We again break the phase into Start and Rest.
 * In Start, every vertex whose ID equals its colorID propagates its ID to
 * the vertices in transposeNeighbors and sets itself inactive. Messages
 * subsequently sent to the vertex are ignored. In each of the Rest phase supersteps,
 * each vertex receiving a message that matches its colorID: (1) propagates
 * its colorID in the transpose graph; (2) sets itself inactive. Messages
 * subsequently sent to the vertex are ignored. Set the phase back to Trimming
 * if not all vertex are inactive.
 *
 * http://ilpubs.stanford.edu:8090/1077/3/p535-salihoglu.pdf
 */
public class StronglyConnectedComponents {
  public final static int STAGE_TRANSPOSE_1 = 0;
  public final static int STAGE_TRANSPOSE_2 = 1;
  public final static int STAGE_TRIMMING = 2;
  public final static int STAGE_FW_START = 3;
  public final static int STAGE_FW_REST = 4;
  public final static int STAGE_BW_START = 5;
  public final static int STAGE_BW_REST = 6;
  /**
   * The value is composed of component id, incoming neighbors,
   * active status and updated status.
   */
  public static class MyValue implements Writable {
    LongWritable sccID;// strongly connected component id
    Tuple inNeighbors; // transpose neighbors
    BooleanWritable active; // vertex is active or not
    BooleanWritable updated; // sccID is updated or not
    public MyValue() {
      this.sccID = new LongWritable(Long.MAX_VALUE);
      this.inNeighbors = new Tuple();
      this.active = new BooleanWritable(true);
      this.updated = new BooleanWritable(false);
    }
    public void setSccID(LongWritable sccID) {
      this.sccID = sccID;
    }
    public LongWritable getSccID() {
      return this.sccID;
    }
    public void setInNeighbors(Tuple inNeighbors) {
      this.inNeighbors = inNeighbors;
    }
```

```
  public Tuple getInNeighbors() {
    return this.inNeighbors;
  }
  public void addInNeighbor(LongWritable neighbor) {
    this.inNeighbors.append(new LongWritable(neighbor.get()));
  }
  public boolean isActive() {
    return this.active.get();
  }
  public void setActive(boolean status) {
    this.active.set(status);
  }
  public boolean isUpdated() {
    return this.updated.get();
  }
  public void setUpdated(boolean update) {
    this.updated.set(update);
  }
  @Override
  public void write(DataOutput out) throws IOException {
    this.sccID.write(out);
    this.inNeighbors.write(out);
    this.active.write(out);
    this.updated.write(out);
  }
  @Override
  public void readFields(DataInput in) throws IOException {
    this.sccID.readFields(in);
    this.inNeighbors.readFields(in);
    this.active.readFields(in);
    this.updated.readFields(in);
  }
  @Override
  public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("sccID: " + sccID.get());
    sb.append(" inNeighbores: " + inNeighbors.toDelimitedString(','));
    sb.append(" active: " + active.get());
    sb.append(" updated: " + updated.get());
    return sb.toString();
  }
}
public static class SCCVertex extends
Vertex<LongWritable, MyValue, NullWritable, LongWritable> {
  public SCCVertex() {
    this.setValue(new MyValue());
  }
  @Override
  public void compute(
      ComputeContext<LongWritable, MyValue, NullWritable, LongWritable> context,
      Iterable<LongWritable> msgs) throws IOException {
    // Messages sent to inactive vertex are ignored.
    if (! this.getValue().isActive()) {
      this.voteToHalt();
```

```
      return;
    }
    int stage = ((SCCAggrValue)context.getLastAggregatedValue(0)).getStage();
    switch (stage) {
    case STAGE_TRANSPOSE_1:
      context.sendMessageToNeighbors(this, this.getId());
      break;
    case STAGE_TRANSPOSE_2:
      for (LongWritable msg: msgs) {
        this.getValue().addInNeighbor(msg);
      }
    case STAGE_TRIMMING:
      this.getValue().setSccID(getId());
      if (this.getValue().getInNeighbors().size() == 0 ||
          this.getNumEdges() == 0) {
        this.getValue().setActive(false);
      }
      break;
    case STAGE_FW_START:
      this.getValue().setSccID(getId());
      context.sendMessageToNeighbors(this, this.getValue().getSccID());
      break;
    case STAGE_FW_REST:
      long minSccID = Long.MAX_VALUE;
      for (LongWritable msg : msgs) {
        if (msg.get() < minSccID) {
          minSccID = msg.get();
        }
      }
      if (minSccID < this.getValue().getSccID().get()) {
        this.getValue().setSccID(new LongWritable(minSccID));
        context.sendMessageToNeighbors(this, this.getValue().getSccID());
        this.getValue().setUpdated(true);
      } else {
        this.getValue().setUpdated(false);
      }
      break;
    case STAGE_BW_START:
      if (this.getId().equals(this.getValue().getSccID())) {
        for (Writable neighbor : this.getValue().getInNeighbors().getAll()) {
          context.sendMessage((LongWritable)neighbor, this.getValue().getSccID());
        }
        this.getValue().setActive(false);
      }
      break;
    case STAGE_BW_REST:
      this.getValue().setUpdated(false);
      for (LongWritable msg : msgs) {
        if (msg.equals(this.getValue().getSccID())) {
          for (Writable neighbor : this.getValue().getInNeighbors().getAll()) {
            context.sendMessage((LongWritable)neighbor, this.getValue().getSccID());
          }
          this.getValue().setActive(false);
          this.getValue().setUpdated(true);
```

```
            break;
          }
        }
        break;
      }
      context.aggregate(0, getValue());
    }
    @Override
    public void cleanup(
        WorkerContext<LongWritable, MyValue, NullWritable, LongWritable> context)
        throws IOException {
      context.write(getId(), getValue().getSccID());
    }
  }
  /**
   * The SCCAggrValue maintains global stage and graph updated and active status.
   * updated is true only if one vertex is updated.
   * active is true only if one vertex is active.
   */
  public static class SCCAggrValue implements Writable {
    IntWritable stage = new IntWritable(STAGE_TRANSPOSE_1);
    BooleanWritable updated = new BooleanWritable(false);
    BooleanWritable active = new BooleanWritable(false);
    public void setStage(int stage) {
      this.stage.set(stage);
    }
    public int getStage() {
      return this.stage.get();
    }
    public void setUpdated(boolean updated) {
      this.updated.set(updated);
    }
    public boolean getUpdated() {
      return this.updated.get();
    }
    public void setActive(boolean active) {
      this.active.set(active);
    }
    public boolean getActive() {
      return this.active.get();
    }
    @Override
    public void write(DataOutput out) throws IOException {
      this.stage.write(out);
      this.updated.write(out);
      this.active.write(out);
    }
    @Override
    public void readFields(DataInput in) throws IOException {
      this.stage.readFields(in);
      this.updated.readFields(in);
      this.active.readFields(in);
    }
  }
  /**
```

```
/**
 * The job of SCCAggregator is to schedule global stage in every superstep.
 */
public static class SCCAggregator extends Aggregator<SCCAggrValue> {
  @SuppressWarnings("rawtypes")
  @Override
  public SCCAggrValue createStartupValue(WorkerContext context) throws IOException {
    return new SCCAggrValue();
  }
  @SuppressWarnings("rawtypes")
  @Override
  public SCCAggrValue createInitialValue(WorkerContext context)
      throws IOException {
    return (SCCAggrValue) context.getLastAggregatedValue(0);
  }
  @Override
  public void aggregate(SCCAggrValue value, Object item) throws IOException {
    MyValue v = (MyValue)item;
    if ((value.getStage() == STAGE_FW_REST || value.getStage() == STAGE_BW_REST)
        && v.isUpdated()) {
      value.setUpdated(true);
    }
    // only active vertex invoke aggregate()
    value.setActive(true);
  }
  @Override
  public void merge(SCCAggrValue value, SCCAggrValue partial)
      throws IOException {
    boolean updated = value.getUpdated() || partial.getUpdated();
    value.setUpdated(updated);
    boolean active = value.getActive() || partial.getActive();
    value.setActive(active);
  }
  @SuppressWarnings("rawtypes")
  @Override
  public boolean terminate(WorkerContext context, SCCAggrValue value)
      throws IOException {
    // If all vertices is inactive, job is over.
    if (! value.getActive()) {
      return true;
    }
    // state machine
    switch (value.getStage()) {
    case STAGE_TRANSPOSE_1:
      value.setStage(STAGE_TRANSPOSE_2);
      break;
    case STAGE_TRANSPOSE_2:
      value.setStage(STAGE_TRIMMING);
      break;
    case STAGE_TRIMMING:
      value.setStage(STAGE_FW_START);
      break;
    case STAGE_FW_START:
      value.setStage(STAGE_FW_REST);
      break;
```

```
        break;
      case STAGE_FW_REST:
        if (value.getUpdated()) {
          value.setStage(STAGE_FW_REST);
        } else {
          value.setStage(STAGE_BW_START);
        }
        break;
      case STAGE_BW_START:
        value.setStage(STAGE_BW_REST);
        break;
      case STAGE_BW_REST:
        if (value.getUpdated()) {
          value.setStage(STAGE_BW_REST);
        } else {
          value.setStage(STAGE_TRIMMING);
        }
        break;
      }
      value.setActive(false);
      value.setUpdated(false);
      return false;
    }
  }
  public static class SCCVertexReader extends
  GraphLoader<LongWritable, MyValue, NullWritable, LongWritable> {
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, MyValue, NullWritable, LongWritable> context)
    throws IOException {
      SCCVertex vertex = new SCCVertex();
      vertex.setId((LongWritable) record.get(0));
      String[] edges = record.get(1).toString().split(",");
      for (int i = 0; i < edges.length; i++) {
        try {
          long destID = Long.parseLong(edges[i]);
          vertex.addEdge(new LongWritable(destID), NullWritable.get());
        } catch(NumberFormatException nfe) {
          System.err.println("Ignore " + nfe);
        }
      }
      context.addVertexRequest(vertex);
    }
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2) {
      System.out.println("Usage: <input> <output>");
      System.exit(-1);
    }
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(SCCVertexReader.class);
    job.setVertexClass(SCCVertex.class);
    job.setAggregatorClass(SCCAggregator.class);
```

```
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    long startTime = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
  }
}
```

4.4.8.6. Connected component

Two vertices are connected if a path exists between them. If any two vertices in undirected graph G are connected, G is called a connected graph. Otherwise, G is called an unconnected graph. A connected sub-graph with a large number of vertices is called a connected component.

This algorithm calculates connected component members of each vertex, and provides the connected component of the vertex value that includes the smallest vertex ID. The smallest vertex ID is transmitted along edges to all vertices of the connected component.

Sample code

Code for connected components:

```
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.graph.examples.SSSP.MinLongCombiner;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.WritableRecord;
/**
 * Compute the connected component membership of each vertex and output
 * each vertex which's value containing the smallest id in the connected
 * component containing that vertex.
 *
 * Algorithm: propagate the smallest vertex id along the edges to all
 * vertices of a connected component.
 *
 */
public class ConnectedComponents {
  public static class CCVertex extends
    Vertex<LongWritable, LongWritable, NullWritable, LongWritable> {
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, NullWritable, LongWritable> context,
        Iterable<LongWritable> msgs) throws IOException {
      if (context.getSuperstep() == 0L) {
        this.setValue(getId());
```

```
        context.sendMessageToNeighbors(this, getValue());
        return;
      }
      long minID = Long.MAX_VALUE;
      for (LongWritable id : msgs) {
        if (id.get() < minID) {
          minID = id.get();
        }
      }
      if (minID < this.getValue().get()) {
        this.setValue(new LongWritable(minID));
        context.sendMessageToNeighbors(this, getValue());
      } else {
        this.voteToHalt();
      }
    }
    /**
     * Output Table Description:
     * +-----------------+----------------------------------------+
     * | Field | Type    | Comment                                |
     * +-----------------+----------------------------------------+
     * | v     | bigint  | vertex id                              |
     * | minID | bigint  | smallest id in the connected component |
     * +-----------------+----------------------------------------+
     */
    @Override
    public void cleanup(
        WorkerContext<LongWritable, LongWritable, NullWritable, LongWritable> context)
        throws IOException {
      context.write(getId(), getValue());
    }
  }
  /**
   * Input Table Description:
   * +-----------------+-----------------------------------------------------+
   * | Field | Type    | Comment                                             |
   * +-----------------+-----------------------------------------------------+
   * | v     | bigint  | vertex id                                           |
   * | es    | string  | comma separated target vertex id of outgoing edges |
   * +-----------------+-----------------------------------------------------+
   *
   * Example:
   * For graph:
   *       1 ----- 2
   *       |       |
   *       3 ----- 4
   * Input table:
   * +-----------+
   * | v  | es   |
   * +-----------+
   * | 1  | 2,3  |
   * | 2  | 1,4  |
   * | 3  | 1,4  |
   * | 4  | 2,3  |
   * +-----------+
```

```
   * +------------+
   */
  public static class CCVertexReader extends
    GraphLoader<LongWritable, LongWritable, NullWritable, LongWritable> {
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, NullWritable, LongWritable> context)
    throws IOException {
      CCVertex vertex = new CCVertex();
      vertex.setId((LongWritable) record.get(0));
      String[] edges = record.get(1).toString().split(",");
      for (int i = 0; i < edges.length; i++) {
        long destID = Long.parseLong(edges[i]);
        vertex.addEdge(new LongWritable(destID), NullWritable.get());
      }
      context.addVertexRequest(vertex);
    }
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2) {
      System.out.println("Usage: <input> <output>");
      System.exit(-1);
    }
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(CCVertexReader.class);
    job.setVertexClass(CCVertex.class);
    job.setCombinerClass(MinLongCombiner.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    long startTime = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
  }
}
```

4.4.8.7. Topological sorting

For a directed edge (u,v), all vertex sequences with u coming before v in the ordering are called topological sequences. Topological sorting is an algorithm that is used to calculate the topological sequence of a directed graph.

Procedure:

1. Identify a vertex without incoming edges and generate an output record.
2. Delete the vertex and all its outgoing edges from the graph.
3. Repeat the preceding steps until output records are generated for all the vertices without incoming edges.

Sample code

Sample code for the topological sorting algorithm:

```
import java.io.IOException;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.Combiner;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.BooleanWritable;
import com.aliyun.odps.io.WritableRecord;
public class TopologySort {
  private final static Log LOG = LogFactory.getLog(TopologySort.class);
  public static class TopologySortVertex extends
      Vertex<LongWritable, LongWritable, NullWritable, LongWritable> {
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, NullWritable, LongWritable> context,
        Iterable<LongWritable> messages) throws IOException {
      // in superstep 0, each vertex sends message whose value is 1 to its
      // neighbors
      if (context.getSuperstep() == 0) {
        if (hasEdges()) {
          context.sendMessageToNeighbors(this, new LongWritable(1L));
        }
      } else if (context.getSuperstep() >= 1) {
        // compute each vertex's indegree
        long indegree = getValue().get();
        for (LongWritable msg : messages) {
          indegree += msg.get();
        }
        setValue(new LongWritable(indegree));
        if (indegree == 0) {
          voteToHalt();
          if (hasEdges()) {
            context.sendMessageToNeighbors(this, new LongWritable(-1L));
          }
          context.write(new LongWritable(context.getSuperstep()), getId());
          LOG.info("vertex: " + getId());
        }
        context.aggregate(new LongWritable(indegree));
      }
    }
  }
  public static class TopologySortVertexReader extends
      GraphLoader<LongWritable, LongWritable, NullWritable, LongWritable> {
    @Override
```

```
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, NullWritable, LongWritable> context)
        throws IOException {
      TopologySortVertex vertex = new TopologySortVertex();
      vertex.setId((LongWritable) record.get(0));
      vertex.setValue(new LongWritable(0));
      String[] edges = record.get(1).toString().split(",");
      for (int i = 0; i < edges.length; i++) {
        long edge = Long.parseLong(edges[i]);
        if (edge >= 0) {
          vertex.addEdge(new LongWritable(Long.parseLong(edges[i])),
              NullWritable.get());
        }
      }
      LOG.info(record.toString());
      context.addVertexRequest(vertex);
    }
  }
  public static class LongSumCombiner extends
      Combiner<LongWritable, LongWritable> {
    @Override
    public void combine(LongWritable vertexId, LongWritable combinedMessage,
        LongWritable messageToCombine) throws IOException {
      combinedMessage.set(combinedMessage.get() + messageToCombine.get());
    }
  }
  public static class TopologySortAggregator extends
      Aggregator<BooleanWritable> {
    @SuppressWarnings("rawtypes")
    @Override
    public BooleanWritable createInitialValue(WorkerContext context)
        throws IOException {
      return new BooleanWritable(true);
    }
    @Override
    public void aggregate(BooleanWritable value, Object item)
        throws IOException {
      boolean hasCycle = value.get();
      boolean inDegreeNotZero = ((LongWritable) item).get() == 0 ? false : true;
      value.set(hasCycle && inDegreeNotZero);
    }
    @Override
    public void merge(BooleanWritable value, BooleanWritable partial)
        throws IOException {
      value.set(value.get() && partial.get());
    }
    @SuppressWarnings("rawtypes")
    @Override
    public boolean terminate(WorkerContext context, BooleanWritable value)
        throws IOException {
      if (context.getSuperstep() == 0) {
        // since the initial aggregator value is true, and in superstep we don't
        // do aggregate
```

```
        // do aggregate
        return false;
      }
      return value.get();
    }
  }
  public static void main(String[] args) throws IOException {
    if (args.length != 2) {
      System.out.println("Usage : <inputTable> <outputTable>");
      System.exit(-1);
    }
    // Format of the input table:
    // 0 1, 2
    // 1 3
    // 2 3
    // 3 -1
    // The first column is vertexid, and the second column is destination vertexid of the o
utgoing edge. If the value is -1, the vertex does not have outgoing edges.
    // Format of the output table:
    // 0 0
    // 1 1
    // 1 2
    // 2 3
    // The first column is the supstep value, in which the topological sequence is hidden.
The second column is vertexid.
    // TopologySortAggregator is used to determine whether the graph has loops.
    // If the input graph has a loop and the indegree of all active vertices is not 0, the
iteration ends.
    // You can use records in the input and output tables to determine whether the graph ha
s loops.
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(TopologySortVertexReader.class);
    job.setVertexClass(TopologySortVertex.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    job.setCombinerClass(LongSumCombiner.class);
    job.setAggregatorClass(TopologySortAggregator.class);
    long startTime = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
  }
}
```

4.4.8.8. Linear regression

In statistics, linear regression is a statistical analysis method used to determine the dependency between two or more variables. Linear regression is different from the classification algorithm that predicts discrete values

The regression algorithm can predict continuous values. The linear regression algorithm defines the loss function as the sum of the least square errors of a sample set and solves the weight vector by minimizing the loss function.

A common solution is the gradient descent method, which is implemented in the following steps:

1. Initialize the weight vector to provide the descent speed and iterations or iteration convergence condition.
2. Calculate the least square error for each sample.
3. Calculate the sum of the least square errors and update the weight based on the descent speed.
4. Repeat iterations until convergence occurs.

Sample code

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
/**
 * LineRegression input: y,x1,x2,x3,......
 **/
public class LinearRegression {
  public static class GradientWritable implements Writable {
    Tuple lastTheta;
    Tuple currentTheta;
    Tuple tmpGradient;
    LongWritable count;
    DoubleWritable lost;
    @Override
    public void readFields(DataInput in) throws IOException {
      lastTheta = new Tuple();
      lastTheta.readFields(in);
      currentTheta = new Tuple();
      currentTheta.readFields(in);
      tmpGradient = new Tuple();
      tmpGradient.readFields(in);
      count = new LongWritable();
      count.readFields(in);
      /* update 1: add a variable to store lost at every iteration */
      lost = new DoubleWritable();
      lost.readFields(in);
    }
    @Override
    public void write(DataOutput out) throws IOException {
```

```
      lastTheta.write(out);
      currentTheta.write(out);
      tmpGradient.write(out);
      count.write(out);
      lost.write(out);
    }
  }
  public static class LinearRegressionVertex extends
      Vertex<LongWritable, Tuple, NullWritable, NullWritable> {
    @Override
    public void compute(
        ComputeContext<LongWritable, Tuple, NullWritable, NullWritable> context,
        Iterable<NullWritable> messages) throws IOException {
      context.aggregate(getValue());
    }
  }
  public static class LinearRegressionVertexReader extends
      GraphLoader<LongWritable, Tuple, NullWritable, NullWritable> {
    @Override
    public void load(LongWritable recordNum, WritableRecord record,
        MutationContext<LongWritable, Tuple, NullWritable, NullWritable> context)
        throws IOException {
      LinearRegressionVertex vertex = new LinearRegressionVertex();
      vertex.setId(recordNum);
      vertex.setValue(new Tuple(record.getAll()));
      context.addVertexRequest(vertex);
    }
  }
  public static class LinearRegressionAggregator extends
      Aggregator<GradientWritable> {
    @SuppressWarnings("rawtypes")
    @Override
    public GradientWritable createInitialValue(WorkerContext context)
        throws IOException {
      if (context.getSuperstep() == 0) {
        /* set initial value, all 0 */
        GradientWritable grad = new GradientWritable();
        grad.lastTheta = new Tuple();
        grad.currentTheta = new Tuple();
        grad.tmpGradient = new Tuple();
        grad.count = new LongWritable(1);
        grad.lost = new DoubleWritable(0.0);
        int n = (int) Long.parseLong(context.getConfiguration()
            .get("Dimension"));
        for (int i = 0; i < n; i++) {
          grad.lastTheta.append(new DoubleWritable(0));
          grad.currentTheta.append(new DoubleWritable(0));
          grad.tmpGradient.append(new DoubleWritable(0));
        }
        return grad;
      } else
        return (GradientWritable) context.getLastAggregatedValue(0);
    }
    public static double vecMul(Tuple value, Tuple theta) {
      /* perform this partial computing: y(i)-h0(x(i)) for each sample */
```

```
    /* perform this partial computing: y(i) h0(x(i)) for each sample */
    /* value denote a piece of sample and value(0) is y */
    double sum = 0.0;
    for (int j = 1; j < value.size(); j++)
      sum += Double.parseDouble(value.get(j).toString())
          * Double.parseDouble(theta.get(j).toString());
    Double tmp = Double.parseDouble(theta.get(0).toString()) + sum
        - Double.parseDouble(value.get(0).toString());
    return tmp;
  }
  @Override
  public void aggregate(GradientWritable gradient, Object value)
      throws IOException {
    /*
     * perform on each vertex--each sample i: set theta(j) for each sample i
     * for each dimension
     */
    double tmpVar = vecMul((Tuple) value, gradient.currentTheta);
    /*
     * update 2:local worker aggregate(), perform like merge() below. This
     * means the variable gradient denotes the previous aggregated value
     */
    gradient.tmpGradient.set(0, new DoubleWritable(
        ((DoubleWritable) gradient.tmpGradient.get(0)).get() + tmpVar));
    gradient.lost.set(Math.pow(tmpVar, 2));
    /*
     * calculate (y(i)-hθ(x(i))) x(i)(j) for each sample i for each
     * dimension j
     */
    for (int j = 1; j < gradient.tmpGradient.size(); j++)
      gradient.tmpGradient.set(j, new DoubleWritable(
          ((DoubleWritable) gradient.tmpGradient.get(j)).get() + tmpVar
              * Double.parseDouble(((Tuple) value).get(j).toString())));
  }
  @Override
  public void merge(GradientWritable gradient, GradientWritable partial)
      throws IOException {
    /* perform SumAll on each dimension for all samples. */
    Tuple master = (Tuple) gradient.tmpGradient;
    Tuple part = (Tuple) partial.tmpGradient;
    for (int j = 0; j < gradient.tmpGradient.size(); j++) {
      DoubleWritable s = (DoubleWritable) master.get(j);
      s.set(s.get() + ((DoubleWritable) part.get(j)).get());
    }
    gradient.lost.set(gradient.lost.get() + partial.lost.get());
  }
  @SuppressWarnings("rawtypes")
  @Override
  public boolean terminate(WorkerContext context, GradientWritable gradient)
      throws IOException {
    /*
     * 1. calculate new theta 2. judge the diff between last step and this
     * step, if smaller than the threshold, stop iteration
     */
    gradient.lost = new DoubleWritable(gradient.lost.get()
```

```
        / (2 * context.getTotalNumVertices()));
    /*
     * we can calculate lost in order to make sure the algorithm is running on
     * the right direction (for debug)
     */
    System.out.println(gradient.count + " lost:" + gradient.lost);
    Tuple tmpGradient = gradient.tmpGradient;
    System.out.println("tmpGra" + tmpGradient);
    Tuple lastTheta = gradient.lastTheta;
    Tuple tmpCurrentTheta = new Tuple(gradient.currentTheta.size());
    System.out.println(gradient.count + " terminate_start_last:" + lastTheta);
    double alpha = 0.07; // learning rate
    // alpha =
    // Double.parseDouble(context.getConfiguration().get("Alpha"));
    /* perform theta(j) = theta(j)-alpha*tmpGradient */
    long M = context.getTotalNumVertices();
    /*
     * update 3: add (/M) on the code. The original code forget this step
     */
    for (int j = 0; j < lastTheta.size(); j++) {
      tmpCurrentTheta
          .set(
              j,
              new DoubleWritable(Double.parseDouble(lastTheta.get(j)
                  .toString())
                  - alpha
                  / M
                  * Double.parseDouble(tmpGradient.get(j).toString())));
    }
    System.out.println(gradient.count + " terminate_start_current:"
        + tmpCurrentTheta);
    // judge if convergence is happening.
    double diff = 0.00d;
    for (int j = 0; j < gradient.currentTheta.size(); j++)
      diff += Math.pow(((DoubleWritable) tmpCurrentTheta.get(j)).get()
          - ((DoubleWritable) lastTheta.get(j)).get(), 2);
    if (/*
         * Math.sqrt(diff) < 0.00000000005d ||
         */Long.parseLong(context.getConfiguration().get("Max_Iter_Num")) == gradient.cou
nt
        .get()) {
      context.write(gradient.currentTheta.toArray());
      return true;
    }
    gradient.lastTheta = tmpCurrentTheta;
    gradient.currentTheta = tmpCurrentTheta;
    gradient.count.set(gradient.count.get() + 1);
    int n = (int) Long.parseLong(context.getConfiguration().get("Dimension"));
    /*
     * update 4: Important!!! Remember this step. Graph won't reset the
     * initial value for global variables at the beginning of each iteration
     */
    for (int i = 0; i < n; i++) {
      gradient.tmpGradient.set(i, new DoubleWritable(0));
```

```
      }
      return false;
    }
  }
  public static void main(String[] args) throws IOException {
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(LinearRegressionVertexReader.class);
    job.setRuntimePartitioning(false);
    job.setNumWorkers(3);
    job.setVertexClass(LinearRegressionVertex.class);
    job.setAggregatorClass(LinearRegressionAggregator.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    job.setMaxIteration(Integer.parseInt(args[2])); // Numbers of Iteration
    job.setInt("Max_Iter_Num", Integer.parseInt(args[2]));
    job.setInt("Dimension", Integer.parseInt(args[3])); // Dimension
    job.setFloat("Alpha", Float.parseFloat(args[4])); // Learning rate
    long start = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - start) / 1000.0 + " seconds");
  }
}
```

4.4.8.9. Triangle count

The triangle count algorithm calculates the number of triangles that pass through each vertex in a graph.

The algorithm is implemented in the following steps:

1. Send the ID of each vertex to all its outgoing neighbors.
2. Store information about incoming and outgoing neighbors, and send the information to outgoing neighbors.
3. Calculate the number of endpoint intersections for each edge, calculate the sum, and write the output results to a table.
4. Sum up the output results in the table and divide the sum by 3 to obtain the number of triangles that pass through each vertex.

Sample code

The following code is a sample implementation of the triangle count algorithm:

```
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.Edge;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.LongWritable;
```

```
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
/**
 * Compute the number of triangles passing through each vertex.
 *
 * The algorithm can be computed in three supersteps:
 * I. Each vertex sends a message with its ID to all its outgoing
 * neighbors.
 * II. The incoming neighbors and outgoing neighbors are stored and
 * send to outgoing neighbors.
 * III. For each edge compute the intersection of the sets at destination
 * vertex and sum them, then output to table.
 *
 * The triangle count is the sum of output table and divide by three since
 * each triangle is counted three times.
 *
 **/
public class TriangleCount {
  public static class TCVertex extends
    Vertex<LongWritable, Tuple, NullWritable, Tuple> {
    @Override
    public void setup(
        WorkerContext<LongWritable, Tuple, NullWritable, Tuple> context)
      throws IOException {
      // collect the outgoing neighbors
      Tuple t = new Tuple();
      if (this.hasEdges()) {
        for (Edge<LongWritable, NullWritable> edge : this.getEdges()) {
          t.append(edge.getDestVertexId());
        }
      }
      this.setValue(t);
    }
    @Override
    public void compute(
        ComputeContext<LongWritable, Tuple, NullWritable, Tuple> context,
        Iterable<Tuple> msgs) throws IOException {
      if (context.getSuperstep() == 0L) {
        // sends a message with its ID to all its outgoing neighbors
        Tuple t = new Tuple();
        t.append(getId());
        context.sendMessageToNeighbors(this, t);
      } else if (context.getSuperstep() == 1L) {
        // store the incoming neighbors
        for (Tuple msg : msgs) {
          for (Writable item : msg.getAll()) {
            if (! this.getValue().getAll().contains((LongWritable)item)) {
              this.getValue().append((LongWritable)item);
            }
          }
        }
        // send both incoming and outgoing neighbors to all outgoing neighbors
```

```
        context.sendMessageToNeighbors(this, getValue());
      } else if (context.getSuperstep() == 2L) {
        // count the sum of intersection at each edge
        long count = 0;
        for (Tuple msg : msgs) {
          for (Writable id : msg.getAll()) {
            if (getValue().getAll().contains(id)) {
              count ++;
            }
          }
        }
        // output to table
        context.write(getId(), new LongWritable(count));
        this.voteToHalt();
      }
    }
  }
  public static class TCVertexReader extends
  GraphLoader<LongWritable, Tuple, NullWritable, Tuple> {
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, Tuple, NullWritable, Tuple> context)
    throws IOException {
      TCVertex vertex = new TCVertex();
      vertex.setId((LongWritable) record.get(0));
      String[] edges = record.get(1).toString().split(",");
      for (int i = 0; i < edges.length; i++) {
        try {
          long destID = Long.parseLong(edges[i]);
          vertex.addEdge(new LongWritable(destID), NullWritable.get());
        } catch(NumberFormatException nfe) {
          System.err.println("Ignore " + nfe);
        }
      }
      context.addVertexRequest(vertex);
    }
  }
  public static void main(String[] args) throws IOException {
    if (args.length < 2) {
      System.out.println("Usage: <input> <output>");
      System.exit(-1);
    }
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(TCVertexReader.class);
    job.setVertexClass(TCVertex.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    long startTime = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
        + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
  }
```

```
}
```

4.4.8.10. Vertex table import

This topic provides the sample code that is used to import a vertex table.

Sample code:

```
import java.io.IOException;
import com.aliyun.odps.conf.Configuration;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.VertexResolver;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.VertexChanges;
import com.aliyun.odps.graph.Edge;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.WritableComparable;
import com.aliyun.odps.io.WritableRecord;
/**
 * This example describes how to compile a graph job program to load data of different data
types. It covers how GraphLoader and VertexResolver are used together to build a graph.
 * A MaxCompute Graph job uses MaxCompute tables as the input. For example, a job uses two
tables as the input. One stores information about vertices, and the other stores informatio
n about edges.
 * Format of the table that stores information about vertices:
 * +------------------------+
 * | VertexID | VertexValue |
 * +------------------------+
 * |       id0|            9|
 * +------------------------+
 * |       id1|            7|
 * +------------------------+
 * |       id2|            8|
 * +------------------------+
 *
 * Format of the table that stores information about edges:
 * +-----------------------------------+
 * | VertexID | DestVertexID| EdgeValue|
 * +-----------------------------------+
 * |       id0|          id1|         1|
 * +-----------------------------------+
 * |       id0|          id2|         2|
 * +-----------------------------------+
 * |       id2|          id1|         3|
 * +-----------------------------------+
 *
 * The two tables show that id0 has two outgoing edges that point to id1 and id2. id2 has a
n outgoing edge that points to id1, and id1 has no outgoing edges.
 *
```

```
 * For data of this type, in GraphLoader::load(LongWritable, Record, MutationContext),
 * MutationContext#addVertexRequest(Vertex) can be used to add vertices to the graph.
 * link MutationContext#addEdgeRequest(WritableComparable, Edge) can be used to add edges t
o the graph. In
 * link VertexResolver#resolve(WritableComparable, Vertex, VertexChanges, boolean),
 * vertices and edges added by using the load() method are combined to a vertex object. Thi
s object is used as the returned value and added to the graph for computing.
 *
 **/
public class VertexInputFormat {
  private final static String EDGE_TABLE = "edge.table";
  /**
   * Resolve records to vertices and edges. Each record indicates a vertex or edge based on
its source.
   *
   * The following process is similar to the map stage of com.aliyun.odps.mapreduce.Mapper.
Enter a record to generate key-value pairs. The keys are vertex IDs,
   * and the values are vertices or edges that are written based on context. These key-valu
e pairs are aggregated by using LoadingVertexResolver based on the vertex IDs.
   *
   * Note: The vertices or edges added here are requests sent based on the record content,
and are not used in computing. Only the vertices or edges
   * added by using VertexResolver are used for computing.
   **/
  public static class VertexInputLoader extends
      GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
    private boolean isEdgeData;
    /**
     * Configure VertexInputLoader.
     *
     * @param conf
     *          Indicates the configured parameters of a job. These parameters are configur
ed in the console or in the MAIN function of GraphJob.
     * @param workerId
     *          Indicates the serial number of the running worker. The number starts from 0
and can be used to build a unique vertex ID.
     * @param inputTableInfo
     *          Indicates the information about the input table that is loaded to the runni
ng worker. The information can be used to determine the data type of the input data (the fo
rmat of the record).
     **/
    @Override
    public void setup(Configuration conf, int workerId, TableInfo inputTableInfo) {
      isEdgeData = conf.get(EDGE_TABLE).equals(inputTableInfo.getTableName());
    }
    /**
     * Resolve the record to edges based on the record content and send a request to add th
em to the graph.
     *
     * @param recordNum
     *          Indicates the serial number of the record. The number starts from 1 and is
separately counted in each worker.
     * @param record
     *          Indicates the records in the input table. The table contains three columns,
which indicate the source vertex, destination vertex, and edge weight
```

```
which indicate the source vertex, destination vertex, and edge weight.
     * @param context
     *          Indicates the context. The context is used when you send a request to add r
esolved edges to the graph.
     **/
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context)
        throws IOException {
      if (isEdgeData) {
        /**
         * Data comes from the table that stores information about edges.
         *
         * 1. The first column indicates the IDs of source vertices.
         **/
        LongWritable sourceVertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the IDs of destination vertices.
         **/
        LongWritable destinationVertexID = (LongWritable) record.get(1);
        /**
         * 3. The third column indicates edge weights.
         **/
        LongWritable edgeValue = (LongWritable) record.get(2);
        /**
         * 4. Create an edge based on a destination vertex ID and an edge weight.
         **/
        Edge<LongWritable, LongWritable> edge = new Edge<LongWritable, LongWritable>(
            destinationVertexID, edgeValue);
        /**
         * 5. Send a request to add the edge to a source vertex.
         **/
        context.addEdgeRequest(sourceVertexID, edge);
        /**
         * 6. If each record indicates a bidirectional edge, repeat Step 4 and Step 5.
          * Edge<LongWritable, LongWritable> edge2 = new
         * Edge<LongWritable, LongWritable>( sourceVertexID, edgeValue);
         * context.addEdgeRequest(destinationVertexID, edge2);
         **/
      } else {
        /**
         * Data comes from the table that stores information about vertices.
         *
         * 1. The first column indicates the IDs of vertices.
         **/
        LongWritable vertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the values of vertices.
         **/
        LongWritable vertexValue = (LongWritable) record.get(1);
        /**
         * 3. Create a vertex based on an ID and a value.
         **/
```

```
        MyVertex vertex = new MyVertex();
        /**
         * 4. Initialize the vertex.
         **/
        vertex.setId(vertexID);
        vertex.setValue(vertexValue);
        /**
         * 5. Send a request to add the vertex.
         **/
        context.addVertexRequest(vertex);
      }
    }
  }
  /**
   * Summarize key-value pairs generated by using GraphLoader::load(LongWritable, Record, M
utationContext). This process is similar to
   * the reduce stage of com.aliyun.odps.mapreduce.Reducer. For a unique vertex ID, all act
ions, such as
   * adding or removing vertices or edges for the ID, are stored in VertexChanges.
   *
   * Note: Not only conflicting vertices or edges added by using the load() method are call
ed. A conflict occurs when multiple same vertex objects or duplicate edges are added.
   * All the IDs that are requested to be generated by using the load() method are called.
   **/
  public static class LoadingResolver extends
      VertexResolver<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Process a request for adding or removing vertices or edges for an ID.
     *
     * VertexChanges has four APIs, which correspond to the four APIs of MutationContext:
     * VertexChanges::getAddedVertexList() corresponds to
     * MutationContext::addVertexRequest(Vertex).
     * In the load() method, if vertex objects with the same ID are requested to be added,
they are collected to the returned list.
     * VertexChanges::getAddedEdgeList() corresponds to
     * MutationContext::addEdgeRequest(WritableComparable, Edge).
     * If edge objects with the same source vertex ID are requested to be added, they are c
ollected to the returned list.
     * VertexChanges::getRemovedVertexCount() corresponds to
     * MutationContext::removeVertexRequest(WritableComparable).
     * If vertices with the same ID are requested to be removed, the total number of remova
l requests is returned.
     * VertexChanges#getRemovedEdgeList() corresponds to
     * MutationContext#removeEdgeRequest(WritableComparable, WritableComparable).
     * If edge objects with the same source vertex ID are requested to be removed, they are
collected to the returned list.
     *
     * You can process the changes in the ID and state whether the ID is used in computing
in the returned value. If the returned vertex is not null,
     * the ID is used in subsequent computing. If the returned vertex is null, the ID is no
t used in subsequent computing.
     *
     * @param vertexId
     *          Indicates the ID of the vertex that is requested to be added or the source
```

```
vertex ID of the edge that is requested to be added.
     * @param vertex
     *          Indicates an existing vertex object. The value of this parameter is always
null in the data loading phase.
     * @param vertexChanges
     *          Indicates a collection of vertices or edges that are requested to be added
or removed for the ID.
     * @param hasMessages
     *          Indicates whether messages are sent to the ID. The value of this parameter
is always false in the data loading phase.
     **/
    @Override
    public Vertex<LongWritable, LongWritable, LongWritable, LongWritable> resolve(
        LongWritable vertexId,
        Vertex<LongWritable, LongWritable, LongWritable, LongWritable> vertex,
        VertexChanges<LongWritable, LongWritable, LongWritable, LongWritable> vertexChanges
,
        boolean hasMessages) throws IOException {
      /**
       * 1. Obtain the vertex object for computing.
       **/
      MyVertex computeVertex = null;
      if (vertexChanges.getAddedVertexList() == null
          || vertexChanges.getAddedVertexList().isEmpty()) {
        computeVertex = new MyVertex();
        computeVertex.setId(vertexId);
      } else {
        /**
         * Each record indicates a unique vertex in the table that stores information about
vertices.
         **/
        computeVertex = (MyVertex) vertexChanges.getAddedVertexList().get(0);
      }
      /**
       * 2. Add the edge, which is requested to be added to the vertex, to the vertex objec
t. If the data is a possible duplicate, deduplicate it based on algorithm requirements.
       **/
      if (vertexChanges.getAddedEdgeList() != null) {
        for (Edge<LongWritable, LongWritable> edge : vertexChanges
            .getAddedEdgeList()) {
          computeVertex.addEdge(edge.getDestVertexId(), edge.getValue());
        }
      }
      /**
       * 3. Return the vertex object and add it to the final graph for computing.
       **/
      return computeVertex;
    }
  }
  /**
   * Determine the actions of the vertex that is used in computing.
   *
   **/
  public static class MyVertex extends
```

```
      Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Write the edge of the vertex to the result table based on the format of the input ta
ble. Make sure that the formats and data of the input and output tables are the same.
     *
     * @param context
     *          Indicates the runtime context.
     * @param messages
     *          Indicates the input messages.
     **/
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context,
        Iterable<LongWritable> messages) throws IOException {
      /**
       * Write the ID and value of the vertex to the result table that stores information a
bout vertices.
       **/
      context.write("vertex", getId(), getValue());
      /**
       * Write the edge of the vertex to the result table that stores information about edg
es.
       **/
      if (hasEdges()) {
        for (Edge<LongWritable, LongWritable> edge : getEdges()) {
          context.write("edge", getId(), edge.getDestVertexId(),
              edge.getValue());
        }
      }
      /**
       * Perform only one iteration.
       **/
      voteToHalt();
    }
  }
  /**
   * @param args
   * @throws IOException
   */
  public static void main(String[] args) throws IOException {
    if (args.length < 4) {
      throw new IOException(
          "Usage: VertexInputFormat <vertex input> <edge input> <vertex output> <edge outpu
t>");
    }
    /**
     * Use GraphJob to configure a Graph job.
     */
    GraphJob job = new GraphJob();
    /**
     * 1. Specify input graph data and the table that stores information about edges.
     */
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addInput(TableInfo.builder().tableName(args[1]).build());
```

```
    job.set(EDGE_TABLE, args[1]);
    /**
     * 2. Specify the data loading mode and resolve the records to edges. This process is s
imilar to the map stage. The generated key is the vertex ID, and the generated value is the
edge.
     */
    job.setGraphLoaderClass(VertexInputLoader.class);
    /**
     * 3. Specify the data loading phase to generate the vertex that is used in computing.
This process is similar to the reduce stage. In the reduce stage, edges that are generated
in the map stage are combined into a vertex.
     */
    job.setLoadingVertexResolverClass(LoadingResolver.class);
    /**
     * 4. Specify the actions of the vertex that is used in computing. The vertex.compute()
method is used for each iteration.
     */
    job.setVertexClass(MyVertex.class);
    /**
     * 5. Specify the result table of the Graph job and write the computing results to the
result table.
     */
    job.addOutput(TableInfo.builder().tableName(args[2]).label("vertex").build());
    job.addOutput(TableInfo.builder().tableName(args[3]).label("edge").build());
    /**
     * 6. Submit the job for execution.
     */
    job.run();
  }
}
```

4.4.8.11. Edge table import

This topic provides the sample code that is used to import an edge table.

Sample code:

```
import java.io.IOException;
import com.aliyun.odps.conf.Configuration;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.VertexResolver;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.VertexChanges;
import com.aliyun.odps.graph.Edge;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.WritableComparable;
import com.aliyun.odps.io.WritableRecord;
/**
 * This example describes how to compile a Graph job program to load data of different data
```

```
 * This example describes how to compile a Graph job program to load data of different data
types. It covers how GraphLoader
 * and VertexResolver are used together to build a graph.
 *
 * A MaxCompute Graph job uses MaxCompute tables as the input. For example, a job uses two
tables as the input. One stores information about vertices, and the other stores informatio
n about edges.
 * Format of the table that stores information about vertices:
 * +------------------------+
 * | VertexID | VertexValue |
 * +------------------------+
 * |       id0|            9|
 * +------------------------+
 * |       id1|            7|
 * +------------------------+
 * |       id2|            8|
 * +------------------------+
 *
 * Format of the table that stores information about edges:
 * +-----------------------------------+
 * | VertexID | DestVertexID| EdgeValue|
 * +-----------------------------------+
 * |       id0|          id1|         1|
 * +-----------------------------------+
 * |       id0|          id2|         2|
 * +-----------------------------------+
 * |       id2|          id1|         3|
 * +-----------------------------------+
 *
 * The two tables show that id0 has two outgoing edges that point to id1 and id2. id2 has o
ne outgoing edge that points to id1, and id1 has no outgoing edges.
 *
 * For data of this type, in GraphLoader::load(LongWritable, Record, MutationContext), Muta
tionContext#addVertexRequest(Vertex) can be used to add vertices to the graph.
 * link MutationContext#addEdgeRequest(WritableComparable, Edge) can be used to add edges t
o the graph. In
 * link VertexResolver#resolve(WritableComparable, Vertex, VertexChanges, boolean),
 * vertices and edges added by using the load() method are combined to a vertex object. Thi
s object is used as the return value and added to the graph for computing.
 *
 **/
public class VertexInputFormat {
  private final static String EDGE_TABLE = "edge.table";
  /**
   * Resolve records to vertices and edges. Each record indicates a vertex or edge based on
its source.
   * <p>
   * The following process is similar to the map stage of com.aliyun.odps.mapreduce.Mapper.
   * Enter a record to generate key-value pairs.
   * The keys are vertex IDs, and the values are vertices or edges that are written based o
n context. These key-value pairs are aggregated by using LoadingVertexResolver based on the
vertex IDs.
   *
   * Note: The vertices or edges added here are requests sent based on the record content,
and are not used in computing.
```

```
   * Only the vertices or edges added by using VertexResolver are used for computing.
   **/
  public static class VertexInputLoader extends
      GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
    private boolean isEdgeData;
    /**
     * Configure VertexInputLoader.
     *
     * @param conf
     *          Indicates the configured parameters of a job. These parameters are configur
ed in the MAIN function of GraphJob or by running the SET command on the client.
     * @param workerId
     *          Indicates the serial number of the running worker. The number starts from 0
and can be used to build a unique vertex ID.
     * @param inputTableInfo
     *          Indicates the information about the input table that is loaded to the runni
ng worker. The information can be used to determine the data type of the input data (the fo
rmat of the record).
     **/
    @Override
    public void setup(Configuration conf, int workerId, TableInfo inputTableInfo) {
      isEdgeData = conf.get(EDGE_TABLE).equals(inputTableInfo.getTableName());
    }
    /**
     * Resolve the record to edges based on the record content and send a request to add th
em to the graph.
     *
     * @param recordNum
     *          Indicates the serial number of the record. The number starts from 1 and is
separately counted in each worker.
     * @param record
     *          Indicates the records in the input table. The table contains three columns,
which indicate the source vertex, destination vertex, and edge weight.
     * @param context
     *          Indicates the context. The context is used when you send a request to add r
esolved edges to the graph.
     **/
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context)
        throws IOException {
      if (isEdgeData) {
        /**
         * Data comes from the table that stores information about edges.
         *
         * 1. The first column indicates the IDs of source vertices.
         **/
        LongWritable sourceVertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the IDs of destination vertices.
         **/
        LongWritable destinationVertexID = (LongWritable) record.get(1);
```

```
        /**
         * 3. The third column indicates edge weights.
         **/
        LongWritable edgeValue = (LongWritable) record.get(2);
        /**
         * 4. Create an edge based on a destination vertex ID and an edge weight.
         **/
        Edge<LongWritable, LongWritable> edge = new Edge<LongWritable, LongWritable>(
            destinationVertexID, edgeValue);
        /**
         * 5. Send a request to add the edge to a source vertex.
         **/
        context.addEdgeRequest(sourceVertexID, edge);
        /**
         * 6. If each record indicates a bidirectional edge, repeat Step 4 and Step 5.
          * Edge<LongWritable, LongWritable> edge2 = new
         * Edge<LongWritable, LongWritable>( sourceVertexID, edgeValue);
         * context.addEdgeRequest(destinationVertexID, edge2);
         **/
      } else {
        /**
         * Data comes from the table that stores information about vertices.
         *
         * 1. The first column indicates the IDs of vertices.
         **/
        LongWritable vertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the values of vertices.
         **/
        LongWritable vertexValue = (LongWritable) record.get(1);
        /**
         * 3. Create a vertex based on an ID and a value.
         **/
        MyVertex vertex = new MyVertex();
        /**
         * 4. Initialize the vertex.
         **/
        vertex.setId(vertexID);
        vertex.setValue(vertexValue);
        /**
         * 5. Send a request to add the vertex.
         **/
        context.addVertexRequest(vertex);
      }
    }
  }
  /**
   * Summarize key-value pairs generated by using GraphLoader::load(LongWritable, Record, M
utationContext).
   * This process is similar to the reduce stage of com.aliyun.odps.mapreduce.Reducer. For
a unique vertex ID, all actions, such as
   * adding or removing vertices or edges for the ID, are stored in VertexChanges.
   *
   * Note: Not only conflicting vertices or edges added by using the load() method are call
```

```
ed. A conflict occurs when multiple same vertex objects or duplicate edges are added.
   * All the IDs that are requested to be generated by using the load() method are called.
   **/
  public static class LoadingResolver extends
      VertexResolver<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Process a request to add or remove vertices or edges for an ID.
     *
     * VertexChanges has four APIs, which correspond to the four APIs of MutationContext:
     * VertexChanges::getAddedVertexList() corresponds to
     * MutationContext::addVertexRequest(Vertex).
     * In the load() method, if a request is sent to add vertex objects with the same ID, t
he objects are collected to the returned list.
     * VertexChanges::getAddedEdgeList() corresponds to
     * MutationContext::addEdgeRequest(WritableComparable, Edge).
     * If a request is sent to add edge objects with the same source vertex ID, the objects
are collected to the returned list.
     * VertexChanges::getRemovedVertexCount() corresponds to
     * MutationContext::removeVertexRequest(WritableComparable).
     * If a request is sent to remove vertices with the same ID, the number of total remova
l requests is returned.
     * VertexChanges#getRemovedEdgeList() corresponds to
     * MutationContext#removeEdgeRequest(WritableComparable, WritableComparable).
     * If a request is sent to remove edge objects with the same source vertex ID, the obje
cts are collected to the returned list.
     *
     * You can process the changes in the ID and state whether the ID is used in computing
in the return value.
     * If the returned vertex is not null, the ID is used in subsequent computing. If the r
eturned vertex is null, the ID is not used in subsequent computing.
     *
     * @param vertexId
     *          Indicates the ID of the vertex that is requested to be added or the source
vertex ID of the edge that is requested to be added.
     * @param vertex
     *          Indicates an existing vertex object. The value of this parameter is always
null in the data loading phase.
     * @param vertexChanges
     *          Indicates a collection of vertices or edges that are requested to be added
or removed for the ID.
     * @param hasMessages
     *          Indicates whether messages are sent to the ID. The value of this parameter
is always false in the data loading phase.
     **/
    @Override
    public Vertex<LongWritable, LongWritable, LongWritable, LongWritable> resolve(
        LongWritable vertexId,
        Vertex<LongWritable, LongWritable, LongWritable, LongWritable> vertex,
        VertexChanges<LongWritable, LongWritable, LongWritable, LongWritable> vertexChanges
,
        boolean hasMessages) throws IOException {
      /**
       * 1. Obtain the vertex object for computing.
       **/
```

```
      MyVertex computeVertex = null;
      if (vertexChanges.getAddedVertexList() == null
          || vertexChanges.getAddedVertexList().isEmpty()) {
        computeVertex = new MyVertex();
        computeVertex.setId(vertexId);
      } else {
        /**
         * Each record indicates a unique vertex in the table that stores information about
vertices.
         **/
        computeVertex = (MyVertex) vertexChanges.getAddedVertexList().get(0);
      }
      /**
       * 2. Add the edge, which is requested to be added to the vertex, to the vertex objec
t. If the data is a possible duplicate, deduplicate it based on algorithm requirements.
       **/
      if (vertexChanges.getAddedEdgeList() != null) {
        for (Edge<LongWritable, LongWritable> edge : vertexChanges
            .getAddedEdgeList()) {
          computeVertex.addEdge(edge.getDestVertexId(), edge.getValue());
        }
      }
      /**
       * 3. Return the vertex object and add it to the final graph for computing.
       **/
      return computeVertex;
    }
  }
  /**
   * Determine the actions of the vertex that is used in computing.
   *
   **/
  public static class MyVertex extends
      Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Write the edge of the vertex to the result table based on the format of the input ta
ble. Make sure that the formats and data of the input and output tables are the same.
     *
     * @param context
     *          Indicates the runtime context.
     * @param messages
     *          Indicates the input messages.
     **/
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context,
        Iterable<LongWritable> messages) throws IOException {
      /**
       * Write the ID and value of the vertex to the result table that stores information a
bout vertices.
       **/
      context.write("vertex", getId(), getValue());
      /**
       * Write the edge of the vertex to the result table that stores information about edg
es.
```

```
       **/
      if (hasEdges()) {
        for (Edge<LongWritable, LongWritable> edge : getEdges()) {
          context.write("edge", getId(), edge.getDestVertexId(),
              edge.getValue());
        }
      }
      /**
       * Perform only one iteration.
       **/
      voteToHalt();
    }
  }
  /**
   * @param args
   * @throws IOException
   */
  public static void main(String[] args) throws IOException {
    if (args.length < 4) {
      throw new IOException(
          "Usage: VertexInputFormat <vertex input> <edge input> <vertex output> <edge outpu
t>");
    }
    /**
     * Use GraphJob to configure a Graph job.
     */
    GraphJob job = new GraphJob();
    /**
     * 1. Specify input graph data and the table that stores information about edges.
     */
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addInput(TableInfo.builder().tableName(args[1]).build());
    job.set(EDGE_TABLE, args[1]);
    /**
     * 2. Specify the data loading mode and resolve the records to edges. This process is s
imilar to the map stage. The generated key is the vertex ID, and the generated value is the
edge.
     */
    job.setGraphLoaderClass(VertexInputLoader.class);
    /**
     * 3. Specify the data loading phase to generate the vertex that is used in computing.
This process is similar to the reduce stage. In the reduce stage, edges that are generated
in the map stage are combined into a vertex.
     */
    job.setLoadingVertexResolverClass(LoadingResolver.class);
    /**
     * 4. Specify the actions of the vertex that is used in computing. The vertex.compute()
method is used for each iteration.
     */
    job.setVertexClass(MyVertex.class);
    /**
     * 5. Specify the result table of the Graph job and write the computing results to the
result table.
     */
```

```
        job.addOutput(TableInfo.builder().tableName(args[2]).label("vertex").build());
        job.addOutput(TableInfo.builder().tableName(args[3]).label("edge").build());
        /**
         * 6. Submit the job for execution.
         */
        job.run();
    }
}
```

4.5. Spark

4.5.1. Overview

Spark on MaxCompute is a computing service that is provided by MaxCompute and compatible with open source Spark. This service provides a Spark computing framework based on unified computing resource and dataset permission systems. The service allows you to use your preferred development method to submit and run Spark jobs. Spark on MaxCompute can fulfill diverse data processing and analysis requirements.

Limits

- You can use Spark on MaxCompute to perform the following operations:
  - Perform offline computing by using Spark components, such as GraphX, MLlib, Resilient Distributed Dataset (RDD), Spark SQL, and PySpark.
  - Read data from and write data to MaxCompute tables.
  - Reference files in MaxCompute.
  - Read data from and write data to services that are deployed in virtual private clouds (VPCs), such as ApsaraDB RDS, ApsaraDB for Redis, ApsaraDB for HBase, and services that are deployed on Elastic Compute Service (ECS) instances.
  - Read and write Object Storage Service (OSS) unstructured storage data.
- You cannot use Spark on MaxCompute to perform the following operations:
  - Run interactive jobs, such as Spark-Shell, Spark-SQL-Shell, and PySpark-Shell jobs.
  - Access MaxCompute external tables, built-in functions, and user-defined functions (UDFs).
- Use checkpoints.

Benefits

- Supports different versions of native Spark jobs.

  MaxCompute supports native community Spark and is fully compatible with the APIs of all native Spark versions. Different versions of Spark can run in MaxCompute at the same time. Spark on MaxCompute provides native Spark web UIs.
- Runs based on centralized computing resources.

  Similar to MaxCompute SQL jobs and MapReduce jobs, Spark on MaxCompute runs based on the centralized computing resources that are purchased for MaxCompute projects.

- Supports centralized data and permission management.

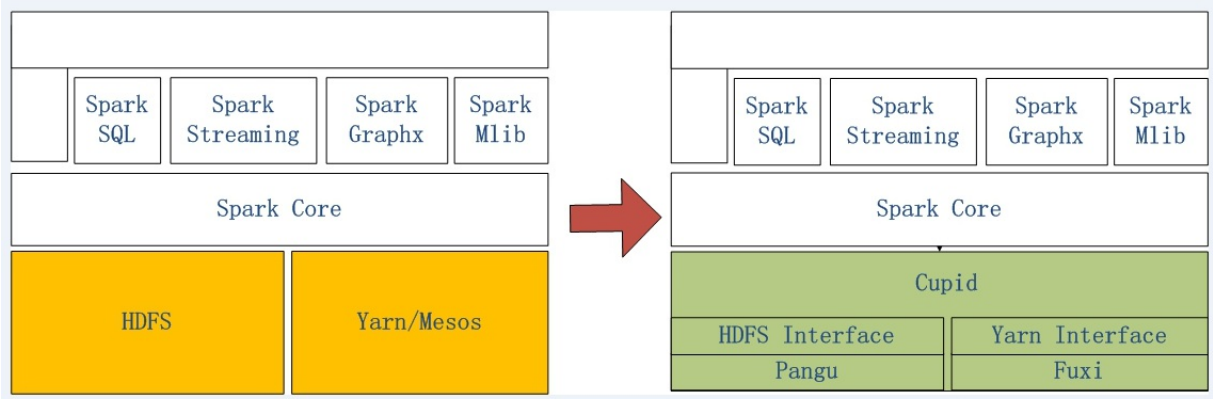
  Spark on MaxCompute complies with the permissions that are configured for MaxCompute projects. This allows you to query data without the need to modify permissions on your MaxCompute projects.

- Provides the same user experience as open source systems.

  Spark on MaxCompute provides the same user experience as open source systems, such as open source application UIs and online interactions. Spark on MaxCompute supports native, open source, and real-time UIs that can be used to debug open source applications. Spark on MaxCompute also allows you to query historical logs. For some open source applications, Spark on MaxCompute can run real-time interactions in the backend. This implements an interactive experience.

Architecture

Spark on MaxCompute is an Alibaba Cloud solution that allows native Spark to run in MaxCompute.



The left part of the preceding figure shows the architecture of native Spark. The right part shows the architecture of Spark on MaxCompute, which runs on the Cupid platform developed by Alibaba Cloud. The Cupid platform is fully compatible with the computing framework supported by open source YARN.

4.5.2. Set up a Spark on MaxCompute development environment

This topic describes how to set up a Spark on MaxCompute development environment.

Prerequisites

Before you set up a Spark on MaxCompute development environment, make sure that the following software is installed in your operating system:

- JDK 1.8

  The following sample command shows how to install JDK 1.8 in a Linux operating system. The actual JDK package name prevails. You can run the `yum -y list java*` command to obtain the JDK package name.

  ```
  yum install -y java-1.8.0-openjdk-devel.x86_64
  ```

- Python 2.7

  The following sample commands show how to install Python 2.7 in a Linux operating system. The actual Python package name prevails.

```
# Obtain the Python package.
wget https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz
# Decompress the Python package.
tar -zxvf Python-2.7.10.tgz
# Switch to the path to which the Python package is decompressed and specify the installa
tion path.
cd Python-2.7.10
./configure --prefix=/usr/local/python2
# Compile and install the Python package.
make
make install
```

- Maven

  The following sample commands show how to install Maven in a Linux operating system. The actual path of the Maven package prevails.

  ```
  # Obtain the Maven package.
  wget https://dlcdn.apache.org/maven/maven-3/3.8.4/binaries/apache-maven-3.8.4-bin.tar.gz
  # Decompress the Maven package.
  tar -zxvf apache-maven-3.8.4-bin.tar.gz
  ```

- Git

  The following sample commands show how to install Git in a Linux operating system.

  ```
  # Obtain the Git package.
  wget https://github.com/git/git/archive/v2.17.0.tar.gz
  # Decompress the Git package.
  tar -zxvf v2.17.0.tar.gz
  # Install the dependencies that are required for compiling the source code of Git.
  yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel gcc perl-ExtUti
  ls-MakeMaker
  # Switch to the path to which the Git package is decompressed.
  cd git-2.17.0
  # Compile the source code of Git.
  make prefix=/usr/local/git all
  # Install Git in the /usr/local/git path.
  make prefix=/usr/local/git install
  ```

Download the Spark on MaxCompute client package and upload the package to your operating system

The Spark on MaxCompute client package is released with the MaxCompute authorization feature. This allows Spark on MaxCompute to serve as a client that submits jobs to your MaxCompute project by using the spark-submit script. MaxCompute provides release packages for Spark 1.x, Spark 2.x, and Spark 3.x. You can download these packages from the following links:

- Spark-1.6.3: used to develop Spark 1.x applications.
- Spark-2.3.0: used to develop Spark 2.x applications.
- Spark-2.4.5: used to develop Spark 2.x applications. For more information about the notes on using Spark-2.4.5, see Notes on using Spark 2.4.5.
- Spark-3.1.1: used to develop Spark 3.x applications.

Upload the Spark on MaxCompute client package to your Linux operating system and decompress the package. You can go to the path in which the Spark on MaxCompute client package is located and run the following command to decompress the package:

```
tar -xzvf spark-2.3.0-odps0.33.0.tar.gz
```

Configure environment variables

You must configure the following environment variables in the CLI of your operating system. The following content describes how to configure environment variables in a Linux operating system and provides related information.

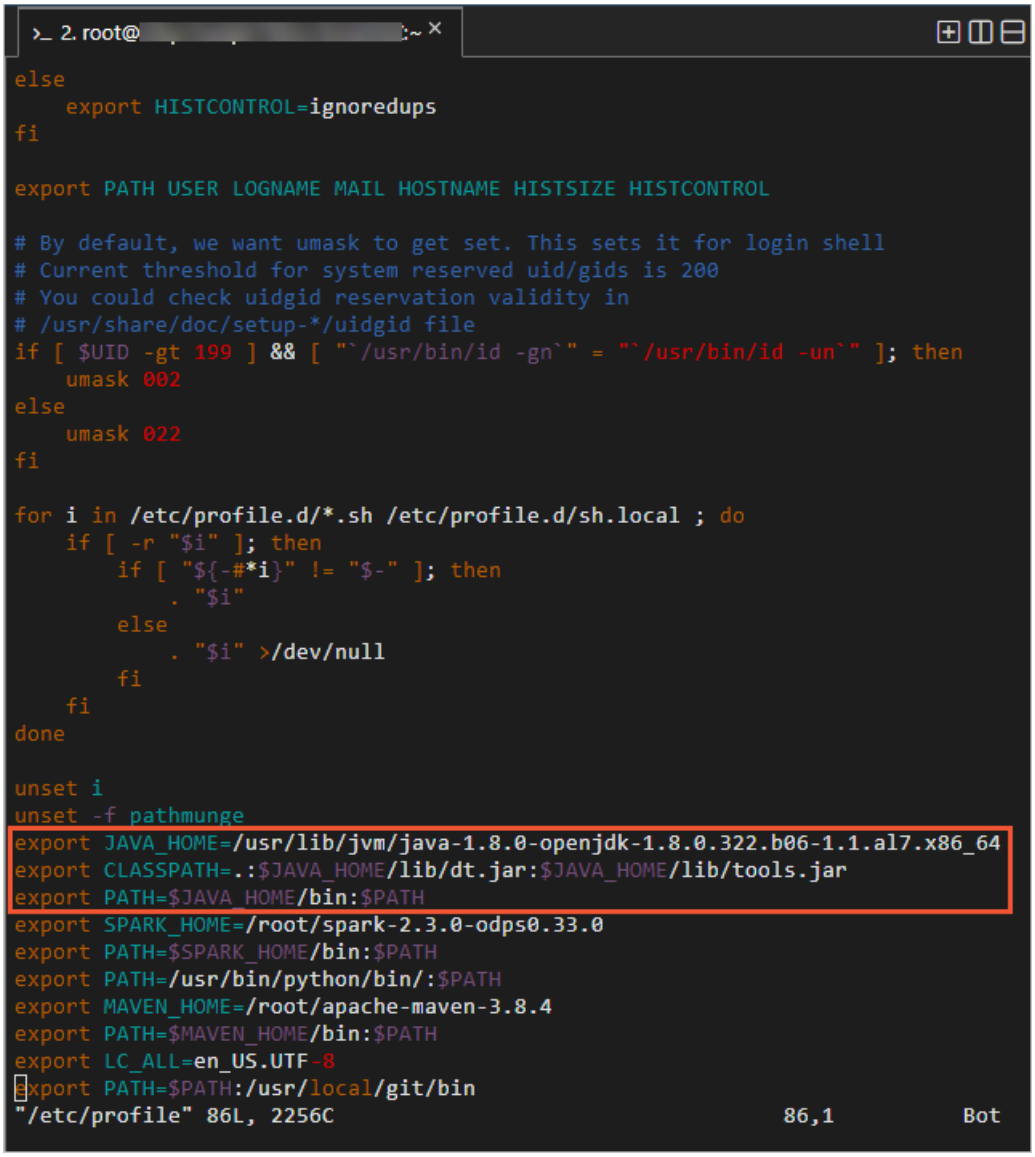
- Configure Java environment variables.
  - Obtain the Java installation path. Sample command:

```
# If you use yum to install Java, Java is installed in the /usr path by default. You ca
n run the following command to query the Java installation path. If you use a custom in
stallation path, the actual path prevails.
whereis java
ls -lrt /usr/bin/java
ls -lrt /etc/alternatives/java
# The following information is returned. /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-
1.1.al7.x86_64 is the Java installation path.
/etc/alternatives/java -> /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-1.1.al7.x86_64/
jre/bin/jav
```

```
[root@ ~]# whereis java
java: /usr/bin/java /usr/lib/java /etc/java /usr/share/java /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.3
22.b06-1.1.al7.x86_64/bin/java /usr/share/man/man1/java.1.gz
[root@ ~]# ls -lrt /usr/bin/java
lrwxrwxrwx 1 root root 22 Feb 10 16:49 /usr/bin/java -> /etc/alternatives/java
[root@ ~]# ls -lrt /etc/alternatives/java
lrwxrwxrwx 1 root root 73 Feb 10 16:49 /etc/alternatives/java -> /usr/lib/jvm/java-1.8.0-openjdk-1.
8.0.322.b06-1.1.al7.x86_64/jre/bin/java
```

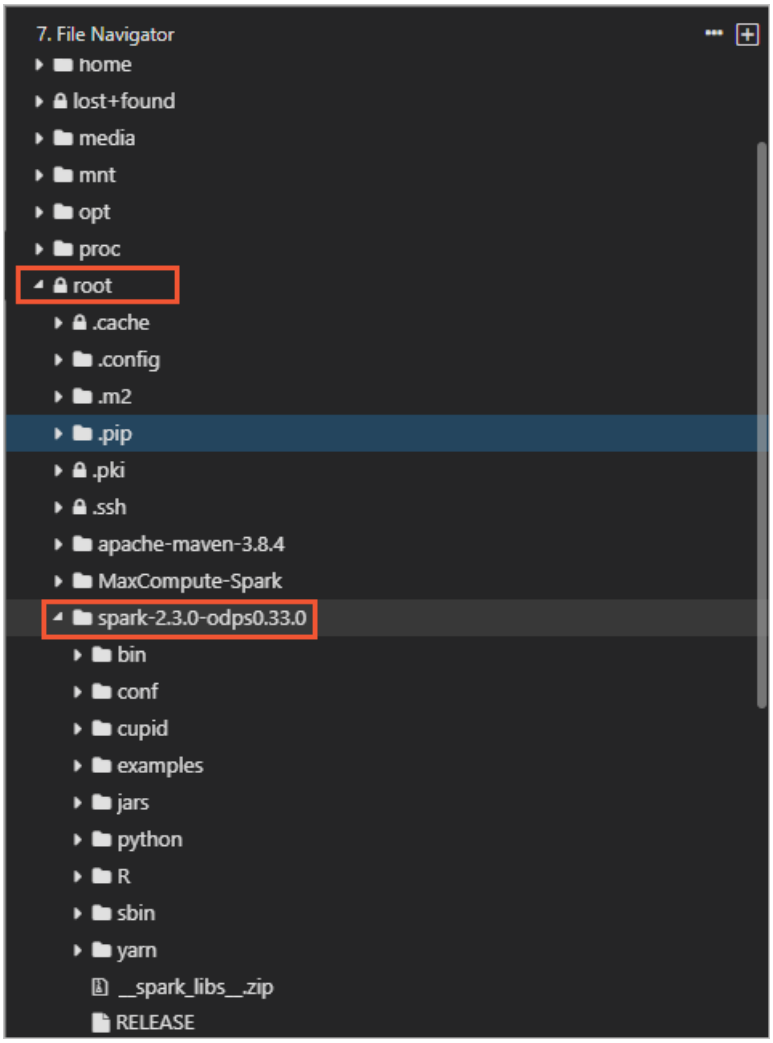
  - Edit Java environment variables. Sample commands:

```
# Edit the configuration file for environment variables.
vim /etc/profile
# Press i to enter the edit mode and add environment variables to the end of the config
uration file.
# Set JAVA_HOME to the actual Java installation path.
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-1.1.al7.x86_64
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$JAVA_HOME/bin:$PATH
# Press the Esc key to exit the edit mode and enter :wq to close the configuration file
.
# Run the following command to make the modification take effect:
source /etc/profile
# Check whether the Java environment variables are successfully configured.
java -version
# Sample output:
openjdk version "1.8.0_322"
OpenJDK Runtime Environment (build 1.8.0_322-b06)
OpenJDK 64-Bit Server VM (build 25.322-b06, mixed mode)
```



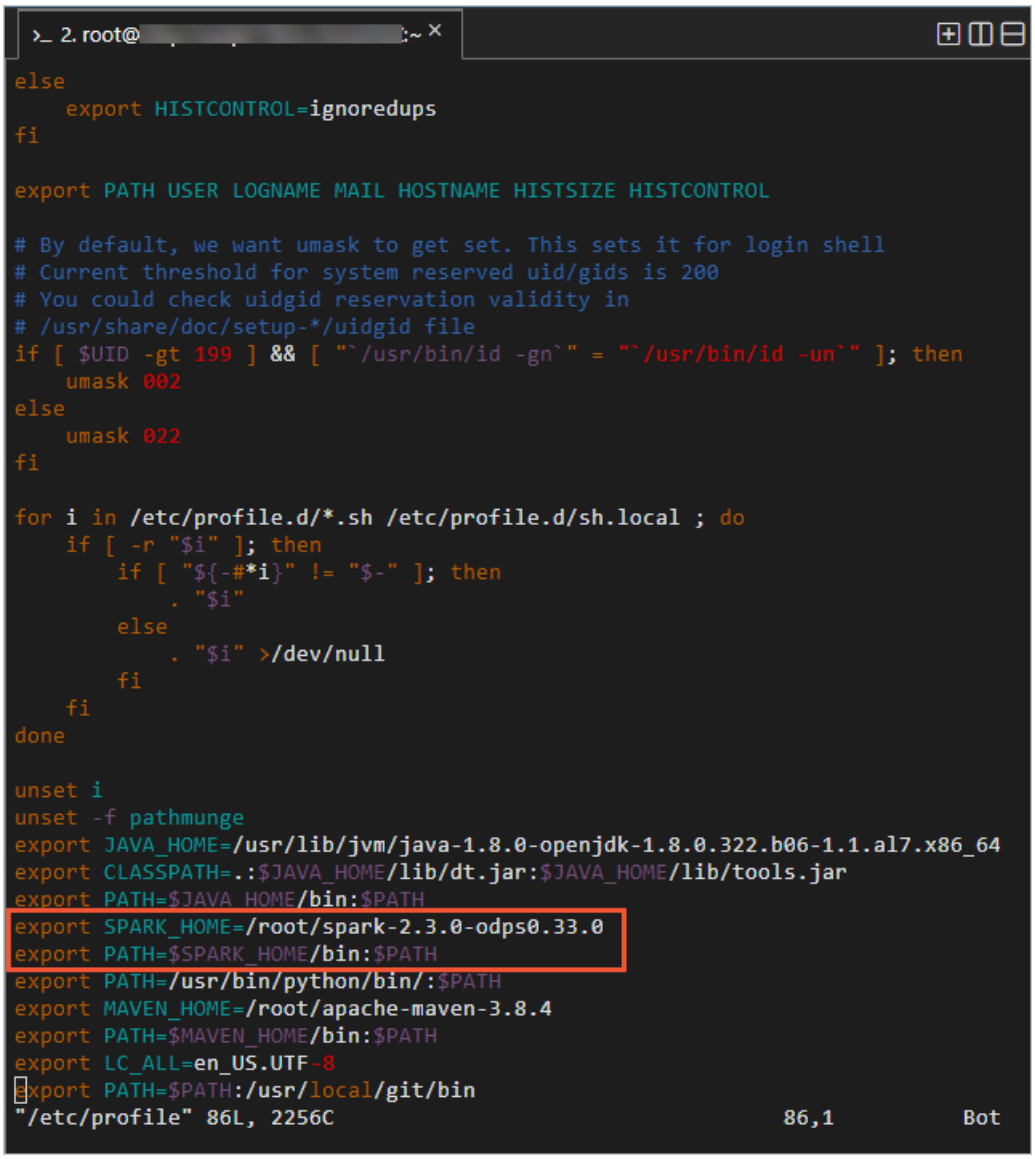
- Configure Spark environment variables.

- Obtain the path to which the Spark on MaxCompute package is decompressed. The following figure shows that the path is `/root/spark-2.3.0-odps0.33.0`. The actual decompression path and package name prevail.



- Edit Spark environment variables. Sample commands:

```
# Edit the configuration file for environment variables.
vim /etc/profile
# Press i to enter the edit mode and add environment variables to the end of the config
uration file.
# Replace SPARK_HOME with the actual path to which the Spark on MaxCompute client packa
ge is decompressed.
export SPARK_HOME=/root/spark-2.3.0-odps0.33.0
export PATH=$SPARK_HOME/bin:$PATH
# Press the Esc key to exit the edit mode and enter :wq to close the configuration file
.
# Run the following command to make the modification take effect:
source /etc/profile
```



- Configure Python environment variables.

  If you use PySpark, you must configure Python environment variables.

- Obtain the Python installation path. Sample command:

```
[root@ ~]# whereis python
python: /usr/bin/python2.7 /usr/bin/python3.6m /usr/bin/python3.6 /usr/bin/python /usr/lib/python2.7 /usr/lib/python3.6 /usr/lib64/python2.7 /usr/lib64/python3.6 /etc/python /usr/include/python2.7 /usr/include/python3.6m /usr/share/man/man1/python.1.gz
```

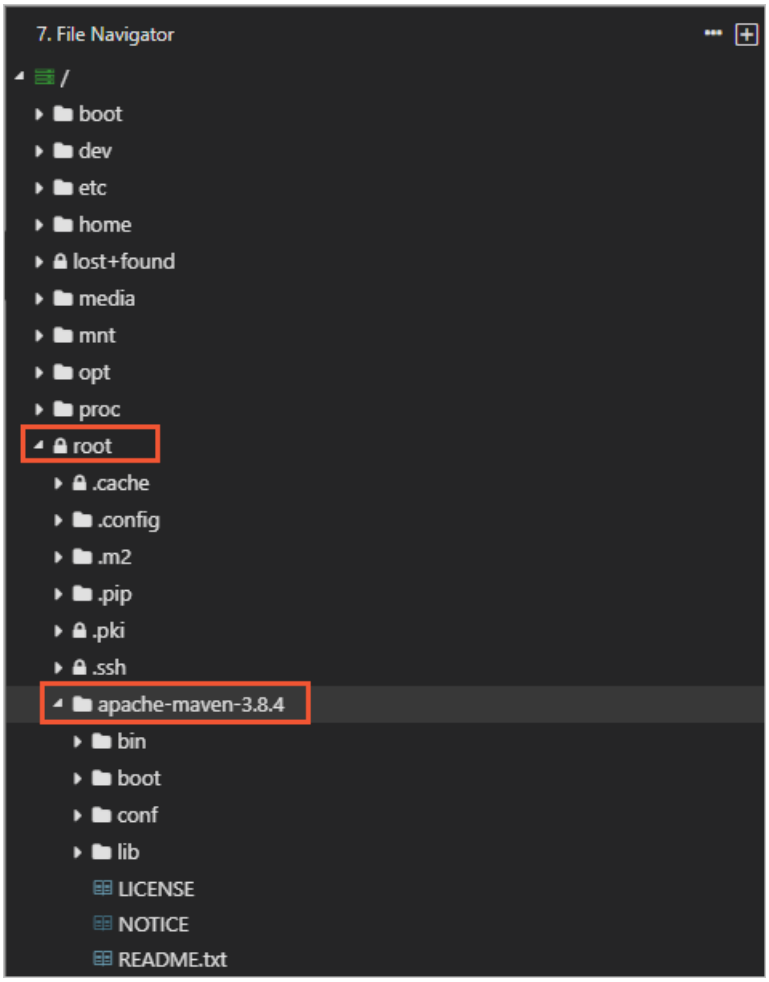
- Edit Python environment variables. Sample commands:

```
# Edit the configuration file for environment variables.
vim /etc/profile
# Press i to enter the edit mode and add environment variables to the end of the configuration file.
# Replace PATH with the actual Python installation path.
export PATH=/usr/bin/python/bin/:$PATH
# Press the Esc key to exit the edit mode and enter :wq to close the configuration file.
# Run the following command to make the modification take effect:
source /etc/profile
# Check whether the Python environment variables are successfully configured.
python --version
# Sample output:
Python 2.7.5
```

```
>_ 2. root@                    :~ ×
else
    export HISTCONTROL=ignoredups
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

# By default, we want umask to get set. This sets it for login shell
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-1.1.al7.x86_64
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$JAVA_HOME/bin:$PATH
export SPARK_HOME=/root/spark-2.3.0-odps0.33.0
export PATH=$SPARK_HOME/bin:$PATH
export PATH=/usr/bin/python/bin/:$PATH
export MAVEN_HOME=/root/apache-maven-3.8.4
export PATH=$MAVEN_HOME/bin:$PATH
export LC_ALL=en_US.UTF-8
export PATH=$PATH:/usr/local/git/bin
"/etc/profile" 86L, 2256C                                    86,1          Bot
```

- Configure Maven environment variables.

- Obtain the path to which the Maven package is decompressed. The following figure shows that the path is `/root/apache-maven-3.8.4`. The actual decompression path and package name prevail.



- Edit Maven environment variables. Sample commands:

```
# Edit the configuration file for environment variables.
vim /etc/profile
# Press i to enter the edit mode and add environment variables to the end of the config
uration file.
# Replace MAVEN_HOME with the actual path to which the Maven package is decompressed.
export MAVEN_HOME=/root/apache-maven-3.8.4
export PATH=$MAVEN_HOME/bin:$PATH
# Press the Esc key to exit the edit mode and enter :wq to close the configuration file
.
# Run the following command to make the modification take effect:
source /etc/profile
# Check whether the Maven environment variables are successfully configured.
mvn -version
# Sample output:
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: /root/apache-maven-3.8.4
Java version: 1.8.0_322, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-1.8.0-openjd
k-1.8.0.322.b06-1.1.al7.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.19.91-25.1.al7.x86_64", arch: "amd64", family: "unix"
```

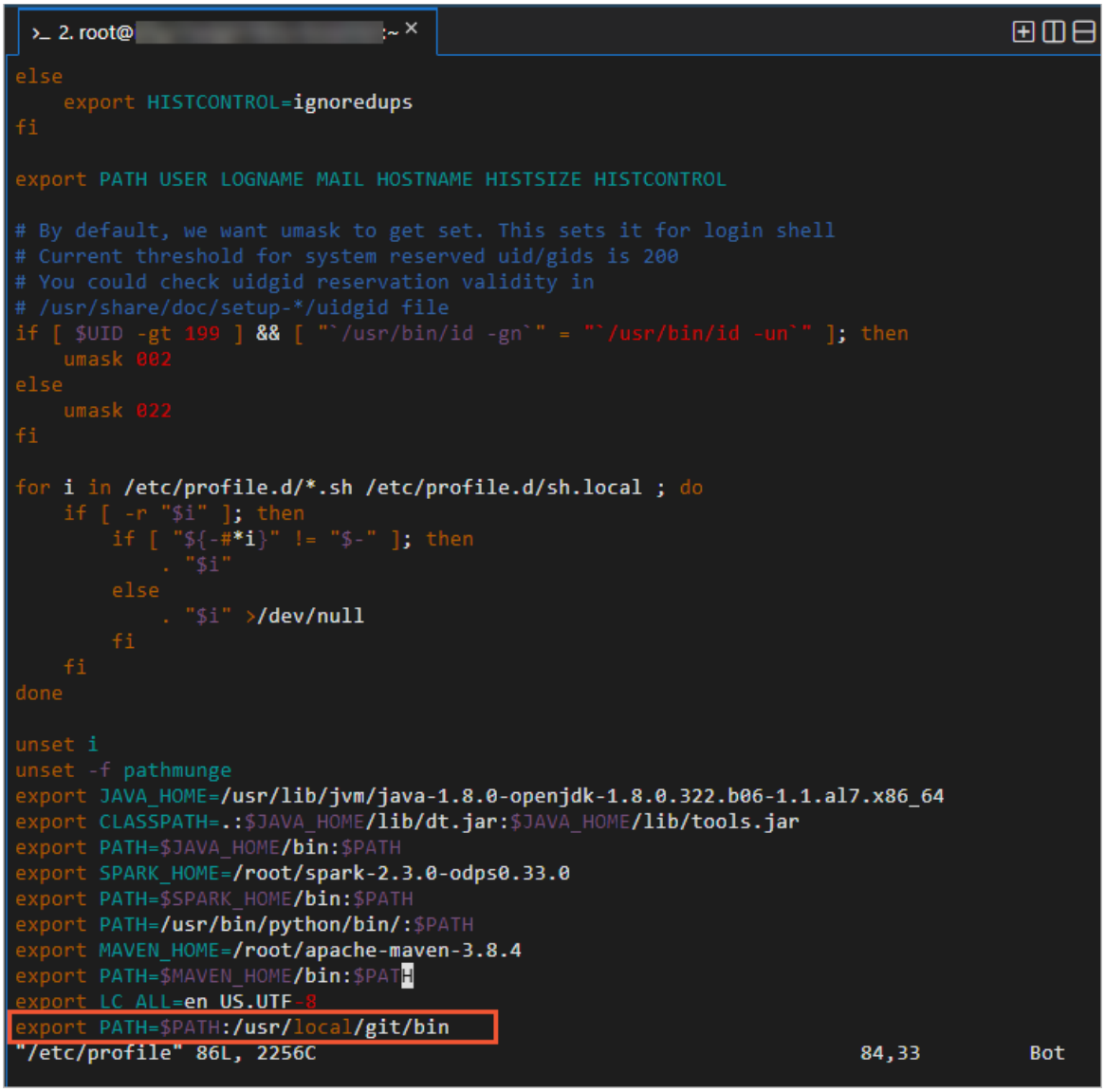
- Configure Git environment variables.
  - Obtain the Git installation path. Sample command:

```
whereis git
```

```
[root@ ~]# whereis git
git: /usr/bin/git /usr/local/git /usr/share/man/man1/git.1.gz
```

  - Edit Git environment variables. Sample commands:

```
# Edit the configuration file for environment variables.
vim /etc/profile
# Press i to enter the edit mode and add environment variables to the end of the config
uration file. # Replace PATH with the actual Git installation path.
export PATH=/usr/local/git/bin/:$PATH
# Press the Esc key to exit the edit mode and enter :wq to close the configuration file
.
# Run the following command to make the modification take effect:
source /etc/profile
# Check whether the Git environment variables are successfully configured.
git --version
# Sample output:
git version 2.24.4
```



Configure the spark-defaults.conf file

If you use the Spark on MaxCompute client for the first time, rename the *spark-defaults.conf.template* file in the `conf` folder of the decompression path of the Spark on MaxCompute client package as *spark-defaults.conf*. Then, configure information in the file. If you do not rename the file, the configuration cannot take effect. Sample commands:

```
# Switch to the path to which the Spark on MaxCompute client package is decompressed and go
to the conf folder. The actual path prevails.
cd /root/spark-2.3.0-odps0.33.0/conf
# Rename the spark-defaults.conf.template file.
mv spark-defaults.conf.template spark-defaults.conf
# Edit the spark-defaults.conf file.
vim spark-defaults.conf
# Press i to enter the edit mode and add the following configuration to the end of the conf
iguration file.
spark.hadoop.odps.project.name = <MaxCompute_project_name>
spark.hadoop.odps.access.id = <AccessKey_id>
spark.hadoop.odps.access.key = <AccessKey_secret>
spark.hadoop.odps.end.point = <Endpoint>   # The endpoint that is used to connect the Spark
on MaxCompute client to your MaxCompute project. You can modify the endpoint based on your
business requirements. For more information about endpoints, see Endpoints.
spark.hadoop.odps.runtime.end.point = <VPC_endpoint>  # The virtual private cloud (VPC) end
point of the region where your MaxCompute project resides. You can modify the endpoint base
d on your business requirements.
# For Spark 2.3.0, set spark.sql.catalogImplementation to odps. For Spark 2.4.5, set spark.
sql.catalogImplementation to hive.
spark.sql.catalogImplementation={odps|hive}
# Retain the configurations of the following parameters:
spark.hadoop.odps.task.major.version = cupid_v2
spark.hadoop.odps.cupid.container.image.enable = true
spark.hadoop.odps.cupid.container.vm.engine.type = hyper
spark.hadoop.odps.cupid.webproxy.endpoint = http://service.cn.maxcompute.aliyun-inc.com/api
spark.hadoop.odps.moye.trackurl.host = http://jobview.odps.aliyun.com
```

- MaxCompute_project_name: the name of the MaxCompute project that you want to access.

  This parameter specifies the name of your MaxCompute project instead of the DataWorks workspace to which the MaxCompute project corresponds. You can log on to the MaxCompute console. In the top navigation bar, select a region. Then, view the name of the MaxCompute project on the **Project management** tab.

- AccessKey_id: the AccessKey ID that is used to access the MaxCompute project.

  You can obtain the AccessKey ID from the AccessKey Pair page.

- AccessKey_secret: the AccessKey secret that corresponds to the AccessKey ID.

  You can obtain the AccessKey secret from the AccessKey Pair page.

- Endpoint: the public endpoint of the region where your MaxCompute project resides.

  For more information about the public endpoint of each region, see Endpoints in different regions (Internet).

- VPC_endpoint: the VPC endpoint of the region where your MaxCompute project resides.

  For more information about the VPC endpoint of each region, see Endpoints in different regions (VPC).

Other configuration parameters are required for special scenarios and features. For more information, see Spark on MaxCompute configuration details.

Prepare a project

Spark on MaxCompute provides a demo project template. We recommend that you download and copy the template to develop your application.

> **Notice** In the demo project, the dependency scope for Spark on MaxCompute is provided. Do not change this scope. Otherwise, the job that you submit may not run as expected.

The following commands are used to prepare a project in a Linux operating system.

- Download the Spark-2.x template and compile the template.

```
git clone https://github.com/aliyun/MaxCompute-Spark.git
cd MaxCompute-Spark/spark-2.x
mvn clean package
```

- Download the Spark-1.x template and compile the template.

```
git clone https://github.com/aliyun/MaxCompute-Spark.git
cd MaxCompute-Spark/spark-1.x
mvn clean package
```

After you run the preceding commands, if the project fails to be created, the environment configurations are invalid. Follow the preceding instructions to check the configurations. If any invalid configurations are found, modify the configurations.

Configure dependencies

In the Spark on MaxCompute project that you prepared, configure the dependencies. The following commands show an example.

- Configure the dependencies that are required for accessing tables in your MaxCompute project.
  - The Spark-1.x template is used.

```
# Go to the spark-1.x folder.
cd MaxCompute-Spark/spark-1.x
# Edit the POM file to add the odps-spark-datasource dependency.
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-spark-datasource_2.10</artifactId>
  <version>3.3.8-public</version>
</dependency>
```

  - The Spark-2.x template is used.

```
# Go to the spark-2.x folder.
cd MaxCompute-Spark/spark-2.x
# Edit the POM file to add the odps-spark-datasource dependency.
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-spark-datasource_2.11</artifactId>
    <version>3.3.8-public</version>
</dependency>
```

- Configure the dependency that is required for accessing Object Storage Service (OSS).

  If your job needs to access OSS, add the following dependency:

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>hadoop-fs-oss</artifactId>
    <version>3.3.8-public</version>
</dependency>
```

For more information about the dependencies that are required when the Spark-1.x or Spark-2.x template is used, see Spark-1.x POM file or Spark-2.x POM file.

Reference external files

If you develop the following types of Spark jobs, you must reference external files:

- Spark jobs that need to read data from specific configuration files.
- Spark jobs that require additional resource packages or third-party libraries such as JAR files or Python libraries.

In practical application, you must upload external files before you reference the files. You can upload external files by using one of the following methods:

- Method 1: Upload external files by using Spark parameters.

  Spark on MaxCompute supports the following parameters that are defined by Apache Spark: `--jars`, `--py-files`, `--files`, and `--archives`. You can configure these parameters to upload external files when you submit jobs. When you run the jobs, these external files are uploaded to your working directories.

  - Use the spark-submit script to upload files on the Spark on MaxCompute client.

    > **Note**
    > - `--jars`: This parameter specifies the JAR packages that you want to upload to the current working directories of the driver and each executor. Package names are separated by commas (,). These JAR packages are added to the classpaths of the driver and each executor. You can use `"./your_jar_name"` in the configurations of Spark jobs to specify these packages. This is the case in Apache Spark.
    > - `--files` and `--py-files`: The two parameters specify the common files or Python files that you want to upload to the current working directories of the driver and each executor. File names are separated by commas (,). You can use `"./your_file_name"` in the configurations of Spark jobs to specify these files. This is the case in Apache Spark.
    > - `--archives`: This parameter is slightly different from the parameter defined by Apache Spark. Configure this parameter in the `xxx#yyy` format and separate file names with commas (,). After you configure the --archives parameter, the archive files that you specify, such as ZIP files, are decompressed to the subdirectories of the current working directories of the driver and each executor. For example, if this parameter is set to `xx.zip#yy`, you must use `"./yy/xx/"` to reference the content in the archive file. If this parameter is set to `xx.zip`, you must use `"./xx.zip/xx/"` to reference the content in the archive file. If you need to use `"./xxx/"` to decompress the archive file directly to the current working directory, you must use the `spark.hadoop.odps.cupid.resources` parameter.

- Use DataWorks to add the resources that are required by a job. For more information, see Create a MaxCompute resource.

  > **Note** In the DataWorks console, you can upload files with a size of up to 50 MB. If the size of files that you want to upload exceeds 50 MB, you must use the MaxCompute client to upload these files as MaxCompute resources and add these resources to the DataStudio page in the DataWorks console. For more information about MaxCompute resources, see MaxCompute resources.

- Method 2: Upload files as MaxCompute resources.

  Spark on MaxCompute provides the `spark.hadoop.odps.cupid.resources` parameter. This parameter allows you to directly reference resources in MaxCompute. When you run a job, resources that are referenced are uploaded to the working directories. To upload files as MaxCompute resources, perform the following steps:

  i. Log on to the MaxCompute client and upload files as resources to your MaxCompute project. The maximum size of a file that you can upload is 500 MB.

  ii. Add the `spark.hadoop.odps.cupid.resources` parameter to the configurations of your Spark job. This parameter specifies the MaxCompute resources that are required for running the Spark job. Configure this parameter in the `<projectname>.<resourcename>` format. If you need to reference multiple files, separate the file names with commas (,). The following configuration shows an example.

  ```
  spark.hadoop.odps.cupid.resources=public.python-python-2.7-ucs4.zip,public.myjar.jar
  ```

  The specified resource file is downloaded to the current working directories of the driver and each executor. By default, the downloaded file is named in the `<projectname>.<resourcename>` format.

  You can also rename the file in the `<projectname>.<resourcename>:<newresourcename>` format. The following configuration shows an example.

  ```
  spark.hadoop.odps.cupid.resources=public.myjar.jar:myjar.jar
  ```

  > **Notice** The spark.hadoop.odps.cupid.resources parameter takes effect only after you configure this parameter in the spark-defaults.conf file or in the DataWorks console. This parameter does not take effect if this parameter is written in code.

After you upload files by using one of the preceding methods, you can reference external files in code. The following code shows an example:

```
val targetFile = "File name"
val file = Source.fromFile(targetFile)
for (line <- file.getLines)
    println(line)
file.close
```

Conduct SparkPi smoke testing

After you complete the preceding operations, conduct smoke testing to check the end-to-end connectivity of Spark on MaxCompute. For example, you can run the following commands to conduct SparkPi smoke testing for a Spark 2.x application:

```
# /path/to/MaxCompute-Spark Configure a valid path for the compiled JAR package of the appl
ication.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.SparkPi \
/path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar
# If the following log information is displayed, smoke testing is successful.
19/06/11 11:57:30 INFO Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: 11.222.166.90
         ApplicationMaster RPC port: 38965
         queue: queue
         start time: 1560225401092
         final status: SUCCEEDED
```

Notes on running Spark on MaxCompute in local mode by using IntelliJ IDEA

In most cases, code is run in cluster mode after local debugging is successful. However, Spark on MaxCompute allows you to run code in local mode by using IntelliJ IDEA. When you run code in local mode by using IntelliJ IDEA, take note of the following points:

- Specify the spark.master parameter in the code.

  ```
  val spark = SparkSession
        .builder()
        .appName("SparkPi")
        .config("spark.master", "local[4]") // The code can run only after you set the spar
  k.master parameter to local[N]. N indicates the parallelism.
        .getOrCreate()
  ```

- Add the following dependency of the Spark on MaxCompute client in IntelliJ IDEA.

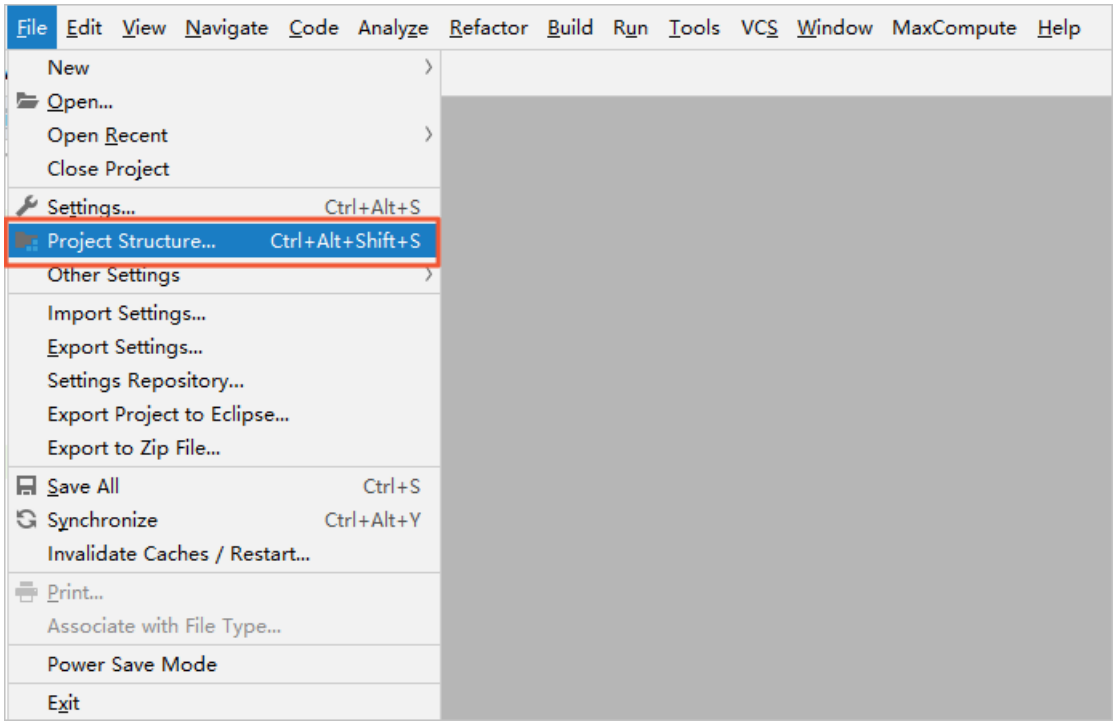
  ```
  <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_${scala.binary.version}</artifactId>
      <version>${spark.version}</version>
      <scope>provided</scope>
  </dependency>
  ```

  In the *pom.xml* file, the scope parameter is set to provided. This setting may cause the "NoClassDefFoundError" error when you run the code.

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/spark/sql/SparkSess
ion$
    at com.aliyun.odps.spark.examples.SparkPi$.main(SparkPi.scala:27)
    at com.aliyun.odps.spark.examples.Spa. r. kPi.main(SparkPi.scala)
Caused by: java.lang.ClassNotFoundException: org.apache.spark.sql.SparkSession$
    at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:335)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    ... 2 more
```

You can perform the following steps to manually add the directory specified by the --jars parameter on Spark on MaxCompute to the project template of IntelliJ IDEA. This way, the configuration `scope=provided` can be retained and the "NoClassDefFoundError" error is not returned when you run the code in local mode by using IntelliJ IDEA.

i. In the main menu bar of IntelliJ IDEA, choose **File** > **Project Structure...**.



ii. On the **Project Structure** page, click **Modules** in the left-side navigation pane. Then, select the resource package, and click the **Dependencies** tab for the resource package.

iii. On the **Dependencies** tab, click the plus sign (**+**) in the lower-left corner and select **JARs or directories...** to add the directory specified by the --jars parameter on Spark on MaxCompute.

- The *spark-defaults.conf* file cannot be directly referenced when you run the code in local mode. You must manually configure related parameters in the code.

  If you submit jobs by using the spark-submit script, the system reads the configurations from the *spark-defaults.conf* file. If you submit jobs in local mode, you must manually configure related parameters in the code. The following code shows how to use Spark SQL to read data from MaxCompute tables in local mode.

```
val spark = SparkSession
      .builder()
      .appName("SparkPi")
      .config("spark.master", "local[4]") // The code can run after you set spark.master
to local[N]. N indicates the parallelism.
      .config("spark.hadoop.odps.project.name", "****")
      .config("spark.hadoop.odps.access.id", "****")
      .config("spark.hadoop.odps.access.key", "****")
      .config("spark.hadoop.odps.end.point", "http://service.cn.maxcompute.aliyun.com/api
")
      .config("spark.sql.catalogImplementation", "odps")
      .getOrCreate()
```

Notes on using Spark 2.4.5

- Use Spark 2.4.5 to submit jobs
  - Submit a job in a Yarn cluster.
  - Set the `spark.hadoop.odps.spark.version` parameter to spark-2.4.5-odps0.33.0 in the DataWorks console. If the Spark version of exclusive resource groups in DataWorks is not updated to Spark 2.4.5, you can use shared resource groups to schedule jobs or contact the MaxCompute technical support team to update the Spark version.
- Changes in using Spark 2.4.5
  - If you submit jobs in a Yarn cluster, you must run the `export HADOOP_CONF_DIR=$SPARK_HOME/conf` command to add the SPARK_HOME environment variable.
  - If you perform debugging in local mode, you must create a file named odps.conf in the `$SPARK_HOME/conf` path and add the following configurations to the file.

    ```
    odps.project.name =
    odps.access.id =
    odps.access.key =
    odps.end.point =
    ```
- Changes in the parameter settings of Spark 2.4.5
  - `spark.sql.catalogImplementation` is set to `hive`.
  - `spark.sql.sources.default` is set to `hive`.
  - `spark.sql.odps.columnarReaderBatchSize`: specifies the number of rows from which the vectorized reader reads data at a time. Default value: 4096.
  - `spark.sql.odps.enableVectorizedReader`: specifies whether to enable the vectorized reader. Default value: True.
  - `spark.sql.odps.enableVectorizedWriter`: specifies whether to enable the vectorized writer. Default value: True.
  - `spark.sql.odps.split.size`: This parameter can be used to adjust the concurrency of data reading operations on MaxCompute tables. By default, this parameter is set to 256 for each partition. Unit: MB.

4.5.3. Running modes

Spark on MaxCompute supports three running modes: local, cluster, and DataWorks.

Local mode

Spark on MaxCompute allows you to debug jobs in local mode that is used in native Spark.

The local mode is similar to the YARN cluster mode. To use the local mode, you must make the following preparations:

1. Create a MaxCompute project and obtain the AccessKey ID and AccessKey secret of the account that can be used to access the MaxCompute project.
2. Download the Spark on MaxCompute client.
3. Prepare environment variables.
4. Configure the spark-defaults.conf file.
5. Download and compile a demo project template.

For more information, see Set up a Spark on MaxCompute development environment.

Submit a job by running the spark-submit script on the Spark on MaxCompute client. The following code provides an example:

```
## Java/Scala
cd $SPARK_HOME
./bin/spark-submit --master local[4] --class com.aliyun.odps.spark.examples.SparkPi \
/path/to/odps-spark-examples/spark-examples/target/spark-examples-2.0.0-SNAPSHOT-shaded.jar
## PySpark
cd $SPARK_HOME
./bin/spark-submit --master local[4] \
/path/to/odps-spark-examples/spark-examples/src/main/python/odps_table_rw.py
```

Precautions

- In local mode, Tunnel is used to read data from and write data to MaxCompute tables. As a result, the read and write operations in local mode are slower than those in YARN cluster mode.
- In local mode, Spark on MaxCompute is run on your on-premises machine. Therefore, you may encounter the situation that you can access Spark on MaxCompute that runs in local mode over a virtual private cloud (VPC), but you cannot access Spark on MaxCompute that runs in YARN cluster mode over a VPC.

  In local mode, the network is not isolated. However, in YARN cluster mode, the network is isolated, and you must configure the required parameters for access over a VPC.
- In local mode, you must use a public endpoint to access Spark on MaxCompute over a VPC. However, in YARN cluster mode, you must use an internal endpoint to access Spark on MaxCompute over a VPC. For more information about how to obtain an endpoint, see Endpoints.
- If you run Spark on MaxCompute in IntelliJ IDEA in local mode, you must specify the related configurations in the code. However, these configurations must be deleted from the code if you want to run Spark on MaxCompute in IntelliJ IDEA in YARN cluster mode.

Run Spark on MaxCompute in IntelliJ IDEA in local mode

Spark on MaxCompute allows you to directly run code in IntelliJ IDEA in local mode by using N threads. This frees you from submitting code on the Spark on MaxCompute client. Take note of the following items when you run the code:

- You must manually specify the related configurations in the code when you run the code in IntelliJ IDEA in local mode. You cannot directly reference the configurations in the spark-defaults.conf file. The following code provides an example:

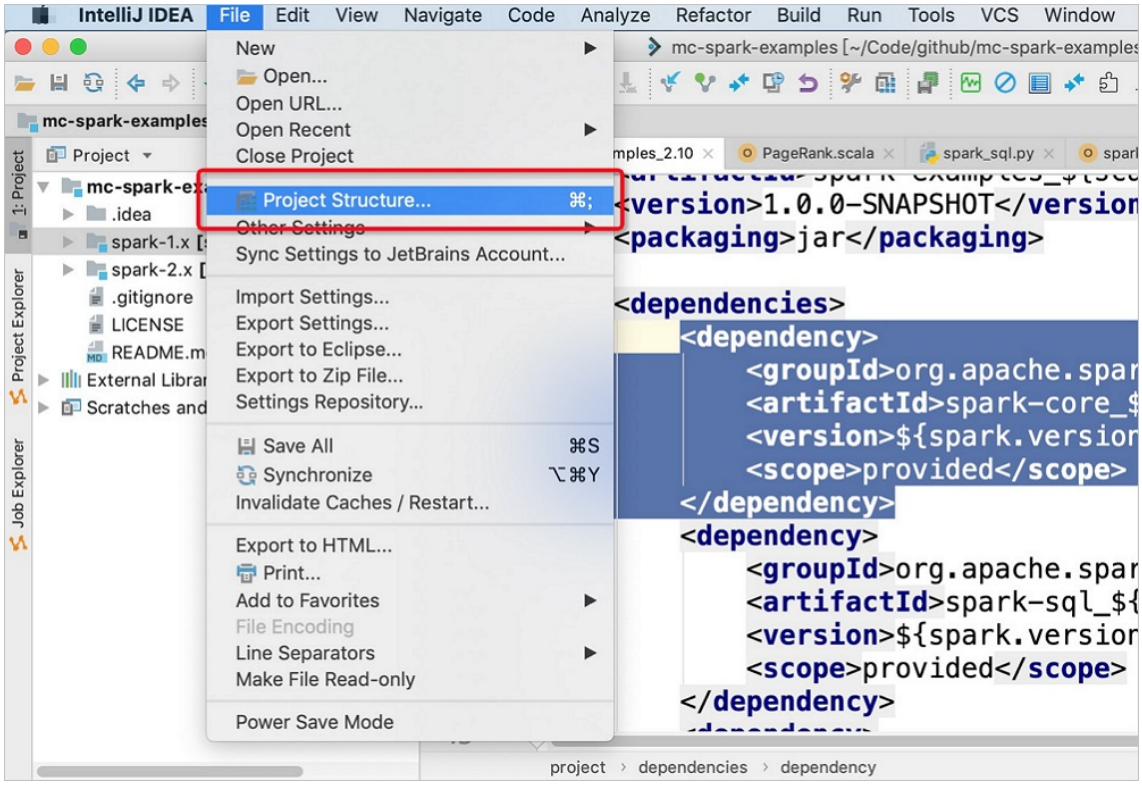
```
val spark = SparkSession
      .builder()
      .appName("SparkPi")
      .config ("spark.master", "local [4]") // The code can run directly after you set sp
ark.master to local[N]. N is the number of threads.
      .config("spark.hadoop.odps.project.name", "<project_name>")
      .config("spark.hadoop.odps.access.id", "<accesskey_id>")
      .config("spark.hadoop.odps.access.key", "<accesskey_secret>")
      .config("spark.hadoop.odps.end.point", "http://service.cn.maxcompute.aliyun.com/api
")
      .config("spark.sql.catalogImplementation", "odps")
      .getOrCreate()
```

- You must manually add the required dependencies in the `jars` folder for the Spark on MaxCompute client in IntelliJ IDEA. Otherwise, the system reports the following error:

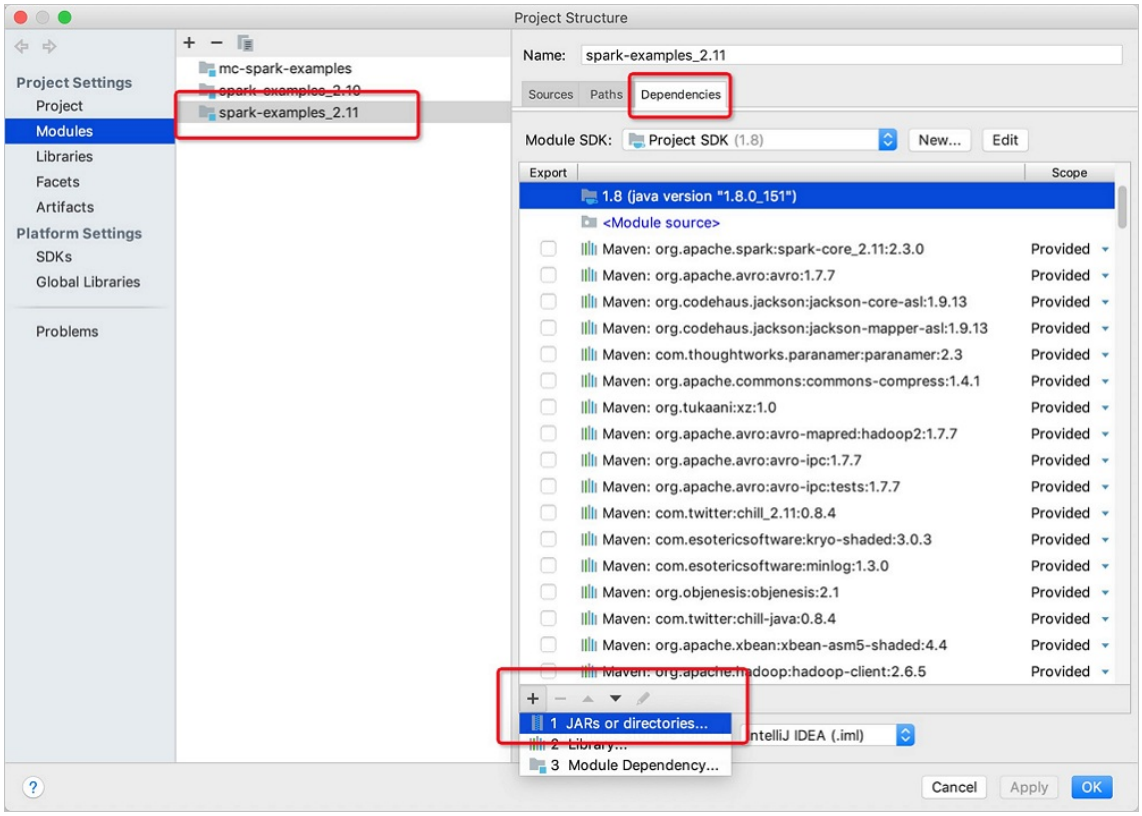
```
 the value of spark.sql.catalogimplementation should be one of hive in-memory but was odp
s
```

  You can refer to the following steps to configure the dependencies:

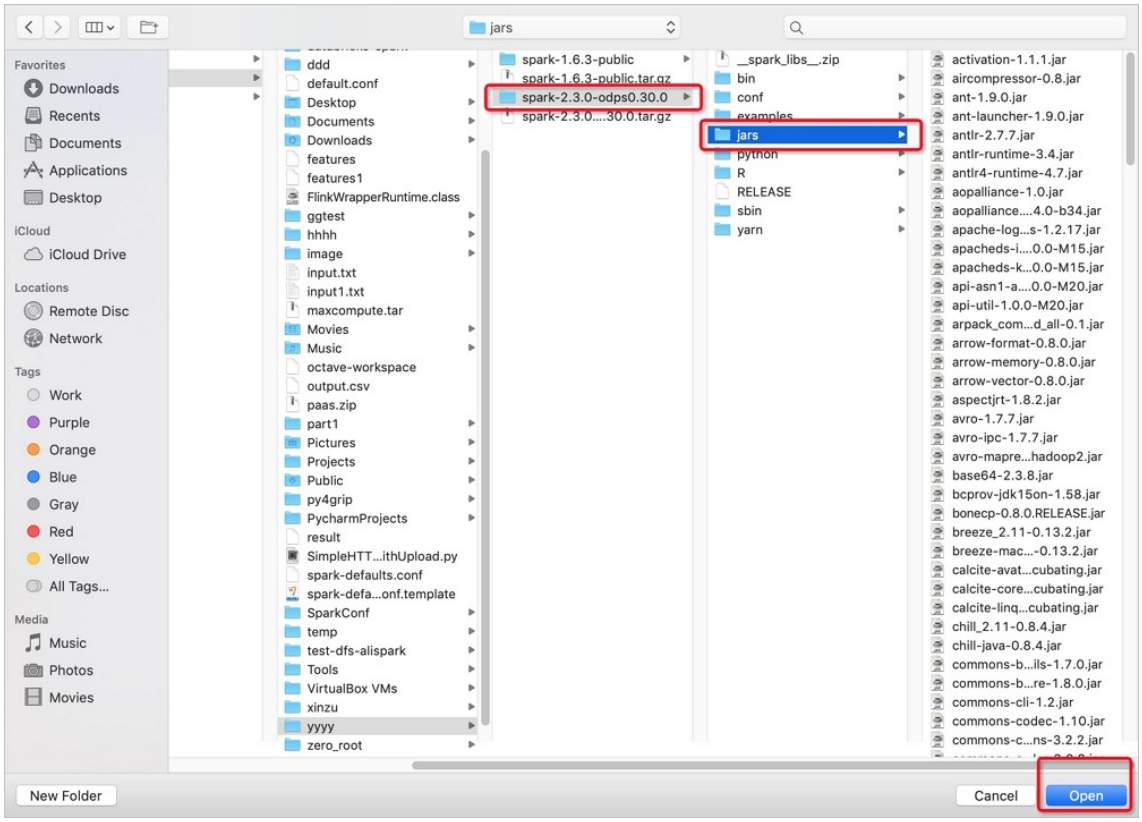
  i. In the main menu bar of IntelliJ IDEA, choose **File > Project Structure**.



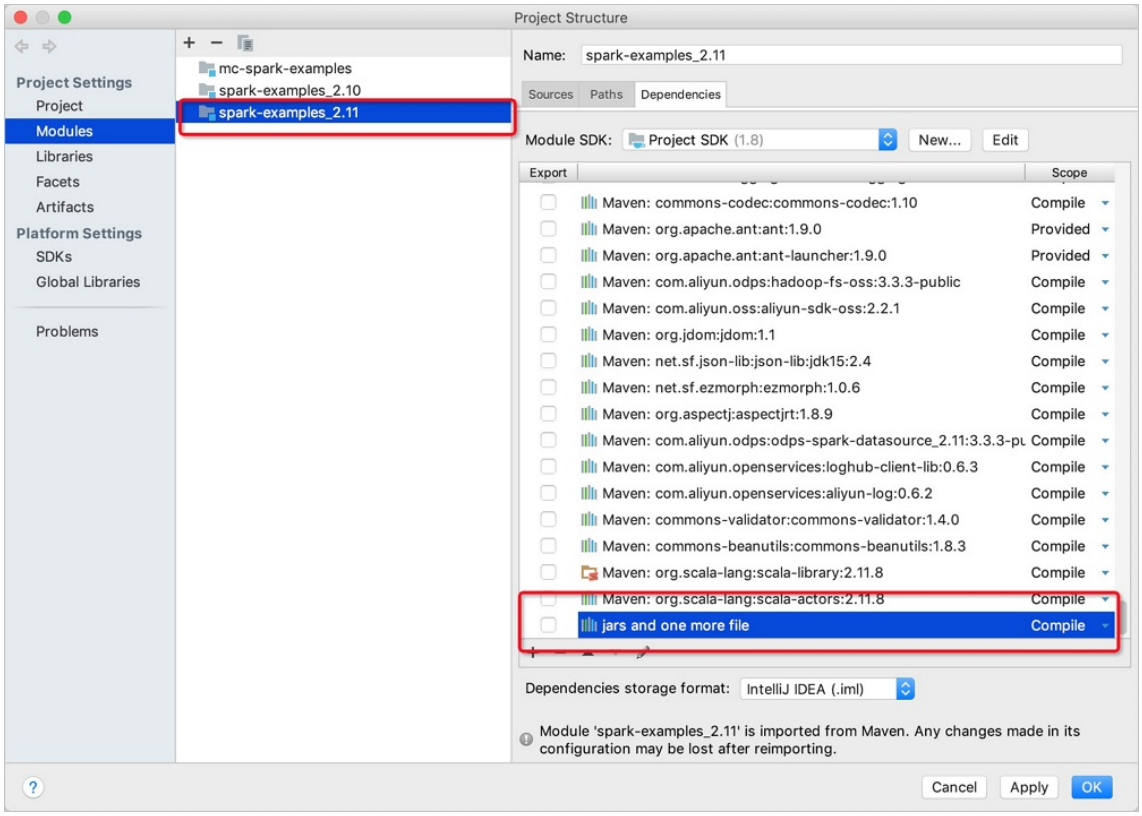
  ii. On the **Project Structure** page, click **Modules** in the left-side navigation pane. On the tab that appears, click spark-examples_2.11. In the panel that appears, click the **Dependencies** tab. Then, click the + icon in the lower-left corner and select **JARS or directories**.



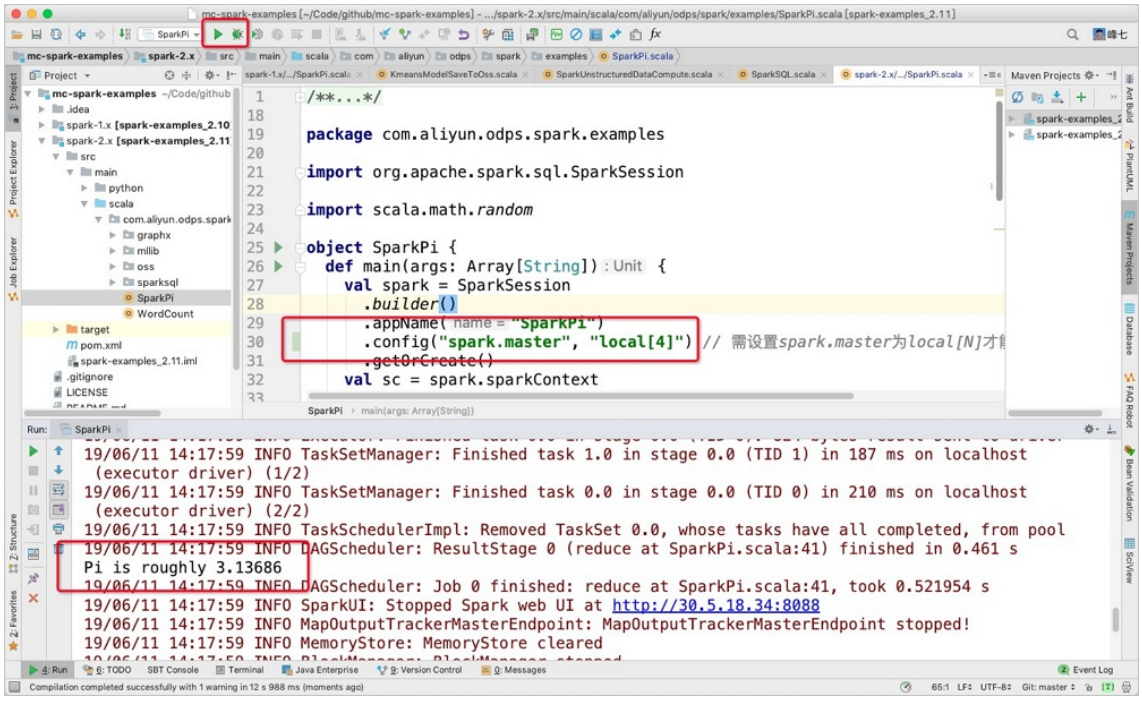
iii. From the opened **jars** folder, choose the required version of the Spark on MaxCompute package > jars > required JAR file and click **Open** in the lower-right corner.



iv. Click **OK**.



v. Submit the configurations in IntelliJ IDEA.



Cluster mode

In cluster mode, you must specify the Main method as the entry point of a custom application. A Spark job ends when Main succeeds or fails. This mode is suitable for offline jobs. You can use Spark on MaxCompute in this mode together with DataWorks to schedule jobs. The following code provides an example on how to use command lines to run Spark on MaxCompute in this mode:

```
# /path/to/MaxCompute-Spark: the path where the compiled application JAR package is saved.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.SparkPi \
/path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar
```

DataWorks mode

You can run the offline jobs of Spark on MaxCompute in cluster mode in DataWorks to integrate and schedule the other types of nodes.

> **Note** DataWorks allows you to create Spark nodes in the following regions: China (Hangzhou), China (Beijing), China (Shanghai), China (Shenzhen), China (Hong Kong), US (Silicon Valley), Germany (Frankfurt), India (Mumbai), and Singapore (Singapore).

Procedure:

1. Upload the required resource in the DataWorks workflow and click the **Submit** icon.

   The resource that is uploaded appears in the navigation tree, as shown in the following figure.

2. In the created workflow, select **ODPS Spark** from **Data Analytics**.
3. Double-click the ODPS Spark node and configure the parameters for the Spark job.Each of the **Spark Version** and **Language** parameters has two options for the ODPS Spark node. The other parameters that you need to configure vary based on the **Language** parameter. You can configure the parameters as prompted. For more information, see Create a MaxCompute Spark node. Where:
   - **Main JAR Resource**: the resource file used by the job. You must upload the resource file to DataWorks before you perform this operation.
   - **Configuration Items**: the configuration items required to submit the job.

     You do not need to configure `spark.hadoop.odps.access.id`, `spark.hadoop.odps.access.key`, and `spark.hadoop.odps.end.point`. By default, the values of these configuration items are the same as those of the MaxCompute project. You can also explicitly specify these configuration items to overwrite their default values.

     You must add the configurations in the `spark-default.conf` file to the configuration items of the ODPS Spark node one by one. The configurations include the number of executors, memory size, and `spark.hadoop.odps.runtime.end.point`.

     The resource file and configuration items of the ODPS Spark node map the parameters and items of the spark-submit command, as described in the following table. You do not need to upload the *spark-defaults.conf* file. Instead, you must add the configurations in the *spark-defaults.conf* file to the configuration items of the ODPS Spark node one by one.

ODPS Spark node	spark-submit
Main JAR Resource and Main Python Resource	`app jar or python file`
Configuration Items	`--conf PROP=VALUE`
Main Class	`--class CLASS_NAME`
Arguments	`[app arguments]`

ODPS Spark node	spark-submit
JAR Resources	`--jars JARS`
Python Resources	`--py-files PY_FILES`
File Resources	`--files FILES`
Archive Resources	`--archives ARCHIVES`

4. Manually run the ODPS Spark node to view the operational logs of the job and obtain the URLs of both Logview and Jobview from the logs for further analysis and diagnosis.

   After the Spark job is defined, you can orchestrate and schedule services of different types in the workflow.

4.5.4. Java and Scala development examples

4.5.4.1. Overview

This topic describes how to develop a demo project on Spark on MaxCompute by using Java or Scala.

Download a demo project

Spark on MaxCompute provides a demo project template. We recommend that you download and copy the template to develop your application.

Run the following commands to download the demo project template:

```
# Download and compile the Spark 1.x template.
git clone https://github.com/aliyun/MaxCompute-Spark.git
cd spark-1.x
mvn clean package
# Download and compile the Spark 2.x template.
git clone https://github.com/aliyun/MaxCompute-Spark.git
cd spark-2.x
mvn clean package
```

Notice In the demo project, the scope parameter for the Spark dependency is set to provided. Do not modify this parameter. Otherwise, the submitted job does not run normally.

Spark 1.x examples

Examples of a Spark 1.x demo project:

- WordCount example (Scala)
- Example of reading data from or writing data to a MaxCompute table (Scala)
- Example of reading data from or writing data to a MaxCompute table (Python)
- Example of reading data from or writing data to a MaxCompute table (Java)

Spark 2.x examples

Examples of a Spark 2.x demo project:

- WordCount example (Scala)
- Example of reading data from or writing data to a MaxCompute table (Scala)
- GraphX PageRank example (Scala)
- MLlib KMeans-ON-OSS example (Scala)
- OSS UnstructuredData example (Scala)
- SparkPi example (Scala)
- Spark Streaming LogHub example (Scala)
- Example of using Spark Streaming LogHub to write data to MaxCompute (Scala)
- Spark Streaming DataHub example (Scala)
- Example of using Spark Streaming DataHub to write data to MaxCompute (Scala)
- Spark Streaming Kafka example (Scala)
- Spark StructuredStreaming DataHub example (Scala)
- Spark StructuredStreaming Kafka example (Scala)
- Spark StructuredStreaming LogHub example (Scala)
- Example of using PySpark to read data from or write data to a MaxCompute table (Python)
- Example of using PySpark to write data to OSS (Python)
- Spark SQL example (Java)
- Example of reading data from MaxCompute and writing the data to HBase
- Examples of reading data from and writing data to OSS objects
- Example of reading data from MaxCompute and writing the data to OSS

4.5.4.2. Spark 1.x examples

This topic describes how to configure Spark 1.x dependencies and provides some examples.

Configure dependencies for Spark 1.x

If you want to submit your Spark 1.x application by using Spark on MaxCompute, you must add the following dependencies to the *pom.xml* file.

```
<properties>
    <spark.version>1.6.3</spark.version>
    <cupid.sdk.version>3.3.3-public</cupid.sdk.version>
    <scala.version>2.10.4</scala.version>
    <scala.binary.version>2.10</scala.binary.version>
</properties>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
```

```
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-mllib_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-streaming_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>cupid-sdk</artifactId>
    <version>${cupid.sdk.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>hadoop-fs-oss</artifactId>
    <version>${cupid.sdk.version}</version>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-spark-datasource_${scala.binary.version}</artifactId>
    <version>${cupid.sdk.version}</version>
</dependency>
<dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-library</artifactId>
    <version>${scala.version}</version>
</dependency>
<dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-actors</artifactId>
    <version>${scala.version}</version>
</dependency>
```

In the preceding code, set the scope parameter based on the following instructions:

- Set it to provided for all packages that are released in the Apache Spark community, such as spark-core and spark-sql.
- Set it to compile for the odps-spark-datasource module.

WordCount example (Scala)

- Sample code

  WordCount.scala

- How to commit

```
cd /path/to/MaxCompute-Spark/spark-1.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.WordCount \
    /path/to/MaxCompute-Spark/spark-1.x/target/spark-examples_2.10-1.0.0-SNAPSHOT-shaded.
jar
```

Example of reading data from or writing data to a MaxCompute table (Scala)

- Sample code

  SparkSQL.scala

- How to commit

```
cd /path/to/MaxCompute-Spark/spark-1.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.sparksql.Sp
arkSQL \
    /path/to/MaxCompute-Spark/spark-1.x/target/spark-examples_2.10-1.0.0-SNAPSHOT-shaded.
jar
```

Example of reading data from or writing data to a MaxCompute table (Python)

For more information about the Python sample code for reading data from or writing data to a MaxCompute table, see spark_sql.py.

Example of reading data from or writing data to a MaxCompute table (Java)

For more information about the Java sample code for reading data from or writing data to a MaxCompute table, see JavaSparkSQL.java.

4.5.4.3. Spark 2.x examples

This topic describes how to configure Spark 2.x dependencies and provides some examples.

Configure dependencies for Spark 2.x

If you want to submit your Spark 2.x application by using Spark on MaxCompute, you must add the following dependencies to the *pom.xml* file. For more information about *pom.xml*, see pom.xml.

```
<properties>
    <spark.version>2.3.0</spark.version>
    <cupid.sdk.version>3.3.8-public</cupid.sdk.version>
    <scala.version>2.11.8</scala.version>
    <scala.binary.version>2.11</scala.binary.version>
```

```
</properties>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-mllib_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-streaming_${scala.binary.version}</artifactId>
    <version>${spark.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>cupid-sdk</artifactId>
    <version>${cupid.sdk.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>hadoop-fs-oss</artifactId>
    <version>${cupid.sdk.version}</version>
</dependency>
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-spark-datasource_${scala.binary.version}</artifactId>
    <version>${cupid.sdk.version}</version>
</dependency>
<dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-library</artifactId>
    <version>${scala.version}</version>
</dependency>
<dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-actors</artifactId>
    <version>${scala.version}</version>
</dependency>
```

In the preceding code, set the scope parameter based on the following instructions:

- Set scope to provided for all packages that are released in the Apache Spark community, such as spark-core and spark-sql.
- Set scope to compile for the odps-spark-datasource module.

WordCount example (Scala)

- Sample code

  WordCount.scala

- How to commit

```
cd /path/to/MaxCompute-Spark/spark-2.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class \
    com.aliyun.odps.spark.examples.WordCount \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Example of reading data from or writing data to a MaxCompute table (Scala)

- Sample code

  SparkSQL.scala

- How to commit

```
cd /path/to/MaxCompute-Spark/spark-2.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.sparksql.Sp
arkSQL \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

GraphX PageRank example (Scala)

- Sample code

  PageRank.scala

- How to commit

```
cd /path/to/MaxCompute-Spark/spark-2.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.graphx.Page
Rank \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

MLlib KMeans-ON-OSS example (Scala)

For more information about how to configure `spark.hadoop.fs.oss.ststoken.roleArn` and `spark.hadoop.fs.oss.endpoint`, see OSS access notes.

- Sample code

  KmeansModelSaveToOss.scala

- How to commit

```
# Edit code.
val modelOssDir = "oss://bucket/kmeans-model" // Enter the path of the OSS bucket.
val spark = SparkSession
  .builder()
  .config("spark.hadoop.fs.oss.credentials.provider", "org.apache.hadoop.fs.aliyun.oss.Al
iyunStsTokenCredentialsProvider")
  .config("spark.hadoop.fs.oss.ststoken.roleArn", "acs:ram::****:role/aliyunodpsdefaultro
le")
  .config("spark.hadoop.fs.oss.endpoint", "oss-cn-hangzhou-zmf.aliyuncs.com")
  .appName("KmeansModelSaveToOss")
  .getOrCreate()
cd /path/to/MaxCompute-Spark/spark-2.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.mllib.Kmean
sModelSaveToOss \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

OSS UnstructuredData example (Scala)

For more information about how to configure `spark.hadoop.fs.oss.ststoken.roleArn` and `spark.hadoop.fs.oss.endpoint`, see OSS access notes.

- Sample code

  SparkUnstructuredDataCompute.scala

- How to commit

```
# Edit code.
val pathIn = "oss://bucket/inputdata/" // Enter the path of the OSS bucket.
val spark = SparkSession
  .builder()
  .config("spark.hadoop.fs.oss.credentials.provider", "org.apache.hadoop.fs.aliyun.oss.Al
iyunStsTokenCredentialsProvider")
  .config("spark.hadoop.fs.oss.ststoken.roleArn", "acs:ram::****:role/aliyunodpsdefaultro
le")
  .config("spark.hadoop.fs.oss.endpoint", "oss-cn-hangzhou-zmf.aliyuncs.com")
  .appName("SparkUnstructuredDataCompute")
  .getOrCreate()
cd /path/to/MaxCompute-Spark/spark-2.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.oss.SparkUn
structuredDataCompute \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

SparkPi example (Scala)

- Sample code

  SparkPi.scala

- How to commit

```
cd /path/to/MaxCompute-Spark/spark-2.x
mvn clean package
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.SparkPi \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Spark Streaming LogHub example (Scala)

- Sample code

  LogHubStreamingDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.streaming.l
oghub.LogHubStreamingDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Example of using Spark Streaming LogHub to write data to MaxCompute (Scala)

- Sample code

  LogHub2OdpsDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.streaming.l
oghub.LogHub2OdpsDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Spark Streaming DataHub example (Scala)

- Sample code

  DataHubStreamingDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.streaming.d
atahub.DataHubStreamingDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Example of using Spark Streaming DataHub to write data to MaxCompute (Scala)

- Sample code

  DataHub2OdpsDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.streaming.d
atahub.DataHub2OdpsDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Spark Streaming Kafka example (Scala)

- Sample code

  KafkaStreamingDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.streaming.k
afka.KafkaStreamingDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

> **Note** For more information, see Spark on MaxCompute.

Spark StructuredStreaming DataHub example (Scala)

- Sample code

  DatahubStructuredStreamingDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.structureds
treaming.datahub.DatahubStructuredStreamingDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Spark StructuredStreaming Kafka example (Scala)

- Sample code

  KafkaStructuredStreamingDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.structureds
treaming.kafka.KafkaStructuredStreamingDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Spark StructuredStreaming LogHub example (Scala)

- Sample code

  LoghubStructuredStreamingDemo.scala

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.structureds
treaming.loghub.LoghubStructuredStreamingDemo \
    /path/to/MaxCompute-Spark/spark-2.x/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.
jar
```

Example of using PySpark to read data from or write data to a MaxCompute table (Python)

- Sample code

  spark_sql.py

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --jars /path/to/odps-spark-datasource_2.11-3.3.8-p
ublic.jar \
    /path/to/MaxCompute-Spark/spark-2.x/src/main/python/spark_sql.py
```

Example of using PySpark to write data to OSS (Python)

- Sample code

  spark_oss.py

- How to commit

```
# For more information about how to configure the environment variables in the spark-defa
ults.conf file, see Set up a Spark on MaxCompute development environment.
# For more information about OSS configurations, see OSS access notes.
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --jars /path/to/spark-examples_2.11-1.0.0-SNAPSHOT
-shaded.jar \
    /path/to/MaxCompute-Spark/spark-2.x/src/main/python/spark_oss.py
# Compile Spark 2.x to obtain the spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar package.
```

Spark SQL example (Java)

For more information about Spark SQL Java sample code, see JavaSparkSQL.java.

Example of reading data from MaxCompute and writing the data to HBase

Use IntelliJ IDEA to write code to read data from MaxCompute and write the data to HBase.

- Sample code

```
object McToHbase {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder()
      .appName("spark_sql_ddl")
      .config("spark.sql.catalogImplementation", "odps")
      .config("spark.hadoop.odps.end.point","http://service.cn.maxcompute.aliyun.com/api"
)
      .config("spark.hadoop.odps.runtime.end.point","http://service.cn.maxcompute.aliyun-
inc.com/api")
      .getOrCreate()
      val sc = spark.sparkContext
      val config = HBaseConfiguration.create()
      val zkAddress = ""
      config.set(HConstants.ZOOKEEPER_QUORUM, zkAddress);
      val jobConf = new JobConf(config)
      jobConf.setOutputFormat(classOf[TableOutputFormat])
      jobConf.set(TableOutputFormat.OUTPUT_TABLE,"test")
    try{
      import spark._
      spark.sql("select '7', 'long'").rdd.map(row => {
        val id = row(0).asInstanceOf[String]
        val name = row(1).asInstanceOf[String]
        val put = new Put(Bytes.toBytes(id))
        put.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("a"), Bytes.toBytes(name))
        (new ImmutableBytesWritable, put)
    }).saveAsHadoopDataset(jobConf)
  } finally {
    sc.stop()
  }
  }
}
```

- How to commit: Use IntelliJ IDEA to commit and run the sample code. For more information about how to configure the Spark development environment, see Running modes of Spark on MaxCompute.

Examples of reading data from and writing data to OSS objects

Use IntelliJ IDEA or DataWorks to read and write OSS objects.

- Sample code

- Example 1: Sample code for the local mode

```
package com.aliyun.odps.spark.examples
import java.io.ByteArrayInputStream
import org.apache.spark.sql.SparkSession
object SparkOSS {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder()
      .config("spark.master", "local[4]") // The code can run after you set spark.maste
r to local[N]. N indicates the number of concurrent Spark jobs.
      .config("spark.hadoop.fs.oss.accessKeyId", "")
      .config("spark.hadoop.fs.oss.accessKeySecret", "")
      .config("spark.hadoop.fs.oss.endpoint", "oss-cn-beijing.aliyuncs.com")
      .appName("SparkOSS")
      .getOrCreate()
    val sc = spark.sparkContext
    try {
      // Read data from OSS objects.
      val pathIn = "oss://spark-oss/workline.txt"
      val inputData = sc.textFile(pathIn, 5)
            // Write Resilient Distributed Datasets (RDD).
      inputData.repartition(1).saveAsTextFile("oss://spark-oss/user/data3")
    } finally {
      sc.stop()
    }
  }
}
```

> **Note** Before you run the sample code, check whether the `hadoop-fs-oss` dependency is added. If the dependency is not added, an error is returned.

- Example 2: Sample code for the local mode

```
package com.aliyun.odps.spark.examples
import java.io.ByteArrayInputStream
import com.aliyun.oss.{OSSClientBuilder,OSSClient}
import org.apache.spark.sql.SparkSession
object SparkOSS {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder()
      .config("spark.master", "local[4]") // The code can run after you set spark.maste
r to local[N]. N indicates the number of concurrent Spark jobs.
      .config("spark.hadoop.fs.oss.accessKeyId", "")
      .config("spark.hadoop.fs.oss.accessKeySecret", "")
      .config("spark.hadoop.fs.oss.endpoint", "oss-cn-beijing.aliyuncs.com")
      .appName("SparkOSS")
      .getOrCreate()
    val sc = spark.sparkContext
    try {
      // Read data from OSS objects.
      val pathIn = "oss://spark-oss/workline.txt"
      val inputData = sc.textFile(pathIn, 5)
      val cnt = inputData.count
      inputData.count()
      println(s"count: $cnt")
      // Write data to OSS objects.
      val ossClient = new OSSClientBuilder().build("oss-cn-beijing.aliyuncs.com", "<acc
essKeyId>", "<accessKeySecret>")
      val filePath="user/data"
      ossClient.putObject("spark-oss",filePath , new ByteArrayInputStream(cnt.toString.
getBytes()))
      ossClient.shutdown()
    } finally {
      sc.stop()
    }
  }
}
```

- Example 3: Sample code for the cluster mode

```
package com.aliyun.odps.spark.examples
import java.io.ByteArrayInputStream
import com.aliyun.oss.{OSSClientBuilder,OSSClient}
import org.apache.spark.sql.SparkSession
object SparkOSS {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder()
      .appName("SparkOSS")
      .getOrCreate()
    val sc = spark.sparkContext
    try {
      // Read data from OSS objects.
      val pathIn = "oss://spark-oss/workline.txt"
      val inputData = sc.textFile(pathIn, 5)
      val cnt = inputData.count
      inputData.count()
      println(s"count: $cnt")
      // inputData.repartition(1).saveAsTextFile("oss://spark-oss/user/data3")
      // Read data from OSS objects.
      val ossClient = new OSSClientBuilder().build("oss-cn-beijing.aliyuncs.com", "<acc
essKeyId>", "<accessKeySecret>")
      val filePath="user/data"
      ossClient.putObject("spark-oss",filePath , new ByteArrayInputStream(cnt.toString.
getBytes()))
      ossClient.shutdown()
    } finally {
      sc.stop()
    }
  }
}
```

- How to commit
  - Use IntelliJ IDEA to develop, test, and commit the code for the local mode. For more information about how to configure the Spark development environment, see Running modes of Spark on MaxCompute.
  - Use an ODPS Spark node to commit and run the code in the DataWorks console. For more information, see Create a MaxCompute Spark node.

Example of reading data from MaxCompute and writing the data to OSS

Use IntelliJ IDEA or DataWorks to read MaxCompute data and write the data to OSS.

- Sample code

- Sample code for the local mode

```
package com.aliyun.odps.spark.examples.userpakage
import org.apache.spark.sql.{SaveMode, SparkSession}
object SparkODPS2OSS {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
      .builder()
      .appName("Spark2OSS")
      .config("spark.master", "local[4]")// The code can run after you set spark.master
to local[N]. N indicates the number of concurrent Spark jobs.
      .config("spark.hadoop.odps.project.name", "")
      .config("spark.hadoop.odps.access.id", "")
      .config("spark.hadoop.odps.access.key", "")
      .config("spark.hadoop.odps.end.point", "http://service.cn.maxcompute.aliyun.com/a
pi")
      .config("spark.sql.catalogImplementation", "odps")
      .config("spark.hadoop.fs.oss.accessKeyId","")
      .config("spark.hadoop.fs.oss.accessKeySecret","")
      .config("spark.hadoop.fs.oss.endpoint","oss-cn-beijing.aliyuncs.com")
      .getOrCreate()
    try{
      // Use Spark SQL to query tables.
      val data = spark.sql("select * from  user_detail")
     // Show the query results.
      data.show(10)
      // Store the query results to an OSS object.
      data.toDF().coalesce(1).write.mode(SaveMode.Overwrite).csv("oss://spark-oss/user/
data3")
    }finally {
      spark.stop()
    }
  }
}
```

- Sample code for the cluster mode

```
package com.aliyun.odps.spark.examples.userpakage
import org.apache.spark.sql.{SaveMode, SparkSession}
object SparkODPS2OSS {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
      .builder()
      .appName("SparkODPS2OSS")
      .getOrCreate()
    try{
      // Use Spark SQL to query tables.
      val data = spark.sql("select * from  user_detail")
     // Show the query results.
      data.show(10)
      // Store the query results to an OSS object.
      data.toDF().coalesce(1).write.mode(SaveMode.Overwrite).csv("oss://spark-oss/user/
data3")
    }finally {
      spark.stop()
    }
  }
}
```

- How to commit:
  - Use IntelliJ IDEA to develop, test, and commit the code for the local mode.
  - Use an ODPS Spark node to commit and run the code in the DataWorks console. For more information, see Create a MaxCompute Spark node.

> **Note** For more information about how to configure the Spark development environment, see Running modes of Spark on MaxCompute.

4.5.5. Develop a Spark on MaxCompute application by using PySpark

This topic describes how to develop a Spark on MaxCompute application by using PySpark.

If you want to access MaxCompute tables in your application, you must compile the odps-spark-datasource package. For more information, see Set up a Spark on MaxCompute development environment.

Develop a Spark SQL application in Spark 1.6

Sample code:

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import OdpsContext
if __name__ == '__main__':
    conf = SparkConf().setAppName("odps_pyspark")
    sc = SparkContext(conf=conf)
    sql_context = OdpsContext(sc)
    sql_context.sql("DROP TABLE IF EXISTS spark_sql_test_table")
    sql_context.sql("CREATE TABLE spark_sql_test_table(name STRING, num BIGINT)")
    sql_context.sql("INSERT INTO TABLE spark_sql_test_table SELECT 'abc', 100000")
    sql_context.sql("SELECT * FROM spark_sql_test_table").show()
    sql_context.sql("SELECT COUNT(*) FROM spark_sql_test_table").show()
```

Run the following command to commit and run the code:

```
./bin/spark-submit \
--jars cupid/odps-spark-datasource_xxx.jar \
example.py
```

Develop a Spark SQL application in Spark 2.3

Sample code:

```
from pyspark.sql import SparkSession
if __name__ == '__main__':
    spark = SparkSession.builder.appName("spark sql").getOrCreate()
    spark.sql("DROP TABLE IF EXISTS spark_sql_test_table")
    spark.sql("CREATE TABLE spark_sql_test_table(name STRING, num BIGINT)")
    spark.sql("INSERT INTO spark_sql_test_table SELECT 'abc', 100000")
    spark.sql("SELECT * FROM spark_sql_test_table").show()
    spark.sql("SELECT COUNT(*) FROM spark_sql_test_table").show()
```

Commit and run the code.

- Run the following command to commit and run the code in cluster mode:

  ```
  spark-submit --master yarn-cluster \
  --jars cupid/odps-spark-datasource_xxx.jar \
  example.py
  ```

- Run the following command to commit and run the code in local mode:

  ```
  cd $SPARK_HOME
  ./bin/spark-submit --master local[4] \
  --driver-class-path cupid/odps-spark-datasource_xxx.jar \
  /path/to/odps-spark-examples/spark-examples/src/main/python/spark_sql.py
  ```

  > **Note**
  > - If Spark runs in local mode, MaxCompute Tunnel is required for accessing MaxCompute tables.
  > - If Spark runs in local mode, you must use the **--driver-class-path** option instead of the **--jars** option.

Upload required packages

A MaxCompute cluster does not allow you to install Python libraries. If your Spark on MaxCompute application depends on Python libraries, plug-ins, or projects, you can package the required resources on your on-premises machine and run the spark-submit script to upload the packages to MaxCompute. For some special resources, the Python version on your on-premises machine that you use to package the resources must be the same as the Python version in the MaxCompute cluster in which your application runs. Select one of the following methods to package resources for your application based on the complexity of your business.

- Use public resources without packaging
  - Use public resources for Python 2.7.13.

```
spark.hadoop.odps.cupid.resources = public.python-2.7.13-ucs4.tar.gz
spark.pyspark.python = ./public.python-2.7.13-ucs4.tar.gz/python-2.7.13-ucs4/bin/python
```

The following third-party libraries are available:

```
$./bin/pip list
Package                       Version
----------------------------- -----------
absl-py                       0.11.0
aenum                         2.2.4
asn1crypto                    0.23.0
astor                         0.8.1
astroid                       1.6.1
atomicwrites                  1.4.0
attrs                         20.3.0
backports.functools-lru-cache 1.6.1
backports.lzma                0.0.14
backports.weakref             1.0.post1
beautifulsoup4                4.9.3
bleach                        2.1.2
boto                          2.49.0
boto3                         1.9.147
botocore                      1.12.147
bz2file                       0.98
cachetools                    3.1.1
category-encoders             2.2.2
certifi                       2019.9.11
cffi                          1.11.2
chardet                       3.0.4
click                         6.7
click-plugins                 1.1.1
cligj                         0.7.0
cloudpickle                   0.5.3
configparser                  4.0.2
contextlib2                   0.6.0.post1
cryptography                  2.6.1
cssutils                      1.0.2
cycler                        0.10.0
Cython                        0.29.5
dask                          0.18.1
DBUtils                       1.2
decorator                     4.2.1
```

docutils	0.16
entrypoints	0.2.3
enum34	1.1.10
fake-useragent	0.1.11
Fiona	1.8.17
funcsigs	1.0.2
functools32	3.2.3.post2
future	0.16.0
futures	3.3.0
gast	0.2.2
gensim	3.8.3
geopandas	0.6.3
getpass3	1.2
google-auth	1.23.0
google-auth-oauthlib	0.4.1
google-pasta	0.2.0
grpcio	1.33.2
h5py	2.7.0
happybase	1.1.0
html5lib	1.0.1
idna	2.10
imbalanced-learn	0.4.3
imblearn	0.0
importlib-metadata	2.0.0
ipaddress	1.0.23
ipython-genutils	0.2.0
isort	4.3.4
itchat	1.3.10
itsdangerous	0.24
jedi	0.11.1
jieba	0.42.1
Jinja2	2.10
jmespath	0.10.0
jsonschema	2.6.0
kafka-python	1.4.6
kazoo	2.5.0
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.2
kiwisolver	1.1.0
lazy-object-proxy	1.3.1
libarchive-c	2.8
lightgbm	2.3.1
lml	0.0.2
lxml	4.2.1
MarkupSafe	1.0
matplotlib	2.2.5
mccabe	0.6.1
missingno	0.4.2
mistune	0.8.3
mock	2.0.0
more-itertools	5.0.0
munch	2.5.0
nbconvert	5.3.1
nbformat	4.4.0

networkx	2.1
nose	1.3.7
numpy	1.16.1
oauthlib	3.1.0
opt-einsum	2.3.2
packaging	20.4
pandas	0.24.2
pandocfilters	1.4.2
parso	0.1.1
pathlib2	2.3.5
patsy	0.5.1
pbr	3.1.1
pexpect	4.4.0
phpserialize	1.3
pickleshare	0.7.4
Pillow	6.2.0
pip	20.2.4
pluggy	0.13.1
ply	3.11
prompt-toolkit	2.0.1
protobuf	3.6.1
psutil	5.4.3
psycopg2	2.8.6
ptyprocess	0.5.2
py	1.9.0
py4j	0.10.6
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycosat	0.6.3
pycparser	2.18
pydot	1.4.1
Pygments	2.2.0
pykafka	2.8.0
pylint	1.8.2
pymongo	3.11.0
PyMySQL	0.10.1
pynliner	0.8.0
pyodps	0.9.3.1
pyOpenSSL	17.5.0
pyparsing	2.2.0
pypng	0.0.20
pyproj	2.2.2
PyQRCode	1.2.1
pytest	4.6.11
python-dateutil	2.8.1
pytz	2020.4
PyWavelets	0.5.2
PyYAML	3.12
redis	3.2.1
requests	2.25.0
requests-oauthlib	1.3.0
rope	0.10.7
rsa	4.5
ruamel.ordereddict	0.4.15

```
ruamel.yaml                   0.11.14
s3transfer                    0.2.0
scandir                       1.10.0
scikit-image                  0.14.0
scikit-learn                  0.20.3
scipy                         1.2.3
seaborn                       0.9.1
Send2Trash                    1.5.0
setuptools                    41.0.0
Shapely                       1.7.1
simplegeneric                 0.8.1
singledispatch                3.4.0.3
six                           1.15.0
sklearn2                      0.0.13
smart-open                    1.8.1
soupsieve                     1.9.6
SQLAlchemy                    1.3.20
statsmodels                   0.11.0
subprocess32                  3.5.4
tabulate                      0.8.7
tensorflow                    2.0.0
tensorflow-estimator          2.0.1
termcolor                     1.1.0
testpath                      0.3.1
thriftpy                      0.3.9
timeout-decorator             0.4.1
toolz                         0.9.0
tqdm                          4.32.2
traitlets                     4.3.2
urllib3                       1.24.3
wcwidth                       0.2.5
webencodings                  0.5.1
Werkzeug                      1.0.1
wheel                         0.35.1
wrapt                         1.11.1
xgboost                       0.82
xlrd                          1.2.0
XlsxWriter                    1.0.7
zipp                          1.2.0
```

- Use public resources for Python 3.7.9.

```
spark.hadoop.odps.cupid.resources = public.python-3.7.9-ucs4.tar.gz
spark.pyspark.python = ./public.python-3.7.9-ucs4.tar.gz/python-3.7.9-ucs4/bin/python3
```

The following third-party libraries are available:

```
Package                       Version
----------------------------- -----------
appnope                       0.1.0
asn1crypto                    0.23.0
astroid                       1.6.1
attrs                         20.3.0
autopep8                      1.3.4
backcall                      0.2.0
```

```
backports.functools-lru-cache 1.5
backports.weakref             1.0rc1
beautifulsoup4                4.6.0
bidict                        0.17.3
bleach                        2.1.2
boto                          2.49.0
boto3                         1.9.147
botocore                      1.12.147
bs4                           0.0.1
bz2file                       0.98
cached-property               1.5.2
cachetools                    3.1.1
category-encoders             2.2.2
certifi                       2019.11.28
cffi                          1.11.2
chardet                       3.0.4
click                         6.7
click-plugins                 1.1.1
cligj                         0.7.0
cloudpickle                   0.5.3
cryptography                  2.6.1
cssutils                      1.0.2
cycler                        0.10.0
Cython                        0.29.21
dask                          0.18.1
DBUtils                       1.2
decorator                     4.2.1
docutils                      0.16
entrypoints                   0.2.3
fake-useragent                0.1.11
Fiona                         1.8.17
future                        0.16.0
gensim                        3.8.3
geopandas                     0.8.0
getpass3                      1.2
h5py                          3.1.0
happybase                     1.1.0
html5lib                      1.0.1
idna                          2.10
imbalanced-learn              0.4.3
imblearn                      0.0
importlib-metadata            2.0.0
iniconfig                     1.1.1
ipykernel                     5.3.4
ipython                       7.19.0
ipython-genutils              0.2.0
isort                         4.3.4
itchat                        1.3.10
itsdangerous                  0.24
jedi                          0.11.1
jieba                         0.42.1
Jinja2                        2.10
jmespath                      0.10.0
jsonschema                    2.6.0
```

jupyter-client	6.1.7
jupyter-core	4.6.3
kafka-python	1.4.6
kazoo	2.5.0
kiwisolver	1.3.1
lazy-object-proxy	1.3.1
libarchive-c	2.8
lightgbm	2.3.1
lml	0.0.2
lxml	4.2.1
Mako	1.0.10
MarkupSafe	1.0
matplotlib	3.3.3
mccabe	0.6.1
missingno	0.4.2
mistune	0.8.3
mock	2.0.0
munch	2.5.0
nbconvert	5.3.1
nbformat	4.4.0
networkx	2.1
nose	1.3.7
numpy	1.19.4
packaging	20.4
pandas	1.1.4
pandocfilters	1.4.2
parso	0.1.1
patsy	0.5.1
pbr	3.1.1
pexpect	4.4.0
phpserialize	1.3
pickleshare	0.7.4
Pillow	6.2.0
pip	20.2.4
plotly	4.12.0
pluggy	0.13.1
ply	3.11
prompt-toolkit	2.0.1
protobuf	3.6.1
psutil	5.4.3
psycopg2	2.8.6
ptyprocess	0.5.2
py	1.9.0
py4j	0.10.6
pycodestyle	2.3.1
pycosat	0.6.3
pycparser	2.18
pydot	1.4.1
Pygments	2.2.0
pykafka	2.8.0
pylint	1.8.2
pymongo	3.11.0
PyMySQL	0.10.1
pynliner	0.8.0

pyodps	0.9.3.1
pyOpenSSL	17.5.0
pyparsing	2.2.0
pypng	0.0.20
pyproj	3.0.0.post1
PyQRCode	1.2.1
pytest	6.1.2
python-dateutil	2.8.1
pytz	2020.4
PyWavelets	0.5.2
PyYAML	3.12
pyzmq	17.0.0
qtconsole	4.3.1
redis	3.2.1
requests	2.25.0
retrying	1.3.3
rope	0.10.7
ruamel.yaml	0.16.12
ruamel.yaml.clib	0.2.2
s3transfer	0.2.0
scikit-image	0.14.0
scikit-learn	0.20.3
scipy	1.5.4
seaborn	0.11.0
Send2Trash	1.5.0
setuptools	41.0.0
Shapely	1.7.1
simplegeneric	0.8.1
six	1.15.0
sklearn2	0.0.13
smart-open	1.8.1
SQLAlchemy	1.3.20
statsmodels	0.12.1
tabulate	0.8.7
testpath	0.3.1
thriftpy	0.3.9
timeout-decorator	0.4.1
toml	0.10.2
toolz	0.9.0
tornado	6.1
tqdm	4.32.2
traitlets	4.3.2
urllib3	1.24.3
wcwidth	0.2.5
webencodings	0.5.1
wheel	0.35.1
wrapt	1.11.1
xgboost	1.2.1
xlrd	1.2.0
XlsxWriter	1.0.7
zipp	3.4.0

- Upload a single wheel package

If a few Python dependencies are required, you can upload only one wheel package. In most cases, a manylinux wheel package is uploaded. To upload a single wheel package, perform the following steps:

i. Package the wheel package into a ZIP package. For example, package the pymysql.whl package into the pymysql.zip package.

ii. Upload the ZIP package and store the package with the storage class of Archive.

iii. Select the ZIP package on the Spark node page in the DataWorks console.

iv. Change environment variables in the code to import the ZIP package.

```
sys.path.append('pymysql')
import pymysql
```

- Use a requirements.txt file

  If a large number of additional dependencies are required, you must repeat the procedure for uploading a wheel package several times. In this case, you can download the script and create a requirements.txt file that lists the required dependencies and use the script and the requirements.txt file to generate a Python environment package. This way, you can develop a Spark on MaxCompute application by using PySpark.

  - **Usage**

    ```
    $ chmod +x generate_env_pyspark.sh
    $ generate_env_pyspark.sh -h
    Usage:
    generate_env_pyspark.sh [-p] [-r] [-t] [-c] [-h]
    Description:
    -p ARG, the version of python, currently supports python 2.7, 3.5, 3.6 and 3.7 versions
    .
    -r ARG, the local path of your python requirements.
    -t ARG, the output directory of the gz compressed package.
    -c, clean mode, we will only package python according to your requirements, without oth
    er pre-provided dependencies.
    -h, display help of this script.
    ```

  - **Example**

    ```
    # Generate a Python environment package with preinstalled dependencies.
    $ generate_env_pyspark.sh -p 3.7 -r your_path_to_requirements -t your_output_directory
    # Generate a Python environment package in clean mode. This way, the package that you g
    enerate does not contain preinstalled dependencies.
    generate_env_pyspark.sh -p 3.7 -r your_path_to_requirements -t your_output_directory -c
    ```

- Notes
  - The script can run on macOS or Linux. To use the script, you must install Docker in advance. For more information about how to install Docker, see Docker documentation.
  - The following Python versions are supported: Python 2.7, Python 3.5, Python 3.6, and Python 3.7. If you do not have special requirements for the Python version, we recommend that you use Python 3.7.
  - The `-c` option specifies whether to enable the clean mode. In clean mode, preinstalled dependencies cannot be packaged. Therefore, the Python package is small in size. For more information about the dependencies of each Python version, see Preinstalled dependencies of Python 2.7, Preinstalled dependencies of Python 3.5, Preinstalled dependencies of Python 3.6, and Preinstalled dependencies of Python 3.7.
  - MaxCompute allows you to upload resource packages with a maximum size of 500 MB. If most preinstalled dependencies are not used, we recommend that you use the `-c` option to package resources in clean mode.
- Use packages in Spark

  You can use the generate_env_pyspark.sh script to generate a .tar.gz package of a specified Python version in a specified path. The `-t` option specifies the path, and the `-p` option specifies the Python version. For example, if Python 3.7 is used, the py37.tar.gz package is generated. You can upload the package by using the MaxCompute client or a MaxCompute SDK, and store the package with the storage class of Archive. For more information about how to manage resources, see Resource operations. To use the MaxCompute client to upload the py37.tar.gz package, perform the following steps:

  a. Run the following command on the MaxCompute client to add the package:

  ```
  add archive /your/path/to/py37.tar.gz -f;
  ```

  b. Add the following parameters to the configurations of your Spark job.

  ```
  spark.hadoop.odps.cupid.resources = your_project.py37.tar.gz
  spark.pyspark.python = your_project.py37.tar.gz/bin/python
  ```

  If the preceding parameters do not take effect, you also need to add the following parameters to the configurations of your Spark job. For example, if you use Apache Zeppelin to debug your Spark job, you need to add the following Python environment configurations to an Apache Zeppelin notebook.

  ```
  spark.yarn.appMasterEnv.PYTHONPATH = ./your_project.py37.tar.gz/bin/python
  spark.executorEnv.PYTHONPATH = ./your_project.py37.tar.gz/bin/python
  ```

- Use a Docker container

  This method is suitable for the following scenarios:

  - If you want to package the .so file, you cannot use the method described in "Upload a single wheel package" or run `pip install`.
  - The Python version that you use is not Python 2.7, Python 3.5, Python 3.6, or Python 3.7.

  In the preceding scenarios, you must make sure that the Python environment that you package is the same as the Python environment in which your Spark job is running. For example, the Python environment that you package in macOS may not be compatible with the Python environment in which your Spark job is running. To package the Python 3.7 environment by using a Docker container,

perform the following steps:

i. Create a Dockerfile on your Docker host.

- Sample code in Python 3:

```
FROM centos:7.6.1810
RUN set -ex \
    # Preinstall the required components.
    && yum install -y wget tar libffi-devel zlib-devel bzip2-devel openssl-devel nc
urses-devel sqlite-devel readline-devel tk-devel gcc make initscripts zip\
    && wget https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz \
    && tar -zxvf Python-3.7.0.tgz \
    && cd Python-3.7.0 \
    && ./configure prefix=/usr/local/python3 \
    && make \
    && make install \
    && make clean \
    && rm -rf /Python-3.7.0* \
    && yum install -y epel-release \
    && yum install -y python-pip
# Set the default Python version to Python 3.
RUN set -ex \
    # Back up resources of Python 2.7.
    && mv /usr/bin/python /usr/bin/python27 \
    && mv /usr/bin/pip /usr/bin/pip-python27 \
    # Set the default Python version to Python 3.
    && ln -s /usr/local/python3/bin/python3.7 /usr/bin/python \
    && ln -s /usr/local/python3/bin/pip3 /usr/bin/pip
# Fix the YUM bug that is caused by the change in the Python version.
RUN set -ex \
    && sed -i "s#/usr/bin/python#/usr/bin/python27#" /usr/bin/yum \
    && sed -i "s#/usr/bin/python#/usr/bin/python27#" /usr/libexec/urlgrabber-ext-do
wn \
    && yum install -y deltarpm
# Update the pip version.
RUN pip install --upgrade pip
```

- Sample code in Python 3:

```
FROM centos:7.6.1810
RUN set -ex \
    # Preinstall the required components.
    && yum install -y wget tar libffi-devel zlib-devel bzip2-devel openssl-devel nc
urses-devel sqlite-devel readline-devel tk-devel gcc make initscripts zip\
    && wget https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tgz \
    && tar -zxvf Python-2.7.18.tgz \
    && cd Python-2.7.18 \
    && ./configure prefix=/usr/local/python2 \
    && make \
    && make install \
    && make clean \
    && rm -rf /Python-2.7.18*
# Set the default Python version.
RUN set -ex \
    && mv /usr/bin/python /usr/bin/python27 \
    && ln -s /usr/local/python2/bin/python /usr/bin/python
RUN set -ex \
    && wget https://bootstrap.pypa.io/get-pip.py \
    && python get-pip.py
RUN set -ex \
    && rm -rf /usr/bin/pip \
    && ln -s /usr/local/python2/bin/pip /usr/bin/pip
# Fix the YUM bug that is caused by the change in the Python version.
RUN set -ex \
    && sed -i "s#/usr/bin/python#/usr/bin/python27#" /usr/bin/yum \
    && sed -i "s#/usr/bin/python#/usr/bin/python27#" /usr/libexec/urlgrabber-ext-do
wn \
    && yum install -y deltarpm
# Update the pip version.
RUN pip install --upgrade pip
```

ii. Build an image and run the Docker container.

```
# Run the following commands in the path in which the Dockerfile is stored:
docker build -t python-centos:3.7
docker run -itd --name python3.7 python-centos:3.7
```

iii. Install the required Python libraries in the container.

```
docker attach python3.7
pip install [Required library]
```

iv. Package the Python environment.

```
cd /usr/local/
zip -r python3.7.zip python3/
```

v. Copy the Python environment package from the container to your host.

```
# Exit the container.
ctrl+P+Q
# Run the following command on your host to copy the Python environment package:
docker cp python3.7:/usr/local/python3.7.zip
```

vi. Upload the Python3.7.zip package as a MaxCompute resource. You can use DataWorks to upload a package with a maximum size of 50 MB. If the size of a package exceeds 50 MB, you can use the MaxCompute client to upload the package and store the package with the storage class of Archive. For more information about how to upload resources, see Add resources.

```
add archive /path/to/python3.7.zip -f;
```

vii. When you submit a job, you need only to add the following configurations to the spark-defaults.conf file or DataWorks configurations.

```
spark.hadoop.odps.cupid.resources=[Project name].python3.7.zip
spark.pyspark.python=./[Project name].python3.7.zip/python3/bin/python3.7
```

> **Note** When you package resources in a Docker container, you need to manually place the .so package in the Python environment if the .so package cannot be found. In most cases, you can find the .so package in the container. After you find this package, add the following environment variables to the configurations of your Spark job:
>
> ```
> spark.executorEnv.LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./[Project name].python3.7.zip
> /python3/[Directory of the created .so package]
> spark.yarn.appMasterEnv.LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./[Project name].python3
> .7.zip/python3/[Directory of the created .so package]
> ```

- Reference a custom Python package

  In most cases, you need to use custom Python files. To prevent the workload caused by uploading multiple files in sequence, you can package these files by performing the following steps:

  i. Create an empty file named __init__.py and package code into a ZIP package.

  ii. Upload the ZIP package as a MaxCompute resource and rename the package. This package is decompressed into the working directory.

  iii. Configure the parameter `spark.executorEnv.PYTHONPATH=` .

  iv. Import the main Python file to the path that is specified by spark.executorEnv.PYTHONPATH=.

4.5.6. Access instances in a VPC from Spark on MaxCompute

This topic describes how to access instances in an Alibaba Cloud Virtual Private Cloud (VPC) from Spark on MaxCompute.

Directly access instances in a VPC

You can access instances in an Alibaba Cloud VPC or custom private domain names from Spark on MaxCompute. Instances in an Alibaba Cloud VPC include Elastic Compute Service (ECS) instances, ApsaraDB for HBase instances, and ApsaraDB RDS instances.

When you access instances in a VPC from Spark on MaxCompute, add the `spark.hadoop.odps.cupid.vpc.domain.list` parameter to the *spark-defaults.conf* file or the *DataWorks* file to specify one or more instances. The value of this parameter is in the JSON format. When you configure this parameter, you must remove spaces and line feeds from the text in this parameter and merge JSON text into one line.

The following examples show the configurations of the `spark.hadoop.odps.cupid.vpc.domain.list` parameter when you access different instances. You must use the actual values to replace the values of the regionId, vpcId, domain, and port parameters in the following examples. For information about the ID of each region, see Project operations.

> **Notice**
>
> - You must add the classless inter-domain routing (CIDR) block 100.104.0.0/16 to the whitelist of the instance that you want to access.
> - If the regionId parameter is set to *cn-shanghai* or *cn-beijing*, you must set `spark.hadoop.odps.cupid.smartnat.enable` to true.
> - You can access only the services in one VPC in the current region from Spark on MaxCompute.
> - You do not need to set `spark.hadoop.odps.cupid.eni.enable` to true.

Access instances in a VPC by using an ENI

Compared with the direct access method described in Directly access instances in a VPC, this access method provides high stability and better performance. In addition, this access method supports Internet access.

When you use this access method, take note of the following points:

- You can use this access method to access instances in a VPC. If your Spark job needs to access instances across multiple VPCs at the same time, you can establish connections between the VPC that you have accessed and other VPCs.
- For a Spark job that runs in a MaxCompute project, the user ID (UID) must be the same for the Alibaba Cloud account that owns the MaxCompute project and the Alibaba Cloud account that owns the VPC. Otherwise, the following error message appears: `You are not allowed to use this vpc - vpc owner and project owner must be the same person`.

To access instances in a VPC by using an ENI, perform the following steps:

1. Provide the following information of the VPC that you want to access to the MaxCompute technical support team.
   - Region ID: the ID of the region in which the VPC is deployed, such as cn-beijing.
   - UID: the UID of the Alibaba Cloud account that owns the MaxCompute project. It is also the UID of the Alibaba Cloud account that owns the VPC that you want to access.
   - VPC ID: the ID of the VPC that you want to access by using an ENI.
   - vSwitch ID: the ID of the vSwitch that belongs to the VPC. You can go to the VPC console to query the ID. If multiple vSwitch IDs are displayed in the VPC console, select one from these IDs.
   - Security group ID: the ID of the security group that belongs to the VPC. To access instances in a VPC from Spark on MaxCompute, you must create a security group for the VPC to control access requests. For more information about how to create a security group, see Create a security group.

2. Grant Spark on MaxCompute the permissions to create an ENI in the VPC.

   After the permissions are granted, you can create an ENI in the VPC to allow access from Spark on MaxCompute. To grant the permissions, you can use an Alibaba Cloud account to log on to the Resource Access Management (RAM) console and click Grants in the left-side navigation pane.

3. Wait for the MaxCompute technical support team to enable the ENI.
4. Add security group rules.

   After the ENI is enabled, you must add rules to the security group provided in Step 1. These rules specify the ports that you can use to access the instances from Spark on MaxCompute, such as ports 9200 and 31000.

   For example, if you want to access an ApsaraDB RDS instance, you must add such rules to the security group provided in Step 1. If such rules cannot be added to the security group and only the rules that specify IP addresses can be added, the rules that you want to add must include the CIDR block of the vSwitch provided in Step 1.

5. Configure your Spark job.

   To run your Spark job, you must add the following configurations to access the instances in the VPC by using the ENI.

   ```
   spark.hadoop.odps.cupid.eni.enable = true
   spark.hadoop.odps.cupid.eni.info = regionid:vpc id
   ```

   In this case, the following configurations are not required:

   ```
   spark.hadoop.odps.cupid.vpc.domain.list
   spark.hadoop.odps.cupid.smartnat.enable
   spark.hadoop.odps.cupid.pvtz.rolearn   # (Used to access a custom domain name)
   spark.hadoop.odps.cupid.vpc.usepvtz    # (Used to access a custom domain name)
   ```

Example 1: Access an ApsaraDB for MongoDB instance

The following code shows the value of `spark.hadoop.odps.cupid.vpc.domain.list` when you access an ApsaraDB for MongoDB instance. In this example, ApsaraDB for MongoDB has a primary instance and a secondary instance.

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "vpcId":"vpc-2zeaeq21mb1dmkqh0****",
      "zones":[
        {
          "urls":[
            {
              "domain":"dds-2ze3230cfea08****.mongodb.rds.aliyuncs.com",
              "port": 3717
            },
            {
              "domain":"dds-2ze3230cfea08****.mongodb.rds.aliyuncs.com",
              "port":3717
            }
          ]
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"vpcId":"vpc-2zeaeq21mb1dmkqh0****","zones":[{"urls":[{"d
omain":"dds-2ze3230cfea08****.mongodb.rds.aliyuncs.com","port": 3717},{"domain":"dds-2ze323
0cfea08****.mongodb.rds.aliyuncs.com","port":3717}]}]}]}
```

Example 2: Access an ApsaraDB RDS instance

The following code shows the value of `spark.hadoop.odps.cupid.vpc.domain.list` when you access an ApsaraDB RDS instance.

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "vpcId":"vpc-2zeaeq21mb1dmkqh0****",
      "zones":[
        {
          "urls":[
            {
              "domain":"rm-2zem49k73c54z****.mysql.rds.aliyuncs.com",
              "port": 3306
            }
          ]
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"vpcId":"vpc-2zeaeq21mb1dmkqh0****","zones":[{"urls":[{"d
omain":"rm-2zem49k73c54z****.mysql.rds.aliyuncs.com","port": 3306}]}]}]}
```

Example 3: Access an ApsaraDB for HBase instance

The following code shows the value of `spark.hadoop.odps.cupid.vpc.domain.list` when you access an ApsaraDB for HBase instance.

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "vpcId":"vpc-2zeaeq21mb1dmkqh0exox",
      "zones":[
        {
          "urls":[
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":2181
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":16000
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":16020
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":2181
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":16000
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":16020
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":2181
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":16000
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com",
              "port":16020
            },
```

```
            {
              "domain":"hb-2zecxg2ltnpeg8me4-cor*-***.hbase.rds.aliyuncs.com",
              "port":16020
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-cor*-***.hbase.rds.aliyuncs.com",
              "port":16020
            },
            {
              "domain":"hb-2zecxg2ltnpeg8me4-cor*-***.hbase.rds.aliyuncs.com",
              "port":16020
            }
          ]
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"vpcId":"vpc-2zeaeq21mb1dmkqh0exox","zones":[{"urls":[{"d
omain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com","port":2181},{"domain":"hb
-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com","port":16000},{"domain":"hb-2zecxg2l
tnpeg8me4-master*-***.hbase.rds.aliyuncs.com","port":16020},{"domain":"hb-2zecxg2ltnpeg8me4
-master*-***.hbase.rds.aliyuncs.com","port":2181},{"domain":"hb-2zecxg2ltnpeg8me4-master*-*
**.hbase.rds.aliyuncs.com","port":16000},{"domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.
rds.aliyuncs.com","port":16020},{"domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyu
ncs.com","port":2181},{"domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com","
port":16000},{"domain":"hb-2zecxg2ltnpeg8me4-master*-***.hbase.rds.aliyuncs.com","port":160
20},{"domain":"hb-2zecxg2ltnpeg8me4-cor*-***.hbase.rds.aliyuncs.com","port":16020},{"domain
":"hb-2zecxg2ltnpeg8me4-cor*-***.hbase.rds.aliyuncs.com","port":16020},{"domain":"hb-2zecxg
2ltnpeg8me4-cor*-***.hbase.rds.aliyuncs.com","port":16020}]}]}]}
```

Example 4: Access an ApsaraDB for Redis instance

The following code shows the value of `spark.hadoop.odps.cupid.vpc.domain.list` when you access an ApsaraDB for Redis instance.

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "vpcId":"vpc-2zeaeq21mb1dmkqh0****",
      "zones":[
        {
          "urls":[
            {
              "domain":"r-2zebda0d3c05****.redis.rds.aliyuncs.com",
              "port":3717
            }
          ]
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"vpcId":"vpc-2zeaeq21mb1dmkqh0****","zones":[{"urls":[{"d
omain":"r-2zebda0d3c05****.redis.rds.aliyuncs.com","port":3717}]}]}]}
```

Example 5: Access a LogHub instance

The following code shows the value of `spark.hadoop. odps.cupid.vpc.domain.list` when you access a LogHub instance.

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "zones":[
        {
          "urls":[
            {
              "domain":"cn-beijing-intranet.log.aliyuncs.com",
              "port":80
            }
          ]
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"zones":[{"urls":[{"domain":"cn-beijing-intranet.log.aliy
uncs.com","port":80}]}]}]}
```

Set the domain parameter to the classic network endpoint or the VPC endpoint of the LogHub instance. For the endpoint of each region, see Endpoints.

Example 6: Access a DataHub instance

The following code shows the value of `spark.hadoop.odps.cupid.vpc.domain.list` when you access a DataHub instance.

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "zones":[
        {
          "urls":[
            {
              "domain":"dh-cn-beijing.aliyun-inc.com",
              "port":80
            }
          ]
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"zones":[{"urls":[{"domain":"dh-cn-beijing.aliyun-inc.com
","port":80}]}]}]}
```

Set the domain parameter to the ECS endpoint on the classic network.

Example 7: Access a custom domain name

In this example, you use the custom domain name `example.aliyundoc.com` in a VPC and initiate a request to access this domain name from Spark on MaxCompute by using `example.aliyundoc.com:80`. 80 indicates the port number. Perform the following operations before you access the domain name:

1. Associate a zone with a VPC in PrivateZone.
2. On the Cloud Resource Access Authorization page in the RAM console, click Confirm Authorization Policy to grant MaxCompute the read-only permissions on PrivateZone.
3. Add the following parameters to the configurations of the Spark node:

   ```
   spark.hadoop.odps.cupid.pvtz.rolearn=acs:ram::xxxxxxxxxxx:role/aliyunodpsdefaultrole
   spark.hadoop.odps.cupid.vpc.usepvtz=true
   ```

   The `spark.hadoop.odps.cupid.pvtz.rolearn` parameter specifies the Alibaba Cloud Resource Name (ARN), which can be obtained from the RAM console.
4. Add the `spark.hadoop.odps.cupid.vpc.domain.list` parameter to the configuration file of your Spark job. The following code shows the value of this parameter:

```
{
  "regionId":"cn-beijing",
  "vpcs":[
    {
      "vpcId":"vpc-2zeaeq21mb1dmkqh0****",
      "zones":[
        {
          "urls":[
            {
              "domain":"example.aliyundoc.com",
              "port":80
            }
          ],
          "zoneId":"9b7ce89c6a6090e114e0f7c415ed****"
        }
      ]
    }
  ]
}
```

Results of merging JSON text into one line:

```
{"regionId":"cn-beijing","vpcs":[{"vpcId":"vpc-2zeaeq21mb1dmkqh0****","zones":[{"urls":
[{"domain":"example.aliyundoc.com","port":80}],"zoneId":"9b7ce89c6a6090e114e0f7c415ed**
**"}]}]}
```

Example 8: Access an HDFS instance

- To enable HDFS support, add the *hdfs-site.xml* file. Sample configurations in the file:

```
<?xml version="1.0"?>
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>dfs://DfsMountpointDomainName:10290</value>
    </property>
    <property>
        <name>fs.dfs.impl</name>
        <value>com.alibaba.dfs.DistributedFileSystem</value>
    </property>
    <property>
        <name>fs.AbstractFileSystem.dfs.impl</name>
        <value>com.alibaba.dfs.DFS</value>
    </property>
</configuration>
```

- Add the `spark.hadoop.odps.cupid.vpc.domain.list` parameter to the configuration file of your Spark job. The following code shows the value of this parameter:

```
{
    "regionId": "cn-shanghai",
    "vpcs": [{
        "vpcId": "vpc-xxxxxx",
        "zones": [{
            "urls": [{
                "domain": "DfsMountpointDomainName",
                "port": 10290
            }]
        }]
    }]
}
```

Results of merging JSON text into one line:

```
{"regionId": "cn-shanghai","vpcs": [{"vpcId": "vpc-xxxxxx","zones": [{"urls": [{"domain":
"DfsMountpointDomainName","port": 10290}]}]}]}
```

4.5.7. Configure Spark on MaxCompute to access OSS resources

This topic describes how to configure Spark on MaxCompute to access Object Storage Service (OSS) resources.

Configure the OSS endpoint

Use the public endpoint of OSS in the target region when you debug features. Use the OSS endpoint in the virtual private cloud (VPC) if MaxCompute is deployed in a production environment. For more information, see Regions and endpoints.

Configure the OSS access method

- Access OSS by using the AccessKey ID and AccessKey secret of an account.

```
spark.hadoop.fs.oss.accessKeyId = xxxxxx
spark.hadoop.fs.oss.accessKeySecret = xxxxxx
spark.hadoop.fs.oss.endpoint = oss-xxxxxx-internal.aliyuncs.com
```

- Access OSS by using a Security Token Service (STS) token.

  Spark on MaxCompute can access OSS by using the AccessKey ID and AccessKey secret of an account. In this case, you must write the plaintext AccessKey ID and AccessKey secret in the configuration, which incurs security risks. We recommend that you configure Spark on MaxCompute to access OSS by using an STS token.

  i. Go to the Cloud Resource Access Authorization page and click Confirm Authorization Policy. Then, the MaxCompute project can access OSS resources of the current Alibaba Cloud account by using an STS token.

  > **Note** You can authorize a MaxCompute project to access OSS resources by using this method only when the owner of the MaxCompute project is an Alibaba Cloud account that owns the OSS resources to be accessed.

ii. Obtain the Alibaba Cloud Resource Name (ARN) of the role that Spark on MaxCompute assumes.
   a. Log on to the RAM console.
   b. In the left-side navigation pane, click **RAM Roles**.
   c. On the **RAM Roles** page, search for AliyunODPSDefaultRole.
   d. In the search result, click **AliyunODPSDefaultRole** in the RAM Role Name column. On the page that appears, obtain the value of **ARN** in the **Basic Information** section. The value is in the `acs:ram::xxxxxxxxxxxxxxx:role/aliyunodpsdefaultrole` format.

iii. Add the following content to the configurations of Spark on MaxCompute:

```
# Configure Spark on MaxCompute to access OSS resources by using an STS token.
spark.hadoop.fs.oss.credentials.provider=org.apache.hadoop.fs.aliyun.oss.AliyunStsTok
enCredentialsProvider
# Configure the ARN of the role that Spark on MaxCompute assumes.
spark.hadoop.fs.oss.ststoken.roleArn=acs:ram::xxxxxxxxxxxxxxx:role/aliyunodpsdefaultr
ole
# Configure the OSS endpoint in the VPC.
spark.hadoop.fs.oss.endpoint=oss-cn-hangzhou-internal.aliyuncs.com
```

Configure a whitelist

```
spark.hadoop.odps.cupid.vpc.domain.list = {"regionId":"xxxxxx", "vpcs":[{"zones":[{"urls":[
{ "domain":"bucketname.oss-xxxxxx-internal.aliyuncs.com", "port":80} ] }]}]}
```

4.5.8. Check jobs

This topic describes how to check whether a job is submitted and executed based on job logs. MaxCompute provides both Logview and Spark Web UI for you to check jobs.

Context

A job is submitted by using spark-submit. Logs are also generated when you use DataWorks to execute Spark jobs. Example:

```
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class  SparkPi /tmp/spark-2.x-demo/target/AliSpark
-2.x-quickstart-1.0-SNAPSHOT-shaded.jar
```

After a job is submitted, MaxCompute creates an instance and displays the Logview and tracking URLs of the instance in a log.

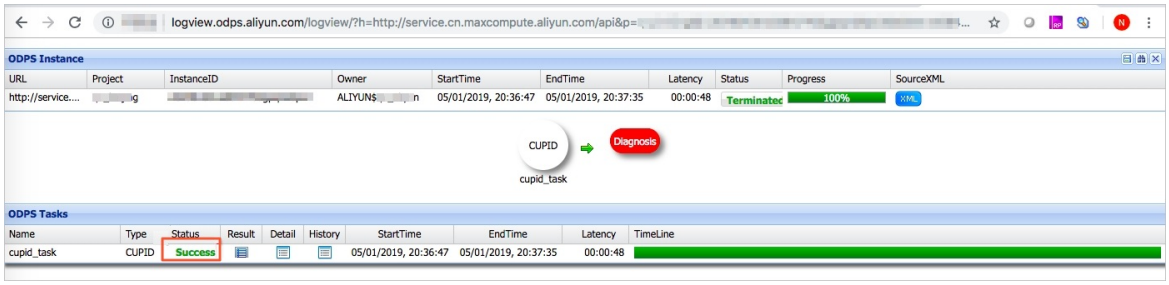
```
19/01/05 20:36:47 INFO YarnClientImplUtil: logview url: http://logview.odps.aliyun.com/logv
iew/?h=http://service.cn.maxcompute.aliyun.com/api&p=qn_beijing&i=xxx&token=xxx
<The operation succeeded if an output similar to the following result is displayed.>
19/01/05 20:37:34 INFO Client:
   client token: N/A
   diagnostics: N/A
   ApplicationMaster host: 11.220.xxx.xxx
   ApplicationMaster RPC port: 30002
   queue: queue
   start time: 1546691807945
   final status: SUCCEEDED
   tracking URL: http://jobview.odps.aliyun.com/proxyview/jobview/?h=http://service.cn.maxc
ompute.aliyun-inc.com/api&p=project_name&i=xxx&t=spark&id=application_xxx&metaname=xxx&toke
n=xxx
```

Use Logview to check a job

A URL that starts with `logview.odps.aliyun.com` is a Logview URL. Logview is a distributed job tracking tool developed for MaxCompute. It can be used to:

- Obtain the status of a job.
- Obtain the startup, stop, and scheduling information of each node in a job.
- Obtain the standard input and output logs of each node in a job. We recommend that the Spark output be written into stdout. By default, Spark log4j logs are written into stderr.
- Store execution log data. The data is retained for three to five days. When the local disk is fully occupied, both stdout and stderr are cleared.

1. Enter the Logview URL in the address bar of the browser and view the execution information of a job of the CUPID type.



2. On the page that appears, perform the following operations:
   i. Click **Detail** to view details about the job. **master-0** indicates the node on which Spark Driver resides.
   ii. Click **master-0** and select All Tab to view the information of the node.
   iii. Click **StdOut** to view the output of the node.
   iv. Click **StdErr** to view the log4j log of the node.

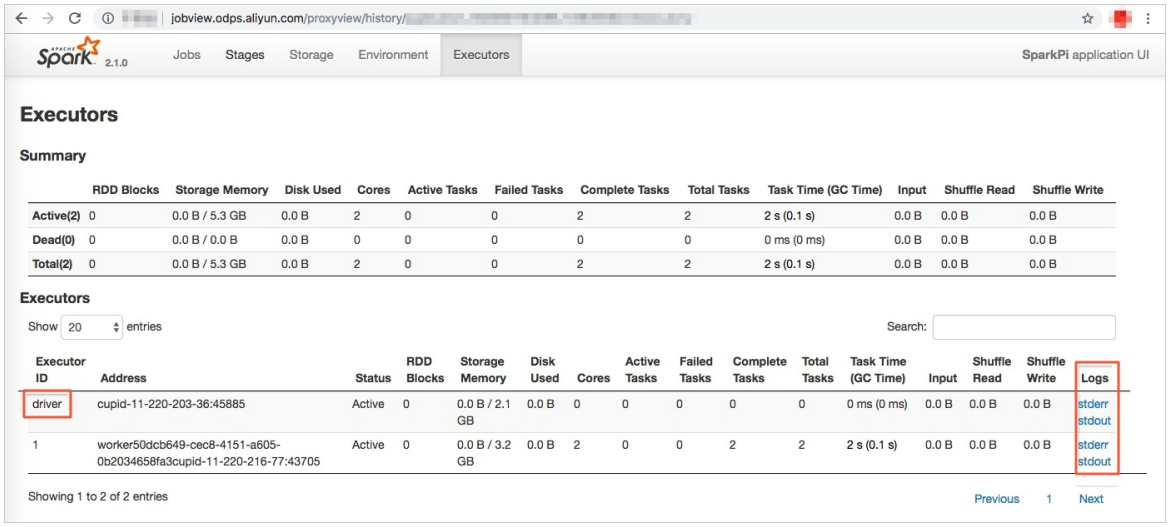
Use the Spark Web UI to check a job

The tracking URL in a log indicates that your job is submitted to MaxCompute. You can use the tracking URL to log on to the Spark Web UI or HistoryServer. The Spark Web UI can be used to:

- Obtain information of the native Spark Web UI.

- View the information of a running job in real time.
- Transfer events from Spark Driver to HistoryServer after a job is complete. This process may take one to three minutes. If you open the tracking URL immediately after a job is complete, error message `Application application_1560240626712_2078769635 not found.` may appear. If this happens, try again later.

1. Enter the tracking URL in the address bar of the browser and view the execution information of a Spark job.



2. On the page that appears, perform the following operations:
   i. Click the **Environment** tab to check whether the parameters of the Spark job are correctly configured.
   ii. Click the **Executors** tab to check whether dead nodes exist and whether stdout and stderr logs are generated for Spark Driver.
   iii. Click **stdout** to view the output of the node.
   iv. Click **stderr** to view the log4j log of the node.

5.Develop scheduling nodes through DataWorks

5.1. Create an ODPS SQL node

This topic describes how to create an ODPS SQL node. ODPS SQL nodes can process terabytes of data in distributed scenarios that do not require real-time processing by using the SQL-like syntax.

Prerequisites

The **MaxCompute** folder appears on the page only after you add a MaxCompute compute engine instance on the **Workspace Management** page. For more information, see Configure a workspace.

Context

Generally, it takes a long time from preparing to committing a job. You can use ODPS SQL nodes to process thousands to tens of thousands of transactions. ODPS SQL nodes are online analytical processing (OLAP) applications that are designed to deal with large amounts of data..

Limits

ODPS SQL nodes have the following limits:

- You cannot use the SET, USE, or SQL alias statement independently in the code of an ODPS SQL node. They must be executed with other SQL statements. For example, you can use a SET statement together with a CREATE TABLE statement.

```
set a=b;
create table name(id string);
```

- You cannot add comments to statements containing keywords, including the SET, USE, and SQL alias statements, in the code of an ODPS SQL node. For example, the comment is not allowed in the following code:

```
create table name(id string);
set a=b; // You cannot add a comment here.
create table name1(id string);
```

- The execution of an ODPS SQL node during workflow development and the scheduled execution of an ODPS SQL node have the following differences:
  - Execution during workflow development: combines all the statements containing keywords, including the SET, USE, and SQL alias statements, in the node code and executes them before you execute other SQL statements.
  - Scheduled execution: executes all SQL statements in sequence.

```
set a=b;
create table name1(id string);
set c=d;
create table name2(id string);
```

The following table describes the differences between the two execution modes for the preceding SQL statements.

SQL statement	Execution during workflow development	Scheduled execution

First SQL statement	`set a=b;`
	`set c=d;`
	`create table name1(id string);`
		`set a=b;`
		`create table name1(id string);`

Second SQL statement	`set a=b;`
	`set c=d;`
	`create table name2(id string);`
		`set c=d;`
		`create table name2(id string);`

- You must specify a scheduling parameter in the format of `key=value`. Do not add spaces before or after the equal sign (=). Examples:

```
time = {yyyymmdd hh:mm:ss} // The format is invalid.
a =b // The format is invalid.
```

- If you use keywords such as bizdate and date as scheduling parameters, you must specify the values in the format of yyyymmdd. If you want to use other time formats, do not use the preceding keywords as scheduling parameters. Example:

```
bizdate=201908 // Incorrect format.
```

- You can only use statements starting with SELECT, READ, or WITH to query the result data for a node during the workflow development. Otherwise, no results are returned.
- Separate multiple SQL statements with semicolons (;). Each SQL statement must occupy a line.
  - Incorrect example

    ```
    create table1;create table2
    ```

  - Correct example

    ```
    create table1;
    create table2;
    ```

- If extension functions in MaxCompute V2.0 use new data types, you must add `set odps.sql.type.system.odps2=true;` before the SQL statements that use the functions, and commit and execute the code together with the SQL statements.
- If you want to add comments to SQL statements, you cannot use semicolons (;) in comments.

  Incorrect example:

```
create table1; // Create a table named table1; Then, create a table named table2.
create table2;
```

- An ODPS SQL node can contain up to 200 KB of SQL code and up to 200 SQL statements.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. In the Scheduled Workflow pane, move the pointer over the **+Create** icon and choose **MaxCompute > ODPS SQL**.

   Alternatively, you can click a workflow in the Scheduled Workflow pane, right-click **MaxCompute**, and then choose **Create > ODPS SQL**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Write and run the code of the ODPS SQL node.

   After the node is created, write the code of the ODPS SQL node. The code must conform to the syntax. For more information about SQL syntax, see Overview of MaxCompute SQL.

   > **Note** Due to the adjustment made by the International Organization for Standardization (ISO) on the UTC+8 time zone, differences exist between the real time and the output time when you execute related SQL statements in DataWorks. Between the years of 1900 and 1928, the time difference is 352 seconds. Before the year of 1900, the time difference is 9 seconds.

   You cannot use the `SET` statement independently in the node code. The SET statement must be executed together with other SQL statements. For example, you can use a SET statement together with a SELECT statement.

   ```
   set odps.sql.allow.fullscan=true;
   select 1;
   ```

   For more information about the SET statement, see SET operations.

   The following example creates a table, inserts data to the table, and queries data in the table:

   i. Create a table named test1.

   ```
   CREATE TABLE IF NOT EXISTS test1
   ( id BIGINT COMMENT '' ,
     name STRING COMMENT '' ,
     age BIGINT COMMENT '' ,
     sex STRING COMMENT '');
   ```

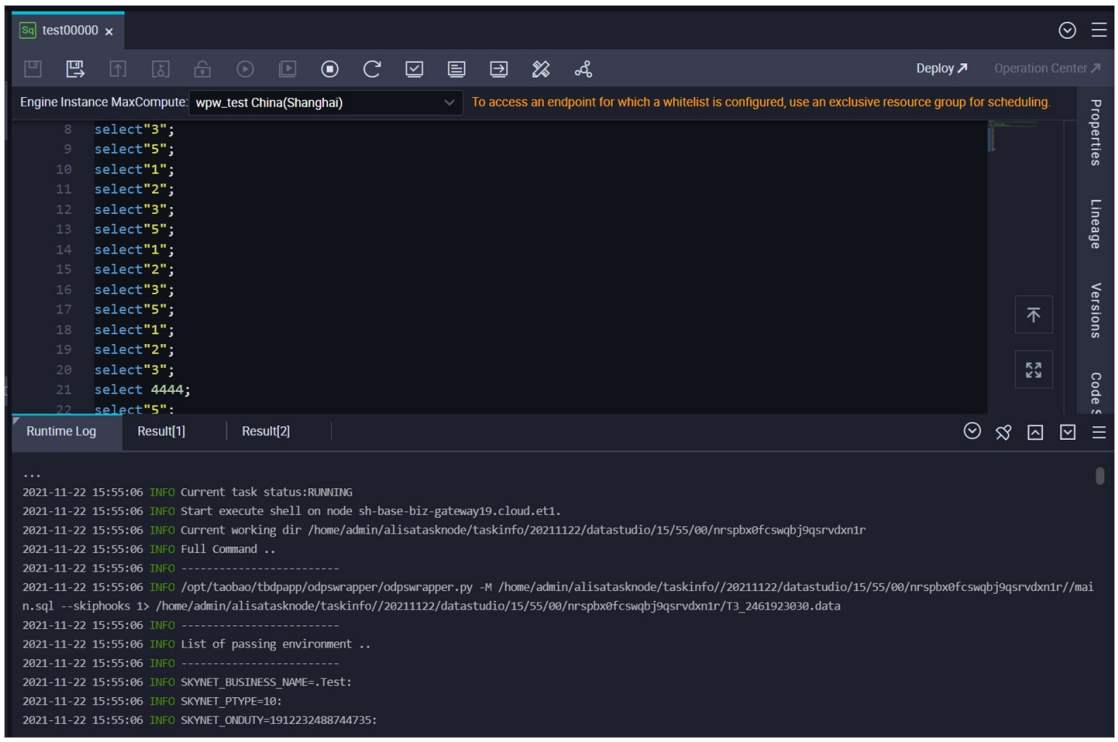
ii. Insert data to the table.

```
INSERT INTO test1 VALUES (1,'Zhang San',43,'Male');
INSERT INTO test1 VALUES (1,'Li Si',32,'Male');
INSERT INTO test1 VALUES (1,'Chen Xia',27,'Female');
INSERT INTO test1 VALUES (1,'Wang Wu',24,'Male');
INSERT INTO test1 VALUES (1,'Ma Jing',35,'Female');
INSERT INTO test1 VALUES (1,'Zhao Qian',22,'Female');
INSERT INTO test1 VALUES (1,'Zhou Zhuang',55,'Male');
```

iii. Query data in the table.

```
select * from test1;
```

iv. After you enter the preceding SQL statements in the code editor, click the icon in the toolbar. DataWorks executes your SQL statements from top to bottom and displays logs.



> **Note**
>
> - If multiple MaxCompute compute engine instances are associated with the current workspace, select one MaxCompute compute engine instance before you click the Run icon.
> - If the MaxCompute compute engine instance that you selected uses the custom resource group in pay-as-you-go mode, click the icon in the toolbar to estimate the cost. The actual expenses generated for running this node are subject to the bill.

The `INSERT INTO` statement may result in unexpected data duplication. DataWorks does not re-execute the `INSERT INTO` statement, but it may rerun nodes that contain the statement. We recommend that you avoid using the `INSERT INTO` statement. If DataWorks runs a node that contains the INSERT INTO statement, the following information appears in operational logs:

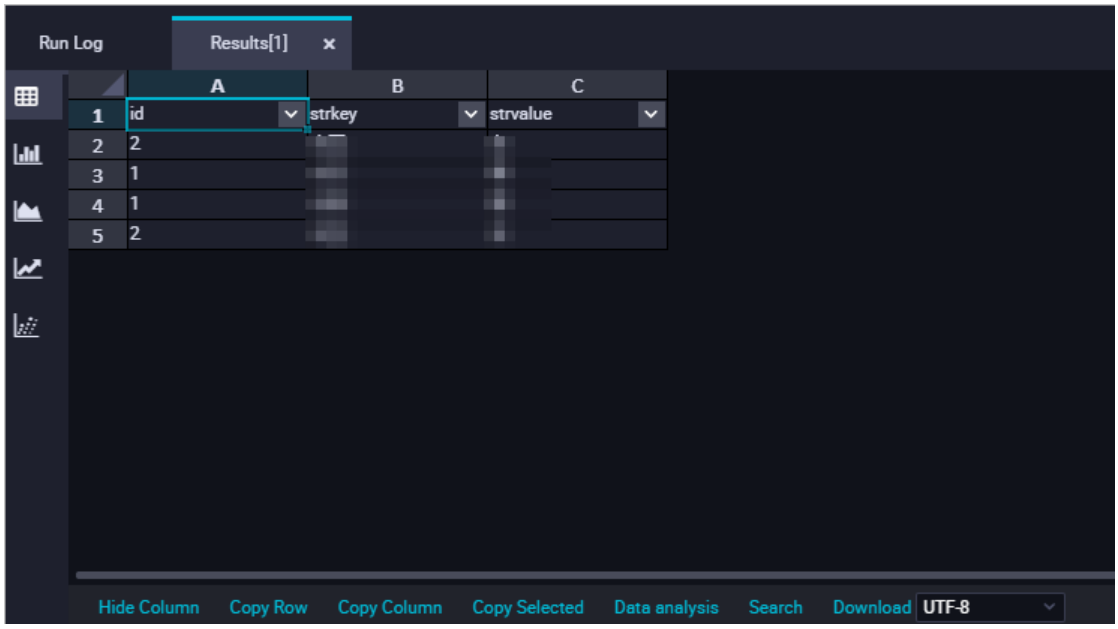
```
The INSERT INTO statement may cause repeated data insertion. DataWorks does not re-execute the INSERT INTO statement, but it may rerun nodes that contain the statement. We recommend that you avoid using the INSERT INTO statement.
If you continue to use the INSERT INTO statement, you are deemed to be aware of the risks and are willing to take the consequences of potential data duplication.
```

v. After the SQL statements are executed, click the icon in the toolbar to save the SQL code.

vi. View the query result.

DataWorks displays the query result in a workbook.

You can view or manage the query result in the workbook, or copy the query result to an Excel file on your on-premises machine.



Button	Description
Hide Column	Select one or more columns and click **Hide Column** in the lower part to hide the selected columns.
Copy Row	Select one or more rows and click **Copy Row** in the lower part to copy the selected rows.
Copy Column	Select one or more columns and click **Copy Column** in the lower part to copy the selected columns.
Copy Selected	Select one or more cells in the workbook and click **Copy Selected** in the lower part to copy the selected cells.
Data Analysis	Click **Data Analysis** in the lower part to go to the workbook editing page.
Search	Click **Search** in the lower part to search for data in the workbook. After you click the button, a search box appears in the upper-right corner of the Result tab.
Download	Download the result data in the encoding format of **GBK** or **UTF-8**.

6. On the node configuration tab, click the **Properties** tab in the right-side navigation pane. In the Properties panel, configure the scheduling properties for the node. For more information, see Configure basic properties.
7. Commit the node.

> **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, set the **Change description** parameter and select **I confirm to proceed with the commit operation**.

If an alert appears, indicating that the input and output that you set do not match with those identified in code lineage analysis, you can choose to ignore the alert or click **Cancel** and then modify the dependencies.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.2. Create an SQL component node

An SQL component is an SQL script template that contains multiple input and output parameters. You can create and run an SQL component node to filter source table data, join source tables, and aggregate source tables to generate a result table.

Prerequisites

- DataWorks Standard Edition or higher is activated.
- A MaxCompute compute engine is bound to the workspace where you want to create an SQL component node. The MaxCompute service is available in a workspace only after you bind a MaxCompute compute engine to the workspace on the **Workspace Management** page. For more information, see Configure a workspace.
- SQL script templates are prepared. For more information, see Create a script template.

Context

When a new version is released for a script template, you can decide whether to update the version of the script template used in your nodes to the new version.

The script template update feature allows developers to update script template versions. This feature helps improve the process execution efficiency and optimize the business performance.

Assume that User A uses a script template that is released by User B in Node C. After User B updates the version of the script template, User A receives an update notification. User A can decide whether to update the version of the script template in Node C.

To update the version of the SQL script template in an SQL component node, perform the following steps: Go to the configuration tab of the node, click **Update code version** in the upper-right corner of the code editor, and then check whether the parameter settings of the old-version SQL script template are valid in the new version. If the parameter settings are invalid in the new version, modify the settings based on the description of the new-version SQL script template. Then, save the node and commit it for deployment.

Procedure

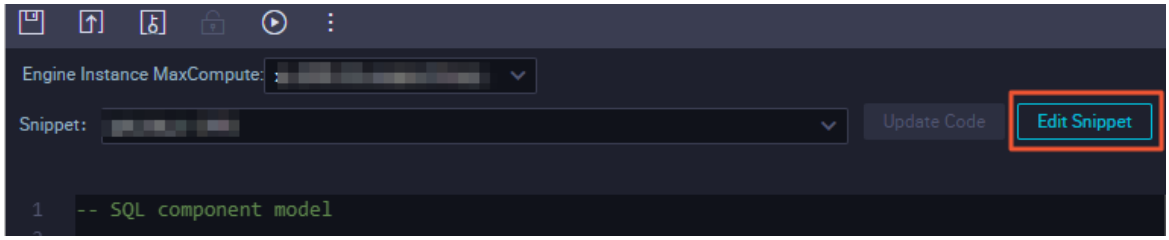
1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the Data Development tab, move the pointer over the Create icon and choose **MaxCompute > SQL component node**.

   Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > SQL component node**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the node configuration tab, select an SQL script template from the **Select code components** drop-down list.

   If multiple MaxCompute compute engines are bound to the current workspace, select one from the **MaxCompute Engine instance** drop-down list.

   After you select an SQL script template, you can click **Open component** to go to the details page of the template.

   To improve development efficiency, you can create data analytics nodes by using the script templates that are provided by workspace members and tenants.
   - The script templates that are provided by members of the current workspace are available on the **Components** tab.
   - The script templates that are provided by tenants are available on the **Common components** tab.
6. Click the **Parameter configuration** tab in the right-side navigation pane and set parameters for the SQL script template.
7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.3. Create a MaxCompute Spark node

A MaxCompute Spark node processes data by using Java and Python. This topic describes how to create and configure a MaxCompute Spark node.

Context

Python resources are referenced in the user-defined functions (UDFs) of Python. However, Python resources have limited usage because you can obtain only a few dependent third-party packages.

PyODPS 2 and PyODPS 3 nodes support Python resources. For more information, see Create a PyODPS 2 node and Create a PyODPS 3 node.

This topic describes how to create JAR and Python resources and upload them based on your business requirements after you create a MaxCompute Spark node.

Create a JAR resource

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the Create icon and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can find your desired workflow, right-click the workflow name, and then choose **Create > MaxCompute > Resource > JAR**.
3. In the **Create Resource** dialog box, specify **Resource Name** and **Location**.

   > **Note**
   > - If multiple MaxCompute compute engine instances are bound to the current workspace, you must select one from the **Engine Instance MaxCompute** drop-down list.
   > - If the selected JAR package has been uploaded from the MaxCompute client, clear **Upload to MaxCompute**. If you do not clear it, an error occurs during the upload process.
   > - The resource name can be different from the name of the uploaded file.
   > - The resource name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). The name is not case-sensitive. A JAR resource name must end with .jar, and a Python resource name must end with .py.
4. Click **Upload** and select the file to upload.

5. Click **Create**.
6. Click the icon in the top toolbar to commit the code.
7. In the **Commit Node** dialog box, enter your comments in the **Change description** field and click **OK**.

Create a Python resource

1. On the **DataStudio** page, move the pointer over the +Create icon and choose **MaxCompute > Resource > Python**.

   Alternatively, you can find your desired workflow, right-click the workflow name, and then choose **Create > MaxCompute > Resource > Python**.
2. In the **Create Resource** dialog box, specify **Resource Name** and **Location**.

   > Note
   > - If multiple MaxCompute compute engine instances are bound to the current workspace, you must select one from the **Engine Instance MaxCompute** drop-down list.
   > - The resource name can contain letters, digits, periods (.), underscores (_), and hyphens (-), and must end with .py.
   > - The created Python resource can be run by using only the code of Python 2.X or 3.X.
3. Click **Create**.
4. On the node configuration tab, enter the Python code.

   In the following example, the Python code defines the logic for checking whether parameter values are correct instead of the logic for processing data.

```
# -*- coding: utf-8 -*-
import sys
from pyspark.sql import SparkSession
try:
    # for python 2
    reload(sys)
    sys.setdefaultencoding('utf8')
except:
    # python 3 not needed
    pass
if __name__ == '__main__':
    spark = SparkSession.builder\
        .appName("spark sql")\
        .config("spark.sql.broadcastTimeout", 20 * 60)\
        .config("spark.sql.crossJoin.enabled", True)\
        .config("odps.exec.dynamic.partition.mode", "nonstrict")\
        .config("spark.sql.catalogImplementation", "odps")\
        .getOrCreate()
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass
    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass
    return False
print(is_number('foo'))
print(is_number('1'))
print(is_number('1.3'))
print(is_number('-1.37'))
print(is_number('1e3'))
```

5. Click the icon in the top toolbar to commit the code.
6. In the **Commit Node** dialog box, enter your comments in the **Change description** field and click **OK**.

Create a MaxCompute Spark node

1. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **MaxCompute > ODPS Spark**.

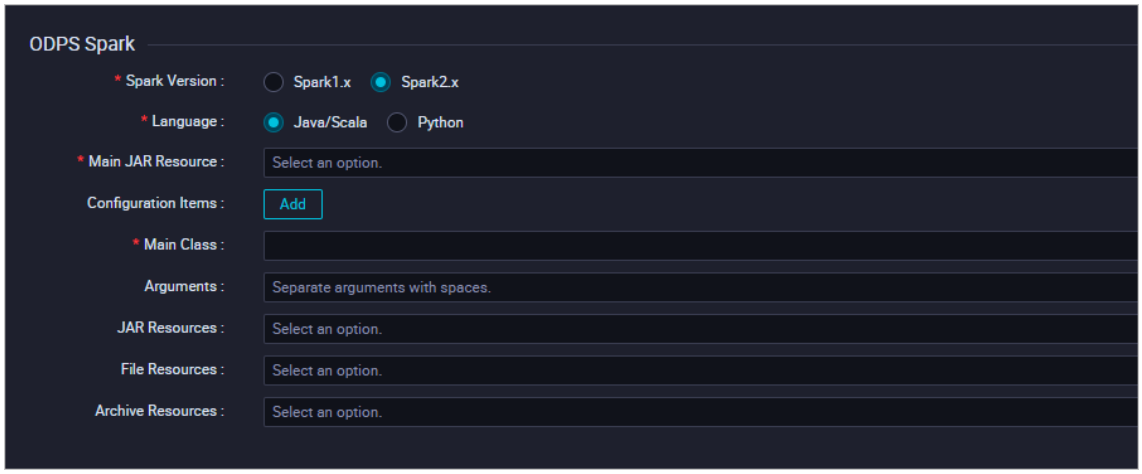
   Alternatively, you can find your desired workflow, right-click the workflow name, and then choose **Create > MaxCompute > ODPS Spark**.
2. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

3. Click **Commit**.
4. On the configuration tab of the MaxCompute Spark node, configure the parameters. For more information about MaxCompute Spark nodes, see Overview.

   Two options are available for the **Spark Version** and **Language** parameters of the MaxCompute Spark node. The parameters on this tab vary based on the value of the **Language** parameter. You can configure the parameters as prompted.

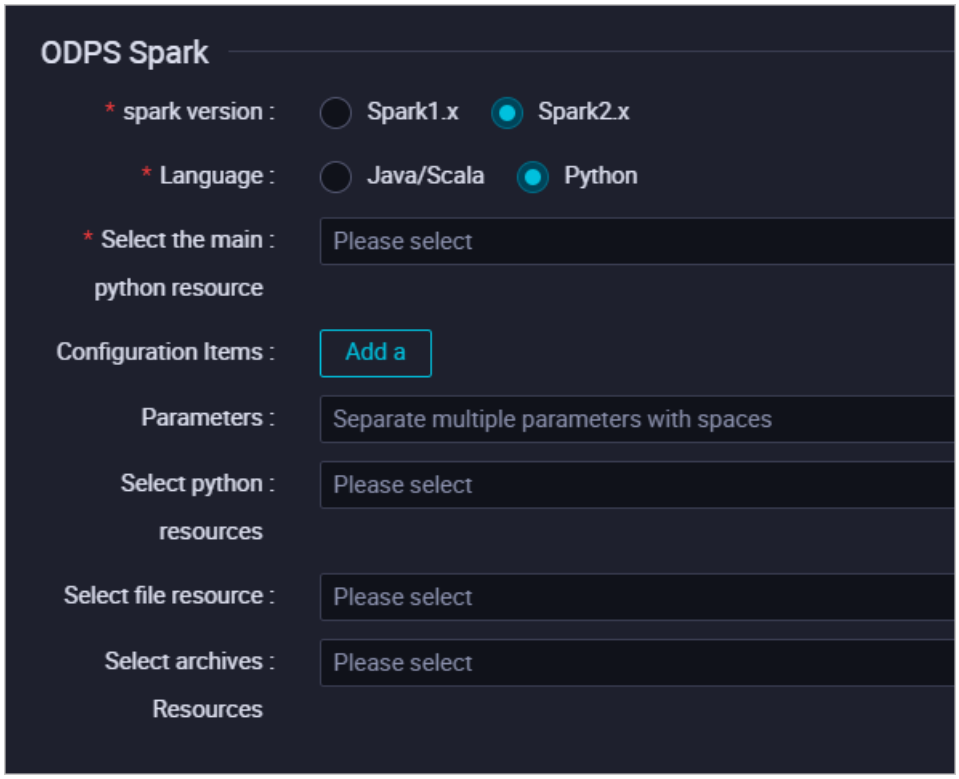
   - The following table describes the parameters that appear on this tab when you set **Language** to **Java/Scala**.



Parameter	Description
Spark Version	The Spark version of the node. Valid values: **Spark1.x** and **Spark2.x**.
Language	The programming language used by the node. Select **Java/Scala**.
Main JAR Resource	The main JAR resource referenced by the node. Select the desired JAR resource from the drop-down list.
Configuration Items	The configuration items of the node. Click **Add**, and enter a key and a value to add a configuration item. Alternatively, you can directly select a key from the field that appears. In this case, a value is automatically entered for the key.
Main Class	The class name of the node.

Parameter	Description
Arguments	The parameters of the node. Separate multiple parameters with spaces. You can configure scheduling parameters. If you want to perform such an operation, click Properties in the right-side navigation pane. For more information, see Overview of scheduling parameters. **Note** After you configure the scheduling parameters, you must continue to configure the parameters for the node. The scheduling parameters and node parameters are run in sequence.
JAR Resources	The JAR resource referenced by the node. The system displays all the uploaded JAR resources. Select the desired JAR resource from the drop-down list.
File Resources	The file resource referenced by the node. The system displays all the uploaded file resources. Select the desired file resource from the drop-down list.
Archive Resources	The archive resource referenced by the node. The system displays all the uploaded archive resources that are compressed. Select the desired archive resource from the drop-down list.

- The following table describes the parameters that appear when you set **Language** to **Python**.



Parameter	Description
Spark Version	The Spark version of the node. Valid values: **Spark1.x** and **Spark2.x**.
Language	The programming language used by the node. Select **Python**.
Main Python Resource	The main Python resource referenced by the node. Select the desired Python resource from the drop-down list.
Configuration Items	The configuration items of the node. Click **Add**, and enter a key and a value to add a configuration item. Alternatively, you can directly select a key from the field that appears. In this case, a value is automatically entered for the key.
Arguments	The parameters of the node. Separate multiple parameters with spaces.
Python Resources	The Python resource referenced by the node. The system displays all the uploaded Python resources. Select the desired Python resource from the drop-down list.
File Resources	The file resource referenced by the node. The system displays all the uploaded file resources. Select the desired file resource from the drop-down list.
Archive Resources	The archive resource referenced by the node. The system displays all the uploaded archive resources that are compressed. Select the desired archive resource from the drop-down list.

5. On the node configuration tab, click **Properties** in the right-side navigation pane. On the Properties tab, configure the scheduling properties for the node. For more information, see Configure basic properties.
6. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
7. Test the node. For more information, see View auto triggered nodes.

5.4. Create a PyODPS 2 node

DataWorks supports PyODPS 2 nodes, which are integrated with MaxCompute SDK for Python. You can edit Python code in PyODPS 2 nodes of DataWorks to process data in MaxCompute.

Context

MaxCompute provides SDK for Python. You can use the SDK for Python to process data in MaxCompute. For more information, see SDK for Python.

> **Note**
> - The Python version of PyODPS 2 nodes is 2.7.
> - Each PyODPS 2 node can process a maximum of 50 MB data and can occupy a maximum of 1 GB memory. Otherwise, the PyODPS 2 node stops running. Avoid writing excessive data processing code for a PyODPS 2 node.
> - For more information about the hints parameter, see SET operations.

PyODPS 2 nodes are designed to use MaxCompute SDK for Python. If you want to run pure Python code, you can create a Shell node to run the Python scripts that are uploaded to DataWorks. For information about how to reference a third-party package in a PyODPS 2 node, see Use a PyODPS node to reference a third-party package.

Create a PyODPS 2 node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **MaxCompute > PyODPS 2**.

   Alternatively, you can click the required workflow in the **Business Flow** section, right-click **MaxCompute**, and then choose **Create > PyODPS 2**.

   For more information about how to create a workflow, see Create a workflow.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Edit the PyODPS 2 node.
   i. Use the MaxCompute entry.

   In DataWorks, each PyODPS 2 node includes the global variable odps or o, which is the MaxCompute entry. Therefore, you do not need to manually specify the MaxCompute entry.

   ```
   print(odps.exist_table('PyODPS_iris'))
   ```

ii. Execute SQL statements.

PyODPS 2 supports MaxCompute SQL queries and allows you to obtain the query results. The return value of the execute_sql or run_sql method is running instances.

Not all SQL statements that you can execute on the MaxCompute client are supported by PyODPS 2. To execute statements other than data definition language (DDL) and data manipulation language (DML) statements, use other methods.

For example, to execute a GRANT or REVOKE statement, use the run_security_query method. To run a Machine Learning Platform for AI (PAI) command, use the run_xflow or execute_xflow method.

```
o.execute_sql('select * from dual')  # Execute the statement in synchronous mode. O
ther nodes are blocked until the SQL statement is executed.
instance = o.run_sql('select * from dual') # Execute the statement in asynchronous
mode.
print(instance.get_logview_address()) # Obtain the Logview URL.
instance.wait_for_success() # Block other nodes until the SQL statement is executed
.
```

iii. Set runtime parameters.

You can use the hints parameter to set the runtime parameters. The type of the hints parameter is dict.

```
o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper.split.size': 16}
)
```

If you set the sql.settings parameter for the global configuration, you must set the runtime parameters each time you run the code.

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris')  # The hints parameter is set based on g
lobal configuration.
```

iv. Obtain SQL query results.

You can perform the open_reader operation on the instance that executes the SQL statement in the following scenarios:

- The SQL statement returns structured data.

```
with o.execute_sql('select * from dual').open_reader() as reader:
for record in reader:  # Process each record.
```

- SQL statements such as DESC are executed. In this case, you can use the reader.raw property to obtain raw SQL query results.

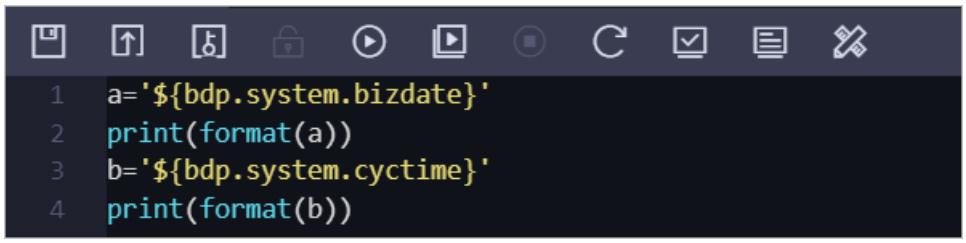
```
with o.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

> **Note** If you use a custom scheduling parameter and run the PyODPS 2 node on the page, you must fix the parameter to a time. PyODPS nodes do not support direct replacement.

6. On the node configuration tab, click **Properties** in the right-side navigation pane. Set the scheduling properties for the node. For more information, see Configure basic properties.

   PyODPS 2 nodes can use built-in scheduling parameters and custom scheduling parameters:

   - If the PyODPS 2 node uses built-in scheduling parameters, you can assign values on the node configuration tab.

     
     ```
     a='${bdp.system.bizdate}'
     print(format(a))
     b='${bdp.system.cyctime}'
     print(format(b))
     ```

     > **Note** The shared resource group cannot be connected to the Internet. If you want to connect to the Internet, we recommend that you use a custom resource group or exclusive resources for scheduling. Only DataWorks Professional Edition supports custom resource groups. You can purchase exclusive resources for scheduling in all DataWorks editions. For more information, see DataWorks exclusive resources.

   - You can also set custom scheduling parameters in the **Properties > General** section.

     > **Note** You must specify a custom parameter in the format of args['Parameter name'], for example, `print (args['ds'])` .

7. Commit the node.

   > **Notice** Before you commit the node, you must set the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node.

8. Test the node. For more information, see View auto triggered nodes.

Built-in services for a PyODPS node

A PyODPS node contains the following built-in services:

- setuptools
- cython
- psutil
- pytz
- dateutil
- requests
- pyDes

- numpy
- pandas
- scipy
- scikit_learn
- greenlet
- six
- Other built-in services in Python 2.7, such as smtplib

5.5. Create a PyODPS 3 node

This topic describes the usage limits of PyODPS 3 nodes and how to create a PyODPS 3 node in DataWorks.

Limits

- Python 3 defines bytecode differently in its different subversions such as Python 3.7 and Python 3.8.

  MaxCompute is compatible with Python 3.7. A MaxCompute client that uses another subversion of Python 3 will return an error when it executes code with specific syntax. For example, a MaxCompute client that uses Python 3.8 will return an error when it executes code with the finally block syntax. We recommend that you use Python 3.7.

- Each PyODPS 3 node can process a maximum of 50 MB data and can occupy a maximum of 1 GB memory. Otherwise, DataWorks terminates the PyODPS 3 node. Do not write code that will process an extra large amount of data in a PyODPS 3 node.
- PyODPS 3 nodes can run on a shared resource group and an exclusive resource group for scheduling that is purchased after April 2020. If you create an exclusive resource group for scheduling before April 2020, submit a ticket to upgrade the resource group.

Create a PyODPS 3 node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **MaxCompute > PyODPS 3**.

   Alternatively, click the required workflow under **Business Flow**, right-click **MaxCompute**, and choose **Create > PyODPS 3**.

   For more information about how to create a workflow, see Create a workflow.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Edit and run the PyODPS 3 node.

For example, if you want to use the execute_sql() method, you must specify runtime parameters for the SQL statements.

```
hints={'odps.sql.python.version': 'cp37', 'odps.isolation.session.enable': True}
```

If you want to use a user-defined function (UDF) for DataFrame, such as df.map, df.map_reduce, df.apply, and df.agg, specify the following settings:

```
hints={'odps.isolation.session.enable': True}
```

PyODPS determines the runtime environment of the UDF and commits SQL statements based on the Python version that the client uses. Assume that a public Python UDF is used to call DataFrame. When the client uses Python 3, statements are interpreted to Python 3. If the UDF executes a print statement specific to Python 2, the client returns the ScriptError error. For more information about how to reference a third-party package in a PyODPS 2 node, see Use a PyODPS node to reference a third-party package.

6. Click the **Properties** tab in the right-side navigation pane and set the scheduling properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

   Notice You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
8. Test the node. For more information, see View auto triggered nodes.

5.6. Create an ODPS Script node

You can create an ODPS Script node to develop an SQL script by using the code editor provided by the MaxCompute V2.0 SQL engine.

Context

An ODPS Script node allows DataWorks to compile an SQL script as a whole, instead of compiling the SQL statements in the script one by one. This way, the SQL script is committed and run as a whole. This ensures that an execution plan is queued and executed only once, making full use of MaxCompute computing resources.

ODPS Script nodes allow you to write SQL statements based on your business logic in a way similar to that of using a common programming language. You do not need to consider how to organize the SQL statements.

```
-- SET statements
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=***;]
[...]
-- Data definition language (DDL) statements
create table table1 xxx;
[create table table2 xxx;]
[...]
-- Data manipulation language (DML) statements
@var1 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM table3
    [WHERE where_condition];
@var2 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM table4
    [WHERE where_condition];
@var3 := SELECT [ALL | DISTINCT] var1.select_expr, var2.select_expr, ...
    FROM @var1 join @var2 on ... ;
INSERT OVERWRITE|INTO TABLE [PARTITION (partcol1=val1, partcol2=val2 ...)]
    SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM @var3;
[@var4 := SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var1
    UNION ALL | UNION
    SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var2;
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
    AS
    SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM var4;]
```

ODPS Script nodes have the following limits:

- ODPS Script nodes support SET statements, DML statements, and some DDL statements. The DDL statements, such as `DESC` and `SHOW` statements, that are used to return data are not supported.
- A complete script consists of SET statements, DDL statements, and DML statements in sequence. You can write one or more statements of each type, or skip a type without writing any statements of that type. However, you cannot mix different types of statements together. You must strictly follow this sequence: SET statements > DDL statements > DML statements.
- The at signs (@) preceding several statements indicate that these statements are connected by using variables.
- You can write only one statement, for example, a SELECT statement, that returns data in a script. If you write multiple such statements in a script, an error occurs. We recommend that you do not use SELECT statements in a script.
- You can write only one `CREATE TABLE AS` statement in a script, and this statement must be the last statement in the script. We recommend that you write CREATE TABLE statements and INSERT statements in different sections to separate them.
- If one statement in a script fails, the whole script fails.
- A job is generated to process data only after all the input data is prepared for a script.
- If you specify a statement for writing data to a table and then a statement for reading data from the table in the same script, an error occurs. For example, if you write the following statements in a script, an error occurs:

```
insert overwrite table src2 select * from src where key > 0;
@a := select * from src2;
select * from @a;
```

To avoid the error, write the statements in the following format:

```
@a := select * from src where key > 0;
insert overwrite table src2 select * from @a;
select * from @a;
```

Sample script:

```
create table if not exists dest(key string , value bigint) partitioned by (d string);
create table if not exists dest2(key string,value bigint ) partitioned by (d string);
@a := select * from src where value >0;
@b := select * from src2 where key is not null;
@c := select * from src3 where value is not null;
@d := select a.key,b.value from @a left outer join @b on a.key=b.key and b.value>0;
@e := select a.key,c.value from @a inner join @c on a.key=c.key;
@f := select * from @d union select * from @e union select * from @a;
insert overwrite table dest partition (d='20171111') select * from @f;
@g := select e.key,c.value  from @e join @c on e.key=c.key;
insert overwrite table dest2 partition (d='20171111') SELECT * from @g;
```

ODPS Script nodes are applicable to the following scenarios:

- You can use an ODPS Script node to rewrite a single statement with nested subqueries, or a script that must be split into multiple statements to make it simpler.
- Data from different data stores may be ready at different time points, and the time difference may be large. For example, the data from one data store can be ready at 01:00, whereas that from the other data store can be ready at 07:00. In this case, table variables are not suitable for connecting statements. You can use an ODPS Script node to combine the statements to a script.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the Data Development tab, move the pointer over the **+Create** icon and choose **MaxCompute > ODPS Script**.

   Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > ODPS Script**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.

5. On the node configuration tab, write the SQL script of the ODPS Script node. For more information, see Develop and submit an SQL script.
6. On the node configuration tab, click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
8. Test the node. For more information, see View auto triggered nodes.

5.7. Create an ODPS MR node

MaxCompute supports the MapReduce API. You can create and commit ODPS MR nodes that call the Java API operations of MapReduce to develop MapReduce programs for processing data in MaxCompute.

Prerequisites

Required resources are uploaded and committed.

For more information about how to edit and use an ODPS MR node, see WordCount example.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Create a JAR resource.
   i. On the Data Development tab, move the pointer over the Create icon and choose **MaxCompute > Resource > JAR**.

      Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > Resource > JAR**.

ii. In the **New resource** dialog box, set the **Resource Name** and **Destination folder** parameters.

> **Note**
> - If multiple MaxCompute compute engines are bound to the current workspace, you must select one from the **MaxCompute Engine instance** drop-down list.
> - If the selected JAR package has been uploaded from the MaxCompute client, clear **Upload as an ODPS resource**. If you do not clear it, an error will occur during the upload process.
> - The resource name can be different from the name of the uploaded file.
> - A resource name can contain letters, digits, underscores (_), and periods (.), and is not case-sensitive. It must be 1 to 128 characters in length. A JAR resource name must end with .jar, and a Python resource name must end with .py.

iii. Click **Click Upload**, select a local JAR package, and then click **Open**.

In this example, upload the mapreduce example.jar package.

iv. Click **Confirm**.

v. Click the and icons in the toolbar to save and commit the resource to the development environment.

3. Create an ODPS MR node.

i. On the Data Development tab, move the pointer over the Create icon and choose **MaxCompute > ODPS MR**.

Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > ODPS MR**.

ii. In the **New node** dialog box, set the **Node name** and **Destination folder** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

iii. Click **Submit**.

4. On the node configuration tab, enter the following sample code:

```
-- Create an input table.
CREATE TABLE if not exists jingyan_wc_in (key STRING, value STRING);
-- Create an output table.
CREATE TABLE if not exists jingyan_wc_out (key STRING, cnt BIGINT);
    --- Create the dual table.
    drop table if exists dual;
    create table dual(id bigint); -- Create the dual table if no dual table exists in t
he current workspace.
    --- Initialize the dual table.
    insert overwrite table dual select count(*)from dual;
    --- Insert the sample data to the wc_in table.
    insert overwrite table jingyan_wc_in select * from (
    select 'project','val_pro' from dual
    union all
    select 'problem','val_pro' from dual
    union all
    select 'package','val_a' from dual
    union all
    select 'pad','val_a' from dual
      ) b;
-- Reference the created JAR resource. You can find the JAR resource in the resource li
st, right-click the JAR resource, and then select Reference Resources to reference the
resource.
--@resource_reference{"mapreduce-examples.jar"}
jar -resources mapreduce-examples.jar -classpath ./mapreduce-examples.jar com.aliyun.od
ps.mapred.open.example.WordCount jingyan_wc_in jingyan_wc_out
```

Code description:

- `--@resource_reference`: references a resource. Find the resource to be referenced in the resource list, right-click it, and then select **Reference Resources** to generate the reference statement.
- `-resources`: the name of the referenced JAR resource.
- `-classpath`: the path of the referenced JAR resource. You need to enter only *./Resource name* because the resource has been referenced.
- `com.aliyun.odps.mapred.open.example.WordCount`: the name of the main class in the JAR resource to be called during node running. The main class name must be the same as that in the JAR resource.
- `jingyan_wc_in`: the name of the input table of the ODPS MR node. The input table is created by using the preceding code.
- `jingyan_wc_out`: the name of the output table of the ODPS MR node. The output table is created by using the preceding code.
- If you reference multiple JAR resources in an ODPS MR node, separate the resource paths with commas (,), for example, `-classpath ./xxxx1.jar,./xxxx2.jar`.

5. On the node configuration tab, click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.
6. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

7. Test the node. For more information, see View auto triggered nodes.

5.8. Create a MaxCompute table

This topic describes how to create a MaxCompute table.

Prerequisites

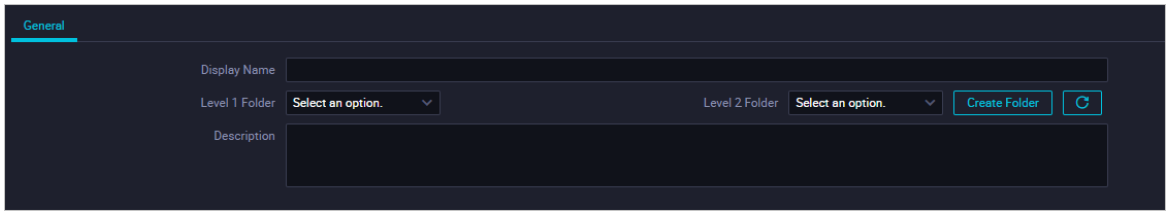
A MaxCompute compute engine is bound to the workspace where you want to create a MaxCompute table. The MaxCompute service is available in a workspace only after you bind a MaxCompute compute engine to the workspace on the **Workspace Management** page. For more information, see Configure a workspace.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the Data Development tab, move the pointer over the +Create icon and choose **MaxCompute > Table**.

   Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > Table**.
3. In the **New table** dialog box, set the **Table name** parameter and click **Submit**.

   > **Notice**
   > - The table name must be 1 to 64 characters in length. It must start with a letter and cannot contain special characters.
   > - If multiple MaxCompute compute engines are bound to the current workspace, you must select one from the **MaxCompute Engine instance** drop-down list.
4. In the **Basic properties** section, set the parameters as required.



Parameter	Description
Display Name	The name of the MaxCompute table.
Level 1 Folder	The name of the level-1 folder where the table resides. **Note** Level-1 and level-2 folders show the table locations in DataWorks for you to manage tables more conveniently.
Level 2 Folder	The name of the level-2 folder where the table resides.
Create Folder	Click **Create Folder** to go to the **Theme management** tab. On this tab, you can create level-1 and level-2 folders for tables. After you create a folder, click the icon next to New theme to synchronize the folder.
Description	The description of the table.

5. Create a table.

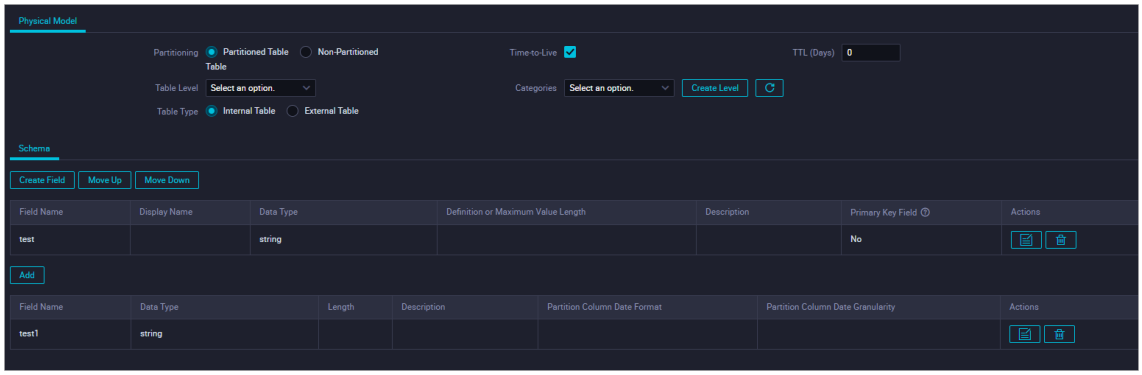
   Use one of the following methods to create a table:

   - Create a table in DDL mode.

     Click **DDL mode** in the toolbar. In the dialog box that appears, enter the table creation statement and click **Generate table structure**. Then, parameters in the **Physical model design** and **Table structure design** sections are automatically set. For more information about the table creation statement, see the "Create tables" section in Table operations.

   - Create a table on the graphical user interface (GUI).

     DataWorks allows you to create tables on the GUI.



Section	GUI element	Description
Physical model design	**Partition type**	The type of the table. Valid values: **Partition Table** and **Non-partitioned table**.
	Life cycle	Specifies whether to enable the lifecycle feature for the table. If you select Life cycle, you must enter a number in the **Select lifecycle (days)** field. The system automatically deletes data after the data has been stored in the table for the specified period.
	Hierarchy	The layer where the table data is stored or processed. Generally, a data warehouse consists of the operational data store (ODS), common data model (CDM), and application data store (ADS) layers. You can specify a custom name for each layer.
	Physical classification	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can specify a custom name for each category. If you want to create a table category or layer, click **New Level** to go to the **Hierarchical management** tab. **Note** Categories are designed only for your management convenience and do not involve underlying implementation.
	Add Field	The button for adding a field. To add a field, click **Add Field**, configure the field information, and then click the Save icon.
	Move up	The buttons for adjusting the field sequence of the table. If you adjust the sequence of fields in an existing table and then commit the table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Move down	
	Field name	The name of the field. The name can contain letters, digits, and underscores (_).
	Field Chinese name	The display name of the field.

Section	GUI element	Description
Table structure design	**Field type**	The data type of the field. MaxCompute supports the following data types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, VARCHAR, CHAR, STRING, BINARY, DATETIME, DATE, TIMESTAMP, BOOLEAN, ARRAY, MAP, and STRUCT. For more information, see Data types.
	Length/settings	The length limit of the field. You must set this parameter if the data type that you specify for the field has a length limit.
	Description	The description of the field.
	Primary key	Specifies whether the field serves as the primary key or part of a composite primary key.
	Edit icon	The icon for editing the field. After you save the field, you can click this icon to edit the field and then click the Save icon to save the edited field.
	Delete icon	The icon for deleting the field. **Note** If you delete a field from an existing table and then commit the table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
Partition field design	**Add partition**	The button for adding a partition. If you set **Partition type** to **Partition Table** in the **Physical model design** section, you must add and configure a partition for the table. You can click Add partition to add a partition to the current table. If you add a partition to an existing table and then commit the table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Field type	The data type of the partition field. We recommend that you use the STRING type for all partition fields.
	Date partition format	The format of the date partition. If the partition field is a date, although the data type may be STRING, select or enter a date format, for example, *yyyymmdd* or *yyyy-mm-dd*.

Section	GUI element	Description
Note This section appears only when you set Partition type to Partition Table in the Physical model design section.	**Date partition granularity**	The granularity of the date partition. The granularity can be second, minute, hour, day, month, quarter, or year. You can enter a partition granularity as required. If you want to add multiple partitions with different granularities, note that a greater granularity corresponds to a higher partition level. For example, if you add three partitions whose granularities are day, hour, and month, respectively, to the table, the table contains three partitions: a level-1 partition (month), a level-2 partition (day), and a level-3 partition (hour).
	Delete icon	The icon for deleting the partition. If you delete a partition from an existing table and then commit the table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.

6. Click **Submit to development environment** and **Submit to production environment** in sequence.

   If you are using a workspace in basic mode, you need to click **Submit to production environment** only.

Button	Description
Load from development environment	If the table has been committed to the development environment, this button is clickable. After you click this button, the information about the table you create in the development environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Submit to development environment	Before you click this button, make sure that you have filled in all required parameters on the table configuration tab. You cannot click this button if any parameter is not specified.
Load from production environment	After you click this button, the information about the table that is committed to the production environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Submit to production environment	After you click this button, the table is created in the workspace of the production environment.

What's next

After the table is created, you can query the table data and modify or delete the table. For more

information, see Manage tables.

5.9. Create MaxCompute resources

This topic describes how to create, reference, and download JAR and Python resources.

Prerequisites

A MaxCompute compute engine is associated with a workspace on the **Workspace Management** page so that the MaxCompute service is displayed in the workspace. For more information, see Configure a workspace.

Context

If your code or function requires resources such as.jar files, you can upload the resources to your workspace and reference them.

If the existing built-in functions do not meet your requirements, DataWorks allows you to create user-defined functions (UDFs) and customize processing logic. You can upload the required JAR packages to your workspace so that you can reference them when you create UDFs.

> **Note**
> - You can view the built-in functions on the **Built-In Functions** tab. For more information, see Functions.
> - You can view the UDFs that you have committed or deployed in DataWorks on the **MaxCompute Functions** tab. For more information, see MaxCompute functions.

You can upload different types of resources including text files, Python code, and compressed packages in the .zip, .tgz, .tar.gz, .tar, and .jar formats to MaxCompute. You can read or use these resources when you run UDFs or MapReduce jobs.

MaxCompute provides interfaces for you to read and use resources. The following types of resources are supported:

- **Python**: the Python code you have written. You can use Python code to register Python UDFs.
- **JAR**: the compiled Java JAR packages.
- **Archive**: the compressed files that have different file name extensions. Supported file types include .zip, .tgz, .tar.gz, .tar, and .jar.
- **File**: the files in the .zip, .so, or .jar format.

JAR resources and **file** resources have the following differences:

- You can write Java code in an offline Java environment, compress the code to a JAR package, and then upload the package as a JAR resource to DataWorks.
- You can create and edit a small-sized file resource in the DataWorks console.
- To upload a file resource that is larger than 500 KB from your computer, you can select **Large File (over 500 KB)** when you create the file resource.

> **Note** Each resource to be uploaded cannot exceed 30 MB. You can use the MaxCompute client to upload a resource that is larger than 30 MB. Then, commit it to DataWorks on the MaxCompute Resources tab. For more information, see MaxCompute resources.

Create a JAR resource

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **Create** icon and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can click the required workflow in the **Business Flow** section, right-click **MaxCompute**, and then choose **Create > Resource > JAR**.

   For more information about how to create a workflow, see Create a workflow.
3. In the **Create Resource** dialog box, set the **Resource Name** and **Location** parameters.

   > **Note**
   > - If the selected JAR package has been uploaded from the MaxCompute client, clear **Upload to MaxCompute**. If you do not clear it, an error occurs during the upload process.
   > - The resource name can be different from the name of the uploaded file.
   > - The resource name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). The name is not case-sensitive. A JAR resource name must end with .jar, and a Python resource name must end with .py.
4. Click **Upload** and select the file that you want to upload.
5. Click **Create**.
6. Click the icon in the toolbar to commit the resource to the development environment.

   If the workspace that you use is in standard mode, you must click Deploy in the upper-right corner after you commit the node. For more information, see Deploy nodes.

Create a Python resource and register a UDF

1. Create a Python resource.
   i. On the **Data Analytics** tab, move the pointer over the **Create** icon and choose **MaxCompute > Resource > Python**.

   Alternatively, you can click the required workflow in the **Business Flow** section, right-click **MaxCompute**, and then choose **Create > Resource > Python**.
   ii. In the **Create Resource** dialog box, set the **Resource Name** and **Location** parameters.

   > **Notice** The resource name can contain letters, digits, periods (.), underscores (_), and hyphens (-). It must end with .py.
   iii. Click **Create**.

iv. Enter the code of the created Python resource in the code editor. Sample code:

```
from odps.udf import annotate
@annotate("string->bigint")
class ipint(object):
    def evaluate(self, ip):
        try:
            return reduce(lambda x, y: (x << 8) + y, map(int, ip.split('.')))
        except:
            return 0
```

v. Click the icon in the toolbar to commit the node.

If the workspace that you use is in standard mode, you must click Deploy in the upper-right corner after you commit the node. For more information, see Deploy nodes.

2. Register a UDF.

i. On the **Data Analytics** tab, move the pointer over the **+Create** icon and choose **MaxCompute > Function**.

Alternatively, you can click the required workflow, right-click **MaxCompute**, and then choose **Create > Function**.

ii. In the **Create Function** dialog box, set the **Function Name** and **Location** parameters.

iii. Click **Create**.

iv. In the **Register Function** section, enter the class name of the function and the name of the Python resource that has been created, and then click the icon in the toolbar. In this example, the class name is `ipint.ipint`.

v. Check whether the ipint function is valid and meets your expectation. You can create an ODPS SQL node in the DataWorks console and test the ipint function by running an SQL statement.

You can also create an ipint.py file in your computer and upload it by using the MaxCompute client. For more information, see Client.

```
odps@ MaxCompute_DOC>add py D:/ipint.py;
OK: Resource 'ipint.py' have been created.
```

```
odps@ MaxCompute_DOC>create function ipint as ipint.ipint using ipint.py;
Success: Function 'ipint' have been created.
```

After the resource is uploaded, register a UDF on the MaxCompute client. For more information, see Functions operations. You can use the UDF after it is registered.

Reference and download resources

- For more information about how to reference resources in a function, see Functions operations.
- For more information about how to reference resources in a node, see Create an ODPS MR node.

To download a resource, double-click **Resource**, select the required resource, and then click **Download**. For more information about how to download a resource by using the MaxCompute client, see Resource operations.

Other operations

After a resource is created, click the required workflow in the Business Flow section, choose MaxCompute > Resource, and then right-click the created resource to rename, reference, or delete the resource.**MaxCompute > Resource** For more information about how to delete a resource, see DataStudio.

5.10. Create a MaxCompute function

DataWorks allows you to develop functions in Python and Java. This topic describes how to create a MaxCompute function.

Prerequisites

Required resources are uploaded.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Create a workflow. For more information, see Create a workflow.
3. Create JAR or Python resources, and commit and deploy them. For more information, see Create a MaxCompute resource.
4. Create a function.
   i. Find the created workflow, right-click **MaxCompute**, and then choose **Create > Function**.
   ii. In the **Create Function** dialog box, set the **Function Name** and **Location** parameters.
   iii. Click **Create**.
   iv. In the **Register Function** section, set the parameters.

Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Function**, **Aggregate Function**, **String Function**, **Date Function**, **Analytic Function**, and **Other Functions**.
Engine Instance MaxCompute	By default, the parameter cannot be modified.
Function Name	The name of the user defined function (UDF). You can reference the function in SQL statements by using the function name. The function name must be globally unique and cannot be changed after the function is created.
Owner	This parameter is automatically set.
Class Name	Required. The name of the class that implements the function. **Note** If the resource type is Python, enter the class name in the `Python resource name.Class name` format. Do not include the .py extension in the resource name. You can use a Python resource after the resource is committed and deployed. For more information, see Create a MaxCompute resource.
Resources	Required. You can search for existing resources in the current workspace in fuzzy match mode.
Description	The brief description of the current UDF.
Expression Syntax	The syntax of the UDF. Example: `test`.
Parameter Description	The descriptions of the input and output parameters that are supported.
Return Value	Optional. The value to return. Example: 1.
Example	Optional. An example of the function.

5. Click the icon in the toolbar.
6. Commit the function.
   i. Click the icon in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

6.View Job Running Information

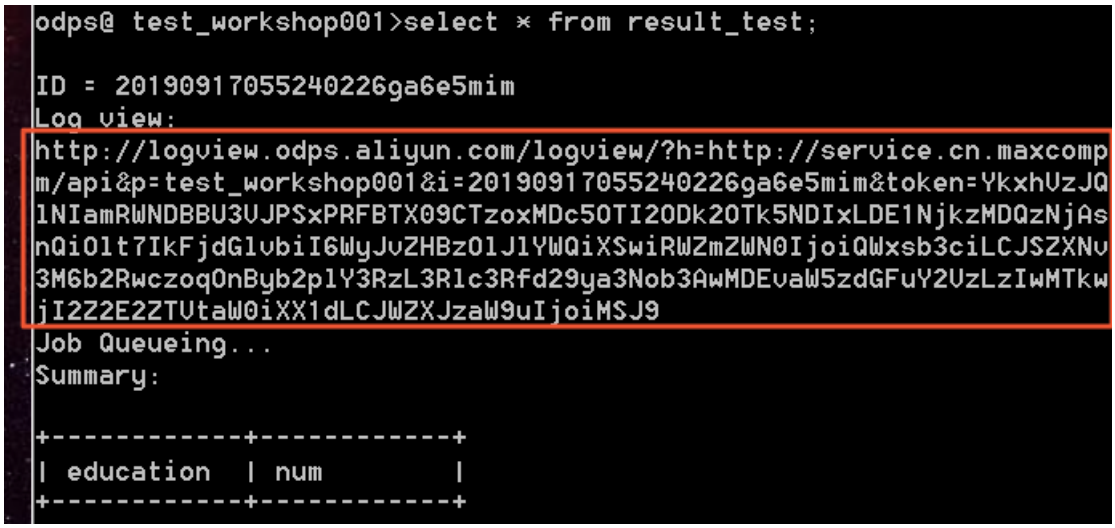
6.1. Use Logview to view job information

Logview is a tool that you can use to view and debug jobs after you submit jobs to MaxCompute.

You can use Logview to view the following information of a job:

- Task status
- Task results
- Details of each task and the progress of each step

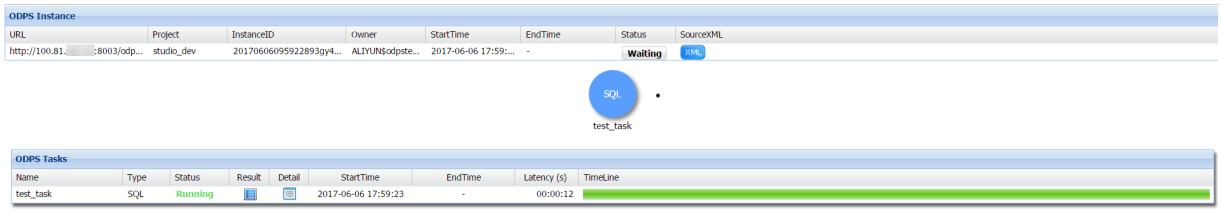
After a job is submitted to MaxCompute, the system generates a Logview URL. You can enter the Logview URL on your web browser and press Enter to view job information.



> **Note** The Logview page of each job is valid for seven days.

UI elements

This section describes elements on the Logview UI.



The Logview page consists of two sections:

- ODPS Instance
- ODPS Tasks

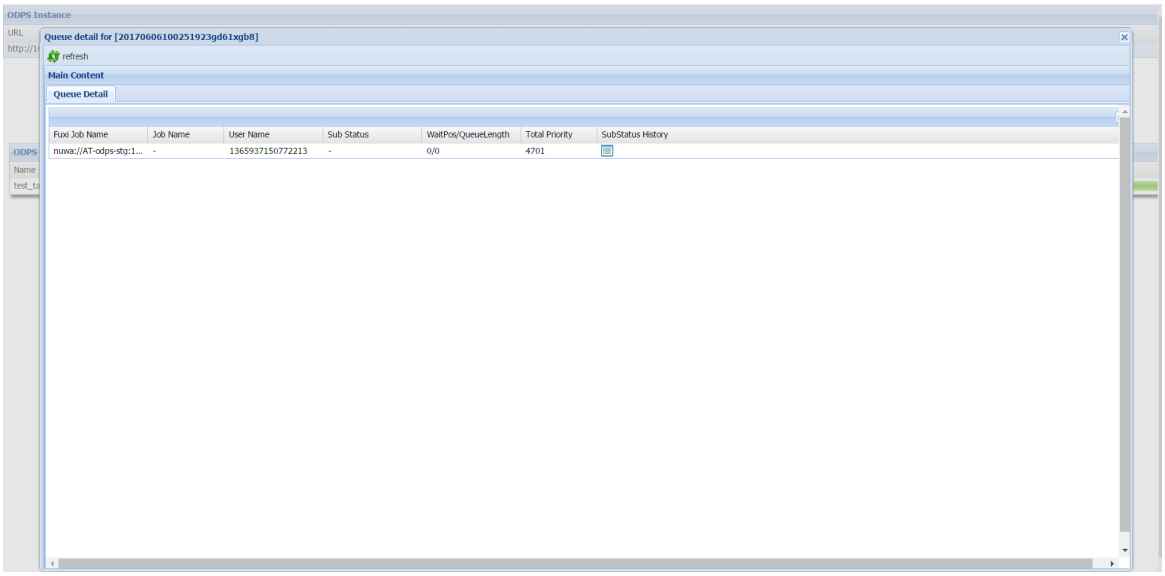
ODPS Instance

The ODPS Instance section shows the information about the instance that corresponds to the submitted SQL job. The information includes URL, Project, InstanceID, Owner, StartTime, EndTime, and Status.

- You can click the value in the Status column to view queue information. Valid values of Status:
  - Waiting: The job is being processed in MaxCompute and is not submitted to Job Scheduler.
  - Waiting List: n: The job is submitted to Job Scheduler and is queued. n indicates the order number of the job in the queue.
  - Running: The job is running in Job Scheduler.

  > **Note** If the value of Status is Terminated, the job is terminated and no queue information is available.

- After you click the value of Status, the following queue information is displayed:
  - Sub Status: the current sub-status information.
  - WaitPos: the position of the job in the queue. The value `0` indicates that the job is running. The value `-` indicates that the job is not submitted to Job Scheduler.
  - QueueLength: the total queue length in Job Scheduler.
  - Total Priority: the running priority assigned by the system.
  - SubStatus History: You can click the icon in this column to view the status history. The status history includes the status code, status description, start time, and duration of the state of a job. The information is unavailable in some versions.

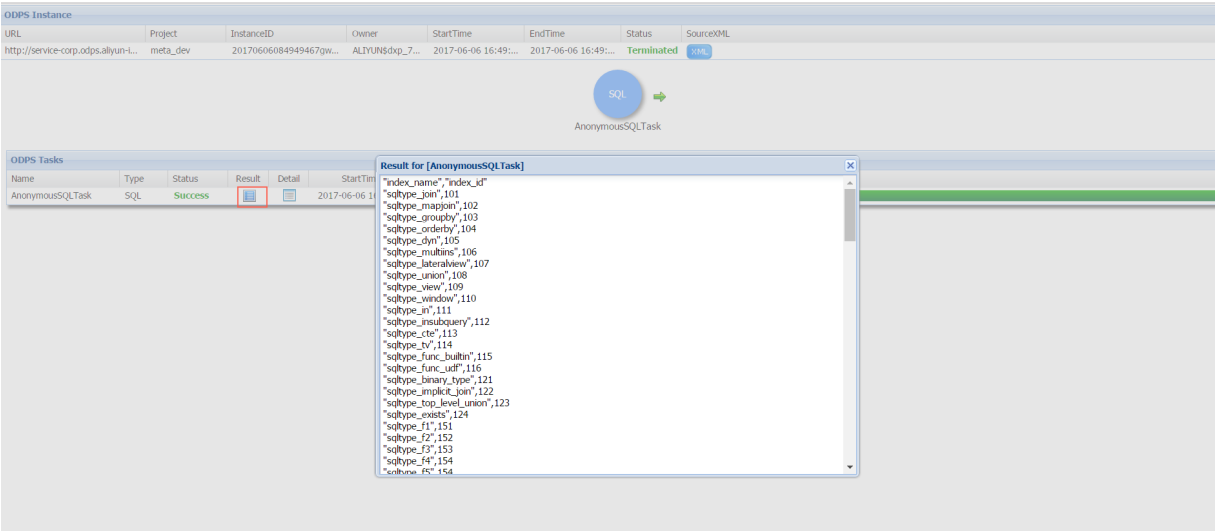


ODPS Tasks

The ODPS Tasks section shows the information about the task that corresponds to the instance. The information includes Name, Type, Status, Result, Detail, StartTime, EndTime, Latency (s), and TimeLine. Latency (s) in the ODPS Tasks section indicates the total running duration.

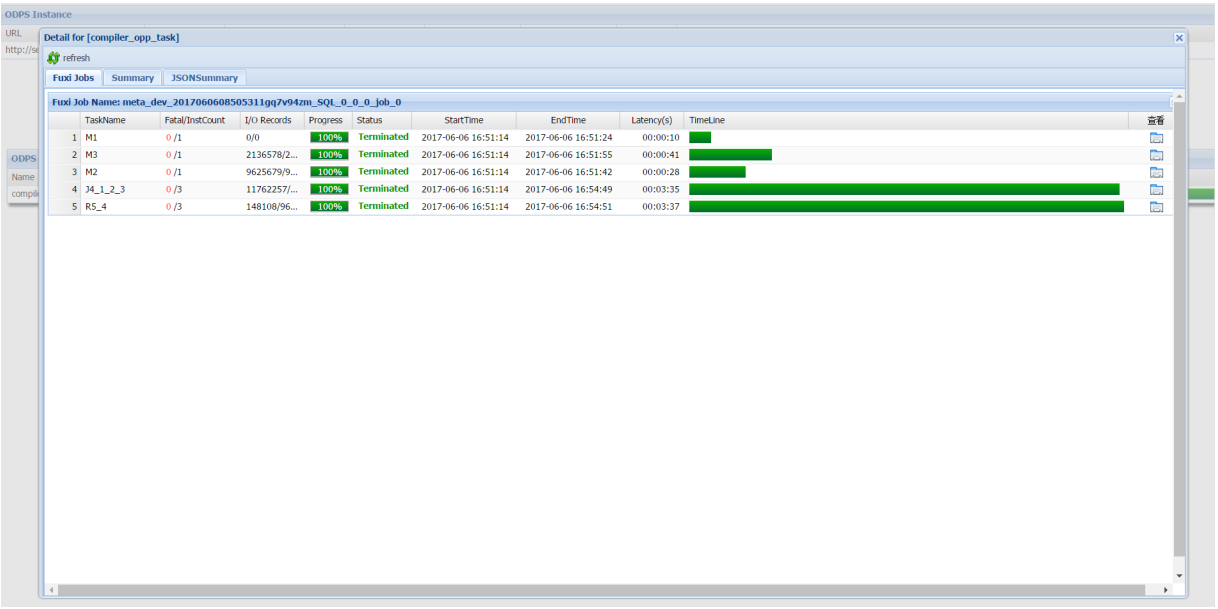
Result:

After a job is completed, you can click the icon in the Result column to view the job results. The following figure shows the execution results of a SELECT statement.



Detail:

You can click the icon in the **Detail** column to view the running details. You can view the details of both running and completed jobs.



In the dialog box that shows the details of a MaxCompute task, you can view the following information:

- A MaxCompute job consists of one or more Fuxi jobs. For example, if a complex SQL job is submitted, MaxCompute automatically submits multiple Fuxi jobs to Job Scheduler.
- Each Fuxi job consists of one or more Fuxi tasks. For example, a simple MapReduce job generates two Fuxi tasks: map task (M1) and reduce task (R2). If an SQL job is complex, multiple Fuxi tasks may be generated.
- The name of a Fuxi task. In most cases, a task name consists of letters and digits. A letter represents the type of a task. For example, M indicates a map task. The digits that follow the letter represents the ID and dependencies of a task. For example, R5_4 indicates that the reduce task can be executed only after the J4 task is completed. J4_1_2_3 indicates that the join task can be executed only after

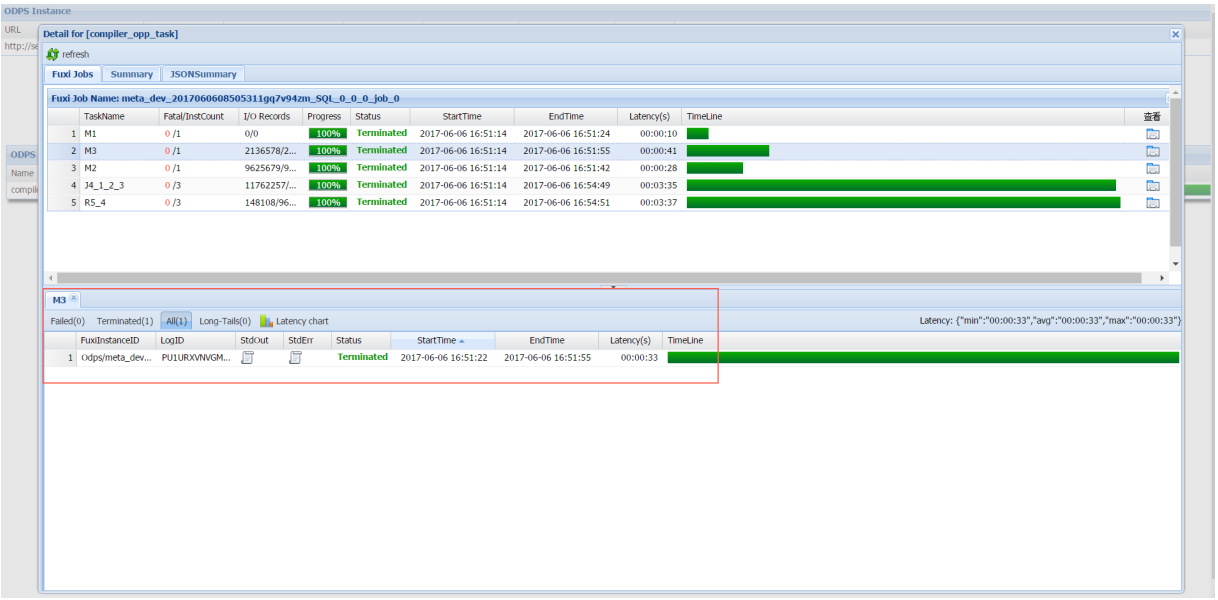
the M1, M2, and M3 tasks are completed.

> **Note** I/O Records indicates the numbers of the input and output records of a Fuxi task.

- I/O Records indicates the numbers of the input and output records of this task.

To view the information about the instance that corresponds to a Fuxi task, you can click the icon in the Show Detail column of the Fuxi task or double-click the Fuxi task.

> **Note** Each Fuxi task consists of one or more Fuxi instances. If the amount of input data for a Fuxi task increases, MaxCompute starts more nodes for the task to process the data. A node corresponds to a Fuxi instance.



In the lower part of the dialog box, the status information about Fuxi instances at different stages is displayed in groups. For example, you can click the Failed tab to view the nodes on which errors occurred. You can click the icon in the StdOut column to view standard output information. You can also click the icon in the StdErr column to view standard error information.

> **Note** To-be-displayed information written in the submitted MaxCompute job is also displayed in the standard output information and standard error information.

Use Logview to handle issues

- **Tasks with errors**

  If an error occurs during a task, find the task and click the icon in the Result column in the lower part of the Logview page to view the error information. You can also click the icon in the StdErr column of a Fuxi instance in the details dialog box to view the error information of the instance.

- **Data skew**

  The long tail of one or more Fuxi instances may decelerate the execution of a Fuxi task. The long tail is caused by uneven data distribution in a task. After a task is completed, you can view the running results on the Summary tab of the details dialog box. The following output provides an example of the running results.

```
output records:
R2_1_Stg1: 199998999  (min: 22552459, max: 177446540, avg: 99999499)
```

If a large difference exists between the min and max values, data skew occurs. For example, if a specific value appears more often than other values in a column, data skew occurs when you execute a JOIN operation based on this column.

6.2. Use Logview V2.0 to view job information

This topic describes the entry point and features of Logview V2.0. You can use Logview V2.0 to view job information.

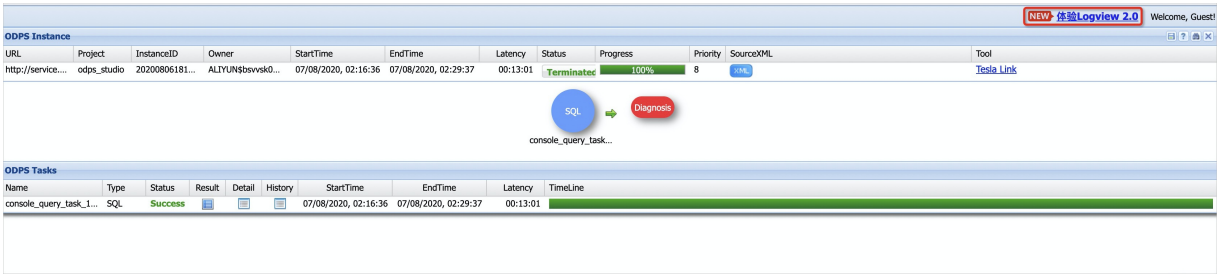
Overview

Logview V2.0 provides a newly designed UI, increases the loading speed, and delivers the following new features:

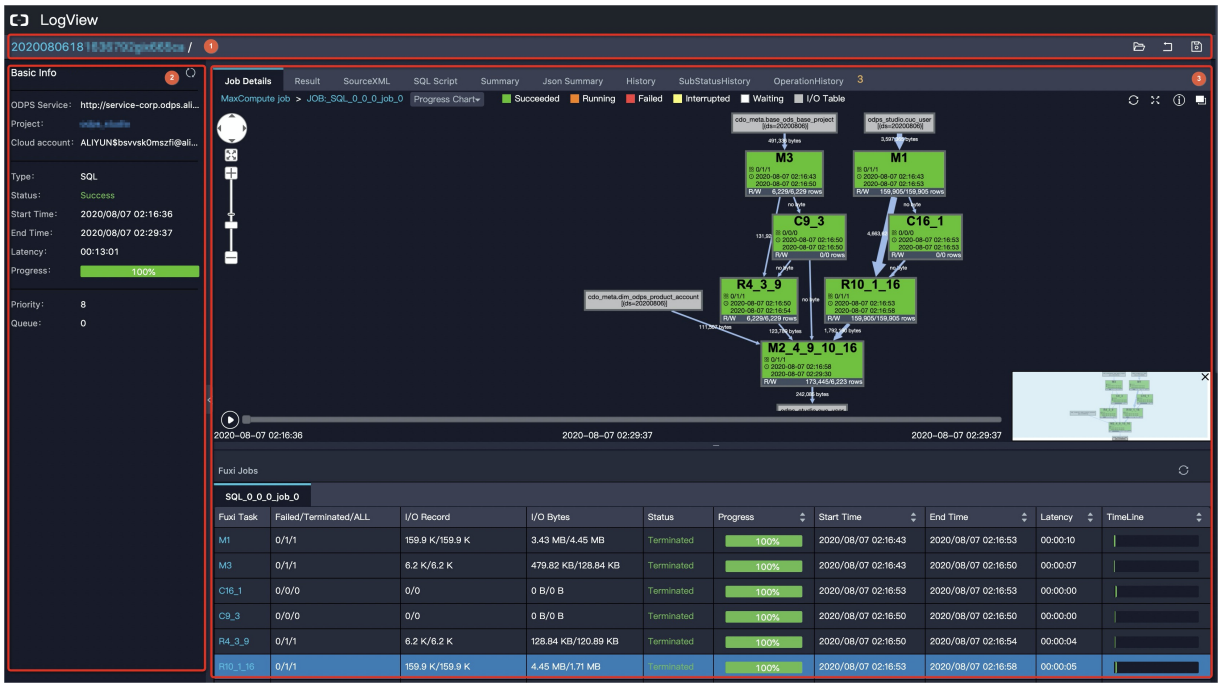
- Provides an interactive directed acyclic graph (DAG) to display the logical architecture of job processing. You can also view the operators of a job.
- Supports job running playback.
- Allows you to view memory usage and CPU utilization by using Fuxi Sensor.

Entry point

After you use the MaxCompute client (odpscmd) to submit a job, the system generates a Logview URL. To go to Logview V2.0, you can perform the following operations: Enter the Logview URL in the search engine and press Enter. On the Logview UI, click **Go to Logview V2.0**.



Logview V2.0 page



No.	Section
1	The title and functionality section. For more information, see Title and functionality section.
2	The Basic Info section. For more information, see Basic Info section.
3	The job details section. For more information, see Job details section.

Title and functionality section

This section shows the job ID and job name. The job ID uniquely identifies a MaxCompute job. The job ID is generated when you submit the job. The job name is displayed only if the job is submitted by using an SDK. You can also click the icons on the right of this section to perform operations.

Icon	Description
	Open the Logview_detail.txt file that contains job details. The file is saved on your computer.
	Return to the original Logview UI.
	Save the job details as a file to your computer.

Basic Info section

This section shows the basic information about a job.

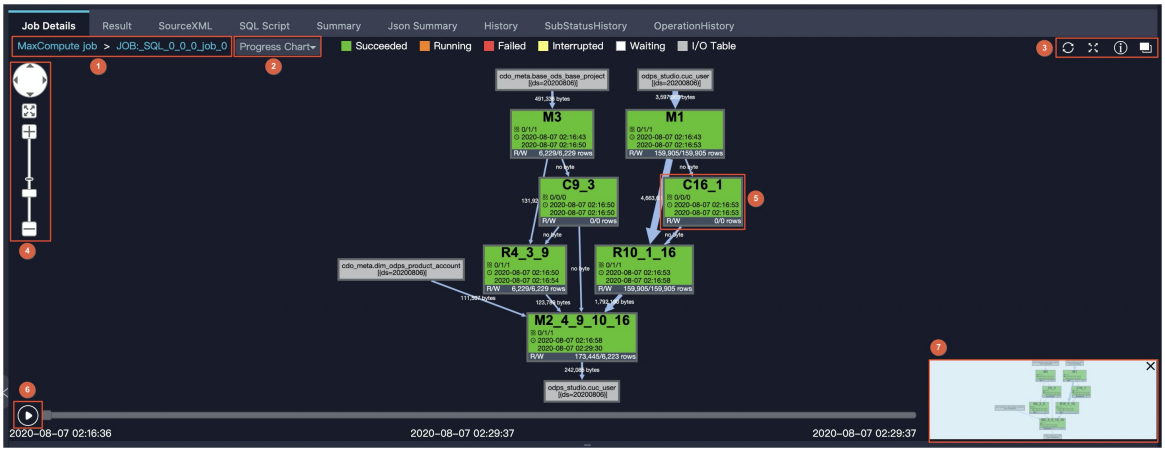
Parameter	Description
MaxCompute Service	The endpoint of MaxCompute on which the job runs. For more information, see Endpoints.
Project	The name of the MaxCompute project to which the job belongs.
Cloud account	The Alibaba Cloud account that is used to submit the job.
Type	The type of the job. Valid values: SQL, SQLRT, LOT, XLib, CUPID, AlgoTask, and Graph.
Status	The status of the job. Valid values: • Success: The job succeeds. • Failed: The job fails. • Canceled: The job is canceled. • Waiting: The job is being processed in MaxCompute but is not submitted to Job Scheduler. • Running: The job is being processed in Job Scheduler. • Terminated: The job is completed.
Start Time	The time at which the job was submitted.
End Time	The time at which the job is completed.
Latency	The period during which the job is run.
Progress	The progress of the job.
Priority	The priority of the job.
Queue	The position of the job in the queue in the resource quota group.

Job details section

In the job details section, you can query details about a job. This section consists of the following tabs:

- **Job Details**
  - Progress chart

In the upper part of the **Job Details** tab, the progress chart of a job is displayed. The progress chart shows the subtask dependencies from three dimensions: Fuxi jobs, Fuxi tasks, and operators. It also provides a series of tools to help you locate issues. The following figure shows the upper part of the Job Details tab.



No.	Description
1	The breadcrumb navigation that is used to switch Fuxi jobs. JOB:_SQL_0_0_0_job_0 is the name of a Fuxi job.
2	The troubleshooting tool. You can use Progress Chart, Input Heat Chart, Output Heat Chart, TaskTime Heart Chart, and InstanceTime Heart Chart for troubleshooting.
3	You can click the icon to refresh the job status and click the icon to zoom in or out on the progress chart. You can also click the icon to obtain MaxCompute Studio documentation and the icon to switch to the upper level of the job.
4	The zoom tool.

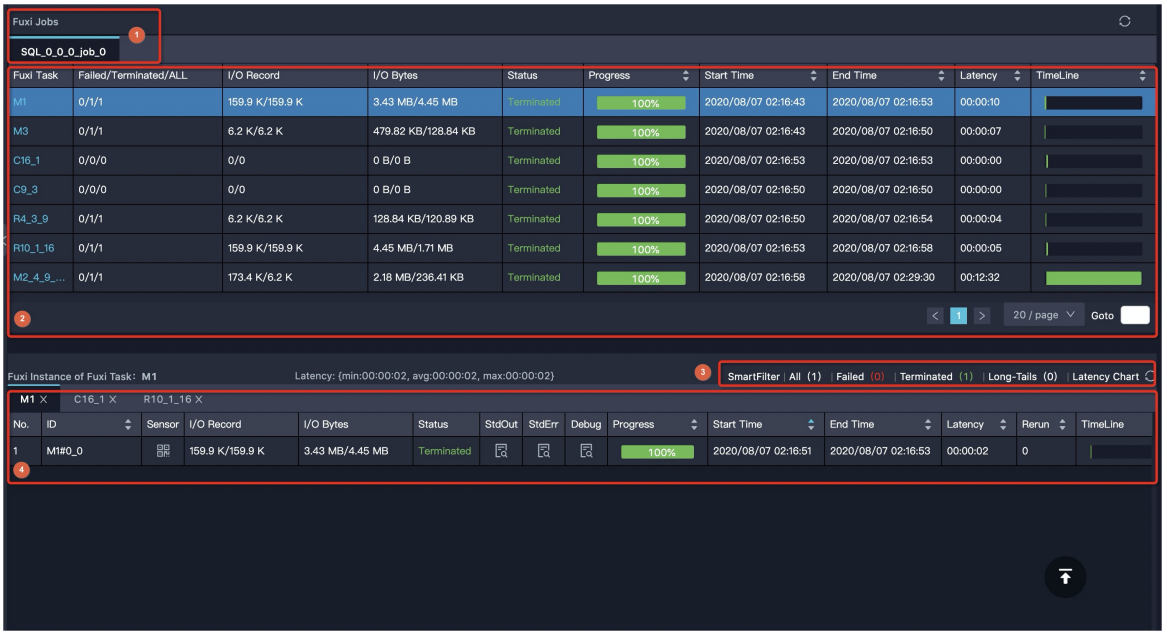
No.	Description
5	The Fuxi task. A MaxCompute job consists of one or more Fuxi jobs. Each Fuxi job consists of one or more Fuxi tasks. Each Fuxi task consists of one or more Fuxi instances. If the amount of input data increases, MaxCompute starts more nodes for each task to process the data. A node is equivalent to a Fuxi instance. For example, a simple MapReduce job generates two Fuxi tasks: map task (M1) and reduce task (R2). If an SQL statement is complex, multiple Fuxi tasks may be generated. You can view the name of each Fuxi task on the execution page. For example, M1 indicates a map task. The 3 and 9 fields in R4_3_9 indicate that the map task can be executed only after M3 and C9_3 are completed. Similarly, M2_4_9_10_16 indicates that the M2 task can be executed only after R4_3_9, C9_3, R10_1_16, and C16_1 are completed. R/W indicates the numbers of rows that the task reads and writes. Click or right-click a task to view the operator dependencies and operator graphs of the task. Logview V2.0 provides table dependencies so that you can view the input and output tables.
6	The Fuxi task playback. You can click the icon to start or stop the playback. You can also drag the progress bar. The start time and end time are displayed on the sides of the progress bar. The playing time is displayed in the middle.
7	The thumbnail.

Note

- The playback feature is not applicable to Fuxi tasks that are in the Running state.
- An AlgoTask job, such as a Machine Learning Platform for AI (PAI) job, contains only one Fuxi task. Therefore, no progress charts are provided for these jobs.
- For non-SQL jobs, only Fuxi jobs and Fuxi tasks are displayed.
- If only one Fuxi job exists, the progress chart shows the dependencies among Fuxi tasks. If multiple Fuxi jobs exist, the progress chart shows the dependencies among the Fuxi jobs.

- Job status

In the lower part of the **Job Details** tab, detailed information about the job is displayed. The following figure shows the upper part of the Job Details tab.



No.	Description
1	The Fuxi Jobs tab. You can switch Fuxi jobs on this tab.
2	The details about the Fuxi tasks of the Fuxi job. Click a Fuxi task to display information about the Fuxi instance of this task. By default, information about the Fuxi instance of the first Fuxi task for the first Fuxi job is displayed. For AlgoTask jobs, the **Sensor** column appears in this section. You can click the sensor that corresponds to a Fuxi task to view the CPU and memory information of the Fuxi instance. For more information, see Fuxi Sensor. **Note** Fuxi Sensor is available in the China (Chengdu), China (Shenzhen), China (Shanghai), China (Hangzhou), China (Zhangjiakou), and China (Beijing) regions.
3	Logview divides instances into groups based on their status. You can click the value next to Failed to query information about faulty nodes.
4	Fuxi instances. ■ StdOut and StdErr. You can view the output messages, error messages, and information that you want to display. You can also download the information. ■ Debug. You can debug and troubleshoot errors.

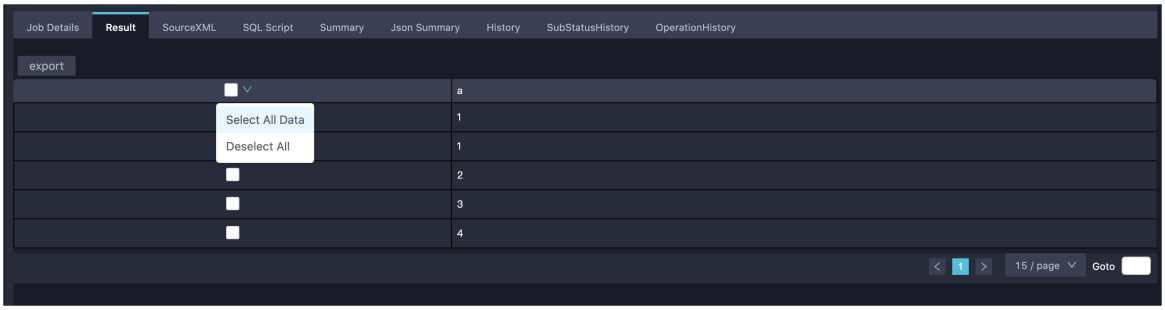
- **Result**

  This tab shows the result of a job. If a job fails, this tab appears and shows the cause of the failure.

> **Note** MaxCompute allows you to configure the project-level parameter `odps.forbid.fetch.result.by.bearertoken` to specify whether to display the results of jobs on the **Result** tab of Logview. For more information about the parameters, see Project operations.

The system displays data in plain text or a table based on the response format specified in MaxCompute. You can log on to the MaxCompute client and configure odps.sql.select.output.format to specify the format. If you set this parameter to HumanReadable, the result is displayed in plain text. Otherwise, the result is displayed in the CSV format.

- Table

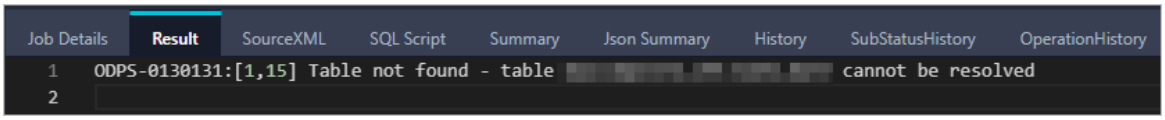


> **Note** NULL is represented as `\N` in the table. \N can be distinguished from NULL strings and is compatible with the display mode of Logview V1.0.

You can perform the following operations on this tab:

- Select a row to export data in the row.
- Select the table heading to export data from the current page.
- In the table heading, click the icon and select **Select All Data** or **Deselect All** based on your business requirements.
- Click **export** to export data in the CSV format. CSV is a file name extension. You can use a text file or Sublime Text to open CSV files.

- Plain text



- **SourceXML**

  This tab shows the source XML information about the job that is submitted to MaxCompute.

- **SQL Script**

  This tab shows the SQL script of the job that is submitted to MaxCompute.

- **Summary**

  This tab shows the overall information about the job that is submitted to MaxCompute. You can also view the memory usage and CPU utilization of the job on this tab.

- **Json Summary**

  This tab shows the overall information about the job in the JSON format.

- **History**

  This tab shows the historical information about a Fuxi instance if the instance is run multiple times.

- **Substatus History**

  This tab shows the detailed information about a job, including the status code, status description, start time, duration, and end time.

Fuxi Sensor

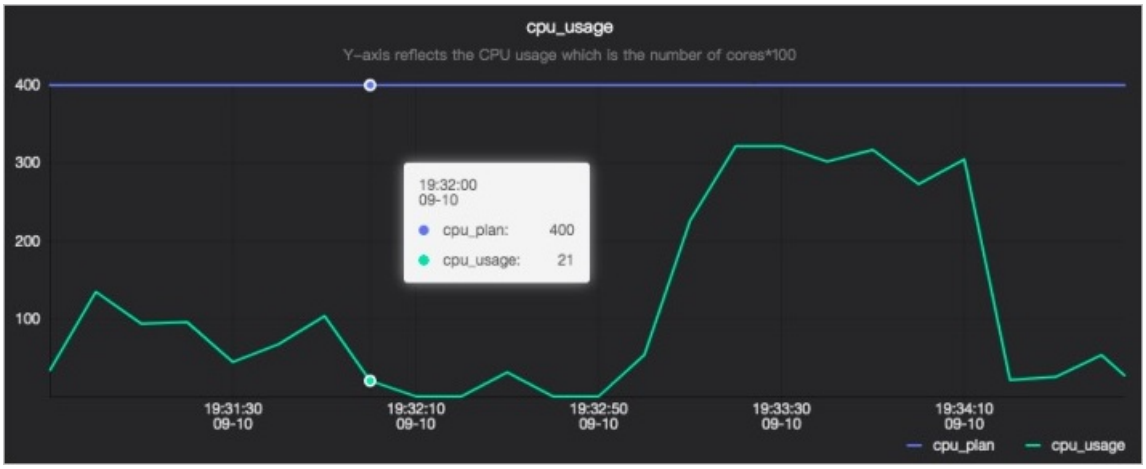
Fuxi Sensor is a resource view that shows a MaxCompute job in all dimensions. You can use Fuxi Sensor to view the memory usage and CPU utilization of a Fuxi instance. You can also use Fuxi Sensor to locate job issues and analyze job performance. For example, you can use Fuxi Sensor in the following scenarios:

- If an out-of-memory (OOM) error occurs, analyze the size of memory used.
- Compare the numbers of requested and used resources to optimize the resource request process.

For example, you can use Fuxi Sensor to view the resource usage of a Fuxi instance.

- CPU utilization

  The cpu_usage chart has two lines. One indicates the number of CPUs requested (cpu_plan), and the other one indicates the number of CPUs used (cpu_usage). In the y-axis, 400 indicates four processors.

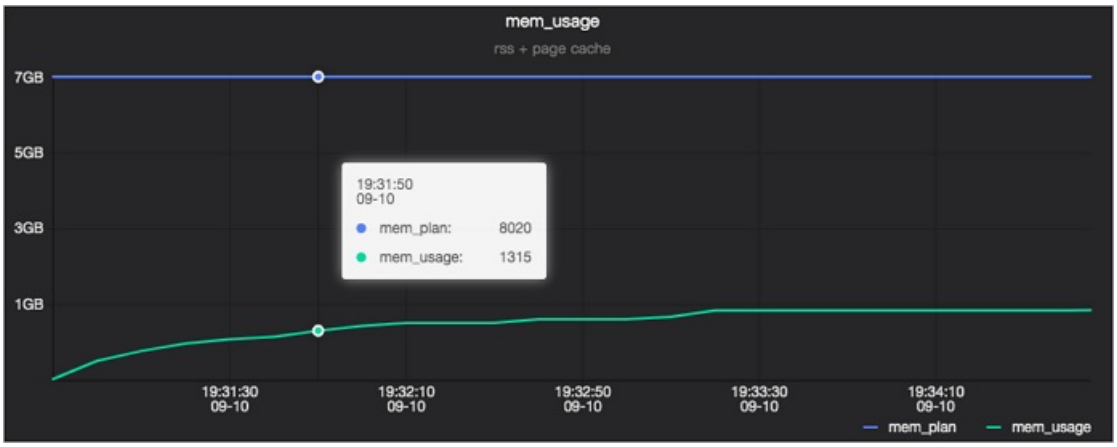


- Memory usage

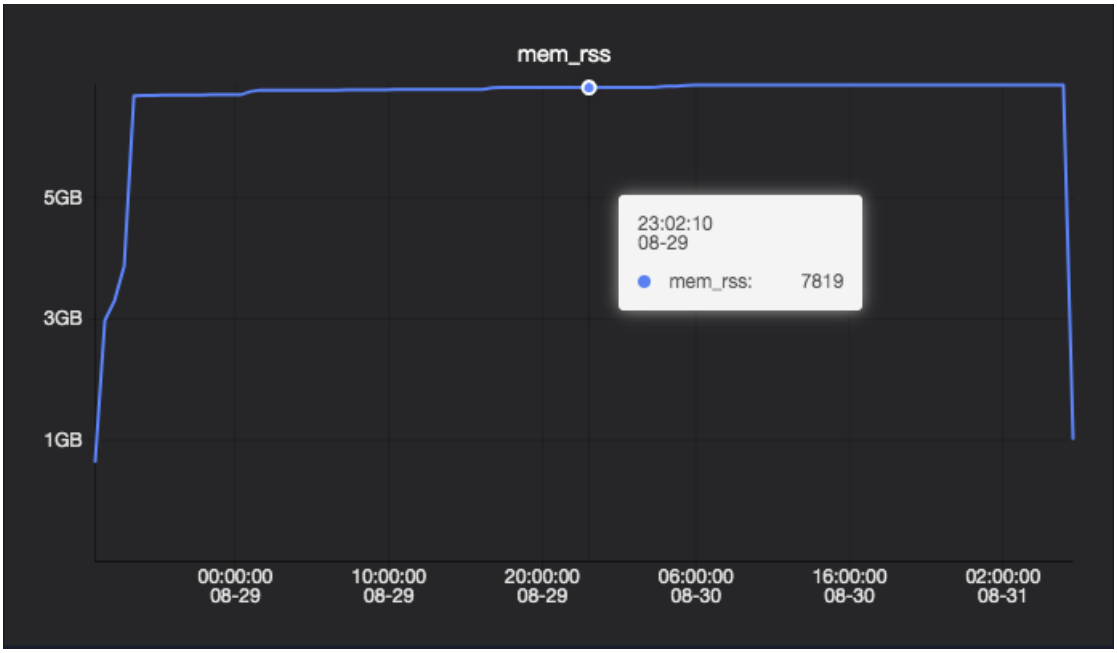
  The mem_usage chart has two lines. One indicates the number of memory resources requested (mem_plan), and the other one indicates the number of memory resources used (mem_usage).

  mem_usage contains Resident Set Size (RSS) and PageCache. RSS indicates the memory that is allocated after kernel page faults occur. This applies when you call Malloc to request memory by using non-file mappings. If the memory is insufficient, RSS cannot be reclaimed. PageCache is the memory occupied by the kernel to cache the files that are required by read and write requests, such as log files. If the memory is insufficient, PageCache can be reclaimed.

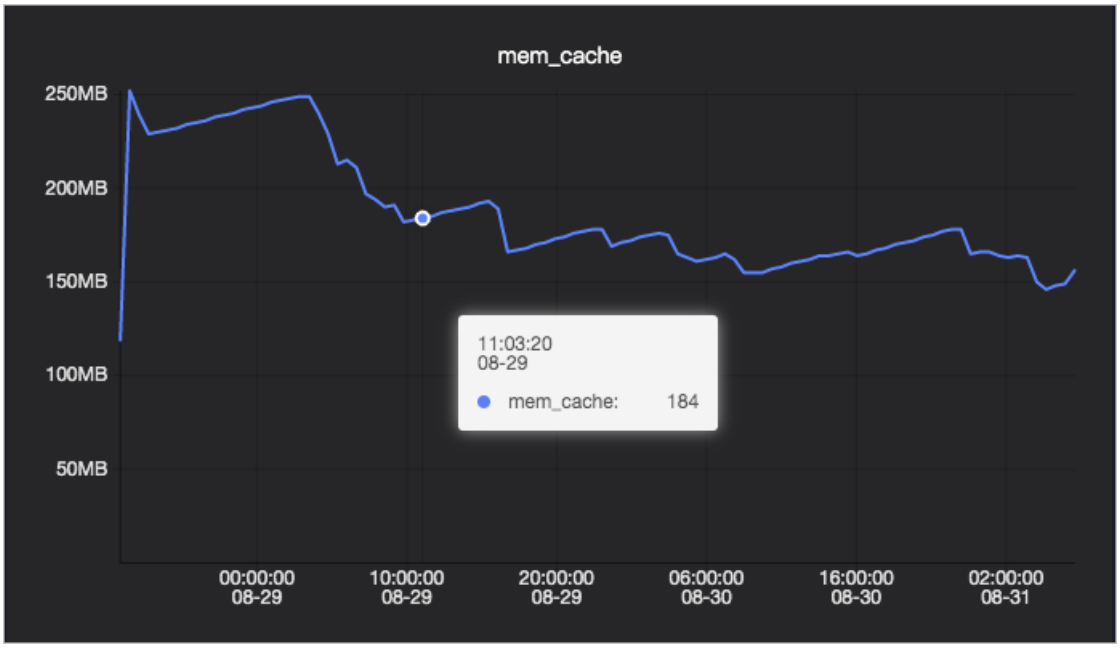
- Memory details



- RSS usage



- PageCache usage

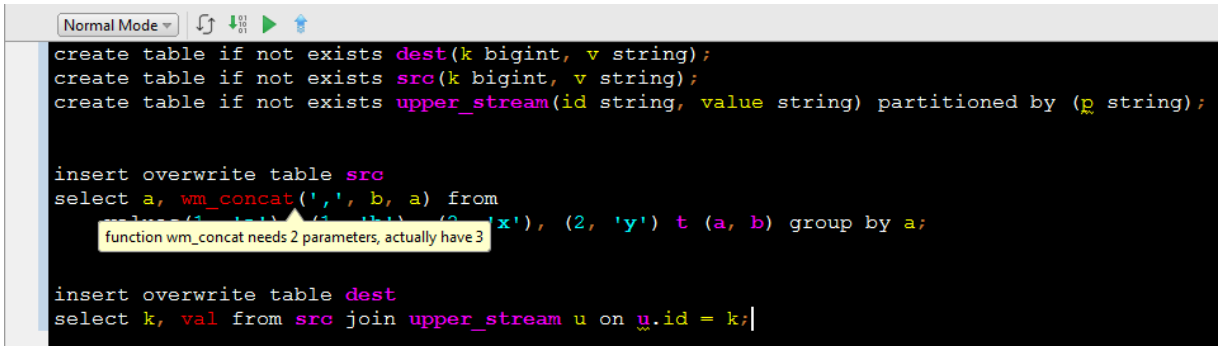


6.3. Use errors and alerts in the MaxCompute compiler for troubleshooting

The MaxCompute compiler is based on the new-generation SQL engine of MaxCompute V2.0. This simplifies the SQL compilation and improves language expressions. This topic describes how to use errors and alerts in the MaxCompute compiler for troubleshooting.

We recommend that you use MaxCompute Studio in combination with the compiler. This improves the usability of the MaxCompute compiler.

Install MaxCompute Studio, add a MaxCompute project, create a MaxCompute Studio project, and then create a MaxCompute script file.

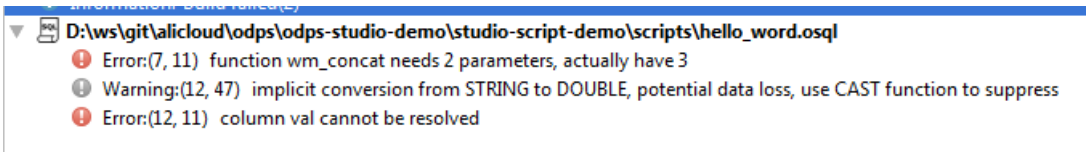


In the preceding figure, the following errors exist:

- In the first INSERT statement, an error occurs on the use of the wm_concat function.
- In the second INSERT statement, a column name error occurs and a data type conversion alert is generated. In MaxCompute, if a BIGINT value is compared with a DOUBLE value, an implicit conversion

to the DOUBLE type is performed. A conversion error may occur. Therefore, the MaxCompute compiler generates an alert for you to check whether the conversion meets your expectation.

Move the pointer over an error or alert. The details of the error or alert are displayed. If you ignore the error or alert and submit your script, the script is blocked by MaxCompute Studio.

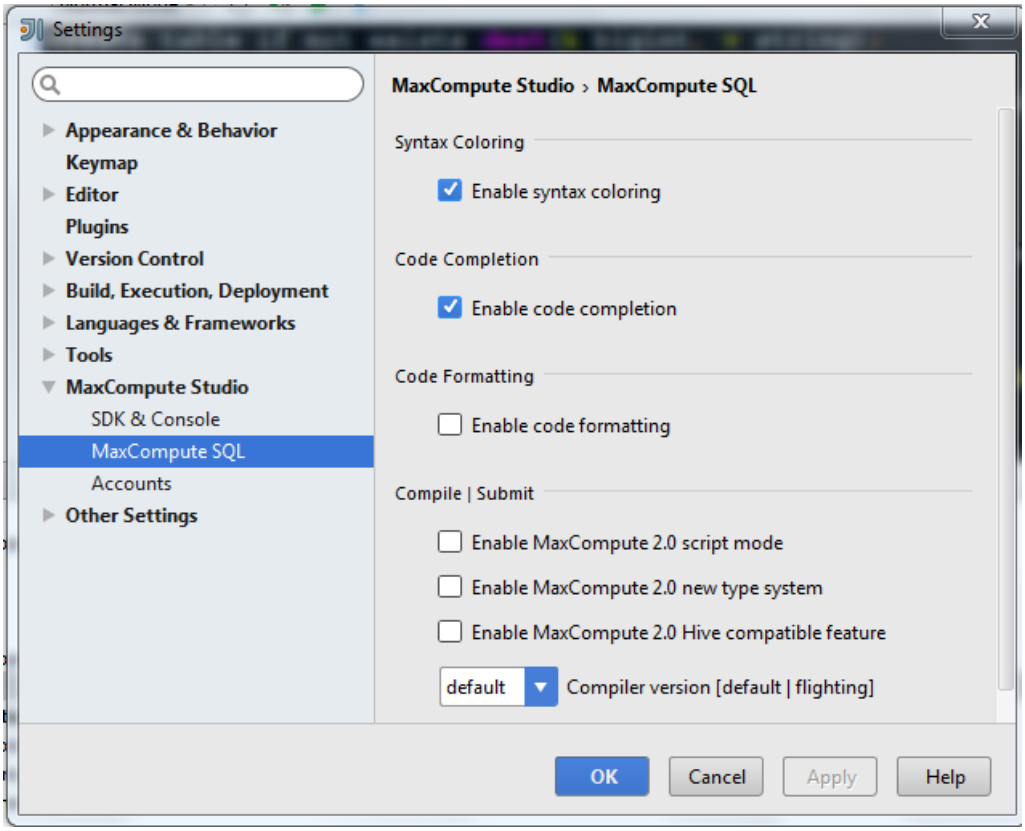


Therefore, you must handle the error or alert based on the message.

```
create table if not exists dest(k bigint, v string);
create table if not exists src(k bigint, v string);
create table if not exists upper_stream(id string, value string) partitioned by (dt string);


insert overwrite table src
select a, wm_concat(',', b) from
    values(1, 'a'), (1, 'b'), (2, 'x'), (2, 'y') t (a, b) group by a;


insert overwrite table dest
select k, value from src join upper_stream u on u.id = string(k);
```

After the modification, submit the script again.

You can also use MaxCompute Studio to specify all alerts as errors.



This way, no errors are missed.

Before you submit scripts, we recommend that you use MaxCompute Studio to perform static compilation checks for the scripts. We also recommend that you use MaxCompute Studio to specify alerts as errors and handle all alerts before you submit the scripts. This saves time and resources. If you submit a script with errors, your health score is reduced. This affects the priorities and order that you use to submit tasks. In later versions, unhandled alerts will be considered as a factor of the health score system. Therefore, you can fully utilize the details of the errors and alerts in the MaxCompute compiler to avoid priority downgrade.

Some alerts indicate the implicit type conversions that are not secure. If you are sure of a conversion, use **cast (xxx as)** to clear the alert. The MaxCompute compiler also allows you to use **(xxx)** to clear alerts.

7.Collect information for the optimizer of MaxCompute

MaxCompute uses a cost-based optimizer to accurately estimate the cost of each execution plan based on metadata, such as the number of rows and the average length of strings. This topic describes how to collect metadata for the optimizer so that you can use the optimizer to optimize query performance.

Background information

If the optimizer estimates the cost based on inaccurate metadata, the estimation result is inaccurate and a bad execution plan is generated. Therefore, accurate metadata is crucial to the optimizer. The core metadata of a table is the column stats metrics of the data in the table. Other metadata is estimated based on the column stats metrics.

MaxCompute allows you to use the following methods to collect column stats metrics:

- Analyze: an asynchronous collection method. You can run the `analyze` command to asynchronously collect column stats metrics. Active collection is required.

  > **Note** The version of the MaxCompute client must be later than 0.35.

- Freeride: a synchronous collection method. While data is generated in a table, the column stats metrics of the data are automatically collected. This method is automated but has an impact on query latency.

The following table lists the column stats metrics that can be collected for different data types.

Column stats metric/Data type	Numeric (TINYINT, SMALLINT, INT, BIGINT, DOUBLE, DECIMAL, and NUMERIC)	Character (STRING, VARCHAR, and CHAR)	Binary (BINARY)	Boolean (BOOLEAN)	Date and time (TIMESTAMP, DATE, and INTERVAL)	Complex (MAP, STRUCT, and ARRAY)
min (minimum value)	Y	N	N	N	Y	N
max (maximum value)	Y	N	N	N	Y	N
nNulls (number of null values)	Y	Y	Y	Y	Y	Y

Column stats metric/Data type	Numeric (TINYINT, SMALLINT, INT, BIGINT, DOUBLE, DECIMAL, and NUMERIC)	Character (STRING, VARCHAR, and CHAR)	Binary (BINARY)	Boolean (BOOLEAN)	Date and time (TIMESTAMP, DATE, and INTERVAL)	Complex (MAP, STRUCT, and ARRAY)
avgColLen (average column length)	N	Y	Y	N	N	N
maxColLen (maximum column length)	N	Y	Y	N	N	N
ndv (number of distinct values)	Y	Y	Y	Y	Y	N
topK (top K values with the highest frequency of occurrence)	Y	Y	Y	Y	Y	N

Note Y indicates that a metric is supported. N indicates that a metric is not supported.

Scenarios

The following table describes the use scenarios of each column stats metric.

Column stats metric	Optimization objective	Scenario	Description
		Scenario 1: Estimate the number of output records.	If only the data type is provided, the value range is too large for the optimizer. If the values of the min and max metrics are provided, the optimizer can more accurately estimate the selection of filter conditions and provide a better execution plan.

Column stats metric	Optimization objective	Scenario	Description
min (minimum value) or max (maximum value)	Enhance the accuracy of performance optimization.	Scenario 2: Push filter conditions down to the storage layer to reduce the amount of data that needs to be read.	In MaxCompute, the filter condition `a < -90` can be pushed down to the storage layer, whereas the filter condition `a + 100 < 10` cannot be pushed down. If the overflow of a is considered, the two filter conditions are not equivalent. However, if a has a maximum value, the filter conditions are equivalent and can be converted to each other. Therefore, the min and max metrics can enable more filter conditions to be pushed down. This reduces the amount of data that needs to be read, and reduces costs.
nNulls (number of null values)	Improve the efficiency of the null value check.	Scenario 1: Reduce checks for null values when a job is run.	When a job is run, null values must be checked for all types of data. If the value of the nNulls metric is 0, the checking logic can be ignored. This improves computing performance.
		Scenario 2: Filter out data based on filter conditions.	If a column has only null values, the optimizer uses the `always false` filter condition to filter out data in the whole column. This improves the efficiency of data filtering.

Column stats metric	Optimization objective	Scenario	Description
avgColLen (average column length) or maxColLen (maximum column length)	Estimate resource consumption to reduce shuffle operations.	Scenario 1: Estimate the memory of a hash-clustered table.	For example, the optimizer can estimate the memory usage of variable length fields based on the avgColLen metric to obtain the memory usage of data records. This way, the optimizer can selectively perform automatic map join operations. A broadcast join mechanism is established for the hash-clustered table to reduce shuffle operations. For a large input table, shuffle operations can be reduced to significantly improve performance.
		Scenario 2: Reduce the amount of data that needs to be shuffled.	N/A.

Column stats metric	Optimization objective	Scenario	Description
ndv (number of distinct values)	Improve the quality of an execution plan.	Scenario 1: Estimate the number of output records of a join operation.	• Data expansion: If the values of the ndv metric for the join keys of both tables are much smaller than the numbers of rows, a large number of data records are duplicates. In this case, data expansion probably occurred. The optimizer can take relevant measures to prevent the problems caused by data expansion. • Data filtering: If ndv of the small table is much smaller than that of the large table, large amounts of data in the large table are filtered out after the join operation. The optimizer can make relevant optimization decisions based on the comparison result.
		Scenario 2: Sort join operations.	The optimizer can automatically adjust the join sequence based on the estimated number of output records. For example, it can move the join operations that involve data filtering forward and the join operations that involve data expansion backward.

Column stats metric	Optimization objective	Scenario	Description
topK (top K values with the highest frequency of occurrence)	Estimate data distribution to reduce the impact of data skew on performance.	Scenario 1: Optimize join operations that involve skewed data.	If both tables of a join operation have large input and map join operations cannot be used to fully load the smaller table to memory, data skew occurs. The output of one join key is much larger than that of the other join keys. MaxCompute can automatically use map join operations to process skewed data and use merge join operations to process non-skewed data, and then merge the computing results. This feature is especially effective for join operations that involve a large amount of data. It significantly reduces the cost of manual troubleshooting.
		Scenario 2: Estimate the number of output records.	The ndv, min, and max metrics can be used to accurately estimate the number of output records only if the assumption that data is evenly distributed is true. If data is obviously skewed, the estimation based on this assumption is distorted. Therefore, special processing is required for skewed data. Other data can be estimated based on the assumption.

Use Analyze

This section uses a partitioned table and a non-partitioned table as examples to describe how to use Analyze.

- **Non-partitioned table**

You can use Analyze to collect the column stats metrics of one or more specific columns or all the columns in a non-partitioned table.

i. Run the following command on the MaxCompute client to create a non-partitioned table named analyze2_test:

```
create table if not exists analyze2_test (tinyint1 tinyint, smallint1 smallint, int1
int, bigint1 bigint, double1 double, decimal1 decimal, decimal2 decimal(20,10), strin
g1 string, varchar1 varchar(10), boolean1 boolean, timestamp1 timestamp, datetime1 da
tetime ) lifecycle 30;
```

ii. Run the following command to insert data into the table:

```
insert overwrite table analyze2_test select * from values (1Y, 20S, 4, 8L, 123452.3,
12.4, 52.5, 'str1', 'str21', false, timestamp '2018-09-17 00:00:00', datetime '2018-0
9-17 00:59:59') ,(10Y, 2S, 7, 11111118L, 67892.3, 22.4, 42.5, 'str12', 'str200', true
, timestamp '2018-09-17 00:00:00', datetime '2018-09-16 00:59:59') ,(20Y, 7S, 4, 2222
228L, 12.3, 2.4, 2.57, 'str123', 'str2', false, timestamp '2018-09-18 00:00:00', date
time '2018-09-17 00:59:59') ,(null, null, null, null, null, null, null, null, null, n
ull, null , null) as t(tinyint1, smallint1, int1, bigint1, double1, decimal1, decimal
2, string1, varchar1, boolean1, timestamp1, datetime1);
```

iii. Run the `analyze` command to collect the column stats metrics of one or more specific columns or all the columns in the table. Examples:

```
-- Collect the column stats metrics of the tinyint1 column.
analyze table analyze2_test compute statistics for columns (tinyint1);
-- Collect the column stats metrics of the smallint1, string1, boolean1, and timestam
p1 columns.
analyze table analyze2_test compute statistics for columns (smallint1, string1, boole
an1, timestamp1);
-- Collect the column stats metrics of all columns.
analyze table analyze2_test compute statistics for columns;
```

iv. Run the `show statistic` command to test the collection results. Examples:

```
-- Test the collection result of the tinyint1 column.
show statistic analyze2_test columns (tinyint1);
-- Test the collection results of the smallint1, string1, boolean1, and timestamp1 co
lumns.
show statistic analyze2_test columns (smallint1, string1, boolean1, timestamp1);
-- Test the collection results of all columns.
show statistic analyze2_test columns;
```

The following information is returned:

```
-- Collection result of the tinyint1 column:
ID = 20201126085225150gnqo****
tinyint1:MaxValue:      20                  -- The value of max.
tinyint1:DistinctNum:   4.0                 -- The value of ndv.
tinyint1:MinValue:      1                   -- The value of min.
tinyint1:NullNum:       1.0                 -- The value of nNulls.
tinyint1:TopK:  {1=1.0, 10=1.0, 20=1.0}     -- The value of topK. 10=1.0 indicates t
hat the occurrence frequency of column value 10 is 1. A maximum of 20 values with the
highest frequency of occurrence can be returned.
-- Collection results of the smallint1, string1, boolean1, and timestamp1 columns:
```

```
ID = 20201126091636149gxgf****
smallint1:MaxValue:     20
smallint1:DistinctNum:  4.0
smallint1:MinValue:     2
smallint1:NullNum:      1.0
smallint1:TopK:         {2=1.0, 7=1.0, 20=1.0}
string1:MaxLength       6.0                 -- The value of maxColLen.
string1:AvgLength:      3.0                 -- The value of avgColLen.
string1:DistinctNum:    4.0
string1:NullNum:        1.0
string1:TopK:   {str1=1.0, str12=1.0, str123=1.0}
boolean1:DistinctNum:   3.0
boolean1:NullNum:       1.0
boolean1:TopK:  {false=2.0, true=1.0}
timestamp1:DistinctNum:         3.0
timestamp1:NullNum:     1.0
timestamp1:TopK:        {2018-09-17 00:00:00.0=2.0, 2018-09-18 00:00:00.0=1.0}
-- Collection results of all columns:
ID = 20201126092022636gzm1****
tinyint1:MaxValue:      20
tinyint1:DistinctNum:   4.0
tinyint1:MinValue:      1
tinyint1:NullNum:       1.0
tinyint1:TopK:  {1=1.0, 10=1.0, 20=1.0}
smallint1:MaxValue:     20
smallint1:DistinctNum:  4.0
smallint1:MinValue:     2
smallint1:NullNum:      1.0
smallint1:TopK:         {2=1.0, 7=1.0, 20=1.0}
int1:MaxValue:  7
int1:DistinctNum:       3.0
int1:MinValue:  4
int1:NullNum:   1.0
int1:TopK:      {4=2.0, 7=1.0}
bigint1:MaxValue:       11111118
bigint1:DistinctNum:    4.0
bigint1:MinValue:       8
bigint1:NullNum:        1.0
bigint1:TopK:   {8=1.0, 2222228=1.0, 11111118=1.0}
double1:MaxValue:       123452.3
double1:DistinctNum:    4.0
double1:MinValue:       12.3
double1:NullNum:        1.0
double1:TopK:   {12.3=1.0, 67892.3=1.0, 123452.3=1.0}
decimal1:MaxValue:      22.4
decimal1:DistinctNum:   4.0
decimal1:MinValue:      2.4
decimal1:NullNum:       1.0
decimal1:TopK:  {2.4=1.0, 12.4=1.0, 22.4=1.0}
decimal2:MaxValue:      52.5
decimal2:DistinctNum:   4.0
decimal2:MinValue:      2.57
decimal2:NullNum:       1.0
decimal2:TopK:  {2.57=1.0, 42.5=1.0, 52.5=1.0}
string1:MaxLength       6.0
```

```
string1:MaxLength       6.0
string1:AvgLength:      3.0
string1:DistinctNum:    4.0
string1:NullNum:        1.0
string1:TopK:   {str1=1.0, str12=1.0, str123=1.0}
varchar1:MaxLength      6.0
varchar1:AvgLength:     3.0
varchar1:DistinctNum:   4.0
varchar1:NullNum:       1.0
varchar1:TopK:  {str2=1.0, str200=1.0, str21=1.0}
boolean1:DistinctNum:   3.0
boolean1:NullNum:       1.0
boolean1:TopK:  {false=2.0, true=1.0}
timestamp1:DistinctNum:         3.0
timestamp1:NullNum:     1.0
timestamp1:TopK:        {2018-09-17 00:00:00.0=2.0, 2018-09-18 00:00:00.0=1.0}
datetime1:DistinctNum:  3.0
datetime1:NullNum:      1.0
datetime1:TopK:         {1537117199000=2.0, 1537030799000=1.0}
```

- **Partitioned table**

  You can use Analyze to collect the column stats metrics of a specific partition in a partitioned table.

  i. Run the following command on the MaxCompute client to create a partitioned table named srcpart:

  ```
  create table if not exists srcpart_test (key string, value string) partitioned by (ds string, hr string) lifecycle 30;
  ```

  ii. Run the following commands to insert data into the table:

  ```
  insert into table srcpart_test partition(ds='20201220', hr='11') values ('123', 'val_123'), ('76', 'val_76'), ('447', 'val_447'), ('1234', 'val_1234');
  insert into table srcpart_test partition(ds='20201220', hr='12') values ('3', 'val_3'), ('12331', 'val_12331'), ('42', 'val_42'), ('12', 'val_12');
  insert into table srcpart_test partition(ds='20201221', hr='11') values ('543', 'val_543'), ('2', 'val_2'), ('4', 'val_4'), ('9', 'val_9');
  insert into table srcpart_test partition(ds='20201221', hr='12') values ('23', 'val_23'), ('56', 'val_56'), ('4111', 'val_4111'), ('12333', 'val_12333');
  ```

  iii. Run the `analyze` command to collect the column stats metrics of a specific partition in the table. Example:

  ```
  analyze table srcpart_test partition(ds='20201221') compute statistics for columns (key , value);
  ```

  iv. Run the `show statistic` command to test the collection results. Example:

  ```
  show statistic srcpart_test partition (ds='20201221') columns (key , value);
  ```

  The following information is returned:

```
ID = 20210105121800689g28p****
(ds=20201221,hr=11) key:MaxLength       3.0
(ds=20201221,hr=11) key:AvgLength:      1.0
(ds=20201221,hr=11) key:DistinctNum:    4.0
(ds=20201221,hr=11) key:NullNum:        0.0
(ds=20201221,hr=11) key:TopK:   {2=1.0, 4=1.0, 543=1.0, 9=1.0}
(ds=20201221,hr=11) value:MaxLength     7.0
(ds=20201221,hr=11) value:AvgLength:    5.0
(ds=20201221,hr=11) value:DistinctNum:  4.0
(ds=20201221,hr=11) value:NullNum:      0.0
(ds=20201221,hr=11) value:TopK:         {val_2=1.0, val_4=1.0, val_543=1.0, val_9=1.0
}
(ds=20201221,hr=12) key:MaxLength       5.0
(ds=20201221,hr=12) key:AvgLength:      3.0
(ds=20201221,hr=12) key:DistinctNum:    4.0
(ds=20201221,hr=12) key:NullNum:        0.0
(ds=20201221,hr=12) key:TopK:   {12333=1.0, 23=1.0, 4111=1.0, 56=1.0}
(ds=20201221,hr=12) value:MaxLength     9.0
(ds=20201221,hr=12) value:AvgLength:    7.0
(ds=20201221,hr=12) value:DistinctNum:  4.0
(ds=20201221,hr=12) value:NullNum:      0.0
(ds=20201221,hr=12) value:TopK:         {val_12333=1.0, val_23=1.0, val_4111=1.0, val
_56=1.0}
```

Use Freeride

To use Freeride, you must run the following commands at the session level at the same time to configure properties:

- `set odps.optimizer.stat.collect.auto=true;`: Enable Freeride to automatically collect the column stats metrics of tables.
- `set odps.optimizer.stat.collect.plan=xx;`: Configure a collection plan to collect specific column stats metrics of specific columns.

```
-- Collect the avgColLen metric of the key column in the target_table table.
set odps.optimizer.stat.collect.plan={"target_table":"{\"key\":\"AVG_COL_LEN\"}"}
-- Collect the min and max metrics of the s_binary column in the target_table table, and
the topK and nNulls metrics of the s_int column in the table.
set odps.optimizer.stat.collect.plan={"target_table":"{\"s_binary\":\"MIN,MAX\",\"s_int\"
:\"TOPK,NULLS\"}"};
```

> **Note** If no data is collected after the commands are run, Freeride may fail to be enabled. If this occurs, you must check whether the `odps.optimizer.stat.collect.auto` property exists on the **Json Summary** tab of the Logview UI. If this property is not found, the current server version does not support Freeride. The server will be upgraded to a version that supports Freeride in the future.

Mappings between column stats metrics and parameters in the `set odps.optimizer.stat.collect.plan=xx;` command:

- min: MIN
- max: MAX

- nNulls: NULLS
- avgColLen: AVG_COL_LEN
- maxColLen: MAX_COL_LEN
- ndv: NDV
- topK: TOPK

MaxCompute allows you to use Freeride to collect column stats metrics when you run the `CREATE TABLE`, `INSERT INTO`, or `INSERT OVERWRITE` command.

Before you can use Freeride, you must first prepare a source table. For example, run the following commands to create a source table named src_test and insert data into the table:

```
create table if not exists src_test (key string, value string);
insert overwrite table src_test values ('100', 'val_100'), ('100', 'val_50'), ('200', 'val_
200'), ('200', 'val_300');
```

- `CREATE TABLE`: Use Freeride to collect column stats metrics while you create a destination table named target. Example:

  ```
  -- Create a destination table.
  set odps.optimizer.stat.collect.auto=true;
  set odps.optimizer.stat.collect.plan={"target_test":"{\"key\":\"AVG_COL_LEN,NULLS\"}"};
  create table target_test as select key, value from src_test;
  -- Test the collection results.
  show statistic target_test columns;
  ```

  The following information is returned:

  ```
  key:AvgLength:  3.0
  key:NullNum:    0.0
  ```

- `INSERT INTO`: Use Freeride to collect column stats metrics while you run the `insert into` command to append data to a table. Example:

  ```
  -- Create a destination table.
  create table freeride_insert_into_table like src_test;
  -- Append data to the table.
  set odps.optimizer.stat.collect.auto=true;
  set odps.optimizer.stat.collect.plan={"freeride_insert_into_table":"{\"key\":\"AVG_COL_LE
  N,NULLS\"}"};
  insert into table freeride_insert_into_table select key, value from src order by key, val
  ue limit 10;
  -- Test the collection results.
  show statistic freeride_insert_into_table columns;
  ```

- `INSERT OVERWRITE`: Use Freeride to collect column stats metrics while you run the `insert overwrite` command to overwrite data in a table. Example:

```
-- Create a destination table.
create table freeride_insert_overwrite_table like src_test;
-- Overwrite data in the table.
set odps.optimizer.stat.collect.auto=true;
set odps.optimizer.stat.collect.plan={"freeride_insert_overwrite_table":"{\"key\":\"AVG_C
OL_LEN,NULLS\"}"};
insert overwrite table freeride_insert_overwrite_table select key, value from src_test or
der by key, value limit 10;
-- Test the collection results.
show statistic freeride_insert_overwrite_table columns;
```

8.Open source features of MaxCompute

MaxCompute is a fast and fully managed data warehouse that can process terabytes, petabytes, or even exabytes of data. This topic describes the open source features of MaxCompute.

SDK

MaxCompute provides SDK for Java and SDK for Python to create, view, and delete MaxCompute tables. You can edit code in SDKs to manage MaxCompute.

- Java SDK

  For more information about how to use SDK for Java, see SDK for Java.

  How to obtain service support: See the official documentation or submit a ticket.

- Python SDK

  PyODPS is the SDK for Python of MaxCompute. PyODPS provides the DataFrame framework and allows you to perform basic operations on MaxCompute objects. This helps you analyze data in MaxCompute. For more information, see aliyun-odps-python-sdk on GitHub and PyODPS documentation, which describes all related interfaces and classes in detail.

  - You are welcome to help build the PyODPS ecosystem. Install PyODPS before you use it. For more information, see Installation guide and limits.
  - For more information about how to use PyODPS in DataWorks, see Use PyODPS in DataWorks. PyODPS provides the DataFrame API. For more information, see PyODPS DataFrame overview.
  - Feel free to share your feedback and suggestions to help PyODPS evolve. For more information, see aliyun-odps-python-sdk on GitHub.

  How to obtain service support: See the official documentation or submit a ticket.

MaxCompute RODPS

RODPS is a plug-in that MaxCompute provides for R. For more information, see ODPS Plugin for R on GitHub.

How to obtain service support: Leave a message or create an issue in ODPS Plugin for R on GitHub.

MaxCompute JDBC is an official JDBC driver provided by MaxCompute. It provides a set of interfaces to run SQL tasks for Java programs. The project is hosted in ODPS JDBC on GitHub.

How to obtain service support: Leave a message or create an issue in ODPS JDBC on GitHub.

Mars

Mars is a tensor-based unified distributed computing framework. Mars makes it possible to run large-scale scientific computing tasks by using only several lines of code, whereas MapReduce requires hundreds of lines of code. In addition, Mars improves computing performance.

The source code of Mars is now available on GitHub. You are welcome to contribute to Mars. You can visit Mars on GitHub to obtain its open source code.

For more information about Mars, see Mars Documentation.

How to obtain service support: Leave a message or create an issue in Mars on GitHub.

Data collector

MaxCompute provides a set of open source data collectors.

MaxCompute provides data collectors for the following services:

- Flume
- Oracle GoldenGate (OGG)
- Sqoop
- Kettle
- Hive Data Transfer UDTF

  The Flume and OGG data collectors are implemented based on the DataHub SDK, whereas the data collectors for Sqoop, Kettle, and Hive Data Transfer UDTF are implemented based on the Tunnel SDK. DataHub is a real-time data transfer channel. Tunnel is a batch data transfer channel. The Flume and OGG data collectors are used to transfer data in real time. The data collectors for Sqoop, Kettle, and Hive Data Transfer UDTF are used to transfer data in batches in offline mode.

For more information about the source code, see Aliyun MaxCompute Data Collectors on GitHub. For more information about data collectors, see wiki.

How to obtain service support: Leave a message or create an issue in Aliyun MaxCompute Data Collectors on GitHub.