ALIBABA CLOUD

阿里云

云原生数据仓库AnalyticDB MySQL版

系统函数

文档版本：20201218

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.控制流函数

本文介绍AnalyticDB for MySQL中的控制流函数。

- CASE
- IF
- IFNULL
- NULLIF

本文中的控制流函数以 `conditiontest` 表为测试数据。

```
create table conditiontest(a int) distributed by hash(a);
```

```
insert into conditiontest values (1),(2),(3);
```

```
SELECT * FROM conditiontest;
+---+
| a |
+---+
| 2 |
| 1 |
| 3 |
```

CASE

```
CASE expression
  WHEN value THEN result
  [ WHEN ... ]
  [ ELSE result ]
  END
```

- 命令说明：简单 `CASE` 表达式会从左到右依次查找 `value` ，直到找到和 `expression` 相等的 `value` ，并返回对应的 `result` 结果；如果没有找到相等的 `value` ，则返回 `ELSE` 语句后的 `result` 结果。
- 示例：

```
SELECT a,
    CASE a
    WHEN 1 THEN 'one'
    WHEN 2 THEN 'two'
    ELSE 'three'
    END as caseresult
FROM conditiontest;
+---+------------+
| a | caseresult |
+---+------------+
| 2 | two      |
| 1 | one      |
| 3 | three    |
```

```
CASE
  WHEN condition THEN result
  [ WHEN ... ]
  [ ELSE result ]
  END
```

- 命令说明：高级 `CASE` 表达式会从左到右依次计算 `condition` ，直到第一个为 `TRUE` 的 `condition` ，并返回对应的 `result` 结果；如果没有找到为 `TRUE` 的 `condition` ，则返回 `ELSE` 语句后的 `result` 结果。
- 示例：

```
SELECT a,
    CASE a
    WHEN a=1 THEN 'one1'
    WHEN a=2 THEN 'two2'
    ELSE 'three3'
    END as caseresult
FROM conditiontest;
+---+------------+
| a | caseresult |
+---+------------+
| 1 | one1     |
| 3 | three3   |
| 2 | three3   |
```

IF

```
if(condition, true_value)
```

- 命令说明：如果 condition 为 true ，结果返回 true_value ；否则返回 null 。
- 示例：

```
SELECT IF((2+3)>4,5);
+-------+
| _col0 |
+-------+
|  5 |
```

```
if(condition, true_value, false_value)
```

- 命令说明：如果 condition 为 true ，结果返回 true_value ；否则结果返回 false_value 。
- 示例：

```
SELECT IF((2+3)<5,5,6);
+-------+
| _col0 |
+-------+
|  6 |
```

IFNULL

```
IFNULL(expr1,expr2)
```

- 命令说明：如果 expr1 结果不为空，则返回 expr1 的值；否则返回 expr2 的值。
- 示例：

```
SELECT IFNULL(NULL,2);
+-------+
| _col0 |
+-------+
|  2 |
SELECT IFNULL(1,0);
+-------+
| _col0 |
+-------+
|  1 |
```

NULLIF

```
NULLIF(expr1,expr2)
```

- 命令说明：如果 `expr1` 与 `expr2` 值相等，结果返回 `null` ；否则结果返回 `expr1` 的值。
- 示例：

```
SELECT NULLIF (2,1);
+-------+
| _col0 |
+-------+
|   2 |
SELECT NULLIF (2,2);
+-------+
| _col0 |
+-------+
| NULL |
```

2.数值函数和运算符

2.1. 算术运算符

AnalyticDB for MySQL支持以下算术运算符。

+	加
-	减
*	乘
/	除法
DIV	除法（整数视角）
%或MOD	取模
-	改变参数符号

+

- 命令说明：加法。
- 示例：

```
select 3+5;
+-------+
| _col0 |
+-------+
|     8 |
```

```
select 3+2.9875;
+--------+
| _col0  |
+--------+
| 5.9875 |
```

-

- 命令说明：减法。
- 示例：

```
select 3-5;
+-------+
| _col0 |
+-------+
|   -2 |
```

```
select 3-1.5;
+-----------+
| _col0    |
+-----------+
| 1.5     |
```

*

- 命令说明：乘法。
- 示例：

```
select 3*pi();
+------------------+
| _col0         |
+------------------+
| 9.42477796076938 |
```

/

- 命令说明：除法。
- 示例：

```
select 3/pi();
+-------------------+
| _col0         |
+-------------------+
| 0.954929658551372 |
```

DIV

- 命令说明：除法，从除法结果中舍弃小数点右侧的小数部分。
- 示例：

```
select 3 div pi();
+-------+
| _col0 |
+-------+
|    0 |
```

```
select 33 div 2;
+-------+
| _col0 |
+-------+
|   16 |
```

%或MOD

- 命令说明：返回两个参数除法后的余数。
- 示例：

```
select 3 mod pi();
+-------+
| _col0 |
+-------+
|  3.0 |
```

```
select 33 % 2;
+-------+
| _col0 |
+-------+
|    1 |
```

-

- 命令说明：将正数变为负数或者将负数变为正数。
- 示例：

```
select - 2;
+-------+
| _col0 |
+-------+
|   -2 |
```

```
select - 2;
+-------+
| _col0 |
+-------+
|   -2 |
```

2.2. 数值函数

AnalyticDB for MySQL支持以下数值函数。

- ABS：返回参数的绝对值
- ROUND：返回参数四舍五入后的值
- SQRT：返回参数的平方根
- LN：返回参数的自然对数
- LOG：对数函数
- LOG2：返回以2为底的对数
- LOG10：返回以10为底的对数
- PI：返回圆周率
- POWER/POW：返回x的y次幂
- RADIANS：角度转换为弧度
- DEGREES：弧度转换为度
- SIGN：返回参数的符号的值
- CEILING/CEIL：返回大于参数的最小整数值。
- FLOOR：返回小于参数的最大整数值
- EXP：返回以e为底、x为幂的值
- COS：返回参数的余弦值
- ACOS：返回参数的反余弦值
- TAN：返回参数的正切值
- ATAN：返回参数的反正切值
- ATAN2：返回参数x除以参数y之后的反正切值
- COT：返回参数的余切值
- ASIN：返回参数的反正弦值
- SIN：返回参数的正弦值
- CRC32：返回参数的的循环冗余码

ABS

```
abs(tinyint x)
abs(smallint x)
abs(int x)
abs(bigint x)
abs(float x)
abs(double x)
abs(decimal x)
```

- 命令说明：返回 `x` 的绝对值。
- 返回值类型：LONG、DECIMAL或DOUBLE。
- 示例：

```
select abs(4.5);
+----------+
| abs(4.5) |
+----------+
|    4.5 |
```

```
select abs(4);
+--------+
| abs(4) |
+--------+
|    4 |
```

ROUND

```
round(tinyint x)
round(smallint x)
round(int x)
round(bigint x)
round(float x)
round(double x)
round(x, d)
```

- 命令说明：将 `x` 四舍五入， `d` 是要保留的小数位数，默认 `d` 为 `0` ，舍入算法取决于 `x` 的数据类型。
  - 如果 `x` 为 `null` ，返回结果为 `null` 。
  - 如果 `d>0` ，则四舍五入到指定的小数位。
  - 如果 `d=0` ，则四舍五入到最接近的整数。
  - 如果 `d<0` ，则在小数点左侧进行四舍五入。
- 返回值类型：LONG、DECIMAL或DOUBLE。
- 示例：

```
select round(4);
+----------+
| round(4) |
+----------+
|     4 |
```

```
select round(4.5);
+------------+
| round(4.5) |
+------------+
|        5.0 |
```

```
select round(345.984, -1);
+-----------------------------+
| round(345.984, INTEGER '-1') |
+-----------------------------+
|                      350.0 |
```

SQRT

```
sqrt(double x)
```

- 命令说明：返回 `x` 的平方根。
- 返回值类型：DOUBLE。
- 示例：

```
select sqrt(4.222);
+-------------------+
| sqrt(4.222)       |
+-------------------+
| 2.054750593137766 |
```

LN

```
ln(double x)
```

- 命令说明：返回 `x` 的自然对数。
- 返回值类型：DOUBLE。
- 示例：

```
select ln(2.718281828459045);
+-----------------------+
| ln(2.718281828459045) |
+-----------------------+
|                   1.0 |
```

LOG

```
log(double x)
log(double x, double y)
```

- 命令说明：调用一个参数时，返回 x 的自然对数。调用两个参数时，返回以 x 为底的 y 的对数。
- 返回值类型：DOUBLE。
- 示例：

```
select log(16);
+------------------+
| log(16)          |
+------------------+
| 2.772588722239781 |
```

```
select log(10,100);
+--------------+
| log(10, 100) |
+--------------+
|          2.0 |
```

LOG2

```
log2(double x)
```

- 命令说明：返回以 2 为底的对数。
- 返回值类型：DOUBLE。
- 示例：

```
select log2(8.7654);
+------------------+
| log2(8.7654)     |
+------------------+
| 3.131819928389146 |
```

LOG10

```
log10(double x)
```

- 命令说明：返回以10位底的对数。
- 返回值类型：DOUBLE。
- 示例：

```
select log10(100.876);
+-------------------+
| log10(100.876)    |
+-------------------+
| 2.0037878529824615 |
```

PI

```
pi()
```

- 命令说明：返回圆周率。
- 返回值类型：DOUBLE。
- 示例：

```
select pi();
+-------------------+
| pi()              |
+-------------------+
| 3.141592653589793 |
```

POWER/POW

```
power(double x, double y)
pow(double x, double y)
```

- 命令说明：返回 `x` 的 `y` 次幂。
- 返回值类型：DOUBLE。
- 示例：

```
select power(1.2,3.4);
+-------------------+
| power(1.2, 3.4)   |
+-------------------+
| 1.858729691979481 |
```

```
select pow(1.2,3.4);
+-------------------+
| pow(1.2, 3.4)     |
+-------------------+
| 1.858729691979481 |
```

RADIANS

```
radians(double x)
```

- 命令说明：角度转换为弧度。
- 返回值类型：DOUBLE。
- 示例：

```
select radians(60.0);
+--------------------+
| radians(60.0)      |
+--------------------+
| 1.0471975511965976 |
```

DEGREES

```
degrees(double x)
```

- 命令说明：弧度转换为度。
- 返回值类型：DOUBLE。
- 示例：

```
select degrees(1.3);
+-------------------+
| degrees(1.3)      |
+-------------------+
| 74.48451336700703 |
```

SIGN

```
sign(smallint x)
sign(tinyint x)
sign(int x)
sign(bigint x)
sign(float x)
sign(double x)
sign(decimal x)
```

- 命令说明：返回数字 `x` 的符号的值。
- 返回值类型：LONG。
- 示例：

```
select sign(12);
+----------+
| sign(12) |
+----------+
|      1 |
```

```
select sign(4.5);
+-----------+
| sign(4.5) |
+-----------+
|       1 |
```

CEILING/CEIL

```
ceiling(tinyint x)
ceiling(smallint x)
ceiling(int x)
ceiling(bigint x)
ceiling(float x)
ceiling(double x)
```

- 命令说明：返回大于 `x` 的最小整数值。
- 返回值类型：LONG、DECIMAL或DOUBLE。
- 示例：

```
select ceiling(4);
+------------+
| ceiling(4) |
+------------+
|       4 |
```

```
select ceiling(2.3);
+--------------+
| ceiling(2.3) |
+--------------+
|       3.0 |
```

FLOOR

```
floor(tinyint x)
floor(smallint x)
floor(int x)
floor(bigint x)
floor(float x)
floor(double x)
```

- 命令说明：返回小于 `x` 的最大整数值。
- 返回值类型：LONG、DECIMAL或DOUBLE。
- 示例：

```
select floor(4.5);
+------------+
| floor(4.5) |
+------------+
|        4.0 |
```

```
select floor(7);
+----------+
| floor(7) |
+----------+
|        7 |
```

EXP

```
exp(double x)
```

- 命令说明：返回以 `e` 为底、 `x` 为幂的值。
- 返回值类型：DOUBLE。
- 示例：

```
select exp(4.5);
+--------------------+
| exp(4.5)           |
+--------------------+
| 90.01713130052181 |
```

COS

```
cos(double x)
```

- 命令说明：返回 x 的余弦值。
- 返回值类型：DOUBLE。
- 示例：

```
select cos(1.3);
+--------------------+
| cos(1.3)           |
+--------------------+
| 0.26749882862458735 |
```

ACOS

```
acos(double x)
```

- 命令说明：返回 x 的反余弦值。

  如果 x>1 或者 x<-1 ，返回结果为 null 。
- 返回值类型：DOUBLE。
- 示例：

```
select acos(0.5);
+--------------------+
| acos(0.5)          |
+--------------------+
| 1.0471975511965979 |
```

TAN

```
tan(double x)
```

- 命令说明：返回 x 的正切值。
- 返回值类型：DOUBLE。
- 示例：

```
select tan(8);
+--------------------+
| tan(8)             |
+--------------------+
| -6.799711455220379 |
```

ATAN

```
atan(double x)
```

- 命令说明：返回 x 的反正切值。
- 返回值类型：DOUBLE。
- 示例：

```
select atan(0.5);
+--------------------+
| atan(0.5)          |
+--------------------+
| 0.4636476090008061 |
```

ATAN2

```
atan2(double x, double y)
atan(double x, double y)
```

- 命令说明：返回参数 x 除以参数 y 之后的反正切值。
- 返回值类型：DOUBLE。
- 示例：

```
select atan2(0.5,0.3);
+--------------------+
| atan2(0.5, 0.3)    |
+--------------------+
| 1.0303768265243125 |
```

```
select atan(0.5,0.3);
+--------------------+
| atan(0.5, 0.3)     |
+--------------------+
| 1.0303768265243125 |
```

COT

```
cot(double x)
```

- 命令说明：返回 x 的余切值。
- 返回值类型：DOUBLE。
- 示例：

```
select cot(1.234);
+--------------------+
| cot(1.234)         |
+--------------------+
| 0.35013639786701445 |
```

ASIN

```
asin(double x)
```

- 命令说明：返回 `x` 的反正弦值。
- 返回值类型：DOUBLE。
- 示例：

```
select asin(0.5);
+-------------------+
| asin(0.5)         |
+-------------------+
| 0.5235987755982989 |
```

SIN

```
sin(double x)
```

- 命令说明：返回 `x` 的正弦值。
- 返回值类型：DOUBLE。
- 示例：

```
select sin(1.234);
+-------------------+
| sin(1.234)        |
+-------------------+
| 0.9438182093746337 |
```

CRC32

```
crc32(varbinary x)
```

- 命令说明：返回参数 `x` 的循环冗余码。
- 返回值类型：LONG
- 示例：

```
select crc32(CAST('中国' AS VARBINARY));
+-----------------------------------+
| crc32(CAST('中国' AS varbinary)) |
+-----------------------------------+
|                 737014929 |
```

3.日期和时间函数

本文介绍AnalyticDB for MySQL中的日期和时间函数。

- ADDDATE：返回添加指定时间后的日期。
- ADDTIME：返回添加指定时间后的时间。
- CONVERT_TZ：转换时区，从from_tz转到to_tz给出的时区，并返回结果值。
- CURDATE：返回当前日期。
- CURTIME：返回当前时间。
- DATE：返回日期或日期时间表达式中的日期。
- DATE_FORMAT：按照Format指定的格式，将日期时间格式化成字符串。
- SUBDATE/DATE_SUB：返回Date减去指定INTERVAL间隔后的日期。
- DATEDIFF：返回Expr1减去Expr2后的天数。
- DAY/DAYOFMONTH：返回Date中的日，取值范围1~31。
- DAYNAME：返回日期对应的工作日的名称，例如星期一为Monday。
- DAYOFWEEK：返回日期对应的工作日索引值。
- DAYOFYEAR：返回指定日期是当年的哪一天。
- EXTRACT：返回日期或时间的单独部分，例如年、月、日、小时、分钟等。
- FROM_DAYS：根据指定的天数N，返回对应的DATE值。
- FROM_UNIXTIME：返回Unixtime时间戳。
- HOUR：返回时间中的小时。
- LAST_DAY：返回日期或者日期时间中对应月份的最后一天。
- LOCALTIME/LOCALTIMESTAMP/NOW：返回当前时间戳。
- MAKEDATE：按照参数Year和DayOfYear，返回一个日期。
- MAKETIME：按照参数Hour、Minute和Second，返回一个时间。
- MINUTE：返回时间中的分钟。
- MONTH：返回日期中的月份。
- MONTHNAME：返回日期中月份的全名。
- PERIOD_ADD：将日期格式的参数P增加N个月。
- PERIOD_DIFF：返回P1和P2之间相差的月数。
- QUARTER：返回日期在一年中的季度。
- SEC_TO_TIME：将Seconds转换为时间。
- SECOND：返回时间中的秒。
- STR_TO_DATE：按照指定日期或时间显示格式，将字符串转换为日期或日期时间类型。
- SUBTIME：返回Expr1减去Expr2后的时间。
- SYSDATE：获取系统时间。
- TIME：以字符串形式返回Expr中的时间。
- TIME_FORMAT：按照Format指定的格式，以字符串形式显示时间。
- TIME_TO_SEC：返回Time转换为秒的结果。
- TIMEDIFF：返回Expr1减去Expr2后的时间。

- TIMESTAMP：返回Expr表示的日期或日期时间。
- TIMESTAMPADD：将Interval添加到日期或日期时间表达式datetime_expr中。
- TIMESTAMPDIFF：返回日期或日期时间表达式datetime_expr1减去datetime_expr2后的结果。
- TO_DAYS：根据给定日期Date，返回自0年开始的天数。
- TO_SECONDS：根据给定的Expr，返回自0年开始的秒数。
- UNIX_TIMESTAMP：返回自1970-01-01 00:00:00 UTC以来秒数的Unix时间戳。
- UTC_DATE：返回UTC日期。
- UTC_TIME：返回UTC时间。
- UTC_TIMESTAMP：返回UTC时间戳。
- WEEK：返回日期对应的周数。
- WEEKDAY：返回日期对应的工作日。
- WEEKOFYEAR：返回日期对应的日历周。
- YEAR：返回日期中的年份。
- YEARWEEK：返回日期的年份和星期。

ADDDATE

```
ADDDATE(date,INTERVAL expr unit)
ADDDATE(expr,days
```

- 参数类型：

```
adddate(date, INTERVAL expr unit)
adddate(timestamp, INTERVAL expr unit)
adddate(datetime, INTERVAL expr unit)
adddate(varchar, INTERVAL expr unit)
adddate(date, varchar)
adddate(date, bigint)
adddate(datetime, bigint)
adddate(datetime, varchar)
adddate(timestamp, varchar)
adddate(timestamp, bigint)
adddate(varchar, bigint)
adddate(varchar, varchar)
```

- 返回值类型：DATE。
- 命令说明：返回添加指定时间后的日期。
  - `unit` 可取值为：`second` 、`minute` 、`hour` 、`day` 、`month` 、`year` 、`minute_second` 、`hour_second` 、`hour_minute` 、`day_second` 、`day_minute` 、`day_hour` 、`year_month` 。`unit` 默认值为 `day` 。
  - `days` 、`expr` ：系统将返回 `expr` 加上 `days` 之后的结果。

- 示例：

```
select adddate(date '2001-1-22',interval '3' day);
+--------------------------------------------+
| adddate(DATE '2001-1-22', INTERVAL  '3' DAY) |
+--------------------------------------------+
| 2001-01-25                                 |
```

```
select adddate(timestamp '2001-1-22',interval '3' day);
+-------------------------------------------------+
| adddate(TIMESTAMP '2001-1-22', INTERVAL  '3' DAY) |
+-------------------------------------------------+
| 2001-01-25 00:00:00                             |
```

```
select adddate(datetime '2001-1-22',interval '3' day);
+-------------------------------------------------+
| adddate(DATETIME '2001-1-22', INTERVAL  '3' DAY) |
+-------------------------------------------------+
| 2001-01-25 00:00:00                             |
```

```
select adddate('2001-1-22',interval '3' day);
+---------------------------------------+
| adddate('2001-1-22', INTERVAL  '3' DAY) |
+---------------------------------------+
| 2001-01-25
```

```
select adddate(datetime '2001-1-22',interval '3' second);
+--------------------------------------------------+
| adddate(DATETIME '2001-1-22', INTERVAL  '3' SECOND) |
+--------------------------------------------------+
|                      2001-01-22 00:00:03 |
```

```
select adddate(datetime '2001-1-22',interval '3' minute);
+--------------------------------------------------+
| adddate(DATETIME '2001-1-22', INTERVAL  '3' MINUTE) |
+--------------------------------------------------+
|                      2001-01-22 00:03:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' hour);
+-------------------------------------------------+
| adddate(DATETIME '2001-1-22', INTERVAL  '3' HOUR) |
+-------------------------------------------------+
|                     2001-01-22 03:00:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' month);
+-------------------------------------------------+
| adddate(DATETIME '2001-1-22', INTERVAL  '3' MONTH) |
+-------------------------------------------------+
|                      2001-04-22 00:00:00 |
select adddate(datetime '2001-1-22',interval '3' year);
+-------------------------------------------------+
| adddate(DATETIME '2001-1-22', INTERVAL  '3' YEAR) |
+-------------------------------------------------+
|                     2004-01-22 00:00:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' hour_second) as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-22 03:00:00 |
select adddate(datetime '2001-1-22',interval '3' hour_minute) as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-22 03:00:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' day_second) as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-25 00:00:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' minute_second) as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-22 00:03:00 |
```

```
adddate(datetime '2001-1-22',interval '3' day_minute) as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-25 00:00:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' day_hour) as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-25 00:00:00 |
```

```
select adddate(datetime '2001-1-22 12:32:1',interval '4' year_month) as result;
+---------------------+
| result              |
+---------------------+
| 2005-01-22 12:32:01 |
```

```
select adddate('2001-1-22','3');
+--------------------------+
| adddate('2001-1-22', '3') |
+--------------------------+
| 2001-01-25               |
```

```
select adddate('2001-1-22',3);
+-------------------------+
| adddate('2001-1-22', 3) |
+-------------------------+
| 2001-01-25              |
```

```
select adddate(datetime '2001-1-22 12:12:32',3);
+-----------------------------------------+
| adddate(DATETIME '2001-1-22 12:12:32', 3) |
+-----------------------------------------+
|                  2001-01-25 12:12:32 |
```

```
select adddate(datetime '2001-1-22 12:12:32','3');
+-------------------------------------------+
| adddate(DATETIME '2001-1-22 12:12:32', '3') |
+-------------------------------------------+
|                  2001-01-25 12:12:32 |
```

```
select adddate(timestamp '2001-1-22 12:12:32','3');
+-------------------------------------------+
| adddate(TIMESTAMP '2001-1-22 12:12:32', '3') |
+-------------------------------------------+
|                  2001-01-25 12:12:32 |
```

```
select adddate(timestamp '2001-1-22 12:12:32',3);
+-------------------------------------------+
| adddate(TIMESTAMP '2001-1-22 12:12:32', 3) |
+-------------------------------------------+
|                  2001-01-25 12:12:32 |
```

```
select adddate('2001-1-22 12:12:32',3);
+---------------------------------+
| adddate('2001-1-22 12:12:32', 3) |
+---------------------------------+
| 2001-01-25 12:12:32          |
```

```
select adddate('2001-1-22 12:12:32','3');
+-----------------------------------+
| adddate('2001-1-22 12:12:32', '3') |
+-----------------------------------+
| 2001-01-25 12:12:32           |
```

ADDTIME

```
ADDTIME(expr1,expr2)
```

- 命令说明：返回添加指定时间后的时间，即返回 `expr1` 增加 `expr2` 后的结果。
- 参数类型：

```
addtime(date,varchar)
addtime(time,varchar)
addtime(datetime,varchar)
addtime(timestamp,varchar)
addtime(varchar,varchar)
```

- 返回值类型：VARCHAR。
- 示例：

```
select addtime(date '1998-01-01','01:01:01');
+----------------------------------------+
| addtime(DATE '1998-01-01', '01:01:01') |
+----------------------------------------+
| 1998-01-01 01:01:01                    |
```

```
select addtime(time '00:00:00','01:01:01');
+----------------------------------------+
| addtime(TIME '00:00:00', '01:01:01') |
+----------------------------------------+
| 01:01:01                               |
```

```
select addtime(datetime '2001-1-22 00:00:00','01:01:01');
+--------------------------------------------------+
| addtime(DATETIME '2001-1-22 00:00:00', '01:01:01') |
+--------------------------------------------------+
| 2001-01-22 01:01:01                              |
```

```
select addtime(timestamp '2001-1-22 00:00:00','01:01:01');
+--------------------------------------------------+
| addtime(TIMESTAMP '2001-1-22 00:00:00', '01:01:01') |
+--------------------------------------------------+
| 2001-01-22 01:01:01                              |
```

```
select addtime('2001-1-22 00:00:00','01:01:01');
+-----------------------------------------+
| addtime('2001-1-22 00:00:00', '01:01:01') |
+-----------------------------------------+
| 2001-01-22 01:01:01                     |
```

CONVERT_TZ

```
CONVERT_TZ(dt,from_tz,to_tz)
```

- 命令说明：转换 `dt` ，从 `from_tz` 转到 `to_tz` 给出的时区，并返回结果。
- 参数类型：

```
convert_tz(varchar, varchar, varchar)
```

- 返回值类型：DATETIME。
- 示例：

```
select convert_tz('2004-01-01 12:00:00','+00:00','+10:00');
+-----------------------------------------------------+
| convert_tz('2004-01-01 12:00:00', '+00:00', '+10:00') |
+-----------------------------------------------------+
|                      2004-01-01 22:00:00 |
```

```
select convert_tz('2004-01-01 12:00:00','GMT','MET');
+-------------------------------------------------+
| convert_tz('2004-01-01 12:00:00', 'GMT', 'MET') |
+-------------------------------------------------+
|                   2004-01-01 13:00:00 |
```

CURDATE

```
CURDATE()
```

- 命令说明：返回当前日期。
- 返回值类型：DATE。
- 示例：

```
select curdate;
+------------+
| curdate()  |
+------------+
| 2019-05-25 |
```

CURTIME

```
CURTIME()
```

- 命令说明：返回当前时间。
- 返回值类型：TIME。
- 示例：

```
select curtime();
+--------------+
| curtime()    |
+--------------+
| 14:39:22.109 |
```

DATE

```
DATE(expr)
```

- 命令说明：返回日期或日期时间表达式中的日期。
- 参数类型：

```
date(timestamp)
date(datetime)
date(varchar)
```

- 返回值类型：DATE。
- 示例：

```
select date(timestamp '2003-12-31 01:02:03');
+---------------------------------------+
| date(TIMESTAMP '2003-12-31 01:02:03') |
+---------------------------------------+
| 2003-12-31                            |
```

```
select date(datetime '2003-12-31 01:02:03');
+-------------------------------------+
| date(DATETIME '2003-12-31 01:02:03') |
+-------------------------------------+
| 2003-12-31                          |
```

```
select date('2003-12-31 01:02:03');
+-----------------------------+
| date('2003-12-31 01:02:03') |
+-----------------------------+
| 2003-12-31                  |
```

DATE_FORMAT

```
DATE_FORMAT(date,format)
```

- 命令说明：按照 `format` 指定的格式，将日期时间格式化成字符串。format格式如下所示。

%a	工作日缩写名称（Sun.. Sat）
%b	缩写的月份名称（Jan.. Dec）
%c	月，数字（0.. 12）
%d	每月的某一天，数字（00.. 31）
%e	每月的某一天，数字（0.. 31）
%f	微秒（000000... 999999）
%H	小时（00.. 23）
%h	小时（01.. 12）
%I	小时（01.. 12）
%i	分钟，数字（00.. 59）
%j	一年中的一天（001.. 366）
%k	小时（0.. 23）
%l	小时（1.. 12）
%M	月份名称（January.. December）
%m	月，数字（00.. 12）
%p	AM或PM
%r	时间，12小时（hh:mm:ss其次是AM或PM）
%S	秒（00... 59）
%s	秒（00... 59）
%T	时间，24小时（hh:mm:ss）
%v	本周是当年的第几周，星期一是一周的第一天，WEEK()模式3; 与%x 使用
%W	工作日名称（Sunday.. Saturday）
%x	本周所属年份，星期一是一周的第一天，四位数; 与%v使用
%Y	年份，数字，四位数
%y	年份，数字，两位数

%%	文字%字符
%x	x，对于上面未列出的任何x

- 参数类型：

```
date_format(timestamp, varchar)
date_format(varchar, varchar)
date_format(datetime, varchar)
date_format(date, varchar)
```

- 返回值类型：VARCHAR。
- 示例：

```
select date_format(timestamp '2019-05-27 13:23:00', '%W %M %Y')as result;
+-----------------+
| result      |
+-----------------+
| Monday May 2019 |
```

```
select date_format('2019-05-27 13:23:00', '%W %M %Y')as result;
+-----------------+
| result      |
+-----------------+
| Monday May 2019 |
```

```
select date_format(datetime '2019-05-27 13:23:00', '%W %M %Y')as result;
+-----------------+
| result      |
+-----------------+
| Monday May 2019 |
```

```
select date_format(date '2019-05-27', '%W %M %Y')as result;
+-----------------+
| result      |
+-----------------+
| Monday May 2019 |
```

SUBDATE/DATE_SUB

```
DATE_SUB(date,INTERVAL expr unit)
```

- 命令说明：返回 `date` 减去指定 `INTERVAL` 间隔后的日期。

unit 可取值为：second 、minute 、hour 、day 、month 、year 、minute_second 、hour_second 、hour_minute 、day_second 、day_minute 、day_hour 、year_month 。unit 默认值为 day 。

- 参数类型：

```
subdate(date, INTERVAL expr unit)
subdate(timestamp, INTERVAL expr unit)
subdate(datetime, INTERVAL expr unit)
subdate(varchar, INTERVAL expr unit)
subdate(date, bigint)
subdate(date, varchar)
subdate(datetime, bigint)
subdate(datetime, varchar)
subdate(timestamp, bigint)
subdate(timestamp, varchar)
subdate(varchar, bigint)
subdate(varchar, varchar)
```

- 返回值类型：DATE。
- 示例：

```
select date_sub(date '2001-1-22',interval '3' day);
+--------------------------------------------+
| date_sub(DATE '2001-1-22', INTERVAL  '3' DAY) |
+--------------------------------------------+
| 2001-01-19                       |
```

```
select date_sub(timestamp '2001-1-22 00:00:00',interval '3' day)as result;
+---------------------+
| result          |
+---------------------+
| 2001-01-19 00:00:00 |
```

```
select date_sub(datetime '2001-1-22 00:00:00',interval '3' day)as result;
+---------------------+
| result          |
+---------------------+
| 2001-01-19 00:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' day);
+-------------------------------------------------+
| date_sub('2001-1-22 00:00:00', INTERVAL  '3' DAY) |
+-------------------------------------------------+
| 2001-01-19 00:00:00                             |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' second);
+--------------------------------------------------+
| date_sub('2001-1-22 00:00:00', INTERVAL  '3' SECOND) |
+--------------------------------------------------+
| 2001-01-21 23:59:57                              |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' minute);
+--------------------------------------------------+
| date_sub('2001-1-22 00:00:00', INTERVAL  '3' MINUTE) |
+--------------------------------------------------+
| 2001-01-21 23:57:00                              |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' hour);
+------------------------------------------------+
| date_sub('2001-1-22 00:00:00', INTERVAL  '3' HOUR) |
+------------------------------------------------+
| 2001-01-21 21:00:00                            |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' month);
+-------------------------------------------------+
| date_sub('2001-1-22 00:00:00', INTERVAL  '3' MONTH) |
+-------------------------------------------------+
| 2000-10-22 00:00:00                             |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' year);
+------------------------------------------------+
| date_sub('2001-1-22 00:00:00', INTERVAL  '3' YEAR) |
+------------------------------------------------+
| 1998-01-22 00:00:00                            |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' minute_second)as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-21 23:57:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' hour_second)as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-21 21:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' hour_minute)as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-21 21:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' day_second)as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-19 00:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' day_minute)as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-19 00:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' day_hour)as result;
+---------------------+
| result              |
+---------------------+
| 2001-01-19 00:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',interval '3' year_month)as result;
+---------------------+
| result              |
+---------------------+
| 1998-01-22 00:00:00 |
```

```
select date_sub(date '2001-1-22 00:00:00',3);
+----------------------------------------+
| date_sub(DATE '2001-1-22 00:00:00', 3) |
+----------------------------------------+
| 2001-01-19                             |
```

```
select date_sub(date '2001-1-22 00:00:00','3');
+----------------------------------------+
| date_sub(DATE '2001-1-22 00:00:00', '3') |
+----------------------------------------+
| 2001-01-19                             |
```

```
select date_sub(datetime '2001-1-22 00:00:00',3);
+----------------------------------------+
| date_sub(DATETIME '2001-1-22 00:00:00', 3) |
+----------------------------------------+
|                  2001-01-19 00:00:00 |
```

```
select date_sub(datetime '2001-1-22 00:00:00','3');
+----------------------------------------+
| date_sub(DATETIME '2001-1-22 00:00:00', '3') |
+----------------------------------------+
|                  2001-01-19 00:00:00 |
```

```
select date_sub(timestamp '2001-1-22 00:00:00',3);
+----------------------------------------+
| date_sub(TIMESTAMP '2001-1-22 00:00:00', 3) |
+----------------------------------------+
|                  2001-01-19 00:00:00 |
```

```
select date_sub(timestamp '2001-1-22 00:00:00','3');
+-----------------------------------------------+
| date_sub(TIMESTAMP '2001-1-22 00:00:00', '3') |
+-----------------------------------------------+
|                   2001-01-19 00:00:00 |
```

```
select date_sub('2001-1-22 00:00:00',3);
+----------------------------------+
| date_sub('2001-1-22 00:00:00', 3) |
+----------------------------------+
| 2001-01-19 00:00:00           |
```

```
select date_sub('2001-1-22 00:00:00','3');
+-------------------------------------+
| date_sub('2001-1-22 00:00:00', '3') |
+-------------------------------------+
| 2001-01-19 00:00:00             |
```

DATEDIFF

```
DATEDIFF(expr1,expr2)
```

- 命令说明：返回 `expr1` 减去 `expr2` 后的天数。
- 参数类型：

```
datediff(varchar, varchar)
datediff(datetime, varchar)
datediff(varchar, datetime)
datediff(datetime, datetime)
datediff(varchar, timestamp)
datediff(timestamp, timestamp)
datediff(timestamp, varchar)
datediff(date, date)
datediff(date, varchar)
datediff(varchar, date)
```

- 返回值类型：BIGINT。
- 示例：

```
select datediff('2007-12-31 23:59:59','2007-12-30');
+----------------------------------------------+
| datediff('2007-12-31 23:59:59', '2007-12-30') |
+----------------------------------------------+
|                                            1 |
```

```
select datediff(datetime '2007-12-31 23:59:59','2007-12-30')as result;
+--------+
| result |
+--------+
|      1 |
```

```
select datediff('2007-12-31 23:59:59',datetime '2007-12-30')as result;
+--------+
| result |
+--------+
|      1 |
```

```
select datediff(datetime '2007-12-31 23:59:59',datetime '2007-12-30')as result;
+--------+
| result |
+--------+
|      1 |
```

```
select datediff('2007-12-31 23:59:59',timestamp '2007-12-30')as result;
+--------+
| result |
+--------+
|      1 |
```

```
select datediff(timestamp '2007-12-31 23:59:59',timestamp '2007-12-30')as result;
+--------+
| result |
+--------+
|      1 |
```

```
select datediff(timestamp '2007-12-31 23:59:59','2007-12-30')as result;
+--------+
| result |
+--------+
|    1 |
```

```
select datediff(date '2007-12-31 23:59:59',date '2007-12-30')as result;
+--------+
| result |
+--------+
|    1 |
```

```
select datediff(date '2007-12-31 23:59:59','2007-12-30')as result;
+--------+
| result |
+--------+
|    1 |
```

```
select datediff('2008-12-31',date '2007-12-30');
+-----------------------------------------+
| datediff('2008-12-31', DATE '2007-12-30') |
+-----------------------------------------+
                      367
```

DAY/DAYOFMONTH

```
DAY(date)
DAYOFMONTH(date)
```

- 命令说明：返回 `date` 中的日，取值范围 `[1,31]` 。
- 参数类型：

  ```
  dayofmonth(timestamp)
  dayofmonth(datetime)
  dayofmonth(date)
  dayofmonth(time)
  dayofmonth(varchar)
  ```

- 返回值类型：BIGINT。
- 示例：

```
select dayofmonth(timestamp '2007-02-03 12:23:09');
+-----------------------------------------------+
| dayofmonth(TIMESTAMP '2007-02-03 12:23:09') |
+-----------------------------------------------+
|                                             3 |
```

```
select dayofmonth(date '2007-02-03');
+--------------------------------+
| dayofmonth(DATE '2007-02-03') |
+--------------------------------+
|                              3 |
```

```
select dayofmonth(time '17:01:10');
+-----------------------------+
| dayofmonth(TIME '17:01:10') |
+-----------------------------+
|                          30 |
```

```
select day('2007-02-03');
+-------------------+
| day('2007-02-03') |
+-------------------+
|                 3 |
```

```
select dayofmonth(datetime '2007-02-03 00:00:00');
+-------------------------------------------------+
| dayofmonth(DATETIME '2007-02-03 00:00:00') |
+-------------------------------------------------+
|                                               3 |
```

DAYNAME

```
DAYNAME(date)
```

- 命令说明：返回日期对应的工作日的名称，例如星期一为 `Monday` 。
- 参数类型：

```
dayname(timestamp)
dayname(datetime)
dayname(date)
dayname(varchar)
```

- 返回值类型：VARCHAR。
- 示例：

```
select dayname(timestamp '2007-02-03 00:00:00');
+-----------------------------------------+
| dayname(TIMESTAMP '2007-02-03 00:00:00') |
+-----------------------------------------+
| Saturday                                |
```

```
select dayname(datetime '2007-02-03 00:00:00');
+-----------------------------------------+
| dayname(DATETIME '2007-02-03 00:00:00') |
+-----------------------------------------+
| Saturday                                |
```

```
select dayname(date '2007-02-04');
+----------------------------+
| dayname(DATE '2007-02-04') |
+----------------------------+
| Sunday                     |
```

```
select dayname('2007-02-03');
+----------------------+
| dayname('2007-02-03') |
+----------------------+
| Saturday              |
```

DAYOFWEEK

```
DAYOFWEEK(date)
```

- 命令说明：返回日期对应的工作日索引值，即星期日为 `1` ，星期一为 `2` ，星期六为 `7` 。
- 参数类型：

```
dayofweek(timestamp)
dayofweek(datetime)
dayofweek(date)
dayofweek(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select dayofweek(timestamp '2007-02-03 00:00:00');
+-------------------------------------------+
| dayofweek(TIMESTAMP '2007-02-03 00:00:00') |
+-------------------------------------------+
|                                         7
```

```
select dayofweek(datetime '2007-02-03 00:00:00');
+-----------------------------------------+
| dayofweek(DATETIME '2007-02-03 00:00:00') |
+-----------------------------------------+
|                                       7 |
```

```
select dayofweek(date '2007-02-03');
+-----------------------------+
| dayofweek(DATE '2007-02-03') |
+-----------------------------+
|                           7 |
```

```
select dayofweek('2007-02-03');
+-------------------------+
| dayofweek('2007-02-03') |
+-------------------------+
|       7         |
```

DAYOFYEAR

```
DAYOFYEAR(date)
```

- 命令说明：返回指定日期是当年的哪一天，返回值范围为 `[1,366]` 。
- 参数类型：

```
dayofyear(timestamp)
dayofyear(datetime)
dayofyear(date)
dayofyear(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select dayofyear(timestamp '2007-02-03 00:12:12');
+-------------------------------------------+
| dayofyear(TIMESTAMP '2007-02-03 00:12:12') |
+-------------------------------------------+
|                                        34 |
```

```
select dayofyear(datetime '2007-02-03 00:12:12');
+-------------------------------------------+
| dayofyear(DATETIME '2007-02-03 00:12:12') |
+-------------------------------------------+
|                                        34 |
```

```
select dayofyear(date '2007-02-03');
+-----------------------------+
| dayofyear(DATE '2007-02-03') |
+-----------------------------+
|                           34 |
```

```
select dayofyear('2007-02-03');
+-------------------------+
| dayofyear('2007-02-03') |
+-------------------------+
|          34             |
```

EXTRACT

```
EXTRACT(unit FROM date)
```

- 命令说明：返回日期或时间的单独部分，由 unit 指定，比如年、月、日、小时、分钟等。

  unit 可取值为： second 、 minute 、 hour 、 day 、 month 、 year 、 minute_second 、 hour_second 、 hour_minute 、 day_second 、 day_minute 、 day_hour 、 year_month 。
- 支持抽取的入参时间类型：VARCHAR、TIMESTAMP、DATETIME、TIME。
- 返回值类型：BIGINT。
- 示例：

```
select extract(second from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
|    34 |
```

```
select extract(minute from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
|  12 |
```

```
select extract(hour from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
|  0 |
```

```
select extract(month from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
|  7 |
```

```
select extract(minute_second from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
| 1234 |
```

```
select extract(hour_second from '2019-07-02 12:12:34');
+--------+
| _col0  |
+--------+
| 121234 |
```

```
select extract(hour_minute from '2019-07-02 12:12:34');
+-------+
| _col0 |
+-------+
| 1212 |
```

```
select extract(day_second from '2019-07-02 12:12:34');
+---------+
| _col0  |
+---------+
| 2121234 |
```

```
select extract(day_hour from '2019-07-02 12:12:34');
+-------+
| _col0 |
+-------+
|  212 |
```

```
select extract(day from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
|    2 |
```

```
select extract(year_month from '2019-07-02 00:12:34');
+--------+
| _col0  |
+--------+
| 201907 |
```

```
select extract(day_minute from '2019-07-02 00:12:34');
+-------+
| _col0 |
+-------+
| 20012 |
```

```
select extract(year from timestamp  '2019-05-30');
+-------+
| _col0 |
+-------+
| 2019 |
```

```
select extract(year from datetime  '2019-05-30');
+-------+
| _col0 |
+-------+
| 2019 |
```

```
select extract(year from time '15:23:22');
+-------+
| _col0 |
+-------+
| 2019 |
```

FROM_DAYS

```
FROM_DAYS(N)
```

- 命令说明：根据指定的天数 `N` ，返回对应的 `DATE` 值。
- 参数类型：

```
from_days(varchar)
from_days(bigint)
```

- 返回值类型：DATE。
- 示例：

```
select from_days(730669);
+------------------+
| from_days(730669) |
+------------------+
| 2000-07-03      |
```

```
select from_days('730669');
+--------------------+
| from_days('730669') |
+--------------------+
| 2000-07-03       |
```

FROM_UNIXTIME

```
FROM_UNIXTIME(unix_timestamp[,format])
```

- 命令说明：返回 `unixtime` 时间戳。

format 遵从DATE_FORMAT函数中的 format 格式。

- 参数类型：

```
from_unixtime(varchar, varchar)
from_unixtime(varchar)
from_unixtime(double, varchar)
from_unixtime(double)
```

- 返回值类型：DATETIME。
- 示例：

```
select from_unixtime('1447430881','%Y %M %h:%i:%s %x');
+------------------------------------------------+
| from_unixtime('1447430881', '%Y %M %h:%i:%s %x') |
+------------------------------------------------+
| 2015 November 12:08:01 2015                    |
```

```
select from_unixtime('1447430881');
+----------------------------+
| from_unixtime('1447430881') |
+----------------------------+
|      2015-11-14 00:08:01 |
```

```
select from_unixtime(1447430881);
+--------------------------+
| from_unixtime(1447430881) |
+--------------------------+
|     2015-11-14 00:08:01 |
```

```
select from_unixtime(1447430881,'%Y %M %h:%i:%s %x');
+------------------------------------------------+
| from_unixtime(1447430881, '%Y %M %h:%i:%s %x') |
+------------------------------------------------+
| 2015 November 12:08:01 2015                    |
```

HOUR

```
HOUR(time)
```

- 命令说明：返回时间中的小时。
- 参数类型：

```
hour(timestamp)
hour(datetime)
hour(date)
hour(time)
hour(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select hour(timestamp '2019-12-07 10:05:03');
+---------------------------------------+
| hour(TIMESTAMP '2019-12-07 10:05:03') |
+---------------------------------------+
|                                    10 |
```

```
select hour(datetime '2019-12-07 10:05:03');
+---------------------------------------+
| hour(DATETIME '2019-12-07 10:05:03') |
+---------------------------------------+
|                                    10 |
```

```
select hour(date '2019-12-07');
+-------------------------+
| hour(DATE '2019-12-07') |
+-------------------------+
|                       0 |
```

```
select hour(time '10:05:03');
+-----------------------+
| hour(TIME '10:05:03') |
+-----------------------+
|                    10 |
```

```
select hour('10:05:03');
+-------------------+
| hour('10:05:03') |
+-------------------+
|    10      |
```

LAST_DAY

```
LAST_DAY(date)
```

- 命令说明：返回日期或者日期时间中对应月份的最后一天。
- 参数类型：

```
last_day(varchar)
last_day(timestamp)
last_day(datetime)
last_day(date)
```

- 返回值类型：DATE。
- 示例：

```
select last_day('2003-02-05');
+-----------------------+
| last_day('2003-02-05') |
+-----------------------+
| 2003-02-28            |
```

```
select last_day(timestamp '2003-02-05 12:12:12');
+-----------------------------------------+
| last_day(TIMESTAMP '2003-02-05 12:12:12') |
+-----------------------------------------+
| 2003-02-28                              |
```

```
select last_day(datetime '2003-02-05 12:12:12');
+-----------------------------------------+
| last_day(DATETIME '2003-02-05 12:12:12') |
+-----------------------------------------+
| 2003-02-28                              |
```

```
select last_day(date '2003-02-05');
+-----------------------------+
| last_day(DATE '2003-02-05') |
+-----------------------------+
| 2003-02-28                  |
```

LOCALTIME/LOCALTIMESTAMP/NOW

```
localtime
localtime()
localtimestamp
localtimestamp()
now()
```

- 命令说明：返回当前时间戳。
- 返回值类型：DATETIME。
- 示例：

```
select now();
+---------------------+
| now()               |
+---------------------+
| 2019-05-25 00:28:37
```

```
select localtime;
+---------------------+
| localtime()         |
+---------------------+
| 2019-05-28 20:44:25 |
```

```
select localtime();
+---------------------+
| localtime()         |
+---------------------+
| 2019-05-31 17:37:36 |
```

```
select localtimestamp;
+---------------------+
| localtimestamp()    |
+---------------------+
| 2019-05-28 20:44:44 |
```

```
select localtimestamp();
+---------------------+
| localtimestamp()    |
+---------------------+
| 2019-05-31 17:38:13 |
```

MAKEDATE

```
MAKEDATE(year,dayofyear)
```

- 命令说明：按照参数 `year` 和 `dayofyear` ，返回一个日期。
- 参数类型：

```
makedate(bigint, bigint)
makedate(varchar, varchar)
```

- 返回值类型：DATE。
- 示例：

```
select makedate(2011,31), makedate(2011,32);
+-------------------+-------------------+
| makedate(2011, 31) | makedate(2011, 32) |
+-------------------+-------------------+
| 2011-01-31        | 2011-02-01        |
```

```
select makedate('2011','31'), makedate('2011','32');
+-----------------------+-----------------------+
| makedate('2011', '31') | makedate('2011', '32') |
+-----------------------+-----------------------+
| 2011-01-31            | 2011-02-01            |
```

MAKETIME

```
MAKETIME(hour,minute,second)
```

- 命令说明：按照参数 `hour` 、 `minute` 和 `second` ，返回一个时间。
- 参数类型：

```
maketime(bigint, bigint, bigint)
maketime(varchar, varchar, varchar)
```

- 返回值类型：TIME。
- 示例：

```
select maketime(12,15,30);
+---------------------+
| maketime(12, 15, 30) |
+---------------------+
| 12:15:30            |
```

```
select maketime('12','15','30');
+---------------------------+
| maketime('12', '15', '30') |
+---------------------------+
| 12:15:30                  |
```

MINUTE

```
MINUTE(time)
```

- 命令说明：返回时间中的分钟。
- 参数类型：

```
minute(timestamp)
minute(datetime)
minute(date)
minute(time)
minute(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select minute(timestamp '2008-02-03 10:05:03');
+-----------------------------------------+
| minute(TIMESTAMP '2008-02-03 10:05:03') |
+-----------------------------------------+
|                                       5 |
```

```
select minute(datetime '2008-02-03 10:05:03');
+-----------------------------------------+
| minute(DATETIME '2008-02-03 10:05:03') |
+-----------------------------------------+
|                                       5 |
```

```
select minute(date '2008-02-03');
+---------------------------+
| minute(DATE '2008-02-03') |
+---------------------------+
|                         0 |
```

```
select minute(time '12:12:12');
+-------------------------+
| minute(TIME '12:12:12') |
+-------------------------+
|                      12 |
```

```
select minute('2008-02-03 10:05:03');
+------------------------------+
| minute('2008-02-03 10:05:03') |
+------------------------------+
|       5            |
```

MONTH

```
MONTH(date)
```

- 命令说明：返回日期中的月份。
- 参数类型：

```
month(timestamp)
month(datetime)
month(date)
month(time)
month(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select month(timestamp '2008-02-03 00:00:00');
+---------------------------------------+
| month(TIMESTAMP '2008-02-03 00:00:00') |
+---------------------------------------+
|                          2 |
```

```
select month(datetime '2008-02-03 00:00:00');
+--------------------------------------+
| month(DATETIME '2008-02-03 00:00:00') |
+--------------------------------------+
|                          2 |
```

```
select month(date '2008-02-03');
+-------------------------+
| month(DATE '2008-02-03') |
+-------------------------+
|                2 |
```

MONTH函数也可以返回SQL执行时的月份，例如以下SQL是2019年5月执行的， 返回结果为5。

```
select month(time '12:12:12');
+-----------------------+
| month(TIME '12:12:12') |
+-----------------------+
|                     5 |
```

```
select month('2008-02-03');
+--------------------+
| month('2008-02-03') |
+--------------------+
|                  2 |
```

MONTHNAME

```
MONTHNAME(date)
```

- 命令说明：返回日期中月份的全名。
- 参数类型：

```
monthname(timestamp)
monthname(datetime)
monthname(date)
monthname(varchar)
```

- 返回值类型：VARCHAR。
- 示例：

```
select monthname(timestamp '2008-02-03 00:00:00');
+-------------------------------------------+
| monthname(TIMESTAMP '2008-02-03 00:00:00') |
+-------------------------------------------+
| February                                  |
```

```
select monthname(datetime '2008-02-03 00:00:00');
+----------------------------------------+
| monthname(DATETIME '2008-02-03 00:00:00') |
+----------------------------------------+
| February                               |
```

```
select monthname(date '2008-02-03');
+------------------------------+
| monthname(DATE '2008-02-03') |
+------------------------------+
| February                     |
```

```
select monthname('2008-02-03');
+-------------------------+
| monthname('2008-02-03') |
+-------------------------+
| February                |
```

PERIOD_ADD

```
PERIOD_ADD(P,N)
```

- 命令说明：将日期格式的参数 `P` 增加 `N` 个月。
- 参数类型：

```
period_add(bigint, bigint)
period_add(varchar, varchar)
period_add(varchar, bigint)
```

- 返回值类型：BIGINT。
- 示例：

```
select period_add(200801,2);
+-----------------------+
| period_add(200801, 2) |
+-----------------------+
|                200803 |
```

```
select period_add('200801','2');
+---------------------------+
| period_add('200801', '2') |
+---------------------------+
|                    200803 |
```

```
select period_add('200801',2);
+-------------------------+
| period_add('200801', 2) |
+-------------------------+
|                  200803 |
```

PERIOD_DIFF

```
PERIOD_DIFF(P1,P2)
```

- 命令说明：返回 `P1` 和 `P2` 之间相差的月数。
- 参数类型：

```
period_diff(bigint, bigint)
period_diff(varchar, varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select period_diff(200802,200703);
+-----------------------------+
| period_diff(200802, 200703) |
+-----------------------------+
|                          11 |
```

```
select period_diff('200802','200703');
+---------------------------------+
| period_diff('200802', '200703') |
+---------------------------------+
|                              11 |
```

QUARTER

```
QUARTER(date)
```

- 命令说明：返回日期在一年中的季度，取值范围为 `[1,4]` 。
- 参数类型：

```
quarter(datetime)
quarter(varchar)
quarter(timestamp)
quarter(date)
```

- 返回值类型：BIGINT。
- 示例：

```
select quarter(datetime '2008-04-01 12:12:12');
+-----------------------------------------+
| quarter(DATETIME '2008-04-01 12:12:12') |
+-----------------------------------------+
|                                       2 |
```

```
select quarter('2008-04-01');
+----------------------+
| quarter('2008-04-01') |
+----------------------+
|                    2 |
```

```
select quarter(timestamp '2008-04-01 12:12:12');
+------------------------------------------+
| quarter(TIMESTAMP '2008-04-01 12:12:12') |
+------------------------------------------+
|                                        2 |
```

```
select quarter(date '2008-04-01');
+---------------------------+
| quarter(DATE '2008-04-01') |
+---------------------------+
|                          2 |
```

SEC_TO_TIME

```
SEC_TO_TIME(seconds)
```

- 命令说明：将 `seconds` 转换为时间。
- 参数类型

```
sec_to_time(bigint)
sec_to_time(varchar)
```

- 返回值类型：TIME。
- 示例：

```
select sec_to_time(2378);
+------------------+
| sec_to_time(2378) |
+------------------+
| 00:39:38         |
```

```
select sec_to_time('2378');
+--------------------+
| sec_to_time('2378') |
+--------------------+
| 00:39:38
```

SECOND

```
SECOND(time)
```

- 命令说明：返回时间中的秒，范围为 `[0,59]` 。
- 参数类型：

```
second(timestamp)
second(datetime)
second(date)
second(time)
second(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select second(timestamp '2019-03-12 12:13:14');
+-----------------------------------------+
| second(TIMESTAMP '2019-03-12 12:13:14') |
+-----------------------------------------+
|                                      14 |
```

```
select second(datetime '2019-03-12 12:13:14');
+-----------------------------------------+
| second(DATETIME '2019-03-12 12:13:14') |
+-----------------------------------------+
|                                      14 |
```

```
select second(date '2019-03-12');
+--------------------------+
| second(DATE '2019-03-12') |
+--------------------------+
|                        0 |
```

```
select second(time '12:13:14');
+-------------------------+
| second(TIME '12:13:14') |
+-------------------------+
|                      14 |
```

```
select second('12:12:23');
+--------------------+
| second('12:12:23') |
+--------------------+
|                 23 |
```

STR_TO_DATE

```
STR_TO_DATE(str,format)
```

- 命令说明：按照指定日期或时间显示格式，将字符串转换为日期或日期时间类型。

  `format` 遵从DATE_FORMAT函数中的 `format` 格式。
- 参数类型：

  ```
  str_to_date(varchar, varchar)
  ```
- 返回值类型：DATETIME。
- 示例：

  ```
  select str_to_date('2017-01-06 10:20:30','%Y-%m-%d %H:%i:%s') as result;
  +---------------------+
  | result              |
  +---------------------+
  | 2017-01-06 10:20:30 |
  ```

SUBTIME

```
SUBTIME(expr1,expr2)
```

- 命令说明：返回 `expr1` 减去 `expr2` 后的时间。

- 参数类型：

```
subtime(date, varchar)
subtime(datetime, varchar)
subtime(timestamp, varchar)
subtime(time, varchar)
subtime(varchar, varchar)
```

- 返回值类型：DATETIME。
- 示例：

```
select subtime(date '2018-10-31','0:1:1');
+-------------------------------------+
| subtime(DATE '2018-10-31', '0:1:1') |
+-------------------------------------+
|             2018-10-30 23:58:59 |
```

```
select subtime(datetime '2018-10-31 12:12:12','0:1:1');
+-------------------------------------------------+
| subtime(DATETIME '2018-10-31 12:12:12', '0:1:1') |
+-------------------------------------------------+
|                     2018-10-31 12:11:11 |
```

```
select subtime(timestamp '2018-10-31 12:12:12','0:1:1');
+-------------------------------------------------+
| subtime(TIMESTAMP '2018-10-31 12:12:12', '0:1:1') |
+-------------------------------------------------+
|                     2018-10-31 12:11:11 |
```

```
select subtime(time '12:12:12','0:1:1');
+----------------------------------+
| subtime(TIME '12:12:12', '0:1:1') |
+----------------------------------+
| 12:11:11                         |
+----------------------------------+
```

```
select subtime('2018-10-31 23:59:59','0:1:1');
+----------------------------------------+
| subtime('2018-10-31 23:59:59', '0:1:1') |
+----------------------------------------+
| 2018-10-31 23:58:58                    |
```

SYSDATE

```
SYSDATE()
```

- 命令说明：获取系统时间。
- 返回值类型：DATETIME。
- 示例：

```
select sysdate();
+---------------------+
| sysdate()           |
+---------------------+
| 2019-05-26 00:47:21 |
```

TIME

```
TIME(expr)
```

- 命令说明：以字符串形式返回 `expr` 中的时间。
- 参数类型：

```
time(varchar)
time(datetime)
time(timestamp)
```

- 返回值类型：VARCHAR。
- 示例：

```
select time('2003-12-31 01:02:03');
+-----------------------------+
| time('2003-12-31 01:02:03') |
+-----------------------------+
| 01:02:03                    |
```

```
select time(datetime '2003-12-31 01:02:03');
+---------------------------------------+
| time(DATETIME '2003-12-31 01:02:03') |
+---------------------------------------+
| 01:02:03                              |
```

```
select time(timestamp '2003-12-31 01:02:03');
+----------------------------------------+
| time(TIMESTAMP '2003-12-31 01:02:03') |
+----------------------------------------+
| 01:02:03                               |
```

TIME_FORMAT

```
TIME_FORMAT(time,format)
```

- 命令说明：按照 format 指定的格式，以字符串格式显示时间 time 。

  format 遵从DATE_FORMAT函数中的 format 格式。
- 参数类型：

```
time_format(varchar, varchar)
time_format(timestamp, varchar)
time_format(datetime, varchar)
time_format(time, varchar)
time_format(date, varchar)
```

- 返回值类型：VARCHAR。
- 示例：

```
select time_format('12:00:00', '%H %k %h %I %l');
+-------------------------------------------+
| time_format('12:00:00', '%H %k %h %I %l') |
+-------------------------------------------+
| 12 12 12 12 12                            |
```

```
select time_format(timestamp '1998-01-01 23:00:00','%H %k %h %I %l')as result;
+----------------+
| result         |
+----------------+
| 23 23 11 11 11 |
```

```
select time_format(datetime '1998-01-01 23:00:00','%H %k %h %I %l')as result;
+----------------+
| result         |
+----------------+
| 23 23 11 11 11 |
```

```
select time_format(time '23:00:00','%H %k %h %I %l');
+----------------------------------------------+
| time_format(TIME '23:00:00', '%H %k %h %I %l') |
+----------------------------------------------+
| 23 23 11 11 11                               |
```

```
select time_format(date '1998-01-01','%H %k %h %I %l');
+----------------------------------------------+
| time_format(DATE '1998-01-01', '%H %k %h %I %l') |
+----------------------------------------------+
| 00 0 12 12 12                                |
```

TIME_TO_SEC

```
TIME_TO_SEC(time)
```

- 命令说明：返回 `time` 转换为秒的结果。
- 参数类型：

```
time_to_sec(varchar)
time_to_sec(datetime)
time_to_sec(timestamp)
time_to_sec(date)
time_to_sec(time)
```

- 返回值类型：BIGINT。
- 示例：

```
select time_to_sec(datetime '2009-12-12 22:23:00');
+-------------------------------------------+
| time_to_sec(DATETIME '2009-12-12 22:23:00') |
+-------------------------------------------+
|                                     80580 |
```

```
select time_to_sec(timestamp '2009-12-12 22:23:00');
+---------------------------------------------+
| time_to_sec(TIMESTAMP '2009-12-12 22:23:00') |
+---------------------------------------------+
|                                       80580 |
```

```
select time_to_sec(date '2009-12-12');
+--------------------------------+
| time_to_sec(DATE '2009-12-12') |
+--------------------------------+
|                              0 |
```

```
select time_to_sec(time '12:12:12');
+-----------------------------+
| time_to_sec(TIME '12:12:12') |
+-----------------------------+
|                       43932 |
```

```
select time_to_sec('22:23:00');
+-------------------------+
| time_to_sec('22:23:00') |
+-------------------------+
|                   80580 |
```

TIMEDIFF

```
TIMEDIFF(expr1,expr2)
```

- 命令说明：返回 `expr1` 减去 `expr2` 后的时间，与SUBTIME作用相同。
- 参数类型：

```
timediff(time, varchar)
timediff(time, time)
timediff(varchar, varchar)
```

- 返回值类型：DATETIME。
- 示例：

```
select timediff(time '12:00:00','10:00:00');
+---------------------------------------+
| timediff(TIME '12:00:00', '10:00:00') |
+---------------------------------------+
| 02:00:00                              |
```

```
select timediff('12:00:00','10:00:00');
+----------------------------------+
| timediff('12:00:00', '10:00:00') |
+----------------------------------+
| 02:00:00                         |
```

```
select timediff(time '12:00:00',time '10:00:00');
+-------------------------------------------+
| timediff(TIME '12:00:00', TIME '10:00:00') |
+-------------------------------------------+
| 02:00:00                                  |
```

TIMESTAMP

```
TIMESTAMP(expr)
```

- 命令说明：返回 `expr` 表示的日期或日期时间。
- 参数类型：

```
timestamp(date)
timestamp(varchar)
```

- 返回值类型：DATETIME。
- 示例：

```
select timestamp(date '2019-05-27');
+-----------------------------+
| timestamp(DATE '2019-05-27') |
+-----------------------------+
| 2019-05-27 00:00:00         |
```

```
select timestamp('2019-05-27');
+-------------------------+
| timestamp('2019-05-27') |
+-------------------------+
|    2019-05-27 00:00:00 |
```

TIMESTAMPADD

```
TIMESTAMPADD(unit,interval,datetime_expr)
```

- 命令说明：将 `interval` 添加到日期或日期时间表达式 `datetime_expr` 中。 `interval` 的单位由 `unit` 规定。

unit 可取值为：second 、minute 、hour 、day 、week 、month 、quarter 、year 。

- 参数类型：

```
timestampadd(varchar, varchar, timestamp)
timestampadd(varchar, bigint, timestamp)
timestampadd(varchar, varchar, date)
timestampadd(varchar, bigint, date)
timestampadd(varchar, varchar, datetime)
timestampadd(varchar, bigint, datetime)
timestampadd(varchar, varchar, varchar)
timestampadd(varchar, bigint, varchar)
```

- 返回值类型：DATETIME。
- 示例：

```
select timestampadd(second,'1',timestamp '2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 12:12:13 |
```

```
select timestampadd(second,1,timestamp '2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 12:12:13 |
```

```
select timestampadd(second,'1',date '2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 00:00:01 |
```

```
select timestampadd(second,1,date '2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 00:00:01 |
```

```
select timestampadd(second,'1',datetime '2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 12:12:13 |
```

```
select timestampadd(second,1,datetime '2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 12:12:13 |
```

```
select timestampadd(second,'1','2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 12:12:13 |
```

```
select timestampadd(second,1,'2003-01-02 12:12:12')as result;
+---------------------+
| result              |
+---------------------+
| 2003-01-02 12:12:13 |
```

```
select timestampadd(second,1,'2003-01-02 12:12:12');
+-----------------------------------------------------+
| timestampadd('second', 1, '2003-01-02 12:12:12') |
+-----------------------------------------------------+
| 2003-01-02 12:12:13                                 |
```

```
select timestampadd(minute,8820,'2019-08-24 09:00:00');
+-------------------------------------------------------+
| timestampadd('MINUTE', 8820, '2019-08-24 09:00:00') |
+-------------------------------------------------------+
| 2019-08-30 12:00:00
```

```
select timestampadd(hour,1,'2003-01-02 12:12:12');
+-------------------------------------------------+
| timestampadd('hour', 1, '2003-01-02 12:12:12') |
+-------------------------------------------------+
| 2003-01-02 13:12:12                             |
```

```
select timestampadd(day,1,'2003-01-02 12:12:12');
+------------------------------------------------+
| timestampadd('day', 1, '2003-01-02 12:12:12') |
+------------------------------------------------+
| 2003-01-03 12:12:12                            |
```

```
select timestampadd(week,1,'2003-01-02 12:12:12');
+-------------------------------------------------+
| timestampadd('week', 1, '2003-01-02 12:12:12') |
+-------------------------------------------------+
| 2003-01-09 12:12:12                             |
```

```
select timestampadd(month,1,'2003-01-02 12:12:12');
+--------------------------------------------------+
| timestampadd('month', 1, '2003-01-02 12:12:12') |
+--------------------------------------------------+
| 2003-02-02 12:12:12                              |
```

```
select timestampadd(year,1,'2003-01-02 12:12:12');
+--------------------------------------------------+
| timestampadd('year', 1, '2003-01-02 12:12:12') |
+--------------------------------------------------+
| 2004-01-02 12:12:12                              |
```

```
select timestampadd(quarter,1,'2003-01-02 12:12:12');
+----------------------------------------------------+
| timestampadd('quarter', 1, '2003-01-02 12:12:12') |
+----------------------------------------------------+
| 2003-04-02 12:12:12                                |
```

TIMESTAMPDIFF

```
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)
```

- 命令说明：返回日期或日期时间表达式 `datetime_expr1` 减去 `datetime_expr2` 后的结果，结果的单位由

unit 指定。

unit 可取值为： second 、 minute 、 hour 、 day 、 week 、 month 、 quarter 或 year 。

使用方法和TIMESTAMPADD相同。

- 参数类型：

```
timestampdiff(varchar, timestamp, timestamp)
timestampdiff(varchar, date, date)
timestampdiff(varchar, datetime, datetime)
timestampdiff(varchar, varchar, varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select timestampdiff(second,datetime '2003-02-01 10:12:13',datetime '2003-05-01 10:12:13')as result;
+---------+
| result  |
+---------+
| 7689600 |
```

```
select timestampdiff(minute,datetime '2003-02-01 10:12:13',datetime '2003-05-01 10:12:13')as result;
+--------+
| result |
+--------+
| 128160 |
```

```
select timestampdiff(hour,datetime '2003-02-01 10:12:13',datetime '2003-05-01 10:12:13')as result;
+--------+
| result |
+--------+
|   2136 |
```

```
select timestampdiff(day,timestamp '2003-02-01',timestamp '2003-05-01')as result;
+--------+
| result |
+--------+
|     89 |
```

```
select timestampdiff(day,date '2003-02-01',date '2003-05-01');
+-----------------------------------------------------------+
| timestampdiff('day', DATE '2003-02-01', DATE '2003-05-01') |
+-----------------------------------------------------------+
|                                                        89 |
```

```
select timestampdiff(day,datetime '2003-02-01 10:12:13',datetime '2003-05-01 10:12:13')as result;
+--------+
| result |
+--------+
|     89 |
```

```
select timestampdiff(day,'2003-02-01','2003-05-01');
+-------------------------------------------------+
| timestampdiff('day', '2003-02-01', '2003-05-01') |
+-------------------------------------------------+
|                                              89 |
```

```
select timestampdiff(week,'2003-02-01','2003-05-01');
+--------------------------------------------------+
| timestampdiff('week', '2003-02-01', '2003-05-01') |
+--------------------------------------------------+
|                                              12 |
```

```
select timestampdiff(quarter,'2003-02-01','2003-05-01');
+----------------------------------------------------+
| timestampdiff('quarter', '2003-02-01', '2003-05-01') |
+----------------------------------------------------+
|                                                  1 |
```

```
select timestampdiff(month,'2003-02-01','2003-05-01');
+---------------------------------------------------+
| timestampdiff('month', '2003-02-01', '2003-05-01') |
+---------------------------------------------------+
|                                                 3 |
```

```
select timestampdiff(year,datetime '2003-02-01 10:12:13',datetime '2001-05-01 10:12:13')as result;
+--------+
| result |
+--------+
|   -1 |
```

TO_DAYS

```
TO_DAYS(date)
```

- 命令说明：根据给定日期 `date` ，返回自 `0` 年开始的天数。
- 参数类型：

```
to_days(date)
to_days(time)
to_days(varchar)
to_days(timestamp)
to_days(datetime)
```

- 返回值类型：BIGINT。
- 示例：

```
select to_days(date '2018-12-12');
+----------------------------+
| to_days(DATE '2018-12-12') |
+----------------------------+
|                     737405 |
```

```
select to_days(time '12:12:12');
+--------------------------+
| to_days(TIME '12:12:12') |
+--------------------------+
|                   737572 |
```

```
select to_days(now());
+----------------+
| to_days(now()) |
+----------------+
|         737572 |
```

上述查询等价于 `to_days(curdate())`

```
select to_days(curdate());
+-------------------+
| to_days(curdate()) |
+-------------------+
|         737573 |
```

```
select to_days(datetime '2019-09-08 12:12:12');
+----------------------------------------+
| to_days(DATETIME '2019-09-08 12:12:12') |
+----------------------------------------+
|                       737675 |
```

```
select to_days('2019-09-08 12:12:12');
+--------------------------------+
| to_days('2019-09-08 12:12:12') |
+--------------------------------+
|                  737675 |
```

```
select to_days(timestamp '2019-09-08 12:12:12');
+-----------------------------------------+
| to_days(TIMESTAMP '2019-09-08 12:12:12') |
+-----------------------------------------+
|                       737675 |
```

TO_SECONDS

```
TO_SECONDS(expr)
```

- 命令说明：根据给定的 `expr` ，返回自 `0` 年开始的秒数。
- 参数类型：

```
to_seconds(date)
to_seconds(datetime)
to_seconds(timestamp)
to_seconds(varchar)
to_seconds(time)
```

- 返回值类型：BIGINT。
- 示例：

```
select to_seconds(date '2019-09-08');
+------------------------------+
| to_seconds(DATE '2019-09-08') |
+------------------------------+
|                  63735120000 |
```

```
select to_seconds(datetime '2019-09-08 09:09:00');
+-------------------------------------------+
| to_seconds(DATETIME '2019-09-08 09:09:00') |
+-------------------------------------------+
|                        63735152940 |
```

```
select to_seconds(timestamp '2019-09-08 09:09:00');
+-------------------------------------------+
| to_seconds(TIMESTAMP '2019-09-08 09:09:00') |
+-------------------------------------------+
|                        63735152940 |
```

执行以下SQL，系统将返回 `'09:09:00'` 加上 `curdate()` 后的结果。

```
select to_seconds(time '09:09:00');
+----------------------------+
| to_seconds(TIME '09:09:00') |
+----------------------------+
|                 63726253740 |
```

```
select to_seconds('2019-09-08');
+--------------------------+
| to_seconds('2019-09-08') |
+--------------------------+
|              63735120000 |
```

UNIX_TIMESTAMP

```
UNIX_TIMESTAMP([date])
```

- 命令说明：`UNIX_TIMESTAMP()` 返回自 `'1970-01-01 00:00:00' UTC` 以来秒数的Unix时间戳。`UNIX_TIMESTAMP(date)` 将参数的值返回为 `'1970-01-01 00:00:00'UTC` 后的秒数的Unix时间戳。
- 参数类型：

```
unix_timestamp()
unix_timestamp(varchar)
unix_timestamp(timestamp)
unix_timestamp(date)
unix_timestamp(datetime)
```

- 返回值类型：BIGINT。
- 示例：

```
select unix_timestamp();
+------------------+
| unix_timestamp() |
+------------------+
|     1558935850 |
```

```
select unix_timestamp(timestamp '2019-09-08 12:12:12');
+-------------------------------------------------+
| unix_timestamp(TIMESTAMP '2019-09-08 12:12:12') |
+-------------------------------------------------+
|                          1567915932 |
```

```
select unix_timestamp(date '2019-09-08');
+----------------------------------+
| unix_timestamp(DATE '2019-09-08') |
+----------------------------------+
|                 1567872000 |
```

```
select unix_timestamp(datetime '2019-09-08 12:12:12');
+-----------------------------------------------+
| unix_timestamp(DATETIME '2019-09-08 12:12:12') |
+-----------------------------------------------+
|                          1567915932 |
```

```
select unix_timestamp('2019-09-08 12:12:12');
+---------------------------------------+
| unix_timestamp('2019-09-08 12:12:12') |
+---------------------------------------+
|                   1567915932 |
```

UTC_DATE

```
UTC_DATE()
```

- 命令说明：返回UTC日期。
- 返回值类型：VARCHAR。
- 示例：

```
select utc_date();
+------------+
| utc_date() |
+------------+
| 2019-05-27 |
```

UTC_TIME

```
UTC_TIME()
```

- 命令说明：返回UTC时间。
- 返回值类型：VARCHAR。
- 示例：

```
select utc_time();
+------------+
| utc_time() |
+------------+
| 05:53:19   |
```

UTC_TIMESTAMP

```
utc_timestamp()
```

- 命令说明：返回UTC时间戳。
- 返回值类型：VARCHAR。
- 示例：

```
select utc_timestamp();
+---------------------+
| utc_timestamp()     |
+---------------------+
| 2019-05-27 05:55:15 |
```

WEEK

```
WEEK(date[,mode])
```

- 命令说明：返回 `date` 对应的周数，即 `date` 是日期年份中的哪一周。

- `date` 是要获取周数的日期。
- `mode` 可选参数，用于确定周数计算的逻辑。它允许您指定本周是从星期一还是星期日开始，返回的周数应在 `0` 到 `52` 之间或 `0` 到 `53` 之间。 `mode` 支持的格式如下表所示。

0	星期日	0-53
1	星期一	0-53
2	星期日	1-53
3	星期一	1-53
4	星期日	0-53
5	星期一	0-53
6	星期日	1-53
7	星期一	1-53

- 参数类型：

```
week(varchar)
week(varchar, bigint)
week(date)
week(date, bigint)
week(datetime)
week(datetime, bigint)
week(timestamp)
week(timestamp, bigint)
```

- 返回值类型：BIGINT。
- 示例：

```
select week('2019-05-27');
+--------------------+
| week('2019-05-27') |
+--------------------+
|                 21 |
```

```
select week('2008-02-20',1);
+-----------------------+
| week('2008-02-20', 1) |
+-----------------------+
|                     8 |
```

```
select week(date '2008-02-20');
+-------------------------+
| week(DATE '2008-02-20') |
+-------------------------+
|                       7 |
```

```
select week(date '2008-02-20',1);
+----------------------------+
| week(DATE '2008-02-20', 1) |
+----------------------------+
|                          8 |
```

```
select week(datetime '2008-02-20 00:00:00',1);
+-----------------------------------------+
| week(DATETIME '2008-02-20 00:00:00', 1) |
+-----------------------------------------+
|                                       8 |
```

```
select week(datetime '2008-02-20 00:00:00');
+--------------------------------------+
| week(DATETIME '2008-02-20 00:00:00') |
+--------------------------------------+
|                                    7 |
```

```
select week(timestamp '2008-02-20 00:00:00');
+---------------------------------------+
| week(TIMESTAMP '2008-02-20 00:00:00') |
+---------------------------------------+
|                                     7 |
```

```
select week(timestamp '2008-02-20 00:00:00',1);
+-----------------------------------------+
| week(TIMESTAMP '2008-02-20 00:00:00', 1) |
+-----------------------------------------+
|                                       8 |
```

WEEKDAY

```
WEEKDAY(date)
```

- 命令说明：返回 `date` 对应的工作日即 `0= Monday,1= Tuesday,... 6= Sunday` 。

- 参数类型：

```
weekday(timestamp)
weekday(datetime)
weekday(date)
weekday(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select weekday(timestamp '2019-05-27 00:09:00');
+-----------------------------------------+
| weekday(TIMESTAMP '2019-05-27 00:09:00') |
+-----------------------------------------+
|                                       0 |
```

```
select weekday(datetime '2019-05-27 00:09:00');
+-----------------------------------------+
| weekday(DATETIME '2019-05-27 00:09:00') |
+-----------------------------------------+
|                                       0 |
```

```
select weekday(date '2019-05-27 00:09:00');
+-------------------------------------+
| weekday(DATE '2019-05-27 00:09:00') |
+-------------------------------------+
|                                   0 |
```

```
select weekday('2019-05-27');
+----------------------+
| weekday('2019-05-27') |
+----------------------+
|                    0 |
```

WEEKOFYEAR

```
WEEKOFYEAR(date)
```

- 命令说明：返回 `date` 对应的日历周，取值范围为 `[1,53]` 。
- 参数类型：

```
weekofyear(timestamp)
weekofyear(datetime)
weekofyear(date)
weekofyear(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select weekofyear(timestamp '2019-05-27 09:00:00');
+-----------------------------------------------+
| weekofyear(TIMESTAMP '2019-05-27 09:00:00') |
+-----------------------------------------------+
|                                            22 |
```

```
select weekofyear(datetime '2019-05-27 09:00:00');
+-----------------------------------------------+
| weekofyear(DATETIME '2019-05-27 09:00:00') |
+-----------------------------------------------+
|                                            22 |
```

```
select weekofyear(date '2019-05-27');
+-------------------------------+
| weekofyear(DATE '2019-05-27') |
+-------------------------------+
|                            22 |
```

```
select weekofyear('2019-05-27');
+--------------------------+
| weekofyear('2019-05-27') |
+--------------------------+
|                       22 |
```

YEAR

```
YEAR(date)
```

- 命令说明：返回 `date` 中的年份。
- 参数类型：

```
year(timestamp)
year(datetime)
year(date)
year(time)
year(varchar)
```

- 返回值类型：BIGINT。
- 示例：

```
select year(timestamp '2019-05-27 00:00:00');
+---------------------------------------+
| year(TIMESTAMP '2019-05-27 00:00:00') |
+---------------------------------------+
|                                  2019 |
```

```
select year(datetime '2019-05-27 00:00:00');
+---------------------------------------+
| year(DATETIME '2019-05-27 00:00:00') |
+---------------------------------------+
|                                  2019 |
```

```
select year(date '2019-05-27');
+-------------------------+
| year(DATE '2019-05-27') |
+-------------------------+
|                    2019 |
```

执行以下SQL，系统将返回 `'00:00:00'` 加上 `curdate` 时间部分后的结果，结果数据类型为字符串。

```
select year(time '00:00:00');
+-----------------------+
| year(TIME '00:00:00') |
+-----------------------+
|                  2019 |
```

```
select year('2019-05-27');
+--------------------+
| year('2019-05-27') |
+--------------------+
|               2019 |
```

YEARWEEK

```
YEARWEEK(date)
YEARWEEK(date,mode)
```

- 命令说明：返回日期的年份和星期。

  返回结果中的年份可能与一年中第一周和最后一周的日期参数中的年份不同。

  `mode` 与WEEK函数中的 `mode` 作用相同。对于单参数语法， `mode` 值为 `0` 。
- 参数类型：

```
yearweek(timestamp)
yearweek(timestamp, bigint)
yearweek(datetime)
yearweek(datetime, bigint)
yearweek(date, bigint)
yearweek(date)
yearweek(varchar)
yearweek(varchar, bigint)
```

- 返回值类型：BIGINT。
- 示例：

```
select yearweek(timestamp '2019-05-27 00:00:00');
+-----------------------------------------+
| yearweek(TIMESTAMP '2019-05-27 00:00:00') |
+-----------------------------------------+
|                                  201921 |
```

```
select yearweek(timestamp '2019-05-27 00:00:00',1);
+--------------------------------------------+
| yearweek(TIMESTAMP '2019-05-27 00:00:00', 1) |
+--------------------------------------------+
|                                     201922 |
```

```
select yearweek(datetime '2019-05-27 00:00:00');
+---------------------------------------+
| yearweek(DATETIME '2019-05-27 00:00:00') |
+---------------------------------------+
|                                201921 |
```

```
select yearweek(datetime '2019-05-27 00:00:00',1);
+-------------------------------------------+
| yearweek(DATETIME '2019-05-27 00:00:00', 1) |
+-------------------------------------------+
|                                    201922 |
```

```
select yearweek(date '2019-05-27',1);
+--------------------------------+
| yearweek(DATE '2019-05-27', 1) |
+--------------------------------+
|                         201922 |
```

```
select yearweek(date '2019-05-27');
+-----------------------------+
| yearweek(DATE '2019-05-27') |
+-----------------------------+
|                      201921 |
```

```
select yearweek('2019-05-27');
+------------------------+
| yearweek('2019-05-27') |
+------------------------+
|                 201921 |
```

```
select yearweek('2019-05-27',1);
+---------------------------+
| yearweek('2019-05-27', 1) |
+---------------------------+
|                    201922 |
```

4.字符串函数

AnalyticDB for MySQL支持以下字符串函数。

- ASCII：返回字符或者字符串最左边字符对应的ASCII值。
- BIN：返回整数的二进制字符串。
- BIT_LENGTH：以位为单位返回参数str的长度。
- CHAR：返回整数对应的ASCII码组成的字符串。
- CHAR_LENGTH/CHARACTER_LENGTH：以字符为单位返回字符串的长度。
- CONCAT：连接字符串。
- CONCAT_WS：连接字符串，字符串中间以分隔串间隔。
- ELT：返回整数N指定的字符串。
- EXPORT_SET：根据bits中的比特值，返回组合后的字符串。
- FIELD：返回指定字符串在字符串列表中的索引位置。
- FIND_IN_SET：返回字符或字符串在另一个字符串中的位置。
- FORMAT：将数字N格式化，返回字符串。
- HEX：返回整数对应的十六进制字符串，或者返回字符串中每个字符对应的十六进制数所组成的字符串。
- INSTR：返回字符串中子字符串首次出现的位置。
- LEFT：从字符串最左边开始，返回N个字符。
- LENGTH/OCTET_LENGTH：字符串长度。
- LIKE：简单的模式匹配。
- LOCATE：返回字符串首次出现在另一个字符串中的位置信息。
- LOWER/LCASE：将字符串转换为小写。
- LPAD：左拼接字符串。
- LTRIM：删除字符串的前导空格。
- MAKE_SET：返回一组以逗号分隔的字符串。
- MID：从字符串的指定位置开始，返回指定长度的子字符串。作用同SUBSTR/SUBSTRING。
- OCT：返回指定整数的八进制字符串表示形式。
- POSITION：返回字符串中子字符串首次出现的位置。
- REPEAT：返回字符串重复多次后的字符串。
- REPLACE：用指定字符串替换另一个字符串中的部分字符。
- REVERSE：将字符串逆序。
- SPLIT：将字符串按分隔符进行分隔，并返回数组。
- RIGHT：返回字符串最右边的指定数量的字符。
- RLIKE/REGEXP：将字符串expression与pattern进行正则匹配，匹配成功返回1，否则返回0。
- RPAD：右拼接字符串。
- RTRIM：删除字符串的后置空格。
- SPACE：返回由指定数量空格组成的字符串。
- STRCMP：根据两个字符串的大小，返回0、1或者-1。
- SUBSTR/SUBSTRING：返回从指定位置开始的指定长度的子字符串。
- SUBSTRING_INDEX：返回字符串str中最后一次分隔符delim出现之前的子字符串。

- TRIM：删除字符串前后所有的空格。
- UPPER/UCASE：将字符串转换为大写。
- ORD：如果字符串最左边的字符是多字节字符，则返回该字符的代码。
- UNHEX：将参数中的每对字符转换为十六进制形式，再将其转换为数字表示的字节，最后以二进制字符串形式返回结果。
- ENCRYPT：对字符串进行加密。
- TO_BASE64：返回字符串的BASE64编码形式。
- FROM_BASE64：解码BASE64编码的字符串并返回结果。

ASCII

```
ASCII(varchar str)
```

- 命令说明：返回字符 `str` 或者字符串 `str` 最左边字符对应的十进制ASCII值。
- 返回值类型：BIGINT。
- 示例：

```
select ascii('2');
+------------+
| ascii('2') |
+------------+
|         50 |
```

```
select ascii('dx');
+-------------+
| ascii('dx') |
+-------------+
|         100 |
```

BIN

```
BIN(bigint N)
```

- 命令说明：返回 `N` 的二进制字符串。

  如果 `N` 为 `null`，则返回结果为 `null`。
- 返回值类型：VARCHAR。
- 示例：

```
select bin(12);
+---------+
| bin(12) |
+---------+
| 1100    |
```

BIT_LENGTH

```
BIT_LENGTH(varchar str)
```

- 命令说明：以位为单位返回参数 `str` 的长度。
- 返回值类型：BIGINT。
- 示例：

```
select bit_length('text');
+--------------------+
| bit_length('text') |
+--------------------+
|                 32 |
```

```
select bit_length('中国');
+--------------------+
| bit_length('中国') |
+--------------------+
|                 48 |
```

CHAR

```
CHAR(bigint N1, bigint N2...)
```

- 命令说明： 返回整数 `N1` 、 `N2` ...对应的十进制ASCII码组成的字符串。
- 返回值类型：VARBINARY。
- 示例：

```
select char(97,110,97,108,121,116,105,99,100,98);
+-------------------------------------------------+
| char(97, 110, 97, 108, 121, 116, 105, 99, 100, 98) |
+-------------------------------------------------+
| analyticdb                                      |
```

CHAR_LENGTH/CHARACTER_LENGTH

```
CHAR_LENGTH(varchar str)
```

- 命令说明： 以字符为单位返回字符串 `str` 的长度。

  一个汉字所对应的字符长度是 `1` 。
- 返回值类型：BIGINT。
- 示例：

```
select char_length('中国');
+----------------------+
| char_length('中国') |
+----------------------+
|                    2 |
```

```
select char_length('abc');
+--------------------+
| char_length('abc') |
+--------------------+
|                  3 |
```

CONCAT

```
concat(varchar str1, ···, varchar strn)
```

- 命令说明：字符串连接操作，其中任何一个参数为 `null` ，则返回值为 `null` 。
- 返回值类型：VARCHAR。
- 示例：

```
select concat('aliyun', ', ', 'analyticdb');
+---------------------------------------+
| concat('aliyun', ', ', 'analyticdb') |
+---------------------------------------+
| aliyun, analyticdb                    |
```

```
select concat('abc',null,'def');
+-----------------------------+
| concat('abc', null, 'def') |
+-----------------------------+
| NULL                        |
```

CONCAT_WS

```
concat_ws(varchar separator, varchar str1, ···, varchar strn)
```

- 命令说明：字符串连接操作，第一个参数 `separator` 是其余参数的分隔符，连接时会跳过任何为 `null` 值的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
select concat_ws(',', 'First name', 'Second name', 'Last Name')as result;
+--------------------------------+
| result                         |
+--------------------------------+
| First name,Second name,Last Name |
```

```
select concat_ws(',','First name',NULL,'Last Name')as result;
+---------------------+
| result              |
+---------------------+
| First name,Last Name |
```

ELT

```
ELT(bigint N, varchar str1, varchar str2, varchar str3,...)
```

- 命令说明：返回第 `N` 个字符串。

  若 `N<1` 或大于后面字符串参数的数量，则返结果为 `null` 。
- 返回值类型：VARCHAR。
- 示例：

```
select elt(4, 'Aa', 'Bb', 'Cc', 'Dd');
+-------------------------------+
| elt(4, 'Aa', 'Bb', 'Cc', 'Dd') |
+-------------------------------+
| Dd                            |
```

EXPORT_SET

```
EXPORT_SET(bigint bits, varchar on, varchar off[, varchar separator[, bigint number_of_bits]])
```

- 命令说明：返回一个字符串，根据整数 `bits` 的二进制位 `01` 值，从右到左（从低位到高位）放置 `on` 、`off` 字符串。 `1` 的位置放 `on` 字符串， `0` 的位置放 `off` 字符串，由分隔符字符串（默认为逗号字符）分隔。 检查位数由 `number_of_bits` 指定，如果未指定，默认值为 `64` 。

如果检查位数大于 64 ， number_of_bits 将被静默剪裁为 64 。

检查位数为 -1 、 64 ，返回结果相同。

- 返回值类型：VARCHAR。
- 示例：

```
select export_set(5,'1','0',',',2);
+---------------------------------+
| export_set(5, '1', '0', ',', 2) |
+---------------------------------+
| 1,0                             |
```

```
select  export_set(5,'1','0',',',10);
+----------------------------------+
| export_set(5, '1', '0', ',', 10) |
+----------------------------------+
| 1,0,1,0,0,0,0,0,0,0              |
```

FIELD

```
field(varchar str, varchar str1, varchar str2, varchar str3,...)
```

- 命令说明：返回 str 在 str1 、 str2 、 str3 ，...列表中的索引位置。 如果未找到 str ，则返回 0 。
- 返回值类型：BIGINT。
- 示例：

```
select field('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff');
+-----------------------------------------+
| field('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff') |
+-----------------------------------------+
|                                       2 |
```

FIND_IN_SET

```
FIND_IN_SET(varchar str, varchar strlist)
```

- 命令说明：返回 str 在列表 strlist 中的位置。

  如果 str 不在 strlist 中或者 strlist 是空字符串，返回结果为 0 。

  如果 str 、 strlist 任一参数为 null ，返回结果为 null 。
- 返回值类型：BIGINT。
- 示例：

```
select find_in_set('b','a,b,c,d');
+----------------------------+
| find_in_set('b', 'a,b,c,d') |
+----------------------------+
|                          2 |
```

FORMAT

```
format(double X, bigint D)
```

- 命令说明：将数字 `X` 格式化为 `#,###,###.##` 样式，舍入到 `D` 小数位，并将结果作为字符串返回。

  如果 `D` 为 `0`，则返回结果没有小数点或小数部分。
- 返回值类型：BIGINT。
- 示例：

```
select format(12332.123456, 4)as result1, format(12332.1,4)as result2, format(12332.2,0)as result3;
+-------------+-------------+---------+
| result1     | result2     | result3 |
+-------------+-------------+---------+
| 12,332.1235 | 12,332.1000 | 12,332  |
```

HEX

```
HEX(bigint N)
HEX(varchar str)
```

- 命令说明：返回整数 `N` 所对应的十六进制字符串，或者返回 `str` 中每个字符对应的十六进制数所组成的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
select hex(16);
+---------+
| hex(16) |
+---------+
| 10      |
```

```
select hex('16');
+-----------+
| hex('16') |
+-----------+
| 3136      |
```

INSTR

```
INSTR(varchar str, varchar substr)
```

- 命令说明：返回字符串 `str` 中子字符串 `substr` 首次出现的位置。
- 返回值类型：BIGINT。
- 示例：

```
select instr('foobarbar', 'bar');
+--------------------------+
| instr('foobarbar', 'bar') |
+--------------------------+
|                        4 |
```

LEFT

```
LEFT(varchar str, bigint len)
```

- 命令说明：返回字符串 `str` 中最左边的 `len` 个字符。

  如果 `str` 或者 `len` 为 `null`，则返回结果为 `null`。
- 返回值类型：VARCHAR。
- 示例：

```
select left('foobarbar', 5);
+----------------------------+
| left('foobarbar', 5) |
+----------------------------+
| fooba                |
```

LENGTH/OCTET_LENGTH

```
length(varchar str)
```

- 命令说明：返回字符串 `str` 的长度。
- 返回值类型：BIGINT。

- 示例：

```
select length('aliyun');
+-----------------+
| length('aliyun') |
+-----------------+
|               6 |
```

LIKE

```
expression [ NOT ] LIKE pattern [ESCAPE 'escape_char']
```

- 命令说明： `LIKE` 运算符用于将字符串 `expression` 与 `pattern` 进行匹配，匹配成功返回 `1` ，匹配失败返回 `0` 。

  `pattern` 为通配符模式，通配符包括：

  - `%` ：匹配任意长度的字符串。
  - `_` ：匹配单个字符。

  `escape_char` ：对 `pattern` 中的 `%` 、 `_` 进行转义，使得转义字符后面的 `%` 、 `_` 不作通配符使用。
- 返回值类型：BIGINT。
- 示例：

```
select 'David!' like 'David_' as result1, 'David!' not like 'David_' as result2, 'David!' like '%D%v%' as resul
t3;
+---------+---------+---------+
| result1 | result2 | result3 |
+---------+---------+---------+
|       1 |       0 |       1 |
```

```
select 'David_' like 'David|_' ESCAPE '|';
+-----------------------------------+
| 'David_' LIKE 'David|_' ESCAPE '|' |
+-----------------------------------+
|                                 1 |
```

LOCATE

```
LOCATE(varchar substr, varchar str)
LOCATE(varchar substr, varchar str, bigint pos)
```

- 命令说明：返回字符串 `str` 中首次出现 `substr` 的位置信息，或者返回字符串 `str` 中从指定位置 `pos` 开

始首次出现 substr 的位置信息。

如果 substr 不在 str 中，返回结果为 0 。

如果 substr 或者 str 为 null ，返回结果为 null 。

- 返回值类型：BIGINT。
- 示例：

```
select locate('bar', 'foobarbar');
+----------------------------+
| locate('bar', 'foobarbar') |
+----------------------------+
|                          4 |
```

```
select locate('bar', 'foobarbar', 7);
+-------------------------------+
| locate('bar', 'foobarbar', 7) |
+-------------------------------+
|                             7 |
```

LOWER/LCASE

```
lower(varchar str)
```

- 命令说明：将字符串 str 中的字母转换为小写。
- 返回值类型：VARCHAR。
- 示例：

```
select lower('Aliyun');
+-----------------+
| lower('Aliyun') |
+-----------------+
| aliyun          |
```

LPAD

```
LPAD(varchar str, bigint len, varchar padstr)
```

- 命令说明：将字符串 str 左边拼接 padstr 直到长度达到 len ，并返回拼接后的字符串。

  如果 str 长于 len ，则返回值将缩短为 len 个字符。
- 返回值类型：VARCHAR。
- 示例：

```
select lpad('Aliyun',9,'#');
+-----------------------+
| lpad('Aliyun', 9, '#') |
+-----------------------+
| ###Aliyun
```

LTRIM

```
LTRIM(varchar str)
```

- 命令说明：删除字符串 `str` 所有前导空格。
- 返回值类型：VARCHAR。
- 示例：

```
select ltrim('  abc');
+----------------+
| ltrim('  abc') |
+----------------+
| abc        |
```

MAKE_SET

```
MAKE_SET(bits, str1, str2,...)
```

- 命令说明：返回一个设置值（包含由字符分隔的子字符串的字符串），其中包含具有相应位设置的字符串。

  `str1` 对应于 `0` 位， `str2` 对应于 `1` 位，依此类推。 `str1` ， `str2` ，...中的 `null` 值不会附加到结果中。
- 返回值类型：VARCHAR。
- 示例：

```
select make_set(5,'hello','nice','world');
+-------------------------------------+
| make_set(5, 'hello', 'nice', 'world') |
+-------------------------------------+
| hello,world                 |
```

```
select make_set(1 | 4,'hello','nice',NULL,'world')as result;
+--------+
| result |
+--------+
| hello  |
```

MID

```
MID(varchar str, bigint pos, bigint len)
```

- 命令说明：与SUBSTR/SUBSTRING功能相同，从字符串 `str` 的 `pos` 开始返回 `len` 长度的子字符串。
- 返回值类型：VARCHAR。
- 示例：

```
select mid('Quadratically',5,6);
+----------------------------+
| mid('Quadratically', 5, 6) |
+----------------------------+
| ratica                     |
```

```
select mid('Sakila', -5, 3);
+--------------------------------+
| mid('Sakila', INTEGER '-5', 3) |
+--------------------------------+
| aki                            |
```

OCT

```
OCT(bigint N)
```

- 命令说明：返回整数 `N` 的八进制字符串表示形式。

  如果 `N` 为 `null` ，返回结果为 `null` 。
- 返回值类型：VARCHAR。
- 示例：

```
select oct(12);
+---------+
| oct(12) |
+---------+
| 14
```

POSITION

```
POSITION(varchar substr IN varchar str)
```

- 命令说明： 返回字符串 `str` 中子字符串 `substr` 首次出现位置，位置从 `1` 开始，如果未找到则返回 `0` 。
- 返回值类型：BIGINT。

- 示例：

```
select position('bar' in 'foobarbar');
+----------------------------+
| locate('bar', 'foobarbar') |
+----------------------------+
|                          4 |
```

REPEAT

```
REPEAT(varchar str, bigint count)
```

- 命令说明：返回由字符串 `str` 重复 `count` 次数组成的字符串。

  如果 `count<1` ，则返回空字符串。

  如果 `str` 或 `count` 为 `null` ，则返回 `null` 。
- 返回值类型：VARCHAR。
- 示例：

```
select repeat('a', 3);
+----------------+
| repeat('a', 3) |
+----------------+
| aaa            |
```

```
select  repeat('abc', null);
+---------------------+
| repeat('abc', null) |
+---------------------+
| NULL                |
```

```
select  repeat(null, 3);
+-----------------+
| repeat(null, 3) |
+-----------------+
| NULL            |
```

REPLACE

```
REPLACE(varchar str, varchar from_str, varchar to_str)
```

- 命令说明：将 `str` 中的 `from_str` 内容替换为 `to_str` 。

- 返回值类型：VARCHAR。
- 示例：

```
select replace('WWW.aliyun.com', 'W', 'w');
+-------------------------------------+
| replace('WWW.aliyun.com', 'W', 'w') |
+-------------------------------------+
| www.aliyun.com                      |
```

REVERSE

```
REVERSE(varchar str)
```

- 命令说明：返回 `str` 逆序后的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
select reverse('123456');
+-------------------+
| reverse('123456') |
+-------------------+
| 654321            |
```

SPLIT

```
split(string, delimiter)
split_part(string, delimiter, index)
split_to_map(string, entryDelimiter, keyValueDelimiter)
```

- 命令说明：
  - `split(string, delimiter)` ：将字符串string按分隔符delimiter进行分隔，并返回数组。
  - `split_part(string, delimiter, index)` ：将字符串string按分隔符delimiter分隔，并返回分隔后数组下标为index的子串，index以1开头，如果大于字段数则返回null。
  - `split_to_map(string, entryDelimiter, keyValueDelimiter)` ：通过entryDelimiter和keyValueDelimiter拆分字符串并返回map。entryDelimiter将字符串分解为key-value对，keyValueDelimiter将每对分隔成key、value。
- 返回值类型：VARCHAR或map<varchar, varchar>。
- 示例：

```
SELECT Split('1#2#3', '#'), Split('#1#2#3#', '#'),
    Split('123', '#');
| _col0   | _col1      | _col2   |
+----------+---------------+----------+
| [1, 2, 3] | [, 1, 2, 3, ] | [123]   |
SELECT Split_part('A#B#C', '#', 2),
    Split_part('A#B#C', '#', 4);
+--------------------+-------------------------+
| _col0          | _col1            |
+--------------------+-------------------------+
| B              | NULL             |
SELECT Split_to_map('k1:v1,k2:v2', ',', ':'),
    Split_to_map('', ',', ':');
+--------------------+-------------------------+
| _col0          | _col1            |
+--------------------+-------------------------+
| {k1=v1, k2=v2}    | {}               |
```

RIGHT

```
RIGHT(varchar str, bigint len
```

- 命令说明：返回字符串 `str` 中最右边的 `len` 个字符。

  如果 `str` 或者 `len` 为 `null` ，返回结果为 `null` 。
- 返回值类型：VARCHAR。
- 示例：

```
select right('abc',3);
+-----------------------+
| presto_right('abc', 3) |
+-----------------------+
| abc              |
```

RLIKE/REGEXP

```
expression  RLIKE pattern
expression  REGEXP pattern
```

- 命令说明：将字符串 `expression` 与 `pattern` 进行正则匹配，匹配成功返回 `1` ，否则返回 `0` 。

  如果 `expression` 或者 `pattern` 为 `null` ，返回结果为 `null` 。

- 返回值类型：BIGINT。
- 示例：

```
select 'Michael!' regexp '.*';
+------------------------------+
| regexp_like('Michael!', '.*') |
+------------------------------+
|                            1 |
```

```
select 'new*\n*line' regexp 'new\\*.\\*line';
+-----------------------------------------+
| regexp_like('new*
*line', 'new\\*.\\*line') |
+-----------------------------------------+
|                                       0 |
```

```
select 'c' regexp '^[a-d]';
+---------------------------+
| regexp_like('c', '^[a-d]') |
+---------------------------+
|                         1 |
```

RPAD

```
RPAD(varchar str, bigint len, varchar padstr)
```

- 命令说明：将字符串 `str` 右边拼接 `padstr` 直到长度达到 `len`，并返回拼接后的字符串。

  如果 `str` 长于 `len`，则返回值将缩短为 `len` 个字符。
- 返回值类型：VARCHAR。
- 示例：

```
select rpad('Aliyun',9,'#');
+-----------------------+
| rpad('Aliyun', 9, '#') |
+-----------------------+
| Aliyun###         |
```

RTRIM

```
RTRIM(varchar str)
```

- 命令说明：删除字符串 `str` 所有后置空格。

- 返回值类型：VARCHAR。
- 示例：

```
select rtrim('barbar   ');
+--------------------+
| rtrim('barbar   ') |
+--------------------+
| barbar             |
```

SPACE

```
SPACE(bigint N)
```

- 命令说明：返回由指定数量空格组成的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
select concat("#", space(6), "#");
+----------------------------+
| concat('#', space(6), '#') |
+----------------------------+
| #    #                     |
```

STRCMP

```
STRCMP(varchar str1, varchar str2)
```

- 命令说明：如果字符串 `str1` 、 `str1` 相同，返回结果为 `0` 。如果 `str1` 根据当前排序顺序小于 `str2` ，返回结果为 `-1` ，否则返回结果为 `1` 。
- 返回值类型：BIGINT。
- 示例：

```
select strcmp('text', 'text2');
+-------------------------+
| strcmp('text', 'text2') |
+-------------------------+
|                      -1 |
```

SUBSTR/SUBSTRING

```
SUBSTRING(varchar str, bigint pos)
SUBSTRING(varchar str FROM pos)
SUBSTRING(varchar str, bigint pos, bigint len)
SUBSTRING(varchar str FROM pos FOR len)
```

- 命令说明：
  - `SUBSTRING(varchar str, bigint pos)`、`SUBSTRING(varchar str FROM pos)`返回从`pos`位置开始到字符串结束的子串。如果`pos<0`，则起始位置从字符串的末尾开始倒数。
  - `SUBSTRING(varchar str, bigint pos, bigint len)`、`SUBSTRING(varchar str FROM pos FOR len)`返回从`pos`位置开始长度为`len`的子串。 如果`pos<0`，则起始位置从字符串的末尾开始倒数。
- 返回值类型：VARCHAR。
- 示例：

```
select substr('helloworld', 6);
+------------------------+
| substr('helloworld', 6) |
+------------------------+
| world
```

```
select substr('helloworld' from 6);
+------------------------+
| substr('helloworld', 6) |
+------------------------+
| world                  |
```

```
select substr('helloworld', 6, 3);
+---------------------------+
| substr('helloworld', 6, 3) |
+---------------------------+
| wor                       |
```

```
select substr('helloworld' from 6 for 3);
+---------------------------+
| substr('helloworld', 6, 3) |
+---------------------------+
| wor                       |
```

SUBSTRING_INDEX

```
SUBSTRING_INDEX(varchar str, varchar delim, bigint count)
```

- 命令说明：返回字符串 `str` 中最后一次分隔符 `delim` 出现之前的子字符串。

  如果 `count>0` ，返回最后一次 `delim` 左侧的所有内容，即从左侧开始计算。

  如果 `count<0` ，返回最后一次 `delim` 右侧的所有内容，即从右侧开始计算。

  搜索 `delim` 时， `SUBSTRING_INDEX` 函数区分大小写。
- 返回值类型：VARCHAR。
- 示例：

```
select substring_index('www.aliyun.com', '.', 2);
+-----------------------------------------+
| substring_index('www.aliyun.com', '.', 2) |
+-----------------------------------------+
| www.aliyun                              |
```

TRIM

```
TRIM([remstr FROM] str)
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str)
```

- 命令说明：通过删除前导空格和尾随空格或删除与可选的指定字符串匹配的字符来剪裁字符串。
- 返回值类型：VARCHAR。
- 示例：

```
select trim('  bar   ');
+------------------+
| trim('  bar   ') |
+------------------+
| bar              |
```

```
select trim(both 'x' from 'xxxbarxxx');
+------------------------+
| trim('x', 'xxxbarxxx') |
+------------------------+
| bar                    |
```

```
select trim(leading 'x' from 'xxxbarxxx');
+-------------------------+
| ltrim('x', 'xxxbarxxx') |
+-------------------------+
| barxxx                  |
```

```
select trim(trailing 'x' from 'xxxbarxxx');
+-------------------------+
| rtrim('x', 'xxxbarxxx') |
+-------------------------+
| xxxbar                  |
```

UPPER/UCASE

```
upper(varchar str)
```

- 命令说明：将字符串 `str` 中的字母转换为大写。
- 返回值类型：VARCHAR。
- 示例：

```
select upper('Aliyun');
+-----------------+
| upper('Aliyun') |
+-----------------+
| ALIYUN          |
```

ORD

```
ord(varbinary x)
```

- 命令说明：如果字符串 `x` 最左边的字符是多字节字符，则返回该字符的代码。
- 返回值类型：LONG
- 示例：

```
select ord(CAST('中国' AS VARBINARY));
+---------------------------------+
| ord(CAST('中国' AS varbinary))  |
+---------------------------------+
|                             228 |
```

UNHEX

```
unhex(varbinary x)
```

- 命令说明：将参数 `x` 中的每对字符解释为十六进制数，并将其转换为数字表示的字节，以二进制字符串将其返回。
- 返回值类型：VARBINARY
- 示例：

```
select unhex(CAST('中国' AS VARBINARY));
+-----------------------------------+
| unhex(CAST('中国' AS varbinary)) |
+-----------------------------------+
| NULL                              |
```

ENCRYPT

```
encrypt(varbinary x, varchar y)
```

- 命令说明：对参数 `x` 进行加密， `y` 为Salt值。
- 返回值类型：VARBINARY
- 示例：

```
select encrypt('abdABC123','key');
+----------------------------+
| encrypt('abdABC123', 'key') |
+----------------------------+
| kezazmcIo.aCw              |
```

TO_BASE64

```
to_base64(varbinary x)
```

- 命令说明：返回字符串 `x` 的BASE64编码形式。
- 返回值类型：VARCHAR
- 示例：

```
select to_base64(CAST('中国' AS VARBINARY));
+--------------------------------------+
| to_base64(CAST('中国' AS varbinary)) |
+--------------------------------------+
| 5Lit5Zu9                             |
```

FROM_BASE64

```
from_base64(varbinary x)
```

- 命令说明：解码BASE64编码的字符串 `x` 并返回结果。
- 返回值类型：VARBINARY
- 示例：

```
select from_base64(to_base64(CAST('abc' AS VARBINARY)));
+-------------------------------------------------+
| from_base64(to_base64(CAST('abc' AS varbinary))) |
+-------------------------------------------------+
| abc                                             |
```

5.位函数和操作符

AnalyticDB for MySQL支持以下位函数和操作符。

- BIT_COUNT：系统先将参数转换为二进制，然后返回二进制中1的个数。
- &：按位AND。
- ~：反转所有位。
- |：按位OR。
- ^：按位异或。
- >>（BIT WISE_RIGHT_SHIFT）：向右移位。
- <<（BIT WISE_LEFT_SHIFT）：向左移位。

BIT_COUNT

```
bit_count(bigint x)
bit_count(double x)
bit_count(varchar x)
```

- 命令说明：系统先将参数转换为二进制，然后返回二进制中 `1` 的个数。
- 返回值类型：BIGINT。
- 示例：

```
select bit_count(2);
+--------------+
| bit_count(2) |
+--------------+
|            1 |
```

```
select bit_count(pi());
+-----------------+
| bit_count(pi()) |
+-----------------+
|               2 |
```

```
select bit_count('123');
+------------------+|
bit_count('123') |
+------------------+
|               6 |
```

&

- 命令说明：按位 `AND` 。

- 返回值类型：BIGINT。
- 示例：

```
 select 12 & 15;
+--------------------+
| bitwise_and(12, 15) |
+--------------------+
|                 12 |
```

~

- 命令说明：反转所有位。
- 返回值类型：BIGINT。
- 示例：

```
select 2 & ~1;
+--------------------------------+
| bitwise_and(2, bitwise_not(1)) |
+--------------------------------+
|                              2 |
```

|

- 命令说明：按位 `OR` 。
- 返回值类型：BIGINT。
- 示例：

```
select 29 | 15;
+--------------------+
| bitwise_or(29, 15) |
+--------------------+
|                 31
```

^

- 命令说明：按位异或。
- 返回值类型：BIGINT。
- 示例：

```
select 1 ^ 10;
+--------------------+
| bitwise_xor(1, 10) |
+--------------------+
|                 11 |
```

>>（BITWISE_RIGHT_SHIFT）

```
bitwise_right_shift(double x, double y)
bitwise_right_shift(varchar x, varchar y)
bitwise_right_shift(bigint x, bigint y)
```

- 命令说明：向右移位。
- 返回值类型：BIGINT。
- 示例：

```
select 3 >> 2;
+--------------------------+
| bitwise_right_shift(3, 2) |
+--------------------------+
|                        0 |
```

```
select 3.4 >> 23.2;
+-------------------------------+
| bitwise_right_shift(3.4, 23.2) |
+-------------------------------+
|                             0 |
```

<<（BITWISE_LEFT_SHIFT）

```
bitwise_left_shift(double x, double y)
bitwise_left_shift(varchar x, varchar y)
bitwise_left_shift(bigint x, bigint y)
```

- 命令说明：向左移位。
- 返回值类型：BIGINT。
- 示例：

```
SELECT 3 << 2;
+-------------------------+
| bitwise_left_shift(3, 2) |
+-------------------------+
|                       12 |
```

```
select '3' << '2';
+------------------------------+
| bitwise_left_shift('3', '2') |
+------------------------------+
|                           12 |
```

```
select 3.4 << 23.2;
+--------------------------------+
| bitwise_left_shift(3.4, 23.2) |
+--------------------------------+
|                       25165824 |
```

6.GEO函数

AnalyticDB for MySQL支持的GEO函数包括ST_Point或Point、ST_AsText、ST_GeometryFromText或ST_GeomFromText、ST_Distance、ST_Distance_Sphere，本文介绍GEO函数的用法。

ST_Point或Point

```
ST_Point(x, y)
```

- 命令说明：先将两个DOUBLE类型的数值x和y进行除法运算 `x/y` ，然后将 `x/y` 的结果值为坐标构造一个POINT类型的值。
- 返回值类型：GEOMETRY类型的POINT对象，直接通过SELECT查询函数结果将显示乱码。
- 示例：

```
SELECT ST_Point(1,1);
+----------------------+
| ST_Point(1,1)      |
+----------------------+
|    ?    ? |
```

ST_AsText

```
ST_AsText(g)
```

- 命令说明：返回g的WKT(Well-Known Text)格式。
- 示例：

```
SELECT ST_AsText(ST_Point(1,1));
+--------------------------+
| ST_AsText(ST_Point(1,1)) |
+--------------------------+
| POINT (1 1)          |
```

ST_GeometryFromText或ST_GeomFromText

```
ST_GeometryFromText(wkt)
```

- 命令说明：使用WKT格式的字符串构造GEOMETRY值。
- 返回值类型：GEOMETRY类型的对象，直接通过SELECT查询函数结果将显示乱码。
- 示例：

```
SELECT ST_GeometryFromText('Point(1 1)');
+-----------------------------------+
| ST_GeometryFromText('Point(1 1)') |
+-----------------------------------+
|    ?   ?          |
```

```
SELECT ST_AsText(ST_GeometryFromText('Point(1 1)'));
+---------------------------------------------+
| ST_AsText(ST_GeometryFromText('Point(1 1)')) |
+---------------------------------------------+
| POINT (1 1)                  |
```

ST_Distance

```
ST_Distance(g1, g2)
```

- 命令说明：返回g1、g2之间的直线距离。
- 返回值类型：DOUBLE。
- 示例：

```
SELECT ST_Distance(ST_Point(1,1), ST_Point(2,2));
+-----------------------------------------+
| ST_Distance(ST_Point(1,1), ST_Point(2,2)) |
+-----------------------------------------+
|              1.4142135623730951 |
```

ST_Distance_Sphere

```
ST_Distance_Sphere(g1, g2 [, radius])
```

- 命令说明：返回g1、g2之间的球面距离，可以指定球的半径radius，radius默认值为6370986米。
- 返回值类型：DOUBLE。
- 示例：

```
SELECT ST_Distance_Sphere(point(1,1), point(2,2));
+-----------------------------------------+
| ST_Distance_Sphere(point(1,1), point(2,2)) |
+-----------------------------------------+
|              157225.08654191086 |
```

```
SELECT ST_Distance_Sphere(point(1,1), point(2,2), 6370986);
+-----------------------------------------------------+
| ST_Distance_Sphere(point(1,1), point(2,2), 6370986) |
+-----------------------------------------------------+
|                   157225.08654191086 |
```

7.JSON函数

本文介绍AnalyticDB for MySQL中JSON函数的用法。

JSON _EXTRACT

```
json_extract(json, jsonpath)
```

- 命令说明：从 `json` 中返回 `jsonpath` 指定的值。
- 返回值类型：JSON。
- 示例：

```
select json_extract('[10, 20, [30, 40]]', '$[1]');
 +-------------------------------------------------+
| 20 |
```

JSON _ARRAY_CONTAINS

```
json_array_contains(json, value)
```

- 命令说明：判断 `json` 中是否包含 `value` 指定的值。
- 返回值类型：BOOLEAN。
- 示例：

```
select json_array_contains('[1, 2, 3]', 2);
 +-----------------------------------------+
| 1 |
```

JSON_SIZE

```
json_size(json, jsonpath)
```

- 命令说明：返回 `json` 的大小。
- 返回值类型：BIGINT。
- 示例：

```
select json_size('{"x":{"a":1, "b": 2}}', '$.x') as result;
 +-----------------------------------------------------------+
| 2 |
```

```
select json_size('{"x": {"a": 1, "b": 2}}', '$.x.a') as result;
 +-----------------------------------------------------------+
| 0 |
```

JSON_ARRAY_LENGTH

```
json_array_length(json)
```

- 命令说明：返回 `json` 数组的长度。
- 返回值类型：BIGINT。
- 示例：

```
select json_array_length('[1, 2, 3]')
 +----------------------------------+
|          3          |
```

JSON_KEYS

```
json_keys(json, jsonpath)
```

- 命令说明：获取 `json` 在指定路径下的所有键值。
- 返回值类型：JSON ARRAY。
- 示例：

```
select json_keys(cast('{"a": 1, "b": {"c": 30}}' as json),'$.b')
 +-----------------------------------------------------------+
|               ["c"]                 |
```

8.窗口函数

AnalyticDB for MySQL支持以下窗口函数。

- 聚合函数
- 排序函数
  - CUME_DIST：返回一组数值中每个值的累计分布。
  - RANK：返回数据集中每个值的排名。
  - DENSE_RANK：返回一组数值中每个数值的排名。
  - NTILE：将每个窗口分区的数据分散到桶号从1到n的n个桶中。
  - ROW_NUMBER：根据行在窗口分区内的顺序，为每行数据返回一个唯一的有序行号，行号从1开始。
  - PERCENT_RANK：返回数据集中每个数据的排名百分比，其结果由 `(r - 1) / (n - 1)` 计算得出。其中r为RANK()计算的当前行排名， n为当前窗口分区内总的行数。
- 值函数
  - FIRST_VALUE：返回窗口分区第1行的值。
  - LAST_VALUE返回窗口分区最后1行的值。
  - LAG：返回窗口内距离当前行之前偏移offset后的值。
  - LEAD：返回窗口内距离当前行偏移offset后的值。
  - NTH_VALUE：返回窗口内偏移指定offset后的值，偏移量从1开始。

概述

窗口函数基于查询结果的行数据进行计算，窗口函数运行在 `HAVING` 子句之后、 `ORDER BY` 子句之前。窗口函数需要特殊的关键字 `OVER` 子句来指定窗口即触发一个窗口函数。

分析型数据库MySQL版支持三种类型的窗口函数：聚合函数、排序函数和值函数。

语法

```
function over (partition by a order by b RANGE|ROWS BETWEEN start AND end)
```

窗口函数包含以下三个部分。

- 分区规范：用于将输入行分散到不同的分区中，过程和 `GROUP BY` 子句的分散过程相似。
- 排序规范：决定输入数据行在窗口函数中执行的顺序。
- 窗口区间：指定计算数据的窗口边界。

  窗口区间支持 `RANGE` 、 `ROWS` 两种模式：
  - `RANGE` 按照计算列值的范围进行定义。
  - `ROWS` 按照计算列的行数进行范围定义。

- `RANGE`、`ROWS`中可以使用`BETWEEN start AND end`指定边界可取值。`BETWEEN start AND end`取值为：
  - `CURRENT ROW`，当前行。
  - `N PRECEDING`，前`n`行。
  - `UNBOUNDED PRECEDING`，直到第`1`行。
  - `N FOLLOWING`，后`n`行。
  - `UNBOUNDED FOLLOWING`，直到最后`1`行。

例如，以下查询根据当前窗口的每行数据计算`profit`的部分总和。

```
select year,country,profit,sum(profit) over (partition by country order by year ROWS BETWEEN UNBOUNDE
D PRECEDING and CURRENT ROW) as slidewindow from testwindow;
+------+---------+--------+-------------+
| year | country | profit | slidewindow |
+------+---------+--------+-------------+
| 2001 | USA     |   50 |       50 |
| 2001 | USA     | 1500 |     1550 |
| 2000 | India   |   75 |       75 |
| 2000 | India   |   75 |      150 |
| 2001 | India   |   79 |      229 |
| 2000 | Finland | 1500 |     1500 |
| 2001 | Finland |   10 |     1510 |
```

而以下查询只能计算出`profit`的总和。

```
select country,sum(profit) over (partition by country) from testwindow;
+---------+----------------------------------------+
| country | sum(profit) OVER (PARTITION BY country) |
+---------+----------------------------------------+
| India   |                                    229 |
| India   |                                    229 |
| India   |                                    229 |
| USA     |                                   1550 |
| USA     |                                   1550 |
| Finland |                                   1510 |
| Finland |                                   1510 |
```

注意事项

边界值的取值有如下要求：

- `start`不能为`UNBOUNDED FOLLOWING`，否则提示`Window frame start cannot be UNBOUNDED FOLLOWI`

NG 错误。

- end 不能为 UNBOUNDED PRECEDING ，否则提示 Window frame end cannot be UNBOUNDED PRECEDING 错误。
- start 为 CURRENT ROW 并且 end 为 N PRECEDING 时，将提示 Window frame starting from CURRENT ROW cannot end with PRECEDING 错误。
- start 为 N FOLLOWING 并且 end 为 N PRECEDING 时，将提示 Window frame starting from FOLLOWING cannot end with PRECEDING 错误。
- start 为 N FOLLOWING 并且 end 为 CURRENT ROW ，将提示 Window frame starting from FOLLOWING cannot end with CURRENT ROW 错误。

当模式为 RANGE 时：

- start 或者 end 为 N PRECEDING 时，将提示 Window frame RANGE PRECEDING is only supported with UNBOUNDED 错误。
- start 或者 end 为 N FOLLOWING 时，将提示 Window frame RANGE FOLLOWING is only supported with UNBOUNDED 错误。

准备工作

本文中的窗口函数均以 testwindow 表为测试数据。

```
create table testwindow(year int, country varchar(20), product varchar(20), profit int) distributed by hash(year);
```

```
insert into testwindow values (2000,'Finland','Computer',1500);
insert into testwindow values (2001,'Finland','Phone',10);
insert into testwindow values (2000,'India','Calculator',75);
insert into testwindow values (2000,'India','Calculator',75);
insert into testwindow values (2001,'India','Calculator',79);
insert into testwindow values (2001,'USA','Calculator',50);
insert into testwindow values (2001,'USA','Computer',1500);
```

```
SELECT * FROM testwindow;
+------+---------+------------+--------+
| year | country | product    | profit |
+------+---------+------------+--------+
| 2000 | Finland | Computer   |   1500 |
| 2001 | Finland | Phone      |     10 |
| 2000 | India   | Calculator |     75 |
| 2000 | India   | Calculator |     75 |
| 2001 | India   | Calculator |     79 |
| 2001 | USA     | Calculator |     50 |
| 2001 | USA     | Computer   |   1500 |
```

聚合函数

所有聚合函数都可以通过添加 `OVER` 子句来作为窗口函数使用，聚合函数将基于当前滑动窗口内的数据行计算每一行数据。

例如，通过以下查询循环显示每个店员每天的订单额总和。

```
SELECT clerk, orderdate, orderkey, totalprice,sum(totalprice) OVER (PARTITION BY clerk ORDER BY orderdate) AS rolling_sum FROM orders ORDER BY clerk, orderdate, orderkey
```

CUME_DIST

```
CUME_DIST()
```

- 命令说明：返回一组数值中每个值的累计分布。

  返回结果：在窗口分区中对窗口进行排序后的数据集，包括当前行和当前行之前的数据行数。排序中任何关联值均会计算成相同的分布值。
- 返回值类型：DOUBLE。
- 示例：

```
select year,country,product,profit,cume_dist() over (partition by country order by profit) as cume_dist
from testwindow;
+------+---------+------------+--------+--------------------+
| year | country | product    | profit | cume_dist          |
+------+---------+------------+--------+--------------------+
| 2001 | USA     | Calculator |     50 |                0.5 |
| 2001 | USA     | Computer   |   1500 |                1.0 |
| 2001 | Finland | Phone      |     10 |                0.5 |
| 2000 | Finland | Computer   |   1500 |                1.0 |
| 2000 | India   | Calculator |     75 | 0.6666666666666666 |
| 2000 | India   | Calculator |     75 | 0.6666666666666666 |
| 2001 | India   | Calculator |     79 |                1.0 |
```

RANK

```
RANK()
```

- 命令说明：返回数据集中每个值的排名。

  排名值是将当前行之前的行数加1，不包含当前行。因此，排序的关联值可能产生顺序上的空隙，而且这个排名会对每个窗口分区进行计算。
- 返回值类型：BIGINT。
- 示例：

```
select year,country,product,profit,rank() over (partition by country order by profit) as rank from testwi
ndow;
+------+---------+------------+--------+------+
| year | country | product    | profit | rank |
+------+---------+------------+--------+------+
| 2001 | Finland | Phone      |     10 |    1 |
| 2000 | Finland | Computer   |   1500 |    2 |
| 2001 | USA     | Calculator |     50 |    1 |
| 2001 | USA     | Computer   |   1500 |    2 |
| 2000 | India   | Calculator |     75 |    1 |
| 2000 | India   | Calculator |     75 |    1 |
| 2001 | India   | Calculator |     79 |    3 |
```

DENSE_RANK

```
DENSE_RANK()
```

- 命令说明：返回一组数值中每个数值的排名。

`DENSE_RANK()` 与 `RANK()` 功能相似，但是 `DENSE_RANK()` 关联值不会产生顺序上的空隙。

- 返回值类型：BIGINT。
- 示例：

```
select year,country,product,profit,dense_rank() over (partition by country order by profit) as dense_rank from testwindow;
+------+---------+------------+--------+------------+
| year | country | product    | profit | dense_rank |
+------+---------+------------+--------+------------+
| 2001 | Finland | Phone      |  10    |      1     |
| 2000 | Finland | Computer   |  1500  |      2     |
| 2001 | USA     | Calculator |  50    |      1     |
| 2001 | USA     | Computer   |  1500  |      2     |
| 2000 | India   | Calculator |  75    |      1     |
| 2000 | India   | Calculator |  75    |      1     |
| 2001 | India   | Calculator |  79    |      2     |
```

NTILE

```
NTILE(n)
```

- 命令说明：将每个窗口分区的数据分散到桶号从 `1` 到 `n` 的 `n` 个桶中。

  桶号值最多间隔 `1` ，如果窗口分区中的数据行数不能均匀地分散到每一个桶中，则剩余值将从第 `1` 个桶开始，每 `1` 个桶分 `1` 行数据。例如，有6行数据和4个桶， 最终桶号值为 `112234` 。

- 返回值类型：BIGINT。
- 示例：

```
select year,country,product,profit,ntile(2) over (partition by country order by profit) as ntile2 from testwindow;
+------+---------+------------+--------+--------+
| year | country | product    | profit | ntile2 |
+------+---------+------------+--------+--------+
| 2001 | USA     | Calculator |  50    |   1    |
| 2001 | USA     | Computer   |  1500  |   2    |
| 2001 | Finland | Phone      |  10    |   1    |
| 2000 | Finland | Computer   |  1500  |   2    |
| 2000 | India   | Calculator |  75    |   1    |
| 2000 | India   | Calculator |  75    |   1    |
| 2001 | India   | Calculator |  79    |   2    |
```

ROW_NUMBER

```
ROW_NUMBER()
```

- 命令说明：根据行在窗口分区内的顺序，为每行数据返回一个唯一的有序行号，行号从 1 开始。
- 返回值类型：BIGINT。
- 示例：

```
SELECT year, country, product, profit, ROW_NUMBER() OVER(PARTITION BY country) AS row_num1 FRO
M testwindow;
+------+---------+------------+--------+----------+
| year | country | product    | profit | row_num1 |
+------+---------+------------+--------+----------+
| 2001 | USA     | Calculator |   50 |        1 |
| 2001 | USA     | Computer   | 1500 |        2 |
| 2000 | India   | Calculator |   75 |        1 |
| 2000 | India   | Calculator |   75 |        2 |
| 2001 | India   | Calculator |   79 |        3 |
| 2000 | Finland | Computer   | 1500 |        1 |
| 2001 | Finland | Phone      |   10 |        2 |
```

PERCENT_RANK

```
PERCENT_RANK()
```

- 命令说明：返回数据集中每个数据的排名百分比，其结果由 (r - 1) / (n - 1) 计算得出。其中， r 为 RANK() 计算的当前行排名， n 为当前窗口分区内总的行数。
- 返回值类型：DOUBLE。
- 示例：

```
select year,country,product,profit,PERCENT_RANK() over (partition by country order by profit) as ntile3
from testwindow;
+------+---------+------------+--------+--------+
| year | country | product    | profit | ntile3 |
+------+---------+------------+--------+--------+
| 2001 | Finland | Phone      |   10 |  0.0 |
| 2000 | Finland | Computer   | 1500 |  1.0 |
| 2001 | USA     | Calculator |   50 |  0.0 |
| 2001 | USA     | Computer   | 1500 |  1.0 |
| 2000 | India   | Calculator |   75 |  0.0 |
| 2000 | India   | Calculator |   75 |  0.0 |
| 2001 | India   | Calculator |   79 |  1.0 |
```

FIRST_VALUE

```
FIRST_VALUE(x)
```

- 命令说明：返回窗口分区第一行的值。
- 返回值类型：与输入参数类型相同。
- 示例：

```
 select year,country,product,profit,first_value(profit) over (partition by country order by profit) as firstV
alue from testwindow;
 +------+---------+------------+--------+------------+
 | year | country | product    | profit | firstValue |
 +------+---------+------------+--------+------------+
 | 2000 | India   | Calculator |   75 |      75 |
 | 2000 | India   | Calculator |   75 |      75 |
 | 2001 | India   | Calculator |   79 |      75 |
 | 2001 | USA     | Calculator |   50 |      50 |
 | 2001 | USA     | Computer   | 1500 |      50 |
 | 2001 | Finland | Phone      |   10 |      10 |
 | 2000 | Finland | Computer   | 1500 |      10 |
```

LAST_VALUE

```
LAST_VALUE(x)
```

- 命令说明：返回窗口分区最后一行的值。LAST_VALUE默认统计范围是 rows between unbounded preceding and current row，即取当前行数据与当前行之前的数据进行比较。如果像FIRST_VALUE那样直接在每行数据中显示最后一行数据，需要在 order by 条件的后面加上语句：rows between unbounded preceding and unbounded following。
- 返回值类型：与输入参数类型相同。
- 示例1：

```
select year,country,product,profit,last_value(profit) over (partition by country order by profit) as firstVal
ue from testwindow;
+----------------+-------------------+-------------------+-----------------+----------------------+
| year       | country       | product       | profit       | firstValue       |
+----------------+-------------------+-------------------+-----------------+----------------------+
|       2001 | USA         | Calculator     |         50 |           50 |
|       2001 | USA         | Computer       |       1500 |         1500 |
|       2001 | Finland       | Phone        |         10 |           10 |
|       2000 | Finland       | Computer       |       1500 |         1500 |
|       2000 | India         | Calculator     |         75 |           75 |
|       2000 | India         | Calculator     |         75 |           75 |
|       2001 | India         | Calculator     |         79 |           79 |
```

- 示例2：

```
select year,country,product,profit,last_value(profit) over (partition by country order by profit rows betw
een unbounded preceding and unbounded following) as lastValue from testwindow;
+------+---------+------------+--------+-----------+
| year | country | product    | profit | lastValue |
+------+---------+------------+--------+-----------+
| 2001 | Finland | Phone      |   10 |    1500 |
| 2000 | Finland | Computer   | 1500 |    1500 |
| 2000 | India   | Calculator |   75 |      79 |
| 2000 | India   | Calculator |   75 |      79 |
| 2001 | India   | Calculator |   79 |      79 |
| 2001 | USA     | Calculator |   50 |    1500 |
| 2001 | USA     | Computer   | 1500 |    1500 |
+------+---------+------------+--------+-----------+
```

LAG

```
LAG(x[, offset[, default_value]])
```

- 命令说明：返回窗口内距离当前行之前偏移 `offset` 后的值。

  偏移量起始值是 `0` ，也就是当前数据行。偏移量可以是标量表达式，默认 `offset` 是 `1` 。

  如果偏移量的值是 `null` 或者大于窗口长度，则返回 `default_value` ；如果没有指定 `default_value` ，则返回 `null` 。
- 返回值类型：与输入参数类型相同。
- 示例：

```
select year,country,product,profit,lag(profit) over (partition by country order by profit) as lag from test window;
+------+---------+------------+--------+------+
| year | country | product    | profit | lag  |
+------+---------+------------+--------+------+
| 2001 | USA     | Calculator |     50 | NULL |
| 2001 | USA     | Computer   |   1500 |   50 |
| 2000 | India   | Calculator |     75 | NULL |
| 2000 | India   | Calculator |     75 |   75 |
| 2001 | India   | Calculator |     79 |   75 |
| 2001 | Finland | Phone      |     10 | NULL |
| 2000 | Finland | Computer   |   1500 |   10 |
```

LEAD

```
LEAD(x[,offset[, default_value]])
```

- 命令说明：返回窗口内距离当前行偏移 `offset` 后的值。

  偏移量 `offset` 起始值是 `0` ，也就是当前数据行。偏移量可以是标量表达式，默认 `offset` 是 `1` 。

  如果偏移量的值是 `null` 或者大于窗口长度，则返回 `default_value` ；如果没有指定 `default_value` ，则返回 `null` 。
- 返回值类型：与输入参数类型相同。
- 示例：

```
select year,country,product,profit,lead(profit) over (partition by country order by profit) as lead from testwindow;
+------+---------+------------+--------+------+
| year | country | product    | profit | lead |
+------+---------+------------+--------+------+
| 2000 | India   | Calculator |     75 |   75 |
| 2000 | India   | Calculator |     75 |   79 |
| 2001 | India   | Calculator |     79 | NULL |
| 2001 | Finland | Phone      |     10 | 1500 |
| 2000 | Finland | Computer   |   1500 | NULL |
| 2001 | USA     | Calculator |     50 | 1500 |
| 2001 | USA     | Computer   |   1500 | NULL |
```

NTH_VALUE

```
NTH_VALUE(x, offset)
```

- 命令说明：返回窗口内偏移指定 `offset` 后的值，偏移量从 `1` 开始。

  如果偏移量 `offset` 是 `null` 或者大于窗口内值的个数，则返回 `null` ；如果偏移量 `offset` 为 `0` 或者负数，则系统提示报错。
- 返回值类型：与输入参数类型相同。
- 示例：

```
select year,country,product,profit,nth_value(profit,1) over (partition by country order by profit) as nth_value from testwindow;
+------+---------+------------+--------+-----------+
| year | country | product    | profit | nth_value |
+------+---------+------------+--------+-----------+
| 2001 | Finland | Phone      |   10 |     10 |
| 2000 | Finland | Computer   | 1500 |     10 |
| 2001 | USA     | Calculator |   50 |     50 |
| 2001 | USA     | Computer   | 1500 |     50 |
| 2000 | India   | Calculator |   75 |     75 |
| 2000 | India   | Calculator |   75 |     75 |
| 2001 | India   | Calculator |   79 |     75 |
```

9.聚合函数

本文介绍AnalyticDB for MySQL中的聚合函数。

- AVG：该函数用于计算平均值。
- BIT_AND：返回参数所有位按位AND后的结果。
- BIT_OR：返回参数所有位按位OR后的结果。
- BIT_XOR：返回参数所有位按位异或后的结果。
- COUNT：该函数用于计算记录数。
- MAX：该函数用于计算最大值。
- MIN：该函数用于计算最小值。
- STD/STDDEV：返回数值的样本标准偏差。
- STDDEV_POP：返回数值的总体标准差。
- STDDEV_SAMP：返回一组数值（整数、小数或浮点）的总体标准差。
- SUM：该函数用于计算汇总值。
- VAR_POP：返回一组数值（整数、小数或浮点）的总体方差。
- VAR_SAMP：返回一组数值（整数、小数或浮点）的样本方差。
- VARIANCE：返回一组数值（整数、小数或浮点）的总体方差。

本文中的聚合函数均以 `testtable` 表为测试数据。

```
create table testtable(a int) distributed by hash(a);
```

```
insert into testtable values (1),(2),(3);
```

AVG

```
avg(bigint x)
avg(double x)
avg(float x)
```

- 命令说明：该函数用于计算平均值。
- 返回值类型：DOUBLE。
- 示例：

```
select avg(a) from testtable;
+--------+
| avg(a) |
+--------+
|   2.0 |
```

BIT_AND

```
bit_and(float x)
bit_and(bigint x)
bit_and(double x)
```

- 命令说明：返回参数所有位按位 `AND` 后的结果。
- 返回值类型：BIGINT。
- 示例：

```
select bit_and(a) from testtable;
+------------+
| bit_and(a) |
+------------+
|          0 |
```

BIT_OR

```
bit_or(float x)
bit_or(bigint x)
bit_or(double x)
```

- 命令说明：返回参数所有位按位 `OR` 后的结果。
- 返回值类型：BIGINT。
- 示例：

```
select bit_or(a) from testtable;
+----------------------------+
|       bit_or(a)            |
+----------------------------+
|       3                    |
```

BIT_XOR

```
bit_xor(double x)
bit_xor(bigint x)
bit_xor(float x)
```

- 命令说明：返回参数所有位按位异或后的结果。
- 返回值类型：BIGINT。
- 示例：

```
select bit_xor(a) from testtable;
+----------------------------+
|      bit_xor(a)     |
+----------------------------+
|      0          |
```

COUNT

```
count([distinct|all] value x)
```

- 命令说明：该函数用于计算记录数。

  `distinct` 、 `all` 指明在计数时是否去除重复记录，默认 `all` ，即返回全部记录。如果指定 `distinct` ，返回结果只计算唯一值数量。
- 返回值类型：BIGINT。
- 示例：

```
select count(distinct a) from testtable;
+------------------+
| count(DISTINCT a) |
+------------------+
|          3 |
```

MAX

```
max(value x)
```

- 命令说明：该函数用于计算最大值。

  `value` 可以为任意数据类型，但是 `BOOLEAN` 类型的数据不允许参与运算。

  当列中的值为 `null` 时，该行不参与计算。
- 返回值类型：LONG。
- 示例：

```
select max(a) from testtable;
+--------+
| max(a) |
+--------+
|    3 |
```

MIN

```
min(value x)
```

- 命令说明：该函数用于计算最小值。

  `value` 可以为任意数据类型，但是 `BOOLEAN` 类型的数据不允许参与运算。

  当列中的值为 `null` 时，该行不参与计算。
- 返回值类型：LONG。
- 示例：

```
select min(a) from testtable;
+--------+
| min(a) |
+--------+
|    1 |
```

STD/STDDEV

```
std(double x)
std(bigint x)
stddev(double x)
stddev(bigint x)
```

- 命令说明：返回数值的样本标准偏差。
- 返回值类型：DOUBLE。
- 示例：

```
select std(a) from testtable;
+-------------------+
| std(a)        |
+-------------------+
| 0.816496580927726 |
```

STDDEV_POP

```
stddev_pop(double x)
stddev_pop(bigint x)
```

- 命令说明：返回数值的总体标准差。
- 返回值类型：DOUBLE。
- 示例：

```
select stddev_pop(a) from testtable;
+----------------------------+
|  stddev_pop(a)       |
+----------------------------+
|  0.816496580927726    |
```

STDDEV_SAMP

```
stddev_samp(double x)
stddev_samp(bigint x)
```

- 命令说明：返回一组数值（整数、小数或浮点）的总体标准差。
- 返回值类型：DOUBLE。
- 示例：

```
select stddev_samp(a) from testtable;
+----------------------------+
|  stddev_samp(a)     |
+----------------------------+
|     1.0          |
```

SUM

```
sum(double x)
sum(float x)
sum(bigint x)
```

- 命令说明：该函数用于计算汇总值。
- 返回值类型：BIGINT。
- 示例：

```
select sum(a) from testtable;
+----------------------------+
|     sum(a)        |
+----------------------------+
|     6          |
```

VAR_POP

```
var_pop(double x)
var_pop(bigint x)
```

- 命令说明：返回一组数值x（整数、小数或浮点）的总体标准方差。也可以使用VARIANCE()函数，具有相同的意义，但VARIANCE()不是标准的SQL。若找不到匹配的项，则VAR_POP()返回NULL。
- 返回值类型：DOUBLE。
- 示例：

```
select var_pop(a)  from testtable;
+---------------------------+
|     var_pop(a)      |
+---------------------------+
| 0.6666666666666666    |
```

VAR_SAMP

```
var_samp(double x)
var_samp(bigint x)
```

- 命令说明：返回一组数值（整数、小数或浮点）的样本方差。
- 返回值类型：DOUBLE。
- 示例：

```
select var_samp(a)  from testtable;
+---------------------------+
|     var_samp(a)     |
+---------------------------+
|       1.0         |
```

VARIANCE

```
variance(double x)
variance(bigint x)
```

- 命令说明：返回一组数值（整数、小数或浮点）的总体标准方差。VARIANCE()作为标准SQL的延伸，也可以使用标准SQL函数 VAR_POP()来代替。若找不到匹配的项，则VARIANCE()返回NULL。
- 返回值类型：DOUBLE。
- 示例：

```
select variance(a)  from testtable;
+---------------------------+
|     variance(a)     |
+---------------------------+
| 0.6666666666666666    |
```

10.加密和压缩函数

AnalyticDB for MySQL支持以下功能为加密和压缩的函数。

- MD5：计算参数MD5的hash值。
- SHA1：计算字符串的SHA-1校验和。
- SHA2：计算SHA-2校验和。
- COMPRESS：压缩一个字符串并将结果作为二进制字符串返回。
- UNCOMPRESS：解压缩由COMPRESS()函数压缩的字符串。
- UNCOMPRESSED_LENGTH：在压缩之前返回一个字符串的长度。
- AES_ENCRYPT：使用AES算法进行数据加密。
- AES_DECRYPT：使用AES算法进行数据解密。

MD5

```
md5(varbinary x)
```

- 命令说明：返回参数 `x` MD5的hash值。
- 返回值类型：VARCHAR
- 示例：

```
select md5(CAST('中国' AS VARBINARY));
+----------------------------------+
| md5(CAST('中国' AS varbinary))  |
+----------------------------------+
| c13dceabcb143acd6c9298265d618a9f |
```

SHA1

```
sha1(varbinary x)
```

- 命令说明：计算字符串 `x` 的SHA-1校验和。
- 返回值类型：VARCHAR
- 示例：

```
select sha1(CAST('中国' AS VARBINARY));
+------------------------------------------+
| sha1(CAST('中国' AS varbinary))       |
+------------------------------------------+
| 101806f57c322fb403a9788c4c24b79650d02e77 |
```

SHA2

```
sha2(varbinary x, integer y)
sha2(varbinary x, varchar y)
```

- 命令说明：计算字符串 x SHA-2系列散列函数（SHA-224、SHA-256、SHA-384和SHA-512）。 x 是要清理的明文字符串， y 表示结果所需的位数长度，其值必须为224、256、384、512或0。
- 返回值类型：VARCHAR
- 示例：

```
select sha2(CAST('中国' AS VARBINARY),256);
+------------------------------------------------------------------+
| sha2(CAST('中国' AS varbinary), 256)                             |
+------------------------------------------------------------------+
| f0e9521611bb290d7b09b8cd14a63c3fe7cbf9a2f4e0090d8238d22403d35182 |
```

COMPRESS

```
compress(varbinary x)
```

- 命令说明：压缩字符串 x 并将结果作为二进制字符串返回。
- 返回值类型：VARBINARY
- 示例：

```
select hex(compress(CAST('中国' AS VARBINARY)));
+-------------------------------------------+
| hex(compress(CAST('中国' AS varbinary)))  |
+-------------------------------------------+
| 06000000789C7BB263EDD3D97B01104C0487      |
```

UNCOMPRESS

```
uncompress(varbinary x)
```

- 命令说明：解压缩 x 由COMPRESS()函数压缩的字符串。
- 返回值类型：VARBINARY
- 示例：

```
select uncompress(compress(CAST('中国' AS VARBINARY)));
+-------------------------------------------------+
| uncompress(compress(CAST('中国' AS varbinary))) |
+-------------------------------------------------+
| 中国                                            |
```

UNCOMPRESSED_LENGTH

```
uncompressed_length(varbinary x)
```

- 命令说明：在压缩之前返回字符串 `x` 的长度。
- 返回值类型：LONG
- 示例：

```
select uncompressed_length(compress(CAST('中国' AS VARBINARY)));
+-----------------------------------------------------------+
| uncompressed_length(compress(CAST('中国' AS varbinary))) |
+-----------------------------------------------------------+
|                                                         6 |
```

AES_ENCRYPT

```
aes_encrypt(varbinary x, varchar y)
```

- 命令说明：使用AES算法加密 `x` ， `y` 为密钥。
- 返回值类型：VARBINARY
- 示例：

```
select HEX(aes_encrypt(CAST('中国' AS VARBINARY), '0123'));
+-----------------------------------------------------+
| HEX(aes_encrypt(CAST('中国' AS VARBINARY), '0123')) |
+-----------------------------------------------------+
| DFB166F0A03113AA848C0CE545D58757                    |
```

AES_DECRYPT

```
aes_decrypt(varbinary x, varchar y)
```

- 命令说明：使用AES算法解密 `x` ， `y` 为密钥。
- 返回值类型：VARBINARY
- 示例：

```
select aes_decrypt(aes_encrypt(CAST('中国' AS VARBINARY), '0123'),'0123');
+-------------------------------------------------------------------+
| aes_decrypt(aes_encrypt(CAST('中国' AS varbinary), '0123'), '0123') |
+-------------------------------------------------------------------+
| 中国                                                              |
```