ALIBABA CLOUD

Alibaba Cloud

AnalyticDB for PostgreSQL

Performance index

Document Version: 20210709

Alibaba Cloud

Legal disclaimer

Document conventions

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

Table of Contents

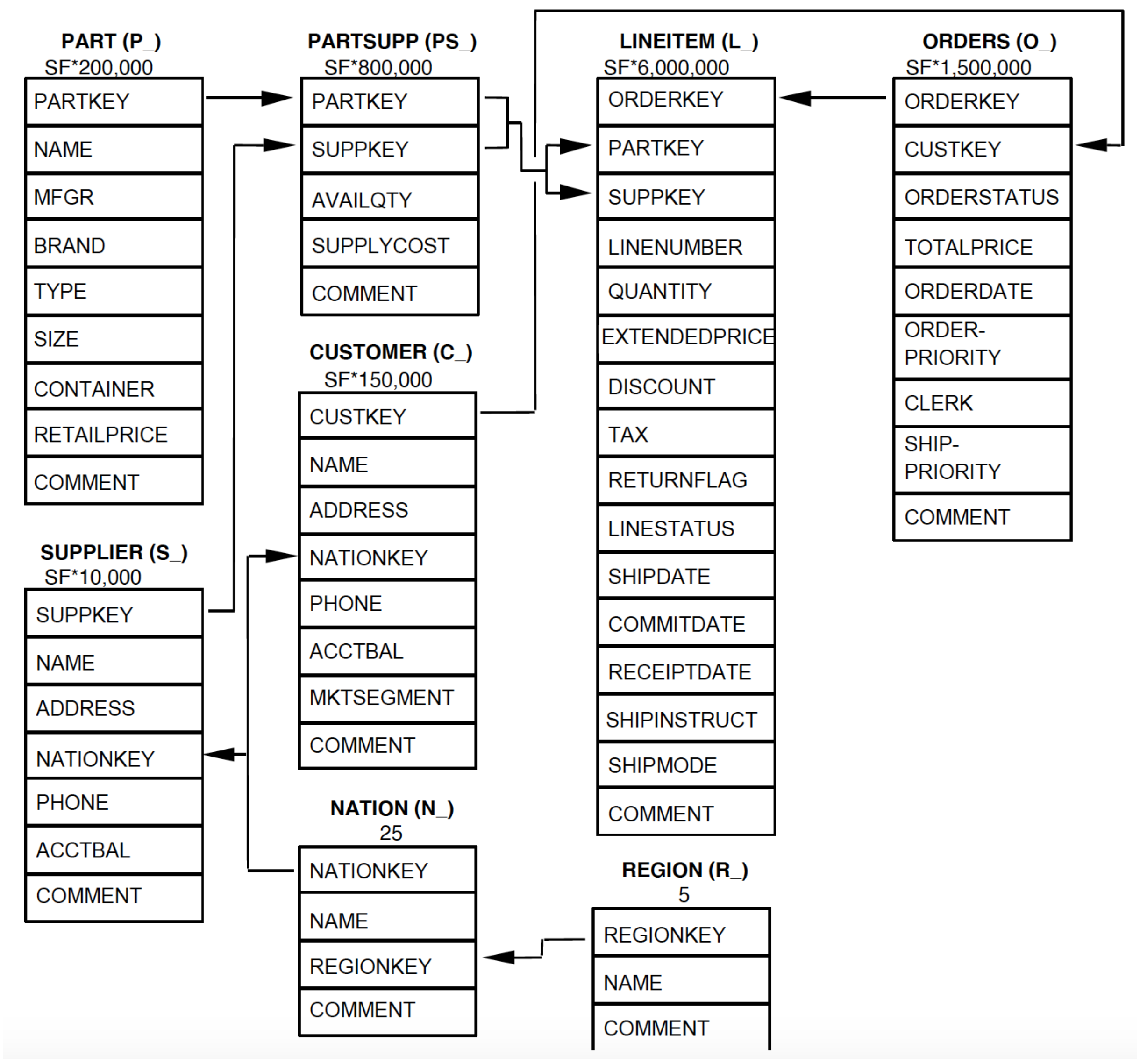
1.TPC-H

This topic describes how to test the transaction processing capabilities of an AnalyticDB for PostgreSQL V6.0 instance by using TPC Benchmark H (TPC-H). The transaction processing capabilities of AnalyticDB for PostgreSQL V6.0 are greatly improved over AnalyticDB for PostgreSQL V4.3.

Introduction

TPC-H is provided by the Transaction Processing Performance Council (TPC) to test decision support systems. TPC-H is used in academia and industries to evaluate the performance of decision support applications. TPC-H models data in production environments to simulate the data warehouse of a sales system. The data warehouse contains eight relationships and the data volume ranges from 1 GB to 3 TB. The benchmark includes 22 queries to evaluate the system response time for each query. The response time is the time between the query submission and the result return. The test result shows the query processing capability of the system. For more information, see TPC-H specifications.

Logical relationships of eight tables



Data volume

The volume of data affects the query speed. In TPC-H, scale factor (SF) is used to describe the data volume. One SF is equal to 1 GB, and 1,000 SF is equal to 1 TB. The eight tables contain 1 SF of data in total, excluding the space occupied by indexes. You must reserve more than 1 SF of data space.

Test specifications

Select one of the following AnalyticDB for PostgreSQL V6.0 specifications for the test instance:

Standard SSDs or enhanced SSDs, four cores per node, and 32 nodes per instance.

Use standard SQL queries provided by TPC-H.

Test procedure

1. **Create an ECS instance.**

   Create an ECS instance to generate 1 TB of data, upload data to the database, and test the client. We recommend that you create an ECS instance of the ecs.g6.4xlarge instance type that uses a 2 TB enhanced SSD and runs CentOS.

2. **Create an AnalyticDB for PostgreSQL V6.0 instance.**

   The instance must be in the same region, zone, and VPC as the ECS instance.

3. **Generate 1 TB of test data for TPC-H.**

   - Log on to the ECS instance by using an SSH key pair and download the TPC-H dbgen program. An executable program dbgen/qgen is generated in the dbgen directory after compilation.

     ```
     git clone https://github.com/gregrahn/tpch-kit.git
     cd tpch-kit/dbgen
     make
     ```

   - **Generate 1 TB of data and run the following command.**

     ```
     ./dbgen --help
     ```

   - Run the following command to view how to generate data:

     ```
     ./dbgen -vf -s 1000
     ```

   - Execute the following shell script to concurrently generate a dataset composed of 10 data files:

     ```
     for((i=1;i<=10;i++));
     do
       ./dbgen -s 1000 -S $i -C 10 -f &
     done
     ```

   - A vertical bar (|) is added at the end of each line in the generated TBL file. Execute the following shell script to delete the vertical bars by using the sed command:

```
sed -i 's/. $//' ./region.tbl &
sed -i 's/. $//' ./nation.tbl &
for((i=1;<=10;i++));
do
  sed -i 's/. $//' ./lineitem.tbl.$i &
  sed -i 's/. $//' ./orders.tbl.$i &
  sed -i 's/. $//' ./customer.tbl.$i &
  sed -i 's/. $//' ./partsupp.tbl.$i &
  sed -i 's/. $//' ./part.tbl.$i &
  sed -i 's/. $//' ./supplier.tbl.$i &
done
```

Create tables

Column store tables are suitable for vector computing and the Just-in-time (JIT) compilation, and can access data and collect statistics more efficiently. You can use the table creation statements to perform the following operations:

- Create append-optimized (AO) column store tables.
- Disable data compression.
- Set tables to replicated tables.

```
create table nation (
  n_nationkey  integer not null,
  n_name       char(25) not null,
  n_regionkey  integer not null,
  n_comment    varchar(152))
with (appendonly=true, orientation=column)
distributed REPLICATED;

create table region (
  r_regionkey  integer not null,
  r_name       char(25) not null,
  r_comment    varchar(152))
with (appendonly=true, orientation=column)
distributed REPLICATED;

create table part (
  p_partkey     integer not null,
  p_name        varchar(55) not null,
  p_mfgr        char(25) not null,
  p_brand       char(10) not null,
  p_type        varchar(25) not null,
  p_size        integer not null,
  p_container   char(10) not null,
  p_retailprice DECIMAL(15,2) not null,
  p_comment     varchar(23) not null)
with (appendonly=true, orientation=column)
distributed by (p_partkey);

create table supplier (
  s_suppkey     integer not null,
  s_name        char(25) not null,
  s_address    varchar(40) not null,
  s_nationkey  integer not null,
  s_phone      char(15) not null,
  s_acctbal    DECIMAL(15,2) not null,
  s_comment    varchar(101) not null)
with (appendonly=true, orientation=column)
distributed by (s_suppkey);

create table partsupp (
  ps_partkey    integer not null,
  ps_suppkey    integer not null,
  ps_availqty   integer not null,
  ps_supplycost  DECIMAL(15,2)  not null,
  ps_comment    varchar(199) not null)
with (appendonly=true, orientation=column)
distributed by (ps_partkey);

create table customer (
  c_custkey    integer not null,
  c_name       varchar(25) not null,
  c_address    varchar(40) not null,
  c_nationkey  integer not null,
  c_phone      char(15) not null,
  c_acctbal    DECIMAL(15,2)  not null,
  c_mktsegment  char(10) not null,
  c_comment    varchar(117) not null)
with (appendonly=true, orientation=column)
distributed by (c_custkey);

create table orders (
  o_orderkey      bigint not null,
  o_custkey       integer not null,
  o_orderstatus   char(1) not null,
  o_totalprice    DECIMAL(15,2) not null,
  o_orderdate     date not null,
  o_orderpriority  char(15) not null,
  o_clerk        char(15) not null,
  o_shippriority  integer not null,
  o_comment       varchar(79) not null)
with (appendonly=true, orientation=column)
distributed by (o_orderkey);

create table lineitem (
  l_orderkey   bigint not null,
  l_partkey    integer not null,
  l_suppkey    integer not null,
  l_linenumber  integer not null,
  l_quantity   DECIMAL(15,2) not null,
  l_extendedprice  DECIMAL(15,2) not null,
  l_discount   DECIMAL(15,2) not null,
  l_tax       DECIMAL(15,2) not null,
  l_returnflag  char(1) not null,
  l_linestatus  char(1) not null,
  l_shipdate   date not null,
  l_commitdate date not null,
  l_commitdate  date not null,
  l_receiptdate date not null,
  l_shipinstruct char(25) not null,
  l_shipmode    char(10) not null,
  l_comment     varchar(44) not null)
with (appendonly=true, orientation=column)
distributed by (l_orderkey);
```

Import data

You can use one of the following methods to import data:

- Execute the COPY statements.
- Use OSS external tables.

The following sections describe the details of the methods.

Execute the COPY statements to import data

Execute the following SQL script:

```
\copy nation from '/data/tpch_1t/nation.tbl' DELIMITER '|';
\copy region from '/data/tpch_1t/region.tbl' DELIMITER '|';
\copy supplier from '/data/tpch_1t/supplier.tbl' DELIMITER '|';
\copy part from '/data/tpch_1t/part.tbl' DELIMITER '|';
\copy partsupp from '/data/tpch_1t/partsupp.tbl' DELIMITER '|';
\copy customer from '/data/tpch_1t/customer.tbl' DELIMITER '|';
\copy orders from '/data/tpch_1t/orders.tbl' DELIMITER '|';
\copy lineitem from '/data/tpch_1t/lineitem.tbl' DELIMITER '|';
```

Replace the example path of the TBL file with the actual path. For more information about the shell script, see the shell script of table creation. You can also use psql to connect to the database and execute the SQL script. To improve the import efficiency that is allowed by the network bandwidth of the ECS instance, you can use multiple psql connections to concurrently execute COPY statements.

Use external tables to import data

Upload the generated data file to OSS.

```
./ossutil64 cp -r <TBL file directory> oss://<oss bucket>/<directory>/
            -i <AccessKey ID> -k <Access Key Secret>
            -e <EndPoint>
```

For more information, see Import or export OSS data by using OSS external tables.

Create OSS external tables

```
create readable external table ext_nation ( n_nationkey int, n_name varchar(25), n_regionkey integer,
  n_comment varchar(152))
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/nation.tbl
  id=$AccessKey key=$AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;
```

```
create readable external table ext_region ( R_REGIONKEY int, R_NAME CHAR(25),R_COMMENT VARCHAR(152))

  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/region.tbl
  id=$AccessKey key=$AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;

CREATE readable external TABLE ext_lineitem ( l_orderkey bigint, l_partkey bigint, l_suppkey bigint,
 l_linenumber bigint, l_quantity double precision, l_extendedprice double precision,
 l_discount double precision, l_tax double precision, l_returnflag CHAR(1),
 l_linestatus CHAR(1), l_shipdate DATE, l_commitdate DATE, l_receiptdate DATE,
 l_shipinstruct CHAR(25), l_shipmode CHAR(10), l_comment VARCHAR(44))
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/lineitem.tbl
  id= $AccessKey key= $AccessKeySecret
  bucket=oss-y ') FORMAT 'TEXT' (DELIMITER '|' ) ;

CREATE readable external TABLE ext_orders ( o_orderkey bigint , o_custkey bigint , o_orderstatus CHAR(1) ,
 o_totalprice double precision, o_orderdate DATE , o_orderpriority CHAR(15) , o_clerk CHAR(15) ,
 o_shippriority bigint , o_comment VARCHAR(79) )
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/orders.tbl
  id=$AccessKey key=$AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;

CREATE readable external TABLE ext_part ( p_partkey bigint , p_name VARCHAR(55) , p_mfgr CHAR(25) ,
 p_brand CHAR(10) , p_type VARCHAR(25) , p_size bigint , p_container CHAR(10) ,
 p_retailprice double precision , p_comment VARCHAR(23) )
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/part.tbl
  id= $AccessKey key= $AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;

CREATE readable external TABLE ext_partsupp ( ps_partkey bigint , ps_suppkey bigint ,
 ps_availqty bigint , ps_supplycost double precision , ps_comment VARCHAR(199) )
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/partsupp.tbl
  id= $AccessKey key= $AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;

CREATE readable external TABLE ext_supplier ( s_suppkey bigint , s_name CHAR(25) ,
 s_address VARCHAR(40) , s_nationkey bigint , s_phone CHAR(15) , s_acctbal DECIMAL(15,2) ,
 s_comment VARCHAR(101) )
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/supplier.tbl
  id= $AccessKey key= $AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;

CREATE readable external TABLE ext_customer ( c_custkey bigint , c_name VARCHAR(25) ,
 c_address VARCHAR(40) , c_nationkey bigint , c_phone CHAR(15) , c_acctbal double precision ,
 c_mktsegment CHAR(10) , c_comment VARCHAR(117) )
  location('oss://oss-cn-beijing.aliyuncs.com
  filepath=data/tpch_data_1000x/customer.tbl
  id= $AccessKey key= $AccessKeySecret
  id=$AccessKey key=$AccessKeySecret
  bucket=oss-y') FORMAT 'TEXT' (DELIMITER '|' ) ;
```

Write TPC-H data from OSS external tables to the AnalyticDB for PostgreSQL instance

```
insert into nation select * from ext_nation;
insert into region select * from ext_region;
insert into lineitem select * from ext_lineitem;
insert into orders select * from ext_orders;
insert into customer select * from ext_customer;
insert into part select * from ext_part;
insert into partsupp select * from ext_partsupp;
insert into supplier select * from ext_supplier;
```

Data is imported. Perform the following steps to execute queries.

Collect table statistics

```
analyze nation;
analyze region;
analyze lineitem;
analyze orders;
analyze customer;
analyze part;
analyze partsupp;
analyze supplier;
```

Execute queries

Execute the following shell script to start the test. You can also use clients such as psql to execute SQL queries one by one. The 22 SQL queries are listed in the lower part of this topic.

Accelerate queries

The vector computing acceleration engine for AnalyticDB for PostgreSQL V6.0, Odyssey, can double query performance in TPC-H scenarios.

Usage:

Set enable_odyssey to on at the session level to enable Odyssey. Execute the following SQL statement:

```
set enable_odyssey = on;
```

Set enable_odyssey to off to disable Odyssey.

```
set enable_odyssey = off;
```

If you execute the following script to execute the 22 SQL queries, you must add `set enable_odyssey = on;` at the beginning of each query.

Execute all queries and record the time consumed by each query and the overall time consumed

```
total_cost=0

for i in {1..22}
do
     echo "begin run Q${i}, query/q$i.sql , `date` "
     begin_time=`date +%s. %N`
     #psql -h ${instance endpoint} -p ${port} -U ${database user} -f query/q${i}.sql > ./log/log_q${i}.out
     rc=$?
     end_time=`date +%s. %N`
     cost=`echo "$end_time-$begin_time"|bc`
     total_cost=`echo "$total_cost+$cost"|bc`
     if [ $rc -ne 0 ] ; then
         printf "run Q%s fail, cost: %.2f, totalCost: %.2f, `date`\n" $i $cost $total_cost
      else
         printf "run Q%s succ, cost: %.2f, totalCost: %.2f, `date`\n" $i $cost $total_cost
      fi
done
```

Test results

The following table describes the number of data entries in each table. The total amount of data is 1 TB, excluding indexes.

Table name	Data entries
customer	150,000,000
lineitem	5,999,989,709
nation	25
orders	1,500,000,000
part	200,000,000
partsupp	800,000,000
region	5
supplier	10,000,000

The following table describes the execution duration.

Total execution duration (Unit: seconds)	4-core CPU, 32 nodes, standard SSD or enhanced SSD	4-core CPU, 32 nodes, standard SSD or enhanced SSD, Odyssey enabled
Total	**2179.85**	**1258.24**
Q1	399.38	171.05
Q2	25.32	12.24
Q3	56.91	38.26
Q4	54.26	20.20
Q5	145.64	118.72
Q6	30.61	21.19
Q7	71.43	63.79
Q8	73.58	37.84
Q9	174.09	169.28
Q10	51.56	36.96
Q11	11.63	4.56
Q12	44.25	27.74
Q13	59.13	40.00
Q14	27.90	15.18
Q15	48.62	26.27

Total execution duration (Unit: seconds)	4-core CPU, 32 nodes, standard SSD or enhanced SSD	4-core CPU, 32 nodes, standard SSD or enhanced SSD, Odyssey enabled
Q16	19.15	13.02
Q17	294.83	178.73
Q18	293.15	98.39
Q19	41.84	48.15
Q20	61.87	32.22
Q21	151.44	58.85
Q22	43.26	25.60

22 SQL queries

```
-- Q1
-- Enable Odyssey.
set enable_odyssey = on;
select
   l_returnflag,
   l_linestatus,
   sum(l_quantity) as sum_qty,
   sum(l_extendedprice) as sum_base_price,
   sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
   sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
   avg(l_quantity) as avg_qty,
   avg(l_extendedprice) as avg_price,
   avg(l_discount) as avg_disc,
   count(*) as count_order
from
   lineitem
where
   l_shipdate <= date '1998-12-01' - interval '93 day'
group by
   l_returnflag,
   l_linestatus
order by
   l_returnflag,
   l_linestatus;

-- Q2
-- Enable Odyssey.
set enable_odyssey = on;
select
  s_acctbal,
  s_name,
  n_name,
  p_partkey,
  p_mfgr,
  s_address,
  s_phone,
  s_comment
from
  part,
  supplier,
  partsupp,
  nation,
  region
where
  p_partkey = ps_partkey
  and s_suppkey = ps_suppkey
  and p_size = 23
  and p_type like '%STEEL'
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'EUROPE'
  and ps_supplycost = (
    select
      min(ps_supplycost)
    from
      partsupp,
      supplier,
      nation,
      region
    where
      p_partkey = ps_partkey
      and s_suppkey = ps_suppkey
      and s_nationkey = n_nationkey
      and n_regionkey = r_regionkey
      and r_name = 'EUROPE'
  )
order by
  s_acctbal desc,
  n_name,
  s_name,
  p_partkey
limit 100;

-- Q3
-- Enable Odyssey.
set enable_odyssey = on;
select
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  o_orderdate,
  o_shippriority
from
  customer,
  orders,
  lineitem
where
  c_mktsegment = 'MACHINERY'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < date '1995-03-24'
  and l_shipdate > date '1995-03-24'
group by
  l_orderkey,
  o_orderdate,
  o_shippriority
order by
  revenue desc,
  o_orderdate
limit 10;

-- Q4
-- Enable Odyssey.
set enable_odyssey = on;
select
  o_orderpriority,
  count(*) as order_count
from
  orders
where
  o_orderdate >= date '1996-08-01'
  and o_orderdate < date '1996-08-01' + interval '3' month
  and exists (
    select
      *
    from
      lineitem
    where
      l_orderkey = o_orderkey
      and l_commitdate < l_receiptdate
  )
group by
  o_orderpriority
order by
  o_orderpriority;

-- Q5
-- Enable Odyssey.
set enable_odyssey = on;
select
  n_name,
  sum(l_extendedprice * (1 - l_discount)) as revenue
from
  customer,
  orders,
  lineitem,
  supplier,
  nation,
  region
where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and l_suppkey = s_suppkey
  and c_nationkey = s_nationkey
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'MIDDLE EAST'
  and o_orderdate >= date '1994-01-01'
  and o_orderdate < date '1994-01-01' + interval '1' year
group by
  n_name
order by
  revenue desc;

-- Q6
-- Enable Odyssey.
set enable_odyssey = on;
select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate >= date '1994-01-01'
  and l_shipdate < date '1994-01-01' + interval '1' year
  and l_discount between 0.06 - 0.01 and 0.06 + 0.01
  and l_quantity < 24;

-- Q7
-- Enable Odyssey.
set enable_odyssey = on;
select
  supp_nation,
  cust_nation,
  l_year,
  sum(volume) as revenue
from
  (
    select
      n1.n_name as supp_nation,
      n2.n_name as cust_nation,
      extract(year from l_shipdate) as l_year,
      l_extendedprice * (1 - l_discount) as volume
    from
      supplier,
      lineitem,
      orders,
      customer,
      nation n1,
      nation n2
    where
      s_suppkey = l_suppkey
      and o_orderkey = l_orderkey
      and c_custkey = o_custkey
      and s_nationkey = n1.n_nationkey
      and c_nationkey = n2.n_nationkey
      and (
        (n1.n_name = 'JORDAN' and n2.n_name = 'INDONESIA')
        or (n1.n_name = 'INDONESIA' and n2.n_name = 'JORDAN')
      )
      and l_shipdate between date '1995-01-01' and date '1996-12-31'
  ) as shipping
group by
  supp_nation,
  cust_nation,
  l_year
order by
  supp_nation,
  cust_nation,
  l_year;

-- Q8
-- Enable Odyssey.
set enable_odyssey = on;
select
  o_year,
  sum(case
    when nation = 'INDONESIA' then volume
    else 0
  end) / sum(volume) as mkt_share
from
  (
    select
      extract(year from o_orderdate) as o_year,
      l_extendedprice * (1 - l_discount) as volume,
      n2.n_name as nation
    from
      part,
      supplier,
      lineitem,
      orders,
      customer,
      nation n1,
      nation n2,
      region
    where
      p_partkey = l_partkey
      and s_suppkey = l_suppkey
      and l_orderkey = o_orderkey
      and o_custkey = c_custkey
      and c_nationkey = n1.n_nationkey
      and n1.n_regionkey = r_regionkey
      and r_name = 'ASIA'
      and s_nationkey = n2.n_nationkey
      and o_orderdate between date '1995-01-01' and date '1996-12-31'
```

```
        and o_orderdate between date '1995-01-01' and date '1996-12-31'
        and p_type = 'STANDARD BRUSHED BRASS'
  ) as all_nations
group by
  o_year
order by
  o_year;

-- Q9
-- Enable Odyssey.
set enable_odyssey = on;
select
  nation,
  o_year,
  sum(amount) as sum_profit
from
  (
    select
      n_name as nation,
      extract(year from o_orderdate) as o_year,
      l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
    from
      part,
      supplier,
      lineitem,
      partsupp,
      orders,
      nation
    where
      s_suppkey = l_suppkey
      and ps_suppkey = l_suppkey
      and ps_partkey = l_partkey
      and p_partkey = l_partkey
      and o_orderkey = l_orderkey
      and s_nationkey = n_nationkey
      and p_name like '%chartreuse%'
  ) as profit
group by
  nation,
  o_year
order by
  nation,
  o_year desc;

-- Q10
-- Enable Odyssey.
set enable_odyssey = on;
select
  c_custkey,
  c_name,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  c_acctbal,
  n_name,
  c_address,
  c_phone
```

```
   c_comment
from
   customer,
   orders,
   lineitem,
   nation
where
   c_custkey = o_custkey
   and l_orderkey = o_orderkey
   and o_orderdate >= date '1994-08-01'
   and o_orderdate < date '1994-08-01' + interval '3' month
   and l_returnflag = 'R'
   and c_nationkey = n_nationkey
group by
   c_custkey,
   c_name,
   c_acctbal,
   c_phone,
   n_name,
   c_address,
   c_comment
order by
   revenue desc
limit 20;

-- Q11
-- Enable Odyssey.
set enable_odyssey = on;
select
   ps_partkey,
   sum(ps_supplycost * ps_availqty) as value
from
   partsupp,
   supplier,
   nation
where
   ps_suppkey = s_suppkey
   and s_nationkey = n_nationkey
   and n_name = 'INDONESIA'
group by
   ps_partkey having
      sum(ps_supplycost * ps_availqty) > (
         select
            sum(ps_supplycost * ps_availqty) * 0.0001000000
         from
            partsupp,
            supplier,
            nation
         where
            ps_suppkey = s_suppkey
            and s_nationkey = n_nationkey
            and n_name = 'INDONESIA'
      )
order by
```

```
  value desc;

-- Q12
-- Enable Odyssey.
set enable_odyssey = on;
select
  l_shipmode,
  sum(case
    when o_orderpriority = '1-URGENT'
      or o_orderpriority = '2-HIGH'
      then 1
    else 0
  end) as high_line_count,
  sum(case
    when o_orderpriority <> '1-URGENT'
      and o_orderpriority <> '2-HIGH'
      then 1
    else 0
  end) as low_line_count
from
  orders,
  lineitem
where
  o_orderkey = l_orderkey
  and l_shipmode in ('REG AIR', 'TRUCK')
  and l_commitdate < l_receiptdate
  and l_shipdate < l_commitdate
  and l_receiptdate >= date '1994-01-01'
  and l_receiptdate < date '1994-01-01' + interval '1' year
group by
  l_shipmode
order by
  l_shipmode;

-- Q13
-- Enable Odyssey.
set enable_odyssey = on;
select
  c_count,
  count(*) as custdist
from
  (
    select
      c_custkey,
      count(o_orderkey)
    from
      customer left outer join orders on
        c_custkey = o_custkey
        and o_comment not like '%pending%requests%'
    group by
      c_custkey
  ) as c_orders (c_custkey, c_count)
group by
  c_count
```

```
order by
  custdist desc,
  c_count desc;

-- Q14
-- Enable Odyssey.
set enable_odyssey = on;
select
  100.00 * sum(case
    when p_type like 'PROMO%'
      then l_extendedprice * (1 - l_discount)
    else 0
  end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
  lineitem,
  part
where
  l_partkey = p_partkey
  and l_shipdate >= date '1994-11-01'
  and l_shipdate < date '1994-11-01' + interval '1' month;

-- Q15
-- Enable Odyssey.
set enable_odyssey = on;
create view revenue0 (supplier_no, total_revenue) as
  select
    l_suppkey,
    sum(l_extendedprice * (1 - l_discount))
  from
    lineitem
  where
    l_shipdate >= date '1997-10-01'
    and l_shipdate < date '1997-10-01' + interval '3' month
  group by
    l_suppkey;
select
  s_suppkey,
  s_name,
  s_address,
  s_phone,
  total_revenue
from
  supplier,
  revenue0
where
  s_suppkey = supplier_no
  and total_revenue = (
    select
      max(total_revenue)
    from
      revenue0
  )
order by
  s_suppkey;
```

```
drop view revenue0;

-- Q16
-- Enable Odyssey.
set enable_odyssey = on;
select
   p_brand,
   p_type,
   p_size,
   count(distinct ps_suppkey) as supplier_cnt
from
   partsupp,
   part
where
   p_partkey = ps_partkey
   and p_brand <> 'Brand#44'
   and p_type not like 'SMALL BURNISHED%'
   and p_size in (36, 27, 34, 45, 11, 6, 25, 16)
   and ps_suppkey not in (
      select
         s_suppkey
      from
         supplier
      where
         s_comment like '%Customer%Complaints%'
   )
group by
   p_brand,
   p_type,
   p_size
order by
   supplier_cnt desc,
   p_brand,
   p_type,
   p_size;

-- Q17
-- Enable Odyssey.
set enable_odyssey = on;
select
   sum(l_extendedprice) / 7.0 as avg_yearly
from
   lineitem,
   part
where
   p_partkey = l_partkey
   and p_brand = 'Brand#42'
   and p_container = 'JUMBO PACK'
   and l_quantity < (
      select
         0.2 * avg(l_quantity)
      from
         lineitem
      where
```

```
        l_partkey = p_partkey
    );

-- Q18
-- Enable Odyssey.
set enable_odyssey = on;
select
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 312
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate
limit 100;

-- Q19
-- Enable Odyssey.
set enable_odyssey = on;
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#43'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 5 and l_quantity <= 5 + 10
        and p_size between 1 and 5
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 12 and l_quantity <= 12 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#11'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 24 and l_quantity <= 24 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );

-- Q20
-- Enable Odyssey.
set enable_odyssey = on;
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'magenta%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
```

```
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= date '1996-01-01'
                    and l_shipdate < date '1996-01-01' + interval '1' year
            )
    )
    and s_nationkey = n_nationkey
    and n_name = 'RUSSIA'
order by
    s_name;

-- Q21
-- Enable Odyssey.
set enable_odyssey = on;
select
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3.l_orderkey = l1.l_orderkey
            and l3.l_suppkey <> l1.l_suppkey
            and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'MOZAMBIQUE'
group by
    s_name
order by
    numwait desc,
    s_name
limit 100;
```

```
-- Q22
-- Enable Odyssey.
set enable_odyssey = on;
select
    cntrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone from 1 for 2) as cntrycode,
            c_acctbal
        from
            customer
        where
            substring(c_phone from 1 for 2) in
                ('13', '31', '23', '29', '30', '18', '17')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
                    and substring(c_phone from 1 for 2) in
                        ('13', '31', '23', '29', '30', '18', '17')
            )
            and not exists (
                select
                    *
                from
                    orders
                where
                    o_custkey = c_custkey
            )
    ) as custsale
group by
    cntrycode
order by
    cntrycode;
```