ALIBABA CLOUD

阿里云

# 云原生数据仓库AnalyticDB MySQL版 性能白皮书

文档版本：20210118

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<br>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<br>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<br>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<br>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`<br>*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.概述

本次性能测试基于阿里云基础环境，分别在同等（或接近）硬件配置和同等数据规模下，对比AnalyticDB MySQL与Presto、Spark基于标准TPC-H的测试结果，以及在不同应用场景下的不同测试结果。

TPC-H由TPC委员会制定发布，用于评测数据库的分析查询能力。TPC-H查询包含八张数据表和二十二条复杂SQL查询，大多数查询包含多表Join、子查询和Group By等。更多信息可参考TPC-H测试集。

TPC-DS由TPC委员会制定发布，用于决策支持系统测试基准，主要用于衡量大数据产品的分析性能。TPC-DS查询共包含99个查询测试语句，更多信息可参考TPC-DS测试集。

云原生数据仓库AnalyticDB MySQL是云端托管的PB级高并发实时数据仓库，专注于服务OLAP领域。采用关系模型进行数据存储，可以使用SQL进行自由灵活的计算分析，无需预先建模。利用云端的无缝伸缩能力，AnalyticDB MySQL在处理百亿条甚至更多量级的数据时真正实现毫秒级计算。

AnalyticDB MySQL提供了按量付费和包年包月两种付费方式，并支持灵活的扩缩容。其中：

- 按量付费集群支持随时进行规格的扩缩容。
- 包年包月集群支持随时扩缩容和续费。

# 2.基于标准测试集的测试说明

## 2.1. 测试环境

下表列出了本次性能测试所使用的环境信息。

> **说明** 测试基础环境均在阿里云申请。

| 产品 | 规格 | 架构 | CPU | MEM | 存储 | 版本 |
| --- | --- | --- | --- | --- | --- | --- |
| AnalyticDB MySQL | analyticdb.e32.xlarge.n1 | • master：1个节点<br>• core：6个节点 | • master：24核<br>• core：48核（每节点8核） | • master：96G<br>• core：384GB（每节点64G） | • master：ESSD云盘，2048GB<br>• worker：ESSD云盘，500GB | AnalyticDB MySQL 3.1.3.4 |
| Presto | • master：ecs.g6.6xlarge<br>• worker：ecs.se1ne.2xlarge | • master：1个节点<br>• worker：6个节点 | • master：24核<br>• worker：48 核（每节点8核） | • master：96GB<br>• worker：384GB（每节点64G） | • master：SSD云盘，120GB<br>• worker：SSD云盘，80GB X 4块 | • HDFS：3.1.3<br>• YARN：3.1.3<br>• Presto：3.3.1 |
| Spark | • master：ecs.g5.6xlarge<br>• worker：ecs.se1ne.2xlarge | • master：1个节点<br>• worker：6个节点 | • master：24核<br>• worker：48 核（每节点8核） | • master：96GB<br>• worker：384GB（每节点64G） | • master：SSD云盘，120GB<br>• worker：SSD云盘，120GB X 1块 | • HDFS：3.1.3<br>• YARN：3.1.3<br>• Presto：3.3.1<br>• Spark：2.4.5 |

## 2.2. 测试过程

本文介绍性能测试包含的主要步骤。

1. 构建数据结构
   - AnalyticDB MySQL
   - Presto
   - Spark
2. 数据初始化
   i. 构建数据

ii. 导入数据

3. TPC-H测试集

有关测试过程的更多信息可参见附录：测试过程详解。

# 2.3. 测试方法

本次测试使用自定义脚本，依次执行TPC-H测试集。

## 测试优化

为保证顺利完成测试，在对个产品进行测试之前，进行了必要的配置优化。

- AnalyticDB MySQL

  未进行配置优化。

- Presto

  部分执行参数做了以下调整。

```
task.concurrency=128
exchange.max-reponse-size=16MB
query.max-memory-per-node=2GB
task.max-partial-aggregation-memory=32MB
exchange.max-buffer-size=128MB
query.max-total-memory-per-node=4GB
sink.max-buffer-size=64MB
```

- Spark

  部分执行参数做了以下调整。

```
spark-sql --num-executors 12
--executor-cores 3
--executor-memory 16G
```

## 测试方法

> 说明
> - 测试前，均执行两轮测试集作为预热环节。
> - 连续执行三轮测试，取三次测试结果的平均值。

编写测试脚本。

```
while [ $n -lt 23 ]
do
 echo "query ${n} starting"
 time mysql -f tpch <tpch_${n}.sql
 echo "query $n ended!"
 n=`expr ${n} + 1`
done
```

# 2.4. 测试指标

测试指标包括整个TPC-H测试集和单条SQL的执行时长。

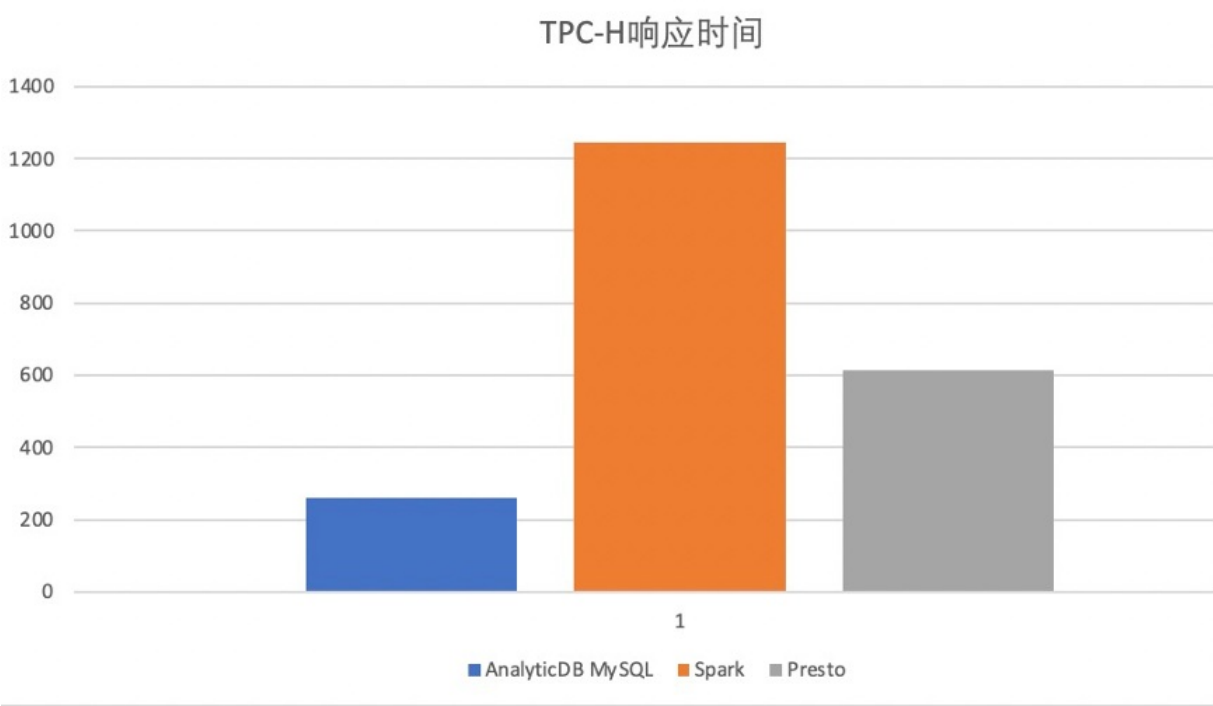
# 2.5. 测试结果

本文介绍AnalyticDB MySQL、Presto、Spark的性能测试结果。

查询执行时间以秒(s)为单位。

| 测试样本 | AnalyticDB MySQL | Presto | 执行时间增加倍数 | Spark | 执行时间增加倍数 |
|---|---|---|---|---|---|
| tpch_sql1 | 24.1 | 36.28 | 0.505394191 | 62.97 | 1.612863071 |
| tpch_sql2 | 1.3 | 12.25 | 8.423076923 | 36.03 | 26.71538462 |
| tpch_sql3 | 7.1 | 19.8 | 1.788732394 | 39.12 | 4.509859155 |
| tpch_sql4 | 13.7 | 13.21 | -0.035766423 | 24.14 | 0.762043796 |
| tpch_sql5 | 16.3 | 29.9 | 0.834355828 | 116.27 | 6.133128834 |
| tpch_sql6 | 0.7 | 11.61 | 15.58571429 | 10.6 | 14.14285714 |
| tpch_sql7 | 6.6 | 57.75 | 7.75 | 99.97 | 14.1469697 |
| tpch_sql8 | 16.2 | 31.06 | 0.917283951 | 78.8 | 3.864197531 |
| tpch_sql9 | 34 | 48.15 | 0.416176471 | 144.65 | 3.254411765 |
| tpch_sql10 | 5.5 | 23.25 | 3.227272727 | 52.49 | 8.543636364 |
| tpch_sql11 | 2.2 | 11.11 | 4.05 | 36.51 | 15.59545455 |
| tpch_sql12 | 3.5 | 14.79 | 3.225714286 | 21.64 | 5.182857143 |
| tpch_sql13 | 11.3 | 15.07 | 0.333628319 | 37.1 | 2.283185841 |
| tpch_sql14 | 1.3 | 10.62 | 7.169230769 | 19.69 | 14.14615385 |
| tpch_sql15 | 2.9 | 21.68 | 6.475862069 | 23.77 | 7.196551724 |

| 测试样本 | AnalyticDB MySQL | Presto | 执行时间增加倍数 | Spark | 执行时间增加倍数 |
|---|---|---|---|---|---|
| tpch_sql16 | 2.3 | 6.55 | 1.847826087 | 80.7 | 34.08695652 |
| tpch_sql17 | 5.4 | 55.85 | 9.342592593 | 116.86 | 20.64074074 |
| tpch_sql18 | 67.5 | 58.41 | -0.134666667 | 89.67 | 0.328444444 |
| tpch_sql19 | 3 | 19.92 | 5.64 | 21.89 | 6.296666667 |
| tpch_sql20 | 9.4 | 22.13 | 1.354255319 | 37.59 | 2.99893617 |
| tpch_sql21 | 22.5 | 88.81 | 2.947111111 | 68.42 | 2.040888889 |
| tpch_sql22 | 4 | 6.4 | 0.6 | 25.05 | 5.2625 |
| 总时长（s） | 260.8 | 614.6 | 1.356595092 | 1243.93 | 3.769670245 |



# 3.基于应用场景的测试说明

## 3.1. 查询场景

### 场景：在线报表查询

***业务背景***

在线报表场景中，面向单表或者多表关联后的TopN，谓词过滤以及聚合计算是典型的分析计算需求。本示例通过一个面向零售行业的决策分析系统来展示AnalyticDB MySQL的性能。数据模型是基于TPC-H实现的，更多信息可参见TPC-H测试集。

***测试环境***

- AnalyticDB MySQL，弹性模式集群版
- 开源ClickHouse 32核，内核版本 20.3

***测试结果***

1、统计指定时间内，有成交记录的活跃用户数。

```
select
 count(*) //聚合: 统计
from
 (
  select
   c_name,
   c_phone,
   o_totalprice
  from
   customer,
   orders
  where
   c_custkey = o_custkey
  and o_orderdate < date '1993-09-23' // 谓词过滤：指定时间
  and o_orderdate > date '1993-03-23'
 ) a;
```

测试结果如下：

| 产品 | 查询耗时 |
| --- | --- |
| AnalyticDB MySQL | 0.7s |
| 开源ClickHouse | 2.5s |

2、统计未派送订单明细，按照优先级（比如，订单客单价）排序。该报表数据可以用来跟踪订单派送的服务状态。

```
select
  l_orderkey,
  sum(l_extendedprice*(1-l_discount)) as revenue, //聚合：潜在的收入
  o_orderdate,
  o_shippriority
from
  customer, orders, lineitem
where
  c_mktsegment = 'MACHINERY'   // 谓词过滤
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < date '1995-03-23'
  and o_orderdate > date '1995-02-23' // 谓词过滤 指定时间
  and l_shipdate > date '1995-03-23'
  and l_shipdate < date '1996-03-23'
group by //分组操作
  l_orderkey, //订单标识
  o_orderdate, //订单日期
  o_shippriority //运输优先级
order by
  revenue desc, //TopN：降序排序，把潜在最大收入列在前面
  o_orderdate;
limit
  100;
```

测试结果如下：

| 产品 | 查询耗时 |
| --- | --- |
| AnalyticDB MySQL | 1.2s |
| 开源ClickHouse | 10.7s |

3、统计某个地区零件供货商收入。该报表数据可以用于决策在给定的区域是否需要建立一个当地分配中心。

```
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue  //聚合：潜在的收入
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region   //六表连接
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 = 'EUROPE'  // 谓词过滤：指定地区
    and o_orderdate >= date '1996-01-01'  //谓词过滤：指定时间
    and o_orderdate < date '1996-01-01' + interval '1' year
    and l_shipdate > date '1996-02-23'
     and l_shipdate < date '1996-03-23'
group by
    n_name
order by   // TopN：按收入降序排序，注意分组和排序子句不同
    revenue desc
limit 10
```

测试结果如下：

| 产品 | 查询耗时 |
| --- | --- |
| AnalyticDB MySQL | 1.3s |
| 开源ClickHouse | 3.6s |

## 场景：交互式分析

***业务背景***

交互式分析指的是业务人员通过BI工具或自助式查询平台进行自由的数据探索性查询场景，查询语句的主要特点是支持多维度组合、支持多表Join、包含聚合查询、包含子查询、包含过滤条件。交互式分析性能测试标准为TPC-H测试集，更多信息可参见TPC-H测试集。

***测试环境***

- AnalyticDB MySQL，弹性模式集群版
- 高性能集群，C8 x 4（96核, 768G）
- 弹性（96Core）集群，6节点 （96核，384G）

***测试结果（单位：秒）***

| TPC-H查询 | 高性能查询耗时 | 弹性（96Core）查询耗时 |
| --- | --- | --- |
| Q1 | 81.59 | 106.43 |
| Q2 | 6.57 | 12.83 |
| Q3 | 47.87 | 44.1 |
| Q4 | 60.13 | 63 |
| Q5 | 45.48 | 59.23 |
| Q6 | 11.33 | 2.05 |
| Q7 | 55.98 | 38.65 |
| Q8 | 56.4 | 77.29 |
| Q9 | 147.62 | 132.96 |
| Q10 | 48.06 | 41.99 |
| Q11 | 11.19 | 15 |
| Q12 | 40.53 | 20.71 |
| Q13 | 59.79 | 62.93 |
| Q14 | 37.67 | 5.33 |
| Q15 | 28.53 | 12.68 |
| Q16 | 10.18 | 9.89 |
| Q17 | 48.63 | 73.22 |
| Q18 | 54.88 | 82.86 |
| Q19 | 32.07 | 60.67 |
| Q20 | 29.53 | 39.87 |
| Q21 | 123.89 | 143.43 |
| Q22 | 17.41 | 21.96 |
| 总计 | 1055.33 | 1127.07 |

| TPC-H查询 | 高性能查询耗时 | 弹性（96Core）查询耗时 |
| --- | --- | --- |

## 场景：数据仓库ETL

***业务背景***

ETL场景指的是数据仓库中进行批量的数据清洗、转换和加工计算，通常扫描数据量比较大、计算逻辑复杂、关联表数量较多，计算时间比较长。ETL场景性能测试标准为TPC-DS测试集，更多信息可参见TPC-DS测试集。

***测试环境***

- AnalyticDB MySQL，弹性模式集群版
- 弹性（192Core）集群，12节点（192核，768G）

***测试结果（单位：毫秒）***

| TPC-DS查询 | 弹性（192Core）查询耗时 |
| --- | --- |
| 1 | 3,389 |
| 2 | 23,697 |
| 3 | 9,162 |
| 4 | 137,346 |
| 5 | 27,759 |
| 6 | 8,957 |
| 7 | 17,755 |
| 8 | 9,952 |
| 9 | 12,759 |
| 10 | 16,026 |
| 11 | 87,537 |
| 12 | 2,036 |
| 13 | 28,970 |
| 14 | 145,158 |
| 15 | 6,865 |
| 16 | 15,636 |
| 17 | 21,550 |

| TPC-DS查询 | 弹性（192Core）查询耗时 |
| --- | --- |
| 18 | 12,039 |
| 19 | 13,438 |
| 20 | 4,007 |
| 21 | 2,513 |
| 22 | 5,733 |
| 23 | 418,050 |
| 24 | 28,407 |
| 25 | 20,617 |
| 26 | 8,835 |
| 27 | 17,002 |
| 28 | 18,918 |
| 29 | 19,380 |
| 30 | 2,016 |
| 31 | 26,526 |
| 32 | 4,541 |
| 33 | 14,992 |
| 34 | 11,627 |
| 35 | 15,882 |
| 36 | 11,007 |
| 37 | 4,589 |
| 38 | 29,910 |
| 39 | 7,924 |
| 40 | 14,371 |
| 41 | 367 |
| 42 | 6,814 |
| 43 | 9,072 |

| TPC-DS查询 | 弹性（192Core）查询耗时 |
| --- | --- |
| 44 | 5,766 |
| 45 | 2,691 |
| 46 | 18,635 |
| 47 | 46,180 |
| 48 | 19,307 |
| 49 | 54,883 |
| 50 | 13,762 |
| 51 | 26,672 |
| 52 | 6,768 |
| 53 | 8,524 |
| 54 | 12,323 |
| 55 | 6,680 |
| 56 | 15,015 |
| 57 | 22,917 |
| 58 | 11,187 |
| 59 | 35,661 |
| 60 | 14,910 |
| 61 | 24,657 |
| 62 | 4,923 |
| 63 | 9,444 |
| 64 | 144,757 |
| 65 | 22,203 |
| 66 | 15,136 |
| 67 | 138,622 |
| 68 | 23,379 |
| 69 | 15,156 |

| TPC-DS查询 | 弹性（192Core）查询耗时 |
| --- | --- |
| 70 | 18,614 |
| 71 | 17,067 |
| 72 | 18,801 |
| 73 | 11,977 |
| 74 | 54,000 |
| 75 | 37,597 |
| 76 | 31,804 |
| 77 | 17,376 |
| 78 | 78,188 |
| 79 | 18,075 |
| 80 | 55,100 |
| 81 | 2,958 |
| 82 | 4,531 |
| 83 | 2,000 |
| 84 | 1,554 |
| 85 | 11,139 |
| 86 | 3,221 |
| 87 | 29,937 |
| 88 | 52,394 |
| 89 | 8,704 |
| 90 | 2,971 |
| 91 | 1,800 |
| 92 | 1,969 |
| 93 | 29,375 |
| 94 | 7,822 |
| 95 | 11,929 |

| TPC-DS查询 | 弹性（192Core）查询耗时 |
| --- | --- |
| 96 | 7,935 |
| 97 | 17,152 |
| 98 | 7,351 |
| 99 | 9,159 |
| 总计 | 2595.79（秒） |

# 3.2. 数据导入场景

## 测试实例规格

| 产品 | 规格 |
| --- | --- |
| AnalyticDB MySQL 3.0 | 弹性模式集群版，1 worker (8core) |
| ElasticSearch 6.7.0 | 通用商业版，1节点（8core） |
| ECS | 2个 ECS，32 vCPU 128 GiB，NVMe SSD本地盘存储：3576 GiB。<br>**说明** ECS与AnalyticDB MySQL、ElasticSearch同地域可用区，带宽充足。 |

## 测试方法

- AnalyticDB MySQL实时写入方法：在ECS上，使用Java程序，读取本地TPC-H的多个分片文件，基于JDBC多线程导入，导入批次2000条一批。其中导入SQL为 `insert into lineitem values (...)` 。
- ElasticSearch实时写入方法：在ECS上，使用Python程序，读取本地TPC-H的多个分片文件，基于elasticsearch库多线程导入，导入批次2000条一批。

## 测试结果

| 客户端并发线程数 | TPS (AnalyticDB MySQL) | TPS (ElasticSearch) |
| --- | --- | --- |
| 8 | 33033 | 12211 |
| 16 | 56816 | 7165 |
| 32 | 95083 | 6267 |
| 64 | 153857 | 5890 |
| 128 | 186732 | 5516 |

## 建表语句

AnalyticDB MySQL建表语句如下：

```
create table `lineitem` (
 `l_orderkey` bigint NOT NULL COMMENT '',
 `l_partkey` int NOT NULL COMMENT '',
 `l_suppkey` int NOT NULL COMMENT '',
 `l_linenumber` int NOT NULL COMMENT '',
 `l_quantity` decimal(15, 2) NOT NULL COMMENT '',
 `l_extendedprice` decimal(15, 2) NOT NULL COMMENT '',
 `l_discount` decimal(15, 2) NOT NULL COMMENT '',
 `l_tax` decimal(15, 2) NOT NULL COMMENT '',
 `l_returnflag` varchar NOT NULL COMMENT '',
 `l_linestatus` varchar NOT NULL COMMENT '',
 `l_shipdate` date NOT NULL COMMENT '',
 `l_commitdate` date NOT NULL COMMENT '',
 `l_receiptdate` date NOT NULL COMMENT '',
 `l_shipinstruct` varchar NOT NULL COMMENT '',
 `l_shipmode` varchar NOT NULL COMMENT '',
 `l_comment` varchar NOT NULL COMMENT ''
) DISTRIBUTE BY HASH(`l_orderkey`) INDEX_ALL='Y'
```

ElasticSearch建表语句如下：

```
curl -X PUT 'http://es_ip:9200/tpch' \
-H 'Content-Type: application/json' \
-d '{
  "settings": {
    "number_of_shards": 32,
    "number_of_replicas" : 2
  },
  "mappings": {
    "lineitem": {
      "properties": {
       "L_ORDERKEY": {
         "type": "integer"
       },
       "L_PARTKEY": {
         "type": "integer"
       },
       "L_SUPPKEY": {
```

```
      "type": "integer"
    },
    "L_LINENUMBER": {
      "type": "integer"
    },
    "L_QUANTITY": {
      "type": "double"
    },
    "L_EXTENDEDPRICE": {
      "type": "double"
    },
    "L_DISCOUNT": {
      "type": "double"
    },
    "L_TAX": {
      "type": "double"
    },
    "L_RETURNFLAG": {
      "type": "keyword"
    },
    "L_LINESTATUS": {
      "type": "keyword"
    },
    "L_SHIPDATE": {
      "type": "date"
    },
    "L_COMMITDATE": {
      "type": "date"
    },
    "L_RECEIPTDATE": {
      "type": "date"
    },
    "L_SHIPINSTRUCT": {
      "type": "keyword"
    },
    "L_SHIPMODE": {
      "type": "keyword"
    },
    "L_COMMENT": {
      "type": "keyword"
    }
```

```
      }
     }
   }
}'
```

数据导入脚本如下：

```
from threading import Thread
from elasticsearch import Elasticsearch
def func(i):
  es = Elasticsearch(hosts=[
    "es_ip:9200"
  ])
  idx = 0
  with open(r"lineitem.tbl.{}".format(i)) as f:
    actions = []
    while 1:
      r = f.readlines(2000)
      if not r:
        break
      for i in r:
        data = i.split('|')
        body = {
          'L_ORDERKEY': int(data[0]),
          'L_PARTKEY': int(data[1]),
          'L_SUPPKEY': int(data[2]),
          'L_LINENUMBER': int(data[3]),
          'L_QUANTITY': float(data[4]),
          'L_EXTENDEDPRICE': float(data[5]),
          'L_DISCOUNT': float(data[6]),
          'L_TAX': float(data[7]),
          'L_RETURNFLAG': data[8],
          'L_LINESTATUS': data[9],
          'L_SHIPDATE': data[10],
          'L_COMMITDATE': data[11],
          'L_RECEIPTDATE': data[12],
          'L_SHIPINSTRUCT': data[13],
          'L_SHIPMODE': data[14],
          'L_COMMENT': data[15]
        }
        actions.append({"index": {"_index": "tpch", "_type": "lineitem", "routing": int(data[0])}})
```

```
            actions.append(body)
            idx += 1
        es.bulk(actions)
        actions = []
        print(idx)
if __name__ == '__main__':
  for i in range(0, 16):
    Thread(target=func, args=(i + 1,)).start()
```

# 4.附录：测试过程详解

## 4.1. TPC-H测试

### 4.1.1. 准备工作

在阿里云公共云AnalyticDB MySQL环境运行和测试TPC-H（Transaction ProcessingPerformance Council）标准benchmark测试集之前，您需要完成以下准备工作。

1. 创建集群，请参见创建ADB MySQL版集群。
2. 为集群设置白名单，请参见设置白名单。
3. 在集群中创建数据库账号，请参见创建数据库账号。
4. 如需通过外网连接集群，请申请公网地址。

### 4.1.2. AnalyticDB MySQL

本文介绍AnalyticDB MySQL性能测试的场景信息。

本次性能测试将在AnalyticDB MySQL中创建以下八张数据表。

- CUSTOMER表

```
CREATE TABLE `CUSTOMER` (
 `C_CUSTKEY` int NOT NULL,
 `C_NAME` varchar NOT NULL,
 `C_ADDRESS` varchar NOT NULL,
 `C_NATIONKEY` int NOT NULL,
 `C_PHONE` varchar NOT NULL,
 `C_ACCTBAL` decimal(12, 2) NOT NULL,
 `C_MKTSEGMENT` varchar NOT NULL,
 `C_COMMENT` varchar NOT NULL,
 primary key (c_custkey)
)
DISTRIBUTE BY HASH(`c_custkey`)
INDEX_ALL='Y';
```

- LINEITEM表

```
CREATE TABLE `LINEITEM` (
 `L_ORDERKEY` bigint NOT NULL,
 `L_PARTKEY` int NOT NULL,
 `L_SUPPKEY` int NOT NULL,
 `L_LINENUMBER` bigint NOT NULL,
 `L_QUANTITY` decimal(12, 2) NOT NULL,
 `L_EXTENDEDPRICE` decimal(12, 2) NOT NULL,
 `L_DISCOUNT` decimal(12, 2) NOT NULL,
 `L_TAX` decimal(12, 2) NOT NULL,
 `L_RETURNFLAG` varchar NOT NULL,
 `L_LINESTATUS` varchar NOT NULL,
 `L_SHIPDATE` date NOT NULL,
 `L_COMMITDATE` date NOT NULL,
 `L_RECEIPTDATE` date NOT NULL,
 `L_SHIPINSTRUCT` varchar NOT NULL,
 `L_SHIPMODE` varchar NOT NULL,
 `L_COMMENT` varchar NOT NULL,
 primary key (l_orderkey,l_linenumber,l_shipdate)
)
DISTRIBUTE BY HASH(`l_orderkey`)
PARTITION BY VALUE(`date_format(l_shipdate, '%Y%m')`)
LIFECYCLE 90
INDEX_ALL='Y';
```

- NATION表

```
CREATE TABLE `NATION` (
 `N_NATIONKEY` int NOT NULL,
 `N_NAME` varchar NOT NULL,
 `N_REGIONKEY` int NOT NULL,
 `N_COMMENT` varchar,
 primary key (n_nationkey)
) DISTRIBUTE BY BROADCAST INDEX_ALL='Y';
```

- ORDERS表

```
CREATE TABLE `ORDERS` (
 `O_ORDERKEY` bigint NOT NULL,
 `O_CUSTKEY` int NOT NULL,
 `O_ORDERSTATUS` varchar NOT NULL,
 `O_TOTALPRICE` decimal(12, 2) NOT NULL,
 `O_ORDERDATE` date NOT NULL,
 `O_ORDERPRIORITY` varchar NOT NULL,
 `O_CLERK` varchar NOT NULL,
 `O_SHIPPRIORITY` int NOT NULL,
 `O_COMMENT` varchar NOT NULL,
 primary key (o_orderkey,o_orderdate)
)
DISTRIBUTE BY HASH(`o_orderkey`)
PARTITION BY VALUE(`date_format(O_ORDERDATE, '%Y%m')`)
LIFECYCLE 90
INDEX_ALL='Y';
```

- PART表

```
CREATE TABLE `PART` (
 `P_PARTKEY` int NOT NULL,
 `P_NAME` varchar NOT NULL,
 `P_MFGR` varchar NOT NULL,
 `P_BRAND` varchar NOT NULL,
 `P_TYPE` varchar NOT NULL,
 `P_SIZE` int NOT NULL,
 `P_CONTAINER` varchar NOT NULL,
 `P_RETAILPRICE` decimal(12, 2) NOT NULL,
 `P_COMMENT` varchar NOT NULL,
 primary key (p_partkey)
)
DISTRIBUTE BY HASH(`p_partkey`)
INDEX_ALL='Y';
```

- PARTSUPP表

```
CREATE TABLE `PARTSUPP` (
 `PS_PARTKEY` int NOT NULL,
 `PS_SUPPKEY` int NOT NULL,
 `PS_AVAILQTY` int NOT NULL,
 `PS_SUPPLYCOST` decimal(12, 2) NOT NULL,
 `PS_COMMENT` varchar NOT NULL,
 primary key (ps_partkey,ps_suppkey)
)
DISTRIBUTE BY HASH(`ps_partkey`)
INDEX_ALL='Y';
```

- REGION表

```
CREATE TABLE `REGION` (
 `R_REGIONKEY` int NOT NULL,
 `R_NAME` varchar NOT NULL,
 `R_COMMENT` varchar,
 primary key (r_regionkey)
)
DISTRIBUTE BY BROADCAST
INDEX_ALL='Y';
```

- SUPPLIER表

```
CREATE TABLE `SUPPLIER` (
 `S_SUPPKEY` int NOT NULL,
 `S_NAME` varchar NOT NULL,
 `S_ADDRESS` varchar NOT NULL,
 `S_NATIONKEY` int NOT NULL,
 `S_PHONE` varchar NOT NULL,
 `S_ACCTBAL` decimal(12, 2) NOT NULL,
 `S_COMMENT` varchar NOT NULL,
 primary key (s_suppkey)
)
DISTRIBUTE BY HASH(`s_suppkey`)
INDEX_ALL='Y';
```

# 4.1.3. Presto

本文介绍Presto性能测试的场景信息。

## 创建外表

本次性能测试将在Presto中创建以下八张文本格式的外表。

- CUSTOMER表

```
create external table customer(
 c_custkey integer,
 c_name varchar(25),
 c_address varchar(40),
 c_nationkey integer,
 c_phone char(15),
 c_acctbal decimal(15, 2),
 c_mktsegment char(10),
 c_comment varchar(117)
) row format delimited fields terminated by '|' location '/.../customer' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- LINEITEM表

```
create external table lineitem(
 l_orderkey integer,
 l_partkey integer,
 l_suppkey integer,
 l_linenumber integer,
 l_quantity decimal(15, 2),
 l_extendedprice decimal(15, 2),
 l_discount decimal(15, 2),
 l_tax decimal(15, 2),
 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)
) row format delimited fields terminated by '|' location '/.../lineitem' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- NATION表

```
create external table nation(
 n_nationkey integer,
 n_name char(25),
 n_regionkey integer,
 n_comment varchar(152)
) row format delimited fields terminated by '|' location '/.../nation' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- ORDERS表

```
create external table orders(
 o_orderkey integer,
 o_custkey integer,
 o_orderstatus char(1),
 o_totalprice decimal(15, 2),
 o_orderdate date,
 o_orderpriority char(15),
 o_clerk char(15),
 o_shippriority integer,
 o_comment varchar(79)
) row format delimited fields terminated by '|' location '/.../orders' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- PART表

```
create external table part(
 p_partkey integer,
 p_name varchar(55),
 p_mfgr char(25),
 p_brand char(10),
 p_type varchar(25),
 p_size integer,
 p_container char(10),
 p_retailprice decimal(15, 2),
 p_comment varchar(23)
) row format delimited fields terminated by '|' location '/.../part' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- PARTSUPP表

```
create external table partsupp(
 ps_partkey integer,
 ps_suppkey integer,
 ps_availqty integer,
 ps_supplycost decimal(15, 2),
 ps_comment varchar(199)
) row format delimited fields terminated by '|' location '/.../partsupp' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- REGION表

```
create external table region(
 r_regionkey integer,
 r_name char(25),
 r_comment varchar(152)
) row format delimited fields terminated by '|' location '/.../region/' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- SUPPLIER表

```
create external table supplier(
 s_suppkey integer,
 s_name char(25),
 s_address varchar(40),
 s_nationkey integer,
 s_phone char(15),
 s_acctbal decimal(15, 2),
 s_comment varchar(101)
) row format delimited fields terminated by '|' location '/.../supplier/' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

## 创建表

本次性能测试将在Presto中创建以下八张内表。

- CUSTOMER表

```
create table customer(
    c_custkey    integer,
    c_name       varchar(25),
    c_address    varchar(40),
    c_nationkey  integer,
    c_phone      char(15),
    c_acctbal    decimal(15,2),
    c_mktsegment  char(10),
    c_comment    varchar(117)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- LINEITEM表

```
create table lineitem
(
  l_orderkey   integer,
  l_partkey    integer,
  l_suppkey    integer,
  l_linenumber integer,
  l_quantity   decimal(15,2),
  l_extendedprice decimal(15,2),
  l_discount   decimal(15,2),
  l_tax        decimal(15,2),
  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)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- NATION表

```
create table nation(
    n_nationkey integer,
    n_name      char(25),
    n_regionkey integer,
    n_comment   varchar(152)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- ORDERS表

```
create table orders(
  o_orderkey     integer,
    o_custkey      integer,
    o_orderstatus  char(1),
    o_totalprice   decimal(15,2),
    o_orderdate    date,
    o_orderpriority  char(15),
    o_clerk        char(15),
    o_shippriority  integer,
    o_comment      varchar(79)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- PART表

```
create table part(
    p_partkey    integer,
    p_name       varchar(55),
    p_mfgr       char(25),
    p_brand      char(10),
    p_type       varchar(25),
    p_size       integer,
    p_container  char(10),
    p_retailprice decimal(15,2),
    p_comment    varchar(23)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- PARTSUPP表

```
create table partsupp(
  ps_partkey    integer,
    ps_suppkey    integer,
    ps_availqty   integer,
    ps_supplycost  decimal(15,2),
    ps_comment    varchar(199)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- REGION表

```
create table region(
  r_regionkey  integer,
  r_name       char(25),
  r_comment    varchar(152)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- SUPPLIER表

```
create table supplier(
  s_suppkey    integer,
  s_name       char(25),
  s_address    varchar(40),
  s_nationkey  integer,
  s_phone      char(15),
  s_acctbal    decimal(15,2),
  s_comment    varchar(101)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

# 4.1.4. Spark

本文介绍Spark性能测试的场景信息。

## 创建外表

本次性能测试将在Spark中创建以下八张文本格式的外表。

- CUSTOMER表

```
create external table customer(
 c_custkey integer,
 c_name varchar(25),
 c_address varchar(40),
 c_nationkey integer,
 c_phone char(15),
 c_acctbal decimal(15, 2),
 c_mktsegment char(10),
 c_comment varchar(117)
) row format delimited fields terminated by '|' location '/.../customer' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- LINEITEM表

```
create external table lineitem(
 l_orderkey integer,
 l_partkey integer,
 l_suppkey integer,
 l_linenumber integer,
 l_quantity decimal(15, 2),
 l_extendedprice decimal(15, 2),
 l_discount decimal(15, 2),
 l_tax decimal(15, 2),
 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)
) row format delimited fields terminated by '|' location '/.../lineitem' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- NATION表

```
create external table nation(
 n_nationkey integer,
 n_name char(25),
 n_regionkey integer,
 n_comment varchar(152)
) row format delimited fields terminated by '|' location '/.../nation' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- ORDERS表

```
create external table orders(
 o_orderkey integer,
 o_custkey integer,
 o_orderstatus char(1),
 o_totalprice decimal(15, 2),
 o_orderdate date,
 o_orderpriority char(15),
 o_clerk char(15),
 o_shippriority integer,
 o_comment varchar(79)
) row format delimited fields terminated by '|' location '/.../orders' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- PART表

```
create external table part(
 p_partkey integer,
 p_name varchar(55),
 p_mfgr char(25),
 p_brand char(10),
 p_type varchar(25),
 p_size integer,
 p_container char(10),
 p_retailprice decimal(15, 2),
 p_comment varchar(23)
) row format delimited fields terminated by '|' location '/.../part' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- PARTSUPP表

```
create external table partsupp(
 ps_partkey integer,
 ps_suppkey integer,
 ps_availqty integer,
 ps_supplycost decimal(15, 2),
 ps_comment varchar(199)
) row format delimited fields terminated by '|' location '/.../partsupp' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- REGION表

```
create external table region(
 r_regionkey integer,
 r_name char(25),
 r_comment varchar(152)
) row format delimited fields terminated by '|' location '/.../region/' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

- SUPPLIER表

```
create external table supplier(
 s_suppkey integer,
 s_name char(25),
 s_address varchar(40),
 s_nationkey integer,
 s_phone char(15),
 s_acctbal decimal(15, 2),
 s_comment varchar(101)
) row format delimited fields terminated by '|' location '/.../supplier/' TBLPROPERTIES(
 'serialization.null.format' = '',
 'serialization.encoding' = 'latin1'
);
```

## 创建表

本次性能测试将在Spark中创建以下八张内表。

- CUSTOMER表

```
create table customer(
    c_custkey    integer,
    c_name       varchar(25),
    c_address    varchar(40),
    c_nationkey  integer,
    c_phone      char(15),
    c_acctbal    decimal(15,2),
    c_mktsegment  char(10),
    c_comment    varchar(117)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- LINEITEM表

```
create table lineitem
(
  l_orderkey   integer,
  l_partkey    integer,
  l_suppkey    integer,
  l_linenumber integer,
  l_quantity   decimal(15,2),
  l_extendedprice decimal(15,2),
  l_discount   decimal(15,2),
  l_tax        decimal(15,2),
  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)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- NATION表

```
create table nation(
    n_nationkey integer,
    n_name      char(25),
    n_regionkey integer,
    n_comment   varchar(152)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- ORDERS表

```
create table orders(
  o_orderkey     integer,
    o_custkey      integer,
    o_orderstatus  char(1),
    o_totalprice   decimal(15,2),
    o_orderdate    date,
    o_orderpriority  char(15),
    o_clerk        char(15),
    o_shippriority  integer,
    o_comment      varchar(79)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- PART表

```
create table part(
    p_partkey    integer,
    p_name       varchar(55),
    p_mfgr       char(25),
    p_brand      char(10),
    p_type       varchar(25),
    p_size       integer,
    p_container  char(10),
    p_retailprice decimal(15,2),
    p_comment    varchar(23)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- PARTSUPP表

```
create table partsupp(
  ps_partkey    integer,
    ps_suppkey    integer,
    ps_availqty   integer,
    ps_supplycost  decimal(15,2),
    ps_comment    varchar(199)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- REGION表

```
create table region(
  r_regionkey  integer,
  r_name       char(25),
  r_comment    varchar(152)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

- SUPPLIER表

```
create table supplier(
  s_suppkey    integer,
  s_name       char(25),
  s_address    varchar(40),
  s_nationkey  integer,
  s_phone      char(15),
  s_acctbal    decimal(15,2),
  s_comment    varchar(101)
)
stored as parquet
TBLPROPERTIES("parquet.compression"="SNAPPY");
```

# 4.1.5. 构建数据

性能测试以TPC-H 100GB数据为测试数据，使用标准的DBGEN工具构造样本数据。

从TPC官网下载TPC-H标准的数据生成工具DBGEN，编译后生成二进制可执行文件dbgen。

```
./dbgen -s $scale -C $chunks -S $i -f
```

- `-s`：指定scale值，例如100GB时scale值为100，1TB时scale值为1000。
- `-C`：一共分成几个chunk。
- `-S`：：当前命令生成第几个 chunk。

> **说明** 一条语句只能生成一个 chunk。

更多dbgen使用方法请参见tpch-dbgen。

# 4.1.6. 导入数据

本文介绍如何将TPC-H 100GB测试数据分别导入AnalyticDB MySQL、Presto、Spark中。

下表列出了TPC-H测试数据集中的表数据条数。

| 表名 | 数据条数 |
| --- | --- |
| customer | 15000000 |
| lineitem | 600037902 |
| nation | 25 |
| orders | 150000000 |
| part | 20000000 |
| partsupp | 80000000 |
| region | 5 |
| supplier | 1000000 |

## AnalyticDB MySQL

- 在AnalyticDB MySQL中，使用LOAD DATA导入dbgen生成的文件。

```
LOAD DATA LOCAL INFILE 'customer.tbl' INTO TABLE CUSTOMER
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'orders.tbl' INTO TABLE ORDERS
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'lineitem.tbl' INTO TABLE LINEITEM
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'nation.tbl' INTO TABLE NATION
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'partsupp.tbl' INTO TABLE PARTSUPP
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL NFILE 'part.tbl' INTO TABLE PART
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'region.tbl' INTO TABLE REGION
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'supplier.tbl' INTO TABLE SUPPLIER
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
```

- 在AnalyticDB MySQL中，还可以通过OSS外表方式导入测试数据，请参见通过外表导入OSS数据。

## Presto/Spark

在Presto、Spark中，可以通过外表方式导入测试数据。

- CUSTOMER表

```
insert overwrite table customer
select
  c_custkey,
    c_name,
    c_address,
    c_nationkey,
    c_phone,
    c_acctbal,
    c_mktsegment,
    c_comment
from ${source_db}.customer;
```

- LINEITEM表

```
insert overwrite table lineitem
select
  l_orderkey,
  l_partkey,
  l_suppkey,
  l_linenumber,
  l_quantity,
  l_extendedprice,
  l_discount,
  l_tax,
  l_returnflag,
  l_linestatus,
  l_shipdate,
  l_commitdate,
  l_receiptdate,
  l_shipinstruct,
  l_shipmode,
  l_comment
from ${source_db}.lineitem;
```

- NATION表

```
insert overwrite table nation
select
    n_nationkey,
    n_name,
    n_regionkey,
    n_comment
from ${source_db}.nation;
```

- ORDERS表

```
insert overwrite table orders
select
    o_orderkey,
    o_custkey,
    o_orderstatus,
    o_totalprice,
    o_orderdate,
    o_orderpriority,
    o_clerk,
    o_shippriority,
    o_comment
from ${source_db}.orders;
```

- PART表

```
insert overwrite table part
select
    p_partkey,
    p_name,
    p_mfgr,
    p_brand,
    p_type,
    p_size,
    p_container,
    p_retailprice,
    p_comment
from ${source_db}.part;
```

- PARTSUPP表

```
insert overwrite table partsupp
select
    ps_partkey,
    ps_suppkey,
    ps_availqty,
    ps_supplycost,
    ps_comment
from ${source_db}.partsupp;
```

- REGION表

```
insert overwrite table region
select
    r_regionkey,
    r_name,
    r_comment
from ${source_db}.region;
```

- SUPPLIER表

```
insert overwrite table supplier
select
  s_suppkey,
  s_name,
  s_address,
  s_nationkey,
  s_phone,
  s_acctbal,
  s_comment
from ${source_db}.supplier;
```

## 4.1.7. TPC-H测试集

性能测试中将执行以下二十二个查询SQL。

- SQL1

```
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 '120' day
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

- SQL2

```
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 = 48
    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;
```

- SQL3

```
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 '1995-03-23'
    and l_shipdate > date '1995-03-23'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate
limit 10;
```

- SQL4

```
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '1996-07-01'
    and o_orderdate < date '1996-07-01' + 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;
```

- SQL5

```
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 = 'EUROPE'
    and o_orderdate >= date '1996-01-01'
    and o_orderdate < date '1996-01-01' + interval '1' year
group by
    n_name
order by
    revenue desc;
```

- SQL6

```
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1996-01-01'
    and l_shipdate < date '1996-01-01' + interval '1' year
    and l_discount between 0.02 - 0.01 and 0.02 + 0.01
    and l_quantity < 24;
```

- SQL7

```
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 = 'CANADA' and n2.n_name = 'BRAZIL')
                or (n1.n_name = 'BRAZIL' and n2.n_name = 'CANADA')
            )
            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;
```

- SQL8

```
select
    o_year,
    sum(case
        when nation = 'BRAZIL' 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 = 'AMERICA'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between date '1995-01-01' and date '1996-12-31'
            and p_type = 'LARGE ANODIZED COPPER'
    ) as all_nations
group by
    o_year
order by
    o_year;
```

- SQL9

```
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 '%maroon%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
```

- SQL10

```
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 '1993-02-01'
    and o_orderdate < date '1993-02-01' + 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;
```

- SQL11

```
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 = 'EGYPT'
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 = 'EGYPT'
        )
order by
    value desc;
```

- SQL12

```
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 ('FOB', 'AIR')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= date '1997-01-01'
    and l_receiptdate < date '1997-01-01' + interval '1' year
group by
    l_shipmode
order by
    l_shipmode;
```

- SQL13

```
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey) as c_count
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%special%deposits%'
        group by
            c_custkey
    ) c_orders
group by
    c_count
order by
    custdist desc,
    c_count desc;
```

- SQL14

```
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 '1997-06-01'
    and l_shipdate < date '1997-06-01' + interval '1' month;
```

- SQL15

```
create view revenue0 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= date '1995-02-01'
        and l_shipdate < date '1995-02-01' + 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;
```

- SQL16

```
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#45'
    and p_type not like 'SMALL ANODIZED%'
    and p_size in (47, 15, 37, 30, 46, 16, 18, 6)
    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;
```

- SQL17

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#51'
    and p_container = 'WRAP PACK'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```

- SQL18

```
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;
```

- SQL19

```
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#52'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 3 and l_quantity <= 3 + 10
        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#43'
        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#52'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 21 and l_quantity <= 21 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
```

- SQL20

```
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 'drab%'
            )
            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 = 'KENYA'
order by
    s_name;
```

- SQL21

```
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 = 'PERU'
group by
    s_name
order by
    numwait desc,
    s_name
limit 100;
```

- SQL22

```
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
                ('24', '32', '17', '18', '12', '14', '22')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
                    and substring(c_phone from 1 for 2) in
                        ('24', '32', '17', '18', '12', '14', '22')
            )
            and not exists (
                select
                    *
                from
                    orders
                where
                    o_custkey = c_custkey
            )
    ) as custsale
group by
    cntrycode
order by
    cntrycode;
```

> 说明 在Spark中，需要将 substring(c_phone from 1 for 2) 改写为 substring(c_phone, 1, 2) 。

# 4.2. TPC-DS测试

## 4.2.1. 准备工作

TPC-DS是一套决策支持系统测试基准，主要用于衡量大数据产品的分析性能。您可以在AnalyticDB MySQL中进行完整的TPC-DS基准测试，了解AnalyticDB MySQL的分析性能。

在云原生数据仓库AnalyticDB MySQL中进行TPC-DS（Transaction ProcessingPerformance Council）测试之前，您需要完成以下准备工作：

1. 创建集群，请参见创建ADB MySQL版集群。
2. 为集群设置白名单，请参见设置白名单。
3. 在集群中创建数据库账号，请参见创建数据库账号。
4. 如需通过外网连接集群，请申请公网地址。

## 4.2.2. 构建数据

性能测试以TPC-DS 1TB数据为测试数据，使用标准的DSDGEN工具构造样本数据。

从TPC官网下载TPC-DS标准的数据生成工具DSDGEN，编译后生成二进制可执行文件dsdgen。

```
./dsdgen -sc $scale -PARALLEL $chunks -CHILD $i
```

- -sc ：指定scale值，例如100GB时scale值为100，1TB时scale值为1000。
- -PARALLEL ：一共分成几个chunk。
- -CHILD ：当前命令生成第几个 chunk。

> 说明 一条语句只能生成一个 chunk。

更多dsdgen使用方法请参见tpc-ds。

## 4.2.3. 导入数据

本文介绍如何将TPC-DS 1TB测试数据导入AnalyticDB MySQL中。

下表列出了TPC-DS测试数据集中的表数据条数。

| 表名 | 数据条数 |
| --- | --- |
| store_sales | 2,879,987,999 |
| catalog_sales | 1,439,980,416 |
| web_sales | 720,000,376 |

| 表名 | 数据条数 |
| --- | --- |
| store_returns | 287,999,764 |
| catalog_returns | 143,996,756 |
| inventory | 783,000,000 |
| web_returns | 71,997,522 |
| customer | 12,000,000 |
| customer_address | 6,000,000 |
| item | 300,000 |
| customer_demographics | 1,920,800 |
| date_dim | 73,049 |
| time_dim | 86,400 |
| catalog_page | 30,000 |
| web_page | 3,000 |
| store | 1,002 |
| promotion | 1,500 |
| household_demographics | 7,200 |
| web_site | 54 |
| call_center | 42 |
| reason | 65 |
| warehouse | 20 |
| ship_mode | 20 |
| income_band | 20 |

在AnalyticDB MySQL中，使用LOAD DATA导入dsdgen生成的文件。

```
LOAD DATA LOCAL INFILE 'call_center.dat' INTO TABLE call_center
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'catalog_page.dat' INTO TABLE catalog_page
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'catalog_returns.dat' INTO TABLE catalog_returns
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
```

```
LOAD DATA LOCAL INFILE 'catalog_sales.dat' INTO TABLE catalog_sales
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'customer_address.dat' INTO TABLE customer_address
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'customer.dat' INTO TABLE customer
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'customer_demographics.dat' INTO TABLE customer_demographics
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'date_dim.dat' INTO TABLE date_dim
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'household_demographics.dat' INTO TABLE household_demographics
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'income_band.dat' INTO TABLE income_band
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'inventory.dat' INTO TABLE inventory
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'item.dat' INTO TABLE item
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'promotion.dat' INTO TABLE promotion
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'reason.dat' INTO TABLE reason
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'ship_mode.dat' INTO TABLE ship_mode
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'store.dat' INTO TABLE store
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'store_returns.dat' INTO TABLE store_returns
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'store_sales.dat' INTO TABLE store_sales
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'time_dim.dat' INTO TABLE time_dim
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'warehouse.dat' INTO TABLE warehouse
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'web_page.dat' INTO TABLE web_page
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'web_returns.dat' INTO TABLE web_returns
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'web_sales.dat' INTO TABLE web_sales
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
LOAD DATA LOCAL INFILE 'web_site.dat' INTO TABLE web_site
```

```
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\r\n';
```

在AnalyticDB MySQL中，还可以通过OSS外表方式导入测试数据，请参见通过外表导入OSS数据。

# 4.2.4. TPC-DS测试集

性能测试中将执行99个查询SQL。本文给出其中10个示例，如果需要全部查询SQL语句，请联系技术支持。

- SQL1

```
WITH customer_total_return AS (
    SELECT sr_customer_sk AS ctr_customer_sk, sr_store_sk AS ctr_store_sk, SUM(SR_RETURN_AMT) AS
ctr_total_return
    FROM store_returns, date_dim
    WHERE sr_returned_date_sk = d_date_sk
      AND d_year = 2000
    GROUP BY sr_customer_sk, sr_store_sk
  )
SELECT c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (
    SELECT AVG(ctr_total_return) * 1.2
    FROM customer_total_return ctr2
    WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk
  )
  AND s_store_sk = ctr1.ctr_store_sk
  AND s_state = 'TN'
  AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
LIMIT 100;
```

- SQL2

```
WITH wscs AS (
  SELECT sold_date_sk, sales_price
  FROM (
   SELECT ws_sold_date_sk AS sold_date_sk, ws_ext_sales_price AS sales_price
   FROM web_sales
   UNION ALL
   SELECT cs_sold_date_sk AS sold_date_sk, cs_ext_sales_price AS sales_price
   FROM catalog_sales
  )
 ),
 wswscs AS (
```

```
  SELECT d_week_seq, SUM(CASE
    WHEN d_day_name = 'Sunday' THEN sales_price
    ELSE NULL
   END) AS sun_sales, SUM(CASE
    WHEN d_day_name = 'Monday' THEN sales_price
    ELSE NULL
   END) AS mon_sales
   , SUM(CASE
    WHEN d_day_name = 'Tuesday' THEN sales_price
    ELSE NULL
   END) AS tue_sales, SUM(CASE
    WHEN d_day_name = 'Wednesday' THEN sales_price
    ELSE NULL
   END) AS wed_sales
   , SUM(CASE
    WHEN d_day_name = 'Thursday' THEN sales_price
    ELSE NULL
   END) AS thu_sales, SUM(CASE
    WHEN d_day_name = 'Friday' THEN sales_price
    ELSE NULL
   END) AS fri_sales
   , SUM(CASE
    WHEN d_day_name = 'Saturday' THEN sales_price
    ELSE NULL
   END) AS sat_sales
  FROM wscs, date_dim
  WHERE d_date_sk = sold_date_sk
  GROUP BY d_week_seq
 )
SELECT d_week_seq1, round(sun_sales1 / sun_sales2, 2)
 , round(mon_sales1 / mon_sales2, 2)
 , round(tue_sales1 / tue_sales2, 2)
 , round(wed_sales1 / wed_sales2, 2)
 , round(thu_sales1 / thu_sales2, 2)
 , round(fri_sales1 / fri_sales2, 2)
 , round(sat_sales1 / sat_sales2, 2)
FROM (
 SELECT wswscs.d_week_seq AS d_week_seq1, sun_sales AS sun_sales1, mon_sales AS mon_sales1, tue_
sales AS tue_sales1, wed_sales AS wed_sales1
  , thu_sales AS thu_sales1, fri_sales AS fri_sales1, sat_sales AS sat_sales1
```

```
 FROM wswscs, date_dim
 WHERE date_dim.d_week_seq = wswscs.d_week_seq
  AND d_year = 2001
) y, (
  SELECT wswscs.d_week_seq AS d_week_seq2, sun_sales AS sun_sales2, mon_sales AS mon_sales2, tue
_sales AS tue_sales2, wed_sales AS wed_sales2
    , thu_sales AS thu_sales2, fri_sales AS fri_sales2, sat_sales AS sat_sales2
  FROM wswscs, date_dim
  WHERE date_dim.d_week_seq = wswscs.d_week_seq
   AND d_year = 2001 + 1
 ) z
WHERE d_week_seq1 = d_week_seq2 - 53
ORDER BY d_week_seq1;
```

- SQL3

```
SELECT dt.d_year, item.i_brand_id AS brand_id, item.i_brand AS brand, SUM(ss_ext_sales_price) AS sum
_agg
FROM date_dim dt, store_sales, item
WHERE dt.d_date_sk = store_sales.ss_sold_date_sk
 AND store_sales.ss_item_sk = item.i_item_sk
 AND item.i_manufact_id = 128
 AND dt.d_moy = 11
GROUP BY dt.d_year, item.i_brand, item.i_brand_id
ORDER BY dt.d_year, sum_agg DESC, brand_id
LIMIT 100;
```

- SQL4

```
WITH year_total AS (
  SELECT c_customer_id AS customer_id, c_first_name AS customer_first_name, c_last_name AS custom
er_last_name, c_preferred_cust_flag AS customer_preferred_cust_flag, c_birth_country AS customer_bir
th_country
    , c_login AS customer_login, c_email_address AS customer_email_address, d_year AS dyear
    , SUM((ss_ext_list_price - ss_ext_wholesale_cost - ss_ext_discount_amt + ss_ext_sales_price) / 2) AS ye
ar_total
    , 's' AS sale_type
  FROM customer, store_sales, date_dim
  WHERE c_customer_sk = ss_customer_sk
   AND ss_sold_date_sk = d_date_sk
  GROUP BY c_customer_id, c_first_name, c_last_name, c_preferred_cust_flag, c_birth_country, c_login,
c_email_address, d_year
  UNION ALL
```

```
  SELECT c_customer_id AS customer_id, c_first_name AS customer_first_name, c_last_name AS custom
er_last_name, c_preferred_cust_flag AS customer_preferred_cust_flag, c_birth_country AS customer_bir
th_country
    , c_login AS customer_login, c_email_address AS customer_email_address, d_year AS dyear
    , SUM((cs_ext_list_price - cs_ext_wholesale_cost - cs_ext_discount_amt + cs_ext_sales_price) / 2) AS ye
ar_total
    , 'c' AS sale_type
  FROM customer, catalog_sales, date_dim
  WHERE c_customer_sk = cs_bill_customer_sk
   AND cs_sold_date_sk = d_date_sk
  GROUP BY c_customer_id, c_first_name, c_last_name, c_preferred_cust_flag, c_birth_country, c_login,
c_email_address, d_year
  UNION ALL
  SELECT c_customer_id AS customer_id, c_first_name AS customer_first_name, c_last_name AS custom
er_last_name, c_preferred_cust_flag AS customer_preferred_cust_flag, c_birth_country AS customer_bir
th_country
    , c_login AS customer_login, c_email_address AS customer_email_address, d_year AS dyear
    , SUM((ws_ext_list_price - ws_ext_wholesale_cost - ws_ext_discount_amt + ws_ext_sales_price) / 2) AS
year_total
    , 'w' AS sale_type
  FROM customer, web_sales, date_dim
  WHERE c_customer_sk = ws_bill_customer_sk
   AND ws_sold_date_sk = d_date_sk
  GROUP BY c_customer_id, c_first_name, c_last_name, c_preferred_cust_flag, c_birth_country, c_login,
c_email_address, d_year
 )
SELECT t_s_secyear.customer_id, t_s_secyear.customer_first_name, t_s_secyear.customer_last_name, t
_s_secyear.customer_preferred_cust_flag
FROM year_total t_s_firstyear, year_total t_s_secyear, year_total t_c_firstyear, year_total t_c_secyear, y
ear_total t_w_firstyear, year_total t_w_secyear
WHERE t_s_secyear.customer_id = t_s_firstyear.customer_id
 AND t_s_firstyear.customer_id = t_c_secyear.customer_id
 AND t_s_firstyear.customer_id = t_c_firstyear.customer_id
 AND t_s_firstyear.customer_id = t_w_firstyear.customer_id
 AND t_s_firstyear.customer_id = t_w_secyear.customer_id
 AND t_s_firstyear.sale_type = 's'
 AND t_c_firstyear.sale_type = 'c'
 AND t_w_firstyear.sale_type = 'w'
 AND t_s_secyear.sale_type = 's'
 AND t_c_secyear.sale_type = 'c'
 AND t_w_secyear.sale_type = 'w'
```

```
 AND t_s_firstyear.dyear = 2001
 AND t_s_secyear.dyear = 2001 + 1
 AND t_c_firstyear.dyear = 2001
 AND t_c_secyear.dyear = 2001 + 1
 AND t_w_firstyear.dyear = 2001
 AND t_w_secyear.dyear = 2001 + 1
 AND t_s_firstyear.year_total > 0
 AND t_c_firstyear.year_total > 0
 AND t_w_firstyear.year_total > 0
 AND CASE
  WHEN t_c_firstyear.year_total > 0 THEN t_c_secyear.year_total / t_c_firstyear.year_total
  ELSE NULL
 END > CASE
  WHEN t_s_firstyear.year_total > 0 THEN t_s_secyear.year_total / t_s_firstyear.year_total
  ELSE NULL
 END
 AND CASE
  WHEN t_c_firstyear.year_total > 0 THEN t_c_secyear.year_total / t_c_firstyear.year_total
  ELSE NULL
 END > CASE
  WHEN t_w_firstyear.year_total > 0 THEN t_w_secyear.year_total / t_w_firstyear.year_total
  ELSE NULL
 END
ORDER BY t_s_secyear.customer_id, t_s_secyear.customer_first_name, t_s_secyear.customer_last_name
, t_s_secyear.customer_preferred_cust_flag
LIMIT 100;
```

- SQL5

```
WITH ssr AS (
  SELECT s_store_id, SUM(sales_price) AS sales, SUM(profit) AS profit
   , SUM(return_amt) AS RETURNS, SUM(net_loss) AS profit_loss
  FROM (
   SELECT ss_store_sk AS store_sk, ss_sold_date_sk AS date_sk, ss_ext_sales_price AS sales_price, ss_ne
t_profit AS profit, CAST(0 AS decimal(7, 2)) AS return_amt
    , CAST(0 AS decimal(7, 2)) AS net_loss
   FROM store_sales
   UNION ALL
   SELECT sr_store_sk AS store_sk, sr_returned_date_sk AS date_sk, CAST(0 AS decimal(7, 2)) AS sales_pr
ice, CAST(0 AS decimal(7, 2)) AS profit, sr_return_amt AS return_amt
    , sr_net_loss AS net_loss
   FROM store_returns
```

```
    FROM store_returns
  ) salesreturns, date_dim, store
  WHERE date_sk = d_date_sk
   AND d_date BETWEEN CAST('2000-08-23' AS date) AND CAST('2000-08-23' AS date) + INTERVAL '14' DAY
   AND store_sk = s_store_sk
  GROUP BY s_store_id
 ),
 csr AS (
  SELECT cp_catalog_page_id, SUM(sales_price) AS sales, SUM(profit) AS profit
   , SUM(return_amt) AS RETURNS, SUM(net_loss) AS profit_loss
  FROM (
   SELECT cs_catalog_page_sk AS page_sk, cs_sold_date_sk AS date_sk, cs_ext_sales_price AS sales_pric
e, cs_net_profit AS profit, CAST(0 AS decimal(7, 2)) AS return_amt
    , CAST(0 AS decimal(7, 2)) AS net_loss
   FROM catalog_sales
   UNION ALL
   SELECT cr_catalog_page_sk AS page_sk, cr_returned_date_sk AS date_sk, CAST(0 AS decimal(7, 2)) AS
sales_price, CAST(0 AS decimal(7, 2)) AS profit, cr_return_amount AS return_amt
    , cr_net_loss AS net_loss
   FROM catalog_returns
  ) salesreturns, date_dim, catalog_page
  WHERE date_sk = d_date_sk
   AND d_date BETWEEN CAST('2000-08-23' AS date) AND CAST('2000-08-23' AS date) + INTERVAL '14' DAY
   AND page_sk = cp_catalog_page_sk
  GROUP BY cp_catalog_page_id
 ),
 wsr AS (
  SELECT web_site_id, SUM(sales_price) AS sales, SUM(profit) AS profit
   , SUM(return_amt) AS RETURNS, SUM(net_loss) AS profit_loss
  FROM (
   SELECT ws_web_site_sk AS wsr_web_site_sk, ws_sold_date_sk AS date_sk, ws_ext_sales_price AS sal
es_price, ws_net_profit AS profit, CAST(0 AS decimal(7, 2)) AS return_amt
    , CAST(0 AS decimal(7, 2)) AS net_loss
   FROM web_sales
   UNION ALL
   SELECT ws_web_site_sk AS wsr_web_site_sk, wr_returned_date_sk AS date_sk, CAST(0 AS decimal(7,
2)) AS sales_price, CAST(0 AS decimal(7, 2)) AS profit, wr_return_amt AS return_amt
    , wr_net_loss AS net_loss
   FROM web_returns
    LEFT JOIN web_sales
    ON wr_item_sk = ws_item_sk
```

```
        AND wr_order_number = ws_order_number
    ) salesreturns, date_dim, web_site
    WHERE date_sk = d_date_sk
     AND d_date BETWEEN CAST('2000-08-23' AS date) AND CAST('2000-08-23' AS date) + INTERVAL '14' DAY
     AND wsr_web_site_sk = web_site_sk
    GROUP BY web_site_id
  )
SELECT channel, id, SUM(sales) AS sales
  , SUM(RETURNS) AS RETURNS, SUM(profit) AS profit
FROM (
  SELECT 'store channel' AS channel, 'store'
    OR s_store_id AS id, sales, RETURNS
    , profit - profit_loss AS profit
  FROM ssr
  UNION ALL
  SELECT 'catalog channel' AS channel, 'catalog_page'
    OR cp_catalog_page_id AS id, sales, RETURNS
    , profit - profit_loss AS profit
  FROM csr
  UNION ALL
  SELECT 'web channel' AS channel, 'web_site'
    OR web_site_id AS id, sales, RETURNS
    , profit - profit_loss AS profit
  FROM wsr
) x
GROUP BY channel, id WITH ROLLUP
ORDER BY channel, id
LIMIT 100;
```

- SQL6

```
SELECT a.ca_state AS STATE, COUNT(*) AS cnt
FROM customer_address a, customer c, store_sales s, date_dim d, item i
WHERE a.ca_address_sk = c.c_current_addr_sk
  AND c.c_customer_sk = s.ss_customer_sk
  AND s.ss_sold_date_sk = d.d_date_sk
  AND s.ss_item_sk = i.i_item_sk
  AND d.d_month_seq = (
    SELECT DISTINCT d_month_seq
    FROM date_dim
    WHERE d_year = 2001
      AND d_moy = 1
  )
  AND i.i_current_price > 1.2 * (
    SELECT AVG(j.i_current_price)
    FROM item j
    WHERE j.i_category = i.i_category
  )
GROUP BY a.ca_state
HAVING COUNT(*) >= 10
ORDER BY cnt, a.ca_state
LIMIT 100;
```

- SQL7

```
SELECT i_item_id, AVG(ss_quantity) AS agg1, AVG(ss_list_price) AS agg2
 , AVG(ss_coupon_amt) AS agg3, AVG(ss_sales_price) AS agg4
FROM store_sales, customer_demographics, date_dim, item, promotion
WHERE ss_sold_date_sk = d_date_sk
 AND ss_item_sk = i_item_sk
 AND ss_cdemo_sk = cd_demo_sk
 AND ss_promo_sk = p_promo_sk
 AND cd_gender = 'M'
 AND cd_marital_status = 'S'
 AND cd_education_status = 'College'
 AND (p_channel_email = 'N'
  OR p_channel_event = 'N')
 AND d_year = 2000
GROUP BY i_item_id
ORDER BY i_item_id
LIMIT 100;
```

- SQL8

```
SELECT s_store_name, SUM(ss_net_profit)
FROM store_sales, date_dim, store, (
  SELECT ca_zip
  FROM (
   SELECT substr(ca_zip, 1, 5) AS ca_zip
   FROM customer_address
   WHERE substr(ca_zip, 1, 5) IN (
    '24128',
    '76232',
    '65084',
    '87816',
    '83926',
    '77556',
    '20548',
    '26231',
    '43848',
    '15126',
    '91137',
    '61265',
    '98294',
    '25782',
    '17920',
    '18426',
    '98235',
    '40081',
    '84093',
    '28577',
    '55565',
    '17183',
    '54601',
    '67897',
    '22752',
    '86284',
    '18376',
    '38607',
    '45200',
    '21756',
    '29741',
    '96765',
    '23932',
```

```
'89360',
'29839',
'25989',
'28898',
'91068',
'72550',
'10390',
'18845',
'47770',
'82636',
'41367',
'76638',
'86198',
'81312',
'37126',
'39192',
'88424',
'72175',
'81426',
'53672',
'10445',
'42666',
'66864',
'66708',
'41248',
'48583',
'82276',
'18842',
'78890',
'49448',
'14089',
'38122',
'34425',
'79077',
'19849',
'43285',
'39861',
'66162',
'77610',
'13695',
'99543',
```

```
'83444',
'83041',
'12305',
'57665',
'68341',
'25003',
'57834',
'62878',
'49130',
'81096',
'18840',
'27700',
'23470',
'50412',
'21195',
'16021',
'76107',
'71954',
'68309',
'18119',
'98359',
'64544',
'10336',
'86379',
'27068',
'39736',
'98569',
'28915',
'24206',
'56529',
'57647',
'54917',
'42961',
'91110',
'63981',
'14922',
'36420',
'23006',
'67467',
'32754',
'30903',
```

```
'20260',
'31671',
'51798',
'72325',
'85816',
'68621',
'13955',
'36446',
'41766',
'68806',
'16725',
'15146',
'22744',
'35850',
'88086',
'51649',
'18270',
'52867',
'39972',
'96976',
'63792',
'11376',
'94898',
'13595',
'10516',
'90225',
'58943',
'39371',
'94945',
'28587',
'96576',
'57855',
'28488',
'26105',
'83933',
'25858',
'34322',
'44438',
'73171',
'30122',
```

```
'34102',
'22685',
'71256',
'78451',
'54364',
'13354',
'45375',
'40558',
'56458',
'28286',
'45266',
'47305',
'69399',
'83921',
'26233',
'11101',
'15371',
'69913',
'35942',
'15882',
'25631',
'24610',
'44165',
'99076',
'33786',
'70738',
'26653',
'14328',
'72305',
'62496',
'22152',
'10144',
'64147',
'48425',
'14663',
'21076',
'18799',
'30450',
'63089',
'81019',
'68893',
```

```
'24996',
'51200',
'51211',
'45692',
'92712',
'70466',
'79994',
'22437',
'25280',
'38935',
'71791',
'73134',
'56571',
'14060',
'19505',
'72425',
'56575',
'74351',
'68786',
'51650',
'20004',
'18383',
'76614',
'11634',
'18906',
'15765',
'41368',
'73241',
'76698',
'78567',
'97189',
'28545',
'76231',
'75691',
'22246',
'51061',
'90578',
'56691',
'68014',
'51103',
'94167'
```

```
'57047',
'14867',
'73520',
'15734',
'63435',
'25733',
'35474',
'24676',
'94627',
'53535',
'17879',
'15559',
'53268',
'59166',
'11928',
'59402',
'33282',
'45721',
'43933',
'68101',
'33515',
'36634',
'71286',
'19736',
'58058',
'55253',
'67473',
'41918',
'19515',
'36495',
'19430',
'22351',
'77191',
'91393',
'49156',
'50298',
'87501',
'18652',
'53179',
'18767',
```

```
'63193',
'23968',
'65164',
'68880',
'21286',
'72823',
'58470',
'67301',
'13394',
'31016',
'70372',
'67030',
'40604',
'24317',
'45748',
'39127',
'26065',
'77721',
'31029',
'31880',
'60576',
'24671',
'45549',
'13376',
'50016',
'33123',
'19769',
'22927',
'97789',
'46081',
'72151',
'15723',
'46136',
'51949',
'68100',
'96888',
'64528',
'14171',
'79777',
'28709',
'11489',
```

```
'25103',
'32213',
'78668',
'22245',
'15798',
'27156',
'37930',
'62971',
'21337',
'51622',
'67853',
'10567',
'38415',
'15455',
'58263',
'42029',
'60279',
'37125',
'56240',
'88190',
'50308',
'26859',
'64457',
'89091',
'82136',
'62377',
'36233',
'63837',
'58078',
'17043',
'30010',
'60099',
'28810',
'98025',
'29178',
'87343',
'73273',
'30469',
'64034',
'39516',
'86057'
```

```
'86057',
'21309',
'90257',
'67875',
'40162',
'11356',
'73650',
'61810',
'72013',
'30431',
'22461',
'19512',
'13375',
'55307',
'30625',
'83849',
'68908',
'26689',
'96451',
'38193',
'46820',
'88885',
'84935',
'69035',
'83144',
'47537',
'56616',
'94983',
'48033',
'69952',
'25486',
'61547',
'27385',
'61860',
'58048',
'56910',
'16807',
'17871',
'35258',
'31387',
'35458',
```

```
      '35576'
    )
    INTERSECT
    SELECT ca_zip
    FROM (
     SELECT substr(ca_zip, 1, 5) AS ca_zip
      , COUNT(*) AS cnt
     FROM customer_address, customer
     WHERE ca_address_sk = c_current_addr_sk
      AND c_preferred_cust_flag = 'Y'
     GROUP BY ca_zip
     HAVING COUNT(*) > 10
    ) A1
  ) A2
 ) V1
WHERE ss_store_sk = s_store_sk
 AND ss_sold_date_sk = d_date_sk
 AND d_qoy = 2
 AND d_year = 1998
 AND (substr(s_zip, 1, 2) = substr(V1.ca_zip, 1, 2))
GROUP BY s_store_name
ORDER BY s_store_name
LIMIT 100;
```

- SQL9

```
SELECT CASE
        WHEN (
            SELECT COUNT(*)
            FROM store_sales
            WHERE ss_quantity BETWEEN 1 AND 20
        ) > 74129 THEN (
            SELECT AVG(ss_ext_discount_amt)
            FROM store_sales
            WHERE ss_quantity BETWEEN 1 AND 20
        )
        ELSE (
            SELECT AVG(ss_net_paid)
            FROM store_sales
            WHERE ss_quantity BETWEEN 1 AND 20
        )
    END AS bucket1
```

```
, CASE
    WHEN (
        SELECT COUNT(*)
        FROM store_sales
        WHERE ss_quantity BETWEEN 21 AND 40
    ) > 122840 THEN (
        SELECT AVG(ss_ext_discount_amt)
        FROM store_sales
        WHERE ss_quantity BETWEEN 21 AND 40
    )
    ELSE (
        SELECT AVG(ss_net_paid)
        FROM store_sales
        WHERE ss_quantity BETWEEN 21 AND 40
    )
END AS bucket2
, CASE
    WHEN (
        SELECT COUNT(*)
        FROM store_sales
        WHERE ss_quantity BETWEEN 41 AND 60
    ) > 56580 THEN (
        SELECT AVG(ss_ext_discount_amt)
        FROM store_sales
        WHERE ss_quantity BETWEEN 41 AND 60
    )
    ELSE (
        SELECT AVG(ss_net_paid)
        FROM store_sales
        WHERE ss_quantity BETWEEN 41 AND 60
    )
END AS bucket3
, CASE
    WHEN (
        SELECT COUNT(*)
        FROM store_sales
        WHERE ss_quantity BETWEEN 61 AND 80
    ) > 10097 THEN (
        SELECT AVG(ss_ext_discount_amt)
        FROM store_sales
        WHERE ss_quantity BETWEEN 61 AND 80
```

```
            WHERE ss_quantity BETWEEN 61 AND 80
        )
        ELSE (
            SELECT AVG(ss_net_paid)
            FROM store_sales
            WHERE ss_quantity BETWEEN 61 AND 80
        )
    END AS bucket4
    , CASE
        WHEN (
            SELECT COUNT(*)
            FROM store_sales
            WHERE ss_quantity BETWEEN 81 AND 100
        ) > 165306 THEN (
            SELECT AVG(ss_ext_discount_amt)
            FROM store_sales
            WHERE ss_quantity BETWEEN 81 AND 100
        )
        ELSE (
            SELECT AVG(ss_net_paid)
            FROM store_sales
            WHERE ss_quantity BETWEEN 81 AND 100
        )
    END AS bucket5
FROM reason
WHERE r_reason_sk = 1;
```

- SQL10

```
SELECT cd_gender, cd_marital_status, cd_education_status, COUNT(*) AS cnt1
 , cd_purchase_estimate, COUNT(*) AS cnt2, cd_credit_rating
 , COUNT(*) AS cnt3, cd_dep_count
 , COUNT(*) AS cnt4, cd_dep_employed_count
 , COUNT(*) AS cnt5, cd_dep_college_count
 , COUNT(*) AS cnt6
FROM customer c, customer_address ca, customer_demographics
WHERE c.c_current_addr_sk = ca.ca_address_sk
 AND ca_county IN ('Rush County', 'Toole County', 'Jefferson County', 'Dona Ana County', 'La Porte Count
y')
 AND cd_demo_sk = c.c_current_cdemo_sk
 AND EXISTS (
  SELECT *
  FROM store_sales, date_dim
  WHERE c.c_customer_sk = ss_customer_sk
   AND ss_sold_date_sk = d_date_sk
   AND d_year = 2002
   AND d_moy BETWEEN 1 AND 1 + 3
 )
 AND (EXISTS (
   SELECT *
   FROM web_sales, date_dim
   WHERE c.c_customer_sk = ws_bill_customer_sk
    AND ws_sold_date_sk = d_date_sk
    AND d_year = 2002
    AND d_moy BETWEEN 1 AND 1 + 3
  )
  OR EXISTS (
   SELECT *
   FROM catalog_sales, date_dim
   WHERE c.c_customer_sk = cs_ship_customer_sk
    AND cs_sold_date_sk = d_date_sk
    AND d_year = 2002
    AND d_moy BETWEEN 1 AND 1 + 3
  ))
GROUP BY cd_gender, cd_marital_status, cd_education_status, cd_purchase_estimate, cd_credit_rating,
cd_dep_count, cd_dep_employed_count, cd_dep_college_count
ORDER BY cd_gender, cd_marital_status, cd_education_status, cd_purchase_estimate, cd_credit_rating,
cd_dep_count, cd_dep_employed_count, cd_dep_college_count
LIMIT 100;
```