Задание на Хакатон.

Вам предлагается поэкспериментировать с промышленным тестом для аналитических систем – TPC-H. Полное описание теста доступно по вот этой ссылке:
https://www.tpc.org/TPC_Documents_Current_Versions/pdf/TPC-H_v3.0.1.pdf

Несмотря на уже почтенный возраст этого теста, структура данных представляет собой довольно типичный вариант для компании из сферы оптовой торговли. Тест содержит 23 типовых бизнес вопроса и предлагает типовые SQL запросы, которые получают соответствующие данные. Вопросы сформулированы достаточно сложно и, как результат, запросы также довольно сложны.

Мы подготовили данные в объектном хранилище S3 в колоночном формате Parquet. Для работы с этими данными будем использовать Trino. При заливке этих данных Trino выполнил некоторое разбиение файлов по умолчанию, судя по всему, просто согласно некоторому внутреннему пониманию о предельном размере файлов. В результате типичный файл большой таблицы имеет размер 1Gb. Общий объем сырых данных в этом тесте – 1Тб. Общий объем файлов в формате Parquet меньше ввиду колоночного принципа хранения и сжатия – около 270Гб.

Задача.

Найти новый способ разбиения файлов (партиционирование) и/или другой способ написания сложного SQL запроса, при котором время выполнения запроса уменьшается. Чтобы понимать, получилось у вас лучше или нет мы заранее выполнили эти запросы и в таблице ниже сведены результаты.

Не все запросы из теста будут задействованы, для экономии времени мы отбросили самые длинные запросы, простая проверка которых может занять у вас слишком много времени.

Однако, в такой конфигурации удалось выполнить не все запросы. Некоторые оказались «не по зубам» для движка Trino в силу специфики его внутренней организации. Поэтому задача со звёздочкой - найти такой способ организации данных ИЛИ написания запроса, чтобы ответ все-таки получить и не упираться в ограничения по памяти.

Некоторые рекомендации по работе над данными.

1. Главная рекомендация – не пытайтесь сразу работать над полным набором данных. Их слишком много, запросы выполняются долго и простые эксперименты просто сожрут время. Для проверки плана запроса и оценки влияния сделайте небольшое подмножество данных. Более того, вы можете воспользоваться встроенным механизмом генерации данных в Трино для теста tpch, в Трино всегда есть каталог tpch в котором есть несколько схем с загадочными именами sf1, sf10, sf100, sf1000. Эти имена означают не что иное как Scale Factor – параметр, который используется при генерации базы и который определяет объем исходных сырых данных. Для проверки гипотезы используйте 1 или 10 фактор, не больше.
2. Для экспериментов с различным написанием запросов можете использовать встроенные схемы где данные будут сгенерированы на лету. Но для экспериментов с партиционированием вам потребуется физически создать таблицу в хранилище. Для этого сначала создайте таблицу в том виде, в котором вы ее хотите видеть (естественно, что состав полей должен быть точно такой же), а потом воспользуйтесь командой

Insert into mycatalog.myschema.mytable

Select * from tpch.sf10.<table>

Чтобы влить туда данные из генератора.

3. Вам не обязательно сразу копировать все таблицы в свою схему. Вы можете использовать таблицы из разных каталогов и разных схем в любых комбинациях чтобы проверить вашу гипотезу. К примеру, вы можете создать 2-3 таблицы с разным разбиением для заказов (orders) и подставлять их поочереди в запрос, который берет остальные таблицы из основного набора или из вашего уменьшенного набора. Помните что Трино умеет работать с множеством источников данных в одном запросе.

Принцип оценивания результата.

- В зачет пойдут только те варианты, где скорость возрастает хотя бы на 20%. Меньше считается погрешностью выполнения и флуктуацией текущей загруженности системы (вы работаете в публичном облаке и это вполне обычная история).

- За каждый запрос, который стал быстрее на >20% от исходного мы начисляем 5 баллов.

- За каждый запрос, который стал быстрее на 50% от исходного мы начисляем 10 баллов.

- За получение работающего результата в задачах со звёздочкой – 20 баллов.

- Возможно вам придется создать несколько одинаковых по составу таблиц с другой структурой разбиения. И хотя место в хранилище нынче стоит недорого, увеличивать объемы нехорошо, это усложняет систему и делает данные потенциально несогласованными. Единственное оправдание – такое разбиение позволяет улучшить сразу несколько запросов. Исходя из сказанного: «штраф» 10 баллов, если вам понадобилось создать больше 2х одинаковых таблиц с разным разбиением по хранению.

Формат сдачи задания.

Ваш результат по каждому запросу будет состоять максимум из двух SQL скриптов:

1. Скрипт создания таблиц и наливки данных. Этот шаг НЕ ОБЯЗАТЕЛЕН если вам не пришлось создавать какие-то новые таблицы, то есть вы не применяли альтернативный способ разбиения данных
2. Файл с запросом. Собственно ваш новый запрос, который демонстрирует улучшение по времени по сравнению с оригинальным. Очевидно что выходные данные оригинального и вашего запроса отличатся не должны.

Рекомендуем именовать файлы по формату вида

<team>-Qxx-prepare.sql – для файлов подготовки данных

<team>-Qxx-run.sql – для файлов с запросами.

Приложение.

Опорные «тайминги» запросов, которые нужно оптимизировать:

Запрос	Время выполнения на модельном стенде.
Q1	2m13sec
Q2	41sec
Q3	3m35sec
Q5	6m34s
Q6	1m33s
Q7	6m
Q8	4m15s
Q11	33s
Q12	2m16s
Q13	3m45s
Q14	2m30s
Q17	5m
Q19	2m45s
Q20	2m52s

Время может немного «плавать» при повторных запусках.

Запросы.

Детальное описание схемы данных вы найдете по ссылке в документе с описанием теста TPC-H, здесь же мы приведем текст запросов, уже подготовленных к вашему набору данных.

Единственный нюанс, перед выполнением запроса нужно выполнить команду

USE <catalog>.<schema> чтобы имена таблиц не требовали полного трехярусного префикса.

В подготовленных стендах каталог называется data и схема hackaton, таким образом нужна команда USE data.hackaton;

Query 1.

```
select returnflag,
       linestatus,
       sum(quantity) as sum_qty,
       sum(extendedprice) as sum_base_price,
       sum(extendedprice*(1-discount)) as sum_disc_price,
       sum(extendedprice*(1-discount)*(1+tax)) as sum_charge,
       avg(quantity) as avg_qty,
       avg(extendedprice) as avg_price,
       avg(discount) as avg_disc,
       count(*) as count_order
from lineitem
where shipdate <= date '1998-12-01' - interval '1' year
group by returnflag,
         linestatus
order by returnflag,
         linestatus;
```

Query 2.

```
WITH min_supplycost AS (
    SELECT ps2.partkey, MIN(ps2.supplycost) AS min_cost
    FROM partsupp ps2
    JOIN supplier s2 ON s2.suppkey = ps2.suppkey
    JOIN nation n2 ON s2.nationkey = n2.nationkey
    JOIN region r2 ON n2.regionkey = r2.regionkey
    WHERE r2.name = 'EUROPE'
    GROUP BY ps2.partkey
)
SELECT s.acctbal,
       s.name,
       n.name AS nation_name,
       p.partkey,
       p.mfgr,
       s.address,
       s.phone,
       s.comment
FROM part p
JOIN partsupp ps ON p.partkey = ps.partkey
JOIN supplier s ON s.suppkey = ps.suppkey
JOIN nation n ON s.nationkey = n.nationkey
JOIN region r ON n.regionkey = r.regionkey
JOIN min_supplycost msc ON p.partkey = msc.partkey AND ps.supplycost =
msc.min_cost
WHERE p.size = 16
  AND p.type LIKE '%ANODIZED TIN'
  AND r.name = 'EUROPE'
ORDER BY s.acctbal DESC,
         n.name,
         s.name,
         p.partkey
limit 200;
```

Query 3.

```
select l.orderkey,
       sum(l.extendedprice*(1-l.discount)) as revenue,
       o.orderdate,
       o.shippriority
from customer c,
     orders o,
     lineitem l
where c.mktsegment = 'BUILDING'
    and c.custkey = o.custkey
    and l.orderkey = o.orderkey
    and o.orderdate <date '1995-03-11'
    and l.shipdate > date '1995-03-11'
group by l.orderkey,
         o.orderdate,
         o.shippriority
order by revenue desc,
         o.orderdate
limit 200;
```

Query 5.

```
Select n.name,
       sum(l.extendedprice * (1 - l.discount)) as revenue
from customer c,
     orders o,
     lineitem l,
     supplier s,
     nation n,
     region r
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 = 'ASIA'
    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;
```

Query 6.

```
select sum(l.extendedprice*l.discount) as revenue
from lineitem l
where l.shipdate >= date '1996-01-01'
    and l.shipdate < date '1996-01-01' + interval '1' year
    and l.discount between 0.05 - 0.01 and 0.05 + 0.01
    and l.quantity < 25;
```

Query 7.

```
select supp_nation,
       cust_nation,
       l_year,
       sum(volume) as revenue
from
    (select n1.name as supp_nation,
            n2.name as cust_nation,
            extract(year
                    from l.shipdate) as l_year,
            l.extendedprice * (1 - l.discount) as volume
     from supplier s,
          lineitem l,
          orders o,
          customer c,
          nation n1,
          nation n2
     where s.suppkey = l.suppkey
         and o.orderkey = l.orderkey
         and c.custkey = o.custkey
         and s.nationkey = n1.nationkey
         and c.nationkey = n2.nationkey
         and ((n1.name = 'FRANCE'
               and n2.name = 'GERMANY')
              or (n1.name = 'GERMANY'
                  and n2.name = 'FRANCE'))
         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;
```

Query 8.

```sql
select o_year,
       sum(case
               when nation = 'GERMANY' 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.name as nation
     from part p,
          supplier s,
          lineitem l,
          orders o,
          customer c,
          nation n1,
          nation n2,
          region r
     where p.partkey = l.partkey
         and s.suppkey = l.suppkey
         and l.orderkey = o.orderkey
         and o.custkey = c.custkey
         and c.nationkey = n1.nationkey
         and n1.regionkey = r.regionkey
         and r.name = 'EUROPE'
         and s.nationkey = n2.nationkey
         and o.orderdate between date '1995-01-01' and date '1996-12-31'
         and p.type = 'ECONOMY ANODIZED STEEL' ) as all_nations
group by o_year
order by o_year;
```

Query 11.

```sql
select ps.partkey,
       sum(ps.supplycost * ps.availqty) as value
from partsupp ps,
     supplier s,
     nation n
where ps.suppkey = s.suppkey
    and s.nationkey = n.nationkey
    and n.name = 'GERMANY'
group by ps.partkey
having sum(ps.supplycost * ps.availqty) >
    (select sum(ps1.supplycost * ps1.availqty) * 0.0000003
     from partsupp ps1,
          supplier s1,
          nation n1
     where ps1.suppkey = s1.suppkey
         and s1.nationkey = n1.nationkey
         and n1.name = 'GERMANY' )
order by value desc;
```

Query 12.

```sql
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 o,
     lineitem l
where o.orderkey = l.orderkey
    and l.shipmode in ('MAIL',
                       'SHIP')
    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;
```

Query 13.

```sql
select c_count,
       count(*) as custdist
from
    (select c2.custkey,
            count(o2.orderkey)
     from customer c2
     left outer join orders o2 on c2.custkey = o2.custkey
     and o2.comment not like '%special%requests%'
     group by c2.custkey) as c_orders (c_custkey, c_count)
group by c_count
order by custdist desc,
         c_count desc;
```

Query 14.

```
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 l,
     part p
where l.partkey = p.partkey
    and l.shipdate >= date '1995-09-01'
    and l.shipdate < date '1995-09-01' + interval '1' month;
```

Query 17.

```
select sum(l.extendedprice) / 7.0 as avg_yearly
from lineitem l,
     part p
where p.partkey = l.partkey
    and p.brand = 'Brand#23'
    and p.container = 'MED BOX'
    and l.quantity <
        (select 0.2 * avg(l2.quantity)
         from lineitem l2
         where l2.partkey = p.partkey );
```

Query 19

```
select sum(l.extendedprice * (1 - l.discount)) as revenue
from lineitem l,
     part p
where (p.partkey = l.partkey
       and p.brand = 'Brand#23'
       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 l.shipmode in ('AIR',
                          'AIR REG')
       and l.shipinstruct = 'DELIVER IN PERSON')
    or (p.partkey = l.partkey
        and p.brand = 'Brand#24'
        and p.container in ('MED BAG',
                            'MED BOX',
                            'MED PKG',
                            'MED PACK')
        and l.quantity >= 15
        and l.quantity <= 15 + 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#25'
        and p.container in ('LG CASE',
                            'LG BOX',
                            'LG PACK',
                            'LG PKG')
        and l.quantity >= 25
        and l.quantity <= 25 + 10
        and p.size between 1 and 15
        and l.shipmode in ('AIR',
                           'AIR REG')
        and l.shipinstruct = 'DELIVER IN PERSON' );
```

Query 20

```
select s.name,
       s.address
from supplier s,
     nation n
where s.suppkey in
        (select ps.suppkey
         from partsupp ps
         where ps.partkey in
                 (select p.partkey
                  from part p
                  where p.name like 'forest%' )
             and ps.availqty >
                 (select 0.5 * sum(l.quantity)
                  from lineitem l
                  where l.partkey = ps.partkey
                      and l.suppkey = ps.suppkey
                      and l.shipdate >= date '1994-01-01'
and l.shipdate < date '1994-01-01' + interval '1' year ) )
    and s.nationkey = n.nationkey
    and n.name = 'CANADA'
order by s.name
limit 20;
```

Запросы со звёздочкой*.

Эти три запроса не могут быть выполнены из-за особенностей реализации Trino. Будет существенным плюсом если вы найдете способ запустить их.

Query 9.

```
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 p,
          supplier s,
          lineitem l,
          partsupp ps,
          orders o,
          nation n
     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 '%white%' ) as profit
group by nation,
         o_year
order by nation,
         o_year desc;
```

Query 10.

```
--Memory limit exceeded
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 c,
     orders o,
     lineitem l,
     nation n
where c.custkey = o.custkey
    and l.orderkey = o.orderkey
    and o.orderdate >= date '1993-10-01'
    and o.orderdate < date '1993-10-01' + 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;
```

Query 22.

```
select cntrycode,
       count(*) as numcust,
       sum(c_acctbal) as totacctbal
from
    (select substring(c.phone
                      from 1
                      for 2) as cntrycode,
            c.acctbal as c_acctbal
     from customer c
     where substring(c.phone
                     from 1
                     for 2) in ('13', '31', '23', '29', '30', '18', '17')
         and c.acctbal >
             (select avg(c2.acctbal)
              from customer c2
              where c2.acctbal > 0.00
                  and substring (c2.phone
                                 from 1
                                 for 2) in ('13', '31', '23', '29', '30', '18',
'17') )
         and not exists
             (select *
              from orders o
              where o.custkey = c.custkey ) ) as custsale
group by cntrycode
order by cntrycode;
```