CHAPTER 4

TPCH DATABASE BENCHMARK TEST [APACHE DRILL]

4.1 CONFIGURATION

4.1.1 TPCH Configuration

4.1.1.a Update Compiler, Database and Machine information

```
################
## CHANGE NAME OF ANSI COMPILER HERE
################
CC    = gcc
# Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata)
#                                   SQLSERVER, SYBASE, ORACLE, VECTORWISE
# Current values for MACHINE are:  ATT, DOS, HP, IBM, ICL, MVS,
#                                   SGI, SUN, U2200, VMS, LINUX, WIN32
# Current values for WORKLOAD are:  TPCH
DATABASE= SQLSERVER
MACHINE = LINUX
WORKLOAD = TPCH
#
CFLAGS  = -g -DDBNAME=\"dss\" -D$(MACHINE) -D$(DATABASE) -D$(WORKLOAD)
-DRNG_TEST -D_FILE_OFFSET_BITS=64
LDFLAGS = -O
# The OBJ,EXE and LIB macros will need to be changed for compilation under
#  Windows NT
OBJ   = .o
EXE   =
LIBS  = -lm
#
# NO CHANGES SHOULD BE NECESSARY BELOW THIS LINE
###############
```

4.1.1.b Update values for SQLSERVER in *tpcd.h* header file.

```
#ifdef  SQLSERVER
#define GEN_QUERY_PLAN  "set showplan on\nset noexec on\ngo\n"
#define START_TRAN      "**BEGIN WORK;**"
#define END_TRAN        "**COMMIT WORK;**"
#define SET_OUTPUT      ""
#define SET_ROWCOUNT    "limit %d;\n\n"
#define SET_DBASE       "use %s;\n"
#endif
```

4.1.1.c Generate Data [100 MB].

```
$ ./dbgen -s 0.1
```

4.1.2 MongoDB Configuration

4.1.2.a Start MongoDB

```
$ sudo service mongod start
```

4.1.2.b Verify that MongoDB has started successfully

```
$ sudo service mongod status
```

4.1.2.c Open Mongo Shell

```
$ mongo
```

4.2 INSTANTIATE AND POPULATE DATABASE

4.2.1 Convert Data Tables from TBL to JSON Format

Python Script:

```python
import pandas as pd
import os
import time

schema = {
    'customer': ['C_CUSTKEY', 'C_NAME', 'C_ADDRESS', 'C_NATIONKEY',
'C_PHONE', 'C_ACCTBAL', 'C_MKTSEGMENT', 'C_COMMENT'],
    'lineitem': ['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'],
    'nation': ['N_NATIONKEY', 'N_NAME', 'N_REGIONKEY', 'N_COMMENT'],
    'orders': ['O_ORDERKEY', 'O_CUSTKEY', 'O_ORDERSTATUS', 'O_TOTALPRICE',
'O_ORDERDATE', 'O_ORDERPRIORITY', 'O_CLERK', 'O_SHIPPRIORITY',
'O_COMMENT'],
    'part': ['P_PARTKEY', 'P_NAME', 'P_MFGR', 'P_BRAND', 'P_TYPE',
'P_SIZE', 'P_CONTAINER', 'P_RETAILPRICE', 'P_COMMENT'],
    'partsupp': ['PS_PARTKEY', 'PS_SUPPKEY', 'PS_AVAILQTY',
'PS_SUPPLYCOST', 'PS_COMMENT'],
    'region': ['R_REGIONKEY', 'R_NAME', 'R_COMMENT'],
    'supplier': ['S_SUPPKEY', 'S_NAME', 'S_ADDRESS', 'S_NATIONKEY',
'S_PHONE', 'S_ACCTBAL', 'S_COMMENT']
}

for table_name in schema:
    schema[table_name] = ','.join(schema[table_name]).lower().split(',')

try:
    os.mkdir('jsons')
except:
    pass

tick_f = time.time()

for table_name in schema:
    tick = time.time()
    df = pd.read_table(f'tables/{table_name}.tbl',
                       sep='|', index_col=False, header=None)
    df.drop(df.columns[-1], axis=1, inplace=True)
    df.columns = schema[table_name]
    df.to_json(f'jsons/{table_name}.json', orient='records')

    tock = time.time()
    print("Time taken to convert %s table: %.3f s" % (table_name, tock -
tick))

tock = time.time()

print("Total time taken: %.3f s" % (tock - tick_f))
```

Shell command to run script:

```
python tbl_to_json.py
```

Output:

```
(base)vivek@voyager:/mnt/c/Users/iamvi/Desktop/projects/MajorProjects/Codes
/TPCH$ python tbl_to_json.py

Time taken to convert customer table: 0.110 s
Time taken to convert lineitem table: 4.681 s
Time taken to convert nation table: 0.028 s
Time taken to convert orders table: 0.788 s
Time taken to convert part table: 0.119 s
Time taken to convert partsupp table: 0.385 s
Time taken to convert region table: 0.012 s
Time taken to convert supplier table: 0.016 s

Total time taken: 6.140 s
```

4.2.2 Create and Populate Database

Python Script:

```
#!/usr/bin/env python
# coding: utf-8

import os
import time
import glob

table_names = [x.split('/')[-1][:-5] for x in glob.glob('./jsons/*.json')]

tick_f = time.time()

for table_name in table_names:
    tick = time.time()

    os.system(
        f"""mongoimport --jsonArray --db tpch --collection {table_name}
--file jsons/{table_name}.json""")

    tock = time.time()
    print("Time taken to populate %s table: %.2f s" %
          (table_name, tock - tick))

tock = time.time()

print("Total time taken to populate: %.2f s" % (tock - tick_f))
```

Shell command to run script:

```
$ python populate_mongodb.py
```

Output:

```
Time taken to populate customer table: 0.61 s
Time taken to populate lineitem table: 34.69 s
Time taken to populate nation table: 0.09 s
Time taken to populate orders table: 9.04 s
Time taken to populate part table: 0.81 s
Time taken to populate partsupp table: 2.71 s
Time taken to populate region table: 0.04 s
Time taken to populate supplier table: 0.07 s

Total time taken to populate: 48.08 s
```

4.2.3 Preparing Apache Drill for MongoDB

4.2.3.a. Navigate to the Drill installation director:

```
$ cd apache-drill-1.19.0/
```

4.2.3.b. Start the Drill shell:

```
$ bin/drill-embedded

Apache Drill 1.19.0
"Say hello to my little Drill."
apache drill>
```

4.2.3.c. Show databases;

```
apache drill> SHOW DATABASES;

+--------------------+
|    SCHEMA_NAME     |
+--------------------+
| cp.default         |
| dfs.default        |
| dfs.root           |
| dfs.tmp            |
| information_schema |
| mongo.admin        |
| mongo.config       |
| mongo.database     |
| mongo.local        |
| mongo.temp         |
| mongo.tpch         |
| sys                |
+--------------------+
12 rows selected (5.562 seconds)
```

4.2.3.a. Select *mongo.tpch* Database

```
apache drill> use mongo.tpch;

+------+----------------------------------------+
|  ok  |                summary                 |
+------+----------------------------------------+
| true | Default schema changed to [mongo.tpch] |
+------+----------------------------------------+
1 row selected (0.145 seconds)

apache drill (mongo.tpch)>
```

4.2 TPCH 22 BENCHMARK QUERIES

SQL queries can be executed in the drill terminal. e.g.

```
apache drill (mongo.tpch)> Select * FROM nation;
```

4.2.1 Pricing Summary Report Query (Q1)

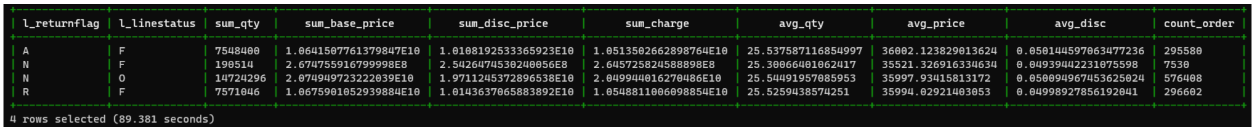
Statement:
This query reports the amount of business that was billed, shipped, and returned.

SQL Query:

```
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 '108' day(3)
group by
  l_returnflag,
  l_linestatus
order by
  l_returnflag,
  l_linestatus;
```

Output :



```
+--------------+--------------+----------+-----------------------+-----------------------+-----------------------+--------------------+--------------------+----------------------+-------------+
| l_returnflag | l_linestatus | sum_qty  |    sum_base_price     |    sum_disc_price     |      sum_charge       |      avg_qty       |     avg_price      |       avg_disc       | count_order |
+--------------+--------------+----------+-----------------------+-----------------------+-----------------------+--------------------+--------------------+----------------------+-------------+
| A            | F            | 7548400  | 1.0641507761379847E10 | 1.0108192533365923E10 | 1.0513502662898764E10 | 25.537587116854997 | 36002.123829013624 | 0.050144597063477236 | 295580      |
| N            | F            | 190514   | 2.674755916799998E8   | 2.5426474530240056E8  | 2.645725824588898E8   | 25.30066401062417  | 35521.326916334634 | 0.04939442231075598  | 7530        |
| N            | O            | 14724296 | 2.074949723222039E10  | 1.9711245372896538E10 | 2.049944016270486E10  | 25.54491957085953  | 35997.93415813172  | 0.050094967453625024 | 576408      |
| R            | F            | 7571046  | 1.0675901052939884E10 | 1.0143637065883892E10 | 1.0548811006098854E10 | 25.5259438574251   | 35994.02921403053  | 0.04998927856192041  | 296602      |
+--------------+--------------+----------+-----------------------+-----------------------+-----------------------+--------------------+--------------------+----------------------+-------------+
4 rows selected (89.381 seconds)
```

Execution Time: 89.381 sec

4.2.2 Minimum Cost Supplier Query (Q2)

Statement:
This query finds which supplier should be selected to place an order for a given part in a given region.

SQL Query:

```
select
  s.s_acctbal,
  s.s_name,
  n.n_name,
  p.p_partkey,
  p.p_mfgr,
  s.s_address,
  s.s_phone,
  s.s_comment
from
  part p,
  supplier s,
  partsupp ps,
  nation n,
  region r
where
  p.p_partkey = ps.ps_partkey
  and s.s_suppkey = ps.ps_suppkey
  and p.p_size = 30
  and p.p_type like '%STEEL'
  and s.s_nationkey = n.n_nationkey
  and n.n_regionkey = r.r_regionkey
  and r.r_name = 'ASIA'
  and ps.ps_supplycost = (
    select
      min(ps_supplycost)
    from
      partsupp,
      supplier,
      nation,
      region
    where
      p.p_partkey = ps.ps_partkey
      and s.s_suppkey = ps.ps_suppkey
      and s.s_nationkey = n.n_nationkey
      and n.n_regionkey = r.r_regionkey
      and r.r_name = 'ASIA'
  )
order by
  s.s_acctbal desc,
  n.n_name,
  s.s_name,
  p.p_partkey
limit
  100;
```

4.2.3 Shipping Priority Query (Q3)

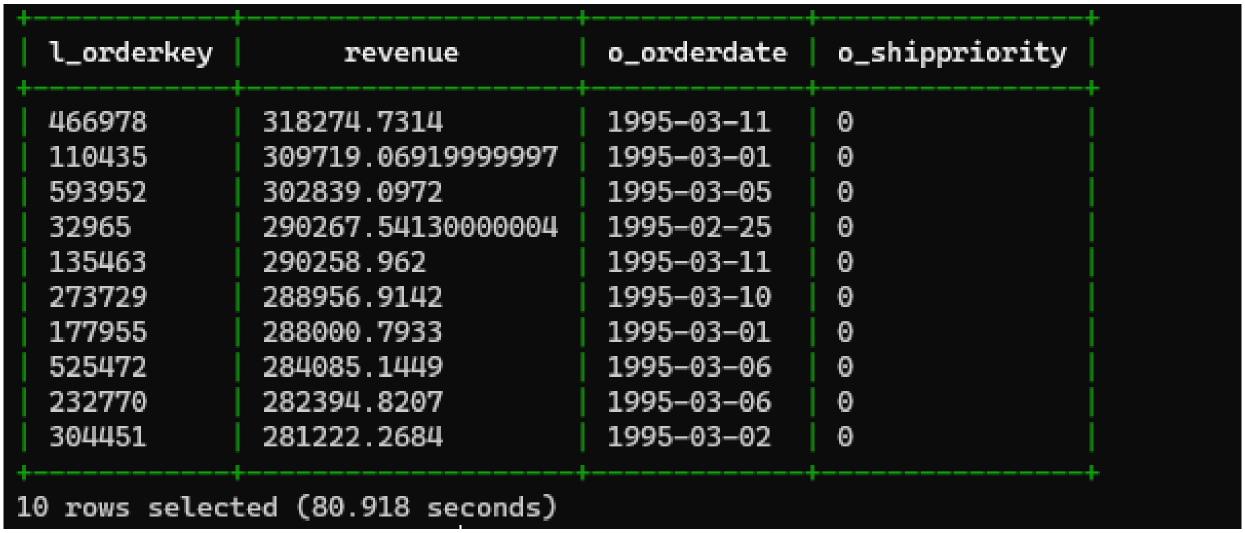
Statement:
This query retrieves the 10 unshipped orders with the highest value.

SQL Query:

```
select
  l.l_orderkey,
  sum(l.l_extendedprice * (1 - l.l_discount)) as revenue,
  o.o_orderdate,
  o.o_shippriority
```

```
from
  customer c,
  orders o,
  lineitem l
where
  c.c_mktsegment = 'AUTOMOBILE'
  and c.c_custkey = o.o_custkey
  and l.l_orderkey = o.o_orderkey
  and o.o_orderdate < date '1995-03-13'
  and l.l_shipdate > date '1995-03-13'
group by
  l.l_orderkey,
  o.o_orderdate,
  o.o_shippriority
order by
  revenue desc,
  o.o_orderdate
limit
  10;
```

Output :



```
+------------+---------------------+------------+----------------+
| l_orderkey |       revenue       | o_orderdate | o_shippriority |
+------------+---------------------+------------+----------------+
| 466978     | 318274.7314         | 1995-03-11 | 0              |
| 110435     | 309719.06919999997  | 1995-03-01 | 0              |
| 593952     | 302839.0972         | 1995-03-05 | 0              |
| 32965      | 290267.54130000004  | 1995-02-25 | 0              |
| 135463     | 290258.962          | 1995-03-11 | 0              |
| 273729     | 288956.9142         | 1995-03-10 | 0              |
| 177955     | 288000.7933         | 1995-03-01 | 0              |
| 525472     | 284085.1449         | 1995-03-06 | 0              |
| 232770     | 282394.8207         | 1995-03-06 | 0              |
| 304451     | 281222.2684         | 1995-03-02 | 0              |
+------------+---------------------+------------+----------------+
10 rows selected (80.918 seconds)
```

Execution Time: 80.918 sec

4.2.4 Order Priority Checking Query (Q4)

Statement:

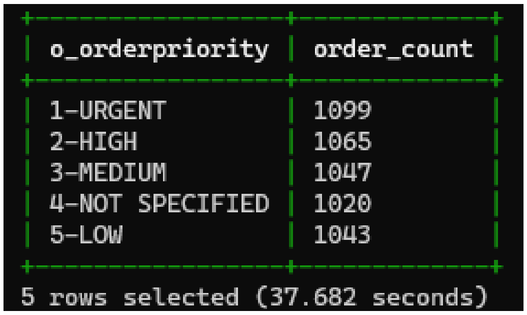
This query determines how well the order priority system is working and gives an assessment of customer satisfaction.

SQL Query:

```
select
  o.o_orderpriority,
  count(*) as order_count
from
  orders o

where
  o.o_orderdate >= date '1996-10-01'
  and o.o_orderdate < date '1996-10-01' + interval '3' month
  and
  exists (
    select
      *
    from
      lineitem l
    where
      l.l_orderkey = o.o_orderkey
      and l.l_commitdate < l.l_receiptdate
  )
group by
  o.o_orderpriority
order by
  o.o_orderpriority;
```

Output :



```
+-----------------+-------------+
| o_orderpriority | order_count |
+-----------------+-------------+
| 1-URGENT        | 1099        |
| 2-HIGH          | 1065        |
| 3-MEDIUM        | 1047        |
| 4-NOT SPECIFIED | 1020        |
| 5-LOW           | 1043        |
+-----------------+-------------+
5 rows selected (37.682 seconds)
```

Execution Time: 37.682 sec

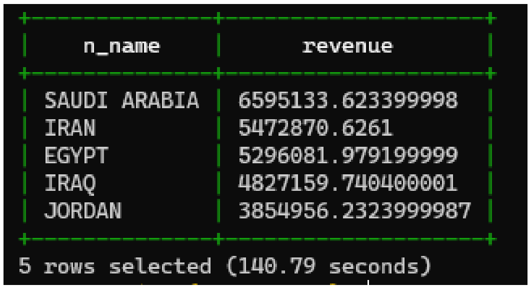
4.2.5 Local Supplier Volume Query (Q5)

Statement:
This query lists the revenue volume done through local suppliers.

SQL Query:

```sql
select
  n.n_name,
  sum(l.l_extendedprice * (1 - l.l_discount)) as revenue
from
  customer c,
  orders o,
  lineitem l,
  supplier s,
  nation n,
  region r
where
  c.c_custkey = o.o_custkey
  and l.l_orderkey = o.o_orderkey
  and l.l_suppkey = s.s_suppkey
  and c.c_nationkey = s.s_nationkey
  and s.s_nationkey = n.n_nationkey
  and n.n_regionkey = r.r_regionkey
  and r.r_name = 'MIDDLE EAST'
  and o.o_orderdate >= date '1994-01-01'
  and o.o_orderdate < date '1994-01-01' + interval '1' year
group by
  n.n_name
order by
  revenue desc;
```

Output :



```
+---------------+--------------------+
|    n_name     |      revenue       |
+---------------+--------------------+
| SAUDI ARABIA  | 6595133.623399998  |
| IRAN          | 5472870.6261       |
| EGYPT         | 5296081.979199999  |
| IRAQ          | 4827159.740400001  |
| JORDAN        | 3854956.2323999987 |
+---------------+--------------------+
5 rows selected (140.79 seconds)
```

Execution Time: 140.79 sec

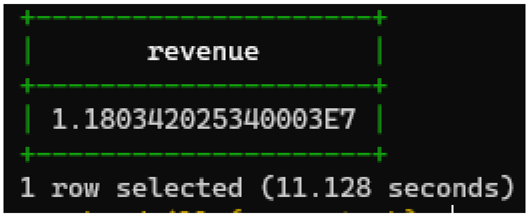
4.2.6 Forecasting Revenue Change Query (Q6)

Statement:
This query quantifies the amount of revenue increase that would have resulted from eliminating certain company-wide discounts in a given percentage range in a given year. Asking this type of "what if" query can be used to look for ways to increase revenues.

SQL Query:

```sql
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;
```

Output :



```
+----------------------+
|       revenue        |
+----------------------+
| 1.180342025340003E7  |
+----------------------+
1 row selected (11.128 seconds)
```

Execution Time: 11.128 sec

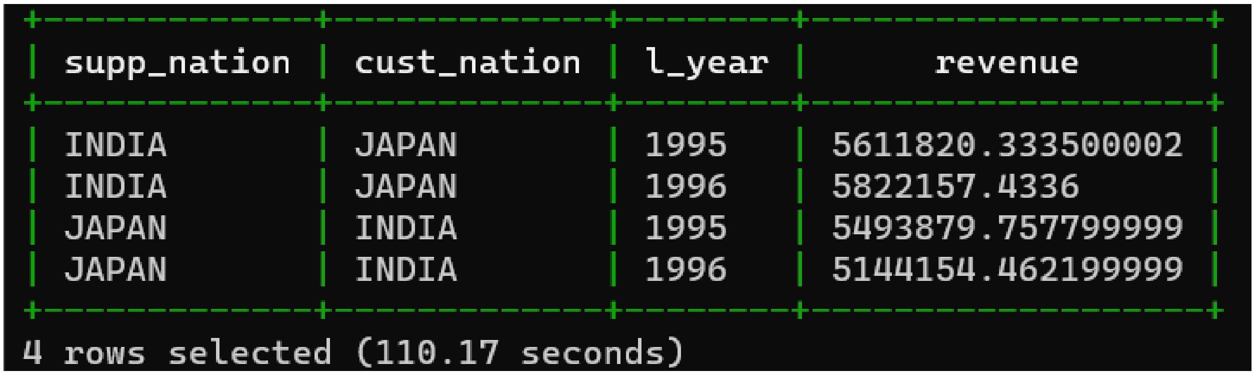
4.2.7 Volume Shipping Query (Q7)

Statement:
This query determines the value of goods shipped between certain nations to help in the re-negotiation of shipping contracts.

SQL Query:

```sql
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.l_shipdate) as l_year,
      l.l_extendedprice * (1 - l.l_discount) as volume
    from
      supplier s,
      lineitem l,
      orders o,
      customer c,
      nation n1,
      nation n2
    where
      s.s_suppkey = l.l_suppkey
      and o.o_orderkey = l.l_orderkey
      and c.c_custkey = o.o_custkey
      and s.s_nationkey = n1.n_nationkey
      and c.c_nationkey = n2.n_nationkey
      and (
        (n1.n_name = 'JAPAN' and n2.n_name = 'INDIA')
        or (n1.n_name = 'INDIA' and n2.n_name = 'JAPAN')
      )
      and l.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;
```

Output :



```
+---------------+---------------+---------+--------------------+
| supp_nation   | cust_nation   | l_year  |      revenue       |
+---------------+---------------+---------+--------------------+
| INDIA         | JAPAN         | 1995    | 5611820.333500002  |
| INDIA         | JAPAN         | 1996    | 5822157.4336       |
| JAPAN         | INDIA         | 1995    | 5493879.757799999  |
| JAPAN         | INDIA         | 1996    | 5144154.462199999  |
+---------------+---------------+---------+--------------------+
4 rows selected (110.17 seconds)
```

Execution Time: 110.17 sec

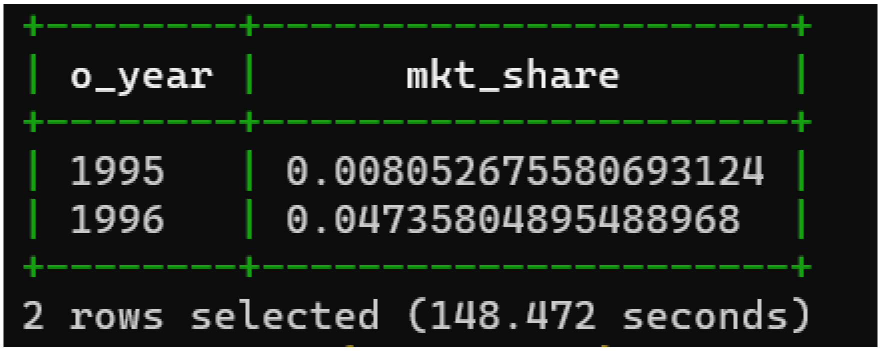
4.2.8 National Market Share Query (Q8)

Statement:
This query determines how the market share of a given nation within a given region has changed over two years for a given part type.

SQL Query:

```
select
  o_year,
  sum(case
    when nation = 'EGYPT' then volume
    else 0
  end) / sum(volume) as mkt_share
from
  (
    select
      extract(year from o.o_orderdate) as o_year,
      l.l_extendedprice * (1 - l.l_discount) as volume,
      n2.n_name as nation
    from
      part p,
      supplier s,
      lineitem l,
      orders o,
      customer c,
      nation n1,
      nation n2,
      region r
    where
      p.p_partkey = l.l_partkey
      and s.s_suppkey = l.l_suppkey
      and l.l_orderkey = o.o_orderkey
      and o.o_custkey = c.c_custkey
      and c.c_nationkey = n1.n_nationkey
      and n1.n_regionkey = r.r_regionkey
      and r.r_name = 'MIDDLE EAST'
      and s.s_nationkey = n2.n_nationkey
      and o.o_orderdate between date '1995-01-01' and date '1996-12-31'
      and p.p_type = 'PROMO BRUSHED COPPER'
  ) as all_nations
group by
  o_year
order by
  o_year;
```

Output :



```
+---------+-----------------------+
| o_year  |       mkt_share       |
+---------+-----------------------+
| 1995    | 0.008052675580693124  |
| 1996    | 0.04735804895488968   |
+---------+-----------------------+
2 rows selected (148.472 seconds)
```

Execution Time: 148.472 sec

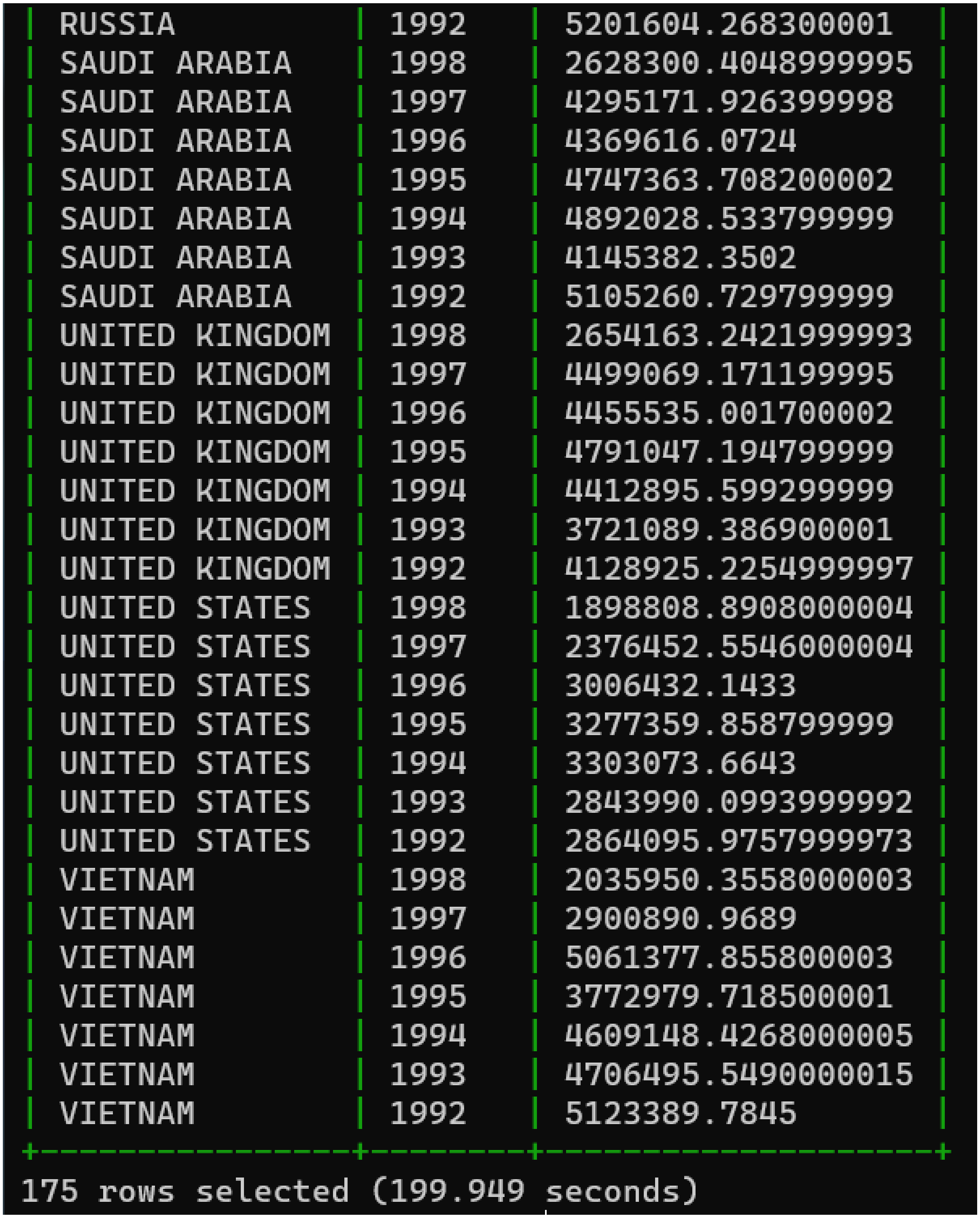
4.2.9 Product Type Profit Measure Query (Q9)

Statement:
This query determines how much profit is made on a given line of parts, broken out by supplier nation and year.

SQL Query:

```
select
  nation,
  o_year,
  sum(amount) as sum_profit
from
  (
    select
      n.n_name as nation,
      extract(year from o.o_orderdate) as o_year,
      l.l_extendedprice * (1 - l.l_discount) - ps.ps_supplycost *
l.l_quantity as amount
    from
      part p,
      supplier s,
      lineitem l,
      partsupp ps,
      orders o,
      nation n
    where
      s.s_suppkey = l.l_suppkey
      and ps.ps_suppkey = l.l_suppkey
      and ps.ps_partkey = l.l_partkey
      and p.p_partkey = l.l_partkey
      and o.o_orderkey = l.l_orderkey
      and s.s_nationkey = n.n_nationkey
      and p.p_name like '%yellow%'
  ) as profit
group by
  nation,
  o_year
order by
  nation,
  o_year desc;
```

Output :



```
| RUSSIA         | 1992    | 5201604.268300001  |
| SAUDI ARABIA   | 1998    | 2628300.4048999995 |
| SAUDI ARABIA   | 1997    | 4295171.926399998  |
| SAUDI ARABIA   | 1996    | 4369616.0724       |
| SAUDI ARABIA   | 1995    | 4747363.708200002  |
| SAUDI ARABIA   | 1994    | 4892028.533799999  |
| SAUDI ARABIA   | 1993    | 4145382.3502       |
| SAUDI ARABIA   | 1992    | 5105260.729799999  |
| UNITED KINGDOM | 1998    | 2654163.2421999993 |
| UNITED KINGDOM | 1997    | 4499069.171199995  |
| UNITED KINGDOM | 1996    | 4455535.001700002  |
| UNITED KINGDOM | 1995    | 4791047.194799999  |
| UNITED KINGDOM | 1994    | 4412895.599299999  |
| UNITED KINGDOM | 1993    | 3721089.386900001  |
| UNITED KINGDOM | 1992    | 4128925.2254999997 |
| UNITED STATES  | 1998    | 1898808.8908000004 |
| UNITED STATES  | 1997    | 2376452.5546000004 |
| UNITED STATES  | 1996    | 3006432.1433       |
| UNITED STATES  | 1995    | 3277359.858799999  |
| UNITED STATES  | 1994    | 3303073.6643       |
| UNITED STATES  | 1993    | 2843990.0993999992 |
| UNITED STATES  | 1992    | 2864095.9757999973 |
| VIETNAM        | 1998    | 2035950.3558000003 |
| VIETNAM        | 1997    | 2900890.9689       |
| VIETNAM        | 1996    | 5061377.855800003  |
| VIETNAM        | 1995    | 3772979.718500001  |
| VIETNAM        | 1994    | 4609148.4268000005 |
| VIETNAM        | 1993    | 4706495.5490000015 |
| VIETNAM        | 1992    | 5123389.7845       |
+----------------+---------+--------------------+
175 rows selected (199.949 seconds)
```

Execution Time: 199.949 sec

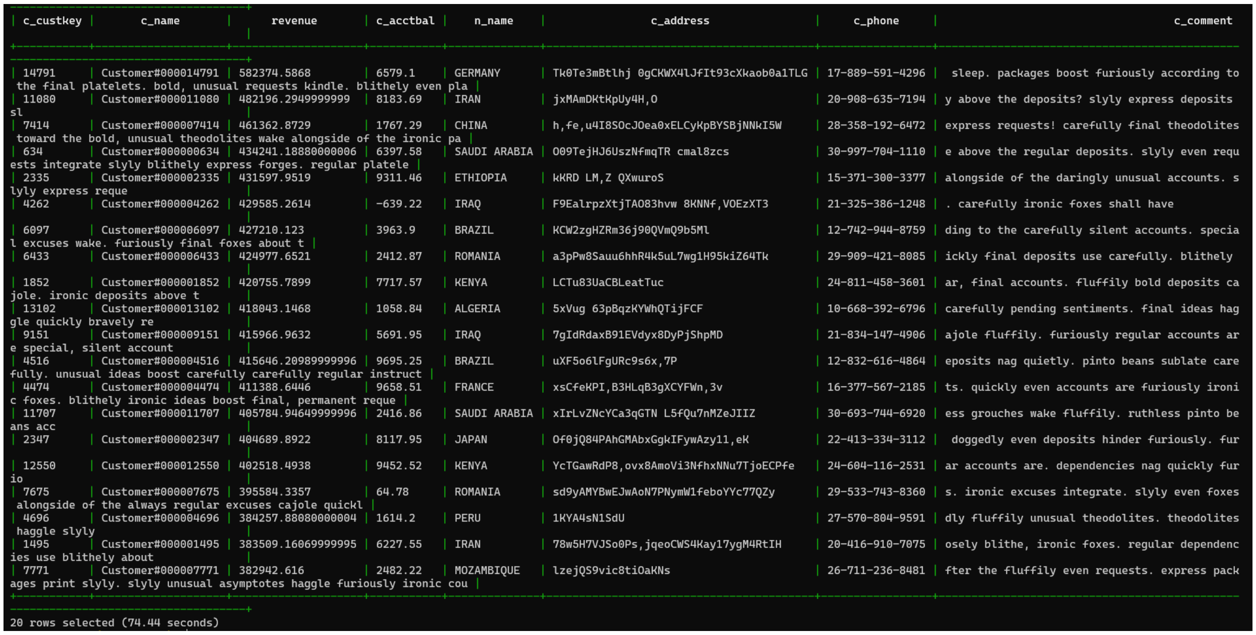
4.2.10 Returned Item Reporting Query (Q10)

Statement:
The query identifies customers who might be having problems with the parts that are shipped to them.

SQL Query:

```sql
select
  c.c_custkey,
  c.c_name,
  sum(l.l_extendedprice * (1 - l.l_discount)) as revenue,
  c.c_acctbal,
  n.n_name,
  c.c_address,
  c.c_phone,
  c.c_comment
from
  customer c,
  orders o,
  lineitem l,
  nation n
where
  c.c_custkey = o.o_custkey
  and l.l_orderkey = o.o_orderkey
  and o.o_orderdate >= date '1994-03-01'
  and o.o_orderdate < date '1994-03-01' + interval '3' month
  and l.l_returnflag = 'R'
  and c.c_nationkey = n.n_nationkey
group by
  c.c_custkey,
  c.c_name,
  c.c_acctbal,
  c.c_phone,
  n.n_name,
  c.c_address,
  c.c_comment
order by
  revenue desc
limit 20;
```

Output :



```
-------------------------------------+
| c_custkey |       c_name       |       revenue        | c_acctbal |   n_name     |                c_address                 |     c_phone     |                c_comment
                                     |
+-----------+--------------------+----------------------+-----------+--------------+------------------------------------------+-----------------+------------------------------------------
-------------------------------------+
| 14791     | Customer#000014791 | 582374.5868          | 6579.1    | GERMANY      | Tk0Te3mBtlhj 0gCKWX4lJfIt93cXkaob0a1TLG | 17-889-591-4296 |  sleep. packages boost furiously according to
 the final platelets. bold, unusual requests kindle. blithely even pla |
| 11080     | Customer#000011080 | 482196.2949999999    | 8183.69   | IRAN         | jxMAmDKtKpUy4H,O                         | 20-908-635-7194 | y above the deposits? slyly express deposits
sl                                   |
| 7414      | Customer#000007414 | 461362.8729          | 1767.29   | CHINA        | h,fe,u4I8SOcJOea0xELCyKpBYSBjNNkI5W      | 28-358-192-6472 | express requests! carefully final theodolites
 toward the bold, unusual theodolites wake alongside of the ironic pa |
| 634       | Customer#000000634 | 434241.18880000006   | 6397.58   | SAUDI ARABIA | O09TejHJ6UszNfmqTR cmal8zcs              | 30-997-704-1110 | e above the regular deposits. slyly even requ
ests integrate slyly blithely express forges. regular platele |
| 2335      | Customer#000002335 | 431597.9519          | 9311.46   | ETHIOPIA     | kKRD LM,Z QXwuroS                        | 15-371-300-3377 | alongside of the daringly unusual accounts. s
lyly express reque                   |
| 4262      | Customer#000004262 | 429585.2614          | -639.22   | IRAQ         | F9EalrpzXtjTAO83hvw 8KNNf,VOEzXT3        | 21-325-386-1248 | . carefully ironic foxes shall have
                                     |
| 6097      | Customer#000006097 | 427210.123           | 3963.9    | BRAZIL       | KCW2zgHZRm36j90QVmQ9b5Ml                 | 12-742-944-8759 | ding to the carefully silent accounts. specia
l excuses wake. furiously final foxes about t |
| 6433      | Customer#000006433 | 424977.6521          | 2412.87   | ROMANIA      | a3pPw8Sauu6hhR4k5uL7wg1H95kiZ64Tk        | 29-909-421-8085 | ickly final deposits use carefully. blithely
                                     |
| 1852      | Customer#000001852 | 420755.7899          | 7717.57   | KENYA        | LCTu83UaCBLeatTuc                        | 24-811-458-3601 | ar, final accounts. fluffily bold deposits ca
jole. ironic deposits above t        |
| 13102     | Customer#000013102 | 418043.1468          | 1058.84   | ALGERIA      | 5xVug 63pBqzKYWhQTijFCF                  | 10-668-392-6796 | carefully pending sentiments. final ideas hag
gle quickly bravely re               |
| 9151      | Customer#000009151 | 415966.9632          | 5691.95   | IRAQ         | 7gIdRdaxB91EVdyx8DyPjShpMD               | 21-834-147-4906 | ajole fluffily. furiously regular accounts ar
e special, silent account            |
| 4516      | Customer#000004516 | 415646.20989999996   | 9695.25   | BRAZIL       | uXF5o6lFgURc9s6x,7P                      | 12-832-616-4864 | eposits nag quietly. pinto beans sublate care
fully. unusual ideas boost carefully carefully regular instruct |
| 4474      | Customer#000004474 | 411388.6446          | 9658.51   | FRANCE       | xsCfeKPI,B3HLqB3gXCYFWn,3v               | 16-377-567-2185 | ts. quickly even accounts are furiously ironi
c foxes. blithely ironic ideas boost final, permanent reque |
| 11707     | Customer#000011707 | 405784.94649999996   | 2416.86   | SAUDI ARABIA | xIrLvZNcYCa3qGTN L5fQu7nMZeJIIZ          | 30-693-744-6920 | ess grouches wake fluffily. ruthless pinto be
ans acc                              |
| 2347      | Customer#000002347 | 404689.8922          | 8117.95   | JAPAN        | Of0jQ84PAhGMAbxGgkIFywAzy11,eK           | 22-413-334-3112 |  doggedly even deposits hinder furiously. fur
                                     |
| 12550     | Customer#000012550 | 402518.4938          | 9452.52   | KENYA        | YcTGawRdP8,ovx8AmoVi3NfhxNNu7TjoECPfe    | 24-604-116-2531 | ar accounts are. dependencies nag quickly fur
io                                   |
| 7675      | Customer#000007675 | 395584.3357          | 64.78     | ROMANIA      | sd9yAMYBwEJwAoN7PNymW1feboYYc77QZy       | 29-533-743-8360 | s. ironic excuses integrate. slyly even foxes
 alongside of the always regular excuses cajole quickl |
| 4696      | Customer#000004696 | 384257.88080000004   | 1614.2    | PERU         | 1KYA4sN1SdU                              | 27-570-804-9591 | dly fluffily unusual theodolites. theodolites
 haggle slyly                        |
| 1495      | Customer#000001495 | 383509.16069999995   | 6227.55   | IRAN         | 78w5H7VJSo0Ps,jqeoCWS4Kay17ygM4RtIH      | 20-416-910-7075 | osely blithe, ironic foxes. regular dependenc
ies use blithely about               |
| 7771      | Customer#000007771 | 382942.616           | 2482.22   | MOZAMBIQUE   | lzejQS9vic8tiOaKNs                       | 26-711-236-8481 | fter the fluffily even requests. express pack
ages print slyly. slyly unusual asymptotes haggle furiously ironic cou |
+-----------+--------------------+----------------------+-----------+--------------+------------------------------------------+-----------------+------------------------------------------
-------------------------------------+
20 rows selected (74.44 seconds)
```

Execution Time: 74.44 sec

4.2.11 Important Stock Identification Query (Q11)

Statement:

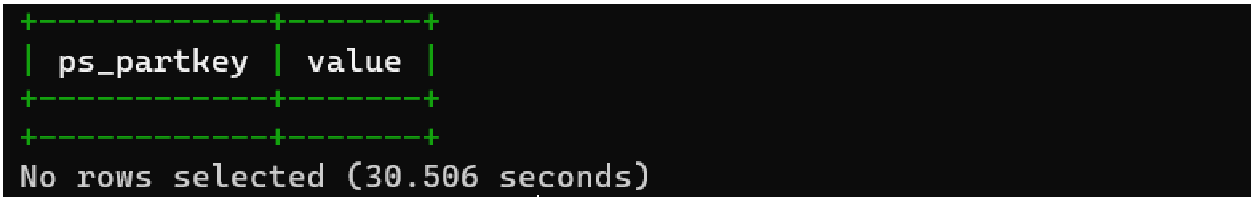
This query finds the most important subset of suppliers' stock in a given nation.

SQL Query:

```
select
  ps.ps_partkey,
  sum(ps.ps_supplycost * ps.ps_availqty) as `value`
from
  partsupp ps,
  supplier s,
  nation n
where
  ps.ps_suppkey = s.s_suppkey
  and s.s_nationkey = n.n_nationkey
  and n.n_name = 'JAPAN'
group by
  ps.ps_partkey having
    sum(ps.ps_supplycost * ps.ps_availqty) > (
      select
```

```
        sum(ps.ps_supplycost * ps.ps_availqty) * 0.0001 / 0.0001000000
      from
        partsupp ps,
        supplier s,
        nation n
      where
        ps.ps_suppkey = s.s_suppkey
        and s.s_nationkey = n.n_nationkey
        and n.n_name = 'JAPAN'
    )
order by
  `value` desc;
```

Output :



```
+------------+-------+
| ps_partkey | value |
+------------+-------+
+------------+-------+
No rows selected (30.506 seconds)
```

Execution Time: 30.506 sec

4.2.12 Shipping Modes and Order Priority Query (Q12)

Statement:

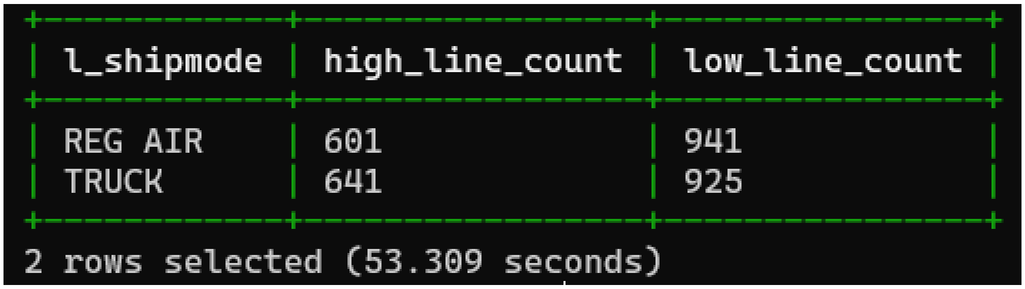
This query determines whether selecting less expensive modes of shipping is negatively affecting the critical-priority orders by causing more parts to be received by customers after the committed date.

SQL Query:

```
select
  l.l_shipmode,
  sum(case
    when o.o_orderpriority = '1-URGENT'
      or o.o_orderpriority = '2-HIGH'
      then 1
    else 0
  end) as high_line_count,
  sum(case
    when o.o_orderpriority <> '1-URGENT'
      and o.o_orderpriority <> '2-HIGH'
      then 1
    else 0
  end) as low_line_count
```

```
from
  orders o,
  lineitem l
where
  o.o_orderkey = l.l_orderkey
  and l.l_shipmode in ('TRUCK', 'REG AIR')
  and l.l_commitdate < l.l_receiptdate
  and l.l_shipdate < l.l_commitdate
  and l.l_receiptdate >= date '1994-01-01'
  and l.l_receiptdate < date '1994-01-01' + interval '1' year
group by
  l.l_shipmode
order by
  l.l_shipmode;
```

Output :



```
+------------+-----------------+----------------+
| l_shipmode | high_line_count | low_line_count |
+------------+-----------------+----------------+
| REG AIR    | 601             | 941            |
| TRUCK      | 641             | 925            |
+------------+-----------------+----------------+
2 rows selected (53.309 seconds)
```

Execution Time: 53.309 sec

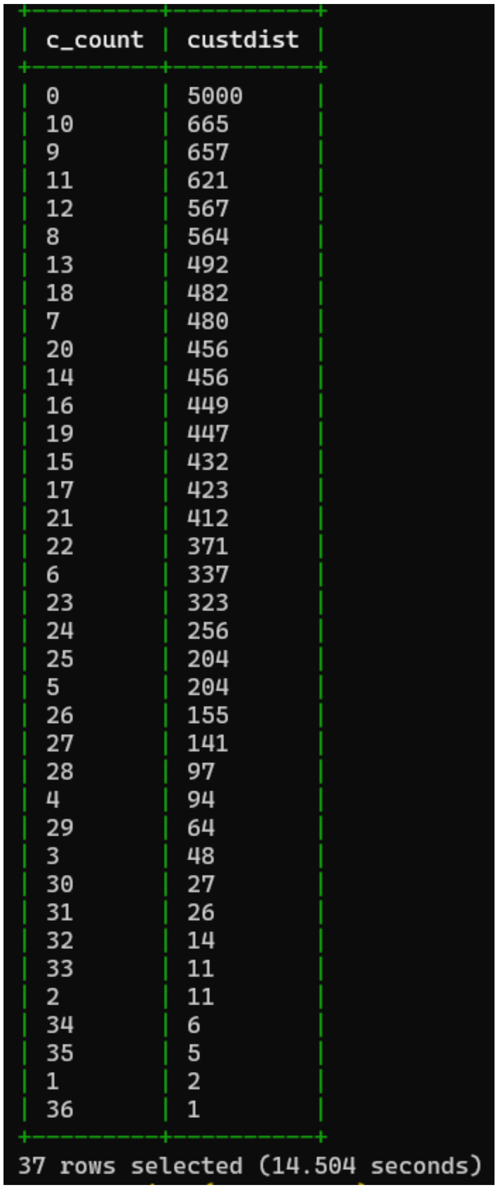
4.2.13 Customer Distribution Query (Q13)

Statement: This query seeks relationships between customers and the size of their orders.

Python SQL Query function:

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

Output :



```
+----------+----------+
| c_count  | custdist |
+----------+----------+
| 0        | 5000     |
| 10       | 665      |
| 9        | 657      |
| 11       | 621      |
| 12       | 567      |
| 8        | 564      |
| 13       | 492      |
| 18       | 482      |
| 7        | 480      |
| 20       | 456      |
| 14       | 456      |
| 16       | 449      |
| 19       | 447      |
| 15       | 432      |
| 17       | 423      |
| 21       | 412      |
| 22       | 371      |
| 6        | 337      |
| 23       | 323      |
| 24       | 256      |
| 25       | 204      |
| 5        | 204      |
| 26       | 155      |
| 27       | 141      |
| 28       | 97       |
| 4        | 94       |
| 29       | 64       |
| 3        | 48       |
| 30       | 27       |
| 31       | 26       |
| 32       | 14       |
| 33       | 11       |
| 2        | 11       |
| 34       | 6        |
| 35       | 5        |
| 1        | 2        |
| 36       | 1        |
+----------+----------+
37 rows selected (14.504 seconds)
```

Execution Time: 14.504 sec

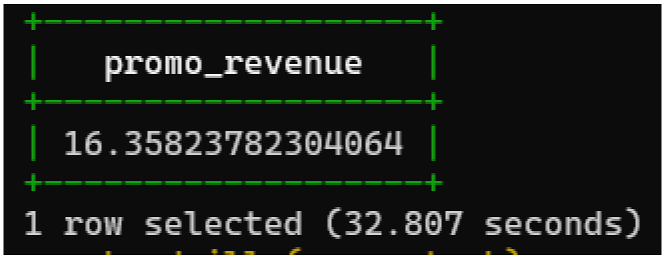
4.2.14 Promotion Effect Query (Q14)

Statement: This query monitors the market response to a promotion such as TV advertisements or a special campaign.

SQL Query:

```
select
  100.00 * sum(case
    when p.p_type like 'PROMO%'
      then l.l_extendedprice * (1 - l.l_discount)
    else 0
  end) / sum(l.l_extendedprice * (1 - l.l_discount)) as promo_revenue
from
  lineitem l,
  part p
where
  l.l_partkey = p.p_partkey
  and l.l_shipdate >= date '1994-08-01'
  and l.l_shipdate < date '1994-08-01' + interval '1' month;
```

Output :



```
+--------------------+
|   promo_revenue    |
+--------------------+
| 16.35823782304064 |
+--------------------+
1 row selected (32.807 seconds)
```

Execution Time: 32.807 sec

4.2.15 Top Supplier Query (Q15)

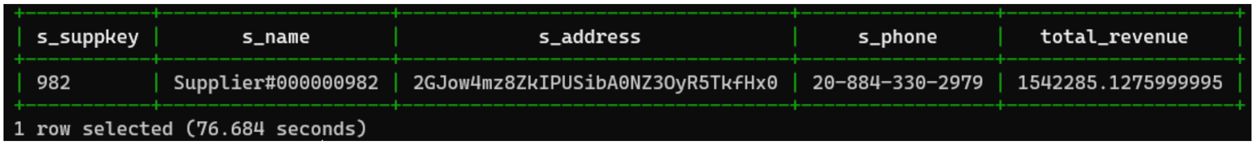
Statement: This query determines the top supplier so it can be rewarded, given more business, or identified for special recognition.

SQL Query:

```
with revenue0 (supplier_no, total_revenue) as
  (select
    l_suppkey,
    sum(l_extendedprice * (1 - l_discount))
  from
    lineitem
  where
    l_shipdate >= date '1993-05-01'
    and l_shipdate < date '1993-05-01' + interval '3' month
  group by
    l_suppkey)

select
  s.s_suppkey,
  s.s_name,
  s.s_address,
  s.s_phone,
  r.total_revenue
from
  supplier s,
  revenue0 r
where
  s.s_suppkey = r.supplier_no
  and r.total_revenue = (
    select
      max(total_revenue)
    from
      revenue0
  )
order by
  s.s_suppkey;
```

Output :



```
+-----------+--------------------+-----------------------------------+-----------------+--------------------+
| s_suppkey |       s_name       |             s_address             |     s_phone     |   total_revenue    |
+-----------+--------------------+-----------------------------------+-----------------+--------------------+
| 982       | Supplier#000000982 | 2GJow4mz8ZkIPUSibA0NZ3OyR5TkfHx0 | 20-884-330-2979 | 1542285.1275999995 |
+-----------+--------------------+-----------------------------------+-----------------+--------------------+
1 row selected (76.684 seconds)
```

Execution Time: 76.684 sec

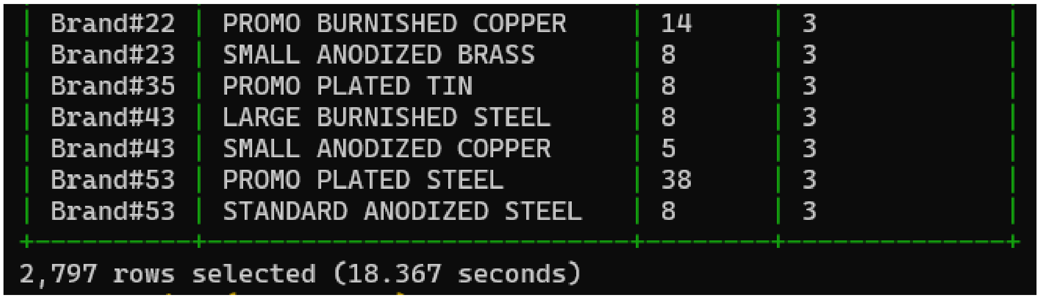
4.2.16 Parts/Supplier Relationship Query (Q16)

Statement: This query finds out how many suppliers can supply parts with given attributes. It might be used, for example, to determine whether there is a sufficient number of suppliers for heavily ordered parts.

SQL Query:

```
select
  p.p_brand,
  p.p_type,
  p.p_size,
  count(distinct ps.ps_suppkey) as supplier_cnt
from
  partsupp ps,
  part p
where
  p.p_partkey = ps.ps_partkey
  and p.p_brand <> 'Brand#21'
  and p.p_type not like 'MEDIUM PLATED%'
  and p.p_size in (38, 2, 8, 31, 44, 5, 14, 24)
  and ps.ps_suppkey not in (
    select
      s.s_suppkey
    from
      supplier s
    where
      s.s_comment like '%Customer%Complaints%'
  )
group by
  p.p_brand,
  p.p_type,
  p.p_size
order by
  supplier_cnt desc,
  p.p_brand,
  p.p_type,
  p.p_size;
```

Output :



```
| Brand#22 | PROMO BURNISHED COPPER    | 14      | 3              |
| Brand#23 | SMALL ANODIZED BRASS      | 8       | 3              |
| Brand#35 | PROMO PLATED TIN          | 8       | 3              |
| Brand#43 | LARGE BURNISHED STEEL     | 8       | 3              |
| Brand#43 | SMALL ANODIZED COPPER     | 5       | 3              |
| Brand#53 | PROMO PLATED STEEL        | 38      | 3              |
| Brand#53 | STANDARD ANODIZED STEEL   | 8       | 3              |
+----------+---------------------------+---------+----------------+
2,797 rows selected (18.367 seconds)
```

Execution Time: 18.367 sec

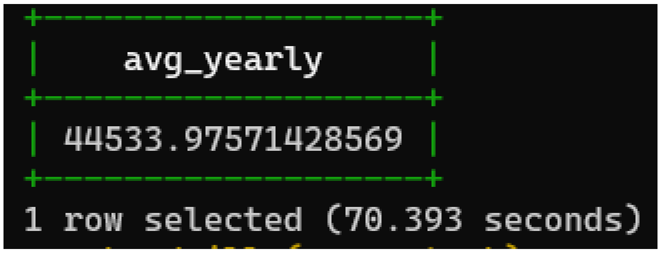
4.2.17 Small-Quantity-Order Revenue Query (Q17)

Statement: This query determines how much average yearly revenue would be lost if orders were no longer filled for small quantities of certain parts. This may reduce overhead expenses by concentrating sales on larger shipments.

SQL Query:

```sql
select
  sum(l.l_extendedprice) / 7.0 as avg_yearly
from
  lineitem l,
  part p
where
  p.p_partkey = l.l_partkey
  and p.p_brand = 'Brand#13'
  and p.p_container = 'JUMBO CAN'
  and l.l_quantity < (
    select
      0.2 * avg(l2.l_quantity)
    from
      lineitem l2
    where
      l2.l_partkey = p.p_partkey
  );
```

Output :



```
+--------------------+
|     avg_yearly     |
+--------------------+
| 44533.97571428569  |
+--------------------+
1 row selected (70.393 seconds)
```

Execution Time: 70.393 sec

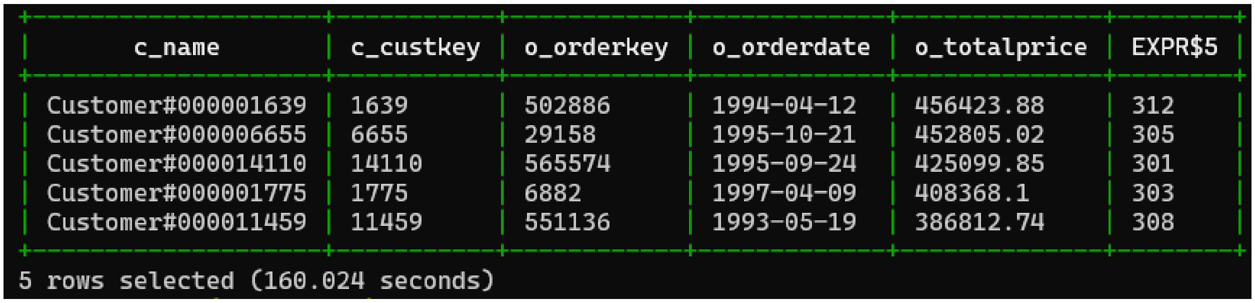
4.2.18 Large Volume Customer Query (Q18)

The Large Volume Customer Query ranks customers based on their having placed a large quantity order. Large quantity orders are defined as those orders whose total quantity is above a certain level.

SQL Query:

```sql
select
  c.c_name,
  c.c_custkey,
  o.o_orderkey,
  o.o_orderdate,
  o.o_totalprice,
  sum(l.l_quantity)
from
  customer c,
  orders o,
  lineitem l
where
  o.o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 300
  )
  and c.c_custkey = o.o_custkey
  and o.o_orderkey = l.l_orderkey
group by
  c.c_name,
  c.c_custkey,
  o.o_orderkey,
  o.o_orderdate,
  o.o_totalprice
order by
  o.o_totalprice desc,
  o.o_orderdate
limit 100;
```

Output :



```
+---------------------+-----------+------------+-------------+--------------+---------+
|       c_name        | c_custkey | o_orderkey | o_orderdate | o_totalprice | EXPR$5  |
+---------------------+-----------+------------+-------------+--------------+---------+
| Customer#000001639  | 1639      | 502886     | 1994-04-12  | 456423.88    | 312     |
| Customer#000006655  | 6655      | 29158      | 1995-10-21  | 452805.02    | 305     |
| Customer#000014110  | 14110     | 565574     | 1995-09-24  | 425099.85    | 301     |
| Customer#000001775  | 1775      | 6882       | 1997-04-09  | 408368.1     | 303     |
| Customer#000011459  | 11459     | 551136     | 1993-05-19  | 386812.74    | 308     |
+---------------------+-----------+------------+-------------+--------------+---------+
5 rows selected (160.024 seconds)
```

Execution Time: 160.024 sec

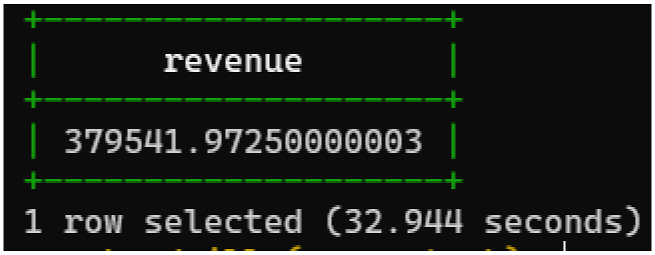
4.2.19 Discounted Revenue Query (Q19)

The Discounted Revenue Query reports the gross discounted revenue attributed to the sale of selected parts handled in a particular manner. This query is an example of code such as might be produced programmatically by a data mining tool.

SQL Query:

```
select
  sum(l.l_extendedprice* (1 - l.l_discount)) as revenue
from
  lineitem l,
  part p
where
-- Impala requires at least one conjunctive equality predicate.
-- Impala suggestion was to perform a Cartesian product between two tables,
use a CROSS JOIN
-- DRILL: Matching with Impala
    p.p_partkey = l.l_partkey
    and
  (
   (
    p.p_brand = 'Brand#41'
    and p.p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
    and l.l_quantity >= 2 and l.l_quantity <= 2 + 10
    and p.p_size between 1 and 5
    and l.l_shipmode in ('AIR', 'AIR REG')
    and l.l_shipinstruct = 'DELIVER IN PERSON'
  )
  or
  (
    p.p_brand = 'Brand#13'
    and p.p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
    and l.l_quantity >= 14 and l.l_quantity <= 14 + 10
    and p.p_size between 1 and 10
    and l.l_shipmode in ('AIR', 'AIR REG')
    and l.l_shipinstruct = 'DELIVER IN PERSON'
  )
  or
  (
    p.p_brand = 'Brand#55'
    and p.p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
    and l.l_quantity >= 23 and l.l_quantity <= 23 + 10
    and p.p_size between 1 and 15
    and l.l_shipmode in ('AIR', 'AIR REG')
    and l.l_shipinstruct = 'DELIVER IN PERSON'
    )
  );
```

Output :



```
+---------------------+
|       revenue       |
+---------------------+
| 379541.97250000003  |
+---------------------+
1 row selected (32.944 seconds)
```

Execution Time: 32.944 sec

4.2.20 Potential Part Promotion Query (Q20)

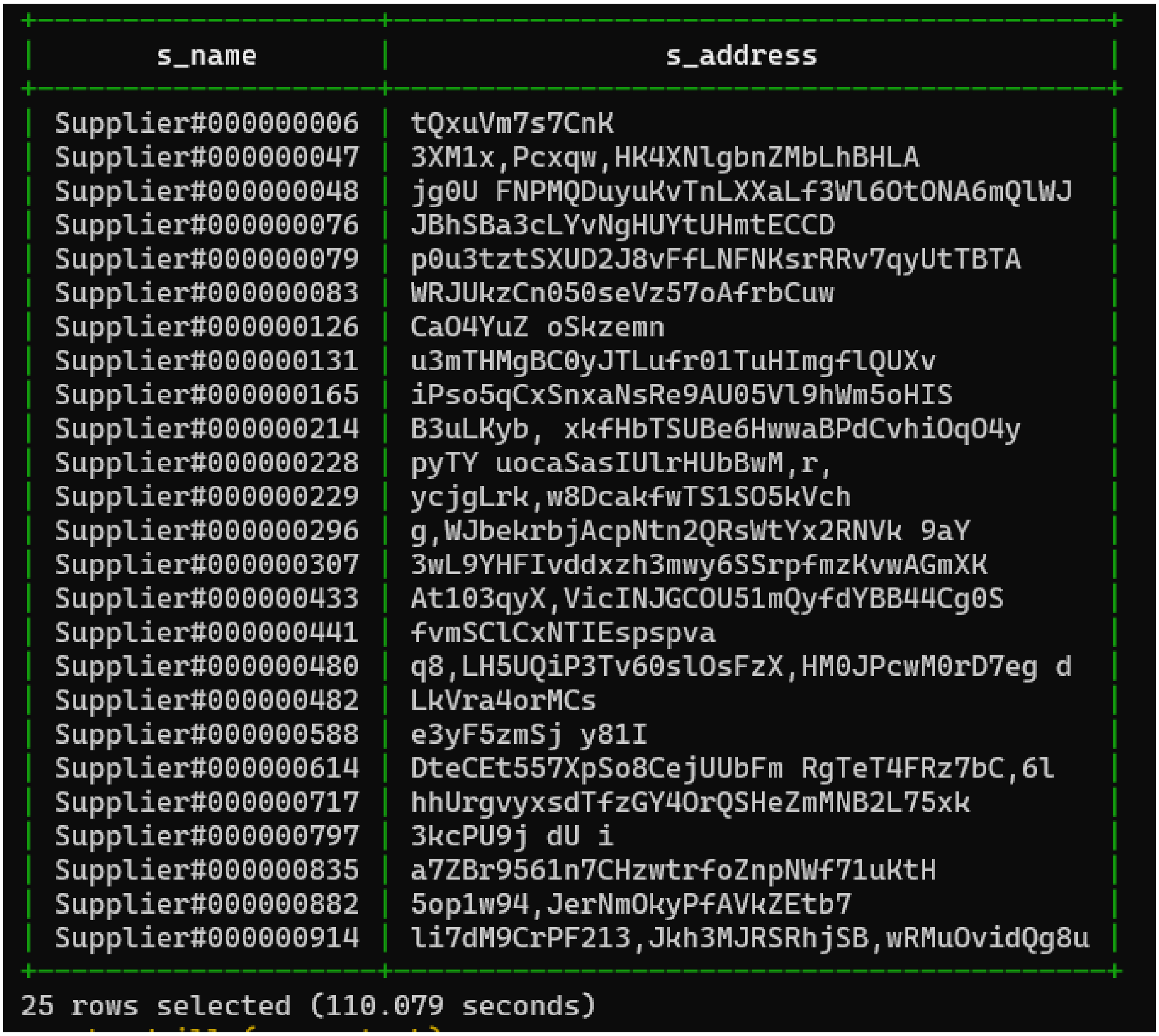
The Potential Part Promotion Query identifies suppliers in a particular nation having selected parts that may be candidates for a promotional offer.

Python SQL Query function:

```
select
  s.s_name,
  s.s_address
from
```

```
  supplier s,
  nation n
where
  s.s_suppkey in (
    select
      ps.ps_suppkey
    from
      partsupp ps
    where
      ps. ps_partkey in (
        select
          p.p_partkey
        from
          part p
        where
          p.p_name like 'antique%'
      )
      and ps.ps_availqty > (
        select
          0.5 * sum(l.l_quantity)
        from
          lineitem l
        where
          l.l_partkey = ps.ps_partkey
          and l.l_suppkey = ps.ps_suppkey
          and l.l_shipdate >= date '1993-01-01'
          and l.l_shipdate < date '1993-01-01' + interval '1' year
      )
  )
  and s.s_nationkey = n.n_nationkey
  and n.n_name = 'KENYA'
order by
  s.s_name;
```

Output :



```
+--------------------+------------------------------------------+
|       s_name       |                 s_address                |
+--------------------+------------------------------------------+
| Supplier#000000006 | tQxuVm7s7CnK                             |
| Supplier#000000047 | 3XM1x,Pcxqw,HK4XNlgbnZMbLhBHLA           |
| Supplier#000000048 | jg0U FNPMQDuyuKvTnLXXaLf3Wl6OtONA6mQlWJ  |
| Supplier#000000076 | JBhSBa3cLYvNgHUYtUHmtECCD                |
| Supplier#000000079 | p0u3tztSXUD2J8vFfLNFNKsrRRv7qyUtTBTA     |
| Supplier#000000083 | WRJUkzCn050seVz57oAfrbCuw                |
| Supplier#000000126 | CaO4YuZ oSkzemn                          |
| Supplier#000000131 | u3mTHMgBC0yJTLufr01TuHImgflQUXv          |
| Supplier#000000165 | iPso5qCxSnxaNsRe9AU05Vl9hWm5oHIS         |
| Supplier#000000214 | B3uLKyb, xkfHbTSUBe6HwwaBPdCvhiOqO4y     |
| Supplier#000000228 | pyTY uocaSasIUlrHUbBwM,r,                |
| Supplier#000000229 | ycjgLrk,w8DcakfwTS1SO5kVch               |
| Supplier#000000296 | g,WJbekrbjAcpNtn2QRsWtYx2RNVk 9aY        |
| Supplier#000000307 | 3wL9YHFIvddxzh3mwy6SSrpfmzKvwAGmXK       |
| Supplier#000000433 | At103qyX,VicINJGCOU51mQyfdYBB44Cg0S      |
| Supplier#000000441 | fvmSClCxNTIEspspva                       |
| Supplier#000000480 | q8,LH5UQiP3Tv60slOsFzX,HM0JPcwM0rD7eg d  |
| Supplier#000000482 | LkVra4orMCs                              |
| Supplier#000000588 | e3yF5zmSj y81I                           |
| Supplier#000000614 | DteCEt557XpSo8CejUUbFm RgTeT4FRz7bC,6l   |
| Supplier#000000717 | hhUrgvyxsdTfzGY4OrQSHeZmMNB2L75xk        |
| Supplier#000000797 | 3kcPU9j dU i                             |
| Supplier#000000835 | a7ZBr9561n7CHzwtrfoZnpNWf71uKtH          |
| Supplier#000000882 | 5op1w94,JerNmOkyPfAVkZEtb7               |
| Supplier#000000914 | li7dM9CrPF213,Jkh3MJRSRhjSB,wRMuOvidQg8u |
+--------------------+------------------------------------------+
25 rows selected (110.079 seconds)
```

Execution Time: 110.079 sec

4.2.21 Suppliers Who Kept Orders Waiting Query (Q21)

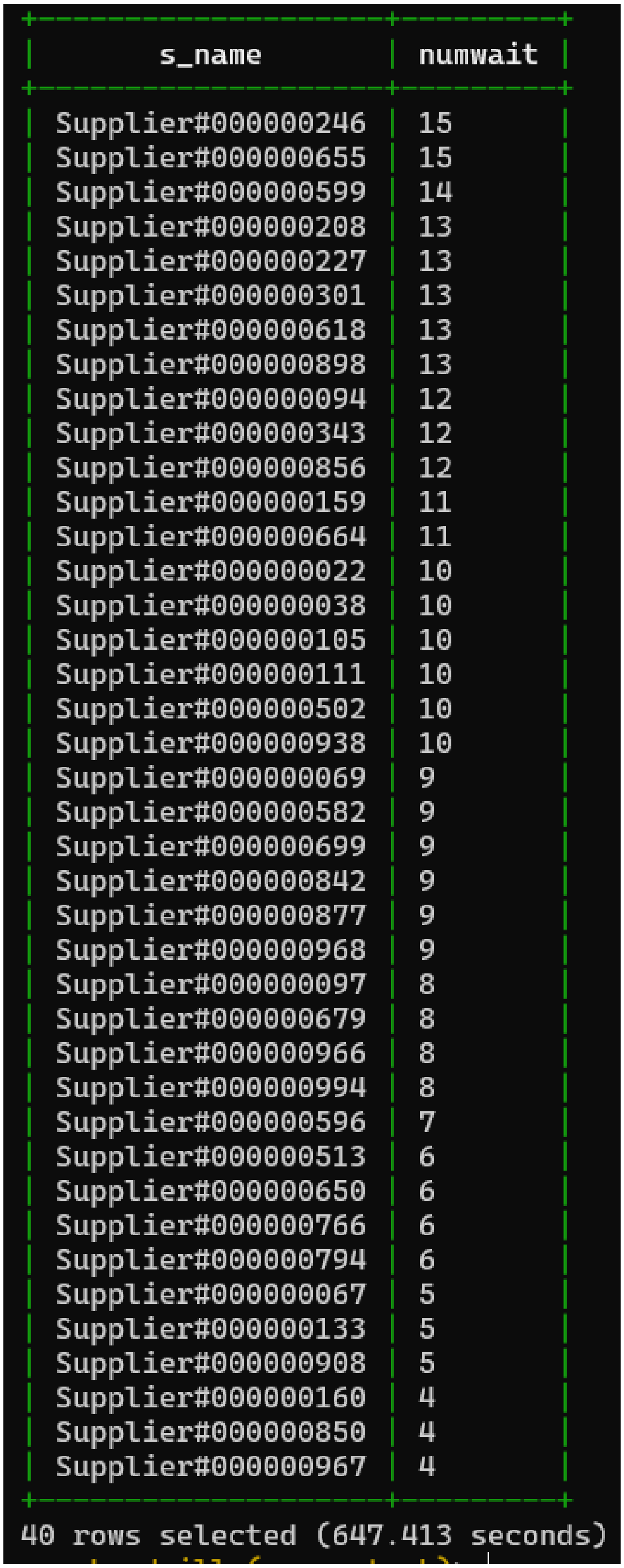
This query identifies certain suppliers who were not able to ship required parts in a timely manner.

SQL Query:

```
select
  s.s_name,
  count(*) as numwait
from
```

```sql
  supplier s,
  lineitem l1,
  orders o,
  nation n
where
  s.s_suppkey = l1.l_suppkey
  and o.o_orderkey = l1.l_orderkey
  and o.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.s_nationkey = n.n_nationkey
  and n.n_name = 'EGYPT'
group by
  s.s_name
order by
  numwait desc,
  s.s_name
limit
  100;
```

Output :



```
+----------------------+----------+
|        s_name        | numwait  |
+----------------------+----------+
| Supplier#000000246   | 15       |
| Supplier#000000655   | 15       |
| Supplier#000000599   | 14       |
| Supplier#000000208   | 13       |
| Supplier#000000227   | 13       |
| Supplier#000000301   | 13       |
| Supplier#000000618   | 13       |
| Supplier#000000898   | 13       |
| Supplier#000000094   | 12       |
| Supplier#000000343   | 12       |
| Supplier#000000856   | 12       |
| Supplier#000000159   | 11       |
| Supplier#000000664   | 11       |
| Supplier#000000022   | 10       |
| Supplier#000000038   | 10       |
| Supplier#000000105   | 10       |
| Supplier#000000111   | 10       |
| Supplier#000000502   | 10       |
| Supplier#000000938   | 10       |
| Supplier#000000069   | 9        |
| Supplier#000000582   | 9        |
| Supplier#000000699   | 9        |
| Supplier#000000842   | 9        |
| Supplier#000000877   | 9        |
| Supplier#000000968   | 9        |
| Supplier#000000097   | 8        |
| Supplier#000000679   | 8        |
| Supplier#000000966   | 8        |
| Supplier#000000994   | 8        |
| Supplier#000000596   | 7        |
| Supplier#000000513   | 6        |
| Supplier#000000650   | 6        |
| Supplier#000000766   | 6        |
| Supplier#000000794   | 6        |
| Supplier#000000067   | 5        |
| Supplier#000000133   | 5        |
| Supplier#000000908   | 5        |
| Supplier#000000160   | 4        |
| Supplier#000000850   | 4        |
| Supplier#000000967   | 4        |
+----------------------+----------+
40 rows selected (647.413 seconds)
```

Execution Time: 647.413 sec

4.2.22 Global Sales Opportunity Query (Q22)

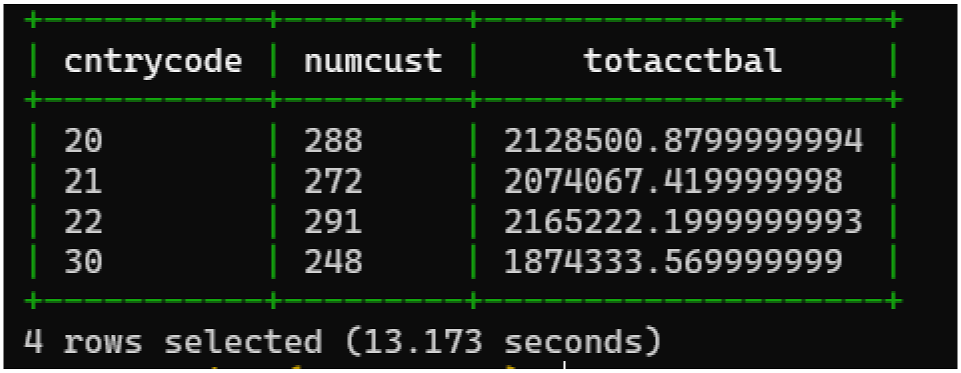
The Global Sales Opportunity Query identifies geographies where there are customers who may be likely to make a purchase.

SQL Query:

```
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 ('20', '40', '22', '30', '39', '42', '21')
      and c_acctbal > (
        select
          avg(c_acctbal)
        from
          customer
        where
          c_acctbal > 0.00
          and substring(
            c_phone
            from
              1 for 2
          ) in ('20', '40', '22', '30', '39', '42', '21')
      )
      and not exists (
        select
          *
        from
```

```
          orders
        where
          o_custkey = c_custkey
      )
  ) as custsale
group by
  cntrycode
order by
  cntrycode;
```

Output :



```
+------------+----------+---------------------+
| cntrycode  | numcust  |      totacctbal     |
+------------+----------+---------------------+
| 20         | 288      | 2128500.8799999994 |
| 21         | 272      | 2074067.419999998  |
| 22         | 291      | 2165222.1999999993 |
| 30         | 248      | 1874333.569999999  |
+------------+----------+---------------------+
4 rows selected (13.173 seconds)
```

Execution Time: 13.173 sec