**CSED 516 Homework 1 Oct 13, 2019 Mengying(Monique) Bi**

Turing in:

* SQL for the queries
* Runtime for each query
* Number of rows returned
* First two rows from the result set (or all rows if a query returns fewer than 2 rows)
* A brief discussion of the query runtimes that you observed in different settings

1. Setting up Amazon Redshift (0 points)

2. Ingest Data into Amazon Redshift (0 points)

3. Run Queries

Run each query listed below multiple times.

Plot the average and either min/max or standard deviation.

Use the warm cache timing, which means you discard the first time the query is run.

Go to the Query tab on the AWS web console for your redshift cluster to view the runtime of the queries.

**Question 1 (1GB 2 Nodes)**

-- 1.1

-- What is the total number of parts offered by each supplier?

-- The query should return the name of the supplier and the total number of parts.

SELECT s.s\_name AS supplier\_name, SUM(ps.ps\_availqty) AS Total\_part\_num

FROM supplier AS s, partsupp AS ps

WHERE ps.ps\_suppkey = s.s\_suppkey

GROUP BY supplier\_name;

Result:

|  |  |
| --- | --- |
| supplier\_name | total\_part\_num |
| Supplier#000007502 | 420818 |
| Supplier#000007509 | 421890 |

Number of Rows: 100000

-- 1.2

-- What is the cost of the most expensive part by any supplier?

-- The query should return only the price of that most expensive part. No need to return the name.

SELECT MAX(p.p\_retailprice) AS max\_price

FROM partsupp AS ps, part AS p

WHERE ps.ps\_partkey = p.p\_partkey;

Result:

|  |
| --- |
| max\_price |
| 2098.99 |

Number of Rows: 1

-- 1.3

-- What is the cost of the most expensive part for each supplier?

-- The query should return the name of the supplier and the cost of the most

-- expensive part but you do not need to return the name of that part.

SELECT s.s\_name AS supplier\_name, MAX(p.p\_retailprice) AS max\_price

FROM supplier AS s, partsupp AS ps, part AS p

WHERE ps.ps\_suppkey = s.s\_suppkey AND ps.ps\_partkey = p.p\_partkey

GROUP BY supplier\_name;

Result:

|  |  |
| --- | --- |
| supplier\_name | max\_price |
| Supplier#000007502 | 2076.98 |
| Supplier#000007509 | 2083.98 |

Number of Rows: 100000

-- 1.4

-- What is the total number of customers per nation?

-- The query should return the name of the nation and the number of unique customers.

SELECT n.n\_name AS nation, count(c.c\_custkey) AS total\_customer\_num

FROM customer AS c, nation AS n

WHERE n.n\_nationkey = c.c\_nationkey

GROUP BY nation;

Result:

|  |  |
| --- | --- |
| nation | total\_customer\_num |
| MOZAMBIQUE | 5974 |
| ROMANIA | 6100 |

Number of Rows: 25

-- 1.5

-- What is number of parts shipped between 10 oct, 1996 and 10 nov, 1996 for each supplier?

-- The query should return the name of the supplier and the number of parts

SELECT s.s\_name AS supplier\_name, SUM(l.l\_quantity) AS num\_of\_parts

FROM lineitem AS l, supplier AS s

WHERE l.l\_suppkey= s.s\_suppkey AND (l.l\_shipdate >= '1996-10-10 00:00:00'

AND l.l\_shipdate < '1996-11-11 00:00:00')

GROUP BY supplier\_name;

Result:

|  |  |
| --- | --- |
| supplier\_name | num\_of\_parts |
| Supplier#000006515 | 256 |
| Supplier#000002290 | 229 |

Number of Rows: 99966

**Question 2 (10GB 2 Nodes)**

Change database to 10GB and then run the same queries as in Question 1.

DELETE FROM customer;

DELETE FROM lineitem;

DELETE FROM nation;

DELETE FROM orders;

DELETE FROM part;

DELETE FROM partsupp;

DELETE FROM region;

DELETE FROM supplier;

copy customer from 's3://uwdb/tpch/uniform/10GB/customer.tbl' REGION 'us-west-2'CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy orders from 's3://uwdb/tpch/uniform/10GB/orders.tbl' REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.1'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.2'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.3'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.4'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.5'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.6'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.7'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.8'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.9'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/10GB-split/lineitem.tbl.10'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy nation from 's3://uwdb/tpch/uniform/10GB/nation.tbl'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy part from 's3://uwdb/tpch/uniform/10GB/part.tbl'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy partsupp from 's3://uwdb/tpch/uniform/10GB/partsupp.tbl'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy region from 's3://uwdb/tpch/uniform/10GB/region.tbl'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

copy supplier from 's3://uwdb/tpch/uniform/10GB/supplier.tbl'REGION 'us-west-2' CREDENTIALS 'aws\_iam\_role=arn:aws:iam::295193264972:role/RedshiftRole' delimiter '|';

Query 1:

Result:

|  |  |
| --- | --- |
| supplier\_name | total\_part\_num |
| Supplier#000050002 | 410759 |
| Supplier#000050005 | 390050 |

Number of Rows: 100000

Query 2:

Result:

|  |
| --- |
| max\_price |
| 2098.99 |

Number of Rows: 1

Query 3:

Result:

|  |  |
| --- | --- |
| supplier\_name | max\_price |
| Supplier#000075003 | 2098.99 |
| Supplier#000075007 | 2096.98 |

Number of Rows: 100000

Query 4:

Result:

|  |  |
| --- | --- |
| nation | total\_customer\_num |
| MOZAMBIQUE | 59796 |
| ROMANIA | 60048 |

Number of Rows: 25

Query 5:

Result:

|  |  |
| --- | --- |
| supplier\_name | num\_of\_parts |
| Supplier#000000008 | 70 |
| Supplier#000000012 | 42 |

Number of Rows: 99964

**Question 3 (10GB 4 Nodes)**

Resize the cluster to 4 nodes and then run the same queries as in Question 1.

Query 1:

Result:

|  |  |
| --- | --- |
| supplier\_name | total\_part\_num |
| Supplier#000025010 | 400897 |
| Supplier#000025011 | 413553 |

Number of Rows: 100000

Query 2:

Result:

|  |
| --- |
| max\_price |
| 2098.99 |

Number of Rows: 1

Query 3:

Result:

|  |  |
| --- | --- |
| supplier\_name | max\_price |
| Supplier#000075004 | 2093.98 |
| Supplier#000075013 | 2096.97 |

Number of Rows: 100000

Query 4:

Result:

|  |  |
| --- | --- |
| nation | total\_customer\_num |
| IRAN | 60101 |
| RUSSIA | 60065 |

Number of Rows: 25

Query 5:

Result:

|  |  |
| --- | --- |
| supplier\_name | num\_of\_parts |
| Supplier#000000013 | 150 |
| Supplier#000000105 | 305 |

Number of Rows: 99964

**Question 4 (1GB 2 Nodes)**

-- A customer is considered a Gold customer if they have orders totalling more than $1,000,000.00.

-- Customers with orders totalling between $1,000,000.00 and $500,000.00 are considered Silver.

-- Write a SQL query to compute the number of customers in these two categories.

-- Try different methods of writing the query (only SQL or use a UDF or a View to categorize a user).

-- Discuss your experience with the various methods to carry out such analysis.

-- Use the 1GB data set and the 2-node cluster. (10 points)

**-- SQL Only Method**

SELECT Category, COUNT(customer) as num\_of\_customer

FROM (

SELECT CASE WHEN SUM( o\_totalprice )> 1000000.00 THEN 'gold\_member'

WHEN SUM(o\_totalprice ) <= 1000000.00 AND SUM(o\_totalprice ) > 500000.00 THEN 'silver\_member'

ELSE NULL

END AS Category, c\_custkey AS customer

FROM orders, customer

WHERE o\_custkey = c\_custkey

GROUP BY customer)

WHERE Category is not NULL

GROUP BY Category;

Result:

|  |  |
| --- | --- |
| category | num\_of\_customer |
| silver\_member | 8054 |
| gold\_member | 91037 |

Number of Rows: 2

**-- Create a view method**

DROP VIEW membership;

CREATE VIEW membership AS

SELECT c\_custkey AS customer,

CASE WHEN SUM(o\_totalprice) > 1000000.00 THEN 'gold\_member'

WHEN SUM(o\_totalprice) > 500000.00 AND SUM(o\_totalprice) < 1000000.00 THEN 'silver\_member'

ELSE NULL

END AS Category

FROM customer, orders

WHERE o\_custkey = c\_custkey

GROUP BY c\_custkey;

SELECT Category, COUNT(customer) AS number\_of\_customer

FROM membership

WHERE Category is not NULL

GROUP BY Category;

Result:

|  |  |
| --- | --- |
| category | number\_of\_customer |
| silver\_member | 8054 |
| gold\_member | 91037 |

Number of Rows: 2

**-- UDF method**

CREATE FUNCTION membership(float)

returns VARCHAR

stable

AS $$

SELECT CASE WHEN $1 > 1000000.00 THEN 'gold\_member'

WHEN $1 > 500000.00 AND $1 <= 1000000.00 THEN 'silver\_member'

ELSE NULL

END

$$ language sql;

SELECT Category, COUNT(customer) AS num\_of\_customer

FROM (SELECT membership(SUM(o\_totalprice)) AS Category, c\_custkey AS customer

FROM orders, customer

WHERE o\_custkey = c\_custkey

GROUP BY customer)

WHERE Category IS NOT NULL

GROUP BY Category;

Result:

|  |  |
| --- | --- |
| category | num\_of\_customer |
| silver\_member | 8054 |
| gold\_member | 91037 |

Number of Rows: 2

Runtime:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q4 (in ms) | 1 | 2 | 3 | 4 | 5 | 6 | avg | std |
| Only SQL Method | 0.044 | 0.046 | 0.043 | 0.042 | 0.037 | 0.04 | 0.042 | 0.003 |
| Create a view method | 0.035 | 0.036 | 0.041 | 0.033 | 0.032 | 0.034 | 0.035 | 0.003 |
| UDF method | 344.214 | 361.985 | 372.957 | 357.799 | 352.456 | 376.15 | 360.927 | 12.153 |

**Discussion:**

Using “create a view method” is the method with the shortest runtime, followed by using SQL only method. The runtime doesn’t differ much from each other. However, the User Defined Function method takes way longer, almost 10\*\*4 times of the other two. On the other hand, the code structures of the “view method” and “UDF method” are clearer than using “SQL only” method. Considering the two factors, I would choose to use the “view method” prior to trying other methods.

**Question 5 (10GB 2 Nodes External)**

Connect to external database, and then run queries as in Question 1 (needs to add permission ‘Glue’ to IAM role).

create external schema tpchs3

from data catalog

database 'tpchs3'

iam\_role 'arn:aws:iam::295193264972:role/RedshiftRole'

create external database if not exists;

create external table tpchs3.customer(

C\_CustKey int ,

C\_Name varchar(64) ,

C\_Address varchar(64) ,

C\_NationKey int ,

C\_Phone varchar(64) ,

C\_AcctBal decimal(13, 2) ,

C\_MktSegment varchar(64) ,

C\_Comment varchar(120) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/customer/'

table properties ('numRows'='1500000');

create external table tpchs3.lineitem(

L\_OrderKey int ,

L\_PartKey int ,

L\_SuppKey int ,

L\_LineNumber int ,

L\_Quantity int ,

L\_ExtendedPrice decimal(13, 2) ,

L\_Discount decimal(13, 2) ,

L\_Tax decimal(13, 2) ,

L\_ReturnFlag varchar(64) ,

L\_LineStatus varchar(64) ,

L\_ShipDate datetime ,

L\_CommitDate datetime ,

L\_ReceiptDate datetime ,

L\_ShipInstruct varchar(64) ,

L\_ShipMode varchar(64) ,

L\_Comment varchar(64) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/lineitem/'

table properties ('numRows'='65984650');

create external table tpchs3.nation(

N\_NationKey int ,

N\_Name varchar(64) ,

N\_RegionKey int ,

N\_Comment varchar(160) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/nation/'

table properties ('numRows'='25');

create external table tpchs3.orders(

O\_OrderKey int ,

O\_CustKey int ,

O\_OrderStatus varchar(64) ,

O\_TotalPrice decimal(13, 2) ,

O\_OrderDate datetime ,

O\_OrderPriority varchar(15) ,

O\_Clerk varchar(64) ,

O\_ShipPriority int ,

O\_Comment varchar(80) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/orders/'

table properties ('numRows'='15000000');

create external table tpchs3.part(

P\_PartKey int ,

P\_Name varchar(64) ,

P\_Mfgr varchar(64) ,

P\_Brand varchar(64) ,

P\_Type varchar(64) ,

P\_Size int ,

P\_Container varchar(64) ,

P\_RetailPrice decimal(13, 2) ,

P\_Comment varchar(64) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/part/'

table properties ('numRows'='2000000');

create external table tpchs3.partsupp(

PS\_PartKey int ,

PS\_SuppKey int ,

PS\_AvailQty int ,

PS\_SupplyCost decimal(13, 2) ,

PS\_Comment varchar(200) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/partsupp/'

table properties ('numRows'='8000000');

create external table tpchs3.region(

R\_RegionKey int ,

R\_Name varchar(64) ,

R\_Comment varchar(160) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/region/'

table properties ('numRows'='5');

create external table tpchs3.supplier(

S\_SuppKey int ,

S\_Name varchar(64) ,

S\_Address varchar(64) ,

S\_NationKey int ,

S\_Phone varchar(18) ,

S\_AcctBal decimal(13, 2) ,

S\_Comment varchar(105) ,

skip varchar(64))

row format delimited

fields terminated by '|'

stored as textfile

location 's3://uwdb/tpch/athena/supplier/'

table properties ('numRows'='100000');

Query 1:

Result:

|  |  |
| --- | --- |
| supplier\_name | total\_part\_num |
| Supplier#000060929 | 378426 |
| Supplier#000085933 | 421681 |

Number of Rows: 100000

Query 2:

Result:

|  |
| --- |
| max\_price |
| 2098.99 |

Number of Rows: 1

Query 3:

Result:

|  |  |
| --- | --- |
| supplier\_name | max\_price |
| Supplier#000037469 | 1553.46 |
| Supplier#000062470 | 1553.46 |

Number of Rows: 100000

Query 4:

Result:

|  |  |
| --- | --- |
| nation | total\_customer\_num |
| PERU | 59788 |
| CANADA | 59849 |

Number of Rows: 25

Query 5:

Result:

|  |  |
| --- | --- |
| supplier\_name | num\_of\_parts |
| Supplier#000065196 | 193 |
| Supplier#000075232 | 230 |

Number of Rows: 99964

**Discussion:**

Runtime charts & plots:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query 1 (in ms) | 1 | 2 | 3 | 4 | 5 | 6 | avg | std |
| 1GB 2 Nodes | 7.111 | 7.195 | 7.131 | 7.073 | 7.099 | 6.99 | 7.100 | 0.068 |
| 10GB 2nodes | 1962 | 1804 | 1353 | 2154 | 2025 | 1506 | 1800.667 | 312.552 |
| 10GB 4 nodes | 4677 | 4782 | 4384 | 5364 | 2715 | 3958 | 4313.333 | 910.135 |
| 10GB S3 2nodes | 3415 | 3151 | 3030 | 3340 | 3102 | 3168 | 3201.000 | 146.798 |
| 10GB S3 4nodes | 3556 | 3149 | 3435 | 3263 | 4272 | 5426 | 3850.167 | 866.775 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query 2 (in ms) | 1 | 2 | 3 | 4 | 5 | 6 | avg | std |
| 1GB 2 Nodes | 0.031 | 0.03 | 0.031 | 0.032 | 0.032 | 0.03 | 0.031 | 0.001 |
| 10GB 2nodes | 0.034 | 0.034 | 0.029 | 0.058 | 0.032 | 0.029 | 0.036 | 0.011 |
| 10GB 4 nodes | 0.027 | 0.024 | 0.026 | 0.028 | 0.026 | 0.026 | 0.026 | 0.001 |
| 10GB S3 2nodes | 2919 | 2843 | 2859 | 4132 | 2967 | 3103 | 3137.167 | 496.251 |
| 10GB S3 4nodes | 3926 | 3107 | 3803 | 3392 | 4123 | 3770 | 3686.833 | 371.749 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query 3 (in ms) | 1 | 2 | 3 | 4 | 5 | 6 | avg | std |
| 1GB 2 Nodes | 6.316 | 6.371 | 6.424 | 6.411 | 6.403 | 6.389 | 6.386 | 0.039 |
| 10GB 2nodes | 3728 | 3460 | 2826 | 3552 | 3592 | 2524 | 3280.333 | 486.227 |
| 10GB 4 nodes | 5525 | 4681 | 4155 | 4649 | 4612 | 4347 | 4661.500 | 469.909 |
| 10GB S3 2nodes | 4447 | 4133 | 4038 | 4378 | 4273 | 4400 | 4278.167 | 162.539 |
| 10GB S3 4nodes | 5025 | 4411 | 5655 | 6686 | 4798 | 5565 | 5356.667 | 801.989 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query 4 (in ms) | 1 | 2 | 3 | 4 | 5 | 6 | avg | std |
| 1GB 2 Nodes | 0.045 | 0.047 | 0.046 | 0.046 | 0.047 | 0.045 | 0.046 | 0.001 |
| 10GB 2nodes | 0.047 | 0.044 | 0.046 | 0.047 | 0.051 | 0.046 | 0.047 | 0.002 |
| 10GB 4 nodes | 0.042 | 0.042 | 0.054 | 0.044 | 0.041 | 0.042 | 0.044 | 0.005 |
| 10GB S3 2nodes | 1376 | 1595 | 1417 | 1698 | 1551 | 1398 | 1505.833 | 129.070 |
| 10GB S3 4nodes | 1703 | 2030 | 2091 | 1519 | 1766 | 1722 | 1805.167 | 215.885 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query 5 (in ms) | 1 | 2 | 3 | 4 | 5 | 6 | avg | std |
| 1GB 2 Nodes | 7.164 | 7.033 | 6.968 | 7.021 | 6.998 | 7.008 | 7.032 | 0.068 |
| 10GB 2nodes | 4472 | 5193 | 4447 | 5273 | 4726 | 3881 | 4665.333 | 519.923 |
| 10GB 4 nodes | 4604 | 4895 | 3700 | 4785 | 9903 | 4721 | 5434.667 | 2231.040 |
| 10GB S3 2nodes | 6066 | 5331 | 6414 | 5437 | 5528 | 5382 | 5693.000 | 442.605 |
| 10GB S3 4nodes | 7406 | 5694 | 6590 | 5184 | 5238 | 5799 | 5985.167 | 860.715 |

Discussion:

By observing the runtime table and plots, we observe there are huge increase for some of the queries when we 1) increase the size of the database, 2) create and connect to external tables.

For query 2 and 4, runtime doesn’t change a lot when switching the database from 1GB to 10 GB, or change from 2 nodes to 4 nodes. Primary reasons should be 1) they only have one join, 2) no complicated computation like sum, and 3) results have much less rows comparing to others.

For changing the cluster size from 2 nodes to 4 nodes, I was expecting to see a decrease in runtime, however in some of the cases, it has an increase instead. It’s probably because that the process of distributing data to different nodes is taking more time than increasing efficiency.

When switching local database to external, I experienced a huge jump in runtime. It’s most likely caused by the communication time fetching data externally.

Another observation is that runtime in the evening is sometimes shorter than during daytime. It is possible that cluster runs faster when it has larger capacity.