ASSESSMENT - 2

GOALS In this project, we will learn how to use snowflake as a query engine. We store our data in aws s3 and we will learn various methods to query it from snowflake.

A. Query data in s3 from snowflake.

B. Create view over data in aws s3.

C. Disadvantages and advantages of this approach.

2. PREPARATION

Create table: table is created in TEST warehouse, DEMO\_DB database, PUBLIC schema, CUSTOMER\_TEST table

--create a warehouse

CREATE WAREHOUSE TEST;

USE WAREHOUSE TEST;

--create a database

CREATE DATABASE DEMO\_DB;

USE DATABASE DEMO\_DB;

--create a transient table with values from another tables.

CREATE OR REPLACE TRANSIENT TABLE DEMO\_DB.PUBLIC.CUSTOMER\_TEST

AS

SELECT \* FROM

"SNOWFLAKE\_SAMPLE\_DATA"."TPCDS\_SF100TCL"."CUSTOMER";

AWS:

--create a role in aws:

AWS – IAM – ROLE –

CREATE ROLE – S3 – CREATE ROLE

ADD PERMISSIONS – S3FULLACCESS – NEXT – ROLE\_NAME – CREATE ROLE

--create a bucket in s3

AWS – S3 - CREATE BUCKET – BUCKET\_NAME – BLOCK PUBLIC ACCESS – ACKNOWLEGMENT – CREATE BUCKET

SELECT BUCKET – INSIDE BUCKET – create folder

--create a folder

CREATE FOLDER – FOLDER\_NAME – CREATE FOLDER

--integration object

CREATE OR REPLACE STORAGE INTEGRATION ASSESMENT

TYPE = EXTERNAL\_STAGE

STORAGE\_PROVIDER = S3

ENABLED = TRUE

STORAGE\_AWS\_ROLE\_ARN = 'ARN:AWS:IAM::880001414910:ROLE/PIPE23'

STORAGE\_ALLOWED\_LOCATIONS = ('S3://PIPE22-07/PIPECSV/');

--aws – iam – role – select role – summary – arn

-- aws – s3 – select bucket – select folder – objects – copy s3 url

--describe integration object

DESC INTEGRATION ASSESMENT;

When run this query, below is the output, in this copy 5 and 7 property\_values and change it accordingly in AWS – ROLE – TRUST RELATIONSHIP – EDIT POLICY.

A screenshot of a computer

Description automatically generated

--create csv format

CREATE OR REPLACE FILE FORMAT my\_csv\_format

TYPE = 'CSV'

FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"'

SKIP\_HEADER = 1;

--create external stage

CREATE OR REPLACE STAGE DEMO\_DB.PUBLIC.CSV\_EXT\_STAGE

STORAGE\_INTEGRATION = ASSESMENT

URL = 'S3://PIPE22-07/PIPECSV/'

FILE\_FORMAT = DEMO\_DB.PUBLIC.MY\_CSV\_FORMAT;

--copy into s3

COPY INTO @DEMO\_DB.public.csv\_ext\_stage/pipecsv/

from DEMO\_DB.PUBLIC.CUSTOMER\_TEST;

3. QUERY DATA IN S3 FROM SNOWFLAKE.

Now data got uploaded to s3. We have 100 Million records uploaded and data size is 4.5 GB. Uploaded files will be csv compressed files.

Let’s try to query this data in s3 from snowflake

SELECT $1 C\_CUSTOMER\_SK,

$2 C\_CUSTOMER\_ID ,

$3 C\_CURRENT\_CDEMO\_SK ,

$4 C\_CURRENT\_HDEMO\_SK ,

$5 C\_CURRENT\_ADDR\_SK,

$6 C\_FIRST\_SHIPTO\_DATE\_SK ,

$7 C\_FIRST\_SALES\_DATE\_SK ,

$8 C\_SALUTATION ,

$9 C\_FIRST\_NAME ,

$10 C\_LAST\_NAME,

$11 C\_PREFERRED\_CUST\_FLAG ,

$12 C\_BIRTH\_DAY ,

$13 C\_BIRTH\_MONTH ,

$14 C\_BIRTH\_YEAR,

$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @DEMO\_DB.public.csv\_ext\_stage/pipecsv/

(file\_format => DEMO\_DB.public.my\_csv\_format);

--filter data from s3, customer with customer\_key:

SELECT $1 C\_CUSTOMER\_SK,

$2 C\_CUSTOMER\_ID ,

$3 C\_CURRENT\_CDEMO\_SK ,

$4 C\_CURRENT\_HDEMO\_SK ,

$5 C\_CURRENT\_ADDR\_SK,

$6 C\_FIRST\_SHIPTO\_DATE\_SK ,

$7 C\_FIRST\_SALES\_DATE\_SK ,

$8 C\_SALUTATION ,

$9 C\_FIRST\_NAME ,

$10 C\_LAST\_NAME,

$11 C\_PREFERRED\_CUST\_FLAG ,

$12 C\_BIRTH\_DAY ,

$13 C\_BIRTH\_MONTH ,

$14 C\_BIRTH\_YEAR,

$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @DEMO\_DB.public.csv\_ext\_stage/pipecsv/ ---replace it with new stage

(file\_format => DEMO\_DB.public.my\_csv\_format)

WHERE C\_CUSTOMER\_SK ='64596949';

--execute group by

SELECT $9 C\_FIRST\_NAME,$10 C\_LAST\_NAME,COUNT(\*)

FROM @DEMO\_DB.public.csv\_ext\_stage/pipecsv/

(file\_format => DEMO\_DB.public.my\_csv\_format)

GROUP BY $9,$10;

4. CREATE A VIEW

CREATE OR REPLACE VIEW ASGN\_CUST\_DATA\_VIEW

AS

SELECT $1 C\_CUSTOMER\_SK,

$2 C\_CUSTOMER\_ID ,

$3 C\_CURRENT\_CDEMO\_SK ,

$4 C\_CURRENT\_HDEMO\_SK ,

$5 C\_CURRENT\_ADDR\_SK,

$6 C\_FIRST\_SHIPTO\_DATE\_SK ,

$7 C\_FIRST\_SALES\_DATE\_SK ,

$8 C\_SALUTATION ,

$9 C\_FIRST\_NAME ,

$10 C\_LAST\_NAME,

$11 C\_PREFERRED\_CUST\_FLAG ,

$12 C\_BIRTH\_DAY ,

$13 C\_BIRTH\_MONTH ,

$14 C\_BIRTH\_YEAR,

$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @DEMO\_DB.public.csv\_ext\_stage/pipecsv/

(file\_format => DEMO\_DB.public.my\_csv\_format);

--Query data directly on view

SELECT \* FROM ASGN\_CUST\_DATA\_VIEW;

Now we can directly query data from s3 through view. What is the disadvantage of using this approach? Can you see partitions being scanned in the backend?

1. **Performance Overload**
2. **Cost Implications**
3. **Latency**
4. **Complexity in Partitioning**

There are no partitions being scanned in the back when we query the data from view.

--Create a sample snowflake table

Create or replace transient table CUST\_SNOWFLAKE\_TABLE1

AS

SELECT \* FROM CUSTOMER\_TEST limit 10000;

--Join this with the view we created earlier

SELECT B.\*

FROM CUST\_SNOWFLAKE\_TABLE1 B

LEFT OUTER JOIN

ASGN\_CUST\_DATA\_VIEW A

ON

A.C\_CUSTOMER\_SK = B.C\_CUSTOMER\_SK;

Now we successfully joined data in s3 with snowflake table. It may look simple but this approach has lot of potential. Can you mention few below page and observe the execution plan. How many partitions got scanned from snowflake table:

1. **Cost Efficiency**
2. **Scalability**
3. **Data Integration**
4. **Performance Optimization**
5. **Data Management**

A screenshot of a statistics

Description automatically generated

5. UNLOAD DATA BACK TO S3

COPY INTO @DEMO\_DB.public.csv\_ext\_stage/Customer\_joined\_data/

from(

SELECT B.\*

FROM CUST\_SNOWFLAKE\_TABLE1 B

LEFT OUTER JOIN

ASGN\_CUST\_DATA\_VIEW A

ON

A.C\_CUSTOMER\_SK = B.C\_CUSTOMER\_SK

);

6. ADVANTAGES AND DISADVANTAGES

**Advantages:**

1. **Cost efficiency:** Storing large volumes of data in S3 can be more cost-effective than storing it in Snowflake. Snowflake charges based on the amount of data scanned, allowing for cost control.
2. **Scalability:** S3 offers virtually unlimited storage capacity, making it suitable for large datasets. Snowflake can dynamically scale compute resources to handle large queries.
3. **Data integration:** Enables querying across both Snowflake and S3 without data movement.
4. **Flexibility:** Supports a hybrid approach where active data resides in Snowflake and historical data in S3.

**Disadvantages**

1. **Performance overhead:** Queries on S3 data can be slower due to network latency and the overhead of reading from external storage.
2. **Cost implications**: Increased compute costs due to the additional resources required for querying external data.
3. **Complexity:** Managing and optimizing data partitioning in S3 is more complex compared to Snowflake's internal tables.

OUTPUTS:

1. Creating table

A close up of a message

Description automatically generated

1. Creating format

A close up of a sign

Description automatically generated

1. Creating integration object

A close up of a message

Description automatically generated

1. Description of integration object

A screenshot of a computer

Description automatically generated

1. Creating stage

A screenshot of a computer

Description automatically generated

1. Querying data from s3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Filter data on s3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

A screenshot of a computer

Description automatically generated

1. Using group by

A screenshot of a computer

Description automatically generated

1. Creating a view

A close up of a text

Description automatically generated

1. Sample snowflake table

A close up of a text

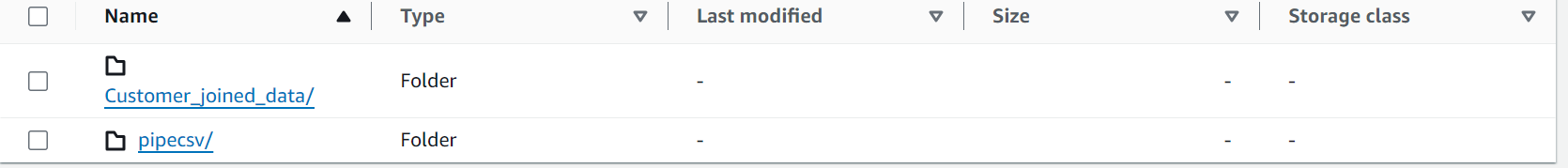
Description automatically generated

1. Creating join with view

A screenshot of a computer

Description automatically generated

Can see the folder created in s3.



When data gets uploaded in s3, it creates partitions with compressed gzip format.

A screenshot of a computer

Description automatically generated