1.GOALS

/\*In this project, we will learn how to use snowflake as a query engine.

We store our data from snowflake to 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.

USE ROLE ACCOUNTADMIN;

USE WAREHOUSE COMPUTE\_WH;

CREATE OR REPLACE DATABASE DEMO\_DB;

2.PREPARATION

/\*Before we start, let’s establish connection between snowflake and s3, then

upload some sample data from snowflake to s3. \*/

-- Create integration object with s3

create or replace storage integration S3\_INTEG\_CUS\_DATA\_CSV

type = external\_stage

storage\_provider = s3

enabled = true

storage\_aws\_role\_arn = 'arn:aws:iam::730335494857:role/aws-ext-bulkdata-pipeline-a2'

storage\_allowed\_locations = ('s3://snowflake-aws-ext-bulkdata-pipeline-a2/aws-cus-data-csv-a2/');

-- Create csv file format

create or replace file format demo\_db.public.cus\_data\_csv\_format

type = 'csv';

-- Create external stage object

create or replace stage demo\_db.public.cus\_data\_ext\_csv\_stage

URL = 's3://snowflake-aws-ext-bulkdata-pipeline-a2/aws-cus-data-csv-a2/'

STORAGE\_INTEGRATION = S3\_INTEG\_CUS\_DATA\_CSV

file\_format = demo\_db.public.cus\_data\_csv\_format;

-- Describe integration object to fetch external\_id and to be used in s3 Trust Partnership

DESC INTEGRATION S3\_INTEG\_CUS\_DATA\_CSV;

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

AS

SELECT \* FROM "SNOWFLAKE\_SAMPLE\_DATA"."TPCDS\_SF100TCL"."CUSTOMER";

-- Execute below copy command to copy snowflake customer data to s3,

COPY INTO

@demo\_db.public.cus\_data\_ext\_csv\_stage/aws-cus-data-csv-a2/

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.\*/

-- Unload data from s3, query in snowflake

-- Query-1d 01b5dae8-0001-27d8-0000-00054c2a23d9

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,

$15 C\_BIRTH\_COUNTRY,

$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @demo\_db.public.cus\_data\_ext\_csv\_stage/aws-cus-data-csv-a2/ ---replace it with new stage

(file\_format => demo\_db.public.cus\_data\_csv\_format);

-- Query ID: 01b5daef-0001-27da-0000-00054c2a3449

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,

$15 C\_BIRTH\_COUNTRY,

$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @demo\_db.public.cus\_data\_ext\_csv\_stage ---replace it with new stage

(file\_format => demo\_db.public.cus\_data\_csv\_format)

WHERE C\_CUSTOMER\_SK ='64596949';

-- Execute group by,

-- Query - 01b5daf4-0001-27d8-0000-00054c2a2441

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

FROM @demo\_db.public.cus\_data\_ext\_csv\_stage/

(file\_format => DEMO\_DB.PUBLIC.CUS\_DATA\_CSV\_FORMAT)

GROUP BY $9,$10;

4.CREATE VIEW OVER S3 DATA

CREATE OR REPLACE VIEW V\_CUSTOMER\_DATA

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,

$15 C\_BIRTH\_COUNTRY,

$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @demo\_db.public.cus\_data\_ext\_csv\_stage/aws-cus-data-csv-a2/ ---replace it with new stage

(file\_format => demo\_db.public.cus\_data\_csv\_format);

-- Query data directly on view,

select \* from V\_CUSTOMER\_DATA;

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 ?

View only look data from table and cannot store data itself. In this case,

V\_CUSTOMER\_DATA not scanning for partition.

Considering Query Performance, Since we are Querying from a simple view,

DML statements are allowed and data retrieval time minimum.

But in complex view, group functions for aggregations, Distinct, joins, group by can take more time than querying directly from the table.

Dependency & Maintenance also occurs because VIEWS are dependent on the underlying table structures and we cannot do SCHEMABINDING to prevent changes in main table.

We can query only SHOW VIEWS and SELECT GET\_DDL ('VIEW', 'V\_CUSTOMER\_DATA') function to tracking and managing schema changes.

/\*Now let’s try to Join the view we created with a table on snowflake\*/

-- Create a sample snowflake table as below

CREATE OR REPLACE TRANSIENT TABLE CUSTOMER\_SNOWFLAKE\_TRA\_TABLE

AS

SELECT \* FROM CUSTOMER\_TEST LIMIT 10000;

SELECT \* FROM CUSTOMER\_SNOWFLAKE\_TRA\_TABLE;

-- Join this with the view we created earlier,

-- Query - 01b5db05-0001-27da-0000-00054c2a3515

SELECT B.\* FROM CUSTOMER\_SNOWFLAKE\_TRA\_TABLE B

LEFT OUTER JOIN

V\_CUSTOMER\_DATA 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.

Initially, we established connection between snowflake and s3 using storage integration and uploaded CUSTOMER\_TEST data from snowflake to s3 with help of external stage.

In the above case, DEMO\_DB.CUSTOMER\_TEST data designed to query data from the perspective one-to-many relationship. So are fetch data from TRANSIENT table with limit 10000 and VIEW by using LEFT OUTER JOIN.

How many partitions got scanned from snowflake table : 355

5.UNLOAD DATA BACK TO S3

/\*This approach leverages micro partitions in snowflake for lookup table still giving

us the freedom to query data which we have stored in s3.

Once we are done looking up we can copy data back to s3 with new derived lookup column.\*/

-- We will try to query joined data in s3 and snowflake.\*/

-- Query - 01b5db0d-0001-27d8-0000-00054c2a2515

COPY INTO @demo\_db.public.cus\_data\_ext\_csv\_stage/aws-cus-joined-data-upload-csv-a2/

from

(

SELECT B.\* FROM CUSTOMER\_SNOWFLAKE\_TRA\_TABLE B

LEFT OUTER JOIN

V\_CUSTOMER\_DATA A ON

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

);

6.ADVANTAGES AND DISADVANTAGES

Advantages of using Snowflake querying from S3:

Scalability: Snowflake's cloud-based architecture allows it to scale up or down to handle large amounts of data from S3, making it an ideal choice for big data analytics & reporting.

Performance: Snowflake's columnar storage and parallel processing capabilities enable fast query performance, even on large datasets from S3.

Security: Snowflake provides robust security features, such as encryption and access controls, to ensure that data from S3 is protected during querying and analysis.

Flexibility: Snowflake supports a wide range of data formats and can handle semi-structured and unstructured data from S3, making it a versatile choice for data analytics.

Disadvantages of using Snowflake during querying from S3:

Cost: Snowflake can be a costly solution, especially for large-scale data analytics workloads that involve querying data from S3.

Complexity: Snowflake requires expertise in SQL and data warehousing, which can be a barrier for organizations without experienced personnel.

Data Ingestion: Snowflake requires data to be ingested from S3, which can be a time-consuming process, especially for large datasets.