BUILD A SNOWFLAKE BULK DATA PIPELINE USING AWS AND EXTERNAL STAGES

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

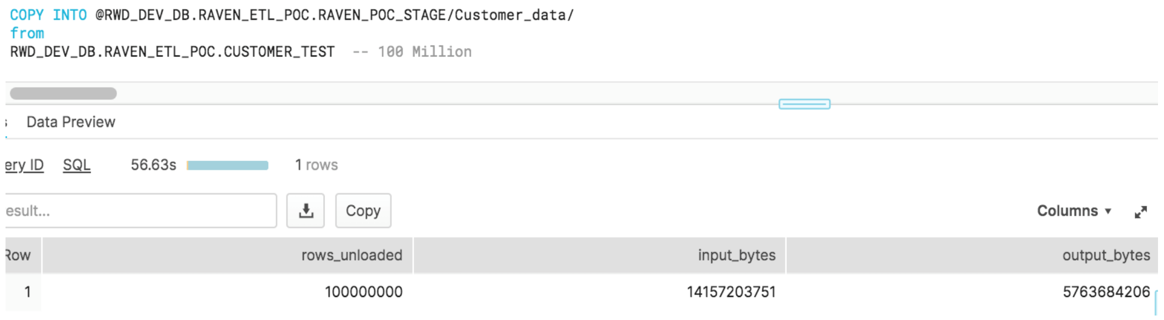
Before we start, let's upload some sample data from snowflake to s3. Then we will try to query data in s3 from snowflake.

Create table,

```
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 data to s3,

```
COPY INTO @DEMO_DB.PUBLIC.MY_S3_STAGE/Customer_data/
from
DEMO_DB.PUBLIC.CUSTOMER_TEST
```



```
COPY INTO @RWD_DEV_DB.RAVEN_ETL_POC.RAVEN_POC_STAGE/Customer_data/
from
RWD_DEV_DB.RAVEN_ETL_POC.CUSTOMER_TEST  -- 100 Million
```

Row	rows_unloaded	input_bytes	output_bytes
1	100000000	14157203751	5763684206

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.MY_S3_STAGE/Customer_data/.  ---replace it with new stage
(file_format => DEMO_DB.PUBLIC.MY_CSV_FORMAT)
```

Filter data directly from s3,

```
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.MY_S3_STAGE/Customer_data/
(file_format => DEMO_DB.PUBLIC.MY_CSV_FORMAT)
WHERE C_CUSTOMER_SK ='64596949'
```

Execute group by,

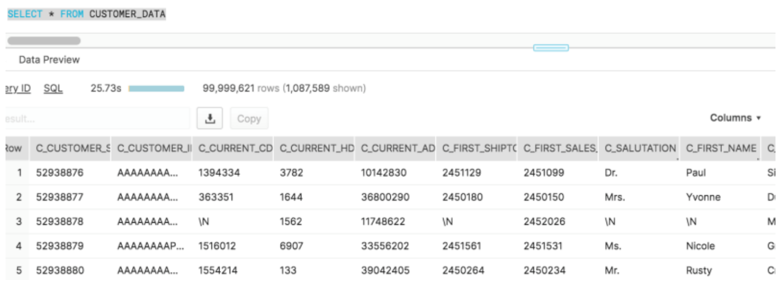
```
SELECT $9 C_FIRST_NAME,$10 C_LAST_NAME,COUNT(*)
FROM @DEMO_DB.PUBLIC.MY_S3_STAGE/Customer_data/
(file_format => DEMO_DB.PUBLIC.MY_CSV_FORMAT)
GROUP BY $9,$10
```

4. CREATE VIEW OVER S3 DATA

```
CREATE OR REPLACE VIEW 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,
$16 C_LOGIN          ,
$17 C_EMAIL_ADDRESS      ,
$18 C_LAST_REVIEW_DATE
FROM @DEMO_DB.PUBLIC.MY_S3_STAGE/Customer_data/
(file_format => DEMO_DB.PUBLIC.MY_CSV_FORMAT)
```

Query data directly on view,

SELECT * FROM CUSTOMER_DATA;



```
SELECT * FROM CUSTOMER_DATA
```

25.73s 99,999,621 rows (1,087,589 shown)

Row	C_CUSTOMER_S	C_CUSTOMER_I	C_CURRENT_CD	C_CURRENT_HD	C_CURRENT_AD	C_FIRST_SHIPT(	C_FIRST_SALES_	C_SALUTATION	C_FIRST_NAME	C
1	52938876	AAAAAAAA...	1394334	3782	10142830	2451129	2451099	Dr.	Paul	Si
2	52938877	AAAAAAAA...	363351	1644	36800290	2450180	2450150	Mrs.	Yvonne	D
3	52938878	AAAAAAAA...	\N	1562	11748622	\N	2452026	\N	\N	M
4	52938879	AAAAAAAAP...	1516012	6907	33556202	2451561	2451531	Ms.	Nicole	G
5	52938880	AAAAAAAA...	1554214	133	39042405	2450264	2450234	Mr.	Rusty	C

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 ?

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_TABLE
AS
SELECT * FROM CUSTOMER_TEST limit 10000
```

Join this with the view we created earlier,

```
SELECT B.*
FROM CUSTOMER_SNOWFLAKE_TABLE B
LEFT OUTER JOIN
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.

How many partitions got scanned from snowflake table :

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.

```
COPY INTO @DEMO_DB.PUBLIC.MY_S3_STAGE/Customer_joined_data/
```

```
from(
SELECT B.*
FROM CUSTOMER_SNOWFLAKE_TABLE B
LEFT OUTER JOIN
CUSTOMER_DATA A
ON
A.C_CUSTOMER_SK = B.C_CUSTOMER_SK
)
```

6. ADVANTAGES AND DISADVANTAGES

Write your views below,