SNOWFLAKE

ASSIGNMENT 2

A white background with black text

Description automatically generated

Upload sample data from snowflake to S3 through external stage.

--STEPS TO ULOAD DATA FROM SNOWFLAKE TABLE TO AWS S3 BUCKET

Execution query in SNOWFLAKE:

--craete warehouse

create warehouse customer;

--create database

create database customer;

--use the database created

use customer;

--change the role to accountadmin

USE ROLE ACCOUNTADMIN;

--create a table and load the data from snowflake\_sample\_data

CREATE OR REPLACE TRANSIENT TABLE cust\_data

AS

SELECT \* FROM

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

SELECT \* FROM cust\_data limit 10;

--CREATE FIEL FORMAT

create or replace file format csv\_unload

type = csv field\_delimiter = ',' skip\_header = 1 null\_if = ('NULL', 'null') empty\_field\_as\_null = true compression = gzip;

--CREATE STORAGE INTEGRATION(CREATE ROLE THROUGH IAM WITH S3 ALL ACCESS AND COPY ARM FROM ROLE)

create or replace storage integration s3\_cav

type = external\_stage

storage\_provider = s3

enabled = true

storage\_aws\_role\_arn = 'arn:aws:iam::891377129521:role/csv\_unload'

storage\_allowed\_locations = ('s3://csv11072024/sandhyas/');

--describe the integration created(copy the 5,6 arn and id in trust relations in roles created)

desc integration s3\_cav;

--CREATE EXTERNAL STAGE

create or replace stage customer.public.csv\_ext\_stage

storage\_integration = s3\_cav

url = 's3://csv11072024/sandhyas/'

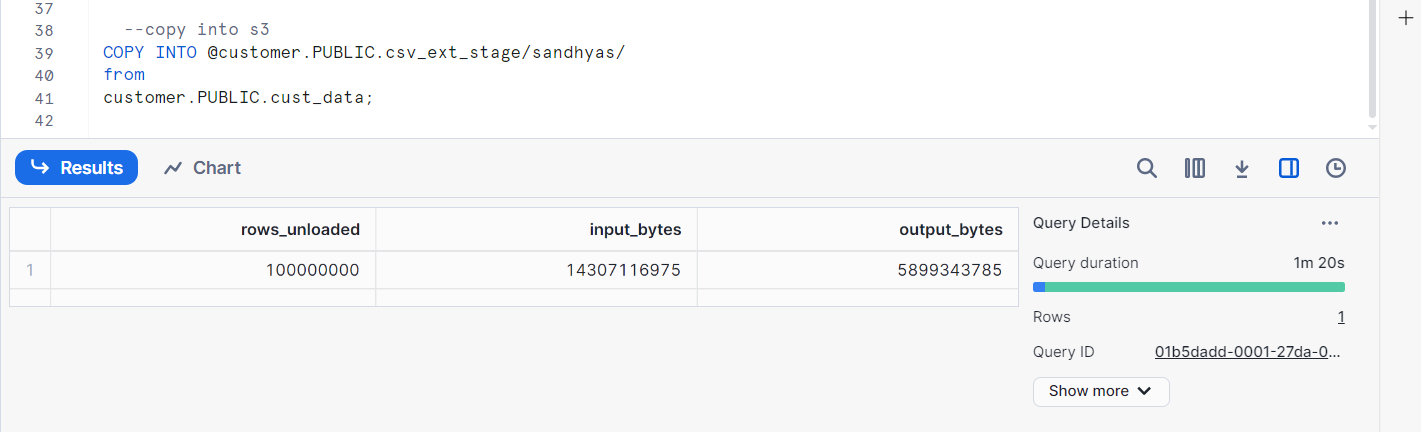
file\_format = customer.public.csv\_unload;

--copy the data from external stage to S3 bucket

COPY INTO @customer.PUBLIC.csv\_ext\_stage/sandhyas/

from

customer.PUBLIC.cust\_data;



Csv files in S3 bucket in GZ format:

A screenshot of a computer

Description automatically generated

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.

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,

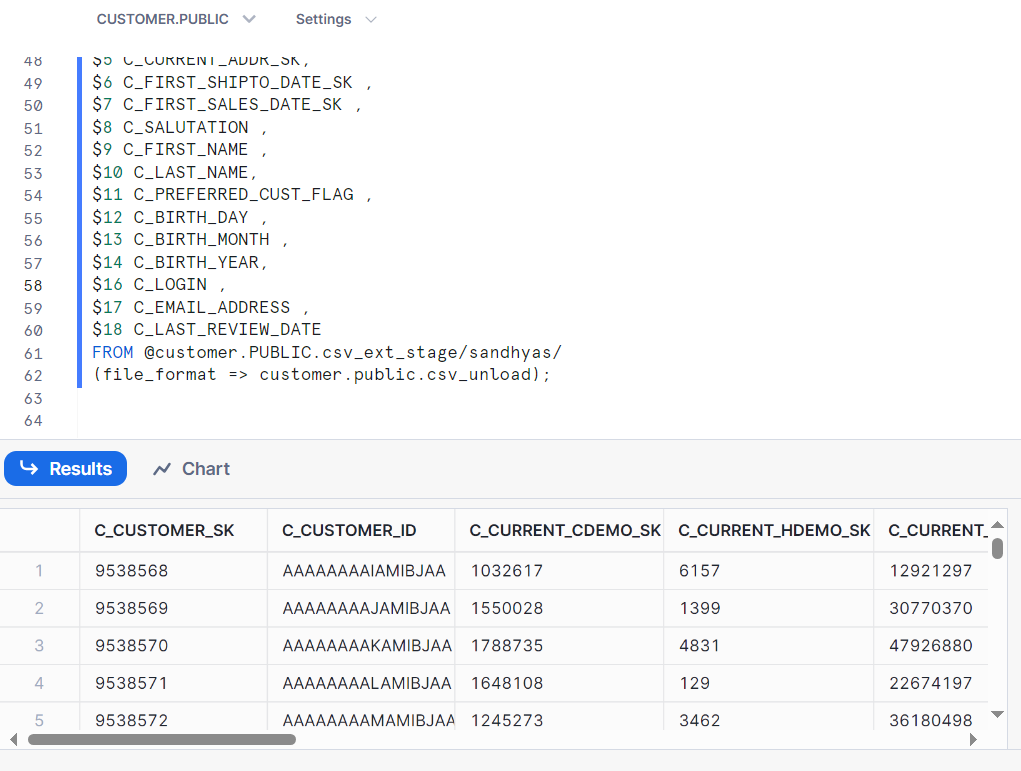
$16 C\_LOGIN ,

$17 C\_EMAIL\_ADDRESS ,

$18 C\_LAST\_REVIEW\_DATE

FROM @customer.PUBLIC.csv\_ext\_stage/sandhyas/

(file\_format => customer.public.csv\_unload);



--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 @customer.PUBLIC.csv\_ext\_stage/sandhyas/

(file\_format => customer.public.csv\_unload)

WHERE C\_CUSTOMER\_SK ='64596949'

A screenshot of a computer

Description automatically generated

**--Execute group by**

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

FROM @customer.PUBLIC.csv\_ext\_stage/sandhyas/

(file\_format => customer.public.csv\_unload)

GROUP BY $9,$10;

A screenshot of a computer

Description automatically generated

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 @customer.PUBLIC.csv\_ext\_stage/sandhyas/

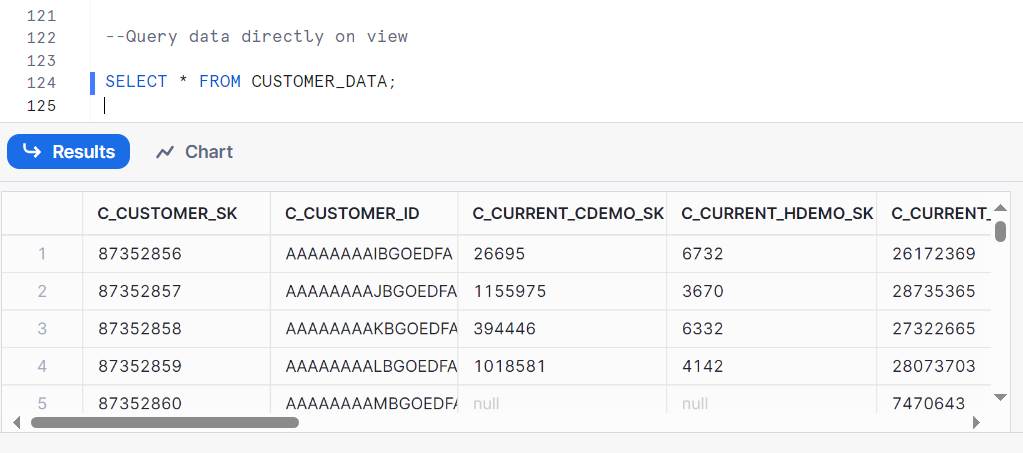
(file\_format => customer.public.csv\_unload);

A screenshot of a computer

Description automatically generated

--Query data directly on view

SELECT \* FROM 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??**

Yes, we can query data from S3 directly through views in Snowflake using external stages.

Disadvantages of this approach is mentioned below:

* **Performance**: Queries through views may have slower performance when we query directly from Snowflake internal storage. This is because data retrieval involves fetching from external storage over the network, we will face performance issues.
* **Cost**: Depending on the Amazon S3, Azure & frequency of data access, there might be extra costs along with data.
* **Data Consistency**: External data may change with out snowflake control, which leads to inconsistencies if we do not managed carefully.

**We can not see any partitions created in the backend for views**. Screenshot attached below:

A screenshot of a computer

Description automatically generated

--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;

A screenshot of a computer

Description automatically generated

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:

Joining data in S3 with Snowflake tables has a wide range of benefits related to **scalability** where S3 bucket can store vast amounts of data, **cost efficiency** in which storing data in S3 is typically cheaper than storing it in Snowflake's internal storage, **data integration** where we can integrate different verieties of data like structured, semi-structured also unstructured data. **performance optimization** through intelligent query planning and execution. **data governance** and **security and privacy**.

How many partitions got scanned from snowflake table: **scanned partitions are 356 and total partition is 1**, when checked through query profile:

A screenshot of a statistics

Description automatically generated

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 @customer.PUBLIC.csv\_ext\_stage/sandhyas\_join\_data/

from(

SELECT B.\*

FROM CUSTOMER\_SNOWFLAKE\_TABLE B

LEFT OUTER JOIN

customer\_data A

ON

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

A screenshot of a computer

Description automatically generated

In S3 Bucket:

A screenshot of a computer

Description automatically generated

6. ADVANTAGES AND DISADVANTAGES

**Advantages:**

* Joining data in S3 with Snowflake tables has a wide range of benefits related to scalability where S3 bucket can store vast amounts of data.
* cost efficiency in which storing data in S3 is typically cheaper than storing it in Snowflake's internal storage.
* data integration where we can integrate different varieties of data like structured, semi-structured also unstructured data.
* performance optimization through intelligent query planning and execution.
* data governance and security and privacy.

**Disadvantages:**

* **Performance**: Queries through views may have slower performance when we query directly from Snowflake internal storage. This is because data retrieval involves fetching from external storage over the network, we will face performance issues.
* **Cost**: Depending on the Amazon S3, Azure & frequency of data access, there might be extra costs along with data.
* **Data Consistency**: External data may change with out snowflake control, which leads to inconsistencies if we do not managed carefully.

///////////////////////////////////////////THANK YOU//////////////////////////////////////////////////////////

From,

Sandhya Shankar