**-- Create a Stage in Snowflake to load data from S3**

**CREATE OR REPLACE STAGE my\_s3\_stage**

**URL='s3://your\_bucket\_name'**

**CREDENTIALS=(AWS\_KEY\_ID='YOUR\_AWS\_KEY\_ID' AWS\_SECRET\_KEY='YOUR\_AWS\_SECRET\_KEY');**

* CREATE OR REPLACE STAGE my\_s3\_stage: This command creates a new stage named my\_s3\_stage in Snowflake. If a stage with the same name already exists, it is replaced with the new one.
* URL='s3://your\_bucket\_name': The URL parameter specifies the location of the S3 bucket from which the data will be loaded. Replace your\_bucket\_name with the actual name of your S3 bucket.
* CREDENTIALS=(AWS\_KEY\_ID='YOUR\_AWS\_KEY\_ID' AWS\_SECRET\_KEY='YOUR\_AWS\_SECRET\_KEY'): This part provides the AWS access credentials needed to authenticate and access the data in the specified S3 bucket. Replace YOUR\_AWS\_KEY\_ID and YOUR\_AWS\_SECRET\_KEY with your actual AWS access key ID and secret key.

**-- Create or Replace a Table in Snowflake to store the data**

**CREATE OR REPLACE TABLE my\_snowflake\_table (**

**data\_variant VARIANT**

**);**

* CREATE OR REPLACE TABLE my\_snowflake\_table: This command creates a new table named my\_snowflake\_table in Snowflake. If a table with the same name already exists, it will be replaced with the new one.
* column1\_name VARCHAR, column2\_name VARCHAR, ...: This part defines the table columns where you need to specify the column names and their appropriate data types. Adjust the column names and data types according to the structure of the data that will be loaded from the S3 bucket.

**-- Copy data from the S3 bucket to Snowflake**

**COPY INTO my\_snowflake\_table**

**FROM (**

**SELECT**

**$1 as column1\_name,**

**$2 as column2\_name**

**FROM @my\_s3\_stage**

**)**

**FILE\_FORMAT = (TYPE = 'CSV' FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"');**

* COPY INTO my\_snowflake\_table: This command instructs Snowflake to copy the data from the specified source into the Snowflake table named my\_snowflake\_table.
* FROM (SELECT $1 as column1\_name, $2 as column2\_name FROM @my\_s3\_stage): The FROM clause specifies the source location as the Snowflake stage named my\_s3\_stage, indicating that the data should be loaded from this stage into the Snowflake table.
* FILE\_FORMAT = (TYPE = 'CSV' FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"'): This parameter defines the file format for the data being loaded. In this case, it assumes a CSV file format with double quotes (") as the optional field enclosure. Snowflake automatically detects the file format based on the content, simplifying the loading process.

**-- Perform data quality checks and transformations as needed**

**-- Example data quality check:**

**SELECT COUNT(\*) AS total\_records FROM my\_snowflake\_table;**

* -- Perform data quality checks and transformations as needed: This comment encourages you to implement any required data quality checks or transformations that are specific to your data migration process. These checks can include ensuring data completeness, identifying duplicates, or validating data consistency.
* SELECT COUNT(\*) AS total\_records FROM my\_snowflake\_table;: This example demonstrates a data quality check by counting the total number of records in the my\_snowflake\_table. By executing this query, you can verify the successful migration of data and confirm the total number of records loaded into the Snowflake table.

**-- Verify data has been successfully loaded**

**SELECT \* FROM my\_snowflake\_table LIMIT 10; -- Adjust the limit as per your requirement**

* -- Verify data has been successfully loaded: This comment signifies the importance of confirming that the data has been effectively loaded into the my\_snowflake\_table.
* SELECT \* FROM my\_snowflake\_table LIMIT 10;: This query retrieves the first 10 rows from the my\_snowflake\_table, allowing you to inspect the data and ensure that it aligns with the data from the source S3 bucket. You can modify the LIMIT clause as needed to retrieve a different number of rows for verification.

**-- Create a view to parse data from combined\_data into separate columns**

**CREATE OR REPLACE VIEW my\_parsed\_table\_view AS**

**SELECT**

**combined\_data:column1\_name::VARCHAR AS column1\_name,**

**combined\_data:column2\_name::VARCHAR AS column2\_name,**

**combined\_data:column3\_name::INT AS column3\_name,**

**combined\_data:column4\_name::TIMESTAMP AS column4\_name**

**FROM my\_snowflake\_table;**

**-- Create or Replace a Table in Snowflake to store the data**

**CREATE OR REPLACE TABLE my\_snowflake\_table (**

**data\_variant VARIANT**

**);**

**-- Copy data from the S3 bucket to Snowflake**

**COPY INTO my\_snowflake\_table (data\_variant)**

**FROM (**

**SELECT PARSE\_JSON($1) as data\_variant**

**FROM @my\_s3\_stage**

**(FILE\_FORMAT => (TYPE => 'CSV', SKIP\_HEADER => 1))**

**);**

**-- Extract column names from the header row**

**WITH header AS (**

**SELECT PARSE\_JSON(column\_value) AS header\_array**

**FROM my\_snowflake\_table,**

**LATERAL FLATTEN(input => data\_variant[0])**

**)**

**-- Generate dynamic SQL to create the table with appropriate column names**

**, dynamic\_sql AS (**

**SELECT 'CREATE OR REPLACE TABLE my\_snowflake\_table (' ||**

**LISTAGG('"' || header\_array || '" VARCHAR', ',') ||**

**');' AS create\_table\_sql**

**FROM header**

**)**

**-- Execute the dynamic SQL to create the final table**

**SELECT EXECUTE\_IMMEDIATE(create\_table\_sql)**

**FROM dynamic\_sql;**

CREATE OR REPLACE TABLE your\_table

AS

SELECT \* FROM @your\_stage\_name

(FILE\_FORMAT => (TYPE => 'CSV', FIELD\_OPTIONALLY\_ENCLOSED\_BY => '"', SKIP\_HEADER => 1, AUTO\_DETECT => TRUE));

CREATE OR REPLACE TABLE your\_table (data VARIANT);

COPY INTO your\_table (data)

FROM @your\_stage\_name

FILES = ('s3://your\_bucket\_name/your\_file\_name.csv')

CREDENTIALS = (

AWS\_KEY\_ID='YOUR\_AWS\_ACCESS\_KEY\_ID'

AWS\_SECRET\_KEY='YOUR\_AWS\_SECRET\_KEY'

)

FILE\_FORMAT = (TYPE = CSV, FIELD\_OPTIONALLY\_ENCLOSED\_BY='"', SKIP\_HEADER = 1, AUTO\_DETECT = TRUE);

<https://docs.snowflake.com/en/user-guide/tables-external-s3>

<https://www.youtube.com/watch?v=6a61K-3Oh1g>

-- Create an external stage

CREATE OR REPLACE STAGE my\_stage FILE\_FORMAT = (TYPE=CSV);

-- Copy the CSV file from S3 to the external stage

COPY INTO my\_stage FROM @my\_s3\_bucket/my\_csv\_file.csv;

-- Get the header row of the CSV file

SELECT \* FROM @my\_stage LIMIT 1;

-- Generate a DDL statement to create a Snowflake table with the same columns and data types as the header row

SELECT GENERATE\_COLUMN\_DESCRIPTION('my\_table', 1) FROM @my\_stage LIMIT 1;

-- Execute the DDL statement to create the Snowflake table

EXECUTE IMMEDIATE '<DDL statement>';

-- Load the data from the CSV file in the external stage into the Snowflake table

COPY INTO my\_table FROM @my\_stage FILE\_FORMAT = (TYPE=CSV) ON\_ERROR = SKIP\_FILE;

—---------------------------

-- Get the header row of the CSV file

SELECT \* FROM @my\_stage LIMIT 1;

-- Generate a DDL statement to create a Snowflake table with the same columns and data types as the header row

SET ddl\_statement = GENERATE\_COLUMN\_DESCRIPTION('my\_table', 1);

-- Execute the DDL statement to create the Snowflake table

EXECUTE IMMEDIATE ddl\_statement;

—----------------------------

-- Create an external stage

CREATE OR REPLACE STAGE my\_csv\_file\_stage FILE\_FORMAT = (TYPE=CSV);

-- Copy the CSV file from S3 to the external stage

COPY INTO my\_csv\_file\_stage FROM @my\_s3\_bucket/my\_csv\_file.csv;

-- Get the header row of the CSV file

SELECT \* FROM @my\_csv\_file\_stage LIMIT 1;

-- Generate a DDL statement to create a Snowflake table with the same columns and data types as the header row

SET ddl\_statement = GENERATE\_COLUMN\_DESCRIPTION('my\_table', 1) FROM @my\_csv\_file\_stage LIMIT 1;

-- Execute the DDL statement to create the Snowflake table

EXECUTE IMMEDIATE ddl\_statement;

-- Load the data from the CSV file in the external stage into the Snowflake table

COPY INTO my\_table FROM @my\_csv\_file\_stage FILE\_FORMAT = (TYPE=CSV) ON\_ERROR = SKIP\_FILE;

—--------------

COPY INTO your\_staging\_table

FROM @your\_s3\_stage

FILE\_FORMAT = (FORMAT\_NAME = your\_csv\_format)

PATTERN = '.\*.csv';

SELECT t.$1, t.$2,t.$3,t.$4 FROM @my\_s3\_stage2 t;

**– – AZURE BLOG STORAGE TO SNOWFLAKE**

CREATE STORAGE INTEGRATION Azure\_azsqldwdevdiag839  
 TYPE = EXTERNAL\_STAGE  
 STORAGE\_PROVIDER = 'AZURE'  
 ENABLED = TRUE  
 AZURE\_TENANT\_ID = '897dbc0d-c02d-4347-9a71-3e589c67f8aa'  
 STORAGE\_ALLOWED\_LOCATIONS = ('azure://azsqldwdevdiag839.blob.core.windows.net/python-dev/Joey\_data/woodlands\_json/json\_for\_storageblob/')  
 -- [ STORAGE\_BLOCKED\_LOCATIONS = ('azure://<account>.blob.core.windows.net/<container>/<path>/', 'azure://<account>.blob.core.windows.net/<container>/<path>/') ]

DESC STORAGE INTEGRATION Azure\_azsqldwdevdiag839;

create database azure\_blob\_storage\_azsqldwdevdiag839;

create schema azure\_blob\_storage\_azsqldwdevdiag839.woodlands;  
 USE SCHEMA mydb.public;

create FILE FORMAT my\_json\_format   
 type = 'JSON'

CREATE STAGE woodlands\_stage  
 STORAGE\_INTEGRATION = Azure\_azsqldwdevdiag839  
 URL = 'azure://azsqldwdevdiag839.blob.core.windows.net/python-dev/Joey\_data/woodlands\_json/json\_for\_storageblob/'  
 FILE\_FORMAT = my\_json\_format;

list @woodlands\_stage;

-- creating a json file format  
 create or replace file format json\_file\_format   
 type = 'JSON'  
 STRIP\_OUTER\_ARRAY = true;

create or replace file format json\_file\_format\_outer\_array\_false   
 type = 'JSON'  
 STRIP\_OUTER\_ARRAY = false;

select \* from table(infer\_schema(  
 location => '@woodlands\_stage/' ,  
 files => '2020-11-23\_816935.json',  
 file\_format=>'json\_file\_format',  
 ignore\_case => true

));

select \* from table(infer\_schema(  
 location => '@woodlands\_stage/' ,  
 files => '2020-11-23\_816935.json',  
 file\_format=>'json\_file\_format\_outer\_array\_false'

));

create or replace transient table file20201123\_816935  
 using template (  
 select array\_agg(object\_construct(\*)) from table(  
 infer\_schema(  
 location => '@woodlands\_stage/' ,  
 files => '2020-11-23\_816935.json',  
 file\_format=>'json\_file\_format'  
 )  
 )  
 );

desc table file20201123\_816935;

copy into file20201123\_816935 from (  
 select  
 $1:Duration::TIME(9),  
 $1:Identifier::VARCHAR(16777216),  
 $1:Segments::ARRAY,  
 $1:StartOfMessage::TIME(9),  
 $1:TC\_Anchor::TIME(9),  
 $1:Title::VARCHAR(16777216),  
 $1:Type::VARCHAR(16777216)  
 from @woodlands\_stage/2020-11-23\_816935.json  
 (file\_format => json\_file\_format) t)  
 on\_error = 'Continue' ;

select \* from file20201123\_816935