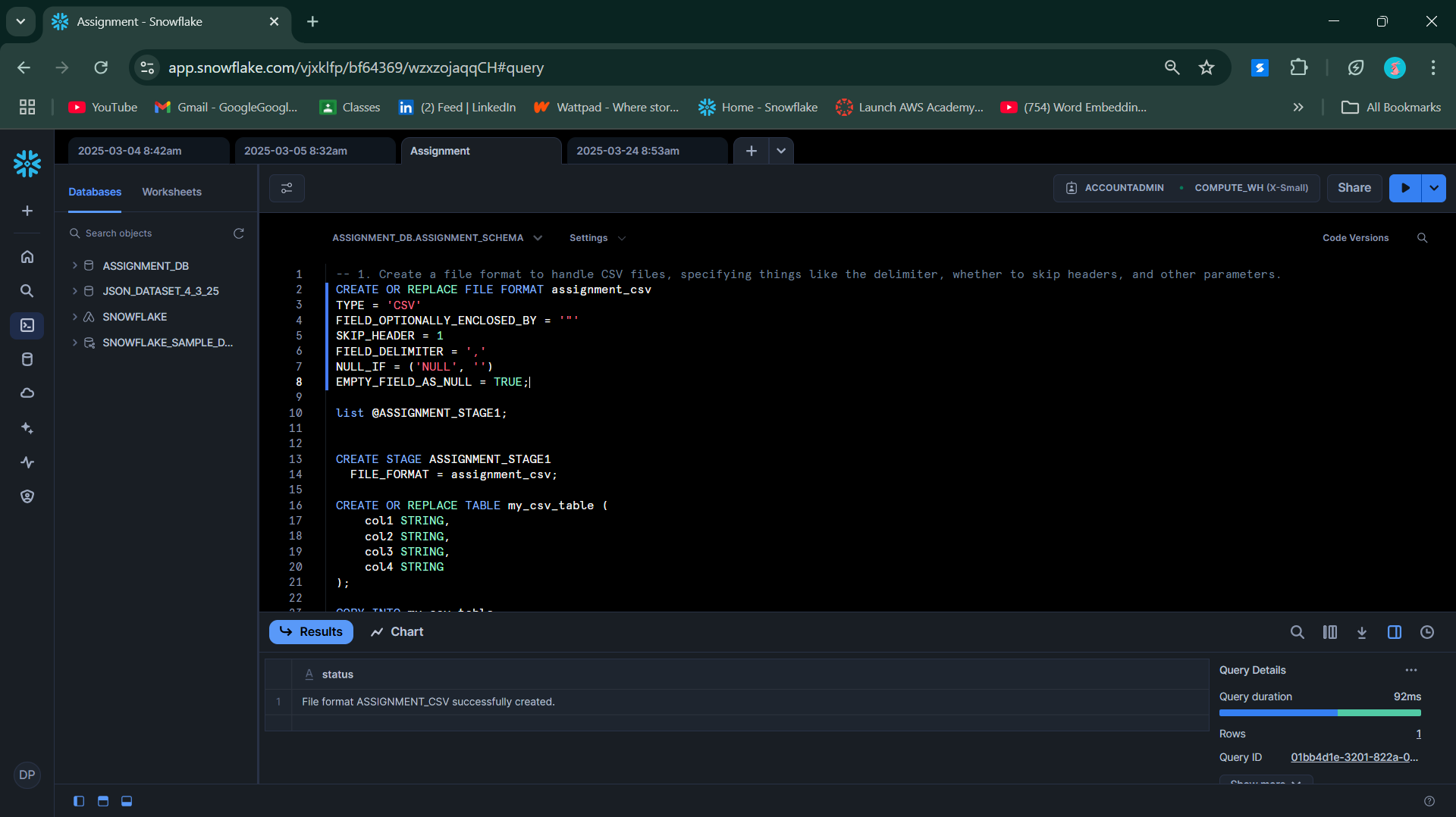
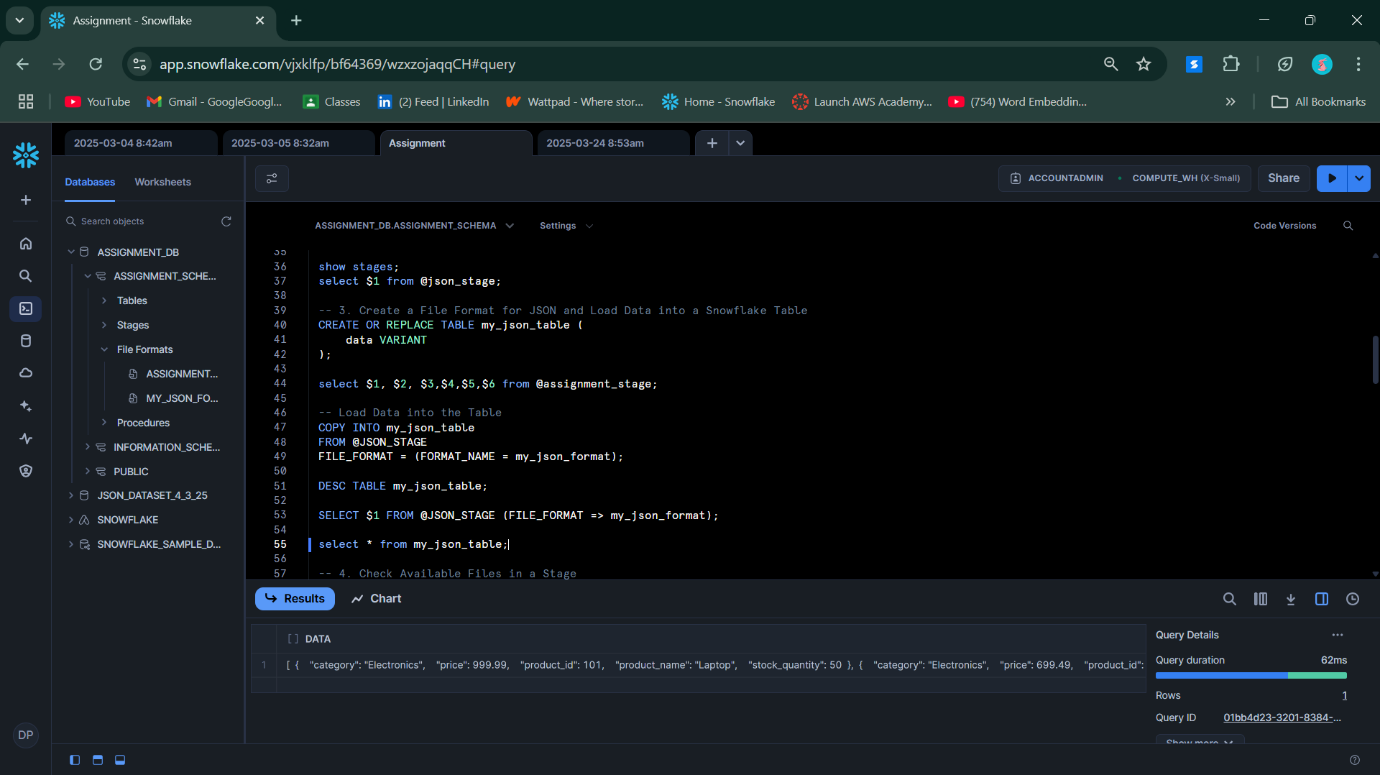
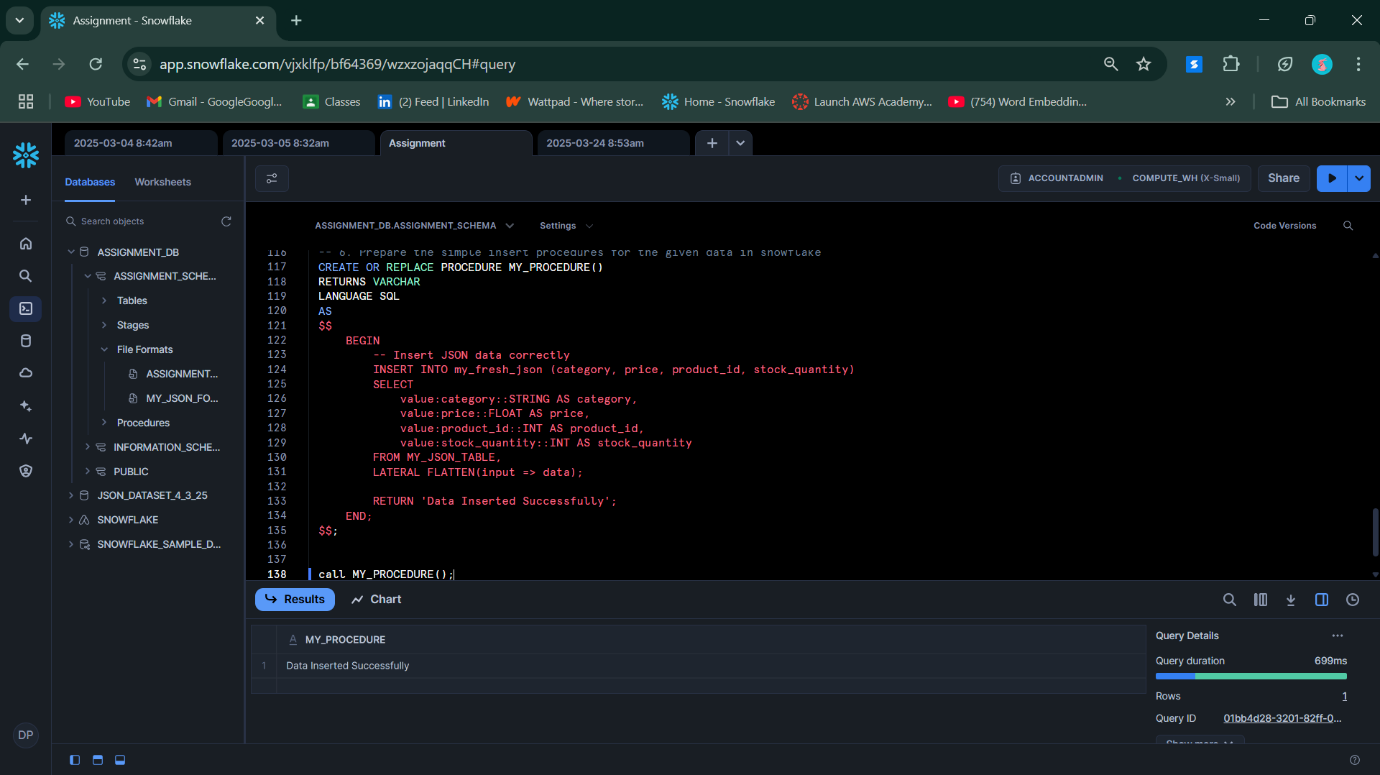
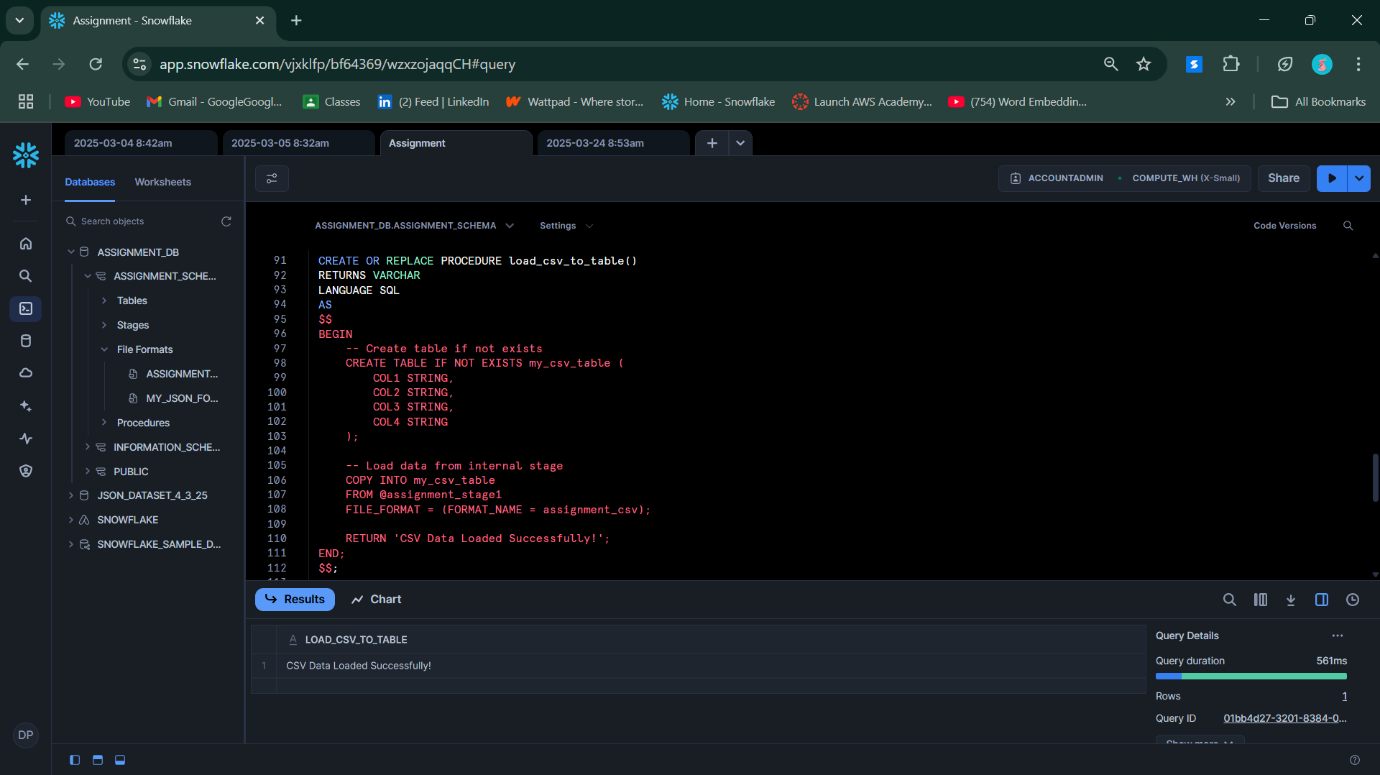
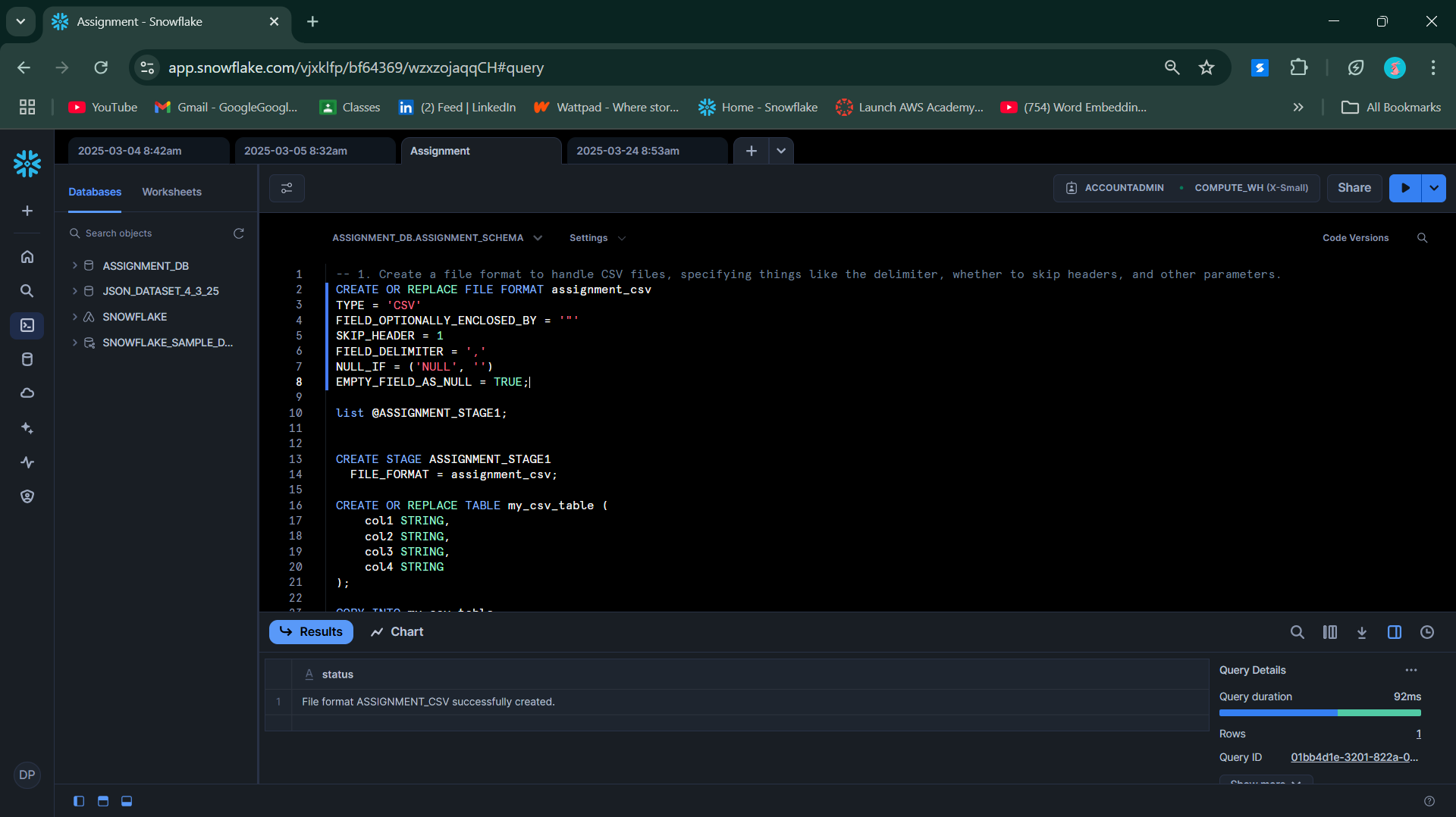
# **Snowflake Assignment**

1. Create a file format to handle CSV files, specifying things like the delimiter, whether to skip headers, and other parameters.



1. Create a file format to handle JSON files.



1. Create a file format for JSON files and load data into a Snowflake table.
2. 
3. 
4. 

-- 1. Create a file format to handle CSV files, specifying things like the delimiter, whether to skip headers, and other parameters.

CREATE OR REPLACE FILE FORMAT assignment\_csv

TYPE = 'CSV'

FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"'

SKIP\_HEADER = 1

FIELD\_DELIMITER = ','

NULL\_IF = ('NULL', '')

EMPTY\_FIELD\_AS\_NULL = TRUE;

list @ASSIGNMENT\_STAGE1;

CREATE STAGE ASSIGNMENT\_STAGE1

FILE\_FORMAT = assignment\_csv;

CREATE OR REPLACE TABLE my\_csv\_table (

col1 STRING,

col2 STRING,

col3 STRING,

col4 STRING

);

COPY INTO my\_csv\_table

FROM @assignment\_stage1

FILE\_FORMAT = (FORMAT\_NAME = assignment\_csv);

SELECT \* FROM my\_csv\_table;

CREATE OR REPLACE FILE FORMAT assignment\_csv

TYPE = 'CSV';

show stages;

select $1 from @json\_stage;

-- 3. Create a File Format for JSON and Load Data into a Snowflake Table

CREATE OR REPLACE TABLE my\_json\_table (

data VARIANT

);

select $1, $2, $3,$4,$5,$6 from @assignment\_stage;

-- Load Data into the Table

COPY INTO my\_json\_table

FROM @JSON\_STAGE

FILE\_FORMAT = (FORMAT\_NAME = my\_json\_format);

DESC TABLE my\_json\_table;

SELECT $1 FROM @JSON\_STAGE (FILE\_FORMAT => my\_json\_format);

select \* from my\_json\_table;

-- 4. Check Available Files in a Stage

list @ASSIGNMENT\_STAGE;

list @JSON\_STAGE;

-- Select all data from a table

SELECT \* FROM MY\_JSON\_TABLE;

-- Select specific columns

-- SELECT 1, 2 FROM MY\_JSON\_TABLE;

SELECT COUNT(\*) FROM MY\_JSON\_TABLE;

SELECT data FROM MY\_JSON\_TABLE LIMIT 1;

SELECT data FROM MY\_JSON\_TABLE;

SELECT

value:category::STRING AS category,

value:price::FLOAT AS price,

value:product\_id::INT AS product\_id,

value:product\_name::STRING AS product\_name,

value:stock\_quantity::INT AS stock\_quantity

FROM MY\_JSON\_TABLE,

LATERAL FLATTEN(input => data);

-- WHERE value:category = 'Electronics';

SELECT \* FROM MY\_JSON\_TABLE;

-- WHERE data:column1 = 'value';

CREATE OR REPLACE TABLE my\_fresh\_json (

category STRING,

price FLOAT,

product\_id INT,

stock\_quantity INT

);

CREATE OR REPLACE PROCEDURE load\_csv\_to\_table()

RETURNS VARCHAR

LANGUAGE SQL

AS

$$

BEGIN

-- Create table if not exists

CREATE TABLE IF NOT EXISTS my\_csv\_table (

COL1 STRING,

COL2 STRING,

COL3 STRING,

COL4 STRING

);

-- Load data from internal stage

COPY INTO my\_csv\_table

FROM @assignment\_stage1

FILE\_FORMAT = (FORMAT\_NAME = assignment\_csv);

RETURN 'CSV Data Loaded Successfully!';

END;

$$;

CALL load\_csv\_to\_table();

-- 6. Prepare the simple insert procedures for the given data in snowflake

CREATE OR REPLACE PROCEDURE MY\_PROCEDURE()

RETURNS VARCHAR

LANGUAGE SQL

AS

$$

BEGIN

-- Insert JSON data correctly

INSERT INTO my\_fresh\_json (category, price, product\_id, stock\_quantity)

SELECT

value:category::STRING AS category,

value:price::FLOAT AS price,

value:product\_id::INT AS product\_id,

value:stock\_quantity::INT AS stock\_quantity

FROM MY\_JSON\_TABLE,

LATERAL FLATTEN(input => data);

RETURN 'Data Inserted Successfully';

END;

$$;

call MY\_PROCEDURE();