ASSESSMENT 2

1. **Scale Virtual Warehouses and Test Performance with Large Datasets Using Snowpark**

```
#Create or use a warehouse
CREATE OR REPLACE WAREHOUSE DEMO_WH
  WAREHOUSE_SIZE = 'XSMALL'
  AUTO_SUSPEND = 60
  AUTO_RESUME = TRUE;

ALTER WAREHOUSE DEMO_WH SET WAREHOUSE_SIZE = 'LARGE';

SHOW WAREHOUSES;

#SNOWPARK DATASET TESTING

from snowflake.snowpark import Session
from snowflake.snowpark.functions import col

connection_parameters = {
    "account": "sb81969.me-central2.gcp",
    "user": "yuvasri310",
    "password": "Yuvasri3102004",
    "role": "SYSADMIN",
    "warehouse": "DEMO_WH",
    "database": "TEST_DB",
    "schema": "PUBLIC"
}

session = Session.builder.configs(connection_parameters).create()

# Create a large DataFrame from an existing table
df = session.table("LARGE_SALES")

# Example transformation
result = df.group_by(col("REGION")).agg({"AMOUNT": "sum"})

result.show()
```

2.Set Up Snowpipe with Azure Blob Storage and Monitor Using COPY_HISTORY

#Create a Storage Integration:

```
CREATE OR REPLACE STORAGE INTEGRATION azure_snowpipe_integration
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = AZURE
ENABLED = TRUE
AZURE_TENANT_ID = '7540734b-e567-46c3-9ad3-ec9fb9e50140'
STORAGE_ALLOWED_LOCATIONS =
('azure://yuvacontainer.blob.core.windows.net/datafiles/');

DESC INTEGRATION azure_snowpipe_integration;

#CREATE EXTERNAL STAGE
CREATE OR REPLACE STAGE azure_stage
STORAGE_INTEGRATION = azure_snowpipe_integration
URL = 'azure://yuvacontainer.blob.core.windows.net/datafiles/';

#CREATE TARGET TABLE
CREATE OR REPLACE TABLE SALES_RAW (
   ID INT,
   PRODUCT STRING,
   AMOUNT FLOAT,
   REGION STRING
);

#CREATE SNOWPIPE
CREATE OR REPLACE PIPE sales_pipe
AUTO_INGEST = TRUE
AS
COPY INTO SALES_RAW
FROM @azure_stage
FILE_FORMAT = (TYPE = 'CSV' FIELD_OPTIONALLY_ENCLOSED_BY='');

SELECT * FROM
TABLE(INFORMATION_SCHEMA.COPY_HISTORY(TABLE_NAME =>
'SALES_RAW', START_TIME => DATEADD('hour', -1,
CURRENT_TIMESTAMP())));
```

3.Automate Snowpipe with Azure Functions

```
#CREATE AZURE FUNCTION
import json
import requests
import os

def main(event: dict):
    for record in event:
        file_url = record['data']['url']
        print(f"New file detected: {file_url}")

        # Trigger Snowpipe REST API
        headers = {
            "Authorization": f"Bearer {os.environ['SNOWFLAKE_TOKEN']}"
        }
        data = {
            "stageName": "azure_stage",
            "pipeName": "sales_pipe"
        }
        response = requests.post(

"https://yuvasristorage.snowflakecomputing.com/v1/data/pipes/sales_pipe/insertFiles"
,
            headers=headers,
            json=data
        )
        print(response.text)

#TEST THE AUTOMATION
SELECT COUNT(*) FROM SALES_RAW;

#MONITOR SNOWPIPE LOADS
SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.LOAD_HISTORY WHERE
PIPE_NAME = 'SALES_PIPE';
```