Migration of On-Premises SQL Server Database to Snowflake via Azure Databricks

Objective

Migrate data from an **on-premises SQL Server** to **Snowflake** using **Azure Databricks** for transformation with **Snowpark**.

Step 1: Extract Data from SQL Server

Connection Setup in Databricks

```
jdbc_hostname = "<your-sql-server-hostname>"
jdbc_port = "1433"
jdbc_database = "<your-database-name>"
jdbc_url = f"jdbc:sqlserver://{jdbc_hostname}:{jdbc_port};databaseName={jdbc_database}"
jdbc_username = dbutils.secrets.get(scope="sql_scope", key="username")
jdbc_password = dbutils.secrets.get(scope="sql_scope", key="password")
```

Read Data from SQL Server

```
source_table = "dbo.EmployeeDetails"
df_sql = spark.read \
    .format("jdbc") \
    .option("url", jdbc_url) \
    .option("dbtable", source_table) \
    .option("user", jdbc_username) \
    .option("password", jdbc_password) \
    .load()

df_sql.show(5)
```

Step 2: Transform Data Using Snowpark

Create Snowpark Session in Databricks

```
from snowflake.snowpark import Session
from snowflake.snowpark.functions import col, upper

snowflake_conn = {
    "account": "<your-snowflake-account>",
    "user": dbutils.secrets.get(scope="snow_scope", key="username"),
    "password": dbutils.secrets.get(scope="snow_scope", key="password"),
    "role": "SYSADMIN",
    "warehouse": "COMPUTE_WH",
    "database": "TARGET_DB",
    "schema": "PUBLIC"
}
```

```
session = Session.builder.configs(snowflake_conn).create()
```

Apply Transformations

```
df_transformed = df_sql.withColumn("EmployeeName", upper(col("EmployeeName")))
df_transformed = df_transformed.dropDuplicates(["EmployeeID"])
df_transformed.display()
```

Step 3: Load Data into Snowflake

Write Data to Snowflake Table

```
sf_options = {
    "sfURL": "<your-snowflake-account>.snowflakecomputing.com",
    "sfDatabase": "TARGET_DB",
    "sfSchema": "PUBLIC",
    "sfWarehouse": "COMPUTE_WH",
    "sfRole": "SYSADMIN",
    "sfUser": dbutils.secrets.get(scope="snow_scope", key="username"),
    "sfPassword": dbutils.secrets.get(scope="snow_scope", key="password")
}

df_transformed.write \
    .format("snowflake") \
    .options(**sf_options) \
    .option("dbtable", "EMPLOYEE_MASTER") \
    .mode("overwrite") \
    .save()
```

Step 4: Validate Data in Snowflake

```
USE DATABASE TARGET_DB;
USE SCHEMA PUBLIC;

SELECT COUNT(*) FROM EMPLOYEE_MASTER;
SELECT * FROM EMPLOYEE_MASTER LIMIT 10;
```

Step 5: Performance Optimization

1. Use **Snowflake Warehouse Scaling** for large data loads.
2. Use **partitioning and caching** in Databricks.
3. Run **COPY_HISTORY** in Snowflake to monitor load history.

Step 6: Automation Using Azure Functions

You can automate this pipeline using an **Azure Function** triggered by schedule or event.

Sample Azure Function (Python)

```
import pyodbc
import snowflake.connector

def main(mytimer):
    # Extract from SQL Server
    conn_sql = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL
Server};SERVER=<server>;DATABASE=<db>;UID=<user>;PWD=<pwd>')
    cursor = conn_sql.cursor()
    cursor.execute('SELECT * FROM dbo.EmployeeDetails')

    # Load to Snowflake
    conn_sf = snowflake.connector.connect(
        user='<user>',
        password='<password>',
        account='<account>',
        warehouse='COMPUTE_WH',
        database='TARGET_DB',
        schema='PUBLIC'
    )

    cs = conn_sf.cursor()
    for row in cursor.fetchall():
        cs.execute(f"INSERT INTO EMPLOYEE_MASTER VALUES ({','.join(map(str, row))})")

    conn_sql.close()
    conn_sf.close()
```