**Case Study 2 SF**

**Scenario: Build a real-time log analytics pipeline using Azure Databricks, Snowpark, and Snowflake.**

**Task: Ingest JSON logs from ADLS, transform them in Databricks using Snowpark, and load them into Snowflake for analysis.**

**Step 1: Understand the Data & Scenario**

* Scenario: You have logs in JSON format stored in Azure Data Lake Storage (ADLS). These logs contain information like:

event: login/logout

timestamp: 2025-10-16T09:23:00Z

user: pooja

* Goal: Read these logs in Databricks, clean/transform them using Snowpark, and load them into a Snowflake table for analytics.

Think of it as a stream of logs coming from your application, which you want to analyze in near real-time.

**Step 2: Set Up Your Environment**

A. Azure Databricks

* Create a Databricks workspace in Azure.
* Create a cluster (make sure it’s running).
* Install the Snowflake connector for Spark and Snowpark library on the cluster.

B. Snowflake

* Create a database and schema for logs.
* Create a table for cleaned logs:

CREATE OR REPLACE DATABASE LOG\_ANALYTICS;

USE DATABASE LOG\_ANALYTICS;

CREATE OR REPLACE SCHEMA PUBLIC;

CREATE OR REPLACE TABLE LOGS\_CLEANED (

event STRING,

timestamp STRING,

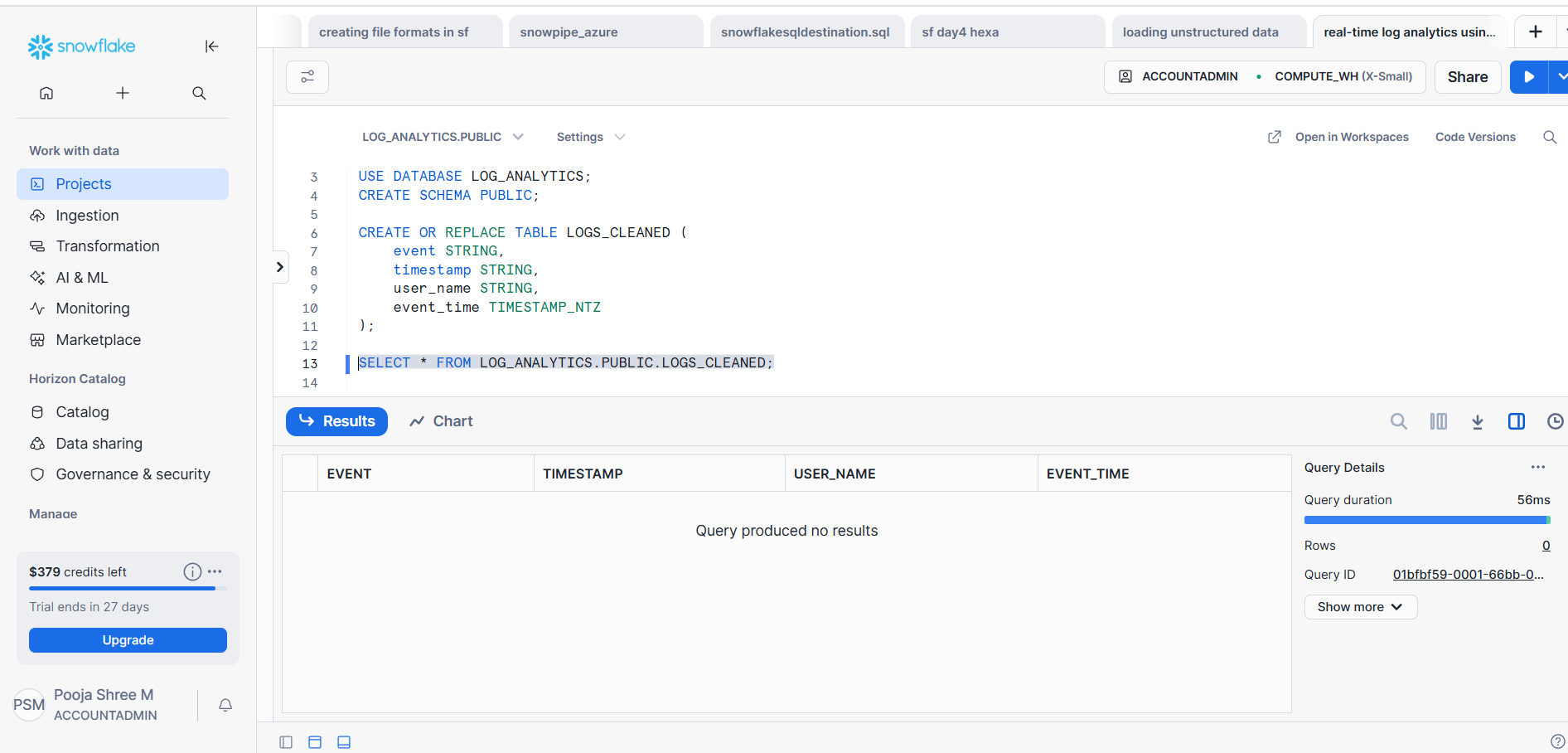
user\_name STRING,

event\_time TIMESTAMP\_NTZ

);

Notes:

* + user\_name instead of user because USER is reserved in Snowflake.
  + event\_time will store proper timestamp values.



**Step 3: Connect Databricks to ADLS**

Since you don’t have secrets management or role assignment authority, the simplest method is to use Storage Account Key:

storage\_account\_name = "sfhexastorage"

storage\_account\_key = "<your-storage-account-key>"

spark.conf.set(

f"fs.azure.account.key.{storage\_account\_name}.dfs.core.windows.net",

storage\_account\_key

)

# Container + path

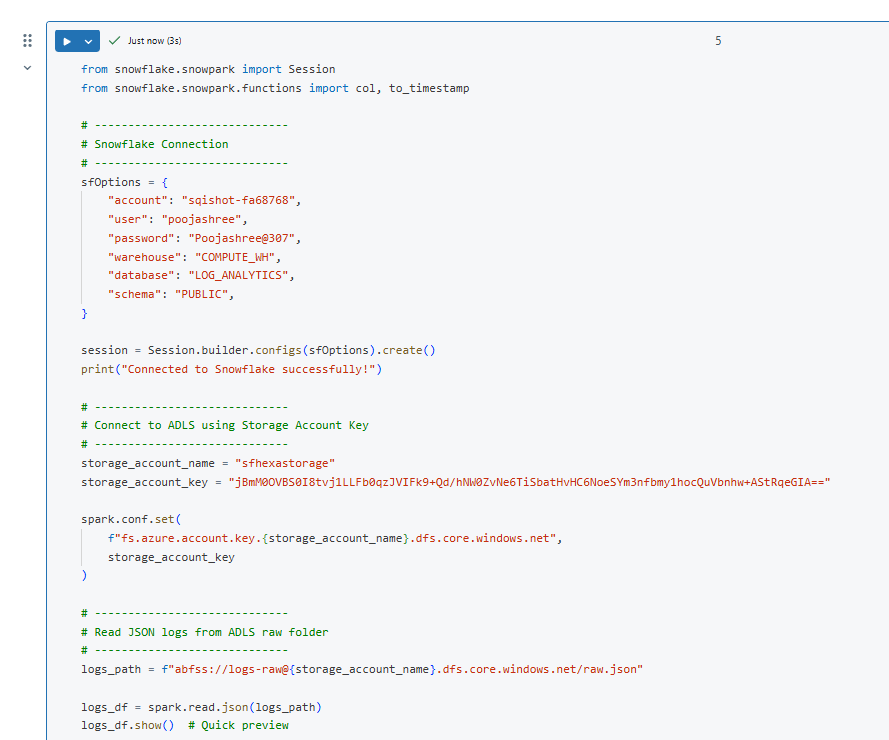
logs\_path = f"abfss://logs-raw@{storage\_account\_name}.dfs.core.windows.net/raw.json"

# Read JSON

logs\_df = spark.read.json(logs\_path)

display(logs\_df)

After this step, your logs are loaded into a Spark DataFrame in Databricks.



**Step 4: Transform Data Using Snowpark**

* Snowpark is like pandas but for Snowflake, integrated into Databricks via Spark.
* You can do transformations, e.g., converting timestamp strings to proper TIMESTAMP\_NTZ, renaming columns to match Snowflake table:

from snowflake.snowpark import Session

from snowflake.snowpark.functions import col, to\_timestamp

# Transform the DataFrame

sp\_df\_cleaned = logs\_df.select(

col("event").alias("event"),

col("timestamp").alias("timestamp"),

col("user").alias("user\_name"),

to\_timestamp(col("timestamp")).alias("event\_time")

)

At this point, the DataFrame matches the Snowflake table schema.

**Step 5: Connect to Snowflake from Databricks**

* Use Snowflake connector / Snowpark session:

from snowflake.snowpark import Session

connection\_parameters = {

"account": "<your\_account>",

"user": "<your\_user>",

"password": "<your\_password>",

"warehouse": "<your\_warehouse>",

"database": "LOG\_ANALYTICS",

"schema": "PUBLIC"

}

session = Session.builder.configs(connection\_parameters).create()



**Step 6: Write the Data to Snowflake**

sp\_df\_cleaned.write.mode("append").save\_as\_table("LOGS\_CLEANED")

print("Data written to Snowflake successfully!")

Now your cleaned logs are in Snowflake and ready for analysis.

**Step 7: Analysis in Snowflake**

* You can now query the logs, e.g.:

SELECT user\_name, COUNT(\*) AS logins

FROM LOGS\_CLEANED

WHERE event = 'login'

GROUP BY user\_name;

* Or track activity trends over time:

SELECT DATE\_TRUNC('hour', event\_time) AS hour, COUNT(\*) AS events

FROM LOGS\_CLEANED

GROUP BY hour

ORDER BY hour;

**Step 8: Optional - Automate for Real-Time**

* For near real-time ingestion, you can use:
  1. Databricks Structured Streaming to watch ADLS folder.
  2. Transform each batch using Snowpark.
  3. Append to Snowflake continuously.

This avoids manually re-running jobs every time logs are generated.

