**Name -Monika Gariya**

**Email-** [**monikagariya2023@gmail.com**](mailto:monikagariya2023@gmail.com)

**Data Engineering Batch 1**

**Date – 14-02-2024**

**Topic - Delta Lake, Delta Table ,Creating, Updating, Deleting Tables**

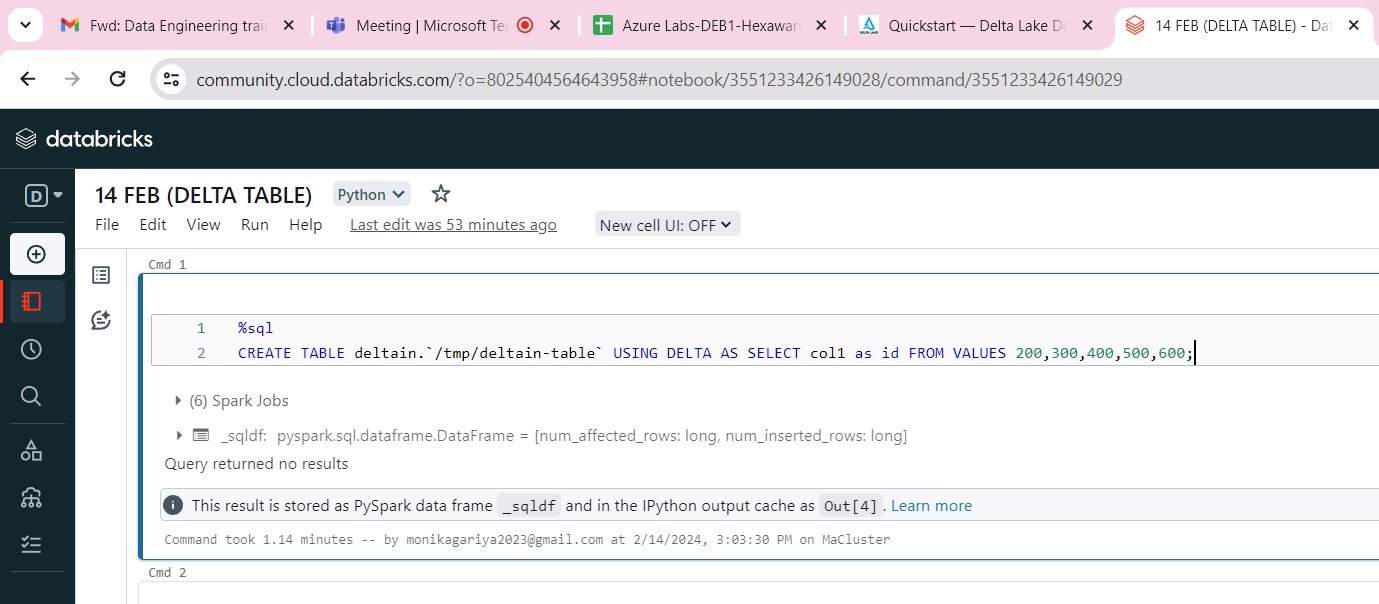
Delta Lake is an open-source storage layer that brings ACID (Atomicity, Consistency, Isolation, Durability) transactions to Apache Spark and big data workloads. It provides a reliable and scalable data lake platform by adding a layer of reliability to data lakes, enabling features like ACID transactions, schema enforcement, and time travel queries for large-scale data processing.

Delta Lake is built on top of existing data lake solutions like Apache Hadoop and cloud storage systems like Azure Data Lake Storage (ADLS). It uses Parquet format for storage and provides efficient data management capabilities.

A Delta Table is essentially a table in Delta Lake. It organizes data into a directory of Parquet files and maintains transaction logs. These transaction logs allow Delta Lake to provide ACID transactions, schema enforcement, and other features.

**1)CREATE A TABLE**

To create a Delta table, write a DataFrame out in the delta format. You can use existing Spark SQL code and change the format from parquet, csv, json, and so on, to delta.



**2)READ DATA**

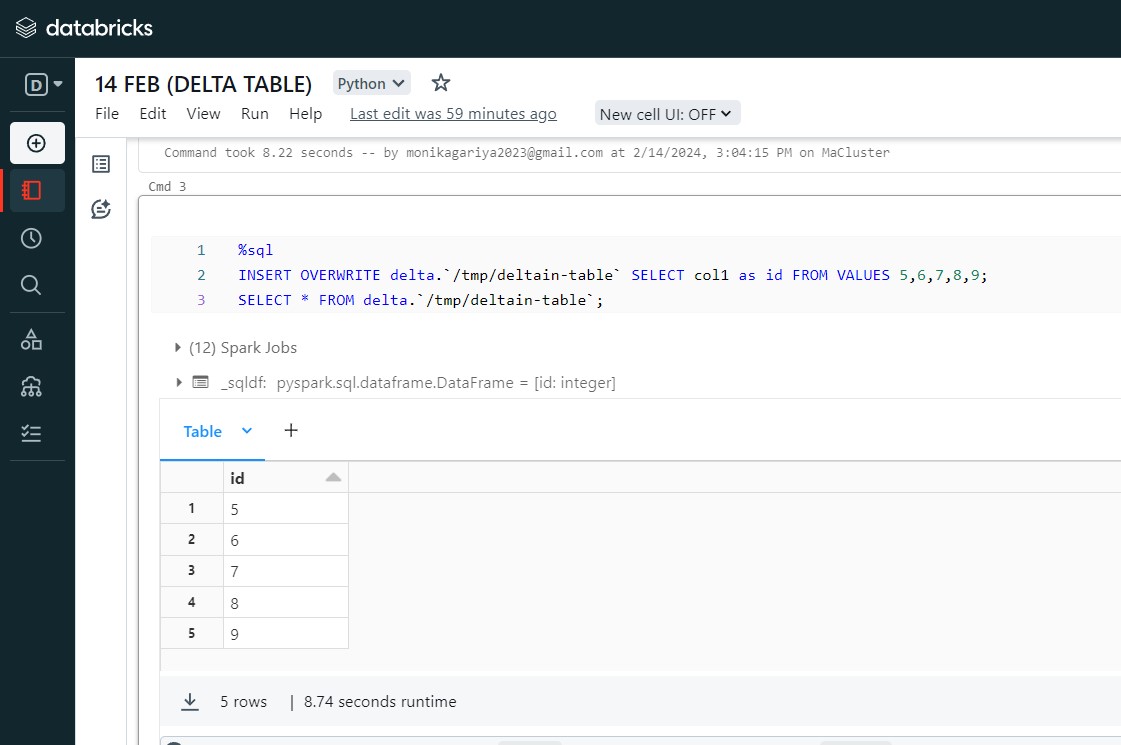
You read data in your Delta table by specifying the path to the files: "/tmp/deltain-table":



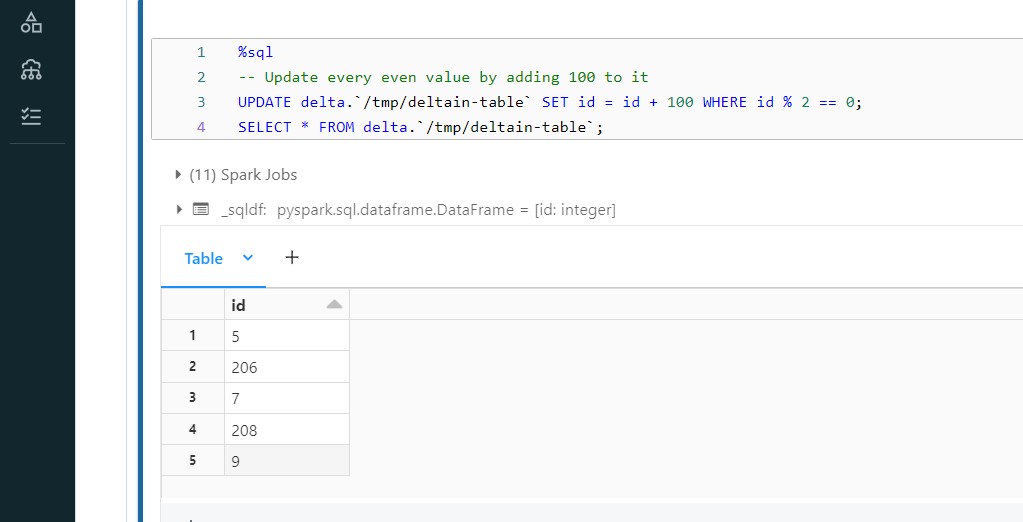
**3)UPDATE TABLE DATA**

Delta Lake supports several operations to modify tables using standard DataFrame APIs. This example runs a batch job to overwrite the data in the table:

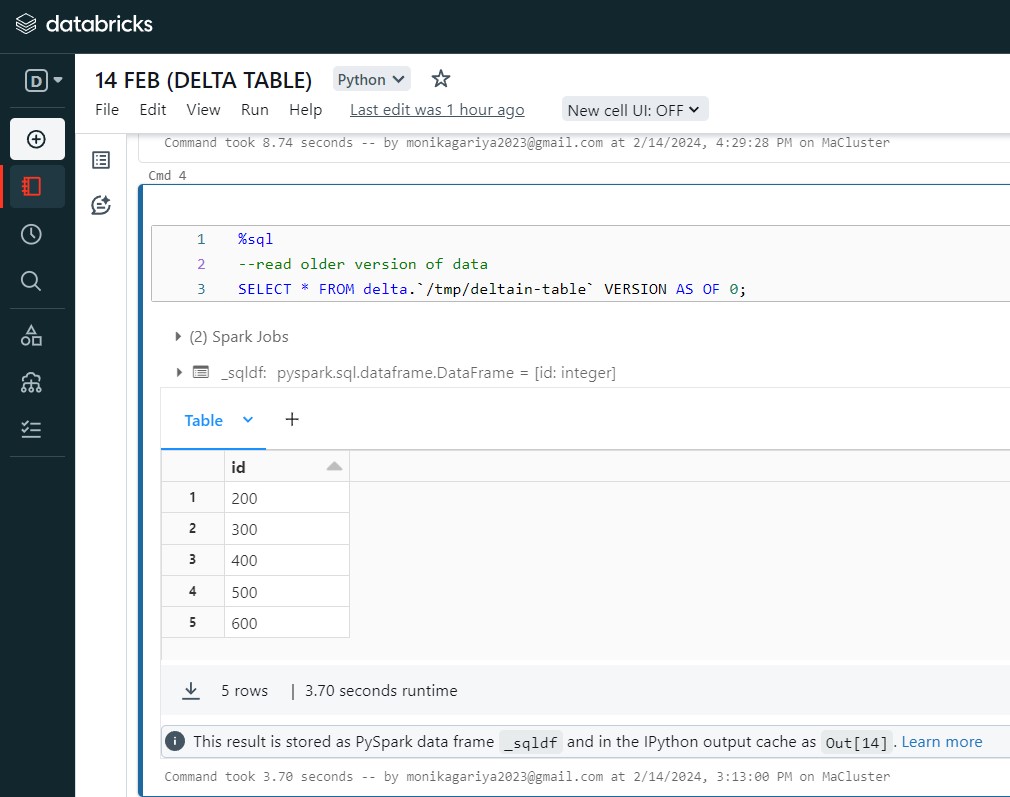
**overwrite**

****

**4)Update the table data**

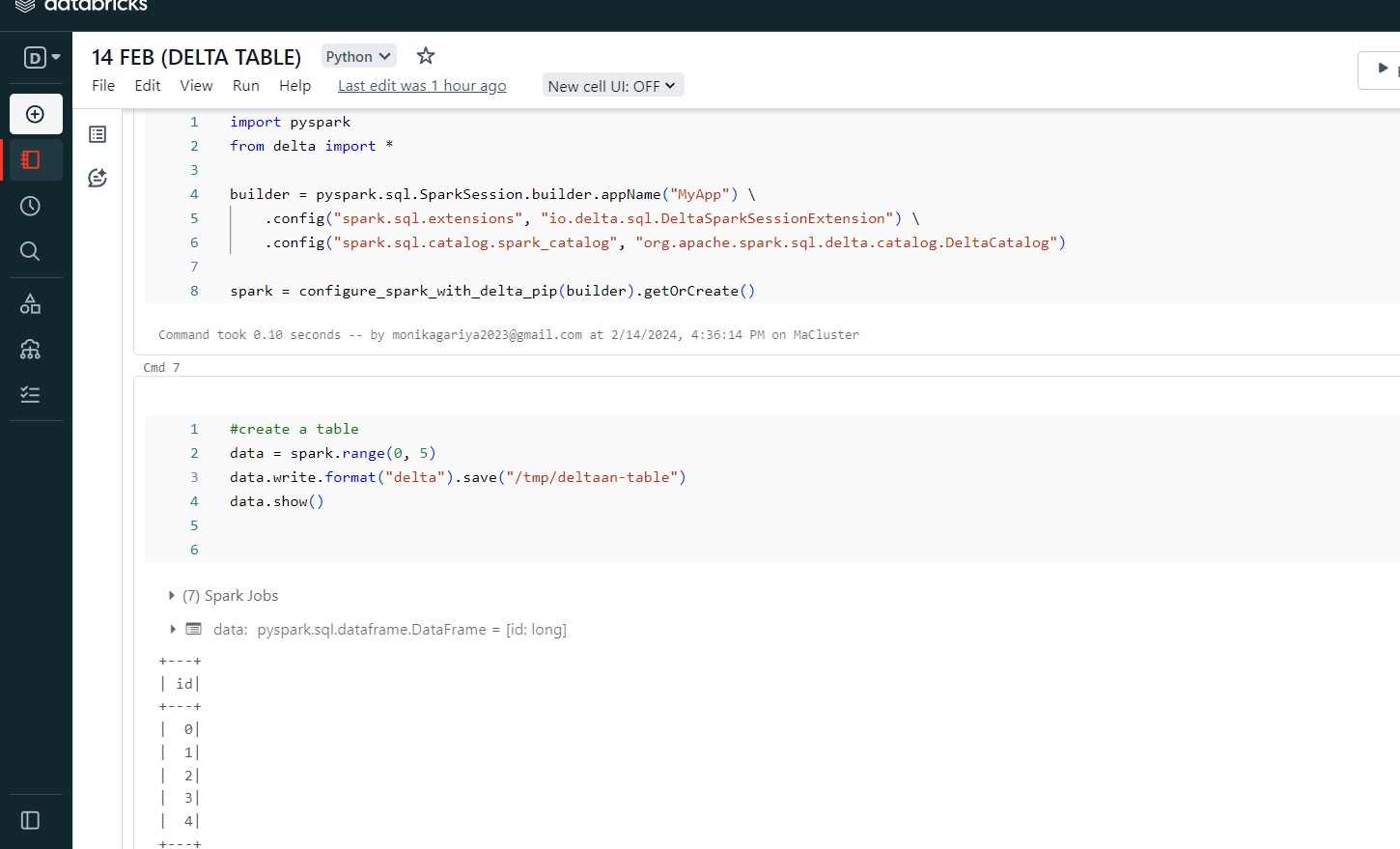
****

**5)Read older version of data using time travelling**

****

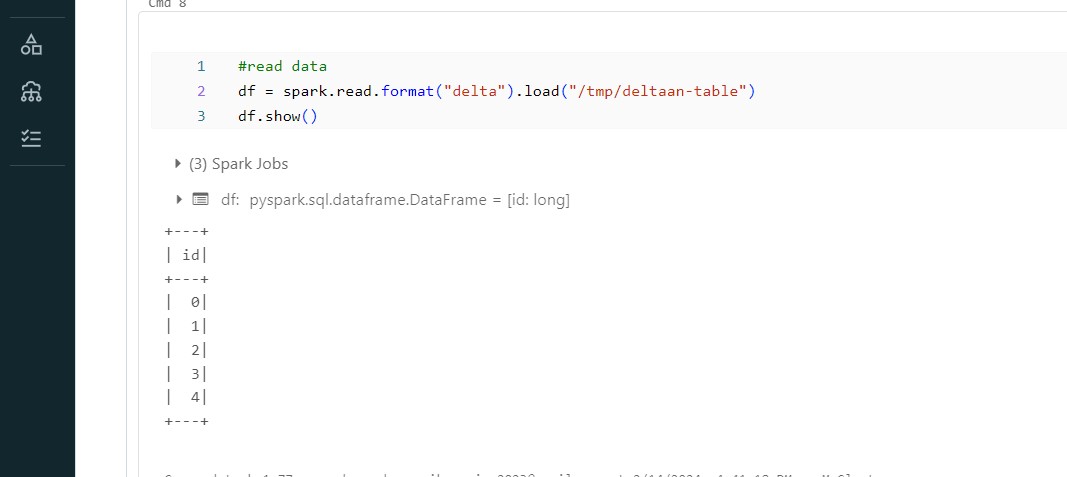
**In Python**

**1)CREATE A TABLE**



**2)READ DATA**

You read data in your Delta table by specifying the path to the files: "/tmp/deltaan-table":



**3)UPDATE TABLE DATA**

Delta Lake supports several operations to modify tables using standard DataFrame APIs. This example runs a batch job to overwrite the data in the table:

USING OVERWRITE



**4)READ OLDER VERSION OF DATA USING TIME TRAVELLING**

You can query previous snapshots of your Delta table by using time travel. If you want to access the data that you overwrote, you can query a snapshot of the table before you overwrote the first set of data using the versionAsOf option.

****

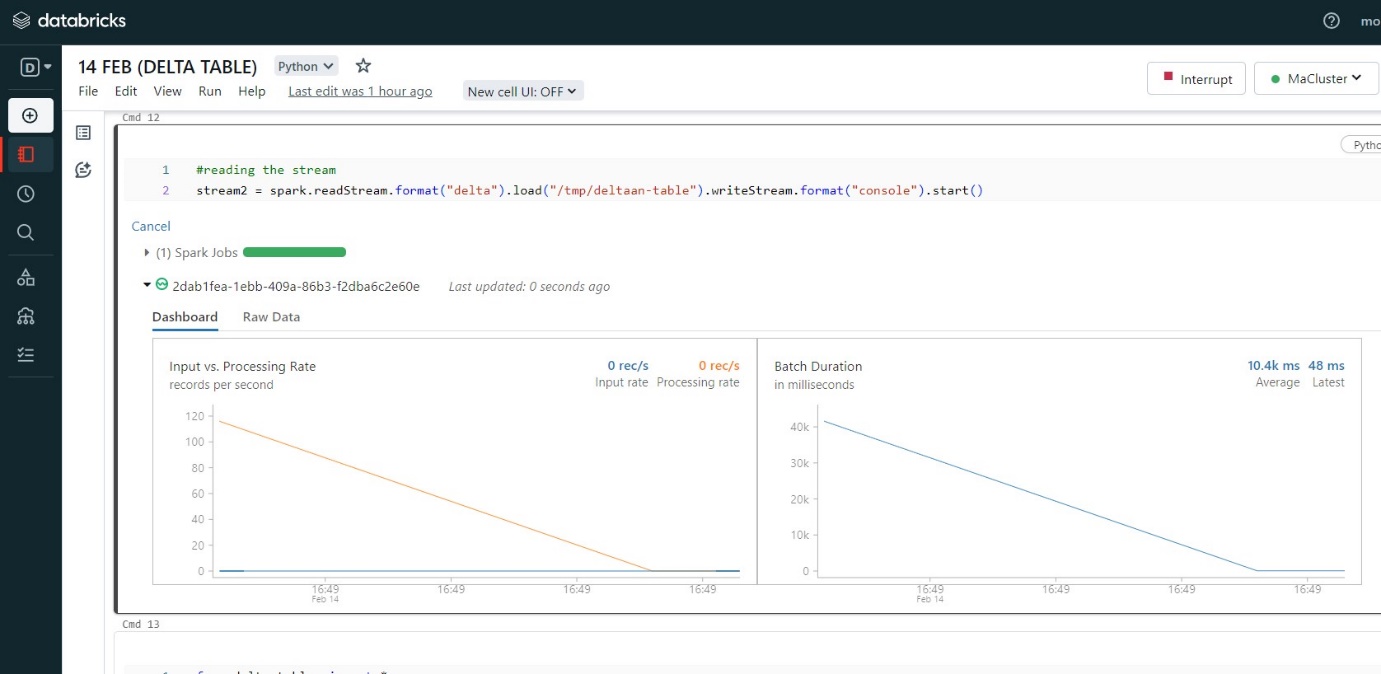
**5)WRITE STREAM DATA TO THE TABLE**

write to a Delta table using Structured Streaming. The Delta Lake transaction log guarantees exactly-once processing, even when there are other streams or batch queries running concurrently against the table. By default, streams run in append mode, which adds new records to the table:

****

**6)READ A STREAM FROM A CHANGES FROM A TABLE**

While the stream is writing to the Delta table, you can also read from that table as streaming source. For example, you can start another streaming query that prints all the changes made to the Delta table. You can specify which version Structured Streaming should start from by providing the starting Version or starting Timestamp option to get changes from that point onwards



**7)CONDITIONAL UPDATE WITHOUT OVERWRITE**

Delta Lake provides programmatic APIs to conditional update, delete, and merge (upsert) data into tables.

