

# Assignment-18

## Name-Ajay Chaudhary Batch-Data Engineering (Batch-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`.

### Read data

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

The screenshot shows a Databricks Delta lake notebook interface. The notebook is titled "Delta lake" and is in Python mode. It contains two commands:

```
Cmd 1
1 data = spark.range(0, 5)
2 data.write.format("delta").save("/tmp/delta-table")

(6) Spark Jobs
data: pyspark.sql.dataframe.DataFrame = [id: long]
Command took 27.54 seconds -- by azuser1071_mml.local@iihtl.onmicrosoft.com at 14/02/2024, 15:24:42 on azuser1071_mml.local's Cluster
```

```
Cmd 2
1 df = spark.read.format("delta").load("/tmp/delta-table")
2 df.show()

(3) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [id: long]
+----+
| id |
+----+
|  3 |
|  4 |
|  0 |
|  1 |
|  2 |
+----+
```

Command took 3.81 seconds -- by azuser1071\_mml.local@iihtl.onmicrosoft.com at 14/02/2024, 15:25:51 on azuser1071\_mml.local's Cluster

Cmd 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:

```
Command took 3.81 seconds — by azuser1071_mml.local@iihtl.onmicrosoft.com at 14/02/2024, 15:25:51 on azuser1071_mml.local's Cluster
```

```
Cmd 3
```

```
1 data = spark.range(5, 10)
2 data.write.format("delta").mode("overwrite").save("/tmp/delta-table")
3 data.show()
```

```
▶ (8) Spark Jobs
▶ data: pyspark.sql.dataframe.DataFrame = [id: long]
```

| id |
|----|
| 5  |
| 6  |
| 7  |
| 8  |
| 9  |

```
Command took 3.18 seconds — by azuser1071_mml.local@iihtl.onmicrosoft.com at 14/02/2024, 15:27:22 on azuser1071_mml.local's Cluster
```

# Read older versions of data using time travel

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.

```
Cmd 4
```

```
1 df = spark.read.format("delta").option("versionAsOf", 0).load("/tmp/delta-table")
2 df.show()
```

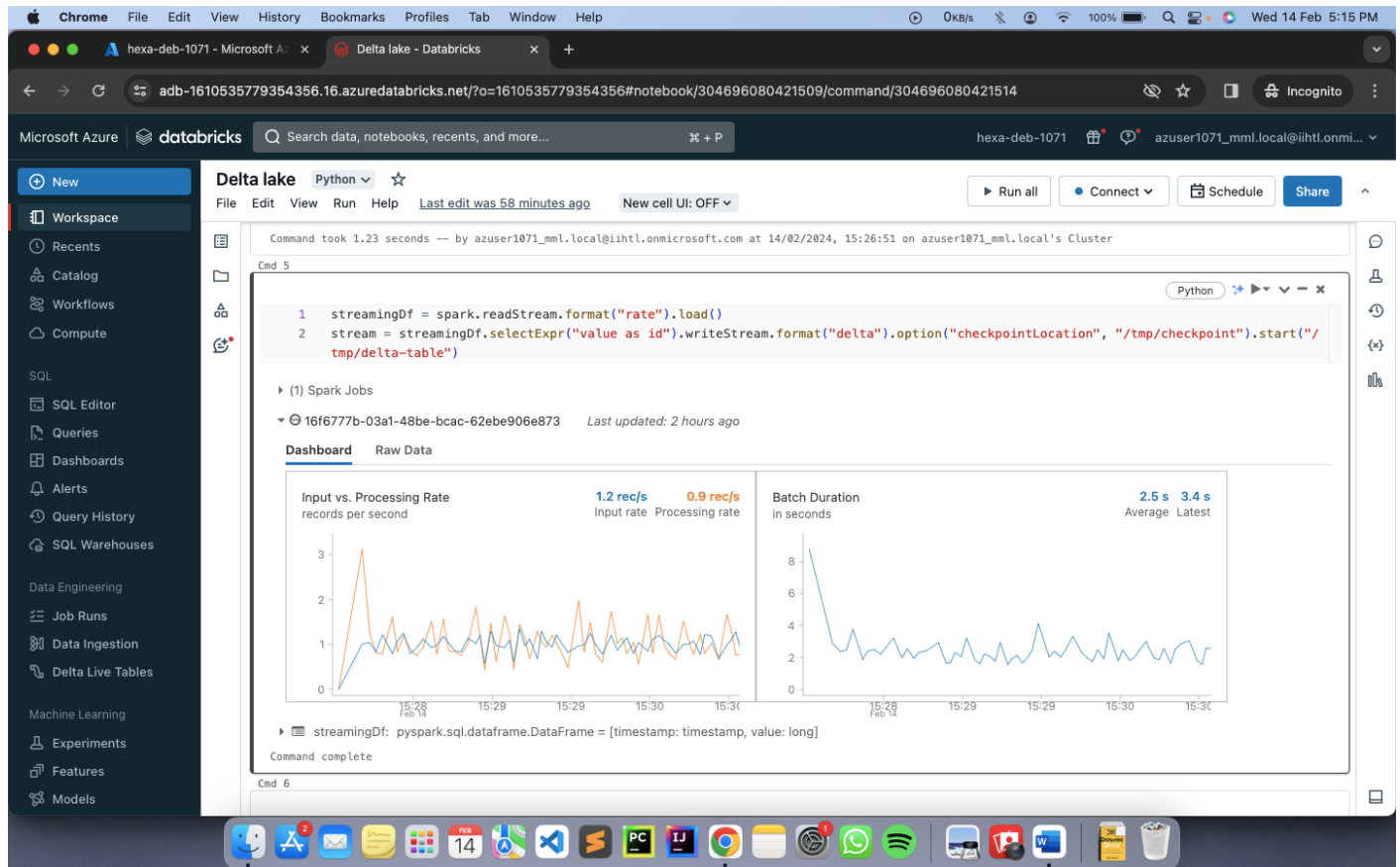
```
▶ (3) Spark Jobs
▶ df: pyspark.sql.dataframe.DataFrame = [id: long]
```

| id |
|----|
| 3  |
| 4  |
| 0  |
| 1  |
| 2  |

```
Command took 1.23 seconds — by azuser1071_mml.local@iihtl.onmicrosoft.com at 14/02/2024, 15:26:51 on azuser1071_mml.local's Cluster
```

# Write a stream of data to a table

You can also 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:



# Read a stream of 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.

The screenshot displays the Databricks Delta Lake interface within a web browser. The top navigation bar includes the Microsoft Azure logo, the Databricks logo, a search bar, and user information. The left sidebar contains a navigation menu with options like Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, Experiments, Features, and Models.

The main workspace area shows a Python cell with the following code:

```
streamingDf: pyspark.sql.dataframe.DataFrame = [timestamp: timestamp, value: long]
Command complete
```

Below the code cell, a Spark job is listed with the ID 9d54de3b-f110-4698-96e3-cb6f43201964, last updated 2 hours ago. The job's dashboard is displayed, showing two charts:

- Input vs. Processing Rate:** A line chart showing records per second. The input rate is 0 rec/s (blue line) and the processing rate is 12.9 rec/s (orange line). The x-axis represents time from 15:37 to 15:41 on Feb 14.
- Batch Duration:** A line chart showing batch duration in milliseconds. The average duration is 843.7 ms (blue line) and the latest duration is 12.5k ms (orange line). The x-axis represents time from 15:37 to 15:41 on Feb 14.

The dashboard also includes a 'Command complete' status and a 'Cmd 7' label.