🔰Delta Lake Lab 4 — Time Travel

Author: Sandeep Kumar Sharma

This lab demonstrates one of the most exciting features of Delta Lake:

🔆TIME TRAVEL

Time Travel allows you to query **older versions** of your Delta tables. It is extremely useful for:

- Auditing
- Debugging
- Recovering deleted/updated data
- Comparing historical snapshots

This lab is intentionally **very simple** so that every beginner can understand it.

🈂️What is Time Travel?

Delta Lake keeps a **transaction log** called `_delta_log`. Each time you write/update/overwrite data, Delta creates a **new version**.

Time Travel allows you to query:

- A previous **VERSION NUMBER**, or
- A previous **TIMESTAMP**

Example:

```
SELECT * FROM table VERSION AS OF 0
```

✅Step 1: Create Initial Data

```
from pyspark.sql import Row

initial_data = [
    Row(id=1, name="Alice"),
    Row(id=2, name="Bob")
]

df_initial = spark.createDataFrame(initial_data)
```

✅Step 2: Write Initial Data as a Delta Table

```
path = "/mnt/delta/time_travel_demo"

df_initial.write.format("delta").mode("overwrite").save(path)
```

This becomes **Version 0**.

✅Step 3: Create New (Updated) Data

```
new_data = [
    Row(id=1, name="Alice", city="Delhi"),
    Row(id=2, name="Bob", city="Mumbai"),
    Row(id=3, name="Charlie", city="Pune")
]

df_new = spark.createDataFrame(new_data)
```

✅Step 4: Overwrite Table with New Data (Creates Version 1)

```
df_new.write.format("delta").mode("overwrite").save(path)
```

Now the table has:

- Version **0** = old data
- Version **1** = new data

🛄Step 5: Read Current Version (Version 1)

```
display(spark.read.format("delta").load(path))
```

You will see **Alice, Bob, Charlie** with the `city` column.

🔅Step 6: Time Travel to Version 0 (Old Data)

```
df_old = spark.read.format("delta").option("versionAsOf", 0).load(path)
display(df_old)
```

You will now see only:

- Alice
- Bob

(without the `city` column!)

🔅Step 7: Alternate Method — Time Travel Using Timestamp

You can get the timestamp of each version:

```
spark.sql(f"DESCRIBE HISTORY delta.`{path}`").show(truncate=False)
```

Then use it:

```
df_old_ts = spark.read.format("delta").option("timestampAsOf", "<PASTE-
TIMESTAMP>").load(path)
display(df_old_ts)
```

---

6 Step 8: Compare Both Versions (Optional)

```
print("CURRENT VERSION (1):")
display(spark.read.format("delta").load(path))

print("PREVIOUS VERSION (0):")
display(spark.read.format("delta").option("versionAsOf", 0).load(path))
```

This clearly shows how Delta Lake preserves historical snapshots.

‼️ Lab Completed!

You mastered:

- Writing initial data (Version 0)
- Overwriting with new data (Version 1)
- Querying using **versionAsOf**
- Querying using **timestampAsOf**