**DELTA LAKE ASSIGNMENT**

**Theophila Murphy S**

**DE-Batch 6**

1. **Database vs Data Warehouse vs Data Lake vs Delta Lake**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Database** | **Data Warehouse** | **Data Lake** | **Data Lake** |
| Purpose | Store operational/transactional data (OLTP) | Store and analyze structured historical data (OLAP) | Store all types of raw data (structured, semi-structured, unstructured) | Brings reliability to Data Lakes with ACID transactions and schema enforcement |
| Data Type | Structured | Structured | Structured, Semi-structured, Unstructured | Same as Data Lake but with schema validation |
| Latency | Low latency for transactions | Low latency for transactions | Depends on tools | Low latency + ACID guarantees |
| Scalability | Scales vertically (limited) | Scales horizontally (costly) | Highly scalable (cheap storage) | |  | | --- | | Scalable with  reliability |  |  | | --- | |  | |
| Examples | MySQL, PostgreSQL, Oracle | |  | | --- | | Snowflake, Redshift, BigQuery | | |  | | --- | | Snowflake, Redshift, BigQuery | | Databricks Delta Lake |
| Best For | |  | | --- | | Day-to-day operations | | |  | | --- | | Day-to-day operations | | |  | | --- | | Day-to-day operations | | |  | | --- | | Day-to-day operations | |

**2. Internals of Delta Lake**

Delta Lake is built on **Apache Parquet** format and adds:

* **ACID Transactions** → Ensures consistency during concurrent writes.
* **Schema Enforcement & Evolution** → Prevents bad data from corrupting tables, but can evolve schema when needed.
* **Time Travel** → Query older versions of data using VERSION AS OF or TIMESTAMP AS OF.
* **Transaction Log (\_delta\_log folder)** → JSON & Parquet log files that track changes to the table.
* **Optimized Reads/Writes** → Uses file pruning, caching, and compaction.

**3. Optimizing Delta Tables**

**Why optimize?**

* Over time, small files accumulate (especially in streaming or frequent writes).
* Small files → more metadata overhead → slower queries.

**How to optimize in Databricks**

# Optimize to combine small files

spark.sql("""

OPTIMIZE my\_delta\_table

""")

# Z-Ordering for faster filtering

spark.sql("""

OPTIMIZE my\_delta\_table

ZORDER BY (customer\_id)

""")

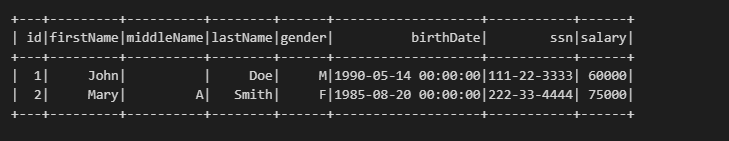
**Benefits:**

* Faster queries
* Reduced metadata load
* Better compression

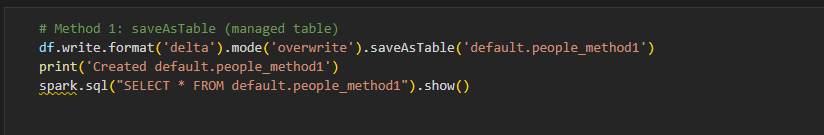
**DELTA LAKE**

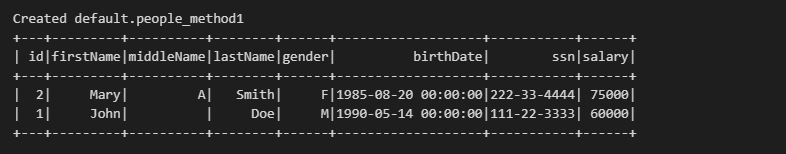
**Define Schema and Sample DataFrame**

****

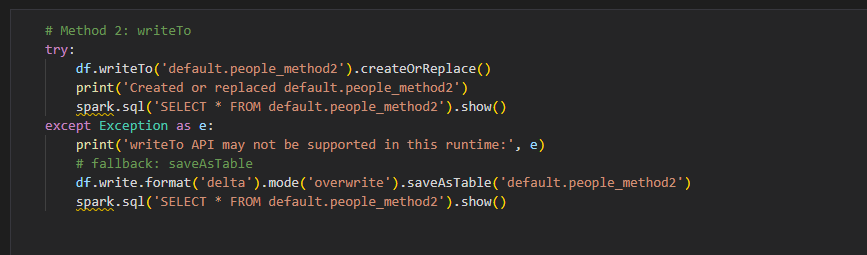
****

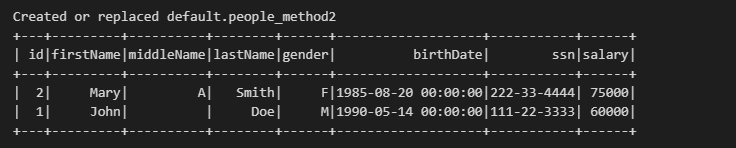
**Creating Delta Table — Method 1: `saveAsTable`**

****

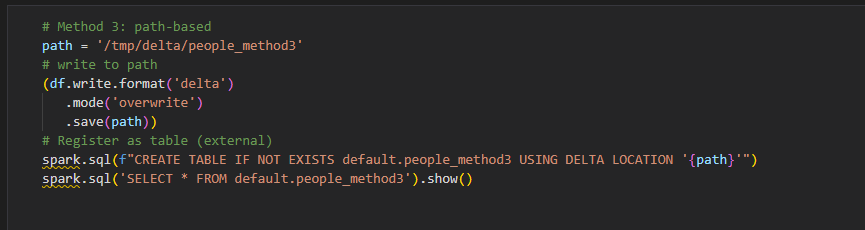
****

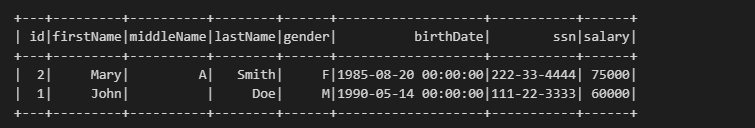
**Creating Delta Table — Method 2: `writeTo(...).createOrReplace()`**

****

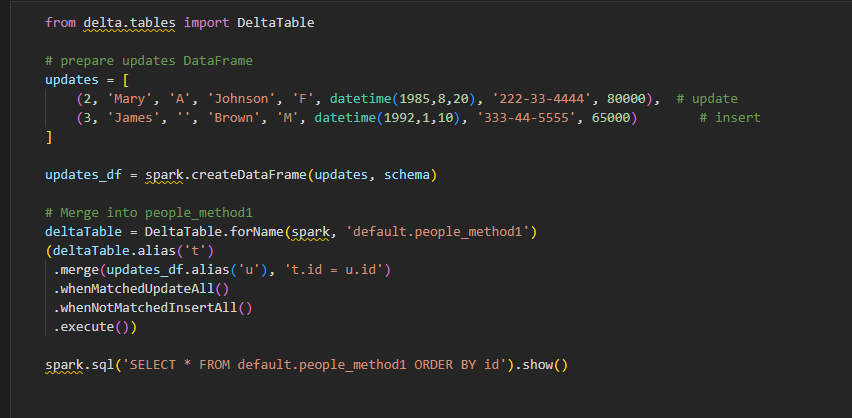
****

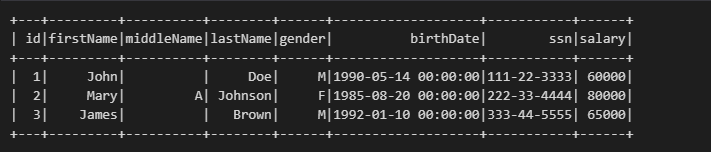
**Creating Delta Table — Method 3: Path-based Delta (external table)**

****

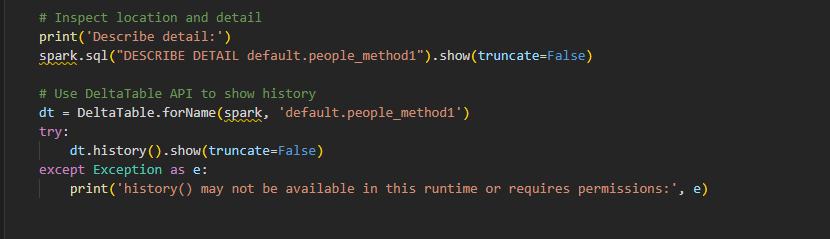
****

**Merge & Upsert (SCD Type 1 example)**

****

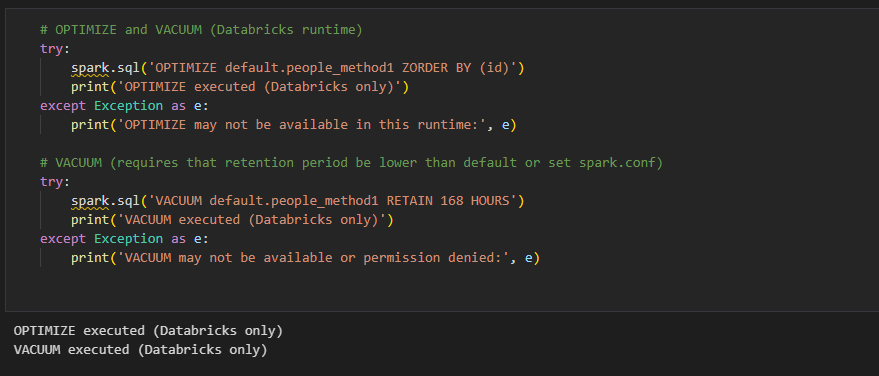
****

**Internals of Delta Table**

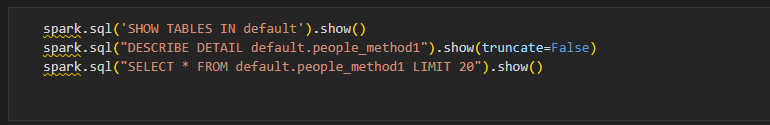
****

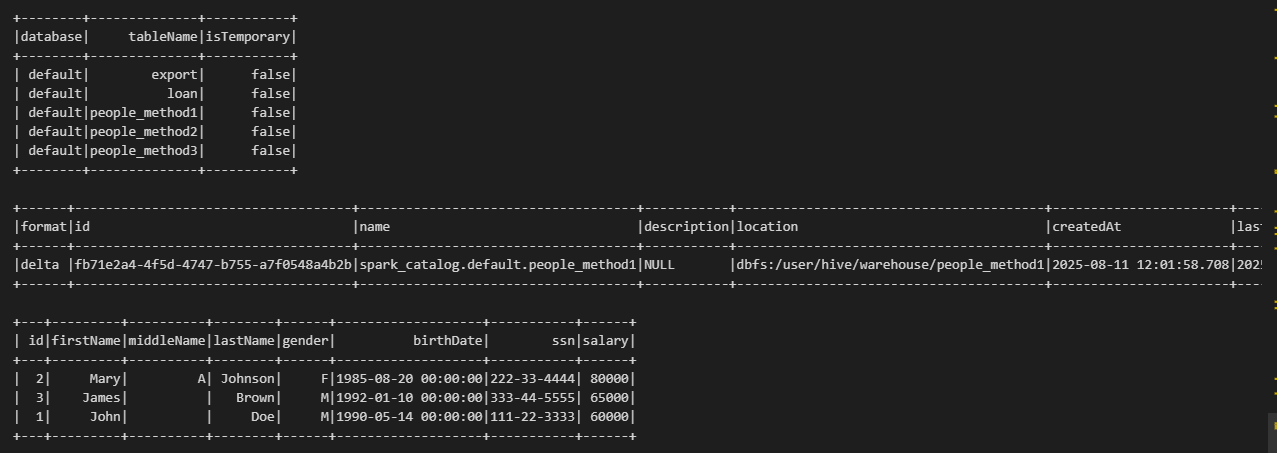
****

**Optimize Delta Table**

****

**Show Tables**

****

****