**Delta Lake - Stepwise Guide**

**1. Delta Log Anatomy**

**Purpose:** Maintain ACID properties and version history of the table.

**Step-by-Step:**

1. Each Delta table has a \_delta\_log/ directory containing transaction logs.
2. Every commit generates a JSON file with incremental changes (add/remove files, metadata updates).
3. Periodically, checkpoint files (in Parquet format) are created to speed up reads.
4. Log records include:
   * add / remove file actions
   * metaData describing schema and partitioning
   * protocol indicating minimum reader/writer versions

**Tools:** Spark SQL, dbutils.fs

**2. Schema Evolution**

**Purpose:** Allow changes in table structure without full rebuild.

**Step-by-Step:**

1. Enable schema evolution by setting mergeSchema = true in write options.
2. Append or overwrite data with new columns or changed types.
3. Verify schema changes with DESCRIBE DETAIL or DESCRIBE HISTORY.
4. Supported operations: Add column, change data type (upcasting).

**Commands:**

spark.write.option("mergeSchema", "true").format("delta").mode("append").save(path)

**3. OPTIMIZE with ZORDER**

**Purpose:** Improve performance through file compaction and data skipping.

**Step-by-Step:**

1. Use OPTIMIZE to compact small files.
2. Apply ZORDER BY to colocate frequently filtered columns.
3. Example:

OPTIMIZE sales ZORDER BY (region, sale\_date)

1. Analyze benefits with DESCRIBE HISTORY and Spark UI.

**Notes:** ZORDER doesn't affect data correctness, only layout.

**4. VACUUM and Retention**

**Purpose:** Remove obsolete data files and free up storage.

**Step-by-Step:**

1. Default retention: 7 days.
2. Run:

VACUUM tableName RETAIN 168 HOURS

1. To override retention safety:

SET spark.databricks.delta.retentionDurationCheck.enabled = false

1. Post-GDPR deletes must wait retention period + run VACUUM to fully erase data.

**Caution:** Always understand retention policy before running VACUUM.

**5. Time Travel and Restore**

**Purpose:** Query or revert to historical data state.

**Step-by-Step:**

1. Use version or timestamp-based queries:

SELECT \* FROM table VERSION AS OF 3

SELECT \* FROM table TIMESTAMP AS OF '2024-01-01'

1. To restore:

RESTORE TABLE my\_table TO VERSION AS OF 3

1. Validate with DESCRIBE HISTORY.

**Use Case:** Data audit, rollback, disaster recovery.

**6. Generated Columns**

**Purpose:** Define computed columns based on existing ones.

**Step-by-Step:**

1. Define column with expression:

col\_name GENERATED ALWAYS AS (expression) STORED

1. Values automatically computed and stored.
2. Supported for partitioning, indexing, consistency.

**Example:**

CREATE TABLE events (

ts TIMESTAMP,

day STRING GENERATED ALWAYS AS (CAST(ts AS DATE)) STORED

) USING DELTA

**7. GDPR Delete Pattern**

**Purpose:** Comply with privacy regulations via logical and physical deletion.

**Step-by-Step:**

1. Run delete command:

DELETE FROM users WHERE user\_id = 'abc123'

1. Wait for retention period (default 7 days).
2. Run VACUUM to physically delete files.
3. Optional: Tag sensitive columns with PII metadata.
4. Use masking/row filters as needed for soft protections.

**Notes:** Physical deletion occurs only after VACUUM.

**8. Delta Metadata Tables**

**Purpose:** Analyze table metadata, history, and lineage.

**Step-by-Step:**

1. Use system tables like:
   * delta.information\_schema.tables
   * delta.information\_schema.columns
   * delta.table\_history
   * delta.table\_files
2. Query with standard SQL.
3. Useful for audits, optimization, lineage tracing.

**Example:**

SELECT \* FROM delta.`my\_table` HISTORY

SELECT \* FROM delta.table\_files('my\_table')