**Delta Lake Practical Lab Assignments**

**Lab 1: Delta Log Anatomy**

**Objective:** Understand how Delta Lake tracks data changes using transaction logs.

**Tasks:**

* Create a Delta table and insert some data.
* Check the \_delta\_log/ directory contents using %fs ls or dbutils.
* Read a sample JSON transaction log using spark.read.json(...).
* Count how many actions (add/remove) are in the first transaction.

**Challenge:**

* Trigger multiple transactions (INSERT, DELETE) and identify which file represents each in the logs.

**Lab 2: Schema Evolution**

**Objective:** Practice safe schema evolution in Delta Lake.

**Tasks:**

* Create a Delta table with two columns and write initial data.
* Append a new DataFrame that adds a third column.
* Use .option("mergeSchema", "true") while writing.
* Read the table back and inspect the schema.

**Challenge:**

* Try appending a new DataFrame with a conflicting data type for an existing column and record the error message.

**Lab 3: OPTIMIZE ZORDER**

**Objective:** Learn how physical data clustering boosts query performance.

**Tasks:**

* Create a large Delta table with at least 1000 rows across multiple regions.
* Use OPTIMIZE ... ZORDER BY (region) to optimize file layout.
* Run a filtered query (e.g., region = 'US') before and after ZORDER and record execution time.

**Challenge:**

* Try ZORDERing by multiple columns and explain the difference it made in query planning.

**Lab 4: VACUUM & Retention**

**Objective:** Understand Delta file retention policies and garbage collection.

**Tasks:**

* Insert and delete some data from a Delta table.
* Run DESCRIBE HISTORY to check commit logs.
* Attempt to run VACUUM ... RETAIN 0 HOURS and observe the error.
* Set spark.databricks.delta.retentionDurationCheck.enabled = false and retry.

**Challenge:**

* Time-travel to a pre-deletion version after running VACUUM and explain the result.

**Lab 5: Time Travel & Restore**

**Objective:** Restore older table versions using Delta Time Travel.

**Tasks:**

* Perform at least three changes (INSERT/DELETE) to a Delta table.
* Use DESCRIBE HISTORY to get version numbers.
* Read the table using .option("versionAsOf", <version>).
* Restore an older version using CREATE OR REPLACE TABLE.

**Challenge:**

* Compare row counts and explain what changed between two specific versions.

**Lab 6: Generated Columns**

**Objective:** Implement computed columns in Delta with schema enforcement.

**Tasks:**

* Create a Delta table with a generated column that uppercases another column.
* Insert data into only the base column.
* Verify the generated column auto-fills on read.

**Challenge:**

* Try inserting a value into the generated column explicitly and log the error.
* Try querying using the generated column as a filter.

**Lab 7: GDPR Delete Pattern**

**Objective:** Permanently delete user data and validate Delta’s compliance capabilities.

**Tasks:**

* Create a table with personal data (e.g., name, email).
* DELETE a user by ID.
* Try accessing their record using standard queries and confirm it’s gone.
* Use time travel to verify it's still accessible.

**Challenge:**

* VACUUM the table with zero retention and prove the deleted record can no longer be recovered, even via time travel.

**Lab 8: Delta Metadata Tables**

**Objective:** Explore Delta Lake's internal metadata via built-in functions.

**Tasks:**

* Run and interpret SELECT \* FROM table\_changes(...).
* Run table\_snapshot() and table\_version() on any Delta table.
* Record the output and explain what each field means.

**Challenge:**

* Build a changelog audit view that uses table\_changes() for real-time data tracking.