**Lab Guide: Unity Catalog – Data Governance & Security**

**Lab 1: Catalog-Schema-Table ACL Hierarchy**

**Objective**

Set up access controls across catalog, schema, and tables.

**Steps:**

1. **Create Catalog & Schema**

sql

CREATE CATALOG sales\_catalog;

CREATE SCHEMA sales\_catalog.q3\_data;

1. **Create Table**

sql

CREATE TABLE sales\_catalog.q3\_data.transactions (

id STRING,

amount DOUBLE,

region STRING

);

1. **Grant Permissions**

sql

GRANT USE CATALOG ON CATALOG sales\_catalog TO `data\_analyst\_group`;

GRANT USE SCHEMA ON SCHEMA sales\_catalog.q3\_data TO `data\_analyst\_group`;

GRANT SELECT ON TABLE sales\_catalog.q3\_data.transactions TO `data\_analyst\_group`;

1. **Test Access**
   * Login as a user in data\_analyst\_group and run a SELECT query.

**Lab 2: Dynamic Row Filters**

**Objective**

Restrict access to rows dynamically based on the user’s region.

**Steps:**

1. **Define Function to Fetch User Region**

sql

CREATE OR REPLACE FUNCTION get\_user\_region() RETURNS STRING

RETURN current\_user\_region(); -- Assume an external function exists

1. **Create Row Filter**

sql

CREATE OR REPLACE ROW FILTER region\_filter AS (region = get\_user\_region());

1. **Apply Row Filter to Table**

sql

ALTER TABLE sales\_catalog.q3\_data.transactions

SET ROW FILTER region\_filter;

1. **Test**:
   * Login as different users and confirm they only see their own regional data.

**Lab 3: Column-Mask UDF**

**Objective**

Protect sensitive columns by applying masking based on user role.

**Steps:**

1. **Create Masking UDF**

sql

CREATE FUNCTION mask\_amount(amt DOUBLE) RETURNS STRING

RETURN IF(current\_user() IN ('admin'), amt, '\*\*\*MASKED\*\*\*');

1. **Apply to Table View**

sql

CREATE OR REPLACE VIEW transactions\_masked AS

SELECT id, mask\_amount(amount) AS amount, region

FROM sales\_catalog.q3\_data.transactions;

1. **Grant SELECT on View**

sql

GRANT SELECT ON VIEW transactions\_masked TO `readonly\_users`;

1. **Test**:
   * Verify that non-admins see masked amounts.

**Lab 4: Tagging PII & GDPR Flags**

**Objective**

Tag specific columns to indicate sensitive data for discovery and compliance.

**Steps:**

1. **Tag Columns**

sql

ALTER TABLE sales\_catalog.q3\_data.transactions

ALTER COLUMN id SET TAGS ('pii': 'true', 'gdpr': 'yes');

ALTER COLUMN amount SET TAGS ('financial': 'true');

1. **List Tags**

sql

DESCRIBE DETAIL sales\_catalog.q3\_data.transactions;

1. **Data Discovery Integration**
   * Run a scan using Unity Catalog Lineage or third-party scanner to confirm tags appear.

**Lab 5: Lineage Explorer**

**Objective**

Trace the data flow from raw ingestion to report.

**Steps:**

1. **Open Lineage Explorer**
   * Navigate to **Data > Unity Catalog > Lineage Explorer** in the Databricks UI.
   * Search for sales\_catalog.q3\_data.transactions.
2. **Observe**
   * Check upstream (notebooks, sources) and downstream (dashboards, views).
   * Click column-level lineage for amount.
3. **Exercise**
   * Create a transformation or downstream report view and verify it's added in lineage.

**Lab 6: Audit Logs → Log Analytics**

**Objective**

Track all access and permission changes for compliance and alerting.

**Steps:**

1. **Enable Audit Logs in Workspace Settings**
   * Navigate to Admin Console → Workspace Settings → Enable Audit Logs.
2. **Configure Destination**
   * Point logs to a secure location (e.g., Azure Log Analytics / AWS S3 / Splunk).
3. **Run Some Operations**
   * Create table, update permissions, perform data access.
4. **Query Logs**
   * In Log Analytics:

kql

DatabricksAuditLogs

| where ActionName == "select"

| summarize count() by UserEmail

1. **(Optional)**: Create Alerts
   * Alert on large read activity or unauthorized access.

**Wrap-Up Exercises**

1. **Create a new catalog with appropriate ACL hierarchy.**
2. **Apply row filter based on a user attribute like department.**
3. **Mask multiple columns using UDFs.**
4. **Add GDPR/PII tags and explore lineage integration.**
5. **Generate logs and validate events using Log Analytics dashboard.**