LAB 09: Governance & Security

Duration: ~40 min | **Day:** 3 | **After module:** M09: Governance & Security | **Difficulty:** Advanced

---

Scenario

> *"Implement fine-grained access controls for your Silver tables using Unity Catalog. Grant privileges, create column masks to protect PII, and apply row filters for team-based access. Finally, query INFORMATION_SCHEMA to audit permissions."*

Objectives

After completing this lab you will be able to: - Grant `USE CATALOG`, `USE SCHEMA`, and `SELECT` privileges - Query `INFORMATION_SCHEMA` for table and privilege metadata - Create and apply column mask functions (PII protection) - Create and apply row filter functions (team-based access) - Clean up masks and filters

---

Prerequisites

- Cluster running and attached to notebook
- Setup cell creates sample `customers` and `orders` tables in Silver schema

---

Tasks Overview

Open **`LAB_09_code.ipynb`** and complete the `# TODO` cells.

Task	What to do	Key concept
Task 1	Grant Privileges	`GRANT USE CATALOG`, `USE SCHEMA`, `SELECT` to group
Task 2	Query INFORMATION_SCHEMA	`INFORMATION_SCHEMA.TABLES` — list tables
Task 3	Create Column Mask Function	`is_account_group_member('admins')` conditional mask
Task 4	Apply Column Mask	`ALTER TABLE ... ALTER COLUMN email SET MASK mask_email`
Task 5	Create Row Filter Function	Filter rows by `store_region` per team
Task 6	Apply Row Filter	`ALTER TABLE ... SET ROW FILTER ... ON (store_region)`
Task 7	Query Table Privileges	`INFORMATION_SCHEMA.TABLE_PRIVILEGES`
Task 8	Remove Mask and Filter (Cleanup)	`DROP MASK`, `DROP ROW FILTER`

Detailed Hints

Task 1: Grant Privileges

Three-level access pattern: 1. `GRANT USE CATALOG ON CATALOG {catalog} TO \` analysts` `2.` GRANT USE SCHEMA ON SCHEMA {catalog}.{schema} TO `analysts` `3.` GRANT SELECT ON SCHEMA {catalog}.{schema} TO `analysts``

Task 2: INFORMATION_SCHEMA.TABLES

- View name: `TABLES`
- Filter: `WHERE table_schema = '{SILVER_SCHEMA}'`

Task 3: Column Mask Function

- Use `is_account_group_member('admins')` to check group membership
- Admins see the real email; others see masked version
- Mask format: `CONCAT(LEFT(email, 2), '***@***.***')`

Task 4: Apply Mask

- `ALTER TABLE ... ALTER COLUMN email SET MASK {catalog}.{schema}.mask_email`

Task 5: Row Filter Function

- Admins see all rows (`is_account_group_member('admins')`)
- East team sees only `region = 'East'`
- West team sees only `region = 'West'`
- Use `OR` to combine conditions

Task 6: Apply Row Filter

- `ALTER TABLE ... SET ROW FILTER {catalog}.{schema}.region_filter ON (store_region)`
- The `ON (column)` clause specifies which column maps to the function parameter

Task 7: TABLE_PRIVILEGES

- View name: `TABLE_PRIVILEGES`

Task 8: Cleanup

- `ALTER TABLE ... ALTER COLUMN email DROP MASK`
- `ALTER TABLE ... DROP ROW FILTER`

Summary

In this lab you: - Granted three-level privileges (CATALOG → SCHEMA → SELECT) - Queried INFORMATION_SCHEMA for metadata discovery - Created and applied column masks for PII protection - Created and applied row filters for team-based access - Cleaned up governance objects

Governance feature	Purpose	Key syntax
GRANT	Control access	`GRANT privilege ON object TO principal`
Column Mask	Hide PII	`ALTER COLUMN ... SET MASK function`
Row Filter	Restrict rows	`SET ROW FILTER function ON (column)`
INFORMATION_SCHEMA	Audit	`TABLES`, `TABLE_PRIVILEGES`, `COLUMNS`

> **Exam Tip:** Three-level access: `USE CATALOG` + `USE SCHEMA` + `SELECT`. Column masks and row filters use SQL UDFs with `is_account_group_member()`. `INFORMATION_SCHEMA` views are the standard way to discover and audit metadata.

> **Congratulations!** You have completed all 9 labs of the Databricks Data Engineer Associate training.