Exercise: Mini Project Using Unity Catalog and Data Governance

Part 1: Setting Up the Environment

Task 1: Create a Metastore

>> CREATE METASTORE central_metastore;

Task 2: Create Department-Specific Catalogs

>> CREATE CATALOG marketing;

>> CREATE CATALOG engineering;

>> CREATE CATALOG operations;

Task 3: Create Schemas for Each Department

1. For the Marketing catalog, create schemas such as ads_data and customer_data .

>> CREATE SCHEMA marketing.ads_data;

>> CREATE SCHEMA marketing.customer_data;

2. For the Engineering catalog, create schemas such as projects and development_data .

>> CREATE SCHEMA engineering.projects;

>> CREATE SCHEMA engineering.development_data;

3. For the Operations catalog, create schemas such as logistics_data and supply_chain

>> CREATE SCHEMA operations.logistics_data;

>> CREATE SCHEMA operations.supply_chain;

 # Verify the Schemas

>> SHOW SCHEMAS IN marketing;

>> SHOW SCHEMAS IN engineering;

>> SHOW SCHEMAS IN operations;

Part 2: Loading Data and Creating Tables

Task 4: Prepare Datasets (Use CSV or JSON files)

1. **Marketing - Ads Data** (ads_data.csv):

ad_id	impressions	clicks	cost_per_click
101	1500	45	0.50
102	2000	60	0.40
103	2500	80	0.60

2. **Engineering - Projects** (projects.csv):

project_id	project_name	start_date	end_date
201	Data Warehouse	2024-01-01	2024-06-30
202	API Development	2024-02-15	2024-05-31
203	Mobile App	2024-03-10	2024-09-30

3. **Operations - Logistics** (logistics_data.csv):

shipment_id	origin	destination	status
301	New York	Los Angeles	Delivered
302	Houston	Miami	In Transit
303	Chicago	Boston	Pending

Load csv files into databricks

dbutils.fs.cp("file:/Workspace/Shared/ads_data.csv","dbfs:/Filestore/ads_data.csv")

dbutils.fs.cp("file:/Workspace/Shared/projects.csv","dbfs:/Filestore/projects.csv")

dbutils.fs.cp("file:/Workspace/Shared/logistics_data.csv","dbfs:/Filestore/logistics_data.csv")

Task 5: Create Tables from the Datasets

>> Create a table for ads_data in the marketing catalog.

```
CREATE TABLE marketing.ads_data.marketing_sales (
  ad_id INT,
  impressions INT,
  clicks INT,
  cost_per_click DOUBLE
)
USING csv
OPTIONS (path 'dbfs:/FileStore/ads_data.csv', header = 'true');
```

>> Create a table for projects in the engineering catalog.

```
CREATE TABLE engineering.projects.engineering_project (
  project_id INT,
  project_name STRING,
  start_date DATE,
  end_date DATE
)
USING csv
OPTIONS (path 'dbfs:/FileStore/projects.csv', header = 'true');
```

>> Create the Logistics Data Table in Operations Catalog

```
CREATE TABLE operations.logistics_data.transportation_details (
  shipment_id INT,
  origin STRING,
  destination STRING,
  status STRING
)
USING csv
OPTIONS (path 'dbfs:/FileStore/logistics_data.csv', header = 'true');
```

Part 3: Data Governance Capabilities

Task 6: Create Roles and Grant Access

Step 1: Create Roles

>> CREATE ROLE marketing_role;

>> CREATE ROLE engineering_role;

>> CREATE ROLE operations_role;

Step 2: Grant Access to Catalogs and Schemas

1. **Grant Access to Marketing Role**

>> GRANT USAGE ON CATALOG marketing TO ROLE marketing_role;

>> GRANT USAGE ON SCHEMA marketing.ads_data TO ROLE marketing_role;

>> GRANT USAGE ON SCHEMA marketing.customer_data TO ROLE marketing_role;

2. **Grant Access to Engineering Role**

>> GRANT USAGE ON CATALOG engineering TO ROLE engineering_role;

>> GRANT USAGE ON SCHEMA engineering.projects TO ROLE engineering_role;

>> GRANT USAGE ON SCHEMA engineering.development_data TO ROLE engineering_role;

3. **Grant Access to Operations Role**

>> GRANT USAGE ON CATALOG operations TO ROLE operations_role;

>> GRANT USAGE ON SCHEMA operations.logistics_data TO ROLE operations_role;

>> GRANT USAGE ON SCHEMA operations.supply_chain TO ROLE operations_role;

 # Grant Select Permissions for the roles

>> GRANT SELECT ON TABLE marketing.ads_data.marketing_sales TO ROLE marketing_role;

>> GRANT SELECT ON TABLE engineering.projects.engineering_project TO ROLE engineering_role;

>> GRANT SELECT ON TABLE operations.logistics_data.transportation_details TO ROLE operations_role;

Task 7: Configure Fine-Grained Access Control

Step 1: Restrict Access to Specific Schemas or Tables

1. **Restrict Marketing Role to Only Customer Data**

>> REVOKE USAGE ON SCHEMA marketing.ads_data FROM ROLE marketing_role;

>> GRANT USAGE ON SCHEMA marketing.customer_data TO ROLE marketing_role;

2. Restrict Engineering Role to Project Data

>> REVOKE USAGE ON SCHEMA engineering.development_data FROM ROLE engineering_role;

>> GRANT USAGE ON SCHEMA engineering.projects TO ROLE engineering_role;

3. Operations Role Specific Access

>> REVOKE USAGE ON SCHEMA operations.supply_chain FROM ROLE operations_role;

>> GRANT USAGE ON SCHEMA operations.logistics_data TO ROLE operations_role;

Task 8: Enable and Explore Data Lineage

 # Perform Queries and Track Data Lineage

1. Marketing Catalog

```
SELECT ad_id, SUM(clicks) AS total_clicks, AVG(cost_per_click) AS avg_cost
FROM marketing.ads_data. .marketing_sales
GROUP BY ad_id;
```

2. Engineering Catalog

```
SELECT project_name, QUARTER(start_date) AS start_quarter
FROM engineering.projects.engineering_project;
```

3. Operations Catalog

```
SELECT status, COUNT(*) AS shipment_count
FROM operations.logistics_data.transportation_details
GROUP BY status;
```

Task 9: Monitor Data Access and Modifications

Step 1: Azure Diagnostic logs configuration

>> In the databricks workspace, go to diagnostic settings and enable logging for categories like Workspace, Clusters, and SQL Queries.

 # Configuring unity catalog to send logs

>> In Databricks, navigate to the admin console > audit logs, there we can set up a log delivery to a cloud storage location

>> These logs can contain details such as userID, timestamp, actions performed like SELECT and INSERT and the objects that are accessed.

Step 2: Monitor Data Access Patterns

1. **View logs**

>> Go to location where the audit logs are being delivered.

>> Each entry log contain details like:

- UserID
- TimeStamp of the action
- Action type
- Resources affected
- Operation details

Task 10: Explore Metadata in Unity Catalog

Step 1: Explore Metadata for Tables and Schemas

1. **Retrieve Table Schema**

>> DESCRIBE TABLE marketing.ads_data.marketing_sales;

>> DESCRIBE TABLE engineering.projects.engineering_project

>> DESCRIBE TABLE operations.logistics_data.transportation_details

2. **Check Number of Rows in Tables**

>> SELECT COUNT(*) FROM marketing.ads_data.marketing_sales;

>> SELECT COUNT(*) FROM engineering.projects.engineering_project

>> SELECT COUNT(*) FROM operations.logistics_data.transportation_details

3. **View Table Properties**

>> DESCRIBE EXTENDED marketing.ads_data.marketing_sales;

>> DESCRIBE EXTENDED engineering.projects.engineering_project;

>> DESCRIBE EXTENDED operations.logistics_data.transportation_details;

Step 2: Add Descriptions and Properties

1. **Add Descriptions to Catalogs**

>> ALTER CATALOG marketing SET PROPERTIES ('description' = 'Catalog for marketing department, containing ads and customer data.');

2. **Add Descriptions to Schemas**

>> ALTER SCHEMA marketing.ads_data SET PROPERTIES ('description' = 'Schema for storing marketing advertisements data.');

3. **Add Descriptions to Tables**

>> ALTER TABLE marketing.ads_data.marketing_sales SET PROPERTIES ('description' = 'Table storing ad performance data including impressions, clicks, and cost per click.');