Mini Project: Data Governance Using Unity Catalog- Advanced Capabilities

Task 1: Set Up Unity Catalog Objects with Multiple Schemas

```
-- 1. Create a Catalog

CREATE CATALOG finance_data_catalog;

-- 2. Create Schemas inside the Catalog

CREATE SCHEMA finance_data_catalog.transaction_data;

CREATE SCHEMA finance_data_catalog.customer_data;

-- 3. Create Tables in Each Schema

-- Table for transaction_data schema

CREATE TABLE finance_data_catalog.transaction_data.transactions (

   TransactionID STRING,

   CustomerID STRING,

   TransactionAmount DECIMAL(10, 2),

   TransactionDate DATE

);

-- Table for customer_data schema

CREATE TABLE finance_data_catalog.customer_data.customers (

   CustomerID STRING,

   CustomerName STRING,

   Email STRING,

   Country STRING

);
```

Task 2: Data Discovery & Profiling

```
-- 1. Explore metadata

DESCRIBE TABLE finance_data_catalog.transaction_data.transactions;

DESCRIBE TABLE finance_data_catalog.customer_data.customers;

-- 2. Data profiling

-- Summary statistics for transaction amounts

SELECT

   MIN(TransactionAmount) as MinAmount,
    MAX(TransactionAmount) as MaxAmount,
    AVG(TransactionAmount) as AvgAmount
FROM finance_data_catalog.transaction_data.transactions;
-- Discover customer locations
SELECT Country, COUNT(*) as CustomerCount
FROM finance_data_catalog.customer_data.customers
GROUP BY Country;
--Transaction counts over time
SELECT TransactionDate,
COUNT(*) AS TotalTransactions
FROM finance_data_catalog.transaction_data.transactions
GROUP BY TransactionDate
ORDER BY TransactionDate;
-- 3. Tagging Sensitive Data
-- Adding Sensitive Data Tag for Customer Email
ALTER TABLE finance_data_catalog.customer_data.customers
ADD TAG (sensitive='true') FOR COLUMN Email;
-- Adding Sensitive Data Tag for Transaction Amount
ALTER TABLE finance_data_catalog.transaction_data.transactions
ADD TAG (sensitive='true') FOR COLUMN TransactionAmount;
```

Task 3: Implement Data Lineage and Auditing

```
-- 1. Track Data Lineage
CREATE OR REPLACE VIEW finance_data_catalog.transaction_summary AS
SELECT t.TransactionID, t.TransactionAmount, c.CustomerName, c.Email
FROM finance_data_catalog.transaction_data.transactions t
JOIN finance_data_catalog.customer_data.customers c
ON t.CustomerID = c.CustomerID;
```

- Navigate to Data Explorer in Databricks, access Unity Catalog to view data lineage, and track changes over time.

-- 2. Audit User Actions

1. Activate Audit Logs:

    - Access the Admin Console in Databricks.

- Head to the Audit Logs section and turn on audit logging for operations on tables.

2. Monitor Data Access and Changes:

   - After enabling audit logs, you can track user activities, such as:

     - Who viewed or queried the tables.
     - Who made changes like inserts, updates, or deletions on the tables.

Task 4: Access Control and Permissions

1. Set Up Roles and Groups

- - Create two groups: DataEngineers and DataAnalysts

CREATE GROUP DataEngineers;

CREATE GROUP DataAnalysts;

- - Assign appropriate roles

- - For data engineers full access

GRANT ALL PRIVILEGES ON SCHEMA finance_data_catalog.transaction_data

TO `DataEngineers`;

GRANT ALL PRIVILEGES ON SCHEMA finance_data_catalog.customer_data

TO `DataEngineers`;

GRANT ALL PRIVILEGES ON TABLE finance_data_catalog.transaction_data.transactions

TO `DataEngineers`;

GRANT ALL PRIVILEGES ON TABLE finance_data_catalog.customer_data.customers

TO `DataEngineers`;

- - For data analysts Read-only access

GRANT SELECT ON SCHEMA finance_data_catalog.customer_data

TO `DataAnalysts`;

GRANT SELECT ON TABLE finance_data_catalog.customer_data.customers

TO `DataAnalysts`;

GRANT SELECT ON TABLE finance_data_catalog.transaction_data.transactions

TO `DataAnalysts`;

2. Row-Level Security

- - Create a Dynamic View for High-Value Transactions

CREATE OR REPLACE VIEW finance_data_catalog.transaction_data.secure_transactions AS

```
SELECT * FROM finance_data_catalog.transaction_data.transactions

WHERE (TransactionAmount <= 10000)

OR (TransactionAmount > 10000 AND CURRENT_USER() IN ('authorized_user1', 'authorized_user2'));

- - Restrict Access to the Original Table

REVOKE SELECT ON TABLE finance_data_catalog.transaction_data.transactions

FROM `DataAnalysts`;

GRANT SELECT ON VIEW finance_data_catalog.transaction_data.secure_transactions

TO `DataAnalysts`;
```

Task 5: Data Governance Best Practices

1.Create Data Quality Rules

```
-- Transaction Amounts are Non-Negative

SELECT * FROM finance_data_catalog.transaction_data.transactions

WHERE TransactionAmount < 0;

-- Customer emails follow the correct format

SELECT * FROM finance_data_catalog.customer_data.customers

WHERE Email NOT LIKE '%_@__%.__%';
```

2. Validate Data Governance

a) Validate Data Quality Rules:

Transaction Amount Validation:

Ensure no transactions have negative amounts using the first query above.

Email Format Validation:

Ensure customer emails follow the correct format using the second query above.

b) Validate Data Lineage:

```
-- View data lineage between the customer and transaction tables

DESCRIBE HISTORY finance_data_catalog.transaction_summary;

- - Verify Audit Logs

SELECT eventName,userIdentity, objectName, actionName, timestamp

FROM <audit_log_table>

WHERE objectName IN
```

```
('finance_data_catalog.transaction_data.transactions',

'finance_data_catalog.customer_data.customers')

AND actionName IN ('INSERT', 'UPDATE');
```

Task 6: Data Lifecycle Management

1. Implement Time Travel

```
SELECT * FROM finance_data_catalog.transaction_data.transactions

VERSION AS OF 1;

RESTORE TABLE finance_data_catalog.transaction_data.transactions

TO VERSION AS OF 5;
```

2. Run a Vacuum Operation

```
VACUUM finance_data_catalog.transaction_data.transactions RETAIN 168 HOURS;

VACUUM finance_data_catalog.customer_data.customers RETAIN 168 HOURS;
```