Mini Project: Advanced Data Governance and Security Using Unity Catalog

Task 1: Set Up Multi-Tenant Data Architecture Using Unity Catalog

1. Create a New Catalog:

```
CREATE CATALOG corporate_data_catalog;
```

2. Create Schemas for Each Department:

```
CREATE SCHEMA corporate_data_catalog.sales_data;
CREATE SCHEMA corporate_data_catalog.hr_data;
CREATE SCHEMA corporate_data_catalog.finance_data;
```

3. Create Tables in Each Schema:

a) For sales_data Schema:

```
CREATE TABLE corporate_data_catalog.sales_data.sales (
    SalesID STRING,
    CustomerID STRING,
    SalesAmount DECIMAL(10, 2),
    SalesDate DATE
);
```

b) For hr_data Schema:

```
CREATE TABLE corporate_data_catalog.hr_data.employees (
    EmployeeID STRING,
    EmployeeName STRING,
    Department STRING,
    Salary DECIMAL(10, 2)
);
```

c) For finance_data Schema:

```
CREATE TABLE corporate_data_catalog.finance_data.invoices (
    InvoiceID STRING,
    VendorID STRING,
    InvoiceAmount DECIMAL(10, 2),
    PaymentDate DATE
);
```

Task 2: Enable Data Discovery for Cross-Departmental Data

1. Search for Tables Across Departments

```
SHOW TABLES IN corporate_data_catalog.sales_data;

SHOW TABLES IN corporate_data_catalog.hr_data;

SHOW TABLES IN corporate_data_catalog.finance_data;
```

2. Tag Sensitive Information

a) Tag the Salary column in the hr_data schema:

```
ALTER TABLE corporate_data_catalog.hr_data.employees
ADD TAG (sensitive='true') FOR COLUMN Salary;
```

b) Tag the InvoiceAmount column in the finance_data schema:

```
ALTER TABLE corporate_data_catalog.finance_data.invoices
ADD TAG (sensitive='true') FOR COLUMN InvoiceAmount;
```

3. Data Profiling

a) Analyze Trends in Sales (sales_data):

```
SELECT
   SUM(SalesAmount) AS TotalSales,
   AVG(SalesAmount) AS AvgSalesAmount,
   MIN(SalesAmount) AS MinSalesAmount,
   MAX(SalesAmount) AS MaxSalesAmount
FROM corporate_data_catalog.sales_data.sales;
```

b) Analyze Employee Salaries (hr_data):

```
SELECT
   AVG(Salary) AS AvgSalary,
   SUM(Salary) AS TotalSalaries,
   MIN(Salary) AS MinSalary,
   MAX(Salary) AS MaxSalary
FROM corporate_data_catalog.hr_data.employees;
```

c) Analyze Financial Transactions (finance_data):

```
SELECT
   SUM(InvoiceAmount) AS TotalInvoices,
   AVG(InvoiceAmount) AS AvgInvoiceAmount,
   MIN(InvoiceAmount) AS MinInvoiceAmount,
```

MAX(InvoiceAmount) AS MaxInvoiceAmount

FROM corporate_data_catalog.finance_data.invoices;

Task 3: Implement Data Lineage and Data Auditing

a) Create a Reporting Table Merging Sales and Finance Data:

CREATE OR REPLACE TABLE corporate_data_catalog.reporting.sales_finance_report AS

SELECT s.SalesID, s.CustomerID, s.SalesAmount, s.SalesDate, f.InvoiceID, f.InvoiceAmount, f.PaymentDate

FROM corporate_data_catalog.sales_data.sales s

JOIN corporate_data_catalog.finance_data.invoices f

ON s.CustomerID = f.VendorID;

b) Track Data Lineage in Unity Catalog:

- Unity Catalog automatically tracks the data lineage at the table, view, and column levels. To visualize the flow, use the Unity Catalog UI or CLI tools for lineage tracking:
- View the lineage from the sales_finance_report table in the Unity Catalog interface to trace back to the source tables (sales_data and finance_data).

2. Enable Data Audit Logs

a) Enable Audit Logs for hr_data and finance_data Tables:

- To ensure that audit logs are enabled, you'll need to configure Unity Catalog to capture these logs. This is typically done at the account or workspace level through the Databricks admin console.
- Once configured, Unity Catalog captures all operations performed on tables such as:

Reads: Who queried the data

Writes: Who inserted/updated/deleted data

Updates: Changes made to the data

Task 4: Data Access Control and Security

1. Set Up Roles and Permissions

a) Create Groups and Assign Schema-Level Permissions:

i. SalesTeam: Access to the sales_data schema only

GRANT SELECT ON SCHEMA corporate_data_catalog.sales_data TO `SalesTeam`;

ii. FinanceTeam: Access to sales_data and finance_data schemas

GRANT SELECT ON SCHEMA corporate_data_catalog.sales_data TO `FinanceTeam`;

GRANT SELECT ON SCHEMA corporate_data_catalog.finance_data TO `FinanceTeam`;

iii. HRTeam: Access to hr_data schema with permission to update employee records

GRANT SELECT, UPDATE ON SCHEMA corporate_data_catalog.hr_data TO `HRTeam`;

2. Implement Column-Level Security

a) Restrict Access to the Salary Column for Non-HR Managers:

```
-- Create a role for HR managers with access to the Salary column
CREATE ROLE HRManager;
-- Grant SELECT access only to HRManager role for the Salary column
GRANT SELECT (Salary) ON TABLE corporate_data_catalog.hr_data.employees TO `HRManager`;
-- Revoke access to the Salary column for non-HR users
REVOKE SELECT (Salary) ON TABLE corporate_data_catalog.hr_data.employees FROM PUBLIC;
```

3. Implement Row-Level Security

```
-- Create a policy that restricts access to sales records based on the SalesID
CREATE ROW POLICY sales_rep_policy
ON corporate_data_catalog.sales_data.sales
FOR SELECT USING (SalesID = current_user());
```

Task 5: Data Governance Best Practices

1. Define Data Quality Rules

a) Ensure Sales Amounts are Positive in the sales_data Table

```
-- Check for any negative or zero sales amounts
SELECT *
FROM corporate_data_catalog.sales_data.sales
WHERE SalesAmount <= 0;
-- Implement validation to prevent negative or zero sales amounts
ALTER TABLE corporate_data_catalog.sales_data.sales
ADD CONSTRAINT positive_sales_amount CHECK (SalesAmount > 0);
```

b) Ensure Employee Salaries are Greater Than Zero in the hr_data Table

```
-- Check for any salaries that are zero or negative
SELECT *
FROM corporate_data_catalog.hr_data.employees
WHERE Salary <= 0;
-- Implement validation to prevent non-positive salaries
ALTER TABLE corporate_data_catalog.hr_data.employees
ADD CONSTRAINT positive_salary CHECK (Salary > 0);
```

c) Ensure Invoice Amounts in the finance_data Table Match Payment Records

```
SELECT *
FROM corporate_data_catalog.finance_data.invoices i
LEFT JOIN corporate_data_catalog.finance_data.payments p
    ON i.InvoiceID = p.InvoiceID
WHERE i.InvoiceAmount != p.PaymentAmount;
```

2. Apply Time Travel for Data Auditing

Step 1: View Delta Table History

```
DESCRIBE HISTORY corporate_data_catalog.finance_data.invoices;
```

Step 2: Restore the Table to a Previous Version

```
RESTORE TABLE corporate_data_catalog.finance_data.invoices TO VERSION AS OF 1;
```

Task 6: Optimize and Clean Up Delta Tables

1. Optimize Delta Tables

a) Optimize the sales_data Table

```
OPTIMIZE corporate_data_catalog.sales_data.sales;
```

b) Optimize the finance_data Table

```
OPTIMIZE corporate_data_catalog.finance_data.invoices;
```

2. Vacuum Delta Tables

a) Vacuum the sales_data Table

```
VACUUM corporate_data_catalog.sales_data.sales RETAIN 168 HOURS;
```

b) Vacuum the finance_data Table

```
VACUUM corporate_data_catalog.finance_data.invoices RETAIN 168 HOURS;
```