Mini Project: Building a Secure Data Platform with Unity Catalog

Task 1: Set Up Unity Catalog for Multi-Domain Data Management

1. Create a New Catalog

```
CREATE CATALOG enterprise_data_catalog;
```

2. Create Domain-Specific Schemas

```
CREATE SCHEMA enterprise_data_catalog.marketing_data;

CREATE SCHEMA enterprise_data_catalog.operations_data;

CREATE SCHEMA enterprise_data_catalog.it_data;
```

3. Create Tables in Each Schema

a) For marketing_data Schema:

```
CREATE TABLE enterprise_data_catalog.marketing_data.campaigns (
    CampaignID STRING,
    CampaignName STRING,
    Budget DECIMAL(10, 2),
    StartDate DATE
);
```

b) For operations_data Schema:

```
CREATE TABLE enterprise_data_catalog.operations_data.orders (
    OrderID STRING,
    ProductID STRING,
    Quantity INT,
    ShippingStatus STRING
);
```

c) For it_data Schema:

```
CREATE TABLE enterprise_data_catalog.it_data.incidents (
    IncidentID STRING,
    ReportedBy STRING,
    IssueType STRING,
    ResolutionTime DECIMAL(10, 2)
);
```

Task 2: Data Discovery and Classification

1. Search for Data Across Schemas

Listing All Tables in the Catalog:

```
SHOW TABLES IN enterprise_data_catalog.marketing_data;

SHOW TABLES IN enterprise_data_catalog.operations_data;

SHOW TABLES IN enterprise_data_catalog.it_data;
```

Searching for Tables Based on Data Types:

```
SELECT table_name

FROM enterprise_data_catalog.information_schema.columns

WHERE column_name IN ('Budget', 'ResolutionTime');
```

2. Tag Sensitive Information

Tagging the Budget Column in marketing_data:

```
ALTER TABLE enterprise_data_catalog.marketing_data.campaigns

ADD COLUMN TAGS ('sensitive') FOR (Budget);
```

Tagging the ResolutionTime Column in it_data:

```
ALTER TABLE enterprise_data_catalog.it_data.incidents

ADD COLUMN TAGS ('sensitive') FOR (ResolutionTime);
```

3. Data Profiling

Profiling Marketing Budgets:

```
SELECT

   AVG(Budget) AS avg_budget,

   MIN(Budget) AS min_budget,

   MAX(Budget) AS max_budget,

   COUNT(*) AS total_campaigns

FROM enterprise_data_catalog.marketing_data.campaigns;
```

Profiling Operational Shipping Statuses:

```
SELECT

   ShippingStatus,

   COUNT(*) AS order_count

FROM enterprise_data_catalog.operations_data.orders
```

```
GROUP BY ShippingStatus

ORDER BY ShippingStatus;
```

Task 3: Data Lineage and Auditing

1.Track Data Lineage Across Schemas

```
SELECT  m.CampaignID, m.CampaignName, m.Budget, o.OrderID, o.ProductID, o.Quantity, o.ShippingStatus

FROM enterprise_data_catalog.marketing_data.campaigns m

JOIN enterprise_data_catalog.operations_data.orders o

ON m.CampaignID = o.ProductID;
```

2. Enable and Analyze Audit Logs

```
SELECT timestamp, user_name, operation, object_name, object_type

FROM audit_logs

WHERE object_name LIKE 'enterprise_data_catalog.it_data.%'  -- Filter for the it_data schema

ORDER BY timestamp DESC;
```

Task 4: Implement Fine-Grained Access Control

1. Create User Roles and Groups

```
CREATE GROUP MarketingTeam;

CREATE GROUP OperationsTeam;

CREATE GROUP ITSupportTeam;
```

Assigning Permissions:

```
GRANT USAGE ON SCHEMA enterprise_data_catalog.marketing_data TO `MarketingTeam`;

GRANT USAGE ON SCHEMA enterprise_data_catalog.operations_data TO `OperationsTeam`;

GRANT USAGE ON SCHEMA enterprise_data_catalog.marketing_data TO `OperationsTeam`;

GRANT UPDATE ON TABLE enterprise_data_catalog.it_data.incidents TO `ITSupportTeam`;
```

2. Implement Column-Level Security

```
-- Deny access to the Budget column for other teams

REVOKE SELECT ON COLUMN enterprise_data_catalog.marketing_data.campaigns.Budget FROM `OperationsTeam`, `ITSupportTeam`;

-- Grant access to the Budget column for MarketingTeam

GRANT SELECT ON COLUMN enterprise_data_catalog.marketing_data.campaigns.Budget TO `MarketingTeam`;
```

3. Row-Level Security

```
CREATE ROW ACCESS POLICY operations_team_access_policy
AS (user_name STRING) RETURNS BOOLEAN
AS (Department = 'Operations');  -- Adjust based on how you identify users and departments

ALTER TABLE enterprise_data_catalog.operations_data.orders
ADD ROW ACCESS POLICY operations_team_access_policy;
```

Task 5: Data Governance and Quality Enforcement

1. Set Data Quality Rules

```
ALTER TABLE enterprise_data_catalog.marketing_data.campaigns
ADD CONSTRAINT check_budget_positive CHECK (Budget > 0);

ALTER TABLE enterprise_data_catalog.operations_data.orders
ADD CONSTRAINT check_shipping_status
CHECK (ShippingStatus IN ('Pending', 'Shipped', 'Delivered'));

ALTER TABLE enterprise_data_catalog.it_data.incidents
ADD CONSTRAINT check_resolution_time CHECK (ResolutionTime >= 0);
```

2. Apply Delta Lake Time Travel

```
DESCRIBE HISTORY enterprise_data_catalog.operations_data.orders;
RESTORE TABLE enterprise_data_catalog.operations_data.orders TO VERSION AS OF 0;
```

Task 6: Performance Optimization and Data Cleanup

Optimizing the operations_data Table:

```
OPTIMIZE enterprise_data_catalog.operations_data.orders;
```

Vacuuming the it_data Table:

```
VACUUM enterprise_data_catalog.it_data.incidents RETAIN 168 HOURS;
```