Assignment 1:

Task 1: Set Up Unity Catalog Objects with Multiple Schemas:

1. **Create a Catalog:**
   - CREATE CATALOG finance_data_catalog;

2. **Create Multiple Schemas:**
   - CREATE SCHEMA finance_data_catalog.transaction_data;
   - CREATE SCHEMA finance_data_catalog.customer_data;

3. **Create Tables:**
   - CREATE TABLE finance_data_catalog.transaction_data.transactions (
     TransactionID int,
     CustomerID int,
     TransactionAmount double,
     TransactionDate date);
   - CREATE TABLE finance_data_catalog.transaction_data.customers (
     CustomerID int,
     CustomerName string,
     Email string,
     Country string);

Task 2: Data Discovery Across Schemas

1. **Explore Metadata:**
   - DESCRIBE TABLE finance_data_catalog.transaction_data.transactions;
   - DESCRIBE TABLE finance_data_catalog.customer_data.customers;

2. **Data Profiling:**
   - SELECT COUNT(*) AS TotalTransactions,
     AVG(TransactionAmount) AS AvgTransactionAmount,
     MIN(TransactionAmount) AS MinTransactionAmount,
     MAX(TransactionAmount) AS MaxTransactionAmount
     FROM finance_data_catalog.transaction_data.transactions;

   - SELECT Country, COUNT(*) AS NumberOfCustomers
     FROM finance_data_catalog.customer_data.customers
     GROUP BY Country
     ORDER BY NumberOfCustomers DESC;

3. **Tagging Sensitive Data:**
   - ALTER TABLE finance_data_catalog.customer_data.customers
     ALTER COLUMN Email SET TAG 'sensitive_data' = 'true';
   - ALTER TABLE finance_data_catalog.transaction_data.transactions
     ALTER COLUMN TransactionAmount SET TAG ‘sensitive_data’ = ‘true’;

Task 3: Implement Data Lineage and Auditing:

1. **Track Data Lineage:**
   - Merging the data
   - CREATE OR REPLACE VIEW finance_data_catalog.comprehensive_view AS
     SELECT t.TransactionID, t.CustomerID, c.CustomerName, c.Email, c.Country,
     t.TransactionAmount, t.TransactionDate
     FROM finance_data_catalog.transaction_data.transactions t
     JOIN finance_data_catalog.transaction_data.customers c
     ON t.CustomerID = c.CustomerID;
   - Navigate to Databricks UI under Catalog explorer to check the lineage of the view

2. Audit Logs:
   - Navigate to the admin console to enable the audit logs and view the operations performed

Task 4: Access Control and Permissions

1. Set up roles:
   - CREATE ROLE DataEngineers;
   - CREATE ROLE DataAnalysts;

     Assigning Roles:
   - 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 SELECT ON SCHEMA finance_data_catalog.customer_data TO DataAnalysts;
   - GRANT SELECT ON SCHEMA finance_data_catalog.transaction_data TO DataAnalysts;

2. Row Level Security:
   - CREATE OR REPLACE VIEW finance_data_catalog.transaction_data.secure_transactions AS
     SELECT * FROM finance_data_catalog.transaction_data.transactions
     WHERE (TransactionAmount <= 10000)
     OR (current_user() IN ('authorized_user1', 'authorized_user2'));
   - GRANT SELECT ON VIEW finance_data_catalog.transaction_data.secure_transactions TO DataAnalysts;

Task 5: Data Governance Best Practices

1. Create Data Quality Rules:
   - Add a CHECK constraint for non-negative transaction amounts
     ALTER TABLE finance_data_catalog.transaction_data.transactions
     ADD CONSTRAINT chk_non_negative_amount
     CHECK (TransactionAmount >= 0);
   - Add a CHECK constraint for valid email format (basic regex pattern)
     ALTER TABLE finance_data_catalog.transaction_data.customers
     ADD CONSTRAINT chk_valid_email_format
     CHECK (Email LIKE '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$');

2. Validate Data Governance:
    - Navigate to lineage and audit logs to check if the operations are performed correctly
    - Query to check for any transactions with negative amounts (should return 0 rows)
    - SELECT * FROM finance_data_catalog.transaction_data.transactions
      WHERE TransactionAmount < 0;

Task 6: Data Lifecycle Management:

1. Implement Time Travel:
    - SELECT * FROM finance_data_catalog.transaction_data.transactions
      VERSION AS OF 2;
2. Implement Vacuum:
    - VACUUM finance_data_catalog.transaction_data.transactions RETAIN 168 HOURS;

Assignment 2:

Task 1:

1. Create a new Catalog:
    - CREATE CATALOG corporate_data_catalog;

2. Create Schema 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:
    - CREATE TABLE corporate_data_catalog.sales_data.sales (
      SalesID INT,
      CustomerID INT,
      SalesAmount DOUBLE,
      SalesDate DATE
      );
    - CREATE TABLE corporate_data_catalog.hr_data.employees (
      EmployeeID INT,
      EmployeeName STRING,
      Department STRING,
      Salary DOUBLE
      );
    - CREATE TABLE corporate_data_catalog.finance_data.invoices (
      InvoiceAmount DOUBLE,
      PaymentDate DATE
      );

Task 2: Enable Data Discovery for Cross-Departmental Data

1. Search for tables across departments:
    - Using the unity catalog interface to search for tables across the schemas.

2. Tag Sensitive Informations:
    - ALTER TABLE corporate_data_catalog.hr_data.employees
      ALTER COLUMN Salary SET TAG 'sensitive' = 'true';
    - ALTER TABLE corporate_data_catalog.finance_data.invoices
      ALTER COLUMN InvoiceAmount SET TAG 'sensitive' = 'true';

3. Data Profiling:
    - Calculate the total sales amount:
    - SELECT SUM(SalesAmount) AS total_sales_amount
      FROM corporate_data_catalog.sales_data.sales;
    - Calculate the average salary for each department:
    - SELECT Department, AVG(Salary) AS avg_salary
      FROM corporate_data_catalog.hr_data.employees
      GROUP BY Department
      ORDER BY avg_salary DESC;

Task 3: Implement Data Lineage and Data Auditing:

1. Track Data Lineage:
    - Using the Data bricks lineage feature, we visualize how the data flows between tables

2. Enable Audit logs:
    - Navigate to the admin console to enable the Audit log to view the operations being performed.

Task 4: Data Access Control and Security:

1. Set up User Roles:
    - CREATE GROUP SalesTeam;
    - GRANT SELECT ON ALL TABLES IN SCHEMA corporate_data_catalog.sales_data TO SalesTeam;
    - CREATE GROUP FinanceTeam;
    - GRANT SELECT ON ALL TABLES IN SCHEMA corporate_data_catalog.sales_data TO FinanceTeam;
    - GRANT SELECT ON ALL TABLES IN SCHEMA corporate_data_catalog.finance_data TO FinanceTeam;
    - CREATE GROUP HRTeam;
    - GRANT SELECT, UPDATE ON ALL TABLES IN SCHEMA corporate_data_catalog.hr_data TO HRTeam;

2. Column Level Security:
    - CREATE GROUP HRManagers;
    - GRANT SELECT(Salary) ON TABLE corporate_data_catalog.hr_data.employees TO HRManagers;

3. Row Level Security:
    - CREATE SECURITY POLICY sales_rep_policy
      ON corporate_data_catalog.sales_data.sales
      AS (CustomerID = current_user());
    - GRANT SELECT ON TABLE corporate_data_catalog.sales_data.sales TO SalesTeam WITH POLICY sales_rep_policy;

Task 5: Data Governance and Quality Enforcement:

1. Set Data Quality Rules:
    - Ensure sales amounts are positive
    - ALTER TABLE corporate_data_catalog.sales_data.sales
      ADD CONSTRAINT check_positive_sales_amount
      CHECK (SalesAmount > 0);
    - Ensure employee salaries are greater than zero
    - ALTER TABLE corporate_data_catalog.hr_data.employees
      ADD CONSTRAINT check_positive_salary
      CHECK (Salary > 0);

2. Applying time travel:
    - SELECT * FROM corporate_data_catalog.finance_data.invoices VERSION AS OF 1;

Task 6: Optimize and Clean up:

1. Optimize:
    - OPTIMIZE corporate_data_catalog.sales_data.sales;
    - OPTIMIZE corporate_data_catalog.finance_data.invoices;
2. Vacuum:
    - VACUUM corporate_data_catalog.sales_data.sales RETAIN 168 HOURS;
    - VACUUM corporate_data_catalog.finance_data.invoices;

Assignment 3:

Task 1: SetUp Unity Catalog:

1. Create a new Catalog:
    - CREATE CATALOG enterprise_data_catalog;

2. Create 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:
    - CREATE TABLE enterprise_data_catalog.marketing_data.campaigns (
      CampaignID INT,
      CampaignName STRING,
      Budget DOUBLE,
      StartDate DATE
      );
    - CREATE TABLE enterprise_data_catalog.operations_data.orders (
      OrderID INT,
      ProductID INT,
      Quantity INT,
      ShippingStatus STRING
      );
    - CREATE TABLE enterprise_data_catalog.it_data.incidents (
      IncidentID INT,
      ReportedBy STRING,
      IssueType STRING,
      ResolutionTime DOUBLE
      );

Task 2 : Data Discovery and Classification:

1. List all tables in the catalog:
    - SHOW TABLES IN enterprise_data_catalog;
    - Use the data discovery feature to list all tables

2. Tag Sensitive information:
    - ALTER TABLE enterprise_data_catalog.marketing_data.campaigns
      ALTER COLUMN Budget SET TAG 'sensitive' = 'true';
    - ALTER TABLE enterprise_data_catalog.it_data.incidents
      ALTER COLUMN ResolutionTime SET TAG 'sensitive' = 'true';

3. Data Profiling:
    - SELECT MIN(Budget) AS MinBudget, MAX(Budget) AS MaxBudget, AVG(Budget) AS AvgBudget, COUNT(*) AS CampaignCount
      FROM enterprise_data_catalog.marketing_data.campaigns;
    - SELECT ShippingStatus, COUNT(*) AS StatusCount
      FROM enterprise_data_catalog.operations_data.orders
      GROUP BY ShippingStatus;

Task 3: Data Lineage and Auditing:

1. Track Data Lineage:
    - Using Unity Catalog we can track the lineage of this data flow, we can visualize the relationships between datasets, tables
2. Enable and analyze audit logs:
    - Navigate to Admin console, under the security section enable Audit logging
    - We can track who accessed or modified the data in the incidents table

Task 4: Implement Fine-Grained Access Control:

1. Creating Groups:
    - CREATE GROUP MarketingTeam;
    - CREATE GROUP OperationsTeam;
    - CREATE GROUP ITSupportTeam;

2. Assigning roles:
    - Access to MarketingTeam for the marketing_data schema:
    - GRANT USAGE ON SCHEMA enterprise_data_catalog.marketing_data TO MarketingTeam;
    - GRANT SELECT ON ALL TABLES IN SCHEMA enterprise_data_catalog.marketing_data TO MarketingTeam;

    - access to OperationsTeam for the operations_data and marketing_data schemas:
    - GRANT USAGE ON SCHEMA enterprise_data_catalog.operations_data TO OperationsTeam;
    - GRANT USAGE ON SCHEMA enterprise_data_catalog.marketing_data TO OperationsTeam;
    - GRANT SELECT ON ALL TABLES IN SCHEMA enterprise_data_catalog.operations_data TO OperationsTeam;
    - GRANT SELECT ON ALL TABLES IN SCHEMA enterprise_data_catalog.marketing_data TO OperationsTeam;

    - -ITSupportTeam access to it_data schema and permission to update resolution times:
    - GRANT USAGE ON SCHEMA enterprise_data_catalog.it_data TO ITSupportTeam;
    - GRANT SELECT, UPDATE ON TABLE enterprise_data_catalog.it_data.incidents TO ITSupportTeam;

3. Column level security:
    - Grant MarketingTeam access to view the Budget column in the marketing_data schema
    - GRANT SELECT(Budget) ON TABLE enterprise_data_catalog.marketing_data.campaigns TO MarketingTeam;

    - Revoke access to the Budget column from other groups
    - REVOKE SELECT(Budget) ON TABLE enterprise_data_catalog.marketing_data.campaigns FROM OperationsTeam;

Task 5: Data Governance and Quality enforcement:

1. Set Data Quality Rules:
    - Positive Budget value:
    - ALTER TABLE enterprise_data_catalog.marketing_data.campaigns
      ADD CONSTRAINT budget_check CHECK (Budget > 0);
    - Check to Ensure Valid Shipping Status:
    - ALTER TABLE enterprise_data_catalog.operations_data.orders

ADD CONSTRAINT shipping_status_check CHECK (ShippingStatus IN ('Pending', 'Shipped', 'Delivered'));

- Constraint to enforce non-negative resolution times
- ALTER TABLE enterprise_data_catalog.it_data.incidents
  ADD CONSTRAINT resolution_time_check CHECK (ResolutionTime >= 0);

2. Time Travel:
   - SELECT * FROM enterprise_data_catalog.operations_data.orders
     VERSION AS OF 1;

Task 6: Optimize and Vacuum:

1. Optimize Tables:
   - OPTIMIZE enterprise_data_catalog.operations_data.orders;
   - OPTIMIZE enterprise_data_catalog.it_data.incidents;
2. Vacuum Tables:
   - VACUUM enterprise_data_catalog.operations_data.orders RETAIN 168 HOURS;
   - VACUUM enterprise_data_catalog.it_data.incidents RETAIN 168 HOURS;