**Assignment 1:**

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

1. **Create a Catalog:**

* CREATE CATALOG finance\_data\_catalog;

1. **Create Multiple Schemas:**

* CREATE SCHEMA finance\_data\_catalog.transaction\_data;
* CREATE SCHEMA finance\_data\_catalog.customer\_data;

1. **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;

1. **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;

1. **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

1. 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;

1. 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,}$](mailto:+@[A-Za-z0-9.-]+\\.%5bA-Za-z%5d%7b2,%7d$)');

1. 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;

1. 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;

1. 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;

1. 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.

1. 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';

1. 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

1. 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;

1. Column Level Security:

* CREATE GROUP HRManagers;
* GRANT SELECT(Salary) ON TABLE corporate\_data\_catalog.hr\_data.employees TO HRManagers;

1. 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);

1. 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;

1. 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;

1. Create Schemas:

* CREATE SCHEMA enterprise\_data\_catalog.marketing\_data;
* CREATE SCHEMA enterprise\_data\_catalog.operations\_data;
* CREATE SCHEMA enterprise\_data\_catalog.it\_data;

1. 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

1. 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';

1. 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

1. 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;

1. 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;

1. 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);

1. 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;

1. Vacuum Tables:

* VACUUM enterprise\_data\_catalog.operations\_data.orders RETAIN 168 HOURS;
* VACUUM enterprise\_data\_catalog.it\_data.incidents RETAIN 168 HOURS;