#original data

CREATE TABLE OriginalData (

    TransactionID INT,

    CustomerName VARCHAR(255),

    Email VARCHAR(255),

    TransactionAmount DECIMAL(10, 2),

    TransactionDate DATE

);

INSERT INTO OriginalData VALUES (1, 'Alice', 'alice@example.com', 100.0, '2024-01-01');

INSERT INTO OriginalData VALUES (2, ' Bob ', 'bob@example', NULL, '2024-02-15');

INSERT INTO OriginalData VALUES (3, 'Charlie', 'charlie@gmail.com', 150.5, NULL);

INSERT INTO OriginalData VALUES (4, NULL, 'eve@gmail.com', 200.0, '2024-03-10');

INSERT INTO OriginalData VALUES (5, 'Eve', NULL, 100.0, '2024-03-10');

# cleaned data

CREATE TABLE cleaneddata (

    TransactionID INT,

    CustomerName VARCHAR(255),

    Email VARCHAR(255),

    TransactionAmount DECIMAL(10, 2),

    TransactionDate DATE

);

INSERT INTO cleaneddata VALUES (1, 'Alice', 'alice@example.com', 100.0, '2024-01-01');

INSERT INTO cleaneddata VALUES (2, ' Bob ', 'bob@example', 0, '2024-02-15');

INSERT INTO cleaneddata VALUES (3, 'Charlie', 'charlie@gmail.com', 150, '2024-01-01');

INSERT INTO cleaneddata VALUES (4, 'Ali', 'eve@gmail.com', 200.0, '2024-03-10');

INSERT INTO cleaneddata VALUES (5, 'Eve', 'unknown', 100.0, '2024-03-10');

# customer\_profiles data

CREATE TABLE customer\_profiles (

    CustomerID INT PRIMARY KEY,

    CustomerName VARCHAR(255) NOT NULL

);

INSERT INTO customer\_profiles (CustomerID, CustomerName)

VALUES

    (1, 'John Doe'),

    (2, 'Jane Smith'),

    (3, 'Michael Brown'),

    (4, 'Emily Davis'),

    (5, 'William Johnson');

#transactions data

CREATE TABLE transactions (

    TransactionID INT PRIMARY KEY,

    CustomerName VARCHAR(255),

    TransactionDate DATE,

    TransactionAmount DECIMAL(10, 2),

    Email VARCHAR(255)

);

INSERT INTO transactions (TransactionID, CustomerName, TransactionDate, TransactionAmount, Email)

VALUES

    (101, 'John Doe', '2023-12-01', 150.00, 'john.doe@example.com'),

    (102, 'Jane Smith', '2023-12-01', -50.00, 'jane.smith@example.com'),

    (103, 'Michael Brown', '2023-12-02', 200.00, 'michael.brown@example'),

    (104, 'Emily Davis', '2023-12-02', NULL, 'emily.davis@example.com'),

    (105, 'Unknown Customer', '2023-12-03', 120.00, 'unknown@example.com');

# Data Profiling:

#Generate summary statistics for the TransactionAmount column to identify anomalies or outliers.

SELECT

    COUNT(TransactionAmount) AS TotalTransactions,

    MIN(TransactionAmount) AS MinAmount,

    MAX(TransactionAmount) AS MaxAmount,

    AVG(TransactionAmount) AS AvgAmount,

    SUM(TransactionAmount) AS TotalAmount,

    STDDEV(TransactionAmount) AS StdDevAmount;

  #  2. Comparison with Original Data:

#Compare the number of rows in the original dataset with the cleaned dataset.

SELECT

    (SELECT COUNT(\*) FROM OriginalData) AS OriginalRowCount,

    (SELECT COUNT(\*) FROM cleaneddata) AS CleanedRowCount;

#2. Comparison with Original Data:

#Compare the number of rows in the original dataset with the cleaned dataset.

    SELECT

    (SELECT COUNT(\*) FROM OriginalData) AS OriginalRowCount,

    (SELECT COUNT(\*) FROM cleaneddata) AS CleanedRowCount;

#3. Validation Rules:

#. Verify email addresses:

#Check that all email addresses in the cleaned dataset contain an "@" symbol and have no invalid entries.

SELECT \*

FROM cleaneddata

WHERE Email NOT LIKE '%@%' OR Email IS NULL;

# Check TransactionAmounts:

#Ensure all TransactionAmounts are non-negative and missing values have been replaced.

SELECT \*

FROM cleaneddata

WHERE TransactionAmount < 0 OR TransactionAmount IS NULL;

# Consistency Checks:

#Validate that transactions occurring on the same TransactionDate do not have duplicate TransactionIDs.

SELECT TransactionDate, COUNT(TransactionID) AS DuplicateCount

FROM cleaneddata

GROUP BY TransactionDate

HAVING COUNT(TransactionID) > 1;

# Cross-Table Validation (Simulated):

# Identify transactions with CustomerName values not in customer\_profiles.

SELECT cd.\*

FROM cleaneddata cd

LEFT JOIN customer\_profiles cp ON cd.CustomerName = cp.CustomerName

WHERE cp.CustomerID IS NULL;

#This query finds rows in the cleaneddata table where the CustomerName does not exist in the customer\_profiles table.

# Join the two tables to verify valid customer references.

sql

Copy

SELECT cd.\*, cp.CustomerID

FROM cleaneddata cd

INNER JOIN customer\_profiles cp ON cd.CustomerName = cp.CustomerName;

#null check values

SELECT \*

FROM cleaneddata

WHERE TransactionID IS NULL

   OR CustomerName IS NULL

   OR Email IS NULL

   OR TransactionAmount IS NULL

   OR TransactionDate IS NULL;

#Impact of Cleaning:

#a. Compare the count of missing values in the original and cleaned datasets.

   -- Original Data

SELECT

    SUM(CASE WHEN CustomerName IS NULL THEN 1 ELSE 0 END) AS OriginalNullCustomerName,

    SUM(CASE WHEN Email IS NULL THEN 1 ELSE 0 END) AS OriginalNullEmail,

    SUM(CASE WHEN TransactionAmount IS NULL THEN 1 ELSE 0 END) AS OriginalNullTransactionAmount,

    SUM(CASE WHEN TransactionDate IS NULL THEN 1 ELSE 0 END) AS OriginalNullTransactionDate

FROM OriginalData;

-- Cleaned Data

SELECT

    SUM(CASE WHEN CustomerName IS NULL THEN 1 ELSE 0 END) AS CleanedNullCustomerName,

    SUM(CASE WHEN Email IS NULL THEN 1 ELSE 0 END) AS CleanedNullEmail,

    SUM(CASE WHEN TransactionAmount IS NULL THEN 1 ELSE 0 END) AS CleanedNullTransactionAmount,

    SUM(CASE WHEN TransactionDate IS NULL THEN 1 ELSE 0 END) AS CleanedNullTransactionDate

FROM cleaneddata;

#b. Compare the number of duplicate entries in the original and cleaned datasets.

-- Original Data

SELECT COUNT(\*) AS OriginalDuplicateCount

FROM (

    SELECT TransactionID, COUNT(\*)

    FROM OriginalData

    GROUP BY TransactionID

    HAVING COUNT(\*) > 1

) AS OriginalDuplicates;

-- Cleaned Data

SELECT COUNT(\*) AS CleanedDuplicateCount

FROM (

    SELECT TransactionID, COUNT(\*)

    FROM cleaneddata

    GROUP BY TransactionID

    HAVING COUNT(\*) > 1

) AS CleanedDuplicates;