```
-- ###############################################################
-- Project : Customer Churn Analysis for an E-Commerce Platform
-- Objective: Identify churn patterns, key drivers, and business
-- opportunities to improve retention.
-- ###############################################################

-- ###############################################################
-- Business Questions:
-- 1. What is the churn rate across different customer segments?
-- 2. Which customer behaviors indicate higher churn risk?
-- 3. Does payment mode, device usage, or satisfaction score influence churn?
-- 4. Which high-value customers are at risk of churning?
-- 5. What recommendations can be made to reduce churn?
-- ###############################################################

-- ###############################################################
-- 1. Database Setup and Initial Data Exploration
-- ###############################################################

-- Create Database
CREATE DATABASE IF NOT EXISTS Ecommerce_db;
USE Ecommerce_db;

-- View raw customer data
SELECT * FROM customers;

-- Create a duplicate table for data cleaning
CREATE TABLE customers_data LIKE customers;

INSERT INTO customers_data
SELECT * FROM customers;

-- View data in the duplicate table
SELECT * FROM customers_data;

-- #####################
-- 2. Data Cleaning
-- #####################

-- 2.1 Identify Duplicate Records
SELECT CustomerID, COUNT(*) AS duplicate_count
FROM customers_data
GROUP BY CustomerID
HAVING COUNT(*) > 1;

-- Alternative approach using ROW_NUMBER
WITH duplicate_cte AS (
    SELECT *,
        ROW_NUMBER() OVER (
            PARTITION BY CustomerID, Churn, Tenure, PreferredLoginDevice, CityTier,
            WarehouseToHome, PreferredPaymentMode, Gender, HourSpendOnApp, PreferedOrderCat,
            SatisfactionScore, MaritalStatus, NumberOfAddress, Complain, OrderAmountHikeFromlastYear
            CouponUsed, OrderCount, DaySinceLastOrder, CashbackAmount
        ) AS row_num
    FROM customers_data
)
SELECT *
FROM duplicate_cte
WHERE row_num > 1;

-- 2.2 Create clean table with row number
CREATE TABLE customers_data2 (
    CustomerID INT DEFAULT NULL,
    Churn INT DEFAULT NULL,
    Tenure TEXT,
    PreferredLoginDevice TEXT,
    CityTier INT DEFAULT NULL,
    WarehouseToHome INT DEFAULT NULL,
    PreferredPaymentMode TEXT,
    Gender TEXT,
    HourSpendOnApp TEXT,
    NumberOfDeviceRegistered INT DEFAULT NULL,
    PreferedOrderCat TEXT,
    SatisfactionScore INT DEFAULT NULL,
    MaritalStatus TEXT,
    NumberOfAddress INT DEFAULT NULL,
    Complain INT DEFAULT NULL,
    OrderAmountHikeFromlastYear INT DEFAULT NULL,
    CouponUsed INT DEFAULT NULL,
    OrderCount INT DEFAULT NULL,
    DaySinceLastOrder INT DEFAULT NULL,
    CashbackAmount INT DEFAULT NULL,
    row_num INT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- Insert values into customers_data2 with row number
INSERT INTO customers_data2
SELECT *,
    ROW_NUMBER() OVER (
        PARTITION BY CustomerID, Churn, Tenure, PreferredLoginDevice, CityTier,
        WarehouseToHome, PreferredPaymentMode, Gender, HourSpendOnApp, PreferedOrderCat,
        SatisfactionScore, MaritalStatus, NumberOfAddress, Complain, OrderAmountHikeFromlastYear,
        CouponUsed, OrderCount, DaySinceLastOrder, CashbackAmount
    ) AS row_num
FROM customers_data;

-- View duplicate records
SELECT * FROM customers_data2
WHERE row_num > 1;

-- Disable safe updates to delete duplicates
SET SQL_SAFE_UPDATES = 0;

-- Remove duplicates
DELETE FROM customers_data2
WHERE row_num > 1;

-- View cleaned data
SELECT * FROM customers_data2;

-- #############################
-- 2.3 Standardize Columns
-- #############################

-- Check unique payment modes
SELECT DISTINCT PreferredPaymentMode FROM customers_data2;

-- Standardize payment modes
UPDATE customers_data2
SET PreferredPaymentMode = 'Cash on Delivery'
WHERE PreferredPaymentMode = 'COD';

UPDATE customers_data2
SET PreferredPaymentMode = 'Credit Card'
WHERE PreferredPaymentMode = 'CC';

-- Standardize order categories
UPDATE customers_data2
SET PreferedOrderCat = 'Mobile Phone'
WHERE PreferedOrderCat LIKE 'Mobile%';

-- ################################
-- 2.4 Handle Missing Values
-- ################################

-- Check missing/null values
SELECT
    SUM(CASE WHEN CustomerID IS NULL OR CustomerID = '' THEN 1 ELSE 0 END) AS missing_CustomerID,
    SUM(CASE WHEN Churn IS NULL OR Churn = '' THEN 1 ELSE 0 END) AS missing_Churn,
    SUM(CASE WHEN Tenure IS NULL OR Tenure = '' THEN 1 ELSE 0 END) AS missing_Tenure,
    SUM(CASE WHEN PreferredLoginDevice IS NULL OR PreferredLoginDevice = '' THEN 1 ELSE 0 END) AS mi
    SUM(CASE WHEN CityTier IS NULL OR CityTier = '' THEN 1 ELSE 0 END) AS missing_CityTier,
    SUM(CASE WHEN WarehouseToHome IS NULL OR WarehouseToHome = '' THEN 1 ELSE 0 END) AS missing_Ware
    SUM(CASE WHEN PreferredPaymentMode IS NULL OR PreferredPaymentMode = '' THEN 1 ELSE 0 END) AS mi
    SUM(CASE WHEN Gender IS NULL OR Gender = '' THEN 1 ELSE 0 END) AS missing_Gender,
    SUM(CASE WHEN HourSpendOnApp IS NULL OR HourSpendOnApp = '' THEN 1 ELSE 0 END) AS missing_HourSp
    SUM(CASE WHEN NumberOfDeviceRegistered IS NULL OR NumberOfDeviceRegistered = '' THEN 1 ELSE 0 EN
    SUM(CASE WHEN PreferedOrderCat IS NULL OR PreferedOrderCat = '' THEN 1 ELSE 0 END) AS missing_Pr
    SUM(CASE WHEN SatisfactionScore IS NULL OR SatisfactionScore = '' THEN 1 ELSE 0 END) AS missing_
    SUM(CASE WHEN MaritalStatus IS NULL OR MaritalStatus = '' THEN 1 ELSE 0 END) AS missing_MaritalS
    SUM(CASE WHEN NumberOfAddress IS NULL OR NumberOfAddress = '' THEN 1 ELSE 0 END) AS missing_Numb
    SUM(CASE WHEN Complain IS NULL OR Complain = '' THEN 1 ELSE 0 END) AS missing_Complain,
    SUM(CASE WHEN OrderAmountHikeFromlastYear IS NULL OR OrderAmountHikeFromlastYear = '' THEN 1 ELS
    SUM(CASE WHEN CouponUsed IS NULL OR CouponUsed = '' THEN 1 ELSE 0 END) AS missing_CouponUsed,
    SUM(CASE WHEN OrderCount IS NULL OR OrderCount = '' THEN 1 ELSE 0 END) AS missing_OrderCount,
    SUM(CASE WHEN DaySinceLastOrder IS NULL OR DaySinceLastOrder = '' THEN 1 ELSE 0 END) AS missing_
    SUM(CASE WHEN CashbackAmount IS NULL OR CashbackAmount = '' THEN 1 ELSE 0 END) AS missing_Cashba
    SUM(CASE WHEN row_num IS NULL OR row_num = '' THEN 1 ELSE 0 END) AS missing_row_num
FROM customers_data2;

-- Update missing Tenure with average
UPDATE customers_data2
SET Tenure = (
    SELECT avg_tenure FROM (
        SELECT ROUND(AVG(Tenure), 0) AS avg_tenure
        FROM customers_data2
        WHERE Tenure IS NOT NULL AND Tenure <> ''
    ) AS t
)
WHERE Tenure IS NULL OR Tenure = '';

-- Update missing HourSpendOnApp with average
UPDATE customers_data2
SET HourSpendOnApp = (
    SELECT avg_HourSpendOnApp FROM (
        SELECT ROUND(AVG(HourSpendOnApp), 0) AS avg_HourSpendOnApp
        FROM customers_data2
        WHERE HourSpendOnApp IS NOT NULL AND HourSpendOnApp <> ''
    ) AS t
)
WHERE HourSpendOnApp IS NULL OR HourSpendOnApp = '';

-- Verify updates
SELECT * FROM customers_data2;

-- ######################################
-- 3. Exploratory Data Analysis (EDA)
-- ######################################

-- Total Customers, Churned Customers, Churn Rate
SELECT
    COUNT(*) AS Total_Customers,
    SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers,
    ROUND(100.0 * SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS Churn_Rate_Percent
FROM customers_data2;


-- Churn by City Tier (Demographics)
WITH City_Churn AS (
    SELECT
        CityTier,
        COUNT(*) AS Total_Customers,
        SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers
    FROM customers_data2
    GROUP BY CityTier
)
SELECT
    CityTier,
    Total_Customers,
    Churned_Customers,
    ROUND(100.0 * Churned_Customers / Total_Customers, 2) AS Churn_Rate_Percent
FROM City_Churn
ORDER BY Churn_Rate_Percent DESC;


-- Churn by Preferred Payment Mode
WITH Payment_Churn AS (
    SELECT
        PreferredPaymentMode,
        COUNT(*) AS Total_Customers,
        SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers
    FROM customers_data2
    GROUP BY PreferredPaymentMode
)
SELECT
    PreferredPaymentMode,
    Total_Customers,
    Churned_Customers,
    ROUND(100.0 * Churned_Customers / Total_Customers, 2) AS Churn_Rate_Percent
FROM Payment_Churn
ORDER BY Churn_Rate_Percent DESC;
```

```
-- Churn rate by Satisfaction Score
SELECT
    SatisfactionScore,
    COUNT(*) AS Total_Customers,
    SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers,
    ROUND(100.0 * SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS Churn_Rate_Percent
FROM customers_data2
GROUP BY SatisfactionScore
ORDER BY SatisfactionScore desc;


-- Churn rate by Number of Complaints
SELECT
    Complain,
    COUNT(*) AS Total_Customers,
    SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers,
    ROUND(100.0 * SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS Churn_Rate_Percent
FROM customers_data2
GROUP BY Complain
ORDER BY Churn_Rate_Percent DESC;


-- Churn by App usage
SELECT
    CASE
        WHEN HourSpendOnApp < 2 THEN 'Low Usage (<2 hrs)'
        WHEN HourSpendOnApp BETWEEN 2 AND 4 THEN 'Medium Usage (2-4 hrs)'
        ELSE 'High Usage (>4 hrs)'
    END AS Usage_Group,
    COUNT(*) AS Total_Customers,
    SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers,
    ROUND(100.0 * SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS Churn_Rate_Percent
FROM customers_data2
GROUP BY Usage_Group
ORDER BY Churn_Rate_Percent DESC;


-- Device usage vs churn
SELECT
    PreferredLoginDevice,
    COUNT(*) AS Total_Customers,
    SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers,
    ROUND(100.0 * SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS Churn_Rate_Percent
FROM customers_data2
GROUP BY PreferredLoginDevice
ORDER BY Churn_Rate_Percent DESC;


-- Churn by Order Category Preference
SELECT
    PreferedOrderCat,
    ROUND(100.0 * SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS Churn_Rate_Percent
FROM customers_data2
GROUP BY PreferedOrderCat
ORDER BY Churn_Rate_Percent DESC;


-- Cashback Bucket Analysis
WITH CashbackBuckets AS (
    SELECT
        CASE
            WHEN CashbackAmount = 0 THEN 'No Cashback'
            WHEN CashbackAmount BETWEEN 1 AND 100 THEN 'Low Cashback'
            WHEN CashbackAmount BETWEEN 101 AND 500 THEN 'Medium Cashback'
            ELSE 'High Cashback'
        END AS Cashback_Group,
        COUNT(*) AS Total_Customers,
        SUM(CASE WHEN Churn = 1 THEN 1 ELSE 0 END) AS Churned_Customers
    FROM customers_data2
    GROUP BY Cashback_Group
)
SELECT
    Cashback_Group,
    Total_Customers,
    Churned_Customers,
    ROUND(100.0 * Churned_Customers / Total_Customers, 2) AS Churn_Rate_Percent
FROM CashbackBuckets
```

```
ORDER BY Churn_Rate_Percent DESC;


-- Which retained customers are at risk due to inactivity?
SELECT
    CustomerID,
    DaySinceLastOrder,
    SatisfactionScore,
    RANK() OVER(ORDER BY DaySinceLastOrder DESC) AS Risk_Rank
FROM customers_data2
WHERE Churn = 0 AND DaySinceLastOrder > 30;
```