E-Commerce Customer Churn Analysis Project

By

S.Shenbagalakshmi

Problem Statement: In the realm of e-commerce, businesses face the challenge of understanding customer churn patterns to ensure customer satisfaction and sustained profitability. This project aims to delve into the dynamics of customer churn within an e-commerce domain, utilizing historical transactional data to uncover underlying patterns and drivers of churn. By analyzing customer attributes such as tenure, preferred payment modes, satisfaction scores, and purchase behavior, the project seeks to investigate and understand the dynamics of customer attrition and their propensity to churn. The ultimate objective is to equip e-commerce enterprises with actionable insights to implement targeted retention strategies and mitigate churn, thereby fostering long-term customer relationships and ensuring business viability in a competitive landscape.

Dataset

```
    OrderCount                  INT,
    DaySinceLastOrder           INT,
    CashbackAmount              INT
    );

INSERT INTO customer_churn(CustomerID,Churn,Tenure,PreferredLoginDevice,CityTier,WarehouseToHome,PreferredPaymentMode,
    (50001,1,4,'Mobile Phone',3,6,'Debit Card','Female',3,3,'Laptop & Accessory',2,'Single',9,1,11,1,1,5,160),
    (50002,1,NULL,'Phone',1,8,'UPI','Male',3,4,'Mobile',3,'Single',7,1,15,0,1,0,121),
    (50003,1,NULL,'Phone',1,30,'Debit Card','Male',2,4,'Mobile',3,'Single',6,1,14,0,1,3,120),
    (50004,1,0,'Phone',3,15,'Debit Card','Male',2,4,'Laptop & Accessory',5,'Single',8,0,23,0,1,3,134),
    (50005,1,0,'Phone',1,12,'CC','Male',NULL,3,'Mobile',5,'Single',3,0,11,1,1,3,130),
    (50006,1,0,'Computer',1,22,'Debit Card','Female',3,5,'Mobile Phone',5,'Single',2,1,22,4,6,7,139),
    (50007,1,NULL,'Phone',3,11,'Cash on Delivery','Male',2,3,'Laptop & Accessory',2,'Divorced',4,0,14,0,1,0,121),
    (50008,1,NULL,'Phone',1,6,'CC','Male',3,3,'Mobile',2,'Divorced',3,1,16,2,2,0,123),
    (50009,1,13,'Phone',3,9,'E wallet','Male',NULL,4,'Mobile',3,'Divorced',2,1,14,0,1,2,127),
    (50010,1,NULL,'Phone',1,31,'Debit Card','Male',2,5,'Mobile',3,'Single',2,0,12,1,1,1,123),
(55616,0,14,'Mobile Phone',1,9,'Debit Card','Male',1,5,'Mobile Phone',3,'Single',4,0,15,1,3,3,153),
(55617,0,9,'Computer',1,25,'Debit Card','Male',4,4,'Laptop & Accessory',3,'Married',3,1,15,1,2,3,182),
(55618,0,14,'Phone',1,9,'Credit Card','Female',4,4,'Mobile Phone',3,'Married',4,0,15,1,2,3,145),
(55619,0,9,'Mobile Phone',1,8,'Debit Card','Female',4,6,'Mobile Phone',1,'Married',3,0,13,2,2,2,155),
(55620,0,3,'Mobile Phone',1,20,'UPI','Male',3,5,'Laptop & Accessory',4,'Married',6,1,14,1,2,4,165),
(55621,0,3,'Mobile Phone',1,35,'Credit Card','Female',4,5,'Mobile Phone',5,'Single',3,0,15,1,2,5,163),
(55622,1,14,'Mobile Phone',3,35,'E wallet','Male',3,5,'Fashion',5,'Married',6,1,14,3,NULL,1,234),
(55623,0,13,'Mobile Phone',3,31,'E wallet','Female',3,5,'Grocery',1,'Married',2,0,12,4,NULL,7,245),
(55624,0,5,'Computer',1,12,'Credit Card','Male',4,4,'Laptop & Accessory',5,'Single',2,0,20,2,2,NULL,224),
(55625,0,1,'Mobile Phone',3,12,'UPI','Female',2,5,'Mobile Phone',3,'Single',2,0,19,2,2,1,155),
(55626,0,10,'Computer',1,30,'Credit Card','Male',3,2,'Laptop & Accessory',1,'Married',6,0,18,1,2,4,151),
(55627,0,13,'Mobile Phone',1,13,'Credit Card','Male',3,5,'Fashion',5,'Married',6,0,16,1,2,NULL,225),
(55628,0,1,'Mobile Phone',1,11,'Debit Card','Male',3,2,'Laptop & Accessory',4,'Married',3,1,21,1,2,4,186),
(55629,0,23,'Computer',3,9,'Credit Card','Male',4,5,'Laptop & Accessory',4,'Married',4,0,15,2,2,9,179),
(55630,0,8,'Mobile Phone',1,15,'Credit Card','Male',3,2,'Laptop & Accessory',3,'Married',4,0,13,2,2,3,169);
```

We insert the values to the created table named customer_churn

Project Steps and Objectives:

Data Cleaning:

Handling Missing Values and Outliers:

➢ Impute mean for the following columns, and round off to the nearest integer if required:
WarehouseToHome,HourSpendOnApp,OrderAmountHikeFromlastYear,

DaySinceLastOrder.

Query :

```
-- WarehouseToHome

 SET SQL_SAFE_UPDATES=0;

SET @avg_warehousetohome=(SELECT ROUND(AVG(WarehouseToHome))AS
avg_warehousetohome FROM customer_churn);


UPDATE customer_churn

SET WarehouseToHome = @avg_warehousetohome

WHERE WarehouseToHome IS NULL ;


-- HourSpendOnApp


SET SQL_SAFE_UPDATES=0;


SET @avg_HourSpendOnApp=(SELECT ROUND(AVG(HourSpendOnApp))AS
avg_hourSpendOnApp FROM customer_churn);


UPDATE customer_churn

SET HourSpendOnApp = @avg_HourSpendOnApp

WHERE  HourSpendOnApp IS NULL ;
```

```
-- OrderAmountHikeFromlastYear

SET SQL_SAFE_UPDATES=0;

SET @avg_OrderAmountHikeFromlastYear=(SELECT
ROUND(AVG(OrderAmountHikeFromlastYear))
                                AS avg_OrderAmountHikeFromlastYear FROM
customer_churn);
UPDATE customer_churn
SET OrderAmountHikeFromlastYear = @avg_OrderAmountHikeFromlastYear
WHERE  OrderAmountHikeFromlastYear IS NULL ;

-- DaySinceLastOrder
SET SQL_SAFE_UPDATES=0;

SET @avg_DaySinceLastOrder=(SELECT
ROUND(AVG(DaySinceLastOrder))AS avg_DaySinceLastOrder FROM
customer_churn);

UPDATE customer_churn
SET DaySinceLastOrder = @avg_DaySinceLastOrder
WHERE DaySinceLastOrder IS NULL ;
```

We use these query to change the null values into the average value of the columns

➢ **Impute mode for the following columns: Tenure, CouponUsed, OrderCount.**

Query :

```
SELECT Tenure , COUNT(*) FROM customer_churn GROUP BY Tenure ORDER
BY COUNT(*) DESC LIMIT 1;
```

```
SELECT CouponUsed , COUNT(*) FROM customer_churn GROUP BY
CouponUsed ORDER BY COUNT(*) DESC LIMIT 1;

SELECT OrderCount , COUNT(*) FROM customer_churn GROUP BY
OrderCount ORDER BY COUNT(*) DESC LIMIT 1;


-- Tenure

SET SQL_SAFE_UPDATES=0;


SET @Tenure_mode = (SELECT Tenure FROM customer_churn GROUP BY
Tenure ORDER BY COUNT(*) DESC LIMIT 1);


UPDATE customer_churn

SET Tenure = @Tenure_mode

WHERE Tenure IS NULL ;


-- CouponUsed

SET SQL_SAFE_UPDATES=0;


SET @CouponUsed_mode = (SELECT CouponUsed FROM customer_churn

                GROUP BY CouponUsed ORDER BY COUNT(*) DESC LIMIT 1) ;


UPDATE customer_churn

SET CouponUsed = @CouponUsed_mode

WHERE CouponUsed IS NULL ;


-- OrderCount

SET SQL_SAFE_UPDATES = 0;


SET @OrderCount_mode = (SELECT OrderCount FROM customer_churn

                GROUP BY OrderCount ORDER BY COUNT(*) DESC LIMIT 1);
```

```
UPDATE customer_churn
SET OrderCount = @OrderCount_mode
WHERE OrderCount IS NULL ;
```

We use these query for the column Tenure, CouponUsed, OrderCount to fill the null values

➢ Handle outliers in the 'WarehouseToHome' column by deleting rows where the values are greater than 100.

Query :

```
DELETE FROM customer_churn
WHERE WarehouseToHome > 100;
```

With the use of above query we deleting rows where the values are graeter than 100 in WareHouseToHome column.

Dealing with Inconsistencies:

➢ Replace occurrences of “Phone” in the 'PreferredLoginDevice' column and “Mobile” in the 'PreferedOrderCat' column with “Mobile Phone” to ensure uniformity.

Query :

```
-- PreferredLoginDevice
UPDATE customer_churn
SET PreferredLoginDevice = IF(PreferredLoginDevice = 'Phone' , 'Mobile Phone', PreferredLoginDevice);
-- PreferedOrderCat
UPDATE customer_churn
SET PreferedOrderCat  = IF(PreferedOrderCat = 'Mobile' , 'Mobile Phone',PreferedOrderCat );
```

In PreferredLoginDevice column we replace “Phone” into ‘Mobile Phone’

In PreferedOrderCat column we replace “Mobile” into ‘Mobile Phone’ for uniformity.

➢ Standardize payment mode values: Replace "COD" with "Cash on Delivery" and "CC" with "Credit Card" in the PreferredPaymentMode column.

Query :

```
UPDATE customer_churn
SET PreferredPaymentMode = CASE
        WHEN PreferredPaymentMode = 'COD' THEN 'Cash on Delivery'
        WHEN PreferredPaymentMode = 'CC' THEN 'Credit Card'
        ELSE PreferredPaymentMode
    END ;
```

In PreferredPaymentMode column we replace the value “COD” into ‘Cash On Delivery’.

Data Transformation:

Column Renaming:

➢ Rename the column "PreferedOrderCat" to "PreferredOrderCat".

➢ Rename the column "HourSpendOnApp" to "HoursSpentOnApp".

Query :

```
ALTER TABLE customer_churn
RENAME COLUMN PreferedOrderCat TO PreferredOrderCat ,
RENAME COLUMN HourSpendOnApp TO HoursSpentOnApp ;
```

With the use of above query we rename the coumn “PreferedOrderCat” into ‘PreferredOrderCat’ , “HourSpendOnApp” into ‘HoursSpentOnApp’’.

Creating New Columns:

➢ Create a new column named ‘ComplaintReceived’ with values "Yes" if the corresponding value in the ‘Complain’ is 1, and "No" otherwise.

➢ Create a new column named 'ChurnStatus'. Set its value to “Churned” if the corresponding value in the 'Churn' column is 1, else assign “Active”.

Query :

```
ALTER TABLE customer_churn
ADD COLUMN ComplaintReceived ENUM('Yes','No'),
ADD COLUMN Churnstatus ENUM('Churned','Active');

-- Set values for the new columns based on  existing data

SET SQL_SAFE_UPDATES = 0;

UPDATE customer_churn
SET ComplaintReceived = IF(Complain = 1 ,'Yes' , 'No'),
   ChurnStatus       = IF(Churn =1 , 'Churned','Active');
```

In ComplaintReceived column the complain get 1 we add ‘Yes” otherwise its ‘No’.

In Churnstatus column the churn is 1 we add ‘Churned’otherwise its ‘Active’.

Column Dropping:

➢ Drop the columns "Churn" and "Complain" from the table.

Query :

```
ALTER TABLE customer_churn
DROP COLUMN Churn,
DROP COLUMN Complain;
```

We drop the churn and complain column.

Data Exploration and Analysis:

➢ **Retrieve the count of churned and active customers from the dataset.**

Query :

```
SELECT Churnstatus , COUNT(*) AS churnstatus_count
FROM customer_churn
WHERE Churnstatus IN ('Churned','Active')
GROUP BY Churnstatus;
```

Result Grid | Filter Rows:

Churnstatus	churnstatus_count
Churned	948
Active	4680

➢ **Display the average tenure and total cashback amount of customers who churned.**

Query :

```
SELECT ROUND(AVG(Tenure))AS avg_tenure,
      SUM(CashbackAmount) AS total_cashback_amount
 FROM customer_churn
 WHERE Churnstatus IN ('Churned')
 ORDER BY Churnstatus;
```

Result Grid | Filter Rows:

avg_tenure	total_cashback_amount
3	152030

➢ **Determine the percentage of churned customers who complained.**

Query :

```
SELECT ComplaintReceived , CONCAT(ROUND(COUNT(*) / (SELECT COUNT(*)
FROM customer_churn)* 100,2) , '%' )
  AS percentage_churned_cust
  FROM customer_churn
  WHERE Churnstatus IN('Churned')
  GROUP BY ComplaintReceived;
```

Result Grid | Filter Rows:

ComplaintReceived	percentage_churned_cust
Yes	9.03%
No	7.82%

➢ **Find the gender distribution of customers who complained.**

Query :

```
SELECT gender ,COUNT(*) AS complained_gender_count
 FROM customer_churn
WHERE ComplaintReceived = 'Yes'
 GROUP BY gender;
```

Result Grid | Filter Rows:

gender	complained_gender_count
Female	690
Male	914

➢ **Identify the city tier with the highest number of churned customers whose preferred order category is Laptop & Accessory.**

Query :

```
SELECT CityTier, COUNT(*)
FROM customer_churn
```

WHERE Churnstatus = 'Churned'

AND preferredOrderCat = 'Laptop & Accessory'

GROUP BY CityTier

ORDER BY Churnstatus DESC LIMIT 1 ;

Result Grid | Filter

	CityTier	COUNT(*)
▶	3	150

➢ Identify the most preferred payment mode among active customers.

Query :

SELECT PreferredPaymentMode , COUNT(*)

AS Payment_count FROM customer_churn

WHERE Churnstatus = 'Active'

GROUP BY PreferredPaymentMode

ORDER BY Payment_count DESC LIMIT 1 ;

Result Grid | Filter Rows:

	PreferredPaymentMode	Payment_count
▶	Debit Card	1956

➢ Calculate the total order amount hike from last year for customers who are single and prefer mobile phones for ordering.

Query :

SELECT SUM(OrderAmountHikeFromlastYear)

AS total_order_amount_hike_fromlastyear

FROM customer_churn

WHERE Maritalstatus = 'Single'

AND PreferredOrderCat = 'Mobile Phone';

Result Grid | Filter Rows:

	total_order_amount_hike_fromlastyear
▶	12177

➢ **Find the average number of devices registered among customers who used UPI as their preferred payment mode.**

Query :

```
SELECT ROUND( AVG(NumberOfDeviceRegistered))
AS avg_number_of_device_registered
FROM customer_churn
WHERE PreferredPaymentMode = 'UPI'
GROUP BY PreferredPaymentMode  ;
```

Result Grid | Filter Rows:

	avg_number_of_device_registered
▶	4

➢ **Determine the city tier with the highest number of customers.**

Query :

```
SELECT CityTier,COUNT(*)
AS city_tier_count
FROM customer_churn
GROUP BY CityTier
ORDER BY city_tier_count DESC LIMIT 1 ;
```

Result Grid | Filter Rows:

	CityTier	city_tier_count
▶	1	3666

➢ **Identify the gender that utilized the highest number of coupons.**

Query:

```
SELECT gender,COUNT(CouponUsed) AS coupon_use_count
FROM customer_churn
GROUP BY gender
ORDER BY coupon_use_count DESC LIMIT 1 ;
```

Result Grid | Filter Rows:

gender	coupon_use_count
Male	3382

➢ **List the number of customers and the maximum hours spent on the app in each preferred order category.**

Query :

```
SELECT PreferredOrderCat, COUNT(CustomerID) AS customer_count,
    MAX(HoursSpentOnApp) AS max_houserspentonapp_cust
FROM customer_churn
GROUP BY PreferredOrderCat
ORDER BY max_houserspentonapp_cust;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PreferredOrderCat	customer_count	max_houserspentonapp_cust
Others	264	4
Grocery	410	4
Laptop & Accessory	2050	5
Mobile Phone	2078	5
Fashion	826	5

➢ **Calculate the total order count for customers who prefer using credit cards and have the maximum satisfaction score.**

Query :

```
SELECT SUM(OrderCount) AS total_order_count,
    MAX(SatisfactionScore) AS max_satisfation_score
FROM customer_churn
WHERE PreferredPaymentMode = 'Credit Card';
```

Result Grid | Filter Rows:

total_order_count	max_satisfation_score
5409	5

➢ **How many customers are there who spent only one hour on the app and days since their last order was more than 5?**

Query :

```
SELECT COUNT(*) AS cust_count
FROM customer_churn
WHERE HoursSpentOnApp = 1
AND DaySinceLastOrder > 5 ;
```

Result Grid

	cust_count
▸	8

➢ **What is the average satisfaction score of customers who have complained?**

Query :

```
SELECT ComplaintReceived,ROUND(Avg(SatisfactionScore)) AS
avg_satisfaction_score
 FROM customer_churn
 WHERE ComplaintReceived = 'Yes'
 GROUP BY ComplaintReceived ;
```

Result Grid | Filter Rows:

	ComplaintReceived	avg_satisfaction_score
▸	Yes	3

➢ **List the preferred order category among customers who used more than 5 coupons.**

Query :

```
SELECT PreferredOrderCat,COUNT(CouponUsed) AS coupon_use_cust
FROM customer_churn
WHERE CouponUsed > 5
```

GROUP BY PreferredOrderCat

ORDER BY coupon_use_cust;

Result Grid | Filter Rows:

PreferredOrderCat	coupon_use_cust
Others	28
Grocery	42
Mobile Phone	45
Fashion	89
Laptop & Accessory	99

➢ List the top 3 preferred order categories with the highest average cashback amount.

Query :

SELECT PreferredOrderCat,ROUND(AVG(CashbackAmount)) AS avg_cashback_amt

FROM customer_churn

GROUP BY PreferredOrderCat

ORDER BY avg_cashback_amt DESC LIMIT 3 ;

Result Grid | Filter Rows:

PreferredOrderCat	avg_cashback_amt
Others	304
Grocery	266
Fashion	210

➢ Find the preferred payment modes of customers whose average tenure is 10 months and have placed more than 500 orders.

Query :

SELECT PreferredPaymentMode , ROUND(AVG(Tenure)) AS avg_tenure_10months ,

COUNT(OrderCount) AS ordercount_morethan_500

FROM customer_churn

GROUP BY PreferredPaymentMode

ORDER BY avg_tenure_10months DESC LIMIT 3 ;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PreferredPaymentMode	avg_tenure_10months	ordercount_morethan_500
Debit Card	10	2312
Credit Card	10	1774
E wallet	10	614

➢ Categorize customers based on their distance from the warehouse to home such as 'Very Close Distance' for distances <=5km, 'Close Distance' for <=10km, 'Moderate Distance' for <=15km, and 'Far Distance' for >15km. Then, display the churn status breakdown for each distance category.

Query :

```
SELECT
 CASE WHEN WareHouseToHome <=5 THEN 'Very Close Distance'
 WHEN WareHouseToHome <=10  THEN 'Close Distance'
 WHEN WareHouseToHome <=15 THEN 'Moderate Distance'
 ELSE 'Far Distance'
 END AS WareHouseToHome,ChurnStatus,
 COUNT(CustomerID) AS cust_count
 FROM customer_churn
 GROUP BY WareHouseToHome ,ChurnStatus
 ORDER BY WareHouseToHome ,ChurnStatus ;
```

Result Grid | Filter Rows:

WareHouseToHome	ChurnStatus	cust_count
Close Distance	Churned	34
Close Distance	Churned	58
Close Distance	Churned	80
Close Distance	Churned	44
Close Distance	Churned	49
Close Distance	Active	261
Close Distance	Active	230
Close Distance	Active	386
Close Distance	Active	479
Close Distance	Active	340
Far Distance	Churned	26

➢ List the customer's order details who are married, live in City Tier-1, and their order counts are more than the average number of orders placed by all customers.

Query :

```
WITH avg_ordercount  AS (SELECT ROUND(AVG(OrderCount))
 AS avg_order FROM customer_churn)
 SELECT CustomerID,Ordercount,
ROUND(AVG(OrderCount)) AS avg_ordercount
    FROM customer_churn
    WHERE MaritalStatus = 'Married'
    AND CityTier = 1
GROUP BY CustomerID
HAVING (SELECT avg_order FROM avg_ordercount) < OrderCount
ORDER BY OrderCount  ;
```

Result Grid | Filter Rows:

CustomerID	Ordercount	avg_ordercount
50119	4	4
50367	4	4
50385	4	4
50453	4	4
50727	4	4
50734	4	4
51060	4	4
51084	4	4
51198	4	4
51227	4	4
51253	4	4

➢ a) Create a 'customer_returns' table in the 'ecomm' database and insert the following data:

ReturnID	CustomerID	ReturnDate	RefundAmount
1001	50022	2023-01-01	2130
1002	50316	2023-01-23	2000
1003	51099	2023-02-14	2290
1004	52321	2023-03-08	2510
1005	52928	2023-03-20	3000
1006	53749	2023-04-17	1740
1007	54206	2023-04-21	3250
1008	54838	2023-04-30	1990

Query :

```
CREATE TABLE customer_returns ( ReturnID INT PRIMARY KEY,
 CustomerID INT UNIQUE ,
 ReturnDate DATE DEFAULT (current_date),
 RefundAmount DECIMAL(10,2)
 );

 INSERT INTO
customer_returns(ReturnID,CustomerID,ReturnDate,RefundAmount) values
 (1001 , 50022 , '2023-01-01' , 2130),
 (1002 , 50316 , '2023-01-23' , 2000),
 (1003 , 51099 , '2023-02-14' , 2290),
 (1004 , 52321 , '2023-03-08' , 2510),
 (1005 , 52928 , '2023-03-20' , 3000),
 (1006 , 53749 , '2023-04-17' , 1740),
 (1007 , 54206 , '2023-04-21' , 3250),
 (1008 , 54838 , '2023-04-30' , 1990);
```

We create customer_returns table and insert the values.

b) Display the return details along with the customer details of those who have churned and have made complaints.

Query :

```
SELECT r.ReturnID,r.ReturnDate,r.RefundAmount,
 c.CustomerID,c.ChurnStatus,c.ComplaintReceived
 FROM customer_churn AS c
JOIN customer_returns AS r
```

```
ON c.CustomerID = r.CustomerID
WHERE ChurnStatus = 'Churned'
AND ComplaintReceived = 'Yes' ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ReturnID	ReturnDate	RefundAmount	CustomerID	ChurnStatus	ComplaintReceived
1002	2023-01-23	2000.00	50316	Churned	Yes
1004	2023-03-08	2510.00	52321	Churned	Yes
1006	2023-04-17	1740.00	53749	Churned	Yes

Conclusion :

I hope the above queries are helpful to understand the database ecomm and all the questions are rectified by the queries . we get the clear information of the customer_churn table.with this sql queries help to change the null values,rename the columns ,add and creating column ,values etc..and finally we create another table named customer_returns finally we joined the both table to find out the customer details.