E-Commerce Customers Churn Analysis Using SQL

Introduction:

The data set belongs to a leading online E-Commerce company. An online retail (E commerce) company wants to know the customers who are going to churn, so accordingly they can approach customer to offer some promos.

In the fast-paced world of online retail, e-commerce companies face the challenge of retaining customers. To address this, identifying at-risk customers and implementing targeted retention strategies are crucial.

Problem Statement:

The goal of this project is to detailed analysis of a dataset from an online retail company, revealing valuable insights on customer churn. These insights offer essential guidance for decision-making, enabling proactive measures to reduce attrition and foster long-term loyalty.

Project Approach:

Dataset contains information such as customers' personal details, satisfaction scores, preferred payment mode, days since the last order, and cashback amount. I used SQL (Postgres SQL) to clean and analyze this dataset, and performed visualizations using Microsoft Power BI. This analysis is divided into several stages: data cleaning, data exploration, an insight section.

Data Dictionary:

Variable	Discription
CustomerID	Unique customer ID
Churn	Churn Flag
Tenure	Tenure of customer in organization
PreferredLoginDevice	Preferred login device of customer
CityTier	City tier
WarehouseToHome	Distance in between warehouse to home of customer
PreferredPaymentMode	Preferred payment method of customer
Gender	Gender of customer
HourSpendOnApp	Number of hours spend on mobile application or website
NumberOfDeviceRegistered	Total number of deceives is registered on particular customer
PreferedOrderCat	Preferred order category of customer in last month
SatisfactionScore	Satisfactory score of customers on service
MaritalStatus	Marital status of customer
NumberOfAddress	Total number of added on particular customer
Complain	Any complaint has been raised in last month
OrderAmountHikeFromlastYear	Percentage increases in order from last year
CouponUsed	Total number of coupons has been used in last month
OrderCount	Total number of orders has been places in last month
DaySinceLastOrder	Day Since last order by customer
CashbackAmount	Average cashback in last month

Data Cleaning:

E-Commerce Customers Churn Analysis Using SQL

Before going to analysis, it is essential to ensure the dataset is clean and reliable. The data cleaning process involves handling missing values, correcting inconsistencies, and formatting the data for analysis. In this project, we carefully cleaned the dataset to ensure the accuracy and integrity of our findings.

(1) Creation of Customer_churn Table

Query Query History

```sql
-- Creation customer_churn Table
CREATE TABLE customer_churn(
CustomerID integer, Churn integer, Tenure integer, PreferredLoginDevice text, CityTier integer,
WarehouseToHome integer, PreferredPaymentMode text, Gender  text, HourSpendOnApp integer,
NumberOfDeviceRegistered integer, PreferedOrderCat text, SatisfactionScore text,MaritalStatus text,
NumberOfAddress integer, Complain integer, OrderAmountHikeFromlastYear integer, CouponUsed integer,
OrderCount integer, DaySinceLastOrder integer, CashbackAmount integer
)
```

(2) Total No. of Customers

Query Query History

```sql
-- Total Number of customer
select distinct count(customerid) as Total_customers
from customer_churn
```

Data Output Messages

	total_customers bigint
1	5630

Findings: There are 5630 customers in given dataset

(3) Checking for Duplicate Values

```sql
-- (2) Checking for Duplicate values
select
customerid,
count(customerid) as count
from customer_churn
group by customerid
having count(customerid)>1
```

Data Output Messages Notifications

	customerid integer	count bigint

Findings: The result showing empty table that means there is no duplicate values in the dataset

(4) Checking for Null Values

E-Commerce Customers Churn Analysis Using SQL

```sql
-- (3) Checking for Null values
select 'churn' as column_name, count(*) as NullCount
from customer_churn
where churn is null
union
select 'tenure' as columns_name, count(*) as NullCount
from customer_churn
where tenure is null
union
select 'PreferredLoginDevice' as column_name, count(*) as NullCount
from customer_churn
where PreferredLoginDevice is null
union
select 'CityTier' as column_name, count(*) as NullCount
from customer_churn
where CityTier is null
union
select 'WarehouseToHome' as column_name, count(*) as NullCount
from customer_churn
where WarehouseToHome is null
union
select 'PreferredPaymentMode' as column_name, count(*) as NullCount
from customer_churn
where PreferredPaymentMode is null
union
select 'Gender' as column_name, count(*) as NullCount
from customer_churn
where Gender is null
union
```

```sql
union
select 'HourSpendOnApp' as column_name, count(*) as NullCount
from customer_churn
where HourSpendOnApp is null
union
select 'NumberOfDeviceRegistered' as column_name, count(*) as NullCount
from customer_churn
where NumberOfDeviceRegistered is null
union
select 'PreferedOrderCat' as column_name, count(*) as NullCount
from customer_churn
where PreferedOrderCat is null
union
select 'SatisfactionScore' as column_name, count(*) as NullCount
from customer_churn
where SatisfactionScore is null
union
select 'MaritalStatus' as column_name, count(*) as NullCount
from customer_churn
where MaritalStatus is null
union
```

E-Commerce Customers Churn Analysis Using SQL

```sql
union
select 'NumberOfAddress' as column_name, count(*) as NullCount
from customer_churn
where NumberOfAddress is null
union
select 'Complain' as columns_name, count(*) as NullCount
from customer_churn
where Complain is null
union
select 'OrderAmountHikeFromlastYear' as columns_name, count(*) as NullCount
from customer_churn
where OrderAmountHikeFromlastYear is null
union
select 'CouponUsed' as column_name, count(*) as NullCount
from customer_churn
where CouponUsed is null
union
select 'OrderCount' as column_name, count(*) as NullCount
from customer_churn
where OrderCount is null
union
select 'DaySinceLastOrder' as column_name, count(*) as NullCount
from customer_churn
where DaySinceLastOrder is null
union
select 'CashbackAmount' as column_name, count(*) as NullCount
from customer_churn
where CashbackAmount is null
```

Data Output Messages Graph Visualiser × Notifications

	column_name text	nullcount bigint
1	PreferredPaymentMode	0
2	tenure	264
3	HourSpendOnApp	255
4	WarehouseToHome	251
5	churn	0
6	Gender	0
7	CityTier	0
8	PreferredLoginDevice	0
9	DaySinceLastOrder	307
10	NumberOfAddress	0
11	OrderCount	258
12	OrderAmountHikeFromlastYear	265
13	Complain	0
14	MaritalStatus	0
15	CouponUsed	256
16	PreferedOrderCat	0
17	NumberOfDeviceRegistered	0
18	CashbackAmount	0
19	SatisfactionScore	0

Findings: CouponUsed, DaysSinceLastOrder, HourSpendOnApp, OrderAmountHikeFromLastYear, OrderCount, Tenure, and WarehouseToHome all have null values present, and the number of null values present for each column can be seen in the above table.

E-Commerce Customers Churn Analysis Using SQL

(4) Handling Null Values

```sql
-- (3.1) Handling Null Values
update customer_churn
set tenure = (select avg(tenure) from customer_churn)
where tenure is null

update customer_churn
set HourSpendOnApp = (select avg(HourSpendOnApp) from customer_churn)
where HourSpendOnApp is null

update customer_churn
set WarehouseToHome = (select avg(WarehouseToHome)from customer_churn)
where WarehouseToHome is null

update customer_churn
set DaySinceLastOrder = (select avg(DaySinceLastOrder)from customer_churn)
where DaySinceLastOrder is null

update customer_churn
set OrderCount = (select avg(OrderCount)from customer_churn)
where OrderCount is null

update customer_churn
set OrderAmountHikeFromlastYear = (select avg(OrderAmountHikeFromlastYear)from customer_churn)
where OrderAmountHikeFromlastYear is null

update customer_churn
set CouponUsed = (select avg(CouponUsed)from customer_churn)
where CouponUsed is null
```

Data Output Messages Notifications

	column_name text	nullcount bigint
1	CouponUsed	0
2	churn	0
3	Gender	0
4	CityTier	0
5	PreferredLoginDevice	0
6	OrderAmountHikeFromlastYear	0
7	NumberOfAddress	0
8	DaySinceLastOrder	0
9	Complain	0
10	MaritalStatus	0
11	HourSpendOnApp	0
12	PreferedOrderCat	0
13	tenure	0
14	NumberOfDeviceRegistered	0
15	CashbackAmount	0
16	SatisfactionScore	0
17	WarehouseToHome	0
18	OrderCount	0
19	PreferredPaymentMode	0

Findings: so, there is no null values in the dataset

(4) Creating new column for an already existing column "Churn"

we can observe the "churn" column contained 0 and 1 values.o means that customer did not churn, while 1 means that customer churned.it is difficult to remember this,i we will create a new column called "customer_status" that shows "stayed" or "churned

E-Commerce Customers Churn Analysis Using SQL

```sql
alter table customer_churn
add customer_status nvarchar(50)

update customer_churn
set customer_status =
case
    when churn = 1 then 'churned'
    when churn = 0 then 'stayed'
end

select distinct customer_status from customer_churn
```

Data Output Messages Notifi

	customer_status character varying (50)
1	stayed
2	churned

Findings: The new column “customer_status” has affected in table.it has two distinct values “churned”, “stayed”.

(5) Creating new column for an already existing column "complain"

we noticed that the complaint column also contained 0 and 1. ‘0’ means that the customer did not record any complaints, while ‘1’ means the customer recorded a complaint. For clarity purposes, we will create a new column called ‘complain_recieved’ that shows “No” when a customer did not complain and “Yes” when a customer complained.

```sql
alter table customer_churn
add complaint_received nvarchar(10)

update customer_churn
set complaint_received =
case
    when complain = 1 then 'yes'
    when complain = 0 then 'no'
end

select distinct complaint_received from customer_churn
```

Data Output Messages Not

	complaint_received character varying (10)
1	no
2	yes

Findings: the new columns complaint_received affected in the database and it has two distinct values “yes”, “no”.

(6) Checking Redundancies in each column

(6.1) Checking redundancies in “PreferredLoginDevice” column

```sql
-- (6)(1) Fixing redundancey in "preferredlogindevice" column
select distinct(Preferredlogindevice)
from customer_churn
```

Data Output Messages

	preferredlogindevice text
1	Mobile Phone
2	Computer
3	Phone

Findings: I have observed phone and mobile phone in the same column, but they mean same thing. so, we will replace the mobile phone with phone.

E-Commerce Customers Churn Analysis Using SQL

```sql
update customer_churn
set preferredlogindevice = 'Phone'
where preferredlogindevice = 'Mobile Phone'

select distinct preferredlogindevice from customer_churn
```

	preferredlogindevice text
1	Computer
2	Phone

(6.2) Checking redundancies in "preferedordercat" column

```sql
-- (6.2) fixing redundancy in "PreferedOrderCat"
select distinct preferedordercat
from customer_churn
```

	preferedordercat text
1	Grocery
2	Mobile
3	Mobile Phone
4	Others
5	Laptop & Accessory
6	Fashion

Findings: I have observed mobile phone and phone appear in the column, but their meaning is same.so we will replace the phone with mobile phone.

```sql
update customer_churn
set preferedordercat = 'Mobile Phone'
where preferedordercat = 'Mobile'

select distinct preferedordercat from customer_churn
```

	preferedordercat text
1	Grocery
2	Mobile Phone
3	Others
4	Laptop & Accessory
5	Fashion

Findings: I have observed Finally redundancy in "preferedordecat" column has been fixed

(6.3) Checking redundancies in "preferredpaymentmode" column

```sql
-- (6.3) Fixing redundancy in "preferredpaymentmode" column
select distinct preferredpaymentmode
from customer_churn
```

	preferredpaymentmode text
1	UPI
2	E wallet
3	CC
4	Debit Card
5	Cash on Delivery
6	COD
7	Credit Card

E-Commerce Customers Churn Analysis Using SQL

Findings: **I have observed "Cash on Delivery" and "COD" both appear in same column,but their meaning is same. So we will replace COD with Cash on Delivery.**

```sql
update customer_churn
set preferredpaymentmode = 'Cash on Delivery'
where preferredpaymentmode = 'COD'

select distinct preferredpaymentmode from customer_churn
```

	preferredpaymentmode text
1	UPI
2	E wallet
3	CC
4	Debit Card
5	Cash on Delivery
6	Credit Card

Findings: **I have observed the redundancy in "preferredpaymentmode" has been fixed.**

(6.4) Fixing wrongly entered values in "warehousetohome" column

```sql
-- (6.4) Fixing wrongly entered values in "warehousetohome" column
select distinct warehousetohome
from customer_churn
```

	warehousetohome integer
24	33
25	13
26	5
27	18
28	127
29	16
30	27
31	23
32	126

Findings: **I have observed 126 and 127 values. They are definitely outliers and most likely wrongly entered.to fix this we will replace those values with 26 and 27 to fall within range of the values.**

```sql
-- (6.4) Fixing wrongly entered values in "warehousetohome" column
select distinct warehousetohome
from customer_churn

update customer_churn
set warehousetohome = '27'
where warehousetohome = '127'

update customer_churn
set warehousetohome = '26'
where warehousetohome = '126'
```

	warehousetohome integer
23	22
24	33
25	13
26	5
27	18
28	16
29	27
30	23
31	8
32	11

Findings: **I have observed 126 and 127 values are replaced with 26 and 27.**

Now data cleaning is completed , so we will look into Data Exploration.

E-Commerce Customers Churn Analysis Using SQL

Data Exploration:

(1) The Overall customer churn rate.

```sql
/* Data Exploration
(1) The overall customer churn rate.
*/
select Totalnumberofcustomers,
Totalnumberofchurnedcustomers,
cast((Totalnumberofchurnedcustomers*1.0/Totalnumberofcustomers*1.0)*100 as decimal(10,2)) as churn_rate
from
(select count(*) as Totalnumberofcustomers
 from customer_churn)as total,
(select count(*) as Totalnumberofchurnedcustomers
from customer_churn
where customer_status = 'churned') as churned
```

Data Output Messages Graph Visualiser × Notifications

	totalnumberofcustomers bigint	totalnumberofchurnedcustomers bigint	churn_rate numeric (10,2)
1	5630	948	16.84

Findings: **The overall customer churn_rate is approx 17%.**

(2) Churn_Rate based on preferred login device.

```sql
-- (2) Churn_rate based on preferred login device
select preferredlogindevice,
count(*) as Totalcustomers,
sum(churn) as churnedcustomers,
cast((sum(churn)*1.0/count(*))*100 as decimal(10,2)) as churn_rate
from customer_churn
group by preferredlogindevice

```

Data Output Messages Graph Visualiser × Notifications

	preferredlogindevice text	totalcustomers bigint	churnedcustomers bigint	churn_rate numeric (10,2)
1	Phone	3996	624	15.62
2	Computer	1634	324	19.83

Findings:

(1) **Customers who prefer login using computer have a slightly higher churn_rate compared to customers who prefer login using phone.**

E-Commerce Customers Churn Analysis Using SQL

(2) **Appox 16% customers churned who are prefer to login using phone.**
(3) **Approx 20% of the customers churned who are prefer to login using computer.**
(4) **Most of the customers churned who are using computer.**

(3) Churn_Rate based on different city tiers.

```sql
-- (3) Churn rate based on city tiers
select
citytier,
count(*) as totalcustomer,
sum(churn) as churnedcustomers,
cast((sum(churn)*1.0/count(*))*100 as decimal(10,2))as churn_rate
from customer_churn
group by citytier
order by churn_rate
```

Data Output | Messages | Graph Visualiser × | Notifications

	citytier integer	totalcustomer bigint	churnedcustomers bigint	churn_rate numeric (10,2)
1	1	3666	532	14.51
2	2	242	48	19.83
3	3	1722	368	21.37

Findings:

(1) **Customers who prefer login using computer have a slightly higher churn_rate compared to customers who prefer login using phone.**
(2) **Most of the cutomers churned from city tier1**
(3) **City tier 1 has lower churn_rate compare to the tier 2 and tier 3 cities and city tier 3 is highest churn_rate.**

(4) Churn_Rate based on warehouse to home distance.

We will create a new column called "WareHouseToHomeRange" that groups the distance into very close, close, moderate and far using CASE Statement

```sql
-- (4) Churn_rate by WareHouseToHomeDistance
/* now we will create new colum "warehousetohomerange"
*/
alter table customer_churn
add warehousetohomerange varchar(50)

update customer_churn
set warehousetohomerange =
CASE
    when warehousetohome <= 10 then 'Very Close Distance'
    when warehousetohome > 10 and warehousetohome <= 20 then 'Close Distance'
    when warehousetohome > 20 and warehousetohome <= 30 then 'Moderate Distance'
    when warehousetohome > 30 then 'Far Distance'
end
```

E-Commerce Customers Churn Analysis Using SQL

```sql
select distinct warehousetohomerange from customer_churn
```

Data Output | Messages | Gra

	warehousetohomerange character varying (50)
1	Far Distance
2	Close Distance
3	Moderate Distance
4	Very Close Distance

```sql
select warehousetohomerange,
count(*) as Total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
Group by 1
order by 4 desc
```

	warehousetohomerange character varying (50)	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	Very Close Distance	1969	266	13.51
2	Close Distance	2318	408	17.60
3	Moderate Distance	874	176	20.14
4	Far Distance	469	98	20.90

Findings: **The distance between warehouse and customer's home seems to have some influence on churn_rate.**

(1) Customers who residing very close to warehouse tend to have lower churn_rate.
(2) Customers who residing very far to warehouse tend to have higher churn_rate, so we will conclude that warehousetohome distance influence the churn_rate.

(5) Churn_Rate based on preferred payment mode.

```sql
-- (5)  Churn_rate by preferredpaymentmode
select preferredpaymentmode,
count(*) as Total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
group by 1
order by 4 desc
```

E-Commerce Customers Churn Analysis Using SQL

	preferredpaymentmode text	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	Cash on Delivery	514	128	24.90
2	E wallet	614	140	22.80
3	CC	273	59	21.61
4	UPI	414	72	17.39
5	Debit Card	2314	356	15.38
6	Credit Card	1501	193	12.86

Findings:

(1) Which customers had payment mode as "Cash on Delivery", those customers having higher churn rate.

(2) Payment mode like "Credit Card" and "Debit Card" have lower churn rate.

(3) The preferred payment mode has some influence on churn rate.

(6) Churn_Rate based on Tenure.

First of all, I have created new column called "tenure_range". That groups the customers tenure into 6months, 1 year, 2 years and more than 2 years using case statement

```sql
-- (6) Churn rate by Tenure
/* Create new column called "tenure_range"
*/
alter table customer_churn
add tenure_range varchar(20)

update customer_churn
set tenure_range =
case
    when tenure <=6 then '6 Months'
    when tenure >6 and tenure <=12 then '1 Year'
    when tenure >12 and tenure <=24 then '2 Years'
    when tenure >24 then 'More than 2 years'

end
```

Data Output Messages Not

	tenure_range character varying (20)
1	2 Years
2	6 Months
3	More than 2 years
4	1 Year

```sql
select tenure_range,
count(*) as Total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
group by 1
order by 4 desc
```

E-Commerce Customers Churn Analysis Using SQL

	tenure_range character varying (20)	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	6 Months	2150	697	32.42
2	1 Year	1584	156	9.85
3	2 Years	1467	95	6.48
4	More than 2 years	429	0	0.00

Findings:

(1) Most of the customers had churned whose tenure _range is 6 Months.

(2) The customers who had tenure range is more than 2 years, there were not interested to churn. i.e., the churn rate Is "0".

(7) Churn_Rate based on Gender.

```sql
-- (7) Churn_rate by Gender
select gender,
count(*) as Total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
group by 1
order by 4 desc
```

	gender text	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	Male	3384	600	17.73
2	Female	2246	348	15.49

Findings:

(1) Churn rate of Male customers has higher when compare to Female customer.

(2) 600 Male customers churned.

(8) Customer_status based on hourspentonapp.

```sql
-- (8) Churn_status by  average hourspendonapp
select customer_status,
avg(hourspendonapp) as averagehourspentonApp
from customer_churn
group by customer_status
```

	customer_status character varying (50)	averagehourspentonapp numeric
1	stayed	2.9286629645450662
2	churned	2.9641350210970464

Findings:

(1) Both Stayed and Churned customers had the same average hours spent on app.

(2) Hourspentonapp might not be influenced churned customers.

E-Commerce Customers Churn Analysis Using SQL

(9) Churn_Rate based on Numberofregistered devices.

```sql
-- (9) Churn_rate by NumberOfRegisteredDevices
select numberofdeviceregistered,
count(*) as Total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2))as Churn_rate
from customer_churn
group by 1
order by 4
```

	numberofdeviceregistered integer	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	1	235	22	9.36
2	2	276	26	9.42
3	3	1699	254	14.95
4	4	2377	392	16.49
5	5	881	198	22.47
6	6	162	56	34.57

Findings:

(1) **We can observe there is positive correlation between "Numberofdeviceregistered" and "Churn rate"**

(2) **Customers with less registered devices more likely to low churn rate.**

(3) **Numberofdeviceregistered is influence the churn rate.**

(10) Churn_Rate based on preferordercat.

```sql
-- (10) Churn_rate by PreferorderCat
select preferedordercat,
count(*) as total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as churn_rate
from customer_churn
group by 1
order by 4 desc
```

	preferedordercat text	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	Mobile Phone	2080	570	27.40
2	Fashion	826	128	15.50
3	Laptop & Accessory	2050	210	10.24
4	Others	264	20	7.58
5	Grocery	410	20	4.88

Findings:

(1) **Mobile Phone category had Highest Churn_rate compare to all other categories.**

(2) **Grocery category had lowest churn_rate.**

E-Commerce Customers Churn Analysis Using SQL

(11) Churn_Rate based on SatisfactionScores.

```
-- (11) Churn_rate by SatisfactionScores.
select satisfactionscore,
count(*) as total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
group by 1
order by 4 asc
```

	satisfactionscore text	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	1	1164	134	11.51
2	2	586	74	12.63
3	4	1074	184	17.13
4	3	1698	292	17.20
5	5	1108	264	23.83

Findings:

(1) The satisfaction score ranges from 1 to 5
(2) Most of customers had satisfaction score as 1 and 5
(3) Those who had Higher satisfaction score as 5, had higher churn_rate. And satisfaction score positive correlated with churn_rate.

(12) Churn_Rate based on Marital_Status.

```
-- (12) Churn_rate by Marrital_status
select maritalstatus,
count(*) as total_customers,
sum(churn) as Churned_Customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
group by 1
```

	maritalstatus text	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	Married	2986	344	11.52
2	Divorced	848	124	14.62
3	Single	1796	480	26.73

Findings:

(1) Single Customers have the highest churn_rate compare to customers with other marital_status
(2) Married customers have the lowest churn_rate followed by Divorced.

E-Commerce Customers Churn Analysis Using SQL

(13) Churn_Rate based on Numberof Address.

```sql
-- (13) Churn_rate by avrage NumberofAddresses
select avg(numberofaddress) as AverageNumberofchurnedcustomers
from customer_churn
where customer_status = 'churned'
```

Data Output | Messages | Notifications

	averagenumberofchurnedcustomers numeric
1	4.4662447257383966

Findings:

(1) **Customers churned who were 4 addresses changed on an Average.**

(14) Churn_Rate based on Complaints_received.

```sql
-- (14) Churn_rate by Complaint_received
select complaint_received,
count(*) as Total_customers,
sum(churn) as Churned_customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as Churn_rate
from customer_churn
group by 1
order by 4
```

	complaint_received character varying (10)	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	no	4026	440	10.93
2	yes	1604	508	31.67

Findings:

(1) **The customers who had raised the complaint last month they were more likely to churn.**
(2) **Churn_rate of the complaint raised is more than complaints not raised customers.**

(15) Customer_Status by by Couponsused.

```sql
-- (15) customer_status by Couponused
select customer_status,
sum(couponused) as SumofCouponsUsed
from customer_churn
group by customer_status
```

	customer_status character varying (50)	sumofcouponsused bigint
1	stayed	8292
2	churned	1630

Findings:

(1) **Most of the Stayed customers used coupons.**
(2) **churned customers used 1630 coupons.**

E-Commerce Customers Churn Analysis Using SQL

(16) Average Number of days since last order for churned customer.

```sql
-- (16) Average Numberofdays since the last for churned customers.
select avg(daysincelastorder) as AverageNumofdayssinceLastOrder
from customer_churn
where customer_status = 'churned'
```

Data Output Messages Notifications

	averagenumofdayssincelastorder numeric
1	3.3364978902953586

Findings:

(1) the churned customers have, on average, a short period of time since their last order indicates that they recently stopped engaging with the company. By focusing on the overall customer experience, implementing targeted retention initiatives, and maintaining continuous engagement.

(17) Churn_rate by CashbackAmount.

First of all I have created new column called "CashbackAmountRange" that groups the cashbackamount into low (<100), Moderate (between 100 and 200), High(between 200 and 300), and very High (more than 300) using the CASE Statement

```sql
-- (17) Churn_rate by CashbackAmount
-- Create new column called cashbackamountRange
alter table customer_churn
add cashbackamountRange varchar(50)

update customer_churn
set cashbackamountrange =
case
    when cashbackamount <= 100 then 'Low Cashback Amount'
    when cashbackamount > 100 and cashbackamount <= 200 then 'Moderate Cashback Amount'
    when cashbackamount > 200 and cashbackamount <= 300 then 'High Cashback Amount'
    when cashbackamount >300 then 'Very High Cashback Amount'
end
```

```sql
select cashbackamountRange,
count(*) as total_customers,
sum(churn) as Churned_Customers,
cast((sum(churn)*1.0/count(*)*1.0)*100 as decimal(10,2)) as churn_rate
from customer_churn
group by 1
order by 4 asc
```

	cashbackamountrange character varying (50)	total_customers bigint	churned_customers bigint	churn_rate numeric (10,2)
1	Low Cashback Amount	12	0	0.00
2	Very High Cashback Amount	156	10	6.41
3	High Cashback Amount	1157	124	10.72
4	Moderate Cashback Amount	4305	814	18.91

E-Commerce Customers Churn Analysis Using SQL

Findings:

(1) **Customers who had moderate cashback amount had a relatively higher churn_rate.**
(2) **Those who received Higher and very High cashback amount having lower churn_rate.**

Key Insights of Analysis

- **The dataset has 5630 customers.**
- **The overall churn rate is 16.84%.**
- **Customers who prefer logging in with a computer have slightly higher churn rate compare to phone users.**
- **Tier 1 cities have lower churn rates than tier 2 and tier 3 cities.**
- **Customers who residing very close to warehouse tend to have lower churn rate and Customers who residing very far to warehouse tend to have higher churn rate.**
- **"Cash on Delivery" and "E wallet" payment modes have higher churn rates, while "Credit Card" and "Debit Card" have lower churn rates.**
- **Longer tenure is associated with lower churn rates.**
- **Male customers have slightly higher churn rates than Female Customers.**
- **Both Stayed and Churned customers had the same average hours spent on app.**
- **"Mobile Phone" order category has the highest churn rate, while "Grocery" has the lowest churn rate.**
- **More registered devices correlated with higher churn rates.**
- **Highly satisfied customers (rating 5) have a relatively higher churn rate.**
- **Single customers have the highest churn rate, while married customers have lower churn rate.**
- **Churned customers have an average of 4 associated addresses.**
- **Customer complaints are prevalent among churned customers, emphasizing the importance of addressing concerns to minimize churn.**
- **Coupon usage is higher among non-churned customers.**
- **Moderate cashback amounts correspond to higher churn rates, while higher amounts leads to lower churn rates.**