E-Commerce Customer Churn Analysis

Dataset Download:

https://drive.google.com/uc?export=download&id=1iKKCze_Fpk2n_g3BIZBiSjcDFdFcEn3D

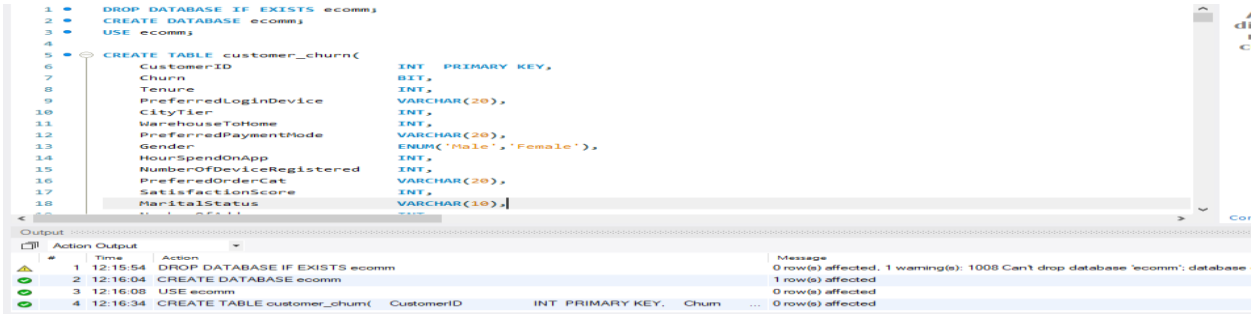
- DROP DATABASE IF EXISTS ecomm;
- CREATE DATABASE ecomm;
- USE ecomm;

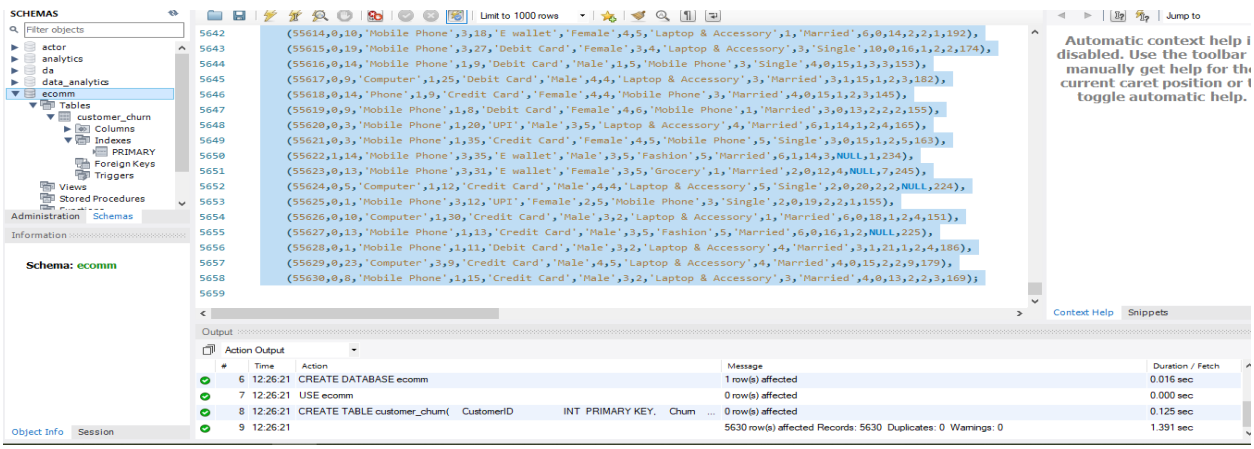
```
CREATE TABLE customer_churn(
    CustomerID                  INT  PRIMARY KEY,
    Churn                       BIT,
    Tenure                      INT,
    PreferredLoginDevice        VARCHAR(20),
    CityTier                    INT,
    WarehouseToHome             INT,
    PreferredPaymentMode        VARCHAR(20),
    Gender                      ENUM('Male','Female'),
    HourSpendOnApp              INT,
    NumberOfDeviceRegistered    INT,
    PreferedOrderCat            VARCHAR(20),
    SatisfactionScore           INT,
    MaritalStatus               VARCHAR(10),
    NumberOfAddress             INT,
    Complain                    BIT,
    OrderAmountHikeFromlastYear INT,
    CouponUsed                  INT,
    OrderCount                  INT,
```

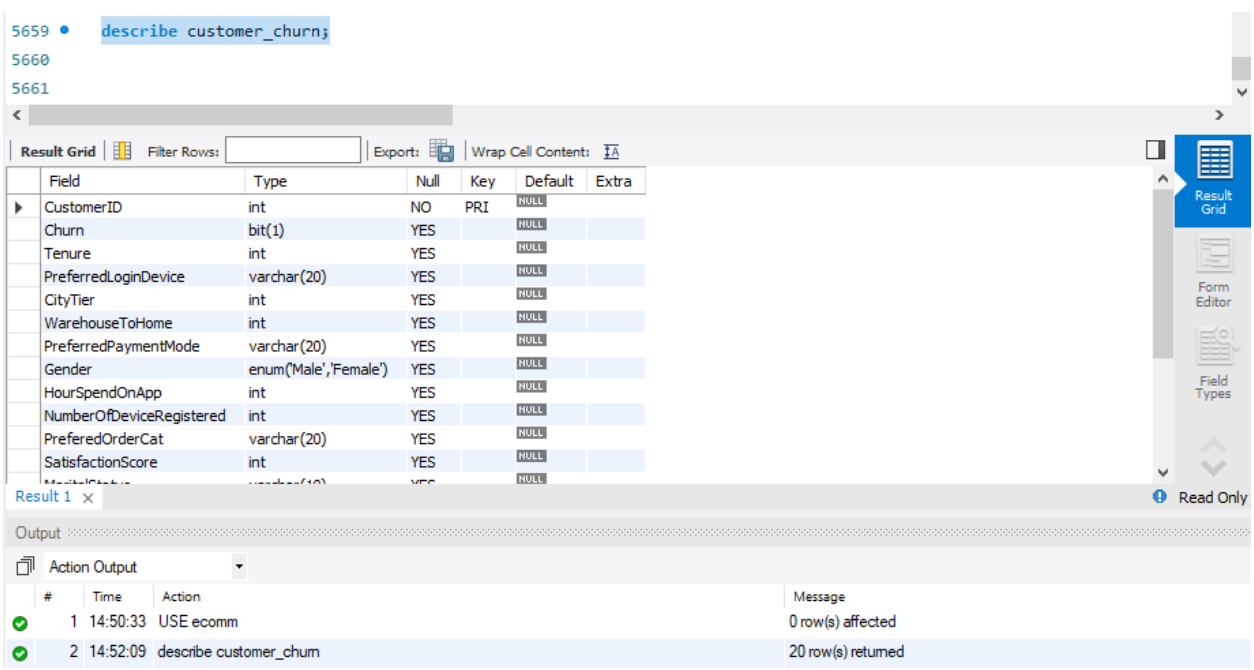
DaySinceLastOrder INT,

CashbackAmount INT

);







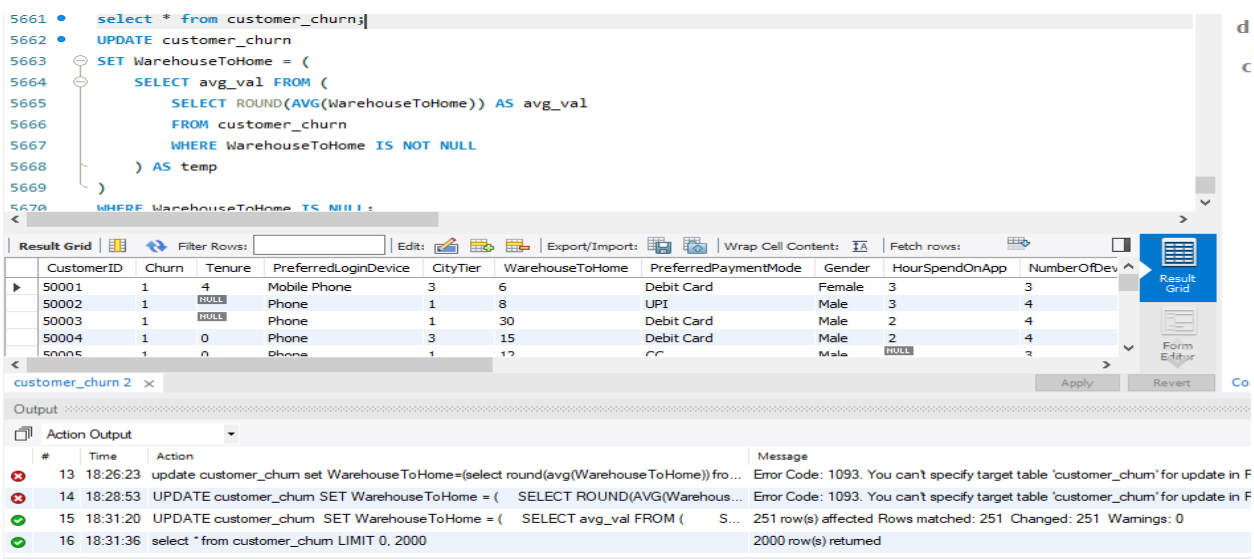
Data Cleaning:

Mean:

Updating WarehouseToHome:

```
UPDATE customer_churn
  SET WarehouseToHome = (
    SELECT avg_val FROM (
      SELECT ROUND(AVG(WarehouseToHome)) AS avg_val
       FROM customer_churn
       WHERE WarehouseToHome IS NOT NULL
  ) AS temp
)
WHERE WarehouseToHome IS NULL;
```

- The inner query SELECT ROUND(AVG(...)) runs first and stores the result in a temporary derived table (temp).
- From above syntax it calculates average value of already existing value for that it is mention as is not null. One WHERE to calculate the mean correctly (IS NOT NULL) One WHERE to apply the update only where it's needed (IS NULL)
- And other where clause is used as a filling values where there is a Null value.Then, the outer UPDATE can use this result safely without directly referencing customer_churn again.



similarly for HourSpendOnApp, OrderAmountHikeFromlastYear, DaySinceLastOrder.

Syntax: HourSpendOnApp

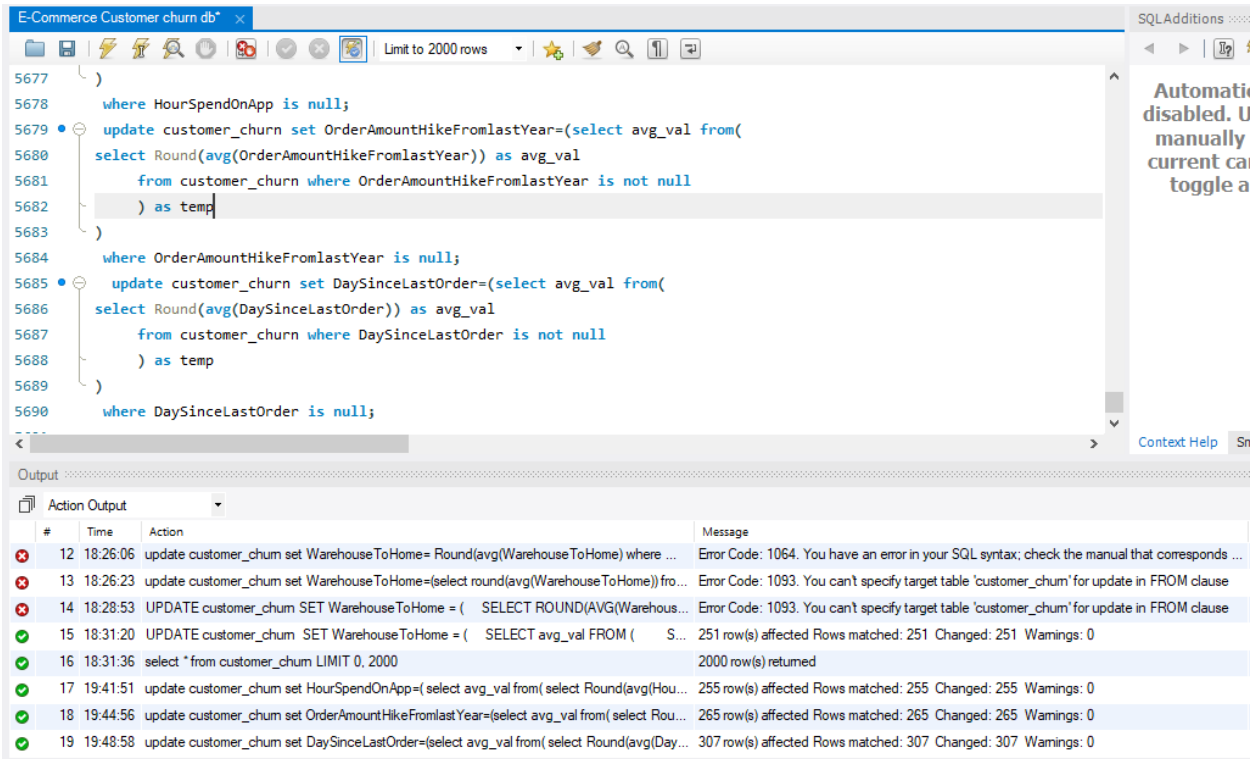
```
update customer_churn set HourSpendOnApp=(
select avg_val from(
select Round(avg(HourSpendOnApp)) as avg_val
    from customer_churn
    where HourSpendOnApp is not null
    ) as temp
)
 where HourSpendOnApp is null;
```

syntax: Orderamounthikefromlastyear

```
 update customer_churn set OrderAmountHikeFromlastYear=(select avg_val from(
select Round(avg(OrderAmountHikeFromlastYear)) as avg_val
    from customer_churn where OrderAmountHikeFromlastYear is not null
    ) as temp
)
 where OrderAmountHikeFromlastYear is null;
```

syntax: DaySinceLastOrder

```
  update customer_churn set DaySinceLastOrder=(select avg_val from(
select Round(avg(DaySinceLastOrder)) as avg_val
    from customer_churn where DaySinceLastOrder is not null
    ) as temp
)
 where DaySinceLastOrder is null;
```



MODE:

-- Impute mode for categorical columns

```
UPDATE customer_churn SET Tenure = (
  SELECT Tenure FROM (
    SELECT Tenure FROM customer_churn
    WHERE Tenure IS NOT NULL
    GROUP BY Tenure ORDER BY COUNT(*) DESC LIMIT 1
  ) AS temp
) WHERE Tenure IS NULL;
```

- **For coupon used:**

```
UPDATE customer_churn SET couponused = (SELECT couponused FROM (
    SELECT couponused FROM customer_churn WHERE couponused IS NOT NULL
    GROUP BY couponused ORDER BY COUNT(*) DESC LIMIT 1
  ) AS temp) WHERE couponused IS NULL;
```

- **For mode for OrderCount:**

UPDATE customer_churn SET OrderCount = (SELECT OrderCount FROM (

SELECT OrderCount FROM customer_churn WHERE OrderCount IS NOT NULL

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

) AS temp) WHERE OrderCount IS NULL;

```
5692    -- mode
5693    -- Impute mode for categorical columns
5694 UPDATE customer_churn SET Tenure = (SELECT Tenure FROM (
5695        SELECT Tenure FROM customer_churn WHERE Tenure IS NOT NULL
5696        GROUP BY Tenure ORDER BY COUNT(*) DESC LIMIT 1
5697      ) AS temp) WHERE Tenure IS NULL;
5698    -- mode for  coupon used
5699 UPDATE customer_churn SET couponused = (SELECT couponused FROM (
5700        SELECT couponused FROM customer_churn WHERE couponused IS NOT NULL
5701        GROUP BY couponused ORDER BY COUNT(*) DESC LIMIT 1
5702      ) AS temp) WHERE couponused IS NULL;
5703    -- mode for OrderCount
5704 UPDATE customer_churn SET OrderCount = (SELECT OrderCount  FROM (
5705        SELECT OrderCount  FROM customer_churn WHERE OrderCount  IS NOT NULL
5706        GROUP BY OrderCount  ORDER BY COUNT(*) DESC LIMIT 1
5707      ) AS temp) WHERE OrderCount  IS NULL;
```

Output

Action Output

#	Time	Action	Message
3	12:34:52	UPDATE customer_churn SET Tenure = (SELECT Tenure FROM (SELECT Tenure FRO...	264 row(s) affected Rows matched: 264 Changed: 264 Warnings: 0
4	12:39:18	UPDATE customer_churn SET couponused = (SELECT couponused FROM (SELECT c...	256 row(s) affected Rows matched: 256 Changed: 256 Warnings: 0
5	12:43:47	UPDATE customer_churn SET OrderCount = (SELECT OrderCount FROM (SELECT OrderC...	Error Code: 1064. You have an error in your SQL syntax; check the manual that correspo
6	12:46:17	UPDATE customer_churn SET OrderCount = (SELECT OrderCount FROM (SELECT Ord...	258 row(s) affected Rows matched: 258 Changed: 258 Warnings: 0

Handling Outliers by deleting:

DELETE FROM customer_churn WHERE WarehouseToHome > 100;

```
5708 DELETE FROM customer_churn WHERE WarehouseToHome > 100;
5709
```

Output

Action Output

#	Time	Action	Message
4	12:39:18	UPDATE customer_churn SET couponused = (SELECT couponused FROM (SELECT c...	256 row(s) affected Rows matched: 256 Changed: 256 Warnings: 0
5	12:43:47	UPDATE customer_churn SET OrderCount = (SELECT OrderCount FROM (SELECT OrderC...	Error Code: 1064. You have an error in your SQL syntax; check the man
6	12:46:17	UPDATE customer_churn SET OrderCount = (SELECT OrderCount FROM (SELECT Ord...	258 row(s) affected Rows matched: 258 Changed: 258 Warnings: 0
7	13:28:26	DELETE FROM customer_churn WHERE WarehouseToHome > 100	2 row(s) affected

Dealing with Inconsistencies:

Before updating:

```
5661 •  select * from customer_churn;
5662    -- mean
```

CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDev
50001	1	4	Mobile Phone	3	6	Debit Card	Female	3	3
50002	1	1	Phone	1	8	UPI	Male	3	4
50003	1	1	Phone	1	30	Debit Card	Male	2	4
50004	1	0	Phone	3	15	Debit Card	Male	2	4
50005	1	0	Phone	1	12	CC	Male	3	3
50006	1	0	Computer	1	22	Debit Card	Female	3	5

Need to update/replace the records where there is a Phone we have to update it as a "Mobile" in a attribute PreferredLoginDevice.

Syntax:

- update customer_churn set PreferredLoginDevice='Mobile Phone' where PreferredLoginDevice='Phone';
- update customer_churn set PreferedOrderCat='Mobile Phone' where PreferedOrderCat='Mobile';

Before replacing of PreferedOrderCat of Mobile.

```
5661 •  select * from customer_churn;
5662    -- mean
```

WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDeviceRegistered	PreferedOrderCat	SatisfactionScore	MaritalSta
6	Debit Card	Female	3	3	Laptop & Accessory	2	Single
8	UPI	Male	3	4	Mobile	3	Single
30	Debit Card	Male	2	4	Mobile	3	Single
15	Debit Card	Male	2	4	Laptop & Accessory	5	Single
12	CC	Male	3	3	Mobile	5	Single
22	Debit Card	Female	3	5	Mobile Phone	5	Single

in column name PreferredLoginDevice **Phone** is replaced with **Mobile Phone**&

in column name PreferedOrderCat '**Mobile**' is replaced with '**Mobile Phone'**

```
5709 •  update customer_churn set PreferredLoginDevice='Mobile Phone' where PreferredLoginDevice='Phone';
5710 •  update customer_churn set PreferedOrderCat='Mobile Phone' where PreferedOrderCat='Mobile';
```

Output

Action Output

#	Time	Action	Message
2	14:59:42	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
3	15:08:48	update customer_churn set PreferredLoginDevice='Mobile Phone' where PreferredLoginDevi...	1231 row(s) affected Rows matched: 1231 Changed: 1231 Warnings: 0
4	15:11:32	update customer_churn set PreferedOrderCat='Mobile Phone' where PreferedOrderCat='Mobi...	808 row(s) affected Rows matched: 808 Changed: 808 Warnings: 0

Syntax:

- update customer_churn set PreferredPaymentMode='Cash on Delivery' where PreferredPaymentMode='COD';
- update customer_churn set PreferredPaymentMode='Credit Card' where PreferredPaymentMode='CC’;

```
5711 • update customer_churn set PreferredPaymentMode='Cash on Delivery' where PreferredPaymentMode='COD';
5712 • update customer_churn set PreferredPaymentMode='Credit Card' where PreferredPaymentMode='CC';
5713
```

Output

Action Output

#	Time	Action	Message
5	15:45:48	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
6	15:53:01	update customer_churn set PreferredPaymentMode='Cash on Delivery' where PreferredPaym...	365 row(s) affected Rows matched: 365 Changed: 365 Warnings: 0
7	15:54:51	update customer_churn set PreferredPaymentMode='Credit Card' where PreferredPaymentM...	273 row(s) affected Rows matched: 273 Changed: 273 Warnings: 0

Data Transformation:

Column Renaming:

Syntax:

ALTER TABLE customer_churn

RENAME COLUMN PreferedOrderCat TO PreferredOrderCat, RENAME COLUMN HourSpendOnApp TO HoursSpentOnApp;

```
5713 -- Data Transformation
5714 • ALTER TABLE customer_churn
5715 RENAME COLUMN PreferedOrderCat TO PreferredOrderCat, RENAME COLUMN HourSpendOnApp TO HoursSpentOnApp;
5716
```

Context Help

Output

Action Output

#	Time	Action	Message
7	15:54:51	update customer_churn set PreferredPaymentMode='Credit Card' where PreferredPaymentM...	273 row(s) affected Rows matched: 273 Changed: 273 Warnings: 0
8	16:57:21	alter table customer_churn rename column "PreferedOrderCat" to "PreferredOrderCat",renam...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds ...
9	16:58:48	ALTER TABLE customer_churn RENAME COLUMN PreferedOrderCat TO PreferredOrderCa...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

```
5661 • select * from customer_churn;
5662 -- mean
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HoursSpentOnApp	NumberOfDe
50001	1	4	Mobile Phone	3	6	Debit Card	Female	3	3
50002	1	1	Mobile Phone	1	8	UPI	Male	3	4
50003	1	1	Mobile Phone	1	30	Debit Card	Male	2	4
50004	1	0	Mobile Phone	3	15	Debit Card	Male	2	4

customer_churn 3 | Apply | Revert

Output

Action Output

#	Time	Action	Message
8	16:57:21	alter table customer_churn rename column "PreferedOrderCat" to "PreferredOrderCat",renam...	Error Code: 1064. You have an error in your SQL syntax; check the manual that co
9	16:58:48	ALTER TABLE customer_churn RENAME COLUMN PreferedOrderCat TO PreferredOrderCa...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
10	17:01:01	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned

Can observe HourSpentOnApp as HoursSSpentOnApp,similarly "PreferedOrderCat" to "PreferredOrderCat".

Creating New Columns:

Added new column:

Using alter because alter is used for renaming or adding the column header.

Syntax:

alter table customer_churn add column ‘ComplaintReceived’ varchar(3);

```
5716 • alter table customer_churn add column ‘ComplaintReceived’ varchar(3);
```

Output

Action Output

#	Time	Action	Message
9	16:58:48	ALTER TABLE customer_churn RENAME COLUMN PreferedOrderCat TO PreferredOrderCa...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
10	17:01:01	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
11	18:41:57	alter table customer_churn add column 'ComplaintReceived' varchar(3)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

This creates a new column to store ‘YES’ ,’No’ in new column.

Now, updating new column based on attribute complain, here I am using update because update editing the cell inside the column

- **Syntax: Create ComplaintReceived column**

ALTER TABLE customer_churn ADD ComplaintReceived VARCHAR(3);

UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No');

describe customer_churn;

shows the structure of the table

```
5718   -- Creating New Columns
5719 • describe customer_churn;
5720 • ALTER TABLE customer_churn ADD ComplaintReceived VARCHAR(3);
5721 • UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No');
```

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

Field	Type	Null	Key	Default	Extra
CouponUsed	int	YES		NULL	
OrderCount	int	YES		NULL	
DaySinceLastOrder	int	YES		NULL	
CashbackAmount	int	YES		NULL	
ComplaintReceived	varchar(3)	YES		NULL	

Result Grid

Form Editor

Result 4 Read Only

Output

Action Output

#	Time	Action	Message
23	19:51:25	ALTER TABLE customer_churn ADD ComplaintReceived VARCHAR(3)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
24	19:51:25	UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No')	5628 row(s) affected Rows matched: 5628 Changed: 5628 Warnings: 0
25	19:51:35	describe customer_churn	21 row(s) returned

```
5722 • ALTER TABLE customer_churn ADD ComplaintReceived VARCHAR(3);
5723 • UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No');
5724
5725
5726
```

Output

Action Output

#	Time	Action	Message
24	20:16:49	alter table customer_churn drop column ComplaintReceived	Error Code: 1091. Can't DROP 'ComplaintReceived'; check that column/key e
25	20:17:26	ALTER TABLE customer_churn ADD ComplaintReceived VARCHAR(3)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
26	20:17:26	UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No')	5628 row(s) affected Rows matched: 5628 Changed: 5628 Warnings: 0

Select * from customer_churn;

It shows all the records and attributes of a table.

```
5722 • select * from customer_churn;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

s	NumberOfAddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount	ComplaintReceived
9		1	11	1	1	5	160	Yes
7		1	15	0	1	0	121	Yes
6		1	14	0	1	3	120	Yes
8		0	23	0	1	3	134	No
3		0	11	1	1	3	130	No
2		1	22	4	6	7	139	Yes

customer_churn 5 | Apply | Revert

Output

Action Output

#	Time	Action	Message
24	19:51:25	UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No')	5628 row(s) affected Rows matched: 5628 Changed: 5628 Warnings: 0
25	19:51:35	describe customer_churn	21 row(s) returned
26	19:54:21	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned

- **Syntax: Create ChurnStatus column**
  ALTER TABLE customer_churn ADD ChurnStatus VARCHAR(10);

UPDATE customer_churn SET ChurnStatus = IF(Churn = 1, 'Churned', 'Active');

```
5723 • ALTER TABLE customer_churn ADD ChurnStatus VARCHAR(10);
5724 • UPDATE customer_churn SET ChurnStatus = IF(Churn = 1, 'Churned', 'Active');
5725
5726
5727
5728
```

Output

Action Output

#	Time	Action	Message
26	19:54:21	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
27	19:59:56	ALTER TABLE customer_churn ADD ChurnStatus VARCHAR(10)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
28	19:59:56	UPDATE customer_churn SET ChurnStatus = IF(Churn = 1, 'Churned', 'Active')	5628 row(s) affected Rows matched: 5628 Changed: 5628 Warnings: 0

Output:

```
-- Creating New Columns
describe customer_churn;
ALTER TABLE customer_churn ADD ComplaintReceived VARCHAR(3);
UPDATE customer_churn SET ComplaintReceived = IF(Complain = 1, 'Yes', 'No');
select * from customer_churn;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

ddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount	ComplaintReceived	ChurnStatus
	0	16	1	11	9	300	No	Active
	0	16	1	1	3	123	No	Active
	1	13	1	1	1	159	Yes	Churned
	0	14	0	2	2	121	No	Active
	0	23	1	11	8	268	No	Active
	1	12	0	1	7	136	Yes	Active

customer_churn 6 | Apply | Revert | Cont

Output

Action Output

#	Time	Action	Message
27	19:59:56	ALTER TABLE customer_churn ADD ChurnStatus VARCHAR(10)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
28	19:59:56	UPDATE customer_churn SET ChurnStatus = IF(Churn = 1, 'Churned', 'Active')	5628 row(s) affected Rows matched: 5628 Changed: 5628 Warnings: 0
29	20:01:08	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned

Column Dropping:

Syntax:

alter table customer _churn drop column Churn, drop column Complain;

```
-- column droping
alter table customer_churn drop column Churn, drop column Complain;
```

Cont

Output

Action Output

#	Time	Action	Message
29	20:01:08	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
30	20:06:48	alter table customer _churn drop column Churn, drop column Complain	Error Code: 1064. You have an error in your SQL syntax; check the manual that co
31	20:07:22	alter table customer_churn drop column Churn, drop column Complain	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Data Exploration and Analysis:

- **Count of churned and active customers**
- Syntax:

SELECT ChurnStatus, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY ChurnStatus;

```
-- Data Exploration and Analysis
SELECT ChurnStatus, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY ChurnStatus;
```

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

ChurnStatus	CustomerCount
Churned	948
Active	4680

Result 7 × Read Only

Output

Action Output

#	Time	Action	Message
30	20:06:48	alter table customer _churn drop column Churn, drop column Complain	Error Code: 1064. You have an error in your SQL syntax; check the manu
31	20:07:22	alter table customer_churn drop column Churn, drop column Complain	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
32	20:09:54	SELECT ChurnStatus, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY Ch...	2 row(s) returned

- **Average tenure & total cashback of churned customers**
- Syntax:

SELECT ROUND(AVG(Tenure), 2) AS AvgTenure, SUM(CashbackAmount) AS TotalCashback

FROM customer_churn WHERE ChurnStatus = 'Churned';

AvgTenure	TotalCashback
3.18	152030

```
-- Average tenure & total cashback of churned customers
SELECT ROUND(AVG(Tenure), 2) AS AvgTenure, SUM(CashbackAmount) AS TotalCashback
FROM customer_churn WHERE ChurnStatus = 'Churned';
```

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

AvgTenure	TotalCashback
3.18	152030

Result 8 × Read Only

Output

Action Output

#	Time	Action	Message
31	20:07:22	alter table customer_churn drop column Churn, drop column Complain	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
32	20:09:54	SELECT ChurnStatus, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY Ch...	2 row(s) returned
33	20:17:00	SELECT ROUND(AVG(Tenure), 2) AS AvgTenure, SUM(CashbackAmount) AS TotalCashba...	1 row(s) returned

- **% of churned customers who complained**

Syntax:

SELECT

ROUND((SUM(CASE WHEN ComplaintReceived = 'Yes' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 2) AS PercentComplained FROM customer_churn WHERE ChurnStatus = 'Churned';

PercentComplained
53.59

```
-- % of churned customers who complained
select * from customer_churn;
SELECT
  ROUND((SUM(CASE WHEN ComplaintReceived = 'Yes' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 2) AS PercentComplained
FROM customer_churn WHERE ChurnStatus = 'Churned';
```

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

PercentComplained
53.59

Result 10

Output

Action Output

#	Time	Action	Message
33	20:17:00	SELECT ROUND(AVG(Tenure), 2) AS AvgTenure, SUM(CashbackAmount) AS TotalCashba...	1 row(s) returned
34	20:25:05	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
35	20:29:48	SELECT ROUND((SUM(CASE WHEN ComplaintReceived = 'Yes' THEN 1 ELSE 0 END) ...	1 row(s) returned

- **City tier with most churned Laptop & Accessory customers**

Syntax:

SELECT CityTier, COUNT(*) AS Count

FROM customer_churn

WHERE ChurnStatus = 'Churned' AND PreferredOrderCat = 'Laptop & Accessory'

GROUP BY CityTier ORDER BY Count DESC LIMIT 1;

Output:

CityTier	Count
3	150

```
-- City tier with most churned Laptop & Accessory customers
SELECT CityTier, COUNT(*) AS Count
FROM customer_churn
WHERE ChurnStatus = 'Churned' AND PreferredOrderCat = 'Laptop & Accessory'
GROUP BY CityTier ORDER BY Count DESC LIMIT 1;
```

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

CityTier	Count
3	150

Result 11

Output

Action Output

#	Time	Action	Message
34	20:25:05	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
35	20:29:48	SELECT ROUND((SUM(CASE WHEN ComplaintReceived = 'Yes' THEN 1 ELSE 0 END) ...	1 row(s) returned
36	20:34:18	SELECT CityTier, COUNT(*) AS Count FROM customer_churn WHERE ChurnStatus = 'Chur...	1 row(s) returned

- **Most preferred payment mode among active customers**

Syntax:

SELECT PreferredPaymentMode, COUNT(*) AS Count FROM customer_churn WHERE ChurnStatus = 'Active'

GROUP BY PreferredPaymentMode ORDER BY Count DESC LIMIT 1;

	PreferredPaymentMode	Count
▶	Debit Card	1956

```
-- Most preferred payment mode among active customers
SELECT PreferredPaymentMode, COUNT(*) AS Count FROM customer_churn WHERE ChurnStatus = 'Active'
GROUP BY PreferredPaymentMode ORDER BY Count DESC LIMIT 1;
```

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

	PreferredPaymentMode	Count
▶	Debit Card	1956

Result 13 ×

Output

Action Output

#	Time	Action	Message
36	20:34:18	SELECT CityTier, COUNT(*) AS Count FROM customer_churn WHERE ChurnStatus = 'Chur...	1 row(s) returned
37	21:23:23	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
38	21:42:26	SELECT PreferredPaymentMode, COUNT(*) AS Count FROM customer_churn WHERE Chu...	1 row(s) returned

- **Total order amount hike for single customers preferring mobile phone**

Syntax:

SELECT SUM(OrderAmountHikeFromlastYear) AS TotalHike

FROM customer_churn

WHERE MaritalStatus = 'Single' AND PreferredOrderCat = 'Mobile Phone';

Output:

	TotalHike
▶	12177

```
-- Total order amount hike for single customers preferring mobile phone
SELECT SUM(OrderAmountHikeFromlastYear) AS TotalHike
FROM customer_churn
WHERE MaritalStatus = 'Single' AND PreferredOrderCat = 'Mobile Phone';
```

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

TotalHike
12177

Result 14 × Read Only

Output

Action Output

#	Time	Action	Message
37	21:23:23	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
38	21:42:26	SELECT PreferredPaymentMode, COUNT(*) AS Count FROM customer_churn WHERE Chu...	1 row(s) returned
39	21:46:19	SELECT SUM(OrderAmountHikeFromlastYear) AS TotalHike FROM customer_churn WHER...	1 row(s) returned

- **Avg devices registered among UPI users**

Syntax:

SELECT ROUND(AVG(NumberOfDeviceRegistered), 2) AS AvgDevices

FROM customer_churn WHERE PreferredPaymentMode = 'UPI';

Output:

AvgDevices
3.72

```
-- Avg devices registered among UPI users
SELECT ROUND(AVG(NumberOfDeviceRegistered), 2) AS AvgDevices
FROM customer_churn WHERE PreferredPaymentMode = 'UPI';
```

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

AvgDevices
3.72

Result 15 × Read Only Cor

Output

Action Output

#	Time	Action	Message
38	21:42:26	SELECT PreferredPaymentMode, COUNT(*) AS Count FROM customer_churn WHERE Chu...	1 row(s) returned
39	21:46:19	SELECT SUM(OrderAmountHikeFromlastYear) AS TotalHike FROM customer_churn WHER...	1 row(s) returned
40	21:52:16	SELECT ROUND(AVG(NumberOfDeviceRegistered), 2) AS AvgDevices FROM customer_c...	1 row(s) returned

- **City tier with the highest number of customers**

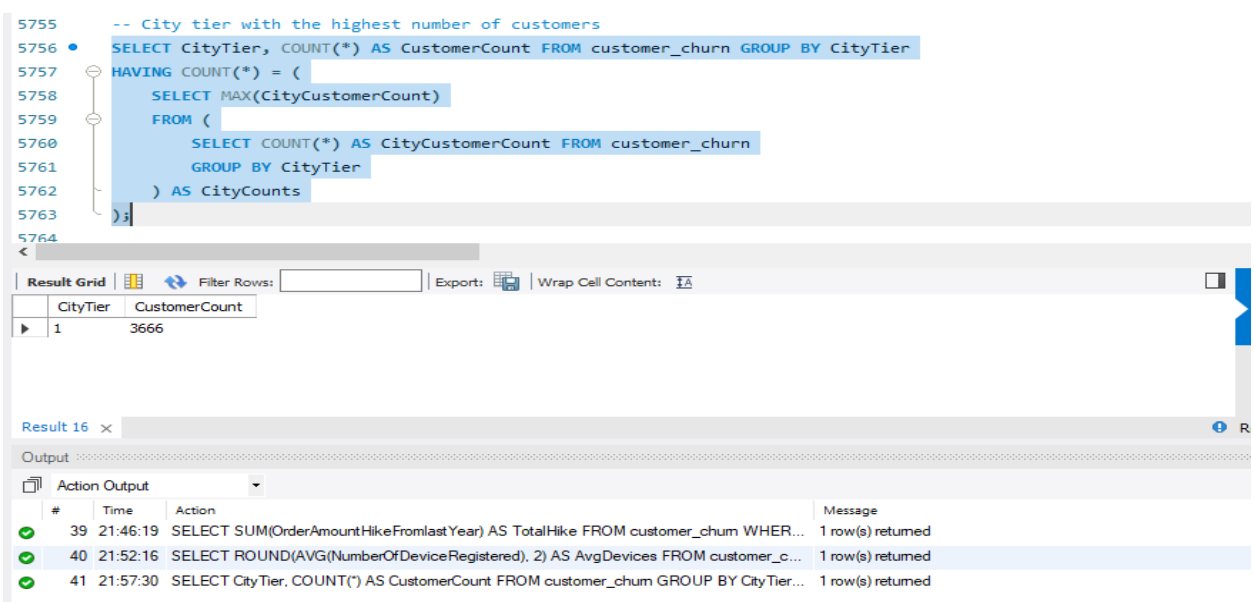
Syntax:

```
SELECT CityTier, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY CityTier
HAVING COUNT(*) = (
   SELECT MAX(CityCustomerCount)
   FROM (
      SELECT COUNT(*) AS CityCustomerCount FROM customer_churn
      GROUP BY CityTier
   ) AS CityCounts
);
```

Output:

	CityTier	CustomerCount
▶	1	3666

GROUP BY CityTier: counts how many customers are in each city tier.The inner subquery gets the **maximum count** using MAX(). HAVING filters to return the **CityTier(s)** with that max count.



- **Gender that used highest number of coupons:**

Syntax:

SELECT Gender, SUM(CouponUsed) AS TotalCouponsUsed FROM customer_churn GROUP BY Gender

HAVING SUM(CouponUsed) = (

SELECT MAX(GenderCouponSum)

FROM (

SELECT SUM(CouponUsed) AS GenderCouponSum FROM customer_churn

GROUP BY Gender

) AS GenderSums

);

Gender	TotalCouponsUsed
Male	5629

```
--
SELECT Gender, SUM(CouponUsed) AS TotalCouponsUsed FROM customer_churn  GROUP BY Gender
HAVING SUM(CouponUsed) = (
    SELECT MAX(GenderCouponSum)
    FROM (
        SELECT SUM(CouponUsed) AS GenderCouponSum FROM customer_churn
        GROUP BY Gender
    ) AS GenderSums
);
```

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

Gender	TotalCouponsUsed
Male	5629

Result Grid

Result 17 | Read Only

Output

Action Output

#	Time	Action	Message
40	21:52:16	SELECT ROUND(AVG(NumberOfDeviceRegistered), 2) AS AvgDevices FROM customer_c...	1 row(s) returned
41	21:57:30	SELECT CityTier, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY CityTier...	1 row(s) returned
42	22:02:43	SELECT Gender, SUM(CouponUsed) AS TotalCouponsUsed FROM customer_churn GRO...	1 row(s) returned

- **No. of customers & max hours spent per order category**

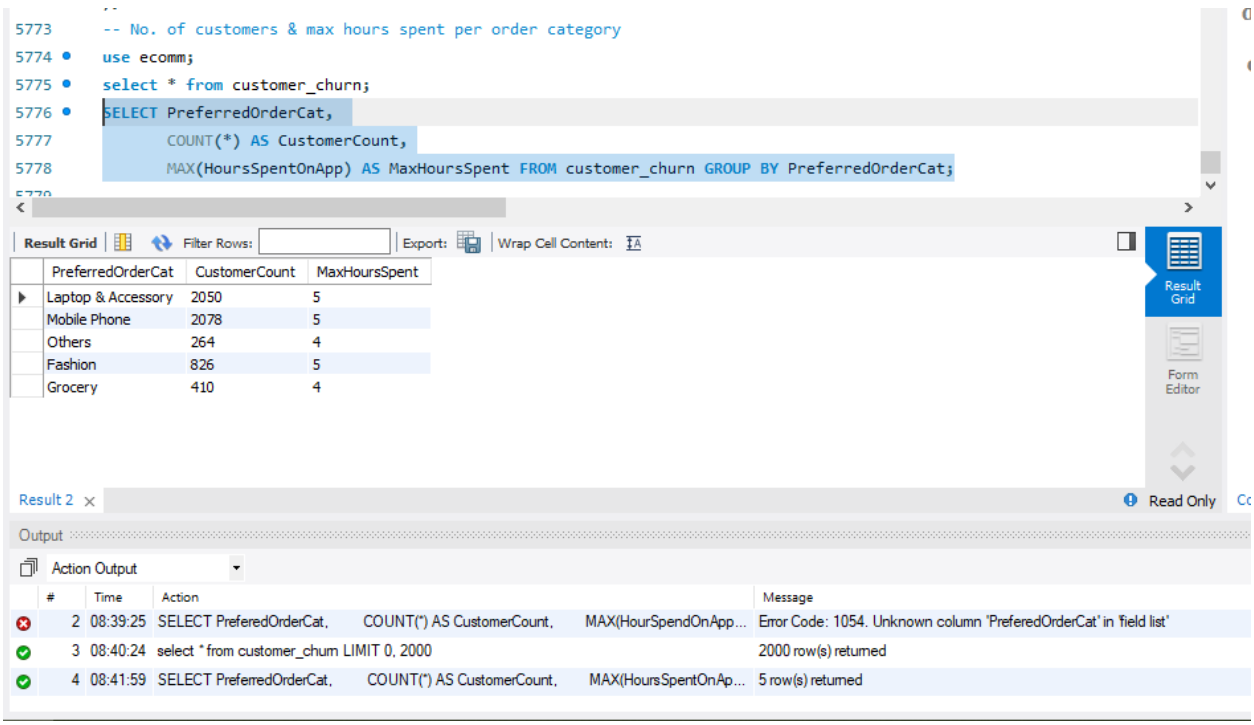
Syntax:

SELECT PreferredOrderCat,

COUNT(*) AS CustomerCount,

MAX(HoursSpentOnApp) AS MaxHoursSpent FROM customer_churn GROUP BY PreferredOrderCat;

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



Explanation:

- COUNT(*) → Total number of customers in each preferred order category.
- MAX(HourSpendOnApp) → Highest time any customer spent on the app in that category.
- GROUP BY PreferedOrderCat → Groups the results per product category.

- **Total order count for customers using credit cards with max satisfaction**

Syntax:

SELECT SUM(OrderCount) AS TotalOrders FROM customer_churn

WHERE PreferredPaymentMode = 'Credit Card' AND SatisfactionScore = 5;

Output:

	TotalOrders
▶	1122

```
-- Total order count for customers using credit cards with max satisfaction
SELECT SUM(OrderCount) AS TotalOrders FROM customer_churn
WHERE PreferredPaymentMode = 'Credit Card' AND SatisfactionScore = 5;
```

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

	TotalOrders
▶	1122

Result 3 × Read Only

Output

Action Output

#	Time	Action	Message
3	08:40:24	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
4	08:41:59	SELECT PreferredOrderCat, COUNT(*) AS CustomerCount, MAX(HoursSpentOnAp...	5 row(s) returned
5	08:50:47	SELECT SUM(OrderCount) AS TotalOrders FROM customer_churn WHERE PreferredPaym...	1 row(s) returned

- **Avg satisfaction of customers who complained**

Syntax:

SELECT ROUND(AVG(SatisfactionScore), 2) AS AvgSatisfaction FROM customer_churn
WHERE ComplaintReceived = 'Yes';

Output:

	AvgSatisfaction
▶	3.00

```
-- Avg satisfaction of customers who complained
SELECT ROUND(AVG(SatisfactionScore), 2) AS AvgSatisfaction FROM customer_churn WHERE ComplaintReceived = 'Yes';
```

AvgSatisfaction
3.00

Result 5 — Read Only

Output — Action Output

#	Time	Action	Message
5	08:50:47	SELECT SUM(OrderCount) AS TotalOrders FROM customer_churn WHERE PreferredPaym...	1 row(s) returned
6	08:56:20	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
7	08:57:34	SELECT ROUND(AVG(SatisfactionScore), 2) AS AvgSatisfaction FROM customer_churn W...	1 row(s) returned

- **Preferred order category for customers using > 5 coupons**

Syntax:

SELECT PreferredOrderCat, COUNT(*) AS Count

FROM customer_churn WHERE CouponUsed > 5 GROUP BY PreferredOrderCat ORDER BY Count DESC;

Output:

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

Result 7

```
-- Preferred order category for customers using > 5 coupons
SELECT PreferredOrderCat, COUNT(*) AS Count
FROM customer_churn WHERE CouponUsed > 5 GROUP BY PreferredOrderCat ORDER BY Count DESC;
```

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

Result 7 — Read Only

Output — Action Output

#	Time	Action	Message
7	08:57:34	SELECT ROUND(AVG(SatisfactionScore), 2) AS AvgSatisfaction FROM customer_churn W...	1 row(s) returned
8	09:01:42	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
9	09:08:24	SELECT PreferredOrderCat, COUNT(*) AS Count FROM customer_churn WHERE CouponU...	5 row(s) returned

Explanation:

1. SELECT PreferredOrderCat, COUNT(*) AS Count

- You're selecting each unique PreferredOrderCat (like "Mobile", "Laptop & Accessory", etc.).
- COUNT(*) AS Count counts how many customers fall into each category after filtering (you rename this count column as Count).

2. FROM customer_churn

- This is the source table you're querying.

3. WHERE CouponUsed > 5

- This filters the data before any grouping happens.
- Only rows where a customer used more than 5 coupons will be considered.

4. GROUP BY PreferredOrderCat

- After filtering, the remaining rows are grouped by PreferredOrderCat.
- So now each group represents a unique category, like:
    - Mobile
    - Grocery
    - Laptop & Accessory
    - etc.

5. ORDER BY Count DESC

- The result is sorted based on the Count (number of customers in each category) in descending order.
- This means the most preferred order category among customers who used more than 5 coupons will appear first.

➢ **Top 3 categories by avg cashback**

Syntax:

SELECT PreferredOrderCat, ROUND(AVG(CashbackAmount), 2) AS AvgCashback

FROM customer_churn GROUP BY PreferredOrderCat ORDER BY AvgCashback DESC LIMIT 3;

Output:

PreferredOrderCat	AvgCashback
Others	304.45
Grocery	266.24
Fashion	210.40

```
-- Top 3 categories by avg cashback
SELECT PreferredOrderCat, ROUND(AVG(CashbackAmount), 2) AS AvgCashback
FROM customer_churn GROUP BY PreferredOrderCat ORDER BY AvgCashback DESC LIMIT 3;
```

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

PreferredOrderCat	AvgCashback
Others	304.45
Grocery	266.24
Fashion	210.40

Result 9 × Read Only

Output

Action Output

#	Time	Action	Message
10	09:18:35	SELECT PreferedOrderCat, AVG(CashbackAmount) AS AvgCashback FROM customer_chu...	Error Code: 1054. Unknown column 'PreferedOrderCat' in 'field list'
11	09:19:08	SELECT PreferredOrderCat, AVG(CashbackAmount) AS AvgCashback FROM customer_ch...	0 row(s) returned
12	09:20:09	SELECT PreferredOrderCat, ROUND(AVG(CashbackAmount), 2) AS AvgCashback FROM c...	3 row(s) returned

- **Payment modes of customers with avg tenure = 10 & >500 orders**

SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderCount > 500;

Output:

```
-- Payment modes of customers with avg tenure = 10 & >500 orders
SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderCount > 500;
```

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

PreferredPaymentMode

customer_churn 10 × Read Only

Output

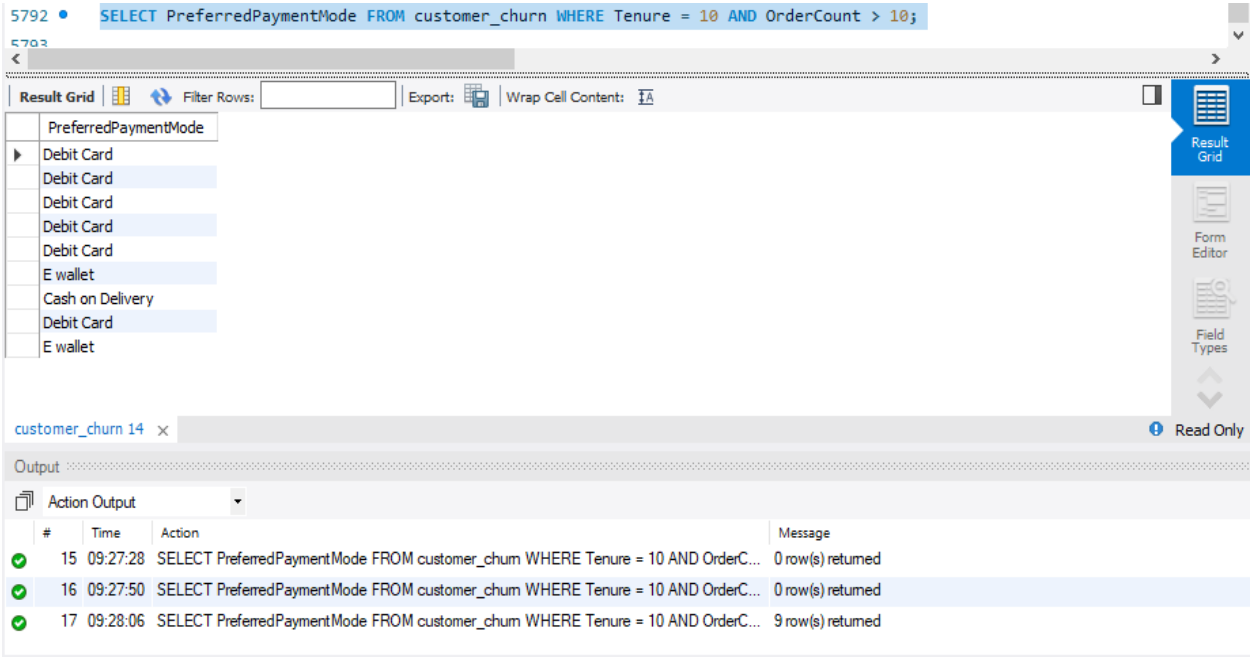
Action Output

#	Time	Action	Message
11	09:19:08	SELECT PreferredOrderCat, AVG(CashbackAmount) AS AvgCashback FROM customer_ch...	0 row(s) returned
12	09:20:09	SELECT PreferredOrderCat, ROUND(AVG(CashbackAmount), 2) AS AvgCashback FROM c...	3 row(s) returned
13	09:25:18	SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderC...	0 row(s) returned

There no records of >500.

SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderCount > 10;

Output:



its working for 10.

- **Categorize customers by distance & churn status:**

Syntax:

```
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 DistanceCategory, ChurnStatus,
  COUNT(*) AS Count FROM customer_churn GROUP BY DistanceCategory, ChurnStatus;
```

Output:

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

DistanceCategory	ChurnStatus	Count
Close Distance	Churned	265
Far Distance	Churned	498
Moderate Distance	Churned	184
Close Distance	Active	1696
Moderate Distance	Active	1106
Far Distance	Active	1871
Very Close Distance	Active	7
Very Close Distance	Churned	1

```
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 DistanceCategory, ChurnStatus,
  COUNT(*) AS Count FROM customer_churn GROUP BY DistanceCategory, ChurnStatus;
```

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

DistanceCategory	ChurnStatus	Count
Close Distance	Churned	265
Far Distance	Churned	498
Moderate Distance	Churned	184
Close Distance	Active	1696
Moderate Distance	Active	1106
Far Distance	Active	1871
Very Close Distance	Active	7
Very Close Distance	Churned	1

Result Grid | Form Editor | Result 1 | Read Only

Output

Action Output

#	Time	Action	Message
1	15:25:13	SELECT CASE WHEN WarehouseToHome <= 5 THEN 'Very Close Distance' WHEN ...	Error Code: 1046. No database selected Select the default DB to be used by do
2	15:25:44	use ecomm	0 row(s) affected
3	15:25:55	SELECT CASE WHEN WarehouseToHome <= 5 THEN 'Very Close Distance' WHEN ...	8 row(s) returned

Explanation:

CASE Statement:

- Used to create a new category called DistanceCategory based on WarehouseToHome.
  - <= 5 → Very Close Distance
  - <= 10 → Close Distance
  - <= 15 → Moderate Distance
  - > 15 → Far Distance

ChurnStatus:

- Grouping also by ChurnStatus to show how many customers churned or stayed in each distance range.

COUNT(*):

counting the number of customers in each combination of:

- Distance category (DistanceCategory)
- Churn status (ChurnStatus)

This tells you **how many churned or stayed active** in each distance range.

GROUP BY:

- Needed to get separate counts for each distance category and churn status combination. group the data **by the two fields you're selecting**:
- The calculated DistanceCategory
- The existing ChurnStatus

ORDER BY:

- Ensures the result is sorted neatly by distance category and then by churn status.

➢ **Order details of married customers in Tier-1 with more than average order count**

Syntax:

```
SELECT * FROM customer_churn
WHERE MaritalStatus = 'Married' AND CityTier = 1
  AND OrderCount > (
    SELECT AVG(OrderCount)
    FROM customer_churn);
```

Output:

```
-- Order details of married customers in Tier-1 with more than average order count
SELECT * FROM customer_churn
WHERE MaritalStatus = 'Married' AND CityTier = 1
  AND OrderCount > (
      SELECT AVG(OrderCount)
      FROM customer_churn);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HoursSpentOnApp	NumberOfDeviceRegist
50049	3	Computer	1	15	Credit Card	Male	3	3
50119	13	Mobile Phone	1	30	Cash on Delivery	Male	3	3
50132	15	Mobile Phone	1	16	Credit Card	Female	2	4
50133	13	Mobile Phone	1	8	Credit Card	Male	2	3
50152	2	Mobile Phone	1	28	Debit Card	Female	3	2
50180	12	Mobile Phone	1	6	Credit Card	Male	2	2
50187	18	Computer	1	13	Debit Card	Male	3	3
50229	10	Mobile Phone	1	12	Credit Card	Male	2	2
50262	10	Mobile Phone	1	20	Debit Card	Female	3	3
50308	7	Mobile Phone	1	8	Debit Card	Male	3	4

customer_churn 2 | Apply | Revert

Output

Action Output

#	Time	Action	Message
2	15:25:44	use ecomm	0 row(s) affected
3	15:25:55	SELECT CASE WHEN WarehouseToHome <= 5 THEN 'Very Close Distance' WHEN...	8 row(s) returned
4	15:42:07	SELECT * FROM customer_churn WHERE MaritalStatus = 'Married' AND CityTier = 1 AND ...	576 row(s) returned

Explanation:

- MaritalStatus = 'Married': Filters only married customers.
- CityTier = 1: Filters only customers who live in **City Tier-1**.
- OrderCount > (SELECT AVG(OrderCount)...): Compares each customer's order count to the **average order count** of all customers.

➢ A)Create return table:

Syntax:

```
CREATE TABLE customer_returns (
  ReturnID INT PRIMARY KEY,
  CustomerID INT,
  ReturnDate DATE,
  RefundAmount INT
);
```

```sql
CREATE TABLE customer_returns (
  ReturnID INT PRIMARY KEY,
  CustomerID INT,
  ReturnDate DATE,
  RefundAmount INT
);
```

Output

Action Output

#	Time	Action	Message
3	15:25:55	SELECT CASE WHEN WarehouseToHome <= 5 THEN 'Very Close Distance' WHEN...	8 row(s) returned
4	15:42:07	SELECT * FROM customer_churn WHERE MaritalStatus = 'Married' AND CityTier = 1 AND ...	576 row(s) returned
5	15:46:59	CREATE TABLE customer_returns (ReturnID INT PRIMARY KEY, CustomerID INT, Ret...	0 row(s) affected

- Insert table:

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);

Output:

```sql
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);
```

Output

Action Output

#	Time	Action	Message
4	15:42:07	SELECT * FROM customer_churn WHERE MaritalStatus = 'Married' AND CityTier = 1 AND ...	576 row(s) returned
5	15:46:59	CREATE TABLE customer_returns (ReturnID INT PRIMARY KEY, CustomerID INT, Ret...	0 row(s) affected
6	15:48:59	INSERT INTO customer_returns (ReturnID, CustomerID, ReturnDate, RefundAmount) VALU...	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0

b) Join with customers who churned and complained:

syntax:

SELECT cr.*, cc.*

FROM customer_returns cr

JOIN customer_churn cc ON cr.CustomerID = cc.CustomerID

WHERE cc.ChurnStatus = 'Churned' AND cc.ComplaintReceived = 'Yes';

Explanation:

- cr.* → Selects **all columns** from the customer_returns table.
- cc.* → Selects **all columns** from the customer_churn table.
- So together, this returns **all columns from both tables**.

Output:

```
5824 •  SELECT cr.*, cc.*
5825    FROM customer_returns cr
5826    JOIN customer_churn cc ON cr.CustomerID = cc.CustomerID
5827    WHERE cc.ChurnStatus = 'Churned' AND cc.ComplaintReceived = 'Yes';
5828
```

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

ReturnID	CustomerID	ReturnDate	RefundAmount	CustomerID	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode
1002	50316	2023-01-23	2000	50316	0	Computer	2	29	UPI
1004	52321	2023-03-08	2510	52321	18	Mobile Phone	3	19	E wallet
1006	53749	2023-04-17	1740	53749	1	Mobile Phone	3	31	Credit Card

Result Grid | Form Editor

Result 3 | Read Only

Output

Action Output

#	Time	Action	Message
5	15:46:59	CREATE TABLE customer_returns (ReturnID INT PRIMARY KEY, CustomerID INT, Ret...	0 row(s) affected
6	15:48:59	INSERT INTO customer_returns (ReturnID, CustomerID, ReturnDate, RefundAmount) VALU...	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0
7	15:55:53	SELECT cr.*, cc.* FROM customer_returns cr JOIN customer_churn cc ON cr.CustomerID = ...	3 row(s) returned