

## E-Commerce Customer Churn Analysis

### Dataset Download:

[https://drive.google.com/uc?export=download&id=1iKKCze\\_Fpk2n\\_g3BIZBiSjcDFdFcEn3D](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,  
  PreferredOrderCat    VARCHAR(20),  
  SatisfactionScore    INT,  
  MaritalStatus        VARCHAR(10),  
  NumberOfAddress      INT,  
  Complain             BIT,  
  OrderAmountHikeFromlastYear INT,  
  CouponUsed           INT,  
  OrderCount           INT,
```

DaySinceLastOrder INT,

CashbackAmount INT

);

```
1 DROP DATABASE IF EXISTS ecomm;
2 CREATE DATABASE ecomm;
3 USE ecomm;
4
5 CREATE TABLE customer_churn(
6     CustomerID INT PRIMARY KEY,
7     Churn BIT,
8     Tenure INT,
9     PreferredLoginDevice VARCHAR(20),
10    CityTier INT,
11    WarehouseToHome INT,
12    PreferredPaymentMode VARCHAR(20),
13    Gender ENUM('Male','Female'),
14    HourSpendOnApp INT,
15    NumberOfDeviceRegistered INT,
16    PreferredOrderCat VARCHAR(20),
17    SatisfactionScore INT,
18    MaritalStatus VARCHAR(10);
19
```

Output

#	Time	Action	Message
1	12:15:54	DROP DATABASE IF EXISTS ecomm	0 row(s) affected. 1 warning(s): 1008 Can't drop database 'ecomm': database
2	12:16:04	CREATE DATABASE ecomm	1 row(s) affected
3	12:16:08	USE ecomm	0 row(s) affected
4	12:16:34	CREATE TABLE customer_churn( CustomerID INT PRIMARY KEY, Churn ...	0 row(s) affected

Schemas

Filter objects

- actor
- da
- data\_analytics
- ecomm
  - Tables
    - customer\_churn
      - Columns
      - Indexes
      - PRIMARY
      - Foreign Keys
      - Triggers
      - Views
      - Stored Procedures
    - Administration
    - Schemas

Schema: ecomm

Automatic context help is disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.

Output

#	Time	Action	Message	Duration / Fetch
6	12:26:21	CREATE DATABASE ecomm	1 row(s) affected	0.016 sec
7	12:26:21	USE ecomm	0 row(s) affected	0.000 sec
8	12:26:21	CREATE TABLE customer_churn( CustomerID INT PRIMARY KEY, Churn ...	0 row(s) affected	0.125 sec
9	12:26:21		5630 row(s) affected Records: 5630 Duplicates: 0 Warnings: 0	1.391 sec

5659 describe customer\_churn;

5660

5661

Result Grid

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	
Churn	bit(1)	YES		NULL	
Tenure	int	YES		NULL	
PreferredLoginDevice	varchar(20)	YES		NULL	
CityTier	int	YES		NULL	
WarehouseToHome	int	YES		NULL	
PreferredPaymentMode	varchar(20)	YES		NULL	
Gender	enum('Male','Female')	YES		NULL	
HourSpendOnApp	int	YES		NULL	
NumberOfDeviceRegistered	int	YES		NULL	
PreferredOrderCat	varchar(20)	YES		NULL	
SatisfactionScore	int	YES		NULL	
MaritalStatus	varchar(10)	YES		NULL	

Result 1 x

Output

#	Time	Action	Message
1	14:50:33	USE ecomm	0 row(s) affected
2	14:52:09	describe customer_churn	20 row(s) returned

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.

```
5661 • select * from customer_churn;
5662 • UPDATE customer_churn
5663 SET WarehouseToHome = (
5664     SELECT avg_val FROM (
5665         SELECT ROUND(AVG(WarehouseToHome)) AS avg_val
5666         FROM customer_churn
5667         WHERE WarehouseToHome IS NOT NULL
5668     ) AS temp
5669 )
5670 WHERE WarehouseToHome IS NULL;
```

CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDev
50001	1	4	Mobile Phone	3	6	Debit Card	Female	3	3
50002	1	NULL	Phone	1	8	UPI	Male	3	4
50003	1	NULL	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	NULL	3

customer\_churn 2 x

Output

#	Time	Action	Message
13	18:26:23	update customer_churn set WarehouseToHome=(select round(avg(WarehouseToHome)) from ...	Error Code: 1093. You can't specify target table 'customer_churn' for update in F
14	18:28:53	UPDATE customer_churn SET WarehouseToHome = ( SELECT ROUND(AVG(Warehous...	Error Code: 1093. You can't specify target table 'customer_churn' for update in F
15	18:31:20	UPDATE customer_churn SET WarehouseToHome = ( SELECT avg_val FROM ( S...	251 row(s) affected Rows matched: 251 Changed: 251 Warnings: 0
16	18:31:36	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned

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;
```

E-Commerce Customer churn db\*

Limit to 2000 rows

```

5677 )
5678 where HourSpendOnApp is null;
5679 • update customer_churn set OrderAmountHikeFromLastYear=(select avg_val from(
5680 select Round(avg(OrderAmountHikeFromLastYear)) as avg_val
5681 from customer_churn where OrderAmountHikeFromLastYear is not null
5682 ) as temp
5683 )
5684 where OrderAmountHikeFromLastYear is null;
5685 • update customer_churn set DaySinceLastOrder=(select avg_val from(
5686 select Round(avg(DaySinceLastOrder)) as avg_val
5687 from customer_churn where DaySinceLastOrder is not null
5688 ) as temp
5689 )
5690 where DaySinceLastOrder is null;

```

Automatic disabled. U manually current ca toggle a

Output

#	Time	Action	Message
12	18:26:06	update customer_churn set WarehouseToHome= Round(avg(WarehouseToHome)) where ...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds ...
13	18:26:23	update customer_churn set WarehouseToHome=(select round(avg(WarehouseToHome)) fro...	Error Code: 1093. You can't specify target table 'customer_churn' for update in FROM clause
14	18:28:53	UPDATE customer_churn SET WarehouseToHome = ( SELECT ROUND(AVG(Warehous...	Error Code: 1093. You can't specify target table 'customer_churn' for update in FROM clause
15	18:31:20	UPDATE customer_churn SET WarehouseToHome = ( SELECT avg_val FROM ( S...	251 row(s) affected Rows matched: 251 Changed: 251 Warnings: 0
16	18:31:36	select * from customer_churn LIMIT 0, 2000	2000 row(s) returned
17	19:41:51	update customer_churn set HourSpendOnApp=(select avg_val from(select Round(avg(Hou...	255 row(s) affected Rows matched: 255 Changed: 255 Warnings: 0
18	19:44:56	update customer_churn set OrderAmountHikeFromLastYear=(select avg_val from(select Rou...	265 row(s) affected Rows matched: 265 Changed: 265 Warnings: 0
19	19:48:58	update customer_churn set DaySinceLastOrder=(select avg_val from(select Round(avg(Day...	307 row(s) affected Rows matched: 307 Changed: 307 Warnings: 0

## 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;
```

The screenshot shows a SQL script with three UPDATE statements. The first two are for 'Tenure' and 'couponused', and the third is for 'OrderCount'. The output window shows that the first two statements executed successfully, while the third one failed with an error.

```
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;
```

#	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 corresp
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;
```

The screenshot shows a SQL script with a DELETE statement. The output window shows that the statement executed successfully, deleting 2 rows.

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

#	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 PreferredOrderCat='Mobile Phone' where PreferredOrderCat='Mobile';

Before replacing of PreferredOrderCat of Mobile.

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

	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDeviceRegistered	PreferredOrderCat	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 PreferredOrderCat '**Mobile**' is replaced with '**Mobile Phone**'

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

#	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 PreferredOrderCat='Mobile Phone' where PreferredOrderCat='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 PreferredOrderCat TO PreferredOrderCat, RENAME COLUMN HourSpendOnApp TO HoursSpentOnApp;`

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

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 "PreferredOrderCat" 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 PreferredOrderCat TO PreferredOrderCa...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

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

Result Grid

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

Output

Action Output

#	Time	Action	Message
✗ 8	16:57:21	alter table customer_churn rename column "PreferredOrderCat" 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 PreferredOrderCat 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".

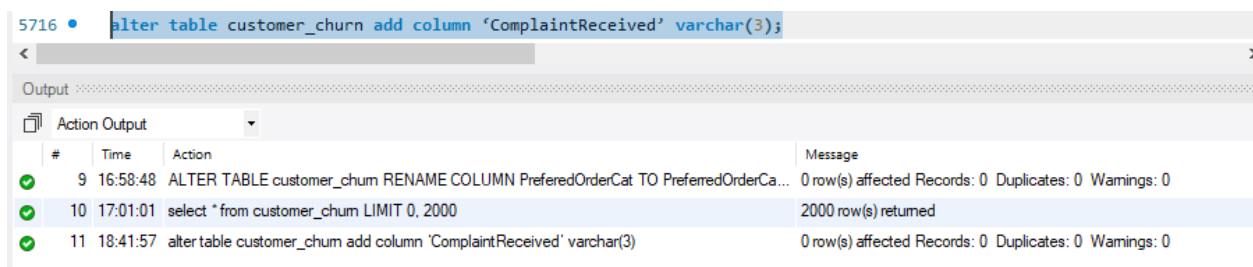
## Creating New Columns:

### 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

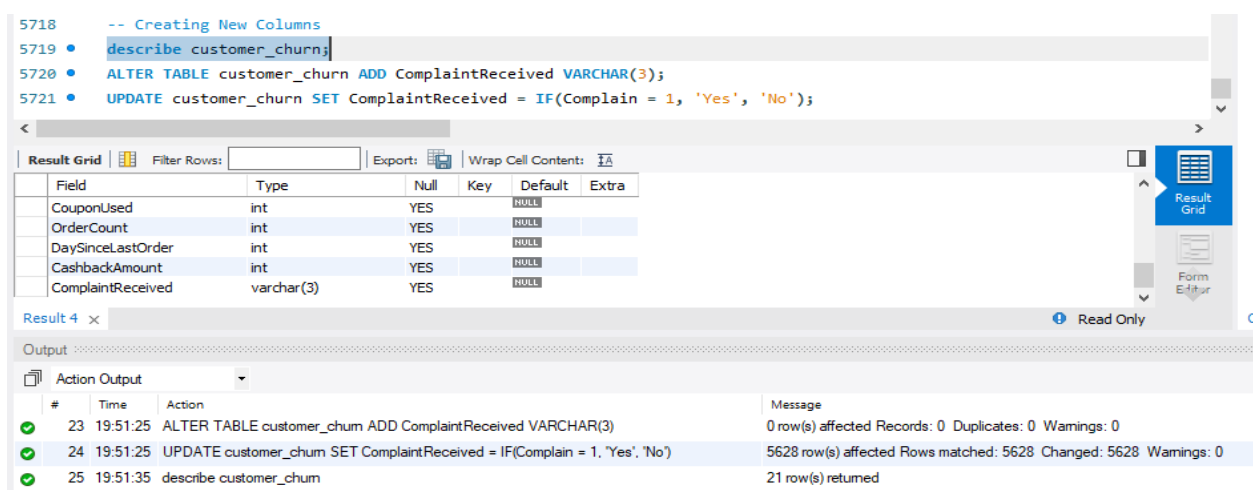
- **Syntax: Create ComplaintReceived column**

```
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

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 4 x 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

#	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

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

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');

```

5722 • select * from customer_churn;
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

#	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:

```

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');
5722 • select * from customer_churn;

```

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 x Apply Revert Cont

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;

```

5727 -- column dropping
5728 • alter table customer_churn drop column Churn, drop column Complain;
5729 |
5730
5731
5732

```

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 cc
✓ 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;
```

5729 -- Data Exploration and Analysis  
 5730 • `SELECT ChurnStatus, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY ChurnStatus;`

Result Grid

ChurnStatus	CustomerCount
Churned	948
Active	4680

Result 7 x 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 man...
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

5732 -- Average tenure & total cashback of churned customers  
 5733 • `SELECT ROUND(AVG(Tenure), 2) AS AvgTenure, SUM(CashbackAmount) AS TotalCashback`  
 5734 `FROM customer_churn WHERE ChurnStatus = 'Churned';`

Result Grid

AvgTenure	TotalCashback
3.18	152030

Result 8 x 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

```

5735 -- % of churned customers who complained
5736 • select * from customer_churn;
5737 • SELECT
5738     ROUND((SUM(CASE WHEN ComplaintReceived = 'Yes' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 2) AS PercentComplained
5739 FROM customer_churn WHERE ChurnStatus = 'Churned';
5740

```

PercentComplained
53.59

Result 10 x [Read On](#)

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

```

5740 -- City tier with most churned Laptop & Accessory customers
5741 • SELECT CityTier, COUNT(*) AS Count
5742 FROM customer_churn
5743 WHERE ChurnStatus = 'Churned' AND PreferredOrderCat = 'Laptop & Accessory'
5744 GROUP BY CityTier ORDER BY Count DESC LIMIT 1;
5745
5746

```

CityTier	Count
3	150

Result 11 x [Read Only](#)

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;
```

The screenshot shows a SQL query execution interface. At the top, a small table displays the result of the query:

PreferredPaymentMode	Count
Debit Card	1956

Below this, the SQL query is shown in a code editor:

```
5745 -- Most preferred payment mode among active customers
5746 • SELECT PreferredPaymentMode, COUNT(*) AS Count FROM customer_churn WHERE ChurnStatus = 'Active'
5747 GROUP BY PreferredPaymentMode ORDER BY Count DESC LIMIT 1;
5748
```

The interface also shows a 'Result Grid' tab with the same table as above. Below the grid, there is an 'Output' section with a table showing the execution log:

#	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

```

5748 -- Total order amount hike for single customers preferring mobile phone
5749 • SELECT SUM(OrderAmountHikeFromlastYear) AS TotalHike
5750 FROM customer_churn
5751 WHERE MaritalStatus = 'Single' AND PreferredOrderCat = 'Mobile Phone';
5752
5753

```

TotalHike
12177

Result 14 x 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

```

5752 -- Avg devices registered among UPI users
5753 • SELECT ROUND(AVG(NumberOfDeviceRegistered), 2) AS AvgDevices
5754 FROM customer_churn WHERE PreferredPaymentMode = 'UPI';
5755
5756

```

AvgDevices
3.72

Result 15 x 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.

```
5755 -- City tier with the highest number of customers
5756 • SELECT CityTier, COUNT(*) AS CustomerCount FROM customer_churn GROUP BY CityTier
5757 HAVING COUNT(*) = (
5758     SELECT MAX(CityCustomerCount)
5759     FROM (
5760         SELECT COUNT(*) AS CityCustomerCount FROM customer_churn
5761         GROUP BY CityTier
5762     ) AS CityCounts
5763 );
5764
```

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

	CityTier	CustomerCount
▶	1	3666

Result 16 x

Output

#	Time	Action	Message
39	21:46:19	SELECT SUM(OrderAmountHikeFromLastYear) AS TotalHike FROM customer_churn WHERE...	1 row(s) returned
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



➤ **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

The screenshot shows a SQL IDE interface. The main window displays a SQL query with line numbers 5764 to 5773. The query is a nested SELECT statement to find the gender with the highest total coupons used. The query is as follows:

```
--  
5765 • SELECT Gender, SUM(CouponUsed) AS TotalCouponsUsed FROM customer_churn GROUP BY Gender  
5766 HAVING SUM(CouponUsed) = (  
5767     SELECT MAX(GenderCouponSum)  
5768 FROM (  
5769     SELECT SUM(CouponUsed) AS GenderCouponSum FROM customer_churn  
5770     GROUP BY Gender  
5771 ) AS GenderSums  
5772 );  
5773
```

Below the query editor, there is a 'Result Grid' tab showing the results of the query. The grid has two columns: 'Gender' and 'TotalCouponsUsed'. The results are as follows:

Gender	TotalCouponsUsed
▶ Male	5629

At the bottom of the IDE, there is an 'Output' window showing the execution log. It contains three entries, each with a status icon, a number, a time, an action, and a message:

#	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

```
-- No. of customers & max hours spent per order category  
use ecomm;  
select * from customer_churn;  
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

Result 2 x

Output

Action Output

#	Time	Action	Message
2	08:39:25	SELECT PreferredOrderCat, COUNT(*) AS CustomerCount, MAX(HoursSpentOnApp...	Error Code: 1054. Unknown column 'PreferredOrderCat' in 'field list'
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(HoursSpentOnApp...	5 row(s) returned

**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 PreferredOrderCat → 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:**

The screenshot displays a SQL query execution environment. At the top, a query is entered: `-- Total order count for customers using credit cards with max satisfaction`, followed by `SELECT SUM(OrderCount) AS TotalOrders FROM customer_churn` and `WHERE PreferredPaymentMode = 'Credit Card' AND SatisfactionScore = 5;`. Below the query editor, a small table preview shows the result: a single row with the column `TotalOrders` and the value `1122`. The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. A 'Result 3' tab is active, showing the 'Output' section with a table of execution actions and messages. The table has columns for '#', 'Time', 'Action', and 'Message'. It lists three actions: a select statement returning 2000 rows, a select statement returning 5 rows, and the current query returning 1 row.

	TotalOrders
▶	1122

#	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

```

5781 -- Avg satisfaction of customers who complained
5782 -- SELECT ROUND(AVG(SatisfactionScore), 2) AS AvgSatisfaction FROM customer_churn WHERE ComplaintReceived = 'Yes';
5783
5784

```

Result Grid

AvgSatisfaction
3.00

Result 5 x

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

```

5784 -- Preferred order category for customers using > 5 coupons
5785 • SELECT PreferredOrderCat, COUNT(*) AS Count
5786 FROM customer_churn WHERE CouponUsed > 5 GROUP BY PreferredOrderCat ORDER BY Count DESC;
5787

```

Result Grid

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

Result 7 x

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

```
5787 -- Top 3 categories by avg cashback
5788 • SELECT PreferredOrderCat, ROUND(AVG(CashbackAmount), 2) AS AvgCashback
5789 FROM customer_churn GROUP BY PreferredOrderCat ORDER BY AvgCashback DESC LIMIT 3;
5790
```

Result 9 x Read Only

Output

Action Output

#	Time	Action	Message
✖	10 09:18:35	SELECT PreferredOrderCat, AVG(CashbackAmount) AS AvgCashback FROM customer_chu...	Error Code: 1054. Unknown column 'PreferredOrderCat' 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:

```
5790 -- Payment modes of customers with avg tenure = 10 & >500 orders
5791 • SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderCount > 500;
5792
```

customer\_churn 10 x 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_chum 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:

5792 • SELECT PreferredPaymentMode FROM customer\_churn WHERE Tenure = 10 AND OrderCount > 10;

5793

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

PreferredPaymentMode
Debit Card
Debit Card
Debit Card
Debit Card
Debit Card
E wallet
Cash on Delivery
Debit Card
E wallet

customer\_churn 14 x

Read Only

Output

Action Output

#	Time	Action	Message
✓ 15	09:27:28	SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderC...	0 row(s) returned
✓ 16	09:27:50	SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderC...	0 row(s) returned
✓ 17	09:28:06	SELECT PreferredPaymentMode FROM customer_churn WHERE Tenure = 10 AND OrderC...	9 row(s) returned

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

```
5795 • SELECT
5796 CASE
5797     WHEN WarehouseToHome <= 5 THEN 'Very Close Distance'
5798     WHEN WarehouseToHome <= 10 THEN 'Close Distance'
5799     WHEN WarehouseToHome <= 15 THEN 'Moderate Distance'
5800     ELSE 'Far Distance'
5801 END AS DistanceCategory, ChurnStatus,
5802 COUNT(*) AS Count FROM customer_churn GROUP BY DistanceCategory, ChurnStatus;
5803
```

**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 1 x | Read Only | Cont

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.
  - $\leq 5 \rightarrow$  Very Close Distance
  - $\leq 10 \rightarrow$  Close Distance
  - $\leq 15 \rightarrow$  Moderate Distance
  - $> 15 \rightarrow$  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:

```

5803  -- Order details of married customers in Tier-1 with more than average order count
5804  •  SELECT * FROM customer_churn
5805  WHERE MaritalStatus = 'Married' AND CityTier = 1
5806  AND OrderCount > (
5807  SELECT AVG(OrderCount)
5808  FROM customer_churn);
5809

```

CustomerID	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HoursSpentOnApp	NumberOfDeviceRegistrations
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 x Apply Revert Co

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
);

```

```

5809 ● CREATE TABLE customer_returns (
5810     ReturnID INT PRIMARY KEY,
5811     CustomerID INT,
5812     ReturnDate DATE,
5813     RefundAmount INT
5814 );

```

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_chum 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:

```

5815 ● INSERT INTO customer_returns (ReturnID, CustomerID, ReturnDate, RefundAmount) VALUES
5816     (1001, 50022, '2023-01-01', 2130),
5817     (1002, 50316, '2023-01-23', 2000),
5818     (1003, 51099, '2023-02-14', 2290),
5819     (1004, 52321, '2023-03-08', 2510),
5820     (1005, 52928, '2023-03-20', 3000),
5821     (1006, 53749, '2023-04-17', 1740),
5822     (1007, 54206, '2023-04-21', 3250),
5823     (1008, 54838, '2023-04-30', 1990);
5824

```

Output

#	Time	Action	Message
4	15:42:07	SELECT * FROM customer_chum 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:

The screenshot displays a database query editor interface. At the top, a SQL query is entered in a text area:

```
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
```

Below the query editor, a 'Result Grid' is shown, displaying the results of the query. The grid has the following columns: ReturnID, CustomerID, ReturnDate, RefundAmount, CustomerID, Tenure, PreferredLoginDevice, CityTier, WarehouseToHome, and PreferredPaymentMode. The results are as follows:

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

Below the result grid, an 'Output' section is visible, showing the execution log. The log contains the following entries:

#	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