Ecommerce Customer Churn Analysis

```
import pandas as pd
```

```
df = pd.read_csv('ecommerce_churn.csv')
```

```
df.head()
```

	CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDeviceRegistered	Prefere
0	50001	1	4.0	Mobile Phone	3	6.0	Debit Card	Female	3.0	3	
1	50002	1	NaN	Phone	1	8.0	UPI	Male	3.0	4	
2	50003	1	NaN	Phone	1	30.0	Debit Card	Male	2.0	4	
3	50004	1	0.0	Phone	3	15.0	Debit Card	Male	2.0	4	
4	50005	1	0.0	Phone	1	12.0	CC	Male	NaN	3	

Next steps: Generate code with df | View recommended plots | New interactive sheet

Data Cleaning

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5630 entries, 0 to 5629
Data columns (total 20 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   CustomerID                   5630 non-null   int64
 1   Churn                        5630 non-null   int64
 2   Tenure                       5366 non-null   float64
 3   PreferredLoginDevice         5630 non-null   object
 4   CityTier                     5630 non-null   int64
 5   WarehouseToHome              5379 non-null   float64
 6   PreferredPaymentMode         5630 non-null   object
 7   Gender                       5630 non-null   object
 8   HourSpendOnApp               5375 non-null   float64
 9   NumberOfDeviceRegistered     5630 non-null   int64
 10  PreferedOrderCat             5630 non-null   object
 11  SatisfactionScore            5630 non-null   int64
```

```
 12  MaritalStatus               5630 non-null   object
 13  NumberOfAddress             5630 non-null   int64
 14  Complain                    5630 non-null   int64
 15  OrderAmountHikeFromlastYear 5365 non-null   float64
 16  CouponUsed                  5374 non-null   float64
 17  OrderCount                  5372 non-null   float64
 18  DaySinceLastOrder           5323 non-null   float64
 19  CashbackAmount              5630 non-null   int64
dtypes: float64(7), int64(8), object(5)
memory usage: 879.8+ KB
```

```
cols = df.columns[df.isnull().sum()>0]

df[cols].describe()
```

	Tenure	WarehouseToHome	HourSpendOnApp	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder
count	5366.000000	5379.000000	5375.000000	5365.000000	5374.000000	5372.000000	5323.000000
mean	10.189899	15.639896	2.931535	15.707922	1.751023	3.008004	4.543491
std	8.557241	8.531475	0.721926	3.675485	1.894621	2.939680	3.654433
min	0.000000	5.000000	0.000000	11.000000	0.000000	1.000000	0.000000
25%	2.000000	9.000000	2.000000	13.000000	1.000000	1.000000	2.000000
50%	9.000000	14.000000	3.000000	15.000000	1.000000	2.000000	3.000000
75%	16.000000	20.000000	3.000000	18.000000	2.000000	3.000000	7.000000
max	61.000000	127.000000	5.000000	26.000000	16.000000	16.000000	46.000000

Outlier in *WarehouseToHome*, fix it before filling null values with column mean

```
df[df['WarehouseToHome'] > 100]
```

	CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDeviceRegistered	Prefe
1309	51310	0	25.0	Computer	3	126.0	Debit Card	Male	2.0	3	
4124	54125	0	26.0	Computer	3	127.0	Debit Card	Male	3.0	4	

```
df.loc[df['WarehouseToHome'] > 100,'WarehouseToHome'] = 26
```

```
df[df['WarehouseToHome']>100]
```

CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendOnApp	NumberOfDeviceRegistered	Prefered(

Filling null values with mean

```
for col in cols:
  df.loc[df[col].isnull(),col] = df[col].mean()
```

```
df.columns.isnull().sum()
```

```
0
```

Category Columns

```
obj_cols = df.select_dtypes(include='object').columns
for col in obj_cols:
  print(f'{col} : {df[col].unique()}')
```

```
PreferredLoginDevice : ['Mobile Phone' 'Phone' 'Computer']
PreferredPaymentMode : ['Debit Card' 'UPI' 'CC' 'Cash on Delivery' 'E wallet' 'COD' 'Credit Card']
Gender : ['Female' 'Male']
PreferedOrderCat : ['Laptop & Accessory' 'Mobile' 'Mobile Phone' 'Others' 'Fashion' 'Grocery']
MaritalStatus : ['Single' 'Divorced' 'Married']
```

Redundant values in some category columns

```
df.loc[df.PreferredLoginDevice == 'Mobile Phone','PreferredLoginDevice'] = 'Phone'
df.loc[df.PreferredPaymentMode == 'COD','PreferredPaymentMode'] = 'Cash on Delivery'
df.loc[df.PreferredPaymentMode == 'CC','PreferredPaymentMode'] = 'Credit Card'
df.loc[df.PreferedOrderCat == 'Mobile Phone','PreferedOrderCat'] = 'Mobile'
```

```
df.duplicated().sum()
```

```
0
```

Data Exploration - Churn Analysis in SQL

```
import sqlite3

# Create an in-memory SQLite database
conn = sqlite3.connect(":memory:")
```

```
# Save DataFrame to SQL table
df.to_sql("ecomm_data", conn, index=False, if_exists="replace")
```

```
5630
```

```
def run_query(query):
  return pd.read_sql(query,conn)
```

1. Overall Customer Churn

```
query1 = """SELECT COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate
            FROM ecomm_data"""
run_query(query1)
```

	Total_Customers	Churn	Churn_Rate
0	5630	948	16.838366

2. Warehouse to House Distance and Customer Churn

```
query3 = """SELECT 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 AS warehousetohome,
                   COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate
            FROM ecomm_data
            GROUP BY 1
            ORDER BY 4 DESC"""

run_query(query3)
```

	warehousetohome	Total_Customers	Churn	Churn_Rate
0	Far distance	469	98	20.895522
1	Moderate distance	874	176	20.137300
2	Close distance	2318	408	17.601381
3	Very close distance	1969	266	13.509396

3. Customer Tenure vs Customer Churn

```
query4 = """SELECT 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 AS Tenure,
               COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate
        FROM ecomm_data
        GROUP BY 1
        ORDER BY 4 DESC"""

run_query(query4)
```

	Tenure	Total_Customers	Churn	Churn_Rate
0	6 Months	2150	697	32.418605
1	1 Year	1584	156	9.848485
2	2 Years	1467	95	6.475801
3	more than 2 years	429	0	0.000000

4. Customer Registered Address vs Customer Churn

```
query3 = """SELECT CASE
                  WHEN NumberOfAddress <= 5 THEN 'Less than 5'
                  WHEN NumberOfAddress > 5 AND NumberOfAddress <= 10 THEN 'Between 5 to 10'
                  WHEN NumberOfAddress > 10 AND NumberOfAddress <= 15 THEN 'Between 10 to 15'
                  WHEN NumberOfAddress > 15 THEN 'Above 15'
               END AS NumberOfAddress,
               COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate
        FROM ecomm_data
        GROUP BY 1
        ORDER BY 4 DESC"""

run_query(query3)
```

	NumberOfAddress	Total_Customers	Churn	Churn_Rate
0	Above 15	4	2	50.000000
1	Between 10 to 15	98	23	23.469388
2	Between 5 to 10	1351	277	20.503331
3	Less than 5	4177	646	15.465645

5. Customer Order Frequency vs Customer Churn

```
query3 = """SELECT CASE
                      WHEN DaySinceLastOrder <= 7 THEN 'Less than a week'
                      WHEN DaySinceLastOrder > 7 AND DaySinceLastOrder <= 14 THEN 'Between 1 to 2 weeks'
                      WHEN DaySinceLastOrder > 14 AND DaySinceLastOrder <= 21 THEN 'Between 2 to 3 weeks'
                      WHEN DaySinceLastOrder > 21 THEN 'Above 3 weeks'
                   END AS DaySinceLastOrder,
                   COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate
            FROM ecomm_data
            GROUP BY 1
            ORDER BY 4 DESC"""

run_query(query3)
```

	DaySinceLastOrder	Total_Customers	Churn	Churn_Rate
0	Above 3 weeks	3	1	33.333333
1	Less than a week	4328	825	19.061922
2	Between 1 to 2 weeks	1240	118	9.516129
3	Between 2 to 3 weeks	59	4	6.779661

6. Analysis across category

```
def cat_query(col1, col2=None):
  if col2:
    query = f"""SELECT {col1}, {col2},COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate, AVG(Complain) AS Avg_Complain
              FROM ecomm_data
              GROUP BY {col1}, {col2}
              ORDER BY 5 DESC"""
    return run_query(query)
  else:
    query = f"""SELECT {col1}, COUNT(*) AS Total_Customers, SUM(Churn) AS Churn, SUM(Churn)*100.0/COUNT(*) AS Churn_Rate, AVG(Complain) AS Avg_Complain
              FROM ecomm_data
              GROUP BY {col1}
              ORDER BY 4 DESC"""
    return run_query(query)
```

```
cat_query('PreferredLoginDevice')
```

	PreferredLoginDevice	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	Computer	1634	324	19.828641	0.283966
1	Phone	3996	624	15.615616	0.285285

```
cat_query('PreferredPaymentMode')
```

	PreferredPaymentMode	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	Cash on Delivery	514	128	24.902724	0.260700
1	E wallet	614	140	22.801303	0.299674
2	UPI	414	72	17.391304	0.318841
3	Debit Card	2314	356	15.384615	0.280035
4	Credit Card	1774	252	14.205186	0.285231

```
cat_query('Gender')
```

	Gender	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	Male	3384	600	17.730496	0.270095
1	Female	2246	348	15.494212	0.307213

```
cat_query('PreferedOrderCat')
```

	PreferedOrderCat	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	Mobile	2080	570	27.403846	0.293269
1	Fashion	826	128	15.496368	0.292978
2	Laptop & Accessory	2050	210	10.243902	0.272195
3	Others	264	20	7.575758	0.257576
4	Grocery	410	20	4.878049	0.307317

```
cat_query('MaritalStatus')
```

	MaritalStatus	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	Single	1796	480	26.726058	0.283964
1	Divorced	848	124	14.622642	0.292453
2	Married	2986	344	11.520429	0.283322

```
cat_query('Complain')
```

	Complain	Total_Customers	Churn	Churn_Rate
0	1	1604	508	31.670823
1	0	4026	440	10.928962

```
cat_query('SatisfactionScore')
```

	SatisfactionScore	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	5	1108	264	23.826715	0.290614
1	3	1698	292	17.196702	0.277974
2	4	1074	184	17.132216	0.249534
3	2	586	74	12.627986	0.290102
4	1	1164	134	11.512027	0.319588

```
cat_query('CityTier')
```

	CityTier	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	3	1722	368	21.370499	0.289199
1	2	242	48	19.834711	0.256198
2	1	3666	532	14.511729	0.284779

```
cat_query('NumberOfDeviceRegistered')
```

	NumberOfDeviceRegistered	Total_Customers	Churn	Churn_Rate	Avg_Complain
0	6	162	56	34.567901	0.302469
1	5	881	198	22.474461	0.284904
2	4	2377	392	16.491376	0.283971
3	3	1699	254	14.949971	0.285462
4	2	276	26	9.420290	0.282609
5	1	235	22	9.361702	0.280851

```
query2="""SELECT Churn, AVG(Tenure) AS Avg_Tenure, AVG(HourSpendOnApp) AS Avg_HourSpendOnApp,
                 AVG(OrderAmountHikeFromLastYear) AS Avg_OrderAmountHikeFromLastYear, AVG(CouponUsed) AS Avg_CouponUsed, AVG(OrderCount) AS Avg_OrderCount,
                 AVG(DaySinceLastOrder) AS Avg_DaySinceLastOrder, AVG(CashbackAmount) AS Avg_CashbackAmount
          FROM ecomm_data
```

```
        GROUP BY Churn
        ORDER BY Churn DESC"""
run_query(query2)
```

	Churn	Avg_Tenure	Avg_HourSpendOnApp	Avg_OrderAmountHikeFromLastYear	Avg_CouponUsed	Avg_OrderCount	Avg_DaySinceLastOrder	Avg_CashbackAmount
0	1	3.961373	2.959946	15.628598	1.717308	2.827156	3.310494	160.369198
1	0	11.451036	2.925782	15.723983	1.757850	3.044622	4.793145	180.633704

Insights and Recommendations

- Overall Customer Churn rate is 16.38%, churn is significantly higher among single customers (26.7%) compared to divorced (14.6%) and married (11.5%) customers. **Understanding behavioral differences between these groups can help reduce churn.**
- Warehouse to House - Customers further from warehouses are churning more (~20%), likely due to longer delivery times, high shipping costs, or unreliable deliveries. Possible solutions could be **introduction premium option for expediated shipping, route optimization, or introducing regional warehouses closer to customers**
- Customer Tenure - Customers with lower tenure (< 6 months) are more likely to churn, so the focus should be on **engagement, incentives/cashback, and personalized experiences to encourage long-term retention.**
- Registered Address - Customers with higher registered addresses (> 5) are more likely to churn. Possible due to account sharing. Offer personalized experience based on address, focus on improving convinience and satisfaction
- Days Since Last Order - Churn rate is highest among customers who ordered less than a week ago (19.06%). These might be **first-time buyers** or **impulse shoppers** who didn't form a habit of returning.
- City Tier - Tier 3 cities have the highest churn (21.37%) possibly due to delivery challenges, fewer promotions, or lack of brand trust. Invest in **better delivery logistics, offer localized promotions, and enhance service awareness in Tier 3 cities.**
- Complaints - Customers who complain have a 31.67% churn rate! Complaint resolution is a major driver of retention. **Strengthen customer service with faster complaint resolution, proactive issue handling**, and loyalty rewards for resolved cases.
- Order Category - Mobile category has the highest churn (27.4%). Likely due to competitive pricing, after-sales issues, or fraud. Provide better warranty services, price matching, and financing options. Improve post-purchase support for mobile buyers.
- Login Device - Computer Users Churn More (19.8%) vs. Phone Users (15.6%). Computer users might be facing usability or experience issues. Improve desktop user experience (UI/UX enhancements, faster loading times).
- Payment Mode - Cash on Delivery (COD) has the highest churn (24.9%).COD orders may face delays or customer dissatisfaction.