```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_excel('/content/E Commerce Dataset.xlsx', sheet_name = 'E Comm')
df
```

	CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	Prefe
0	50001	1	4.0	Mobile Phone	3	6.0	
1	50002	1	NaN	Phone	1	8.0	
2	50003	1	NaN	Phone	1	30.0	
3	50004	1	0.0	Phone	3	15.0	
4	50005	1	0.0	Phone	1	12.0	
...	...	...	...	...	...	...	
5625	55626	0	10.0	Computer	1	30.0	
5626	55627	0	13.0	Mobile Phone	1	13.0	
5627	55628	0	1.0	Mobile Phone	1	11.0	
5628	55629	0	23.0	Computer	3	9.0	
5629	55630	0	8.0	Mobile Phone	1	15.0	

5630 rows × 20 columns

Next steps: View recommended plots

```
df.shape
```

```
(5630, 20)
```

```
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   float64
dtypes: float64(8), int64(7), object(5)
memory usage: 879.8+ KB
```

```
df.nunique()
```

```
CustomerID                     5630
Churn                             2
Tenure                           36
PreferredLoginDevice              3
CityTier                          3
WarehouseToHome                  34
PreferredPaymentMode              7
Gender                            2
HourSpendOnApp                    6
NumberOfDeviceRegistered          6
PreferedOrderCat                  6
SatisfactionScore                 5
MaritalStatus                     3
NumberOfAddress                  15
Complain                          2
OrderAmountHikeFromlastYear      16
CouponUsed                       17
OrderCount                       16
DaySinceLastOrder                22
CashbackAmount                 2586
dtype: int64
```

```
columns = df.columns.to_list()
columns
```

```
['CustomerID',
 'Churn',
```

```
       'Tenure',
       'PreferredLoginDevice',
       'CityTier',
       'WarehouseToHome',
       'PreferredPaymentMode',
       'Gender',
       'HourSpendOnApp',
       'NumberOfDeviceRegistered',
       'PreferedOrderCat',
       'SatisfactionScore',
       'MaritalStatus',
       'NumberOfAddress',
       'Complain',
       'OrderAmountHikeFromlastYear',
       'CouponUsed',
       'OrderCount',
       'DaySinceLastOrder',
       'CashbackAmount']
```

```
df.select_dtypes(exclude=np.number).columns
```

```
Index(['PreferredLoginDevice', 'PreferredPaymentMode', 'Gender',
       'PreferedOrderCat', 'MaritalStatus'],
      dtype='object')
```

```
for col in  df.columns:
  if df[col].dtype == object:
    print(str(col) + ':' + str(df[col].unique()))
    print(df[col].value_counts())
    print('--------------------------------------------------------------------------------
```

```
PreferredLoginDevice:['Mobile Phone' 'Phone' 'Computer']
PreferredLoginDevice
Mobile Phone    2765
Computer        1634
Phone           1231
Name: count, dtype: int64
--------------------------------------------------------------------------------------
PreferredPaymentMode:['Debit Card' 'UPI' 'CC' 'Cash on Delivery' 'E wallet' 'COD' 'Cred
PreferredPaymentMode
Debit Card          2314
Credit Card         1501
E wallet             614
UPI                  414
COD                  365
CC                   273
Cash on Delivery     149
Name: count, dtype: int64
--------------------------------------------------------------------------------------
Gender:['Female' 'Male']
Gender
Male      3384
Female    2246
Name: count, dtype: int64
```

```
--------------------------------------------------------------------------------------
PreferedOrderCat:['Laptop & Accessory' 'Mobile' 'Mobile Phone' 'Others' 'Fashion' 'Groc
PreferedOrderCat
Laptop & Accessory    2050
Mobile Phone          1271
Fashion                826
Mobile                 809
Grocery                410
Others                 264
Name: count, dtype: int64
--------------------------------------------------------------------------------------
MaritalStatus:['Single' 'Divorced' 'Married']
MaritalStatus
Married     2986
Single      1796
Divorced     848
Name: count, dtype: int64
--------------------------------------------------------------------------------------
```

```
df.select_dtypes(include=np.number).columns
```

```
Index(['CustomerID', 'Churn', 'Tenure', 'CityTier', 'WarehouseToHome',
       'HourSpendOnApp', 'NumberOfDeviceRegistered', 'SatisfactionScore',
       'NumberOfAddress', 'Complain', 'OrderAmountHikeFromlastYear',
       'CouponUsed', 'OrderCount', 'DaySinceLastOrder', 'CashbackAmount'],
      dtype='object')
```

```
for col in df.columns:
  if df[col].dtype == float or df[col].dtype == int:
    print(str(col) + ' : ' + str(df[col].unique()))
    print(df[col].value_counts())
    print('--------------------------------------------------------------------------------
```

```
2.0     792
1.0     614
8.0     538
0.0     496
7.0     447
4.0     431
9.0     299
5.0     228
10.0    157
6.0     113
11.0     91
12.0     69
13.0     51
14.0     35
15.0     19
17.0     17
16.0     13
18.0     10
30.0      1
46.0      1
31.0      1
Name: count, dtype: int64
---------------------------------------------------------------------------------
CashbackAmount : [159.93 120.9  120.28 ... 173.77 287.91 173.78]
CashbackAmount
123.42    8
149.36    8
148.42    8
188.47    7
154.73    7
         ..
174.84    1
127.74    1
145.05    1
174.28    1
173.78    1
Name: count, Length: 2586, dtype: int64
---------------------------------------------------------------------------------
```

```
df.loc[df['PreferredLoginDevice'] == 'Phone', 'PreferredLoginDevice' ] = 'Mobile Phone'
df.loc[df['PreferedOrderCat'] == 'Mobile', 'PreferedOrderCat' ] = 'Mobile Phone'
```

```
df['PreferredLoginDevice'].value_counts()
```

```
PreferredLoginDevice
Mobile Phone    3996
Computer        1634
Name: count, dtype: int64
```

```
#as cod is also cash on delievery
#as cc is also credit card so i merged them
df.loc[df['PreferredPaymentMode'] == 'COD', 'PreferredPaymentMode' ] = 'Cash on Delivery'
df.loc[df['PreferredPaymentMode'] == 'CC', 'PreferredPaymentMode' ] = 'Credit Card'
```

```
df['PreferredPaymentMode'].value_counts()
```

```
PreferredPaymentMode
Debit Card          2314
Credit Card         1774
E wallet             614
Cash on Delivery     514
UPI                  414
Name: count, dtype: int64
```

```
# convert num_cols to categories
df2 = df.copy()
for col in df2.columns:
  if col == 'CustomerID':
    continue

  else:
    if df2[col].dtype == 'int':
      df2[col] = df[col].astype(str)

df2.dtypes
```

```
CustomerID                       int64
Churn                           object
Tenure                         float64
PreferredLoginDevice            object
CityTier                        object
WarehouseToHome                float64
PreferredPaymentMode            object
Gender                          object
HourSpendOnApp                 float64
NumberOfDeviceRegistered        object
PreferedOrderCat                object
SatisfactionScore               object
MaritalStatus                   object
NumberOfAddress                 object
Complain                        object
OrderAmountHikeFromlastYear    float64
CouponUsed                     float64
OrderCount                     float64
DaySinceLastOrder              float64
CashbackAmount                 float64
dtype: object
```

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

```
0
```

```
# the sum of null values
grouped_data = []
for col in columns:
    n_missing = df[col].isnull().sum()
    percentage = n_missing / df.shape[0] * 100
    grouped_data.append([col, n_missing, percentage])

# Create a new DataFrame from the grouped data
grouped_df = pd.DataFrame(grouped_data, columns=['column', 'n_missing', 'percentage'])

# Group by 'col', 'n_missing', and 'percentage'
result = grouped_df.groupby(['column', 'n_missing', 'percentage']).size()
result
```

```
column                       n_missing  percentage
CashbackAmount               0          0.000000     1
Churn                        0          0.000000     1
CityTier                     0          0.000000     1
Complain                     0          0.000000     1
CouponUsed                   256        4.547069     1
CustomerID                   0          0.000000     1
DaySinceLastOrder            307        5.452931     1
Gender                       0          0.000000     1
HourSpendOnApp               255        4.529307     1
MaritalStatus                0          0.000000     1
NumberOfAddress              0          0.000000     1
NumberOfDeviceRegistered     0          0.000000     1
OrderAmountHikeFromlastYear  265        4.706927     1
OrderCount                   258        4.582593     1
PreferedOrderCat             0          0.000000     1
PreferredLoginDevice         0          0.000000     1
PreferredPaymentMode         0          0.000000     1
SatisfactionScore            0          0.000000     1
Tenure                       264        4.689165     1
WarehouseToHome              251        4.458259     1
dtype: int64
```

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
binary_cat_cols = ['Complain']
outcome = ['Churn']
cat_cols = ['PreferredLoginDevice', 'CityTier', 'PreferredPaymentMode',
       'Gender', 'NumberOfDeviceRegistered', 'PreferedOrderCat',
       'SatisfactionScore', 'MaritalStatus', 'NumberOfAddress', 'Complain']
num_cols = ['Tenure', 'WarehouseToHome', 'HourSpendOnApp', 'OrderAmountHikeFromlastYear', '
```

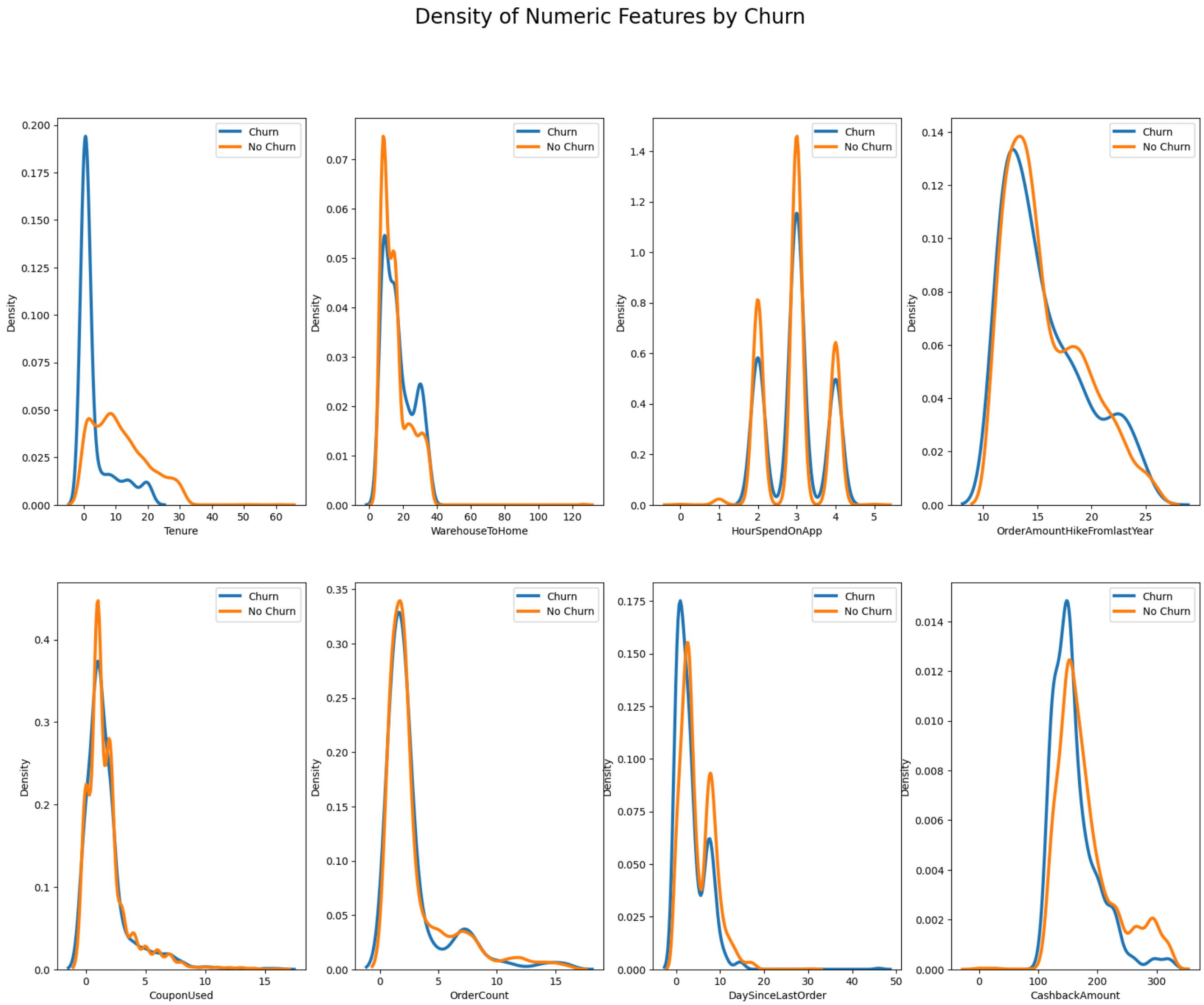
```
df_c = df[df['Churn']==1].copy()
df_nc = df[df['Churn']==0].copy()

fig, ax = plt.subplots(2,4,figsize=(20, 15))
fig.suptitle('Density of Numeric Features by Churn', fontsize=20)
ax = ax.flatten()

for idx,c in enumerate(num_cols):
    sns.kdeplot(df_c[c], linewidth= 3,
             label = 'Churn',ax=ax[idx])
    sns.kdeplot(df_nc[c], linewidth= 3,
             label = 'No Churn',ax=ax[idx])

    ax[idx].legend(loc='upper right')

plt.show()
```


Density of Numeric Features by Churn
Churn
No Churn
Density
Tenure
WarehouseToHome
HourSpendOnApp
OrderAmountHikeFromlastYear
CouponUsed
OrderCount
DaySinceLastOrder
CashbackAmount

- Tenure: Customers with longer tenure seem less likely to churn. Makes sense as longer tenure indicates satisfaction
- CityTier: Churn rate looks similar across tiers. City tier does not seem predictive of churn
- WarehouseToHome: Shorter warehouse to home distances have a lower churn rate. Faster deliveries may improve satisfaction
- HourSpendOnApp: More time spent on app correlates with lower churn. App engagement is a good sign
- NumberOfDeviceRegistered: More registered devices associates with lower churn. Access across devices improves convenience
- SatisfactionScore: Higher satisfaction scores strongly associate with lower churn, as expected. Critical driver
- NumberOfAddress: Slight downward trend in churn as number of addresses increases. More addresses indicates loyalty
- Complain: More complaints associate with higher churn, though relationship isn't very strong. Complaints hurt satisfaction
- OrderAmountHikeFromLastYear: Big spenders from last year are less likely to churn. Good to retain big customers
- CouponUsed: Coupon usage correlates with lower churn. Coupons enhance loyalty
- OrderCount: Higher order counts associate with lower churn. Frequent usage builds habits
- DaySinceLastOrder: Longer since last order correlates with higher churn. Recency is a good predictor

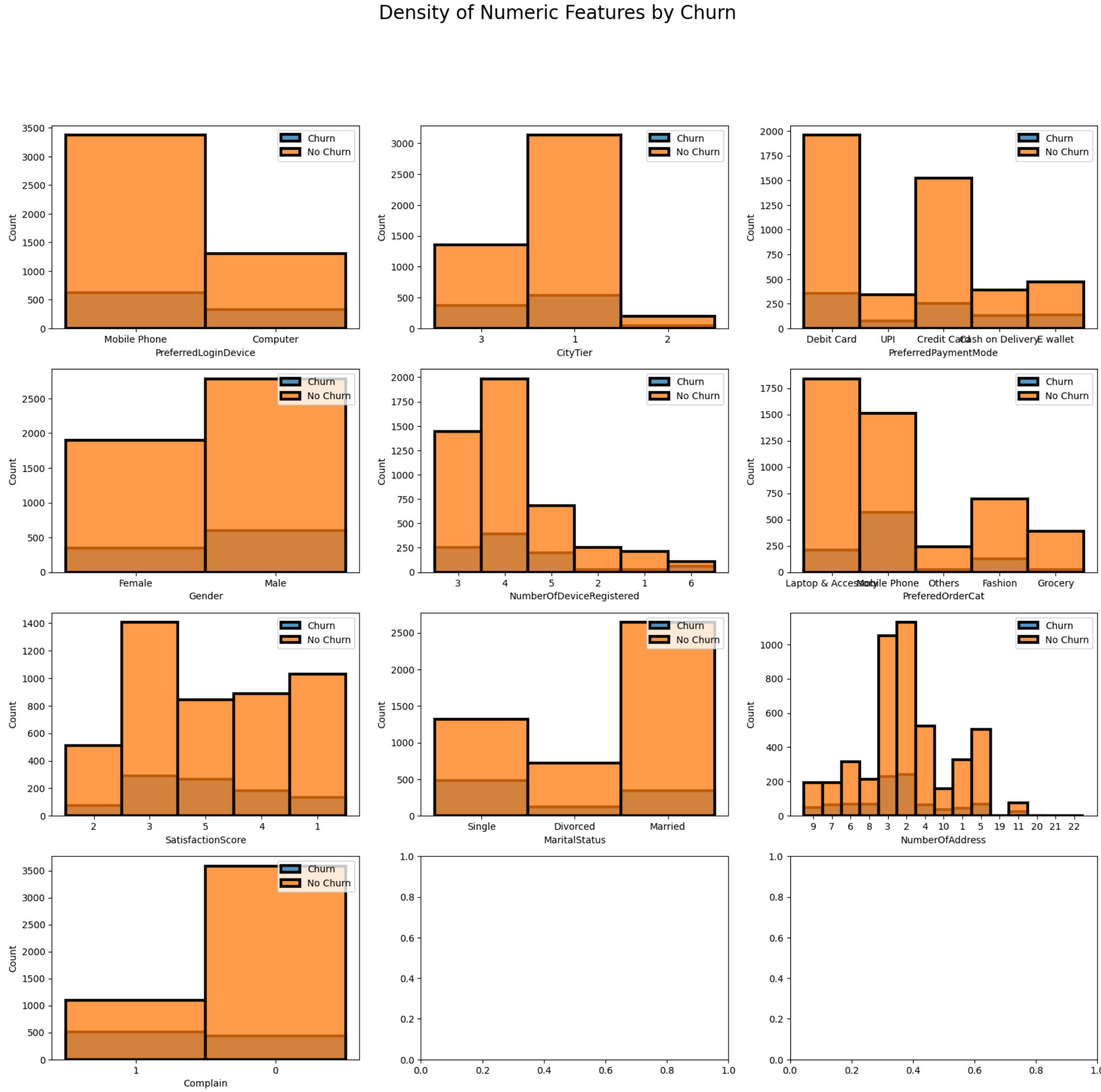
```
df_c = df2[df2['Churn']=='1'].copy()
df_nc = df2[df2['Churn']=='0'].copy()

fig, ax = plt.subplots(4,3,figsize=(20, 18))
fig.suptitle('Density of Numeric Features by Churn', fontsize=20)
ax = ax.flatten()

for idx,c in enumerate(cat_cols):
    sns.histplot(df_c[c], linewidth= 3,
             label = 'Churn',ax=ax[idx])
    sns.histplot(df_nc[c], linewidth= 3,
             label = 'No Churn',ax=ax[idx])

    ax[idx].legend(loc='upper right')

plt.show()
```


Density of Numeric Features by Churn
Churn
No Churn
Count
PreferredLoginDevice
Mobile Phone
Computer
CityTier
PreferredPaymentMode
Debit Card
UPI
Credit Card
Cash on Delivery
E wallet
Gender
Female
Male
NumberOfDeviceRegistered
PreferedOrderCat
Laptop & Accessory
Mobile Phone
Others
Fashion
Grocery
SatisfactionScore
MaritalStatus
Single
Divorced
Married
NumberOfAddress
Complain

Which Gender has more Orders?

```
df['Gender'].value_counts()
```

```
Gender
Male      3384
Female    2246
Name: count, dtype: int64
```

```
df.groupby("Churn")["Gender"].value_counts()
```

```
Churn  Gender
0      Male      2784
       Female    1898
1      Male       600
       Female     348
Name: count, dtype: int64
```

```
df.groupby("PreferredLoginDevice")["OrderCount"].value_counts()
```

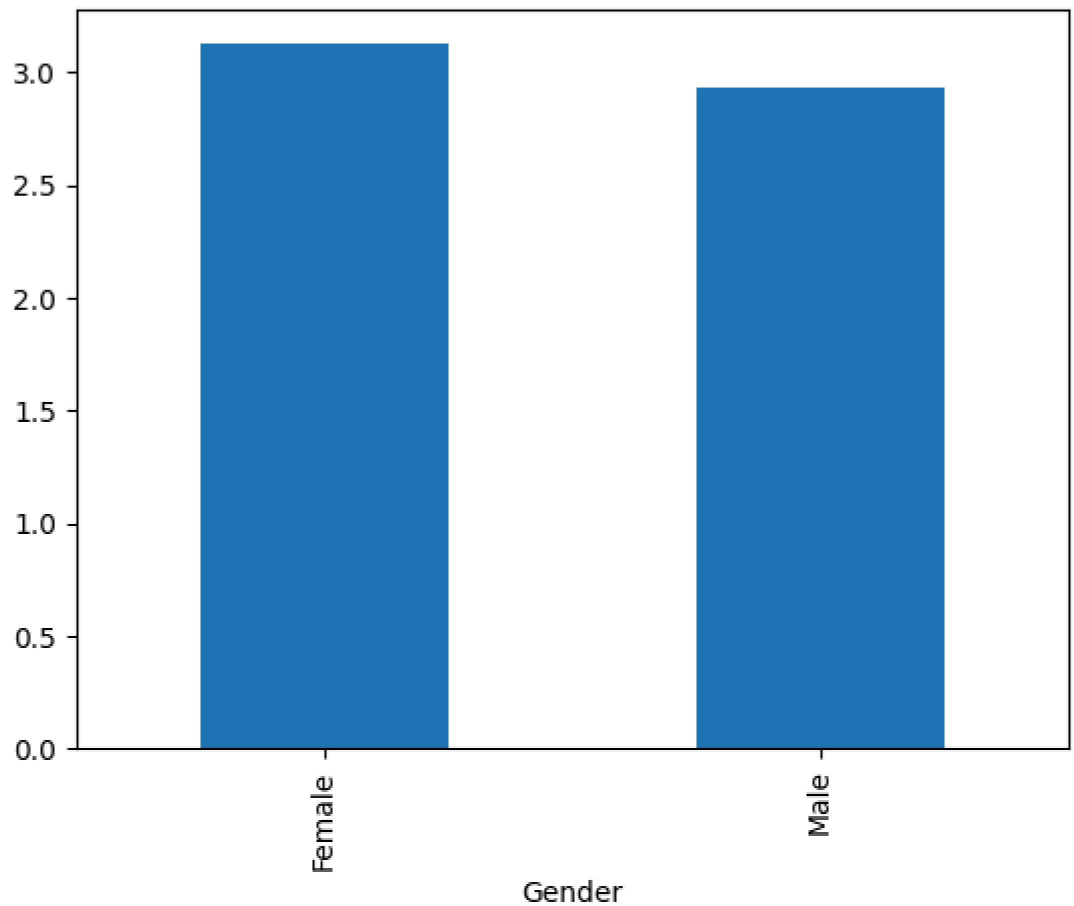
```
PreferredLoginDevice  OrderCount
Computer              2.0           573
                      1.0           486
                      3.0           132
                      4.0            61
                      7.0            59
                      5.0            48
                      8.0            44
                      6.0            40
                      14.0           20
                      9.0            19
                      11.0           16
                      10.0           15
                      12.0           15
                      13.0            9
                      15.0            8
                      16.0            4
Mobile Phone          2.0          1452
                      1.0          1265
                      3.0           239
                      7.0           147
                      4.0           143
                      5.0           133
                      8.0           128
```

```
6.0     97
9.0     43
12.0    39
11.0    35
15.0    25
13.0    21
10.0    21
16.0    19
14.0    16
Name: count, dtype: int64
```

```
gender_orders = df.groupby('Gender')['OrderCount'].mean().plot(kind='bar')

gender_orders
```

```
<Axes: xlabel='Gender'>
```




```
percentageM =600/3384 * 100
#the percentage of the leaving males out of the males
percentageM
```

```
17.73049645390071
```

```
percentageF =348/2246 * 100

percentageF  #the percentage of the leaving females out of the female
```

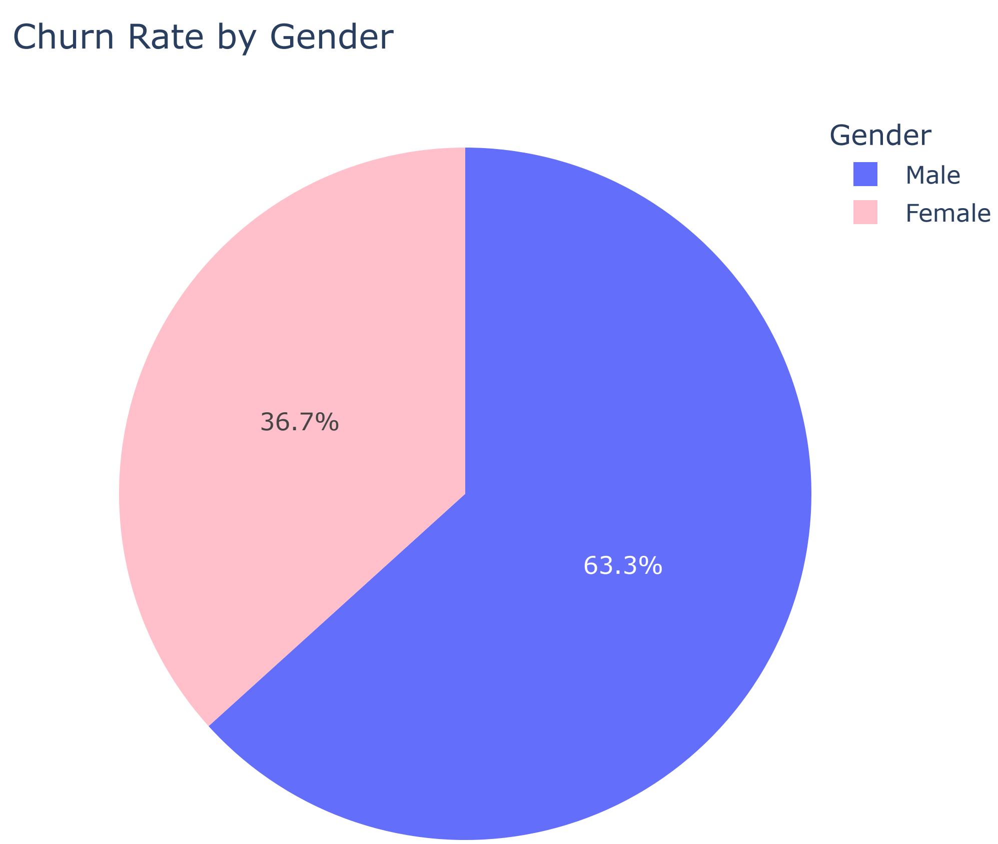
```
15.49421193232413
```

```
import pandas as pd
import plotly.express as px

# Create figure
fig = px.pie(df, values='Churn', names='Gender')
fig.update_traces(marker=dict(colors=['pink ', 'baby blue']))

# Update layout
fig.update_layout(
  title='Churn Rate by Gender',
  legend_title='Gender'
)

# Show plot
fig.show()
```



as we see the males are more likely to churn as we have 63.3 % churned males from the app may be the company should consider incresing the products that grap the males interest and so on.. we are going to see if there is another factors that makes the highest segment of churned customers are males

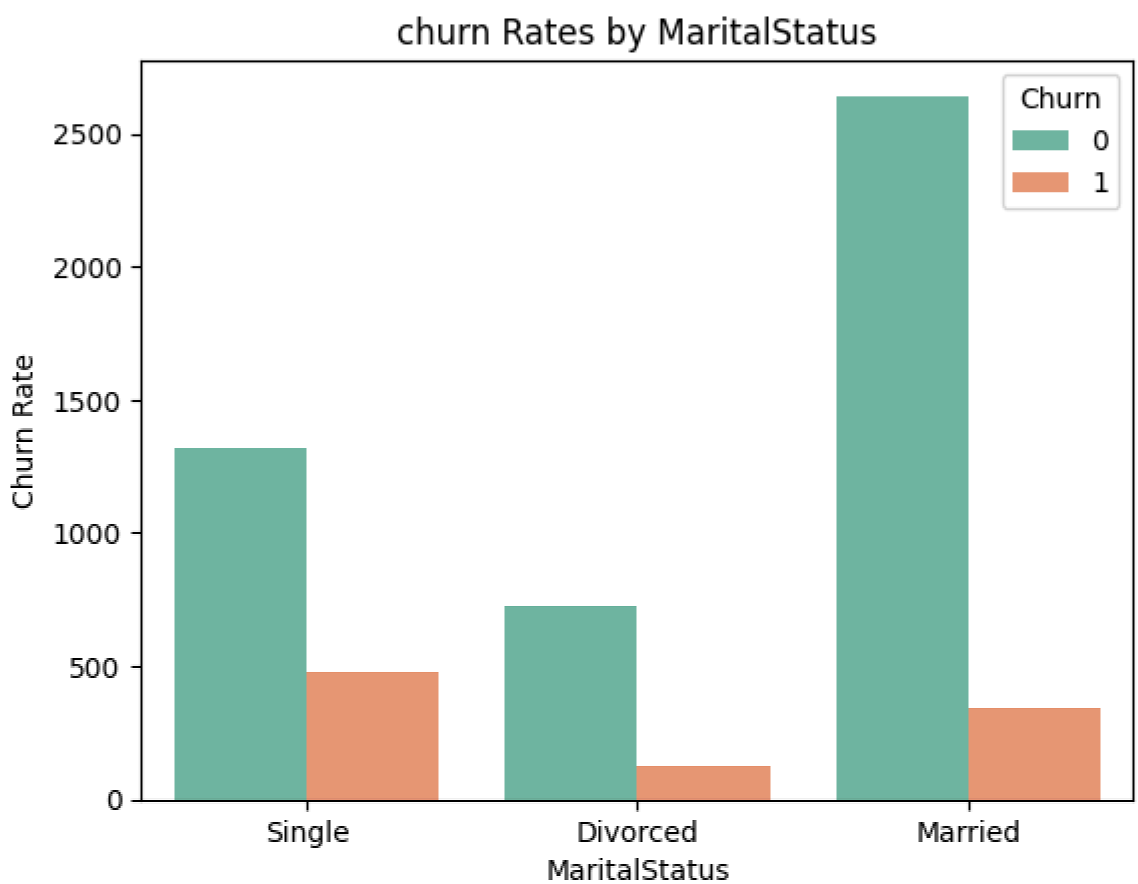
2-Which MartialStatus has the highest Churn rate?

```
df.groupby("Churn")["MaritalStatus"].value_counts()
```

```
Churn  MaritalStatus
0      Married          2642
       Single           1316
       Divorced          724
1      Single            480
       Married           344
       Divorced          124
Name: count, dtype: int64
```

```
sns.countplot(x='MaritalStatus',hue='Churn',data=df,palette='Set2')
plt.title("churn Rates by MaritalStatus")
plt.ylabel("Churn Rate")
```

```
Text(0, 0.5, 'Churn Rate')
```



the married are the highest customer segment in the comapny may be the comapny should consider taking care of the products that suits the single and the married customers as the singles are the most likely to churn from the app

3-Which CityTier has higher Tenure and OrderCount?

```
df_grouped_tenure = df.groupby('CityTier')['Tenure'].agg(['mean', 'max'])
df_grouped_tenure
```

CityTier	mean	max
1	10.528818	51.0
2	11.169725	31.0
3	9.361740	61.0

Next steps: View recommended plots

```
df_grouped_OrderCount = df.groupby('CityTier')['OrderCount'].agg(['mean', 'max'])
df_grouped_OrderCount
```

CityTier	mean	max
1	2.953255	16.0
2	2.584034	13.0
3	3.185185	16.0

Next steps: View recommended plots

citytier 2 has the highest tenure rate but the tenure rate does not seen to be a strong factor

```
df.groupby("CityTier")["OrderCount"].mean()
```

```
CityTier
1    2.953255
2    2.584034
3    3.185185
Name: OrderCount, dtype: float64
```

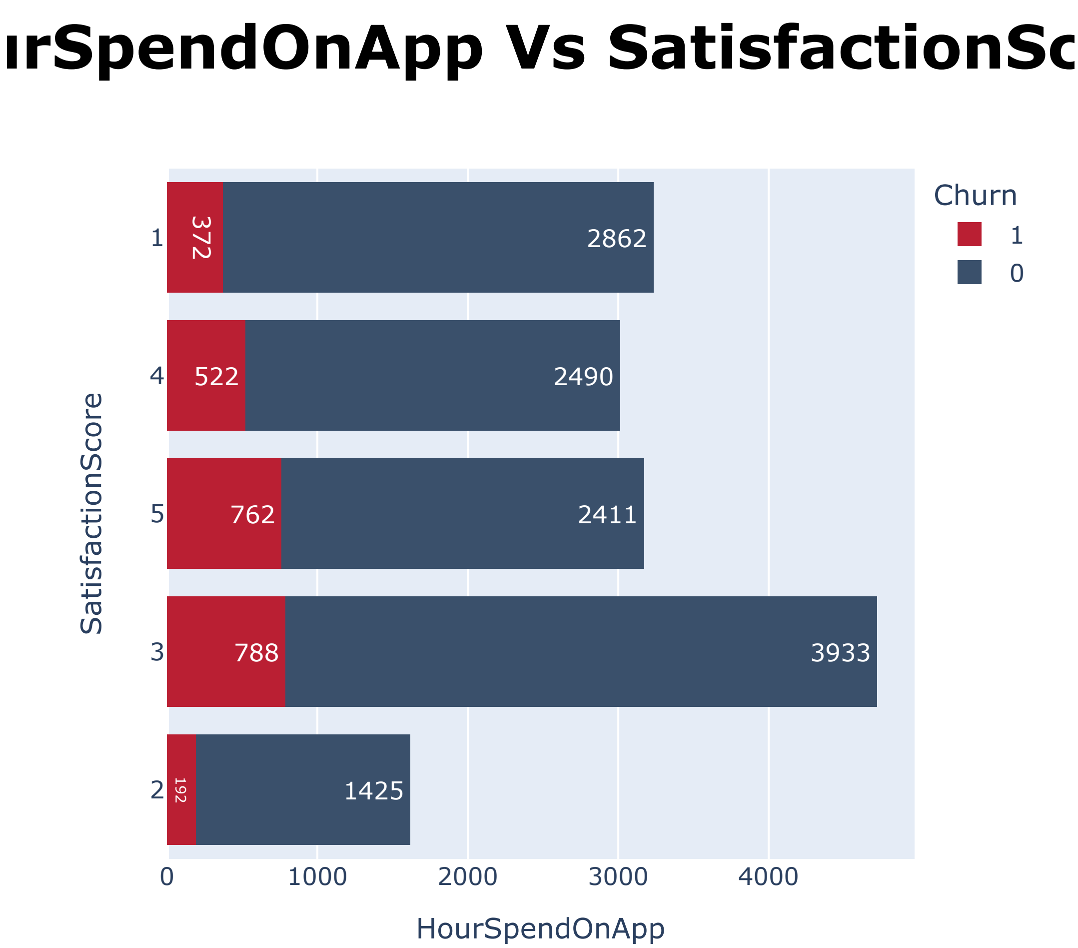
4-Is Customer with High SatisfactionScore have high HourSpendOnApp?

```
df['SatisfactionScore'].dtypes
```

```
dtype('int64')
```

```
fig = px.histogram(df2, x="HourSpendOnApp", y="SatisfactionScore", orientation="h", color="

# Customize the plot
fig.update_layout(hovermode='x',title_font_size=30)
fig.update_layout(
title_font_color="black",
template="plotly",
title_font_size=30,
hoverlabel_font_size=20,
title_x=0.5,
xaxis_title='HourSpendOnApp',
yaxis_title='SatisfactionScore',
)
fig.show()
```



as we see people with less satisfaction score spend less time on the app than the people of satisfaction score 5 but also i do not think there is any realation between the satisfaction score and people's spent time on the app

5-Which CityTier has the most HourSpendOnApp?

```
g = sns.FacetGrid(df, col='CityTier')
g.map(sns.distplot, 'HourSpendOnApp')
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```