In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
tele_data=pd.read_csv("Customer_churn.csv")
```

In [3]:
```
tele_data.head()
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	

5 rows × 21 columns

In [4]:
```
#check the shape of the data
tele_data.shape
```

Out[4]:
```
(7043, 21)
```

In [5]:
```
tele_data.columns.values
```

Out[5]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [6]:
```
#checking the datatypes of the columns
tele_data.dtypes
```

Out[6]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [7]:
```
#Descriptive Statistics
tele_data.describe()
```

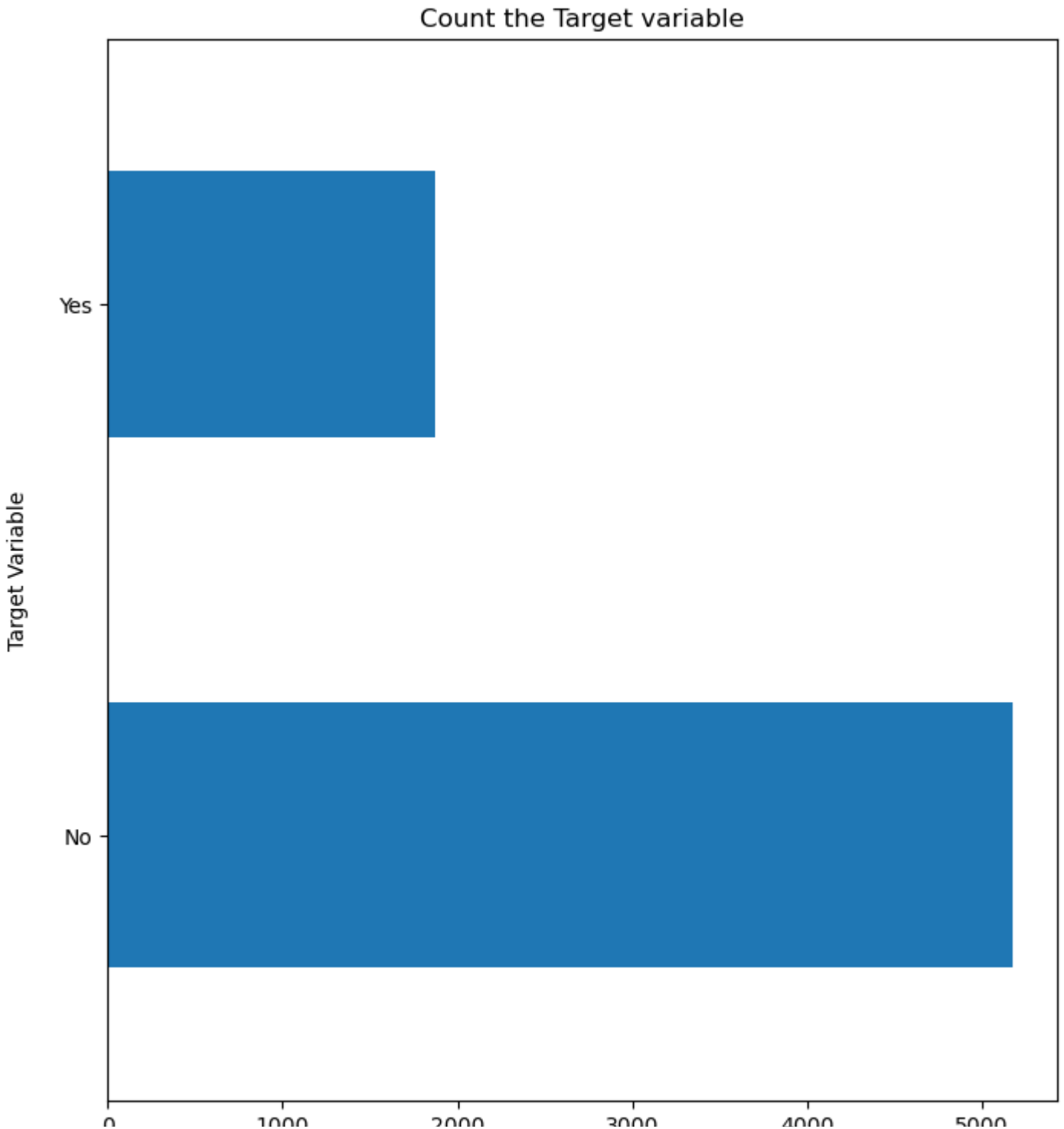
Out[7]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Senior citizen is a categorical data hence mean std are not applicable Average Monthly Spent of a person is 64.76USD and less than 25% people spent 35USD on Mobile Facilities.

In [8]:
```
tele_data['Churn'].value_counts().plot(kind='barh',figsize=(8,9))
plt.xlabel('count',labelpad=14)
plt.ylabel('Target Variable',labelpad=14)
plt.title("Count the Target variable")
```

Out[8]:
```
Text(0.5, 1.0, 'Count the Target variable')
```


Count the Target variable
Target Variable
Yes
No
0
1000
2000
3000
4000
5000

In [9]:
```
100*tele_data['Churn'].value_counts()/len(tele_data['Churn'])
```

Out[9]:
```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

In [10]:
```
tele_data['Churn'].value_counts()
```

Out[10]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

The Data is highly imbalanced with a ratio of 72:26

So we analyse the data with other features while taking the target values seperately to generate insights

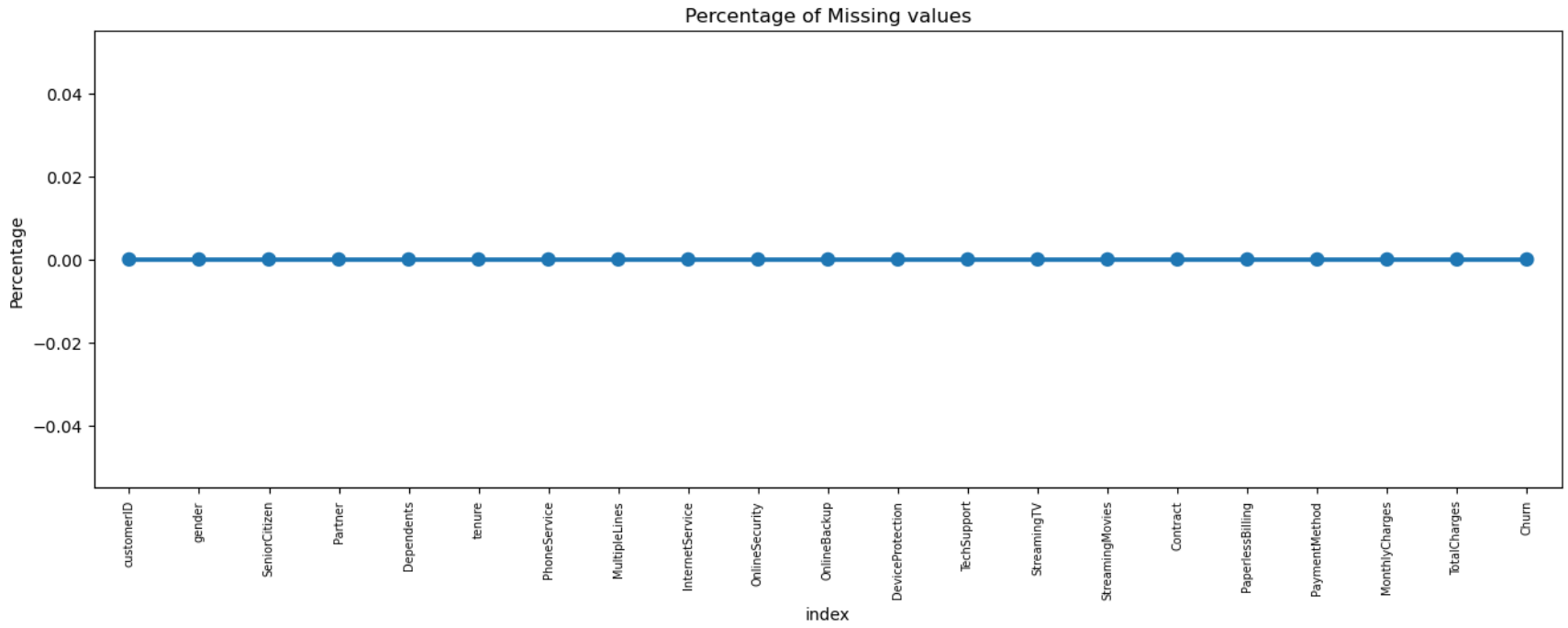
In [11]:
```
tele_data.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [12]:

```
missing=pd.DataFrame((tele_data.isnull().sum())*100/tele_data.shape[0]).reset_index()
plt.figure(figsize=(16,5))
ax=sns.pointplot('index',0,data=missing)
plt.xticks(rotation=90,fontsize=7)
plt.title("Percentage of Missing values")
plt.ylabel("Percentage")
plt.show()
```

```
C:\Users\91800\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as ke
yword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Data Cleaning

In [13]:
```
data_copy=tele_data.copy()
```

Total Charges should be a numeric values

In [14]:
```
data_copy.TotalCharges=pd.to_numeric(data_copy.TotalCharges,errors='coerce')
data_copy.isnull().sum()
```

Out[14]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

Here 11 values are null

In [15]:
```
data_copy.loc[data_copy['TotalCharges'].isnull()==True]
```

Out[15]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Devi
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	...	
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	...	

11 rows × 21 columns

In [16]:
```
#Removing missing values
data_copy.dropna(how='any',inplace=True)
```

Divide customer into bins based on tenure

In [17]:
```
print(data_copy['tenure'].max())
```

```
72
```

In [18]:
```
labels=["{0}-{1}".format(i,i+11) for i in range(1,72,12)]
data_copy['tenure_group']=pd.cut(data_copy.tenure,range(1,80,12),right=False,labels=labels)
```

In [19]:
```
data_copy['tenure_group'].value_counts()
```

Out[19]:
```
1-12     2175
61-72    1407
13-24    1024
25-36     832
49-60     832
37-48     762
Name: tenure_group, dtype: int64
```

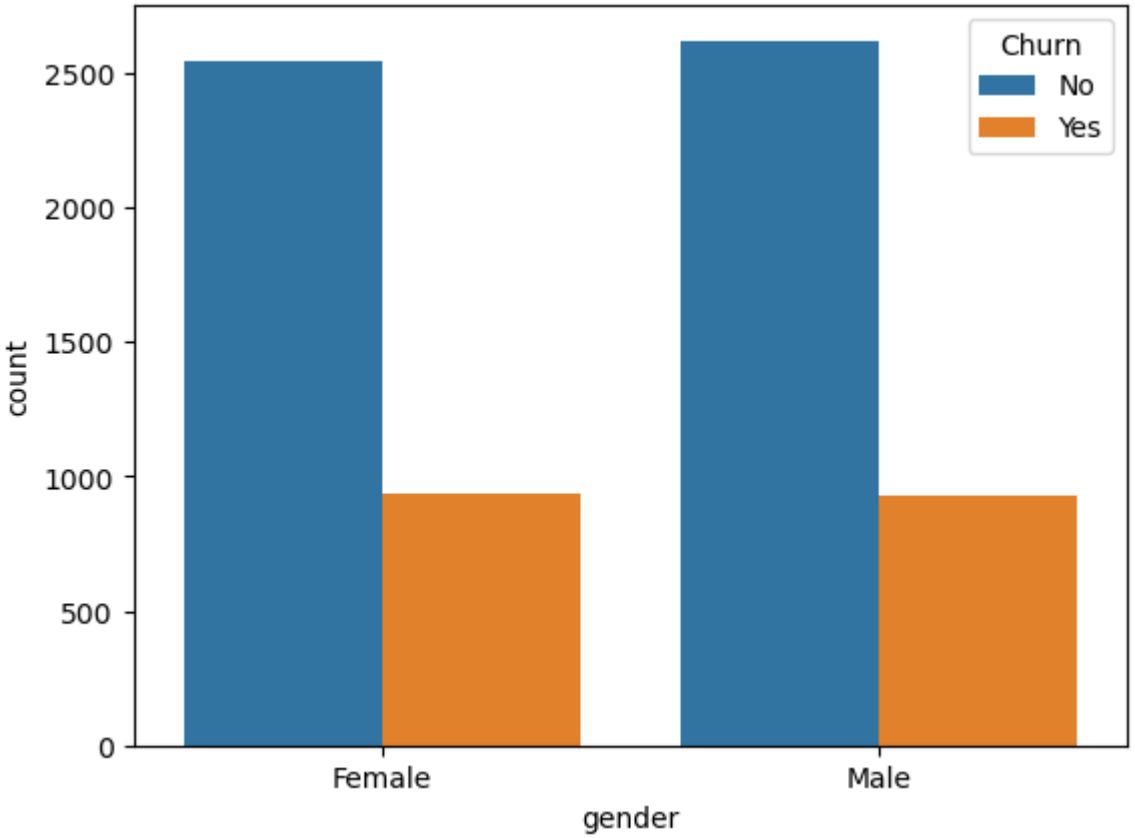
In [20]:
```
data_copy.drop(columns=['customerID','tenure'],axis=1,inplace=True)
data_copy.head()
```

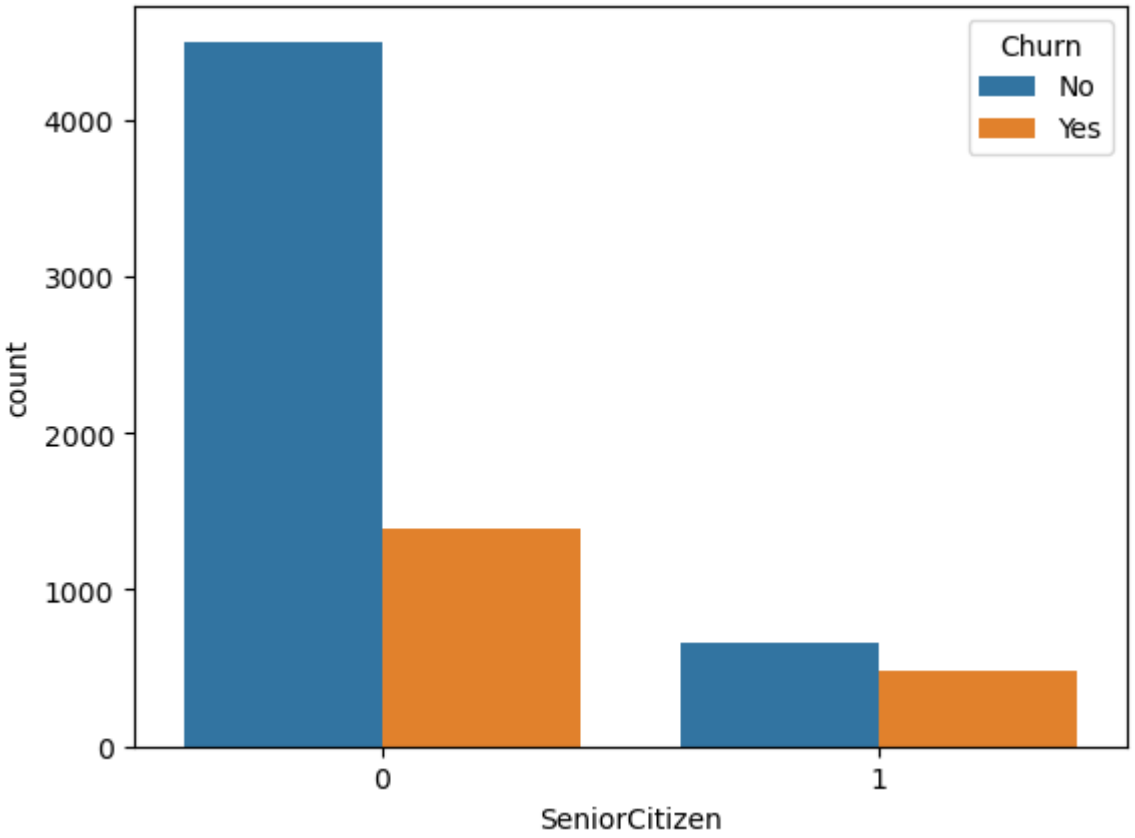
Out[20]:

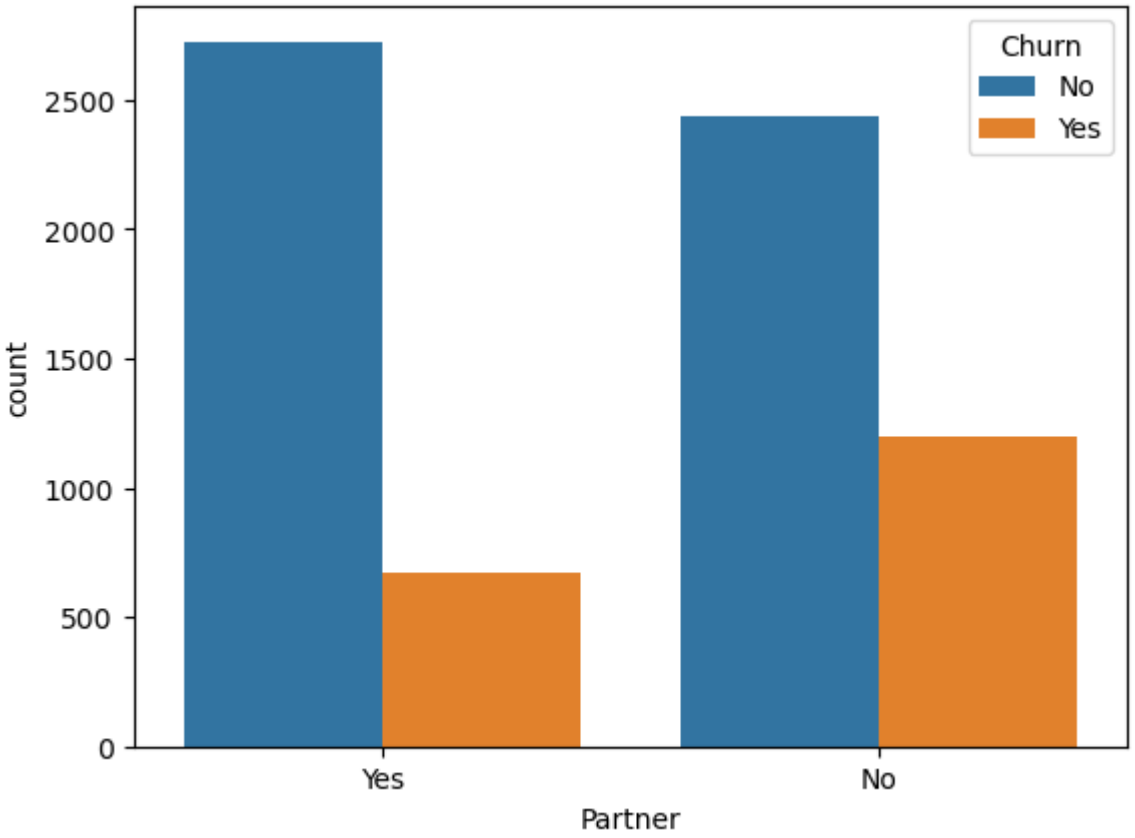
	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No

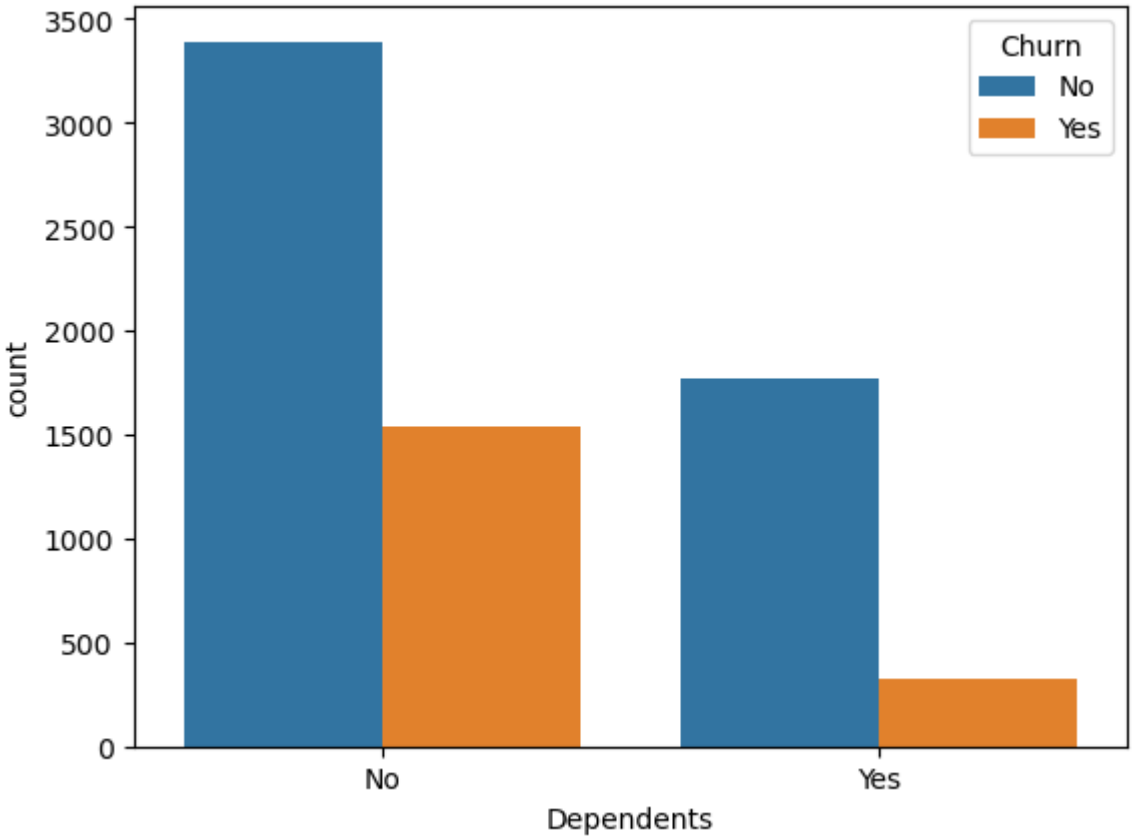
Data Exploration

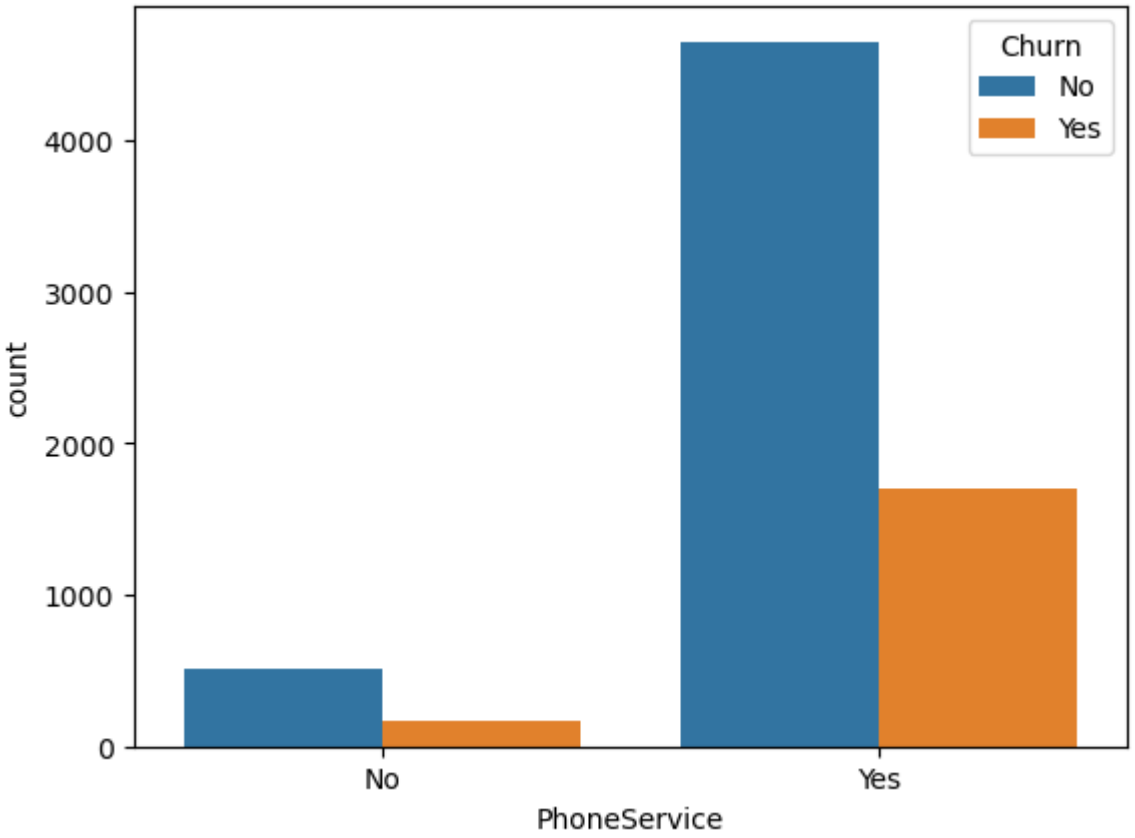
In [21]:
```
for i, predictor in enumerate(data_copy.drop(columns=['MonthlyCharges','TotalCharges','Churn'])):
    plt.figure(i)
    sns.countplot(data=data_copy,x=predictor,hue='Churn')
```

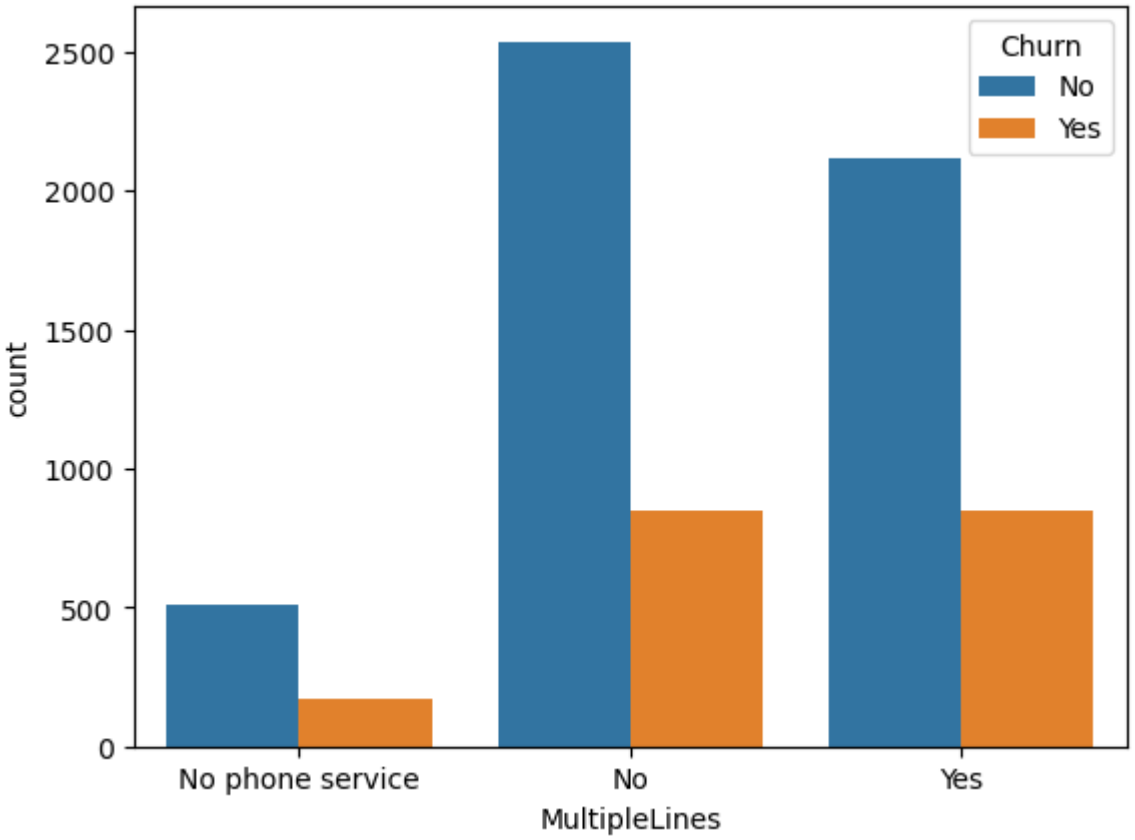

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Female
Male
gender

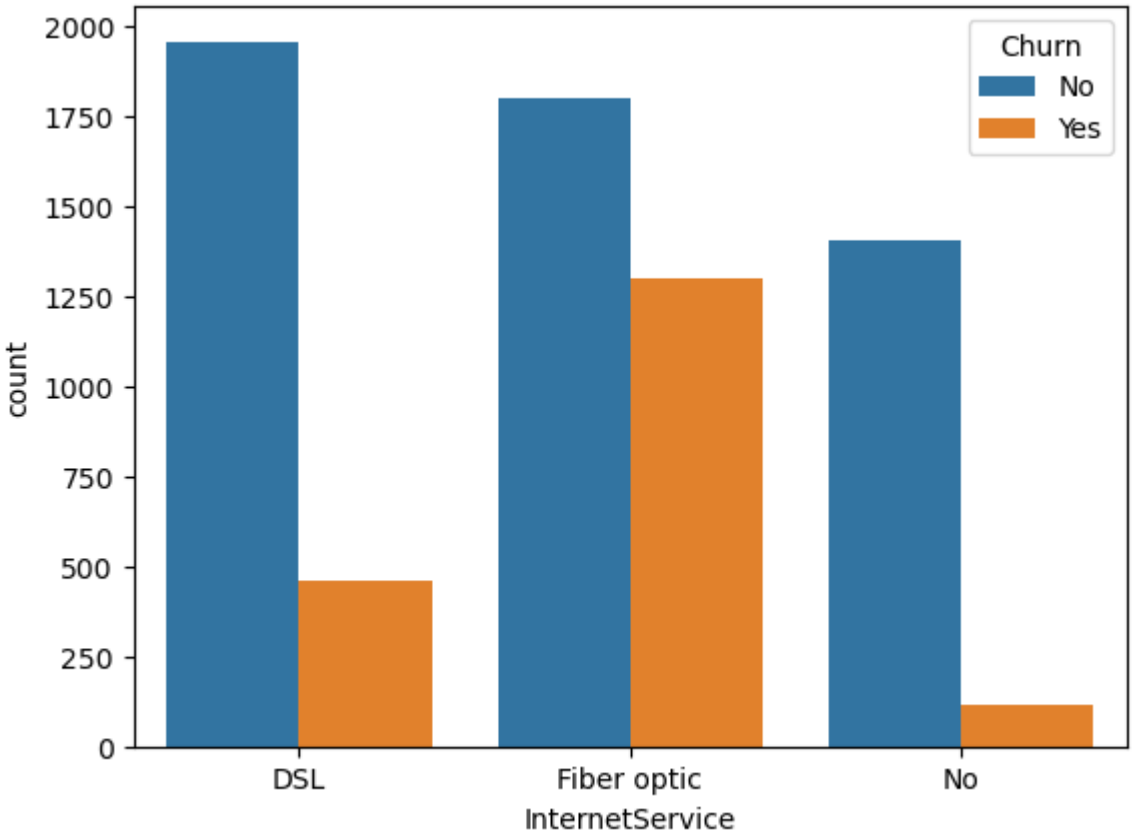

Churn
No
Yes
4000
3000
2000
1000
0
count
0
1
SeniorCitizen

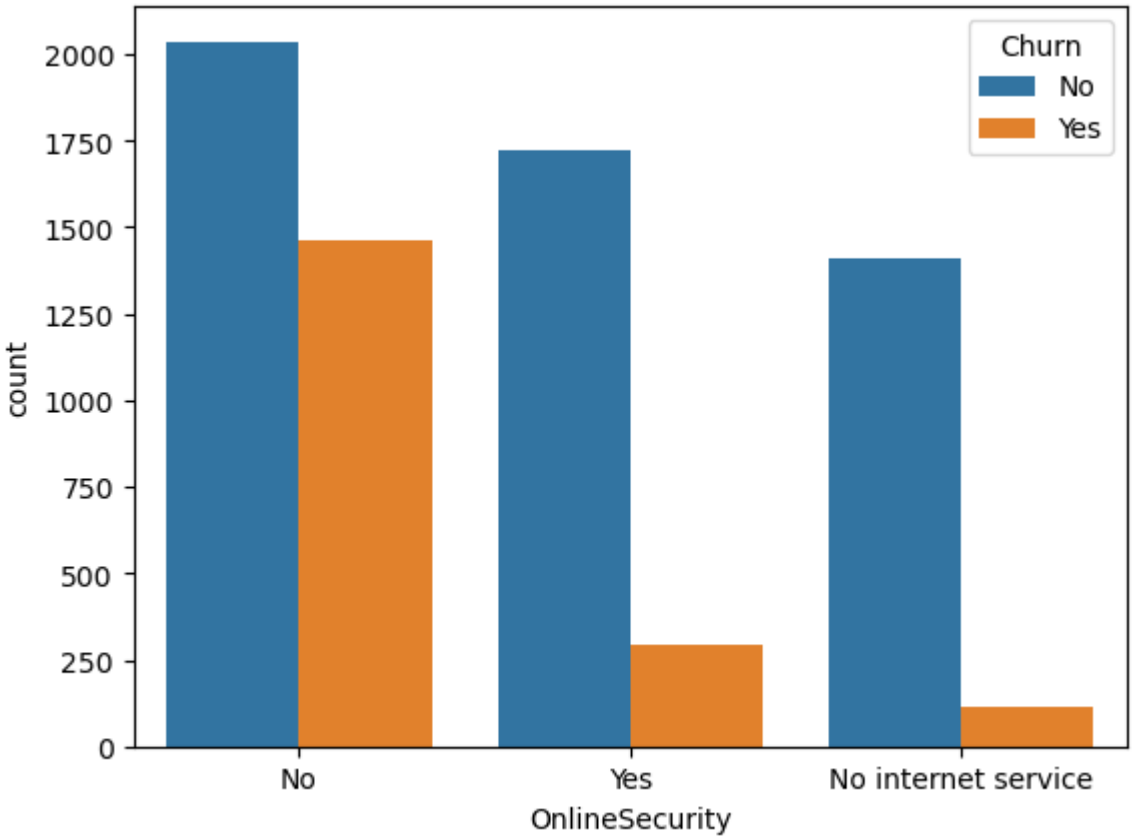

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
Partner

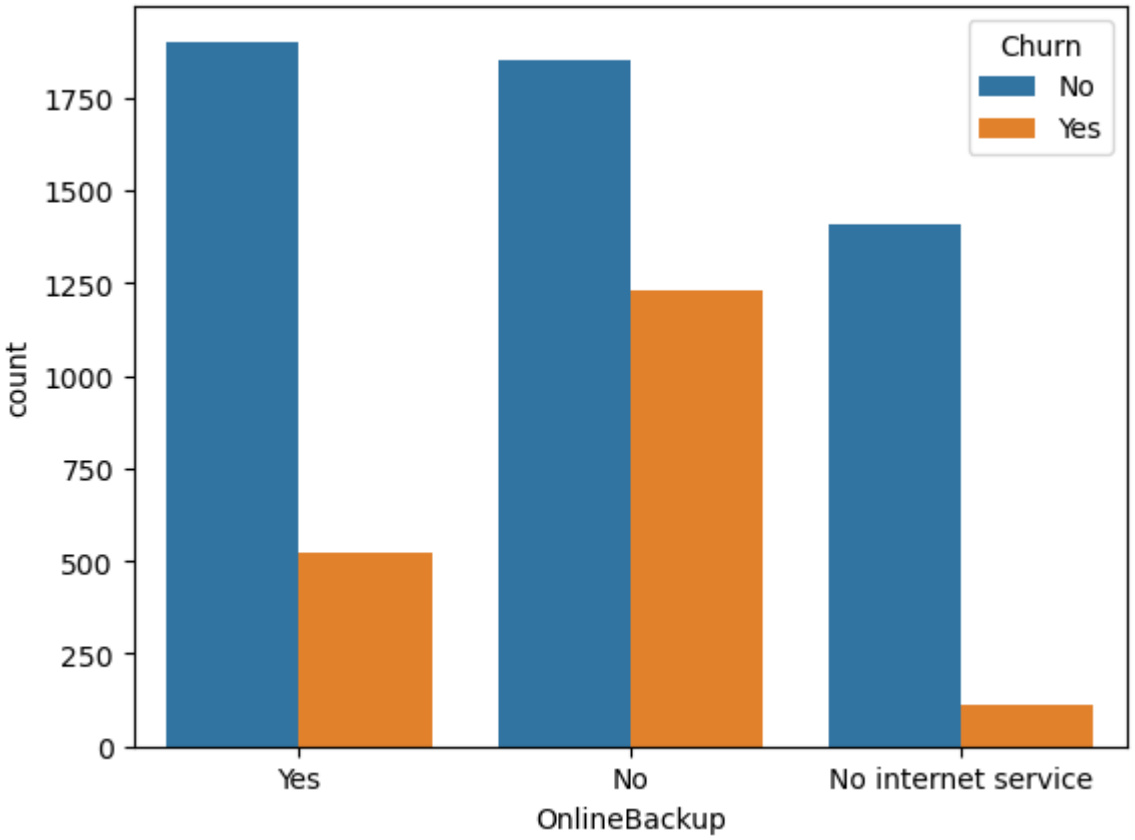

Churn
No
Yes
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents

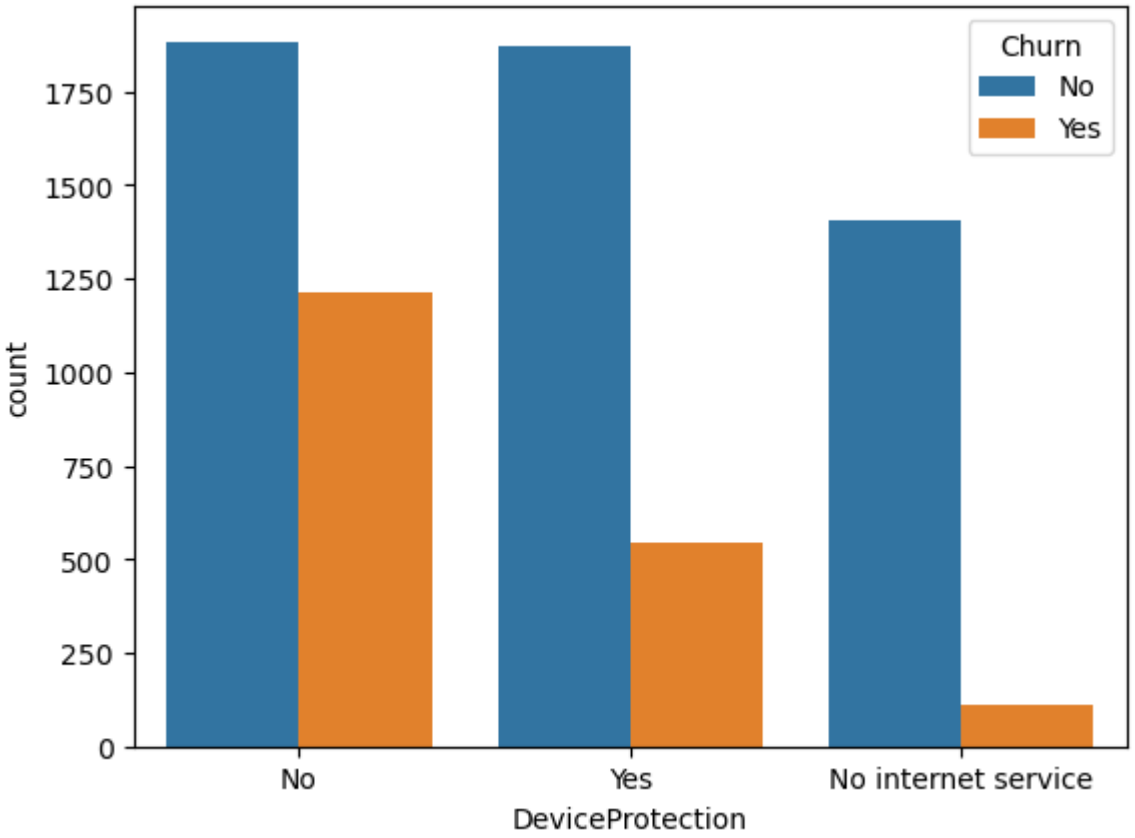

Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PhoneService

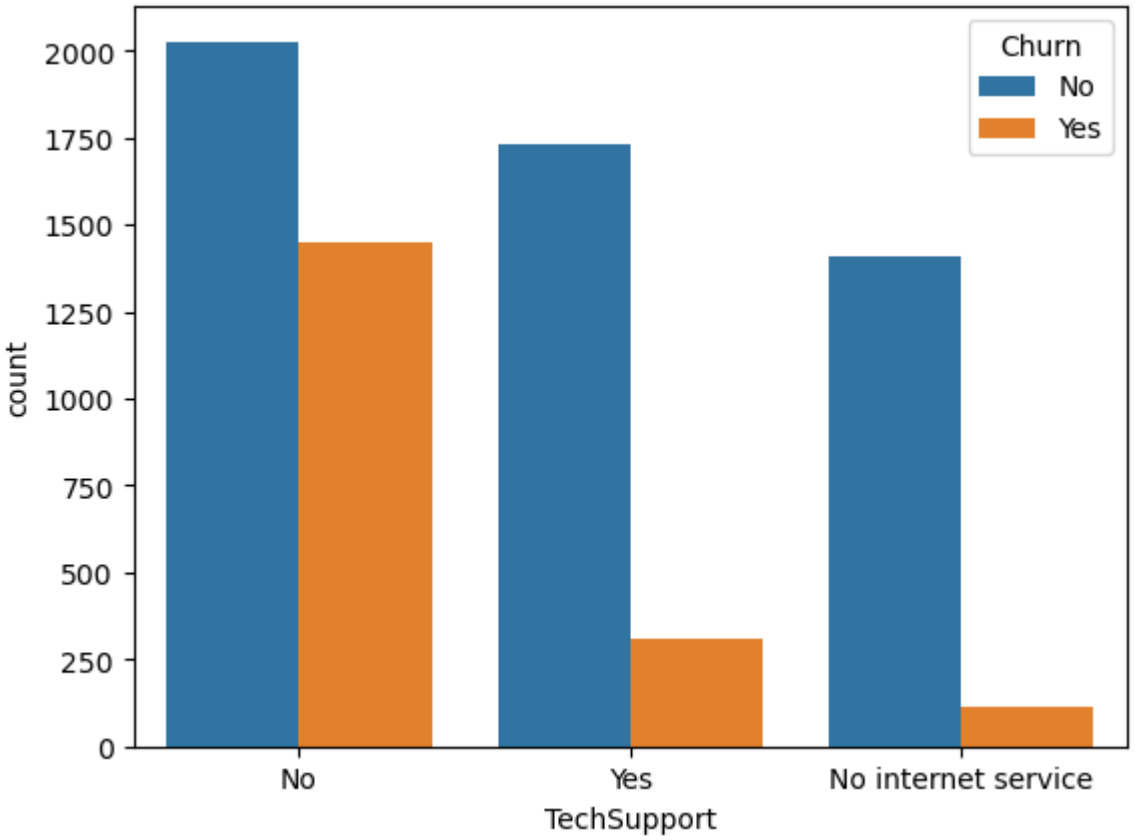

Churn
No
Yes
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

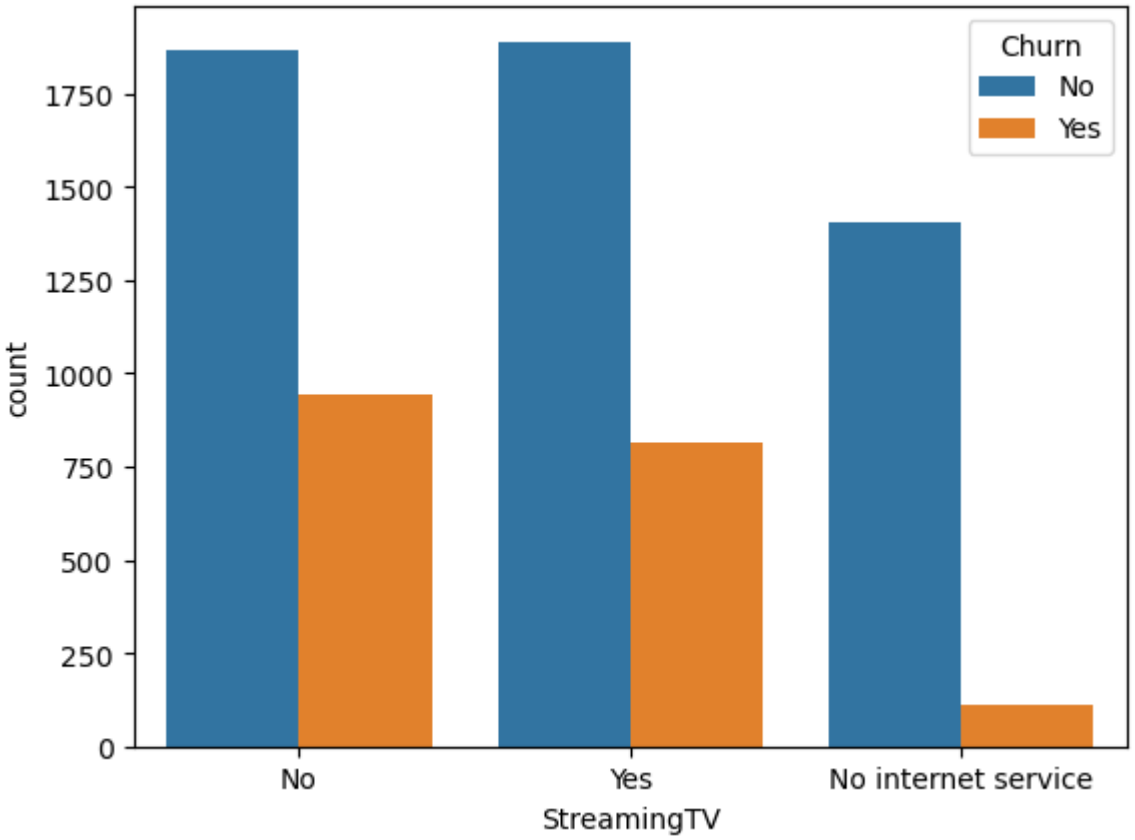

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
DSL
Fiber optic
No
InternetService

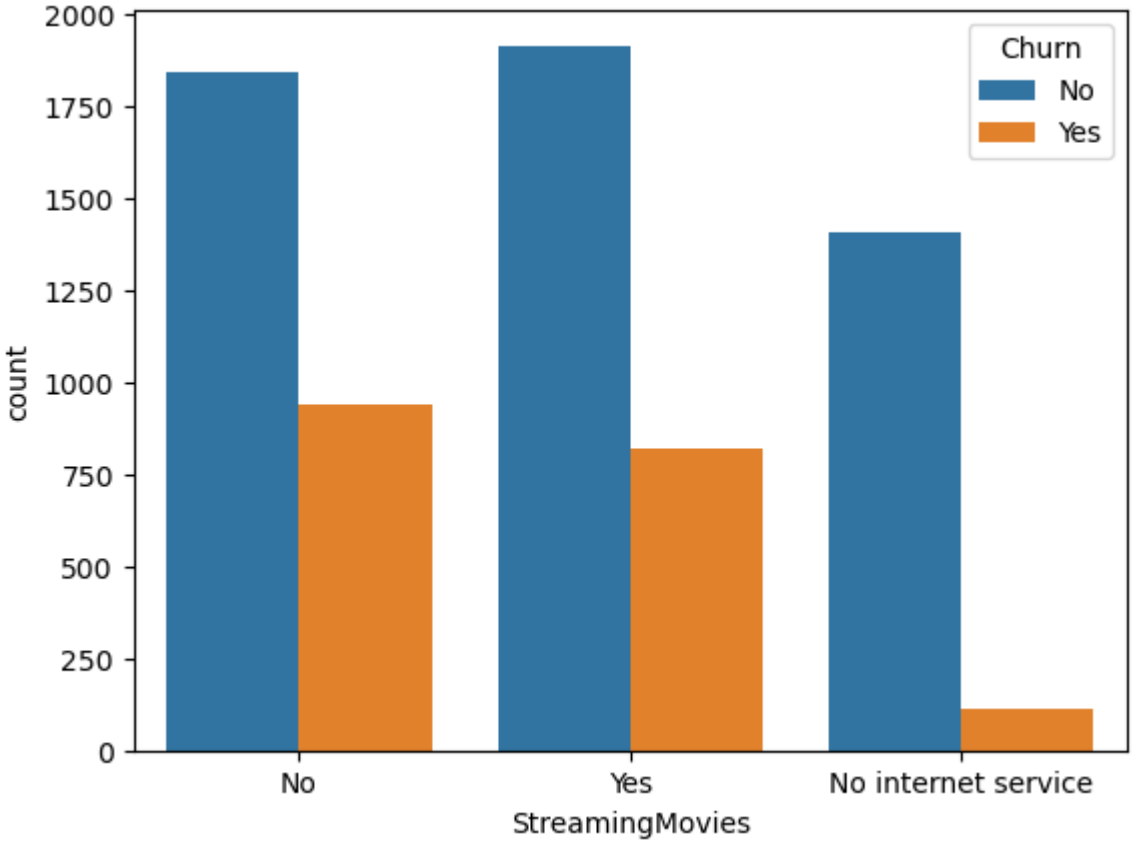

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
OnlineSecurity

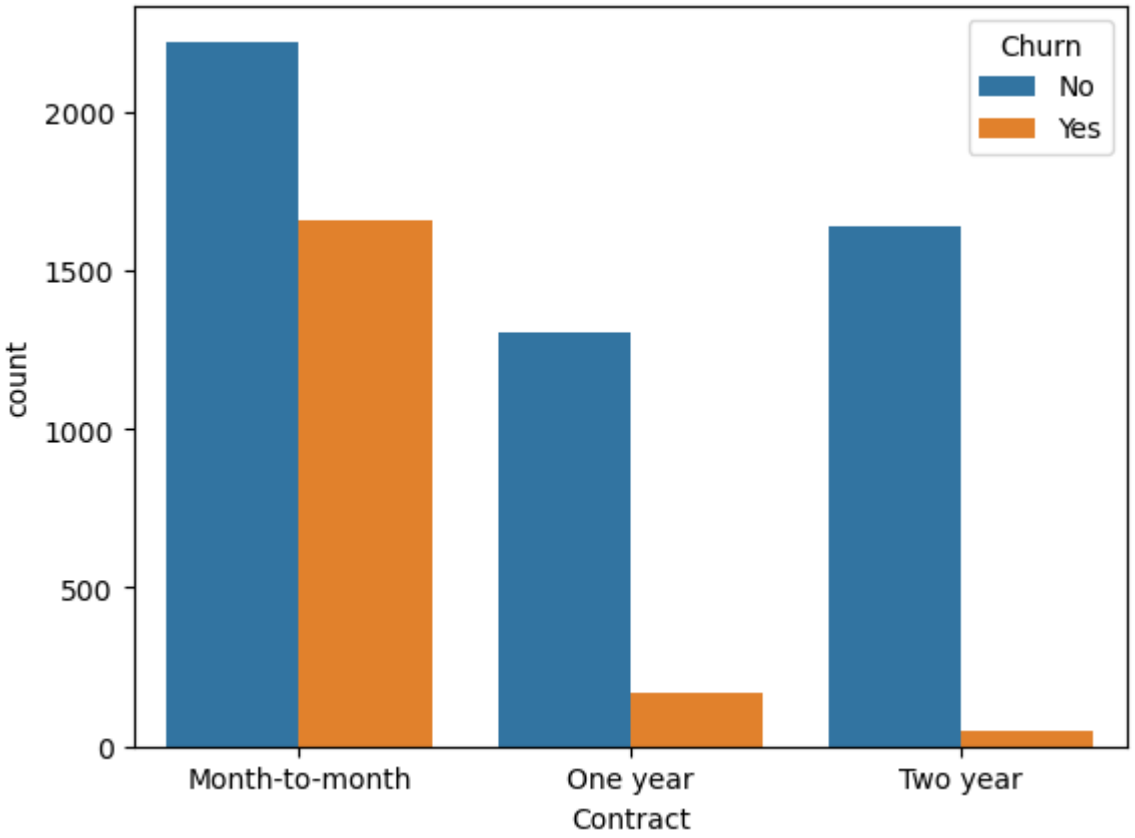

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
Yes
No
No internet service
OnlineBackup

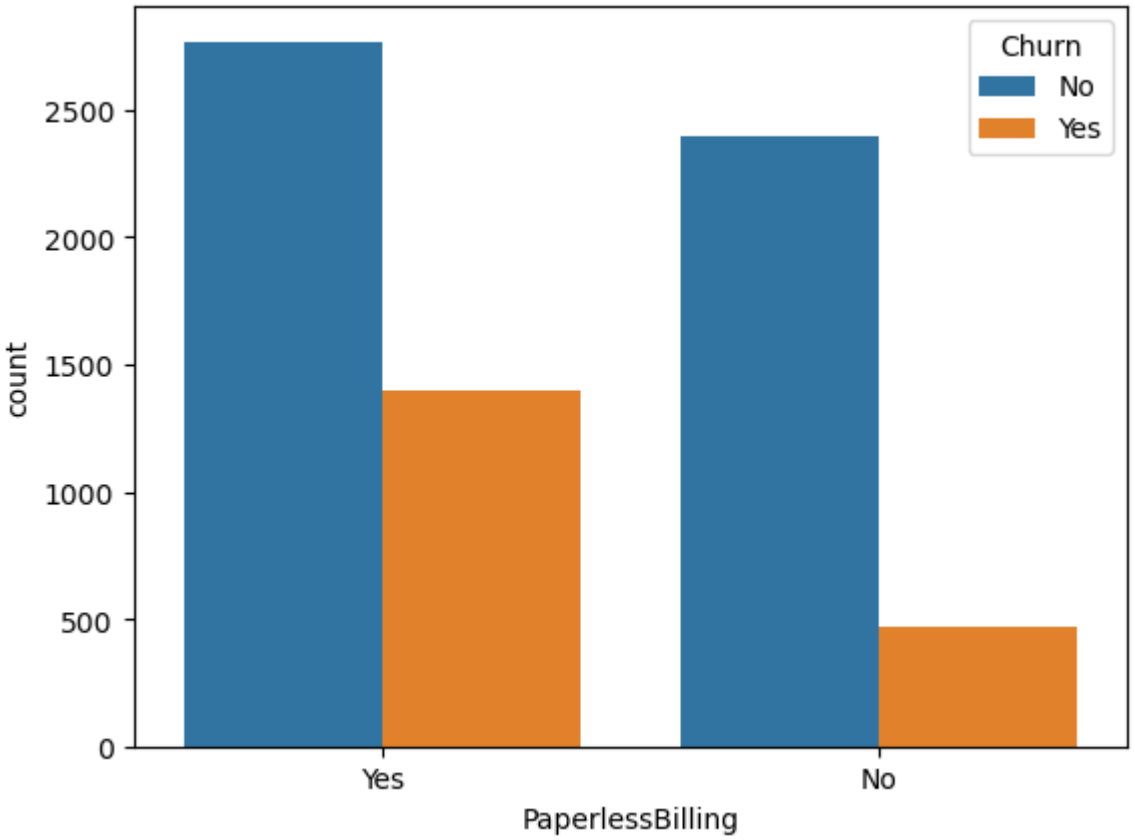

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
DeviceProtection

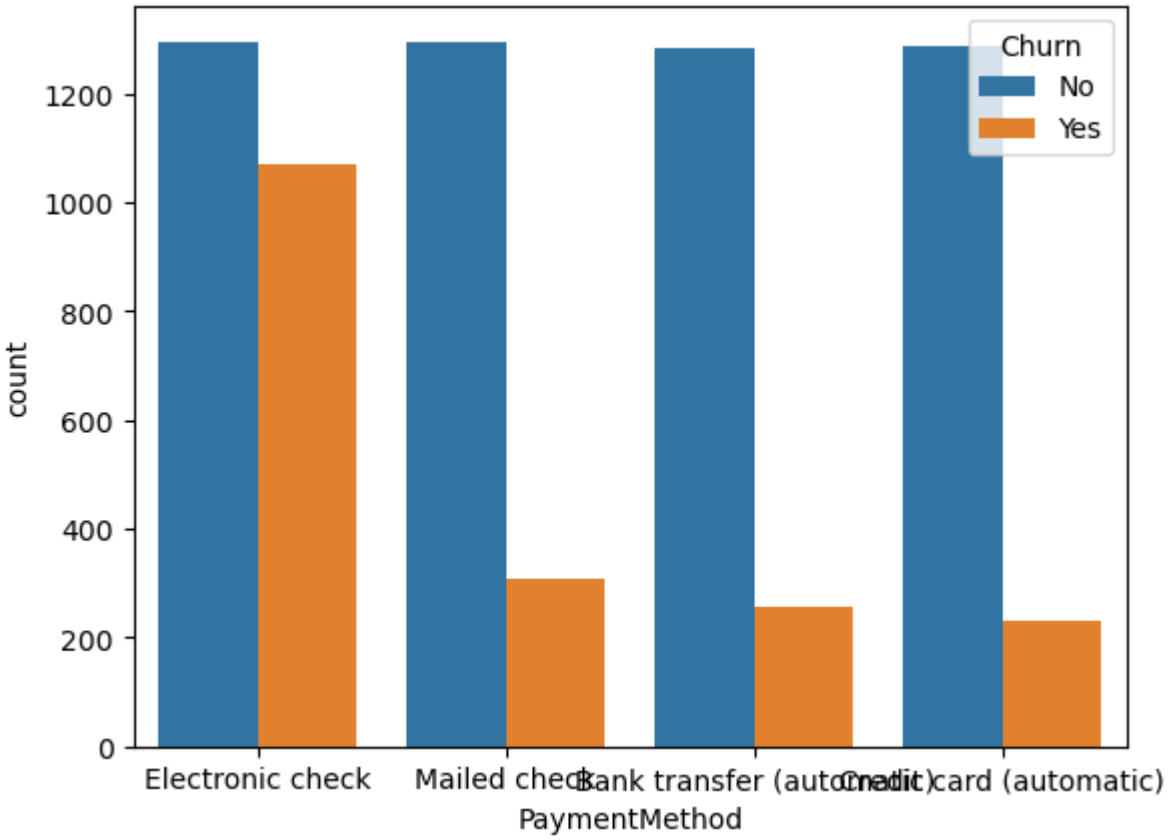

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
TechSupport

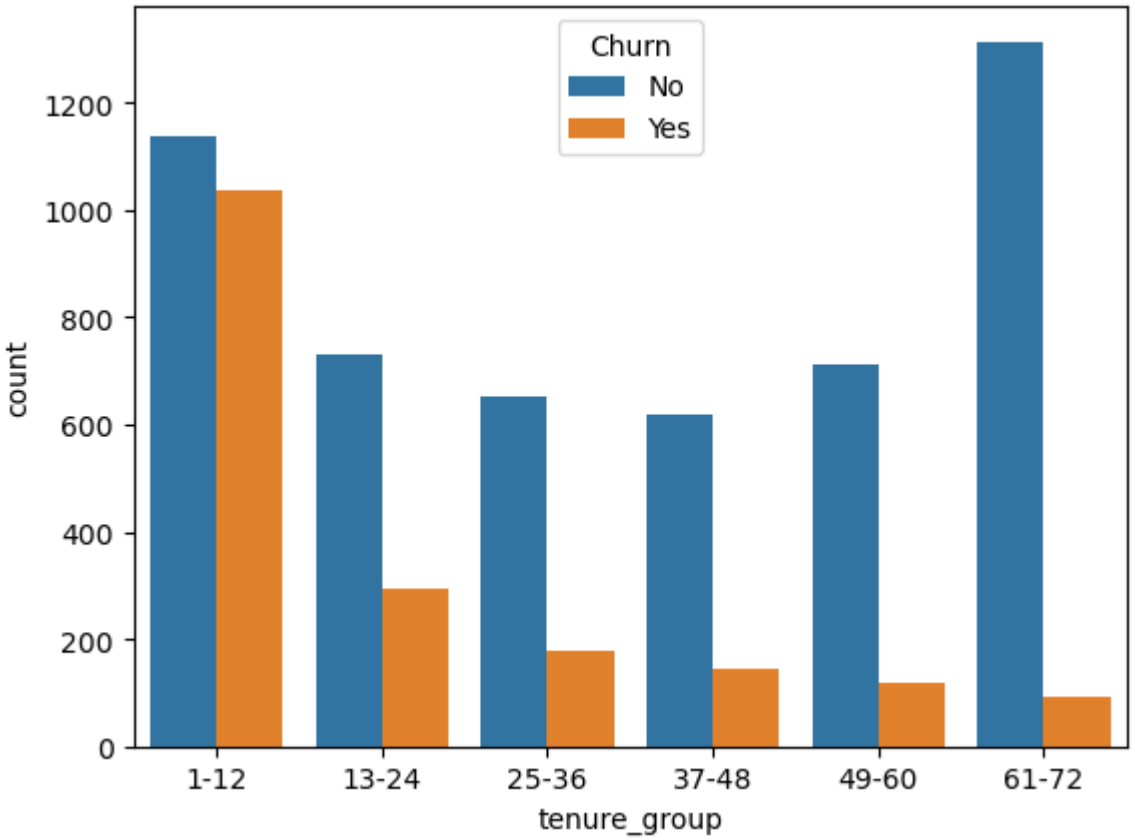

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingTV


Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
StreamingMovies


Churn
No
Yes
2000
1500
1000
500
0
count
Month-to-month
One year
Two year
Contract


Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
PaperlessBilling


Churn
No
Yes
count
0
200
400
600
800
1000
1200
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod



In [22]:
```
#Convert Churn data from Yes or No to Binary Data

data_copy['Churn']=np.where(data_copy.Churn=='Yes',1,0)
```

In [23]:
```
data_copy.head()
```

Out[23]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No

In [24]:
```
data_copy_dummies=pd.get_dummies(data_copy)
data_copy_dummies.head()
```

Out[24]:

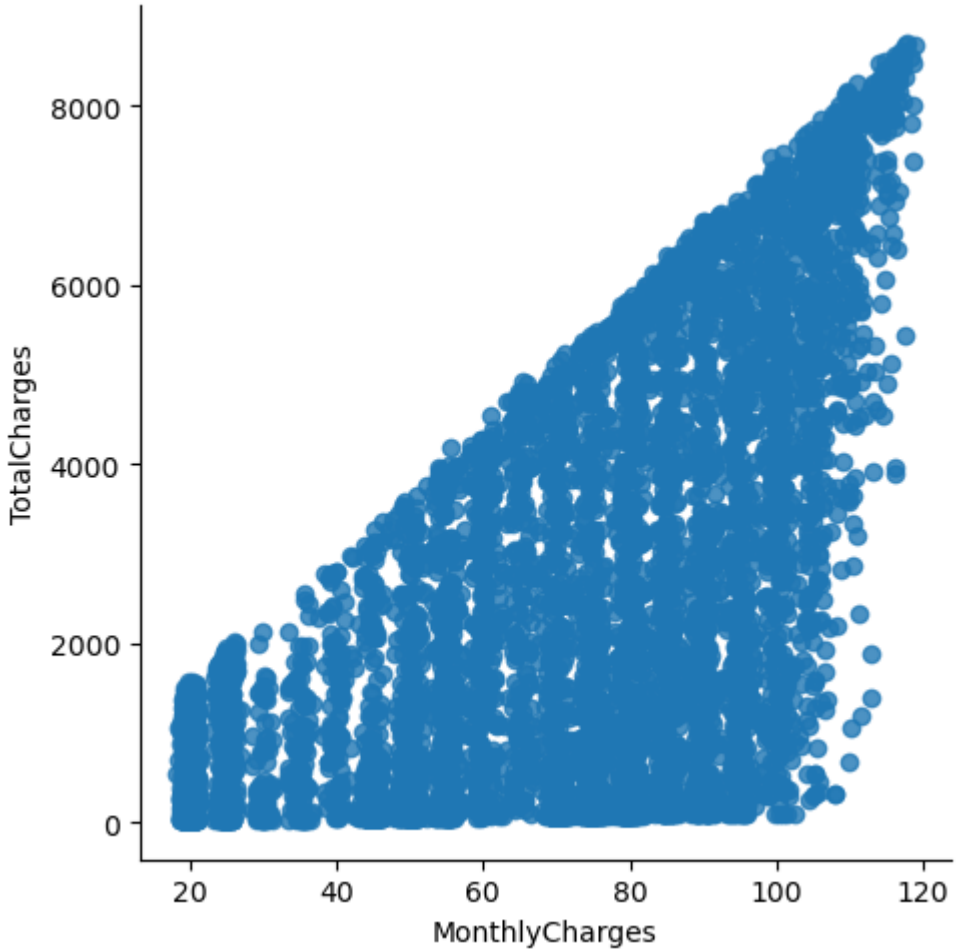
	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Y
0	0	29.85	29.85	0	1	0	0	1	1	
1	0	56.95	1889.50	0	0	1	1	0	1	
2	0	53.85	108.15	1	0	1	1	0	1	
3	0	42.30	1840.75	0	0	1	1	0	1	
4	0	70.70	151.65	1	1	0	1	0	1	

5 rows × 51 columns

Relation between

In [26]:
```
sns.lmplot(data=data_copy_dummies,x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

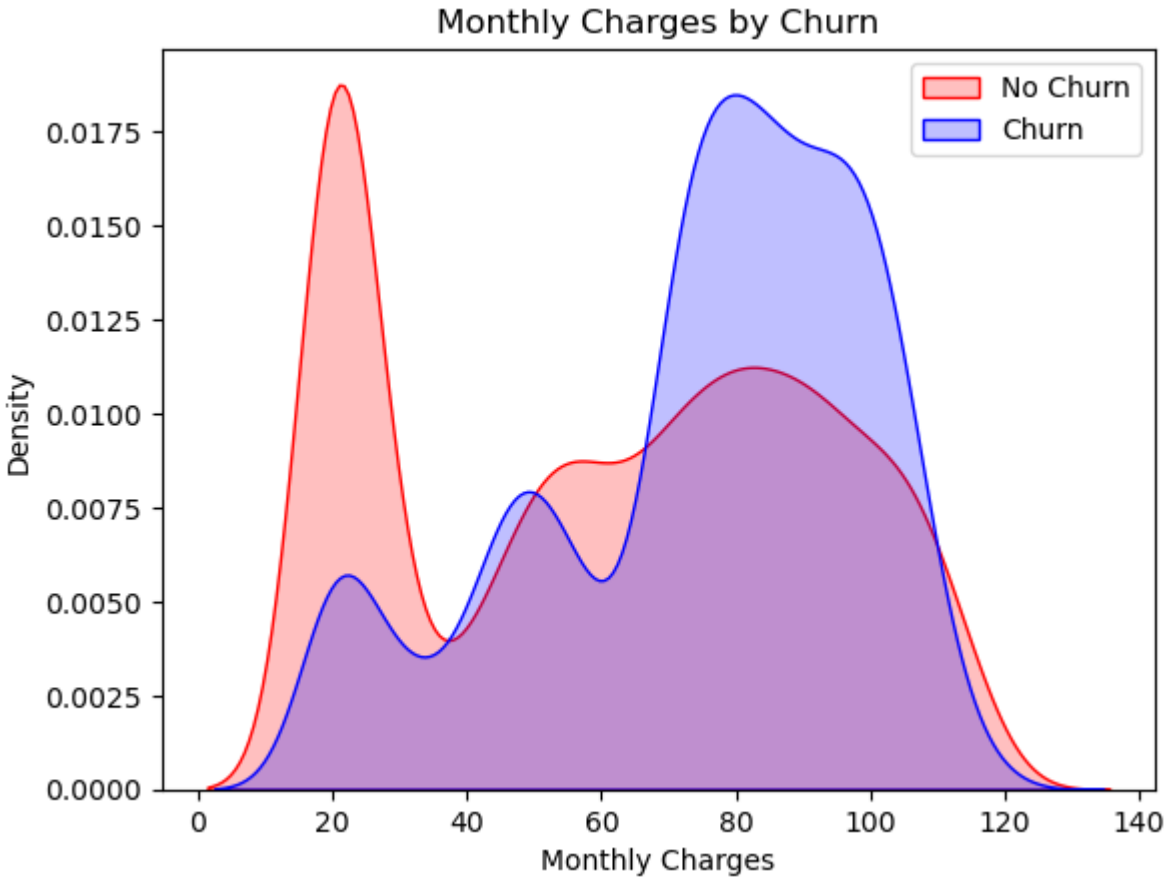
Out[26]:
```
<seaborn.axisgrid.FacetGrid at 0x190c006a6a0>
```



Churn by Monthly Charges and Total charges

In [27]:
```
mth=sns.kdeplot(data_copy_dummies.MonthlyCharges[(data_copy_dummies["Churn"]==0)],color="Red",shade=True)
mth=sns.kdeplot(data_copy_dummies.MonthlyCharges[(data_copy_dummies["Churn"]==1)], ax=mth,color="Blue",shade=True)
mth.legend(["No Churn","Churn"],loc='upper right')
mth.set_ylabel('Density')
mth.set_xlabel('Monthly Charges')
mth.set_title('Monthly Charges by Churn')
```

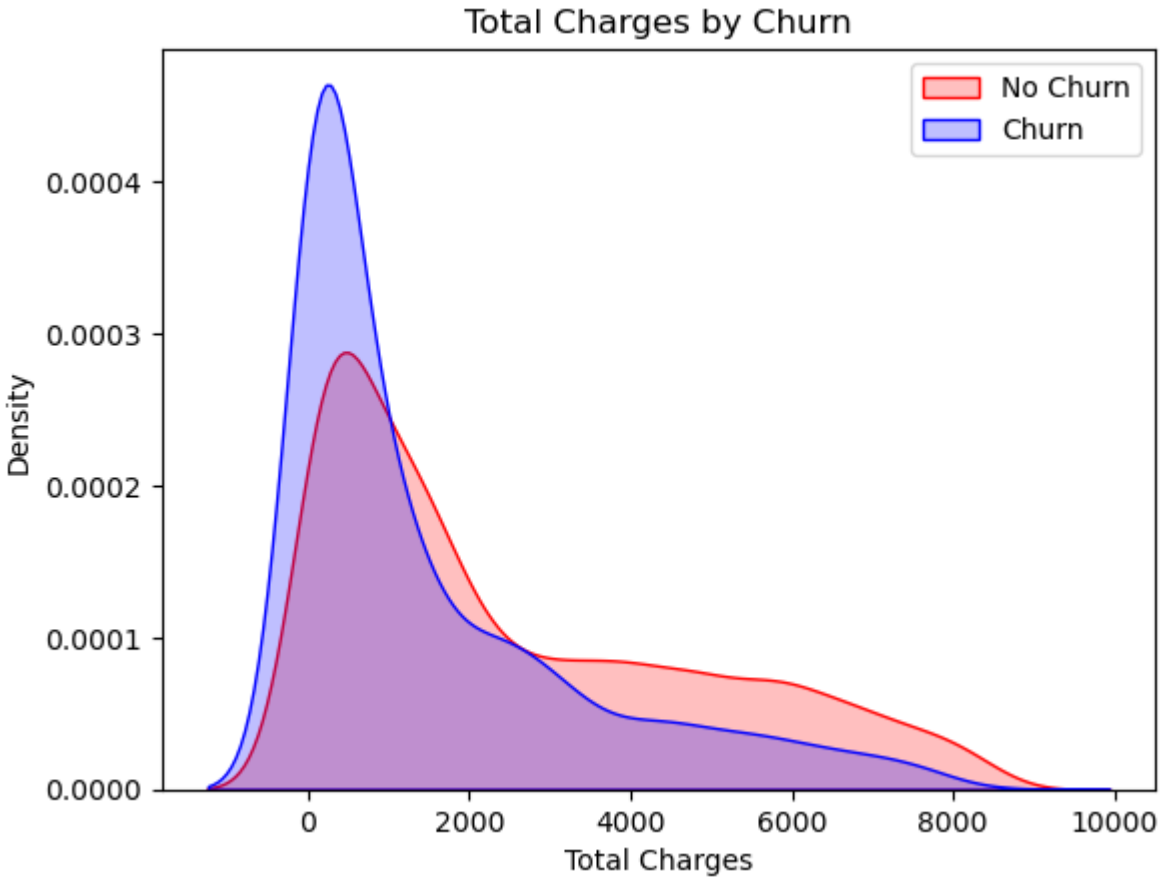
Out[27]:
```
Text(0.5, 1.0, 'Monthly Charges by Churn')
```



Churning is significantly reduced when monthly charges are less and churning increases as monthly charges increases

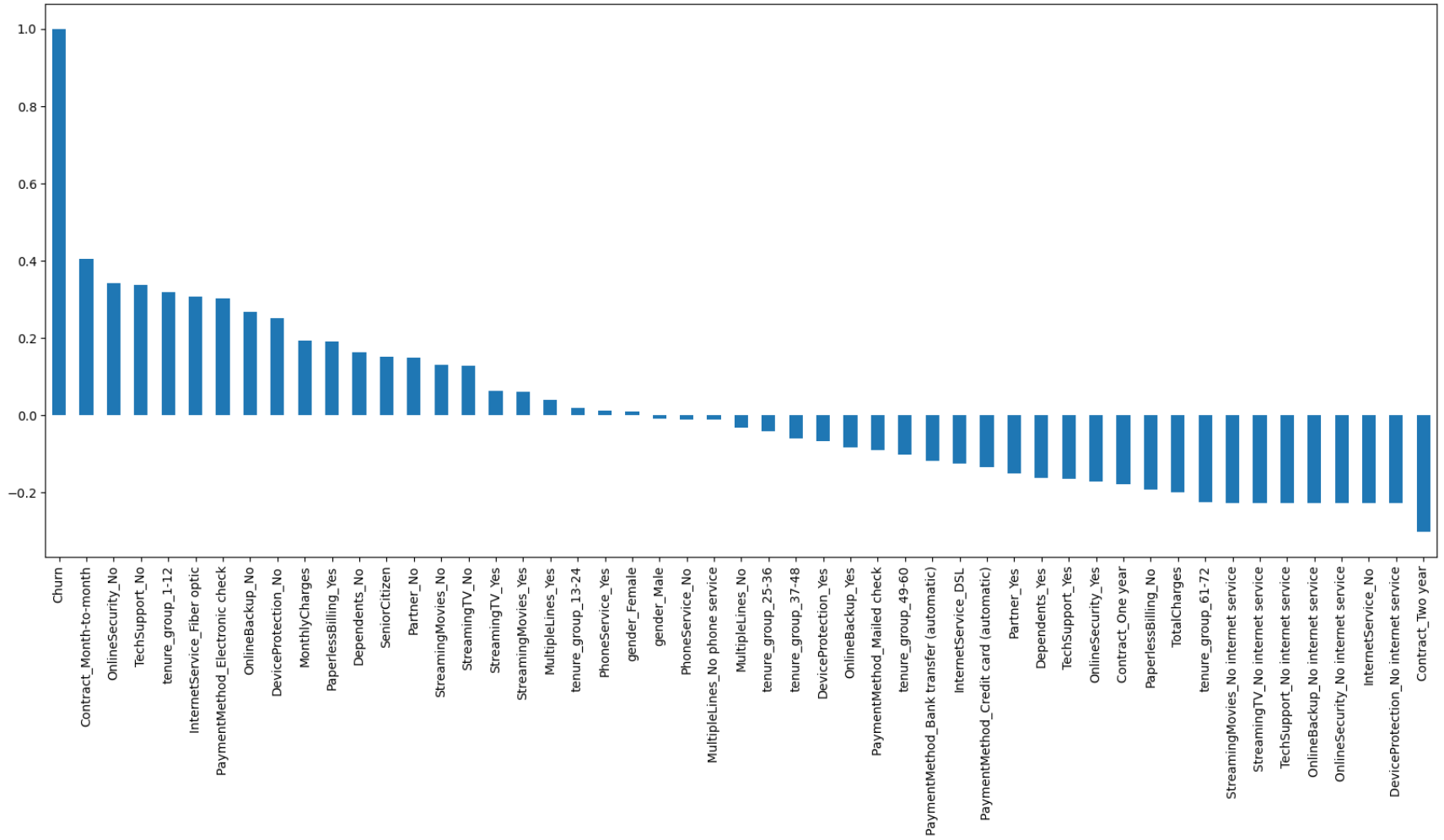
In [28]:
```
tot=sns.kdeplot(data_copy_dummies.TotalCharges[(data_copy_dummies["Churn"]==0)],color="Red",shade=True)
tot=sns.kdeplot(data_copy_dummies.TotalCharges[(data_copy_dummies["Churn"]==1)],ax=tot,color="Blue",shade=True)
tot.legend(["No Churn","Churn"],loc="upper right")
tot.set_ylabel('Density')
tot.set_xlabel('Total Charges')
tot.set_title('Total Charges by Churn')
```

Out[28]:
```
Text(0.5, 1.0, 'Total Charges by Churn')
```



In [29]:
```
plt.figure(figsize=(20,8))
data_copy_dummies.corr()["Churn"].sort_values(ascending=False).plot(kind='bar')
```

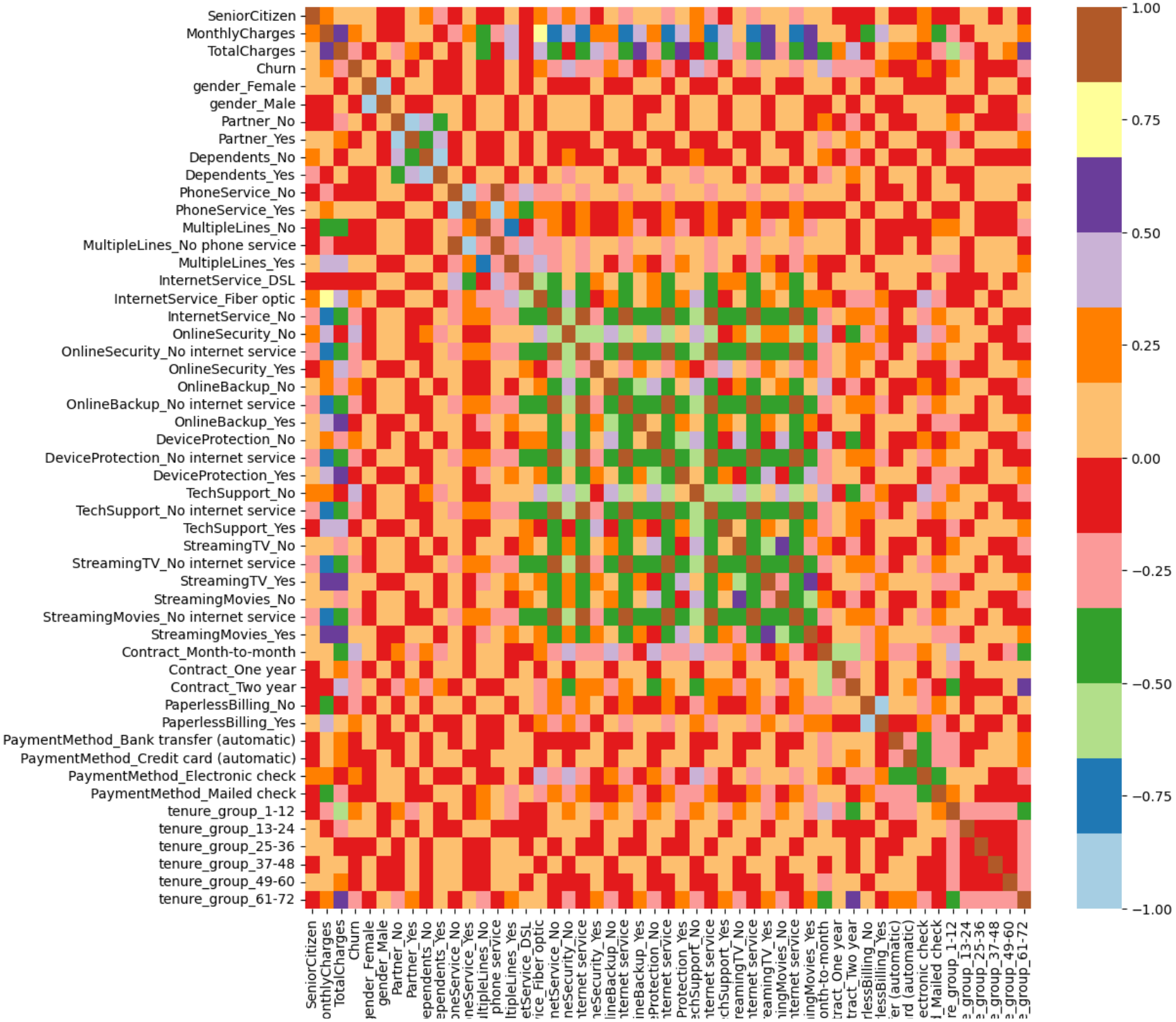
Out[29]:
```
<AxesSubplot:>
```



1. If the contract length is month-to-month then we can see the maximum number of churns where as in case of yearly contract churning is minimum
2. People with no online security or tech-support tends churn more

In [30]:
```
plt.figure(figsize=(12,12))
sns.heatmap(data_copy_dummies.corr(),cmap='Paired')
```

Out[30]:
```
<AxesSubplot:>
```


SeniorCitizen
MonthlyCharges
TotalCharges
Churn
gender_Female
gender_Male
Partner_No
Partner_Yes
Dependents_No
Dependents_Yes
PhoneService_No
PhoneService_Yes
MultipleLines_No
MultipleLines_No phone service
MultipleLines_Yes
InternetService_DSL
InternetService_Fiber optic
InternetService_No
OnlineSecurity_No
OnlineSecurity_No internet service
OnlineSecurity_Yes
OnlineBackup_No
OnlineBackup_No internet service
OnlineBackup_Yes
DeviceProtection_No
DeviceProtection_No internet service
DeviceProtection_Yes
TechSupport_No
TechSupport_No internet service
TechSupport_Yes
StreamingTV_No
StreamingTV_No internet service
StreamingTV_Yes
StreamingMovies_No
StreamingMovies_No internet service
StreamingMovies_Yes
Contract_Month-to-month
Contract_One year
Contract_Two year
PaperlessBilling_No
PaperlessBilling_Yes
PaymentMethod_Bank transfer (automatic)
PaymentMethod_Credit card (automatic)
PaymentMethod_Electronic check
PaymentMethod_Mailed check
tenure_group_1-12
tenure_group_13-24
tenure_group_25-36
tenure_group_37-48
tenure_group_49-60
tenure_group_61-72
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

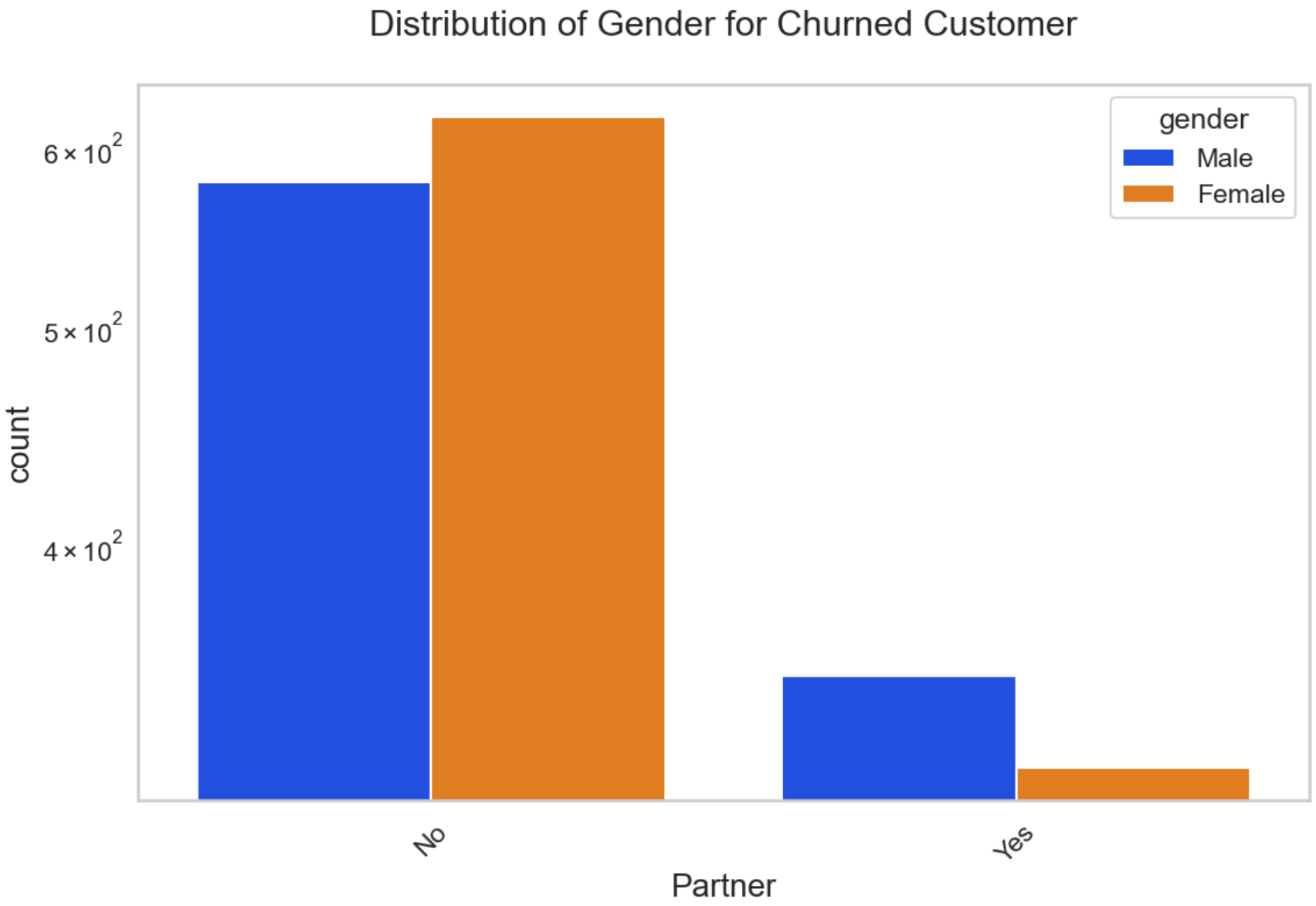
Bivariate Analysis

In [32]:
```
new_df0=data_copy.loc[data_copy['Churn']==0]
new_df1=data_copy.loc[data_copy['Churn']==1]
```

In [35]:
```
def uniplot(df,col,title,hue=None):
    sns.set_style('whitegrid')
    sns.set_context('talk')
    plt.rcParams["axes.labelsize"]=20
    plt.rcParams["axes.titlesize"]=22
    plt.rcParams["axes.titlepad"]=30

    temp=pd.Series(data=hue)
    fig, ax=plt.subplots()
    width=len(df[col].unique()) + 7 + 4*len(temp.unique())
    fig.set_size_inches(width,8)
    plt.xticks(rotation=45)
    plt.yscale('log')
    plt.title(title)
    ax=sns.countplot(data=df,x=col,order=df[col].value_counts().index,hue=hue,palette='bright')
    plt.show()
```

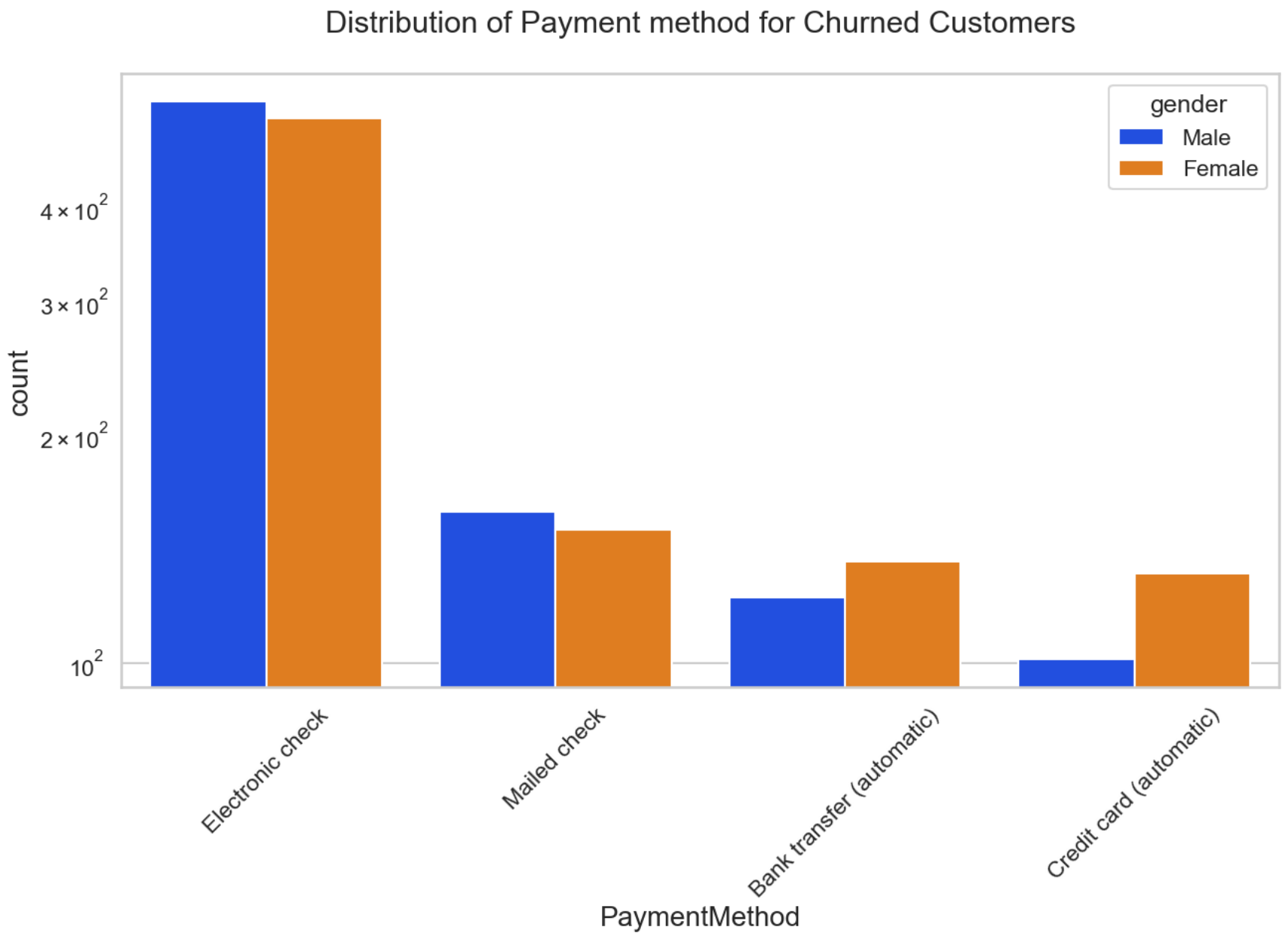
In [36]:
```
uniplot(new_df1,col='Partner',title="Distribution of Gender for Churned Customer",hue='gender')
```



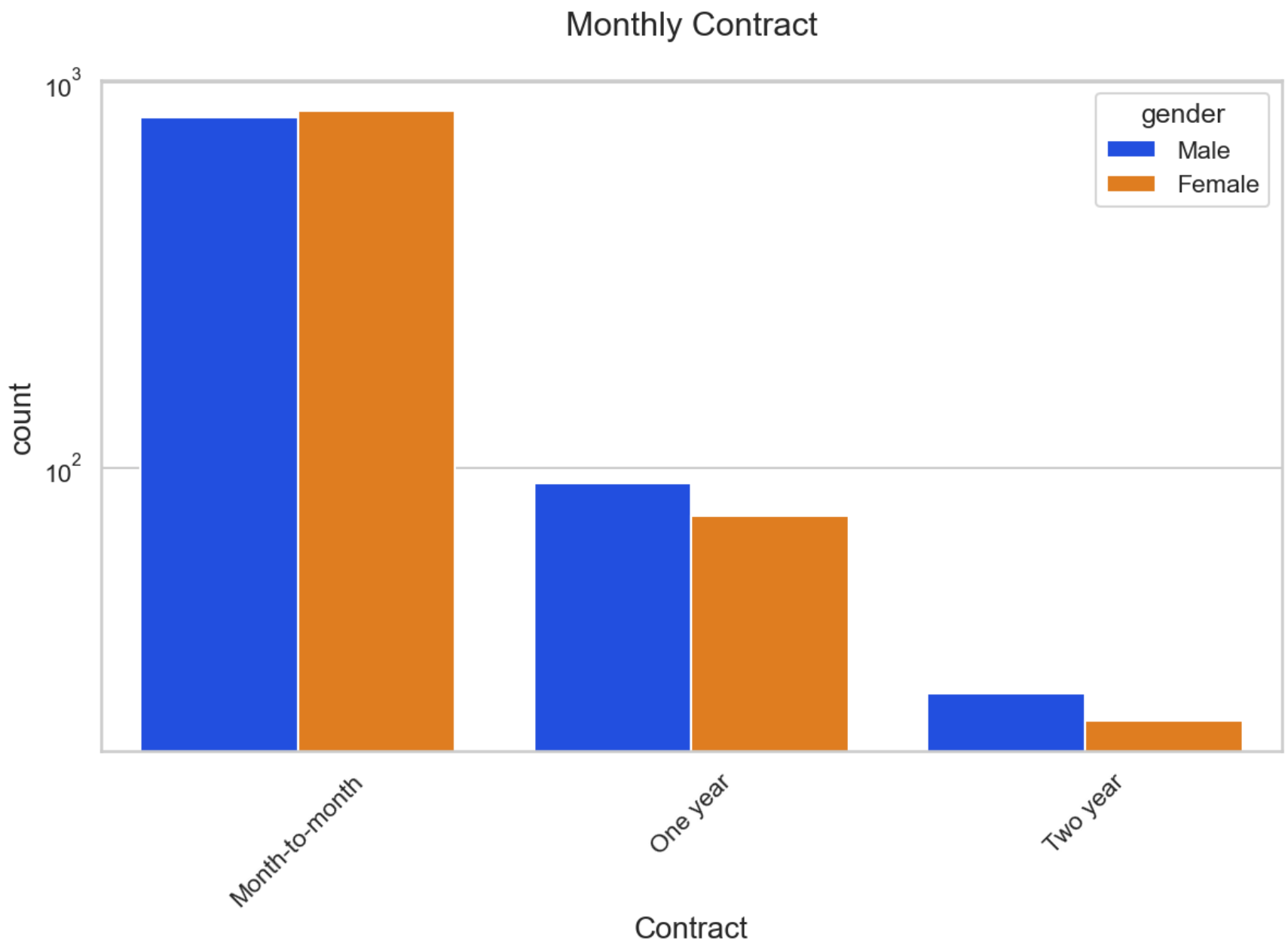
When there is no partner there are 50-50 chance that both male and female will churn

When there is a partner men are most likely to churn

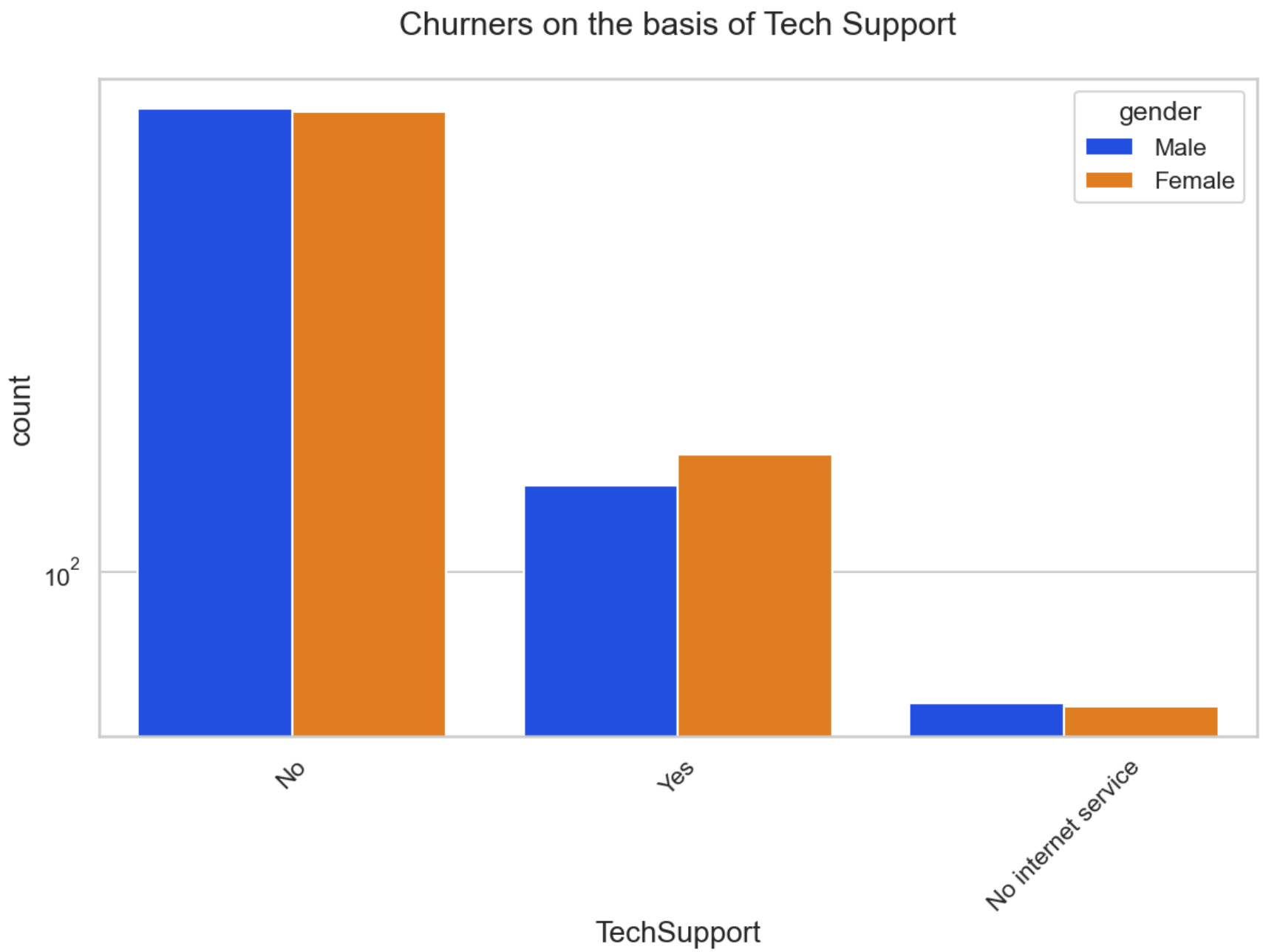
In [37]:
```
uniplot(new_df1,col="PaymentMethod",title="Distribution of Payment method for Churned Customers",hue='gender')
```



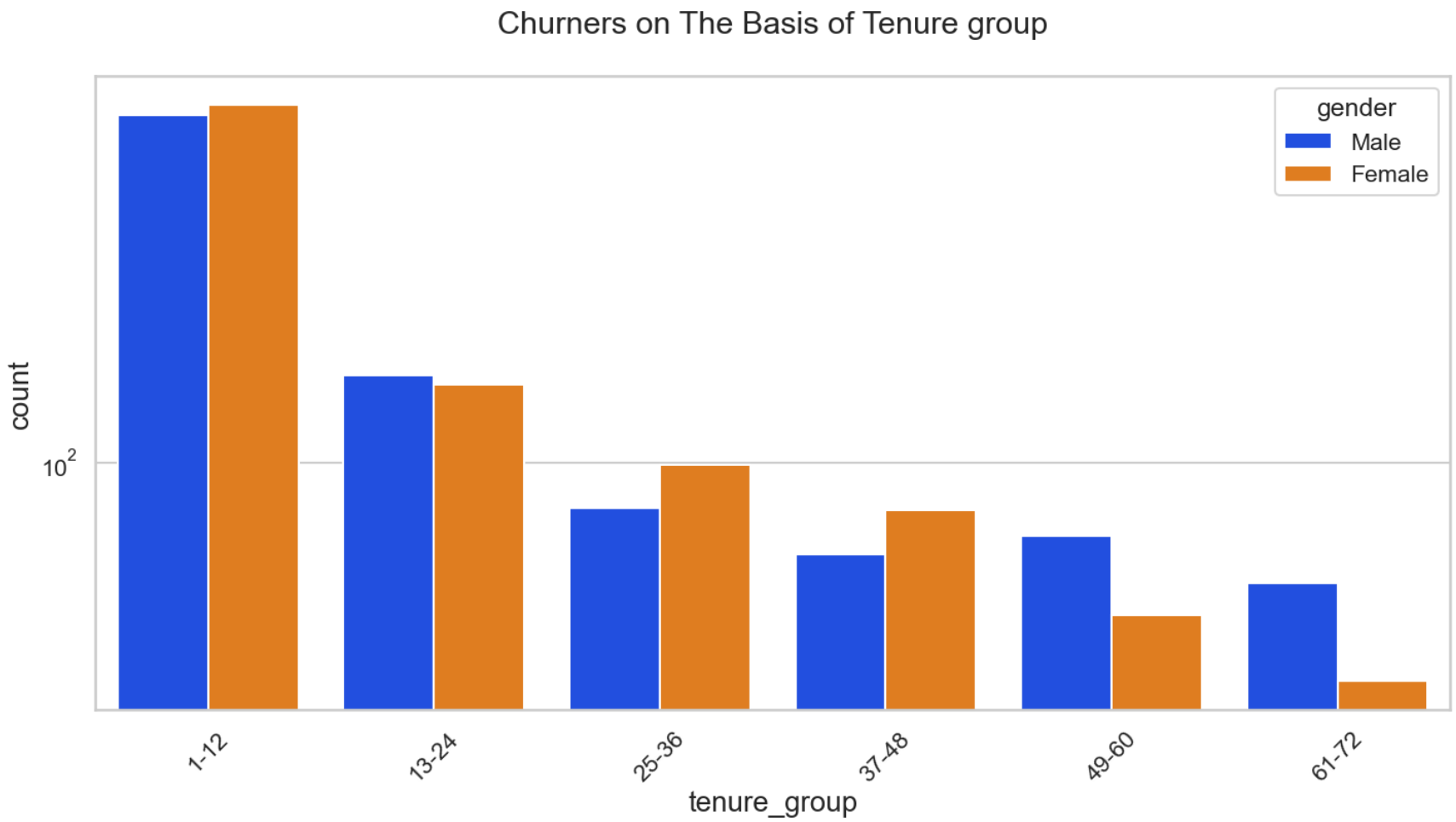
In [38]:
```
uniplot(new_df1,col="Contract",title="Monthly Contract",hue='gender')
```



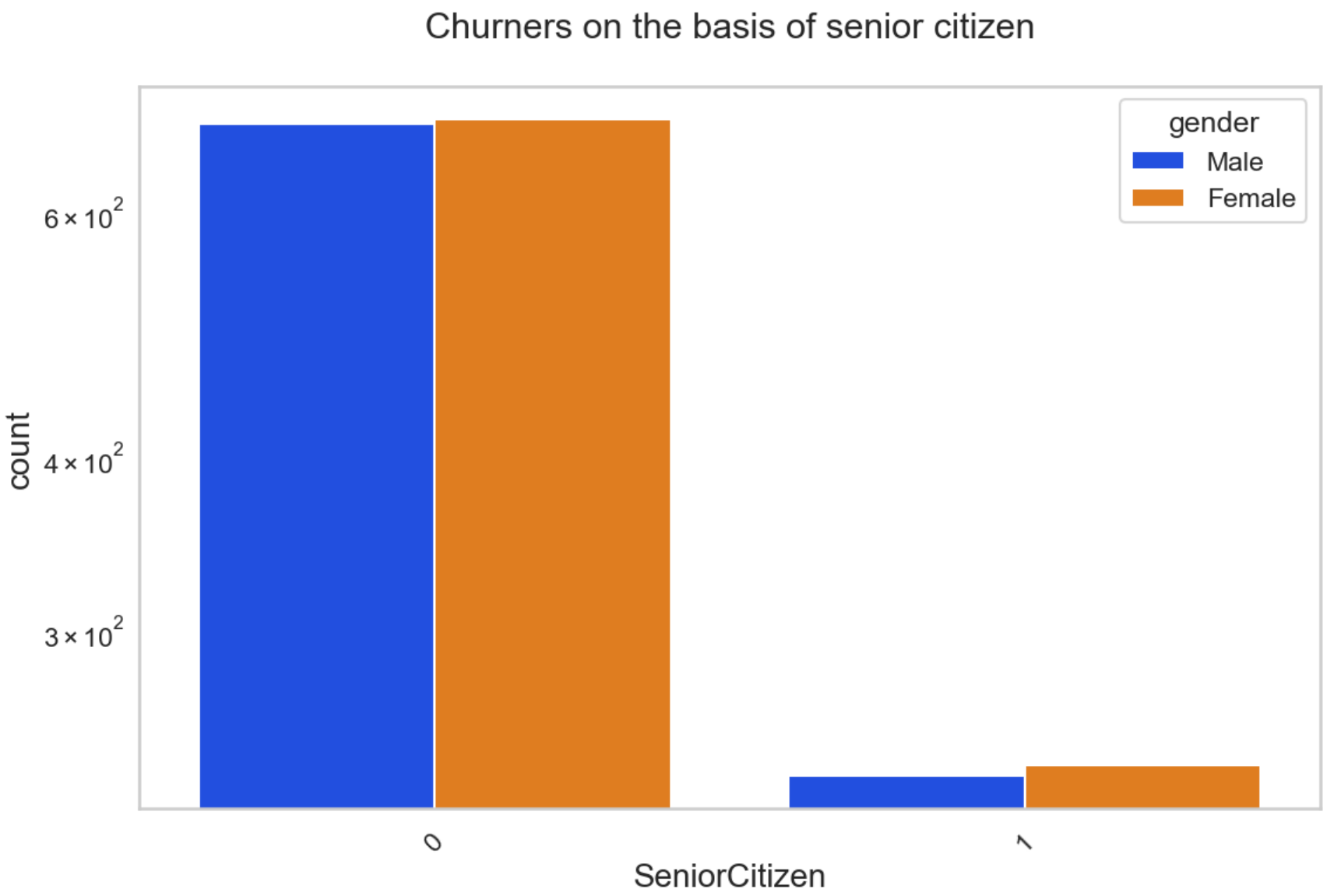
In [40]:
```
uniplot(new_df1,col='TechSupport',title="Churners on the basis of Tech Support",hue='gender')
```



In [41]:
```
uniplot(new_df1,col="tenure_group",title="Churners on The Basis of Tenure group",hue='gender')
```



In [42]:
```
uniplot(new_df1,col='SeniorCitizen',title="Churners on the basis of senior citizen",hue='gender')
```



Conclusion

There are some Usefull insights from the analysis:

1. Electronic Checks are the highest churners

2. Contract:- Monthly customers are more likely to churn as they have no contract and they are free to go
3. No Tech Support are high churners
4. The younger generation is more likely to churn or we can say that they are the highest churners

In [43]:
```
data_copy_dummies.to_csv('tel_churn.csv')
```

In [ ]:
```

```