Telco Churn Analysis

Dataset Info: Sample Data Set containing Telco customer data and showing customers left last month

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
%matplotlib inline
```

In [4]:
```
telco_base_data = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn (1) (1).csv")
telco_base_data
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLin
0	7590-VHVEG	Female	0	Yes	No	1	No	No phor servi
1	5575-GNVDE	Male	0	No	No	34	Yes	N
2	3668-QPYBK	Male	0	No	No	2	Yes	N
3	7795-CFOCW	Male	0	No	No	45	No	No phor servi
4	9237-HQITU	Female	0	No	No	2	Yes	N
...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Y
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Y
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phor servi
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Y
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

7043 rows × 21 columns

In [5]:
```
telco_base_data.head()
```

Out[5]:

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

5 rows × 21 columns

In [6]:
```
telco_base_data.shape
```

Out[6]:
```
(7043, 21)
```

In [7]:
```
telco_base_data.columns.values
```

Out[7]:
```
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 [8]:
```
#checking the datatype of all the columns
telco_base_data.dtypes
```

Out[8]:
```
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 [9]:
```
#check the descriptive statastic of  numeric values
telco_base_data.describe()
```

Out[9]:

	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

1) SeniorCitizen is actually a categorical hence the 25%-50%-75% distribution is not proper

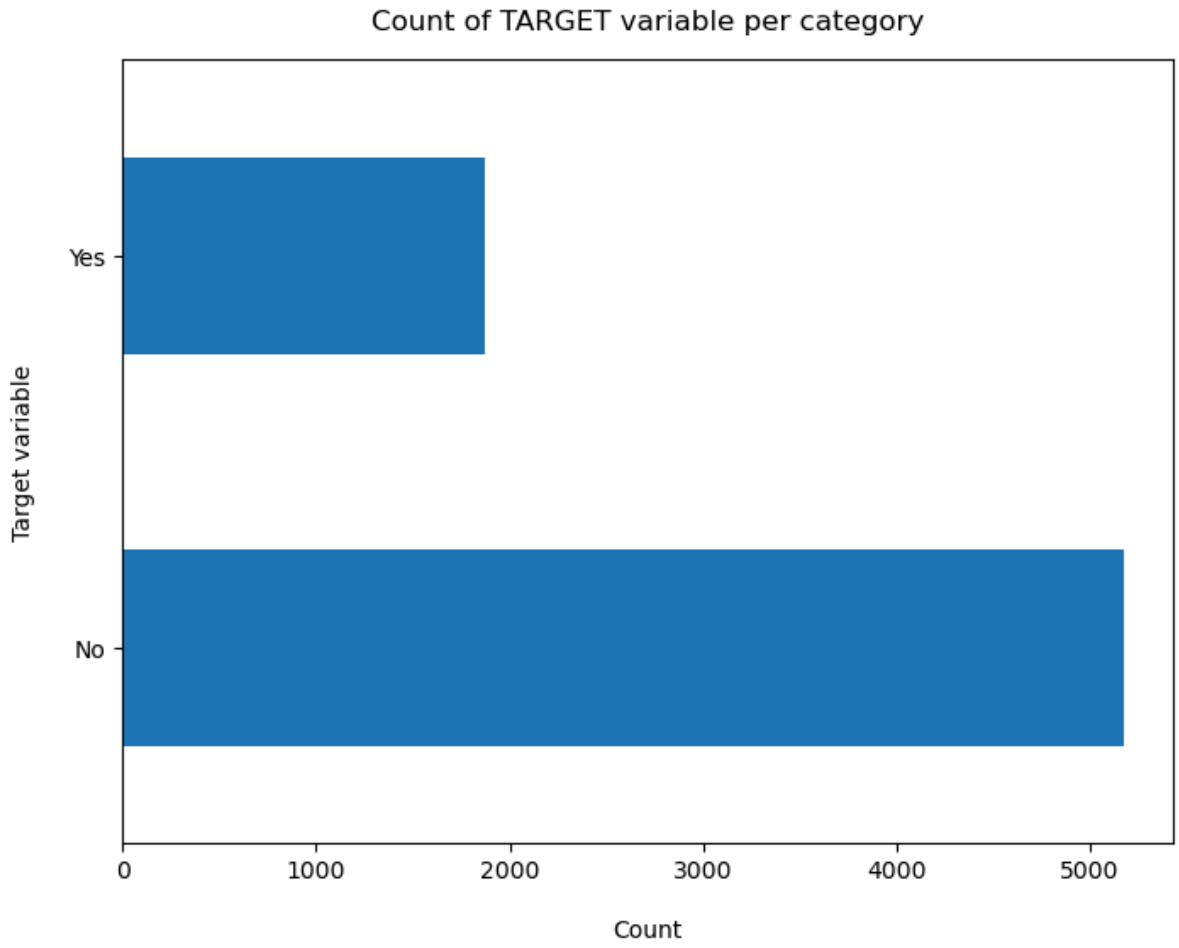
2) 75% customers have tenure less than 55 months

3) Average Monthly charges are USD 64.76 whereas 25% customers pay more than USD 89.85 per month

In [10]:
```
telco_base_data["Churn"].value_counts()
```

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

In [11]:
```
telco_base_data["Churn"].value_counts().plot(kind="barh", figsize=(8,6))
plt.xlabel("Count", labelpad=15)
plt.ylabel("Target variable", labelpad=14)
plt.title("Count of TARGET variable per category",y=1.02);

#labelpad = it is use give axis and box spacing
```



In [12]:
```
# 100*5174/7043 ==  no
# 1869/7043*100 == yes

#100* (5174/1869) / 7043

100*telco_base_data['Churn'].value_counts() / len(telco_base_data['Churn'])
```

Out[12]:
```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

- Data is highly imbalanced, ratio = 73:27
- So we analyse the data with other features while taking the target values separately to get some insights.

In [13]:
```
#Summary of the dataframe, as we have too many columns
telco_base_data.info()
```

```
<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 [14]:
```
telco_base_data.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        0
Churn               0
dtype: int64
```

In [ ]:
```

```

Data cleaning

1. Create a copy of base data for manupulation & processing

In [15]:
```
telco_data = telco_base_data.copy()
```

2. Total Charges should be numeric amount. Let's convert it to numerical data type

In [16]:
```
telco_data.TotalCharges = pd.to_numeric(telco_data.TotalCharges, errors='coerce')
telco_data.isnull().sum()
#Convert argument to a numeric type.
#errors{‘ignore’, ‘raise’, ‘coerce’}, default ‘raise’
```

Out[16]:
```
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
```

In [17]:
```
telco_data.info()
```

```
<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      7032 non-null   float64
 20  Churn             7043 non-null   object 
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

In [18]:
```
telco_base_data.loc[488:489]
```

Out[18]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service
489	8372-JUXUI	Male	0	No	Yes	1	Yes	Yes

2 rows × 21 columns

3. As we can see there are 11 missing values in TotalCharges column. Let's check these records

In [ ]:

```
```

4. Missing Value Treatement

In [19]:

```
#Removing missing values
telco_data.dropna(how = 'any', inplace = True)
```

In [20]:

```
telco_data.head(5)
```

Out[20]:

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

5 rows × 21 columns

5. Divide customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on...

In [21]:

```
#Get the max tenure
print( telco_data['tenure'].max())
```

```
72
```

In [22]:

```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]   # list compres

telco_data['tenure_group'] = pd.cut(telco_data.tenure, range(1, 80, 12), right=Fals
```

```
#Bin values into discrete intervals.

#Use cut when you need to segment and sort data values into bins.
#This function is also useful for going from a continuous variable to a
#categorical variable. For example, cut could convert ages to groups of
#age ranges. Supports binning into an equal number of bins,
#or a pre-specified array of bins.
```

In [23]:
```
telco_data['tenure_group'].value_counts()
```

Out[23]:
```
1 - 12     2175
61 - 72    1407
13 - 24    1024
25 - 36     832
49 - 60     832
37 - 48     762
Name: tenure_group, dtype: int64
```

6. Remove column not required for the preprocessing

In [24]:
```
#drop column 'customerID','tenure'  they are giving any insights
telco_data.drop(columns = ['customerID','tenure'], axis = 1, inplace=True)
```

In [25]:
```
telco_data.head()
```

Out[25]:

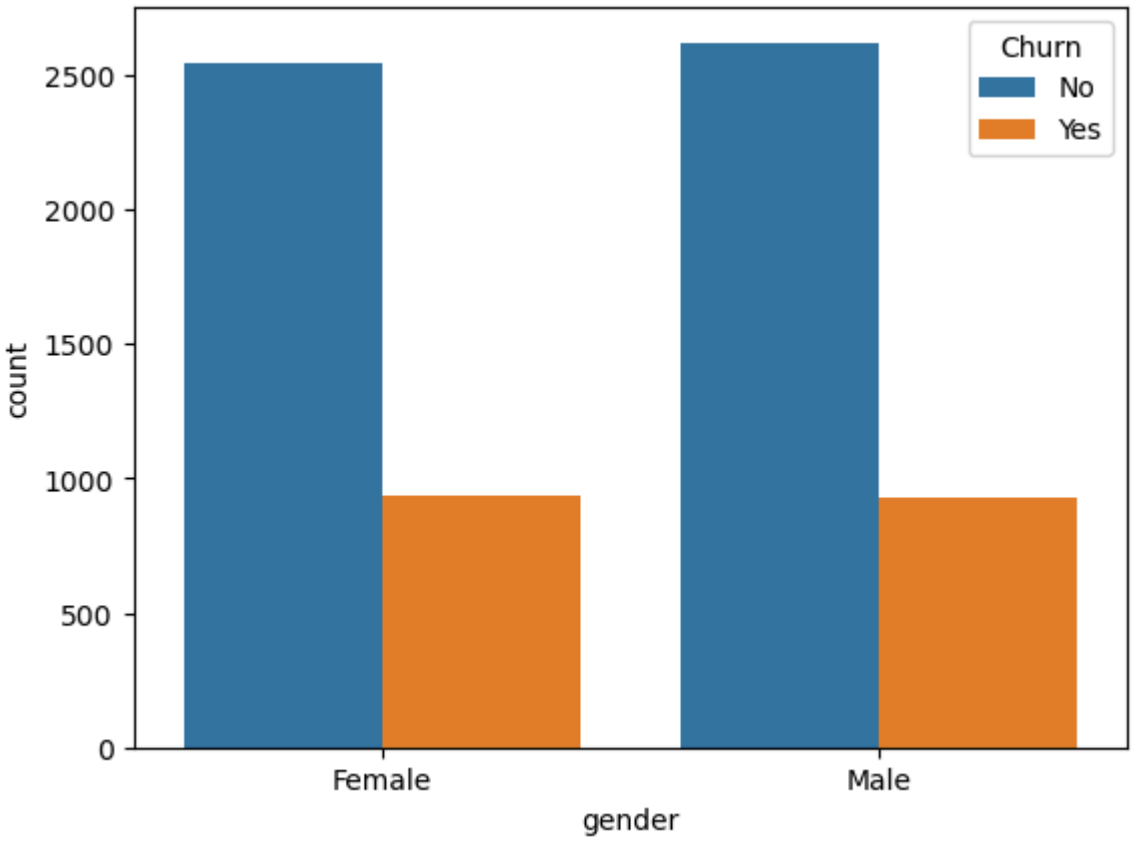
	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	Onli
0	Female	0	Yes	No	No	No phone service	DSL	
1	Male	0	No	No	Yes	No	DSL	
2	Male	0	No	No	Yes	No	DSL	
3	Male	0	No	No	No	No phone service	DSL	
4	Female	0	No	No	Yes	No	Fiber optic	

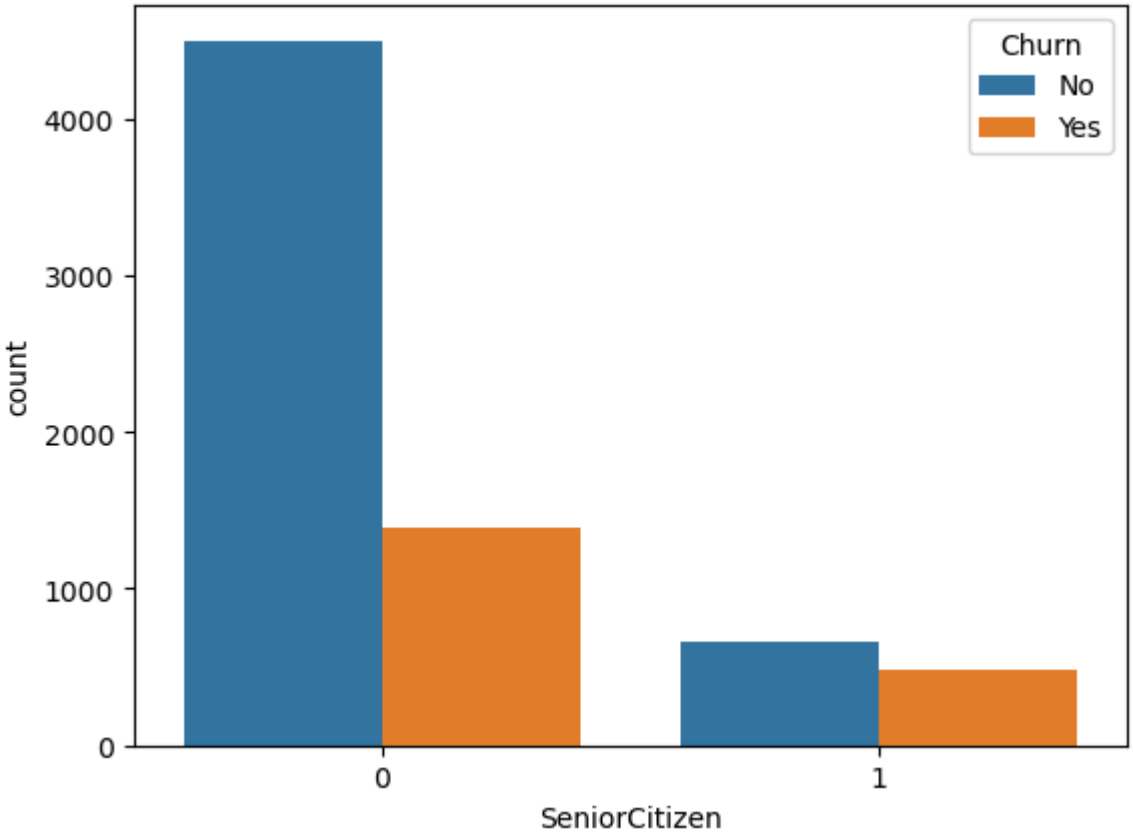
Data Exploration

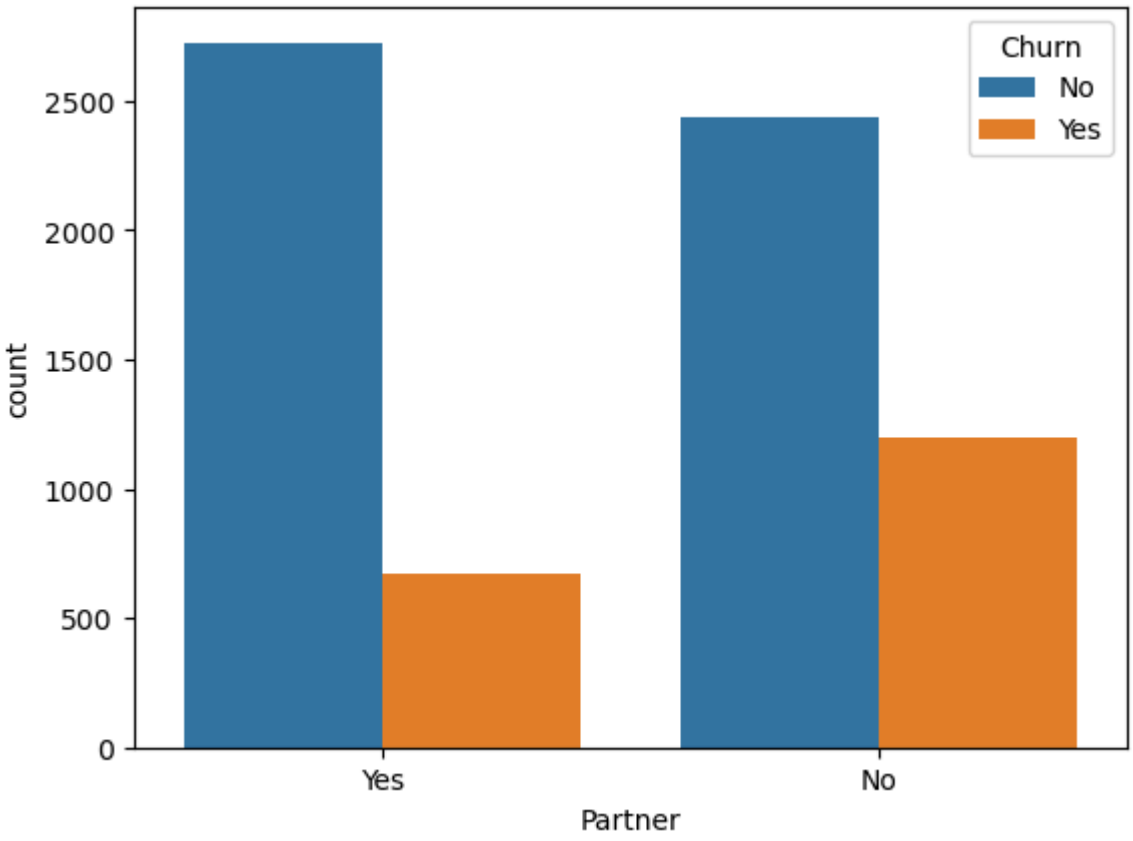
1. Plot distibution of individual predictors by churn

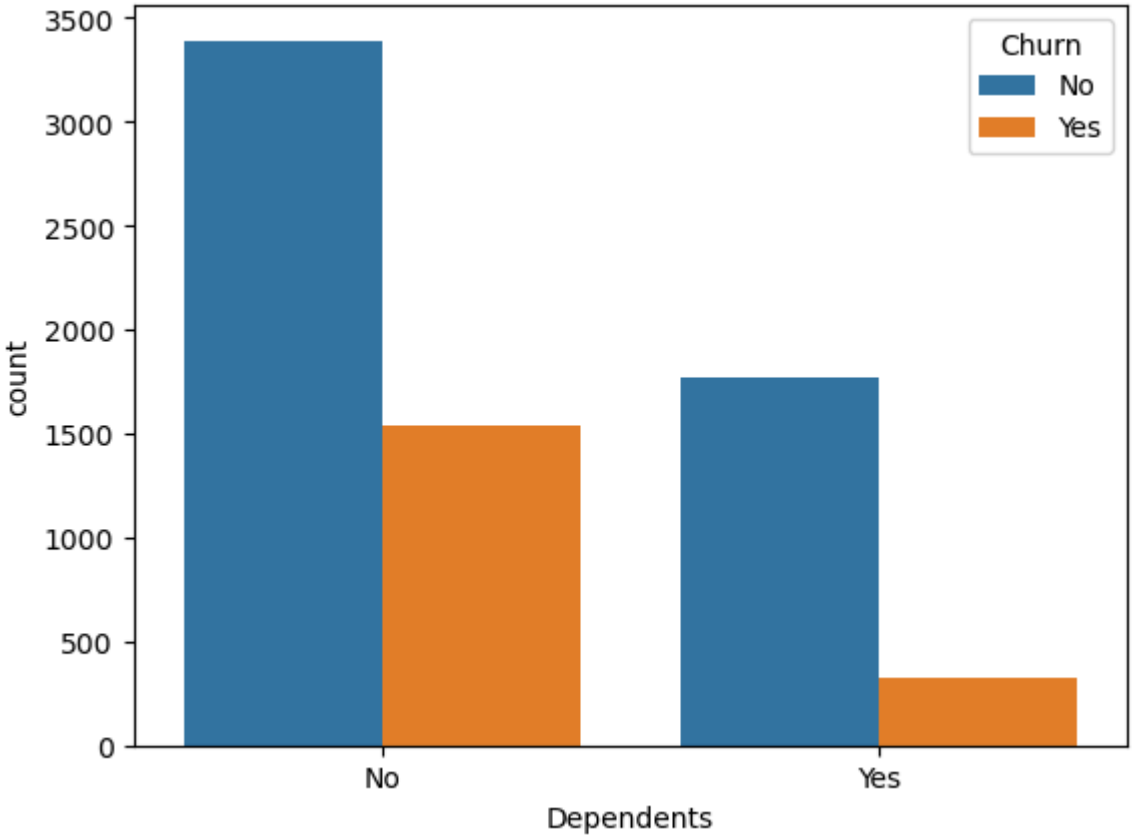
Univariate Analysis

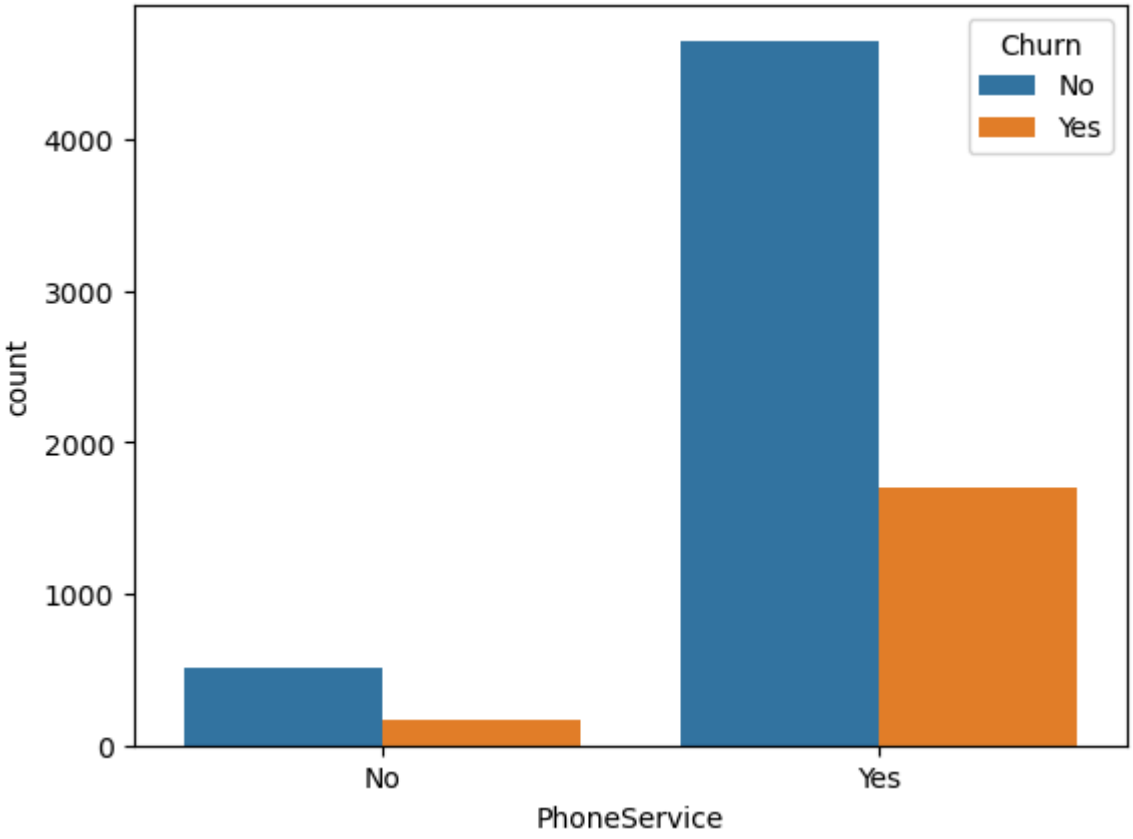
In [27]:
```
for i, predictor in enumerate(telco_data.drop(columns=['Churn','TotalCharges','Mont
    plt.figure(i)
    sns.countplot(data=telco_data, x=predictor, hue="Churn")
```

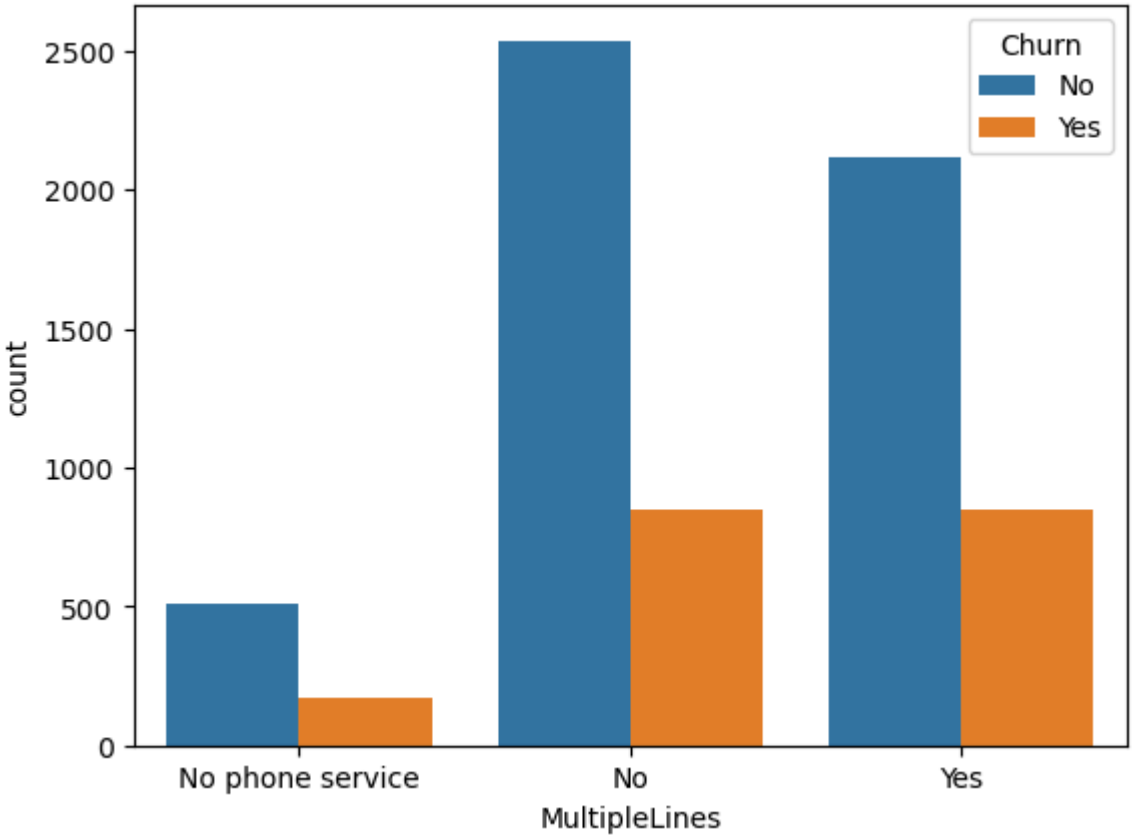

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Female
Male
gender

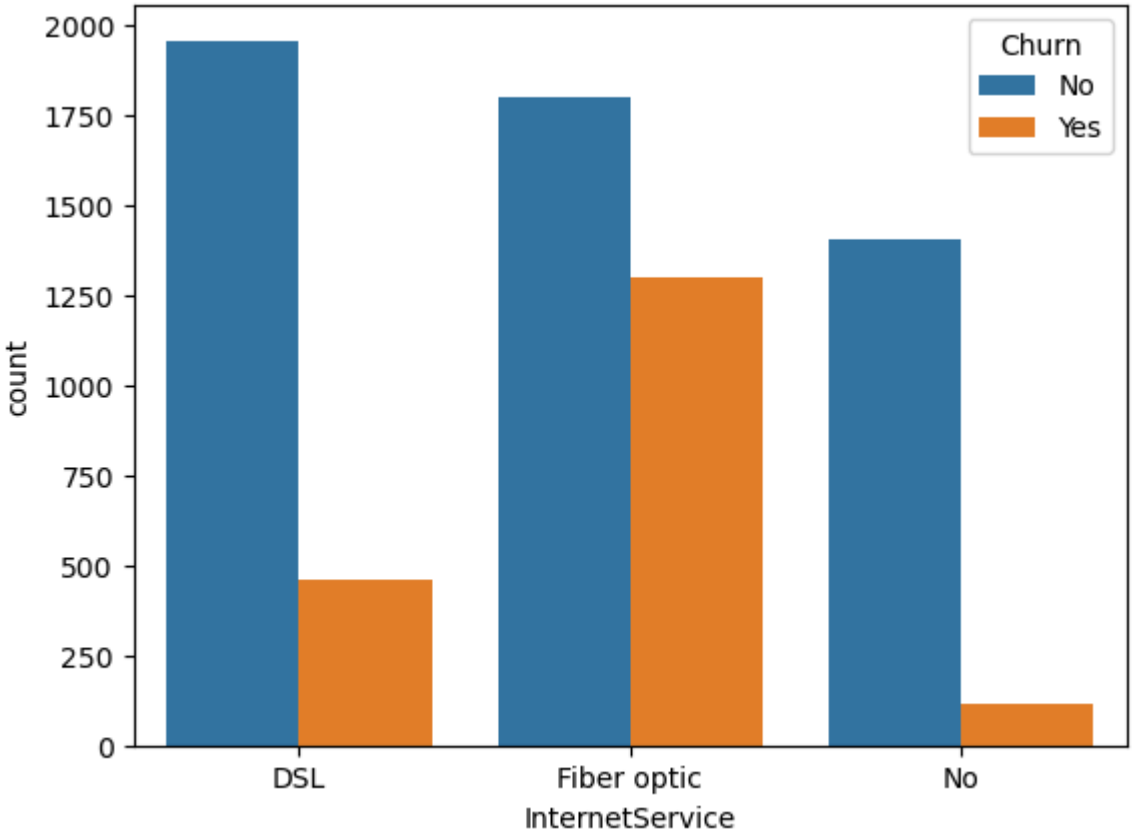

Churn
No
Yes
4000
3000
2000
1000
0
count
0
1
SeniorCitizen

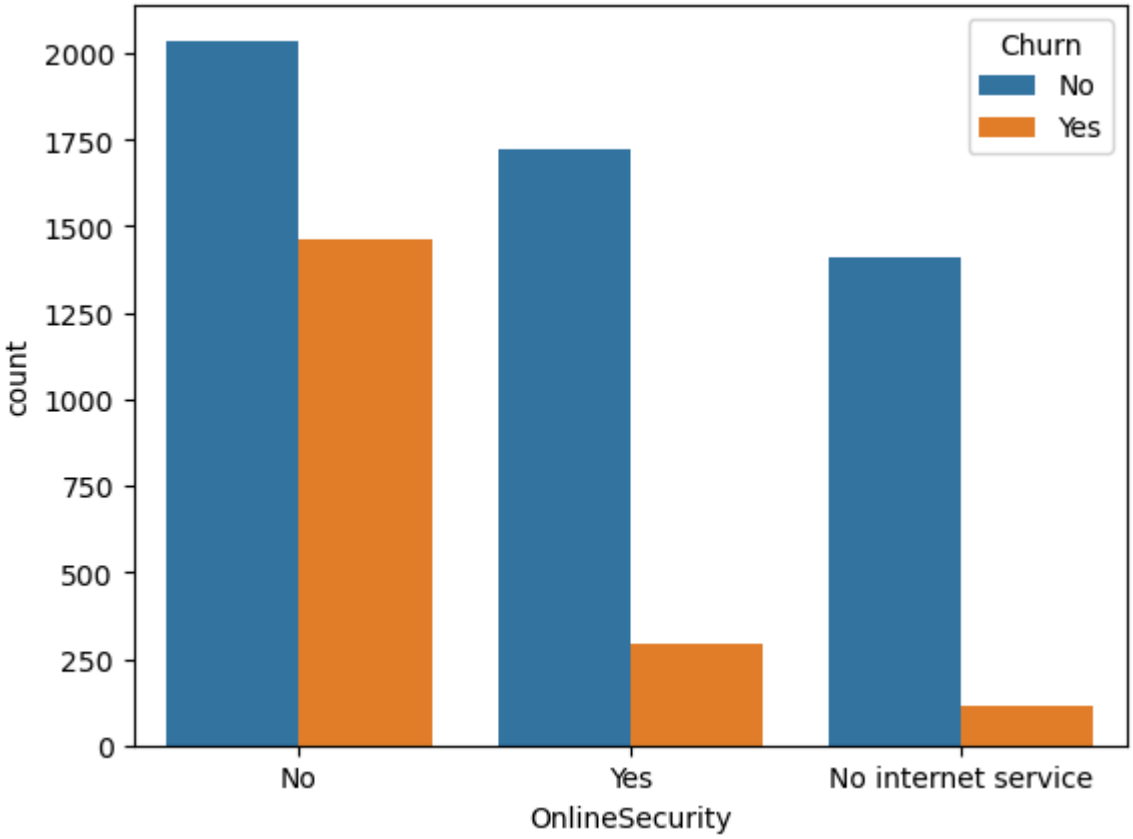

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
Partner

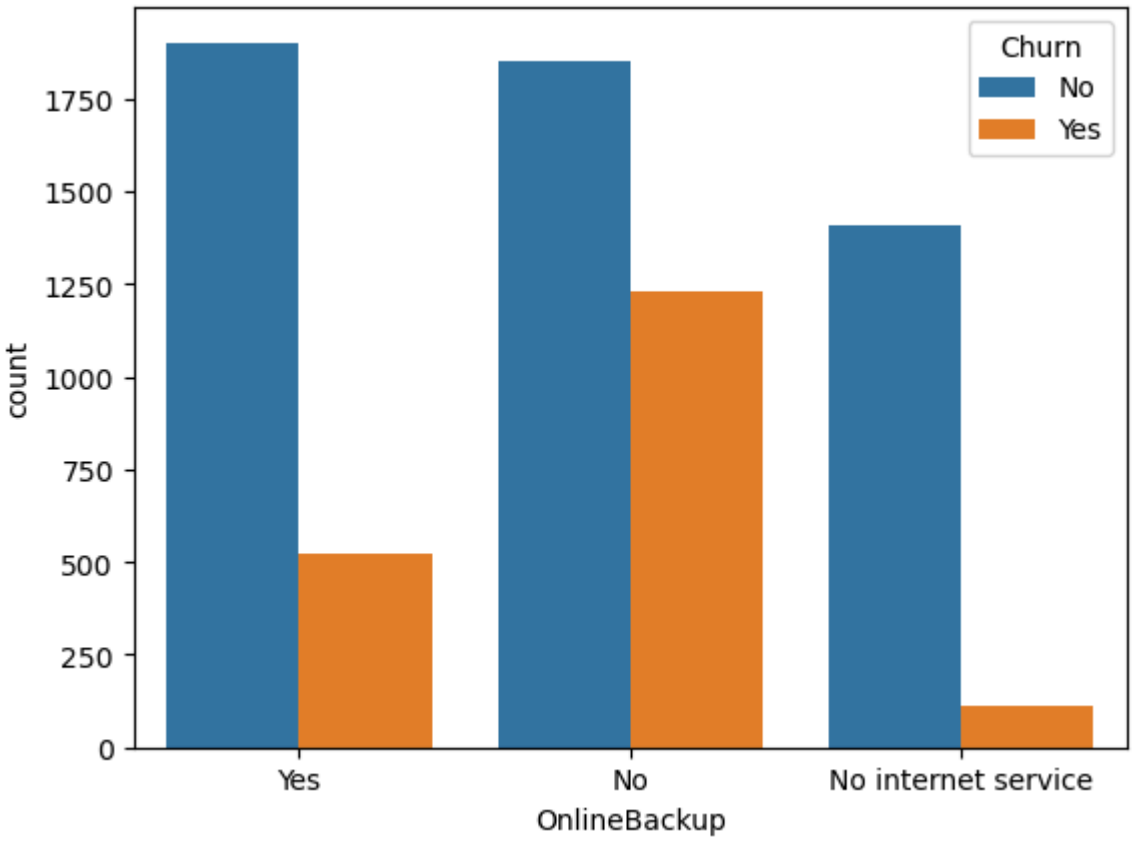

Churn
No
Yes
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents

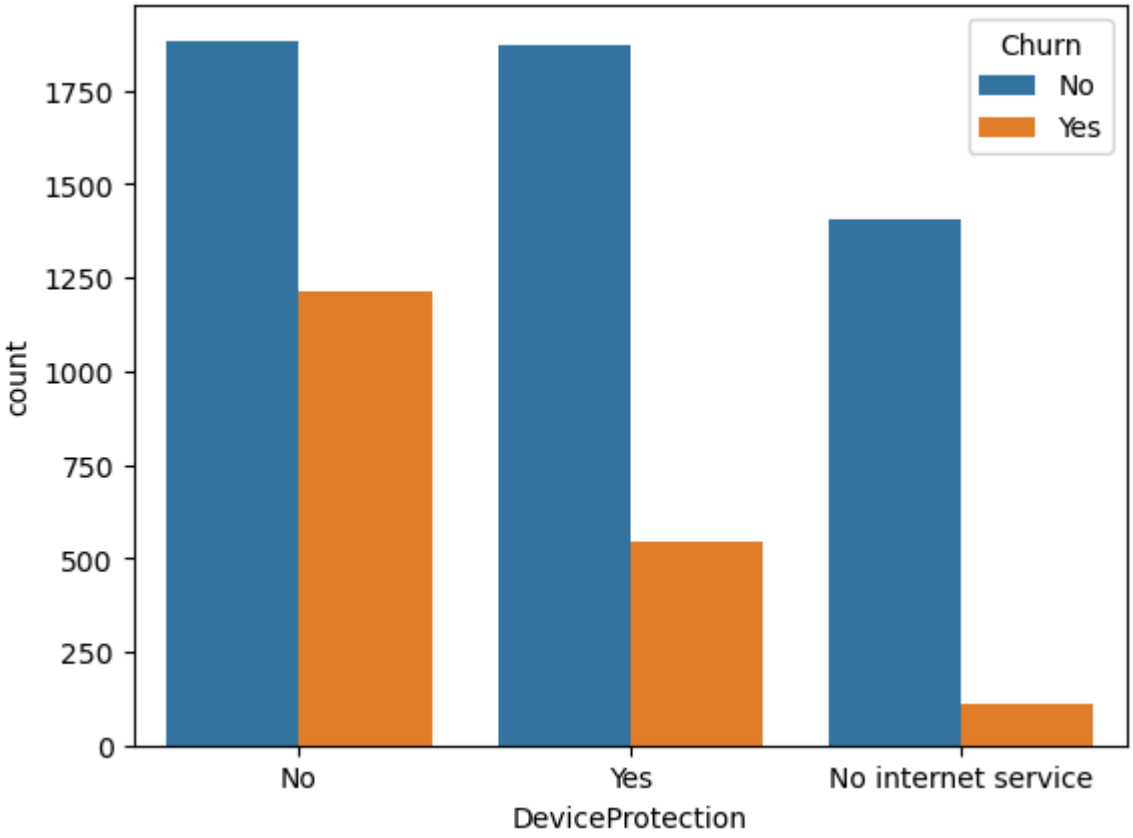

Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PhoneService

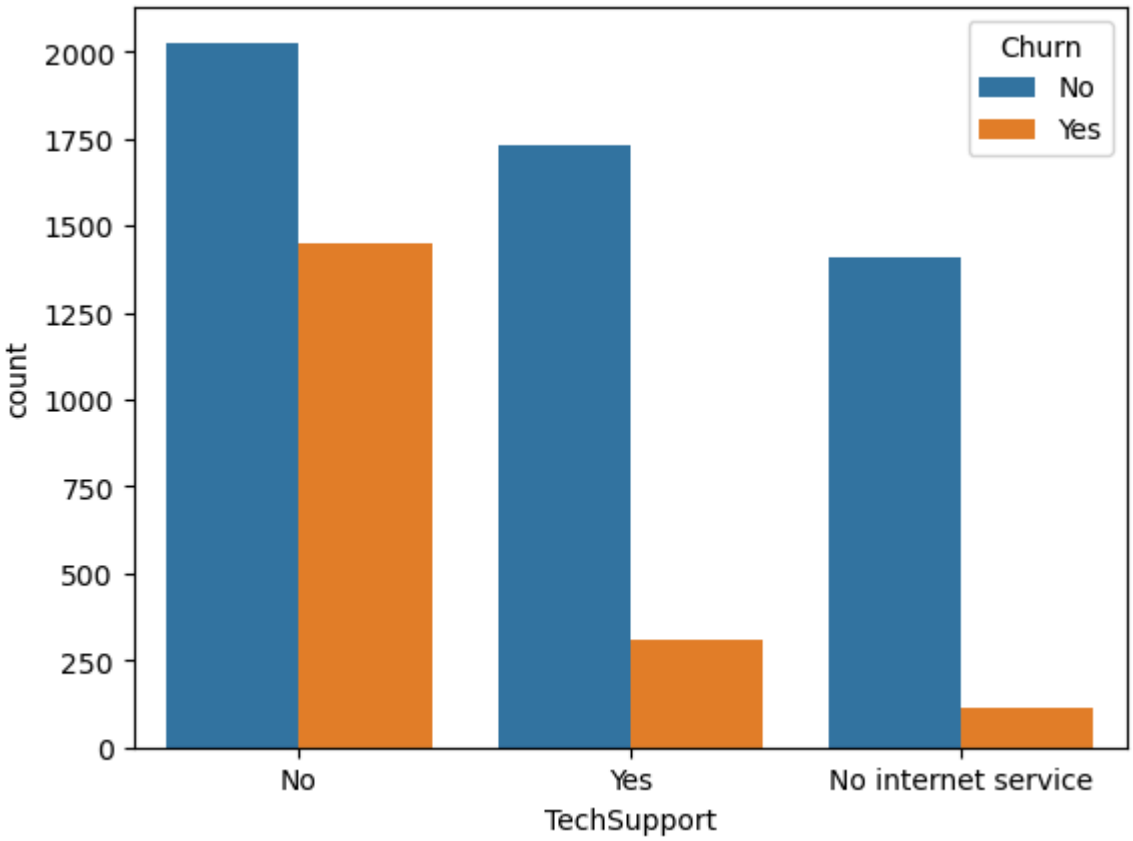

Churn
No
Yes
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

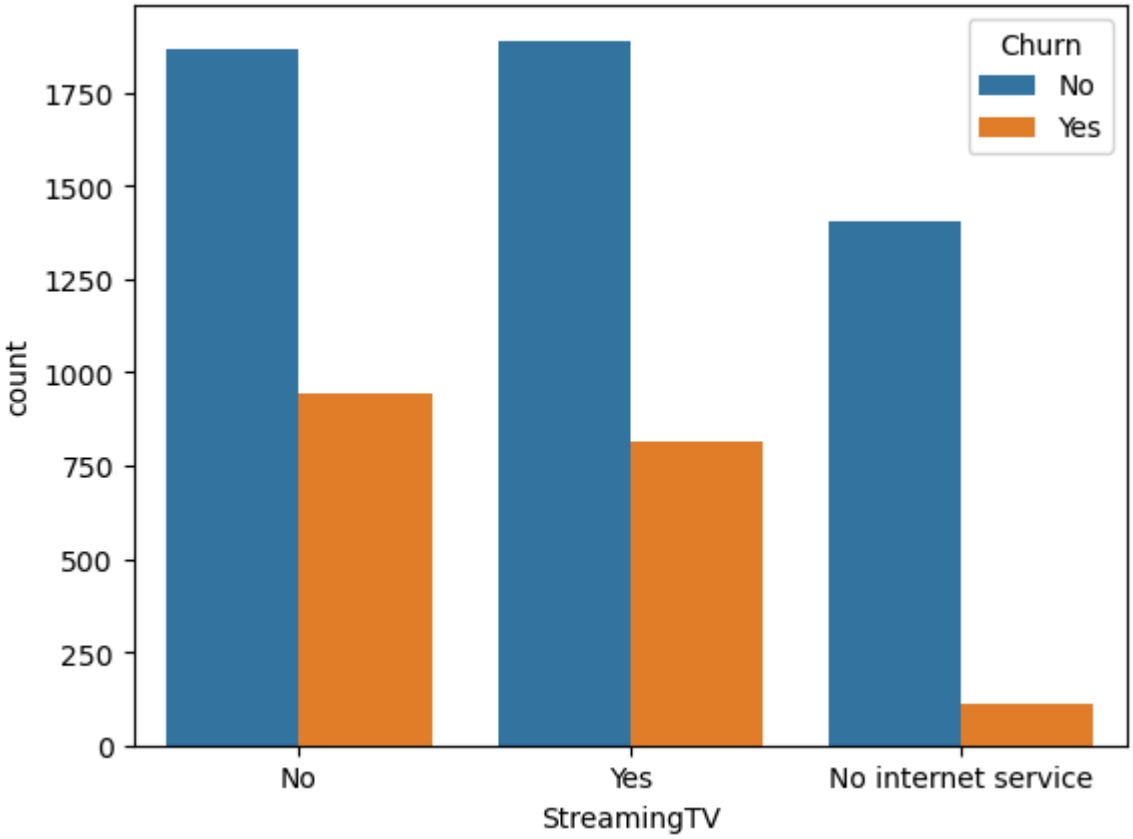

Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
DSL
Fiber optic
No
InternetService

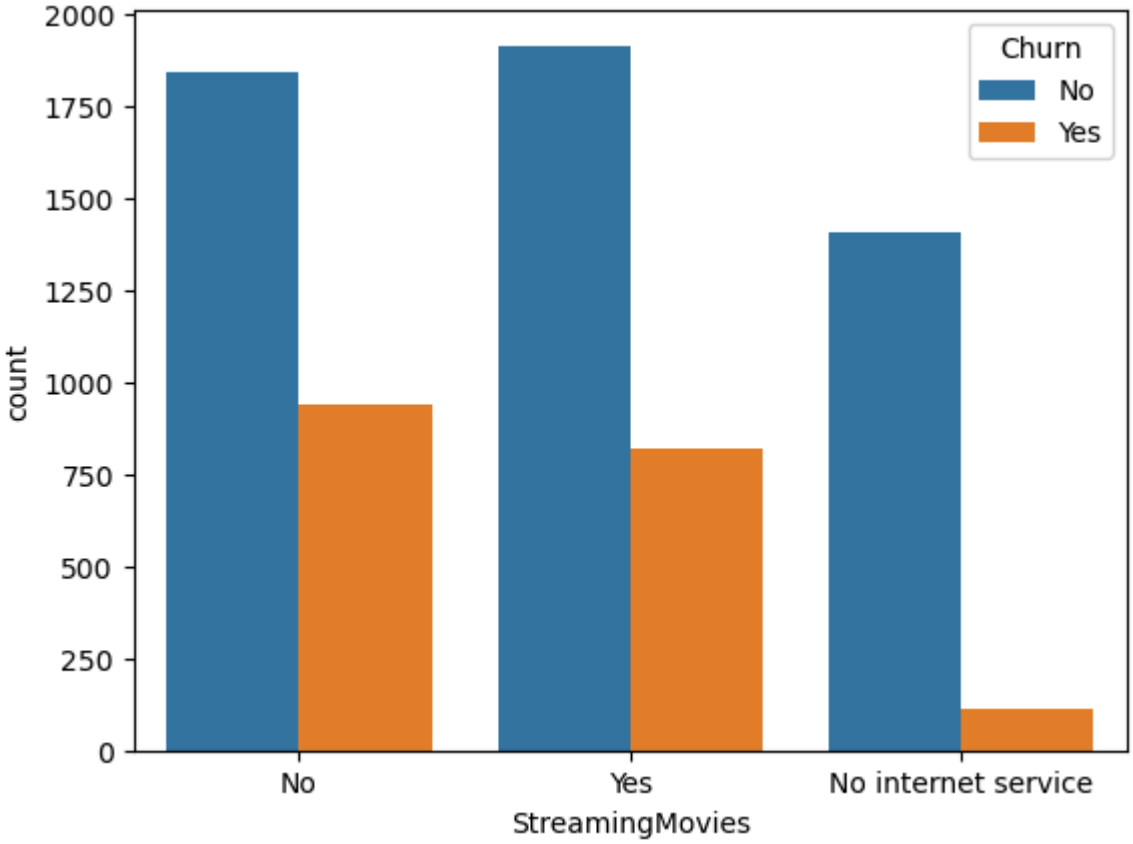

Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
OnlineSecurity

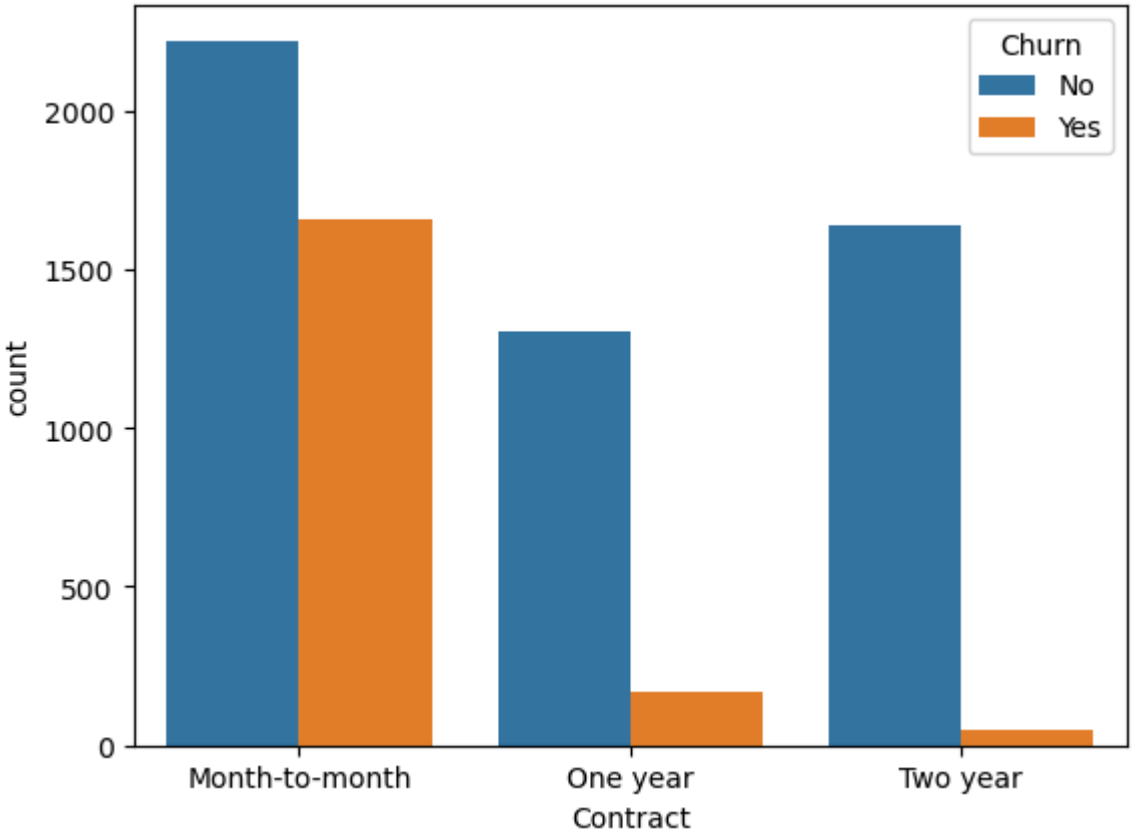

Churn
No
Yes
count
1750
1500
1250
1000
750
500
250
0
Yes
No
No internet service
OnlineBackup

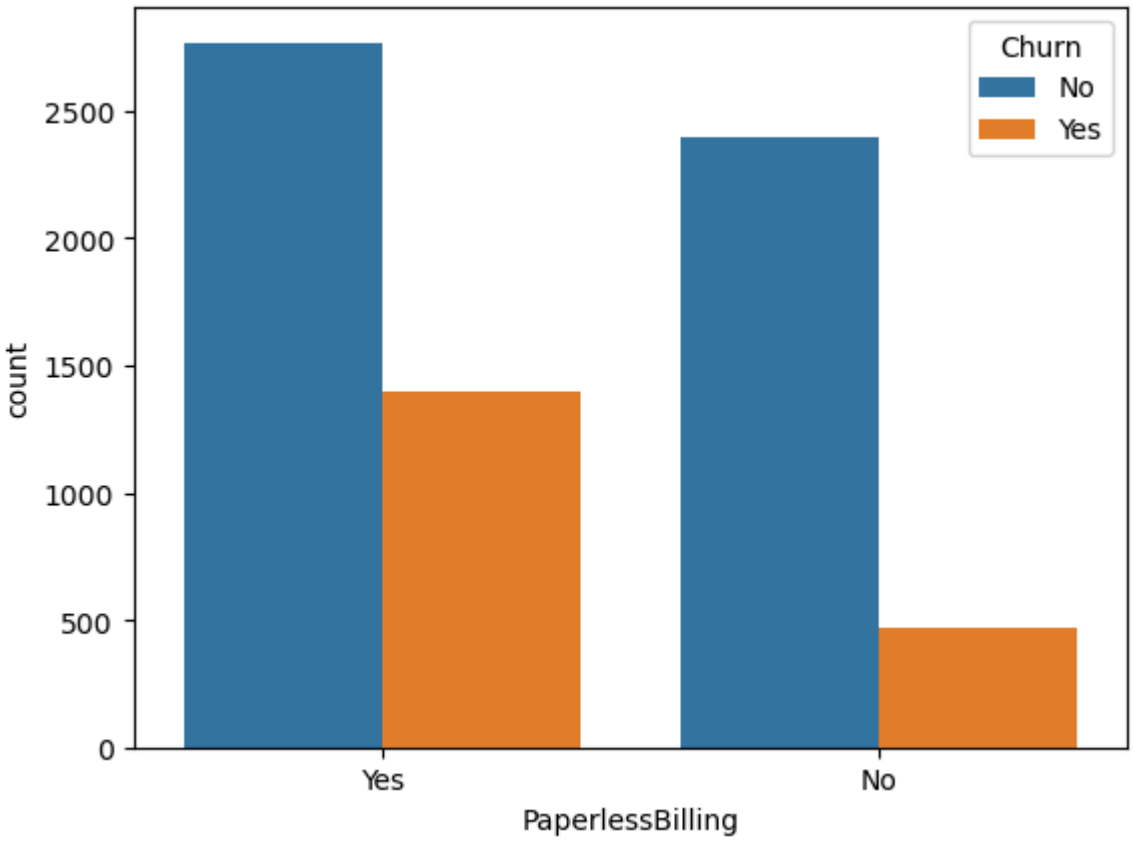

Churn
No
Yes
count
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
DeviceProtection

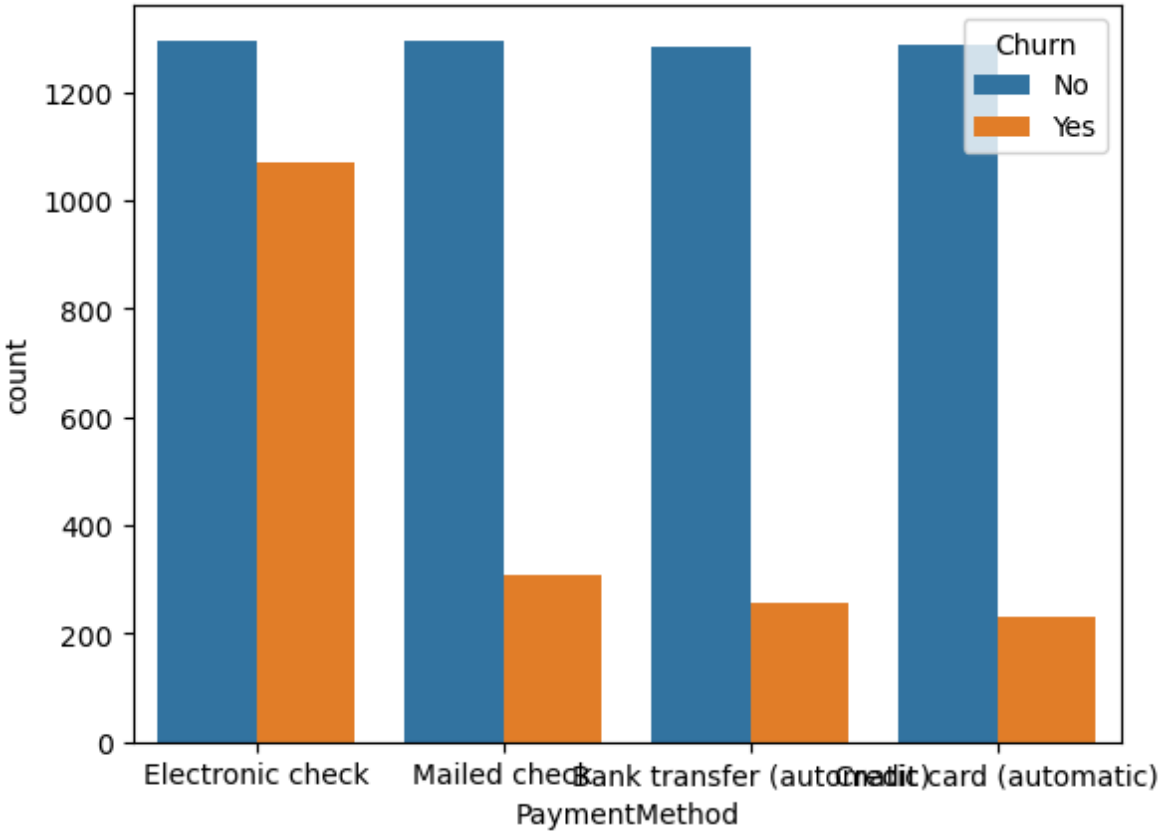

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
TechSupport

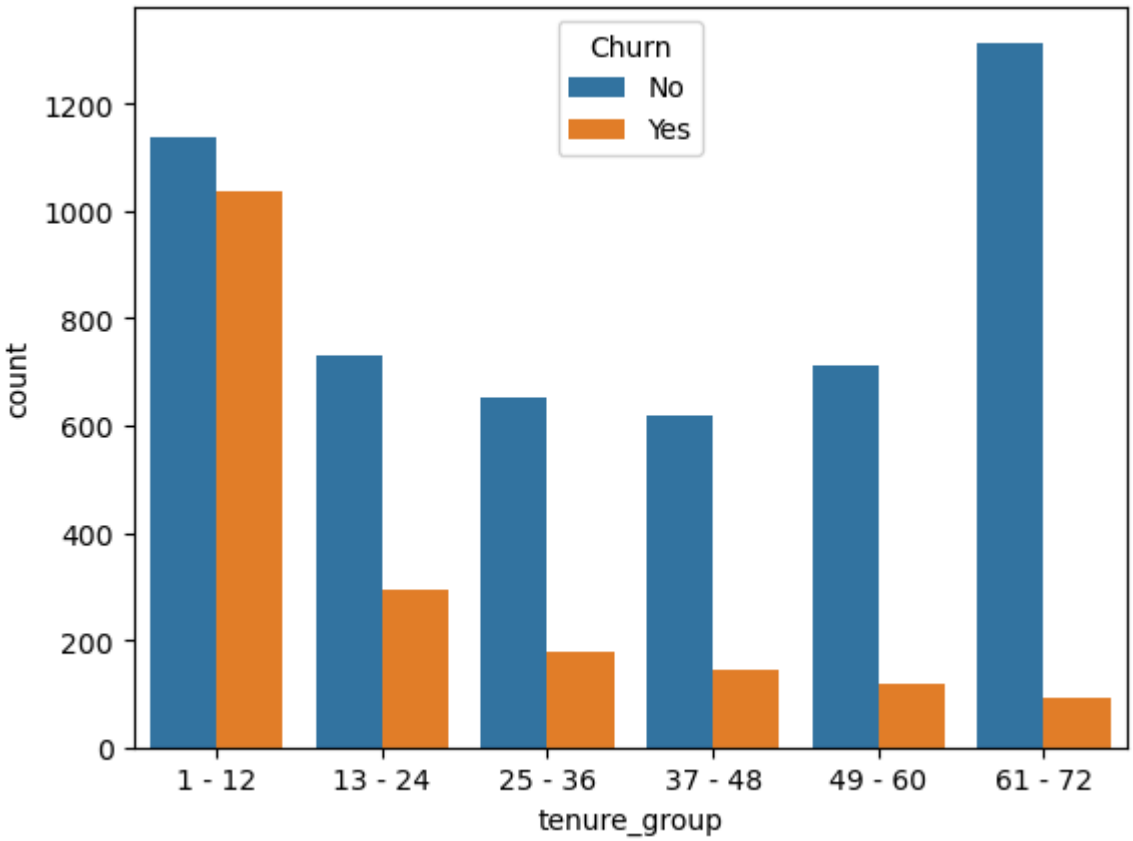

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingTV


Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
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
1200
1000
800
600
400
200
0
count
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod



2. Convert the target variable 'Churn' in a binary numeric variable i.e. Yes=1 ; No = 0

In [28]:
```
telco_data['Churn'] = np.where(telco_data.Churn == 'Yes',1,0)     # manual label enc
```

In [29]:
```
telco_data.head()
```

Out[29]:

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

In [30]:
```
telco_data_dummies = pd.get_dummies(telco_data)
telco_data_dummies.head()
# true == 1
# false == 0
# pd.get_dummies  it is used to convert categorical featutres to the numerical
```

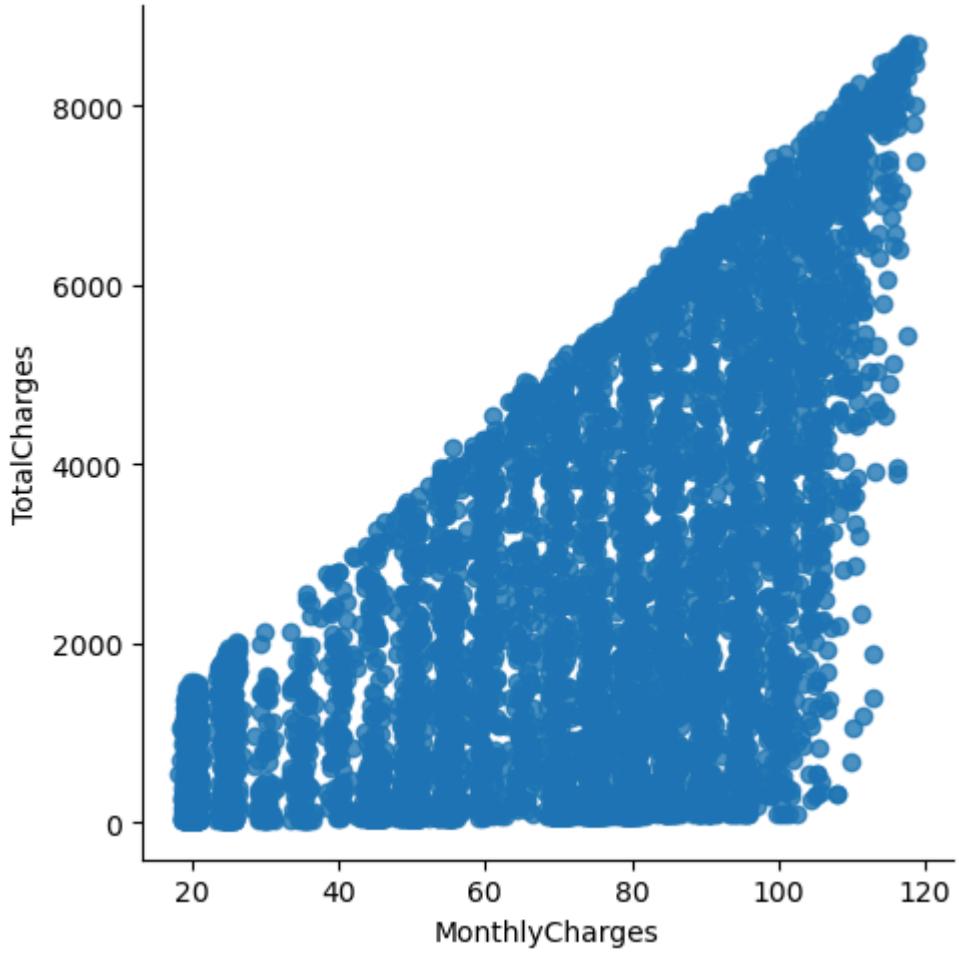
Out[30]:

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

5 rows × 51 columns

In [31]:
```
# Relationship between Monthly Charges and Total Charges
```

In [32]:
```
sns.lmplot(data=telco_data_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=F
plt.show()
```



Total Charges increase as Monthly Charges increase - as expected.

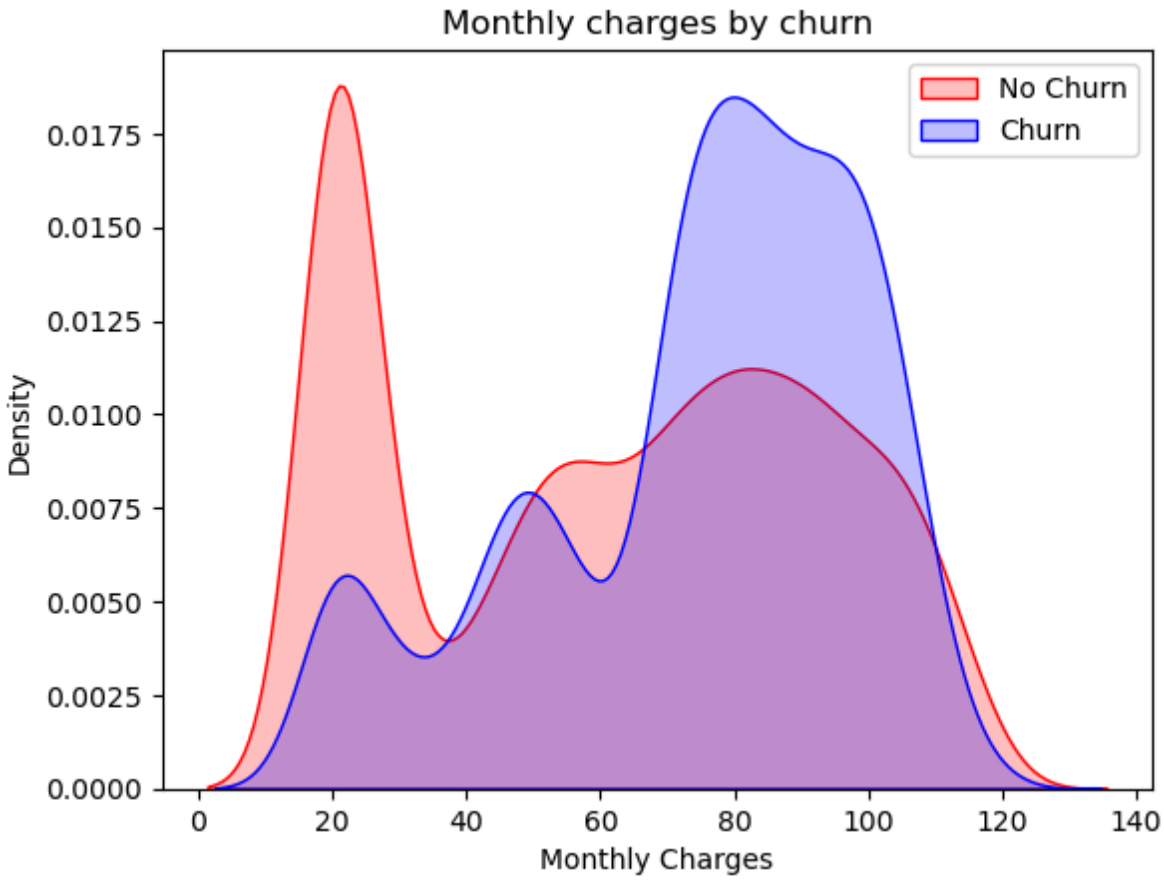
In [33]:
```
# Churn by Monthly Charges and Total Charges
```

In [15]:
```
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] ==
                  color="Red", shade = True)
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] ==
                  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')
```

```
C:\Users\sawan\AppData\Local\Temp\ipykernel_24728\722082952.py:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"]
== 0) ],
C:\Users\sawan\AppData\Local\Temp\ipykernel_24728\722082952.py:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"]
== 1) ],
```

Out[15]:
```
Text(0.5, 1.0, 'Monthly charges by churn')
```



Insight: Churn is high when Monthly Charges are high

Surprising insight as higher Churn at lower Total Charges

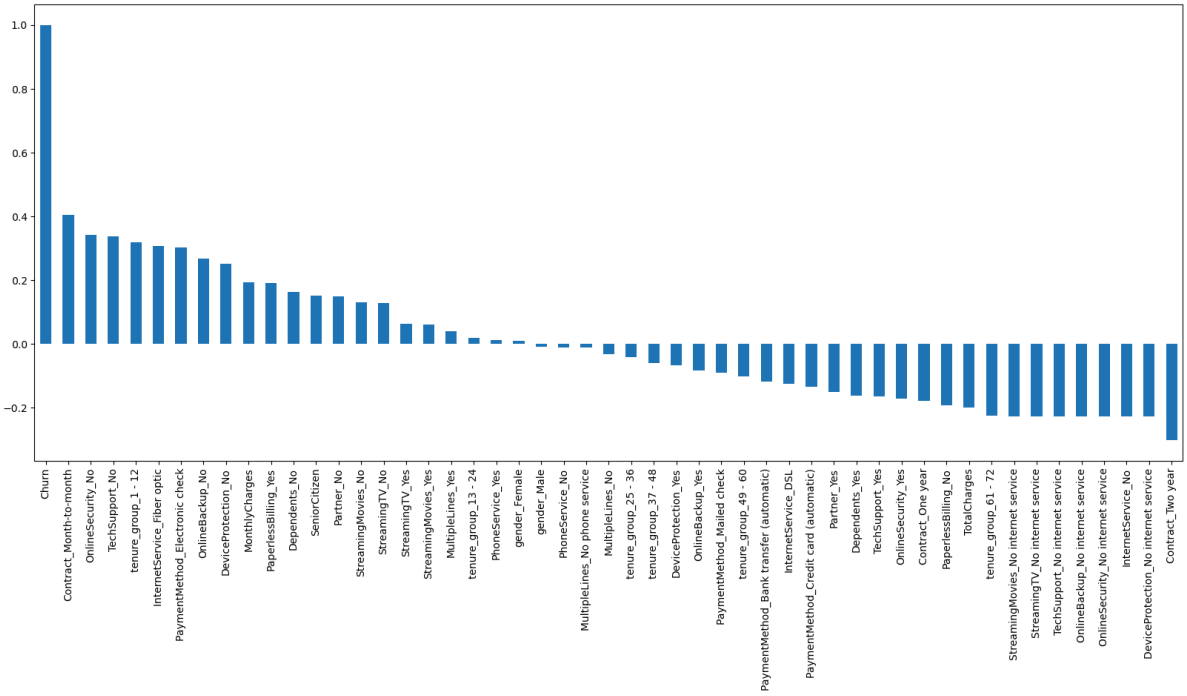
However if we combine the insights of 3 parameters i.e. Tenure, Monthly Charges & Total Charges then the picture is bit clear :- Higher Monthly Charge at lower tenure results into lower Total Charge. Hence, all these 3 factors viz **Higher Monthly Charge**, **Lower tenure** and **Lower Total Charge** are linkd to **High Churn**.

In [ ]:

11. Build a corelation of all predictors with 'Churn'

In [39]:
```
plt.figure(figsize=(20,8))
telco_data_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

Out[39]:
```
<Axes: >
```



Derived Insight:

HIGH Churn seen in case of **Month to month contracts**, **No online security**, **No Tech support**, **First year of subscription** and **Fibre Optics Internet**

LOW Churn is seens in case of **Long term contracts**, **Subscriptions without internet service** and **The customers engaged for 5+ years**

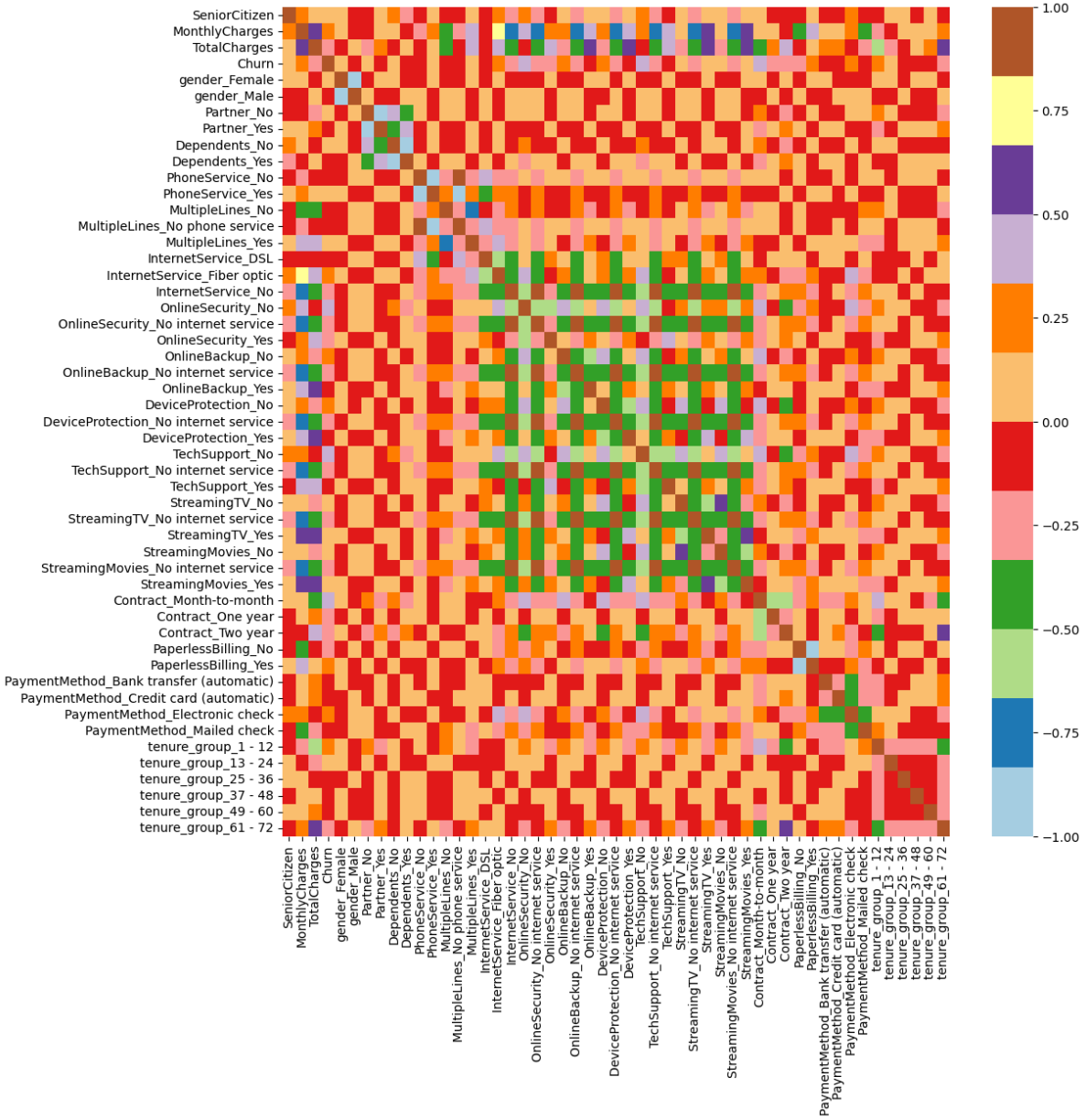
Factors like **Gender**, **Availability of PhoneService** and **# of multiple lines** have alomost **NO** impact on Churn

This is also evident from the **Heatmap** below

In [ ]:
```
telco_data_dummies.corr()
```

In [41]:
```
plt.figure(figsize=(12,12))
sns.heatmap(telco_data_dummies.corr(), cmap="Paired")
```

Out[41]:
```
<Axes: >
```



Bivariate Analysis

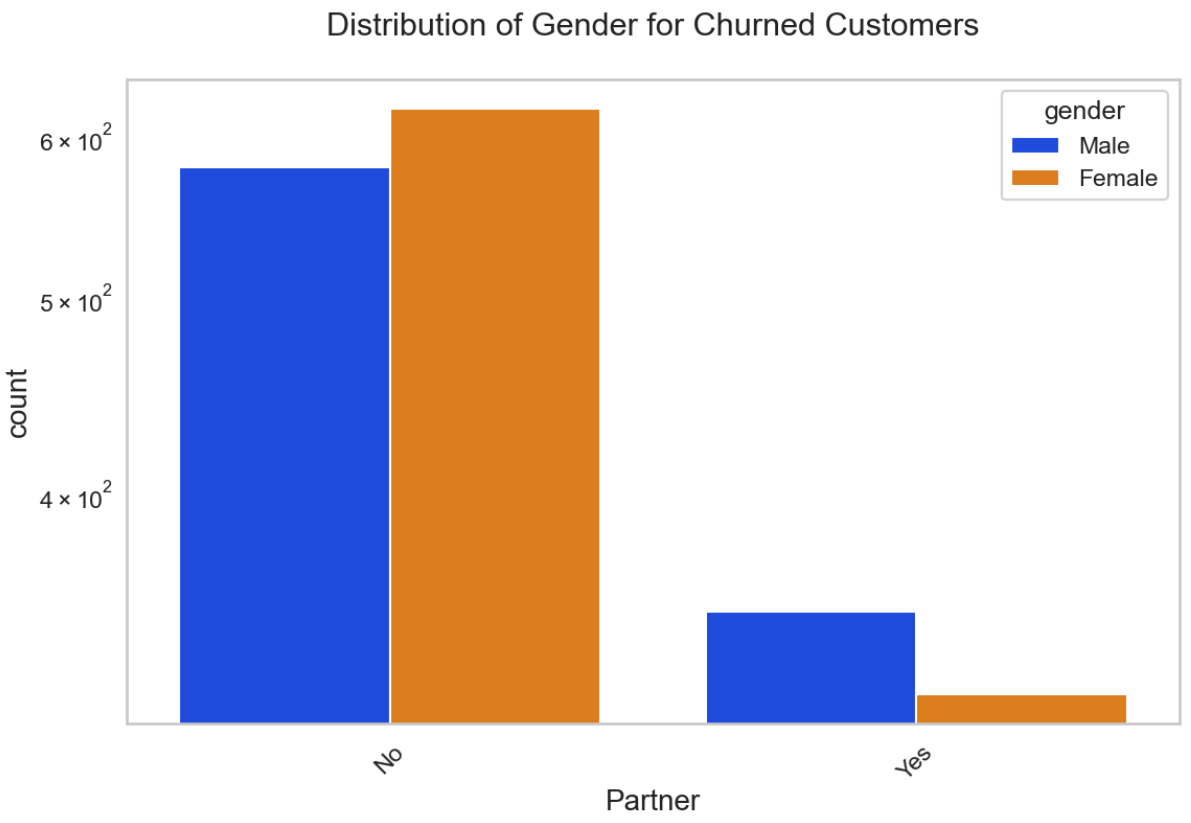
In [50]:
```
new_df1_target0 = telco_data.loc[telco_data["Churn"]==0]
new_df1_target1 = telco_data.loc[telco_data["Churn"]==1]
```

In [51]:
```
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 =
```

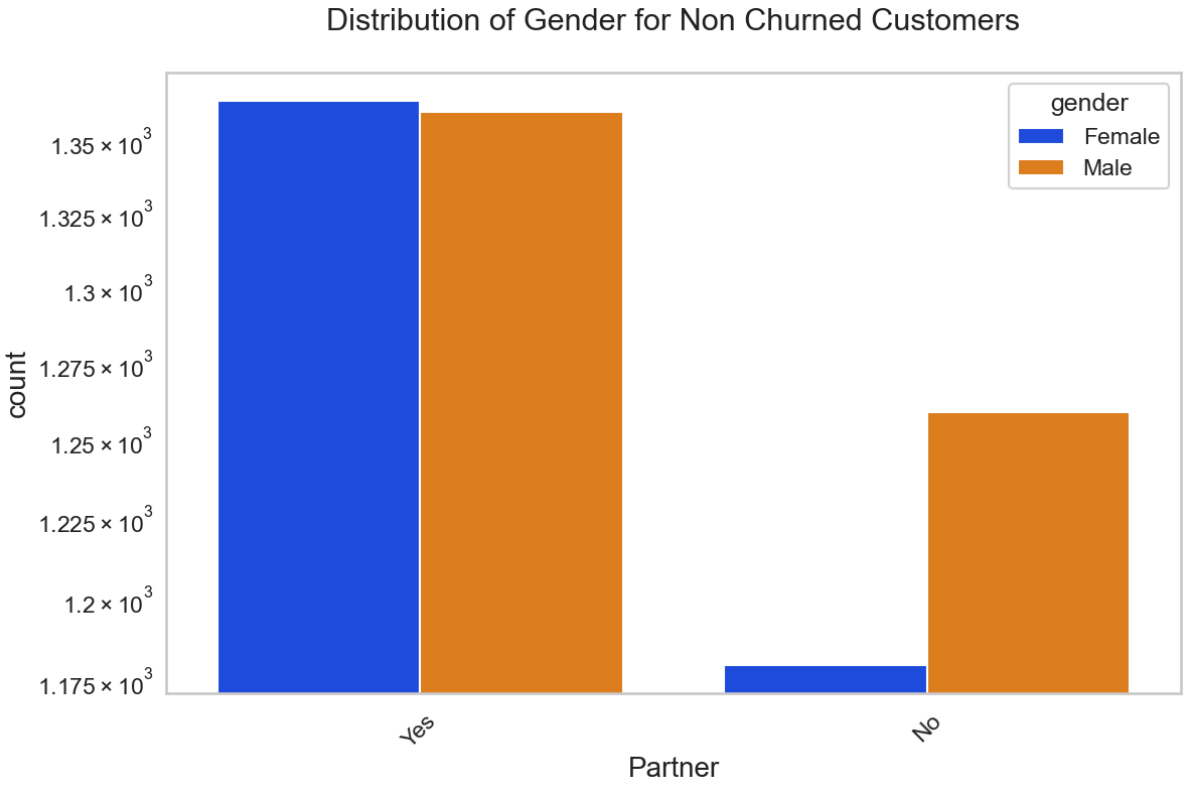
```
    plt.show()
```

In [ ]:

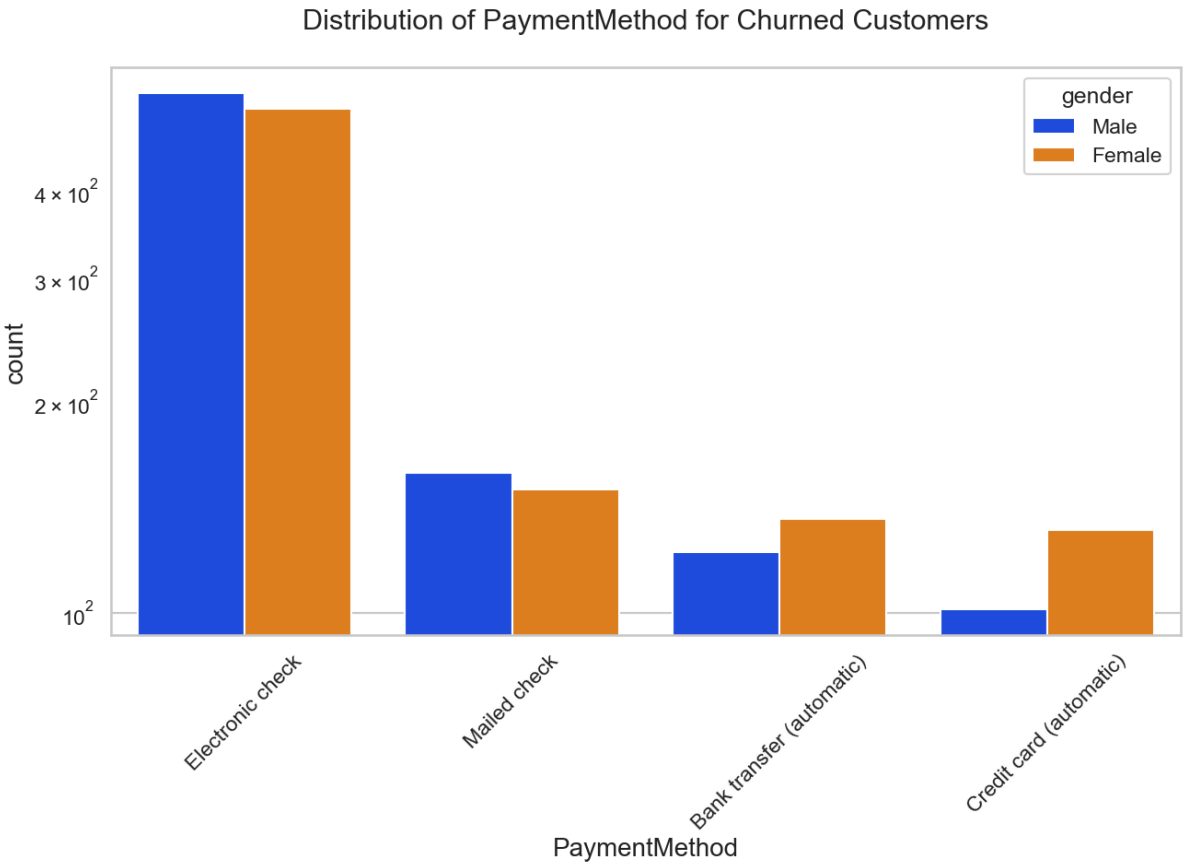
In [52]:
```
uniplot(new_df1_target1,col='Partner',title='Distribution of Gender for Churned Cus
```



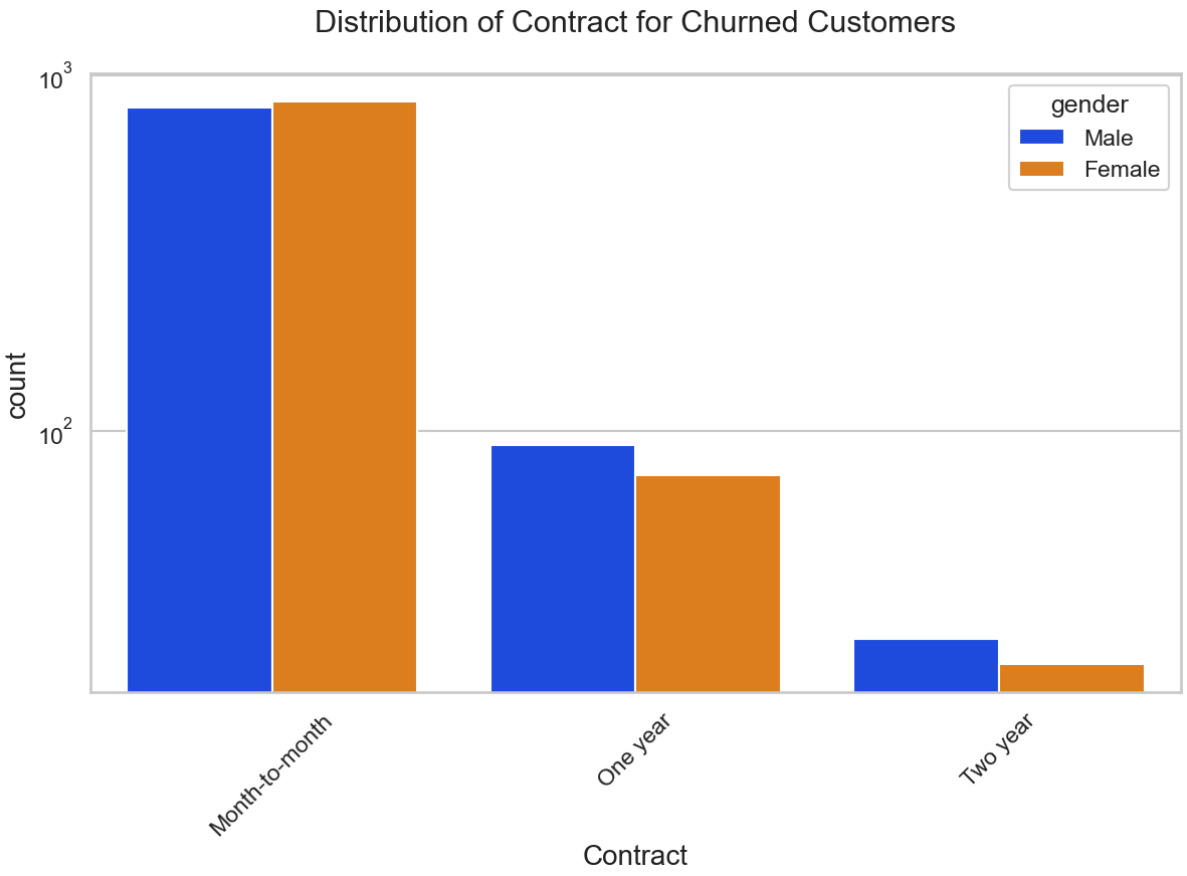
In [53]:
```
uniplot(new_df1_target0,col='Partner',title='Distribution of Gender for Non Churned
```



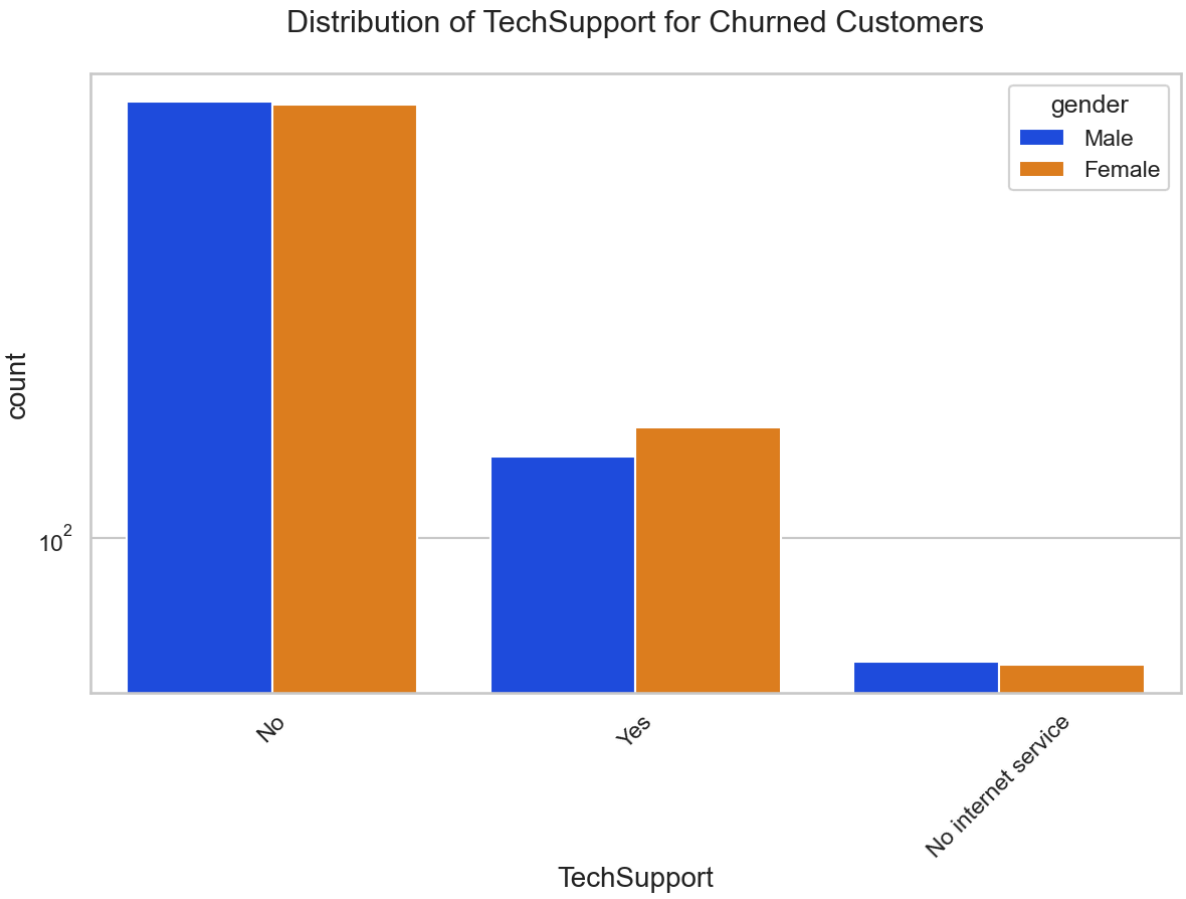
In [54]:
```
uniplot(new_df1_target1,col='PaymentMethod',title='Distribution of PaymentMethod fo
```



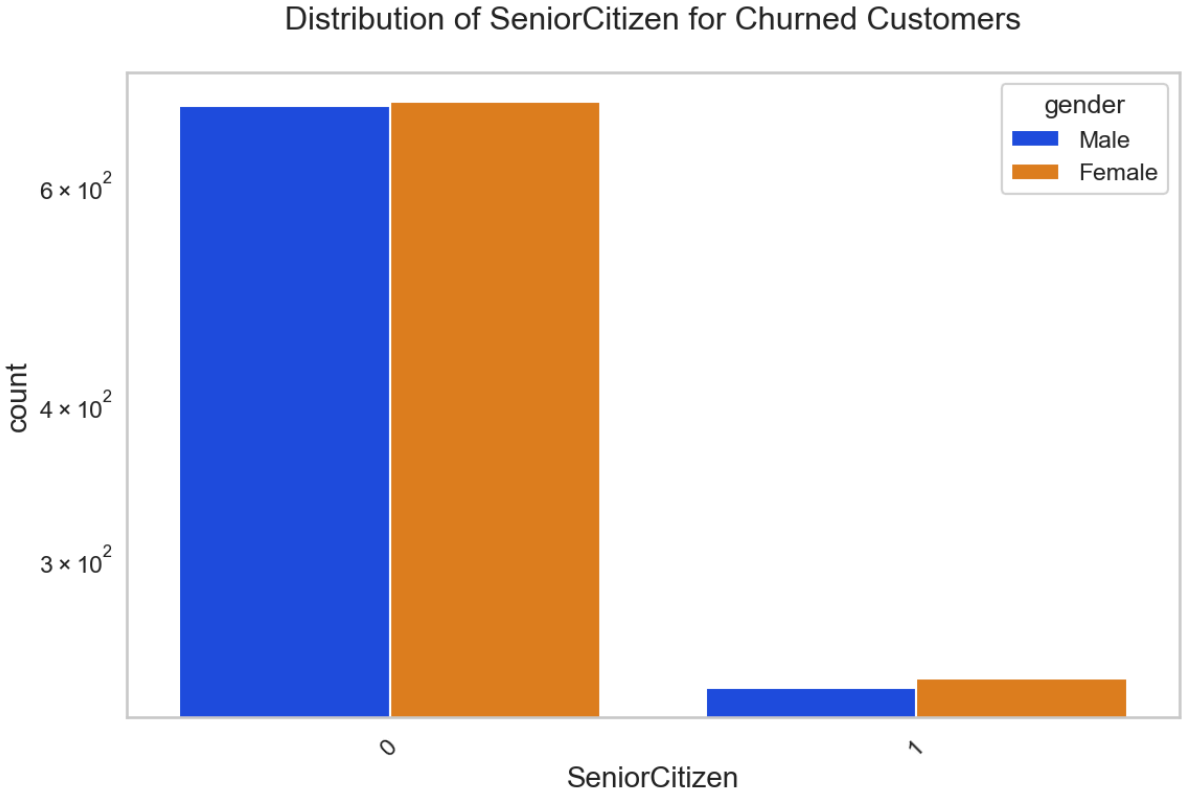
In [55]:
```
uniplot(new_df1_target1,col='Contract',title='Distribution of Contract for Churned
```



In [56]:
```
uniplot(new_df1_target1,col='TechSupport',title='Distribution of TechSupport for Ch
```



In [57]:
```
uniplot(new_df1_target1,col='SeniorCitizen',title='Distribution of SeniorCitizen fo
```



CONCLUSION

These are some of the quick insights from this exercise:

1. Electronic check medium are the highest churners

2. Contract Type - Monthly customers are more likely to churn because of no contract terms, as they are free to go customers.
3. No Online security, No Tech Support category are high churners
4. Non senior Citizens are high churners

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: