# Telecom Churn Analysis EDA

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

## About Dataset

### Context

"Predict behavior to retain customers. You can analyze all relevant customer data and develop focused customer retention programs." [IBM Sample Data Sets]

### Content

Each row represents a customer, each column contains customer’s attributes described on the column Metadata.

The data set includes information about:

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they’ve been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

In [1]:

```
#import the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
%matplotlib inline
```

Load the data file

In [2]:

```
telco_base_data = pd.read_csv('C:/Users/sub13/Downloads/Dataset/ChurnPred
```

Look at the top 5 records of data

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

Out[3]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Int |
|---|---|---|---|---|---|---|---|---|---|
| **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

Check the various attributes of data like shape (rows and cols), Columns, datatypes

In [4]:
```
telco_base_data.shape
```

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

In [5]:
```
telco_base_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 data types of all the columns
telco_base_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]:
```
# Check the descriptive statistics of numeric variables
telco_base_data.describe()
```

Out[7]:

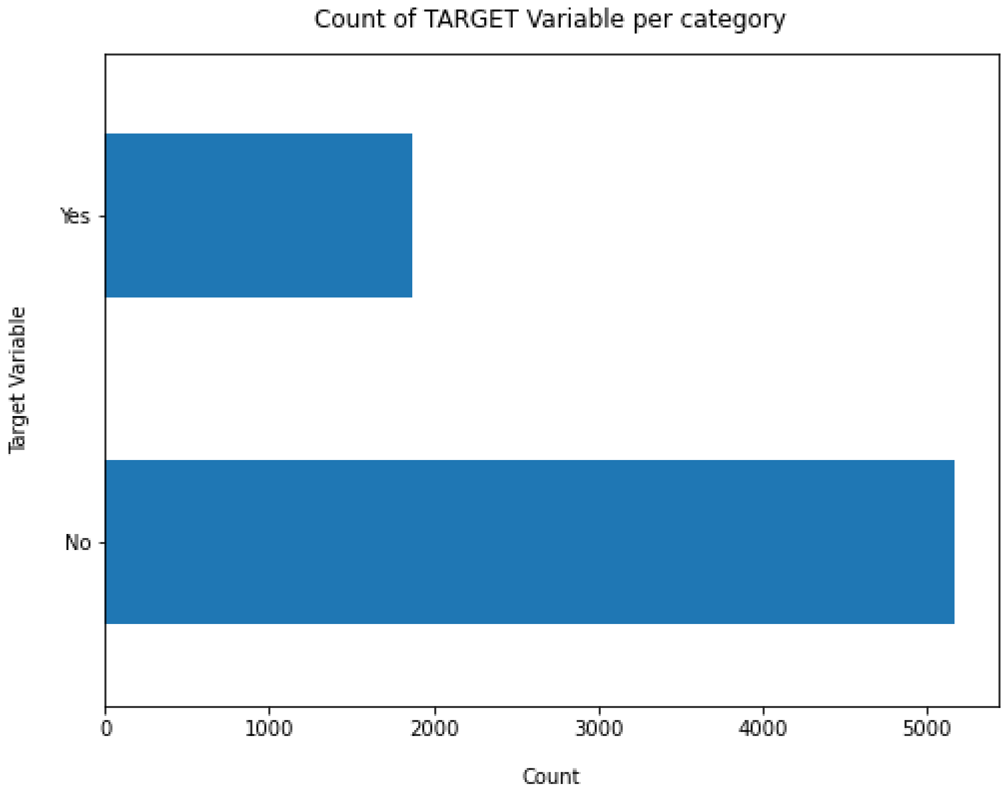
| | SeniorCitizen | tenure | MonthlyCharges |
|---|---|---|---|
| **count** | 7043.000000 | 7043.000000 | 7043.000000 |
| **mean** | 0.162147 | 32.371149 | 64.761692 |
| **std** | 0.368612 | 24.559481 | 30.090047 |
| **min** | 0.000000 | 0.000000 | 18.250000 |
| **25%** | 0.000000 | 9.000000 | 35.500000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

SeniorCitizen is actually a categorical hence the 25%-50%-75% distribution is not propoer

75% customers have tenure less than 55 months

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

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




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

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

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

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

- 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 [12]:

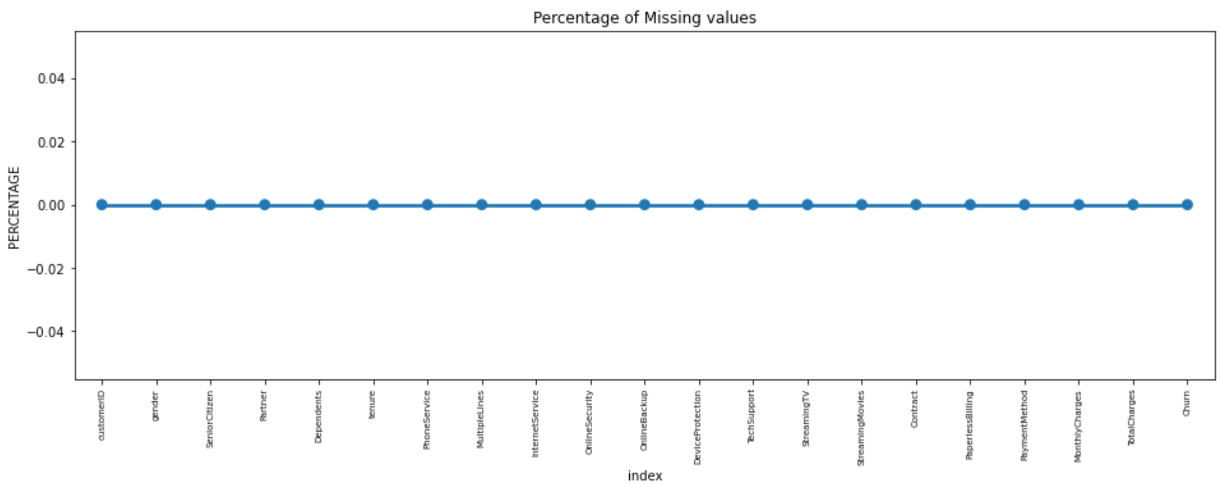
```
# Concise Summary of the dataframe, as we have too many columns, we are u
telco_base_data.info(verbose = True)
```

```
<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 [13]:

```
missing = pd.DataFrame((telco_base_data.isnull().sum())*100/telco_base_da
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\sub13\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```




## Missing Data - Initial Intuition

- Here, we don't have any missing data.

General Thumb Rules:

- For features with less missing values- can use regression to predict the missing values or fill with the mean of the values present, depending on the feature.
- For features with very high number of missing values- it is better to drop those columns as they give very less insight on analysis.
- As there's no thumb rule on what criteria do we delete the columns with high number of missing values, but generally you can delete the columns, if you have more than 30-40% of missing values. But again there's a catch here, for example, Is_Car & Car_Type, People having no cars, will obviously have Car_Type as NaN (null), but that doesn't make this column useless, so decisions has to be taken wisely.

## Data Cleaning

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

In [14]:
```
telco_data = telco_base_data.copy()
```

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

In [15]:
```
telco_data.TotalCharges = pd.to_numeric(telco_data.TotalCharges, errors='
telco_data.isnull().sum()
```

Out[15]:
```
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
```

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

In [16]:
```
telco_data.loc[telco_data ['TotalCharges'].isnull() == True]
```

Out[16]:

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

11 rows × 21 columns

#### 4. Missing Value Treatement

Since the % of these records compared to total dataset is very low ie 0.15%, it is safe to ignore them from further processing.

In [17]:
```
#Removing missing values
telco_data.dropna(how = 'any', inplace = True)

#telco_data.fillna(0)
```

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 [18]:
```
# Get the max tenure
print(telco_data['tenure'].max()) #72
```

```
72
```

In [19]:
```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]

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

In [20]:
```
telco_data['tenure_group'].value_counts()
```

Out[20]:
```
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 columns not required for processing

In [21]:
```
#drop column customerID and tenure
telco_data.drop(columns= ['customerID','tenure'], axis=1, inplace=True)
telco_data.head()
```

Out[21]:

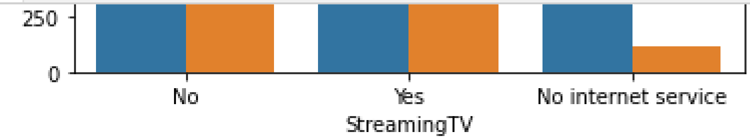
| | gender | SeniorCitizen | Partner | Dependents | PhoneService | MultipleLines | InternetService | Online |
|---|---|---|---|---|---|---|---|---|
| **0** | Female | 0 | Yes | No | No | No phone service | DSL | |
| **1** | Male | 0 | No | No | Yes | No | DSL | |
| **2** | Male | 0 | No | No | Yes |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

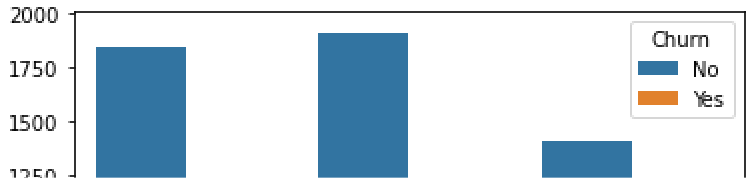
## Data Exploration

1. Plot distibution of individual predictors by churn

### Univariate Analysis

In [22]:
```python
for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCha
    plt.figure(i)
    sns.countplot(data=telco_data, x=predictor, hue='Churn')
```




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

In [23]:
```python
telco_data['Churn'] = np.where(telco_data.Churn == 'Yes',1,0)
```

In [24]:
```python
telco_data.head()
```

Out[24]:

| | gender | SeniorCitizen | Partner | Dependents | PhoneService | MultipleLines | InternetService | OnlineS |
|---|---|---|---|---|---|---|---|---|
| **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 | |

3. Convert all the categorical variables into dummy variables

In [25]:
```
telco_data_dummies = pd.get_dummies(telco_data)
telco_data_dummies.head()
```

Out[25]:

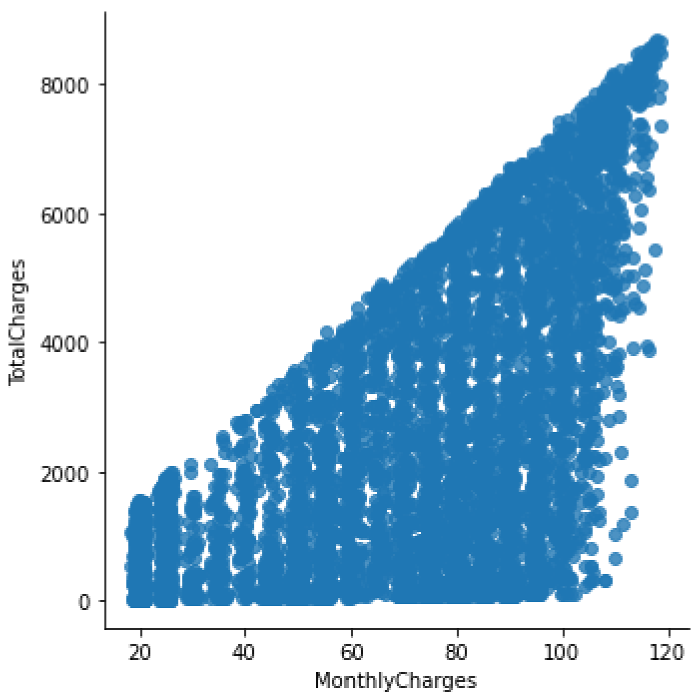
| | SeniorCitizen | MonthlyCharges | TotalCharges | Churn | gender_Female | gender_Male | Partner_No | P |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 29.85 | 29.85 | 0 | 1 | 0 | 0 | |
| **1** | 0 | 56.95 | 1889.50 | 0 | 0 | 1 | 1 | |
| **2** | 0 | 53.85 | 108.15 | 1 | 0 | 1 | 1 | |
| **3** |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

5 rows × 51 columns

9. Relationship between Monthly Charges and Total Charges

In [26]:
```
lmplot(data=telco_data_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x2172554f160>



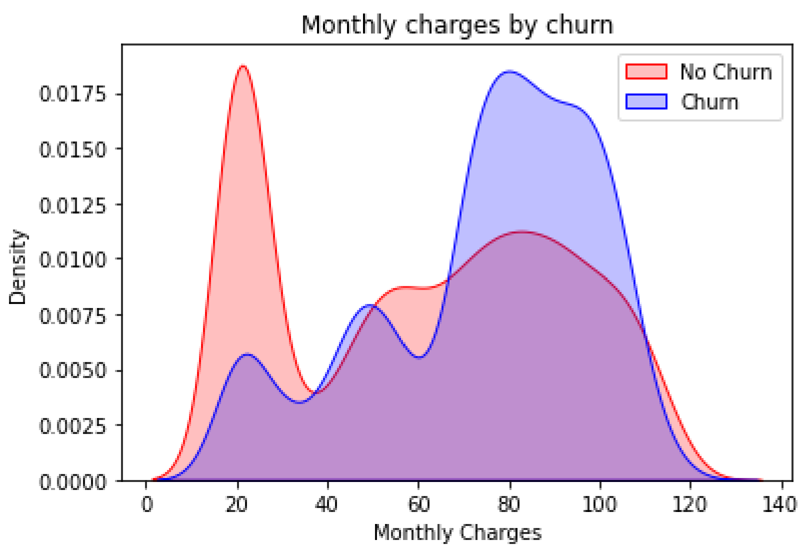

Total Charges increase as Monthly Charges increase - as expected.

10. Churn by Monthly Charges and Total Charges

In [27]:

```
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["
                  color="Red", shade = True)
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["
                  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')
```

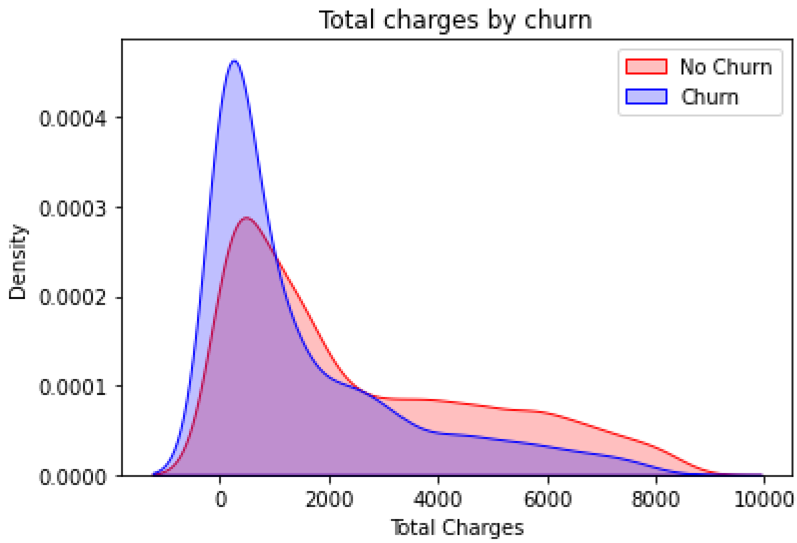
Out[27]: Text(0.5, 1.0, 'Monthly charges by churn')




**Insight:** Churn is high when Monthly Charges ar high

In [28]:
```
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Ch
                color="Red", shade = True)
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Ch
                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')
```



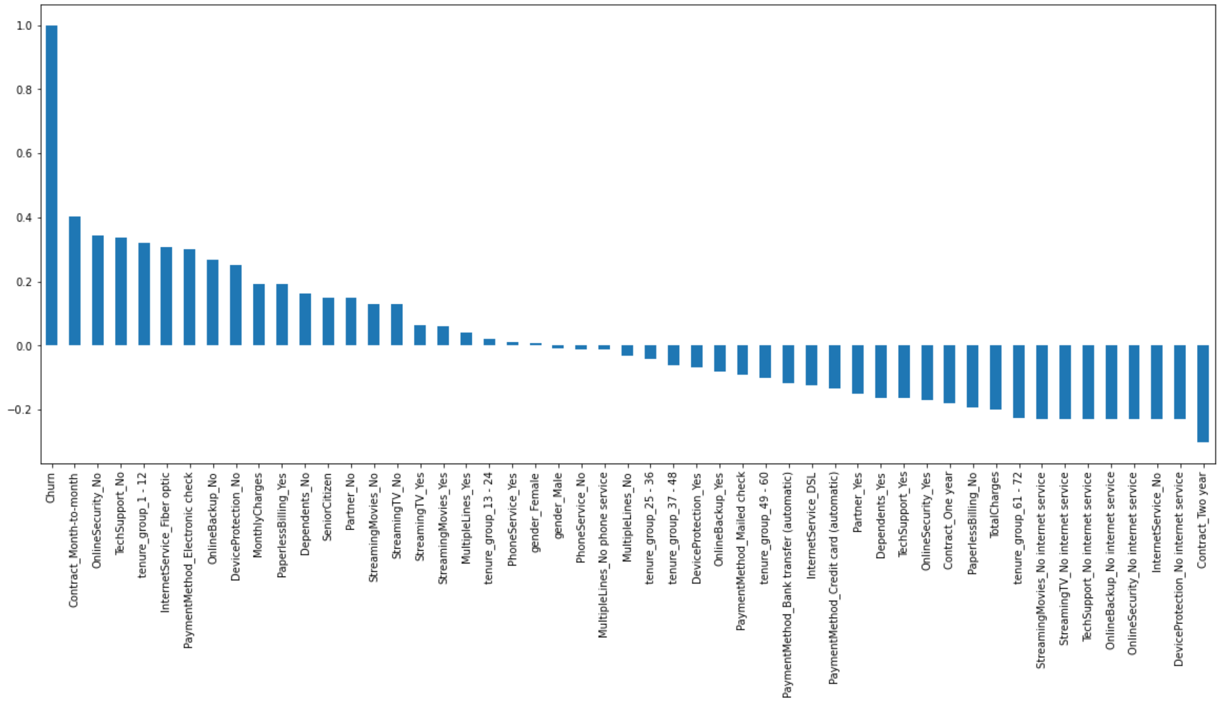

**Surprising insight** as higher Churn at lower Total Charges

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**.

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

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

Out[29]: <AxesSubplot:>




**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

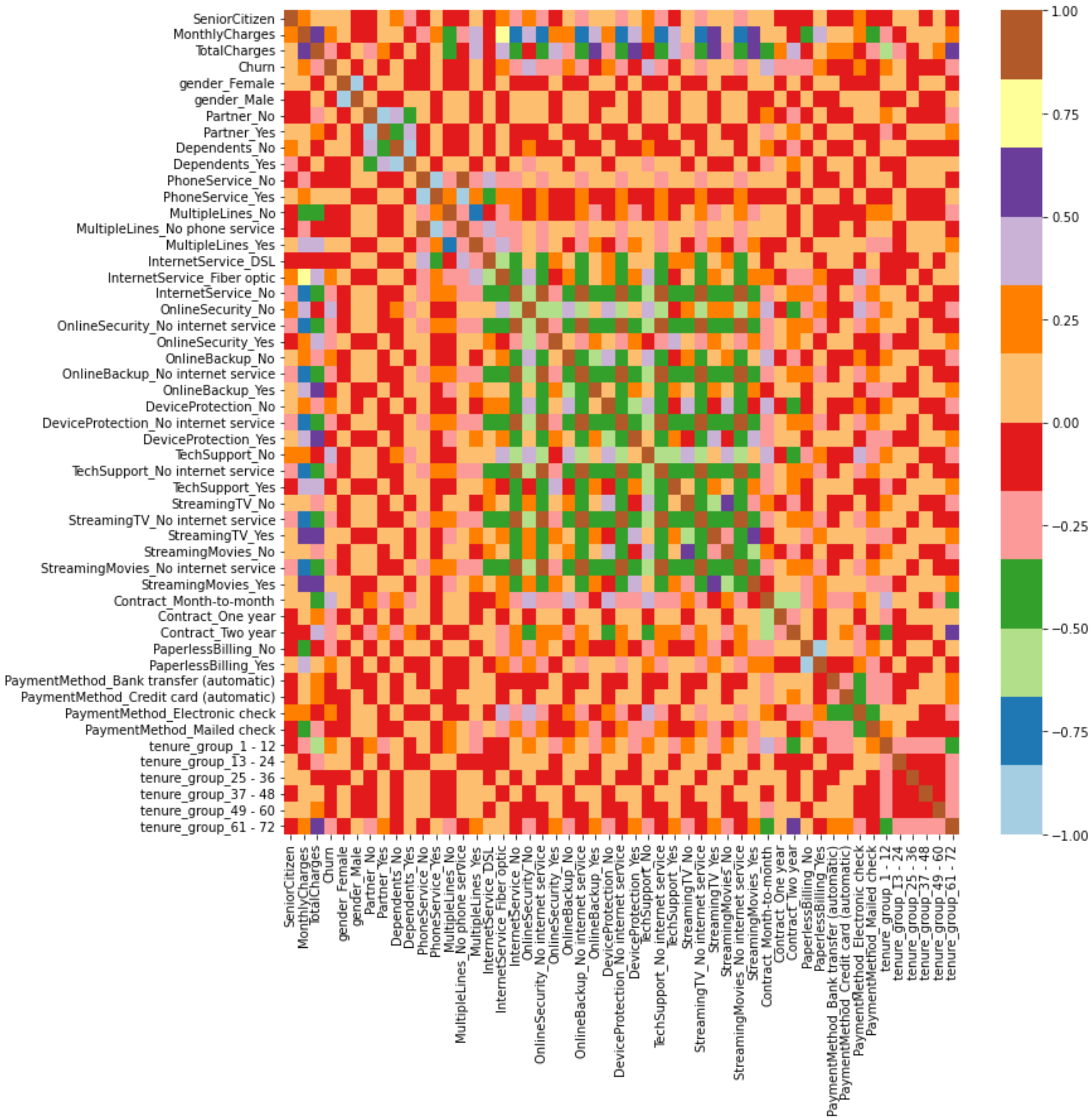
LOW Churn is seens in case of Long term contracts, Subscriptions without internet service and The customers engaged for 5+ years

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 [30]:
```
plt.figure(figsize=(12,12))
sns.heatmap(telco_data_dummies.corr(), cmap="Paired")
```

Out[30]: `<AxesSubplot:>`




### Bivariate Analysis

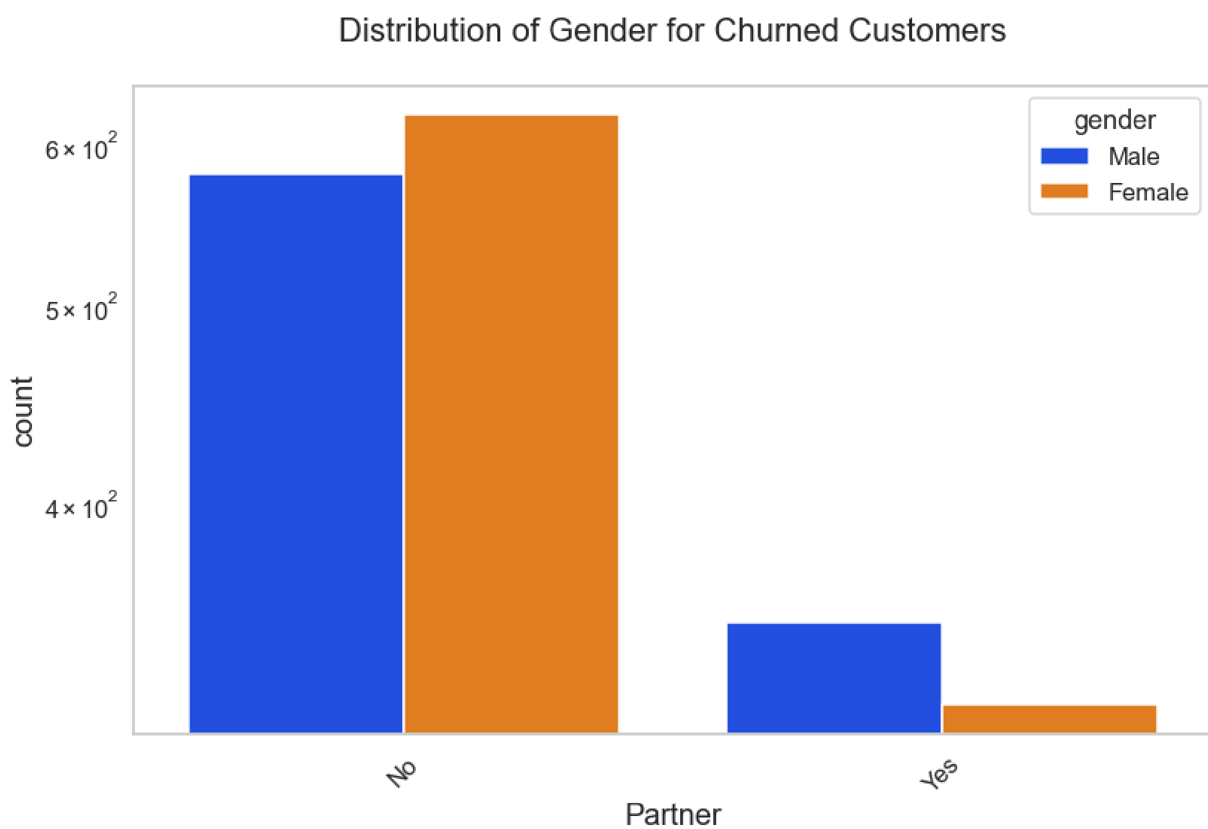
In [31]:
```
new_df1_target0=telco_data.loc[telco_data["Churn"]==0]
new_df1_target1=telco_data.loc[telco_data["Churn"]==1]
```

In [32]:

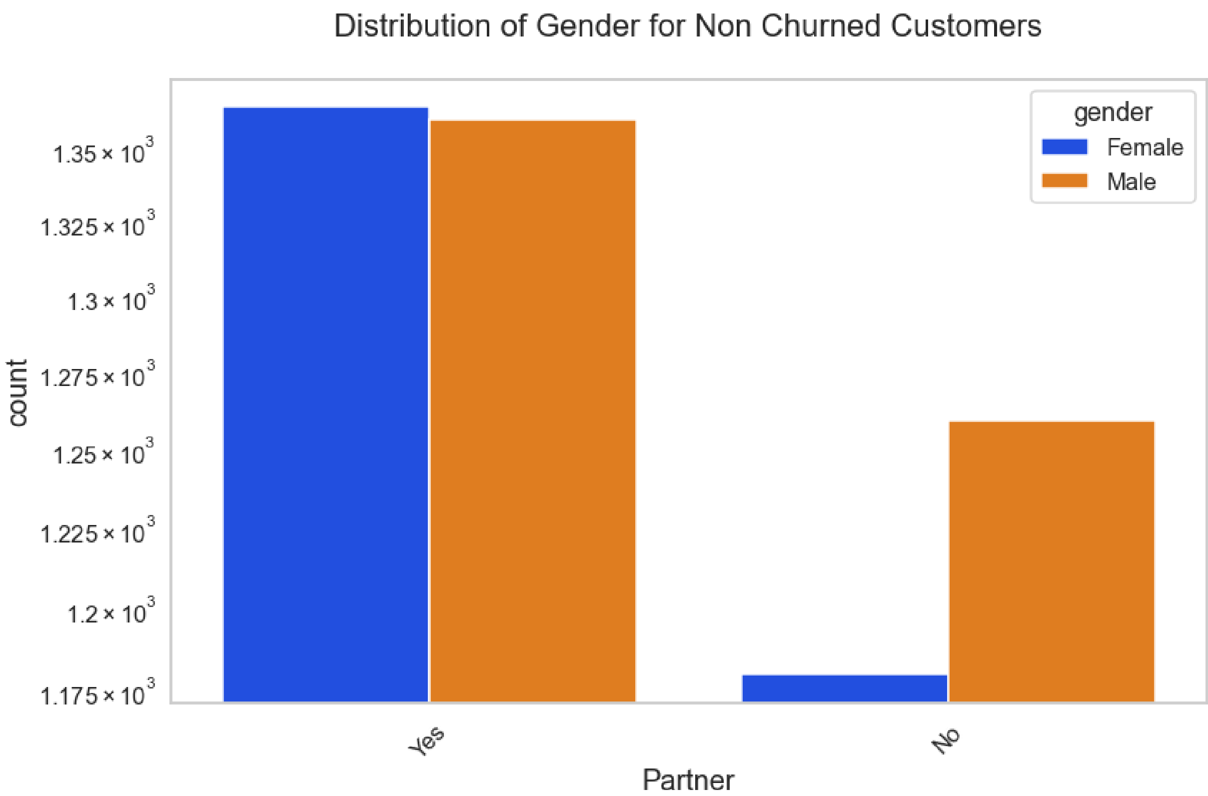
```
def uniplot(df,col,title,hue =None):

    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().in

    plt.show()
```

In [33]:

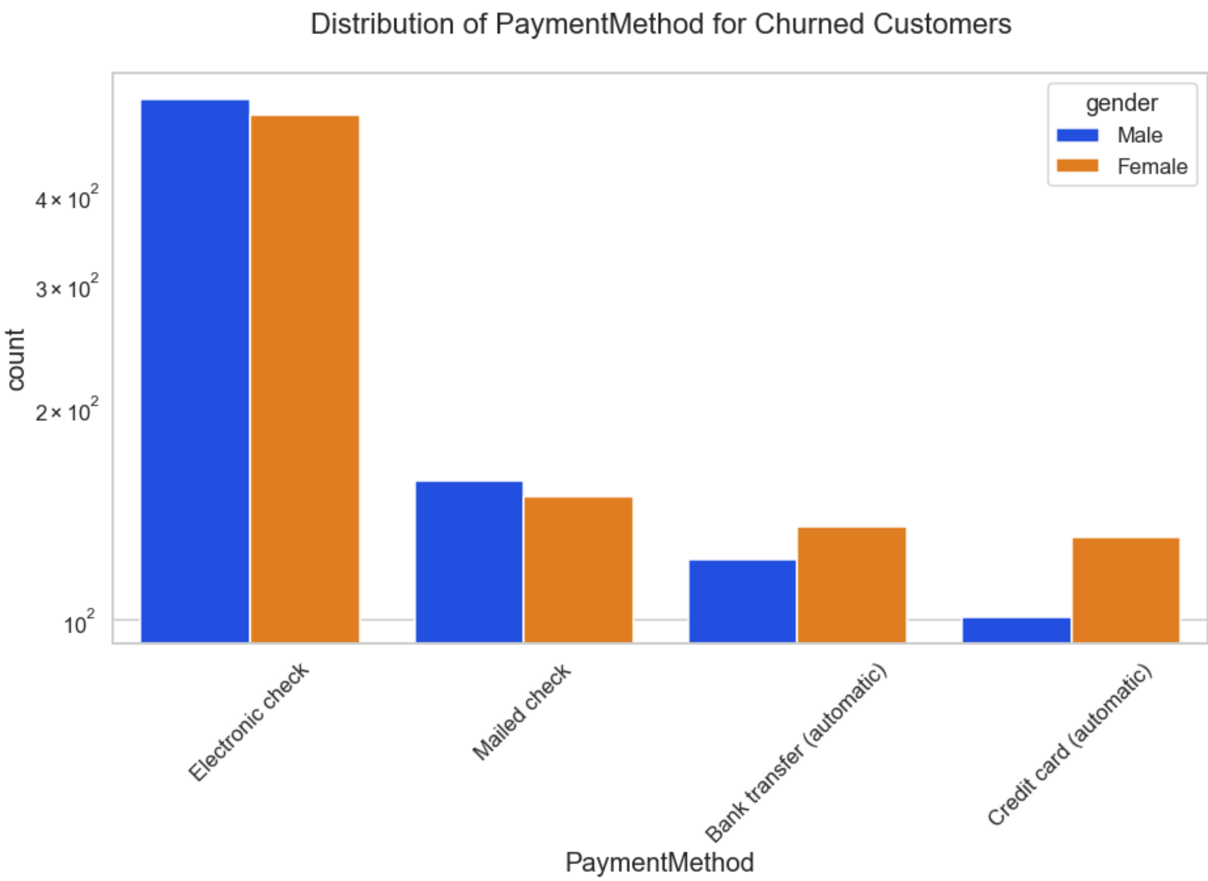
```
t1,col='Partner',title='Distribution of Gender for Churned Customers',hue='ge
```



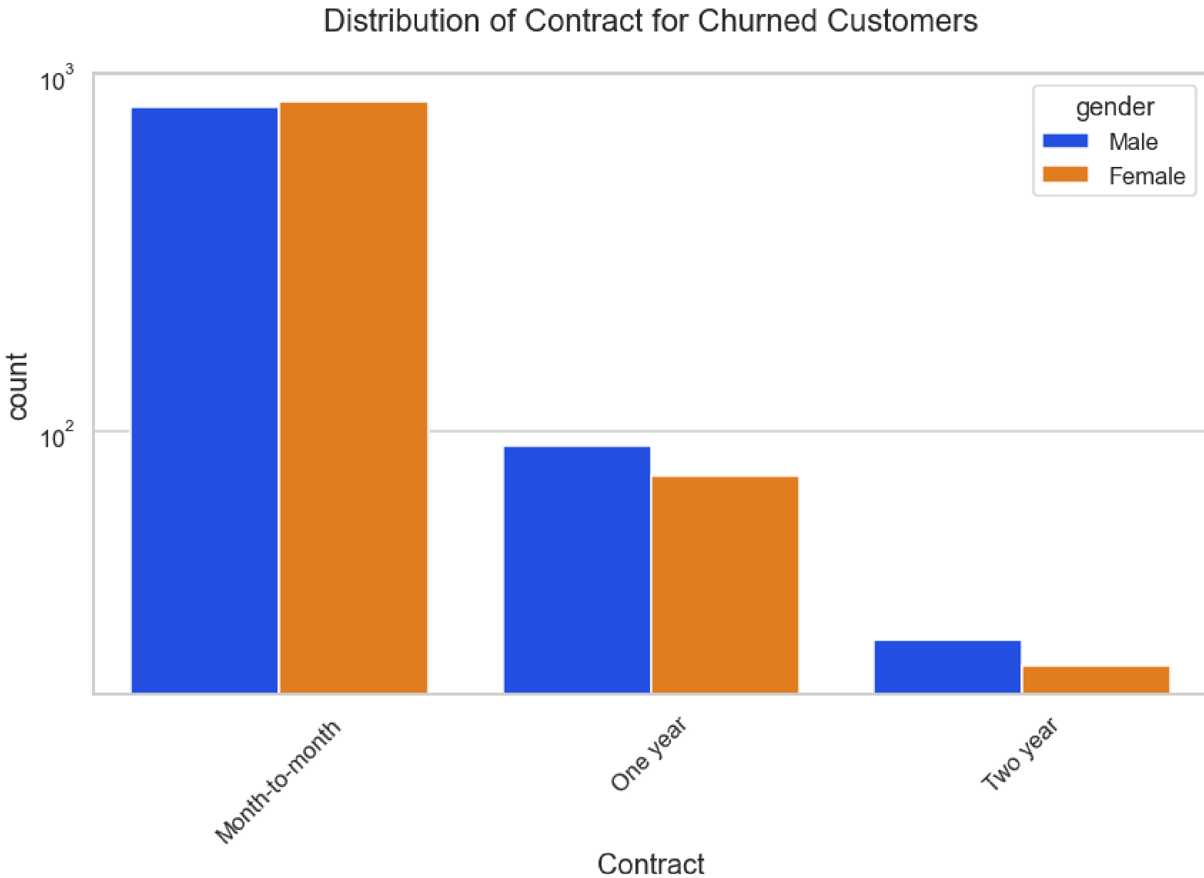
In [34]:
```
col='Partner',title='Distribution of Gender for Non Churned Customers',hue='ge
```



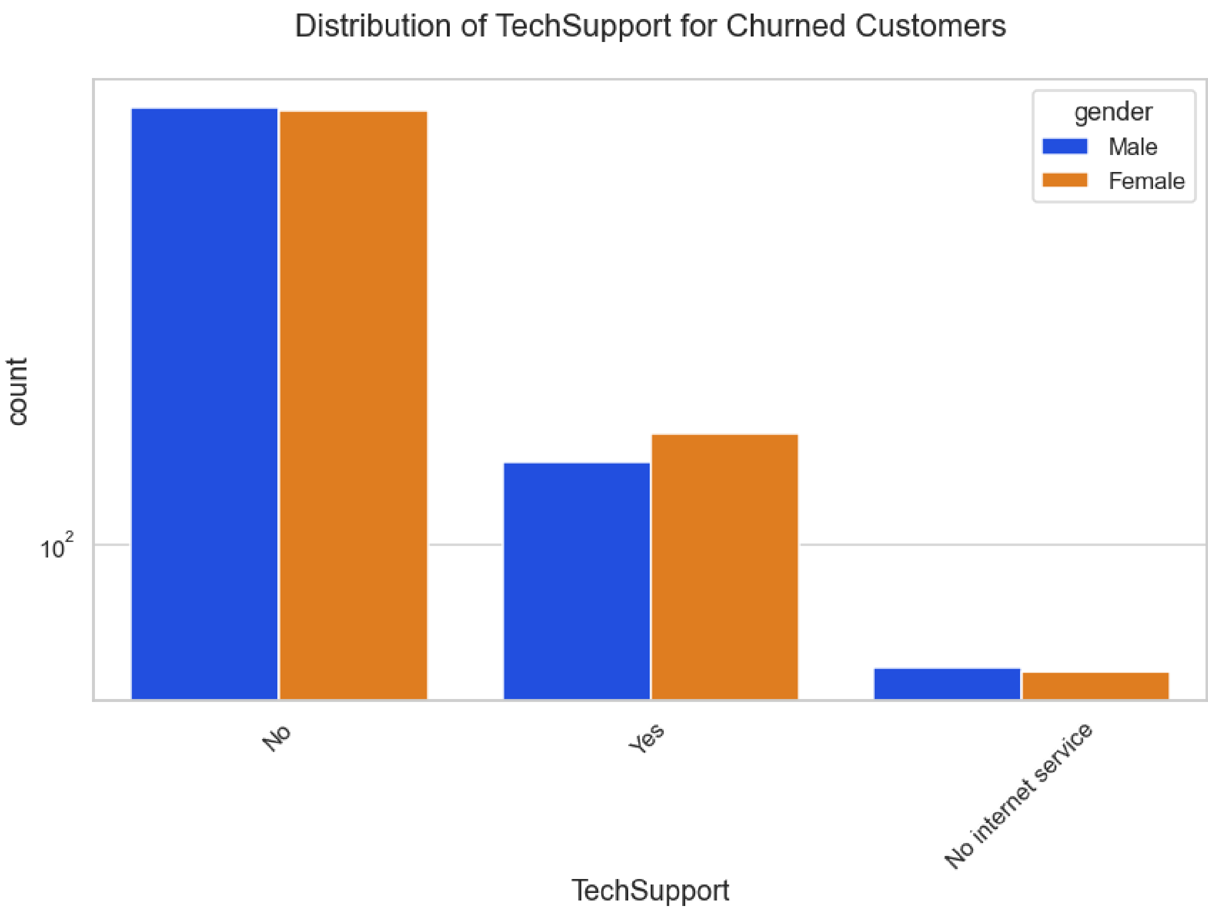
In [35]:
```
entMethod',title='Distribution of PaymentMethod for Churned Customers',hue='ge
```



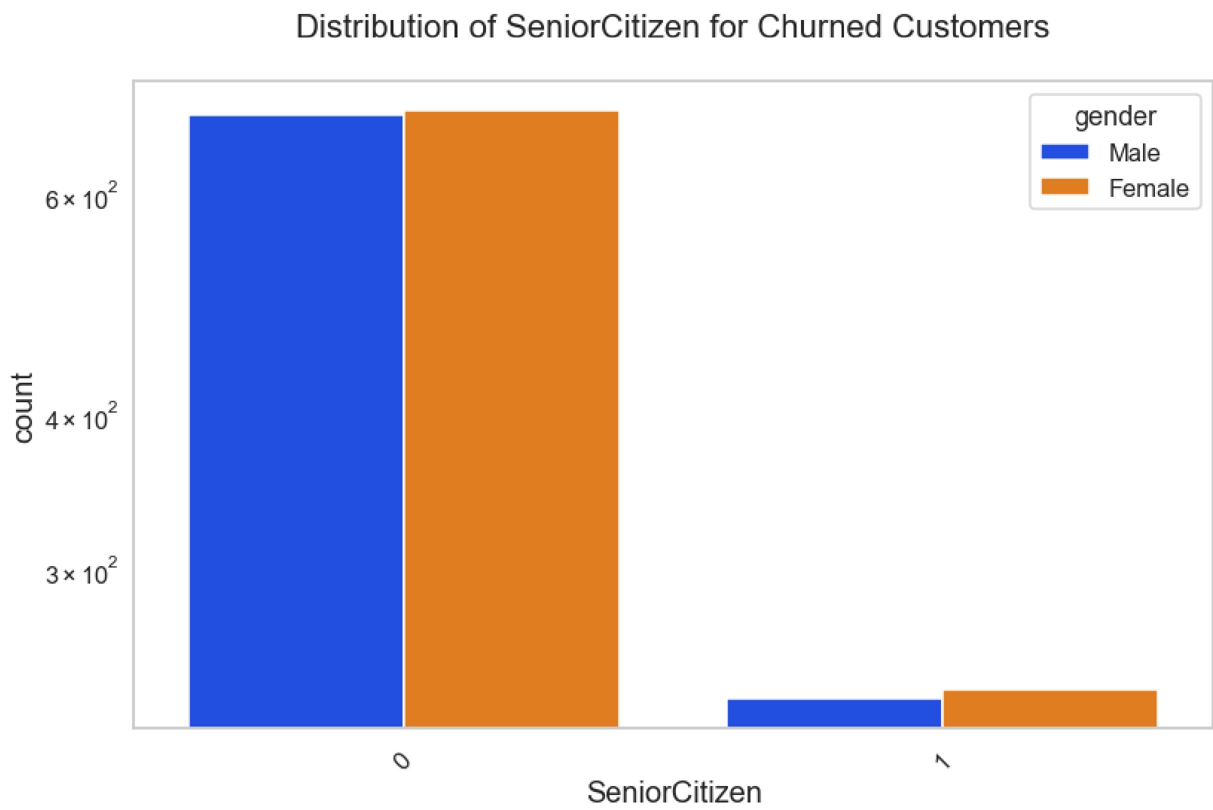
In [36]:
```
,col='Contract',title='Distribution of Contract for Churned Customers',hue='ge
```



In [37]:
```
TechSupport',title='Distribution of TechSupport for Churned Customers',hue='ge
```



In [38]:
```
orCitizen',title='Distribution of SeniorCitizen for Churned Customers',hue='ge
```




# 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

Note: There could be many more such insights, so take this as an assignment and try to get more insights :)

In [39]:
```
telco_data_dummies.to_csv('tel_churn.csv')

```

In [ ]:
```

```