Telco Churn Analysis

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

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('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

Look at the top 5 records of data

In [3]:

```
telco_base_data.head()
```

Out[3]:

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

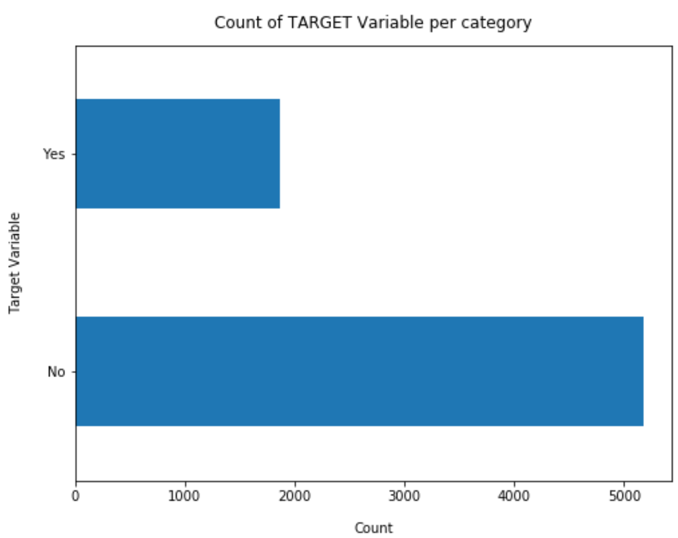
SeniorCitizen is actually a categorical hence the 25%-50%-75% distribution is not propoer

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 [9]:

```
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 [10]:

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

Out[10]:

```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

In [11]:

```python
telco_base_data['Churn'].value_counts()
```

Out[11]:

```
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.

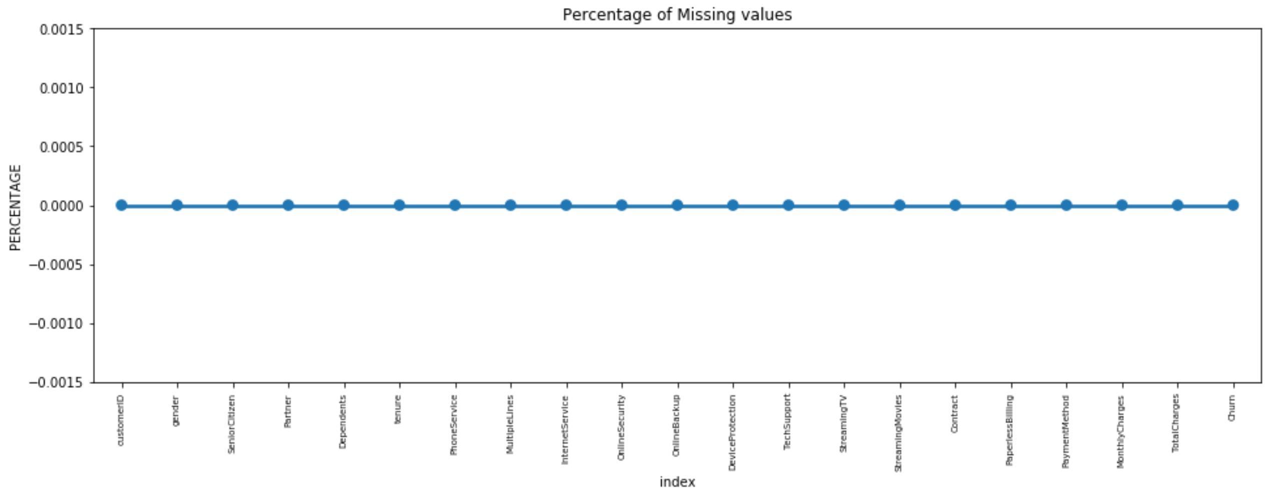
In [12]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
customerID          7043 non-null object
gender              7043 non-null object
SeniorCitizen       7043 non-null int64
Partner             7043 non-null object
Dependents          7043 non-null object
tenure              7043 non-null int64
PhoneService        7043 non-null object
MultipleLines       7043 non-null object
InternetService     7043 non-null object
OnlineSecurity      7043 non-null object
OnlineBackup        7043 non-null object
DeviceProtection    7043 non-null object
TechSupport         7043 non-null object
StreamingTV         7043 non-null object
StreamingMovies     7043 non-null object
Contract            7043 non-null object
PaperlessBilling    7043 non-null object
PaymentMethod       7043 non-null object
MonthlyCharges      7043 non-null float64
TotalCharges        7043 non-null object
Churn               7043 non-null object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [13]:

```python
missing = pd.DataFrame((telco_base_data.isnull().sum())*100/telco_base_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()
```



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 [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()
```

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
```

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

In [17]:

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

Out[17]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	Yes	No
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service

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 [18]:

```
#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 [22]:

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

```
72
```

In [23]:

```
# 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), right=False, labels=labels)
```

In [24]:

```
telco_data['tenure_group'].value_counts()
```

Out[24]:

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

6. Remove columns not required for processing

In [25]:

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

Out[25]:

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

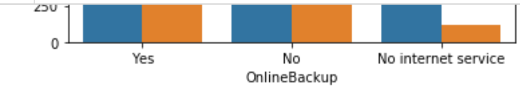
Data Exploration

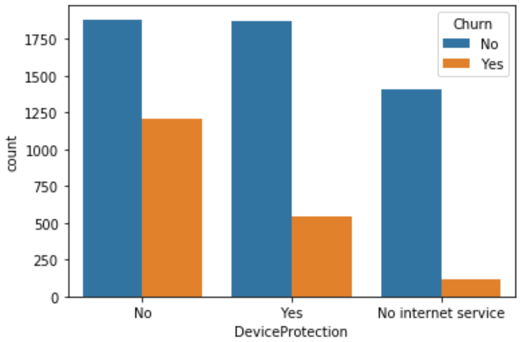
*1. * Plot distibution of individual predictors by churn

Univariate Analysis

In [26]:

```
for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
    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 [27]:

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

In [ ]:

```
telco_data.head()
```

Out[22]:

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

3. Convert all the categorical variables into dummy variables

In [28]:

```
telco_data_dummies = pd.get_dummies(telco_data)
telco_data_dummies.head()
```

Out[28]:

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

5 rows × 51 columns

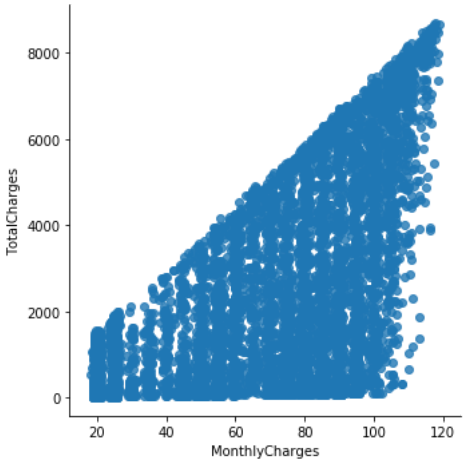
**9.* * Relationship between Monthly Charges and Total Charges

In [29]:

```python
sns.lmplot(data=telco_data_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

Out[29]:

```
<seaborn.axisgrid.FacetGrid at 0x2f17f19ef48>
```



Total Charges increase as Monthly Charges increase - as expected.

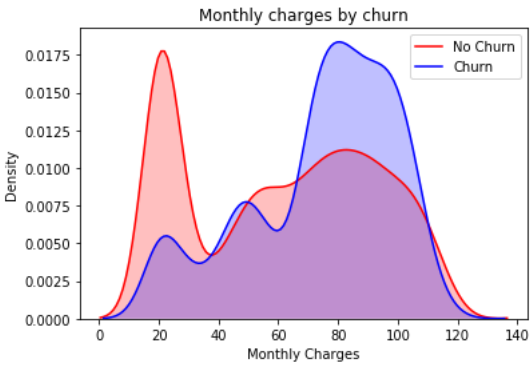
*10. * Churn by Monthly Charges and Total Charges

In [30]:

```python
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 0) ],
                color="Red", shade = True)
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_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[30]:

```
Text(0.5, 1.0, 'Monthly charges by churn')
```



Insight: Churn is high when Monthly Charges ar high

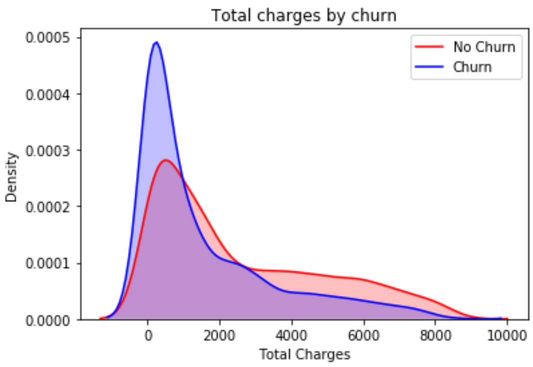
In [ ]:

```
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 0) ],
                color="Red", shade = True)
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_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')
```

```
C:\Users\pattn\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:444: RuntimeWarning: in
valid value encountered in greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # will not work for two columns.
C:\Users\pattn\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:444: RuntimeWarning: in
valid value encountered in less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # will not work for two columns.
```

Out[26]:

```
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**.

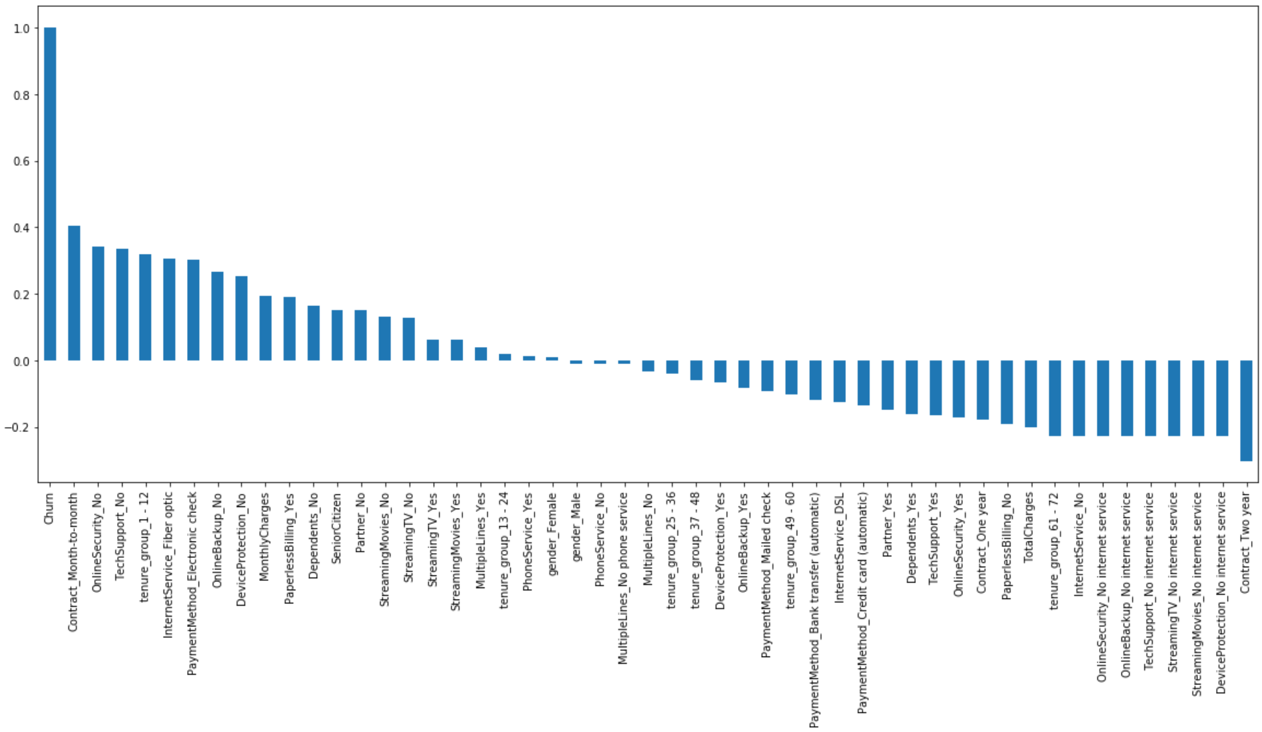
11. Build a corelation of all predictors with 'Churn'

In [31]:

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

Out[31]:

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x2f17f31f048&gt;
```



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 [ ]:

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

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1809ebfef60>
```

Type *Markdown* and LaTeX: α^2

Bivariate Analysis

In [ ]:

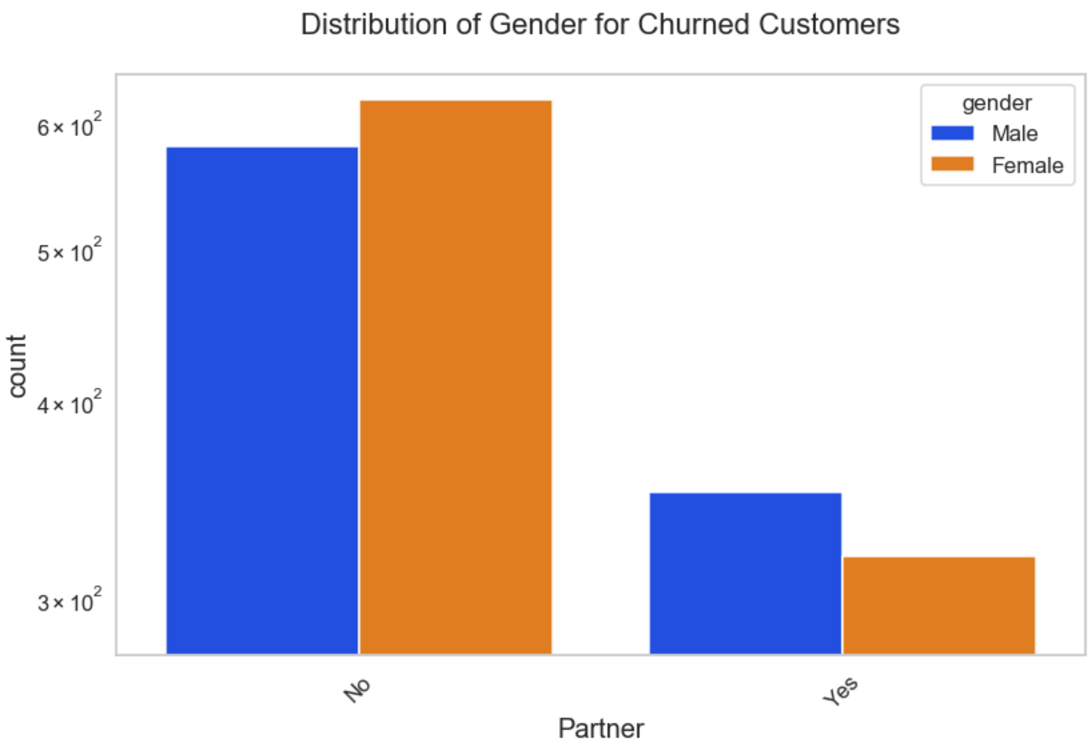
```
new_df1_target0=telco_data.loc[telco_data["Churn"]==0]
new_df1_target1=telco_data.loc[telco_data["Churn"]==1]
```

In [ ]:

```
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()
```

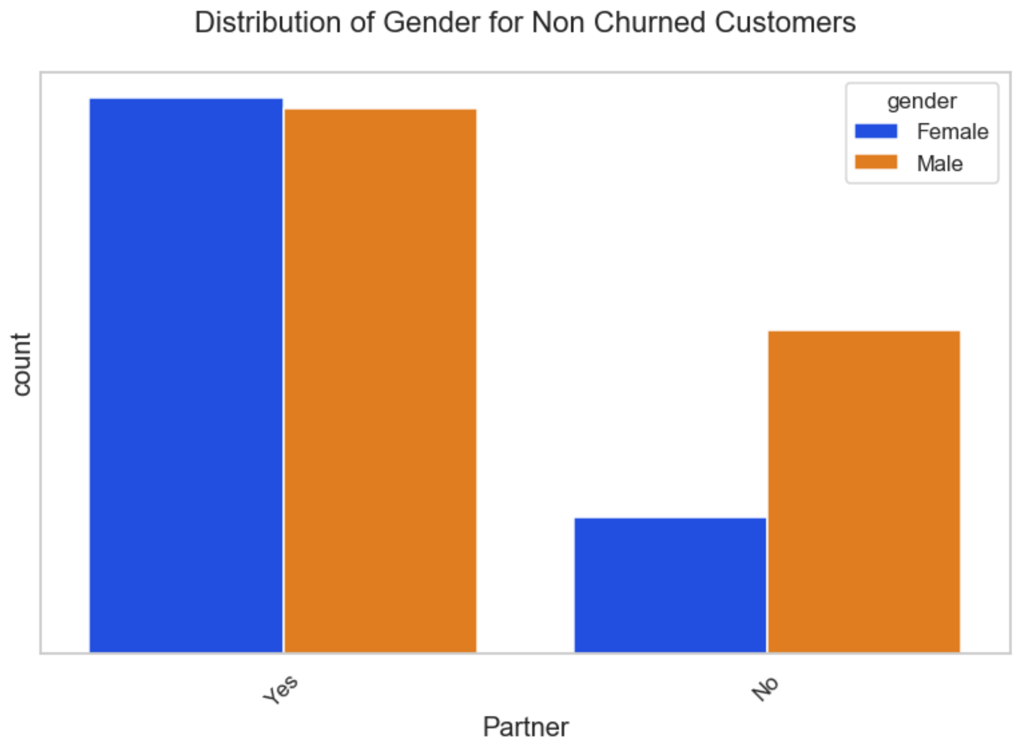
In [ ]:

```
uniplot(new_df1_target1,col='Partner',title='Distribution of Gender for Churned Customers',hue='gender')
```



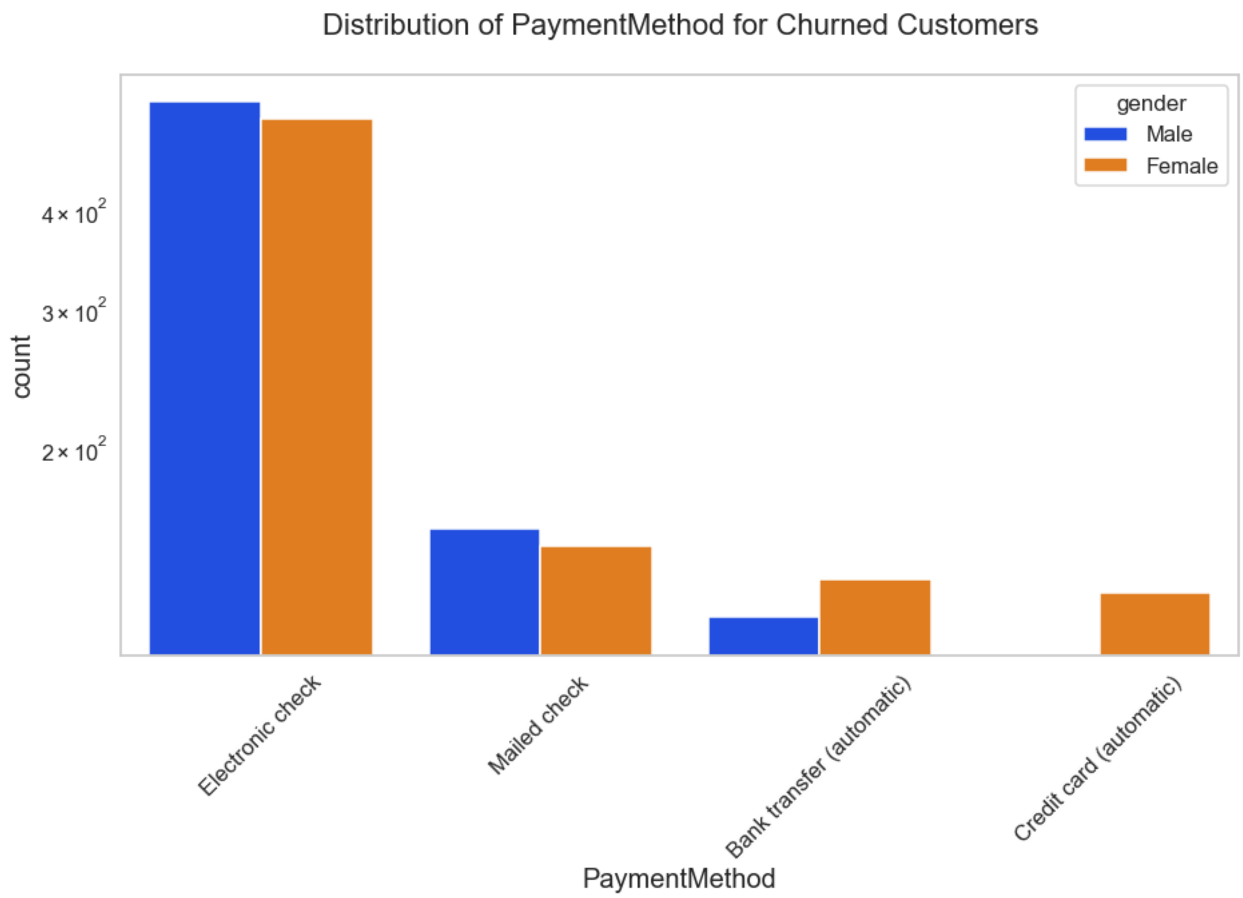
In [ ]:

```
uniplot(new_df1_target0,col='Partner',title='Distribution of Gender for Non Churned Customers',hue='gender')
```



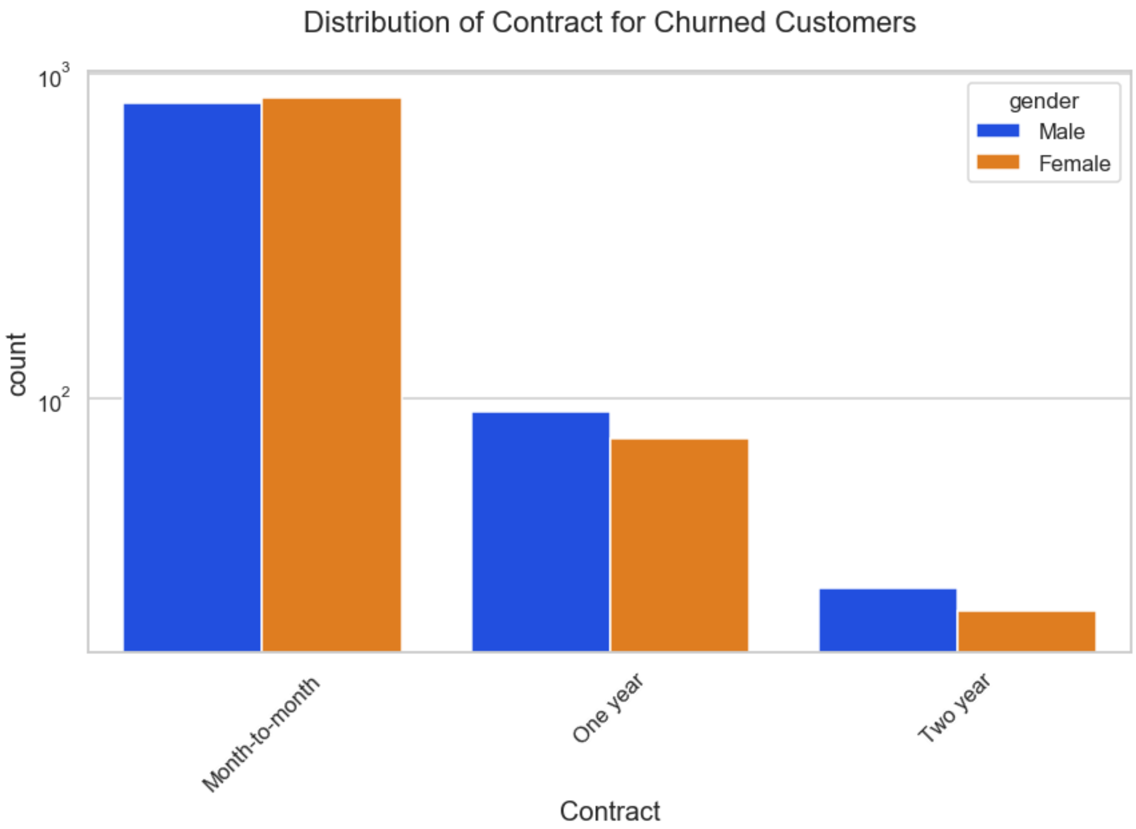
In [ ]:

```
uniplot(new_df1_target1,col='PaymentMethod',title='Distribution of PaymentMethod for Churned Customers',hue='gender')
```



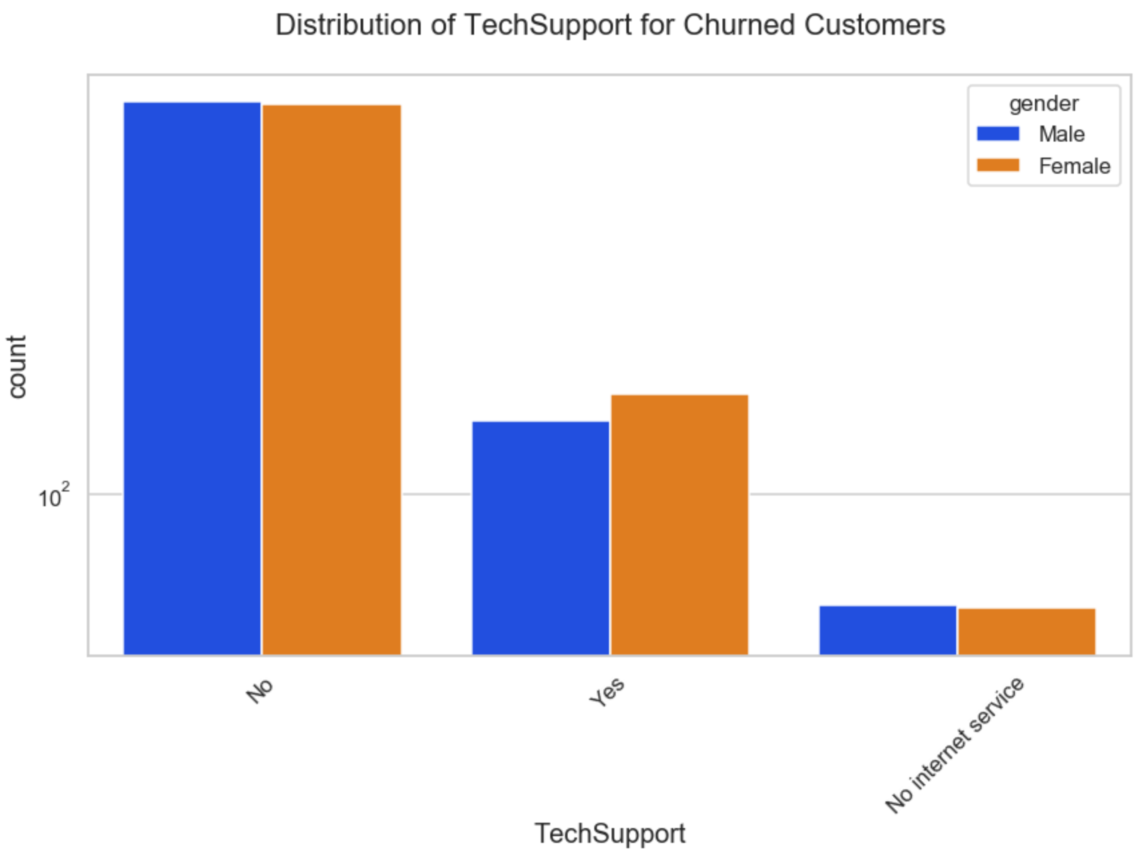
In [ ]:

```
uniplot(new_df1_target1,col='Contract',title='Distribution of Contract for Churned Customers',hue='gender')
```



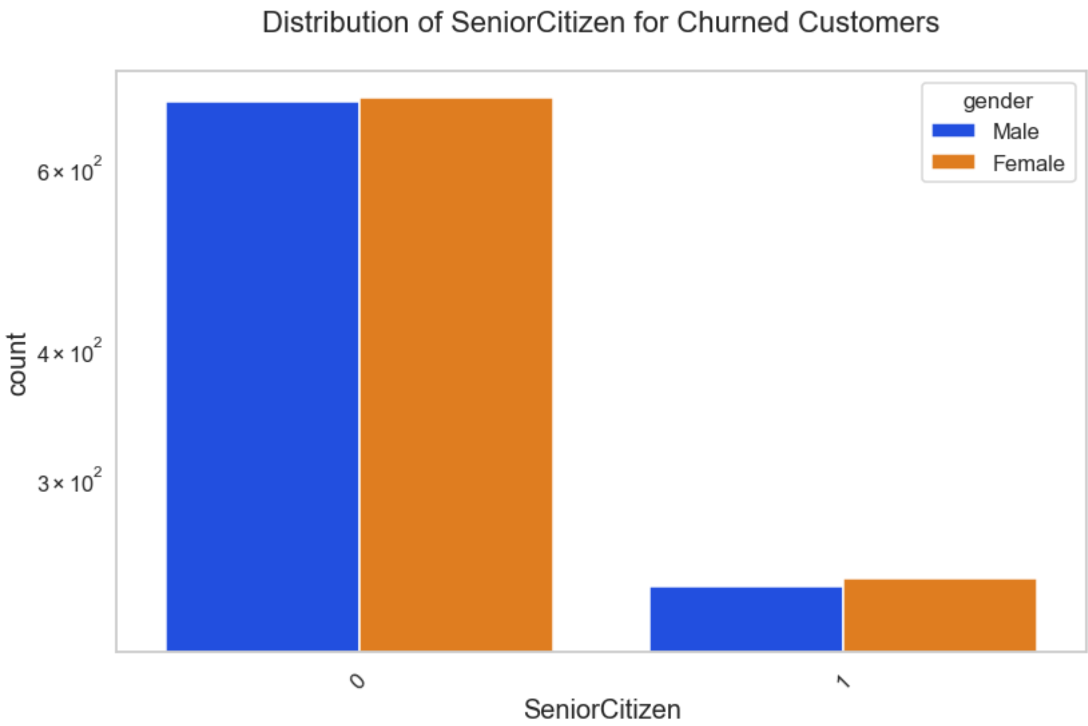
In [ ]:

```
uniplot(new_df1_target1,col='TechSupport',title='Distribution of TechSupport for Churned Customers',hue='gender')
```



In [ ]:

```
uniplot(new_df1_target1,col='SeniorCitizen',title='Distribution of SeniorCitizen for Churned Customers',hue='gender')
```



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 [33]:

```
telco_data_dummies.to_csv('tel_churn.csv')
```

In [ ]:

```

```