```
# Telecom Churn Case Study
# With 21 predictor variables we need to predict whether a particular customer will switch to another telecom provider or not. In telecom ter
```

```
# Step 1: Importing and Merging Data
```

```
# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')
```

```
# Importing Pandas and NumPy
import pandas as pd, numpy as np
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
%matplotlib inline
```

```
churn_data="C:\\Users\\Hp\\Downloads\\churn_data.csv"
```

```
# Importing all datasets
import pandas as pd
data="C:\\Users\\Hp\\Downloads\\churn_data.csv"
churn = pd.read_csv("C:\\Users\\Hp\\Downloads\\churn_data.csv")
```

```
customer_data = pd.read_csv("C:\\Users\\Hp\\Downloads\\customer_data.csv")
customer_data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents
0	7590-VHVEG	Female	0	Yes	No
1	5575-GNVDE	Male	0	No	No
2	3668-QPYBK	Male	0	No	No
3	7795-CFOCW	Male	0	No	No
4	9237-HQITU	Female	0	No	No

```
internet_data = pd.read_csv("C:\\Users\\Hp\\Downloads\internet_data.csv")
internet_data.head()
```

	customerID	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Strea
0	7590-VHVEG	No phone service	DSL	No	Yes	No	No	
1	5575-GNVDE	No	DSL	Yes	No	Yes	No	
2	3668-QPYBK	No	DSL	Yes	Yes	No	No	
	7795-	No phone						

```
# Combining all data files into one consolidated dataframe
```

```
# Merging on 'customerID'
df_1 = pd.merge(churn, customer_data, how='inner', on='customerID')
```

```
# Final dataframe with all predictor variables
telecom = pd.merge(df_1, internet_data, how='inner', on='customerID')
```

```
# Step 2: Inspecting the Dataframe
```

```
# Let's see the head of our master dataset
telecom.head()
```

ustomerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	gei
7590-VHVEG	1	No	Month-to-month	Yes	Electronic check	29.85	29.85	No	Fei
5575-GNVDE	34	Yes	One year	No	Mailed check	56.95	1889.5	No	I
3668-QPYBK	2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Yes	I
7795-CFOCW	45	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No	I
9237-HQITU	2	Yes	Month-to-month	Yes	Electronic check	70.70	151.65	Yes	Fei

```
# Let's check the dimensions of the dataframe
telecom.shape
```

```
(7043, 21)
```

```
# let's look at the statistical aspects of the dataframe
telecom.describe()
```

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

```
# 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
```

```
# Let's see the type of each column
telecom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   tenure            7043 non-null   int64  
 2   PhoneService      7043 non-null   object 
 3   Contract          7043 non-null   object 
 4   PaperlessBilling  7043 non-null   object 
 5   PaymentMethod     7043 non-null   object 
 6   MonthlyCharges    7043 non-null   float64
 7   TotalCharges      7043 non-null   object 
 8   Churn             7043 non-null   object 
 9   gender            7043 non-null   object 
 10  SeniorCitizen     7043 non-null   int64  
 11  Partner           7043 non-null   object 
 12  Dependents        7043 non-null   object 
 13  MultipleLines     7043 non-null   object 
 14  InternetService   7043 non-null   object 
 15  OnlineSecurity    7043 non-null   object 
 16  OnlineBackup      7043 non-null   object 
 17  DeviceProtection  7043 non-null   object 
 18  TechSupport       7043 non-null   object 
 19  StreamingTV       7043 non-null   object 
 20  StreamingMovies   7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.2+ MB
```

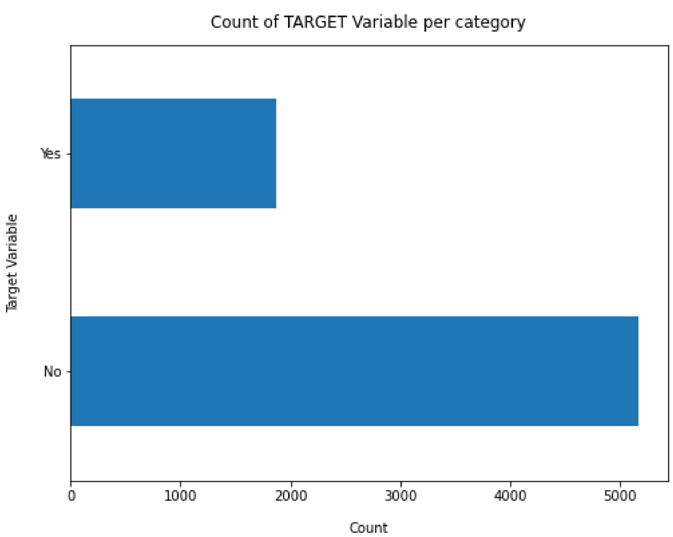
```
telecom.columns.values
```

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

```
telecom.dtypes
```

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

```
telecom['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);
```




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

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

```
telecom['Churn'].value_counts()
```

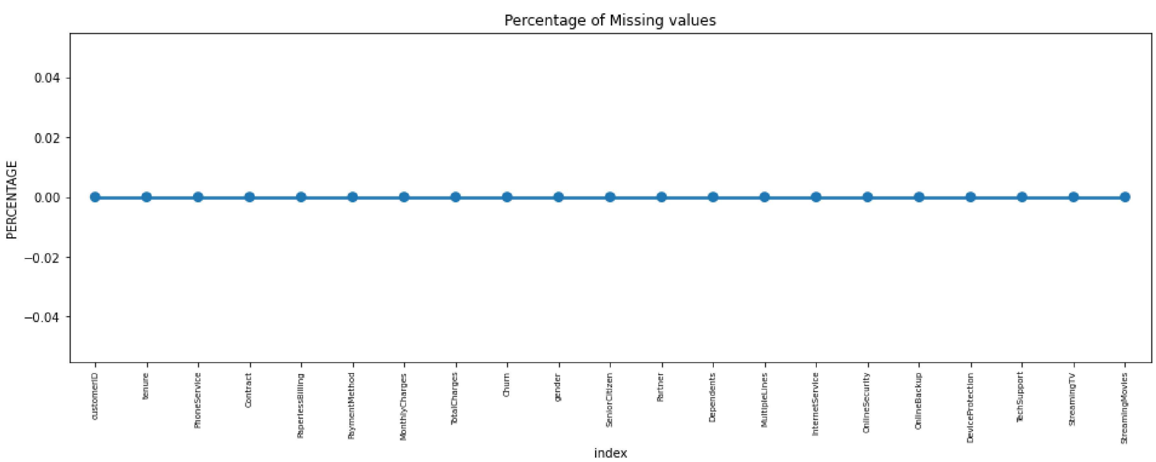
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

```
# 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.
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   tenure            7043 non-null   int64
 2   PhoneService      7043 non-null   object
 3   Contract          7043 non-null   object
 4   PaperlessBilling  7043 non-null   object
 5   PaymentMethod     7043 non-null   object
 6   MonthlyCharges    7043 non-null   float64
 7   TotalCharges      7043 non-null   object
 8   Churn             7043 non-null   object
 9   gender            7043 non-null   object
 10  SeniorCitizen     7043 non-null   int64
 11  Partner           7043 non-null   object
 12  Dependents        7043 non-null   object
 13  MultipleLines     7043 non-null   object
 14  InternetService   7043 non-null   object
 15  OnlineSecurity    7043 non-null   object
 16  OnlineBackup      7043 non-null   object
 17  DeviceProtection  7043 non-null   object
 18  TechSupport       7043 non-null   object
 19  StreamingTV       7043 non-null   object
 20  StreamingMovies   7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.2+ MB
```

```
missing = pd.DataFrame((telecom.isnull().sum())*100/telecom.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, depend
# 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 col
```

```
# Data Cleaning
# 1. Create a copy of base data for manupulation & processing
```

```
telco_data = telecom.copy()
```

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

```
telco_data.TotalCharges = pd.to_numeric(telco_data.TotalCharges, errors='coerce')
telco_data.isnull().sum()
```

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

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

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

	customerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
488	4472-LVYGI	0	No	Two year	Yes	Bank transfer (automatic)	52.55	NaN
753	3115-CZMZD	0	Yes	Two year	No	Mailed check	20.25	NaN
936	5709-LVOEQ	0	Yes	Two year	No	Mailed check	80.85	NaN
1082	4367-NUYAO	0	Yes	Two year	No	Mailed check	25.75	NaN
1340	1371-DWPAZ	0	No	Two year	No	Credit card (automatic)	56.05	NaN
3331	7644-OMVMY	0	Yes	Two year	No	Mailed check	19.85	NaN
3826	3213-VVOLG	0	Yes	Two year	No	Mailed check	25.35	NaN
4380	2520-SGTTA	0	Yes	Two year	No	Mailed check	20.00	NaN
5218	2923-ARZLG	0	Yes	One year	Yes	Mailed check	19.70	NaN
6670	4075-WKNIU	0	Yes	Two year	No	Mailed check	73.35	NaN
6754	2775-SEFEE	0	Yes	Two year	Yes	Bank transfer (automatic)	61.90	NaN

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.
```

```
#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, te
```

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

```
72
```

```
# 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)
```

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

```
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
```

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

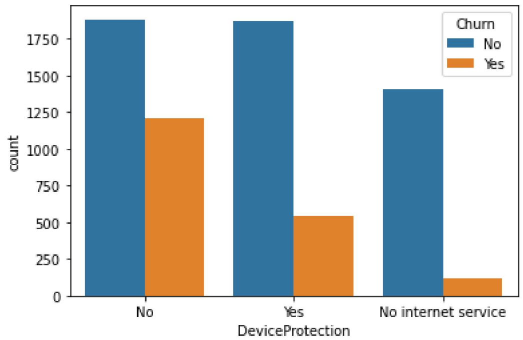
	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	gender	SeniorCi
0	No	Month-to-month	Yes	Electronic check	29.85	29.85	No	Female	
1	Yes	One year	No	Mailed check	56.95	1889.50	No	Male	
2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Yes	Male	
3	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No	Male	
4	Yes	Month-to-month	Yes	Electronic check	70.70	151.65	Yes	Female	

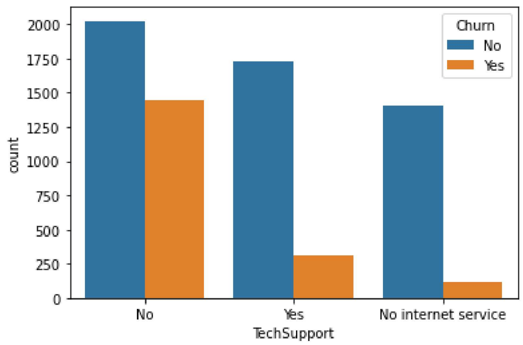
```
# Data Exploration
# 1. Plot distibution of individual predictors by churn
```

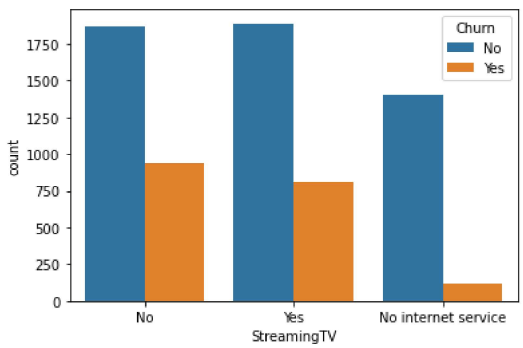
```
# Univariate Analysis
```

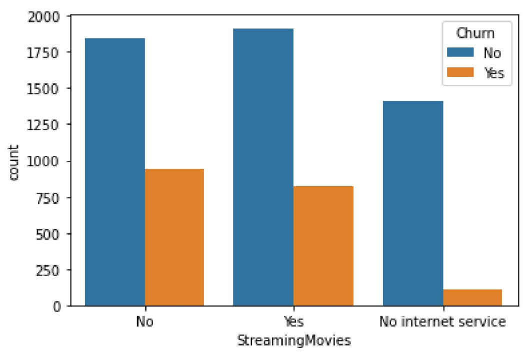
```
for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=telco_data, x=predictor, hue='Churn')
```

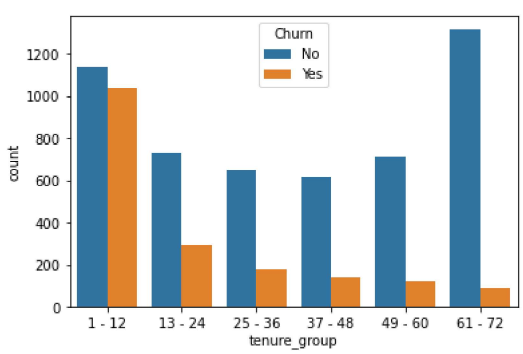

0
Yes
No
No internet service
OnlineBackup


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


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


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


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


Churn
No
Yes
count
0
200
400
600
800
1000
1200
1 - 12
13 - 24
25 - 36
37 - 48
49 - 60
61 - 72
tenure_group

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

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

telco_data.head()
```

	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	gender	SeniorCi
0	No	Month-to-month	Yes	Electronic check	29.85	29.85	0	Female	

#3. Convert all the categorical variables into dummy variables

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

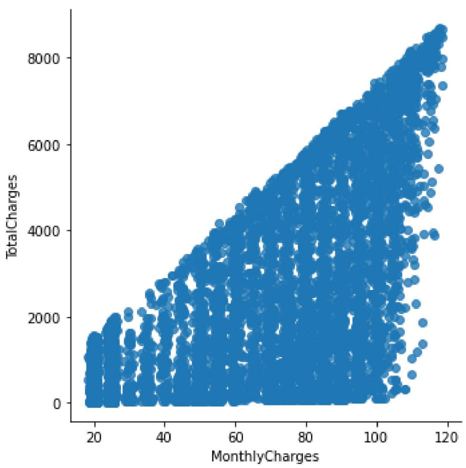
	MonthlyCharges	TotalCharges	Churn	SeniorCitizen	PhoneService_No	PhoneService_Yes	Contract_Month-to-month	Contra
0	29.85	29.85	0	0	1	0	1	
1	56.95	1889.50	0	0	0	1	0	
2	53.85	108.15	1	0	0	1	1	
3	42.30	1840.75	0	0	1	0	0	
4	70.70	151.65	1	0	0	1	1	

5 rows × 51 columns

#9. Relationship between Monthly Charges and Total Charges

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

```
<seaborn.axisgrid.FacetGrid at 0x2c2f3de1fa0>
```



Total Charges increase as Monthly Charges increase - as expected.

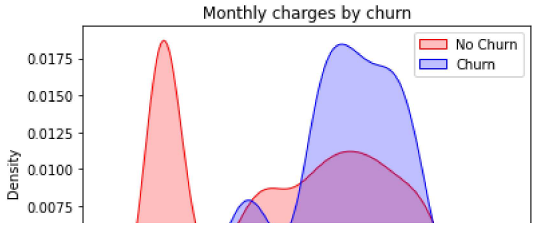
10. Churn by Monthly Charges and Total Charges

```
  Input In [103]
    Total Charges increase as Monthly Charges increase - as expected.
          ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

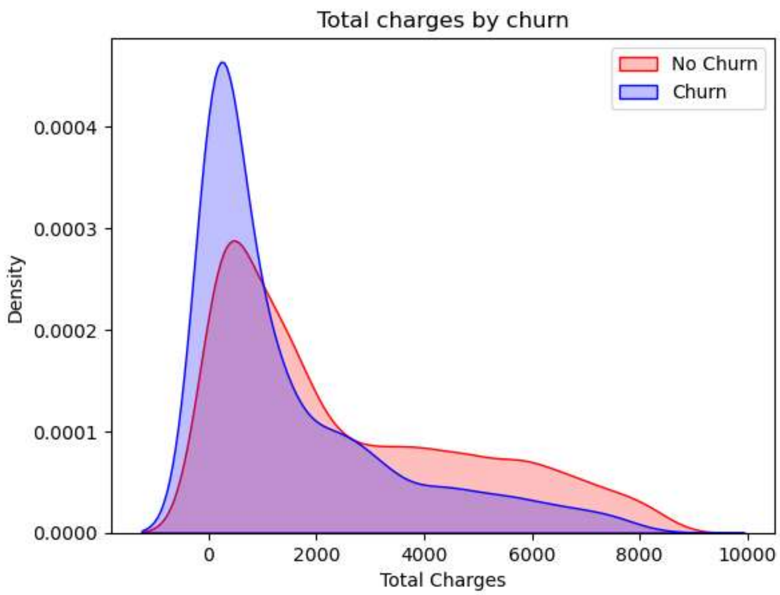
```
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 0) ],
                color="Red", shade = True)
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 1) ],
                ax =Mth, color="Blue", shade= True)
Mth.legend(["No Churn","Churn"],loc='upper right')
Mth.set_ylabel('Density')
Mth.set_xlabel('Monthly Charges')
Mth.set_title('Monthly charges by churn')
```

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




```
#Insight: Churn is high when Monthly Charges ar high

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')
```

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

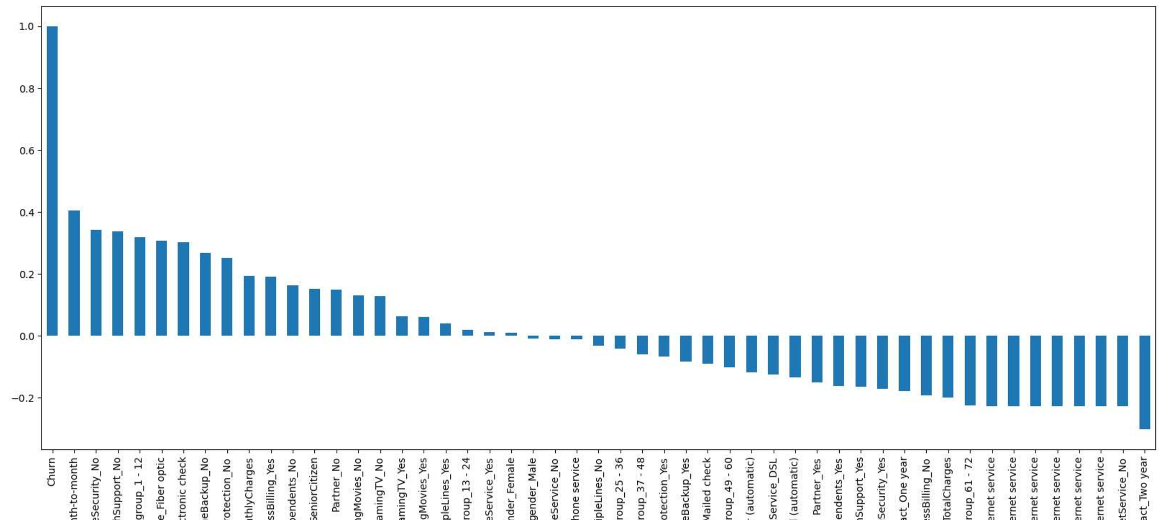


Surprising insight as higher Churn at lower Total Charges

However if we combine the insights of 3 parameters i.e. Tenure, Monthly Charges & Total Charges then the picture is bit clear :- Higher Month

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

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

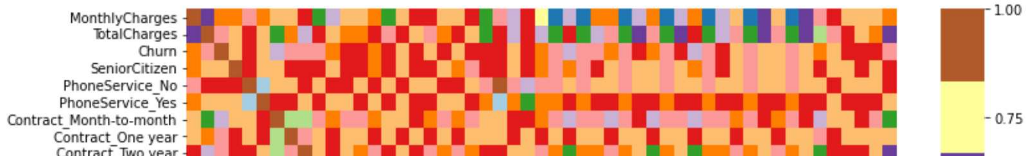
<AxesSubplot:>



```
# 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 Inter

# 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
```

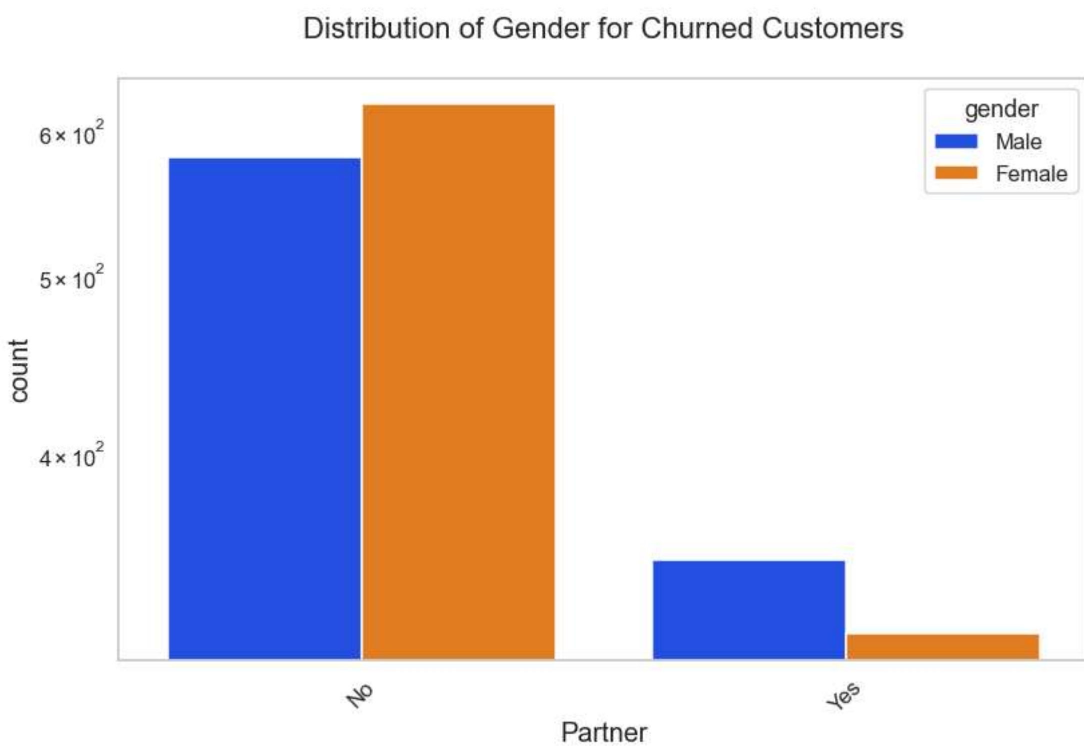
```
plt.figure(figsize=(12,12))
sns.heatmap(telco_data_dummies.corr(), cmap="Paired")
```

```
<AxesSubplot:>
```

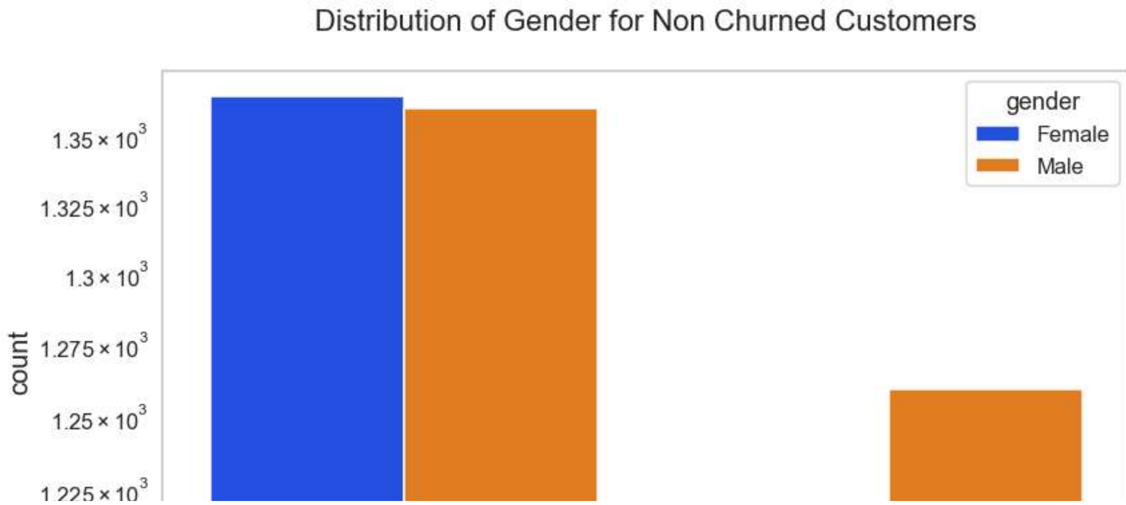



```
#Bivariate Analysis
new_df1_target0=telco_data.loc[telco_data["Churn"]==0]
new_df1_target1=telco_data.loc[telco_data["Churn"]==1]
def uniplot(df,col,title,hue =None):

    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()
```

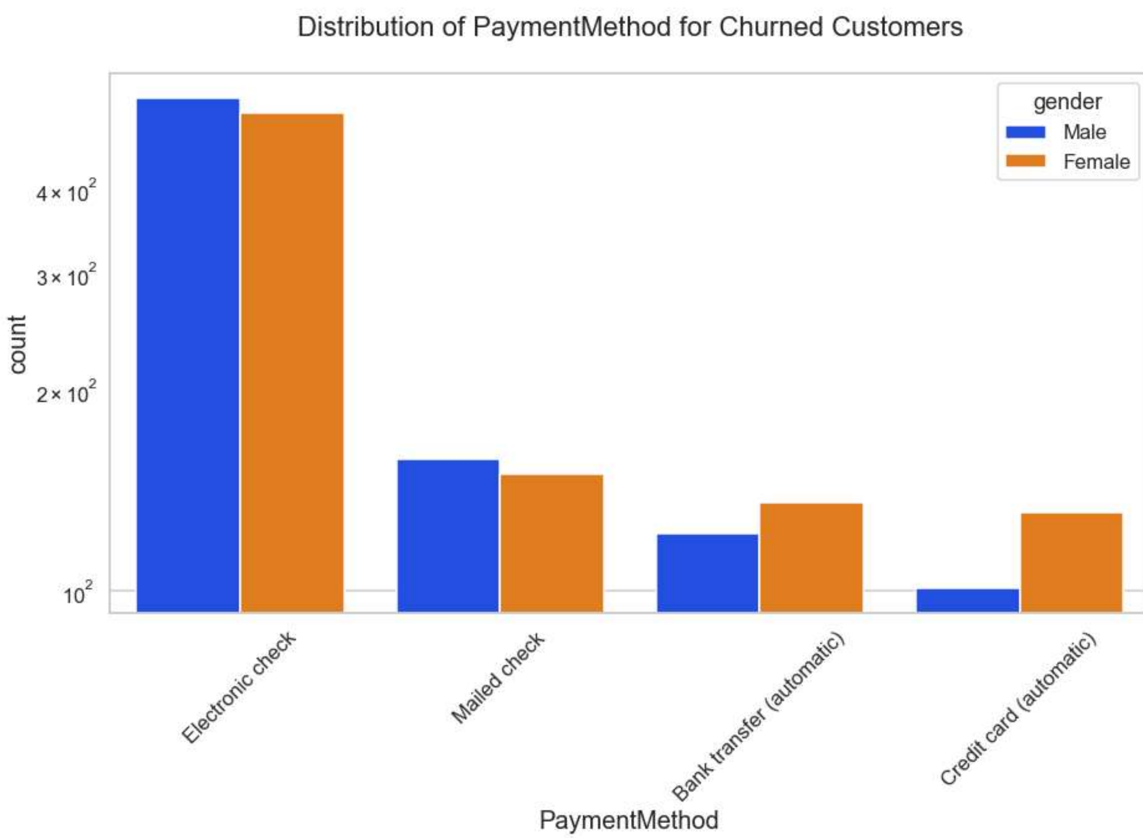

```
uniplot(new_df1_target1,col='Partner',title='Distribution of Gender for Churned Customers',hue='gender')
```



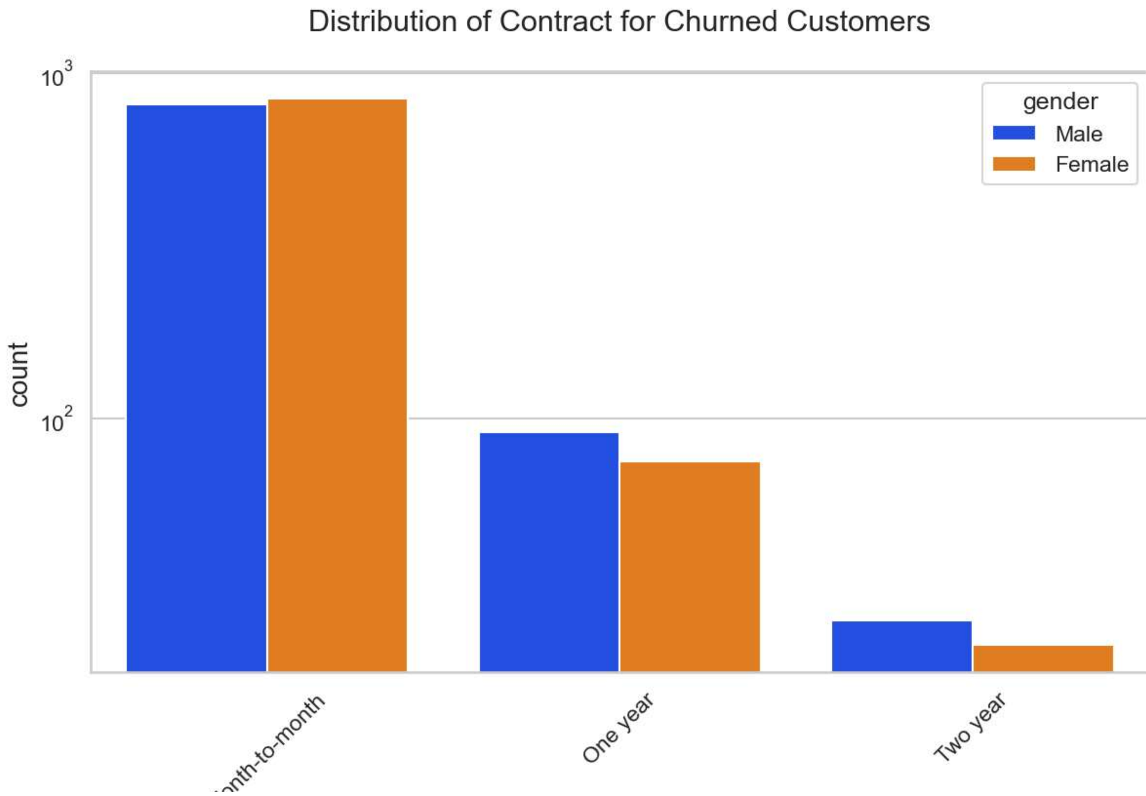

```
uniplot(new_df1_target0,col='Partner',title='Distribution of Gender for Non Churned Customers',hue='gender')
```



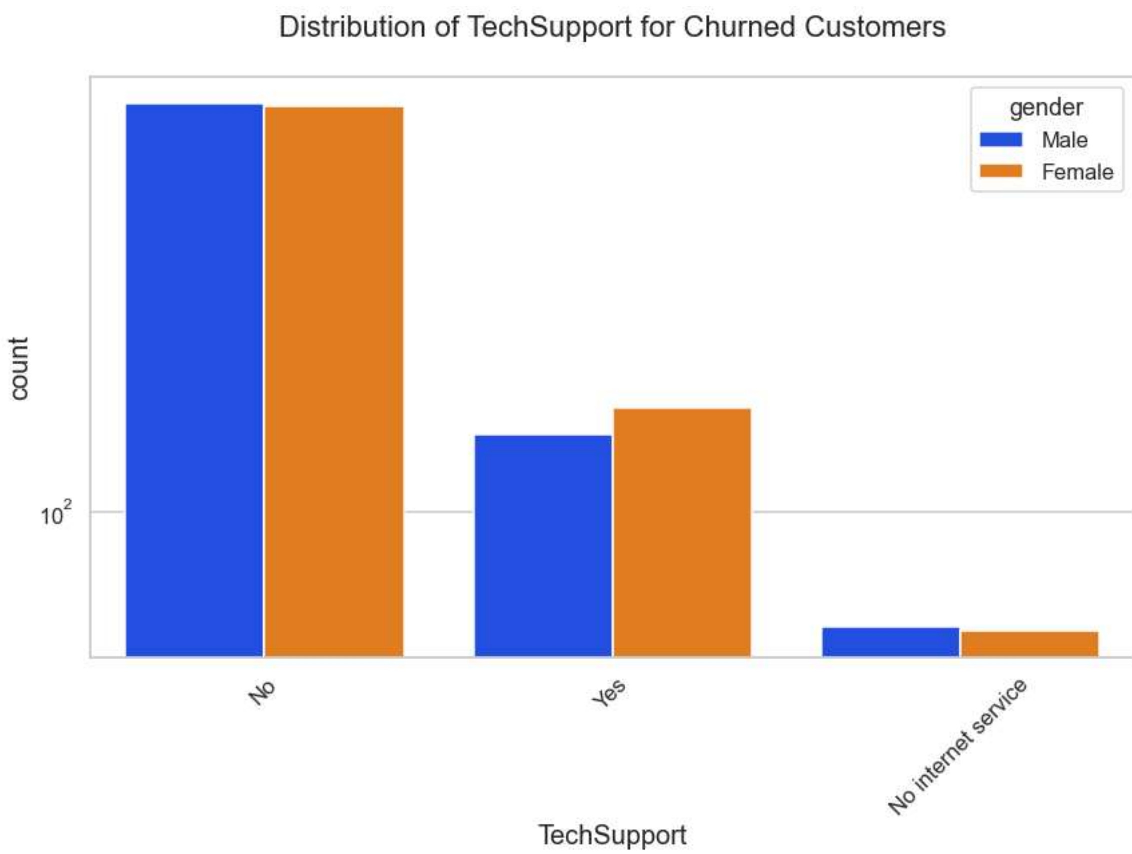

```
uniplot(new_df1_target1,col='PaymentMethod',title='Distribution of PaymentMethod for Churned Customers',hue='gender')
```




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




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




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