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

Loading the data file

In [2]:
```python
df = pd.read_csv("Customer_churn_data.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

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

5 rows × 21 columns

In [ ]:

In [4]:
```python
df.columns.values
```

Out[4]:
```
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 [5]:
```
df.dtypes
```

Out[5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	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

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
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 [8]:

```
df.describe(include = "all")
```

Out[8]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
count	7043	7043	7043.000000	7043	7043	7043.000000	704
unique	7043	2	NaN	2	2	NaN	
top	7590-VHVEG	Male	NaN	No	No	NaN	Ye
freq	1	3555	NaN	3641	4933	NaN	636
mean	NaN	NaN	0.162147	NaN	NaN	32.371149	Na
std	NaN	NaN	0.368612	NaN	NaN	24.559481	Na
min	NaN	NaN	0.000000	NaN	NaN	0.000000	Na
25%	NaN	NaN	0.000000	NaN	NaN	9.000000	Na
50%	NaN	NaN	0.000000	NaN	NaN	29.000000	Na
75%	NaN	NaN	0.000000	NaN	NaN	55.000000	Na
max	NaN	NaN	1.000000	NaN	NaN	72.000000	Na

11 rows × 21 columns

In [9]:
```
df.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

In [10]:
```
df.tail()
```

Out[10]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multi
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	N
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	
7042	3186-AJIEK	Male	0	No	No	66	Yes	

5 rows × 21 columns

In [11]:
```
df = df.dropna(how = "all")
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSuppor
t',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [13]:

```
df["Churn"].value_counts()
```

Out[13]:

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

In [14]:

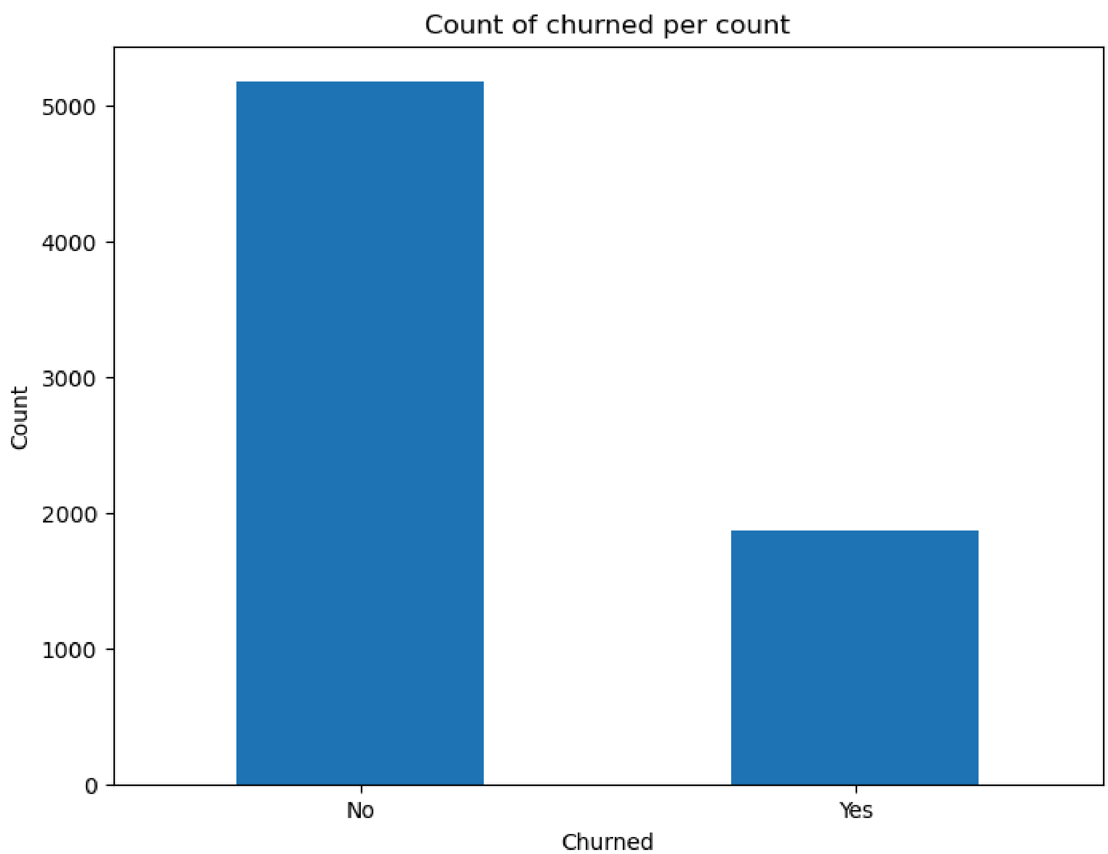
```
100*df["Churn"].value_counts()/len(df["Churn"])
```

Out[14]:

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

In [15]:

```
df["Churn"].value_counts().plot(kind = "bar", figsize = (8,6))
plt.xlabel("Churned")
plt.xticks(rotation = "horizontal")
plt.ylabel("Count")
plt.title("Count of churned per count")
plt.show()
```



By above we can see that the data is highly imbalanced in the ratio 73:27 so we analyze the data with other features while taking the largest values separately to get some more insights

In [16]:

```python
df.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
```

In [17]:

```python
df.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 [18]:
```
df.isnull().sum()
```

Out[18]:
```
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
```

we don't have missing values

Data Cleaning

Creating a copy of base data for manipulation and processing

In [19]:
```
data = df.copy()
```

Total charges should be numeric amount. Let's convert it into numerical data type

In [ ]:
```

```

In [20]:
```
data["TotalCharges"] = pd.to_numeric(data["TotalCharges"],errors = "coerce
```

In [21]:
```
data.isnull().sum()
```

Out[21]:
```
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
```

As we can see there are 11 missing values in TotalCharges column

In [22]:
```
data.loc[data["TotalCharges"].isnull()== True]
```

Out[22]:

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

11 rows × 21 columns

Missing value treatment

since the % of these records compared to total dataset is very low i.e. 0.15% it is safe to ignore them from further processing

In [23]:
```
# Removing the missing values
data.dropna(how = 'any', inplace = True)
```

Dividing the customers into bins based on tenure e.g. for tenure < 12 months assign a tenure group 1-12 for tenure group between 1 to 2 years tenure group of 13-24 and so on

In [24]:
```python
# get the max tenure
print(data["tenure"].max())
```

```
72
```

In [25]:
```python
# Group the tenure in the bins of 12 months
labels = [f"{i}-{i+11}" for i in range(1,72,12)]
```

In [26]:
```python
labels
```

Out[26]:
```
['1-12', '13-24', '25-36', '37-48', '49-60', '61-72']
```

In [27]:
```python
data["tenure_group"] = pd.cut(data["tenure"], range(1,80,12), right = Fals
```

In [28]:
```python
data["tenure_group"].value_counts()
```

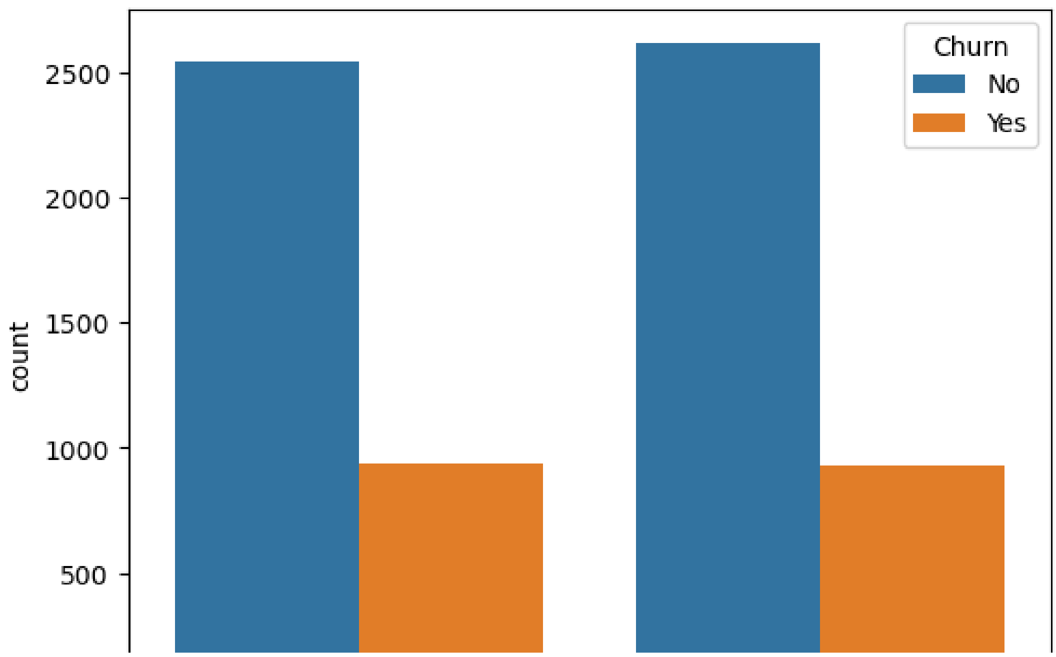
Out[28]:
```
1-12     2175
61-72    1407
13-24    1024
25-36     832
49-60     832
37-48     762
Name: tenure_group, dtype: int64
```

Remove columns not required for processing

In [29]:
```python
# drop column customerID and tenure
data.drop(columns= ["customerID", "tenure"], axis = 1, inplace = True)
```

Data Exploration

In [30]:
```
# Univariate analysis
for i , predictor in enumerate(data.drop(columns = ["Churn", "TotalCharges
    plt.figure(i)
    sns.countplot(data = data , x = predictor, hue = "Churn")
```



In [31]:
```
# Convert the target variable "Churn" in a binary numeric variable i.e. ye
```

In [32]:
```
data["Churn"] = np.where(data.Churn == "Yes", 1, 0)
```

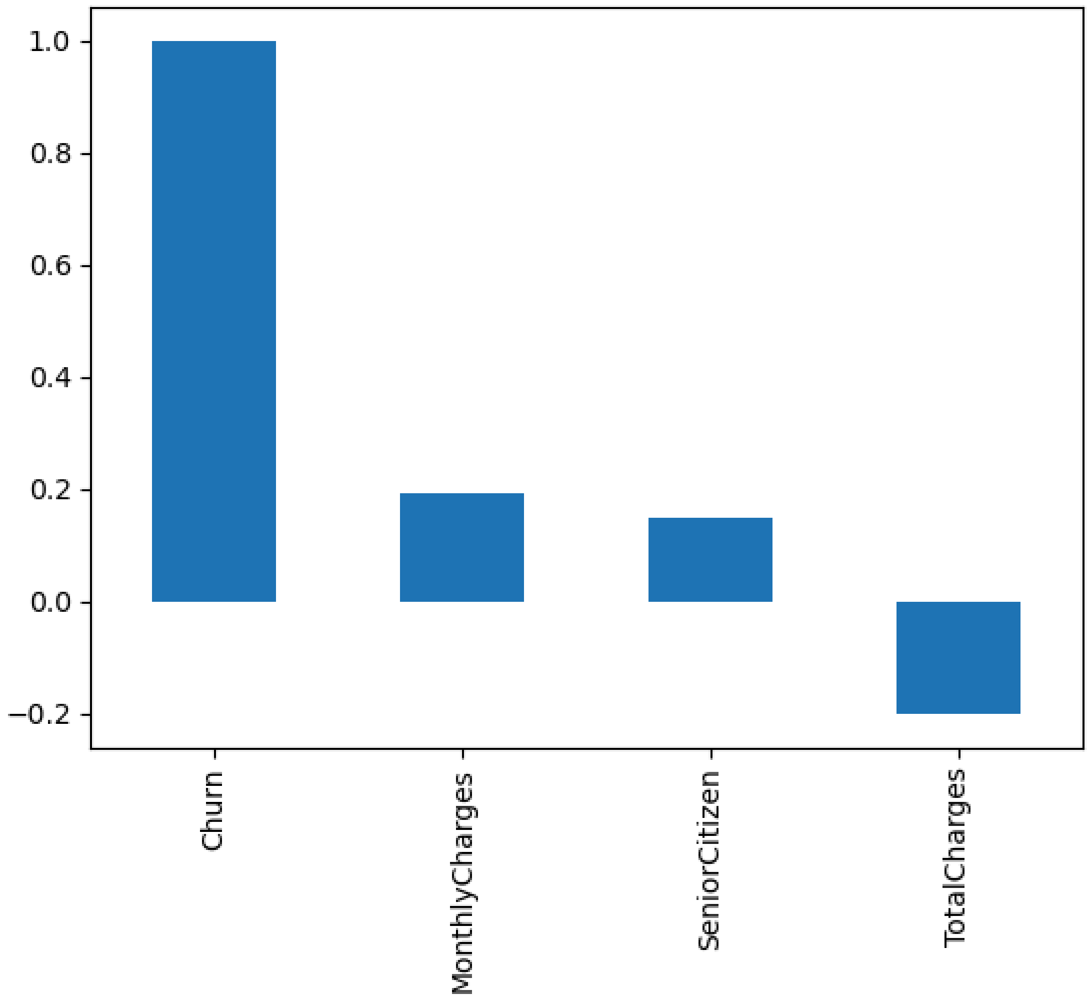
In [33]:
```
data.head()
```

Out[33]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
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 [34]:
```
data.corr()["Churn"].sort_values(ascending= False).plot(kind = "bar")
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_20092\2440828981.py:1: Future
Warning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid colum
ns or specify the value of numeric_only to silence this warning.
  data.corr()["Churn"].sort_values(ascending= False).plot(kind = "bar")
```



In [35]:
```
# converting all categorical variables into dummy variables
```

In [36]:
```
data_dummy = pd.get_dummies(data)
```

In [37]:
```
data_dummy.head(5)
```

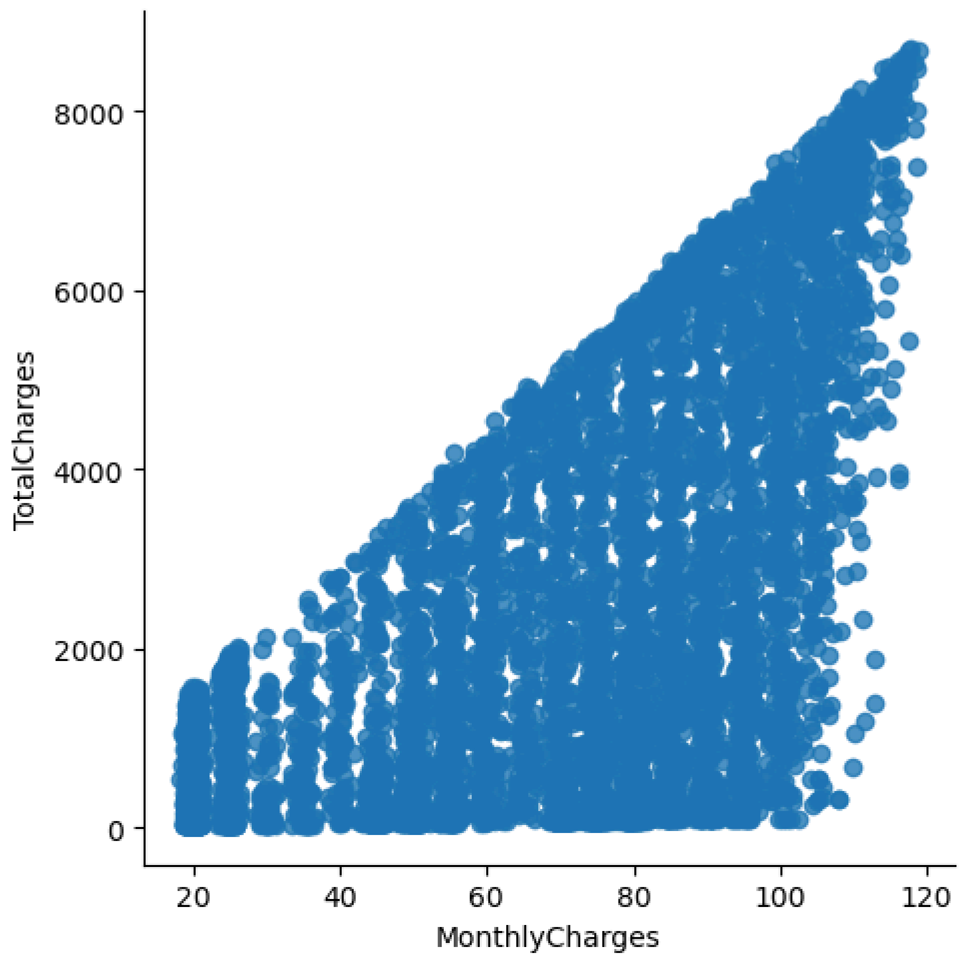
Out[37]:

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

5 rows × 51 columns

In [38]:
```
# Relationship between monthly charges and total charges
sns.lmplot(data = data_dummy, x = "MonthlyCharges", y = "TotalCharges", fi
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x19547e38c90>



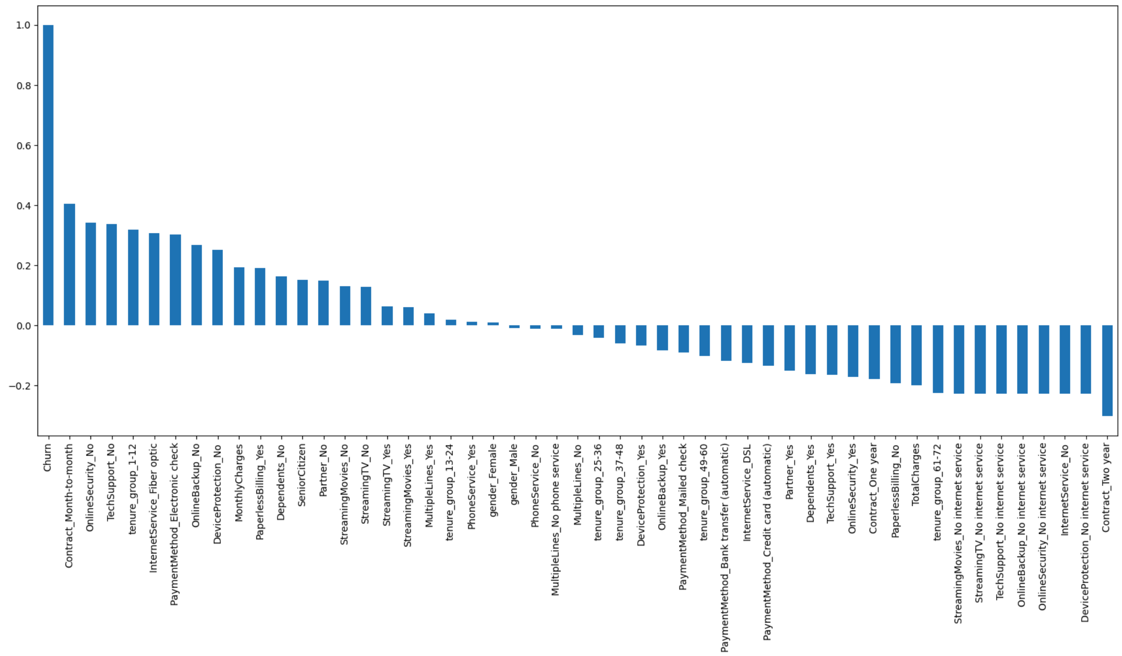
Total Charges increases as monthly charges increases as expected

Churn by Monthly Charges and Total Charges

In [39]:

```python
#Build a co relation of all predictors with churn
plt.figure(figsize= (20,8))
data_dummy.corr()["Churn"].sort_values(ascending = False).plot(kind = "bar
```

Out[39]: <Axes: >



In [40]:

```python
# Derived insights
# High churn is seen in case of month to month contracts, no_online securi
# and fiber optics internet

# Low churn is seen in case of long term contracts, subscriptions without
# 5+ years

# factors like gender, availability of phone service and # of multiple lin
```

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

Out[41]: <Axes: >

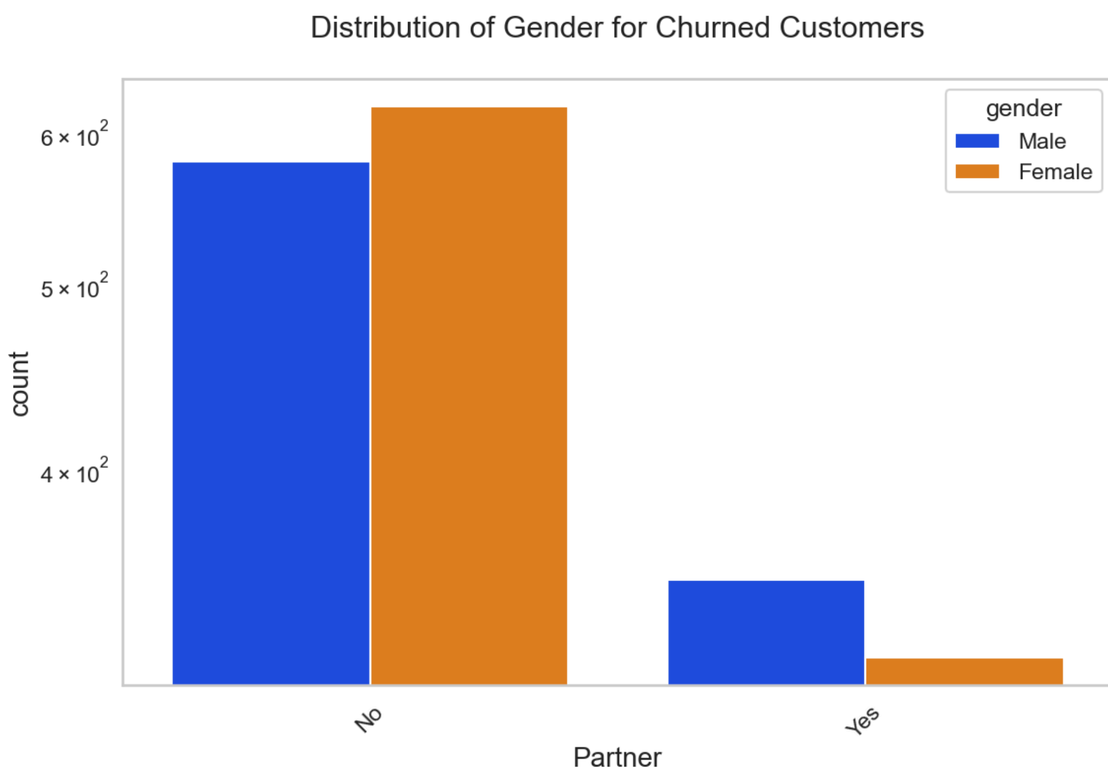
Bivariate Analysis

In [45]:
```python
new_data_target0 = data.loc[data["Churn"]== 0]
```

In [46]:
```python
new_data_target1 = data.loc[data["Churn"]== 1]
```

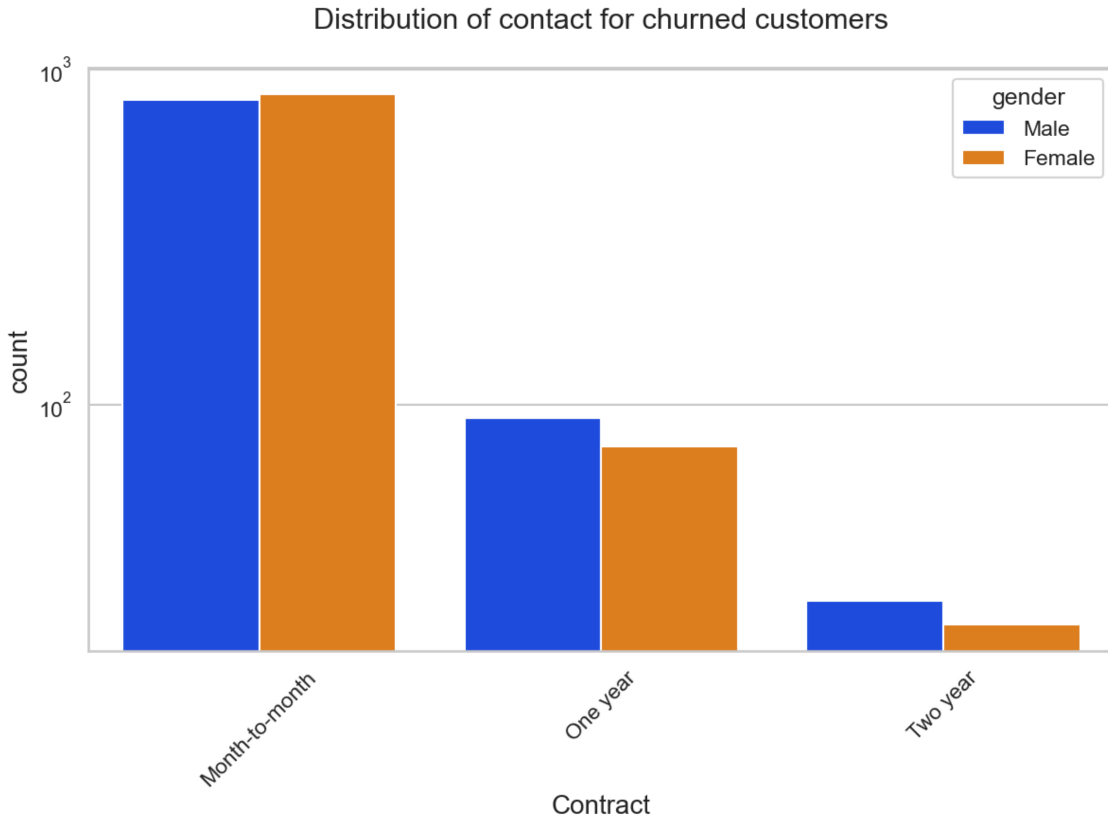
In [48]:
```
def uniplot(df,col,title, hue = None):
    sns.set_style("whitegrid")
    sns.set_context("talk")
    plt.rcParams["axes.labelsize"] = 20
    plt.rcParams["axes.titlesize"] = 22
    plt.rcParams["axes.titlepad"] = 30

    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().
    plt.show()
```

In [52]:
```
uniplot(new_data_target1, col = "Partner", title = "Distribution of Gender
```



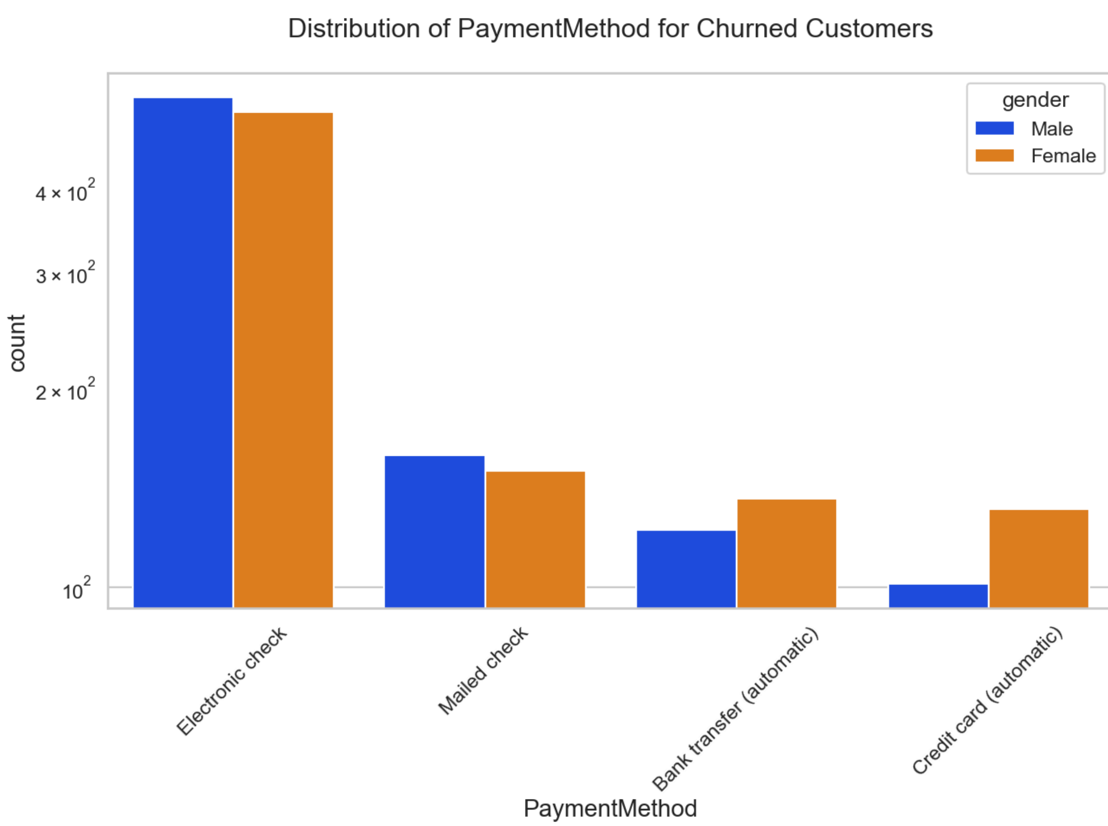
In [53]:

```
uniplot(new_data_target1, col = "Contract", title = "Distribution of conta
```

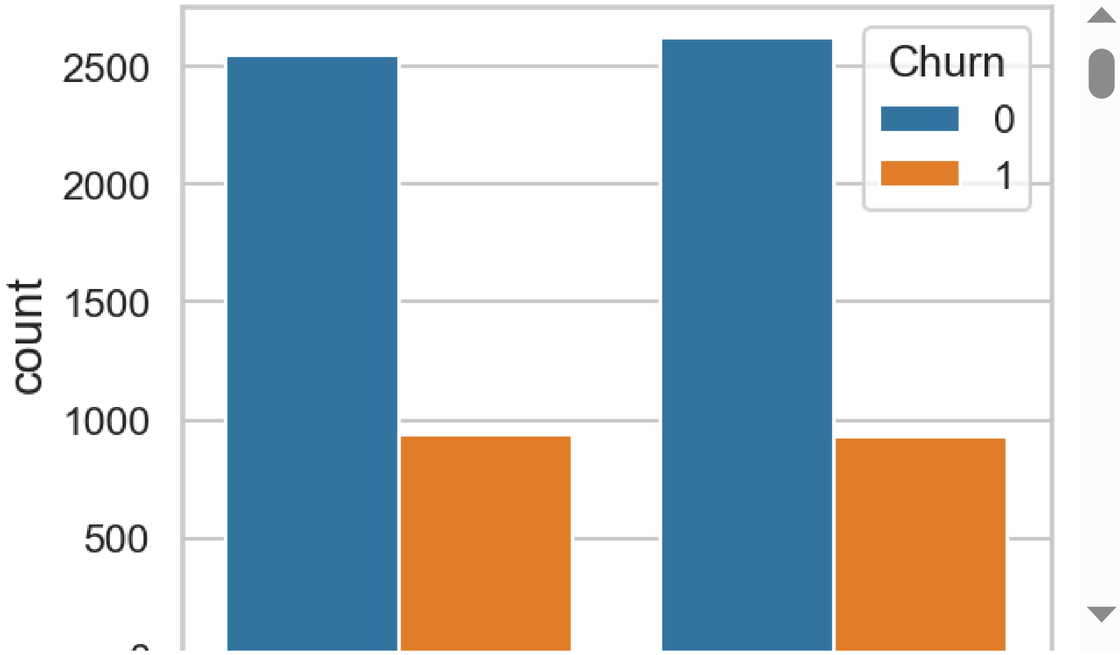


In [54]:

```
uniplot(new_data_target1, col = "PaymentMethod", title = "Distribution of
```



In [55]:
```
for i, predictor in enumerate(data.drop(columns = ["Churn", "TotalCharges"
    plt.figure(i)
    sns.countplot(data = data, x = predictor, hue = "Churn")
```



2. Convert the target variable "Churn" in a binary numerical variable i.e. yes = 1 and no = 0

In [57]:
```
data["Churn"] = np.where(data.Churn  == "Yes", 1, 0)
```

In [58]:
```
data.head()
```

Out[58]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
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

Conclusion These are the 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. Non senior citizens are high churners

In [65]:

In [ ]: