Exploratory Data Analysis

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

```
#import the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick  #for customizing ticks and labels
in plots
import matplotlib.pyplot as plt    #for plotting different charts
%matplotlib inline   #magic command: display plots in jupyter itself

#Load the data file
telco_base_data = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

#Look at the top 10 records of data
telco_base_data.head(10)
```

```
   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1
No
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
4  9237-HQITU  Female              0      No         No       2
Yes
5  9305-CDSKC  Female              0      No         No       8
Yes
6  1452-KIOVK    Male              0      No        Yes      22
Yes
7  6713-OKOMC  Female              0      No         No      10
No
8  7892-POOKP  Female              0     Yes         No      28
Yes
9  6388-TABGU    Male              0      No        Yes      62
Yes

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
```

```
Yes
2                No              DSL             Yes  ...
No
3  No phone service              DSL             Yes  ...
Yes
4                No     Fiber optic              No  ...
No
5               Yes     Fiber optic              No  ...
Yes
6               Yes     Fiber optic              No  ...
No
7  No phone service              DSL             Yes  ...
No
8               Yes     Fiber optic              No  ...
Yes
9                No              DSL             Yes  ...
No

  TechSupport StreamingTV StreamingMovies        Contract
PaperlessBilling  \
0          No          No              No  Month-to-month
Yes
1          No          No              No        One year
No
2          No          No              No  Month-to-month
Yes
3         Yes          No              No        One year
No
4          No          No              No  Month-to-month
Yes
5          No         Yes             Yes  Month-to-month
Yes
6          No         Yes              No  Month-to-month
Yes
7          No          No              No  Month-to-month
No
8         Yes         Yes             Yes  Month-to-month
Yes
9          No          No              No        One year
No

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes
5           Electronic check          99.65         820.5   Yes
6    Credit card (automatic)          89.10        1949.4    No
```

```
7               Mailed check          29.75         301.9    No
8           Electronic check         104.80       3046.05   Yes
9  Bank transfer (automatic)          56.15       3487.95    No

[10 rows x 21 columns]
```

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

```
(7043, 21)
```

```
# Columns
telco_base_data.columns.values
```

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

```
# Checking the data types of all the columns
telco_base_data.dtypes
```

```
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
```

```
telco_base_data.isnull().sum()
```

```
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
```

```
# Check the descriptive statistics of numeric variables
telco_base_data.describe().T
```

```
                  count       mean        std    min   25%    50%
75%  \
SeniorCitizen   7043.0   0.162147   0.368612   0.00   0.0   0.00
0.00
tenure          7043.0  32.371149  24.559481   0.00   9.0  29.00
55.00
MonthlyCharges  7043.0  64.761692  30.090047  18.25  35.5  70.35
89.85

                   max
SeniorCitizen     1.00
tenure           72.00
MonthlyCharges  118.75
```

```
# Analysis:
# Column: SeniorCitizen --> min,25%,50%,75% are zero only. So, let's
check it first.
telco_base_data['SeniorCitizen'].value_counts()
```

```
SeniorCitizen
0    5901
1    1142
Name: count, dtype: int64
```

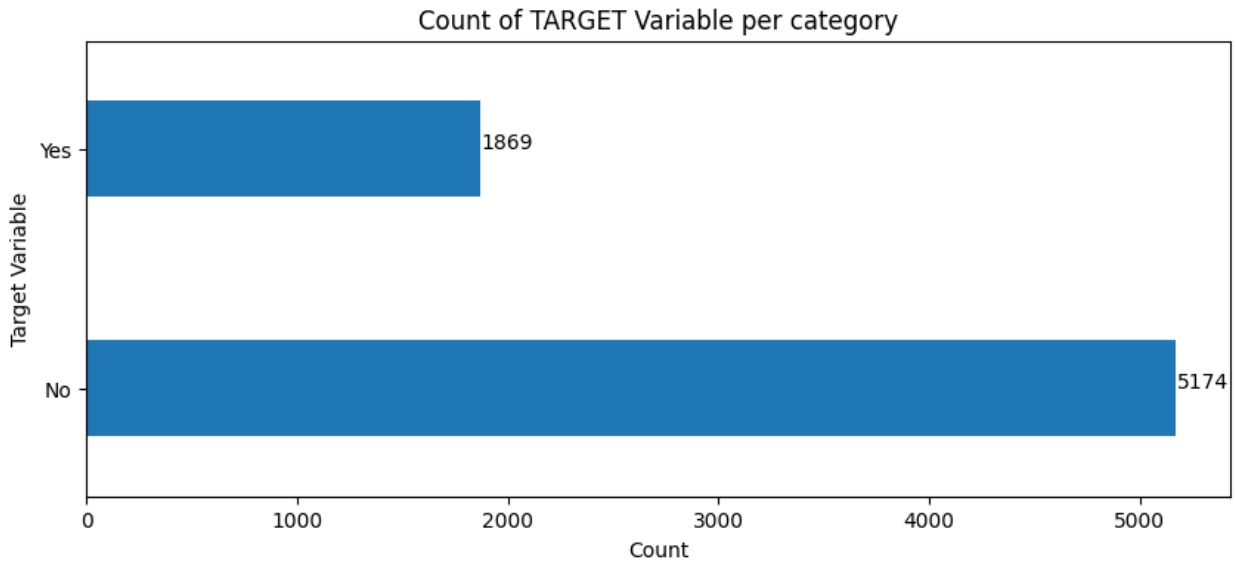
Insights:

1. SeniorCitizen is actually a categorical feature(0,1) hence the 25%-50%-75% distribution is not proper

2. 75% customers have tenure less than 55 months

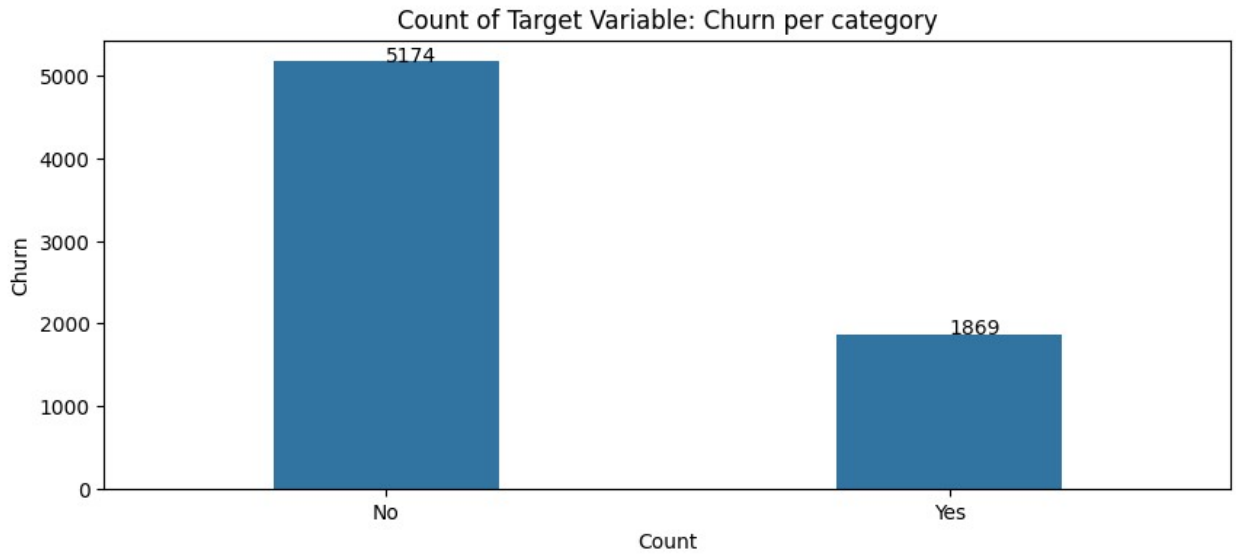
3. Average Monthly charges are USD 64.76 whereas 25% customers pay more than USD 89.85 per month.

```
# Plotting Churn frequency

telco_base_data['Churn'].value_counts().plot(kind='barh',
figsize=(10,4), width=0.4)
plt.xlabel("Count", )
plt.ylabel("Target Variable")
plt.title("Count of TARGET Variable per category")

for index, value in
enumerate(telco_base_data['Churn'].value_counts()):
    plt.text(value, index, str(value))  # `value` is count, `index` is
position on y-axis

plt.show()
```




```
# Using seaborn
# Create a simple bar plot
plt.figure(figsize=(10, 4))
sns.countplot(data=telco_base_data, x='Churn', width=0.4)

plt.xlabel("Count")
plt.ylabel("Churn")
```

```
plt.title("Count of Target Variable: Churn per category")

#for datalabels at the end of each bar
for index, value in
enumerate(telco_base_data['Churn'].value_counts()):
    plt.text(index, value, value)

plt.show()
```




```
# Calculating the percentage of churning
100*telco_base_data['Churn'].value_counts()/len(telco_base_data['Churn
'])
```

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

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

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

Note:

- 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
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
```

Missing Data - Initial Intuition

- We don't have any null values.

General Thumb Rules:

- For features with less missing values- we 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 **(but not always check for dependent features)**.

Data Cleaning

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

```
telco_data = telco_base_data.copy()
```

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

Note:

MonthlyCharges is float type but TotalCharges is object. So, correcting the datatype of TotalCharges to float.

```
telco_data.TotalCharges = pd.to_numeric(telco_data.TotalCharges,
errors='coerce')
# errors='coerce' used to handle errors that arise during the datatype
conversion process.

telco_data.isnull().sum()
```

```
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

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

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

```
6754  2775-SEFEE    Male              0      No        Yes       0

     PhoneService     MultipleLines InternetService
OnlineSecurity  ...  \
488            No  No phone service             DSL
Yes  ...
753           Yes                No              No  No internet
service  ...
936           Yes                No             DSL
Yes  ...
1082          Yes               Yes              No  No internet
service  ...
1340           No  No phone service             DSL
Yes  ...
3331          Yes                No              No  No internet
service  ...
3826          Yes               Yes              No  No internet
service  ...
4380          Yes                No              No  No internet
service  ...
5218          Yes                No              No  No internet
service  ...
6670          Yes               Yes             DSL
No  ...
6754          Yes               Yes             DSL
Yes  ...

         DeviceProtection          TechSupport          StreamingTV  \
488                   Yes                  Yes                  Yes
753   No internet service  No internet service  No internet service
936                   Yes                   No                  Yes
1082  No internet service  No internet service  No internet service
1340                  Yes                  Yes                  Yes
3331  No internet service  No internet service  No internet service
3826  No internet service  No internet service  No internet service
4380  No internet service  No internet service  No internet service
5218  No internet service  No internet service  No internet service
6670                  Yes                  Yes                  Yes
6754                   No                  Yes                   No

         StreamingMovies  Contract PaperlessBilling  \
488                   No  Two year              Yes
753   No internet service  Two year               No
936                   Yes  Two year               No
1082  No internet service  Two year               No
1340                   No  Two year               No
3331  No internet service  Two year               No
3826  No internet service  Two year               No
4380  No internet service  Two year               No
5218  No internet service  One year              Yes
```

```
6670                   No  Two year               No
6754                   No  Two year              Yes

                  PaymentMethod MonthlyCharges  TotalCharges  Churn
488   Bank transfer (automatic)          52.55           NaN     No
753                Mailed check          20.25           NaN     No
936                Mailed check          80.85           NaN     No
1082               Mailed check          25.75           NaN     No
1340    Credit card (automatic)          56.05           NaN     No
3331               Mailed check          19.85           NaN     No
3826               Mailed check          25.35           NaN     No
4380               Mailed check          20.00           NaN     No
5218               Mailed check          19.70           NaN     No
6670               Mailed check          73.35           NaN     No
6754  Bank transfer (automatic)          61.90           NaN     No

[11 rows x 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)
# how='any': In a row, any of its values is NaN, then drop the entire
row.
# how='all': In a row, if all the values are NaN, then only drop the
entire row.

#telco_data.fillna(0)
```

- Analysis on Tenure column

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...

```
# 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()
```

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

```
   gender  SeniorCitizen Partner Dependents PhoneService
MultipleLines  \
0  Female              0     Yes         No            No  No phone
service
1    Male              0      No         No           Yes
No
2    Male              0      No         No           Yes
No
3    Male              0      No         No            No  No phone
service
4  Female              0      No         No           Yes
No

  InternetService OnlineSecurity OnlineBackup DeviceProtection
TechSupport  \
0             DSL             No          Yes               No
No
1             DSL            Yes           No              Yes
No
2             DSL            Yes          Yes               No
No
3             DSL            Yes           No              Yes
Yes
4     Fiber optic             No           No               No
No

  StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No              No  Month-to-month              Yes
1          No              No        One year               No
2          No              No  Month-to-month              Yes
3          No              No        One year               No
4          No              No  Month-to-month              Yes
```

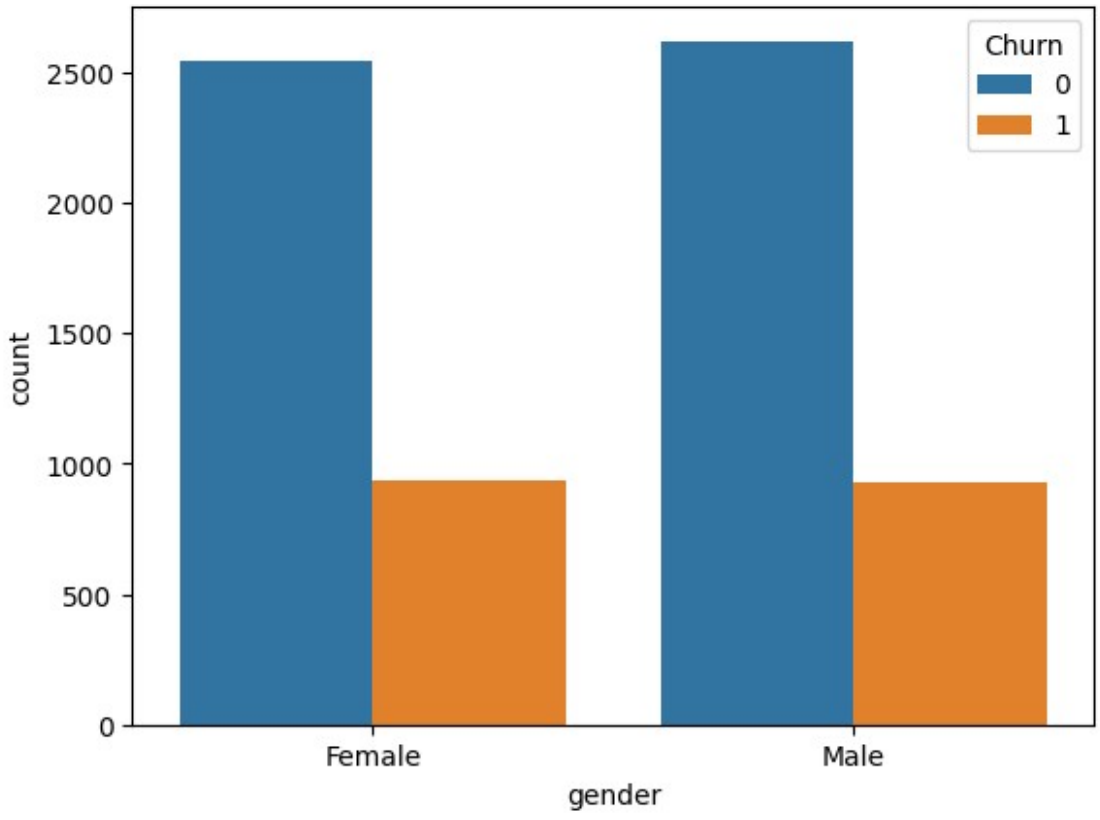
```
                PaymentMethod  MonthlyCharges  TotalCharges Churn
tenure_group
0            Electronic check           29.85         29.85    No
1 - 12
1                Mailed check           56.95       1889.50    No
25 - 36
2                Mailed check           53.85        108.15   Yes
1 - 12
3  Bank transfer (automatic)           42.30       1840.75    No
37 - 48
4            Electronic check           70.70        151.65   Yes
1 - 12
```

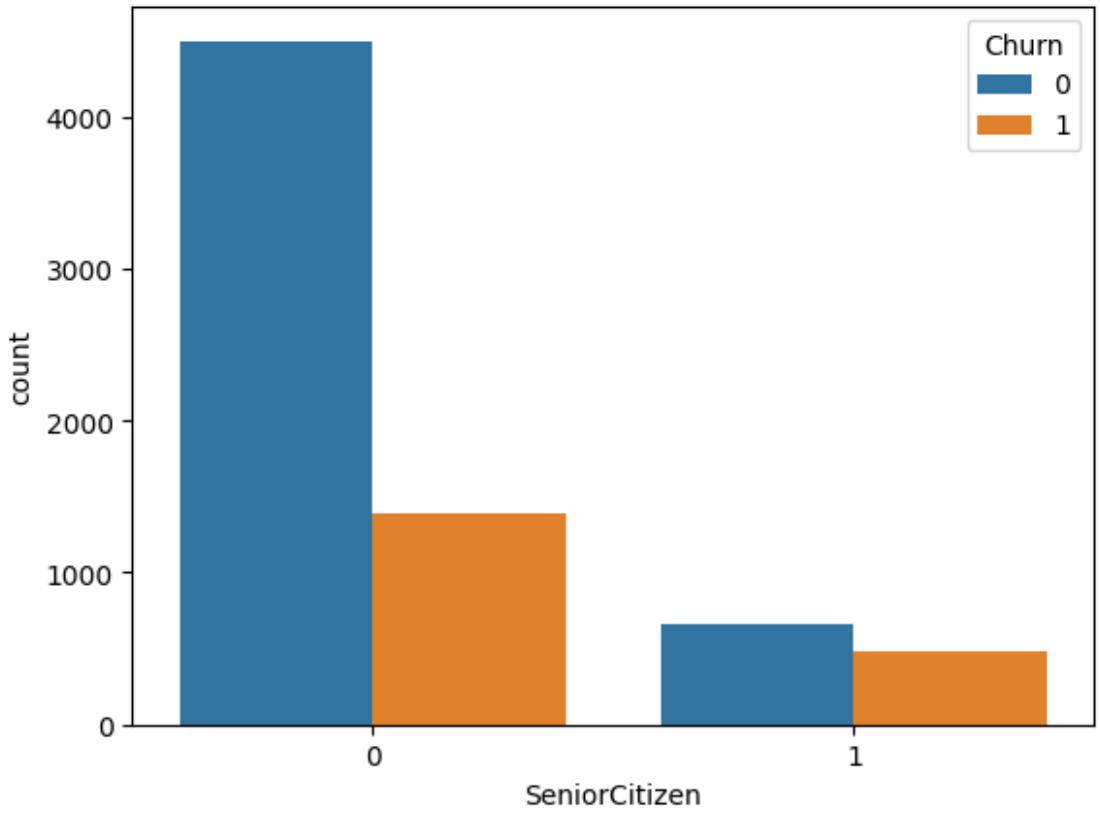
Data Exploration

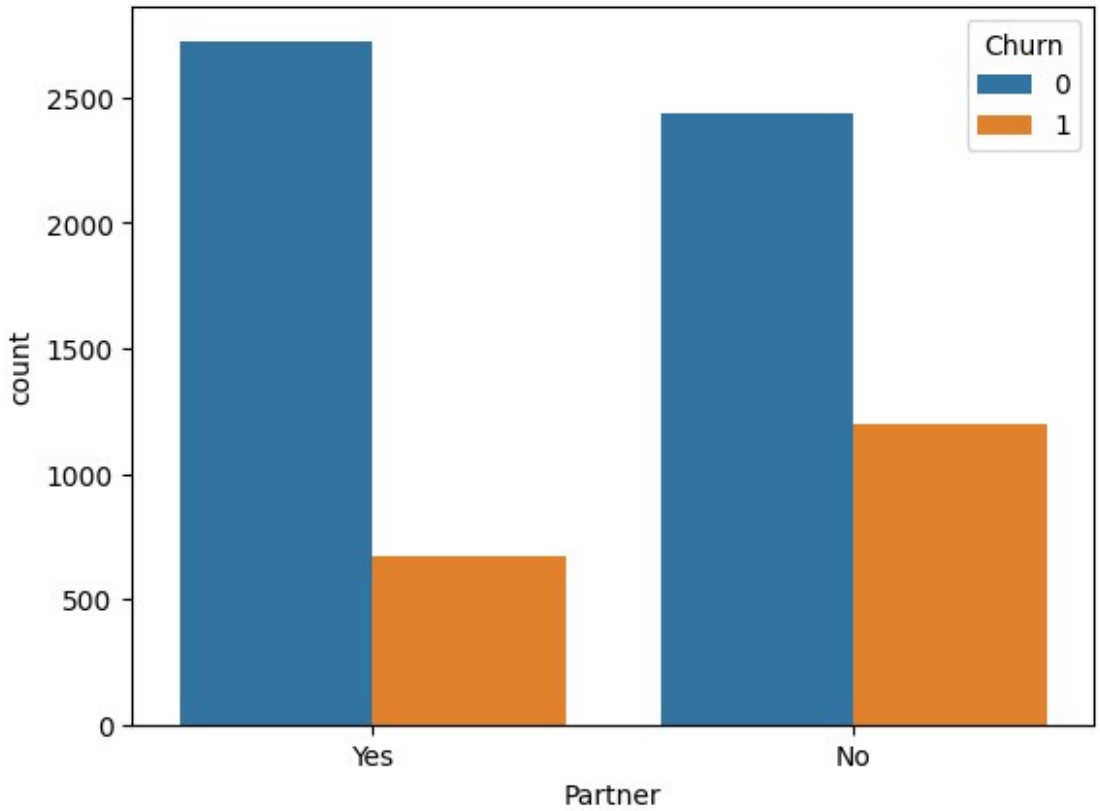
1. Plot distibution of individual predictors by churn

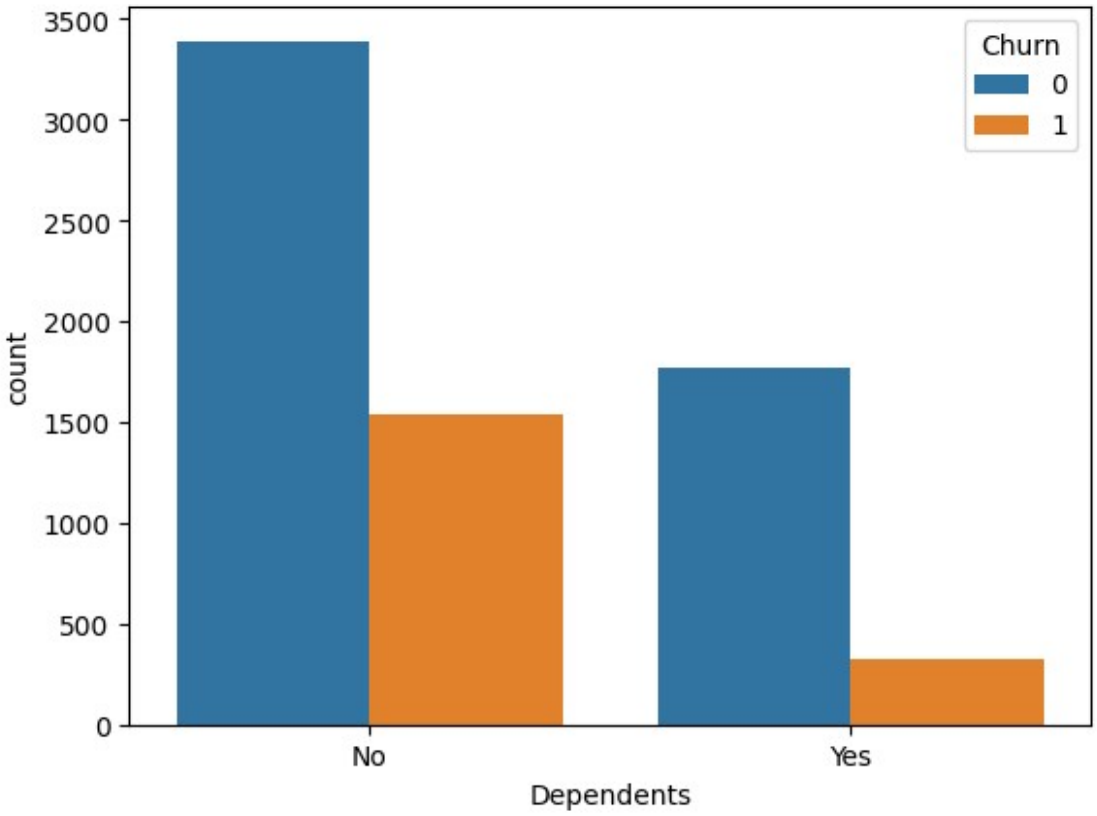
Univariate Analysis

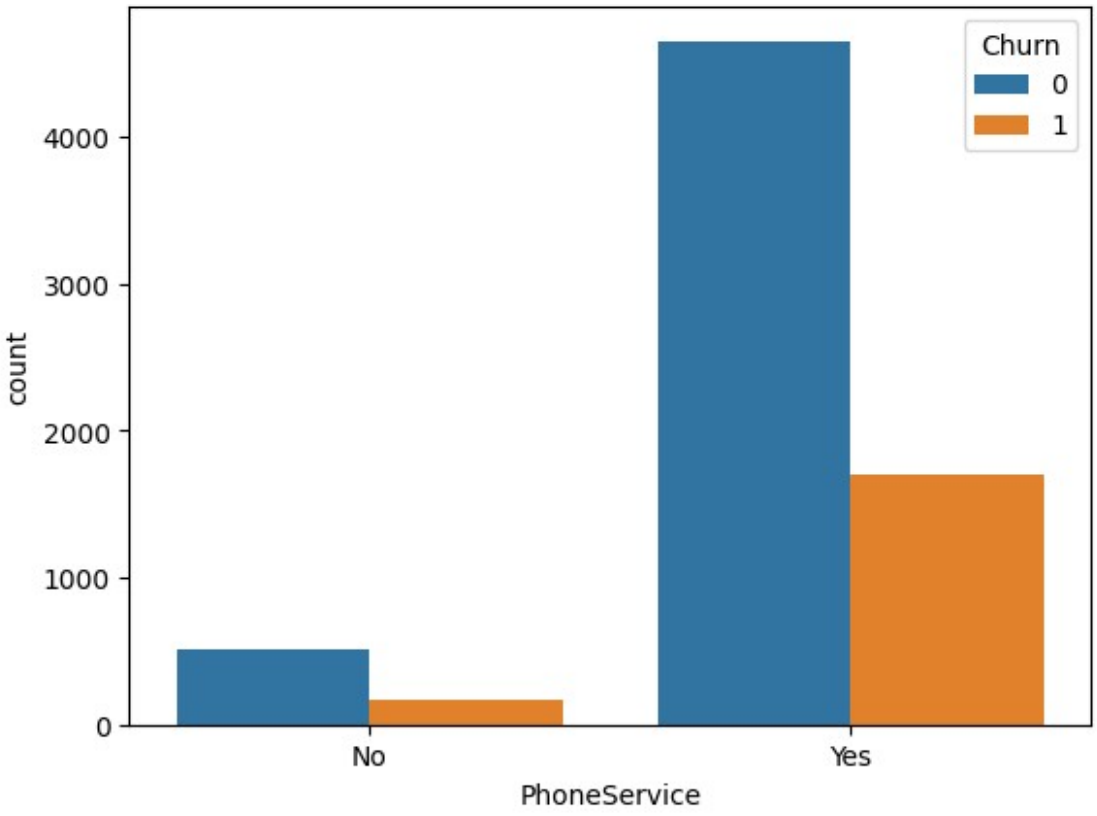
```
for i, predictor in enumerate(telco_data.drop(columns=['Churn',
'TotalCharges', 'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=telco_data, x=predictor, hue='Churn')
```

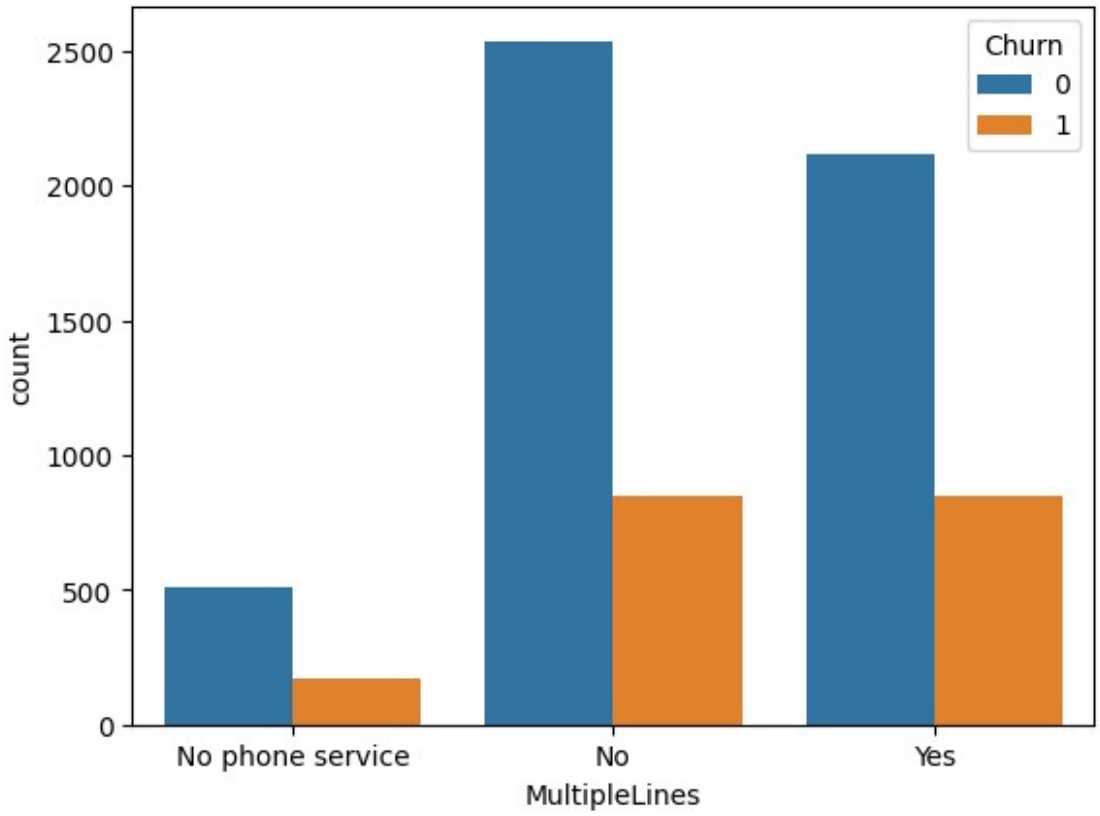


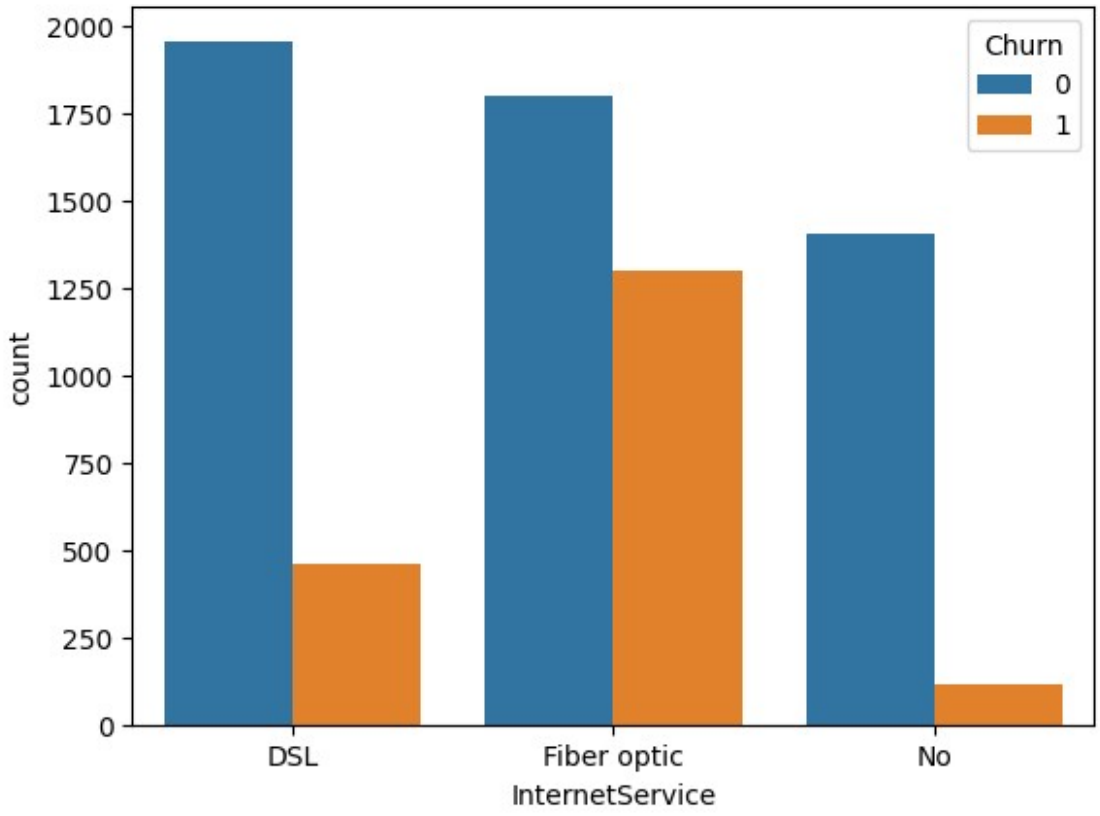

Churn
0
1
4000
3000
2000
1000
0
count
0
1
SeniorCitizen

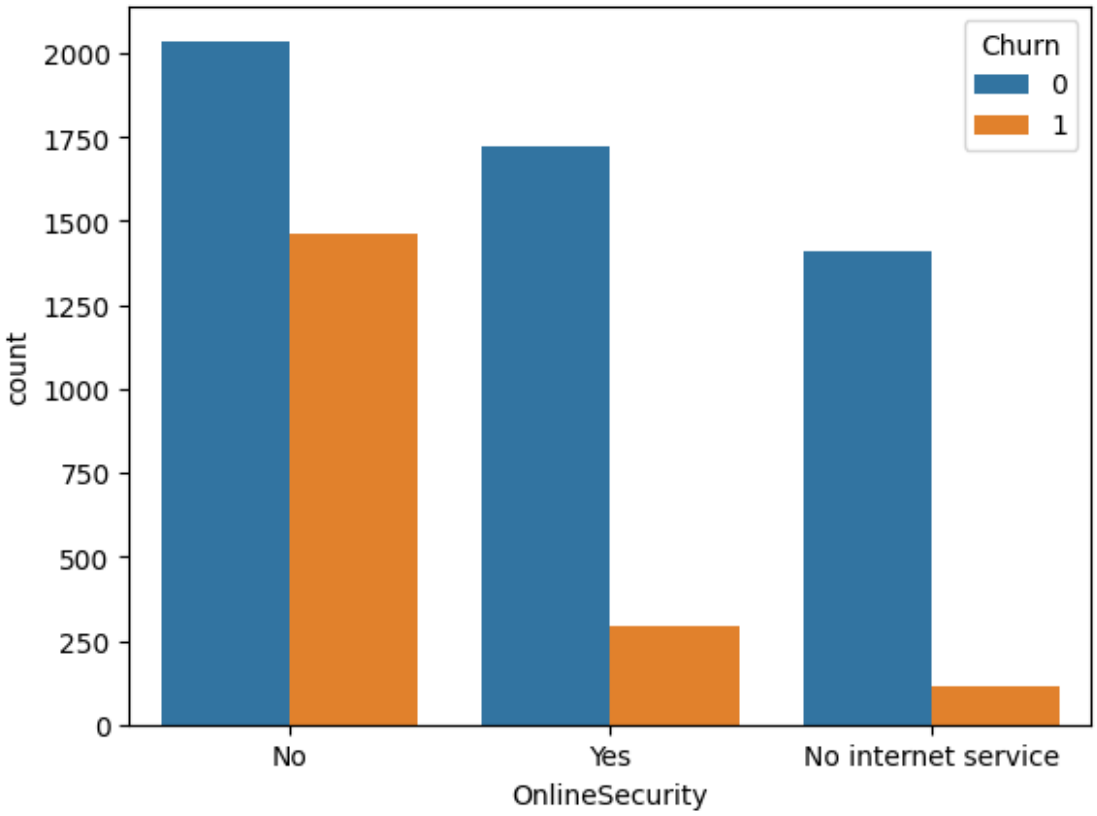

Churn
0
1
2500
2000
1500
1000
500
0
count
Yes
No
Partner

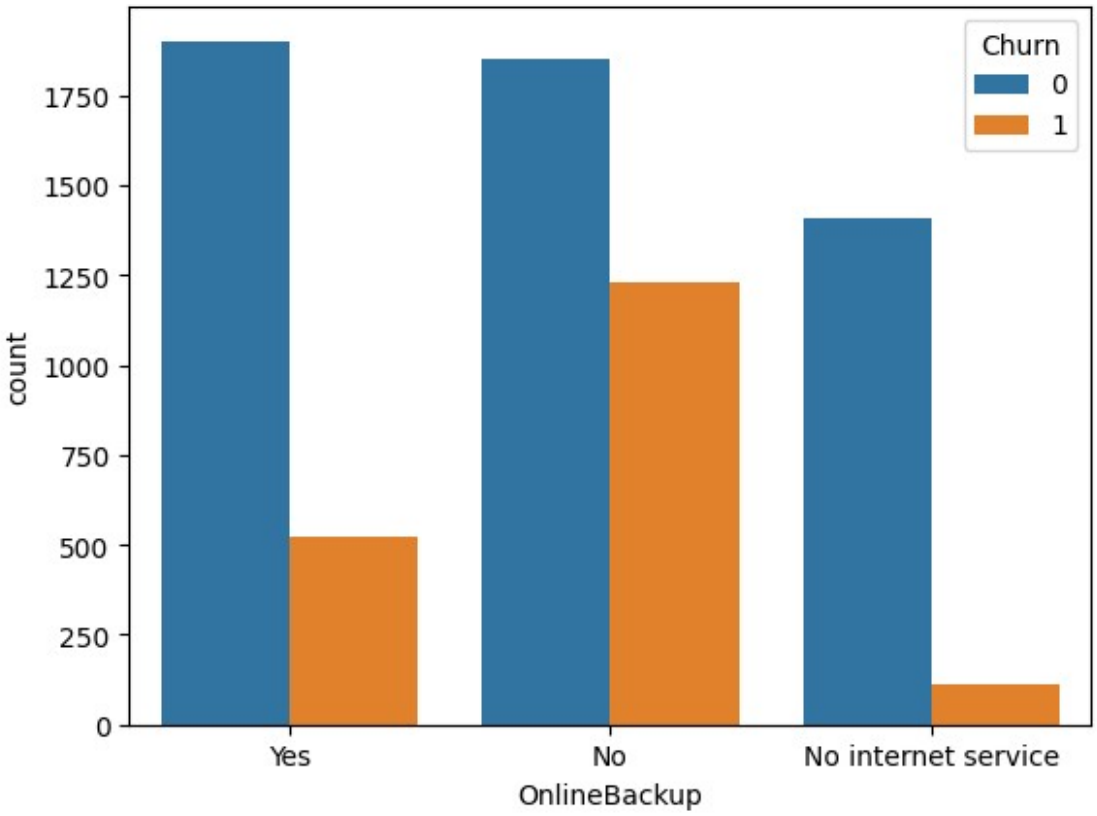

Churn
0
1
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents

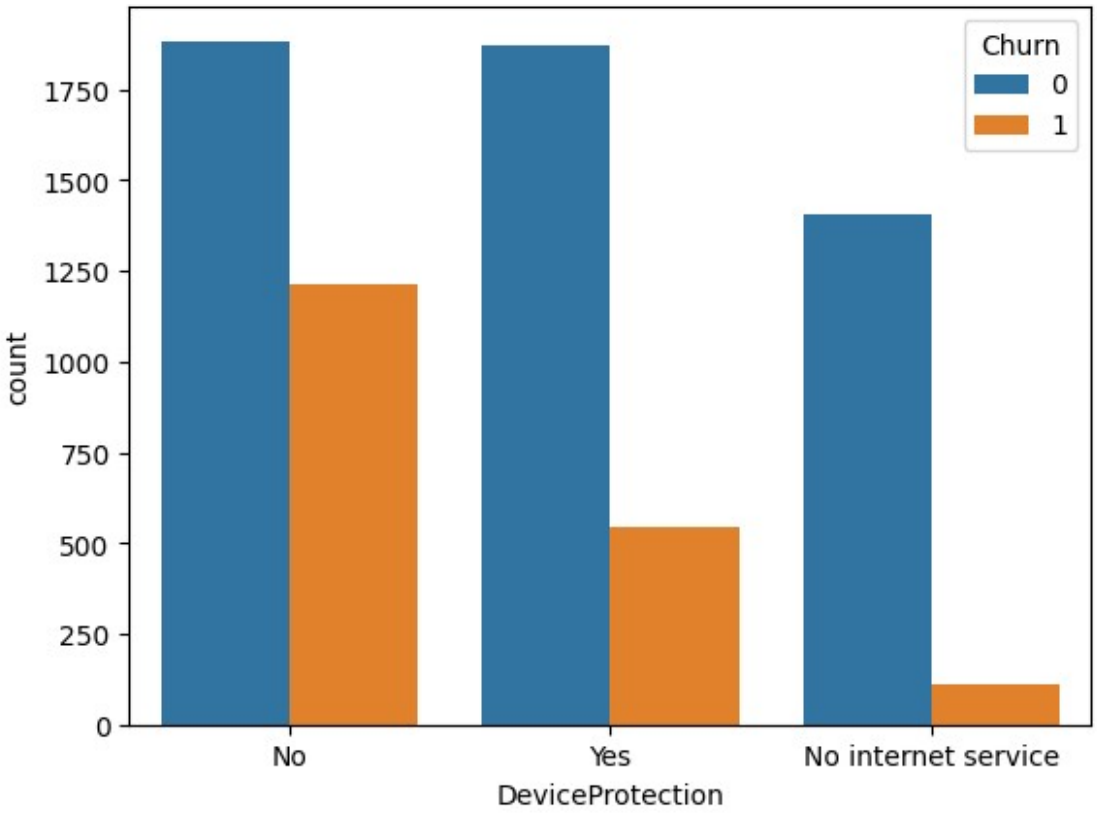

Churn
0
1
4000
3000
2000
1000
0
count
No
Yes
PhoneService

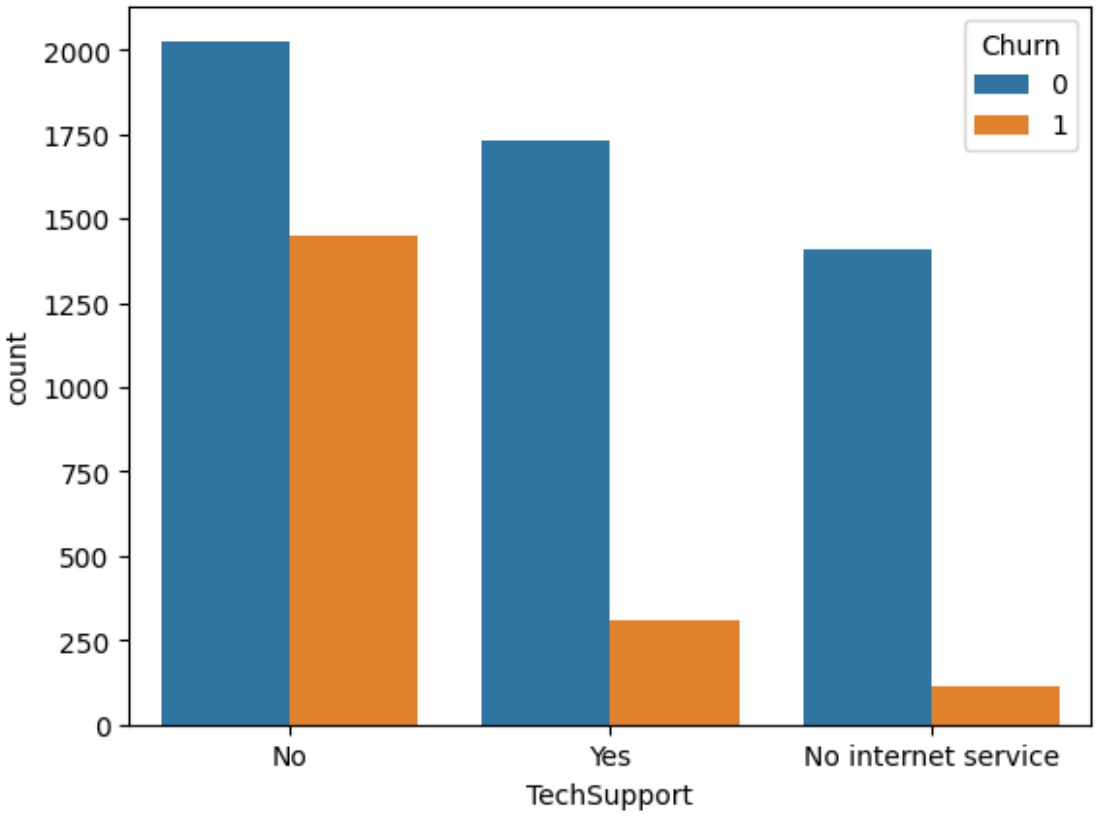

Churn
0
1
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

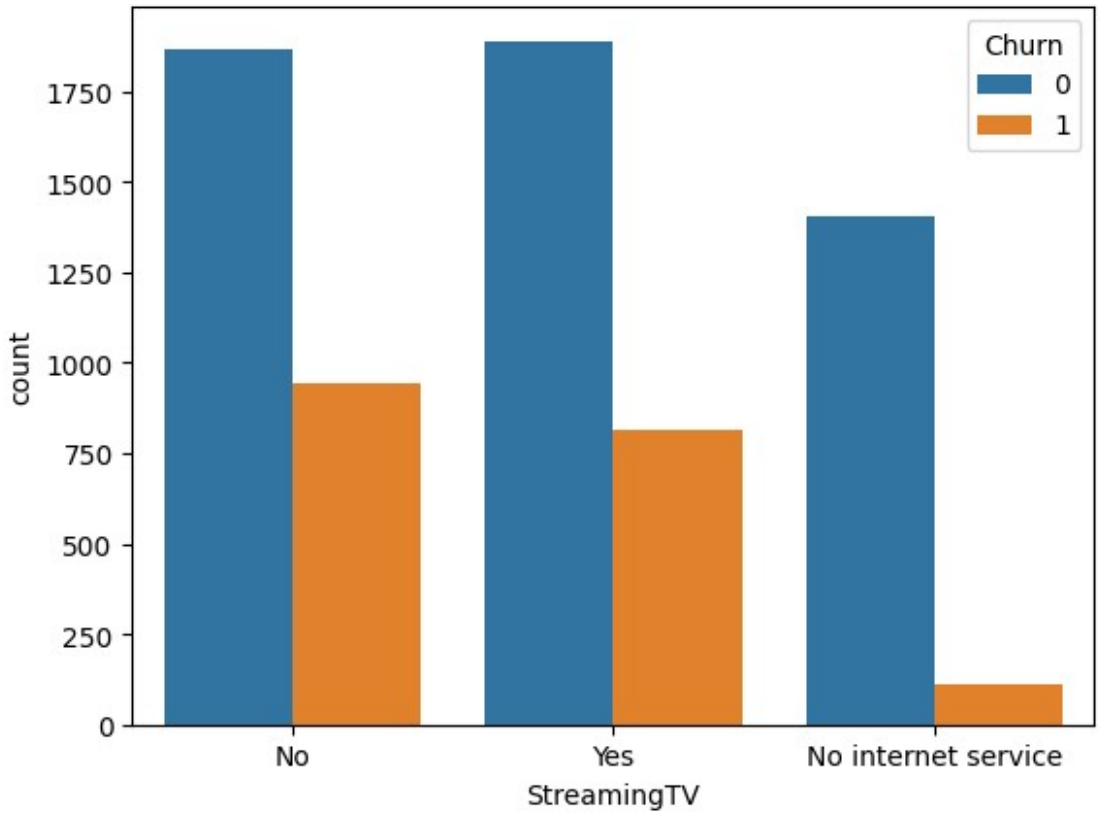

Churn
0
1
2000
1750
1500
1250
1000
750
500
250
0
count
DSL
Fiber optic
No
InternetService

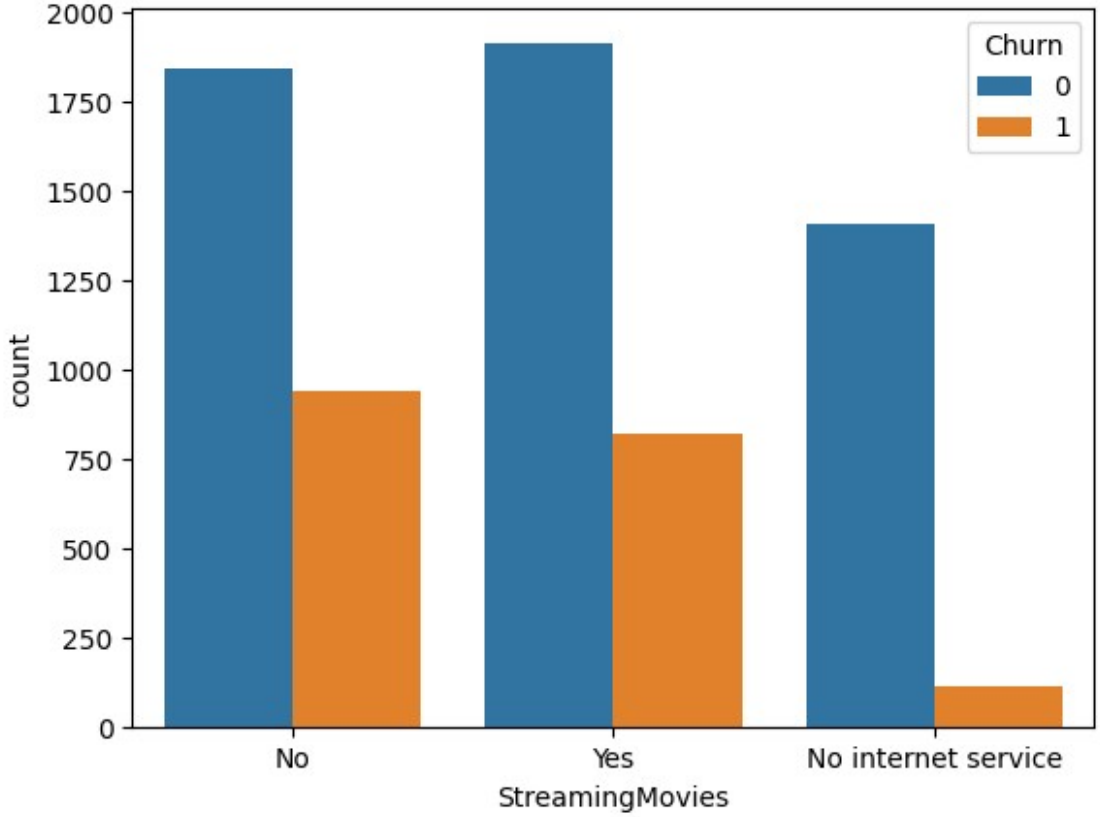

Churn
0
1
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
OnlineSecurity

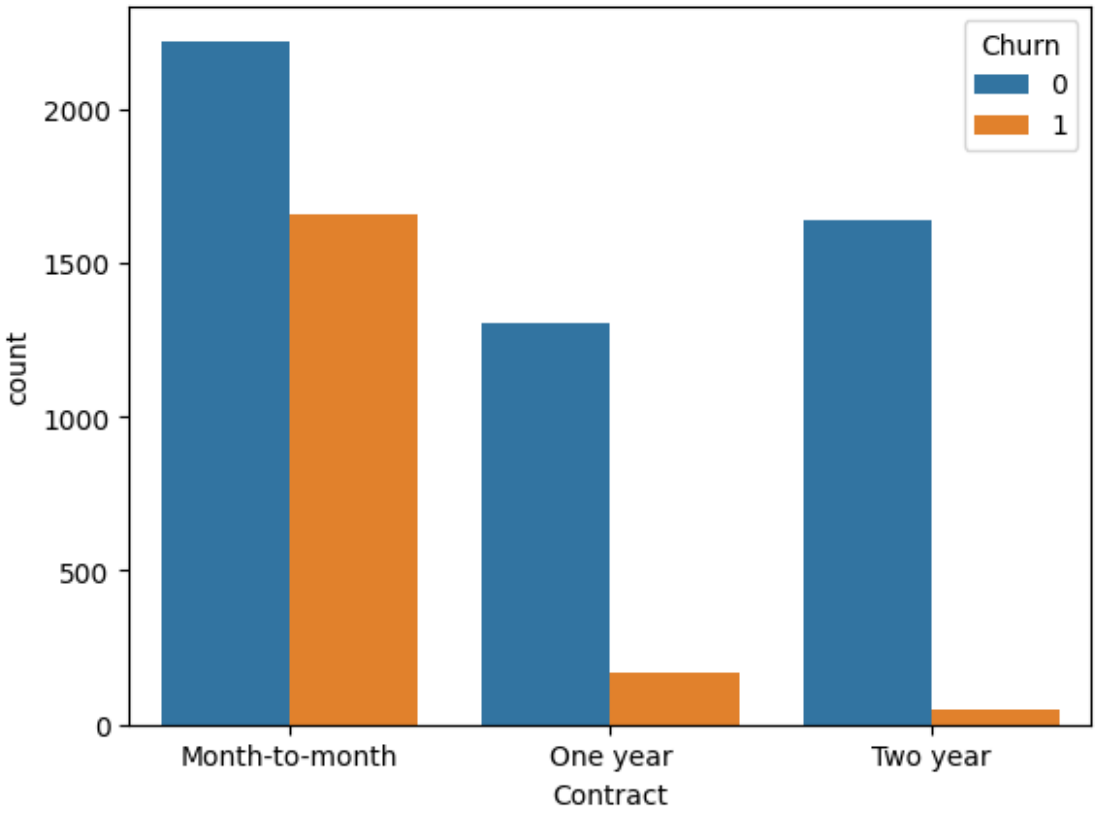

Churn
0
1
1750
1500
1250
1000
750
500
250
0
count
Yes
No
No internet service
OnlineBackup

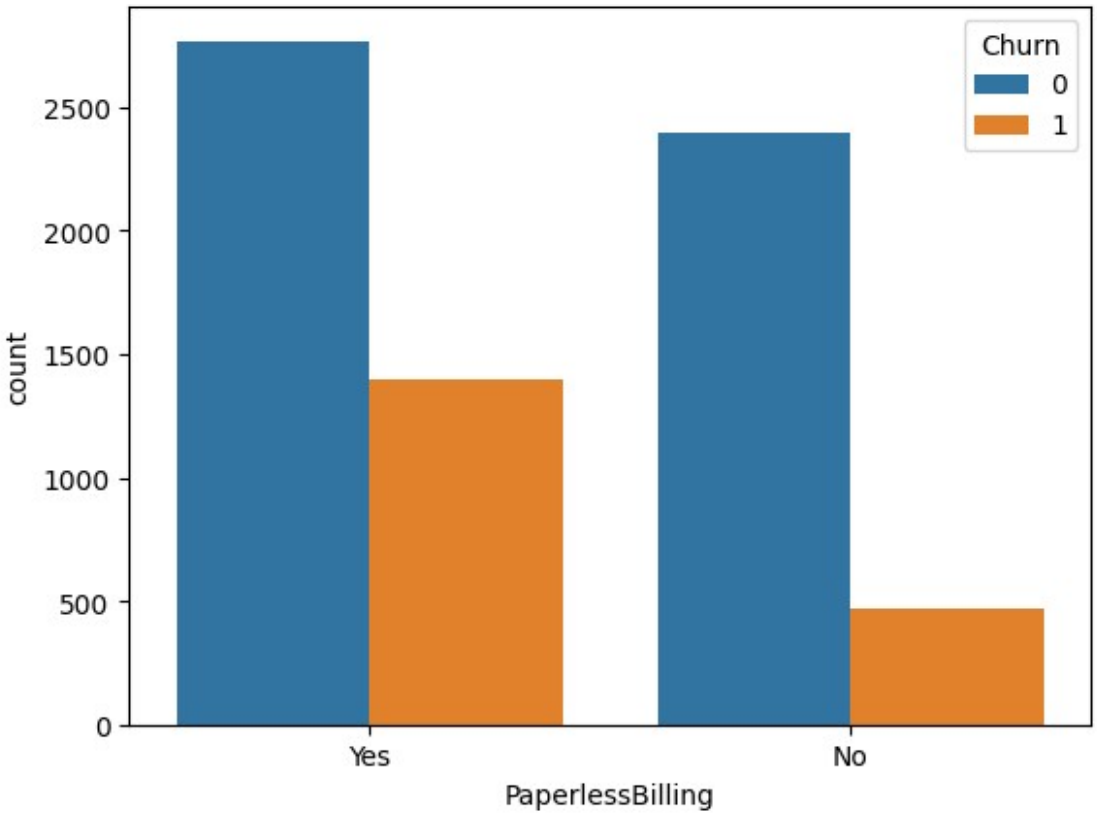

Churn
0
1
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
DeviceProtection

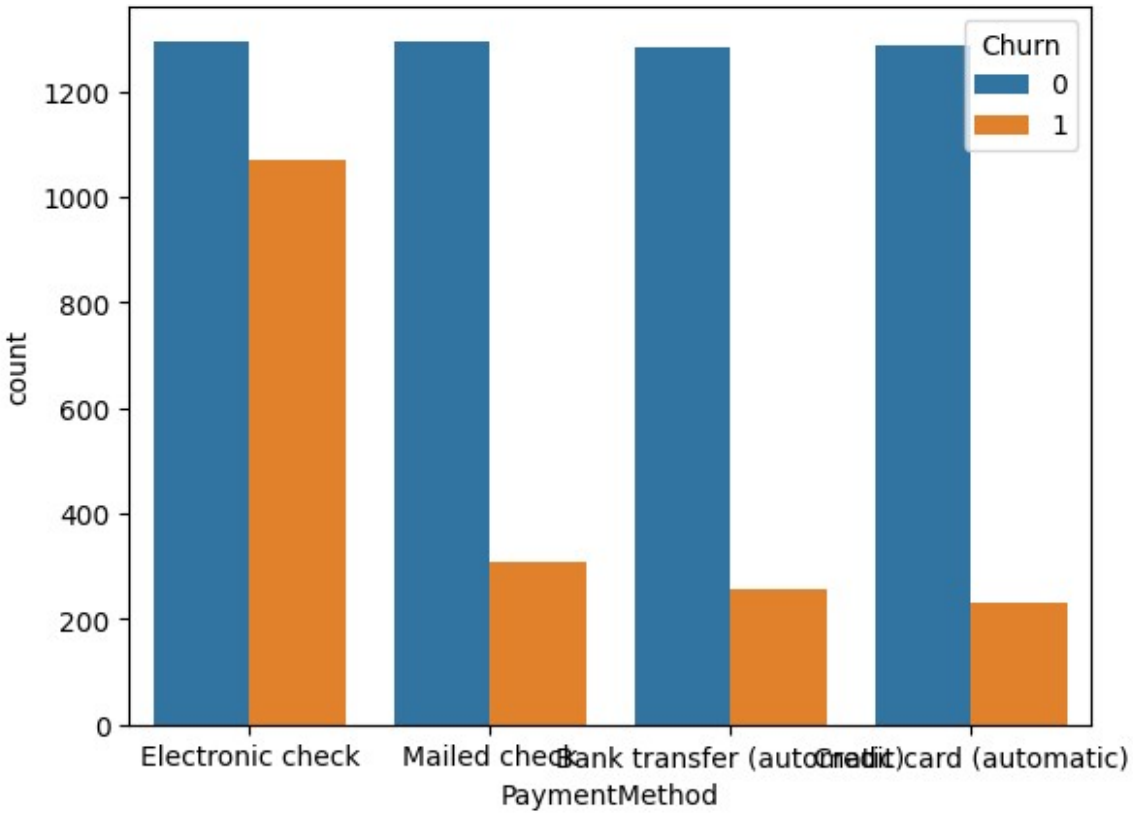

Churn
0
1
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
TechSupport

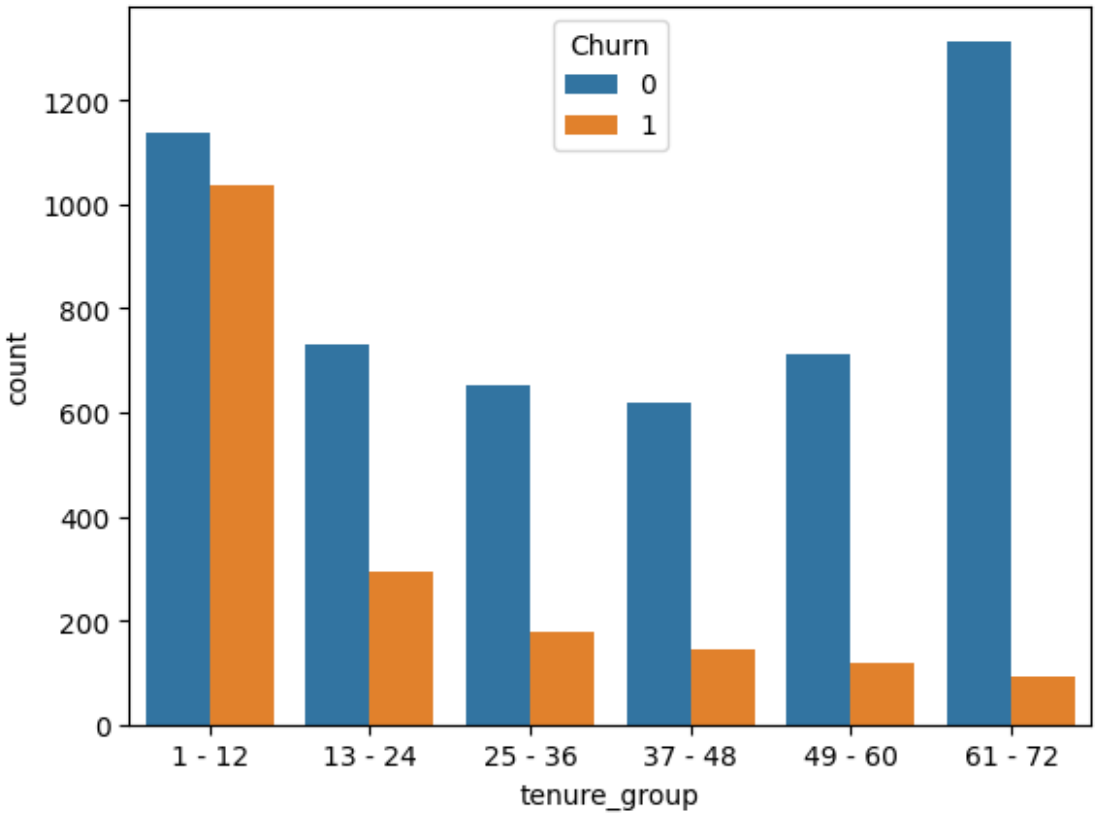

Churn
0
1
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingTV


Churn
0
1
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingMovies


Churn
0
1
2000
1500
1000
500
0
count
Month-to-month
One year
Two year
Contract


Churn
0
1
2500
2000
1500
1000
500
0
count
Yes
No
PaperlessBilling


Churn
0
1
count
1200
1000
800
600
400
200
0
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod


Churn
0
1
count
1200
1000
800
600
400
200
0
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'] = telco_data['Churn'].replace({'Yes': 1, 'No': 0})
```

```
C:\Users\Sanjli Kumari\AppData\Local\Temp\
ipykernel_3588\2627463442.py:1: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To
retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  telco_data['Churn'] = telco_data['Churn'].replace({'Yes': 1, 'No':
0})
```

```
telco_data.dtypes
```

```
gender                object
SeniorCitizen          int64
Partner               object
Dependents            object
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         float64
Churn                  int64
tenure_group        category
dtype: object
```

```
# Saving a copy safer side because the one-hot encoder might get wrong
telco_data1 = telco_data1.copy()
```

3. Convert all the categorical variables into dummy variables

```
telco_data.head()
```

```
   gender  SeniorCitizen Partner Dependents PhoneService
MultipleLines  \
0  Female              0     Yes         No           No  No phone
service
1    Male              0      No         No          Yes
No
```

```
2    Male              0      No         No           Yes
No
3    Male              0      No         No            No  No phone
service
4  Female              0      No         No           Yes
No

  InternetService OnlineSecurity OnlineBackup DeviceProtection
TechSupport  \
0             DSL             No          Yes               No
No
1             DSL            Yes           No              Yes
No
2             DSL            Yes          Yes               No
No
3             DSL            Yes           No              Yes
Yes
4     Fiber optic             No           No               No
No

  StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No              No  Month-to-month              Yes
1          No              No        One year               No
2          No              No  Month-to-month              Yes
3          No              No        One year               No
4          No              No  Month-to-month              Yes

                PaymentMethod  MonthlyCharges  TotalCharges  Churn
tenure_group
0           Electronic check           29.85         29.85      0
1 - 12
1               Mailed check           56.95       1889.50      0
25 - 36
2               Mailed check           53.85        108.15      1
1 - 12
3  Bank transfer (automatic)           42.30       1840.75      0
37 - 48
4           Electronic check           70.70        151.65      1
1 - 12

telco_data = telco_data1.copy()

telco_data.dtypes

gender                object
SeniorCitizen          int64
Partner               object
Dependents            object
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         float64
Churn                  int64
tenure_group        category
dtype: object
```

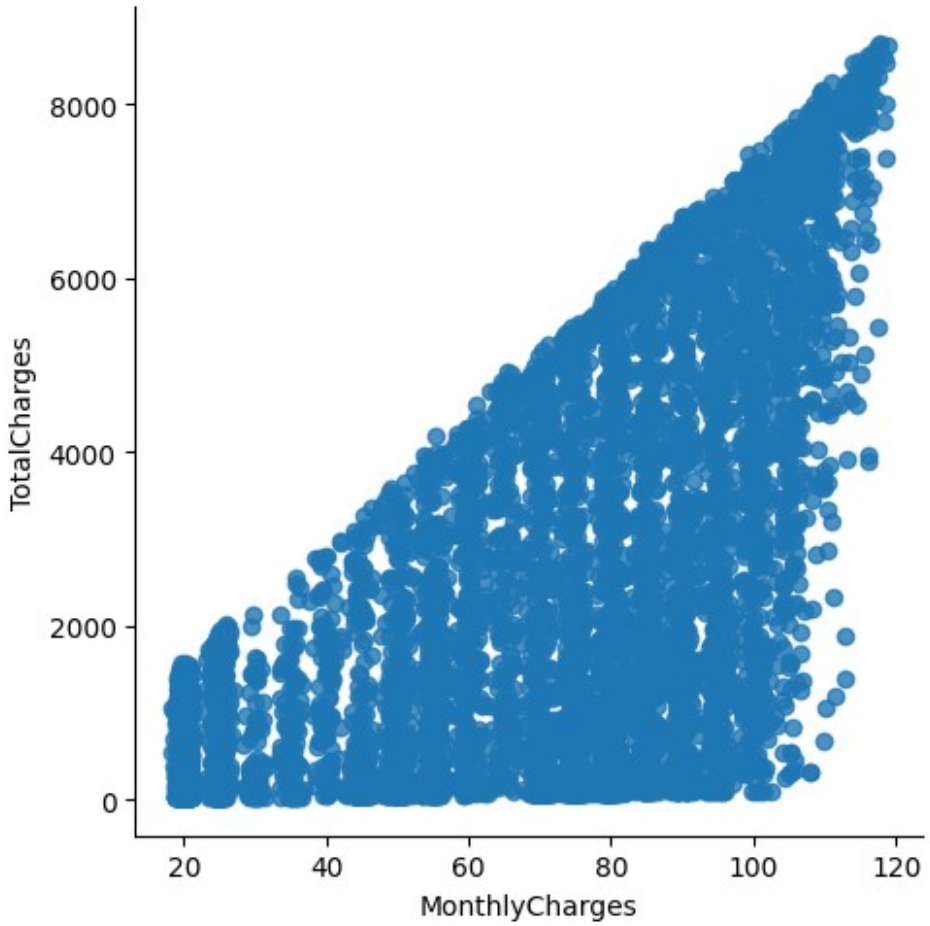
```
telco_data_dummies = pd.get_dummies(telco_data, drop_first=True)
telco_data_dummies.head()
```

```
   SeniorCitizen  MonthlyCharges  TotalCharges  Churn  gender_Male  \
0              0           29.85         29.85      0        False
1              0           56.95       1889.50      0         True
2              0           53.85        108.15      1         True
3              0           42.30       1840.75      0         True
4              0           70.70        151.65      1        False

   Partner_Yes  Dependents_Yes  PhoneService_Yes  \
0         True           False             False
1        False           False              True
2        False           False              True
3        False           False             False
4        False           False              True

   MultipleLines_No phone service  MultipleLines_Yes  ...
Contract_Two year  \
0                            True              False  ...
False
1                           False              False  ...
False
2                           False              False  ...
False
3                            True              False  ...
False
4                           False              False  ...
False

   PaperlessBilling_Yes  PaymentMethod_Credit card (automatic)  \
0                  True                                  False
1                 False                                  False
2                  True                                  False
```

```
3                  False                                  False
4                   True                                  False

   PaymentMethod_Electronic check  PaymentMethod_Mailed check  \
0                           True                       False
1                          False                        True
2                          False                        True
3                          False                       False
4                           True                       False

   tenure_group_13 - 24  tenure_group_25 - 36  tenure_group_37 - 48  \
0                 False                 False                 False
1                 False                  True                 False
2                 False                 False                 False
3                 False                 False                  True
4                 False                 False                 False

   tenure_group_49 - 60  tenure_group_61 - 72
0                 False                 False
1                 False                 False
2                 False                 False
3                 False                 False
4                 False                 False

[5 rows x 35 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 0x2e660cb5010>
```

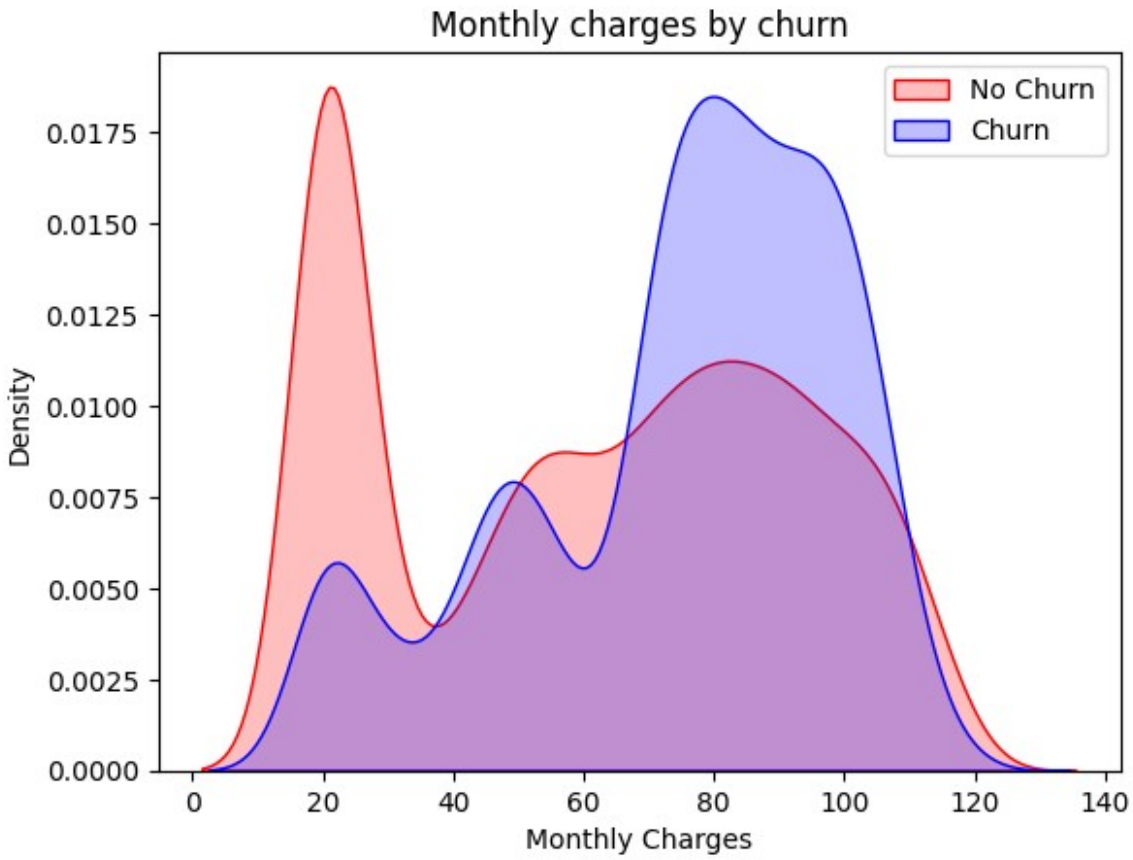


Total Charges increase as Monthly Charges increase - as expected.

10. Churn by Monthly Charges and Total Charges

```
Mth =
sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Chu
rn"] == 0) ],
                color="Red", shade = True)
Mth =
sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Chu
rn"] == 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')

C:\Users\Sanjli Kumari\AppData\Local\Temp\
ipykernel_3588\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\Sanjli Kumari\AppData\Local\Temp\
ipykernel_3588\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) ],

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

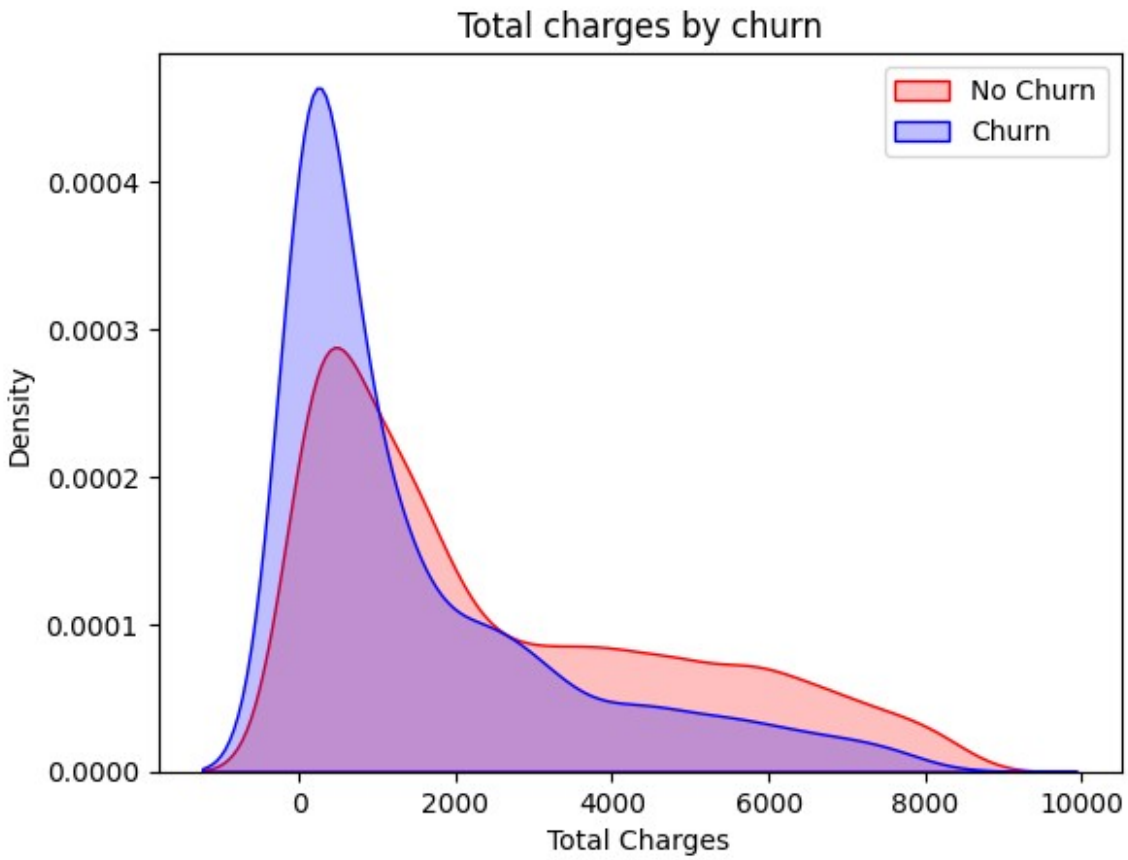


Insight: Churn is high when Monthly Charges ar high

```
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')
```

```
C:\Users\Sanji Kumari\AppData\Local\Temp\
ipykernel_3588\4019118049.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.

  Tot =
sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn
"] == 0) ],
C:\Users\Sanji Kumari\AppData\Local\Temp\
ipykernel_3588\4019118049.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.

  Tot =
sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn
"] == 1) ],

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



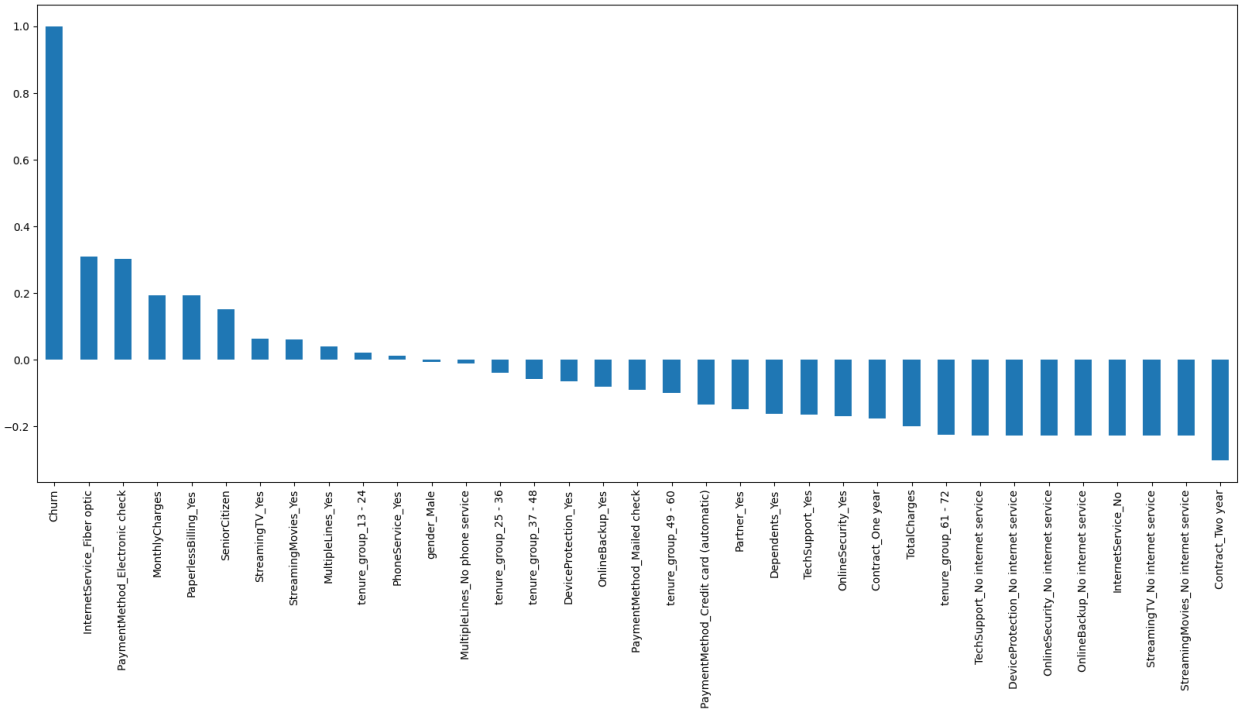
Surprising insight as higher Churn at lower Total Charges

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'

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

```
<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**

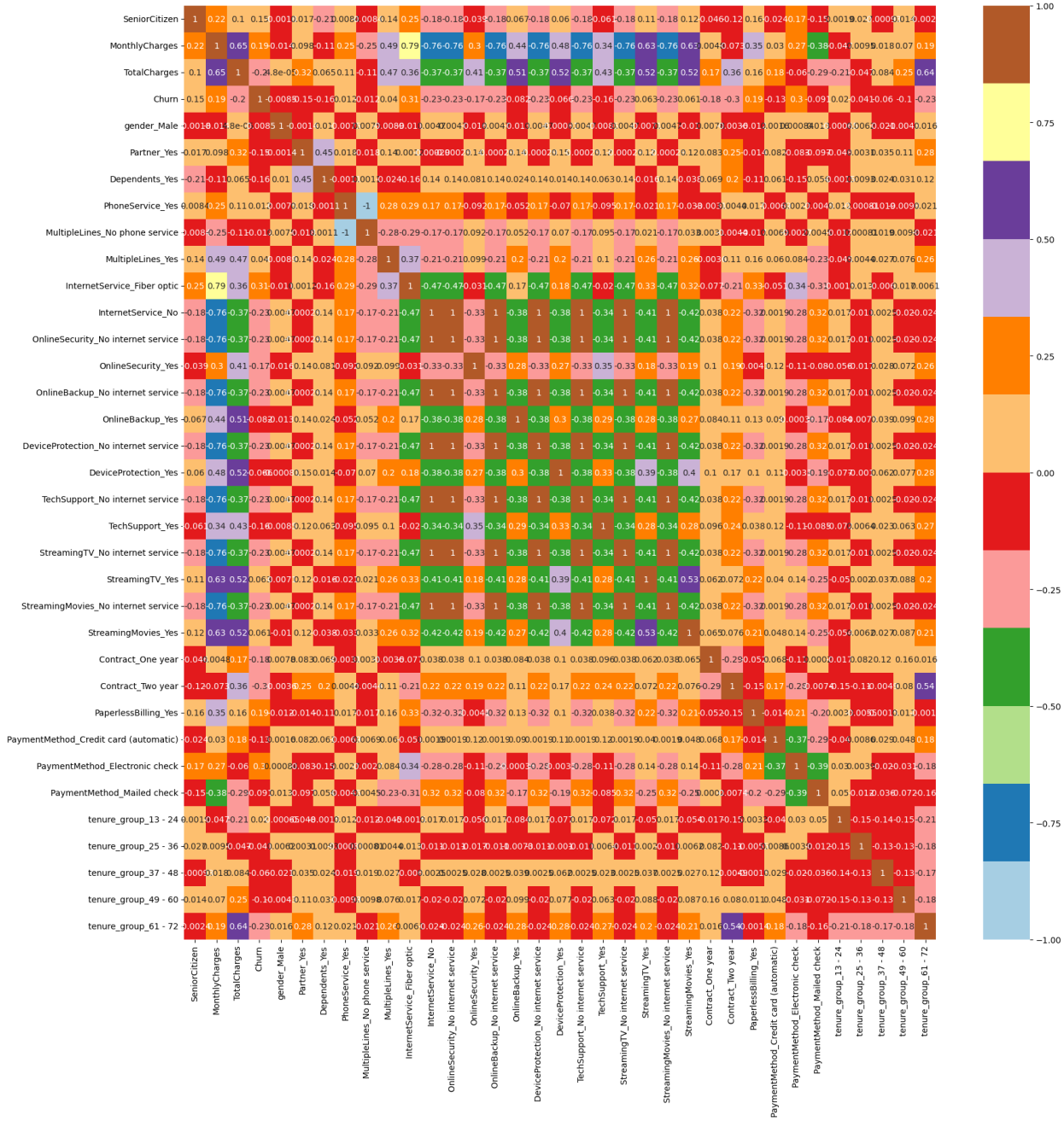
LOW Churn is seens in case of **Long term contracts**, **Subscriptions without internet service** and **The customers engaged for 5+ years**

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

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

```
plt.figure(figsize=(20,20))
sns.heatmap(telco_data_dummies.corr(), cmap="Paired", annot=True)
```

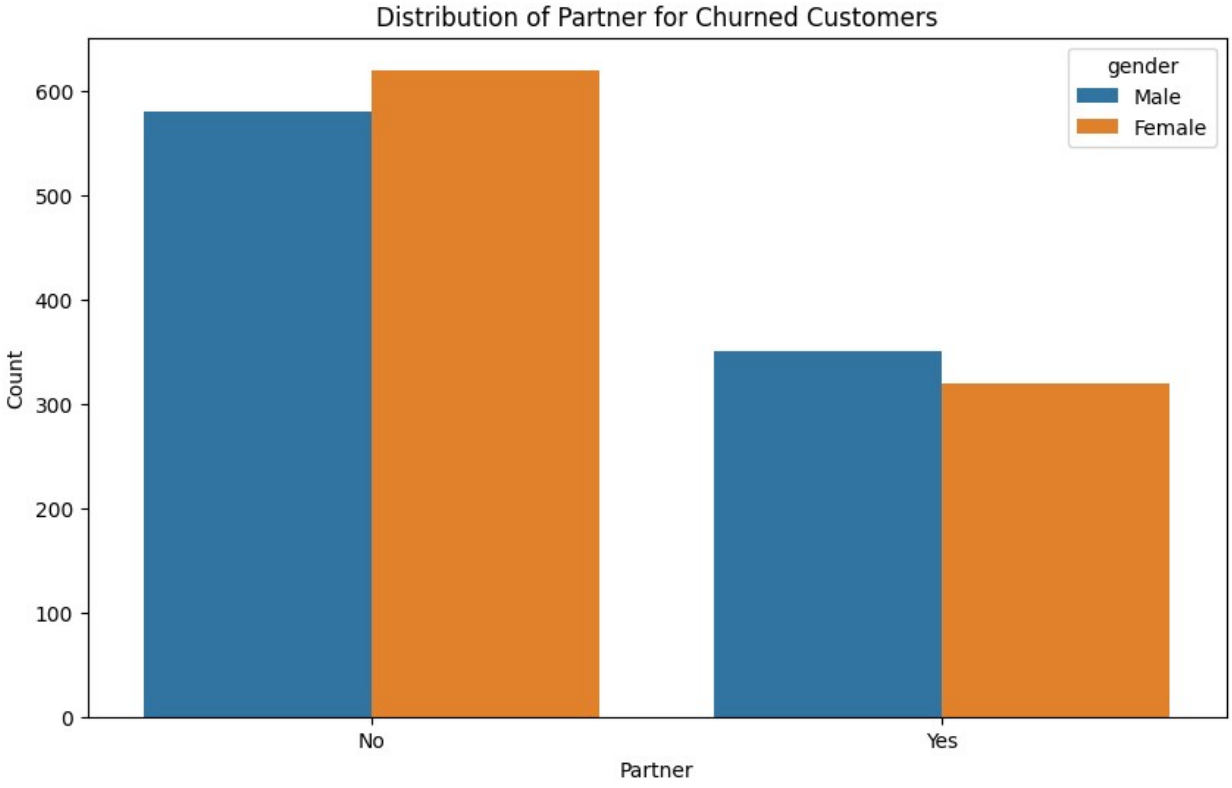
```
<Axes: >
```

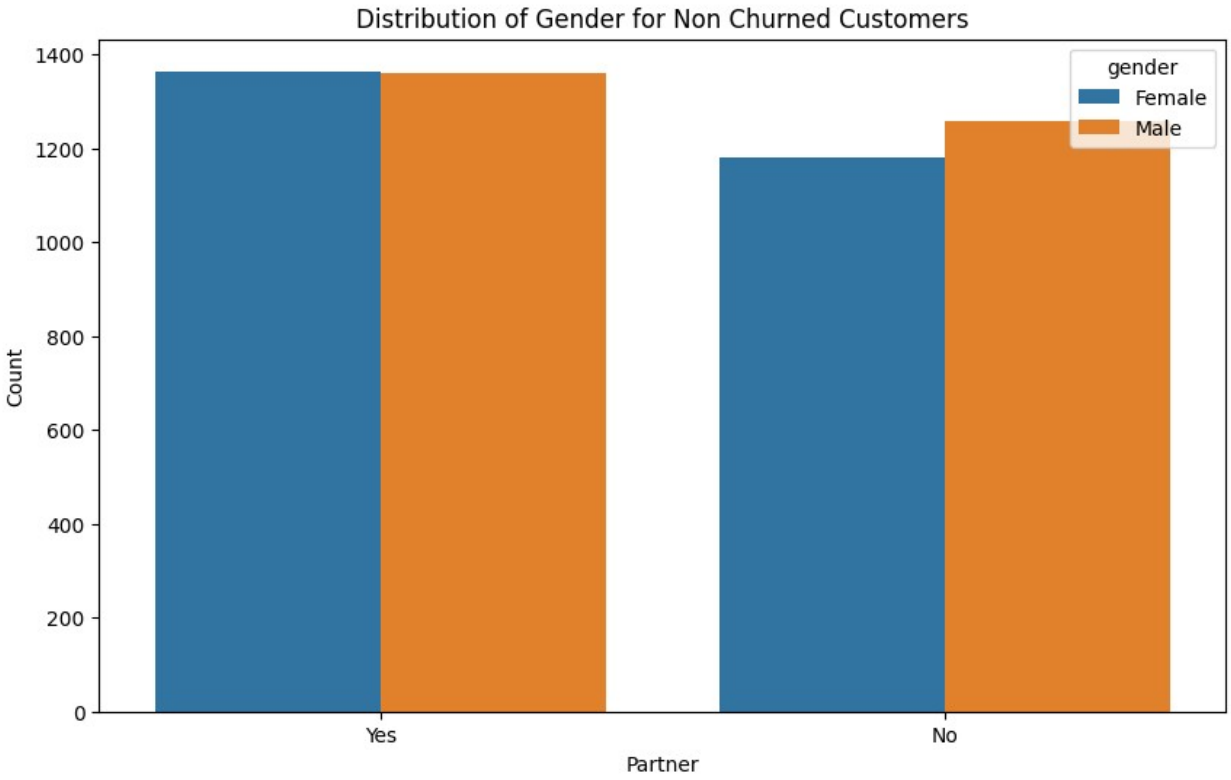


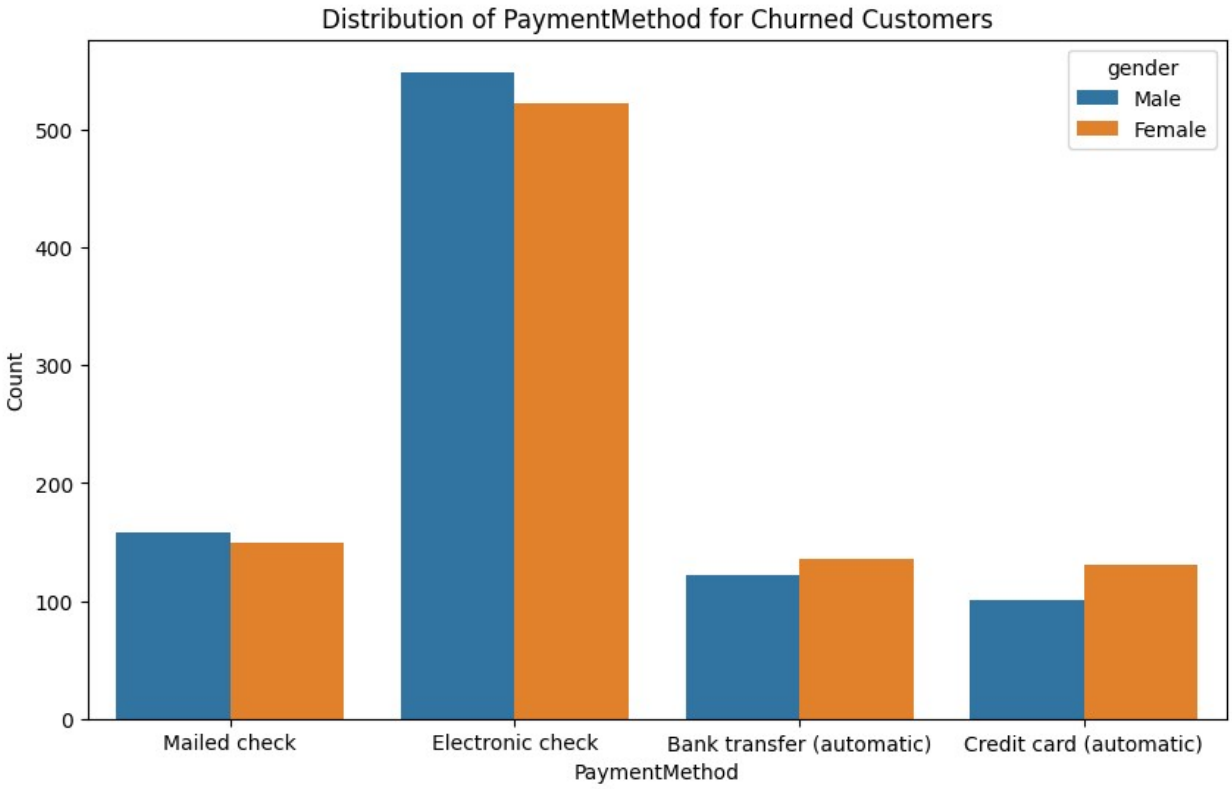
Bivariate Analysis

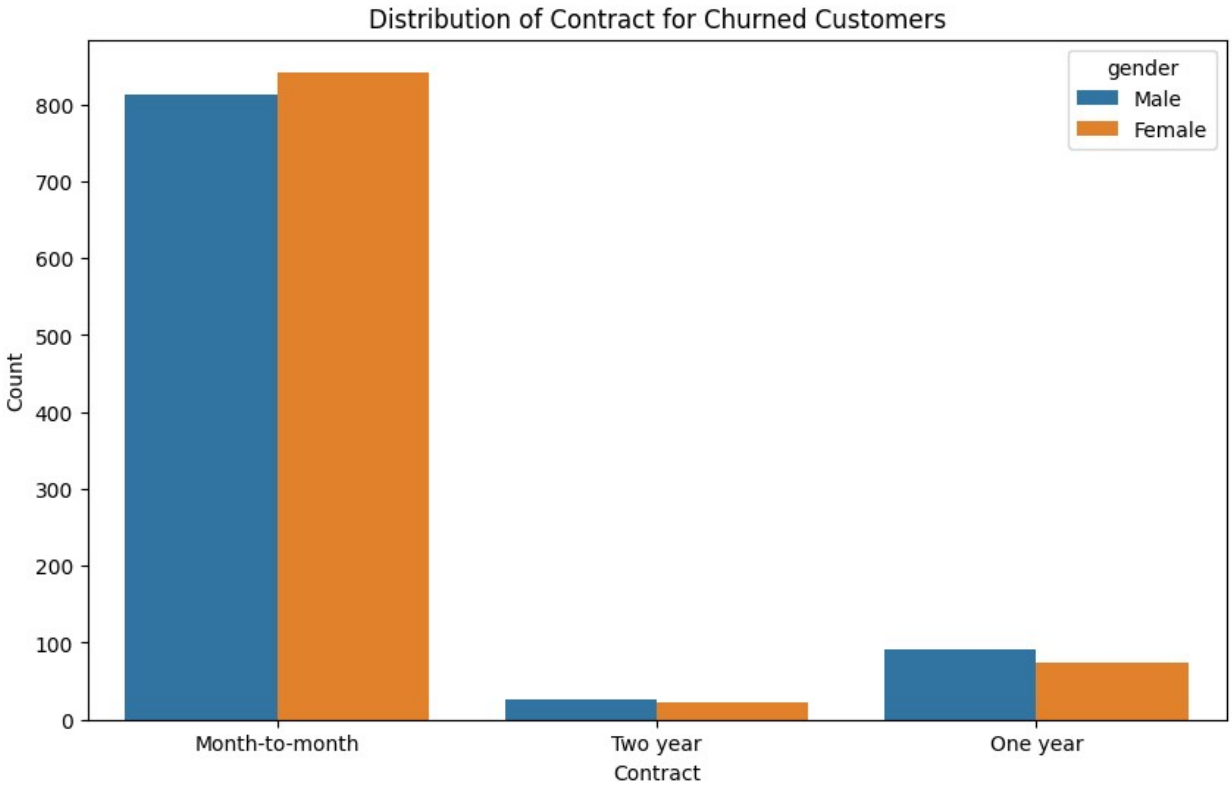
```
new_df1_target0=telco_data.loc[telco_data["Churn"]==0]
new_df1_target1=telco_data.loc[telco_data["Churn"]==1]

def simple_countplot(df, col, title, hue=None):
    plt.figure(figsize=(10, 6))  # Set figure size
    sns.countplot(data=df, x=col, hue=hue)  # Create count plot
    plt.title(title)  # Set the title
    # plt.xticks(rotation=45)  # Rotate x-tick labels for better
readability
```

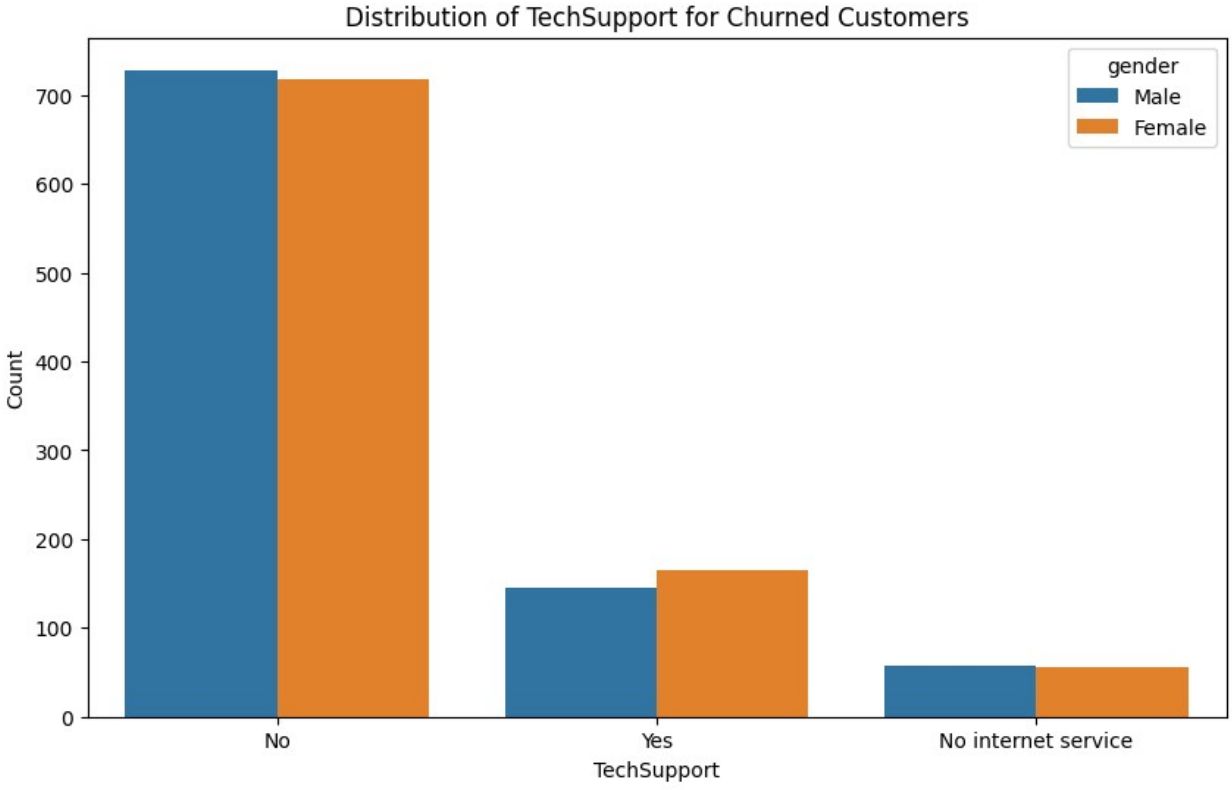
```
    plt.ylabel("Count")  # Label for y-axis
    plt.xlabel(col)  # Label for x-axis
    plt.show()  # Show the plot

# Example usage
simple_countplot(new_df1_target1, col='Partner', title='Distribution
of Partner for Churned Customers', hue='gender')
simple_countplot(new_df1_target0,col='Partner',title='Distribution of
Gender for Non Churned Customers',hue='gender')
simple_countplot(new_df1_target1,col='PaymentMethod',title='Distributi
on of PaymentMethod for Churned Customers',hue='gender')
simple_countplot(new_df1_target1,col='Contract',title='Distribution of
Contract for Churned Customers',hue='gender')
simple_countplot(new_df1_target1,col='TechSupport',title='Distribution
of TechSupport for Churned Customers',hue='gender')
simple_countplot(new_df1_target1,col='SeniorCitizen',title='Distributi
on of SeniorCitizen for Churned Customers',hue='gender')
```

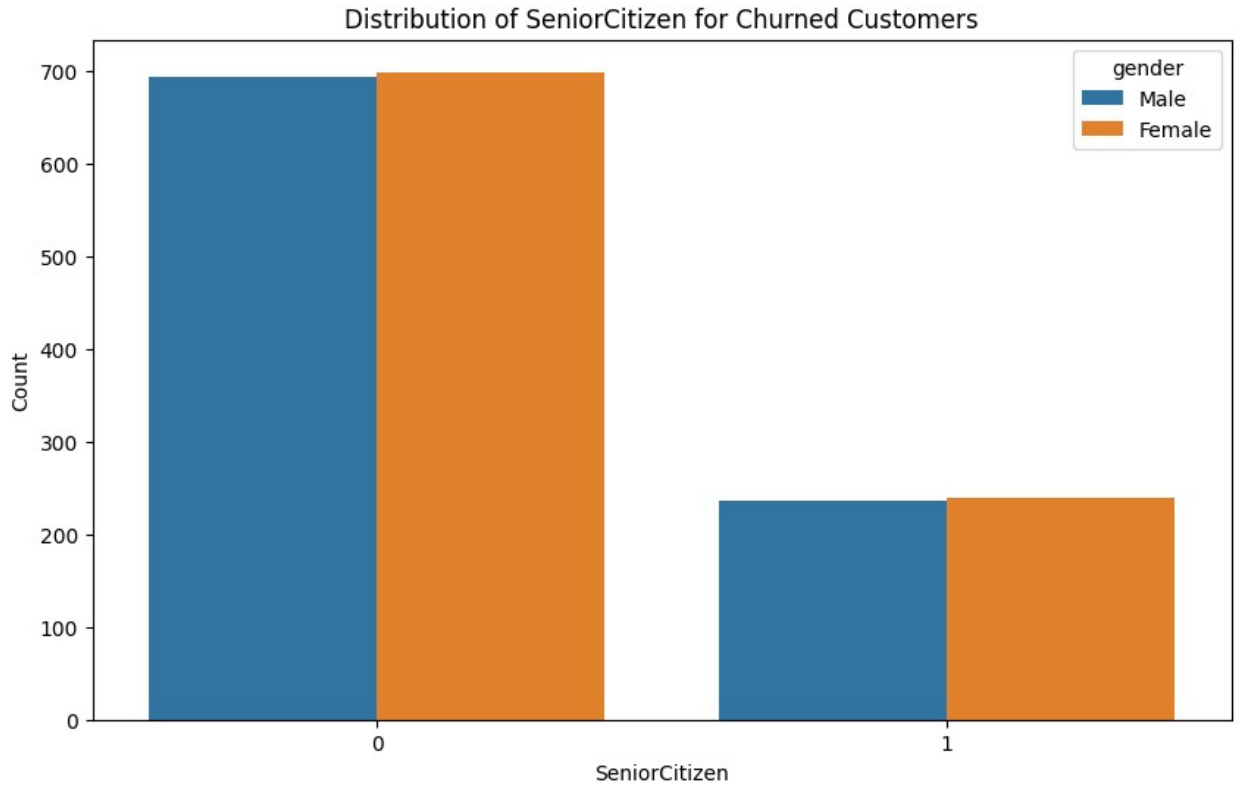



Distribution of Gender for Non Churned Customers
gender
Female
Male
1400
1200
1000
800
600
400
200
0
Count
Yes
No
Partner


Distribution of PaymentMethod for Churned Customers
gender
Male
Female
500
400
300
200
100
0
Count
Mailed check
Electronic check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod


Distribution of Contract for Churned Customers
gender
Male
Female
Count
0
100
200
300
400
500
600
700
800
Month-to-month
Two year
One year
Contract


Distribution of TechSupport for Churned Customers
gender
Male
Female
Count
0
100
200
300
400
500
600
700
No
Yes
No internet service
TechSupport



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 :)

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