CODE CLAUSE PROJECT

PROJECT NAME - Churn Prediction in Telecom Industry using Logistic Regression

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import plotly.express as px
```

Importing Datasets

In [2]:

```
telecom_data = pd.read_csv('D:/CODE CLAUSE DATA SCIENCE/CHURN PREDICTION/archive/Telc
```

In [3]:

```
telecom_data.head()
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleL
0	7590-VHVEG	Female	0	Yes	No	1	No	No ph ser
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 ph ser
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

In [4]:

```
telecom_data.shape
```

Out[4]:

```
(7043, 21)
```

In [5]:

```
telecom_data.describe()
```

Out[5]:

	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 [6]:

```
# Checking Null Values
telecom_data.notnull().sum()
```

Out[6]:

```
customerID          7043
gender              7043
SeniorCitizen       7043
Partner             7043
Dependents          7043
tenure              7043
PhoneService        7043
MultipleLines       7043
InternetService     7043
OnlineSecurity      7043
OnlineBackup        7043
DeviceProtection    7043
TechSupport         7043
StreamingTV         7043
StreamingMovies     7043
Contract            7043
PaperlessBilling    7043
PaymentMethod       7043
MonthlyCharges      7043
TotalCharges        7043
Churn               7043
dtype: int64
```

In [7]:

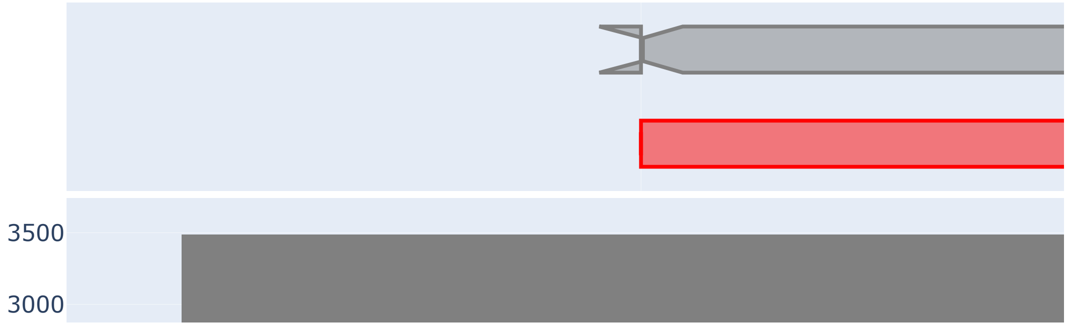
```
#There is no missing value in our
```

In [8]:

```
%matplotlib inline
```

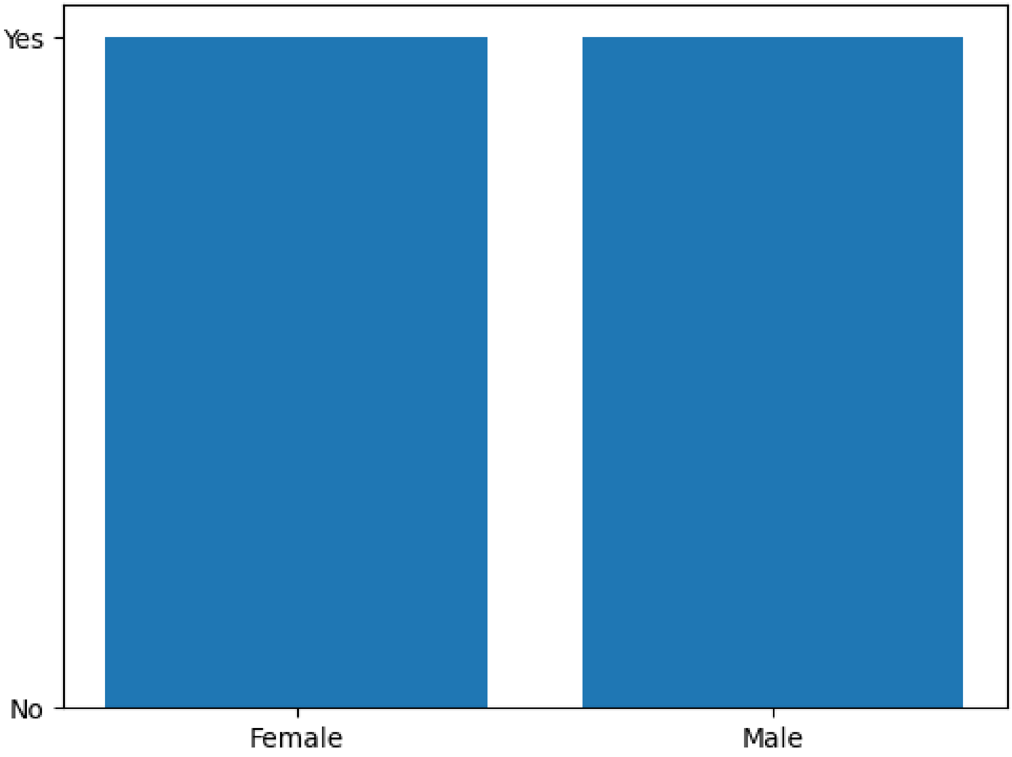
In [9]:

```
telecom_hist = px.histogram(telecom_data, x='gender',color='Churn',marginal='box', co
telecom_hist.update_layout(bargap=0.2)
```



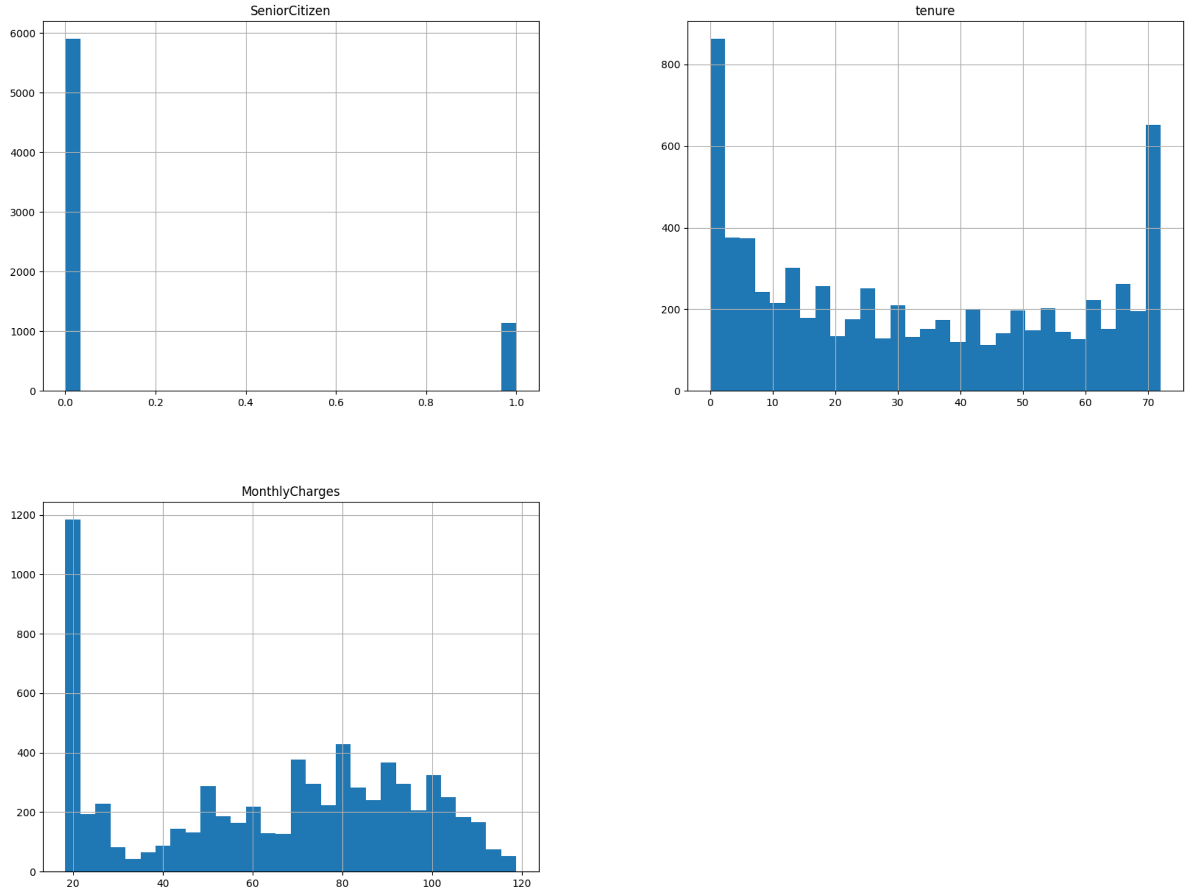
```
In [10]: plt.bar(telecom_data['gender'],telecom_data['Churn'])
```

```
Out[10]: <BarContainer object of 7043 artists>
```



```
In [11]: telecom_data.hist(bins = 30, figsize=(20,15))
```

```
Out[11]: array([[<Axes: title={'center': 'SeniorCitizen'}>,
                 <Axes: title={'center': 'tenure'}>],
                [<Axes: title={'center': 'MonthlyCharges'}>, <Axes: >]],
               dtype=object)
```

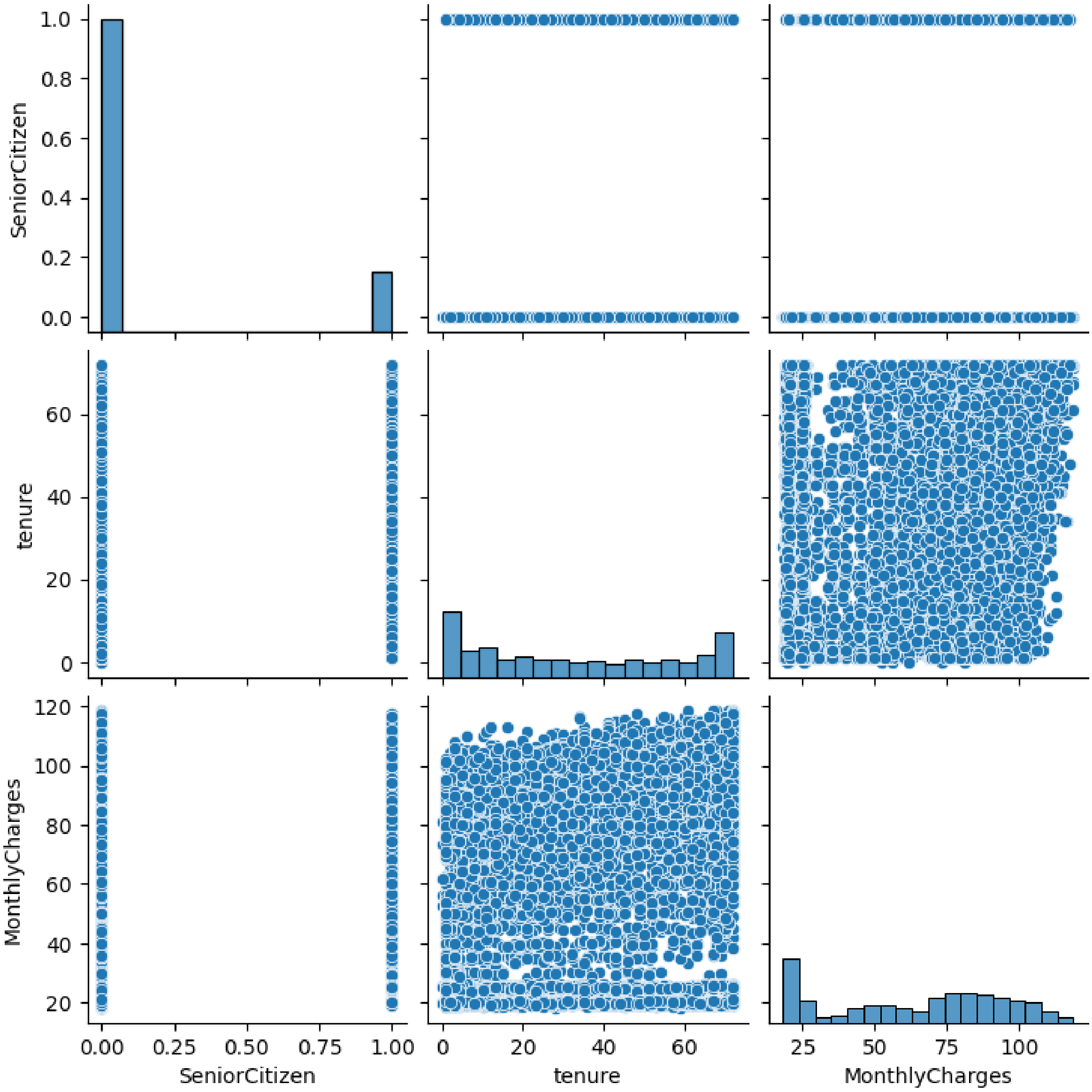


```
In [12]: sns.pairplot(telecom_data)
```

```
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgr
id.py:118: UserWarning:

The figure layout has changed to tight
```

```
Out[12]: <seaborn.axisgrid.PairGrid at 0x15a311c6d10>
```



Cleaning Data

```
In [13]: #Removing gender, customerID,tenture they are not usefull
```

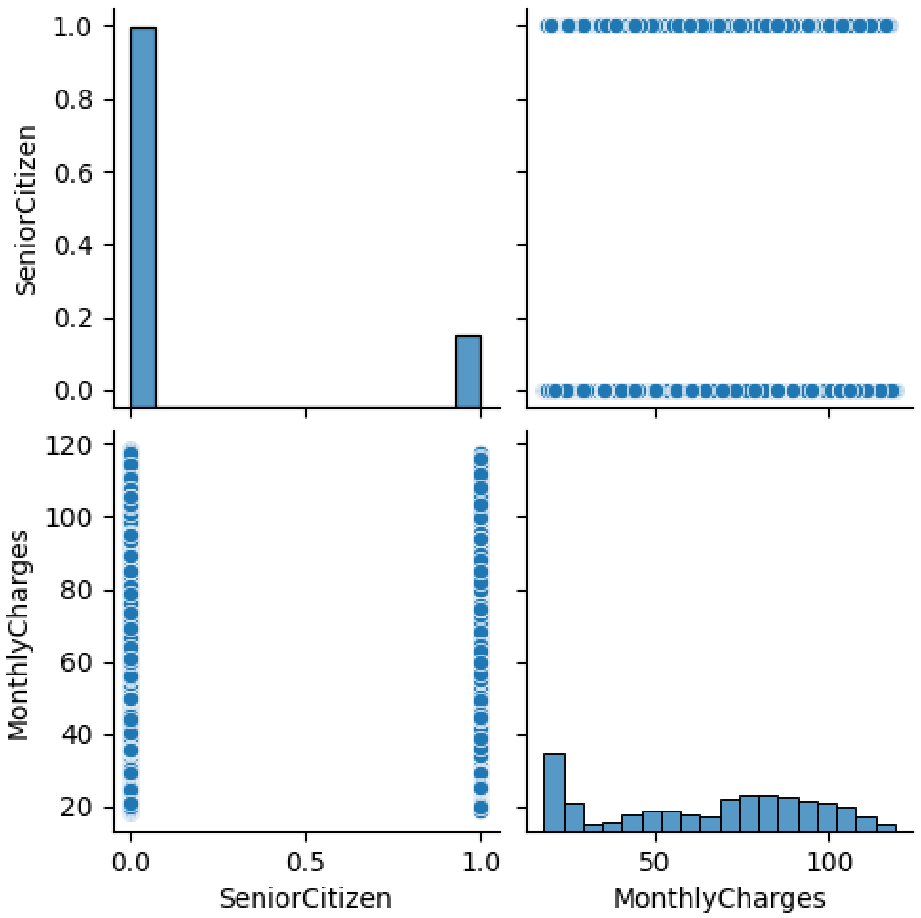
```
In [14]: col = ['gender','customerID','tenure']
         telecom_data = telecom_data.drop(col,axis = 1)
```

```
In [15]: sns.pairplot(telecom_data)
```

```
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgr
id.py:118: UserWarning:

The figure layout has changed to tight
```

```
Out[15]: <seaborn.axisgrid.PairGrid at 0x15a31d90dd0>
```



In [16]:
```
telecom_data.head()
```

Out[16]:

	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineS
0	0	Yes	No	No	No phone service	DSL	
1	0	No	No	Yes	No	DSL	
2	0	No	No	Yes	No	DSL	
3	0	No	No	No	No phone service	DSL	
4	0	No	No	Yes	No	Fiber optic	

In [17]:
```
telecom_data['TotalCharges'].notnull().sum()
```

Out[17]:
```
7043
```

In [18]:
```
telecom_data['MonthlyCharges'].describe()
```

Out[18]:

```
count    7043.000000
mean       64.761692
std        30.090047
min        18.250000
25%        35.500000
50%        70.350000
75%        89.850000
max       118.750000
Name: MonthlyCharges, dtype: float64
```

In [19]:

```
telecom_data['TotalCharges'].describe()
#the data type of the Total Charges is Object so we will change that
```

Out[19]:

```
count     7043
unique    6531
top           
freq        11
Name: TotalCharges, dtype: object
```

In [20]:

```
#due to string(" ") at 488 position you can not change the TotalCharges into Int
#so we will be removing/replacing that string which is --> " "
telecom_data['TotalCharges'] = telecom_data['TotalCharges'].replace(" ",np.nan)
telecom_data['TotalCharges'] = pd.to_numeric(telecom_data['TotalCharges'], errors = '
#dropping all the rows in which there is a null value
telecom_data = telecom_data.dropna(how = "any", axis = 0) #removing all the rows whic
```

In [21]:

```
telecom_data['TotalCharges'].describe()
```

Out[21]:

```
count    7032.000000
mean     2283.300441
std      2266.771362
min        18.800000
25%       401.450000
50%      1397.475000
75%      3794.737500
max      8684.800000
Name: TotalCharges, dtype: float64
```

In [22]:

```
telecom_data.notnull().sum()
```

Out[22]:

```
SeniorCitizen       7032
Partner             7032
Dependents          7032
PhoneService        7032
MultipleLines       7032
InternetService     7032
OnlineSecurity      7032
OnlineBackup        7032
DeviceProtection    7032
TechSupport         7032
StreamingTV         7032
StreamingMovies     7032
Contract            7032
PaperlessBilling    7032
PaymentMethod       7032
MonthlyCharges      7032
TotalCharges        7032
Churn               7032
dtype: int64
```

In [23]:

```
#Total Charges has null values in it
```

In [24]:

```
telecom_data.isnull().sum()
```

Out[24]:

```
SeniorCitizen       0
Partner             0
Dependents          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
```

EDA(Exploratory Data Analysis)

In [25]:

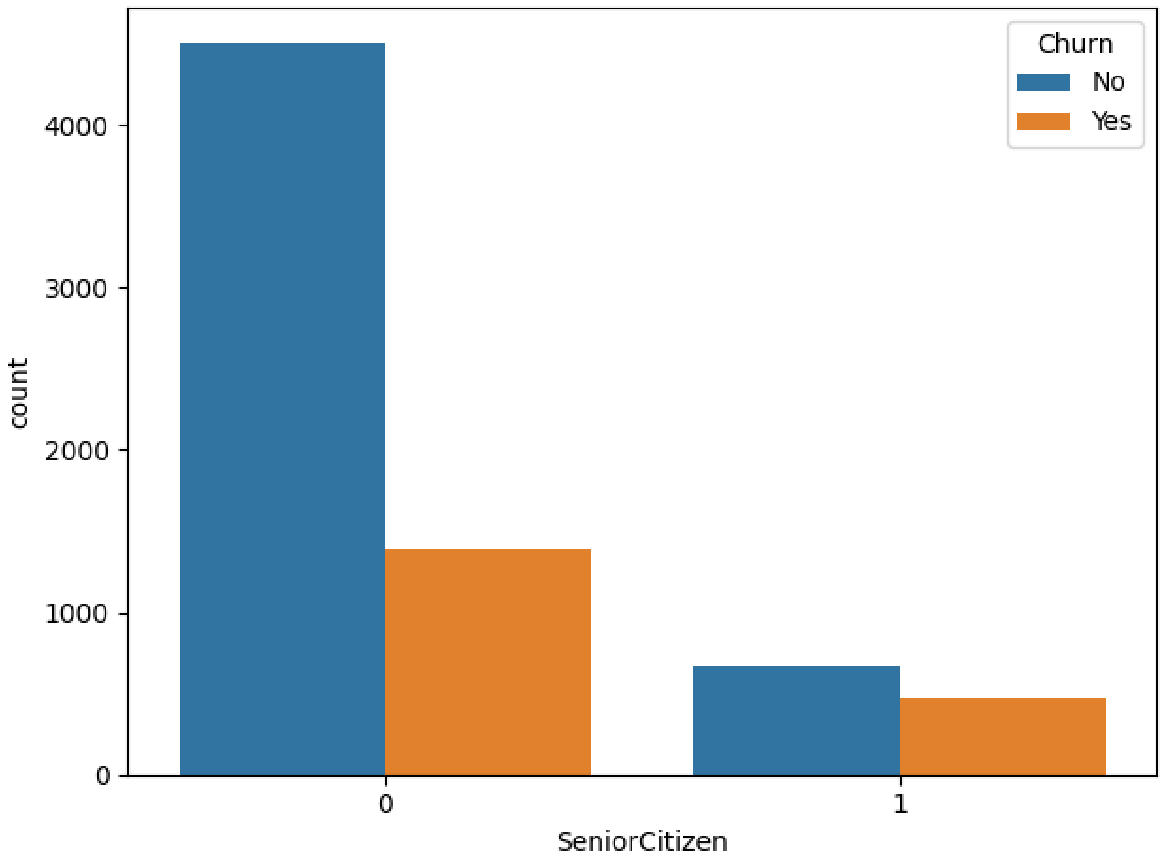
```
telecom_data['Churn'].describe()
```

Out[25]:

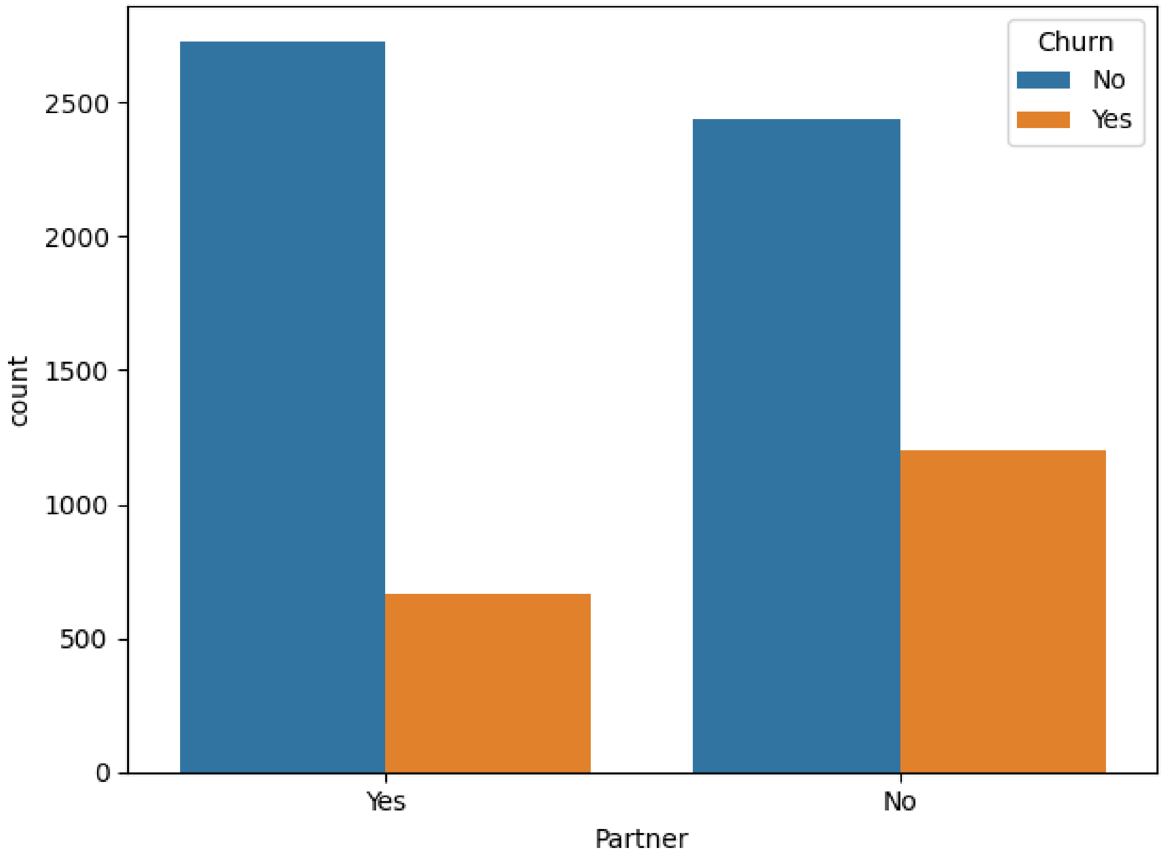
```
count     7032
unique       2
top         No
freq      5163
Name: Churn, dtype: object
```

In [26]:

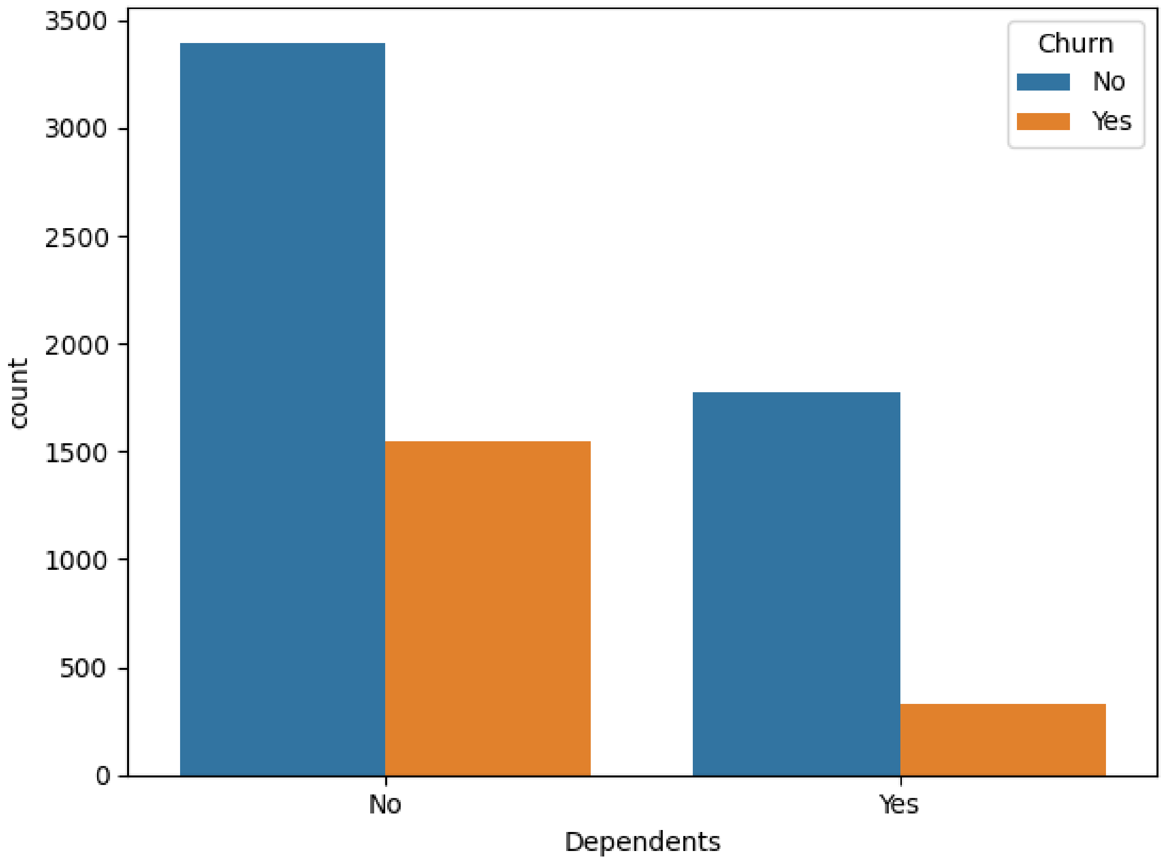
```
for i, predictor in enumerate(telecom_data.drop(columns=['Churn', 'TotalCharges', 'Mo
    ax = sns.countplot(data =telecom_data, x = predictor, hue='Churn')
    if predictor == "PaymentMethod":
        ax.set_xticklabels(ax.get_xticklabels(), fontsize=7)
        plt.tight_layout()
        plt.show()
    else:
        plt.tight_layout()
        plt.show()
```

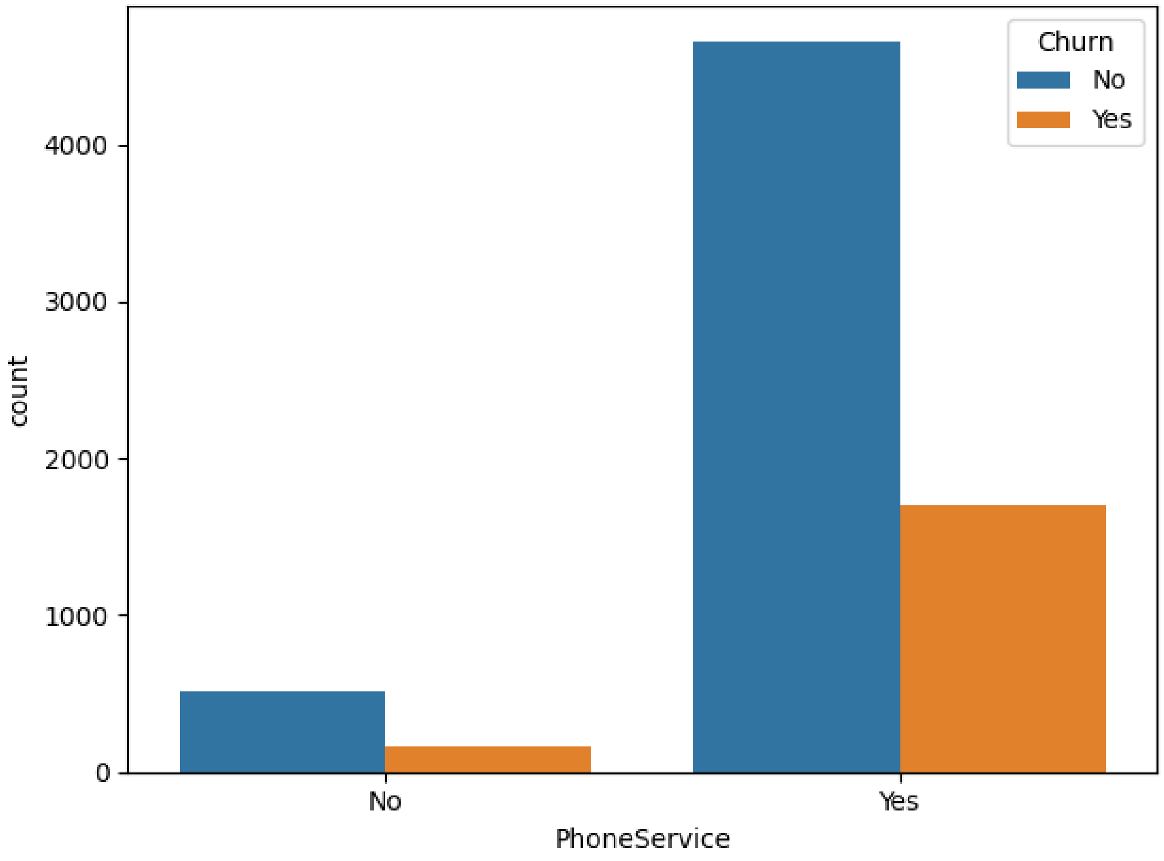


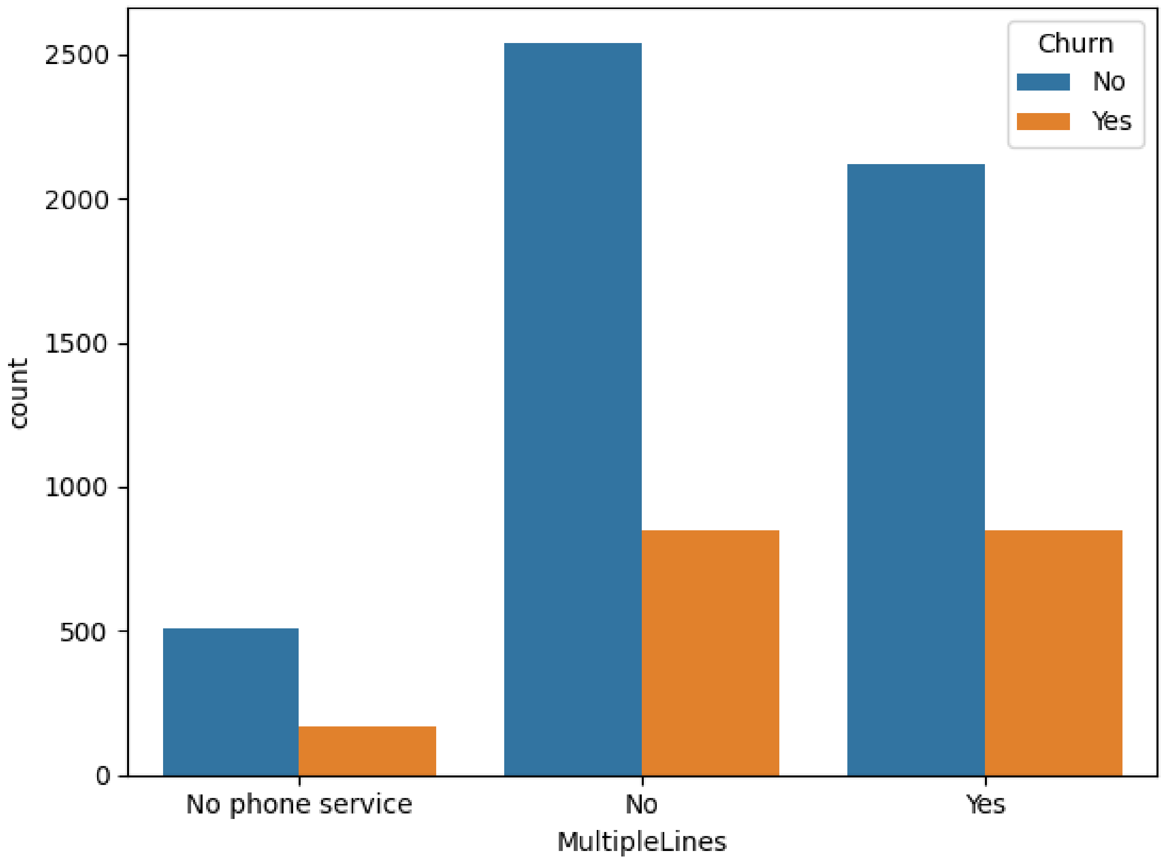
count
0
1000
2000
3000
4000
0
1
SeniorCitizen
Churn
No
Yes

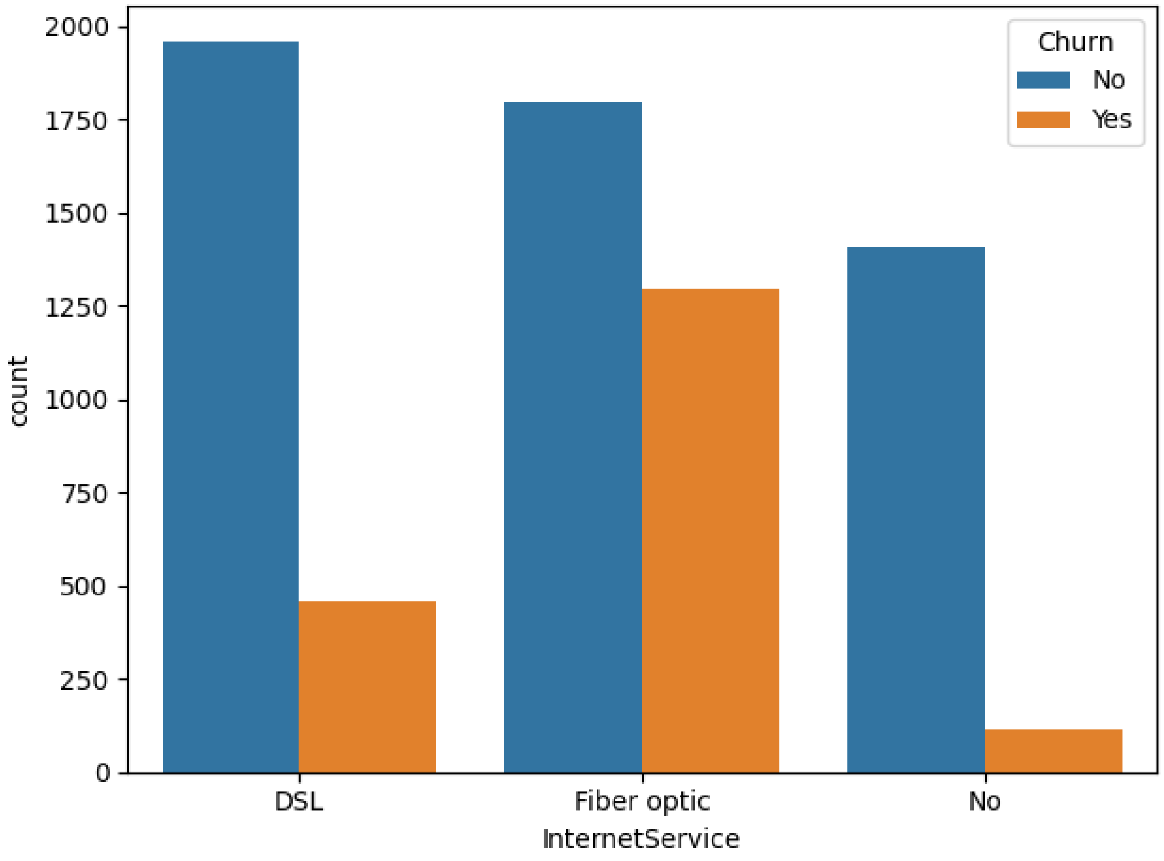


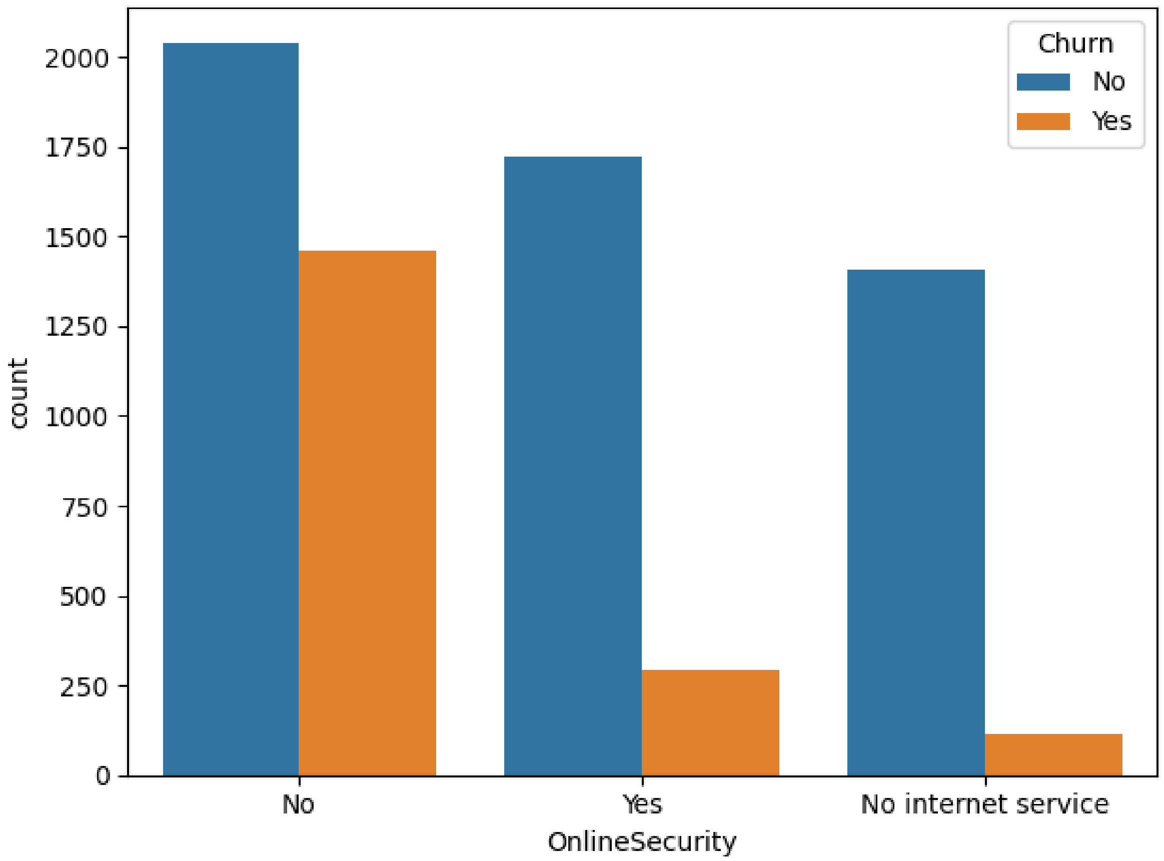
count
0
500
1000
1500
2000
2500
Yes
No
Partner
Churn
No
Yes

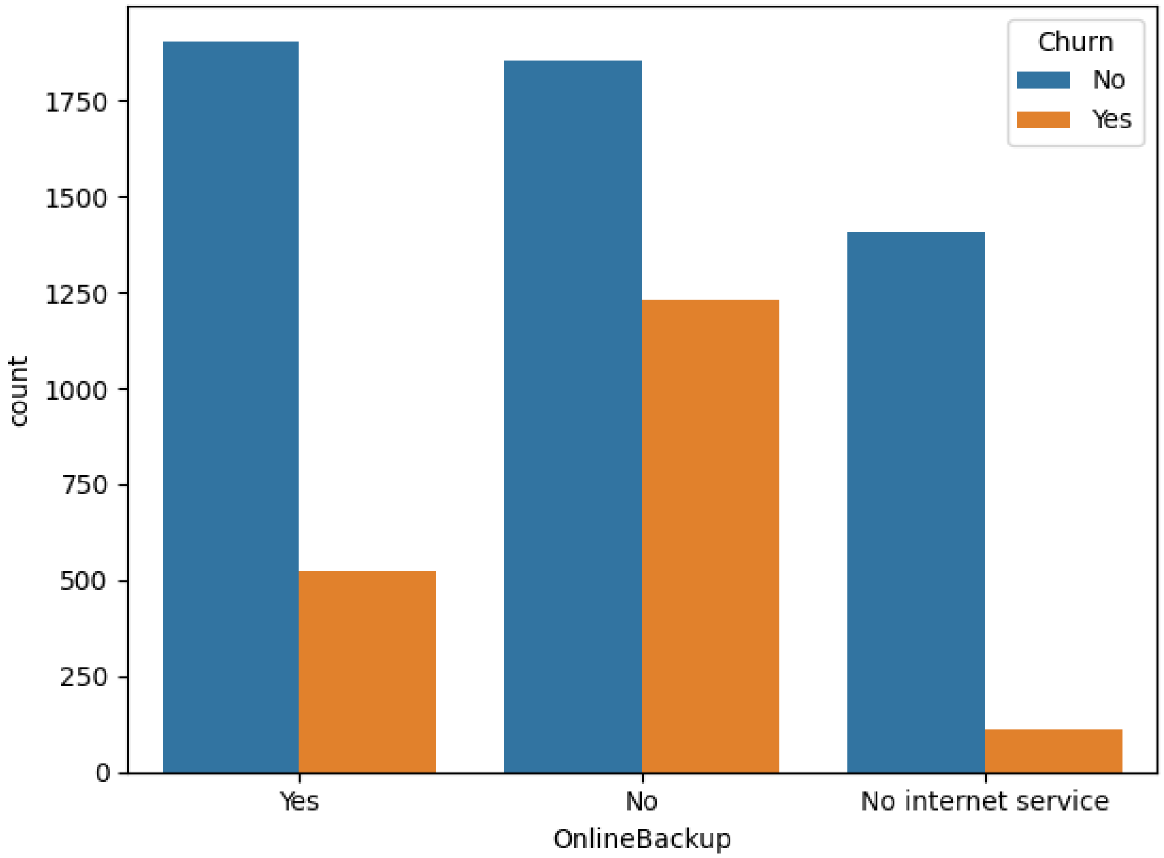

Churn
No
Yes
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents

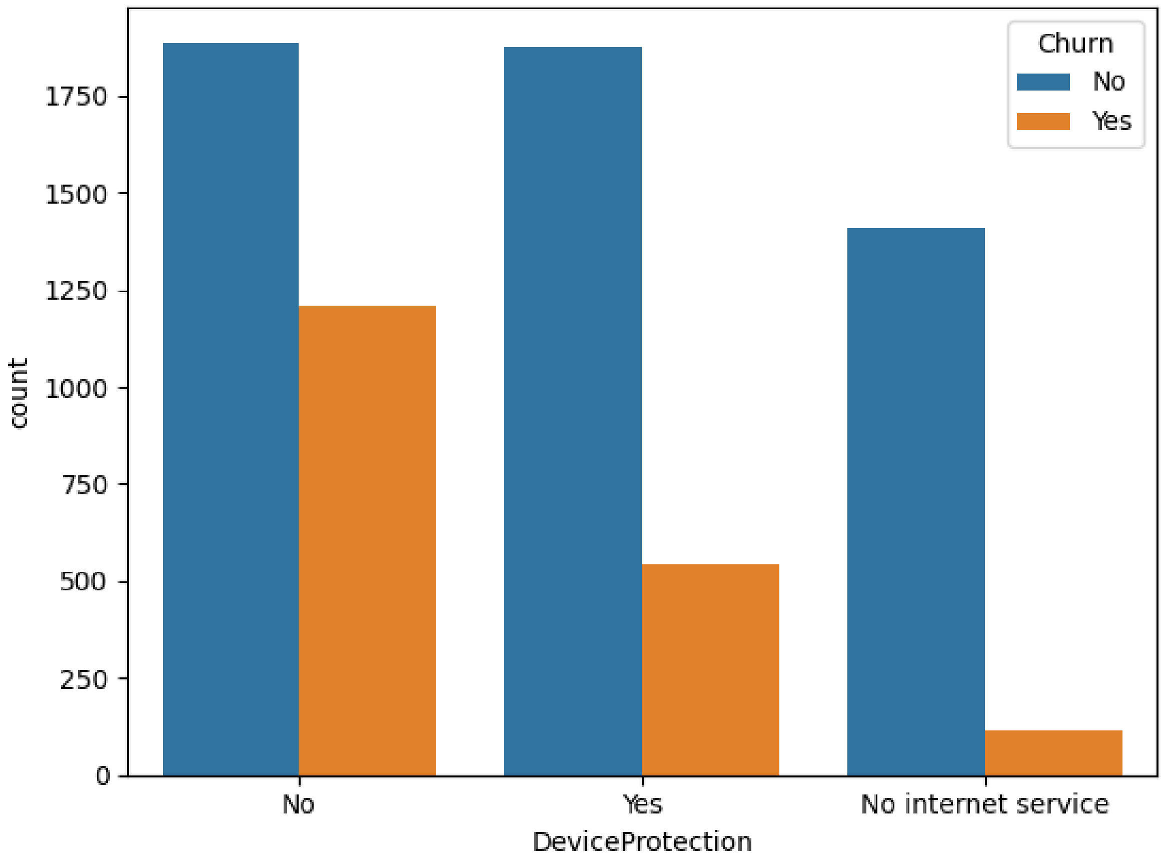

Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PhoneService

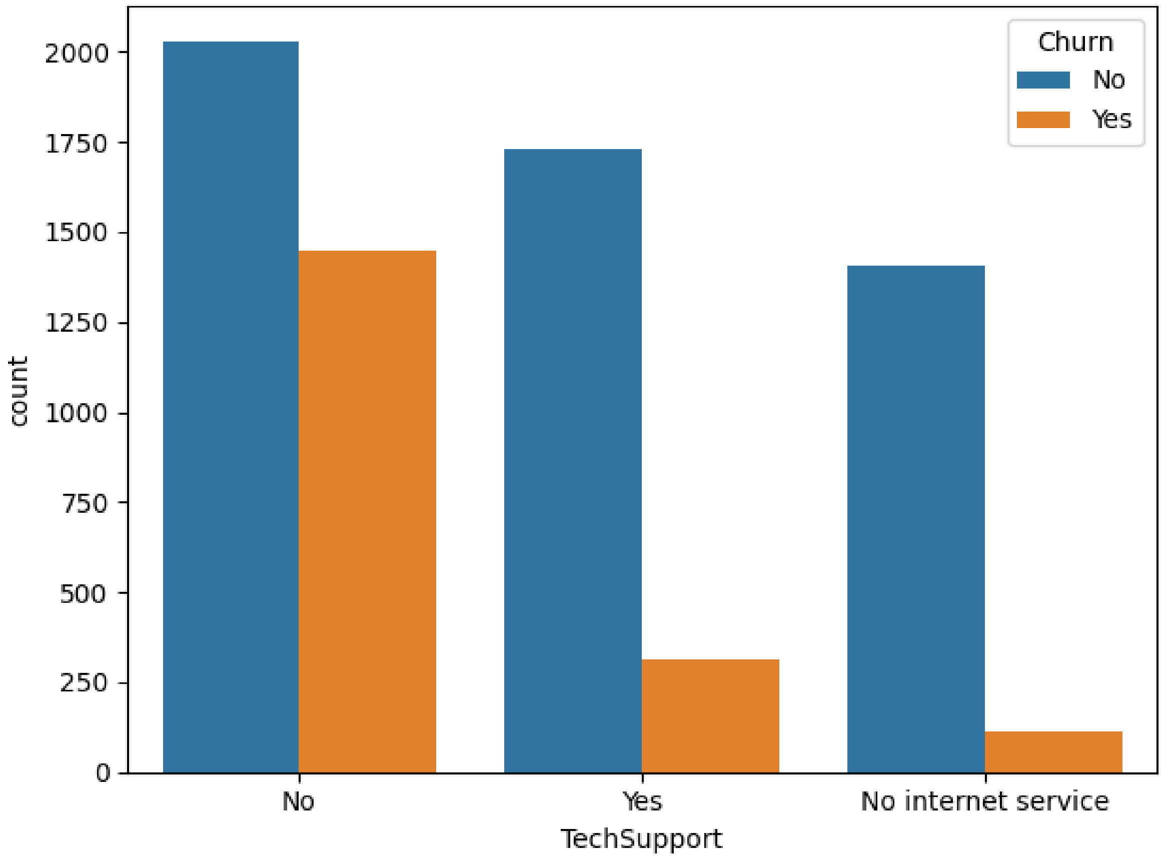

Churn
No
Yes
count
0
500
1000
1500
2000
2500
No phone service
No
Yes
MultipleLines

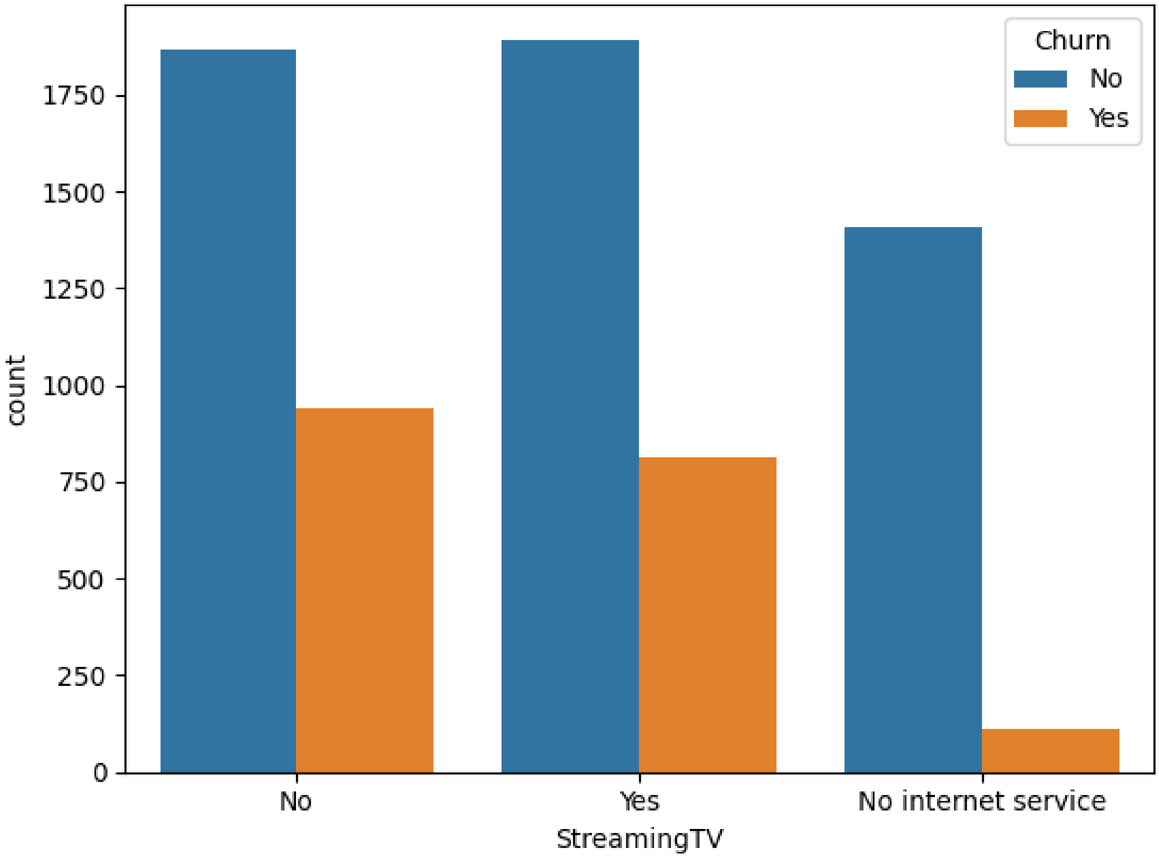

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
DSL
Fiber optic
No
InternetService

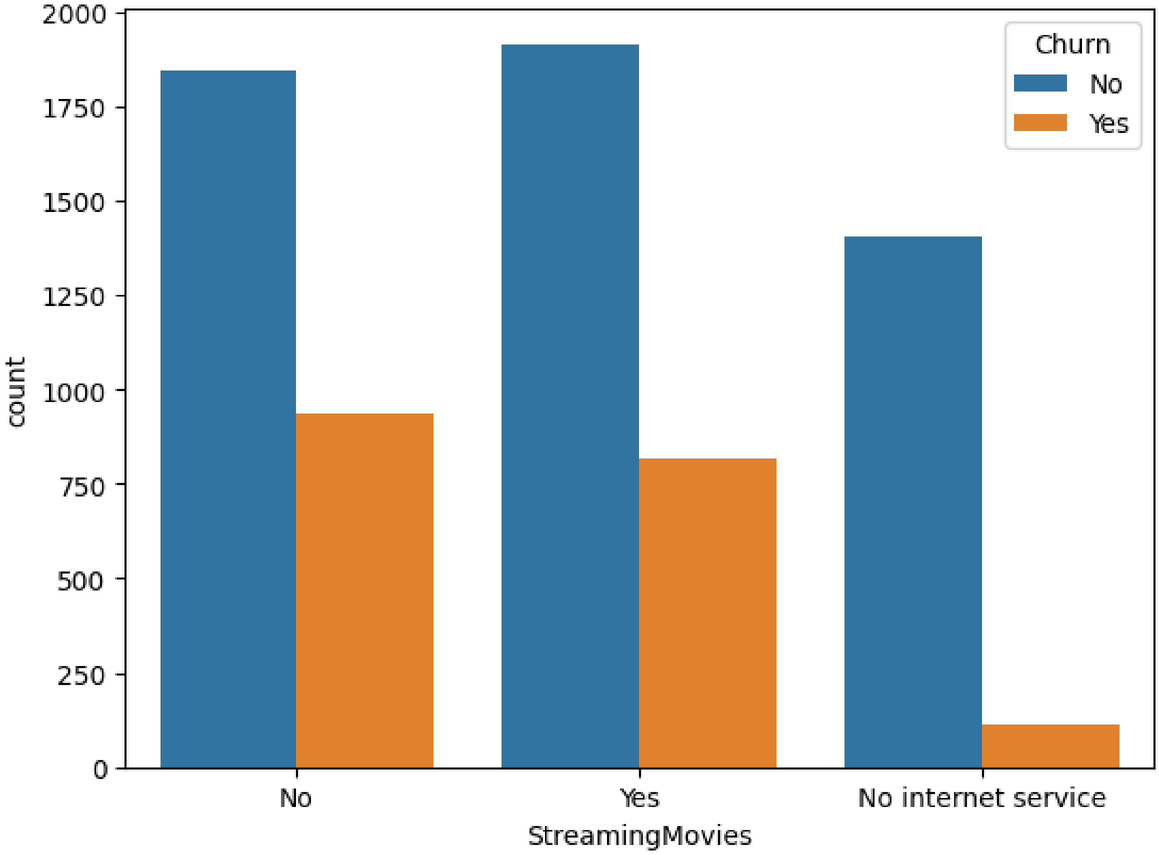

Churn
🟦 No
🟧 Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
OnlineSecurity

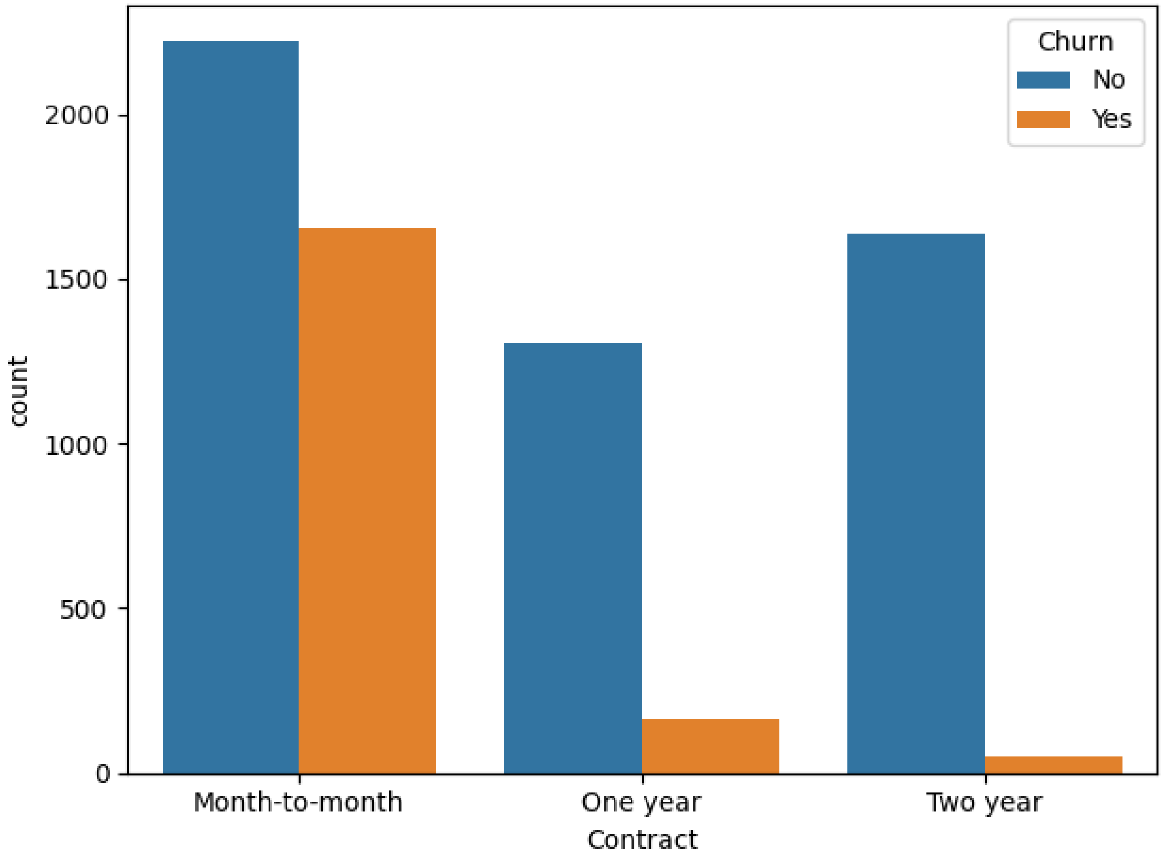

Churn
🟦 No
🟧 Yes
count
0
250
500
750
1000
1250
1500
1750
Yes
No
No internet service
OnlineBackup

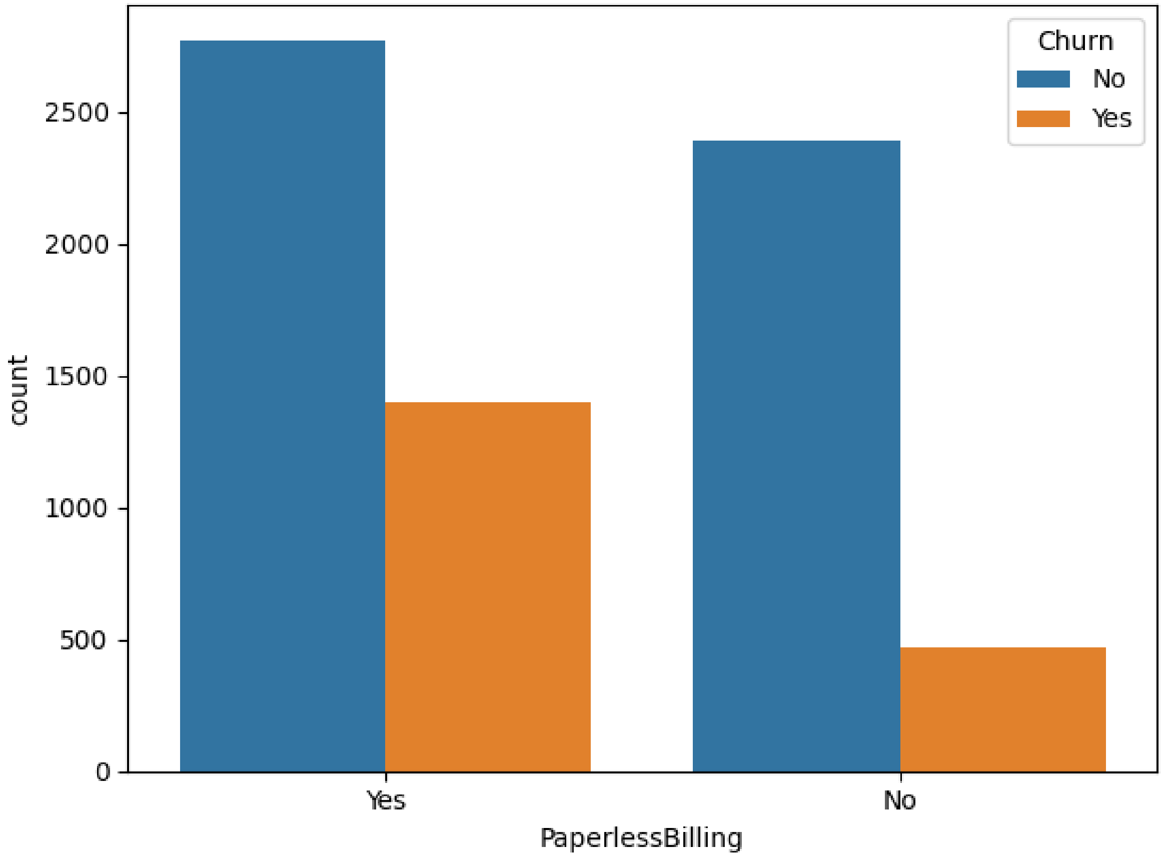

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
DeviceProtection

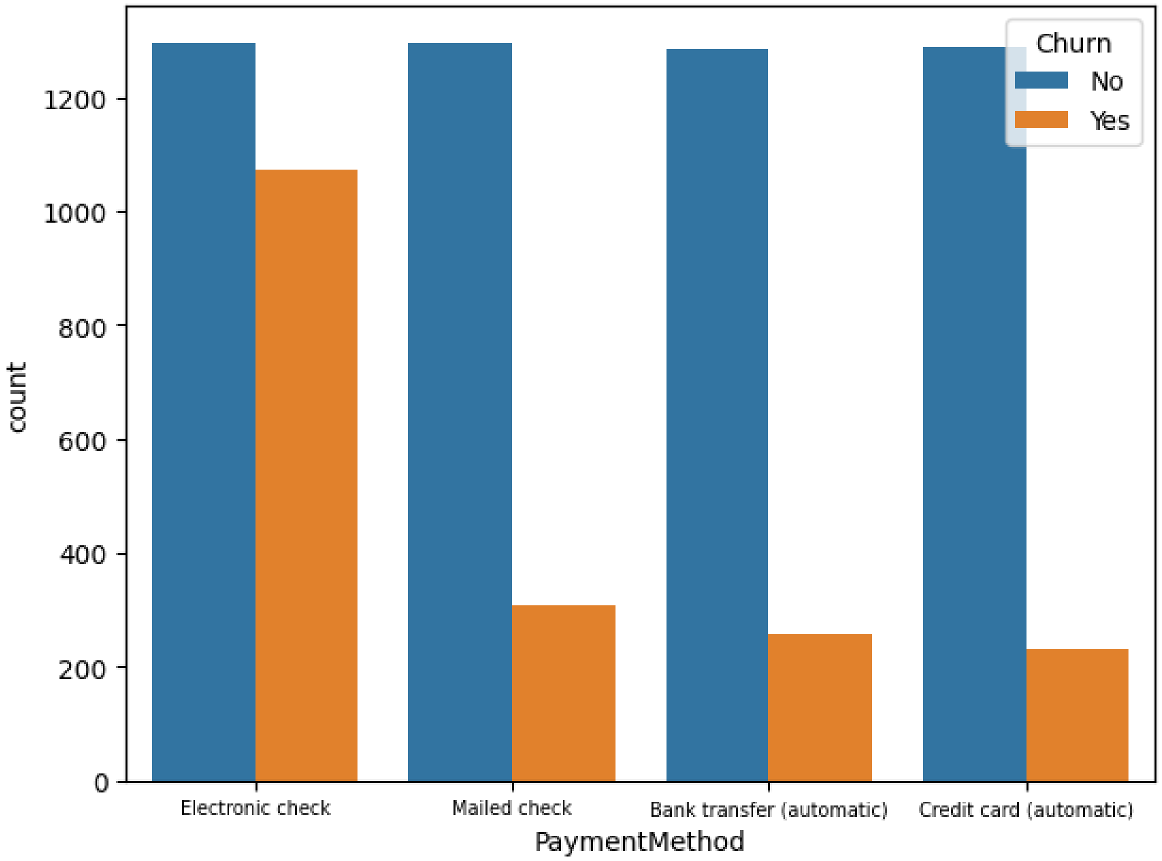

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
TechSupport


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


count
0
500
1000
1500
2000
Month-to-month
One year
Two year
Contract
Churn
No
Yes


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



In [27]:

```
#converting Yes as 1 and No as 0
telecom_data["Churn"] = telecom_data["Churn"].replace(['Yes','No'],[1,0])
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_8512\1073425660.py:2: SettingWithCopyWarnin
g:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
```

In [28]:

```
telecom_data
```

Out[28]:

	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	Onli
0	0	Yes	No	No	No phone service	DSL	
1	0	No	No	Yes	No	DSL	
2	0	No	No	Yes	No	DSL	
3	0	No	No	No	No phone service	DSL	
4	0	No	No	Yes	No	Fiber optic	
...	...	...	...	...	...	...	
7038	0	Yes	Yes	Yes	Yes	DSL	
7039	0	Yes	Yes	Yes	Yes	Fiber optic	
7040	0	Yes	Yes	No	No phone service	DSL	
7041	1	Yes	No	Yes	Yes	Fiber optic	
7042	0	No	No	Yes	No	Fiber optic	

7032 rows × 18 columns

In [29]:

```
telecom_data_dummies = pd.get_dummies(telecom_data)
```

In [30]:

```
telecom_data_dummies
```

Out[30]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	Partner_No	Partner_Yes	Depend
0	0	29.85	29.85	0	False	True	
1	0	56.95	1889.50	0	True	False	
2	0	53.85	108.15	1	True	False	
3	0	42.30	1840.75	0	True	False	
4	0	70.70	151.65	1	True	False	
...	...	...	...	...	...	...	
7038	0	84.80	1990.50	0	False	True	
7039	0	103.20	7362.90	0	False	True	
7040	0	29.60	346.45	0	False	True	
7041	1	74.40	306.60	1	False	True	
7042	0	105.65	6844.50	0	True	False	

7032 rows × 43 columns

In [31]:

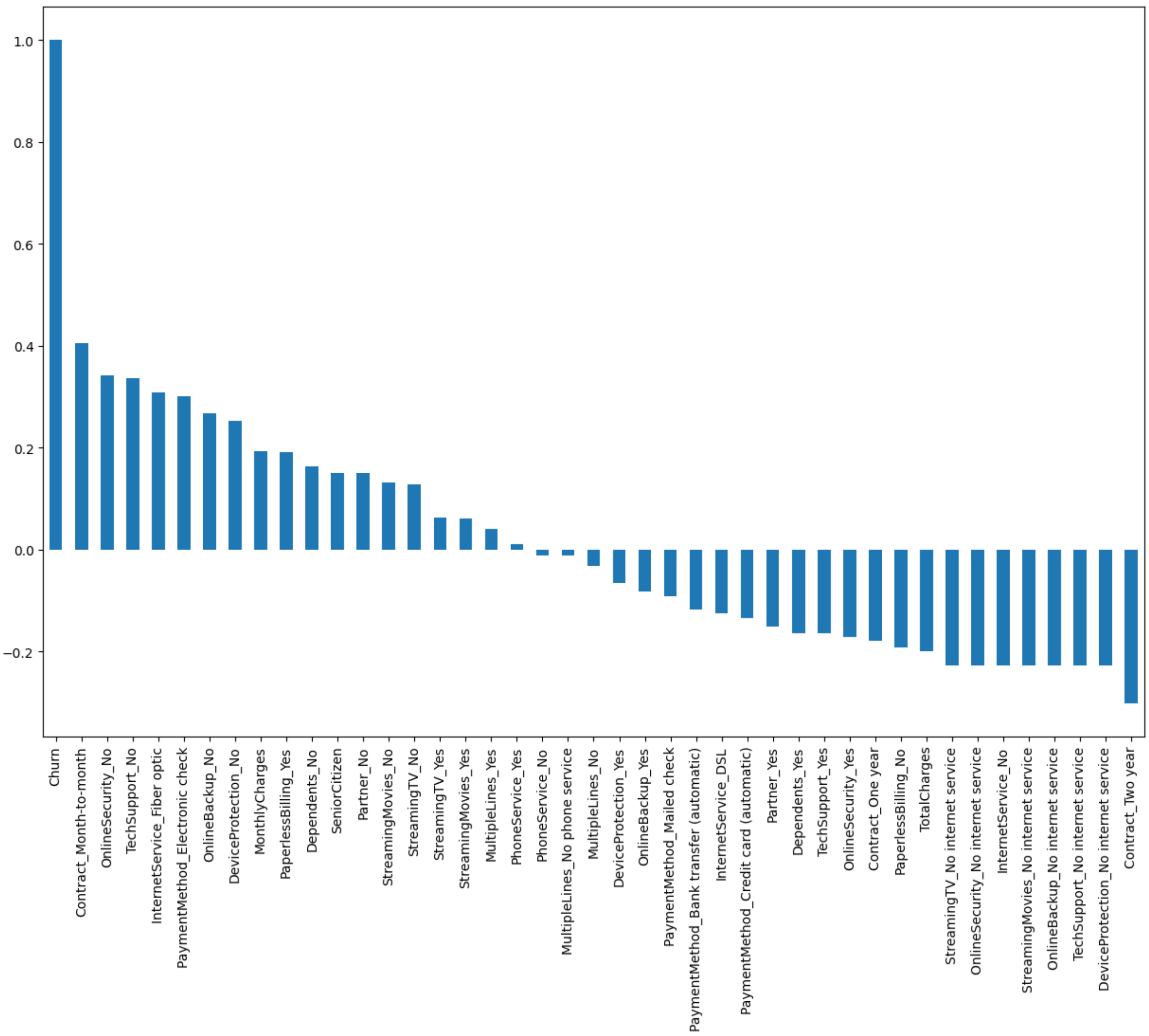
```
churn_corr_matrix = telecom_data_dummies.corr()
```

In [32]:

```
churn_corr_matrix['Churn'].sort_values(ascending = False).plot(kind='bar',figsize = (
```

Out[32]:

```
<Axes: >
```



In [33]:

```
churn_corr_matrix['Churn'].sort_values(ascending = False)
```

Out[33]:

```
Churn                                       1.000000
Contract_Month-to-month                     0.404565
OnlineSecurity_No                           0.342235
TechSupport_No                              0.336877
InternetService_Fiber optic                 0.307463
PaymentMethod_Electronic check              0.301455
OnlineBackup_No                             0.267595
DeviceProtection_No                         0.252056
MonthlyCharges                              0.192858
PaperlessBilling_Yes                        0.191454
Dependents_No                               0.163128
SeniorCitizen                               0.150541
Partner_No                                  0.149982
StreamingMovies_No                          0.130920
StreamingTV_No                              0.128435
StreamingTV_Yes                             0.063254
StreamingMovies_Yes                         0.060860
MultipleLines_Yes                           0.040033
PhoneService_Yes                            0.011691
PhoneService_No                            -0.011691
MultipleLines_No phone service             -0.011691
MultipleLines_No                           -0.032654
DeviceProtection_Yes                       -0.066193
OnlineBackup_Yes                           -0.082307
PaymentMethod_Mailed check                 -0.090773
PaymentMethod_Bank transfer (automatic)    -0.118136
InternetService_DSL                        -0.124141
PaymentMethod_Credit card (automatic)      -0.134687
Partner_Yes                                -0.149982
Dependents_Yes                             -0.163128
TechSupport_Yes                            -0.164716
OnlineSecurity_Yes                         -0.171270
Contract_One year                          -0.178225
PaperlessBilling_No                        -0.191454
TotalCharges                               -0.199484
StreamingTV_No internet service            -0.227578
OnlineSecurity_No internet service         -0.227578
InternetService_No                         -0.227578
StreamingMovies_No internet service        -0.227578
OnlineBackup_No internet service           -0.227578
TechSupport_No internet service            -0.227578
DeviceProtection_No internet service       -0.227578
Contract_Two year                          -0.301552
Name: Churn, dtype: float64
```

In [34]:

```
x = telecom_data_dummies.drop('Churn',axis = 1)
```

In [35]:

```
x
```

Out[35]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Partner_No	Partner_Yes	Dependents_No
0	0	29.85	29.85	False	True	True
1	0	56.95	1889.50	True	False	True
2	0	53.85	108.15	True	False	True
3	0	42.30	1840.75	True	False	True
4	0	70.70	151.65	True	False	True
...	...	...	...	...	...	...
7038	0	84.80	1990.50	False	True	False
7039	0	103.20	7362.90	False	True	False
7040	0	29.60	346.45	False	True	False
7041	1	74.40	306.60	False	True	True
7042	0	105.65	6844.50	True	False	True

7032 rows × 42 columns

In [36]:
```
y = telecom_data_dummies['Churn']
```

In [37]:
```
y
```

Out[37]:
```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7032, dtype: int64
```

In [38]:
```
x.shape
```

Out[38]:
```
(7032, 42)
```

In [39]:
```
y.shape
```

Out[39]:
```
(7032,)
```

In [40]:
```
y.value_counts()
```

Out[40]:
```
Churn
0    5163
1    1869
Name: count, dtype: int64
```

Variable Imbalancing

SMOTE for Imbalanced Classification with Python

In [41]:
```
from imblearn.over_sampling import SMOTE
```

In [42]:
```
smote = SMOTE(random_state=0)
```

In [43]:
```
x_resampled_smote, y_resampled_smote = smote.fit_resample(x,y)
```

In [44]:
```
y_resampled_smote.value_counts()
```

Out[44]:
```
Churn
0    5163
1    5163
Name: count, dtype: int64
```

In [45]:
```
x_resampled_smote
```

Out[45]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Partner_No	Partner_Yes	Dependents_N
0	0	29.850000	29.850000	False	True	Tru
1	0	56.950000	1889.500000	True	False	Tru
2	0	53.850000	108.150000	True	False	Tru
3	0	42.300000	1840.750000	True	False	Tru
4	0	70.700000	151.650000	True	False	Tru
...	...	...	...	...	...	.
10321	0	103.976753	242.804921	False	True	Tru
10322	0	35.824447	35.824447	True	False	Tru
10323	0	44.493077	1061.960339	True	True	Tru
10324	0	19.363055	19.363055	True	False	Tru
10325	0	96.922890	96.922890	True	False	Tru

10326 rows × 42 columns

In [46]:
```
y_resampled_smote.notnull().sum()
```

Out[46]:
```
10326
```

In [47]:
```
x_resampled_smote.notnull().sum()
```

Out[47]:

```
SeniorCitizen                              10326
MonthlyCharges                             10326
TotalCharges                               10326
Partner_No                                 10326
Partner_Yes                                10326
Dependents_No                              10326
Dependents_Yes                             10326
PhoneService_No                            10326
PhoneService_Yes                           10326
MultipleLines_No                           10326
MultipleLines_No phone service             10326
MultipleLines_Yes                          10326
InternetService_DSL                        10326
InternetService_Fiber optic                10326
InternetService_No                         10326
OnlineSecurity_No                          10326
OnlineSecurity_No internet service         10326
OnlineSecurity_Yes                         10326
OnlineBackup_No                            10326
OnlineBackup_No internet service           10326
OnlineBackup_Yes                           10326
DeviceProtection_No                        10326
DeviceProtection_No internet service       10326
DeviceProtection_Yes                       10326
TechSupport_No                             10326
TechSupport_No internet service            10326
TechSupport_Yes                            10326
StreamingTV_No                             10326
StreamingTV_No internet service            10326
StreamingTV_Yes                            10326
StreamingMovies_No                         10326
StreamingMovies_No internet service        10326
StreamingMovies_Yes                        10326
Contract_Month-to-month                    10326
Contract_One year                          10326
Contract_Two year                          10326
PaperlessBilling_No                        10326
PaperlessBilling_Yes                       10326
PaymentMethod_Bank transfer (automatic)    10326
PaymentMethod_Credit card (automatic)      10326
PaymentMethod_Electronic check             10326
PaymentMethod_Mailed check                 10326
dtype: int64
```

In [48]:

```python
from sklearn.linear_model import LogisticRegression
```

In [49]:

```python
#checking on imbalance data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [50]:

```python
LogReg = LogisticRegression(solver='lbfgs', max_iter=1000,multi_class='multinomial')
```

In [51]:

```python
LogReg.fit(x_train,y_train)
```

Out[51]:

```
▾                        LogisticRegression
LogisticRegression(max_iter=1000, multi_class='multinomial')
```

In [52]:

```python
y_pred = LogReg.predict(x_test)
```

In [53]:

```python
from sklearn.metrics import accuracy_score
```

In [54]:
```
accuracy_score(y_test,y_pred)
```

Out[54]:
```
0.7853589196872779
```

In [55]:
```
#checking on balanced data
x_smote_train,x_smote_test,y_smote_train,y_smote_test = train_test_split(x_resampled_
```

In [56]:
```
LogReg.fit(x_smote_train,y_smote_train)
```

Out[56]:
```
▾                        LogisticRegression
LogisticRegression(max_iter=1000, multi_class='multinomial')
```

In [57]:
```
y_smote_pred = LogReg.predict(x_smote_test)
```

In [58]:
```
accuracy_score(y_smote_test,y_smote_pred)
```

Out[58]:
```
0.8388189738625363
```

In [59]:
```
from sklearn.preprocessing import StandardScaler
```

In [60]:
```
std = StandardScaler()
```

In [61]:
```
std_train = std.fit_transform(x_smote_train)
std_test = std.transform(x_smote_test)
```

In [62]:
```
LogReg.fit(std_train,y_smote_train)
```

Out[62]:
```
▾                        LogisticRegression
LogisticRegression(max_iter=1000, multi_class='multinomial')
```

In [63]:
```
std_pred = LogReg.predict(std_test)
```

In [64]:
```
accuracy_score(std_pred,y_smote_test)
```

Out[64]:
```
0.8407550822846079
```

In [65]:
```
np.where(std_pred!=y_smote_test)
```

```
Out[65]: (array([  14,   20,   24,   31,   43,   48,   49,   57,   60,   63,   80,
                  81,   83,   87,   90,   98,  100,  102,  107,  108,  117,  118,
                 125,  126,  130,  136,  161,  162,  183,  193,  194,  207,  230,
                 236,  272,  274,  287,  289,  291,  296,  300,  306,  313,  321,
                 327,  328,  329,  330,  333,  335,  341,  346,  348,  359,  376,
                 380,  393,  397,  400,  414,  415,  421,  425,  427,  428,  434,
                 435,  439,  442,  449,  451,  463,  479,  489,  490,  491,  499,
                 509,  515,  521,  530,  532,  543,  546,  551,  555,  556,  562,
                 563,  571,  573,  575,  585,  588,  595,  602,  608,  612,  625,
                 629,  637,  645,  661,  691,  695,  705,  710,  724,  734,  739,
                 756,  757,  760,  774,  777,  783,  785,  789,  790,  791,  794,
                 799,  805,  814,  820,  821,  841,  855,  862,  865,  866,  869,
                 870,  874,  883,  888,  899,  902,  904,  909,  912,  921,  927,
                 929,  932,  938,  940,  947,  951,  954,  962,  964,  967,  970,
                 973,  974,  986, 1003, 1005, 1015, 1037, 1043, 1045, 1046, 1047,
                1052, 1064, 1066, 1075, 1076, 1095, 1111, 1112, 1115, 1126, 1131,
                1134, 1135, 1137, 1141, 1146, 1148, 1152, 1154, 1157, 1159, 1165,
                1167, 1170, 1181, 1187, 1190, 1198, 1205, 1212, 1216, 1217, 1225,
                1236, 1239, 1241, 1242, 1243, 1250, 1253, 1265, 1273, 1282, 1287,
                1292, 1300, 1301, 1309, 1316, 1332, 1350, 1362, 1375, 1377, 1390,
                1392, 1407, 1413, 1414, 1428, 1439, 1440, 1457, 1458, 1460, 1463,
                1470, 1476, 1483, 1484, 1491, 1496, 1504, 1507, 1514, 1517, 1518,
                1525, 1539, 1547, 1549, 1553, 1562, 1575, 1579, 1580, 1584, 1596,
                1605, 1606, 1614, 1617, 1622, 1625, 1629, 1630, 1631, 1647, 1651,
                1654, 1666, 1667, 1678, 1681, 1683, 1693, 1719, 1721, 1724, 1732,
                1733, 1741, 1748, 1752, 1754, 1768, 1777, 1787, 1791, 1798, 1801,
                1815, 1818, 1822, 1827, 1831, 1832, 1834, 1835, 1836, 1837, 1843,
                1850, 1854, 1874, 1880, 1890, 1902, 1909, 1912, 1917, 1919, 1920,
                1930, 1940, 1944, 1953, 1965, 1979, 1980, 1984, 1997, 2002, 2012,
                2018, 2028, 2029, 2038, 2039, 2041, 2043, 2046, 2050, 2056],
               dtype=int64),)
```

```
In [66]: y_smote_test.shape
```

```
Out[66]: (2066,)
```

```
In [ ]:
```