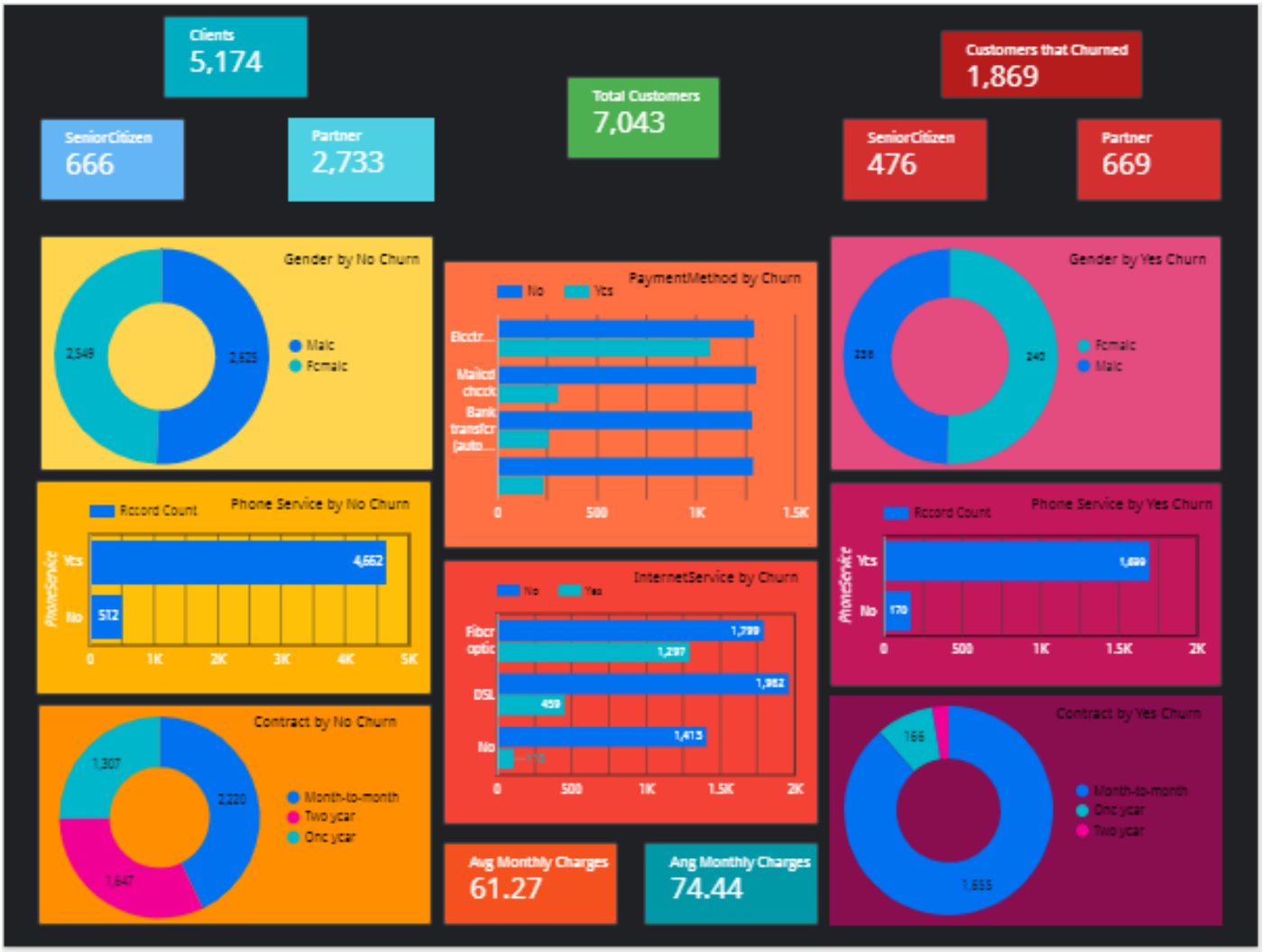
Assignment 8 – Telo Churn Analysis

Hajra Abdul Hai - 14893

URL: https://datastudio.google.com/reporting/8859bbd4-99f4-4340-b113-dfa05282f2de/page/U316B/edit

Dash Board:



I have divided my chart in 3 portions. The right most depicts the customers that stayed loyal, the middle graphs are for both types of customers and the left most are the graphs for churned customers.

On the top most score cards I wrote the customers that stayed loyal(client) on the right corner and the customers that churned at the left corner scorecard.

After that in the displayed scored cards I showed the number of senior citizens and the customers who had partners respective to whether they churned or not. And we can see senior citizens who churned, even though are less than who did not churn is still a big number comparatively. Hence, the company should focus on senior citizens. Customers with spouses and children might churn less to keep the services running for their family.

Next, I made pie carts of gender and no such insight was available through that.

Then the bar graphs of phone services against customers loyal or churned respectively. And as shown by the graph a significant number of customers with phone services churned and the company should look further into this matter.

After that pie charts of contract against clients and churned customers are shown respectively. Customers that have month-to-month contracts churn more than the customers who have long term contracts. The company should provide some incentives to the customers do that that keep on renewing their contract or shift to long term contract.

In the middle panel I made graphs of payment method by churn and internet service by churn respectively.

Payment method electronic check shows much higher churn rate than other payment methods. although electronic check brings the large number of customers but it also causes the large amount of customer churn among all other categories. Hence, organisation must introduce offers that attract the customers who are availing electronic check facility.

Customers with Internet Service fiber optic as part of their contract have much higher churn rate. company must focus on working on the weak areas of fiber optic service because large number of company's customers avail this service.

Churning customers have higher monthly charges and much lower interquartile range compared to that of non-churners.

Python Code:

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

In [2]:
```
#import the data
ChurnDatadf=pd.read_csv("churndata.csv")
```

In [3]:
```
#Show the features ( –columns) and the data in the dataframe
ChurnDatadf.head()
```

Out[3]:

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

In [4]:
```
#Get a summary on the dataframe including datatypes and shape.
ChurnDatadf.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
```

In [5]:
```
#Get info about numerical data
ChurnDatadf.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]:
```
#Gat unique values for every column
for i in ChurnDatadf.columns:
    print(f"Unique {i}'s count: {ChurnDatadf[i].nunique()}")
    print(f"{ChurnDatadf[i].unique()}\n")
```

```
Unique customerID's count: 7043
['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' ... '4801-JZAZL' '8361-LTMKD'
 '3186-AJIEK']

Unique gender's count: 2
['Female' 'Male']

Unique SeniorCitizen's count: 2
[0 1]

Unique Partner's count: 2
['Yes' 'No']

Unique Dependents's count: 2
['No' 'Yes']

Unique tenure's count: 73
[ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]

Unique PhoneService's count: 2
['No' 'Yes']

Unique MultipleLines's count: 3
['No phone service' 'No' 'Yes']

Unique InternetService's count: 3
['DSL' 'Fiber optic' 'No']
```

In [7]:
```
#Changing the data tyoe of "totalCharges" and 'tenure" to float.
ChurnDatadf['TotalCharges'] = pd.to_numeric(ChurnDatadf['TotalCharges'], errors='coerce')
def feature_to_float(feature_list, ChurnDatadf):
    for i in feature_list:
        ChurnDatadf[i] = ChurnDatadf[i].astype(float)
    return ChurnDatadf
feature_to_float(['tenure'], ChurnDatadf)
```

Out[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	Tech
0	7590-VHVEG	Female	0	Yes	No	1.0	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34.0	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2.0	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45.0	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2.0	Yes	No	Fiber optic	No	...	No	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24.0	Yes	Yes	DSL	Yes	...	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72.0	Yes	Yes	Fiber optic	No	...	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11.0	No	No phone service	DSL	Yes	...	No	
7041	8361-LTMKD	Male	1	Yes	No	4.0	Yes	Yes	Fiber optic	No	...	No	

In [8]:
```
#Renaming the data values of "PoymentMethod" for better readability.
payment_column = {'Electronic check': 'E-Check' , 'Mailed check': 'Mailed Check', 'Bank transfer (automatic)': 'Bank Transfer',
                  'Credit card (automatic)': 'Credit Card'}
ChurnDatadf["PaymentMethod"].replace(payment_column, inplace=True)
```

In [9]:
```
#Checking the data types for any unintended data types.
ChurnDatadf.dtypes
```

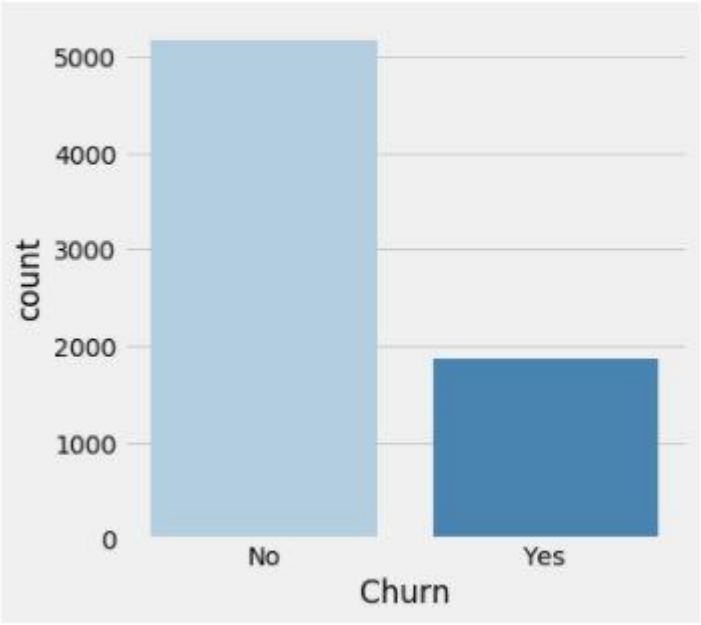
Out[9]:
```
customerID          object
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure             float64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges       float64
Churn               object
dtype: object
```

In [10]:
```
#Counting the number of missing votues.
ChurnDatadf.isna().sum()
```

Out[10]:
```
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
```

The above shows 11 missing values for "TotalCharges". The respective data entries will be deleted for simplicity.

In [11]:
```
#Apply the Fivethirtyeight style to all plots.
plt.style.use("fivethirtyeight")
#Display a frequency distribution for churn.
plt.figure(figsize=(5, 5))
ax = sns.countplot(x=ChurnDatadf['Churn'], palette="Blues", linewidth=1)
plt.show()
```



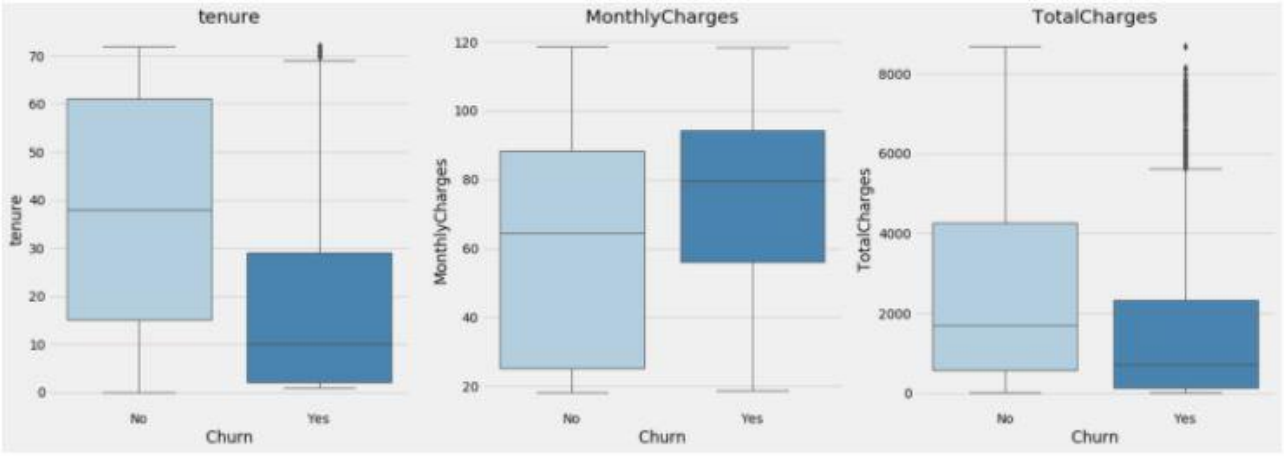
The plot shows a class imbalance of the data between churners and non-churners.

In [12]:
```
#Create a function to generate boxptots.
plots = {1 : [111], 2: [121, 122], 3: [131, 132, 133], 4: [221, 222, 223, 224], 5: [231, 232, 233, 234, 235],
         6: [231, 232, 233, 234, 235, 236]}
def boxplot(x, y, ChurnDatadf):
    rows = int(str(plots[len(y)][0])[0])
    columns = int(str(plots[len(y)][0])[1])
    plt.figure(figsize=(7*columns, 7*rows))

    for i, j in enumerate(y):
        plt.subplot(plots[len(y)][i])
        ax = sns.boxplot(x=x, y=j, data=ChurnDatadf[[x, j]], palette="Blues", linewidth=1)
        ax.set_title(j)

    return plt.show()
```

n [13]:
```
#Generate boxptots for tenure, MonthLyChorges and TotalCharges.
boxplot("Churn", ["tenure", "MonthlyCharges", "TotalCharges"], ChurnDatadf)
```



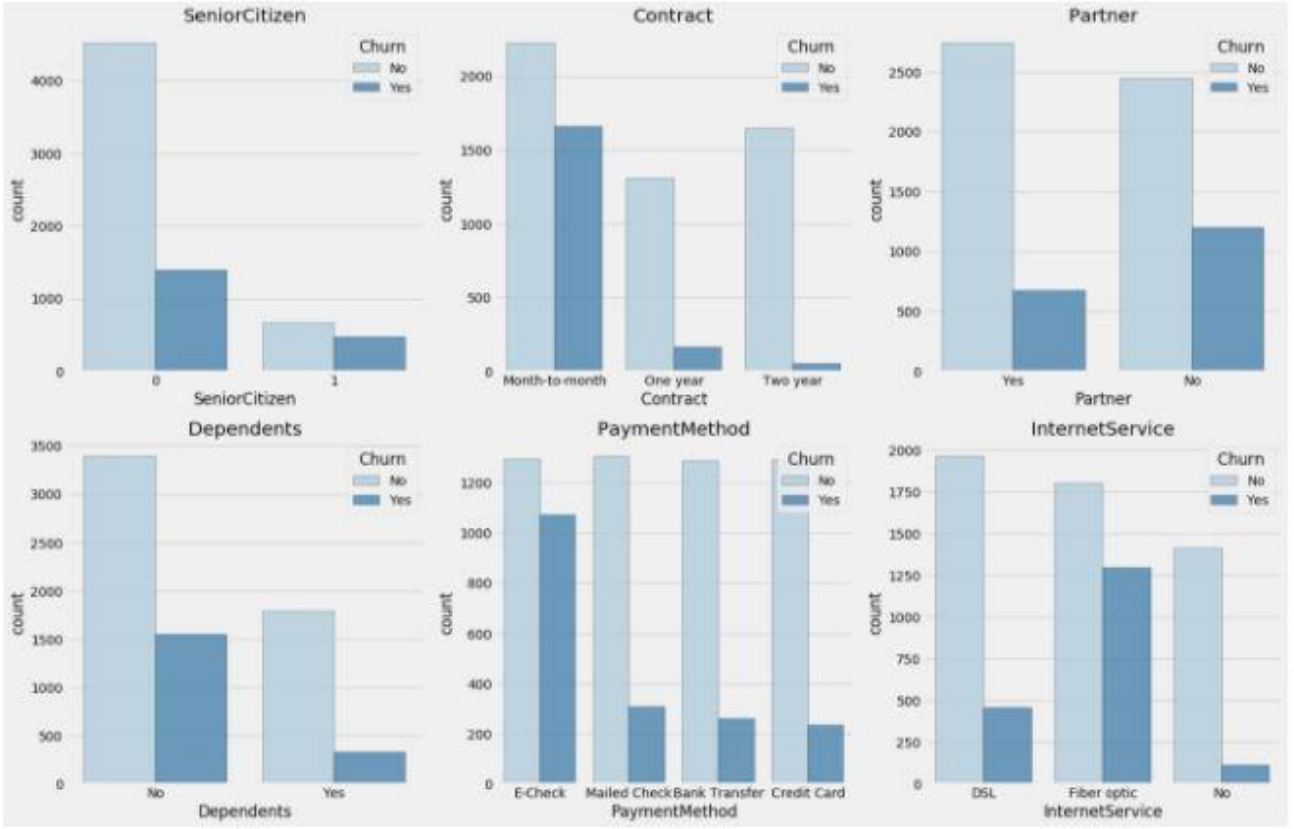
The above box plots show that Churning customers have much lower tenure with a median of 10 months compared to a median of non-churners of 38 months. Churning customers have higher monthly charges with a median of 80 USD and much lower interquartile range compared to that of non-churners (median of 65 USD).

In [14]:
```
#Create a function to generate countpLots:
def countplot(x, y, ChurnDatadf):
    rows = int(str(plots[len(y)][0])[0])
    columns = int(str(plots[len(y)][0])[1])
    plt.figure(figsize=(7*columns, 7*rows))

    for i, j in enumerate(y):
        plt.subplot(plots[len(y)][i])
        ax = sns.countplot(x=j, hue=x, data=ChurnDatadf, palette='Blues', alpha=0.8, linewidth=0.4, edgecolor='black')
        ax.set_title(j)

    return plt.show()
```

In [15]:
```
#Generate countpLots for various features.
countplot("Churn", ['SeniorCitizen', 'Contract', 'Partner', 'Dependents', 'PaymentMethod', 'InternetService'], ChurnDatadf)
```



The above plots indicate that: Senior citizens churn rate is much higher than non-senior churn rate. Churn rate for month-to-month contracts much higher that for other contract durations. Moderately higher churn rate for customers without partners. Much higher churn rate for customers without children. Payment method electronic check shows much higher churn rate than other payment methods. Customers with InternetService fiber optic as part of their contract have much higher churn rate.

In [16]:
```
#Check of outliers by applying the IQR method checking if values are way outside the IQR borders.
numerical_features = ["tenure", "MonthlyCharges", "TotalCharges"]
ChurnDatadf_num = ChurnDatadf[numerical_features]
ChurnDatadf_num.describe()
Q1 = ChurnDatadf_num.quantile(0.25)
Q3 = ChurnDatadf_num.quantile(0.75)
IQR = Q3 - Q1
IQR
((ChurnDatadf_num < (Q1 - 1.5 * IQR)) | (ChurnDatadf_num > (Q3 + 1.5 * IQR))).any()
```

Out[16]:
```
tenure            False
MonthlyCharges    False
TotalCharges      False
dtype: bool
```

No outliers in numerical features detected with the IQR method

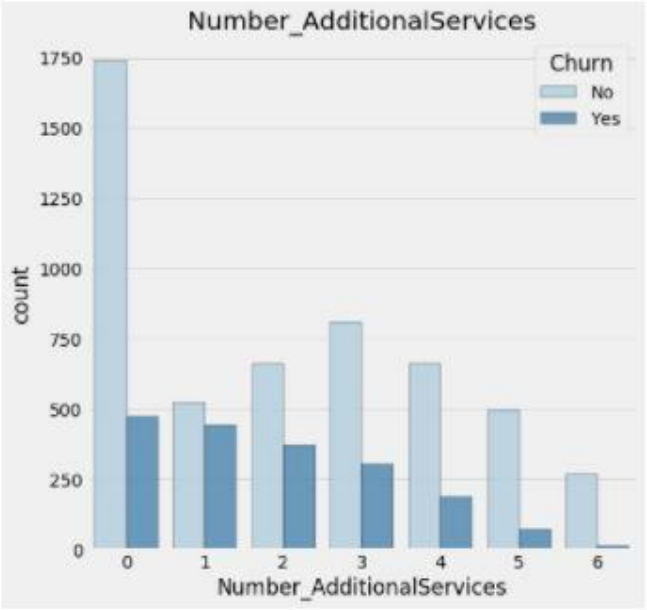
In [17]:
```
#Drop the rows with missing values.
ChurnDatadf = ChurnDatadf.dropna()
```

In [18]:
```
#Drop customerID feature.
ChurnDatadf = ChurnDatadf.drop(columns='customerID')
```

In [19]:
```
#Generate new feature "Number_AdditionalServices" by summing up the number of add-on services consumed.
ChurnDatadf['Number_AdditionalServices'] = (ChurnDatadf[['OnlineSecurity', 'DeviceProtection', 'StreamingMovies', 'TechSupport',
                                            'StreamingTV', 'OnlineBackup']] == 'Yes').sum(axis=1)
```

In [20]:
```
#Generate countplot for the new feature.
countplot('Churn', ['Number_AdditionalServices'], ChurnDatadf)
```

In [20]:
```
#Generate countplot for the new feature.
countplot('Churn', ['Number_AdditionalServices'], ChurnDatadf)
```



Plot analysis: The countplot shows a very high churn rate for customers that have 1 additional service. Customers with a very high number of additional services do have a low churn rate.

]:
```
ChurnDatadf.to_excel("ChurnData.xlsx")
```

In [21]:
```
#Label encoding for identified columns.
features_le = ['gender', 'Partner', 'Dependents', 'Churn', 'PhoneService', 'PaperlessBilling']
def label_encoding(features, ChurnDatadf):
    for i in features:
        ChurnDatadf[i] = ChurnDatadf[i].map({'Yes': 1, 'No': 0})
    return
label_encoding(['Partner', 'Dependents', 'Churn', 'PhoneService', 'PaperlessBilling'], ChurnDatadf)
ChurnDatadf['gender'] = ChurnDatadf['gender'].map({'Female': 1, 'Male': 0})
```

Label encoding: The following features are categorical and each take on 2 values (mostly yes/no) — therefore are transformed to binary integers: gender, Partner, Dependents, Churn, PhoneService, PaperlessBilling.

In [22]:
```
#One-Hot-Encoding for identified columns.
features_ohe = ['MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
                'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod', 'Number_AdditionalServices']
ChurnDatadf = pd.get_dummies(ChurnDatadf, columns=features_ohe)
```

One-Hot Encoding: The following features are categorical, yet not ordinal (no ranking) but take on more than 2 values. For each value, a new variable is created with a binary integer indicating if the value occured in a data entry or not (1 or 0): MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaymentMethod.

In [23]:
```
#Min-Max-Scaling for identified columns.
from sklearn.preprocessing import MinMaxScaler
features_mms = ['tenure', 'MonthlyCharges', 'TotalCharges']
ChurnDatadf_features_mms = pd.DataFrame(ChurnDatadf, columns=features_mms)
ChurnDatadf_remaining_features = ChurnDatadf.drop(columns=features_mms)

mms = MinMaxScaler()
rescaled_features = mms.fit_transform(ChurnDatadf_features_mms)
ChurnDatadf_rescaled_features = pd.DataFrame(rescaled_features, columns=features_mms, index=ChurnDatadf_remaining_features.index

ChurnDatadf = pd.concat([ChurnDatadf_remaining_features, ChurnDatadf_rescaled_features], axis=1)
```

In [24]:
```
#Show correlation plot for correlation of Churn with each of the remaining features.
plt.figure(figsize=(16,10))
ChurnDatadf.corr()['Churn'].sort_values(ascending=False).plot(kind='bar', figsize=(20,5))
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20071c76c48>
```