# Telco Customer Churn Prediction

## 1.1 Introduction

Telco receives a lot of customers who subscribe to their service to get access to the fastest possible communication access through mobile and internet services. There are diverse set of applications which Telco gives to their users such as mobile services and communication tools to name a few.

One of the challenges that the company faces is to get to know beforehand whether a customer who has activated a service under Telco is going to leave or stay in the service (churn). If they know that a customer is going to leave the service based on a set of factors such as Gender and whether they are Senior citizen or not, they can come up with affordable plans or give promotional offers so that they retain the customer without them having to move to options from other companies.

## 1.2 Machine Learning and Data Science

There are a lot of technologies and tools which are build with the aid of machine learning and data science. Considering that the data size is large and has useful features, it is possible to gain insights from the data and make predictions. After performing sufficient training and hyperparameter tuning, it is possible to get the best predictions for our models.

We try to combat the challenge by Telco with the aid of data science and machine learning. We take the output variable (Customer Churn) and we try to build the models for prediction with diverse set of features respectively.

## 1.3 Metrics

Since we are working on a classification problem, we need to ensure we select the metrics that are useful for these problems. Below are the metrics that we are going to be using for our problem.

1. Log Loss
2. Accuracy
3. Precision
4. Recall
5. F1-score

## 1.4 Source

The data was downloaded from Kaggle - a website that hosts data science and machine learning challenges from companies. Below is the link for the dataset along with the definition of various columns used in the data. Feel free to take a look.

https://www.kaggle.com/blastchar/telco-customer-churn

# Table of Contents



---

## 1.5 Importing the libraries

It is now time to read the libraries that are important for our machine learning problem. There are libraries such as numpy that would ensure that we get to perform computation with arrays.

In addition, we import seaborn which is used for data visualization and plotting respectively.

from sklearn, we import preprocessing library which contains StandardScaler we are going to be using for transformed our data respectively.

Matplotlib is similar to seaborn for plotting. It is also sometimes convenient to use this library for plotting rather than solely relying on Seaborn.

Pandas library is used to work with the dataframe and reading the values present in them.

In [24]:
```
import numpy as np
import seaborn as sns
import sklearn
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pandas as pd
import random
import warnings
warnings.filterwarnings("ignore")
```

In [25]:
```
df = pd.read_csv("Telco Customer Churn.csv")
```

## 1.6 Reading the first 5 rows

Let us explore the first 5 rows of our dataframe to get to understand the columns that we are going to be working in the data respectively.

In [26]:
```
df.head()
```

Out[26]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

Using 'describe' ensures that we are getting the values for the numerical columns such as the mean, standard deviation, minimum value and 1st, 2nd, 3rd and Maximum values.

In [27]:
```
df.describe().T
```

Out[27]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **SeniorCitizen** | 7043.0 | 0.162147 | 0.368612 | 0.00 | 0.0 | 0.00 | 0.00 | 1.00 |
| **tenure** | 7043.0 | 32.371149 | 24.559481 | 0.00 | 9.0 | 29.00 | 55.00 | 72.00 |
| **MonthlyCharges** | 7043.0 | 64.761692 | 30.090047 | 18.25 | 35.5 | 70.35 | 89.85 | 118.75 |

## 2.1 Missingno

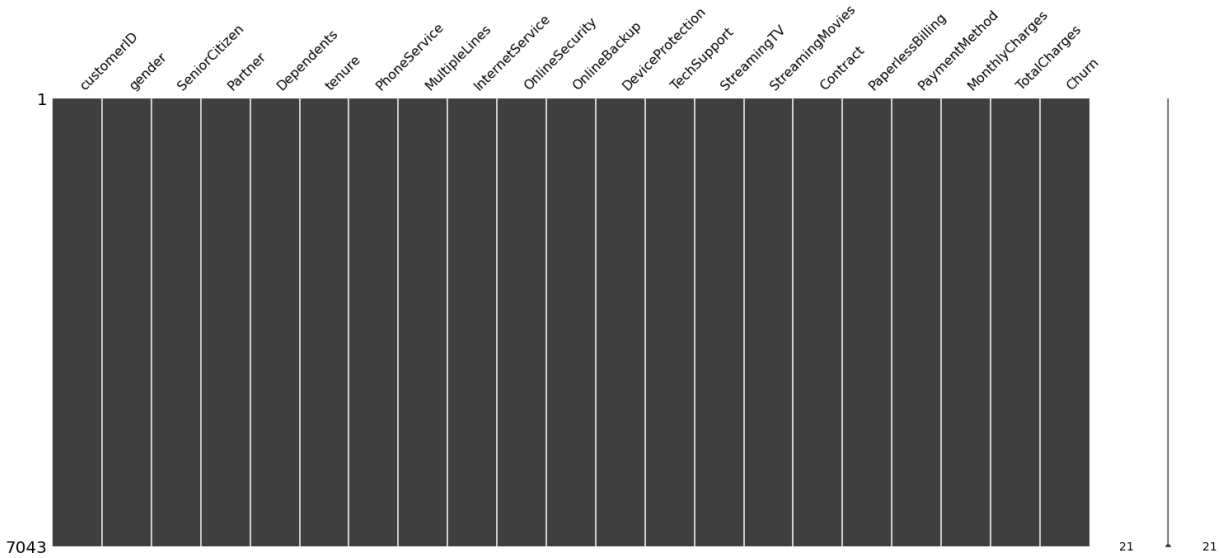
Missingno is a useful library to plot the missing values in our dataframe. If there are any missing values in our data, we get an output for that particular column with white stripes which indicates the presence of missing value.

Let us see from our data if there are any missing values present in them.

In [28]:
```
import missingno as msno
```

In [29]:
```
msno.matrix(df)
```

Out[29]:
```
<AxesSubplot:>
```




## Observation:

We see that there are no missing values in our columsn as presented by the missingno plot. Therefore, we can start processing the data and understand it.

In [30]:
```
print("The shape of the dataframe is: {}".format(df.shape))
```

```
The shape of the dataframe is: (7043, 21)
```

## Observation:

We are currently working with about 7043 customers with many attributes or features such as their gender and whether they are a senior citizen or not. There are many other features that we have considered that makes this problem interesting.

In [31]:
```
df.columns
```

Out[31]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

The above lists the columns that we are going to be working in our dataset.

In [32]:
```
df.head()
```

Out[32]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [33]:
```
df['gender'].unique()
```

Out[33]:
```
array(['Female', 'Male'], dtype=object)
```

In [34]:
```
df['SeniorCitizen'].unique()
```

Out[34]:
```
array([0, 1], dtype=int64)
```

In [35]:
```
df['Partner'].unique()
```

Out[35]:
```
array(['Yes', 'No'], dtype=object)
```

In [36]:
```
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 [37]:
```
df.head()
```

Out[37]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [38]:

```
print("We are ensuring that there are no duplicate customers in our data")
print("The total number of unique customers in the data: {}".format(    len(df['cust
```

```
We are ensuring that there are no duplicate customers in our data
The total number of unique customers in the data: 7043
```

In [39]:

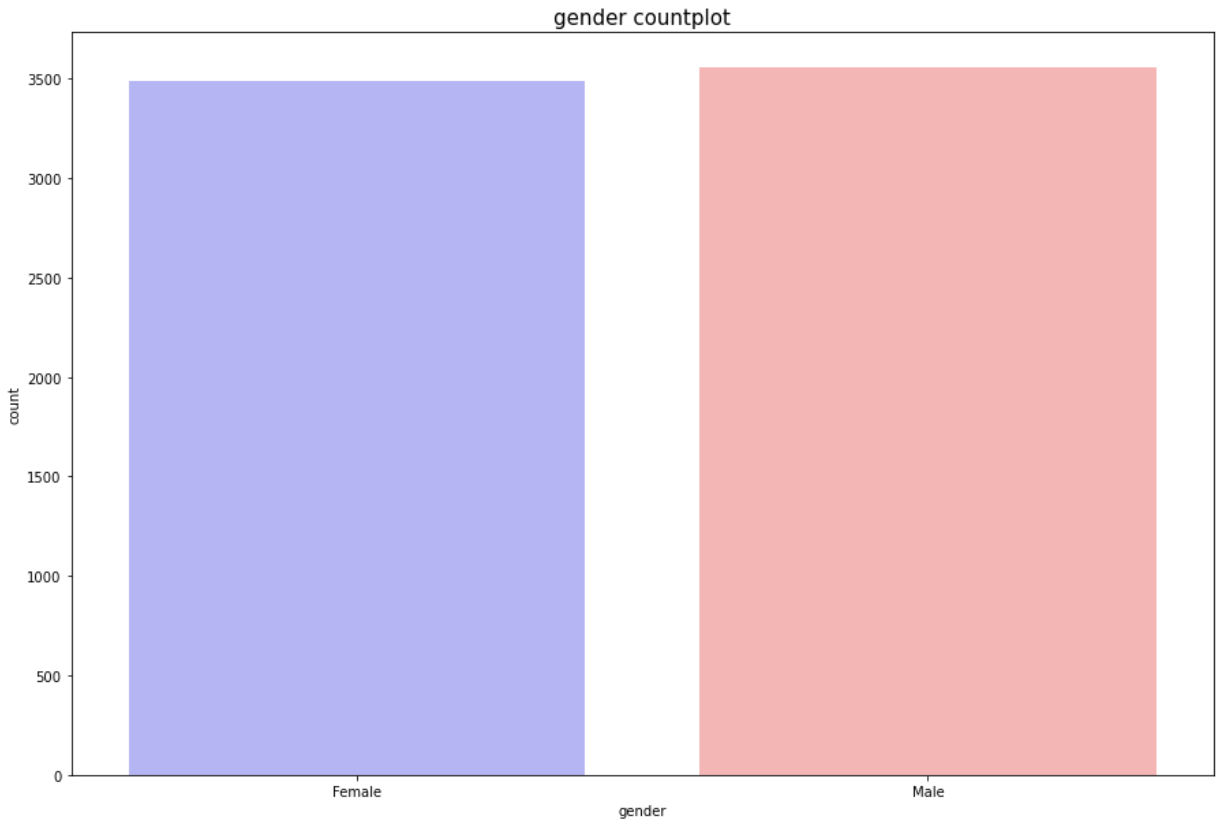
```
## Palette values
## Thanks to https://medium.com/@morganjonesartist/color-guide-to-seaborn-palettes-d
## We were able to use all the palettes that were mentioned in the blog

palette_values = ['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r', 'BuGn'
 'CMRmap', 'CMRmap_r', 'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens', 'Greens_r', '
 'OrRd_r', 'Oranges', 'Oranges_r', 'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1'
 'Pastel1_r', 'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn', 'PuBuGn_r'
 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r', 'Purples', 'Purples_r', 'RdBu', 'RdBu
'RdGy', 'RdGy_r', 'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r', 'Red
 'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3', 'Set3_r', 'Spectral', 'Spectr
 'Wistia','Wistia_r','YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r', 'YlOrBr', 'YlOrBr_r', '
 'YlOrRd_r', 'afmhot', 'afmhot_r', 'autumn', 'autumn_r', 'binary', 'binary_r', 'bone
 'bone_r', 'brg', 'brg_r', 'bwr', 'bwr_r', 'cividis', 'cividis_r', 'cool', 'cool_r',
 'cubehelix', 'cubehelix_r', 'flag', 'flag_r', 'gist_earth', 'gist_earth_r', 'gist_g
 'gist_rainbow', 'gist_rainbow_r', 'gist_stern', 'gist_stern_r', 'gist_yarg',
 'gist_yarg_r', 'gnuplot', 'gnuplot2', 'gnuplot2_r', 'gnuplot_r', 'gray', 'gray_r',
 'hot', 'hot_r', 'hsv', 'hsv_r', 'icefire', 'icefire_r', 'inferno',
 'inferno_r', 'magma', 'magma_r', 'mako', 'mako_r',
 'nipy_spectral', 'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
 'plasma', 'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r',
 'rocket', 'rocket_r', 'seismic', 'seismic_r', 'spring', 'spring_r',
 'summer', 'summer_r', 'tab10', 'tab10_r', 'tab20', 'tab20_r', 'tab20b',
 'tab20b_r', 'tab20c', 'tab20c_r', 'terrain', 'terrain_r', 'twilight',
 'twilight_r', 'twilight_shifted', 'twilight_shifted_r', 'viridis', 'viridis_r', 'vl
```

In [40]:

```
def countplot_function(dataframe, column, figsize = (15, 10), palette = "viridis"):
    plt.figure(figsize = figsize)
    sns.countplot(dataframe[column], palette = palette)
    plt.title("{} countplot".format(column), fontsize = 15)
    plt.xlabel("{}".format(column), fontsize = 10)
    plt.show()
```

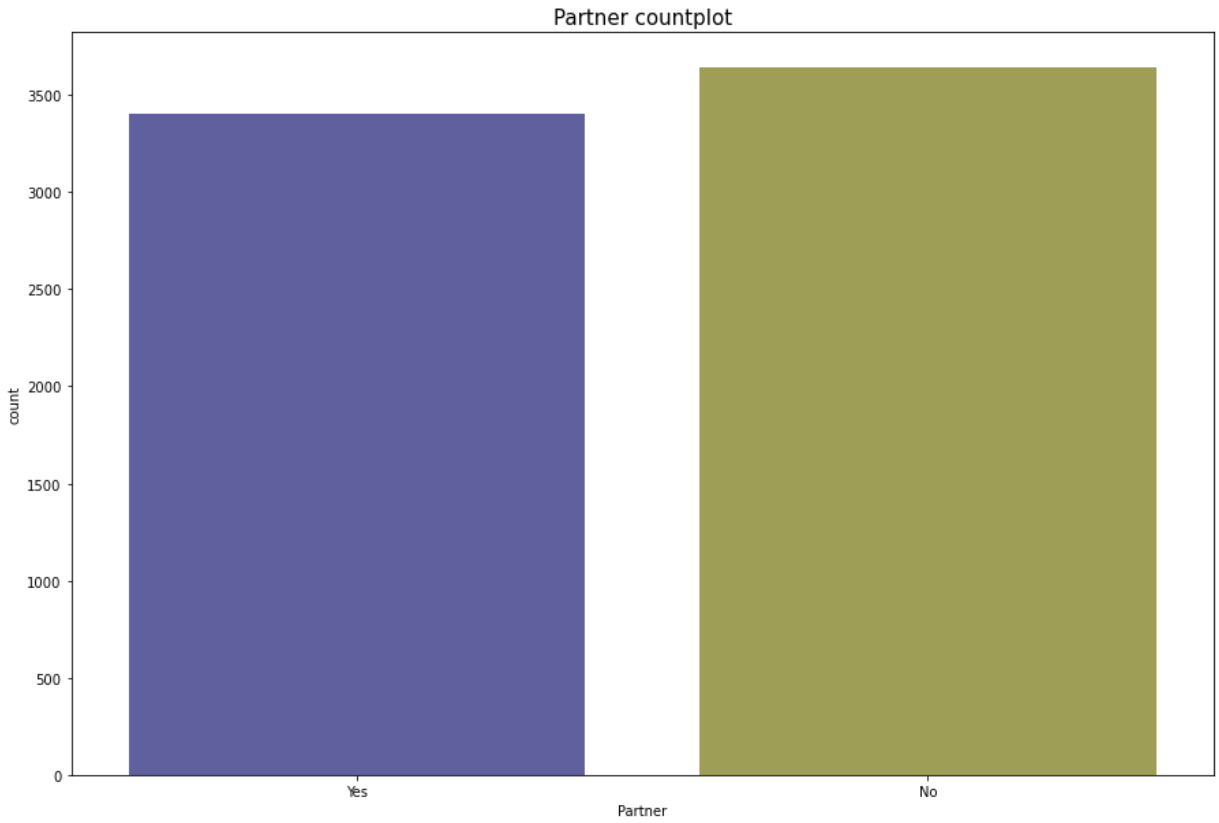
In [41]:
```
countplot_function(dataframe = df, column = 'gender', palette = random.choice(palett
```




# Observation:

1. We see that there are equal number of male and female in our data.
2. Therefore, the 2 groups are represented equally to understand their overall behavior towards Telco.

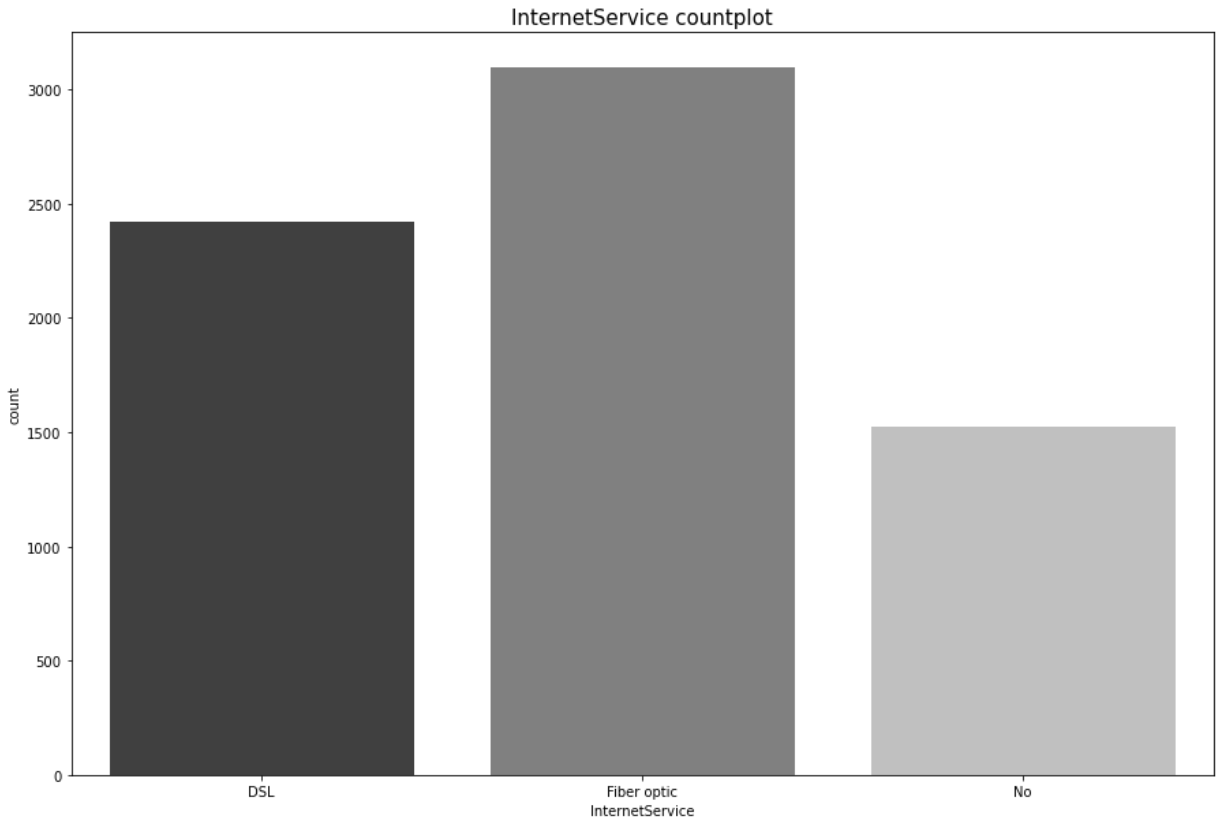
In [42]:
```
countplot_function(dataframe = df, column = "Partner", palette = random.choice(palet
```




# Observation:

1. There are more number of people who do not have partners compared to the ones who have partners.
2. Since we have more data for the people who do not have partners, we should be able to predict their behavior and determine well whether they would leave the telco service or not.

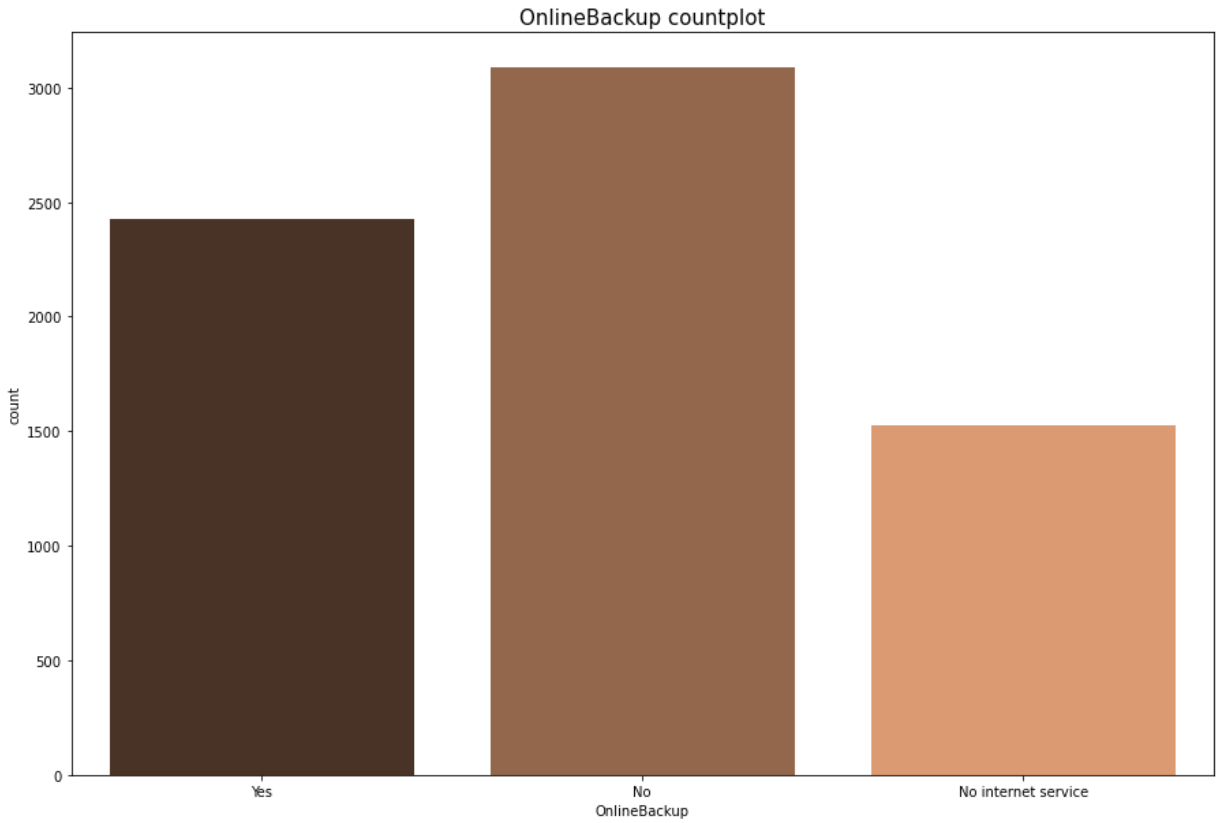
In [43]:
```
countplot_function(dataframe = df, column = "InternetService", palette = random.choi
```




# Observation:

1. It could be seen that most of the Telco services are Fiber Optic service as this option is becoming common among customers.
2. The data does a good job of reflecting the latest trends as most people opt fiber optic services compared to DSL services.

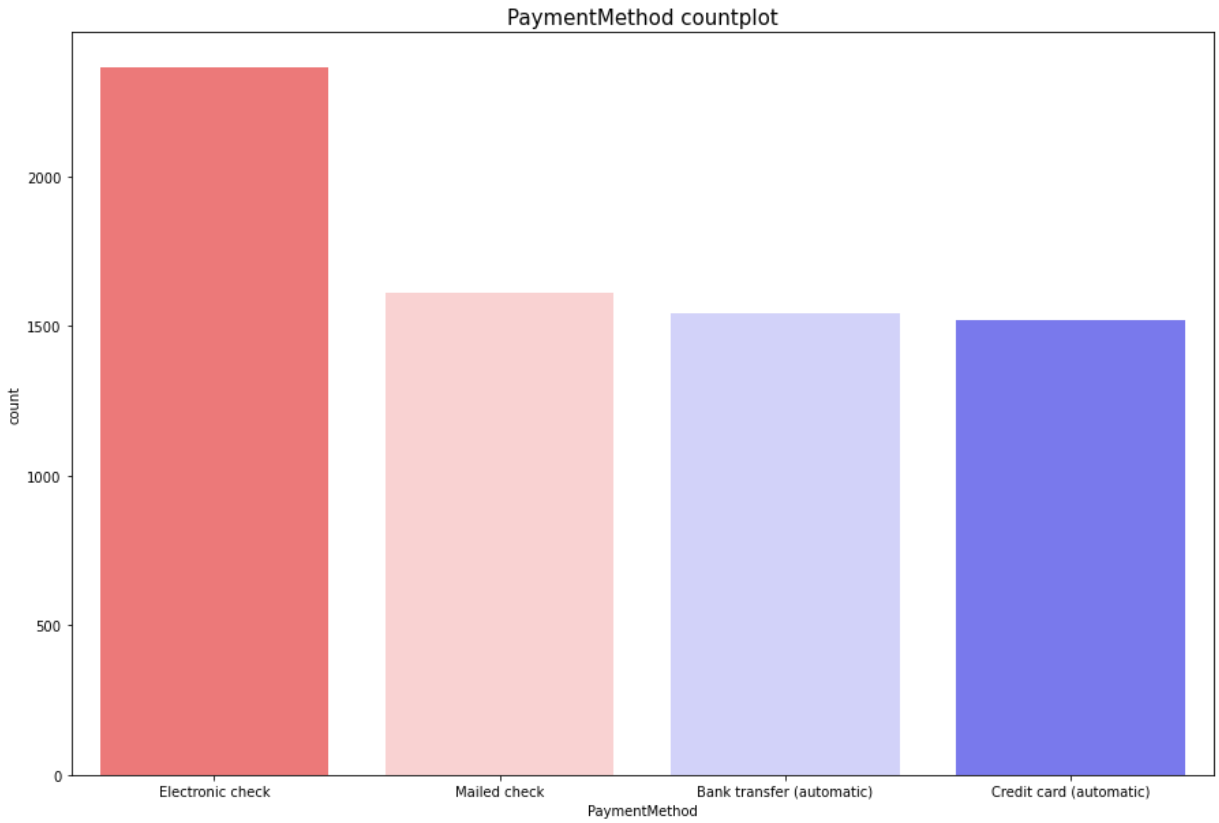
In [44]:
```
countplot_function(dataframe = df, column = "OnlineBackup", palette = random.choice(
```




# Observation:

1. Most people in our data do not prefer Online Backup Services as clearly shown in the above plot.
2. There are quite a number of people who do not opt the internet service as well.
3. Therefore, we should also consider whether a person has an internet service or not before determining whether they would prefer online backup.

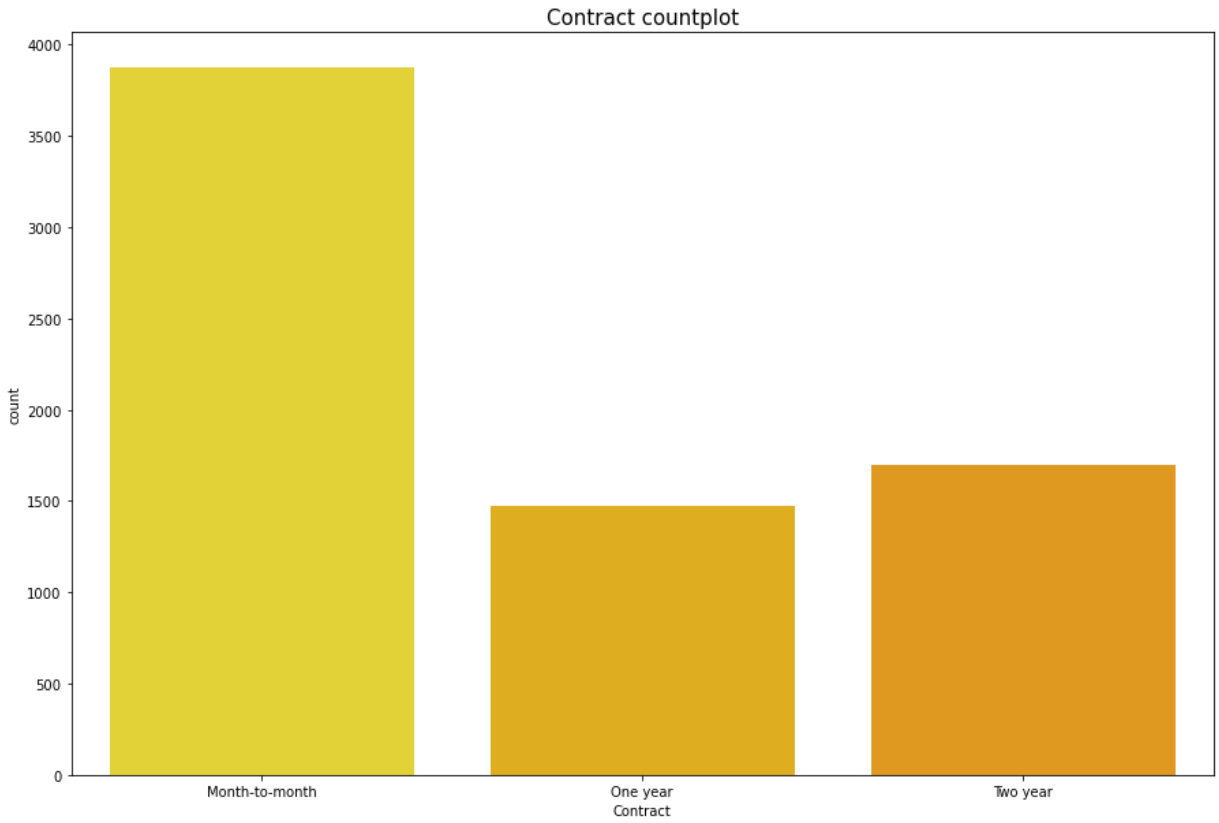
In [45]:
```
countplot_function(dataframe = df, column = "PaymentMethod", palette = random.choice
```




# Observation:

1. A large portion of users from Telco prefer Electronic Check compared to other options.
2. There are other options which are popular among the customers as well such as Mailed check, Bank transfer (automatic) and Credit card (automatic) respectively.
3. As a result, we should be able to more accurately predict the behavior of the customers who use the payment method of electronic check compared to other methods.

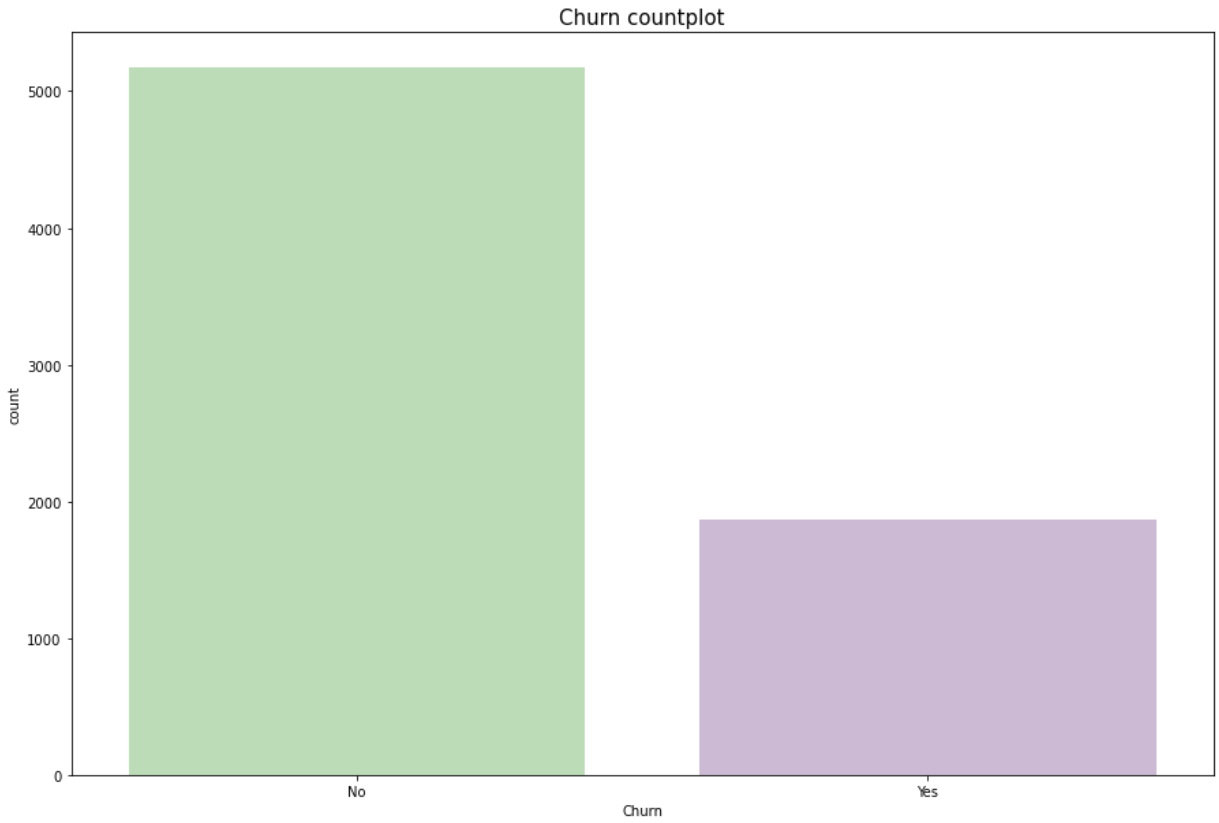
In [46]:
```
countplot_function(dataframe = df, column = "Contract", palette = random.choice(pale
```




# Observation:

1. We have a significantly large portion of the data where the contract is month-to-month compared to either one-year and two-year contracts.
2. This is true in real-life as well because most of the customers prefer to stick with month-to-month contract rather than staying with the same service for a long period of time.
3. Hence, we see that this data is quite reflective of the real-world.

In [47]:
```
countplot_function(dataframe = df, column = "Churn", palette = random.choice(palette
```




# Observation:

1. It is now important to observe the total number of customers who have churned (left the service) after a particular span of time.
2. We see that a large portion of our customers did not leave the service.
3. We also see that there are a few customers who left the service. We see that there is no overwhelming difference between the count of the customers who stayed in the service vs customers who have left or churned.
4. Hence, we can proceed with the data without adding additional customers who decided to churn.

In [50]:
```
df.head()
```

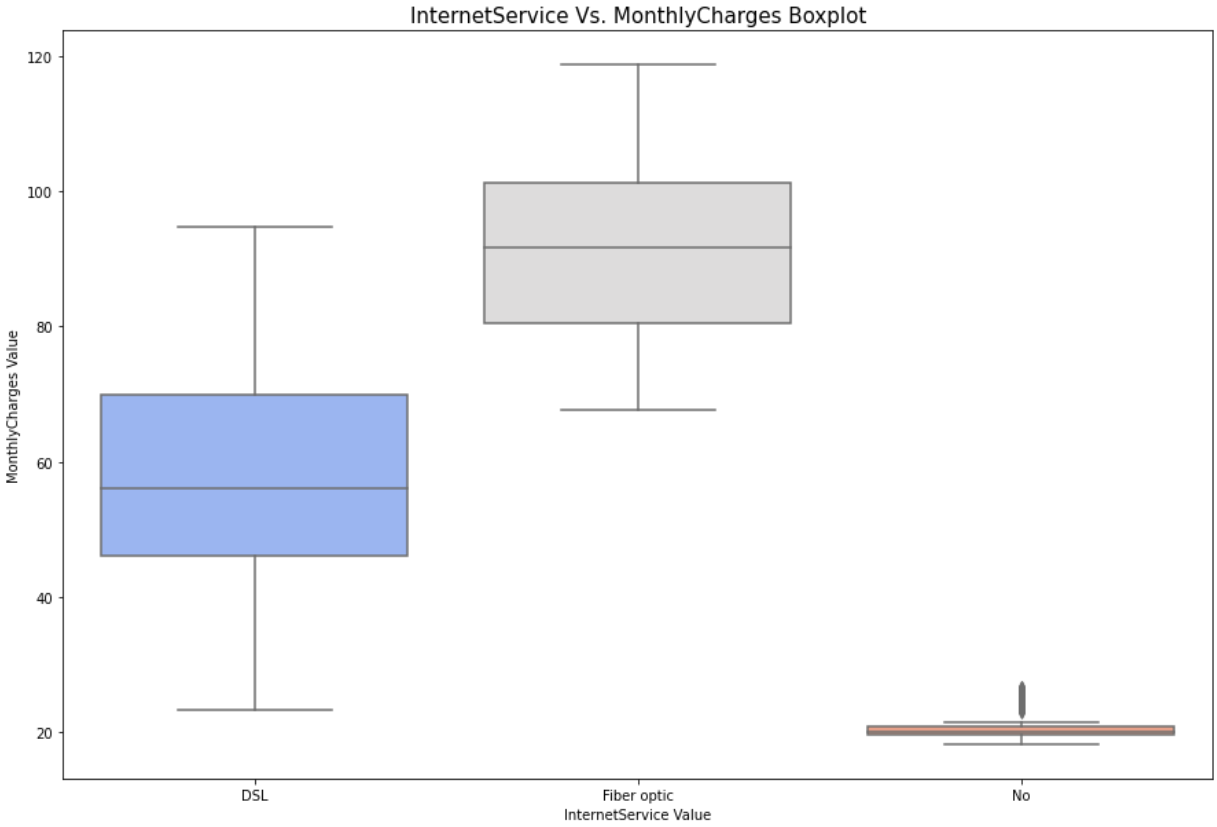
Out[50]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [51]:
```
def boxplot_function(dataframe, x_value, y_value, title_size = 15, label_size = 10,
    plt.figure(figsize = (figsize))
    sns.boxplot(x = x_value, y = y_value, data = dataframe, palette = palette)
    plt.xlabel("{} Value".format(x_value), fontsize = label_size)
    plt.ylabel("{} Value".format(y_value), fontsize = label_size)
    plt.title("{} Vs. {} Boxplot".format(x_value, y_value), fontsize = title_size)
    plt.show()
```

In [52]:
```
boxplot_function(dataframe = df, x_value = "InternetService", y_value = "MonthlyChar
```




# Observation:

1. It could easily be seen from the box plots that the people who opted for 'Fiber optic' service have higher monthly charges.
2. People who opted for 'DSL' service has signifantly lower monthly charges as shown above.
3. As expecte, customers who do not enroll in the internet service have low charges as shown.

In [53]:
```
df.head()
```

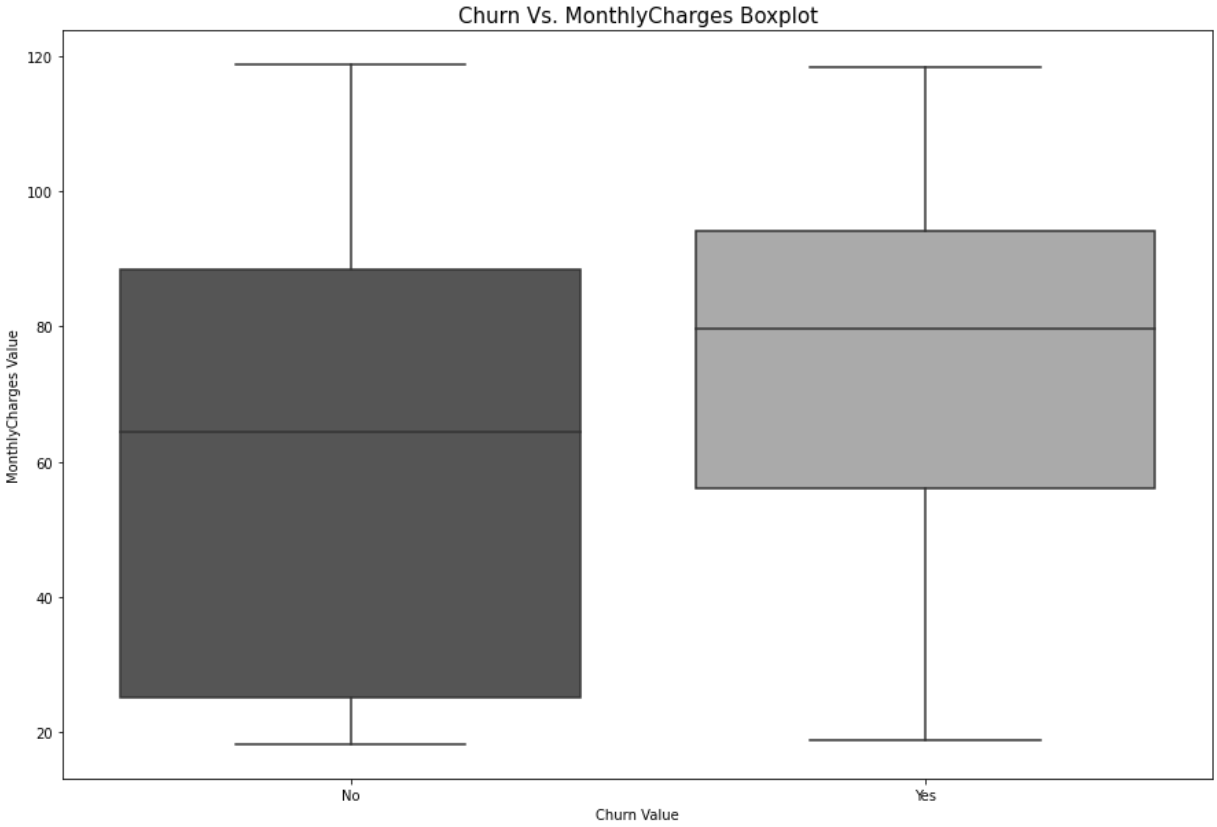
Out[53]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [54]:
```
boxplot_function(dataframe = df, x_value = "Churn", y_value = "MonthlyCharges", pale
```



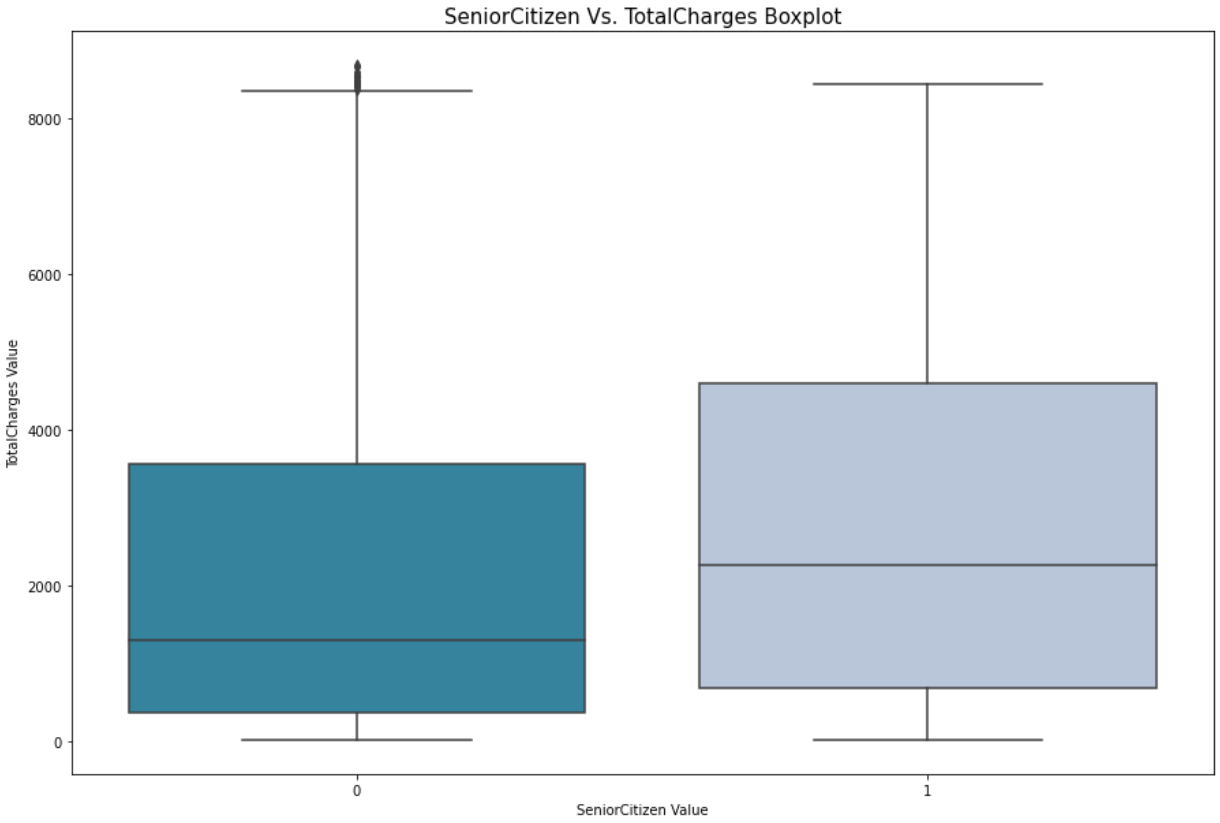

# Observation:

1. It could be seen as the monthly charges are increasing, there is a higher possibility that the customers are inclined to leave the service.
2. People who stay in the service usually have low monthly charges.
3. Therefore, Telco company could take action based on the plots and reduce the prices of various services as this would ensure that most customers are inclined to stay in the service.

In [55]:
```
# https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html
# Used the above link to replace the missing value in 'Total Charges' to '0' so that
# it would be easy to convert the values respectively.
## Replacing the missing value with Medin value from the Total Charges.
## This is because Median is robust to outliers.
```

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'].replace(' ', '1394.55'))
```

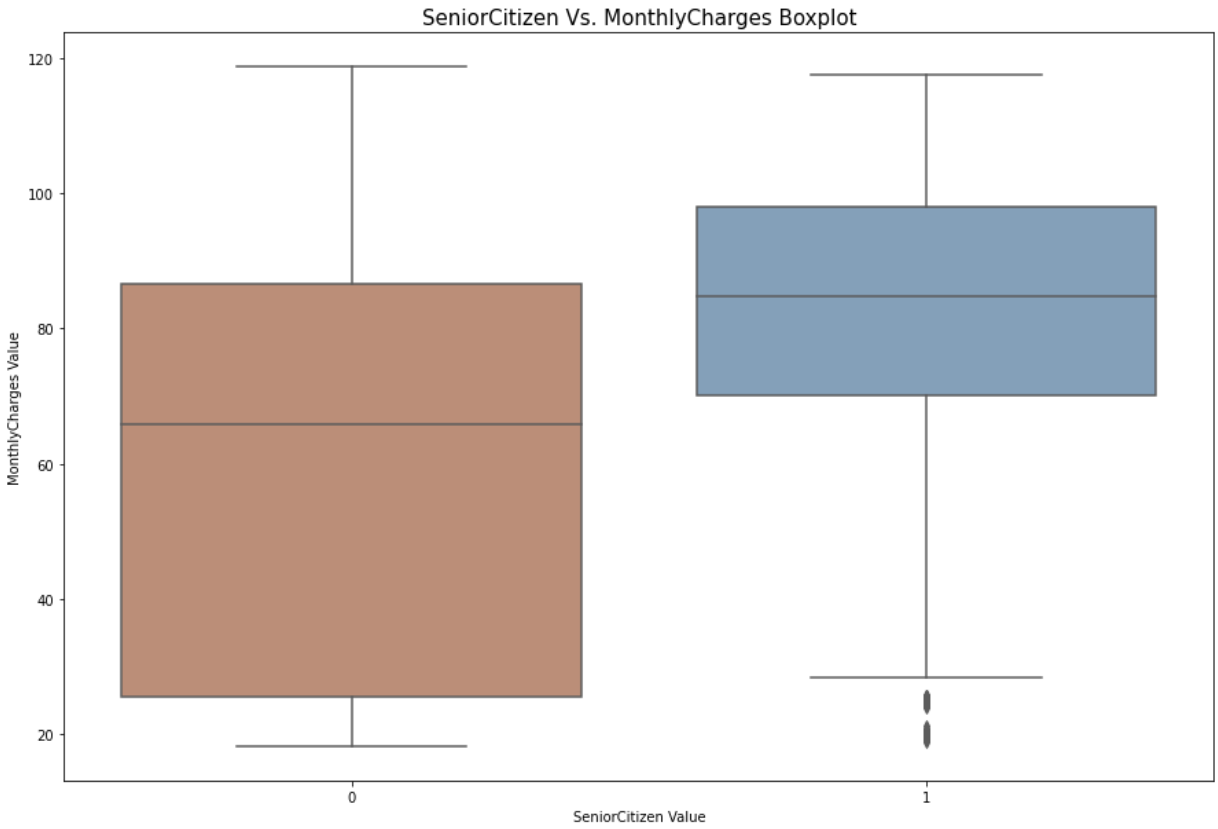
In [56]:
```
boxplot_function(dataframe = df, x_value = "SeniorCitizen", y_value = "TotalCharges"
```




# Observation:

1. Based on the above boxplot, it could be seen that whether a person is a senior citizen or not has an impact of the total charges.
2. Senior citizens usually are quite rich and they usually work which means that they have higher income.
3. As a result, they might be opting for more services from Telco leading to higher total charges.

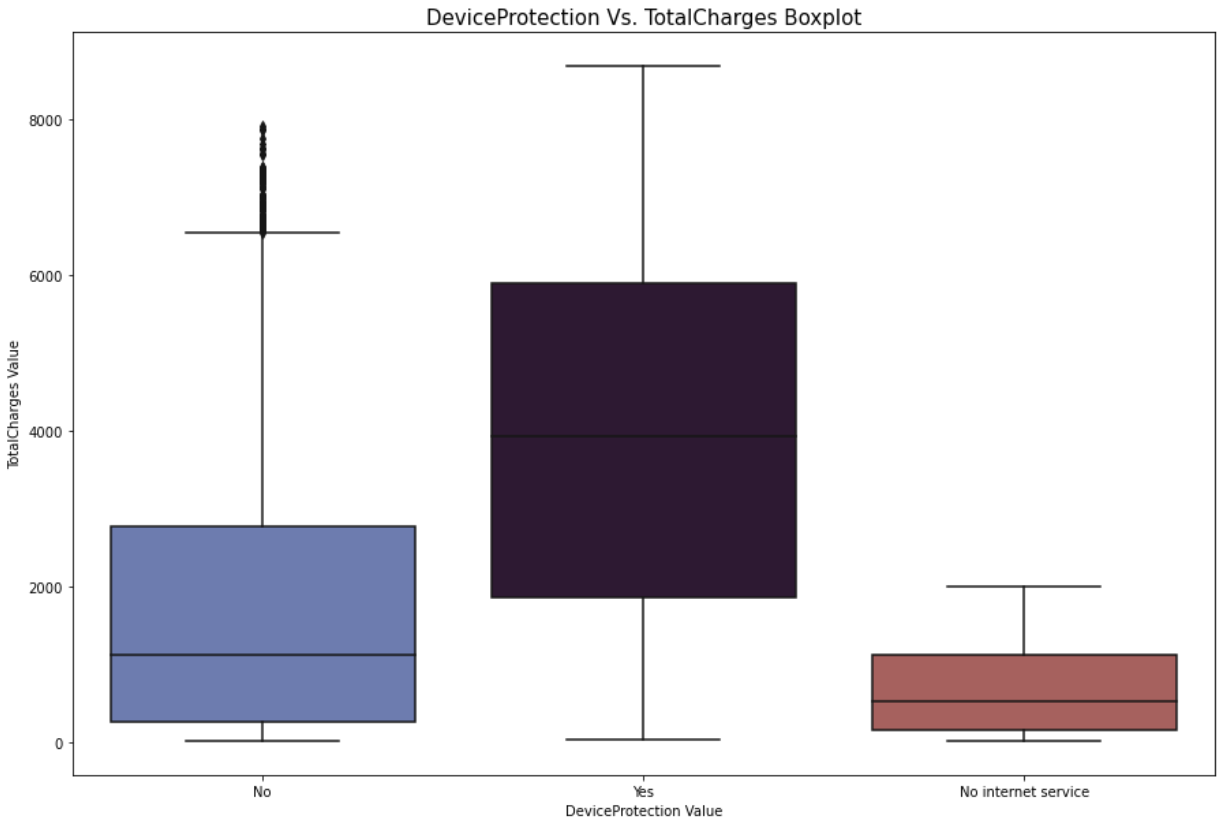
In [57]:
```
boxplot_function(dataframe = df, x_value = 'SeniorCitizen', y_value = 'MonthlyCharge
```




# Observation:

1. Monthly charges are significantly higher for Senior Citizens compared to Non-Senior Citizens respectively.
2. As a result, this leads us to believe that senior citizens are more inclined to add more services from Telco.
3. Therefore, Telco could take action and provide more interesting services to senior citizens compared to non-senior citizens.

In [58]:
```
boxplot_function(dataframe = df, x_value = "DeviceProtection", y_value = "TotalCharg
```

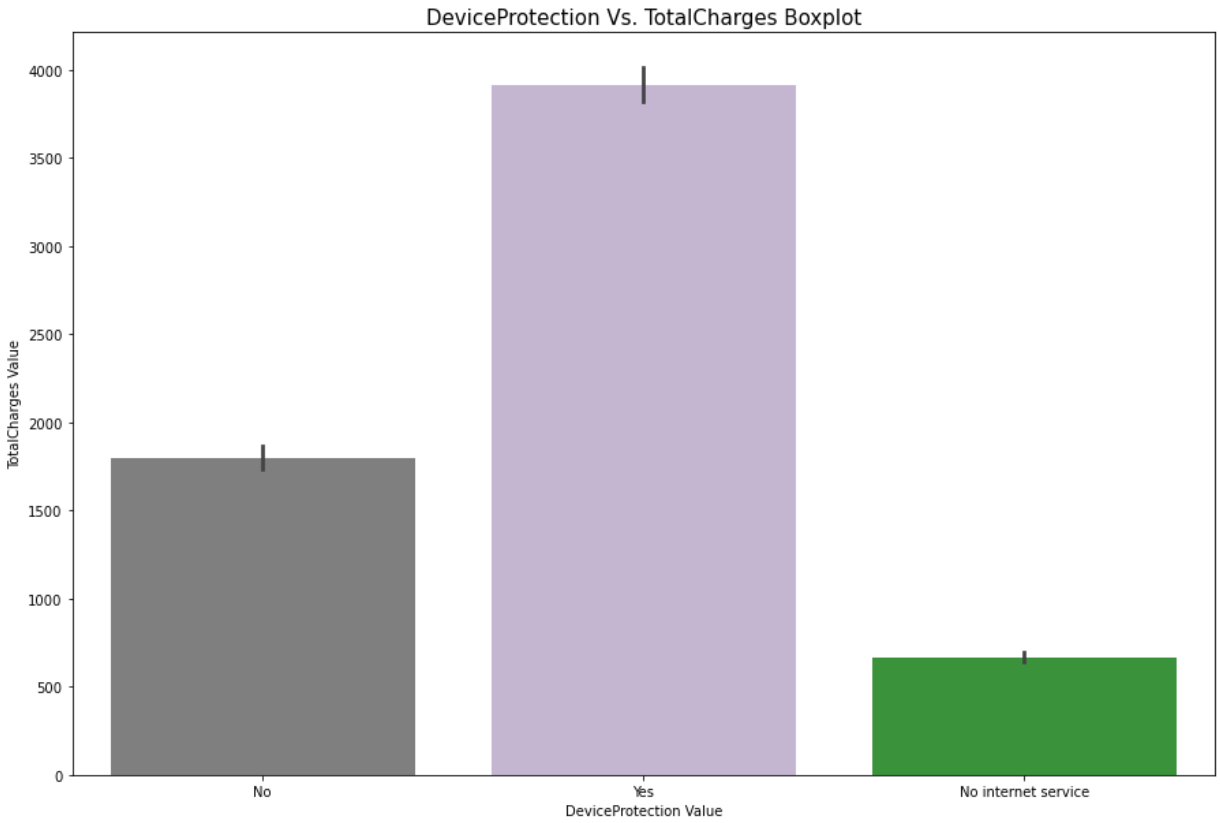



# Observation:

1. Device Protection Plans have a very high cost as could be seen from the above.
2. This means that people are paying a lot for Device Protection plans.
3. We have seen from the previous plots that higher the charges, the more inclined are the customers to leave the Telco service.
4. Hence, Telco could take steps to reduce the prices for the Device Protection plans.

In [59]:
```
def barplot_function(dataframe, x_value, y_value, hue = None, title_size = 15, label
    plt.figure(figsize = (figsize))
    sns.barplot(x = x_value, y = y_value, data = dataframe, hue = hue, palette = pal
    plt.xlabel("{} Value".format(x_value), fontsize = label_size)
    plt.ylabel("{} Value".format(y_value), fontsize = label_size)
    plt.title("{} Vs. {} Boxplot".format(x_value, y_value), fontsize = title_size)
    plt.show()
```

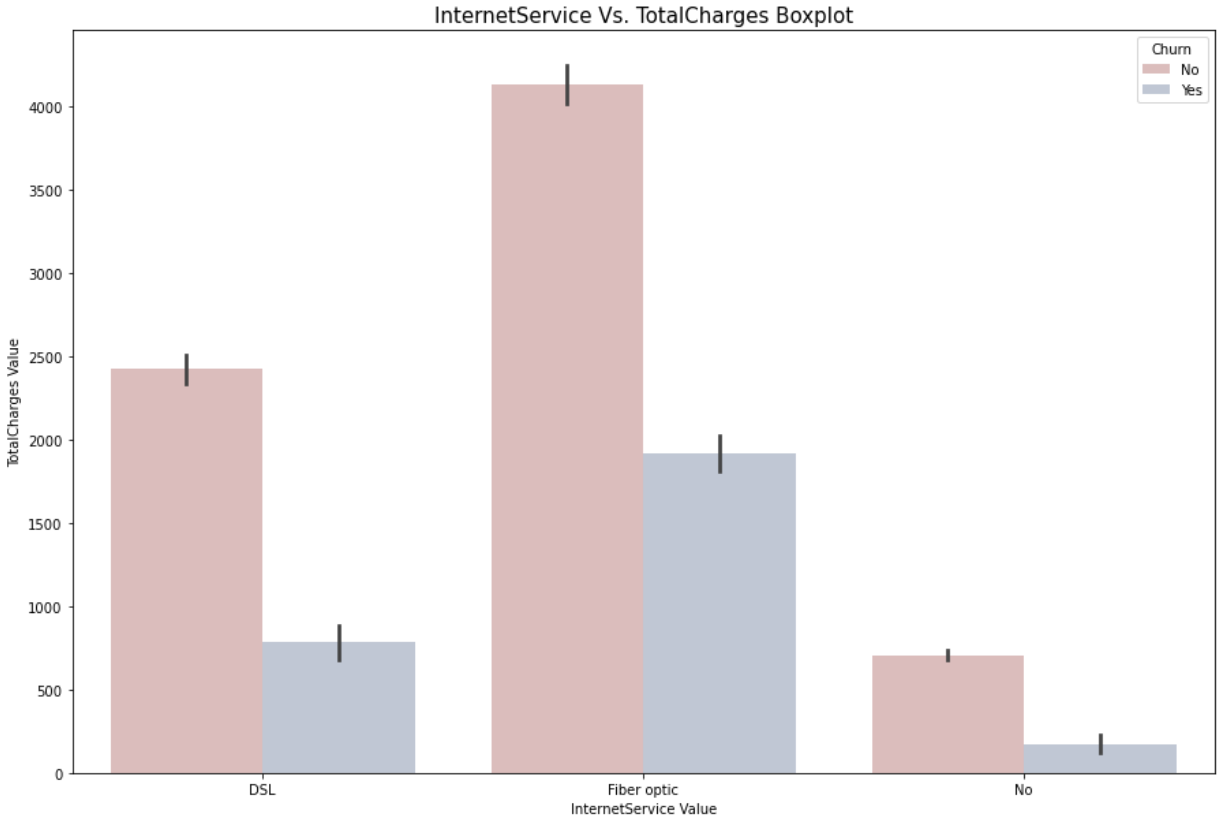
In [60]:
```
barplot_function(dataframe = df, x_value = 'DeviceProtection', y_value = 'TotalCharg
```




# Observation:

1. This is another way to represent the Device Projection plans and the total charges.
2. We have taken a barplot which represents the average total charges for all the customers based on the device projection plans.

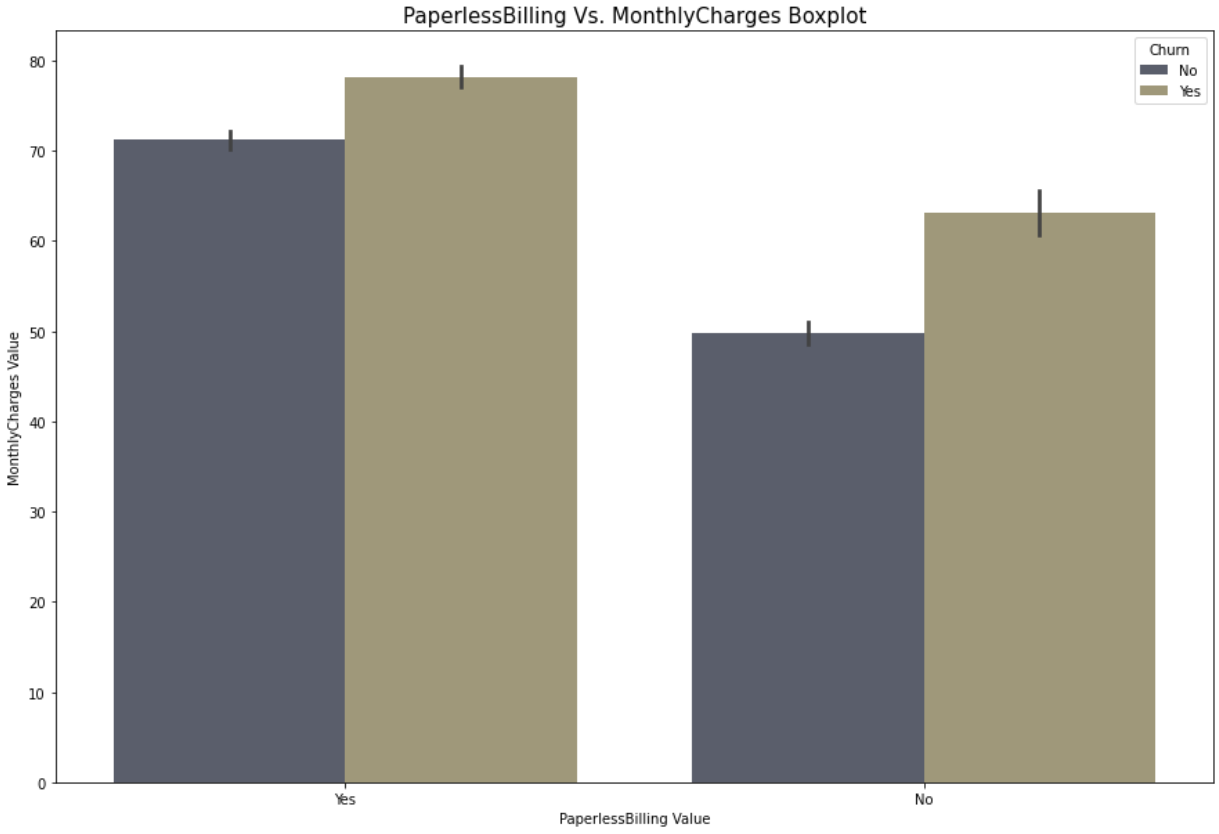
In [61]:
```
barplot_function(dataframe = df, x_value = 'InternetService', y_value = 'TotalCharge
                 hue = 'Churn', palette = random.choice(palette_values))
```



# Observation:

1. Based on this plot, we see that a large portion of customers from Fiber optic option tend to leave the service compared to other internet services.
2. Other services such as DSL service have higher number of customers who are willing to stay with the service.
3. Therefore, Telco might consider what might be the potential case for customers who have taken fiber option to leave the service.
4. If they could come up with the right tactics to improve their fiber option service, this ensures that a large portion of customers are retained.

In [62]:
```
barplot_function(dataframe = df, x_value = 'PaperlessBilling', y_value = 'MonthlyCha
```




# Observation:

1. Paperless billing customers usually leverage the freedom to use their credit/debit cards and pay more amount compared to the others who prefer paper billing options.
2. This is because it is more convenient to use credit/debit cards to make transaction compared to other services.
3. Hence, whenever a new customer is going to register for the service, Telco can estimate the total charges that might be taken into consideration based on whether a customer opts for paperless billing or not.

In [63]:
```
df.head()
```

Out[63]:

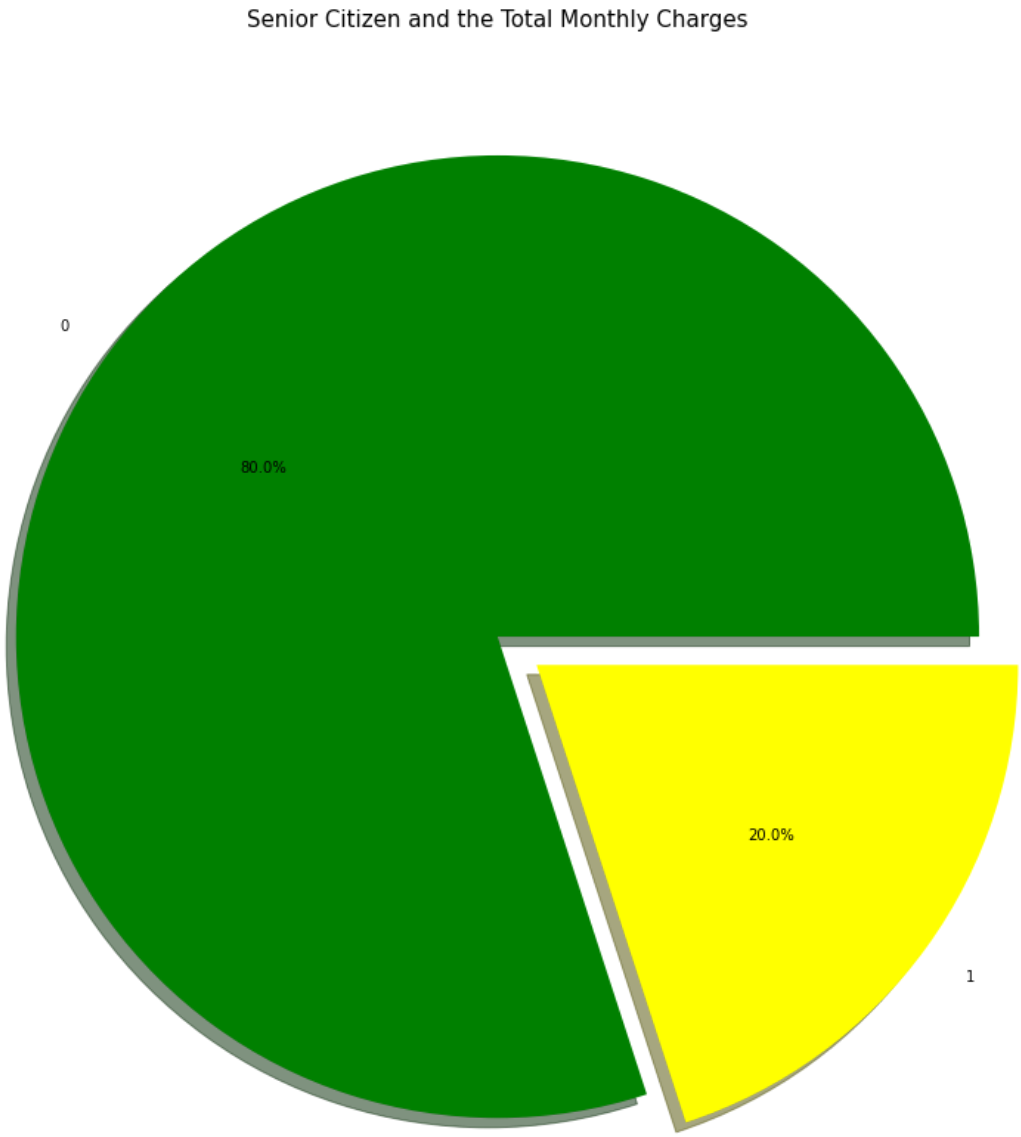
| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [64]:

```
plt.figure(figsize = (15, 15))
plt.pie(x = df.groupby(by = 'SeniorCitizen').sum()['MonthlyCharges'], labels = df.gr
        explode = (0, 0.1), shadow = True,
        autopct = '%1.1f%%', colors = ['Green', 'Yellow'])
plt.title('Senior Citizen and the Total Monthly Charges', fontsize = 15)
plt.show()
```




# Observation:

1. Based on the plots, it could be seen that senior citizens pay less amount compared to the non-seniors as shown in the above plot respectively.
2. We have taken the total sum of the monthly charges for senior citizens by grouping them based on their classes as shown in the pie plot above.

In [65]:
```
df.head()
```

Out[65]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |

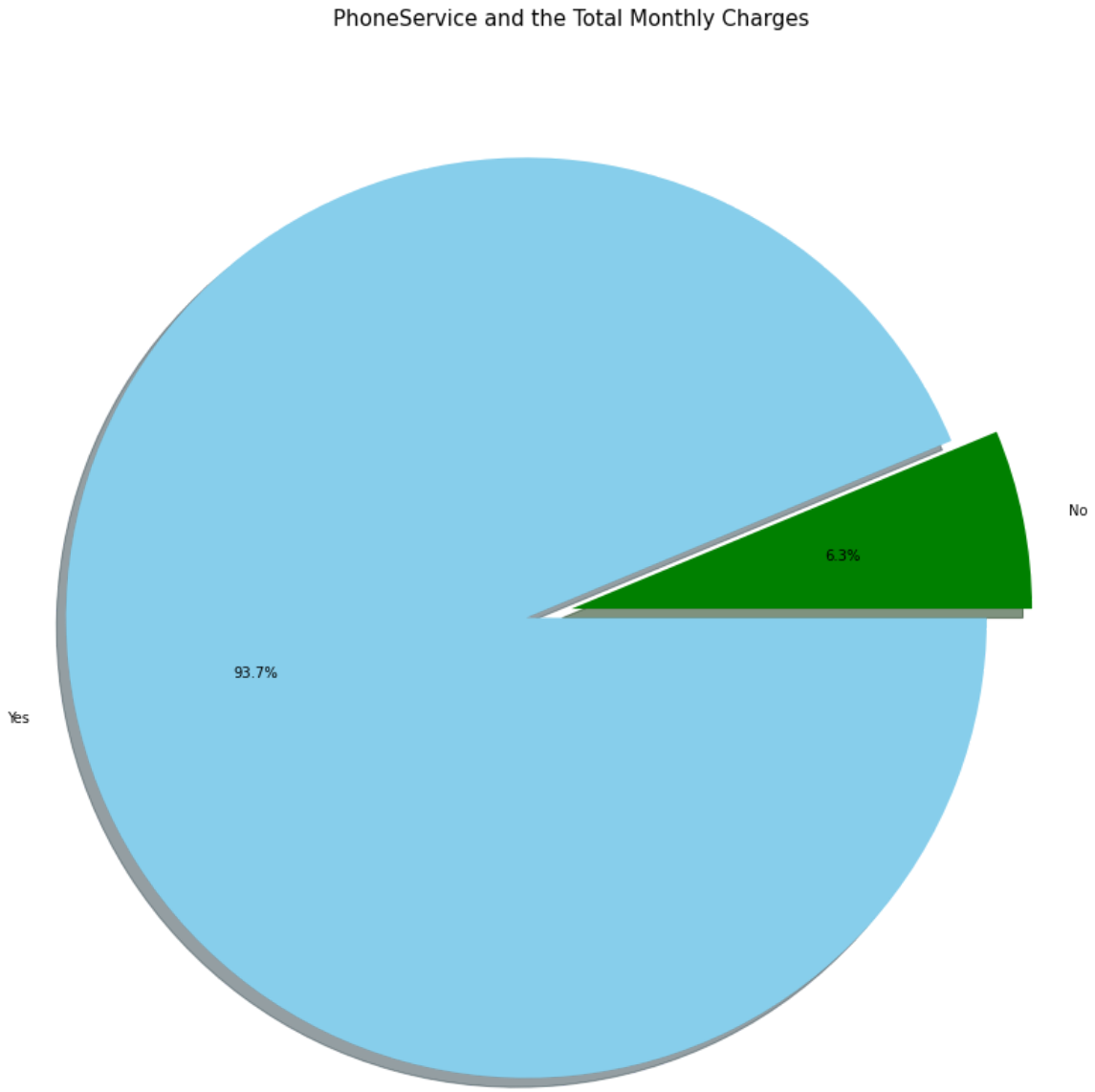
| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [66]:
```
df.groupby(by = 'PhoneService').sum()['MonthlyCharges']
```

Out[66]:
```
PhoneService
No       28663.5
Yes     427453.1
Name: MonthlyCharges, dtype: float64
```

In [67]:
```
plt.figure(figsize = (15, 15))
plt.pie(x = df.groupby(by = 'PhoneService').sum()['MonthlyCharges'], labels = df.gro
        explode = (0, 0.1), shadow = True,
        autopct = '%1.1f%%', colors = ['Green', 'SkyBlue'])
plt.title('PhoneService and the Total Monthly Charges', fontsize = 15)
plt.show()
```




# Observation:

1. People who have enrolled in the phone service have a significantly higher proportion of the amount that is payed monthly.
2. People who did not enroll in the phone service has lower proportion of the amount that is payed monthly.

In [68]:
```
df.head()
```

Out[68]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |

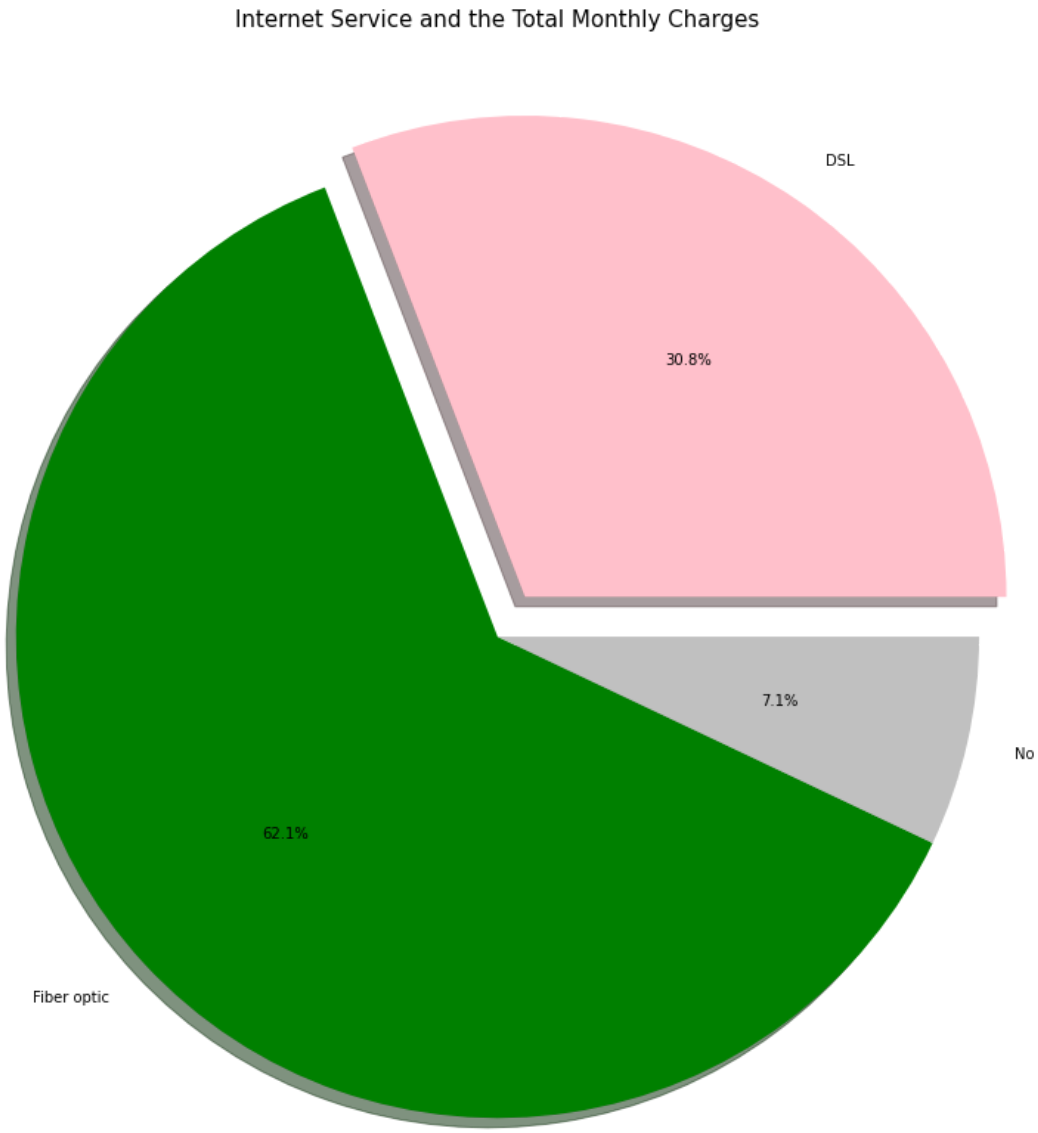
| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Ir |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [69]:
```
df.groupby(by = 'InternetService').sum()['MonthlyCharges']
```

Out[69]:
```
InternetService
DSL             140665.35
Fiber optic     283284.40
No               32166.85
Name: MonthlyCharges, dtype: float64
```

In [70]:
```
plt.figure(figsize = (15, 15))
plt.pie(x = df.groupby(by = 'InternetService').sum()['MonthlyCharges'], labels = df.
       explode = (0.1, 0, 0), shadow = True,
       autopct = '%1.1f%%', colors = ['Pink', 'Green', 'Silver'])
plt.title('Internet Service and the Total Monthly Charges', fontsize = 15)
plt.show()
```




## Observation:

1. The total charges were significantly higher for the Fiber optic customers compared to the other options as indicated in the plots shown above.
2. There are other factors as well that would influence the monthly charges which in turn would influence whether a customer would churn from the service.

In [71]:
```
df_categorical = df.select_dtypes(include = "object")
```

In [72]:
```
df_numerical = df.select_dtypes(exclude = "object")
```

In [73]:
```
print("The columns that are categorical in our data are:\n {}".format(df_categorical
```

```
The columns that are categorical in our data are:
 Index(['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService',
       'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
```

```
       'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn'],
      dtype='object')
```

In [74]:
```
print("The columns that are numerical in our data are:\n {}".format(df_numerical.col
```

```
The columns that are numerical in our data are:
 Index(['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges'], dtype='objec
t')
```

In [75]:
```
df_categorical.head()
```

Out[75]:

| | customerID | gender | Partner | Dependents | PhoneService | MultipleLines | InternetService | OnlineS |
|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | Yes | No | No | No phone service | DSL | |
| **1** | 5575-GNVDE | Male | No | No | Yes | No | DSL | |
| **2** | 3668-QPYBK | Male | No | No | Yes | No | DSL | |
| **3** | 7795-CFOCW | Male | No | No | No | No phone service | DSL | |
| **4** | 9237-HQITU | Female | No | No | Yes | No | Fiber optic | |

In [76]:
```
df_numerical.head()
```

Out[76]:

| | SeniorCitizen | tenure | MonthlyCharges | TotalCharges |
|---|---|---|---|---|
| **0** | 0 | 1 | 29.85 | 29.85 |
| **1** | 0 | 34 | 56.95 | 1889.50 |
| **2** | 0 | 2 | 53.85 | 108.15 |
| **3** | 0 | 45 | 42.30 | 1840.75 |
| **4** | 0 | 2 | 70.70 | 151.65 |

In [77]:
```
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [78]:
```
df.head()
```

Out[78]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | In |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | In |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [79]:
```
df_numerical.head()
```

Out[79]:

| | SeniorCitizen | tenure | MonthlyCharges | TotalCharges |
|---|---|---|---|---|
| **0** | 0 | 1 | 29.85 | 29.85 |
| **1** | 0 | 34 | 56.95 | 1889.50 |
| **2** | 0 | 2 | 53.85 | 108.15 |
| **3** | 0 | 45 | 42.30 | 1840.75 |
| **4** | 0 | 2 | 70.70 | 151.65 |

In [ ]:
```
```

In [80]:
```
df_categorical.head()
```

Out[80]:

| | customerID | gender | Partner | Dependents | PhoneService | MultipleLines | InternetService | OnlineS |
|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | Yes | No | No | No phone service | DSL | |
| **1** | 5575-GNVDE | Male | No | No | Yes | No | DSL | |
| **2** | 3668-QPYBK | Male | No | No | Yes | No | DSL | |
| **3** | 7795-CFOCW | Male | No | No | No | No phone service | DSL | |
| **4** | 9237-HQITU | Female | No | No | Yes | No | Fiber optic | |

In [ ]:
```
```

In [81]:
```
float('314.23')
```

Out[81]: 314.23

In [82]:
```
df_categorical.head()
```

Out[82]:

| | customerID | gender | Partner | Dependents | PhoneService | MultipleLines | InternetService | OnlineS |
|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | Yes | No | No | No phone service | DSL | |
| **1** | 5575-GNVDE | Male | No | No | Yes | No | DSL | |
| **2** | 3668-QPYBK | Male | No | No | Yes | No | DSL | |
| **3** | 7795-CFOCW | Male | No | No | No | No phone service | DSL | |
| **4** | 9237-HQITU | Female | No | No | Yes | No | Fiber optic | |

In [83]:
```
df_categorical.drop(['customerID'], axis = 1, inplace = True)
```

In [84]:
```
pd.get_dummies(df_categorical.gender, drop_first = True).head()
```

Out[84]:

| | Male |
|---|---|
| **0** | 0 |
| **1** | 1 |
| **2** | 1 |
| **3** | 1 |
| **4** | 0 |

In [85]:
```
df_dummy_encoding = pd.get_dummies(df_categorical, drop_first = True)
```

In [86]:
```
df_final = pd.concat([df_dummy_encoding, df_numerical], axis = 1)
```

In [87]:
```
df_final.head()
```

Out[87]:

| | gender_Male | Partner_Yes | Dependents_Yes | PhoneService_Yes | MultipleLines_No phone service | MultipleLines_Yes |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 1 | 0 |
| **1** | 1 | 0 | 0 | 1 | 0 | 0 |
| **2** | 1 | 0 | 0 | 1 | 0 | 0 |
| **3** | 1 | 0 | 0 | 0 | 1 | 0 |

| | gender_Male | Partner_Yes | Dependents_Yes | PhoneService_Yes | MultipleLines_No phone service | MultipleLines_Yes |
|---|---|---|---|---|---|---|
| **4** | 0 | 0 | 0 | 1 | 0 | 0 |

5 rows × 31 columns

In [88]:
```
from sklearn.model_selection import train_test_split
```

In [89]:
```
df_final.head()
```

Out[89]:

| | gender_Male | Partner_Yes | Dependents_Yes | PhoneService_Yes | MultipleLines_No phone service | MultipleLines_Yes |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 1 | 0 |
| **1** | 1 | 0 | 0 | 1 | 0 | 0 |
| **2** | 1 | 0 | 0 | 1 | 0 | 0 |
| **3** | 1 | 0 | 0 | 0 | 1 | 0 |
| **4** | 0 | 0 | 0 | 1 | 0 | 0 |

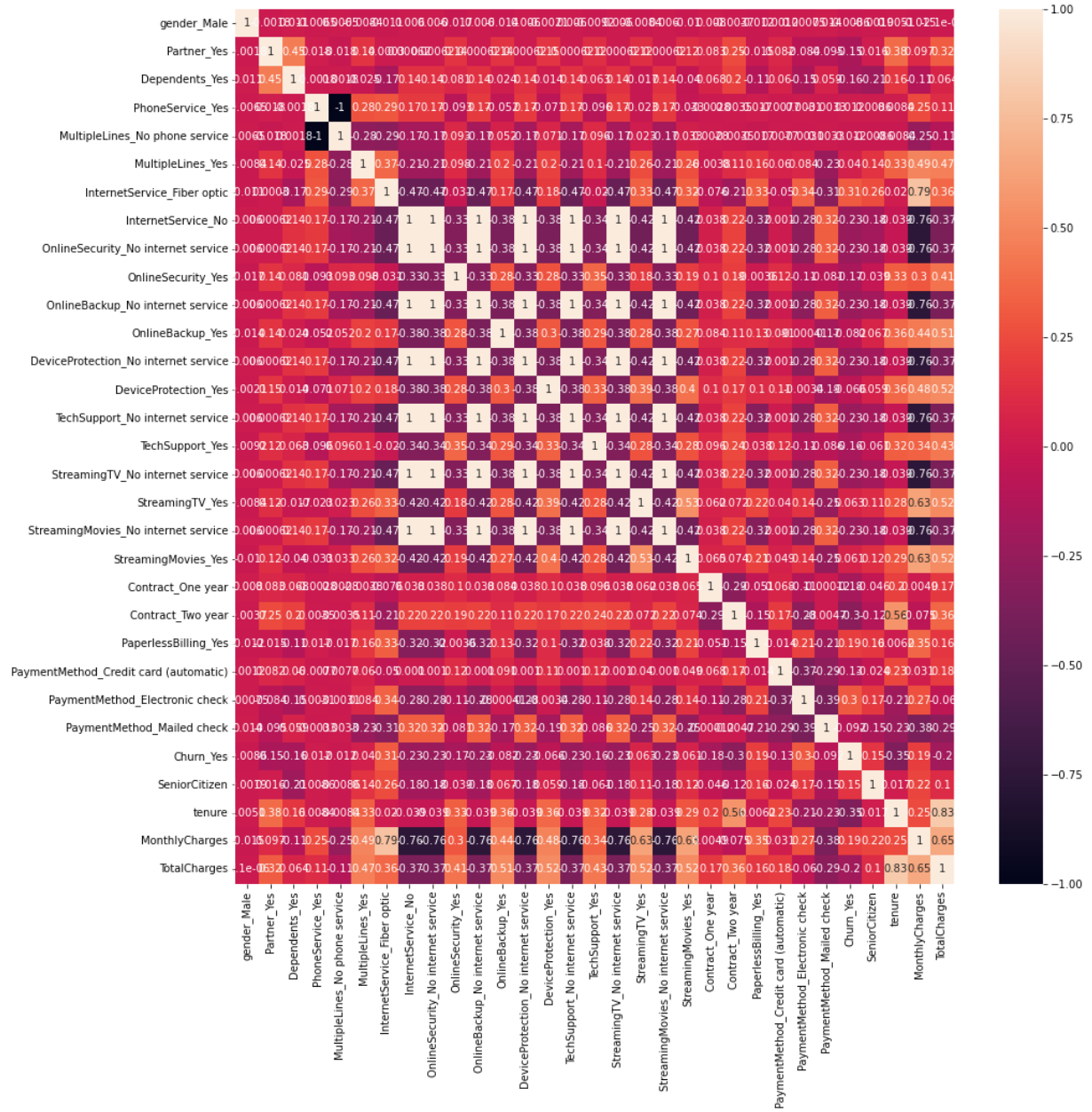
5 rows × 31 columns

In [90]:
```
df.corr()
```

Out[90]:

| | SeniorCitizen | tenure | MonthlyCharges | TotalCharges |
|---|---|---|---|---|
| **SeniorCitizen** | 1.000000 | 0.016567 | 0.220173 | 0.102652 |
| **tenure** | 0.016567 | 1.000000 | 0.247900 | 0.825466 |
| **MonthlyCharges** | 0.220173 | 0.247900 | 1.000000 | 0.650865 |
| **TotalCharges** | 0.102652 | 0.825466 | 0.650865 | 1.000000 |

In [91]:
```
plt.figure(figsize = (15, 15))
sns.heatmap(df_final.corr(), annot = True)
plt.show()
```




In [ ]:

In [69]:
```
X = df_final.drop(['Churn_Yes'], axis = 1)
y = df_final['Churn_Yes']
```

In [70]:
```
X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size = 0.3, random_s
```

In [71]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_transformed = scaler.transform(X_train)
X_test_transformed = scaler.transform(X_test)
```