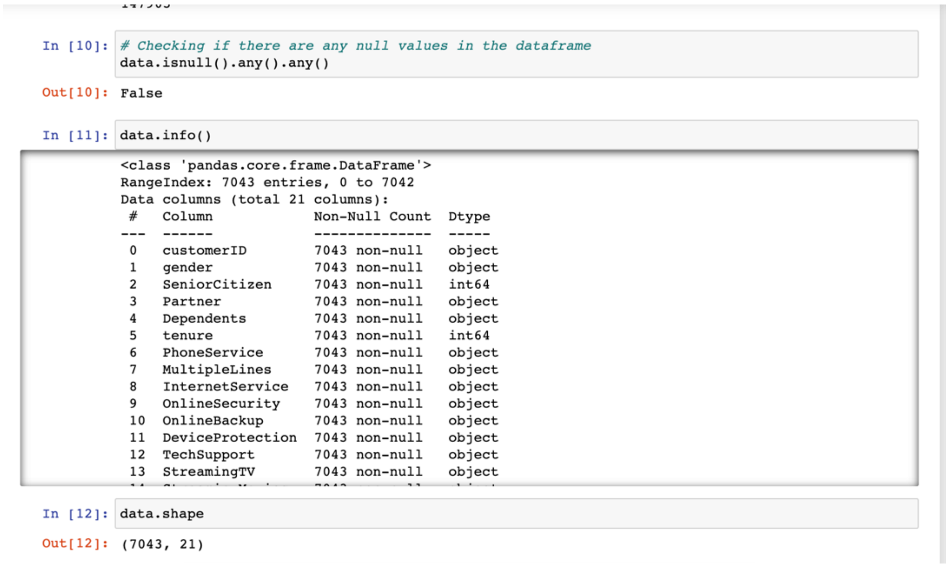
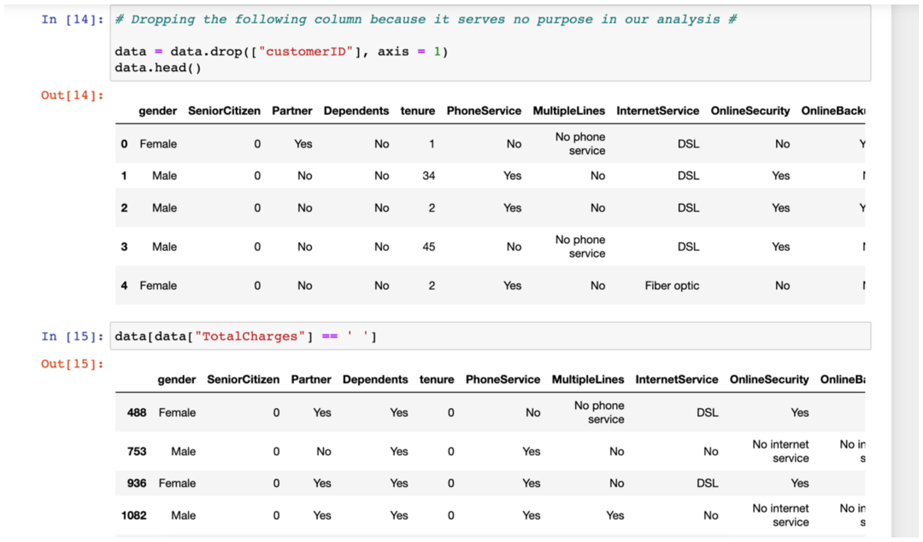
Data Preprocessing and Collection:

We ran a simple line of code to check the number of missing values in the dataset. On running the line of code, we found that the value it returned was ‘False’ indicating that there were no missing values in the dataset.



```
In [10]: # Checking if there are any null values in the dataframe
         data.isnull().any().any()

Out[10]: False

In [11]: 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

In [12]: data.shape

Out[12]: (7043, 21)
```

In the screenshot below you can see that we are dropping the column ‘CustomerID’ because it is of no relevance in our study. Along with that we also run a line of python code to filter the data frame to only include rows where the value in the column ‘TotalCharges’ is an empty string.



```
In [14]: # Dropping the following column because it serves no purpose in our analysis #

         data = data.drop(["customerID"], axis = 1)
         data.head()
```

Out[14]:

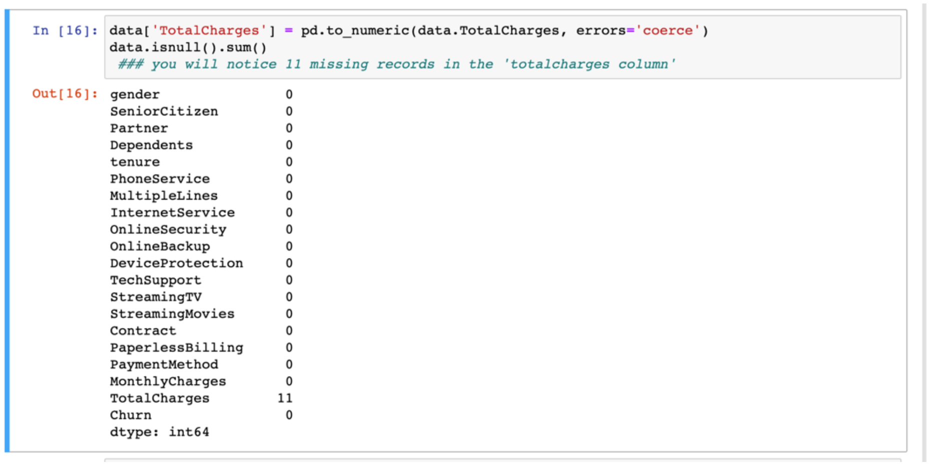
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBack
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Y
1	Male	0	No	No	34	Yes	No	DSL	Yes	I
2	Male	0	No	No	2	Yes	No	DSL	Yes	Y
3	Male	0	No	No	45	No	No phone service	DSL	Yes	I
4	Female	0	No	No	2	Yes	No	Fiber optic	No	I

```
In [15]: data[data["TotalCharges"] == ' ']
```

Out[15]:

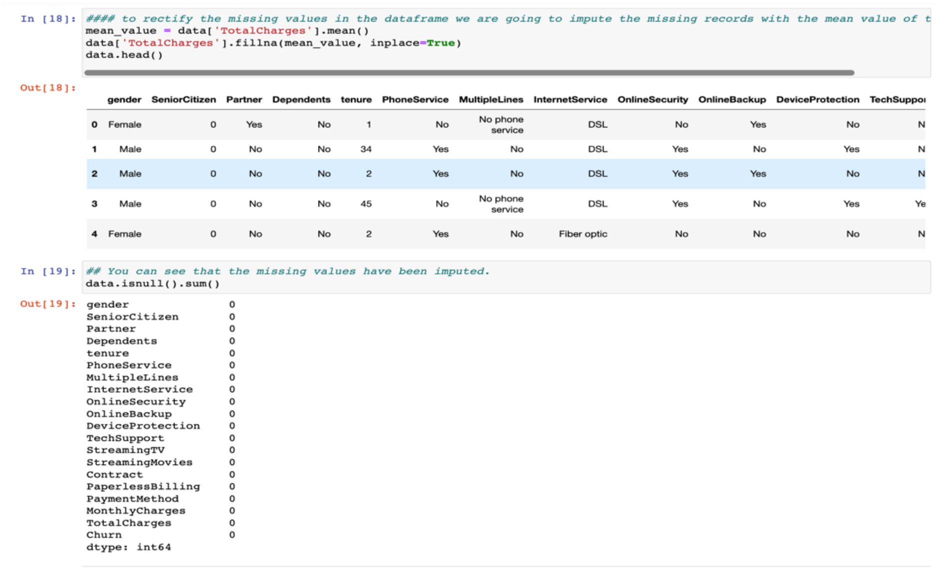
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBa
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No in s
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No in s

In this screenshot, we run a code to the convert the column ‘TotalCharges’ to a numeric datatype using the Pandas “to numeric” function to perform certain numerical calculations on the column. The argument “errors = coerce” is asking the function to convert any non-numeric values in the column to “NaN” (Not a Number) values. You will notice that on running this code, we understand that there are 11 missing records in the column ‘TotalCharges’.



```
In [16]: data['TotalCharges'] = pd.to_numeric(data.TotalCharges, errors='coerce')
         data.isnull().sum()
          ### you will notice 11 missing records in the 'totalcharges column'

Out[16]: 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
```

In this screenshot, we see that we are carrying out the process of imputation. This helps with rectifying or filling the missing values or missing records in the column “TotalCharges”. The purpose of this code is to fill any missing record in the column with a reasonable estimate based on the mean value of the column. This helps with ensuring that the dataset is complete and can be used for further analysis.



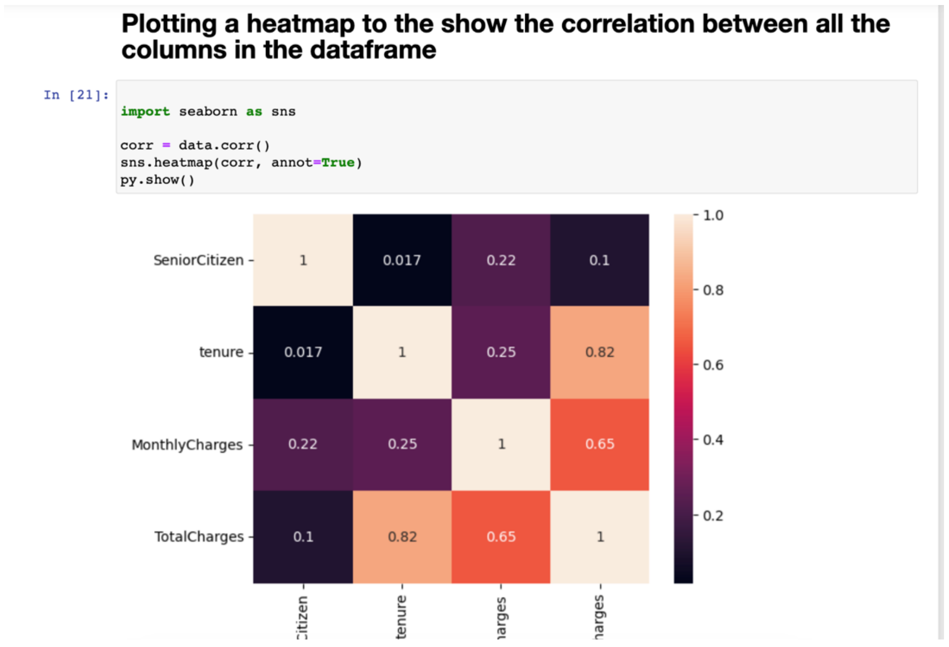
```
In [18]: #### to rectify the missing values in the dataframe we are going to impute the missing records with the mean value of t
         mean_value = data['TotalCharges'].mean()
         data['TotalCharges'].fillna(mean_value, inplace=True)
         data.head()
```

Out[18]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSuppoı
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	N
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	N
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	N
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Ye
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	N

```
In [19]: ## You can see that the missing values have been imputed.
         data.isnull().sum()

Out[19]: gender              0
         SeniorCitizen       0
         Partner             0
         Dependents          0
         tenure              0
         PhoneService        0
         MultipleLines       0
         InternetService     0
         OnlineSecurity      0
         OnlineBackup        0
         DeviceProtection    0
         TechSupport         0
         StreamingTV         0
         StreamingMovies     0
         Contract            0
         PaperlessBilling    0
         PaymentMethod       0
         MonthlyCharges      0
         TotalCharges        0
         Churn               0
         dtype: int64
```

In this, we notice a correlation matrix that is plotted as a heatmap indicating the correlation between the different features in the dataset. With this plot we can explore the different features and come up with various visualization plots to illustrate the key inferences we can draw from the dataset.



Plotting a heatmap to the show the correlation between all the columns in the dataframe

```
In [21]:
         import seaborn as sns

         corr = data.corr()
         sns.heatmap(corr, annot=True)
         py.show()
```

The lines of code below are executed to perform feature scaling on the columns specified. Feature scaling is a very common data preprocessing technique that is used to normalize the range of feature values in a dataset. This is usually done to ensure that the features with larger ranges of values do not dominate features with smaller ranges during model training. This is the last preprocessing step that is executed before attaching all the data mining models to train the dataset which will be explained in the next section.