```
# Importing the required libraries

import numpy as np
import pandas as pd



# 2. Load the dataset

df=pd.read_csv('/content/Churn_Modelling.csv')


# Visualize the data

df.head()
```

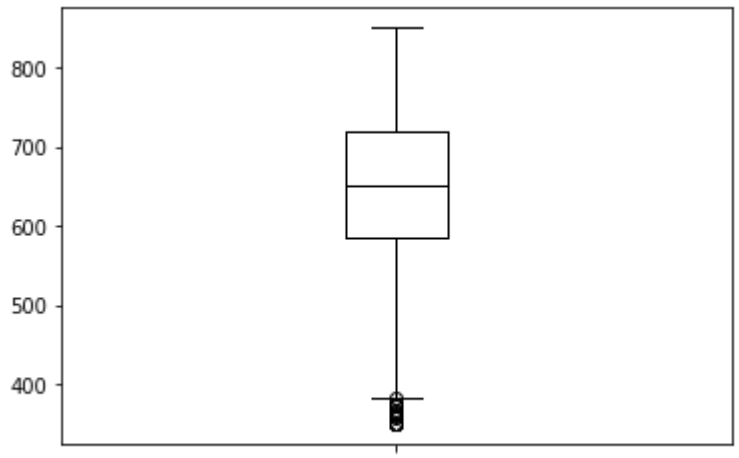
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

```
#  3. Perform Below Visualizations.
#  Univariate Analysis

df['CreditScore'].value_counts
```

```
<bound method IndexOpsMixin.value_counts of 0       619
1       608
2       502
3       699
4       850
       ...
9995    771
9996    516
9997    709
9998    772
9999    792
Name: CreditScore, Length: 10000, dtype: int64>
```

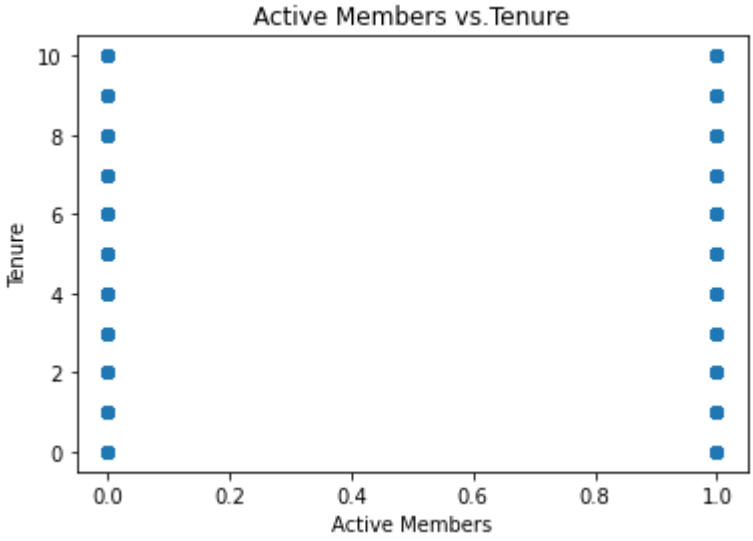
```
import matplotlib.pyplot as plt

df.boxplot(column=['CreditScore'], grid=False, color='black')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5057f5290>
```




```
# Bivariate Analysis

import matplotlib.pyplot as plt
plt.scatter(df.IsActiveMember, df.Tenure)
plt.title('Active Members vs.Tenure')
plt.xlabel('Active Members')
plt.ylabel('Tenure')
```

```
Text(0, 0.5, 'Tenure')
```




```
import seaborn as sns


# Multi-variate Analysis

sns.lmplot("Age","NumOfProducts",df,hue="NumOfProducts",fit_reg=False);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
```


```
import statistics as st
```


```
# 4. Descriptive Statitics
nums=[1,2,3,5,7]
st.mean(nums)
```

```
3.6
```

```
# Mode
nums=[1,2,3,5,7,9,7,2,7,6]
st.mode(nums)
```

```
7
```

```
# Median
st.median(nums)
```

```
5.5
```

```
# 5. Handle Missing Values
df.isnull()
```

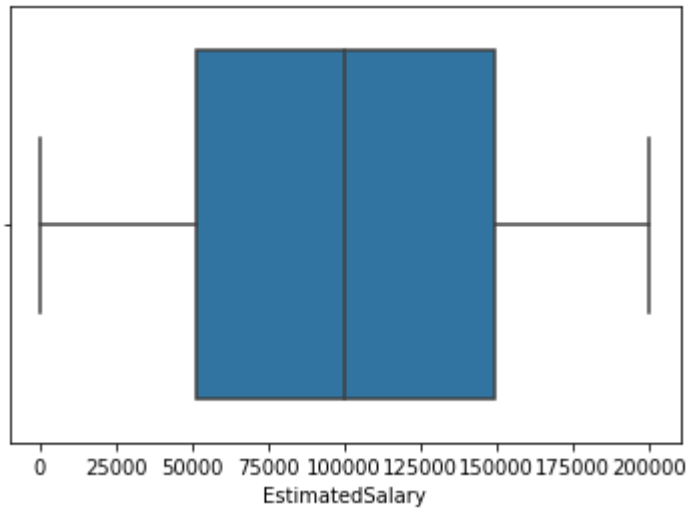
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenur
0	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	.
9995	False	False	False	False	False	False	False	Fals

```
import seaborn as sns
```

9997	False	False	False	False	False	False	False	Fals

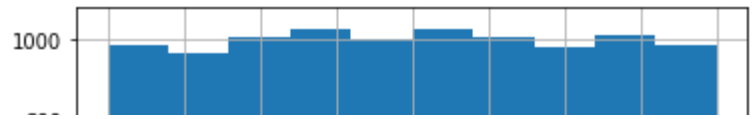
```
# 6. Finding outliers
sns.boxplot(df['EstimatedSalary'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fa50576ab90>
```




```
df['EstimatedSalary'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa505744cd0>
```




```
print('skewness value of Age: ',df['Age'].skew())
print('skewness value of EstimatedSalary  : ',df['EstimatedSalary'].skew())
```

```
skewness value of Age:  1.0113202630234552
skewness value of EstimatedSalary       :  0.0020853576615585162
```

```
Q1 = df['EstimatedSalary'].quantile(0.85)
Q3 = df['EstimatedSalary'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 1.5
Fare_outliers = df[(df['EstimatedSalary'] < Q1 - whisker_width*IQR) | (df['EstimatedSalary']
Fare_outliers.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

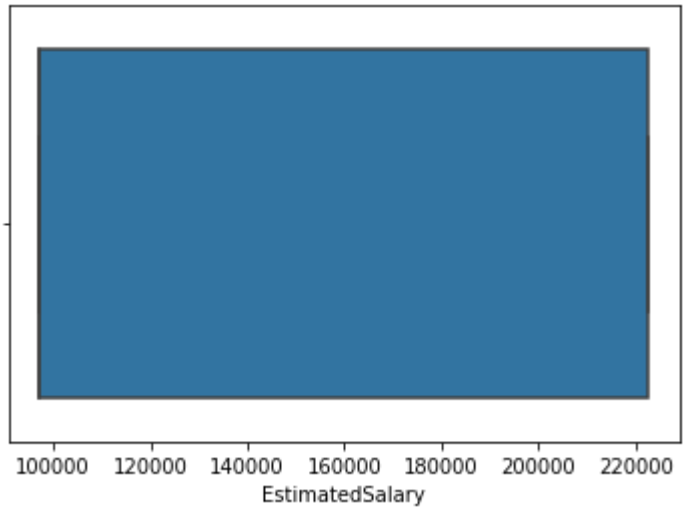
```
# Prints Outliers
fare_mean = df['EstimatedSalary'].mean()
fare_std = df['EstimatedSalary'].std()
low= fare_mean -(1 * fare_std)
high= fare_mean + (1 * fare_std)
fare_outliers = df[(df['EstimatedSalary'] < low) | (df['EstimatedSalary'] > high)]
fare_outliers.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
6	7	15592531	Bartlett	822	France	Male	50	7	
12	13	15632264	Kay	476	France	Female	34	10	
13	14	15691483	Chin	549	France	Female	25	5	
16	17	15737452	Romeo	653	Germany	Male	58	1	1326(
17	18	15788218	Henderson	549	Spain	Female	24	9	

```
# Replace the Outliers
```

```
Q1 = df['EstimatedSalary'].quantile(0.85)
Q3 = df['EstimatedSalary'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 2.5
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
df['EstimatedSalary']=np.where(df['EstimatedSalary']>upper_whisker,upper_whisker,np.where(df[


sns.boxplot(df['EstimatedSalary'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fa505754dd0>
```




```
Q1 = df['EstimatedSalary'].quantile(0.25)
Q3 = df['EstimatedSalary'].quantile(0.90)
IQR = Q3 - Q1
whisker_width = 2.5
lower_whisker = Q1 - (whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
index=df['EstimatedSalary'][(df['EstimatedSalary']>upper_whisker)|(df['EstimatedSalary']<lowe
df.drop(index,inplace=True)


sns.boxplot(df['EstimatedSalary'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fa505629690>
```

```
# 7. Check for Categorical Columns
df._get_numeric_data()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCa
0	1	15634602	619	42	2	0.00	1	
1	2	15647311	608	41	1	83807.86	1	
2	3	15619304	502	42	8	159660.80	3	
3	4	15701354	699	39	1	0.00	2	
4	5	15737888	850	43	2	125510.82	1	
...	...	...	...	...	...	...	...	
9995	9996	15606229	771	39	5	0.00	2	
9996	9997	15569892	516	35	10	57369.61	1	
9997	9998	15584532	709	36	7	0.00	1	
9998	9999	15682355	772	42	3	75075.31	2	
9999	10000	15628319	792	28	4	130142.79	1	

10000 rows × 11 columns

```
cols = df.columns


num_cols = df._get_numeric_data().columns


num_cols
```

```
Index(['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited'],
      dtype='object')
```

```
list(set(cols) - set(num_cols))
```

```
['Surname', 'Gender', 'Geography']
```

```
# 8. Split the data into Independent and Dependent variables

x=df.iloc[:,0:4].values
y=df.iloc[:,4:5].values


# 9. Scale the Independent variable

import pandas
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
df = pandas.read_csv("Churn_Modelling.csv")
X = df[['Age', 'EstimatedSalary']]
scaledX = scale.fit_transform(X)
print(scaledX)

     [[ 0.29351742  0.02188649]
      [ 0.19816383  0.21653375]
      [ 0.29351742  0.2406869 ]
      ...
      [-0.27860412 -1.00864308]
      [ 0.29351742 -0.12523071]
      [-1.04143285 -1.07636976]]


from sklearn.model_selection import train_test_split


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)


# 10. Split the data into Training and Testing

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=0)
```