DSC680_CCPB_EDA

January 30, 2022

```
[1]: from ipynb.fs.full.DSC680_CCPB_Data_Preprocessing import *
```

```
********* Customer Churn Dataset***************


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB


********* number of unique classes of each attributes***************


RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
```

```
Gender               2
Age                 70
Tenure              11
Balance           6382
NumOfProducts        4
HasCrCard            2
IsActiveMember       2
EstimatedSalary   9999
Exited               2
dtype: int64
```

```
********* description of the dataset****************
```

```
         RowNumber    CustomerId   CreditScore           Age        Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000  10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800      5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806      2.892174
min        1.00000  1.556570e+07    350.000000     18.000000      0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000      3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000      5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000      7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000     10.000000

             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
mean    76485.889288       1.530200      0.70550        0.515100
std     62397.405202       0.581654      0.45584        0.499797
min         0.000000       1.000000      0.00000        0.000000
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

```
********* Sample Data from file****************


   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0


********* dropped columns [CustomerId,RowNumber,Surname]****************


CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object


*********Check number of NaN or NULL****************
```

```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

1 Analyze and Visualize features - EDA

[2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

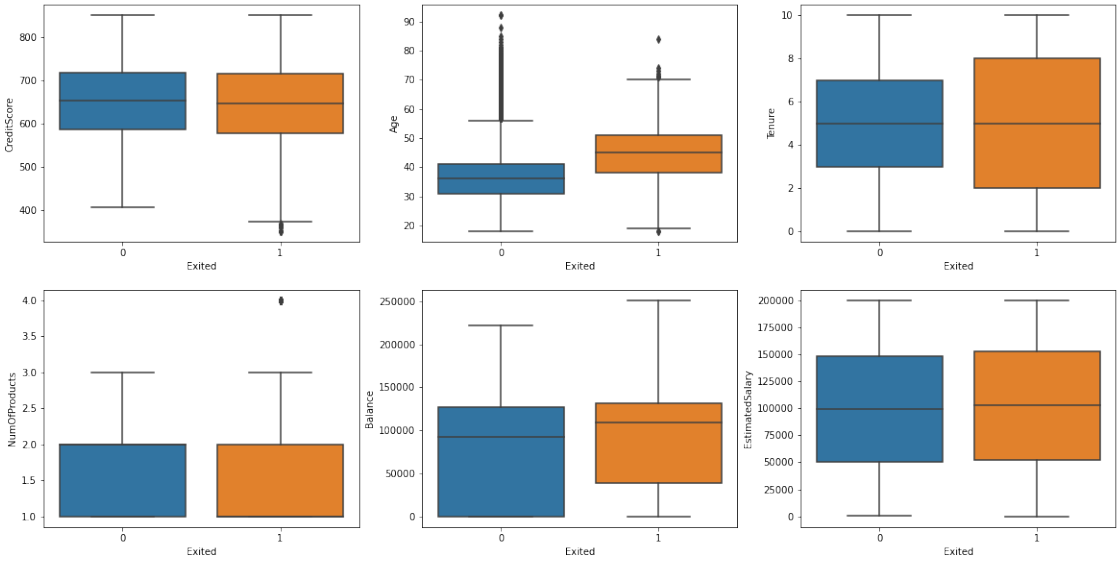
[3]:

```
# boxplot for numerical feature
print ("\n\n*********"+"\033[1m"+"boxplot for numerical features"+␣
 ↪"\033[0m"+"**************** \n\n")
_,axss = plt.subplots(2,3, figsize=[20,10])
sns.boxplot(x='Exited', y ='CreditScore', data=ccpb_df, ax=axss[0][0])
sns.boxplot(x='Exited', y ='Age', data=ccpb_df, ax=axss[0][1])
sns.boxplot(x='Exited', y ='Tenure', data=ccpb_df, ax=axss[0][2])
sns.boxplot(x='Exited', y ='NumOfProducts', data=ccpb_df, ax=axss[1][0])
sns.boxplot(x='Exited', y ='Balance', data=ccpb_df, ax=axss[1][1])
sns.boxplot(x='Exited', y ='EstimatedSalary', data=ccpb_df, ax=axss[1][2])
```

```


*********boxplot for numerical features****************


```

[3]:

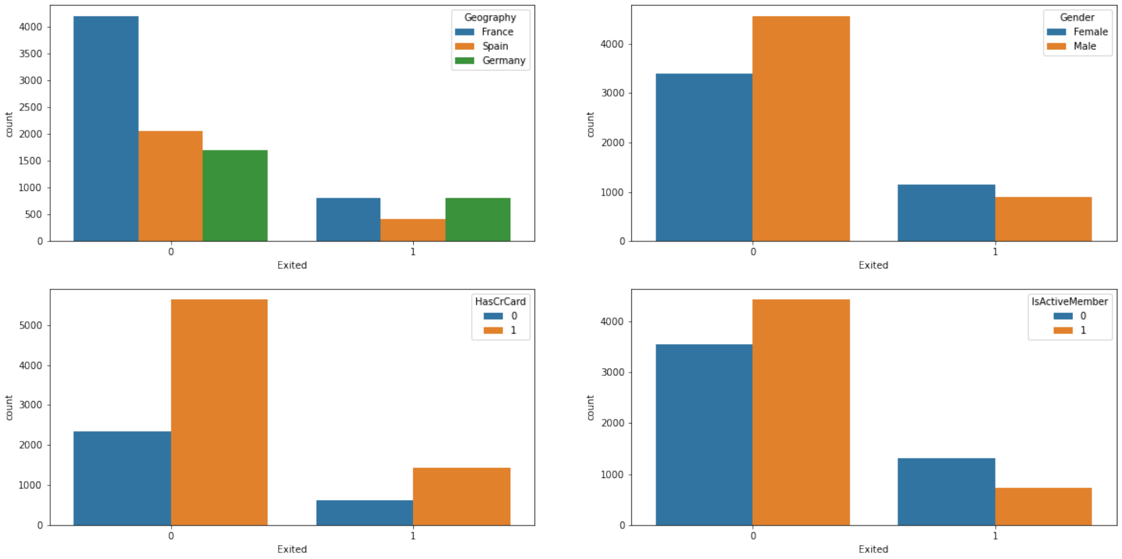
```
<AxesSubplot:xlabel='Exited', ylabel='EstimatedSalary'>
```



[5]:
```python
# Analyze correlation among "Exited" and other Categorical Features
print ("\n\n*********"+"\033[1m"+"Analyze correlation among Exited and other␣
 ↪Categorical Features"+ "\033[0m"+"**************** \n\n")
_,axss = plt.subplots(2,2, figsize=[20,10])
sns.countplot(x='Exited', hue='Geography', data=ccpb_df, ax=axss[0][0])
sns.countplot(x='Exited', hue='Gender', data=ccpb_df, ax=axss[0][1])
sns.countplot(x='Exited', hue='HasCrCard', data=ccpb_df, ax=axss[1][0])
sns.countplot(x='Exited', hue='IsActiveMember', data=ccpb_df, ax=axss[1][1])
```

```
*********Analyze correlation among Exited and other Categorical
Features****************
```

[5]:
```
<AxesSubplot:xlabel='Exited', ylabel='count'>
```



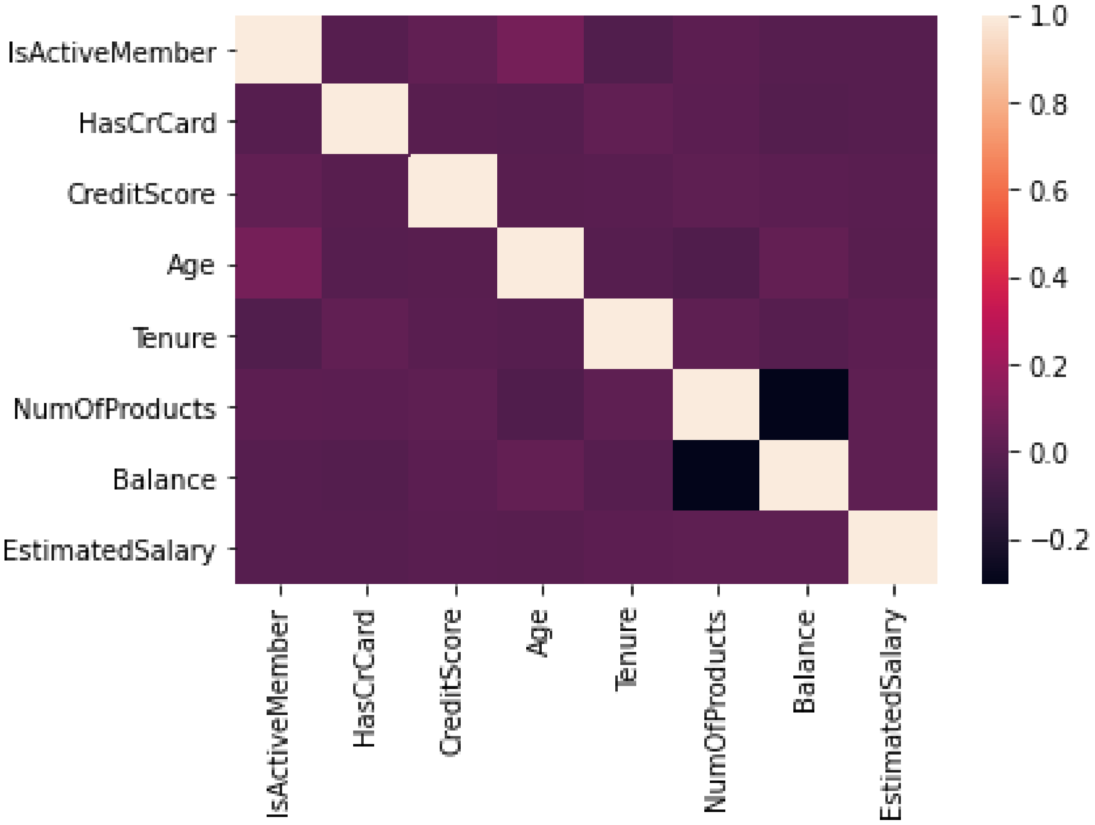
[6]:

```
# Analyze Correlation between numerical feature using Heatmap plot
print ("\n\n*********"+"\033[1m"+"Analyze Correlation between numerical feature␣
↪using Heatmap plot"+ "\033[0m"+"*************** \n\n")
correlation = ccpb_df[['IsActiveMember','HasCrCard','CreditScore', 'Age',␣
↪'Tenure', 'NumOfProducts','Balance', 'EstimatedSalary']].corr()
sns.heatmap(correlation)
```

```
*********Analyze Correlation between numerical feature using Heatmap
plot***************
```

[6]:

```
<AxesSubplot:>
```



```
[7]: print ("\n\n*********"+"\033[1m"+"Correlation statistics"+␣
      ↪"\033[0m"+"**************** \n\n")
     display (correlation)
```

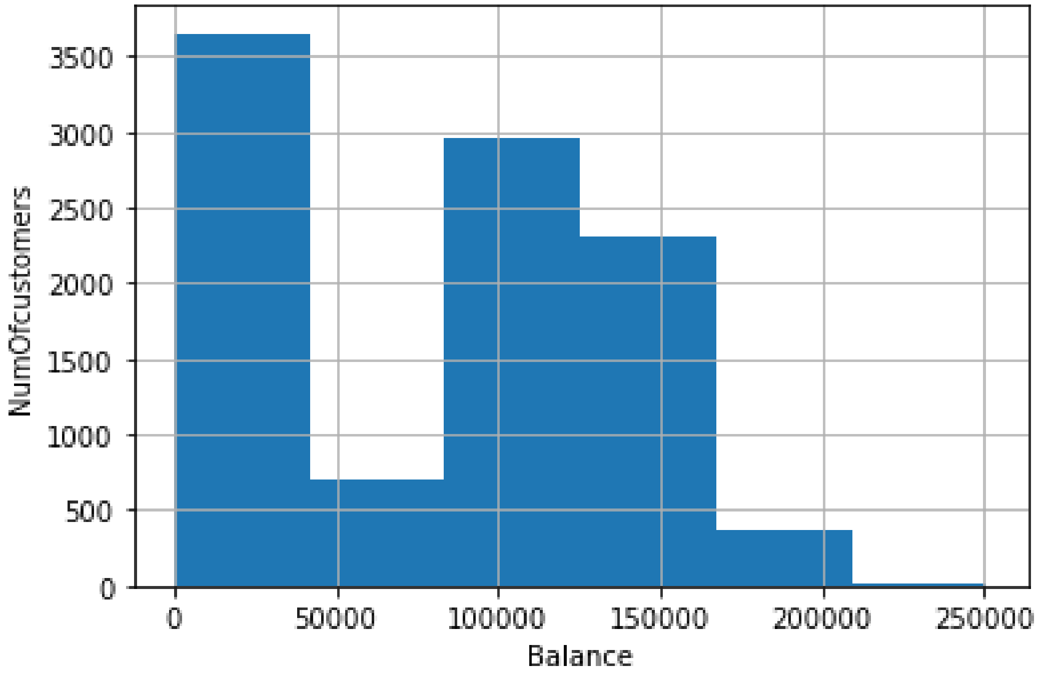
```
*********Correlation statistics****************


                 IsActiveMember  HasCrCard  CreditScore       Age    Tenure  \
IsActiveMember         1.000000  -0.011866     0.025651  0.085472 -0.028362
HasCrCard             -0.011866   1.000000    -0.005458 -0.011721  0.022583
CreditScore            0.025651  -0.005458     1.000000 -0.003965  0.000842
Age                    0.085472  -0.011721    -0.003965  1.000000 -0.009997
Tenure                -0.028362   0.022583     0.000842 -0.009997  1.000000
NumOfProducts          0.009612   0.003183     0.012238 -0.030680  0.013444
Balance               -0.010084  -0.014858     0.006268  0.028308 -0.012254
EstimatedSalary       -0.011421  -0.009933    -0.001384 -0.007201  0.007784

                 NumOfProducts   Balance  EstimatedSalary
```

```
IsActiveMember        0.009612 -0.010084        -0.011421
HasCrCard             0.003183 -0.014858        -0.009933
CreditScore           0.012238  0.006268        -0.001384
Age                  -0.030680  0.028308        -0.007201
Tenure                0.013444 -0.012254         0.007784
NumOfProducts         1.000000 -0.304180         0.014204
Balance              -0.304180  1.000000         0.012797
EstimatedSalary       0.014204  0.012797         1.000000
```

[8]:
```
print ("\n\n*********"+"\033[1m"+"Balance Distribution EDA"+␣
 ↪"\033[0m"+"**************** \n\n")
ccpb_df['Balance'].hist(bins=6)
plt.xlabel('Balance')
plt.ylabel('NumOfcustomers')
```

```
*********Balance Distribution EDA****************
```

[8]:
```
Text(0, 0.5, 'NumOfcustomers')
```



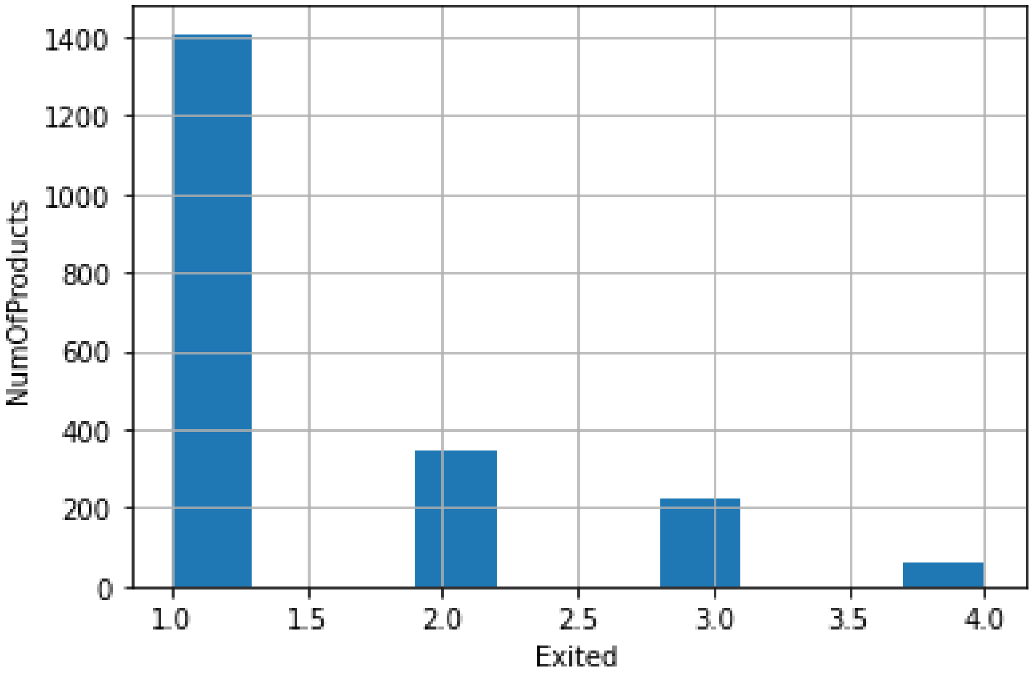
[9]:
```
# what balance did people exit bank?
```

```
print ("\n\n*********"+"\033[1m"+"What is the minumum balance of the customers␣
↪who exited the bank?"+ "\033[0m"+"**************** \n\n")
ccpb_exit_df=ccpb_df[ccpb_df.Exited==1]
ccpb_exit_df.Balance.mean()
plt.xlabel('Exited')
plt.ylabel('NumOfProducts')
# when exited.. was the number of products 0?
ccpb_exit_df.NumOfProducts.hist()
# peope who exited had 1 product
# (affinity of leaving when only 1 product is more as probably : not keen in␣
↪other products.)
```

```
*********What is the minumum balance of the customers who exited the
bank?****************
```

[9]:

```
<AxesSubplot:xlabel='Exited', ylabel='NumOfProducts'>
```

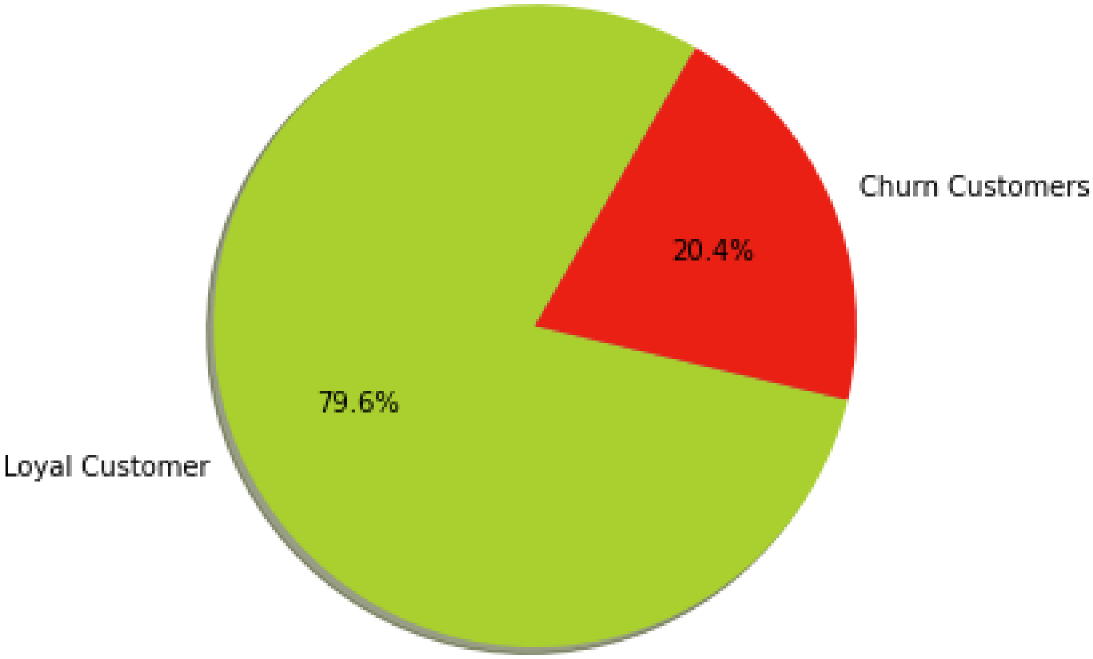


[12]:

```
#No of Exited vs Active  get the percentage split figure
print ("\n\n*********"+"\033[1m"+"What is the Percentage of loyal customers vs␣
↪churn customers?"+ "\033[0m"+"*************** \n\n")
```

```python
ExitedValues = ccpb_df.Exited.value_counts()
labels = ["Loyal Customer","Churn Customers"]
colors = ['#A9D02F','#EA2015']
fig1, f1 = plt.subplots()
f1.pie(ExitedValues,labels=labels, colors = colors, autopct='%1.1f%%',shadow=True, startangle=60)
f1.axis('equal')
plt.tight_layout()
plt.show()
```

```
*********What is the Percentage of loyal customers vs churn
customers?****************
```



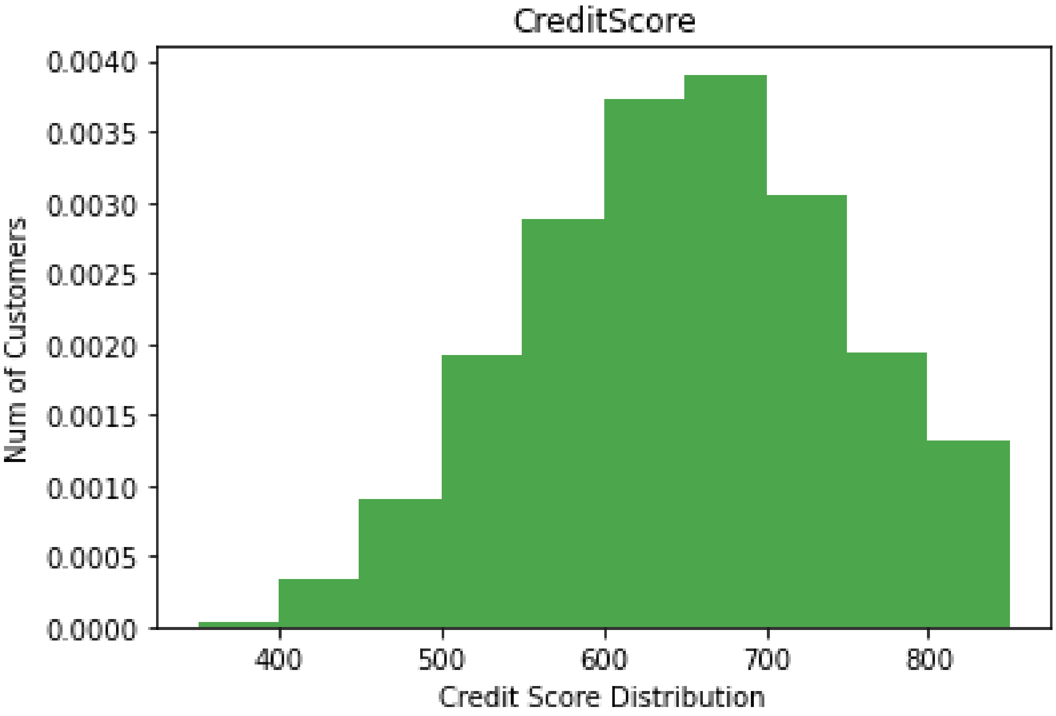
Observation: 79.6% customers are loyal and 20.4 % are not. Hence it is considered as biased distribution

```python
[22]: print ("\n\n*********"+"\033[1m"+"Customer's Credit Score Distribution"+ "\033[0m"+"**************** \n\n")
```

```
plt.hist(ccpb_df['CreditScore'],density = 1,
                             color ='green',
                             alpha = 0.7)
plt.xlabel('Credit Score Distribution')
plt.ylabel('Num of Customers')
plt.title('CreditScore')
```

```
*********Customer's Credit Score Distribution***************
```

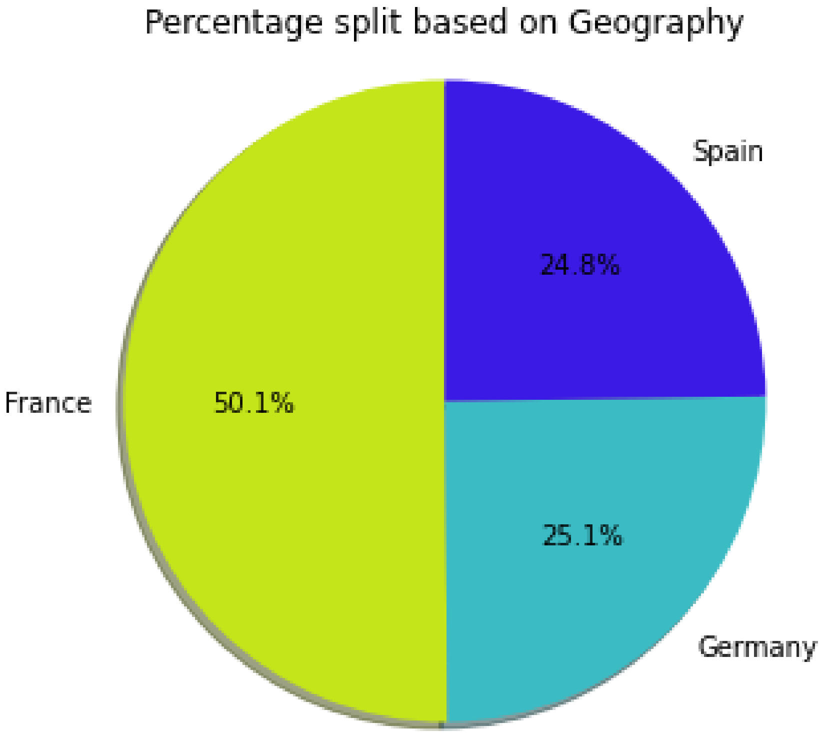
[22]: Text(0.5, 1.0, 'CreditScore')



[23]:
```
# this plot is to show how Geography play a role at the customer churn
print ("\n\n*********"+"\033[1m"+"Customer churn rate w.r.t geography"+␣
 ↪"\033[0m"+"*************** \n\n")
Geosplit = ccpb_df.Geography.value_counts()
Geovalues  = ccpb_df['Geography'].value_counts().values.tolist()
Geolabels  = ccpb_df['Geography'].value_counts().keys().tolist()
colors = ['#C4E51A', '#3BBCC4' , '#3B1AE5']
fig2, f2 = plt.subplots()
```

```
f2.pie(Geovalues,labels=Geolabels, colors = colors, autopct='%1.
↪1f%%',shadow=True, startangle=90)
# Equal aspect ratio ensures that pie is drawn as a circle
f2.axis('equal')
plt.tight_layout()
plt.title('Percentage split based on Geography')
plt.show()
```

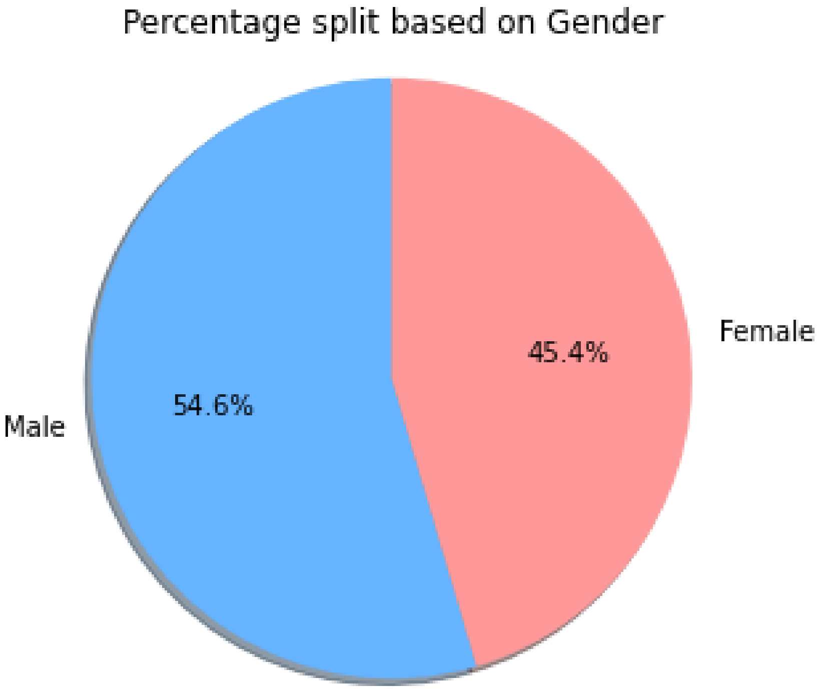
```
*********Customer churn rate w.r.t geography****************
```



```
[24]: #  Analyze gender role in terms of customer churn.
print ("\n\n*********"+"\033[1m"+"Customer churn rate w.r.t Gender"+␣
↪"\033[0m"+"**************** \n\n")
Gendervalues  = ccpb_df['Gender'].value_counts().values.tolist()
GenderLabels = ccpb_df['Gender'].value_counts().keys().tolist()
colors = ['#66b3ff', '#ff9999']
fig3, f3 = plt.subplots()
```

```
f3.pie(Gendervalues,labels=GenderLabels, colors = colors, autopct='%1.
 ↪1f%%',shadow=True, startangle= 90)
# Equal aspect ratio ensures that pie is drawn as a circle
f3.axis('equal')
plt.title('Percentage split based on Gender')
plt.tight_layout()
plt.show()
```

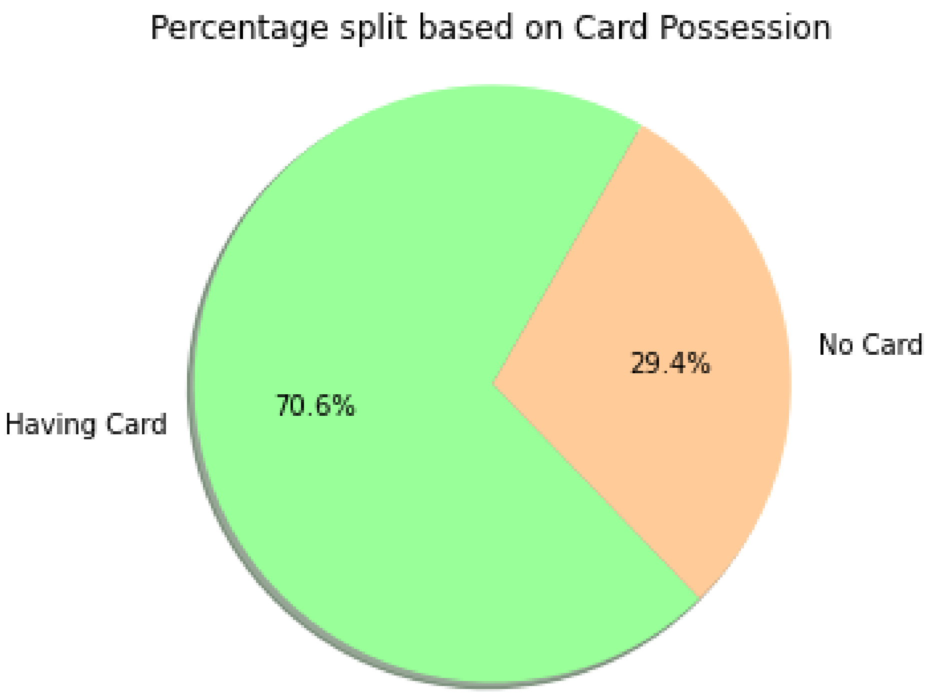
```
*********Customer churn rate w.r.t Gender****************
```



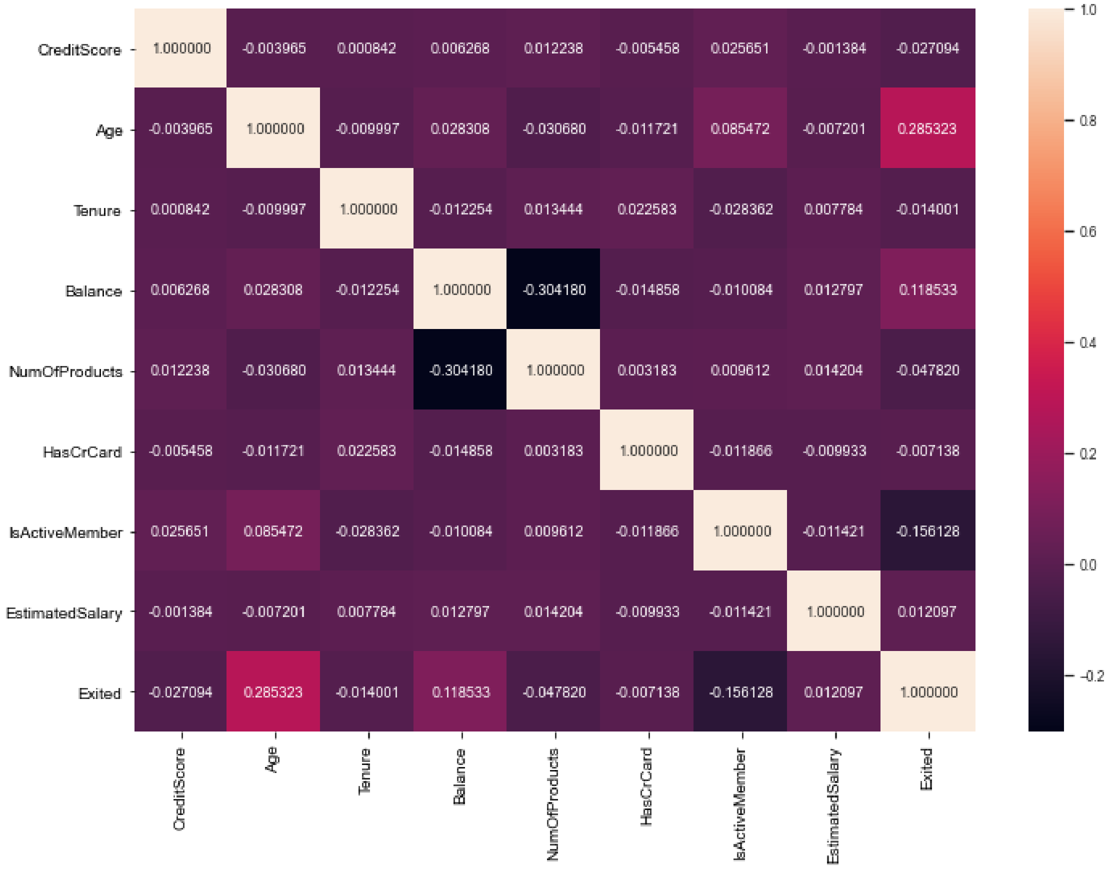
[25]:
```
# this plot is to show how HasCrCard play a role at the customer churn
print ("\n\n*********"+"\033[1m"+"Customer churn rate w.r.t CreditCard"+␣
 ↪"\033[0m"+"**************** \n\n")
HasCardvalues  = ccpb_df['HasCrCard'].value_counts().values.tolist()
HasCardlabels  = ["Having Card" , "No Card"]
colors = ['#99ff99','#ffcc99']
fig5, f5 = plt.subplots()
f5.pie(HasCardvalues ,labels=HasCardlabels, colors = colors,autopct='%1.
 ↪1f%%',shadow=True, startangle=60)
```

```
f5.axis('equal')
plt.title('Percentage split based on Card Possession')
plt.tight_layout()
plt.show()
```

```
*********Customer churn rate w.r.t CreditCard***************
```



```
[19]: print ("\n\n*********"+"\033[1m"+"Correlation Matrix"+␣
       ↪"\033[0m"+"*************** \n\n")
      fig, ax = plt.subplots()
      fig.set_size_inches(11.7, 8.27)
      sns.set(font_scale = 0.75)
      sns.heatmap(ccpb_df.corr(), annot = True, fmt = ".6f")
      plt.show()
```

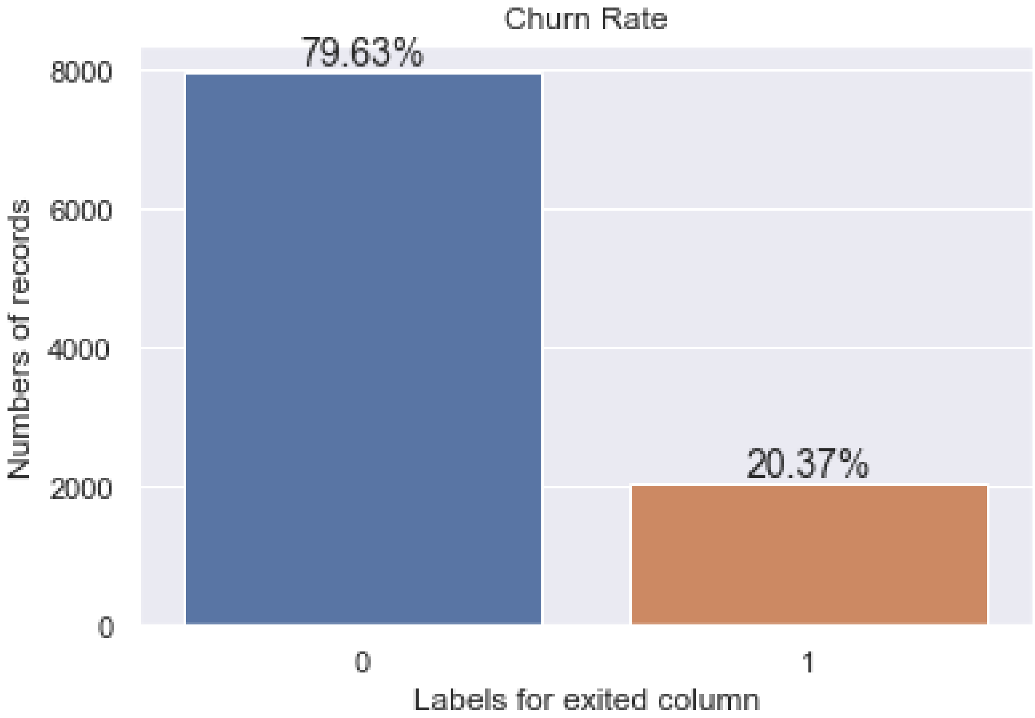


	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

```
[26]: # Graphical representation of the target label percentage before upsampling
print ("\n\n*********"+"\033[1m"+"Churn Rate w.r.t target label -Exited"+␣
 ↪"\033[0m"+"**************** \n\n")
total_len = len(ccpb_df['Exited'])
sns.set()
sns.countplot(ccpb_df.Exited).set_title('Churn Rate')
ax = plt.gca()
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x() + p.get_width()/2.,
            height + 2,
            '{:.2f}%'.format(100 * (height/total_len)),
            fontsize=14, ha='center', va='bottom')
sns.set(font_scale=1.5)
ax.set_xlabel("Labels for exited column")
ax.set_ylabel("Numbers of records")
plt.show()
```

```
*********Churn Rate w.r.t target label -Exited***************
```

```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

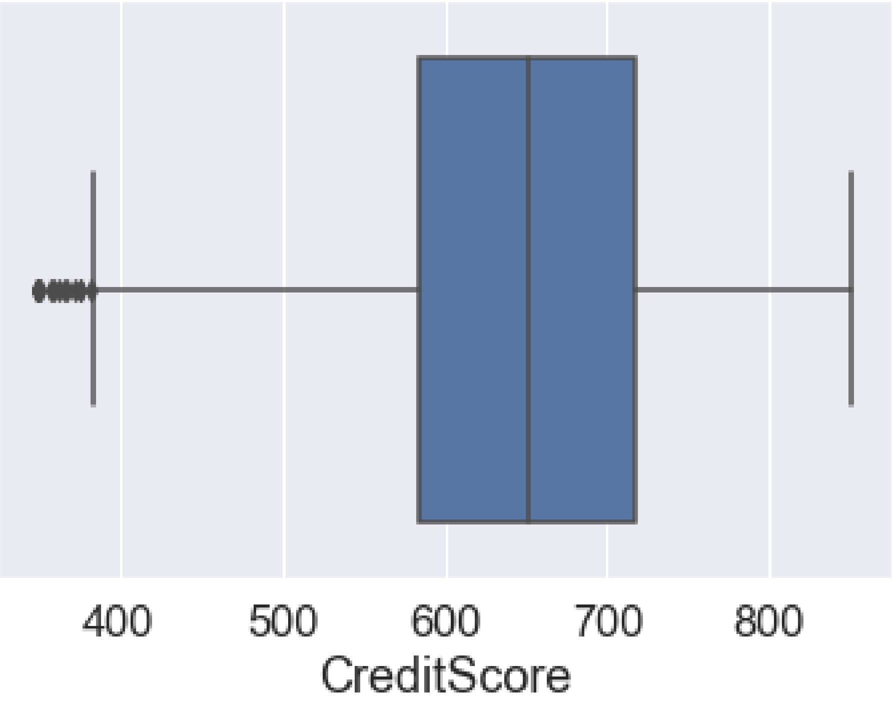


2 Visualizing outliers

[29]:
```
list_order = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts',␣
 ↪'EstimatedSalary']

def v_outliers(var):

    sns.boxplot(ccpb_df[var])
    plt.show()

for i in list_order:
    print ("\n\n*********"+"\033[1m"+"Outlier w.r.t "+ i +␣
↪"\033[0m"+"**************** \n\n")
    v_outliers(i)
```

```
*********Outlier w.r.t CreditScore****************
```

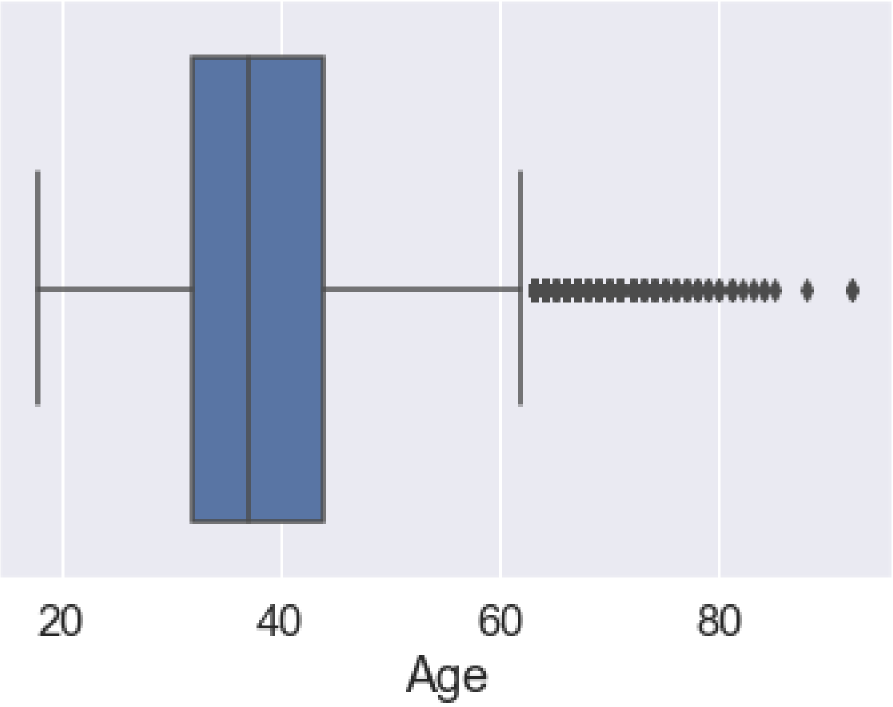
```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



```
*********Outlier w.r.t Age****************
```

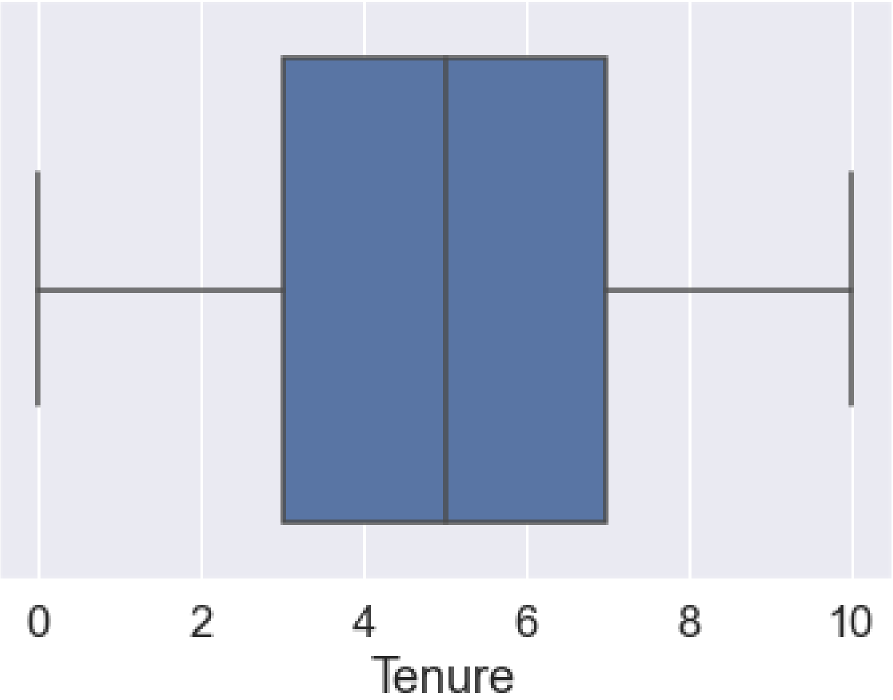
```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
```

```
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



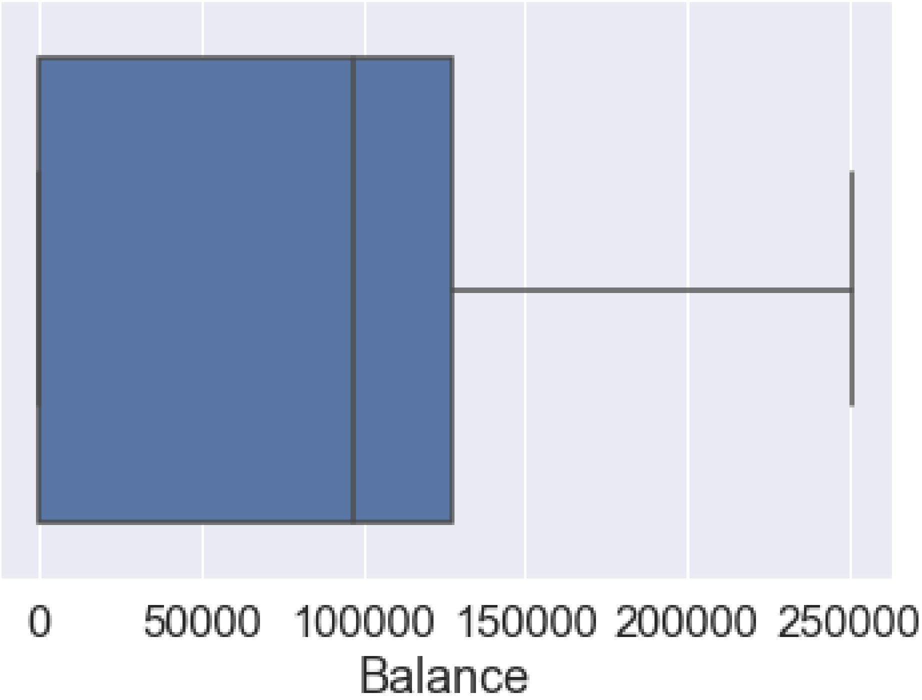
```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

```
*********Outlier w.r.t Tenure****************
```



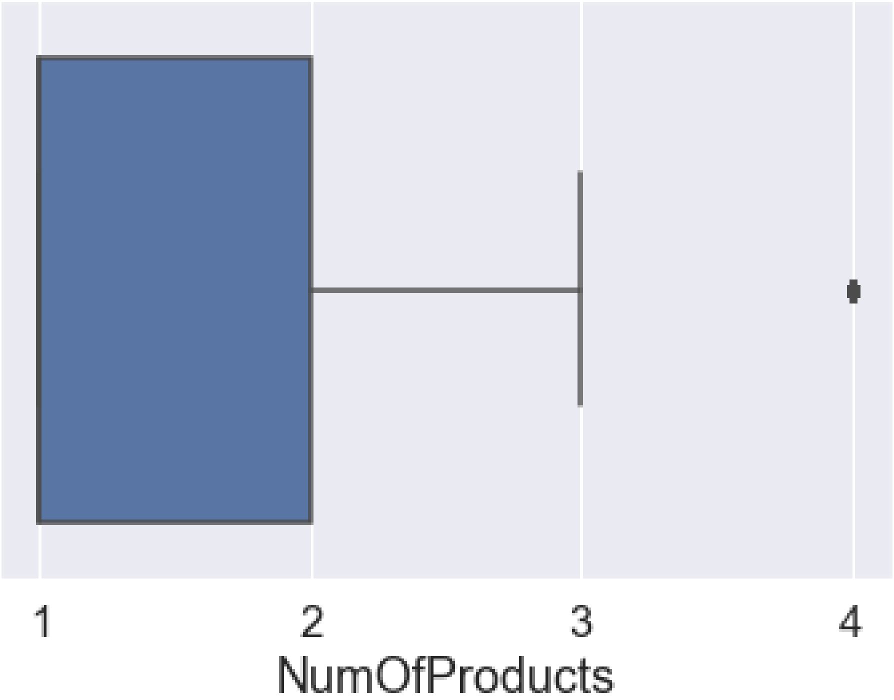
```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

```
*********Outlier w.r.t Balance****************
```



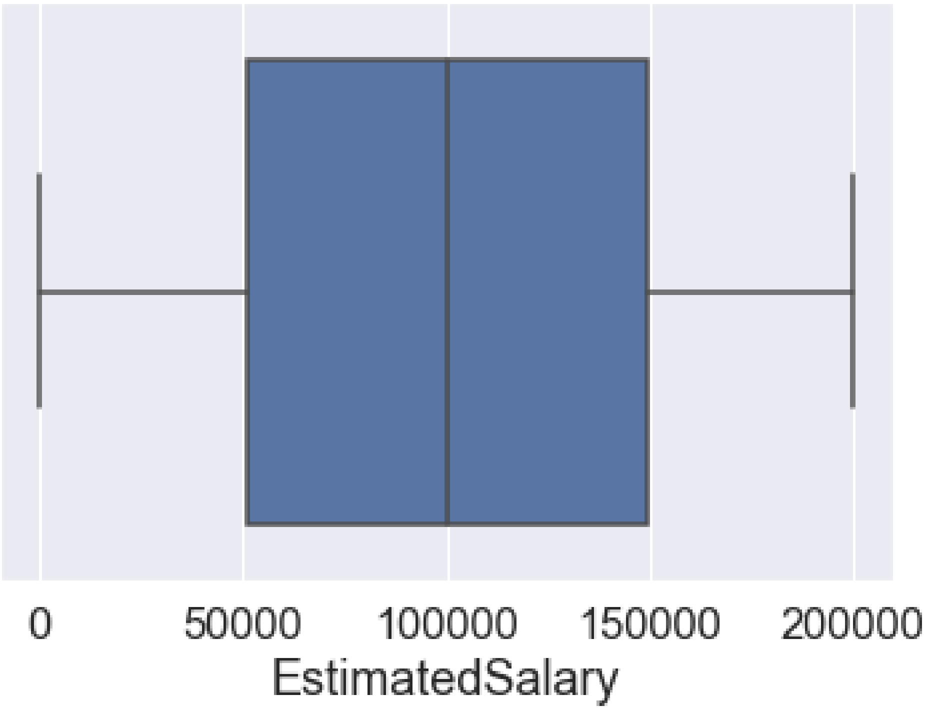
```
*********Outlier w.r.t NumOfProducts****************
```

```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



```
C:\Users\aditya.sumbaraju\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

```
*********Outlier w.r.t EstimatedSalary****************
```



Observations: Seems like CreditScore, Age, NumOfProducts have outliers

```
[27]: outliers = ['Age','CreditScore','NumOfProducts']
```

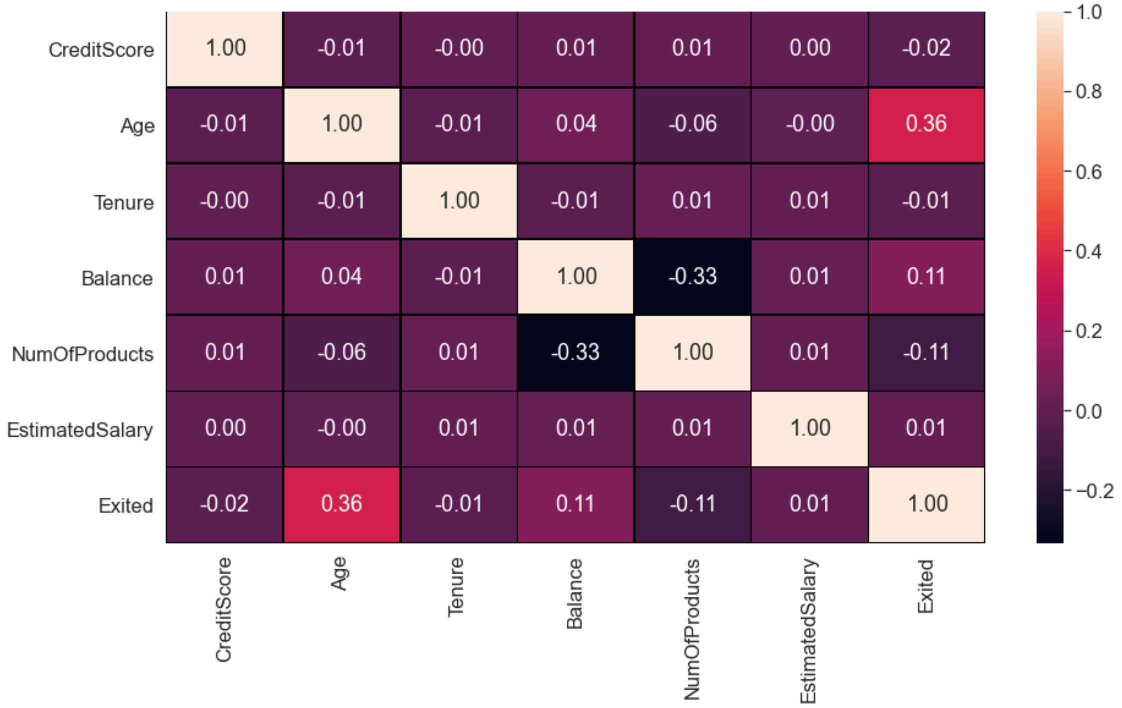
```
[32]: # create a function to remove the outliers
      def rm_outlier(input_data,feature):
          qt1 = input_data[feature].quantile(0.25)
          qt3 = input_data[feature].quantile(0.75)
          iqr = qt3 - qt1
          point_low = qt1 - 1.5 * iqr
          point_high = qt3 + 1.5 * iqr
          cleaned_df = input_data.loc[(input_data[feature] >  point_low) &␣
        ↪(input_data[feature] <  point_high)]
          return cleaned_df
```

```
[33]: # clean the dataset by removing outliers
      ccpb_df_cleaned =␣
        ↪rm_outlier(rm_outlier(rm_outlier(ccpb_df,'Age'),'CreditScore'),'NumOfProducts')

      print(ccpb_df.shape)
      print(ccpb_df_cleaned.shape)
```

```
(10000, 11)
(9516, 11)
```

```
[34]: # CORRELATION MATRIX OF THE DATA
      plt.figure(figsize = (15,8))
      list_corr = ['CreditScore' ,'Age' ,'Tenure' ,'Balance' ,'NumOfProducts'␣
       ↪,'EstimatedSalary' ,'Exited']
      sns.heatmap(ccpb_df_cleaned[list_corr].corr(), annot = True, linecolor =␣
       ↪"black", lw = 0.5, fmt= '.2f')
```

```
[34]: <AxesSubplot:>
```



```
[36]: ccpb_df_cleaned.groupby(ccpb_df_cleaned["Exited"])["Age"].mean()
```

```
[36]: Exited
      0    36.089197
      1    43.793583
      Name: Age, dtype: float64
```

Observation: as the age of the customer increases, the customer losing rate increases.

Average age of customers who did not exit the bank: 36

Average age of customers exit bank : 43

```
[ ]:
```