Task 1 Download the dataset

The Churn_Modelling.csv dataset is downloaded

Task 2 Load the Dataset

```
import pandas as pd
import numpy as np

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

df
```

```
      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
...         ...         ...        ...          ...       ...     ...
...
9995       9996    15606229   Obijiaku          771    France    Male
39
9996       9997    15569892  Johnstone          516    France    Male
35
9997       9998    15584532        Liu          709    France  Female
36
9998       9999    15682355  Sabbatini          772   Germany    Male
42
9999      10000    15628319     Walker          792    France  Female
28

      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
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
```

```
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]
```

Task 3 Perform Visualizations

Univariate Analysis

```
df['Age'].mean()
```

```
38.9218
```

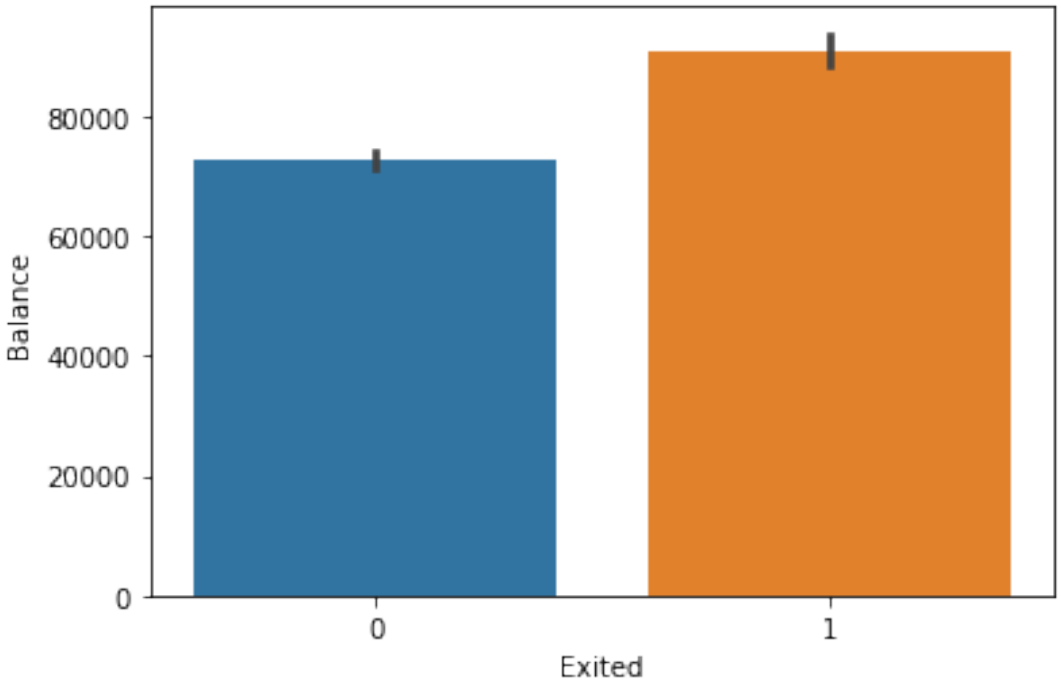
```
df['Balance'].median()
```

```
97198.54000000001
```

Bivariate Analysis

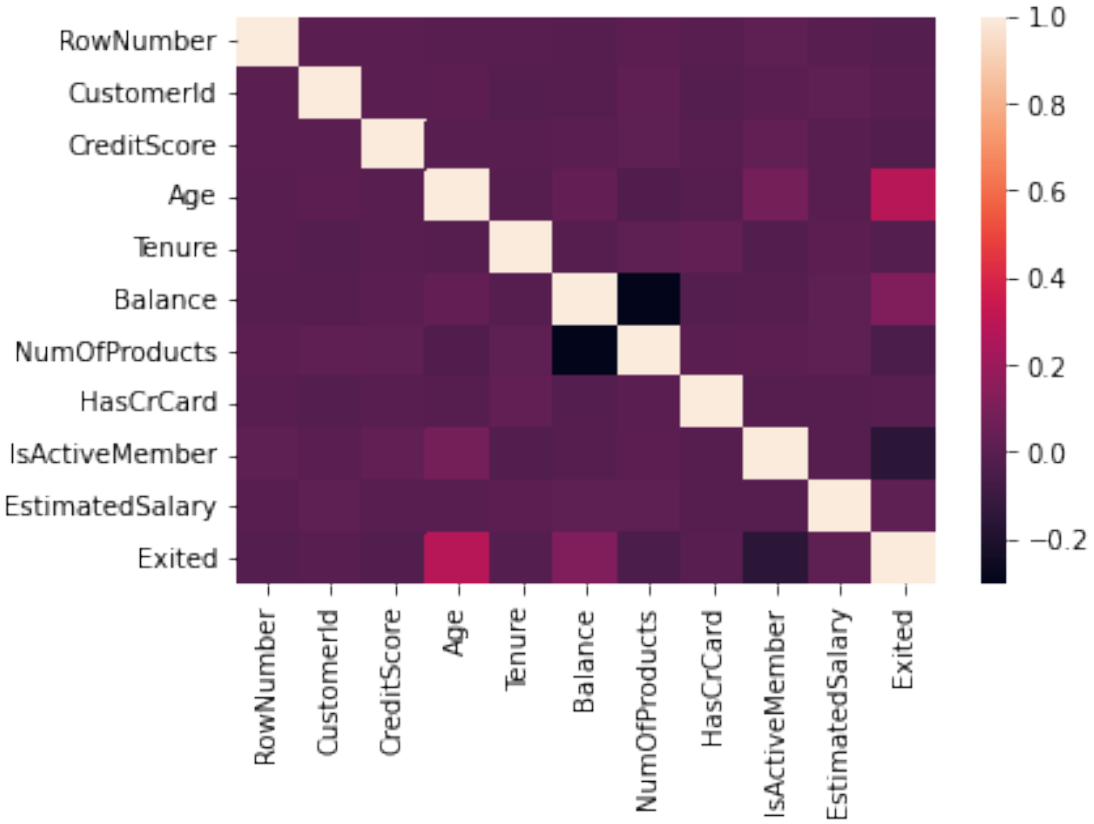
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.barplot(x = df['Exited'] , y = df['Balance']);
```



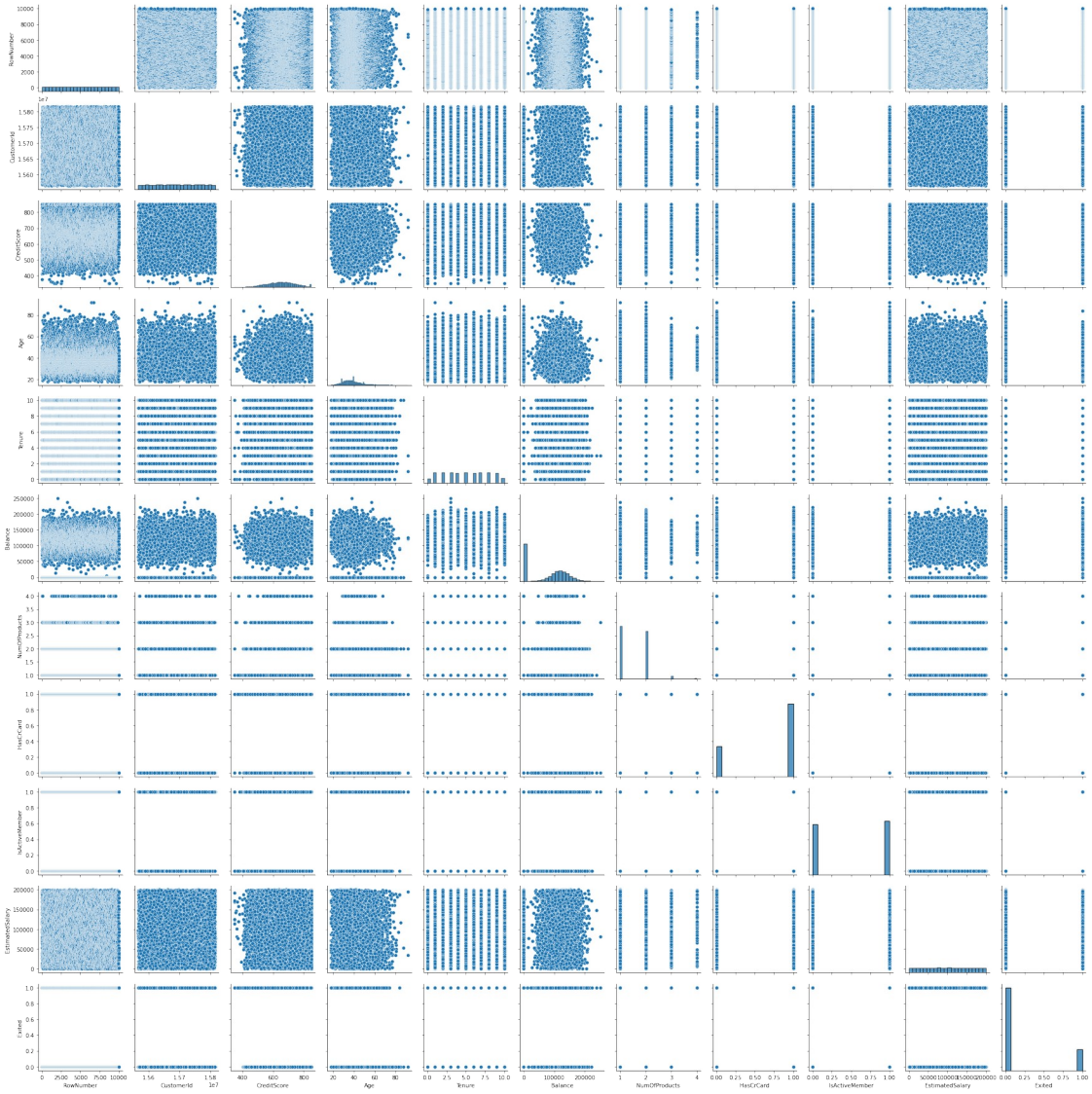
Multi Variate Analysis

```
sns.heatmap(df.corr());
```



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f544f833c50>
```



Task 4 Perform descriptive statistics on the dataset

```
df.describe()
```

```
         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

Task 5 Handle the Missing values

```
df.isnull().sum()
```

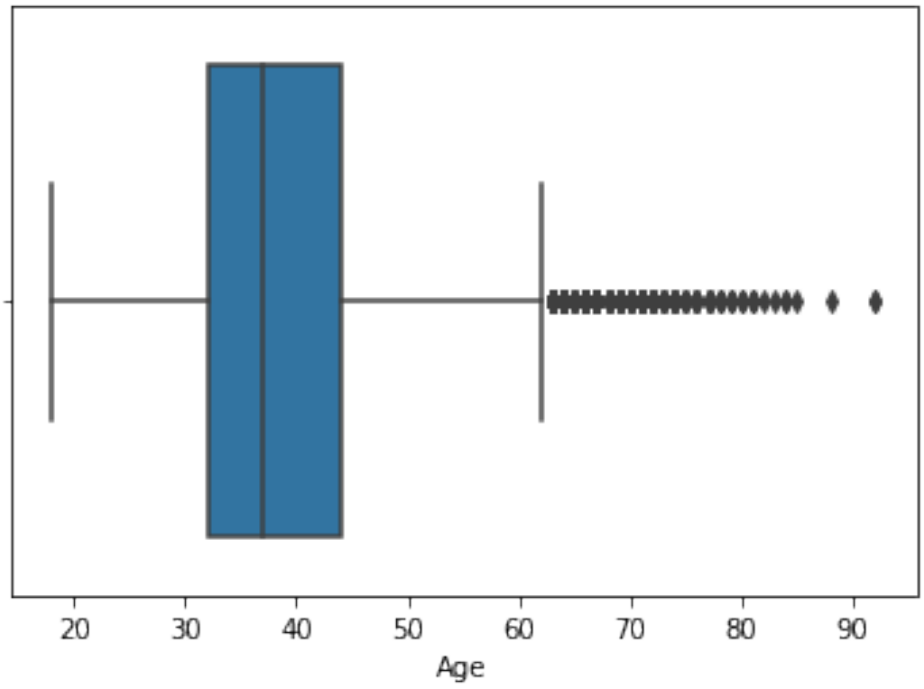
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: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

There are no null values present in the given dataset

Task 6 Find the outliers and replace the outliers

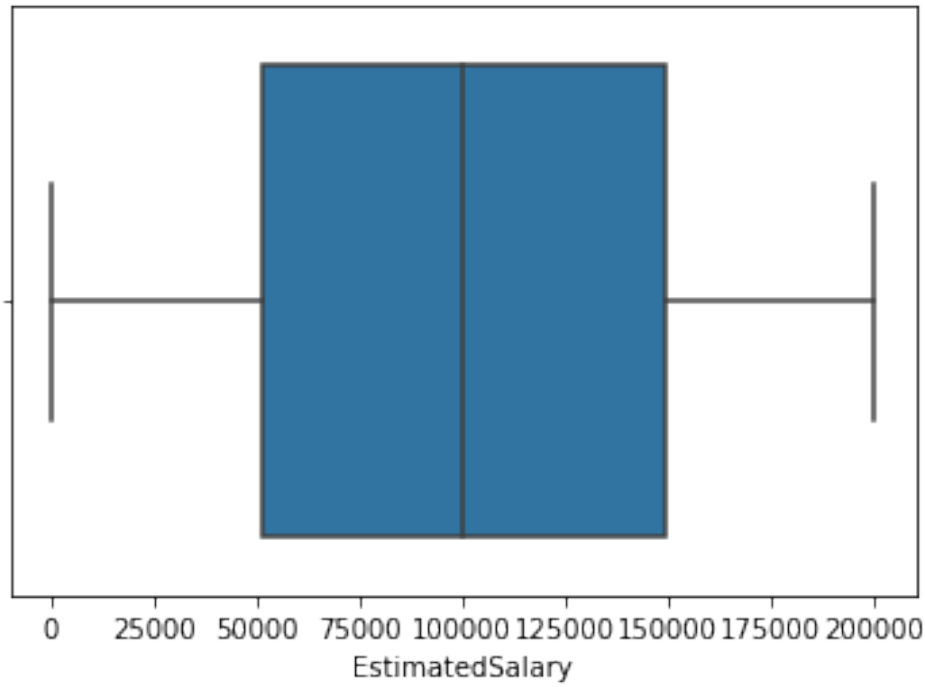
```
sns.boxplot(df['Age']);
```

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



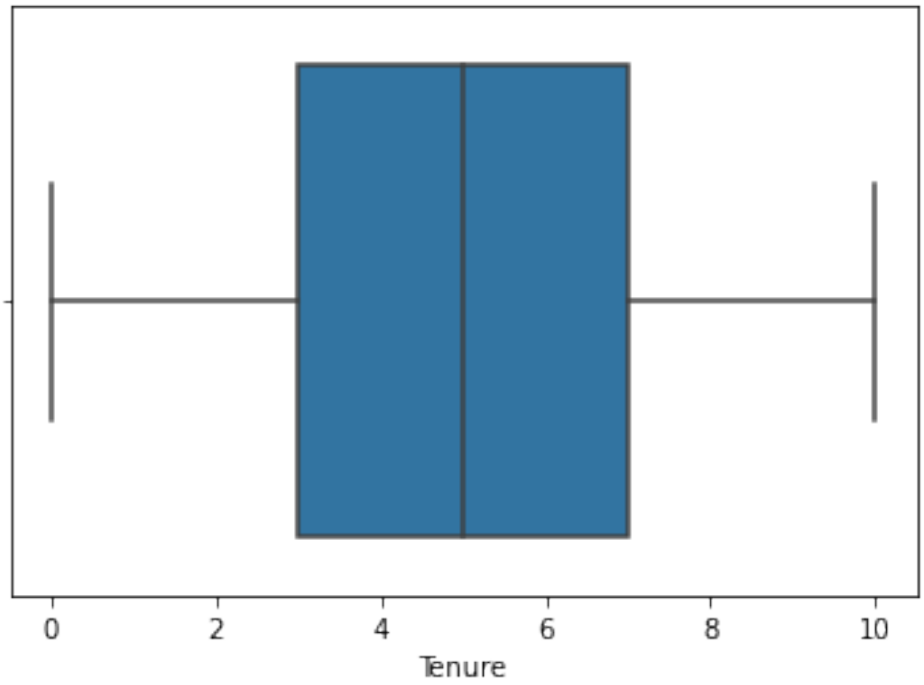

```
sns.boxplot(df['EstimatedSalary']);
```

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



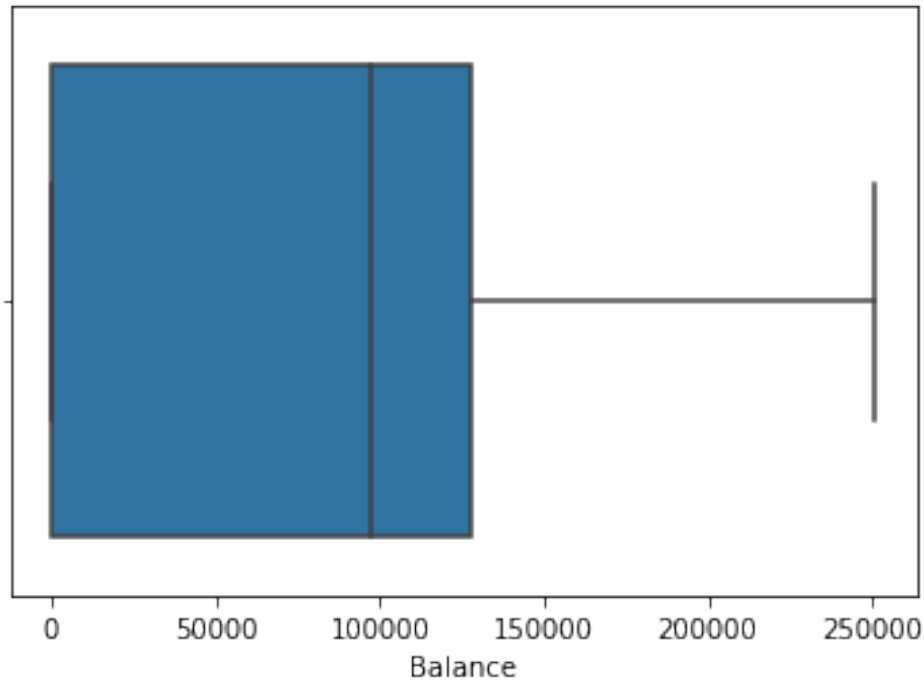

```
sns.boxplot(df['Tenure']);
```

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



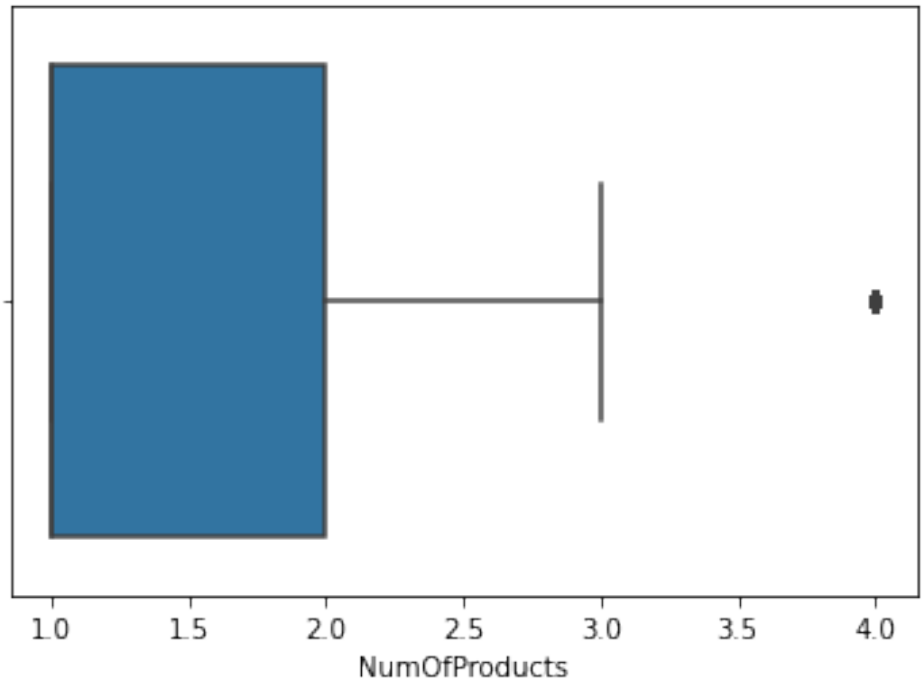

```
sns.boxplot(df['Balance']);
```

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




```
sns.boxplot(df['NumOfProducts']);
```

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




```
outliers=[]
def detect_outliers(data):
  threshold=3
  mean = np.mean(data)
  std =np.std(data)
  for i in data:
    z_score= (i - mean)/std
    if np.abs(z_score) > threshold:
      outliers.append(y)
      return outliers
outlier_pt=detect_outliers(df)
outlier_pt
```

Task 7 Check for Categorical columns and perform encoding

```
df.info() #There are few columns that are in object data type instead
of int64 or float 64 these columns are called Categorial Columns
```

```
<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
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
df['Geography'] = le.fit_transform(df['Geography'])
```

```
df['Gender'] = le.fit_transform(df['Gender'])
```

```
df
```

```
      RowNumber  CustomerId    Surname  CreditScore  Geography  Gender
Age  \
0             1    15634602   Hargrave          619          0       0
42
1             2    15647311       Hill          608          2       0
41
2             3    15619304       Onio          502          0       0
42
3             4    15701354       Boni          699          0       0
39
4             5    15737888   Mitchell          850          2       0
43
...         ...         ...        ...          ...        ...     ...
...
9995       9996    15606229   Obijiaku          771          0       1
39
9996       9997    15569892  Johnstone          516          0       1
35
9997       9998    15584532        Liu          709          0       0
36
9998       9999    15682355  Sabbatini          772          1       1
42
9999      10000    15628319     Walker          792          0       0
28

      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
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]
```

Task 8 and Task 10 Split the data into dependent and independent variables

```
df.drop(columns = ['RowNumber'])
```

```
      CustomerId    Surname  CreditScore  Geography  Gender  Age
Tenure  \
0       15634602   Hargrave          619          0       0   42
2
1       15647311       Hill          608          2       0   41
1
2       15619304       Onio          502          0       0   42
8
3       15701354       Boni          699          0       0   39
1
4       15737888   Mitchell          850          2       0   43
2
...          ...        ...          ...        ...     ...  ...     .
..
9995    15606229   Obijiaku          771          0       1   39
5
9996    15569892  Johnstone          516          0       1   35
10
9997    15584532        Liu          709          0       0   36
7
```

```
9998    15682355  Sabbatini          772          1       1   42
3
9999    15628319     Walker          792          0       0   28
4

        Balance  NumOfProducts  HasCrCard  IsActiveMember
EstimatedSalary  \
0          0.00              1          1               1
101348.88
1      83807.86              1          0               1
112542.58
2     159660.80              3          1               0
113931.57
3          0.00              2          0               0
93826.63
4     125510.82              1          1               1
79084.10
...         ...            ...        ...             ...
...
9995       0.00              2          1               0
96270.64
9996   57369.61              1          1               1
101699.77
9997       0.00              1          0               1
42085.58
9998   75075.31              2          1               0
92888.52
9999  130142.79              1          1               0
38190.78

      Exited
0          1
1          0
2          1
3          0
4          0
...      ...
9995       0
9996       0
9997       1
9998       1
9999       0

[10000 rows x 13 columns]
```

```
x = df.iloc[: , 0:13].values
y = df.iloc[: , 13:14].values
```

```
from sklearn.model_selection import train_test_split
```

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

xtrain.shape , xtest.shape

((7000, 13), (3000, 13))
```

#Task 9 Scale the independent variables

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

n = MinMaxScaler()
s = StandardScaler()

x = df[['Age', 'Tenure']].values
y = df['Gender'].values

n_xtrain = n.fit_transform(xtrain)

n_xtest = n.fit_transform(xtest)
```