### 01. Impot libraries

In [63]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

### 02. Upload database

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

### Read the Dataset

In [3]:
```
mydata = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Churn_Modelling.csv'
```

In [8]:
```
mydata.shape
```

Out[8]:
```
(10000, 14)
```

In [6]:
```
mydata.head()
```

Out[6]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balanc |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.0 |
| **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 | 0.0 |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.8 |

In [11]:
```
mydata.columns
```

Out[11]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [12]:
```
mydata.tail()
```

Out[12]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Ba |
|---|---|---|---|---|---|---|---|---|---|
| **9995** | 9996 | 15606229 | Obijiaku | 771 | France | Male | 39 | 5 | |
| **9996** | 9997 | 15569892 | Johnstone | 516 | France | Male | 35 | 10 | 57 |
| **9997** | 9998 | 15584532 | Liu | 709 | France | Female | 36 | 7 | |

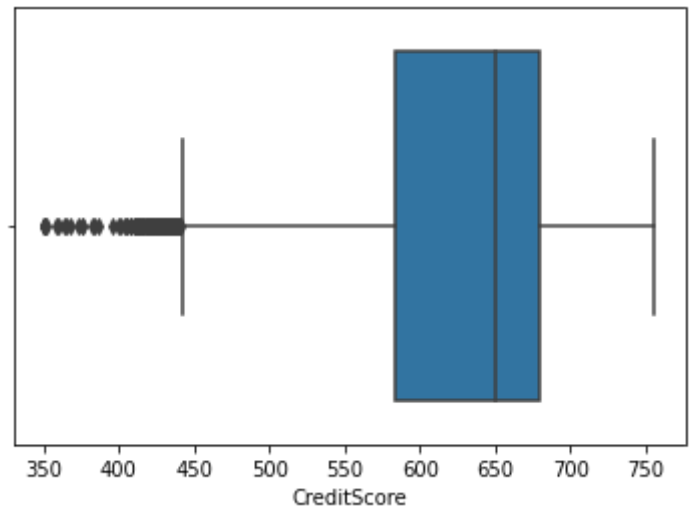
| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Ba |
|---|---|---|---|---|---|---|---|---|---|
| **9998** | 9999 | 15682355 | Sabbatini | 772 | Germany | Male | 42 | 3 | 75( |
| **9999** | 10000 | 15628319 | Walker | 792 | France | Female | 28 | 4 | 130 |

### 03. Perform Visualizations

In [99]:
```
sns.boxplot(mydata['CreditScore'])
```

```
/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 vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

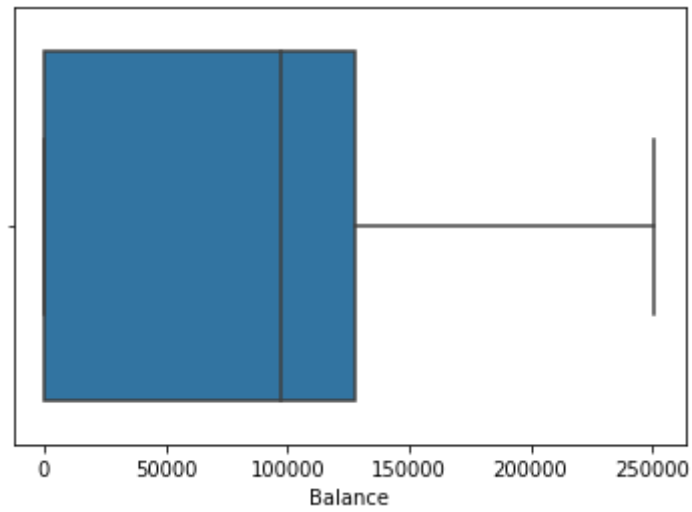
Out[99]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c25826090>
```




In [102…
```
sns.boxplot(mydata['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 vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

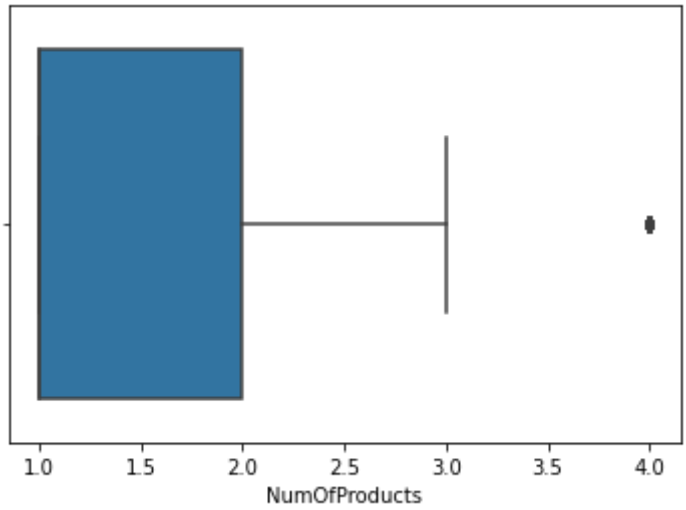
Out[102…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c25724310>
```



In [103…
```
sns.boxplot(mydata['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 vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

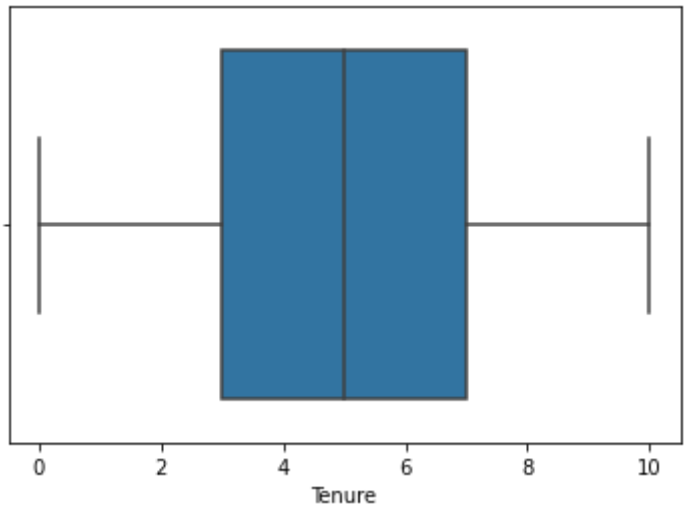
Out[103…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c2570ed10>
```




In [104…
```
sns.boxplot(mydata['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 vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

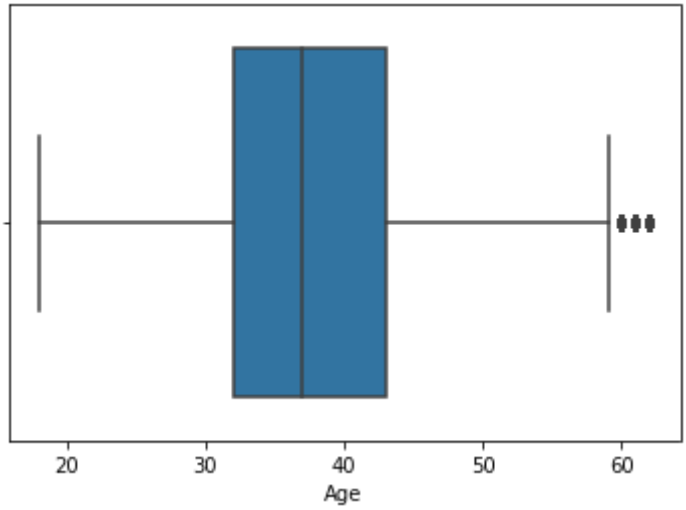
Out[104…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c25687550>
```




In [105…
```
sns.boxplot(mydata['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 vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

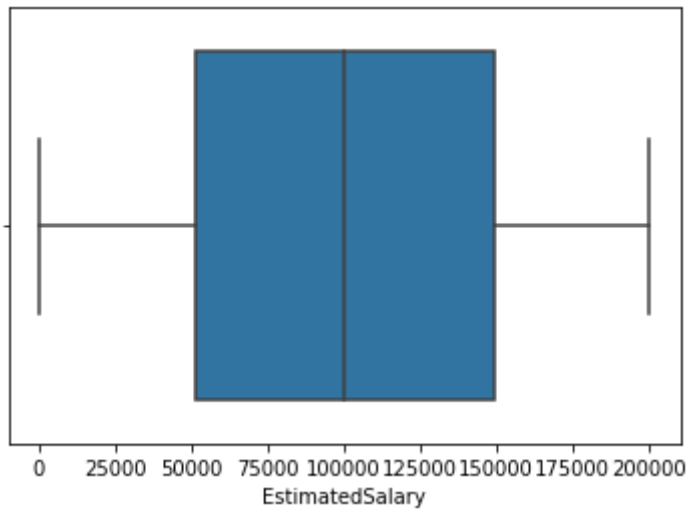
Out[105…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c255fb190>
```




In [106…
```
sns.boxplot(mydata['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 vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

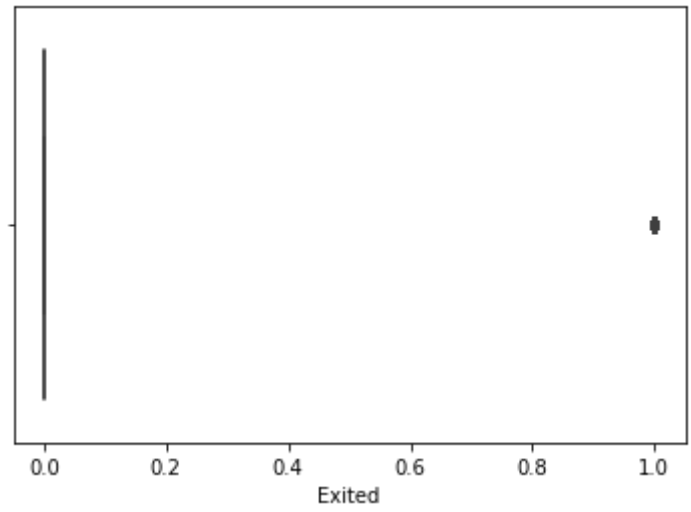
Out[106…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c255e2450>
```




In [107…
```
sns.boxplot(mydata['Exited'])
```

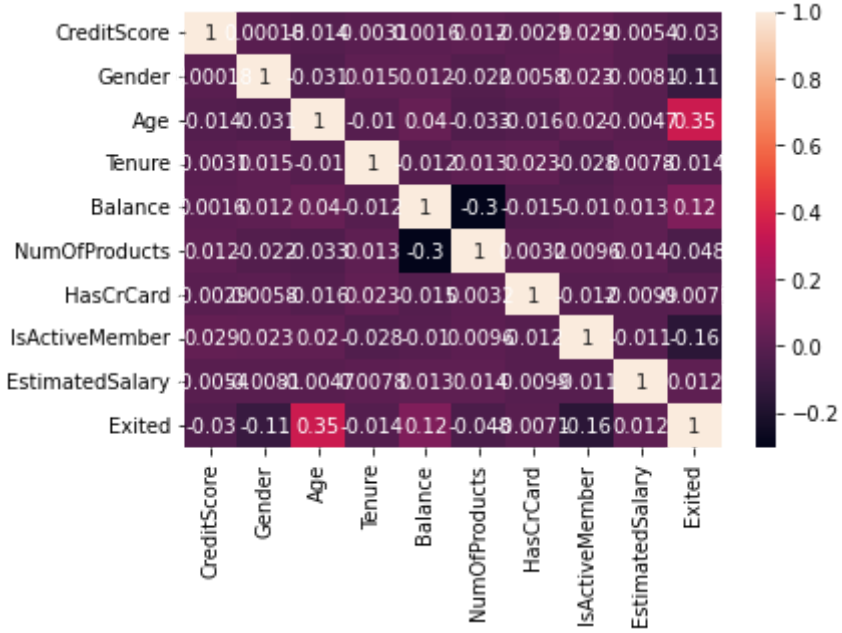
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only vali
d positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[107…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c25543890>
```



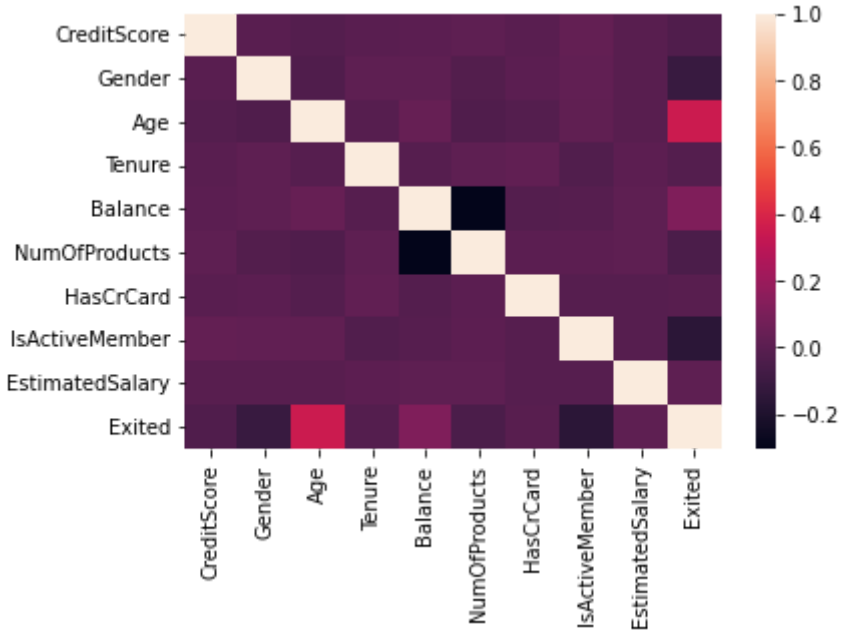

In [108…
```
sns.heatmap(mydata.corr(),annot=True)
```

Out[108…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c25443f10>
```




In [109…
```
sns.heatmap(mydata.corr(),annot=False)
```

Out[109…
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c25443750>
```




### 05. Handling the Missing Values

In [16]:
```
mydata.duplicated().sum()
```

Out[16]:
```
0
```

In [17]:
```
mydata.isna().sum()
```

Out[17]:
```
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
```

In [18]:
```
mydata.nunique()
```

Out[18]:
```
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
```

In [19]:
```
mydata.info()
```

```
<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
```

In [25]:
```
mydata.drop(columns=['Gender','HasCrCard','IsActiveMember','Exited']).describe()
```

Out[25]:

| | RowNumber | CustomerId | CreditScore | Age | Tenure | Balance | Nu |
|---|---|---|---|---|---|---|---|
| **count** | 10000.00000 | 1.000000e+04 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | |
| **mean** | 5000.50000 | 1.569094e+07 | 650.528800 | 38.921800 | 5.012800 | 76485.889288 | |
| **std** | 2886.89568 | 7.193619e+04 | 96.653299 | 10.487806 | 2.892174 | 62397.405202 | |
| **min** | 1.00000 | 1.556570e+07 | 350.000000 | 18.000000 | 0.000000 | 0.000000 | |
| **25%** | 2500.75000 | 1.562853e+07 | 584.000000 | 32.000000 | 3.000000 | 0.000000 | |
| **50%** | 5000.50000 | 1.569074e+07 | 652.000000 | 37.000000 | 5.000000 | 97198.540000 | |
| **75%** | 7500.25000 | 1.575323e+07 | 718.000000 | 44.000000 | 7.000000 | 127644.240000 | |
| **max** | 10000.00000 | 1.581569e+07 | 850.000000 | 92.000000 | 10.000000 | 250898.090000 | |

In [26]:
```
qnt=mydata.drop(columns=['Gender','Tenure','HasCrCard','IsActiveMember'])
```

### 06. Find Outliers

In [110…
```
qnt=mydata.drop(columns=['Gender','Tenure','HasCrCard','IsActiveMember','NumOfPrc
qnt
```

Out[110…

| | CreditScore | Age | Balance | EstimatedSalary |
|---|---|---|---|---|
| **0.25** | 584.0 | 32.0 | 0.0000 | 51002.1100 |
| **0.85** | 705.0 | 47.0 | 140895.0965 | 170322.3935 |

In [111…
```
Q1=qnt.iloc[0]
Q4=qnt.iloc[1]
```

In [112…
```
iqr=Q4-Q1
iqr
```

Out[112…
```
CreditScore          121.0000
Age                   15.0000
Balance            140895.0965
EstimatedSalary    119320.2835
dtype: float64
```

In [113…
```
upper=qnt.iloc[1]+2.5*iqr
upper
```

Out[113…
```
CreditScore          1007.50000
Age                    84.50000
Balance            493132.83775
EstimatedSalary    468623.10225
dtype: float64
```

In [114…
```
lower=qnt.iloc[0]-2.5*iqr
lower
```

Out[114…
```
CreditScore          281.50000
Age                   -5.50000
Balance           -352237.74125
EstimatedSalary   -247298.59875
dtype: float64
```

### Replace Outliers

In [115…
```
mydata['CreditScore']= np.where(mydata['CreditScore']>756,650.5288,mydata['Credit
mydata['Age']= np.where(mydata['Age']>62, 38.9218,mydata['Age'])
```

### 07. Categorical Columns and Performing Encoding

In [132…
```
mydata['Gender'].replace({'Male': 1, 'Female':0}, inplace=True)
mydata.head(8)
```

Out[132…

| | CreditScore | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember |
|---|---|---|---|---|---|---|---|---|
| **0** | 619.0000 | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 |
| **1** | 608.0000 | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 |
| **2** | 502.0000 | 0 | 42.0 | 8 | 159660.80 | 3 | 1 | 0 |
| **3** | 699.0000 | 0 | 39.0 | 1 | 0.00 | 2 | 0 | 0 |
| **4** | 650.5288 | 0 | 43.0 | 2 | 125510.82 | 1 | 1 | 1 |
| **5** | 645.0000 | 1 | 44.0 | 8 | 113755.78 | 2 | 1 | 0 |
| **6** | 650.5288 | 1 | 50.0 | 7 | 0.00 | 2 | 1 | 1 |
| **7** | 376.0000 | 0 | 29.0 | 4 | 115046.74 | 4 | 1 | 0 |

### Dropping Unwanted Columns

In [144…
```
mydata =mydata.drop(columns=['Age'])
mydata.head()
```

Out[144…

| | Gender | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | E |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 | |
| **1** | 0 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 | |
| **2** | 0 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 | |
| **3** | 0 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 | |
| **4** | 0 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 | |

### 08. Split Data Into Dependent and Independent Variables

In [78]:
```
a=mydata.iloc[:,:-2]
a.head()
```

Out[78]:

| | CreditScore | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember |
|---|---|---|---|---|---|---|---|---|
| **0** | 619.0000 | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 |
| **1** | 608.0000 | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 |
| **2** | 502.0000 | 0 | 42.0 | 8 | 159660.80 | 3 | 1 | 0 |
| **3** | 699.0000 | 0 | 39.0 | 1 | 0.00 | 2 | 0 | 0 |
| **4** | 650.5288 | 0 | 43.0 | 2 | 125510.82 | 1 | 1 | 1 |

In [83]:
```
b=mydata.iloc[:,-2]
```

In [84]:
```
b.head()
```

Out[84]:
```
0    101348.88
1    112542.58
2    113931.57
3     93826.63
4     79084.10
Name: EstimatedSalary, dtype: float64
```

### 09. Scale The Independent Variables

In [85]:
```
from sklearn.preprocessing import StandardScaler
```

In [86]:
```
cls=StandardScaler()
a=cls.fit_transform(a)
```

In [130…
```
a
```

Out[130…
```
array([[-0.13284832, -1.09598752,  0.48205148, ..., -0.91158349,
         0.64609167,  0.97024255],
       [-0.28182929, -1.09598752,  0.36638802, ..., -0.91158349,
        -1.54776799,  0.97024255],
       [-1.71746409, -1.09598752,  0.48205148, ...,  2.52705662,
         0.64609167, -1.03067011],
       ...,
       [ 1.08608688, -1.09598752, -0.21192932, ..., -0.91158349,
        -1.54776799,  0.97024255],
       [ 0.29416906,  0.91241915,  0.48205148, ...,  0.80773656,
         0.64609167, -1.03067011],
       [ 0.29416906, -1.09598752, -1.13723705, ..., -0.91158349,
         0.64609167, -1.03067011]])
```

### 10.Split Data Into Training and Testing

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

In [92]:
```
a_train,a_test,b_train,b_test= train_test_split(a,b,test_size=0.4,random_state=0)
```

In [93]:
```
a_train.shape
```

Out[93]:
```
(6000, 8)
```

In [94]:
```
a_test.shape
```

Out[94]:
```
(4000, 8)
```

In [95]:
```
a_train
```

Out[95]:
```
array([[-0.67459731,  0.91241915,  0.59771495, ..., -0.91158349,
         0.64609167,  0.97024255],
       [ 0.31409458, -1.09598752,  0.25072455, ..., -0.91158349,
         0.64609167,  0.97024255],
       [ 0.31409458,  0.91241915, -0.09626585, ...,  0.80773656,
         0.64609167, -1.03067011],
       ...,
       [ 1.47885489,  0.91241915, -0.32759278, ...,  0.80773656,
         0.64609167, -1.03067011],
       [-0.52561634, -1.09598752,  0.01939762, ...,  0.80773656,
         0.64609167,  0.97024255],
       [-0.07867343, -1.09598752,  1.17603229, ..., -0.91158349,
         0.64609167, -1.03067011]])
```

In [96]:
```
a_test
```

Out[96]:
```
array([[-0.43081026, -1.09598752, -0.32759278, ..., -0.91158349,
         0.64609167,  0.97024255],
       [-1.43304588, -1.09598752,  0.25072455, ..., -0.91158349,
         0.64609167, -1.03067011],
       [ 1.04545571, -1.09598752,  0.48205148, ..., -0.91158349,
         0.64609167,  0.97024255],
       ...,
       [ 1.58720469, -1.09598752, -1.59989092, ..., -0.91158349,
        -1.54776799,  0.97024255],
       [ 0.84229984,  0.91241915,  0.01939762, ...,  0.80773656,
        -1.54776799, -1.03067011],
```

```
       [-1.39241471,  0.91241915, -0.90591012, ...,  0.80773656,
        -1.54776799, -1.03067011]])
```

In [131…
```
b_train
```

Out[131…
```
7809    198402.37
5279    124550.88
3279     68789.93
8984     42669.37
8466     83343.73
          ...
9225    162961.79
4859    107753.07
3264    181429.87
9845    148750.16
2732    118855.26
Name: EstimatedSalary, Length: 6000, dtype: float64
```

In [98]:
```
b_test
```

Out[98]:
```
9394    192852.67
898     128702.10
2398     75732.25
5906     89368.59
2343    135662.17
          ...
4758     48545.10
9914    180844.81
7067     94105.00
4578     44653.50
4202    100995.68
Name: EstimatedSalary, Length: 4000, dtype: float64
```

In [ ]:
```

```