```
#Load the data set
import pandas as pd
import numpy as np
import sklearn
```

In [3]:

```
data=pd.read_csv("/content/drive/MyDrive/Database/Database")
```

In [4]:

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

In [5]:

```
data.head()
```

Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

In [6]:

```
data.tail()
```

Out[6]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9999	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0

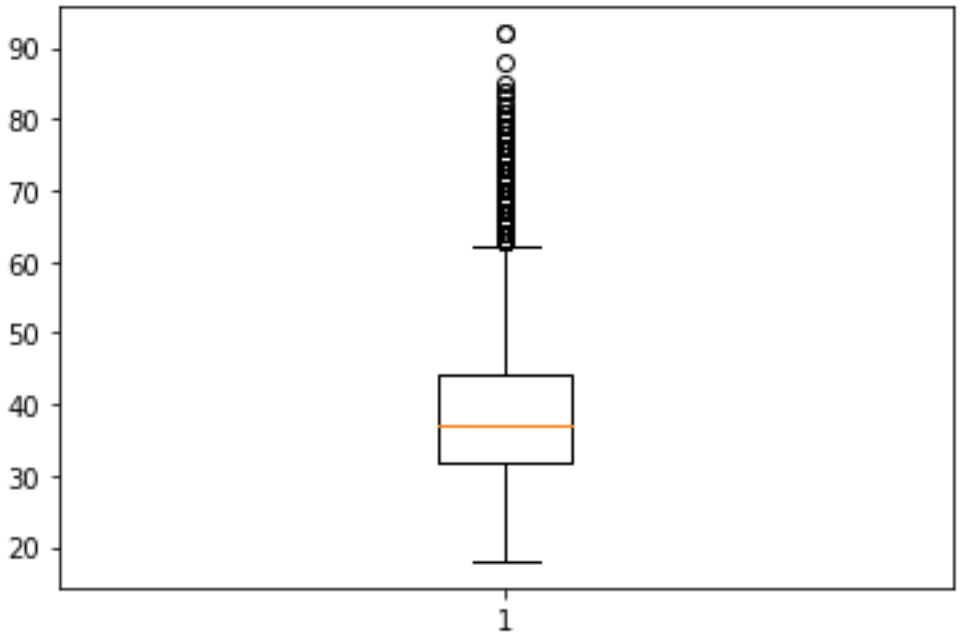
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
995														
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

In [7]:

```
plt.boxplot(data['Age'])
```

Out[7]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f11a7229350>,
  <matplotlib.lines.Line2D at 0x7f11a7229890>],
 'caps': [<matplotlib.lines.Line2D at 0x7f11a7229dd0>,
  <matplotlib.lines.Line2D at 0x7f11a7230350>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f11a7299cd0>],
 'medians': [<matplotlib.lines.Line2D at 0x7f11a7230650>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f11a728e490>],
 'means': []}
```



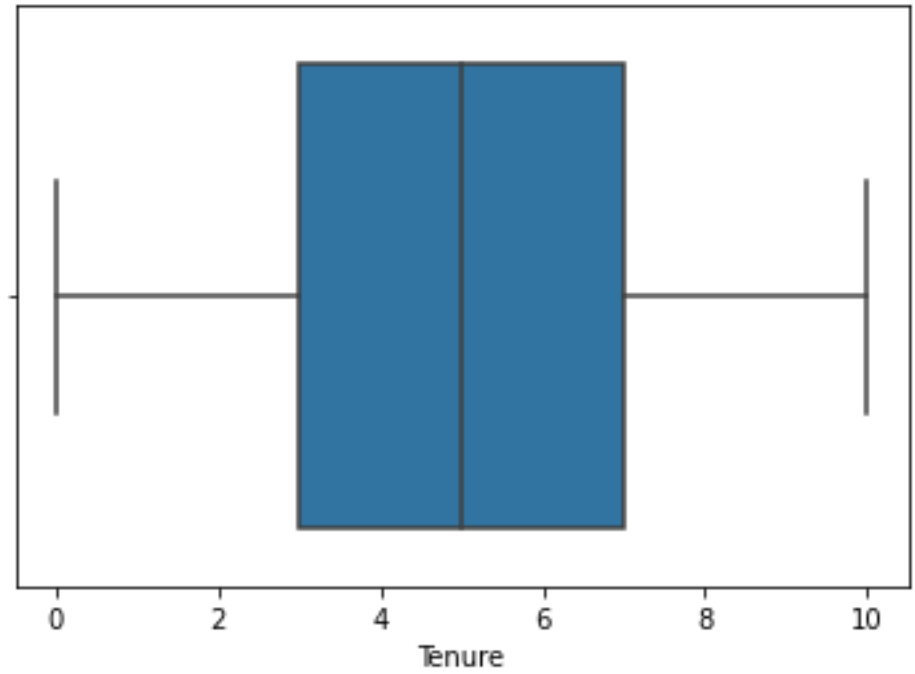
In [8]:

```
sns.boxplot(data['Tenure'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11a72ba290>
```

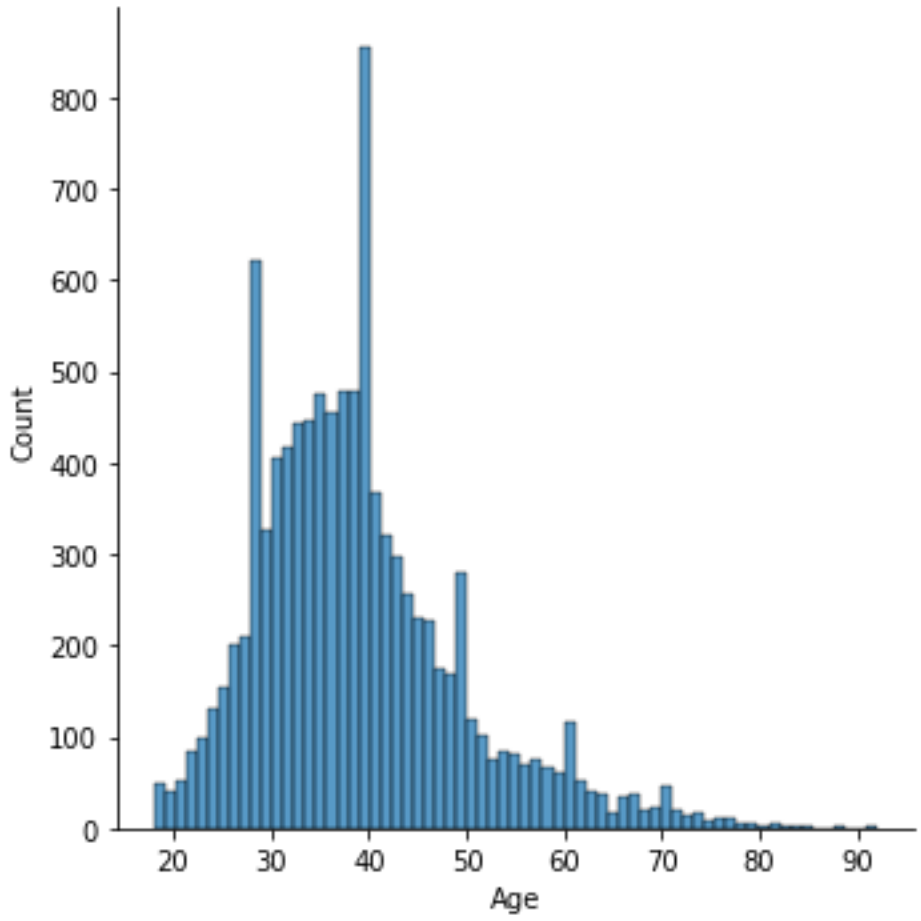


In [9]:

```
# Univariate Analysis
sns.displot(data['Age'])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x7f11a721a050>
```

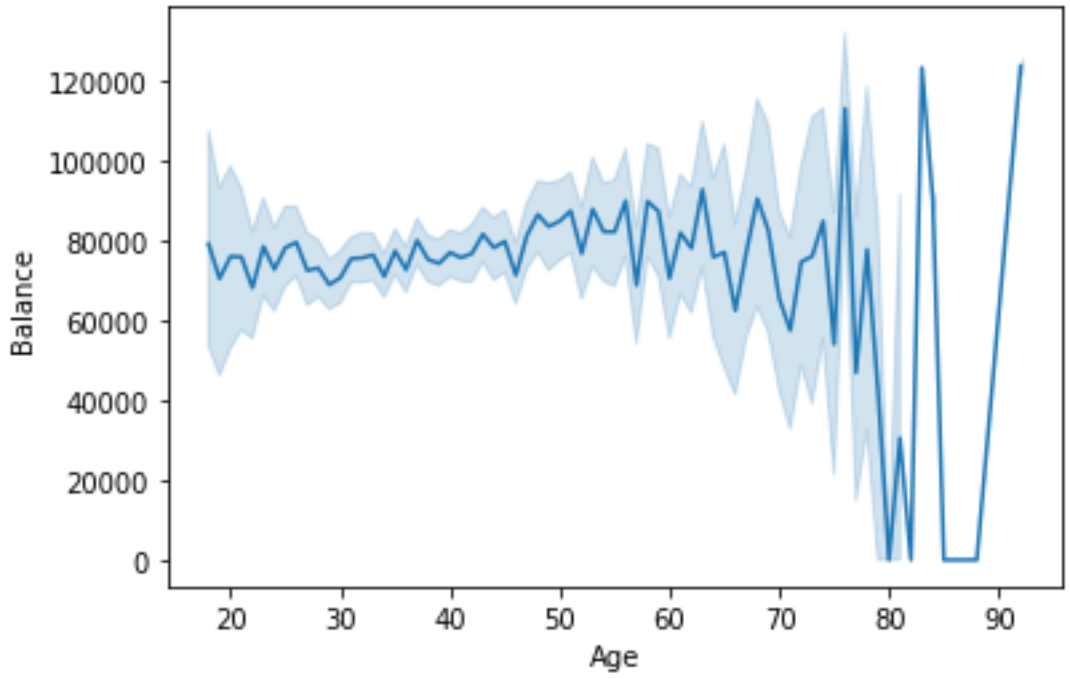


In [10]:

```
sns.lineplot(x="Age",y="Balance",data=data)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11a4355190>
```

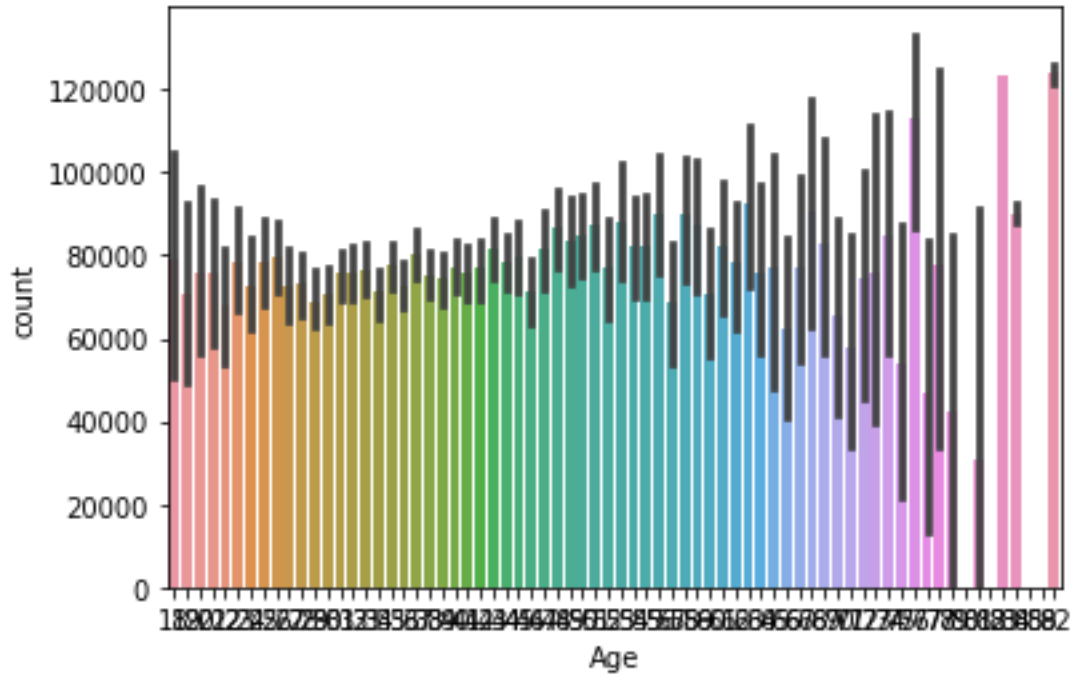


In [11]:

```
#Bi-Variate Analysis
sns.barplot(x='Age',y='Balance',data=data)
sns.countplot(x='Age',data=data)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11a723c1d0>
```



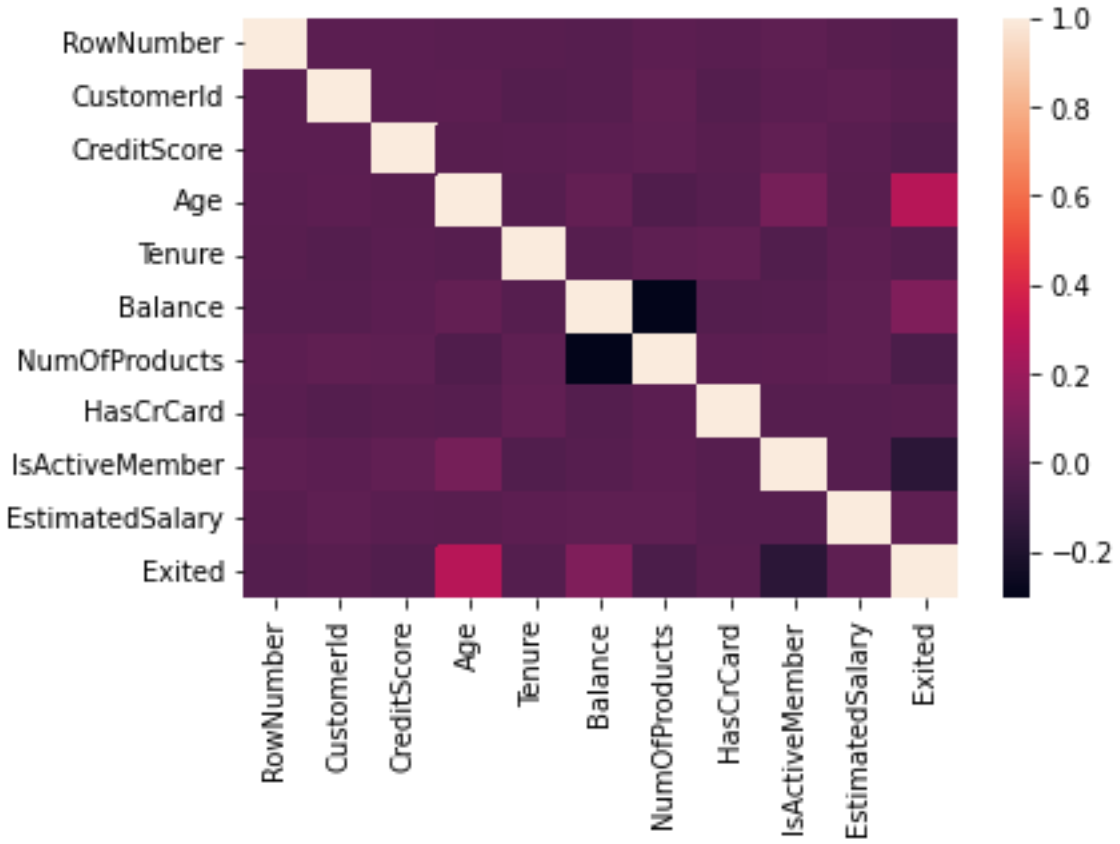
In [12]:

```
# multivariate analysis
corr_matrix=data.corr()
```

```
sns.heatmap(corr_matrix)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11a3f8fe90>
```



In [13]:

```
# Descriptive statistics
data.describe()
```

Out[13]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
min	1.000000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

In [14]:

```
# missing values
data.isnull().sum()
```

Out[14]:

```
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 [15]:

```
# outliers
import seaborn as sns
```

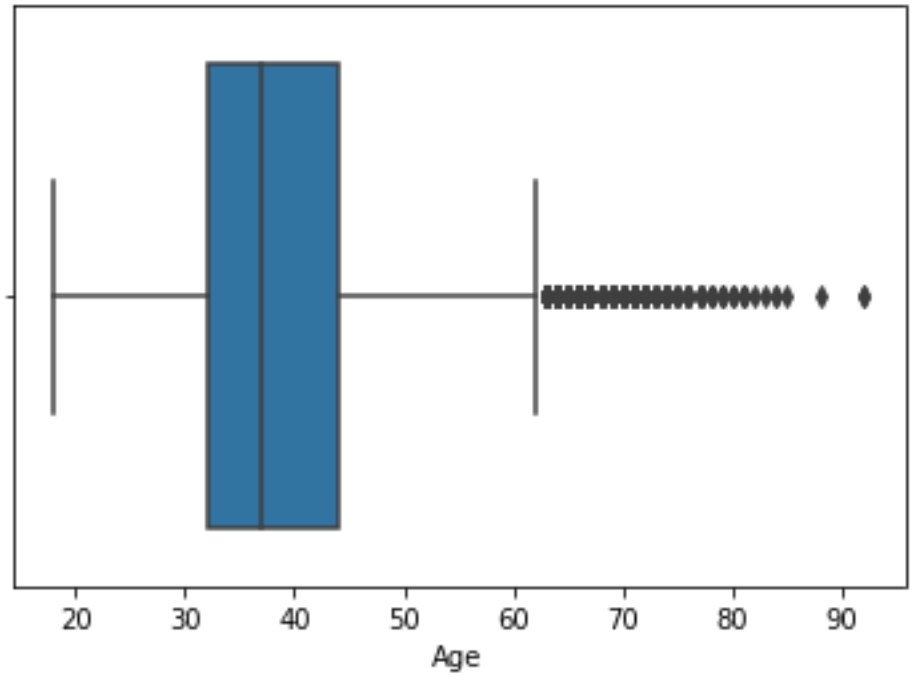
In [16]:

```
sns.boxplot(data['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11a6d5e110>
```



In [17]:

```
# upper extreme =q3+1.5*IQR
#lower extreme=q1-1.5*IQR
# IQR=q3-q1
qnt=data.quantile(q=[0.25,0.75])
```

In [18]:

```
qnt
```

Out[18]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0

In [19]:

```
IQR =qnt.loc[0.75]-qnt.loc[0.25]
```

In [20]:

```
IQR
```

Out[20]:

```
RowNumber            4999.5000
CustomerId         124705.5000
CreditScore           134.0000
Age                    12.0000
Tenure                  4.0000
Balance            127644.2400
NumOfProducts           1.0000
HasCrCard               1.0000
IsActiveMember          1.0000
EstimatedSalary     98386.1375
Exited                  0.0000
dtype: float64
```

In [21]:

```
upper_extreme=qnt.loc[0.75]+1.5*IQR
```

In [22]:

```
upper_extreme
```

Out[22]:

```
RowNumber          1.499950e+04
CustomerId         1.594029e+07
CreditScore        9.190000e+02
Age                6.200000e+01
Tenure             1.300000e+01
Balance            3.191106e+05
NumOfProducts      3.500000e+00
HasCrCard          2.500000e+00
IsActiveMember     2.500000e+00
EstimatedSalary    2.969675e+05
Exited             0.000000e+00
dtype: float64
```

In [23]:

```
lower_extreme=qnt.loc[0.25]-1.5*IQR
```

In [24]:

```
lower_extreme
```

Out[24]:

```
RowNumber         -4.998500e+03
CustomerId         1.544147e+07
CreditScore        3.830000e+02
Age                1.400000e+01
Tenure            -3.000000e+00
Balance           -1.914664e+05
NumOfProducts     -5.000000e-01
HasCrCard         -1.500000e+00
IsActiveMember    -1.500000e+00
EstimatedSalary   -9.657710e+04
Exited             0.000000e+00
dtype: float64
```

In [25]:

```
from sklearn.impute import SimpleImputer
```

In [26]:

```
imp =SimpleImputer(missing_values=np.nan,strategy='main')
```

In [27]:

```
data[data['Age']>88]
```

Out[27]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
6443	6444	15764927	Rogova	753	France	Male	92	3	121513.31	1	0	1	195563.99	0
6759	6760	15660878	T'ien	705	France	Male	92	1	126076.24	2	1	1	34436.83	0

In [28]:

```
data[data['Age']>92]
```

Out[28]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

In [29]:

```
# replacing outlier with mean
data['Age']=np.where(data['Age']>88,data['Age'].mean(),data['Age'])
```

In [30]:

```
data[data['Age']>88]
```

Out[30]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

In [31]:

```
# Encoding
from sklearn.preprocessing import LabelEncoder
```

In [32]:

```
le=LabelEncoder()
```

In [33]:

```
data['Surname']=le.fit_transform(data['Surname'])
```

In [34]:

```
data.head()
```

Out[34]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	1115	619	France	Female	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	1177	608	Spain	Female	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	2040	502	France	Female	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	289	699	France	Female	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	1822	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.10	0

In [35]:

```
# separate the dependent and independent variables
y=data['Exited']
x=data.drop(columns=['Exited'],axis=1)
```

In [36]:

```
names=x.columns
```

In [37]:

```
names
```

Out[37]:

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

In [38]:

```
# scale the independent variable
from sklearn.preprocessing import scale
```

In [39]:

```
x
```

Out[39]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	1115	619	France	Female	42.0	2	0.00	1	1	1	101348.88
1	2	15647311	1177	608	Spain	Female	41.0	1	83807.86	1	0	1	112542.58
2	3	15619304	2040	502	France	Female	42.0	8	159660.80	3	1	0	113931.57
3	4	15701354	289	699	France	Female	39.0	1	0.00	2	0	0	93826.63
4	5	15737888	1822	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	1999	771	France	Male	39.0	5	0.00	2	1	0	96270.64
9996	9997	15569892	1336	516	France	Male	35.0	10	57369.61	1	1	1	101699.77
9997	9998	15584532	1570	709	France	Female	36.0	7	0.00	1	0	1	42085.58
9998	9999	15682355	2345	772	Germany	Male	42.0	3	75075.31	2	1	0	92888.52
9999	10000	15628319	2751	792	France	Female	28.0	4	130142.79	1	1	0	38190.78

10000 rows × 13 columns

In [40]:

```
x=pd.DataFrame(x,columns=names)
```

In [41]:

```
x.head()
```

Out[41]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	1115	619	France	Female	42.0	2	0.00	1	1	1	101348.88
1	2	15647311	1177	608	Spain	Female	41.0	1	83807.86	1	0	1	112542.58
2	3	15619304	2040	502	France	Female	42.0	8	159660.80	3	1	0	113931.57
3	4	15701354	289	699	France	Female	39.0	1	0.00	2	0	0	93826.63
4	5	15737888	1822	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.10

In [42]:

```
# split the data into training and testing
from sklearn.model_selection import train_test_split
```

In [43]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [44]:

```
x_train.shape
```

Out[44]:

```
(8000, 13)
```

In [45]:

```
x_test.shape
```

Out[45]:

```
(2000, 13)
```