NAME:	Ariven
DEGREE:	Bachelor of Engineering
DEPARTMENT:	Electronics and communication Engineering
ASSIGNMENT:	2
PROJECT NAME:	Emerging Methods for Early Detection of Forest Fires

```
import panda as pd
import numpy as np

df = pd.read_csv('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]
```

```
##visulaizations
##univariate
```

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

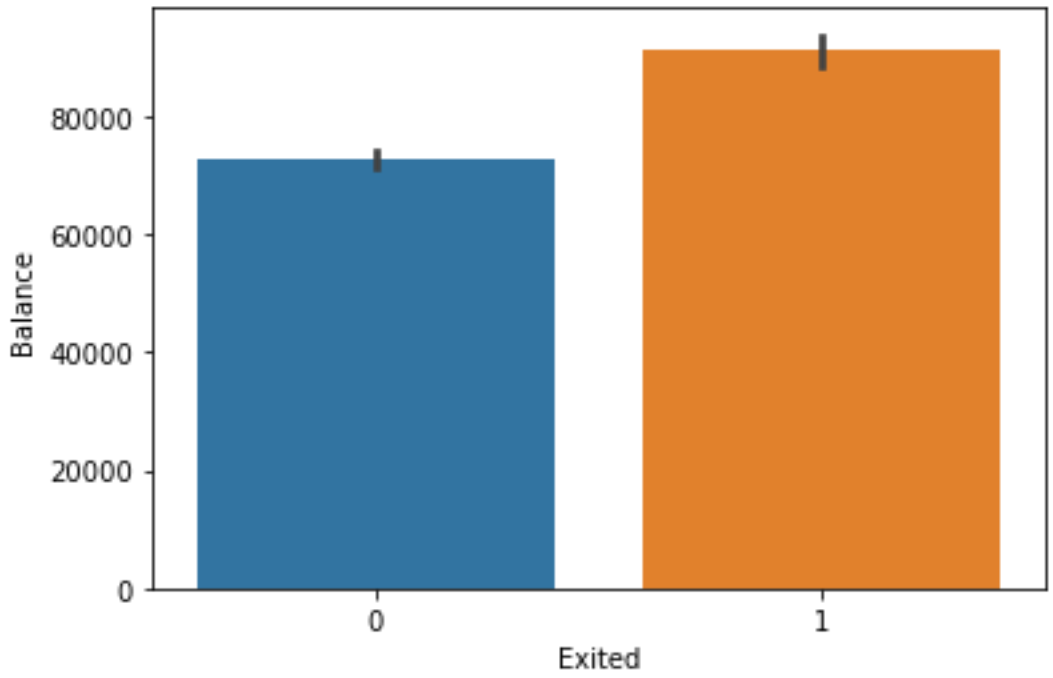
```
38.9218
```

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

```
97198.54000000001
```

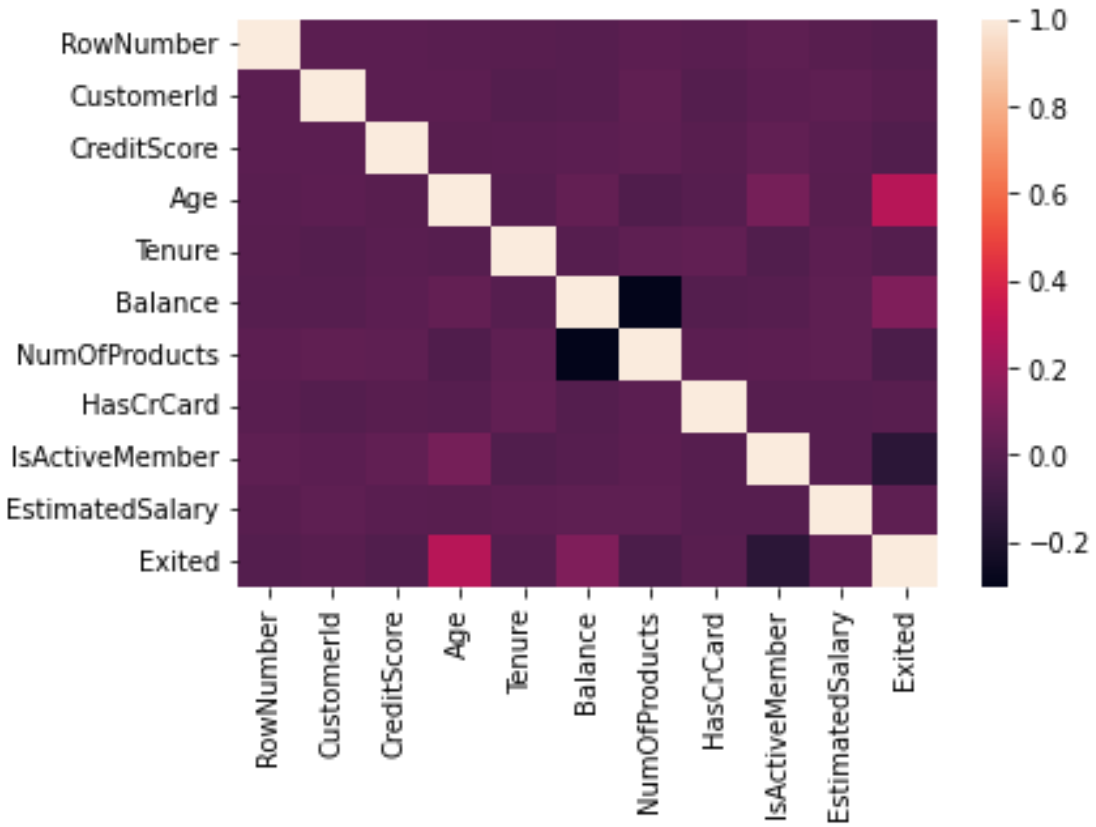
```
##bivariate
```

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.barplot(x = df['Exited'] , y = df['Balance']);
```



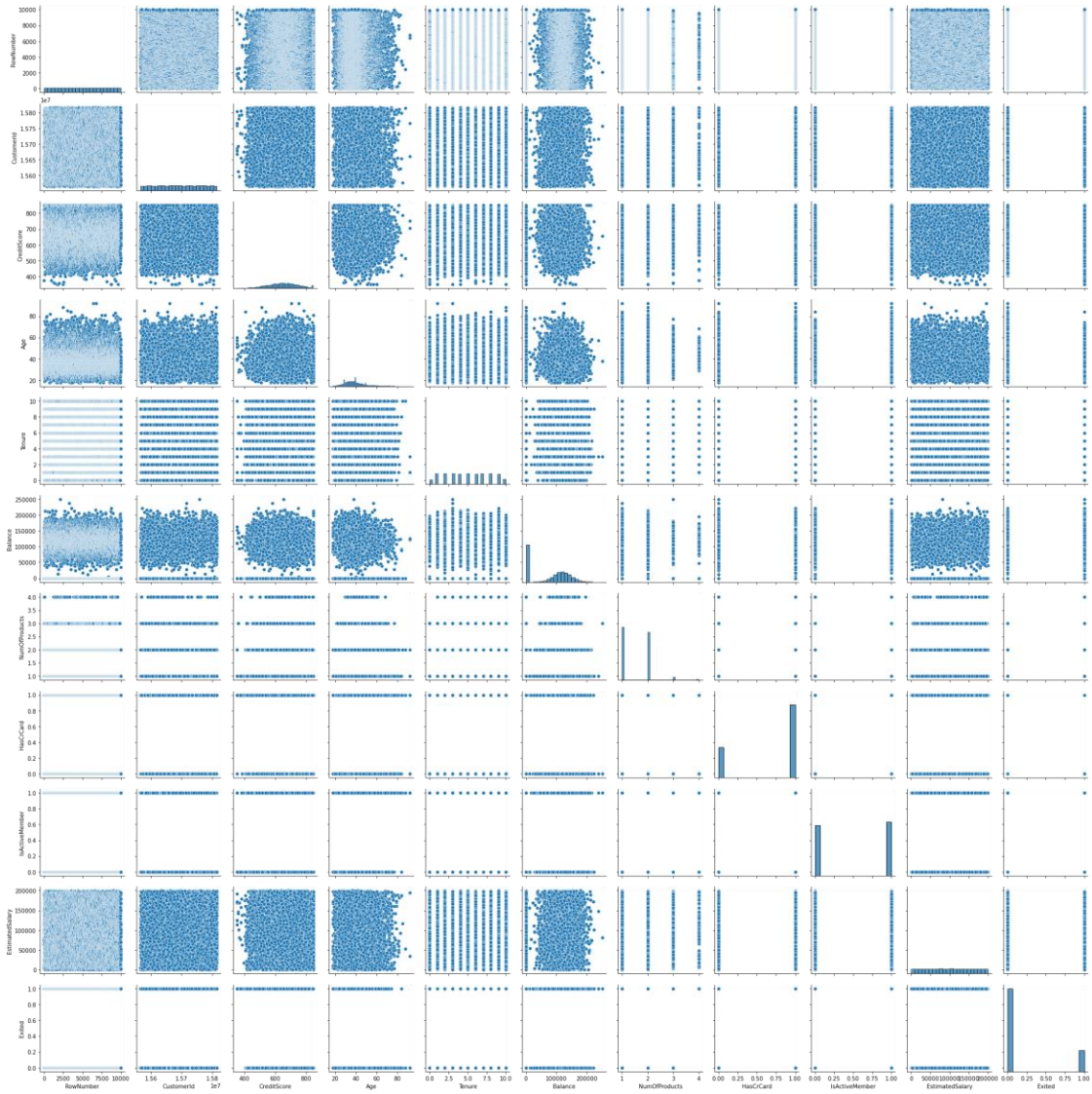

```
#multivariate
```

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




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

```
<seaborn.axisgrid.PairGrid at 0x29387e55970>
```



perform Descriptive statistics

```
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
```

```
#handle missing values
```

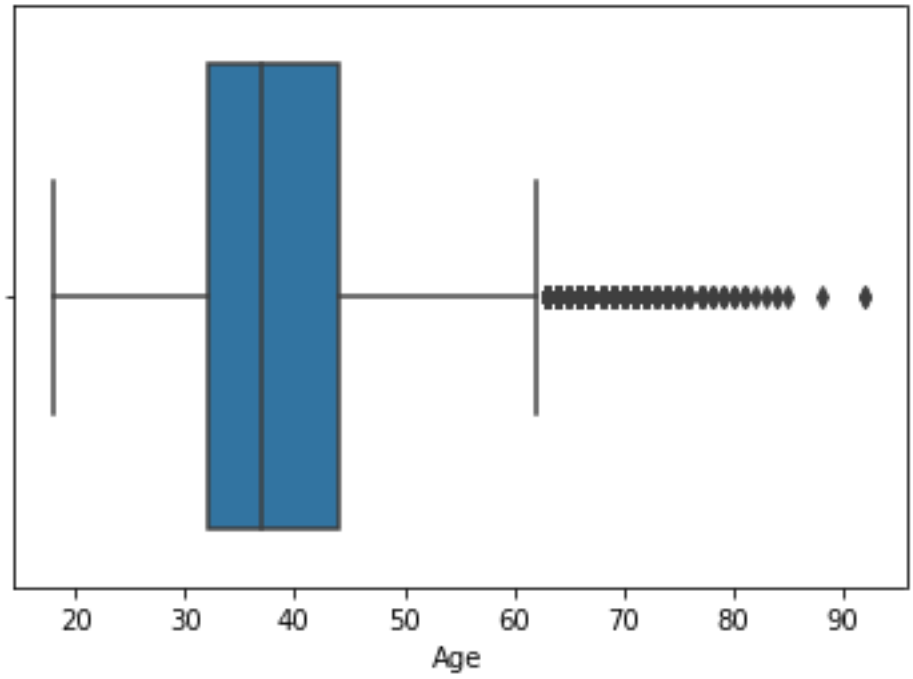
```
df.isnull().sum()
```

```
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
```

```
#find the outliers and replace it
```

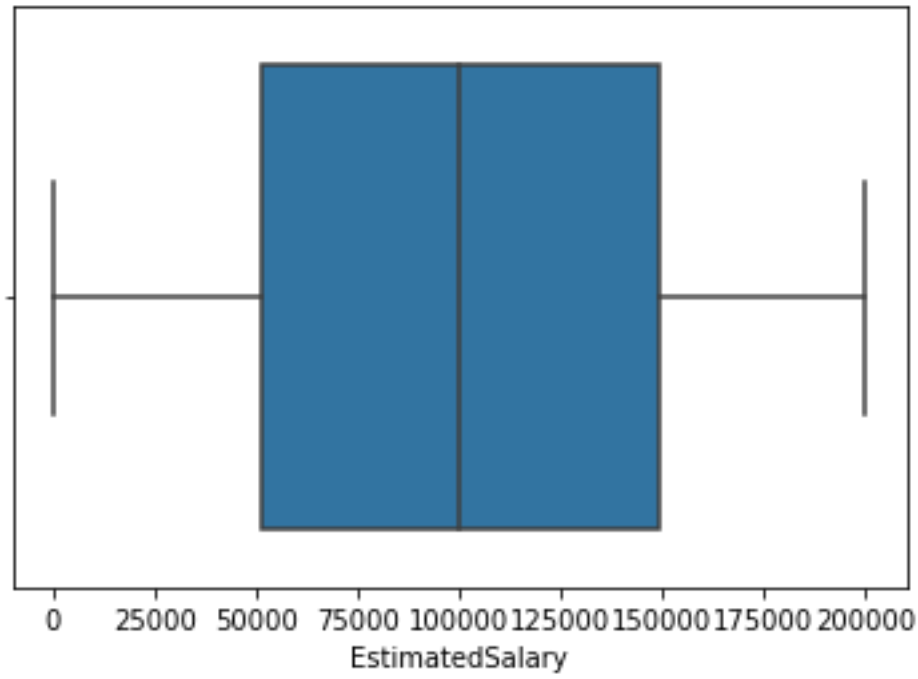
```
sns.boxplot(df['Age']);
```

```
C:\Users\BALAJI POWER MART\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(
```




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

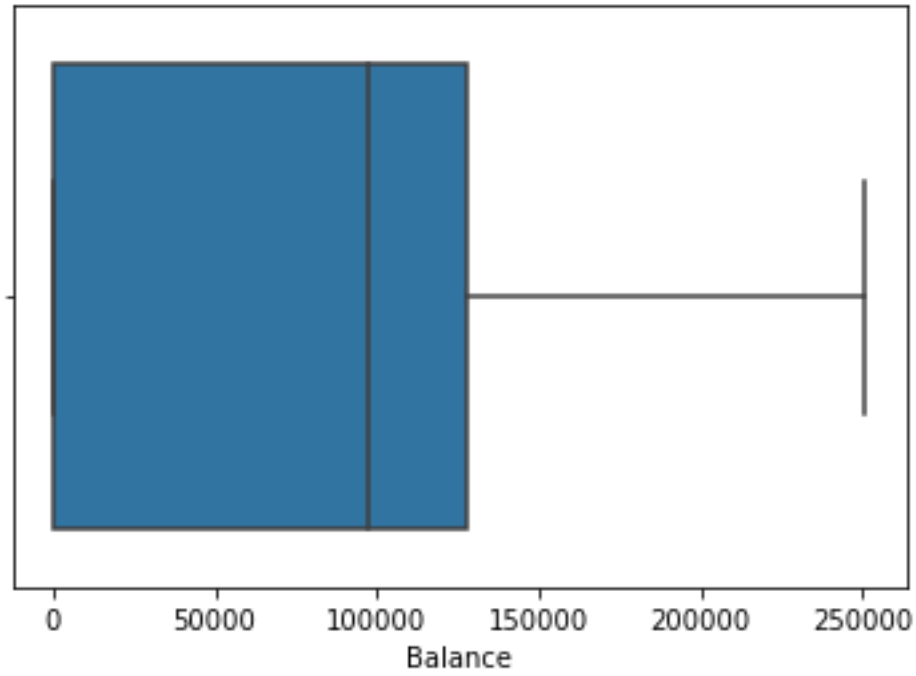
```
C:\Users\BALAJI POWER MART\anaconda3\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following
variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



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

```
C:\Users\BALAJI POWER MART\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(
```

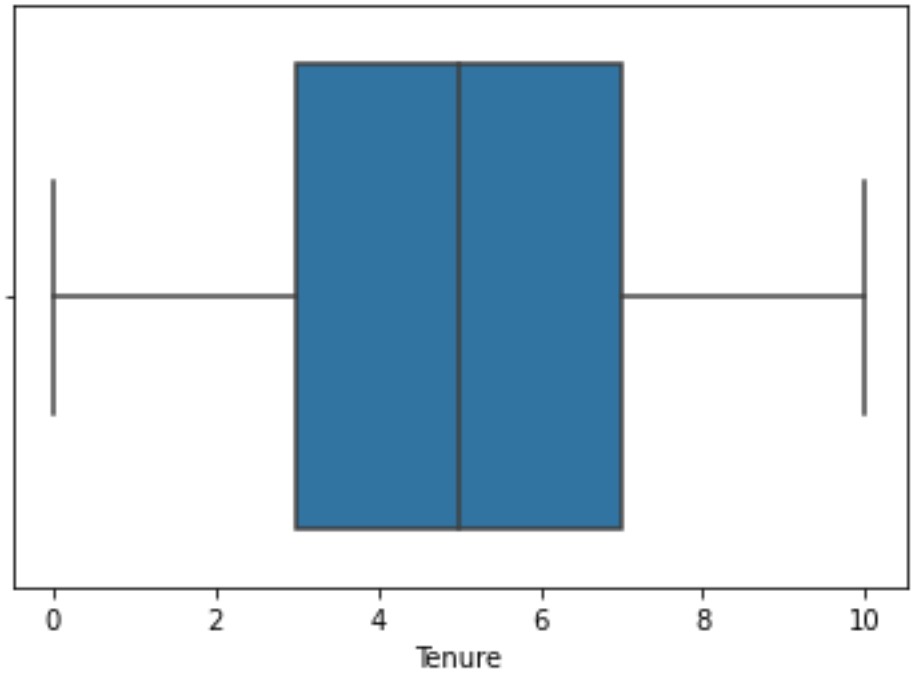
```
<AxesSubplot:xlabel='Balance'>
```




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

```
C:\Users\BALAJI POWER MART\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(
```

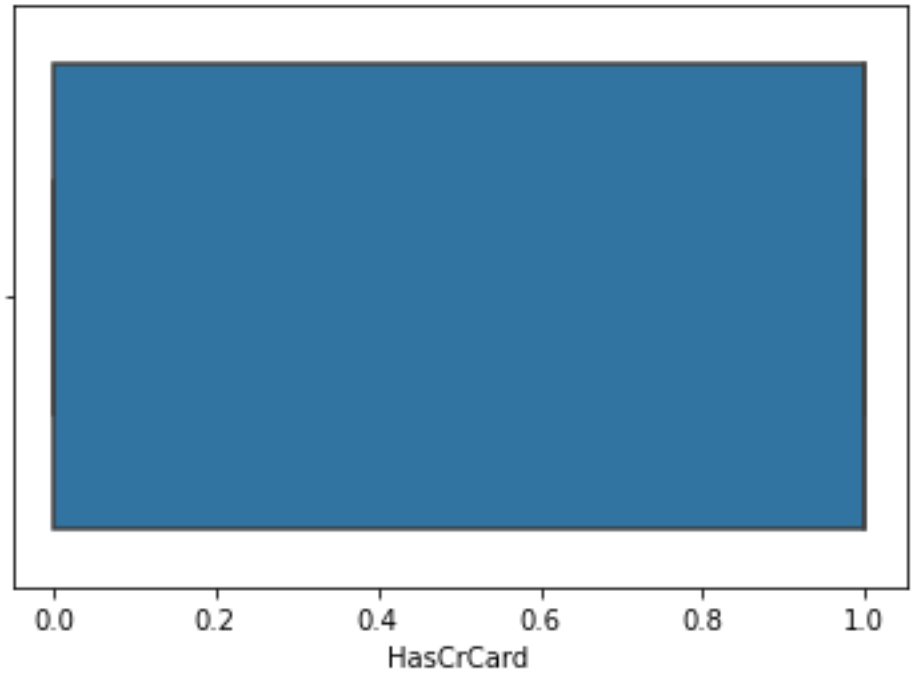
```
<AxesSubplot:xlabel='Tenure'>
```




```
sns.boxplot(df['HasCrCard'])
```

```
C:\Users\BALAJI POWER MART\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(
```

```
<AxesSubplot:xlabel='HasCrCard'>
```




```
#CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING
```

```
df.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
```

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

```
le = LabelEncoder()

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

df['CreditScore'] = le.fit_transform(df['CreditScore'])

df
```

```
      RowNumber  CustomerId    Surname  CreditScore  Geography  Gender  Age
\
0             1    15634602   Hargrave          228          0  Female   42
1             2    15647311       Hill          217          2  Female   41
2             3    15619304       Onio          111          0  Female   42
3             4    15701354       Boni          308          0  Female   39
4             5    15737888   Mitchell          459          2  Female   43
...         ...         ...        ...          ...        ...     ...  ...
9995       9996    15606229   Obijiaku          380          0    Male   39
9996       9997    15569892  Johnstone          125          0    Male   35
9997       9998    15584532        Liu          318          0  Female   36
9998       9999    15682355  Sabbatini          381          1    Male   42
9999      10000    15628319     Walker          401          0  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]
```

```
#SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

X = df.iloc[:, :-1].values
X
```

```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

```
#SCALE THE INDEPENDENT VARIABLES

 from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
Y = sc.fit_transform(X)
y
```

```
array([2, 2, 0, 2, 2, 0, 1, 0, 1, 1, 2, 1, 1, 2, 1, 1, 2, 0, 2, 0, 2, 1,
       1, 0, 1, 2, 1, 2, 0, 2, 2, 2, 0, 0, 0, 1, 2, 2, 0, 0, 1, 1, 0, 2,
       1, 2, 2, 0, 1, 0, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 2, 0, 1, 2, 0, 1,
       2, 0, 2, 1, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2, 2, 2, 1, 0, 2, 0, 2, 1,
       0, 1, 0, 0, 0, 2, 1, 2, 0, 1, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1, 0,
       1, 1, 2, 2, 0, 2, 1, 1, 1, 2, 0, 0, 1, 1, 2, 2, 1, 0, 2, 1, 0, 1,
       1, 2, 0, 0, 0, 2, 1, 0, 1, 1, 0, 2, 0, 0, 2, 0, 1, 0, 1, 2, 2, 1,
       0, 0, 1, 0, 2, 1, 0, 2, 0, 1, 2, 1, 2, 1, 1, 1, 0, 0, 2, 1, 2, 2,
       2, 0, 1, 2, 2, 1, 1, 2, 2, 0, 0, 2, 1, 1, 2, 2, 1, 1, 1, 0, 2, 0,
       0, 0, 0, 0, 1, 1, 2, 2, 2, 1, 0, 1, 1, 2, 0, 1, 0, 0, 0, 2, 1, 2,
       1, 1, 2, 1, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 1, 2, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 2, 1, 1, 2, 0, 1, 1, 0, 1, 2, 0, 1, 2, 2, 2, 0, 0,
       1, 1, 0, 1, 2, 1, 1, 2, 2, 2, 2, 0, 0, 1, 2, 1, 1, 1, 2, 1, 2, 1,
       2, 1, 1, 0, 0, 1, 2, 2, 2, 1, 2, 0, 0, 0, 1, 2, 2, 2, 0, 0, 2, 0,
       1, 2, 2, 2, 2, 2, 0, 1, 1, 2, 2, 0, 0, 1, 1, 1, 2, 1, 1, 0, 2, 0,
       0, 2, 1, 2, 1, 1, 1, 0, 1, 2, 2, 1, 2, 0, 2, 1, 0, 0, 1, 0, 2, 1,
       2, 2, 2, 2, 2, 1, 1, 0, 1, 2, 0, 2, 1, 2, 2, 2, 0, 0, 1, 2, 2, 0,
       0, 2, 0, 0, 1, 2, 2, 0, 0, 1, 1, 2, 2, 2, 0, 0, 2, 1, 1, 2, 0, 1,
       2, 0, 2, 0, 0, 1, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 1, 0, 0, 1, 1, 1,
       0, 2, 0, 0, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 0, 1, 0, 1, 0, 0, 1,
       2, 2, 0, 2, 2, 0, 0, 2, 1, 1, 2, 2, 1, 1, 0, 2, 2, 0, 0, 0, 0, 1,
       0, 1, 2, 1, 2, 0, 0, 0, 1, 2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 2, 1, 1,
       0, 1, 1, 1, 0, 0, 2, 2, 0, 1, 2, 1, 2, 2, 1, 0, 0, 2, 2, 1, 1, 2,
       0, 2, 1, 0, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 2, 2,
       1, 0, 2, 0, 0, 1, 0, 2, 2, 1, 0, 0, 0, 1, 0, 2, 1, 1, 1, 1, 2, 0,
       0, 0, 0, 2, 0, 2, 1, 1, 2, 1, 0, 0, 0, 0, 2, 1, 1, 1, 2, 0, 0, 2,
       1, 0, 2, 0, 0, 0, 2, 2, 0, 1, 0, 1, 1, 0, 0, 1, 2, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 2, 0, 1, 0, 1, 2, 0, 0, 0, 2, 0, 0, 1, 0, 1,
       0, 1, 2, 2, 2, 0, 1, 0, 0, 0, 0, 2, 1, 2, 2, 0, 1, 2, 2, 0, 1, 0,
       1, 2, 1, 2, 2, 1, 2, 0, 1, 2, 1, 1, 0, 0, 2, 1, 0, 1, 2, 1, 0, 1,
       2, 2, 0, 1, 1, 0, 0, 1, 1, 2, 1, 0, 0, 0, 1, 2, 2, 1, 0, 0, 2, 0,
```

```
       2, 0, 1, 2, 2, 0, 1, 1, 1, 1, 0, 0, 1, 2, 0, 0, 1, 2, 0, 0, 2, 2,
       1, 1, 2, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 2, 1, 2, 1, 2, 2, 0, 2,
       0, 2, 0, 1, 2, 1, 2, 1, 2, 2, 2, 1, 1, 0, 1, 0, 0, 2, 2, 0, 2, 1,
       0, 0, 0, 1, 0, 0, 1, 2, 0, 2, 2, 2, 1, 0, 2, 0, 2, 2, 2, 2, 1, 1,
       0, 1, 2, 1, 1, 2, 0, 2, 2, 2, 0, 2, 1, 1, 2, 1, 2, 0, 2, 1, 1, 0,
       2, 0, 1, 0, 0, 1, 0, 0, 0, 0, 2, 2, 2, 1, 1, 1, 1, 2, 2, 0, 2, 2,
       1, 2, 2, 0, 2, 0, 0, 1, 2, 1, 0, 2, 0, 2, 1, 0, 1, 2, 0, 1, 2, 1,
       1, 2, 2, 0, 2, 0, 0, 1, 1, 2, 1, 1, 0, 1, 1, 2, 2, 1, 2, 0, 1, 0,
       0, 2, 1, 0, 2, 2, 2, 2, 1, 0, 2, 1, 1, 2, 2, 1, 2, 1, 2, 0, 1, 0,
       1, 0, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 2, 2, 1, 1, 1, 1, 0, 1, 2, 0,
       0, 2, 2, 2, 2, 0, 2, 1, 0, 1, 1, 2, 1, 1, 0, 2, 2, 1, 2, 1, 1, 0,
       0, 0, 2, 1, 0, 1, 0, 2, 1, 1, 2, 0, 2, 1, 0, 0, 1, 0, 0, 2, 2, 2,
       0, 1, 1, 0, 1, 2, 0, 0, 2, 0, 0, 2, 2, 0, 1, 1, 0, 0, 2, 0, 0, 2,
       2, 0, 2, 1, 1, 2, 0, 1, 2, 1, 1, 1, 0, 2, 0, 1, 0, 0, 0, 2, 0, 2,
       0, 0, 2, 0, 2, 1, 1, 2, 0, 0])
```

```
# split data into training and testing


from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
X, y = make_blobs(n_samples=1000)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(670, 2) (330, 2) (670,) (330,)
```