Importing Libraries :

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sb
```

Uploading Dataset

```
In [2]: import csv
        df = pd.read_csv("C:\\Users\\Narmatha Palnisamy\\Downloads\\Churn_Modelling.csv")
```

Reading Dataset

```
In [3]: df.shape
Out[3]: (10000, 14)
```

```
In [4]: df.head()
```

Out[4]:

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

```
In [5]: df.tail()
```

Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	9627
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	3819

```
In [6]: df.columns
Out[6]: Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
               'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
               'IsActiveMember', 'EstimatedSalary', 'Exited'],
              dtype='object')
```

Categorical Columns and Perform Encoding

```
In [8]: df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname', 'Geography'])
        df.head()
```

Out[8]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Female	43	2	125510.82	1	1	1	79084.10	0

Dropping Unwanted Columns

Handling Missing Values

```
In [9]: df.duplicated().sum()
Out[9]: 0

In [10]: df.isna().sum()
Out[10]: CreditScore        0
         Gender             0
         Age                0
         Tenure             0
         Balance            0
         NumOfProducts      0
         HasCrCard          0
         IsActiveMember     0
         EstimatedSalary    0
         Exited             0
         dtype: int64

In [11]: df.nunique()
Out[11]: CreditScore         460
         Gender                2
         Age                  70
         Tenure               11
         Balance            6382
         NumOfProducts         4
         HasCrCard             2
         IsActiveMember        2
         EstimatedSalary    9999
         Exited                2
         dtype: int64
```

```
In [12]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   CreditScore      10000 non-null  int64
 1   Gender           10000 non-null  object
 2   Age              10000 non-null  int64
 3   Tenure           10000 non-null  int64
 4   Balance          10000 non-null  float64
 5   NumOfProducts    10000 non-null  int64
 6   HasCrCard        10000 non-null  int64
 7   IsActiveMember   10000 non-null  int64
 8   EstimatedSalary  10000 non-null  float64
 9   Exited           10000 non-null  int64
dtypes: float64(2), int64(7), object(1)
memory usage: 781.4+ KB
```

```
In [13]: df.drop(columns=['Gender', 'HasCrCard', 'IsActiveMember', 'Exited']).describe()
```

Out[13]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	100090.239881
std	96.653299	10.487806	2.892174	62397.405202	0.581654	57510.492818
min	350.000000	18.000000	0.000000	0.000000	1.000000	11.580000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	51002.110000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	100193.915000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	149388.247500
max	850.000000	92.000000	10.000000	250898.090000	4.000000	199992.480000

Find Outliers

```
In [14]: qnt = df.drop(columns=['Gender', 'Tenure', 'HasCrCard', 'IsActiveMember', 'NumOfProducts', 'Exited']).quantile(q=[0.25, 0.75])
         qnt
```

Out[14]:

	CreditScore	Age	Balance	EstimatedSalary
0.25	584.0	32.0	0.00	51002.1100
0.75	718.0	44.0	127644.24	149388.2475

```
Q1 = qnt.iloc[0]
Q3 = qnt.iloc[1]
iqr = Q3 - Q1
iqr
```

```
CreditScore           134.0000
Age                    12.0000
Balance            127644.2400
EstimatedSalary     98386.1375
dtype: float64
```

```
upper = qnt.iloc[1] + 1.5*iqr
upper
```

```
CreditScore           919.00000
Age                    62.00000
Balance            319110.60000
EstimatedSalary    296967.45375
dtype: float64
```

```
lower = qnt.iloc[0] - 1.5*iqr
lower
```

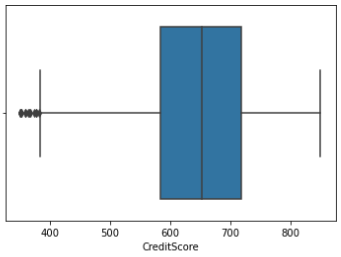
```
CreditScore           383.00000
Age                    14.00000
Balance           -191466.36000
EstimatedSalary    -96577.09625
dtype: float64
```

Performing Visualisations

In [16]:
```
sb.boxplot(df['CreditScore'])
```

```
C:\Users\Narmatha Palnisamy\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(
```

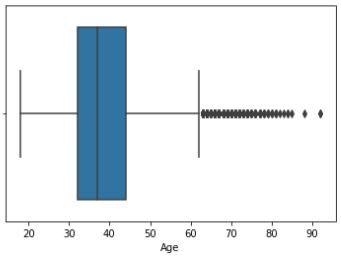
Out[16]: <AxesSubplot:xlabel='CreditScore'>



In [17]:
```
sb.boxplot(df['Age'])
```

```
C:\Users\Narmatha Palnisamy\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(
```

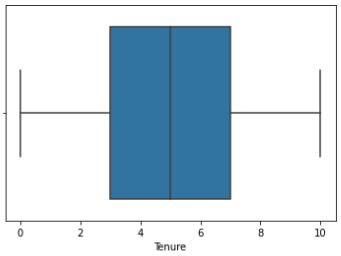
Out[17]: <AxesSubplot:xlabel='Age'>



In [19]:
```
sb.boxplot(df['Tenure'])
```

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

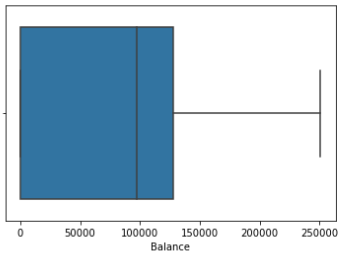
Out[19]: <AxesSubplot:xlabel='Tenure'>



In [20]:
```
sb.boxplot(df['Balance'])
```

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

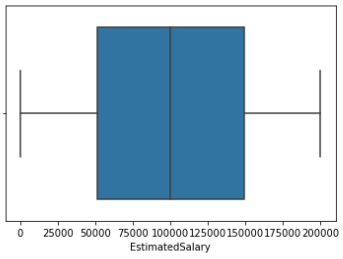
Out[20]: <AxesSubplot:xlabel='Balance'>



In [21]:
```
sb.boxplot(df['EstimatedSalary'])
```

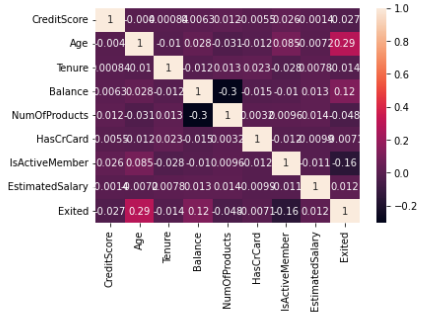
```
C:\Users\Narmatha Palnisamy\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error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

Out[21]: <AxesSubplot:xlabel='EstimatedSalary'>



In [22]:
```
sb.heatmap(df.corr(), annot=True)
```

Out[22]: <AxesSubplot:>



Split Data into Dependent and Independent Variables

```
In [23]: x = df.iloc[:, :-1]
         x.head()
```

Out[23]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	Female	42	2	0.00	1	1	1	101348.88
1	608	Female	41	1	83807.86	1	0	1	112542.58
2	502	Female	42	8	159660.80	3	1	0	113931.57
3	699	Female	39	1	0.00	2	0	0	93826.63
4	850	Female	43	2	125510.82	1	1	1	79084.10

```
In [24]: y = df.iloc[:, -1]
         y.head()
```

Out[24]:

```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
x = ss.fit_transform(x)
```

```
x
```

```
array([[-0.13284832, -1.09598752,  0.48205148, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.28182929, -1.09598752,  0.36638802, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.71746409, -1.09598752,  0.48205148, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 1.08608688, -1.09598752, -0.21192932, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 0.29416906,  0.91241915,  0.48205148, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 0.29416906, -1.09598752, -1.13723705, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

Split the data into training and testing

```
In [26]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

```
In [27]: x_train.shape
```

Out[27]: (8000, 9)

```
In [28]: x_test.shape
```

Out[28]: (2000, 9)

```
In [29]: x_train
```

Out[29]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
7389	667	Female	34	5	0.00	2	1	0	163830.64
9275	427	Male	42	1	75681.52	1	1	1	57098.00
2995	535	Female	29	2	112367.34	1	1	0	185630.76
5316	654	Male	40	5	105683.63	1	1	0	173617.09
356	850	Female	57	8	126776.30	2	1	1	132298.49
...	...	...	...	...	...	...	...	...	...
9225	594	Female	32	4	120074.97	2	1	1	162961.79
4859	794	Female	22	4	114440.24	1	1	1	107753.07
3264	738	Male	35	5	161274.05	2	1	0	181429.87
9845	590	Female	38	9	0.00	2	1	1	148750.16
2732	623	Female	48	1	108076.33	1	1	0	118855.26

8000 rows × 9 columns

In [30]:
```
x_test
```

Out[30]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
9394	597	Female	35	8	131101.04	1	1	1	192852.67
898	523	Female	40	2	102967.41	1	1	0	128702.10
2398	706	Female	42	8	95386.82	1	1	1	75732.25
5906	788	Male	32	4	112079.58	1	0	0	89368.59
2343	706	Male	38	5	163034.82	2	1	1	135662.17
...	...	...	...	...	...	...	...	...	...
1037	625	Female	24	1	0.00	2	1	1	180969.55
2899	586	Female	35	7	0.00	2	1	0	70760.69
9549	578	Male	36	1	157267.95	2	1	0	141533.19
2740	650	Male	34	4	142393.11	1	1	1	11276.48
6690	573	Male	30	8	127406.50	1	1	0	192950.60

2000 rows × 9 columns

In [ ]: