In [ ]:
```
####################################    ASSIGNMENT 2   ############################################
```

In [ ]:
```
####################################### 1. DOWNLOADED THE DATASET ##############################
```

In [ ]:
```
 #################################### 2. LOAD THE DATASET ########################################
```

In [1]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
df = pd.read_csv('Churn_Modelling.csv')
```

In [5]:
```
df
```

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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
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

10000 rows × 14 columns

In [6]:
```
#################################### 3. PERFORM BELOW VISUALIZATIONS ##############################################
```

In [7]:
```
                         ############# UNIVARIATE ##############
```

In [8]:
```
df['Age'].mean()
```

Out[8]: 38.9218

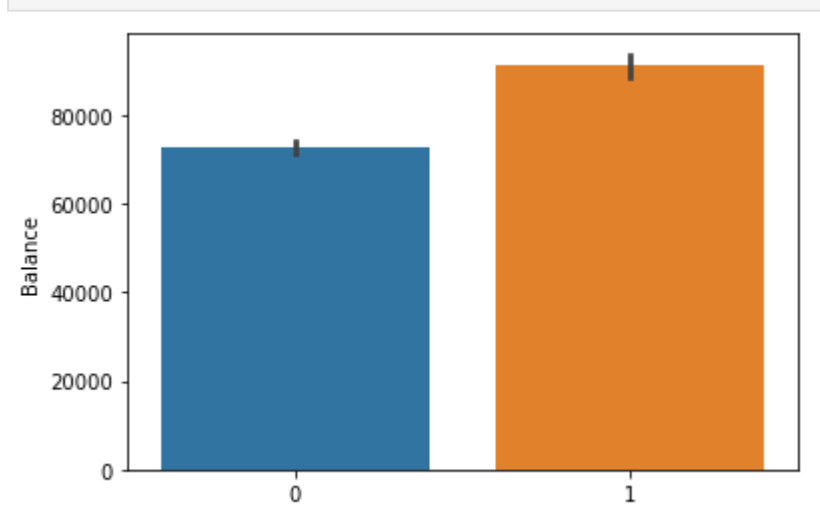
In [9]:
```
df['Balance'].median()
```

Out[9]: 97198.54000000001

In [10]:
```
                        ############# BIVARIATE  ###############
```

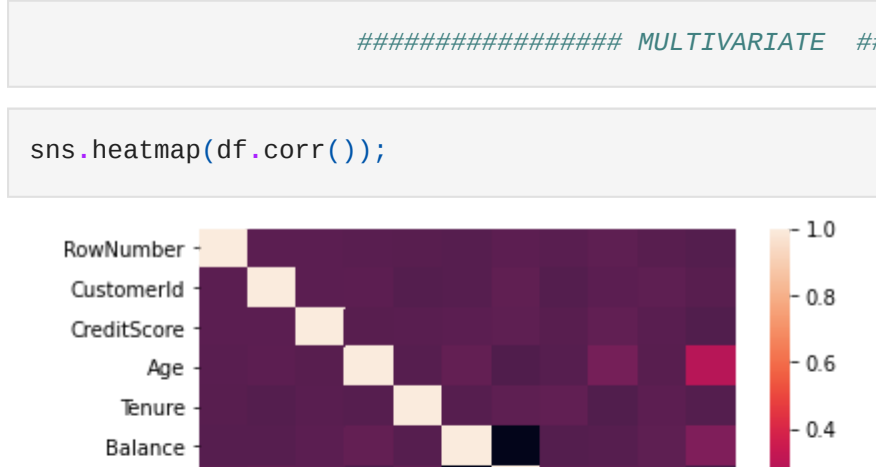
In [11]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:
```
sns.barplot(x = df['Exited'] , y = df['Balance']);
```



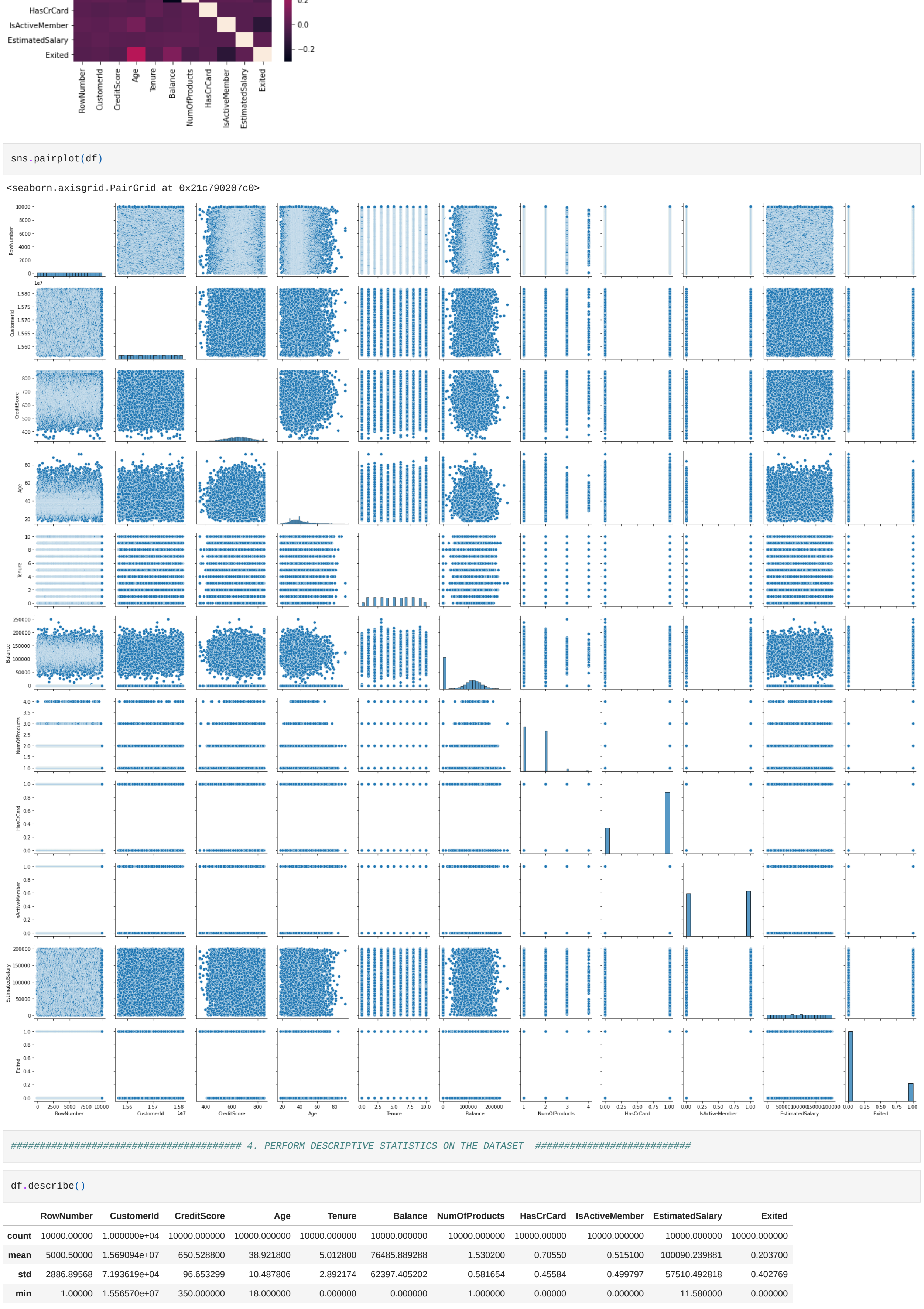
In [14]:
```
                      ################# MULTIVARIATE  #################
```

In [15]:
```
sns.heatmap(df.corr());
```



In [16]:
```
sns.pairplot(df)
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x21c790207c0>



In [17]:
```
####################################### 4. PERFORM DESCRIPTIVE STATISTICS ON THE DATASET  ##########################
```

In [18]:
```
df.describe()
```

Out[18]:

	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
min	1.00000	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 [19]:
```
####################################### 5. HANDLE THE MISSING VALUES  ######################################
```

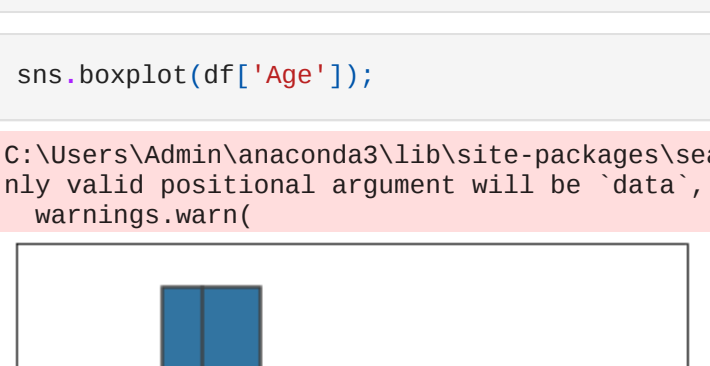
In [20]:
```
df.isnull().sum()
```

Out[20]:
```
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 [21]:
```
###################################### 6. FIND THE OUTLIERS AND REPLACE THE OUTLIERS  ###############################
```

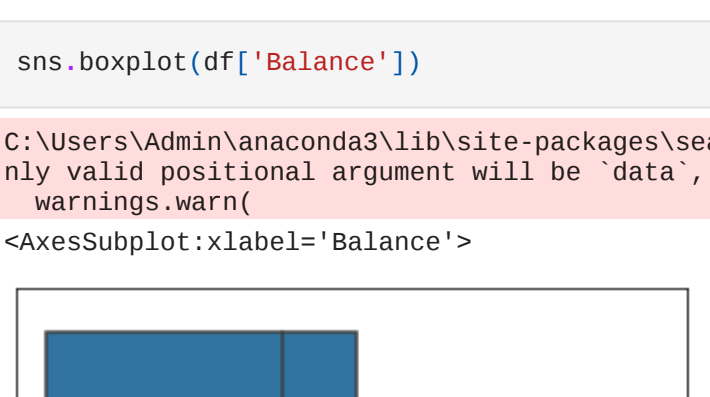
In [22]:
```
sns.boxplot(df['Age']);
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: 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 without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [23]:
```
sns.boxplot(df['EstimatedSalary']);
```

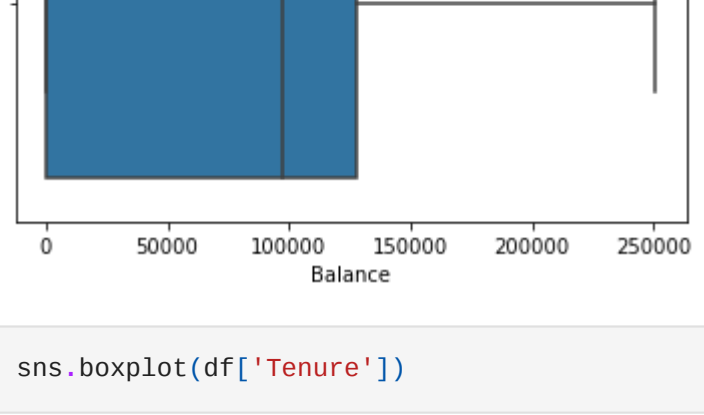
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.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



In [25]:
```
sns.boxplot(df['Balance'])
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: 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 without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

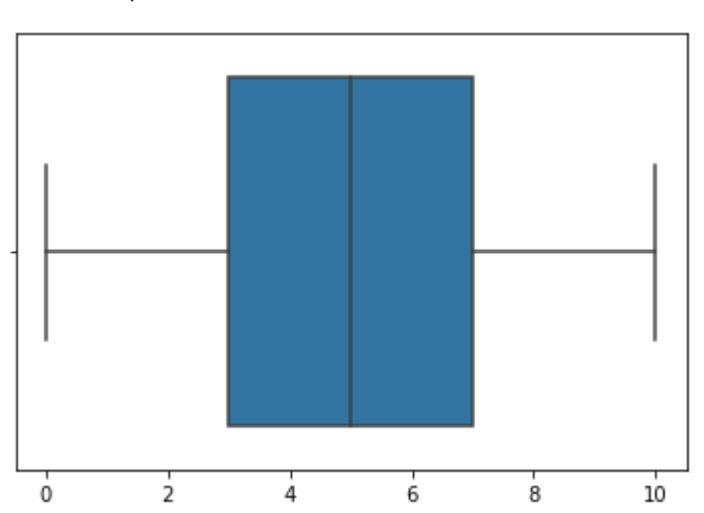
Out[25]: <AxesSubplot:xlabel='Balance'>



In [26]:
```
sns.boxplot(df['Tenure'])
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: 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 without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

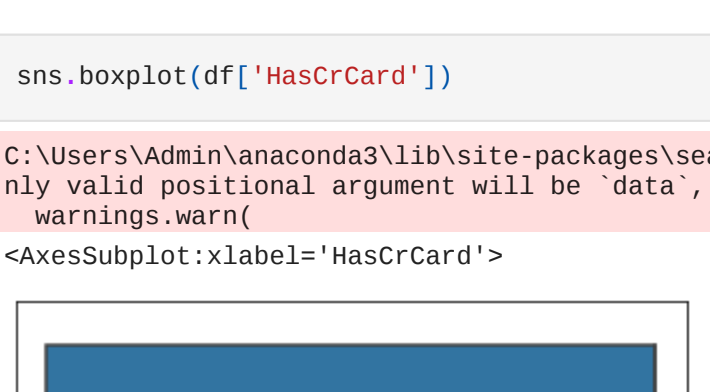
Out[26]: <AxesSubplot:xlabel='Tenure'>



In [27]:
```
sns.boxplot(df['HasCrCard'])
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: 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 without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]: <AxesSubplot:xlabel='HasCrCard'>



In [30]:
```
############################################ 7. CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING ###########################
```

In [31]:
```
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
```

In [32]:
```
from sklearn.preprocessing import LabelEncoder
```

In [33]:
```
le = LabelEncoder()
```

In [34]:
```
df['Geography'] = le.fit_transform(df['Geography'])
```

In [35]:
```
df['CreditScore'] = le.fit_transform(df['CreditScore'])
```

In [36]:
```
df
```

Out[36]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

In [37]:
```
#################################### 8. SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES  ##############################
```

In [38]:
```
####################################  10. SPLIT THE DATA INTO TRAINING AND TESTING ################################
```

In [39]:
```
df.drop(columns = ['RowNumber'])
```

Out[39]:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	Obijiaku	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	15569892	Johnstone	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	Liu	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	15682355	Sabbatini	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	Walker	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 13 columns

In [49]:
```
x = df.iloc[: , 0:13].values
y = df.iloc[: , 13:14].values
```

In [50]:
```
from sklearn.model_selection import train_test_split
```

In [51]:
```
xtrain , xtest , ytrain , ytest = train_test_split(x , y , test_size = 0.3 , random_state = 0)
```

In [52]:
```
xtrain.shape , xtest.shape
```

Out[52]: ((7000, 13), (3000, 13))

In [40]:
```
######################################## 9. SCALE THE INDEPENDENT VARIABLES #################################
```

In [41]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

In [42]:
```
n = MinMaxScaler()
s = StandardScaler()
```

In [43]:
```
x = df[['Age', 'Tenure']].values
y = df['Gender'].values
```

In [45]:
```
n_xtrain = n.fit_transform(xtrain)
```

In [ ]:
```
n_xtest = n.fit_transform(xtest)
```