ASSIGNMENT - 2

1. Download the dataset

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

2.Load the dataset.

In [3]:
```
data=pd.read_csv(r"Churn_Modelling.csv")
```

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

Out[4]:

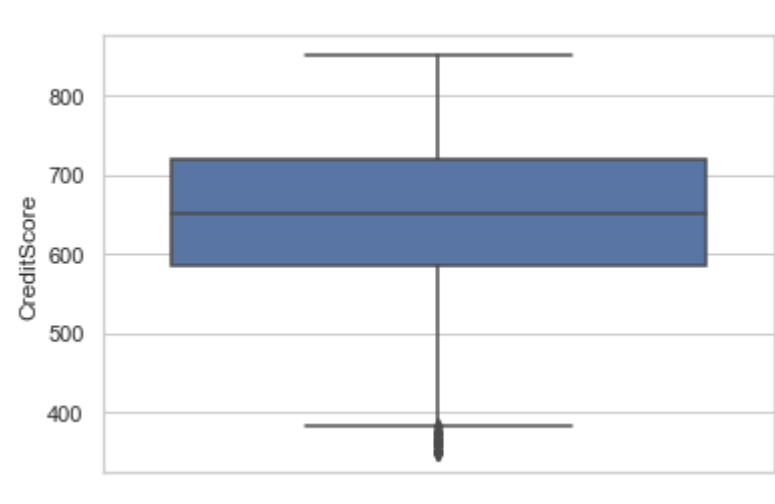
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0
3	4	15701354	Boni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

3.Perform Below Visualizations

3.1 Univariate Analysis

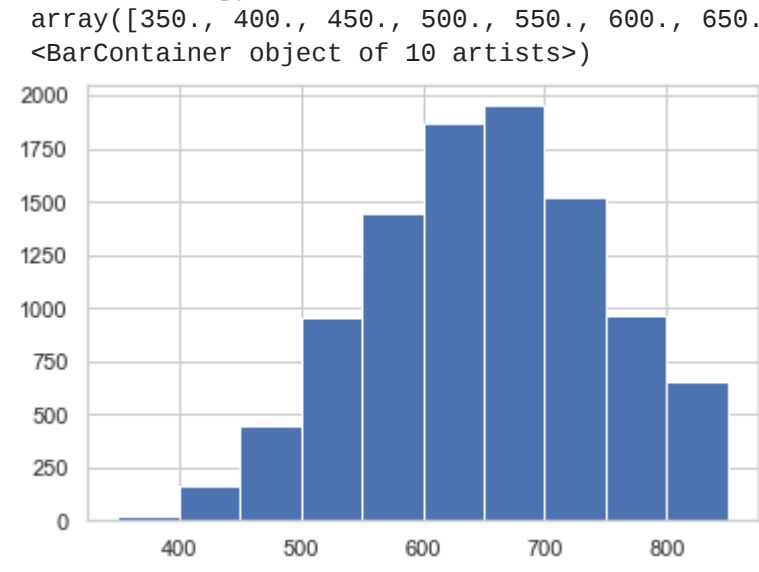
In [8]:
```
sns.set(style="whitegrid")
sns.boxplot(y=data['CreditScore'])
```

Out[8]:
```
<AxesSubplot:ylabel='CreditScore'>
```



In [9]:
```
plt.hist(data['CreditScore'])
```

Out[9]:
```
(array([  19.,  166.,  447.,  958., 1444., 1866., 1952., 1525.,  968.,
         655.]),
 array([350., 400., 450., 500., 550., 600., 650., 700., 750., 800., 850.]),
 <BarContainer object of 10 artists>)
```

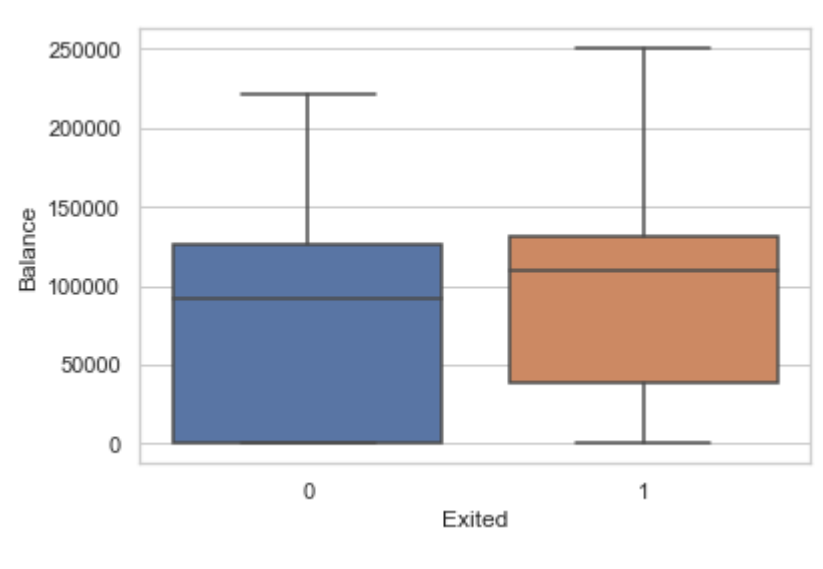


3.2 Bi - Variate Analysis

credit_score = data['CreditScore'] balance = data['Balance'] credit_score.dropna() balance.dropna() print(credit_score[2]) print(balance[2]) plt.scatter(credit_score , balance) plt.show()

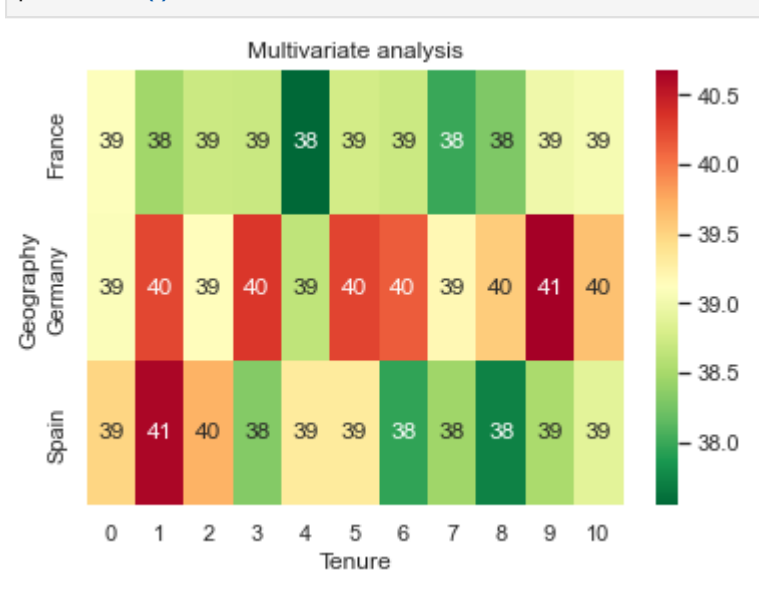
In [11]:
```
sns.boxplot(y=data.Balance,x=data.Exited)
```

Out[11]:
```
<AxesSubplot:xlabel='Exited', ylabel='Balance'>
```



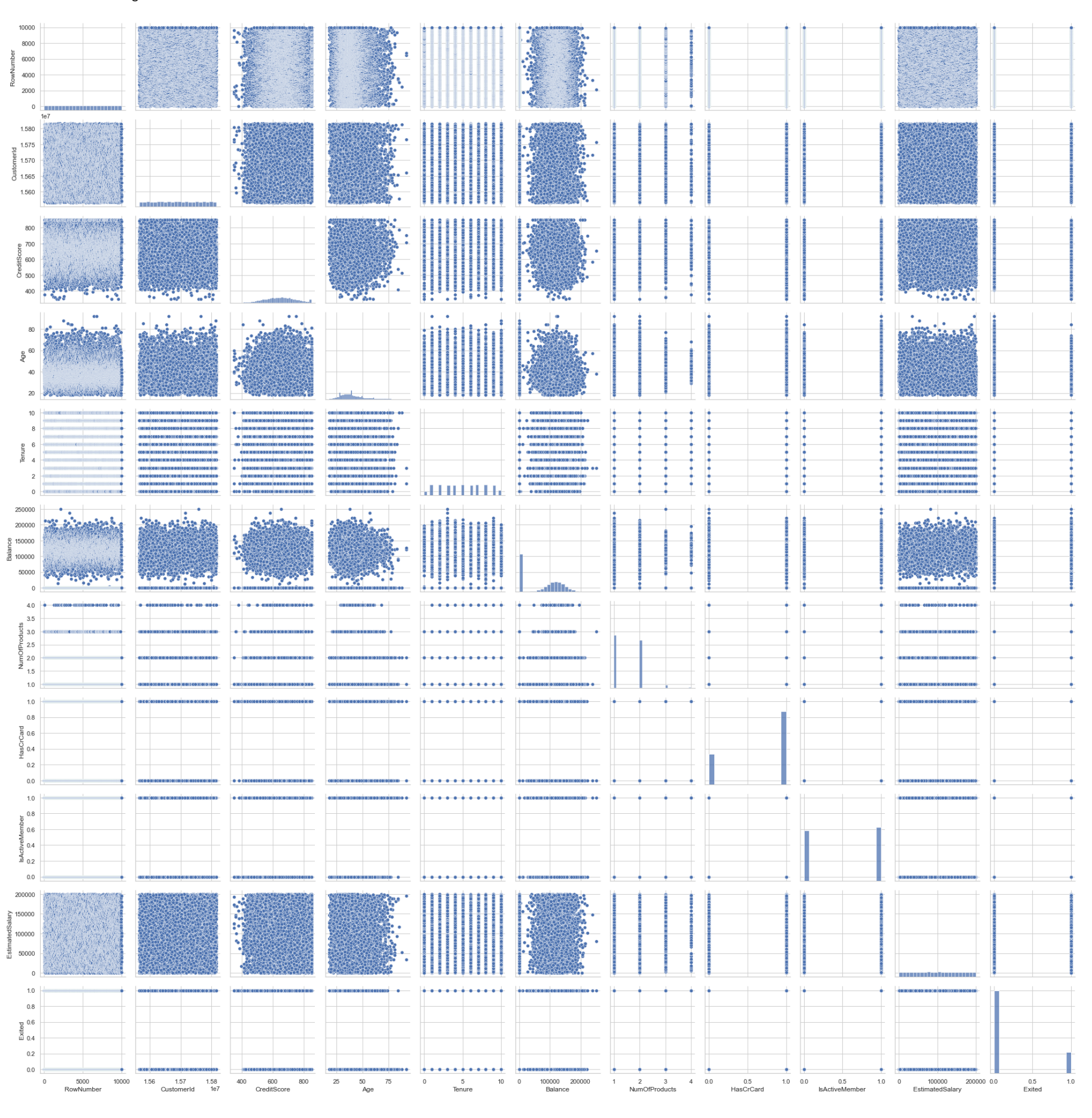
3.3 Multi - Variate Analysis

In [20]:
```
result = pd.pivot_table(data=data, index='Geography', columns='Tenure',values='Age')
sns.heatmap(result, annot=True, cmap = 'RdYlGn_r').set_title('Multivariate analysis')
plt.show()
```



In [21]:
```
sns.pairplot(data)
```

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x1db16be2340>
```



1. Perform descriptive statistics on the dataset

In [22]:
```
data.describe()
```

Out[22]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Est
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	1(
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	5
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	5
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	1(
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	1
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	1

In [23]:
```
data['Age'].mode()
```

Out[23]:
```
0    37
Name: Age, dtype: int64
```

In [24]:
```
data["Age"].mean()
```

Out[24]:
```
38.9218
```

In [25]:
```
data["Age"].median()
```

Out[25]:
```
37.0
```

In [26]:
```
s=round(data['Age'].std())
print(s)
```

```
10
```

In [27]:
```
data.columns
```

Out[27]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [28]:
```
o=data[['Balance','Age']]
```

In [29]:
```
o
```

Out[29]:

	Balance	Age
0	0.00	42
1	83807.86	41
2	159660.80	42
3	0.00	39
4	125510.82	43
...	...	...
9995	0.00	39
9996	57369.61	35
9997	0.00	36
9998	75075.31	42
9999	130142.79	28

10000 rows × 2 columns

In [30]:
```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
st_scale=scale.fit_transform(o)
st_scale
```

Out[30]:
```
array([[-1.22584767,  0.29351742],
       [ 0.11735002,  0.19816383],
       [ 1.33305335,  0.29351742],
       ...,
       [-1.22584767, -0.27860412],
       [-0.02260751,  0.29351742],
       [ 0.85996499, -1.04143285]])
```

In [33]:
```
#range
data.max()
```

Out[33]:
```
RowNumber               10000
CustomerId           15815690
Surname                Zuyeva
CreditScore               850
Geography               Spain
Gender                   Male
Age                        92
Tenure                     10
Balance             250898.09
NumOfProducts               4
HasCrCard                   1
IsActiveMember              1
EstimatedSalary     199992.48
Exited                      1
dtype: object
```

5.Handle the Missing values.

In [34]:
```
data.isna().sum()
```

Out[34]:
```
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 [35]:
```
data.isna().any()
```

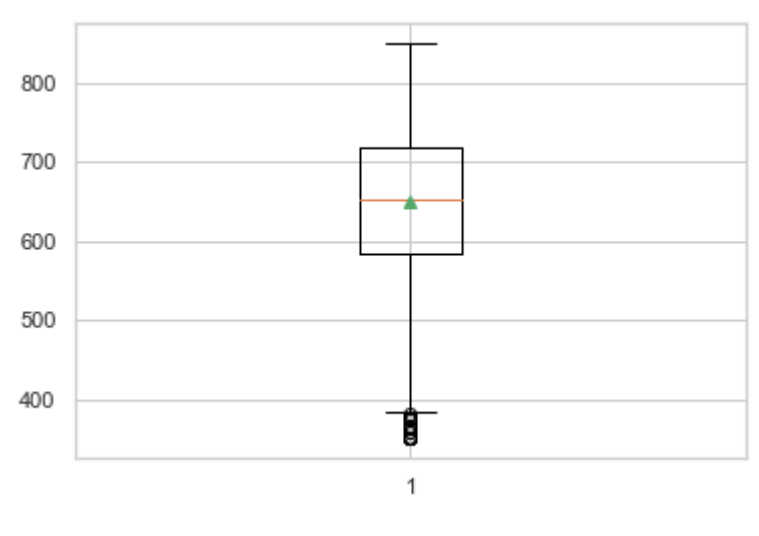
Out[35]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [36]:
```
CreditScores = data['CreditScore']
CreditScores
```

Out[36]:
```
0       619
1       608
2       502
3       699
4       850
       ...
9995    771
9996    516
9997    709
9998    772
9999    792
Name: CreditScore, Length: 10000, dtype: int64
```

1. Find the outliers and replace the outliers

In [37]:
```
plt.boxplot(data['CreditScore'],showmeans = True)
plt.show()
```



In [38]:
```
df = data[data['CreditScore'] >= 378]
for i in data['CreditScore']:
    if(i<378):
        print(i)
print(data['CreditScore'])
```

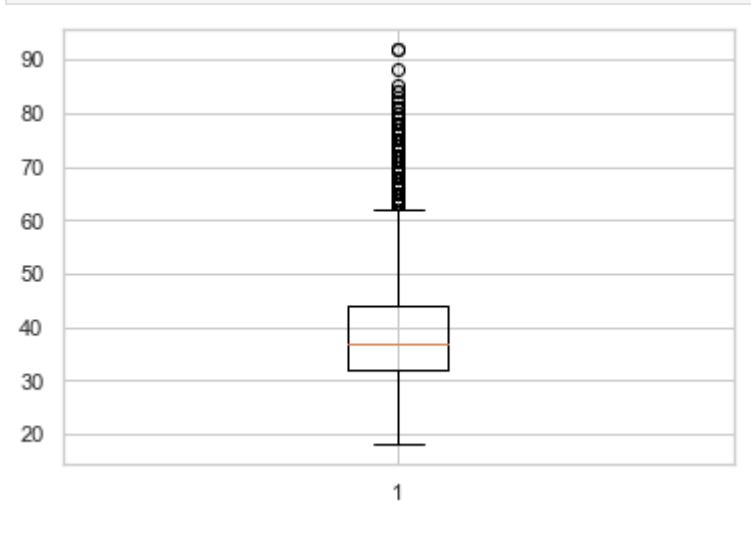
```
376
376
363
359
350
350
358
351
365
367
350
350
373
350
0       619
1       608
2       502
3       699
4       850
       ...
9995    771
9996    516
9997    709
9998    772
9999    792
Name: CreditScore, Length: 10000, dtype: int64
```

In [39]:
```
b = data['Balance']
b
```

Out[39]:
```
0            0.00
1        83807.86
2       159660.80
3            0.00
4       125510.82
          ...
9995         0.00
9996     57369.61
9997         0.00
9998     75075.31
9999    130142.79
Name: Balance, Length: 10000, dtype: float64
```

In [40]:
```
plt.boxplot(b)
plt.show()
```

In [41]:
```
a = data['Age']
plt.boxplot(a)
plt.show()
```



In [42]:
```
ageOutliers =  np.where(df['Age'] > 60)
ageOutliers
```

Out[42]:
```
(array([  41,   43,   57,   84,  103,  157,  180,  229,  233,  242,  251,
         275,  309,  363,  370,  384,  386,  398,  415,  483,  537,  558,
         560,  566,  601,  611,  616,  629,  657,  677,  695,  735,  765,
         768,  806,  810,  822,  858,  883,  887,  920,  927,  946,  950,
         955,  961,  967,  995, 1007, 1037, 1038, 1053, 1112, 1116, 1190,
        1202, 1231, 1232, 1243, 1249, 1275, 1282, 1325, 1339, 1384, 1403,
        1406, 1429, 1435, 1453, 1515, 1539, 1584, 1603, 1610, 1637, 1785,
        1805, 1852, 1860, 1895, 1898, 1901, 1927, 1974, 1989, 1995, 2005,
        2032, 2046, 2071, 2087, 2096, 2101, 2147, 2152, 2157, 2237, 2254,
        2267, 2291, 2294, 2426, 2431, 2451, 2452, 2511, 2512, 2525, 2533,
        2545, 2590, 2606, 2650, 2661, 2704, 2708, 2751, 2763, 2768, 2769,
        2772, 2782, 2846, 2868, 2892, 2899, 2916, 2917, 2999, 3024, 3045,
        3101, 3133, 3157, 3183, 3194, 3220, 3296, 3299, 3302, 3305, 3308,
        3337, 3357, 3359, 3369, 3373, 3375, 3378, 3387, 3394, 3425, 3453,
        3488, 3490, 3518, 3522, 3532, 3540, 3550, 3554, 3564, 3566, 3584,
        3593, 3632, 3637, 3638, 3642, 3681, 3682, 3693, 3710, 3719, 3724,
        3752, 3765, 3804, 3817, 3871, 3872, 3879, 3900, 3901, 3918, 3931,
        3938, 3971, 3985, 4001, 4016, 4039, 4042, 4086, 4133, 4138, 4148,
        4153, 4161, 4232, 4235, 4247, 4264, 4271, 4288, 4304, 4309, 4326,
        4351, 4357, 4369, 4378, 4387, 4426, 4429, 4454, 4481, 4482, 4492,
        4497, 4550, 4554, 4581, 4586, 4635, 4669, 4689, 4738, 4742, 4792,
        4806, 4823, 4840, 4922, 4938, 4957, 4983, 4991, 5011, 5024, 5029,
        5059, 5123, 5127, 5139, 5150, 5188, 5214, 5216, 5226, 5246, 5290,
        5304, 5359, 5368, 5396, 5430, 5448, 5481, 5499, 5505, 5511, 5567,
        5568, 5572, 5630, 5642, 5646, 5651, 5655, 5662, 5674, 5689, 5733,
        5768, 5774, 5808, 5816, 5831, 5858, 5898, 5948, 5987, 6037, 6107,
        6143, 6157, 6158, 6162, 6164, 6203, 6221, 6269, 6280, 6306, 6348,
        6357, 6364, 6366, 6401, 6434, 6506, 6521, 6523, 6572, 6603, 6617,
        6697, 6700, 6706, 6712, 6750, 6754, 6803, 6890, 6961, 6988, 6999,
        7048, 7049, 7054, 7062, 7069, 7085, 7129, 7130, 7133, 7147, 7185,
        7193, 7229, 7234, 7263, 7293, 7353, 7366, 7383, 7490, 7505, 7514,
        7517, 7539, 7543, 7614, 7615, 7620, 7659, 7678, 7683, 7685, 7700,
        7706, 7710, 7711, 7718, 7764, 7767, 7775, 7779, 7793, 7804, 7842,
        7885, 7889, 7900, 7924, 7947, 7986, 8010, 8028, 8085, 8089, 8146,
        8160, 8183, 8197, 8205, 8207, 8294, 8311, 8375, 8384, 8434, 8448,
        8457, 8459, 8468, 8478, 8552, 8558, 8567, 8592, 8664, 8676, 8679,
        8701, 8748, 8750, 8751, 8756, 8775, 8781, 8810, 8853, 8888, 8905,
        8918, 8958, 9006, 9009, 9050, 9068, 9090, 9100, 9104, 9150, 9162,
        9211, 9249, 9267, 9272, 9280, 9297, 9306, 9309, 9312, 9320, 9321,
        9339, 9367, 9378, 9389, 9412, 9415, 9425, 9459, 9477, 9493, 9542,
        9544, 9569, 9574, 9576, 9580, 9582, 9632, 9657, 9659, 9667, 9672,
        9674, 9704, 9719, 9720, 9722, 9733, 9739, 9751, 9818, 9865, 9880,
        9883, 9922], dtype=int64),)
```

In [43]:
```
da = data[data['Age'] <=60 ]
da
```

Out[43]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMem
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	

9536 rows × 14 columns

In [44]:
```
from pandas.api.types import is_string_dtype
continuous=[]
categorical=[]
for data1 in data:
    if is_string_dtype(data[data1]):
        categorical.append(data1)
    else:
        continuous.append(data1)
categorical
```

Out[44]:
```
['Surname', 'Geography', 'Gender']
```

1. Split the data into dependent and independent variables.

In [46]:
```
x = data.iloc[:, 0:1].values
y = data.iloc[:, 1]
x
y
```

Out[46]:
```
0       15634602
1       15647311
2       15619304
3       15701354
4       15737888
          ...
9995    15606229
9996    15569892
9997    15584532
9998    15682355
9999    15628319
Name: CustomerId, Length: 10000, dtype: int64
```

9.Scale the independent variables

In [47]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x)
x_test = sc.transform(x)

x_train=pd.DataFrame(x_train)
x_train.head()
```

Out[47]:

	0
0	-1.731878
1	-1.731531
2	-1.731185
3	-1.730838
4	-1.730492

10.Split the data into training and testing

In [48]:
```
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 [49]:
```
data.shape
```

Out[49]:
```
(10000, 14)
```

In [50]:
```
x_test
```

Out[50]:
```
array([[9395],
       [ 899],
       [2399],
       ...,
       [9550],
       [2741],
       [6691]], dtype=int64)
```

In [51]:
```
y_train
```

Out[51]:
```
7389    15676909
9275    15749265
2995    15582492
5316    15780386
356     15611759
          ...
9225    15584928
4859    15647111
3264    15574372
9845    15664035
2732    15592816
Name: CustomerId, Length: 8000, dtype: int64
```

In [52]:
```
y_test
```

Out[52]:
```
9394    15615753
898     15654700
2398    15633877
5906    15745623
2343    15765902
          ...
1037    15631054
2899    15810944
9549    15772604
2740    15787699
6690    15579223
Name: CustomerId, Length: 2000, dtype: int64
```