Question-1. Download dataset

Question-2. Load the dataset

```
path = '/content/drive/MyDrive/Colab Notebooks/Churn_Modelling.csv'

import pandas as pd


data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Churn_Modelling.csv')

data.head
```

```
<bound method NDFrame.head of       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
```

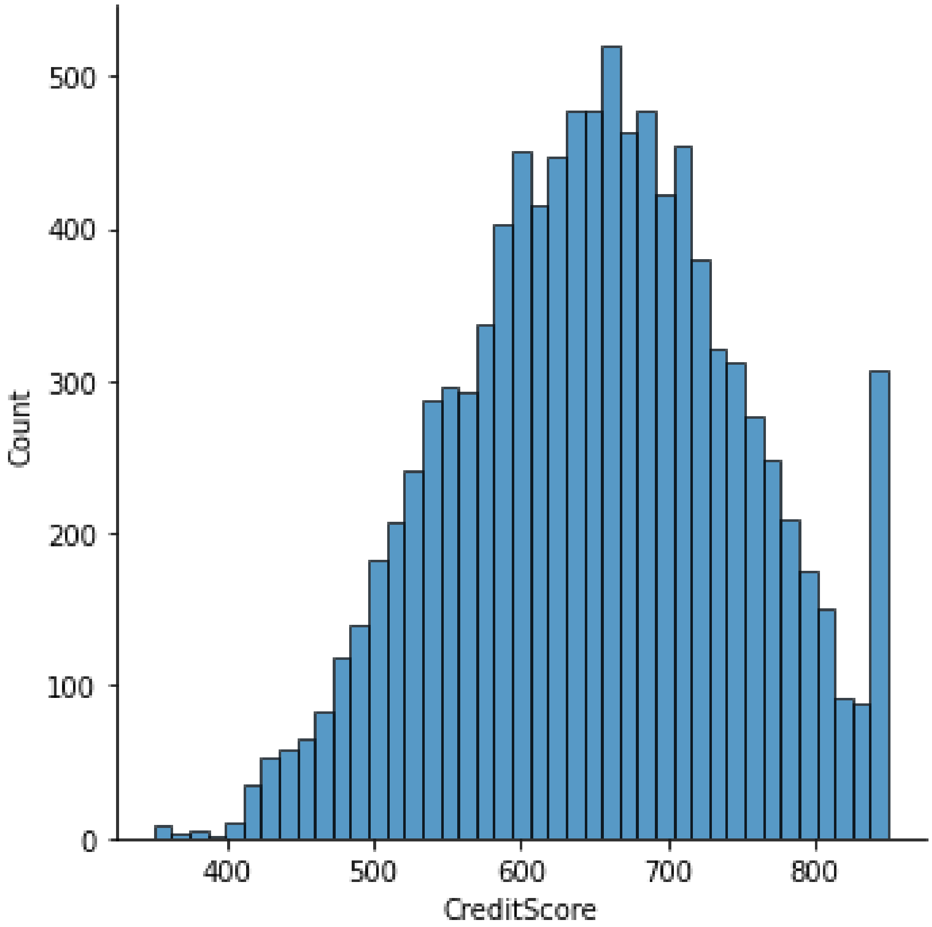
```
      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]>
```

Question-3. Perform Below Visualizations.

3.1 univariate analysis

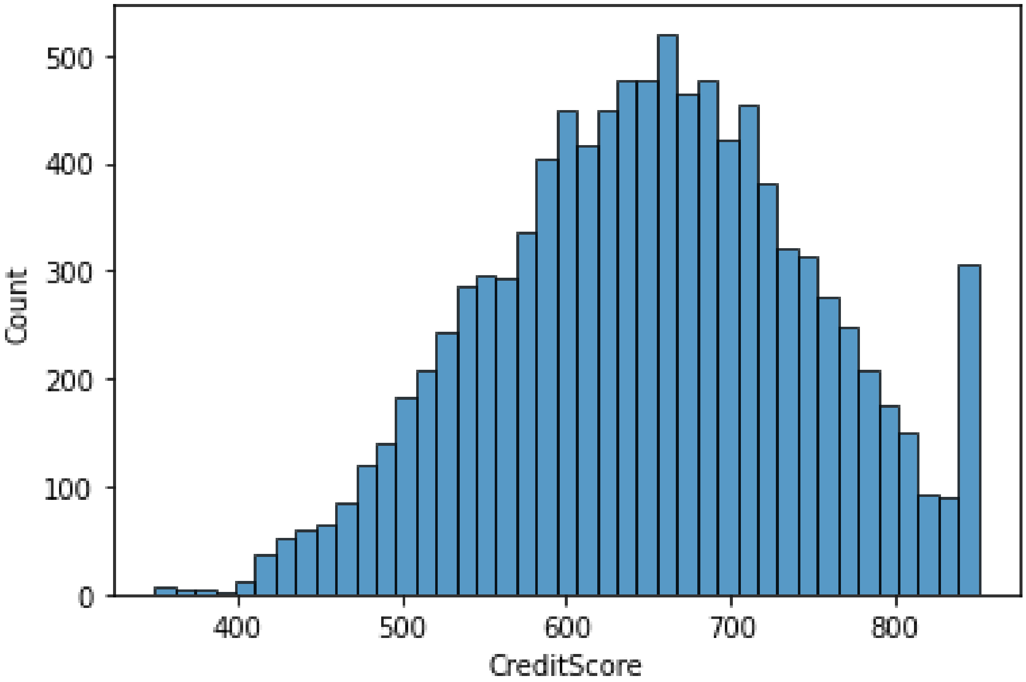
```
sns.displot(data['CreditScore'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7ff47b124f50>
```



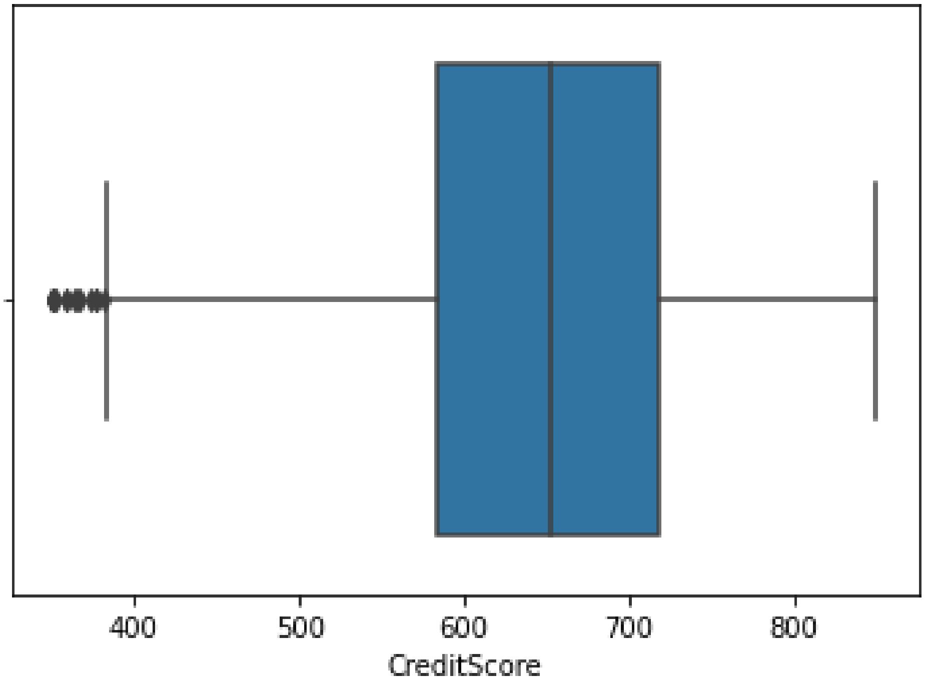
```
sns.histplot(data['CreditScore'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47aec4a50>
```



```
sns.boxplot(x = data['CreditScore'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff4780406d0>
```

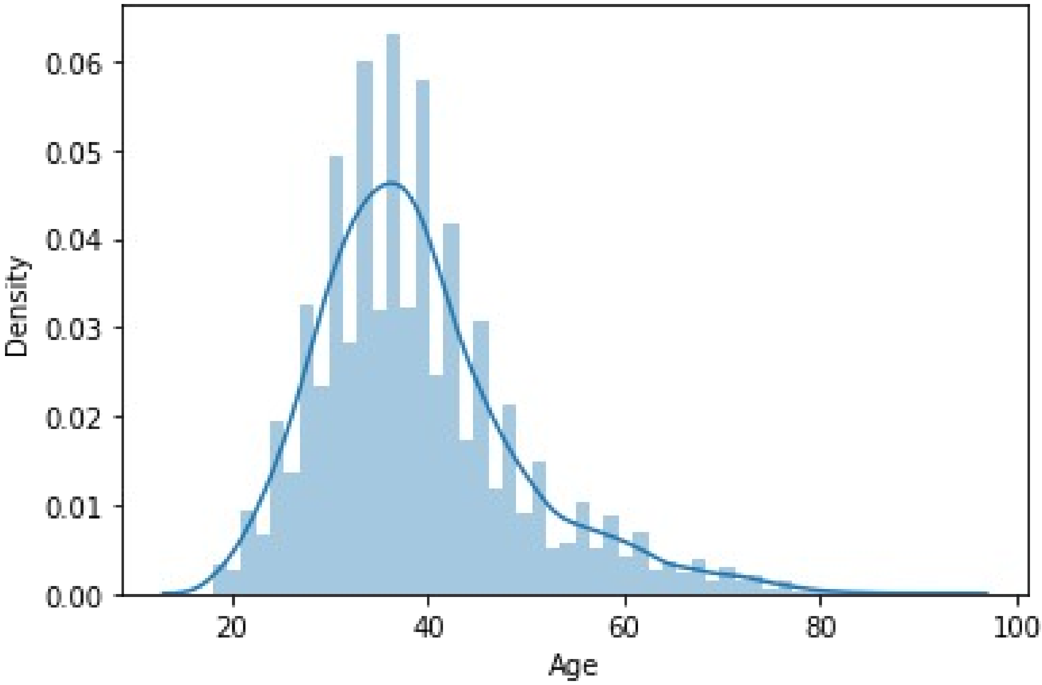


```
sns.distplot(data['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
```

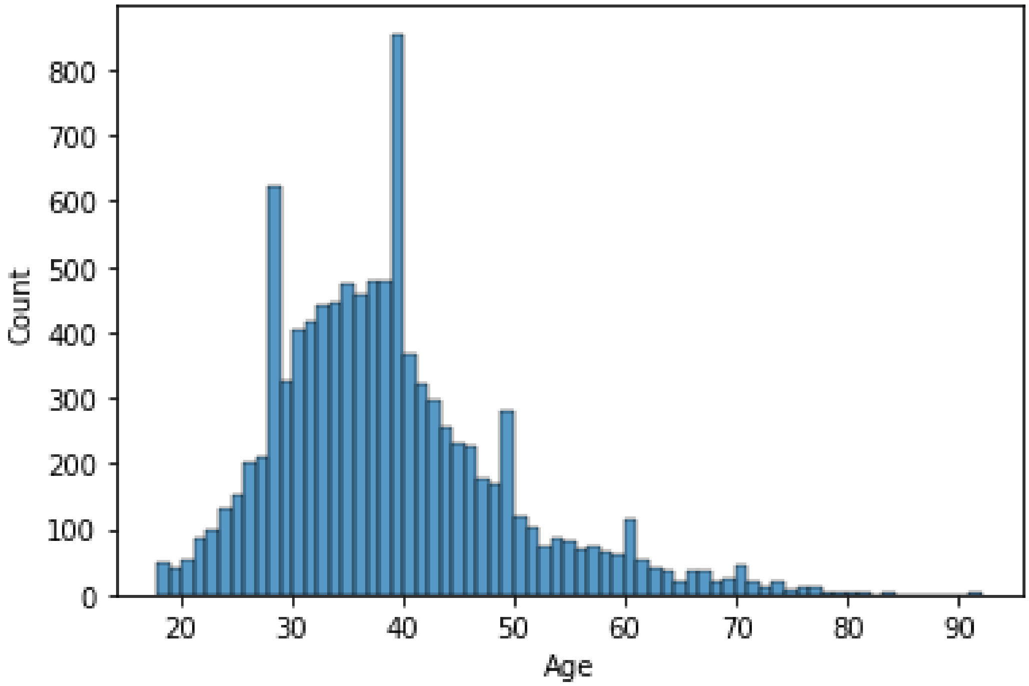
```
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47af787d0>
```



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

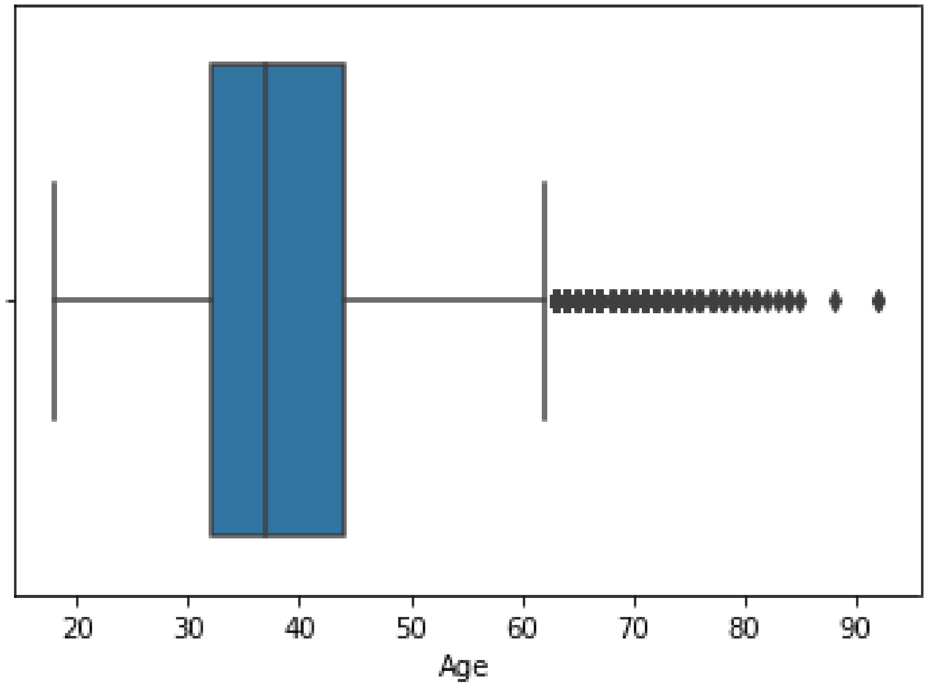
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff477ed9d10>
```



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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

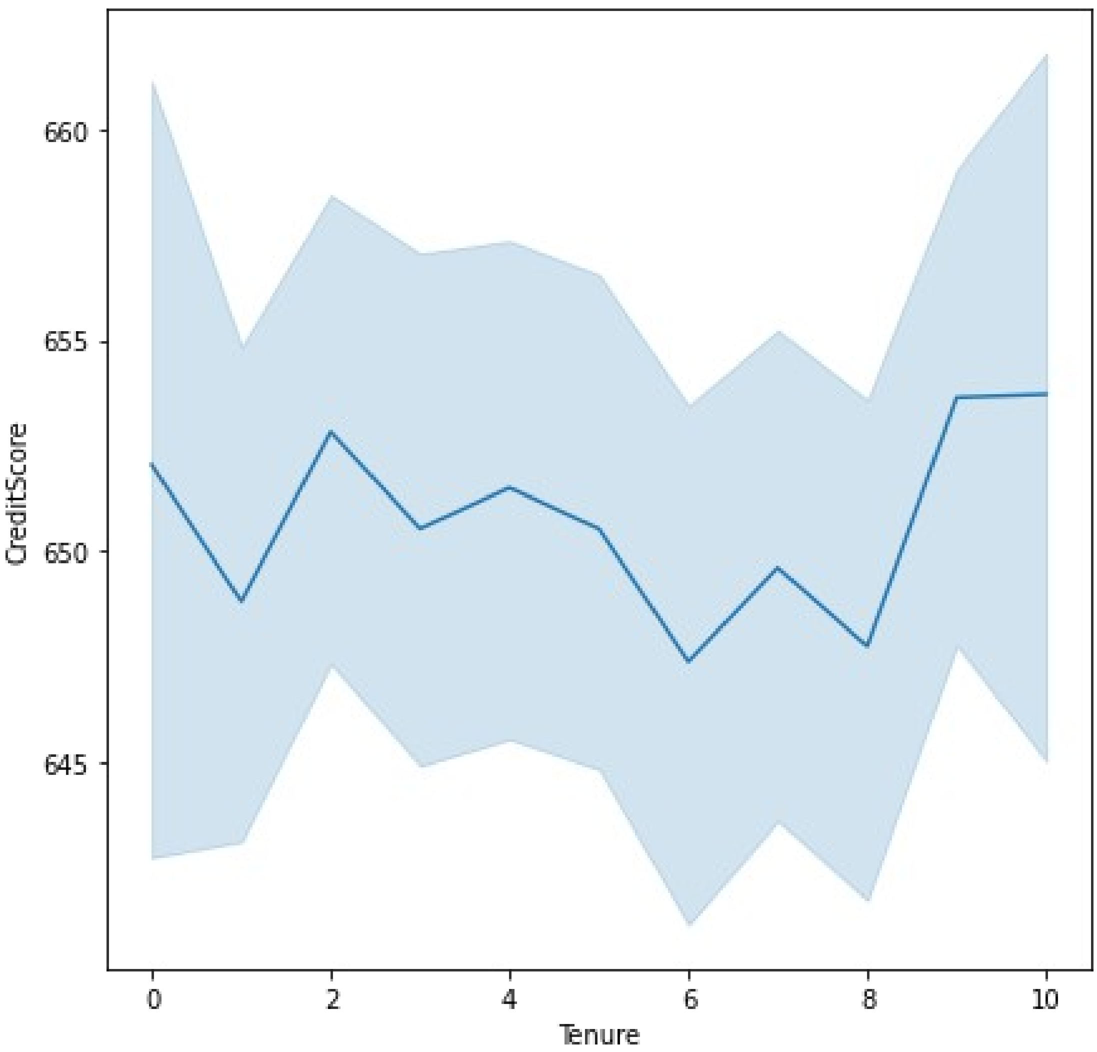
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff477d9ea90>
```



3.2 Bi-variate Analysis

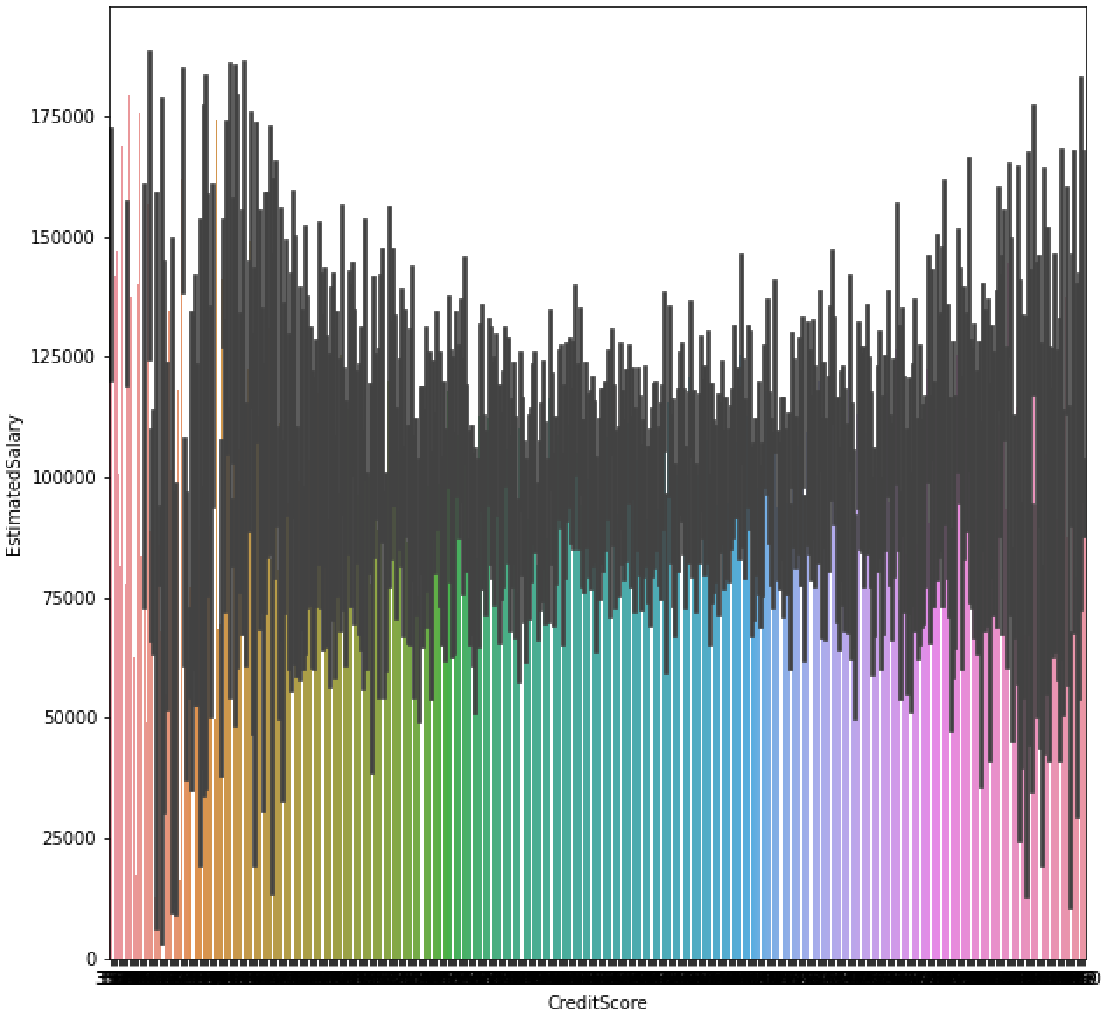
```
plt.figure(figsize=(7,7))
sns.lineplot(data = data, x = 'Tenure', y = 'CreditScore')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47afc7750>
```



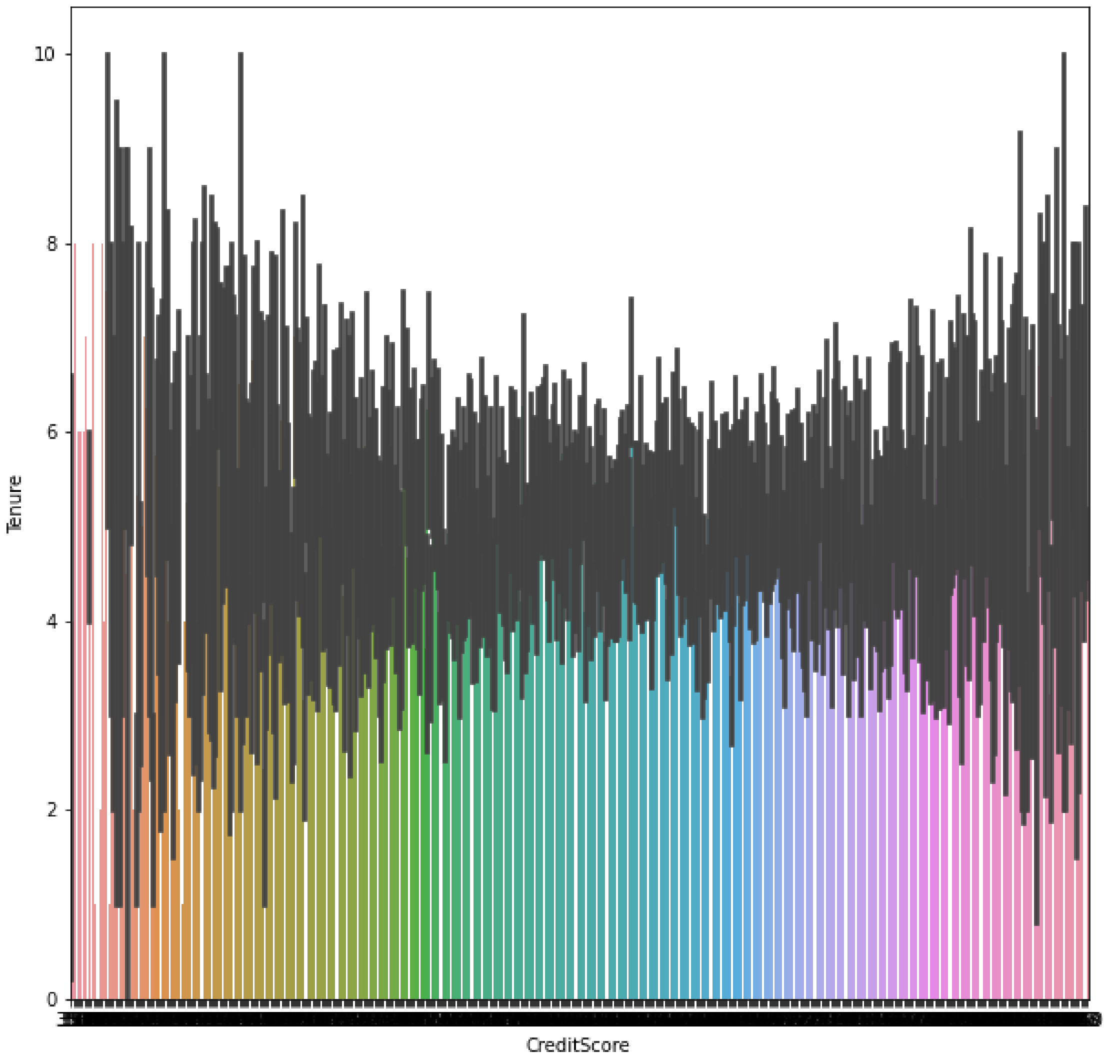
```
plt.figure(figsize=(10,10))
sns.barplot(data = data, x = 'CreditScore', y ='EstimatedSalary')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff477da5e10>
```



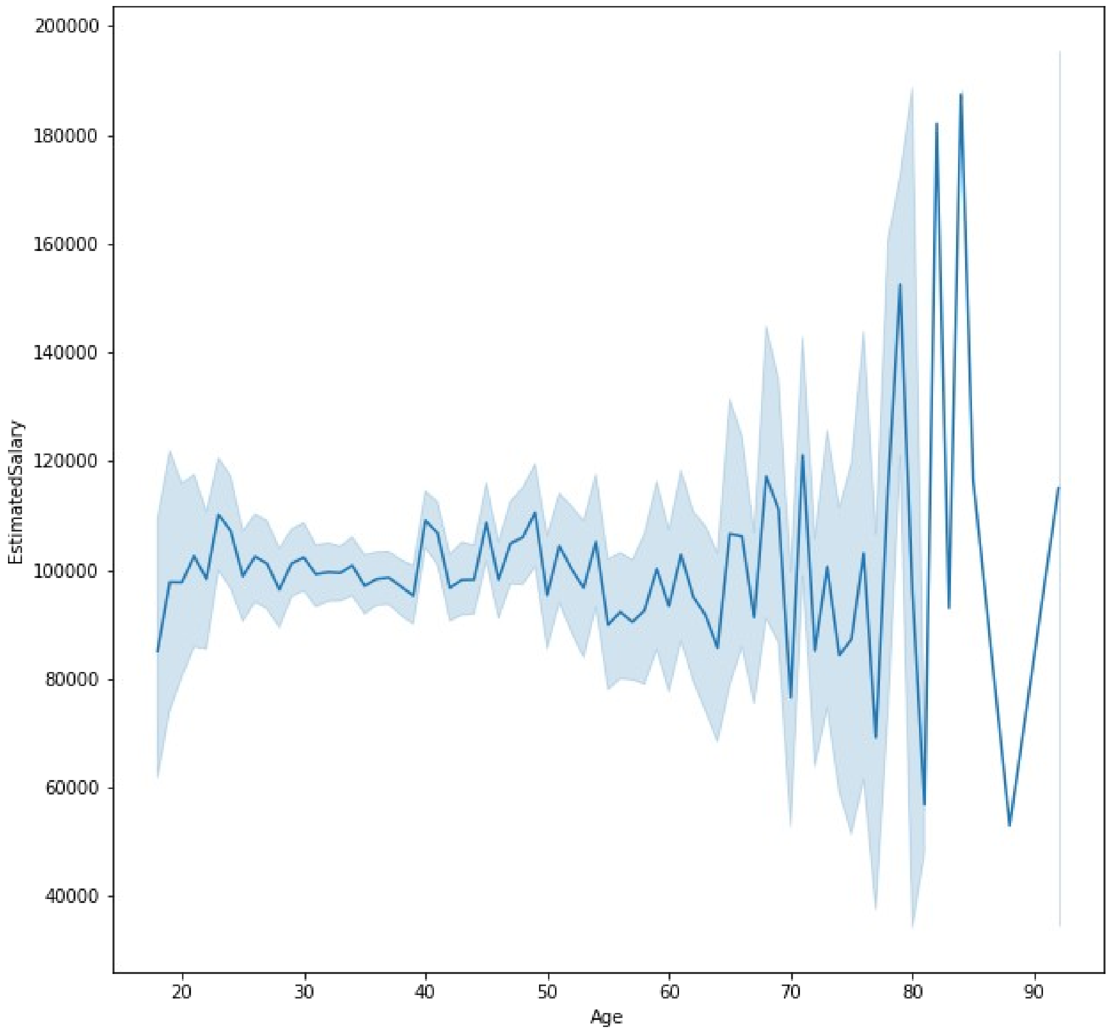
```
plt.figure(figsize=(10,10))
sns.barplot(data = data, x = 'CreditScore', y = 'Tenure')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47b1b3cd0>
```



```
plt.figure(figsize=(10,10))
sns.lineplot(data['Age'], data['EstimatedSalary'])
```

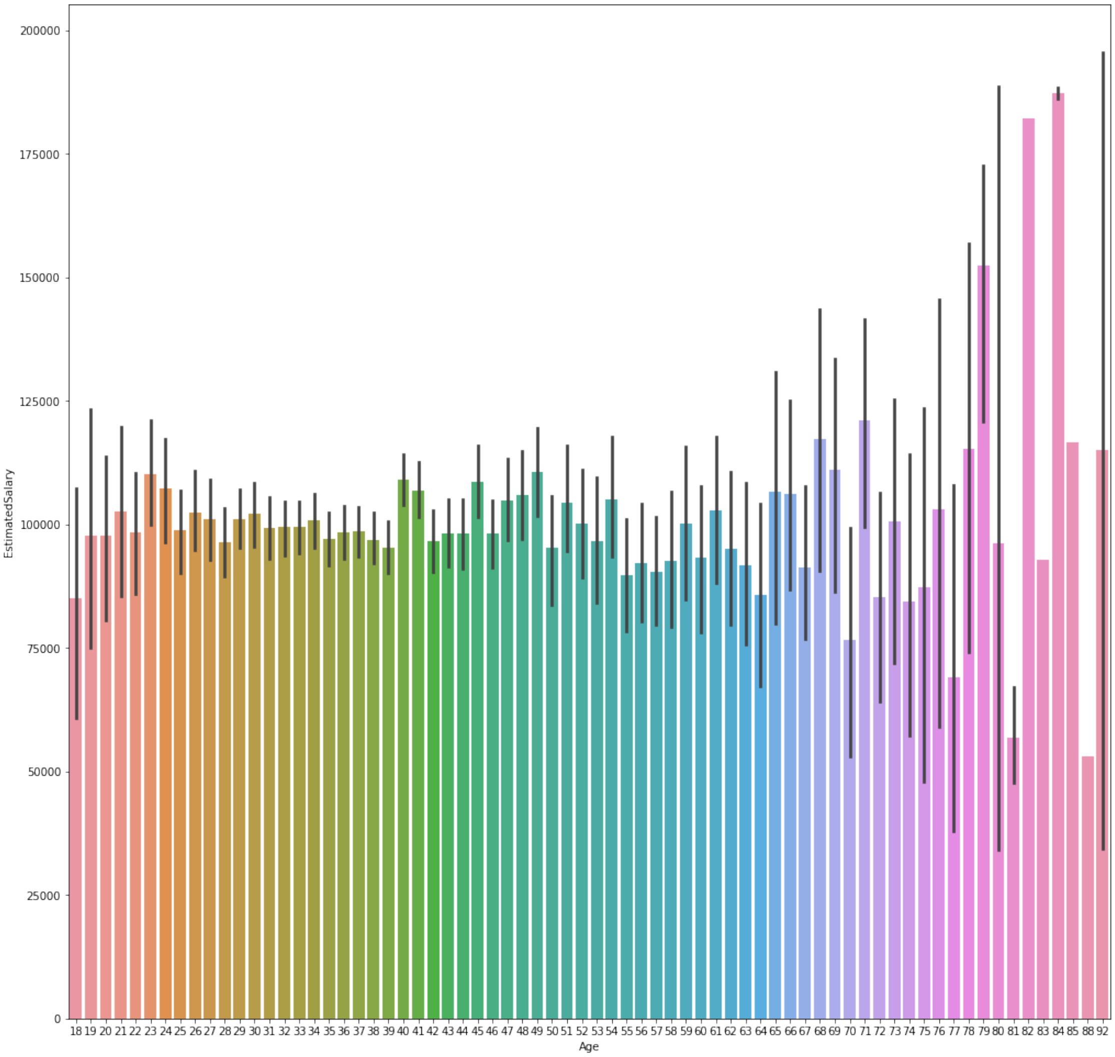
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning

<matplotlib.axes._subplots.AxesSubplot at 0x7ff475933690>
```



```
plt.figure(figsize=(17,17))
sns.barplot(data['Age'], data['EstimatedSalary'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
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.
  FutureWarning
```

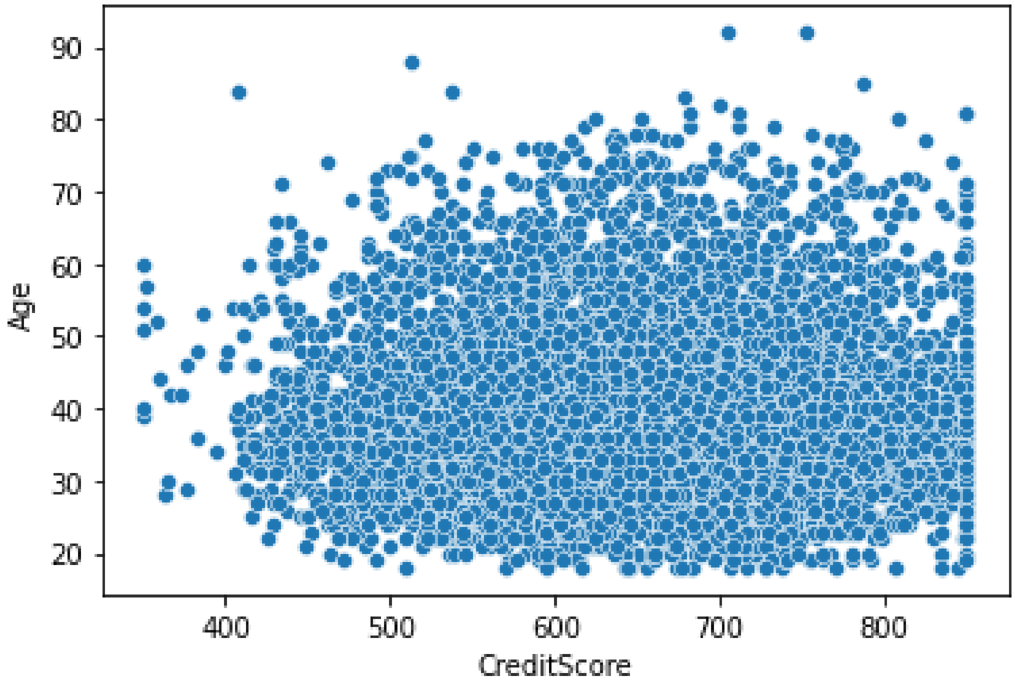
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff4749085d0>
```



3.3 Multivariate Analysis

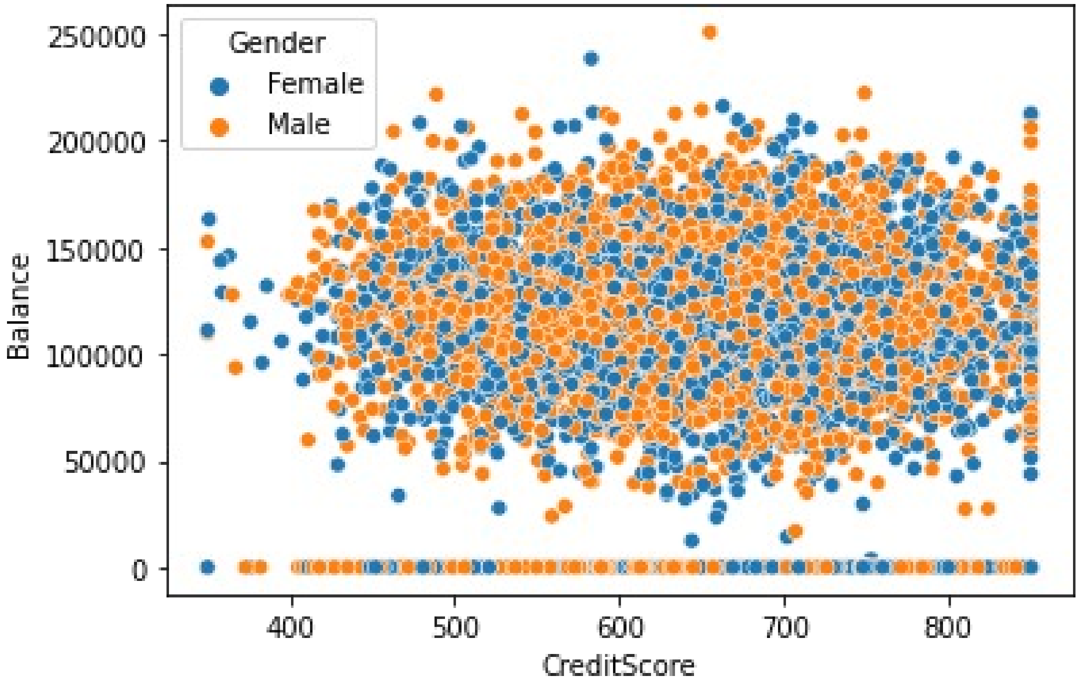
```
sns.scatterplot(data = data, x = 'CreditScore', y = 'Age')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff472f36a10>
```



```
sns.scatterplot(data = data, x = 'CreditScore', y = 'Balance', hue =
'Gender')
```

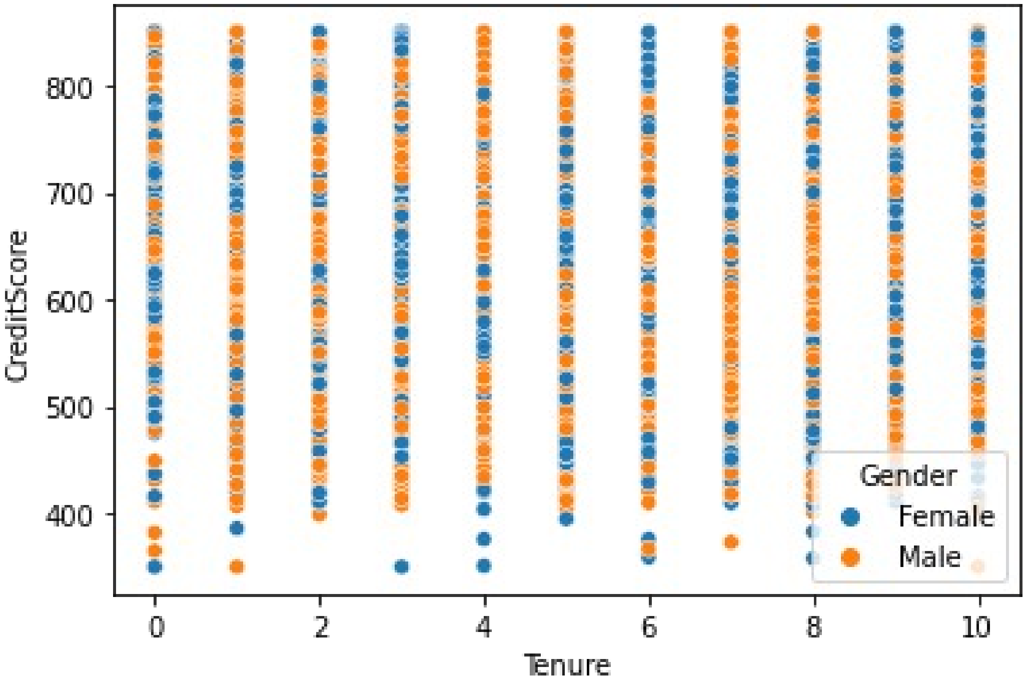
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff472e6d3d0>
```



```
sns.scatterplot(data['Tenure'], data['CreditScore'], hue =
data['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
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.
  FutureWarning
```

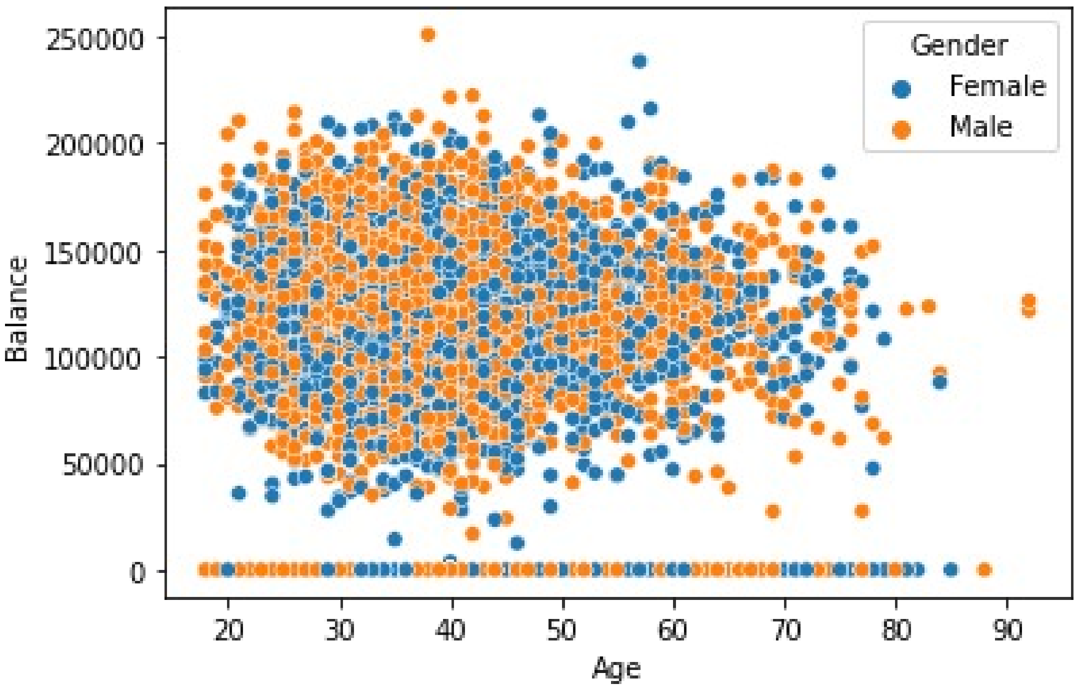
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff472ef03d0>
```



```
sns.scatterplot(data['Age'], data['Balance'], hue = data['Gender'])
```

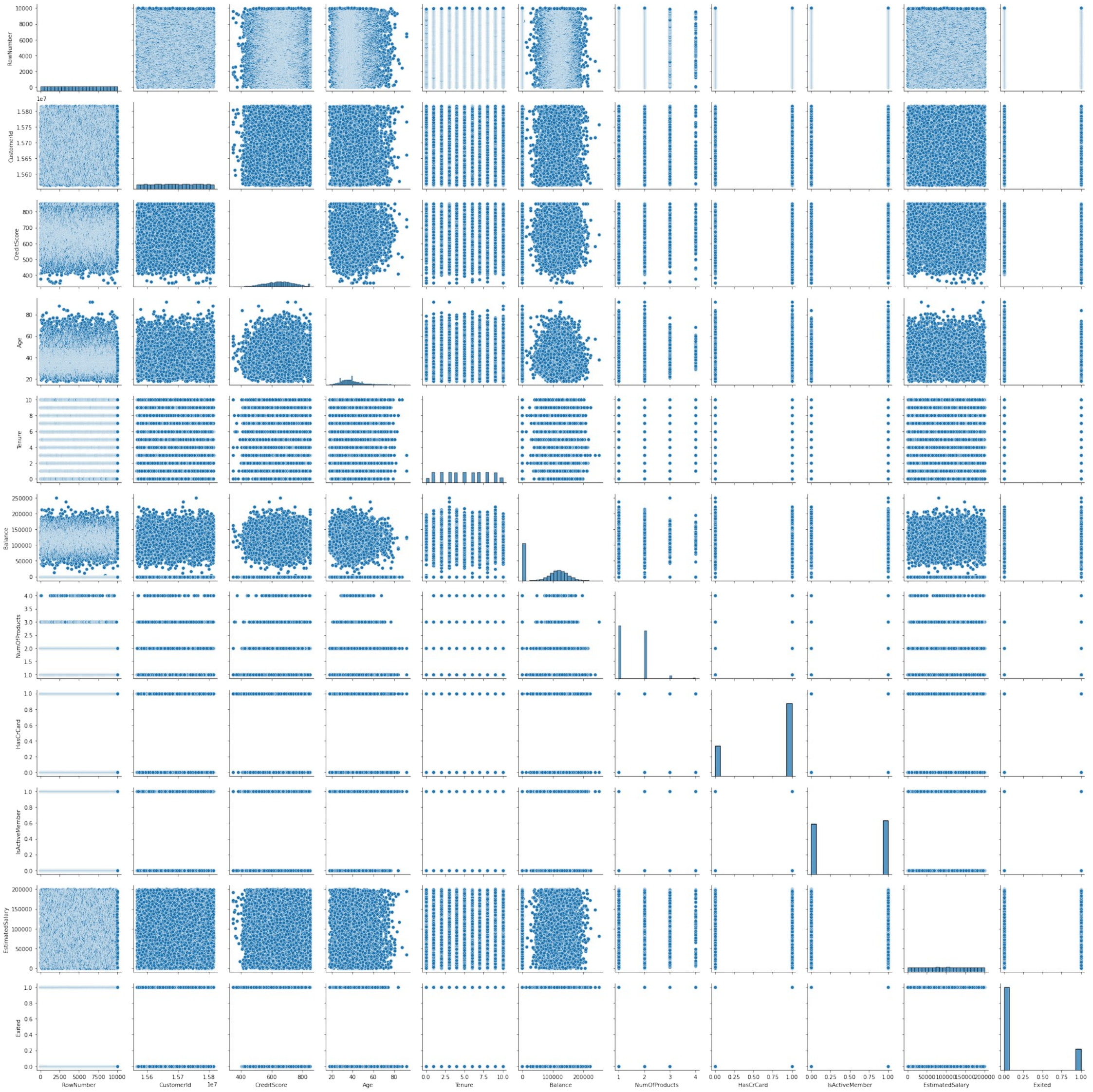
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff472dacf10>
```



```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7ff472db8650>
```



Question-4. Perform descriptive statistics on the dataset

```
data.mean(numeric_only = True)
```

```
RowNumber          5.000500e+03
CustomerId         1.569094e+07
CreditScore        6.505288e+02
Age                3.892180e+01
Tenure             5.012800e+00
Balance            7.648589e+04
NumOfProducts      1.530200e+00
HasCrCard          7.055000e-01
IsActiveMember     5.151000e-01
EstimatedSalary    1.000902e+05
Exited             2.037000e-01
dtype: float64
```

```
data.median(numeric_only = True)
```

```
RowNumber          5.000500e+03
CustomerId         1.569074e+07
CreditScore        6.520000e+02
Age                3.700000e+01
Tenure             5.000000e+00
Balance            9.719854e+04
NumOfProducts      1.000000e+00
HasCrCard          1.000000e+00
IsActiveMember     1.000000e+00
EstimatedSalary    1.001939e+05
Exited             0.000000e+00
dtype: float64
```

```
data['CreditScore'].mode()
```

```
0    850
dtype: int64
```

```
data['EstimatedSalary'].mode()
```

```
0    24924.92
dtype: float64
```

```
data['HasCrCard'].unique()
```

```
array([1, 0])
```

```
data['Tenure'].unique()
```

```
array([ 2,  1,  8,  7,  4,  6,  3, 10,  5,  9,  0])
```

```
data.std(numeric_only=True)
```

```
RowNumber           2886.895680
CustomerId         71936.186123
CreditScore           96.653299
Age                   10.487806
Tenure                 2.892174
Balance            62397.405202
NumOfProducts          0.581654
HasCrCard              0.455840
IsActiveMember         0.499797
EstimatedSalary    57510.492818
Exited                 0.402769
dtype: float64
```

```
data.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
```

Question-5. Handle the Missing values.

```
data.isnull().any()
```

```
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
```

```
data.isnull().sum()
```

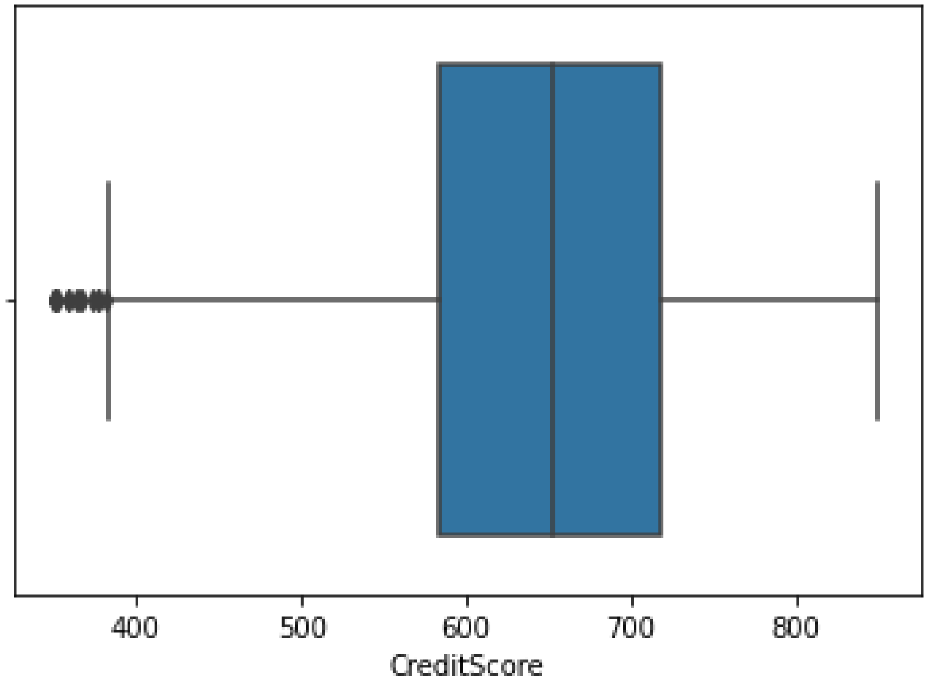
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Question-6. Find the outliers and replace the outliers

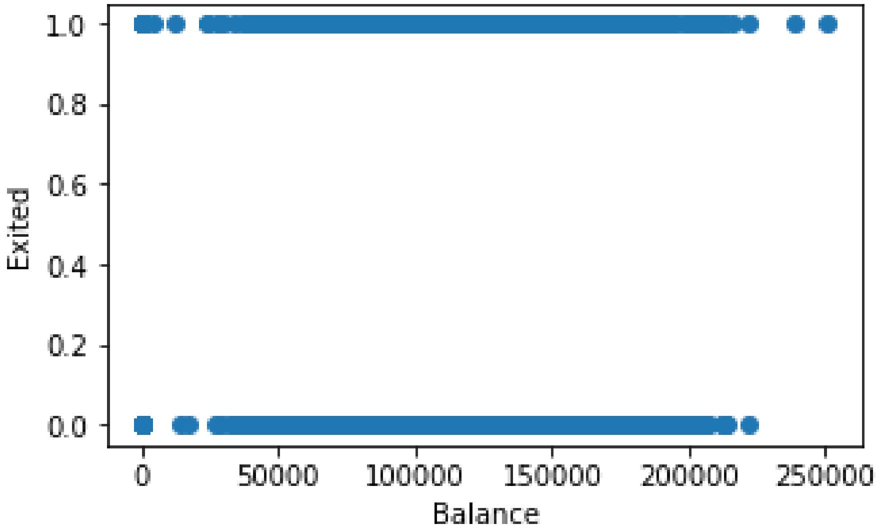
```
sns.boxplot(data['CreditScore']) #Outlier detection - box plot
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

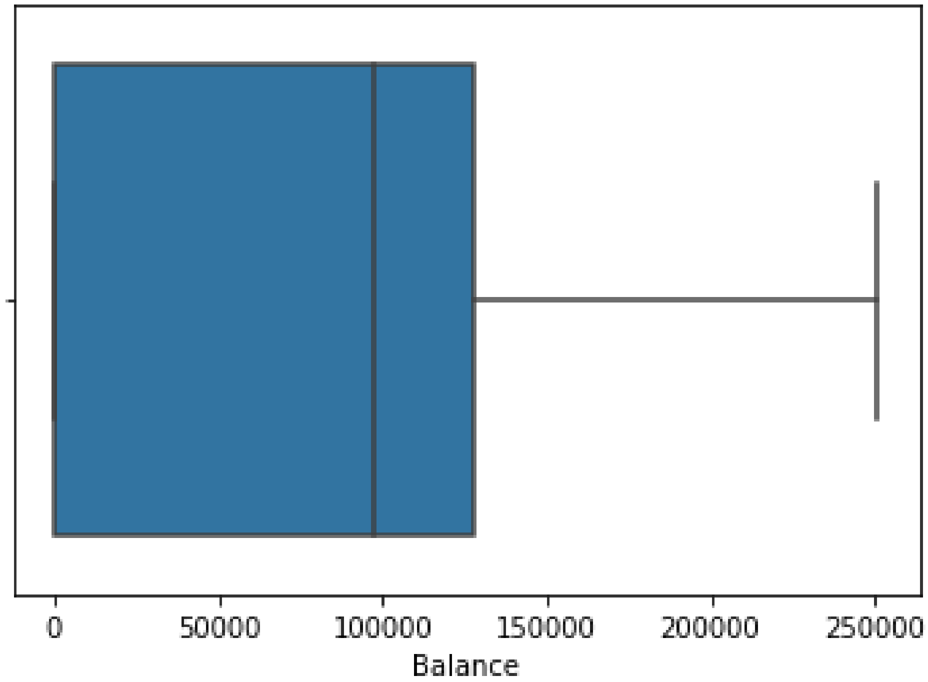
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff46fbdf3d0>
```



```
fig, ax = plt.subplots(figsize = (5,3)) #Outlier detection - Scatter
plot
ax.scatter(data['Balance'], data['Exited'])
# x-axis label
ax.set_xlabel('Balance')
# y-axis label
ax.set_ylabel('Exited')
plt.show()
sns.boxplot(x=data['Balance'])
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47177b710>
```



```
from scipy import stats #Outlier detection - zscore
zscore = np.abs(stats.zscore(data['CreditScore']))
print(zscore)
print('No. of Outliers : ', np.shape(np.where(zscore>3)))
```

```
0       0.326221
1       0.440036
2       1.536794
3       0.501521
4       2.063884
          ...
9995    1.246488
9996    1.391939
9997    0.604988
9998    1.256835
9999    1.463771
Name: CreditScore, Length: 10000, dtype: float64
No. of Outliers :  (1, 8)
```

```
q = data.quantile([0.75,0.25])
q
```

```
      RowNumber   CustomerId  CreditScore   Age  Tenure    Balance  \
0.75    7500.25  15753233.75        718.0  44.0     7.0  127644.24   
0.25    2500.75  15628528.25        584.0  32.0     3.0       0.00   

      NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  
Exited  
0.75            2.0        1.0             1.0      149388.2475  
```

```
0.0
0.25            1.0        0.0             0.0       51002.1100
0.0
```

```
iqr = q.iloc[0] - q.iloc[1]
iqr
```

```
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
```

```
u = q.iloc[0] + (1.5*iqr)
u
```

```
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
```

```
l = q.iloc[1] - (1.5*iqr)
l
```

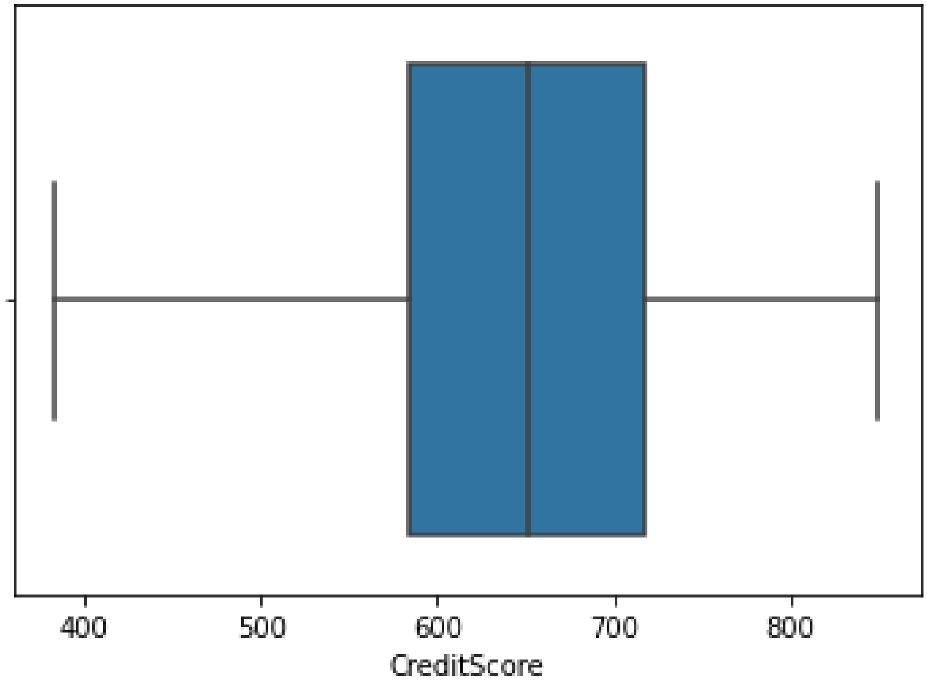
```
RowNumber         -4.998500e+03
CustomerId         1.544147e+07
CreditScore        3.830000e+02
Age                1.400000e+01
Tenure            -3.000000e+00
Balance           -1.914664e+05
NumOfProducts     -5.000000e-01
HasCrCard         -1.500000e+00
IsActiveMember    -1.500000e+00
EstimatedSalary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

```
Q1 = data['EstimatedSalary'].quantile(0.25) #Outlier detection - IQR
Q3 = data['EstimatedSalary'].quantile(0.75)
iqr = Q3 - Q1
print(iqr)
upper=Q3 + 1.5 * iqr
lower=Q1 - 1.5 * iqr
count = np.size(np.where(data['EstimatedSalary'] >upper))
count = count + np.size(np.where(data['EstimatedSalary'] <lower))
print('No. of outliers : ', count)
```

```
98386.1375
No. of outliers :  0
```

```
data['CreditScore'] = np.where(np.logical_or(data['CreditScore']>900,
data['CreditScore']<383), 650, data['CreditScore'])
sns.boxplot(data['CreditScore'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning

<matplotlib.axes._subplots.AxesSubplot at 0x7ff47b1b5910>
```



```
upper = data.Age.mean() + (3 * data.Age.std()) #Outlier detection - 3
sigma
lower = data.Age.mean() - (3 * data.Age.std())
columns = data[ ( data['Age'] > upper ) | ( data['Age']<lower ) ]
```

```
print('Upper range : ', upper)
print('Lower range : ', lower)
print('No. of Outliers : ', len(columns))
```

```
Upper range :  70.38521935511383
Lower range :  7.458380644886169
No. of Outliers :  133
```

```
columns = ['EstimatedSalary', 'Age', 'Balance', 'NumOfProducts',
'Tenure', 'CreditScore']
#After outlier removal
for i in columns:
 Q1 = data[i].quantile(0.25)
 Q3 = data[i].quantile(0.75)
 iqr = Q3 - Q1
 upper=Q3 + 1.5 * iqr
 lower=Q1 - 1.5 * iqr
 count = np.size(np.where(data[i] >upper))
 count = count + np.size(np.where(data[i] <lower))
 print('No. of outliers in ', i, ' : ', count)
```

```
No. of outliers in  EstimatedSalary  :  0
No. of outliers in  Age  :  359
No. of outliers in  Balance  :  0
No. of outliers in  NumOfProducts  :  60
No. of outliers in  Tenure  :  0
No. of outliers in  CreditScore  :  0
```

Question-6. Find the outliers and replace the outliers

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
oneh = OneHotEncoder()
data['Surname'] = le.fit_transform(data['Surname'])
data['Gender'] = le.fit_transform(data['Gender'])
data['Geography'] = le.fit_transform(data['Geography'])
data.head()
```

```
   RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age
\
0          1    15634602     1115          619          0       0   42

1          2    15647311     1177          608          2       0   41

2          3    15619304     2040          502          0       0   42

3          4    15701354      289          699          0       0   39

4          5    15737888     1822          850          2       0   43
```

```
   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

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

Question-8. Split the data into dependent and independent variables split the data in X and Y

```
x = data.iloc[:, 0:13]
x # independent values ( inputs)
```

```
      RowNumber  CustomerId  Surname  CreditScore  Geography  Gender
Age  \
0             1    15634602     1115          619          0       0
42
1             2    15647311     1177          608          2       0
41
2             3    15619304     2040          502          0       0
42
3             4    15701354      289          699          0       0
39
4             5    15737888     1822          850          2       0
43
...         ...         ...      ...          ...        ...     ...
...
9995       9996    15606229     1999          771          0       1
39
9996       9997    15569892     1336          516          0       1
35
9997       9998    15584532     1570          709          0       0
36
9998       9999    15682355     2345          772          1       1
42
9999      10000    15628319     2751          792          0       0
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
0           101348.88
1           112542.58
2           113931.57
3            93826.63
4            79084.10
...               ...
9995         96270.64
9996        101699.77
9997         42085.58
9998         92888.52
9999         38190.78

[10000 rows x 13 columns]
```

```
y = data['Exited']
y # dependent values (output)
```

```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64
```

Question-9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
sc = StandardScaler()
x_scaled = sc.fit_transform(x)
x_scaled
```

```
array([[-1.73187761, -0.78321342, -0.46418322, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-1.7315312 , -0.60653412, -0.3909112 , ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.73118479, -0.99588476,  0.62898807, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 1.73118479, -1.47928179,  0.07353887, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 1.7315312 , -0.11935577,  0.98943914, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 1.73187761, -0.87055909,  1.4692527 , ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

Question-10. Split x and y into Training and Testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y,
test_size = 0.3, random_state = 0)
```

```
x_train
```

```
array([[ 0.92889885, -0.79703192, -1.47580983, ...,  0.64609167,
          0.97024255, -0.77021814],
       [ 1.39655257,  0.71431365, -1.58808148, ...,  0.64609167,
         -1.03067011, -1.39576675],
       [-0.4532777 ,  0.96344969, -0.24082173, ..., -1.54776799,
          0.97024255, -1.49965629],
       ...,
       [-0.60119484, -1.62052514, -0.36136603, ...,  0.64609167,
         -1.03067011,  1.41441489],
       [ 1.67853045, -0.37403866,  0.72589622, ...,  0.64609167,
          0.97024255,  0.84614739],
       [-0.78548505, -1.36411841,  1.3829808 , ...,  0.64609167,
         -1.03067011,  0.32630495]])
```

```
x_train.shape
```

```
(7000, 13)
```

```
x_test
```

```
array([[ 1.52229946, -1.04525042,  1.39834429, ...,  0.64609167,
          0.97024255,  1.61304597],
       [-1.42080128, -0.50381294, -0.78208925, ...,  0.64609167,
         -1.03067011,  0.49753166],
       [-0.90118604, -0.7932923 ,  0.41271742, ...,  0.64609167,
          0.97024255, -0.4235611 ],
       ...,
       [ 1.49216178, -0.14646448,  0.6868966 , ...,  0.64609167,
          0.97024255,  1.17045451],
       [ 1.1758893 , -1.29228727, -1.38481071, ...,  0.64609167,
          0.97024255, -0.50846777],
       [ 0.08088677, -1.38538833,  1.11707427, ...,  0.64609167,
          0.97024255, -1.15342685]])
```

```
x_test.shape
```

```
(3000, 13)
```

```
y_train
```

```
7681    1
9031    0
3691    0
202     1
5625    0
       ..
9225    0
4859    0
3264    0
9845    0
2732    1
Name: Exited, Length: 7000, dtype: int64
```

```
y_test
```

```
9394    0
898     1
2398    0
5906    0
2343    0
       ..
4004    0
7375    0
9307    0
8394    0
5233    1
Name: Exited, Length: 3000, dtype: int64
```