

Assignment -2

Assignment Date	27 September 2022	
Team ID	PNT2022TMID32006	
Project Name	DemandEst- AI Powered Food Demand Forecaster	
Student Name	SAI SIVA SANJAY R	
Student Roll Number	731619104051	
Maximum Marks	2 Marks	

Question-1. Download dataset

Solution:

RowNum	Customer Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfPr	HasCrCard	IsActiveM	Estimated	Exited
1	15634602 Hargrave	619	France	Female	42	2	0	1	1	1	101348.9	1
2	15647311 Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.6	0
3	15619304 Onio	502	France	Female	42	8	159660.8	3	1	0	113931.6	1
4	15701354 Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888 Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012 Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.7	1
7	15592531 Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148 Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.9	1
9	15792365 He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389 H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821 Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173 Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264 Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483 Chin	549	France	Female	25	5	0	2	0	0	190857.8	0
15	15600882 Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966 Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452 Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218 Henderso	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507 Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.8	0
20	15568982 Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
21	15577657 McDonald	732	France	Male	41	8	0	2	1	1	170886.2	0
22	15597945 Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.5	0
23	15699309 Gerasimo	510	Spain	Female	38	4	0	1	1	0	118913.5	1
24	15725737 Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
25	15625047 Yen	846	France	Female	38	5	0	1	1	1	187616.2	0
26	15738191 Maclean	577	France	Male	25	3	0	2	0	1	124508.3	0
27	15736816 Young	756	Germany	Male	36	2	136815.6	1	1	1	170042	0
28	15700772 Nebechi	571	France	Male	44	9	0	2	0	0	38433.35	0
29	15728693 McWilliam	574	Germany	Female	43	3	141349.4	1	1	1	100187.4	0
30	15656300 Lucciano	411	France	Male	29	0	59697.17	2	1	1	53483.21	0
31	15589475 Azikiwe	591	Spain	Female	39	3	0	3	1	0	140469.4	1
32	15706552 Odinakach	533	France	Male	36	7	85311.7	1	0	1	156731.9	0
33	15750181 Sandersor	553	Germany	Male	41	9	110112.5	2	0	0	81898.81	0
34	15659428 Maggard	520	Spain	Female	42	6	0	2	1	1	34410.55	0
35	15732963 Clements	722	Spain	Female	29	9	0	2	1	1	142033.1	0
36	15794171 Lombardo	475	France	Female	45	0	134264	1	1	0	27822.99	1
37	15788448 Watson	490	Spain	Male	31	3	145260.2	1	0	1	114066.8	0
38	15729599 Lorenzo	804	Spain	Male	33	7	76548.6	1	0	1	98453.45	0
39	15717426 Armstrong	850	France	Male	36	7	0	1	1	1	40812.9	0
40	15585768 Cameron	582	Germany	Male	41	6	70349.48	2	0	1	178074	0

Question-2. Load the dataset

Solution:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
data = pd.read_csv(r'Churn_Modelling.csv')
df.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  Obijaku      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

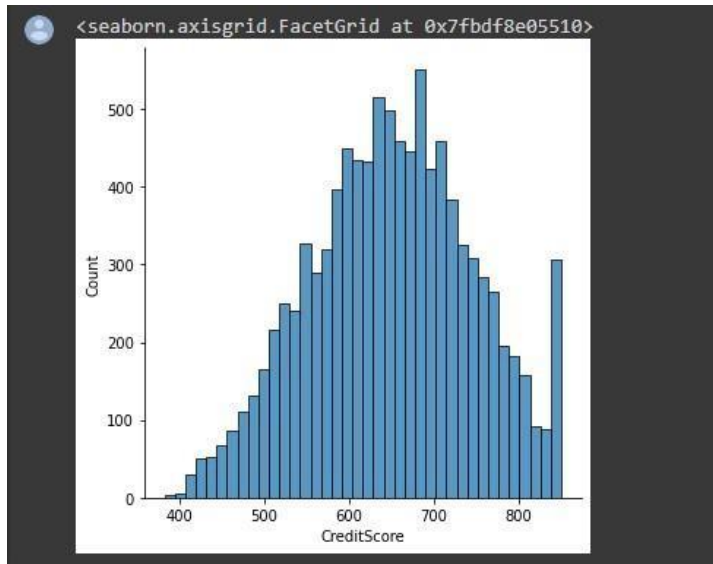
[10000 rows x 14 columns]>
```

Question-3. Perform Below Visualizations.

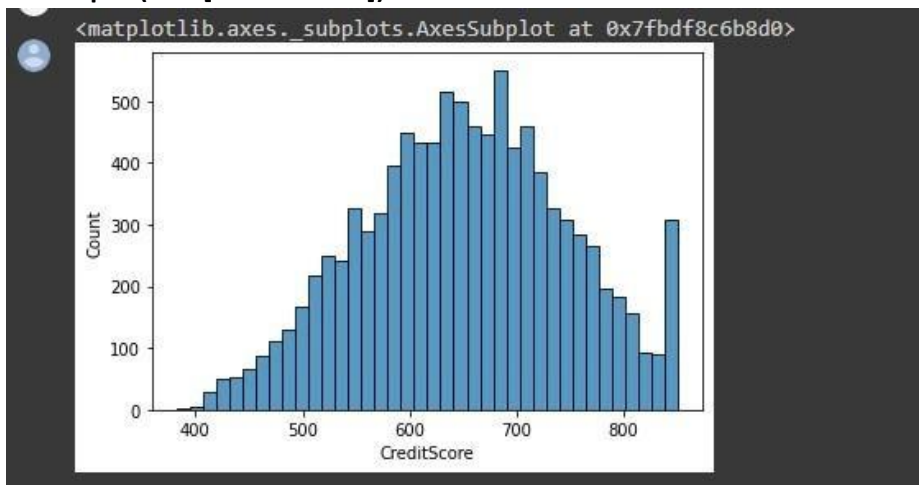
3.1 Univariate Analysis

Solution:

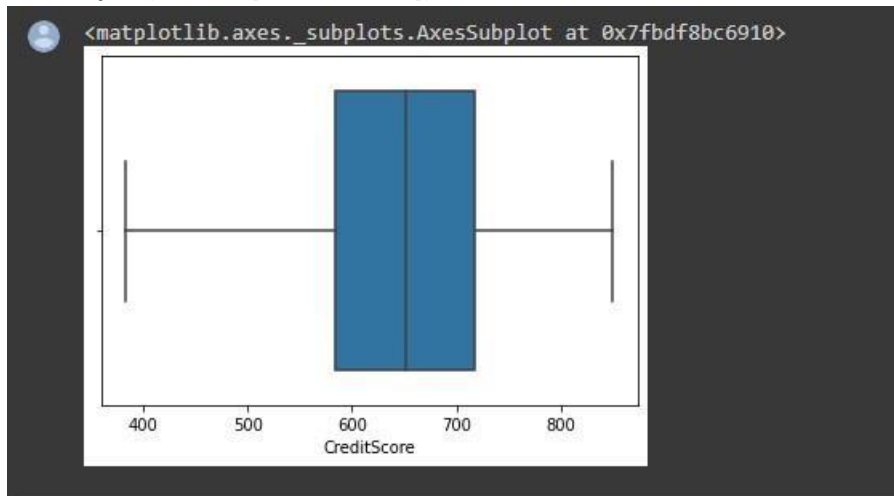
`sns.displot(data['CreditScore'])`



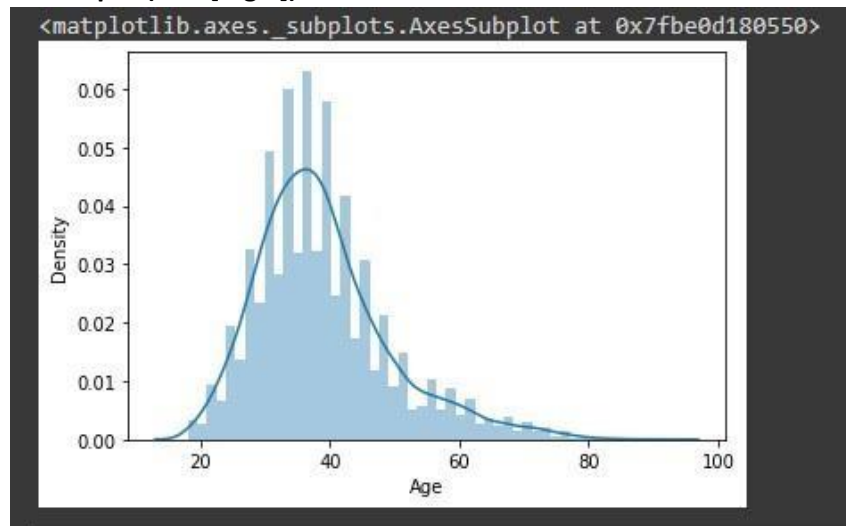
`sns.histplot(data['CreditScore'])`



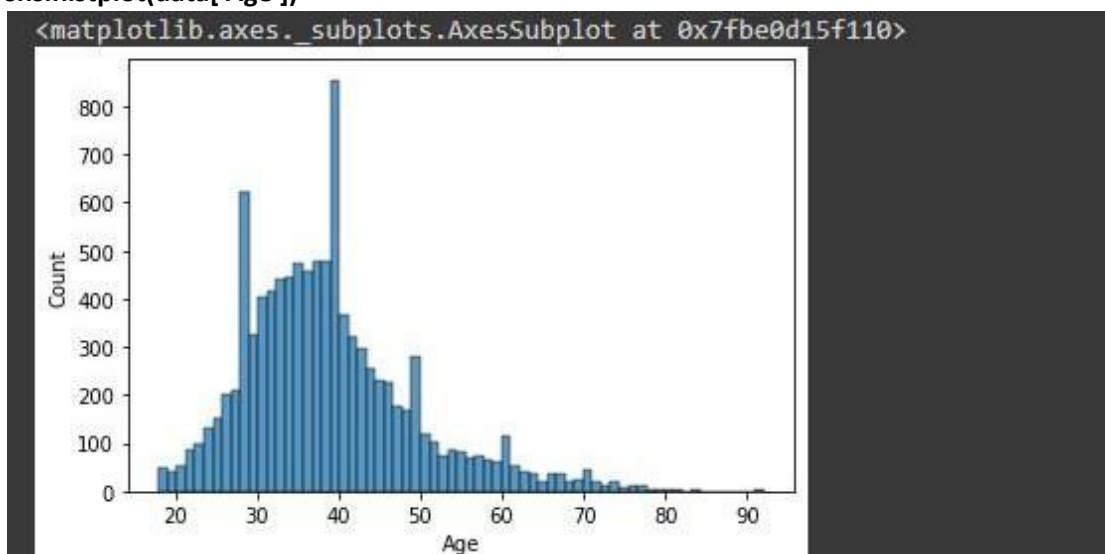
```
sns.boxplot(x = data['CreditScore'])
```



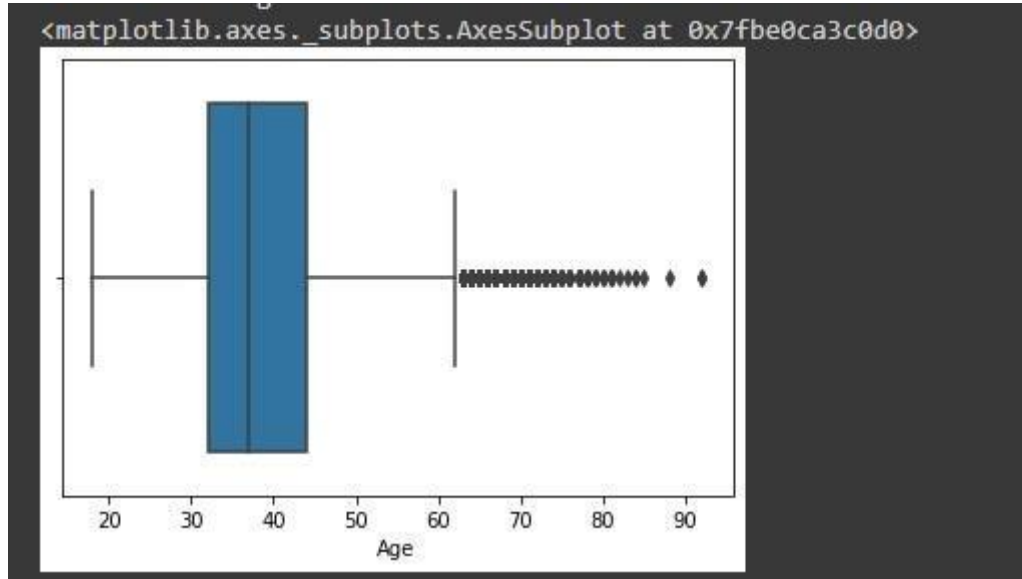
```
sns.distplot(data['Age'])
```



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



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

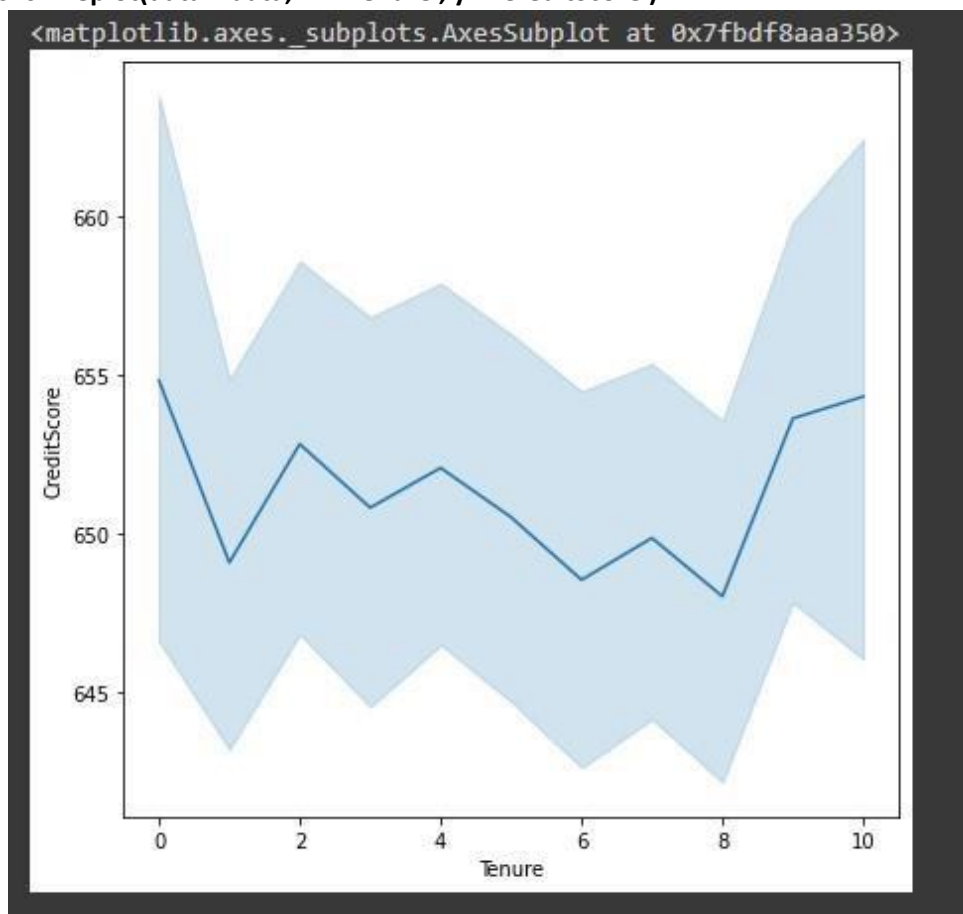


3.2 Bivariate Analysis

Solution:

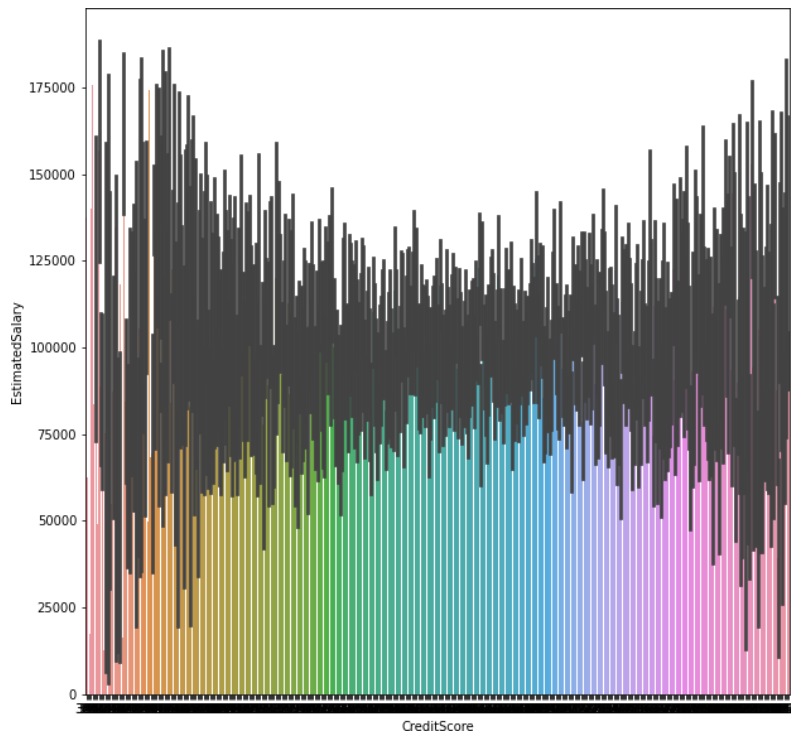
```
plt.figure(figsize=(7,7))
```

```
sns.lineplot(data = data, x = 'Tenure', y = 'CreditScore')
```



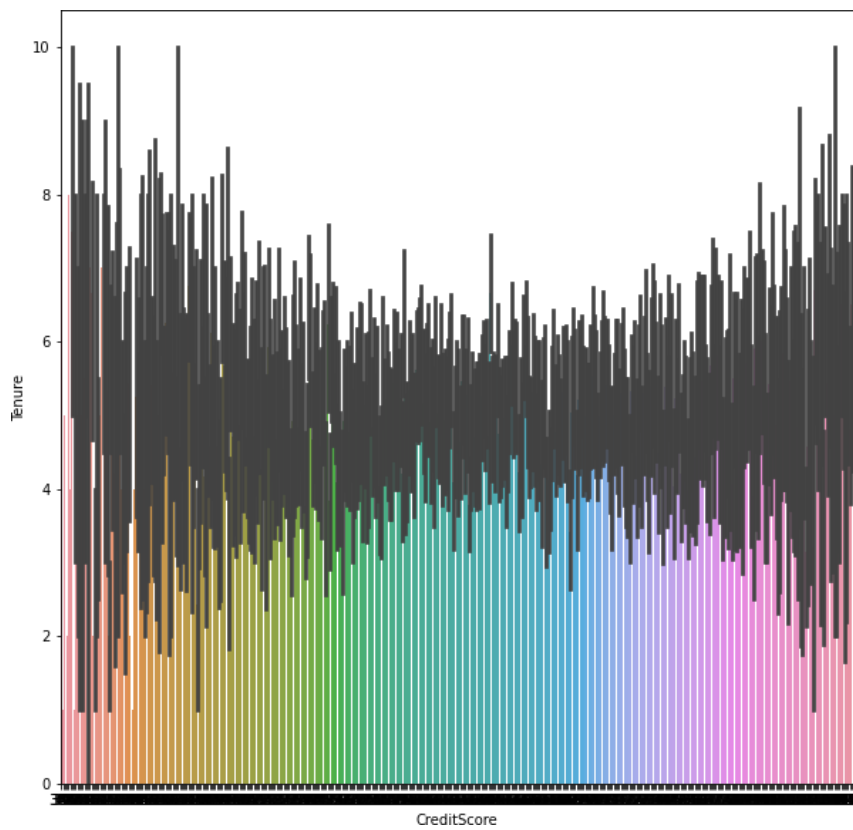
```
plt.figure(figsize=(10,10))
```

```
sns.barplot(data = data, x = 'CreditScore', y = 'EstimatedSalary')
```



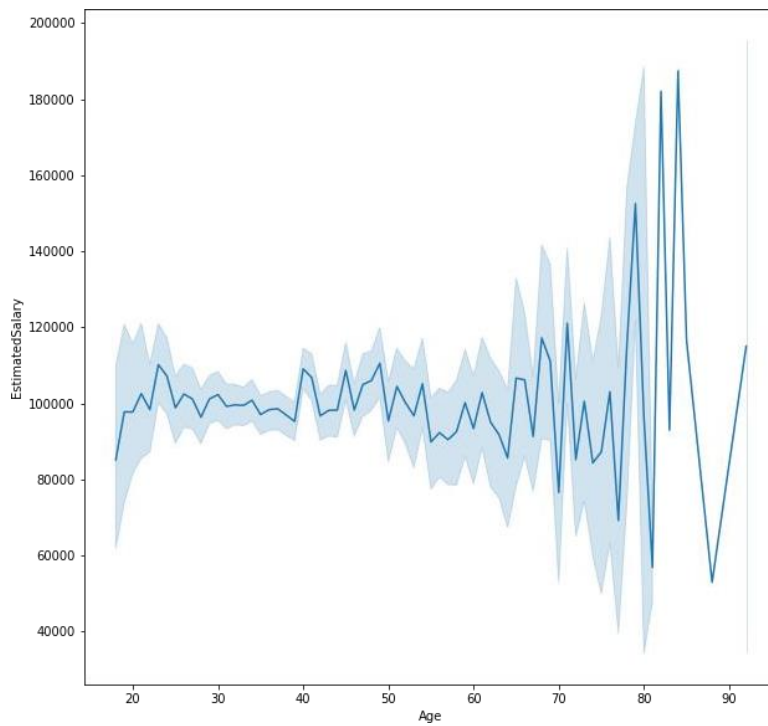
```
plt.figure(figsize=(10,10))
```

```
sns.barplot(data = data, x = 'CreditScore', y = 'Tenure')
```



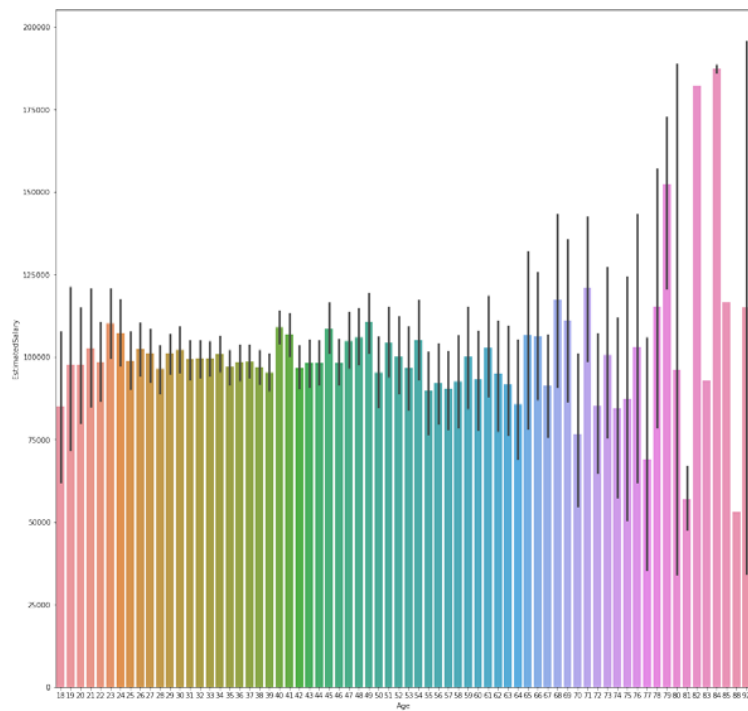
```
plt.figure(figsize=(10,10))
```

```
sns.lineplot(data['Age'], data['EstimatedSalary'])
```

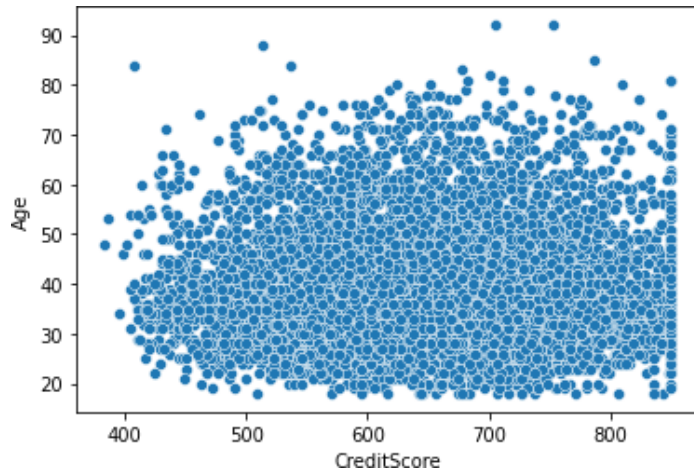


```
plt.figure(figsize=(17,17))
```

```
sns.barplot(data['Age'], data['EstimatedSalary'])
```

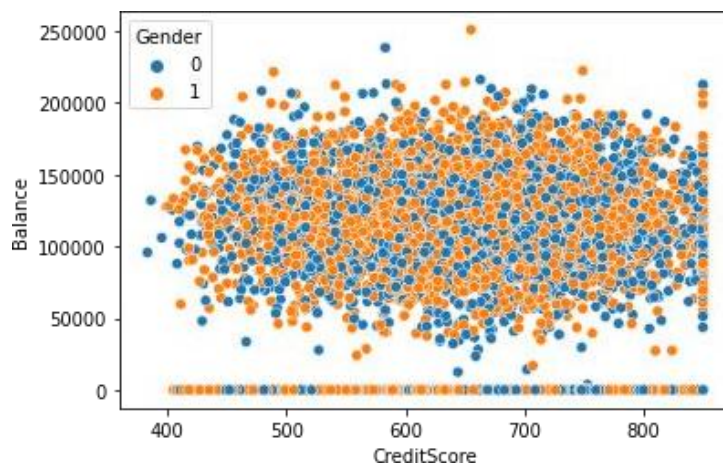



```
sns.scatterplot(data = data, x = 'CreditScore', y = 'Age')
```

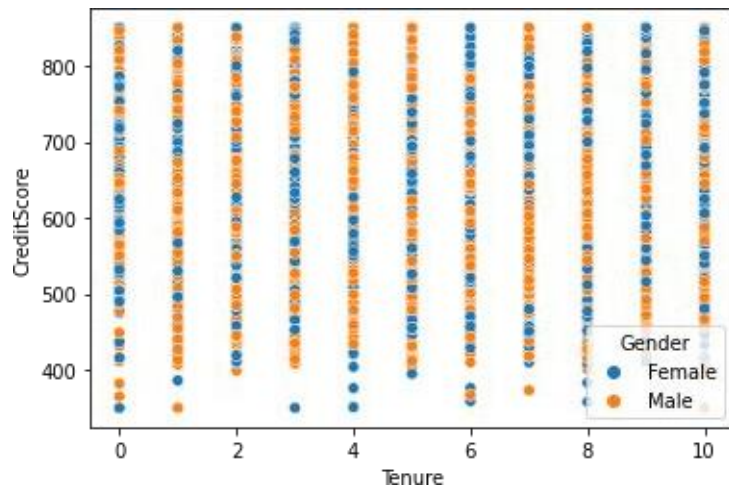


3.3 Multivariate Analysis

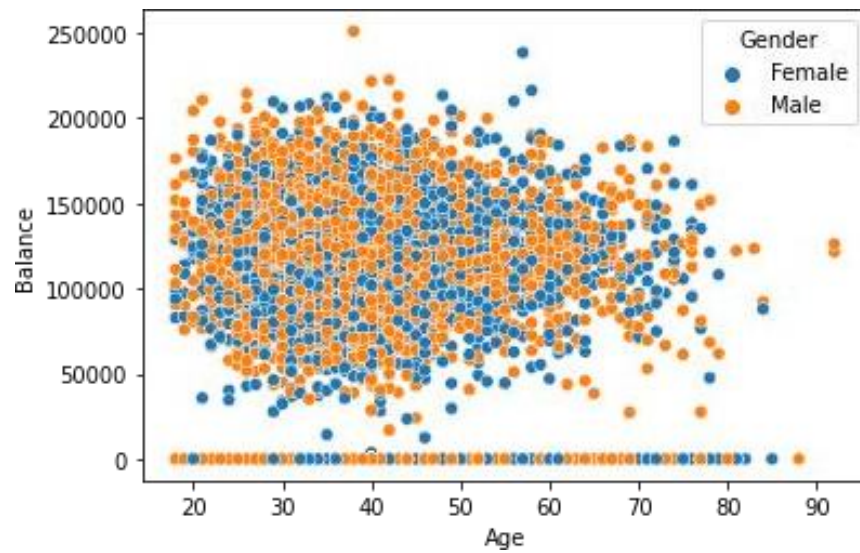
Solution: `sns.scatterplot(data = data, x = 'CreditScore', y = 'Balance', hue = 'Gender')`



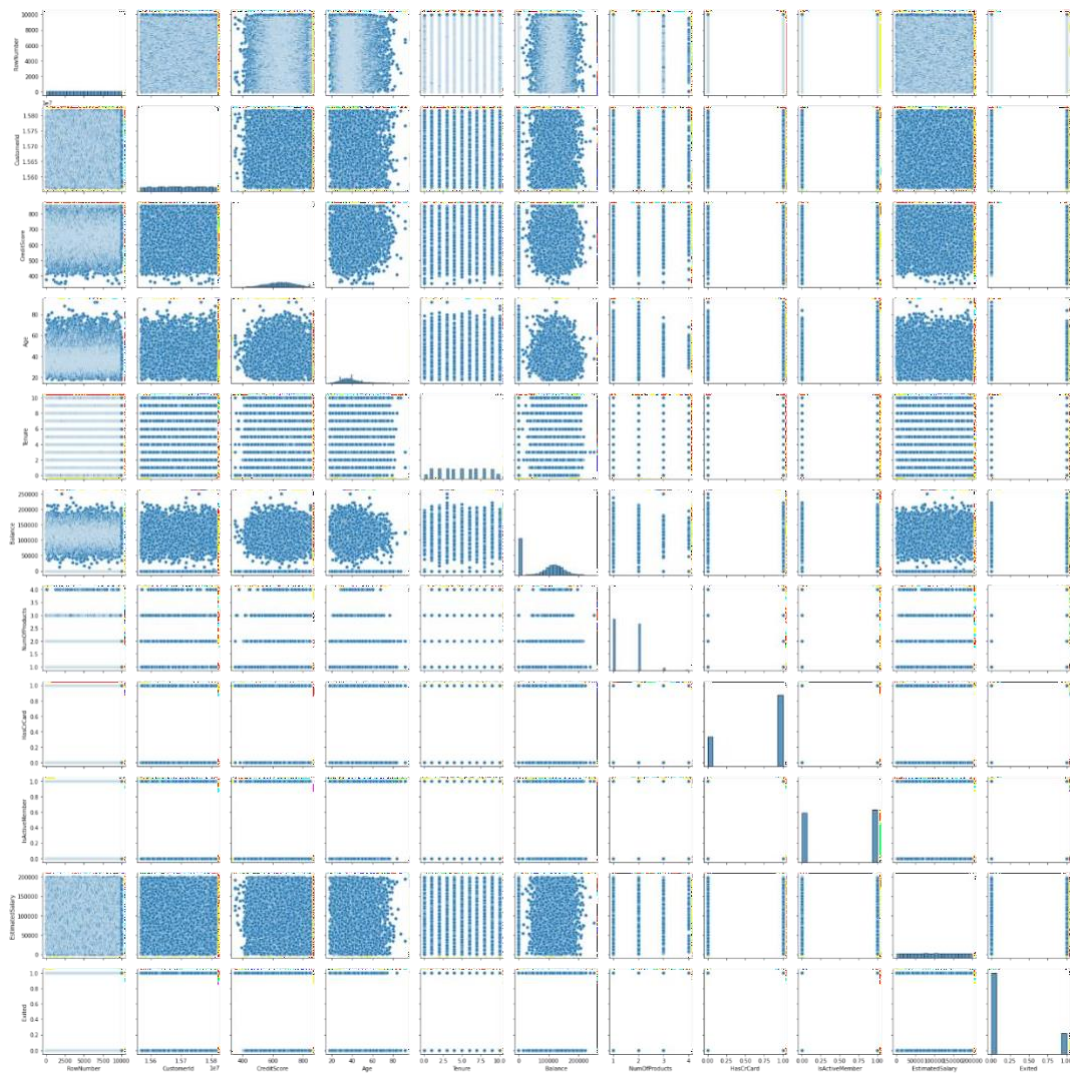
```
sns.scatterplot(data['Tenure'], data['CreditScore'], hue = data['Gender'])
```

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



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



Question-4. Perform descriptive statistics on the dataset.

Solution: `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	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

data['Tenure'].value_counts()

```
2    1048
1    1035
7    1028
8    1025
5    1012
3    1009
4     989
9     984
6     967
10    490
0     413
Name: Tenure, dtype: int64
```

Question-5. Handle the Missing values.

Solution: `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
```

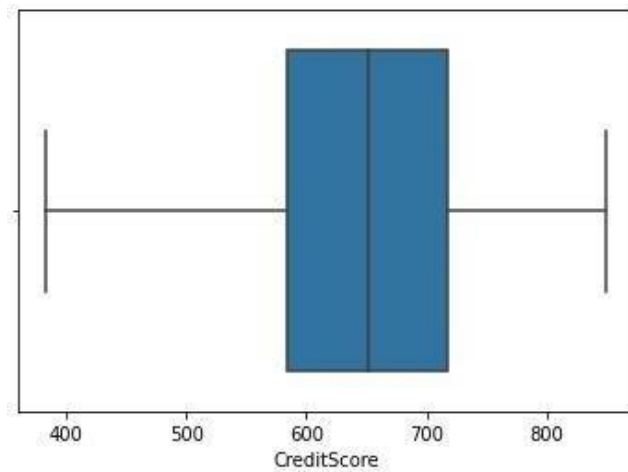
data.isnull().sum()

```
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

Solution: `sns.boxplot(data['CreditScore'])` **#Outlier detection**

- box plot

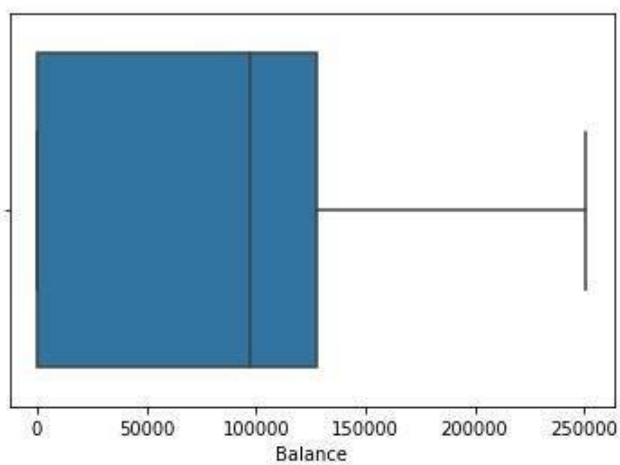
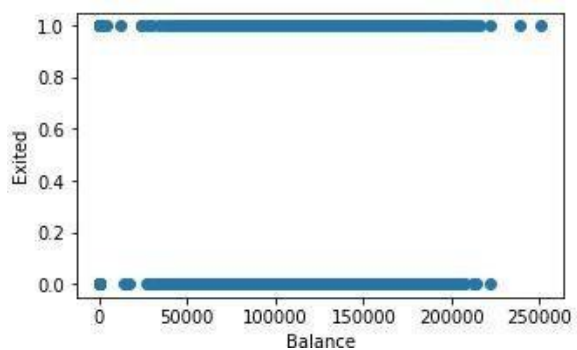


```
fig, ax = plt.subplots(figsize = (5,3)) #Outlier detection - Scatter plot
ax.scatter(data['Balance'], data['Exited'])
```

```
# x-axis label
ax.set_xlabel('Balance')
```

```
# y-axis label ax.set_ylabel('Exited')
plt.show()
```

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




```
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      0.332952
1      0.447540
2      1.551761
3      0.500422
4      2.073415
...
9995    1.250458
9996    1.405920
9997    0.604594
9998    1.260876
9999    1.469219
Name: CreditScore, Length: 10000, dtype: float64
No. of Outliers : (1, 0)
```

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

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.75	7500.25	15753233.75	2238.25	718.0	1.0	1.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0
0.25	2500.75	15628528.25	773.75	584.0	0.0	0.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0

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

```
RowNumber      4999.5000
CustomerId     124705.5000
Surname         1464.5000
CreditScore     134.0000
Geography       1.0000
Gender          1.0000
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
Surname         4.435000e+03
CreditScore     9.190000e+02
Geography      2.500000e+00
Gender         2.500000e+00
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
Surname        -1.423000e+03
CreditScore     3.830000e+02
Geography      -1.500000e+00
Gender         -1.500000e+00
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`

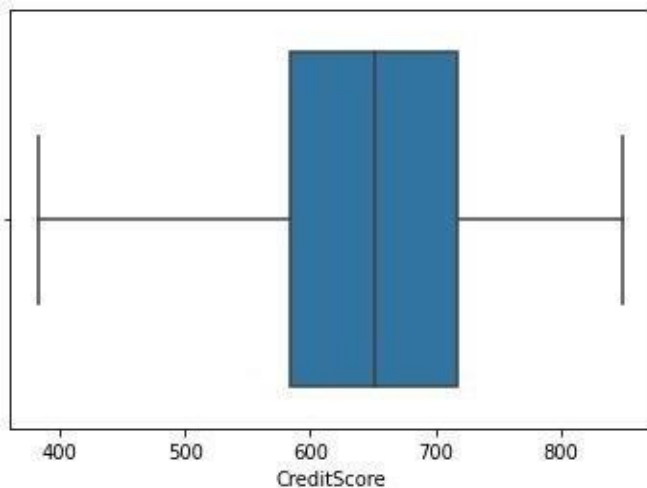
`= count + np.size(np.where(data['EstimatedSalary'] < lower))`

`print('No. of outliers : ', count)`

```
98386.1375
```

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

**`data['CreditScore'] = np.where(np.logical_or(data['CreditScore']>900, data['CreditScore']<383), 65
0, data['CreditScore']) sns.boxplot(data['CreditScore'])`**



```
upper = data.Age.mean() + (3 * data.Age.std()) #Outlier detection - 3 sigma lower
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 : 0
No. of outliers in Balance : 0
No. of outliers in NumOfProducts : 0
No. of outliers in Tenure : 0
No. of outliers in CreditScore : 0
```

Question-7. Check for Categorical columns and perform encoding

Solution:

```
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	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	1115	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	1177	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	2040	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	289	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	1822	850	2	0	43	2	125510.82	1	1	1	79084.10	0

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

Solution:

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

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	1115	619	0	0	42	2	0.00	1	1	1	101348.88
1	2	15647311	1177	608	2	0	41	1	83807.86	1	0	1	112542.58
2	3	15619304	2040	502	0	0	42	8	159660.80	3	1	0	113931.57
3	4	15701354	289	699	0	0	39	1	0.00	2	0	0	93826.63
4	5	15737888	1822	850	2	0	43	2	125510.82	1	1	1	79084.10
...
9995	9996	15606229	1999	771	0	1	39	5	0.00	2	1	0	96270.64
9996	9997	15569892	1336	516	0	1	35	10	57369.61	1	1	1	101699.77
9997	9998	15584532	1570	709	0	0	36	7	0.00	1	0	1	42085.58
9998	9999	15682355	2345	772	1	1	42	3	75075.31	2	1	0	92888.52
9999	10000	15628319	2751	792	0	0	28	4	130142.79	1	1	0	38190.78

10000 rows x 13 columns

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

```

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

Solution:

```

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

Solution:

```

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
```