ASSIGNMENT -2
Data Visualization and Pre-processing

Assignment Date	26 September 2022
Team ID	PNT2022TMID45335
Project Name	AI BASED DISCOURSE FOR BANKING INDUSTRY
Student Name	PRASANNA KUMAR M
Student Roll Number	E1195046
Maximum Marks	2 Marks

Question 1

1.Download the Data set

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'C:\Users\ADMIN\Downloads\Churn_Modelling.csv')
data.head()
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
data = pd.read_csv(r'C:\Users\ADMIN\Downloads\Churn_Modelling.csv')
data.head()
```

	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

Question-3

3.Perform Below Visualizations

Solution:

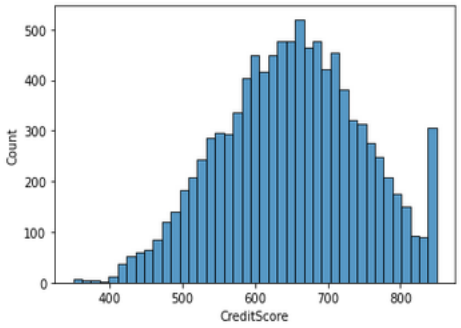
#1.Univariate Analysis

sns.histplot(data['CreditScore'])

```
#1.Univariate Analysis
```

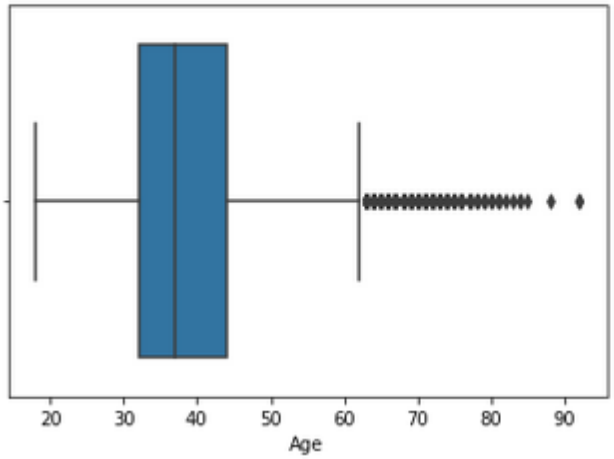
```
sns.histplot(data['CreditScore'])
```

```
<AxesSubplot:xlabel='CreditScore', ylabel='Count'>
```




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

```
<AxesSubplot:xlabel='Age'>
```



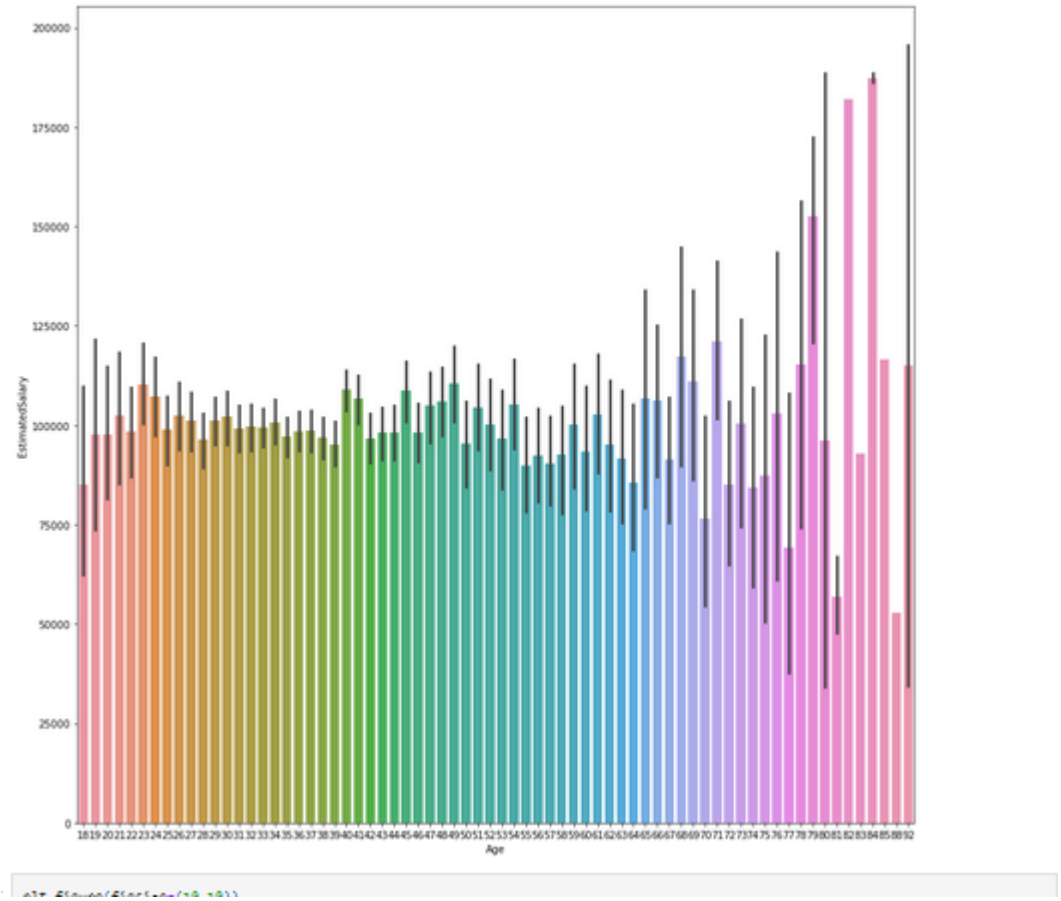
#2.Bivariate Analaysis

plt.figure(figsize=(15,15))
sns.barplot(x=data['Age'],y=data['EstimatedSalary'])

```
#2.Bivariate Analaysis
```

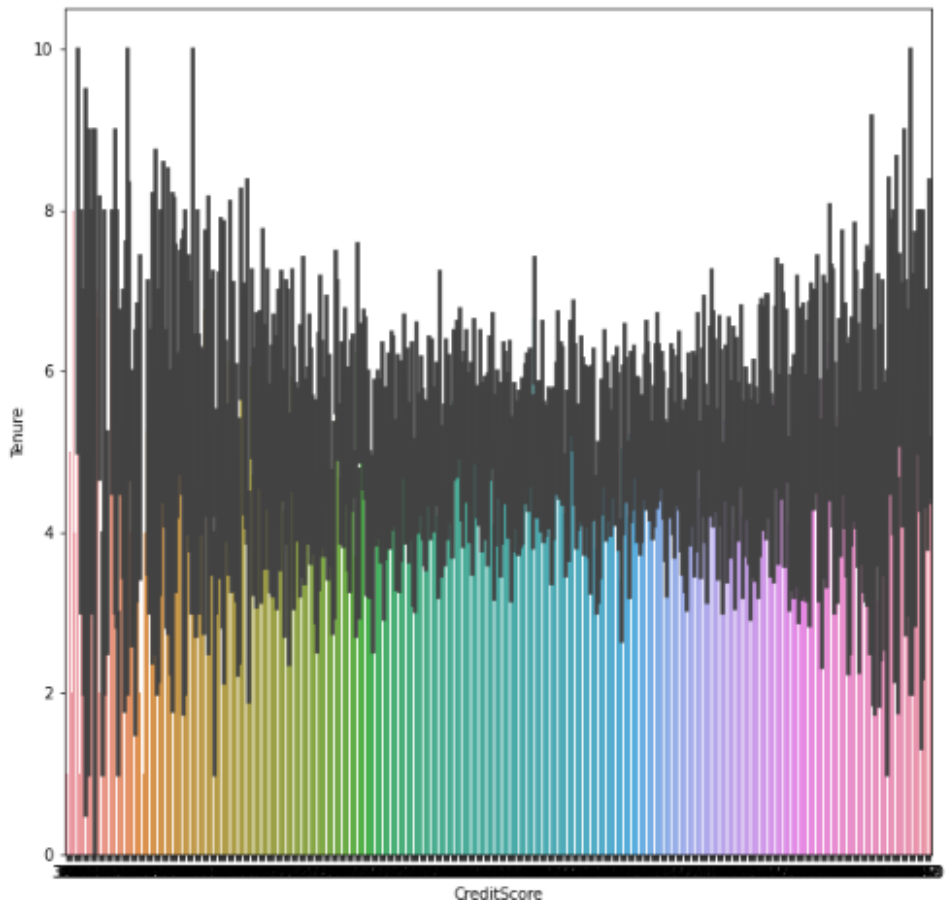
```
plt.figure(figsize=(15,15))
sns.barplot(x=data['Age'],y=data['EstimatedSalary'])
```

```
<AxesSubplot:xlabel='Age', ylabel='EstimatedSalary'>
```



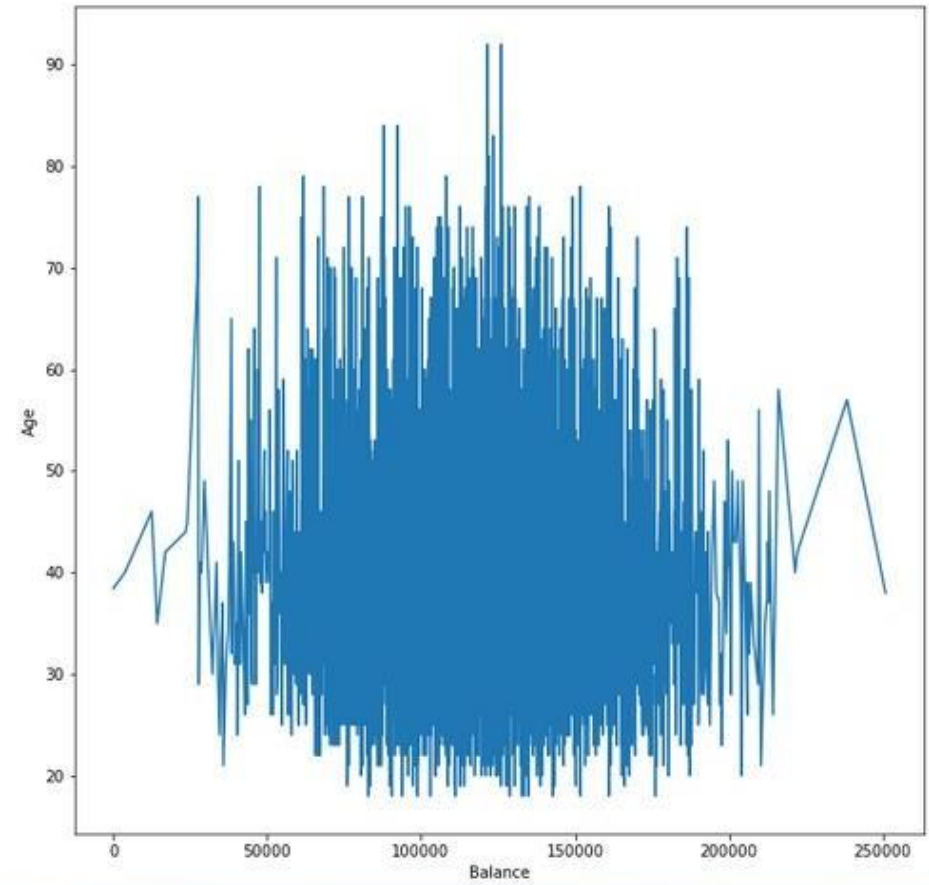
```
plt.figure(figsize=(10,10))
sns.barplot(x=data['CreditScore'],y=data['Tenure'])
```

```
<AxesSubplot:xlabel='CreditScore', ylabel='Tenure'>
```



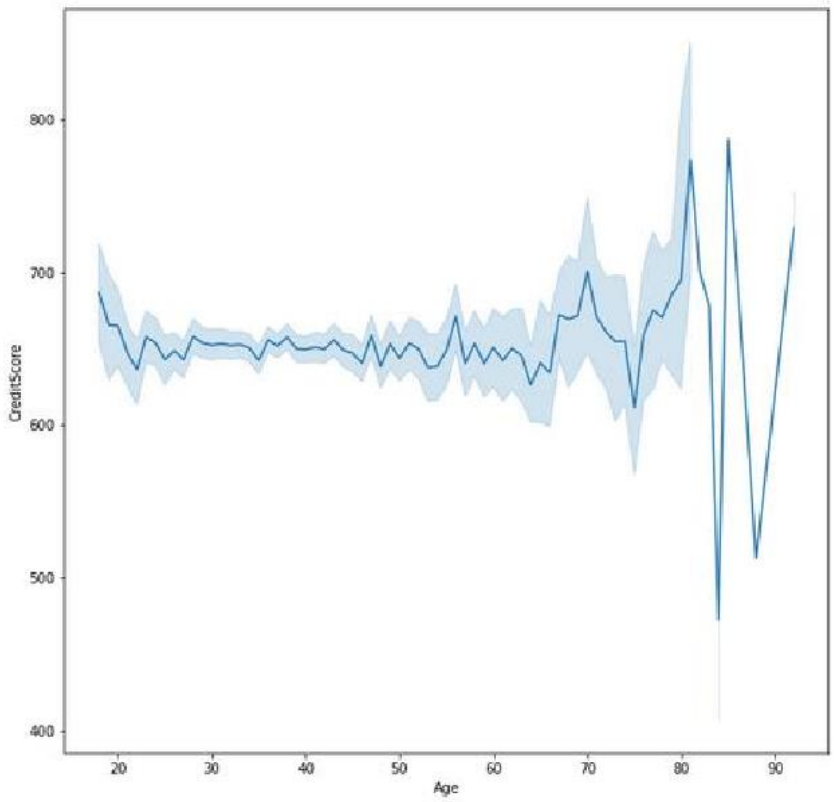
```
plt.figure(figsize=(10,10))
sns.lineplot(x=data['Balance'],y=data['Age'])
```

```
<AxesSubplot:xlabel='Balance', ylabel='Age'>
```



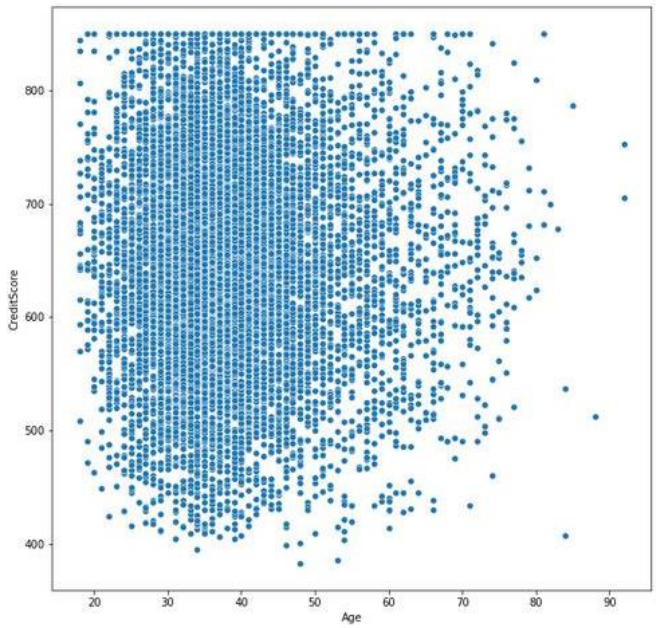
```
plt.figure(figsize=(10,10))
sns.lineplot(x=data['Age'],y=data['CreditScore'])
```

```
<AxesSubplot:xlabel='Age', ylabel='CreditScore'>
```




```
plt.figure(figsize=(10,10))
sns.scatterplot(x=data['Age'],y=data['CreditScore'])
```

```
<AxesSubplot:xlabel='Age', ylabel='CreditScore'>
```



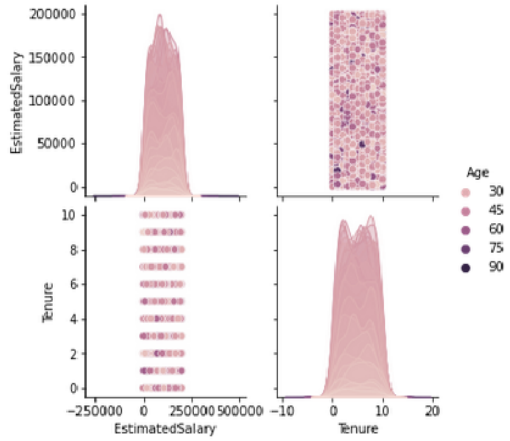
#3.Multi variate analysis

sns.pairplot(data= data [['Age','EstimatedSalary','Tenure']],hue='Age')

```
#3.Multi variate analysis
```

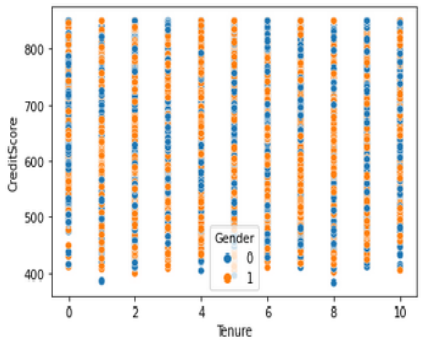
```
sns.pairplot(data= data [['Age','EstimatedSalary','Tenure']],hue='Age')
```

```
<seaborn.axisgrid.PairGrid at 0x2c7679deb50>
```




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

```
<AxesSubplot:xlabel='Tenure', ylabel='CreditScore'>
```

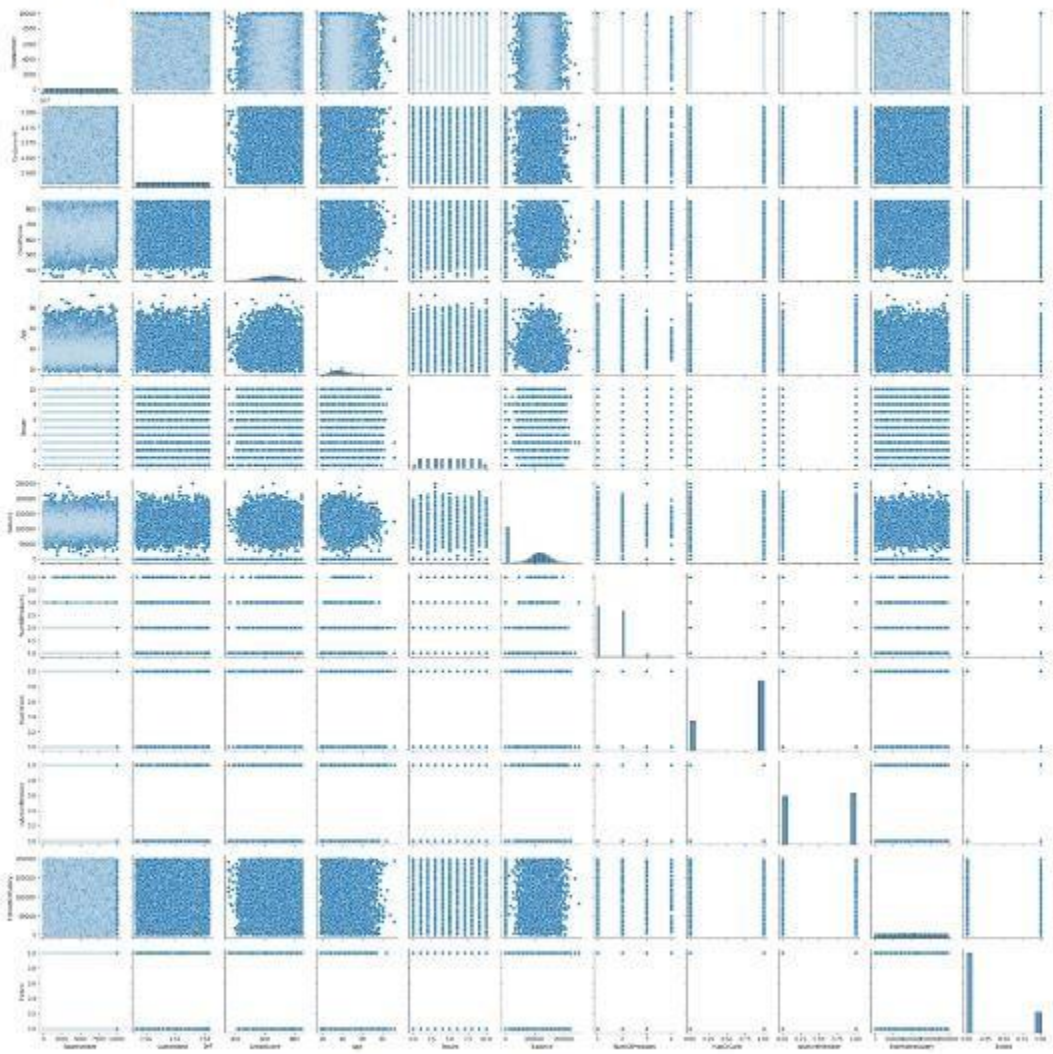


In [110...

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

Out[110...

```
<seaborn.axisgrid.PairGrid at 0x1c768c17f0>
```



Question.4:

Perform descriptive statistics on the dataset

Solution:

data.mean(numeric_only = True)

```
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)

```
In [112...  data.median(numeric_only = True)

Out[112...  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()
data['Age'].mode()
data['Balance'].unique()
data['Tenure'].unique()
data.std(numeric_only=True)

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

```
0    850
Name: CreditScore, dtype: int64
```

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

```
0    37
Name: Age, dtype: int64
```

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

```
array([     0.  ,  83807.86, 159660.8 , ...,  57369.61,  75075.31,
       130142.79])
```

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

```
array([ 2,  1,  8,  7,  4,  6,  3, 10,  5,  9,  0], dtype=int64)
```

data.describe()

```
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['NumOfProducts'].value_counts()

Question.5

Handle the Missing values

Solution:

data.isnull().any()
data.isnull().sum()

```
In [120...  data.isnull().any()

Out[120...  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 [121...  data.isnull().sum()

Out[121...  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
```

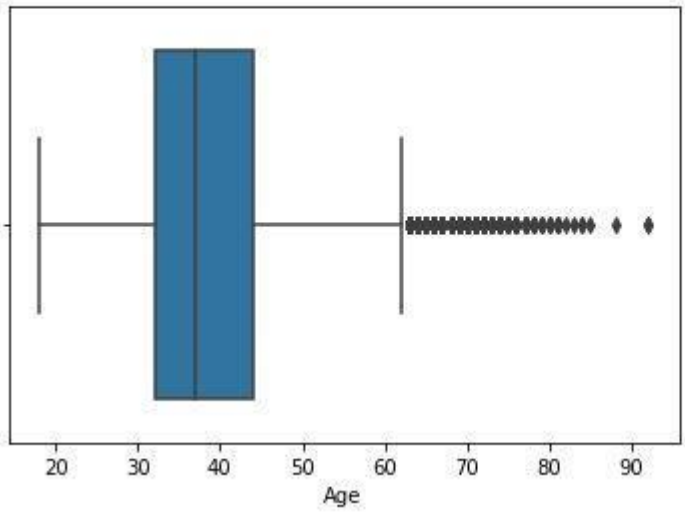
Question.6

Find the outliers and replace the outliers

Solution:

sns.boxplot(x=data['Age'])

```
]: sns.boxplot(x=data['Age'])
]: <AxesSubplot:xlabel='Age'>
```



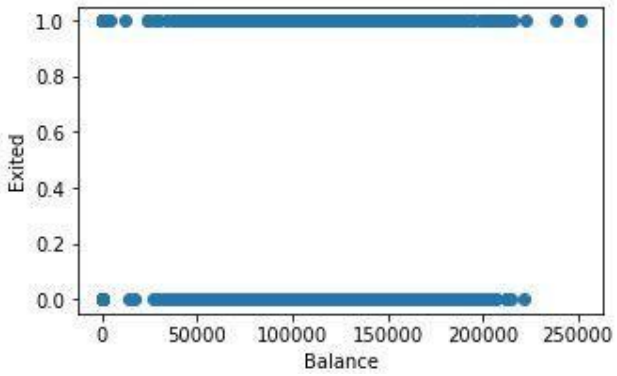
fig, ax = plt.subplots(figsize = (5,3)) #Outlier detection - Scatter plot
ax.scatter(data['Balance'], data['Exited'])

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

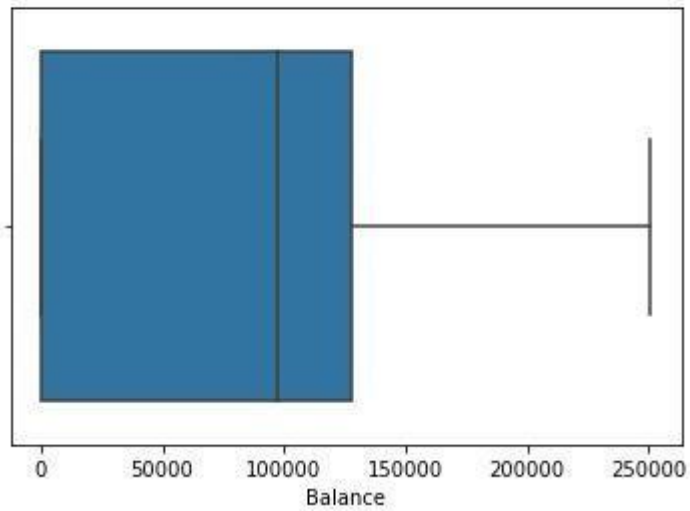
# y-axis label
ax.set_ylabel('Exited')
plt.show()

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

```
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'])
```




```
<AxesSubplot:xlabel='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)))
```

```
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([0.70,0.30])
q

```
q = data.quantile([0.70,0.30])
q
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.7	7000.3	15740461.6	704.0	42.0	7.0	122029.87	2.0	1.0	1.0	139432.236	0.0
0.3	3000.7	15641363.9	598.7	33.0	3.0	0.00	1.0	1.0	0.0	60736.079	0.0

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

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

```
RowNumber            3999.600
CustomerId          99097.700
CreditScore           105.300
Age                     9.000
Tenure                  4.000
Balance            122029.870
NumOfProducts           1.000
HasCrCard               0.000
IsActiveMember          1.000
EstimatedSalary     78696.157
Exited                  0.000
dtype: float64
```

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

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

```
RowNumber          1.299970e+04
CustomerId         1.588911e+07
CreditScore        8.619500e+02
Age                5.550000e+01
Tenure             1.300000e+01
Balance            3.050747e+05
NumOfProducts      3.500000e+00
HasCrCard          1.000000e+00
IsActiveMember     2.500000e+00
EstimatedSalary    2.574765e+05
Exited             0.000000e+00
dtype: float64
```

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

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

```
RowNumber         -2.998700e+03
CustomerId         1.549272e+07
CreditScore        4.407500e+02
Age                1.950000e+01
Tenure            -3.000000e+00
Balance           -1.830448e+05
NumOfProducts     -5.000000e-01
HasCrCard          1.000000e+00
IsActiveMember    -1.500000e+00
EstimatedSalary   -5.730816e+04
Exited             0.000000e+00
dtype: float64
```

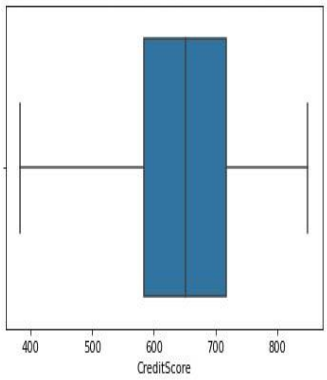
```
Q1 = data['EstimatedSalary'].quantile(0.30) #Outlier detection - IQR
Q3 = data['EstimatedSalary'].quantile(0.70)
iqr = Q3 - Q1
print(iqr)
upper=Q3 + 1.5 * iqr
lower=Q1 - 1.5 * iqr
count = np.size(np.where(data['EstimatedSalary'] >upper))
count = count + np.size(np.where(data['EstimatedSalary'] <lower))
print('No. of outliers : ', count)
```

```
Q1 = data['EstimatedSalary'].quantile(0.30) #Outlier detection - IQR
Q3 = data['EstimatedSalary'].quantile(0.70)
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)
```

```
78696.157
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 vali
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f47eb744d90>
```




```
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', 'Balance',  'Tenure'] #After outlier removal

for i in columns:
  Q1 = data[i].quantile(0.30)
  Q3 = data[i].quantile(0.70)
  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)

```
columns = ['EstimatedSalary', 'Balance',  'Tenure'] #After outlier removal

for i in columns:
  Q1 = data[i].quantile(0.30)
  Q3 = data[i].quantile(0.70)
  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  Balance  :  0
No. of outliers in  Tenure  :  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()

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

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

	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 × 13 columns

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

```
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

Solution:

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

```
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:

import pandas as pd
data=pd.read_csv("/content/Churn_Modelling.csv")

```
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

import numpy as np
x=np.array(data["CustomerId"]).reshape(-1,1)
x.shape

```
import numpy as np
```

```
x=np.array(data["CustomerId"]).reshape(-1,1)
x.shape
```

```
(10000, 1)
```

```
y=np.array(data["EstimatedSalary"])
y.shape
```

```
(10000,)
```

```
[ ] print(y)
```

```
[101348.88 112542.58 113931.57 ...  42085.58  92888.52  38190.78]
```

```
[ ] print(type(x))
```

```
<class 'numpy.ndarray'>
```

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

```
[ ] x_train.shape
```

```
(7000, 1)
```

```
x_test
```

```
array([[15611365],
       [15610379],
       [15641690],
       ...,
       [15724876],
       [15765952],
       [15661330]])
```

```
x_test.shape
```

```
(3000, 1)
```

```
y_test
```

```
array([ 60905.51, 121124.53, 163714.92, ...,  33245.97, 188382.77,
       116141.72])
```

```
y_test.shape
```

```
(3000,)
```

```
y.shape
```

```
(10000,)
```

```
print(y_train.shape)
```

```
(7000,)
```

```
print(y_test.shape)
```

```
(3000,)
```