ASSIGNMENT-02

DATA VISUALIZATION AND PRE PROCESSING

Assignment Date	22 September 2022
Student Name	Dharshan p
Student Roll Number	113219071006
Maximum Marks	2 Marks

1. Download the dataset: Dataset downloaded in csv form.

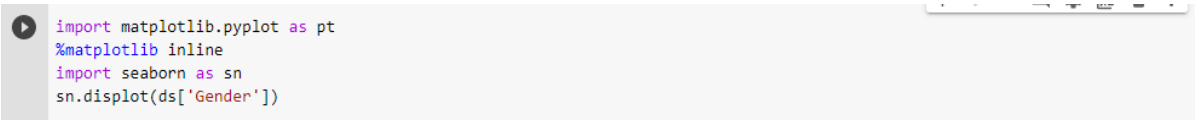
2. Load the dataset.

```
import pandas as pd
ds = pd.read_csv("/content/drive/MyDrive/IBM/Churn_Modelling.csv")
```

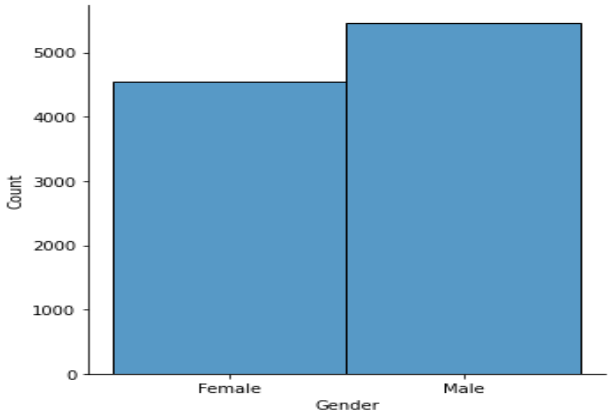
3. Perform Below Visualizations.

   - Univariate Analysis

```
sn.displot(ds['Gender'])
```



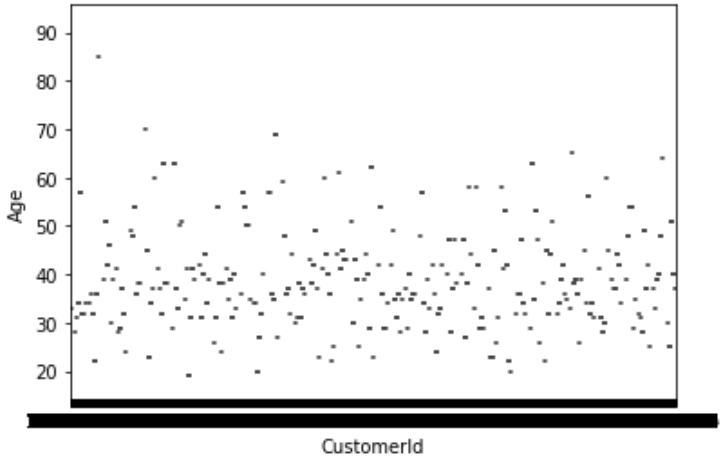

```
<seaborn.axisgrid.FacetGrid at 0x7f09cae8c310>
```



- Bi - Variate Analysis `sn.boxplot(ds['CustomerId'],ds['Age'])`

```
sn.boxplot(ds['CustomerId'],ds['Age'])
```

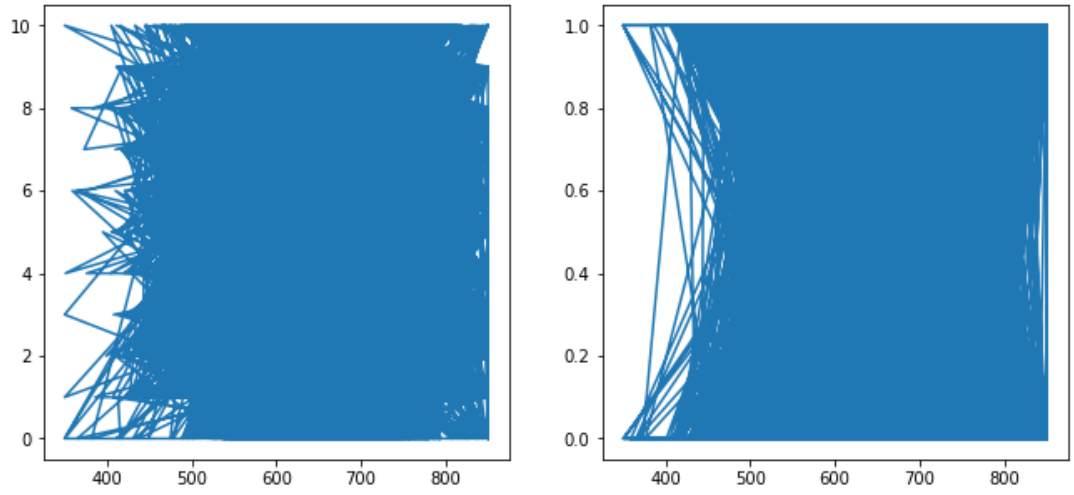
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ef40f68d0>
```



- Multi - Variate Analysis

```
pt.figure(figsize=(11,5))
pt.subplot(1,2,1)
pt.plot(ds['CreditScore'],ds['Tenure'])
pt.subplot(1,2,2)
pt.plot(ds['CreditScore'],ds['IsActiveMember'])
```

```
[<matplotlib.lines.Line2D at 0x7f5eb7c46e50>]
```



4. Perform descriptive statistics on the dataset.

```
ds.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

Mean:

```
ds.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_lau
  """Entry point for launching an IPython kernel.
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
```

5. Handle the Missing values.

```
ds.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
```

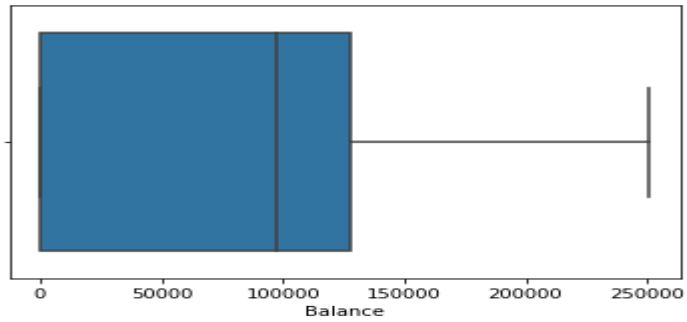
6. Find the outliers and replace the outliers

Finding Outliers:

Using Boxplot

```
sn.boxplot(ds['Balance'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f5eb79661d0>
```



Using method

```
qnt=ds.quantile(q=(0.50,1))
qnt
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.5	5000.5	15690738.0	652.0	37.0	5.0	97198.54	1.0	1.0	1.0	100193.915	0.0
1.0	10000.0	15815690.0	850.0	92.0	10.0	250898.09	4.0	1.0	1.0	199992.480	1.0


```
it=qnt.loc[1]-qnt.loc[0.5]
it
```

```
RowNumber           4999.500
CustomerId        124952.000
CreditScore          198.000
Age                   55.000
Tenure                 5.000
Balance           153699.550
NumOfProducts          3.000
HasCrCard              0.000
IsActiveMember         0.000
EstimatedSalary    99798.565
Exited                 1.000
dtype: float64
```

```
l=qnt.loc[0.50]-2*it
print("lowerbound:",l)
u=qnt.loc[1]+2*it
print("upperbound:",u)
```

```
lowerbound: RowNumber         -4.998500e+03
CustomerId          1.544083e+07
CreditScore         2.560000e+02
Age                -7.300000e+01
Tenure             -5.000000e+00
Balance            -2.102006e+05
NumOfProducts      -5.000000e+00
HasCrCard           1.000000e+00
IsActiveMember      1.000000e+00
EstimatedSalary    -9.940322e+04
Exited             -2.000000e+00
dtype: float64
upperbound: RowNumber             19999.00
CustomerId          16065594.00
CreditScore            1246.00
Age                     202.00
Tenure                   20.00
Balance              558297.19
NumOfProducts           10.00
HasCrCard                1.00
IsActiveMember           1.00
EstimatedSalary     399589.61
Exited                   3.00
dtype: float64
```

Replacing Outliers:

```
'''replacing outliers'''
import numpy as np
ds['Balance']=np.where(ds['Balance']>102016,0.01,ds['Balance'])
```

7. Check for Categorical columns and perform encoding.

Categorical columns: CreditScore,Geography

```
[17] from sklearn.preprocessing import LabelEncoder
     labelencoder_ds=LabelEncoder()
     ds['CreditScore']=labelencoder_ds.fit_transform(ds['CreditScore'])
```

```
ds.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	228	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	217	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	111	France	Female	42	8	0.00	3	1	0	113931.57	1
3	4	15701354	Boni	308	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	459	Spain	Female	43	2	0.01	1	1	1	79084.10	0

```
ds['Geography']=labelencoder_ds.fit_transform(ds['Geography'])
ds.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	228	0	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	217	2	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	111	0	Female	42	8	0.00	3	1	0	113931.57	1
3	4	15701354	Boni	308	0	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	459	2	Female	43	2	0.01	1	1	1	79084.10	0

8. Split the data into dependent and independent variables.

```
a=ds.iloc[:,:-1].values
print(a)
```

```
[[1 15634602 'Hargrave' ... 1 1 101348.88]
 [2 15647311 'Hill' ... 0 1 112542.58]
 [3 15619304 'Onio' ... 1 0 113931.57]
 ...
 [9998 15584532 'Liu' ... 0 1 42085.58]
 [9999 15682355 'Sabbatini' ... 1 0 92888.52]
 [10000 15628319 'Walker' ... 1 0 38190.78]]
```

```
b=ds.iloc[:,-1].values
print(b)
```

```
[1 0 1 ... 1 1 0]
```

9. Scale the independent variables

```
from sklearn.preprocessing import scale
b=scale(b)
b
```

```
array([-0.78321342, -0.60653412, -0.99588476, ..., -1.47928179,
       -0.11935577, -0.87055909])
```

10. Split the data into training and testing

b_train

```
array([-0.50577476, -0.50577476, -0.50577476, ..., -0.50577476,
       -0.50577476,  1.97716468])
```

b_test

```
array([-0.50577476,  1.97716468, -0.50577476, ..., -0.50577476,
       -0.50577476, -0.50577476])
```

a_train

```
array([[7390, 15676909, 'Mishin', ..., 1, 0, 163830.64],
       [9276, 15749265, 'Carslaw', ..., 1, 1, 57098.0],
       [2996, 15582492, 'Moore', ..., 1, 0, 185630.76],
       ...,
       [3265, 15574372, 'Hoolan', ..., 1, 0, 181429.87],
       [9846, 15664035, 'Parsons', ..., 1, 1, 148750.16],
       [2733, 15592816, 'Udokamma', ..., 1, 0, 118855.26]], dtype=object)
```

a_test

```
array([[9395, 15615753, 'Upchurch', ..., 1, 1, 192852.67],
       [899, 15654700, 'Fallaci', ..., 1, 0, 128702.1],
       [2399, 15633877, 'Morrison', ..., 1, 1, 75732.25],
       ...,
       [9550, 15772604, 'Chiemezie', ..., 1, 0, 141533.19],
       [2741, 15787699, 'Burke', ..., 1, 1, 11276.48],
       [6691, 15579223, 'Niu', ..., 1, 0, 192950.6]], dtype=object)
```