ASSIGNMENT 2

ASSIGNMENT DATE	24 SEPTEMBER 2020
STUDENT NAME	THANISH MALAI P
STUDENT ROLL NUMBER	2019504598
MAXIMUM MARKS	2 MARKS

Data Visualization and Pre-processing

Perform Below Tasks to complete the assignment:- Tasks:-

1. Download the dataset
2. Load the dataset.
3. Perform Below Visualizations. ● Univariate Analysis ● Bi - Variate Analysis ● Multi - Variate Analysis
4. Perform descriptive statistics on the dataset.
5. Handle the Missing values.
6. Find the outliers and replace the outliers
7. Check for Categorical columns and perform encoding.
8. Split the data into dependent and independent variables.
9. Scale the independent variables
10. Split the data into training and testing

In [ ]:

```
import numpy as np
import pandas as pd
```

In [ ]:

```
df = pd.read_csv("Churn_Modelling.csv")
```

In [ ]:

```
df
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838
2	3	15619304	Onio	502	France	Female	42	8	1596
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	1255
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	573
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	750
9999	10000	15628319	Walker	792	France	Female	28	4	1301

10000 rows × 14 columns

3.Visualizations

In [ ]:

```
import matplotlib.pyplot as plt
```

In [ ]:

```
import seaborn as sns
```

In [ ]:

```
%matplotlib inline
```

i)Univariate Analysis

In [ ]:

```
df[['CustomerId','Surname','CreditScore','Geography','Age','Tenure']].describe()
```

Out[ ]:

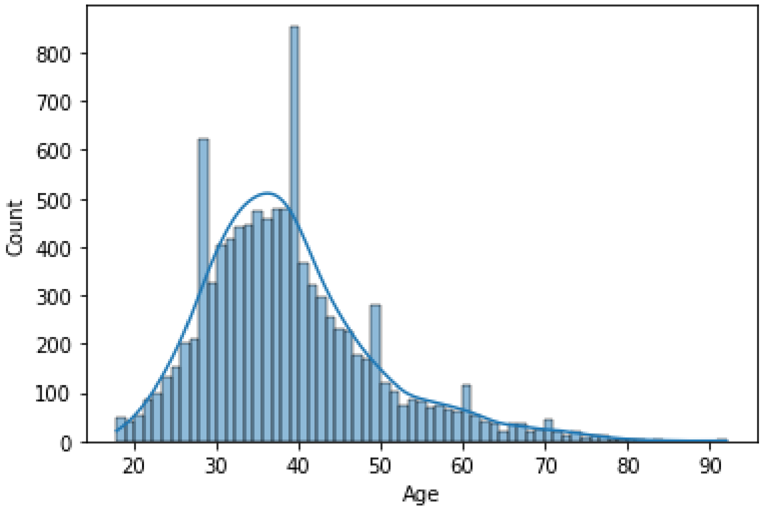
	CustomerId	CreditScore	Age	Tenure
count	1.000000e+04	10000.000000	10000.000000	10000.000000
mean	1.569094e+07	650.528800	38.921800	5.012800
std	7.193619e+04	96.653299	10.487806	2.892174
min	1.556570e+07	350.000000	18.000000	0.000000
25%	1.562853e+07	584.000000	32.000000	3.000000
50%	1.569074e+07	652.000000	37.000000	5.000000
75%	1.575323e+07	718.000000	44.000000	7.000000
max	1.581569e+07	850.000000	92.000000	10.000000

In [ ]:

```
sns.histplot(df.Age,kde=True)
```

Out[ ]:

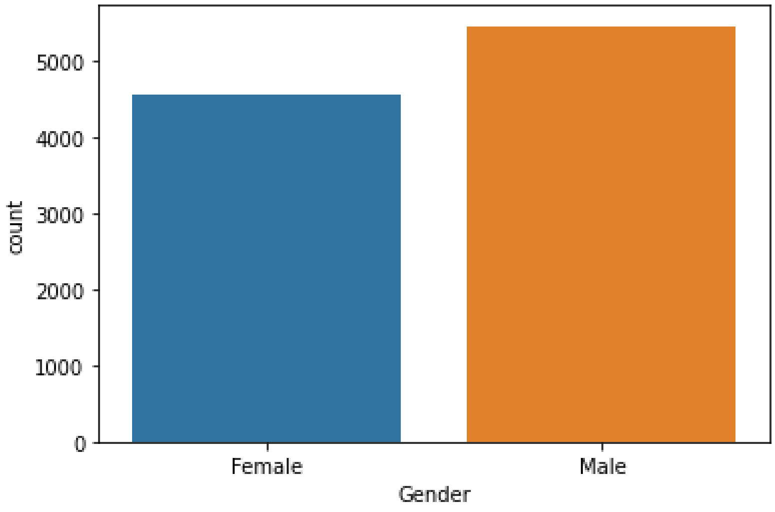
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffa2c5af410>
```



```
In [ ]: # plot count plot for the gender column
        sns.countplot(df.Gender)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
ass 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 explici
t keyword will result in an error or misinterpretation.
  FutureWarning
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa2c06c650>
```



ii)Bivariate Analysis

```
In [ ]: df[['CustomerId','Surname','CreditScore','Geography','Gender','Age']].corr()
```

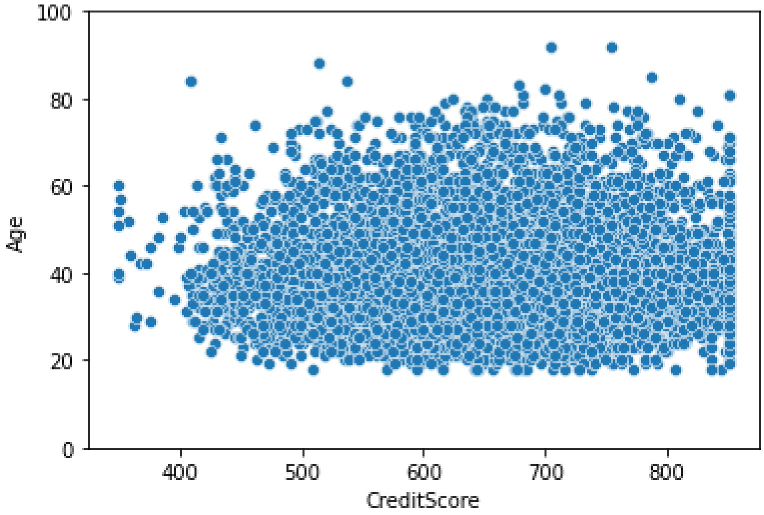
Out[ ]:

	CustomerId	CreditScore	Age
CustomerId	1.000000	0.005308	0.009497
CreditScore	0.005308	1.000000	-0.003965
Age	0.009497	-0.003965	1.000000

```
In [ ]: sns.scatterplot(df.CreditScore,df.Age)
        plt.ylim(0,100)
```

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

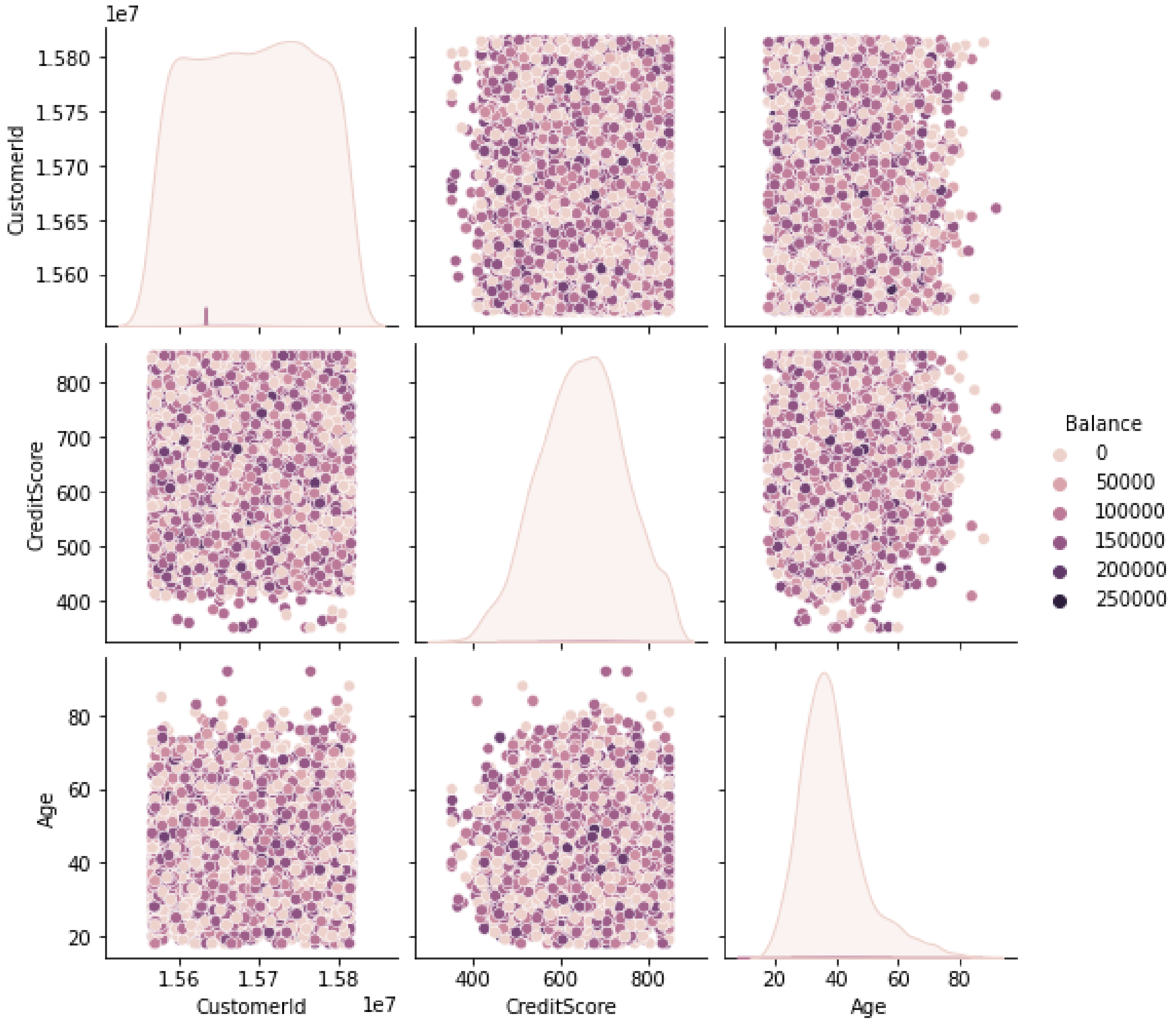
```
Out[ ]: (0.0, 100.0)
```



iii)Multivariate Analysis

```
In [ ]: sns.pairplot(data =df[['CustomerId','Geography','Gender','CreditScore','Age','Balar
```

```
Out[ ]: <seaborn.axisgrid.PairGrid at 0x7ffa2bbc9250>
```



4.Descriptive Statistics

In [ ]:

```
#mode
df['Age'].mode()
```

Out[ ]:

```
0    37
dtype: int64
```

In [ ]:

```
#calculation of the mean (for Age)
df["Age"].mean()
```

Out[ ]:

```
38.9218
```

In [ ]:

```
#calculation of the mean and round the result(for Age)
round(df["Age"].mean(), 2)
```

Out[ ]:

```
38.92
```

In [ ]:

```
#calculation of the median(for Age)
df["Age"].median()
```

Out[ ]:

```
37.0
```

In [ ]:

```
df.columns
```

Out[ ]:

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

In [ ]:

```
df["NumOfProducts"].value_counts()
```

Out[ ]:

```
1    5084
2    4590
3     266
4      60
Name: NumOfProducts, dtype: int64
```

In [ ]:

```
df.dtypes
```

Out[ ]:

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [ ]:

```
df.head()
```

Out[ ]:

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

In [ ]:

```
df.describe()
```

Out[ ]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	Nur
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	1
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	

5.Handling Missing values

In [ ]:

```
df.isna().any()
```

Out[ ]:

```
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 [ ]:

```
df.isnull().sum()
```

Out[ ]:

```
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 [ ]:

```
df.isnull()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	Fal
9996	False	False	False	False	False	False	False	False	Fal
9997	False	False	False	False	False	False	False	False	Fal
9998	False	False	False	False	False	False	False	False	Fal
9999	False	False	False	False	False	False	False	False	Fal

10000 rows × 14 columns

In [ ]:

```
df.notnull()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	True	True	True	True	True	True	True	True	Tru
1	True	True	True	True	True	True	True	True	Tru
2	True	True	True	True	True	True	True	True	Tru
3	True	True	True	True	True	True	True	True	Tru
4	True	True	True	True	True	True	True	True	Tru
...	...	...	...	...	...	...	...	...	
9995	True	True	True	True	True	True	True	True	Tru
9996	True	True	True	True	True	True	True	True	Tru
9997	True	True	True	True	True	True	True	True	Tru
9998	True	True	True	True	True	True	True	True	Tru
9999	True	True	True	True	True	True	True	True	Tru

10000 rows × 14 columns

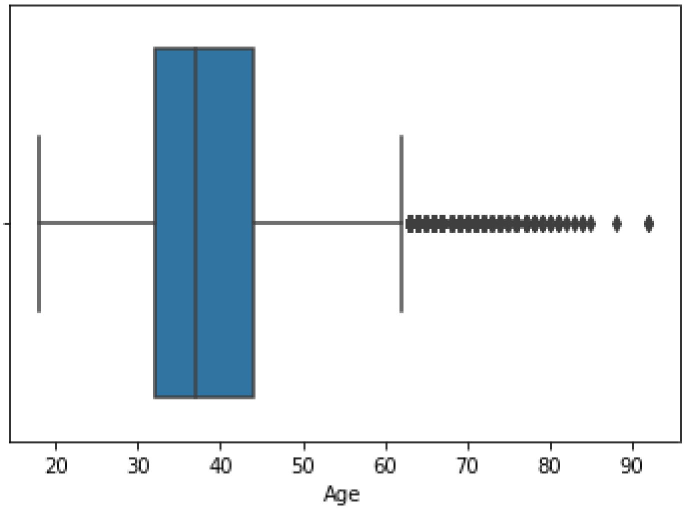
6.Finding and replacing the outliers

In [ ]:

```python
import seaborn as sns
sns.boxplot(x=df['Age'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6dd978fd0>
```

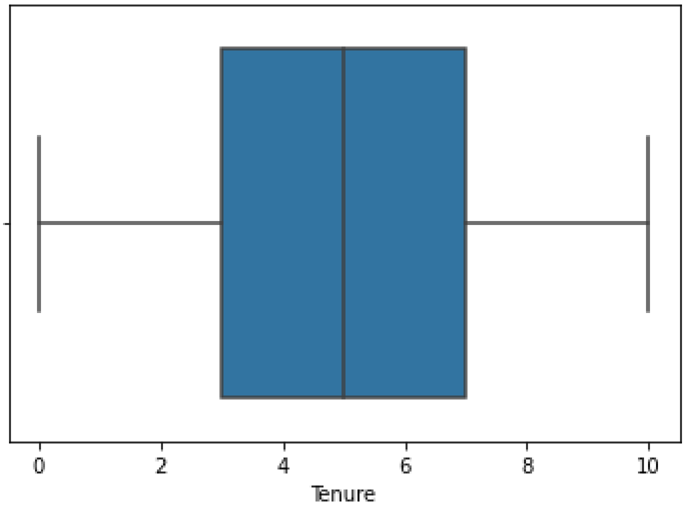


In [ ]:

```python
sns.boxplot(x=df['Tenure'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6d8dca710>
```



7.Check for categorical columns and perform encoding

In [ ]:

```
import pandas as pd
df = pd.read_csv("Churn_Modelling.csv", header=None)
```

In [ ]:

```
cols = df.columns
num_cols = df._get_numeric_data().columns
```

In [ ]:

```
num_cols
```

Out[ ]:

```
Int64Index([], dtype='int64')
```

In [ ]:

```
list(set(cols) - set(num_cols))
```

Out[ ]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

8.Split the data into dependent and independent variables

In [ ]:

```
# x  -Independent
# y  -Dependent
x =df.drop('Exited',axis=1)
y=df['Exited']
```

In [ ]:

```
x.head()
```

Out[ ]:

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

In [ ]:

```
y.head()
```

```
Out[ ]: 0    1
        1    0
        2    1
        3    0
        4    0
        Name: Exited, dtype: int64
```

9.Scale the independent variables

```
In [ ]: from sklearn import linear_model
        from sklearn.preprocessing import StandardScaler
        scale = StandardScaler()
```

```
In [ ]: X = df[['Balance', 'Tenure']]

        scaledX = scale.fit_transform(X)

        print(scaledX)
```

```
[[-1.22584767 -1.04175968]
 [ 0.11735002 -1.38753759]
 [ 1.33305335  1.03290776]
 ...
 [-1.22584767  0.68712986]
 [-0.02260751 -0.69598177]
 [ 0.85996499 -0.35020386]]
```

10.Split the data into training and testing

```
In [ ]: from sklearn.model_selection import train_test_split
```

```
In [ ]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```
In [ ]: print('X Train shape:{},Y.Train SHape:{}'.format(x_train.shape,y_train.shape))
```

```
X Train shape:(8000, 13),Y.Train SHape:(8000,)
```

```
In [ ]: print('X Test Shape :{},Y Test SHape:{}'.format(x_test.shape,y_test.shape))
```

```
X Test Shape :(2000, 13),Y Test SHape:(2000,)
```