

Assignment -2
Python Programming

Assignment Date	26 September 2022
Student Name	Sinthuri S
Student Roll Number	820419106054
Maximum Marks	2 Marks

Questions

1. Download the dataset: 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 libraries
import pandas as pd
import numpy as np
```

In []:

```
df=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Churn_Modelling.csv")
```

In []:

```
df.head()
```

Out[16]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

=

In []:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In []:

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

Out[17]:

	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

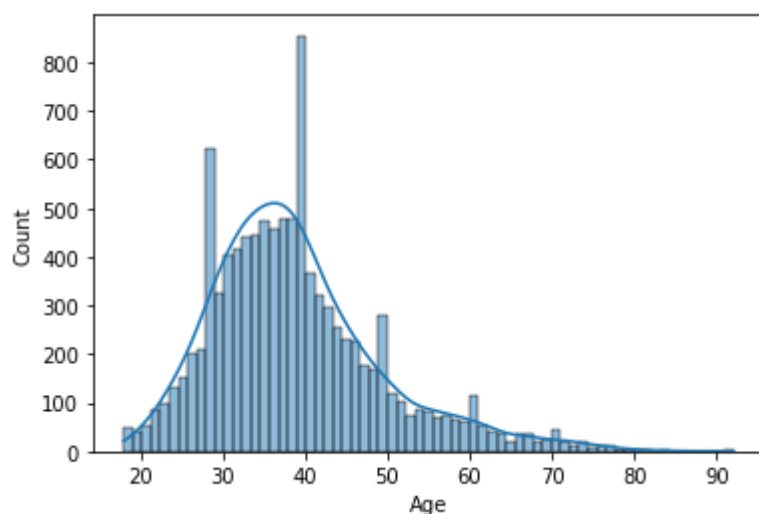
=

In []:

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

Out[18]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fd927df2c90>



In []:

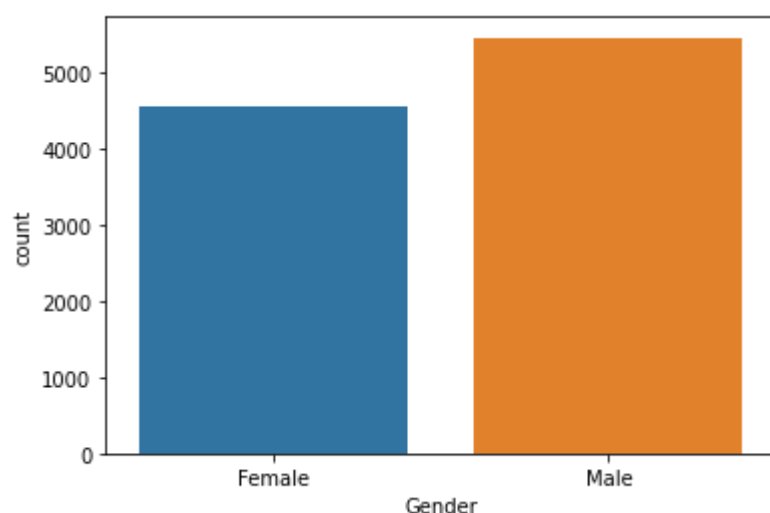
```
#plot for the gender column  
sns.countplot(df.Gender)
```

/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

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fd927a4f1d0>



In []:

```
#Bivariate Analysis
```

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

Out[20]:

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

■

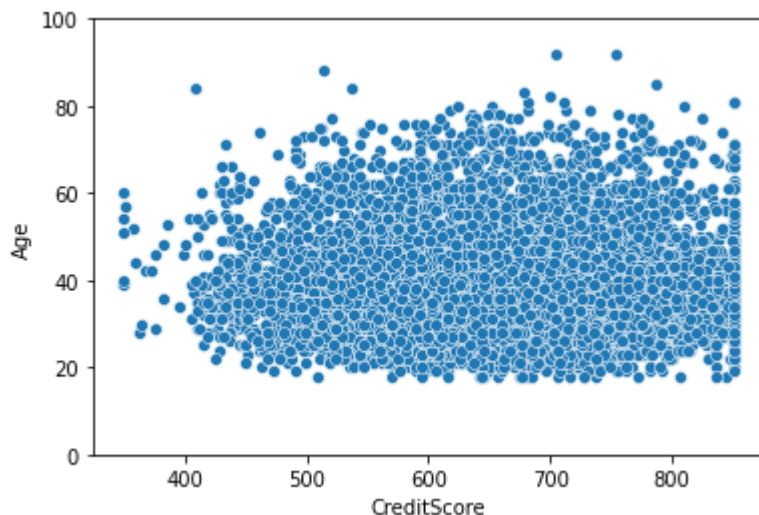
In []:

```
sns.scatterplot(df.CreditScore, df.Age)  
plt.ylim(0, 100)
```

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

Out[21]:

(0.0, 100.0)



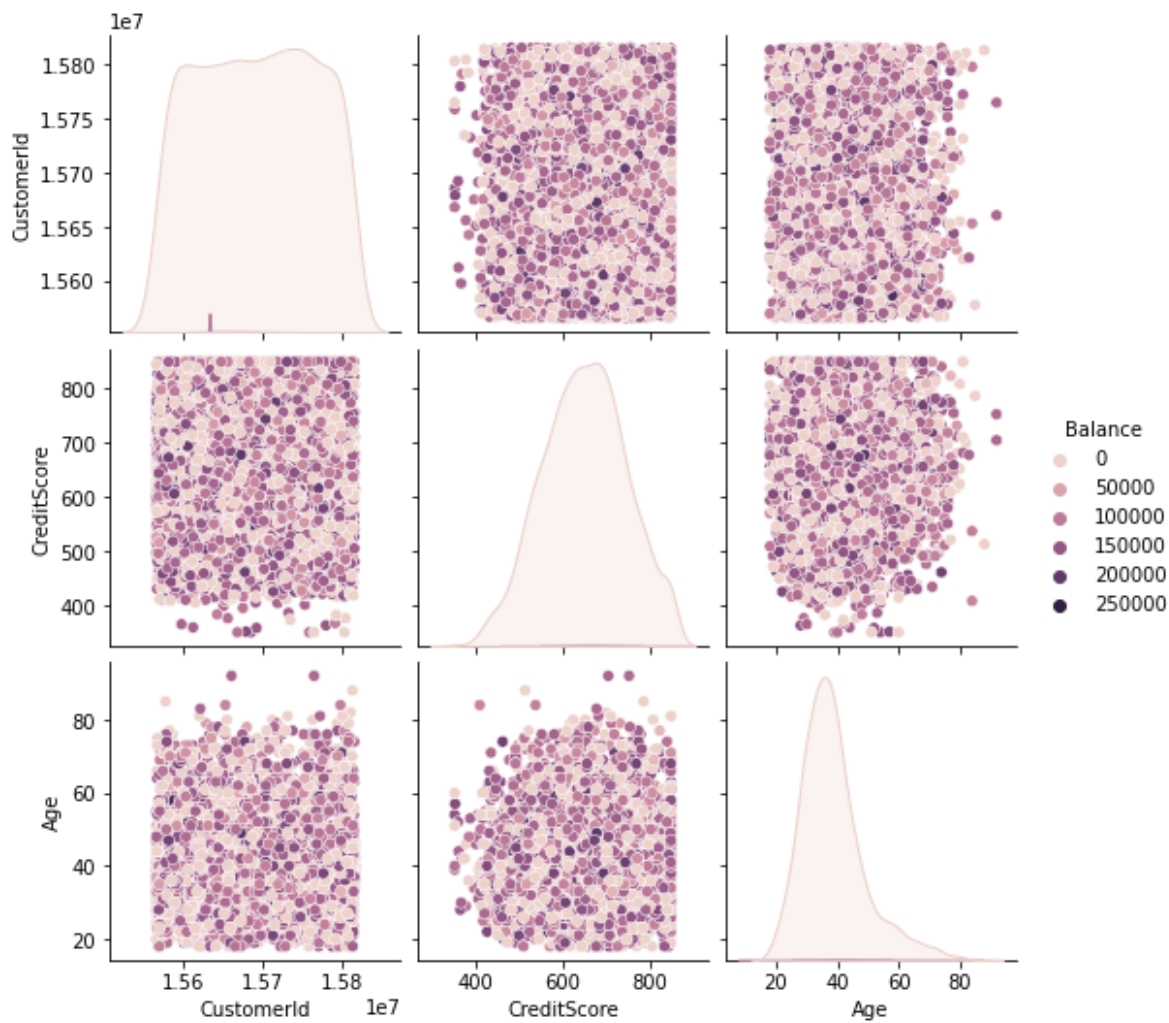
In []:

```
#Multivariate Analysis
```

```
sns.pairplot(data=df[['CustomerId', 'Geography', 'Gender', 'CreditScore', 'Age', 'Balance']], hue
```

Out[22]:

<seaborn.axisgrid.PairGrid at 0x7fd927106990>



In []:

```
data.describe()
```

Out[4]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	N
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	
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	

=



In []:

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

Out[23]:

```
0    37
dtype: int64
```

In []:

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

Out[24]:

38.9218

In []:

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

Out[25]:

38.92

In []:

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

Out[26]:

37.0

In []:

```
#check for missing values  
data.isnull().sum()
```

Out[5]:

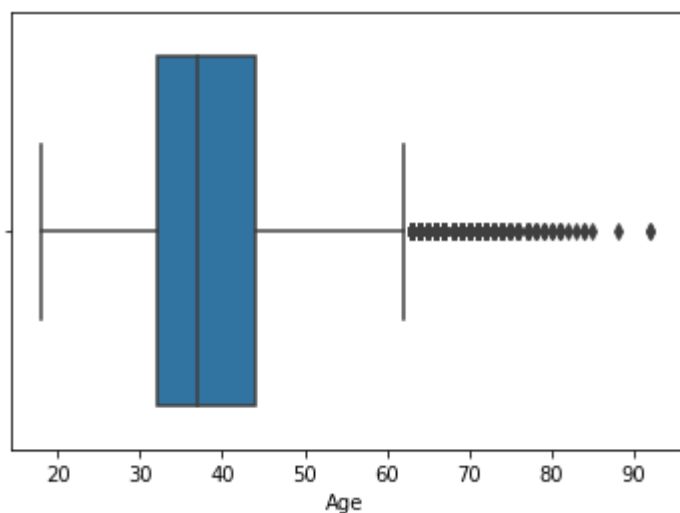
```
RowNumber      0  
CustomerId     0  
Surname        0  
CreditScore    0  
Geography      0  
Gender         0  
Age            0  
Tenure         0  
Balance        0  
NumOfProducts 0  
HasCrCard      0  
IsActiveMember 0  
EstimatedSalary 0  
Exited         0  
dtype: int64
```

In []:

```
#find the outliers and replace the outliers  
import seaborn as sns  
sns.boxplot(x=df['Age'])
```

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fd922451290>

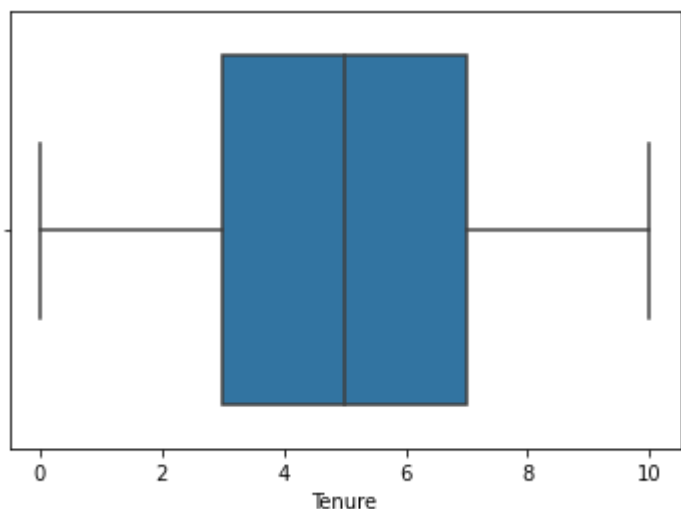


In []:

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

Out[28]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fd927106950>



In []:

```
#encoding  
from sklearn.preprocessing import LabelEncoder
```

In []:

```
le=LabelEncoder()
```

In []:

```
data['Geography']=le.fit_transform(data['Geography'])  
data['Gender']=le.fit_transform(data['Gender'])
```

In []:

```
data.head()
```

Out[11]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	0	0	42	2	0.0
1	2	15647311	Hill	608	2	0	41	1	83807.8
2	3	15619304	Onio	502	0	0	42	8	159660.8
3	4	15701354	Boni	699	0	0	39	1	0.0
4	5	15737888	Mitchell	850	2	0	43	2	125510.8

==

In []:

```
y=data['EstimatedSalary']
x=data.drop(columns=['EstimatedSalary'],axis=1)
```

In []:

```
names=x.columns
```

In []:

```
names
```

Out[17]:

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

In []:

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

In []:

```
x.head()
```

Out[42]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8



In []:

```
y.head()
```

Out[43]:

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

In []:

```
#scale the independent variables
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]]
```

In []:

```
#splitting data into train and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print('X Train shape:{},Y.Train shape:{}'.format(x_train.shape,y_train.shape))
print('X Test Shape:{},Y Test Shape{}'.format(x_test.shape,y_test.shape))
```

```
X Train shape:(8000, 13),Y.Train shape:(8000,)
X Test Shape:(2000, 13),Y Test Shape(2000,)
```