ASSIGNMENT - 2

Assignment Date	24 September 2022
Student Name	KHAMILA BANU K
Student Roll Number	2019504539
Maximum Marks	2 Marks

1) Load the dataset:

Solution : `data=pd.read_csv("Churn_Modelling.csv")`

Output :

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

2) Visualisations:

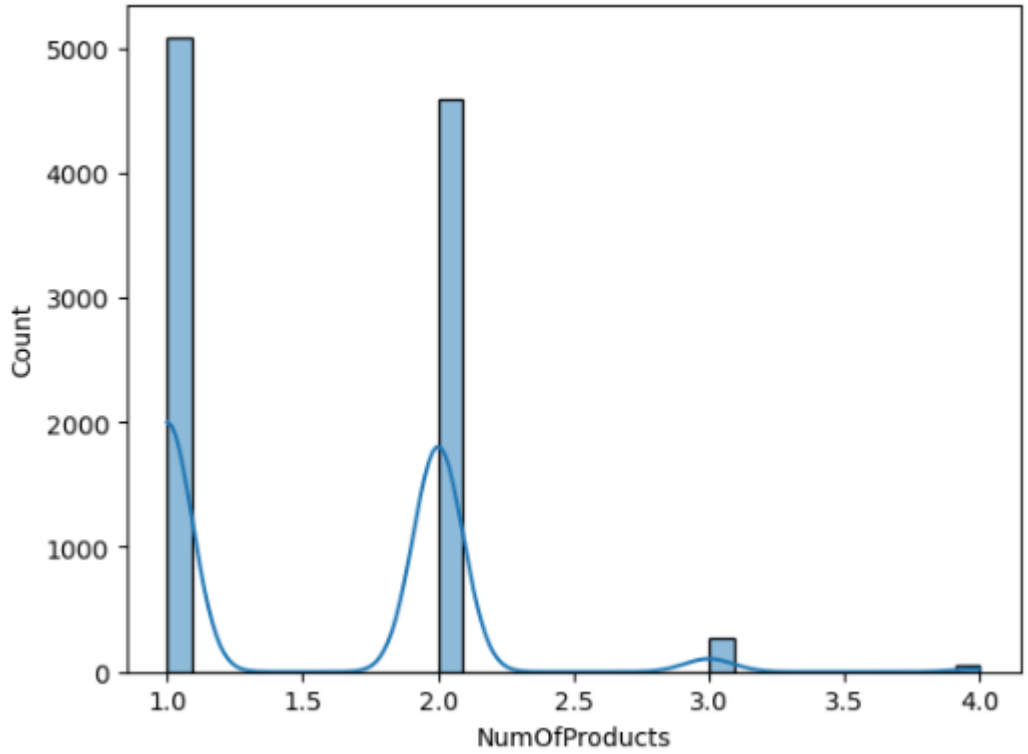
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Univariate Analysis :

```
sns.histplot(data.NumOfProducts,kde= True)
```

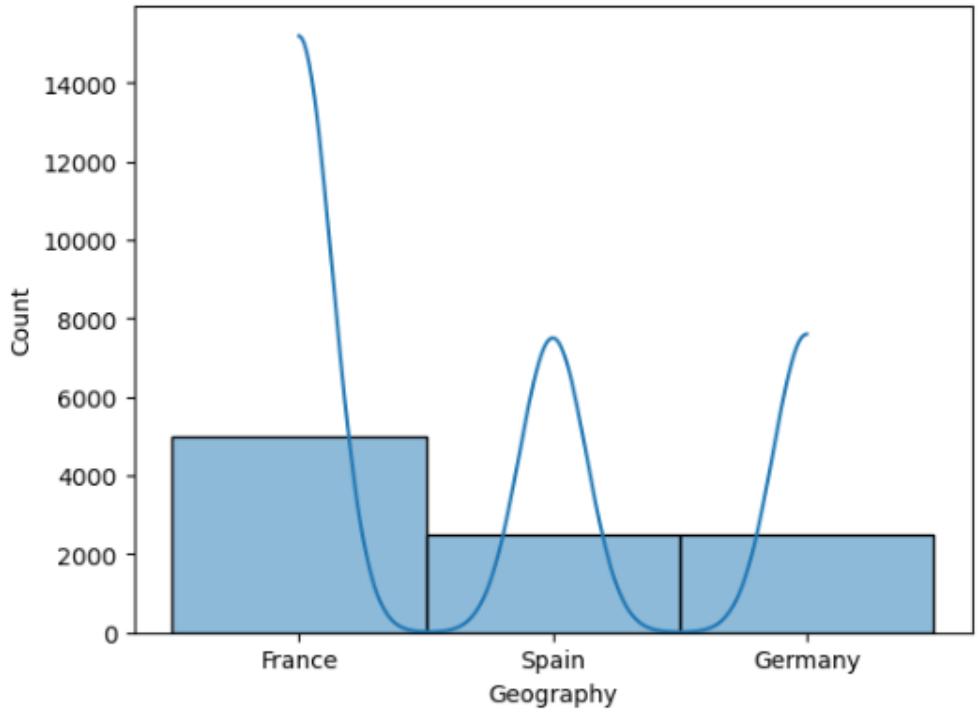
Output:

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



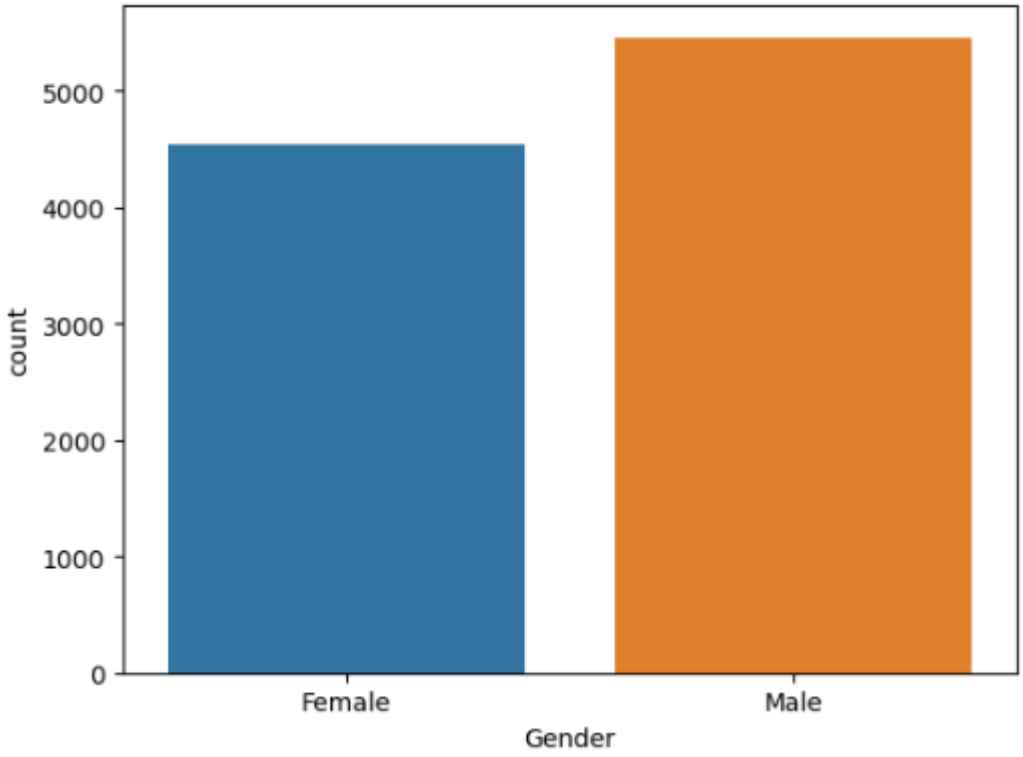

```
sns.histplot(data.Geography,kde= True)
```

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



```
sns.countplot(data.Gender)
```

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



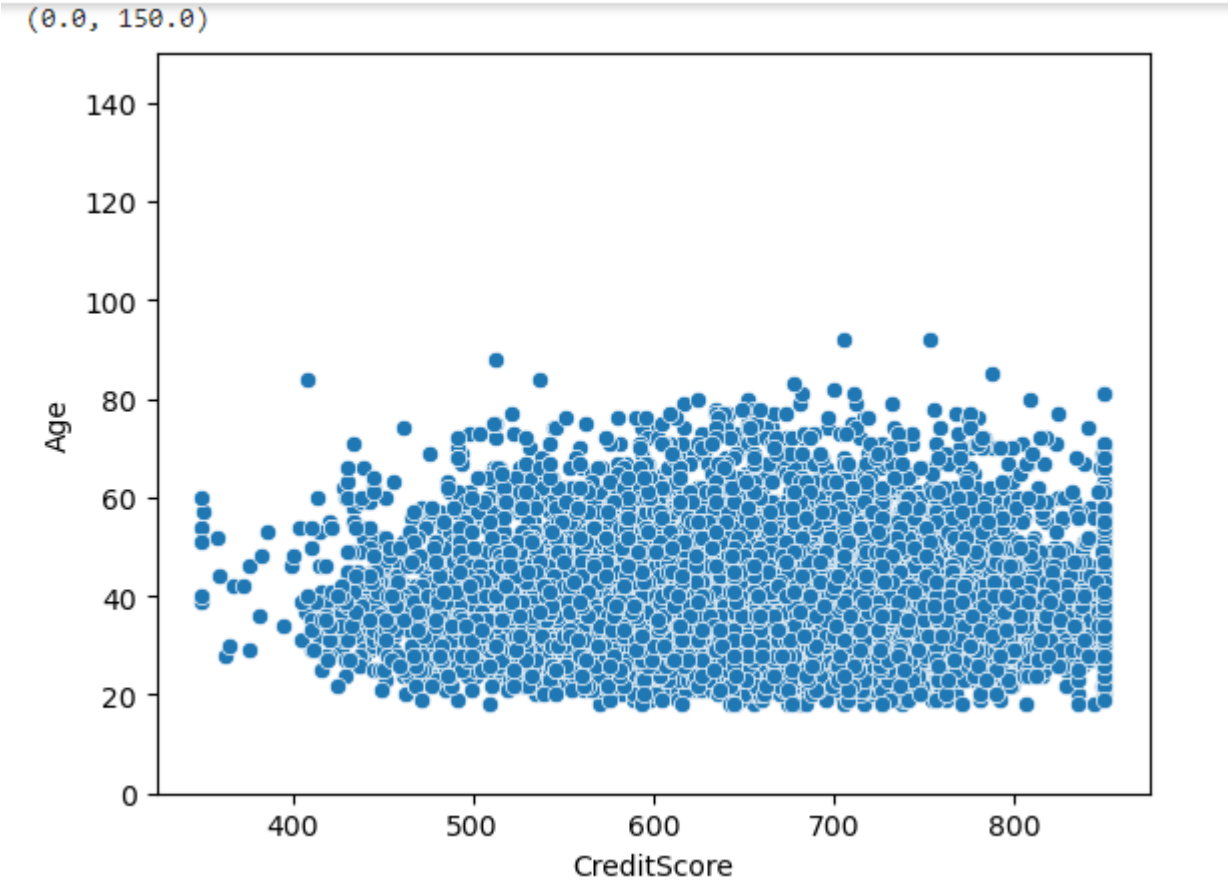
Bivariate Analysis:

```
data[['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure']].corr()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000

```
sns.scatterplot(data.CreditScore,data.Age)
plt.ylim(0,150)
```

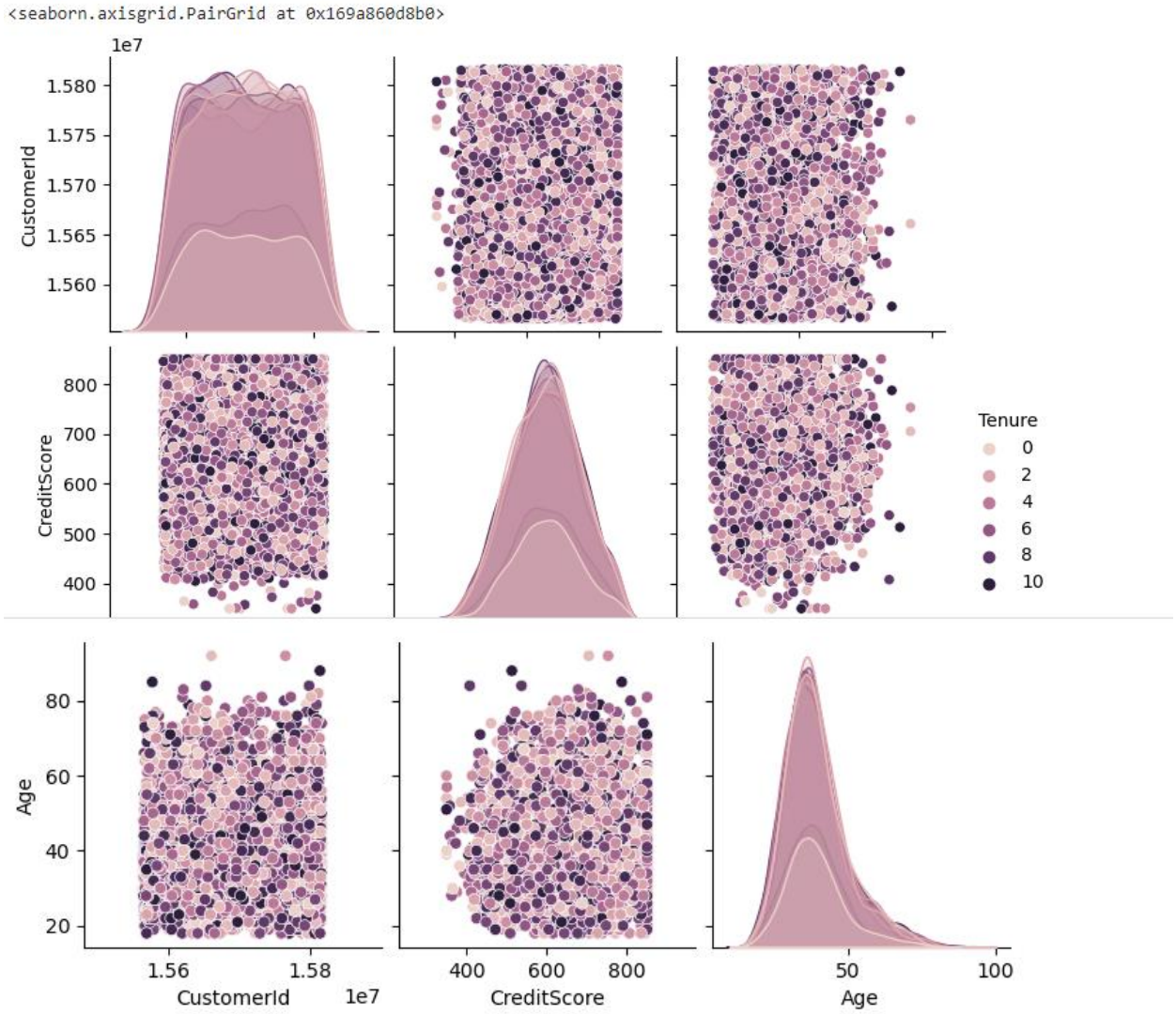
Output :



Multivariate Analysis:

```
sns.pairplot(data=data[['CustomerId', 'Surname', 'CreditScore',
'Geography','Gender', 'Age', 'Tenure']],hue='Tenure')
```

Output :



4) Perform descriptive statistics on the dataset:

Solution : `data.describe()`

Output:

	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.dtypes`

Output:

```
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
```

5) Handling the Missing Values:

Solution : `data.isnull()`

Output :

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

10000 rows × 14 columns

```
data.isnull().sum()
```

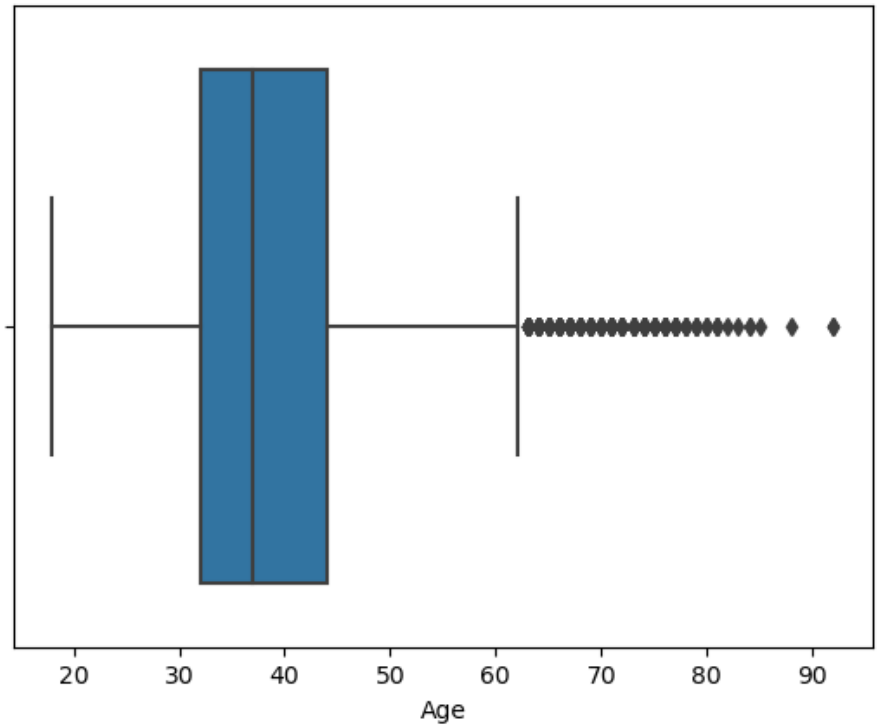
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: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

6) Find the outliers and replace the outliers :

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

Output:

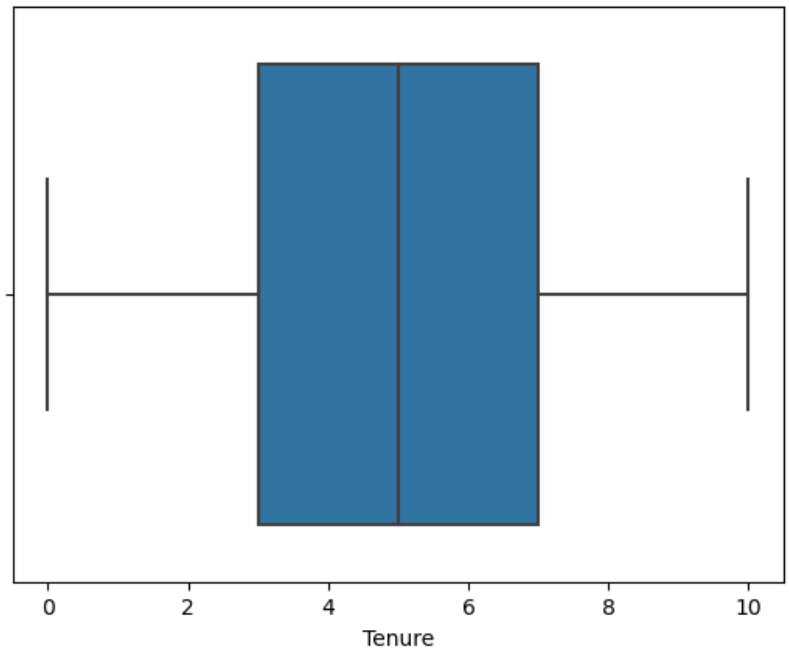
```
<AxesSubplot:xlabel='Age'>
```




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

Output :

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



7) Check for categorical columns and perform encoding:

Solution : `a=data.columns`

```
b=data._get_numeric_data().columns
b
```

Output :

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

Solution : `list(set(a) - set(b))`

Output : *['Gender', 'Surname', 'Geography']*

8) Split the data into dependent and independent variables :

Solution :

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

Output :

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

```
y.head()
```

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

9) Scale the independent Variables :

Solution :

```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
scale = StandardScaler()
x=data[['Age','Tenure']]
scaledx = scale.fit_transform(x)
print(scaledx)
```

Output :

```
[[ 0.29351742 -1.04175968]
 [ 0.19816383 -1.38753759]
 [ 0.29351742  1.03290776]
 ...
 [-0.27860412  0.68712986]
 [ 0.29351742 -0.69598177]
 [-1.04143285 -0.35020386]]
```

10) Split the data into training and testing :

Solution :

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(Ind,Dep,test_size=0.2,ran
dom_state=0)
x_train.shape()
```

Output : (8000,10)

```
x_test.shape()
```

Output : (2000,10)