Assignment 2

1)Download the dataset: Dataset data set is churn_modeling.csv 2)Load the dataset.

```
import pandas as pd


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

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	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.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

3)Perform Below Visualizations. ● Univariate Analysis ● Bi - Variate Analysis ● Multi - Variate Analysis

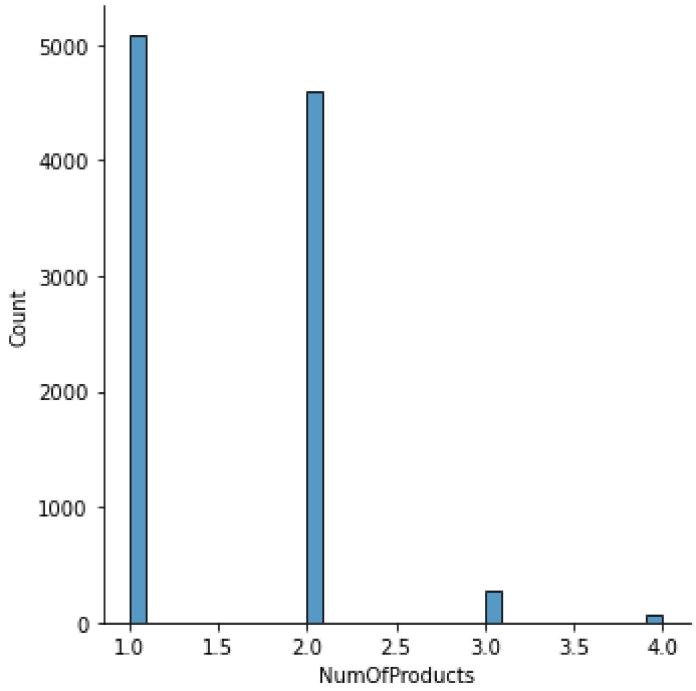
```
import seaborn as sns


#univariate
sns.displot(df['CreditScore'])
```

```
<seaborn.axisgrid.FacetGrid at 0x2c627fbdf70>
```

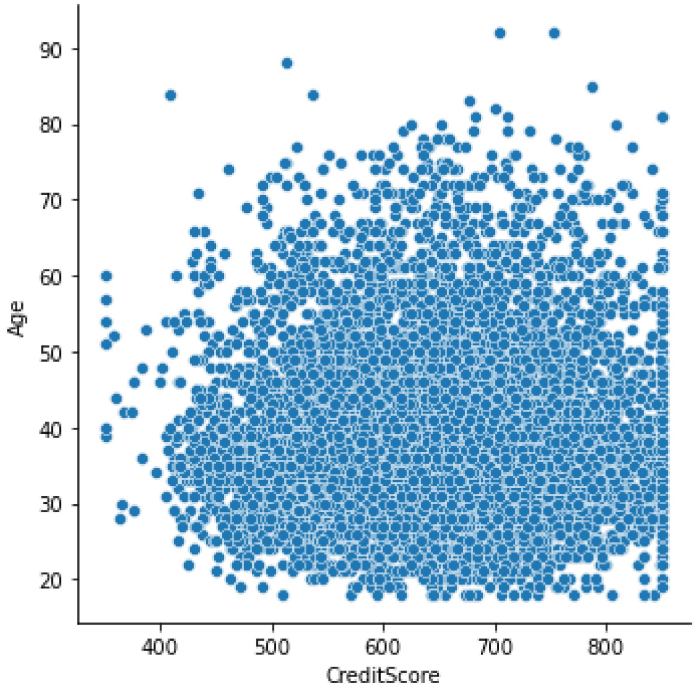

```
sns.displot(df['NumOfProducts'])
```

```
<seaborn.axisgrid.FacetGrid at 0x2c628069160>
```



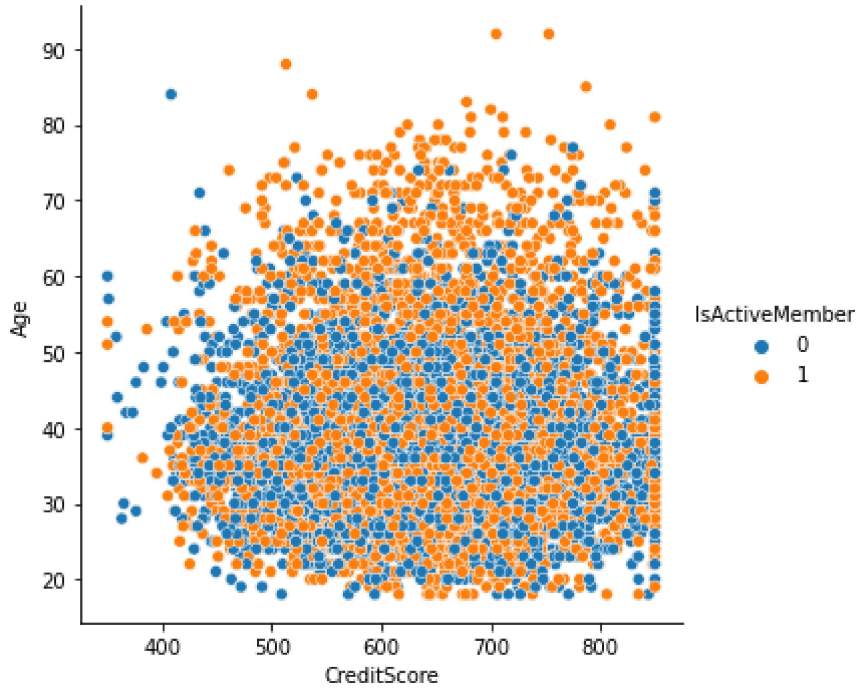

```
#bi variate
sns.relplot(x="CreditScore",y='Age',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x2c628846460>
```



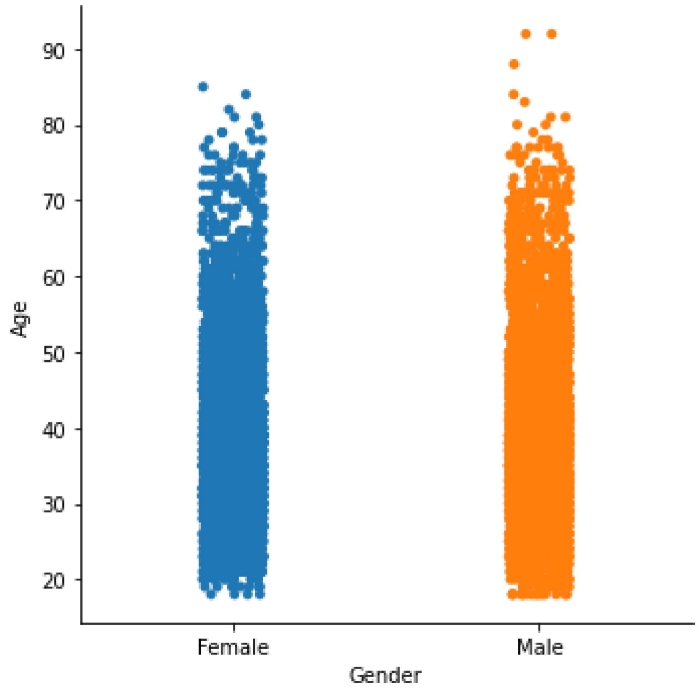
```
sns.relplot(x="CreditScore",y='Age',hue="IsActiveMember",data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x2c62890e760>
```




```
sns.catplot(x="Gender",y='Age',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x2c6299d7c70>
```




```
#multivariate
sns.pairplot(data=df,hue="Exited")
```

<seaborn.axisgrid.PairGrid at 0x2c628948f10>

4)Perform descriptive statistics on the dataset.

```
import pandas as pd
import numpy as np
df = pd.read_csv("Churn_Modelling.csv")
df.head(2)
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8

```
df.isnull().any()
```

```
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
```

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
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

5)Handle the Missing values.

50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	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.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

```
df.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
```

The dataset does not contain any missing values. If an dataset has any missing values,we can handle it in following ways 1) missimg values which above 50% of data-remove 2)less missing values -replace function used-fillna()

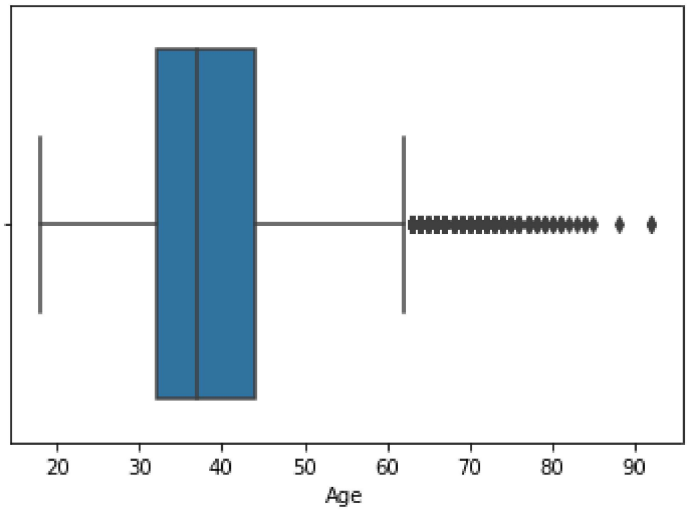
6)Find the outliers and replace the outliers

```
df.skew()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_10472\1665899112.py:1: FutureWarning: Dropping
  df.skew()
RowNumber          0.000000
CustomerId         0.001149
CreditScore       -0.071607
Age                1.011320
Tenure             0.010991
Balance           -0.141109
NumOfProducts      0.745568
HasCrCard         -0.901812
IsActiveMember    -0.060437
EstimatedSalary    0.002085
Exited             1.471611
dtype: float64
```

```
sns.boxplot(df["Age"])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass t
  warnings.warn(
<AxesSubplot:xlabel='Age'>
```




```
q1= df["Age"].describe()["25%"]
q3= df["Age"].describe()["75%"]
```

```
q1
```

```
32.0
```

```
q3
```

```
44.0
```

```
iqr=q3-q1
iqr
```

```
12.0
```

```
l_b=q1-(1.5*iqr)
u_b=q3+(1.5*iqr)
```

```
l_b
```

```
14.0
```

```
l_b=q1-(1.5*iqr)
u_b=q3+(1.5*iqr)
```

```
l_b
```

```
14.0
```

```
u_b
```

```
62.0
```

```
df[df["Age"]<l_b]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance

```
df[df["Age"]>u_b].head()
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance

```
df.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography            int32
Gender               int32
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
```

```
EstimatedSalary    float64
Exited               int64
dtype: object
```

```
outlier_list=list(df[df["Age"]>u_b]["Age"])
outlier_list
```

```
[]
```

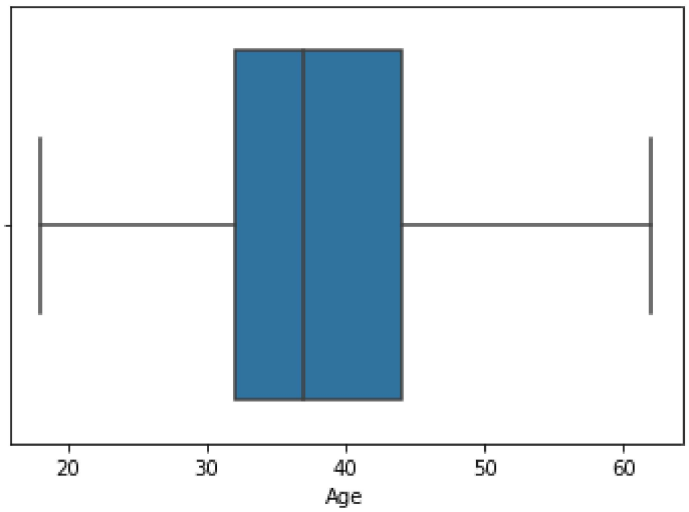
```
outlier_dict={}.fromkeys(outlier_list,u_b)
outlier_dict
```

```
{}
```

After removing outliers

```
df["Age"]=df["Age"].replace(outlier_dict)
sns.boxplot(df["Age"])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass t
  warnings.warn(
<AxesSubplot:xlabel='Age'>
```



7)Check for Categorical columns and perform encoding.

```
df.dtypes
```

```
RowNumber           int64
CustomerId          int64
Surname            object
CreditScore         int64
Geography           int32
Gender              int32
```

```
Age                 int64
Tenure              int64
Balance           float64
NumOfProducts       int64
HasCrCard           int64
IsActiveMember      int64
EstimatedSalary   float64
Exited              int64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder


le=LabelEncoder()
df['Geography']=le.fit_transform(df['Geography'])
df['Gender']=le.fit_transform(df['Gender'])


df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	0	0	42	2	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.(
4	5	15737888	Mitchell	850	2	0	43	2	125510.8

8)Split the data into dependent and independent variables.

```
y=df['Exited']
x=df.drop(columns=['Exited','RowNumber','Surname'],axis=1)


y
```

```
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
```

x

	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
0	15634602	619	0	0	42	2	0.00	1
1	15647311	608	2	0	41	1	83807.86	1
2	15619304	502	0	0	42	8	159660.80	3
3	15701354	699	0	0	39	1	0.00	2
4	15737888	850	2	0	43	2	125510.82	1
...	...	...	...	...	...	...	...	...
9995	15606229	771	0	1	39	5	0.00	2
9996	15569892	516	0	1	35	10	57369.61	1
9997	15584532	709	0	0	36	7	0.00	1
9998	15682355	772	1	1	42	3	75075.31	2
9999	15628319	792	0	0	28	4	130142.79	1

10000 rows × 11 columns

9)Scale the independent variables

```
col_names=x.columns
from sklearn.preprocessing import scale
```

```
x=scale(x)
x
```

```
array([[-0.78321342, -0.32622142, -0.90188624, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.60653412, -0.44003595,  1.51506738, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-0.99588476, -1.53679418, -0.90188624, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [-1.47928179,  0.60498839, -0.90188624, ..., -1.54776799,
         0.97024255, -1.00864308],
       [-0.11935577,  1.25683526,  0.30659057, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [-0.87055909,  1.46377078, -0.90188624, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

```
x=pd.DataFrame(x,columns=col_names) #Convert the array back to the DataFrame
x
```

	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumO
0	-0.783213	-0.326221	-0.901886	-1.095988	0.342615	-1.041760	-1.225848	
1	-0.606534	-0.440036	1.515067	-1.095988	0.240011	-1.387538	0.117350	
2	-0.995885	-1.536794	-0.901886	-1.095988	0.342615	1.032908	1.333053	
3	0.144767	0.501521	-0.901886	-1.095988	0.034803	-1.387538	-1.225848	
4	0.652659	2.063884	1.515067	-1.095988	0.445219	-1.041760	0.785728	
...	...	...	...	...	...	...	...	
9995	-1.177652	1.246488	-0.901886	0.912419	0.034803	-0.004426	-1.225848	
9996	-1.682806	-1.391939	-0.901886	0.912419	-0.375612	1.724464	-0.306379	
9997	-1.479282	0.604988	-0.901886	-1.095988	-0.273008	0.687130	-1.225848	
9998	-0.119356	1.256835	0.306591	0.912419	0.342615	-0.695982	-0.022608	
9999	-0.870559	1.463771	-0.901886	-1.095988	-1.093840	-0.350204	0.859965	

10000 rows × 11 columns

10)Split the data into training 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)
```

```
x_train.shape
```

```
(8000, 11)
```

```
x_test.shape
```

```
(2000, 11)
```

```
y_train.shape
```

```
(8000,)
```

```
y_test.shape
```

```
(2000,)
```

Colab paid products - Cancel contracts here