ASSIGNMENT 2

Assignment Date	29 September 2022
Student Name	SIVALESHWARI.M
Student Roll Number	E1194037
Maximum Marks	2 Marks

Data Visualization and Pre-processing

Perform Below Tasks to complete the assignment:-

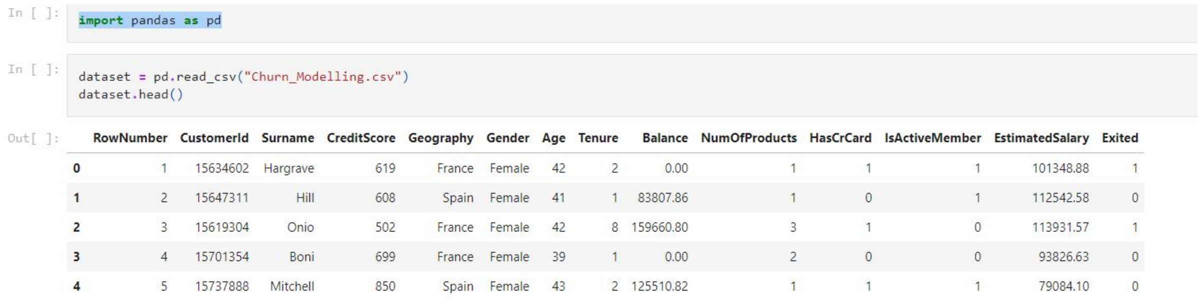
Tasks:-

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 & independentvariables
9. Scale the independent variables
10. Split the data into training and testing

SOLUTIONS:

ASSIGNMENT 2

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



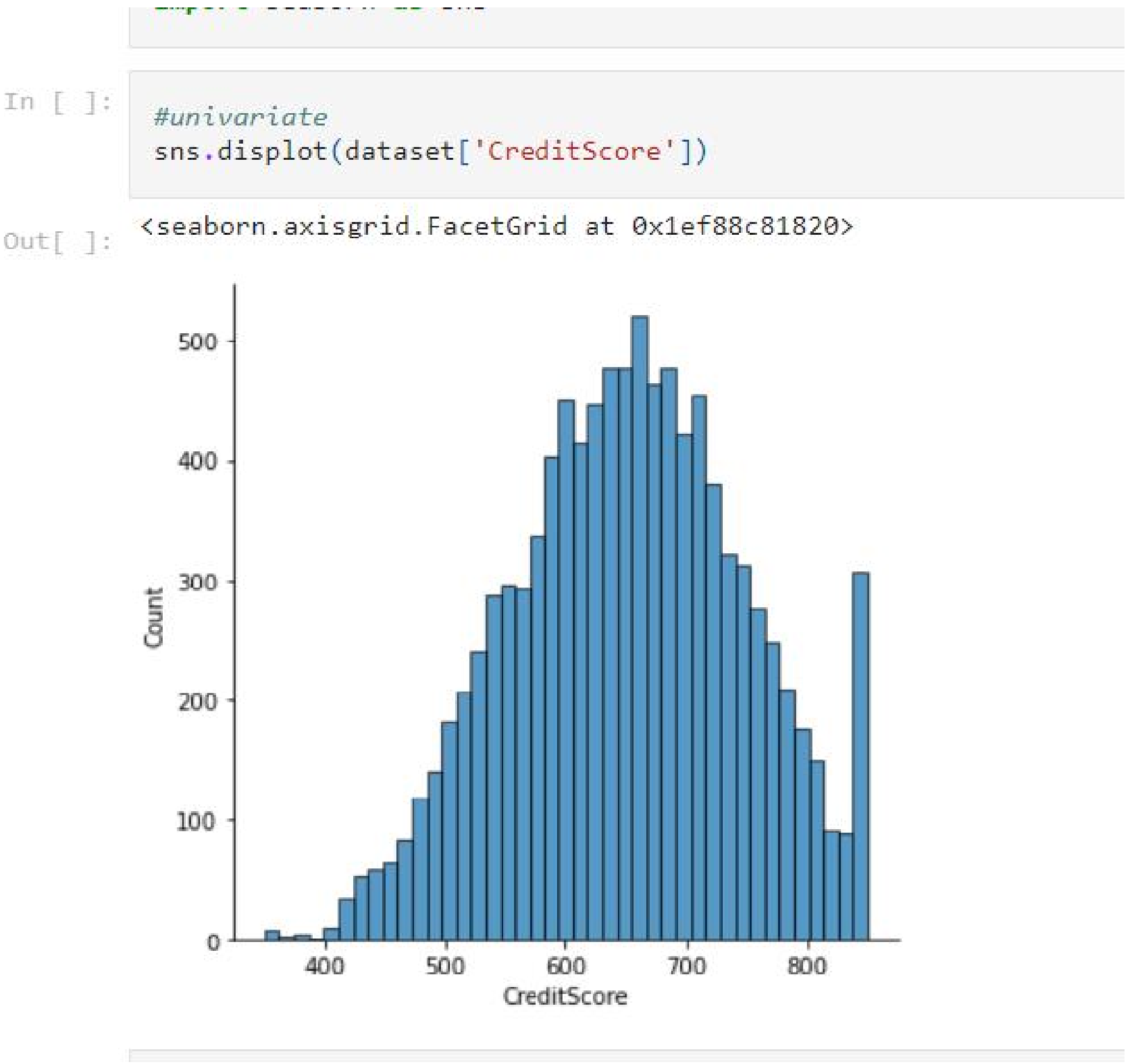
```
In [ ]: import pandas as pd
```

```
In [ ]: dataset = pd.read_csv("Churn_Modelling.csv")
        dataset.head()
```

Out[ ]:

	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

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

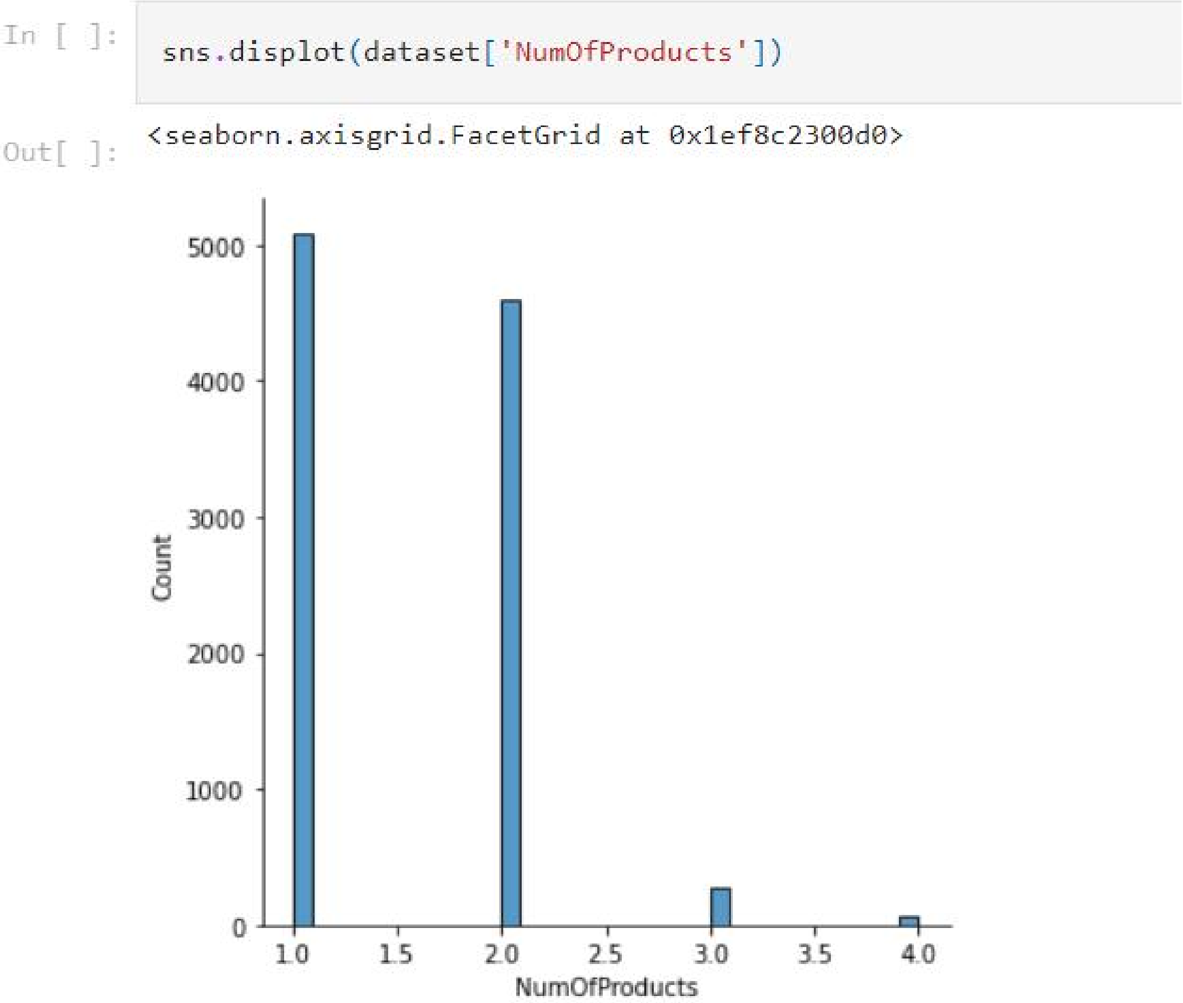


```
import seaborn as sns
```

```
In [ ]: #univariate
        sns.displot(dataset['CreditScore'])
```

```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1ef88c81820>
```

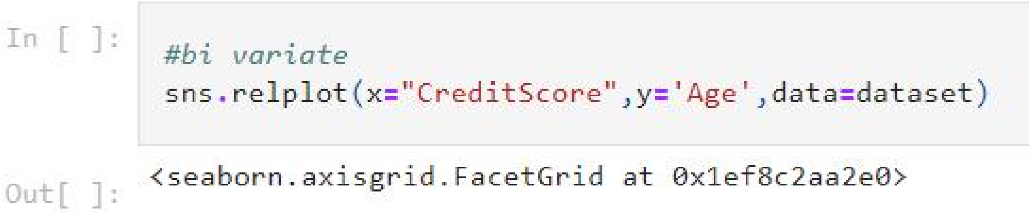
ASSIGNMENT 2



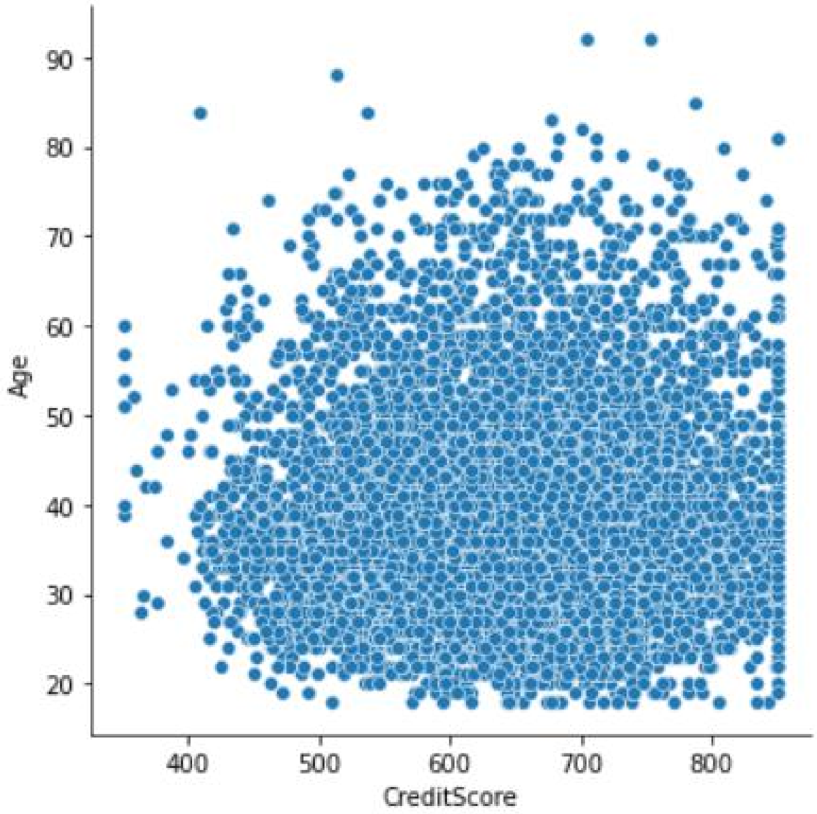
```
In [ ]:  sns.displot(dataset['NumOfProducts'])
```

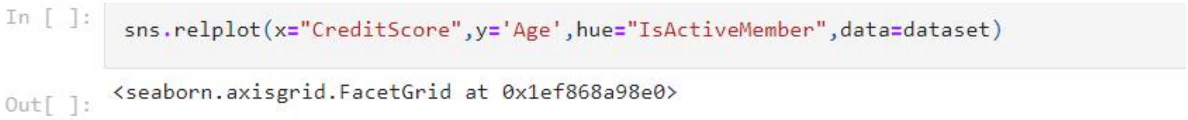
```
Out[ ]:  <seaborn.axisgrid.FacetGrid at 0x1ef8c2300d0>
```

ASSIGNMENT 2

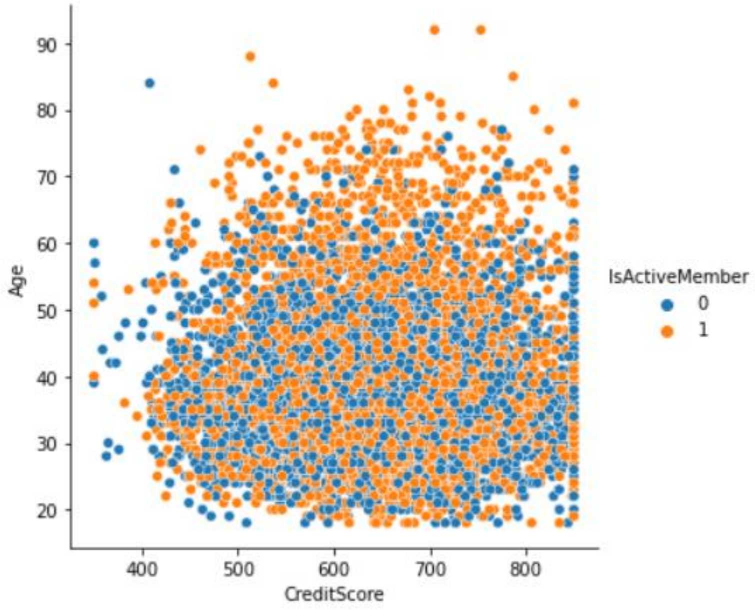


```
In [ ]:  #bi variate
         sns.relplot(x="CreditScore",y='Age',data=dataset)

Out[ ]:  <seaborn.axisgrid.FacetGrid at 0x1ef8c2aa2e0>
```

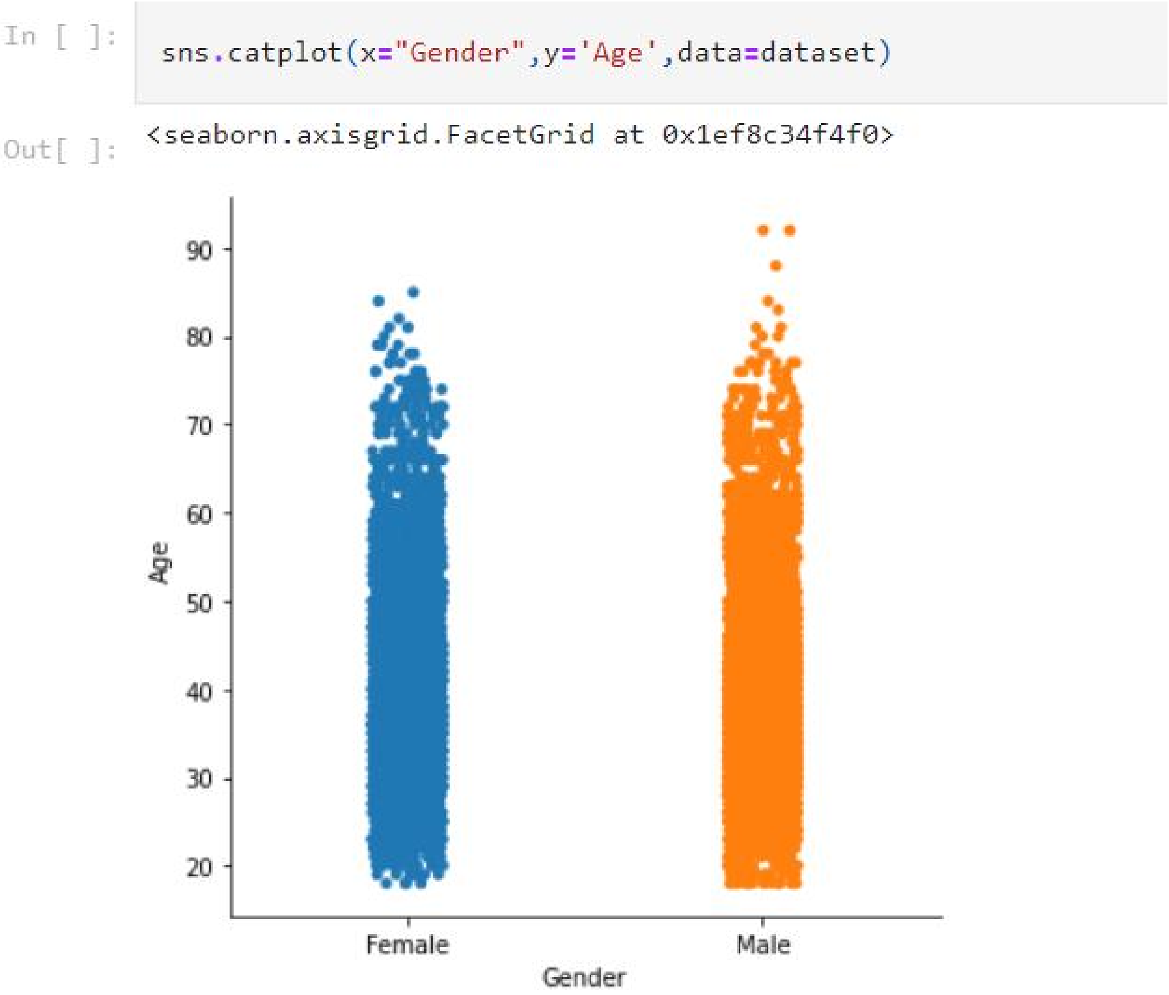




```
In [ ]:  sns.relplot(x="CreditScore",y='Age',hue="IsActiveMember",data=dataset)

Out[ ]:  <seaborn.axisgrid.FacetGrid at 0x1ef868a98e0>
```

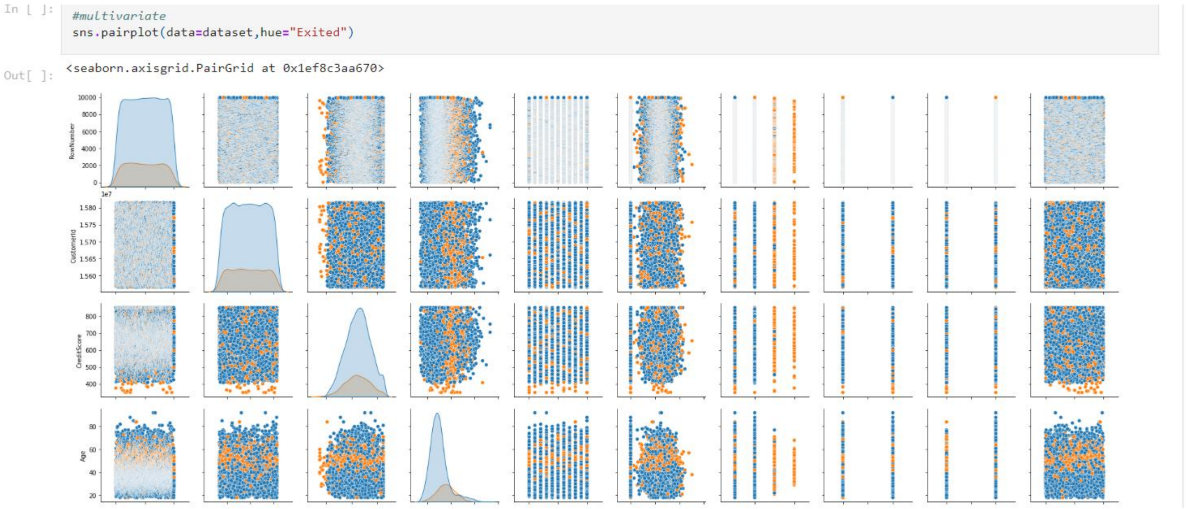


ASSIGNMENT 2



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

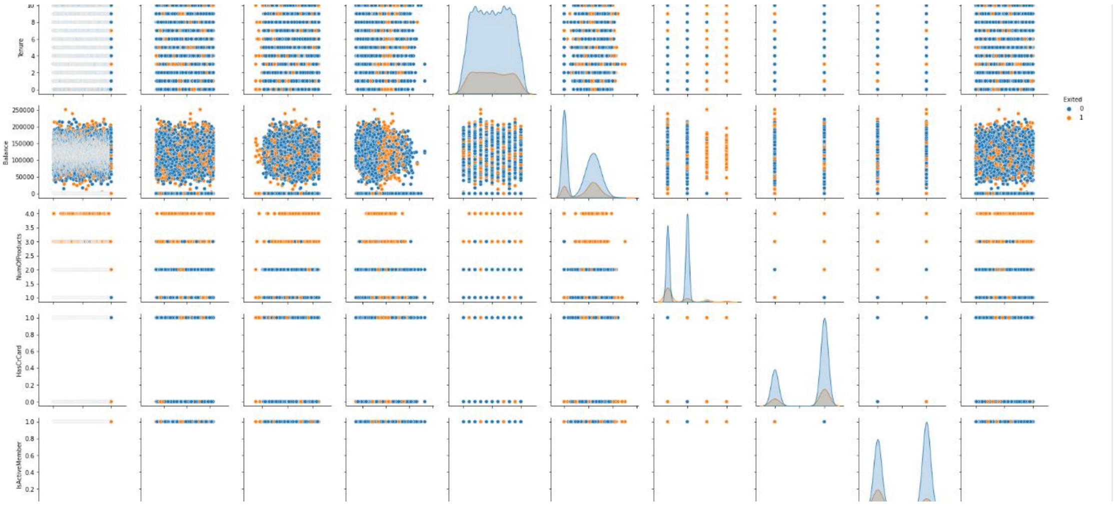
```
<seaborn.axisgrid.FacetGrid at 0x1ef8c34f4f0>
```



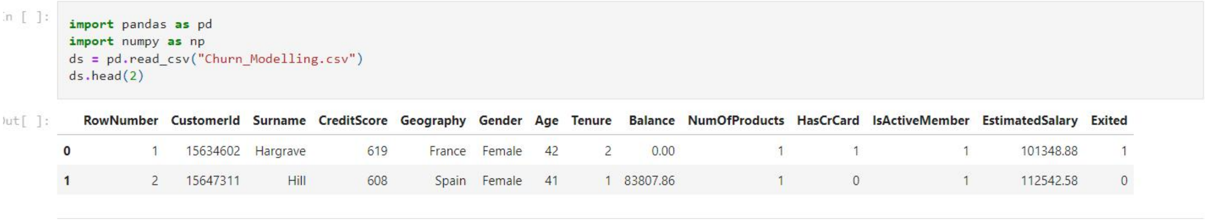
```
#multivariate
sns.pairplot(data=dataset,hue="Exited")
```

```
<seaborn.axisgrid.PairGrid at 0x1ef8c3aa670>
```

ASSIGNMENT 2



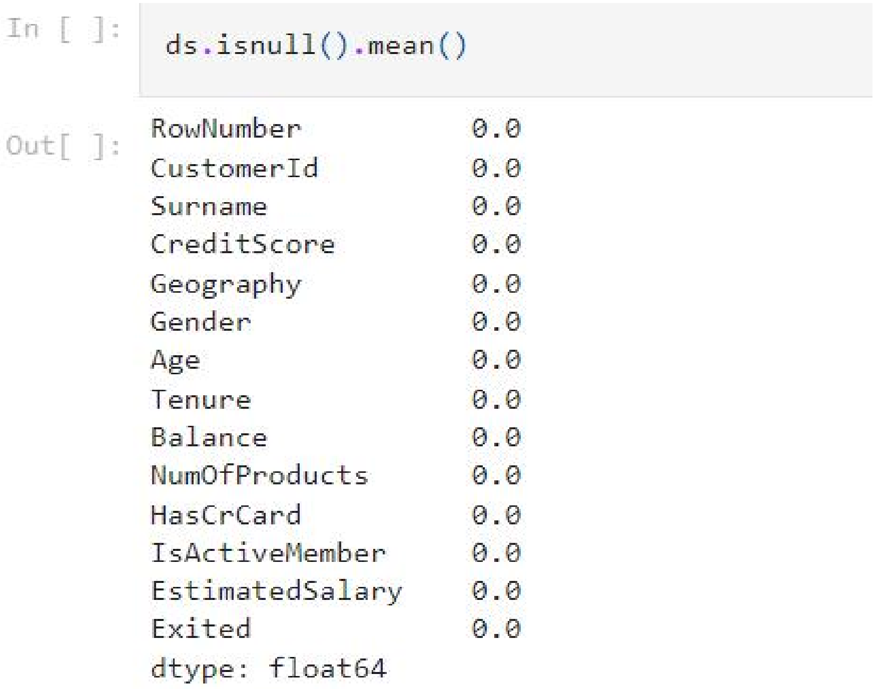
4)Perform descriptive statistics on the dataset.



```
In [ ]: import pandas as pd
        import numpy as np
        ds = pd.read_csv("Churn_Modelling.csv")
        ds.head(2)
```

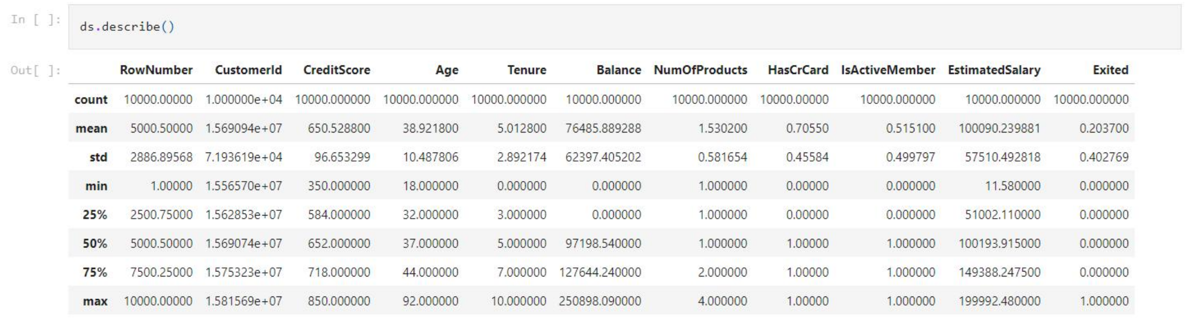
Out[ ]:

	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



```
In [ ]: ds.isnull().mean()

Out[ ]: RowNumber          0.0
        CustomerId         0.0
        Surname            0.0
        CreditScore        0.0
        Geography          0.0
        Gender             0.0
        Age                0.0
        Tenure             0.0
        Balance            0.0
        NumOfProducts      0.0
        HasCrCard          0.0
        IsActiveMember     0.0
        EstimatedSalary    0.0
        Exited             0.0
        dtype: float64
```

ASSIGNMENT 2

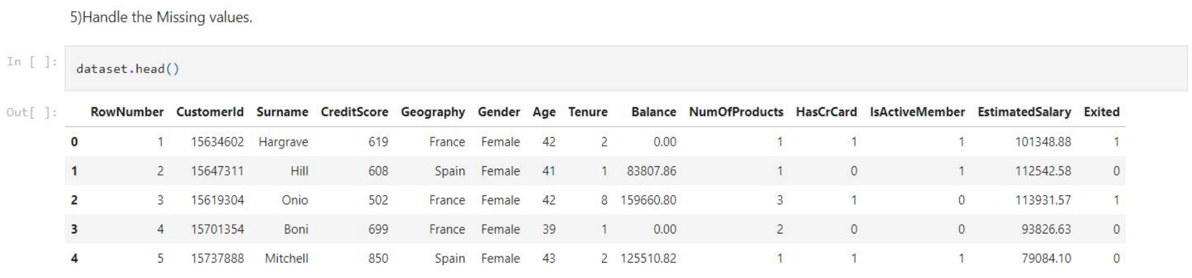


In [ ]:

```
ds.describe()
```

Out[ ]:

	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



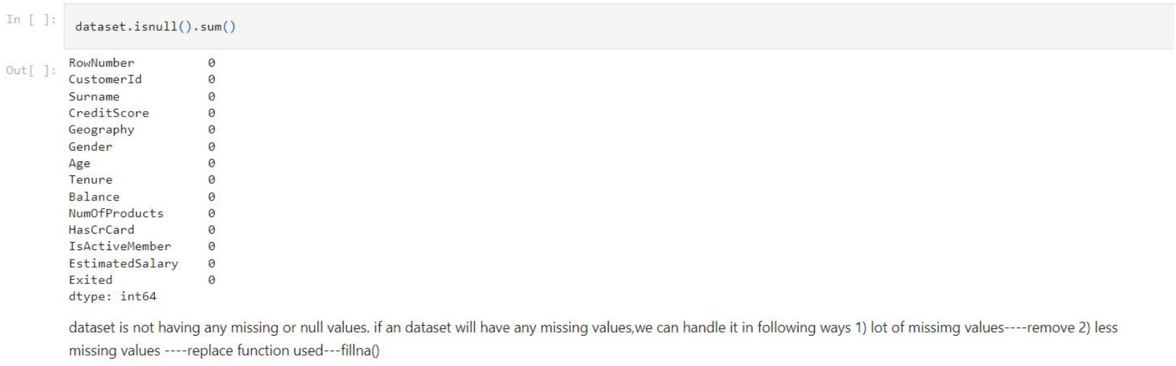
5)Handle the Missing values.

In [ ]:

```
dataset.head()
```

Out[ ]:

	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



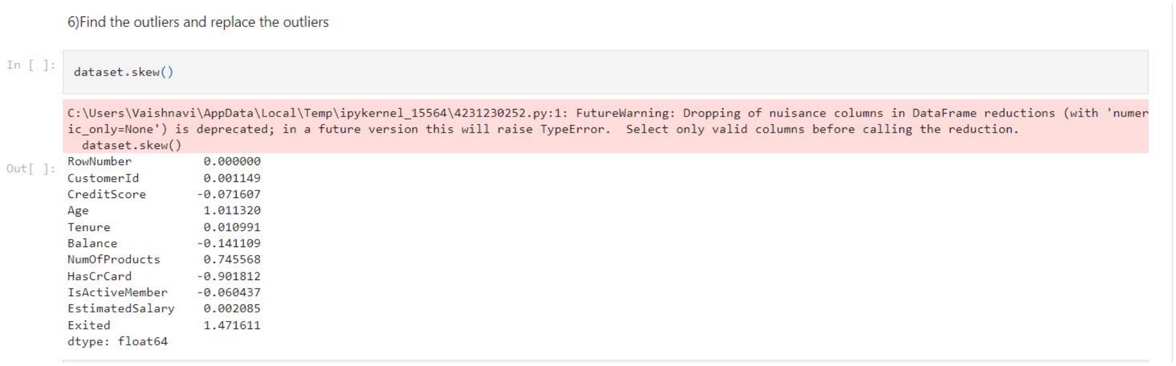
In [ ]:

```
dataset.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
```

dataset is not having any missing or null values. if an dataset will have any missing values,we can handle it in following ways 1) lot of missimg values----remove 2) less missing values ----replace function used---fillna()



6)Find the outliers and replace the outliers

In [ ]:

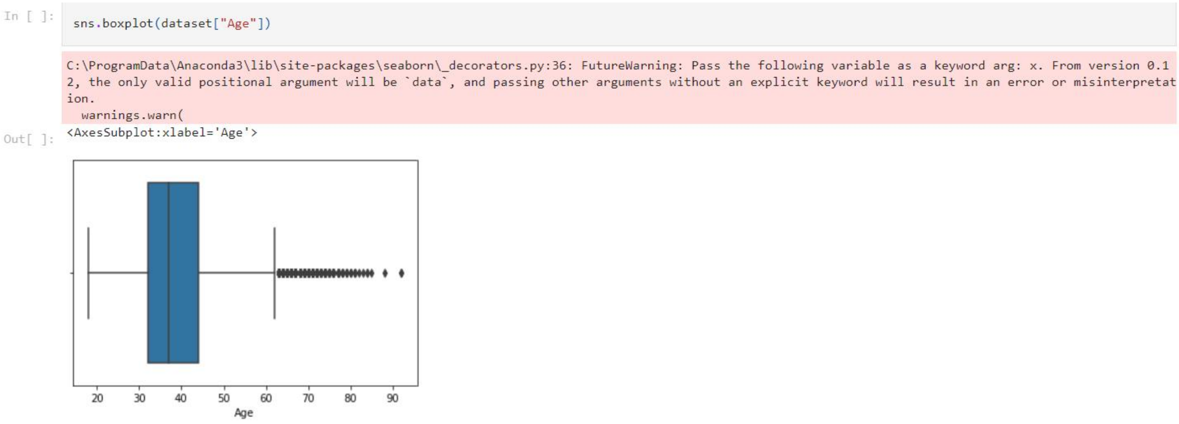
```
dataset.skew()
```

```
C:\Users\Vaishnavi\AppData\Local\Temp\ipykernel_15564\4231230252.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numer
ic_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  dataset.skew()
```

Out[ ]:

```
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
```

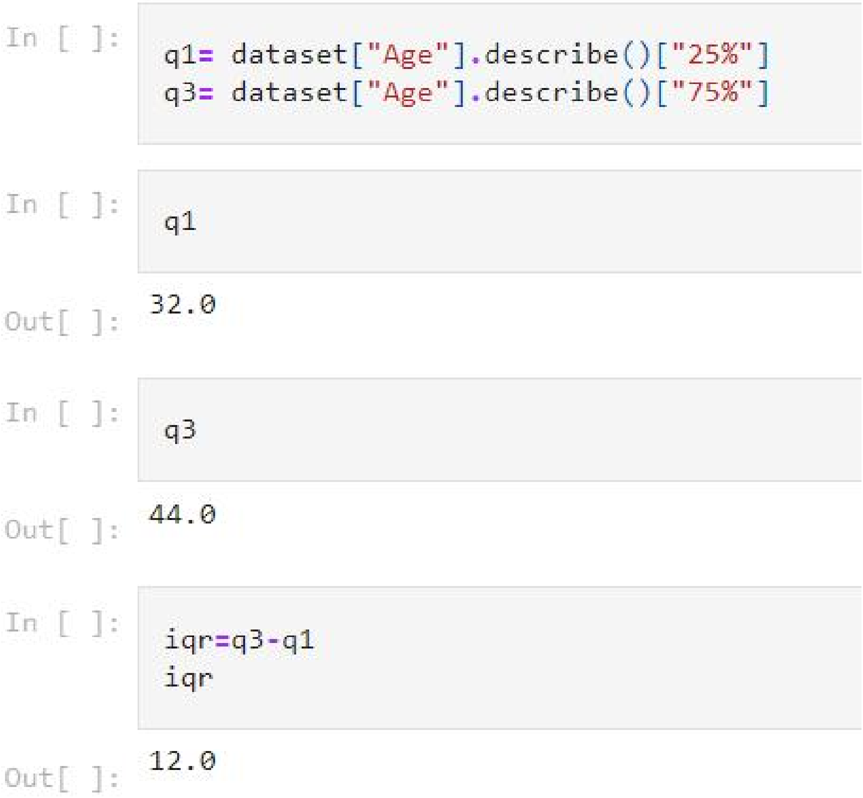
ASSIGNMENT 2



In [ ]:
```
sns.boxplot(dataset["Age"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.1
2, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

Out[ ]:
```
<AxesSubplot:xlabel='Age'>
```



In [ ]:
```
q1= dataset["Age"].describe()["25%"]
q3= dataset["Age"].describe()["75%"]
```

In [ ]:
```
q1
```

Out[ ]:
```
32.0
```

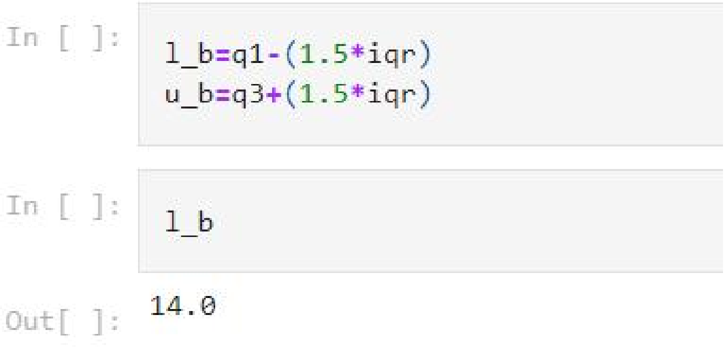
In [ ]:
```
q3
```

Out[ ]:
```
44.0
```

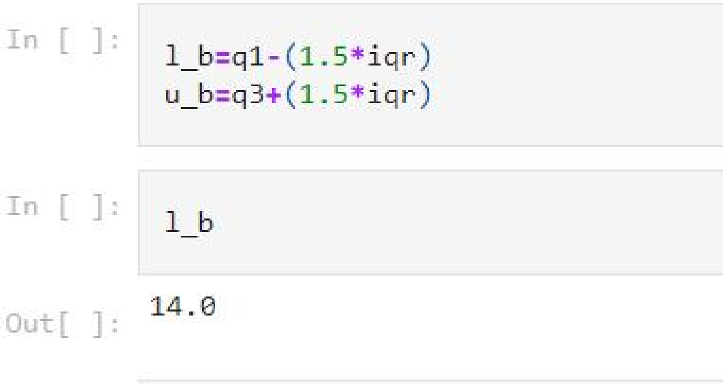
In [ ]:
```
iqr=q3-q1
iqr
```

Out[ ]:
```
12.0
```

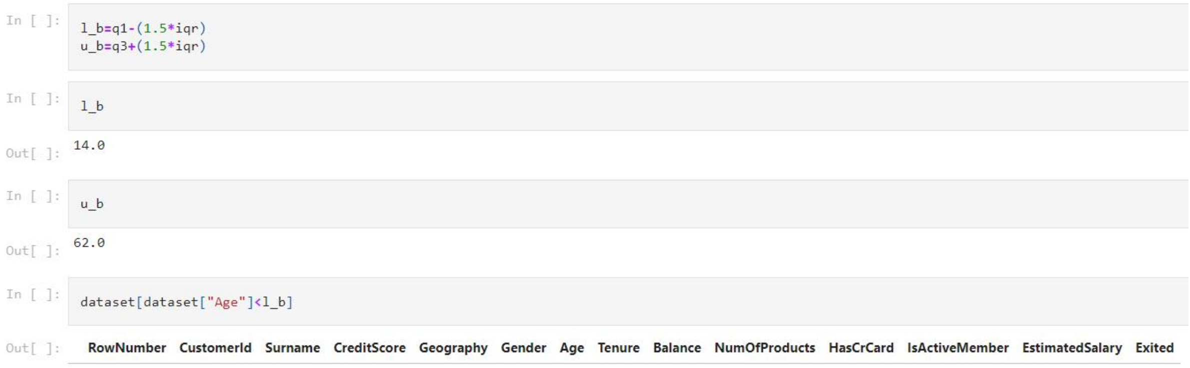
ASSIGNMENT 2



```
In [ ]:  l_b=q1-(1.5*iqr)
         u_b=q3+(1.5*iqr)

In [ ]:  l_b

Out[ ]:  14.0
```

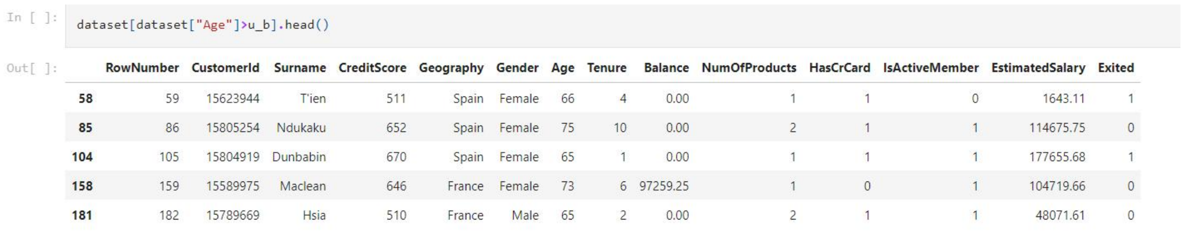


```
In [ ]:  l_b=q1-(1.5*iqr)
         u_b=q3+(1.5*iqr)

In [ ]:  l_b

Out[ ]:  14.0
```



```
In [ ]:  l_b=q1-(1.5*iqr)
         u_b=q3+(1.5*iqr)

In [ ]:  l_b

Out[ ]:  14.0

In [ ]:  u_b

Out[ ]:  62.0

In [ ]:  dataset[dataset["Age"]<l_b]

Out[ ]:
```

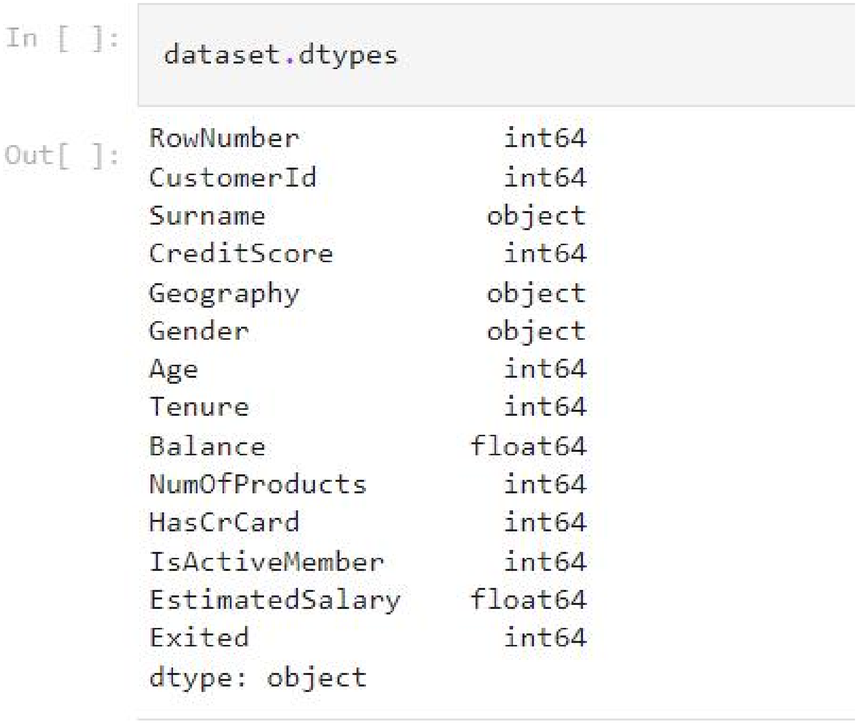
RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited



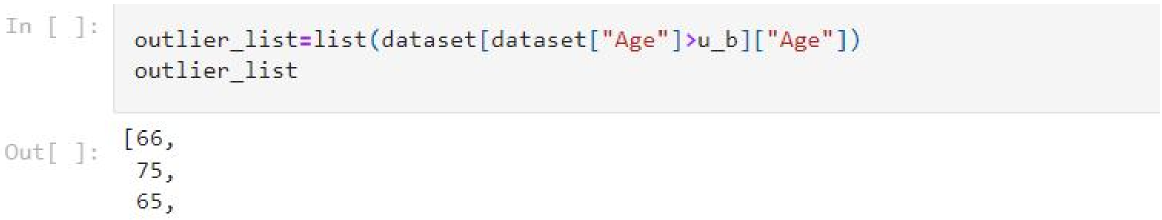
```
In [ ]:  dataset[dataset["Age"]>u_b].head()

Out[ ]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
58	59	15623944	T'ien	511	Spain	Female	66	4	0.00	1	1	0	1643.11	1
85	86	15805254	Ndukaku	652	Spain	Female	75	10	0.00	2	1	1	114675.75	0
104	105	15804919	Dunbabin	670	Spain	Female	65	1	0.00	1	1	1	177655.68	1
158	159	15589975	Maclean	646	France	Female	73	6	97259.25	1	0	1	104719.66	0
181	182	15789669	Hsia	510	France	Male	65	2	0.00	2	1	1	48071.61	0

ASSIGNMENT 2



```
In [ ]:  dataset.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
```



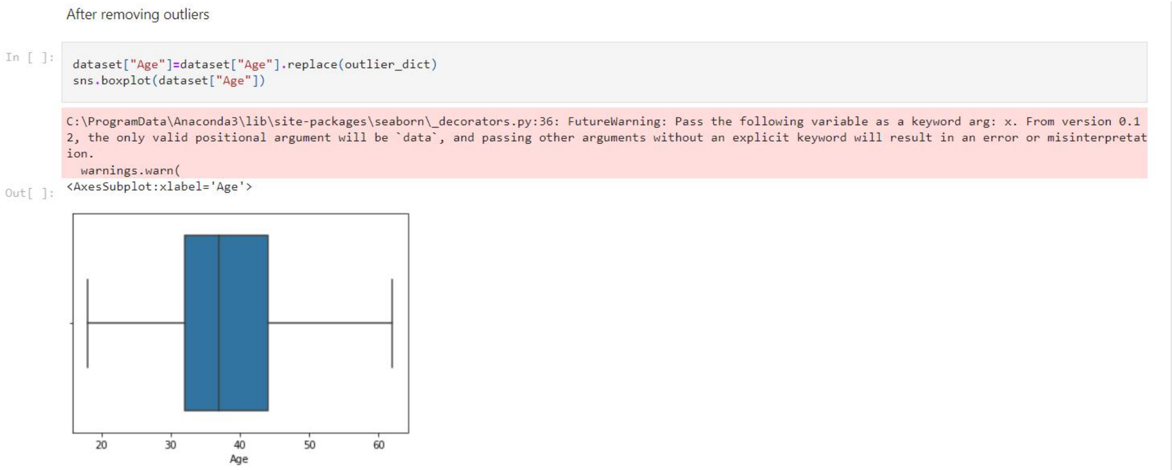
```
In [ ]:  outlier_list=list(dataset[dataset["Age"]>u_b]["Age"])
         outlier_list

Out[ ]:  [66,
          75,
          65,
```

```
65,
 73,
 65,
 72,
 67,
 67,
 79,
 80,
 68,
 75,
 66,
 66,
 70,
 63,
 72,
 64,
 64,
 70,
 67,
 82,
 63,
 69,
 65,
 69,
 64,
 65,
74,
67,
66,
67,
63,
70,
71,
72,
67,
74,
76,
66,
63,
66,
68,
67,
63,
71,
66,
69,
73,
65,
66,
64,
69,
64,
77,
74,
65,
70,
67,
69,
67,
74,
69,
74,
74,
64,
63,
63,
70,
74,
65,
72,
77,
66,
65,
74,
88,
63,
71,
63,
64,
67,
70,
68,
72,
```

ASSIGNMENT 2

```
71,
66,
75,
67,
73,
69,
76,
63,
85,
67,
74,
76,
66,
69,
66,
72,
63,
71,
63,
74,
67,
72,
72,
66,
84,
71,
66,
63,
74,
69,
84,
67,
64,
68,
66,
77,
70,
67,
79,
67,
76,
73,
66,
67,
64,
73,
76,
72,
64,
71,
63,
70,
65,
66,
65,
80,
66,
63,
63,
63,
63,
66,
74,
69,
63,
64,
76,
75,
68,
69,
77,
64,
66,
74,
71,
67,
68,
64,
68,
70,
64,
75,
66,
64,
78,
65,
74,
64,
64,
71,
77,
79,
70,
81,
64,
68,
68,
63,
79,
66,
64,
70,
69,
71,
72,
66,
68,
63,
71,
72,
72,
64,
78,
75,
65,
65,
67,
63,
68,
71,
73,
64,
66,
71,
69,
71,
66,
76,
69,
73,
64,
64,
75,
73,
71,
72,
63,
67,
68,
73,
67,
64,
63,
92,
65,
75,
67,
71,
64,
66,
64,
66,
67,
77,
92,
67,
63,
66,
66,
68,
65,
72,
71,
76,
63,
67,
67,
66,
67,
63,
65,
70,
72,
77,
74,
72,
73,
77,
67,
71,
64,
72,
81,
76,
69,
68,
74,
64,
64,
71,
68,
63,
67,
63,
64,
76,
63,
63,
68,
67,
72,
70,
81,
67,
73,
66,
68,
71,
66,
63,
75,
69,
64,
69,
70,
71,
71,
66,
70,
63,
64,
65,
63,
67,
71,
67,
65,
66,
63,
73,
66,
64,
72,
71,
69,
67,
64,
81,
73,
63,
67,
74,
83,
69,
71,
78,
63,
70,
69,
72,
70,
63,
74,
80,
69,
72,
67,
76,
71,
67,
71,
78,
63,
63,
68,
64,
70,
78,
69,
68,
64,
64,
77,
77]
```

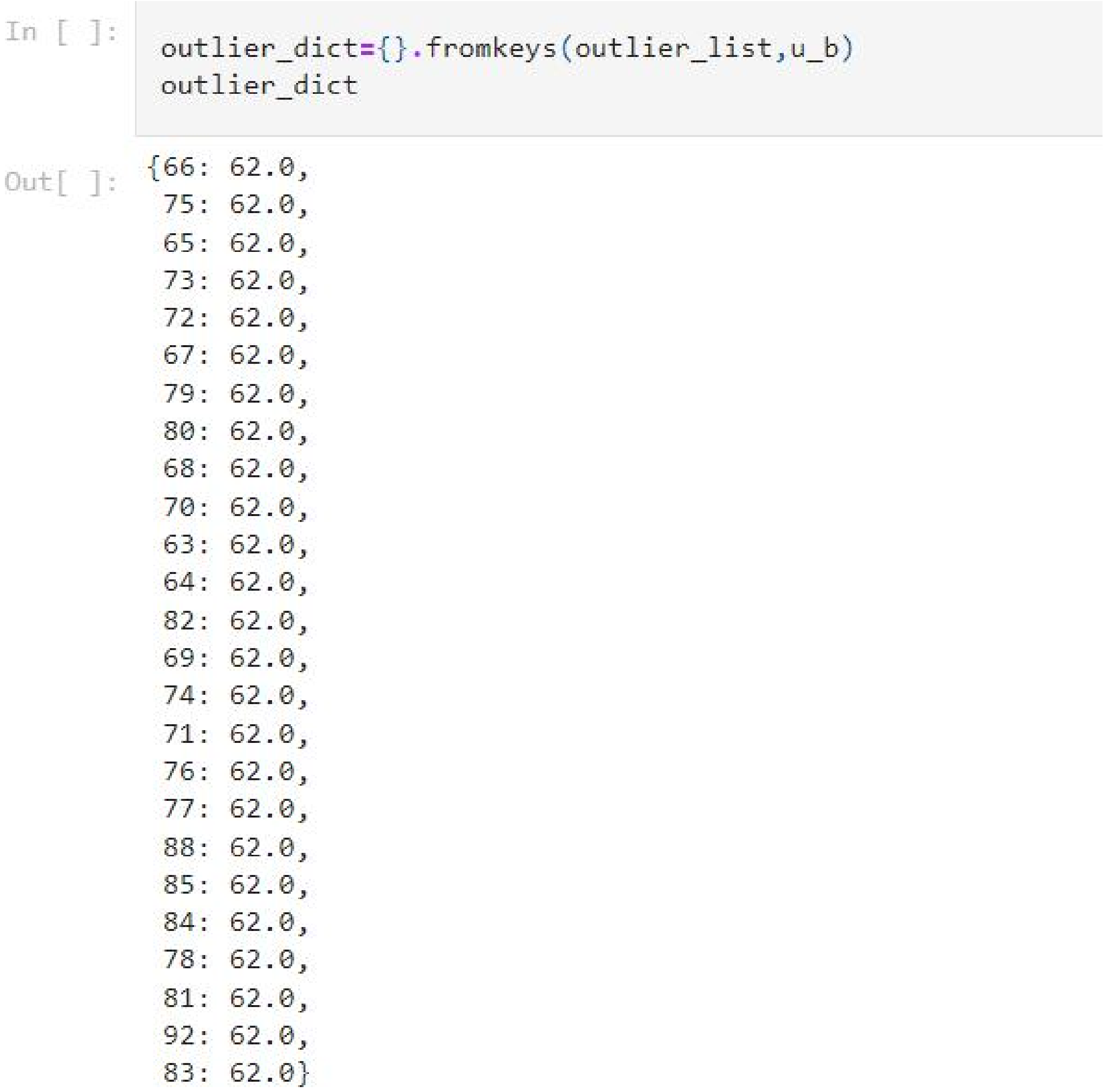
ASSIGNMENT 2



After removing outliers

```
In [ ]: dataset["Age"]=dataset["Age"].replace(outlier_dict)
        sns.boxplot(dataset["Age"])
```

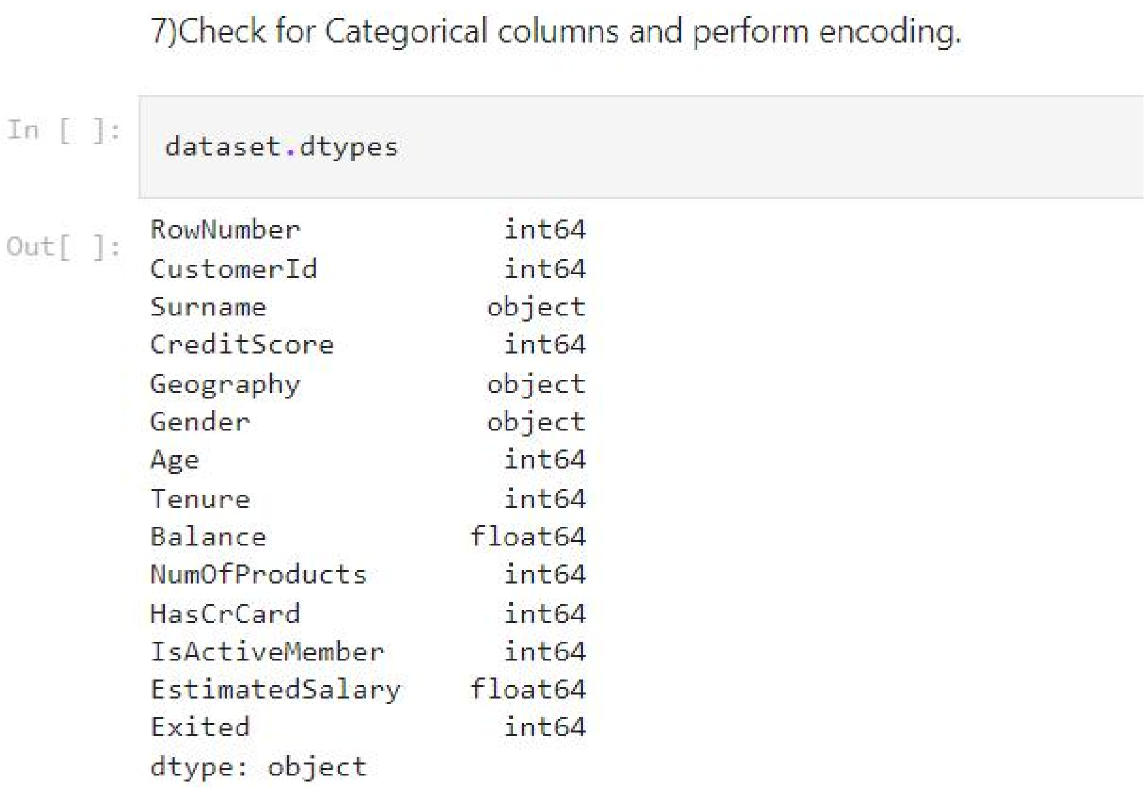
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.1
2, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
Out[ ]: <AxesSubplot:xlabel='Age'>
```



```
In [ ]: outlier_dict={}.fromkeys(outlier_list,u_b)
        outlier_dict
```

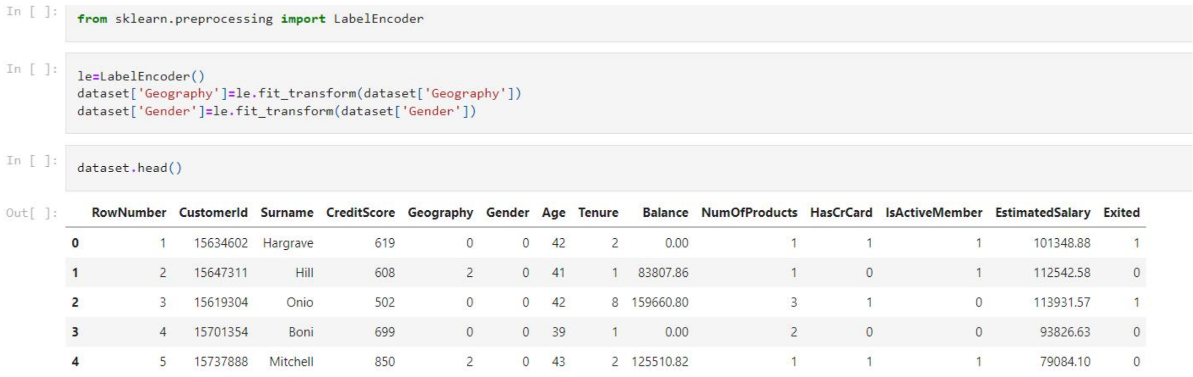
```
Out[ ]: {66: 62.0,
         75: 62.0,
         65: 62.0,
         73: 62.0,
         72: 62.0,
         67: 62.0,
         79: 62.0,
         80: 62.0,
         68: 62.0,
         70: 62.0,
         63: 62.0,
         64: 62.0,
         82: 62.0,
         69: 62.0,
         74: 62.0,
         71: 62.0,
         76: 62.0,
         77: 62.0,
         88: 62.0,
         85: 62.0,
         84: 62.0,
         78: 62.0,
         81: 62.0,
         92: 62.0,
         83: 62.0}
```

ASSIGNMENT 2



7)Check for Categorical columns and perform encoding.

```
In [ ]: dataset.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
```



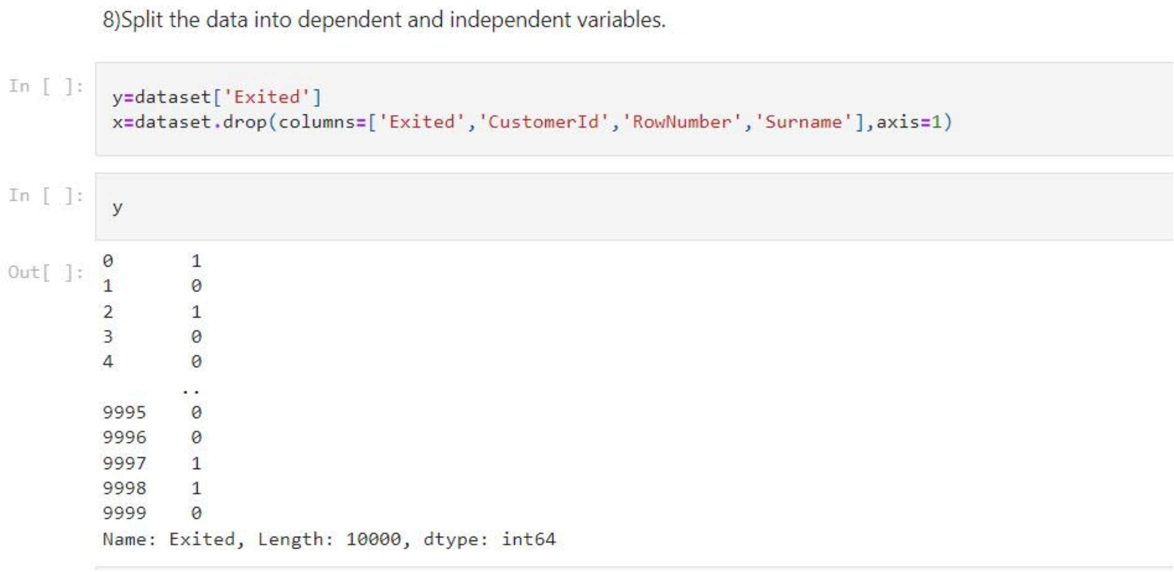
```
In [ ]: from sklearn.preprocessing import LabelEncoder
```

```
In [ ]: le=LabelEncoder()
        dataset['Geography']=le.fit_transform(dataset['Geography'])
        dataset['Gender']=le.fit_transform(dataset['Gender'])
```

```
In [ ]: dataset.head()
```

Out[ ]:

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

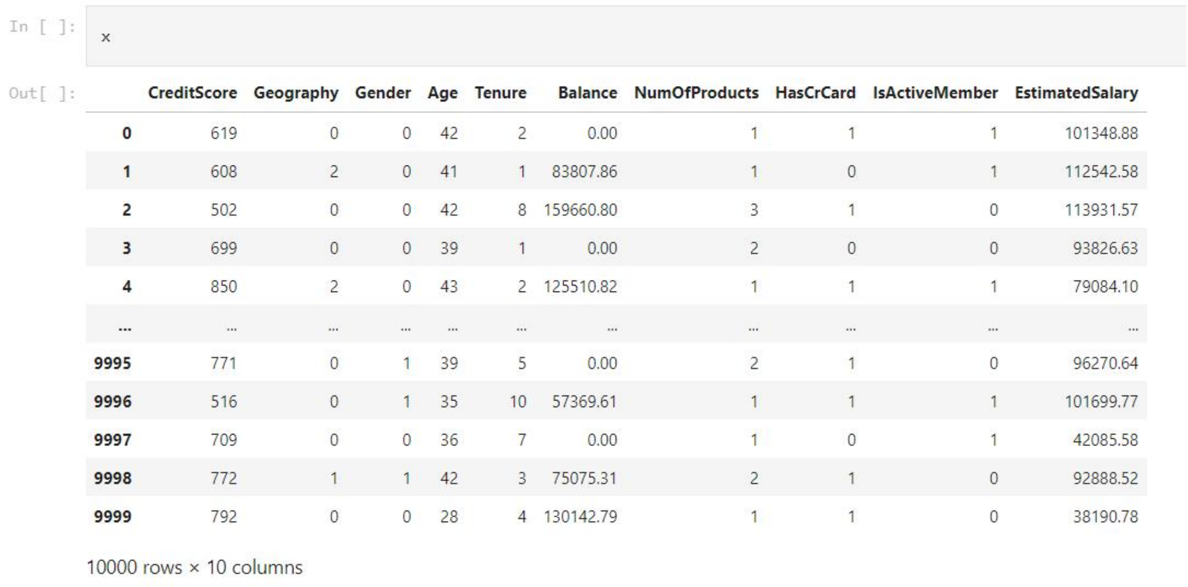


8)Split the data into dependent and independent variables.

```
In [ ]: y=dataset['Exited']
        x=dataset.drop(columns=['Exited','CustomerId','RowNumber','Surname'],axis=1)
```

```
In [ ]: y

Out[ ]: 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
```

ASSIGNMENT 2

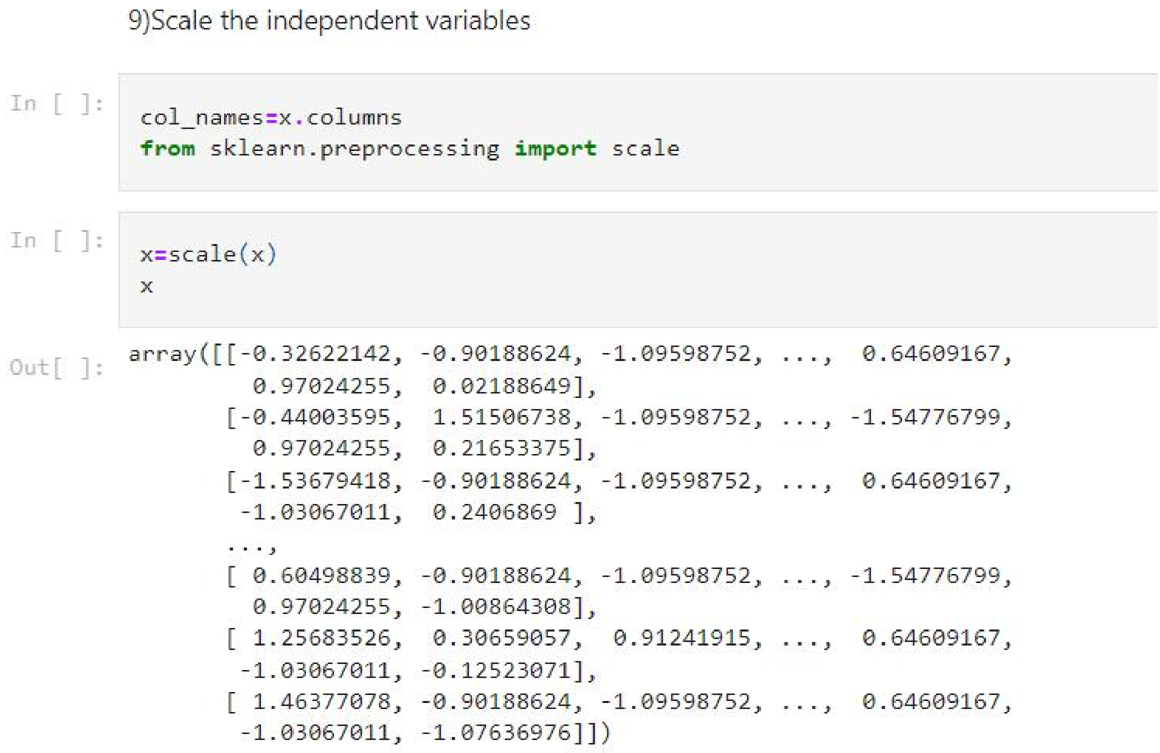


```
In [ ]: x
```

Out[ ]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	0	0	42	2	0.00	1	1	1	101348.88
1	608	2	0	41	1	83807.86	1	0	1	112542.58
2	502	0	0	42	8	159660.80	3	1	0	113931.57
3	699	0	0	39	1	0.00	2	0	0	93826.63
4	850	2	0	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...	...	...
9995	771	0	1	39	5	0.00	2	1	0	96270.64
9996	516	0	1	35	10	57369.61	1	1	1	101699.77
9997	709	0	0	36	7	0.00	1	0	1	42085.58
9998	772	1	1	42	3	75075.31	2	1	0	92888.52
9999	792	0	0	28	4	130142.79	1	1	0	38190.78

10000 rows × 10 columns



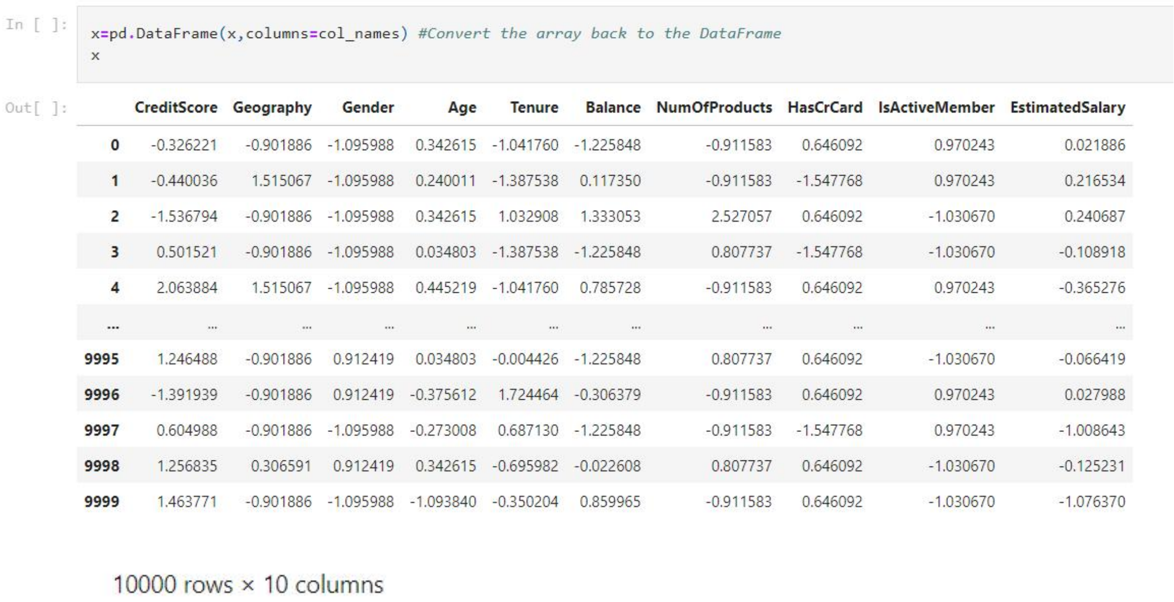
9)Scale the independent variables

```
In [ ]: col_names=x.columns
        from sklearn.preprocessing import scale
```

```
In [ ]: x=scale(x)
        x
```

```
Out[ ]: array([[-0.32622142, -0.90188624, -1.09598752, ...,  0.64609167,
                 0.97024255,  0.02188649],
               [-0.44003595,  1.51506738, -1.09598752, ..., -1.54776799,
                 0.97024255,  0.21653375],
               [-1.53679418, -0.90188624, -1.09598752, ...,  0.64609167,
                -1.03067011,  0.2406869 ],
               ...,
               [ 0.60498839, -0.90188624, -1.09598752, ..., -1.54776799,
                 0.97024255, -1.00864308],
               [ 1.25683526,  0.30659057,  0.91241915, ...,  0.64609167,
                -1.03067011, -0.12523071],
               [ 1.46377078, -0.90188624, -1.09598752, ...,  0.64609167,
                -1.03067011, -1.07636976]])
```

ASSIGNMENT 2



In [ ]:

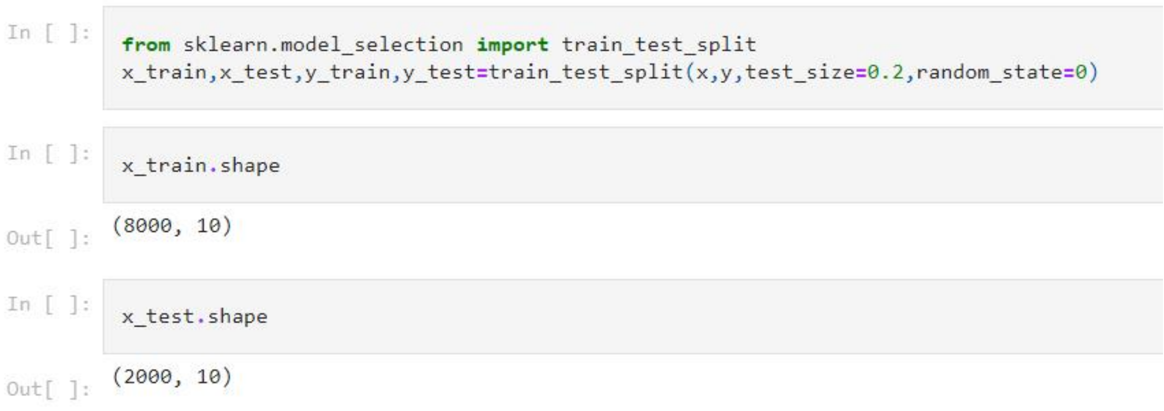
```
x=pd.DataFrame(x,columns=col_names) #Convert the array back to the DataFrame
x
```

Out[ ]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	-0.326221	-0.901886	-1.095988	0.342615	-1.041760	-1.225848	-0.911583	0.646092	0.970243	0.021886
1	-0.440036	1.515067	-1.095988	0.240011	-1.387538	0.117350	-0.911583	-1.547768	0.970243	0.216534
2	-1.536794	-0.901886	-1.095988	0.342615	1.032908	1.333053	2.527057	0.646092	-1.030670	0.240687
3	0.501521	-0.901886	-1.095988	0.034803	-1.387538	-1.225848	0.807737	-1.547768	-1.030670	-0.108918
4	2.063884	1.515067	-1.095988	0.445219	-1.041760	0.785728	-0.911583	0.646092	0.970243	-0.365276
...	...	...	...	...	...	...	...	...	...	...
9995	1.246488	-0.901886	0.912419	0.034803	-0.004426	-1.225848	0.807737	0.646092	-1.030670	-0.066419
9996	-1.391939	-0.901886	0.912419	-0.375612	1.724464	-0.306379	-0.911583	0.646092	0.970243	0.027988
9997	0.604988	-0.901886	-1.095988	-0.273008	0.687130	-1.225848	-0.911583	-1.547768	0.970243	-1.008643
9998	1.256835	0.306591	0.912419	0.342615	-0.695982	-0.022608	0.807737	0.646092	-1.030670	-0.125231
9999	1.463771	-0.901886	-1.095988	-1.093840	-0.350204	0.859965	-0.911583	0.646092	-1.030670	-1.076370

10000 rows × 10 columns

10)Split the data into training and testing



In [ ]:

```
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)
```

In [ ]:

```
x_train.shape
```

Out[ ]:

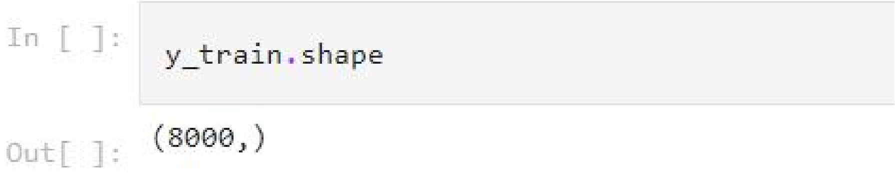
```
(8000, 10)
```

In [ ]:

```
x_test.shape
```

Out[ ]:

```
(2000, 10)
```

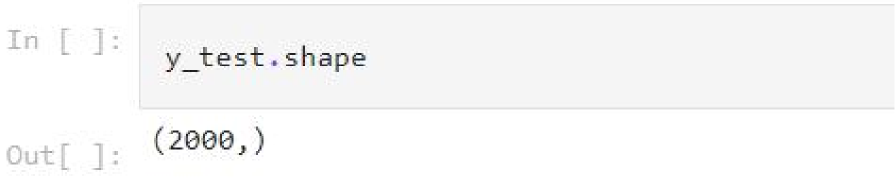


In [ ]:

```
y_train.shape
```

Out[ ]:

```
(8000,)
```



In [ ]:

```
y_test.shape
```

Out[ ]:

```
(2000,)
```

ASSIGNMENT 2