EXCEL COLLEGE OF ENGINEERING(AUTONOMOUS)

NH-544, Salem Main Road, Sankari West, Pallakkapalayam, Pin:637 303. Komarapalayam
Namakkal Dt. Tamilnadu.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING.

WEB PHISHING DETECTION (**ASSIGNMENT 2)**

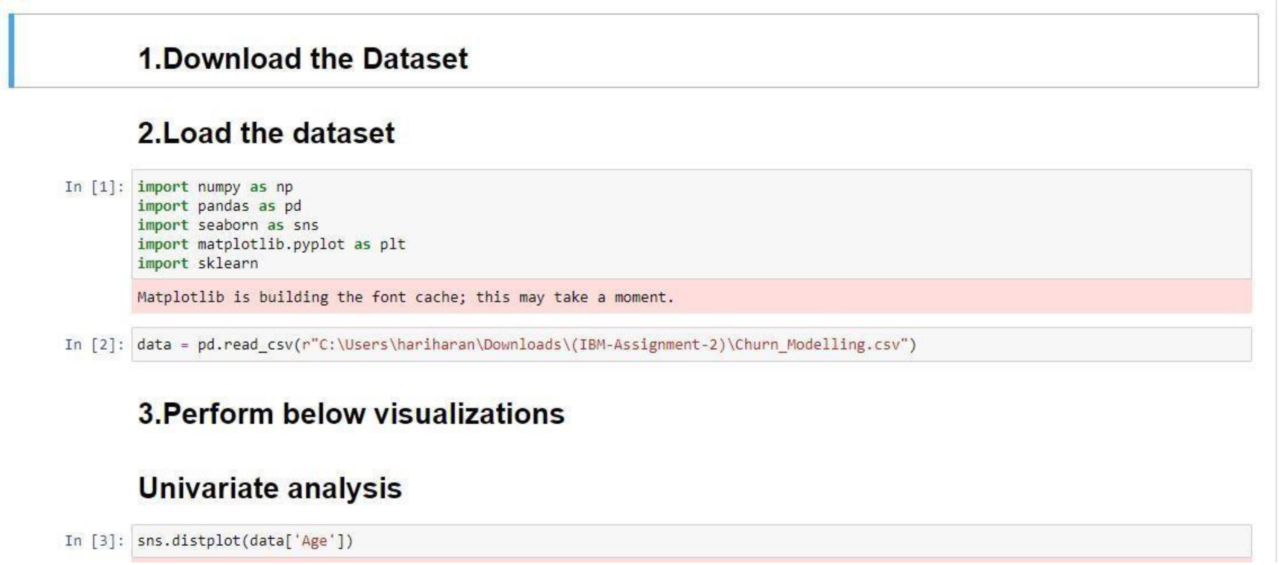
DATE : 26-09-2022

PROBLEM : PERFORM TASKS ACCORDINGLY

NAME : Abdul Quadir

OUTPUT :

SCREENSHOTS :



1.Download the Dataset

2.Load the dataset

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import sklearn

        Matplotlib is building the font cache; this may take a moment.

In [2]: data = pd.read_csv(r"C:\Users\hariharan\Downloads\(IBM-Assignment-2)\Churn_Modelling.csv")
```

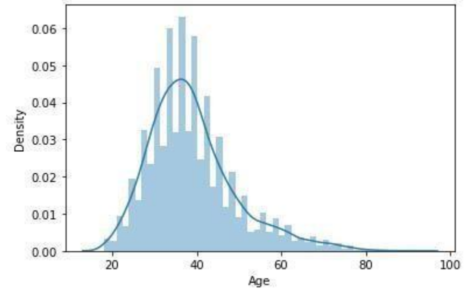
3.Perform below visualizations

Univariate analysis

```
In [3]: sns.distplot(data['Age'])
```

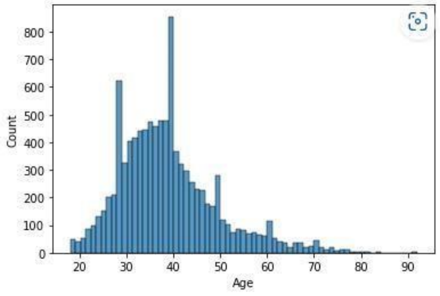
```
D:\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be re
moved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[3]: <AxesSubplot:xlabel='Age', ylabel='Density'>
```



```
In [4]: sns.histplot(data['Age'])
```

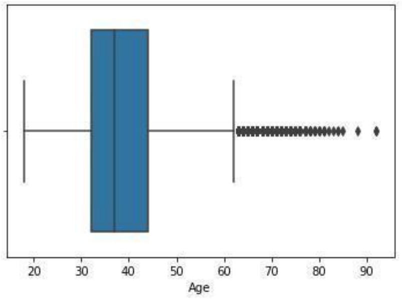
```
Out[4]: <AxesSubplot:xlabel='Age', ylabel='Count'>
```



```
In [5]: sns.boxplot(data['Age'])
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: 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 r
esult in an error or misinterpretation.
  warnings.warn(
```

```
Out[5]: <AxesSubplot:xlabel='Age'>
```

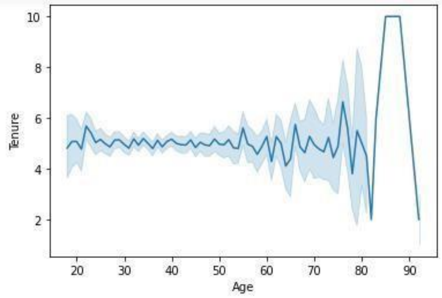


Bi-Variate Analysis

```
In [6]: sns.lineplot(data['Age'],data['Tenure'])
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fr
om version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

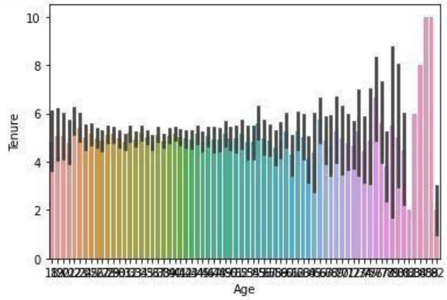
```
Out[6]: <AxesSubplot:xlabel='Age', ylabel='Tenure'>
```



```
In [7]: sns.barplot(data['Age'],data['Tenure'])
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fr
om version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

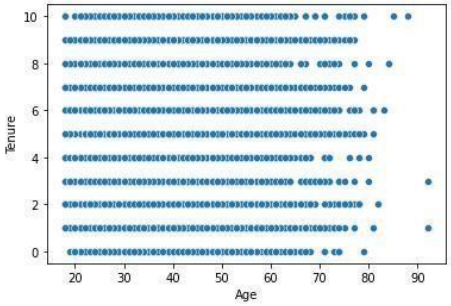
```
Out[7]: <AxesSubplot:xlabel='Age', ylabel='Tenure'>
```



```
In [8]: sns.scatterplot(data['Age'],data['Tenure'])
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fr
om version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

```
Out[8]: <AxesSubplot:xlabel='Age', ylabel='Tenure'>
```

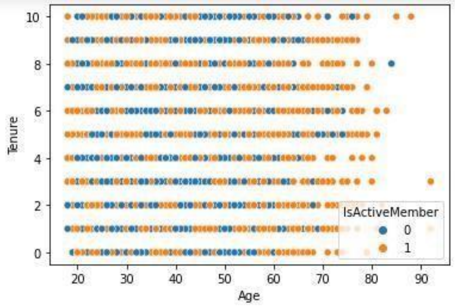


Multi-Variate Analysis

```
In [9]: sns.scatterplot(data['Age'],data['Tenure'], hue=data['IsActiveMember'])
```

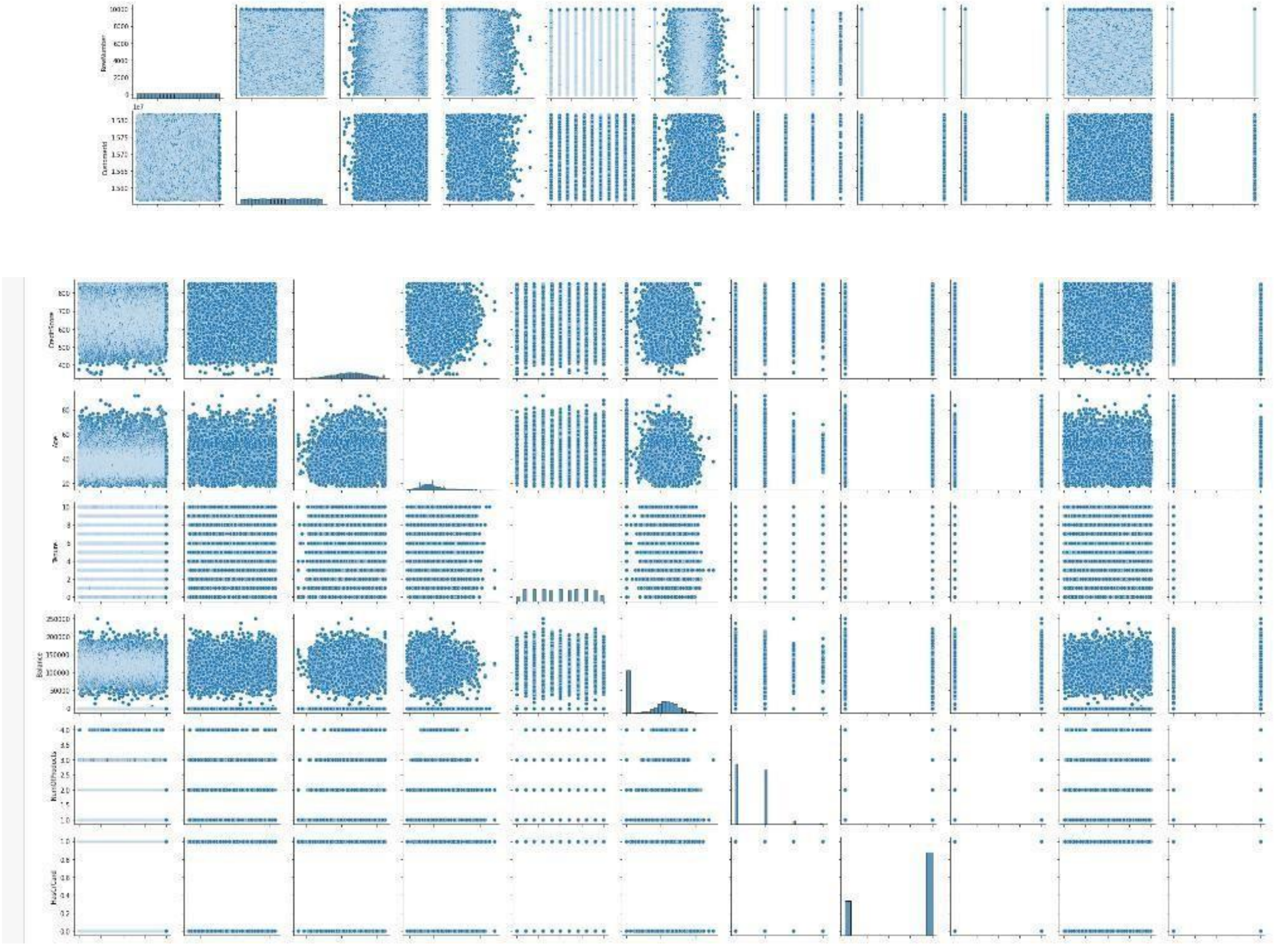
```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fr
om version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

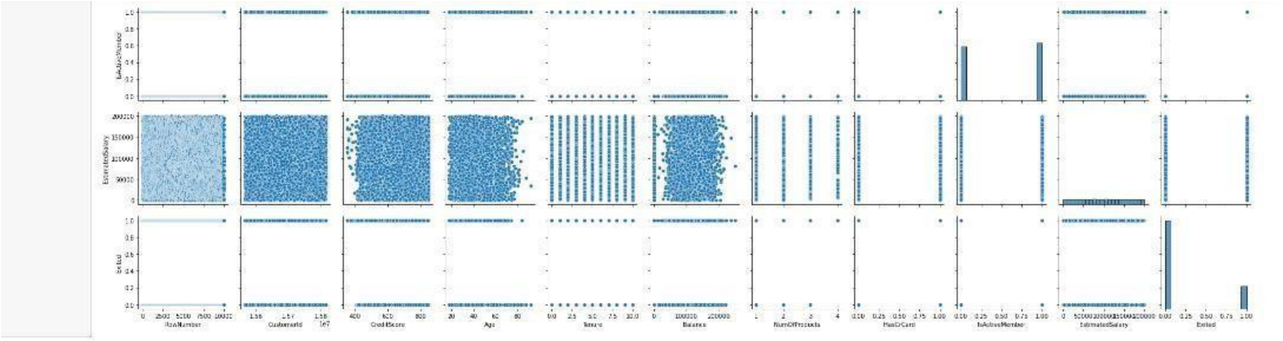
```
Out[9]: <AxesSubplot:xlabel='Age', ylabel='Tenure'>
```

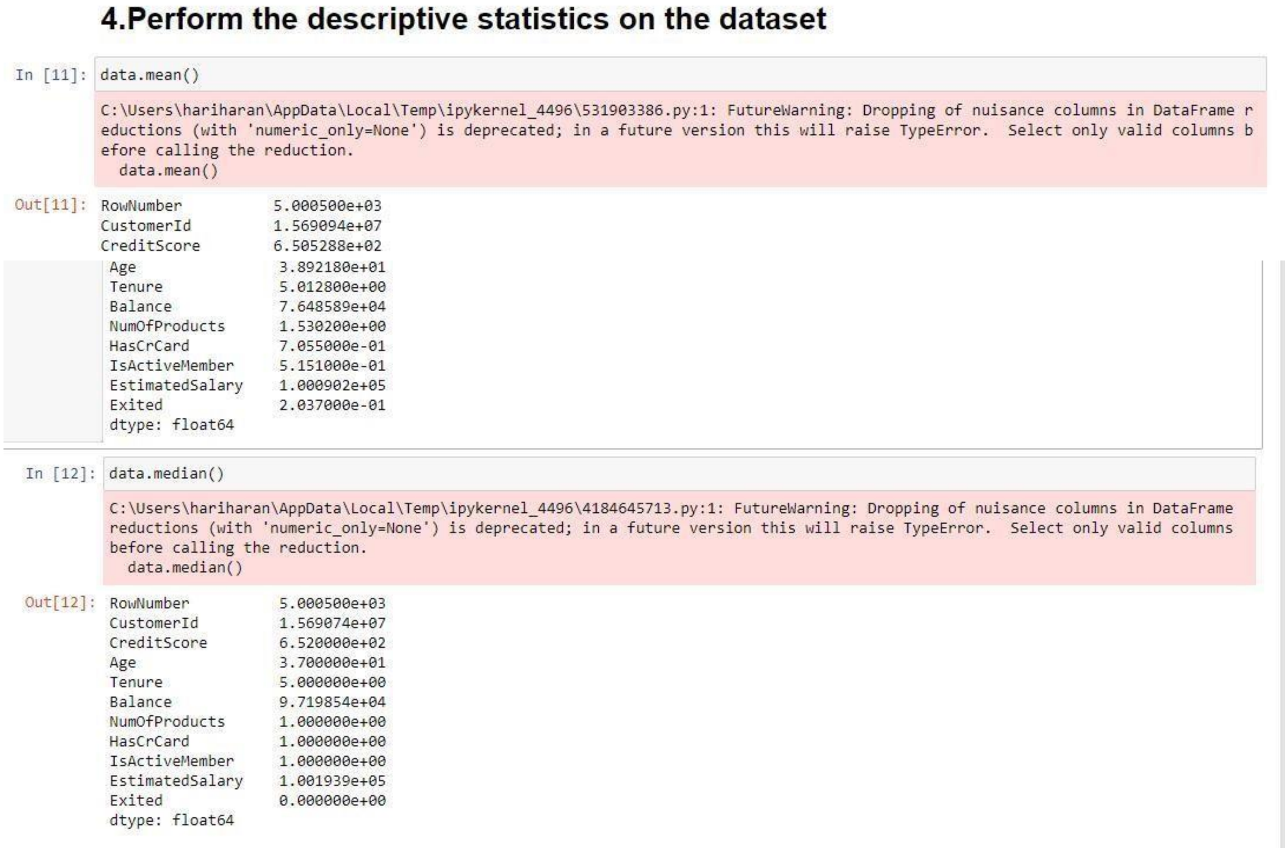


```
In [10]: sns.pairplot(data)

Out[10]: <seaborn.axisgrid.PairGrid at 0x213ae3beee0>
```







4.Perform the descriptive statistics on the dataset

In [11]:
```
data.mean()
```

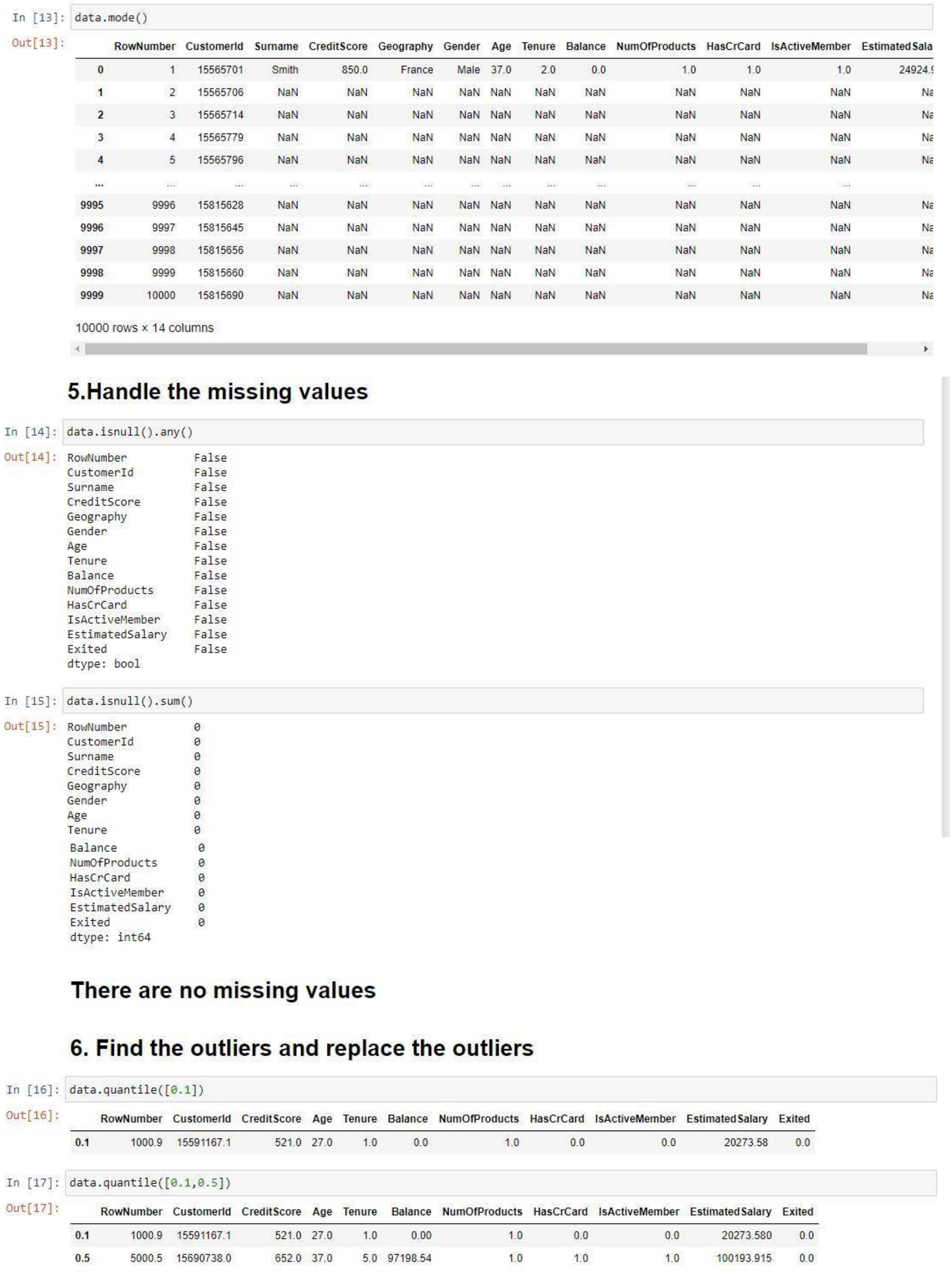
```
C:\Users\hariharan\AppData\Local\Temp\ipykernel_4496\531903386.py:1: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  data.mean()
```

Out[11]:
```
RowNumber          5.000500e+03
CustomerId         1.569094e+07
CreditScore        6.505288e+02
Age                3.892180e+01
Tenure             5.012800e+00
Balance            7.648589e+04
NumOfProducts      1.530200e+00
HasCrCard          7.055000e-01
IsActiveMember     5.151000e-01
EstimatedSalary    1.000902e+05
Exited             2.037000e-01
dtype: float64
```

In [12]:
```
data.median()
```

```
C:\Users\hariharan\AppData\Local\Temp\ipykernel_4496\4184645713.py:1: FutureWarning: Dropping of nuisance columns in DataFrame
reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  data.median()
```

Out[12]:
```
RowNumber          5.000500e+03
CustomerId         1.569074e+07
CreditScore        6.520000e+02
Age                3.700000e+01
Tenure             5.000000e+00
Balance            9.719854e+04
NumOfProducts      1.000000e+00
HasCrCard          1.000000e+00
IsActiveMember     1.000000e+00
EstimatedSalary    1.001939e+05
Exited             0.000000e+00
dtype: float64
```



```
In [13]: data.mode()
```

Out[13]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSala
0	1	15565701	Smith	850.0	France	Male	37.0	2.0	0.0	1.0	1.0	1.0	24924.!
1	2	15565706	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	3	15565714	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	4	15565779	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	5	15565796	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15815628	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9996	9997	15815645	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9997	9998	15815656	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9998	9999	15815660	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9999	10000	15815690	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

10000 rows × 14 columns

5.Handle the missing values

```
In [14]: data.isnull().any()
```

```
Out[14]: 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 [15]: data.isnull().sum()
```

```
Out[15]: 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
```

There are no missing values

6. Find the outliers and replace the outliers

```
In [16]: data.quantile([0.1])
```

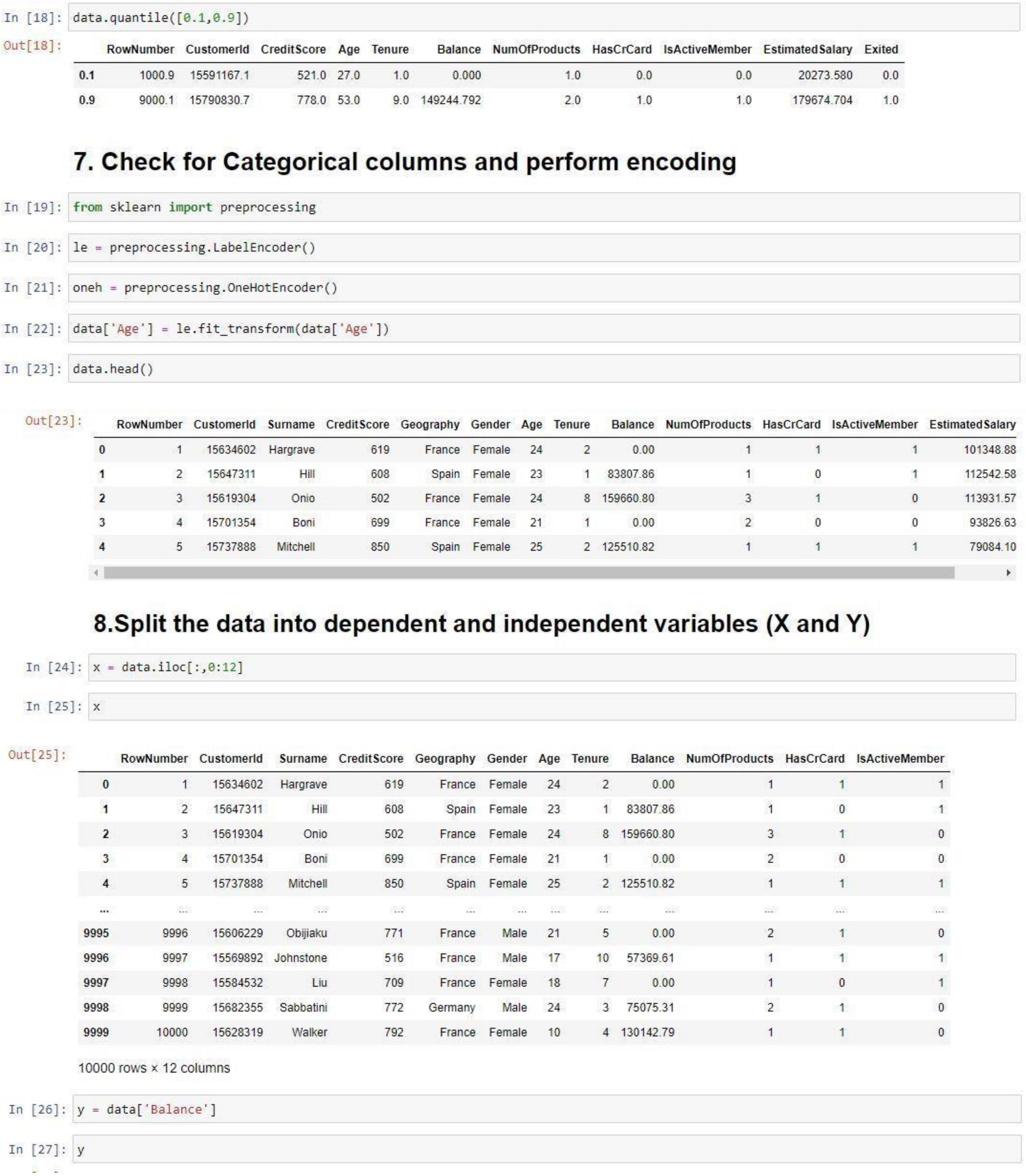
Out[16]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.1	1000.9	15591167.1	521.0	27.0	1.0	0.0	1.0	0.0	0.0	20273.58	0.0

```
In [17]: data.quantile([0.1,0.5])
```

Out[17]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.1	1000.9	15591167.1	521.0	27.0	1.0	0.00	1.0	0.0	0.0	20273.580	0.0
0.5	5000.5	15690738.0	652.0	37.0	5.0	97198.54	1.0	1.0	1.0	100193.915	0.0



```
In [18]: data.quantile([0.1,0.9])
```

Out[18]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.1	1000.9	15591167.1	521.0	27.0	1.0	0.000	1.0	0.0	0.0	20273.580	0.0
0.9	9000.1	15790830.7	778.0	53.0	9.0	149244.792	2.0	1.0	1.0	179674.704	1.0

7. Check for Categorical columns and perform encoding

```
In [19]: from sklearn import preprocessing
```

```
In [20]: le = preprocessing.LabelEncoder()
```

```
In [21]: oneh = preprocessing.OneHotEncoder()
```

```
In [22]: data['Age'] = le.fit_transform(data['Age'])
```

```
In [23]: data.head()
```

Out[23]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	24	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	23	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	24	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	21	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	25	2	125510.82	1	1	1	79084.10

8.Split the data into dependent and independent variables (X and Y)

```
In [24]: x = data.iloc[:,0:12]
```

```
In [25]: x
```

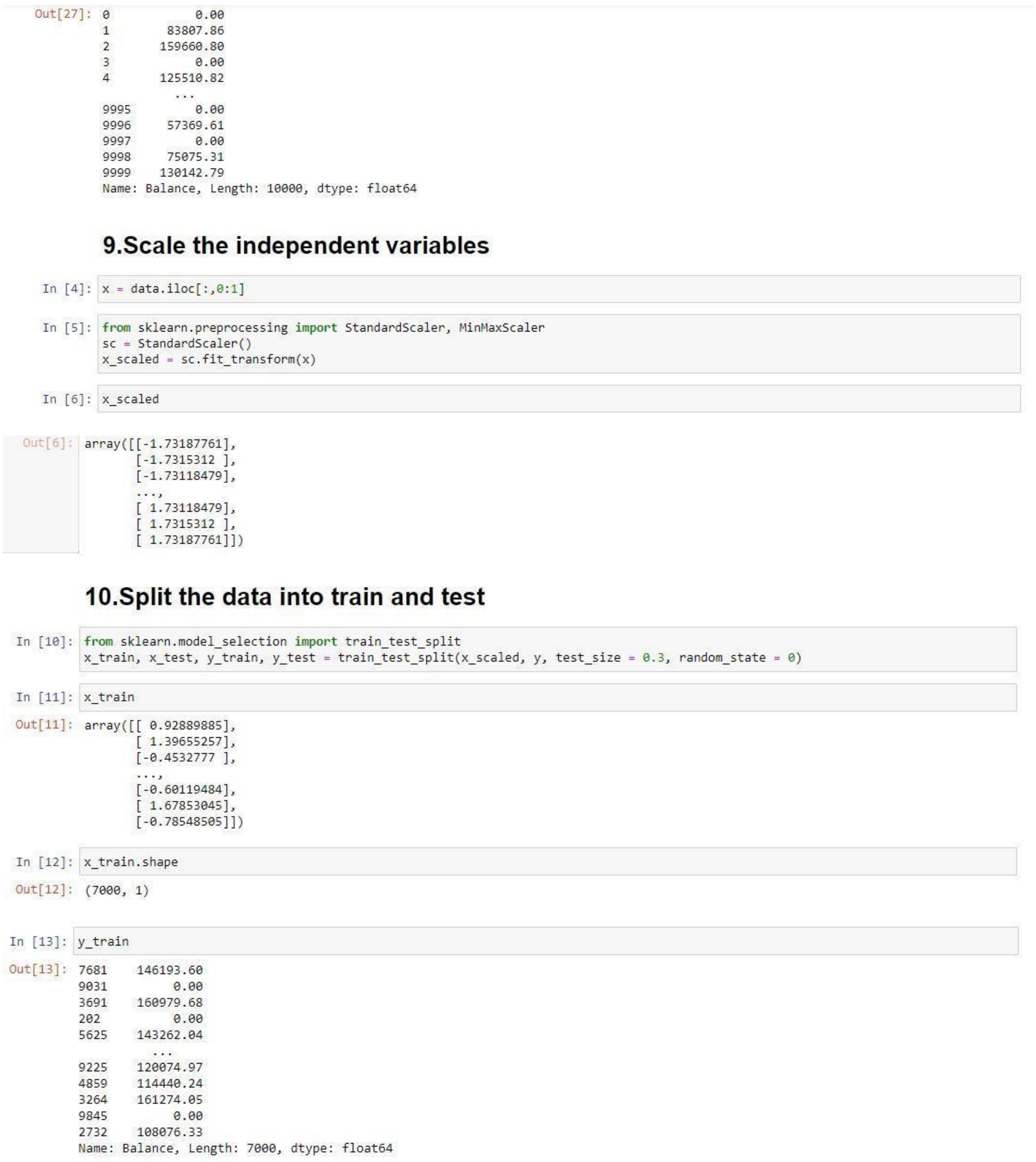
Out[25]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619	France	Female	24	2	0.00	1	1	1
1	2	15647311	Hill	608	Spain	Female	23	1	83807.86	1	0	1
2	3	15619304	Onio	502	France	Female	24	8	159660.80	3	1	0
3	4	15701354	Boni	699	France	Female	21	1	0.00	2	0	0
4	5	15737888	Mitchell	850	Spain	Female	25	2	125510.82	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	21	5	0.00	2	1	0
9996	9997	15569892	Johnstone	516	France	Male	17	10	57369.61	1	1	1
9997	9998	15584532	Liu	709	France	Female	18	7	0.00	1	0	1
9998	9999	15682355	Sabbatini	772	Germany	Male	24	3	75075.31	2	1	0
9999	10000	15628319	Walker	792	France	Female	10	4	130142.79	1	1	0

10000 rows × 12 columns

```
In [26]: y = data['Balance']
```

```
In [27]: y
```



```
Out[27]: 0            0.00
         1        83807.86
         2       159660.80
         3            0.00
         4       125510.82
                   ...
         9995         0.00
         9996     57369.61
         9997         0.00
         9998     75075.31
         9999    130142.79
         Name: Balance, Length: 10000, dtype: float64
```

9.Scale the independent variables

```
In [4]: x = data.iloc[:,0:1]
```

```
In [5]: from sklearn.preprocessing import StandardScaler, MinMaxScaler
        sc = StandardScaler()
        x_scaled = sc.fit_transform(x)
```

```
In [6]: x_scaled
```

```
Out[6]: array([[-1.73187761],
               [-1.7315312 ],
               [-1.73118479],
               ...,
               [ 1.73118479],
               [ 1.7315312 ],
               [ 1.73187761]])
```

10.Split the data into train and test

```
In [10]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.3, random_state = 0)
```

```
In [11]: x_train
```

```
Out[11]: array([[ 0.92889885],
                [ 1.39655257],
                [-0.4532777 ],
                ...,
                [-0.60119484],
                [ 1.67853045],
                [-0.78548505]])
```

```
In [12]: x_train.shape
```

```
Out[12]: (7000, 1)
```

```
In [13]: y_train
```

```
Out[13]: 7681    146193.60
         9031         0.00
         3691    160979.68
         202          0.00
         5625    143262.04
                   ...
         9225    120074.97
         4859    114440.24
         3264    161274.05
         9845         0.00
         2732    108076.33
         Name: Balance, Length: 7000, dtype: float64
```

*****************************THANKING YOU**********************************