

Assignment -2

Data Visualization and Preprocessing

| | |
|---------------------|-------------------|
| Assignment Date | 19 September 2022 |
| Student Name | ABINAYA |
| Student Roll Number | 211419205003 |
| Maximum Marks | 2 Marks |

Question-1:

Download the dataset:

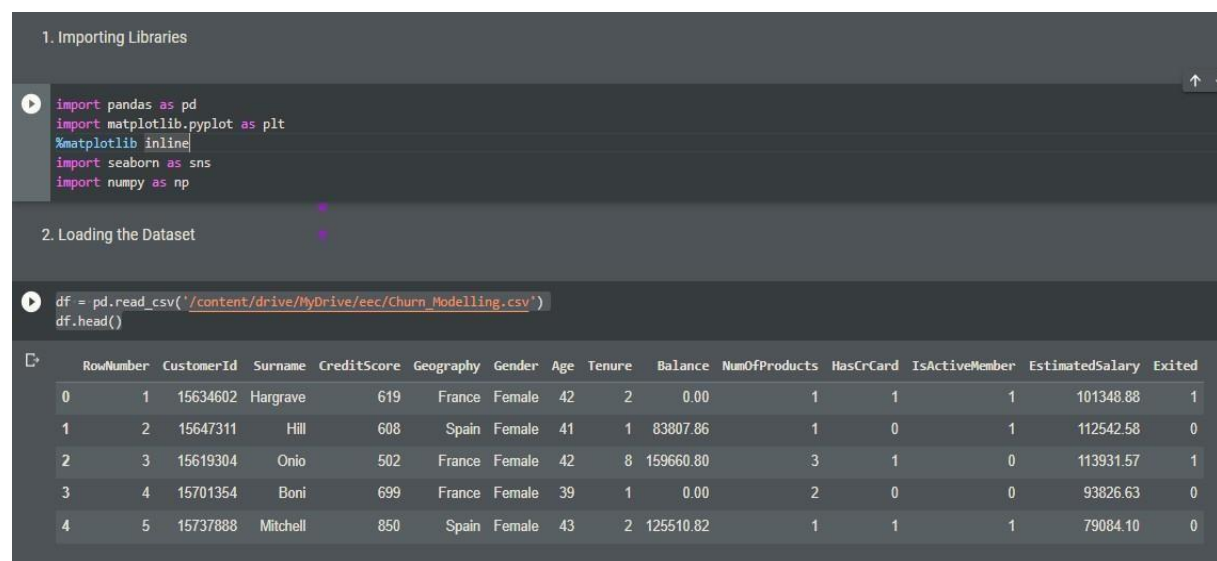
Question-2:

Load the dataset.

Solution:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import numpy as np

df = pd.read_csv('/content/drive/MyDrive/eec/Churn_Modelling.csv')
df.head()
```



```
1. Importing Libraries

import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import numpy as np

2. Loading the Dataset

df = pd.read_csv('/content/drive/MyDrive/eec/Churn_Modelling.csv')
df.head()
```

| | 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 |

Question-3:

Perform Below Visualizations.

1)Univariate Analysis

Solution:

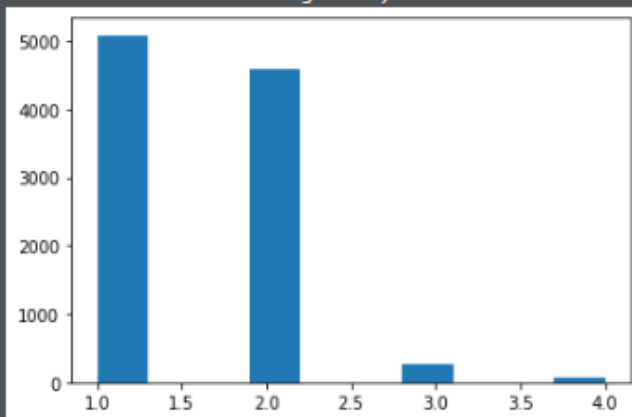
```
plt.hist(df['NumOfProducts'])
```

Univariate Analysis



```
plt.hist(df['NumOfProducts'])
```

```
(array([5084., 0., 0., 4590., 0., 0., 266., 0., 0.,  
        60.]),  
 array([1., 1.3, 1.6, 1.9, 2.2, 2.5, 2.8, 3.1, 3.4, 3.7, 4. ]),  
 <a list of 10 Patch objects>)
```



2)Bi - Variate Analysis

Solution:

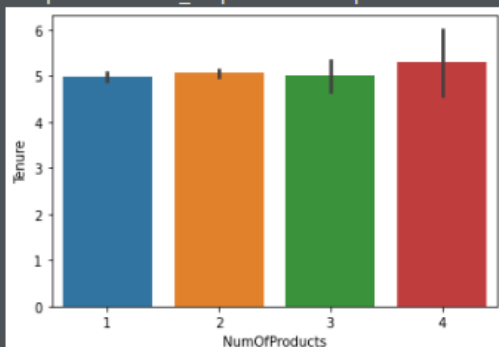
```
sns.barplot(x=df.NumOfProducts,y=df.Tenure)
```

Bivariate Analysis



```
sns.barplot(x=df.NumOfProducts,y=df.Tenure)
```

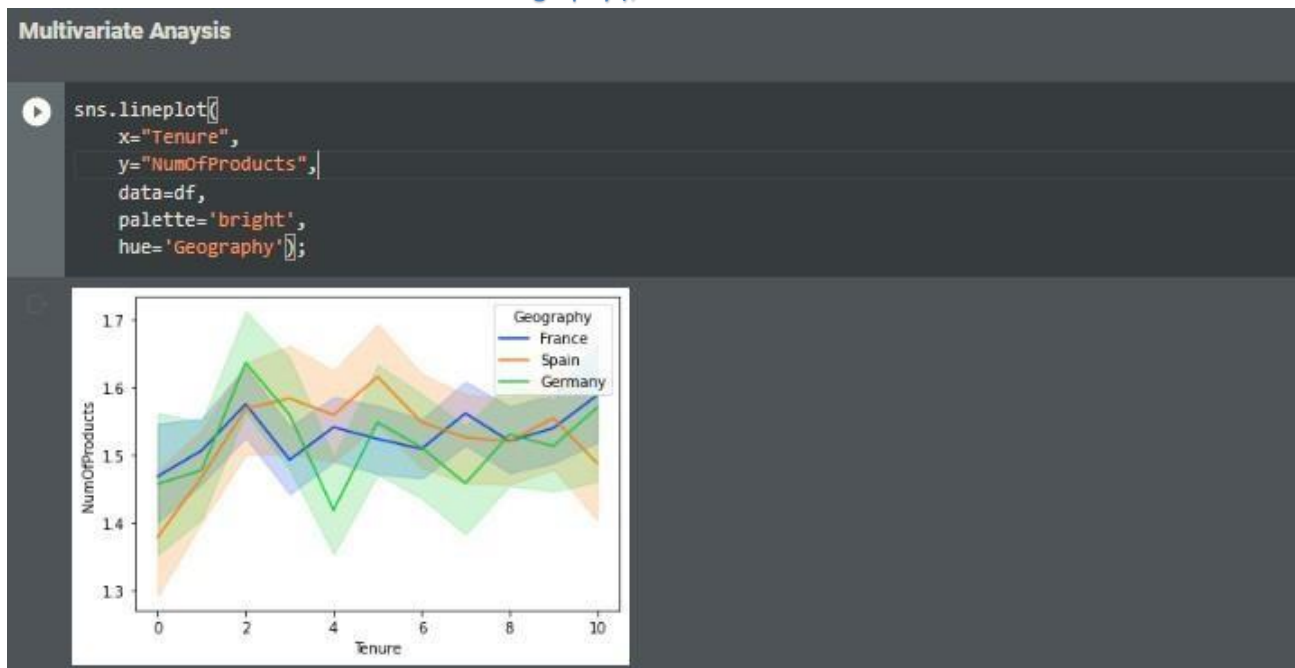
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4e1147f110>
```



1)Multivariate Analysis

Solution:

```
sns.lineplot(  
    x="Tenure",  
    y="NumOfProducts",  
    data=df,  
    palette='bright',  
    hue='Geography');
```



Question-4:

Perform descriptive statistics on the dataset.

Solution:

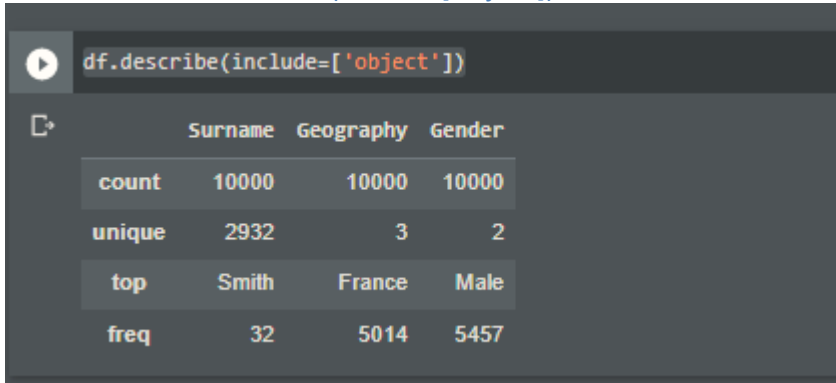
```
df.describe()
```

4. Descriptive Statistics

```
df.describe()
```

| | 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 |

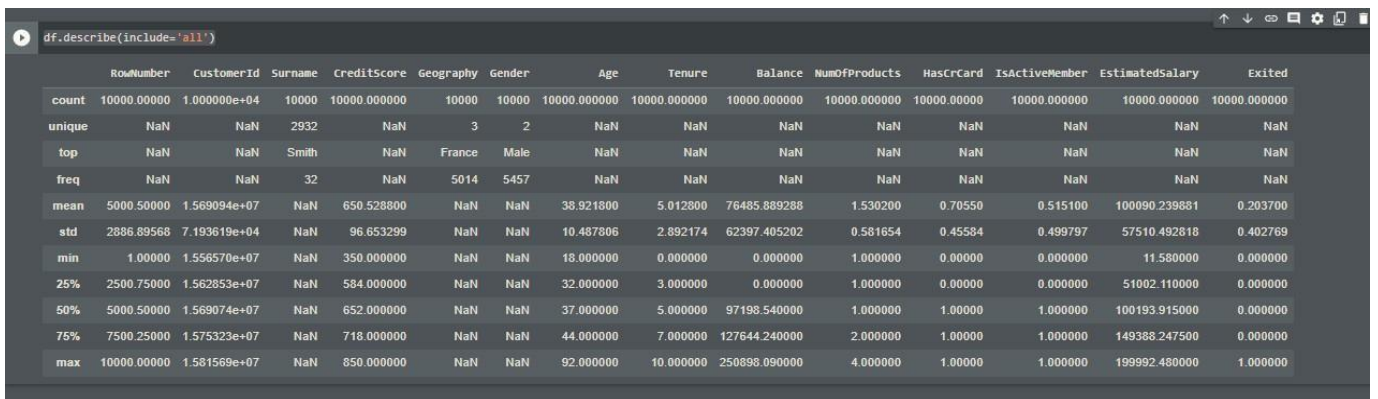
```
df.describe(include=['object'])
```



A Jupyter Notebook interface showing a code cell with the command `df.describe(include=['object'])` and its output. The output is a summary table for object data types.

| | Surname | Geography | Gender |
|--------|---------|-----------|--------|
| count | 10000 | 10000 | 10000 |
| unique | 2932 | 3 | 2 |
| top | Smith | France | Male |
| freq | 32 | 5014 | 5457 |

```
df.describe(include='all')
```



A Jupyter Notebook interface showing a code cell with the command `df.describe(include='all')` and its output. The output is a comprehensive summary table for all data types in the DataFrame.

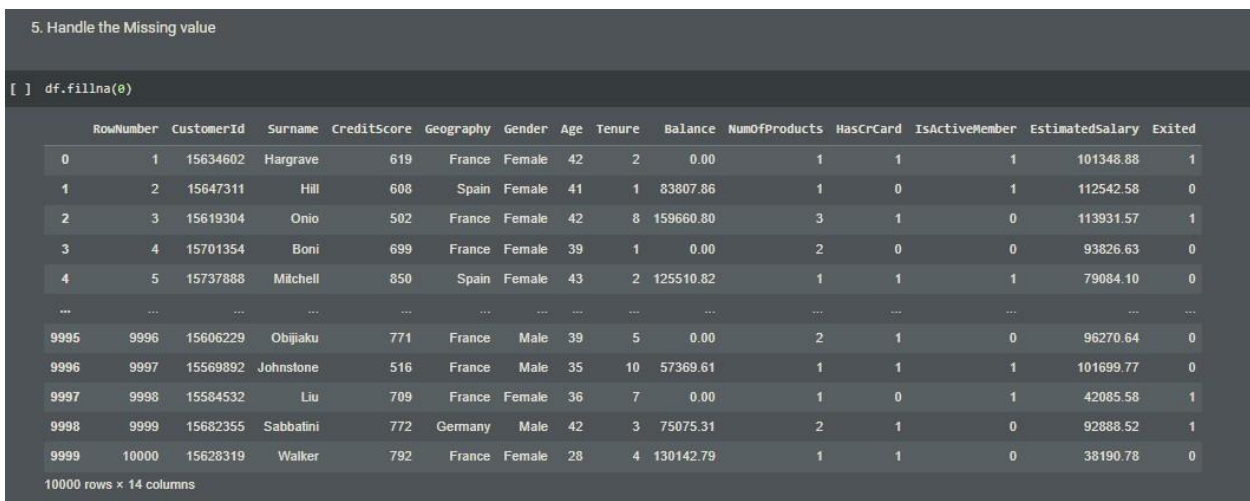
| | RowNumber | CustomerId | Surname | Creditscore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | Exited |
|--------|-------------|--------------|---------|--------------|-----------|--------|--------------|--------------|---------------|---------------|--------------|----------------|-----------------|--------------|
| count | 10000.00000 | 1.000000e+04 | 10000 | 10000.000000 | 10000 | 10000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 |
| unique | NaN | NaN | 2932 | NaN | 3 | 2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| top | NaN | NaN | Smith | NaN | France | Male | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| freq | NaN | NaN | 32 | NaN | 5014 | 5457 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| mean | 5000.50000 | 1.569094e+07 | NaN | 650.528800 | NaN | NaN | 38.921800 | 5.012800 | 76485.889288 | 1.530200 | 0.70550 | 0.515100 | 100090.239881 | 0.203700 |
| std | 2886.89568 | 7.193619e+04 | NaN | 96.653299 | NaN | NaN | 10.487806 | 2.892174 | 62397.405202 | 0.581654 | 0.45584 | 0.499797 | 57510.492818 | 0.402769 |
| min | 1.00000 | 1.556570e+07 | NaN | 350.000000 | NaN | NaN | 18.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 11580000 | 0.000000 |
| 25% | 2500.75000 | 1.562853e+07 | NaN | 584.000000 | NaN | NaN | 32.000000 | 3.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 51002.110000 | 0.000000 |
| 50% | 5000.50000 | 1.569074e+07 | NaN | 652.000000 | NaN | NaN | 37.000000 | 5.000000 | 97198.540000 | 1.000000 | 1.000000 | 1.000000 | 100193.915000 | 0.000000 |
| 75% | 7500.25000 | 1.575323e+07 | NaN | 718.000000 | NaN | NaN | 44.000000 | 7.000000 | 127644.240000 | 2.000000 | 1.000000 | 1.000000 | 149388.247500 | 0.000000 |
| max | 10000.00000 | 1.581569e+07 | NaN | 850.000000 | NaN | NaN | 92.000000 | 10.000000 | 250898.090000 | 4.000000 | 1.000000 | 1.000000 | 199992.480000 | 1.000000 |

Question-5:

Handle the Missing values.

Solution:

```
df.fillna(0)
```



A Jupyter Notebook interface showing a code cell with the command `df.fillna(0)` and its output. The output is a DataFrame with 10000 rows and 14 columns, where missing values have been replaced with 0.

| | 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 | Obijaku | 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 x 14 columns

```
d.isnull(df["HasCrCard"])
```

```
pd.isnull(df["HasCrCard"])

0      False
1      False
2      False
3      False
4      False
...
9995   False
9996   False
9997   False
9998   False
9999   False
Name: HasCrCard, Length: 10000, dtype: bool
```

Question-6:

Find the outliers and replace the outliers

Solution:

```
median = float(df['Tenure'].median())
df["Tenure"] = np.where(df["Tenure"] > 10, median, df['Tenure'])
df["Tenure"]
```

6. Finding the outliers and replace the outliers

```
[ ] median = float(df['Tenure'].median())
df["Tenure"] = np.where(df["Tenure"] > 10, median, df['Tenure'])
df["Tenure"]
```

```
0      2.0
1      1.0
2      8.0
3      1.0
4      2.0
...
9995   5.0
9996  10.0
9997   7.0
9998   3.0
9999   4.0
Name: Tenure, Length: 10000, dtype: float64
```

Question-7:

Check for Categorical columns and perform encoding.

Solution:

```
pd.get_dummies(df, columns=["Tenure", "CreditScore"], prefix=["CreditScore", "Tenure"]).head()
```

7. Check for categorical columns and perform encoding

```
pd.get_dummies(df, columns=["Tenure", "CreditScore"], prefix=["CreditScore", "Tenure"]).head()
```

| RowNumber | CustomerId | Surname | Geography | Gender | Age | Balance | NumOfProducts | HasCrCard | IsActiveMember | ... | Tenure_841 | Tenure_842 | Tenure_843 | Tenure_844 | Tenure_845 | Tenure_846 | Tenure_847 | Tenure_848 |
|-----------|------------|----------|-----------|--------|--------|---------|---------------|-----------|----------------|-----|------------|------------|------------|------------|------------|------------|------------|------------|
| 0 | 1 | 15634602 | Hargrave | France | Female | 42 | 0.00 | 1 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 15647311 | Hill | Spain | Female | 41 | 83807.86 | 1 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 3 | 15619304 | Onio | France | Female | 42 | 159660.80 | 3 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 4 | 15701354 | Boni | France | Female | 39 | 0.00 | 2 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 5 | 15737888 | Mitchell | Spain | Female | 43 | 125510.82 | 1 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 483 columns

Question-8:

Split the data into dependent and independent variables.

Solution:

Dependent Variable

```
x= df.iloc[:, -2].values
```

```
print(x)
```

8. Split the data into dependent and independent variables.

Dependent Variable

```
[ ] x= df.iloc[:, -2].values
    print(x)
```

```
[101348.88 112542.58 113931.57 ... 42085.58 92888.52 38190.78]
```

Independent Variable

```
y= df.iloc[:, :-2].values
```

```
print(y)
```

Independent Variable

```
[ ] y= df.iloc[:, :-2].values
    print(y)
```

```
[[1 15634602 'Hargrave' ... 1 1 1]
 [2 15647311 'Hill' ... 1 0 1]
 [3 15619304 'Onio' ... 3 1 0]
 ...
 [9998 15584532 'Liu' ... 1 0 1]
 [9999 15682355 'Sabbatini' ... 2 1 0]
 [10000 15628319 'Walker' ... 1 1 0]]
```

Question-9:

Scale the independent variables

Solution:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[["Tenure"]] = scaler.fit_transform(df[["Tenure"]])
print(df)
```

9. Scale the Independent variables

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[["Tenure"]] = scaler.fit_transform(df[["Tenure"]])
print(df)
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | \ |
|------|-----------|------------|-----------|-------------|-----------|--------|-----|---|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | |
| 1 | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | |
| 2 | 3 | 15619304 | Onio | 502 | France | Female | 42 | |
| 3 | 4 | 15701354 | Boni | 699 | France | Female | 39 | |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| 9995 | 9996 | 15606229 | Obijaku | 771 | France | Male | 39 | |
| 9996 | 9997 | 15569892 | Johnstone | 516 | France | Male | 35 | |
| 9997 | 9998 | 15584532 | Liu | 709 | France | Female | 36 | |
| 9998 | 9999 | 15682355 | Sabbatini | 772 | Germany | Male | 42 | |
| 9999 | 10000 | 15628319 | Walker | 792 | France | Female | 28 | |

| | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | \ |
|------|--------|-----------|---------------|-----------|----------------|---|
| 0 | 0.2 | 0.00 | 1 | 1 | 1 | |
| 1 | 0.1 | 83807.86 | 1 | 0 | 1 | |
| 2 | 0.8 | 159660.80 | 3 | 1 | 0 | |
| 3 | 0.1 | 0.00 | 2 | 0 | 0 | |
| 4 | 0.2 | 125510.82 | 1 | 1 | 1 | |
| ... | ... | ... | ... | ... | ... | |
| 9995 | 0.5 | 0.00 | 2 | 1 | 0 | |
| 9996 | 1.0 | 57369.61 | 1 | 1 | 1 | |
| 9997 | 0.7 | 0.00 | 1 | 0 | 1 | |
| 9998 | 0.3 | 75075.31 | 2 | 1 | 0 | |
| 9999 | 0.4 | 130142.79 | 1 | 1 | 0 | |

| | EstimatedSalary | Exited |
|------|-----------------|--------|
| 0 | 101348.88 | 1 |
| 1 | 112542.58 | 0 |
| 2 | 113931.57 | 1 |
| 3 | 93826.63 | 0 |
| 4 | 79084.10 | 0 |
| ... | ... | ... |
| 9995 | 96270.64 | 0 |
| 9996 | 101699.77 | 0 |
| 9997 | 42085.58 | 1 |
| 9998 | 92888.52 | 1 |
| 9999 | 38190.78 | 0 |

Question-10:

Testing and training data

Solution:

```
from sklearn.model_selection import train_test_split
train_size=0.7
X = df.drop(columns = ['CreditScore']).copy()
y = df['CreditScore']
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.7)
test_size = 0.4
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.4)
print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)
```

10. Split the data into training and testing

```
from sklearn.model_selection import train_test_split
train_size=0.7
X = df.drop(columns = ['CreditScore']).copy()
y = df['CreditScore']
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.7)
test_size = 0.4
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.4)
print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)
```

```
(7000, 13)
(7000,)
(1800, 13)
(1800,)
(1200, 13)
(1200,)
(None, None)
```