# Assignment -2

## Data Visualization and Pre-Processing

Date: 26 September 2022

Student Name:R.Jeyapriya

Student Roll Number : 9517201906018

**Question 1 - Load the dataset.**

SOLUTION:

import pandas as pd

import numpy as np

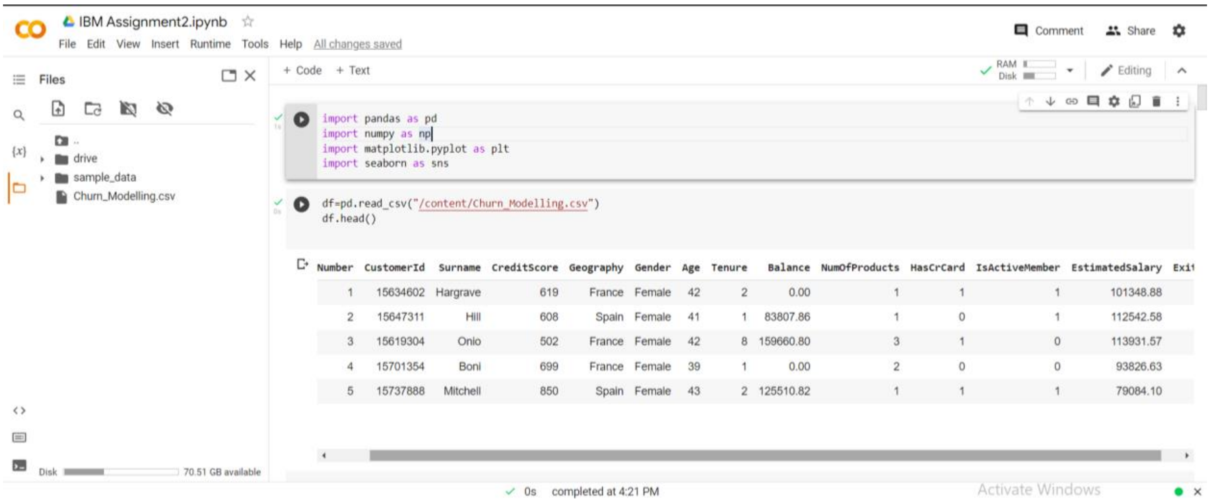
import matplotlib.pyplot as plt

import seaborn as sns

df=pd.read_csv("/content/Churn_Modelling.csv")

df.head()

**OUTPUT:**



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

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

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

**Question 2 - Perform Univariate, Bivariate and Multivariate Analysis**

SOLUTION:

sns.boxplot(df['CreditScore'])

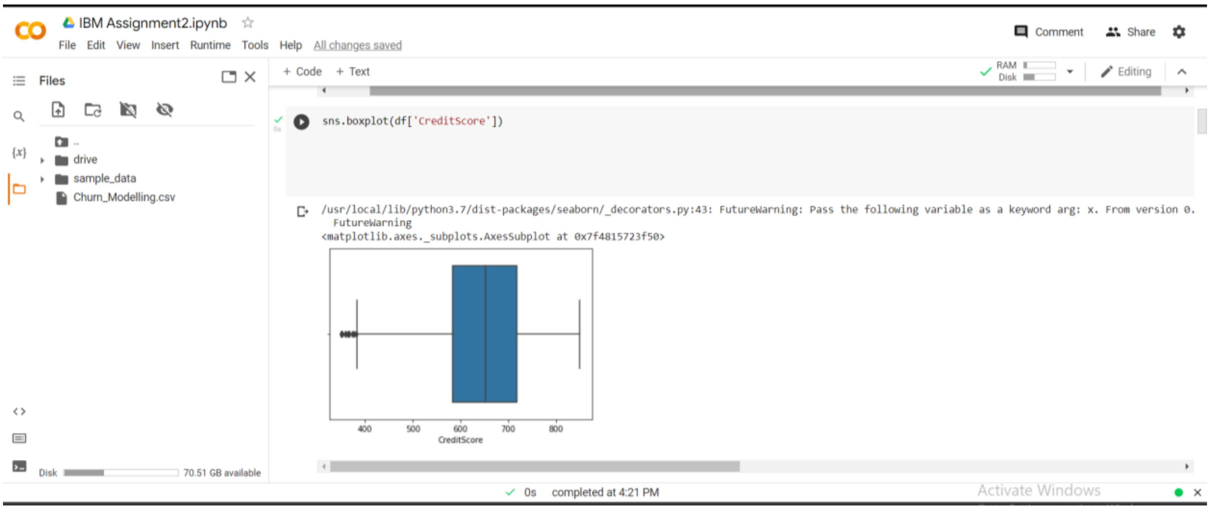
sns.boxplot(df['Age'])

sns.boxplot(df['Tenure'])

sns.boxplot(df['Balance'])

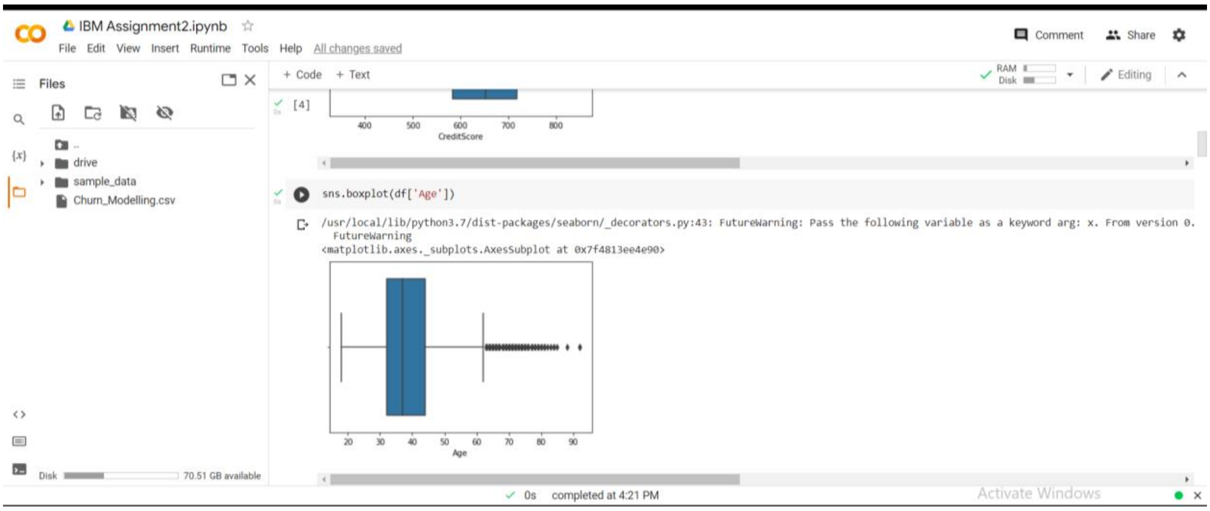
sns.boxplot(df['EstimatedSalary'])

sns.heatmap(df.corr(), annot=True)



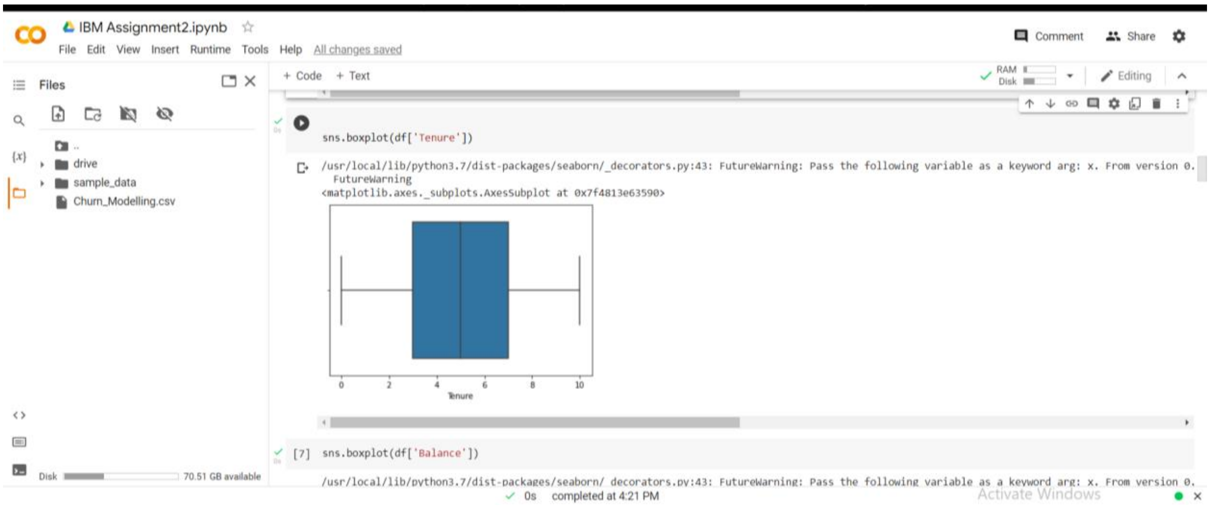
```
sns.boxplot(df['CreditScore'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4815723f50>
```



```
[4]
sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4813ee4e90>
```

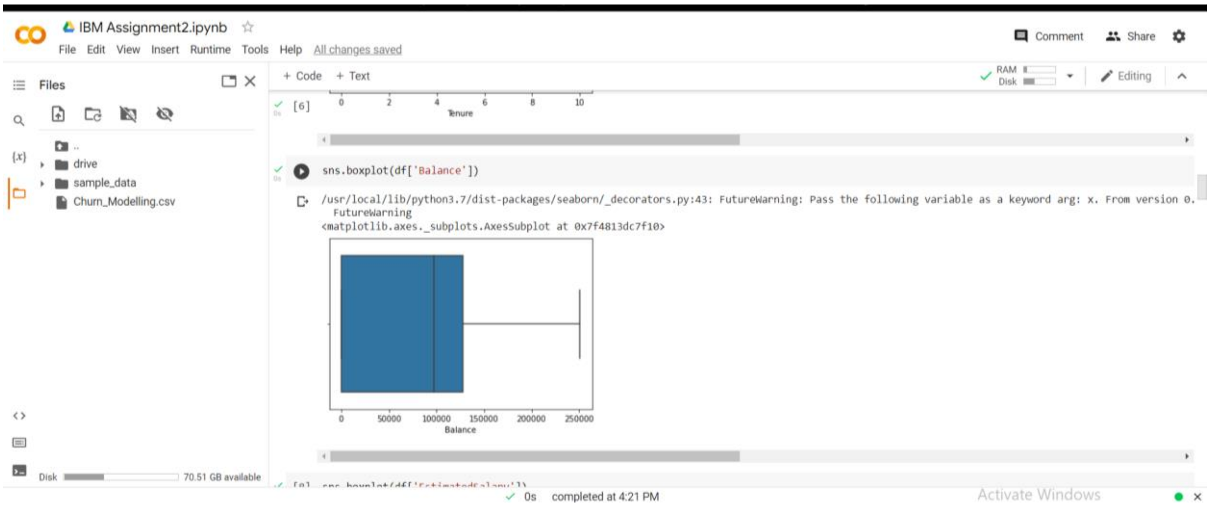


```
sns.boxplot(df['Tenure'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4813e63590>
```

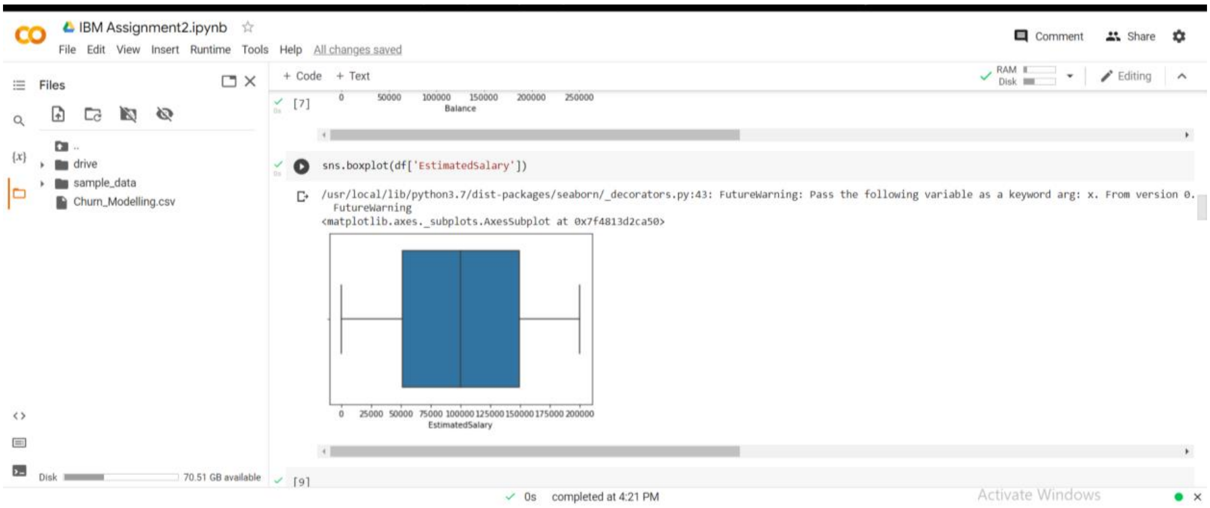
```
[7] sns.boxplot(df['Balance'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
```



```
[6]
sns.boxplot(df['Balance'])
```

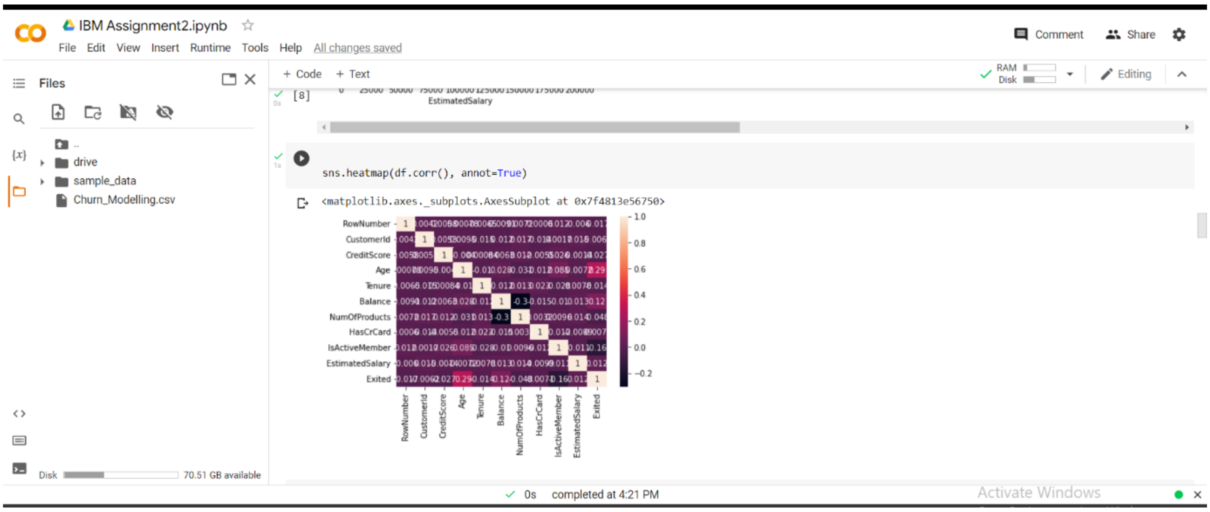
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4813dc7f10>
```



```
[7]
sns.boxplot(df['EstimatedSalary'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4813d2ca50>
```

```
[9]
```



```
[8]
sns.heatmap(df.corr(), annot=True)
```

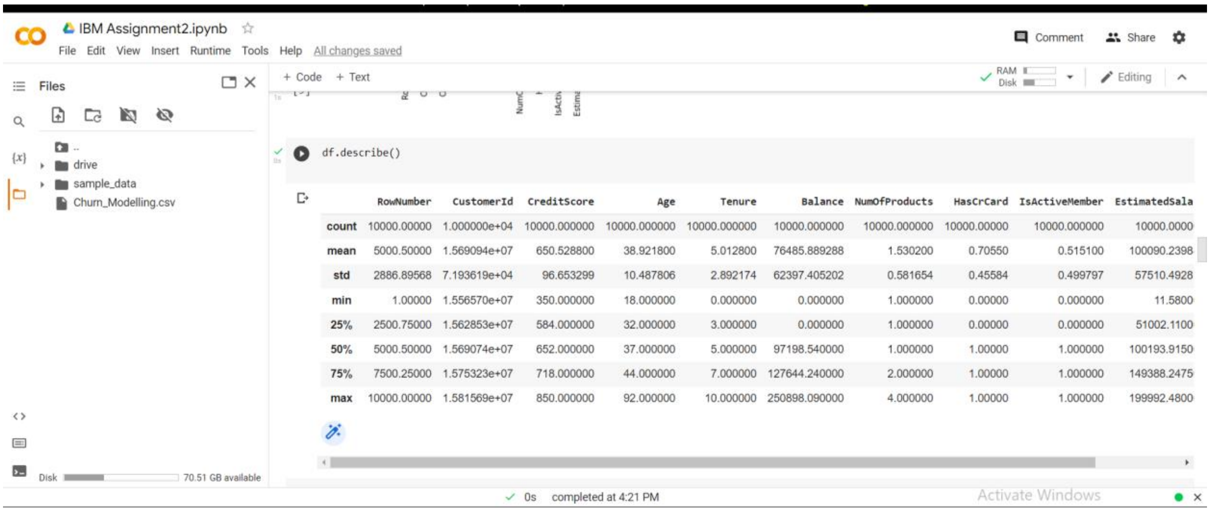
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4813e56750>
```

## Question 3 - Perform descriptive statistics on the dataset.

SOLUTION:

df.describe()

OUTPUT:



```
df.describe()
```

| | RowNumber | CustomerId | CreditScore | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSala |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 10000.00000 | 1.000000e+04 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.00000 | 10000.000000 | 10000.0000 |
| mean | 5000.50000 | 1.569094e+07 | 650.528800 | 38.921800 | 5.012800 | 76485.889288 | 1.530200 | 0.70550 | 0.515100 | 100090.2398 |
| std | 2886.89568 | 7.193619e+04 | 96.653299 | 10.487806 | 2.892174 | 62397.405202 | 0.581654 | 0.45584 | 0.499797 | 57510.4928 |
| min | 1.00000 | 1.556570e+07 | 350.000000 | 18.000000 | 0.000000 | 0.000000 | 1.000000 | 0.00000 | 0.000000 | 11.5800 |
| 25% | 2500.75000 | 1.562853e+07 | 584.000000 | 32.000000 | 3.000000 | 0.000000 | 1.000000 | 0.00000 | 0.000000 | 51002.1100 |
| 50% | 5000.50000 | 1.569074e+07 | 652.000000 | 37.000000 | 5.000000 | 97198.540000 | 1.000000 | 1.00000 | 1.000000 | 100193.9150 |
| 75% | 7500.25000 | 1.575323e+07 | 718.000000 | 44.000000 | 7.000000 | 127644.240000 | 2.000000 | 1.00000 | 1.000000 | 149388.2475 |
| max | 10000.00000 | 1.581569e+07 | 850.000000 | 92.000000 | 10.000000 | 250898.090000 | 4.000000 | 1.00000 | 1.000000 | 199992.4800 |

✓ 0s completed at 4:21 PM

## Question 4 – Handle the missing values

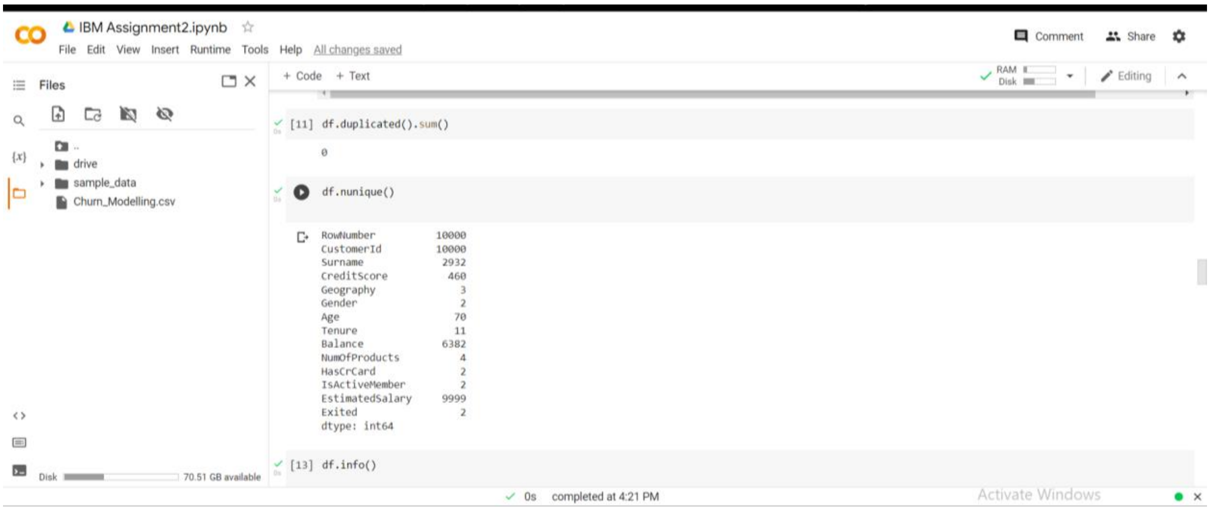
SOLUTION:

df.duplicated().sum()

df.nunique()

df.info()

OUTPUT:



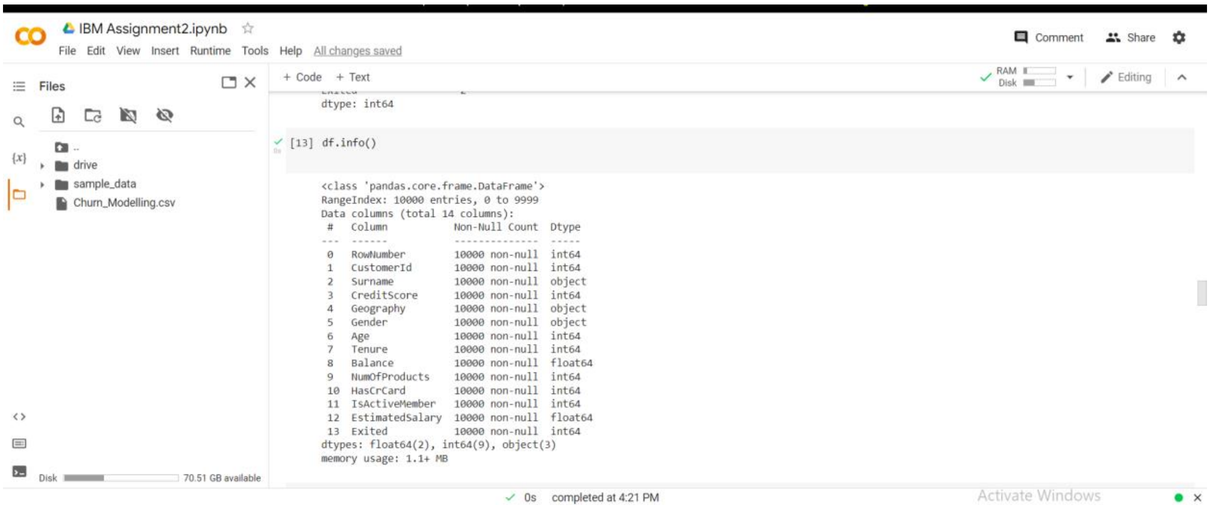
```
[11] df.duplicated().sum()

0
```

```
df.nunique()
```

```
RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

```
[13] df.info()
```

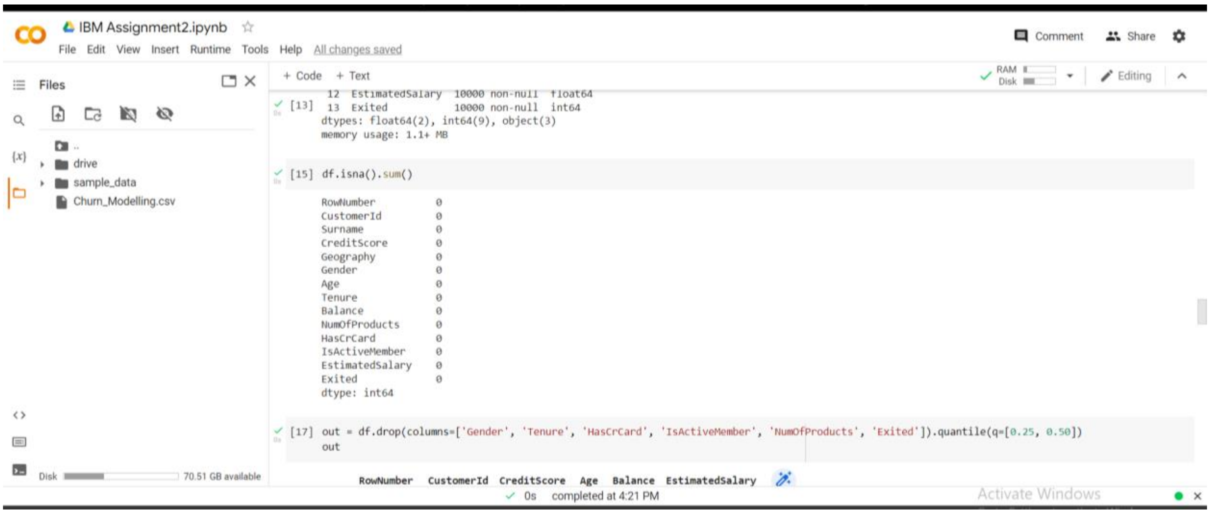
✓ 0s completed at 4:21 PM



```
dtype: int64
```

```
[13] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```



```
[13]  12  EstimatedSalary  10000 non-null  float64
      13  Exited           10000 non-null  int64
     dtypes: float64(2), int64(9), object(3)
     memory usage: 1.1+ MB
```

```
[15] df.isna().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
```

```
[17] out = df.drop(columns=['Gender', 'Tenure', 'HasCrCard', 'IsActiveMember', 'NumOfProducts', 'Exited']).quantile(q=[0.25, 0.50])
     out
```

```
     RowNumber  CustomerId  CreditScore  Age  Balance  EstimatedSalary
```

## Question 5 - Find the outliers and replace the outliers

SOLUTION:

out = df.drop(columns=['Gender', 'Tenure', 'HasCrCard', 'IsActiveMember', 'NumOfProducts', 'Exi ted']).quantile(q=[0.25, 0.50])

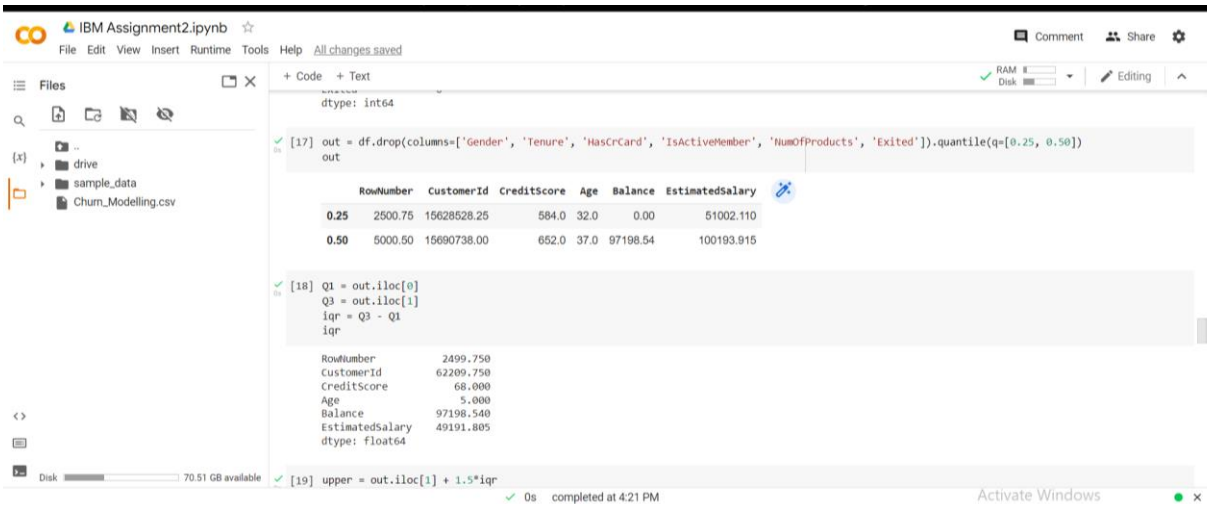
out

Q1 = out.iloc[0]

Q3 = out.iloc[1]

iqr = Q3 - Q1

iqr



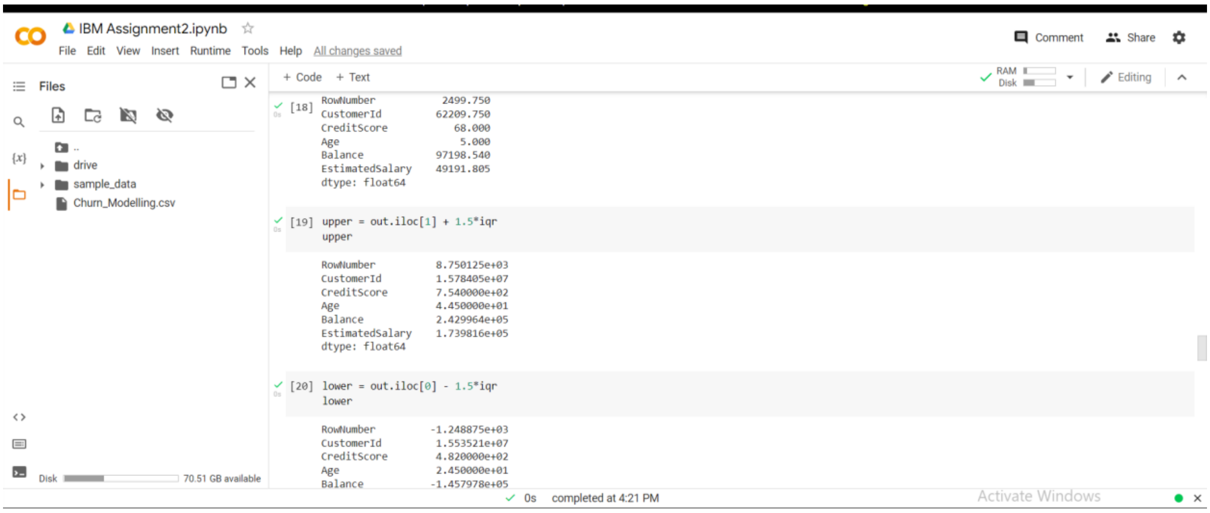
```
dtype: int64

[17] out = df.drop(columns=['Gender', 'Tenure', 'HasCrCard', 'IsActiveMember', 'NumOfProducts', 'Exited']).quantile(q=[0.25, 0.50])
     out
```

| | RowNumber | CustomerId | CreditScore | Age | Balance | EstimatedSalary |
|---|---|---|---|---|---|---|
| 0.25 | 2500.75 | 15628528.25 | 584.0 | 32.0 | 0.00 | 51002.110 |
| 0.50 | 5000.50 | 15690738.00 | 652.0 | 37.0 | 97198.54 | 100193.915 |

```
[18] Q1 = out.iloc[0]
     Q3 = out.iloc[1]
     iqr = Q3 - Q1
     iqr

RowNumber           2499.750
CustomerId         62209.750
CreditScore           68.000
Age                    5.000
Balance            97198.540
EstimatedSalary    49191.805
dtype: float64

[19] upper = out.iloc[1] + 1.5*iqr
```

upper = out.iloc[1] + 1.5*iqr

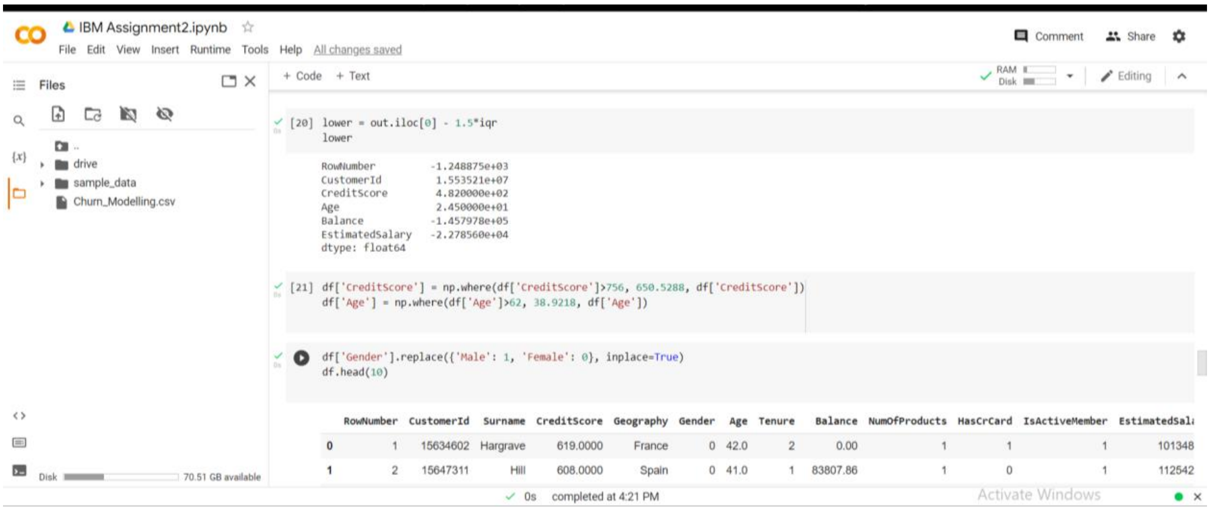
upper



```
[18] RowNumber           2499.750
     CustomerId         62209.750
     CreditScore           68.000
     Age                    5.000
     Balance            97198.540
     EstimatedSalary    49191.805
     dtype: float64

[19] upper = out.iloc[1] + 1.5*iqr
     upper

RowNumber          8.750125e+03
CustomerId         1.578405e+07
CreditScore        7.540000e+02
Age                4.450000e+01
Balance            2.429964e+05
EstimatedSalary    1.739816e+05
dtype: float64

[20] lower = out.iloc[0] - 1.5*iqr
     lower

RowNumber         -1.248875e+03
CustomerId         1.553521e+07
CreditScore        4.820000e+02
Age                2.450000e+01
Balance           -1.457978e+05
```

lower = out.iloc[0] - 1.5*iqr

lower



```
[20] lower = out.iloc[0] - 1.5*iqr
     lower

     RowNumber         -1.248875e+03
     CustomerId         1.553521e+07
     CreditScore        4.820000e+02
     Age                2.450000e+01
     Balance           -1.457978e+05
     EstimatedSalary   -2.278560e+04
     dtype: float64
```

```
[21] df['CreditScore'] = np.where(df['CreditScore']>756, 650.5288, df['CreditScore'])
     df['Age'] = np.where(df['Age']>62, 38.9218, df['Age'])
```

```
df['Gender'].replace({'Male': 1, 'Female': 0}, inplace=True)
df.head(10)
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 15634602 | Hargrave | 619.0000 | France | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 | 101348 |
| 1 | 2 | 15647311 | Hill | 608.0000 | Spain | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 | 112542 |

✓ 0s completed at 4:21 PM

## Replace outliers

SOLUTION:

df['CreditScore'] = np.where(df['CreditScore']>756, 650.5288, df['CreditScore'])

df['Age'] = np.where(df['Age']>62, 38.9218, df['Age'])

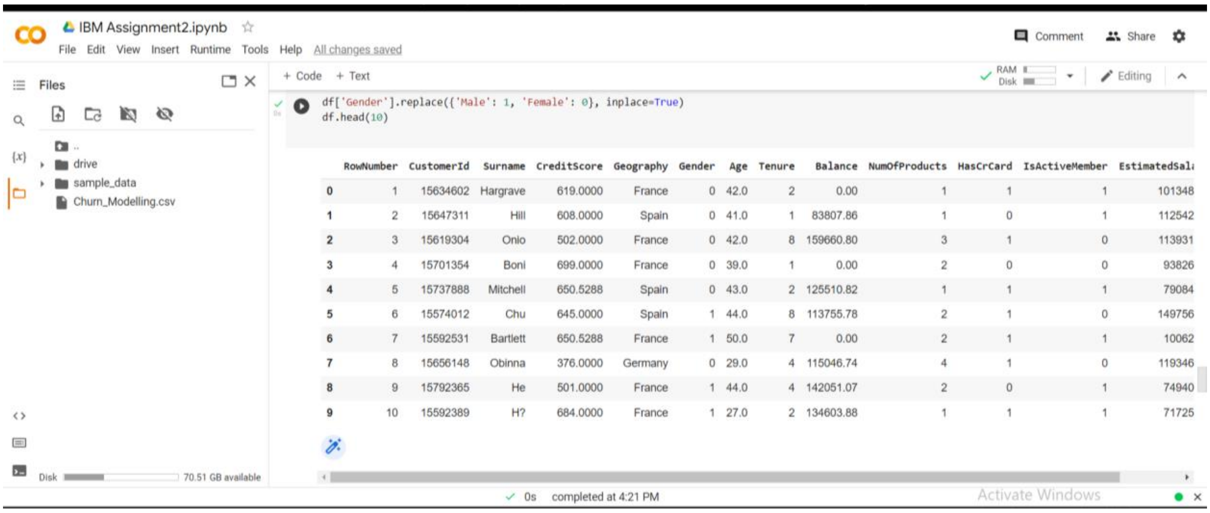
## Question 6 - Check for Categorical columns and perform encoding.

SOLUTION:

df['Gender'].replace({'Male': 1, 'Female': 0}, inplace=True)

df.head(10)

OUTPUT:



```
df['Gender'].replace({'Male': 1, 'Female': 0}, inplace=True)
df.head(10)
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 15634602 | Hargrave | 619.0000 | France | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 | 101348 |
| 1 | 2 | 15647311 | Hill | 608.0000 | Spain | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 | 112542 |
| 2 | 3 | 15619304 | Onio | 502.0000 | France | 0 | 42.0 | 8 | 159660.80 | 3 | 1 | 0 | 113931 |
| 3 | 4 | 15701354 | Boni | 699.0000 | France | 0 | 39.0 | 1 | 0.00 | 2 | 0 | 0 | 93826 |
| 4 | 5 | 15737888 | Mitchell | 650.5288 | Spain | 0 | 43.0 | 2 | 125510.82 | 1 | 1 | 1 | 79084 |
| 5 | 6 | 15574012 | Chu | 645.0000 | Spain | 1 | 44.0 | 8 | 113755.78 | 2 | 1 | 0 | 149756 |
| 6 | 7 | 15592531 | Bartlett | 650.5288 | France | 1 | 50.0 | 7 | 0.00 | 2 | 1 | 1 | 10062 |
| 7 | 8 | 15656148 | Obinna | 376.0000 | Germany | 0 | 29.0 | 4 | 115046.74 | 4 | 1 | 0 | 119346 |
| 8 | 9 | 15792365 | He | 501.0000 | France | 1 | 44.0 | 4 | 142051.07 | 2 | 0 | 1 | 74940 |
| 9 | 10 | 15592389 | H? | 684.0000 | France | 1 | 27.0 | 2 | 134603.88 | 1 | 1 | 1 | 71725 |

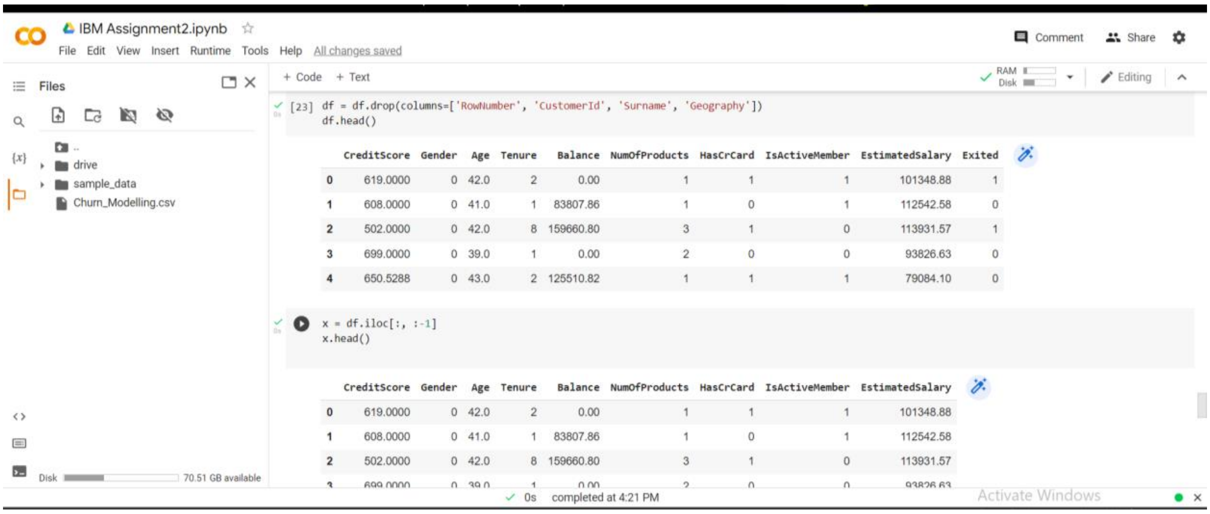
✓ 0s completed at 4:21 PM

## Question 7 – Split the data into dependent and independent variables.

SOLUTION:

df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname', 'Geography'])

df.head()



```
[23] df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname', 'Geography'])
     df.head()
```

|   | CreditScore | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | Exited |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 619.0000 | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 | 1 |
| 1 | 608.0000 | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 | 0 |
| 2 | 502.0000 | 0 | 42.0 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 | 1 |
| 3 | 699.0000 | 0 | 39.0 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 | 0 |
| 4 | 650.5288 | 0 | 43.0 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 | 0 |

```
x = df.iloc[:, :-1]
x.head()
```

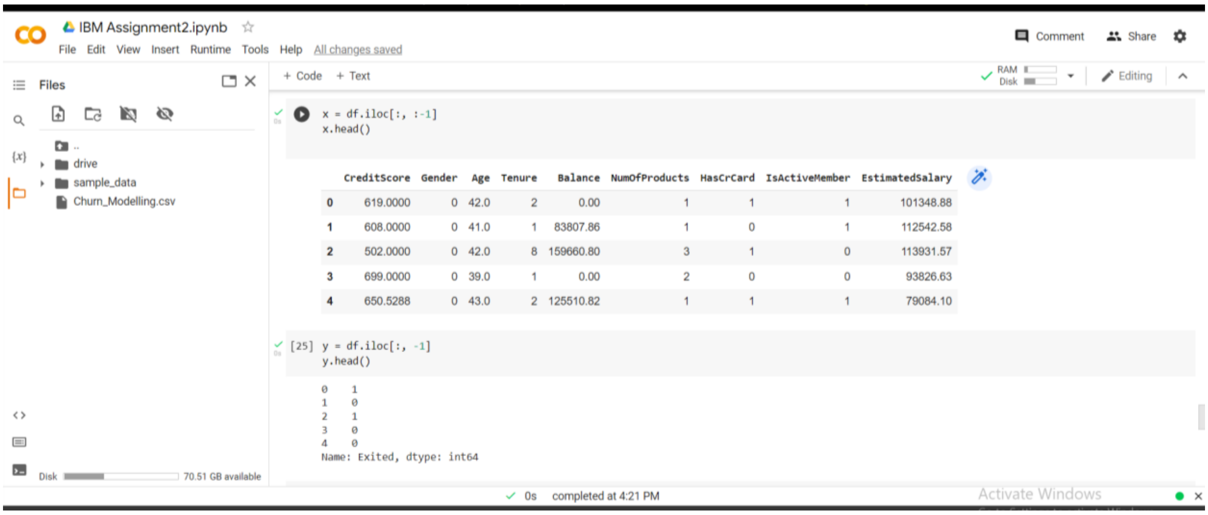
|   | CreditScore | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 619.0000 | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 |
| 1 | 608.0000 | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 |
| 2 | 502.0000 | 0 | 42.0 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 |
| 3 | 699.0000 | 0 | 39.0 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 |

x = df.iloc[:, :-1]

x.head()

y = df.iloc[:, -1]

y.head()



```
x = df.iloc[:, :-1]
x.head()
```

|   | CreditScore | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 619.0000 | 0 | 42.0 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 |
| 1 | 608.0000 | 0 | 41.0 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 |
| 2 | 502.0000 | 0 | 42.0 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 |
| 3 | 699.0000 | 0 | 39.0 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 |
| 4 | 650.5288 | 0 | 43.0 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 |

```
[25] y = df.iloc[:, -1]
     y.head()

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

## Question 8 – Scale the independent variables

SOLUTION:

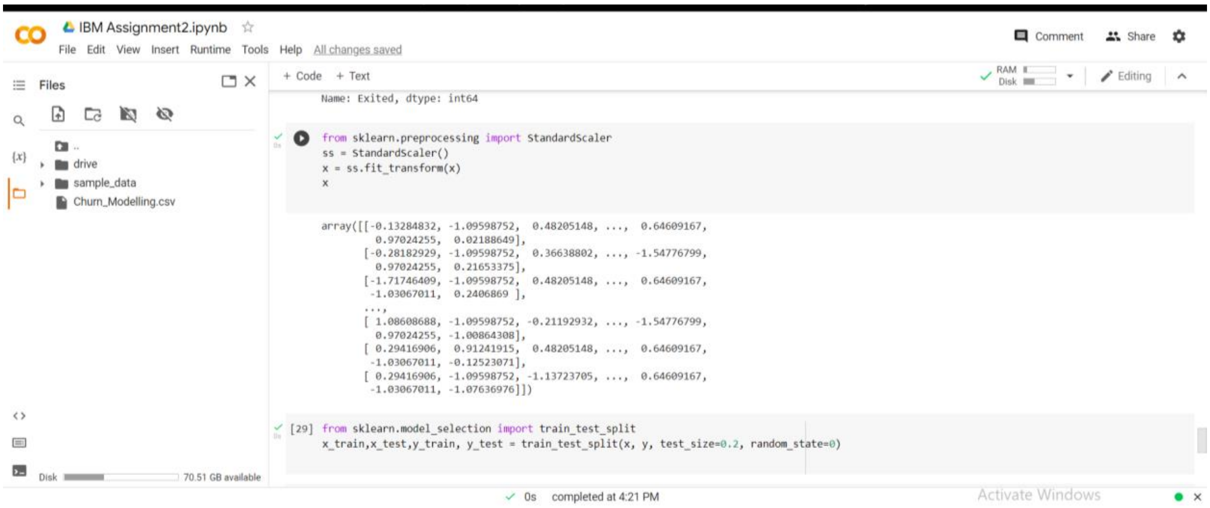
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

x = ss.fit_transform(x)

x

OUTPUT:



```
Name: Exited, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
x = ss.fit_transform(x)
x
```

```
array([[-0.13284832, -1.09598752,  0.48205148, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.28182929, -1.09598752,  0.36638802, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.71746409, -1.09598752,  0.48205148, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 1.08608688, -1.09598752, -0.21192932, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 0.29416906,  0.91241915,  0.48205148, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 0.29416906, -1.09598752, -1.13723705, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

```
[29] 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)
```

✓ 0s completed at 4:21 PM

## Question 9 - Split the data into training and testing

SOLUTION:

from sklearn.model_selection import train_test_split

x_train,x_text,y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

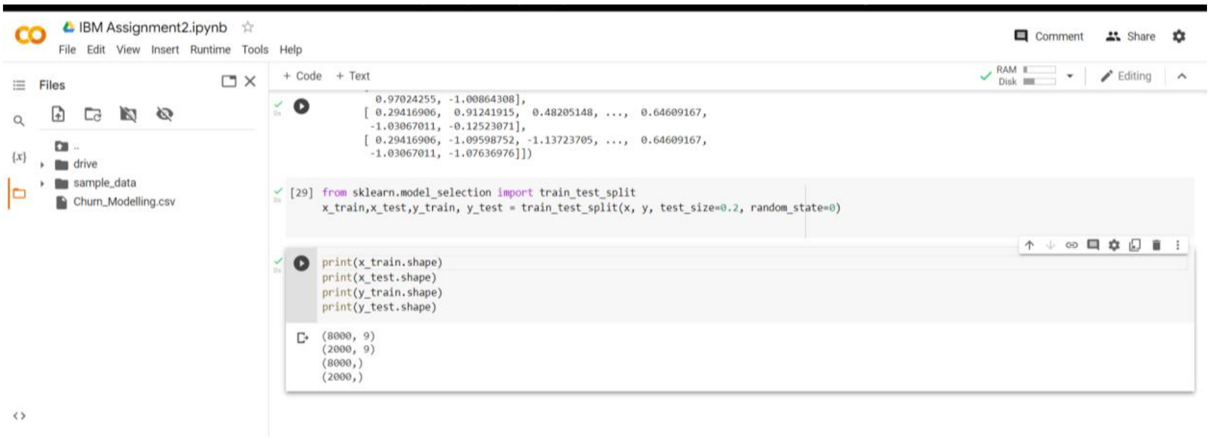
print(x_train.shape)

print(x_test.shape)

print(y_train.shape)

print(y_test.shape)

OUTPUT:



```
  0.97024255, -1.00864308],
[ 0.29416906,  0.91241915,  0.48205148, ...,  0.64609167,
 -1.03067011, -0.12523071],
[ 0.29416906, -1.09598752, -1.13723705, ...,  0.64609167,
 -1.03067011, -1.07636976]])
```

```
[29] 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)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(8000, 9)
(2000, 9)
(8000,)
(2000,)
```