Assignment 2

Assignment Date	21.09.2022
Student Name	V.Srividhya
Student Roll Number	2019115106
Maximum Marks	2 Marks

Question-1:

Download Dataset

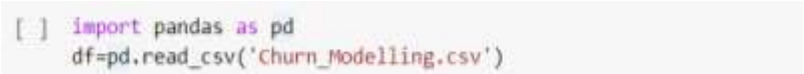
Question-2:

Loading the dataset

Solution:

```
import pandas as pd
df=pd.read_csv('Churn_Modelling.csv')
```

Screenshot:



```
[ ] import pandas as pd
    df=pd.read_csv('Churn_Modelling.csv')
```

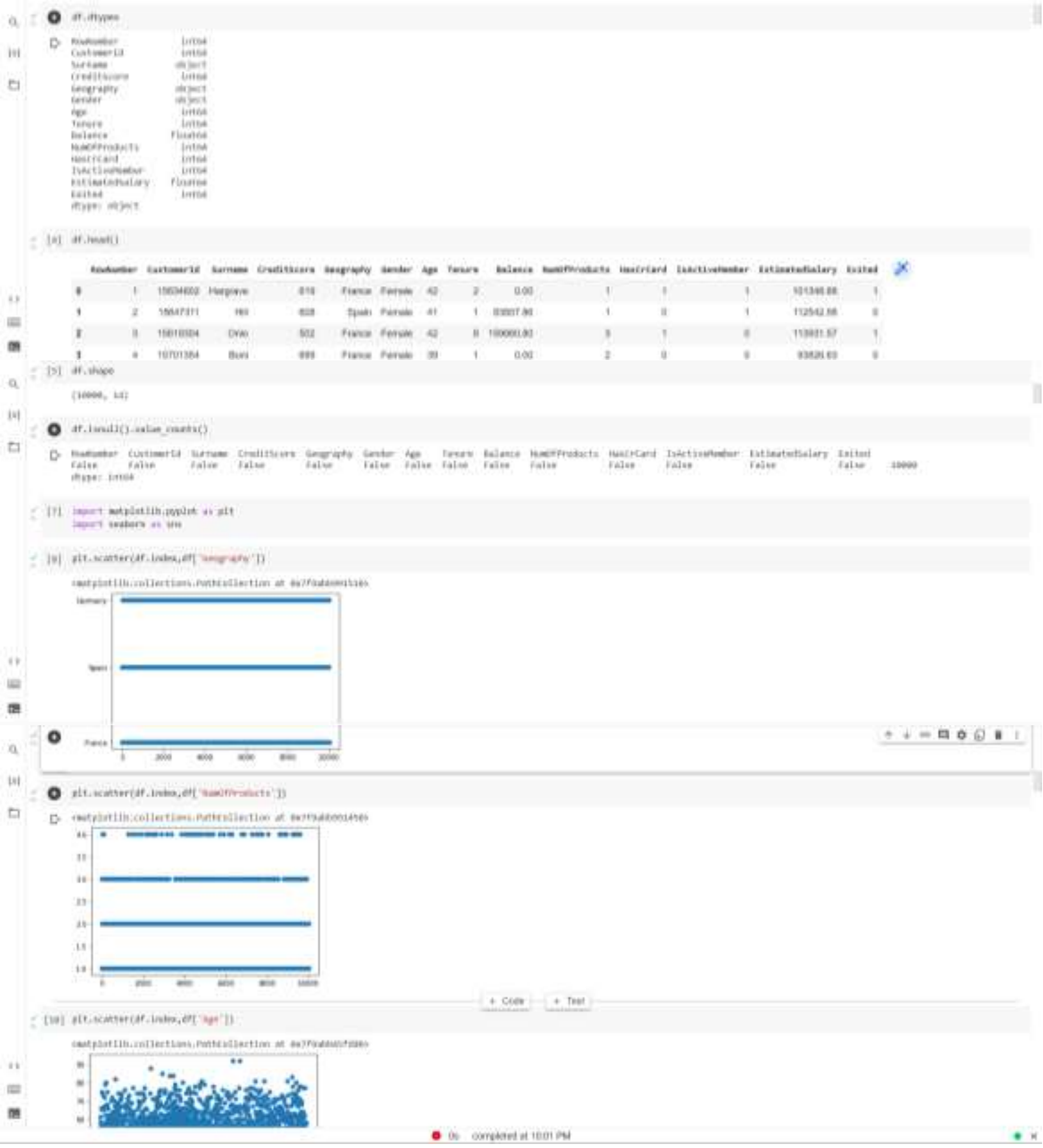
Question-3.1:

Univariate Analysis

Solution:

```
df.dtypes
df.head()
df.shape
import matplotlib.pyplot as plt
import seaborn as sns
plt.scatter(df.index,df['Geography'])
plt.scatter(df.index,df['NumOfProducts'])
plt.scatter(df.index,df['Age'])
import numpy as np
print('Number of users who have a credit card ',np.sum(df['HasCrCard']==1))
print('Number of users who dont have a credit card ',np.sum(df['HasCrCard']==0))
plt.hist(df['HasCrCard'])
```

Screenshot:



```
df.dtypes
```

```
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
```

```
[4] 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

```
[5] df.shape
```

```
(10000, 14)
```

```
df.isnull().value_counts()
```

```
RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age    Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  Exited
False      False       False    False        False      False   False  False   False    False          False      False           False            False     10000
dtype: int64
```

```
[7] import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[8] plt.scatter(df.index,df['Geography'])
```

```
<matplotlib.collections.PathCollection at [illegible]>
```

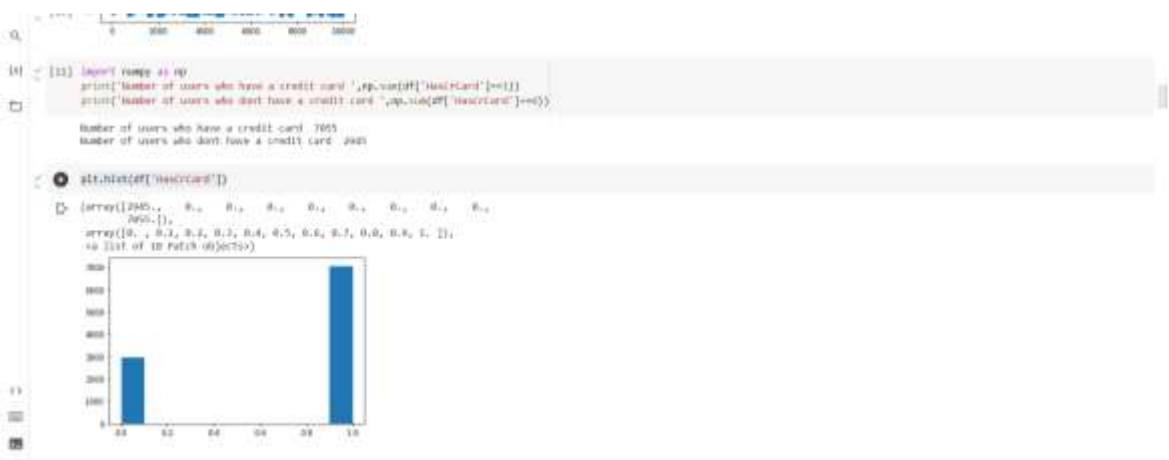
```
plt.scatter(df.index,df['NumOfProducts'])
```

```
<matplotlib.collections.PathCollection at [illegible]>
```

```
[10] plt.scatter(df.index,df['Age'])
```

```
<matplotlib.collections.PathCollection at [illegible]>
```

,



```
[11] import numpy as np
     print('Number of users who have a credit card ',np.sum(df['HasCrCard']==1))
     print('Number of users who dont have a credit card ',np.sum(df['HasCrCard']==0))
```

```
Number of users who have a credit card  7055
Number of users who dont have a credit card  2945
```

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

```
(array([2945.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
        7055.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```

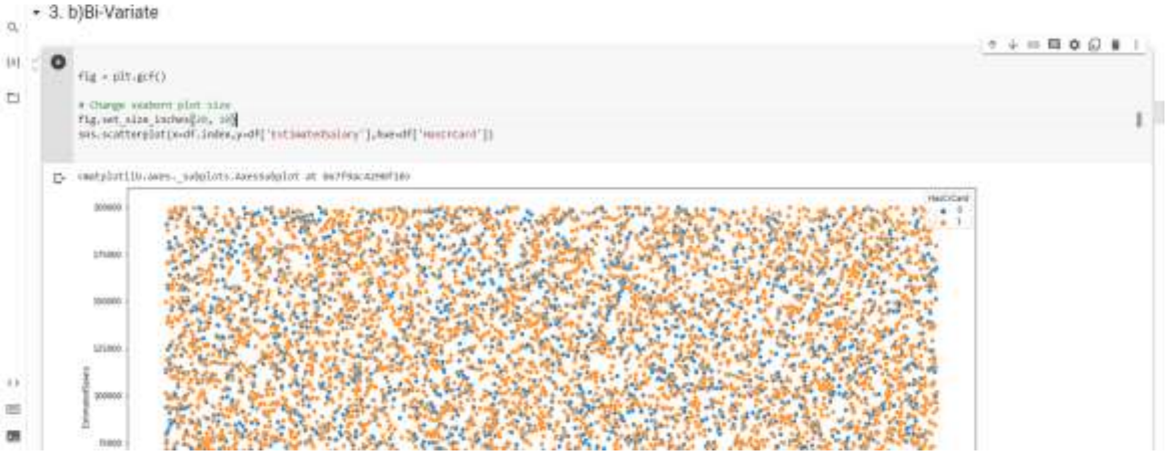
Question-3.2:

Bi-Variate

Solution:

```
fig = plt.gcf()

# Change seaborn plot size
fig.set_size_inches(20, 10)
sns.scatterplot(x=df.index,y=df['EstimatedSalary'],hue=df['HasCrCard'])

df.corr()
```

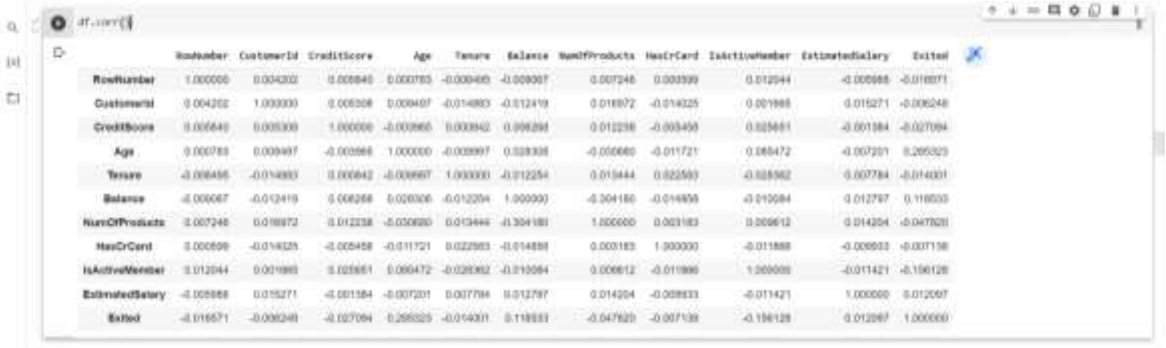
Screenshot:



3. b)Bi-Variate

```
fig = plt.gcf()

# Change seaborn plot size
fig.set_size_inches(20, 10)
sns.scatterplot(x=df.index,y=df['EstimatedSalary'],hue=df['HasCrCard'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac42e0f10>
```



```
df.corr()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599	0.012044	-0.005988	-0.016571
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025	0.001665	0.015271	-0.006248
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

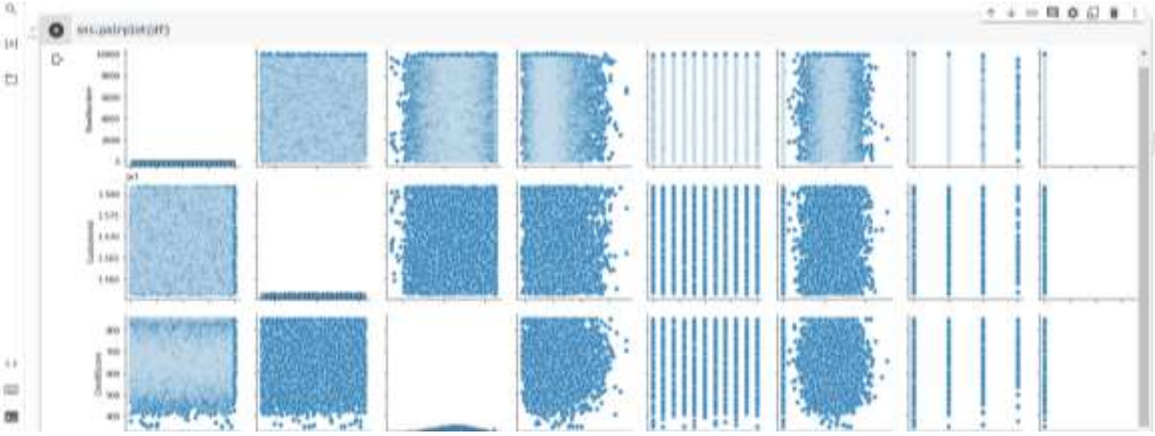
Question-3.3:

Multi Variate

Solution:

```
sns.pairplot(df)
```

Screenshot:



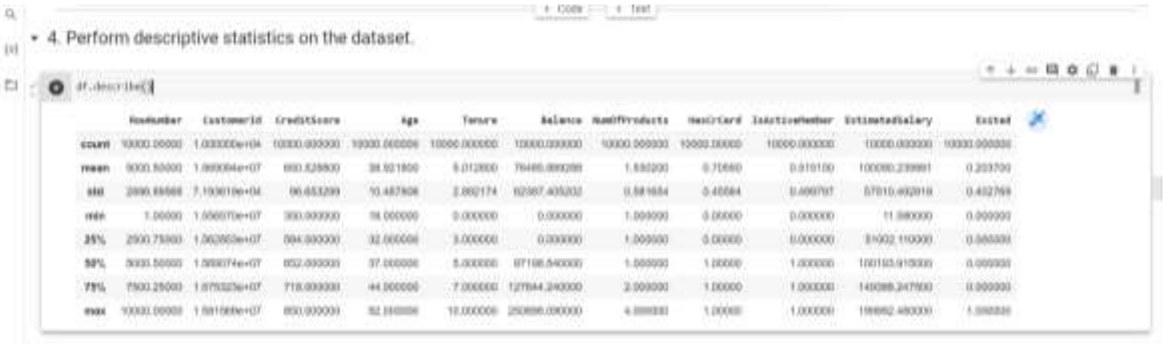
```
sns.pairplot(df)
```

Question-4:

Perform descriptive statistics on the dataset.

Solution:
df.describe()

Screenshot:



4. Perform descriptive statistics on the dataset.

```
df.describe()
```

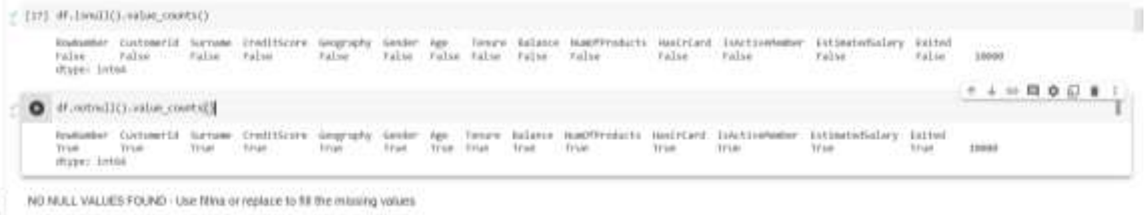
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	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

Question-5:

Handle the Missing values.

Solution:
df.isnull().value_counts()
df.notnull().value_counts()

Screenshot:



```
[17] df.isnull().value_counts()

RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age    Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  Exited
False      False       False    False        False      False   False  False   False    False          False      False           False            False     10000
dtype: int64
```

```
df.notnull().value_counts()

RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age   Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  Exited
True       True        True     True         True       True    True  True    True     True           True       True            True             True     10000
dtype: int64
```

NO NULL VALUES FOUND - Use fillna or replace to fill the missing values

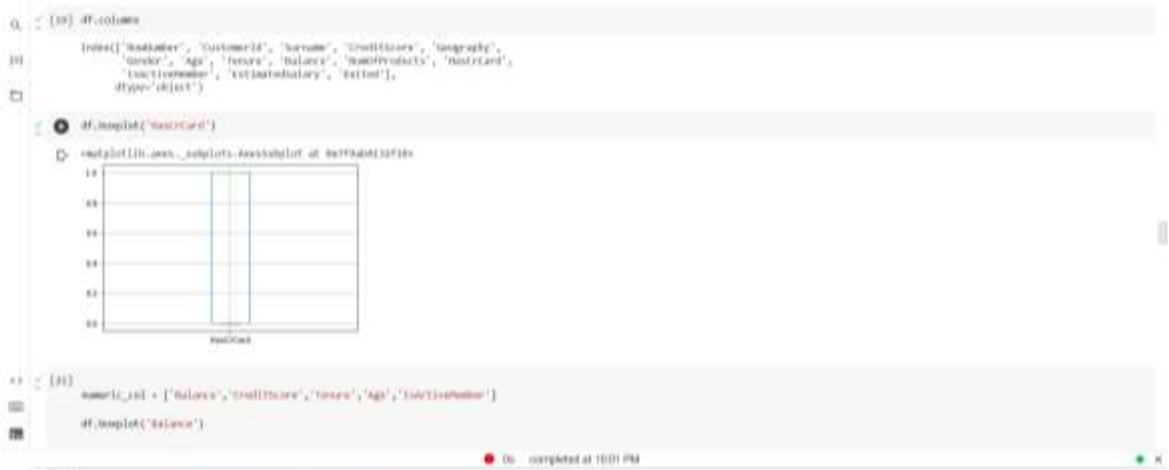
Question-6:

Find the outliers and replace the outliers

Solution:

```
df.columns

df.boxplot('HasCrCard')

numeric_col = ['Balance','CreditScore','Tenure','Age','IsActiveMember']

df.boxplot('Balance')

df.boxplot('CreditScore')
df.boxplot('Tenure')
df.boxplot('Age')
df.boxplot('IsActiveMember')
for x in ['CreditScore']:
    q75,q25 = np.percentile(df.loc[:,x],[75,25])
    intr_qr = q75-q25

    max = q75+(1.5*intr_qr)
    min = q25-(1.5*intr_qr)

    df.loc[df[x] < min,x] = np.nan
    df.loc[df[x] > max,x] = np.nan
for x in ['Age']:
    q75,q25 = np.percentile(df.loc[:,x],[75,25])
    intr_qr = q75-q25

    max = q75+(1.5*intr_qr)
    min = q25-(1.5*intr_qr)

    df.loc[df[x] < min,x] = np.nan
    df.loc[df[x] > max,x] = np.nan
df.boxplot('CreditScore')
df.boxplot('Age')
```

Screenshot:

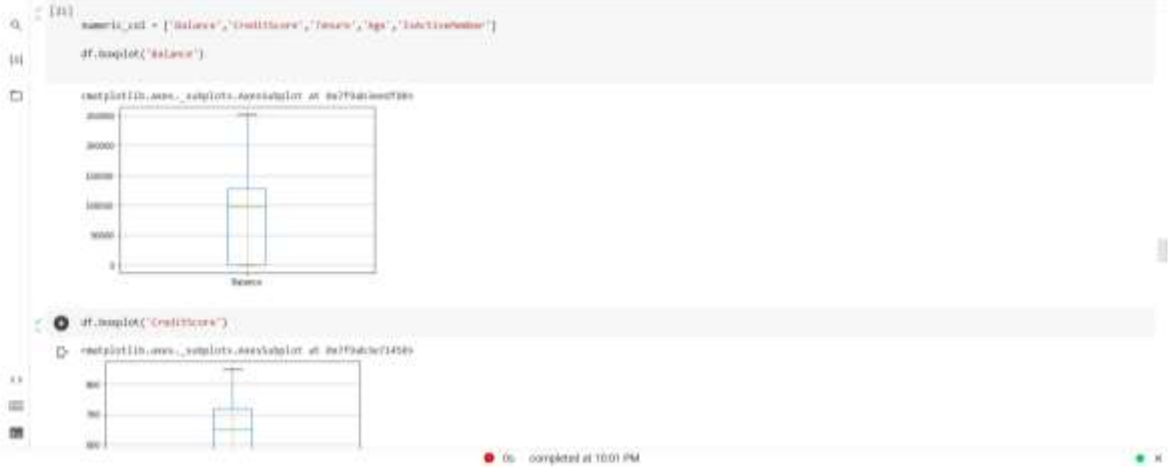


```
[19] df.columns

Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```
df.boxplot('HasCrCard')

<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

```
[21]
numeric_col = ['Balance','CreditScore','Tenure','Age','IsActiveMember']

df.boxplot('Balance')
```

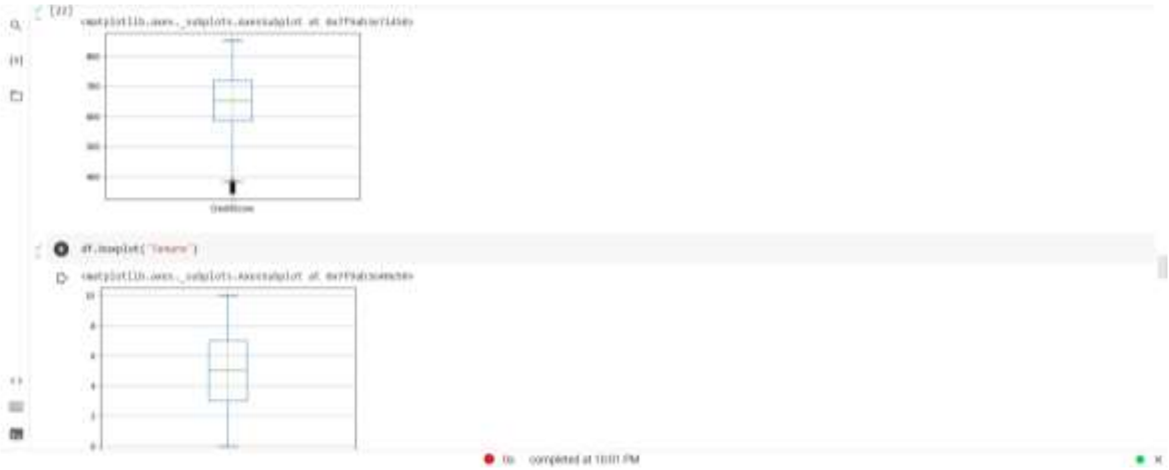
0s completed at 10:01 PM



```
[21]
numeric_col = ['Balance','CreditScore','Tenure','Age','IsActiveMember']

df.boxplot('Balance')

<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

```
df.boxplot('CreditScore')

<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

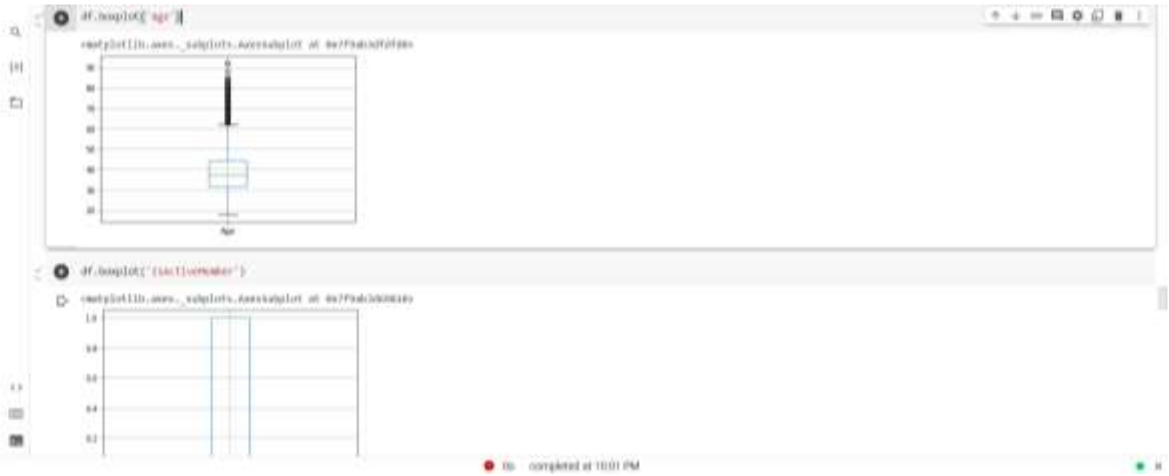
0s completed at 10:01 PM



```
[22]
<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

```
df.boxplot('Tenure')

<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

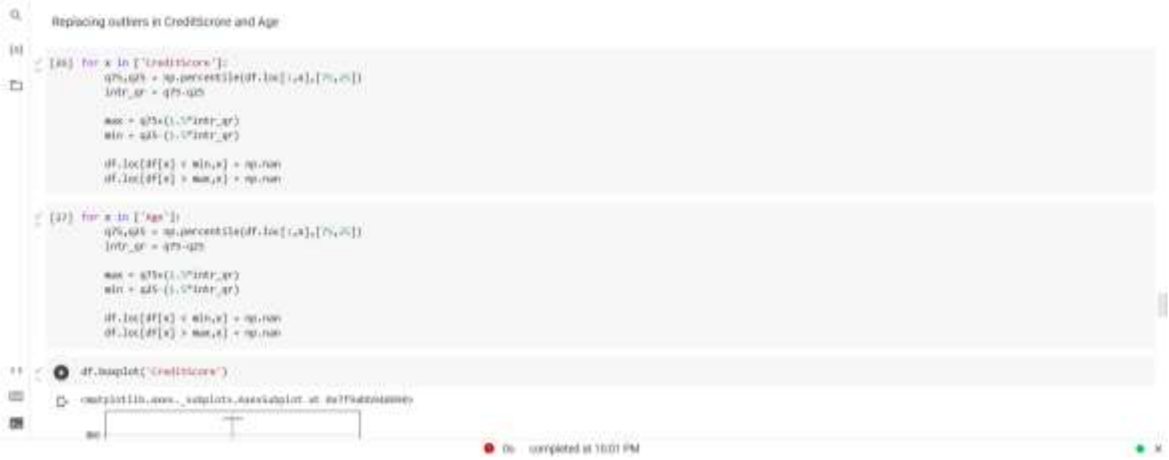
0s completed at 10:01 PM



```
df.boxplot('Age')

<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

```
df.boxplot('IsActiveMember')

<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

0s completed at 10:01 PM



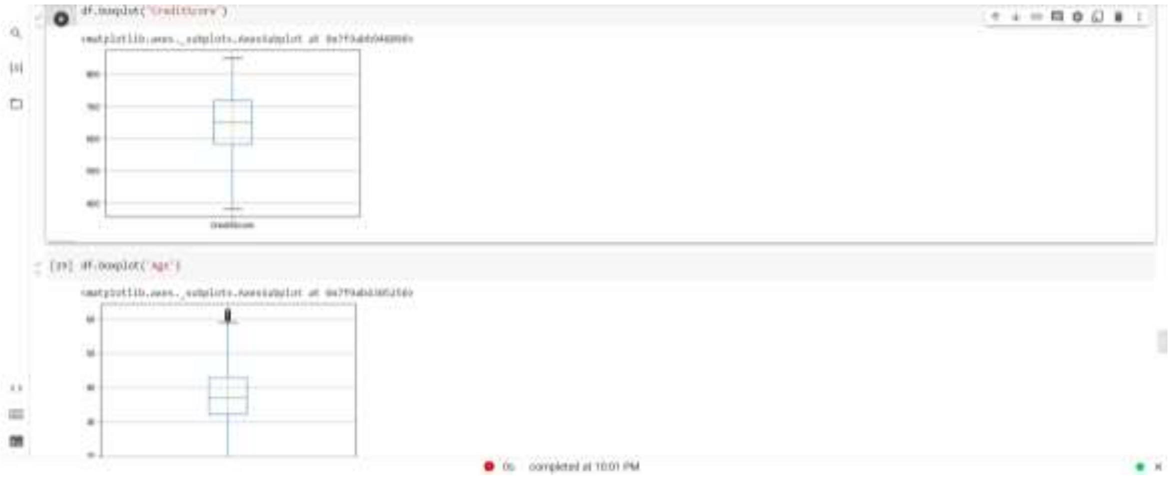
Replacing outliers in CreditScrore and Age

```
[26] for x in ['CreditScore']:
         q75,q25 = np.percentile(df.loc[:,x],[75,25])
         intr_qr = q75-q25

         max = q75+(1.5*intr_qr)
         min = q25-(1.5*intr_qr)

         df.loc[df[x] < min,x] = np.nan
         df.loc[df[x] > max,x] = np.nan
```

```
[27] for x in ['Age']:
         q75,q25 = np.percentile(df.loc[:,x],[75,25])
         intr_qr = q75-q25

         max = q75+(1.5*intr_qr)
         min = q25-(1.5*intr_qr)

         df.loc[df[x] < min,x] = np.nan
         df.loc[df[x] > max,x] = np.nan
```

```
df.boxplot('CreditScore')
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ab[illegible]>
```

0s completed at 10:01 PM



```
df.boxplot('CreditScore')
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ab[illegible]>
```

```
[29] df.boxplot('Age')
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ab[illegible]>
```

0s completed at 10:01 PM

Question-7:

Check for Categorical columns and perform encoding.

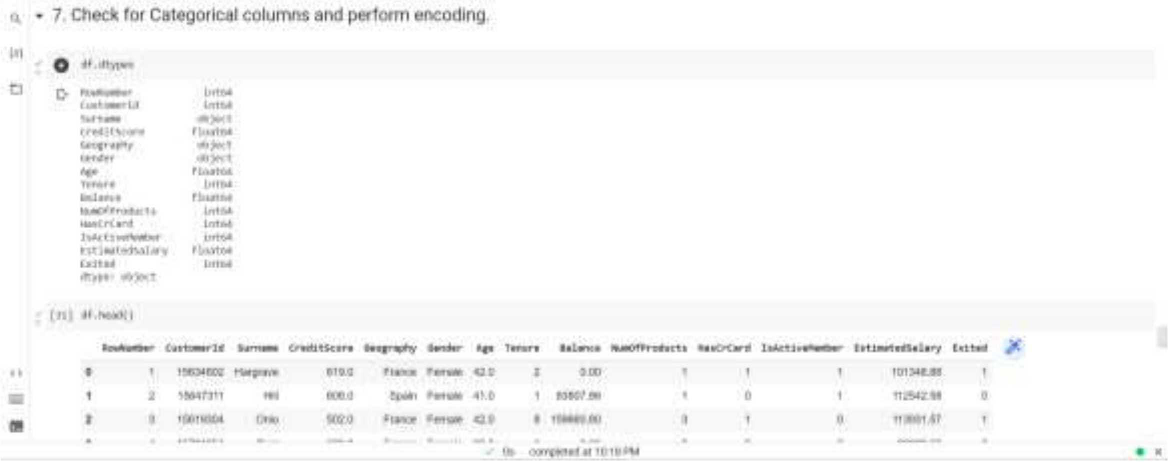
Solution:

df.dtypes
df.head()

df_categorical=df[['Geography','Gender']]

using OneHotEncoding
from sklearn.preprocessing import OneHotEncoder
df = pd.get_dummies(df, columns = ['Geography','Gender'])
df.head()

Screenshot:



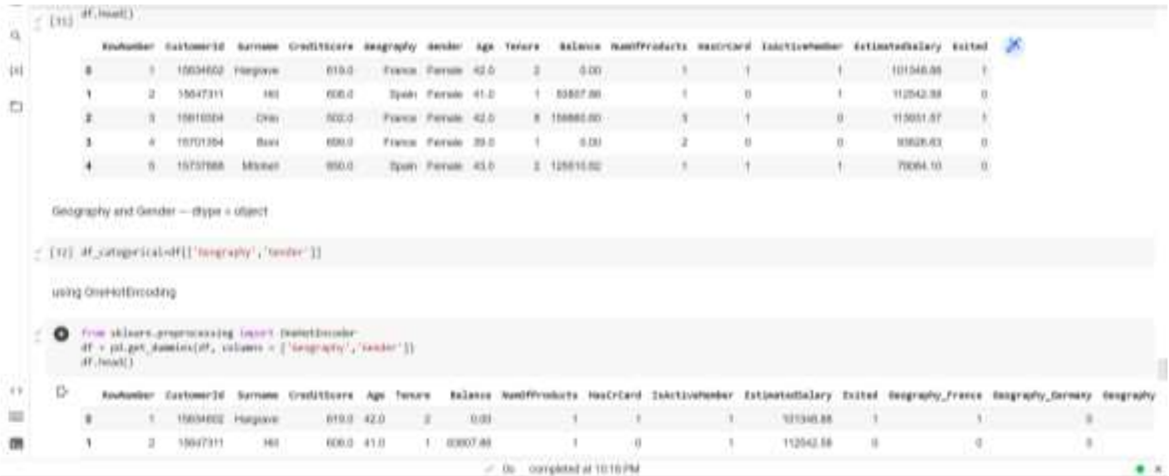
7. Check for Categorical columns and perform encoding.

```
df.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore        float64
Geography           object
Gender              object
Age                float64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

```
[11] df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619.0	France	Female	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608.0	Spain	Female	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502.0	France	Female	42.0	8	159660.80	3	1	0	113931.57	1



```
[11] df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619.0	France	Female	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608.0	Spain	Female	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502.0	France	Female	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699.0	France	Female	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850.0	Spain	Female	43.0	2	125510.82	1	1	1	79084.10	0

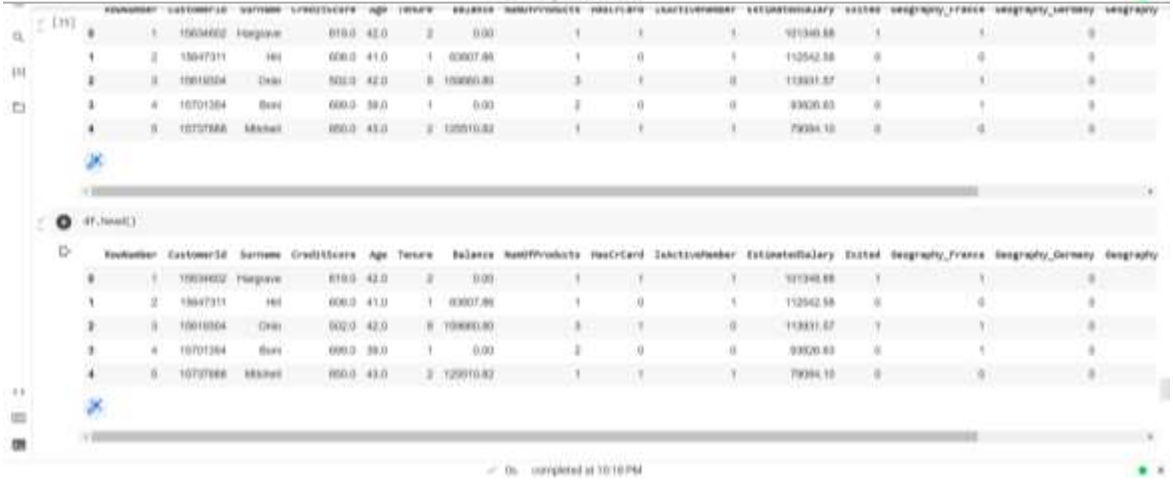
Geography and Gender -- dtype = object

```
[12] df_categorical=df[['Geography','Gender']]
```

using OneHotEncoding

```
from sklearn.preprocessing import OneHotEncoder
df = pd.get_dummies(df, columns = ['Geography','Gender'])
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geography_Germany	Geography
0	1	15634602	Hargrave	619.0	42.0	2	0.00	1	1	1	101348.88	1	1	0	
1	2	15647311	Hill	608.0	41.0	1	83807.86	1	0	1	112542.58	0	0	0	



```
[11]
```

	RowNumber	CustomerId	Surname	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geography_Germany	Geography
0	1	15634602	Hargrave	619.0	42.0	2	0.00	1	1	1	101348.88	1	1	0	
1	2	15647311	Hill	608.0	41.0	1	83807.86	1	0	1	112542.58	0	0	0	
2	3	15619304	Onio	502.0	42.0	8	159660.80	3	1	0	113931.57	1	1	0	
3	4	15701354	Boni	699.0	39.0	1	0.00	2	0	0	93826.63	0	1	0	
4	5	15737888	Mitchell	850.0	43.0	2	125510.82	1	1	1	79084.10	0	0	0	

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geography_Germany	Geography
0	1	15634602	Hargrave	619.0	42.0	2	0.00	1	1	1	101348.88	1	1	0	
1	2	15647311	Hill	608.0	41.0	1	83807.86	1	0	1	112542.58	0	0	0	
2	3	15619304	Onio	502.0	42.0	8	159660.80	3	1	0	113931.57	1	1	0	
3	4	15701354	Boni	699.0	39.0	1	0.00	2	0	0	93826.63	0	1	0	
4	5	15737888	Mitchell	850.0	43.0	2	125510.82	1	1	1	79084.10	0	0	0	

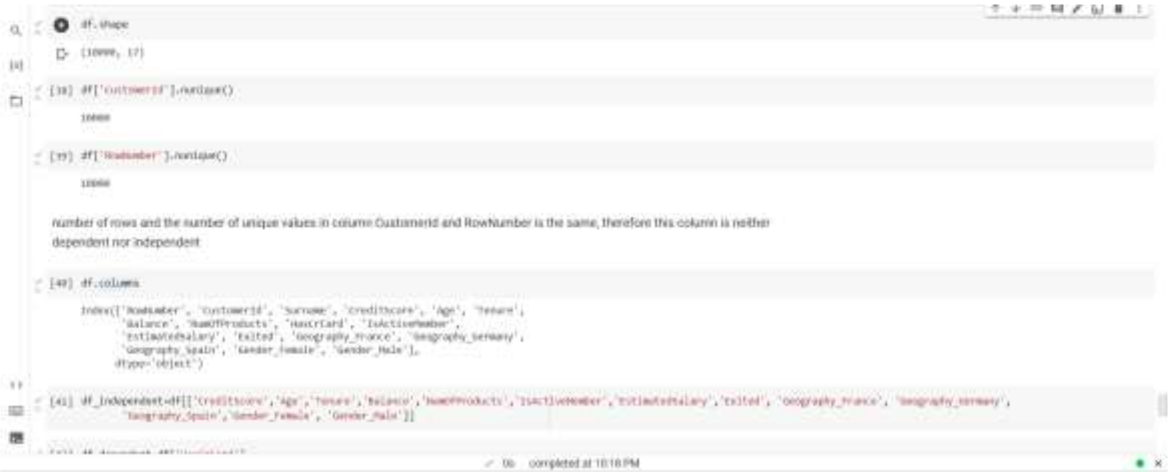
Question-8:

Split the data into dependent and independent variables

Solution:

```
df.shape
df['CustomerId'].nunique()
df['RowNumber'].nunique()
df.columns
df_independent=df[['CreditScore','Age','Tenure','Balance','NumOfProducts','IsActive
Member','EstimatedSalary','Exited', 'Geography_France', 'Geography_Germany',
    'Geography_Spain','Gender_Female', 'Gender_Male']]
df_dependent=df['HasCrCard']
```

Screenshot:



```
df.shape
(10000, 17)
```

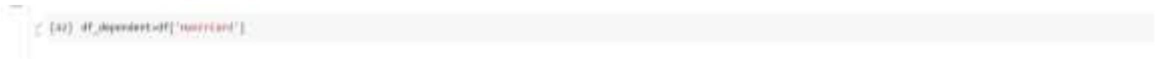
```
[38] df['CustomerId'].nunique()
     10000
```

```
[39] df['RowNumber'].nunique()
     10000
```

number of rows and the number of unique values in column CustomerId and RowNumber is the same, therefore this column is neither dependent nor independent

```
[40] df.columns
     Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Age', 'Tenure',
            'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember',
            'EstimatedSalary', 'Exited', 'Geography_France', 'Geography_Germany',
            'Geography_Spain', 'Gender_Female', 'Gender_Male'],
           dtype='object')
```

```
[41] df_independent=df[['CreditScore','Age','Tenure','Balance','NumOfProducts','IsActiveMember','EstimatedSalary','Exited', 'Geography_France', 'Geography_Germany',
         'Geography_Spain','Gender_Female', 'Gender_Male']]
```

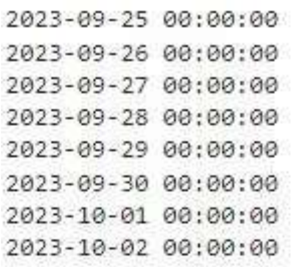


```
[42] df_dependent=df['HasCrCard']
```

.

.

.



```
2023-09-25 00:00:00
2023-09-26 00:00:00
2023-09-27 00:00:00
2023-09-28 00:00:00
2023-09-29 00:00:00
2023-09-30 00:00:00
2023-10-01 00:00:00
2023-10-02 00:00:00
```

Question-9:

Scale the independent variables

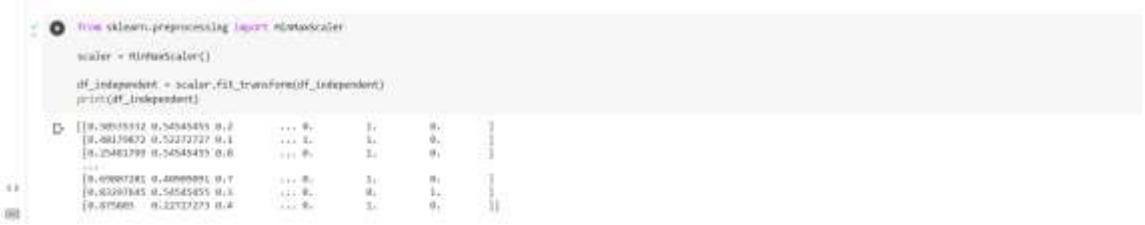
Solution:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df_independent = scaler.fit_transform(df_independent)
print(df_independent)

Screenshot:



```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df_independent = scaler.fit_transform(df_independent)
print(df_independent)
```

```
[[0.50535332 0.54545455 0.2        ... 0.         1.         0.        ]
 [0.48170872 0.52272727 0.1        ... 1.         1.         0.        ]
 [0.25481799 0.54545455 0.8        ... 0.         1.         0.        ]
 ...
 [0.69007281 0.40909091 0.7        ... 0.         1.         0.        ]
 [0.83207845 0.54545455 0.3        ... 0.         0.         1.        ]
 [0.875605   0.22727273 0.4        ... 0.         1.         0.        ]]
```

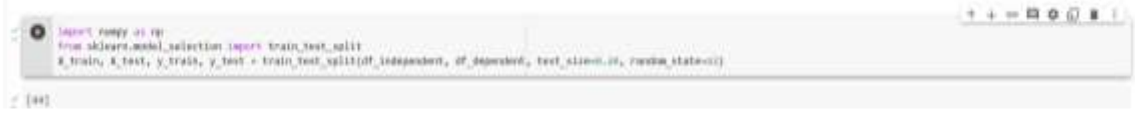
Question-10:

Split the data into training and testing

Solution:

```
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_independent, df_dependent, test_siz
e=0.20, random_state=42)
```

Screenshot:



```
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_independent, df_dependent, test_size=0.20, random_state=42)
```