Assignment – 2

Data Visualization and Data Preprocessing

Assignment Date	24 September 2022
Student Name	Rohit Jacob George
Student Roll Number	2019504572
Maximum Marks	2 Marks

Task - 1: Download the Dataset

Code:

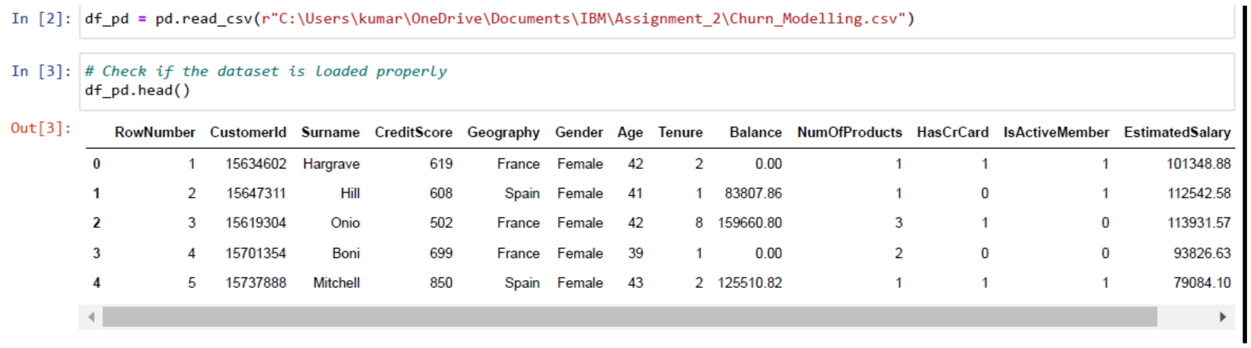
```
import pandas as pd
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

Task - 2: Load the dataset

Code:

```
df_pd =
pd.read_csv(r"C:\Users\kumar\OneDrive\Documents\IBM\Assignment_2\Churn_
Modelling.csv")
df_pd.head()
```

Output:



```
In [2]: df_pd = pd.read_csv(r"C:\Users\kumar\OneDrive\Documents\IBM\Assignment_2\Churn_Modelling.csv")

In [3]: # Check if the dataset is loaded properly
        df_pd.head()

Out[3]:
```

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

Task - 3: Perform Below Visualizations.

3.2 Univariate Analysis

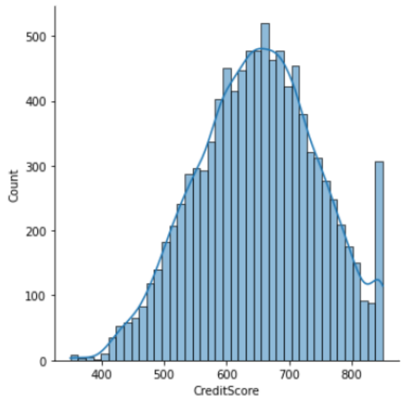
Code:

```
sns.displot(df_pd['CreditScore'], kde=True)
```

Output:

```
In [5]: #Use a distribution plot to analyse the variable - CreditScore
        sns.displot(df_pd['CreditScore'], kde=True)

Out[5]: <seaborn.axisgrid.FacetGrid at 0x25d460e8f10>
```



3.2 Bivariate Analysis

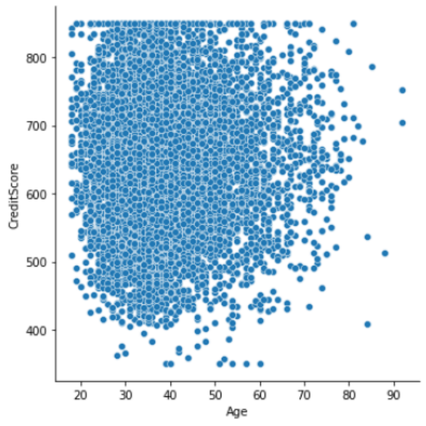
Code:

```
sns.relplot(x='Age', y='CreditScore', data=df_pd)
```

Output:

```
In [9]: sns.relplot(x='Age', y='CreditScore', data=df_pd)

Out[9]: <seaborn.axisgrid.FacetGrid at 0x25d3320ad30>
```



3.2 Bivariate Analysis

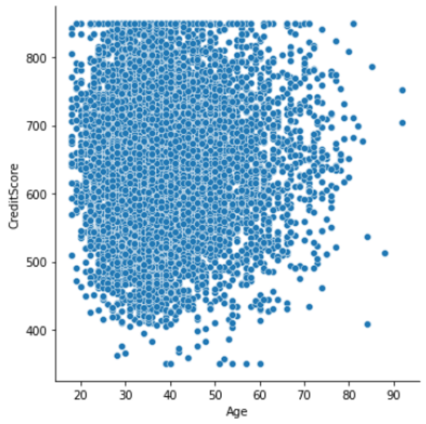
Code:

```
sns.relplot(x='Age', y='CreditScore', data=df_pd)
```

Output:

```
In [9]: sns.relplot(x='Age', y='CreditScore', data=df_pd)

Out[9]: <seaborn.axisgrid.FacetGrid at 0x25d3320ad30>
```



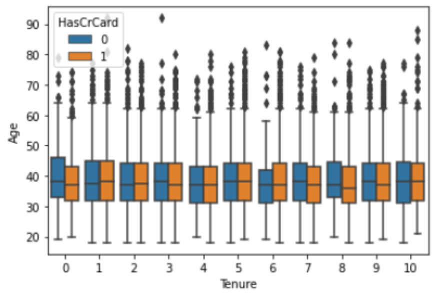
Code:

```
sns.boxplot(x='Tenure', y='Age', hue='HasCrCard', data=df_pd)
```

Output:

```
In [10]: sns.boxplot(x='Tenure', y='Age', hue='HasCrCard', data=df_pd)

Out[10]: <AxesSubplot:xlabel='Tenure', ylabel='Age'>
```



3.2 Multivariate Analysis

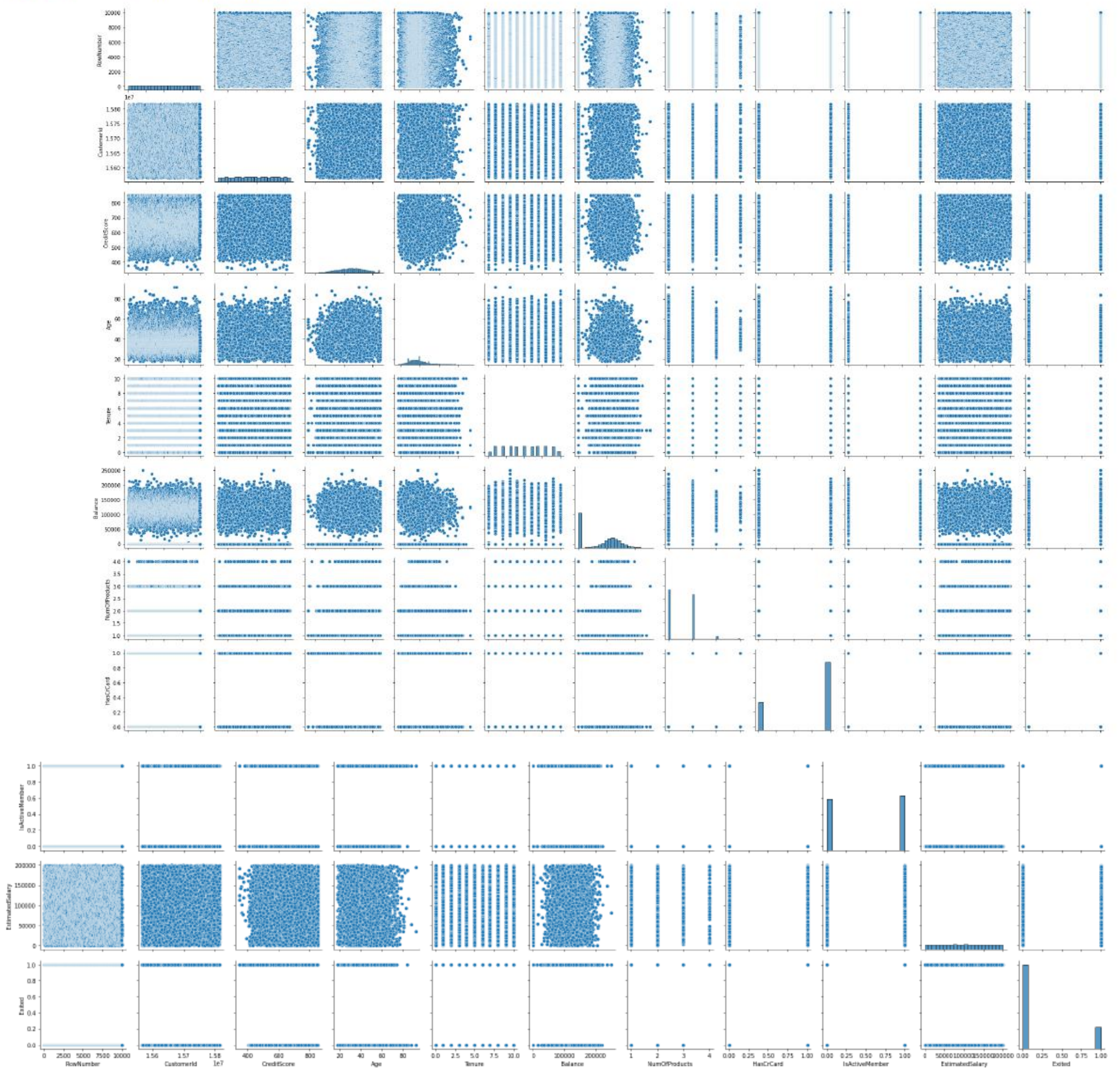
Code:

```
sns.pairplot(df_pd)
```

Output:

```
In [11]: sns.pairplot(df_pd)

Out[11]: <seaborn.axisgrid.PairGrid at 0x25d3447f6d0>
```



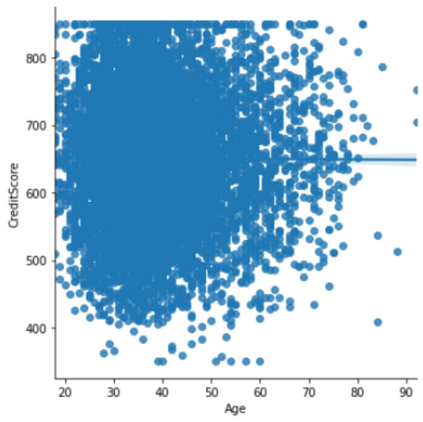
Code:

```
sns.lmplot(x='Age', y='CreditScore', data=df_pd)
```

Output:

```
In [13]: sns.lmplot(x='Age', y='CreditScore', data=df_pd)

Out[13]: <seaborn.axisgrid.FacetGrid at 0x25d3b1be6a0>
```

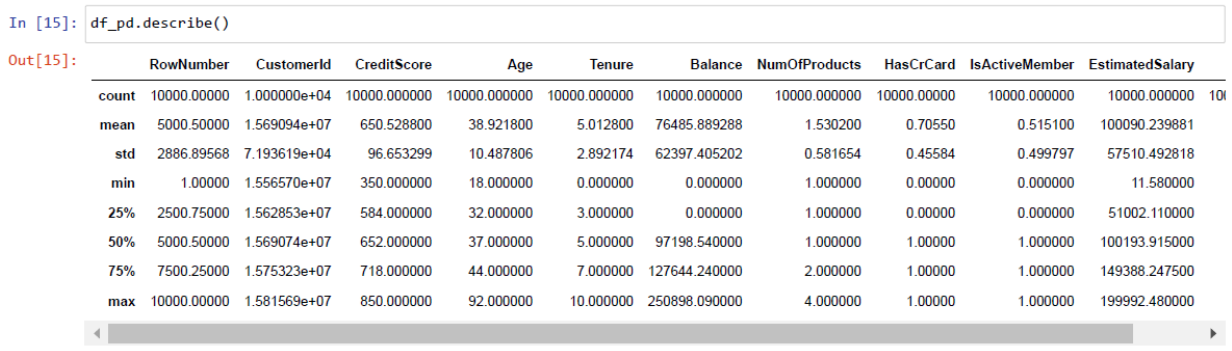


Task - 4 Perform descriptive statistics on the dataset.

Code:

```
df_pd.describe()
```

Output:



```
In [15]: df_pd.describe()

Out[15]:
```

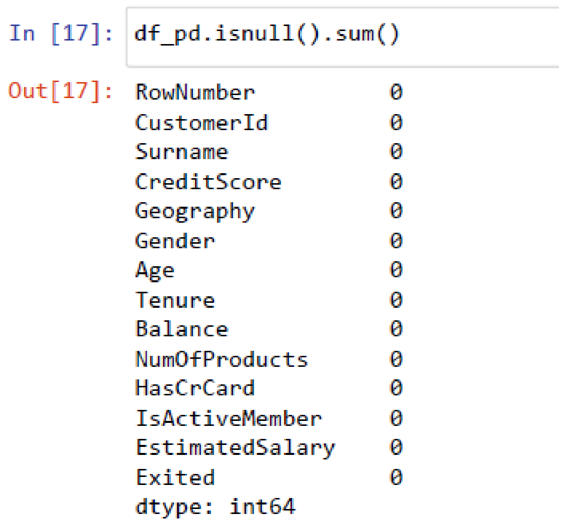
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	

Task - 5 Handle the Missing values.

Code:

```
df_pd.isnull().sum()
```

Output:



```
In [17]: df_pd.isnull().sum()

Out[17]: 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
```

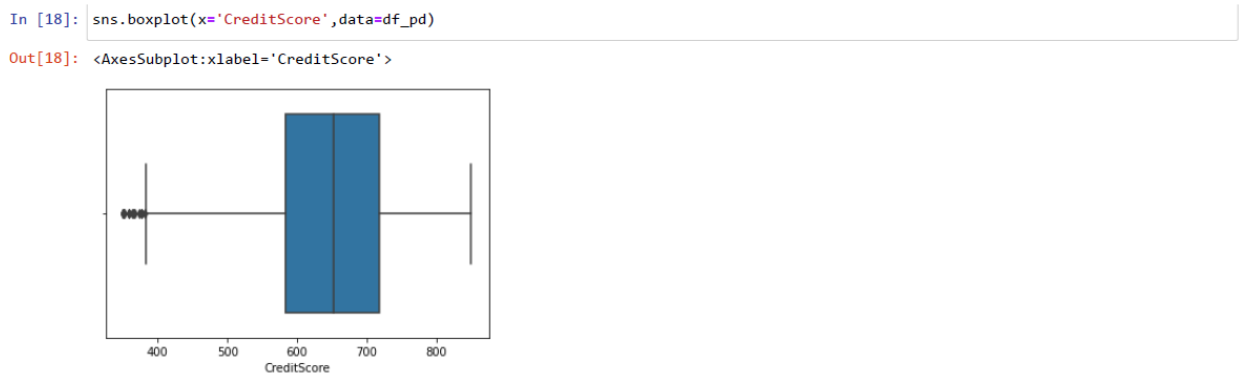
It is inferred that the data does not contain any NULL values. So there's no need to handle missing values in the dataset.

Task - 6 Find the outliers and replace the outliers

Code:

```
sns.boxplot(x='CreditScore',data=df_pd)
```

Output:



```
In [18]: sns.boxplot(x='CreditScore',data=df_pd)

Out[18]: <AxesSubplot:xlabel='CreditScore'>
```

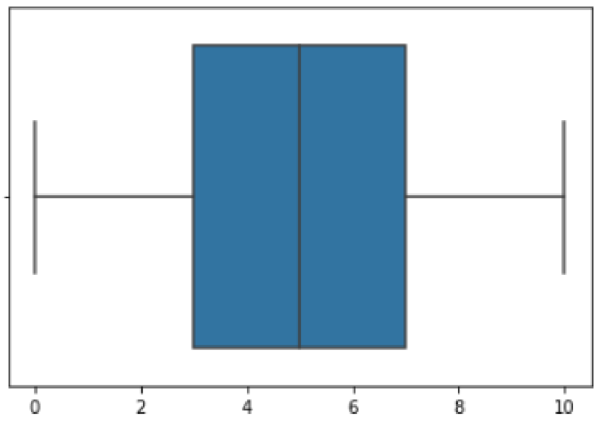
Code:

```
Q1 = df_pd['CreditScore'].quantile(0.25)
Q3 = df_pd['CreditScore'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 1.5
lower_whisker = Q1 - (whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
```

```
df_pd['CreditScore']=np.where(df_pd['CreditScore']>upper_whisker,upper_
whisker,np.where(df_pd['CreditScore']<lower_whisker,lower_whisker,df_pd
['CreditScore']))
sns.boxplot(x='Tenure',data=df_pd)
```

Output:

```
Out[21]: <AxesSubplot:xlabel='Tenure'>
```



Task - 7 Check for Categorical columns and perform encoding.

Code:

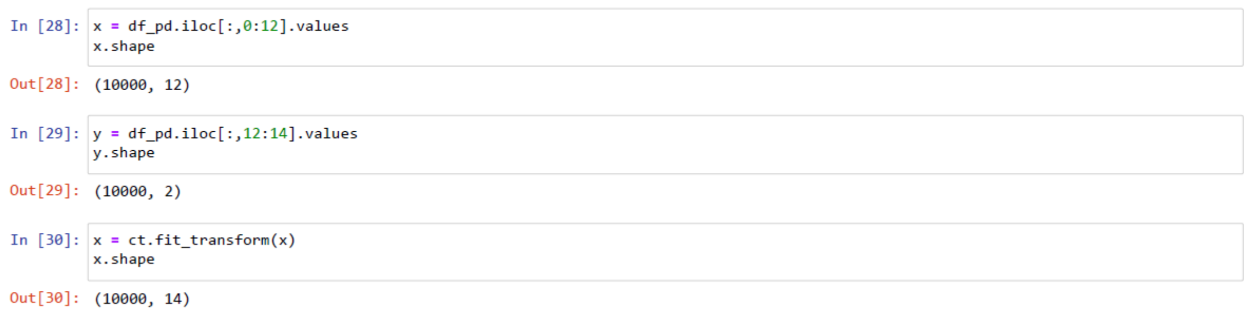
```
df_pd['Geography'].unique()
ct = ColumnTransformer([('encoder', OneHotEncoder(), [4])],
remainder="passthrough")
```

Task - 8 Split the data into dependent and independent variables.

Code:

```
x = df_pd.iloc[:,0:12].values
x.shape
y = df_pd.iloc[:,12:14].values
y.shape
x = ct.fit_transform(x)
x.shape
```

Output:



```
In [28]: x = df_pd.iloc[:,0:12].values
         x.shape

Out[28]: (10000, 12)

In [29]: y = df_pd.iloc[:,12:14].values
         y.shape

Out[29]: (10000, 2)

In [30]: x = ct.fit_transform(x)
         x.shape

Out[30]: (10000, 14)
```

Task - 9 Scale the independent variables

Code:

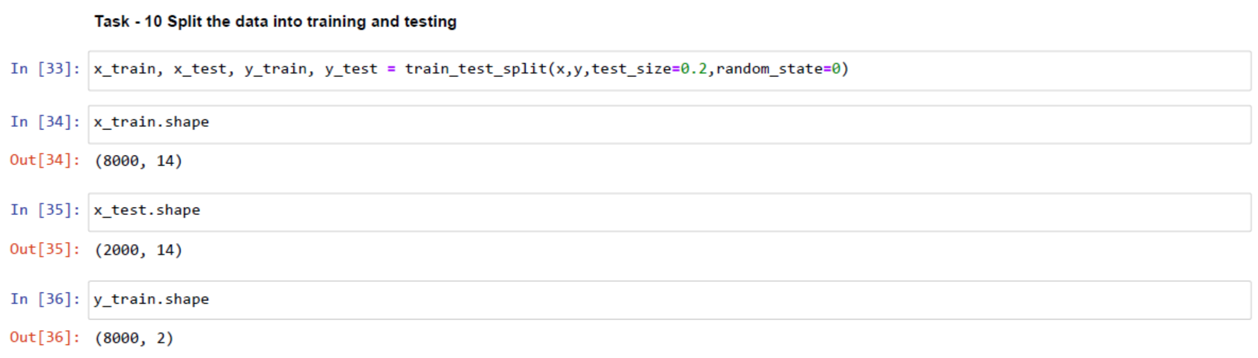
```
sc = StandardScaler()
x[:,8:12] = sc.fit_transform(x[:,8:12])
```

Task - 10 Split the data into training and testing

Code:

```
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.2,random_state=0)
x_train.shape
x_test.shape
y_train.shape
y_test.shape
```

Output:



Task - 10 Split the data into training and testing

```
In [33]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=0)

In [34]: x_train.shape

Out[34]: (8000, 14)

In [35]: x_test.shape

Out[35]: (2000, 14)

In [36]: y_train.shape

Out[36]: (8000, 2)
```