Assignment-2

VirtualEye -
Life Guard for
Swimming Pools
to Detect Active Drowning

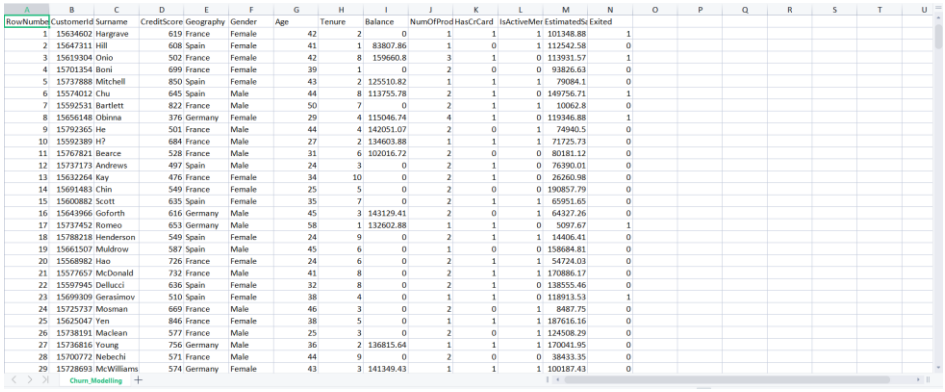
Assignment Date	November 14, 2022
Student Name	SATHEESH RATHNAVEL
Student Roll Number	2127190801073
Maximum Marks	2 Marks

Question-1:

Download the dataset

Solution:

Download the given dataset in the given attached link.



RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProd	HasCrCard	IsActiveMer	EstimatedSa	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.72	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.41	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.88	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.17	0
22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.46	0
23	15699309	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.53	1
24	15725737	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
25	15625047	Yen	846	France	Female	38	5	0	1	1	1	187616.16	0
26	15738191	Maclean	577	France	Male	25	3	0	2	0	1	124508.29	0
27	15736816	Young	756	Germany	Male	36	2	136815.64	1	1	1	170041.95	0
28	15700772	Nebechi	571	France	Male	44	9	0	2	0	0	38433.35	0
29	15728693	McWilliams	574	Germany	Female	43	3	141349.43	1	1	1	100187.43	0

Churn_Modelling

Question-2:

Load the dataset

Solution:

df=pd.read_csv('Churn_Modelling.csv')

df.head()

IMPORT THE DATA SET INTO DATAFRAME

In [2]:
```
df=pd.read_csv('Churn_Modelling.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	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:

- **Univariate analysis**
- **Bi-variate analysis**
- **Multi-variate analysis**

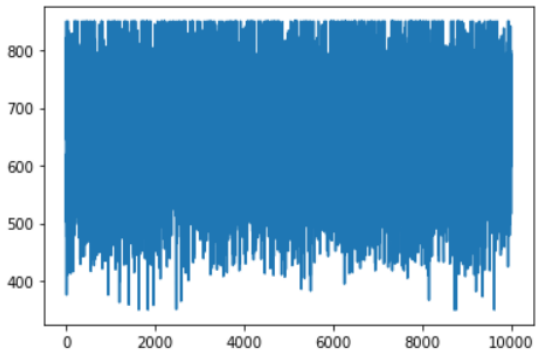
Solution:

Univariate analysis:

df.CreditScore.plot()

```
#univariate analysis
df.CreditScore.plot()
```

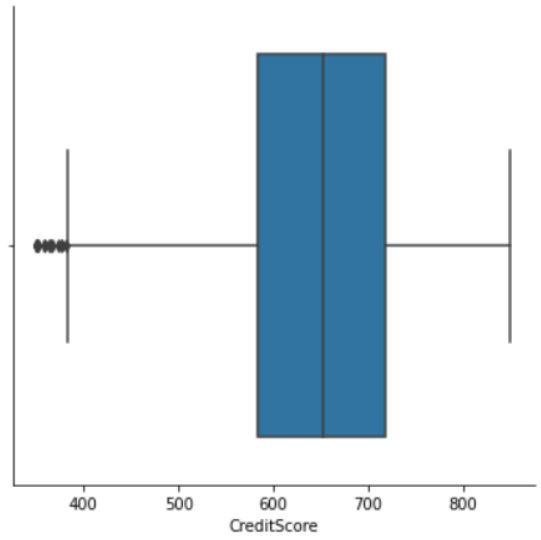
```
<AxesSubplot:>
```



sns.catplot(x='CreditScore',kind='box',data=df)

```
sns.catplot(x='CreditScore',kind='box',data=df)
```

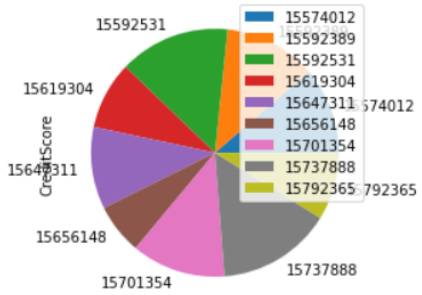
```
<seaborn.axisgrid.FacetGrid at 0x2ca156c06a0>
```



df[1:10].groupby(['CustomerId']).sum().plot(kind='pie', y='CreditScore')

```
df[1:10].groupby(['CustomerId']).sum().plot(kind='pie', y='CreditScore')
```

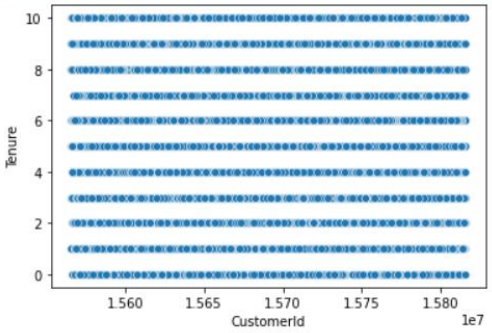
```
<AxesSubplot:ylabel='CreditScore'>
```



sns.scatterplot(df.CustomerId,df.Tenure)
plt.show()

```
sns.scatterplot(df.CustomerId,df.Tenure)
plt.show()
```

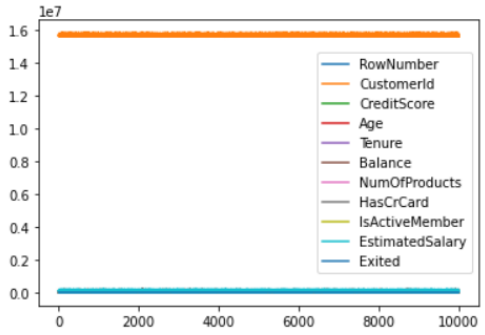
```
C:\Users\darat\AppData\Local\Programs\Python\Python36\lib\site-packages\seaborn\_decorate
rd args: x, y. From version 0.12, the only valid positional argument will be `data`, and
t in an error or misinterpretation.
  FutureWarning
```



Multivariate Analysis:
df.plot()

```
#multivariate analysis
df.plot()
```

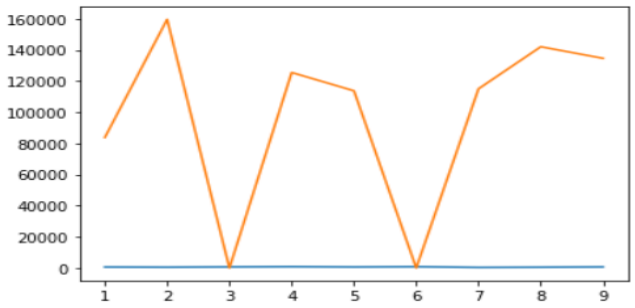
```
<AxesSubplot:>
```



Bivariate Analysis:
df.CreditScore[1:10].plot()
df.Balance[1:10].plot()

```
#bivariate analysis
df.CreditScore[1:10].plot()
df.Balance[1:10].plot()
```

```
<AxesSubplot:>
```



Question 4:

Perform descriptive statistics on the dataset.

Solution:

df.describe()

```
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().any()

```
df.isnull().any()
```

```
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
```

df.isnull().sum()

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

Question 6:

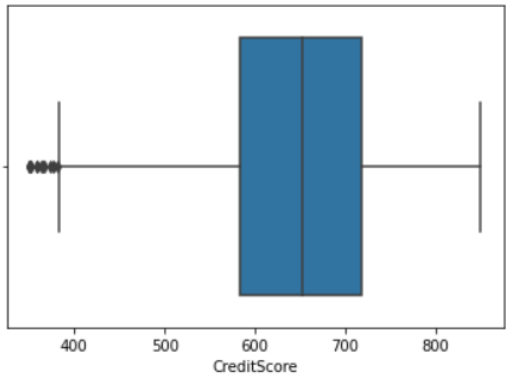
Find the outliers and replace the outliers.

Solution:

#occurence of outliers
sns.boxplot(df.CreditScore)

```
#occurence of outliers
sns.boxplot(df.CreditScore)
```

```
C:\Users\darat\AppData\Local\Programs\Python\Python36\lib\site-packa
ord arg: x. From version 0.12, the only valid positional argument w
n an error or misinterpretation.
  FutureWarning
<AxesSubplot:xlabel='CreditScore'>
```



Q1= df.CreditScore.quantile(0.25)
Q3=df.CreditScore.quantile(0.75)

IQR=Q3-Q1

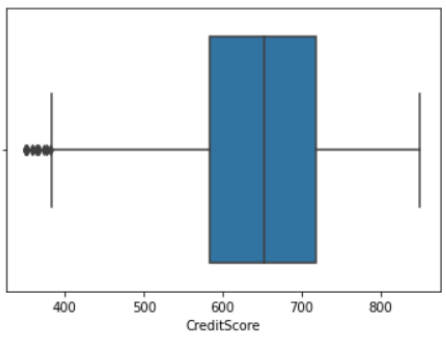
upper_limit =Q3 + 1.5*IQR
lower_limit =Q1 - 1.5*IQR

df['CreditScore'] = np.where(df['CreditScore']>upper_limit,30,df['CreditScore'])

sns.boxplot(df.CreditScore)

```
sns.boxplot(df.CreditScore)
```

```
C:\Users\darat\AppData\Local\Programs\Python\Python36\lib\site-packages\
ord arg: x. From version 0.12, the only valid positional argument will b
n an error or misinterpretation.
  FutureWarning
<AxesSubplot:xlabel='CreditScore'>
```



Question 7:

Check for Categorical columns and perform encoding.

Solution:

```
#label encoder
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df.Gender= le.fit_transform(df.Gender)

df.head(5)
```

```
df.head(5)
```

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

```
#one hot encoding
df_main=pd.get_dummies(df,columns=['Geography'])
df_main.head()
```

```
#one hot encoding
df_main=pd.get_dummies(df,columns=['Geography'])
df_main.head()
```

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

Question 8:

Split the data into dependent and independent variables.

Solution:

X=df_main.drop(columns=['EstimatedSalary'],axis=1)

X.head()

X_scaled=pd.DataFrame(scale(X),columns=X.columns)

X_scaled.head()

```
X=df_main.drop(columns=['EstimatedSalary'],axis=1)
X.head()
X_scaled=pd.DataFrame(scale(X),columns=X.columns)
X_scaled.head()
```

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germar
0	-1.731878	-0.783213	-0.326221	-1.095988	0.293517	-1.041760	-1.225848	-0.911583	0.646092	0.970243	1.977165	0.997204	-0.5787
1	-1.731531	-0.606534	-0.440036	-1.095988	0.198164	-1.387538	0.117350	-0.911583	-1.547768	0.970243	-0.505775	-1.002804	-0.5787
2	-1.731185	-0.995885	-1.536794	-1.095988	0.293517	1.032908	1.333053	2.527057	0.646092	-1.030670	1.977165	0.997204	-0.5787
3	-1.730838	0.144767	0.501521	-1.095988	0.007457	-1.387538	-1.225848	0.807737	-1.547768	-1.030670	-0.505775	0.997204	-0.5787
4	-1.730492	0.652659	2.063884	-1.095988	0.388871	-1.041760	0.785728	-0.911583	0.646092	0.970243	-0.505775	-1.002804	-0.5787

y=df_main.EstimatedSalary
y

```
y=df_main.EstimatedSalary
y
```

```
0       101348.88
1       112542.58
2       113931.57
3        93826.63
4        79084.10
          ...
9995     96270.64
9996    101699.77
9997     42085.58
9998     92888.52
9999     38190.78
Name: EstimatedSalary, Length: 10000, dtype: float64
```

Question 9:

Scale the independent variables.

Solution:

from sklearn.preprocessing import scale

X_scaled=pd.DataFrame(scale(X),columns=X.columns)

X_scaled.head()

```
X_scaled=pd.DataFrame(scale(X),columns=X.columns)
X_scaled.head()
```

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germar
0	-1.731878	-0.783213	-0.326221	-1.095988	0.293517	-1.041760	-1.225848	-0.911583	0.646092	0.970243	1.977165	0.997204	-0.5787
1	-1.731531	-0.606534	-0.440036	-1.095988	0.198164	-1.387538	0.117350	-0.911583	-1.547768	0.970243	-0.505775	-1.002804	-0.5787
2	-1.731185	-0.995885	-1.536794	-1.095988	0.293517	1.032908	1.333053	2.527057	0.646092	-1.030670	1.977165	0.997204	-0.5787
3	-1.730838	0.144767	0.501521	-1.095988	0.007457	-1.387538	-1.225848	0.807737	-1.547768	-1.030670	-0.505775	0.997204	-0.5787
4	-1.730492	0.652659	2.063884	-1.095988	0.388871	-1.041760	0.785728	-0.911583	0.646092	0.970243	-0.505775	-1.002804	-0.5787

Question 10:

Split the data into training and testing.

Solution:

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)

```
X_train.shape
```

```
(7000, 14)
```

```
X_test.shape
```

```
(3000, 14)
```

```
y_train.shape
```

```
(7000,)
```

```
y_test.shape
```

```
(3000,)
```