Assignment -2
Visualization and preprocessing

Assignment Date	29 September 2022
Student Name	Clifford Lebo J
Student Roll Number	CITC1907005
Maximum Marks	2 Marks

Question 1. Download the dataset: Dataset

1. Importing required libraries

In [1]:
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline
```

Question 2. Load the dataset.

```
# 2. Load the dataset
df=pd.read_csv("Churn_Modelling.csv")
df
```

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

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	10134
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	11254
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	11393
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	9382
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	7908
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	9627
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	3819

10000 rows × 14 columns

Question 3. Perform Below Visualizations.

- **Univariate Analysis**

Code:

df.dtypes

df['Age'].value_counts()

sns.kdeplot(df['Age'])

3.1.univatiate Analysis

In [3]:
```
df.dtypes
```

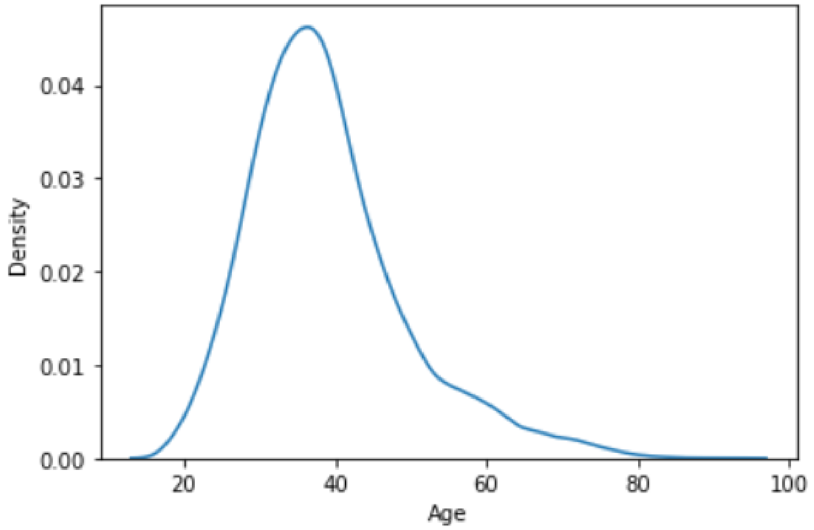
Out[3]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [4]:
```
df['Age'].value_counts()
```

Out[4]:
```
37    478
38    477
35    474
36    456
34    447
     ...
92      2
82      1
88      1
85      1
83      1
Name: Age, Length: 70, dtype: int64
```

```
sns.kdeplot(df['Age'])
```

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



- **Bi - Variate Analysis**

Code:

#1.

df.corr()

#2.

import seaborn as sns

sns.heatmap(df.corr())

#3.

import statsmodels.api as sm

#define response variable

y = df['Age']

#define explanatory variable

x = df[['Exited']]

#add constant to predictor variables

x = sm.add_constant(x)

#fit linear regression model

model = sm.OLS(y, x).fit()

#view model summary

print(model.summary())

3.2.Bi-Variate Analysis

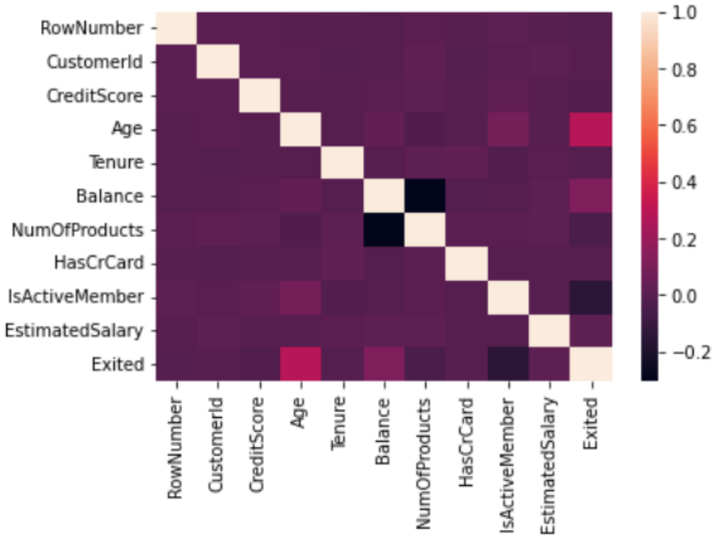
In [6]:
```
df.corr()
```

Out[6]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exitec
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

In [7]:
```
import seaborn as sns
sns.heatmap(df.corr())
```

Out[7]: <AxesSubplot:>



```
In [8]: import statsmodels.api as sm
#define response variable
y = df['Age']

#define explanatory variable
x = df[['Exited']]

#add constant to predictor variables
x = sm.add_constant(x)

#fit linear regression model
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Age   R-squared:                       0.081
Model:                            OLS   Adj. R-squared:                  0.081
Method:                 Least Squares   F-statistic:                     886.1
Date:                Sun, 02 Oct 2022   Prob (F-statistic):          1.24e-186
Time:                        22:44:02   Log-Likelihood:                -37266.
No. Observations:               10000   AIC:                         7.454e+04
Df Residuals:                    9998   BIC:                         7.455e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         37.4084      0.113    332.078      0.000      37.188      37.629
Exited         7.4296      0.250     29.767      0.000       6.940       7.919
==============================================================================
Omnibus:                     1974.048   Durbin-Watson:                   2.027
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4381.188
Skew:                           1.136   Prob(JB):                         0.00
Kurtosis:                       5.314   Cond. No.                         2.60
==============================================================================

Notes:
```

- **Multi - Variate Analysis**

<u>Code</u>

sns.pairplot(data=df[['CustomerId', 'CreditScore','Gender', 'Age', 'Tenure', 'Geography', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember' ,'EstimatedSalary', 'Exited']],hue='Geography')

Question 4. Perform descriptive statistics on the dataset.

Code:

```
df.describe()
```

```
In [11]: df.describe()
```

Out[11]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumC
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	100
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	

Question 5. Handle the Missing values.

Code:
df.fillna(df.mean())

```
In [13]: df.fillna(df.mean())

C:\Users\Cliff\AppData\Local\Temp/ipykernel_24992/634187881.py:1: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  df.fillna(df.mean())
```

Out[13]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8
...	...	...	...	...	...	...	...	...	.
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.6
9997	9998	15584532	Liu	709	France	Female	36	7	0.0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.3
9999	10000	15628319	Walker	792	France	Female	28	4	130142.7

10000 rows × 14 columns

Question 6. Find the outliers and replace the outliers

Code:

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

df["Tenure"]

```
In [14]: df["Tenure"] = np.where(df["Tenure"] >10, np.median,df["Tenure"])
         df["Tenure"]

Out[14]: 0        2
         1        1
         2        8
         3        1
         4        2
                 ..
         9995     5
         9996    10
         9997     7
         9998     3
         9999     4
         Name: Tenure, Length: 10000, dtype: object
```

Question 7. Check for Categorical columns and perform encoding.

Code:

```
x=list(df.columns)
for i in x:
    print(pd.Categorical(df[i]))
    print("\n\n\n")
```

```
In [22]: x=list(df.columns)
         for i in x:
             print(pd.Categorical(df[i]))
             print("\n\n\n")
```

```
[1, 1, 3, 2, 1, ..., 2, 1, 1, 2, 1]
Length: 10000
Categories (4, int64): [1, 2, 3, 4]




[1, 0, 1, 0, 1, ..., 1, 1, 0, 1, 1]
Length: 10000
Categories (2, int64): [0, 1]




[1, 1, 0, 0, 1, ..., 0, 1, 1, 0, 0]
Length: 10000
Categories (2, int64): [0, 1]
```

Question 8. Split the data into dependent and independent variables.

Code:

```
dependent=df[x[:2]]
independent=df[x[2:]]
print("dependent variables\n",dependent.head())
print("\n\nindependent variables\n",independent.head())
```

```
In [16]: dependent=df[x[:2]]
         independent=df[x[2:]]
```

```
In [17]: print("dependent variables\n",dependent.head())
         print("\n\nindependent variables\n",independent.head())
```

```
dependent variables
    RowNumber  CustomerId
0          1    15634602
1          2    15647311
2          3    15619304
3          4    15701354
4          5    15737888


independent variables
     Surname  CreditScore Geography  Gender  Age  Tenure    Balance  \
0  Hargrave          619    France  Female   42       2       0.00
1      Hill          608     Spain  Female   41       1   83807.86
2      Onio          502    France  Female   42       8  159660.80
3      Boni          699    France  Female   39       1       0.00
4  Mitchell          850     Spain  Female   43       2  125510.82

   NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  Exited
0              1          1               1        101348.88       1
1              1          0               1        112542.58       0
2              3          1               0        113931.57       1
3              2          0               0         93826.63       0
4              1          1               1         79084.10       0
```

Question 9. Scale the independent variables

Code:

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df[["RowNumber"]] = scaler.fit_transform(df[["RowNumber"]])

print(df.head())

```
In [18]: import pandas as pd
         from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler()
         df[["RowNumber"]] = scaler.fit_transform(df[["RowNumber"]])
         print(df.head())
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  Tenure  \
0     0.0000    15634602  Hargrave          619    France  Female   42       2
1     0.0001    15647311      Hill          608     Spain  Female   41       1
2     0.0002    15619304      Onio          502    France  Female   42       8
3     0.0003    15701354      Boni          699    France  Female   39       1
4     0.0004    15737888  Mitchell          850     Spain  Female   43       2

     Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  \
0       0.00              1          1               1        101348.88
1   83807.86              1          0               1        112542.58
2  159660.80              3          1               0        113931.57
3       0.00              2          0               0         93826.63
4  125510.82              1          1               1         79084.10

   Exited
0       1
1       0
2       1
3       0
4       0
```

Question 10. Split the data into training and testing

Code:

from sklearn.model_selection import train_test_split

train_size=0.8

X = df.drop(columns = ['Tenure']).copy()

y = df['Tenure']

X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)

test_size = 0.5

X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5)

print(X_train.shape), print(y_train.shape)

print(X_valid.shape), print(y_valid.shape)

print(X_test.shape), print(y_test.shape)

In [19]:
```python
from sklearn.model_selection import train_test_split
train_size=0.8
X = df.drop(columns = ['Tenure']).copy()
y = df['Tenure']
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5)
print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)
```

```
(8000, 13)
(8000,)
(1000, 13)
(1000,)
(1000, 13)
(1000,)
```

Out[19]:
```
(None, None)
```