UNIVERSITY ADMIT ELIGIBILITY PREDICTOR

ASSIGNMENT - 2

Date	26th September 2022
Team ID	PNT2022TMID27839
Student Name	Prem B (311519104047)
Domain Name	Education
Project Name	University Admit Eligibility Predictor
Maximum Marks	2 Marks

1.)IMPORT THE REQUIRED LIBRARIES

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

2.)DOWNLOAD AND UPLOAD THE DATASET

Download and Upload the Dataset

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

Out[2]:

	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

3.)HANDLE MISSING VALUES IN THE DATASET

Handle the Missing Values in the Dataset

In [3]:
```
#Removing Unwanted Values
df = df.drop(columns=['RowNumber','CustomerId','Surname'])
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [5]:
```
df.shape
```

Out[5]:
```
(10000, 11)
```

4.) PERFORM THE DESCRIPTIVE STATISTICS ON THE DATASET

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

Out[6]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

In [7]:
```
df.info()
```

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

5.) PERFORM VARIOUS VISUALISATIONS

a.) UNIVARIANTE ANALYSIS

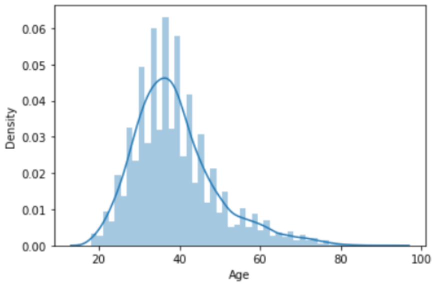
Perform Various Visualisations

Univariante Analysis

In [8]:
```
sns.distplot(df.Age)
```

```
C:\Users\Prem\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

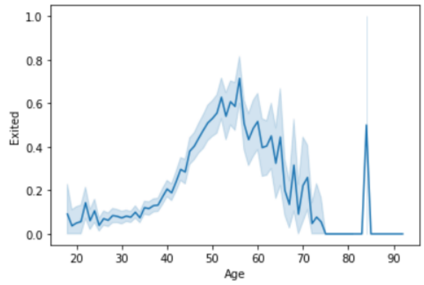
Out[8]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



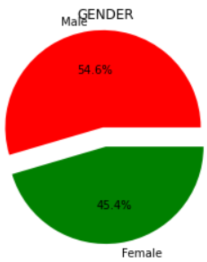
In [9]:
```
sns.lineplot(df.Age,df.Exited)
```

```
C:\Users\Prem\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]: `<AxesSubplot:xlabel='Age', ylabel='Exited'>`



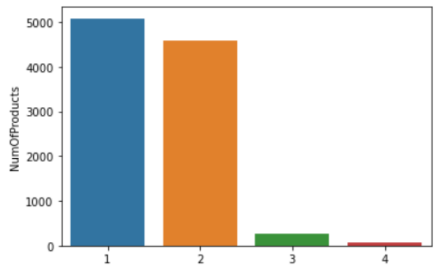
In [10]:
```
plt.pie(df.Gender.value_counts(),[0.2,0],colors=['red','green'],labels=['Male','Female'],autopct='%1.1f%%')
plt.title('GENDER')
plt.show()
```



In [11]:
```
sns.barplot(df.NumOfProducts.value_counts().index,df.NumOfProducts.value_counts())
```

```
C:\Users\Prem\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[11]: `<AxesSubplot:ylabel='NumOfProducts'>`



b.) BI - VARIANTE ANALYSIS

Bi - Variante Analysis

In [12]:
```
def countplot_2(x,hue,title=None,figsize=(6,5)):
  plt.figure(figsize=figsize)
  sns.countplot(data=df[[x,hue]],x=x,hue=hue)
  plt.title(title)
  plt.show()
```

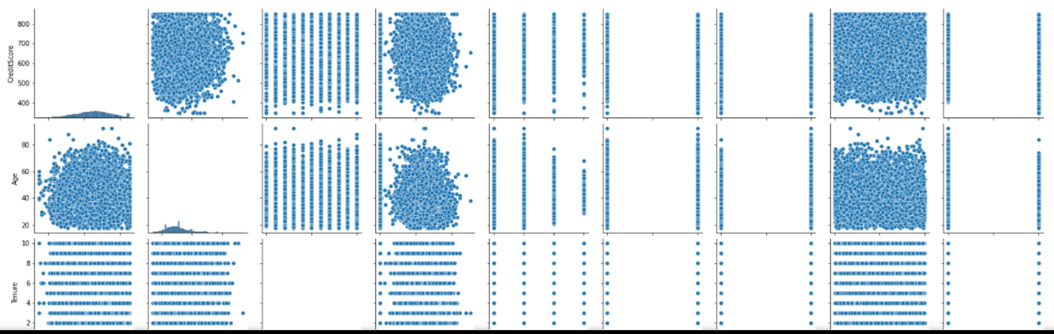
In [13]:
```
countplot_2('IsActiveMember','NumOfProducts','Credit Card Holders Product Details')
```

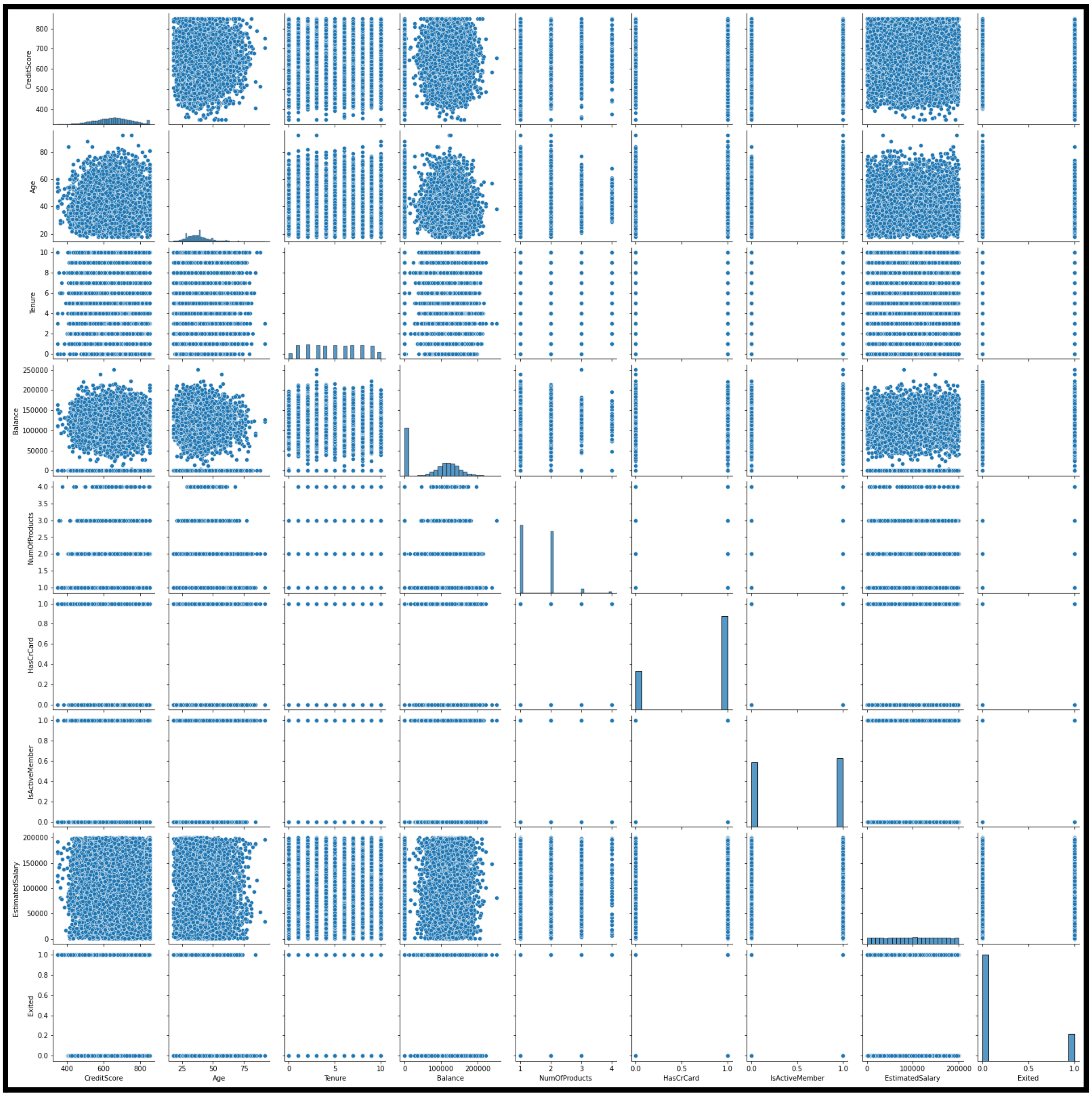
c.) MULTI - VARIANTE ANALYSIS

Multi - Variante Analysis

In [14]:
```
sns.pairplot(df)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1c642d93760>
```






```
In [15]: df.corr()
```

Out[15]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

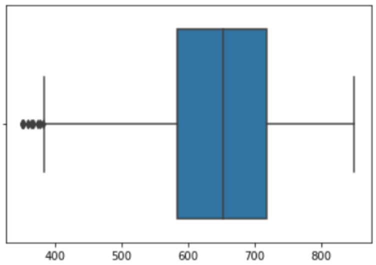
6.) FIND AND REPLACE THE OUTLIERS

Find the Outliers

In [18]:
```
sns.boxplot(df.CreditScore)
```

```
keyword will result in an error or misinterpretation.
  warnings.warn(
```

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



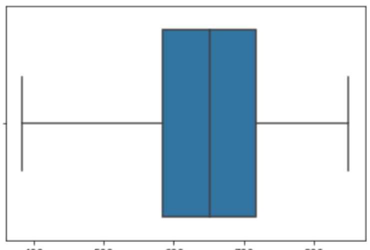
Replace the Outliers

In [19]:
```
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)
```

In [20]:
```
df['CreditScore'] = np.where(df['CreditScore']<lower_limit,650,df['CreditScore'])
sns.boxplot(df.CreditScore)
```

```
C:\Users\Prem\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]: <AxesSubplot:xlabel='CreditScore'>



7.) CHECK FOR CATEGORICAL COLUMNS AND ENCODE THEM

Check for Categorical Columns and Perform Encoding

In [21]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df.Geography = le.fit_transform(df.Geography)
df.Gender = le.fit_transform(df.Gender)
```

In [22]:
```
df.head()
```

Out[22]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0

8.)SPLIT DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

Split the data into Dependent and Independent Variables

In [23]:
```
X = df.drop(columns=['Exited'])
X.head()
```

Out[23]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	0	0	42	2	0.00	1	1	1	101348.88
1	608	2	0	41	1	83807.86	1	0	1	112542.58
2	502	0	0	42	8	159660.80	3	1	0	113931.57
3	699	0	0	39	1	0.00	2	0	0	93826.63
4	850	2	0	43	2	125510.82	1	1	1	79084.10

In [24]:
```
Y = df.Exited
Y.head()
```

Out[24]:
```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

9.) SCALE THE INDEPENDENT VARIABLES

Scale the Independent Variables

In [25]:
```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
X_scaled = pd.DataFrame(scale.fit_transform(X),columns=X.columns)
```

10.)SPLIT THE DATA INTO TRAINING AND TESTING

Split the data into Training and Testing

In [26]:
```
from sklearn.model_selection import train_test_split
x_train , y_train , x_test , y_test = train_test_split(X_scaled,Y,test_size=0.2,random_state=0)
```

In [27]:
```
X_scaled.shape
```

Out[27]:
```
(10000, 10)
```

In [28]:
```
x_train.shape
```

Out[28]:
```
(8000, 10)
```