

# TEAM MEMBER-GIDEON M(REG No: 312319106043)

In [1]:

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

```
df = pd.read_csv(r"./Churn_Modelling.csv")
```

```
df.head()
```

| RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance |
|-----------|------------|---------|-------------|-----------|--------|-----|--------|---------|
|-----------|------------|---------|-------------|-----------|--------|-----|--------|---------|

## 2. Load the data set

In [4]:

In [5]:

Out[5]:N

|   |   |          |          |     |        |        |    |   |           |
|---|---|----------|----------|-----|--------|--------|----|---|-----------|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00      |
| 1 | 2 | 15647311 | Hill     | 608 | Spain  | Female | 41 | 1 | 83807.86  |
| 2 | 3 | 15619304 | Onio     | 502 | France | Female | 42 | 8 | 159660.80 |
| 3 | 4 | 15701354 | Boni     | 699 | France | Female | 39 | 1 | 0.00      |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain  | Female | 43 | 2 | 125510.82 |

## 3. Data Visualizations

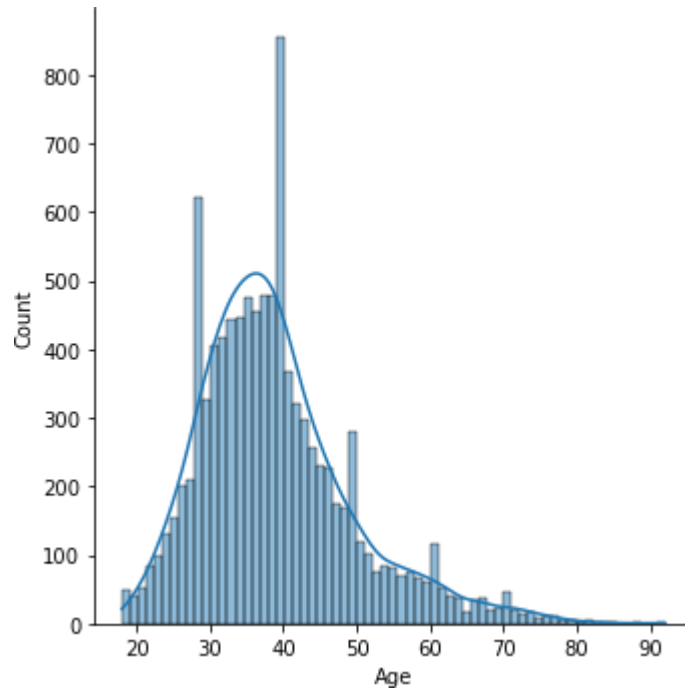
### 3.1. Univariate Analysis

In [6]:

```
sns.displot(df['Age'], kde=True)
```

<seaborn.axisgrid.FacetGrid at 0x1f63a02fa30>

Out[6]:



### 3.2. Bi - Variate Analysis

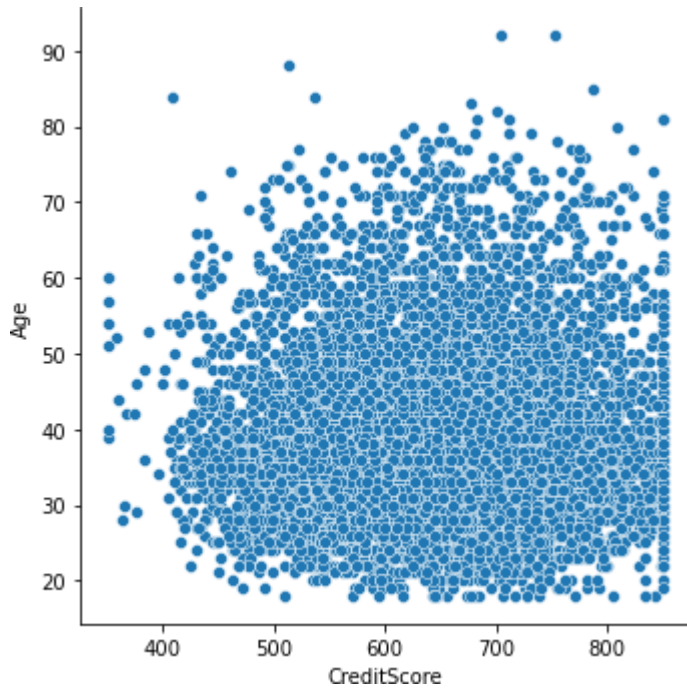
In [7]:

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

<seaborn.axisgrid.FacetGrid at 0x1f63a024160>

Out[7]:

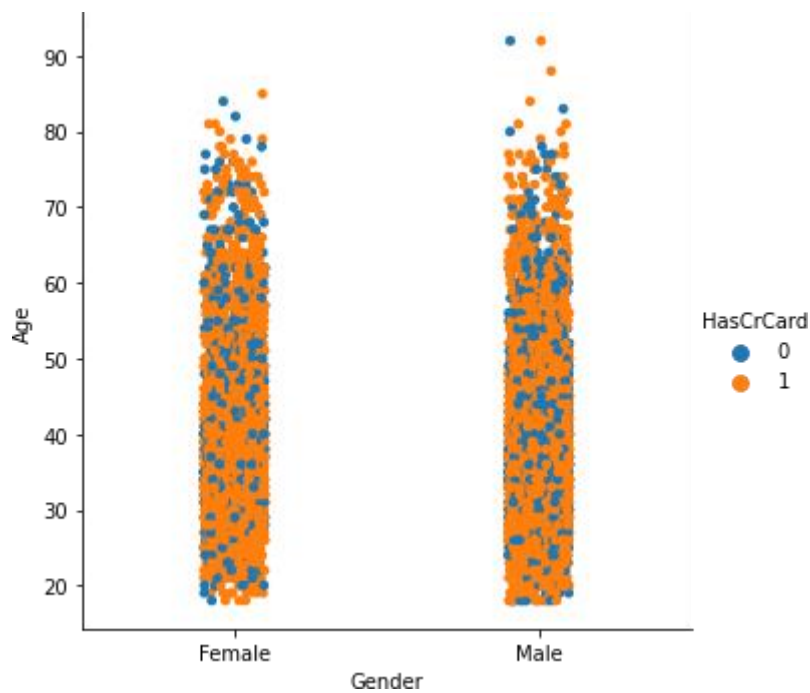
In [8]:



```
sns.catplot(x='Gender', y='Age', hue='HasCrCard', data=df)
```

<seaborn.axisgrid.FacetGrid at 0x1f647affeb0>

Out[8]:



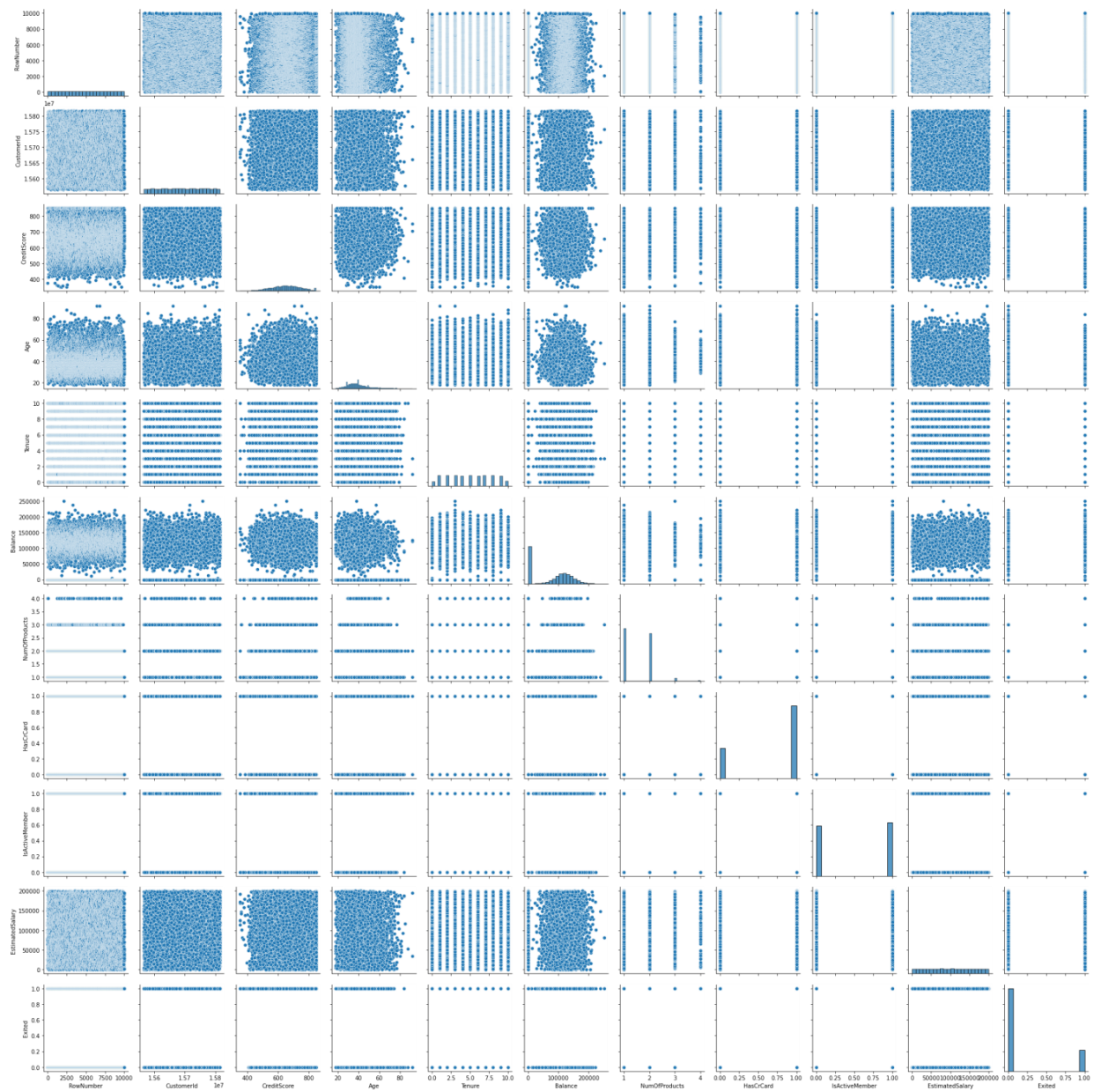
### 3.3. Multi - Variate Analysis

In [9]:

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

<seaborn.axisgrid.PairGrid at 0x1f6483b69a0>

Out[9]:

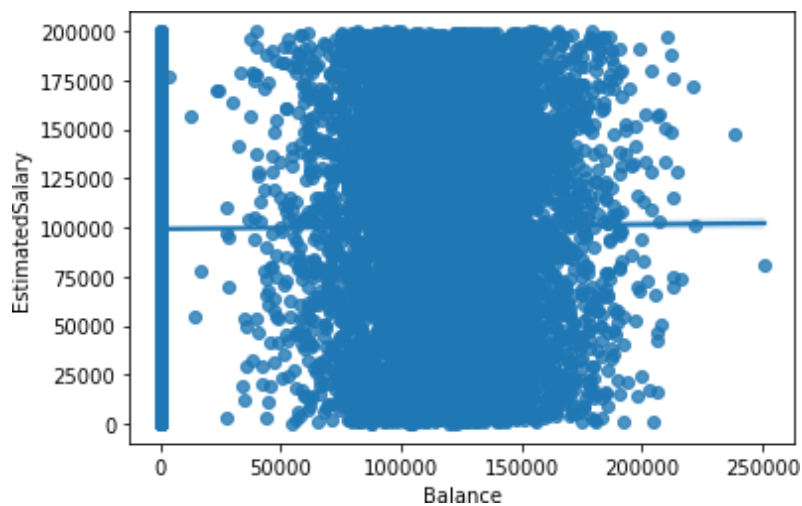


In [10]:

```
sns.regplot(x='Balance', y='EstimatedSalary', data=df)
```

```
<AxesSubplot:xlabel='Balance', ylabel='EstimatedSalary'>
```

Out[10]:



## 4. Descriptive Statistics

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

```
Out[11]:
```

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

## 5. Handle the Missing values

```
In [12]:
```

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

```
Out[12]:
```

|                 |   |
|-----------------|---|
| 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

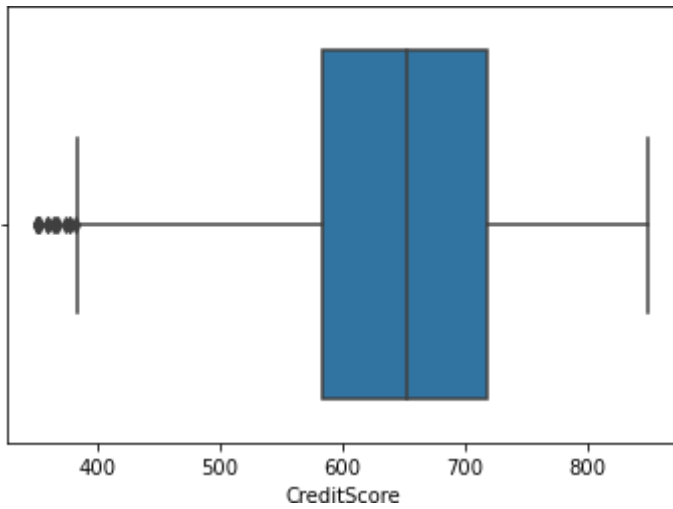
## 6. Find the outliers and replace the outliers

```
In [13]:
```

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

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

```
Out[13]:
```



In [14]:

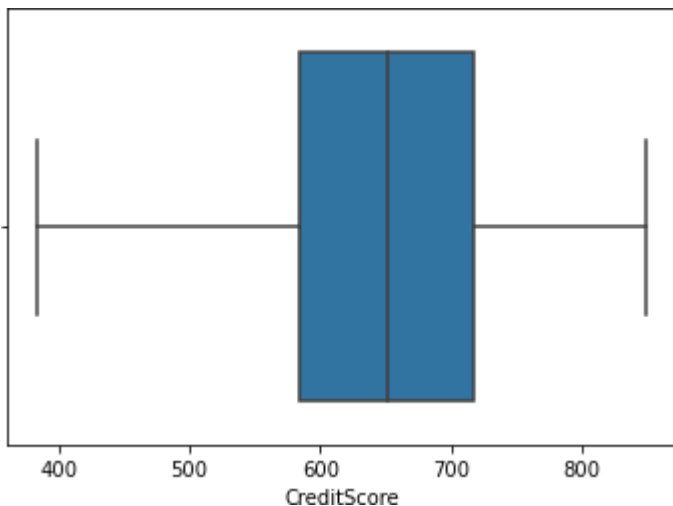
```
Q1 = df['CreditScore'].quantile(0.25)
Q3 = df['CreditScore'].quantile(0.75)
IQR = Q3 - Q1 whisker_width = 1.5
lower_whisker = Q1 - (whisker_width*IQR) upper_whisker = Q3 + (whisker_width*IQR)
df['CreditScore'] = np.where(df['CreditScore'] > upper_whisker, upper_whisker, np.where(df
```

In [15]:

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

<AxesSubplot:xlabel='CreditScore'>

Out[15]:



## 7. Check for Categorical columns and perform encoding

```
In [16]: df['Geography'].unique()
ct= ColumnTransformer([('oh', OneHotEncoder(), [4])],
                      remainder="passthrough")
```

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

```
In [17]: x=df.iloc[:,0:12].values
y=df.iloc[:,12:14].value
s x[0:5,:]
```

```
array([[1, 15634602, 'Hargrave', 619.0, 'France', 'Female', 42, 2, 0.0,
Out[17]:
      1, 1, 1],
      [2, 15647311, 'Hill', 608.0, 'Spain', 'Female', 41, 1, 83807.86,
      1, 0, 1],
      [3, 15619304, 'Onio', 502.0, 'France', 'Female', 42, 8, 159660.8,
      3, 1, 0],
      [4, 15701354, 'Boni', 699.0, 'France', 'Female', 39, 1, 0.0, 2, 0,
      0],
      [5, 15737888, 'Mitchell', 850.0, 'Spain', 'Female', 43, 2,
      125510.82, 1, 1, 1]], dtype=object)
```

```
In [18]:
      x=ct.fit_transform(
      x)
#INDEPENDENT VARIABLES
      x[0:5,:]
```

```
array([[1.0, 0.0, 0.0, 1, 15634602, 'Hargrave', 619.0, 'Female', 42, 2,
Out[18]:
      0.0, 1, 1, 1],
      [0.0, 0.0, 1.0, 2, 15647311, 'Hill', 608.0, 'Female', 41, 1,
      83807.86, 1, 0, 1],
      [1.0, 0.0, 0.0, 3, 15619304, 'Onio', 502.0, 'Female', 42, 8,
      159660.8, 3, 1, 0],
      [1.0, 0.0, 0.0, 4, 15701354, 'Boni', 699.0, 'Female', 39, 1, 0.0,
      2, 0, 0],
      [0.0, 0.0, 1.0, 5, 15737888, 'Mitchell', 850.0, 'Female', 43, 2,
      125510.82, 1, 1, 1]], dtype=object)
```

```
In [19]:
      #DEPENDENT VARIABLES
      y[0:5,:]
```

```
array([[1.0134888e+05, 1.0000000e+00],
Out[19]:
      [1.1254258e+05, 0.0000000e+00],
      [1.1393157e+05, 1.0000000e+00],
      [9.3826630e+04, 0.0000000e+00],
      [7.9084100e+04, 0.0000000e+00]])
```

```
sc= StandardScaler()
x[:,8:12]=sc.fit_transform(x[:,8:12])
x[0:5,:]
```

## 9. Scale the independent variables

```
In [20]:
```

```
array([[1.0, 0.0, 0.0, 1, 15634602, 'Hargrave', 619.0, 'Female',
Out[20]:
      0.29351742289674765, -1.041759679225302, -1.2258476714090163,
      -0.911583494040172, 1, 1],
      [0.0, 0.0, 1.0, 2, 15647311, 'Hill', 608.0, 'Female',
      0.19816383219544578, -1.387537586562431, 0.11735002143511637,
      -0.911583494040172, 0, 1],
```

```
[1.0, 0.0, 0.0, 3, 15619304, 'Onio', 502.0, 'Female',
0.29351742289674765, 1.0329077647974714, 1.333053345722891,
2.5270566192762067, 1, 0],
[1.0, 0.0, 0.0, 4, 15701354, 'Boni', 699.0, 'Female',
0.007456650792842043, -1.387537586562431, -1.2258476714090163,
0.8077365626180174, 0, 0], [0.0, 0.0, 1.0, 5,
15737888, 'Mitchell', 850.0, 'Female',
0.3888710135980495, -1.041759679225302, 0.7857278997960621,
-0.911583494040172, 1, 1]], dtype=object)
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,
random_state=
```

## 10. Split the data into training and testing

In [21]:

```
x_train
```

In [22]:

```
array([[1.0, 0.0, 0.0, ..., 0.8077365626180174, 1, 1],
Out[22]:
```

```
[1.0, 0.0, 0.0, ..., 0.8077365626180174, 1, 0],
[1.0, 0.0, 0.0, ..., -0.911583494040172, 0, 1],
...,
[1.0, 0.0, 0.0, ..., 0.8077365626180174, 1, 0],
[0.0, 0.0, 1.0, ..., 0.8077365626180174, 1, 1],
[0.0, 1.0, 0.0, ..., -0.911583494040172, 1, 0]], dtype=object)
```

In [23]:

```
x_test
```

```
array([[0.0, 1.0, 0.0, ..., -0.911583494040172, 1, 1],
Out[23]:
```

```
[1.0, 0.0, 0.0, ..., -0.911583494040172, 1, 0],
[0.0, 0.0, 1.0, ..., -0.911583494040172, 1, 1],
...,
[1.0, 0.0, 0.0, ..., 0.8077365626180174, 1, 1],
[1.0, 0.0, 0.0, ..., -0.911583494040172, 1, 1],
[0.0, 1.0, 0.0, ..., -0.911583494040172, 1, 1]], dtype=object)
```

In [24]:

```
y_train
```

```
array([[5.5796830e+04, 1.0000000e+00],
Out[24]:
```

```
[1.9823020e+04, 0.0000000e+00],
[1.3848580e+04, 0.0000000e+00],
...,
[1.8142987e+05, 0.0000000e+00],
[1.4875016e+05, 0.0000000e+00],
[1.1885526e+05, 1.0000000e+00]])
```



In [25]:

```
y_test
```

Out[25]:

```
array([[1.9285267e+05, 0.0000000e+00],  
       [1.2870210e+05, 1.0000000e+00],  
       [7.5732250e+04, 0.0000000e+00],  
       ...,  
       [1.6740029e+05, 0.0000000e+00],  
       [7.0849470e+04, 0.0000000e+00],  
       [3.3759410e+04, 1.0000000e+00]])
```