19IT004 Assignment-2

Data Visualization and Pre-processing

Import libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Load dataset

```
data = pd.read_csv('Churn_Modelling.csv')
```

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838
2	3	15619304	Onio	502	France	Female	42	8	1596
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	1255

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
```

```
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

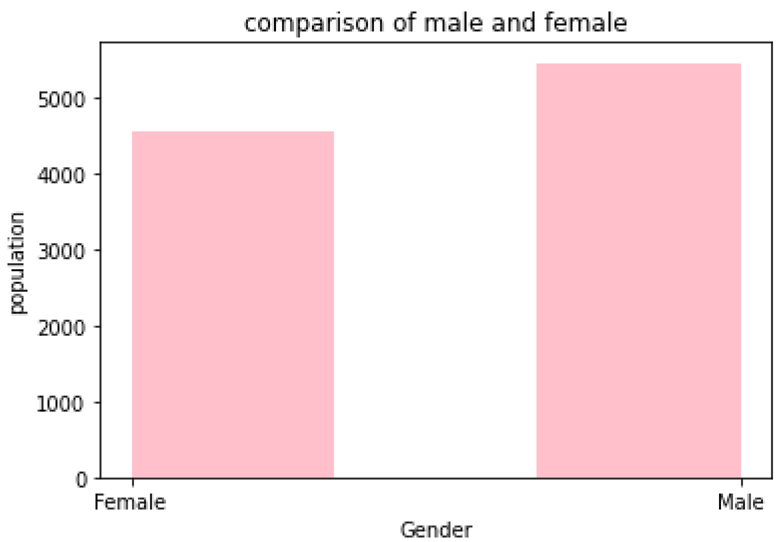
Visualisations

1. Univariate Analysis

```
data['Gender'].value_counts()
```

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

```
# Plotting the features of the dataset to see the correlation between them

plt.hist(x = data.Gender, bins = 3, color = 'pink')
plt.title('comparison of male and female')
plt.xlabel('Gender')
plt.ylabel('population')
plt.show()
```

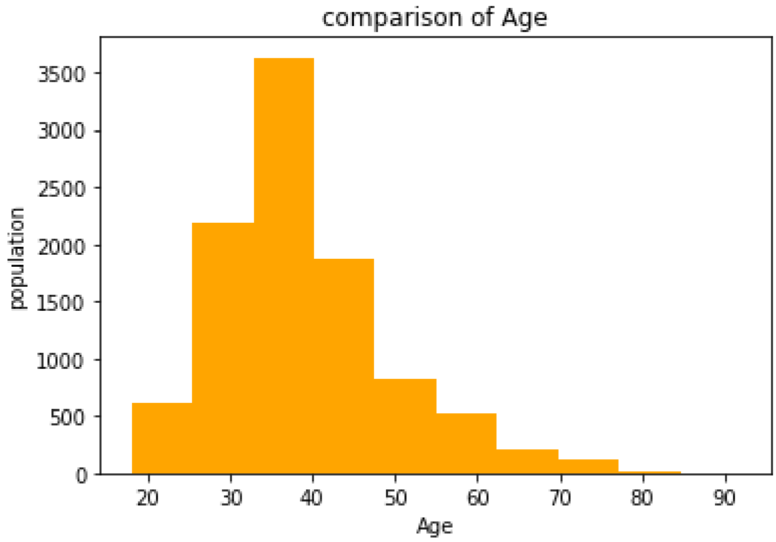



```
data['Age'].value_counts()
```

```
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
```

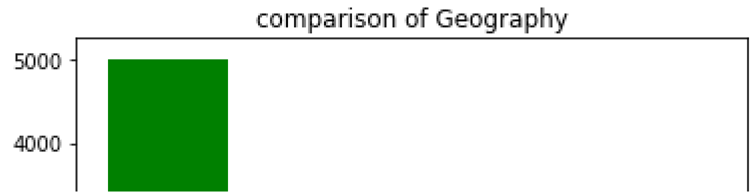
```
# comparison of age in the dataset

plt.hist(x = data.Age, bins = 10, color = 'orange')
plt.title('comparison of Age')
plt.xlabel('Age')
plt.ylabel('population')
plt.show()
```




```
data['Geography'].value_counts()
```

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

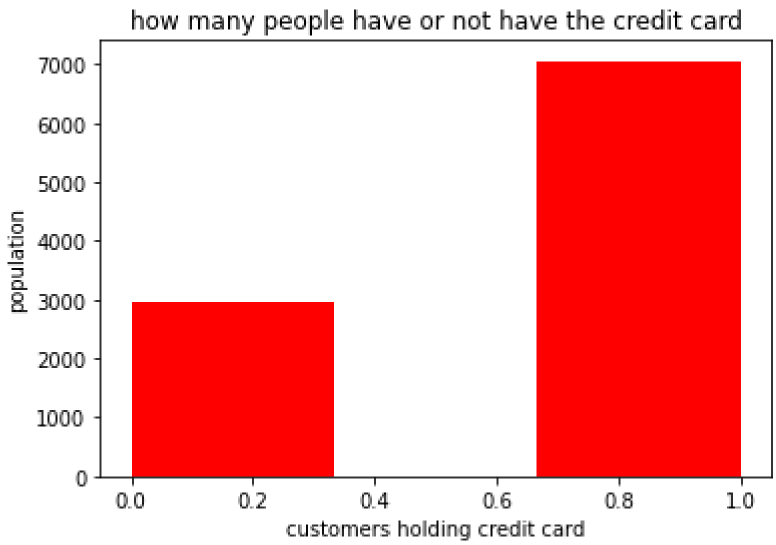
```
# comparison of geography

plt.hist(x = data.Geography, bins = 5, color = 'green')
plt.title('comparison of Geography')
plt.xlabel('Geography')
plt.ylabel('population')
plt.show()
```




```
data['HasCrCard'].value_counts()
```

```
1    7055
0    2945
Name: HasCrCard, dtype: int64
```

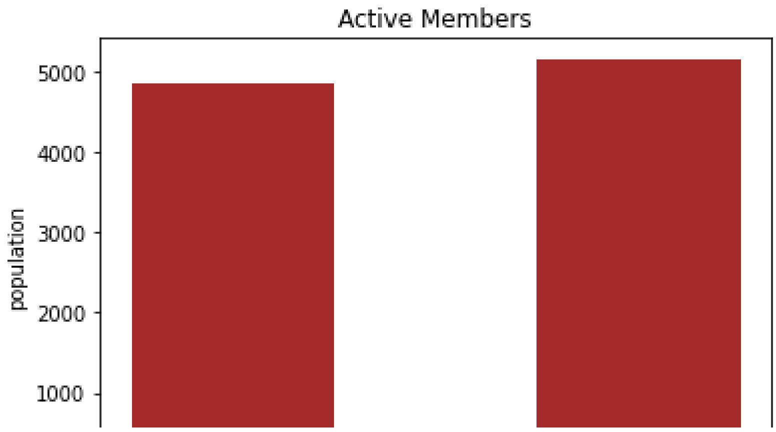
```
# comparision of how many customers hold the credit card

plt.hist(x = data.HasCrCard, bins = 3, color = 'red')
plt.title('how many people have or not have the credit card')
plt.xlabel('customers holding credit card')
plt.ylabel('population')
plt.show()
```




```
data['IsActiveMember'].value_counts()
```

```
1    5151
0    4849
Name: IsActiveMember, dtype: int64
```

```
# How many active member does the bank have ?

plt.hist(x = data.IsActiveMember, bins = 3, color = 'brown')
plt.title('Active Members')
plt.xlabel('Customers')
plt.ylabel('population')
plt.show()
```

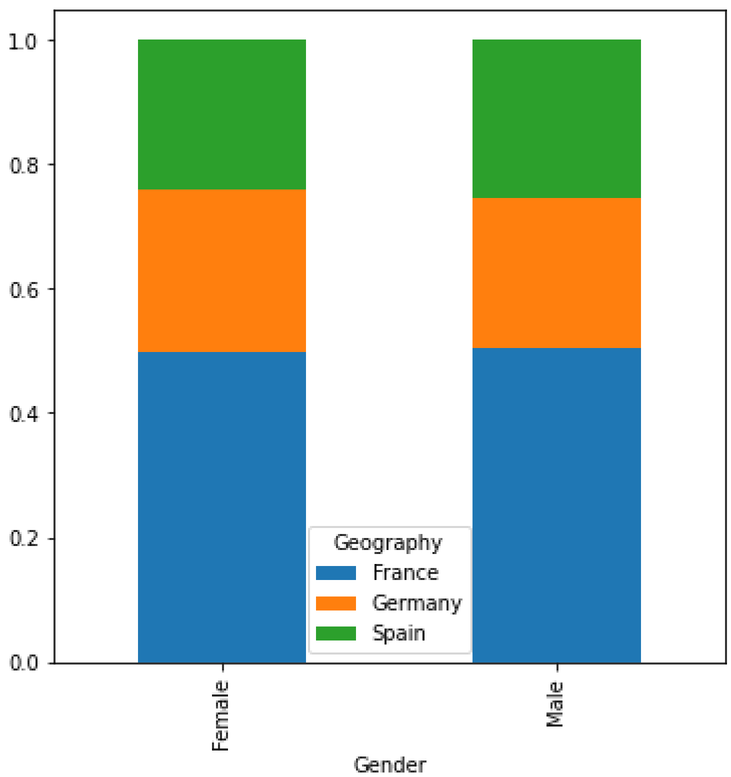


2. Bi - Variate Analysis

Customers

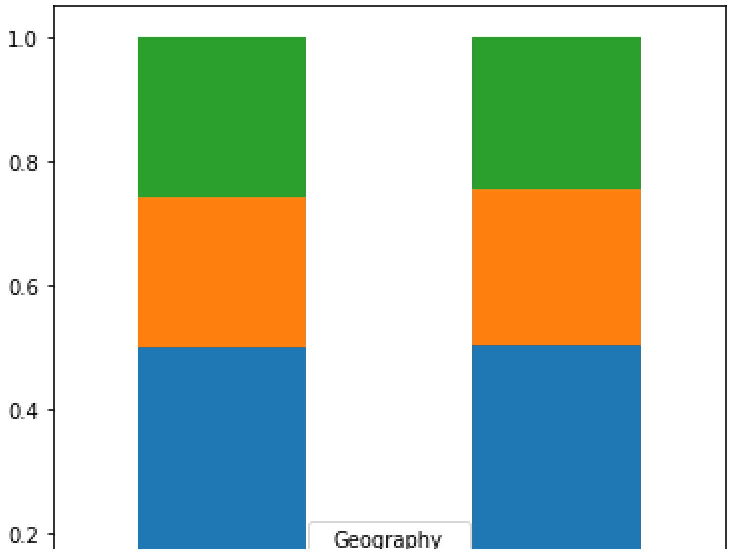
```
# comparison between Geography and Gender

Gender = pd.crosstab(data['Gender'],data['Geography'])
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(6,
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54c8e932d0>
```



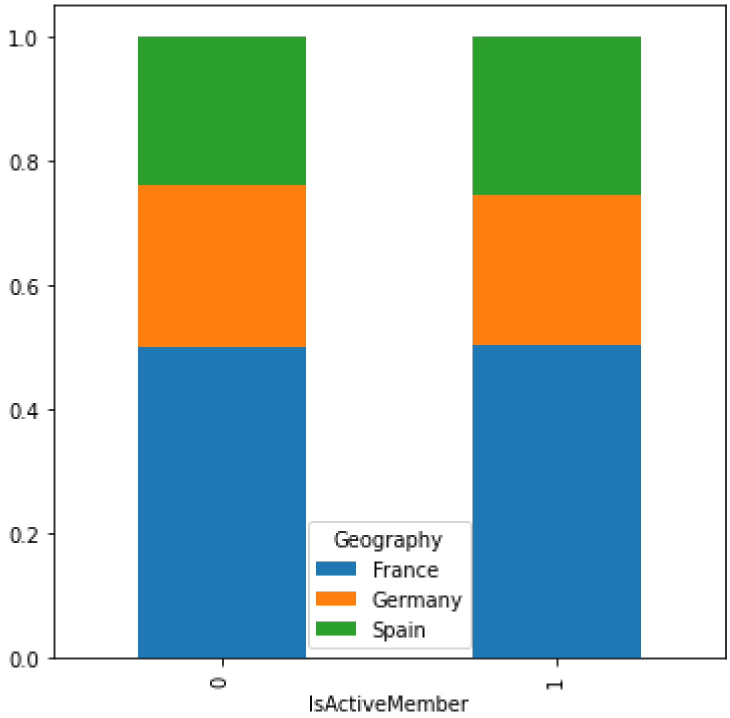

```
# comparison between geography and card holders

HasCrCard = pd.crosstab(data['HasCrCard'], data['Geography'])
HasCrCard.div(HasCrCard.sum(1).astype(float), axis = 0).plot(kind = 'bar',
                                            stacked = True,figsize = (6, 6))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f54c8e339d0>



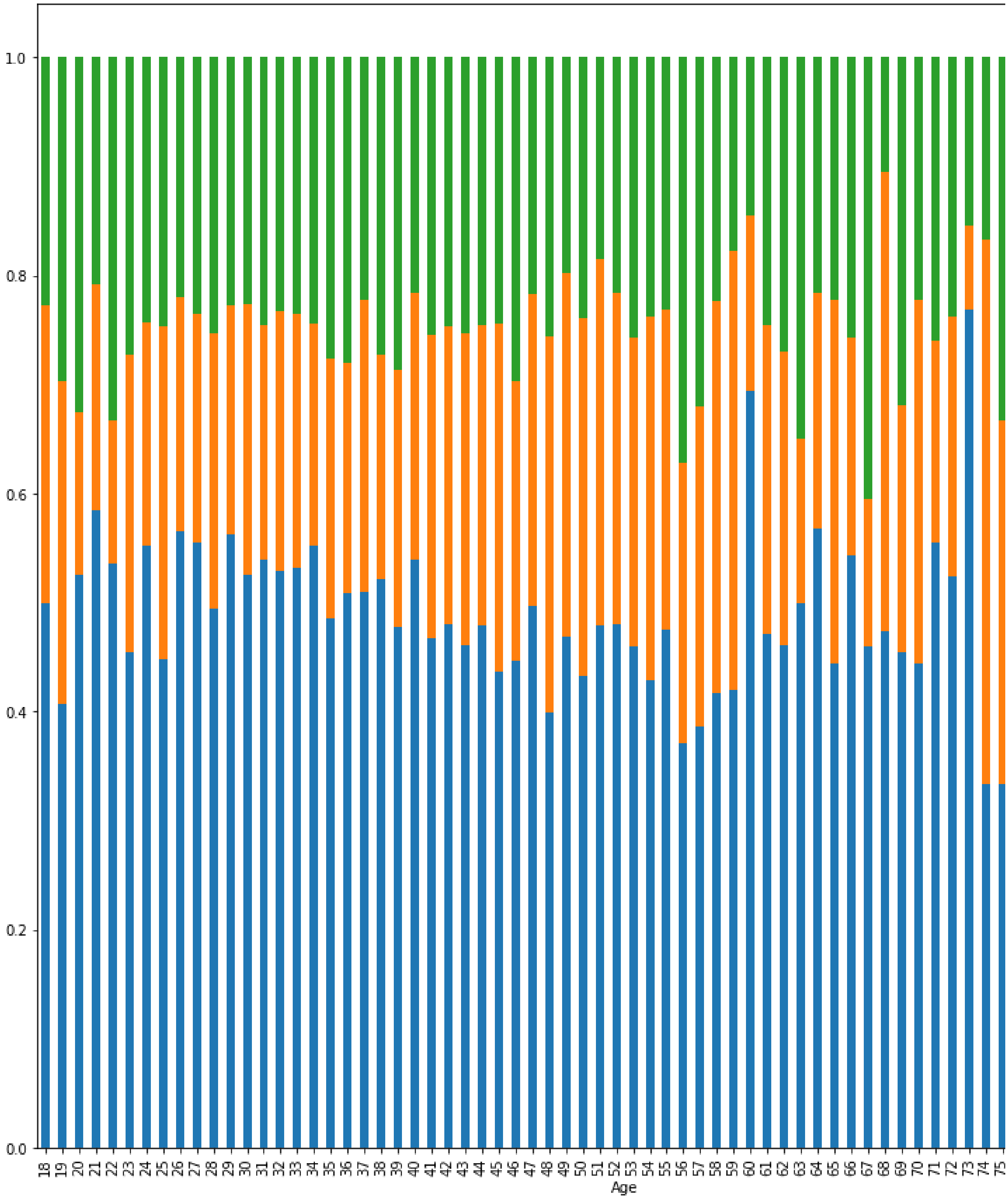

```
# comparison of active member in differnt geographies

IsActiveMember = pd.crosstab(data['IsActiveMember'], data['Geography'])
IsActiveMember.div(IsActiveMember.sum(1).astype(float), axis = 0).plot(kind = 'bar',
                                                   stacked = True, figsize= (6, 6))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f54c8d39fd0>




```
# comparing ages in different geographies

Age = pd.crosstab(data['Age'], data['Geography'])
Age.div(Age.sum(1).astype(float), axis = 0).plot(kind = 'bar',
                                               stacked = True, figsize = (15,15))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54c8ccc3d0>
```

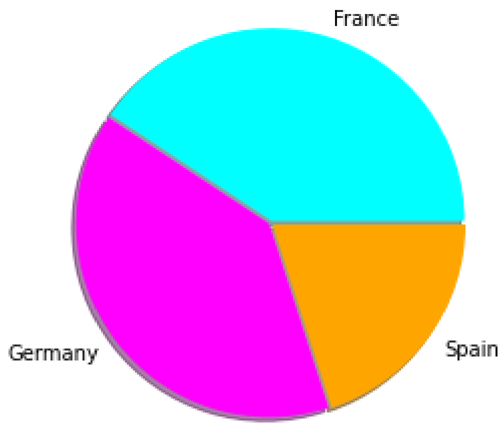



```
# calculating total balance in france, germany and spain

total_france = data.Balance[data.Geography == 'France'].sum()
total_germany = data.Balance[data.Geography == 'Germany'].sum()
total_spain = data.Balance[data.Geography == 'Spain'].sum()

print("Total Balance in France :",total_france)
print("Total Balance in Germany :",total_germany)
print("Total Balance in Spain :",total_spain)
```

```
Total Balance in France : 311332479.49
Total Balance in Germany : 300402861.38
```

```
Total Balance in Spain : 153123552.01
```

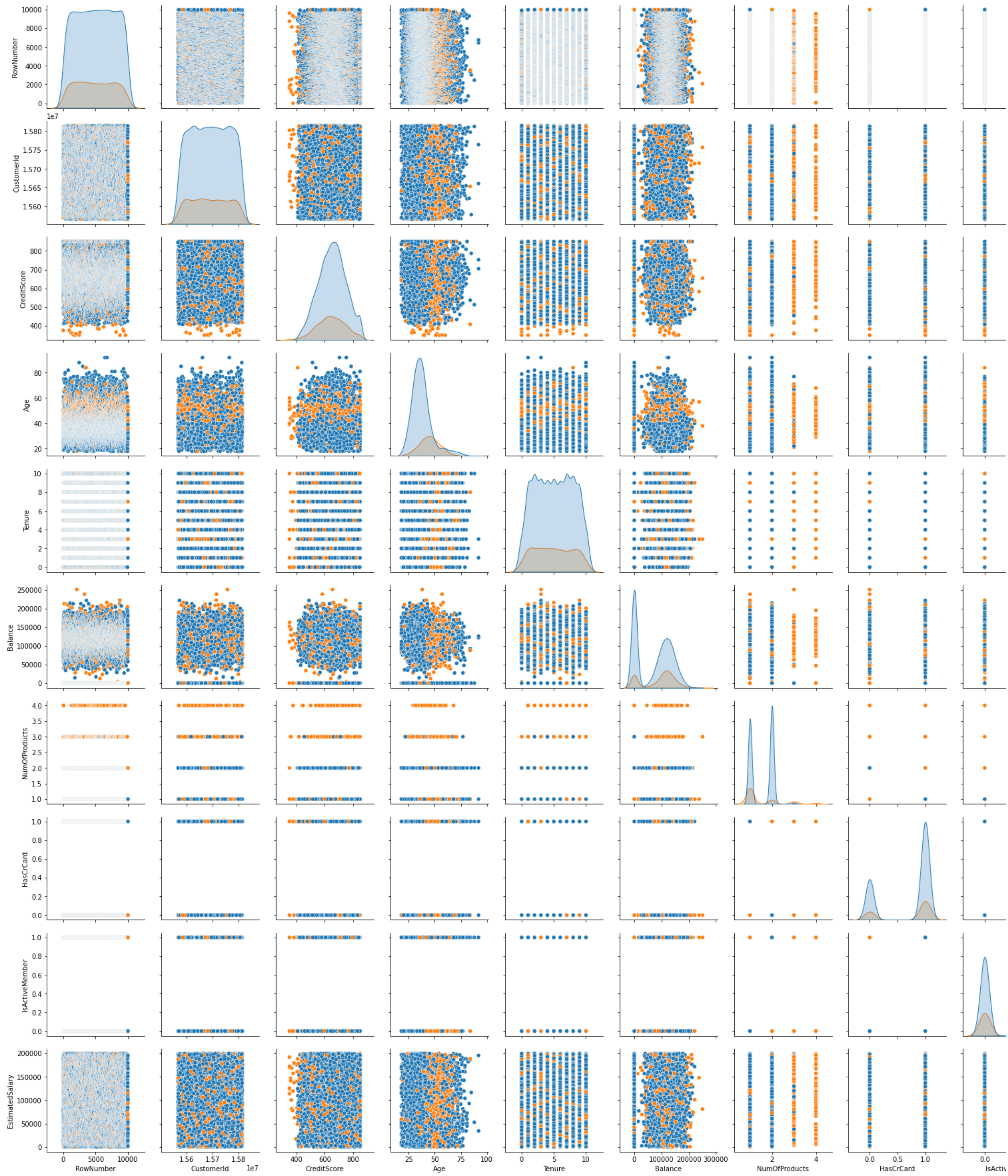
```
# plotting a pie chart

labels = 'France', 'Germany', 'Spain'
colors = ['cyan', 'magenta', 'orange']
sizes =  [311, 300, 153]
explode = [ 0.01, 0.01, 0.01]

plt.pie(sizes, colors = colors, labels = labels, explode = explode, shadow = True)

plt.axis('equal')
plt.show()
```



3. Multi - Variate Analysis

```
sns.pairplot(data=data, hue='Exited')
```

<seaborn.axisgrid.PairGrid at 0x7f54c899bf10>



Descriptive statistics

```
#Statistical analysis
data.describe()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCr
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0
mean	650.561300	38.660800	5.012800	76485.889288	1.527200	0.7
std	96.558702	9.746704	2.892174	62397.405202	0.570081	0.4
min	383.000000	18.000000	0.000000	0.000000	1.000000	0.0
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.0
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.0
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.0
max	850.000000	62.000000	10.000000	250898.090000	3.500000	1.0

Handle the Missing values

```
#Missing Values
data.isnull().sum()
```

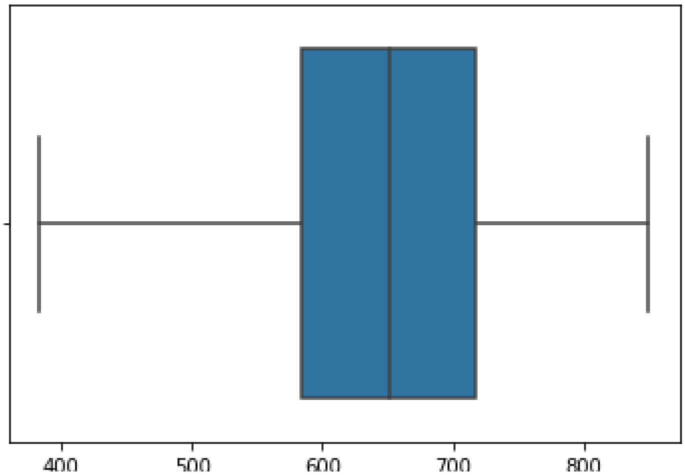
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

No missing values are found.

Find the outliers and replace the outliers

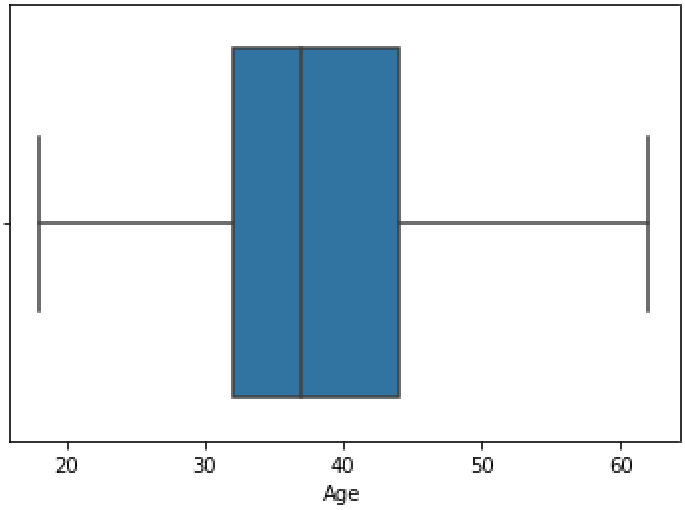
```
sns.boxplot(data = data, x = 'CreditScore')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54c17ad910>
```



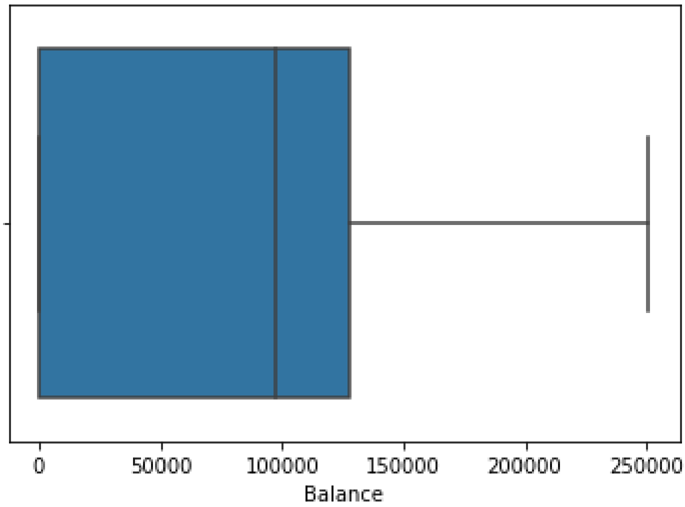

```
sns.boxplot(data = data, x = 'Age')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54c1824bd0>
```



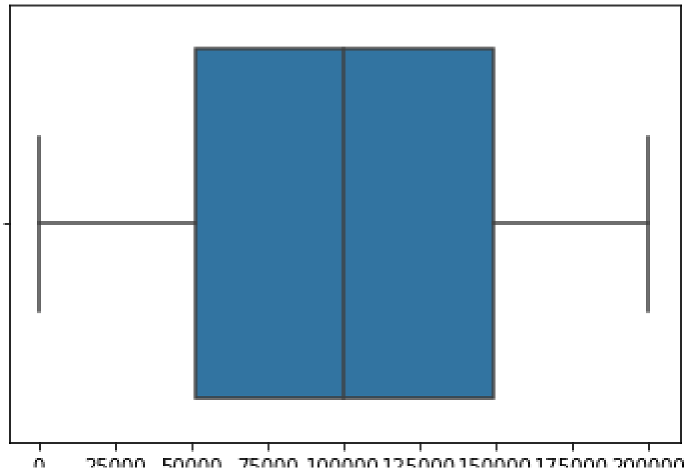

```
sns.boxplot(data = data, x = 'Balance')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54c17af490>
```




```
sns.boxplot(data = data, x = 'EstimatedSalary')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54c1794450>
```




```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i] <lower, lower, data[i])
```

```
data.describe()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCr
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0
mean	650.561300	38.660800	5.012800	76485.889288	1.527200	0.7
std	96.558702	9.746704	2.892174	62397.405202	0.570081	0.4
min	383.000000	18.000000	0.000000	0.000000	1.000000	0.0
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.0
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.0
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.0
max	850.000000	62.000000	10.000000	250898.090000	3.500000	1.0

Preprocessing

```
# Removing the unnecassary features from the dataset

data = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis = 1)

print(data.columns)
```

```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited'],
      dtype='object')
```

```
data.shape
```

```
(10000, 11)
```

Split the data into dependent and independent variables

```
# splitting the dataset into x(independent variables) and y(dependent variables)

x = data.iloc[:,0:10]
y = data.iloc[:,10]

print(x.shape)
print(y.shape)

print(x.columns)
```

```
(10000, 10)
(10000,)
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary'],
      dtype='object')
```

Check for Categorical columns and perform encoding

```
# Encoding Categorical variables into numerical variables
# One Hot Encoding

x = pd.get_dummies(x)

x.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Es
0	619.0	42.0	2.0	0.00	1.0	1.0	1.0	
1	608.0	41.0	1.0	83807.86	1.0	0.0	1.0	
2	502.0	42.0	8.0	159660.80	3.0	1.0	0.0	
3	699.0	39.0	1.0	0.00	2.0	0.0	0.0	
4	850.0	43.0	2.0	125510.82	1.0	1.0	1.0	

Split the data into training and testing

```
# splitting the data into training and testing set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state =

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(7500, 13)
(7500,)
(2500, 13)
(2500,)
```

Scale the independent variables

```
# Feature Scaling
# Only on Independent Variable to convert them into values ranging from -1 to +1

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

x_train = pd.DataFrame(x_train)
x_train.head()
```

	0	1	2	3	4	5	6	7
0	-0.736828	0.042283	0.008860	0.673160	2.583231	-1.553624	-1.034460	-1.640810
1	1.025257	-0.674496	0.008860	-1.207724	0.822578	0.643657	-1.034460	-0.079272
2	0.808861	-0.469702	1.393293	-0.356937	0.822578	0.643657	0.966688	-0.996840
3	0.396677	-0.060114	0.008860	-0.009356	-0.938076	0.643657	0.966688	-1.591746
4	-0.468908	1.373444	0.701077	-1.207724	0.822578	0.643657	0.966688	1.283302