IBM ASSIGNMENT-2

STEP-1: Downloading the dataset

Dataset has been downloaded successfully.

STEP-2: Loading the dataset To

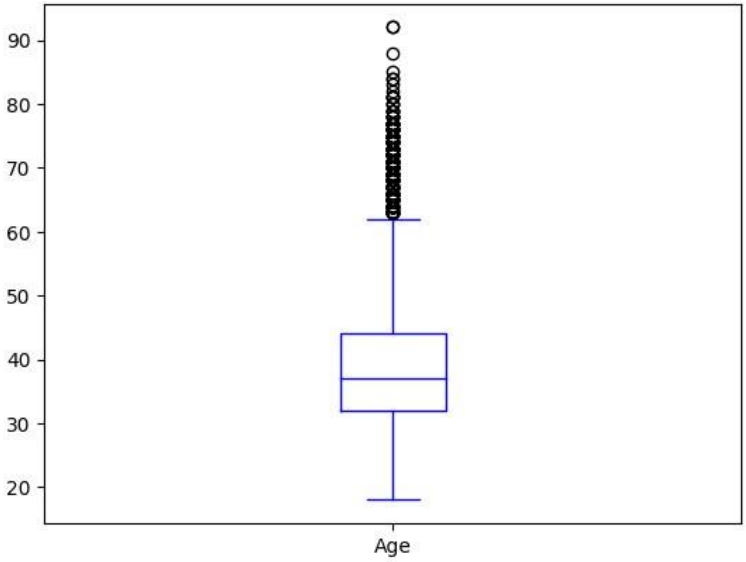
load the dataset

import pandas as pd import numpy as np df = pd.read_csv ('Churn_Modelling.csv')

STEP-3: Performing visualizations

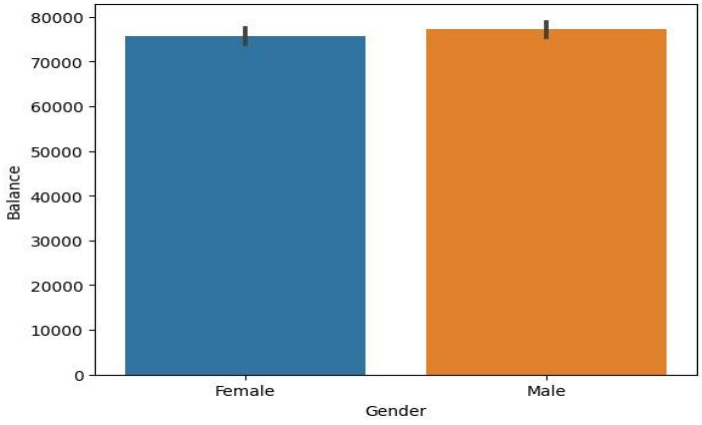
- **UNIVARIATE**

df.boxplot(column=['Age'], grid=False, color='blue')



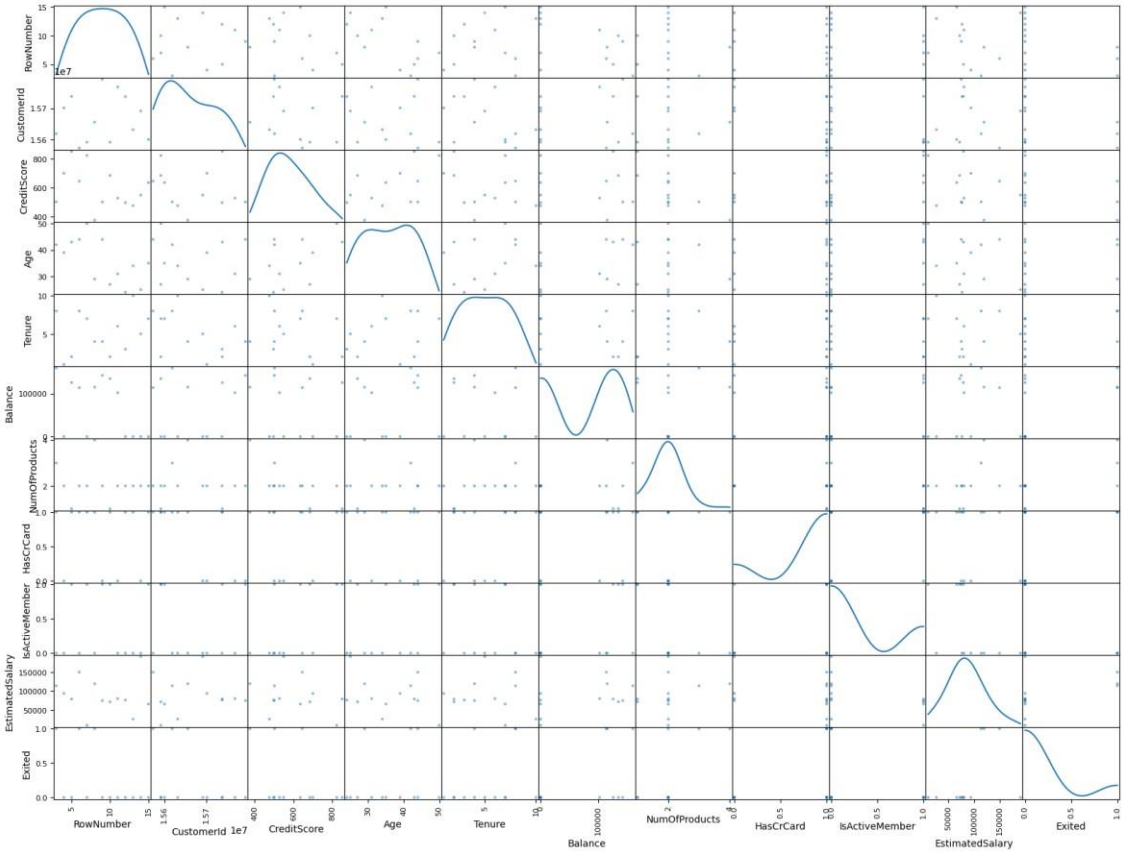
- **BIVARIATE**

sns.barplot(x='Gender', y='Balance', data=df)



- **MULTIVARIATE**

pd.plotting.scatter_matrix(df.loc[2:14], diagonal="kde",figsize=(20,15)) plt.show()



STEP-4: Performing descriptive statistics on a dataset ***df.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
```

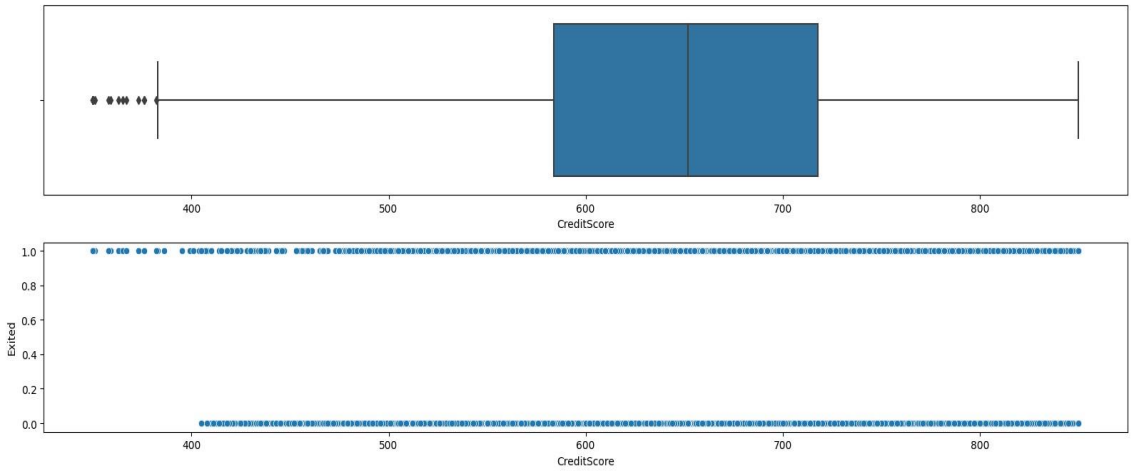
STEP-5: Checking for missing values ***df.isnull().sum()***

```
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
```

STEP-6: Finding and replacing outliers CHECKING FOR OUTLIERS

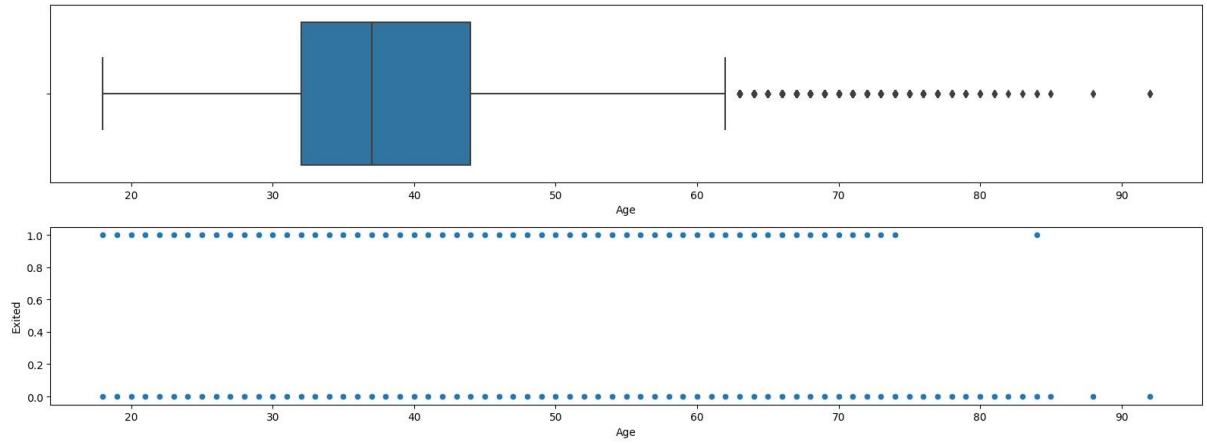
def box_scatter(data, x, y):
fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
sns.boxplot(data=data, x=x, ax=ax1)
sns.scatterplot(data=data, x=x,y=y,ax=ax2)
box_scatter(df,'CreditScore','Exited'); plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] <400])}")

```
# of Bivariate Outliers: 19
```

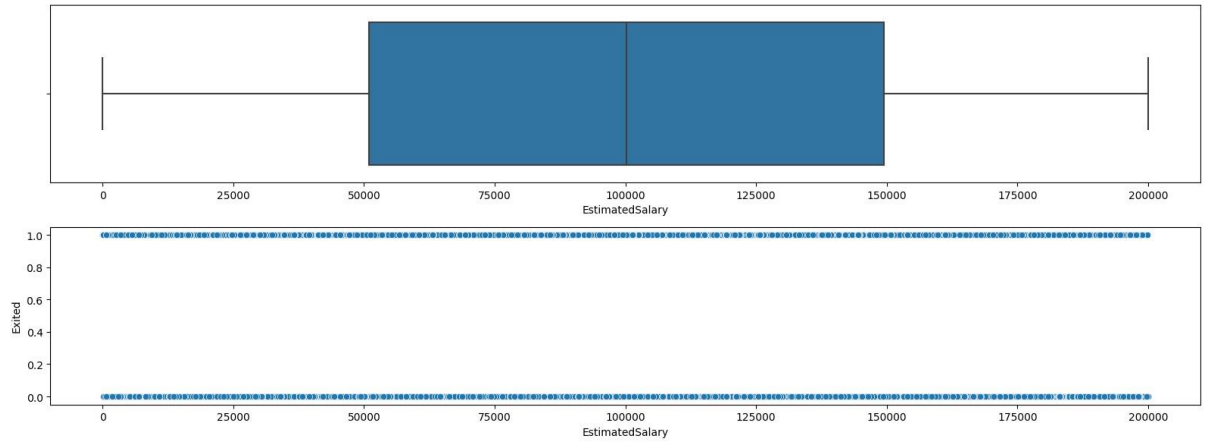


box_scatter(df,'Age','Exited'); plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] >87])}")

```
# of Bivariate Outliers: 3
```



box_scatter(df,'EstimatedSalary','Exited'); plt.tight_layout()



REMOVING THE OUTLIERS

for i in df:

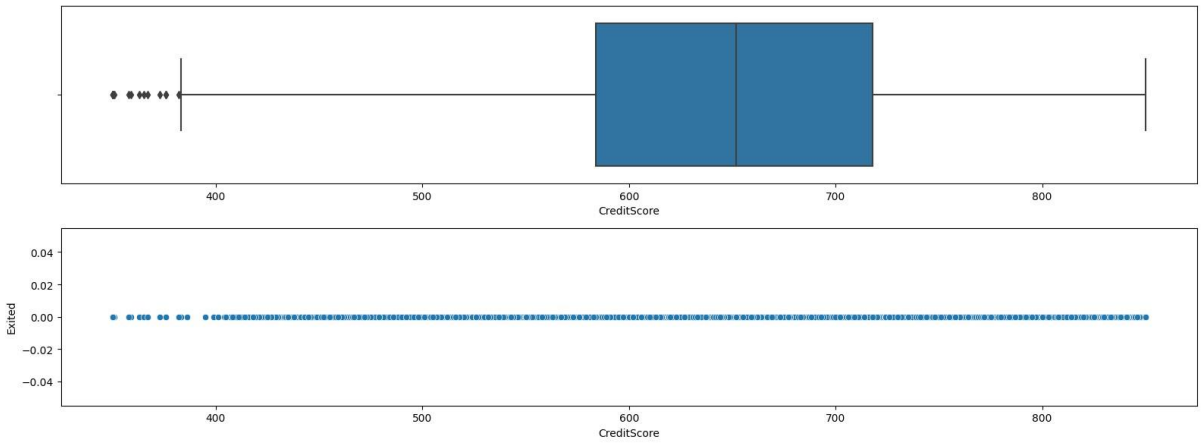
if df[i].dtype=='int64'or df[i].dtypes=='float64':

q1=df[i].quantile(0.25)
q3=df[i].quantile(0.75)
iqr=q3-q1 upper=q3+1.5*iqr
lower=q1-1.5*iqr
df[i]=np.where(df[i] >upper, upper, df[i]) df[i]=np.where(df[i] <lower, lower, df[i])

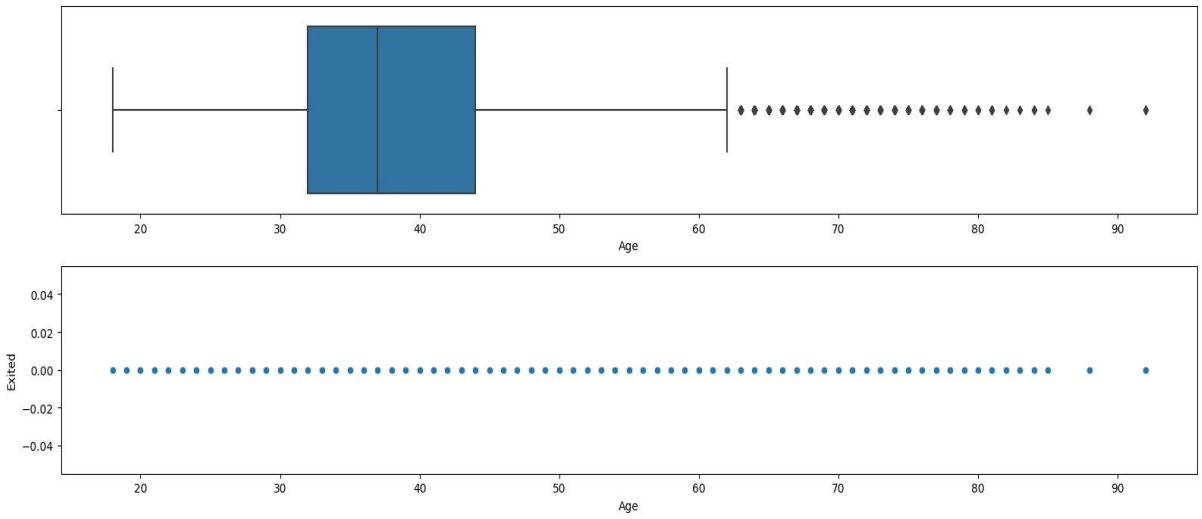
AFTER REMOVING OUTLIERS

box_scatter(df,'CreditScore','Exited'); plt.tight_layout()

print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] <400])}")



box_scatter(df,'Age','Exited'); plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] >87])}")



STEP-7: Check for categorical columns and perform encoding

from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder() for i in df: if
df[i].dtype=='object'or df[i].dtype=='category':
df[i]=encoder.fit_transform(df[i])

STEP-8: Split the data into dependent and independent variables

x=df.iloc[:,:-1] x.head()

Row Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	
0	1	15634602	1115	619	0	0	42	2	0.00	1	1	1	101348.88
1	2	15647311	1177	608	2	0	41	1	83807.86	1	0	1	112542.58
2	3	15619304	2040	502	0	0	42	8	159660.80	3	1	0	113931.57
3	4	15701354	289	699	0	0	39	1	0.00	2	0	0	93826.63
4	5	15737888	1822	850	2	0	43	2	125510.82	1	1	1	79084.10

y=df.iloc[:,-1] y.head()

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: Exited, dtype: float64
```

STEP-9: Split the independent variables ***from sklearn.preprocessing import StandardScaler scaler=StandardScaler() x=scaler.fit_transform(x)***

x

```
array([[-1.73187761,  -0.78321342,  -0.46418322,  ...,   0.64609167,
0.97024255,  0.02188649],
       [-1.7315312 ,  -0.60653412,  -0.3909112 ,  ...,  -1.54776799,
0.97024255,  0.21653375],
       [-1.73118479, -0.99588476,  0.62898807, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 1.73118479, -1.47928179,  0.07353887, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 1.7315312 , -0.11935577,  0.98943914, ...,  0.64609167,       -
1.03067011, -0.12523071],
       [ 1.73187761, -0.87055909,  1.4692527 , ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

STEP-10: Split the data into training and testing

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33)