Import Library Files

```
from google.colab import drive

drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.compose import ColumnTransformer


from sklearn.preprocessing import OneHotEncoder

from sklearn.preprocessing import StandardScaler
```

2. Load the dataset

```
data = pd.read_csv('/content/drive/MyDrive/Churn_Modelling.csv')
```

3.1 UNIVARIATE ANALYSIS

```
l=['CreditScore','Age', 'Tenure','Balance','NumOfProducts','EstimatedSalary']

for i in l:

  sns.displot(data=data[i],kde=True)
```

```
l=['CreditScore','Age', 'Tenure','Balance','NumOfProducts','EstimatedSalary']

fig, (ax1, ax2, ax3, ax4, ax5, ax6) = plt.subplots(nrows=6, ncols=1, figsize=(10,20))

data.boxplot(column=[l[0]],grid='False',color='blue',ax=ax1)

data.boxplot(column=[l[1]],grid='False',color='blue',ax = ax2)
```

```
data.boxplot(column=[l[2]],grid='False',color='blue',ax = ax3)
data.boxplot(column=[l[3]],grid='False',color='blue',ax = ax4)
data.boxplot(column=[l[4]],grid='False',color='blue',ax = ax5)
data.boxplot(column=[l[5]],grid='False',color='blue',ax = ax6)
plt.tight_layout()

import warnings
warnings.filterwarnings("ignore")
fig, (ax1, ax2, ax3) = plt.subplots(nrows=3, ncols=1, figsize=(16,16))
sns.countplot(data.HasCrCard,ax=ax1)
sns.countplot(data.IsActiveMember,ax=ax2)
sns.countplot(data.Exited,ax=ax3)
plt.tight_layout()
```

3.2 BI - VARIATE ANALYSIS

```
for i in range(len(l)-1):
    for j in range(i+1,len(l)):
        sns.relplot(x = l[i],y = l[j],data = data)
```

3.3 MULTI - VARIATE ANALYSIS

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

```
<seaborn.axisgrid.FacetGrid at 0x7fdd89a27210>
```

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

```
<seaborn.axisgrid.PairGrid at 0x7fdd87c0c290>
```

4. Perform descriptive statistics on the dataset

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	2	15647311	645	608	2	0	41	1	83807.86	1	0	1	112542.58	0
5	6	15574012	302	645	2	1	44	8	113755.78	2	1	0	149756.71	1
10	11	15767821	109	528	0	1	31	6	102016.72	2	0	0	80181.12	0
15	16	15643966	561	616	1	1	45	3	143129.41	2	0	1	64327.26	0
26	27	15736816	1605	756	1	1	36	2	136815.64	1	1	1	170041.95	0

```
data.describe()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	3354.000000	3.354000e+03	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000	3354.000000

mean	4993.122242	1.568997e+07	834.928145	651.885808	0.798151	0.551580	38.594812	4.960644	111127.251270	1.386106	0.705426	0.496720	101173.425200	0.237627
std	2889.712337	7.220517e+04	467.984617	66.341508	0.744870	0.497407	6.171482	2.910065	23930.791436	0.579239	0.455919	0.500064	57475.269109	0.425693
min	2.000000	1.556571e+07	0.000000	522.000000	0.000000	0.000000	29.000000	0.000000	3768.690000	1.000000	0.000000	0.000000	11.580000	0.000000
25%	2469.250000	1.562732e+07	443.250000	601.000000	0.000000	0.000000	34.000000	2.000000	96579.825000	1.000000	0.000000	0.000000	53183.340000	0.000000
50%	4986.500000	1.568912e+07	847.500000	652.000000	1.000000	1.000000	38.000000	5.000000	113904.805000	1.000000	1.000000	0.000000	101348.755000	0.000000
75%	7483.750000	1.575380e+07	1250.000000	705.000000	1.000000	1.000000	43.000000	7.000000	129621.140000	2.000000	1.000000	1.000000	150202.787500	0.000000
max	9999.000000	1.581569e+07	1628.000000	777.000000	2.000000	1.000000	53.000000	10.000000	149238.970000	4.000000	1.000000	1.000000	199970.740000	1.000000

data.dtypes

RowNumber int64

CustomerId int64

Surname int64

CreditScore int64

Geography int64

Gender int64

Age int64

Tenure int64

Balance float64

NumOfProducts int64

HasCrCard int64

IsActiveMember int64

EstimatedSalary float64

Exited int64

dtype: object

data.skew()

RowNumber 0.011381

CustomerId 0.010861

Surname -0.033943

CreditScore -0.024726

Geography 0.344510

Gender -0.207520

Age 0.415624

Tenure 0.034787

Balance -0.691529

NumOfProducts 1.450814

HasCrCard -0.901691

IsActiveMember 0.013125

EstimatedSalary 0.002697

Exited 1.233424

dtype: float64

5. Handle the Missing values.

data.isnull().any()

RowNumber False

CustomerId False

Surname False

CreditScore False

Geography False

Gender False

Age False

Tenure False

Balance False

NumOfProducts False

HasCrCard False

IsActiveMember False

EstimatedSalary False

Exited False

dtype: bool

No missing values

6. Find the outliers and replace the outliers

data['CreditScore'].describe()

count 10000.000000

mean 650.528800

std 96.653299

min 350.000000

25% 584.000000

50% 652.000000

75% 718.000000

max 850.000000

Name: CreditScore, dtype: float64

data['Age'].describe()

count 10000.000000

mean 38.921800

std 10.487806

min 18.000000

25% 32.000000

50% 37.000000

75% 44.000000

max 92.000000

```
Name: Age, dtype: float64
data['Balance'].describe()
count     10000.000000
mean      76485.889288
std       62397.405202
min           0.000000
25%           0.000000
50%       97198.540000
75%      127644.240000
max      250898.090000
Name: Balance, dtype: float64
l=['Balance','Age','CreditScore']
for i in l:
    percentile_least = data[i].quantile(0.1)
    percentile90 = data[i].quantile(0.9)
    data = data[(data[i]<percentile90)& (data[i]>percentile_least)]
data['CreditScore'].describe()
count    3354.000000
mean      651.885808
std        66.341508
min       522.000000
25%       601.000000
50%       652.000000
75%       705.000000
max       777.000000
Name: CreditScore, dtype: float64
data['Age'].describe()
count    3354.000000
mean       38.594812
```

std   6.171482

min   29.000000

25%   34.000000

50%   38.000000

75%   43.000000

max   53.000000

Name: Age, dtype: float64

data['Balance'].describe()

count   3354.000000

mean   111127.251270

std   23930.791436

min   3768.690000

25%   96579.825000

50%   113904.805000

75%   129621.140000

max   149238.970000

Name: Balance, dtype: float64

The STD have reduced drastically indicating removal of outliers.

7. Check for Categorical columns and perform encoding.

```
from sklearn.preprocessing import LabelEncoder

encoder=LabelEncoder()

for i in data:

   if data[i].dtype=='object':

      data[i]=encoder.fit_transform(data[i])

data.head()
```

RowNumber CustomerId Surname CreditScore Geography Gender Age Tenure Balance NumOfProductsHasCrCard IsActiveMember EstimatedSalaryExited

1	2	15647311	645	608	2	0	41	1	83807.86	1		
	0	1	112542.58	0								
5	6	15574012	302	645	2	1	44	8	113755.78	2		
	1	0	149756.71	1								
10	11	15767821	109	528	0	1	31	6	102016.72	2		
	0	0	80181.12	0								
15	16	15643966	561	616	1	1	45	3	143129.41	2		
	0	1	64327.26	0								
26	27	15736816	1605	756	1	1	36	2	136815.64	1		
	1	1	170041.95	0								

8. Split the data into dependent and independent variables.

data.shape

(3354, 14)

x = data.iloc[:,:13]

y = data.iloc[:,13]

y.head()

1 0

5 1

10 0

15 0

26 0

Name: Exited, dtype: int64

x.head()

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender Age	Tenure
Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary		

1	2	15647311	645	608	2	0	41	1	83807.86	1
	0	1	112542.58							
5	6	15574012	302	645	2	1	44	8	113755.78	2
	1	0	149756.71							
10	11	15767821	109	528	0	1	31	6	102016.72	2
	0	0	80181.12							

15	16	15643966	561	616	1	1	45	3	143129.41	2	0	1	64327.26
26	27	15736816	1605	756	1	1	36	2	136815.64	1	1	1	170041.95

9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x = sc.fit_transform(x)
```

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=0.2, random_state=0)

x_train.shape

(2683, 13)

y_train.shape

(2683,)

x_test.shape

(671, 13)

y_test.shape

(671,)
```