1.Download Dataset

2.Load Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
df= pd.read_csv("Churn_Modelling.csv")
df
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender
Age  \
0             1    15634602   Hargrave          619    France  Female
42
1             2    15647311       Hill          608     Spain  Female
41
2             3    15619304       Onio          502    France  Female
42
3             4    15701354       Boni          699    France  Female
39
4             5    15737888   Mitchell          850     Spain  Female
43
...         ...         ...        ...          ...       ...     ...
...
9995       9996    15606229   Obijiaku          771    France    Male
39
9996       9997    15569892  Johnstone          516    France    Male
35
9997       9998    15584532        Liu          709    France  Female
36
9998       9999    15682355  Sabbatini          772   Germany    Male
42
9999      10000    15628319     Walker          792    France  Female
28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              1          1               1
1          1   83807.86              1          0               1
2          8  159660.80              3          1               0
3          1       0.00              2          0               0
4          2  125510.82              1          1               1
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
```

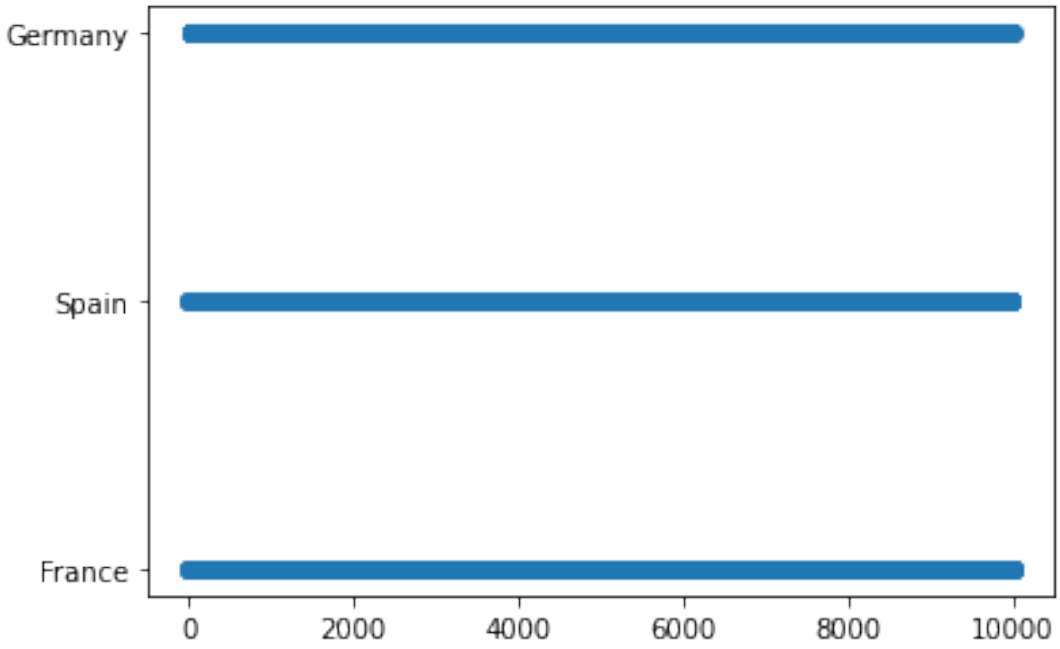
```
0            101348.88       1
1            112542.58       0
2            113931.57       1
3             93826.63       0
4             79084.10       0
...                ...     ...
9995          96270.64       0
9996         101699.77       0
9997          42085.58       1
9998          92888.52       1
9999          38190.78       0

[10000 rows x 14 columns]
```

3.1 Univariate Analysis

```
plt.scatter(df.index,df['Geography'])
```

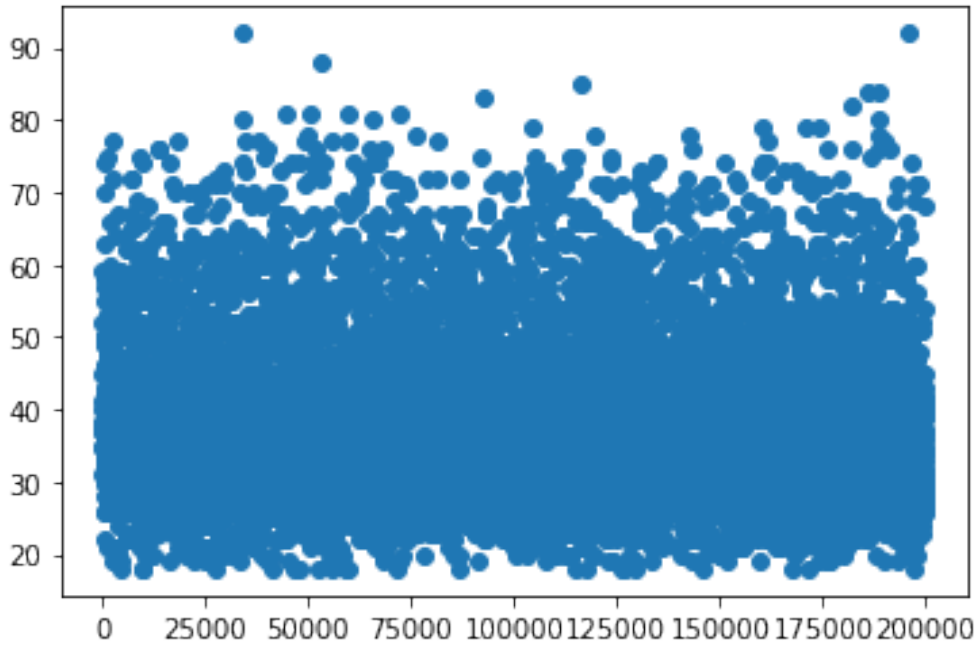
```
<matplotlib.collections.PathCollection at 0x7f3cfc73b290>
```



3.2 Bivariate Analysis

```
plt.scatter(df.EstimatedSalary,df.Age)
```

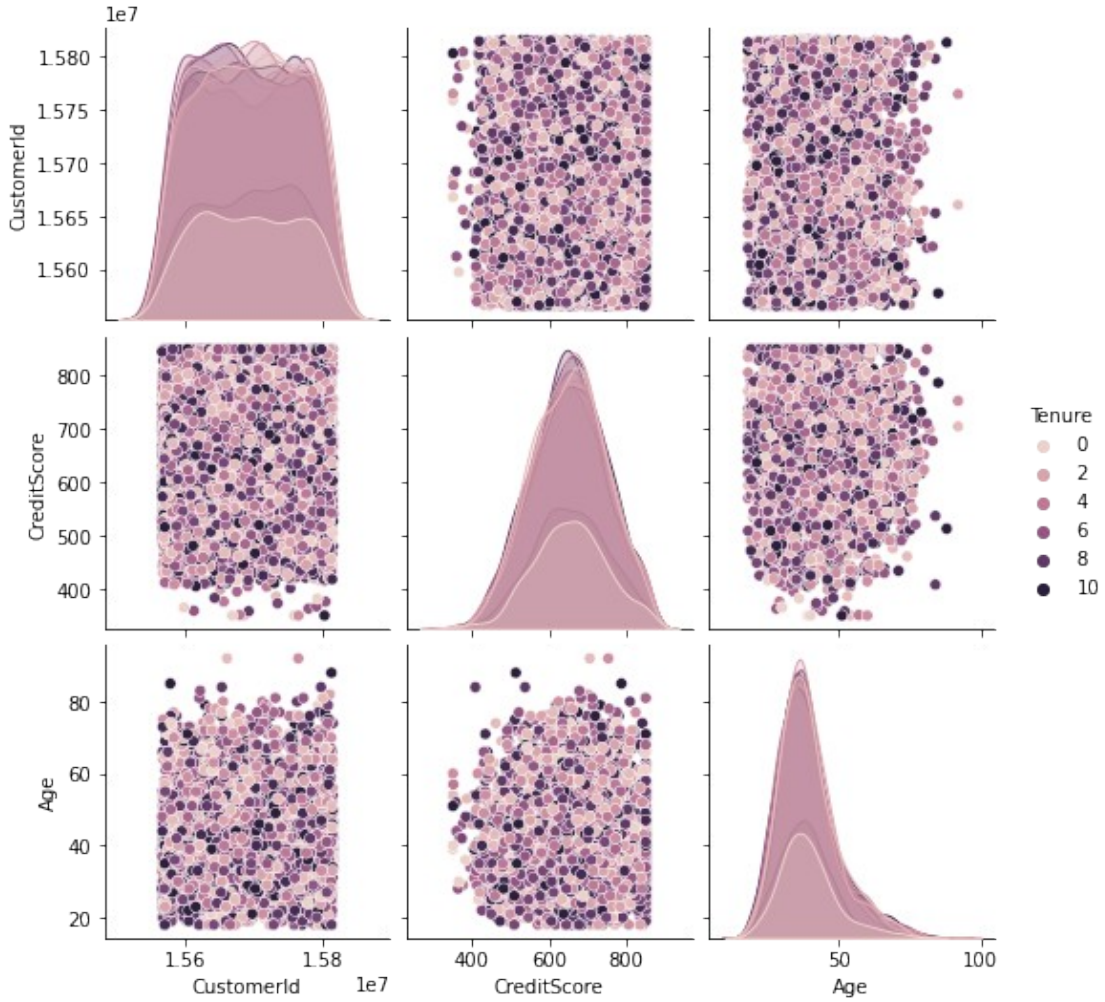
```
<matplotlib.collections.PathCollection at 0x7f3cfc22ff90>
```



3.3 Multivariate Analysis

```
sns.pairplot(data=df[['CustomerId', 'Surname', 'CreditScore',
'Geography','Gender', 'Age', 'Tenure']],hue='Tenure')
```

```
<seaborn.axisgrid.PairGrid at 0x7f3cfc1da9d0>
```



4.Descriptive Statistics

4.1 sum()

```
df.sum(numeric_only = True)
```

```
RowNumber          5.000500e+07
CustomerId         1.569094e+11
CreditScore        6.505288e+06
Age                3.892180e+05
Tenure             5.012800e+04
Balance            7.648589e+08
NumOfProducts      1.530200e+04
HasCrCard          7.055000e+03
IsActiveMember     5.151000e+03
EstimatedSalary    1.000902e+09
Exited             2.037000e+03
dtype: float64
```

4.2 mean()

```
df.mean(numeric_only = True)
```

```
RowNumber          5.000500e+03
CustomerId         1.569094e+07
CreditScore        6.505288e+02
Age                3.892180e+01
Tenure             5.012800e+00
Balance            7.648589e+04
NumOfProducts      1.530200e+00
HasCrCard          7.055000e-01
IsActiveMember     5.151000e-01
EstimatedSalary    1.000902e+05
Exited             2.037000e-01
dtype: float64
```

4.3 std()

```
df.std(numeric_only=True)
```

```
RowNumber           2886.895680
CustomerId         71936.186123
CreditScore           96.653299
Age                   10.487806
Tenure                 2.892174
Balance            62397.405202
NumOfProducts          0.581654
HasCrCard              0.455840
IsActiveMember         0.499797
EstimatedSalary    57510.492818
Exited                 0.402769
dtype: float64
```

4.4 describe()

```
df.describe(include='all')
```

```
          RowNumber    CustomerId Surname   CreditScore Geography
Gender  \
count   10000.00000  1.000000e+04   10000  10000.000000     10000
10000
unique          NaN           NaN    2932           NaN         3
2
top             NaN           NaN   Smith           NaN    France
Male
freq            NaN           NaN      32           NaN      5014
5457
mean     5000.50000  1.569094e+07     NaN    650.528800       NaN
NaN
std      2886.89568  7.193619e+04     NaN     96.653299       NaN
NaN
min         1.00000  1.556570e+07     NaN    350.000000       NaN
NaN
```

```
25%      2500.75000  1.562853e+07     NaN    584.000000       NaN
NaN
50%      5000.50000  1.569074e+07     NaN    652.000000       NaN
NaN
75%      7500.25000  1.575323e+07     NaN    718.000000       NaN
NaN
max     10000.00000  1.581569e+07     NaN    850.000000       NaN
NaN

                Age        Tenure        Balance  NumOfProducts
HasCrCard  \
count   10000.000000  10000.000000   10000.000000   10000.000000
10000.00000
unique           NaN           NaN            NaN            NaN
NaN
top              NaN           NaN            NaN            NaN
NaN
freq             NaN           NaN            NaN            NaN
NaN
mean       38.921800      5.012800   76485.889288       1.530200
0.70550
std        10.487806      2.892174   62397.405202       0.581654
0.45584
min        18.000000      0.000000       0.000000       1.000000
0.00000
25%        32.000000      3.000000       0.000000       1.000000
0.00000
50%        37.000000      5.000000   97198.540000       1.000000
1.00000
75%        44.000000      7.000000  127644.240000       2.000000
1.00000
max        92.000000     10.000000  250898.090000       4.000000
1.00000

        IsActiveMember  EstimatedSalary        Exited
count     10000.000000     10000.000000  10000.000000
unique             NaN              NaN           NaN
top                NaN              NaN           NaN
freq               NaN              NaN           NaN
mean          0.515100    100090.239881      0.203700
std           0.499797     57510.492818      0.402769
min           0.000000        11.580000      0.000000
25%           0.000000     51002.110000      0.000000
50%           1.000000    100193.915000      0.000000
75%           1.000000    149388.247500      0.000000
max           1.000000    199992.480000      1.000000
```

5.Handle Missing Values

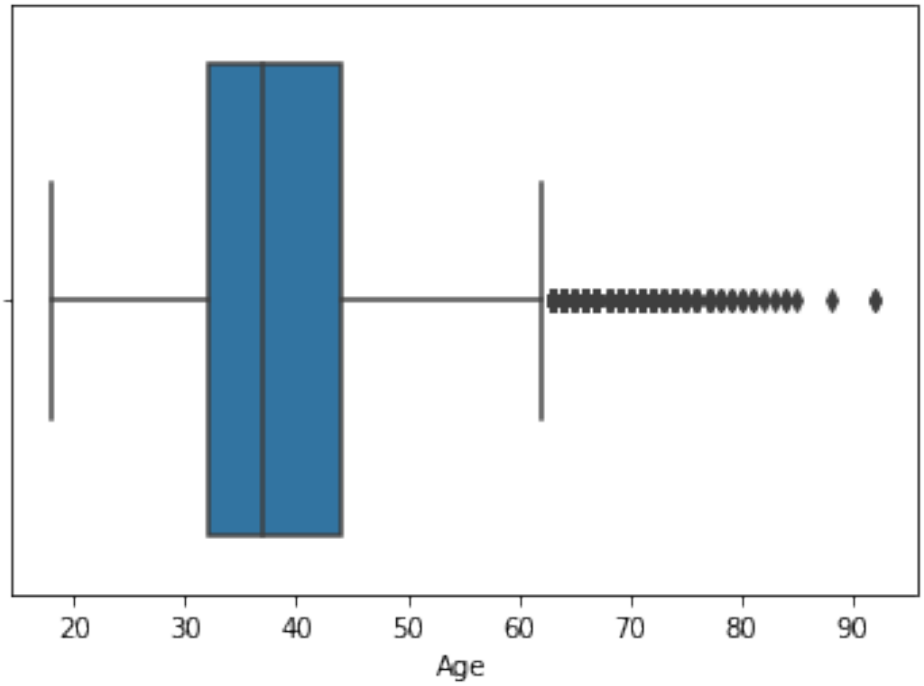
```
df.isnull().sum()
```

```
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
```

6.Find the outliers and replace the outliers

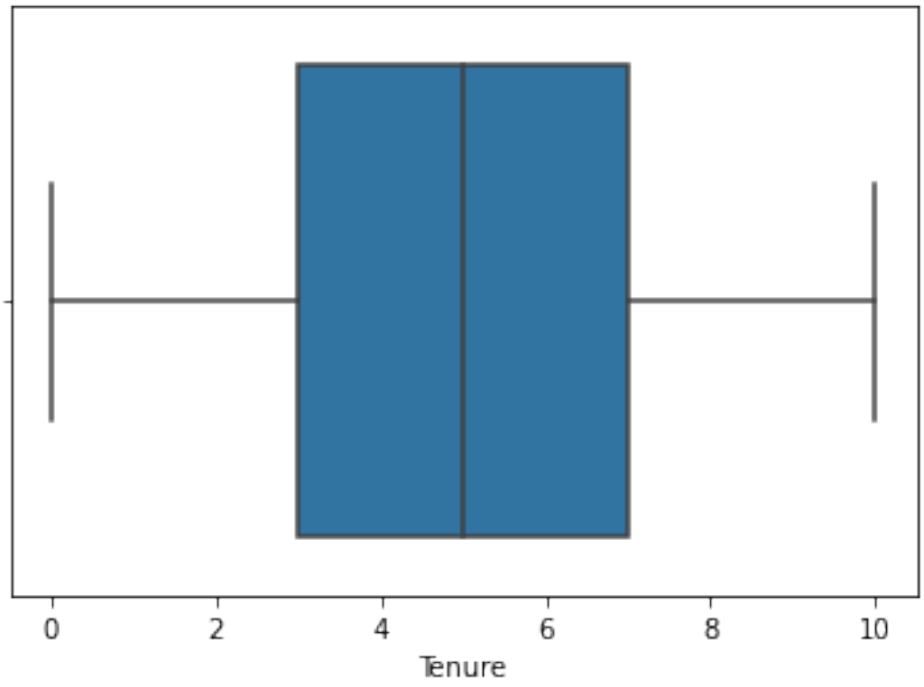
```
sns.boxplot(x=df['Age'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3cfc077dd0>
```




```
sns.boxplot(x=df['Tenure'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3cf9377d50>
```



7.Check for Categorical Columns and perform encoding

```
df.select_dtypes(include=['object']).columns.tolist()
```

```
['Surname', 'Geography', 'Gender']
```

8.Split the data into dependent and independent variables

```
x=df.iloc[:,0:999].values#independent
y=df.iloc[:,999:1000].values#dependent
```

9.Scale the independent variables

```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
x=df[['Age','Tenure']]
scaledx = scale.fit_transform(x)
print(scaledx)
```

```
[[ 0.29351742 -1.04175968]
 [ 0.19816383 -1.38753759]
 [ 0.29351742  1.03290776]
 ...
 [-0.27860412  0.68712986]
 [ 0.29351742 -0.69598177]
 [-1.04143285 -0.35020386]]
```

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)

print('X Train shape:{},Y.Train SHape:
{}'.format(x_train.shape,y_train.shape))
```

```
X Train shape:(8000, 2),Y.Train SHape:(8000, 0)
```

```
print('X Test Shape :{},Y Test SHape:
{}'.format(x_test.shape,y_test.shape))
```

```
X Test Shape :(2000, 2),Y Test SHape:(2000, 0)
```