1. Download the dataset: Dataset

2. Load the dataset.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
df=pd.read_csv(r'Churn_Modelling.csv')
df.head()
```

```
   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


   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

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

3. Perform the Below Visualizations.

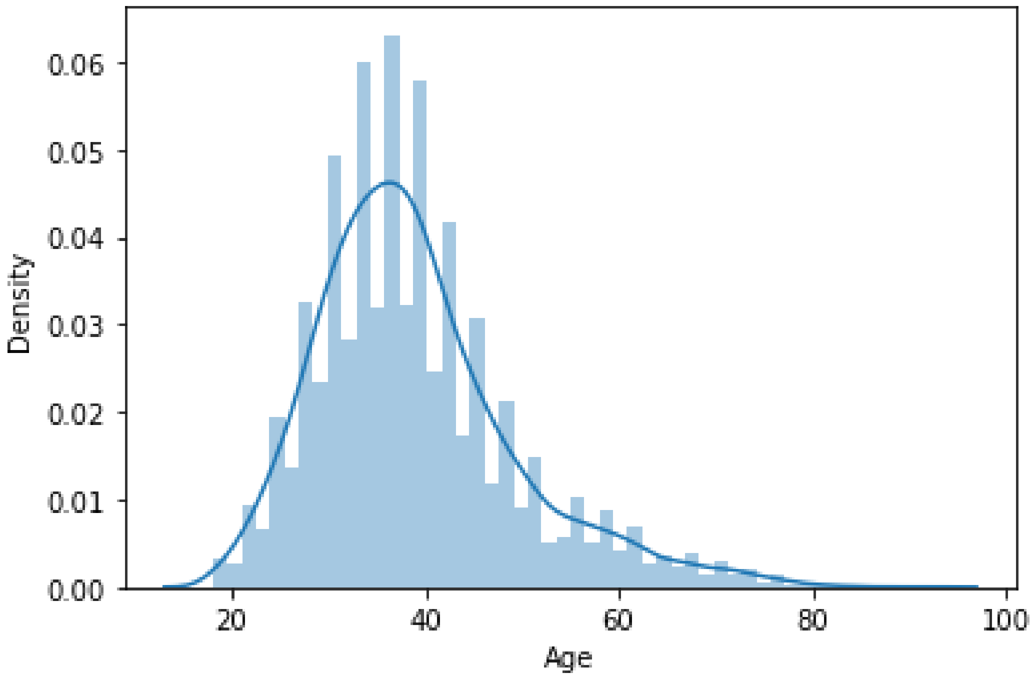
● Univariate Analysis ● Bi - Variate Analysis ● Multivariate Analysis

```
sns.distplot(df['Age'])
```

```
C:\Users\AdhoLOKHAM\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated 
function and will be removed in a future version. Please adapt your
```

```
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

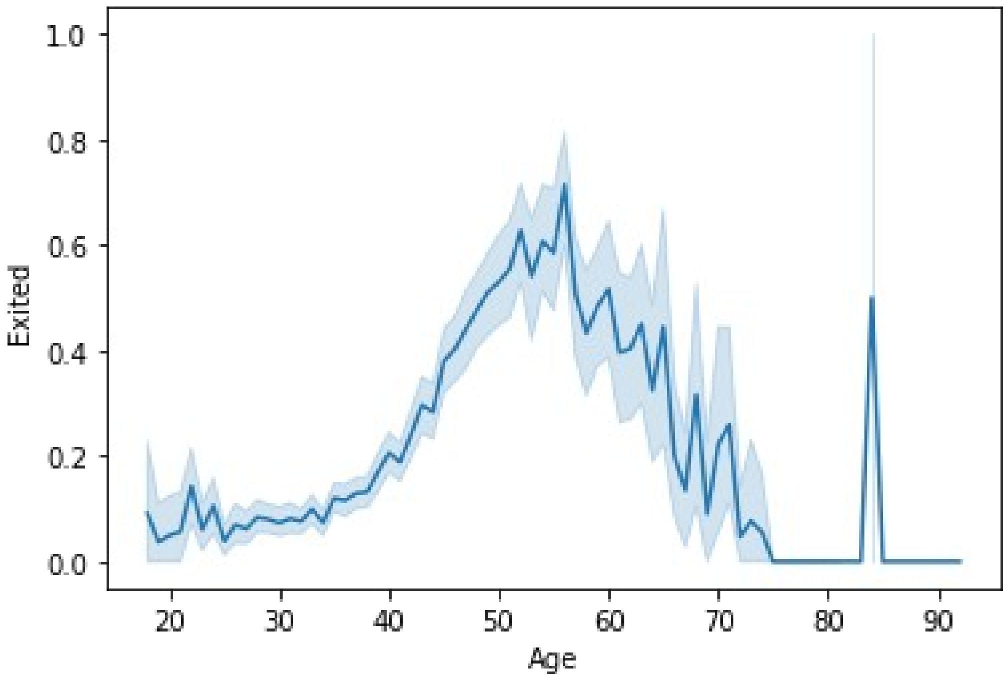
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



```
sns.lineplot(df['Age'],df['Exited'])
```

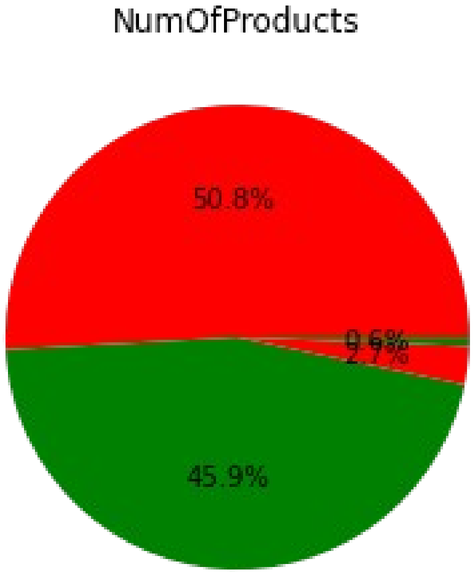
```
C:\Users\AdhoLOKHAM\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variables as
keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Age', ylabel='Exited'>
```



```
plt.pie(df.NumOfProducts.value_counts(),colors=['red','green'],autopct='%.1f%%')
plt.title('NumOfProducts')
```

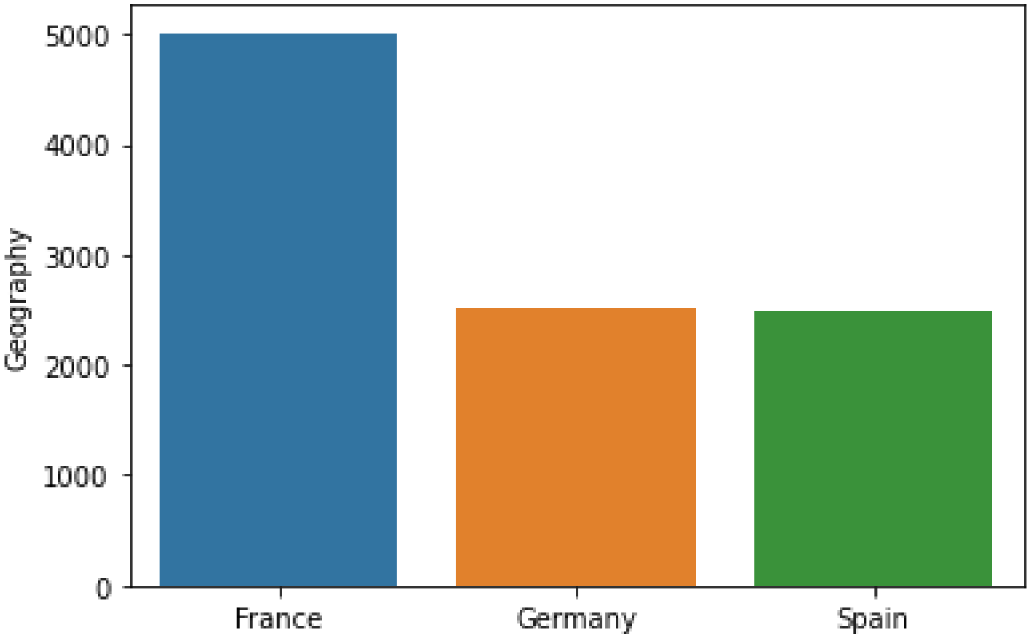
```
Text(0.5, 1.0, 'NumOfProducts')
```



```
sns.barplot((df.Geography.value_counts()).index,df.Geography.value_counts())
```

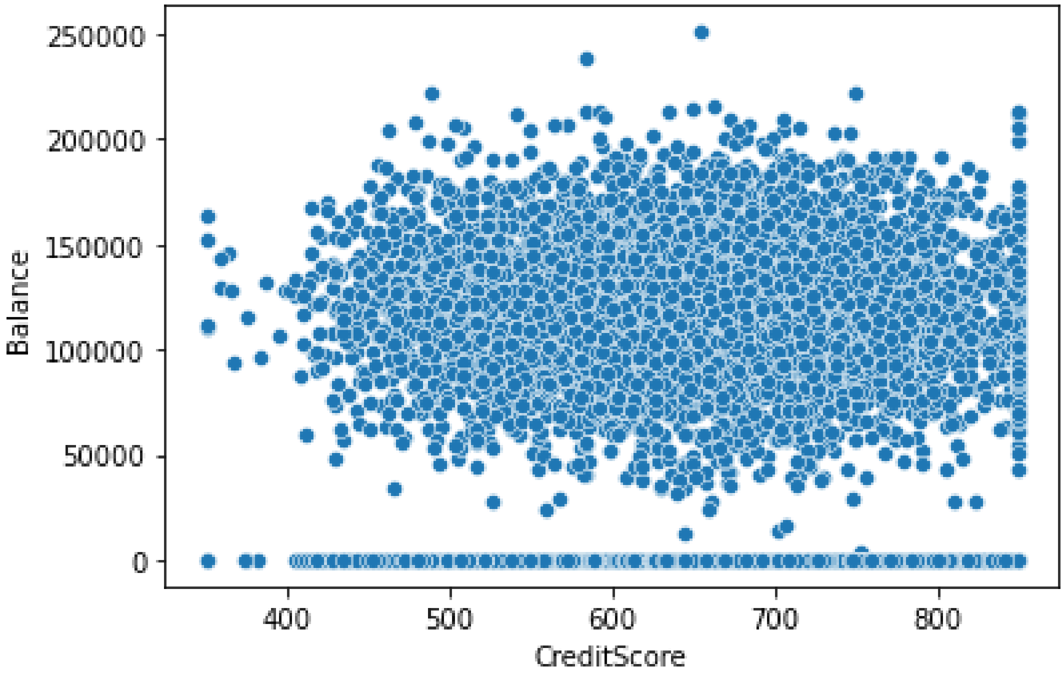
```
C:\Users\AdhoLOKHAM\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variables as
keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:ylabel='Geography'>
```



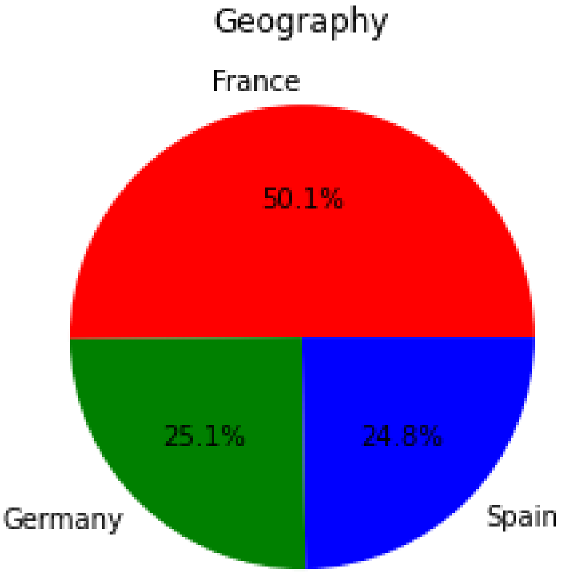
```
sns.scatterplot(x=df.CreditScore,y=df.Balance)
```

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



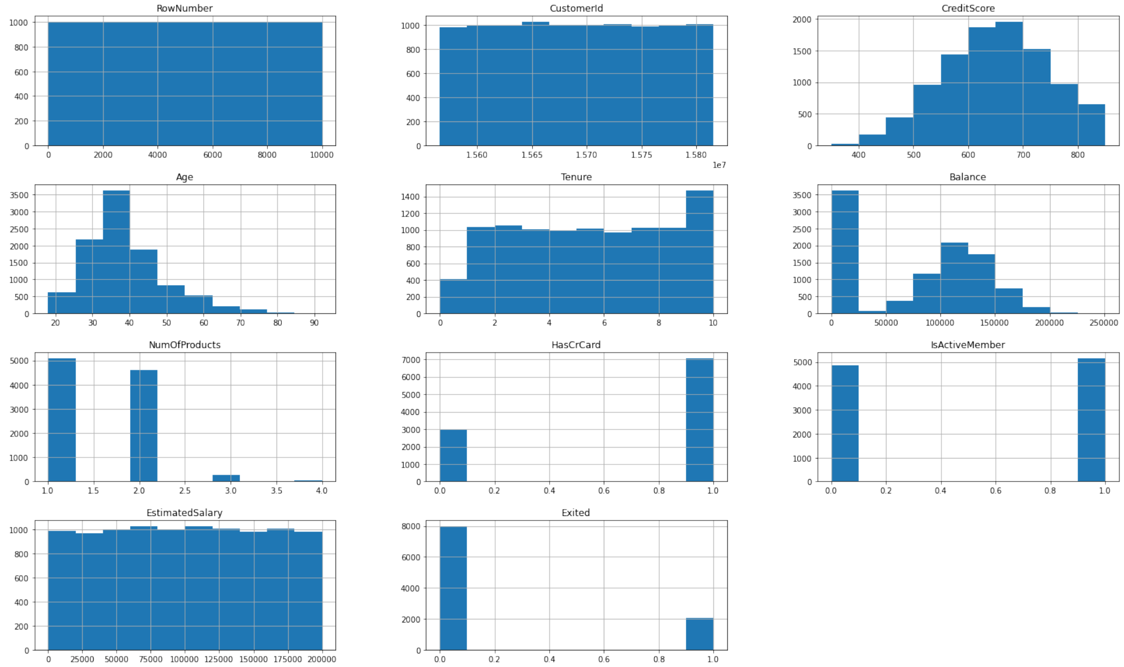
```
plt.pie(df.Geography.value_counts(),colors=['red','green','blue'],labe
ls=['France','Germany','Spain'],autopct='%.1f%%')
plt.title('Geography')
```

```
Text(0.5, 1.0, 'Geography')
```



```
df.hist(figsize=(25,15))
```

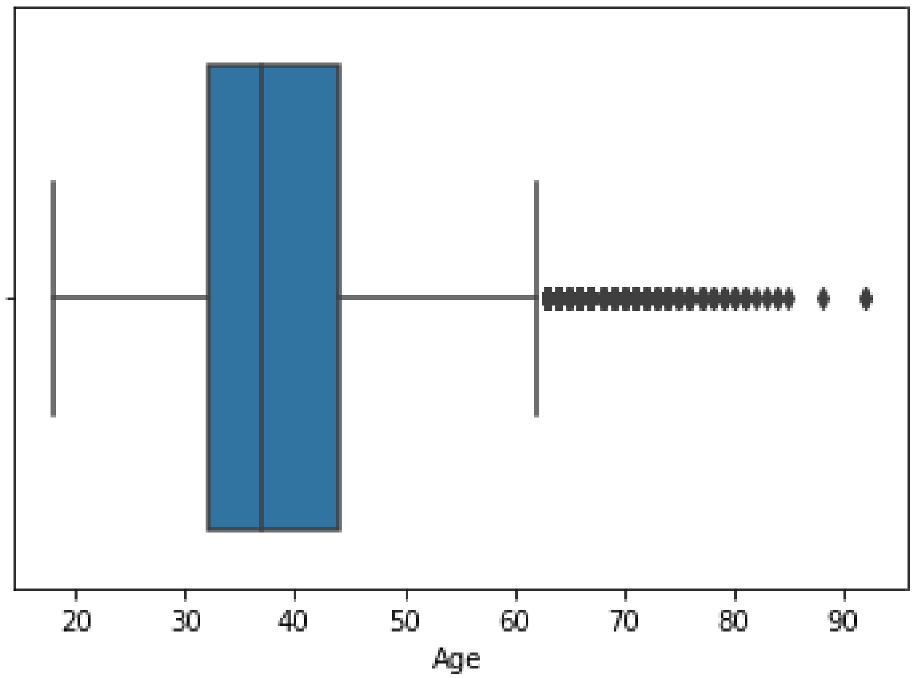
```
array([[<AxesSubplot:title={'center':'RowNumber'}>,
        <AxesSubplot:title={'center':'CustomerId'}>,
        <AxesSubplot:title={'center':'CreditScore'}>],
       [<AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Tenure'}>,
        <AxesSubplot:title={'center':'Balance'}>],
       [<AxesSubplot:title={'center':'NumOfProducts'}>,
        <AxesSubplot:title={'center':'HasCrCard'}>,
        <AxesSubplot:title={'center':'IsActiveMember'}>],
       [<AxesSubplot:title={'center':'EstimatedSalary'}>,
        <AxesSubplot:title={'center':'Exited'}>, <AxesSubplot:>]],
      dtype=object)
```



```
sns.boxplot(df.Age)
```

```
C:\Users\AdhoLOKHAM\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

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



4. Perform descriptive statistics on the dataset.

```
 df.describe()

          RowNumber    CustomerId   CreditScore           Age
Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000
10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800
5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806
2.892174
min        1.00000  1.556570e+07    350.000000     18.000000
0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000
3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000
5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000
7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000
10.000000

              Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count    10000.000000   10000.000000  10000.00000    10000.000000
mean     76485.889288       1.530200      0.70550        0.515100
std      62397.405202       0.581654      0.45584        0.499797
min          0.000000       1.000000      0.00000        0.000000
```

```
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

```
df.mean()
```

```
C:\Users\ADHOLO~1\AppData\Local\Temp/ipykernel_2088/3698961737.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  df.mean()
```

```
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
```

```
df.median()
```

```
C:\Users\ADHOLO~1\AppData\Local\Temp/ipykernel_2088/530051474.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  df.median()
```

```
RowNumber          5.000500e+03
CustomerId         1.569074e+07
CreditScore        6.520000e+02
Age                3.700000e+01
Tenure             5.000000e+00
```

```
Balance            9.719854e+04
NumOfProducts      1.000000e+00
HasCrCard          1.000000e+00
IsActiveMember     1.000000e+00
EstimatedSalary    1.001939e+05
Exited             0.000000e+00
dtype: float64
```

```
df.mode()
```

```
      RowNumber  CustomerId Surname  CreditScore Geography Gender
Age  \
0             1    15565701   Smith        850.0    France   Male
37.0
1             2    15565706     NaN          NaN       NaN    NaN
NaN
2             3    15565714     NaN          NaN       NaN    NaN
NaN
3             4    15565779     NaN          NaN       NaN    NaN
NaN
4             5    15565796     NaN          NaN       NaN    NaN
NaN
...         ...         ...     ...          ...       ...    ...   ..
.
9995       9996    15815628     NaN          NaN       NaN    NaN
NaN
9996       9997    15815645     NaN          NaN       NaN    NaN
NaN
9997       9998    15815656     NaN          NaN       NaN    NaN
NaN
9998       9999    15815660     NaN          NaN       NaN    NaN
NaN
9999      10000    15815690     NaN          NaN       NaN    NaN
NaN

      Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0        2.0      0.0            1.0        1.0             1.0
1        NaN      NaN            NaN        NaN             NaN
2        NaN      NaN            NaN        NaN             NaN
3        NaN      NaN            NaN        NaN             NaN
4        NaN      NaN            NaN        NaN             NaN
...      ...      ...            ...        ...             ...
9995     NaN      NaN            NaN        NaN             NaN
9996     NaN      NaN            NaN        NaN             NaN
9997     NaN      NaN            NaN        NaN             NaN
9998     NaN      NaN            NaN        NaN             NaN
9999     NaN      NaN            NaN        NaN             NaN

      EstimatedSalary  Exited
0            24924.92     0.0
```

```
1                 NaN     NaN
2                 NaN     NaN
3                 NaN     NaN
4                 NaN     NaN
...               ...     ...
9995              NaN     NaN
9996              NaN     NaN
9997              NaN     NaN
9998              NaN     NaN
9999              NaN     NaN

[10000 rows x 14 columns]
```

```
df.std()
```

```
C:\Users\ADHOLO~1\AppData\Local\Temp/ipykernel_2088/3390915376.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  df.std()

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
```

```
df.head()
```

```
   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


   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

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

5. Handle the 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
```

```
df['Age'].fillna(df['Age'].mean(),inplace=True)
```

```
df.head()
```

```
   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   
```

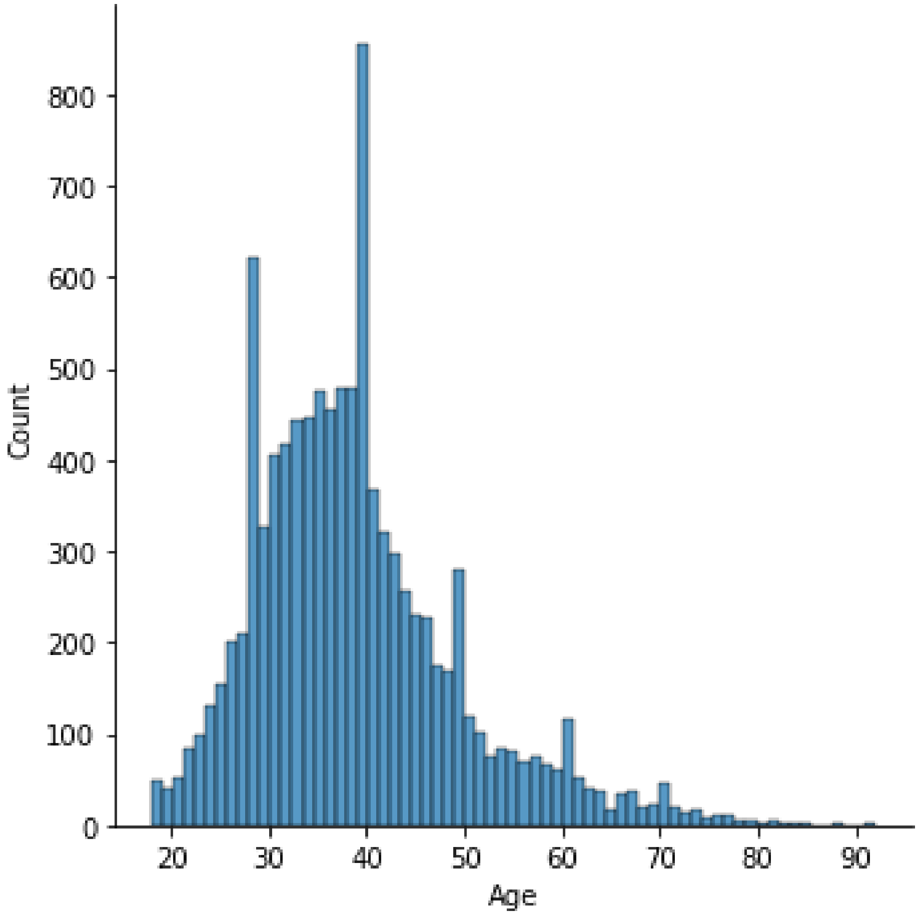
```
   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1   
1       1   83807.86              1          0               1   
2       8  159660.80              3          1               0   
3       1       0.00              2          0               0   
4       2  125510.82              1          1               1   

   EstimatedSalary  Exited  
0        101348.88       1  
1        112542.58       0  
2        113931.57       1  
3         93826.63       0  
4         79084.10       0  
```

6. Find the outliers and replace the outliers

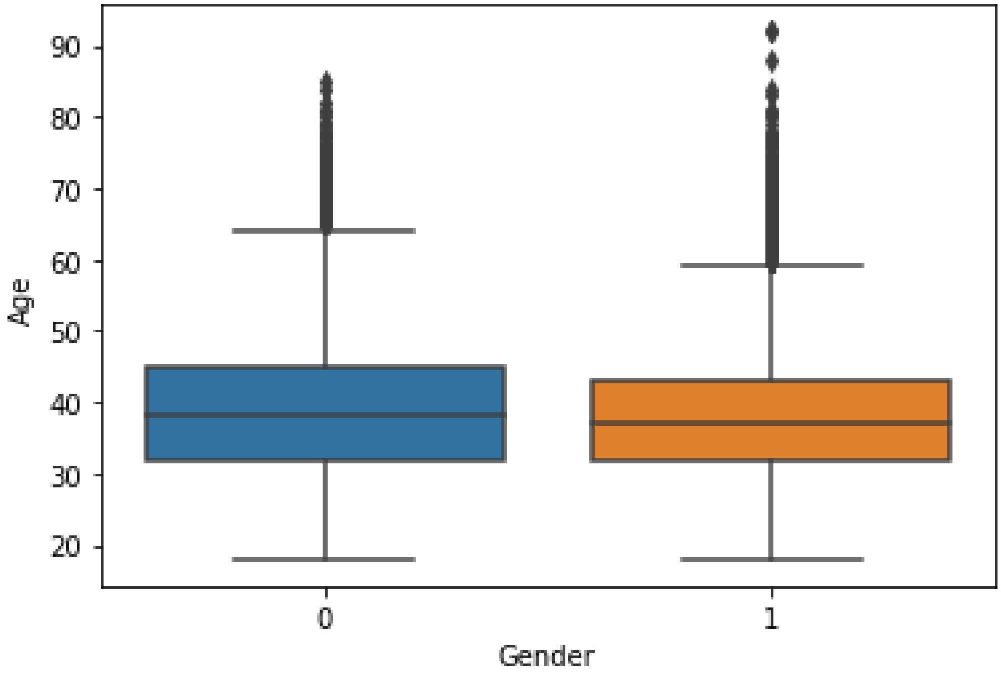
```
sns.displot(df['Age'])
```

```
<seaborn.axisgrid.FacetGrid at 0x1ced3490fd0>
```



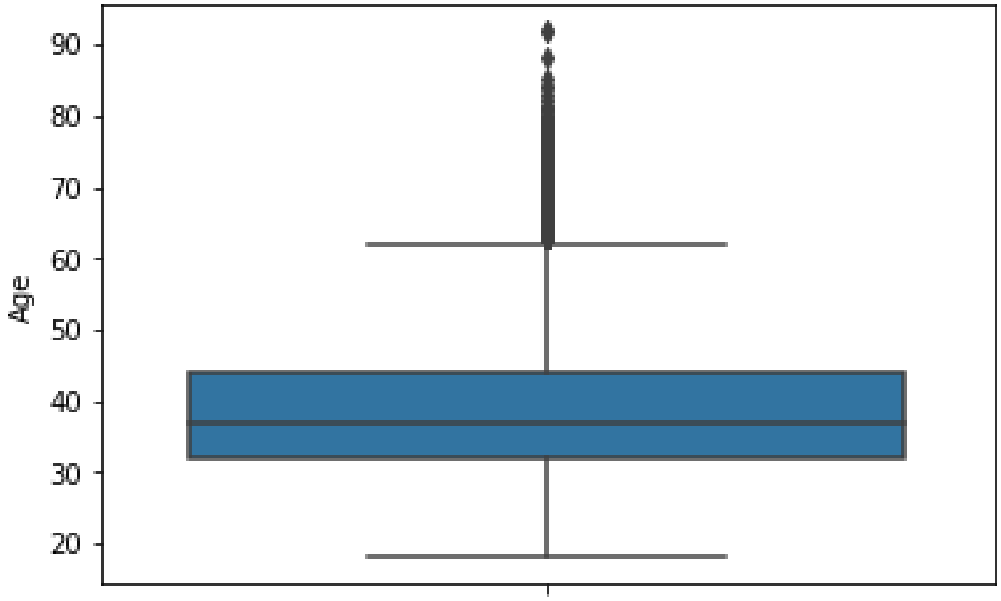
```
sns.boxplot(x='Gender',y='Age',data=df)
```

```
<AxesSubplot:xlabel='Gender', ylabel='Age'>
```



```
sns.boxplot(y='Age',data=df)
```

```
<AxesSubplot:ylabel='Age'>
```



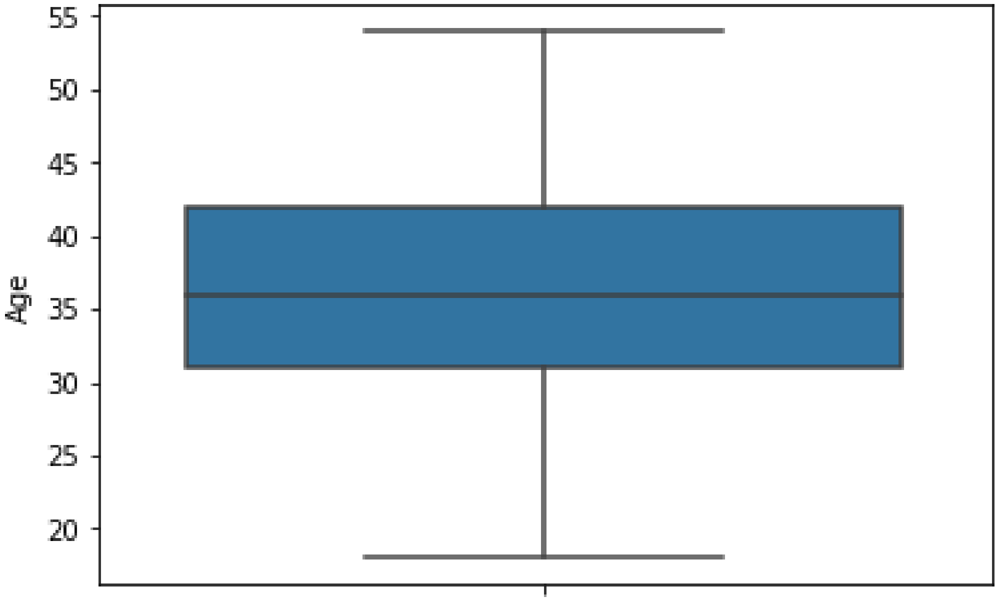
```
df['Age'].mean()
```

```
38.9218
```

```
df1=df[df['Age']<55]
```

```
sns.boxplot(y='Age',data=df1)
```

```
<AxesSubplot:ylabel='Age'>
```

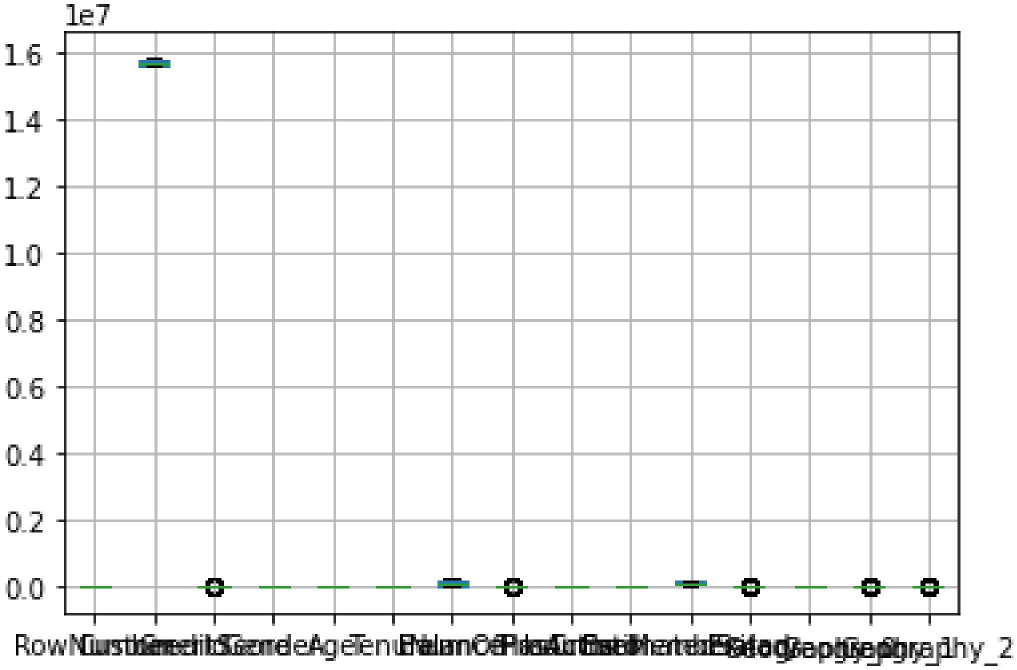


```
df1['Age'].mean()
```

```
36.62250493529283
```

```
df1.boxplot()
```

```
<AxesSubplot:>
```



7. Check for Categorical columns and perform encoding.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Geography']= le.fit_transform(df['Geography'])
df.Gender= le.fit_transform(df.Gender)
df['Geography'].unique()
```

```
array([0, 2, 1], dtype=int64)
```

```
df['Gender'].unique()
```

```
array([0, 1], dtype=int64)
```

```
df=pd.get_dummies(df,columns=['Geography'])
```

```
df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore  Gender  Age  
Tenure  \
0          1    15634602  Hargrave          619  Female   42       2

1          2    15647311      Hill          608  Female   41       1

2          3    15619304      Onio          502  Female   42       8

3          4    15701354      Boni          699  Female   39       1

4          5    15737888  Mitchell          850  Female   43       2


     Balance  NumOfProducts  HasCrCard  IsActiveMember  
EstimatedSalary  \
0       0.00              1          1               1
101348.88
1   83807.86              1          0               1
112542.58
2  159660.80              3          1               0
113931.57
3       0.00              2          0               0
93826.63
4  125510.82              1          1               1
79084.10

   Exited  Geography_France  Geography_Germany  Geography_Spain
0       1                 1                  0                0
1       0                 0                  0                1
2       1                 1                  0                0
3       0                 1                  0                0
4       0                 0                  0                1
```

8. Split the data into dependent and independent variables.

```
df.head()

   RowNumber  CustomerId   Surname  CreditScore  Gender  Age  
Tenure  \
0          1    15634602  Hargrave          619       0   42       2   

1          2    15647311      Hill          608       0   41       1   

2          3    15619304      Onio          502       0   42       8   

3          4    15701354      Boni          699       0   39       1   

4          5    15737888  Mitchell          850       0   43       2   


     Balance  NumOfProducts  HasCrCard  IsActiveMember  
EstimatedSalary  \
0       0.00              1          1               1        
101348.88   
1   83807.86              1          0               1        
112542.58   
2  159660.80              3          1               0        
113931.57   
3       0.00              2          0               0         
93826.63   
4  125510.82              1          1               1         
79084.10   

   Exited  Geography_0  Geography_1  Geography_2  
0       1            1            0            0  
1       0            0            0            1  
2       1            1            0            0  
3       0            1            0            0  
4       0            0            0            1  

#Independent variable
x=df.iloc[:,[4,5,6,11,12]]
x.head()

   Gender  Age  Tenure  EstimatedSalary  Exited
0       0   42       2        101348.88       1
1       0   41       1        112542.58       0
2       0   42       8        113931.57       1
3       0   39       1         93826.63       0
4       0   43       2         79084.10       0

#Dependent variable
y=df.iloc[:,[13]]
y.head()
```

```
   Geography_0
0            1
1            0
2            1
3            1
4            0
```

9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
#x_train = sc.fit_transform(x_train)
data=ss.fit_transform(x)
```

```
data
```

```
array([[-1.09598752,  0.29351742, -1.04175968,  0.02188649,
1.97716468],
       [-1.09598752,  0.19816383, -1.38753759,  0.21653375, -
0.50577476],
       [-1.09598752,  0.29351742,  1.03290776,  0.2406869 ,
1.97716468],
       ...,
       [-1.09598752, -0.27860412,  0.68712986, -1.00864308,
1.97716468],
       [ 0.91241915,  0.29351742, -0.69598177, -0.12523071,
1.97716468],
       [-1.09598752, -1.04143285, -0.35020386, -1.07636976, -
0.50577476]])
```

```
from sklearn.preprocessing import scale
scale=pd.DataFrame(scale(x),columns=x.columns)
```

```
scale
```

```
        Gender       Age    Tenure  EstimatedSalary    Exited
0    -1.095988  0.293517 -1.041760         0.021886  1.977165
1    -1.095988  0.198164 -1.387538         0.216534 -0.505775
2    -1.095988  0.293517  1.032908         0.240687  1.977165
3    -1.095988  0.007457 -1.387538        -0.108918 -0.505775
4    -1.095988  0.388871 -1.041760        -0.365276 -0.505775
...        ...       ...       ...              ...       ...
9995  0.912419  0.007457 -0.004426        -0.066419 -0.505775
9996  0.912419 -0.373958  1.724464         0.027988 -0.505775
9997 -1.095988 -0.278604  0.687130        -1.008643  1.977165
9998  0.912419  0.293517 -0.695982        -0.125231  1.977165
9999 -1.095988 -1.041433 -0.350204        -1.076370 -0.505775

[10000 rows x 5 columns]
```

10. Split the data into training and testing

```
x=df.iloc[:,0:15].values
y=df.iloc[:,15:16].values
```

```
x.shape
```

```
(10000, 15)
```

```
y.shape
```

```
(10000, 1)
```

```
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
```

```
(8000, 15)
```

```
y_train.shape
```

```
(8000, 1)
```