Loading necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
```

#Loading the dataset

```
from google.colab import drive
drive.mount("/content/gdrive")

df = pd.read_csv("/content/gdrive/My Drive/Sample
Data/Churn_Modelling.csv",header = 0)
```

```
Mounted at /content/gdrive
```

```
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
```

```
df.shape
```

```
(10000, 14)
```

```
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
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
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
```

```
ma = df.max()
ma
```

```
RowNumber              10000
CustomerId          15815690
```

```
Surname               Zuyeva
CreditScore              850
Geography              Spain
Gender                  Male
Age                       92
Tenure                    10
Balance             250898.09
NumOfProducts               4
HasCrCard                   1
IsActiveMember              1
EstimatedSalary    199992.48
Exited                      1
dtype: object
```

```
mi = df.min()
mi
```

```
RowNumber                1
CustomerId          15565701
Surname                Abazu
CreditScore              350
Geography             France
Gender                Female
Age                       18
Tenure                     0
Balance                  0.0
NumOfProducts              1
HasCrCard                  0
IsActiveMember             0
EstimatedSalary        11.58
Exited                     0
dtype: object
```

```
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
```

#Measure of central tendency

#Mean

```
df.mean()/100
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.

RowNumber             50.005000
CustomerId         156909.405694
CreditScore            6.505288
Age                    0.389218
Tenure                 0.050128
Balance              764.858893
NumOfProducts          0.015302
HasCrCard              0.007055
IsActiveMember         0.005151
EstimatedSalary     1000.902399
Exited                 0.002037
dtype: float64
```

#Median

```
df.median()/100
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

```
RowNumber             50.00500
CustomerId         156907.38000
CreditScore             6.52000
Age                     0.37000
Tenure                  0.05000
Balance               971.98540
NumOfProducts           0.01000
HasCrCard               0.01000
IsActiveMember          0.01000
EstimatedSalary      1001.93915
Exited                  0.00000
dtype: float64
```

Mode

```
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]
```

#Data Visualization

#Univariate Analysis Univariate analysis is the simplest form of analysis where we explore a single variable.

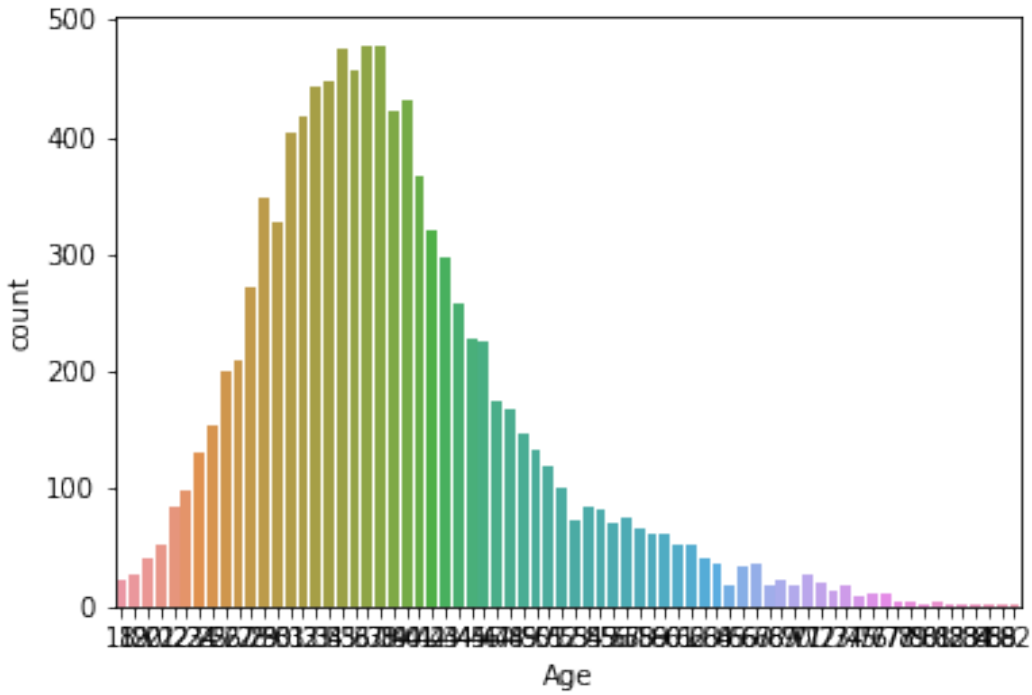
Count Plot

This gives a count of frequency plot in the form of a bar graph

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16252859d0>
```

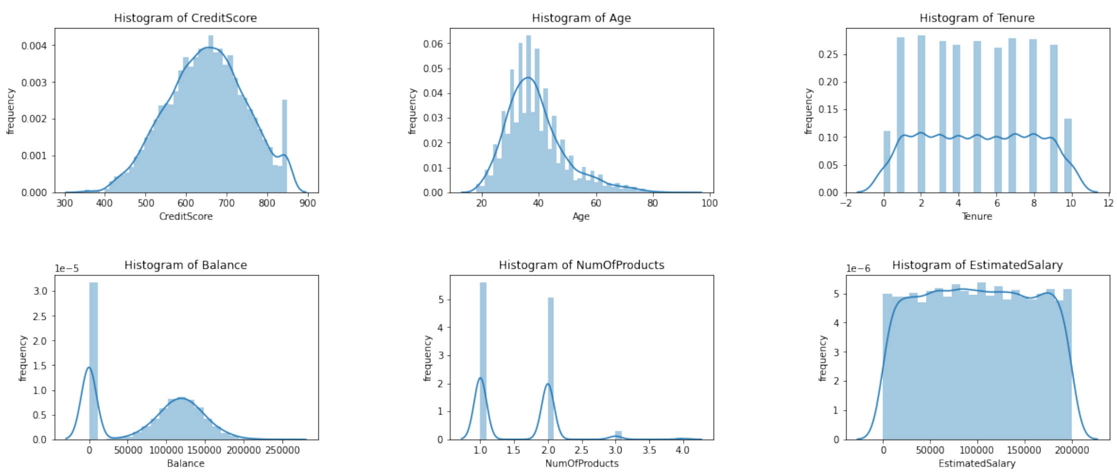


#Histogram

```
plt.figure(1,figsize=(20,8))
n=0
for x in
["CreditScore","Age","Tenure","Balance","NumOfProducts","EstimatedSala
ry"]:
    n += 1
    plt.subplot(2 ,3, n)
    plt.subplots_adjust(hspace=0.5,wspace=0.5)
    plt.ylabel("frequency")
    sns.distplot(df[x])
    plt.title('Histogram of {}'.format(x))
plt.show()
```

```
/usr/local/lib/python3.7/dist-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)
/usr/local/lib/python3.7/dist-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)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
```

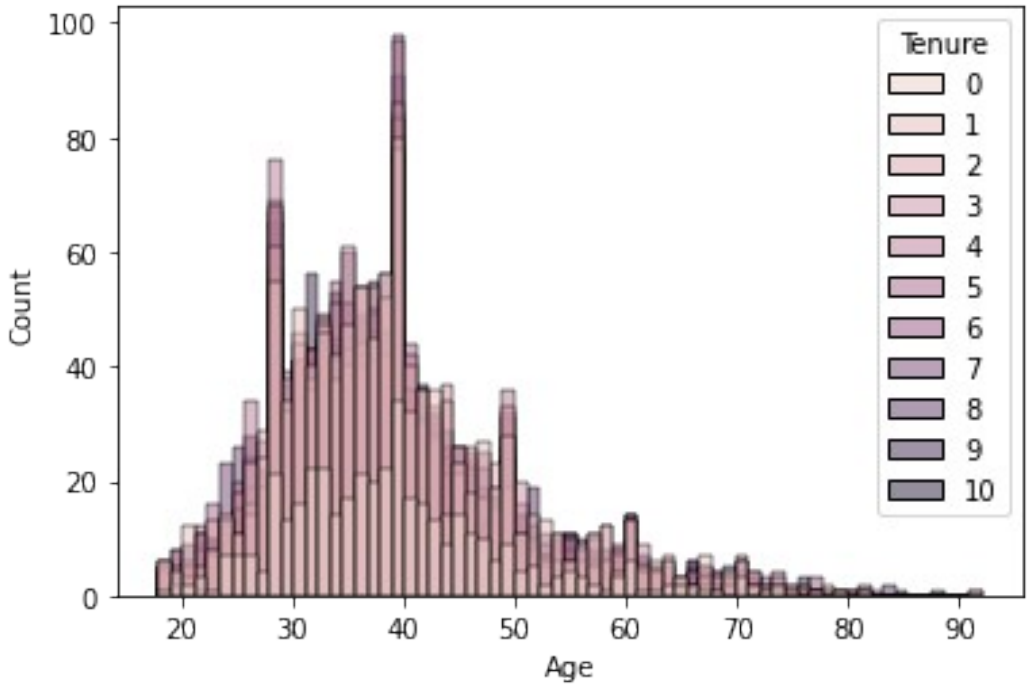
```
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` 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)
/usr/local/lib/python3.7/dist-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)
/usr/local/lib/python3.7/dist-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)
```



For categorical values

```
sns.histplot(x='Age',data=df,hue=df['Tenure'],color='blue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1624b4f050>
```

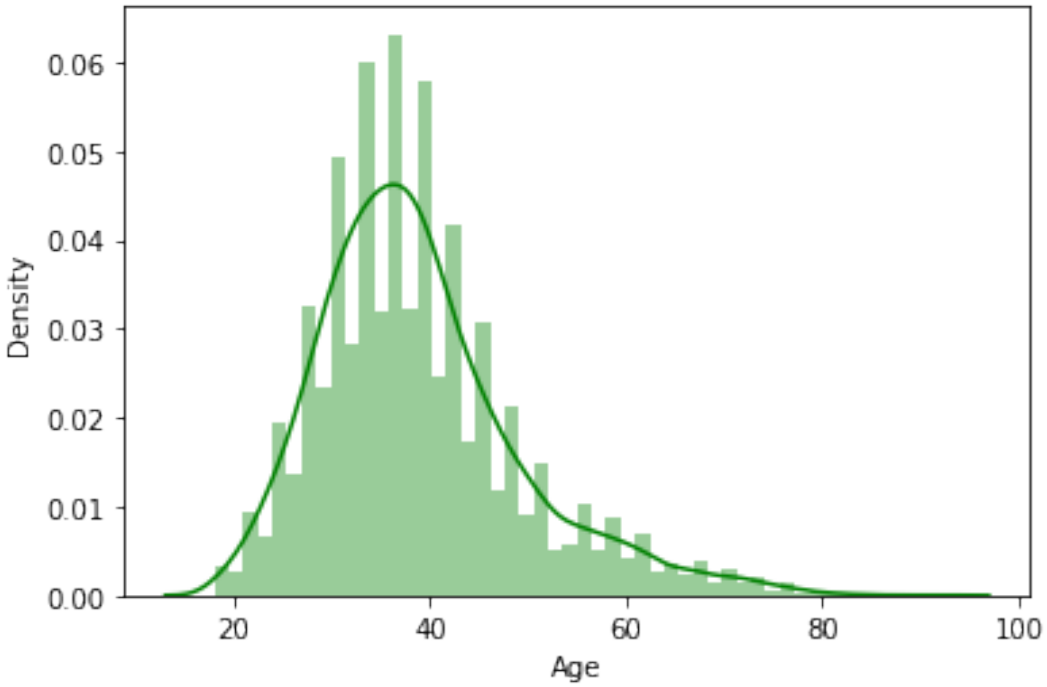


#Distplot

```
sns.distplot(df["Age"],color='green')
```

```
/usr/local/lib/python3.7/dist-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)

<matplotlib.axes._subplots.AxesSubplot at 0x7f162322d1d0>
```



#Skewness of dataset

```
df.skew()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

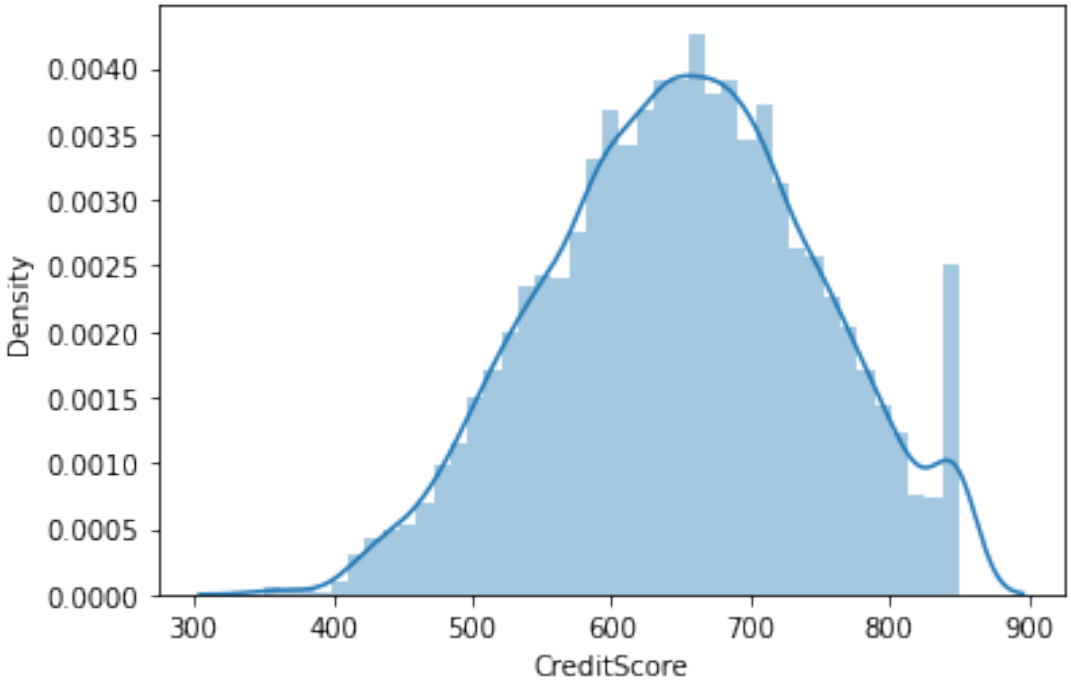
```
RowNumber          0.000000
CustomerId         0.001149
CreditScore       -0.071607
Age                1.011320
Tenure             0.010991
Balance           -0.141109
NumOfProducts      0.745568
HasCrCard         -0.901812
IsActiveMember    -0.060437
EstimatedSalary    0.002085
Exited             1.471611
dtype: float64
```

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

```
/usr/local/lib/python3.7/dist-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)
```

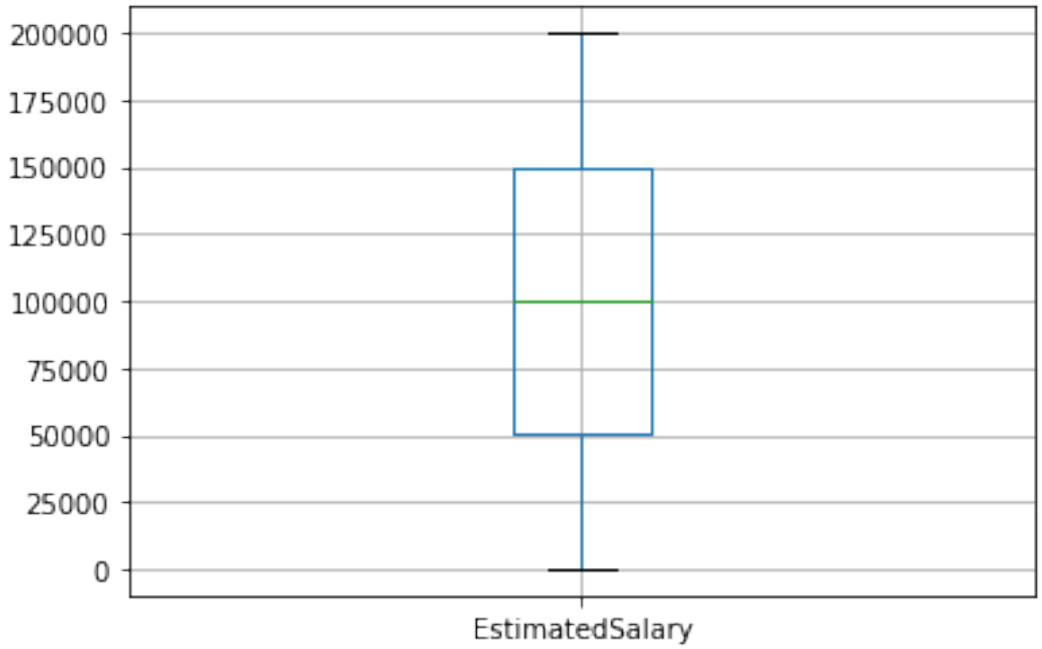
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16230d8190>
```



#Boxplot

```
df.boxplot(column=['EstimatedSalary'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f162470a350>
```



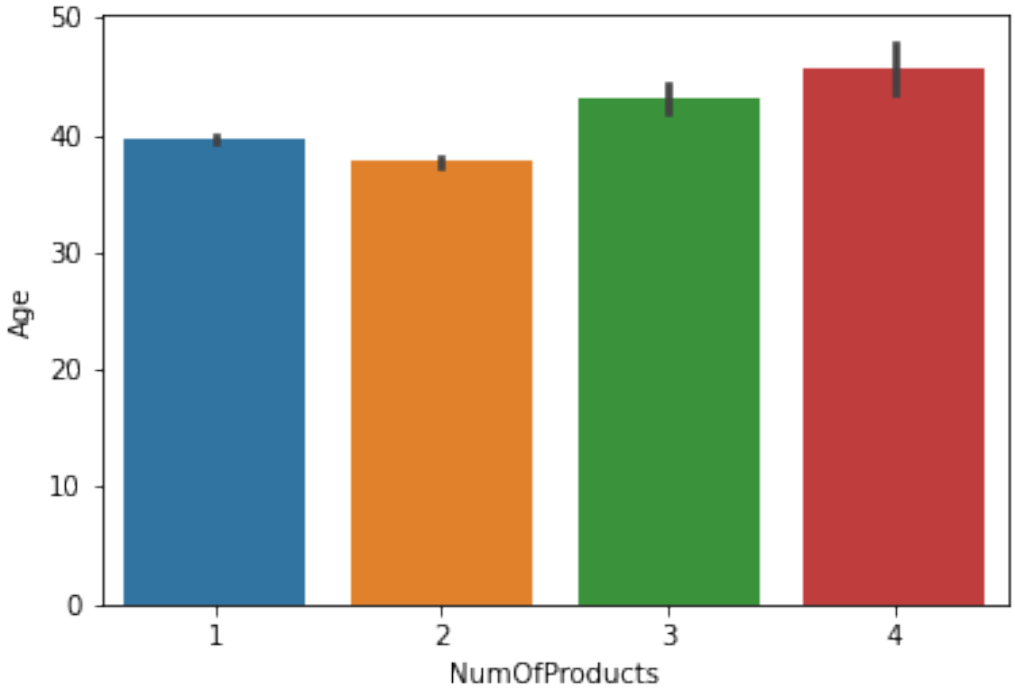
#Bivariate Analysis Bivariate Analysis is used when we have to explore the relationship between 2 different variables

Barplot

```
sns.barplot(df["NumOfProducts"],df["Age"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1623232210>
```

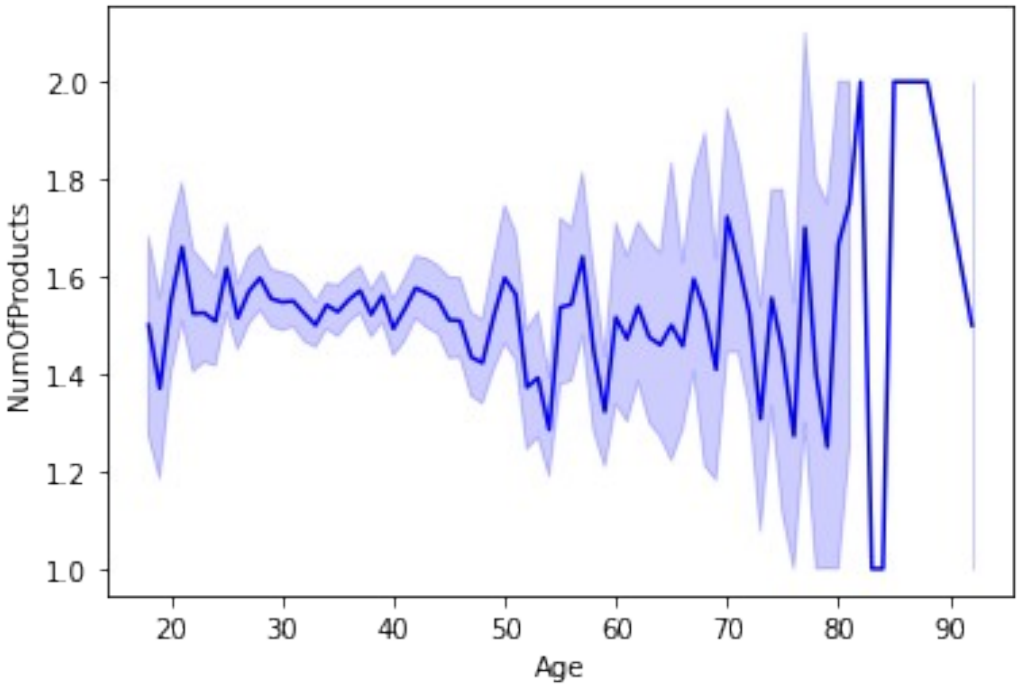


#Linear plot

```
sns.lineplot(df["Age"],df["NumOfProducts"], color='blue')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1622eede10>
```

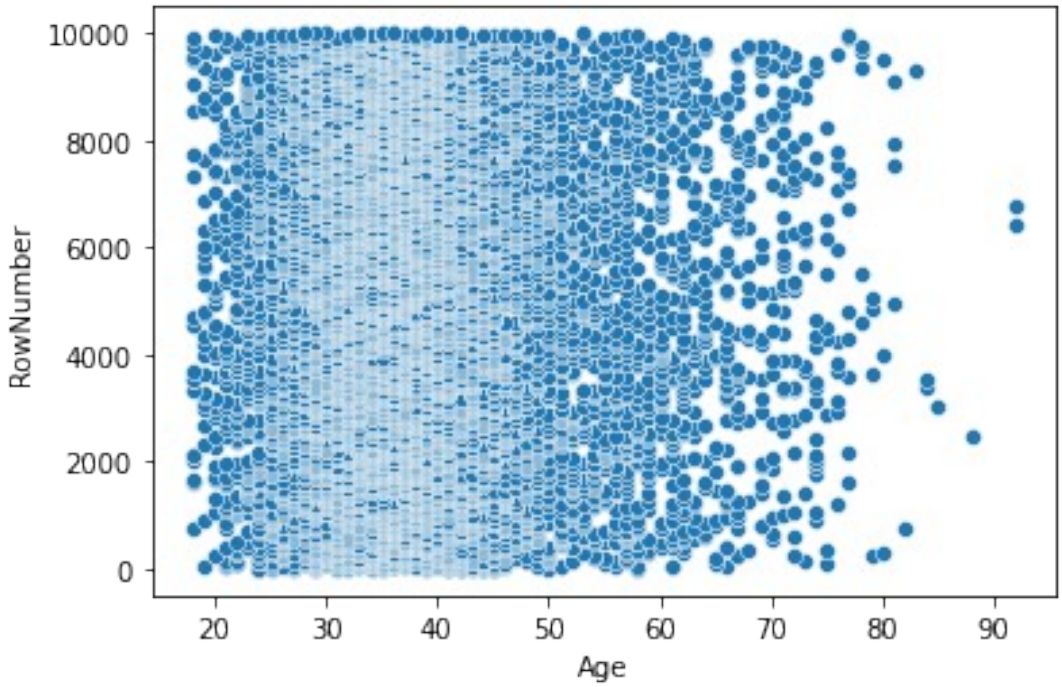


#Scatter Plot

```
sns.scatterplot(df['Age'], df['RowNumber'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

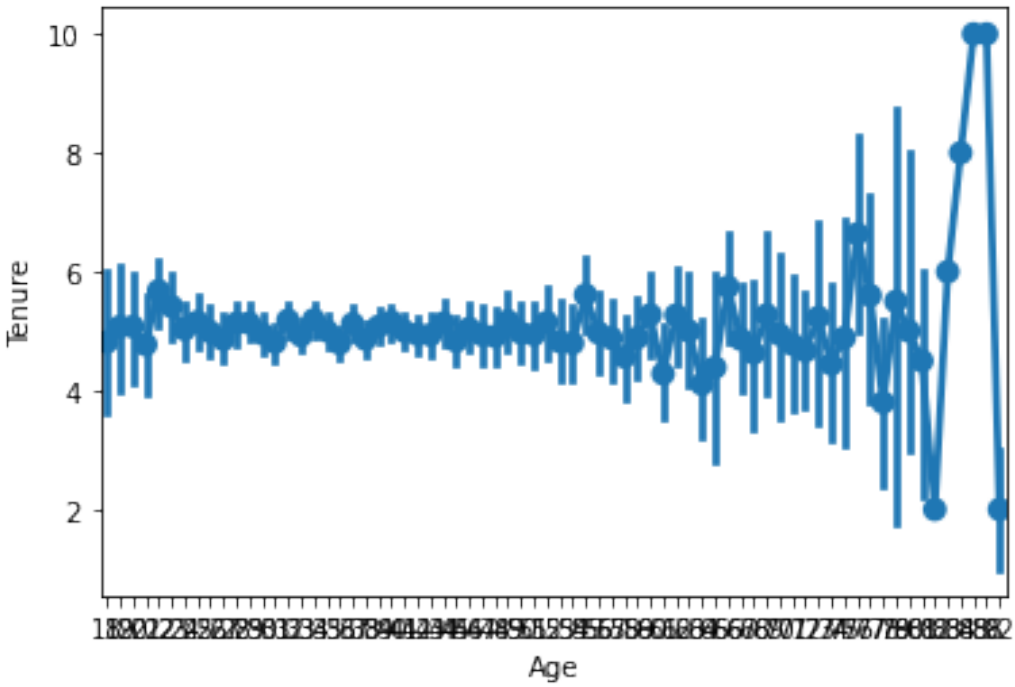
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1622e81c90>
```



#Point PLot

```
sns.pointplot(x='Age',y='Tenure',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1622d9a150>
```

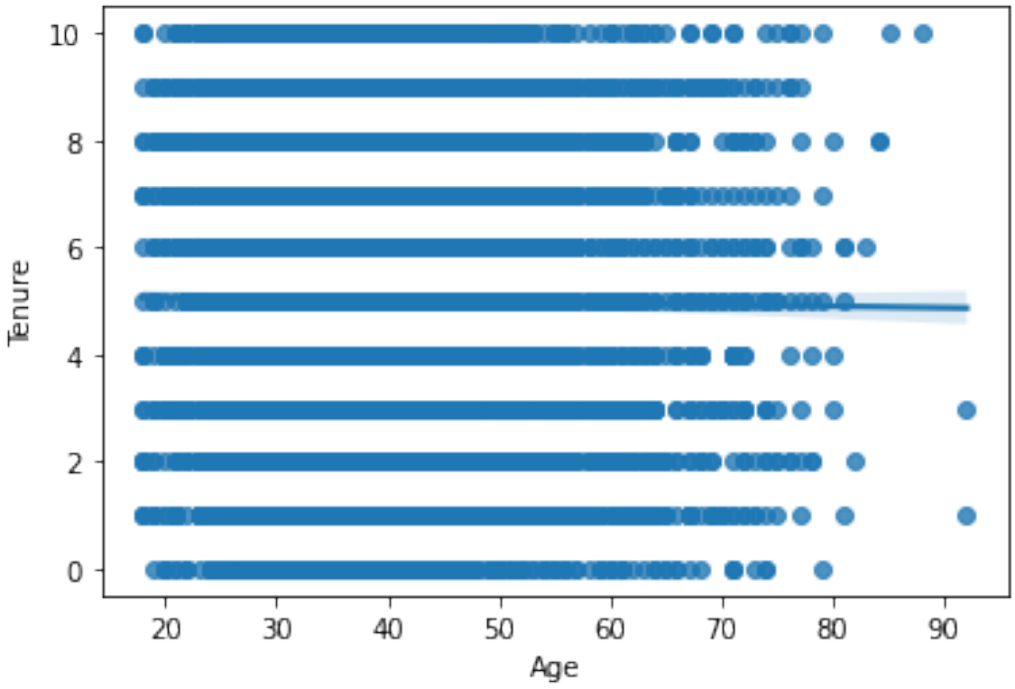


#Regplot

```
sns.regplot(df['Age'],df['Tenure'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

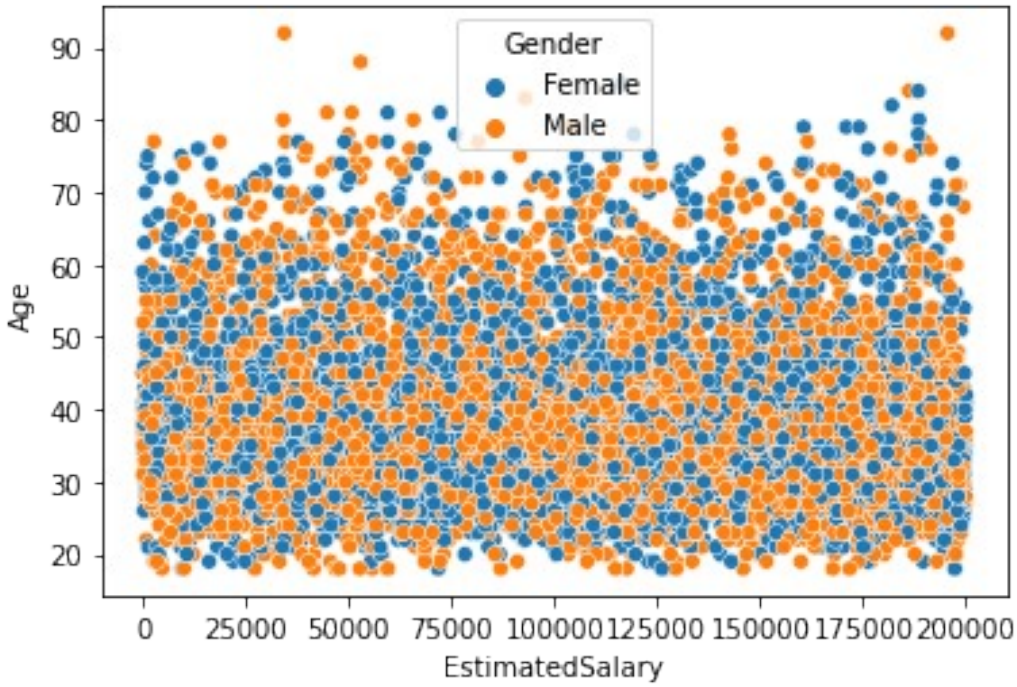
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1622bb91d0>
```



#Multivariate analysis

```
sns.scatterplot(x=df['EstimatedSalary'],y=df['Age'],hue=df["Gender"])
```

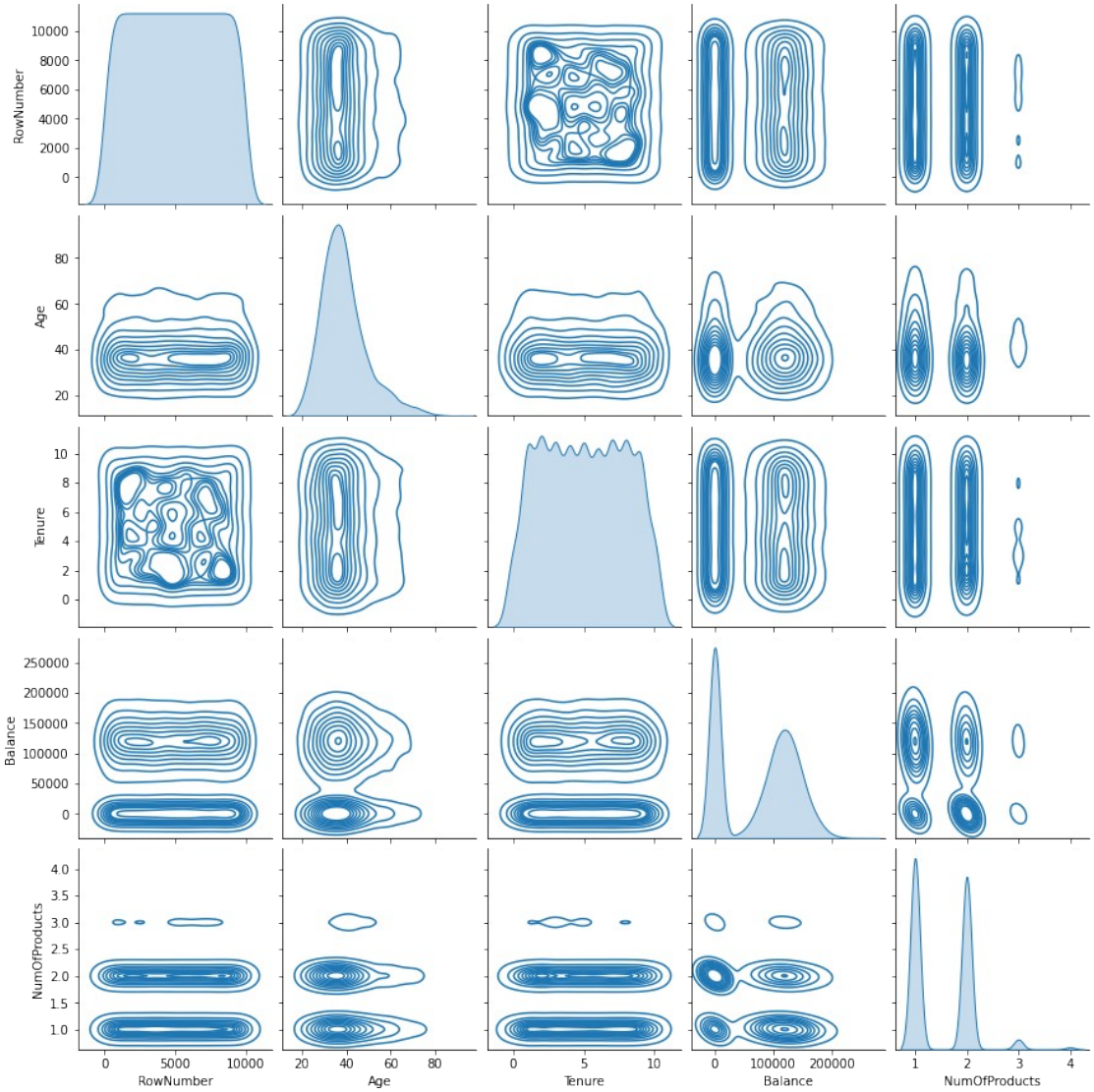
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1620314990>
```



#Pairplot

```
sns.pairplot(data=df[["RowNumber","Age","Tenure","Balance","NumOfProducts"]],kind="kde")
```

```
<seaborn.axisgrid.PairGrid at 0x7f16202bd7d0>
```



Handling missing values

```
df.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
```

The dataset provides doesn't have any null values. Creating a custom dataset to show the missing value handling

```
cust_dataset = pd.DataFrame({
    "a": [1,2,3],
    "b": [4,None,None],
    "c": [5,6,None]
})
cust_dataset
```

```
   a    b    c
0  1  4.0  5.0
1  2  NaN  6.0
2  3  NaN  NaN
```

```
cust_dataset.isnull().any()
```

```
a    False
b     True
c     True
dtype: bool
```

Fill the missing values using the mean

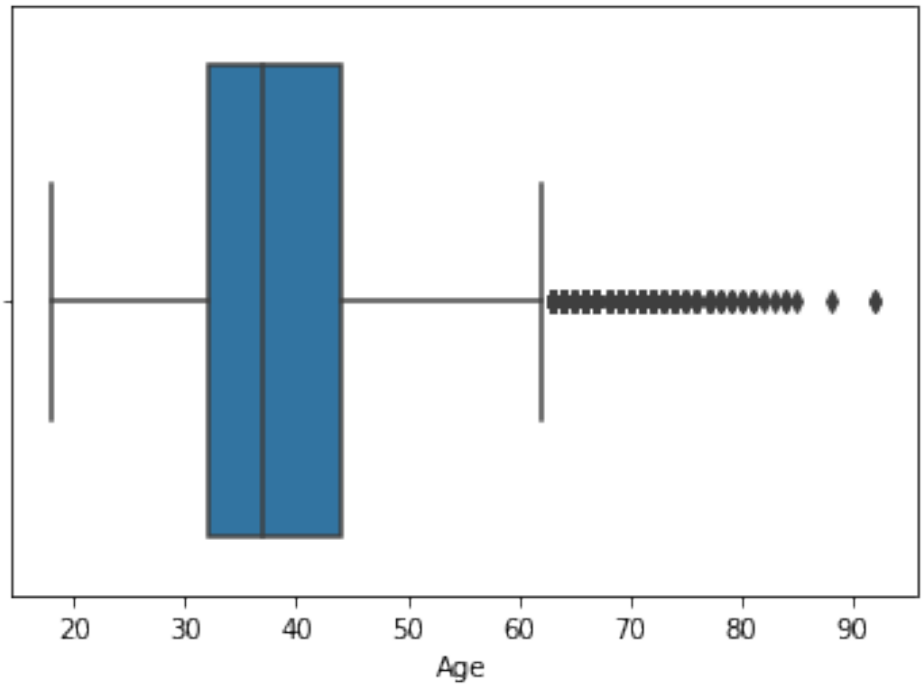
```
mean_values = cust_dataset.mean()
cust_dataset.fillna(mean_values, inplace=True)
cust_dataset
```

```
   a    b    c
0  1  4.0  5.0
1  2  4.0  6.0
2  3  4.0  5.5
```

#Identifying outliers and replacing them

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f162308ad90>
```

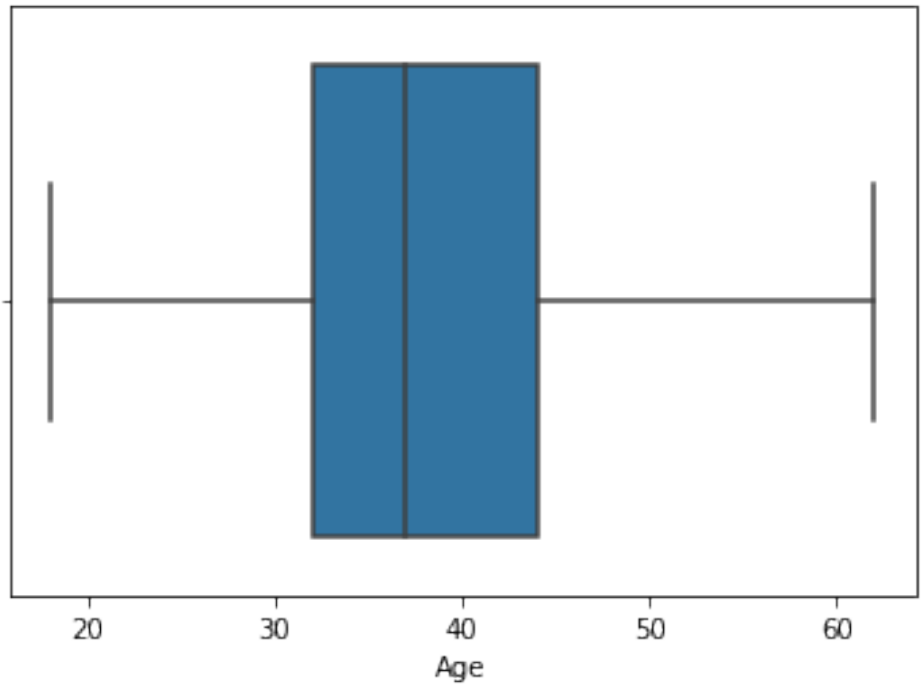



```
q1 = df['Age'].quantile(0.25)
q3 = df['Age'].quantile(0.75)
iqr = q3 - q1
iqr
```

```
12.0
```

```
df.loc[df['Age'] < q1 - 1.5*iqr, 'Age'] = q1
df.loc[df['Age'] > q3 + 1.5*iqr, 'Age'] = q3
sns.boxplot(x=df['Age'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1625ae8b50>
```



Checking for categorical columns and perform encoding

```
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
```

```
df["Geography"].unique()
```

```
array(['France', 'Spain', 'Germany'], dtype=object)
```

```
surnames = df["Surname"].unique()
surname_encoding = {}
for i in range(len(surnames)):
  surname_encoding[surnames[i]] = i
len(surname_encoding)
```

```
2932
```

the categorial columns are gender and geography and surname encoding

```
df["Geography"].replace({"France":0,"Spain":1,"Germany":3},
inplace=True)
df["Gender"].replace({"Female":0,"Male":1}, inplace=True)
df["Surname"].replace(surname_encoding, inplace=True)
```

```
df.head()
```

```
   RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age
\
0          1    15634602        0          619          0       0   42

1          2    15647311        1          608          1       0   41

2          3    15619304        2          502          0       0   42

3          4    15701354        3          699          0       0   39

4          5    15737888        4          850          1       0   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
```

Spliting the data into dependant and independant set

```
y = df['Exited']
X = df.drop(columns=['Exited'], axis=1)
```

```
X.head()
```

```
   RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age
\
0          1    15634602        0          619          0       0   42

1          2    15647311        1          608          1       0   41

2          3    15619304        2          502          0       0   42

3          4    15701354        3          699          0       0   39

4          5    15737888        4          850          1       0   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
0        101348.88
1        112542.58
2        113931.57
3         93826.63
4         79084.10
```

Scaling independant variables

```
column_name = X.columns
```

```
X = scale(X)
X
```

```
array([[-1.73187761, -0.78321342, -1.22470312, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-1.7315312 , -0.60653412, -1.22336735, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.73118479, -0.99588476, -1.22203159, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 1.73118479, -1.47928179, -0.01583695, ..., -1.54776799,
         0.97024255, -1.00864308],
```

```
       [ 1.7315312 , -0.11935577,  1.1569636 , ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 1.73187761, -0.87055909, -0.47667543, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

```
X = pd.DataFrame(X, columns =column_name)
X.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore  Geography
Gender  \
0  -1.731878   -0.783213 -1.224703    -0.326221  -0.815465 -1.095988

1  -1.731531   -0.606534 -1.223367    -0.440036  -0.000326 -1.095988

2  -1.731185   -0.995885 -1.222032    -1.536794  -0.815465 -1.095988

3  -1.730838    0.144767 -1.220696     0.501521  -0.815465 -1.095988

4  -1.730492    0.652659 -1.219360     2.063884  -0.000326 -1.095988


        Age    Tenure   Balance  NumOfProducts  HasCrCard
IsActiveMember  \
0  0.457039 -1.041760 -1.225848      -0.911583   0.646092
0.970243
1  0.342361 -1.387538  0.117350      -0.911583  -1.547768
0.970243
2  0.457039  1.032908  1.333053       2.527057   0.646092       -
1.030670
3  0.113004 -1.387538 -1.225848       0.807737  -1.547768       -
1.030670
4  0.571717 -1.041760  0.785728      -0.911583   0.646092
0.970243

   EstimatedSalary
0         0.021886
1         0.216534
2         0.240687
3        -0.108918
4        -0.365276
```

#Spliting data into training and testing datasets

```
x_train, x_test, y_train, y_test = train_test_split(X,y)
```

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

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