```python
import numpy as np
import pandas as pd

#1.loading the file
df=pd.read_csv('/content/Churn_Modelling.csv')
df.shape
```

```
(10000, 14)
```

```python
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
```

```python
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
```

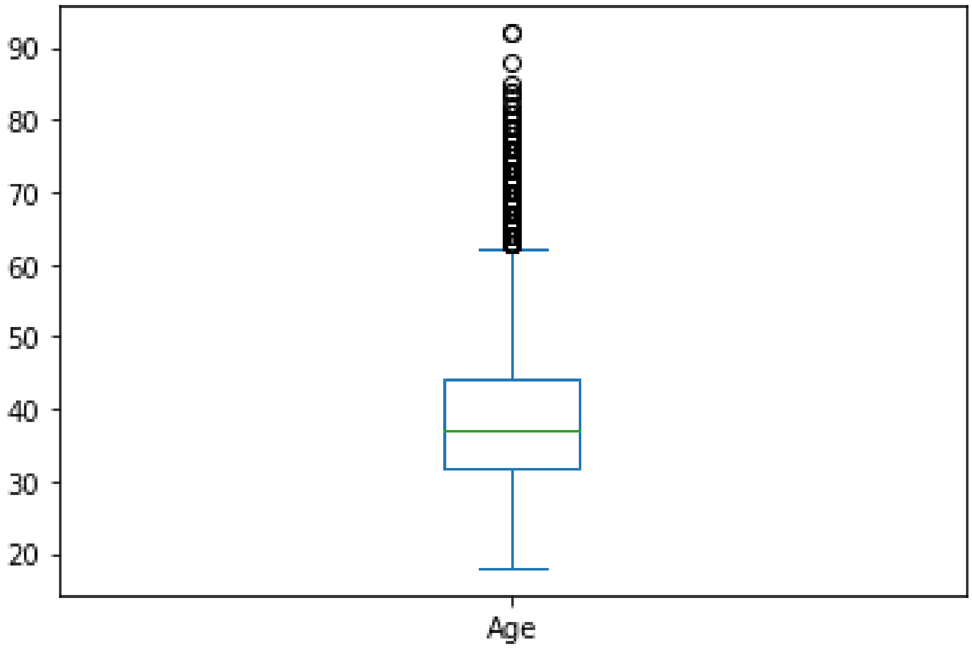
```
3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000
5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000
7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000
10.000000

             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.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

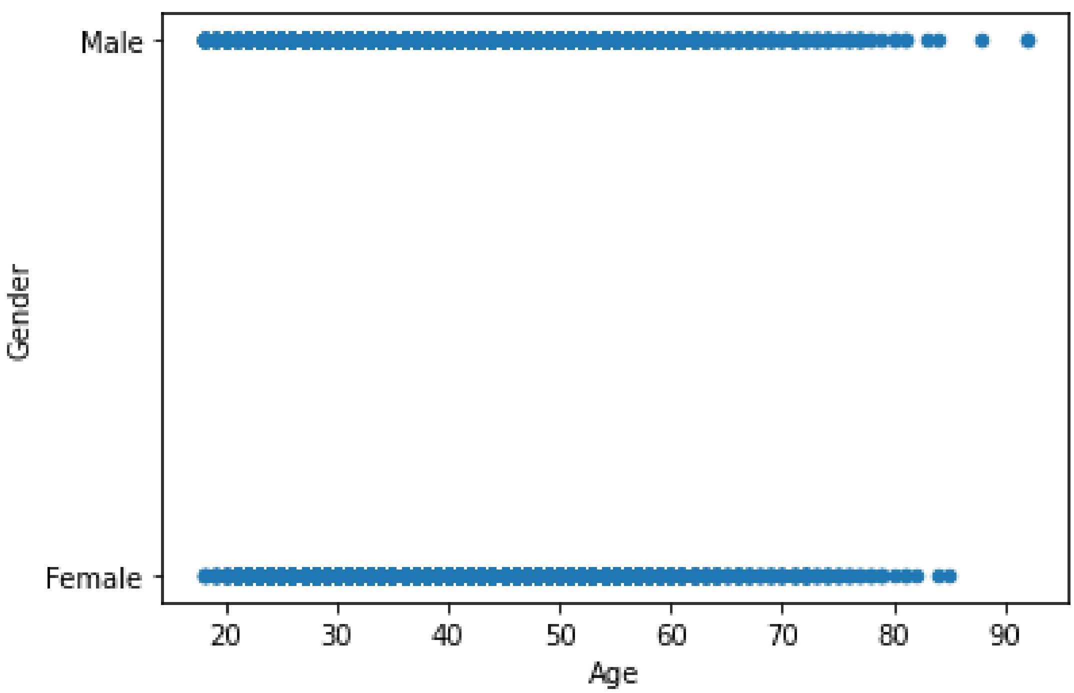
```
#2.1univarient
import matplotlib.pyplot as plt
%matplotlib inline
df['Age'].plot.box()
```

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7f6467a34e10&gt;
```




```
#2.2bivarient
df.plot.scatter('Age','Gender')
```

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7f64678caa50&gt;
```




```
df.corr()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001

	Balance	NumOfProducts	HasCrCard	IsActiveMember
RowNumber	-0.009067	0.007246	0.000599	0.012044
CustomerId	-0.012419	0.016972	-0.014025	0.001665
CreditScore	0.006268	0.012238	-0.005458	0.025651
Age	0.028308	-0.030680	-0.011721	0.085472
Tenure	-0.012254	0.013444	0.022583	-0.028362
Balance	1.000000	-0.304180	-0.014858	-0.010084
NumOfProducts	-0.304180	1.000000	0.003183	0.009612
HasCrCard	-0.014858	0.003183	1.000000	-0.011866
IsActiveMember	-0.010084	0.009612	-0.011866	1.000000
EstimatedSalary	0.012797	0.014204	-0.009933	-0.011421
Exited	0.118533	-0.047820	-0.007138	-0.156128

	EstimatedSalary	Exited
RowNumber	-0.005988	-0.016571
CustomerId	0.015271	-0.006248
CreditScore	-0.001384	-0.027094
Age	-0.007201	0.285323
Tenure	0.007784	-0.014001
Balance	0.012797	0.118533
NumOfProducts	0.014204	-0.047820
HasCrCard	-0.009933	-0.007138
IsActiveMember	-0.011421	-0.156128
EstimatedSalary	1.000000	0.012097
Exited	0.012097	1.000000

```
x=df.drop(['Exited'],axis=1).values
y=df['Exited'].values
```

```
#Descriptive statistics4.1.1  Measures of central tendency4.1.2
Measures of dispersion4.1.3  Summary statistics
```

```
df['Age'].mode()
round(df["Age"].mean(), 2)
df["Age"].median()
print(f'The median of age is {df["Age"].median()}')
```

```
The median of age is 37.0
```

```
df['Age'].quantile([.25, .5, .75])
```

```
0.25    32.0
0.50    37.0
0.75    44.0
Name: Age, dtype: float64
```

```
round(df.describe(),2)
```

```
       RowNumber   CustomerId  CreditScore       Age    Tenure
Balance  \
count   10000.00     10000.00     10000.00  10000.00  10000.00
10000.00
mean     5000.50  15690940.57       650.53     38.92      5.01
76485.89
std      2886.90     71936.19        96.65     10.49      2.89
62397.41
min         1.00  15565701.00       350.00     18.00      0.00
0.00
25%      2500.75  15628528.25       584.00     32.00      3.00
0.00
50%      5000.50  15690738.00       652.00     37.00      5.00
97198.54
75%      7500.25  15753233.75       718.00     44.00      7.00
127644.24
max     10000.00  15815690.00       850.00     92.00     10.00
250898.09

       NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary
Exited
count       10000.00   10000.00        10000.00         10000.00
10000.0
mean            1.53       0.71            0.52        100090.24
0.2
std             0.58       0.46            0.50         57510.49
0.4
min             1.00       0.00            0.00            11.58
0.0
25%             1.00       0.00            0.00         51002.11
```

```
0.0
50%              1.00       1.00            1.00        100193.92
0.0
75%              2.00       1.00            1.00        149388.25
0.0
max              4.00       1.00            1.00        199992.48
1.0
```

```python
df['Age'].groupby(df['CustomerId']).describe()
```

```
            count  mean  std   min   25%   50%   75%   max
CustomerId
15565701      1.0  39.0  NaN  39.0  39.0  39.0  39.0  39.0
15565706      1.0  35.0  NaN  35.0  35.0  35.0  35.0  35.0
15565714      1.0  47.0  NaN  47.0  47.0  47.0  47.0  47.0
15565779      1.0  30.0  NaN  30.0  30.0  30.0  30.0  30.0
15565796      1.0  48.0  NaN  48.0  48.0  48.0  48.0  48.0
...           ...   ...  ...   ...   ...   ...   ...   ...
15815628      1.0  37.0  NaN  37.0  37.0  37.0  37.0  37.0
15815645      1.0  37.0  NaN  37.0  37.0  37.0  37.0  37.0
15815656      1.0  39.0  NaN  39.0  39.0  39.0  39.0  39.0
15815660      1.0  34.0  NaN  34.0  34.0  34.0  34.0  34.0
15815690      1.0  40.0  NaN  40.0  40.0  40.0  40.0  40.0

[10000 rows x 8 columns]
```

```python
#handaling 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
```

```python
data_without_missing_values = df.dropna(axis=1)
```

```python
df
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender
Age  \
0             1    15634602   Hargrave          619    France  Female
```

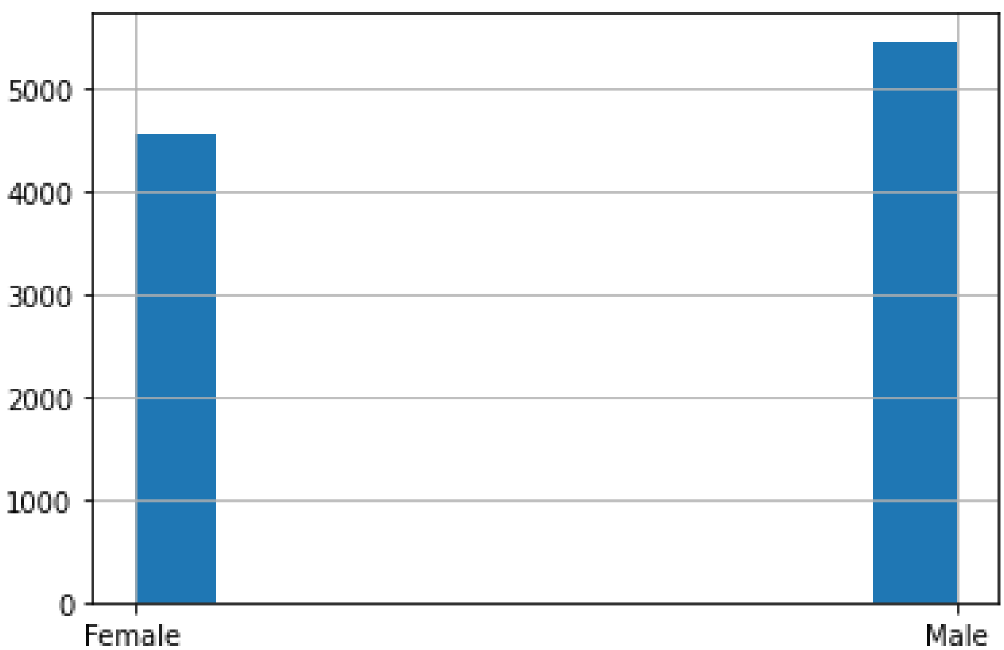
```
42
1           2   15647311      Hill         608     Spain  Female
41
2           3   15619304      Onio         502    France  Female
42
3           4   15701354      Boni         699    France  Female
39
4           5   15737888  Mitchell         850     Spain  Female
43
...       ...        ...       ...         ...       ...     ...
...
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]
```

```
print('skewness value of : ',df['Age'].skew())
print('skewness value of : ',df['RowNumber'].skew())
print('skewness value of : ',df['CustomerId'].skew())
print('skewness value of : ',df['CreditScore'].skew())
print('skewness value of : ',df['Tenure'].skew())
print('skewness value of : ',df['Balance'].skew())
print('skewness value of : ',df['NumOfProducts'].skew())
print('skewness value of : ',df['HasCrCard'].skew())
print('skewness value of : ',df['IsActiveMember'].skew())
print('skewness value of : ',df['EstimatedSalary'].skew())
print('skewness value of: ',df['Exited'].skew())
```

```
skewness value of :  1.0113202630234552
skewness value of :  0.0
skewness value of :  0.001149145900554239
skewness value of :  -0.07160660820092675
skewness value of :  0.010991457977179 04
skewness value of :  -0.14110871094154384
skewness value of :  0.7455678882823168
skewness value of :  -0.9018115952400578
skewness value of :  -0.06043662833499078
skewness value of :  0.0020853576615585162
skewness value of:  1.4716106649378211
```

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

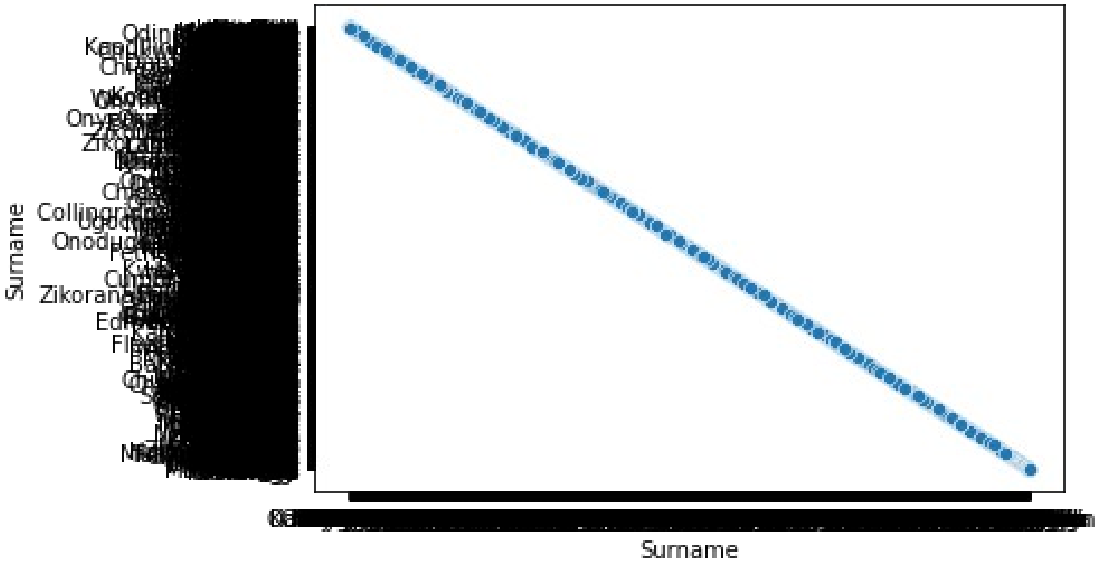
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe059d0cd90>
```




```
import seaborn as sns
sns.scatterplot(df['Surname'], df['Surname'])
```

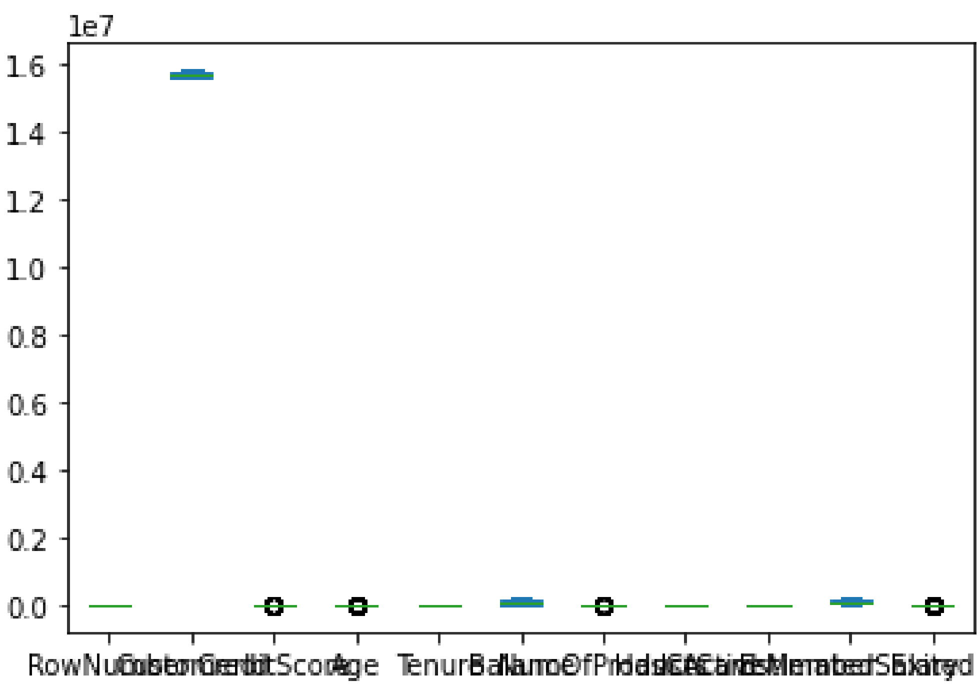
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe055b4ec90>
```




```
df.plot(kind = 'box', sharex = False, sharey = False)
```

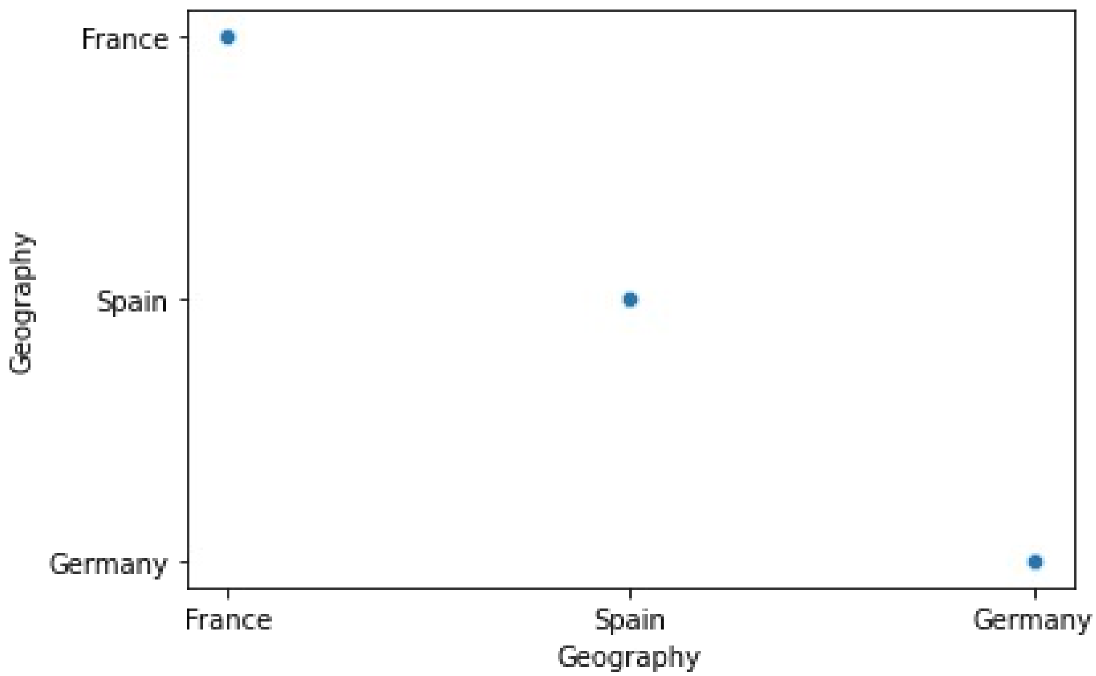
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe059e26ed0>
```



```
sns.scatterplot(df['Geography'], df['Geography'])
```

```
/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 0x7fe04fd64a10>
```




```
df.drop('Surname', axis =1)
```

```
df = df.drop('Surname', axis = 1).reset_index(drop=True)
```

```
df_categorical = df[['Gender','Geography']]
df_categorical.head()
```

```
   Gender Geography
0  Female    France
1  Female     Spain
2  Female    France
3  Female    France
4  Female     Spain
```

```
#encoding of catagorical data
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoder.fit(df_categorical['Gender'])
```

```
LabelEncoder()
values =encoder.transform(df_categorical['Gender'])
print("Before Encoding:", list(df_categorical['Gender'][-10:]))
print("After Encoding:", values[-10:])
```

```
Before Encoding: ['Male', 'Female', 'Male', 'Male', 'Female', 'Male',
'Male', 'Female', 'Male', 'Female']
After Encoding: [1 0 1 1 0 1 1 0 1 0]
```

```
residence_encoder = LabelEncoder()
residence_values =
residence_encoder.fit_transform(df_categorical['Geography'])

print("Before Encoding:", list(df_categorical['Geography'][:5]))
print("After Encoding:", residence_values[:5])
```

```
Before Encoding: ['France', 'Spain', 'France', 'France', 'Spain']
After Encoding: [0 2 0 0 2]
```

```
#spliting dep and indep data
x = df.iloc[:, 0:9].values
print(x)
```

```
[[1 15634602 619 ... 2 0.0 1]
 [2 15647311 608 ... 1 83807.86 1]
 [3 15619304 502 ... 8 159660.8 3]
 ...
 [9998 15584532 709 ... 7 0.0 1]
 [9999 15682355 772 ... 3 75075.31 2]
 [10000 15628319 792 ... 4 130142.79 1]]
```

```
y = df.iloc[:, 9:].values
print(y)
```

```
[[1.0000000e+00 1.0000000e+00 1.0134888e+05 1.0000000e+00]
 [0.0000000e+00 1.0000000e+00 1.1254258e+05 0.0000000e+00]
 [1.0000000e+00 0.0000000e+00 1.1393157e+05 1.0000000e+00]
 ...
 [0.0000000e+00 1.0000000e+00 4.2085580e+04 1.0000000e+00]
 [1.0000000e+00 0.0000000e+00 9.2888520e+04 1.0000000e+00]
 [1.0000000e+00 0.0000000e+00 3.8190780e+04 0.0000000e+00]]
```

```
#spliting the data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_st
ate=0)
xtrain.shape,xtest.shape
```

```
((7000, 9), (3000, 9))
```