1. The Dataset was successfully downloaded.

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection  import train_test_split
```

1. Load the Dataset

In [4]:
```
data = pd.read_csv("Downloads/Churn_Modelling.csv")
```

In [5]:
```
data.head()
```

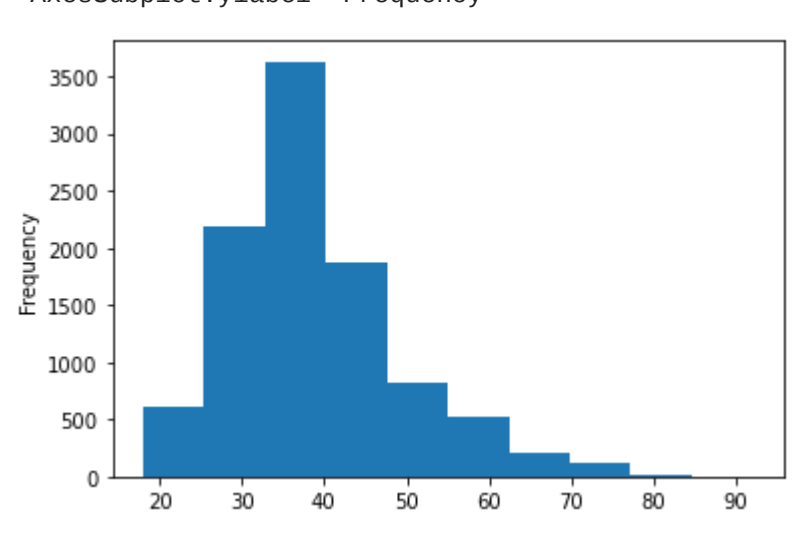
Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

1. Perform Below Visualizations 3.1 Univariate Analysis

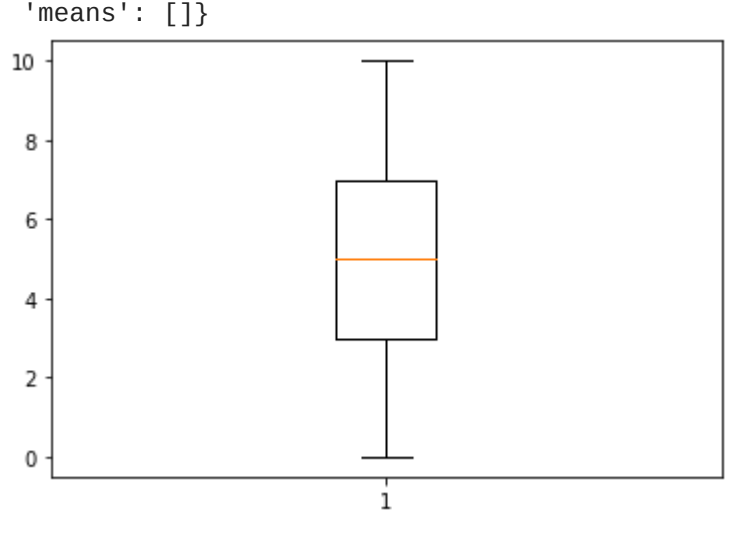
In [6]:
```
#Histogram
data['Age'].plot(kind='hist')
```

Out[6]: <AxesSubplot:ylabel='Frequency'>



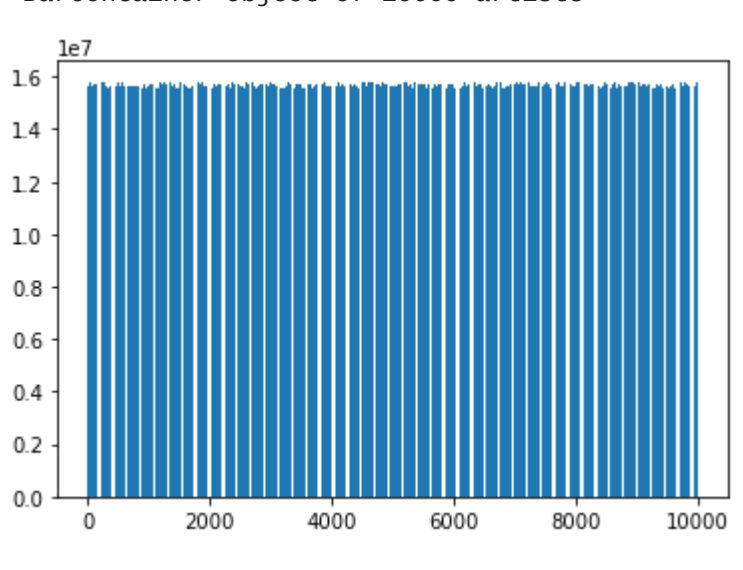
In [7]:
```
#Boxplot
plt.boxplot(data['Tenure'])
```

Out[7]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1036a040>,
  <matplotlib.lines.Line2D at 0x1036a1f0>],
 'caps': [<matplotlib.lines.Line2D at 0x1036a3a0>,
  <matplotlib.lines.Line2D at 0x1036a550>],
 'boxes': [<matplotlib.lines.Line2D at 0x10358e68>],
 'medians': [<matplotlib.lines.Line2D at 0x1036a700>],
 'fliers': [<matplotlib.lines.Line2D at 0x1036a8b0>],
 'means': []}
```



In [9]:
```
#Bar chart
df = pd.DataFrame(data)
X = list(df.iloc[:,0])
Y = list(df.iloc[:,1])
plt.bar(X,Y)
```

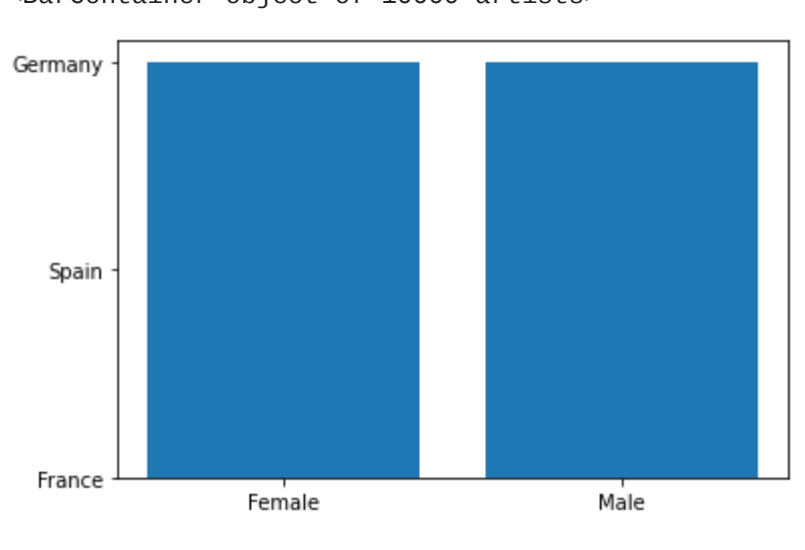
Out[9]: <BarContainer object of 10000 artists>



3.2 Bivariate Analysis

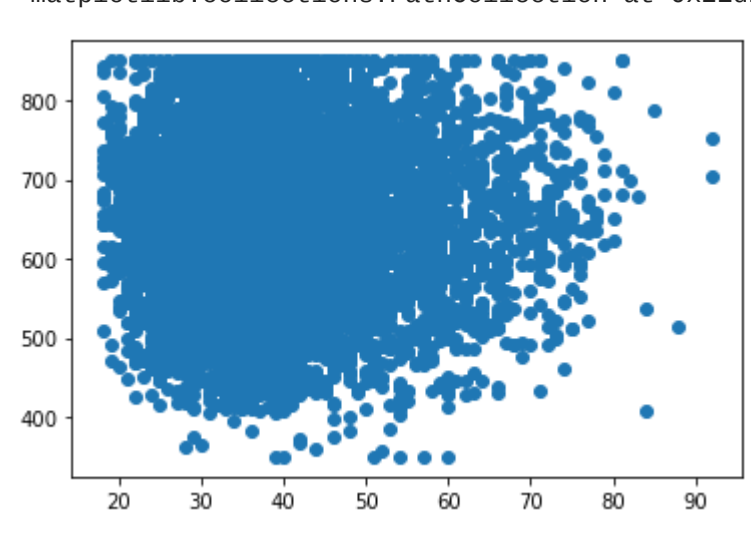
In [13]:
```
#Stacked Bar Chart
plt.bar(data['Gender'],data['Geography'])
```

Out[13]: <BarContainer object of 10000 artists>



In [12]:
```
#Scatter plot
plt.scatter(data['Age'],data['CreditScore'])
```

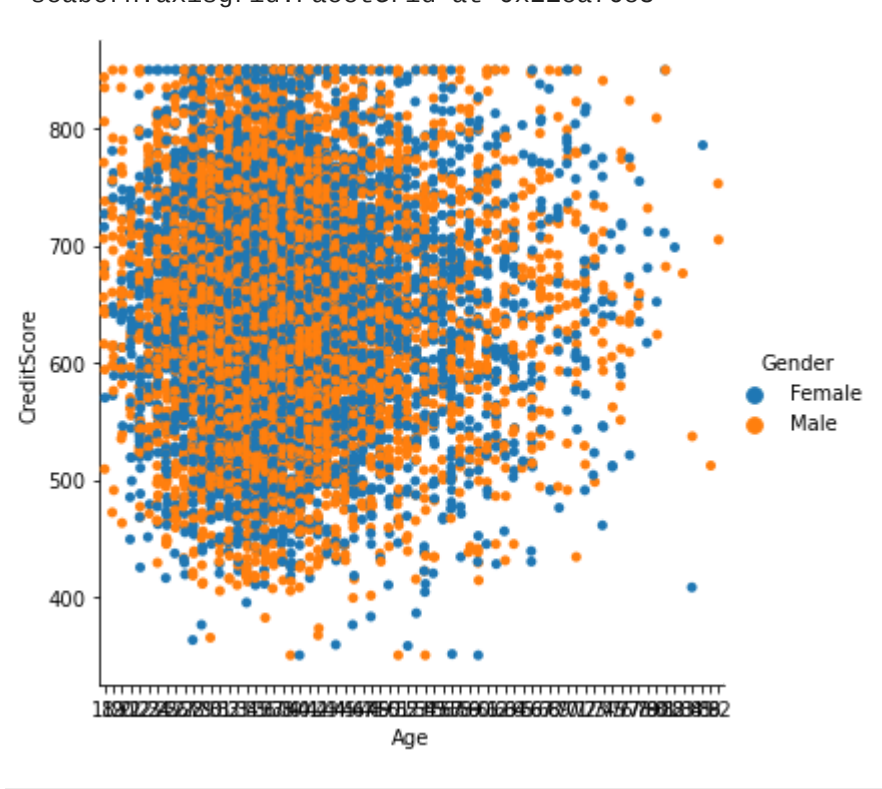
Out[12]: <matplotlib.collections.PathCollection at 0x21d13a90>



3.3 Multivariate Analysis

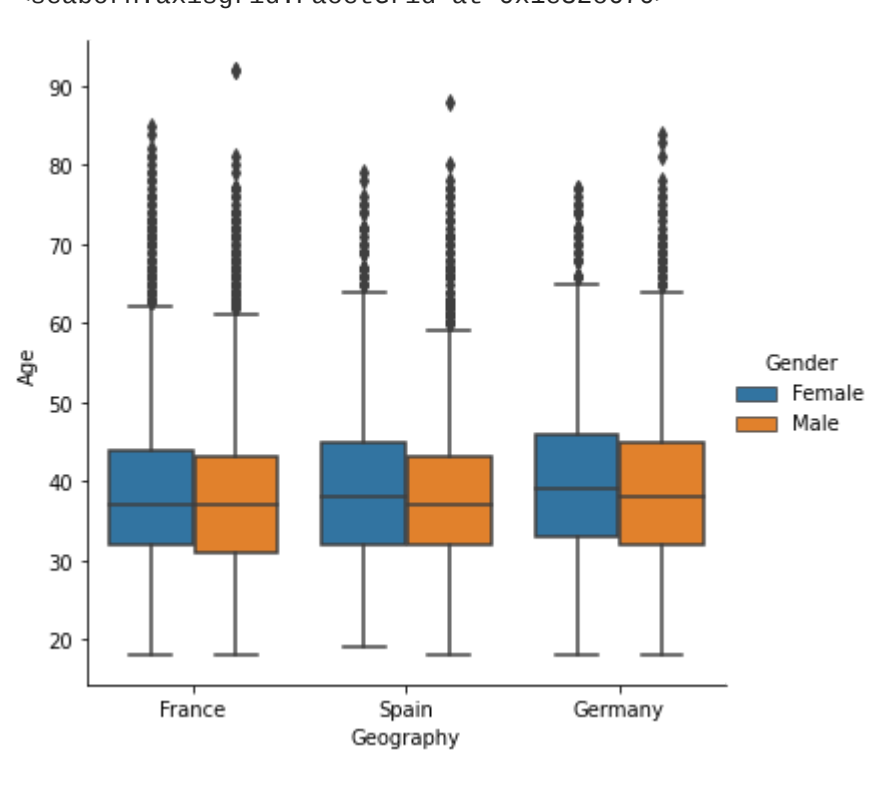
In [14]:
```
sns.catplot(data=data,x='Age',y='CreditScore',hue='Gender')
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x21ca76e8>



In [15]:
```
#Box Plot
sns.catplot(data=data,x='Geography',y='Age',hue='Gender',kind='box')
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x1e32c670>



1. Perform Descriptive Statistics on the dataset

In [16]:
```
data.mean()
```

Out[16]:
```
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
```

In [17]:
```
data.median()
```

Out[17]:
```
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
```

In [18]:
```
data.describe()
```

Out[18]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

In [19]:
```
data.shape
```

Out[19]: (10000, 14)

1. Handle the missing values

In [20]:
```
data.isnull().sum()
```

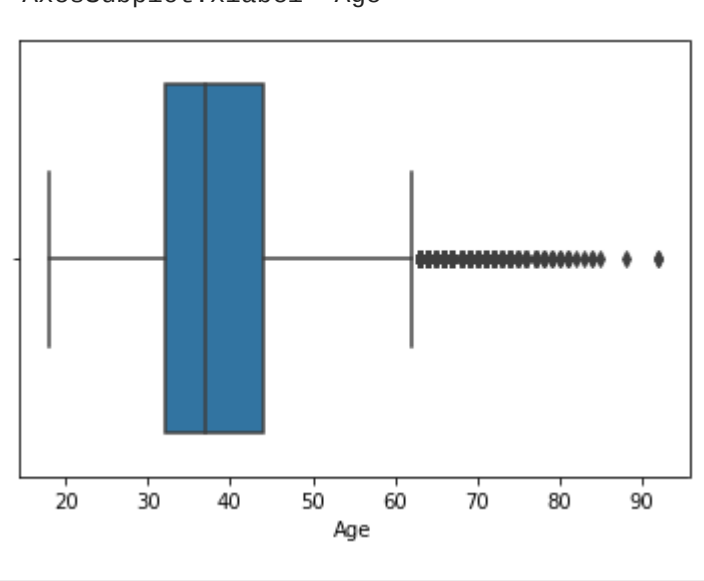
Out[20]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

1. Find the Outliners and replace the Outliners

In [21]:
```
sns.boxplot(data['Age'])
```

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

Out[21]: <AxesSubplot:xlabel='Age'>



In [22]:
```
qnt = data.quantile(q=[0.25,0.75])
qnt
```

Out[22]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0

In [23]:
```
IQR = qnt.loc[0.75] - qnt.loc[0.25]
IQR
```

Out[23]:
```
RowNumber            4999.5000
CustomerId         124705.5000
CreditScore           134.0000
Age                    12.0000
Tenure                  4.0000
Balance            127644.2400
NumOfProducts           1.0000
HasCrCard               1.0000
IsActiveMember          1.0000
EstimatedSalary     98386.1375
Exited                  0.0000
dtype: float64
```

In [24]:
```
upper_extreme = qnt.loc[0.75]+1.5*IQR
upper_extreme
```

Out[24]:
```
RowNumber          1.499950e+04
CustomerId         1.594029e+07
CreditScore        9.190000e+02
Age                6.200000e+01
Tenure             1.300000e+01
Balance            3.191106e+05
NumOfProducts      3.500000e+00
HasCrCard          2.500000e+00
IsActiveMember     2.500000e+00
EstimatedSalary    2.969675e+05
Exited             0.000000e+00
dtype: float64
```

In [25]:
```
lower_extreme = qnt.loc[0.25]-1.5*IQR
lower_extreme
```

Out[25]:
```
RowNumber         -4.998500e+03
CustomerId         1.544147e+07
CreditScore        3.830000e+02
Age                1.400000e+01
Tenure            -3.000000e+00
Balance           -1.914664e+05
NumOfProducts     -5.000000e-01
HasCrCard         -1.500000e+00
IsActiveMember    -1.500000e+00
EstimatedSalary   -9.657710e+04
Exited             0.000000e+00
dtype: float64
```

In [26]:
```
df2 = data[(data['Age']<upper_extreme['Age']) & (data['Age']>lower_extreme['Age'])]
df.shape
```

Out[26]: (10000, 14)

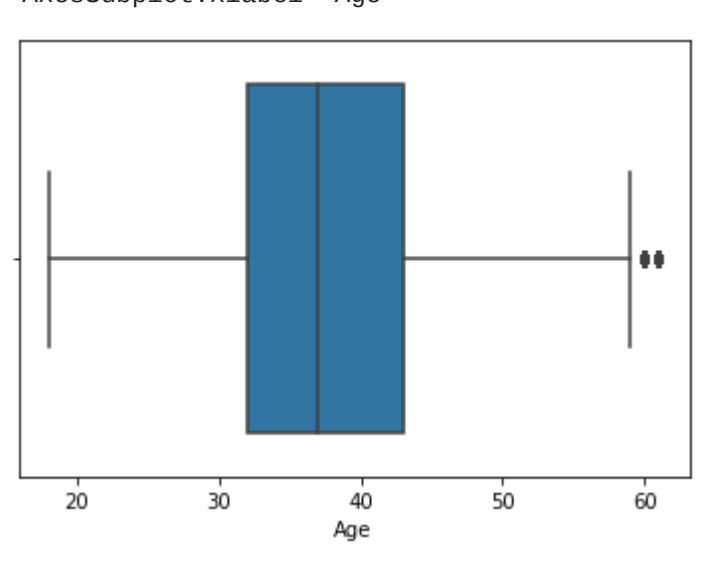
In [27]:
```
df2.shape
```

Out[27]: (9589, 14)

In [28]:
```
sns.boxplot(df2['Age'])
```

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

Out[28]: <AxesSubplot:xlabel='Age'>



1. Check the Categorial columns and perform Encoding

In [29]:
```
le = LabelEncoder()
df2['Geography'] = le.fit_transform(df2['Geography'])
df2['Gender'] = le.fit_transform(df2['Gender'])
df2.head()
```

```
<ipython-input-29-7fbc4f02a1ee>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Geography'] = le.fit_transform(df2['Geography'])
<ipython-input-29-7fbc4f02a1ee>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Gender'] = le.fit_transform(df2['Gender'])
```

Out[29]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0

1. Split the data into dependent and independent variables

In [30]:
```
x = df.iloc[:,:-1].values
x
```

Out[30]:
```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

In [31]:
```
y = df.iloc[:,-1].values
y
```

Out[31]: array([1, 0, 1, ..., 1, 1, 0], dtype=int64)

1. Scale the independent variables

In [32]:
```
scaler = MinMaxScaler()
df[['CustomerId']] = scaler.fit_transform(df[['CustomerId']])
df
```

Out[32]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	0.275616	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	0.326454	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	0.214421	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	0.542636	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	0.688778	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	0.162119	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	0.016765	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	0.075327	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	0.466637	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	0.250483	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

1. Split the data into training and testing

In [33]:
```
train_size=0.8
X = df.drop(columns=['Tenure']).copy()
Y = df['Tenure']
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=0.8)
print(X_train)
print(Y_train)
print(X_test)
print(Y_test)
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender  Age  \
696         697    0.646420      Edgar          445     Spain    Male   62
9445       9446    0.665029     Reilly          689     Spain  Female   28
4344       4345    0.078907    Madueke          528    France  Female   26
5022       5023    0.581742      Tseng          734    France    Male   45
8724       8725    0.111933   Solomina          668     Spain  Female   30
...         ...         ...        ...          ...       ...     ...  ...
2591       2592    0.900620      Ts'ai          634   Germany    Male   38
9841       9842    0.682534      Manna          567   Germany  Female   46
7735       7736    0.836977      Liang          543   Germany    Male   41
6160       6161    0.561733    Zhdanov          632    France  Female   21
6726       6727    0.270060      Rizzo          628     Spain    Male   50

        Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  \
696    64119.38              1          1               1         76569.64
9445   95328.60              1          1               0          6129.61
4344  102073.67              2          0               0        166799.93
5022       0.00              2          1               0         99593.28
8724       0.00              2          1               0        138465.70
...         ...            ...        ...             ...              ...
2591  148430.55              1          1               1         56055.72
9841   68238.51              2          1               1        109572.58
7735  143350.41              1          1               1        192070.16
6160       0.00              2          1               0         84008.66
6726       0.00              1          0               0        144366.83

      Exited
696        1
9445       1
4344       0
5022       0
8724       0
...      ...
2591       0
9841       0
7735       1
6160       0
6726       1

[8000 rows x 13 columns]
696      1
9445     5
4344    10
5022     2
8724     8
        ..
2591     2
9841     1
7735     6
6160     1
6726     8
Name: Tenure, Length: 8000, dtype: int64
      RowNumber  CustomerId     Surname  CreditScore Geography  Gender  Age  \
5320       5321    0.103401     Bianchi          525     Spain    Male   36
9813       9814    0.501898   Vasilyeva          590     Spain  Female   29
6752       6753    0.158899   Oluchukwu          624     Spain    Male   38
128         129    0.039166     Kennedy          828    France  Female   41
4050       4051    0.080136      Gouger          803   Germany    Male   41
...         ...         ...         ...          ...       ...     ...  ...
5101       5102    0.623071      Shubin          622     Spain  Female   58
2981       2982    0.582094       Parry          616   Germany    Male   39
8870       8871    0.895475       Clark          668    France    Male   42
1756       1757    0.420022       Smith          618    France    Male   37
4214       4215    0.308050  Napolitani          686    France    Male   26

        Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  \
5320   77910.23              1          1               0         67238.01
9813   99250.08              1          1               1        129629.41
6752  123906.55              1          1               0        135096.78
128        0.00              2          1               0        171378.77
4050  137742.90              2          1               1        166957.82
...         ...            ...        ...             ...              ...
5101       0.00              2          1               1         33277.31
2981  121704.32              2          1               0         55556.30
8870  187534.79              1          1               1         32900.41
1756  168178.21              2          0               1        101273.23
4214       0.00              2          1               0          1540.89

      Exited
5320       0
9813       0
6752       0
128        0
4050       0
...      ...
5101       0
2981       0
8870       1
1756       0
4214       0

[2000 rows x 13 columns]
5320     3
9813    10
6752     7
128      7
4050     9
        ..
5101     2
2981     2
8870     8
1756     2
4214     7
Name: Tenure, Length: 2000, dtype: int64
```

In [ ]: