```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv('/Churn_Modelling.csv')
df
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75031	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

1)Univariate Analysis

In [3]:
```
plt.scatter(df.index,df['Age'])
```

Out[3]:
```
<matplotlib.collections.PathCollection at 0x7fcded5146d0>
```

In [4]:
```
df['Age'].value_counts().head(12).plot.bar()
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcdecfa96d0>
```

In [5]:
```
plt.hist(df['Age'])
```

Out[5]:
```
(array([ 611., 2179., 3629., 1871.,  828.,  523.,  208.,  127.,   20.,
           4.]),
 array([18. , 25.4, 32.8, 40.2, 47.6, 55. , 62.4, 69.8, 77.2, 84.6, 92. ]),
 <a list of 10 Patch objects>)
```

2)Bi-variate Analysis

In [6]:
```
sns.scatterplot(x='CreditScore',y='Balance',data=df,hue='Gender')
plt.show()
```

In [7]:
```
df.plot.line()
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcdecd93350>
```

3)Multi-Variate Analysis

In [8]:
```
sns.pairplot(df)
plt.show()
```

4)Descriptive Statistics

In [9]:

```
df.describe()
```

Out[9]:

	RowNumber	CustomerId	Credit Score	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 [11]:

```
df.sum()
```

Out[11]:

```
RowNumber                                                50005000
CustomerId                                          156909405694
Surname          HargraveHillOnioBoniMitchellChuBartlettObinnaH...
CreditScore                                              6505288
Geography        FranceSpainFranceFranceSpainSpainFranceGermany...
Gender           FemaleFemaleFemaleFemaleFemaleMaleMaleFemaleMa...
```

```
Age                                                    389218
Tenure                                                  50128
Balance                                            764858892.88
NumOfProducts                                          15302
HasCrCard                                               7055
IsActiveMember                                          5151
EstimatedSalary                                   1000902398.81
Exited                                                  2037
dtype: object
```

In [12]:

```
df.min()
```

Out[12]:

```
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
```

In [13]:

```
df.max()
```

Out[13]:

```
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
```

5)Handling Missing the Values

In [14]:

```
df.isnull()
```

Out[14]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False	False	False	False	False	False	False

10000 rows × 14 columns

In [15]:

```
df.dropna()
```

Out[15]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9999	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
98														
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

6)Find the outliers and replace the outliers

In [16]:

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation
.
  FutureWarning
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcde5a6cf50>
```


In [17]:

```
print(np.where(df['Age']>48))
```

```
(array([   6,   16,   41, ..., 9975, 9979, 9991]),)
```

In [18]:

```
missing_values=df.isnull().sum()
missing_values[missing_values>0]/len(df)*100
```

Out[18]:

```
Series([], dtype: float64)
```

In [19]:

```
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 1.5
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
df['Age']=np.where(df['Age']>upper_whisker,upper_whisker,np.where(df['Age']
<lower_whisker,lower_whisker,df['Age']))
```

In [20]:

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation
.
  FutureWarning
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcde59c8410>
```

7)Check for Categorical columns and perform encoding

In [21]:

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

Out[21]:

	Geography	Gender	CustomerId	Surname
0	France	Female	15634602	Hargrave
1	Spain	Female	15647311	Hill
2	France	Female	15619304	Onio
3	France	Female	15701354	Boni
4	Spain	Female	15737888	Mitchell

In [22]:

```
from sklearn.preprocessing import LabelEncoder

Gender_encoder = LabelEncoder()
Gender_encoder.fit(df_categorical['Gender'])
```

Out[22]:

```
LabelEncoder()
```

In [23]:

```
gender_values = Gender_encoder.transform(df_categorical['Gender'])
print("Before Encoding:", list(df_categorical['Gender'][-10:]))
print("After Encoding:", gender_values[-10:])
print("The inverse from the encoding result:",
Gender_encoder.inverse_transform(gender_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]
The inverse from the encoding result: ['Male' 'Female' 'Male' 'Male' 'Femal
e' 'Male' 'Male' 'Female' 'Male'
 'Female']
```

8)Split the data into dependent and independent variables

In [24]:

```
#independent
X = df.iloc[:, :-1].values
print(X)
```

```
[[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]]
```

In [25]:

```
#dependent
Y = df.iloc[:, -1].values
print(Y)
```

```
[1 0 1 ... 1 1 0]
```

9)Scale the independent variables

In [26]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[["CustomerId"]] = scaler.fit_transform(df[["CustomerId"]])
print(df)
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender   Age
\
0             1    0.275616   Hargrave          619    France  Female  42.0
1             2    0.326454       Hill          608     Spain  Female  41.0
2             3    0.214421       Onio          502    France  Female  42.0
3             4    0.542636       Boni          699    France  Female  39.0
4             5    0.688778   Mitchell          850     Spain  Female  43.0
...         ...         ...        ...          ...       ...     ...   ...
9995       9996    0.162119   Obijiaku          771    France    Male  39.0
9996       9997    0.016765  Johnstone          516    France    Male  35.0
9997       9998    0.075327        Liu          709    France  Female  36.0
9998       9999    0.466637  Sabbatini          772   Germany    Male  42.0
9999      10000    0.250483     Walker          792    France  Female  28.0

      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]
```

10)Split the data into training and testing

In [27]:

```
from sklearn.model_selection import train_test_split
training_data, testing_data = train_test_split(df, test_size=0.2,
random_state=25)
print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

```
No. of training examples: 8000
No. of testing examples: 2000
```