

ASSESSMENT 2

ASSESSMENT DATE	26-09-2022
STUDENT NAME	Sakthi Priya. S
STUDENT ROLL NUMBER	713119104016
MAXIMUM MARKS	2 Marks

1. Download the dataset
2. Load the dataset Solution:

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

```
df=pd.read_csv("C:\\Users\\PC\\Desktop\\Churn_Modelling.csv")
df
```

output:

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10
...
9995	9996	15606229	Obijaku	771	France	Male	39	5	0.00	2	1	0	93826.63
9996	9997	1559692	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101348.88
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	112542.58
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	93826.63
9999	10000	15606160	Widmer	700	France	Female	39	4	125510.82	1	1	1	79084.10

df.head()

Out[8]:

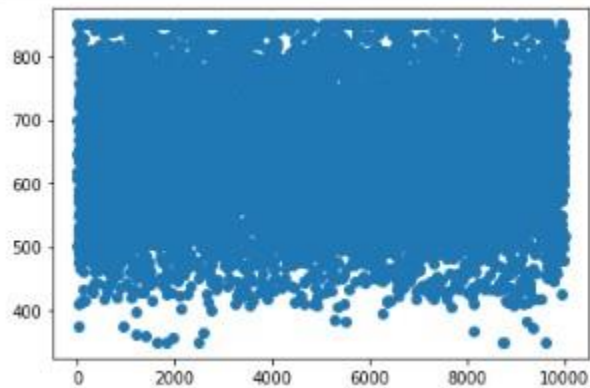
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42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

3. perform following operations

➤ **univariate analysis**

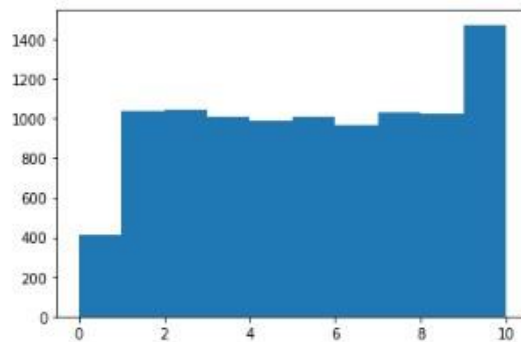
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.scatter(df.index,df['CreditScore'])
plt.show()
```

output:



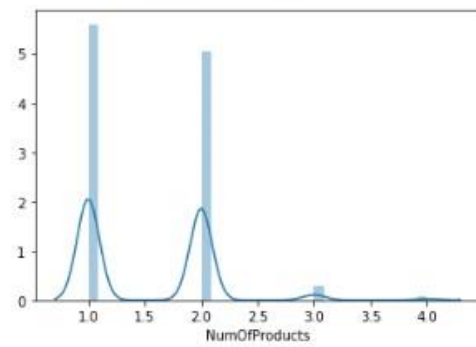
```
plt.hist(df['Tenure'])
```

```
Out[12]: (array([ 413., 1035., 1048., 1009.,  989., 1012.,  967., 1028., 1025.,
        1474.]),
 array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.]),
 <a list of 10 Patch objects>)
```



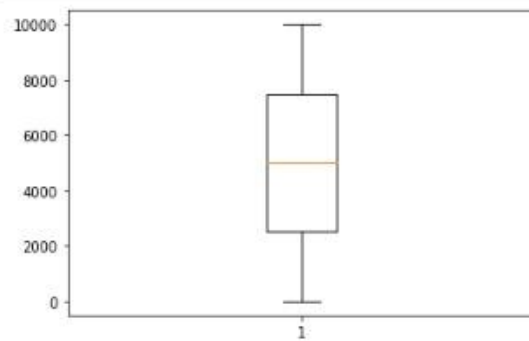
```
sns.distplot(df['NumOfProducts'])
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x25dff3899c8>
```

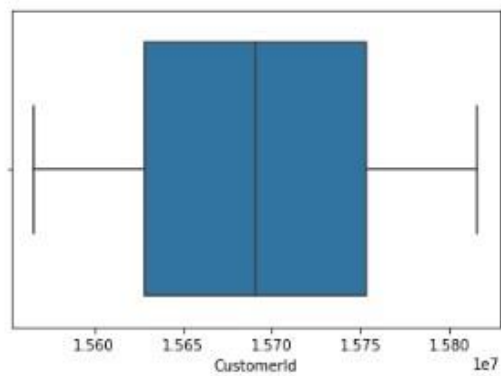


```
plt.boxplot(df['RowNumber'])
```

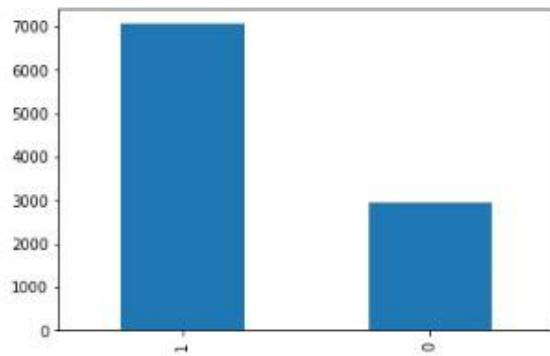
```
plt.show()
```



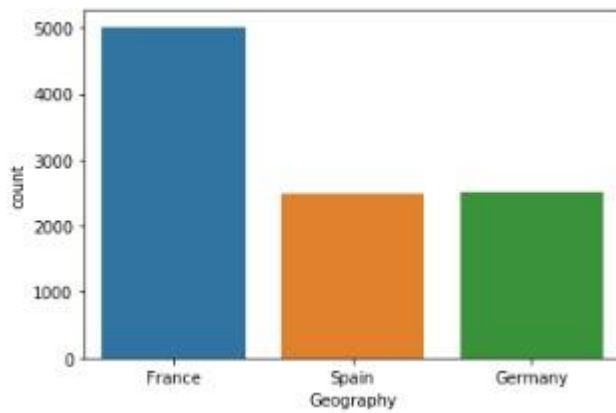
```
sns.boxplot(df['CustomerId'])
```



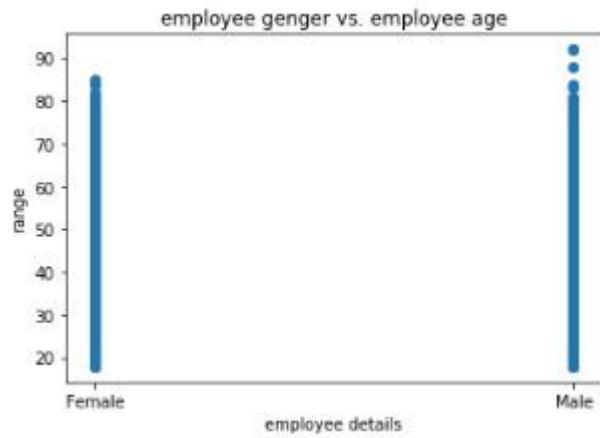
```
df['HasCrCard'].value_counts().plot.bar()
```



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



Bivariate analysis plt.scatter(df.Gender,
df.Age) plt.title('employee genger vs.
employee age') plt.xlabel('employee
details') plt.ylabel('range')



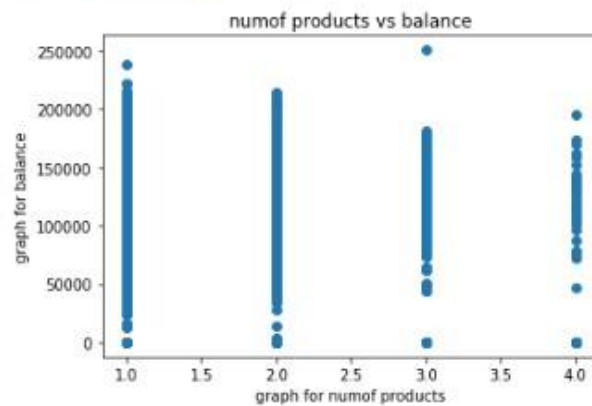
```
plt.scatter(df.NumOfProducts, df.Balance)
```

```
plt.title('numof products vs balance')
```

```
plt.xlabel('graph for numof products')
```

```
plt.ylabel('graph for balance')
```

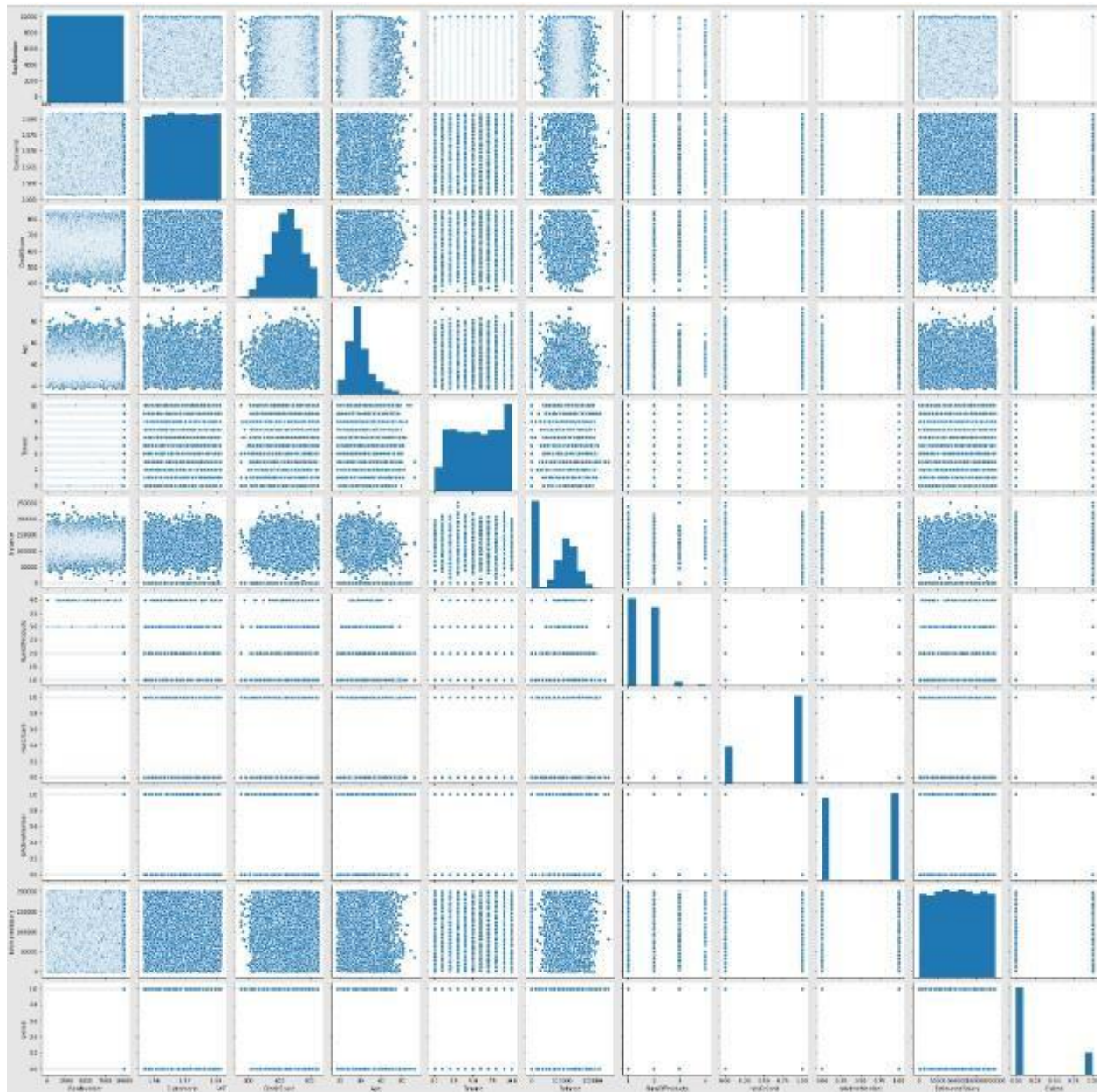
```
Out[20]: Text(0, 0.5, 'graph for balance')
```



Multivariate analysis

```
seaborn.pairplot(df)
```

```
plt.show()
```



4.descriptive function

df.describe()

Out[6]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	78485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.566570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

5.handle the missing data 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
```

df.isnull()

Out[14]:

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

10000 rows x 14 columns

df.notnull()

Out[15]:

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

10000 rows x 14 columns

df.fillna(0)

Out[16]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634802	Hargrave	619	France	Female	42	2	0.00	1	1	1	10134
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	11254
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	11393
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	9382
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	7908
...
9995	9996	15606229	Obijaku	771	France	Male	39	5	0.00	2	1	0	9627
9996	9997	15596892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	3819

10000 rows x 14 columns

df["Gender"].fillna("No Gender", inplace = True) df

Out[21]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634802	Hargrave	619	France	Female	42	2	0.00	1	1	1	10134
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	11254
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	11393
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	9382
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	7908
...
9995	9996	15606229	Obijaku	771	France	Male	39	5	0.00	2	1	0	9627
9996	9997	15596892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	3819

10000 rows x 14 columns

df.drop("RowNumber",axis=1,inplace=True)

df

Out[28]:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	15619304	Onio	502	France	Female	42	8	150860.80	3	1	0	113931.57
3	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10
...
9995	15606229	Obijaku	771	France	Male	39	5	0.00	2	1	0	96270.64
9996	15509892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77
9997	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52
9999	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78

10000 rows x 12 columns

```
print(df.isnull().sum())
```

```
CustomerId      0
Surname          0
CreditScore     0
Geography       0
Gender          0
Age             0
Tenure          0
Balance         0
NumOfProducts  0
HasCrCard       0
IsActiveMember  0
EstimatedSalary 0
dtype: int64
```

```
updated_df = df.dropna(axis=1) updated_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
#   Column             Non-Null Count  Dtype
---  -
0   CustomerId         10000 non-null  int64
1   Surname            10000 non-null  object
2   CreditScore        10000 non-null  int64
3   Geography          10000 non-null  object
4   Gender             10000 non-null  object
5   Age               10000 non-null  int64
6   Tenure            10000 non-null  int64
7   Balance           10000 non-null  float64
8   NumOfProducts     10000 non-null  int64
9   HasCrCard         10000 non-null  int64
10  IsActiveMember    10000 non-null  int64
11  EstimatedSalary    10000 non-null  float64
dtypes: float64(2), int64(7), object(3)
memory usage: 937.6+ KB
```

6.Finding outliers and replace

```
Q1 = df.quantile(0.25)
```

```
Q3 = df.quantile(0.75) IQR
```

```
= Q3 - Q1
```

```
print(IQR)
```

```
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
```

```
print(df < (Q1 - 1.5 * IQR))
```

```
(df > (Q3 + 1.5 * IQR))
```

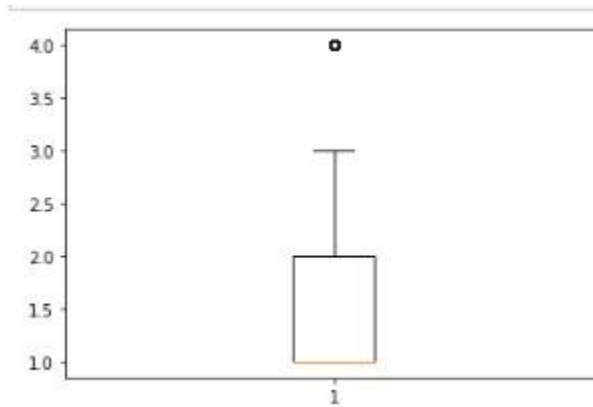
```
   Age  Balance  CreditScore  CustomerId  EstimatedSalary  Exited  \
0  False  False      False      False      False      False
1  False  False      False      False      False      False
2  False  False      False      False      False      False
3  False  False      False      False      False      False
4  False  False      False      False      False      False
...   ...     ...         ...         ...         ...         ...
9995  False  False      False      False      False      False
9996  False  False      False      False      False      False
9997  False  False      False      False      False      False
9998  False  False      False      False      False      False
9999  False  False      False      False      False      False

   Gender  Geography  HasCrCard  IsActiveMember  NumOfProducts  RowNumber  \
0  False      False      False      False      False      False
1  False      False      False      False      False      False
2  False      False      False      False      False      False
3  False      False      False      False      False      False
4  False      False      False      False      False      False
...   ...     ...         ...         ...         ...         ...
9995  False      False      False      False      False      False
9996  False      False      False      False      False      False
9997  False      False      False      False      False      False
9998  False      False      False      False      False      False
9999  False      False      False      False      False      False

   Surname  Tenure
0  False  False
1  False  False
2  False  False
3  False  False
4  False  False
...   ...     ...
9995  False  False
9996  False  False
9997  False  False
9998  False  False
9999  False  False
```

```
plt.boxplot(df["NumOfProducts"])
```

```
plt.show()
```



```
np.where(df.Age>42,42, df.Age)
```

```
Out[16]: array([42, 41, 42, ..., 36, 42, 28], dtype=int64)
```

```
print(df['Age'].skew())
```

```
1.0113202630234552
```

```
print(df['Age'].quantile(0.25))
```

```
print(df['Age'].quantile(0.75))
```

```
df['Age'] = np.where(df['Age'] > 39, 41, df['Age'])
```

```
df.describe()
```

```
32.0
41.0
```

```
Out[22]:
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	35.788800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193819e+04	96.653299	5.659409	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.000000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.000000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.000000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	41.000000	7.000000	127644.240000	2.000000	1.000000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	41.000000	10.000000	250898.090000	4.000000	1.000000	1.000000	199992.480000

7.categorical column

```
df["CustomerId"].value_counts()
```

```
Out[23]: 15812607    1
         15741078    1
         15635776    1
         15740223    1
         15738174    1
         ..
         15743714    1
         15639265    1
         15641312    1
         15684319    1
         15695872    1
         Name: CustomerId, Length: 10000, dtype: int64
```

df.dtypes

```
Out[27]: RowNumber      int64
         CustomerId    int64
         Surname        category
         CreditScore    int64
         Geography      object
         Gender          object
         Age            int64
         Tenure          int64
         Balance         float64
         NumOfProducts  int64
         HasCrCard       int64
         IsActiveMember int64
         EstimatedSalary float64
         Exited          int64
         dtype: object
```

df["Age"].value_counts().sort_index()

```
Out[32]: 18    22
         19    27
         20    40
         21    53
         22    84
         23    99
         24   132
         25   154
         26   200
         27   209
         28   273
         29   348
         30   327
         31   404
         32   418
         33   442
         34   447
         35   474
         36   456
         37   478
         38   477
         39   423
         41  4013
         Name: Age, dtype: int64
```

df_categorical = df[categorical_columns]

df_categorical.head()

```
Out[35]:
```

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

```
pd.get_dummies(df, columns=["Age"]).head()
```

```
Out[36]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Tenure	Balance	NumOfProducts	HasCrCard	...	Age_31	Age_32	Age_33	Age_34
0	1	15634602	Hargrave	819	France	Female	2	0.00	1	1	...	0	0	0	0
1	2	15647311	Hill	608	Spain	Female	1	83807.88	1	0	...	0	0	0	0
2	3	15619304	Onio	502	France	Female	8	150000.80	3	1	...	0	0	0	0
3	4	15701354	Boni	699	France	Female	1	0.00	2	0	...	0	0	0	0
4	5	15737888	Mitchell	850	Spain	Female	2	125510.82	1	1	...	0	0	0	0

5 rows x 36 columns

8.split the data into dependent and independent variables print(df.size)

```
140000
```

```
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]]
```

```
Y = df.iloc[:, -1].values
```

```
print(Y)
```

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

9.minmaxscaler

```
from sklearn.preprocessing import MinMaxScaler
```

```
df
```

```
scaler = MinMaxScaler()
```

```
print(scaler.fit(df))
```

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

```
10.train -split data import pandas as pd from
sklearn.linear_model import LinearRegression from
sklearn.model_selection import train_test_split
df=pd.read_csv("C:\\Users\\PC\\Desktop\\Churn_Modelling.csv")
df.head()
```

```
Out[77]:
```

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

```
y= df.Tenure
```

```
y.head()
```

```
Out[78]: 0    2
         1    1
         2    8
         3    1
         4    2
         Name: Tenure, dtype: int64
```

```
x=df.drop('Tenure', axis=1)
```

```
x.head()
```

```
Out[80]:
```

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```


x_train.shape

```
Out[82]: (8000, 13)
```

y_train.shape

```
Out[83]: (8000,)
```

x_test.shape

```
Out[84]: (2000, 13)
```

y_test.shape

```
Out[85]: (2000,)
```