Assignment -2

Python Programming

Assignment Date	17.09.2022
Student Name	PREETHI R
Student Roll Number	211419104202
Maximum Marks	2 Marks

1. Download the dataset: Dataset

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
import warnings
warnings.filterwarnings('ignore')
```

2. Load the dataset.

In [6]:
```
data= pd.read_csv('E:churn_modelling.csv')
data.head()
```

Out[6]:

	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

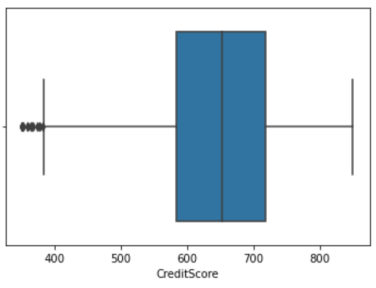
In [ ]:

3. Perform Below Visualizations

Univariate Analysis

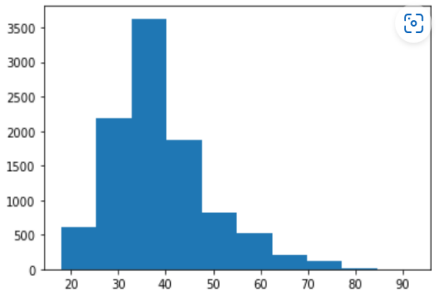
In [8]:
```
# Boxplot
sns.boxplot(data['CreditScore'])
```

Out[8]: <AxesSubplot:xlabel='CreditScore'>



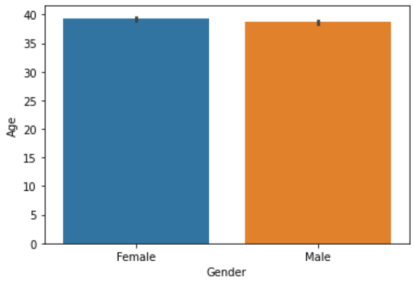
In [9]:
```
plt.hist(data['Age'])
```

Out[9]:
```
(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. ]),
 <BarContainer object of 10 artists>)
```



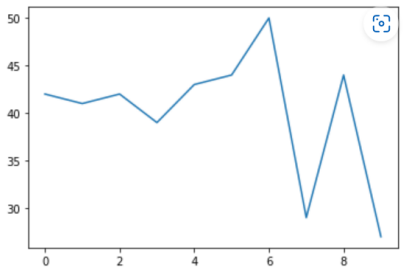
In [10]:
```
sns.barplot(data['Gender'], data['Age'])
```

Out[10]:
```
<AxesSubplot:xlabel='Gender', ylabel='Age'>
```



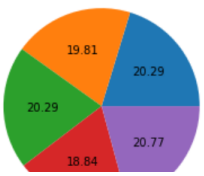
In [11]:
```
plt.plot(data['Age'].head(10))
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x261954e0dc0>]
```



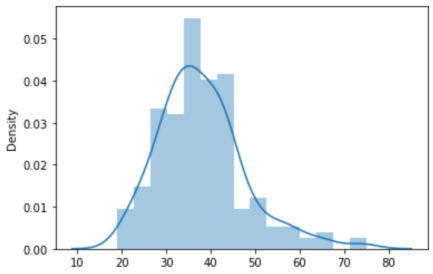
In [12]:
```
plt.pie(data['Age'].head(), autopct="%.2f")
```

Out[12]:
```
([<matplotlib.patches.Wedge at 0x261955451f0>,
  <matplotlib.patches.Wedge at 0x26195545940>,
  <matplotlib.patches.Wedge at 0x261955530d0>,
  <matplotlib.patches.Wedge at 0x261955537f0>,
  <matplotlib.patches.Wedge at 0x26195553f10>],
 [Text(0.8839942345509236, 0.654640506904917, ''),
  Text(-0.3525952068146547, 1.0419580702366729, ''),
  Text(-1.09987331875942, -0.01669379169450419, ''),
  Text(-0.35259525559223215, -1.0419580537304987, ''),
  Text(0.8739574598774371, -0.6679808068534441, '')],
 [Text(0.48217867339141285, 0.3570766401299547, '20.29'),
  Text(-0.19232465826253894, 0.5683407655836397, '19.81'),
  Text(-0.5999309011415017, -0.009105704560638648, '20.29'),
  Text(-0.19232468486849025, -0.5683407565802719, '18.84'),
  Text(0.4767040690240565 4, -0.3643531673746058, '20.77')])
```



In [13]:
```
sns.distplot(data['Age'].head(200))
```

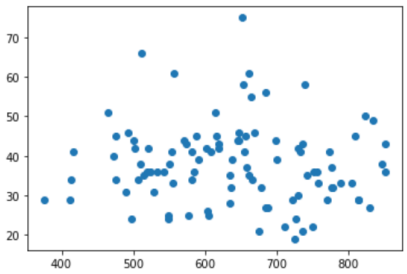
Out[13]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



BI - Variate Analysis

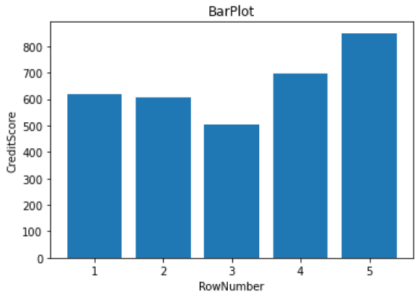
In [14]:
```
plt.scatter(data['CreditScore'].head(100),data['Age'].head(100))
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0x26195a30550>
```



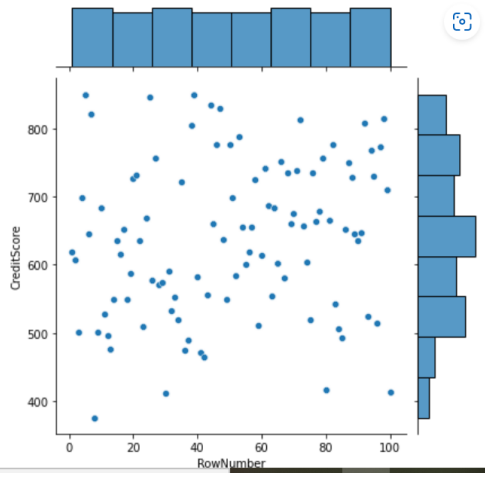
In [15]:
```
plt.bar(data['RowNumber'].head() ,data['CreditScore'].head(),  )

plt.title('BarPlot')
plt.xlabel('RowNumber')
plt.ylabel('CreditScore')
```

Out[15]:
```
Text(0, 0.5, 'CreditScore')
```



In [16]:
```
sns.jointplot(data['RowNumber'].head(100) ,data['CreditScore'].head(100),  )
```

Out[16]:
```
<seaborn.axisgrid.JointGrid at 0x26196a7b760>
```



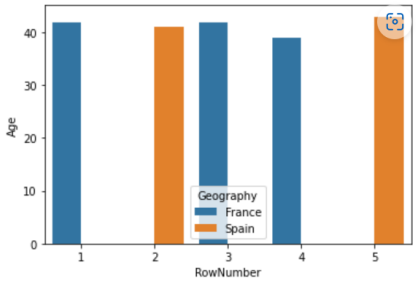
In [17]:
```
data.head()
```

Out[17]:

	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

In [18]:
```
sns.barplot('RowNumber','Age',hue='Geography', data=data.head())
```

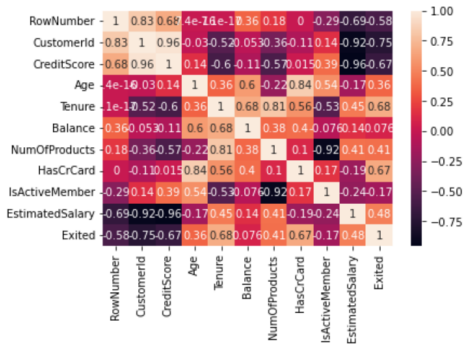
Out[18]:
```
<AxesSubplot:xlabel='RowNumber', ylabel='Age'>
```



Multi - Variate Analysis

In [19]:
```
sns.heatmap(data.head().corr(), annot = True)
```

Out[19]:
```
<AxesSubplot:>
```



4. Perform descriptive statistics on the dataset.

In [20]:
```
data.mean()
```

Out[20]:
```
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 [21]:
```
data.median()
```

Out[21]:
```
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 [22]:
```
data.mode()
```

Out[22]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSala
0	1	15565701	Smith	850.0	France	Male	37.0	2.0	0.0	1.0	1.0	1.0	24924.9
1	2	15565706	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	3	15565714	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	4	15565779	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	5	15565796	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15815628	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9996	9997	15815645	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9997	9998	15815656	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9998	9999	15815660	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9999	10000	15815690	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

10000 rows × 14 columns

In [23]:
```
data.std()
```

Out[23]:
```
RowNumber          2886.895680
CustomerId        71936.186123
CreditScore          96.653299
Age                  10.487806
Tenure                2.892174
Balance           62397.405202
NumOfProducts         0.581654
HasCrCard             0.455840
IsActiveMember        0.499797
EstimatedSalary   57510.492818
Exited                0.402769
dtype: float64
```

In [24]:
```
data.var()
```

Out[24]:
```
RowNumber          8.334167e+06
CustomerId         5.174815e+09
CreditScore        9.341860e+03
Age                1.099941e+02
Tenure             8.364673e+00
Balance            3.893436e+09
NumOfProducts      3.383218e-01
HasCrCard          2.077905e-01
IsActiveMember     2.497970e-01
EstimatedSalary    3.307457e+09
Exited             1.622225e-01
dtype: float64
```

In [25]:
```
data.describe()
```

Out[25]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	1
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	76485.889288	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	1.556570e+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	

In [26]:
```
data.skew()
```

Out[26]:
```
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
```

In [27]:
```
data.kurt()
```

Out[27]:
```
RowNumber         -1.200000
CustomerId        -1.196113
CreditScore       -0.425726
Age                1.395347
Tenure            -1.165225
Balance           -1.489412
NumOfProducts      0.582981
HasCrCard         -1.186973
IsActiveMember    -1.996747
EstimatedSalary   -1.181518
Exited             0.165671
dtype: float64
```

In [28]:
```
quantile= data['Age'].quantile(q=[0.75, 0.25])
quantile
```

Out[28]:
```
0.75    44.0
0.25    32.0
Name: Age, dtype: float64
```

5. Handle the Missing values.

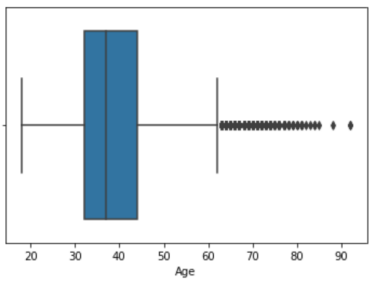
In [29]:
```
data.isna().any()
```

Out[29]:
```
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
```

6. Find the outliers and replace the outliers

In [30]:
```
sns.boxplot(data['Age'])
```

Out[30]:
```
<AxesSubplot:xlabel='Age'>
```



In [31]:
```
data.mean()
```

Out[31]:
```
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 [32]:
```
qut= data.quantile(q=[0.25,0.75])
qut
```

Out[32]:

	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 [33]:
```
irq=qut.loc[0.75]- qut.loc[0.25] # q3-q1
irq
```

Out[33]:
```
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 [34]:
```
lower= qut.loc[0.25]+(1.5*irq)
lower
```

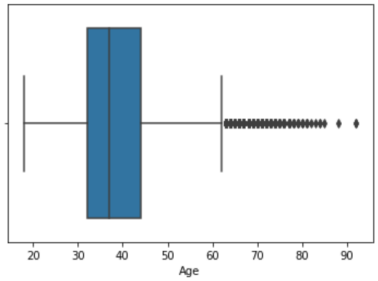
Out[34]:
```
RowNumber          1.000000e+04
CustomerId         1.581559e+07
CreditScore        7.850000e+02
Age                5.000000e+01
Tenure             9.000000e+00
Balance            1.914664e+05
NumOfProducts      2.500000e+00
HasCrCard          1.500000e+00
IsActiveMember     1.500000e+00
EstimatedSalary    1.985813e+05
Exited             0.000000e+00
dtype: float64
```

In [35]:
```
upper= qut.loc[0.75]+(1.5*irq)
upper
```

Out[35]:
```
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 [36]:
```
sns.boxplot(data['Age'])
```

Out[36]: <AxesSubplot:xlabel='Age'>



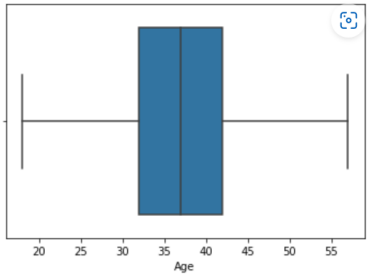
In [37]:
```
data['Age'].mean()
```

Out[37]: 38.9218

In [38]:
```
data['Age']=np.where(data['Age']>57,39, data['Age'])
```

In [39]:
```
sns.boxplot(data['Age'])
```

Out[39]: <AxesSubplot:xlabel='Age'>



7. Check for Categorical columns and perform encoding.

In [40]:
```
data.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
```

In [41]:
```
data.head()
```

Out[41]:

	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

In [42]:
```
data.Gender.unique()
```

Out[42]:
```
array(['Female', 'Male'], dtype=object)
```

In [43]:
```
data.Geography.unique()
```

Out[43]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

In [44]:
```
data['Gender'].replace({'Female':0, 'Male': 1 }, inplace=True)
data['Geography'].replace({'France':0,'Germany':1, 'Spain':2}, inplace=True)
data.head()
```

Out[44]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88

Out[44]:

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

In [ ]:

In [ ]:

In [45]:
```
data_d= pd.get_dummies(data,columns = ['Surname'])
data_d.head()
```

Out[45]:

	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	...	Surname_Zinachukwudi	Surname_Zito
0	1	15634602	619	0	0	42	2	0.00	1	1	...	0	0
1	2	15647311	608	2	0	41	1	83807.86	1	0	...	0	0
2	3	15619304	502	0	0	42	8	159660.80	3	1	...	0	0
3	4	15701354	699	0	0	39	1	0.00	2	0	...	0	0
4	5	15737888	850	2	0	43	2	125510.82	1	1	...	0	0

4	5	15737888	850	2	0	43	2	125510.82	1	1	...	0	0

5 rows × 2945 columns

8. Split the data into dependent and independent variables.

In [46]:
```
data.head()
```

Out[46]:

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

In [47]:
```
x=data_d.drop(columns= ['EstimatedSalary']).values
y=data_d['EstimatedSalary'].values
x
```

Out[47]:
```
array([[1.0000000e+00, 1.5634602e+07, 6.1900000e+02, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00],
       [2.0000000e+00, 1.5647311e+07, 6.0800000e+02, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00],
       [3.0000000e+00, 1.5619304e+07, 5.0200000e+02, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00],
       ...,
       [9.9980000e+03, 1.5584532e+07, 7.0900000e+02, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00],
       [9.9990000e+03, 1.5682355e+07, 7.7200000e+02, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00],
       [1.0000000e+04, 1.5628319e+07, 7.9200000e+02, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00]])
```

In [48]:
```
y
```

Out[48]:
```
array([101348.88, 112542.58, 113931.57, ...,  42085.58,  92888.52,
        38190.78])
```

9. Scale the independent variables

In [49]:
```
from sklearn.preprocessing import scale#, StandardScaler
# Scale - Similar to std
```

In [50]:
```
x = scale(x)
x
```

Out[50]:
```
array([[-1.73187761, -0.78321342, -0.32622142, ..., -0.0100005 ,
        -0.01414355, -0.01414355],
       [-1.7315312 , -0.60653412, -0.44003595, ..., -0.0100005 ,
        -0.01414355, -0.01414355],
       [-1.73118479, -0.99588476, -1.53679418, ..., -0.0100005 ,
        -0.01414355, -0.01414355],
       ...,
       [ 1.73118479, -1.47928179,  0.60498839, ..., -0.0100005 ,
        -0.01414355, -0.01414355],
       [ 1.7315312 , -0.11935577,  1.25683526, ..., -0.0100005 ,
        -0.01414355, -0.01414355],
       [ 1.73187761, -0.87055909,  1.46377078, ..., -0.0100005 ,
        -0.01414355, -0.01414355]])
```

10. Split the data into training and testing

In [51]:
```
from sklearn.model_selection import train_test_split
```

In [52]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
```

In [53]:
```
print(x_train.shape, x_test.shape)
```

```
(8000, 2944) (2000, 2944)
```

In [ ]:

In [ ]: