# ---------- House price Analysis, Visualization and Prediction - ----------

## TOTAL PROJECT DIVIDED INTO THREE PARTS

1. ANALYSIS
   - It loads the data and tells all the information about the dataframe
   - Some are
     - data.head(), data.tail(), data.shape, data.sample() etc..
2. VISUALIZATION
   - Representation of the data in graph
   - These of mainly two types
     - Matplotlib
     - Seaborn
3. PREDICTION
   - Predicting the data using Regression or classification model

# 1. ANALYSIS OF THE DATA

## IMPORT LIBRARIES

In [1]:

```
import numpy as np                                                    #importing
import pandas as pd                                                   #importing
import seaborn as sns                                                 #import sea
import matplotlib.pyplot as plt                                       #import mat
from sklearn.linear_model import LogisticRegression                   #import Log
from sklearn.model_selection import train_test_split                  #import tra
```

## Loading the data set

In [2]:

```
data = pd.read_csv("Houseprice.csv")
```

## Reading full data and displaying

In [3]:

```
1 data
```

Out[3]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | RE |
|---|---|---|---|---|---|---|---|
| **0** | Owner | 0 | 0 | 2 | BHK | 1300.236407 | |
| **1** | Dealer | 0 | 0 | 2 | BHK | 1275.000000 | |
| **2** | Owner | 0 | 0 | 2 | BHK | 933.159722 | |
| **3** | Owner | 0 | 1 | 2 | BHK | 929.921143 | |
| **4** | Dealer | 1 | 0 | 2 | BHK | 999.009247 | |
| **5** | Owner | 0 | 0 | 3 | BHK | 1250.000000 | |
| **6** | Dealer | 0 | 0 | 3 | BHK | 1495.053957 | |
| **7** | Owner | 0 | 1 | 3 | BHK | 1181.012946 | |
| **8** | Dealer | 0 | 1 | 2 | BHK | 1040.000000 | |
| **9** | Owner | 0 | 1 | 2 | BHK | 879.120879 | |
| **10** | Owner | 0 | 0 | 3 | BHK | 1350.308642 | |
| **11** | Dealer | 0 | 0 | 2 | BHK | 1333.010179 | |
| **12** | Owner | 0 | 0 | 2 | BHK | 927.177902 | |
| **13** | Owner | 0 | 1 | 2 | BHK | 1122.171946 | |
| **14** | Owner | 0 | 0 | 1 | BHK | 649.983750 | |
| **15** | Dealer | 1 | 1 | 3 | BHK | 1394.117647 | |
| **16** | Owner | 0 | 0 | 3 | BHK | 1800.084710 | |
| **17** | Dealer | 1 | 1 | 3 | BHK | 2124.896706 | |
| **18** | Owner | 0 | 0 | 2 | BHK | 1100.000000 | |
| **19** | Dealer | 0 | 1 | 3 | BHK | 2178.649237 | |
| **20** | Owner | 0 | 0 | 2 | BHK | 881.143529 | |
| **21** | Dealer | 0 | 0 | 2 | BHK | 944.881890 | |
| **22** | Dealer | 0 | 1 | 3 | BHK | 1310.147689 | |
| **23** | Dealer | 0 | 0 | 1 | BHK | 630.000630 | |
| **24** | Dealer | 0 | 0 | 2 | BHK | 1219.809710 | |
| **25** | Owner | 0 | 0 | 2 | BHK | 780.141844 | |
| **26** | Owner | 0 | 0 | 3 | BHK | 1600.000000 | |
| **27** | Dealer | 0 | 0 | 2 | BHK | 1180.412371 | |
| **28** | Owner | 1 | 0 | 2 | BHK | 1000.000000 | |
| **29** | Dealer | 0 | 0 | 2 | BHK | 1000.000000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **29421** | Dealer | 0 | 0 | 2 | BHK | 1409.064497 | |

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | RE |
|---|---|---|---|---|---|---|---|
| **29422** | Owner | 0 | 0 | 2 | BHK | 646.987465 | |
| **29423** | Dealer | 0 | 0 | 3 | BHK | 1250.000000 | |
| **29424** | Owner | 0 | 0 | 1 | BHK | 632.069195 | |
| **29425** | Dealer | 0 | 0 | 4 | BHK | 2800.074669 | |
| **29426** | Dealer | 0 | 0 | 3 | BHK | 1816.666667 | |
| **29427** | Owner | 0 | 0 | 2 | BHK | 1200.000000 | |
| **29428** | Owner | 0 | 0 | 1 | BHK | 800.000000 | |
| **29429** | Builder | 0 | 1 | 2 | BHK | 974.930362 | |
| **29430** | Owner | 0 | 1 | 3 | BHK | 1265.060241 | |
| **29431** | Owner | 0 | 0 | 2 | BHK | 712.105799 | |
| **29432** | Owner | 0 | 0 | 4 | BHK | 1212.121212 | |
| **29433** | Owner | 0 | 0 | 2 | BHK | 1195.028681 | |
| **29434** | Owner | 0 | 0 | 3 | BHK | 1700.404858 | |
| **29435** | Dealer | 0 | 1 | 4 | BHK | 2189.160468 | |
| **29436** | Owner | 1 | 1 | 2 | BHK | 1175.132676 | |
| **29437** | Dealer | 0 | 0 | 2 | BHK | 760.116253 | |
| **29438** | Builder | 1 | 1 | 3 | BHK | 1560.000000 | |
| **29439** | Dealer | 1 | 1 | 2 | BHK | 975.609756 | |
| **29440** | Builder | 0 | 1 | 3 | BHK | 1652.634861 | |
| **29441** | Owner | 0 | 0 | 3 | BHK | 1500.375094 | |
| **29442** | Dealer | 0 | 0 | 3 | BHK | 1554.968123 | |
| **29443** | Dealer | 0 | 1 | 3 | BHK | 1161.194975 | |
| **29444** | Dealer | 1 | 1 | 1 | BHK | 752.049334 | |
| **29445** | Owner | 0 | 0 | 2 | BHK | 1062.134891 | |
| **29446** | Owner | 0 | 0 | 3 | BHK | 2500.000000 | |
| **29447** | Owner | 0 | 0 | 2 | BHK | 769.230769 | |
| **29448** | Dealer | 0 | 0 | 2 | BHK | 1022.641509 | |
| **29449** | Owner | 0 | 0 | 2 | BHK | 927.079009 | |
| **29450** | Dealer | 0 | 1 | 2 | BHK | 896.774194 | |

29451 rows × 12 columns

## Reading first five data and displaying

In [4]:

```
1 data.head()
```

Out[4]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | READY |
|---|---|---|---|---|---|---|---|
| **0** | Owner | 0 | 0 | 2 | BHK | 1300.236407 | |
| **1** | Dealer | 0 | 0 | 2 | BHK | 1275.000000 | |
| **2** | Owner | 0 | 0 | 2 | BHK | 933.159722 | |
| **3** | Owner | 0 | 1 | 2 | BHK | 929.921143 | |
| **4** | Dealer | 1 | 0 | 2 | BHK | 999.009247 | |

## Reading last five data and displaying

In [5]:

```
1 data.tail()
```

Out[5]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | RE |
|---|---|---|---|---|---|---|---|
| **29446** | Owner | 0 | 0 | 3 | BHK | 2500.000000 | |
| **29447** | Owner | 0 | 0 | 2 | BHK | 769.230769 | |
| **29448** | Dealer | 0 | 0 | 2 | BHK | 1022.641509 | |
| **29449** | Owner | 0 | 0 | 2 | BHK | 927.079009 | |
| **29450** | Dealer | 0 | 1 | 2 | BHK | 896.774194 | |

## Shape of the data

In [6]:

```
1 data.shape
```

Out[6]:

```
(29451, 12)
```

## Sample data i.e random data from the data set

In [7]:

```
1 data.sample()
```

Out[7]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | RE |
|---|---|---|---|---|---|---|---|
| **25603** | Owner | 0 | 0 | 15 | BHK | 8000.0 | |

## Checking null values using isnull()

In [8]:

```
1 data.isnull().sum()
```

Out[8]:

```
POSTED_BY               0
UNDER_CONSTRUCTION      0
RERA                    0
BHK_NO.                 0
BHK_OR_RK               0
SQUARE_FT               0
READY_TO_MOVE           0
RESALE                  0
ADDRESS                 0
LONGITUDE               0
LATITUDE                0
TARGET(PRICE_IN_LACS)   0
dtype: int64
```

In [9]:

```
1 data.isnull().sum().sum()
```

Out[9]:

```
0
```

## Describing data i.e finding mean, standard deviation, min, max, etc..

In [10]:

```
1 data.describe()
```

Out[10]:

| | UNDER_CONSTRUCTION | RERA | BHK_NO. | SQUARE_FT | READY_TO_MOVE |
|---|---|---|---|---|---|
| **count** | 29451.000000 | 29451.000000 | 29451.000000 | 2.945100e+04 | 29451.000000 |
| **mean** | 0.179756 | 0.317918 | 2.392279 | 1.980217e+04 | 0.820244 |
| **std** | 0.383991 | 0.465675 | 0.879091 | 1.901335e+06 | 0.383991 |
| **min** | 0.000000 | 0.000000 | 1.000000 | 3.000000e+00 | 0.000000 |
| **25%** | 0.000000 | 0.000000 | 2.000000 | 9.000211e+02 | 1.000000 |
| **50%** | 0.000000 | 0.000000 | 2.000000 | 1.175057e+03 | 1.000000 |
| **75%** | 0.000000 | 1.000000 | 3.000000 | 1.550688e+03 | 1.000000 |
| **max** | 1.000000 | 1.000000 | 20.000000 | 2.545455e+08 | 1.000000 |

## Checking datatype of the each coulmns in a data set

In [11]:

```
1 data.dtypes
```

Out[11]:

```
POSTED_BY                object
UNDER_CONSTRUCTION        int64
RERA                      int64
BHK_NO.                   int64
BHK_OR_RK                object
SQUARE_FT               float64
READY_TO_MOVE             int64
RESALE                    int64
ADDRESS                  object
LONGITUDE               float64
LATITUDE                float64
TARGET(PRICE_IN_LACS)   float64
dtype: object
```

## Collecting information using info()

In [12]:

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29451 entries, 0 to 29450
Data columns (total 12 columns):
POSTED_BY                29451 non-null object
UNDER_CONSTRUCTION       29451 non-null int64
RERA                     29451 non-null int64
BHK_NO.                  29451 non-null int64
BHK_OR_RK                29451 non-null object
SQUARE_FT                29451 non-null float64
READY_TO_MOVE            29451 non-null int64
RESALE                   29451 non-null int64
ADDRESS                  29451 non-null object
LONGITUDE                29451 non-null float64
LATITUDE                 29451 non-null float64
TARGET(PRICE_IN_LACS)    29451 non-null float64
dtypes: float64(4), int64(5), object(3)
memory usage: 2.7+ MB
```

### Displaying all columns

In [13]:

```
1 data.columns
```

Out[13]:

```
Index(['POSTED_BY', 'UNDER_CONSTRUCTION', 'RERA', 'BHK_NO.', 'BHK_OR_RK',
       'SQUARE_FT', 'READY_TO_MOVE', 'RESALE', 'ADDRESS', 'LONGITUDE',
       'LATITUDE', 'TARGET(PRICE_IN_LACS)'],
      dtype='object')
```

### Counting the total number of values

In [14]:

```
1 data.count().sum()
```

Out[14]:

```
353412
```

## Checking dummies

In [15]:

```
1 pd.get_dummies(data["BHK_OR_RK"]).head()
```

Out[15]:

| | BHK | RK |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 1 | 0 |
| **2** | 1 | 0 |
| **3** | 1 | 0 |
| **4** | 1 | 0 |

In [16]:

```
1 data.head()
```

Out[16]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | READY |
|---|---|---|---|---|---|---|---|
| **0** | Owner | 0 | 0 | 2 | BHK | 1300.236407 | |
| **1** | Dealer | 0 | 0 | 2 | BHK | 1275.000000 | |
| **2** | Owner | 0 | 0 | 2 | BHK | 933.159722 | |
| **3** | Owner | 0 | 1 | 2 | BHK | 929.921143 | |
| **4** | Dealer | 1 | 0 | 2 | BHK | 999.009247 | |

In [17]:

```
1 data["BHK_OR_RK"].value_counts()
```

Out[17]:

```
BHK    29427
RK        24
Name: BHK_OR_RK, dtype: int64
```

In [18]:

```
pd.get_dummies(data["POSTED_BY"]).head()
```

Out[18]:

| | Builder | Dealer | Owner |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **1** | 0 | 1 | 0 |
| **2** | 0 | 0 | 1 |
| **3** | 0 | 0 | 1 |
| **4** | 0 | 1 | 0 |

In [19]:

```
data["POSTED_BY"].value_counts()
```

Out[19]:

```
Dealer     18291
Owner      10538
Builder      622
Name: POSTED_BY, dtype: int64
```

In [20]:

```
data.head()
```

Out[20]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | READY |
|---|---|---|---|---|---|---|---|
| **0** | Owner | 0 | 0 | 2 | BHK | 1300.236407 | |
| **1** | Dealer | 0 | 0 | 2 | BHK | 1275.000000 | |
| **2** | Owner | 0 | 0 | 2 | BHK | 933.159722 | |
| **3** | Owner | 0 | 1 | 2 | BHK | 929.921143 | |
| **4** | Dealer | 1 | 0 | 2 | BHK | 999.009247 | |

## Pre-processing i.e Used to convert categorical data into numerical data

## For single coulmn

In [21]:

```
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
```

In [22]:

```
data["BHK_OR_RK"] = le.fit_transform(data["BHK_OR_RK"])
```

In [23]:

```
data.head()
```

Out[23]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | SQUARE_FT | READY |
|---|---|---|---|---|---|---|---|
| **0** | Owner | 0 | 0 | 2 | 0 | 1300.236407 | |
| **1** | Dealer | 0 | 0 | 2 | 0 | 1275.000000 | |
| **2** | Owner | 0 | 0 | 2 | 0 | 933.159722 | |
| **3** | Owner | 0 | 1 | 2 | 0 | 929.921143 | |
| **4** | Dealer | 1 | 0 | 2 | 0 | 999.009247 | |

In [24]:

```
data["SQUARE_FT"].mean()
```

Out[24]:

```
19802.170190334633
```

In [25]:

```
data["SQUARE_FT"].std()
```

Out[25]:

```
1901334.9125039035
```

In [26]:

```
data.rename(columns={'TARGET(PRICE_IN_LACS)': 'PRICE', 'SQUARE_FT': 'AREA'}, inplace=T
```

In [27]:

```
1 data.head()
```

Out[27]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | AREA | READY |
|---|---|---|---|---|---|---|---|
| **0** | Owner | 0 | 0 | 2 | 0 | 1300.236407 | |
| **1** | Dealer | 0 | 0 | 2 | 0 | 1275.000000 | |
| **2** | Owner | 0 | 0 | 2 | 0 | 933.159722 | |
| **3** | Owner | 0 | 1 | 2 | 0 | 929.921143 | |
| **4** | Dealer | 1 | 0 | 2 | 0 | 999.009247 | |

## For multiple coulumns

In [28]:

```
1
2 from sklearn.preprocessing import LabelEncoder
3 cols = ['POSTED_BY',"ADDRESS"]
4 le = LabelEncoder()
5 for col in cols:
6     data[col] = le.fit_transform(data[col])
```

In [29]:

```
1 data.head()
```

Out[29]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_RK | AREA | READY |
|---|---|---|---|---|---|---|---|
| **0** | 2 | 0 | 0 | 2 | 0 | 1300.236407 | |
| **1** | 1 | 0 | 0 | 2 | 0 | 1275.000000 | |
| **2** | 2 | 0 | 0 | 2 | 0 | 933.159722 | |
| **3** | 2 | 0 | 1 | 2 | 0 | 929.921143 | |
| **4** | 1 | 1 | 0 | 2 | 0 | 999.009247 | |

# VISUALIZING DATA

- Importance of Visualization
  - To understand the data clearly when there is large data
  - to understand easier

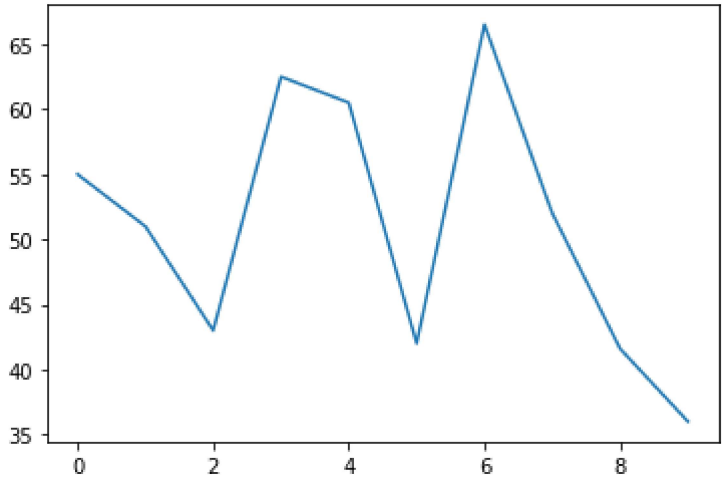
    - For better and quick understanding
- Types of data storage
    - Files like records with csv,excell,html,py,ipnb,etcc..
    - Images/vedios
    - Graphs/plots

- pyplot Module is used to represent the data in the graphical format.
- The various types of plots available in matplotlib are
    1. Line plot : Line plot is the linear grap, in which all points are join by a line.
    2. Scatter Plot. It show the relation between two variables
    3. Histogram : Probability Distribution
    4. Bar Graph
    5. Pie Chart
    6. Box Plot

# line plot

In [30]:

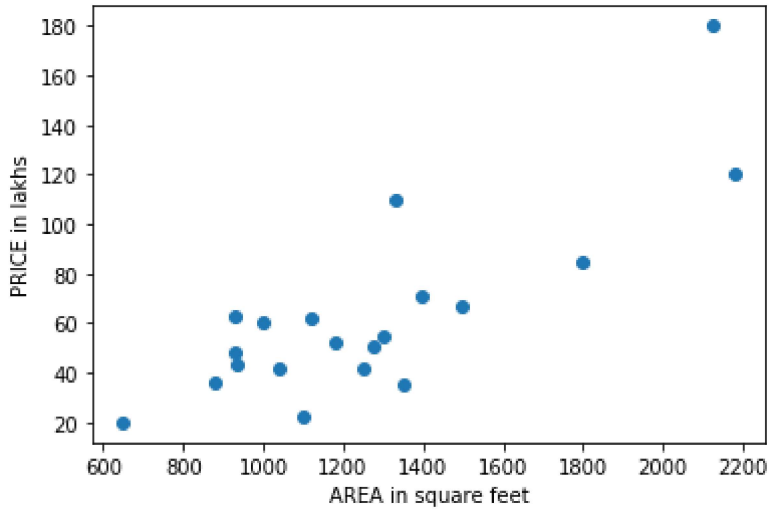
```
## draw a line plot
plt.plot(data["PRICE"][:10])
plt.show()
```




# Scatter plot

In [31]:

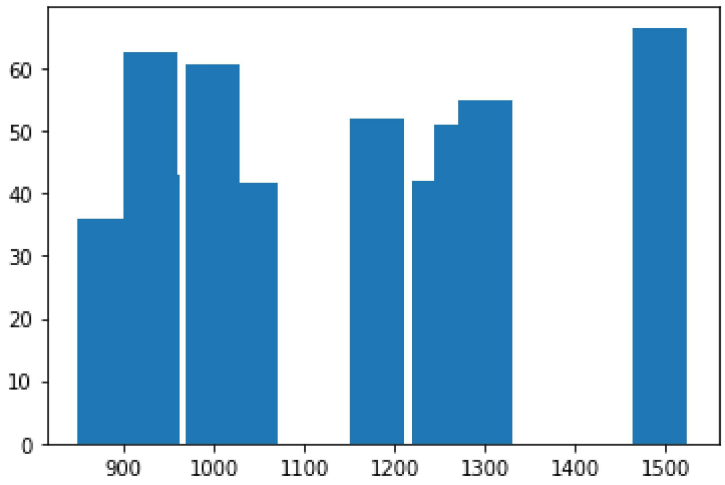
```
## Scatter plot
plt.scatter(data["AREA"][:20],data["PRICE"][:20])
plt.xlabel("AREA in square feet")
plt.ylabel("PRICE in lakhs")
plt.show()
```




## bar graph

In [32]:

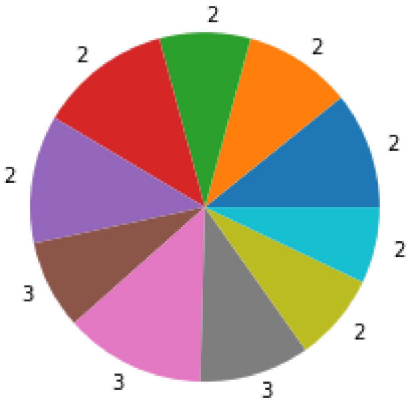
```
# bar graph
plt.bar(data["AREA"][:10],data["PRICE"][:10],width=60)
plt.show()
```




# pie chart

In [33]:

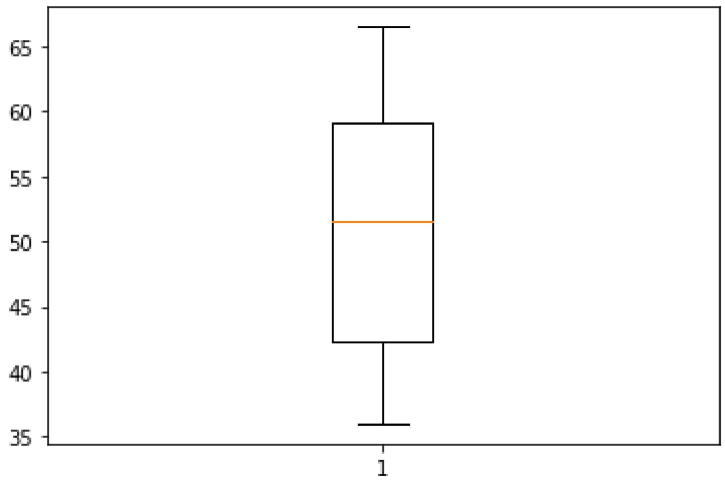
```
## pie chart
plt.pie(data["PRICE"][:10],labels=data["BHK_NO."][:10])
plt.show()
```




# boxplot

In [34]:

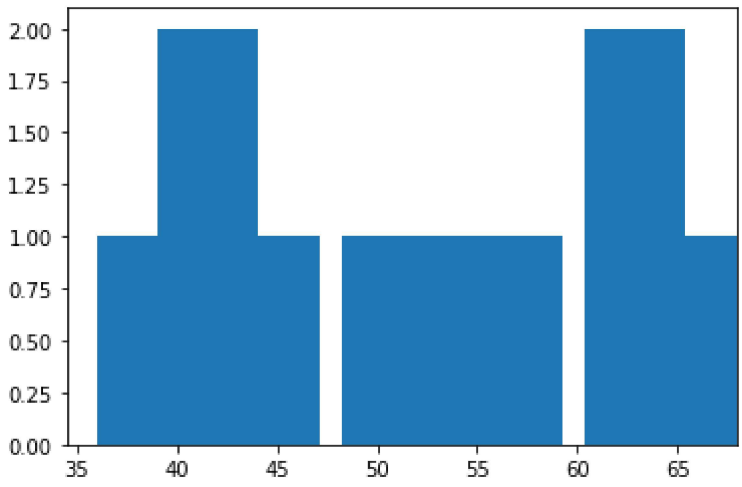
```
## boxplot
plt.boxplot(data["PRICE"][:10])
plt.show()
```




# Histogram

In [35]:

```
plt.hist(data["PRICE"][:10],width=5)
plt.show()
```




# Seaborn

- The various types of plots available in Seaborn are
  - Line plot
  - Distribution plot
  - Joint plot
  - Heat map
  - Pairplot
  - Cluster plot
  - Lmplot

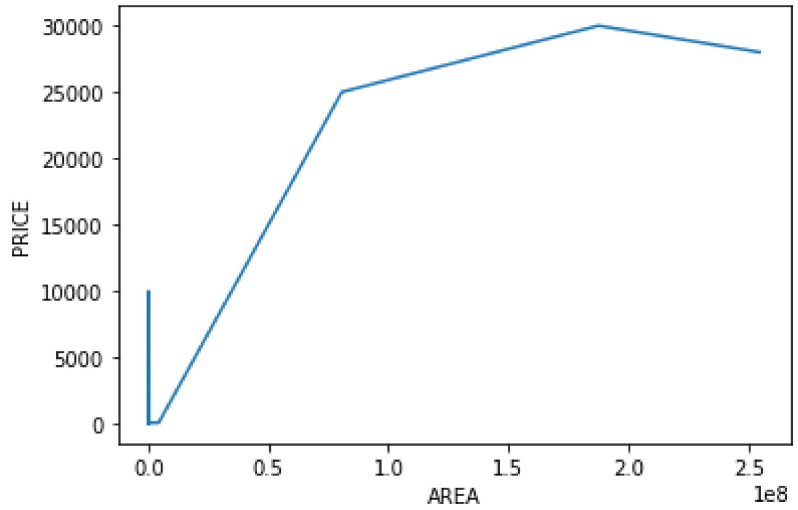
# Lineplot

In [36]:

```
1 sns.lineplot(x="AREA",y="PRICE",data=data)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e38dfeed68>
```



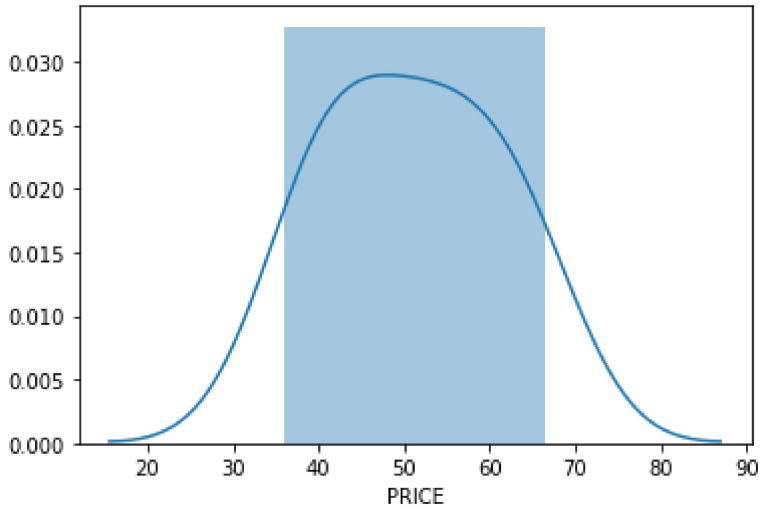

# Distplot

In [37]:

```
1 sns.distplot(data['PRICE'][:10])
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e38db1d198>
```

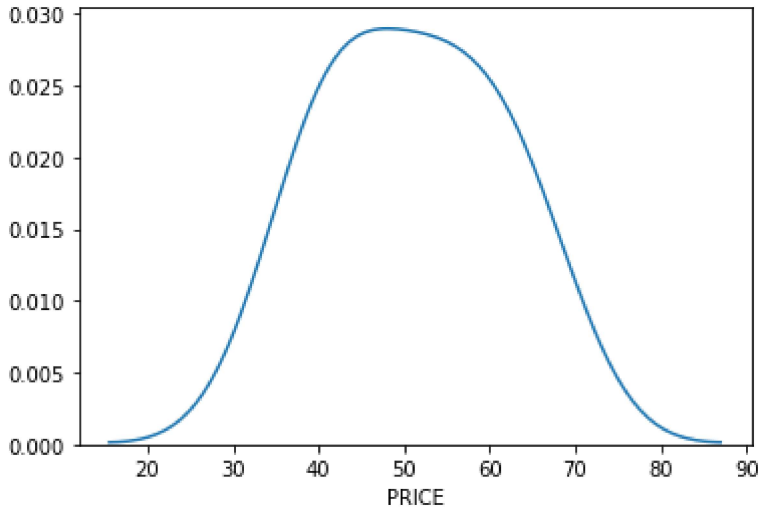



In [38]:

```
1 sns.distplot(data['PRICE'][:10],hist=False)
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e38e08a2e8>
```



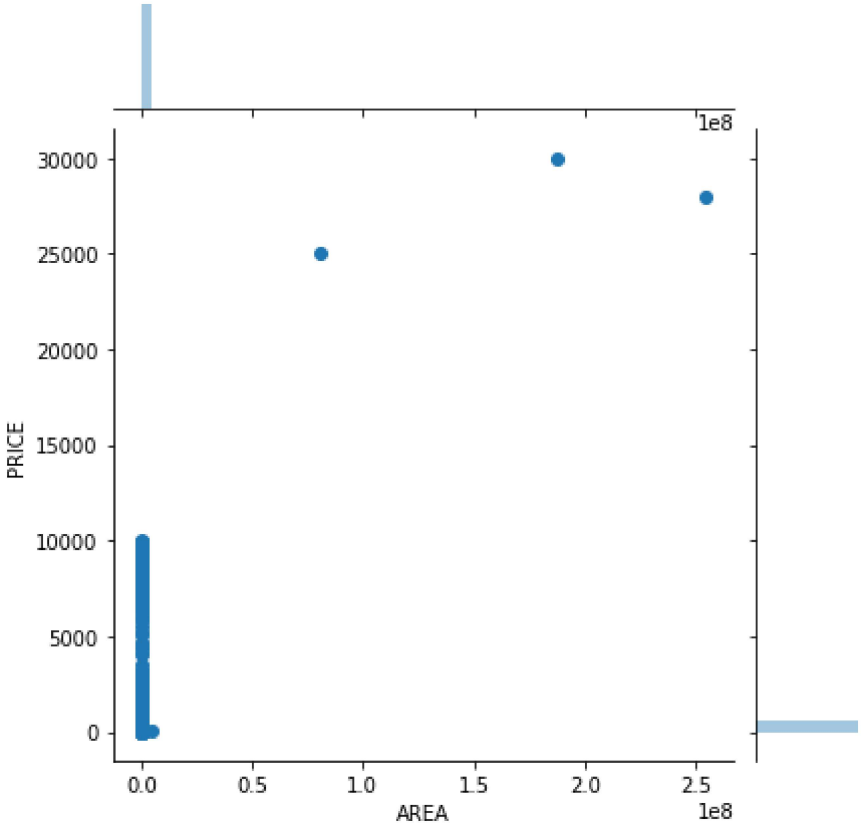

# Jointplot

In [39]:

```
1 sns.jointplot(x="AREA"[:5],y="PRICE"[:5],data=data)
```

Out[39]:

```
<seaborn.axisgrid.JointGrid at 0x1e38e0e8c50>
```



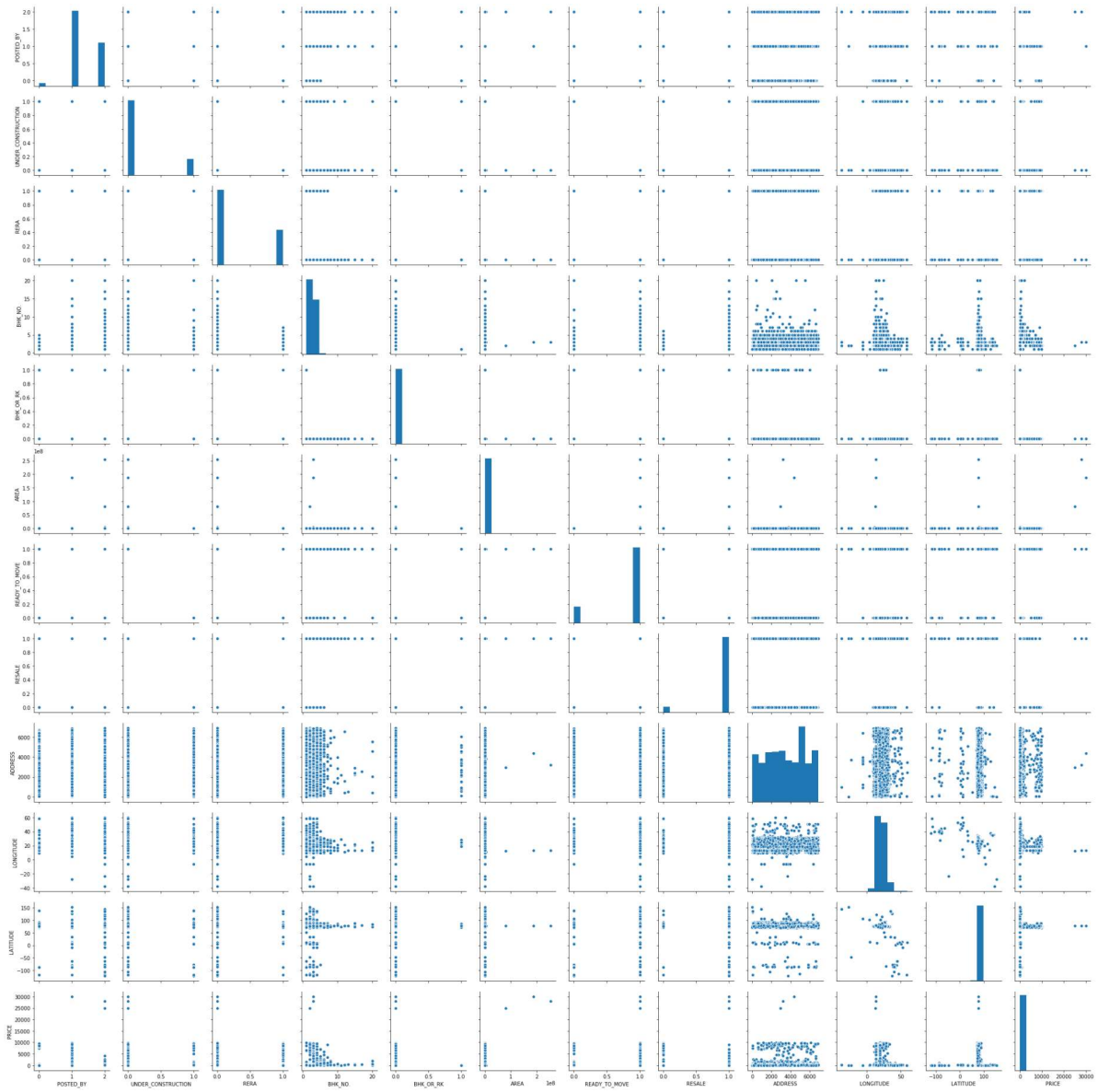

# Pairplot

In [40]:

```
1 sns.pairplot(data)
```

Out[40]:

```
<seaborn.axisgrid.PairGrid at 0x1e38e08a1d0>
```




# Correlation of the data

In [41]:

```
1 data.corr()
```

Out[41]:

| | POSTED_BY | UNDER_CONSTRUCTION | RERA | BHK_NO. | BHK_OR_ |
|---|---|---|---|---|---|
| **POSTED_BY** | 1.000000 | -0.263473 | -0.285316 | -0.072466 | -0.027 |
| **UNDER_CONSTRUCTION** | -0.263473 | 1.000000 | 0.363826 | -0.040712 | 0.020 |
| **RERA** | -0.285316 | 0.363826 | 1.000000 | 0.009547 | 0.006 |
| **BHK_NO.** | -0.072466 | -0.040712 | 0.009547 | 1.000000 | -0.045 |
| **BHK_OR_RK** | -0.027895 | 0.020719 | 0.006056 | -0.045231 | 1.000 |
| **AREA** | 0.005415 | -0.004204 | -0.006229 | 0.005303 | -0.000 |
| **READY_TO_MOVE** | 0.263473 | -1.000000 | -0.363826 | 0.040712 | -0.020 |
| **RESALE** | 0.332280 | -0.347405 | -0.270351 | 0.014581 | -0.029 |
| **ADDRESS** | -0.040621 | -0.013684 | 0.073604 | 0.042386 | -0.011 |
| **LONGITUDE** | -0.049164 | 0.006440 | 0.104976 | 0.068730 | 0.009 |
| **LATITUDE** | 0.025425 | -0.000381 | -0.065106 | 0.046930 | 0.000 |
| **PRICE** | -0.093328 | 0.055399 | 0.067636 | 0.112283 | -0.004 |

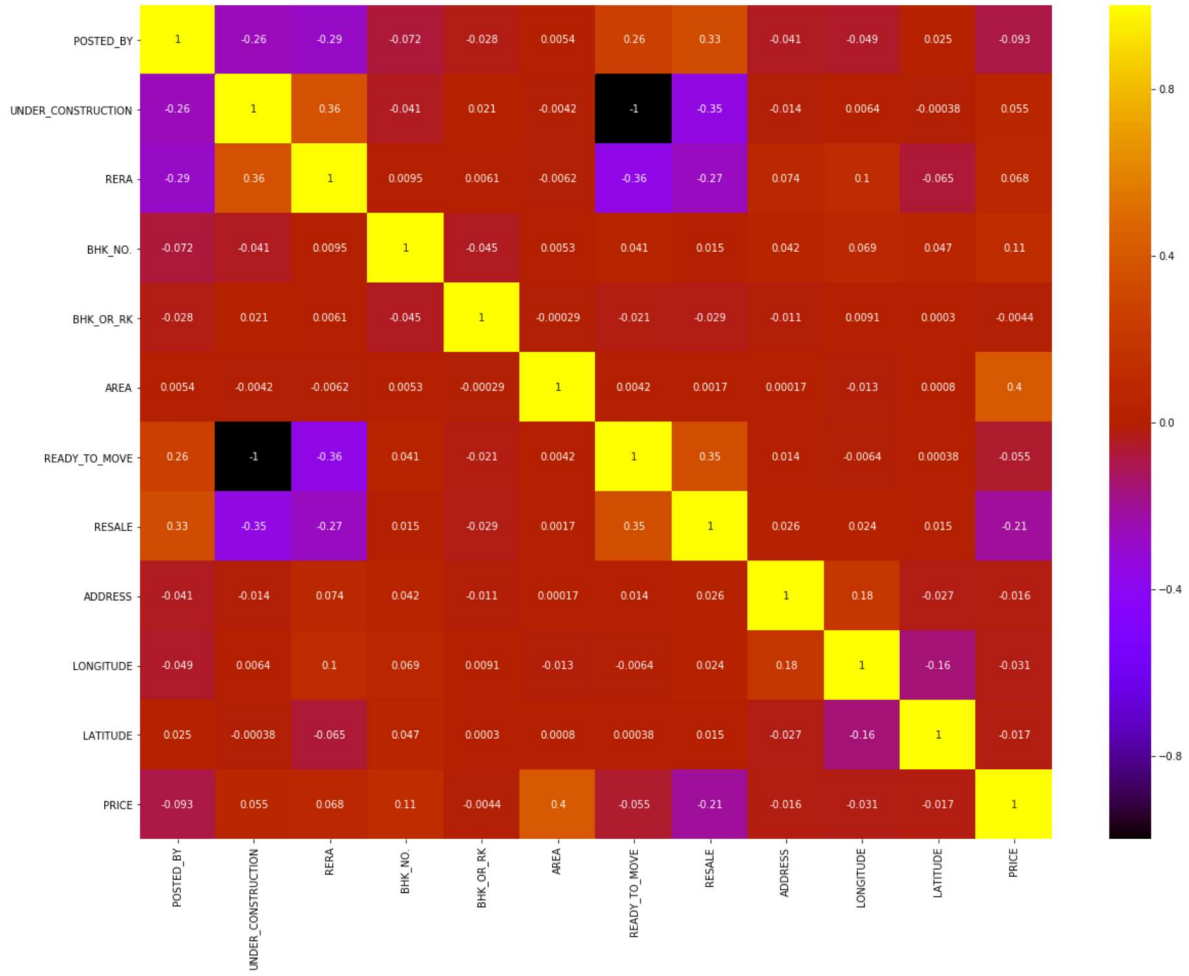
# Heatmap

In [42]:

```
1 plt.figure(figsize=(20,15))
2 sns.heatmap(data.corr(), annot=True, cmap='gnuplot')
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e393b496d8>
```



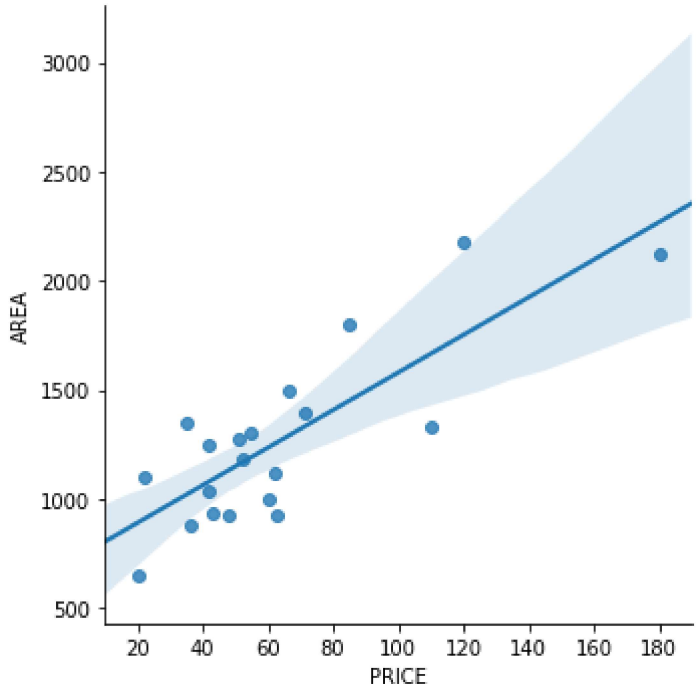

# LmPlot

In [43]:

```
1 sns.lmplot(x="PRICE",y="AREA",data=data[:20])
```

Out[43]:

```
<seaborn.axisgrid.FacetGrid at 0x1e39b01bb00>
```




# PREDICTING THE DATA

# Linear Regression

**Linear Regression:** Linear Regression is a linear approach to modeling the relation between dependent variable and one or more independent variables

**Two different types of linear regression models:**

● Simple linear regression : Simple Linear Regression uses one independent variable to predict the values of the dependent variable.

Ex: House price prediction({“Independent variable”:[‘room size’],”Dependent variable”:[“Price”]})

● Multiple linear regression: Multiple Linear Regression uses two or more independent variables and one dependent variable

Ex: House Price Prediction ({“Independent variable”:[‘room size’,’area size’,’no.of.rooms’],”Dependent variable”: [“Price”]})

**Train/Test splitting of data & Cross Validation**

As with most machine learning algorithms, we have to split our dataset into:

1. Training set: the data used to fit the model(Train the model)
2. Test set: the data partitioned away at the very start of the experiment (to provide an unbiased evaluation of the model)

**Here are the parameters:**

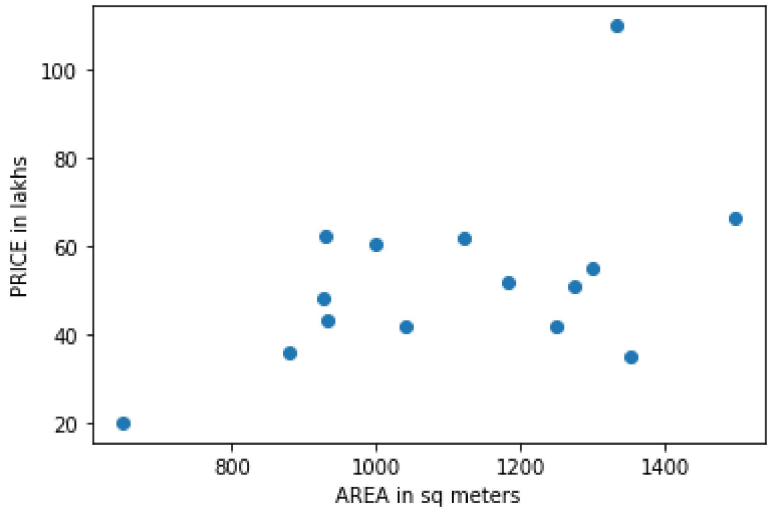
- train_size: the proportion of the dataset to include in the train split (between 0.0 and 1.0)
- test_size: the proportion of the dataset to include in the test split (between 0.0 and 1.0)
- random_state: the seed used by the random number generator [optional]

# Steps in Machine Learning Model

- loading raw Data
- data Analysis/ Exploration of the data
- Seperation of features and Target
- Seperate training data and testing data
- select alogorithm
- Build Ml modal - fit your trained data
- Prediction - testing data
- Evalute the model

In [44]:

```
plt.scatter(data["AREA"][:15],data["PRICE"][:15])
plt.xlabel("AREA in sq meters")
plt.ylabel("PRICE in lakhs")
plt.show()
```



In [45]:

```
X = data[["AREA"]] # input Feature
X.head()
```

Out[45]:

| | AREA |
|---|---|
| **0** | 1300.236407 |
| **1** | 1275.000000 |
| **2** | 933.159722 |
| **3** | 929.921143 |
| **4** | 999.009247 |

In [46]:

```
y = data["PRICE"] # Target variable
y
```

Out[46]:

```
0         55.0
1         51.0
2         43.0
3         62.5
4         60.5
5         42.0
6         66.5
7         52.0
8         41.6
9         36.0
10        35.0
11       110.0
12        48.0
13        62.0
14        20.0
15        71.1
16        85.0
17       180.0
18        22.0
19       120.0
20        45.0
21        42.0
22        55.0
23       300.0
24        50.0
25        27.5
26        46.0
27        22.9
28        39.0
29        12.5
         ...
29421     97.0
29422     16.0
29423    150.0
29424     19.0
29425    750.0
29426     54.5
29427     15.0
29428     24.0
29429     21.0
29430     63.0
29431     21.0
29432    110.0
29433     50.0
29434     63.0
29435     82.4
29436     62.0
29437     34.0
29438     54.6
29439    120.0
29440     53.0
29441     40.0
29442    200.0
29443    220.0
```

```
29444    100.0
29445     40.0
29446     45.0
29447     16.0
29448     27.1
29449     67.0
29450     27.8
Name: PRICE, Length: 29451, dtype: float64
```

In [47]:

```
1 from sklearn.linear_model import LinearRegression
2 lreg_model = LinearRegression()
3 lreg_model.fit(X,y)
```

Out[47]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [48]:

```
1 pred_values = lreg_model.predict(X)
2 pred_values
```

Out[48]:

```
array([140.32473665, 140.32122573, 140.27366854, ..., 140.28611734,
       140.27282259, 140.26860655])
```

In [49]:

```
1 from sklearn.metrics import r2_score
2 print("Accuracy score of the model is " ,r2_score(y,pred_values) * 100,"%")
```

```
Accuracy score of the model is  16.215491796475078 %
```

In [50]:

```
1 X.iloc[100]
```

Out[50]:

```
AREA    989.902989
Name: 100, dtype: float64
```

In [51]:

```
1 y.iloc[100]  # actual value
```

Out[51]:

```
50.0
```

In [52]:

```
lreg_model.predict([[989.902989]])  # 1246 predcited value from the model
```

Out[52]:

```
array([140.28156273])
```

In [53]:

```
140.28156273-50.0
```

Out[53]:

```
90.28156272999999
```

In [54]:

```
lreg_model.predict([[2010]])  # new data point
```

Out[54]:

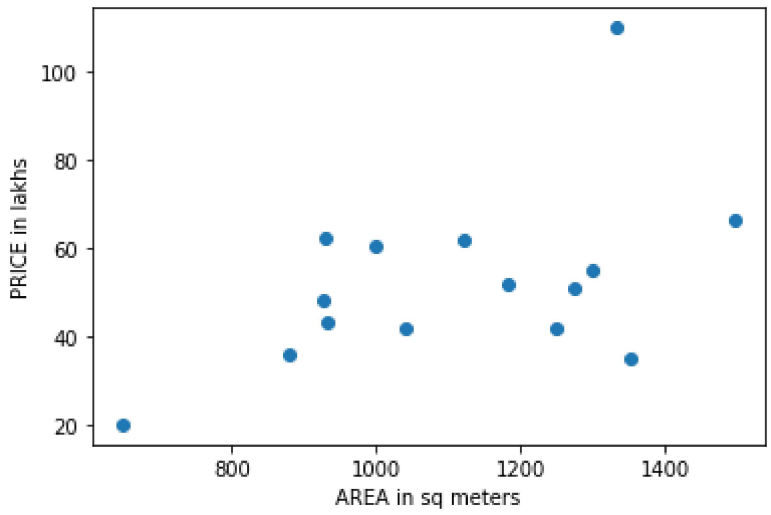
```
array([140.42347973])
```

In [55]:

```
from sklearn.metrics import mean_squared_error
```

In [56]:

```
plt.scatter(data["AREA"][:15],data["PRICE"][:15])
plt.xlabel("AREA in sq meters")
plt.ylabel("PRICE in lakhs")
plt.show()
```




In [57]:

```
y.min()
```

Out[57]:

```
0.25
```

In [58]:

```
1 y.max()
```

Out[58]:

```
30000.0
```

In [59]:

```
1 pred_values.min()
```

Out[59]:

```
140.14426371279333
```

In [60]:

```
1 pred_values.max()
```

Out[60]:

```
35552.78316254658
```

In [61]:

```
1 lreg_model.intercept_
```

Out[61]:

```
140.14384634954416
```

In [62]:

```
1 lreg_model.coef_
```

Out[62]:

```
array([0.00013912])
```

In [63]:

```
1 X.iloc[100]
```

Out[63]:

```
AREA    989.902989
Name: 100, dtype: float64
```

In [64]:

```
1 # y = mx+c
2 y = lreg_model.coef_*X.iloc[100] + lreg_model.intercept_
3 y
```

Out[64]:

```
AREA    140.281563
Name: 100, dtype: float64
```

In [65]:

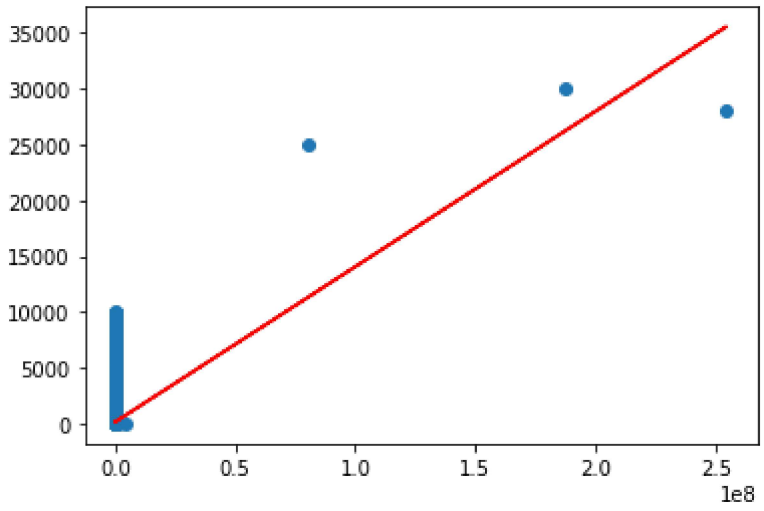
```
lreg_model.predict([[989.902989]])
```

Out[65]:

```
array([140.28156273])
```

In [66]:

```
plt.scatter(data["AREA"],data["PRICE"])
plt.plot(X,pred_values,c = "r")
plt.show()
```




## POLYNOMIAL REGRESSION

- Polynomial regression is one of the linear regression model.
- If the change between the dependent and independent variables is non-uniform,then we go for polynomial regression **Linear Equation**:

  y = mx + c
  - y is the target variable
  - c is the bias
  - x is the predictor
  - m is the weight of the regression equation

**Polynomial Equation**:

```
y = c + c1x + c2x^2 +c3x^3+.....+cnx^n

* y is the target variable
* x is the predictor
* c1,c2,cn are the weight of the regression equation
* c is the bias
```

**Polynomial regression for more input features**:

```
y = c + c1x1x2 +c2x1^2x2^2 + c3x1^3x2^3 +.....cnx1^nx2^n
```

- y is the target variable

- x1 and x2 are the predictors
- c1,c2,cn are the weight of the regression equation
- c is the bias

In [67]:

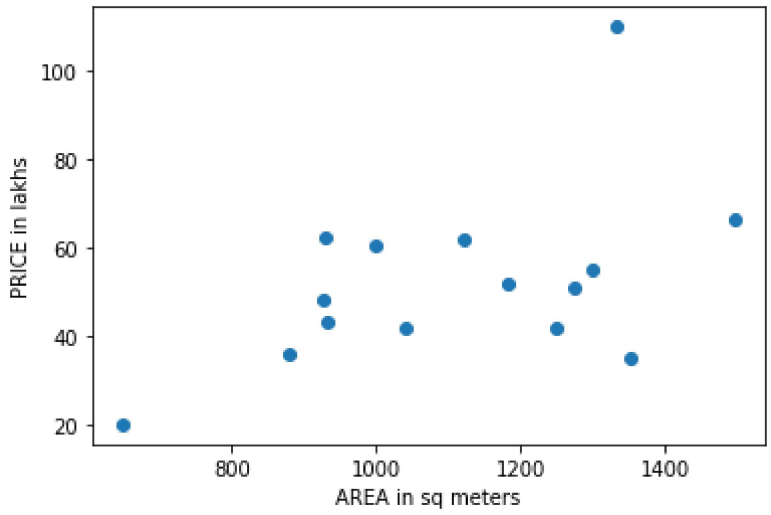
```
1 data.columns
```

Out[67]:

```
Index(['POSTED_BY', 'UNDER_CONSTRUCTION', 'RERA', 'BHK_NO.', 'BHK_OR_RK',
       'AREA', 'READY_TO_MOVE', 'RESALE', 'ADDRESS', 'LONGITUDE', 'LATITUD
E',
       'PRICE'],
      dtype='object')
```

In [68]:

```
1 plt.scatter(data["AREA"][:15],data["PRICE"][:15])
2 plt.xlabel("AREA in sq meters")
3 plt.ylabel("PRICE in lakhs")
4 plt.show()
```




In [69]:

```
1 x = data[["AREA"]]
2 y = data["PRICE"]
```

In [70]:

```
1 from  sklearn.preprocessing import PolynomialFeatures
2 poly = PolynomialFeatures(degree = 3)
3 x_poly = poly.fit_transform(x)
4 from sklearn.linear_model import LinearRegression
5 model = LinearRegression()
6 model.fit(x_poly,y)
```

Out[70]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [71]:

```
y_pred = model.predict(x_poly)
```

In [72]:

```
y_pred
```

Out[72]:

```
array([140.36838026, 140.36838008, 140.36837803, ..., 140.3683785 ,
       140.36837799, 140.36837784])
```

In [73]:

```
from sklearn.metrics import r2_score,mean_squared_error
print("r2_score:",r2_score(y,y_pred))
print("RMSE :",(mean_squared_error(y,y_pred)))
```

```
r2_score: 0.16578774681801844
RMSE : 359943.9172888587
```

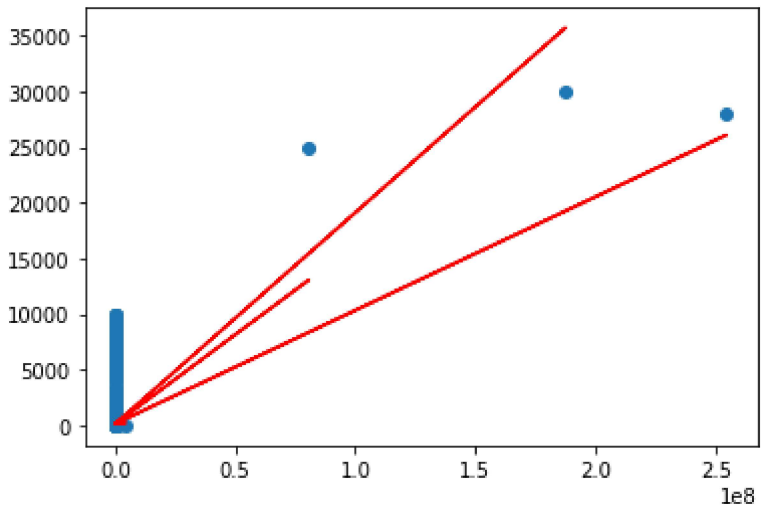
In [74]:

```
from sklearn.metrics import r2_score
print("Accuracy score of the model is " ,r2_score(y,y_pred) * 100,"%")
```

```
Accuracy score of the model is  16.578774681801846 %
```

In [75]:

```
plt.scatter(data["AREA"],data["PRICE"])
plt.plot(x,y_pred,c = "r")
plt.show()
```




## NOTE AS IT IS NOT CONTINOUS DATA WE COULD NOT ATTEMPT THE CLASSIFICATION MODELS

In [ ]:

```

```