importing required libraries and reading dataset into dataframe

In [2]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
```

In [3]:
```
df1 = pd.read_csv("D:\\20761A1276\\ML&DL\\Bengaluru_House_Data.csv")
df1.head()
```

Out[3]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

In [4]:
```
df1.shape
```

Out[4]:
```
(13320, 9)
```

REMOVE THE COLUMNS WHICH DON'T CONTRIBUTE MUCH TO FINAL VALUE

In [5]:
```
df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
df2.shape
```

Out[5]:
```
(13320, 5)
```

HANDLING NA VALUES

In [6]:
```
df2.isnull().sum()
```

Out[6]:
```
location       1
size          16
total_sqft     0
```

```
bath          73
price          0
dtype: int64
```

DROPING NA VALUE ROWS

In [7]:
```
df3 = df2.dropna()
df3.isnull().sum()
```

Out[7]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [8]:
```
df3.shape
```

Out[8]:
```
(13246, 5)
```

FEATURE ENGINEERING

Add new feature(integer) for bhk (Bedrooms Hall Kitchen)

In [9]:
```
df3['size'].unique()
```

Out[9]:
```
array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroom',
       '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',
       '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',
       '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',
       '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',
       '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

In [10]:
```
df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
df3.bhk.unique()
```

Out[10]:
```
C:\Users\admin\AppData\Local\Temp/ipykernel_8064/2716584372.py:1: SettingWithCopyWar
ning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18], dtype=int64)
```

Convert sqr_ft into float

In [11]:
```
df3['total_sqft'].unique()
```

Out[11]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [12]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [13]:
```
df3[~df3['total_sqft'].apply(is_float)].head()
```

Out[13]:

	location	size	total_sqft	bath	price	bhk
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2

Above shows that total_sqft can be a range (e.g. 2100-2850). For such case we can just take average of min and max value in the range. There are other cases such as 34.46Sq. Meter which one can convert to square ft using unit conversion. I am going to just drop such corner cases to keep things simple

In [14]:
```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

In [15]:
```
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4 = df4[df4.total_sqft.notnull()]
df4.head(2)
```

Out[15]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4

In [16]:
```
df4.head(10)
```

Out[16]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3

	location	size	total_sqft	bath	price	bhk
4	Kothanur	2 BHK	1200.0	2.0	51.00	2
5	Whitefield	2 BHK	1170.0	2.0	38.00	2
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3
9	Gandhi Bazar	6 Bedroom	1020.0	6.0	370.00	6

now calculate price per sqr_ft it can be used in outlier detection

In [17]:
```
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
```

Out[17]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000

handling categorical data--Location

In [18]:
```
df5.location.unique()
```

Out[18]:
```
array(['Electronic City Phase II', 'Chikka Tirupathi', 'Uttarahalli', ...,
       '12th cross srinivas nagar banshankari 3rd stage',
       'Havanur extension', 'Abshot Layout'], dtype=object)
```

In [19]:
```
len(df5.location.unique())
```

Out[19]:
```
1298
```

we need to decrease this number because when we convert this into numerical using encoders it will create 198 columns which is very large and can deviate our model.so we need to decrease this.

In [20]:
```
df5.location = df5.location.apply(lambda x: x.strip())
location_stats = df5['location'].value_counts(ascending=False)
location_stats
```

Out[20]:
```
Whitefield                    533
Sarjapur  Road              392
Electronic City               304
Kanakpura Road                264
Thanisandra                   235
                             ...
Rajanna Layout                  1
```

```
Subramanyanagar                1
Lakshmipura Vidyaanyapura      1
Malur Hosur Road               1
Abshot Layout                  1
Name: location, Length: 1287, dtype: int64
```

In [21]:
```
location_stats.values.sum()
```

Out[21]:
```
13200
```

In [22]:
```
len(location_stats[location_stats>10])
```

Out[22]:
```
240
```

In [23]:
```
len(location_stats)
```

Out[23]:
```
1287
```

In [24]:
```
len(location_stats[location_stats<=10])
```

Out[24]:
```
1047
```

DIMENSIONALITY REDUCTION

Any location having less than 10 data points should be tagged as "other" location. This way number of categories can be reduced by huge amount. Later on when we do one hot encoding, it will help us with having fewer dummy columns

In [25]:
```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[25]:
```
BTM 1st Stage                 10
Gunjur Palya                  10
Nagappa Reddy Layout          10
Sector 1 HSR Layout           10
Thyagaraja Nagar              10
                              ..
Rajanna Layout                 1
Subramanyanagar                1
Lakshmipura Vidyaanyapura      1
Malur Hosur Road               1
Abshot Layout                  1
Name: location, Length: 1047, dtype: int64
```

In [26]:
```
len(df5.location.unique())
```

Out[26]:
```
1287
```

In [27]:
```
df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than
len(df5.location.unique())
```

Out[27]:
```
241
```

In [28]:
```
df5.head(10)
```

Out[28]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000
5	Whitefield	2 BHK	1170.0	2.0	38.00	2	3247.863248
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4	7467.057101
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4	18181.818182
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3	4828.244275
9	other	6 Bedroom	1020.0	6.0	370.00	6	36274.509804

In [29]:
```
df5.to_csv("HousePrice.csv")
```

OUTLIER REMOVAL

As a data scientist when you have a conversation with your business manager (who has expertise in real estate), he will tell you that normally square ft per bedroom is 300 (i.e. 2 bhk apartment is minimum 600 sqft. If you have for example 400 sqft apartment with 2 bhk than that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping our minimum thresold per bhk to be 300 sqft

In [30]:
```
df5[df5.total_sqft/df5.bhk<300].head()
```

Out[30]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
9	other	6 Bedroom	1020.0	6.0	370.0	6	36274.509804
45	HSR Layout	8 Bedroom	600.0	9.0	200.0	8	33333.333333
58	Murugeshpalya	6 Bedroom	1407.0	4.0	150.0	6	10660.980810
68	Devarachikkanahalli	8 Bedroom	1350.0	7.0	85.0	8	6296.296296
70	other	3 Bedroom	500.0	3.0	100.0	3	20000.000000

Check above data points. We have 6 bhk apartment with 1020 sqft. Another one is 8 bhk and total sqft is 600. These are clear data errors that can be removed safely

In [31]:
```
df5.shape
```

Out[31]:
```
(13200, 7)
```

In [32]:
```
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.shape
```

Out[32]:
```
(12456, 7)
```

OUTLIER REMOVAL USING STANDARD DEVIATION AND MEAN

In [33]:
```
df6.price_per_sqft.describe()
```

Out[33]:
```
count     12456.000000
mean       6308.502826
std        4168.127339
min         267.829813
25%        4210.526316
50%        5294.117647
75%        6916.666667
max      176470.588235
Name: price_per_sqft, dtype: float64
```

Here we find that min price per sqft is 267 rs/sqft whereas max is 12000000, this shows a wide variation in property prices. We should remove outliers per location using mean and one standard deviation

In [34]:
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
df7 = remove_pps_outliers(df6)
df7.shape
```

Out[34]:
```
(10242, 7)
```

Let's check if for a given location how does the 2 BHK and 3 BHK property prices look like

In [35]:
```
df7.head(20)
```

Out[35]:

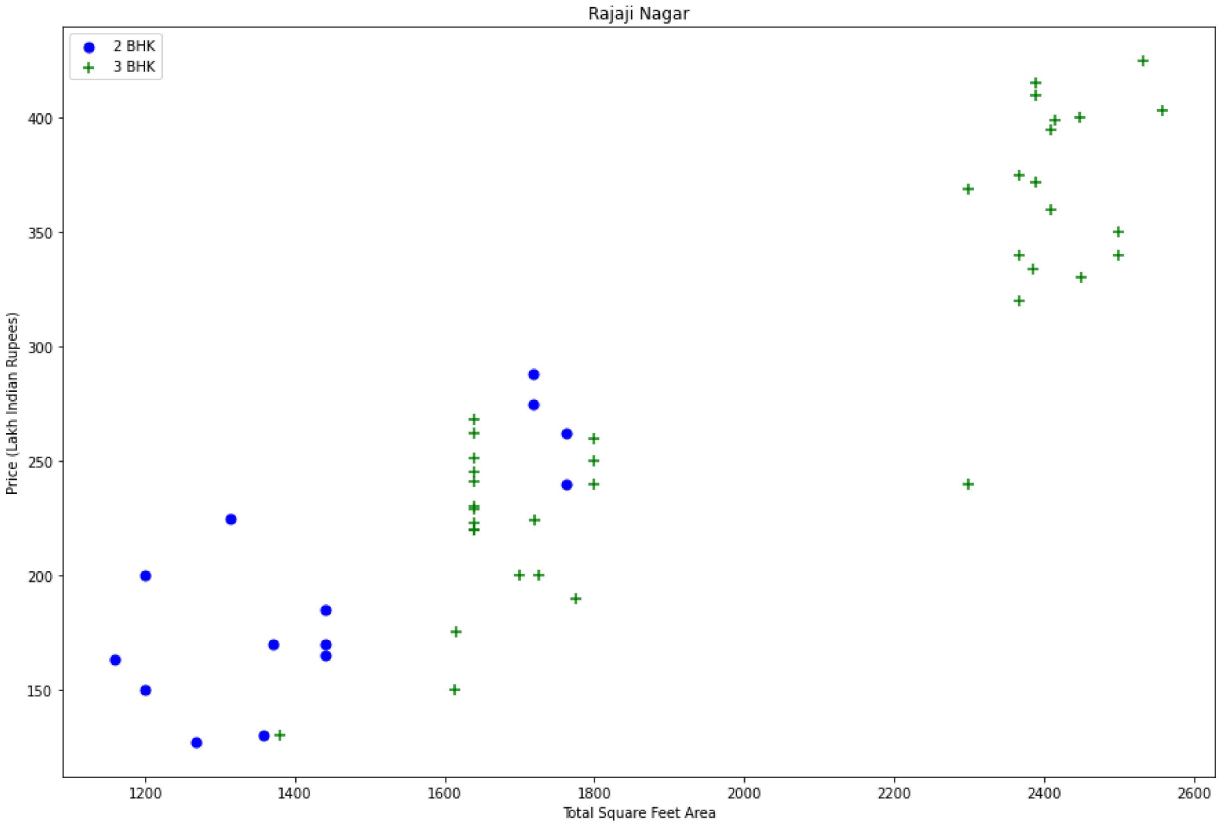
	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668
5	1st Block Jayanagar	4 BHK	2750.0	4.0	413.0	4	15018.181818
6	1st Block Jayanagar	4 BHK	2450.0	4.0	368.0	4	15020.408163
7	1st Phase JP Nagar	4 BHK	2825.0	4.0	250.0	4	8849.557522
8	1st Phase JP Nagar	3 BHK	1875.0	3.0	167.0	3	8906.666667
9	1st Phase JP Nagar	5 Bedroom	1500.0	5.0	85.0	5	5666.666667

	location	size	total_sqft	bath	price	bhk	price_per_sqft
10	1st Phase JP Nagar	3 BHK	2065.0	4.0	210.0	3	10169.491525
11	1st Phase JP Nagar	3 BHK	2024.0	3.0	157.0	3	7756.916996
12	1st Phase JP Nagar	3 BHK	2059.0	3.0	225.0	3	10927.634774
13	1st Phase JP Nagar	2 BHK	1394.0	2.0	100.0	2	7173.601148
14	1st Phase JP Nagar	2 BHK	1077.0	2.0	93.0	2	8635.097493
15	1st Phase JP Nagar	2 Bedroom	1566.0	2.0	180.0	2	11494.252874
16	1st Phase JP Nagar	1 BHK	840.0	2.0	50.0	1	5952.380952
17	1st Phase JP Nagar	3 BHK	1590.0	3.0	131.0	3	8238.993711
18	1st Phase JP Nagar	3 BHK	2180.0	3.0	210.0	3	9633.027523
19	1st Phase JP Nagar	2 BHK	1180.0	2.0	88.5	2	7500.000000

In [50]:

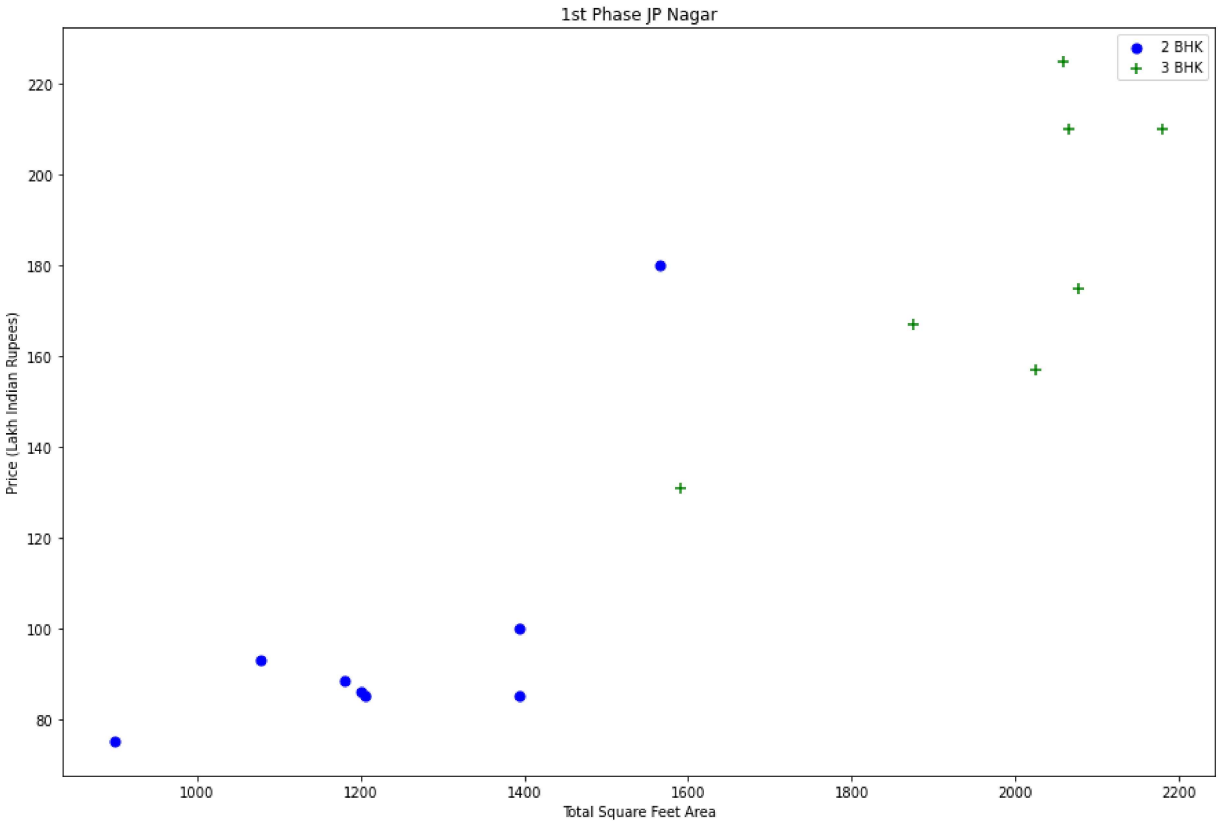
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK',
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()

plot_scatter_chart(df7,"Rajaji Nagar")
```

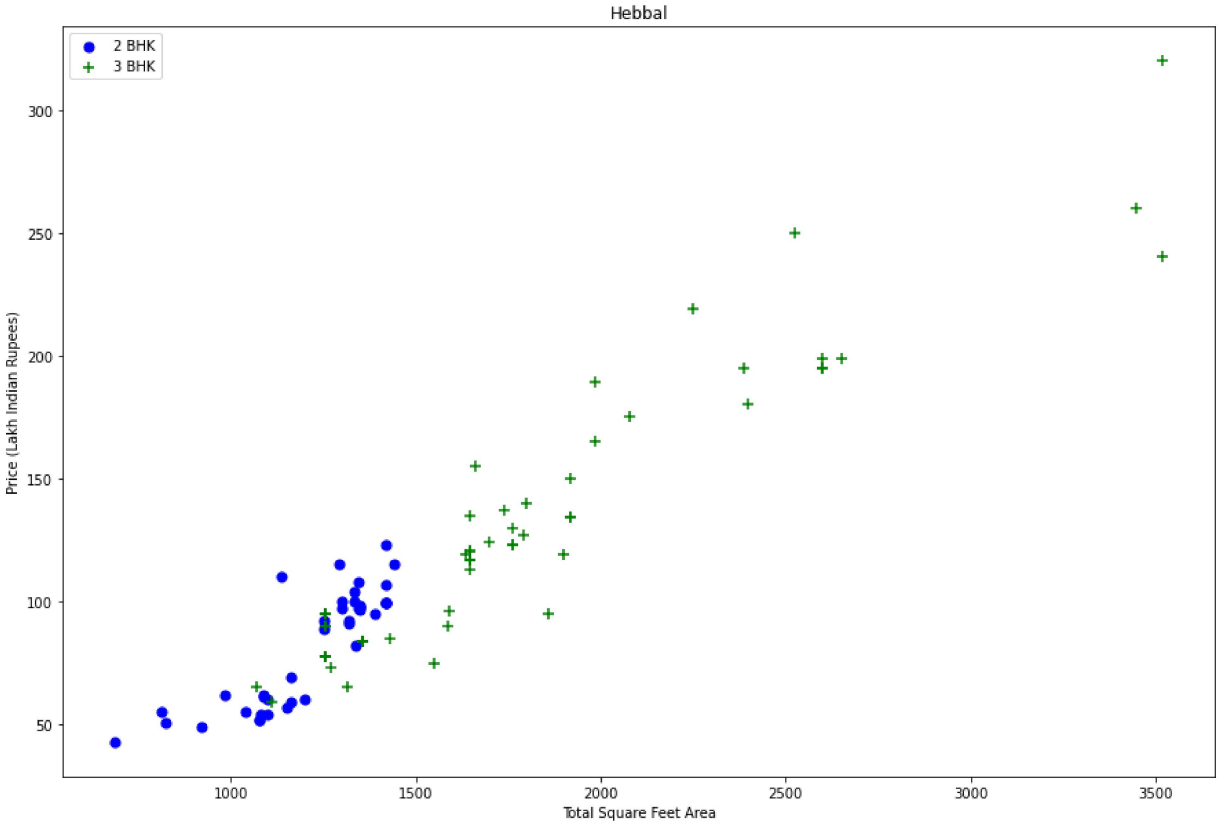


In [41]:

```
plot_scatter_chart(df7,"1st Phase JP Nagar")
```



In [42]:
```
plot_scatter_chart(df7,"Hebbal")
```



In [43]:
```
df7.shape
```

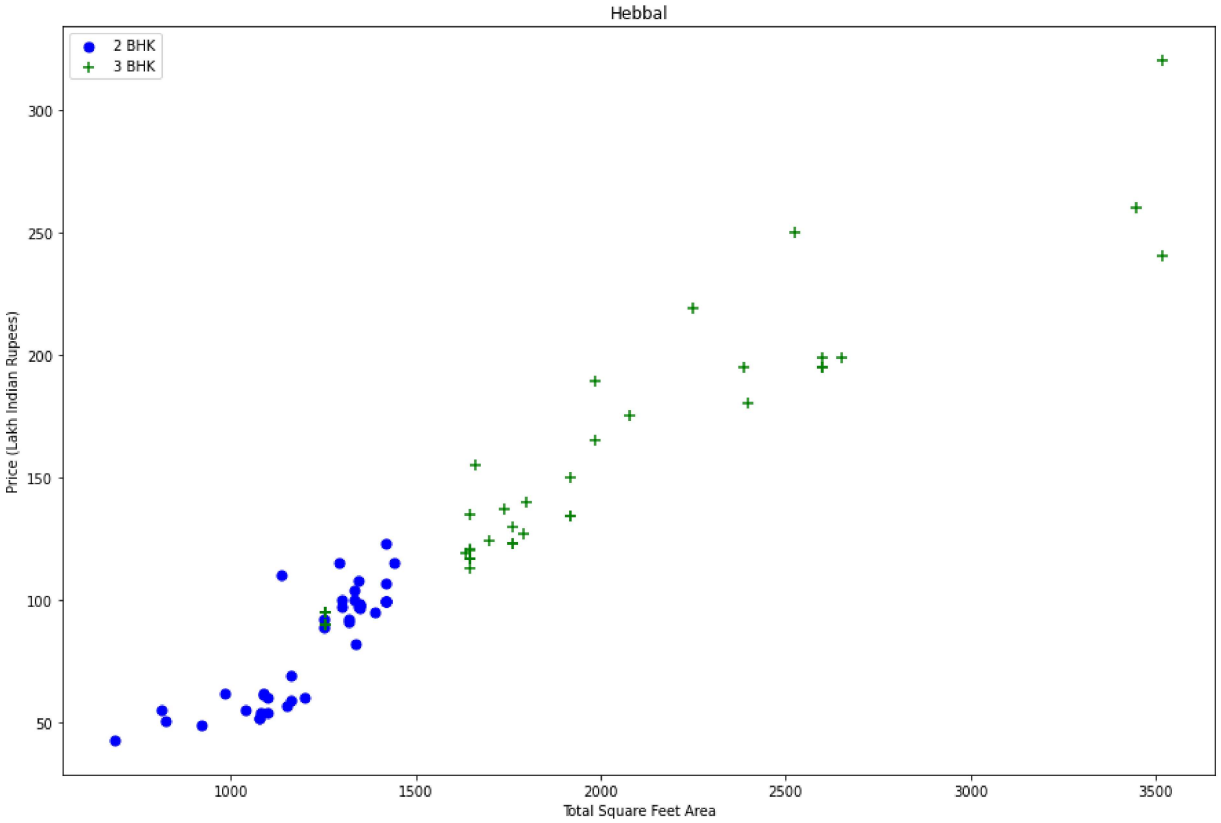
Out[43]:
```
(10242, 7)
```

In [44]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
```

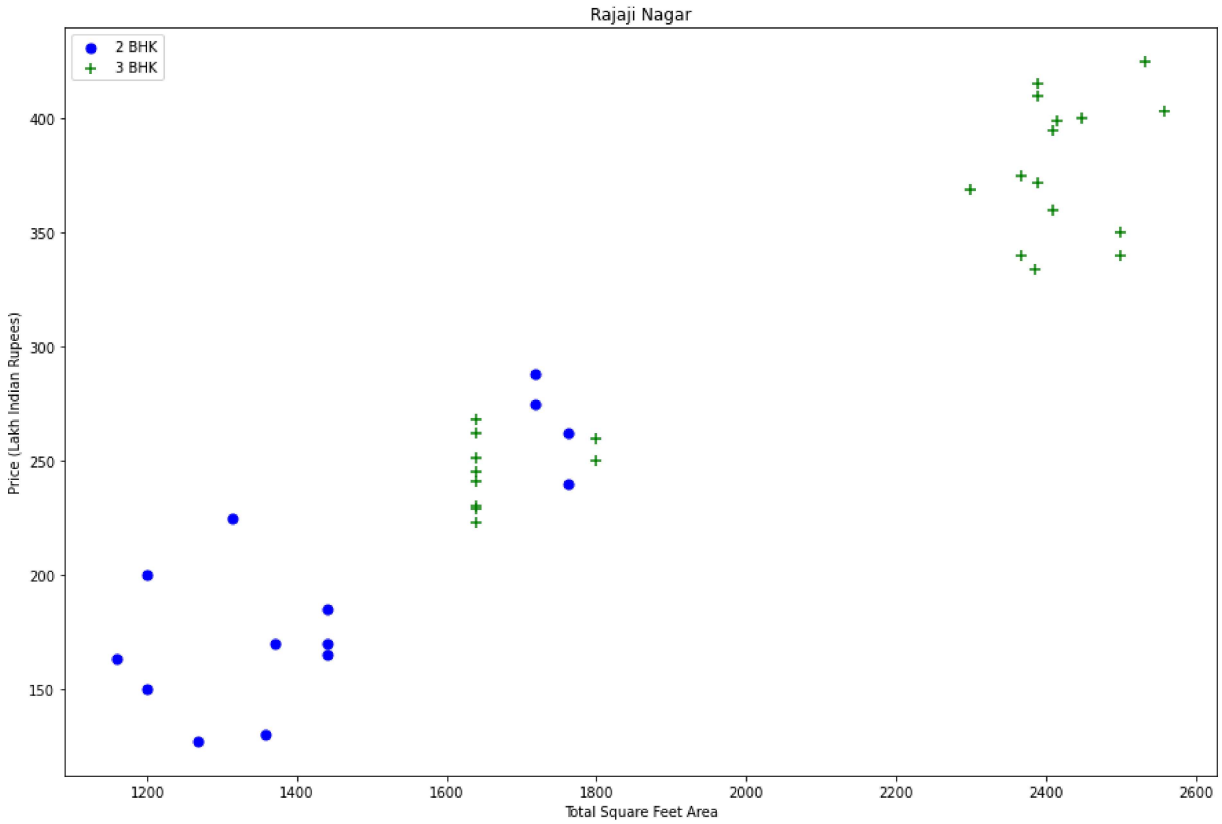
```
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                'std': np.std(bhk_df.price_per_sqft),
                'count': bhk_df.shape[0]
            }
        for bhk, bhk_df in location_df.groupby('bhk'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.price_per
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
```

Out[44]: (7317, 7)

In [46]:
```
plot_scatter_chart(df8,"Hebbal")
```



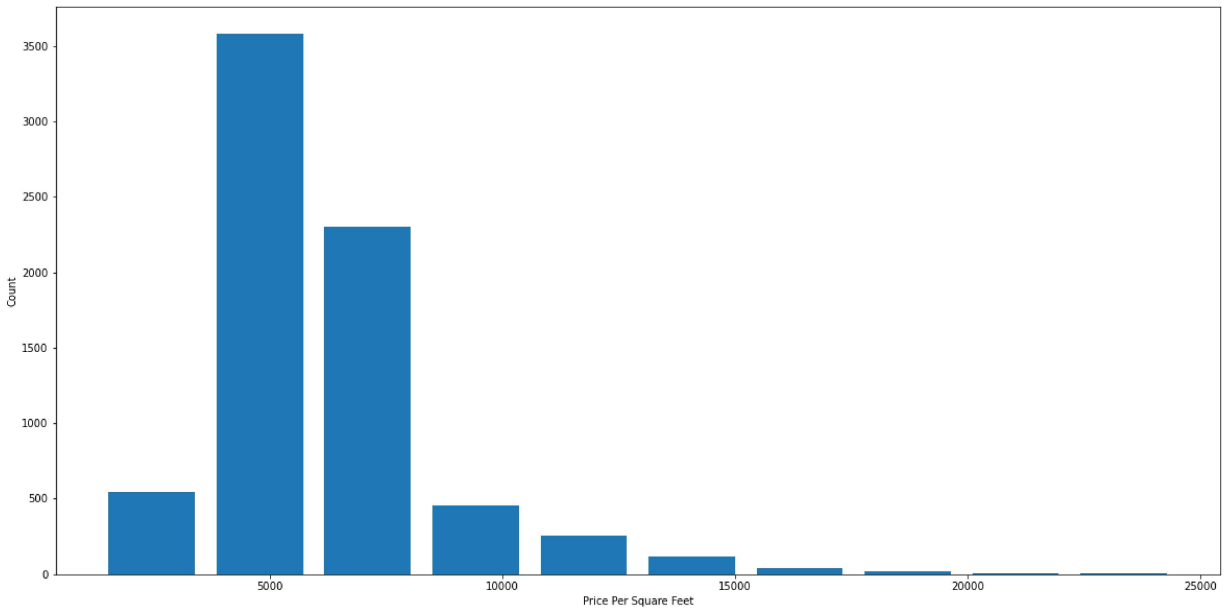
In [52]:
```
plot_scatter_chart(df8,"Rajaji Nagar")
```



Normal Distribution Verification

In [54]:
```
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
plt.hist(df8.price_per_sqft,rwidth=0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

Out[54]:
```
Text(0, 0.5, 'Count')
```



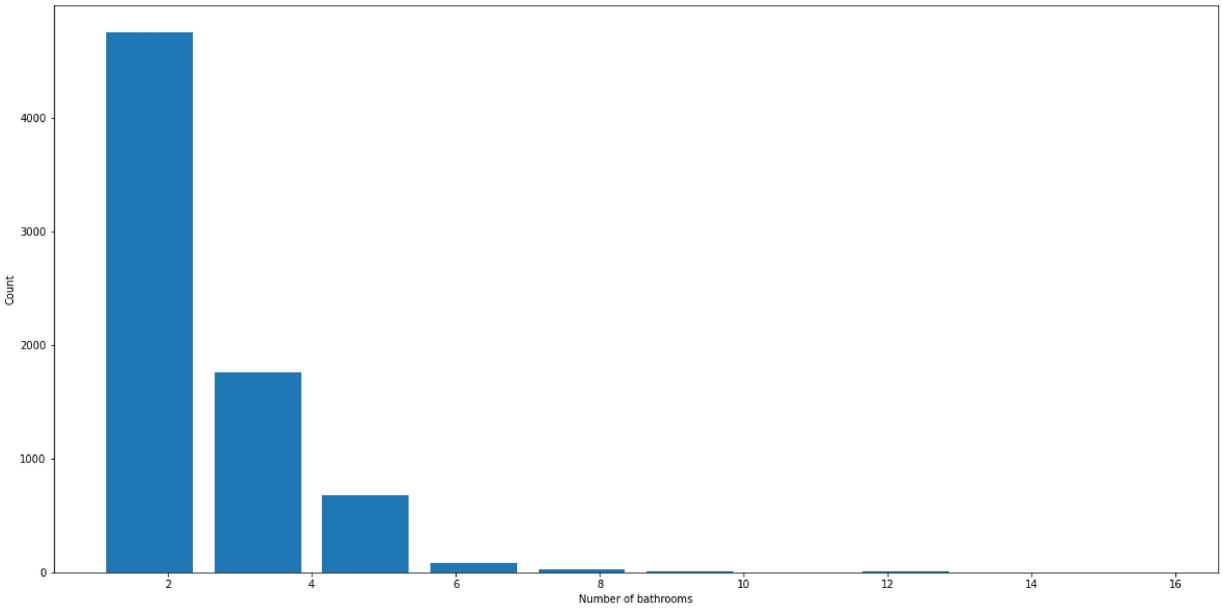
Outlier Removal Using Bathrooms Feature

In [55]:
```
df8.bath.unique()
```

Out[55]:
```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

In [56]:
```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[56]:
```
Text(0, 0.5, 'Count')
```



It is unusual to have 2 more bathrooms than number of bedrooms in a home

In [57]:
```
df8[df8.bath>df8.bhk+2]
```

Out[57]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1626	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5238	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6711	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8408	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

In [58]:
```
df8.shape
```

Out[58]:
```
(7317, 7)
```

Again the business manager has a conversation with you (i.e. a data scientist) that if you have 4 bedroom home and even if you have bathroom in all 4 rooms plus one guest bathroom, you will have total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed

In [59]:
```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

```
(7239, 7)
```

Out[59]:

In [60]:
```
df9.head()
```

Out[60]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668

In [61]:
```
df10=df9.drop(['size','price_per_sqft'],axis='columns')
df10.head()
```

Out[61]:

	location	total_sqft	bath	price	bhk
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3
3	1st Block Jayanagar	1200.0	2.0	130.0	3
4	1st Block Jayanagar	1235.0	2.0	148.0	2

Use One Hot Encoding For Location

In [62]:
```
dummies = pd.get_dummies(df10.location)
dummies.head(3)
```

Out[62]:

	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	...	Vishv
0	1	0	0	0	0	0	0	0	0	0	...	
1	1	0	0	0	0	0	0	0	0	0	...	
2	1	0	0	0	0	0	0	0	0	0	...	

3 rows × 241 columns

In [63]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[63]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	...	Vija

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	...	Vija
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0	...	
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0	...	
3	1st Block Jayanagar	1200.0	2.0	130.0	3	1	0	0	0	0	...	
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...	

5 rows × 245 columns

In [65]:
```
df12 = df11.drop('location',axis='columns')
df12.head(10)
```

Out[65]:

	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijaya
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	
2	1875.0	2.0	235.0	3	1	0	0	0	0	0	...	
3	1200.0	2.0	130.0	3	1	0	0	0	0	0	...	
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	
5	2750.0	4.0	413.0	4	1	0	0	0	0	0	...	
6	2450.0	4.0	368.0	4	1	0	0	0	0	0	...	
8	1875.0	3.0	167.0	3	0	1	0	0	0	0	...	
9	1500.0	5.0	85.0	5	0	1	0	0	0	0	...	
10	2065.0	4.0	210.0	3	0	1	0	0	0	0	...	

10 rows × 244 columns

In [66]:
```
df12.shape
```

Out[66]:
```
(7239, 244)
```

Now we can train our model

In [67]:
```
X = df12.drop(['price'],axis='columns')
```

```
X.head(3)
```

Out[67]:

	total_sqft	bath	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	...	Vijaya
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	

3 rows × 243 columns

In [68]:

```
y = df12.price
y.head(3)
```

Out[68]:

```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

In [ ]:

Find best model using GridSearchCV

In [71]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

def find_best_model_using_gridsearchcv(X,y):
    algos = {
        'linear_regression' : {
            'model': LinearRegression(),
            'params': {
                'normalize': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1,2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion' : ['mse','friedman_mse'],
                'splitter': ['best','random']
            }
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
```

```
    for algo_name, config in algos.items():
        gs =  GridSearchCV(config['model'], config['params'], cv=cv, return_train_sc
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

find_best_model_using_gridsearchcv(X,y)
```

Out[71]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726746	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.733642	{'criterion': 'mse', 'splitter': 'random'}

Based on above results we can say that LinearRegression gives the best score. Hence we will use that.

In [72]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=1
```

In [73]:

```
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[73]:

```
0.8629132245229442
```

Use K Fold cross validation to measure accuracy of our LinearRegression model

In [74]:

```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

cross_val_score(LinearRegression(), X, y, cv=cv)
```

Out[74]:

```
array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])
```

Test the model for few properties

In [75]:

```
def predict_price(location,sqft,bath,bhk):
    loc_index = np.where(X.columns==location)[0][0]

    x = np.zeros(len(X.columns))
    x[0] = sqft
    x[1] = bath
    x[2] = bhk
```

```
    if loc_index >= 0:
        x[loc_index] = 1

    return lr_clf.predict([x])[0]
```

In [76]:
```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

Out[76]:
```
83.86570258312318
```

In [77]:
```
predict_price('Indira Nagar',1000, 3, 3)
```

Out[77]:
```
195.52689759854712
```

In [78]:
```
df12.to_csv("Housedataset.csv")
```