In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
import warnings
warnings.filterwarnings("ignore")
matplotlib.rcParams["figure.figsize"] = (10,10)
```

Loading the data set from csv file

In [2]:
```
dataframe = pd.read_csv("D:\ds\Bengaluru_House_Data.csv")
dataframe.head()
```

Out[2]:

	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

Finding the shape and info of the dataframe

In [3]:
```
dataframe.info()
dataframe.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   society       7818 non-null   object 
 5   total_sqft    13320 non-null  object 
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

Out[3]:
```
(13320, 9)
```

Finding the count of free houses in different areas and analysing the data by different attributes

In [4]:
```
dataframe.groupby('area_type')['availability'].agg('count')
```

Out[4]:
```
area_type
Built-up  Area          2418
Carpet  Area              87
Plot  Area              2025
Super built-up  Area    8790
Name: availability, dtype: int64
```

In [5]:
```
dataframe.groupby('location').agg('count')
```

Out[5]:

location	area_type	availability	size	society	total_sqft	bath	balcony	price
Anekal	1	1	1	0	1	1	1	1
Banaswadi	1	1	1	1	1	1	1	1
Basavangudi	1	1	1	0	1	1	1	1
Bhoganhalli	1	1	1	1	1	1	1	1
Devarabeesana Halli	6	6	6	4	6	6	6	6
...	...	...	...	...	...	...	...	...
t.c palya	1	1	1	0	1	1	1	1
tc.palya	4	4	4	0	4	4	4	4
vinayakanagar	1	1	1	0	1	1	1	1
white field,kadugodi	1	1	1	0	1	1	0	1
whitefiled	1	1	1	0	1	1	1	1

1305 rows × 8 columns

In [6]:
```
dataframe.groupby('size').agg('count')
```

11 Bedroom	2	2	2	0	2	2	2	2
12 Bedroom	1	1	1	0	1	1	1	1
13 BHK	1	1	1	0	1	1	1	1
14 BHK	1	1	1	0	1	1	1	1
16 BHK	1	1	1	0	1	1	0	1
18 Bedroom	1	1	1	1	1	1	0	1
19 BHK	1	1	1	0	1	1	0	1
2 BHK	5199	5199	5199	3439	5199	5198	5152	5199
2 Bedroom	329	329	329	16	329	329	328	329
27 BHK	1	1	1	0	1	1	1	1
3 BHK	4310	4310	4309	3154	4310	4287	4129	4310
3 Bedroom	547	547	547	128	547	546	527	547
4 BHK	591	591	591	416	591	577	489	591

In [7]:
```
dataframe.groupby('bath').agg('count')
```

Out[7]:

bath	area_type	availability	location	size	society	total_sqft	balcony	price
1.0	788	788	788	788	428	788	786	788
2.0	6908	6908	6908	6908	4332	6908	6834	6908
3.0	3286	3286	3285	3286	2160	3286	3146	3286
4.0	1226	1226	1226	1226	509	1226	1108	1226
5.0	524	524	524	524	222	524	430	524
6.0	273	273	273	273	78	273	244	273
7.0	102	102	102	102	11	102	82	102
8.0	64	64	64	64	0	64	42	64
9.0	43	43	43	43	5	43	27	43
10.0	13	13	13	13	1	13	6	13

In [8]:
```
dataframe.groupby('balcony').agg('count')
```

Out[8]:

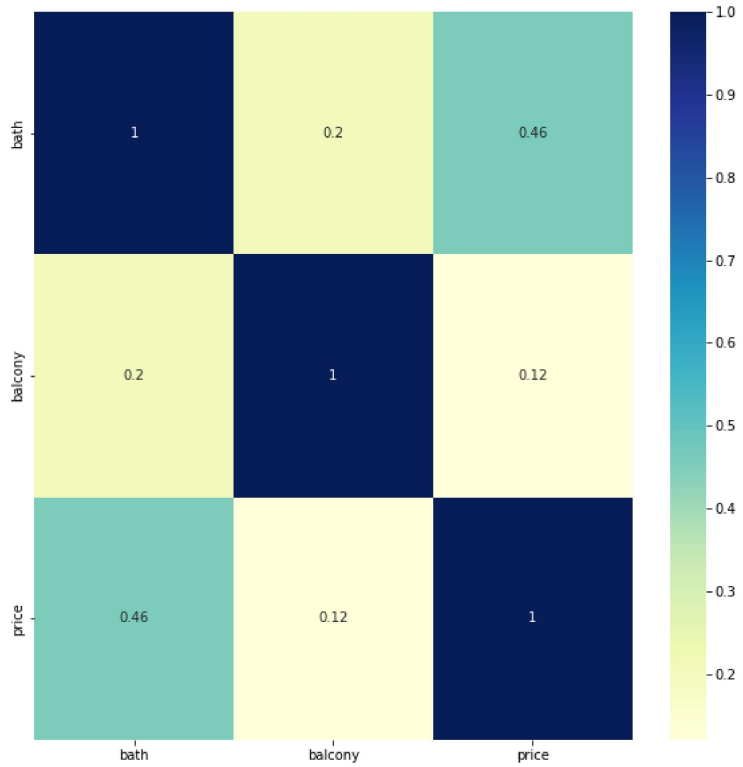
balcony	area_type	availability	location	size	society	total_sqft	bath	price
0.0	1029	1029	1029	1029	402	1029	1029	1029
1.0	4897	4897	4897	4897	2975	4897	4897	4897
2.0	5113	5113	5112	5113	3115	5113	5113	5113
3.0	1672	1672	1672	1672	1005	1672	1672	1672

In [9]:
```
print(dataframe.describe())
```

```
               bath       balcony         price
count  13247.000000  12711.000000  13320.000000
mean       2.692610      1.584376    112.565627
std        1.341458      0.817263    148.971674
min        1.000000      0.000000      8.000000
25%        2.000000      1.000000     50.000000
50%        2.000000      2.000000     72.000000
75%        3.000000      2.000000    120.000000
max       40.000000      3.000000   3600.000000
```

In [10]:
```
import seaborn as sb

dataplot = sb.heatmap(dataframe.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



In [11]:
```
dataframe.columns
```

Out[11]:
```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

In [12]:
```
dataframe1 = dataframe.drop(['area_type','availability','society','balcony'],axis='columns')
```

In [13]:
```
dataframe1.shape
```

Out[13]:
```
(13320, 5)
```

In [14]:
```
dataframe1.isnull().sum()
```

Out[14]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [15]:
```
dataframe2 = dataframe1.dropna()
dataframe2.isnull().sum()
```

Out[15]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [16]:
```
dataframe2['size'].unique()
```

Out[16]:
```
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 [17]:
```
dataframe2['bhk'] = dataframe2['size'].apply(lambda x: int(x.split(' ')[0]))
```

In [18]:
```
dataframe2.head()
```

Out[18]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00	4
2	Uttarahalli	3 BHK	1440	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00	3
4	Kothanur	2 BHK	1200	2.0	51.00	2

In [19]:
```
dataframe2['total_sqft'].unique()
```

Out[19]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

Data Pre-processing

```
Converting the data into numeric values  so we can use the data for analysing
```

In [20]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [21]:
```
dataframe2[~dataframe2['total_sqft'].apply(is_float)].head(15)
```

Out[21]:

	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
410	Kengeri	1 BHK	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	2 BHK	1195 - 1440	2.0	63.770	2
648	Arekere	9 Bedroom	4125Perch	9.0	265.000	9
661	Yelahanka	2 BHK	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	4 Bedroom	3090 - 5002	4.0	445.000	4
772	Banashankari Stage VI	2 BHK	1160 - 1195	2.0	59.935	2
775	Basavanagara	1 BHK	1000Sq. Meter	2.0	93.000	1
850	Bannerghatta Road	2 BHK	1115 - 1130	2.0	58.935	2
872	Singapura Village	2 BHK	1100Sq. Yards	2.0	45.000	2
886	Chandapura	1 BHK	520 - 645	1.0	15.135	1

Converting square feet into numerical values

In [22]:
```
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 [23]:
```
dataframe3 = dataframe2.copy()
dataframe3['total_sqft'] = dataframe3['total_sqft'].apply(convert_sqft_to_num)
dataframe3.head()
```

Out[23]:

	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
4	Kothanur	2 BHK	1200.0	2.0	51.00	2

In [24]:
```
dataframe4 = dataframe3.copy()
dataframe4['price_per_sqft'] = dataframe4['price']*100000/dataframe4['total_sqft']
dataframe4.head()
```

Out[24]:

	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

In [25]:
```
len(dataframe4['location'].unique())
```

Out[25]: 1304

In [26]:
```
dataframe4.location = dataframe4.location.apply(lambda x: x.strip())
```

In [27]:
```
location_stats = dataframe4.groupby('location')['location'].agg('count').sort_values(ascending=False)
location_stats
```

Out[27]:
```
location
Whitefield               535
Sarjapur  Road           392
Electronic City          304
Kanakpura Road           266
Thanisandra              236
                        ...
1 Giri Nagar               1
Kanakapura Road,           1
Kanakapura main  Road      1
Karnataka Shabarimala      1
whitefiled                 1
Name: location, Length: 1293, dtype: int64
```

In [28]:
```
len(location_stats[location_stats<=10])
```

Out[28]: 1052

In [29]:
```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[29]:
```
location
Basapura                 10
1st Block Koramangala    10
Gunjur Palya             10
Kalkere                  10
Sector 1 HSR Layout      10
                         ..
1 Giri Nagar              1
Kanakapura Road,          1
Kanakapura main  Road     1
Karnataka Shabarimala     1
whitefiled                1
Name: location, Length: 1052, dtype: int64
```

In [30]:
```
dataframe4.location = dataframe4.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
```

In [31]:
```
dataframe4.head(25)
```

Out[31]:

	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
10	Whitefield	3 BHK	1800.0	2.0	70.00	3	3888.888889
11	Whitefield	4 Bedroom	2785.0	5.0	295.00	4	10592.459605
12	7th Phase JP Nagar	2 BHK	1000.0	2.0	38.00	2	3800.000000
13	Gottigere	2 BHK	1100.0	2.0	40.00	2	3636.363636
14	Sarjapur	3 Bedroom	2250.0	3.0	148.00	3	6577.777778
15	Mysore Road	2 BHK	1175.0	2.0	73.50	2	6255.319149
16	Bisuvanahalli	3 BHK	1180.0	3.0	48.00	3	4067.796610
17	Raja Rajeshwari Nagar	3 BHK	1540.0	3.0	60.00	3	3896.103896
18	other	3 BHK	2770.0	4.0	290.00	3	10469.314079
19	other	2 BHK	1100.0	2.0	48.00	2	4363.636364
20	Kengeri	1 BHK	600.0	1.0	15.00	1	2500.000000
21	Binny Pete	3 BHK	1755.0	3.0	122.00	3	6951.566952
22	Thanisandra	4 Bedroom	2800.0	5.0	380.00	4	13571.428571
23	Bellandur	3 BHK	1767.0	3.0	103.00	3	5829.088851
24	Thanisandra	1 RK	510.0	1.0	25.25	1	4950.980392

In [32]:
```
len(dataframe4.location.unique())
```

Out[32]:
```
242
```

In [33]:
```
dataframe4.shape
```

Out[33]:
```
(13246, 7)
```

In [34]:
```
dataframe5 = dataframe4[~(dataframe4.total_sqft/dataframe4.bhk<300)]
dataframe5.shape
```

Out[34]:
```
(12502, 7)
```

In [35]:
```
dataframe5.price_per_sqft.describe()
```

Out[35]:
```
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
```

Removing the Out liers

In [36]:
```
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+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
```

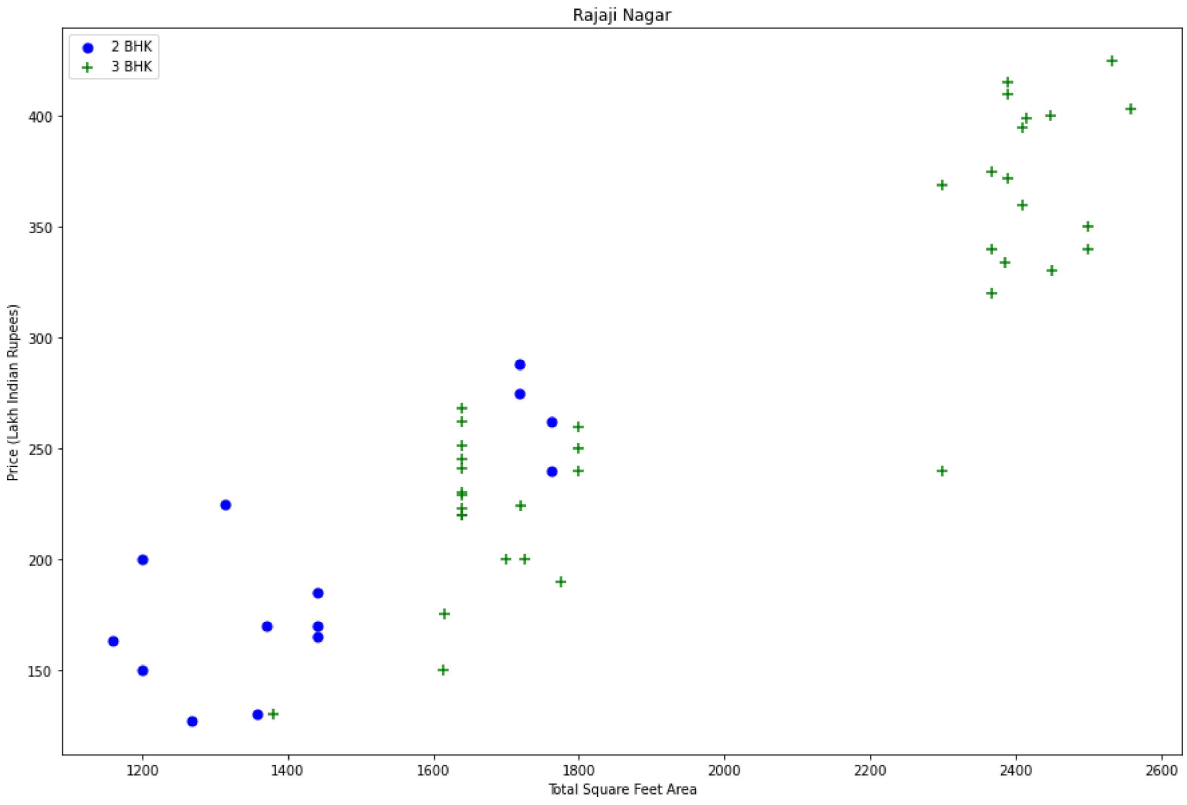
In [37]:
```
dataframe6 = remove_pps_outliers(dataframe5)
dataframe6.shape
```

Out[37]:
```
(10241, 7)
```

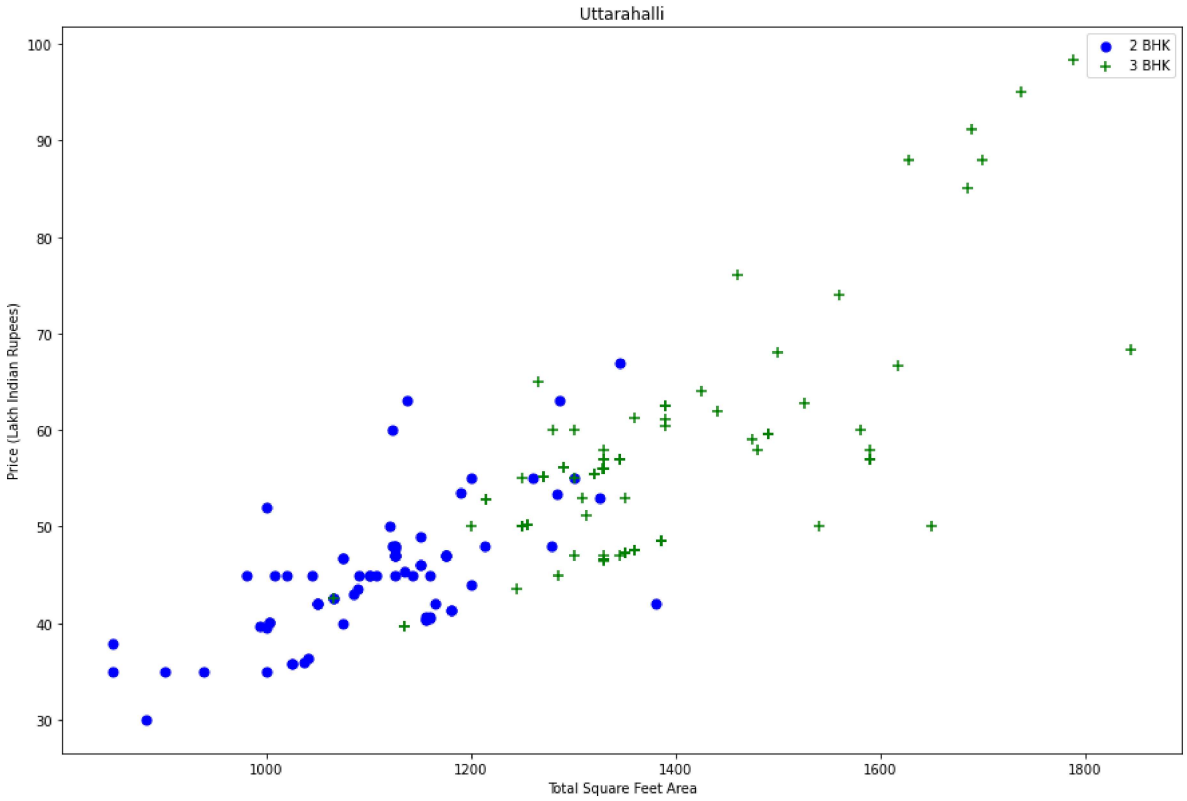
In [38]:
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=50)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()
```

Finding the price of houses based on total square feet based on different locations

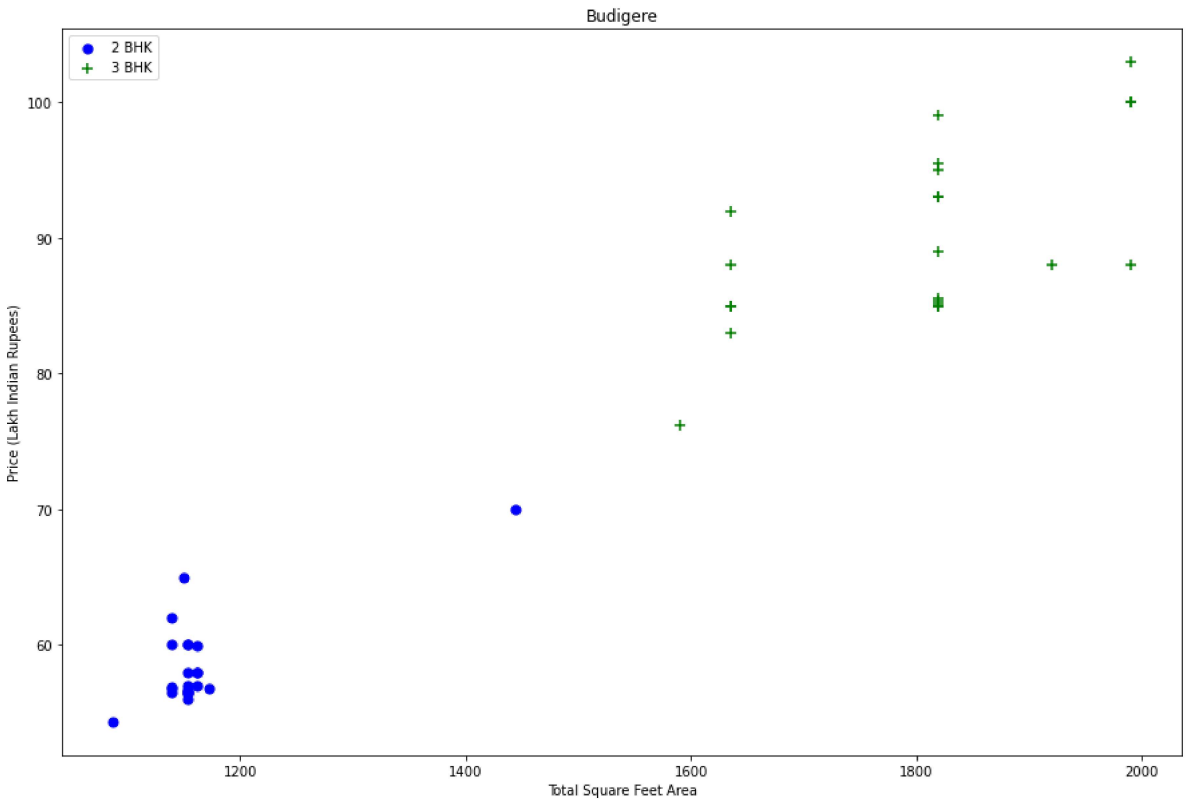
In [39]:
```
plot_scatter_chart(dataframe6,"Rajaji Nagar")
```



In [40]:
```
plot_scatter_chart(dataframe6,"Uttarahalli")
```



In [41]:
```
plot_scatter_chart(dataframe6,"Budigere")
```



We can see that the highest cost is from Rajaji Nagar which is more than 400 lakh rupees this might be due to availability of malls ,hotels, restaurents, hospitals around the area i.e it is a developed area

```
# We should also remove properties where for same location, the price of (for example) 3 bedroom apartment is less than 2
bedroom apartment (with same square ft area). What we will do is for a given location, we will build a dictionary of stats per
bhk, i.e.
```

```
{
    '1' : {
        'mean': 4000,
        'std: 2000,
        'count': 34
    },
    '2' : {
        'mean': 4300,
        'std: 2300,
        'count': 22
    },
}
```

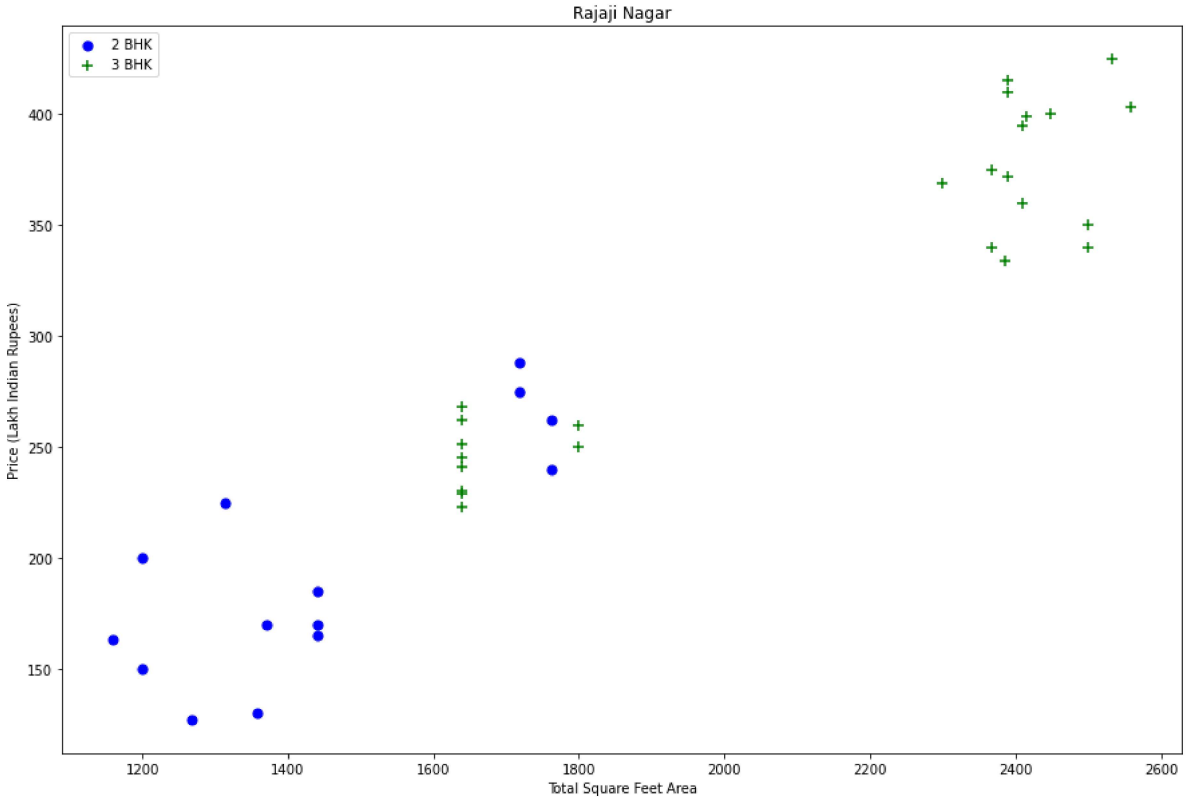
Now we can remove those 2 BHK apartments whose price_per_sqft is less than mean price_per_sqft of 1 BHK apartment

In [42]:
```
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_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices,axis='index')
```

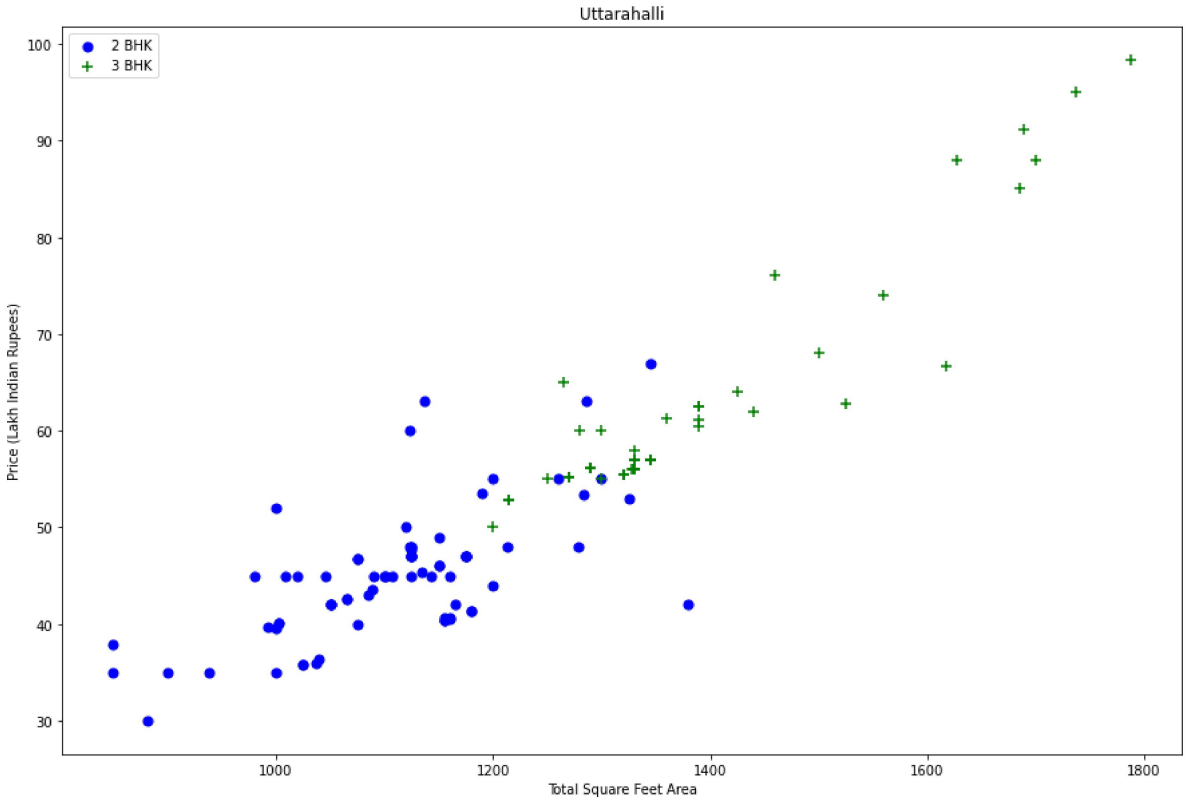
In [43]:
```
dataframe7 = remove_bhk_outliers(dataframe6)
dataframe7.shape
```

Out[43]: (7329, 7)

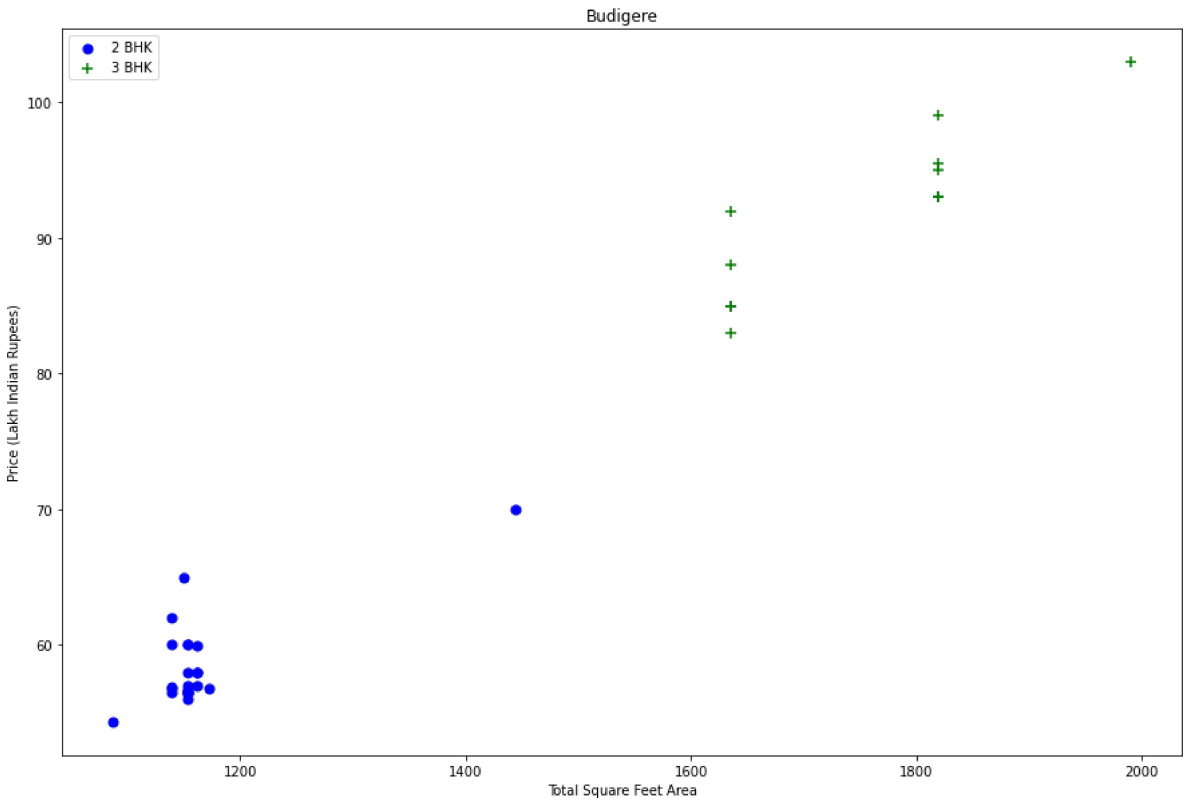
In [44]:
```
plot_scatter_chart(dataframe7,"Rajaji Nagar")
```



In [45]:
```
plot_scatter_chart(dataframe7,"Uttarahalli")
```

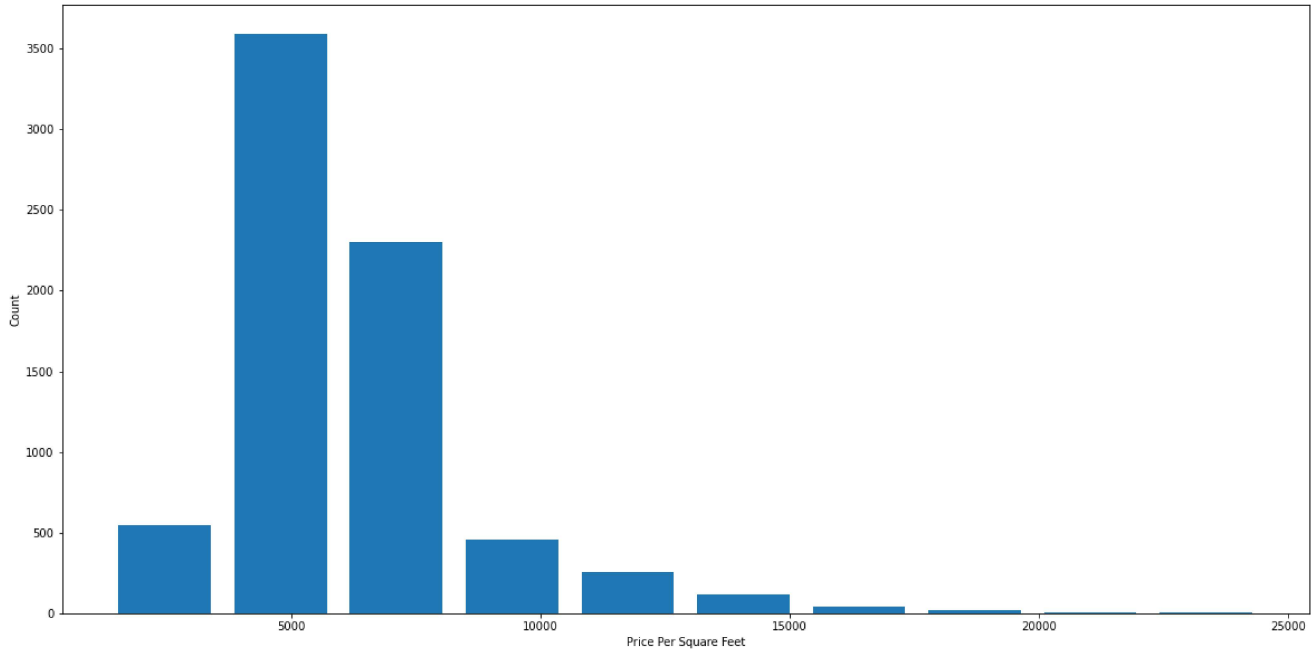


In [46]:
```
plot_scatter_chart(dataframe7,"Budigere")
```



In [47]:
```
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
plt.hist(dataframe7.price_per_sqft,rwidth=0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

Out[47]:
```
Text(0, 0.5, 'Count')
```

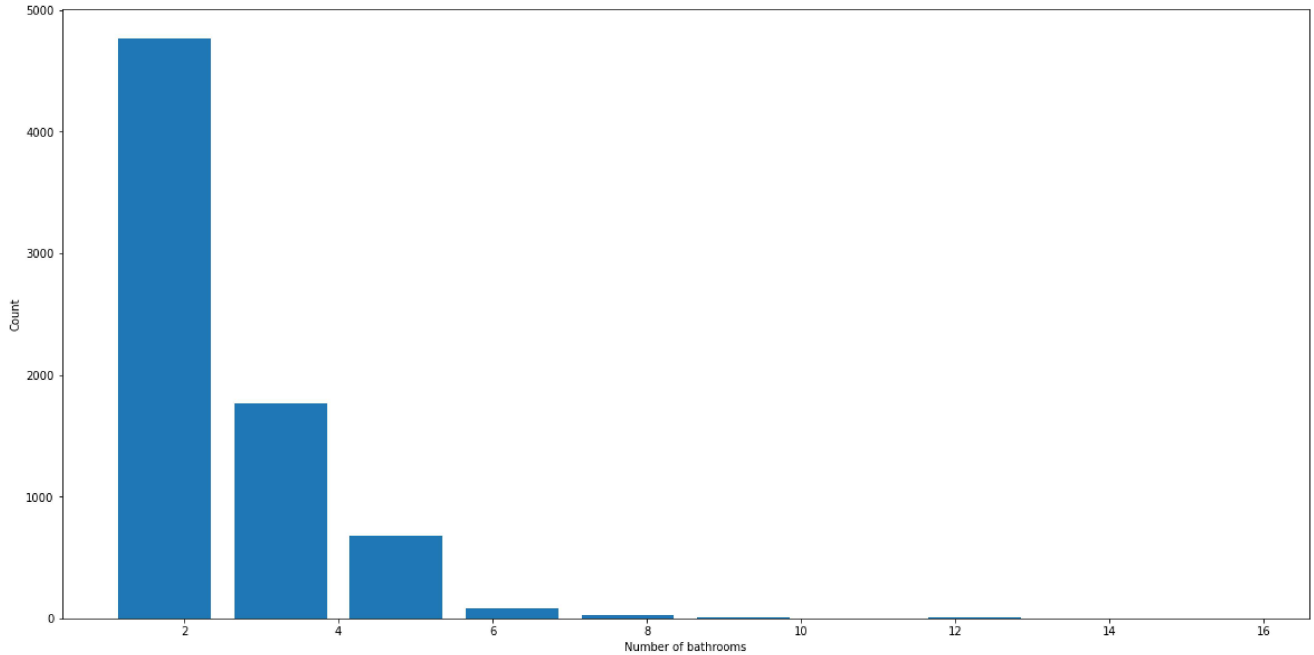


In [48]:
```
dataframe7.bath.unique()
```

Out[48]:
```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

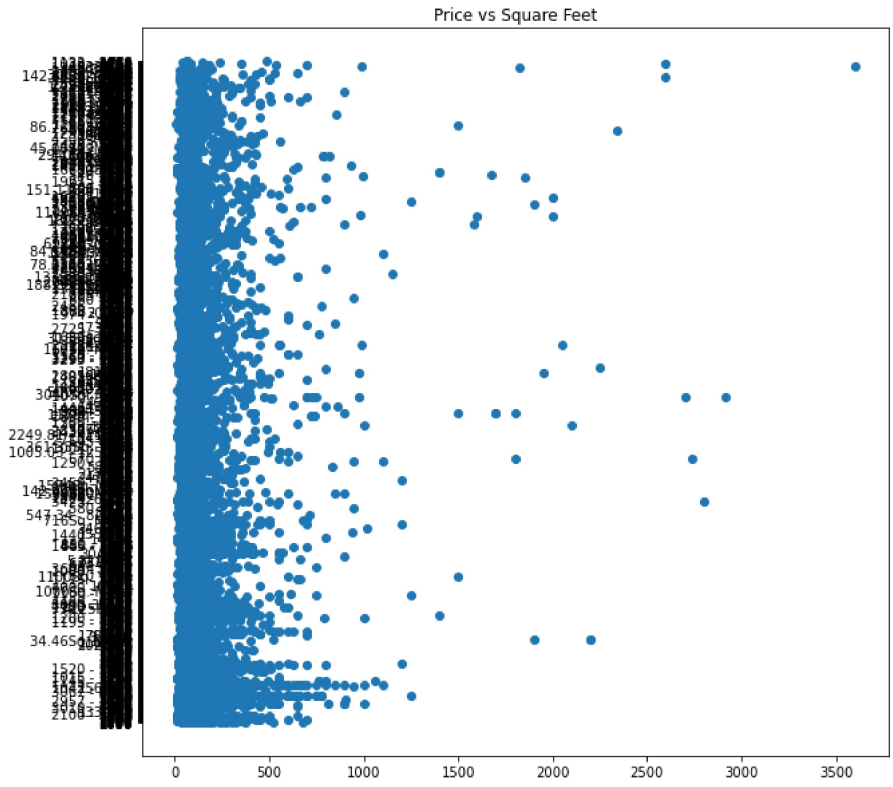
In [49]:
```
plt.hist(dataframe7.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[49]:
```
Text(0, 0.5, 'Count')
```



In [50]:
```
plt.figure(figsize=(10,10))
plt.scatter(dataframe.price,dataframe.total_sqft)
plt.title("Price vs Square Feet")
```

Out[50]:
```
Text(0.5, 1.0, 'Price vs Square Feet')
```



In [51]:
```
dataframe7[dataframe7.bath>10]
```

Out[51]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8486	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8575	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9308	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9639	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

In [52]:
```
dataframe7[dataframe7.bath>dataframe7.bhk+2]
```

Out[52]:

	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
8411	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

In [53]:
```
dataframe8 = dataframe7[dataframe7.bath<dataframe7.bhk+2]
dataframe8.shape
```

Out[53]:
```
(7251, 7)
```

In [54]:
```
dataframe9 = dataframe8.drop(['size','price_per_sqft'],axis='columns')
dataframe9.head()
```

Out[54]:

	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

In [55]:
```
len(dataframe9.location.unique())
```

Out[55]: 242

In [56]:
```
dummies = pd.get_dummies(dataframe9.location)
dummies.head()
```

Out[56]:

	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	...	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenahal
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 242 columns

In [57]:
```
dataframe10 = pd.concat([dataframe9,dummies.drop('other',axis='columns')],axis='columns')
dataframe10.head()
```

Out[57]:

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

5 rows × 246 columns

In [58]:
```
dataframe11 = dataframe10.drop('location',axis='columns')
dataframe11.head()
```

Out[58]:

	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	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Y
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	0	0	0	0	
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	0	0	0	0	
2	1875.0	2.0	235.0	3	1	0	0	0	0	0	...	0	0	0	0	0	
3	1200.0	2.0	130.0	3	1	0	0	0	0	0	...	0	0	0	0	0	
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 245 columns

In [59]:
```
dataframe11.shape
```

Out[59]: (7251, 245)

In [60]:
```
X = dataframe11.drop('price', axis='columns')
X.head()
```

Out[60]:

	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	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	0	0	0	0
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0
3	1200.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0
4	1235.0	2.0	2	1	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 244 columns

In [61]:
```
y = dataframe11.price
y.head()
```

Out[61]:
```
0    428.0
1    194.0
2    235.0
3    130.0
4    148.0
Name: price, dtype: float64
```

In [62]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=10)
```

```
we are finding the best model which gives the highest accuracy by using this function
```

In [65]:
```
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.pipeline import make_pipeline


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_score=False)
        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[65]:

	model	best_score	best_params
0	linear_regression	0.818354	{'normalize': True}
1	lasso	0.687470	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.724016	{'criterion': 'mse', 'splitter': 'best'}

we got the best accuracy by using linear regression so we used linear regression for the process

In [63]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression()
model_lr.fit(X_train,y_train)
model_lr.score(X_test,y_test)
```

Out[63]: 0.8806035205408391

In [64]:
```
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[64]: array([0.82430186, 0.77166234, 0.85089567, 0.80837764, 0.83653286])

In [66]:
```
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 model_lr.predict([x])[0]
```

In [67]:
```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

Out[67]: 85.65663590128247

In [68]:
```
predict_price('1st Phase JP Nagar',1000, 4, 2)
```

Out[68]: 89.9712199742374

In [ ]: