Real Estate Price Prediction

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)
```

In [2]:
```
df1=pd.read_csv(r"C:\Users\maram\Downloads\Bengaluru_House_Data.csv")
df1.head(2)
```

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

In [3]:
```
df1.shape
```

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

In [4]:
```
df1.groupby('area_type')['area_type'].agg('count')
```

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

In [5]:
```
#df1['area_type'].value_counts()
```

In [6]:
```
df1.columns.values
```

Out[6]:
```
array(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'], dtype=object)
```

In [7]:
```
df1.drop(['area_type','availability','society','balcony'],axis=1,inplace=True)
```

In [8]:
```
df1.head(3)
```

Out[8]:

	location	size	total_sqft	bath	price
0	Electronic City Phase II	2 BHK	1056	2.0	39.07
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00
2	Uttarahalli	3 BHK	1440	2.0	62.00

In [9]:
```
df1.isnull().sum()
```

Out[9]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [10]:
```
df1.dropna(inplace=True)
```

In [11]:
```
df1.isnull().sum()
```

Out[11]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [12]:
```
df1['size'].unique()
```

Out[12]:
```
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 [13]:
```
df1['bhk']=df1['size'].apply(lambda x :int(x.split(' ')[0]))
# The lambda function splits each element (which is assumed to be a string) by a space
```

In [14]:
```
df1['bhk'].unique()
```

Out[14]:
```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18], dtype=int64)
```

In [15]:
```
df1.head()
```

Out[15]:

	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 [16]:
```
#df1['bhk1'] = df1['size'].str.replace(r'\D', '', regex=True)
```

In [17]:
```
#df1['bhk1'].unique()
```

In [18]:
```
df1[df1['bhk']>20]
```

Out[18]:

	location	size	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	27 BHK	8000	27.0	230.0	27
4684	Munnekollal	43 Bedroom	2400	40.0	660.0	43

In [19]:
```
# you can not have 43 bedrooms while u have total sqft = 2400
```

In [20]:
```
df1.total_sqft.unique()
```

Out[20]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [21]:
```
def is_float(x):
    try: #Im trying to convert x to float
        float(x)
    except: #if its on form of interval or any return false
        return False
    return True
```

In [22]:
```
df1[~df1['total_sqft'].apply(is_float)].head(10)
```

Out[22]:

	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

In [23]:
```
def tokens_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 [24]:
```
tokens_sqft_to_num('2011')
```

Out[24]:
```
2011.0
```

In [25]:
```
tokens_sqft_to_num('235 - 243')
```

Out[25]:
```
356.5
```

In [26]:
```
df2=df1.copy()
```

In [27]:
```
df2['total_sqft']=df1['total_sqft'].apply(tokens_sqft_to_num)
```

In [28]:
```
df2.head()
```

Out[28]:

	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

Feature Engineering

In [29]:
```
df3=df2.copy()
```

In [30]:
```
df3['price_per_sqft']=df3['price']*100000/df3['total_sqft']
df3.head(3)
```

Out[30]:

	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

In [31]:
```
len(df3.location.unique())
```

Out[31]:
```
1304
```

Dimensionality reduction

In [32]:
```
df3.location=df3.location.apply(lambda x:x.strip())
```

In [33]:
```
location_stats=df3.groupby('location')['location'].agg('count').sort_values(ascending=
location_stats.head(10)
```

Out[33]:
```
location
Whitefield               535
Sarjapur  Road           392
Electronic City          304
Kanakpura Road           266
Thanisandra              236
Yelahanka                210
Uttarahalli              186
Hebbal                   176
Marathahalli             175
Raja Rajeshwari Nagar    171
Name: location, dtype: int64
```

In [34]:
```
len(location_stats[location_stats<=10])
```

Out[34]:
```
1052
```

In [35]:
```
location_stats_less_than_10=location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[35]:
```
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 [36]:
```
len(df3.location.unique())
```

Out[36]:
```
1293
```

In [37]:
```
df3.location=df3.location.apply(lambda x: 'other'if x in location_stats_less_than_10 e
len(df3.location.unique())
```

Out[37]:
```
242
```

Outlier detuction/Removal

In [38]:
```
df3.columns.values
```

Out[38]:
```
array(['location', 'size', 'total_sqft', 'bath', 'price', 'bhk',
       'price_per_sqft'], dtype=object)
```

In [39]:
```
df3[df3.total_sqft/df3.bhk<300].head()
```

Out[39]:

	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

In [40]:
```
df3.shape
```

Out[40]:
```
(13246, 7)
```

In [41]:
```
df4=df3[~(df3.total_sqft/df3.bhk<300)]
```

In [42]:
```
df4.shape
```

Out[42]:
```
(12502, 7)
```

In [43]:
```
df4
```

Out[43]:

	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
...	...	...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453.0	4.0	231.00	5	6689.834926
13316	other	4 BHK	3600.0	5.0	400.00	4	11111.111111
13317	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2	5258.545136
13318	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4	10407.336319
13319	Doddathoguru	1 BHK	550.0	1.0	17.00	1	3090.909091

12502 rows × 7 columns

In [44]:
```
df4.price_per_sqft.describe()
```

Out[44]:
```
count     12456.000000
mean       6290.299983
std        4175.619211
min         267.829813
25%        4186.725844
50%        5281.690141
75%        6904.652015
max      176470.588235
Name: price_per_sqft, dtype: float64
```

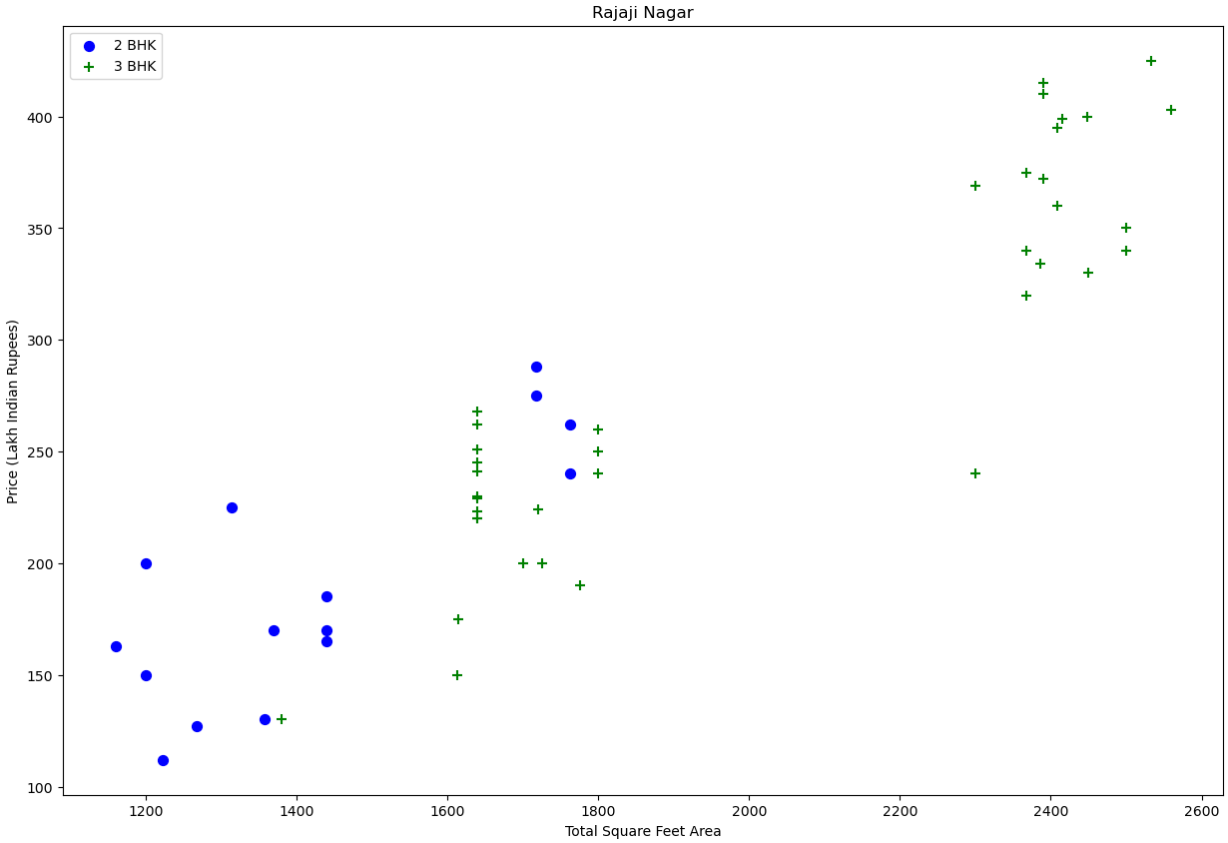
Check above data points. We have 6 bhk apartment with 1020 sqft. Another one is 8 bhk and total sqft is 600. These are clear data errors that can be removed safely

In [45]:
```
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+s
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
df5 = remove_pps_outliers(df4)
df5.shape
```

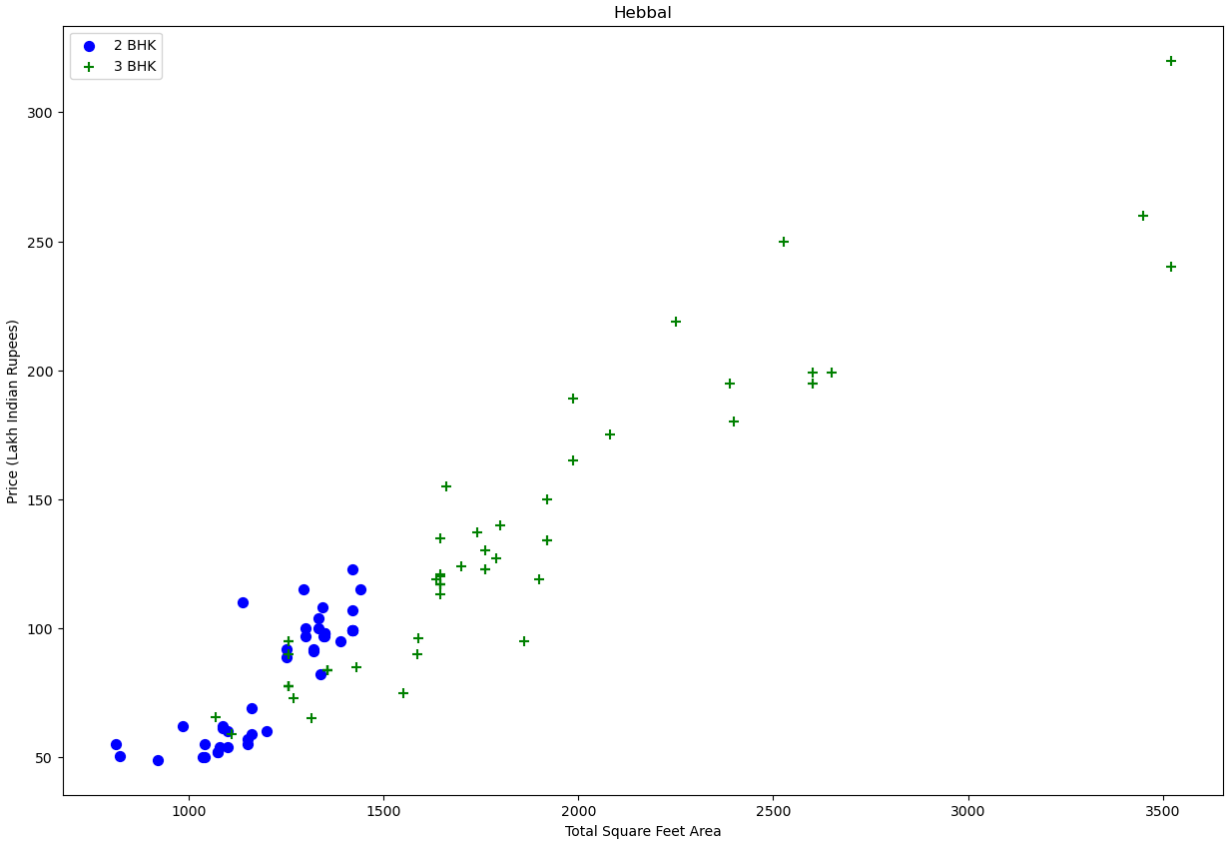
Out[45]:
```
(10231, 7)
```

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

In [46]:
```
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=
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()

plot_scatter_chart(df5,"Rajaji Nagar")
```



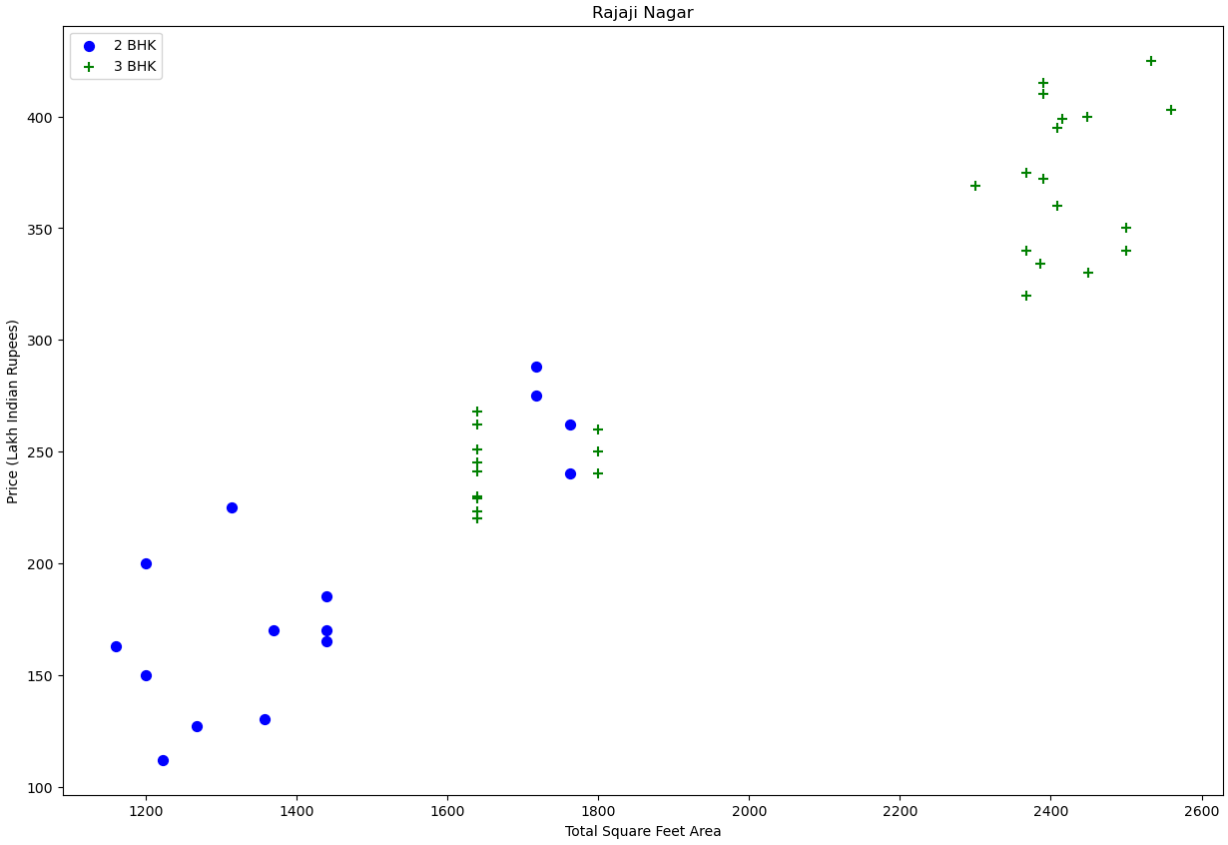
In [47]:
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()

plot_scatter_chart(df5,"Hebbal")
```



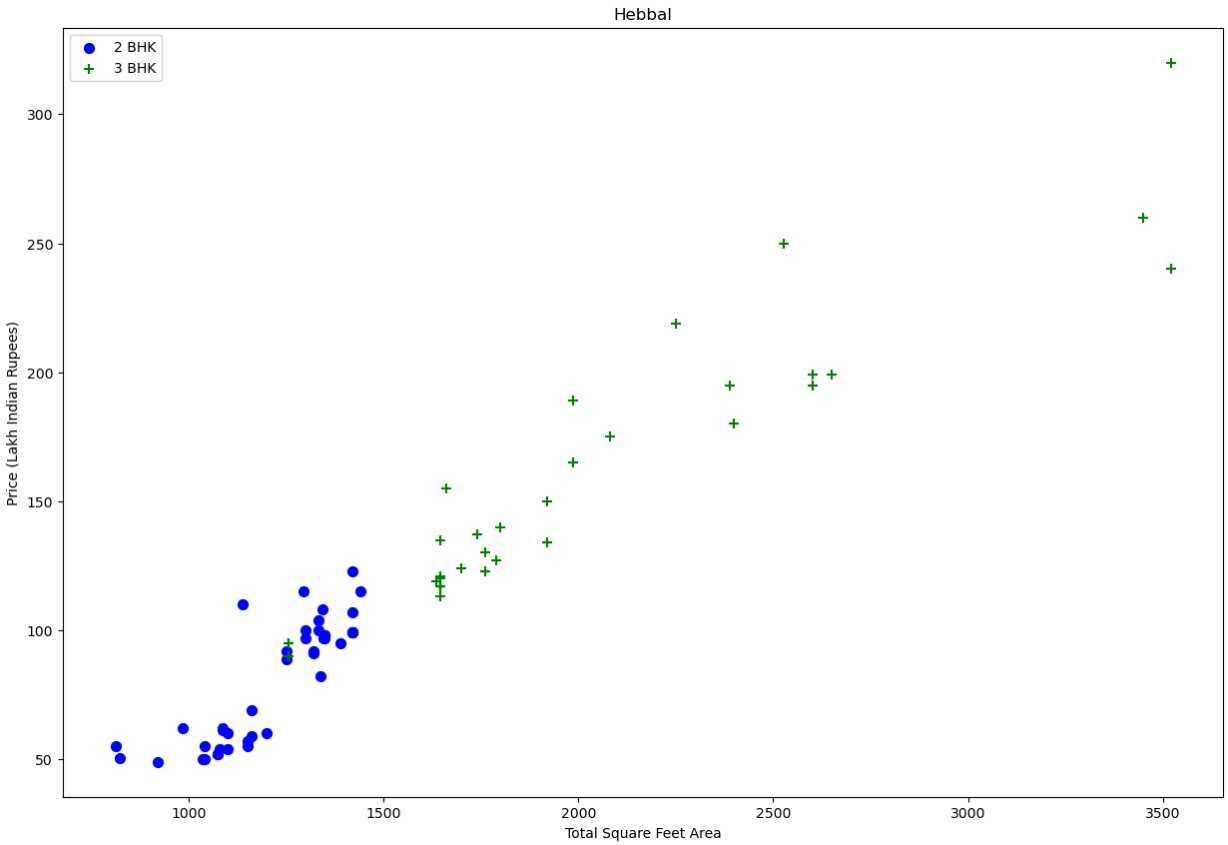
In [48]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                '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_s
    return df.drop(exclude_indices,axis='index')
df6 = remove_bhk_outliers(df5)
# df8 = df7.copy()
df6.shape
```

Out[48]:
```
(7326, 7)
```

In [49]:
```
plot_scatter_chart(df6,"Rajaji Nagar")
```



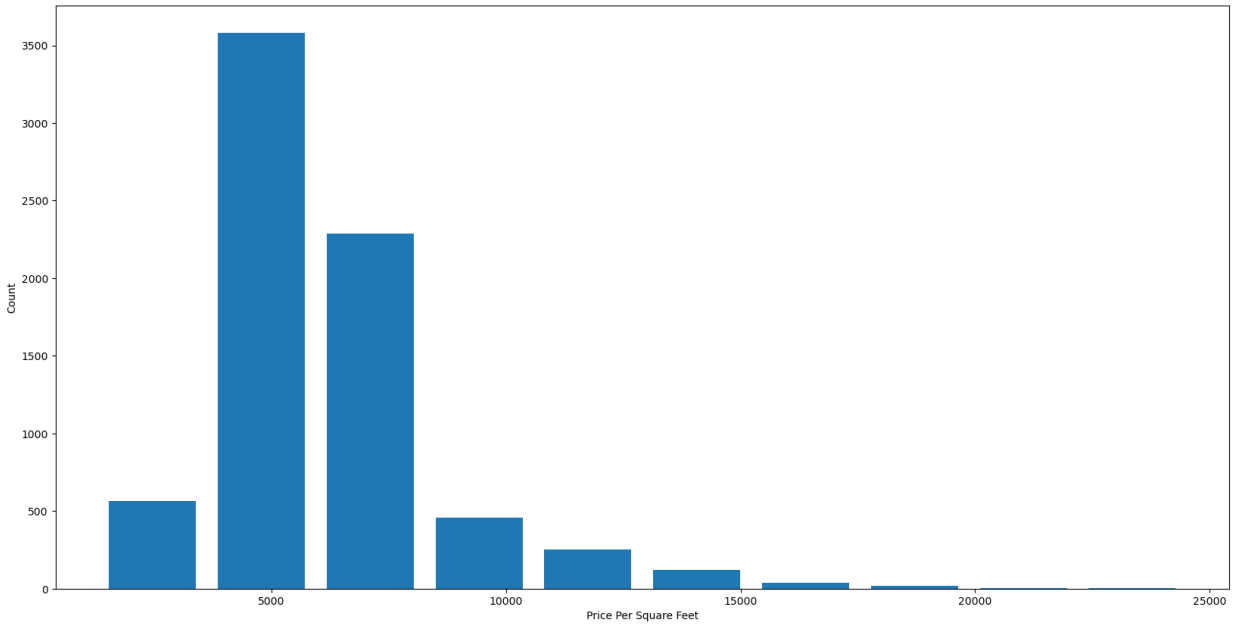
In [50]:
```
plot_scatter_chart(df6,"Hebbal")
```



In [51]:
```
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
plt.hist(df6.price_per_sqft,rwidth=0.8)
```

```
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

Out[51]:
```
Text(0, 0.5, 'Count')
```

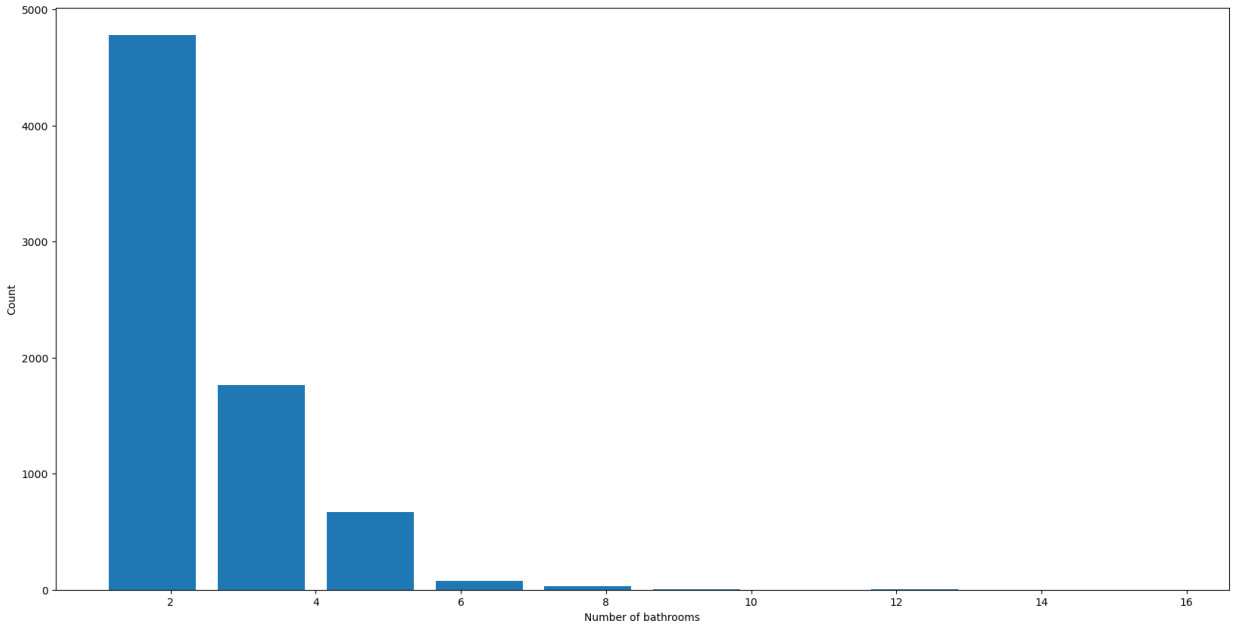


In [52]:
```
df6.bath.unique()
```

Out[52]:
```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

In [53]:
```
plt.hist(df6.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[53]:
```
Text(0, 0.5, 'Count')
```



In [54]:
```
df6[df6.bath>10]
```

Out[54]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
5273	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8476	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8565	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9298	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9629	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

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

In [55]:
```
df6[df6.bath>df6.bhk+2]
```

Out[55]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1625	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5234	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6705	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8401	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

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 [56]:
```
df7 = df6[df6.bath<df6.bhk+2]
df7.shape
```

Out[56]:
```
(7252, 7)
```

Hot encoding

In [57]:
```
df8 = df7.drop(['size','price_per_sqft'],axis='columns')
df8.head(3)
```

Out[57]:

	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

In [58]:
```
dummies = pd.get_dummies(df8.location)
dummies.head(3)
```

Out[58]:

	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	...	Vishves
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 × 242 columns

In [59]:
```
df11 = pd.concat([df8,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[59]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	...	Vijaya
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 × 246 columns

In [60]:
```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

Out[60]:

	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	...	Vijayana
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 rows × 245 columns

Build a Model Now...

In [61]:
```
df12.shape
```

Out[61]:
```
(7252, 245)
```

In [62]:
```
X = df12.drop(['price'],axis='columns')
X.head(3)
```

Out[62]:

	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	...	Vijayana
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 × 244 columns

In [63]:
```
y = df12.price
y.head(3)
```

Out[63]:
```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

In [64]:
```
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=10)
```

In [65]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[65]:
```
0.8515492485425586
```

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

In [66]:
```
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[66]:
```
array([0.83419256, 0.84066871, 0.85637292, 0.84326517, 0.84613103])
```

In [67]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
```

```
def find_best_model_using_gridsearchcv(X, y):
    algos = {
        'linear_regression': {
            'model': LinearRegression(),
            'params': {
                'fit_intercept': [True, False],
                'n_jobs': [-1]  # If you want to specify other parameters for LinearRe
            }
        },
        '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
        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)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:37
8: FitFailedWarning:
10 fits failed out of a total of 20.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score
='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "C:\ProgramData\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\ProgramData\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 12
47, in fit
    super().fit(
  File "C:\ProgramData\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 17
7, in fit
    self._validate_params()
  File "C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py", line 581, in _va
lidate_params
    validate_parameter_constraints(
  File "C:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\_param_validation.p
y", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of D
ecisionTreeRegressor must be a str among {'poisson', 'squared_error', 'absolute_erro
r', 'friedman_mse'}. Got 'mse' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\ProgramData\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:952: Us
erWarning: One or more of the test scores are non-finite: [       nan        nan 0.68
251641 0.62267566]
  warnings.warn(
```

Out[67]:

	model	best_score	best_params
0	linear_regression	0.844581	{'fit_intercept': False, 'n_jobs': -1}
1	lasso	0.717742	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.682516	{'criterion': 'friedman_mse', 'splitter': 'best'}

In [70]:

```
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 [71]:

```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[71]:
```
84.22924041793821
```

In [72]:
```
predict_price('1st Phase JP Nagar',1000, 3, 3)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[72]:
```
87.41872656162407
```

In [73]:
```
predict_price('Indira Nagar',1000, 2, 2)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[73]:
```
181.82117905122465
```

In [74]:
```
predict_price('Indira Nagar',1000, 3, 3)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[74]:
```
185.01066519491047
```

Export the tested model to a pickle file

In [75]:
```
import pickle
with open('banglore_home_prices_model.pickle','wb') as f:
    pickle.dump(lr_clf,f)
```

Export location and column information to a file that will be useful later on in our prediction application

In [76]:
```
import json
columns = {
    'data_columns' : [col.lower() for col in X.columns]
}
with open("columns.json","w") as f:
    f.write(json.dumps(columns))
```