In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Bengaluru_House_Data.csv')
df.head(5)
```

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

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

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

In [4]:
```
df['area_type'].unique()
```

Out[4]:
```
array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
       'Carpet  Area'], dtype=object)
```

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

Out[5]:
```
Super built-up  Area    8790
Built-up  Area          2418
Plot  Area              2025
Carpet  Area              87
Name: area_type, dtype: int64
```

In [6]:
```
df2 = df.drop(['area_type','society','balcony','availability'],axis='columns')
df2.shape
```

Out[6]:
```
(13320, 5)
```

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

Out[7]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [8]:
```
df3 = df2.dropna()
df3.isnull().sum()
```

Out[8]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [9]:
```
df3.shape
```

Out[9]:
```
(13246, 5)
```

Feature Engineeringg

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

```
C:\Users\NISHANT\AppData\Local\Temp\ipykernel_45688\339089679.py:1: SettingWithCopyWarning:
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/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df3['bhk'] = df3['size'].apply(lambda x:int (x.split(' ')[0]))
```

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

In [11]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [12]:
```
df3[~df3['total_sqft'].apply(is_float)].head(10)
```

Out[12]:

	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 [13]:
```
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 [14]:
```
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4 = df4[df4.total_sqft.notnull()]
df4.head()
```

Out[14]:

	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 [15]:
```
df4.loc[30]
```

Out[15]:
```
location      Yelahanka
size              4 BHK
total_sqft       2475.0
bath                4.0
price             186.0
bhk                   4
Name: 30, dtype: object
```

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

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
4	Kothanur	2 BHK	1200.0	2.0	51.00	2

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

In [18]:
```
df5_stats = df5['price_per_sqft'].describe()
df5_stats
```

Out[18]:
```
count    1.320000e+04
mean     7.920759e+03
std      1.067272e+05
min      2.678298e+02
25%      4.267701e+03
50%      5.438331e+03
75%      7.317073e+03
max      1.200000e+07
Name: price_per_sqft, dtype: float64
```

In [19]:
```
df5.to_csv("bhp.csv",index=False)
```

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_less_than_10 = location_stats[location_stats<=10]
location_less_than_10
```

Out[21]:
```
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 [22]:
```
len(df5.location.unique())
```

Out[22]:
```
1287
```

In [23]:
```
df5.location = df5.location.apply(lambda x: 'other' if x in location_less_than_10 else x)
len(df5.location.unique())
```

Out[23]: 241

In [24]:
```
df5.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]:
```
df5.shape
```

Out[25]: (13200, 7)

In [26]:
```
df5[df5.total_sqft/df5.bhk<300].head()
```

Out[26]:

	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 [27]:
```
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.head()
```

Out[27]:

	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 [28]:
```
df6.shape
```

Out[28]:
```
(12456, 7)
```

In [29]:
```
df6.price_per_sqft.describe()
```

Out[29]:
```
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
```

Function to remove outliers using mean and std

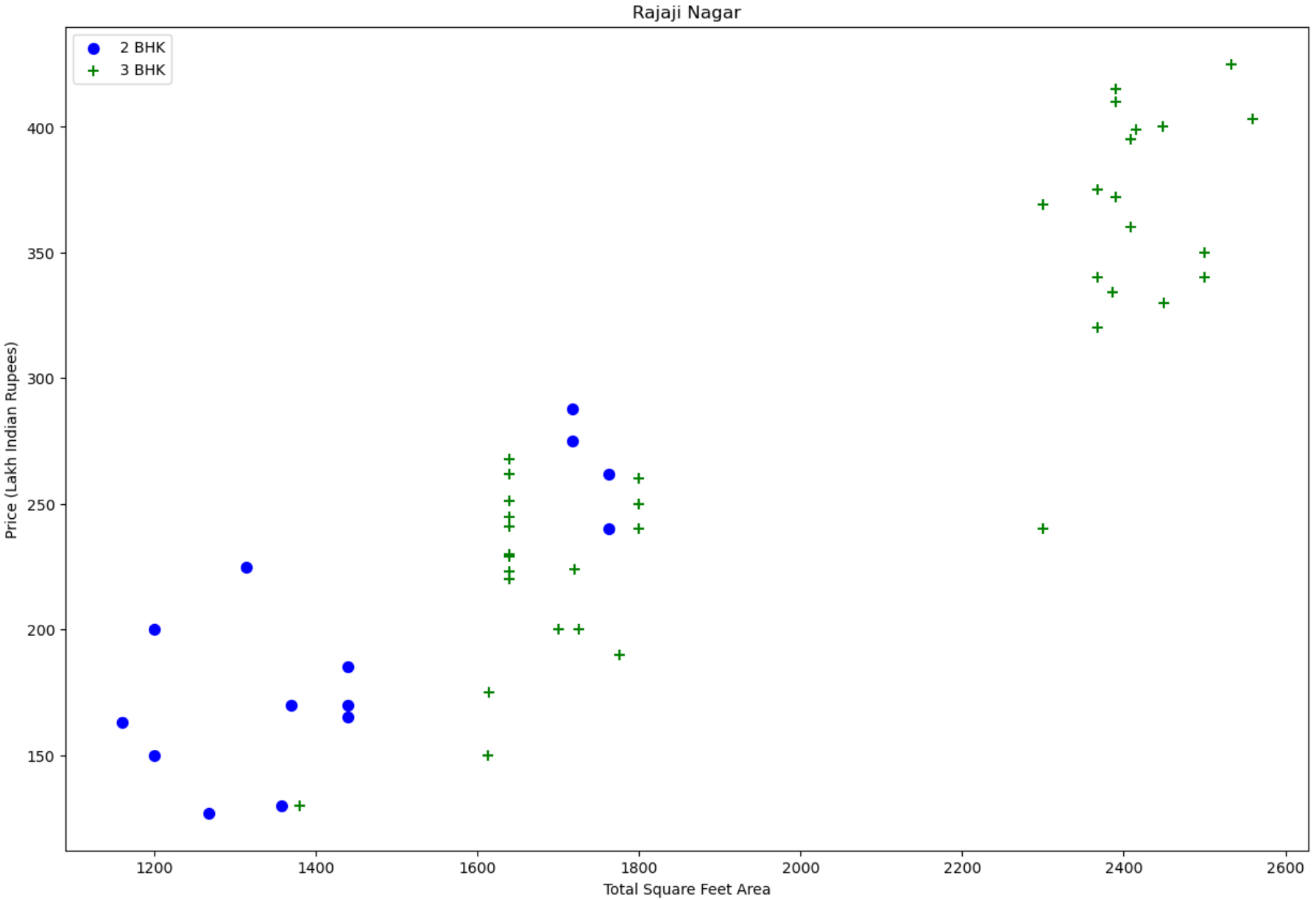
In [30]:
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index = True)
    return df_out
df7 = remove_pps_outliers(df6)
df7.shape
```

Out[30]:
```
(10242, 7)
```

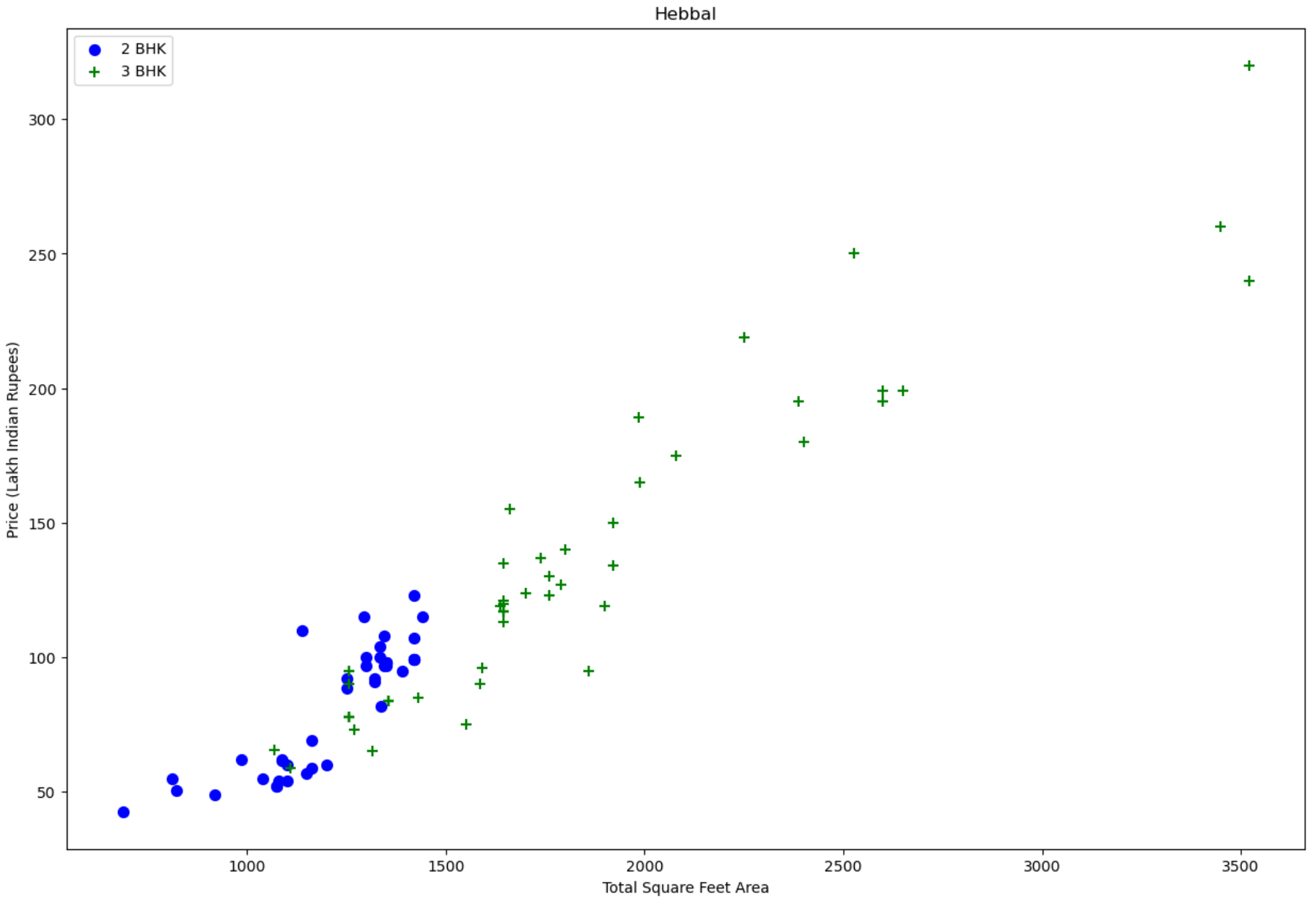
In [31]:
```
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
```

In [32]:
```
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()

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


Rajaji Nagar
2 BHK
3 BHK
Price (Lakh Indian Rupees)
Total Square Feet Area
400
350
300
250
200
150
1200
1400
1600
1800
2000
2200
2400
2600

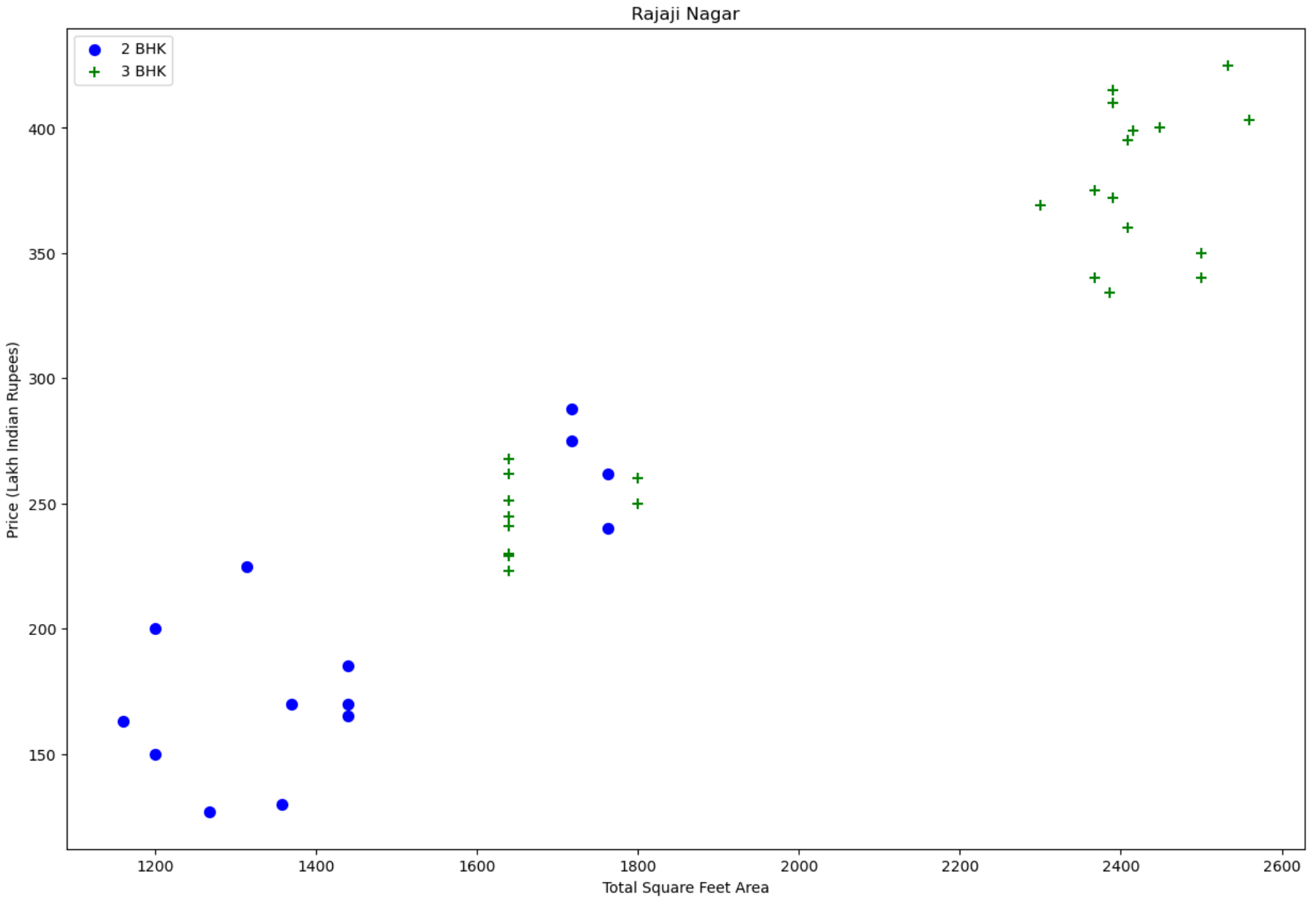
In [33]:
```
plot_scatter_chart(df7,"Hebbal")
```



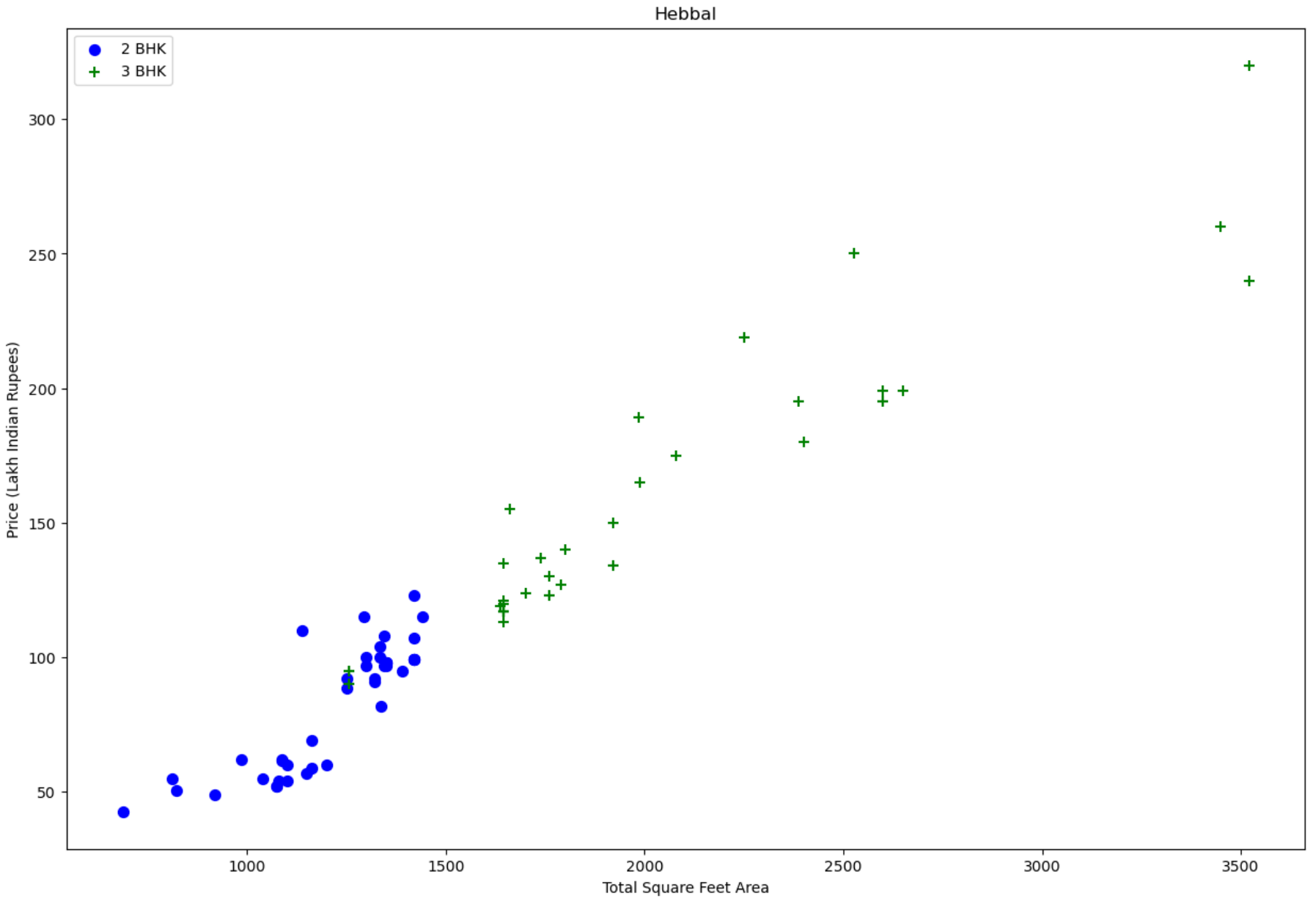
In [34]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                '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.value
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
```

Out[34]:
```
(7317, 7)
```

In [35]:
```
plot_scatter_chart(df8,"Rajaji Nagar")
```

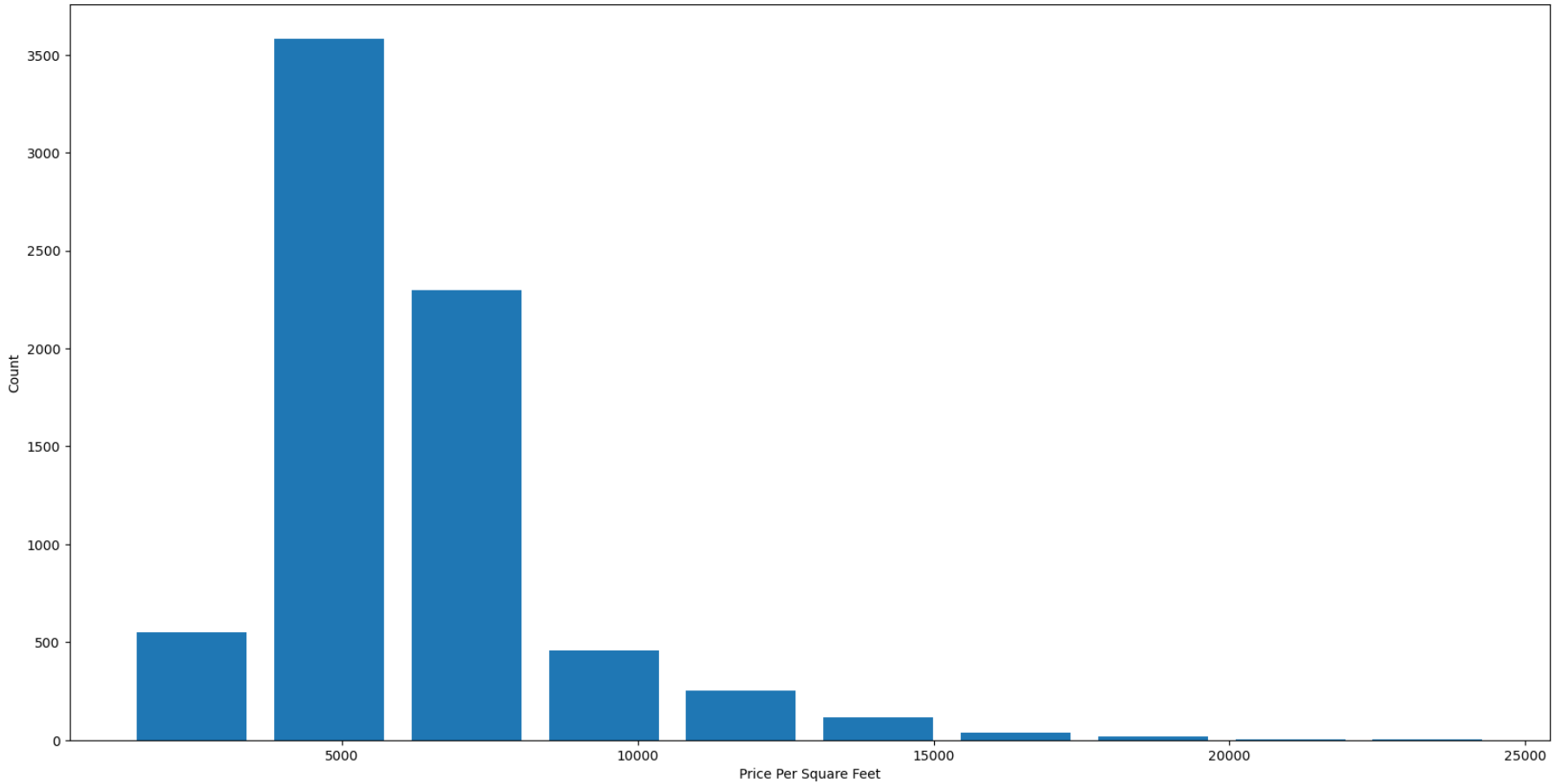


In [36]:
```
plot_scatter_chart(df8,"Hebbal")
```



In [37]:
```
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[37]:
```
Text(0, 0.5, 'Count')
```

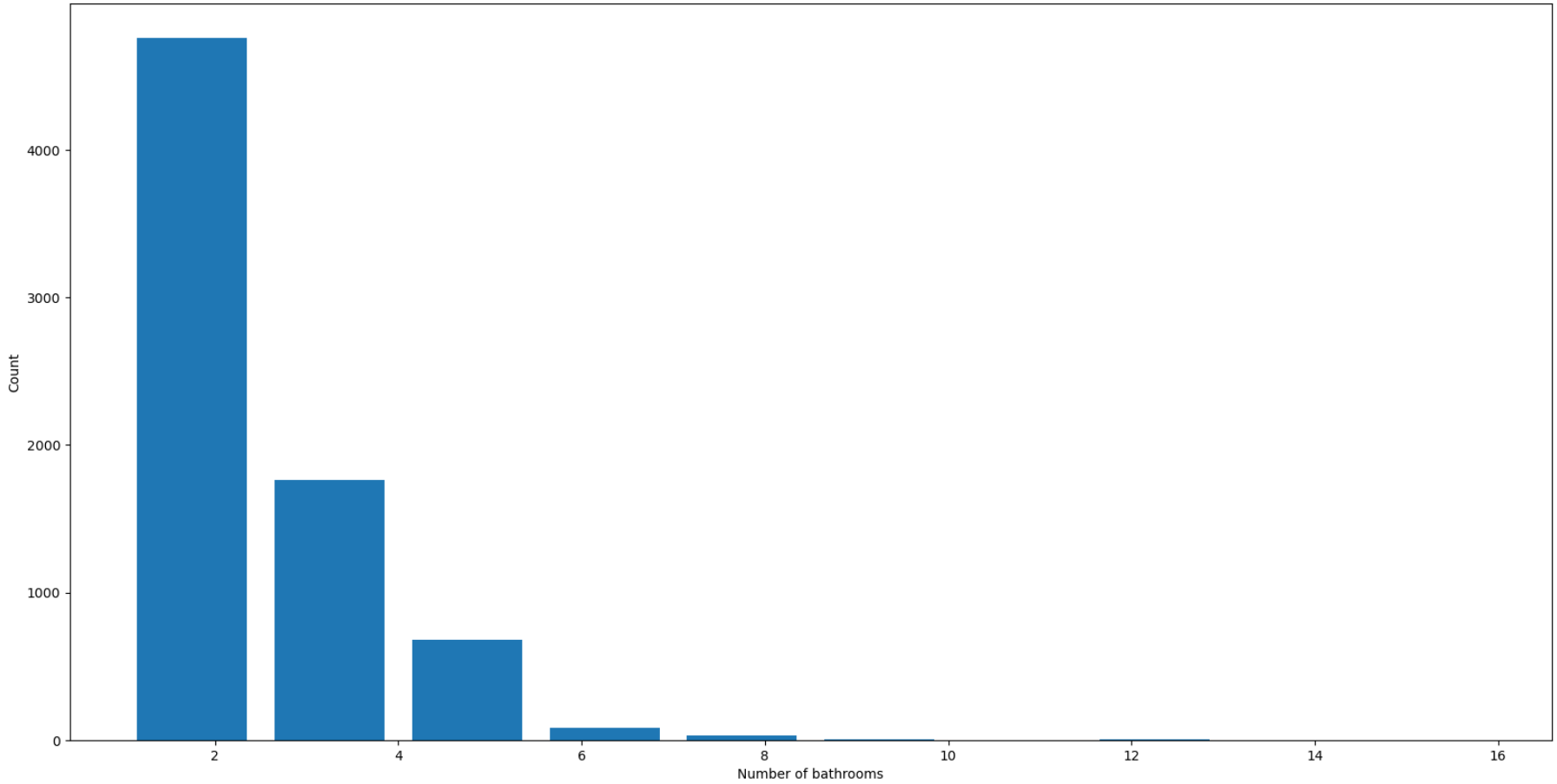


In [38]:
```
df8.bath.unique()
```

Out[38]:
```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

In [39]:
```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[39]:
```
Text(0, 0.5, 'Count')
```



In [40]:
```
df8[df8.bath>10]
```

Out[40]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8483	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8572	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9306	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9637	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

In [41]:
```
df8[df8.bath>df8.bhk+2]
```

Out[41]:

	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 [42]:
```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

Out[42]:
```
(7239, 7)
```

In [43]:
```
df9.head()
```

Out[43]:

	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 [44]:
```
df10 = df9.drop(['size','price_per_sqft'],axis='columns')
df10.head(3)
```

Out[44]:

	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

OHE

In [45]:
```
dummies = pd.get_dummies(df10.location)
dummies.head()
```

Out[45]:

	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
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 × 241 columns

In [46]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[46]:

	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	Vittasan
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	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	
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	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	
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...	0	0	0	

5 rows × 245 columns

In [47]:
```
df12 = df11.drop('location',axis='columns')
df12.head()
```

Out[47]:

	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
0	2850.0	4.0	428.0	4	1	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
2	1875.0	2.0	235.0	3	1	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
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	0	0	0	0

5 rows × 244 columns

In [48]:
```
df12.shape
```

Out[48]: (7239, 244)

In [49]:
```
X = df12.drop(['price'],axis='columns')
X.head(2)
```

Out[49]:

	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
0	2850.0	4.0	4	1	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

2 rows × 243 columns

In [50]:
```
y = df12.price
y.head(2)
```

Out[50]:
```
0    428.0
1    194.0
Name: price, dtype: float64
```

In [51]:
```
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 [52]:
```
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=10)
```

In [53]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[53]:
```
0.8629132245229525
```

In [54]:
```
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': {
                '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)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[54], line 42
     34         scores.append({
     35             'model': algo_name,
     36             'best_score': gs.best_score_,
     37             'best_params': gs.best_params_
     38         })
     40     return pd.DataFrame(scores,columns=['model','best_score','best_params'])
---> 42 find_best_model_using_gridsearchcv(X,y)

Cell In[54], line 33, in find_best_model_using_gridsearchcv(X, y)
     31 for algo_name, config in algos.items():
     32     gs =  GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
---> 33     gs.fit(X,y)
     34     scores.append({
     35         'model': algo_name,
     36         'best_score': gs.best_score_,
     37         'best_params': gs.best_params_
```

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

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