HOUSE PRICE

The data house_price.csv which contains property prices in the city of Bangalore. Examine price per square feet

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

In [2]:
```
data=pd.read_csv("C:/Users/Acer/Downloads/house_price - house_price.csv")
data
```

Out[2]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2	39.07	2	3699
1	Chikka Tirupathi	4 Bedroom	2600.0	5	120.00	4	4615
2	Uttarahalli	3 BHK	1440.0	2	62.00	3	4305
3	Lingadheeranahalli	3 BHK	1521.0	3	95.00	3	6245
4	Kothanur	2 BHK	1200.0	2	51.00	2	4250
...	...	...	...	...	...	...	...
13195	Whitefield	5 Bedroom	3453.0	4	231.00	5	6689
13196	other	4 BHK	3600.0	5	400.00	4	11111
13197	Raja Rajeshwari Nagar	2 BHK	1141.0	2	60.00	2	5258
13198	Padmanabhanagar	4 BHK	4689.0	4	488.00	4	10407
13199	Doddathoguru	1 BHK	550.0	1	17.00	1	3090

13200 rows × 7 columns

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13200 entries, 0 to 13199
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   location        13200 non-null  object 
 1   size            13200 non-null  object 
 2   total_sqft      13200 non-null  float64
 3   bath            13200 non-null  int64  
 4   price           13200 non-null  float64
 5   bhk             13200 non-null  int64  
 6   price_per_sqft  13200 non-null  int64  
dtypes: float64(2), int64(3), object(2)
memory usage: 722.0+ KB
```

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

Out[4]:
```
(13200, 7)
```

Data-PrePocessing

In [5]:
```
data.isnull().sum() # there is no null value in this data
```

Out[5]:
```
location          0
size              0
total_sqft        0
bath              0
price             0
bhk               0
price_per_sqft    0
dtype: int64
```

In [6]:
```
data.duplicated().sum() # 1049 duplicate is present
```

Out[6]:
```
1049
```

In [7]:
```
data=data.drop_duplicates() # drop all the duplicates
```

In [8]:
```
data.duplicated().sum() # duplicates has been droped from the data
```

Out[8]:
```
0
```

In [9]:
```
data.describe()
```

Out[9]:

	total_sqft	bath	price	bhk	price_per_sqft
count	12151.000000	12151.000000	12151.000000	12151.000000	1.215100e+04
mean	1574.846013	2.719941	115.471328	2.827504	8.132642e+03
std	1277.328354	1.372210	154.094133	1.326540	1.112329e+05
min	1.000000	1.000000	8.000000	1.000000	2.670000e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.312000e+03
50%	1290.000000	2.000000	74.000000	3.000000	5.500000e+03
75%	1700.000000	3.000000	123.500000	3.000000	7.461000e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

In [10]:
```
data.shape # after droping duplicates the shape of the data is been reduced from 13200 to 12151
```

Out[10]:
```
(12151, 7)
```

In [11]:
```
data.info()
```

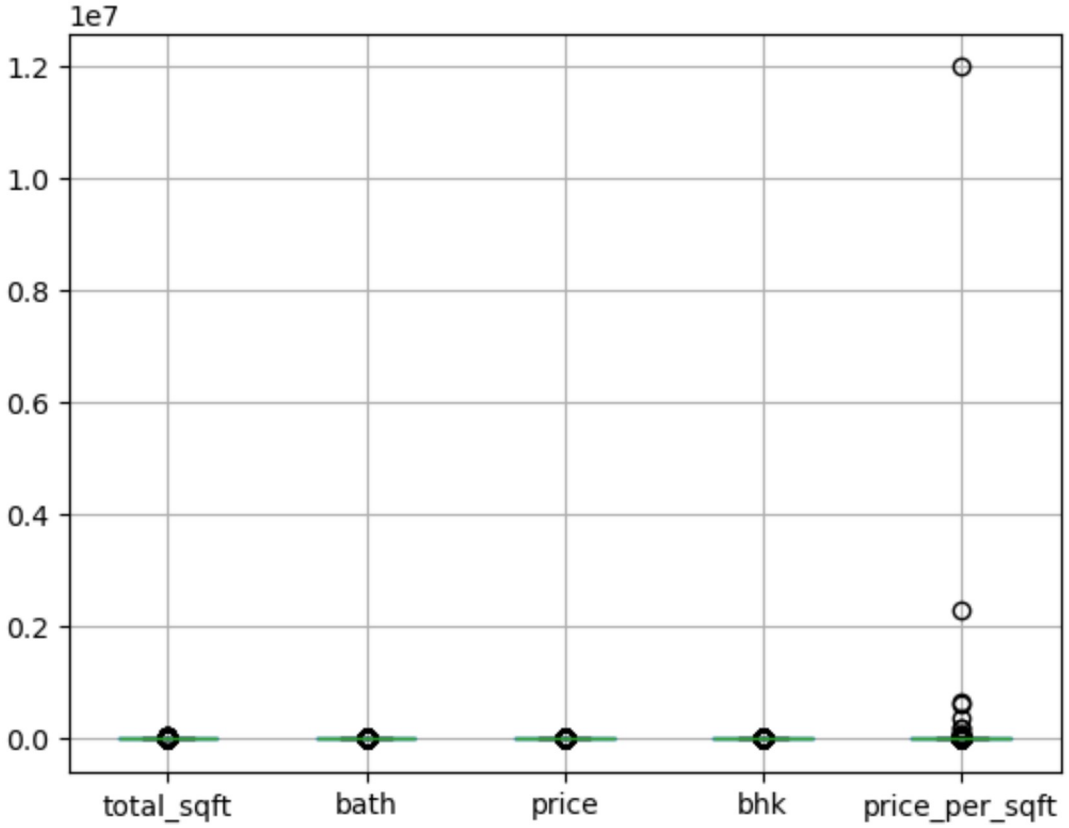
```
<class 'pandas.core.frame.DataFrame'>
Index: 12151 entries, 0 to 13198
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   location        12151 non-null  object 
 1   size            12151 non-null  object 
 2   total_sqft      12151 non-null  float64
 3   bath            12151 non-null  int64  
 4   price           12151 non-null  float64
 5   bhk             12151 non-null  int64  
 6   price_per_sqft  12151 non-null  int64  
dtypes: float64(2), int64(3), object(2)
memory usage: 759.4+ KB
```

Checking Outliers

In [12]:
```
df=data.select_dtypes('number')
df.skew()
```

Out[12]:
```
total_sqft         15.112123
bath                4.214944
price               7.915103
bhk                 4.838129
price_per_sqft    103.902032
dtype: float64
```

In [13]:
```
df.boxplot()
plt.show()  # outlier is present in this boxplot
```



In [14]:
```
df.describe()
```

Out[14]:

	total_sqft	bath	price	bhk	price_per_sqft
count	12151.000000	12151.000000	12151.000000	12151.000000	1.215100e+04
mean	1574.846013	2.719941	115.471328	2.827504	8.132642e+03
std	1277.328354	1.372210	154.094133	1.326540	1.112329e+05
min	1.000000	1.000000	8.000000	1.000000	2.670000e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.312000e+03
50%	1290.000000	2.000000	74.000000	3.000000	5.500000e+03
75%	1700.000000	3.000000	123.500000	3.000000	7.461000e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

In [15]:
```
mean = df.mean()
mean
```

Out[15]:
```
total_sqft        1574.846013
bath                 2.719941
price              115.471328
bhk                  2.827504
price_per_sqft    8132.641840
dtype: float64
```

In [16]:
```
stdvt = df['price_per_sqft'].std()
print("Price per sqft according to Standard deviation  :",stdvt)

# The standard deviation is extremly high so it means there is unusal data recorded .
```

```
Price per sqft according to Standard deviation  : 111232.9008957087
```

In [17]:
```
def remove_outliers(df,columns):
    filt = df.copy()

    for i in columns:
        print('---',i,'---')
        Q1 = df[i].quantile(0.25)
        print('Q1: ',Q1)
        Q3 = df[i].quantile(0.75)
```

```
        print('Q3 :',Q3)
        IQR = Q3 - Q1
        print('IQR :',IQR)

        lower_percentile = Q1 - 1.5 * IQR
        print('lower_percentile :',lower_percentile)
        upper_percentile = Q3 + 1.5 * IQR
        print('upper_percentile :',upper_percentile)

        filt = filt[(filt[i] >= lower_percentile) & (filt[i] <= upper_percentile)]

    return filt
```

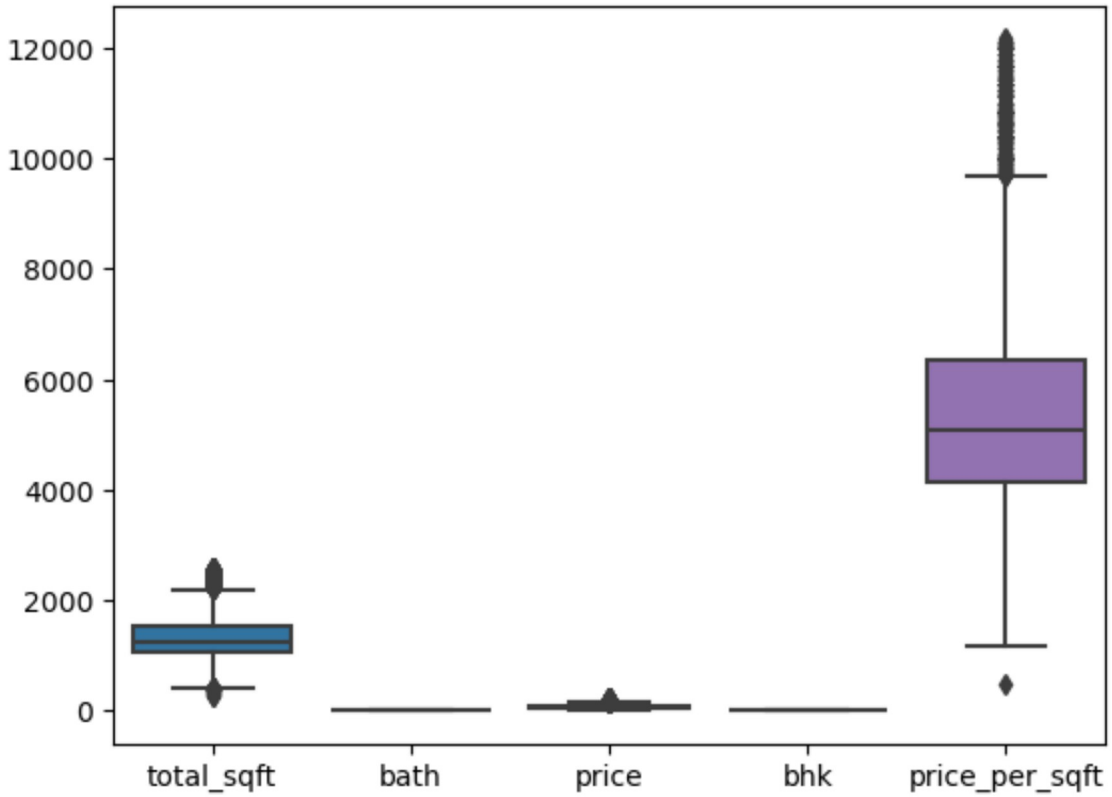
In [18]:
```
ot = remove_outliers(df,['total_sqft','bath','price','bhk','price_per_sqft'])
```

```
--- total_sqft ---
Q1:  1100.0
Q3 : 1700.0
IQR : 600.0
lower_percentile : 200.0
upper_percentile : 2600.0
--- bath ---
Q1:  2.0
Q3 : 3.0
IQR : 1.0
lower_percentile : 0.5
upper_percentile : 4.5
--- price ---
Q1:  50.0
Q3 : 123.5
IQR : 73.5
lower_percentile : -60.25
upper_percentile : 233.75
--- bhk ---
Q1:  2.0
Q3 : 3.0
IQR : 1.0
lower_percentile : 0.5
upper_percentile : 4.5
--- price_per_sqft ---
Q1:  4312.0
Q3 : 7461.0
IQR : 3149.0
lower_percentile : -411.5
upper_percentile : 12184.5
```

In [19]:
```
ot.skew()
```

Out[19]:
```
total_sqft        0.544893
bath              0.557225
price             1.237767
bhk               0.087245
price_per_sqft    1.031238
dtype: float64
```

In [20]:
```
sns.boxplot(ot)
plt.show()
```



In [21]:
```
ot.describe()
```

Out[21]:

	total_sqft	bath	price	bhk	price_per_sqft
count	9762.000000	9762.000000	9762.000000	9762.000000	9762.000000
mean	1310.846302	2.284368	72.596379	2.424503	5422.855460
std	385.797383	0.655382	36.599679	0.661006	1806.044251
min	276.000000	1.000000	8.000000	1.000000	500.000000
25%	1080.000000	2.000000	46.290000	2.000000	4132.000000
50%	1245.000000	2.000000	64.095000	2.000000	5095.000000
75%	1530.000000	3.000000	90.000000	3.000000	6357.750000
max	2600.000000	4.000000	230.000000	4.000000	12166.000000

In [22]:
```
def zscore_filt (df,j):
    mean=df[j].mean()
    print('Mean of ',j,':',mean)

    sd=df[j].std()
    print('standard Deviation of',j,' :',sd)

    df['zscore']=(df[j]-mean)/sd
    print('Zscore of',j,':',ot)

    cln=df[(df['zscore'] >= -1.5 ) & (df['zscore'] <= 1.5)]
    return cln
```

In [23]:
```
ot=zscore_filt(ot,'price_per_sqft')
```

```
Mean of  price_per_sqft : 5422.8554599467325
standard Deviation of price_per_sqft  : 1806.0442506679533
Zscore of price_per_sqft :        total_sqft  bath   price  bhk  price_per_sqft    zscore
0          1056.0     2   39.07    2            3699 -0.954492
2          1440.0     2   62.00    3            4305 -0.618952
3          1521.0     3   95.00    3            6245  0.455218
4          1200.0     2   51.00    2            4250 -0.649406
5          1170.0     2   38.00    2            3247 -1.204763
...           ...   ...     ...  ...             ...       ...
13189      1675.0     3   92.13    3            5500  0.042715
13190      1050.0     2   52.71    2            5020 -0.223060
13192      1262.0     2   47.00    2            3724 -0.940650
13194      1715.0     3  112.00    3            6530  0.613022
13197      1141.0     2   60.00    2            5258 -0.091280

[9762 rows x 6 columns]
```

In [42]:
```
ot=ot.drop(columns=['zscore'])
```

In [25]:
```
ot=zscore_filt(ot,'price')
```

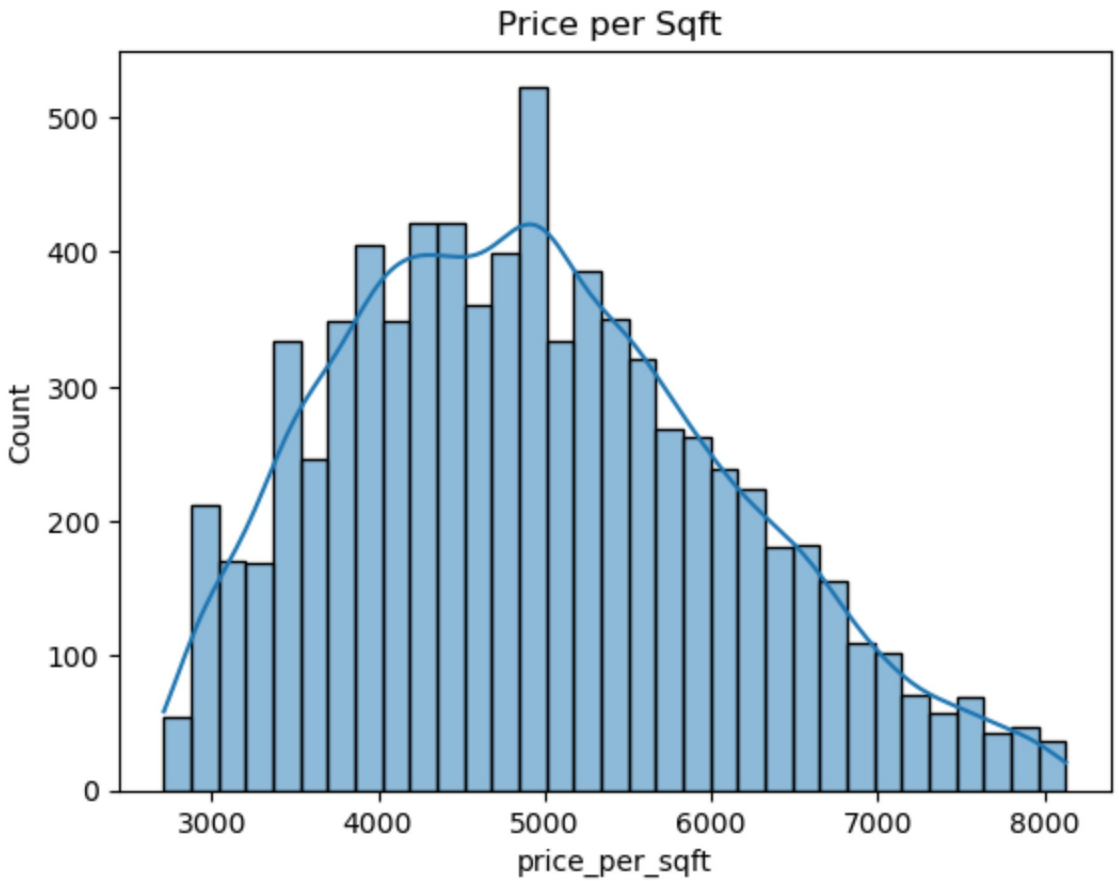
```
Mean of  price : 67.99959861473829
standard Deviation of price  : 30.138931128662396
Zscore of price :        total_sqft  bath   price  bhk  price_per_sqft    zscore
0          1056.0     2   39.07    2            3699 -0.959875
2          1440.0     2   62.00    3            4305 -0.199065
3          1521.0     3   95.00    3            6245  0.895865
4          1200.0     2   51.00    2            4250 -0.564041
5          1170.0     2   38.00    2            3247 -0.995377
...           ...   ...     ...  ...             ...       ...
13189      1675.0     3   92.13    3            5500  0.800639
13190      1050.0     2   52.71    2            5020 -0.507304
13192      1262.0     2   47.00    2            3724 -0.696760
13194      1715.0     3  112.00    3            6530  1.459919
13197      1141.0     2   60.00    2            5258 -0.265424

[8807 rows x 6 columns]
```

In [43]:
```
ot.skew()
```

Out[43]:
```
total_sqft           0.415284
bath                 0.665547
price                0.440850
bhk                  0.180185
price_per_sqft       0.375056
price_per_sqft_log  -0.115217
dtype: float64
```

In [27]:
```
sns.histplot(ot['price_per_sqft'], kde=True)
plt.title('Price per Sqft ')
plt.show()
```

```
C:\Users\Acer\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



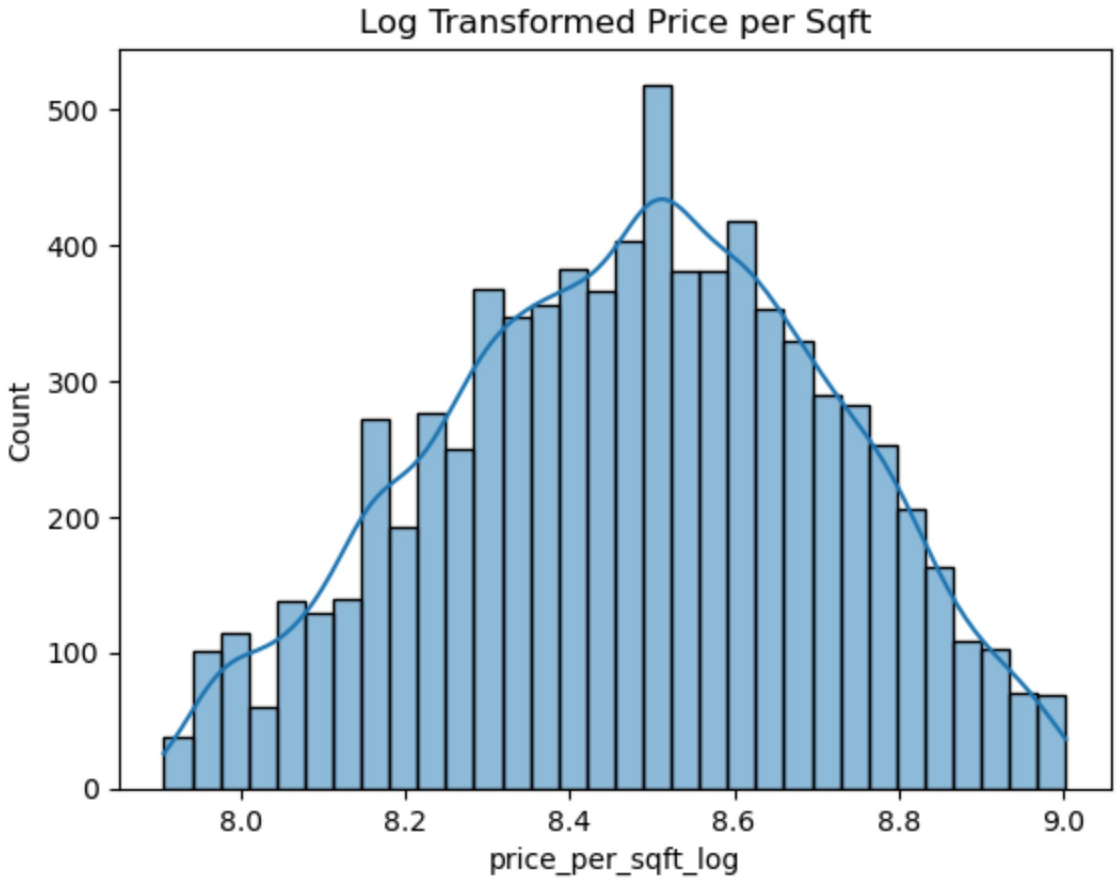
In [28]:
```
os = ot['price_per_sqft'].skew()
ok =ot['price_per_sqft'].kurtosis()

print("Skewness:" ,os)
print("Kurtosis:", ok)
```

```
Skewness: 0.3750559731977
Kurtosis: -0.4257647865559515
```

In [29]:
```
ot['price_per_sqft_log'] = np.log1p(ot['price_per_sqft'])
```

In [30]:
```
sns.histplot(ot['price_per_sqft_log'], kde=True)
plt.title('Log Transformed Price per Sqft')
plt.show()
```

```
C:\Users\Acer\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

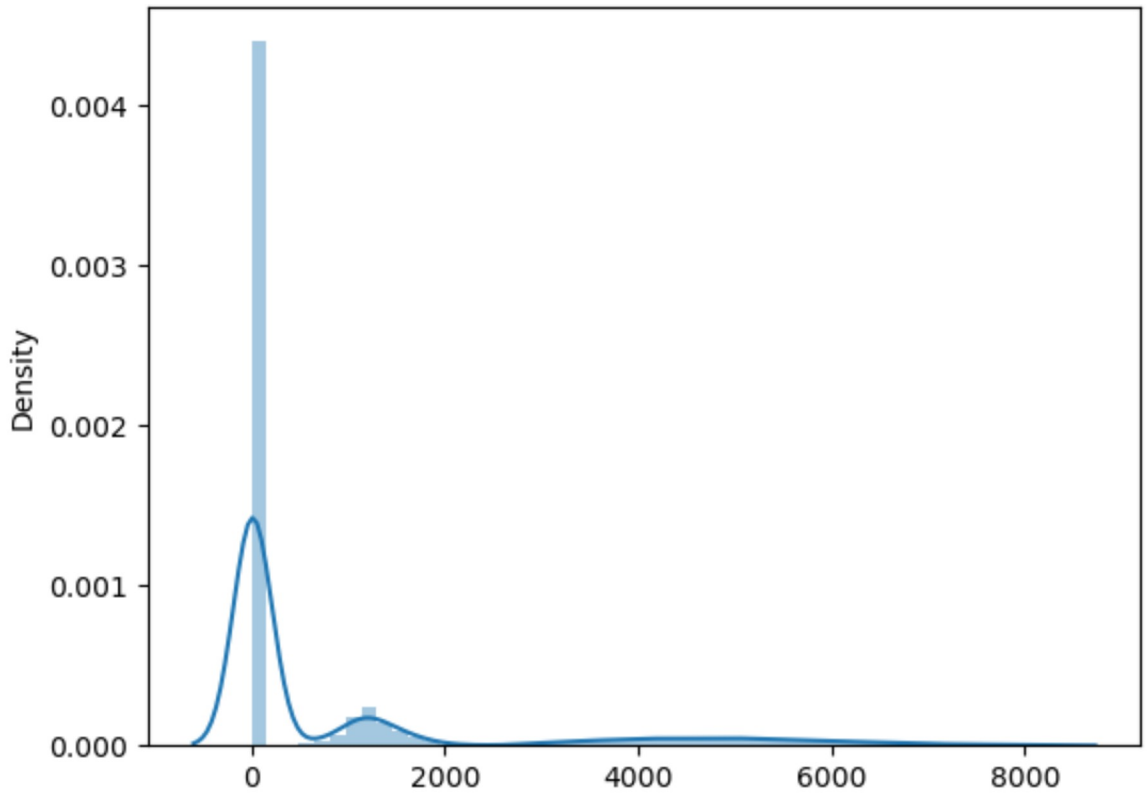


In [31]:
```
ts = ot['price_per_sqft_log'].skew()
tk = ot['price_per_sqft_log'].kurtosis()

print("Transformed Skewness: ",ts)
print("Transformed Kurtosis: ",tk)
```

```
Transformed Skewness:  -0.11521650152110748
Transformed Kurtosis:  -0.6059567937203156
```

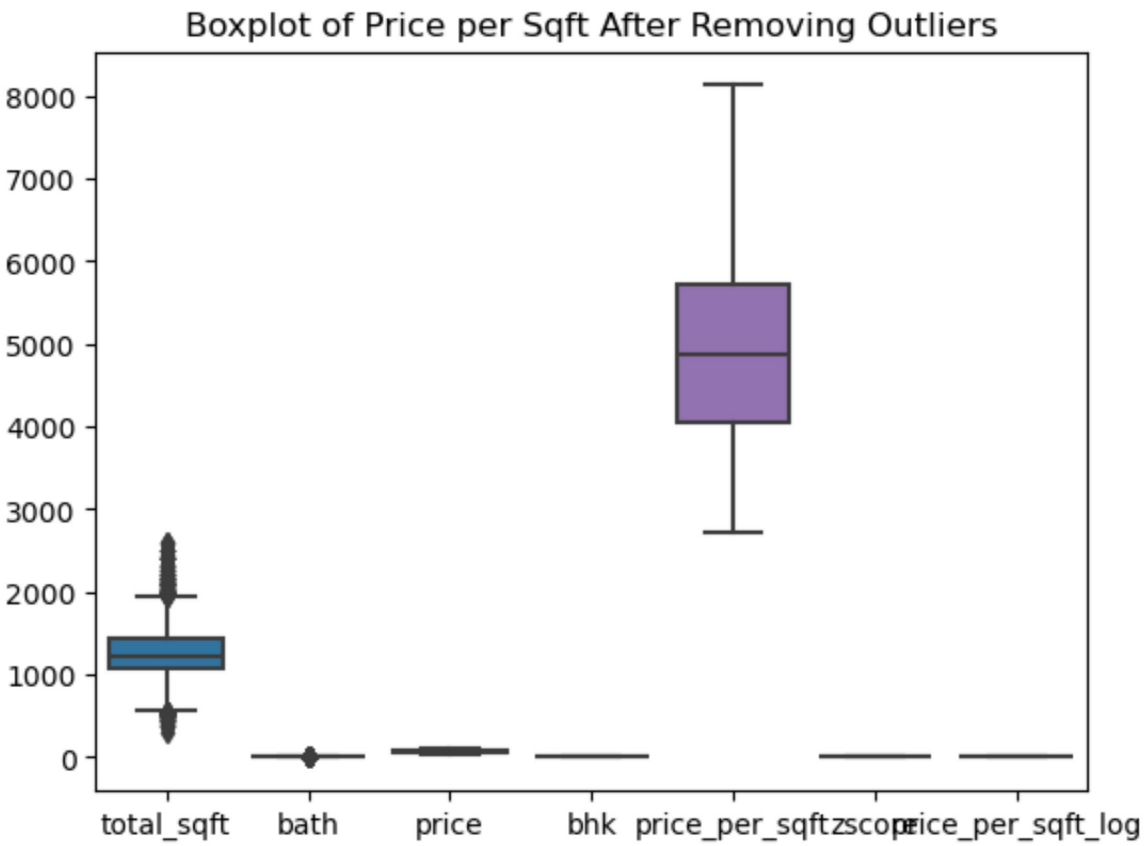
In [32]:
```
sns.distplot(ot)
plt.show()
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_65140\322586427.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(ot)
C:\Users\Acer\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

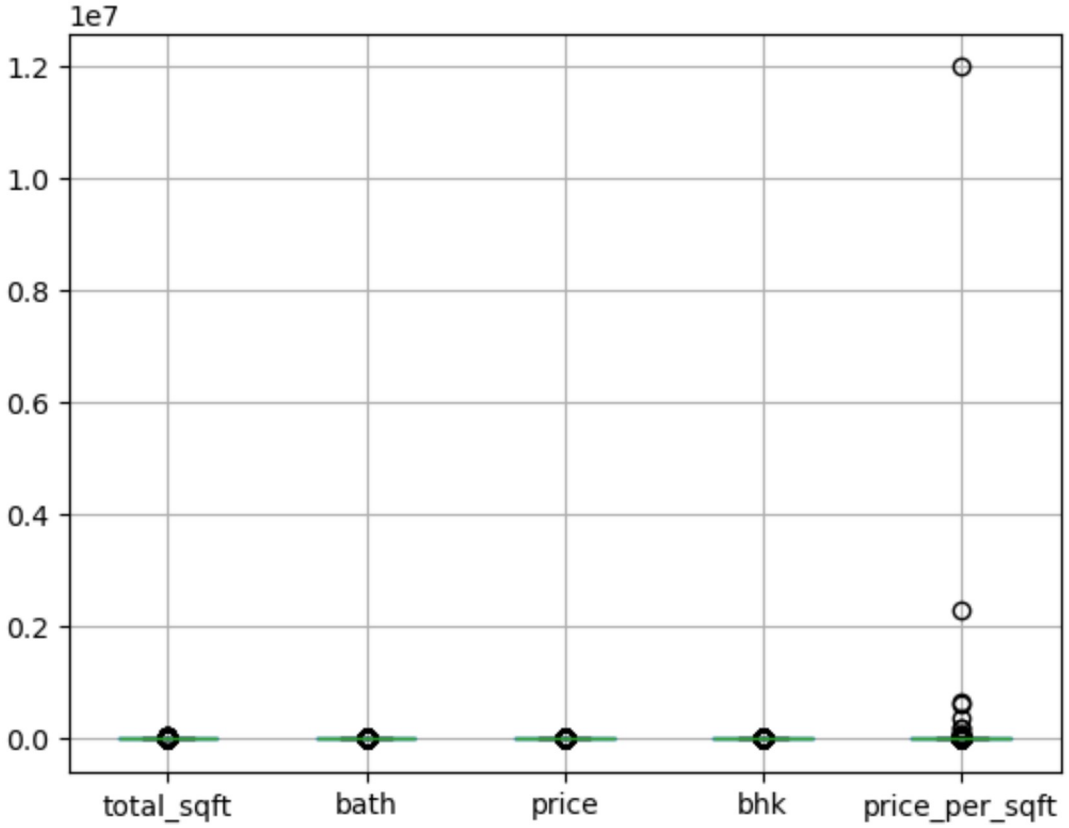


In [33]:
```
sns.boxplot(ot)
plt.title('Boxplot of Price per Sqft After Removing Outliers')
```

```
plt.show()
```

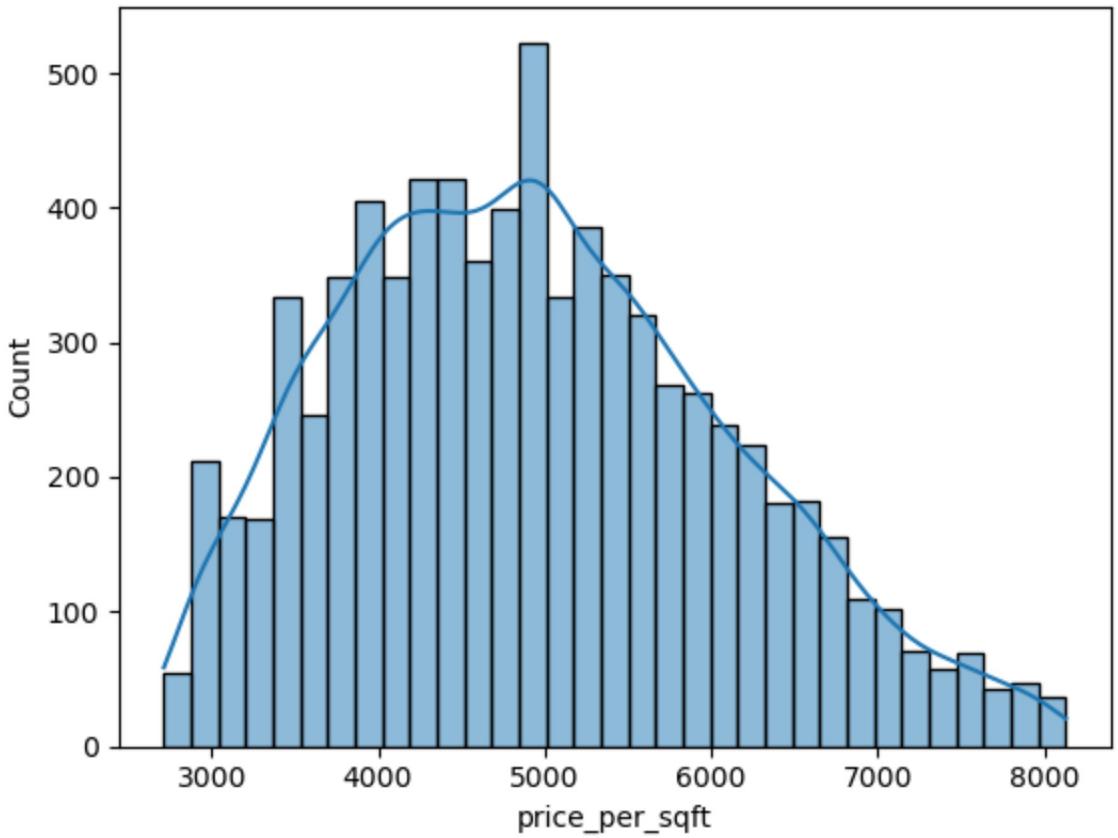


In [34]:
```
df.boxplot()
plt.show()
```

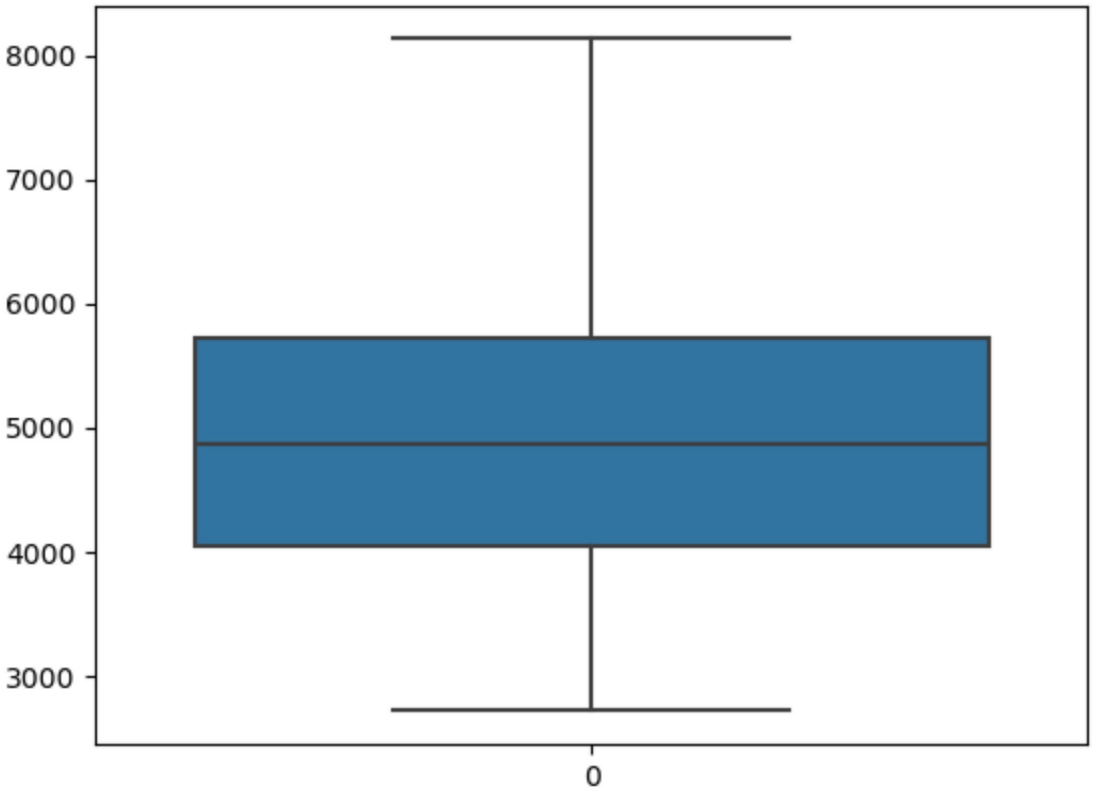


In [35]:
```
sns.histplot(ot['price_per_sqft'],kde=True)
plt.show()
```

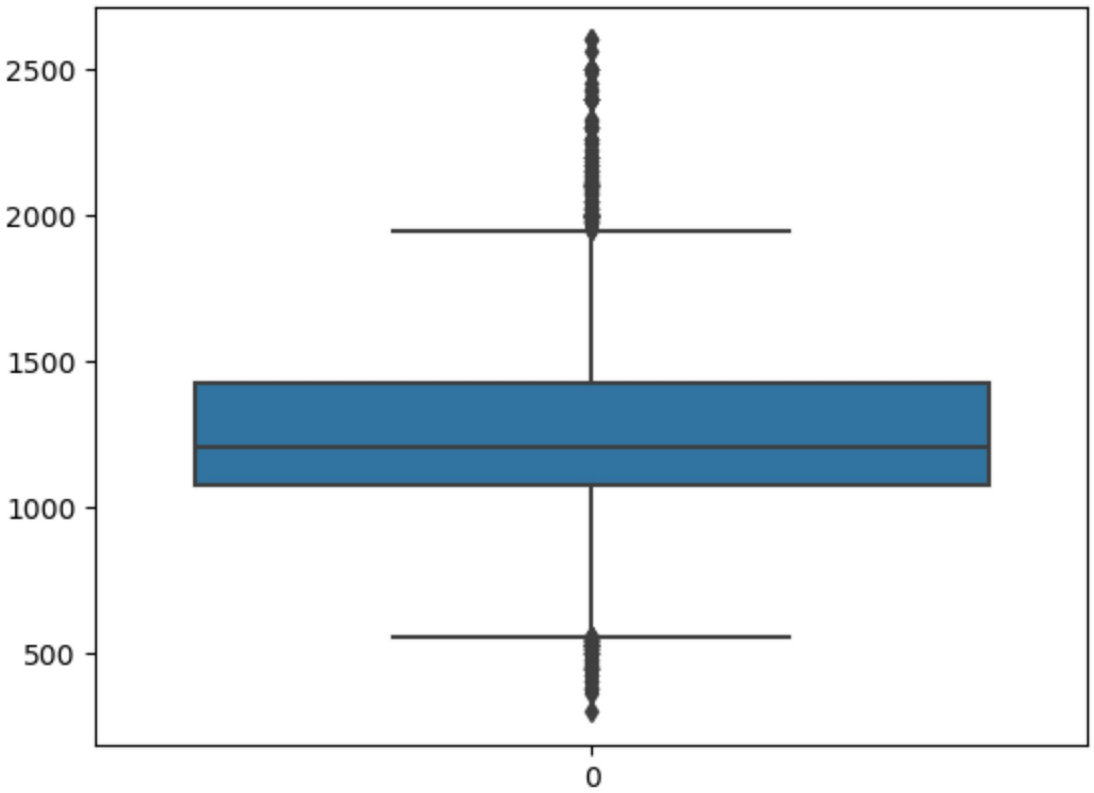
```
C:\Users\Acer\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [36]:
```
sns.boxplot(ot['price_per_sqft'])
plt.show()
```



In [37]:
```
sns.boxplot(ot['total_sqft'])
plt.show()
```



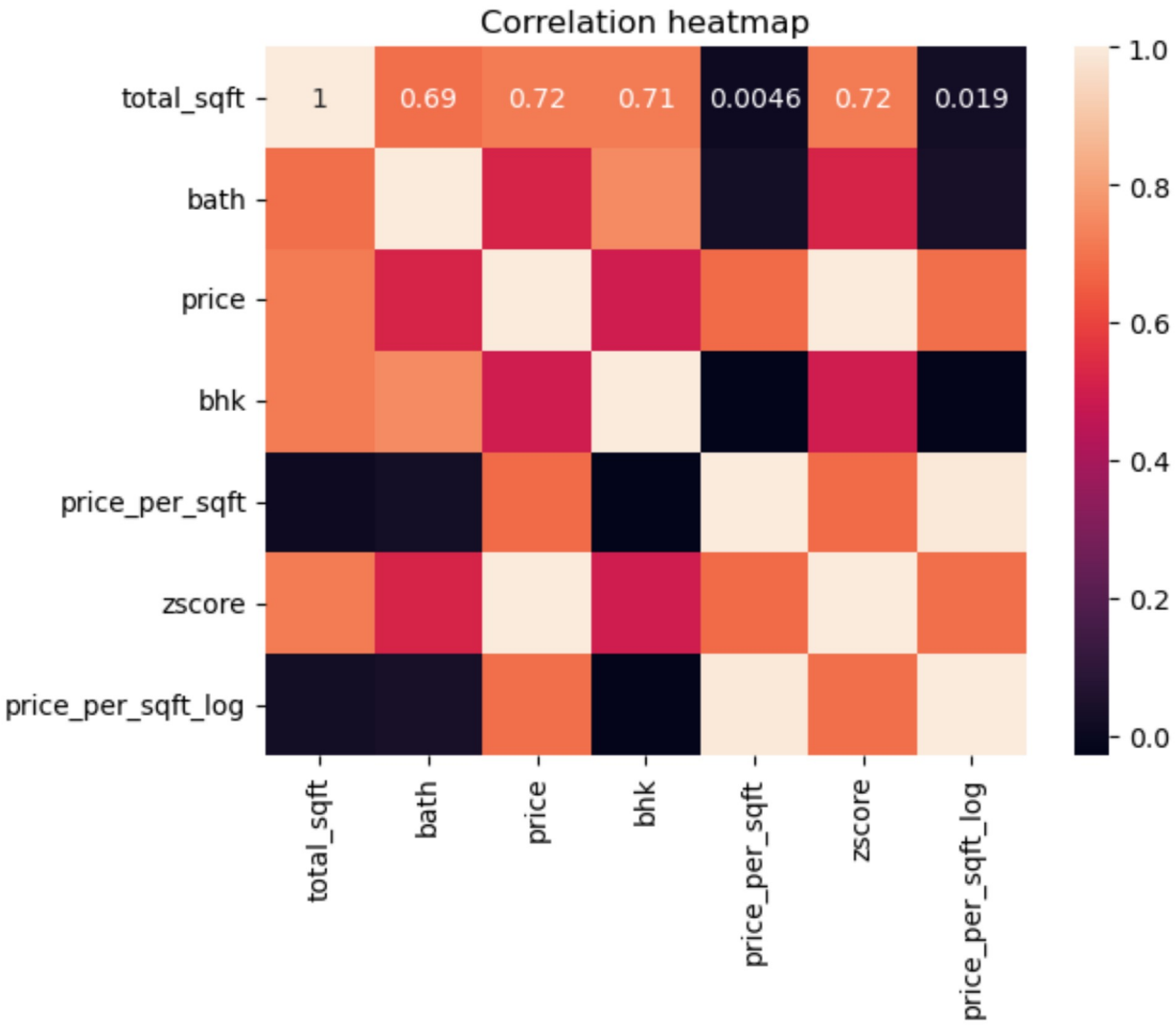
In [38]:
```
corr=ot.corr()
corr.head()
```

Out[38]:

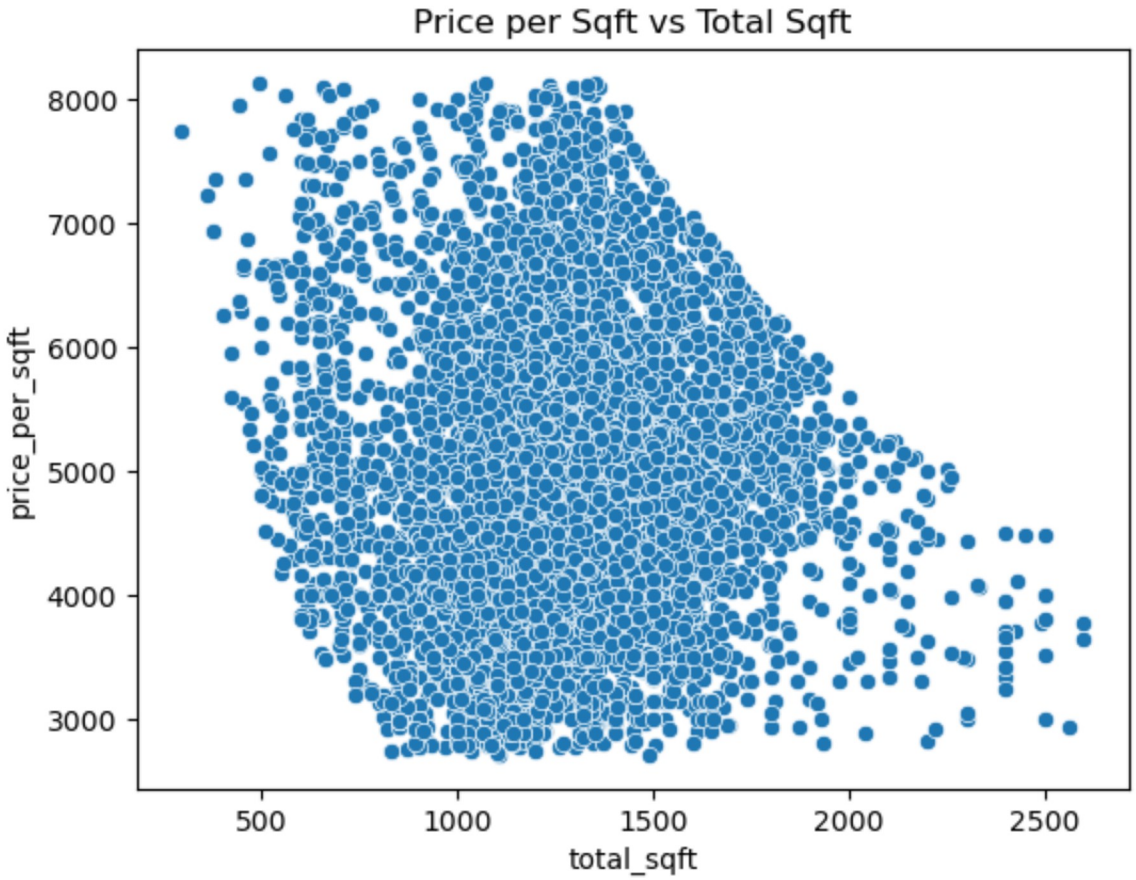
	total_sqft	bath	price	bhk	price_per_sqft	zscore	price_per_sqft_log
total_sqft	1.000000	0.688821	0.717111	0.714675	0.004608	0.717111	0.019469
bath	0.688821	1.000000	0.518934	0.751437	0.026606	0.518934	0.037338
price	0.717111	0.518934	1.000000	0.493999	0.682317	1.000000	0.688087
bhk	0.714675	0.751437	0.493999	1.000000	-0.029196	0.493999	-0.021636
price_per_sqft	0.004608	0.026606	0.682317	-0.029196	1.000000	0.682317	0.990452

In [39]:
```
sns.heatmap(corr ,annot=True)
```

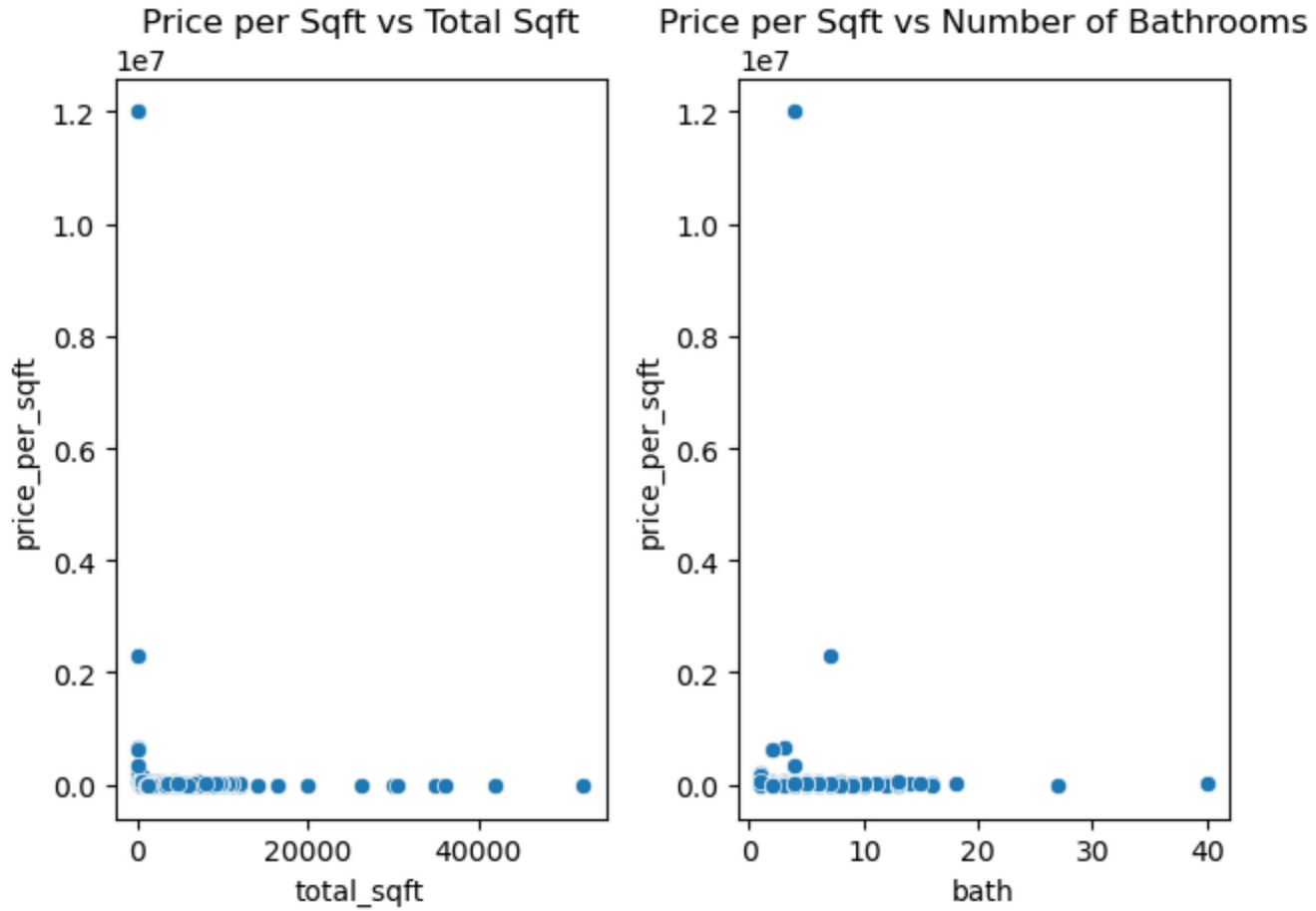
```
plt.title('Correlation heatmap')
plt.show()
```



In [40]:
```
sns.scatterplot(ot,x='total_sqft', y='price_per_sqft')
plt.title('Price per Sqft vs Total Sqft')
plt.show()
```



In [41]:
```
plt.subplot(1, 2, 1)
sns.scatterplot(data=data, x='total_sqft', y='price_per_sqft')
plt.title('Price per Sqft vs Total Sqft')

plt.subplot(1, 2, 2)
sns.scatterplot(data=data, x='bath', y='price_per_sqft')
plt.title('Price per Sqft vs Number of Bathrooms')

plt.tight_layout()
plt.show()
```


Price per Sqft vs Total Sqft
1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
price_per_sqft
0
20000
40000
total_sqft
Price per Sqft vs Number of Bathrooms
1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
price_per_sqft
0
10
20
30
40
bath

In [ ]: