ASSIGNMENT 1 - STATISTICAL MEASURES

You are provided with the file **house_price.csv**, which contains property prices for the city of Bangalore. Your task is to analyze the price per square foot and perform the following steps:

SOURCE

Dataset link: https://drive.google.com/file/d/1UlWRYU0UglE2ex3iFse0J6eCLEU8g
usp=sharing

IMPORTING MODULES

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

LOAD DATASET

In [5]:
```
# LOAD THE DATASET
data = pd.read_csv("house_price.csv")
```

```
data
```

Out[5]:

	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
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250
...	...	...	...	...	...	...	...
13195	Whitefield	5 Bedroom	3453.0	4.0	231.00	5	6689
13196	other	4 BHK	3600.0	5.0	400.00	4	11111
13197	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2	5258
13198	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4	10407
13199	Doddathoguru	1 BHK	550.0	1.0	17.00	1	3090

13200 rows × 7 columns

BASIC EDA

1. DISPLAY FIRST & LAST ROWS

In [8]:

```
# DISPLAY FIRST FEW ROES TO UNDERSTAND THE STRUCTURE OF THE DATA
print("First Few Rows: ")
data.head(10)
```

```
First Few Rows:
```

Out[8]:

	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
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250
5	Whitefield	2 BHK	1170.0	2.0	38.00	2	3247
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4	7467
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4	18181
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3	4828
9	other	6 Bedroom	1020.0	6.0	370.00	6	36274

In [10]:

```
# DISPLAY LAST FEW ROWS
print("Last Few Rows: ")
data.tail(10)
```

```
Last Few Rows:
```

Out[10]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
13190	Rachenahalli	2 BHK	1050.0	2.0	52.71	2	5020
13191	Ramamurthy Nagar	7 Bedroom	1500.0	9.0	250.00	7	16666
13192	Bellandur	2 BHK	1262.0	2.0	47.00	2	3724
13193	Uttarahalli	3 BHK	1345.0	2.0	57.00	3	4237
13194	Green Glen Layout	3 BHK	1715.0	3.0	112.00	3	6530
13195	Whitefield	5 Bedroom	3453.0	4.0	231.00	5	6689
13196	other	4 BHK	3600.0	5.0	400.00	4	11111
13197	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2	5258
13198	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4	10407
13199	Doddathoguru	1 BHK	550.0	1.0	17.00	1	3090

2. MAKE COPY OF ORIGINAL DATASET

In [12]:

```
# CREATE COPY OF ORIGINAL DATASET
data_copy = data.copy()
data_copy
```

Out[12]:

	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
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250
...	...	...	...	...	...	...	...
13195	Whitefield	5 Bedroom	3453.0	4.0	231.00	5	6689
13196	other	4 BHK	3600.0	5.0	400.00	4	11111
13197	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2	5258
13198	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4	10407
13199	Doddathoguru	1 BHK	550.0	1.0	17.00	1	3090

13200 rows × 7 columns

3. SHAPE OF THE DATA

In [14]:
```
# SHAPE OF THE DATASET
print("Shape of the data:")
data.shape
```

```
Shape of the data:
```

Out[14]:
```
(13200, 7)
```

4. DATATYPE OF EACH COLUMN

In [16]:
```
# DISPLAY DATA TYPE OF EACH COLUMN
print("Dataset Info:")
data.info()
```

```
Dataset 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  float64
 4   price           13200 non-null  float64
 5   bhk             13200 non-null  int64  
 6   price_per_sqft  13200 non-null  int64  
dtypes: float64(3), int64(2), object(2)
memory usage: 722.0+ KB
```

5. STATISTICAL SUMMARY OF DATA

In [18]:

```
# DISPLAY STATISTICSL SUMMARY
print("Statistical Summary:")
data.describe()
```

```
Statistical Summary:
```

Out[18]:

	total_sqft	bath	price	bhk	price_per_sqft
count	13200.000000	13200.000000	13200.000000	13200.000000	1.320000e+04
mean	1555.302783	2.691136	112.276178	2.800833	7.920337e+03
std	1237.323445	1.338915	149.175995	1.292843	1.067272e+05
min	1.000000	1.000000	8.000000	1.000000	2.670000e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.267000e+03
50%	1275.000000	2.000000	71.850000	3.000000	5.438000e+03
75%	1672.000000	3.000000	120.000000	3.000000	7.317000e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

6. DISPLAY ALL COLUMN NAMES

In [20]:

```
# DISPLAY PARTICULAR COLUMN
print("Columns of the dataset:")
data.columns
```

```
Columns of the dataset:
```

Out[20]:

```
Index(['location', 'size', 'total_sqft', 'bath', 'price', 'bhk',
       'price_per_sqft'],
      dtype='object')
```

7. NULL / MISSING VALUES IN EACH COLUMN

In [22]:
```
# DISPLAY NULL VALUES IN EACH COLUMN
print("Null values in each column:")
print(data.isnull().sum())
```

```
Null values in each column:
location         0
size             0
total_sqft       0
bath             0
price            0
bhk              0
price_per_sqft   0
dtype: int64
```

8. DUPLICATE VALUES

In [24]:
```
# FINDING THE TOTAL NO OF DUPLICATES
data.duplicated().sum()
```

Out[24]: 1049

In [26]:
```
data.shape
```

Out[26]: (13200, 7)

In [28]:
```
# TO REMOVE DUPLICATES
data.drop_duplicates(inplace=True)
```

In [30]:
```
data.shape
```

Out[30]: (12151, 7)

DETECTING OUTLIERS

1. MEAN AND STANDARD DEVIATION METHOD

In [ ]:
```
In this method, we assume that the data is normally distributed.
Any data point that is more than a certain number of standard deviations
away from the mean is considered an outlier.

Steps:
    1. Calculate the mean and standard deviation of the price_per_sqft.
    2. Define an acceptable range (typically, 3 standard deviations from the mean).
    3. Remove the rows that fall outside this range.
```

In [32]:
```
# Calculate mean and standard deviation
mean_price = data['price_per_sqft'].mean()
```

```
std_price = data['price_per_sqft'].std()

# Define threshold for outliers (e.g., 3 standard deviations)
threshold = 3

lower_limit = mean_price - threshold * std_price
upper_limit = mean_price + threshold * std_price

# Remove rows where price_per_sqft is beyond 3 standard deviations from the mean
data_trimmed_mean_std = data[(data['price_per_sqft'] >= lower_limit) &
                             (data['price_per_sqft'] <= upper_limit)]

# Check the number of rows removed
rows_removed=len(data) - len(data_trimmed_mean_std)
print(f"Number of rows removed: {rows_removed}")
```

```
Number of rows removed: 5
```

2. PERCENTILE METHOD

In [ ]:

```
This method involves using percentiles (typically the 5th and 95th percentiles) to
Any value below the 5th percentile or above the 95th percentile is considered an ou

Steps:
    1. Calculate the 5th and 95th percentiles for the price_per_sqft column.
    2. Remove the rows where price_per_sqft is less than the 5th percentile or grea
```

In [34]:

```
# Calculate 5th and 95th percentiles
lower_percentile = np.percentile(data['price_per_sqft'], 5)
upper_percentile = np.percentile(data['price_per_sqft'], 95)

# Remove rows outside the 5th and 95th percentiles
data_trimmed_percentiles = data[(data['price_per_sqft'] >= lower_percentile) &
                                (data['price_per_sqft'] <= upper_percentile)]

# Check the number of rows removed
rows_removed = len(data) - len(data_trimmed_percentiles)
print(f"Number of rows removed: {rows_removed}")

# You can check the new dataset if needed
data_trimmed_percentiles.head()
```

```
Number of rows removed: 1211
```

Out[34]:

	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
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250

3. IQR(INTER QUARTILE RANGE) METHOD

In [ ]:
```
IQR is the range between the 25th and 75th percentiles (Q1 and Q3). Outliers are de
below Q1−1.5×IQR or above Q3+1.5×IQR

Steps:
    1. Calculate Q1 (25th percentile) and Q3 (75th percentile).
    2. Calculate the IQR.
    3. Remove values outside the acceptable range.
```

In [36]:
```
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = data['price_per_sqft'].quantile(0.25)
Q3 = data['price_per_sqft'].quantile(0.75)

# Calculate IQR
IQR = Q3 - Q1

# Define the upper and lower bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove rows outside the IQR range
data_trimmed_IQR = data[(data['price_per_sqft'] >= lower_bound) &
                        (data['price_per_sqft'] <= upper_bound)]

# Check the number of rows removed
rows_removed = len(data) - len(data_trimmed_IQR)
print(f"Number of rows removed: {rows_removed}")
```

```
Number of rows removed: 1142
```

4. Z SCORE METHOD

In [ ]:
```
The Z-score is the number of standard deviations a data point is from the mean.
If the Z-score is greater than a certain threshold (typically 3), it is considered

Steps:
    1. Calculate the Z-score for each data point in the price_per_sqft column.
    2. Set a critical value (commonly 3) to define the threshold for outliers.
       Any Z-score greater than the critical value (3) or less than the negative c
       will be considered an outlier.
    3. Remove rows where the Z-score is greater than the critical value or less tha
```

In [38]:
```
from scipy.stats import zscore

# Calculate Z-scores for the 'price_per_sqft' column
data['z_score'] = zscore(data['price_per_sqft'])

# Define the critical value (3 is the common threshold)
critical_value = 3

# Remove rows where the Z-score is greater than 3 or less than -3
```

```
data_trimmed_z_score = data[(data['z_score'] > critical_value) | (data['z_score'] <

# Check the number of rows removed
rows_removed = len(data) - len(data_trimmed_z_score)
print(f"Number of rows removed: {rows_removed}")

# You can check the new dataset if needed
data_trimmed_z_score.head()
```

```
Number of rows removed: 12146
```

Out[38]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft	z_score
345	other	3 Bedroom	11.0	3.0	74.0	3	672727	5.975046
1106	other	5 Bedroom	24.0	2.0	150.0	5	625000	5.545956
4044	Sarjapur Road	4 Bedroom	1.0	4.0	120.0	4	12000000	107.813073
4924	other	7 BHK	5.0	7.0	115.0	7	2300000	20.605070
11447	Whitefield	4 Bedroom	60.0	4.0	218.0	4	363333	3.193434

In [40]:

```
data
```

Out[40]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft	z_score
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699	-0.039861
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615	-0.031625
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305	-0.034412
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245	-0.016971
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250	-0.034907
...	...	...	...	...	...	...	...	...
13194	Green Glen Layout	3 BHK	1715.0	3.0	112.00	3	6530	-0.014409
13195	Whitefield	5 Bedroom	3453.0	4.0	231.00	5	6689	-0.012979
13196	other	4 BHK	3600.0	5.0	400.00	4	11111	0.026777
13197	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2	5258	-0.025845
13198	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4	10407	0.020448

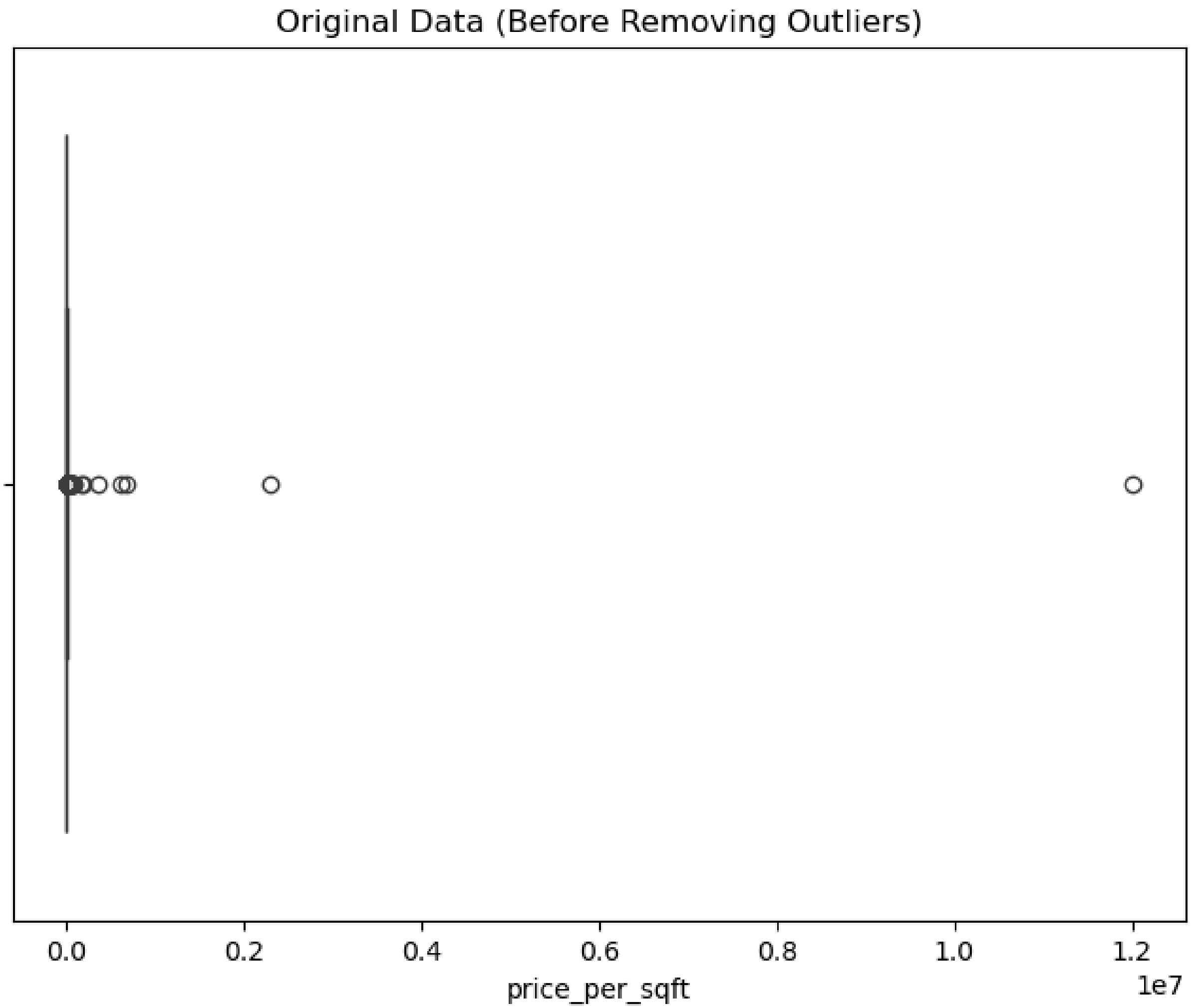
12151 rows × 8 columns

BOX PLOT VISUALIZATION

1. BOX PLOT OF ORIGINAL DATA

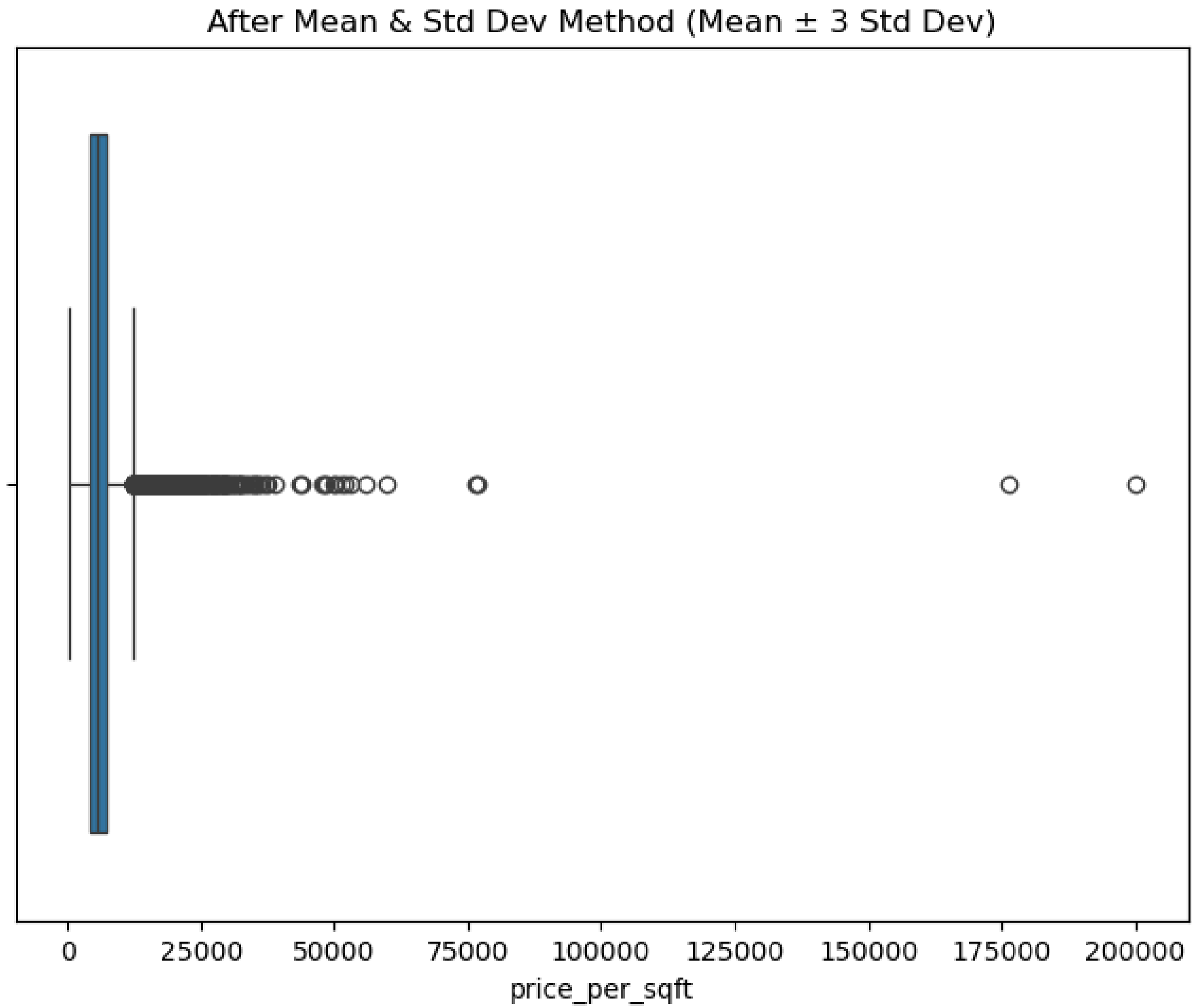
In [42]:

```
# Plotting the box plot for Original Data (before removing outliers)
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['price_per_sqft'])
plt.title('Original Data (Before Removing Outliers)')
plt.show()
```



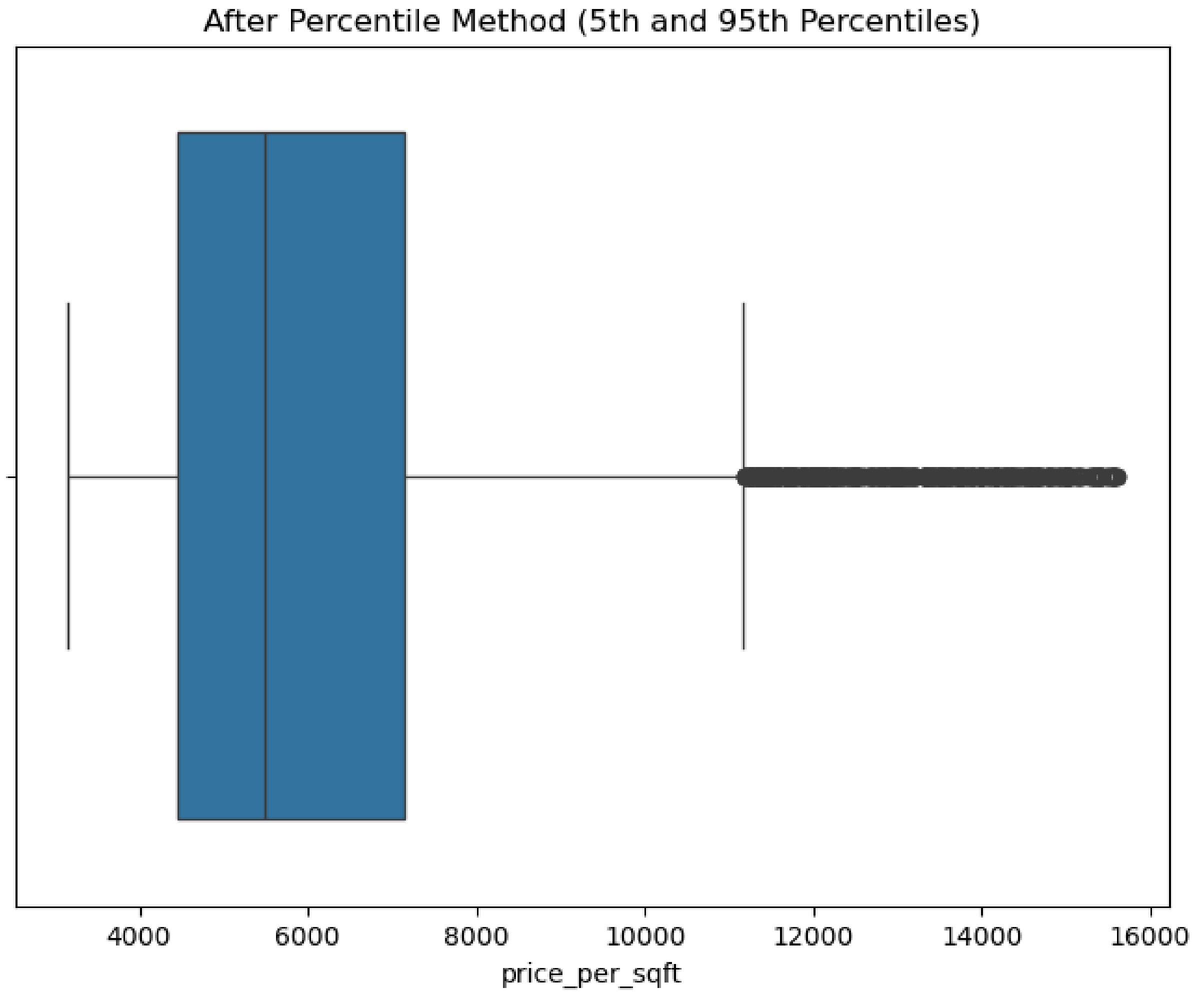
2. BOX PLOT FOR MEAN AND STANDARD DEVIATION METHOD

In [44]:
```
# Plotting the box plot for Mean and Standard Deviation Method (after removing outl
plt.figure(figsize=(8, 6))
sns.boxplot(x=data_trimmed_mean_std['price_per_sqft'])
plt.title('After Mean & Std Dev Method (Mean ± 3 Std Dev)')
plt.show()
```



3. BOX PLOT FOR PERCENTILE METHOD

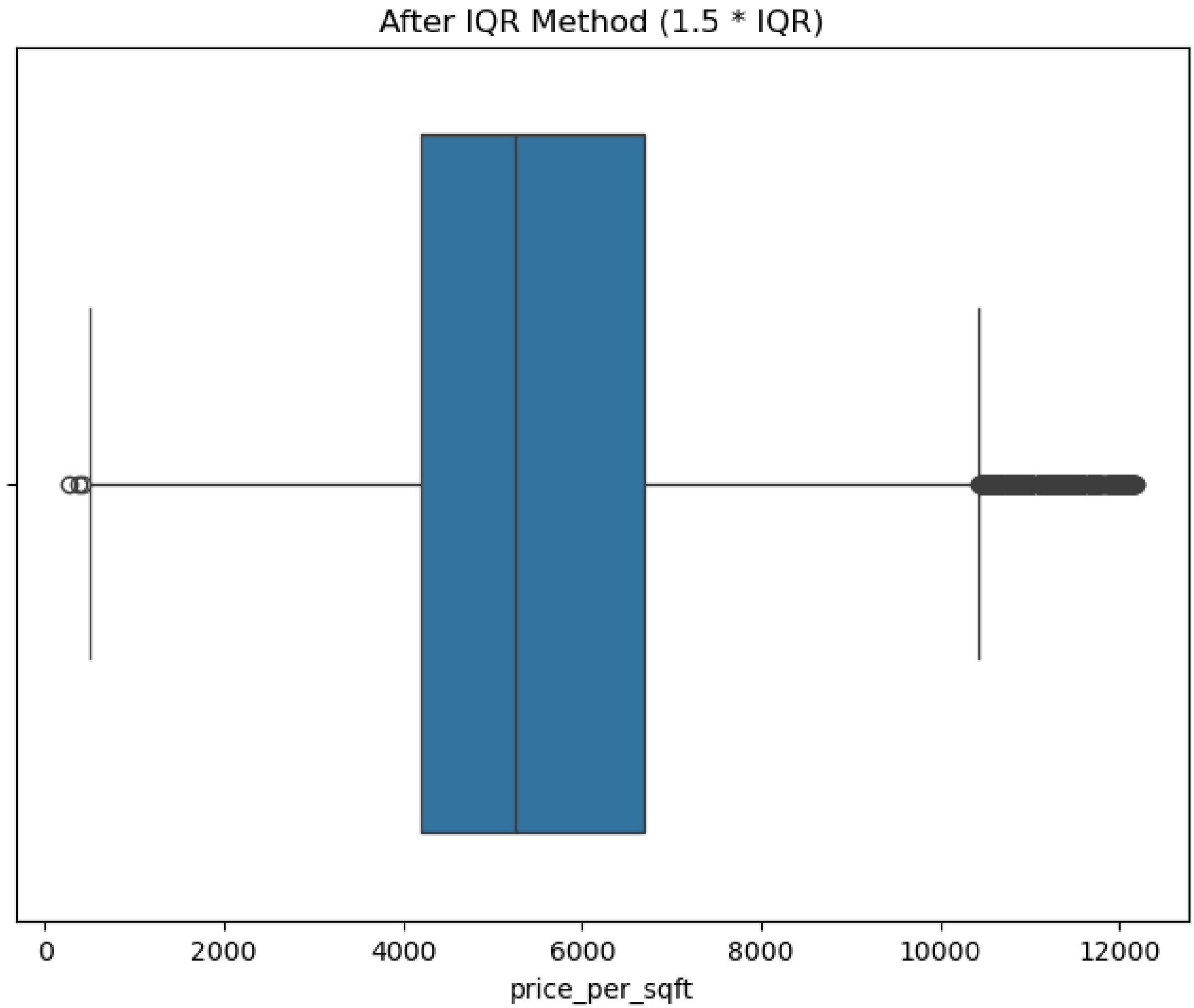
In [46]:
```
# Plotting the box plot for Percentile Method (after removing outliers using 5th an
plt.figure(figsize=(8, 6))
sns.boxplot(x=data_trimmed_percentiles['price_per_sqft'])
plt.title('After Percentile Method (5th and 95th Percentiles)')
plt.show()
```



4. BOX PLOT FOR IQR METHOD

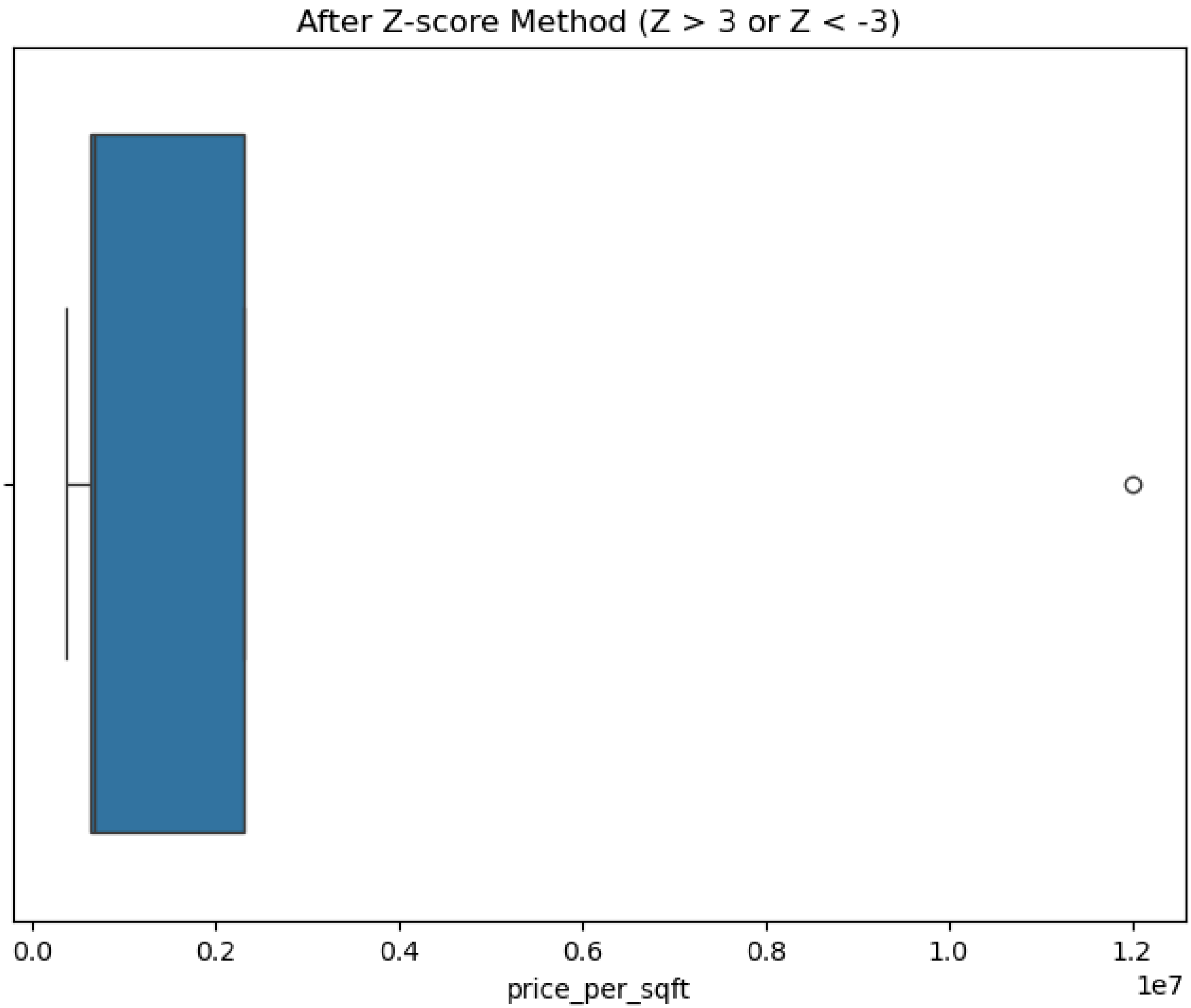
In [48]:

```
# Plotting the box plot for IQR Method (after removing outliers using IQR method)
plt.figure(figsize=(8, 6))
sns.boxplot(x=data_trimmed_IQR['price_per_sqft'])
plt.title('After IQR Method (1.5 * IQR)')
plt.show()
```



5. BOX PLOT FOR Z SCORE METHOD

In [50]:
```
# Plotting the box plot for Z-score Method (after removing outliers using Z-score m
plt.figure(figsize=(8, 6))
sns.boxplot(x=data_trimmed_z_score['price_per_sqft'])
plt.title('After Z-score Method (Z > 3 or Z < -3)')
plt.show()
```



IQR Method: This method offers a balanced approach to identifying and removing extreme outliers while preserving the core structure of the dataset.

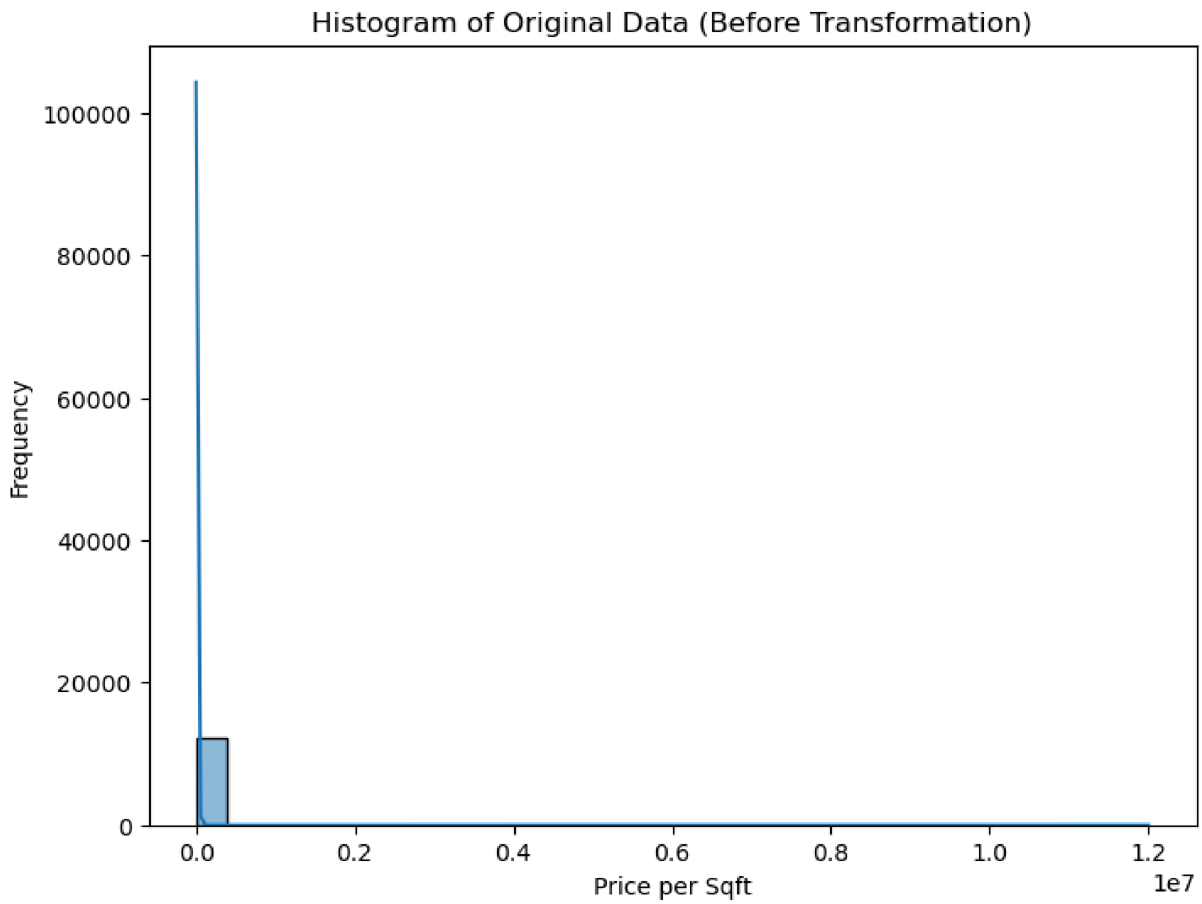
Percentile Method: A strong alternative, particularly when the goal is to retain a larger portion of the data by adjusting the percentiles used for outlier detection.

HISTOGRAM VISUALIZATION

In [52]:

```
# 1. Check the Normality of the Original Data (Before Transformation)
plt.figure(figsize=(8, 6))
sns.histplot(data['price_per_sqft'], kde=True, bins = 30)
plt.title('Histogram of Original Data (Before Transformation)')
```

```
plt.xlabel('Price per Sqft')
plt.ylabel('Frequency')
plt.show()
```



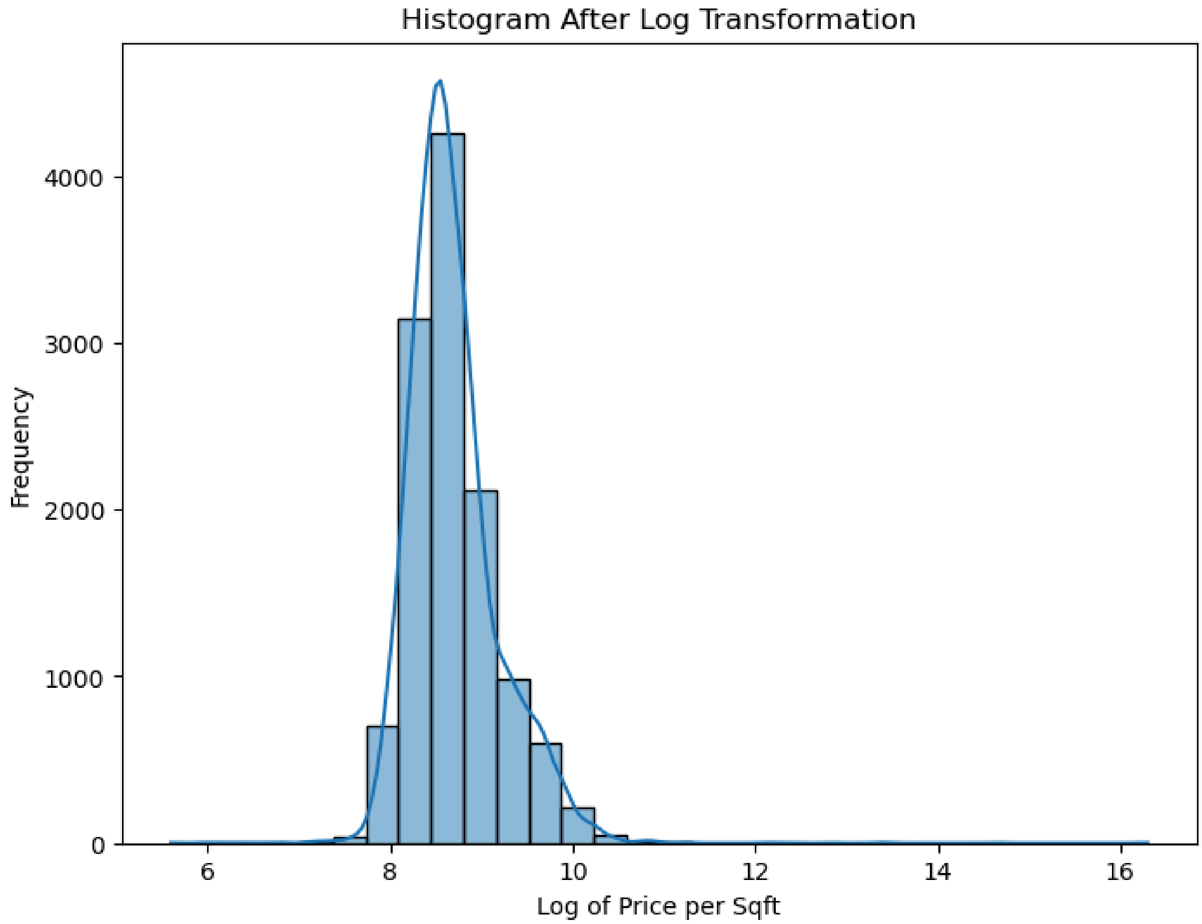
In [60]:
```
from scipy.stats import skew, kurtosis
# Calculate skewness and kurtosis for the original data
original_skewness = skew(data['price_per_sqft'])
original_kurtosis = kurtosis(data['price_per_sqft'])

print(f"Original Skewness: {original_skewness}")
print(f"Original Kurtosis: {original_kurtosis}")
```

```
Original Skewness: 103.88920549434178
Original Kurtosis: 11131.230839805388
```

In [62]:
```
# 2. Apply Log Transformation (For Right-Skewed Data)
data['price_per_sqft_log'] = np.log1p(data['price_per_sqft'])
```

In [64]:
```
# 3. Check the Normality of the Data After Log Transformation
plt.figure(figsize=(8, 6))
sns.histplot(data['price_per_sqft_log'], kde=True,bins=30)
plt.title('Histogram After Log Transformation')
plt.xlabel('Log of Price per Sqft')
plt.ylabel('Frequency')
plt.show()
```



In [66]:
```
# Calculate skewness and kurtosis after log transformation
log_skewness = skew(data['price_per_sqft_log'])
log_kurtosis = kurtosis(data['price_per_sqft_log'])

print(f"Log Transformation Skewness: {log_skewness}")
print(f"Log Transformation Kurtosis: {log_kurtosis}")
```

```
Log Transformation Skewness: 1.400870354404583
Log Transformation Kurtosis: 9.404434549652514
```

CORRELATION

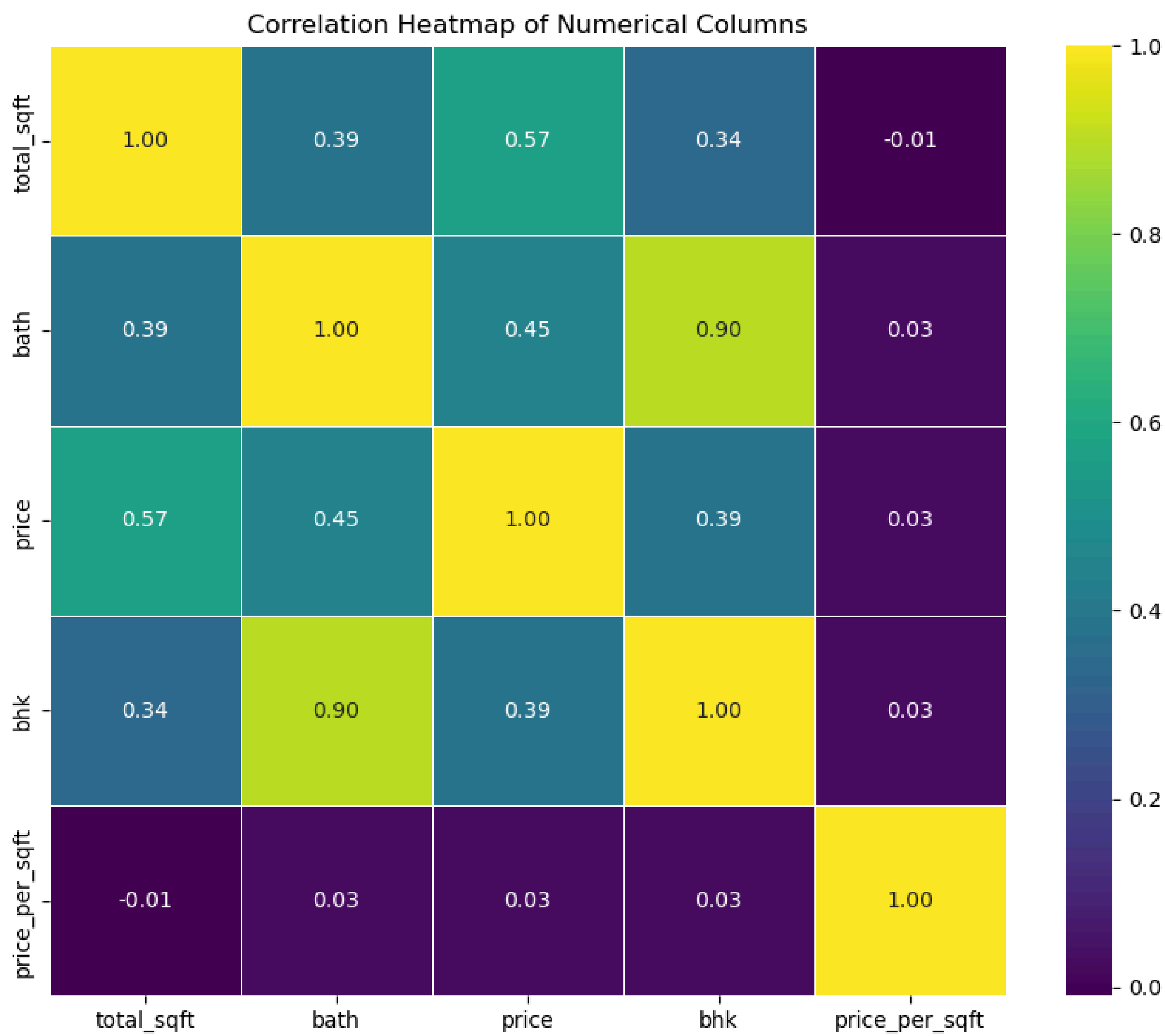
In [68]:
```
# 1. Select the numerical columns from the DataFrame
numerical_columns = data[['total_sqft', 'bath', 'price', 'bhk', 'price_per_sqft']]

# 2. Calculate the correlation matrix
correlation_matrix = numerical_columns.corr()
correlation_matrix
```

Out[68]:

	total_sqft	bath	price	bhk	price_per_sqft
total_sqft	1.000000	0.386694	0.572516	0.339936	-0.008877
bath	0.386694	1.000000	0.448802	0.898875	0.030133
price	0.572516	0.448802	1.000000	0.390008	0.027415
bhk	0.339936	0.898875	0.390008	1.000000	0.030294
price_per_sqft	-0.008877	0.030133	0.027415	0.030294	1.000000

In [70]:
```
# 3. Plot the heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='viridis', fmt='.2f', linewidths=0
plt.title('Correlation Heatmap of Numerical Columns')
plt.show()
```



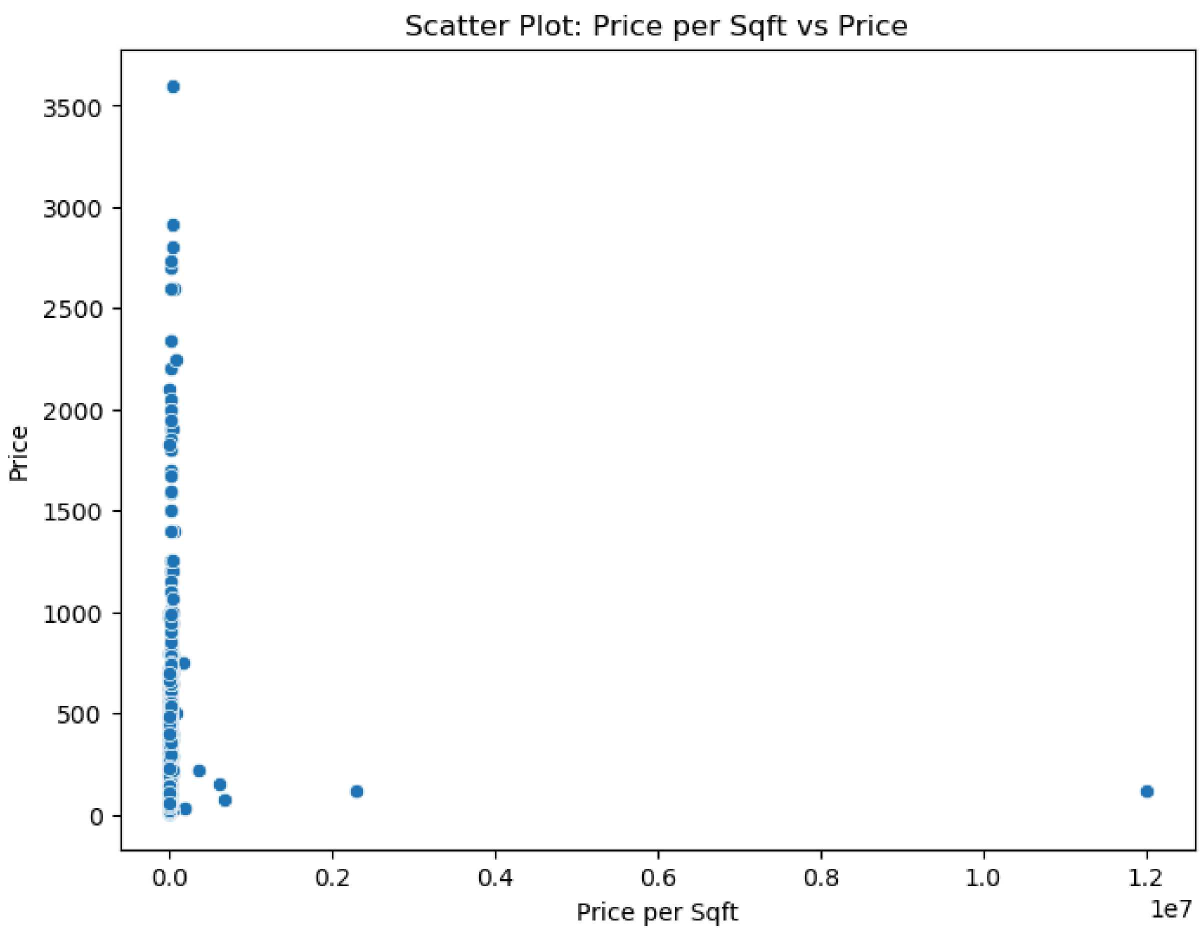
SCATTER PLOT VISUALIZATION

SCATTER PLOT BETWEEN DIFFERENT PAIRS OF NUMERICAL COLUMNS

1. price_per_sqft vs price

In [156…

```
# Plot: price_per_sqft vs price
plt.figure(figsize=(8, 6))
sns.scatterplot(x='price_per_sqft', y='price',data=data)
plt.title('Scatter Plot: Price per Sqft vs Price')
plt.xlabel('Price per Sqft')
plt.ylabel('Price')
plt.show()
```

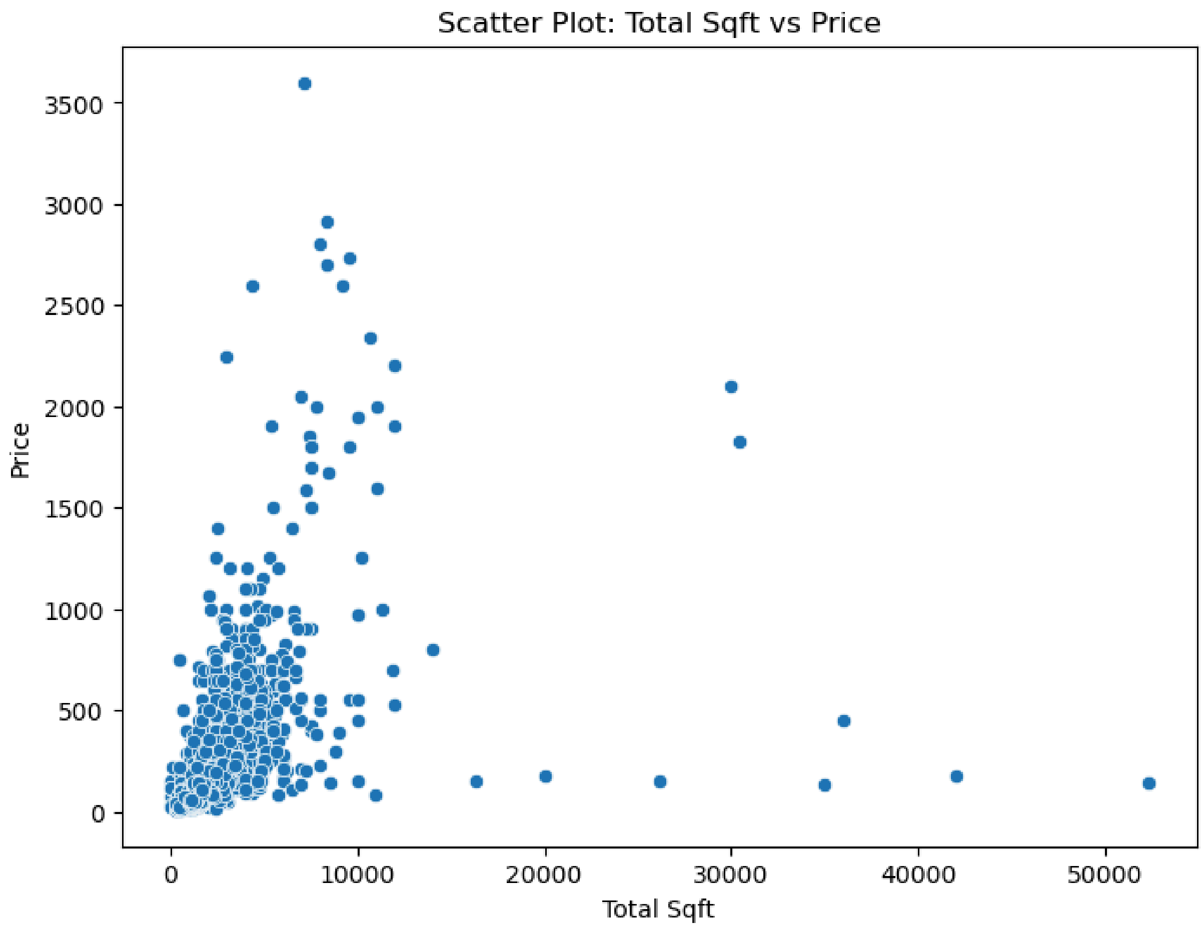


2. total_sqft vs price

In [158…

```
# Plot: total_sqft vs price
plt.figure(figsize=(8, 6))
sns.scatterplot(x='total_sqft', y='price', data=data)
plt.title('Scatter Plot: Total Sqft vs Price')
plt.xlabel('Total Sqft')
```

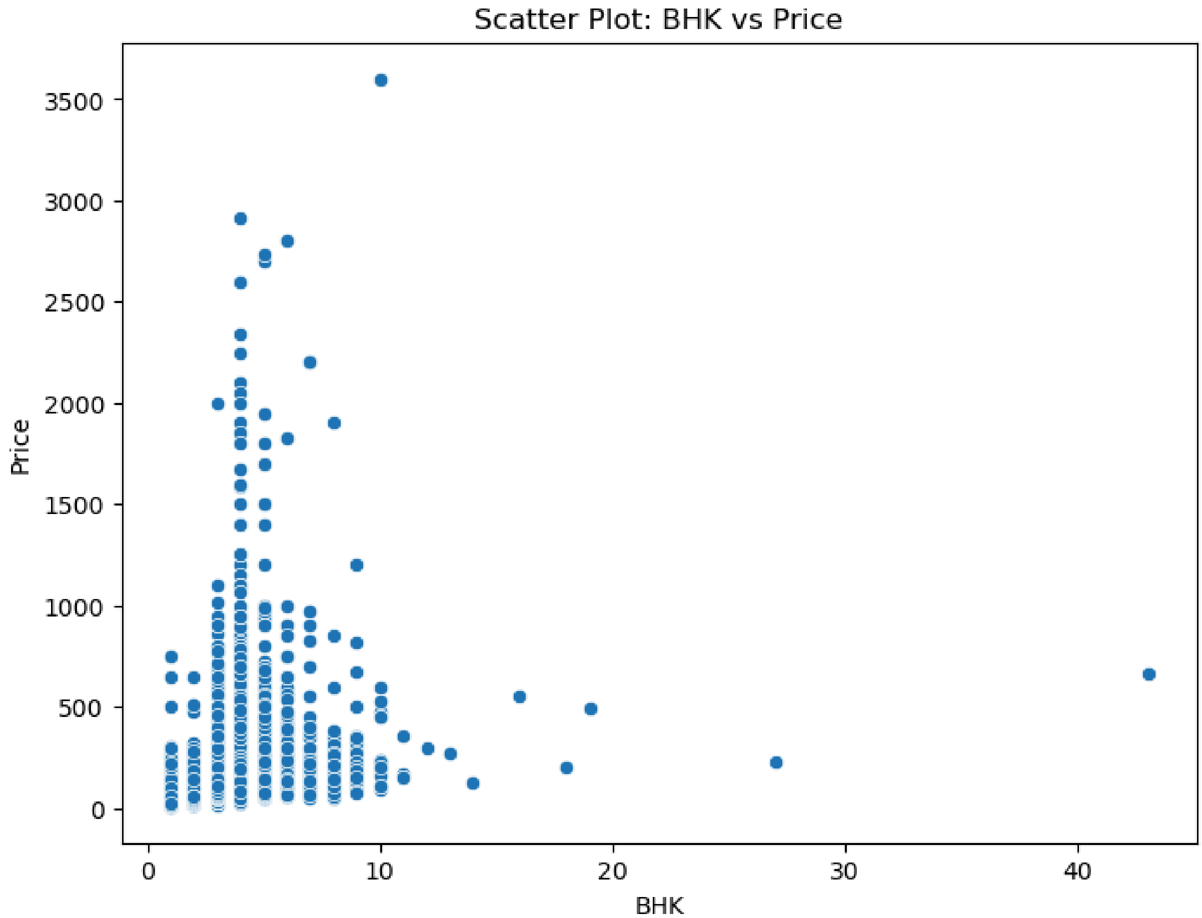
```
plt.ylabel('Price')
plt.show()
```



3. bhk vs price

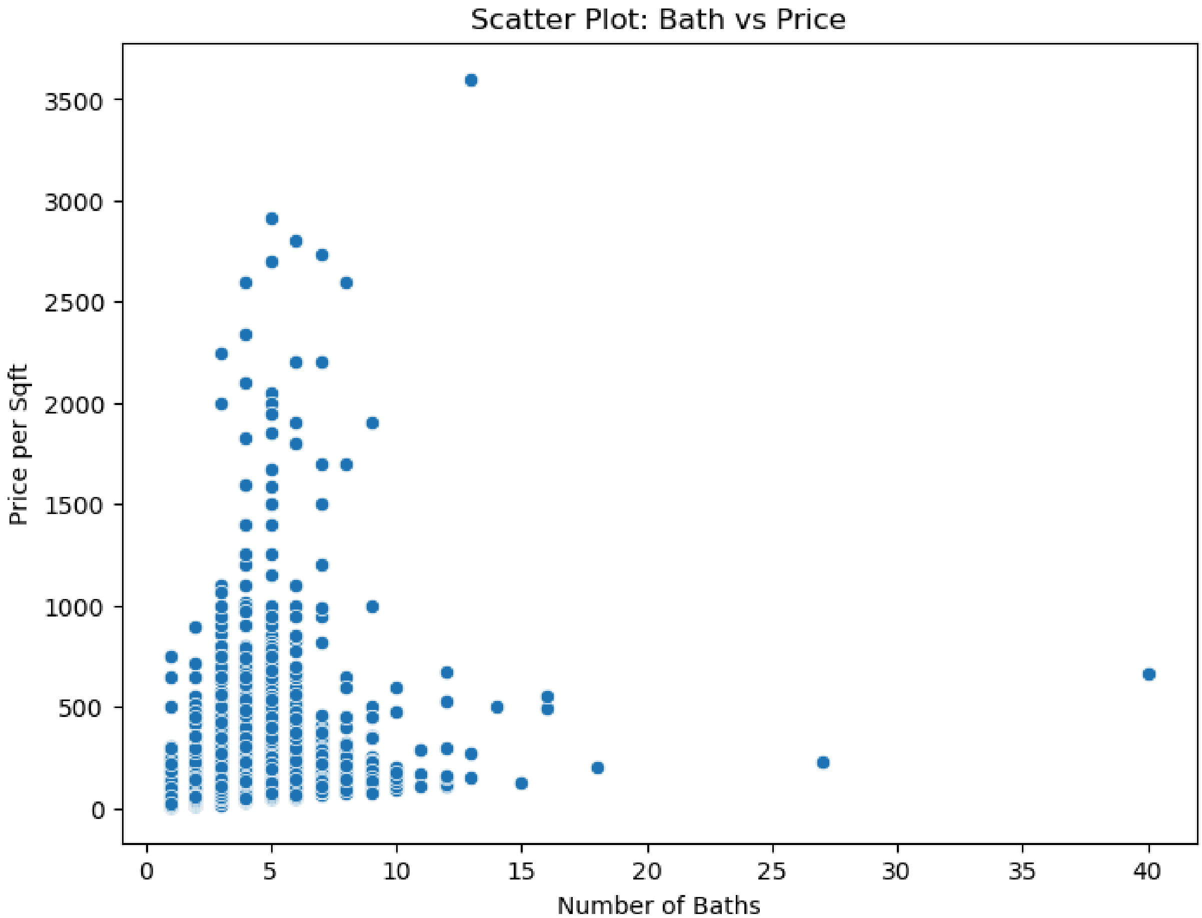
In [160…

```
# Plot: bhk vs price
plt.figure(figsize=(8, 6))
sns.scatterplot(x='bhk', y='price', data=data)
plt.title('Scatter Plot: BHK vs Price')
plt.xlabel('BHK')
plt.ylabel('Price')
plt.show()
```



4. bath vs price

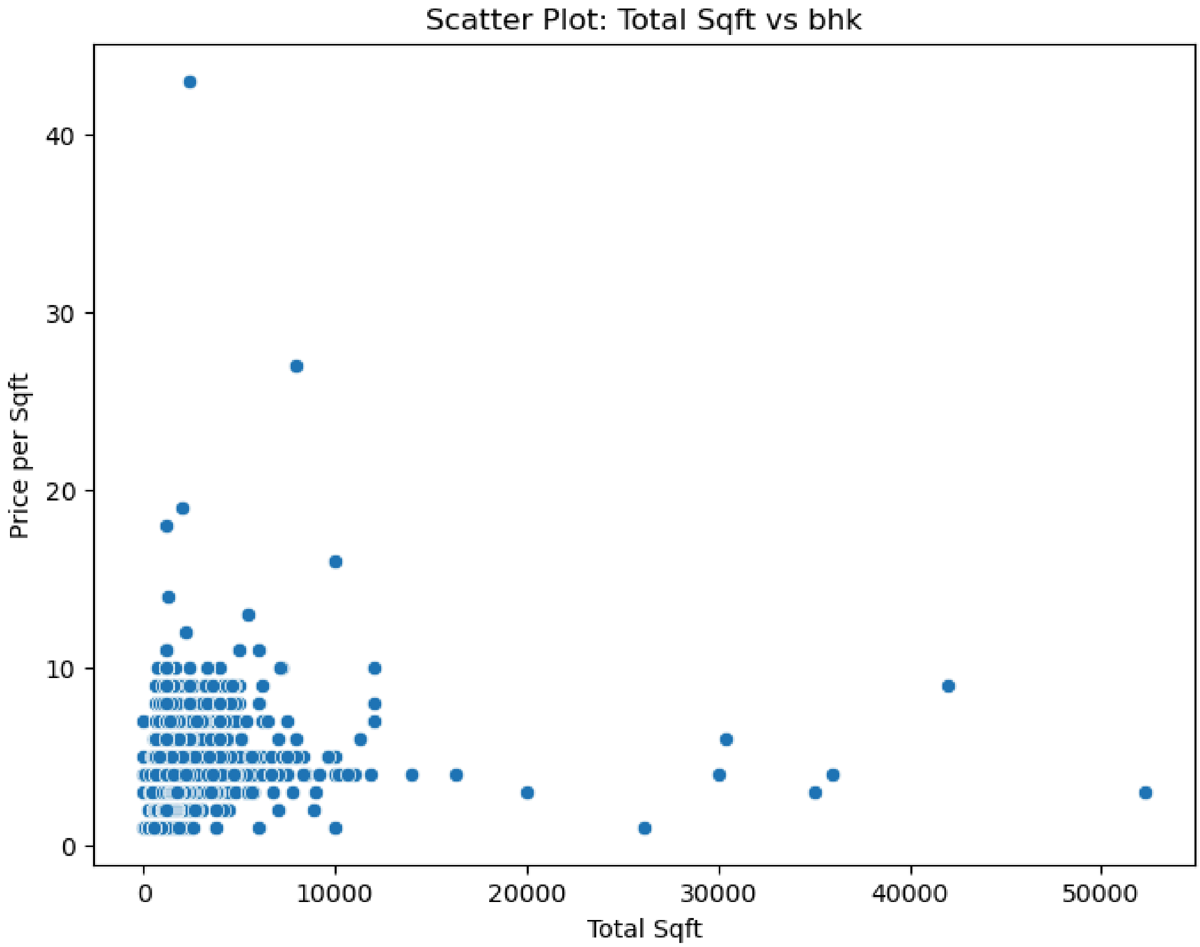
In [72]:
```
# Plot: bath vs price
plt.figure(figsize=(8, 6))
sns.scatterplot(x='bath', y='price', data=data)
plt.title('Scatter Plot: Bath vs Price')
plt.xlabel('Number of Baths')
plt.ylabel('Price per Sqft')
plt.show()
```



5. total_sqft vs bhk

In [75]:

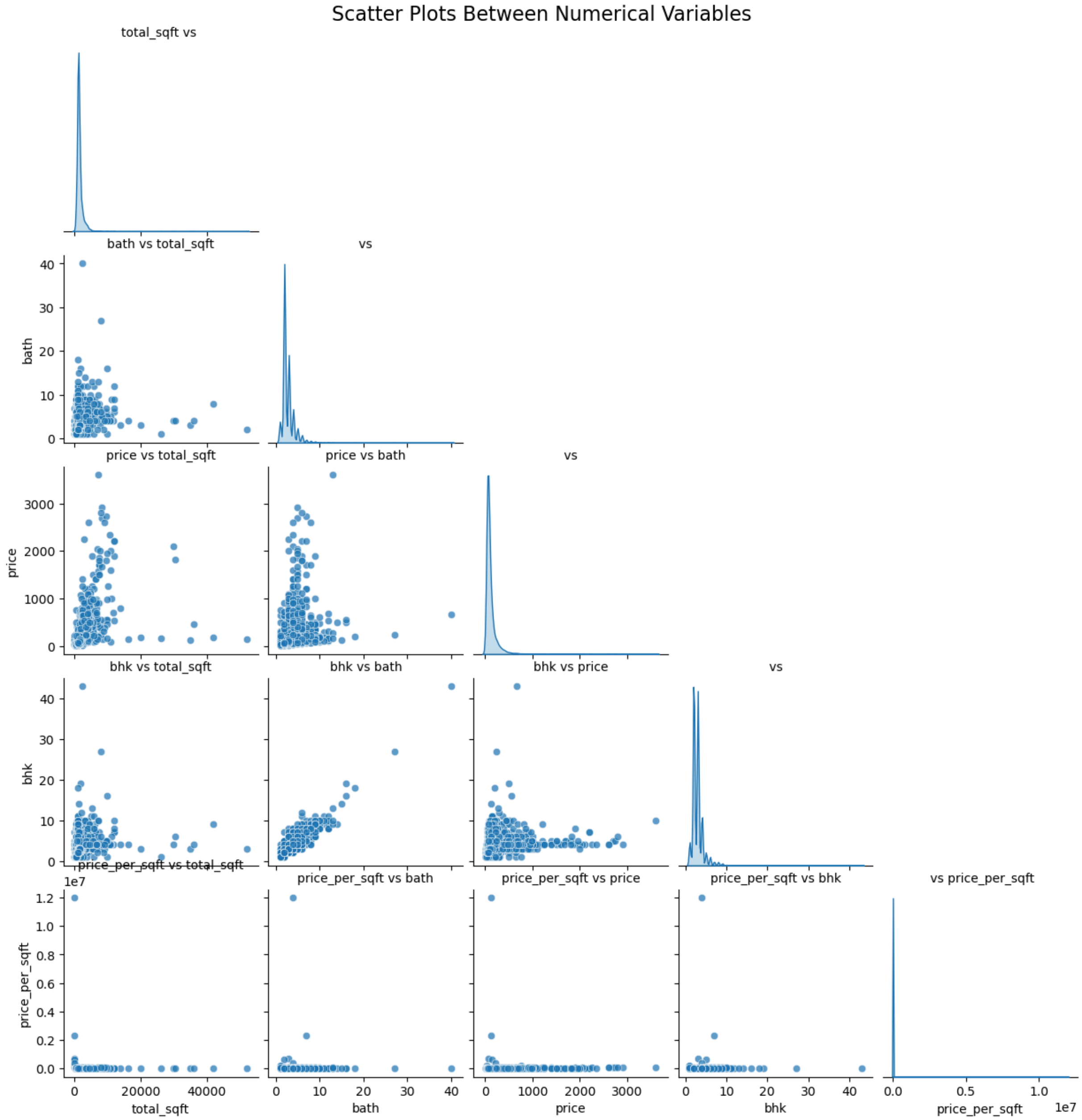
```
# Plot: total_sqft vs bhk
plt.figure(figsize=(8, 6))
sns.scatterplot(x='total_sqft', y='bhk', data=data)
plt.title('Scatter Plot: Total Sqft vs bhk')
plt.xlabel('Total Sqft')
plt.ylabel('Price per Sqft')
plt.show()
```



Scatter plots between numerical variables

In [81]:
```
# Select numerical columns only
num_columns = data[['total_sqft', 'bath', 'price', 'bhk', 'price_per_sqft']]

# Pairplot to create scatter plots between each pair of variables
pairplot = sns.pairplot(num_columns, diag_kind='kde', corner=True, plot_kws={'alpha
# Add labels and titles
for ax in pairplot.axes.flat:
    if ax:  # Check if the subplot exists (some may be None due to corner=True)
        ax.set_xlabel(ax.get_xlabel(), fontsize=10)
        ax.set_ylabel(ax.get_ylabel(), fontsize=10)
        ax.set_title(f'{ax.get_ylabel()} vs {ax.get_xlabel()}', fontsize=10, loc='c
plt.suptitle('Scatter Plots Between Numerical Variables', y=1.02, fontsize=16)
plt.show()
```


Scatter Plots Between Numerical Variables
total_sqft vs
bath vs total_sqft
vs
price vs total_sqft
price vs bath
vs
bhk vs total_sqft
bhk vs bath
bhk vs price
vs
price_per_sqft vs total_sqft
price_per_sqft vs bath
price_per_sqft vs price
price_per_sqft vs bhk
vs price_per_sqft
bath
price
bhk
price_per_sqft
1e7
total_sqft
bath
price
bhk
price_per_sqft
1e7

In [ ]:

In [ ]: