In [ ]:
```
#Q1. Perform basic EDA
```

In [1]:
```
pip install pandas numpy matplotlib seaborn scipy
```

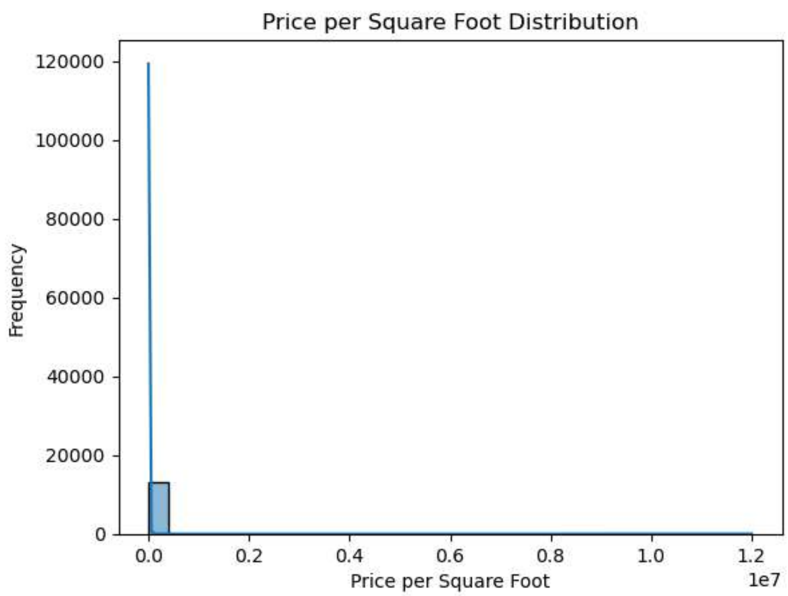
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (1.5.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (1.23.5)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (3.7.0)
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (1.10.0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (10.0.1)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (22.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.
16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
url = "https://drive.google.com/uc?id=1UlWRYU0UglE2ex3iFse0J6eCLEU8g98P"
data = pd.read_csv(url)

# Display the first few rows and basic info
print(data.head())
print(data.info())
```

```
                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

   price_per_sqft
0            3699
1            4615
2            4305
3            6245
4            4250
<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
None
```

In [3]:
```
# Summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Visualize the distribution of price per sqft
sns.histplot(data['price_per_sqft'], bins=30, kde=True)
plt.title('Price per Square Foot Distribution')
plt.xlabel('Price per Square Foot')
plt.ylabel('Frequency')
plt.show()
```

```
         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
location          0
size              0
total_sqft        0
bath              0
price             0
bhk               0
price_per_sqft    0
dtype: int64
```



In [ ]:
```
#Q2. Detect the outliers using following methods and remove it using methods like trimming / capping/ imputation using mean or me
```

In [ ]:
```
#a) Mean and Standard deviation
```

In [4]:
```
mean = data['price_per_sqft'].mean()
std_dev = data['price_per_sqft'].std()

# Define outlier threshold
lower_limit = mean - 3 * std_dev
upper_limit = mean + 3 * std_dev

# Remove outliers
data_mean_std = data[(data['price_per_sqft'] >= lower_limit) & (data['price_per_sqft'] <= upper_limit)]
```

In [ ]:
```
#b) Percentile Method
```

In [5]:
```
lower_percentile = data['price_per_sqft'].quantile(0.01)
upper_percentile = data['price_per_sqft'].quantile(0.99)

# Remove outliers
data_percentile = data[(data['price_per_sqft'] >= lower_percentile) & (data['price_per_sqft'] <= upper_percentile)]
```

In [ ]:
```
#c) IQR Method
```

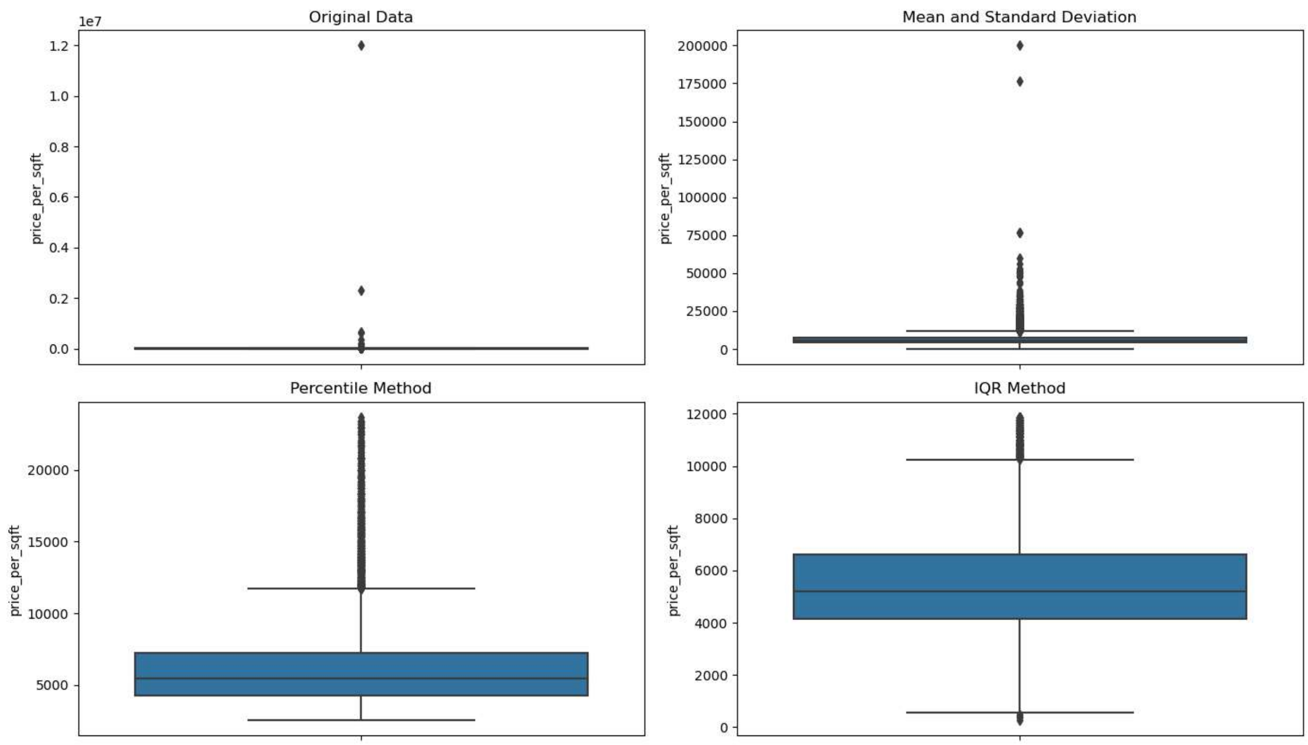
In [6]:
```
Q1 = data['price_per_sqft'].quantile(0.25)
Q3 = data['price_per_sqft'].quantile(0.75)
IQR = Q3 - Q1

# Define limits
lower_iqr = Q1 - 1.5 * IQR
upper_iqr = Q3 + 1.5 * IQR

# Remove outliers
data_iqr = data[(data['price_per_sqft'] >= lower_iqr) & (data['price_per_sqft'] <= upper_iqr)]
```

In [ ]:
```
#d) Z Score Method
```

In [7]:
```
from scipy import stats

z_scores = np.abs(stats.zscore(data['price_per_sqft']))
data_z = data[(z_scores < 3)]
```

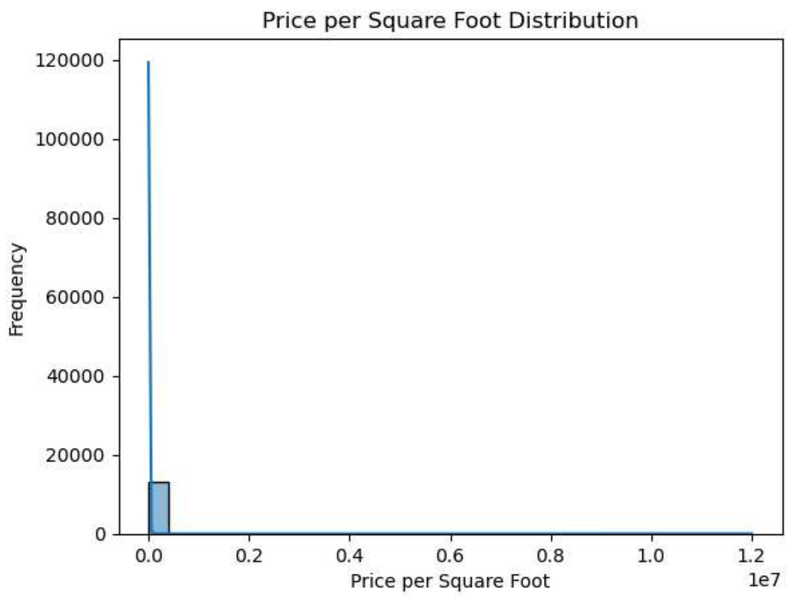
In [ ]:
```
#Q3. Create a box plot and use this to determine which method seems to work best to remove outliers for this data?
```

In [8]:
```
plt.figure(figsize=(14, 8))

plt.subplot(2, 2, 1)
sns.boxplot(y=data['price_per_sqft'])
plt.title('Original Data')

plt.subplot(2, 2, 2)
sns.boxplot(y=data_mean_std['price_per_sqft'])
plt.title('Mean and Standard Deviation')

plt.subplot(2, 2, 3)
sns.boxplot(y=data_percentile['price_per_sqft'])
plt.title('Percentile Method')

plt.subplot(2, 2, 4)
sns.boxplot(y=data_iqr['price_per_sqft'])
plt.title('IQR Method')

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



In [ ]:
```
. Draw histplot to check the normality of the column(price per sqft column) and perform transformations if needed. Check the skew
```

In [9]:
```python
# Histplot for normality check
sns.histplot(data['price_per_sqft'], bins=30, kde=True)
plt.title('Price per Square Foot Distribution')
plt.xlabel('Price per Square Foot')
plt.ylabel('Frequency')
plt.show()

# Skewness and Kurtosis
from scipy.stats import skew, kurtosis

print("Skewness:", skew(data['price_per_sqft']))
print("Kurtosis:", kurtosis(data['price_per_sqft']))

# Log transformation if needed
data['log_price_per_sqft'] = np.log(data['price_per_sqft'])

# Check skewness and kurtosis after transformation
print("Skewness after log transformation:", skew(data['log_price_per_sqft']))
print("Kurtosis after log transformation:", kurtosis(data['log_price_per_sqft']))
```

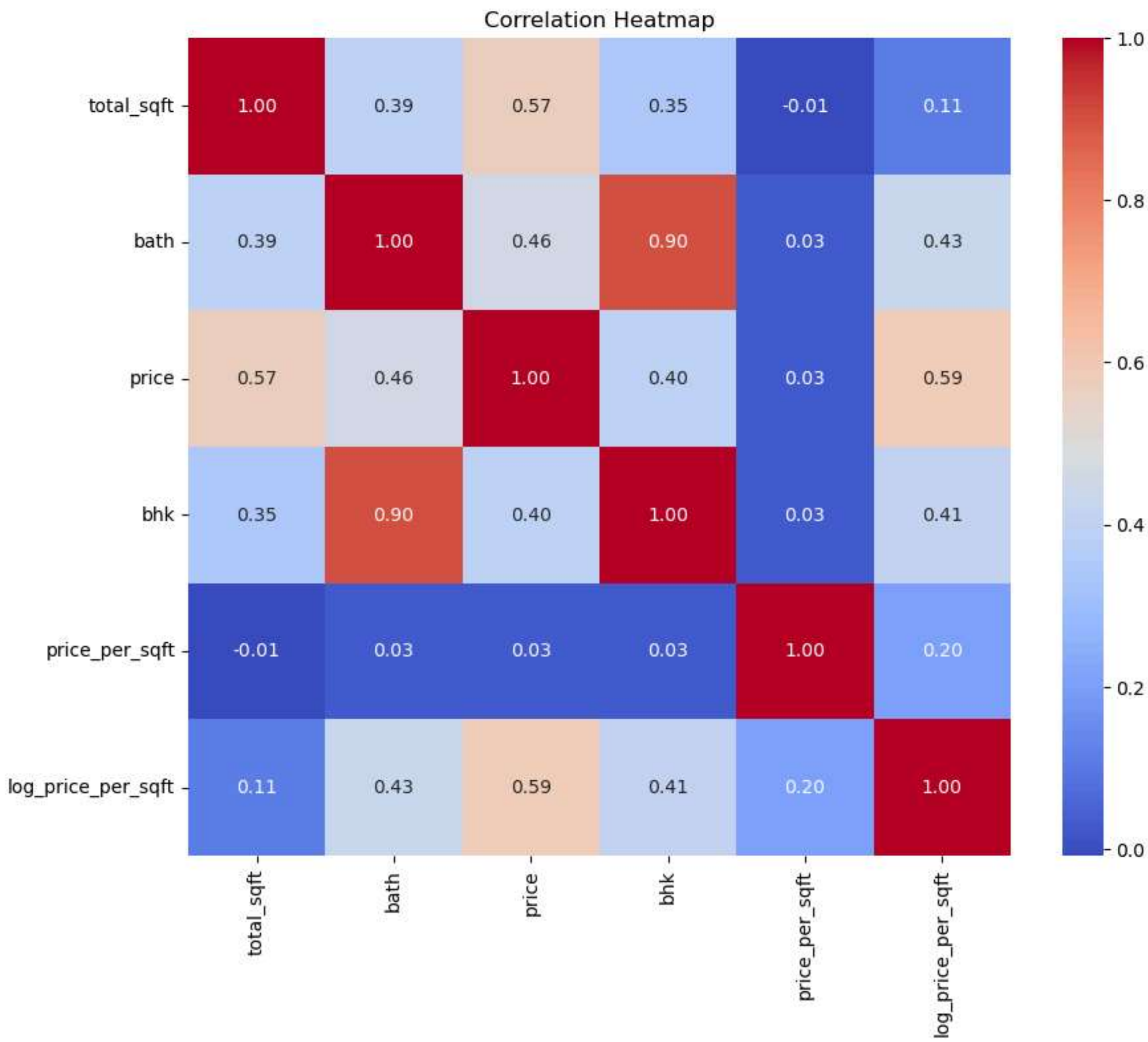



```
Skewness: 108.26875024325159
Kurtosis: 12090.633538860382
Skewness after log transformation: 1.3997035748119977
Kurtosis after log transformation: 9.199636085376468
```

In [ ]:
```python
#Q5. Check the correlation between all the numerical columns and plot heatmap.
```

In [10]:
```python
# Correlation matrix
correlation_matrix = data.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

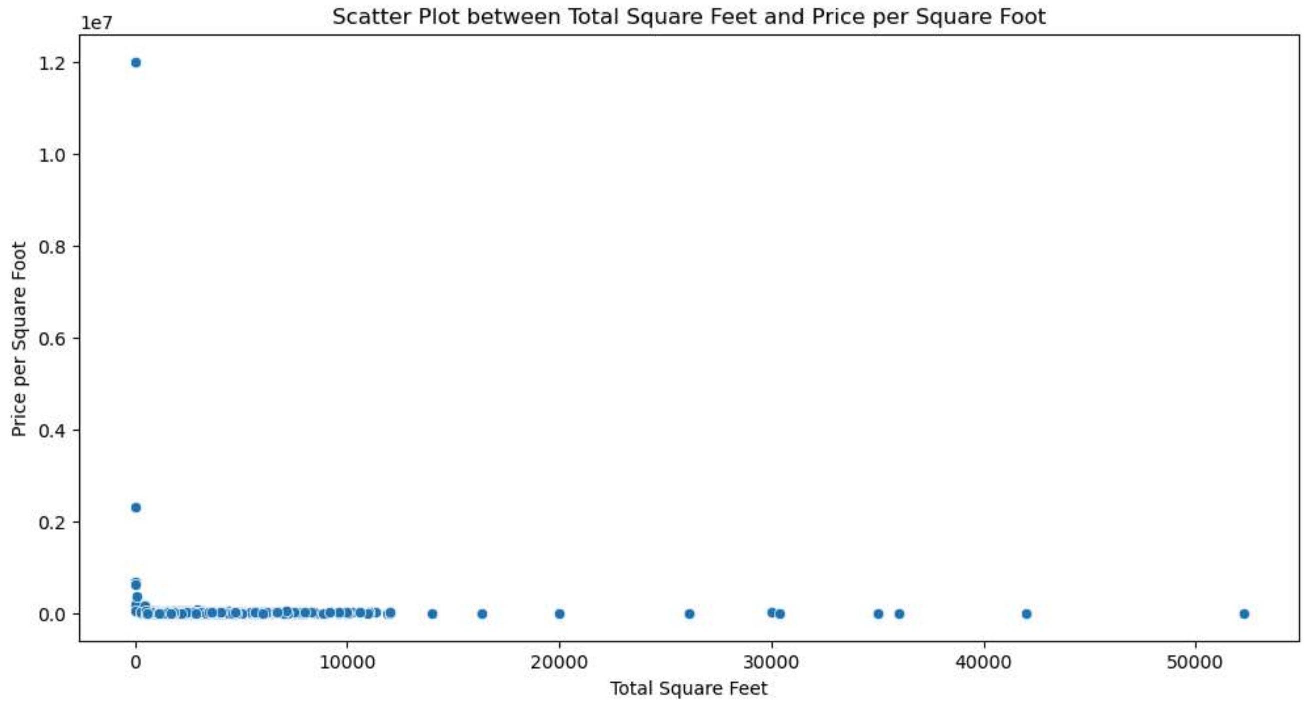
```
C:\Users\SHONIMA S\AppData\Local\Temp\ipykernel_18008\996203314.py:2: FutureWarning: The default value of numeric_only in DataF
rame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numer
ic_only to silence this warning.
  correlation_matrix = data.corr()
```



In [ ]:
```python
#Q6. Draw Scatter plot between the variables to check the correlation between them.
```

In [13]:

```
plt.figure(figsize=(12, 6))
sns.scatterplot(data=data, x='total_sqft', y='price_per_sqft')  # Replace 'total_sqft' with any relevant variable
plt.title('Scatter Plot between Total Square Feet and Price per Square Foot')
plt.xlabel('Total Square Feet')
plt.ylabel('Price per Square Foot')
plt.show()
```



In [ ]: