Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Mini-Project Report

Academic Year 2024-2025

Course: Python for Data Science Laboratory — Course Code: DJS23SLPC303

Class: S.Y.B.Tech. — Semester: III — Division: S

Department: Artificial Intelligence (AI) and Data Science — Batch: A1

Bengaluru Housing Price Analysis

Prepared by

Sr. No.	Roll No.	Name	SAP ID
1	S009	Chaitanya Shah	60018230034
2	S007	Bansari Naik	60018230089
3	S011	Dev Mittal	60018230102
4	S032	Krisha Shah	60018230090

Table of Contents

Sr. No.	Topic	Page No.
1	Project Overview	2
2	Tech Stack	3
3	Code & Output	3
4	Conclusion	13
5	References	17

1 Project Overview

The Bengaluru Housing Price Analysis project is an in-depth study of the real estate market in Bengaluru, India. The city, known as the IT capital of India, has seen rapid urbanization and infrastructural growth, making it a hub for real estate activity. This project focuses on analyzing a dataset containing detailed information about over 13,000 housing units, including attributes such as location, size, and total area, number of bathrooms and balconies, and price. The primary goal is to uncover patterns, trends, and relationships within the data to better understand the factors influencing property prices.

The dataset is a rich source of information for conducting exploratory data analysis (EDA) and applying data visualization techniques to reveal meaningful insights. It includes diverse property types such as apartments, villas, and plots, with a variety of configurations ranging from single-bedroom units to large family homes. The analysis begins with cleaning and preprocessing the data, addressing issues like missing values, inconsistent formats, and outliers. For instance, the `total_sqft` column often contains ranges or textual information, which is converted into numerical values to enable precise analysis.

Visualization plays a key role in this project. Charts and graphs, such as scatter plots, histograms, and bar plots, are used to highlight the distribution of prices, the correlation between features, and the popularity of different locations. For example, a scatter plot of price versus area reveals a general positive correlation, while outliers demonstrate the premium impact of location on pricing. Similarly, a bar chart showing the most listed localities highlights areas like Whitefield, Sarjapur Road, and Electronic City as the city’s real estate hotspots, driven by their proximity to IT corridors and infrastructure.

Approach for the project:

1. Data Cleaning: Address missing values, inconsistent formats, and outliers.
2. Exploratory Data Analysis (EDA): Analyze relationships among variables using statistical summaries and visualizations.
3. Visualization: Highlight significant patterns in housing price trends and correlations between features.

2 Tech Stack

- **Python:** Programming language used for data analysis and data visualization libraries of Bengaluru Housing dataset.
- **Python Libraries:**
  - **Pandas:** To preprocess and analyze the dataset, including handling missing values and filtering data.
  - **NumPy:** For efficient numerical operations, such as data type conversions and mathematical computations.
  - **Matplotlib & Seaborn:** For creating detailed visualizations like histograms, scatter plots, and bar plots.
- **Dataset:** Used available dataset from Kaggle. (Bengaluru_House_Data.csv)
- **Dataset Link:** https://www.kaggle.com/datasets/amitabhajoy/bengaluru-house-price-data/data

3 Code & Output

Import necessary Python Libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Load the dataset:

```
file_path = 'Bengaluru_House_Data.csv'
df = pd.read_csv(file_path)
```

Display few rows of dataset:

```
df.head()
```

	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

```python
df.tail()
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.0
13316	Super built-up Area	Ready To Move	Richards Town	4 BHK	NaN	3600	5.0	NaN	400.0
13317	Built-up Area	Ready To Move	Raja Rajeshwari Nagar	2 BHK	Mahla T	1141	2.0	1.0	60.0
13318	Super built-up Area	18-Jun	Padmanabhanagar	4 BHK	SollyCl	4689	4.0	1.0	488.0
13319	Super built-up Area	Ready To Move	Doddathoguru	1 BHK	NaN	550	1.0	1.0	17.0

Data Preprocessing:

```python
df.shape
```

```
(13320, 9)
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   society       7818 non-null   object 
 5   total_sqft    13320 non-null  object 
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

```python
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
bath	13247.0	2.692610	1.341458	1.0	2.0	2.0	3.0	40.0
balcony	12711.0	1.584376	0.817263	0.0	1.0	2.0	2.0	3.0
price	13320.0	112.565627	148.971674	8.0	50.0	72.0	120.0	3600.0

```python
df.isnull().sum()
```

	0
area_type	0
availability	0
location	1
size	16
society	5502
total_sqft	0
bath	73
balcony	609
price	0

dtype: int64

```
df.duplicated().sum()
```

```
529
```

Handling missing values and irrelevant columns...:

```
df = df.drop(columns=['society', 'balcony'])
df['location'] = df['location'].fillna(df['location'].mode()[0])
df = df.dropna(subset=['size', 'bath'])
```

Convert 'size' to a numeric value representing the number of bedrooms:

```
df['size'] = df['size'].apply(lambda x: int(x.split(' ')[0]))
```

Identify non-numeric values in 'total_sqft':

```
non_numeric_sqft = df[~df['total_sqft'].apply(lambda x:
str(x).replace('.', '', 1).isdigit())]['total_sqft'].unique()
print("Non-numeric values in 'total_sqft':", non_numeric_sqft)
```

```
Non-numeric values in 'total_sqft': ['2100 - 2850' '3067 - 8156' '1042 - 1105' '1145 - 1340' '1015 - 1540'
 '34.46Sq. Meter' '1195 - 1440' '4125Perch' '1120 - 1145' '3090 - 5002'
 '1160 - 1195' '1000Sq. Meter' '1115 - 1130' '1100Sq. Yards' '520 - 645'
 '1000 - 1285' '650 - 665' '633 - 666' '5.31Acres' '30Acres' '1445 - 1455'
 '884 - 1116' '850 - 1093' '716Sq. Meter' '547.34 - 827.31' '580 - 650'
 '3425 - 3435' '1804 - 2273' '3630 - 3800' '4000 - 5249' '1500Sq. Meter'
 '142.61Sq. Meter' '1574Sq. Yards' '1250 - 1305' '670 - 980'
 '1005.03 - 1252.49' '1004 - 1204' '361.33Sq. Yards' '645 - 936'
 '2710 - 3360' '2830 - 2882' '596 - 804' '1255 - 1863' '1300 - 1405'
 '117Sq. Yards' '934 - 1437' '980 - 1030' '2249.81 - 4112.19'
 '1070 - 1315' '3040Sq. Meter' '500Sq. Yards' '2806 - 3019' '613 - 648'
 '704 - 730' '1210 - 1477' '3369 - 3464' '1125 - 1500' '167Sq. Meter'
 '1076 - 1199' '381 - 535' '524 - 894' '540 - 670' '315Sq. Yards'
 '2725 - 3250' '888 - 1290' '660 - 700' '385 - 440' '770 - 841' '3Cents'
 '188.89Sq. Yards' '1469 - 1766' '204Sq. Meter' '1255 - 1350' '870 - 1080'
 '45Sq. Yards' '133.3Sq. Yards' '2580 - 2591' '2563 - 2733' '605 - 624'
 '1349 - 3324' '78.03Sq. Meter' '3300 - 3335' '1180 - 1630' '1365 - 1700'
 '122Sq. Yards' '84.53Sq. Meter' '2.09Acres' '981 - 1249' '1565 - 1595'
 '24Guntha' '1270 - 1275' '840 - 1010' '697Sq. Meter' '655 - 742'
 '1408 - 1455' '942 - 1117' '598 - 958' '1500Cents' '132Sq. Yards'
 '1010 - 1300' '2Acres' '1450 - 1950' '1100Sq. Meter' '15Acres'
 '763 - 805' '3307 - 3464' '1.26Acres' '620 - 934' '2462 - 2467'
 '540 - 740' '3508 - 4201' '4900 - 4940' '755 - 770' '664 - 722'
 '151.11Sq. Yards' '596 - 861' '615 - 985' '540 - 565' '750 - 800'
 '1660 - 1805' '1079 - 1183' '2800 - 2870' '1230 - 1290' '943 - 1220'
 '2041 - 2090' '527 - 639' '1Grounds' '1160 - 1315' '706 - 716'
 '2940Sq. Yards' '45.06Sq. Meter' '799 - 803' '2470 - 2790' '783 - 943'
 '4500 - 5540' '1255 - 1375' '610 - 615' '854 - 960' '2650 - 2990'
 '1.25Acres' '86.72Sq. Meter' '1230 - 1490' '660 - 780' '1150 - 1194'
 '684 - 810' '1510 - 1670' '1550 - 1590' '1235 - 1410' '38Guntha'
 '929 - 1078' '2150 - 2225' '1520 - 1759' '629 - 1026' '1215 - 1495'
 '6Acres' '1140 - 1250' '2400 - 2600' '1052 - 1322' '5666 - 5669'
 '712 - 938' '1783 - 1878' '120Sq. Yards' '24Sq. Meter' '2528 - 3188'
 '650 - 760' '1400 - 1421' '4000 - 4450' '142.84Sq. Meter' '300Sq. Yards'
 '1437 - 1629' '850 - 1060' '1200 - 1470' '1133 - 1384']
```

```python
def convert_sqft_to_num(x):
    try:
        if '-' in x:
            nums = x.split('-')
            return (float(nums[0]) + float(nums[1])) / 2
        return float(x)
    except:
        return np.nan

df['total_sqft'] = df['total_sqft'].apply(convert_sqft_to_num)
df = df.dropna(subset=['total_sqft'])
```

Ensure all relevant columns are numeric:

```python
df['price'] = pd.to_numeric(df['price'], errors='coerce')
df['bath'] = pd.to_numeric(df['bath'], errors='coerce')
```

Drop any remaining rows with NaN values in these columns:

```
df = df.dropna(subset=['price', 'bath', 'total_sqft', 'size'])
```

Cleaned dataset structure:

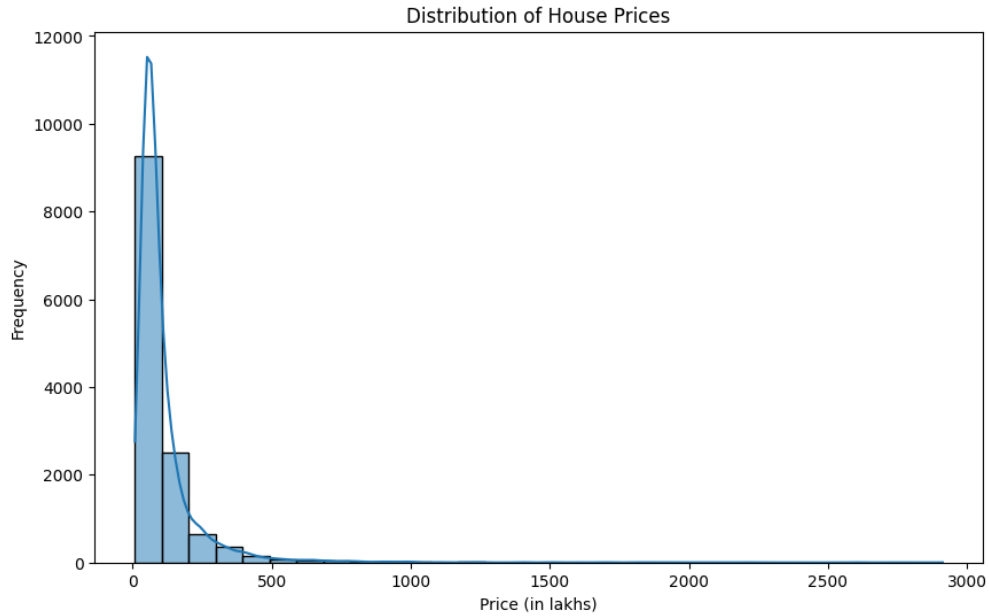
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 13201 entries, 0 to 13319
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13201 non-null  object 
 1   availability  13201 non-null  object 
 2   location      13201 non-null  object 
 3   size          13201 non-null  int64  
 4   total_sqft    13201 non-null  float64
 5   bath          13201 non-null  float64
 6   price         13201 non-null  float64
dtypes: float64(3), int64(1), object(3)
memory usage: 825.1+ KB
```

Data Visualization:

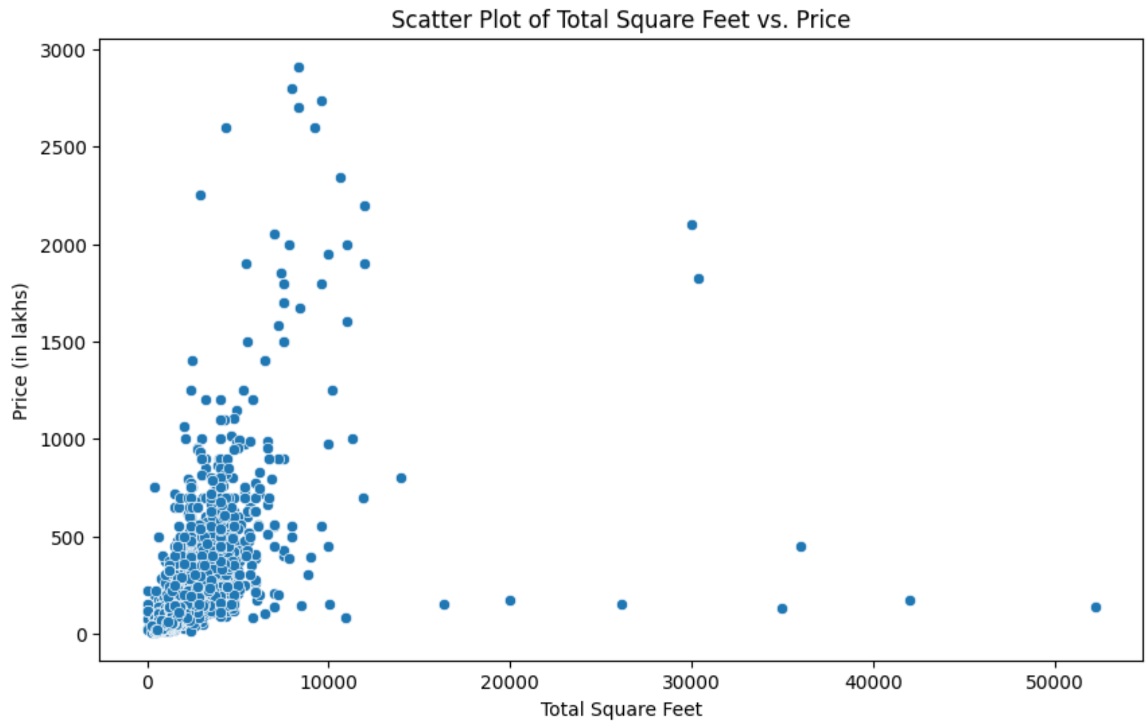
Distribution of prices

```
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], kde=True, bins=30)
plt.title('Distribution of House Prices')
plt.xlabel('Price (in lakhs)')
plt.ylabel('Frequency')
plt.show()
```



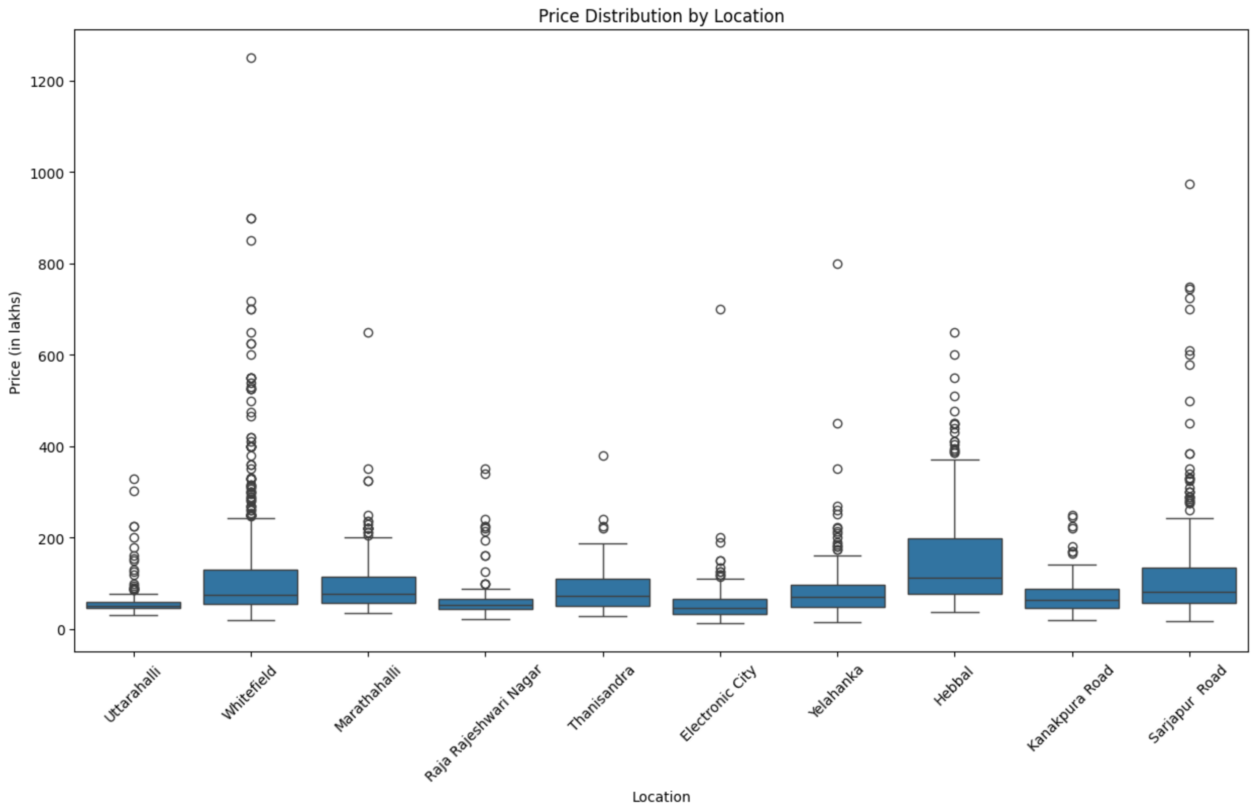
Scatter plot of total_sqft vs. price

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='total_sqft', y='price', data=df)
plt.title('Scatter Plot of Total Square Feet vs. Price')
plt.xlabel('Total Square Feet')
plt.ylabel('Price (in lakhs)')
plt.show()
```



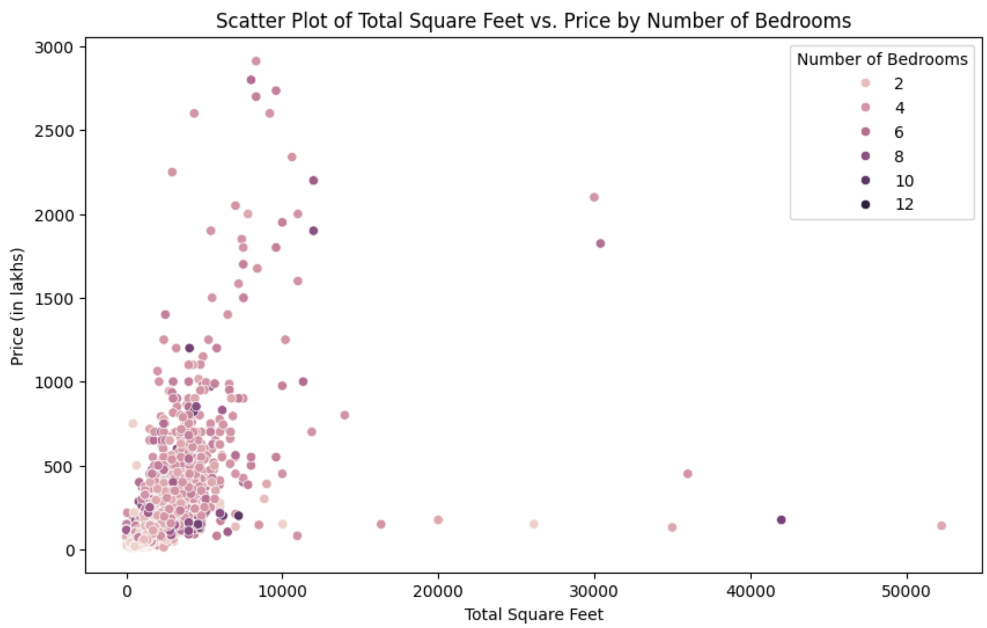
Price Distribution by Location

```
plt.figure(figsize=(15, 8))
top_locations = df['location'].value_counts().nlargest(10).index
sns.boxplot(x='location', y='price',
data=df[df['location'].isin(top_locations)])
plt.title('Price Distribution by Location')
plt.xlabel('Location')
plt.ylabel('Price (in lakhs)')
plt.xticks(rotation=45)
plt.show()
```



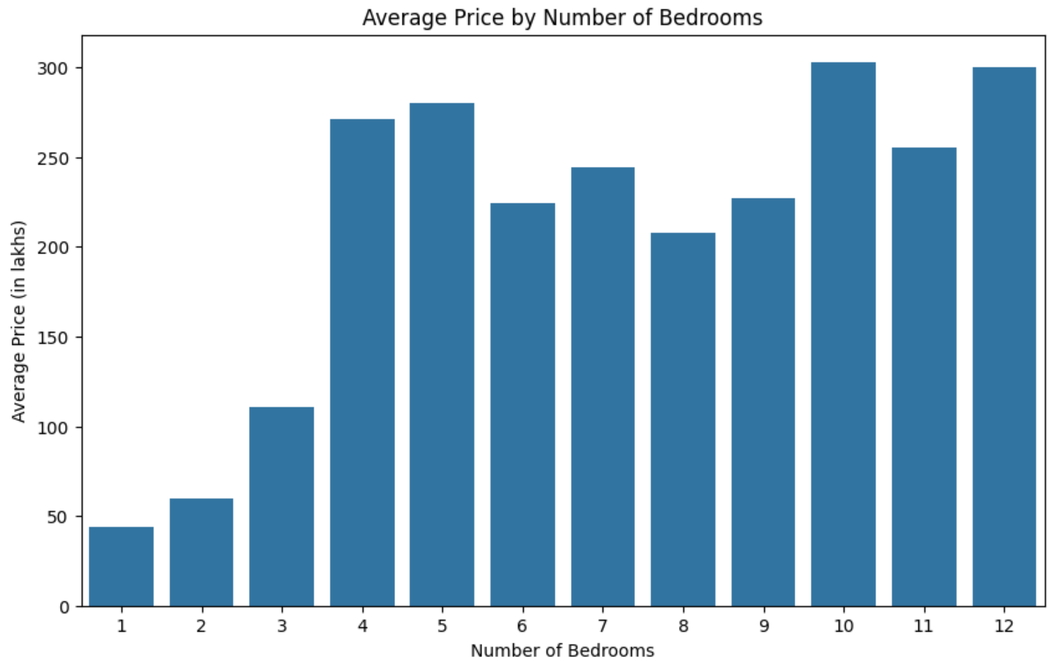
Scatter Plot of Total Square Feet vs. Price by Number of Bedrooms

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='total_sqft', y='price', hue='size', data=df)
plt.title('Scatter Plot of Total Square Feet vs. Price by Number
of Bedrooms')
plt.xlabel('Total Square Feet')
plt.ylabel('Price (in lakhs)')
plt.legend(title='Number of Bedrooms')
plt.show()
```



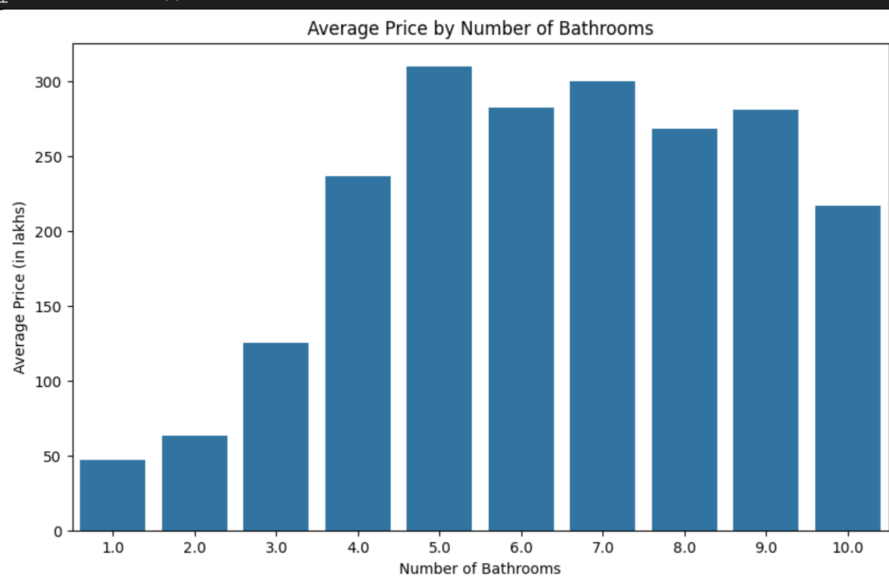
Bar plot of average price by number of bedrooms

```
plt.figure(figsize=(10, 6))
avg_price_by_size =
df.groupby('size')['price'].mean().reset_index()
sns.barplot(x='size', y='price', data=avg_price_by_size)
plt.title('Average Price by Number of Bedrooms')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Average Price (in lakhs)')
plt.show()
```



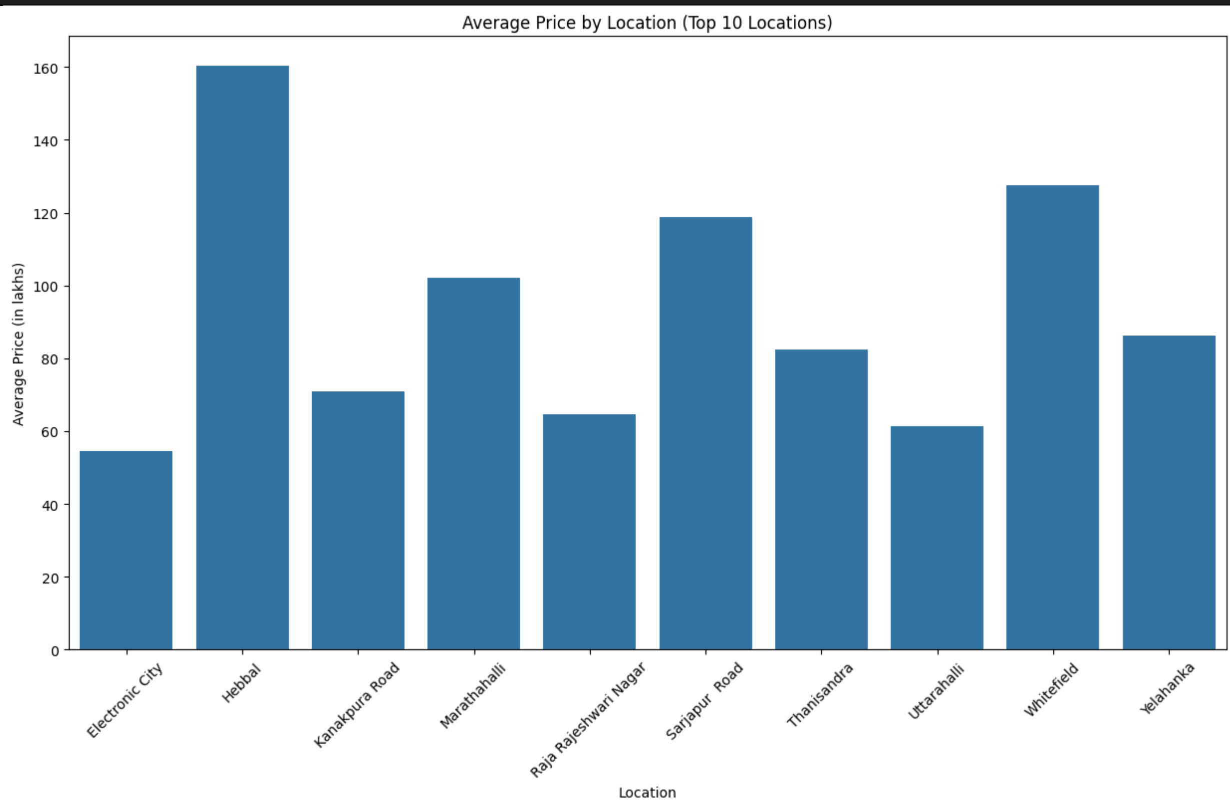
Bar plot of average price by number of bathrooms

```
plt.figure(figsize=(10, 6))
avg_price_by_bath =
df.groupby('bath')['price'].mean().reset_index()
sns.barplot(x='bath', y='price', data=avg_price_by_bath)
plt.title('Average Price by Number of Bathrooms')
plt.xlabel('Number of Bathrooms')
plt.ylabel('Average Price (in lakhs)')
plt.show()
```



Bar plot of average price by location

```
plt.figure(figsize=(15, 8))
avg_price_by_location =
df[df['location'].isin(top_locations)].groupby('location')['pric
e'].mean().reset_index()
sns.barplot(x='location', y='price', data=avg_price_by_location)
plt.title('Average Price by Location (Top 10 Locations)')
plt.xlabel('Location')
plt.ylabel('Average Price (in lakhs)')
plt.xticks(rotation=45)
plt.show()
```

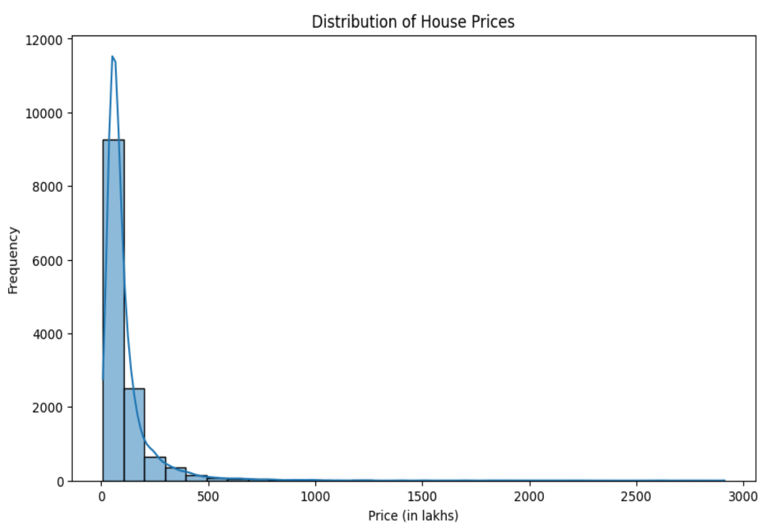


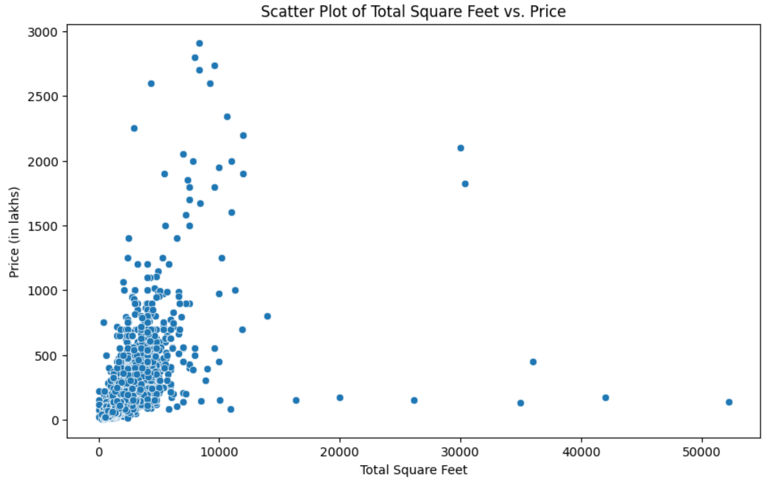
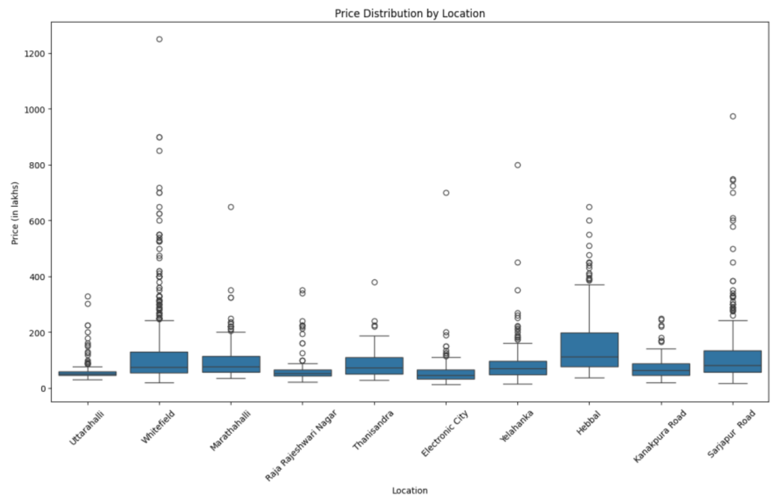
4 Conclusion

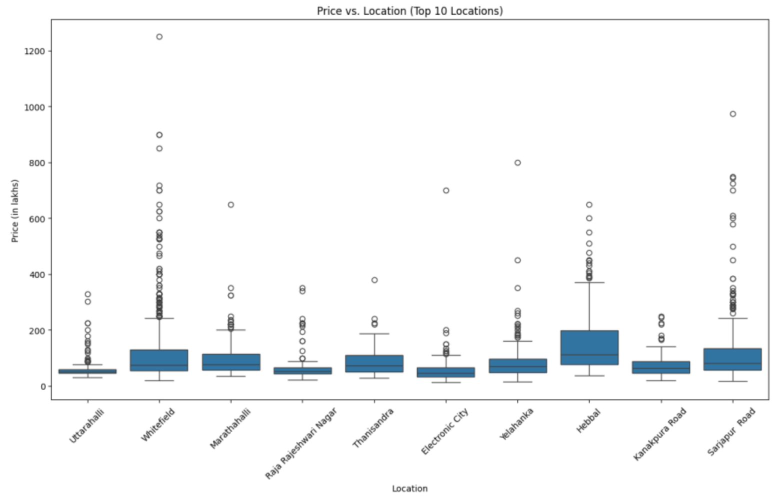
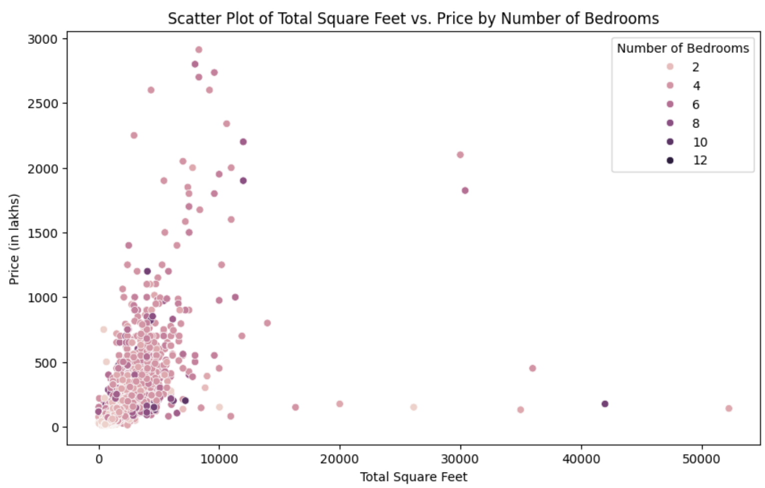
This project successfully analyzed the Bengaluru housing market, providing insights into property pricing and influencing factors. The findings underscore that:

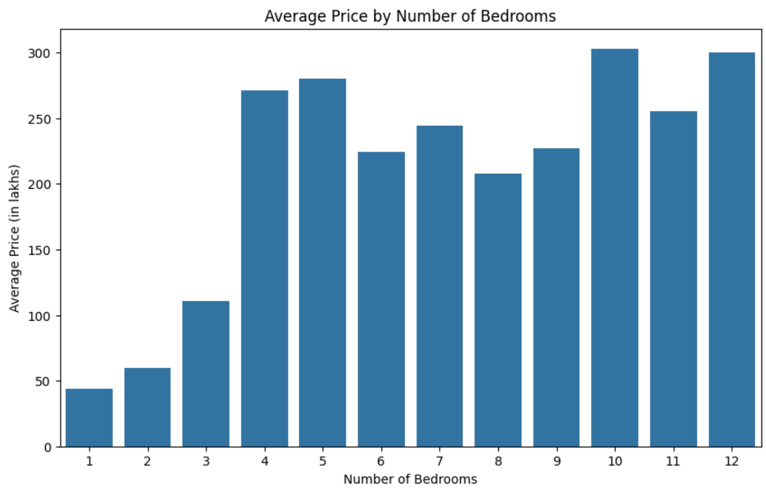
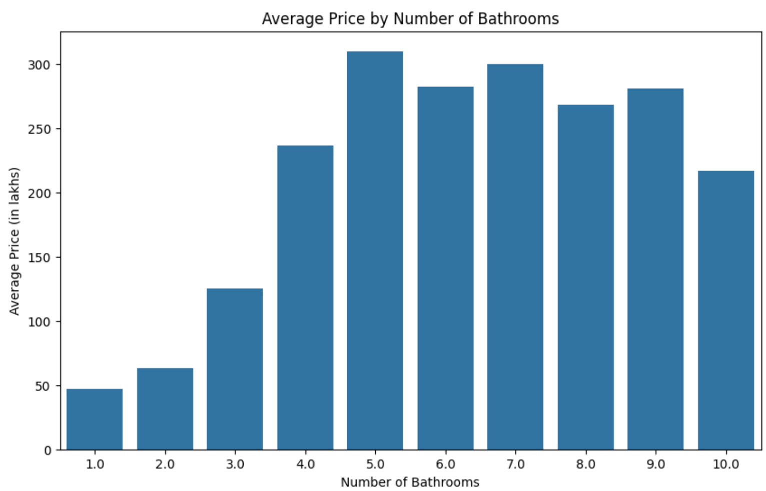
- Larger properties are generally priced higher, but location significantly influences pricing.
- Areas like Whitefield and Sarjapur Road are hotspots for mid-range and premium properties, driven by IT industry proximity.

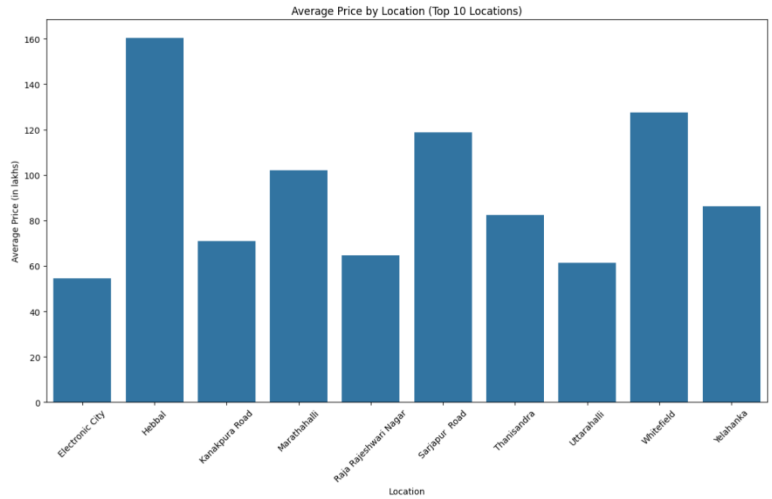
The analysis reveals a skewed market where affordability is a challenge in high-demand localities. For real estate professionals, these insights support strategic planning. Buyers and investors can use the findings to identify locations and property sizes that align with their budgets and expectations.

Analysis of Data Visualization		
Sr. No.	**Plot**	**Observation**
1	**Distribution of prices** 	The histogram of prices reveals a right-skewed distribution, with most properties priced below Rs.1 crore. A smaller number of properties fall in the luxury segment, priced above Rs.2 crore.
2	**Scatter plot of total_sqft vs. price**	A positive trend is observed, with prices

		generally increasing as the total square footage of properties grows. However, outliers are present, such as small properties priced disproportionately high.
3	**Price Distribution by Location** 	The distribution of prices varies widely by location. Areas like Whitefield and Sarjapur Road show a broader range of prices, catering to both affordable and premium segments. Conversely, central locations like Koramangala have consistently higher prices.
4	**Price vs. Location**	Prices vary significantly by location, with upscale areas like Indiranagar and Koramangala showing higher prices, while suburban locations like Electronic City offer more affordable options.

<table>
<tr><td></td><td>
</td><td></td></tr>
<tr><td>5</td><td>Scatter Plot of Total Square Feet vs. Price by Number of Bedrooms

</td><td>For properties with the same number of bedrooms, a general increase in price corresponds to larger square footage. However, outliers exist, where smaller properties are priced unusually high due to location or other premium factors.</td></tr>
<tr><td>6</td><td>Bar plot of average price by number of bedrooms</td><td>Average price increases with the number of bedrooms, but the jump between configurations (e.g., 2 BHK to 3 BHK) is not always linear. Larger homes, such as 4 BHK, show a sharp rise in price.</td></tr>
</table>

<table>
<tr><td></td><td>
</td><td></td></tr>
<tr><td>7</td><td>Bar plot of average price by number of bathrooms

</td><td>Properties with more bathrooms are generally priced higher. However, properties with an unusually high number of bathrooms (5 or more) show price anomalies, likely corresponding to luxury villas or niche properties.</td></tr>
<tr><td>8</td><td>Bar plot of average price by location</td><td>Central and well-connected areas like Koramangala, Indiranagar, and HSR Layout have the highest average prices, while suburban areas like Electronic City and Yelahanka are more affordable.</td></tr>
</table>



5 References

- **Dataset:** https://www.kaggle.com/datasets/amitabhajoy/bengaluru-house-price-data/data

- **Python Libraries & Documentations:**
    - Pandas: https://pandas.pydata.org/
    - NumPy: https://numpy.org/
    - Matplotlib: https://matplotlib.org/
    - Seaborn: https://seaborn.pydata.org/

Bengaluru Housing Price Analysis

Import necessary Python Libraries:

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

Load the dataset:

In [ ]:
```
file_path = 'Bengaluru_House_Data.csv'
df = pd.read_csv(file_path)
```

Display few rows of the dataset:

In [ ]:
```
df.head()
```

Out[ ]:

	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 [ ]:
```
df.tail()
```

Out[ ]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	pric
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.
13316	Super built-up Area	Ready To Move	Richards Town	4 BHK	NaN	3600	5.0	NaN	400.
13317	Built-up Area	Ready To Move	Raja Rajeshwari Nagar	2 BHK	Mahla T	1141	2.0	1.0	60.
13318	Super built-up Area	18-Jun	Padmanabhanagar	4 BHK	SollyCl	4689	4.0	1.0	488.
13319	Super built-up Area	Ready To Move	Doddathoguru	1 BHK	NaN	550	1.0	1.0	17.

Data Preprocessing:

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

Out[ ]: (13320, 9)

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   society       7818 non-null   object 
 5   total_sqft    13320 non-null  object 
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

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

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
bath	13247.0	2.692610	1.341458	1.0	2.0	2.0	3.0	40.0
balcony	12711.0	1.584376	0.817263	0.0	1.0	2.0	2.0	3.0
price	13320.0	112.565627	148.971674	8.0	50.0	72.0	120.0	3600.0

In [ ]:
```
df.isnull().sum()
```

Out[ ]:

	0
area_type	0
availability	0
location	1
size	16
society	5502
total_sqft	0
bath	73
balcony	609
price	0

dtype: int64

In [ ]:
```
df.duplicated().sum()
```

Out[ ]: 529

Handling missing values and irrelevant columns...:

In [ ]:
```
df = df.drop(columns=['society', 'balcony'])
df['location'] = df['location'].fillna(df['location'].mode()[0])
df = df.dropna(subset=['size', 'bath'])
```

Convert 'size' to a numeric value representing the number of bedrooms:

In [ ]:
```
df['size'] = df['size'].apply(lambda x: int(x.split(' ')[0]))
```

Identify non-numeric values in 'total_sqft':

In [ ]:
```
non_numeric_sqft = df[~df['total_sqft'].apply(lambda x: str(x).replace('.', 
'', 1).isdigit())]['total_sqft'].unique()
print("Non-numeric values in 'total_sqft':", non_numeric_sqft)
```

```
Non-numeric values in 'total_sqft': ['2100 - 2850' '3067 - 8156' '1042 - 110
5' '1145 - 1340' '1015 - 1540'
 '34.46Sq. Meter' '1195 - 1440' '4125Perch' '1120 - 1145' '3090 - 5002'
 '1160 - 1195' '1000Sq. Meter' '1115 - 1130' '1100Sq. Yards' '520 - 645'
 '1000 - 1285' '650 - 665' '633 - 666' '5.31Acres' '30Acres' '1445 - 1455'
 '884 - 1116' '850 - 1093' '716Sq. Meter' '547.34 - 827.31' '580 - 650'
 '3425 - 3435' '1804 - 2273' '3630 - 3800' '4000 - 5249' '1500Sq. Meter'
 '142.61Sq. Meter' '1574Sq. Yards' '1250 - 1305' '670 - 980'
 '1005.03 - 1252.49' '1004 - 1204' '361.33Sq. Yards' '645 - 936'
 '2710 - 3360' '2830 - 2882' '596 - 804' '1255 - 1863' '1300 - 1405'
 '117Sq. Yards' '934 - 1437' '980 - 1030' '2249.81 - 4112.19'
 '1070 - 1315' '3040Sq. Meter' '500Sq. Yards' '2806 - 3019' '613 - 648'
 '704 - 730' '1210 - 1477' '3369 - 3464' '1125 - 1500' '167Sq. Meter'
 '1076 - 1199' '381 - 535' '524 - 894' '540 - 670' '315Sq. Yards'
 '2725 - 3250' '888 - 1290' '660 - 700' '385 - 440' '770 - 841' '3Cents'
 '188.89Sq. Yards' '1469 - 1766' '204Sq. Meter' '1255 - 1350' '870 - 1080'
 '45Sq. Yards' '133.3Sq. Yards' '2580 - 2591' '2563 - 2733' '605 - 624'
 '1349 - 3324' '78.03Sq. Meter' '3300 - 3335' '1180 - 1630' '1365 - 1700'
 '122Sq. Yards' '84.53Sq. Meter' '2.09Acres' '981 - 1249' '1565 - 1595'
 '24Guntha' '1270 - 1275' '840 - 1010' '697Sq. Meter' '655 - 742'
 '1408 - 1455' '942 - 1117' '598 - 958' '1500Cents' '132Sq. Yards'
 '1010 - 1300' '2Acres' '1450 - 1950' '1100Sq. Meter' '15Acres'
 '763 - 805' '3307 - 3464' '1.26Acres' '620 - 934' '2462 - 2467'
 '540 - 740' '3508 - 4201' '4900 - 4940' '755 - 770' '664 - 722'
 '151.11Sq. Yards' '596 - 861' '615 - 985' '540 - 565' '750 - 800'
 '1660 - 1805' '1079 - 1183' '2800 - 2870' '1230 - 1290' '943 - 1220'
 '2041 - 2090' '527 - 639' '1Grounds' '1160 - 1315' '706 - 716'
 '2940Sq. Yards' '45.06Sq. Meter' '799 - 803' '2470 - 2790' '783 - 943'
 '4500 - 5540' '1255 - 1375' '610 - 615' '854 - 960' '2650 - 2990'
 '1.25Acres' '86.72Sq. Meter' '1230 - 1490' '660 - 780' '1150 - 1194'
 '684 - 810' '1510 - 1670' '1550 - 1590' '1235 - 1410' '38Guntha'
 '929 - 1078' '2150 - 2225' '1520 - 1759' '629 - 1026' '1215 - 1495'
 '6Acres' '1140 - 1250' '2400 - 2600' '1052 - 1322' '5666 - 5669'
 '712 - 938' '1783 - 1878' '120Sq. Yards' '24Sq. Meter' '2528 - 3188'
 '650 - 760' '1400 - 1421' '4000 - 4450' '142.84Sq. Meter' '300Sq. Yards'
 '1437 - 1629' '850 - 1060' '1200 - 1470' '1133 - 1384']
```

In [ ]:
```
def convert_sqft_to_num(x):
    try:
        if '-' in x:
            nums = x.split('-')
            return (float(nums[0]) + float(nums[1])) / 2
        return float(x)
    except:
        return np.nan

df['total_sqft'] = df['total_sqft'].apply(convert_sqft_to_num)
df = df.dropna(subset=['total_sqft'])
```

Ensure all relevant columns are numeric:

In [ ]:
```
df['price'] = pd.to_numeric(df['price'], errors='coerce')
df['bath'] = pd.to_numeric(df['bath'], errors='coerce')
```

Drop any remaining rows with NaN values in these columns:

In [ ]:
```
df = df.dropna(subset=['price', 'bath', 'total_sqft', 'size'])
```

Cleaned dataset structure:

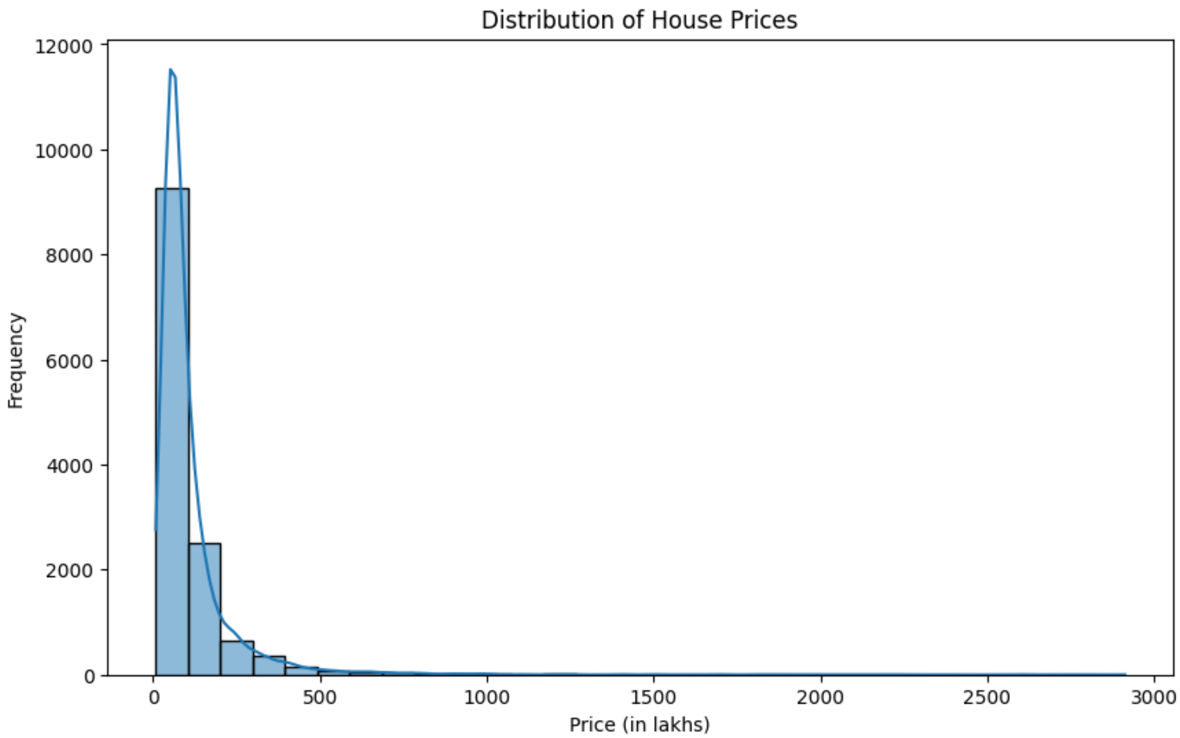
In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 13201 entries, 0 to 13319
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     13201 non-null  object
 1   availability  13201 non-null  object
 2   location      13201 non-null  object
 3   size          13201 non-null  int64
 4   total_sqft    13201 non-null  float64
 5   bath          13201 non-null  float64
 6   price         13201 non-null  float64
dtypes: float64(3), int64(1), object(3)
memory usage: 825.1+ KB
```

Data Visualization

Distribution of prices

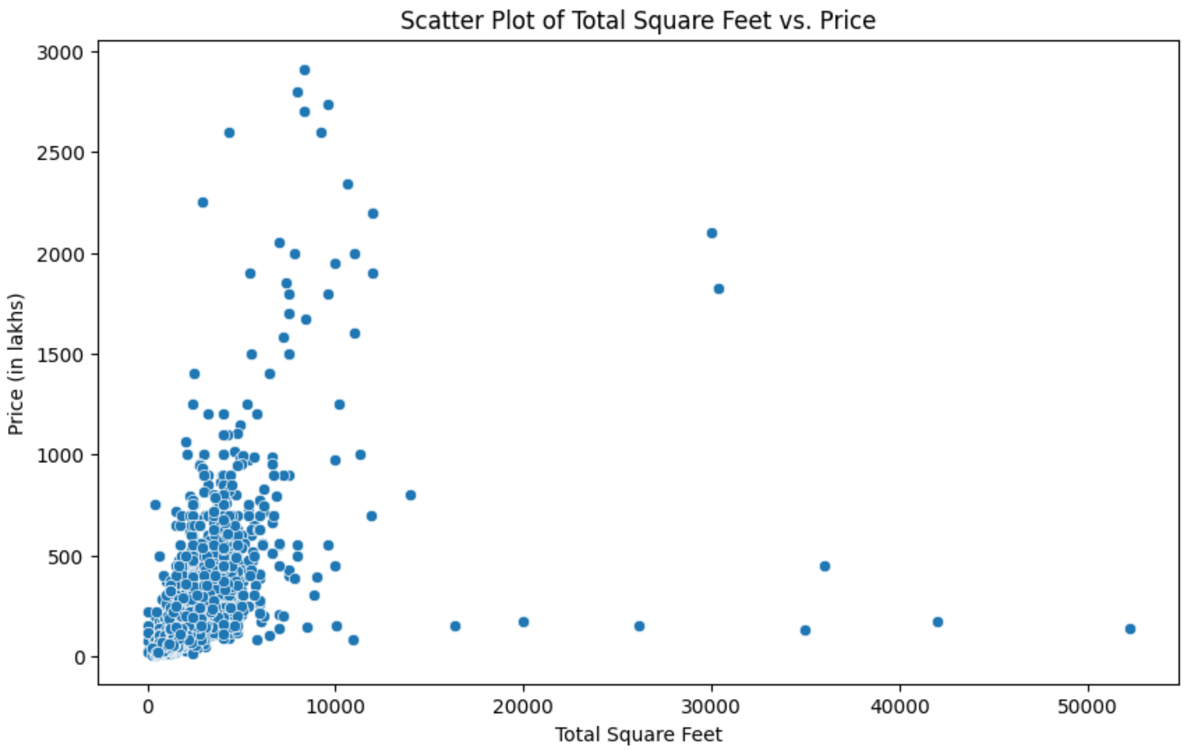
In [ ]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], kde=True, bins=30)
plt.title('Distribution of House Prices')
plt.xlabel('Price (in lakhs)')
plt.ylabel('Frequency')
plt.show()
```



Scatter plot of total_sqft vs. price

In [ ]:

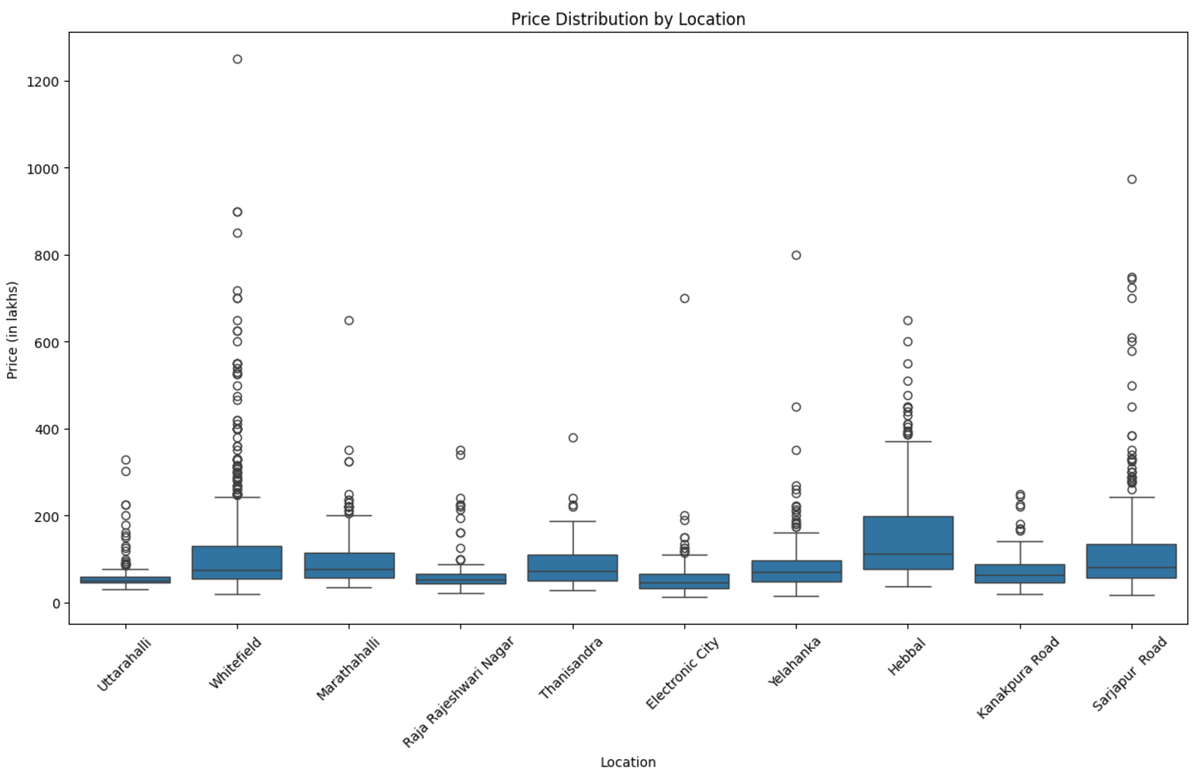
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='total_sqft', y='price', data=df)
plt.title('Scatter Plot of Total Square Feet vs. Price')
plt.xlabel('Total Square Feet')
plt.ylabel('Price (in lakhs)')
plt.show()
```



Price Distribution by Location

In [ ]:

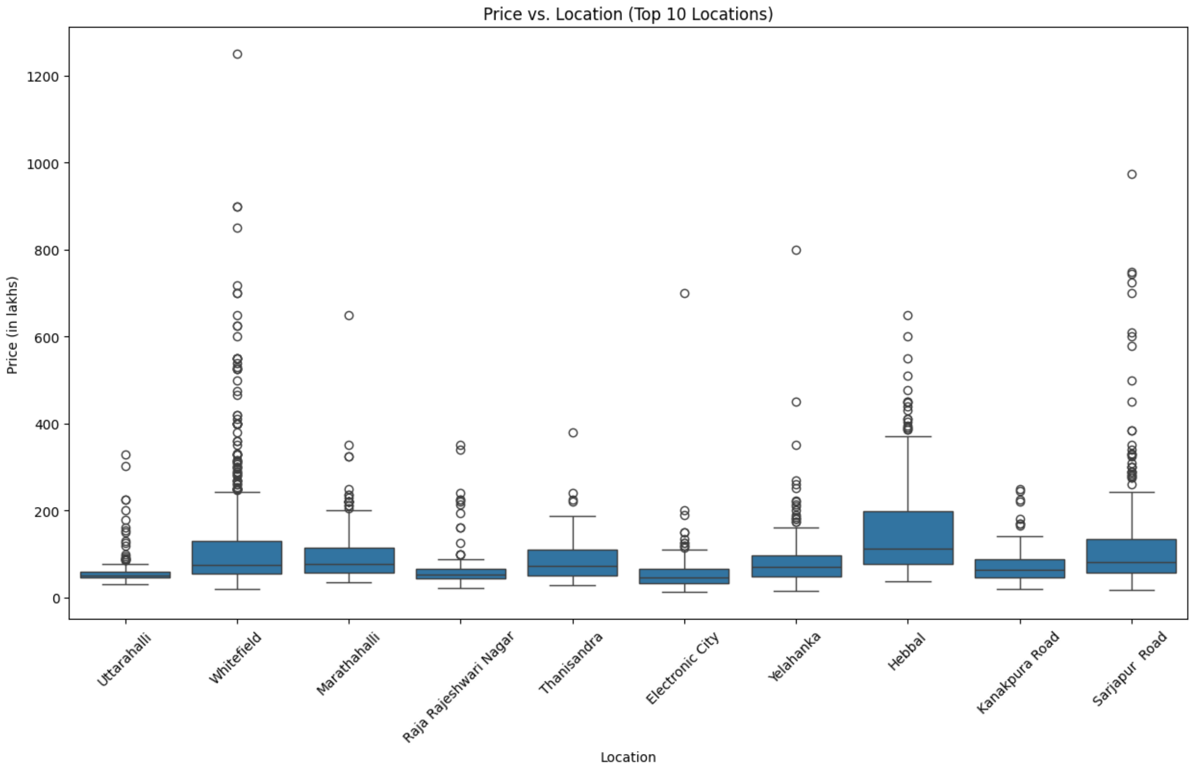
```
plt.figure(figsize=(15, 8))
top_locations = df['location'].value_counts().nlargest(10).index
sns.boxplot(x='location', y='price', data=df[df['location'].isin(top_location
s)])
plt.title('Price Distribution by Location')
plt.xlabel('Location')
plt.ylabel('Price (in lakhs)')
plt.xticks(rotation=45)
plt.show()
```



Price vs. Location

In [ ]:

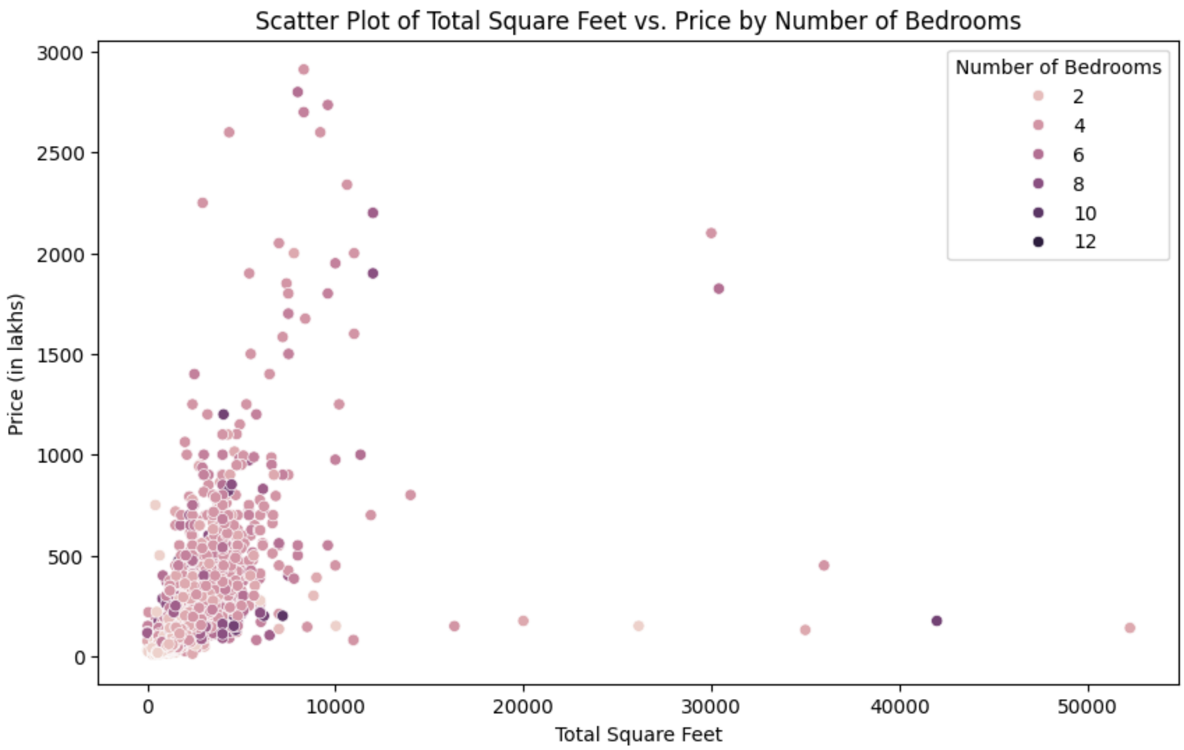
```
plt.figure(figsize=(15, 8))
top_locations = df['location'].value_counts().nlargest(10).index
sns.boxplot(x='location', y='price', data=df[df['location'].isin(top_location
s)])
plt.title('Price vs. Location (Top 10 Locations)')
plt.xlabel('Location')
plt.ylabel('Price (in lakhs)')
plt.xticks(rotation=45)
plt.show()
```



Scatter Plot of Total Square Feet vs. Price by Number of Bedrooms

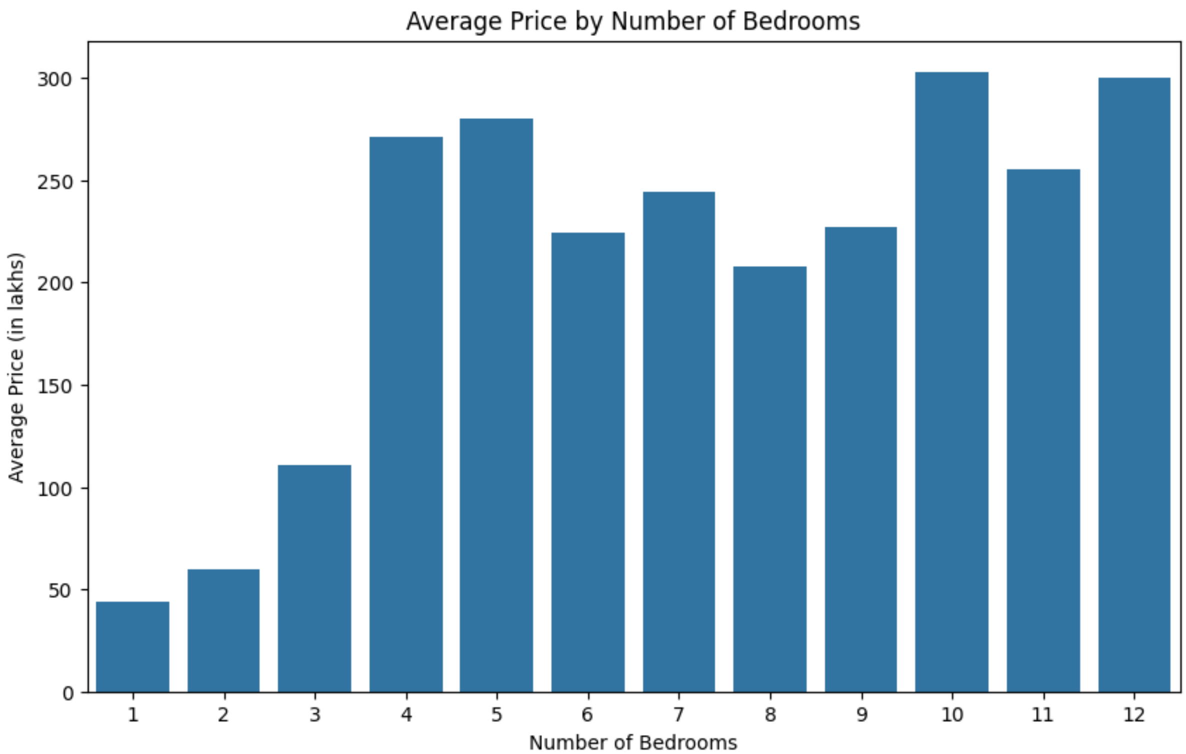
In [ ]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='total_sqft', y='price', hue='size', data=df)
plt.title('Scatter Plot of Total Square Feet vs. Price by Number of Bedrooms')
plt.xlabel('Total Square Feet')
plt.ylabel('Price (in lakhs)')
plt.legend(title='Number of Bedrooms')
plt.show()
```



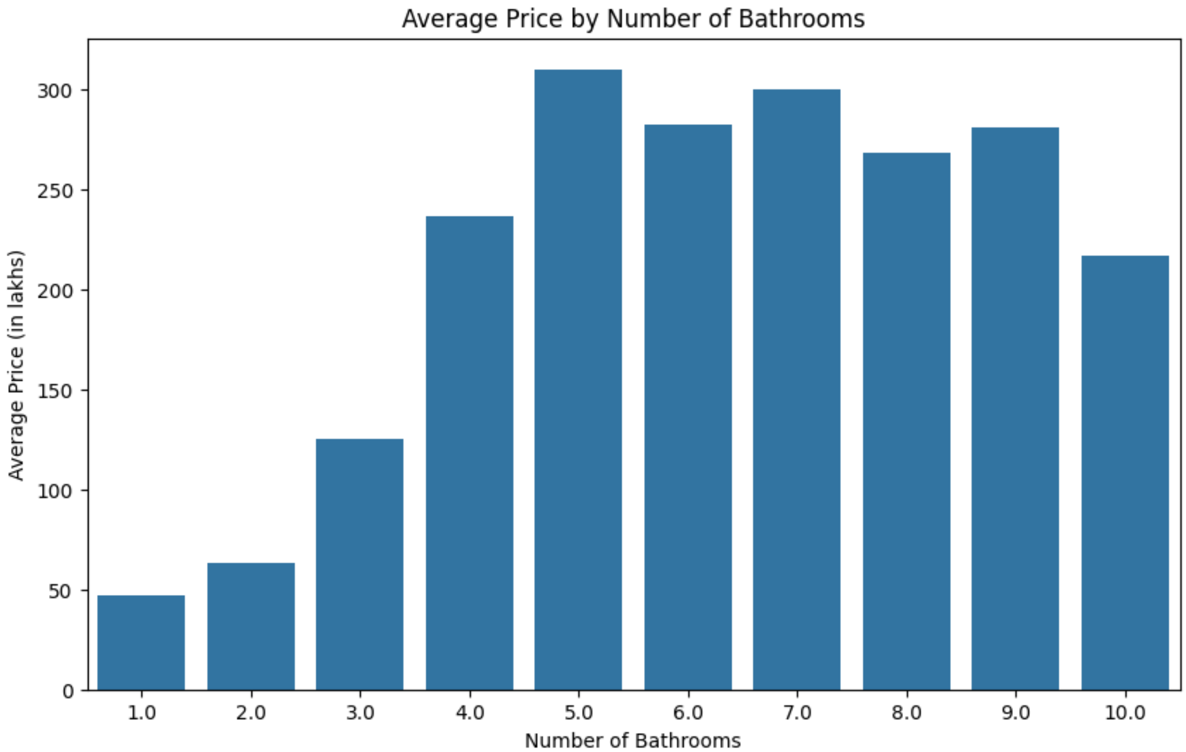
Bar plot of average price by number of bedrooms

In [ ]:
```
plt.figure(figsize=(10, 6))
avg_price_by_size = df.groupby('size')['price'].mean().reset_index()
sns.barplot(x='size', y='price', data=avg_price_by_size)
plt.title('Average Price by Number of Bedrooms')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Average Price (in lakhs)')
plt.show()
```



Bar plot of average price by number of bathrooms

In [ ]:
```
plt.figure(figsize=(10, 6))
avg_price_by_bath = df.groupby('bath')['price'].mean().reset_index()
sns.barplot(x='bath', y='price', data=avg_price_by_bath)
plt.title('Average Price by Number of Bathrooms')
plt.xlabel('Number of Bathrooms')
plt.ylabel('Average Price (in lakhs)')
plt.show()
```



Bar plot of average price by location

In [ ]:

```
plt.figure(figsize=(15, 8))
avg_price_by_location = df[df['location'].isin(top_locations)].groupby('locati
on')['price'].mean().reset_index()
sns.barplot(x='location', y='price', data=avg_price_by_location)
plt.title('Average Price by Location (Top 10 Locations)')
plt.xlabel('Location')
plt.ylabel('Average Price (in lakhs)')
plt.xticks(rotation=45)
plt.show()
```