MACHINE LEARNING: REPORT

Abstract:

This report presents a comprehensive analysis of the Gurgaon real estate dataset focusing on handling missing values and detecting outliers to ensure data quality and resilience for future analysis. The main goals are to find and fix missing values, find, and handle outliers. This study describes the methodology, visualizations, and statistical analysis employed, as well as insights and recommendations for future data preparation jobs in real estate analytics.

Motivation:

In real estate analytics, data quality is critical for generating accurate and meaningful insights. Missing numbers and outliers can skew analytical results, resulting in inaccurate conclusions and poor decisions. This is especially crucial in the Gurgaon real estate market, where different property qualities and volatile market conditions need precise data management.

Improving Data Integrity: To increase dataset dependability, locate and handle outliers and missing values. This guarantees that the data used for further analyses is accurate.

Enhancing the Performance of the Model: Building solid machine learning models requires high-quality data. Managing inconsistent data is essential for developing models that are dependable in their predictions and have good generalization, which is important for real estate industry players.

Dataset:

For this project, we are using the dataset of the Gurgaon_RealEstate. The dataset consists of several variables:

property_type	agePossesion
society	Super_built_up_area
sector	built_up_area
price	carpet_area
price_per_sqft	study room
area	servant room
bedroom	store room
bathroom	pooja room
balcony	others
floorNum	furnishing_type
facing	luxury_score

 # Mount the google drive with the Colab file using this code:

```
from google.colab import drive
drive.mount('/content/drive')
```

TASK - 1:

Loading the dataset (csv file) in google colab file (or any other preferred data analysis environment) using the python code below to read the csv file.

```
data = pd.read_csv("/content/drive/MyDrive/MachineLearning Project/Gurgaon_RealEstate.csv")
data
```

The output after executing the read.csv code:

	property_type	society	sector	price	price_per_sqft	area	areaWithType	bedRoom	bathroom	balcony	...	super_built_up_area	built_up_area	carpet_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_score
0	flat	signature global park 4	sector 36	0.82	7585.0	1081.0	Super Built up area 1081(100.43 sq.m.)Carpet a...	3	2	2	...	1081.0	NaN	650.0	0	0	0	0	0	0	8
1	flat	smart world gems	sector 89	0.95	8600.0	1105.0	Carpet area: 1103 (102.47 sq.m.)	2	2	2	...	NaN	NaN	1103.0	1	1	0	0	0	0	38
2	flat	pyramid elite	sector 86	0.46	79.0	58228.0	Carpet area: 58141 (5401.48 sq.m.)	2	2	1	...	NaN	NaN	58141.0	0	0	0	0	0	0	15
3	flat	breez global hill view	sohna road	0.32	5470.0	585.0	Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...	2	2	1	...	NaN	1000.0	585.0	0	0	0	0	0	0	49
4	flat	bestech park view sanskruti	sector 92	1.60	8020.0	1995.0	Super Built up area 1995(185.34 sq.m.)Built Up...	3	4	3+	...	1995.0	1615.0	1476.0	0	1	0	0	1	1	174
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3798	flat	pivotal devaan	sector 84	0.37	6346.0	583.0	Super Built up area 583(54.16 sq.m.)Carpet are...	2	2	1	...	583.0	NaN	483.0	0	0	0	0	0	0	73
3799	house	international city by sobha phase 1	sector 109	6.00	9634.0	6228.0	Plot area 692(578.6 sq.m.)	5	5	3+	...	NaN	6228.0	NaN	1	1	1	1	0	0	160
3800	flat	ansal api celebrity suites	sector 2	0.60	8163.0	735.0	Super Built up area 735(68.28 sq.m.)	1	1	1	...	735.0	NaN	NaN	0	0	0	0	0	1	67
3801	house	independent	sector 43	15.50	28233.0	5490.0	Plot area 610(510.04 sq.m.)	5	6	3	...	NaN	5490.0	NaN	1	1	1	1	0	0	76
3802	flat	m3m ikonic	sector 68	1.78	9128.0	1950.0	Super Built up area 1950(181.16 sq.m.)Built Up...	3	3	3+	...	1950.0	1845.0	1530.0	0	0	0	0	0	1	126

3803 rows × 23 columns

Deleting the duplicate rows:

#Deleting the duplicates rows from the data and storing the data frame into new_data because deleting the duplicated decreases the bias.

```
[ ]  duplicates = data.duplicated()

     print(f"Number of duplicate rows found: {duplicates.sum()}")
     new_data = data.drop_duplicates()
     print("Duplicates removed!")
     data
```

 # Now the duplicate rows have been deleted

```
Number of duplicate rows found: 126
Duplicates removed!
```

	property_type	society	sector	price	price_per_sqft	area	areaWithType	bedRoom	bathroom	balcony	...	super_built_up_area	built_up_area	carpet_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_score
0	flat	signature global park 4	sector 36	0.82	7585.0	1081.0	Super Built up area 1081(100.43 sq.m.)Carpet a...	3	2	2	...	1081.0	NaN	650.0	0	0	0	0	0	0	8
1	flat	smart world gems	sector 89	0.95	8600.0	1105.0	Carpet area: 1103 (102.47 sq.m.)	2	2	2	...	NaN	NaN	1103.0	1	1	0	0	0	0	38
2	flat	pyramid elite	sector 86	0.46	79.0	58228.0	Carpet area: 58141 (5401.48 sq.m.)	2	2	1	...	NaN	NaN	58141.0	0	0	0	0	0	0	15
3	flat	breez global hill view	sohna road	0.32	5470.0	585.0	Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...	2	2	1	...	NaN	1000.0	585.0	0	0	0	0	0	0	49
4	flat	bestech park view sanskruti	sector 92	1.60	8020.0	1995.0	Super Built up area 1995(185.34 sq.m.)Built Up...	3	4	3+	...	1995.0	1615.0	1476.0	0	1	0	0	1	1	174
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3798	flat	pivotal devaan	sector 84	0.37	6346.0	583.0	Super Built up area 583(54.16 sq.m.)Carpet are...	2	2	1	...	583.0	NaN	483.0	0	0	0	0	0	0	73
3799	house	international city by sobha phase 1	sector 109	6.00	9634.0	6228.0	Plot area 692(578.6 sq.m.)	5	5	3+	...	NaN	6228.0	NaN	1	1	1	1	0	0	160
3800	flat	ansal api celebrity suites	sector 2	0.60	8163.0	735.0	Super Built up area 735(68.28 sq.m.)	1	1	1	...	735.0	NaN	NaN	0	0	0	0	0	1	67
3801	house	independent	sector 43	15.50	28233.0	5490.0	Plot area 610(510.04 sq.m.)	5	6	3	...	NaN	5490.0	NaN	1	1	1	1	0	0	76
3802	flat	m3m ikonic	sector 68	1.78	9128.0	1950.0	Super Built up area 1950(181.16 sq.m.)Built Up...	3	3	3+	...	1950.0	1845.0	1530.0	0	0	0	0	0	1	126

3803 rows × 23 columns

Exploring the columns of the initial data:

 # Exploring the property type, this contains only two residential properties houses and flats.
 # Data type of Property type column is object that it is categorical data

Code used to find the data type:

```
# feature of the coloum property type
property_type = data['property_type'].dtype


print(f"The data type of the 'Property_type' column is: {property_type}")
```

```
The data type of the 'Property_type' column is: object
```

Finding the percentages of houses and flats:

The percentage of houses is = 77.39%
The percentage of flats is = 22.61%
That is the number of houses is 860 and the number of flats is 2943

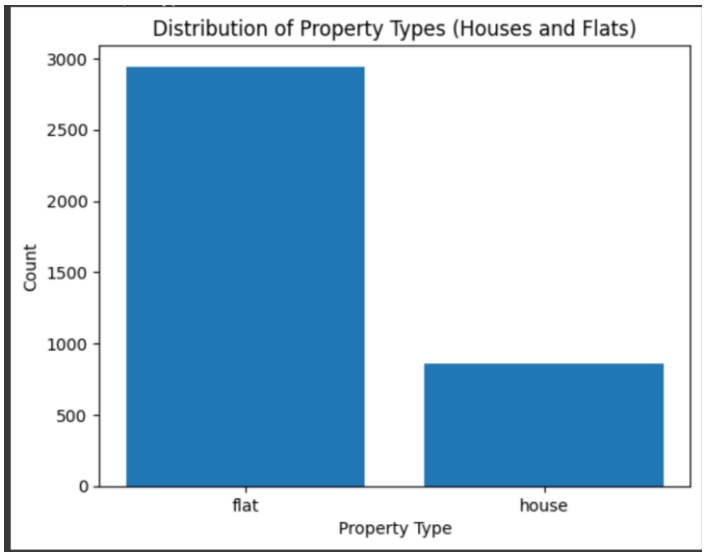
```
print("\nProperty Types:")
property_type_counts = data["property_type"].value_counts()
print(property_type_counts)  # Shows counts of each property type
```

```
Property Types:
property_type
flat     2943
house     860
Name: count, dtype: int64
```

Plotting a histogram of Houses vs Flats using matplotlib.pyplot as plt:

```
plt.bar(property_type_counts.index, property_type_counts.values)
plt.xlabel("Property Type")
plt.ylabel("Count")
plt.title("Distribution of Property Types (Houses and Flats)")

plt.show()
```

Count on y axis and x axis contains the Property Type



Missing values of property type:

```
[260] # Finding missing values in property type

      missing_values_count = data[property_type].isnull().sum()

      print(f"Number of missing values in property types: {missing_values_count}")

      Number of missing values in property types: 0
```

The missing value of property type is ‘0’

Exploring the society column:

 # The data type of the society column is object i.e., it is categorical column

```
[261]

      column_tye = data['society'].dtype

      print(f"The data type of the 'society' column is: {column_tye}")

      The data type of the 'society' column is: object
```

 # This gives the society counts present in the society column

```
society_counts = data['society'].value_counts()

print("Unique type names and their counts:")
print(society_counts)
```

```
Unique type names and their counts:
society
independent                                 486
tulip violet                                 75
ss the leaf                                  74
shapoorji pallonji joyville gurugram         45
dlf new town heights                         42
                                           ...
samadhan shri kamal cghs                      1
private house                                 1
elevate                                       1
dlf pink town house                           1
surendra homes dayaindependentd colony        1
Name: count, Length: 676, dtype: int64
```

Removing the independent:

 # As independent houses don't belong to society, we remove them from data

```
society_column = 'society'

type_counts = data[society_column].value_counts()


n_type_counts = type_counts.drop('independent', errors='ignore')


print("Unique type names and their counts (excluding 'independent'):")
print(n_type_counts)
```

```
Unique type names and their counts (excluding 'independent'):
society
tulip violet                              75
ss the leaf                               74
shapoorji pallonji joyville gurugram      45
dlf new town heights                      42
signature global park                     37
                                          ..
samadhan shri kamal cghs                   1
private house                              1
elevate                                    1
dlf pink town house                        1
surendra homes dayaindependentd colony     1
Name: count, Length: 675, dtype: int64
```

Dropping independent and storing it in new data frame new_data:

```
rows_to_drop = data[data['society'] == 'independent'].index
new_data = societydata = data.drop(rows_to_drop)
new_data
```

Now we have removed and independent column from the data and stored it in new_data

	property_type	society	sector	price	price_per_sqft	area	areaWithType	bedRoom	bathroom	balcony	...	super_built_up_area	built_up_area	carpet_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_scor
0	flat	signature global park 4	sector 36	0.82	7585.0	1081.0	Super Built up area 1081(100.43 sq.m.)Carpet a...	3	2	2	...	1081.0	NaN	650.0	0	0	0	0	0	0	
1	flat	smart world gems	sector 89	0.95	8600.0	1105.0	Carpet area: 1103 (102.47 sq.m.)	2	2	2	...	NaN	NaN	1103.0	1	1	0	0	0	0	3
2	flat	pyramid elite	sector 86	0.46	79.0	58228.0	Carpet area: 58141 (5401.48 sq.m.)	2	2	1	...	NaN	NaN	58141.0	0	0	0	0	0	0	1
3	flat	breez global hill view	sohna road	0.32	5470.0	585.0	Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...	2	2	1	...	NaN	1000.0	585.0	0	0	0	0	0	0	4
4	flat	bestech park view sanskruti	sector 92	1.60	8020.0	1995.0	Super Built up area 1995(185.34 sq.m.)Built Up...	3	4	3+	...	1995.0	1615.0	1476.0	0	1	0	0	1	1	17
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3797	house	surendra homes dayaindependentd colony	sector 6	0.75	15625.0	480.0	Built Up area: 480 (44.59 sq.m.)	4	4	2	...	NaN	480.0	NaN	0	0	0	0	0	0	
3798	flat	pivotal devaan	sector 84	0.37	6346.0	583.0	Super Built up area 583(54.16 sq.m.)Carpet are...	2	2	1	...	583.0	NaN	483.0	0	0	0	0	0	0	7
3799	house	international city by sobha phase 1	sector 109	6.00	9634.0	6228.0	Plot area 692(578.6 sq.m.)	5	5	3+	...	NaN	6228.0	NaN	1	1	1	1	0	0	16
3800	flat	ansal api celebrity suites	sector 2	0.60	8163.0	735.0	Super Built up area 735(68.28 sq.m.)	1	1	1	...	735.0	NaN	NaN	0	0	0	0	0	1	6
3802	flat	m3m ikonic	sector 68	1.78	9128.0	1950.0	Super Built up area 1950(181.16 sq.m.)Built Up...	3	3	3+	...	1950.0	1845.0	1530.0	0	0	0	0	0	1	12

3317 rows × 23 columns

Searching for missing values in new_data:

```
new_data.isnull().sum()
```

```
property_type             0
society                   1
sector                    0
price                    13
price_per_sqft           13
area                     13
areaWithType              0
bedRoom                   0
bathroom                  0
balcony                   0
floorNum                 10
facing                  929
agePossession             0
super_built_up_area    1402
built_up_area          2037
carpet_area            1461
study room                0
servant room              0
store room                0
pooja room                0
others                    0
furnishing_type           0
luxury_score              0
dtype: int64
```

We found that there are

Exploring sector column:

Data type of sector: object (categorical data)

```
sector_type = data['sector'].dtype
print(f"The data type of the 'sector_type' column is: {sector_type}")
```

```
The data type of the 'sector_type' column is: object
```

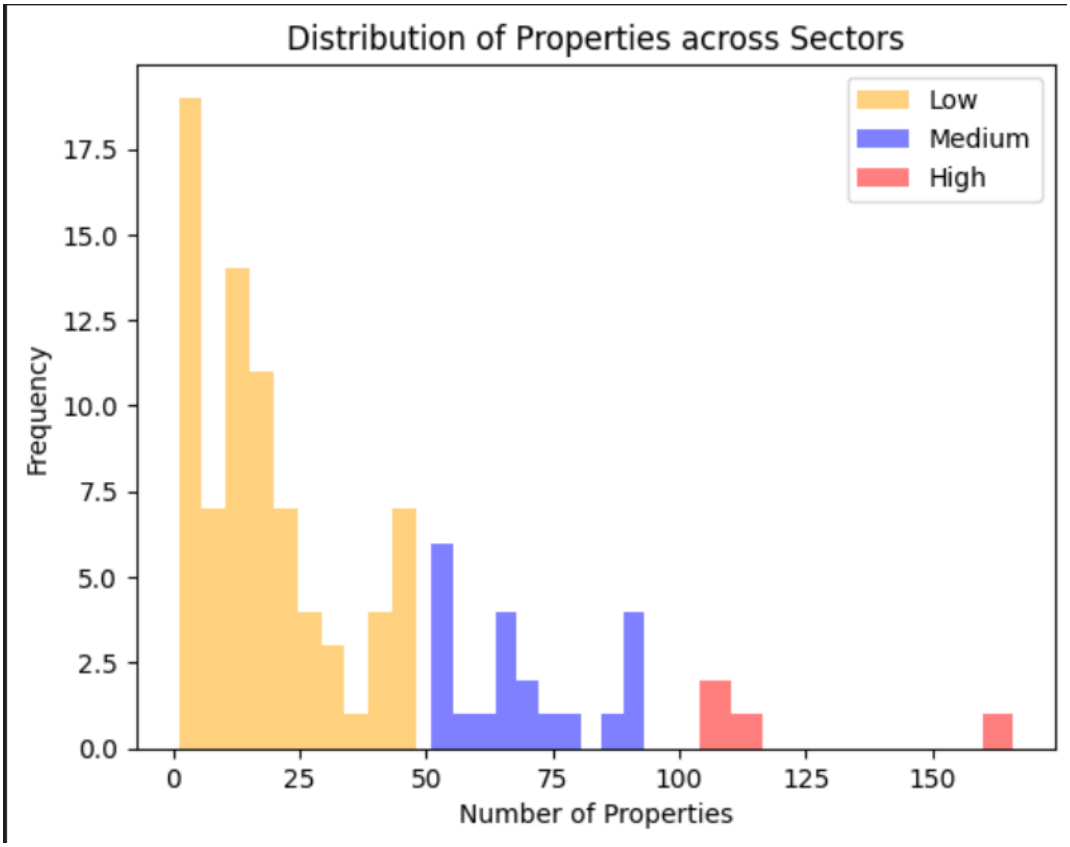
Then the code counts the occurrences of each unique value in the 'sector' column of the new_data DataFrame and categorizes these counts into 'low', 'medium', and 'high' bins based on specified thresholds. It then prints the names of the sectors that fall into each bin category.

```
sector_counts = new_data['sector'].value_counts()
print(sector_counts)
binning = {
    'low': sector_counts[sector_counts <= 50],
    'medium': sector_counts[(sector_counts > 50) & (sector_counts <= 100)],
    'high': sector_counts[sector_counts > 100]
}
print("Low:", binning['low'].index.tolist())
print("Medium:", binning['medium'].index.tolist())
print("High:", binning['high'].index.tolist())
```

```
sector
sohna road    166
sector 102    113
sector 85     110
sector 92     104
sector 69      93
             ...
sector 88       3
sector 46       2
sector 40       2
sector 23       2
sector 13       1
Name: count, Length: 102, dtype: int64
Low: ['sector 99', 'sector 56', 'sector 67', 'sector 49', 'sector 66', 'sector 2', 'sector 103', 'sector 113', 'sector 61', 'sector 82', 'sector 43', 'sector 106', 'sector 28', 'sector 68', 'manesar', 'sector 77', 'sector 72', 'sector 71', 'sector 54', 'sector 112', 'sector 88a', 'sector 67a', 'sector 111', 'sector 1
Medium: ['sector 69', 'sector 90', 'sector 65', 'sector 81', 'sector 109', 'sector 79', 'sector 33', 'sector 104', 'sector 83', 'sector 86', 'sector 95', 'sector 37d', 'sector 89', 'sector 107', 'sector 108', 'sector 50', 'sector 48', 'sector 37', 'sector 70a', 'sector 70', 'sector 84']
High: ['sohna road', 'sector 102', 'sector 85', 'sector 92']
```

Plotting the graph the counts in low, mid, and high

```
plt.hist(binning['low'], bins=10, color='orange', alpha=0.5, label='Low')
plt.hist(binning['medium'], bins=10, color='blue', alpha=0.5, label='Medium')
plt.hist(binning['high'], bins=10, color='red', alpha=0.5, label='High')
plt.xlabel('Number of Properties')
plt.ylabel('Frequency')
plt.title('Distribution of Properties across Sectors')
plt.legend()
plt.show()
```



Code from the top 15 sectors in sector column:

As here we are using binning technique to find the top sectors with high frequency. Taking the frequency with Low,Mid,High as the bins and after taking the society from the high frequency sectors.

```
col = 'sector'
sector_count= new_data['sector'].value_counts()
top_15_sectors = sector_count.head(15)
print("Top 15 Sectors:")
print(top_15_sectors)
```

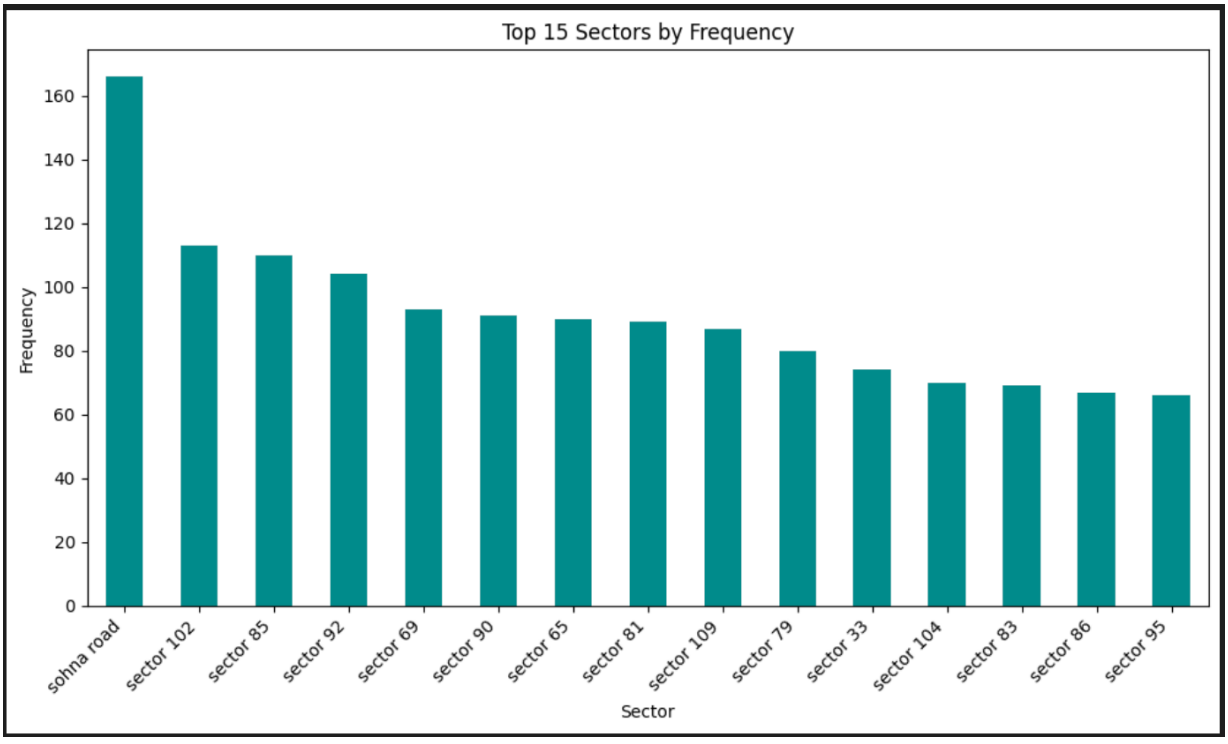
```
Top 15 Sectors:
sector
sohna road    166
sector 102    113
sector 85     110
sector 92     104
sector 69      93
sector 90      91
sector 65      90
sector 81      89
sector 109     87
sector 79      80
sector 33      74
sector 104     70
sector 83      69
sector 86      67
sector 95      66
Name: count, dtype: int64
```

Plotting histogram for top 15 sectors:

Taking the top sectors and plotting it on the histogram with sectors on x-axis and frequency on y-axis we have a histogram.

```
sector_column = 'sector'
sector_counts = new_data[sector_column].value_counts()
top_15_sectors = sector_counts.head(15)


plt.figure(figsize=(10, 6))
top_15_sectors.plot(kind='bar', color='darkcyan')
plt.title('Top 15 Sectors by Frequency')
plt.xlabel('Sector')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

[378]



Exploring price column:

As we explore the price column, we have the following analysis and descriptive analysis

Data type of price: float64 (Numerical)

```
[379]
    feature  = data['price'].dtype
    print(f"The data type of the 'price' column is: {feature}")

...  The data type of the 'price' column is: float64
```

Missing values of price = 13

```
missing_values_count = new_data['price'].isnull().sum()
print(f"Number of missing values in 'price' column: {missing_values_count}")

Number of missing values in 'price' column: 13
```

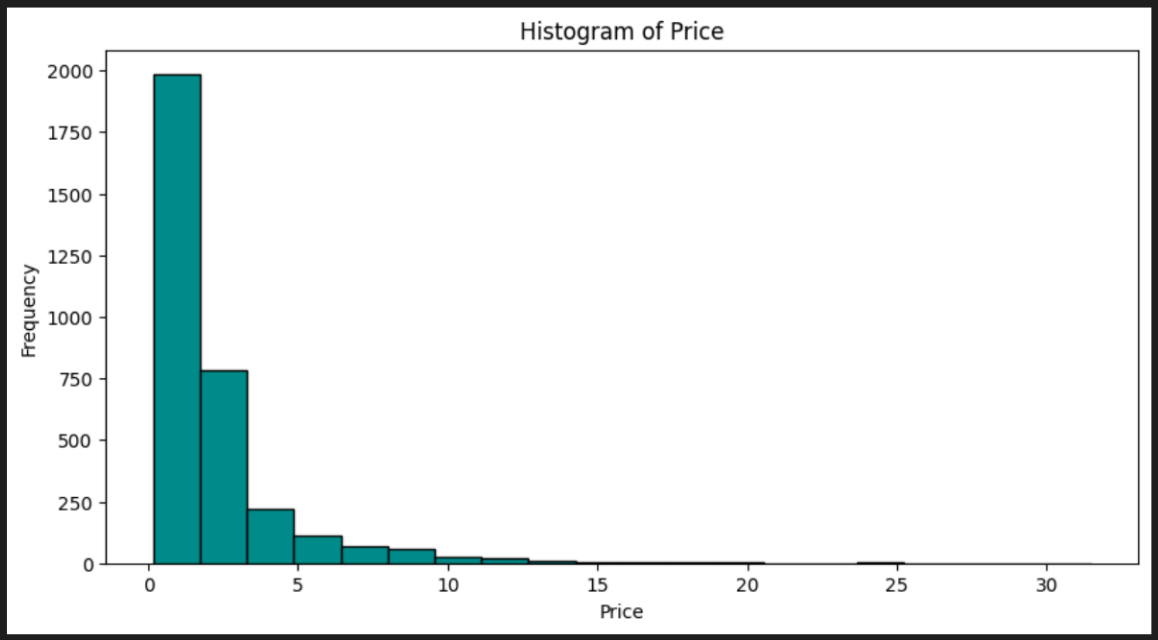
```
    print("Description of Price Column\n")
    new_data.price.describe()
```

```
Description of Price Column


count    3304.000000
mean        2.223284
std         2.552565
min         0.160000
25%         0.920000
50%         1.450000
75%         2.350000
max        31.500000
Name: price, dtype: float64
```

Histogram of price :

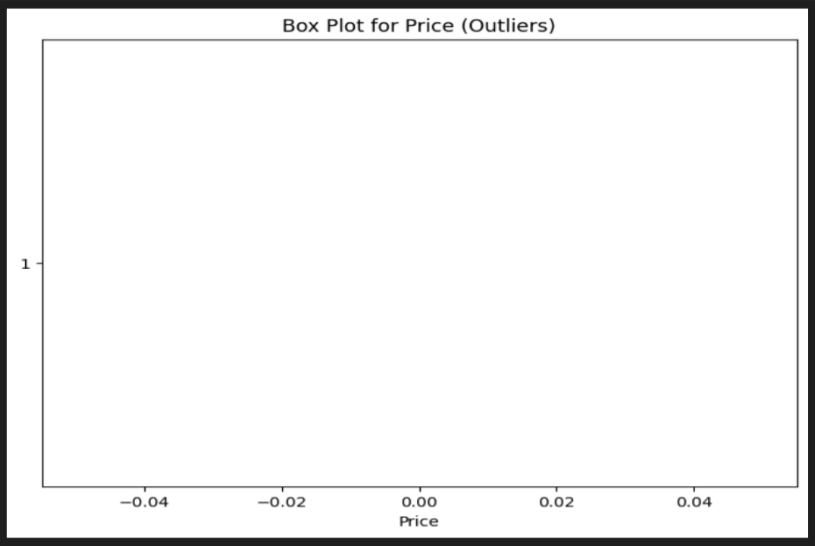
We have the histogram of price column i.e., price v/s Frequency as we have seen here;

```
plt.figure(figsize=(10,5))
plt.hist(new_data['price'],bins=20,color='darkcyan',edgecolor = 'black')
plt.title('Histogram of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



Box plot for price:

```
plt.figure(figsize=(8, 6))
plt.boxplot(new_data['price'], vert=False)

plt.xlabel('Price')
plt.title('Box Plot for Price (Outliers)')
plt.show()
```




```
skewness = new_data['price'].skew()
kurt = new_data['price'].kurtosis()

print("Skewness of price column:", skewness)
print("Kurtosis of price column:", kurt)
```

```
Skewness of price column: 3.803469637490276
Kurtosis of price column: 21.377596968556386
```

Skewness of price column: 3.803469637490276
Kurtosis of price column: 21.377596968556386

Similarly, the code of histogram, boxplot and Skewness and Kurtosis of other columns can be done changing the column name

Exploring ‘Price per sqft’s column:

Now, here explore the price per sqft column, we analysis the missing values as above and descriptive analysis.

#Feature == Price_per_square_foot

```
#fearure of the priceprice_per_sqft coloum

column_type = data['price_per_sqft'].dtype

print(f"The data type of the 'price_per_sqft' column is: {column_type}")
```

```
The data type of the 'price_per_sqft' column is: float64
```

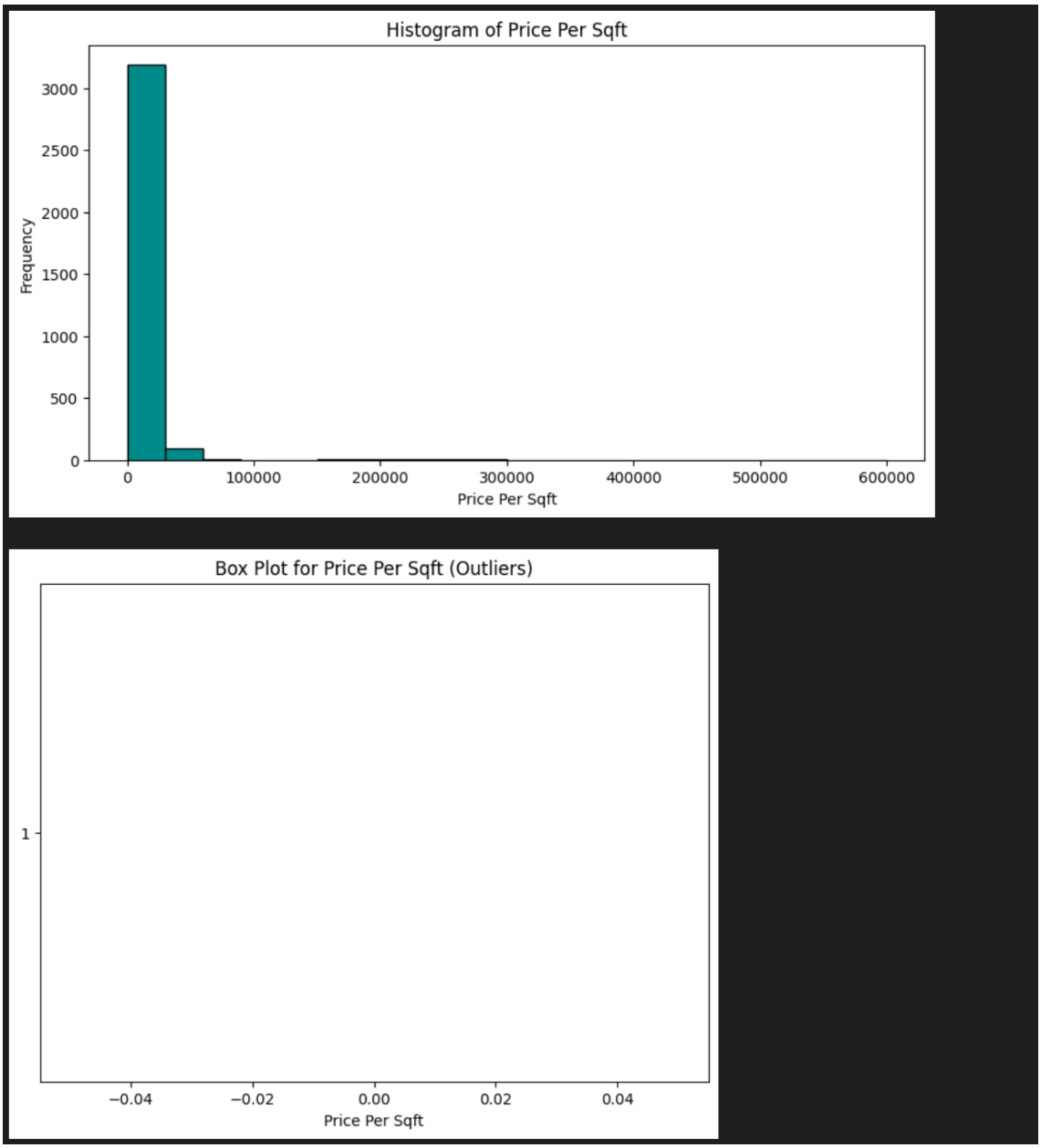
```
price_per_sqft_missing = new_data['price_per_sqft'].isnull().sum()
print(f"Number of missing values in 'price_per_sqft' column: {price_per_sqft_missing}")
```

```
Number of missing values in 'price_per_sqft' column: 13
```

```
print("Statistics for the 'price_per_sqft' column:")
print(data['price_per_sqft'].describe())
```

```
Statistics for the 'price_per_sqft' column:
count      3785.000000
mean      13800.167768
std       23052.005585
min           4.000000
25%        6808.000000
50%        9000.000000
75%       13765.000000
max      600000.000000
```

Missing values = 13
Data type = float64



Skewness of price_per_sqft column: 15.015074679883858

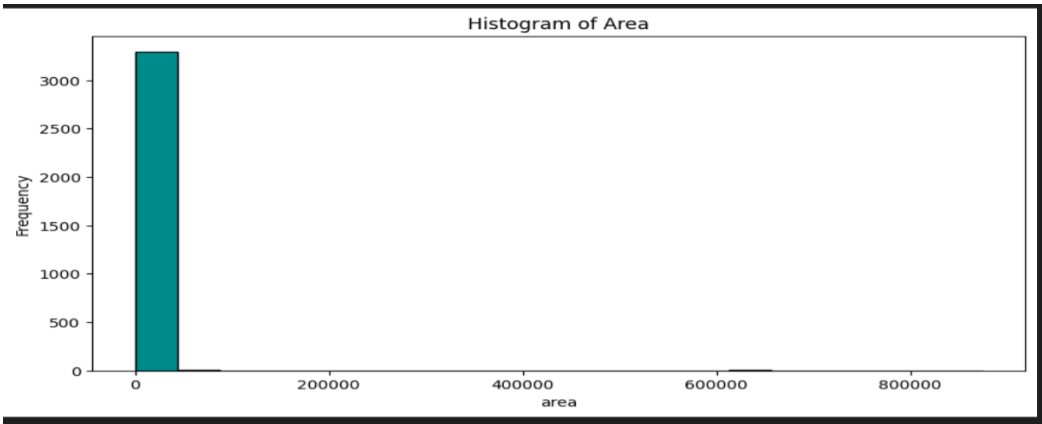
Kurtosis of price_per_sqft column: 307.058982662746

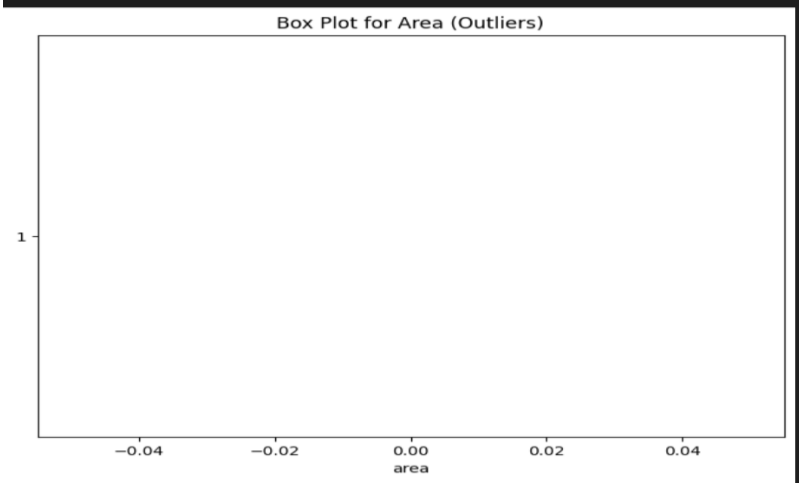
Exploring Area column:

Data type = float64
Missing values = 13

```
Statistics for the 'area' column:
count      3785.000000
mean       2845.999472
std       22783.349053
min          50.000000
25%        1220.000000
50%        1725.000000
75%        2295.000000
max      875000.000000
Name: area, dtype: float64
```

Histogram and box plot of area column:





Skewness of area column: 29.332399715868213
Kurtosis of area column: 901.6388377495501

Exploring ‘area with type ‘:

Data type = object

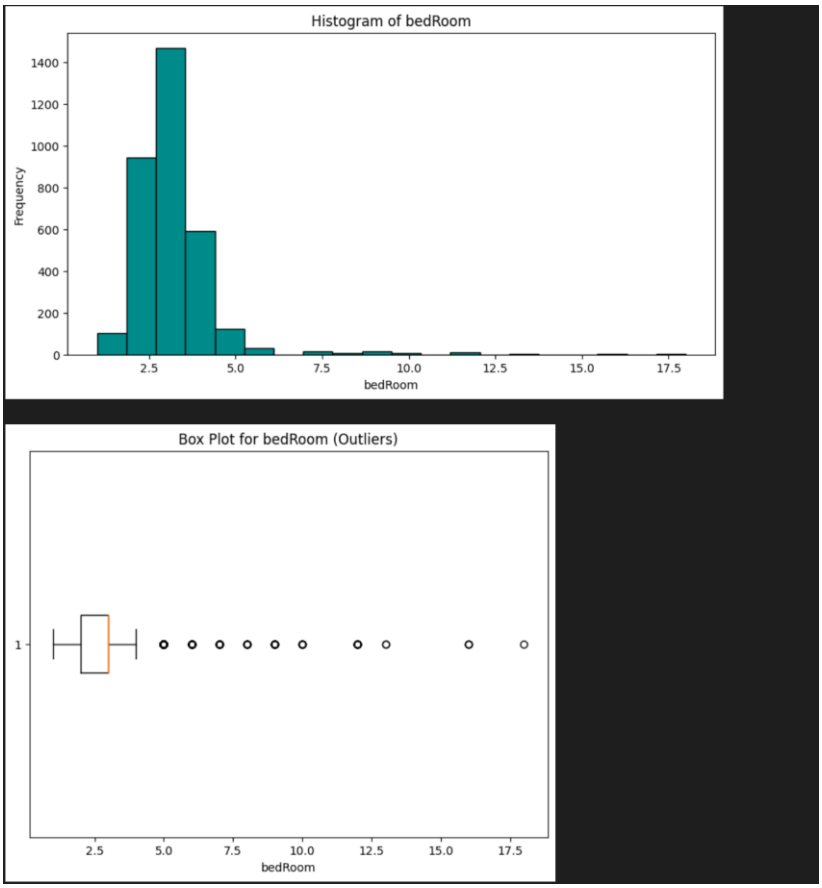
```
Unique type names and their counts:
areaWithType
Plot area 360(301.01 sq.m.)                                                                                    20
Super Built up area 1350(125.42 sq.m.)                                                                         17
Super Built up area 1578(146.6 sq.m.)                                                                          17
Super Built up area 1950(181.16 sq.m.)Carpet area: 1161 sq.ft. (107.86 sq.m.)                                  17
Super Built up area 1650(153.29 sq.m.)Carpet area: 1022.58 sq.ft. (95 sq.m.)                                   15
                                                                                                               ..
Super Built up area 2950(274.06 sq.m.)Built Up area: 2700 sq.ft. (250.84 sq.m.)Carpet area: 2500 sq.ft. (232.26 sq.m.)     1
Super Built up area 2185(202.99 sq.m.)Built Up area: 2140 sq.ft. (198.81 sq.m.)Carpet area: 2000 sq.ft. (185.81 sq.m.)     1
Plot area 3240(301.01 sq.m.)Built Up area: 7500 sq.ft. (696.77 sq.m.)Carpet area: 6000 sq.ft. (557.42 sq.m.)              1
Built Up area: 3150 (292.64 sq.m.)                                                                             1
Super Built up area 583(54.16 sq.m.)Carpet area: 483 sq.ft. (44.87 sq.m.)                                      1
Name: count, Length: 2121, dtype: int64
```

Exploring ‘Bedroom’ column:

Data type = int64
Missing value = 0

```
Statistics for the 'bedRoom' column:
count    3803.000000
mean        3.338154
std         1.876734
min         1.000000
25%         2.000000
50%         3.000000
75%         4.000000
max        21.000000
Name: bedRoom, dtype: float64
```

Plotting for bedroom:



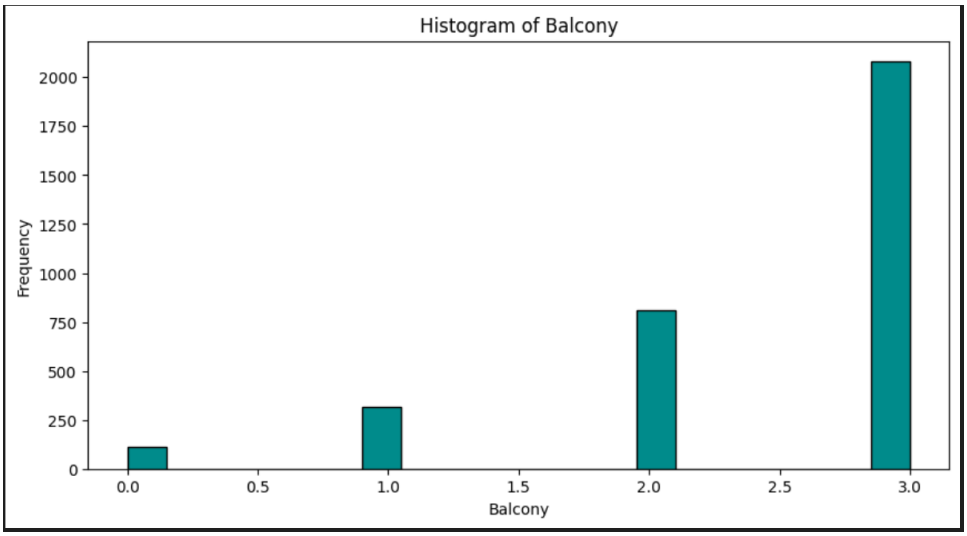
Skewness of bedRoom column: 3.5259893532780717
Kurtosis of bedRoom column: 24.837305345548558

Exploring ‘Balcony’ column:

Data type: object (categorical)

Missing values: 0

```
Statistics for the 'balcony' column:
count     3803
unique       5
top         3+
freq      1202
Name: balcony, dtype: object
```



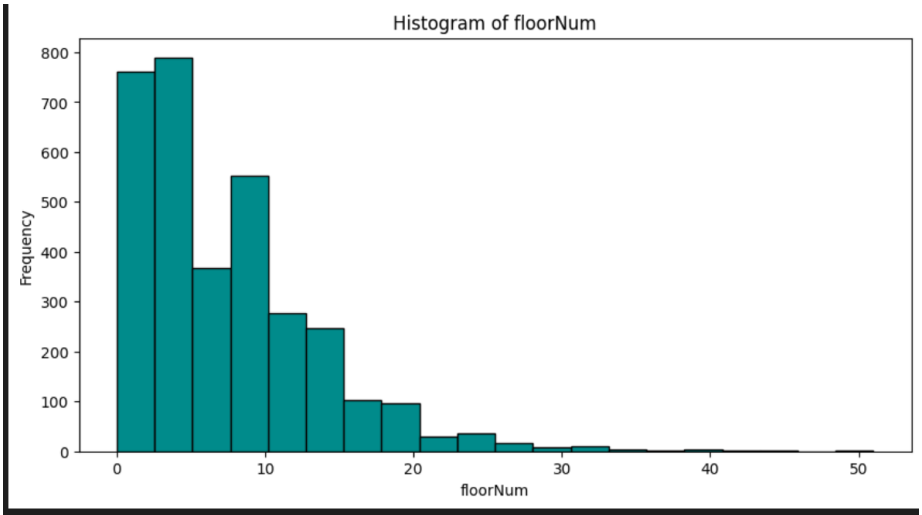
Skewness of balcony column: -1.4289383099551483
Kurtosis of balcony column: 1.2563254843960472

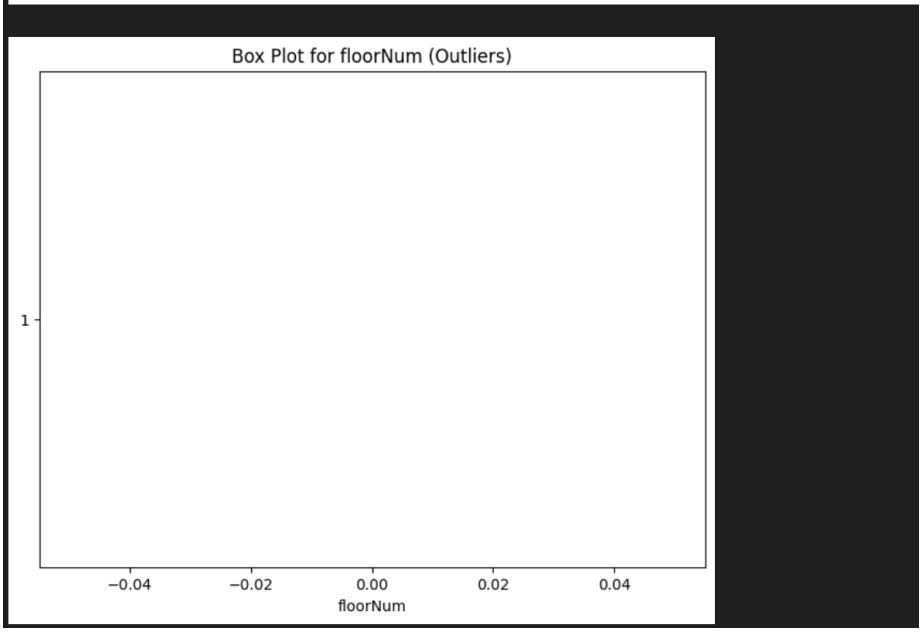
Exploring ‘FloorNum’ column:

Data type: float64
Number of missing values: 10

```
Statistics for the 'floorNum' column:
count    3784.000000
mean        6.810254
std         6.027555
min         0.000000
25%         2.000000
50%         5.000000
75%        10.000000
max        51.000000
Name: floorNum, dtype: float64
```

Plot for ‘FloorNo’:





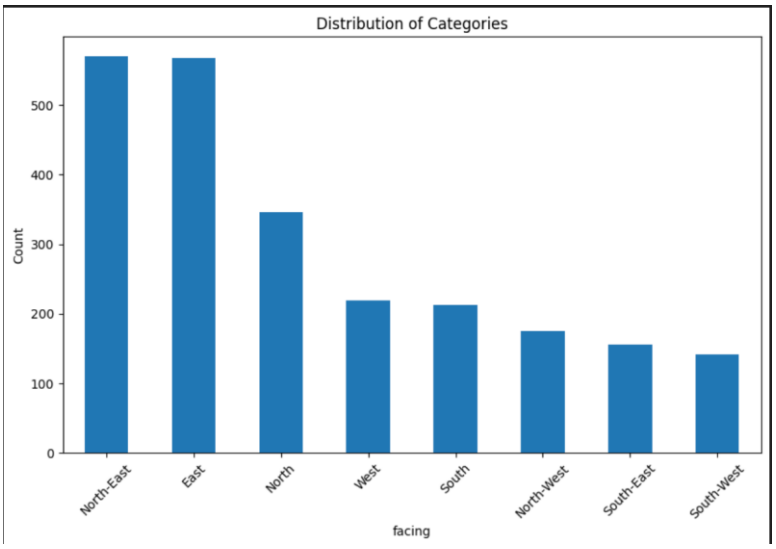
Exploring ‘Facing’ column:

Data type: Object

Unique names and their counts in facing column

```
Unique type names and their counts:
facing
North-East    570
East          567
North         346
West          219
South         213
North-West    175
South-East    156
South-West    142
Name: count, dtype: int64
```

```
Statistics for the 'facing' column:
count     2698
unique       8
top       East
freq       642
Name: facing, dtype: object
```



Exploration of ‘agePossesion’ column:

Data type: object
Missing values: 0

```
Unique type names and their counts:
agePossession
Relatively New        1575
New Property           581
Moderately Old         454
Under Construction     282
Undefined              237
Old Property           188
Name: count, dtype: int64
```

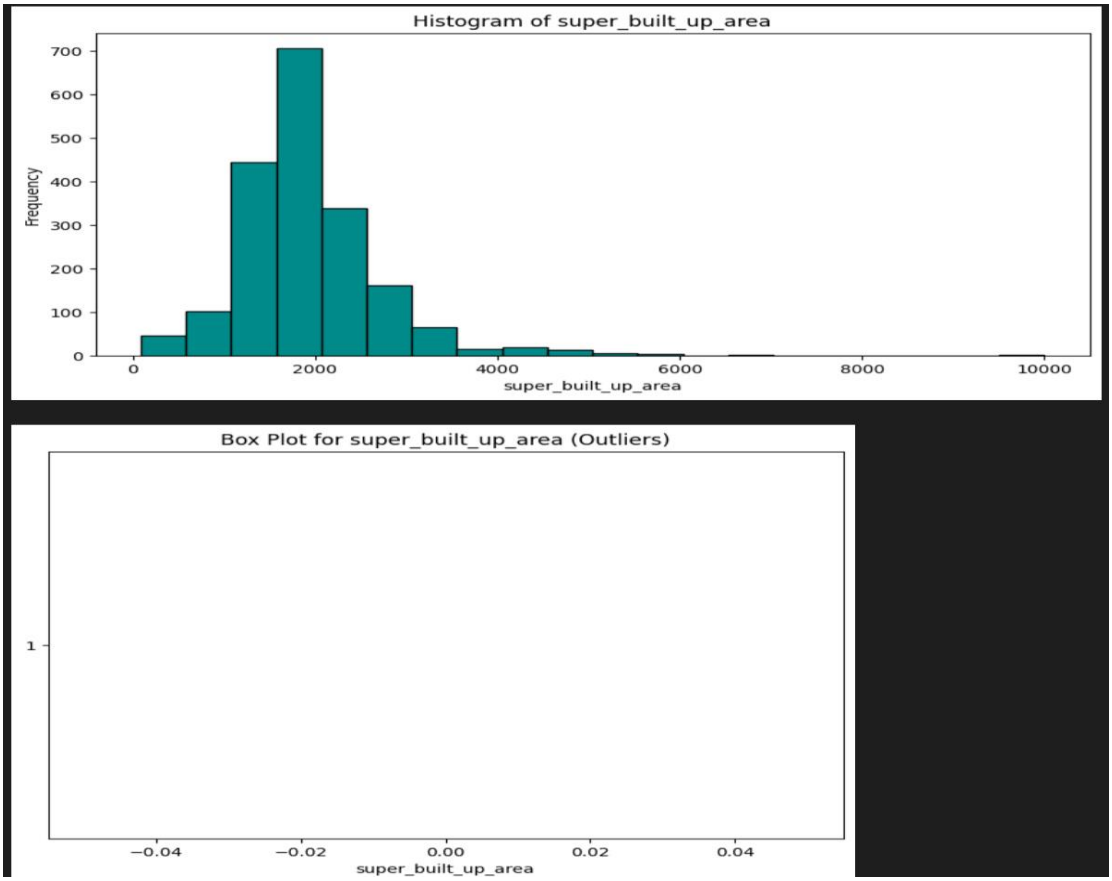
Histogram for age possession:

Exploration of ‘Super_built_up_area’ column:

Data type: float64
Number of missing values:1402

```
Statistics for the 'super_built_up_area' column:
count     1915.000000
mean      1921.658251
std        767.160169
min         89.000000
25%       1457.000000
50%       1828.000000
75%       2215.000000
max      10000.000000
Name: super_built_up_area, dtype: float64
```

Plot for the uper_built_up_area:

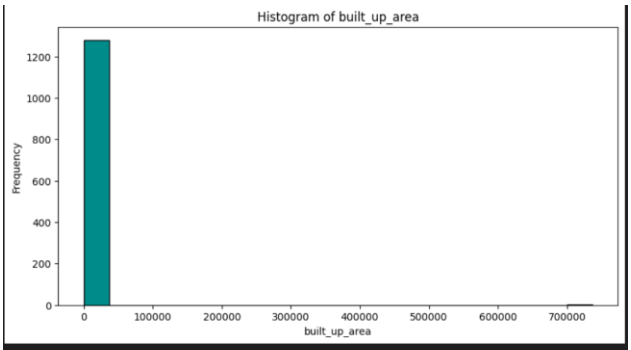


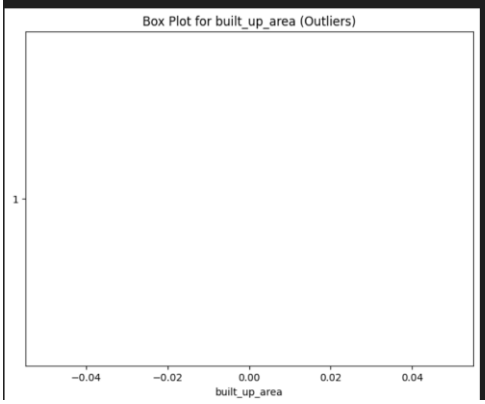
Skewness of super_built_up_area column: 1.8232284983476958
Kurtosis of super_built_up_area column: 10.083066100658108

Exploring ‘Built_up_area’ column:

Data type: float64
No of missing values: 2037

```
Statistics for the 'built_up_area' column:
count      1733.000000
mean       2360.241413
std       17719.603378
min           2.000000
25%        1100.000000
50%        1650.000000
75%        2399.000000
max      737147.000000
Name: built_up_area, dtype: float64
```





Skewness of built_up_area column: 35.549876877488984

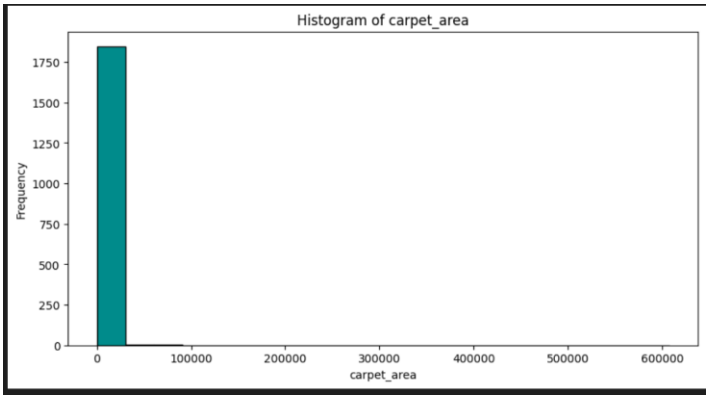
Kurtosis of built_up_area column: 1269.13770135010

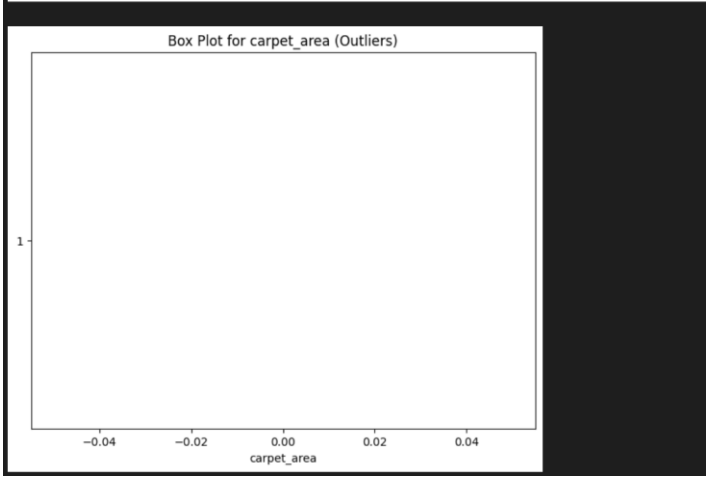
Exploring ‘carpet_area’ column:

Data type: float64

No of massing values :1461

```
Statistics for the 'carpet_area' column:
count      1944.000000
mean       2483.466943
std       22375.239293
min          15.000000
25%         824.000000
50%        1294.000000
75%        1786.250000
max      607936.000000
Name: carpet_area, dtype: float64
```





Skewness of carpet_area column: 24.242399007739884

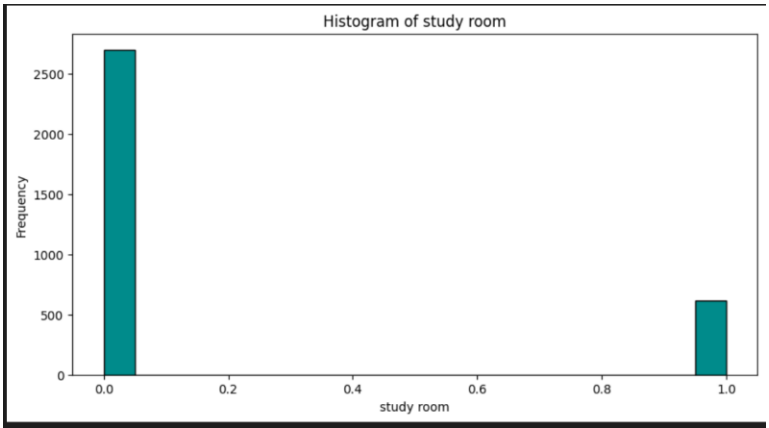
Kurtosis of carpet_area column: 599.7990545259844

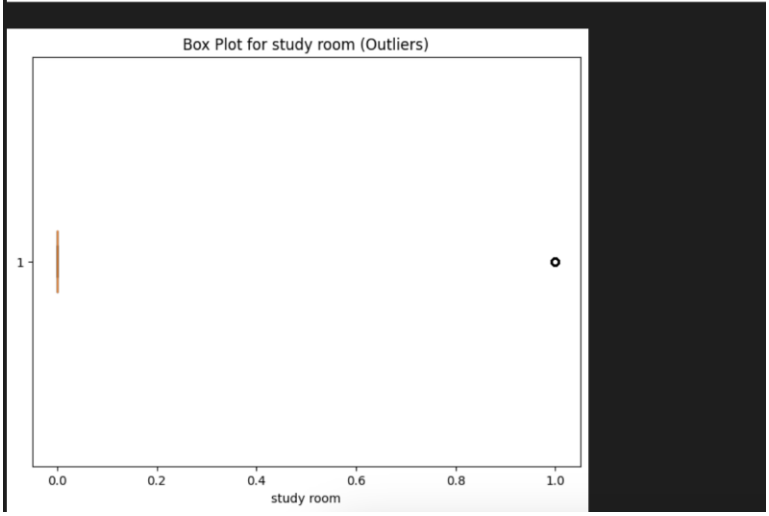
Exploring ‘study room’ column:

Data type:int64

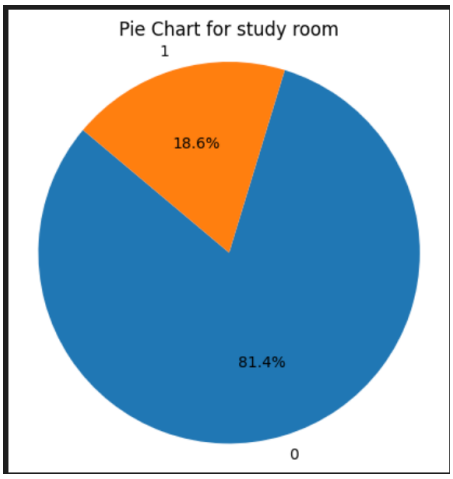
No of missing values :0

```
Statistics for the 'study room' column:
count    3803.000000
mean        0.189587
std         0.392026
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: study room, dtype: float64
```






```python
# Create a pie chart
plt.figure(figsize=(5, 5))
plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=140)
plt.title(f'Pie Chart for {column_name}')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

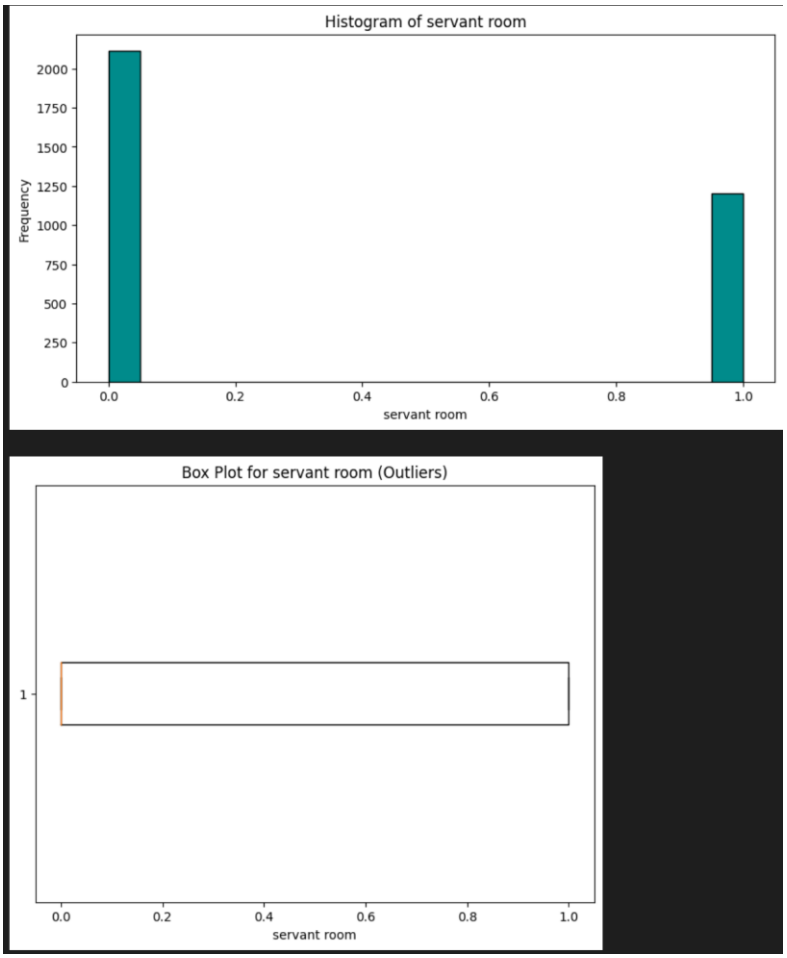


Skewness of study room column: 1.6145863249036156 Kurtosis of study room column: 0.6072547839591671

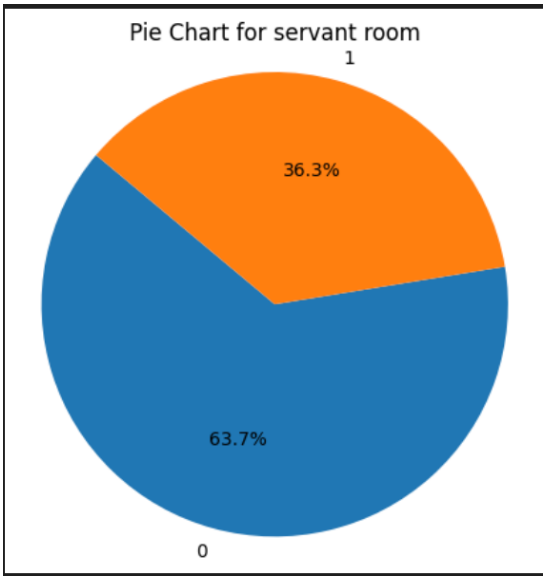
Exploration of ‘servant room’ column:

Data type:int64
No of missing values :0

```
Statistics for the 'servant room' column:
count    3803.000000
mean        0.356824
std         0.479125
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
Name: servant room, dtype: float64
```



Similarly, we can plot pie chart for servant room, storeroom, Pooja room by replacing column name in above mentioned pie chat



Skewness of servant room column: 0.5701608000517789

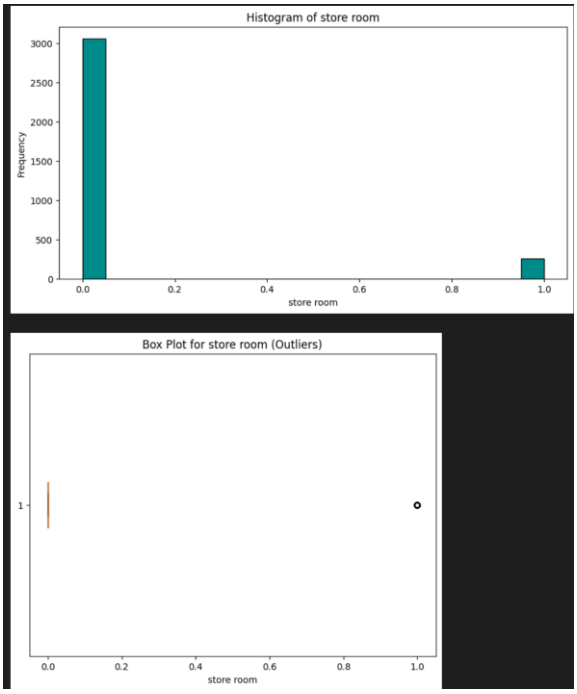
Kurtosis of servant room column: -1.675927533941474

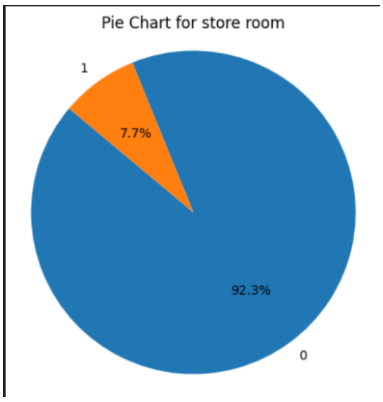
Exploring ‘storeroom’ column:

Data type: Int64
No of missing values:0

```
Statistics for the 'store room' column:
count    3803.000000
mean        0.090455
std         0.286870
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: store room, dtype: float64
```

Histogram and boxplot





Skewness of storeroom column: 3.1622214977560374

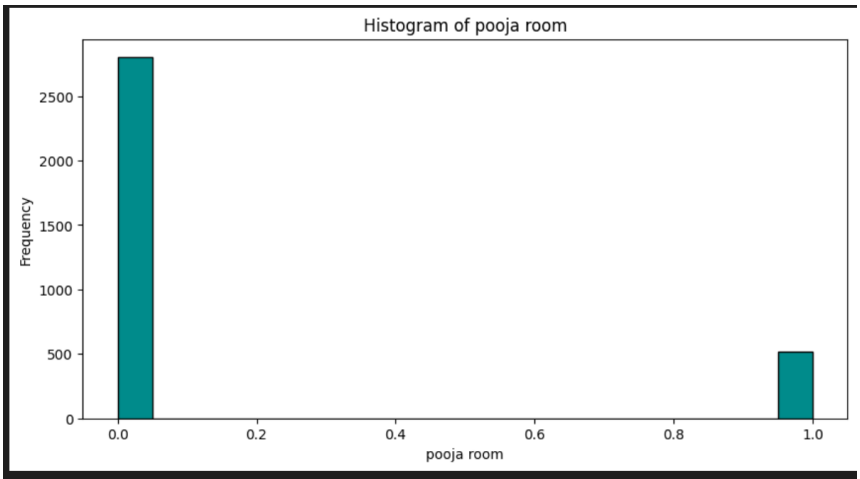
Kurtosis of storeroom column: 8.004470769570238

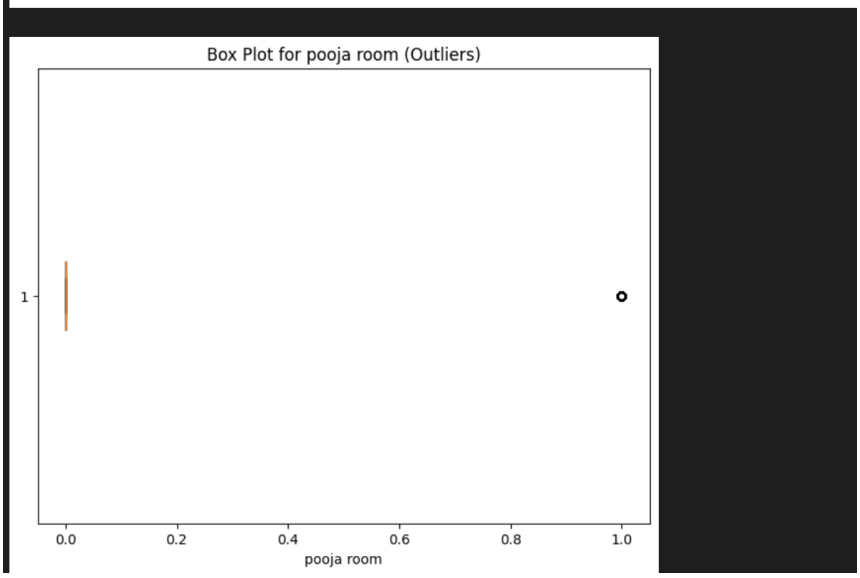
Exploring ‘Pooja room ‘column:

Data type: Int64

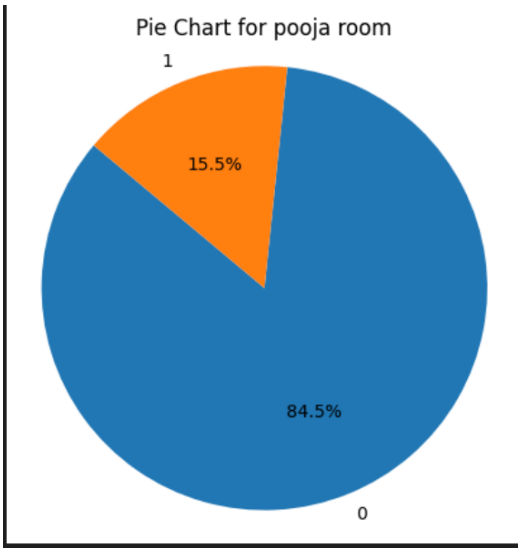
No of missing values:0

```
Statistics for the 'pooja room' column:
count    3803.000000
mean        0.174336
std         0.379448
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: pooja room, dtype: float64
```





Pie chart:

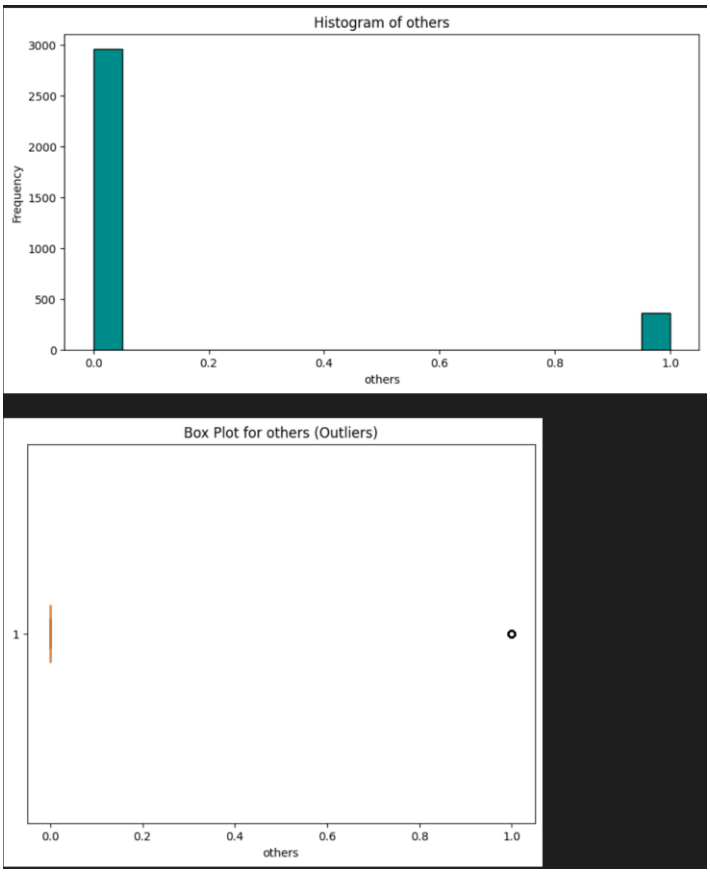


Skewness of pooja room column: 1.904692837124777

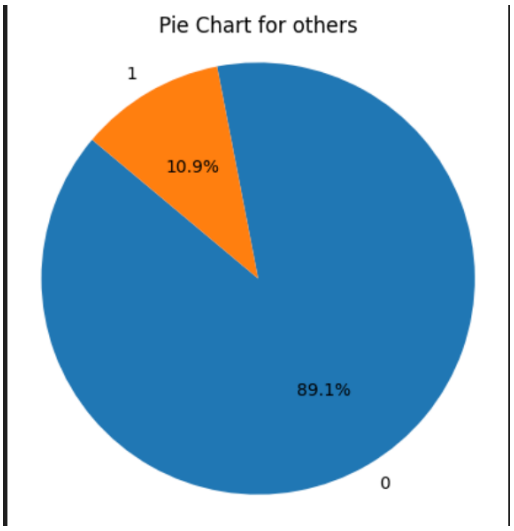
Kurtosis of pooja room column: 1.6288365546020058

Exploring ‘others 'column:
Data type: int64
No of missing value :0

```
Statistics for the 'others' column:
count    3803.000000
mean        0.110702
std         0.313804
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: others, dtype: float64
```



Pie chart:



Skewness of others column: 2.518207429726182

Kurtosis of others column: 4.343987522665149

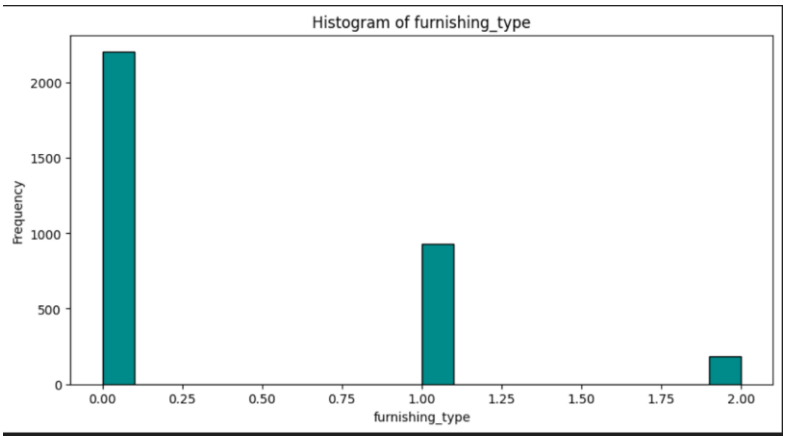
Exploring ‘Furnishing_type’:

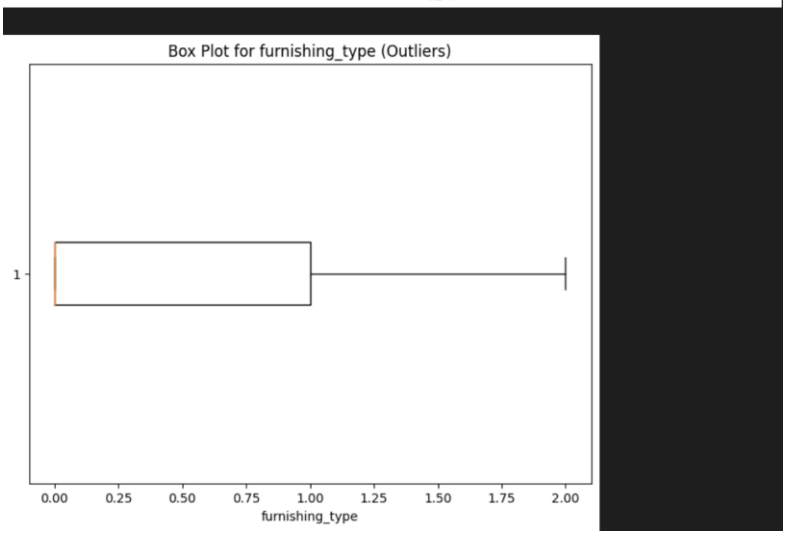
Data type: int64

No of missing values: 0

```
Statistics for the 'furnishing_type' column:
count    3803.000000
mean        0.397055
std         0.594214
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         2.000000
Name: furnishing_type, dtype: float64
```

Plotting histogram and box plot:





Skewness of furnishing_type column: 1.235113979116794

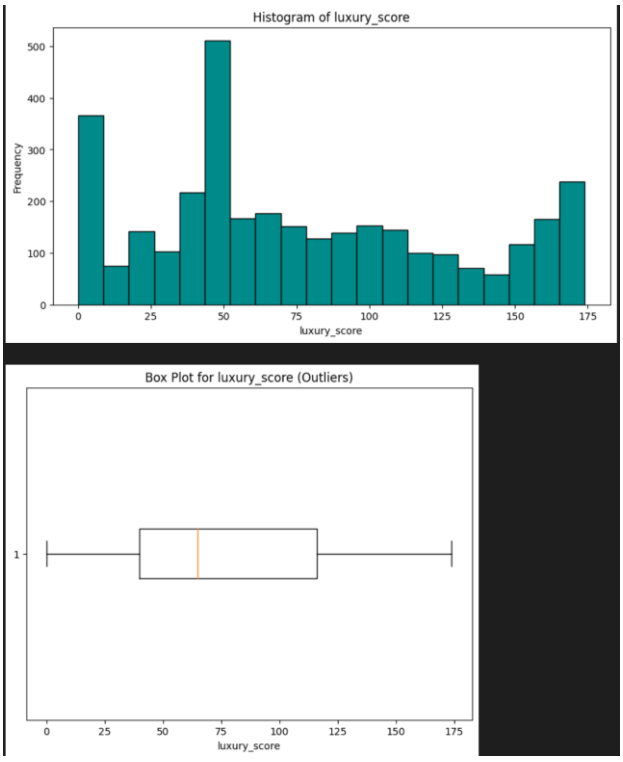
Kurtosis of furnishing_type column: 0.500139485372948

Exploring ‘Luxury score’ column:

Data type: Int64
No of missing values :0

```
Statistics for the 'luxury_score' column:
count    3803.000000
mean       70.947936
std        52.821789
min         0.000000
25%        31.000000
50%        58.000000
75%       109.000000
max       174.000000
Name: luxury_score, dtype: float64
```

Histogram and box plot graphs for luxury score:



Skewness of luxury_score column: 0.3840053193828689
Kurtosis of luxury_score column: -0.9275057995402518

Multivariate analysis:

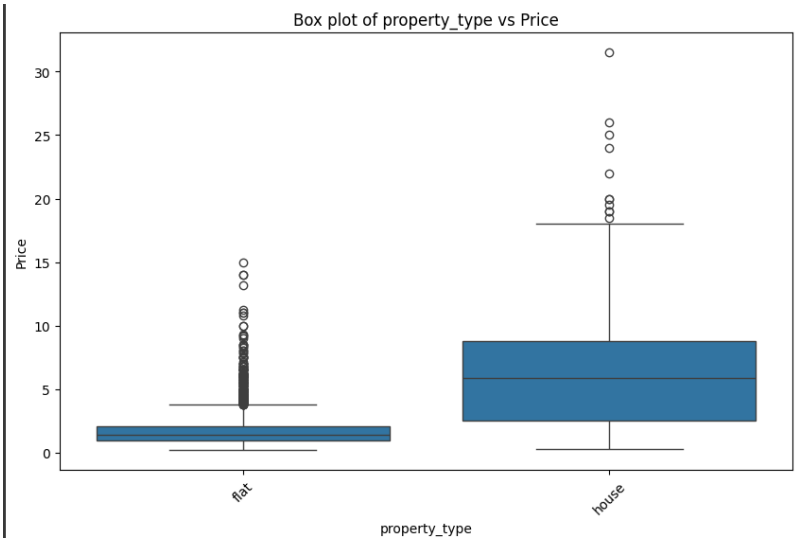
Multivariate Analysis of Features and Price

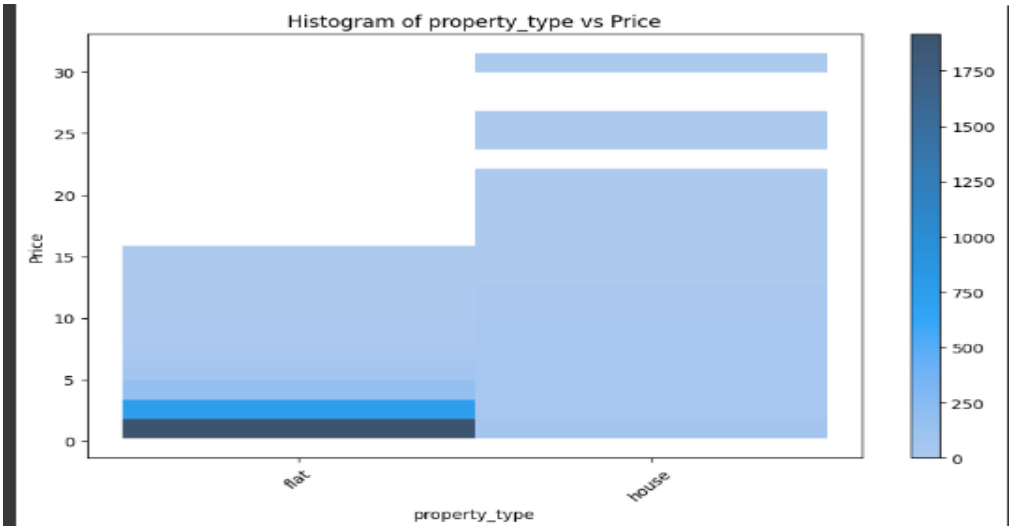
Box plots, histograms, and scatter plots are used in this analysis to study the relationship between categorical (ex: property type) and numerical (ex: area) characteristics and property prices.

Property Type vs Price:

Box Plot: Displays price distribution for various property categories.

Histogram: Shows price density for each property type.

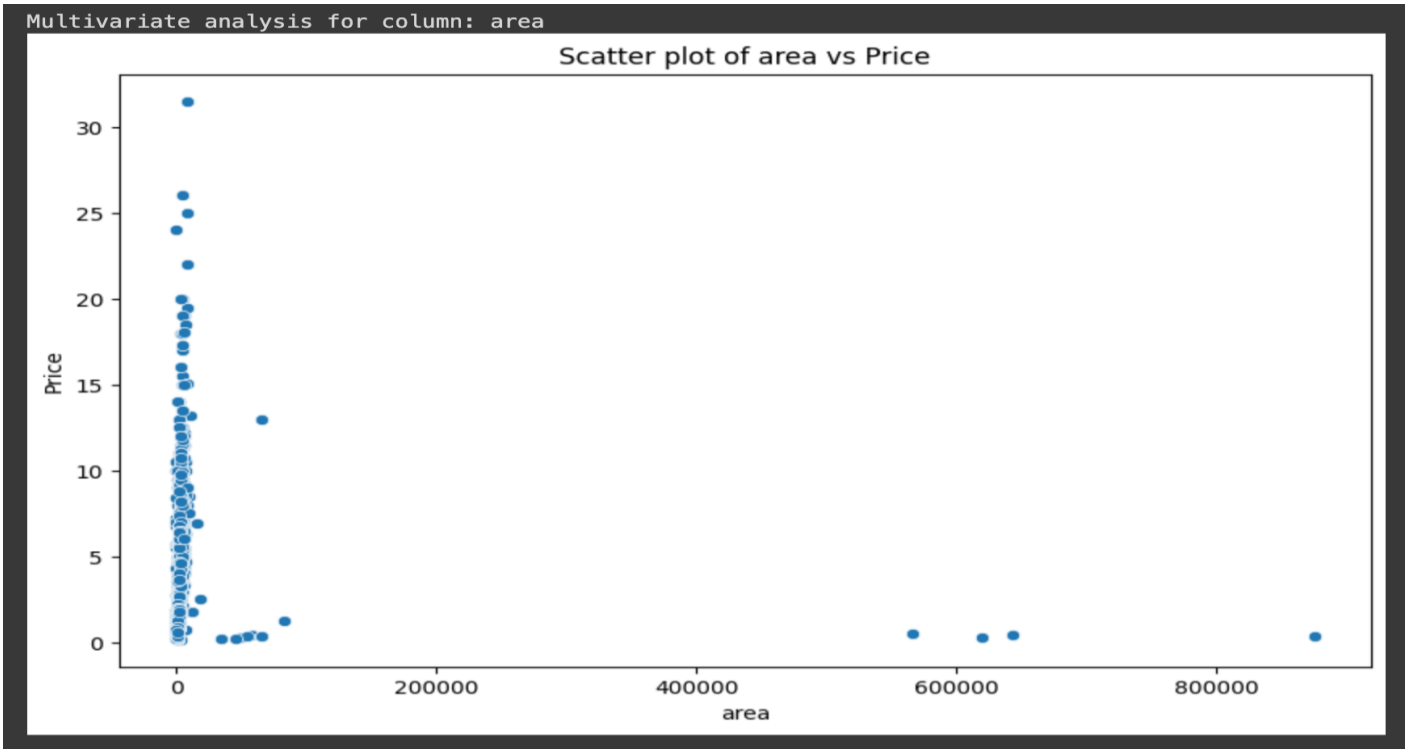




Area vs Price:

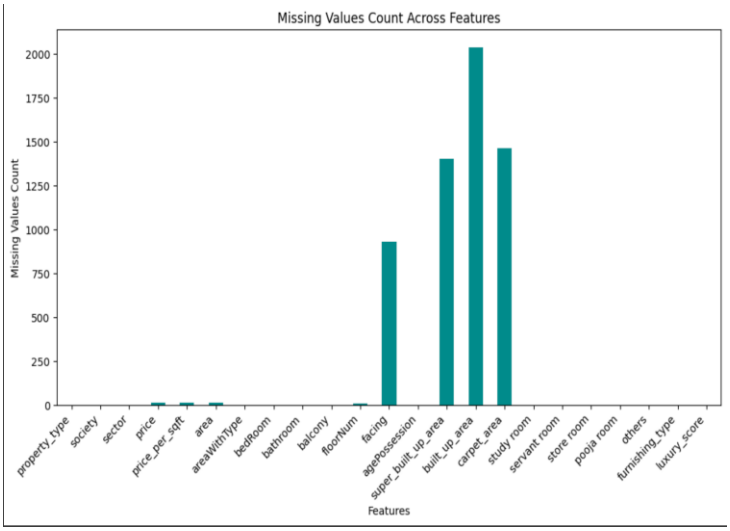
Scatter Plot: Shows the correlation between property size and cost.

These visualizations provide insights on price patterns and feature correlations, allowing for data-driven decision-making.



Missing values count across features (columns)

The Missing_Values_Data method finds and summarizes missing values from the new_data DataFrame. It creates a bar plot of the number of missing values for each feature and provides a summary of the features with missing data. This helps to understand data quality and guide preparation efforts.




```
Missing Values Summary:
society                 1
price                  13
price_per_sqft         13
area                   13
floorNum               10
facing                929
super_built_up_area    1402
built_up_area          2037
carpet_area            1461
dtype: int64
```

TASK – 2:

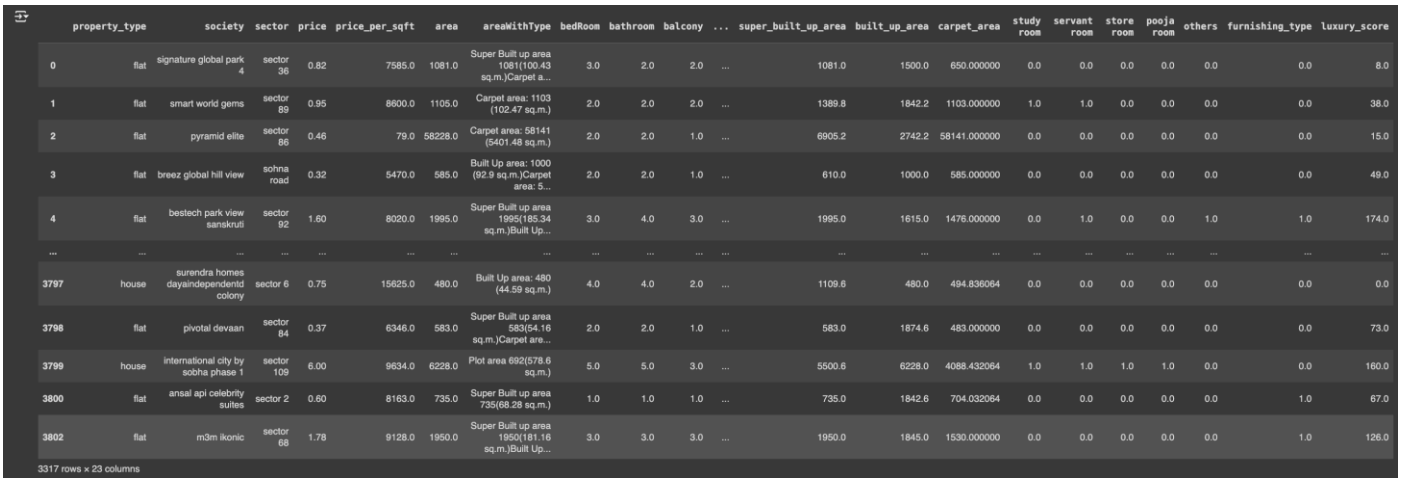
The goal was to handle missing values in the new_data DataFrame with appropriate imputation algorithms for numerical and categorical attributes. The data is chosen for imputation, and the KNN Imputer with 5 neighbors was used to fill the missing values, relying on the similarity of the nearest neighbors for estimation. Missing values in categorical columns with the object data type were filled using each feature's mode. ensuring that missing values were replaced by the most frequent category. After the imputation process, a check confirmed that the total number of missing values was reduced to zero, indicating successful handling of all missing data. The cleaned dataframe super_data is defined as cleaned data it has zero (0) missing values. This is used for further research.

This method ensures that super_data was complete and ready for further data processing operations. All missing values in new_data were successfully imputed, giving rise to a complete and clean dataset saved as super_data.

```python
import pandas as pd
from sklearn.impute import KNNImputer


numerical_features = new_data.select_dtypes(include=['int64', 'float64']).columns
knn_imputer = KNNImputer(n_neighbors=5)
new_data[numerical_features] = knn_imputer.fit_transform(new_data[numerical_features])


categorical_features = new_data.select_dtypes(include=['object']).columns
for feature in categorical_features:
    new_data[feature].fillna(new_data[feature].mode()[0], inplace=True)

missing_values_after_handling = new_data.isnull().sum().sum()
print("Total missing values after handling:", missing_values_after_handling)
super_data = new_data.copy()
```

```
Total missing values after handling: 0
```

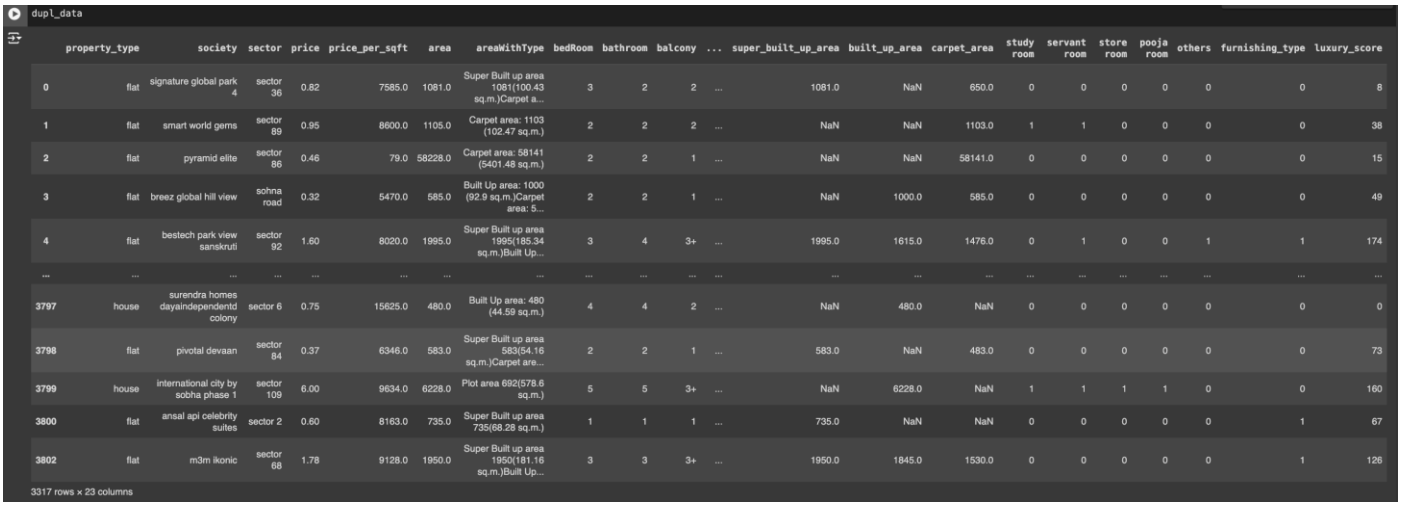
super_data : This data frame contains zero missing values



	property_type	society	sector	price	price_per_sqft	area	areaWithType	bedRoom	bathroom	balcony	...	super_built_up_area	built_up_area	carpet_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_score
0	flat	signature global park 4	sector 36	0.82	7585.0	1081.0	Super Built up area 1081(100.43 sq.m.)Carpet a...	3.0	2.0	2.0	...	1081.0	1500.0	650.000000	0.0	0.0	0.0	0.0	0.0	0.0	8.0
1	flat	smart world gems	sector 89	0.95	8600.0	1105.0	Carpet area: 1103 (102.47 sq.m.)	2.0	2.0	2.0	...	1389.8	1842.2	1103.000000	1.0	1.0	0.0	0.0	0.0	0.0	38.0
2	flat	pyramid elite	sector 86	0.46	79.0	58228.0	Carpet area: 58141 (5401.48 sq.m.)	2.0	2.0	1.0	...	6905.2	2742.2	58141.000000	0.0	0.0	0.0	0.0	0.0	0.0	15.0
3	flat	breez global hill view	sohna road	0.32	5470.0	585.0	Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...	2.0	2.0	1.0	...	610.0	1000.0	585.000000	0.0	0.0	0.0	0.0	0.0	0.0	49.0
4	flat	bestech park view sanskruti	sector 92	1.60	8020.0	1995.0	Super Built up area 1995(185.34 sq.m.)Built Up...	3.0	4.0	3.0	...	1995.0	1615.0	1476.000000	0.0	1.0	0.0	0.0	1.0	1.0	174.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3797	house	surendra homes dayaindependentd colony	sector 6	0.75	15625.0	480.0	Built Up area: 480 (44.59 sq.m.)	4.0	4.0	2.0	...	1109.6	480.0	494.836064	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3798	flat	pivotal devaan	sector 84	0.37	6346.0	583.0	Super Built up area 583(54.16 sq.m.)Carpet are...	2.0	2.0	1.0	...	583.0	1874.6	483.000000	0.0	0.0	0.0	0.0	0.0	0.0	73.0
3799	house	international city by sobha phase 1	sector 109	6.00	9634.0	6228.0	Plot area 692(578.6 sq.m.)	5.0	5.0	3.0	...	5500.6	6228.0	4088.432064	1.0	1.0	1.0	1.0	0.0	0.0	160.0
3800	flat	ansal api celebrity suites	sector 2	0.60	8163.0	735.0	Super Built up area 735(68.28 sq.m.)	1.0	1.0	1.0	...	735.0	1842.6	704.032064	0.0	0.0	0.0	0.0	0.0	1.0	67.0
3802	flat	m3m ikonic	sector 68	1.78	9128.0	1950.0	Super Built up area 1950(181.16 sq.m.)Built Up...	3.0	3.0	3.0	...	1950.0	1845.0	1530.000000	0.0	0.0	0.0	0.0	0.0	1.0	126.0

3317 rows × 23 columns

dupl_data = This is the data frame which is a copy of the new_data data frame which contains the missing values.



dupl_data

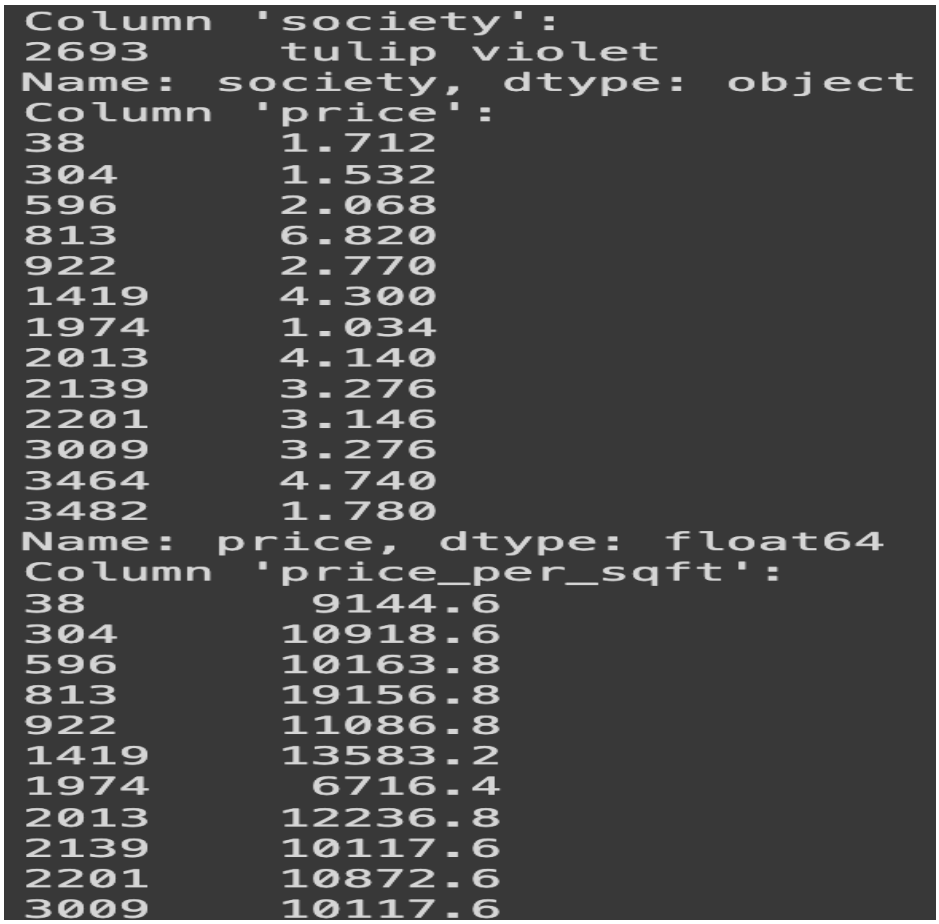
	property_type	society	sector	price	price_per_sqft	area	areaWithType	bedRoom	bathroom	balcony	...	super_built_up_area	built_up_area	carpet_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_score
0	flat	signature global park 4	sector 36	0.82	7585.0	1081.0	Super Built up area 1081(100.43 sq.m.)Carpet a...	3	2	2	...	1081.0	NaN	650.0	0	0	0	0	0	0	8
1	flat	smart world gems	sector 89	0.95	8600.0	1105.0	Carpet area: 1103 (102.47 sq.m.)	2	2	2	...	NaN	NaN	1103.0	1	1	0	0	0	0	38
2	flat	pyramid elite	sector 86	0.46	79.0	58228.0	Carpet area: 58141 (5401.48 sq.m.)	2	2	1	...	NaN	NaN	58141.0	0	0	0	0	0	0	15
3	flat	breez global hill view	sohna road	0.32	5470.0	585.0	Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...	2	2	1	...	NaN	1000.0	585.0	0	0	0	0	0	0	49
4	flat	bestech park view sanskruti	sector 92	1.60	8020.0	1995.0	Super Built up area 1995(185.34 sq.m.)Built Up...	3	4	3+	...	1995.0	1615.0	1476.0	0	1	0	0	1	1	174
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3797	house	surendra homes dayaindependentd colony	sector 6	0.75	15625.0	480.0	Built Up area: 480 (44.59 sq.m.)	4	4	2	...	NaN	480.0	NaN	0	0	0	0	0	0	0
3798	flat	pivotal devaan	sector 84	0.37	6346.0	583.0	Super Built up area 583(54.16 sq.m.)Carpet are...	2	2	1	...	583.0	NaN	483.0	0	0	0	0	0	0	73
3799	house	international city by sobha phase 1	sector 109	6.00	9634.0	6228.0	Plot area 692(578.6 sq.m.)	5	5	3+	...	NaN	6228.0	NaN	1	1	1	1	0	0	160
3800	flat	ansal api celebrity suites	sector 2	0.60	8163.0	735.0	Super Built up area 735(68.28 sq.m.)	1	1	1	...	735.0	NaN	NaN	0	0	0	0	0	1	67
3802	flat	m3m ikonic	sector 68	1.78	9128.0	1950.0	Super Built up area 1950(181.16 sq.m.)Built Up...	3	3	3+	...	1950.0	1845.0	1530.0	0	0	0	0	0	1	126

3317 rows × 23 columns

```python
def print_changed_values(super_data,dupl_data):
    for column in super_data.columns:
        changed_values = super_data.loc[super_data[column] != dupl_data[column], column]
        if not changed_values.empty:
            print(f"Column '{column}':")
            print(changed_values)

print_changed_values( super_data,dupl_data)
```

The print_changed_values function was used to compare the super_data DataFrame, which had gone through imputation to address missing values, to the original duplicated DataFrame, dupl_data.This function found and reported the values in each column that differed between super_data and dupl_data. This comparison showed the specific modifications made throughout the imputation

process, allowing for a clear understanding of how missing data was treated. The output only included columns that changed, resulting in a concentrated and meaningful overview imputed values



```
3464      15268.6
3482       8807.6
Name: price_per_sqft, dtype:
Column 'area':
38         1935.8
304        1681.2
596        2034.0
813        3239.0
922        2461.2
1419       3182.4
1974       1513.2
2013       2864.2
2139       3130.4
2201       2499.4
3009       3130.4
3464       2808.0
3482       1972.2
Name: area, dtype: float64
Column 'balcony':
0          2.0
1          2.0
2          1.0
3          1.0
4          3.0
          ...
3797       2.0
3798       1.0
3799       3.0
3800       1.0
3802       3.0
```

```
Name: balcony, Length: 3317, dtype: float64
Column 'floorNum':
975      4.2
1528     4.2
1541     6.6
2058     3.2
2166     4.2
2405     4.4
2586    14.2
2693     5.6
2702     5.2
3432     7.4
Name: floorNum, dtype: float64
Column 'facing':
0       North-East
1       North-East
2       North-East
3       North-East
22      North-East
           ...
3782    North-East
3785    North-East
3788    North-East
3795    North-East
3797    North-East
Name: facing, Length: 929, dtype: object
Column 'super_built_up_area':
1       1389.8
2       6905.2
3        610.0
```

```
10      2595.4
12      1511.2
         ...
3791    2308.8
3792    1857.8
3793     637.8
3797    1109.6
3799    5500.6
Name: super_built_up_area, Length: 1402, dtype: float64
Column 'built_up_area':
0       1500.0
1       1842.2
2       2742.2
5       1626.0
6       1808.6
         ...
3792    2054.2
3793    2742.2
3795    1874.6
3798    1874.6
3800    1842.6
Name: built_up_area, Length: 2037, dtype: float64
Column 'carpet_area':
6       3791.232064
7       1749.232064
8       1666.432064
10       906.769822
13      1847.432064
           ...
3790     596.610064
3791    1308.432064
3797     494.836064
3799    4088.432064
3800     704.032064
Name: carpet_area, Length: 1461, dtype: float64
```

After addressing the missing values, the total number of missing values in the super_data DataFrame was decreased to zero, showing that the imputation was successful. The number of observations remained constant, as both the original data and the imputed super_data had the same number of rows, assuring no data loss during the process. The initial dataset (data) was compared to the cleaned dataset (super_data), and the head of each DataFrame was printed to confirm the modifications. The original data contained missing values; however, the handled data revealed a complete and imputed dataset, indicating the efficiency of the imputation techniques used.

```python
missing_values_after_handling = super_data.isnull().sum().sum()
print("Total missing values after handling:", missing_values_after_handling)

num_observations_before = len(data)
num_observations_after = len(super_data)
print("Number of observations before handling:", num_observations_before)
print("Number of observations after handling:", num_observations_after)


print("\nOriginal Data:")
print(data.head())

print("\nHandled Data:")
print(super_data.head())
```

```
Total missing values after handling: 0
Number of observations before handling: 3803
Number of observations after handling: 3317

Original Data:
  property_type                    society     sector  price  \
0         flat     signature global park 4   sector 36   0.82
1         flat            smart world gems   sector 89   0.95
2         flat               pyramid elite   sector 86   0.46
3         flat      breez global hill view  sohna road   0.32
4         flat  bestech park view sanskruti   sector 92   1.60

   price_per_sqft     area                                     areaWithType  \
0          7585.0   1081.0  Super Built up area 1081(100.43 sq.m.)Carpet a...
1          8600.0   1105.0                Carpet area: 1103 (102.47 sq.m.)
2            79.0  58228.0              Carpet area: 58141 (5401.48 sq.m.)
3          5470.0    585.0  Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...
4          8020.0   1995.0  Super Built up area 1995(185.34 sq.m.)Built Up...

   bedRoom  bathroom balcony  ...  super_built_up_area built_up_area  \
0        3         2       2  ...               1081.0          NaN
1        2         2       2  ...                  NaN          NaN
2        2         2       1  ...                  NaN          NaN
3        2         2       1  ...                  NaN       1000.0
4        3         4      3+  ...               1995.0       1615.0
```

```
   carpet_area  study room  servant room  store room  pooja room  others  \
0        650.0           0             0           0           0       0
1       1103.0           1             1           0           0       0
2      58141.0           0             0           0           0       0
3        585.0           0             0           0           0       0
4       1476.0           0             1           0           0       1

    furnishing_type  luxury_score
0                 0             8
1                 0            38
2                 0            15
3                 0            49
4                 1           174

[5 rows x 23 columns]

Handled Data:
   property_type                   society     sector  price  \
0           flat     signature global park 4   sector 36   0.82
1           flat           smart world gems   sector 89   0.95
2           flat              pyramid elite   sector 86   0.46
3           flat       breez global hill view  sohna road   0.32
4           flat  bestech park view sanskruti   sector 92   1.60
```

```
   price_per_sqft     area                                      areaWithType  \
0          7585.0   1081.0  Super Built up area 1081(100.43 sq.m.)Carpet a...
1          8600.0   1105.0                   Carpet area: 1103 (102.47 sq.m.)
2            79.0  58228.0                 Carpet area: 58141 (5401.48 sq.m.)
3          5470.0    585.0  Built Up area: 1000 (92.9 sq.m.)Carpet area: 5...
4          8020.0   1995.0  Super Built up area 1995(185.34 sq.m.)Built Up...

   bedRoom  bathroom  balcony  ...  super_built_up_area built_up_area  \
0      3.0       2.0      2.0  ...               1081.0        1500.0
1      2.0       2.0      2.0  ...               1389.8        1842.2
2      2.0       2.0      1.0  ...               6905.2        2742.2
3      2.0       2.0      1.0  ...                610.0        1000.0
4      3.0       4.0      3.0  ...               1995.0        1615.0

  carpet_area  study room  servant room  store room  pooja room  others  \
0       650.0         0.0           0.0         0.0         0.0     0.0
1      1103.0         1.0           1.0         0.0         0.0     0.0
2     58141.0         0.0           0.0         0.0         0.0     0.0
3       585.0         0.0           0.0         0.0         0.0     0.0
4      1476.0         0.0           1.0         0.0         0.0     1.0

   furnishing_type  luxury_score
0              0.0           8.0
1              0.0          38.0
2              0.0          15.0
3              0.0          49.0
4              1.0         174.0

[5 rows x 23 columns]
```

TASK-3

The Z-score approach was used to identify outliers in the numerical features of the super_data DataFrame. By generating Z-scores for each value and finding those with an absolute Z-score greater than 3, the approach identified possible outliers. The indices of these outliers were presented, emphasizing observations that deviated greatly from the mean. This approach aids in identifying extreme numbers that may impact the study, allowing for additional inquiry or suitable handling of these outliers.

```
numerical_features = super_data.select_dtypes(include=['int64', 'float64'])

z_score_threshold = 3
z_scores = ((numerical_features - numerical_features.mean()) / numerical_features.std()).abs()
outliers_zscore = z_scores[(z_scores > z_score_threshold).any(axis=1)].index


print("Outliers detected by Z-score method:", outliers_zscore)
```

Output of the above code:

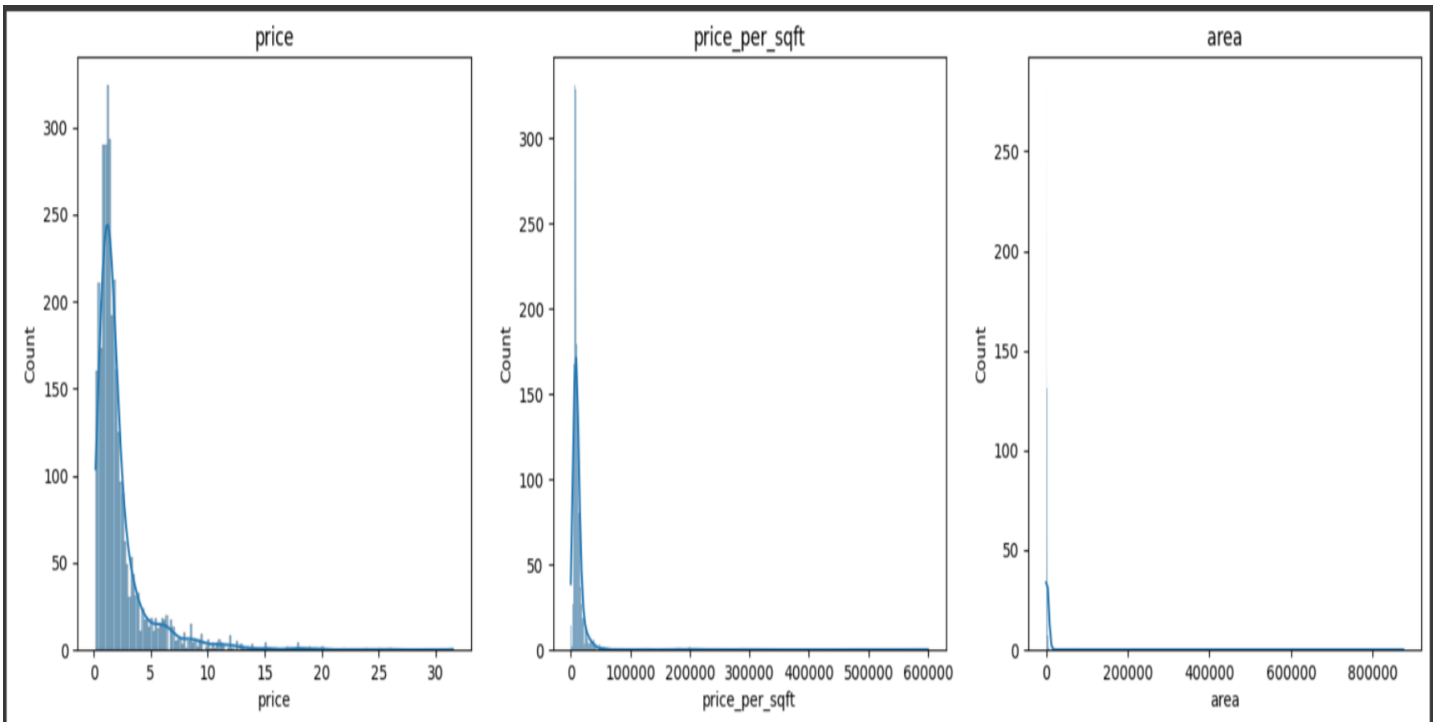
```
Outliers detected by Z-score method: Index([   2,    5,    6,   17,   20,   29,   30,   31,   33,   34,
       ...
       3731, 3736, 3741, 3746, 3750, 3752, 3754, 3791, 3799, 3802],
      dtype='int64', length=561)
```

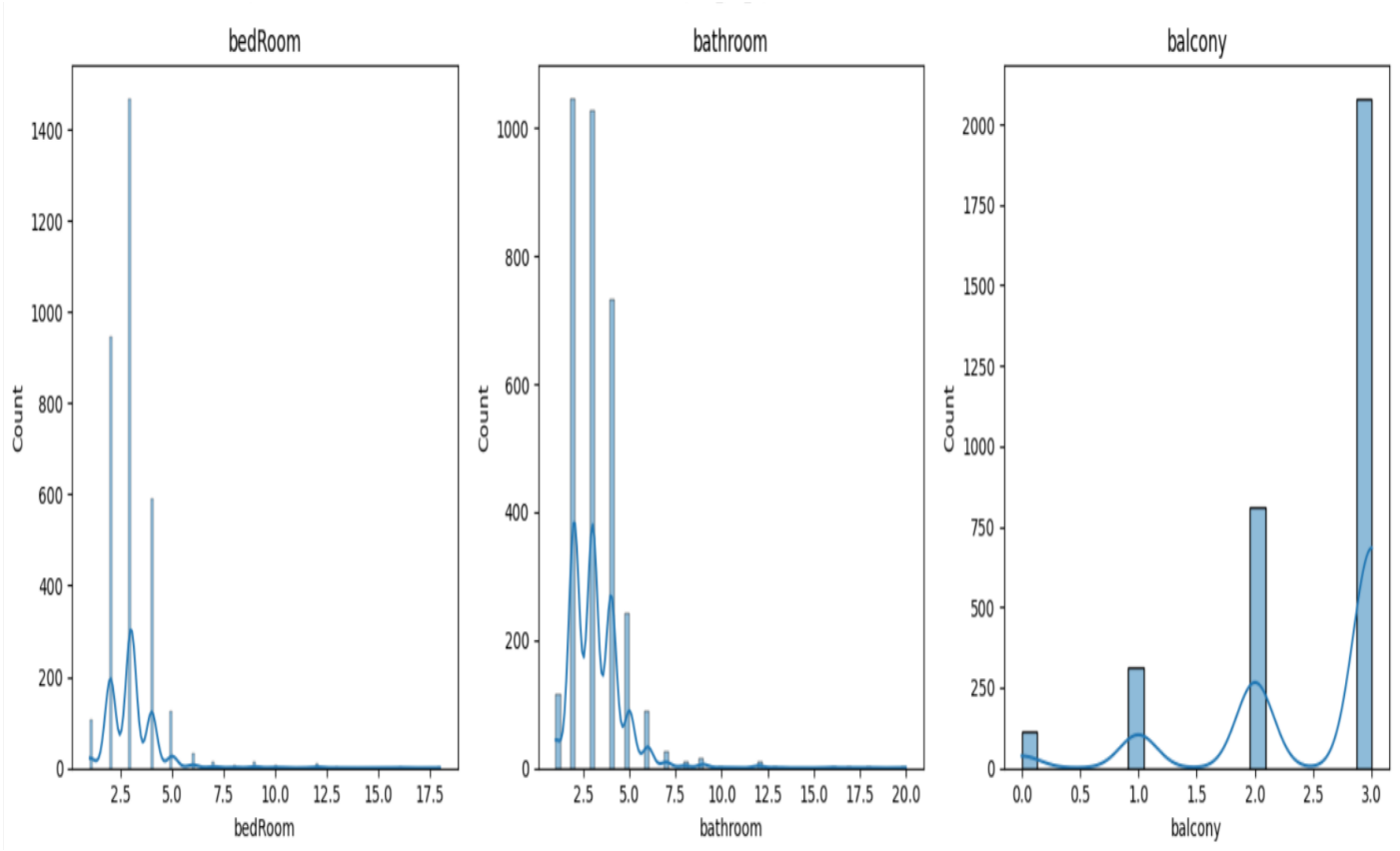
The numerical characteristics of the super_data Data Frames were evaluated using histogram and boxplot visualizations to determine distributions and identify probable outliers. Histograms using kernel density estimates (KDE) revealed the distribution forms of each numerical feature,

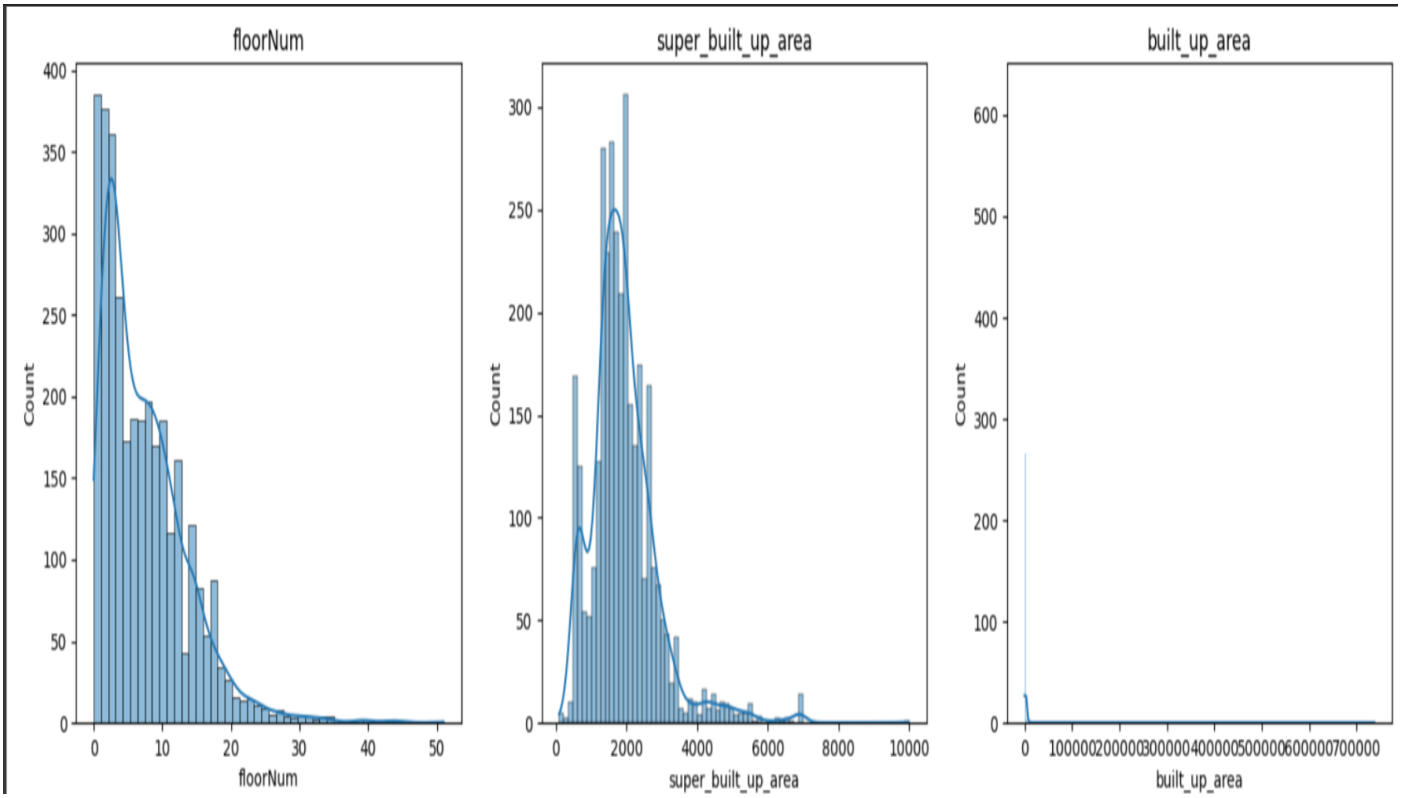
whilst boxplots showed outliers and data spread. To improve clarity, the plots were grouped into a grid format, with each feature given in its own subplot. Furthermore, descriptive statistics were computed and printed, providing an overview of central tendencies, dispersions, and general data features. This detailed examination allows for a better understanding of the numerical data's behavior and variability.

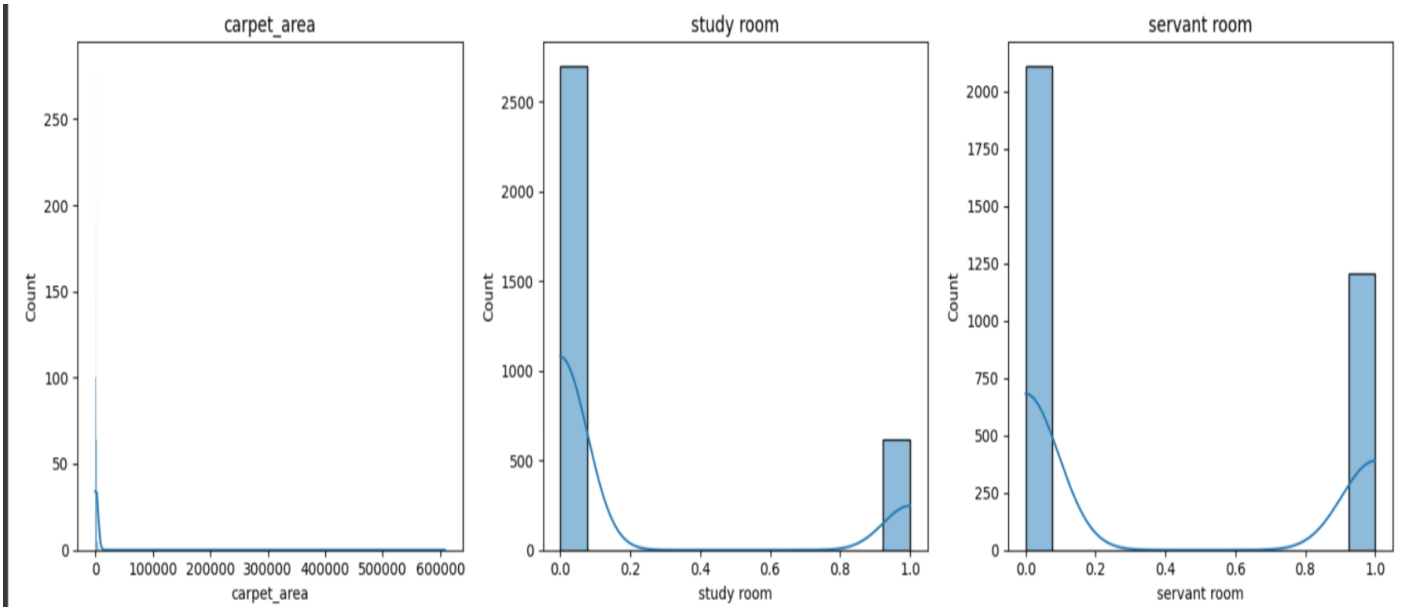
```
numerical_features = super_data.select_dtypes(include=['int64', 'float64'])

num_features = len(numerical_features.columns)
num_cols = 3
num_rows = (num_features + num_cols - 1) // num_cols

plt.figure(figsize=(15, 5*num_rows))
for i, col in enumerate(numerical_features.columns):
    plt.subplot(num_rows, num_cols, i + 1)
    sns.histplot(super_data[col], kde=True)
    plt.title(col)
plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 5*num_rows))
for i, col in enumerate(numerical_features.columns):
    plt.subplot(num_rows, num_cols, i + 1)
    sns.boxplot(y=super_data[col])
    plt.title(col)
plt.tight_layout()
plt.show()

print("Descriptive Statistics:")
print(numerical_features.describe())
```

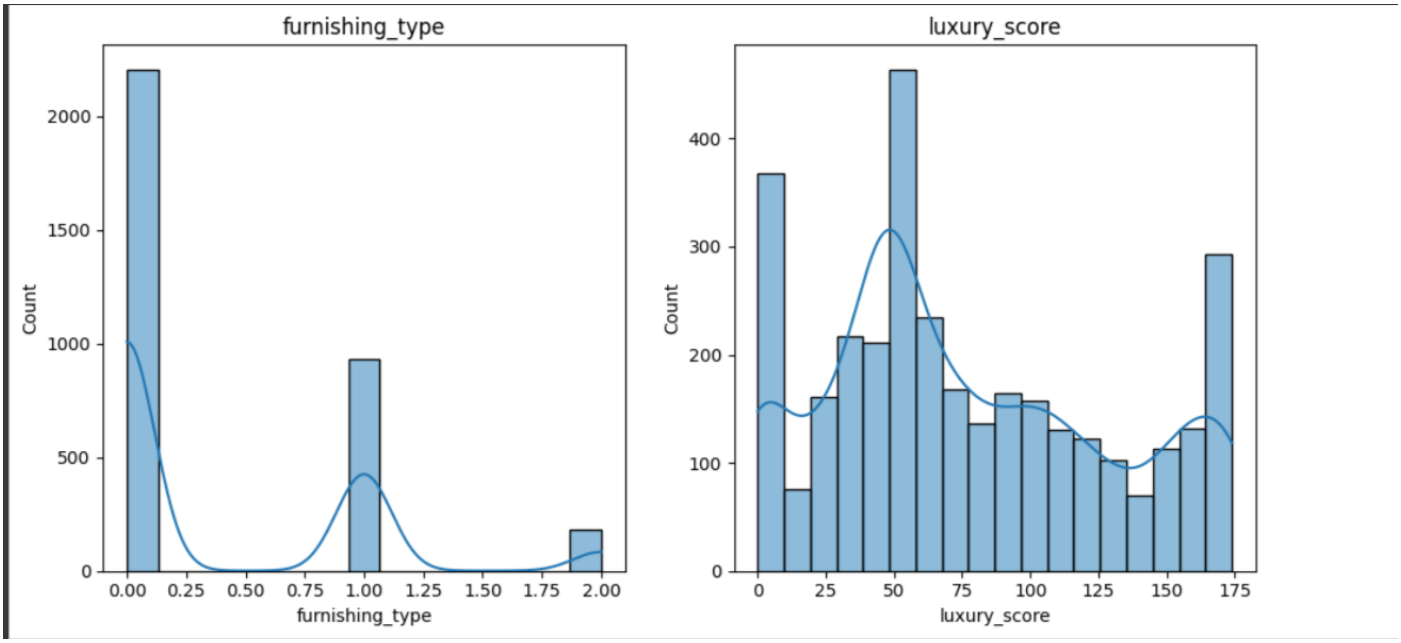
Here are the histograms plotted for the outliers of all the features


price
Count
price
price_per_sqft
Count
price_per_sqft
area
Count
area

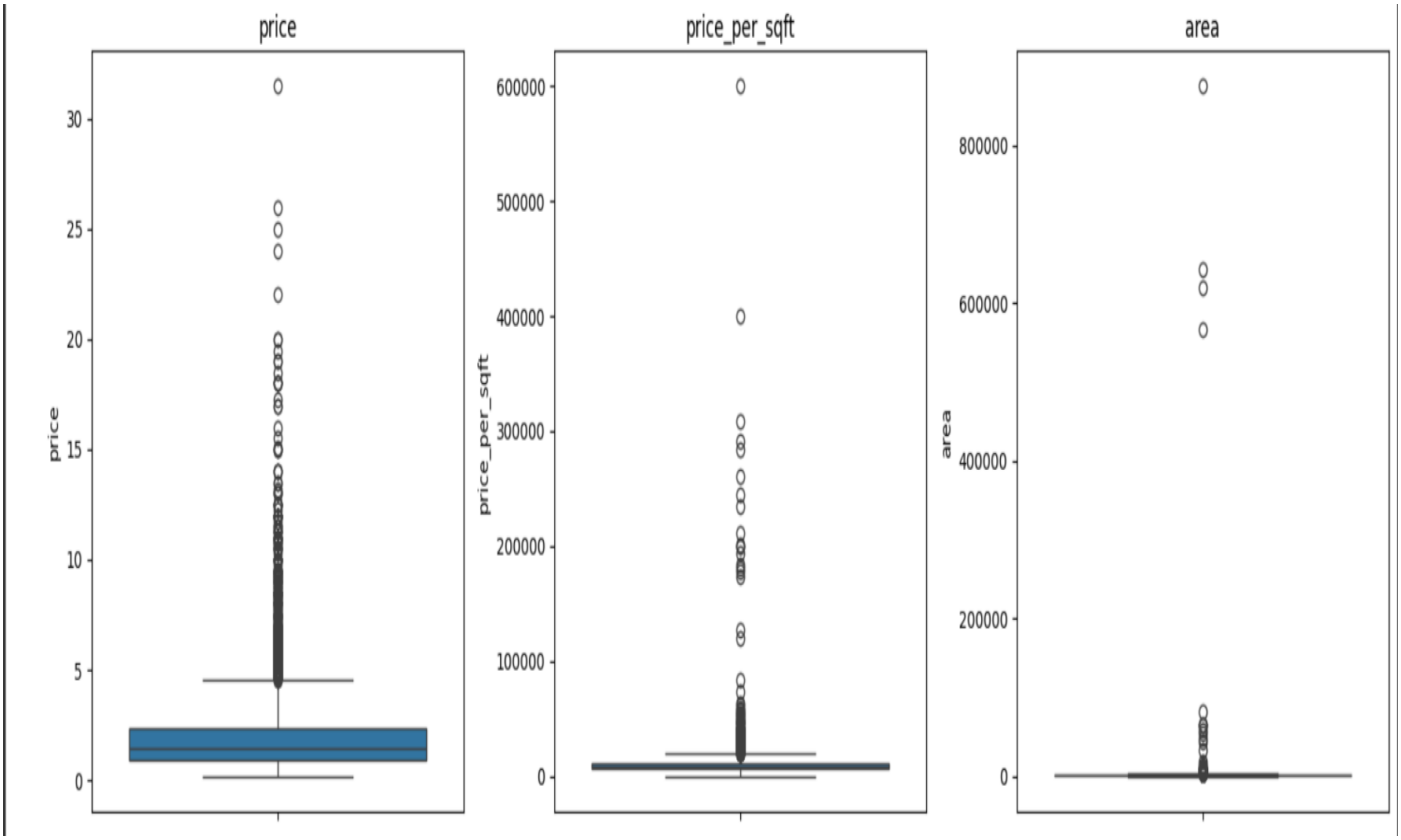

bedRoom
Count
bedRoom
bathroom
Count
bathroom
balcony
Count
balcony

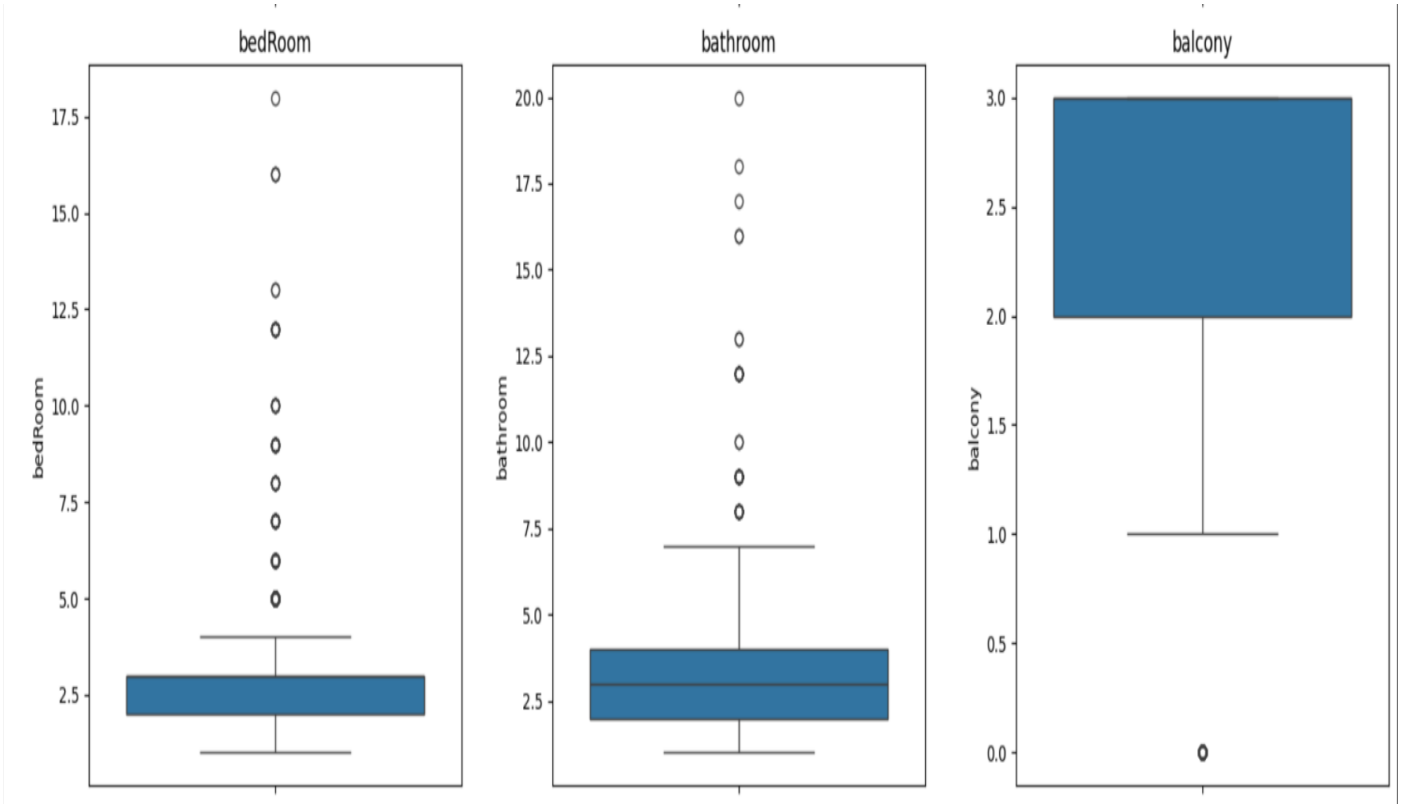

floorNum
Count
floorNum
super_built_up_area
Count
super_built_up_area
built_up_area
Count
built_up_area

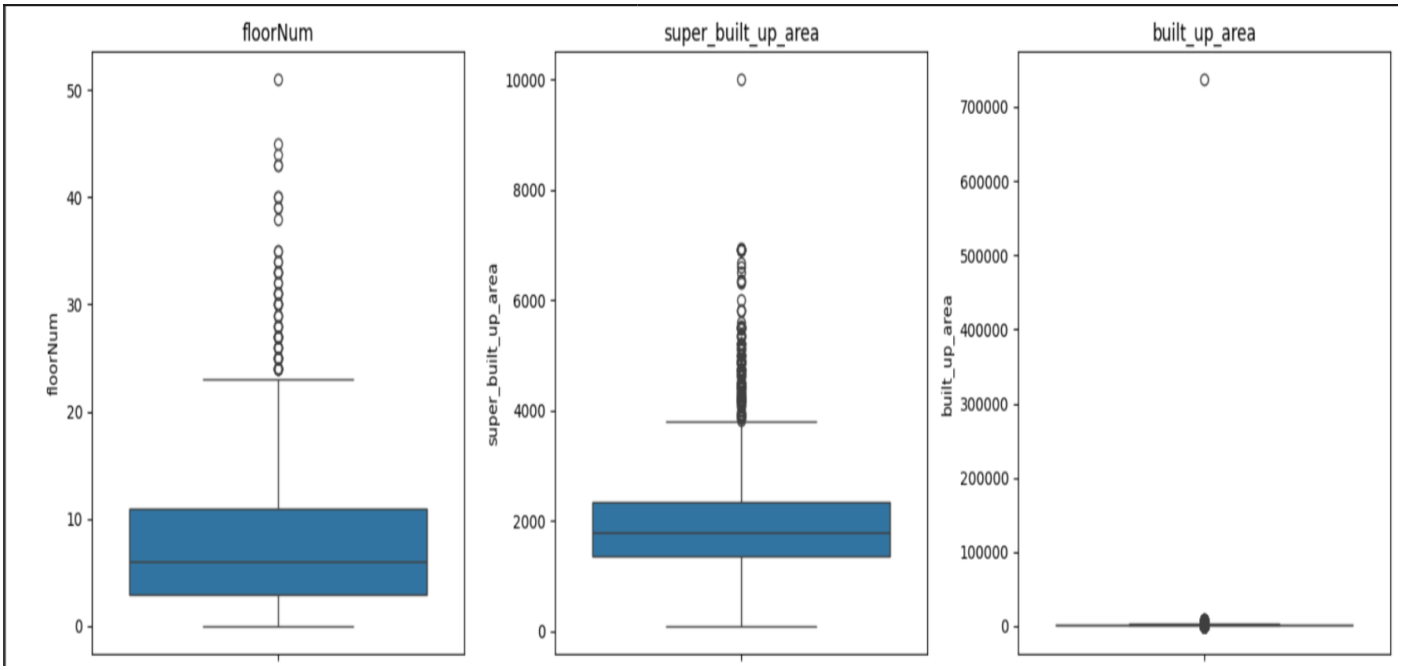

carpet_area
Count
carpet_area
study room
Count
study room
servant room
Count
servant room

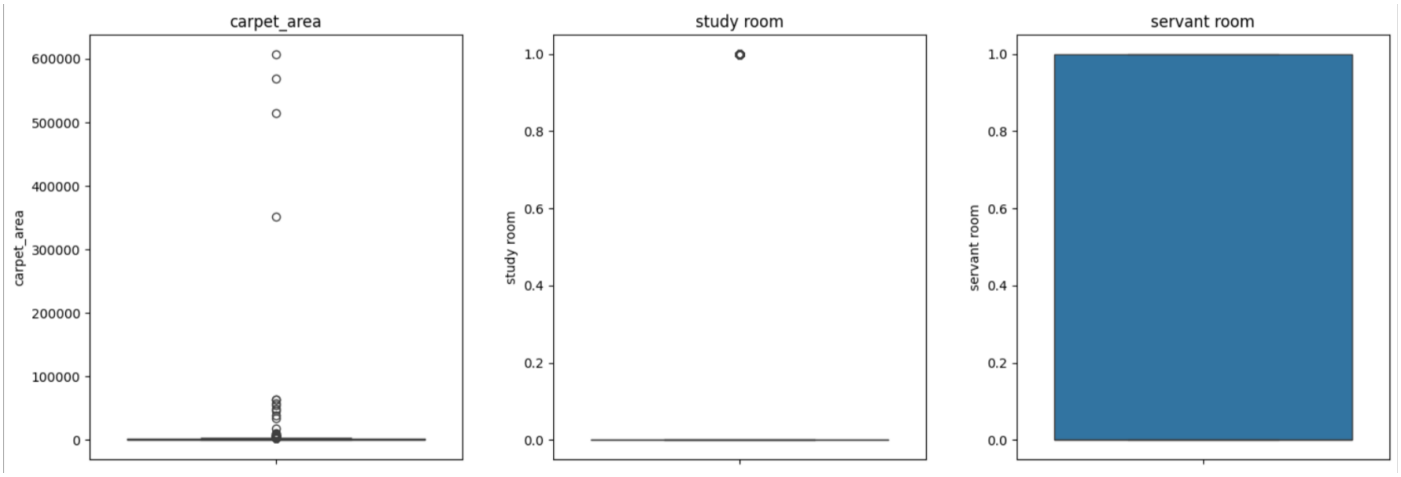


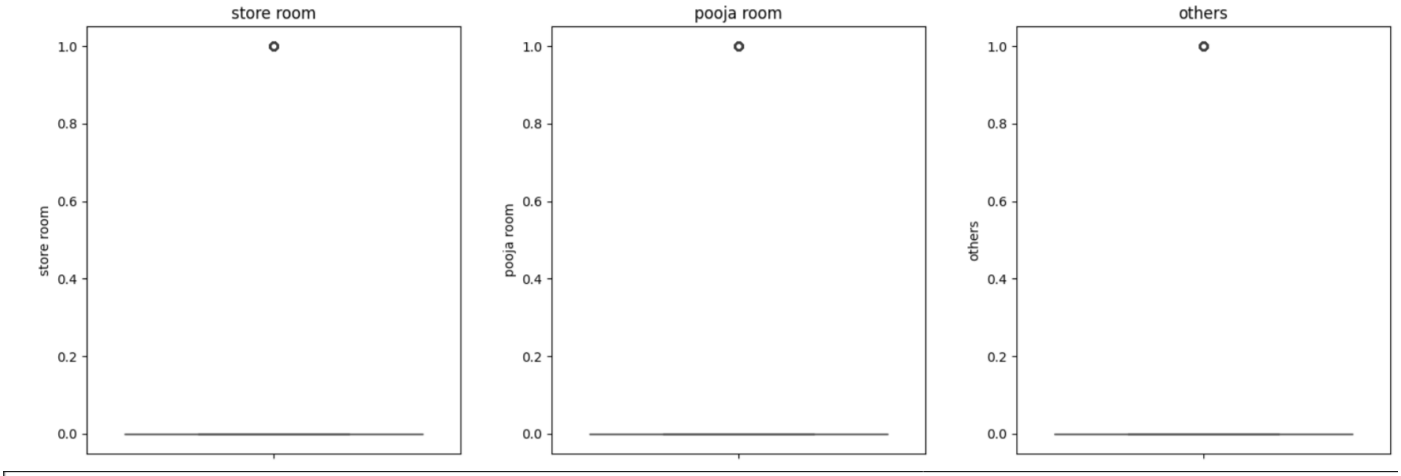
Here are the Boxplot graphs plotted for the outliers of all the features

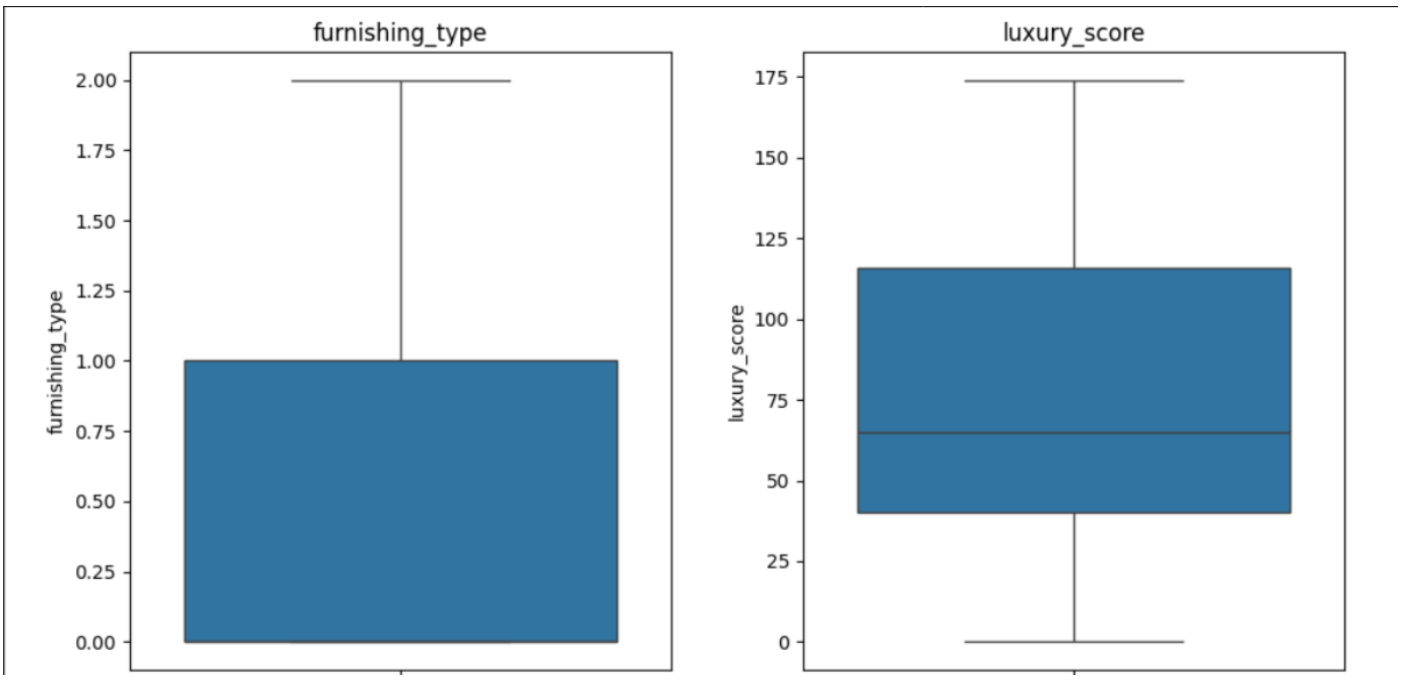



bedRoom
bedRoom
17.5
15.0
12.5
10.0
7.5
5.0
2.5
bathroom
bathroom
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
balcony
balcony
3.0
2.5
2.0
1.5
1.0
0.5
0.0


floorNum
floorNum
50
40
30
20
10
0
super_built_up_area
super_built_up_area
10000
8000
6000
4000
2000
0
built_up_area
built_up_area
700000
600000
500000
400000
300000
200000
100000
0







DESCRIPTIVE STATISTICS:

Descriptive Statistics:

	price	price_per_sqft	area	bedRoom	bathroom
count	3317.000000	3317.00000	3317.000000	3317.000000	3317.000000
mean	2.226809	11988.52035	2846.816521	3.045825	3.186011
std	2.549991	20467.27372	23971.350639	1.300811	1.464109
min	0.160000	4.00000	50.000000	1.000000	1.000000
25%	0.920000	6666.00000	1260.000000	2.000000	2.000000
50%	1.450000	8622.00000	1727.000000	3.000000	3.000000
75%	2.360000	12236.80000	2250.000000	3.000000	4.000000
max	31.500000	600000.00000	875000.000000	18.000000	20.000000

	balcony	floorNum	super_built_up_area	built_up_area
count	3317.000000	3317.000000	3317.000000	3317.000000
mean	2.464576	7.432379	1920.233836	2108.124657
std	0.801666	6.177046	950.108240	12802.870121
min	0.000000	0.000000	89.000000	30.000000
25%	2.000000	3.000000	1366.000000	1600.000000
50%	3.000000	6.000000	1790.000000	1808.600000
75%	3.000000	11.000000	2344.000000	2107.400000
max	3.000000	51.000000	10000.000000	737147.000000

	carpet_area	study room	servant room	store room	pooja room
count	3317.000000	3317.000000	3317.000000	3317.000000	3317.000000
mean	2188.366287	0.186011	0.362979	0.077480	0.155261
std	18189.777046	0.389175	0.480931	0.267391	0.362208
min	15.000000	0.000000	0.000000	0.000000	0.000000
25%	998.000000	0.000000	0.000000	0.000000	0.000000
50%	1300.000000	0.000000	0.000000	0.000000	0.000000
75%	1760.000000	0.000000	1.000000	0.000000	0.000000
max	607936.000000	1.000000	1.000000	1.000000	1.000000

	others	furnishing_type	luxury_score
count	3317.000000	3317.000000	3317.000000
mean	0.108532	0.390112	76.780826
std	0.311098	0.589718	52.212102
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	40.000000
50%	0.000000	0.000000	65.000000
75%	0.000000	1.000000	116.000000
max	1.000000	2.000000	174.000000

TASK-4

To handle outliers in the dataset, the method uses winsorization. Outliers are first identified using box plots and summary statistics. Winsorization lessens the impact of extreme numbers by substituting the 5th and 95th percentiles. Box plots, statistics, and post-Winsorization are used to show an improved outlier distribution. By reducing variance among numerical features, the approach improves the quality of the data and guarantees more dependable analyses or modelling results.

```
[ ]  from scipy.stats import zscore
     def handle_outliers(super_data):
         print("Initial Dataset Information:")
         print(super_data.describe(include='all'))

         plt.figure(figsize=(12, 6))
         sns.boxplot(data=super_data.select_dtypes(include=[np.number]))
         plt.title("Initial Outliers in the Dataset")
         plt.xticks(rotation=45, ha='right')
         plt.show()

         super_data_winsorized = super_data.copy()
         for col in super_data.select_dtypes(include=[np.number]).columns:
             lower_bound = super_data[col].quantile(0.05)
             upper_bound = super_data[col].quantile(0.95)
             super_data_winsorized[col] = np.where(super_data[col] < lower_bound, lower_bound, super_data[col])
             super_data_winsorized[col] = np.where(super_data[col] > upper_bound, upper_bound, super_data[col])

         plt.figure(figsize=(12, 6))
         sns.boxplot(data=super_data_winsorized.select_dtypes(include=[np.number]))
         plt.title("After Winsorization")
         plt.xticks(rotation=45, ha='right')
         plt.show()


         print("\nDataset Information After Winsorization:")
         print(super_data_winsorized.describe(include='all'))


         print("\nEffectiveness of Outlier Handling:")
         print("\nVariance Reduction:")
         print("Original Variance:\n", super_data.select_dtypes(include=[np.number]).var())
         print("Winsorized Variance:\n", super_data_winsorized.select_dtypes(include=[np.number]).var())
```

The function utilizes both Winsorization and Robust Scaling techniques to address outliers in the dataset. Initial analysis reveals outliers through box plots and summary statistics. Winsorization replaces extreme values with the 5th and 95th percentiles, while Robust Scaling standardizes the data, making it less sensitive to outliers. Comparisons of variance reduction demonstrate the effectiveness of each method. Based on variance reduction, the best outlier handling method is determined. This comprehensive approach ensures enhanced data quality for subsequent analyses or modeling tasks.

```python
from sklearn.preprocessing import RobustScaler
from scipy.stats import zscore

def handle_outliers(super_data):
    print("Initial Dataset Information:")
    print(super_data.describe(include='all'))

    plt.figure(figsize=(12, 6))
    sns.boxplot(data=super_data.select_dtypes(include=[np.number]))
    plt.title("Initial Outliers in the Dataset")
    plt.xticks(rotation=45, ha='right')
    plt.show()

    super_data_winsorized = super_data.copy()
    for col in super_data.select_dtypes(include=[np.number]).columns:
        lower_bound = super_data[col].quantile(0.05)
        upper_bound = super_data[col].quantile(0.95)
        super_data_winsorized[col] = np.where(super_data[col] < lower_bound, lower_bound, super_data[col])
        super_data_winsorized[col] = np.where(super_data[col] > upper_bound, upper_bound, super_data[col])

    plt.figure(figsize=(12, 6))
    sns.boxplot(data=super_data_winsorized.select_dtypes(include=[np.number]))
    plt.title("After Winsorization")
    plt.xticks(rotation=45, ha='right')
    plt.show()

    robust_scaler = RobustScaler()
    super_data_robust_scaled = pd.DataFrame(robust_scaler.fit_transform(super_data.select_dtypes(include=[np.number])), columns=super_data.select_dtypes(include=[np.number]).columns)

    plt.figure(figsize=(12, 6))
    sns.boxplot(data=super_data_robust_scaled)
    plt.title("After Robust Scaling")
    plt.xticks(rotation=45, ha='right')
    plt.show()

    print("\nDataset Information After Winsorization:")
    print(super_data_winsorized.describe(include='all'))

    print("\nDataset Information After Robust Scaling:")
    print(super_data_robust_scaled.describe(include='all'))

    print("\nEffectiveness of Outlier Handling:")
    original_variance = super_data.select_dtypes(include=[np.number]).var()
    winsorized_variance = super_data_winsorized.select_dtypes(include=[np.number]).var()
    robust_scaled_variance = super_data_robust_scaled.var()

    print("\nVariance Reduction:")
    print("Original Variance:\n", original_variance)
    print("Winsorized Variance:\n", winsorized_variance)
    print("Robust Scaled Variance:\n", robust_scaled_variance)

    variance_reduction = {
        "Winsorization": original_variance - winsorized_variance,
        "Robust Scaling": original_variance - robust_scaled_variance
    }

    best_method = max(variance_reduction, key=lambda k: variance_reduction[k].sum())
    print(f"\nThe best method for handling outliers based on variance reduction is: {best_method}")

handle_outliers(super_data)
```

```
Initial Dataset Information:
       property_type       society      sector        price  price_per_sqft  \
count           3317          3317        3317  3317.000000      3317.00000
unique             2           675         102          NaN             NaN
top             flat  tulip violet  sohna road          NaN             NaN
freq            2943            76         166          NaN             NaN
mean             NaN           NaN         NaN     2.226809     11988.52035
std              NaN           NaN         NaN     2.549991     20467.27372
min              NaN           NaN         NaN     0.160000         4.00000
25%              NaN           NaN         NaN     0.920000      6666.00000
50%              NaN           NaN         NaN     1.450000      8622.00000
75%              NaN           NaN         NaN     2.360000     12236.80000
max              NaN           NaN         NaN    31.500000    600000.00000
```

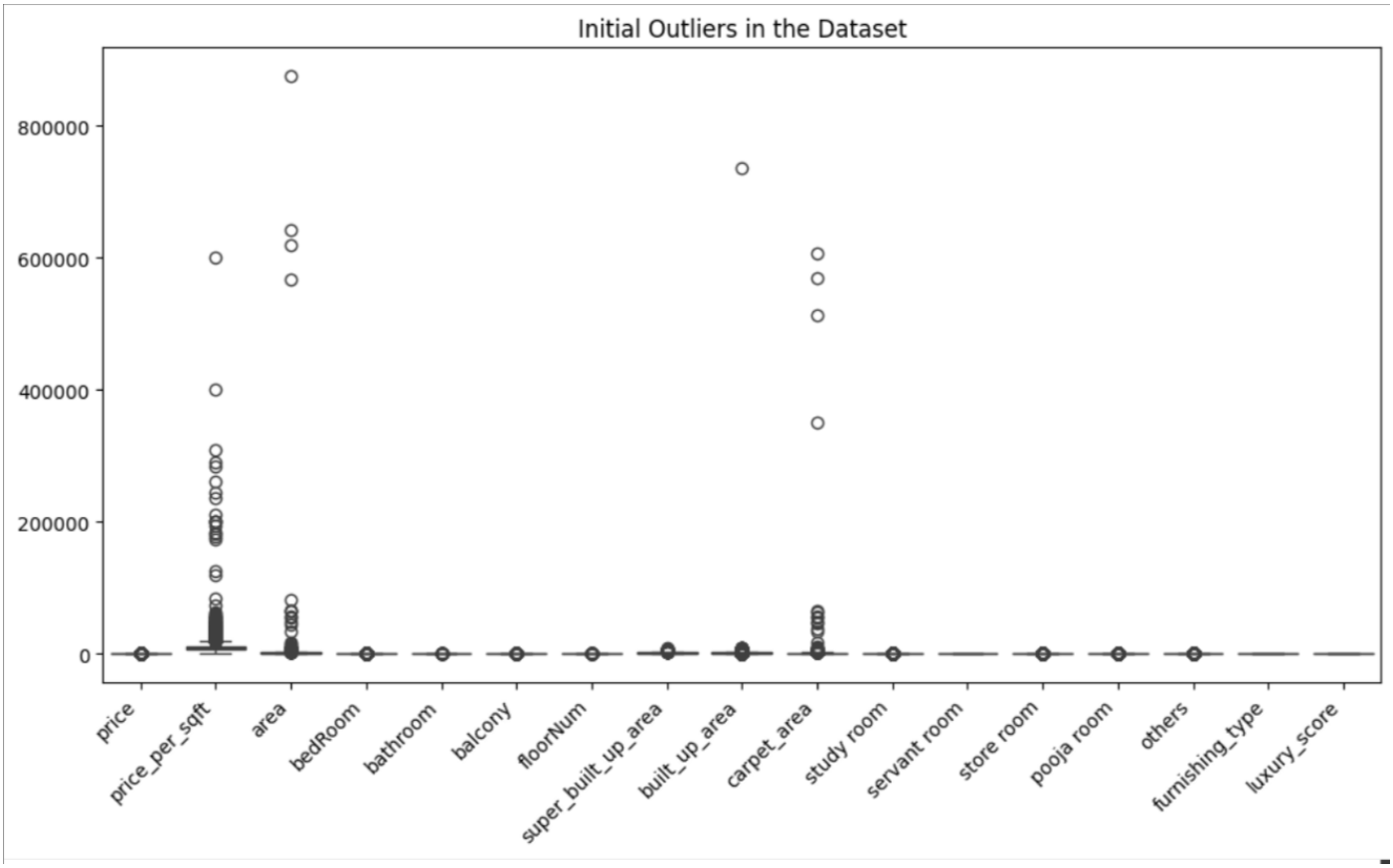
	area	areaWithType	bedRoom	bathroom	\
count	3317.000000	3317	3317.000000	3317.000000	
unique	NaN	2121	NaN	NaN	
top	NaN	Plot area 360(301.01 sq.m.)	NaN	NaN	
freq	NaN	20	NaN	NaN	
mean	2846.816521	NaN	3.045825	3.186011	
std	23971.350639	NaN	1.300811	1.464109	
min	50.000000	NaN	1.000000	1.000000	
25%	1260.000000	NaN	2.000000	2.000000	
50%	1727.000000	NaN	3.000000	3.000000	
75%	2250.000000	NaN	3.000000	4.000000	
max	875000.000000	NaN	18.000000	20.000000	

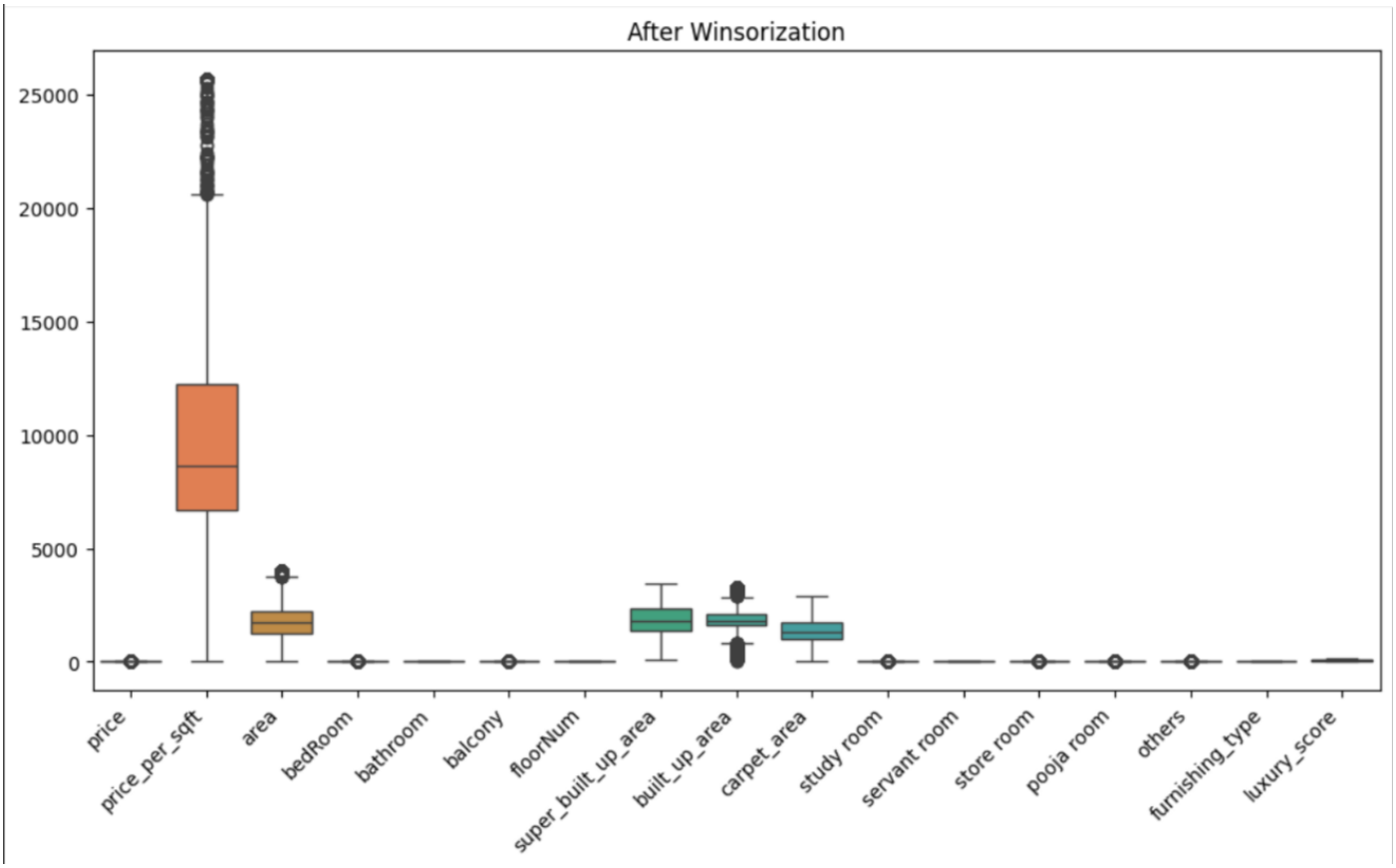
	balcony	...	super_built_up_area	built_up_area	carpet_area	\
count	3317.000000	...	3317.000000	3317.000000	3317.000000	
unique	NaN	...	NaN	NaN	NaN	
top	NaN	...	NaN	NaN	NaN	
freq	NaN	...	NaN	NaN	NaN	
mean	2.464576	...	1920.233836	2108.124657	2188.366287	
std	0.801666	...	950.108240	12802.870121	18189.777046	
min	0.000000	...	89.000000	30.000000	15.000000	
25%	2.000000	...	1366.000000	1600.000000	998.000000	
50%	3.000000	...	1790.000000	1808.600000	1300.000000	
75%	3.000000	...	2344.000000	2107.400000	1760.000000	
max	3.000000	...	10000.000000	737147.000000	607936.000000	

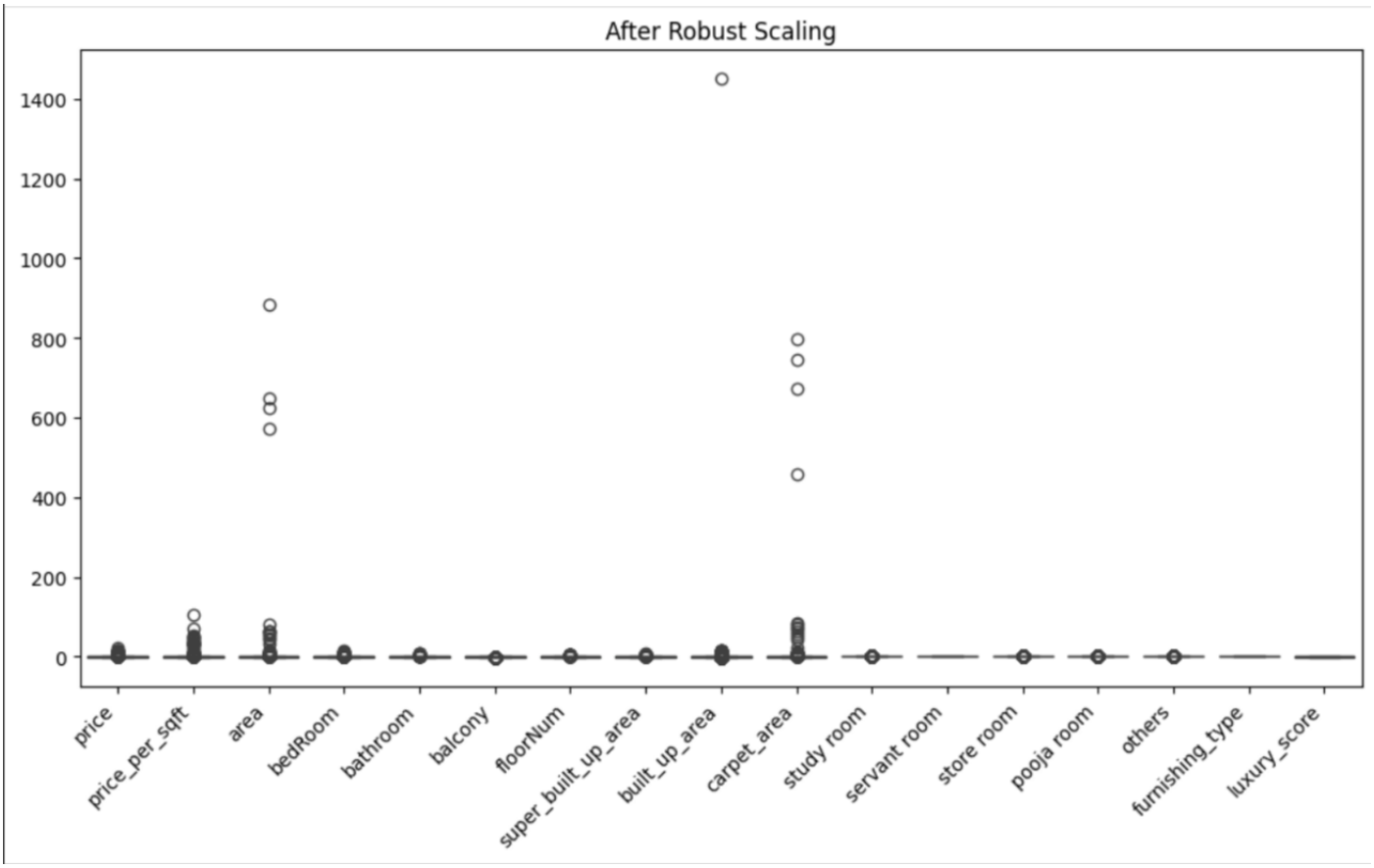
	study room	servant room	store room	pooja room	others	\
count	3317.000000	3317.000000	3317.000000	3317.000000	3317.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	0.186011	0.362979	0.077480	0.155261	0.108532	
std	0.389175	0.480931	0.267391	0.362208	0.311098	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	1.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	furnishing_type	luxury_score
count	3317.000000	3317.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	0.390112	76.780826
std	0.589718	52.212102
min	0.000000	0.000000
25%	0.000000	40.000000
50%	0.000000	65.000000
75%	1.000000	116.000000
max	2.000000	174.000000

Box plot for initial outliers in the datasets




After Winsorization
25000
20000
15000
10000
5000
0
price
price_per_sqft
area
bedRoom
bathroom
balcony
floorNum
super_built_up_area
built_up_area
carpet_area
study room
servant room
store room
pooja room
others
furnishing_type
luxury_score


After Robust Scaling
1400
1200
1000
800
600
400
200
0
price
price_per_sqft
area
bedRoom
bathroom
balcony
floorNum
super_built_up_area
built_up_area
carpet_area
study room
servant room
store room
pooja room
others
furnishing_type
luxury_score

Dataset Information After Winsorization:

	property_type	society	sector	price	price_per_sqft	\
count	3317	3317	3317	3317.000000	3317.000000	
unique	2	675	102	NaN	NaN	
top	flat	tulip violet	sohna road	NaN	NaN	
freq	2943	76	166	NaN	NaN	
mean	NaN	NaN	NaN	2.030842	10314.166657	
std	NaN	NaN	NaN	1.729208	5398.748183	
min	NaN	NaN	NaN	0.160000	4.000000	
25%	NaN	NaN	NaN	0.920000	6666.000000	
50%	NaN	NaN	NaN	1.450000	8622.000000	
75%	NaN	NaN	NaN	2.360000	12236.800000	
max	NaN	NaN	NaN	7.000000	25622.800000	

	area	areaWithType	bedRoom	bathroom	\
count	3317.000000	3317	3317.000000	3317.000000	
unique	NaN	2121	NaN	NaN	
top	NaN	Plot area 360(301.01 sq.m.)	NaN	NaN	
freq	NaN	20	NaN	NaN	
mean	1817.900332	NaN	2.957492	3.073259	
std	878.770925	NaN	0.916444	1.071227	
min	50.000000	NaN	1.000000	1.000000	
25%	1260.000000	NaN	2.000000	2.000000	
50%	1727.000000	NaN	3.000000	3.000000	
75%	2250.000000	NaN	3.000000	4.000000	
max	4000.000000	NaN	5.000000	5.000000	

	balcony	...	super_built_up_area	built_up_area	carpet_area	\
count	3317.000000	...	3317.000000	3317.000000	3317.000000	
unique	NaN	...	NaN	NaN	NaN	
top	NaN	...	NaN	NaN	NaN	
freq	NaN	...	NaN	NaN	NaN	
mean	2.464576	...	1854.056262	1812.736125	1406.095543	
std	0.801666	...	744.710368	715.286534	643.344580	
min	0.000000	...	89.000000	30.000000	15.000000	
25%	2.000000	...	1366.000000	1600.000000	998.000000	
50%	3.000000	...	1790.000000	1808.600000	1300.000000	
75%	3.000000	...	2344.000000	2107.400000	1760.000000	
max	3.000000	...	3434.000000	3240.000000	2900.000000	

	study room	servant room	store room	pooja room	others	\
count	3317.000000	3317.000000	3317.000000	3317.000000	3317.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	0.186011	0.362979	0.077480	0.155261	0.108532	
std	0.389175	0.480931	0.267391	0.362208	0.311098	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	1.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	furnishing_type	luxury_score
count	3317.000000	3317.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	0.390112	76.780826
std	0.589718	52.212102
min	0.000000	0.000000
25%	0.000000	40.000000
50%	0.000000	65.000000
75%	1.000000	116.000000
max	2.000000	174.000000

The initial summary statistics for the numeric columns are as follows:

Statistic	Price	Price_per_sqft	Area	BedRoom	Bathroom
Count	3317	3317	3317	3317	3317
Mean	2.23	11988.52	2846.82	3.05	3.19
Std Dev	2.55	20467.27	23971.35	1.30	1.46
Min	0.16	4.00	50.00	1.00	1.00
25th Percentile	0.92	6666.00	1260.00	2.00	2.00
Median (50%)	1.45	8622.00	1727.00	3.00	3.00
75th Percentile	2.36	12236.80	2250.00	3.00	4.00
Max	31.50	600000.00	875000.00	18.00	20.00

After Winsorization

The dataset statistics after Winsorization are:

Statistic	Price	Price_per_sqft	Area	BedRoom	Bathroom
Count	3317	3317	3317	3317	3317
Mean	2.03	10314.17	1817.90	2.96	3.07
Std Dev	1.73	5398.75	878.77	0.92	1.07
Min	0.16	4.00	50.00	1.00	1.00
25th Percentile	0.92	6666.00	1260.00	2.00	2.00
Median (50%)	1.45	8622.00	1727.00	3.00	3.00
75th Percentile	2.36	12236.80	2250.00	3.00	4.00
Max	7.00	25622.80	4000.00	5.00	5.00

After Robust Scaling

The dataset statistics after Robust Scaling are:

Statistic	Price	Price_per_sqft	Area	BedRoom	Bathroom
Count	3317	3317	3317	3317	3317
Mean	0.54	0.60	1.13	0.05	0.09
Std Dev	1.77	3.67	24.21	1.30	0.73
Min	-0.90	-1.55	-1.69	-2.00	-1.00
25th Percentile	-0.37	-0.35	-0.47	-1.00	-0.50
Median (50%)	0.00	0.00	0.00	0.00	0.00
75th Percentile	0.63	0.65	0.53	0.00	0.50
Max	20.87	106.16	882.09	15.00	8.50

Variance Comparison

Statistic	Original Variance	Winsorized Variance	Robust Scaled Variance
Price	6.502455	2.990159	3.135829
Price_per_sqft	4.189093e+08	2.914648e+07	13.498480
Area	5.746257e+08	7.722383e+05	586.292880
BedRoom	1.692109	0.839869	1.692109
Bathroom	2.143616	1.147527	0.535904
Balcony	0.642668	0.642668	0.642668
FloorNum	38.15590	28.32635	0.596186
Super_built_area	9.027057e+05	5.545935e+05	0.943775
Built_up_area	1.639135e+08	5.116348e+05	636.669073
Carpet_area	3.308680e+08	4.138922e+05	569.829343
Studyroom	0.151457	0.151457	0.151457
Servantroom	0.231078	0.231078	0.231078
Luxury_score	3.266716	0.974888	1.608646

Comparison and Analysis of Variance

Initial Difference

There were notable outliers in the original information, as evidenced by the high variances in a number of columns, including built-up area, area, price per square foot, and carpet area.

Variance Winsorized

Most columns showed a significant decrease in variance following the application of Winsorization, particularly the ones with the largest initial variances.

Important Notes

Original Price Deviation: 6.502455

2.990159 is the winsorized variance.

Diminishment: About 54%

Price Per Square Foot:

Variance at Origin: 4.189093e+08

Variance Winsorized: 2.914648e+07

Diminution: Roughly 93%

Region:

Variance at Origin: 5.746257e+08

Variance Winsorized: 7.722383e+05

Decrease: Greater than 99%

Area Constructed:

Variance at Original: 1.639135e+08

Variance Winsorized: 5.116348e+05

Decrease: Greater than 99%

Area Carpeted:

Variance at Origin: 3.308680e+08

Variance Winsorized: 4.138922e+05

Decrease: Greater than 99%

Interpretation:

The findings reveal a noteworthy decrease in variance subsequent to Winsorization, indicating that this method efficiently lessens the impact of extreme outliers in the majority of columns. Several important points are highlighted by the large reduction percentages, especially in columns with substantial variances initially:

Efficiency in Reducing Variance

Winsorization works incredibly well at minimizing volatility brought on by anomalies. The most significant reductions are displayed in the columns with the highest initial variances, demonstrating effective handling of extreme values.

Data structure preservation:

Winsorization ensures that the overall structure of the dataset is maintained by keeping all of the data points.Study_room, servant_room, and furnishing_type are

examples of columns that do not alter, indicating that either no extreme outliers were present or that the selected percentile had no effect on these values.

Conclusion:

The dataset has shown impressive results when the Winsorization technique is applied. Most notably, the variance across columns like price, price per square foot, area, built-up area, and carpet area has been significantly reduced. These decreases, which range from roughly 54% to more than 99%, demonstrate how effective the method is at reducing the impact of extreme outliers. Additionally, by keeping all of the data points in the dataset, winsorization guarantees that its structure is preserved. The technique-unaffected columns show that the selected percentile has little effect or that there are no extreme outliers. The dataset is now much more stable and resilient overall thanks to winsorization, which also makes it suitable for additional research and interpretation. But robust Scaling is more feasible than winsorization.