REPORT

TASK –1

1) **Importing libraries:** Initially, all the necessary libraries (Pandas, matplotlib, seaborn, NumPy) have been imported

2) **Loading and reading dataset:** Then the CSV file named “Gurgaon\_RealEstate.csv” is read using ‘pd.read\_csv()’

3) **Printing the Data Frame:** The CSV file is read into a data frame ‘d’, then we print the data set.



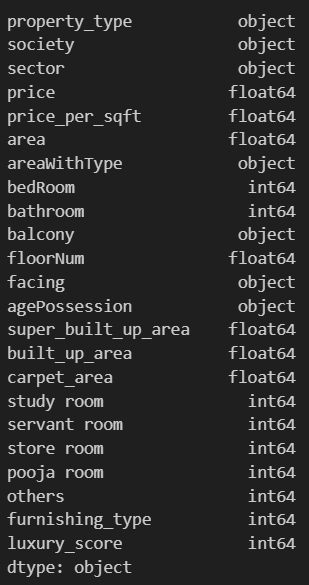
**Observation:** Here in this code, we don't perform any specific data manipulation, analysis, or visualization beyond reading and printing the Data Frame.

**Printing shape of data frame**: we print the shape of the data frame using shape ()



**Observation:** Through this we get an idea of how much data it contains and the structure of dataset

**Printing the datatypes:** We are going to print the datatypes of all the columns(features)



**Observation:** Based on the datatypes observed, we can further plan the data preprocessing like handling missing values, encoding, etc.

**Summary of the dataset:** Now to get the summary of the data frame using d.info () to get information like non-null entries, datatypes, etc.

**Observations:** We observe that we get an output of

* Number of non-null entries for each column
* Total number of rows in the data frame
* Datatypes of each column(feature)
* Memory usage by the data frame.



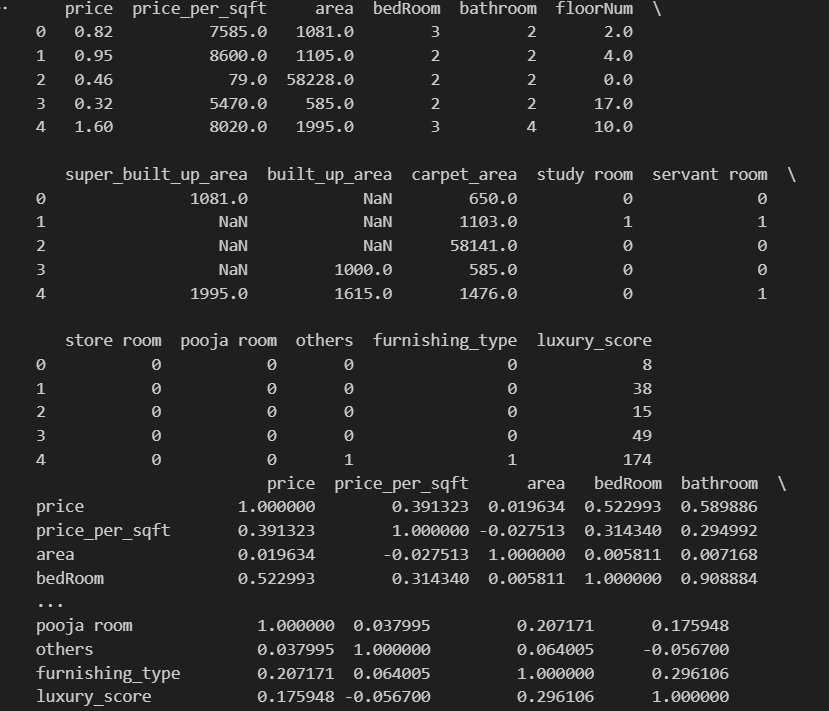
**Checking duplicate rows:** We use the. duplicated() function to indicate whether a row is a duplicate of the previous row or not. It uses ‘True’ or ‘False’ to indicate for each row. Identifying duplicate rows is crucial for data cleaning and ensuring data integrity.



**Observation:** We observe that:

* All the rows with the value ‘True’ indicate that they are duplicates
* All the rows with values ‘False’ indicate that they are not duplicates

**Removing the duplicate rows**: We now remove the duplicate rows using .drop\_duplicates() to reduce the data set size and make it easier to perform other operations.



**Observation:** Through this output, we observe that:

* The size of the dataset is reduced after removing the duplicate rows and they contain unique rows. This operation is useful for data cleaning and ensuring data integrity.
* The number of rows are reduced from 3803 to 3677

1. **Select numerical columns and verify to print:** Now we select only the numerical columns from the data set and we print those numerical columns by inspecting if they are numerical or not.
2. **Calculate and print the correlation matrix:** Then we calculate the correlation matrix for the numerical data selected above and print the matrix**.**

**Observation:** In (1) we observe that:

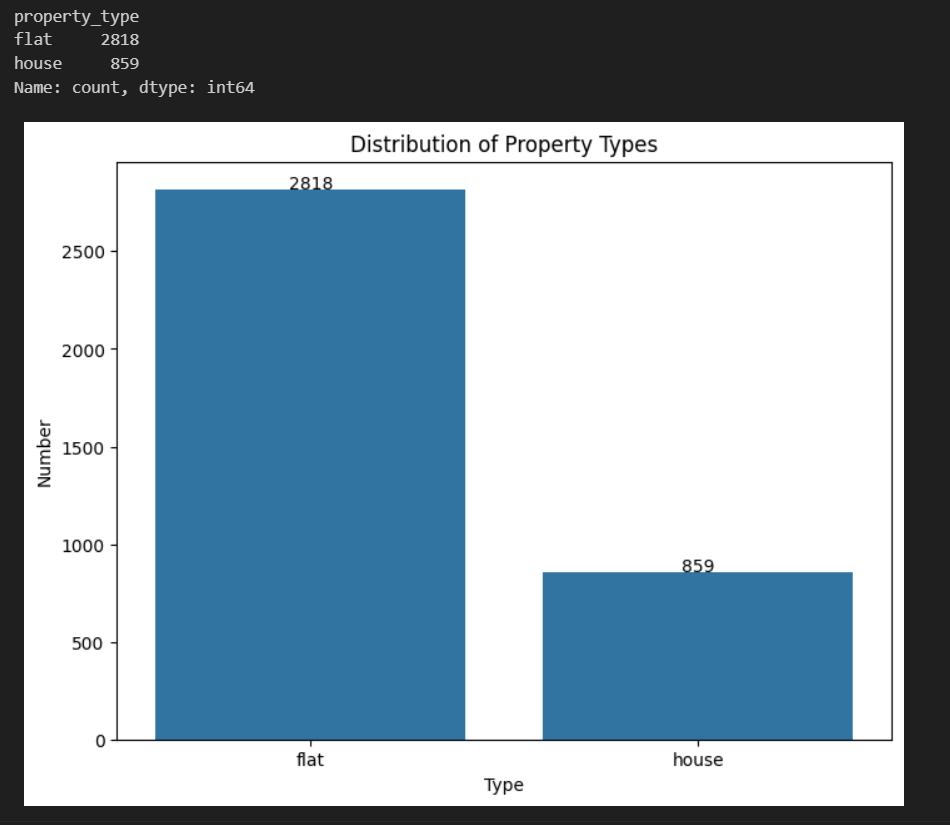
* The resulting dataset will contain only numerical columns like float and int
* We print the first few columns of the selected columns above.
* By checking the output we can verify whether selected columns are numerical or not or if any columns are excluded.

In (2) we observe that:

* On printing the calculated correlation matrix, we get the insights above the relationships between numerical variables.
* High absolute correlation factors indicate (close to 1 or -1) a strong linear relationship, while low values (close to 0) indicate a weak or no relationship.
* Through a matrix, we can identify the pairs of numerical variables which are strongly correlated.

**1.**  **Counting and printing property types**: We are now counting the occurrences of each unique property type in the property\_type column(feature) and then we print the count of each property type showing the distribution in property types across the dataset.

**2. Plotting bar plot for property\_type feature**: We print the bar plot using Seaborn, showing the distribution of property types.



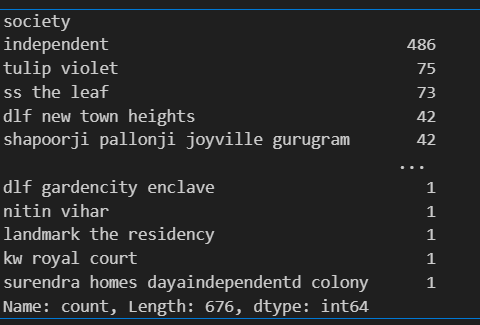
**Observations:** In (1) we observe that:

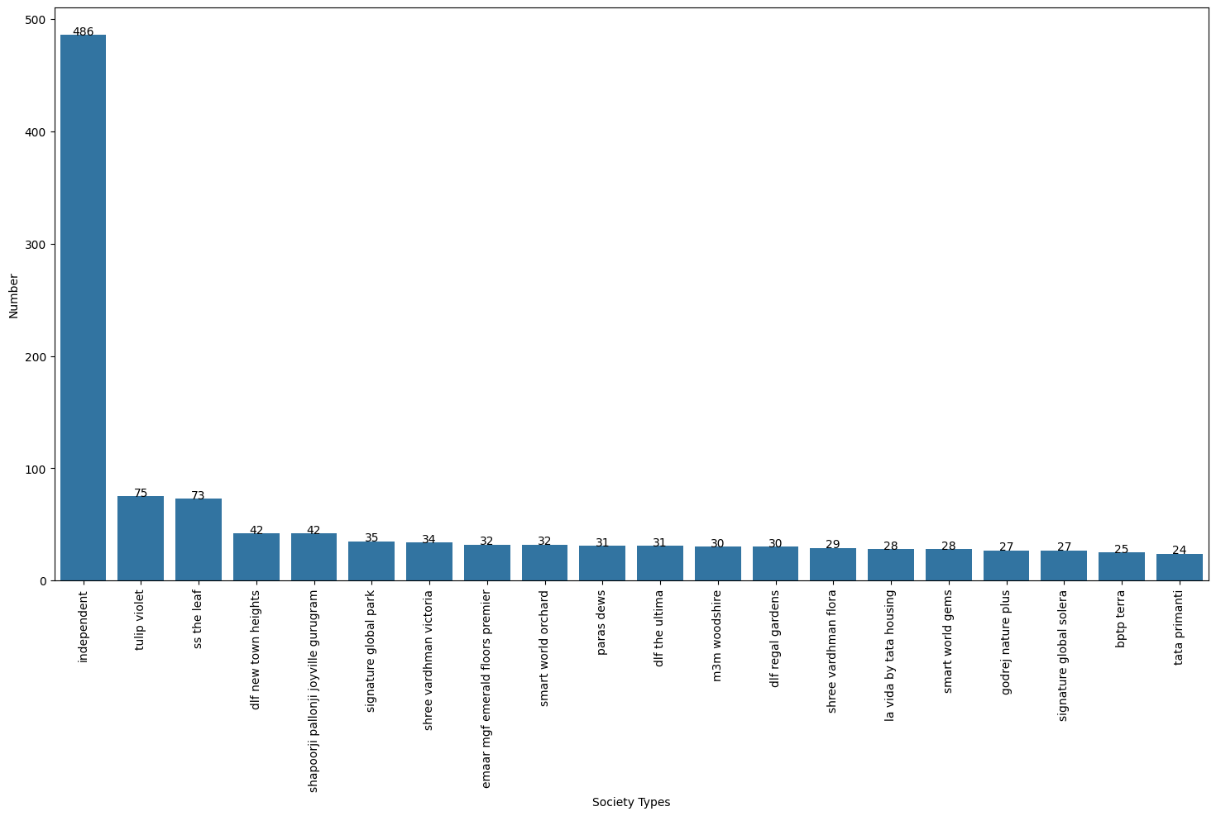
* It indicates the frequency of each property type
* Helps us check if the property type is common or rare.

In (2) we observe that: The bar plot visually represents the distribution of the property types, and we can observe which property types are more prevalent.

**1.Counting and printing the society types**: So now we are going to count the occurrences of each unique society in the society feature and then we print the count of each society type.

**2.**  **Selecting top 20 societies and plotting them**: Now we will select the top 20 societies based on their occurrence and we will plot these top 20 societies using a bar plot.





**Observations**: In (1) we observe that:

* The output shows the occurrence of each society type
* This will help us understand how many properties (property\_types feature) are associated with each society

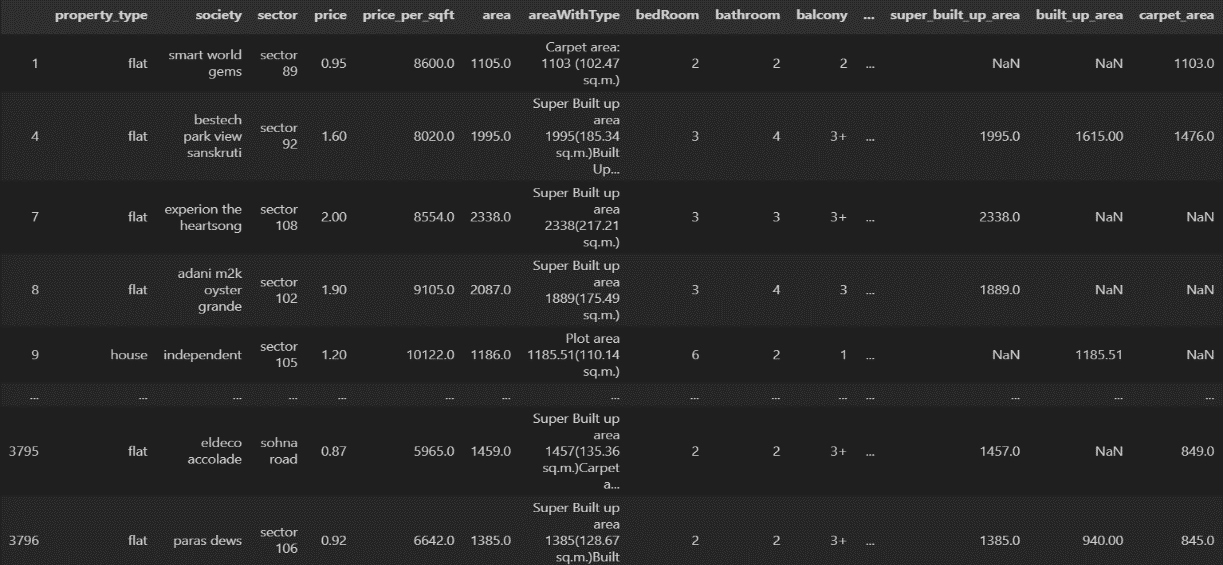
In (2) we observe that:

* The bar plot visually represents the distribution of the top 20 societies, so that it is easier to understand their occurrence
* It helps identify the most prominent societies based on the number of properties associated.

**1.**We are now basically trying to reduce the cardinality by removing the societies having a smaller number of associated properties

**2.Setting threshold and filtering societies**: So, we are setting a threshold of 10 that is all societies with less than 10 flats/houses are removed and the societies column is filtered

**3.Filtering the data frame**: We are now filtering d and storing in d1 such that d1 will only have valid societies after filtering and we print it after removing unnecessary societies.



**Observations:**

In (2) we observe that:

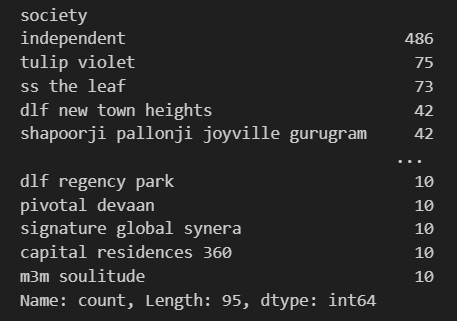
* We do not get any output but all the societies with fewer than 10 occurrences are removed reducing the cardinality of the society column
* On reducing cardinality, the society column becomes more manageable

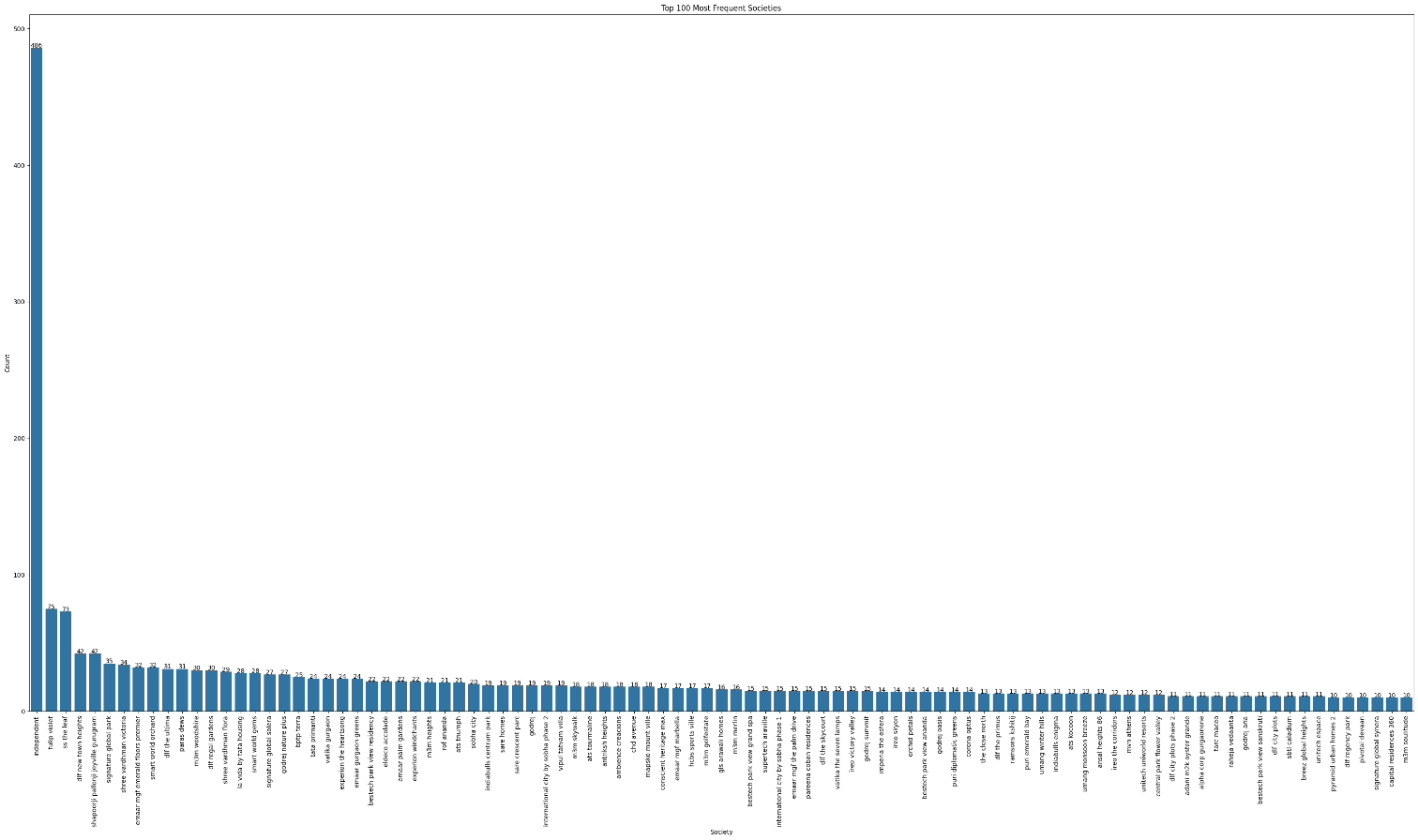
In (3) we observe that:

* The d1 data frame contains a subset of original data focusing only on societies with sufficient occurrences.

**1.Counting the societies after removing**: We are now counting the societies only that have sufficient associated properties to check if the societies with lease occurrences are all excluded

**2.Plotting the societies after reducing cardinality**: Now we plot the society column after reducing cardinality



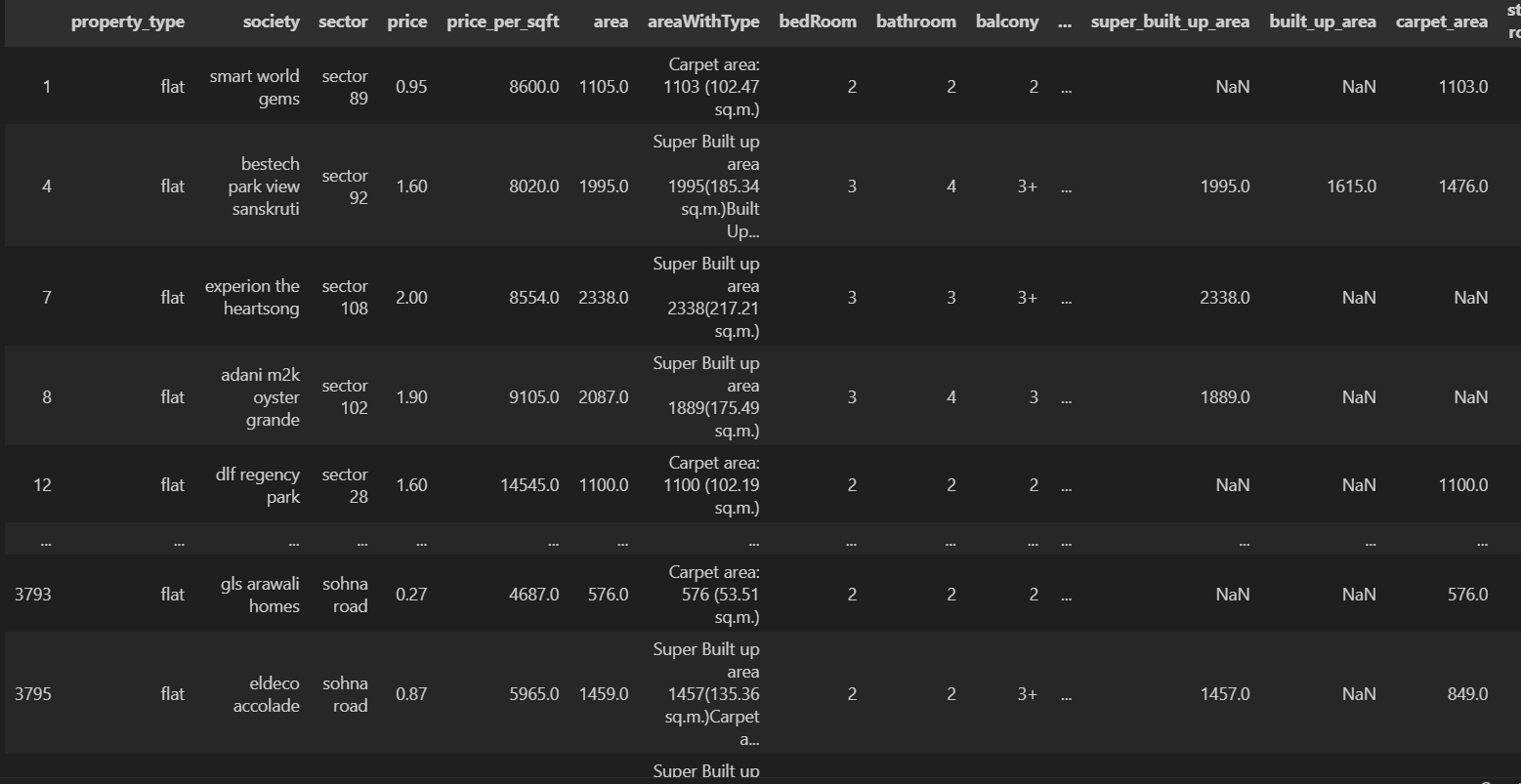
**Observations**: In (1) we observe that:

* We can observe that the count of all the societies present in the column have occurrences greater than 10 as required

In (2) we observe that:

* The bar plot helps us observe the frequency of the societies with sufficient occurrences after reducing cardinality

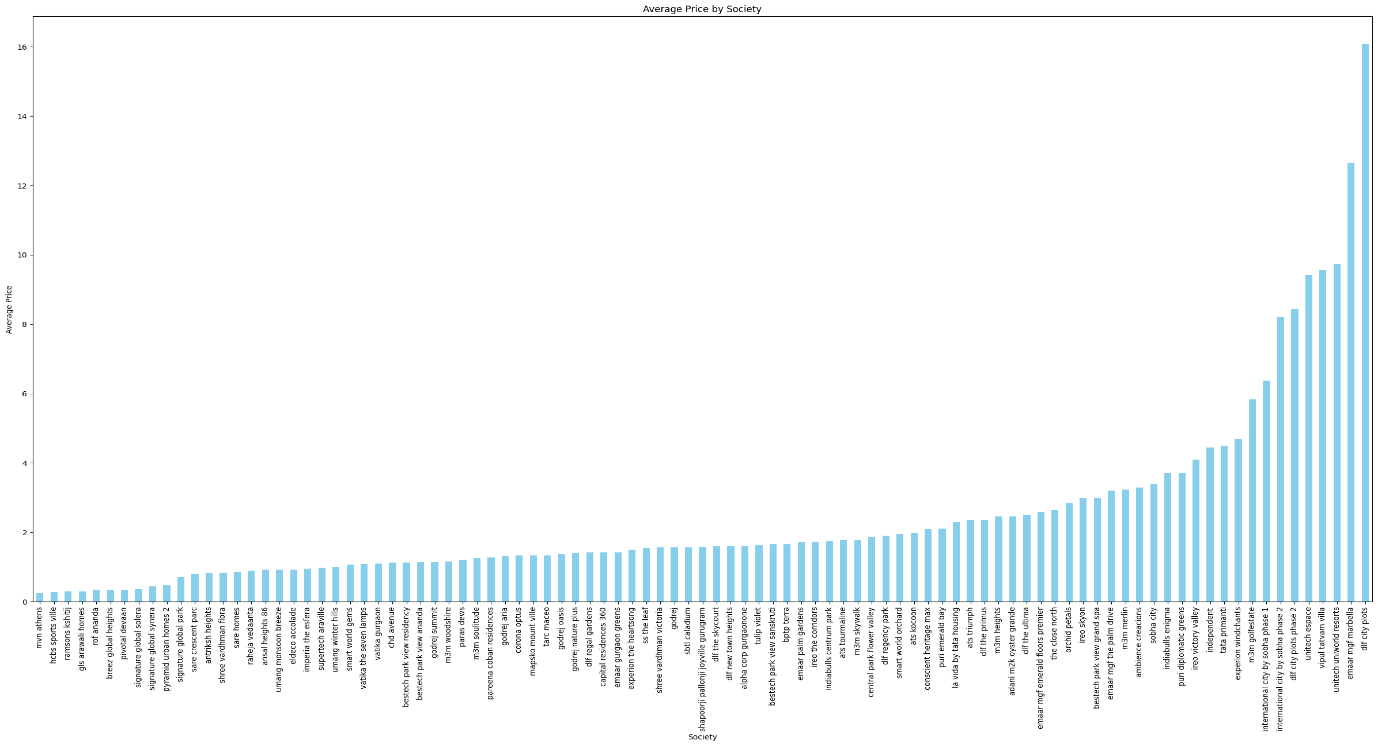
**Removing of ‘independent’ in the society column**: We are now removing all the rows having independent in the society column and then we print it.



**Observation**: Only the rows in the Data Frame d2 that result from the society column lacking the string 'independent' are included. The dataset is narrowed down to remove properties that aren't connected to any particular society by eliminating rows that have "independent" societies.

**1.Calculating the average price by society**: We are trying to average price of properties in each society. It groups the data by society, calculates the mean price for each group, and sorts the results based on the average price from lowest to highest.

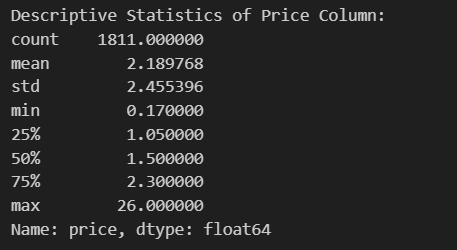
**2.Plotting the bar plot**: A bar plot is plotted between the average price and society

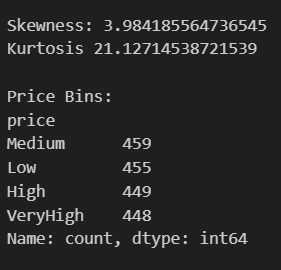


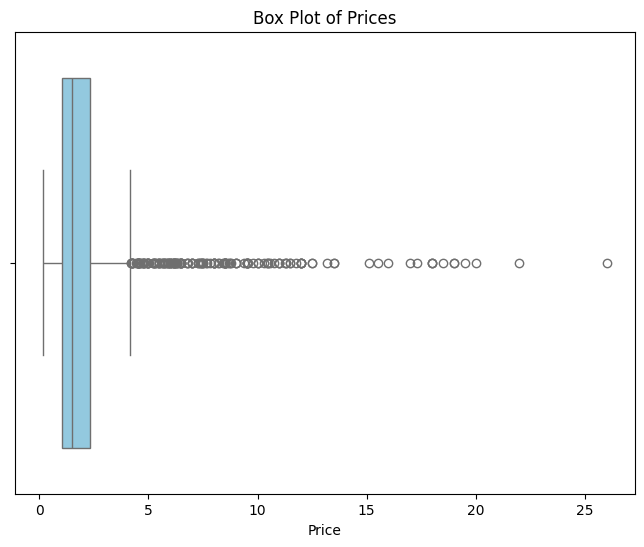
Observation: We observe through the bar plot we can identify the societies that command higher prices in real estate.

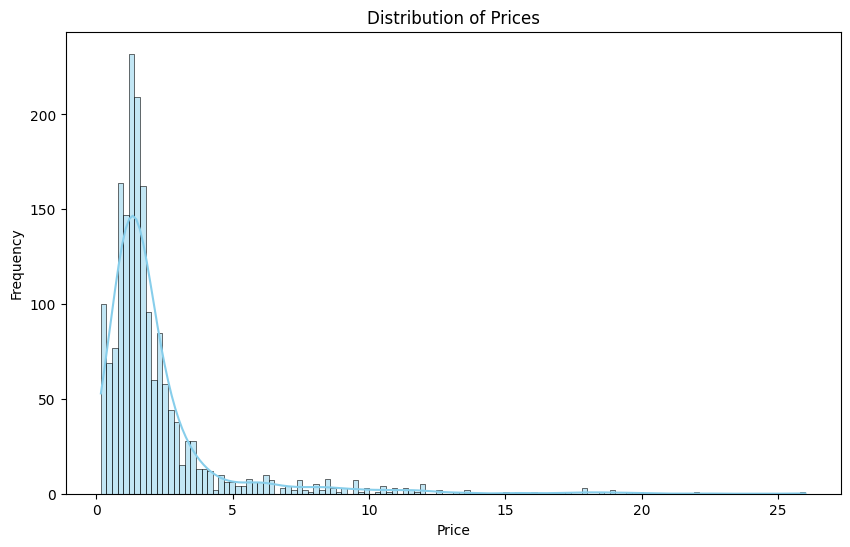
**1. Price column analysis:** We are now performing univariate analysis on the price column

* **Missing values:** First we are going to calculate the number of missing values in the price column and print the total number of missing values in this price column.We are going to use function .isnull(). sum ()
* **Descriptive statistics:**Now we are going to find the descriptive statistics of the price column like mean, standard deviation, etc.
* **Distribution plot:** We are going to plot a histogram to visualize the price distribution. Using kernel density estimation.
* **Box plot:** we are going to plot a box plot to visualize the distribution.
* **Skewness and kurtosis:** We are going to calculate the skewness and kurtosis of price column to measure the asymmetry and tall-heaviness,respectively
* **Price binning:**Now we are dividing the prices into four bins (Low, Medium, High, Very High) using quartile-based binning (using ‘pd.qcut()’) and print counts of each property in each price bin





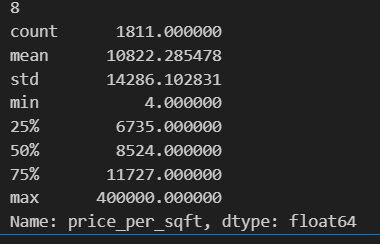


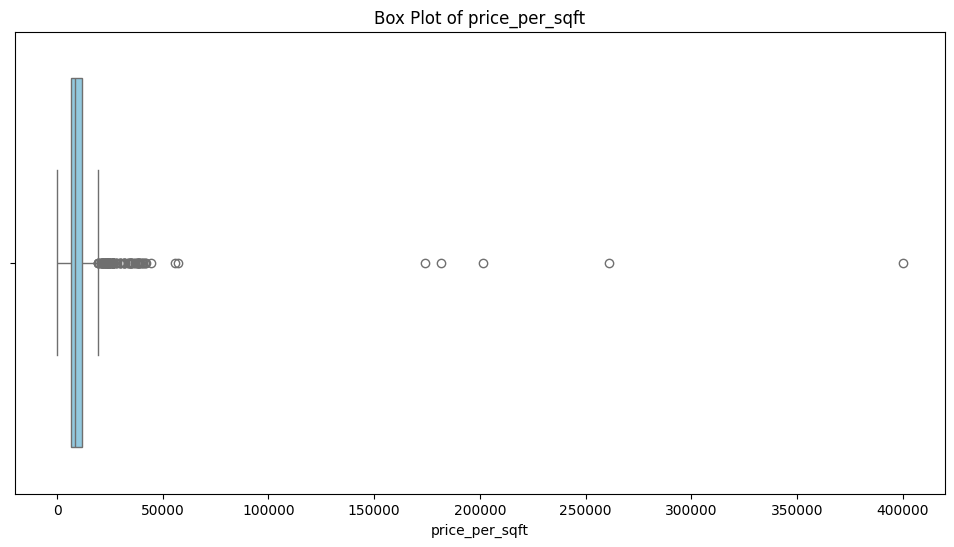


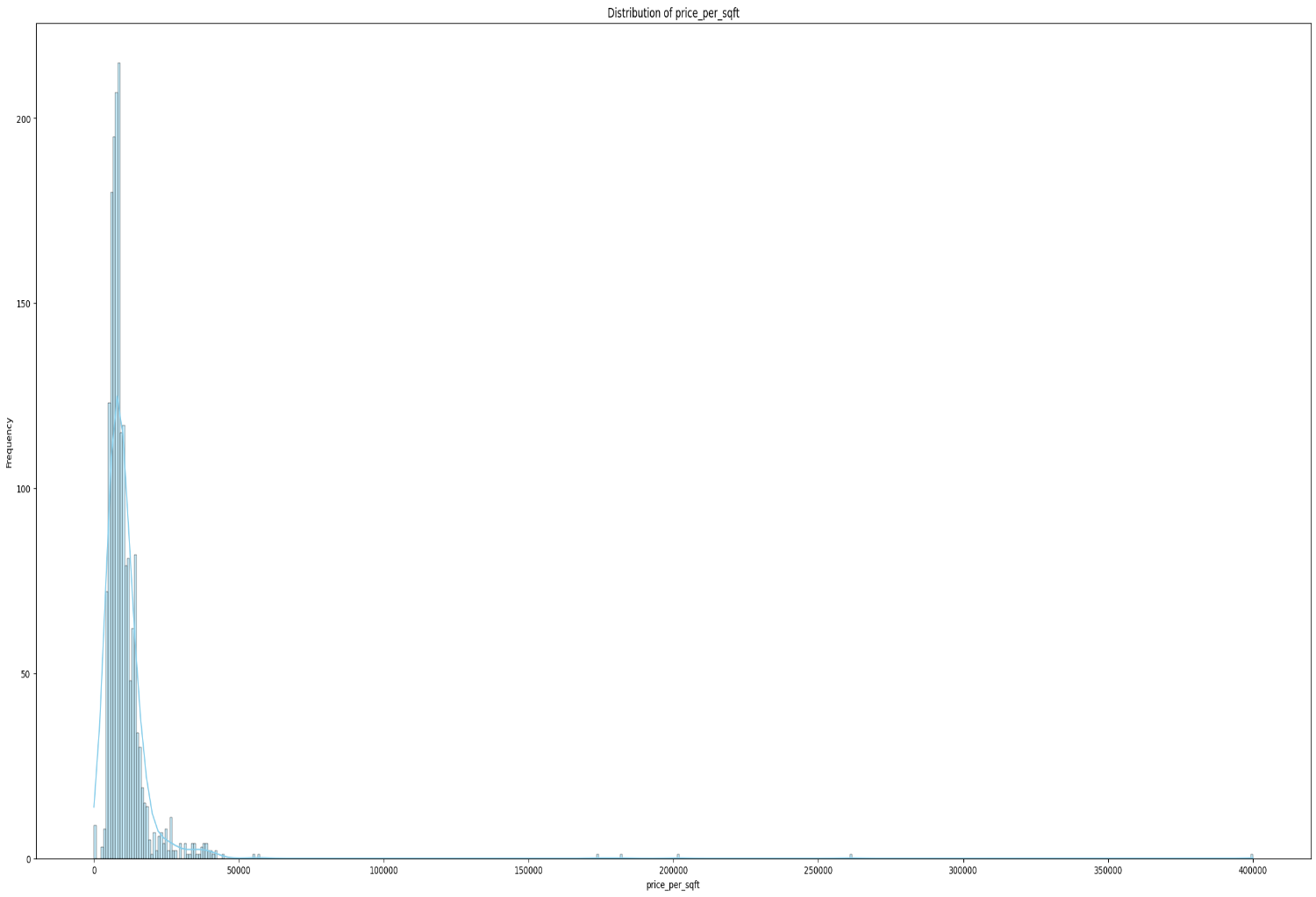
**Observation:** We observe that:

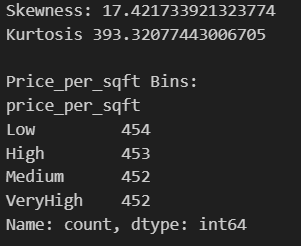
* The descriptive statistics and missing values give insights about the distribution of prices.
* Histograms and box plots help us understand the central tendency, spread and shape of the price distribution.
* Skewness and kurtosis represent the degree of asymmetry and tail-heaviness in the price distribution
* Price binning divides prices into categories based on quartiles, allowing us to have easier interpretation about price ranges.
* Overall helps us understand the distribution of price and also about the real estate market trends

Similarly, we perform univariate analysis as above for price\_per\_sqft column.





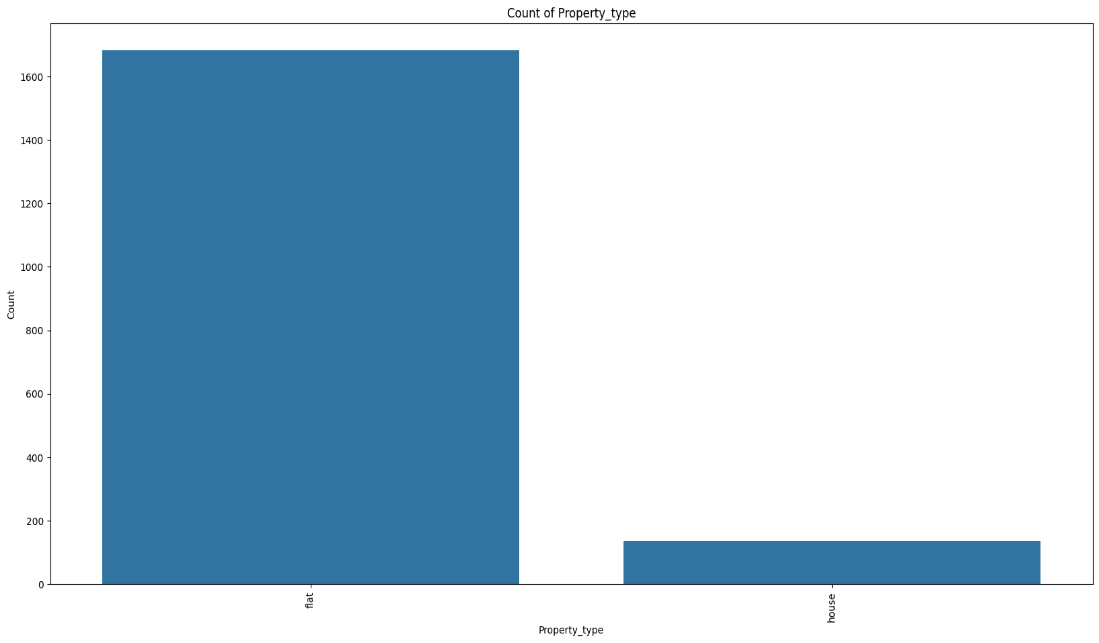


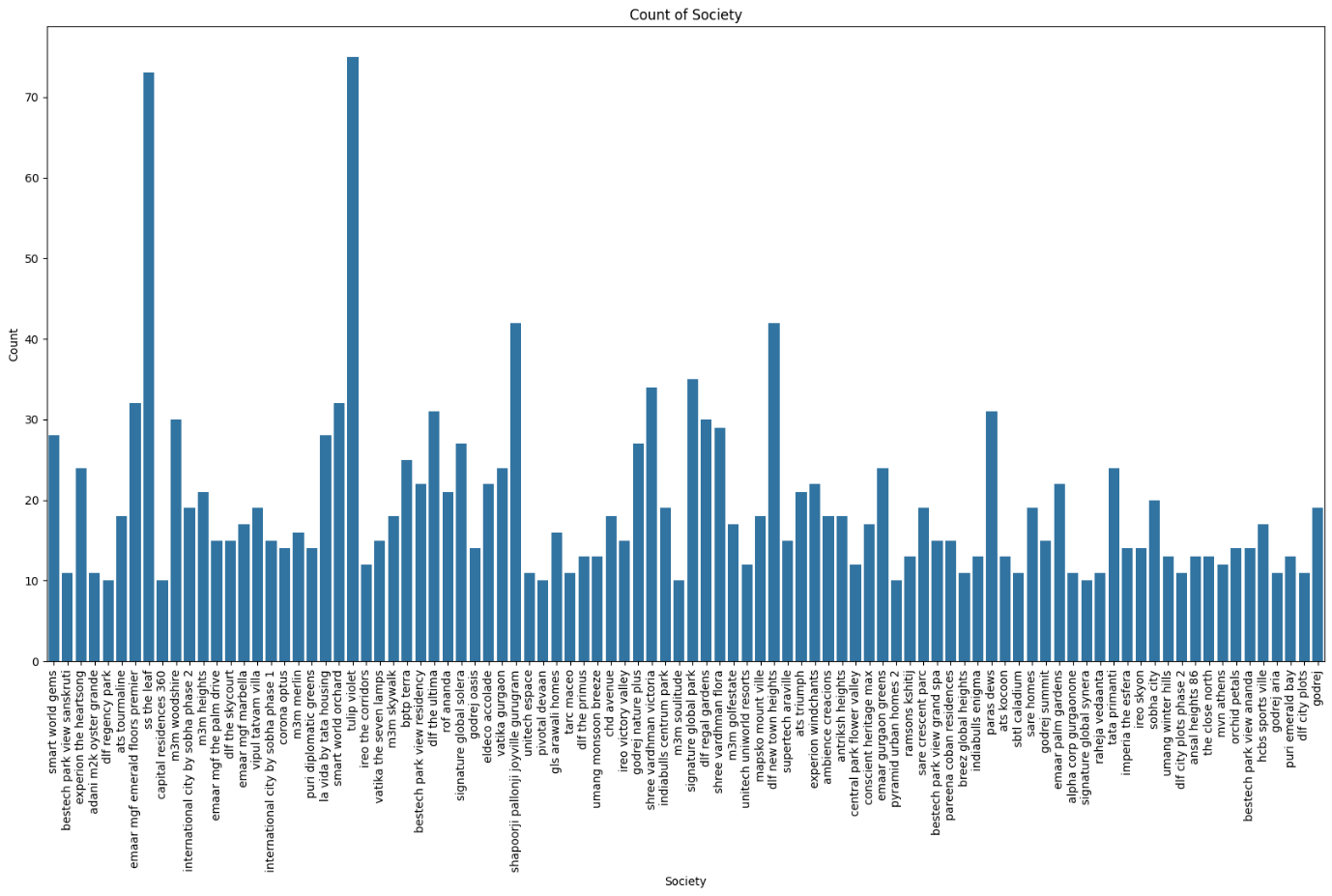


**1.**We are now performing univariate analysis for the categorical columns like property\_type, society, sector, bedRoom, bathroom(bathroom and bedroom are categorical as it is about number of bathrooms or bedrooms in each flat or house)

**2.Creating a count plot:** For each of these columns we are going to plot a count plot using seaborn”s function.This is for visualizing the frequency distribution of categories within each categorical column

**3.**Through this we are trying to find the number of time a category is occurring in this categorical column





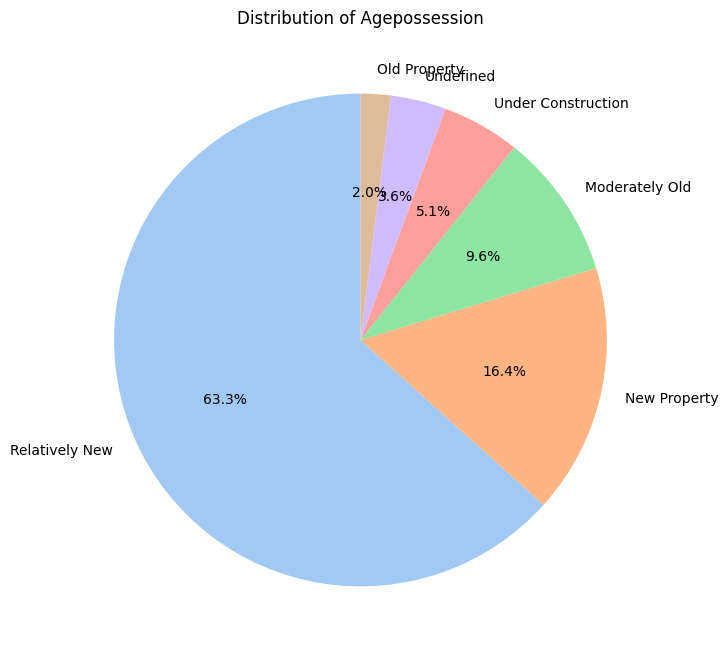
**Observation:**

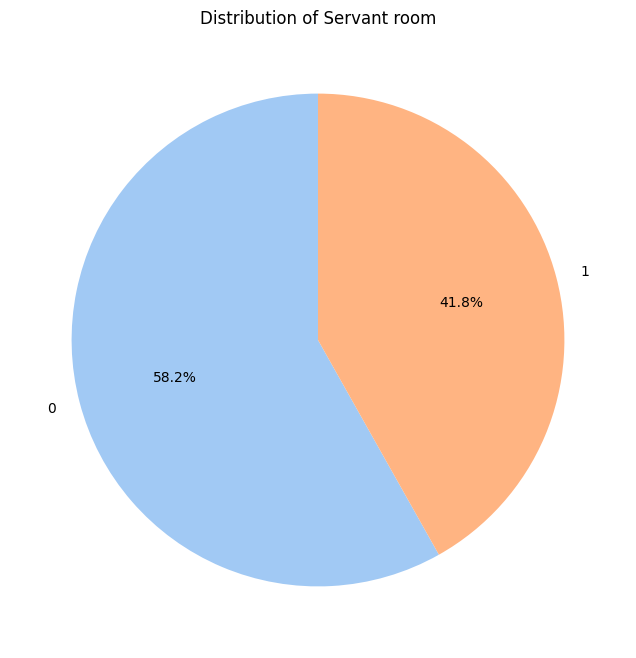
We observe that:

* The count plot for each of this categorical column gives insights about the distribution of categories
* We can also say that for bedroom and bathroom plots we can observe the number of flats having bedrooms (1, 2..) or bathrooms (1,2, 3..) so we can interpret which is most occurred in properties.

**1. Now we are performing univariate analysis for the categorical columns like agePossesion, balcony, study room, servant room, Pooja room, Furnishing type**

**2.Creat a pie chart: We will be creating a pie chart for each of these columns to visualize the distribution of categories within each of the categorical columns.**





**Observation:**

We observe that:

* Each pie chart represents the distribution of categories within specific categorical column the wedge corresponds to the proportion of each category.
* We can also say that it shows that percentage of flats have that category from the categorical column.
* That is like whether the flat has pooja room or not, etc similarly it is for all these other columns.

**MULTIVARIATE ANALYSIS**

1.Now we will be performing multivariate analysis for all the necessary columns. We are going to use scatter plots for numerical data columns and histograms and boxplots for the categorical data columns this done with target column(price).

2. Now we are performing multivariate analysis for categorical column:property\_type, society, sector, facing, agePossession, balcony, study room, servant room, pooja room, furnishing type and numerical columns: area, bedRoom, bathroom, floornum, super\_builtup\_area, built\_up\_area, carpet area, Luxury\_score

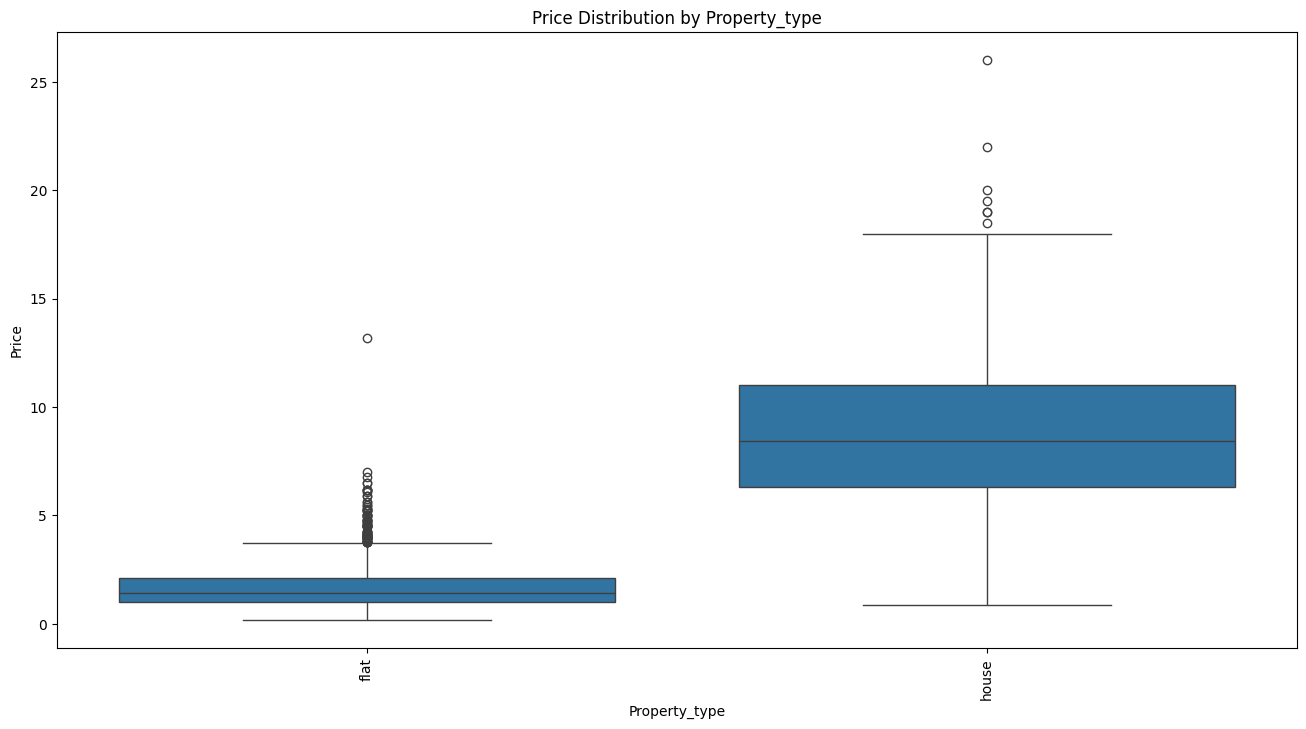
**3. Boxplots for categorical columns:** Using Seaborn's boxplot () function to create box plots that display the price distribution for each category within the column. The prices for various categories are shown using box plots, which also show the median, quartiles, and possible outliers.

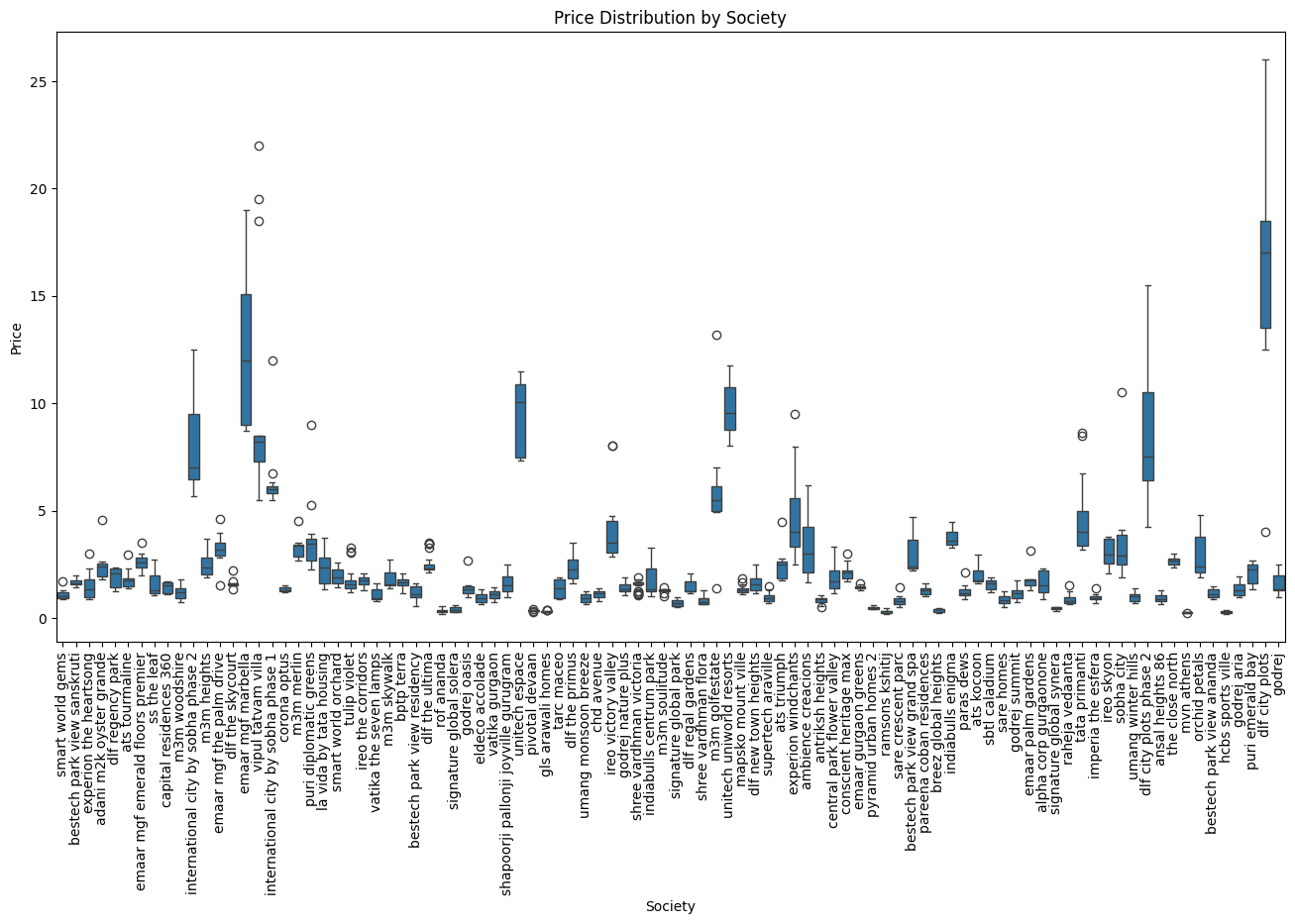
**4. Scatterplot for numerical columns: Creating** scatter plots that illustrate the relationships between each numerical column and the 'price' column using Seaborn's scatterplot () function. The relationship or correlation between prices and numerical features is made easier to see with the use of scatter plots.

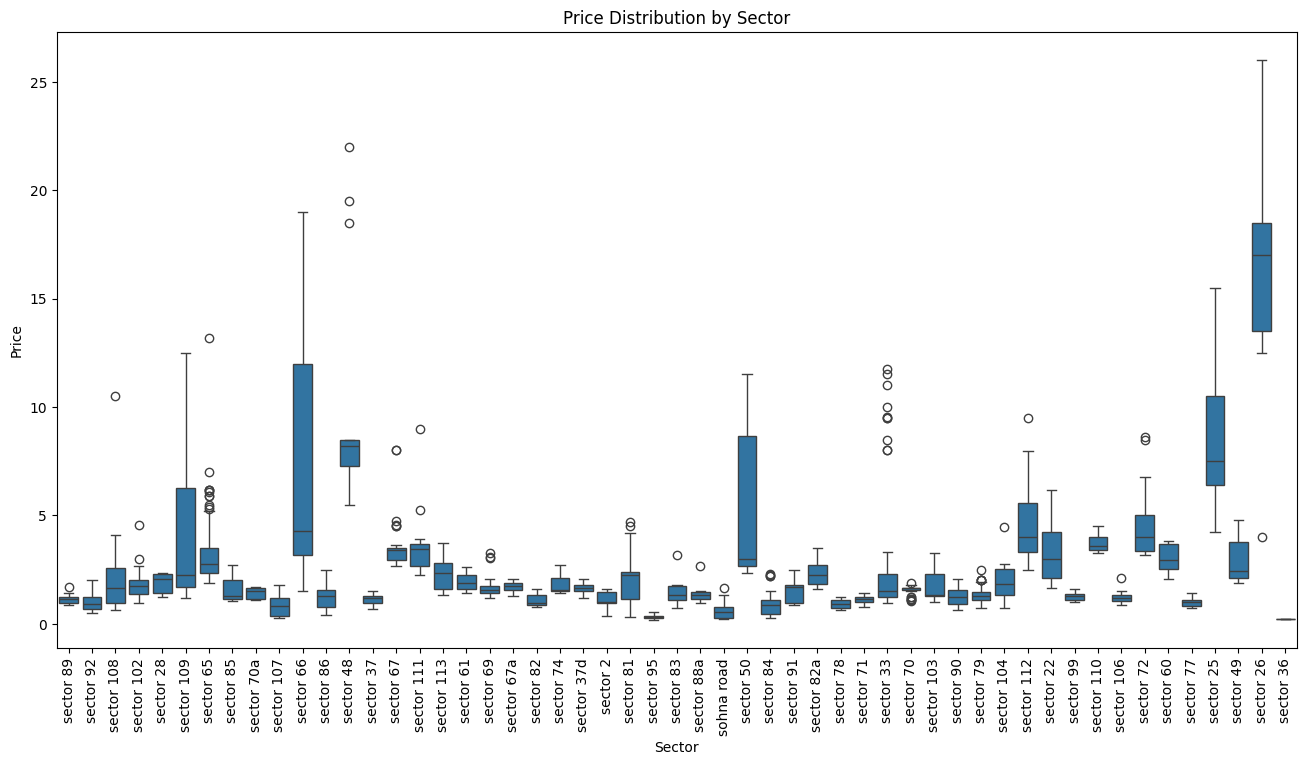
**5.Facet grid plot for categorical columns**: we are going to check for each categorical column:

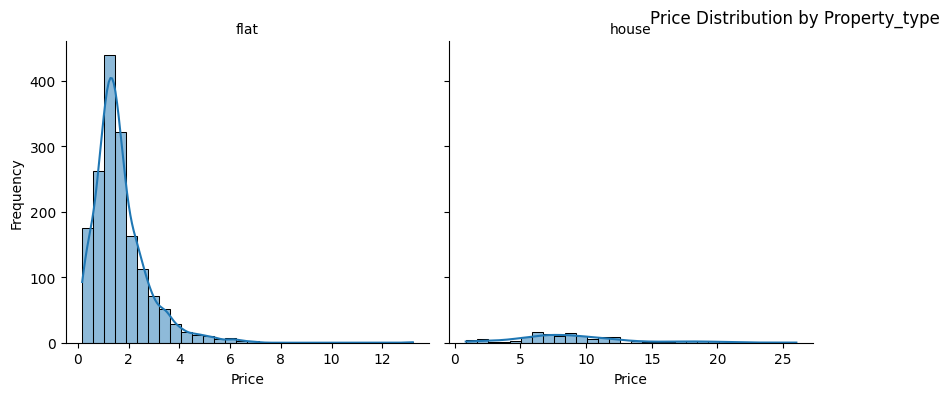
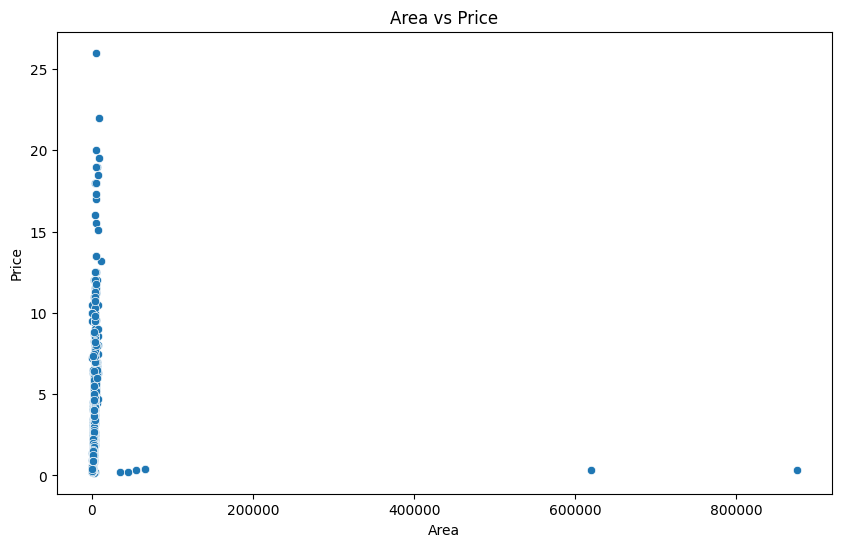
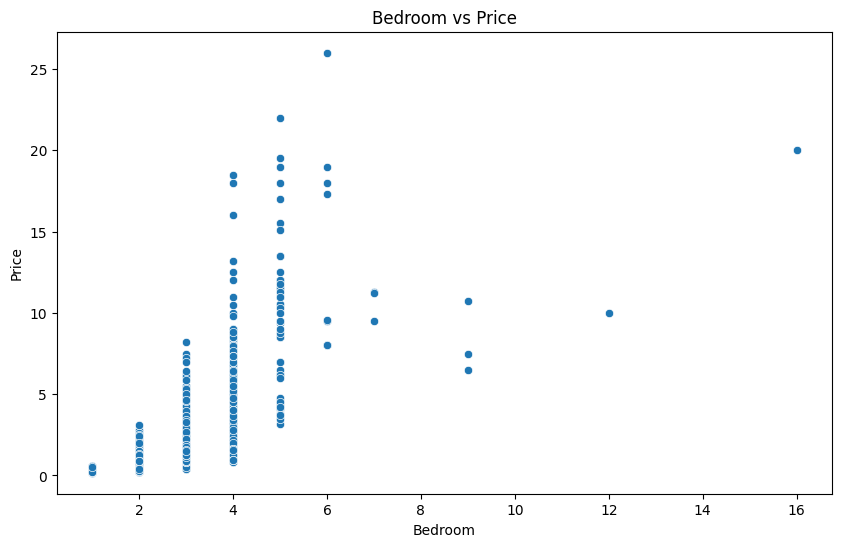
* Creating a facet grid (sns.FacetGrid) with columns according to the categorical column's unique categories.
* Then creates a price histogram plot (sns.histplot) inside the facet grid for every category.
* Based on the column name, creates titles for all facet grid axis labels are set for easier understanding. Reorganizes the subplot arrangement to improve spacing.

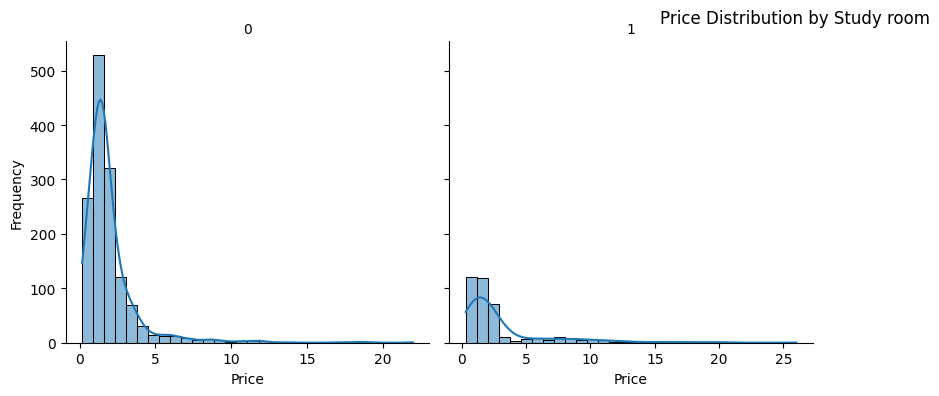
shows the grid plot of the flat.





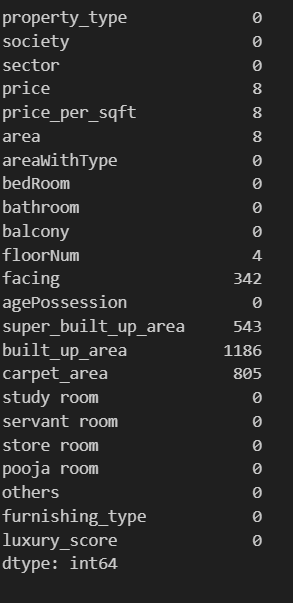






TASK-2

We are checking the total number of missing values in each of the columns in data set



**Observation:**

* The total number of null values for each column in the DataFrame d2 will be shown in the output.Understanding the dataset's completeness and identifying columns with missing data that might need additional research or imputation are made easier with the help of this information.
* Only price,price\_per\_sqft, floornum,facing,super\_built\_up\_area,built\_up\_area,carpet\_area columns have missing values.

We are going to check the shape of the dataset presently.



**Observation:**

We observe that:

* The rows in the dataset at present are 1819 this will help us later to analyze if the missing values are handled properly or not.

We are now going to calculate the percentage of null values in each of the column in the data frame



**Observation:**

We observe that:

* Output will display the percentage of null values for each column
* This provides insight on the amount of missing data in relation to the dataset's overall size.

We will now print the total number of missing values in the data.

**Observation**

:we observe that:

* We have total of 2904 missing values in the dataset.

We will now print the percentage of total number of missing values in the data.

Inserting image...

**Observation:**

we observe that:

* 6.94% of dataset has missing values

We are now checking the number of missing values in the price column and print.



**Observation**

:We observe that:

1.There are 8 missing values in the price column in total.

1.We are now going to makes a duplicate of DataFrame d2 and puts it on d3. By doing this, the original DataFrame d3 is unchanged.

2.Now we will use the median() method to determine the median value of the 'price’' column.

3.We use the fillna() method to calculate the median value and replace any missing values in the 'price’ column.



**Observation:**

We observe that:

* In order to ensure data completeness, the code checks that the 'price\_per\_sqft' column contains no more null values after imputation.We get zero hence satisfied.

Similarly we check for floornum, area , price\_per\_sqft, we check the number of missing values, duplicate datasets for each and impute using the median imputation method.





1.Now for builtup\_area and super\_built\_up\_area we will use mean ratios to impute the missing values.

2.Finding the Mean Ratio: We calculates the average ratio of "built\_up\_area" to "super\_built\_up\_area" for rows in which both columns have values.

**3.Functions of Imputation:**

* We define two functions to impute missing values based on the determined mean ratio: impute super\_built\_up() and impute built\_up().
* The function impute\_super\_built\_up() multiplies the non-null 'built\_up\_area' value by the mean ratio in order to impute missing values in the 'super\_built\_up\_area' column.
* The 'built\_up\_area' column's missing values are imputed by the impute\_built\_up() function, which divides the non-null 'super\_built\_up\_area' value by the mean ratio.

4. Using Functions for Imputation: using the apply () method to apply the defined imputation functions to the DataFrame d5 along the rows (axis=1). Using the specified logic to impute missing values in the "built\_up\_area" and "super\_built\_up\_area" columns.



**Observation:**

We observe that:

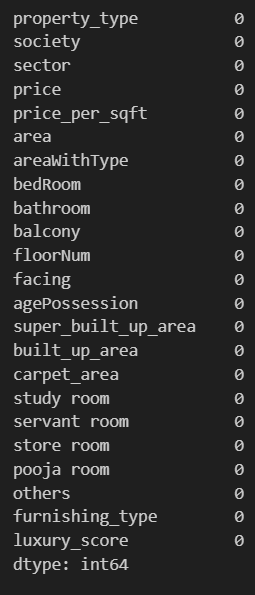
* Following the application of the imputation logic, missing values should be imputed based on the calculated mean ratio in the 'built\_up\_area' and'super\_built\_up\_area' columns.
* With the help of this imputation technique, missing values are filled in a way that makes sense given the dataset's relationship between "super\_built\_up\_area" and "built\_up\_area."

Similarly, we follow this method for built\_up\_area and carpet\_area.

And then we print the null values for the above three column we get null missing values.

1.For the facing column we use the mode imputation to handle the missing values same as price columns and others.

And finally, we print missing values for the whole dataset, and we get null for all columns and all the columns missing values are handled.



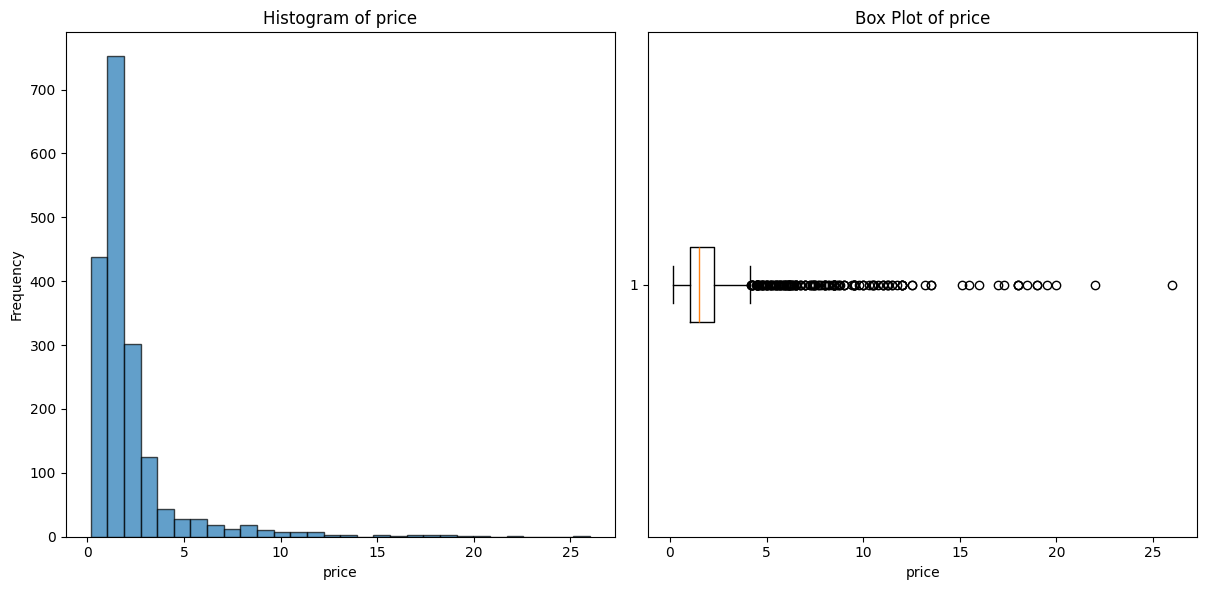
**TASK-3**

**Outlier Detection:**

1. Using box plots to visualize the data aids in the detection of outliers, which, if improperly managed, can have a major impact on the performance of the model.
2. Investigating the link between each attribute and the goal variable—price—is made easier with the use of scatter plots. This is essential to comprehending the ways in which various factors affect the target variable

Process-

1. For each numerical feature, we produced descriptive statistics (mean, median, standard deviation, etc.) that gave a summary of the information.
2. For every feature, we make a figure with three subplots: a scatter plot against price, a box plot, and a histogram.

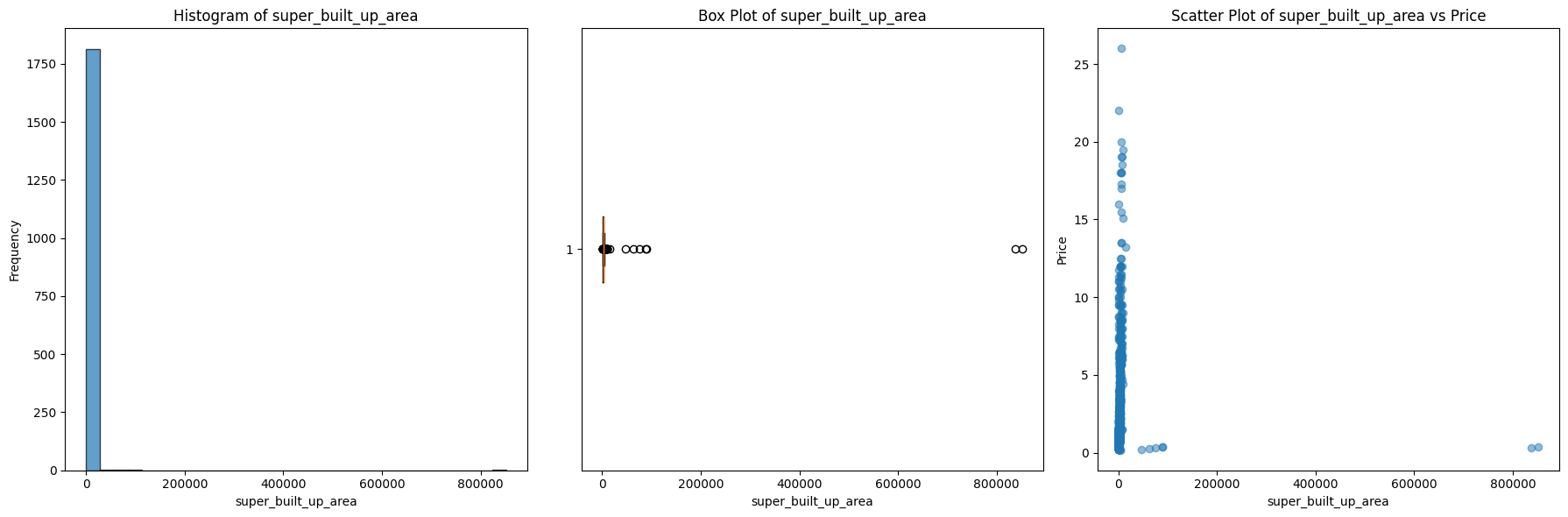


A screenshot of a computer

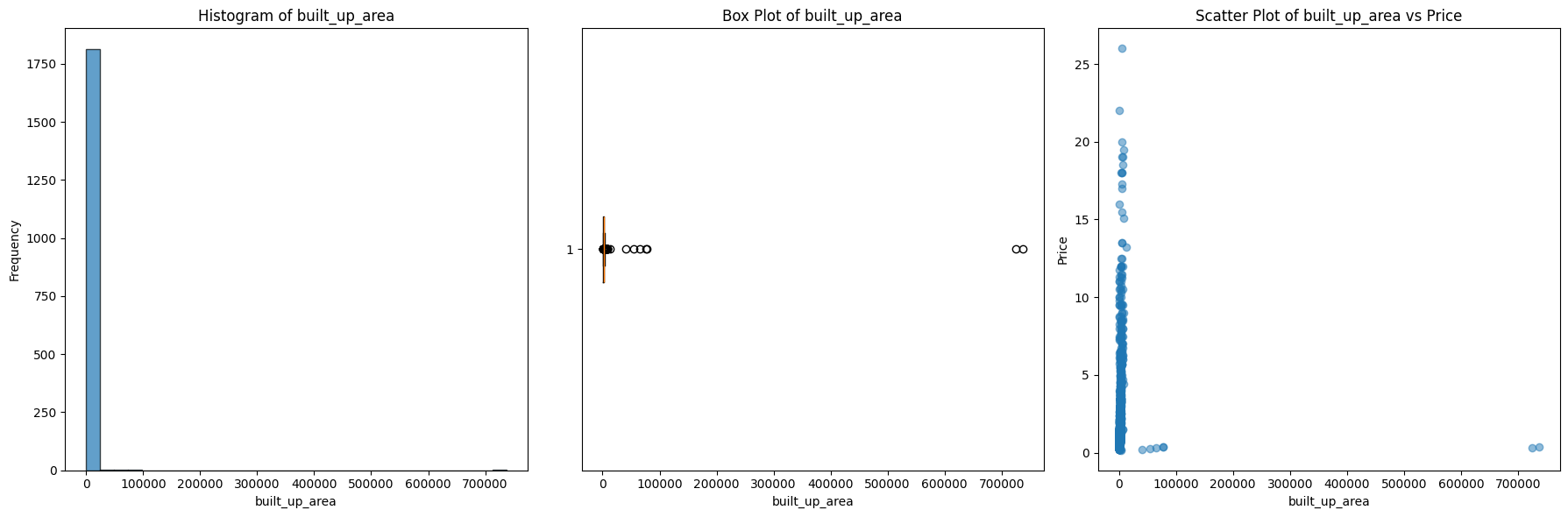
Description automatically generated

A white square with black lines

Description automatically generated

A screenshot of a computer

Description automatically generated

**TASK-4**

**Plotting the distributions of numerical variables:**

1) List the desired numerical features from the dataset.

2)We created a grid of subplots, with each subplot representing one numerical characteristic.

3)We fit a normal distribution (Gaussian distribution) to the feature data to obtain the mean and standard deviation.

We create a histogram of the data.

We apply the probability density function (PDF) of the fitted normal distribution to the histogram.

A graph of a graph

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated with medium confidence

Similarly, distribution plots were considered for remaining numerical variables- carpet area, bedroom, bathroom, luxury score

**This helps in understanding the distribution, skewness. The histogram and the overlaid normal distribution can help in spotting outliers.**

**Handling Outliers:**

We provide the columns (numerical) we wish to analyse for outliers.

For every column:

1) To obtain basic statistics, we describe the column with the describe() function.

2)Before dealing with outliers, we create a boxplot to visually represent the distribution and detect probable outliers.

3) Use the Interquartile Range (IQR) to establish the top and lower boundaries of outliers.

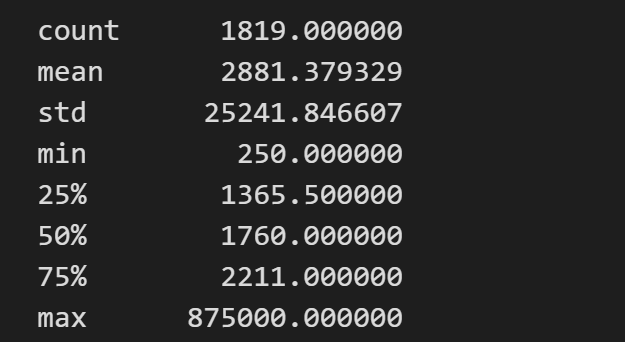
Lower Limit = Q1 - 1.5IQR, Upper Limit = Q3 + 1.5IQR, where Q1 and Q3 are the 25th and 75th percentiles, respectively.

4) Replace outliers with computed upper and lower limits- CAPPING

5) After managing outliers, draw a boxplot to validate that they have been removed.

For example consider- area column

Descriptive statistics:



Before Removing Outliers:

A white rectangular object with black dots

Description automatically generated

After Removing Outliers:

A blue rectangular object with black lines

Description automatically generated

**floorNum column**

Descriptive Statistics:

A black rectangle with white text

Description automatically generated

A graph with a blue rectangular bar

Description automatically generated

A black background with white text

Description automatically generatedA graph of a box with a number of numbers

Description automatically generated with medium confidenceCapping values for Outliers:

Similarly using the same criteria (IQR) all other numerical columns – price\_per\_sqft, super\_built\_up\_area, built\_up\_area, carpet\_area, bedRoom, bathroom, luxury\_score were made free of outliers .

**The reason to choose IQR over Z-score is all the numerical variables do not have normal distribution. They are mostly right skewed. So, Z-score cannot be applied over here.**

**Handling the categorical variables:**

1. Machine learning models usually require numerical input. Frequency encoding translates categorical data to numerical representation by calculating the frequency (or proportion) of each category in the dataset.
2. For some models, such as decision trees and gradient boosting, frequency encoding can outperform alternative approaches, such as one-hot encoding.
3. This preprocessing step helps to improve model performance and efficiency.

Process-

1. We've created a list of categorical variables in the dataset that we wish to encode.
2. For each category feature, we conduct Frequency Encoding:

* We use value\_counts(normalize=True) to determine the frequency of each unique value in the feature. This calculates the relative frequency (percentage) of each category.
* Next, we create a new column in the dataframe called with suffix \_freq.

After performing encoding, we have these columns left:

price float64

price\_per\_sqft float64

area float64

bedRoom float64

bathroom float64

floorNum float64

super\_built\_up\_area float64

built\_up\_area float64

carpet\_area float64

study room int64

servant room int64

store room int64

pooja room int64

others int64

furnishing\_type int64

luxury\_score float64

property\_type\_freq float64

society\_freq float64

sector\_freq float64

balcony\_freq float64

facing\_freq float64

agePossession\_freq float64

dtype: object

**Feature Selection:**

1. The major purpose of this exercise is to determine which characteristics are most essential for predicting the target variable (price) with a machine learning model
2. It improves model performance by emphasizing the most important elements. It reduces model complexity by ignoring minor features.

**Process-**

1. The set of feature columns that will be utilized to forecast the target variable—price—is defined.
2. We use the features and target variable from the dataframe to build a Random Forest Regressor model and train it. The model will employ 100 trees, as indicated by the n\_estimators=100 parameter. Reproducibility is ensured by random\_state=42.
3. We store the features and their corresponding importances in a new dataframe and then sort it in descending order.

Feature Importance

0 price\_per\_sqft 0.562273

1 area 0.375637

3 bathroom 0.027278

15 property\_type\_freq 0.013776

5 super\_built\_up\_area 0.010424

7 carpet\_area 0.002023

2 bedRoom 0.001957

6 built\_up\_area 0.001768

16 society\_freq 0.001638

4 floorNum 0.000924

17 sector\_freq 0.000626

14 luxury\_score 0.000610

18 facing\_freq 0.000296

19 agePossession\_freq 0.000174

20 balcony\_freq 0.000161

13 furnishing\_type 0.000159

9 servant room 0.000145

8 study room 0.000050

10 store room 0.000030

11 pooja room 0.000028

12 others 0.000021

**Machine Learning Model for Price Prediction:**

1. We develop a predictive model that estimates property prices based on certain variables by utilizing Gradient Boosting Regression. To make sure our model is both effective and efficient, we use the most crucial features that have already been found.
2. Using measures such as Mean Squared Error (MSE) and R-squared (R²), we want to determine how well our model works with hypothetical data.

**Process-**

1. The features that were shown to be most crucial for predicting the target variable—price was chosen.
2. The dataset was divided into testing and training sets. The model is trained on the training set, which comprises 80% of the data, and its performance is assessed on the testing set, which contains 20% of the data.
3. A Gradient Boosting Regressor model was constructed using the following hyperparameters:

n\_estimators=100: The quantity of stages used for boosting.

learning rate=0.1

max\_depth=3

1. The training data was then used to train this model. We used the trained model to make predictions on the test data.

To assess the performance of the model, we computed two crucial metrics, and the values are:

**MSE(Mean Squared Error): 0.010442418648929993**

**R-squared (R²): 0.9909968367082335**This low MSE shows that the model is good at forecasting property values based on the provided characteristics and indicates that it has a high degree of accuracy in its predictions.

There is a substantial correlation between the characteristics and the target variable, as indicated by its high R2 value. It suggests that the model is very trustworthy and good at identifying the underlying patterns in the data.