                     REPORT

INTRODUCTION:

                    This assignment focuses on exploring techniques for managing missing values and detecting outliers in the Gurgaon real estate dataset. It mainly focusses on preprocessing task. With crucial attributes such as property type, location, price, and area, this dataset offers insights into the Gurgaon property. By addressing missing data and outliers effectively, we aim to enhance the accuracy of our analysis, providing valuable insights for property investors and market observers.

**PROBLEM DEFINITION AND ALGORITHM:**   
                      Identifying the missing values in the dataset and handle them using appropriate strategies. Identify anomalies within the dataset utilizing statistical approaches and visual aids to pinpoint irregular data points. To handle the outliers using the suitable techniques. It gives the prediction of various prices of the flats present in the Gurgaon estate.

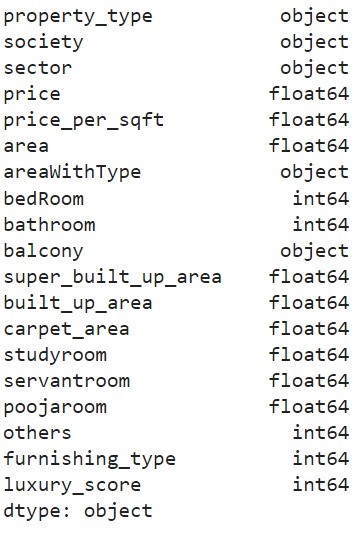
RELATED WORK:

**TASK-1**

         Loading the data set into the Google Collab

Importing NumPy as np and also pandas as pd from the python library, for graphs using matplotlib.pyplot as plt  . We identified the various features in the dataset and also their datatypes using [df.dtypes()].

These are the datatypes observed from the dataset



Next we have removed the duplicate rows using [display(df.drop duplicates())]which helps to  decrease the bias.

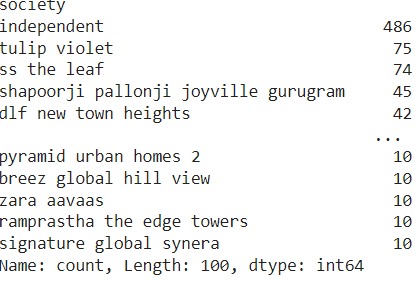
We have checked all the missing values in the given features of the dataset and displayed for both numerical and categorical features.

 After checking the missing values we have explored the property type feature and made a box plot and compared between number of flats and houses.

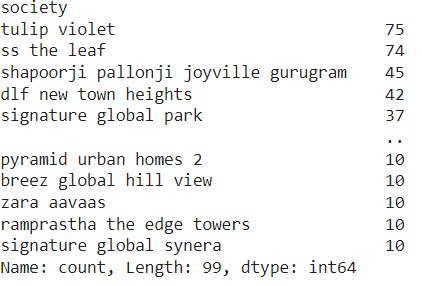
CONCLUSION:

              We have observed that number of houses are less than number of flats. Hence we have removed the independent from the 676 society’s and we are left with 675 societies.

Before removing the independent the below is the output of the society column.



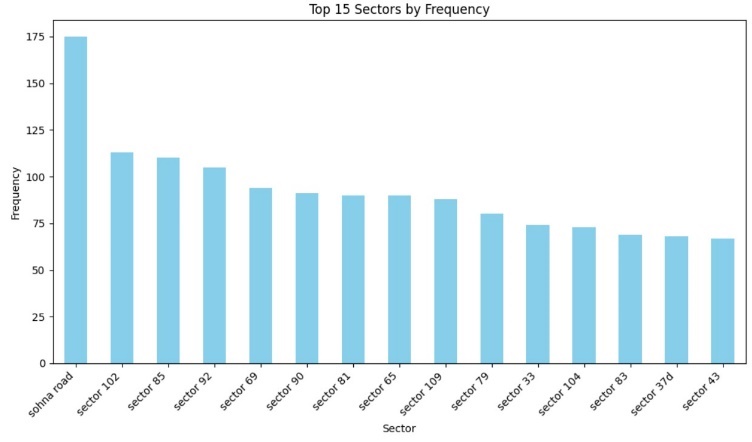
After removing the rows where society column contains independent below is the count of flats or houses for the each society.



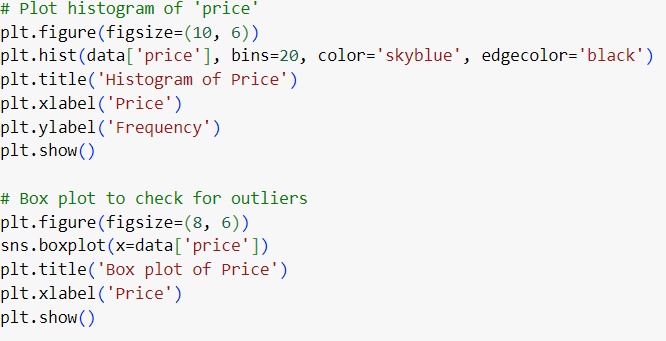
Next we are exploring the sector column using “selected\_column = data['sector']”.We have selected the top 15 sectors using “top\_15\_sectors = sector\_counts.head(15)”.

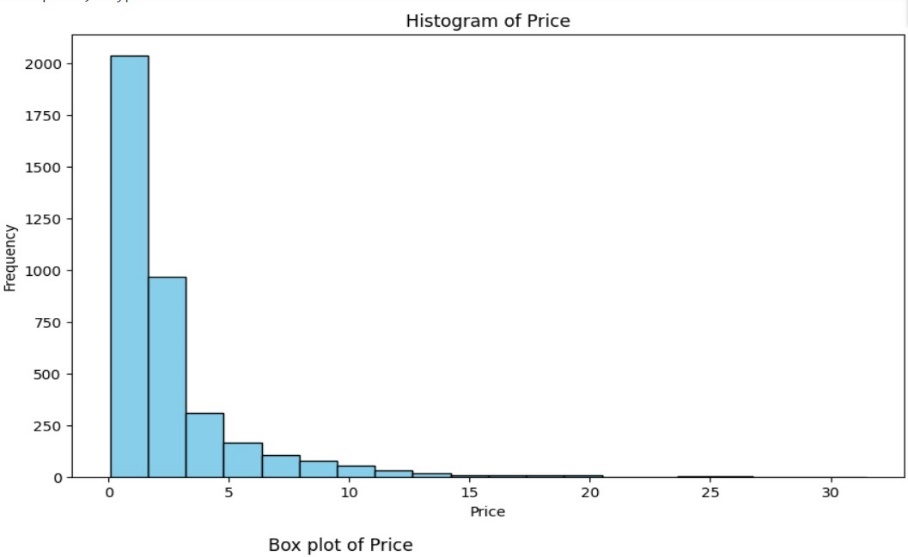


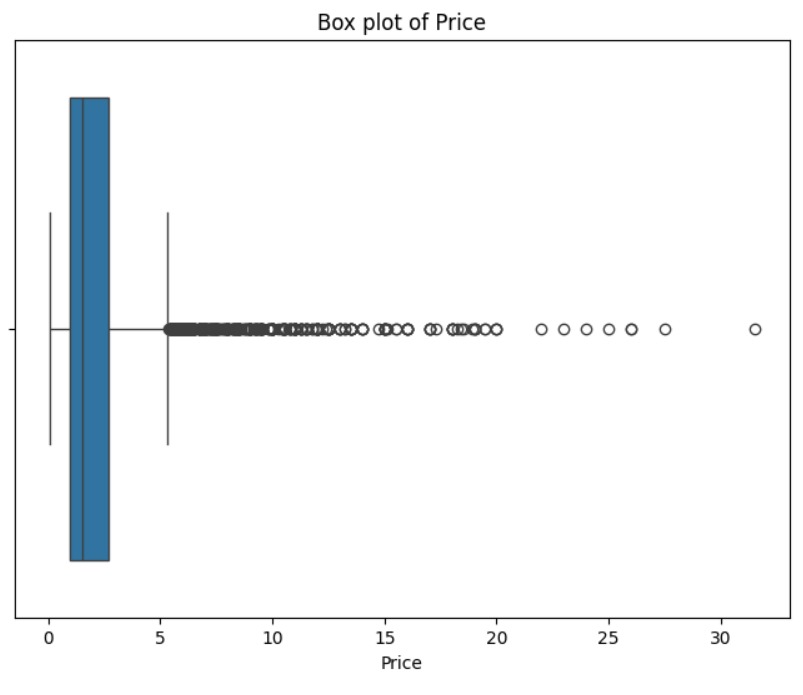
The output of the above code is below











Next we checked the skewness and kurtosis to justify the out layers using the below code.

# Compute skewness and kurtosis

skewness = numeric\_values.skew()

kurtosis = numeric\_values.kurtosis()

print(f"Skewness of '{column\_name}': {skewness}")

print(f"Kurtosis of '{column\_name}': {kurtosis}")

Next we plot the histogram , histogram with binning ,box plot of the price colum using below code

# Histogram

plt.figure(figsize=(10, 6))

sns.histplot(numeric\_values, kde=True, bins=20, color='skyblue', edgecolor='black')

plt.title(f'Histogram of {column\_name}')

plt.xlabel(column\_name)

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Binning

plt.figure(figsize=(10, 6))

numeric\_values.plot(kind='hist', bins=20, color='skyblue', edgecolor='black')

plt.title(f'Histogram with Binning of {column\_name}')

plt.xlabel(column\_name)

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Box plot

plt.figure(figsize=(8, 6))

sns.boxplot(x=numeric\_values)

plt.title(f'Box plot of {column\_name}')

plt.xlabel(column\_name)

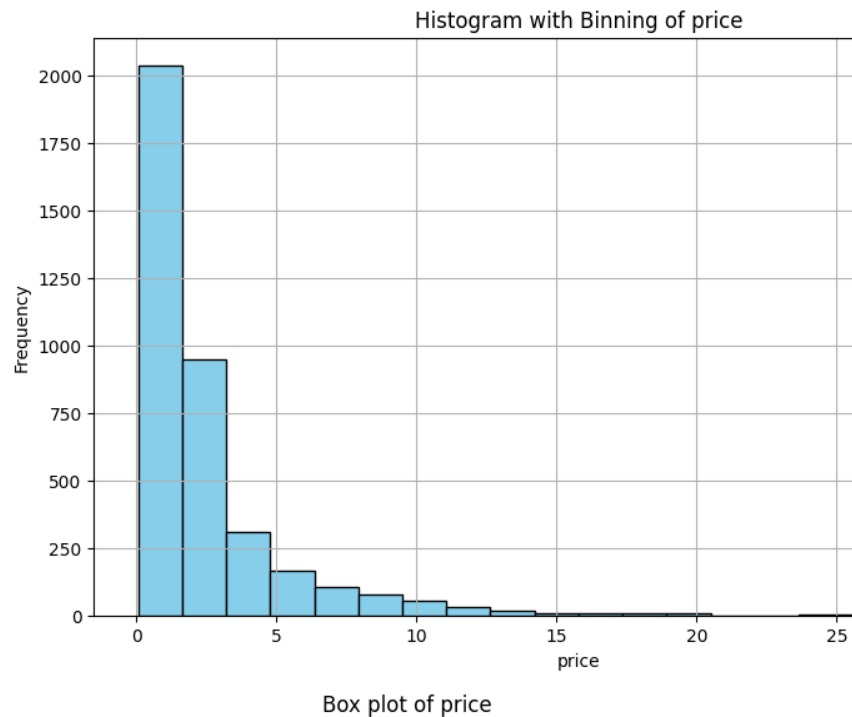
plt.grid(True)

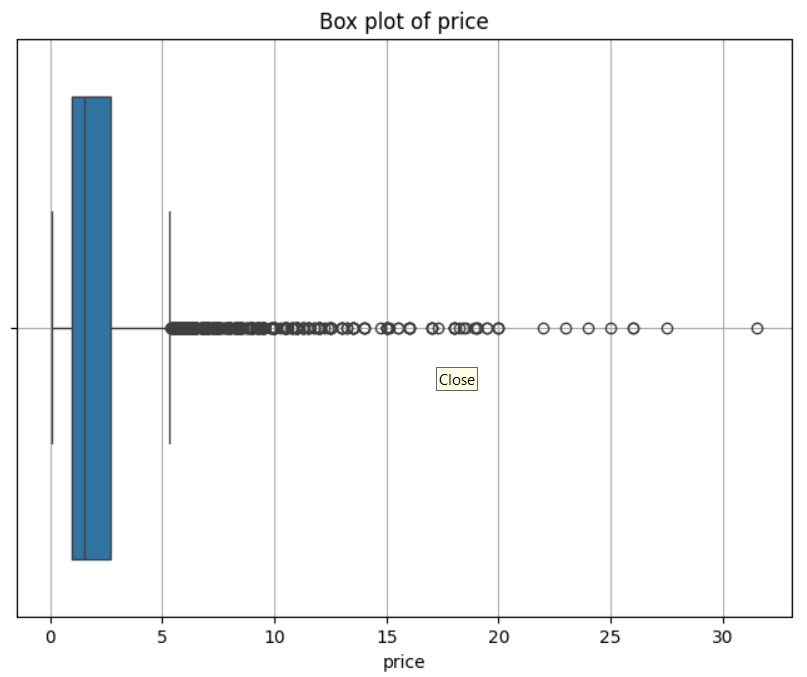
Analysis for column: price

Skewness of 'price': 3.3113346542178137

Kurtosis of 'price': 15.257818585808831







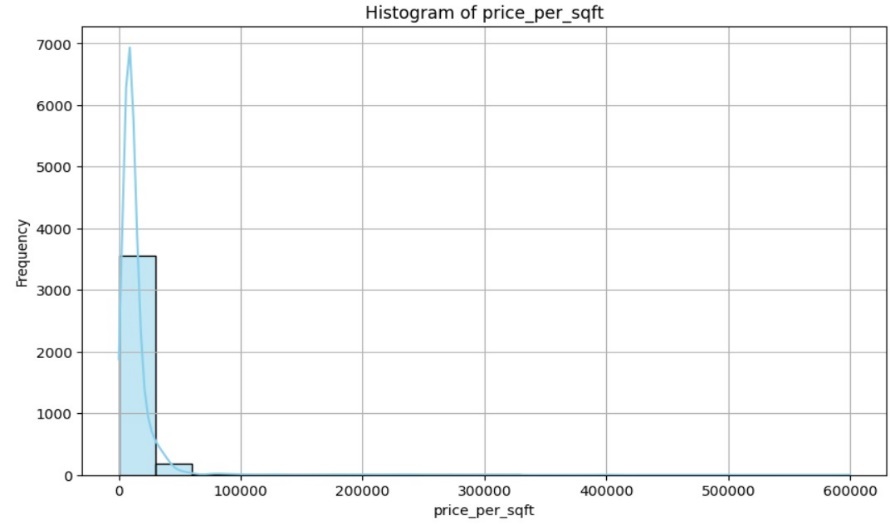
Similarly we have done for the other feature also.

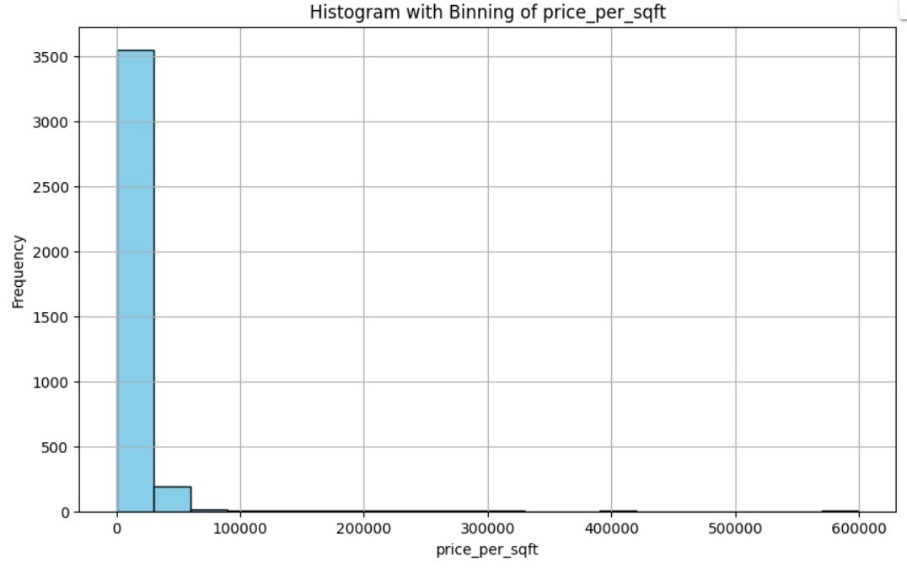
FOR PRICE PER SQRT FEATURE:

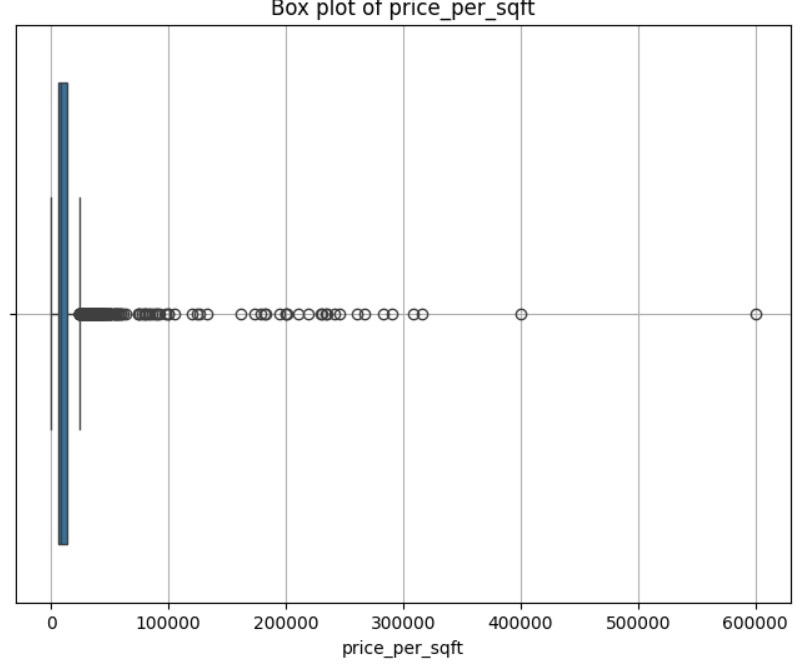
Analysis for column: price\_per\_sqft

Skewness of 'price\_per\_sqft': 11.439219959077516

Kurtosis of 'price\_per\_sqft': 187.04186603461412





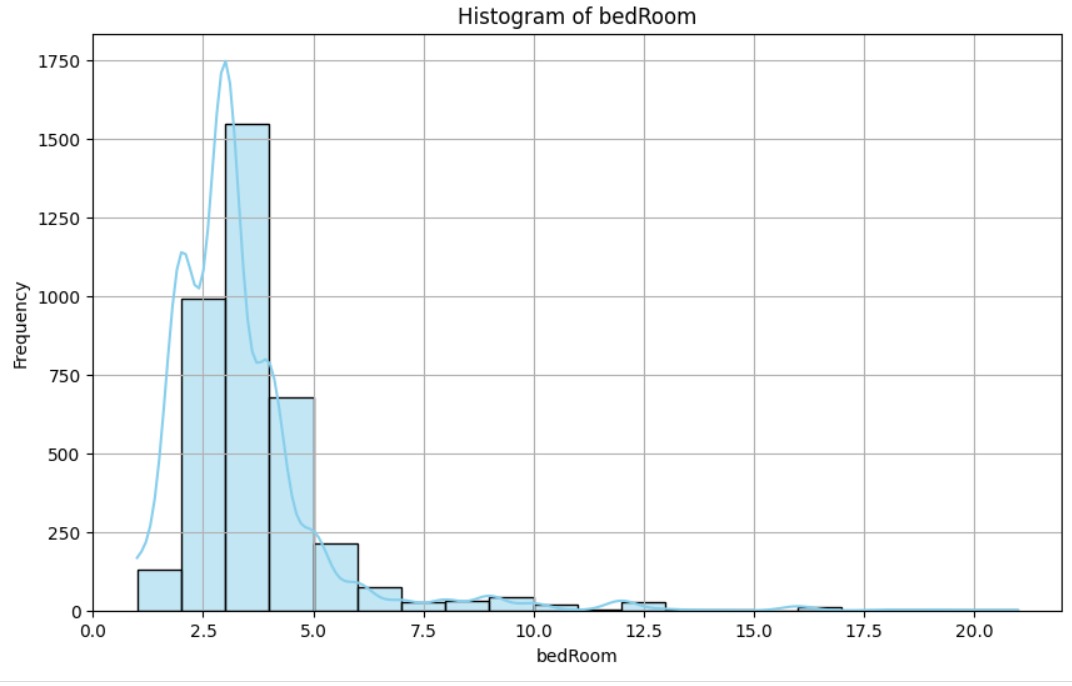


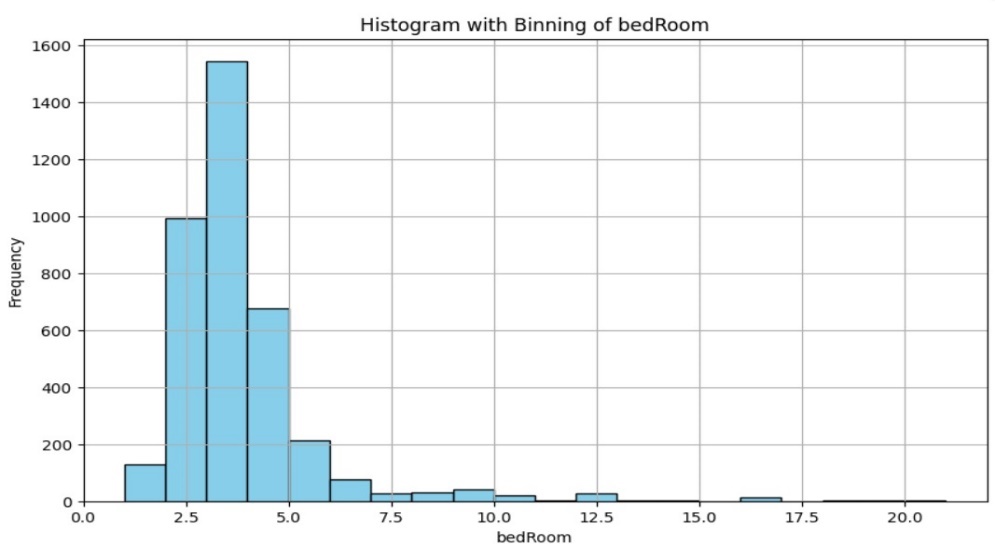
FOR THE BEDROOM FEATURE:

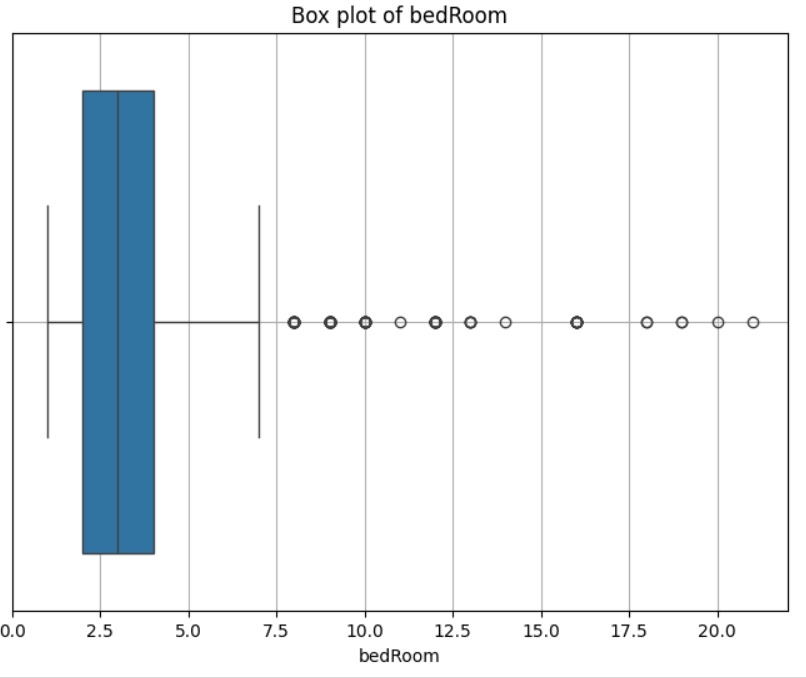
Analysis for column: bedRoom

Skewness of 'bedRoom': 3.5115390021792146

Kurtosis of 'bedRoom': 18.61025394135828



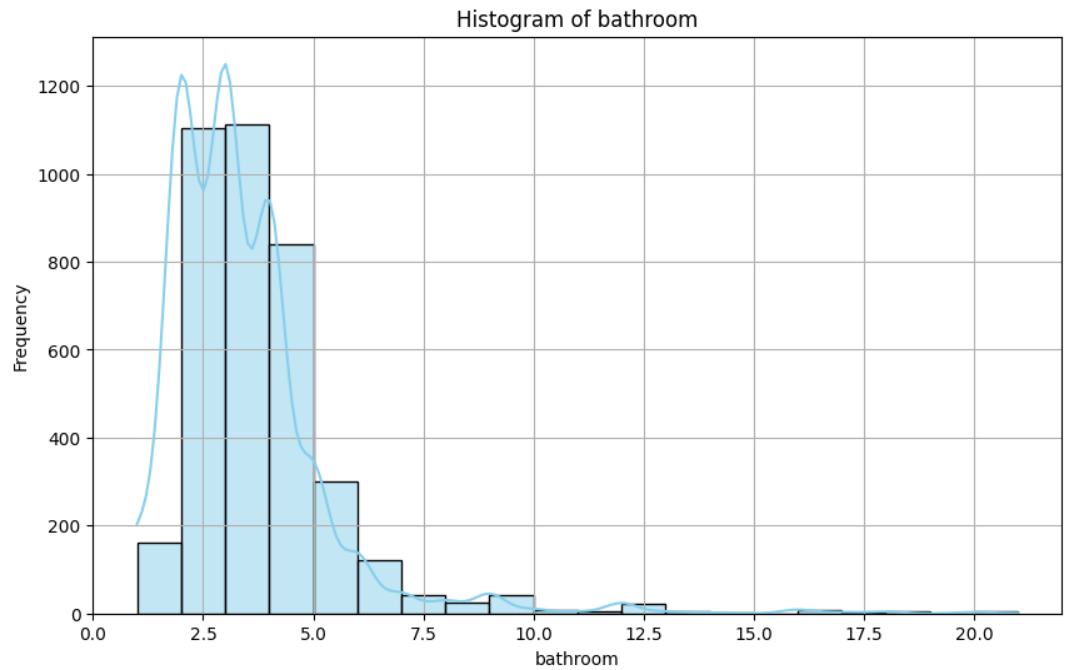


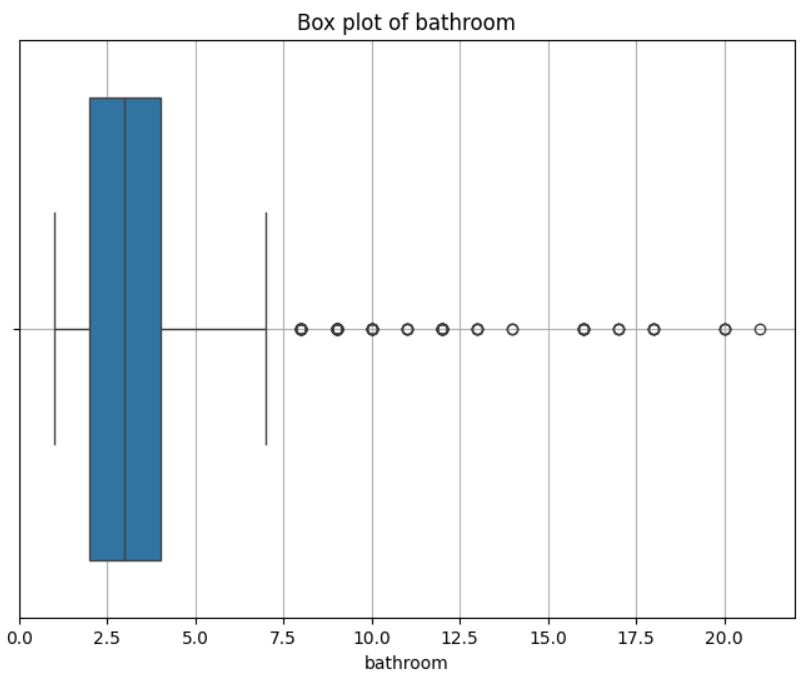


FOR THE BATHROOM FEATURE:

Analysis for column: bathroom

Skewness of 'bathroom': 3.257083204220492

Kurtosis of 'bathroom': 17.74517483567377



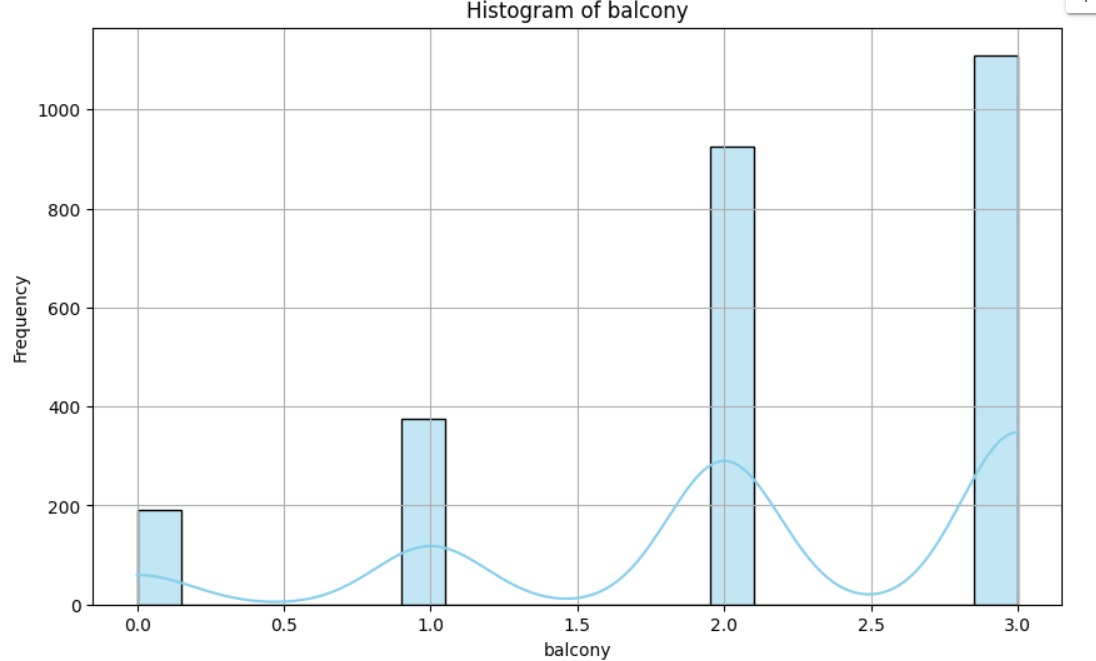


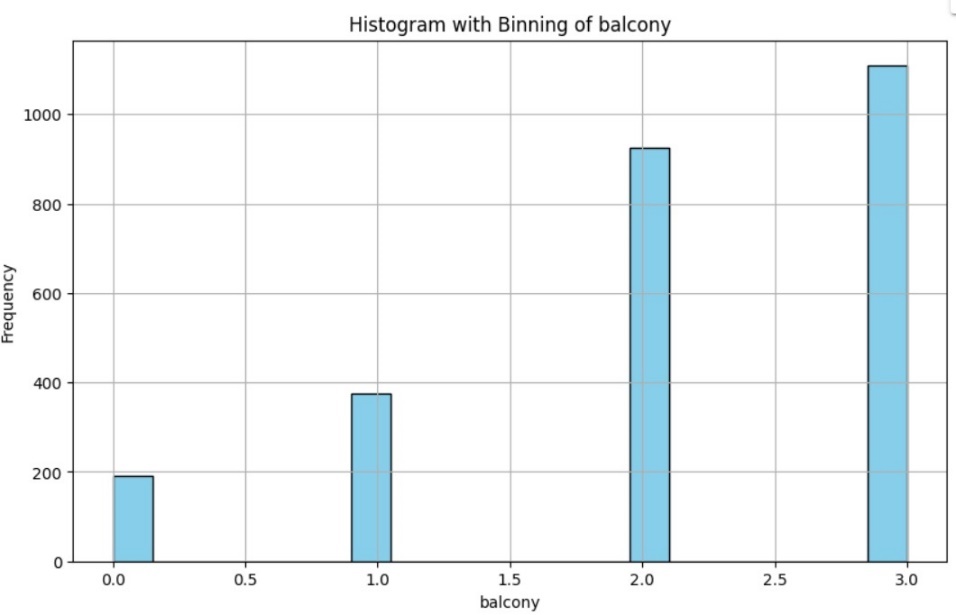
FOR THE BALCONY FEATURE:

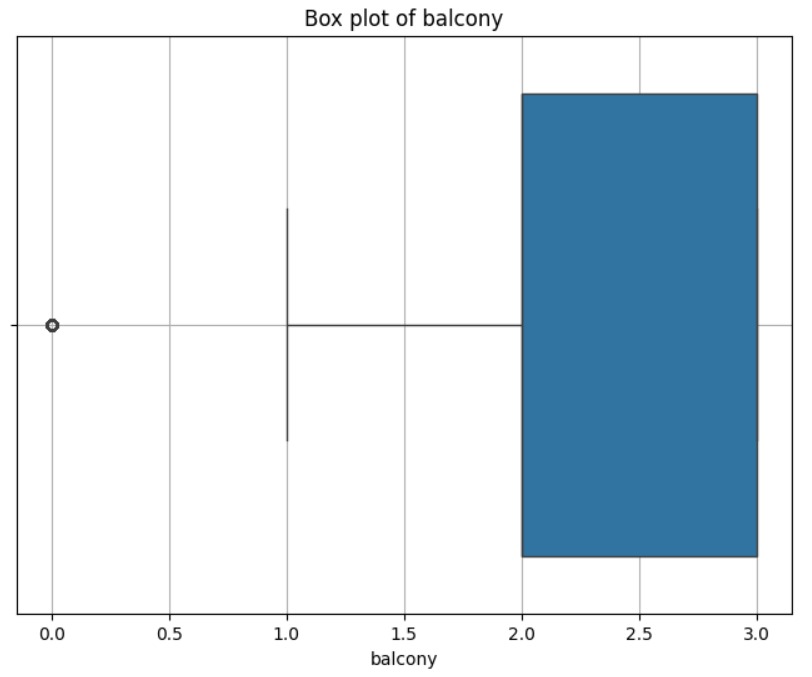
Analysis for column: balcony

Skewness of 'balcony': -0.8369569224756488

Kurtosis of 'balcony': -0.19760706152620688





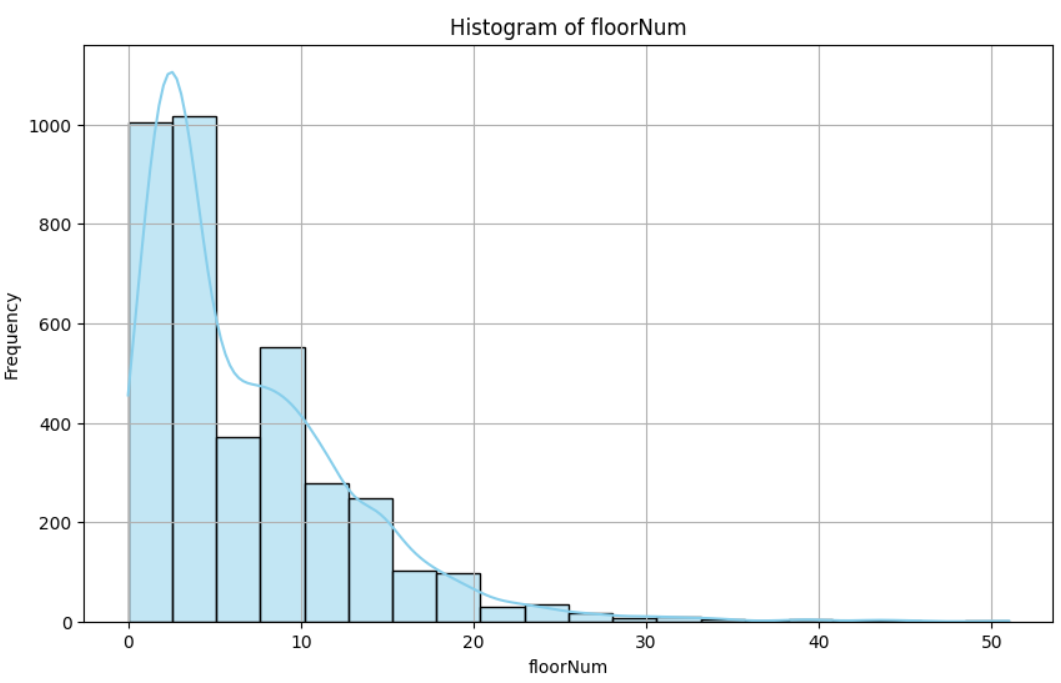


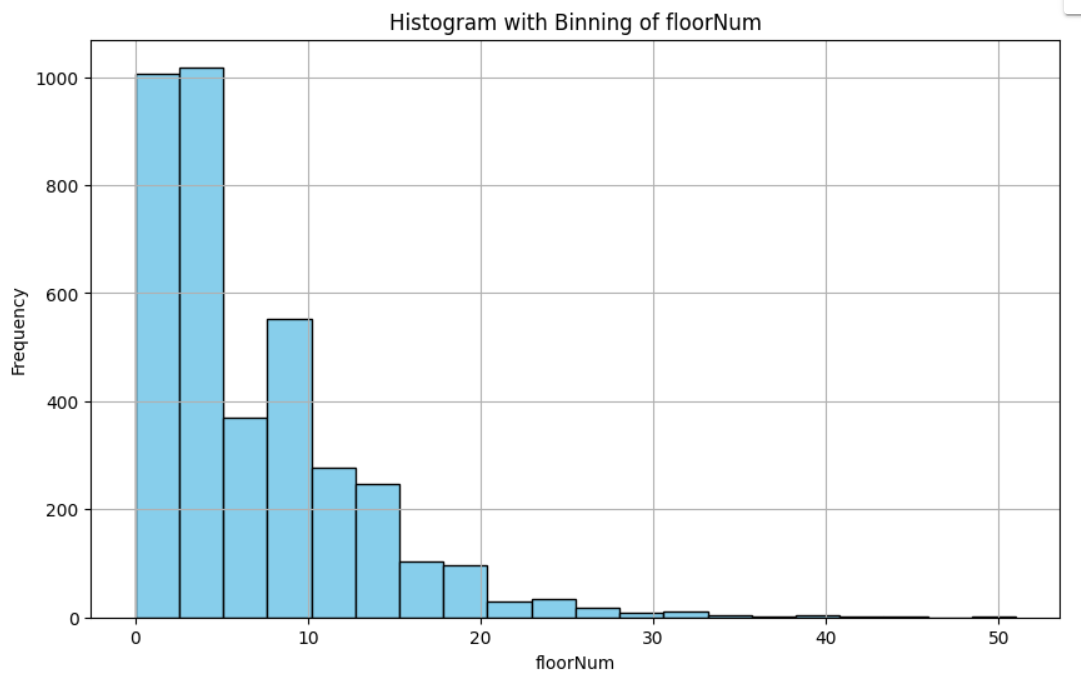
FOR THE FLOORNUM FEATURE:

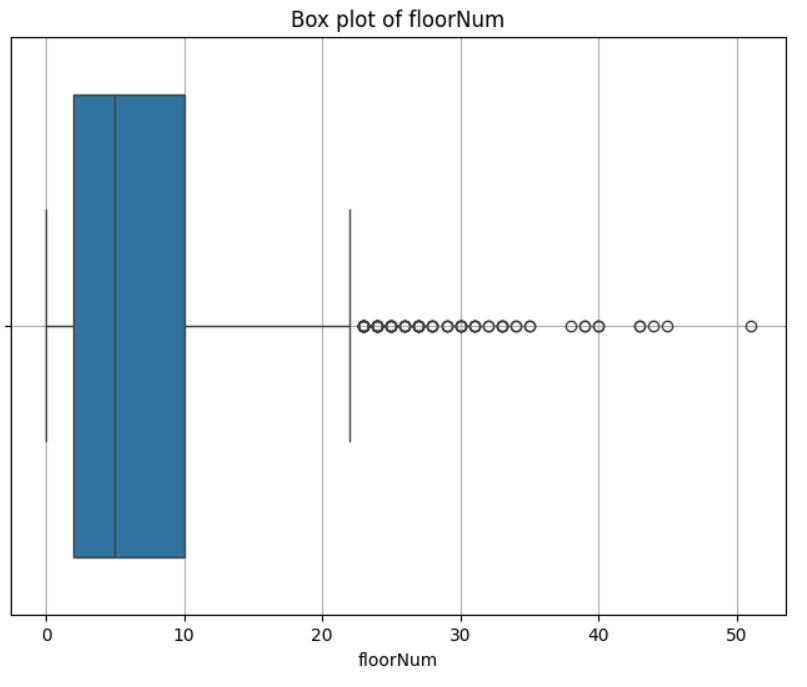
Analysis for column: floorNum

Skewness of 'floorNum': 1.6987333012368482

Kurtosis of 'floorNum': 4.549322940576371







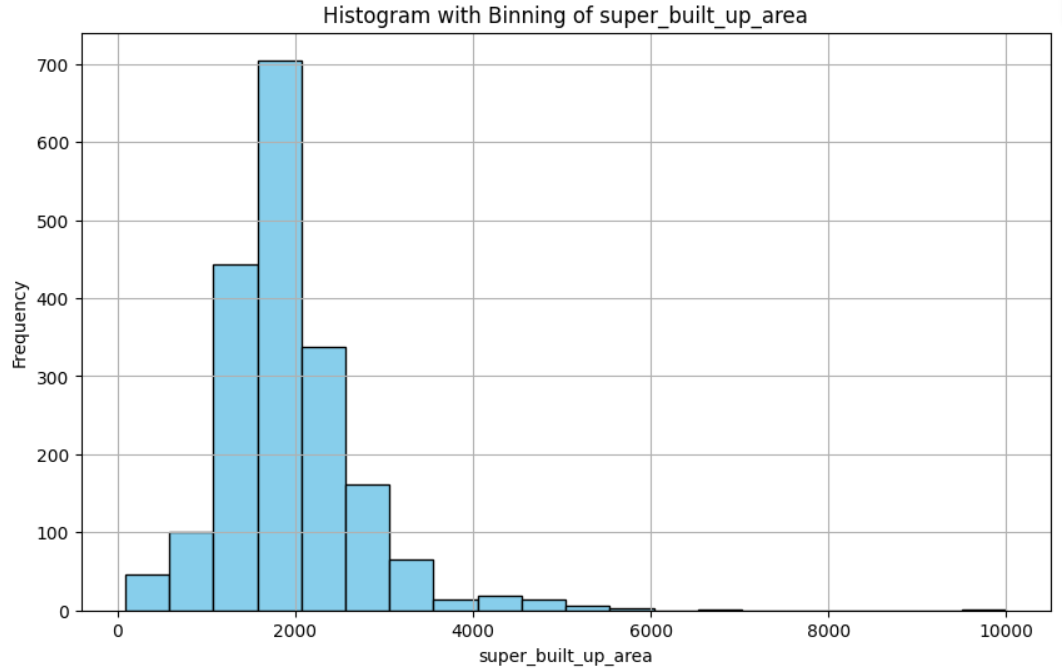
FOR THE SUPER\_BUILT\_UP\_AREA FEATURE:

Analysis for column: super\_built\_up\_area

Skewness of 'super\_built\_up\_area': 1.8232284983476958

Kurtosis of 'super\_built\_up\_area': 10.083066100658105





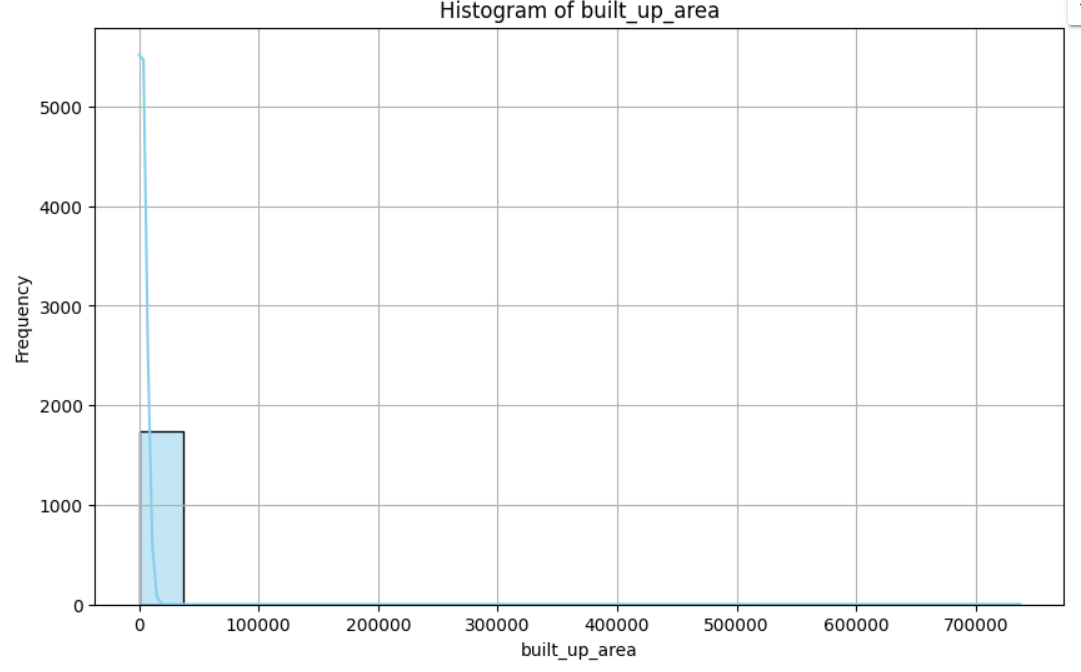


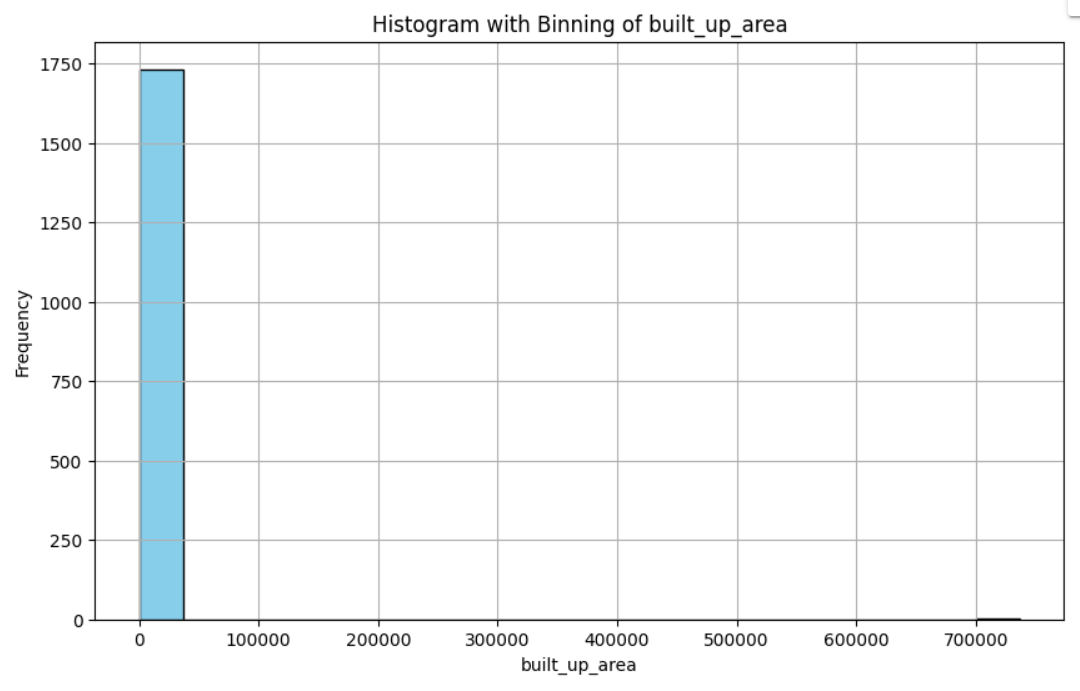
FOR THE BUILT\_UP\_AREA FEATURE:

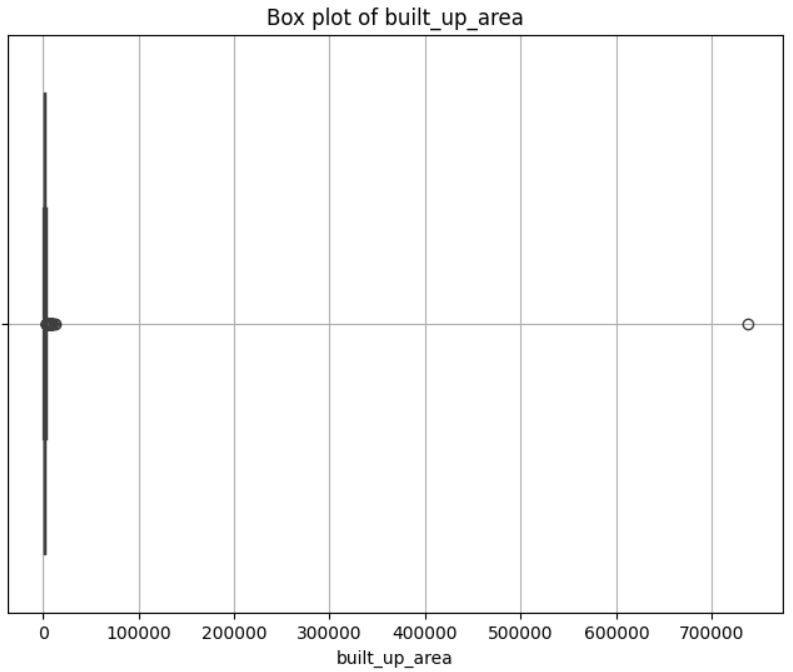
Analysis for column: built\_up\_area

Skewness of 'built\_up\_area': 41.217580082562975

Kurtosis of 'built\_up\_area': 1710.1077185788172





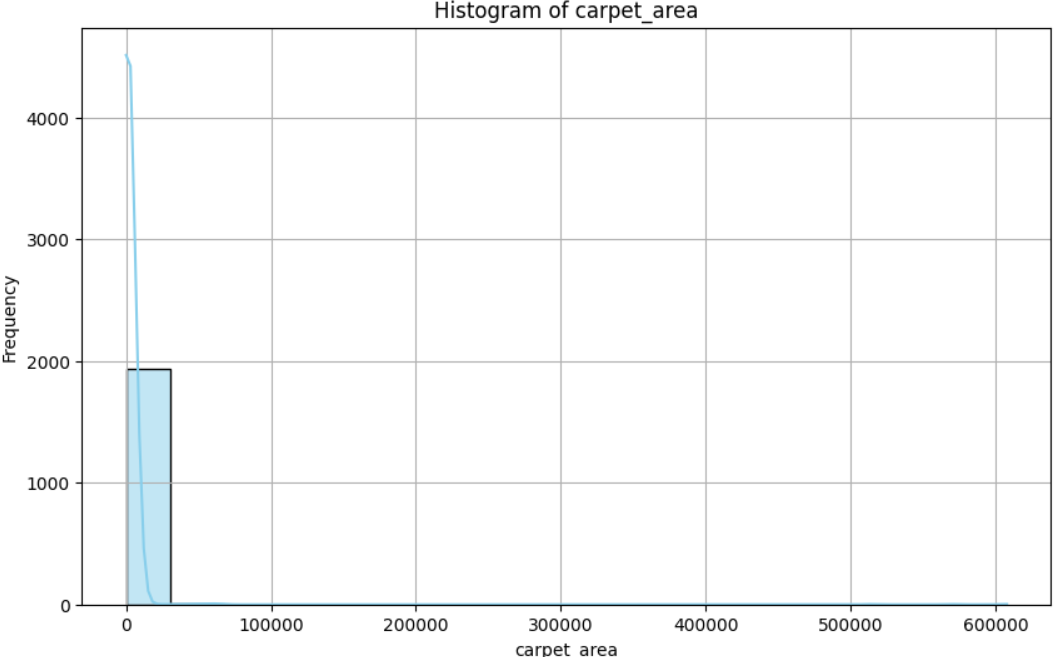


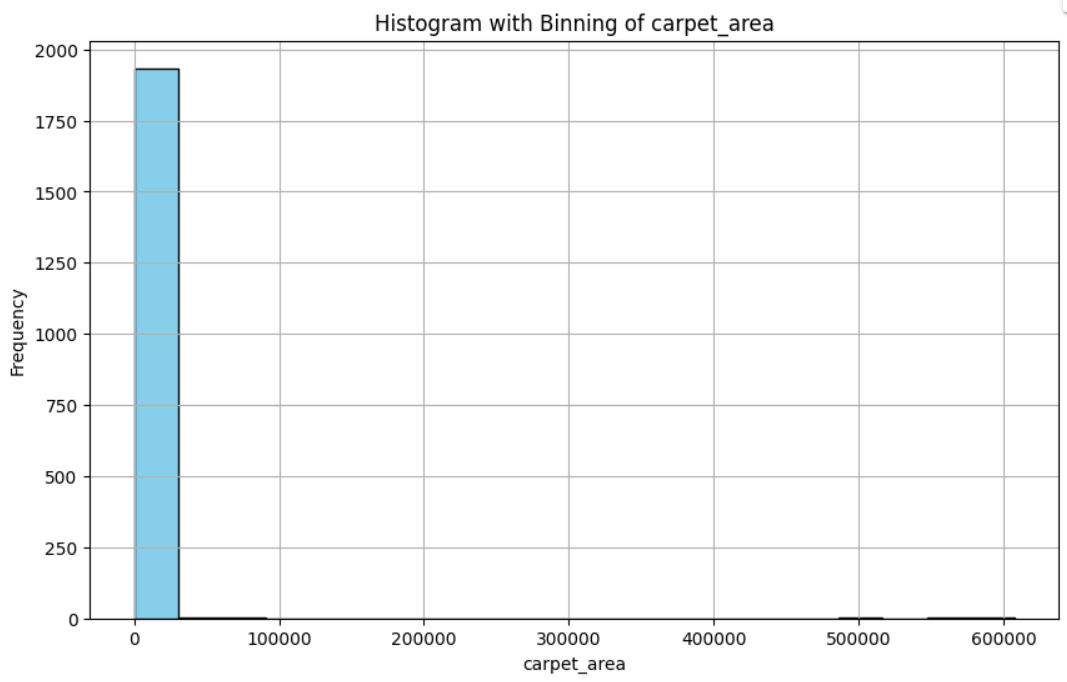
FOR THE CARPET\_AREA FEATURE:

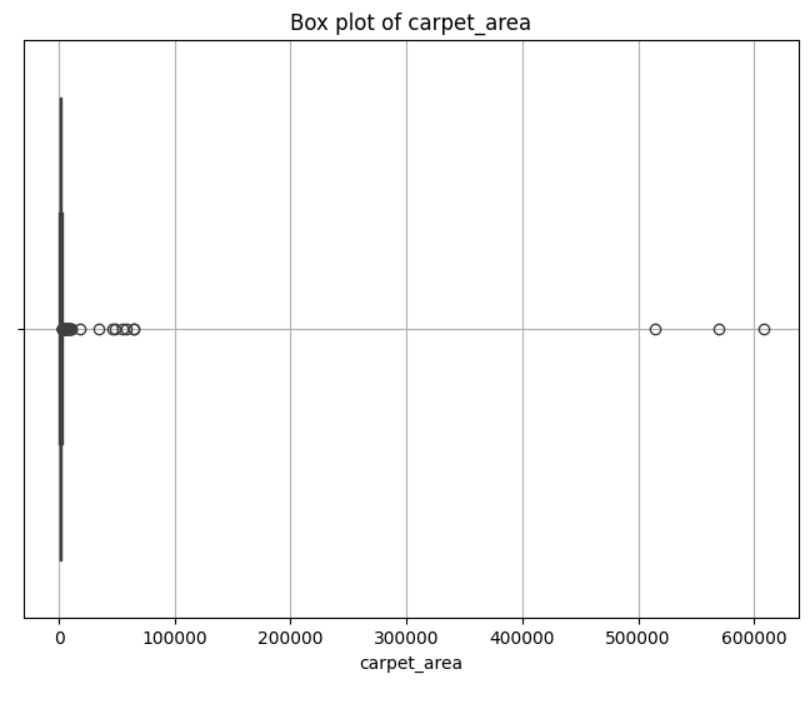
Analysis for column: carpet\_area

Skewness of 'carpet\_area': 24.796083599668638

Kurtosis of 'carpet\_area': 627.839357284124





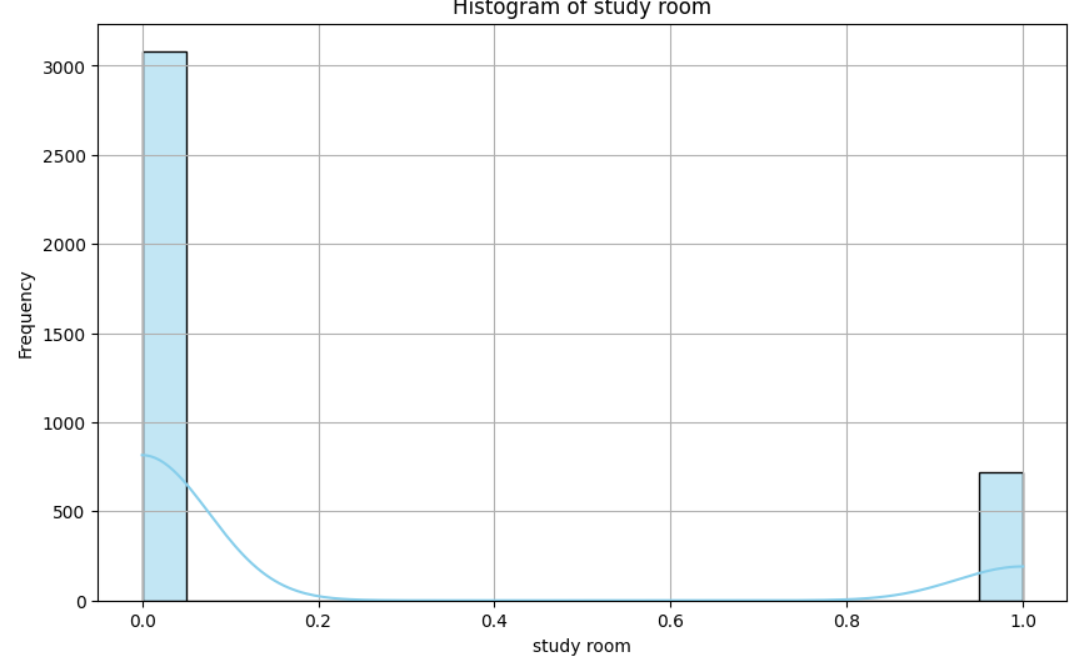


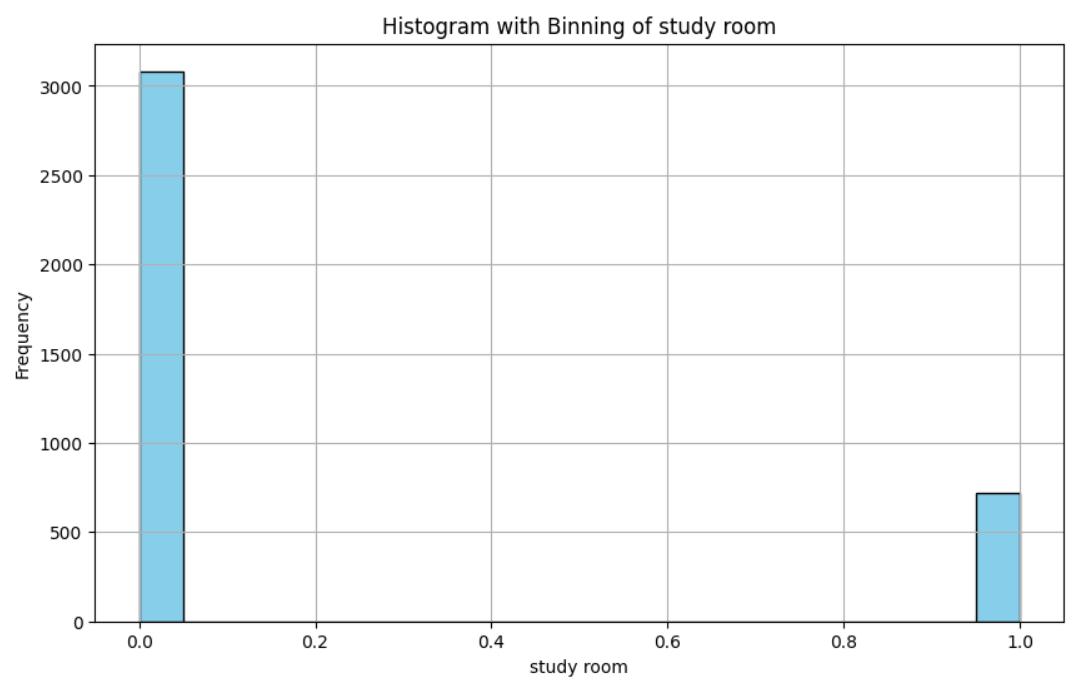
FOR THE STUDY ROOM FEATURE:

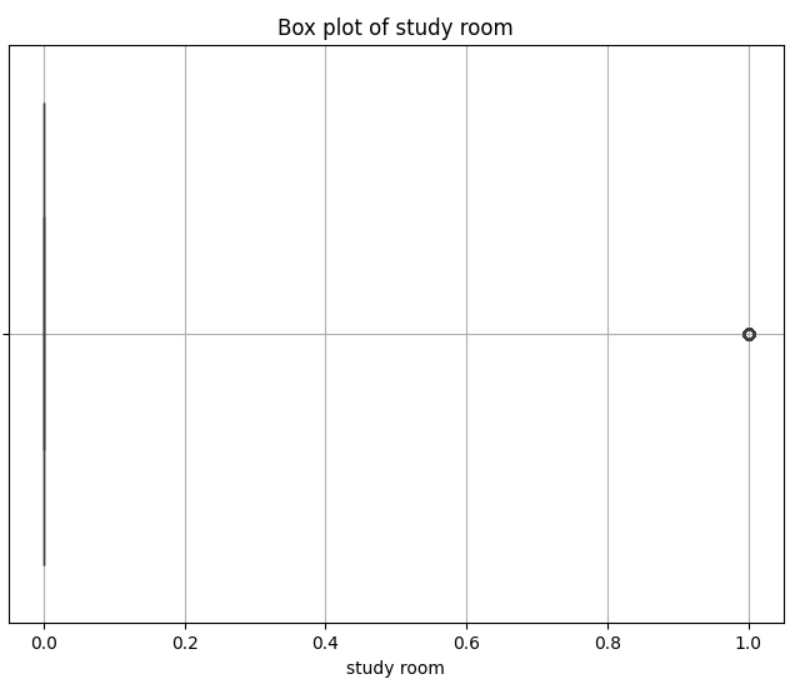
Analysis for column: study room

Skewness of 'study room': 1.5844676873884345

Kurtosis of 'study room': 0.5108062090652594





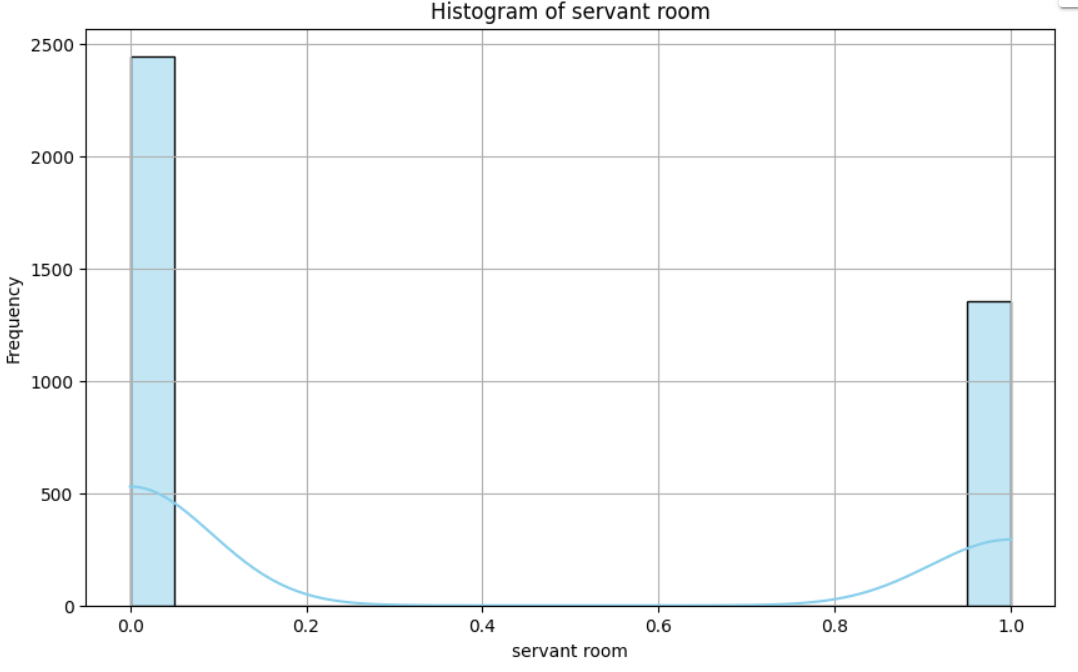
****

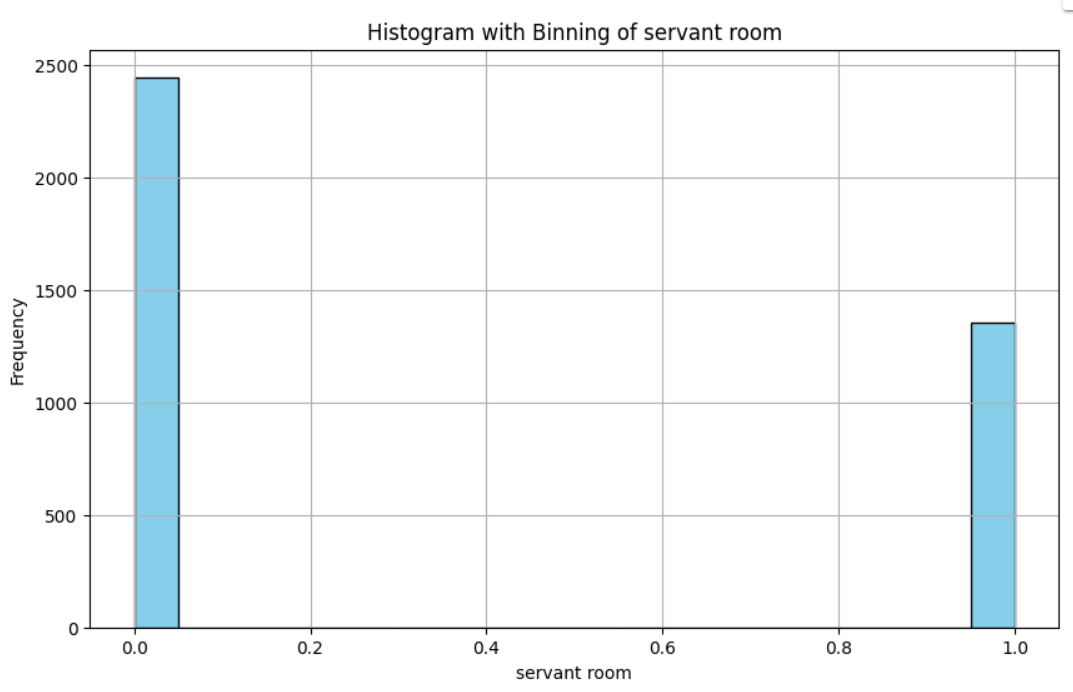
FOR THE SERVANT ROOM FEATURE:

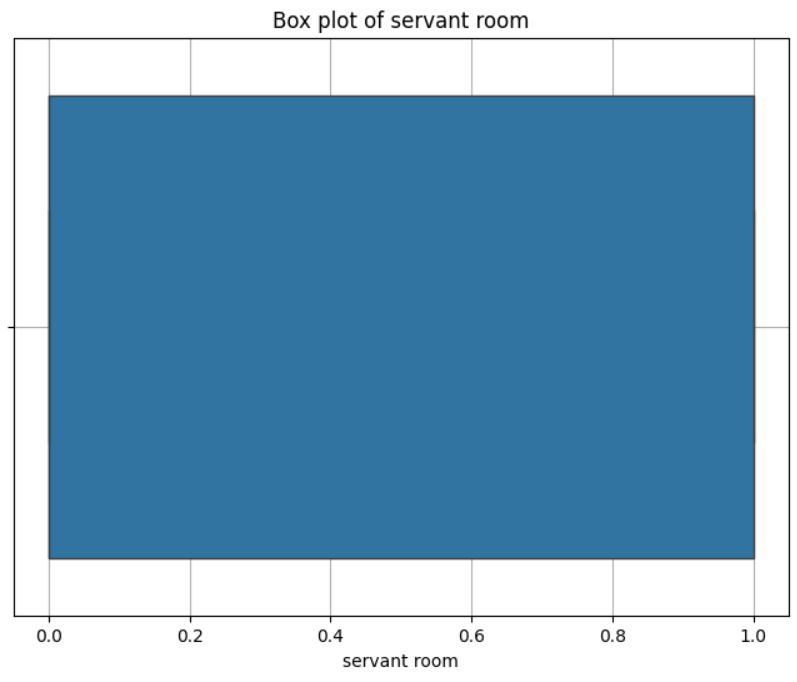
Analysis for column: servant room

Skewness of 'servant room': 0.597972317081455

Kurtosis of 'servant room': -1.6432935938628566





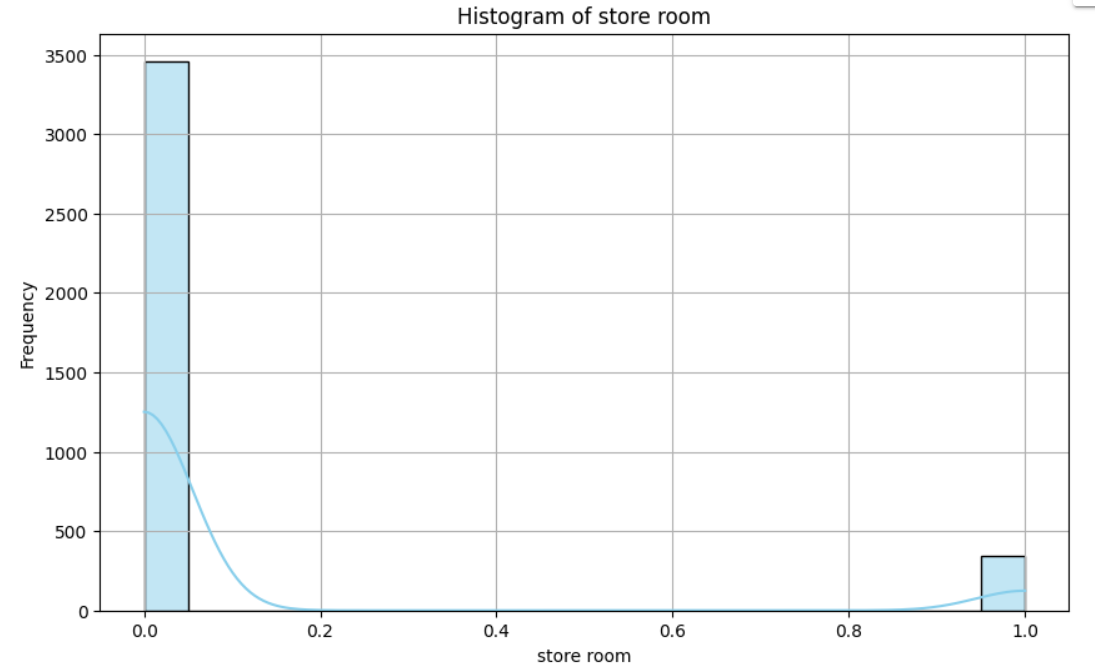


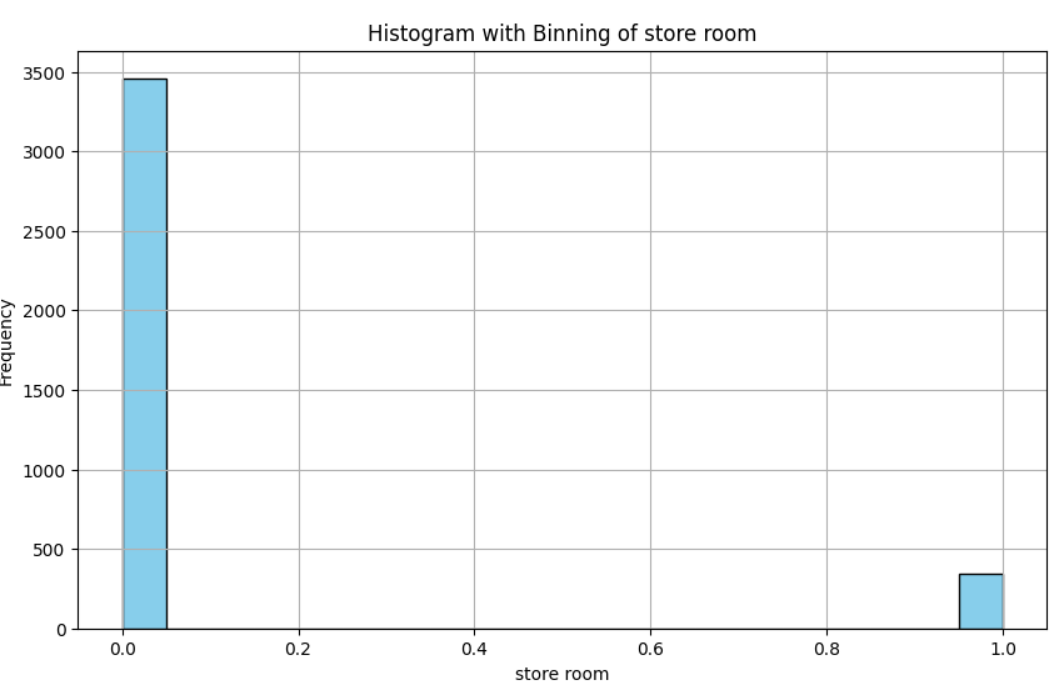
FOR THE STORE ROOM FEATURE:

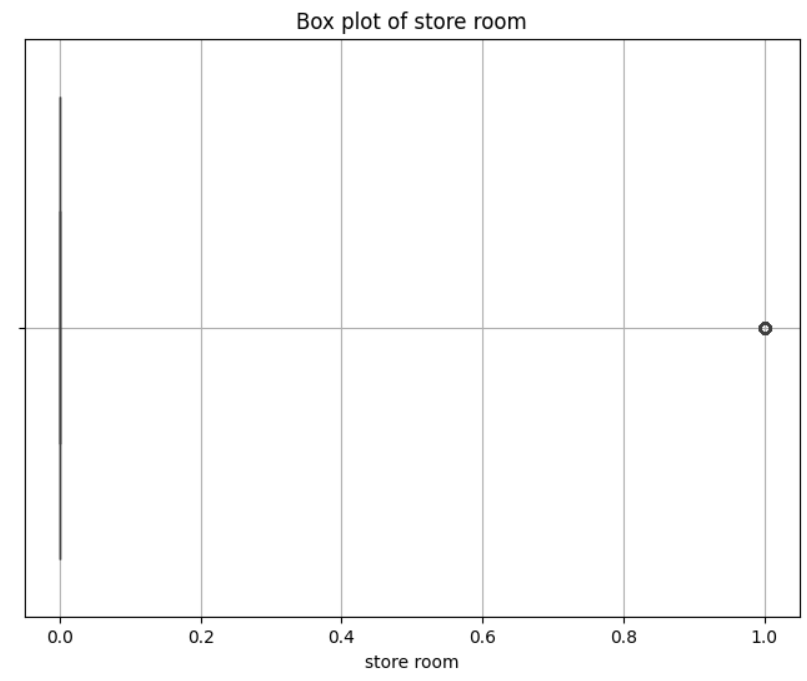
Analysis for column: store room

Skewness of 'store room': 2.856767510561787

Kurtosis of 'store room': 6.164362175001538





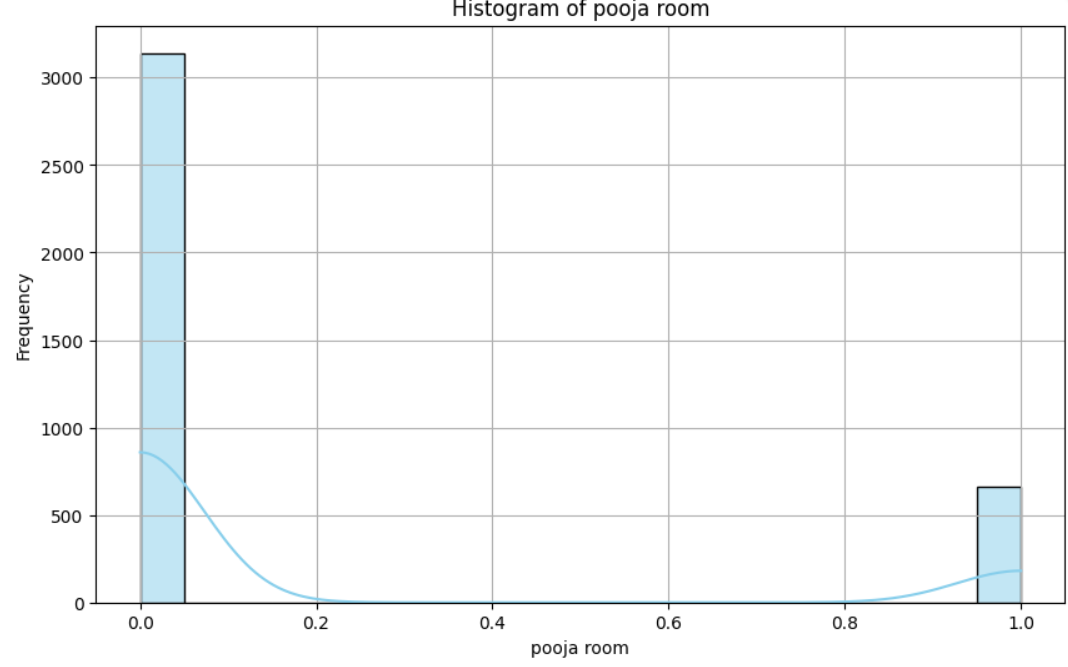


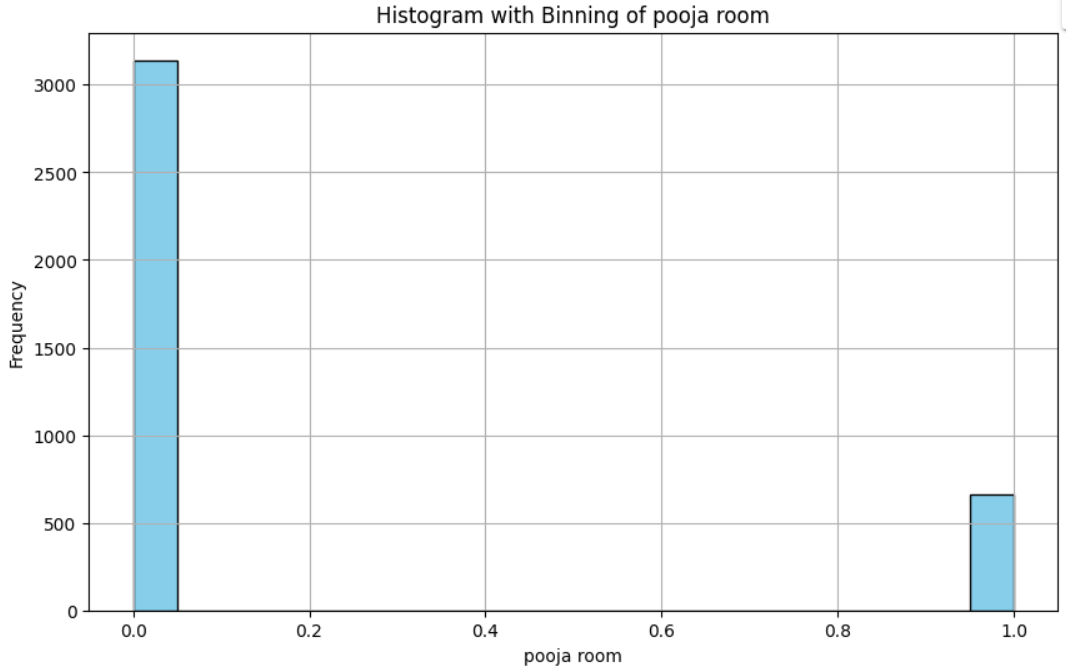
FOR THE POOJA ROOM FEATURE:

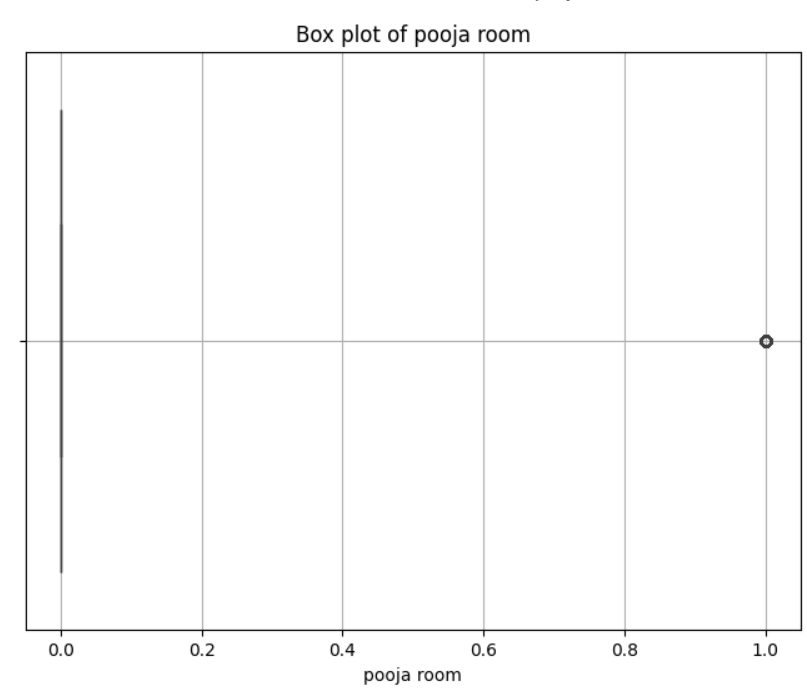
Analysis for column: pooja room

Skewness of 'pooja room': 1.7174170404503821

Kurtosis of 'pooja room': 0.9500206306784613





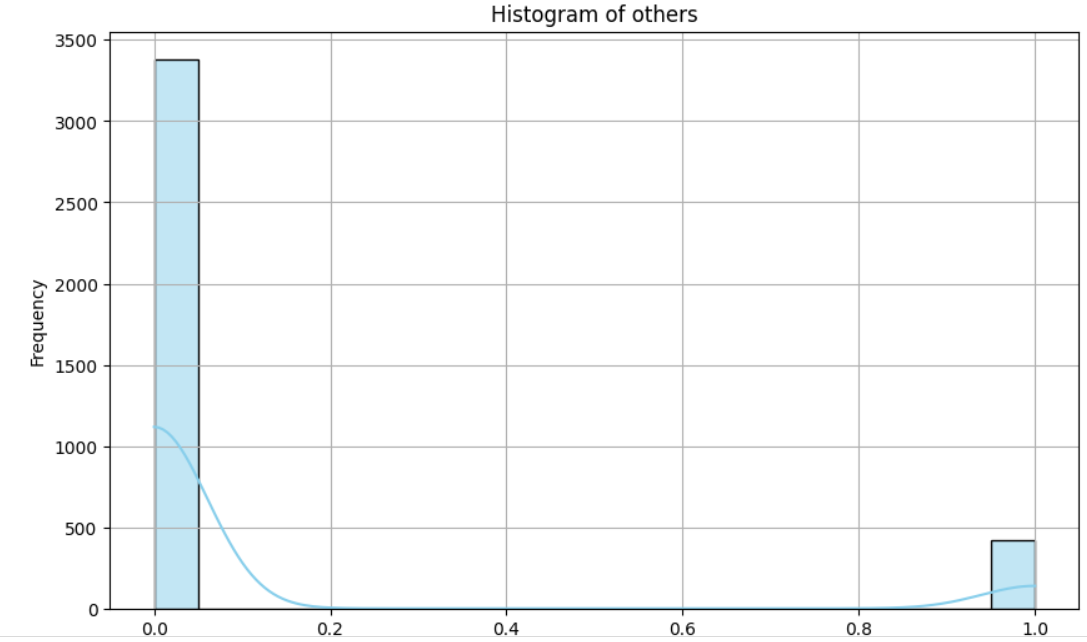


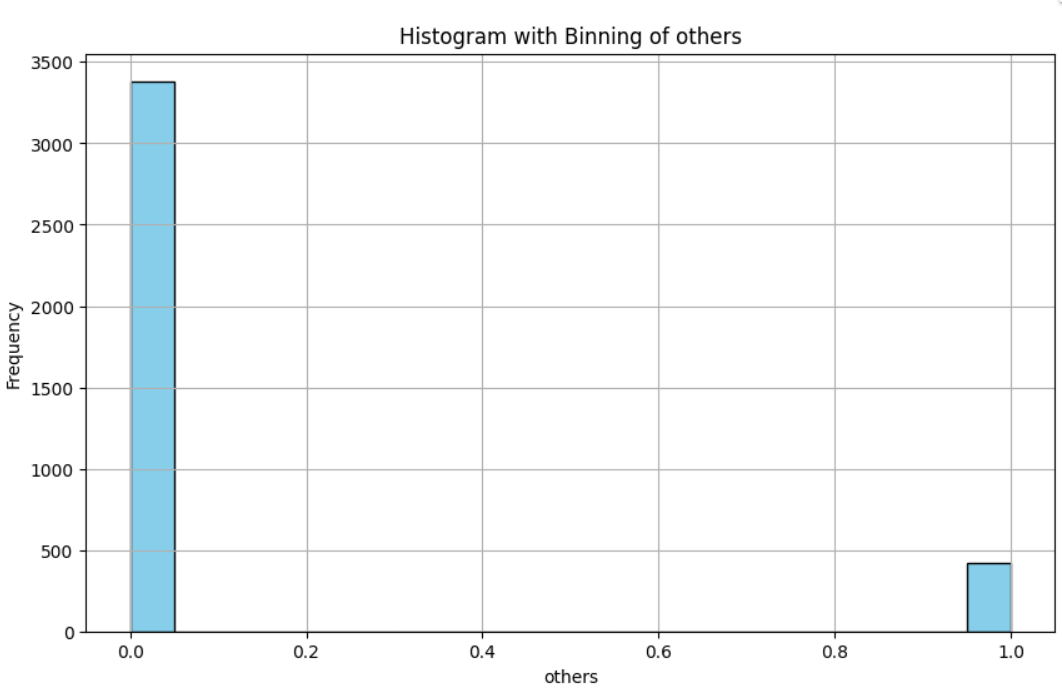
FOR THE OTHERS FEATURE:

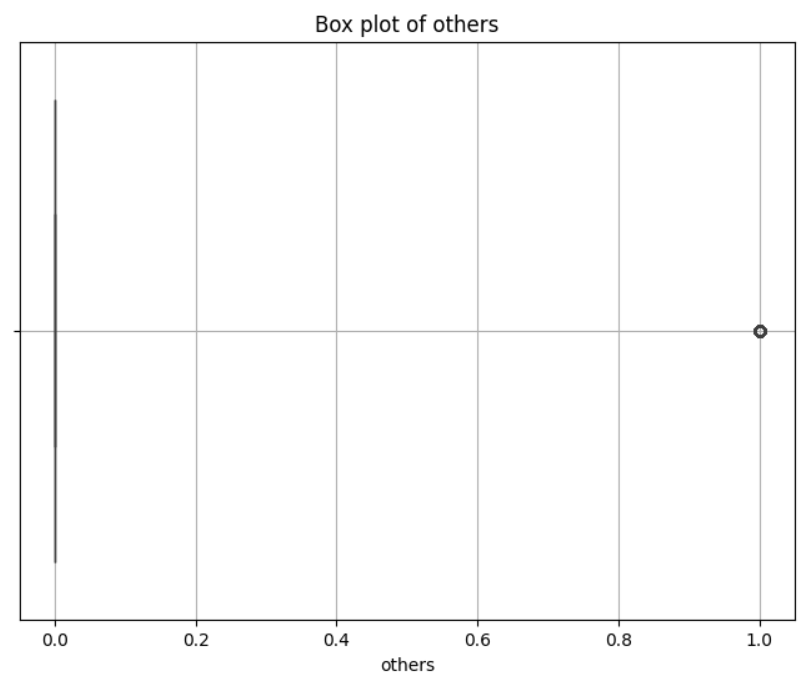
Analysis for column: others

Skewness of 'others': 2.482457986756426

Kurtosis of 'others': 4.164787644010293





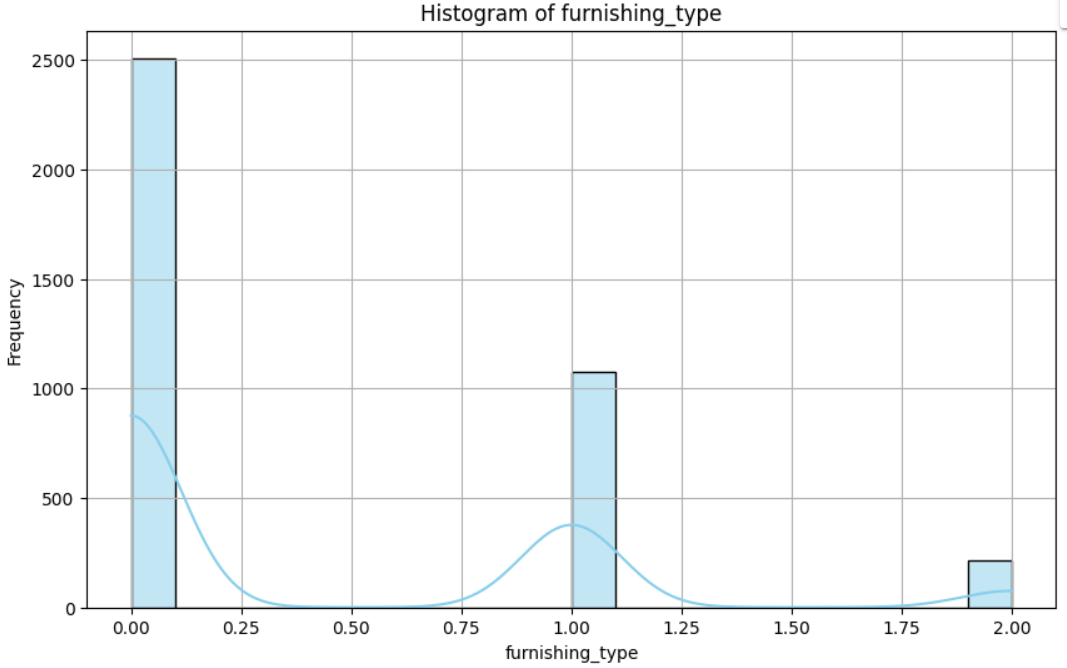


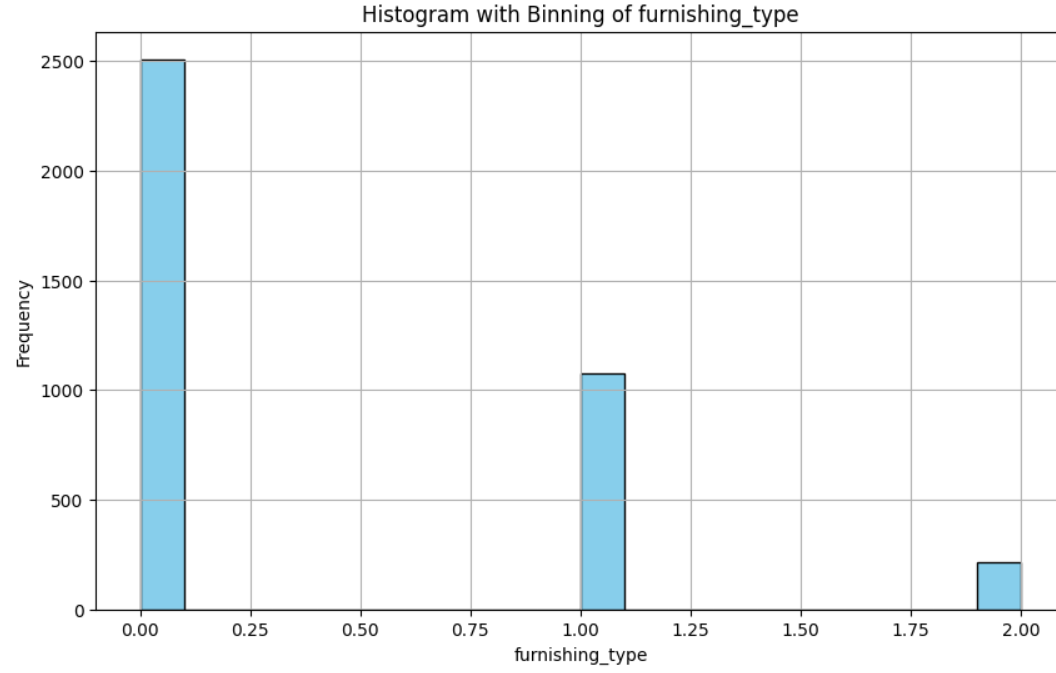
FOR THE furnishing\_type FEATURE:

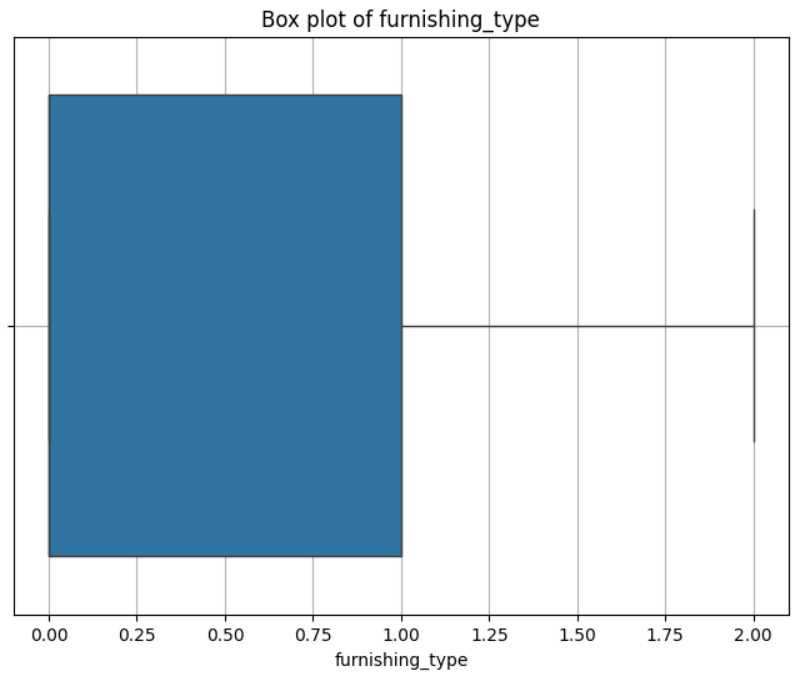
Analysis for column: furnishing\_type

Skewness of 'furnishing\_type': 1.2152167536922434

Kurtosis of 'furnishing\_type': 0.44366852193706885





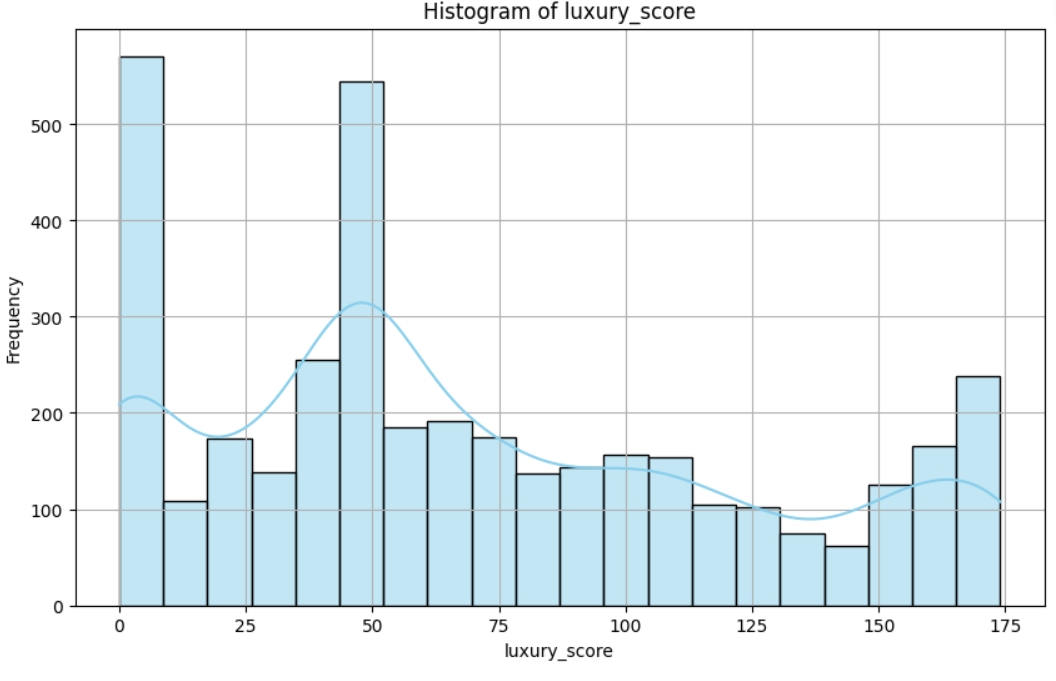


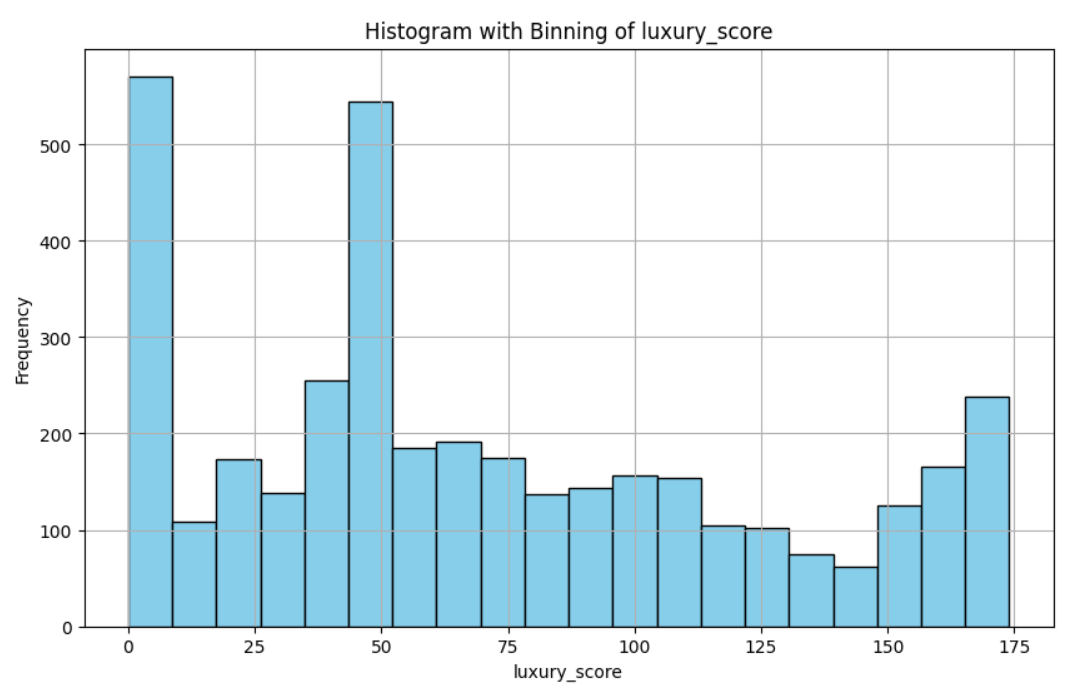
FOR THE luxury\_score FEATURE:

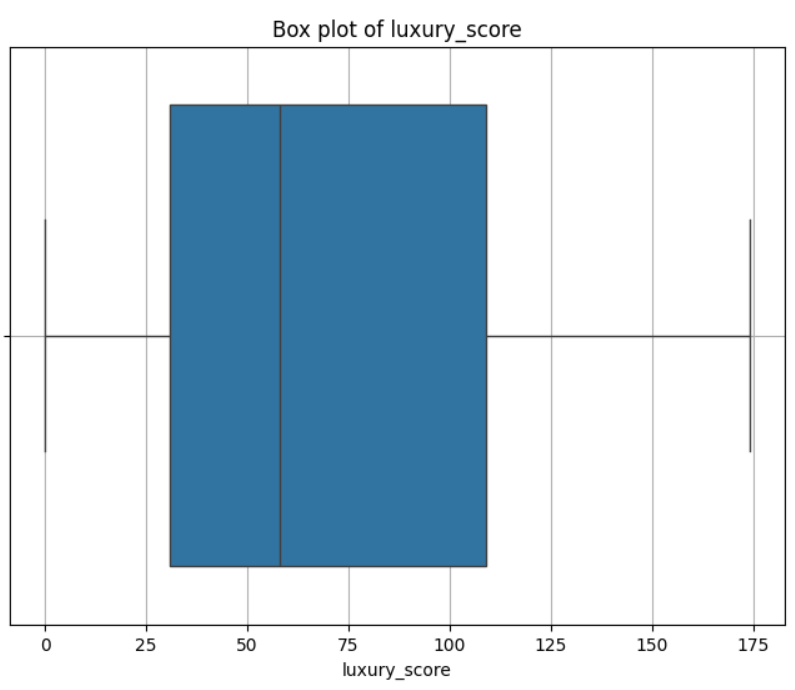
Analysis for column: luxury\_score

Skewness of 'luxury\_score': 0.47028839425636054

Kurtosis of 'luxury\_score': -0.8553365481063526







Similarly, we have done for other features

Next we have done multi variate analysis considering the feature v/s target feature like prize v/s property typer and draw histogram, box plot and scatter plot between prize and area.

# Function to perform multivariate analysis for categorical columns

def analyze\_categorical\_vs\_price(column\_name):

# Check if the column exists in the dataset

if column\_name not in df.columns:

print(f"Column '{column\_name}' not found in the dataset.")

return

# Box plot

plt.figure(figsize=(10, 6))

sns.boxplot(x=column\_name, y='price', data=df)

plt.title(f'Box plot of {column\_name} vs Price')

plt.xlabel(column\_name)

plt.ylabel('Price')

plt.grid(True)

plt.xticks(rotation=45)

plt.show()

# Histogram

plt.figure(figsize=(10, 6))

sns.histplot(data=df, x=column\_name, y='price', bins=20, cmap='coolwarm', cbar=True)

plt.title(f'Histogram of {column\_name} vs Price')

plt.xlabel(column\_name)

plt.ylabel('Price')

plt.grid(True)

plt.xticks(rotation=45)

plt.show()

# Function to perform scatter plot analysis for numerical columns

def analyze\_numerical\_vs\_price(column\_name):

# Scatter plot

plt.figure(figsize=(10, 6))

sns.scatterplot(x=column\_name, y='price', data=df)

plt.title(f'Scatter plot of {column\_name} vs Price')

plt.xlabel(column\_name)

plt.ylabel('Price')

plt.grid(True)

plt.show()

# Multivariate analysis for categorical columns

categorical\_columns = ['property\_type', 'availability']

for column in categorical\_columns:

print("\nMultivariate analysis for column:", column)

analyze\_categorical\_vs\_price(column)

# Multivariate analysis for numerical columns

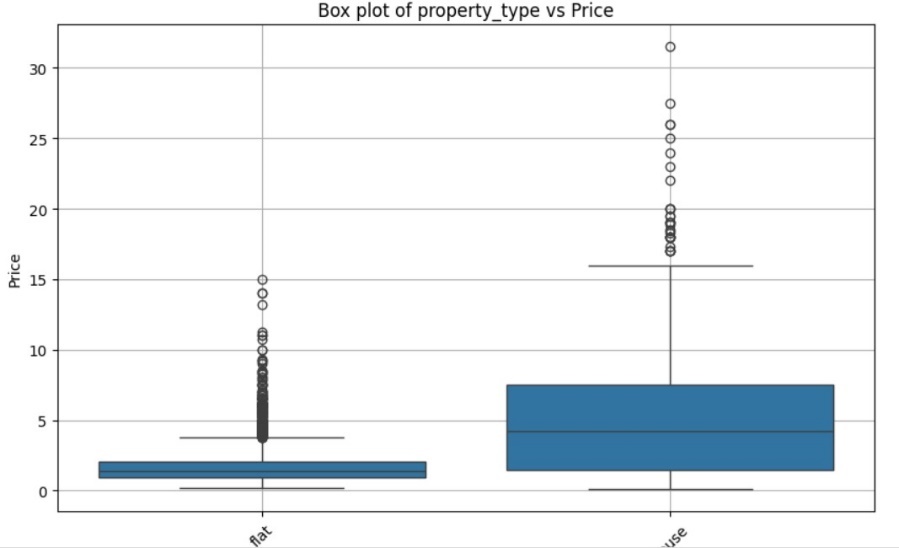
numerical\_columns = ['area']

for column in numerical\_columns:

print("\nMultivariate analysis for column:", column)

analyze\_numerical\_vs\_price(column)

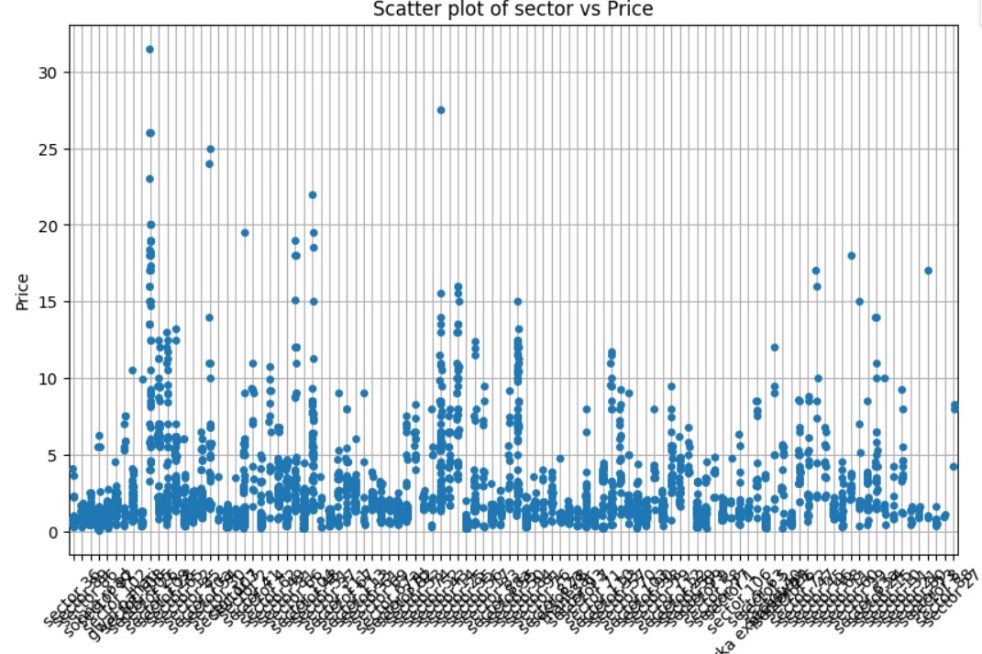
Multivariate analysis for column: Property\_type

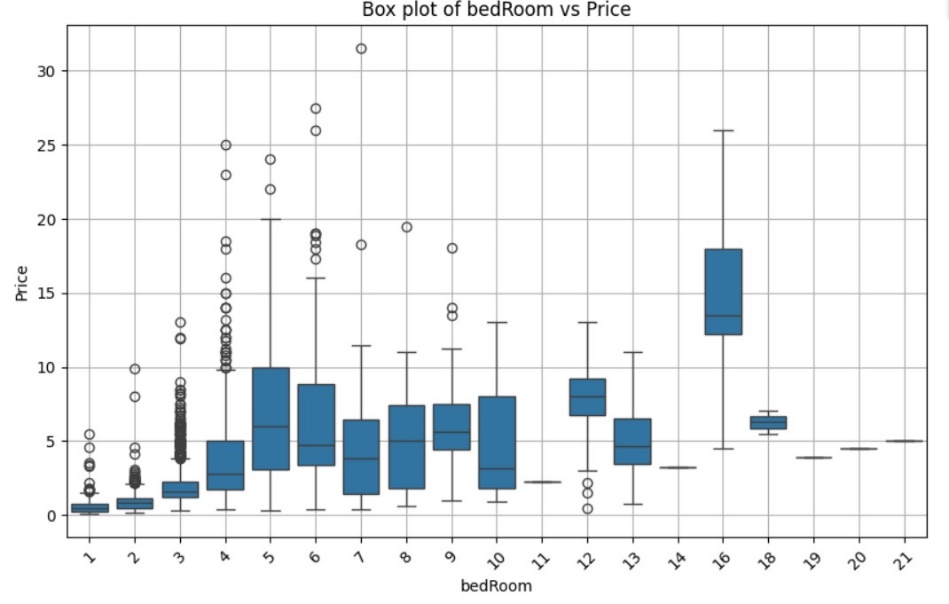




Multivariate analysis for categorical column: property\_type



Multivariate analysis for categorical column: sector

Multivariate analysis for numerical column: bedRoom 

Like this we have done multivariate analysis for different features.

We also done pie charts for the features.

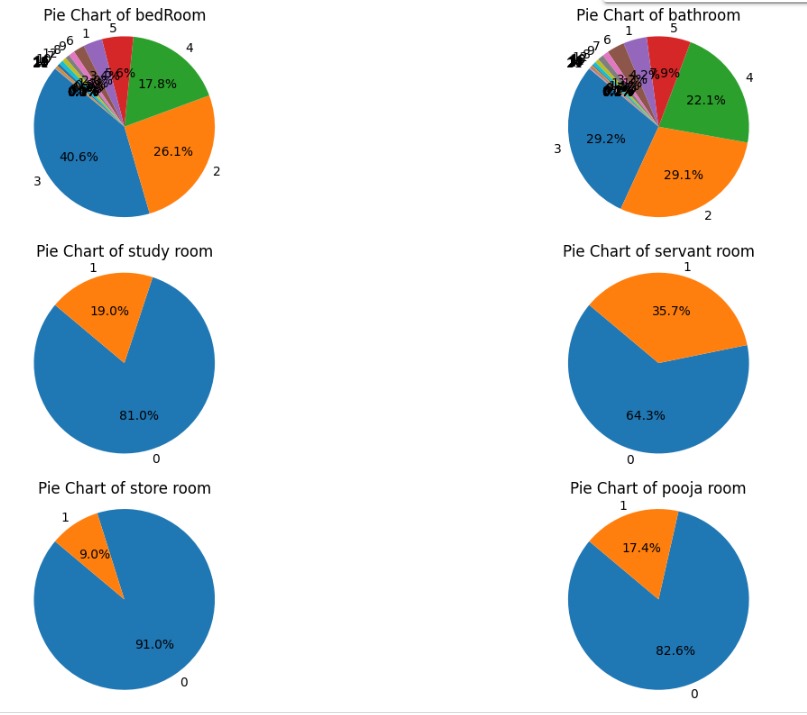
Here is a small snippet:

# Create a pie chart

ax.pie(category\_counts, labels=category\_counts.index, autopct='%1.1f%%', startangle=140)

ax.set\_title('Pie Chart of {}'.format(column))

ax.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.



Now, we checked for the missing values in the data set and determine their distributions across features.Here is the snippet:

def analyze\_missing\_values(df):

# Step 1: Check for missing values

missing\_values = df.isnull()

# Step 2: Summarize missing values

missing\_count = missing\_values.sum()

# Step 3: Visualize missing value distribution

plt.figure(figsize=(10, 6))

sns.heatmap(missing\_values, cmap='viridis', cbar=False)

plt.title('Missing Values Distribution Across Features')

plt.xlabel('Features')

plt.ylabel('Data Points')

plt.show()

# Bar plot for missing value count across features

plt.figure(figsize=(12, 6))

missing\_count.plot(kind='bar', color='skyblue')

plt.title('Missing Values Count Across Features')

plt.xlabel('Features')

plt.ylabel('Missing Values Count')

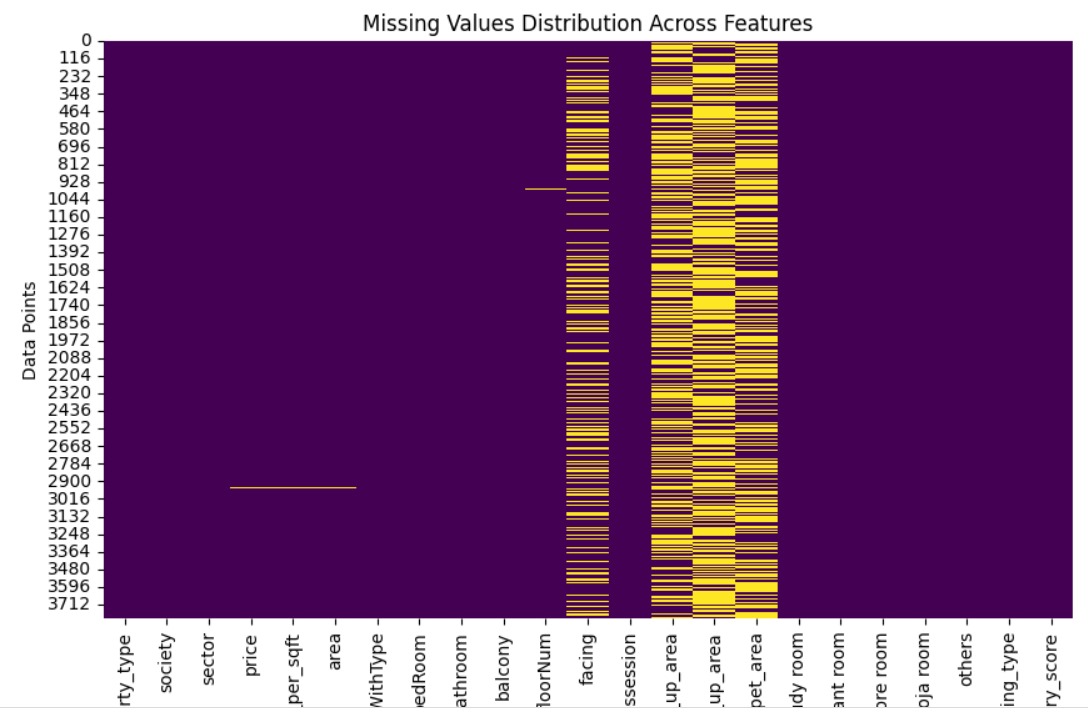
plt.xticks(rotation=45, ha='right')

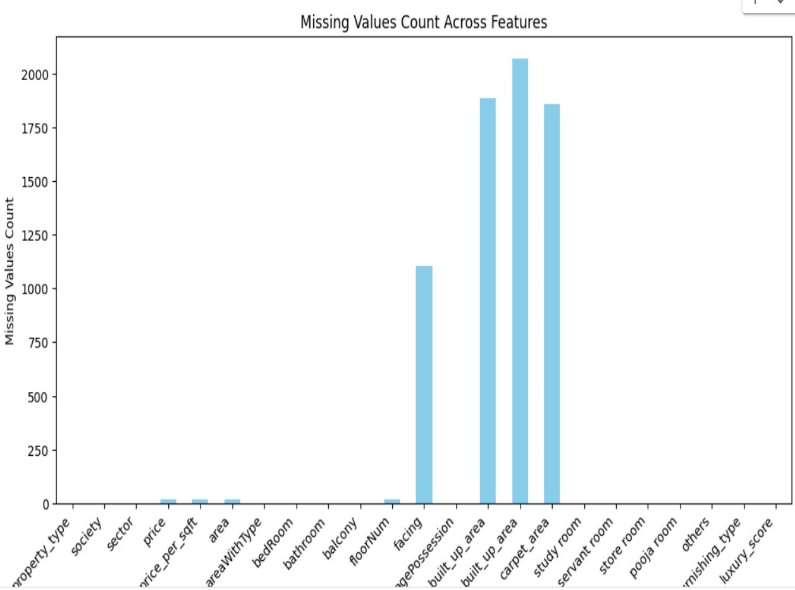
plt.show()

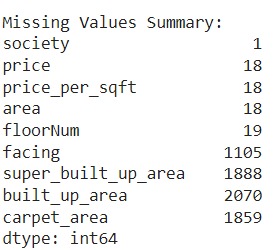
# Print summary

print("Missing Values Summary:")

print(missing\_count[missing\_count > 0])





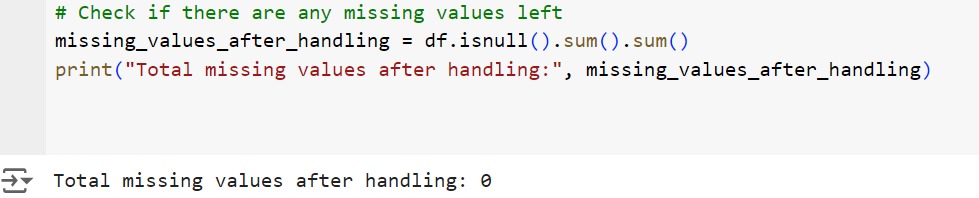


**TASK 2:**

Here we are using Random forest algorithm and applying simple imputer for handling missing values

In some places we have used median and mode but in many cases we have taken the maximum of the given feature and replaced near the missing values in the column

After using the RFM we have recheched our code if there are anu missing values and the verification is given below



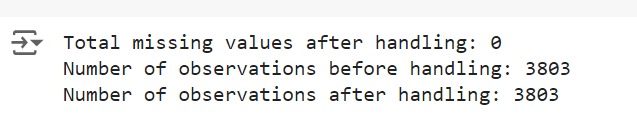
We have checked if there are any missing values in the all the given features

the given is the output after checking if there any missing values



we have handled all the missing values and printed a separate csv file for handled data

the number of observations before and after the handlin is not effected



Original data and handled after the given code

missing\_values\_after\_handling = handled\_data.isnull().sum().sum()

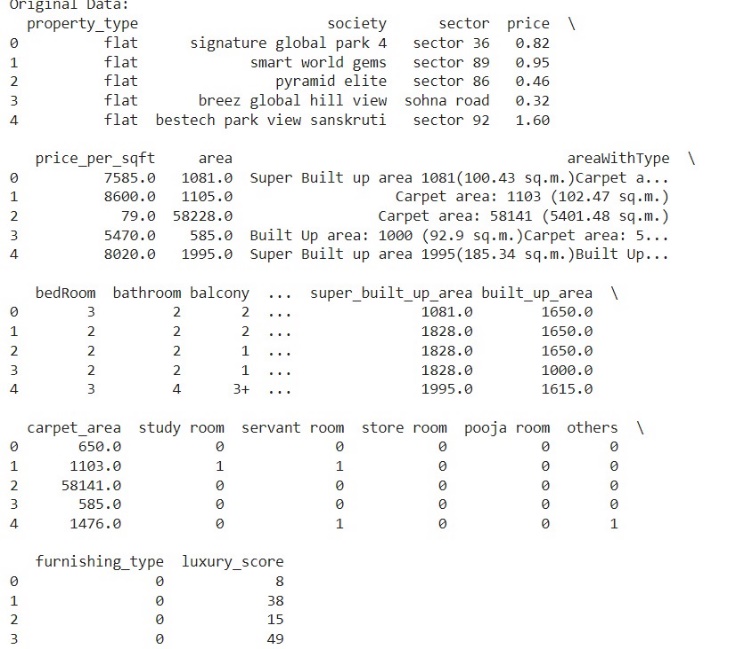
print("Total missing values after handling:", missing\_values\_after\_handling)

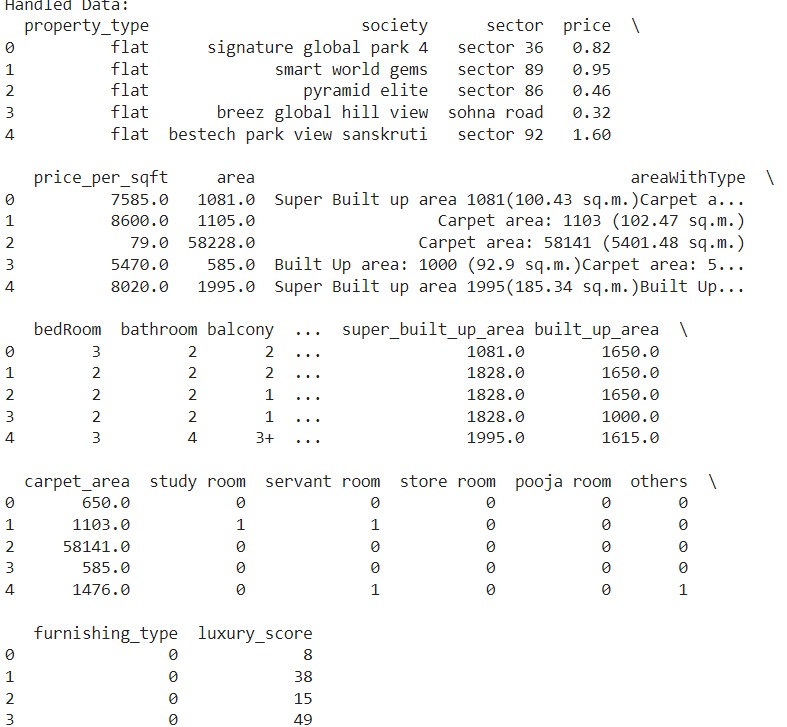
num\_observations\_before = len(df)

num\_observations\_after = len(handled\_data)

print("Number of observations before handling:", num\_observations\_before)

print("Number of observations after handling:", num\_observations\_after)





**TASK -3**

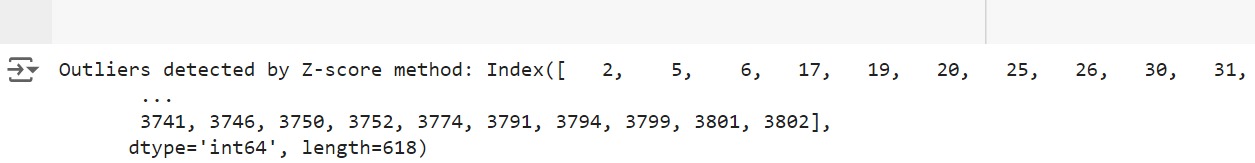
Out of all the given features we are selecting only the numerical columns

we are taking help of z-score for finding out the outliers the task for this code is

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



Plotting for the numerical features and visualising through various ways like histograms , box plots

CODE FOR HISTOGRAM:

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=df[col])

plt.title(col)

plt.tight\_layout()

plt.show()

CODE FOR BOX PLOT:

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=df[col])

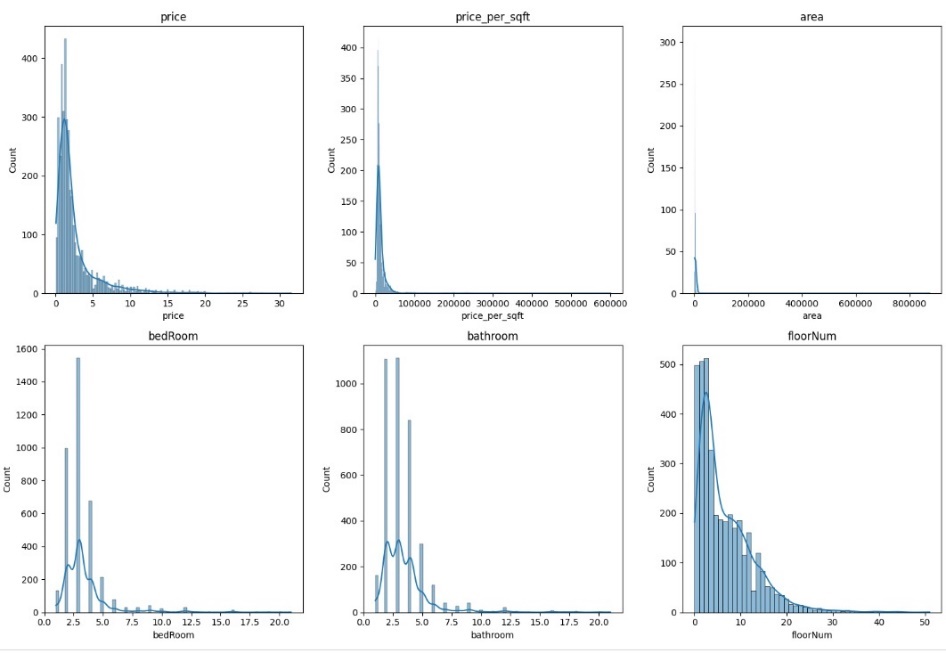
plt.title(col)

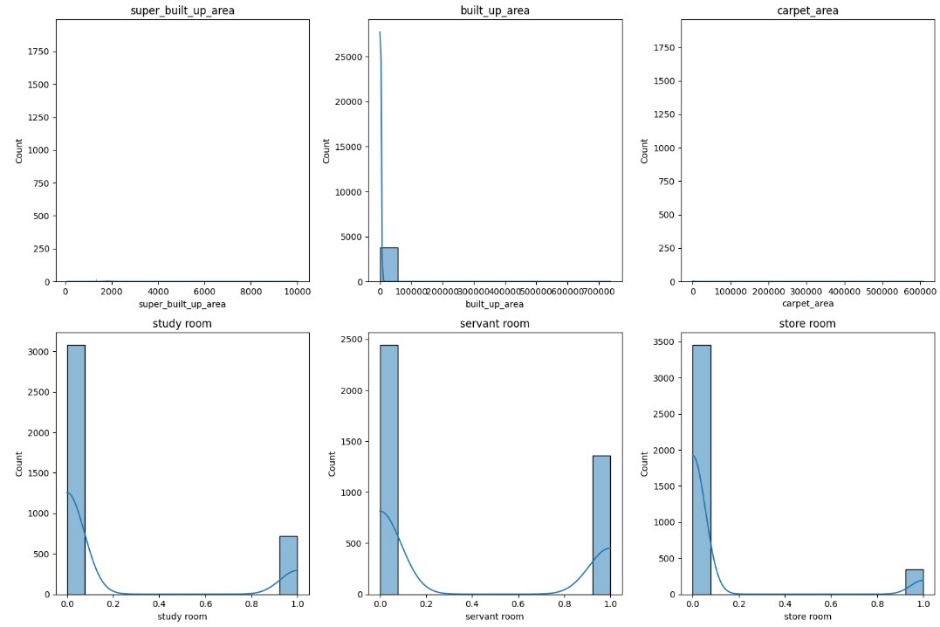
plt.tight\_layout()

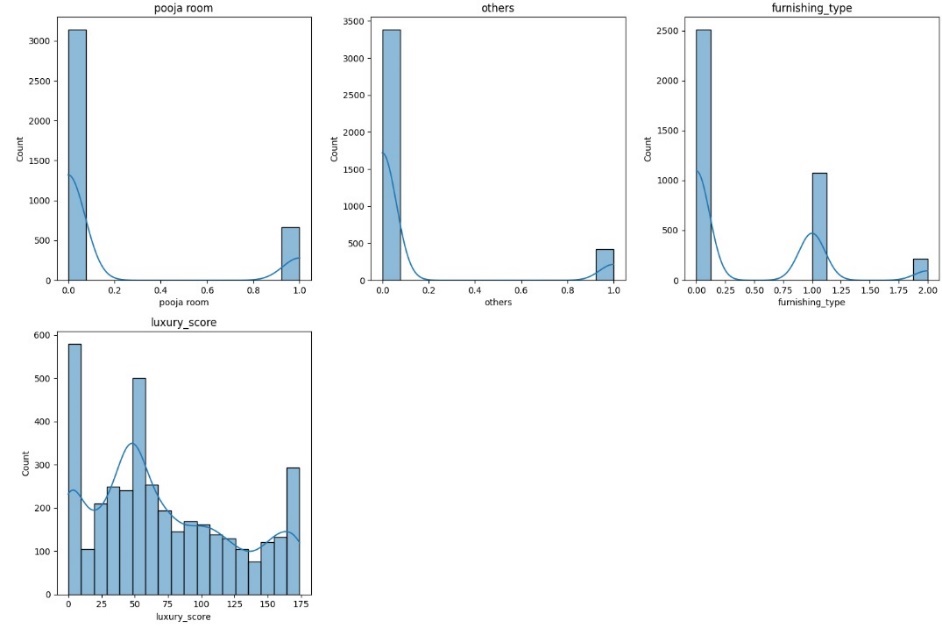
plt.show()

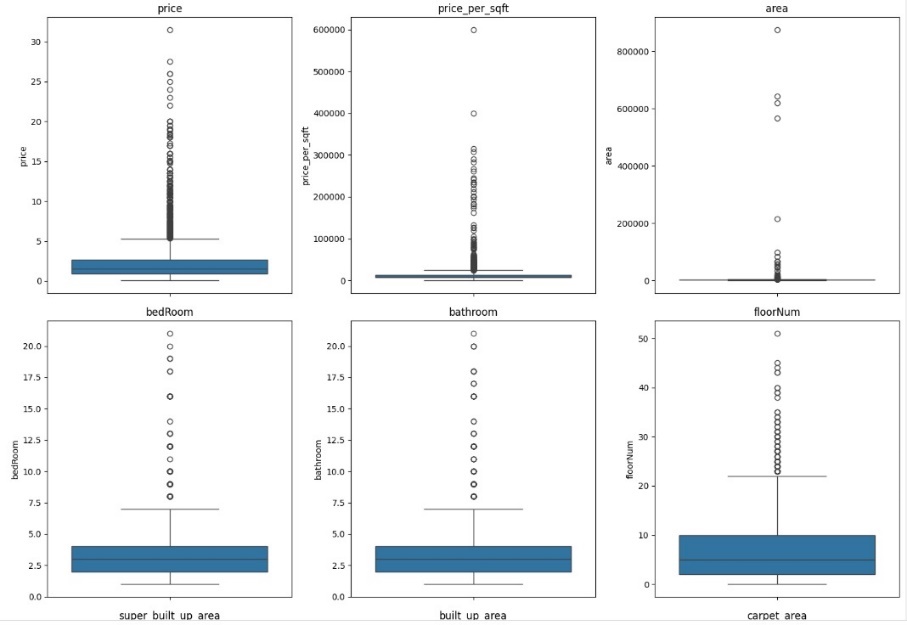
for getting mean , median mode and also other type of descriptive statistics we use

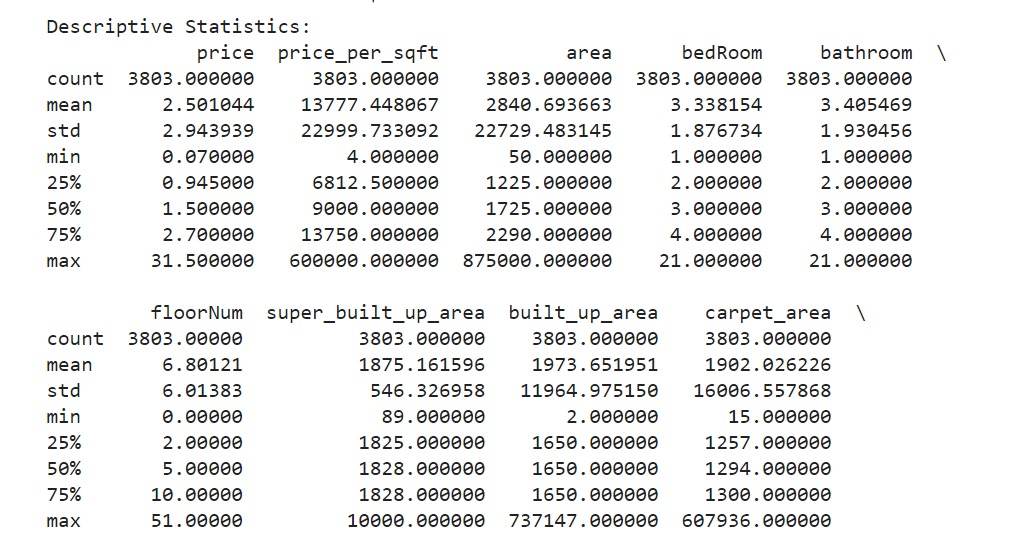
print(numerical\_features.describe())

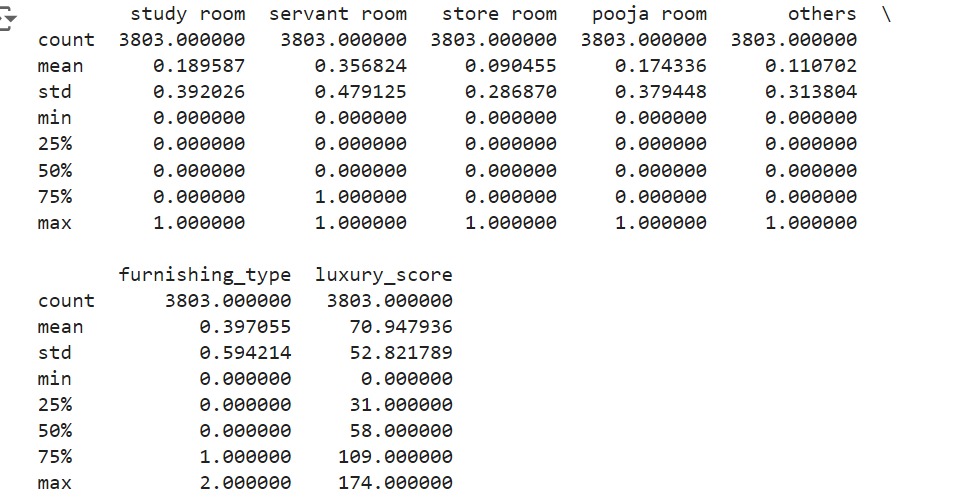












**TASK-4**

In task 3 whatever the outliers we have detected we should handle them in task 4

but there is issue that is some data is categorical and some numerical as code doesn't take categorical, we should convert them to numerical data

for this we use one hot encoding

the handled data and outliers are handled by random forest regressor

for using random forest regressor we use from

sklearn.ensemble import RandomForestRegressor

We are splitting the data into train and test sets and finding the outliers

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

for checking the model performance we use the below code

# Define a function to evaluate model performance

def evaluate\_model(model, X\_test, y\_test):

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

return mse

We are using trimming technique for handling outliers

y\_test\_trimmed = y\_test[X\_test\_trimmed.index]

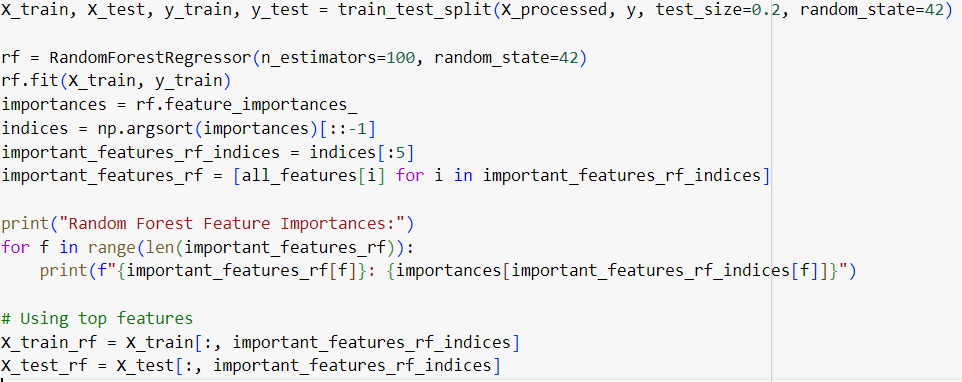
MSE without outlier handling and MSE with trimming

comparing the model performances

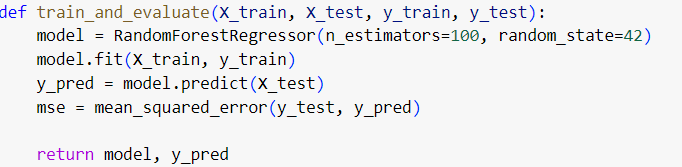


**BROWNIE POINT:**

Using random forest algo we are taking the top main features(for eg: price per square feet, area, built up area) and we are predicting the price of the property. As random forests algorithm splits the data into train and test by taking the main features.



Predicting actual property prices vs predicting prices



Here is the output of the random forest using above algorithm:

