Capstone Project (EDA)

Univariate Analysis

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

```
df = pd.read_csv('gurgaon_properties_cleaned_v2.csv')
```

```
df.shape

Output: (3803, 23)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3803 entries, 0 to 3802
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   property_type       3803 non-null   object
 1   society             3802 non-null   object
 2   sector              3803 non-null   object
 3   price               3785 non-null   float64
 4   price_per_sqft      3785 non-null   float64
 5   area                3785 non-null   float64
 6   areaWithType        3803 non-null   object
 7   bedRoom             3803 non-null   int64
 8   bathroom            3803 non-null   int64
 9   balcony             3803 non-null   object
 10  floorNum            3784 non-null   float64
 11  facing              2698 non-null   object
 12  agePossession       3803 non-null   object
 13  super_built_up_area 1915 non-null   float64
 14  built_up_area       1733 non-null   float64
 15  carpet_area         1944 non-null   float64
 16  study room          3803 non-null   int64
 17  servant room        3803 non-null   int64
 18  store room          3803 non-null   int64
 19  pooja room          3803 non-null   int64
...
 21  furnishing_type     3803 non-null   int64
 22  luxury_score        3803 non-null   int64
dtypes: float64(7), int64(9), object(7)
memory usage: 683.5+ KB
```

Check & Drop Duplicates:

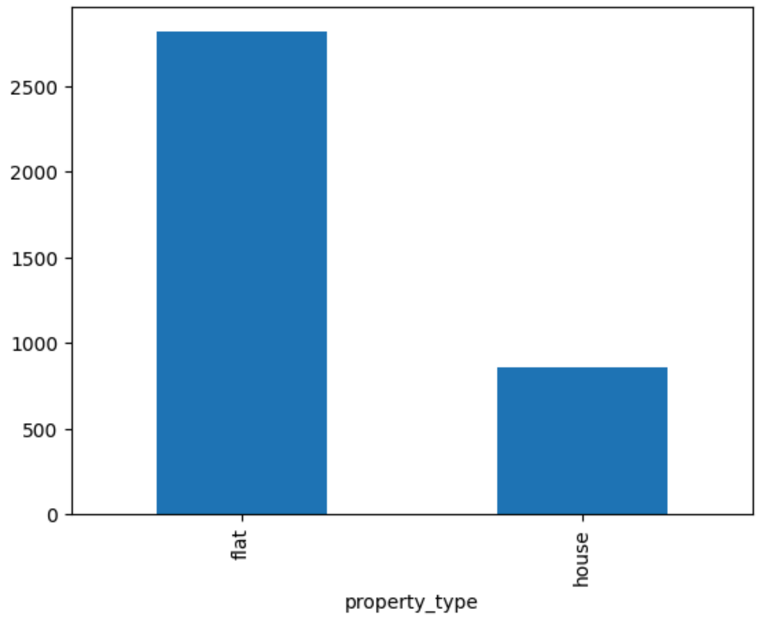
```
df.duplicated().sum()

Output: 126
```

Drop duplicates:

```
df.drop_duplicates(inplace=True)
```

property_type

```
df['property_type'].value_counts().plot(kind='bar',)
```



Observations:

- Flats are in majority(75 percent) and there are less number of houses(~25 percent)
- No missing values

society

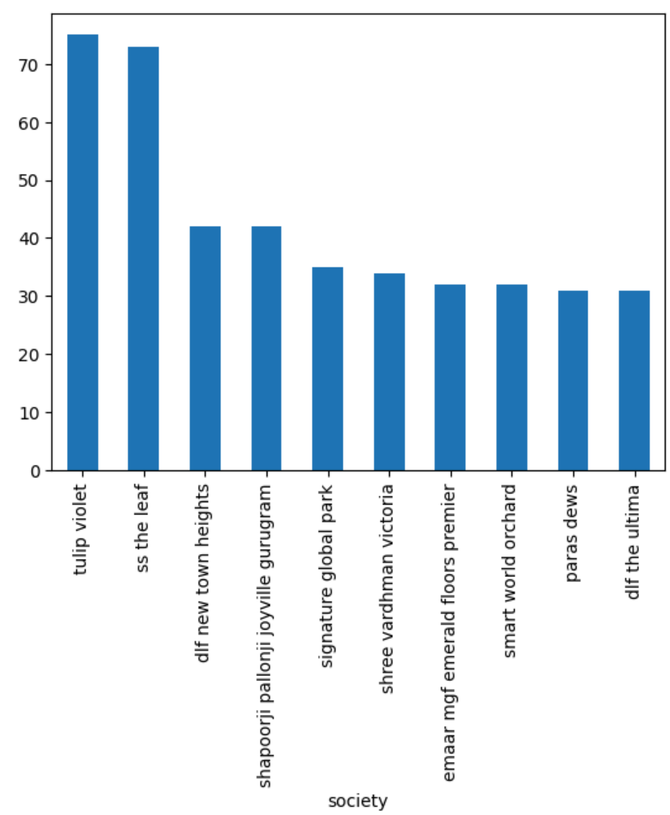
```
df['society'].value_counts().shape

Outout: (676,)
```

```
df['society'].value_counts()
```

```
society
independent                          486
tulip violet                          75
ss the leaf                           73
dlf new town heights                  42
shapoorji pallonji joyville gurugram  42
                                    ...
micasa sec 68                          1
espire south                           1
adani brahma samsara                   1
smart world one dxp                    1
woodstock floors                       1
Name: count, Length: 676, dtype: int64
```

Top 10 societies:

```
df[df['society'] != 'independent']['society'].value_counts().head(10).plot(kind
='bar')
```



Observations:

- Around 13% properties comes under independent tag.
- There are 675 societies.
- The top 75 societies have 50 percent of the properties and the rest 50 percent of the properties come under the remaining 600 societies
    - Very High (>100): Only 1 society has more than 100 listings.
    - High (50-100): 2 societies have between 50 to 100 listings.
    - Average (10-49): 92 societies fall in this range with 10 to 49 listings each.
    - Low (2-9): 273 societies have between 2 to 9 listings.
    - Very Low (1): A significant number, 308 societies, have only 1 listing.
- 1 missing value

Sector

```
# Frequency distribution for sectors
sector_counts = df['sector'].value_counts()

sector_frequency_bins = {
    "Very High (>100)": (sector_counts > 100).sum(),
    "High (50-100)": ((sector_counts >= 50) & (sector_counts <= 100)).sum(),
    "Average (10-49)": ((sector_counts >= 10) & (sector_counts < 50)).sum(),
    "Low (2-9)": ((sector_counts > 1) & (sector_counts < 10)).sum(),
    "Very Low (1)": (sector_counts == 1).sum()
}

sector_frequency_bins
```

```
{'Very High (>100)': 3,
 'High (50-100)': 24,
 'Average (10-49)': 64,
 'Low (2-9)': 23,
 'Very Low (1)': 1}
```

Observations:

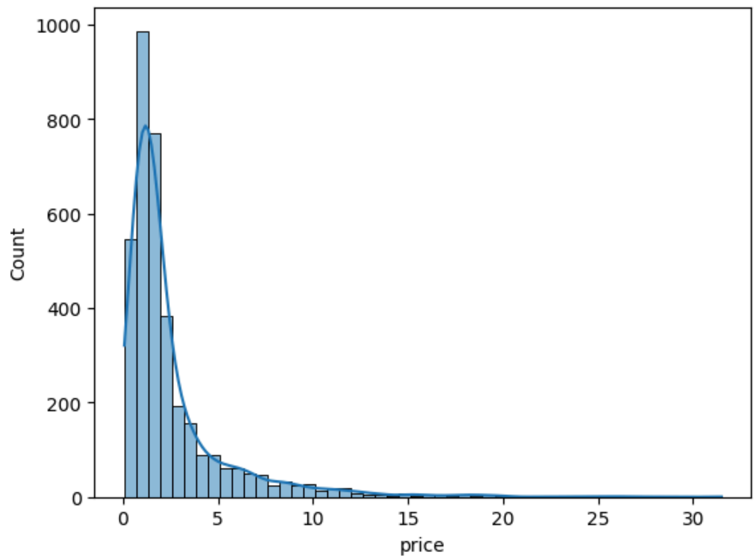
- There are a total of 104 unique sectors in the dataset.
- Frequency distribution of sectors:
    - Very High (>100): 3 sectors have more than 100 listings.
    - High (50-100): 25 sectors have between 50 to 100 listings.
    - Average (10-49): A majority, 60 sectors, fall in this range with 10 to 49 listings each.
    - Low (2-9): 16 sectors have between 2 to 9 listings.
    - Very Low (1): Interestingly, there are no sectors with only 1 listing.

Price (Output Column)

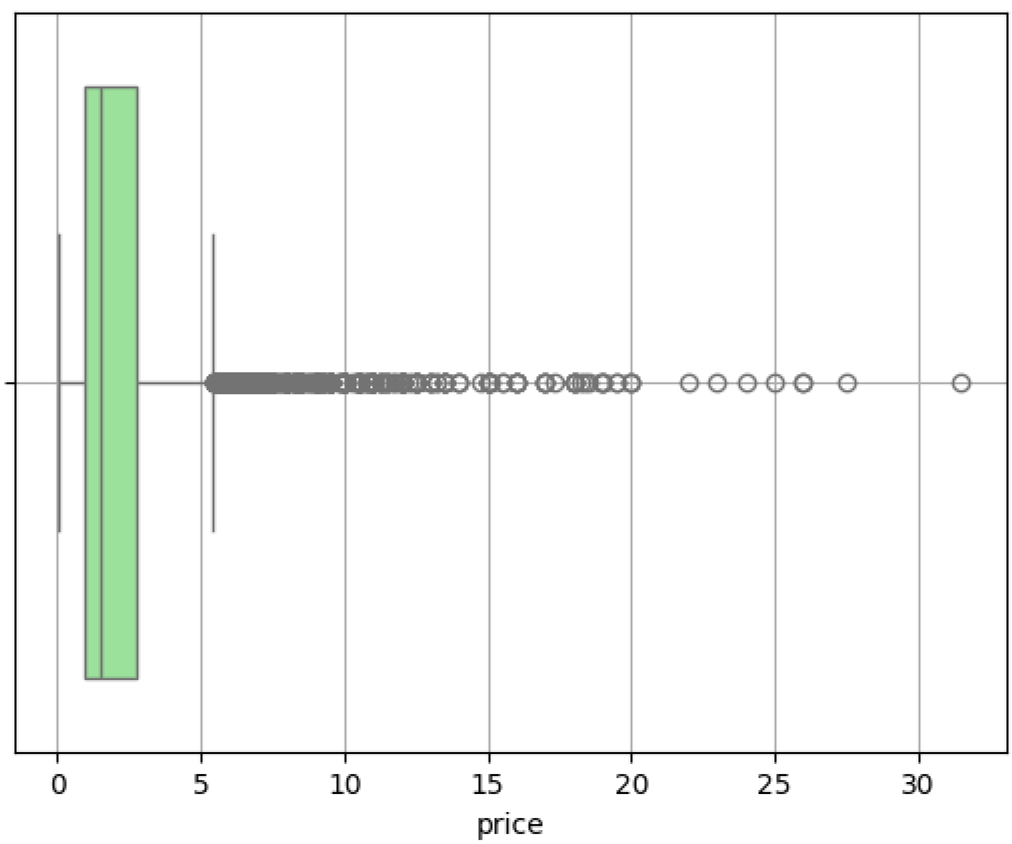
```
df['price'].describe()
```

```
count    3660.000000
mean        2.533664
std         2.980623
min         0.070000
25%         0.950000
50%         1.520000
75%         2.750000
max        31.500000
Name: price, dtype: float64
```

```
sns.histplot(df['price'], kde=True, bins=50)
```




```
sns.boxplot(x=df['price'], color='lightgreen')
plt.grid()
```



Observations:

- **Descriptive Statistics:**
  - **Count**: There are 3,660 non-missing price entries.
  - **Mean Price**: The average price is approximately 2.53 crores.
  - **Median Price**: The median (or 50th percentile) price is 1.52 crores.
  - **Standard Deviation**: The prices have a standard deviation of 2.98, indicating variability in the prices.
  - **Range**: Prices range from a minimum of 0.07 crores to a maximum of 31.5 crores.
  - **IQR**: The interquartile range (difference between 75th and 25th percentile) is from 0.95 crores to 2.75 crores.
- **Visualizations:**
  - **Distribution**: The histogram indicates that most properties are priced in the lower range (below 5 crores), with a few properties going beyond 10 crores.
  - **Box Plot**: The box plot showcases the spread of the data and potential outliers. Properties priced above approximately 10 crores might be considered outliers as they lie beyond the upper whisker of the box plot.
- **Missing Values**: There are 17 missing values in the price column.

Skewness and Kurtosis:

```
# Skewness and Kurtosis
skewness = df['price'].skew()
kurtosis = df['price'].kurt()

print(skewness,kurtosis)

Output: 3.2791704733134623 14.933372629214258
```

- **Kurtosis** measures whether a dataset has **heavy or light tails** compared to a normal distribution. It tells how much of the data is concentrated in the tails.
- It tells **how extreme the outliers** are in a probability distribution

📌**If result is ~0, it's mesokurtic (normal-like).**

📌**If result is positive, it's leptokurtic (heavy tails).**

📌**If result is negative, it's platykurtic (light tails)**

Observations:

Skewness: The price distribution has a skewness of approximately 3.28, indicating a positive skew. This means that the distribution tail is skewed to the right, which aligns with our observation from the histogram where most properties have prices on the lower end with a few high-priced properties.

Kurtosis: The kurtosis value is approximately 14.93. A kurtosis value greater than 3 indicates a distribution with heavier tails and more outliers compared to a normal distribution.

Quantile Analysis:

```
# Quantile Analysis
quantiles = df['price'].quantile([0.01, 0.05, 0.95, 0.99])

quantiles
```

```
0.01     0.250
0.05     0.370
0.95     8.500
0.99    15.264
Name: price, dtype: float64
```

Observations:

Quantile Analysis:

- 1% Quantile: Only 1% of properties are priced below 0.25 crores.
- 5% Quantile: 5% of properties are priced below 0.37 crores.
- 95% Quantile: 95% of properties are priced below 8.5 crores.
- 99% Quantile: 99% of properties are priced below 15.26 crores, indicating that very few properties are priced above this value.

Identify potential outliers using IQR method:

```
# Identify potential outliers using IQR method
Q1 = df['price'].describe()['25%']
Q3 = df['price'].describe()['75%']
IQR = Q3 - Q1

IQR

Output: 1.8
```

```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(lower_bound, upper_bound)

Output:-1.7500000000000002 5.45
```

```
outliers = df[(df['price'] < lower_bound) | (df['price'] > upper_bound)]
outliers.shape

Output: (425, 23)
```

```
outliers.sample(5)
```

	property_type	society	sector	price	price_per_sqft	area	areaWithTy
2856	house	independent	sector 25	16.00	26667.0	6000.0	Built Up are 6000 (557. sq.n
2899	house	international city by sobha phase 1	sector 109	5.70	10556.0	5400.0	Plot ar 600(501. sq.n
3438	flat	ambience caitriona	sector 24	14.00	200000.0	700.0	Built Up are 700 (65. sq.n
2578	house	independent	sector 26	18.25	18250.0	10000.0	Plot ar 550(51 sq.m.)Carp area: 100 sc
872	house	independent	sector 26	15.00	33200.0	4518.0	Plot ar 502(419. sq.n

```
outliers['price'].describe()
```

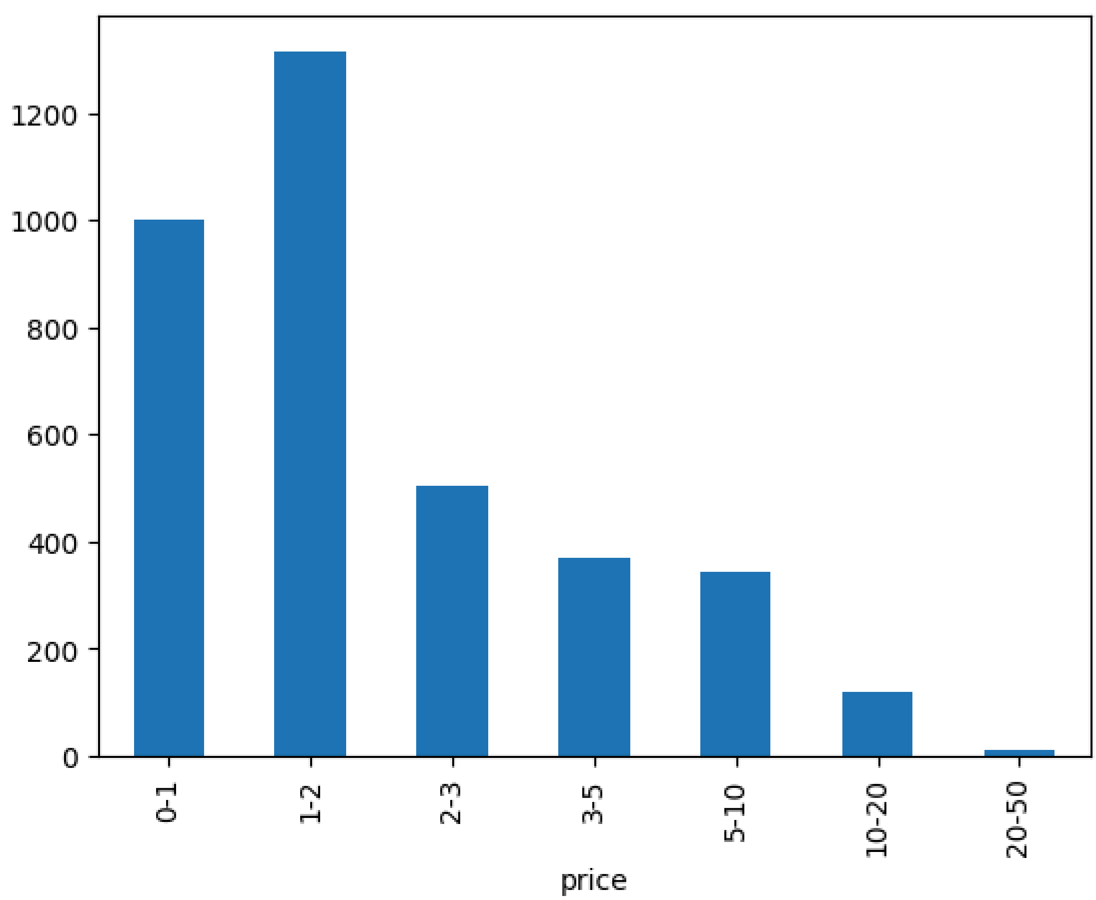
```
count    425.000000
mean       9.235624
std        4.065259
min        5.460000
25%        6.460000
50%        8.000000
75%       10.750000
max       31.500000
Name: price, dtype: float64
```

Observation:

Outliers Analysis (using IQR method):

- Based on the IQR method, there are 425 properties considered as outliers.
- These outliers have an average price of approximately 9.24 crores.
- The range for these outliers is from 5.46 crores to 31.5 crores.

```
# price binning
bins = [0, 1, 2, 3, 5, 10, 20, 50]
bin_labels = ["0-1", "1-2", "2-3", "3-5", "5-10", "10-20", "20-50"]
pd.cut(df['price'], bins=bins, labels=bin_labels, right=False).value_counts().so
rt_index().plot(kind='bar')
```

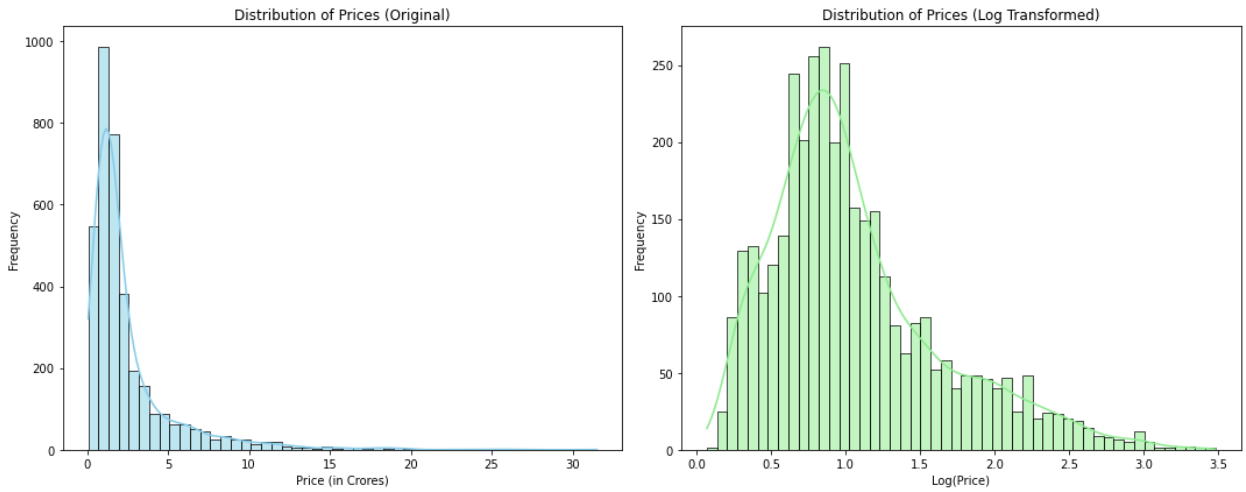


Observation:

- The majority of properties are priced in the "1-2 crores" and "2-3 crores" ranges.
- There's a significant drop in the number of properties priced above "5 crores."

Apply log transformation to the right skewed data:

```
plt.figure(figsize=(15, 6))

# Distribution plot without log transformation
plt.subplot(1, 2, 1)
sns.histplot(df['price'], kde=True, bins=50, color='skyblue')
plt.title('Distribution of Prices (Original)')
plt.xlabel('Price (in Crores)')
plt.ylabel('Frequency')

# Distribution plot with log transformation
plt.subplot(1, 2, 2)
sns.histplot(np.log1p(df['price']), kde=True, bins=50, color='lightgreen')
plt.title('Distribution of Prices (Log Transformed)')
plt.xlabel('Log(Price)')
plt.ylabel('Frequency')

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

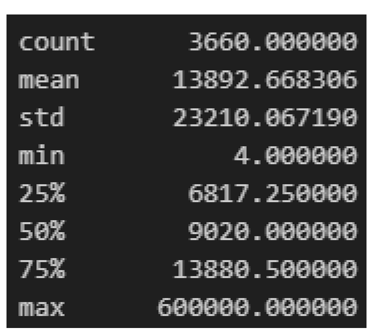


- np.log1p(x): This function computes the natural logarithm of 1+x.
  It's designed to provide more accurate results for values of x that are very close to zero.
- Using np.log1p helps in transforming the price column while ensuring that any value (including zero, if present) is handled appropriately. When we need to reverse the transformation, we can use np.expm1 which computes e^x-1

price_per_sqft

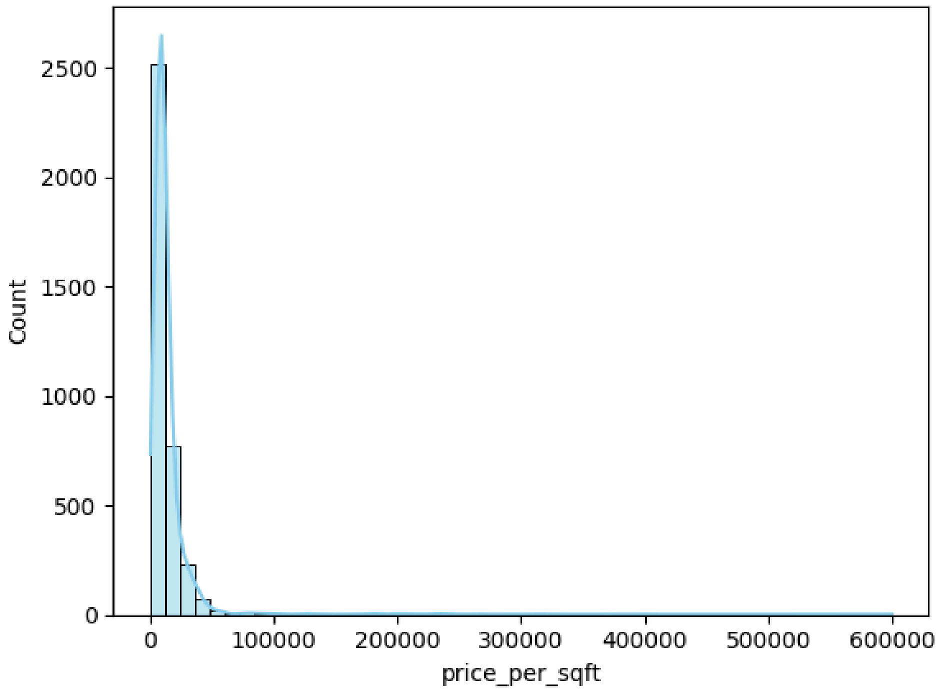
```
df['price_per_sqft'].isnull().sum()

Output: 17
```

```
df['price_per_sqft'].describe()
```



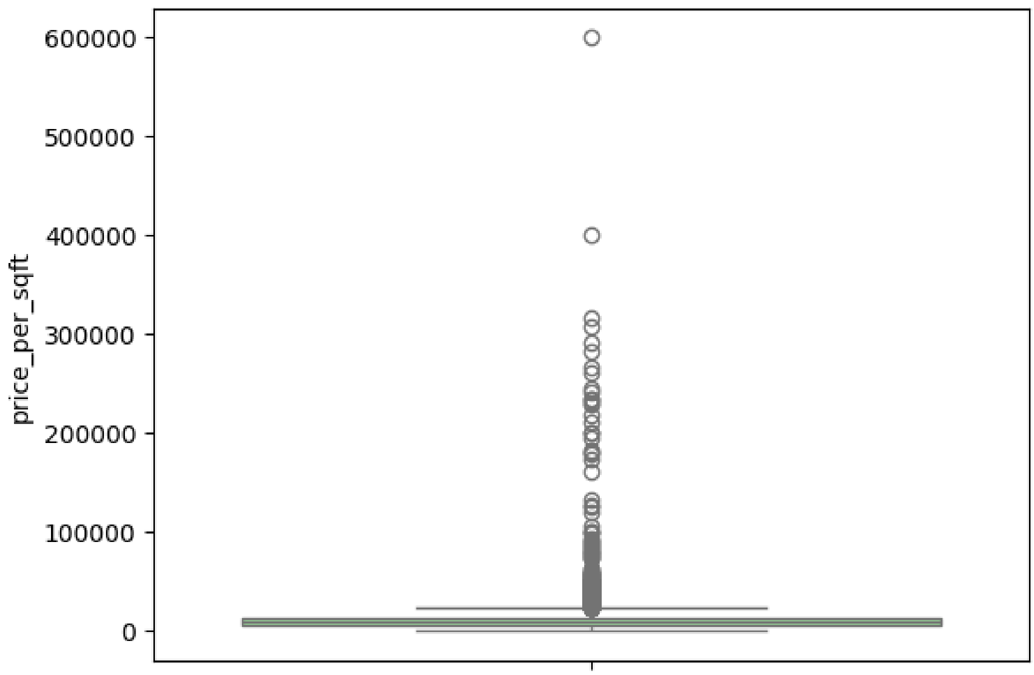
count	3660.000000
mean	13892.668306
std	23210.067190
min	4.000000
25%	6817.250000
50%	9020.000000
75%	13880.500000
max	600000.000000

```
sns.histplot(df['price_per_sqft'], bins=50, color='skyblue', kde=True)
```



- Most properties have a price_per_sqft ranging between approximately ₹0 and ₹40,000. There is a significant concentration in the lower range, with a few properties having exceptionally high **price_per_sqft**.

```
sns.boxplot(df['price_per_sqft'], color='lightgreen')
```



💡 🚩 Data has crazy outlier values. A flat cannot be **6L/sqft**🔴

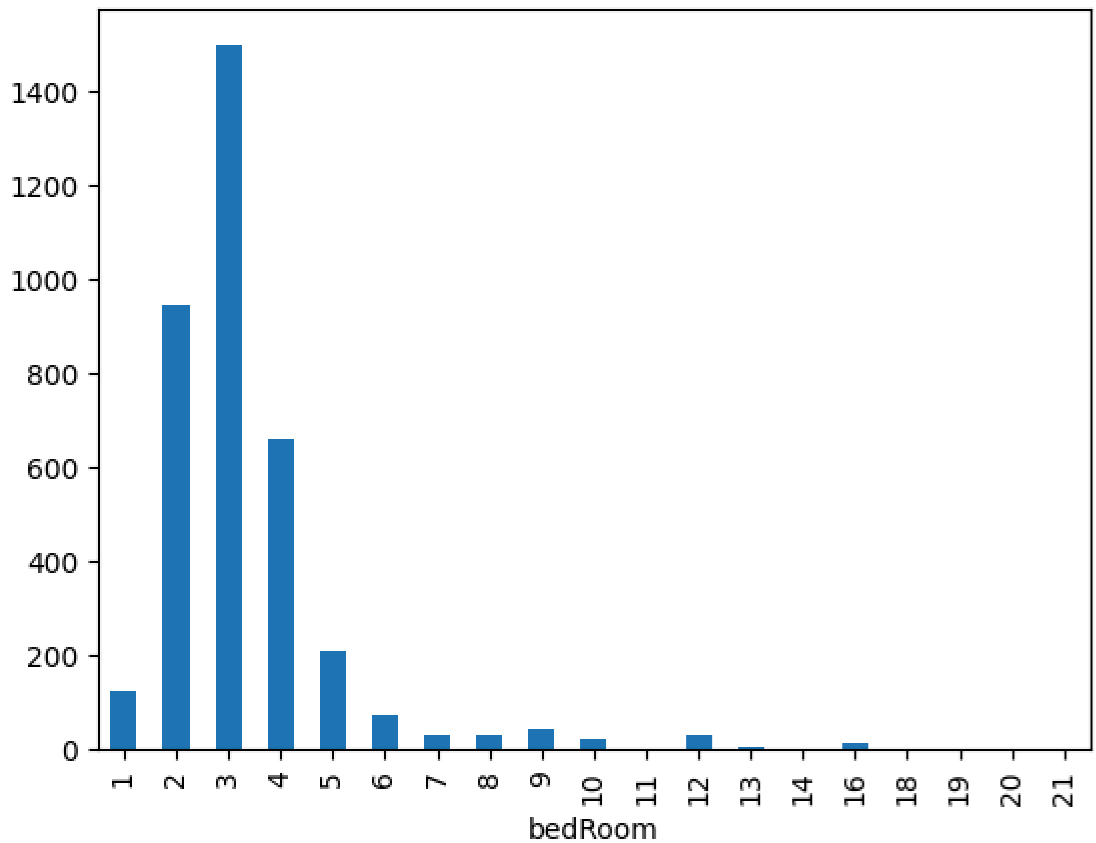
- The box plot clearly shows several outliers, especially on the higher side. The interquartile range (IQR) is relatively compact, but there are many data points beyond the "whiskers" of the box plot, indicating potential outliers

Observations:

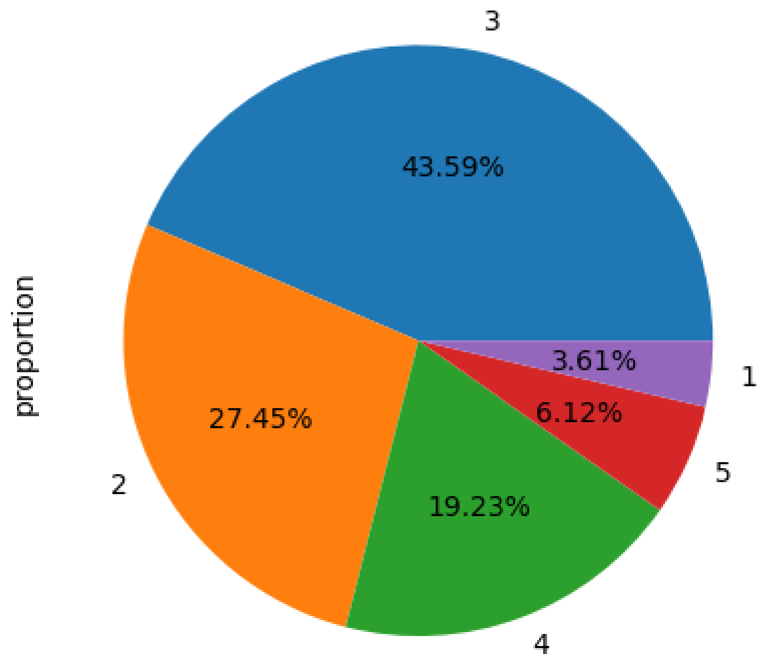
- Potential Outliers
- Right Skewed
- 17 missing values

bedRoom

```
df['bedRoom'].value_counts().sort_index().plot(kind='bar')
```

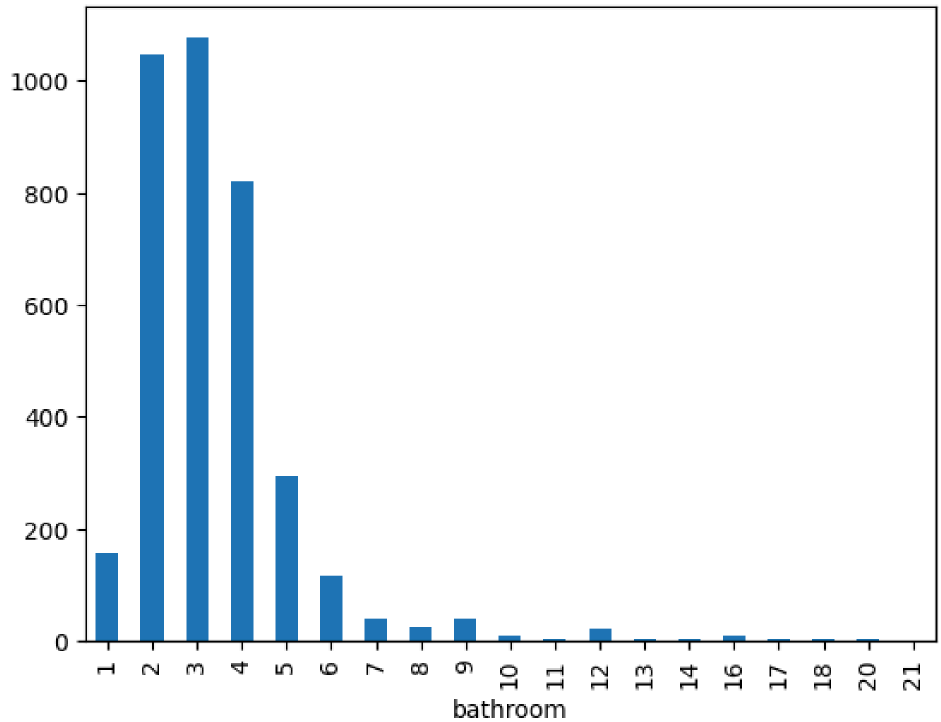



```
df['bedRoom'].value_counts(normalize=True).head().plot(kind='pie',autopct
='%0.2f%%')
```



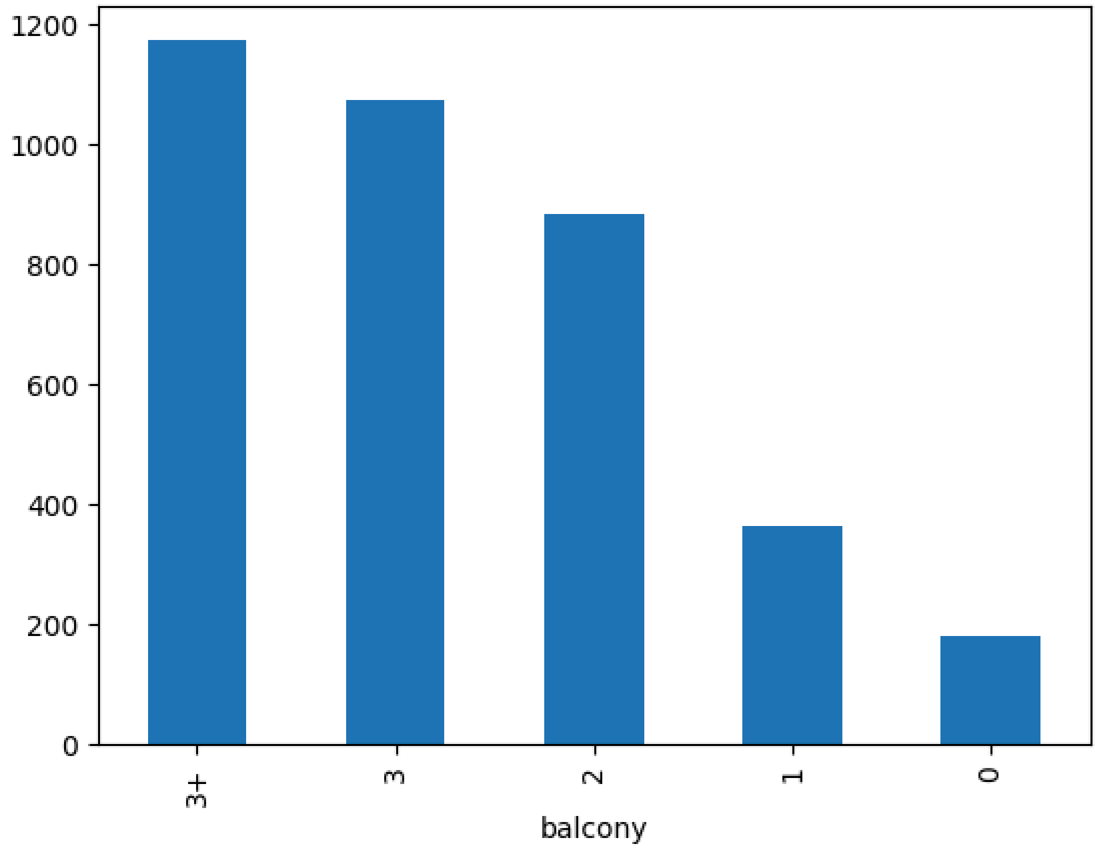
bathroom

```
df['bathroom'].value_counts().sort_index().plot(kind='bar')
```



balcony

```
df['balcony'].value_counts().plot(kind='bar')
```



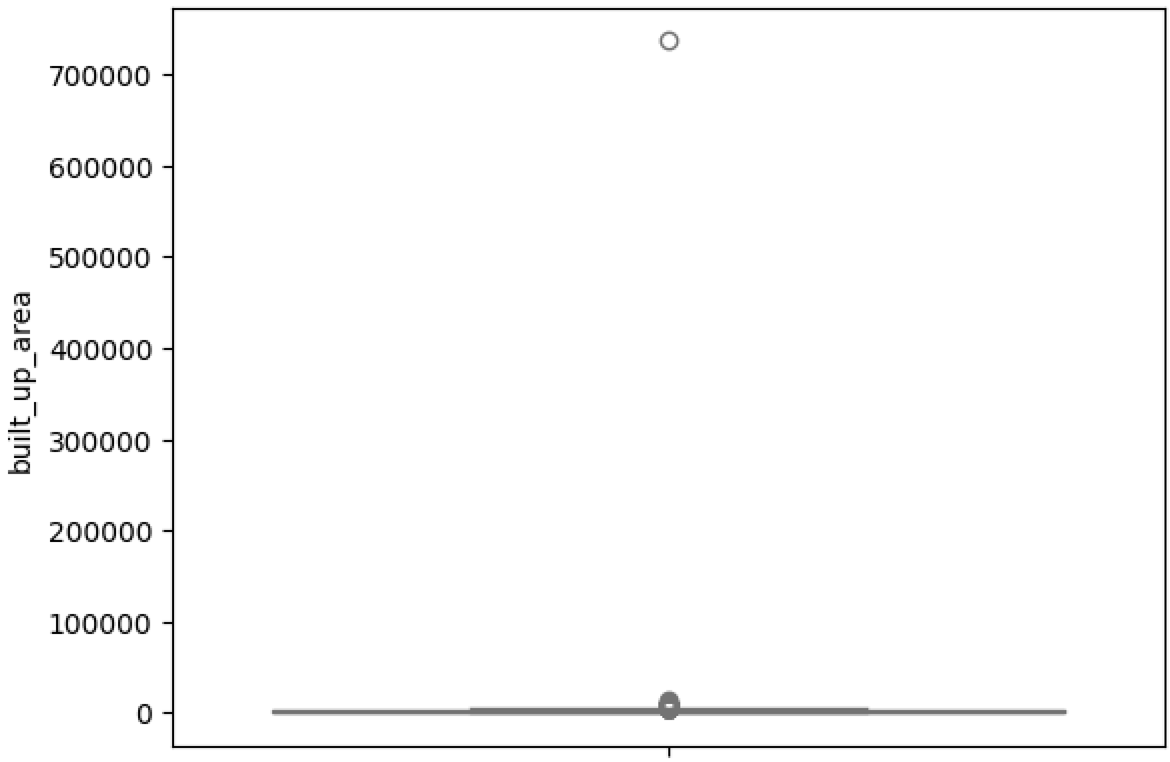
areas

- We'll use **built-up area**

```
df['built_up_area'].describe()
```

```
count      1690.000000
mean       2379.585816
std       17942.880237
min           2.000000
25%        1100.000000
50%        1650.000000
75%        2400.000000
max      737147.000000
Name: built_up_area, dtype: float64
```

```
sns.boxplot(df['built_up_area'].dropna(), color='lightgreen')
```



Observation:

- Most properties have a built-up area ranging roughly between 500 sq.ft and 3,500 sq.ft.
- There are very few properties with a much larger built-up area, leading to a highly right-skewed distribution.
- The box plot confirms the presence of significant outliers on the higher side. The data's interquartile range (IQR) is relatively compact, but the "whiskers" of the box plot are stretched due to the outliers.

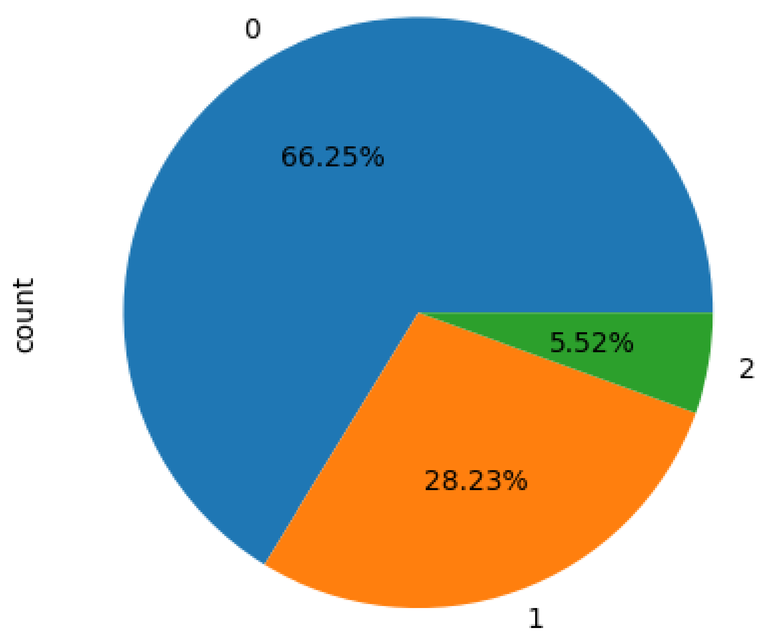
The presence of extreme values, especially on the higher side, suggests that there may be outliers or data errors. This could also be due to some properties being exceptionally large, like a commercial complex or an entire building being listed.

furnishing_type

```
df['furnishing_type'].value_counts()
```

```
furnishing_type
0    2436
1    1038
2     203
Name: count, dtype: int64
```

```
df['furnishing_type'].value_counts().plot(kind='pie',autopct='%0.2f%%')
```



Pandas Profiling

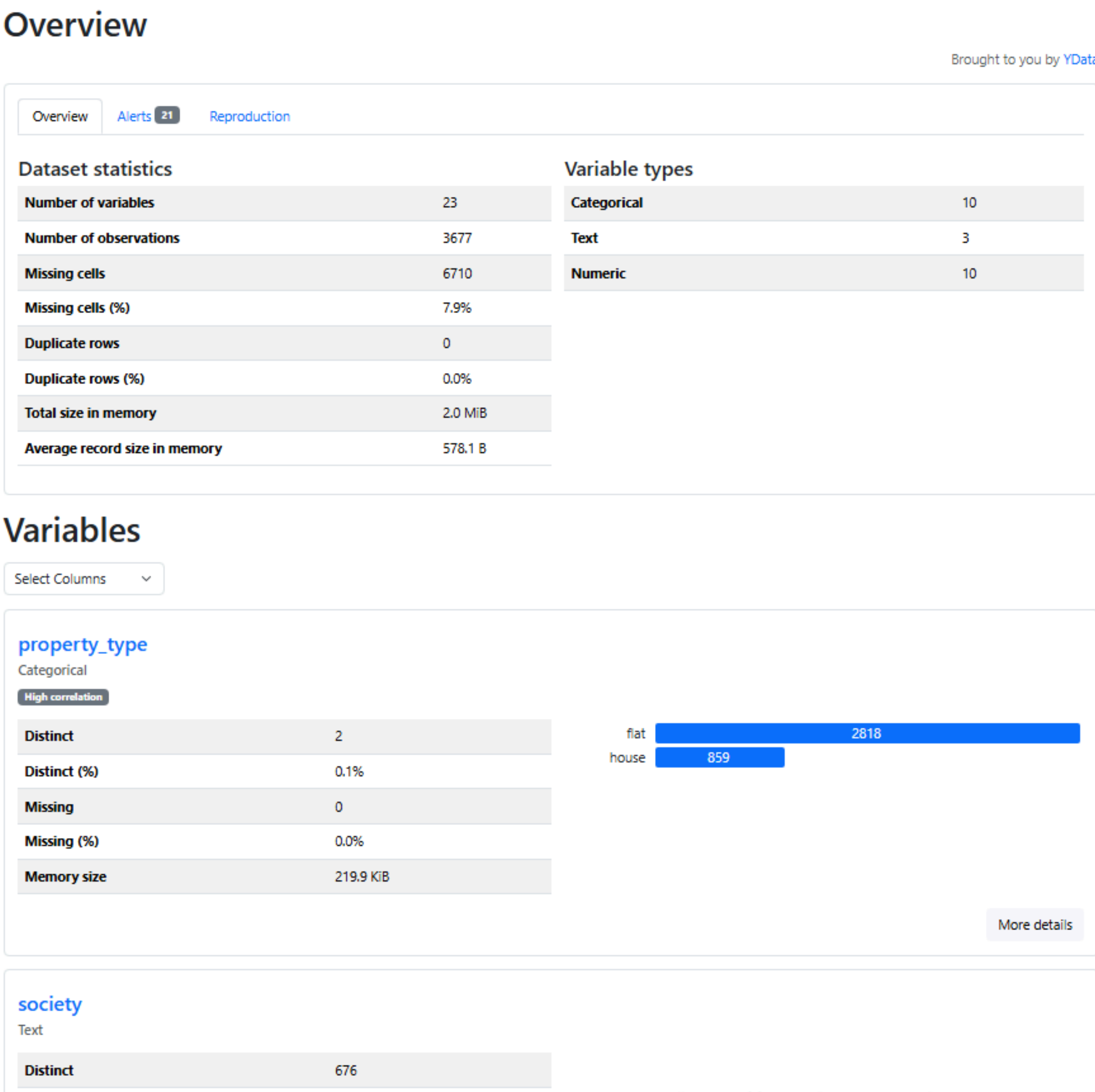
```
pip install ydata-profiling
```

```
!pip install pandas-profiling
```

```
import pandas as pd
from pandas_profiling import ProfileReport

# Load your dataset
df = pd.read_csv('gurgaon_properties_cleaned_v2.csv').drop_duplicates()

# Create the ProfileReport object
profile = ProfileReport(df, title='Pandas Profiling Report', explorative=True)

# Generate the report
profile.to_file("output_report.html")
```

- This will generate an HTML file
- This file has all the analysis

💡 **It performs automated EDA**


Overview

Brought to you by YData

Overview | Alerts 21 | Reproduction

Dataset statistics

Number of variables	23
Number of observations	3677
Missing cells	6710
Missing cells (%)	7.9%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	2.0 MiB
Average record size in memory	578.1 B

Variable types

Categorical	10
Text	3
Numeric	10

Variables

Select Columns

property_type

Categorical

High correlation

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	219.9 KiB

More details

society

Text

Distinct	676

Better Alternative to Pandas Profiling is **Detailed Library**