Exploratory Data Analysis for House Rental Prices using Python

Tools used: Python, Jupyter Notebook

GitHub link:

```
https://github.com/shivam21github/Exploratory-Data-Analysis-for-House-Rental-Prices-using-Python
```

Inspiration for this project

A Twitter thread by Andrea D’Agostino, Senior Data Scientist at DigiTrails

https://twitter.com/theDrewDag/status/1521970081810554881?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1521970081810554881%7Ctwgr%5Eb933cca31cb11bda7e67dd416aa43888f8e09268%7Ctwcon%5Es1_&ref_url=https%3A%2F%2Fcdn.embedly.com%2Fwidgets%2Fmedia.html%3Ftype%3Dtext2Fhtmlkey%3Da19fcc184b9711e1b4764040d3dc5c07schema%3Dtwitterurl%3Dhttps3A%2F%2Ftwitter.com%2Fthedrewdag%2Fstatus%2F1521970081810554881image%3Dhttps3A%2F%2Fi.embed.ly%2F1%2Fimage3Furl3Dhttps253A252F252Fabs.twimg.com252Ferrors252Flogo46x38.png26key3Da19fcc184b9711e1b4764040d3dc5c07

In the early days of my data analytics journey, I would see a lot of my colleagues and other data enthusiasts I knew being obsessed with data science—the idea of building revolutionary and game-changing prediction models and algorithms. And I got pulled in that direction too.

Later, having talked to experienced professionals working in the field and having gained some real-market experience myself, I realized that even though the more advanced data science applications do have their place in business, most people (clients and other business stakeholders) would derive immense value from well-conducted descriptive analysis.

And this project is an attempt to validate that observation.

Why this project?

I am an international student residing in Toronto who comes from the fascinating country of India. Having lived a financially privileged and comfortable life, I never had to worry about finding a place to live or paying my rent. This was before I came to Canada as an immigrant.

After arriving in this country, I was like a fish out of water, humbled, and forced to take on this responsibility. Those of us who reside in Canada (Toronto specifically) or follow the news are familiar with the obnoxious rental market right now. The rental prices are touching the roof and it’s an exhausting challenge to find a suitable place.

Having experienced this firsthand, I got curious to compare the situation in Toronto to some of the major cities in my home country. And that happens to be the objective of this portfolio project.

🎯 To assess the renting scenario for the major cities in India (conduct exploratory data analysis for the same) and identify interesting patterns and observations from the analysis. Simple and fun.

Let's get into the process now.

Importing relevant libraries

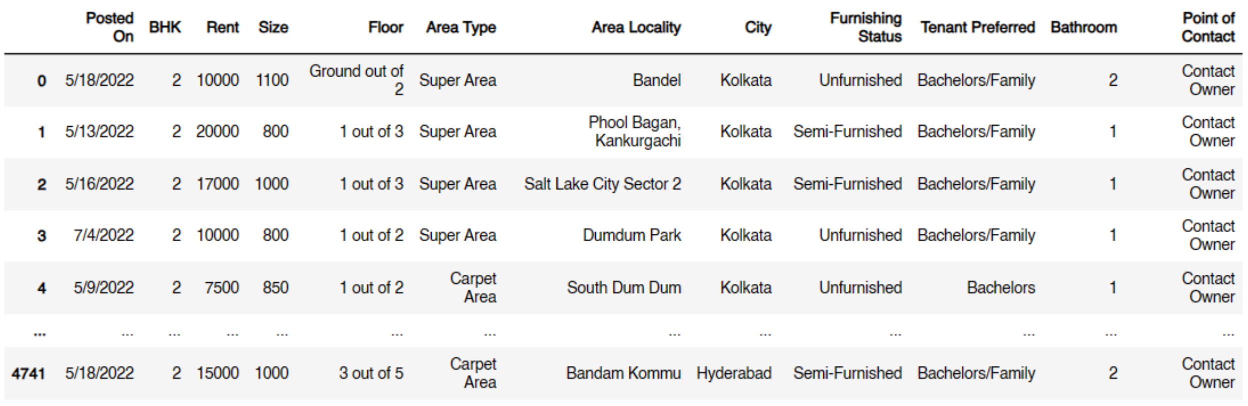
As the first step, it's important to import the necessary Python packages.

```
#For data manipulation
import pandas as pd
import numpy as np
```

```
#For data visualization
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
```

Loading the dataset

```
df = pd.read_csv("House_Rent_Dataset.csv")
#Viewing a sample of our data
df
```

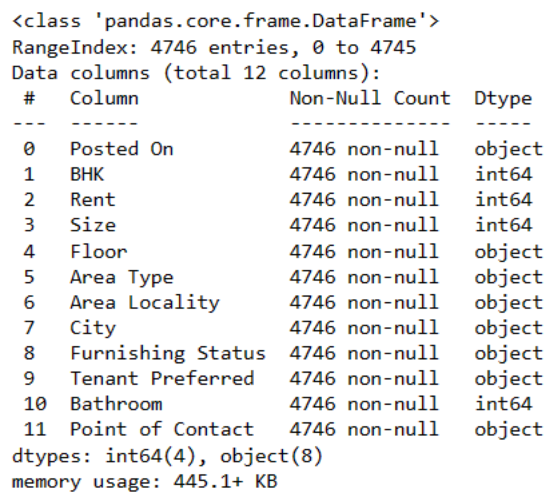


	Posted On	BHK	Rent	Size	Floor	Area Type	Area Locality	City	Furnishing Status	Tenant Preferred	Bathroom	Point of Contact
0	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
1	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
2	5/16/2022	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
4	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner
...	...	...	...	...	...	...	...	...	...	...	...	...
4741	5/18/2022	2	15000	1000	3 out of 5	Carpet Area	Bandam Kommu	Hyderabad	Semi-Furnished	Bachelors/Family	2	Contact Owner

Comprehending and studying the data from a high level

- .info() function

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Posted On          4746 non-null   object
 1   BHK                4746 non-null   int64 
 2   Rent               4746 non-null   int64 
 3   Size               4746 non-null   int64 
 4   Floor              4746 non-null   object
 5   Area Type          4746 non-null   object
 6   Area Locality      4746 non-null   object
 7   City               4746 non-null   object
 8   Furnishing Status  4746 non-null   object
 9   Tenant Preferred   4746 non-null   object
 10  Bathroom           4746 non-null   int64 
 11  Point of Contact   4746 non-null   object
dtypes: int64(4), object(8)
memory usage: 445.1+ KB
```

ℹ️ `.info()` is a Pandas method that provides a concise summary of a DataFrame, including information about the data types, non-null values, and memory usage.

- Creating a data dictionary to clearly recognize different fields.

Data Dictionary

0. Date Posted: The date when the property listing was posted

1. BHK: The no. of bedrooms with a hall and a kitchen

2. Rent: The rent price for the property (in Indian Rupees)

3. Size: The size of the property (in square feet)

4. Floor: The floor where the property is situated along with the total number of floors (in case of a multi-storey building)

5. Area Type: The size of the property classified as

6. Area Locality: The area or neighbourhood of the property.

7. City: The city where the property is located

8. Furnishing Status: The level of furniture of amenities provided in the property

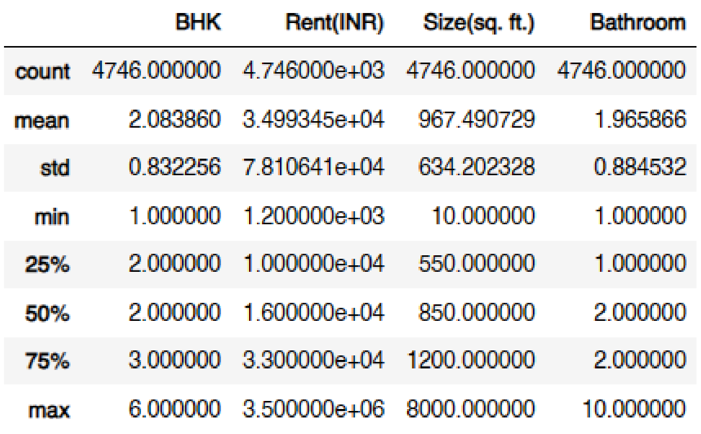
9. Tenant Preferred: The type of tenant as per the desried characteristics of the tenant.

10. Bathroom: The no. of bathrooms available in the property

11. Point of Contact: The assigned person/party that should be contacted to discuss the property arrangement

- Fetching descriptive information about the data using the .describe() function

```
df.describe()
```



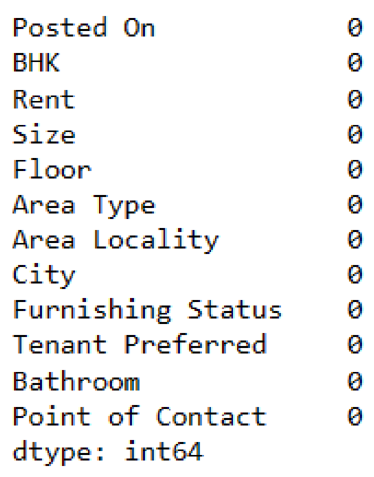
	BHK	Rent(INR)	Size(sq. ft.)	Bathroom
count	4746.000000	4.746000e+03	4746.000000	4746.000000
mean	2.083860	3.499345e+04	967.490729	1.965866
std	0.832256	7.810641e+04	634.202328	0.884532
min	1.000000	1.200000e+03	10.000000	1.000000
25%	2.000000	1.000000e+04	550.000000	1.000000
50%	2.000000	1.600000e+04	850.000000	2.000000
75%	3.000000	3.300000e+04	1200.000000	2.000000
max	6.000000	3.500000e+06	8000.000000	10.000000

`.describe()` function in Python generates summary statistics for a dataset, such as mean, standard deviation, and quartiles.

> Note that descriptive statistics for only **integer (int64)** data types were generated. The **text or object** data types were excluded since it's implausible to generate quantitative descriptive statistics for them.

- Confirming the absence of null values using the .isnull() function

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

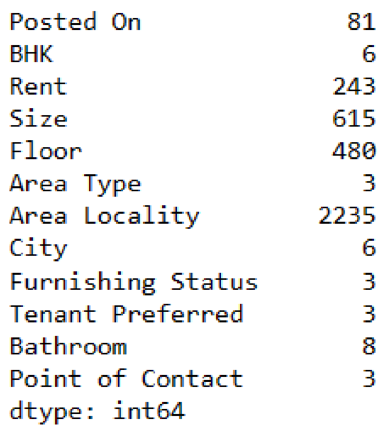


```
Posted On           0
BHK                 0
Rent                0
Size                0
Floor               0
Area Type           0
Area Locality       0
City                0
Furnishing Status   0
Tenant Preferred    0
Bathroom            0
Point of Contact    0
dtype: int64
```

ℹ️ `.isnull()` checks if there are any missing values and returns a True/False mask.

- Counting unique values using the .nunique() function

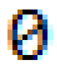
```
df.nunique()
```


```
Posted On              81
BHK                     6
Rent                  243
Size                  615
Floor                 480
Area Type               3
Area Locality        2235
City                    6
Furnishing Status       3
Tenant Preferred        3
Bathroom                8
Point of Contact        3
dtype: int64
```

`.nunique()` is a Pandas method used to count the number of unique elements in a DataFrame or Series, excluding any null or missing values.

- Checking for duplicate values using the .duplicated() function

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


```
0
```

`.duplicated()` is a Pandas method that identifies duplicate rows in a DataFrame and returns a boolean mask indicating whether each row is a duplicate or not.

This confirms that there are **no duplicate values** in the dataset. This is a good sign.

Data Cleaning

Performing some data cleaning steps to ensure data accuracy and consistency before conducting any analysis or modeling

- Renaming columns to make them more descriptive

```
df.rename(columns={"Rent":"Rent(INR)", "Size":"Size(sq. ft.)"}, inplace=True)
```

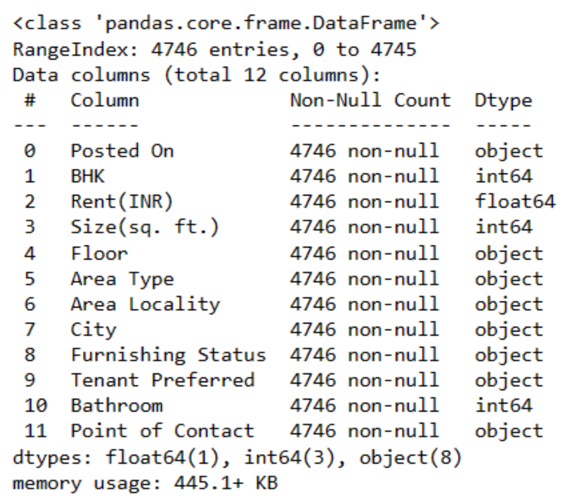
	Posted On	BHK	Rent(INR)	Size(sq. ft.)	Floor	Area Type	Area Locality	City	Furnishing Status	Tenant Preferred	Bathroom	Point of Contact
0	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
1	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
2	5/16/2022	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
4	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner
...	...	...	...	...	...	...	...	...	...	...	...	...
4741	5/18/2022	2	15000	1000	3 out of 5	Carpet Area	Bandam Kommu	Hyderabad	Semi-Furnished	Bachelors/Family	2	Contact Owner
4742	5/15/2022	3	29000	2000	1 out of 4	Super Area	Manikonda, Hyderabad	Hyderabad	Semi-Furnished	Bachelors/Family	3	Contact Owner
4743	7/10/2022	3	35000	1750	3 out of 5	Carpet Area	Himayath Nagar, NH 7	Hyderabad	Semi-Furnished	Bachelors/Family	3	Contact Agent
4744	7/6/2022	3	45000	1500	23 out of 34	Carpet Area	Gachibowli	Hyderabad	Semi-Furnished	Family	2	Contact Agent
4745	5/4/2022	2	15000	1000	4 out of 5	Carpet Area	Suchitra Circle	Hyderabad	Unfurnished	Bachelors	2	Contact Owner

4746 rows × 12 columns

`.rename()` function is used to change the names of columns or index labels in a Pandas DataFrame.

- Changing the data type of the 'Rent(INR)' column

```
df['Rent(INR)'] = df['Rent(INR)'].astype(float)
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Posted On          4746 non-null   object
 1   BHK                4746 non-null   int64
 2   Rent(INR)          4746 non-null   float64
 3   Size(sq. ft.)      4746 non-null   int64
 4   Floor              4746 non-null   object
 5   Area Type          4746 non-null   object
 6   Area Locality      4746 non-null   object
 7   City               4746 non-null   object
 8   Furnishing Status  4746 non-null   object
 9   Tenant Preferred   4746 non-null   object
 10  Bathroom           4746 non-null   int64
 11  Point of Contact   4746 non-null   object
dtypes: float64(1), int64(3), object(8)
memory usage: 445.1+ KB
```

ℹ️ `.astype()` function is used in Pandas to change the data type of a column in a DataFrame to a specified type.

Converted the 'Rent(INR)' column from an integer type to a float type for better precision.

Univariate Analysis (single-variable analysis)

After completing the previous steps, we obtained a high-level overall picture of the dataset. These initial analyses provided us with valuable insights into the data's structure and distribution, giving us a solid foundation for further exploration.

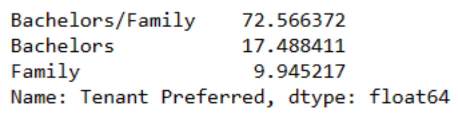
In the next stage, we will focus on a more precise assessment of a subset of variables that includes both categorical and numerical types. This will provide us with deeper insights and a better understanding of their characteristics and contributions to the dataset.

Categorical variables chosen: **Tenant Preferred, City** and **Furnishing Status**

1. **Tenant Preferred**

Understanding the percentage distribution of different categories

```
percentage_dist_tenant = df['Tenant Preferred'].value_counts(normalize=True) * 100
percentage_dist_tenant
```

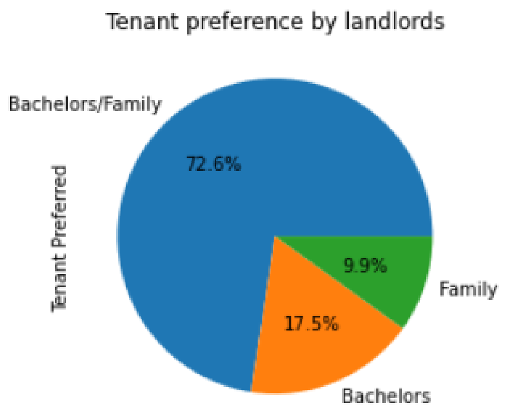


```
Bachelors/Family    72.566372
Bachelors           17.488411
Family               9.945217
Name: Tenant Preferred, dtype: float64
```

ℹ️ `.value_counts()` with `normalize=True` returns the percentage distribution of unique values in a Pandas Series or DataFrame column.

Plotting the above data for better comprehension:

```
percentage_dist_plot_tenant = df['Tenant Preferred'].value_counts().plot(kind="pie", autopct='%1.1f%%')
plt.title("Tenant preference by landlords")
percentage_dist_plot_tenant
```



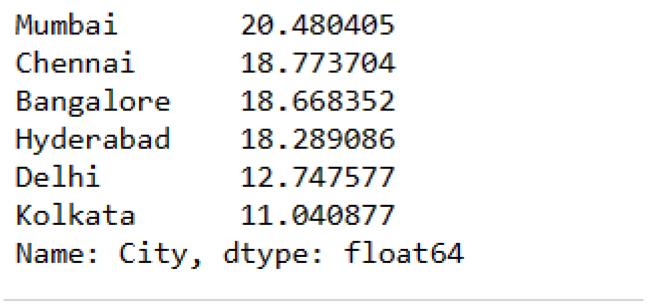
ℹ️ `.value_counts().plot()` visualizes the percentage distribution of unique values in a Pandas Series or DataFrame column in the form of a chart (e.g. bar, pie, etc.)

Insights:

- The majority of landlords are indifferent about the tenants being bachelors or family, as per the data.
- Speaking from personal experience, there exists a general tendency to prefer families as tenants due to reasons like an added sense of financial security and the overall civility that a family arrangement, apparently, brings with it.

2. **City**

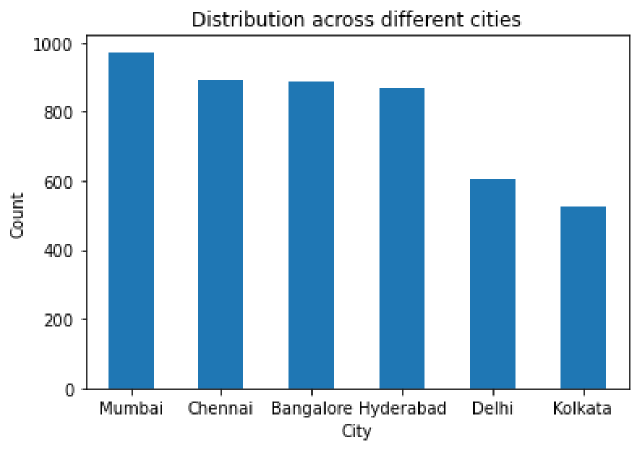
```
percentage_dist_city = df['City'].value_counts(normalize=True) * 100
percentage_dist_city
```



```
Mumbai       20.480405
Chennai      18.773704
Bangalore    18.668352
Hyderabad    18.289086
Delhi        12.747577
Kolkata      11.040877
Name: City, dtype: float64
```

Plotting the above data for better comprehension

```
percentage_dist_plot_city = df['City'].value_counts().plot(kind="bar")
plt.title("Distribution across different cities")
plt.xlabel("City")
plt.xticks(rotation=0)
plt.ylabel("Count")
percentage_dist_plot_city
```



The distribution looks fairly balanced, with the exception of Delhi and Kolkata.

Insights:

- Cities like Mumbai, Bangalore, Chennai, and Hyderabad offer more rental options. One of the potential reasons could be the fact that these cities attract a lot of migrant professionals who come to these cities seeking jobs.

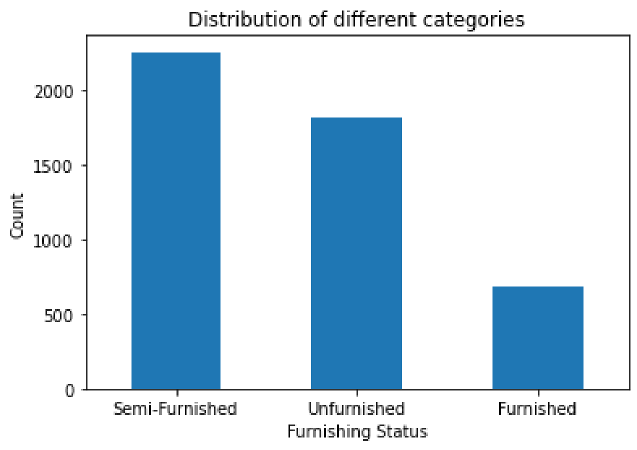
3. **Furnishing Status**

```
percentage_dist_furnish = df['Furnishing Status'].value_counts(normalize=True) * 100
percentage_dist_furnish
```

```
Semi-Furnished    47.429414
Unfurnished       38.242731
Furnished         14.327855
Name: Furnishing Status, dtype: float64
```

Plotting the above data for better comprehension:

```
percentage_dist_plot_furnish = df['Furnishing Status'].value_counts().plot(kind="bar")
plt.title("Distribution of different categories")
plt.xlabel("Furnishing Status")
plt.xticks(rotation=0)
plt.ylabel("Count")
percentage_dist_plot_furnish
```



Similar to the 'Area Type' variable, the 'Furnished' class is considerably smaller than the other two classes.

Insights:

- Most listed properties are semi-furnished or unfurnished. One of the rationales behind doing this would be from the landlord's or owner's point of view, as having minimal or no furnishing facilities is cost-effective as there is no expenditure on the purchase and maintenance of furniture and appliances.

  This makes sense for the Indian market, which is price-sensitive, generally speaking.

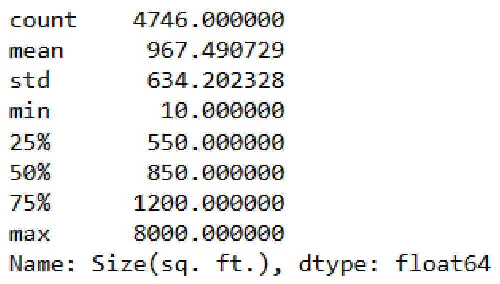
Moving on to numeric variables now…

Numeric variables chosen: **Size (sq. ft.)** and **BHK**

4. **Size (sq. ft.)**

Calculating the summary statistics:

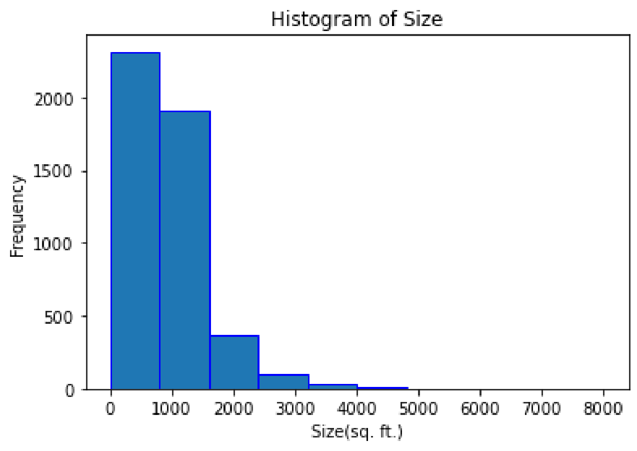
```
size_stats = df['Size(sq. ft.)'].describe()
size_stats
```



```
count    4746.000000
mean      967.490729
std       634.202328
min        10.000000
25%       550.000000
50%       850.000000
75%      1200.000000
max      8000.000000
Name: Size(sq. ft.), dtype: float64
```

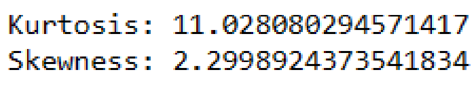
Plotting a histogram to visualize the distribution:

```
size_stats_hist = plt.hist(df['Size(sq. ft.)'], bins = 10, edgecolor='blue')
plt.xlabel('Size(sq. ft.)')
plt.ylabel('Frequency')
plt.title('Histogram of Size')
size_stats_hist
```



Evaluating the **Kurtosis** and **Asymmetry**:

```
print(f"Kurtosis: {df['Size(sq. ft.)'].kurt()}")
print(f"Skewness: {df['Size(sq. ft.)'].skew()}")
```



```
Kurtosis: 11.028080294571417
Skewness: 2.2998924373541834
```

Kurtosis measures the **presence of outliers** in the data.

- A kurtosis value of 11.03 indicates the presence of extreme values.

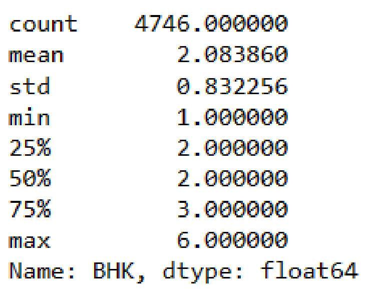
Skewness gauges the **asymmetry of data** and indicates whether the data is skewed towards: the **left (negative)** or **right (positive)**

- A skewness value of 2.29 indicates that the data is positively skewed, i.e., skewed towards higher values.

5. **BHK**

Calculating the summary statistics:

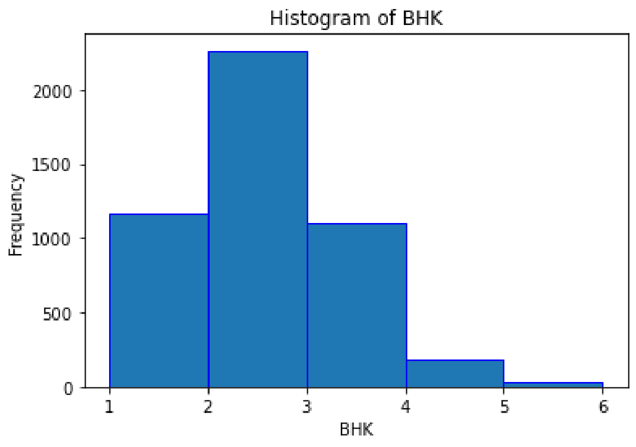
```
bhk_stats = df['BHK'].describe()
bhk_stats
```



```
count    4746.000000
mean        2.083860
std         0.832256
min         1.000000
25%         2.000000
50%         2.000000
75%         3.000000
max         6.000000
Name: BHK, dtype: float64
```

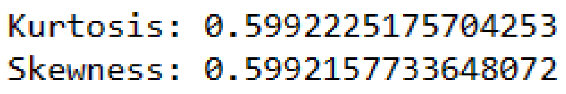
Plotting a histogram to visualize the distribution:

```
bhk_stats_hist = plt.hist(df['BHK'], bins = 5, edgecolor='blue')
plt.xlabel('BHK')
plt.ylabel('Frequency')
plt.title('Histogram of BHK')
bhk_stats_hist
```



Evaluating the Kurtosis and Asymmetry:

```
print(f"Kurtosis: {df['BHK'].kurt()}")
print(f"Skewness: {df['BHK'].skew()}")
```



```
Kurtosis: 0.5992225175704253
Skewness: 0.5992157733648072
```

- A kurtosis value of 0.59 indicates the presence of fewer outliers or extreme values.
- A skewness value of 0.59 indicates that their slight positive skewness i.e. minutely skewed towards higher values.

Multi-variate analysis (multiple-variable analysis)

Having studied the characteristics of variables individually, the next logical step is to assess the relationships between different variables. This involves searching for signs of correlation, dependencies, and other relevant patterns that may exist within the dataset. By exploring these interconnections, we can gain valuable insights into how different variables interact and influence one another, paving the way for a deeper understanding of the underlying data dynamics and potential areas for further analysis.

- Finding correlations between different variables using the corr() function

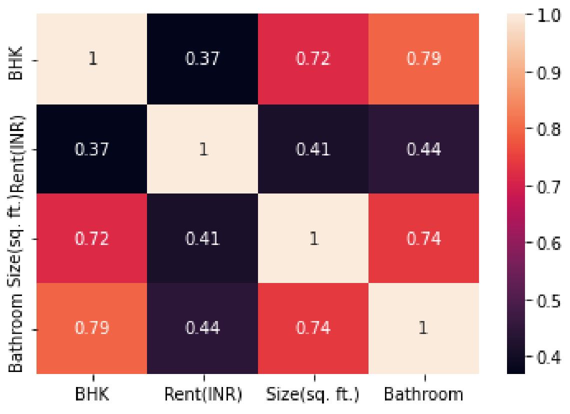
```
df.corr()
```

	BHK	Rent(INR)	Size(sq. ft.)	Bathroom
BHK	1.000000	0.369718	0.716145	0.794885
Rent(INR)	0.369718	1.000000	0.413551	0.441215
Size(sq. ft.)	0.716145	0.413551	1.000000	0.740703
Bathroom	0.794885	0.441215	0.740703	1.000000

ℹ️ `.corr()` function in pandas calculates the correlation between numeric variables in a DataFrame, revealing their linear relationships through a correlation matrix.

To gain a better view of what the above table means, let's create a heatmap:

```
sns.heatmap(df.corr(), annot = True)
plt.rcParams['figure.figsize'] = (20,7)
plt.show()
```



Insights:

- The above heatmap only takes into account numeric variables since they exist in a quantitative form and can be visualized for this chart.

- It is interesting to note that there are no strong correlations between rent and other variables like BHK, Size(sq. ft.) and Bathroom. From common parlance, it is accepted that these factors dictate the rent of a property (the hypothesis) but the numbers do not prove (or at least cement) this hypothesis.

> *Note: As per general standards, a correlation **coefficient value below 0.5** is considered a moderate to weak correlation.*

Comparing Rent against other variables and plotting the results

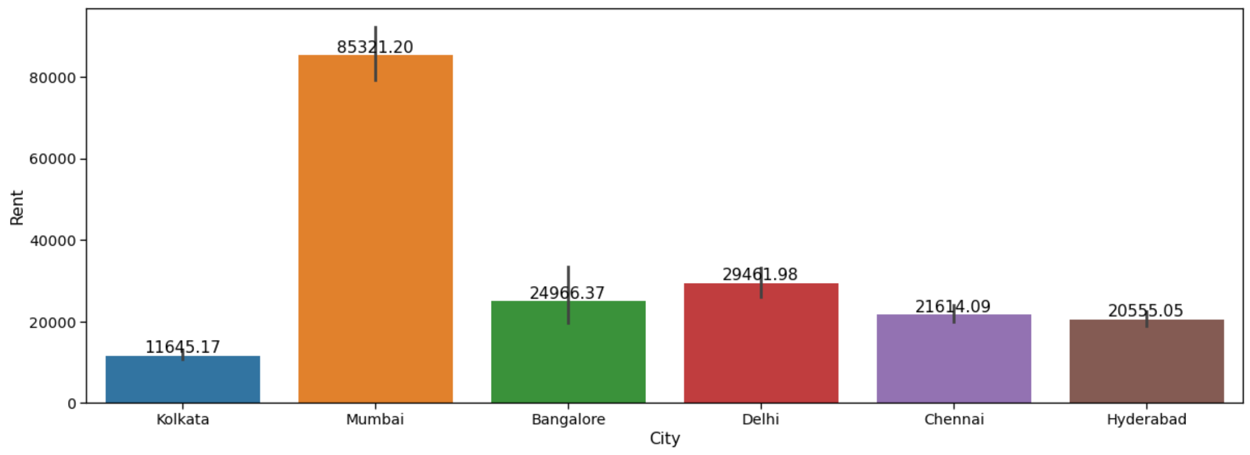
- Comparing Rent prices for different cities

```
#Calculate the average rent for each category (City)
average_rent_by_city = df.groupby('City')['Rent(INR)'].mean()
```

```
#Comparing Rent (prices) for different cities
rent_by_city = sns.barplot(x=df['City'], y=df['Rent(INR)'])
plt.xlabel('City')
plt.ylabel('Rent')
```

```
#Add individual average rent values at the top of the bars
for i, bar in enumerate(rent_by_city.patches):
height = bar.get_height()
plt.text(bar.get_x() + bar.get_width() / 2, height, f'{average_rent_by_city[df["City"].unique()[i]]:.2f}', ha='center', va='bottom')
```

```
#Display the plot
plt.show()
```

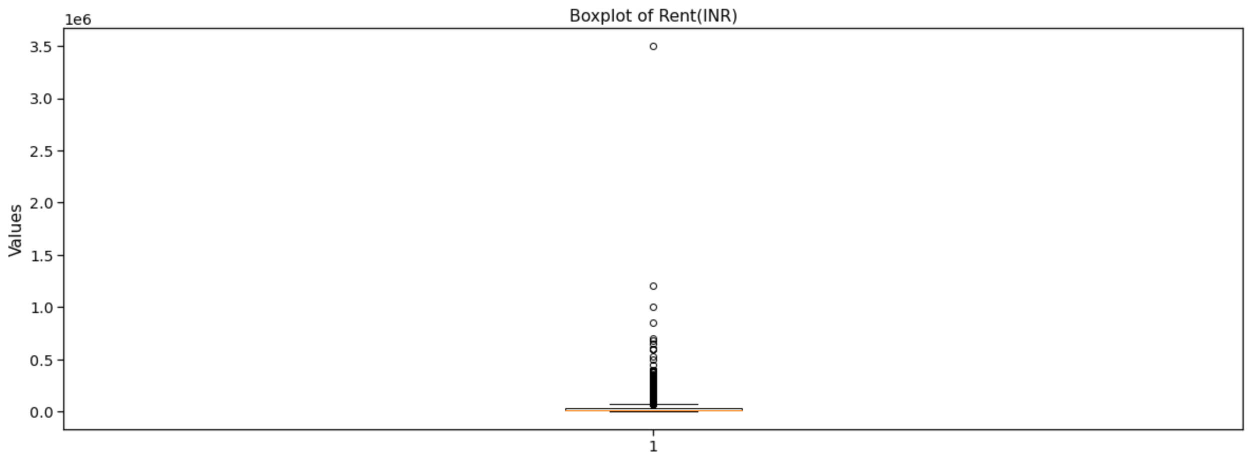


Insights:

- It looks like the average rent amount is the highest in the city of Mumbai, followed by Delhi and Bangalore. That’s what we can deduce from prima facie.
- Though, the rent prices for the city of Mumbai do appear to be abnormally high relative to other cities. This points towards the presence of outliers (extreme values that skew the data results).

Creating a boxplot to identify outliers:

```
plt.boxplot(df['Rent(INR)'])
plt.title('Boxplot of Rent(INR)')
plt.ylabel('Values')
```



Insights:

- The rent field contains a lot of outliers. As can be observed from the graph, a lot of extreme values belong to the upper end of the boxplot whiskers, which indicates that the data is skewed towards the extremely high rent values present in the dataset (this could be the potential reason for Mumbai’s obnoxiously high average rent price)
- If the goal of the project post completing the EDA process is to model the data (i.e., run algorithms or prediction models like linear regression), then these outliers must be treated accordingly in order to avoid the analysis getting distorted.

Removing the outliers

```
# Calculate the Z-score for each data point
z_scores = np.abs((df['Rent(INR)'] - df['Rent(INR)'].mean()) / df['Rent(INR)'].std())
```

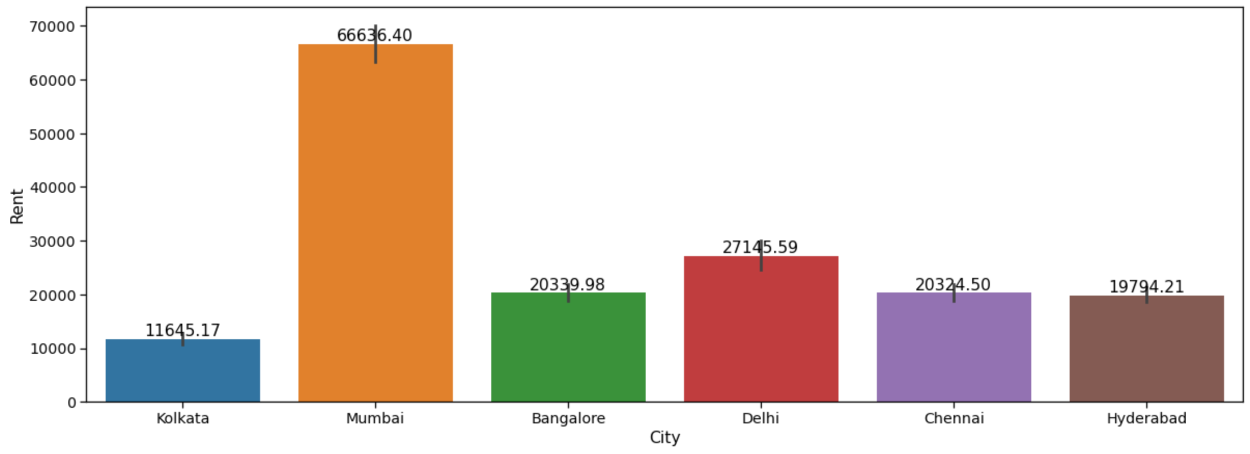
```
# Define a threshold for outlier detection
threshold = 3
```

```
# Identify outliers
outliers = df[z_scores > threshold]
```

```
# Remove outliers from the dataset
df_cleaned = df[z_scores <= threshold]
plt.show()
```

We calculate the **Z-score** for each data point in the 'Rent' column using the formula **(x - mean) / std**, where x is the data point, mean is the mean of the 'Rent' column, and std is the standard deviation of the 'Rent' column. We then define a threshold **(typically set to 3)** to determine which data points are considered outliers. Data points with a Z-score greater than the threshold are identified as outliers, and excluded.

After treating the outliers, now let's again visualize the data to compare rent prices for different cities:



The overall results remain the same with Mumbai having the highest average rent, followed by Delhi. But we can observe the change in values after treating (excluding) the outliers.

This gives a more accurate representation of the data.

Conclusion

The EDA process can be further developed to discover additional patterns and answer more analytics questions. It is an endless exploration.

I would like to conclude this project here as it fulfils my purpose of demonstrating skills: utilizing Python for data analysis.

Potentially, the next steps could involve creating a report containing various dynamic visualizations (using Python or another BI tool like Tableau) to present the results to stakeholders. Or we could indulge in data modelling, where we could run prediction algorithms like multiple linear regression on the data to predict the price of a property given the different factors/variables

References

D’Agostino, A. (2023, June 27). Exploratory Data Analysis in Python — A Step-by-Step Process. *Medium*. https://towardsdatascience.com/exploratory-data-analysis-in-python-a-step-by-step-process-d0dfa6bf94ee

Andrea D'Agostino, [@theDrewDag], (2022, May 4), [Data Analysis] 🧵 Exploratory data analysis is a fundamental step in any analysis work. You don't have to be a data scientist and be proficient at modeling to be a useful asset to your client if you can do great EDA. Here's a template of a basic yet powerful EDA workflow👇 [Tweet]. Twitter. https://twitter.com/theDrewDag/status/1521970081810554881?ref_src=twsrc^tfw|twcamp^tweetembed|twterm^1521970081810554881|twgr^c6ac0d88bae96898bdb126f443c9b05c4bd5fbc3|twcon
project-shivam-sury%2FExploratory-Data-Analysis-for-House-Rental-Prices-using-Python-56cda327c0a142959c3aca6dcb11b755

***Python Glossary*. (n.d.)**. https://www.w3schools.com/python/python_ref_glossary.asp

Alex The Analyst. (2023, May 23). *Data Cleaning in Pandas | Python Pandas Tutorials* [Video]. YouTube. https://www.youtube.com/watch?v=bDhvCp3_lYw

Gulati, A. P. (2022). Dealing with outliers using the Z-Score method. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2022/08/dealing-with-outliers-using-the-z-score-method/