Exploratory Data Analysis on House Rent Analysis in Hyderabad City

Step -1: Import all required Libraries

Firstly, we need to import all the required Libraries to Perform EDA. Libraries include NumPy, Pandas, Matplotlib, and Seaborn.

```
# importing all required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step - 2: Load the Dataset

Now, we read the Data from CSV file into a Pandas DataFrame.

```
# Load the Dataset
df = pd.read_csv(r"C:\Users\mouni\Desktop\All_web_scraping_projects\house_rent_analysis3.csv")
```

```
df
```

	Unnamed: 0	Title	Location	Price	Area	Status
0	0	4 BHK Independent House	Old Bowenpally, Hyderabad	16,500	1000	Unfurnished
1	1	2 BHK Independent House	Khairatabad Road, Hyderabad	14,000	700	Semi-Furnished
2	2	2 BHK Independent House	Shaikpet, Hyderabad	12,000	1200	Unfurnished
3	3	5 BHK Villa	Nallagandla Gachibowli, Hyderabad	80,000	4000	Semi-Furnished
4	4	5 BHK Villa	Moinabad, Hyderabad	200000	7000	Furnished
...	...	...	...	...	...	...
995	995	3 BHK Apartment	Ramachandra Puram, Hyderabad	18,000	1300	Semi-Furnished
996	996	3 BHK Apartment	Gajulramaram Kukatpally, Hyderabad	27,000	1620	Semi-Furnished
997	997	3 BHK Apartment	Krishna Reddy Pet, Hyderabad	18,000	1360	Semi-Furnished
998	998	3 BHK Apartment	Pocharam, Hyderabad	12,000	1500	Semi-Furnished
999	999	2 BHK Independent House	Kothapet, Hyderabad	13,500	1400	Unfurnished

1000 rows × 6 columns

Step – 3: Data Cleaning

If we see the DataFrame, Title Consists of information about BHK and Type of House. So that, we split the Title Column into Two Columns namely, BHK and Type Columns.

```
# split the Title Column into two Columns (i.e., BHK, Type of the House)
df[["BHK", "Type"]] = df["Title"].str.split("BHK", expand=True)
df
```

	Unnamed: 0	Title	Location	Price	Area	Status	BHK	Type
0	0	4 BHK Independent House	Old Bowenpally, Hyderabad	16,500	1000	Unfurnished	4	Independent House
1	1	2 BHK Independent House	Khairatabad Road, Hyderabad	14,000	700	Semi-Furnished	2	Independent House
2	2	2 BHK Independent House	Shaikpet, Hyderabad	12,000	1200	Unfurnished	2	Independent House
3	3	5 BHK Villa	Nallagandla Gachibowli, Hyderabad	80,000	4000	Semi-Furnished	5	Villa
4	4	5 BHK Villa	Moinabad, Hyderabad	200000	7000	Furnished	5	Villa
...	...	...	...	...	...	...	...	...
995	995	3 BHK Apartment	Ramachandra Puram, Hyderabad	18,000	1300	Semi-Furnished	3	Apartment
996	996	3 BHK Apartment	Gajulramaram Kukatpally, Hyderabad	27,000	1620	Semi-Furnished	3	Apartment
997	997	3 BHK Apartment	Krishna Reddy Pet, Hyderabad	18,000	1360	Semi-Furnished	3	Apartment
998	998	3 BHK Apartment	Pocharam, Hyderabad	12,000	1500	Semi-Furnished	3	Apartment
999	999	2 BHK Independent House	Kothapet, Hyderabad	13,500	1400	Unfurnished	2	Independent House

Again, if we see the DataFrame, Location Consists of information about Locality and City. So that, we split the Location Column into Two Columns namely, Locality and City Columns.

```
# split the Location Column into two Columns (i.e., Area, City)
df[["Locality", "City"]] = df["Location"].str.split(",", expand=True)
df
```

	Unnamed: 0	Title	Location	Price	Area	Status	BHK	Type	Locality	City
0	0	4 BHK Independent House	Old Bowenpally, Hyderabad	16,500	1000	Unfurnished	4	Independent House	Old Bowenpally	Hyderabad
1	1	2 BHK Independent House	Khairatabad Road, Hyderabad	14,000	700	Semi-Furnished	2	Independent House	Khairatabad Road	Hyderabad
2	2	2 BHK Independent House	Shaikpet, Hyderabad	12,000	1200	Unfurnished	2	Independent House	Shaikpet	Hyderabad
3	3	5 BHK Villa	Nallagandla Gachibowli, Hyderabad	80,000	4000	Semi-Furnished	5	Villa	Nallagandla Gachibowli	Hyderabad
4	4	5 BHK Villa	Moinabad, Hyderabad	200000	7000	Furnished	5	Villa	Moinabad	Hyderabad
...	...	...	...	...	...	...	...	...	...	...
995	995	3 BHK Apartment	Ramachandra Puram, Hyderabad	18,000	1300	Semi-Furnished	3	Apartment	Ramachandra Puram	Hyderabad
996	996	3 BHK Apartment	Gajulramaram Kukatpally, Hyderabad	27,000	1620	Semi-Furnished	3	Apartment	Gajulramaram Kukatpally	Hyderabad
997	997	3 BHK Apartment	Krishna Reddy Pet, Hyderabad	18,000	1360	Semi-Furnished	3	Apartment	Krishna Reddy Pet	Hyderabad
998	998	3 BHK Apartment	Pocharam, Hyderabad	12,000	1500	Semi-Furnished	3	Apartment	Pocharam	Hyderabad
999	999	2 BHK Independent House	Kothapet, Hyderabad	13,500	1400	Unfurnished	2	Independent House	Kothapet	Hyderabad

1000 rows × 10 columns

Now, we rename the Price Column to the Rent in Rs Column.

```
# Renaming the Price column into Rent in Rs column
df.rename(columns = {'Price': 'Rent in Rs'}, inplace = True)
```

```
df
```

	Unnamed: 0	Title	Location	Rent in Rs	Area	Status	BHK	Type	Locality	City
0	0	4 BHK Independent House	Old Bowenpally, Hyderabad	16,500	1000	Unfurnished	4	Independent House	Old Bowenpally	Hyderabad
1	1	2 BHK Independent House	Khairatabad Road, Hyderabad	14,000	700	Semi-Furnished	2	Independent House	Khairatabad Road	Hyderabad
2	2	2 BHK Independent House	Shaikpet, Hyderabad	12,000	1200	Unfurnished	2	Independent House	Shaikpet	Hyderabad
3	3	5 BHK Villa	Nallagandla Gachibowli, Hyderabad	80,000	4000	Semi-Furnished	5	Villa	Nallagandla Gachibowli	Hyderabad
4	4	5 BHK Villa	Moinabad, Hyderabad	200000	7000	Furnished	5	Villa	Moinabad	Hyderabad
...	...	...	...	...	...	...	...	...	...	...
995	995	3 BHK Apartment	Ramachandra Puram, Hyderabad	18,000	1300	Semi-Furnished	3	Apartment	Ramachandra Puram	Hyderabad
996	996	3 BHK Apartment	Gajulramaram Kukatpally, Hyderabad	27,000	1620	Semi-Furnished	3	Apartment	Gajulramaram Kukatpally	Hyderabad
997	997	3 BHK Apartment	Krishna Reddy Pet, Hyderabad	18,000	1360	Semi-Furnished	3	Apartment	Krishna Reddy Pet	Hyderabad
998	998	3 BHK Apartment	Pocharam, Hyderabad	12,000	1500	Semi-Furnished	3	Apartment	Pocharam	Hyderabad
999	999	2 BHK Independent House	Kothapet, Hyderabad	13,500	1400	Unfurnished	2	Independent House	Kothapet	Hyderabad

1000 rows × 10 columns

If we Observe the DataFrame, Title and Location Columns are not necessary in the DataFrame. So, we drop those Two Columns.

```
# Dropping the Title and Location Columns in a DF
df.drop(["Title", "Location"], axis = 1, inplace = True)
```

```
df
```

	Unnamed: 0	Rent in Rs	Area	Status	BHK	Type	Locality	City
0	0	16,500	1000	Unfurnished	4	Independent House	Old Bowenpally	Hyderabad
1	1	14,000	700	Semi-Furnished	2	Independent House	Khairatabad Road	Hyderabad
2	2	12,000	1200	Unfurnished	2	Independent House	Shaikpet	Hyderabad
3	3	80,000	4000	Semi-Furnished	5	Villa	Nallagandla Gachibowli	Hyderabad
4	4	200000	7000	Furnished	5	Villa	Moinabad	Hyderabad
...	...	...	...	...	...	...	...	...
995	995	18,000	1300	Semi-Furnished	3	Apartment	Ramachandra Puram	Hyderabad
996	996	27,000	1620	Semi-Furnished	3	Apartment	Gajulramaram Kukatpally	Hyderabad
997	997	18,000	1360	Semi-Furnished	3	Apartment	Krishna Reddy Pet	Hyderabad
998	998	12,000	1500	Semi-Furnished	3	Apartment	Pocharam	Hyderabad
999	999	13,500	1400	Unfurnished	2	Independent House	Kothapet	Hyderabad

1000 rows × 8 columns

Now, we need to remove unnecessary characters or text in Rent in Rs Column. Therefore, we replace the comma (,) with no space and then assign that updated Rent values to Rent in Rs Column in a DataFrame.

```
# Assigning the updated values of Rent in Rs column to the Rent in Rs column in a DataFrame
Rent = df["Rent in Rs"].str.replace(",","")
df['Rent in Rs'] = Rent
```

```
df
```

	Unnamed: 0	Rent in Rs	Area	Status	BHK	Type	Locality	City
0	0	16500	1000	Unfurnished	4	Independent House	Old Bowenpally	Hyderabad
1	1	14000	700	Semi-Furnished	2	Independent House	Khairatabad Road	Hyderabad
2	2	12000	1200	Unfurnished	2	Independent House	Shaikpet	Hyderabad
3	3	80000	4000	Semi-Furnished	5	Villa	Nallagandla Gachibowli	Hyderabad
4	4	200000	7000	Furnished	5	Villa	Moinabad	Hyderabad
...	...	...	...	...	...	...	...	...
995	995	18000	1300	Semi-Furnished	3	Apartment	Ramachandra Puram	Hyderabad
996	996	27000	1620	Semi-Furnished	3	Apartment	Gajulramaram Kukatpally	Hyderabad
997	997	18000	1360	Semi-Furnished	3	Apartment	Krishna Reddy Pet	Hyderabad
998	998	12000	1500	Semi-Furnished	3	Apartment	Pocharam	Hyderabad
999	999	13500	1400	Unfurnished	2	Independent House	Kothapet	Hyderabad

1000 rows × 8 columns

Now, arrange the Order of Column names in a DataFrame as we like and assign to new variable called newdf.

```
# arranging the order of Columns
newdf = df[['Locality', 'City', 'Status','BHK','Type', 'Area', 'Rent in Rs']]
```

```
newdf
```

	Locality	City	Status	BHK	Type	Area	Rent in Rs
0	Old Bowenpally	Hyderabad	Unfurnished	4	Independent House	1000	16500
1	Khairatabad Road	Hyderabad	Semi-Furnished	2	Independent House	700	14000
2	Shaikpet	Hyderabad	Unfurnished	2	Independent House	1200	12000
3	Nallagandla Gachibowli	Hyderabad	Semi-Furnished	5	Villa	4000	80000
4	Moinabad	Hyderabad	Furnished	5	Villa	7000	200000
...	...	...	...	...	...	...	...
995	Ramachandra Puram	Hyderabad	Semi-Furnished	3	Apartment	1300	18000
996	Gajulramaram Kukatpally	Hyderabad	Semi-Furnished	3	Apartment	1620	27000
997	Krishna Reddy Pet	Hyderabad	Semi-Furnished	3	Apartment	1360	18000
998	Pocharam	Hyderabad	Semi-Furnished	3	Apartment	1500	12000
999	Kothapet	Hyderabad	Unfurnished	2	Independent House	1400	13500

1000 rows × 7 columns

Now, if we want to know all the information about the DataFrame, we can use info () function. So that, we get to know about the data type of each column in a DataFrame, number of Columns in a DataFrame, Memory Usage, number Null values and non-Null values in each column.

```
]:  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   1000 non-null   int64
 1   Rent in Rs   1000 non-null   object
 2   Area         1000 non-null   int64
 3   Status       1000 non-null   object
 4   BHK          1000 non-null   object
 5   Type         1000 non-null   object
 6   Locality     1000 non-null   object
 7   City         1000 non-null   object
dtypes: int64(2), object(6)
memory usage: 62.6+ KB
```

If we see the Datatype of Rent in Rs Column is in Object datatype. So that, we need to convert Object datatype into float datatype.

```
]:  # Converting Object Datatype into float Datatype of Price Columns
    newdf["Rent in Rs"] = newdf["Rent in Rs"].astype("float")
    newdf.dtypes

C:\Users\mouni\AppData\Local\Temp\ipykernel_20108\1506414328.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy
  newdf["Rent in Rs"] = newdf["Rent in Rs"].astype("float")
]:  Locality       object
    City           object
    Status         object
    BHK            object
    Type           object
    Area            int64
    Rent in Rs    float64
    dtype: object

        or

]:  newdf["Rent in Rs"] = pd.to_numeric(newdf["Rent in Rs"])
    newdf.dtypes

C:\Users\mouni\AppData\Local\Temp\ipykernel_20108\2172267712.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy
  newdf["Rent in Rs"] = pd.to_numeric(newdf["Rent in Rs"])
]:  Locality       object
    City           object
    Status         object
    BHK            object
    Type           object
    Area            int64
    Rent in Rs    float64
    dtype: object
```

After changing the datatype of Rent in Rs Column into float, to check again the datatypes of each column in a DataFrame, we are going to use info () function.

```
]:  # Displays all the information of the Data in a DataFrame
    newdf.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Locality     1000 non-null   object
 1   City         1000 non-null   object
 2   Status       1000 non-null   object
 3   BHK          1000 non-null   object
 4   Type         1000 non-null   object
 5   Area         1000 non-null   int64
 6   Rent in Rs   1000 non-null   float64
dtypes: float64(1), int64(1), object(5)
memory usage: 54.8+ KB
```

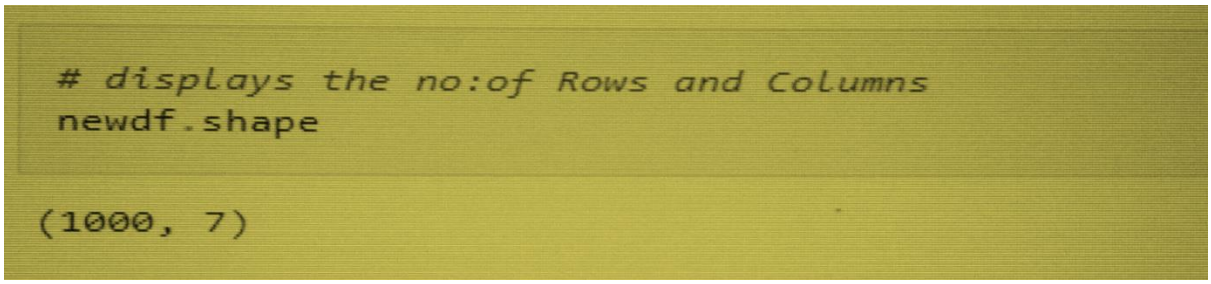
Now, we need to check the number of Null Values in each Column. We will use isnull (). Sum () function. If in case any missing values are exist in any Column, we need to impute them with some value. For Categorical feature, we will impute the Missing values with Mode whereas for Numerical feature, we will impute the Null Values with the Mean or Median.

```
# checking the Count of Null Values in each Column of a Dataset
newdf.isnull().sum()

Locality      0
City          0
Status        0
BHK           0
Type          0
Area          0
Rent in Rs    0
dtype: int64
```

By seeing above picture, we can say that there are no null values in each column. Each Column shows zero null values count.

Step – 4: Data Exploration

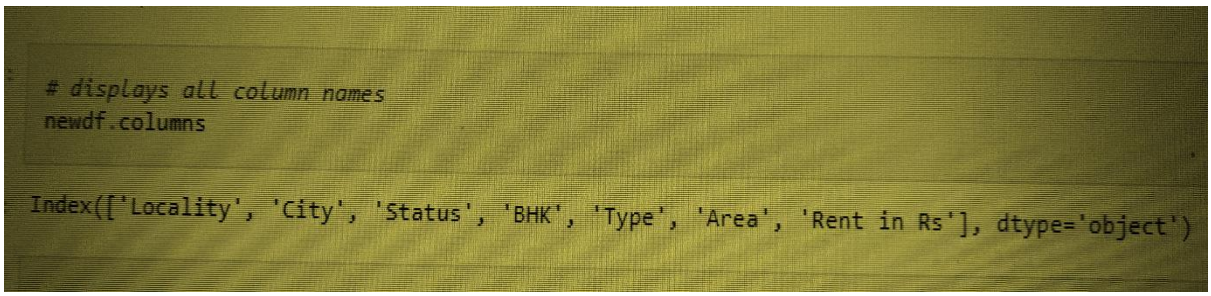
After completion of Data Cleaning, now we check the number of Rows and Columns of a DataFrame.



```
# displays the no:of Rows and Columns
newdf.shape

(1000, 7)
```

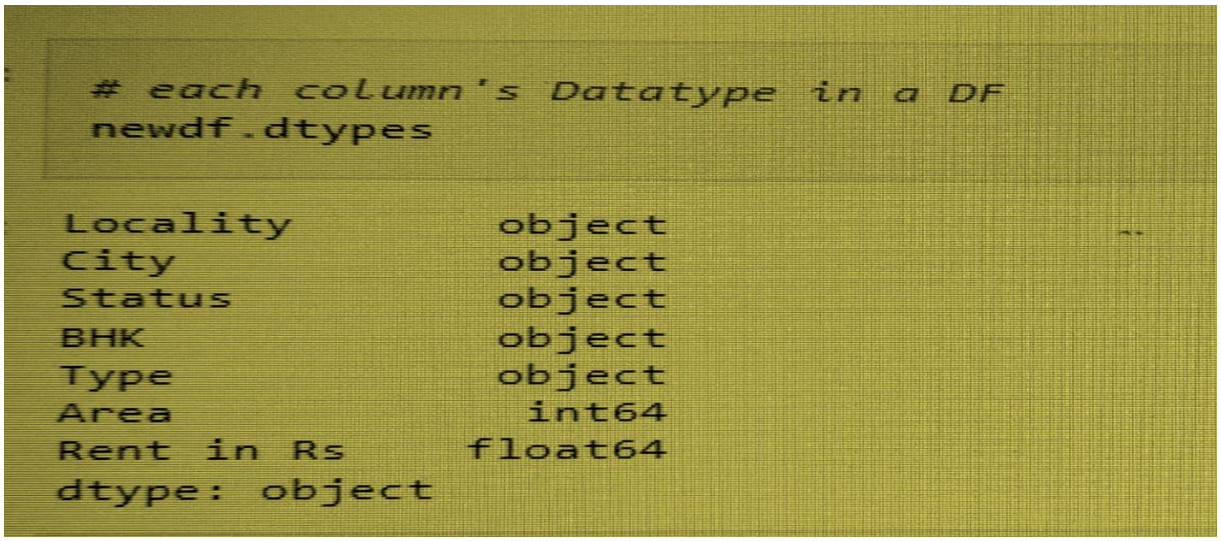
It shows that, DataFrame contains Thousand Rows and Seven Columns.

If we want to know all the Column Names in a DataFrame, we will use columns attribute.



```
# displays all column names
newdf.columns

Index(['Locality', 'City', 'Status', 'BHK', 'Type', 'Area', 'Rent in Rs'], dtype='object')
```

If we want to know the datatypes of each Column, we will use dtypes attribute.



```
# each column's Datatype in a DF
newdf.dtypes

Locality        object
City            object
Status          object
BHK             object
Type            object
Area             int64
Rent in Rs     float64
dtype: object
```

To Return a Series of Rent in Rs column's unique values count. We can use value_counts () function.

```
# Return a Series containing counts of unique values
newdf['Rent in Rs'].value_counts()
```

```
18000.0     200
12000.0     100
16500.0      50
14000.0      50
80000.0      50
200000.0     50
10000.0      50
17000.0      50
16000.0      50
15000.0      50
20000.0      50
40000.0      50
8000.0       50
150000.0     50
27000.0      50
13500.0      50
Name: Rent in Rs, dtype: int64
```

If we want to know the Description of the Data in a DataFrame it means mean, standard deviation, Percentiles, minimum and maximum values. We will use describe () function.

```
# gives description of the Data in a DataFrame
newdf.describe()
```

	Area	Rent in Rs
count	1000.000000	1000.000000
mean	2157.500000	36150.000000
std	2250.823173	49350.663501
min	700.000000	8000.000000
25%	1075.000000	13875.000000
50%	1450.000000	17500.000000
75%	1700.000000	21750.000000
max	9900.000000	200000.000000

We can say that Area and Rent in Rs both Columns are Numerical Columns in a DataFrame and remaining all columns are Categorical Variables.

We can know the minimum Rent of House by using min () function.

```
# minimum Price of House for rent
newdf['Rent in Rs'].min()

8000.0
```

After analyzing the Rent, Minimum Rent of House in Hyderabad is 8000 Rs.

If we want to know the maximum Rent of House in Hyderabad, we will use max () function.

```
# maximum Price of House for rent
newdf['Rent in Rs'].max()

200000.0
```

Maximum Rent of House is 2 Lacks in Hyderabad City.

If we want an average Rent of House in Hyderabad, we can use mean () function.

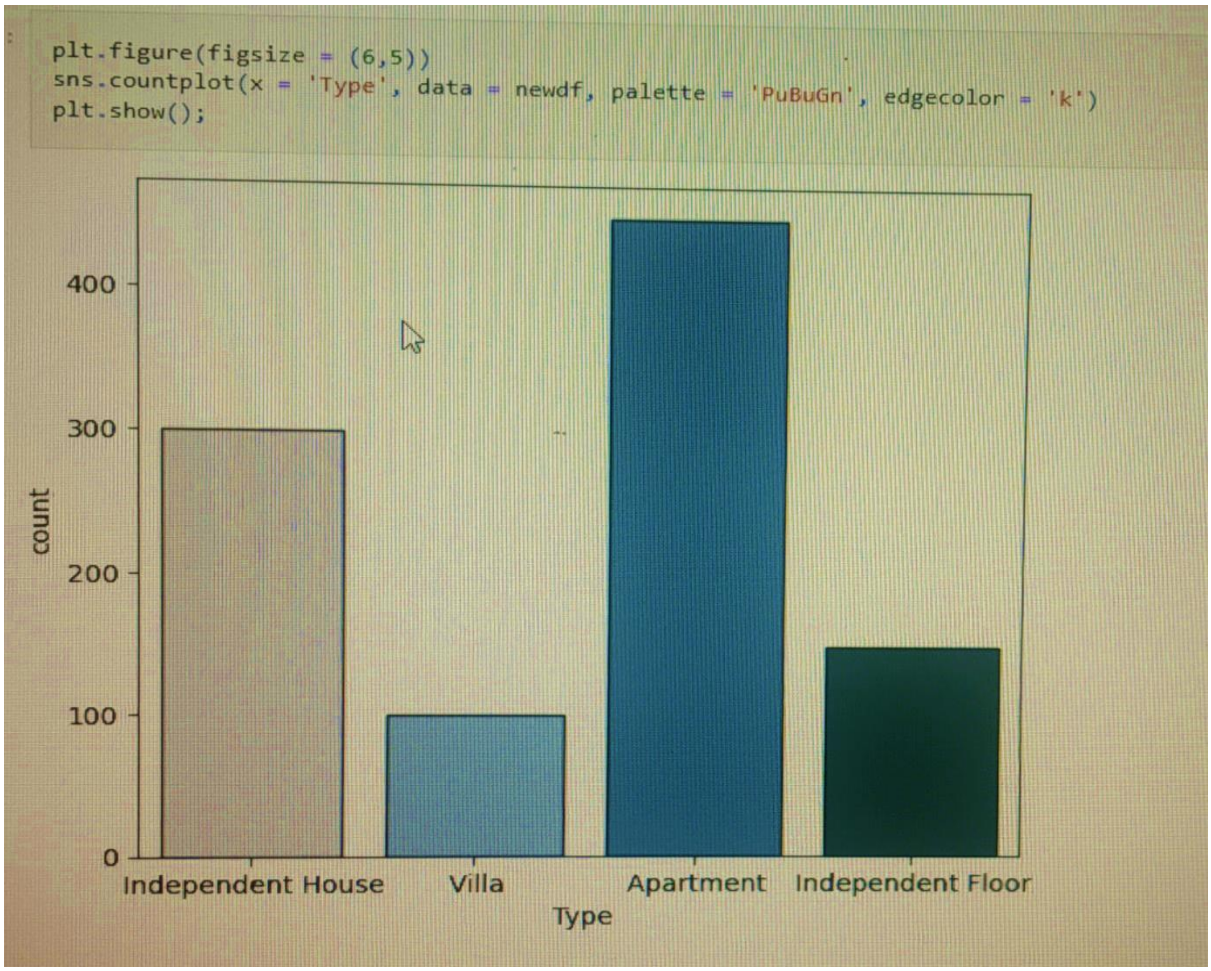
```
# Average Price Of House for Rent
newdf['Rent in Rs'].mean()

36150.0
```

Average Rent of House in Hyderabad City is 36,150.

Step – 5: Data Visualization using Matplotlb and Seaborn Libraries

Univariate Analysis:

If we want to know which type of Houses are Present more in Hyderabad, we can plot seaborn count plot by taking Type Column in X – Axis.

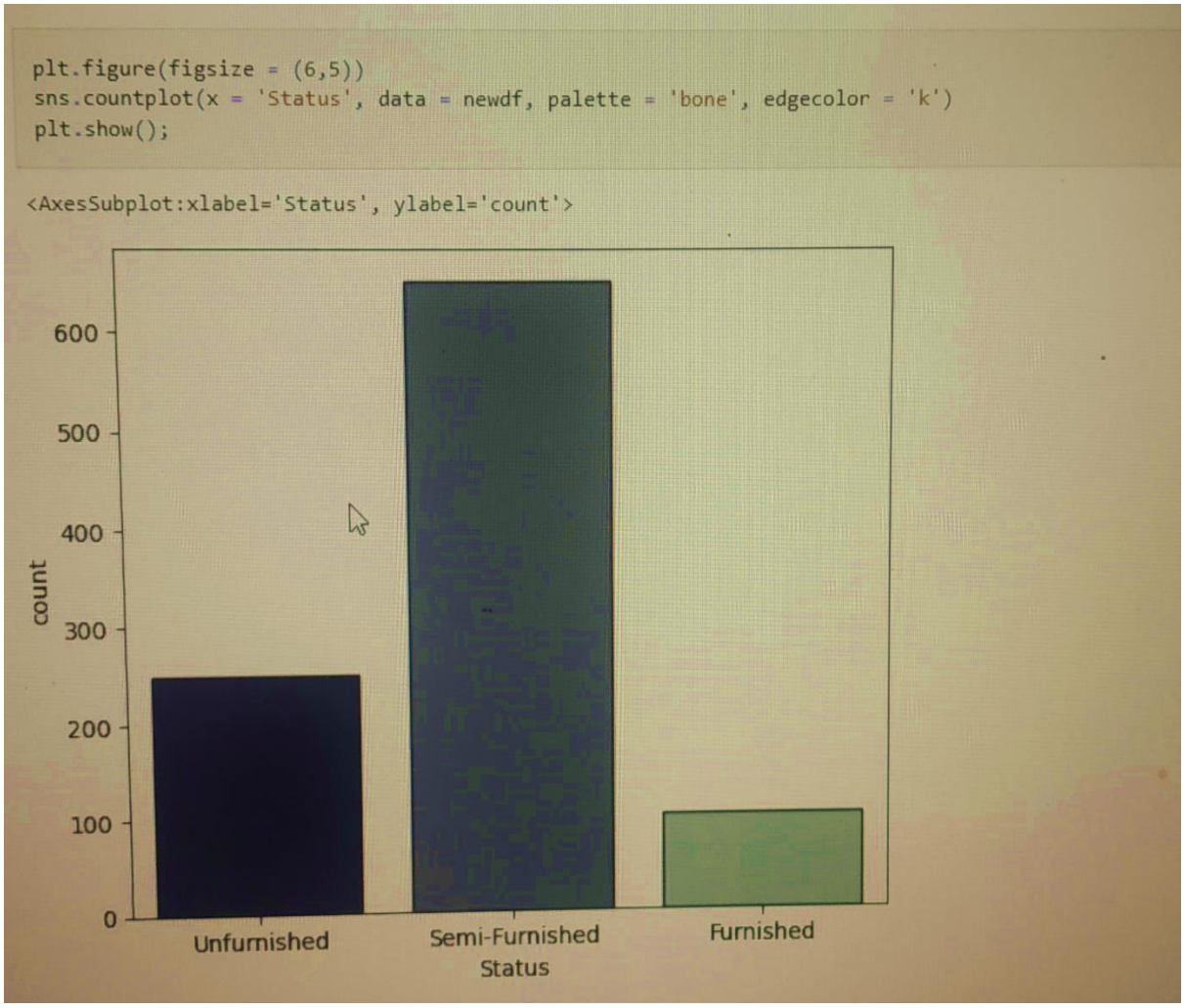


After plotting the count plot, we get some insights from that plot.

Insights:

1. Apartment Type Houses are higher in number which are 450 available for Rent as compared to other type of Houses.
2. Villa Type Houses are less in number which are 100 as compared to other type of Houses.

If we want to know House Status, plot the seaborn count plot for Status Variable by taking Status variable in X – Axis. Count plot counts number of Houses in each category.

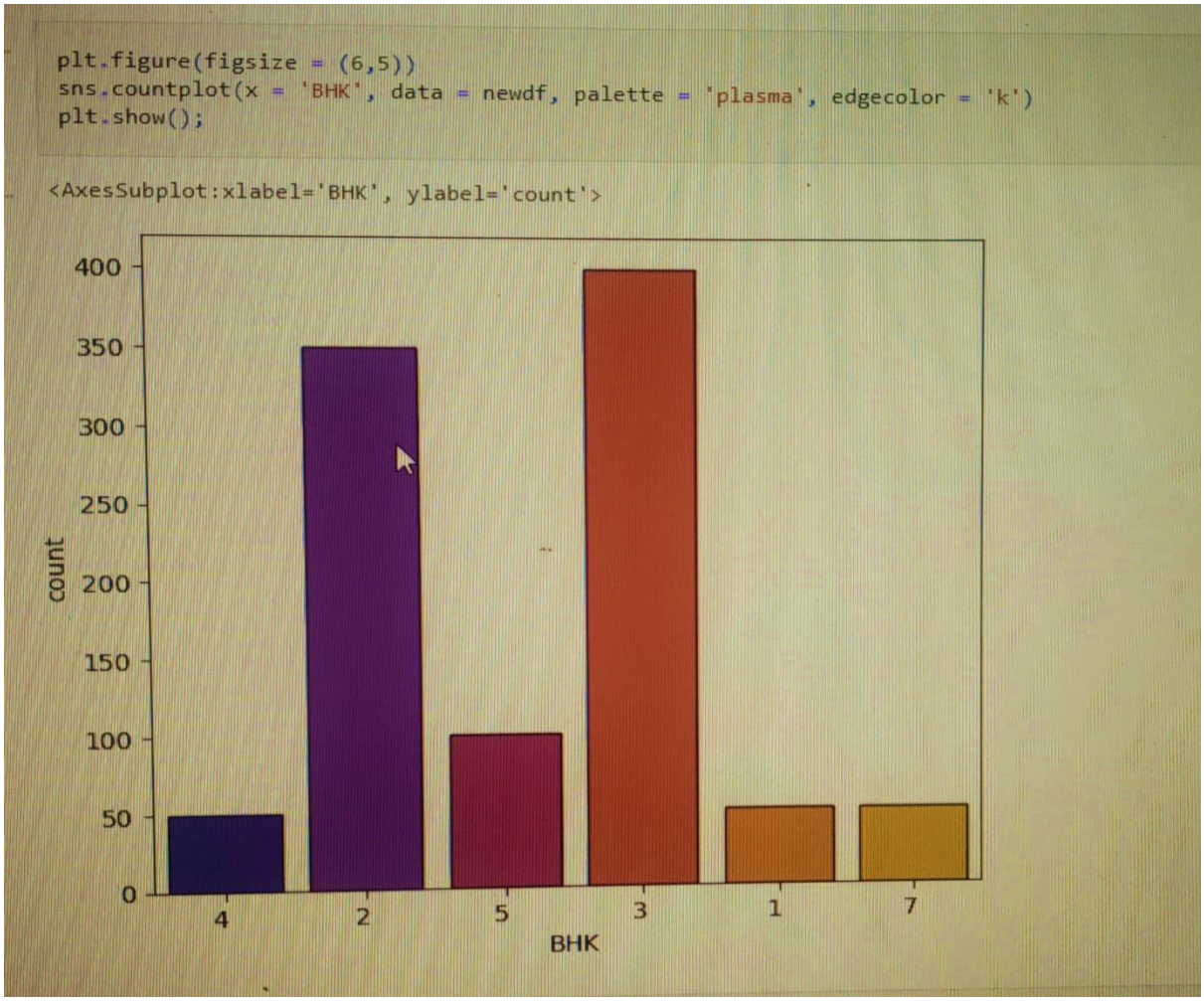


If we observe in the plot, we can say that Semi-Furniture Houses are more in number are available for Rent in Hyderabad.

Insights:

1. Semi-Furniture Houses are higher in number which are 650 available for Rent in Hyderabad City.
2. Furniture Houses are less in number as compare to All. count is 100 are available for rent in Hyderabad City.

If we want check Houses which are having different types of BHK's are more in number (or) less available in Hyderabad City, we should plot countplot and take BHK Variable in X – Axis.



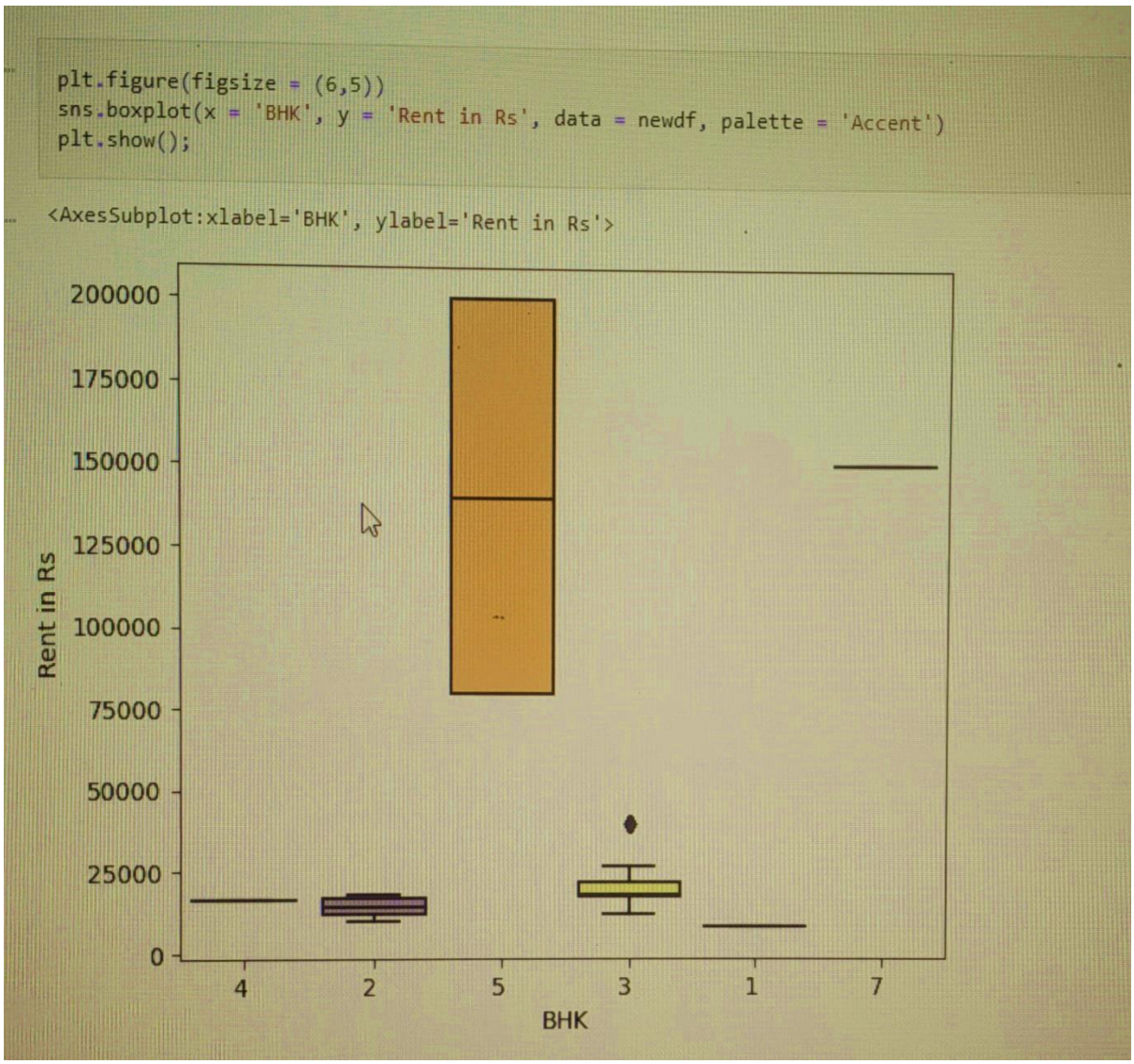
After observing the countplot, we get some insights.

Insights:

1. 3BHK Houses are more in number which are 400 as compared to other BHK are available for Rent in Hyderabad City.
2. 1BHK, 4BHK and 7BHK Houses are same in number and less in number as compared to other BHK and count is 50 are available in Hyderabad City for Rent.

Bivariate Analysis:

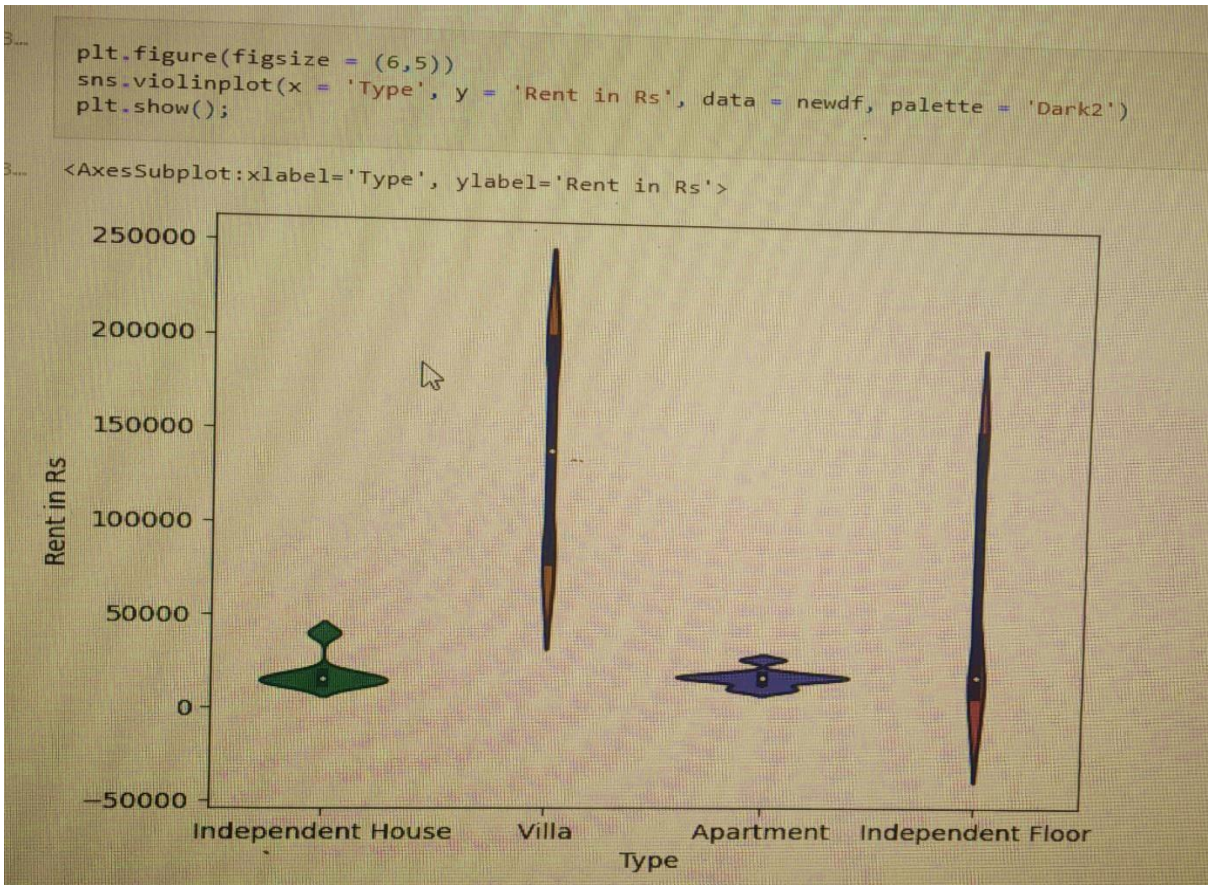
We can Strongly say that, Types of BHK is the key factor for the Rent of the House. So that, if we want to see the Rent for each BHK's in Hyderabad City, we plot the box plot and take BHK variable in X – Axis and Rent in Rs Variable in Y – Axis.



Insights:

1. Rent for 5 BHK House is higher than the other BHK's Rent. 25% to 75% of the Rent for 5 BHK is in between 75,000 and 2 Lacks.
2. 1 BHK and 4 BHK Houses are less in number and also Rent is Low.
3. 2 BHK and 3 BHK Houses are available for Rent below 25000 Rs only.
4. 7 BHK Houses are less in number but the Rent is approximately equal to 1,50,000.

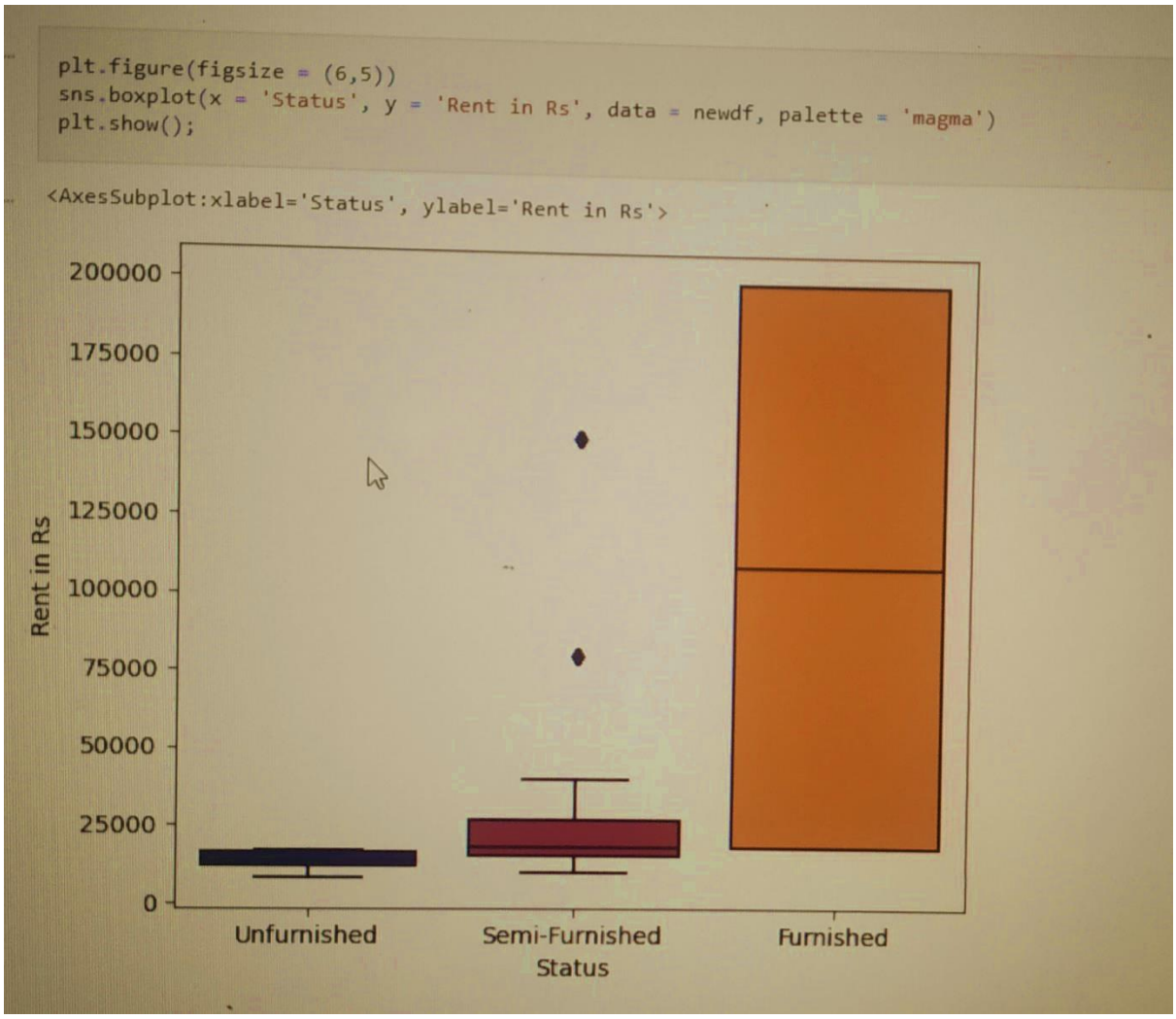
We can also say that the Rent of the House depends on Types of Houses. So that, if we want to see the Rent for each Type of House in Hyderabad City, we plot the box plot and take Type variable in X – Axis and Rent in Rs Variable in Y – Axis.



Insights:

1. Villa Type Houses have More Rent and Starting from approximately 50,000 to 2.5 Lacks.
2. Apartments have Less Rent available in Hyderabad City as compared to other Types of Houses and we can say somewhat cheap cost.
3. After Apartments, Independent Houses are Best for Rent are available in Hyderabad City.
4. Up to 2 Lacks Independent Floor Houses are available for Rent in Hyderabad City.

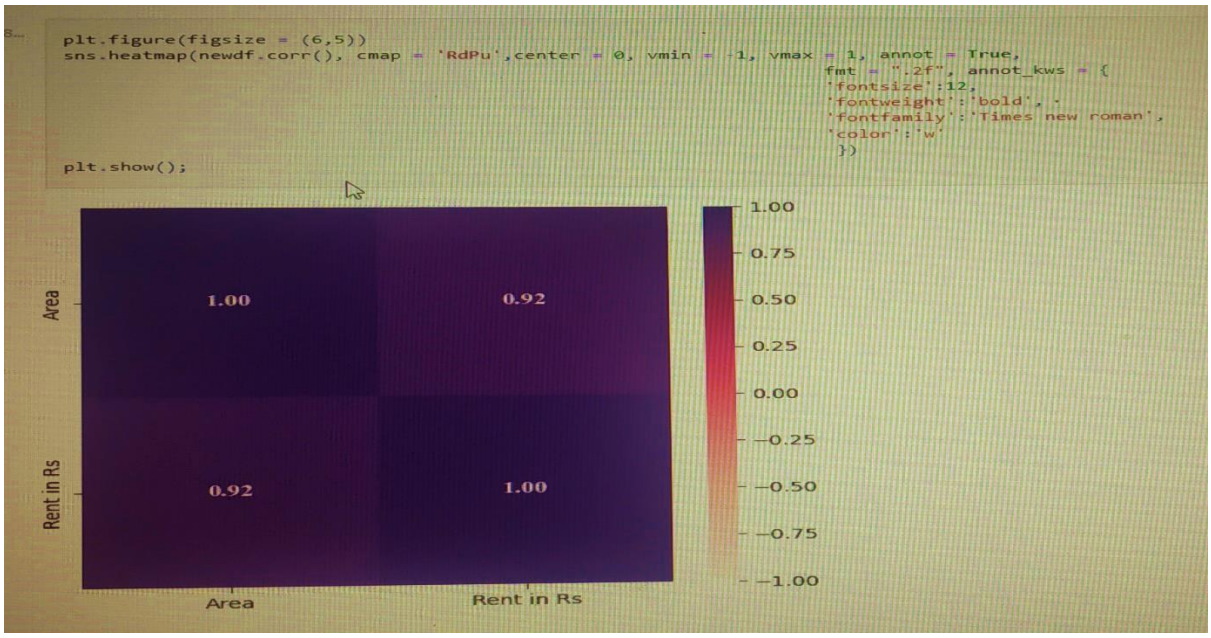
If we want to know the Rent for different types of Status of House, we plot box plot by taking Status Variable in X – Axis and Rent in Rs Variable in Y – Axis.



After Plotting the Box plot, we get some insights.

Insights:

1. Furniture Houses are available for Rent in all ranges of Cost. From approximately 15,000 to 2 lacks Furniture Houses are available for rent in Hyderabad City.
2. Below 50,000 Rs Semi-Furniture Houses are available for Rent in Hyderabad City.
3. Below 25,000 Rs Furniture Houses are available for Rent in Hyderabad City.



Correlation between Area and Rent is about 0.92 it means more positively correlated they are; that is as one increases so does the other and closer to 1 the stronger the relationship is.