INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

APU2F2209IT(BIS)

NAME: Lim Yau Jade

TP NUMBER: TP065086

COURSEWORK TITLE: House Rent Dataset

HAND IN DATE: 2 DECEMBER 2022

WEIGHTTAGE: 50%

LECTURER NAME: Mrs. Minnu Helen Joseph

Table of Contents

Tables of Figures

1.0 Introduction

India, which is in South Asia, has the second highest population in the world. The house rents in India are different according to the culture, location, surroundings, lifestyle and family background. There are six cities in the dataset, which are Mumbai, Chennai, Bangalore, Hyderabad, Delhi, and Kolkata. There are 12 columns and 4746 rows of data provided by the Human Rights Measurement team, where it requires to analysis and research.

1.1 Assumption

To identify the influence of geographical location, house conditions, human relationship and lifestyle on house renting, the writer has made several assumptions.

Questions of the datasets:

1. How does geographical location affect the house rent?
2. How does the house condition affect the house rent?
3. How does the relationship between the tenant and owner affect the house rent?
4. What is the best day to rent a house?

2.0 Data Import, Cleaning, Pre-processing, Data Exploration, Additional Function

2.1 Data Import

2.1.1 Load Package

First, all the packages that are used to analyze the data will be loaded beforehand by using the library() function.

```
#load package
library(ggplot2)
library(crayon)
library(dplyr)
library(scales)
library(plotrix)
library(tidyverse)
```

Figure 1: Load Packages

2.1.2 Import Dataset

The house rent dataset is imported from the "House_Rent_Dataset.csv" file by using read.csv() function. The dataset contains header thus the header will be "TRUE" while the first row will be displayed as columns' header.

```
#import dataset
rent = read.csv("C:\\Users\\kelly\\Desktop\\APU Year 2\\PFDA\\Assignment\\House_Rent_Dataset.csv",
                header=TRUE)

rent
```

Figure 2: Code to Import House_Rent_Dataset with read.csv()

By typing the dataset variables, which is “rent”, the dataset will be printed out on the console.

```
> rent
   Posted.On BHK  Rent Size           Floor   Area.Type                     Area.Locality    City
1   5/18/2022   2 10000 1100 Ground out of 2  Super Area                           Bandel Kolkata
2   5/13/2022   2 20000  800      1 out of 3  Super Area          Phool Bagan, Kankurgachi Kolkata
3   5/16/2022   2 17000 1000      1 out of 3  Super Area           Salt Lake City Sector 2 Kolkata
4    7/4/2022   2 10000  800      1 out of 2  Super Area                       Dumdum Park Kolkata
5    5/9/2022   2  7500  850      1 out of 2 Carpet Area                  South Dum Dum Kolkata
6   4/29/2022   2  7000  600 Ground out of 1  Super Area                     Thakurpukur Kolkata
7   6/21/2022   2 10000  700 Ground out of 4  Super Area                        Malancha Kolkata
8   6/21/2022   1  5000  250      1 out of 2  Super Area                        Malancha Kolkata
9    6/7/2022   2 26000  800      1 out of 2 Carpet Area      Palm Avenue Kolkata, Ballygunge Kolkata
10 6/20/2022   2 10000 1000      1 out of 3 Carpet Area                        Natunhat Kolkata
11 5/23/2022   3 25000 1200      1 out of 4 Carpet Area      Action Area 1, Rajarhat Newtown Kolkata
12   6/7/2022  1  5000  400      1 out of 1 Carpet Area                      Keshtopur Kolkata
13 5/14/2022   1  6500  250      1 out of 4 Carpet Area               Tarulia, Keshtopur Kolkata
14   5/9/2022  1  5500  375      1 out of 2 Carpet Area                  Dum Dum Metro Kolkata
15   5/5/2022  3  8500  900 Ground out of 2 Carpet Area                Paschim Barisha Kolkata
16   6/1/2022  3 40000 1286      1 out of 1 Carpet Area          New Town Action Area 1 Kolkata
17 5/17/2022   2  6000  600      1 out of 2  Super Area                        Barasat Kolkata
18 6/20/2022   2 10000  800 Ground out of 2  Super Area                         Behala Kolkata
19   6/9/2022  2 11000 2000 Ground out of 3 Carpet Area                 Behala Chowrasta Kolkata
20   6/9/2022  2  6000  660      1 out of 2  Super Area                         Behala Kolkata
21   7/2/2022  2  7900  650      1 out of 2 Carpet Area                     Santoshpur Kolkata
22 6/14/2022   2  9000  400      2 out of 3 Carpet Area            Garia Station, Garia Kolkata
23 6/15/2022   1  4000  300 Ground out of 4 Carpet Area            Garia Station, Garia Kolkata
24 6/15/2022   3  6500 1600 Ground out of 2  Super Area                             Joka Kolkata
25 5/28/2022   1  8000  400      1 out of 2  Super Area                       Sreebhumi Kolkata
26 5/22/2022   2  7000 1000      1 out of 1  Super Area                        Rajarhat Kolkata
27 6/18/2022   1  5300  355      1 out of 1 Carpet Area                         Dum Dum Kolkata
28 6/25/2022   2  6000 1000 Ground out of 3  Super Area          Kodalia, Hooghly-Chinsurah Kolkata
29 6/22/2022   2  8500  800      4 out of 5  Super Area                         Baguiati Kolkata
30 6/25/2022   2 12500  850 Ground out of 2  Super Area      Rabindra Sarobar Area, Dhakuria Kolkata
```

```
   Furnishing.Status Tenant.Preferred Bathroom Point.of.Contact
1        Unfurnished Bachelors/Family        2    Contact Owner
2    Semi-Furnished Bachelors/Family        1    Contact Owner
3    Semi-Furnished Bachelors/Family        1    Contact Owner
4        Unfurnished Bachelors/Family        1    Contact Owner
5        Unfurnished        Bachelors        1    Contact Owner
6        Unfurnished Bachelors/Family        2    Contact Owner
7        Unfurnished        Bachelors        2    Contact Agent
8        Unfurnished        Bachelors        1    Contact Agent
9        Unfurnished        Bachelors        2    Contact Agent
10   Semi-Furnished Bachelors/Family        2    Contact Owner
11   Semi-Furnished Bachelors/Family        2    Contact Agent
12       Unfurnished Bachelors/Family        1    Contact Agent
13         Furnished        Bachelors        1    Contact Owner
14       Unfurnished Bachelors/Family        1    Contact Agent
15       Unfurnished        Bachelors        2    Contact Owner
16         Furnished Bachelors/Family        2    Contact Owner
17   Semi-Furnished Bachelors/Family        1    Contact Owner
18       Unfurnished Bachelors/Family        1    Contact Owner
19       Unfurnished Bachelors/Family        1    Contact Owner
20       Unfurnished Bachelors/Family        1    Contact Owner
21       Unfurnished           Family        1    Contact Owner
22       Unfurnished        Bachelors        2    Contact Owner
23       Unfurnished Bachelors/Family        1    Contact Owner
24       Unfurnished Bachelors/Family        1    Contact Owner
25   Semi-Furnished        Bachelors        1    Contact Agent
26       Unfurnished Bachelors/Family        1    Contact Owner
27   Semi-Furnished Bachelors/Family        1    Contact Agent
28   Semi-Furnished Bachelors/Family        1    Contact Owner
29       Unfurnished Bachelors/Family        1    Contact Owner
30       Unfurnished Bachelors/Family        2    Contact Owner
```

Figure 3: Output of rent Dataset on the console

2.1.3 View Dataset

By using the view () function, the dataset could be view in table form.

```
#view all data
View(rent)
```

Figure 4: Code to View All the Data in Table Format

	Posted.On	BHK	Rent	Size	Floor	Area.Type	Area.Locality	City	Furnishing.Status	Tenant.Preferred	Bathroom	Point.of.Contact
1	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
2	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	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
4	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors	1	Contact Owner
5	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
6	4/29/2022	2	7000	600	Ground out of 1	Super Area	Thakurpukur	Kolkata	Unfurnished	Bachelors	2	Contact Agent
7	6/21/2022	2	10000	700	Ground out of 4	Super Area	Malancha	Kolkata	Unfurnished	Bachelors	1	Contact Agent
8	6/21/2022	1	5000	250	1 out of 2	Super Area	Malancha	Kolkata	Unfurnished	Bachelors	2	Contact Agent
9	6/7/2022	2	26000	800	1 out of 2	Carpet Area	Palm Avenue Kolkata, Ballygunge	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
10	6/20/2022	2	10000	1000	1 out of 3	Carpet Area	Naturhat	Kolkata	Semi-Furnished	Bachelors/Family	2	Contact Agent
11	5/23/2022	3	25000	1200	1 out of 4	Carpet Area	Action Area 1, Rajarhat Newtown	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Agent
12	6/7/2022	1	5000	400	1 out of 1	Carpet Area	Keshtopur	Kolkata	Unfurnished	Bachelors	1	Contact Owner
13	5/14/2022	1	6500	250	1 out of 4	Carpet Area	Tarulia, Keshtopur	Kolkata	Furnished	Bachelors/Family	1	Contact Agent
14	5/9/2022	1	5500	375	1 out of 2	Carpet Area	Dum Dum Metro	Kolkata	Unfurnished	Bachelors	2	Contact Owner
15	5/5/2022	3	8500	900	Ground out of 2	Carpet Area	Paschim Barisha	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
16	6/1/2022	3	40000	1286	1 out of 1	Carpet Area	New Town Action Area 1	Kolkata	Furnished	Bachelors/Family	2	Contact Agent
17	5/17/2022	2	6000	600	1 out of 2	Super Area	Barasat	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Agent
18	6/20/2022	2	10000	800	Ground out of 2	Super Area	Behala	Kolkata	Unfurnished	Bachelors	1	Contact Owner
19	6/9/2022	2	11000	2000	Ground out of 3	Carpet Area	Behala Chowrasta	Kolkata	Unfurnished	Bachelors/Family	1	Contact Agent
20	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors	2	Contact Owner
21	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	2	Contact Owner
22	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
23	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
24	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	2	Contact Owner
25	4/29/2022	2	7000	600	Ground out of 1	Super Area	Thakurpukur	Kolkata	Unfurnished	Bachelors	1	Contact Owner
26	6/21/2022	2	10000	700	Ground out of 4	Super Area	Malancha	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
27	6/21/2022	1	5000	250	1 out of 2	Super Area	Malancha	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
28	6/7/2022	2	26000	800	1 out of 2	Carpet Area	Palm Avenue Kolkata, Ballygunge	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
29	6/20/2022	2	10000	1000	1 out of 3	Carpet Area	Naturhat	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
30	5/23/2022	3	25000	1200	1 out of 4	Carpet Area	Action Area 1, Rajarhat Newtown	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner

Figure 5: Table Format of the Rent Dataset

Using head() function, the first six row of the dataset will be printed on the console.

```
#View the first six rows
head(rent)
```

Figure 6: Code to View the First Six Row

```
> head(rent)
  Posted.On BHK  Rent Size           Floor   Area.Type            Area.Locality    City Furnishing.Status
1 5/18/2022   2 10000 1100 Ground out of 2  Super Area                   Bandel Kolkata      Unfurnished
2 5/13/2022   2 20000  800      1 out of 3  Super Area Phool Bagan, Kankurgachi Kolkata    Semi-Furnished
3 5/16/2022   2 17000 1000      1 out of 3  Super Area  Salt Lake City Sector 2 Kolkata    Semi-Furnished
4  7/4/2022   2 10000  800      1 out of 2  Super Area             Dumdum Park Kolkata      Unfurnished
5  5/9/2022   2  7500  850      1 out of 2 Carpet Area           South Dum Dum Kolkata      Unfurnished
6 4/29/2022   2  7000  600 Ground out of 1  Super Area              Thakurpukur Kolkata      Unfurnished
  Tenant.Preferred Bathroom Point.of.Contact
1 Bachelors/Family        2    Contact Owner
2 Bachelors/Family        1    Contact Owner
3 Bachelors/Family        1    Contact Owner
4 Bachelors/Family        1    Contact Owner
5        Bachelors        1    Contact Owner
6 Bachelors/Family        2    Contact Owner
```

Figure 7: Output of the First Six Row on the Console

To view the last six row on the console, tail() function is used to have a brief view of the dataset.

```
#View the last six rows
tail(rent)
```

Figure 8: Code to View the Last Six Row

```
> tail(rent)
     Posted.On BHK  Rent Size         Floor   Area.Type       Area.Locality      City Furnishing.Status
4741  6/2/2022   2 12000 1350   2 out of 2  Super Area           Old Alwal Hyderabad      Unfurnished
4742 5/18/2022   2 15000 1000   3 out of 5 Carpet Area        Bandam Kommu Hyderabad    Semi-Furnished
4743 5/15/2022   3 29000 2000   1 out of 4  Super Area Manikonda, Hyderabad Hyderabad    Semi-Furnished
4744 7/10/2022   3 35000 1750   3 out of 5 Carpet Area Himayath Nagar, NH 7 Hyderabad    Semi-Furnished
4745  7/6/2022   3 45000 1500 23 out of 34 Carpet Area          Gachibowli Hyderabad    Semi-Furnished
4746  5/4/2022   2 15000 1000   4 out of 5 Carpet Area      Suchitra Circle Hyderabad      Unfurnished
     Tenant.Preferred Bathroom Point.of.Contact
4741 Bachelors/Family        2    Contact Owner
4742 Bachelors/Family        2    Contact Owner
4743 Bachelors/Family        3    Contact Owner
4744 Bachelors/Family        3    Contact Agent
4745           Family        2    Contact Agent
4746        Bachelors        2    Contact Owner
```

Figure 9: Output of the Last Six Row on the Console

2.2 Cleaning

2.2.1 Detect Missing Value

There are two ways to identify the presence of the null value in the dataset, which are is.null() and anyNA() function. It could check where there is any empty or missing value in the dataset which will be shown as NA, not available.

```
#check null value
is.null(rent)
anyNA(rent)
```

Figure 10: Code to Indicate Whether the Data Type NULL

```
> is.null(rent)
[1] FALSE
> anyNA(rent)
[1] FALSE
```

Figure 11: Output of the NULL Value Indication

2.2.2 Remove Duplication

The duplicated() function help to remove the duplicate rows, where it could prevent the presence of data redundancy.

```
#check duplicated value
duplicated(rent)
```

Figure 12: Code to Check Duplicated Value

The figure below shows the result rows by rows.

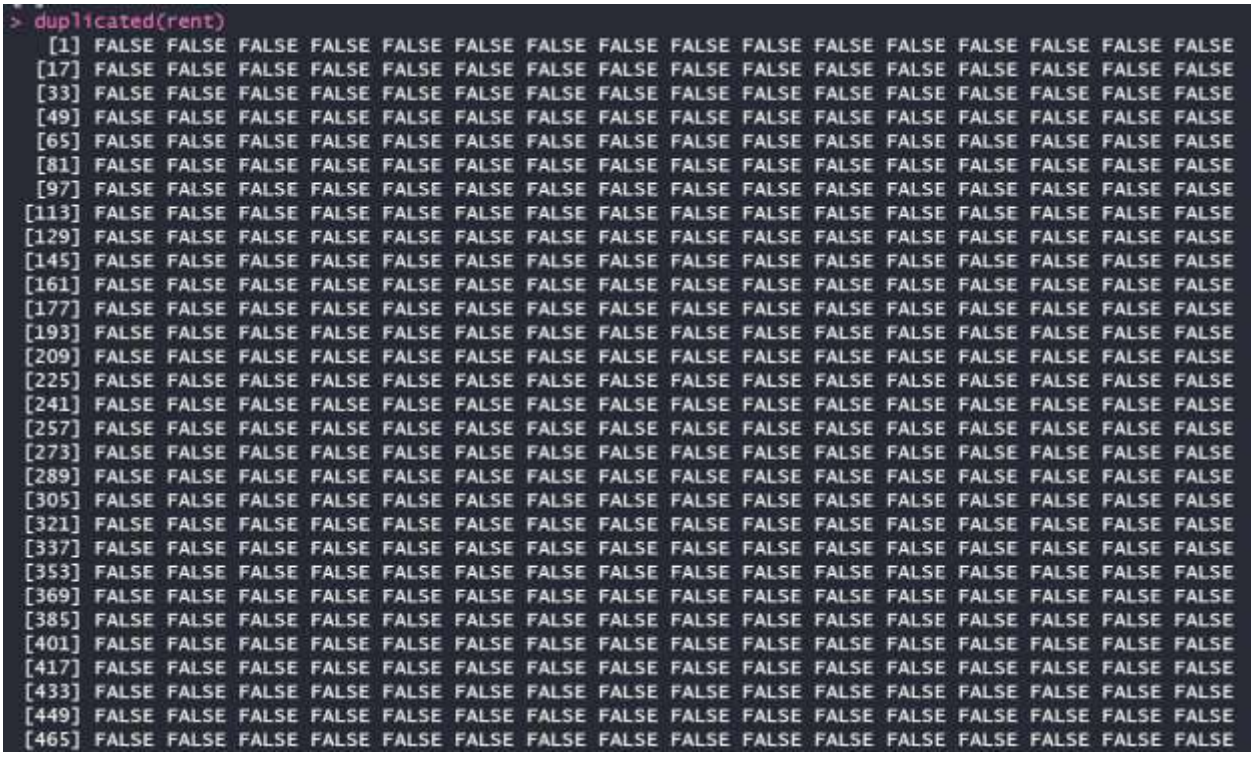

Figure 13: Output of Checking Duplicated Value on the Console

2.3 Pre-processing

2.3.1 Check the Structure

The str() function display all the internal structure of an object. (GeeksforGeeks, 2020) The console will print out the data type of rent, which is a data frame. All the 12 columns will be printed out along with their data type and a few rows of data.

```
#check data type
str(rent)
```

Figure 14: Code to Check the Structure

```
> str(rent)
'data.frame':   4746 obs. of  12 variables:
 $ Posted.On        : chr  "5/18/2022" "5/13/2022" "5/16/2022" "7/4/2022" ...
 $ BHK              : int  2 2 2 2 2 2 2 1 2 2 ...
 $ Rent             : int  10000 20000 17000 10000 7500 7000 10000 5000 26000 10000 ...
 $ Size             : int  1100 800 1000 800 850 600 700 250 800 1000 ...
 $ Floor            : chr  "Ground out of 2" "1 out of 3" "1 out of 3" "1 out of 2" ...
 $ Area.Type        : chr  "Super Area" "Super Area" "Super Area" "Super Area" ...
 $ Area.Locality    : chr  "Bandel" "Phool Bagan, Kankurgachi" "Salt Lake City Sector 2" "Dumdum Park" ...
 $ City             : chr  "Kolkata" "Kolkata" "Kolkata" "Kolkata" ...
 $ Furnishing.Status: chr  "Unfurnished" "Semi-Furnished" "Semi-Furnished" "Unfurnished" ...
 $ Tenant.Preferred : chr  "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" ...
 $ Bathroom         : int  2 1 1 1 1 2 2 1 2 2 ...
 $ Point.of.Contact : chr  "Contact Owner" "Contact Owner" "Contact Owner" "Contact Owner" ...
```

Figure 15: Output of the Structure of Rent Dataset On the Console

The code below is converting the “Posted.On” column’s data type from character to date.

2.3.2 Changing Data Type

The “Posted.On” column is required to be a data format for further analysis. Thus, as.date() function is used and the format month, day and year is represented with the format “%m/%d/%Y”. The format is case sensitive as there comes in different variations.

```
#convert date
rent$Posted.On <- as.Date(rent$Posted.On,"%m/%d/%Y")
```

Figure 16: Code to Convert the Data Type from Character to Date

	Posted.On
1	2022-05-18
2	2022-05-13
3	2022-05-16
4	2022-07-04
5	2022-05-09
6	2022-04-29

Figure 17: Output of the Date in Table Format

2.3.3 Mutate Date

Mutate function allows us to add a new data in the data frame while retaining the old variables to the data frame. The date on “Posted.On” is separated to day, month and year, three column and the format was all changed to numeric format.

```
rent <- mutate(rent, DAY = as.numeric(format(rent$Posted.On, format = "%d")),
               MONTH = as.numeric(format(rent$Posted.On, format = "%m")),
               YEAR = as.numeric(format(rent$Posted.On, format = "%Y")))
```

Figure 18: Code to Mutate the Date in Rent

DAY	MONTH	YEAR
18	5	2022
13	5	2022
16	5	2022
4	7	2022
9	5	2022
29	4	2022

Figure 19: Output of the Mutated Date in Table Form

2.3.4 Modifying Floor Data

For further analysis, the “Floor” column is separate into two columns by removing the “out of” string. The do.call() function is applied to call the rbind function and strsplit() function to separate the floor available and total floor. All the data is converted to data frame and “Floor_Available” and “Total_Floor” header is assigned.

```
#split the floor available and total floor
floor = data.frame(do.call("rbind", strsplit(as.character(rent$Floor), " out of ", fixed = TRUE)))
names(floor)=c("Floor_Available", "Total_Floor")
floor
```

Figure 20: Code to Split the Floor Column

The figure below is the floor data frame. There is character found in the “Floor_Available” column.

```
> floor
   Floor_Available Total_Floor
1           Ground           2
2                1           3
3                1           3
4                1           2
5                1           2
6           Ground           1
7           Ground           4
8                1           2
9                1           2
10               1           3
11               1           4
12               1           1
13               1           4
14               1           2
15          Ground           2
```

Figure 21: Output of the Floor Dataset on the Console

	Floor_Available	Total_Floor
1	Ground	2
2	1	3
3	1	3
4	1	2
5	1	2
6	Ground	1
7	Ground	4
8	1	2
9	1	2

Figure 22: Output of the Floor Dataset in Table Form

The function str() is applied to check the data in “floor” data frame.

```
#check floor structure
str(floor)
```

Figure 23: Code to Check the Floor Structure

```
> str(floor)
'data.frame':   4746 obs. of  2 variables:
 $ Floor_Available: chr  "Ground" "1" "1" "1" ...
 $ Total_Floor    : chr  "2" "3" "3" "2" ...
```

Figure 24: Output of the Floor Structure on the Console

To find all the value contain in floor, unique() function is used to find every element in the data frame and remove the duplicate value.

```
#find element in floor
unique(floor$Floor_Available)
```

Figure 25: Code to Eliminate Duplicated Values in Floor

There is “Ground”, “Upper Basement” and “Lower Basement” in the floor, where it needs to be change to integer.

```
> unique(floor$Floor_Available)
 [1] "Ground"         "1"              "2"              "4"              "3"              "5"
 [7] "7"              "8"              "Upper Basement" "11"             "Lower Basement" "6"
[13] "14"             "43"             "13"             "18"             "17"             "9"
[19] "19"             "60"             "34"             "12"             "26"             "25"
[25] "53"             "16"             "10"             "39"             "32"             "47"
[31] "28"             "20"             "15"             "65"             "40"             "37"
[37] "22"             "21"             "30"             "35"             "33"             "44"
[43] "41"             "46"             "27"             "45"             "48"             "50"
[49] "24"             "23"             "29"             "49"             "36"             "76"
```

Figure 26: Output of the Unique Value in Floor on the Console

Thus, “Ground” will be replaced with “0”, “Upper Basement” will be changed to “0.75” and “Lower Basement” will be switched to “0.25”.

```
floor$Floor_Available[floor$Floor_Available == "Ground"] <- "0"
floor$Floor_Available[floor$Floor_Available == "Upper Basement"] <- "0.75"
floor$Floor_Available[floor$Floor_Available == "Lower Basement"] <- "0.25"
floor$Total_Floor[floor$Total_Floor == "Ground"] <- "0"
unique(floor$Floor_Available)
```

Figure 27: Code to Replace the Characters in Floor

```
> unique(floor$Floor_Available)
 [1] "0"    "1"    "2"    "4"    "3"    "5"    "7"    "8"    "0.75" "11"   "0.25" "6"    "14"   "43"
[15] "13"   "18"   "17"   "9"    "19"   "60"   "34"   "12"   "26"   "25"   "53"   "16"   "10"   "39"
[29] "32"   "47"   "28"   "20"   "15"   "65"   "40"   "37"   "22"   "21"   "30"   "35"   "33"   "44"
[43] "41"   "46"   "27"   "45"   "48"   "50"   "24"   "23"   "29"   "49"   "36"   "76"
```

Figure 28: Output of the Data in Floor on the Console

In order to add the “floor” data frame into “rent” data frame, cbind() function is used to connect these two data frame together.

```
#modify floor column in rent
rent['Floor'] = NULL
rent = cbind(rent, floor)
```

Figure 29: Code to Modify the Floor Column in Rent Dataset

2.3.5 Replacing Value in Point of Contact

Integers is used for easier analysis for Point of Contact category. Thus, it has decided to replace “Contact Owner”, “Contact Agent” and “Contact Builder” to “0”, “1” and “2”.

```
#replace value in point of contact
unique(rent$Point.of.Contact)
rent$Point.of.Contact[rent$Point.of.Contact == "Contact Owner"] <- "0"
rent$Point.of.Contact[rent$Point.of.Contact == "Contact Agent"] <- "1"
rent$Point.of.Contact[rent$Point.of.Contact == "Contact Builder"] <- "2"
```

Figure 30: Code to Replace Value in Point of Contact

```
> unique(rent$Point.of.Contact)
[1] "Contact Owner"   "Contact Agent"   "Contact Builder"
> rent$Point.of.Contact[rent$Point.of.Contact == "Contact Owner"] <- "0"
> rent$Point.of.Contact[rent$Point.of.Contact == "Contact Agent"] <- "1"
> rent$Point.of.Contact[rent$Point.of.Contact == "Contact Builder"] <- "2"
```

```
> rent$Point.of.Contact
  [1] "0" "0" "0" "0" "0" "0" "1" "1" "1" "0" "1" "1" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1"
 [26] "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
 [51] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "0" "0"
 [76] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "0" "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" "1"
[101] "1" "1" "0" "1" "0" "0" "0" "0" "0" "0" "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "1" "0" "0"
[126] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
[151] "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
[176] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1"
[201] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0"
[226] "0" "0" "0" "0" "1" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0"
[251] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0"
[276] "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0"
[301] "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
[326] "1" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "1"
[351] "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "0" "0" "0" "1" "0"
```

Figure 31: Output of the Replace Value in Point of Contact on the Console

2.3.6 Changing Data Type 2

All the data that has been modified previously are required to change the data type to numeric format by using the as.numeric() function.

```
rent$Point.of.Contact <- as.numeric(rent$Point.of.Contact)
rent$Floor_Available <- as.numeric(rent$Floor_Available)
rent$Total_Floor <- as.numeric(rent$Total_Floor)
```

Figure 32: Code to Change Data Type from Characters to Numeric

2.3.7 Rename Column

By using the names() functions, all the header names are changed to their respective name. There are 16 column names to be altered.

```
#assign header
names(rent)=c("DATE","BHK","RENTAL","HOUSE_SIZE","AREA_TYPE",
              "LOCALITY","CITY","FURNISHING_STATUS","TENANT_PREFERRED",
              "BATHROOM","POINT_OF_CONTACT","DAY","MONTH","YEAR","FLOOR_AVAILABLE","TOTAL_FLOOR")
```

Figure 33: Code to Rename the Columns for the Rent Dataset

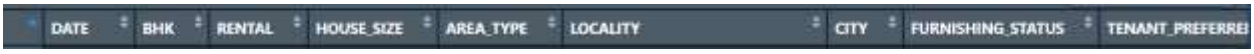

Figure 34: Output of the Renamed Columns in Table Format

2.3.8 Check the Data Type for Each Column

All the data type for each column will be store into "datatype" data frame by using sapply() function to check the class type for each column.

```
#check data type
datatype = data.frame(DATA_TYPE = sapply(rent, class))
datatype
View(datatype)
```

Figure 35: Code to Check the Data Type for Each Column

```
> datatype
                  DATA_TYPE
DATE                   Date
BHK                 integer
RENTAL              integer
HOUSE_SIZE          integer
AREA_TYPE         character
LOCALITY          character
CITY              character
FURNISHING_STATUS character
TENANT_PREFERRED  character
BATHROOM            integer
POINT_OF_CONTACT    numeric
DAY                 numeric
MONTH               numeric
YEAR                numeric
FLOOR_AVAILABLE     numeric
TOTAL_FLOOR         numeric
```

Figure 36: Output of the Data Type for Each Column on the Console

	DATA_TYPE
DATE	Date
BHK	integer
RENTAL	integer
HOUSE_SIZE	integer
AREA_TYPE	character
LOCALITY	character
CITY	character
FURNISHING_STATUS	character
TENANT_PREFERRED	character
BATHROOM	integer
POINT_OF_CONTACT	numeric
DAY	numeric
MONTH	numeric
YEAR	numeric
FLOOR_AVAILABLE	numeric
TOTAL_FLOOR	numeric

Figure 37: Output of the Data Type for Each Column in Table Format

2.3.9 Structure and Summary of the Rent Dataset

This code below helps us to view the structure of the rent dataset.

```
#structure
str(rent)
```

Figure 38: Code to View the Structure of the Rent Dataset

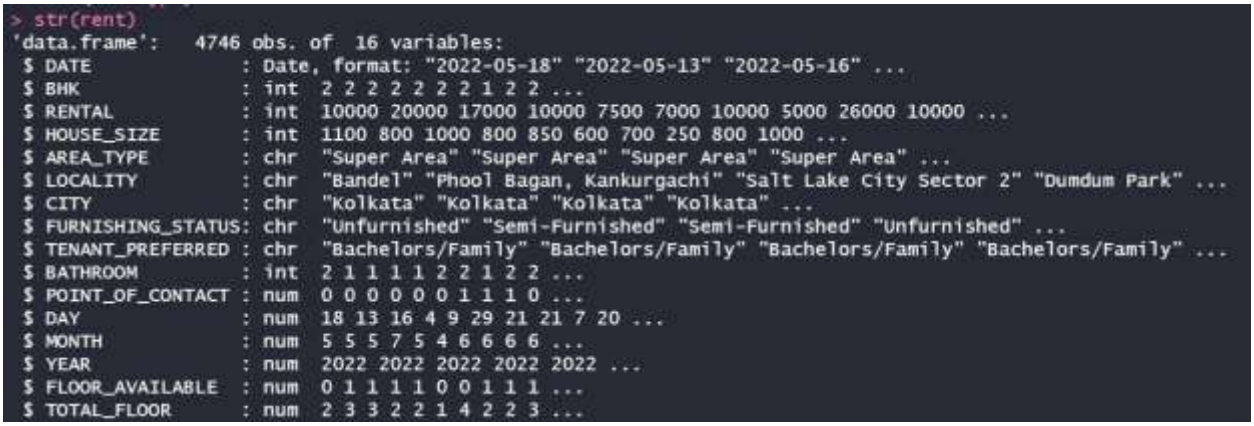

Figure 39: Output of the Structure of the Rent Dataset on the Console

By using summary() function, we could view the summary of the rent dataset.

```
#summarize data
summary(rent)
```

Figure 40: Code to View the Summary of the Rent Dataset

The figure below shows the minimum value, 1st quartile value, median, mean, 3rd quartile calue and the maximum value for the numeric and integers data. Others will show the length, class and mode of the data.

```
      DATE                 BHK              RENTAL          HOUSE_SIZE      AREA_TYPE
 Min.   :2022-04-13   Min.   :1.000   Min.   :   1200   Min.   :  10.0   Length:4746
 1st Qu.:2022-05-20   1st Qu.:2.000   1st Qu.:  10000   1st Qu.: 550.0   Class :character
 Median :2022-06-10   Median :2.000   Median :  16000   Median : 850.0   Mode  :character
 Mean   :2022-06-07   Mean   :2.084   Mean   :  34993   Mean   : 967.5
 3rd Qu.:2022-06-28   3rd Qu.:3.000   3rd Qu.:  33000   3rd Qu.:1200.0
 Max.   :2022-07-11   Max.   :6.000   Max.   :3500000   Max.   :8000.0
   LOCALITY             CITY           FURNISHING_STATUS  TENANT_PREFERRED      BATHROOM
 Length:4746        Length:4746        Length:4746        Length:4746        Min.   : 1.000
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.000
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 2.000
                                                                             Mean   : 1.966
                                                                             3rd Qu.: 2.000
                                                                             Max.   :10.000
 POINT_OF_CONTACT       DAY              MONTH            YEAR      FLOOR_AVAILABLE  TOTAL_FLOOR
 Min.   :0.0000   Min.   : 1.00   Min.   :4.000   Min.   :2022   Min.   : 0.00   Min.   : 0.000
 1st Qu.:0.0000   1st Qu.: 7.00   1st Qu.:5.000   1st Qu.:2022   1st Qu.: 1.00   1st Qu.: 2.000
 Median :0.0000   Median :14.00   Median :6.000   Median :2022   Median : 2.00   Median : 4.000
 Mean   :0.3226   Mean   :15.48   Mean   :5.756   Mean   :2022   Mean   : 3.45   Mean   : 6.969
 3rd Qu.:1.0000   3rd Qu.:23.00   3rd Qu.:6.000   3rd Qu.:2022   3rd Qu.: 3.00   3rd Qu.: 6.000
 Max.   :2.0000   Max.   :31.00   Max.   :7.000   Max.   :2022   Max.   :76.00   Max.   :89.000
```

Figure 41: Output of the Summary of the Rent Dataset

2.4 Data Exploration

2.4.1 The Column, Row and Dimension of the Rent Dataset

To calculate the number of columns available in the dataset, ncol() function is used. The nrow() function calculates the number of rows available in the dataset. Last, the dim() function could show the dimension of the dataset.

```
###DATA EXPLORATION###
#Column, row and dimension
ncol(rent)
nrow(rent)
dim(rent)
```

Figure 42: Code to Calculate the Column, Row and Dimesion of the Rent Dataset

The numbers of column, rows and dimensions will be printed out on the console.

```
> ncol(rent)
[1] 16
> nrow(rent)
[1] 4746
> dim(rent)
[1] 4746   16
```

Figure 43: Output of the Column, Row and Dimension on the Console

2.4.2 Total Area Type in the House Rent Dataset

By using nlevels() and factor() function, it will calculate the total area type in the house rent dataset. The paste() allows us to paste the total area type on the console. The data type is duplicated and converted into a list. By using the [!area_col], it remove the duplicate columns

and convert the data to a data frame. Last, the column name is renamed as “Area Type” and using the View() function to demonstrate the results in table format.

```
#TOTAL AREA TYPE
area_num = nlevels(factor(rent$AREA_TYPE))
paste("Total Area Type: ", area_num)
unique(rent$AREA_TYPE)

#Convert to list
area_col = duplicated(as.list(rent$AREA_TYPE))

#Remove duplicated column
area = as.data.frame(rent$AREA_TYPE[!area_col])
names(area) = c("Area Type")
View(area)
```

Figure 44: Code to Show the Total Area Type

```
> paste("Total Area Type: ", area_num)
[1] "Total Area Type:  3"
> unique(rent$AREA_TYPE)
[1] "Super Area"  "Carpet Area" "Built Area"
```

Figure 45: Output of the Total Area Type on the Console

	Area Type
1	Super Area
2	Carpet Area
3	Built Area

Figure 46: Output of the Total Area Type in Table Format

2.4.3 Total Area Locality

The code beneath calculates the total area locality in the house rent dataset.

```
#TOTAL AREA LOCALITY
locality_num = nlevels(factor(rent$LOCALITY))
paste("Total Area Locality: ", locality_num)
unique(rent$LOCALITY)

#Convert to list
locality_col = duplicated(as.list(rent$LOCALITY))

#Remove duplicated column
locality = as.data.frame(rent$LOCALITY[!locality_col])
names(locality) = c("Area Locality")
View(locality)
```

Figure 47: Code to Show Total Area Locality

```
> locality_num = nlevels(factor(rent$LOCALITY))
> paste("Total Area Locality: ", locality_num)
[1] "Total Area Locality:  2033"
```

Figure 48: Output to the Show Total Area Locality on the Console

	Area Locality
1	Bandel
2	Phool Bagan, Kankurgachi
3	Salt Lake City Sector 2
4	Dumdum Park
5	South Dum Dum
6	Thakurpukur
7	Malancha
8	Palm Avenue Kolkata, Ballygunge
9	Natunhat
10	Action Area 1, Rajarhat Newtown
11	Tarulia, Keshtopur
12	Dum Dum Metro
13	Paschim Barisha
14	Barasat
15	Behala
16	Behala Chowrasta
17	Santoshpur
18	Garia Station, Garia
19	Joka
20	Sreebhumi

Figure 49: Output to the Show Total Area Locality in Table Format

2.4.4 Total City

The code below counts the total city in the house rent dataset.

```
#TOTAL CITY
city_num = nlevels(factor(rent$CITY))
paste("Total City: ", city_num)
unique(rent$CITY)

#Convert to list
city_col = duplicated(as.list(rent$CITY))

#Remove duplicated column
city = as.data.frame(rent$CITY[!city_col])
names(city) = c("City")
View(city)
```

Figure 50: Code to Show Total City

There are six cities in this dataset, which are Kolkata, Mumbai, Bangalore, Delhi, Chennai and Hyderabad.

```
> paste("Total City: ", city_num)
[1] "Total City:  6"
> unique(rent$CITY)
[1] "Kolkata"   "Mumbai"    "Bangalore" "Delhi"     "Chennai"   "Hyderabad"
```

Figure 51: Output to the Show Total City on the Console

	City
1	Kolkata
2	Mumbai
3	Bangalore
4	Delhi
5	Chennai
6	Hyderabad

Figure 52: Output to the Show Total City in Table Format

2.4.5 Total Type of Furnishing Status

The code below counts the total furnishing status in the house rent dataset.

```
#TOTAL TYPE OF FURNISHING STATUS
furnish_num = nlevels(factor(rent$FURNISHING_STATUS))
paste("Total Type of Furnishing Status: ", furnish_num)
unique(rent$FURNISHING_STATUS)

#Convert to list
furnish_col = duplicated(as.list(rent$FURNISHING_STATUS))

#Remove duplicated column
furnish = as.data.frame(rent$FURNISHING_STATUS[!furnish_col])
names(furnish) = c("Furnishing Status")
View(furnish)
```

Figure 53: Code to Show Total Furnishing Status

There are three types of furnishing status, which are unfurnished, semi-furnished and furnished.

```
> paste("Total Type of Furnishing Status: ", furnish_num)
[1] "Total Type of Furnishing Status:  3"
> unique(rent$FURNISHING_STATUS)
[1] "Unfurnished"    "Semi-Furnished" "Furnished"
```

Figure 54: Output to the Show Total Furnishing Status on the Console

	Furnishing Status
1	Unfurnished
2	Semi-Furnished
3	Furnished

Figure 55: Output to the Show Total Furnishing Status in Table Format

2.4.6 Total Type of Tenant Preferred

The code below counts the total tenant preferred in the house rent dataset.

```
#TOTAL TYPE OF TENANT PREFERRED
tenant_num = nlevels(factor(rent$TENANT_PREFERRED))
paste("Total Type of Tenant Preferred: ", tenant_num)
unique(rent$TENANT_PREFERRED)

#Convert to list
tenant_col = duplicated(as.list(rent$TENANT_PREFERRED))

#Remove duplicated column
tenant = as.data.frame(rent$TENANT_PREFERRED[!tenant_col])
names(tenant) = c("Tenant Preferred")
View(tenant)
```

Figure 56: Code to Show Total Type of Tenant Preferred

There are three types of tenants preferred, which are bachelors/family, bachelors and family.

```
> paste("Total Type of Tenant Preferred: ", tenant_num)
[1] "Total Type of Tenant Preferred:  3"
> unique(rent$TENANT_PREFERRED)
[1] "Bachelors/Family" "Bachelors"        "Family"
```

Figure 57: Output to the Show Total Type of Tenant Preferred on the Console

	Tenant Preferred
1	Bachelors/Family
2	Bachelors
3	Family

Figure 58: Output to the Show Total Type of Tenant Preferred in Table Format

2.4.7 Total Type of Point of Contact

The code below counts the total type of point of contact in the house rent dataset.

```
#TOTAL TYPE OF POINT OF CONTACT
contact_num = nlevels(factor(rent$POINT_OF_CONTACT))
paste("Total Type of Point of Contact: ", contact_num)
unique(rent$POINT_OF_CONTACT)

#Convert to list
contact_col = duplicated(as.list(rent$POINT_OF_CONTACT))

#Remove duplicated column
contact = as.data.frame(rent$POINT_OF_CONTACT[!contact_col])
names(contact) = c("Point of Contact")
point_of_contact_label = c("Contact Owner", "Contact Agent", "Contact Builder")
contact <- cbind(point_of_contact_label, contact)
names(contact) <- c("Point of Contact", "Point of Contact Label")
View(contact)
```

Figure 59: Code to Show Total Type of Point of Contact

There are three types of point of contact, which are contact owner, contact agent and contact builder.

```
> paste("Total Type of Point of Contact: ", contact_num)
[1] "Total Type of Point of Contact:  3"
> unique(rent$POINT_OF_CONTACT)
[1] 0 1 2
```

Figure 60: Output to the Show Total Type of Point of Contact on the Console

	Point of Contact	Point of Contact Label
1	Contact Owner	0
2	Contact Agent	1
3	Contact Builder	2

Figure 61: Output to the Show Total Type of Point of Contact in Table Format

2.4.8 House Size Range

The house size is categorized into 4 categories, which are “0-2000”, “2001-4000”, “4001-6000” and “6001-8000”. The cut(), group_by() and dplyr functions is used to separate the house size into 4 categories. Piping is used to connect all the functions together.

```
#House Size Range Data
house_size_range <- rent %>%
  mutate(SIZE_RANGE = cut(HOUSE_SIZE,
                          seq(0,8000,2000))) %>%
  group_by(SIZE_RANGE) %>%
  dplyr::summarise(QUANTITY = n()) %>%
  as.data.frame()
house_size_range$SIZE_RANGE <- c("0-2000", "2001-4000", "4001-6000", "6001-8000")
house_size_range
```

Figure 62: Code to Categorized the House Size into Four Range

```
> house_size_range
  SIZE_RANGE QUANTITY
1     0-2000     4514
2  2001-4000      212
3  4001-6000       16
4  6001-8000        2
```

Figure 63: Output of the Four House Size Range on the Console

	SIZE_RANGE	QUANTITY
1	0-2000	4514
2	2001-4000	212
3	4001-6000	16
4	6001-8000	2

Figure 64: Output of the Four House Size Range in Table Format

2.4.9 Total Type of BHK

The code below counts the total type of bedrooms, halls and kitchens in the house rent dataset.

```
#TOTAL TYPE OF BHK
bhk_num = nlevels(factor(rent$BHK))
paste("Total Type of BHK: ", bhk_num)
unique(rent$BHK)

#Convert to list
bhk_col = duplicated(as.list(rent$BHK))

#Remove duplicated column
bhk = as.data.frame(rent$BHK[!bhk_col])
names(bhk) = c("BHK")
View(bhk)
```

Figure 65: Code to Show Total Type of BHK

```
> paste("Total Type of BHK: ", bhk_num)
[1] "Total Type of BHK:  6"
> unique(rent$BHK)
[1] 2 1 3 4 5 6
```

Figure 66: Output to the Show Total Type of BHK on the Console

	BHK
1	2
2	1
3	3
4	4
5	5
6	6

Figure 67: Output to the Show Total Type of BHK in Table Format

2.4.10 Total Type of Floor Available

The code below counts the total type of floor available in the house rent dataset.

```
#TOTAL TYPE OF FLOOR AVAILABLE
floor_available_num = nlevels(factor(rent$FLOOR_AVAILABLE))
paste("Total Type of Floor Available: ", floor_available_num)
unique(rent$FLOOR_AVAILABLE)

#Convert to list
floor_available_col = duplicated(as.list(rent$FLOOR_AVAILABLE))

#Remove duplicated column
floor_available = as.data.frame(rent$FLOOR_AVAILABLE[!floor_available_col])
names(floor_available) = c("Floor Available")
View(floor_available)
```

Figure 68: Code to Show Total Type of Floor Available

```
> paste("Total Type of Floor Available: ", floor_available_num)
[1] "Total Type of Floor Available:  51"
> unique(rent$FLOOR_AVAILABLE)
 [1]  0  1  2  4  7  3 11  5  6 43 13 18 17 14  9 19 60 34 12 26 25 53 16 10 39 32 47 28  8 20 15 65 40 37
[35] 22 21 30 35 33 44 46 27 45 48 50 24 23 29 49 36 76
```

Figure 69: Output to the Show Total Type of Floor Available on the Console

	Floor Available
1	0
2	1
3	2
4	4
5	7
6	3
7	11
8	5
9	6
10	43
11	13
12	18
13	17
14	14
15	9
16	19
17	60
18	34
19	12

Figure 70: Output to the Show Total Type of Floor Available in Table Format

2.4.11 Total Type of Total Floor

The code below calculates the total type of total floor in the house rent dataset.

```
#TOTAL TYPE OF TOTAL FLOOR
total_floor_num = nlevels(factor(rent$TOTAL_FLOOR))
paste("Total Type of Total Floor: ", total_floor_num)
unique(rent$TOTAL_FLOOR)

#Convert to list
total_floor_col = duplicated(as.list(rent$TOTAL_FLOOR))

#Remove duplicated column
total_floor = as.data.frame(rent$TOTAL_FLOOR[!total_floor_col])
names(total_floor) = c("Total Floor")
View(total_floor)
```

Figure 71: Code to Show Total Type of Total Floor

```
> paste("Total Type of Total Floor: ", total_floor_num)
[1] "Total Type of Total Floor:  66"
> unique(rent$TOTAL_FLOOR)
 [1]  2  3  1  4  5 14  8  6 19 10  7 13 78 18 12 24 31 21 23 20  9 22 58 16 66 48 40 44 42 41 60 32 30 29
[35] 89 15 11 28 17 45 35 75 38 51 43 25 27 26 76 36 37 55 68 77 50 59 62 39 52 54 33 46 85 71 81 34
```

Figure 72: Output to the Show Total Type of Total Floor on the Console

	Total Floor
1	2
2	3
3	1
4	4
5	5
6	14

Figure 73: Output to the Show Total Type of Total Floor in Table Format

2.4.12 Total Type of Bathroom

The code below sums up the total type of bathroom in the house rent dataset.

```
#TOTAL TYPE OF BATHROOM
bathroom_num = nlevels(factor(rent$BATHROOM))
paste("Total Type of Total Floor: ", bathroom_num)
unique(rent$BATHROOM)

#Convert to list
bathroom_col = duplicated(as.list(rent$BATHROOM))

#Remove duplicated column
bathroom = as.data.frame(rent$BATHROOM[!bathroom_col])
names(bathroom) = c("Floor Available")
View(bathroom)
```

Figure 74: Code to Show Total Type of Bathroom

```
> paste("Total Type of Total Floor: ", bathroom_num)
[1] "Total Type of Total Floor:  8"
> unique(rent$BATHROOM)
[1]  2  1  3  5  4  6 10  7
```

Figure 75: Output to the Show Total Type of Bathroom on the Console

	Floor Available
1	2
2	1
3	3
4	5
5	4
6	6
7	10
8	7

Figure 76: Output to the Show Total Type of Bathroom in Table Format

2.4.13 Rental Range

As the range of the rental is wide, the rental is separate into a few categories. The range will be adjusted according to the analysis requirements and the method that is used is similar to the house size range.

```
#Rental Range Data
rental_range_1200_3500000 <- rent %>%
  mutate(RENTAL_RANGE = cut(RENTAL,
                            seq(1200,3500000,50000))) %>%
  group_by(RENTAL_RANGE) %>%
  dplyr::summarise(QUANTITY = n()) %>%
  as.data.frame()

rental_range_1200_3500000$RENTAL_RANGE <- c("1000-51000", "51000-101000", "101000-151000",
                                            "151000-201000", "201000-251000", "251000-301000",
                                            "301000-351000","351000-401000","401000-451000",
                                            "451000-501000", "501000-551000","551000-601000",
                                            "601000-651000","651000-701000","801000-851000",
                                            "951000-1000000","1150000-1200000", "1200000-3500000")
rental_range_1200_3500000
```

```
rental_range_1200_1200000 <- rent %>%
  mutate(RENTAL_RANGE = cut(RENTAL,
                            seq(1200,1200000,50000))) %>%
  group_by(RENTAL_RANGE) %>%
  dplyr::summarise(QUANTITY = n()) %>%
  as.data.frame()

rental_range_1200_1200000$RENTAL_RANGE <- c("1000-51000", "51000-101000", "101000-151000",
                                            "151000-201000", "201000-251000", "251000-301000",
                                            "301000-351000","351000-401000","401000-451000",
                                            "451000-501000", "501000-551000","551000-601000",
                                            "601000-651000","651000-701000","801000-851000",
                                            "951000-1000000","1150000-1200000")
rental_range_1200_1200000
```

```
rental_range1_1200_1200000 <- rent %>%
  mutate(RENTAL_RANGE = cut(RENTAL,
                            seq(1000,1200000,100000))) %>%
  group_by(RENTAL_RANGE) %>%
  dplyr::summarise(QUANTITY = n()) %>%
  as.data.frame()
rental_range1_1200_1200000$RENTAL_RANGE <- c("1000-100000", "100000-201000", "201000-301000",
                                             "301000-401000","401000-501000","501000-601000",
                                             "601000-701000","801000-901000",
                                             "901000-1000000","1100000-1200000")
rental_range1_1200_1200000
```

```
rental_range_1000_10000 <- rent %>%
  mutate(RENTAL_RANGE = cut(RENTAL,
                            seq(1000,10000,1000))) %>%
  group_by(RENTAL_RANGE) %>%
  dplyr::summarise(QUANTITY = n()) %>%
  as.data.frame()
rental_range_1000_10000 <-rental_range_1000_10000 %>% drop_na()

rental_range_1000_10000$RENTAL_RANGE <- c("1000-2000", "2000-3000", "3000-4000",
                                "4000-5000","5000-6000","6000-7000",
                                "7000-8000","8000-9000","9000-10000")
rental_range_1000_10000
```

Figure 77: Code to Categorized the Rental

	RENTAL_RANGE	QUANTITY
1	1000-51000	4026
2	51000-101000	439
3	101000-151000	119
4	151000-201000	62
5	201000-251000	28
6	251000-301000	30
7	301000-351000	16
8	351000-401000	11
9	401000-451000	1
10	451000-501000	1
11	501000-551000	1
12	551000-601000	4
13	601000-651000	1
14	651000-701000	2
15	801000-851000	1
16	951000-1000000	1
17	1150000-1200000	1
18	1200000-3500000	2

	RENTAL_RANGE	QUANTITY
1	1000-51000	4026
2	51000-101000	439
3	101000-151000	119
4	151000-201000	62
5	201000-251000	28
6	251000-301000	30
7	301000-351000	16
8	351000-401000	11
9	401000-451000	1
10	451000-501000	1
11	501000-551000	1
12	551000-601000	4
13	601000-651000	1
14	651000-701000	2
15	801000-851000	1
16	951000-1000000	1
17	1150000-1200000	3

> rental_range1_1200_1200000

	RENTAL_RANGE	QUANTITY
1	1000-100000	4466
2	100000-201000	181
3	201000-301000	58
4	301000-401000	27
5	401000-501000	2
6	501000-601000	5
7	601000-701000	3
8	801000-901000	1
9	901000-1000000	1
10	1100000-1200000	2

> rental_range_1000_10000

	RENTAL_RANGE	QUANTITY
1	1000-2000	4
2	2000-3000	10
3	3000-4000	35
4	4000-5000	116
5	5000-6000	162
6	6000-7000	217
7	7000-8000	257
8	8000-9000	221
9	9000-10000	306

Figure 78: Output of the Rental Range on the Console

2.4.14 Force Full Display

The first line of code below disables the scientific notation printed in the result. The second line of code turn off the significance stars from the regression output. These two code uses options() function.

```
#Force full display
options(scipen=999)
options(show.signif.stars=FALSE)
```

Figure 79: Code to Force Full Display

3.0 Question and Analysis

3.1 Question 1: How does geographical location affect the house rent?

This question intended to find out the influence of the city and locality on house rent.

3.1.1 Analysis 1.1: The Distribution of House Available to Rent According to the Cities

The code below counts the amount of house available in each city. At first, the writer uses piping function to do multiple action at the same time. The rent data set is group by city using group_by() function and the data are then summarize by counting the city quantity using summarize() function from dplyr packages. The amount of the house available will be store in house_city. The data will be shown in pie chart and bar plot to see the distribution of house available to rent according to the cities. First, par() function is used to adjust the margin in the plot. The pie() function includes the design of the pie where it will print out the quantity and name on the pie chart. The colour will be displayed by using topo.colors() function where it required the number of the cities. Thus, length() is used to calculate the amount of the cities. Legend is used to show the colour and city where it could help the viewers to understand the pie chart clearly. The size and the location of the legend could be adjusted through pt.cex, cex and inset.

```
#Question 1: How does geographical location affect the house rent ?
#Analysis 1.1: The Distribution of House Available to Rent According to the Cities
#Pie Chart
house_city <- rent %>%
  group_by(CITY) %>%
  summarise(QUANTITY = n())
house_city

summary(house_city)

par(mar = c(2, 2, 2, 2))
pie(house_city$QUANTITY, paste(house_city$CITY,house_city$QUANTITY),
    radius = 0.7, main = "The Distribution of House Available to Rent According to the Cities",
    col = topo.colors(length(house_city$CITY)), clockwise =  TRUE)
legend("topright", house_city$CITY, fill = topo.colors(length(house_city$CITY)),
       pt.cex = 2, cex = 0.7, horiz = FALSE, inset = c( -0.3 , 0.35)) #Additional Features
```

Figure 80: Code to Calculate the Distribution of House Available to Rent According to the Cities in Pie Chart

The summary could find out the mean of the house available.

```
> summary(house_city)
     CITY               QUANTITY
 Length:6           Min.   :524.0
 Class :character   1st Qu.:670.8
 Mode  :character   Median :877.0
                    Mean   :791.0
                    3rd Qu.:889.8
                    Max.   :972.0
```

Figure 81: Output of Summary of the Distribution of House Available to Rent According to the Cities in Pie Chart

The results below include the numbers of house available to rent in each city, which are 524 houses in Kolkata, 605 houses in Delhi, 868 houses in Hyderabad, 886 houses in Bangalore, 891 houses in Chennai and 972 houses in Mumbai.

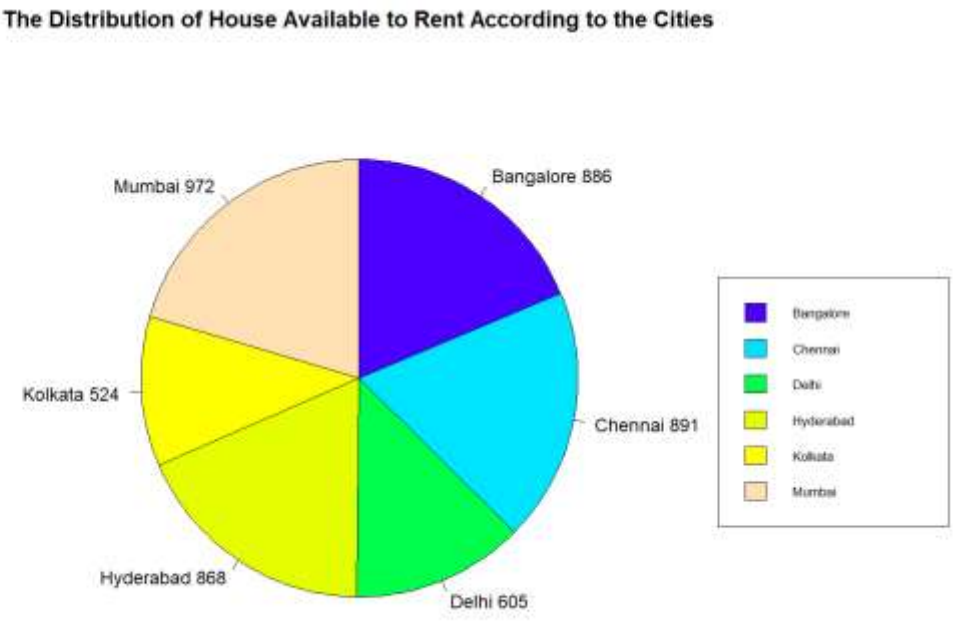

Figure 82: The Distribution of House Available to Rent According to the Cities Pie Chart

The ggplot() function is used to create a bar plot for the house distribution in each city. The writer uses geom_bar to create the bar plot. Color and fill are used to adjust the color for the bar plot. The theme () function is used to adjust the title's font, position and size. To add the label in the bar plot, geom_text() is used to determine the label content, position and size,

```
#Bar Plot
ggplot(house_city, aes(x = CITY, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(house_city$CITY))) +
  ggtitle("The Distribution of House Available to Rent According to the Cities") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) + #Additional Features
  geom_text(aes(CITY, label = QUANTITY), vjust = 1, position = position_dodge(width = 0.1))
```

Figure 83: Code to Calculate the Distribution of House Available to Rent According to the Cities in Bar Plot

The bar plot allows the viewers to see the result clearly. Mumbai has the highest house available to rent compared to Kolkata, it has the lowest house available to rent. Even though Mumbai has lower population density, it has higher Gross Domestic Product (GDP). (Versus, n.d.)

The number of house availability in Kolkata and Delhi is below 791, which is lower than the mean. Bangalore, Chennai, Hyderabad and Mumbai are above the mean.

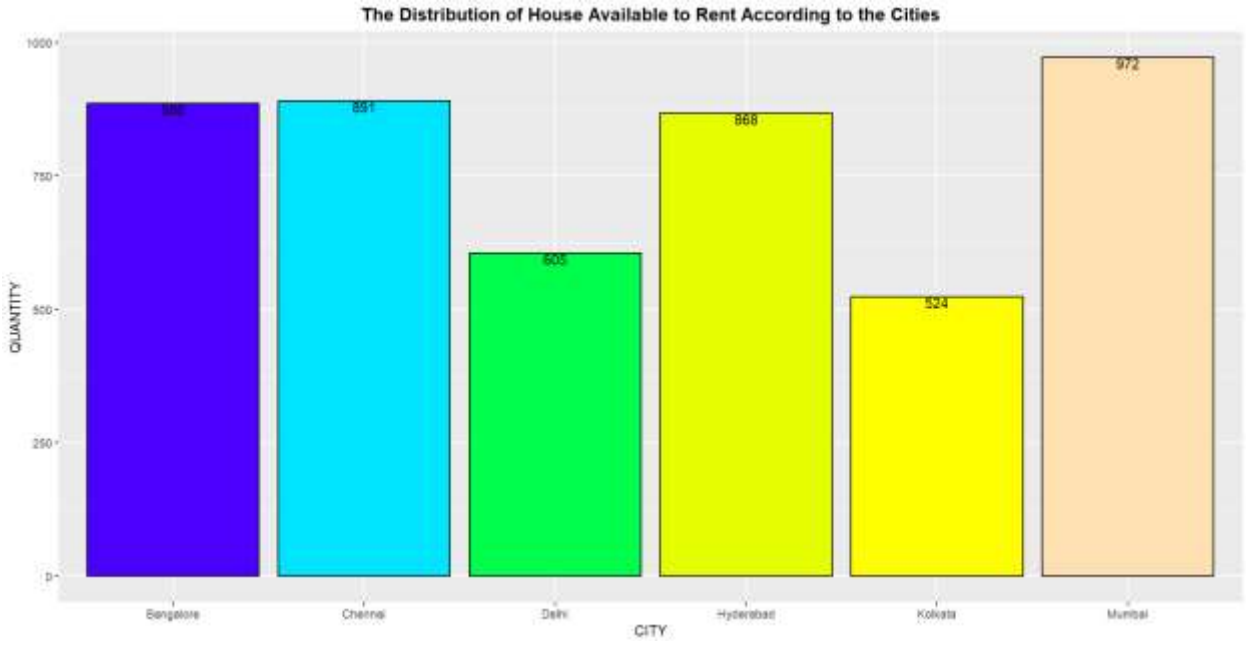

Figure 84: The Distribution of House Available to Rent According to the Cities Bar Plot

3.1.2 Analysis 1.2: The Rent Distribution

The code below shows the rent distribution from 1000 – 3500000.

```
#Analysis 1.2: The Rent Distribution
#Bar Plot
par(mar = c(6, 6, 6, 6))
ggplot(rental_range_1200_3500000, aes(x = RENTAL_RANGE, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.5, color = "Black",
           fill = topo.colors(length(rental_range_1200_3500000$RENTAL_RANGE))) +
  ggtitle("The Rent Distribution (Range Between 1000 - 3500000)") +
  theme(axis.text.x = element_text(angle = 65, vjust = 0.6),
        plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(RENTAL_RANGE, label = QUANTITY), vjust = -1, position = position_dodge(width = 0.1))
```

Figure 85: Code to Show the Rent Distribution from 1000 to 3500000 in Bar Plot

Figure 86 and 87 shows the rental distribution where the highest category is 1000 – 51000, which is approximately 84.83% houses in this category. The second highest category is 51000-101000, which is nearly 9.25% houses rent.

	RENTAL_RANGE	QUANTITY
1	1000-51000	4026
2	51000-101000	439
3	101000-151000	119
4	151000-201000	62
5	201000-251000	28
6	251000-301000	30
7	301000-351000	16
8	351000-401000	11
9	401000-451000	1
10	451000-501000	1
11	501000-551000	1
12	551000-601000	4
13	601000-651000	1
14	651000-701000	2
15	801000-851000	1
16	951000-1000000	1
17	1150000-1200000	1
18	1200000-3500000	2

Figure 86: Output of the Rental Distribution between 1000 to 3500000

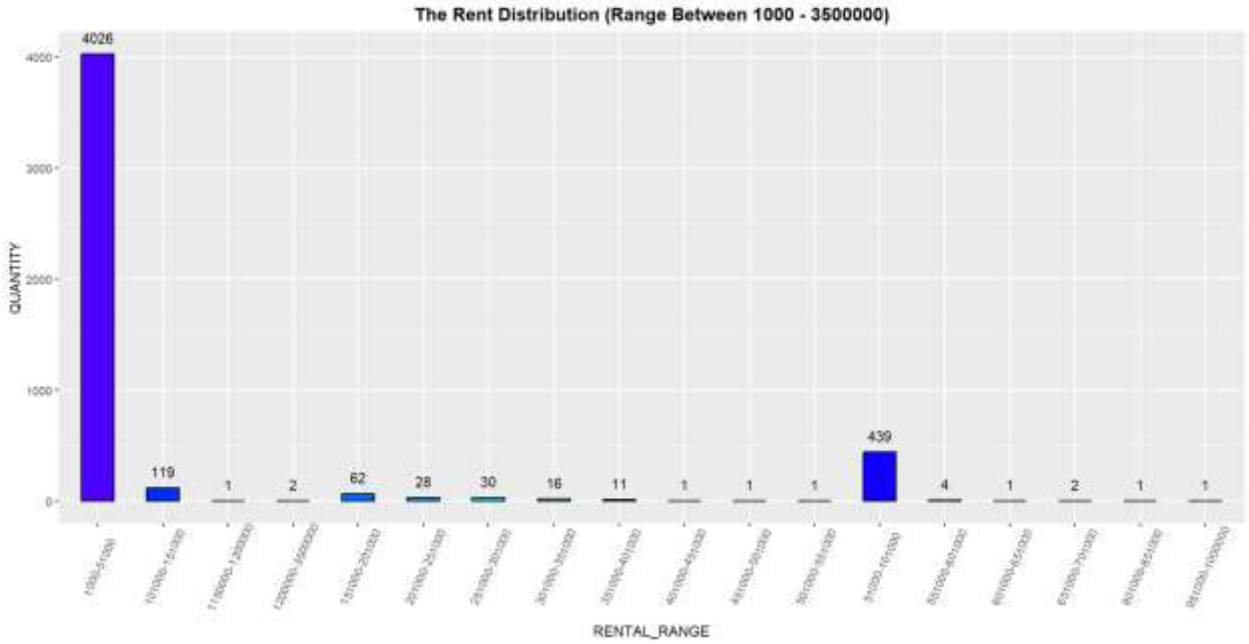

Figure 87: The Rent Distribution from 1000 to 3500000 in Bar Plot

In order to look closely at the rent distribution from 1000 to 10000, a lollipop chart is created by using ggplot() function. The difference between bar plot and lollipop chart is the presence of geom_point() function. It allows the shape to change according to the preferences and by using geom_text(), the label could perfectly fit inside the shape.

```
#Lollipop Chart for Range Between 1000 to 10000 #Additional Features
ggplot(rental_range_1000_10000, aes(x = RENTAL_RANGE, y = QUANTITY)) +
  geom_segment( aes(x = RENTAL_RANGE, xend = RENTAL_RANGE, y = 0, yend = QUANTITY),
                colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(rental_range_1000_10000$RENTAL_RANGE)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = QUANTITY), color = "black", size = 3.5) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The Rent Distribution (Range Between 1000 to 10000)")
```

Figure 88: Code to Show the Rent Distribution from 1000 to 10000 in Lollipop Chart

The highest rent distribution among 1000 to 10000 is the range between 9000 – 10000, where there are 306 houses rent in the range. 7000-8000 is the second highest number of rent distribution, which is 257 houses.

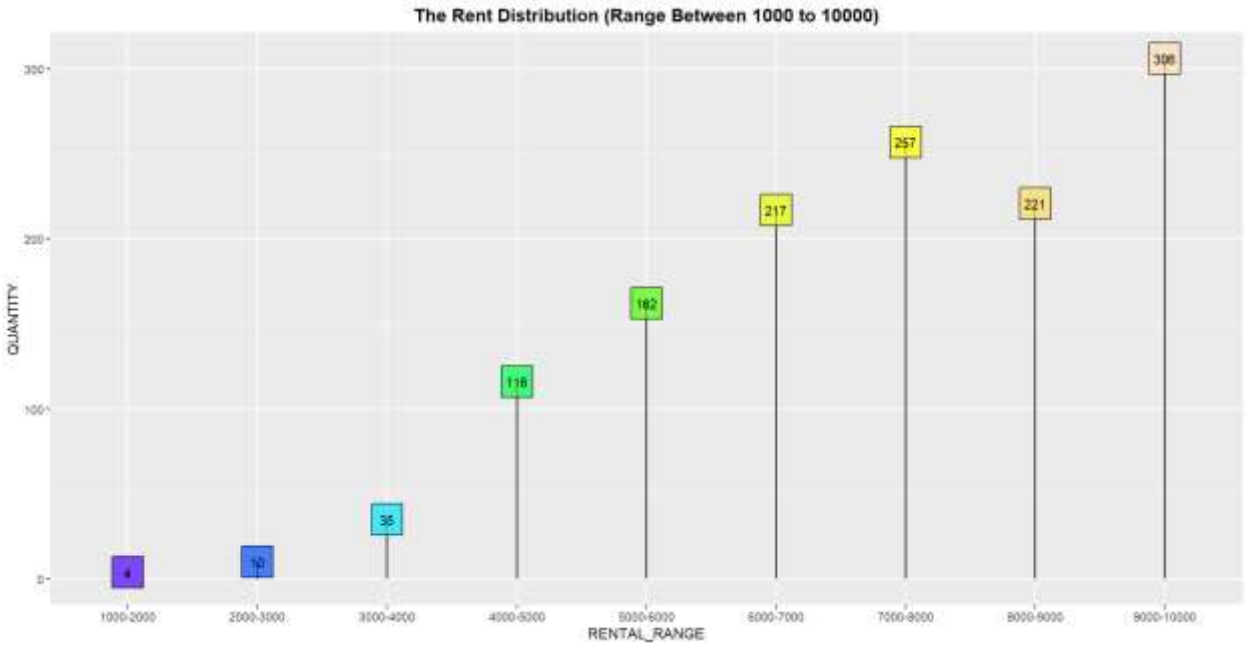

Figure 89: The Rent Distribution from 1000 to 10000 in Lollipop Chart

3.1.3 Analysis 1.3: The House Distribution in Each City According to the Locality

3.1.3.1 Analysis 1.3.1: The House Distribution in Kolkata According to the Locality

The first group of code filters the city with the name of Kolkata by using filter() function. Then, the second group of code count the quantity of house available for each locality. Lollipop chart is used to have a simplified view of the distribution. The scale_x_continuous() function determines the scale in the x axis.

```
#Analysis 1.3: The House Distribution in Each City According to Locality
#Analysis 1.3.1: The House Distribution in Kolkata According to Locality
locality_city_kolkata <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  filter(CITY %in% c("Kolkata")) %>%
  summarise(QUANTITY = n())

locality_city_kolkata

locality_city_kolkata1 <- locality_city_kolkata %>%
  group_by(QUANTITY) %>%
  summarise(COUNT = n())

locality_city_kolkata1

ggplot(locality_city_kolkata1, aes(x = QUANTITY, y = COUNT)) +
  geom_segment( aes(x = QUANTITY, xend = QUANTITY, y = 0, yend = COUNT), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(locality_city_kolkata1$QUANTITY)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = COUNT), color = "black", size = 3.5) +
  scale_x_continuous(breaks = seq(from = 0, to = 16, by = 1)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The House Distribution in Kolkata According to Locality")
```

Figure 90: Code to Find the House Distribution in Kolkata According to Locality

```
> locality_city_kolkata
# A tibble: 252 × 3
# Groups:   LOCALITY [252]
   LOCALITY                          CITY    QUANTITY
   <chr>                             <chr>      <int>
 1 2 BHK                             Kolkata        1
 2 700051                            Kolkata        1
 3 Action Area 1, Rajarhat Newtown   Kolkata        2
 4 Action Area 3, Rajarhat Newtown   Kolkata        1
 5 Agarpara                          Kolkata        4
 6 AH Block, Salt Lake               Kolkata        1
 7 Airport Area                      Kolkata        1
 8 Airport Area Behala               Kolkata        1
 9 Airport Near Gate No. 21/2        Kolkata        1
10 Ajoy Nagar                        Kolkata        2
# … with 242 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 91: Quantity of House Distribution in Each Locality in Kolkata

There is one locality with 16 houses available to rent in Kolkata, where is Salt Lake City Sector 2. 171 localities have only 1 house available to rent, which is 32.63% among the house availability in Kolkata.

```
> locality_city_kolkata1
# A tibble: 13 × 2
   QUANTITY COUNT
      <int> <int>
 1        1   171
 2        2    35
 3        3     9
 4        4    12
 5        5     4
 6        6     5
 7        7     4
 8        8     4
 9        9     2
10       10     1
11       13     2
12       14     2
13       16     1
```

Figure 92: Output of the Amount of House Availability in Each Locality in Kolkata on the Console

	LOCALITY	CITY	QUANTITY
1	Salt Lake City Sector 2	Kolkata	16
2	Behala	Kolkata	14
3	Salt Lake City Sector 1	Kolkata	14
4	Kasba	Kolkata	13
5	Salt Lake City Sector 5	Kolkata	13
6	Garia	Kolkata	10
7	Baghajatin	Kolkata	9
8	Salt Lake City	Kolkata	9
9	Bansdroni	Kolkata	8

Figure 93: Output of the Amount of House Availability in Each Locality in Kolkata in Table Format

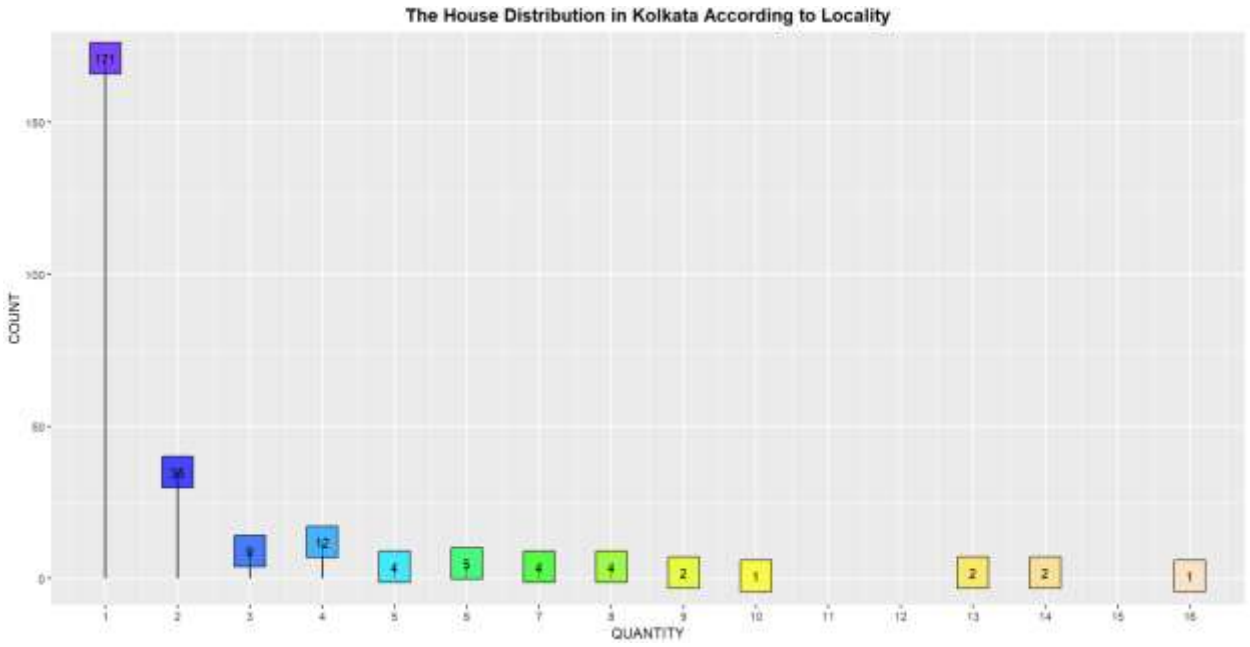

Figure 94: The House Distribution in Kolkata According to Locality in Lollipop Chart

3.1.3.2 Analysis 1.3.2: The House Distribution in Mumbai According to the Locality

The code below shows the house distribution in Mumbai, group by the locality. According to analysis 1.1, there are 972 houses available to rent in Mumbai.

```
#Analysis 1.3.2: The House Distribution in Mumbai According to Locality
locality_city_mumbai <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  filter(CITY %in% c("Mumbai")) %>%
  summarise(QUANTITY = n())

locality_city_mumbai

locality_city_mumbai1 <- locality_city_mumbai %>%
  group_by(QUANTITY) %>%
  summarise(COUNT = n())

locality_city_mumbai1

ggplot(locality_city_mumbai1, aes(x = QUANTITY, y = COUNT)) +
  geom_segment( aes(x = QUANTITY, xend = QUANTITY, y = 0, yend = COUNT), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(locality_city_mumbai1$QUANTITY)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = COUNT), color = "black", size = 3.5) +
  scale_x_continuous(breaks = seq(from = 0, to = 40, by = 5)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The House Distribution in Mumbai According to Locality")
```

Figure 95: Code to Find the House Distribution in Mumbai According to Locality

There are 604 locality in Mumbai.

```
> locality_city_mumbai
# A tibble: 604 × 3
# Groups:   LOCALITY [604]
   LOCALITY                                CITY   QUANTITY
   <chr>                                   <chr>     <int>
 1 117 Residency, Chembur East             Mumbai        1
 2 7 Bungalow, Seven Bungalows             Mumbai        3
 3 7 Bungalows Andheri West                Mumbai        1
 4 90 ft road                              Mumbai        1
 5 Acme Avenue, Kandivali West             Mumbai        1
 6 Adani Western Heights, 4 Bunglows       Mumbai        1
 7 Adani Western Heights, Andheri West     Mumbai        3
 8 Aditi Apartment, Andheri West           Mumbai        1
 9 Adityavardhan Apartment, Raheja Vihar   Mumbai        1
10 Agripada                                Mumbai        1
# … with 594 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 96: Quantity of House Distribution in Each Locality in Mumbai

The locality with the highest house available to rent in Mumbai is Bandra West, where there are 37 of houses available to rent.

```
> locality_city_mumbai1
# A tibble: 12 × 2
   QUANTITY COUNT
      <int> <int>
 1        1   454
 2        2    80
 3        3    33
 4        4    16
 5        5     3
 6        6     7
 7        7     5
 8        8     1
 9       12     2
10       15     1
11       19     1
12       37     1
```

Figure 97: Output of the Amount of House Availability in Each Locality in Mumbai on the Console

	LOCALITY	CITY	QUANTITY
1	Bandra West	Mumbai	37
2	Chembur	Mumbai	19
3	Andheri West	Mumbai	15
4	Goregaon West	Mumbai	12
5	Khar West	Mumbai	12
6	Chandivali	Mumbai	8
7	Ajmera Bhakti Park, Bhakti Park	Mumbai	7
8	Andheri East	Mumbai	7

Figure 98: Output of the Amount of House Availability in Each Locality in Mumbai in Table Format

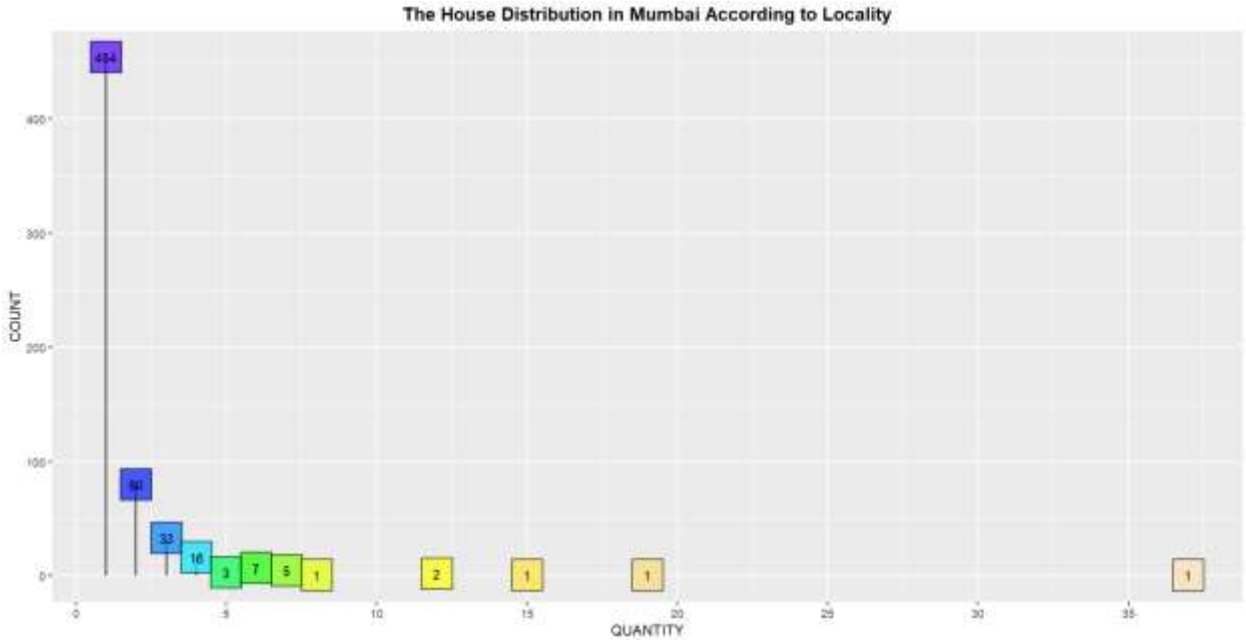

Figure 99: The House Distribution in Mumbai According to Locality in Lollipop Chart

3.1.3.3 Analysis 1.3.3: The House Distribution in Bangalore According to the Locality

The code below reveals the house distribution in Bangalore. There are 886 houses available to rent in Bangalore.

```
#Analysis 1.3.3: The House Distribution in Bangalore According to Locality
locality_city_bangalore <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  filter(CITY %in% c("Bangalore")) %>%
  summarise(QUANTITY = n())

locality_city_bangalore

locality_city_bangalore1 <- locality_city_bangalore %>%
  group_by(QUANTITY) %>%
  summarise(COUNT = n())

locality_city_bangalore1

ggplot(locality_city_bangalore1, aes(x = QUANTITY, y = COUNT)) +
  geom_segment( aes(x = QUANTITY, xend = QUANTITY, y = 0, yend = COUNT), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(locality_city_bangalore1$QUANTITY)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = COUNT), color = "black", size = 3.5) +
  scale_x_continuous(breaks = seq(from = 0, to = 25, by = 5)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The House Distribution in Bangalore According to Locality")
```

Figure 100: Code to Find the House Distribution in Bangalore According to Locality

```
> locality_city_bangalore
# A tibble: 429 × 3
# Groups:   LOCALITY [429]
   LOCALITY                                          CITY      QUANTITY
   <chr>                                             <chr>        <int>
 1 A Narayanapura, Mahadevapura                      Bangalore        2
 2 Aarna Enclave                                     Bangalore        1
 3 Abbiareddy Layout, Kaggadasapura                  Bangalore        1
 4 Abbigere                                          Bangalore        3
 5 Aditya Nagar-Vidyaranyapura, Vidyaranyapura       Bangalore        1
 6 Adugodi                                           Bangalore        2
 7 Aduru                                             Bangalore        1
 8 Aecs Layout-Singasandra, Singasandra, Hosur Road  Bangalore        3
 9 Agrahara Layout                                   Bangalore        1
10 Ags Layout, Hebbal                                Bangalore        1
# … with 419 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 101: Quantity of House Distribution in Each Locality in Bangalore

Electronic City has the most house available to rent in Bangalore.

```
> locality_city_bangalore1
# A tibble: 16 × 2
   QUANTITY COUNT
      <int> <int>
 1        1   278
 2        2    66
 3        3    32
 4        4    12
 5        5    15
 6        6     5
 7        7     4
 8        8     5
 9        9     1
10       10     1
11       11     2
12       12     4
13       13     1
14       14     1
15       19     1
16       24     1
```

Figure 102: Output of the Amount of House Availability in Each Locality in Bangalore on the Console

	LOCALITY	CITY	QUANTITY
1	Electronic City	Bangalore	24
2	K R Puram	Bangalore	19
3	Murugeshpalya, Airport Road	Bangalore	14
4	Mahadevapura	Bangalore	13
5	Hebbal	Bangalore	12
6	Ramamurthy Nagar	Bangalore	12
7	Vijayanagar	Bangalore	12
8	whitefield	Bangalore	12

Figure 103:Output of the Amount of House Availability in Each Locality in Bangalore in Table Format

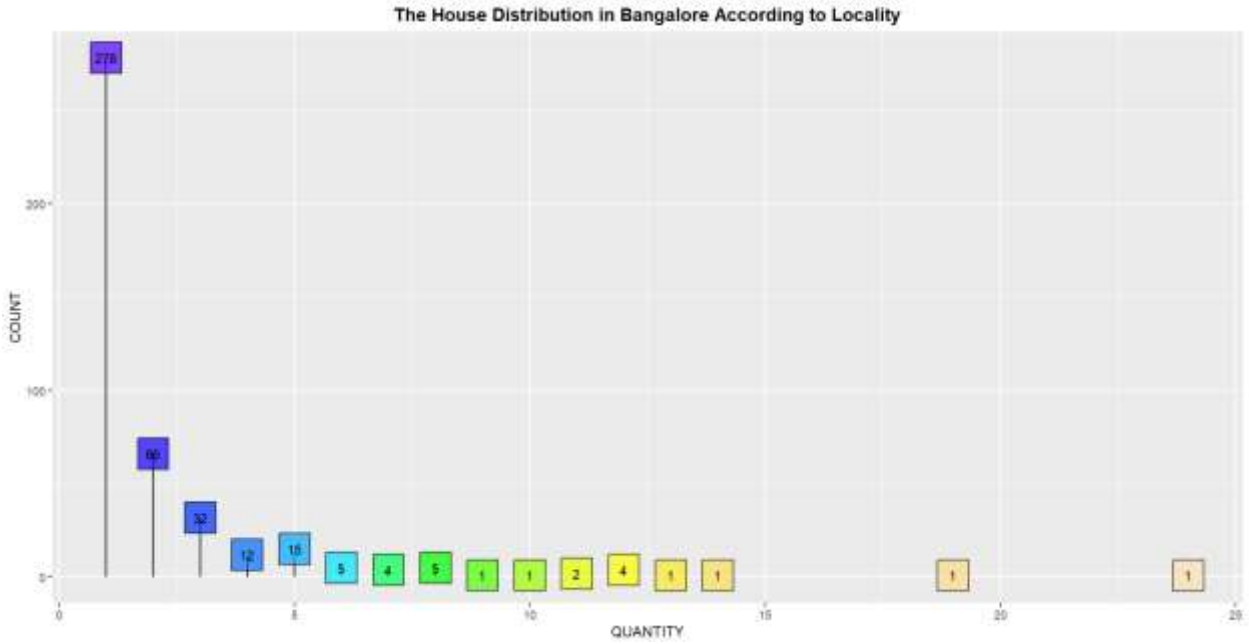

Figure 104: The House Distribution in Bangalore According to Locality in Lollipop Chart

3.1.3.4 Analysis 1.3.4: The House Distribution in Delhi According to the Locality

The code below illustrates the house distribution in Delhi. There are 605 houses available to rent in Bangalore according to the analysis 1.1.

```
#Analysis 1.3.4: The House Distribution in Delhi According to Locality
locality_city_delhi <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  filter(CITY %in% c("Delhi")) %>%
  summarise(QUANTITY = n())

locality_city_delhi

locality_city_delhi1 <- locality_city_delhi %>%
  group_by(QUANTITY) %>%
  summarise(COUNT = n())

locality_city_delhi1

ggplot(locality_city_delhi1, aes(x = QUANTITY, y = COUNT)) +
  geom_segment( aes(x = QUANTITY, xend = QUANTITY, y = 0, yend = COUNT), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(locality_city_delhi1$QUANTITY)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = COUNT), color = "black", size = 3.5) +
  scale_x_continuous(breaks = seq(from = 0, to = 22, by = 4)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The House Distribution in Delhi According to Locality")
```

Figure 105: Code to Find the House Distribution in Delhi According to Locality

```
> locality_city_delhi
# A tibble: 289 × 3
# Groups:   LOCALITY [289]
   LOCALITY                          CITY  QUANTITY
   <chr>                             <chr>    <int>
 1 Acharya Niketan, Mayur Vihar      Delhi        1
 2 Adarsh Nagar                      Delhi        1
 3 AGCR Enclave, Anand Vihar         Delhi        3
 4 Amar Colony, Lajpat Nagar         Delhi        2
 5 Amarpali Apartment                Delhi        1
 6 Anand Niketan                     Delhi        2
 7 Anand Parbat                      Delhi        1
 8 Arjun Garh, Aya Nagar             Delhi        1
 9 Arjun Nagar, Safdarjung Enclave   Delhi       10
10 Ashok Nagar                       Delhi        1
# … with 279 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 106: Quantity of House Distribution in Each Locality in Delhi

Laxmi Nagar has the highest number of houses available to rent in Delhi.

```
> locality_city_delhi1
# A tibble: 14 × 2
   QUANTITY COUNT
      <int> <int>
 1        1   176
 2        2    57
 3        3    21
 4        4    10
 5        5     4
 6        6     5
 7        7     1
 8        8     7
 9       10     3
10       11     1
11       12     1
12       13     1
13       14     1
14       19     1
```

Figure 107: Output of the Amount of House Availability in Each Locality in Delhi on the Console

	LOCALITY	CITY	QUANTITY
1	Laxmi Nagar	Delhi	19
2	Chhattarpur	Delhi	14
3	kst chattarpur Apartments	Delhi	13
4	Saket	Delhi	12
5	Vasant Kunj	Delhi	11
6	Arjun Nagar, Safdarjung Enclave	Delhi	10
7	Chhattarpur Enclave	Delhi	10
8	Paschim Vihar	Delhi	10

Figure 108: Output of the Amount of House Availability in Each Locality in Delhi in Table Format

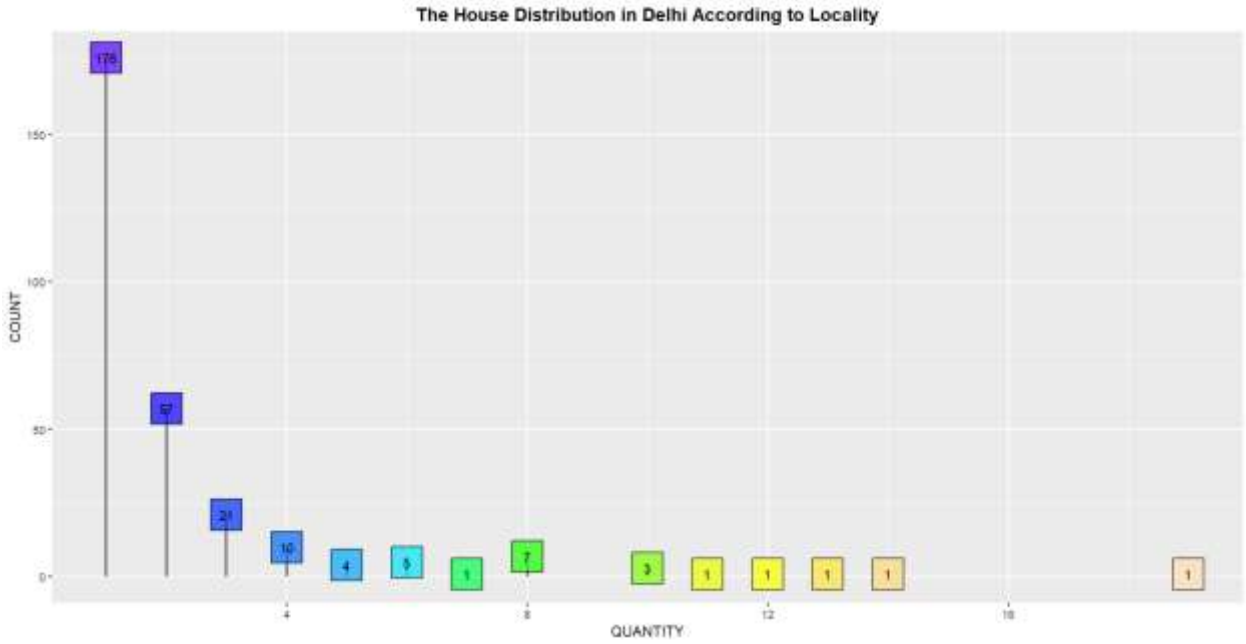

Figure 109: The House Distribution in Delhi According to Locality in Lollipop Chart

3.1.3.5 Analysis 1.3.5: The House Distribution in Chennai According to the Locality

The code below shows the house distribution in Chennai according to the locality. There are 891 houses available to rent in Chennai.

```
#Analysis 1.3.5: The House Distribution in Chennai According to Locality
locality_city_chennai <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  filter(CITY %in% c("Chennai")) %>%
  summarise(QUANTITY = n())

locality_city_chennai

locality_city_chennai1 <- locality_city_chennai %>%
  group_by(QUANTITY) %>%
  summarise(COUNT = n())

locality_city_chennai1

ggplot(locality_city_chennai1, aes(x = QUANTITY, y = COUNT)) +
  geom_segment( aes(x = QUANTITY, xend = QUANTITY, y = 0, yend = COUNT), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(locality_city_chennai1$QUANTITY)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = COUNT), color = "black", size = 3.5) +
  scale_x_continuous(breaks = seq(from = 0, to = 25, by = 5)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The House Distribution in Chennai According to Locality")
```

Figure 110: Code to Find the House Distribution in Chennai According to Locality

```
> locality_city_chennai
# A tibble: 323 × 3
# Groups:   LOCALITY [323]
   LOCALITY                          CITY    QUANTITY
   <chr>                             <chr>      <int>
 1 2nd Main Road                     Chennai        1
 2 355 konnur highroad Ayanavaram    Chennai        1
 3 5000                              Chennai        1
 4 58 block                          Chennai        1
 5 Abiramapuram                      Chennai        2
 6 Adambakkam                        Chennai        7
 7 Adyar, Sardar Patel Road          Chennai       10
 8 Adyar, Thiruvanmiyur, Chennai     Chennai        1
 9 AGS Colony-Velachery              Chennai        2
10 Alandur                           Chennai        2
# … with 313 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 111: Quantity of House Distribution in Each Locality in Chennai

Velachery has the highest amount of house available in Chennai.

```
> locality_city_chennai1
# A tibble: 19 × 2
   QUANTITY COUNT
      <int> <int>
 1        1   184
 2        2    45
 3        3    23
 4        4    15
 5        5    14
 6        6     7
 7        7     8
 8        8     8
 9        9     2
10       10     3
11       11     3
12       12     1
13       13     1
14       14     2
15       15     2
16       16     1
17       17     2
18       20     1
19       22     1
```

Figure 112: Output of the Amount of House Availability in Each Locality in Chennai on the Console

	LOCALITY	CITY	QUANTITY
1	Velachery	Chennai	22
2	Madipakkam	Chennai	20
3	Iyyappanthangal	Chennai	17
4	Medavakkam	Chennai	17
5	Sholinganallur	Chennai	16
6	Chromepet, GST Road	Chennai	15
7	Vadapalani	Chennai	15
8	Ambattur	Chennai	14

Figure 113: Output of the Amount of House Availability in Each Locality in Chennai in Table Format

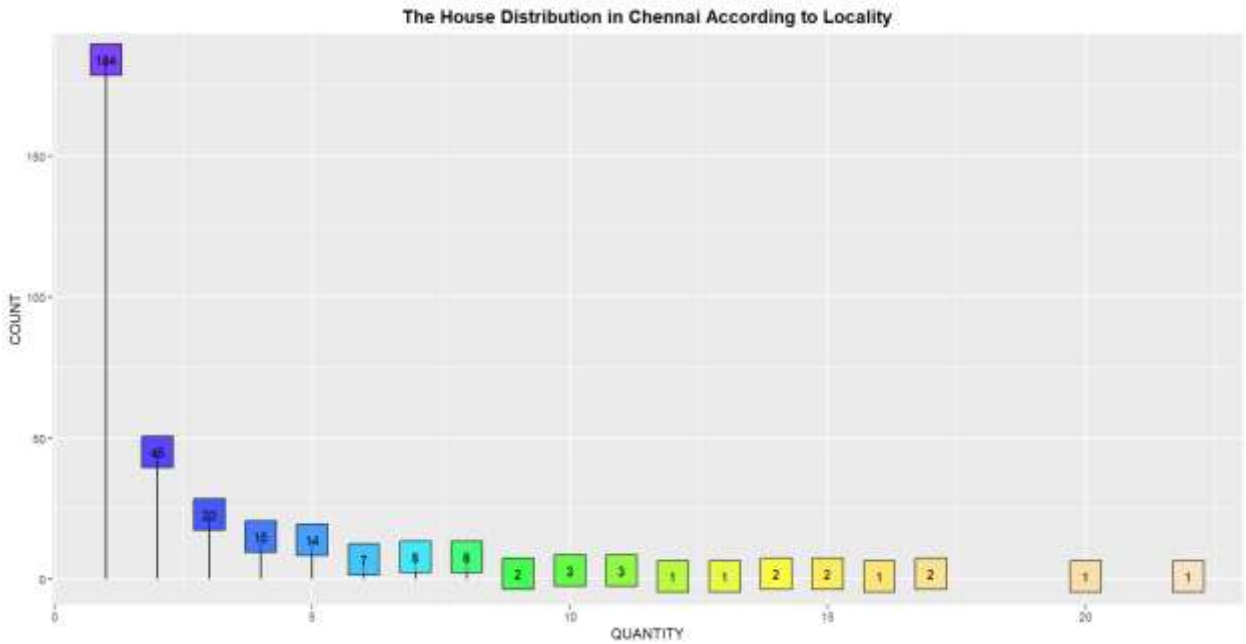

Figure 114: The House Distribution in Chennai According to Locality in Lollipop Chart

3.1.3.6 Analysis 1.3.6: The House Distribution in Hyderabad According to the Locality

The code below illustrates the house distribution in Hyderabad according to the locality. There are 868 houses available to rent in Hyderabad.

```
#Analysis 1.3.6: The House Distribution in Hyderabad According to Locality
locality_city_hyderabad <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  filter(CITY %in% c("Hyderabad")) %>%
  summarise(QUANTITY = n())

locality_city_hyderabad

locality_city_hyderabad1 <- locality_city_hyderabad %>%
  group_by(QUANTITY) %>%
  summarise(COUNT = n())

locality_city_hyderabad1

ggplot(locality_city_hyderabad1, aes(x = QUANTITY, y = COUNT)) +
  geom_segment( aes(x = QUANTITY, xend = QUANTITY, y = 0, yend = COUNT), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(locality_city_hyderabad1$QUANTITY)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = COUNT), color = "black", size = 3.5) +
  scale_x_continuous(breaks = seq(from = 0, to = 30, by = 5)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The House Distribution in Hyderabad According to Locality")
```

Figure 115: Code to Find the House Distribution in Hyderabad According to Locality

```
> locality_city_hyderabad
# A tibble: 343 × 3
# Groups:   LOCALITY [343]
   LOCALITY                                  CITY      QUANTITY
   <chr>                                     <chr>        <int>
 1 " Beeramguda, Ramachandra Puram, NH 9"    Hyderabad        1
 2 " in Boduppal, NH 2 2"                    Hyderabad        1
 3 " in Erragadda, NH 9"                     Hyderabad        1
 4 " in Miyapur, NH 9"                       Hyderabad        1
 5 "5-20 Adharshnagar"                       Hyderabad        1
 6 "A 307 Blossom Heights"                   Hyderabad        1
 7 "Abhyudaya Nagar"                         Hyderabad        1
 8 "Abids, NH 7"                             Hyderabad        1
 9 "Adibatla"                                Hyderabad        1
10 "Adikmet"                                 Hyderabad        1
# … with 333 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 116: The Quantity of House Distribution in Each Locality in Hyderabad

Gachibowli has 29 houses to rent in Hyderabad, which is the highest among the other locality.

```
> locality_city_hyderabad1
# A tibble: 17 × 2
   QUANTITY COUNT
      <int> <int>
 1        1   207
 2        2    47
 3        3    26
 4        4    14
 5        5    12
 6        6     8
 7        7     7
 8        8     3
 9        9     3
10       10     1
11       11     8
12       13     1
13       14     2
14       17     1
15       18     1
16       22     1
17       29     1
```

Figure 117: Output of the Amount of House Availability in Each Locality in Hyderabad on the Console

	LOCALITY	CITY	QUANTITY
1	Gachibowli	Hyderabad	29
2	Miyapur, NH 9	Hyderabad	22
3	Kondapur	Hyderabad	18
4	Banjara Hills, NH 9	Hyderabad	17
5	Attapur	Hyderabad	14
6	Manikonda, Outer Ring Road	Hyderabad	14
7	Kukatpally, NH 9	Hyderabad	13
8	Bandlaguda Jagir	Hyderabad	11
9	Boduppal, NH 2 2	Hyderabad	11

Figure 118: Output of the Amount of House Availability in Each Locality in Hyderabad in Table Format

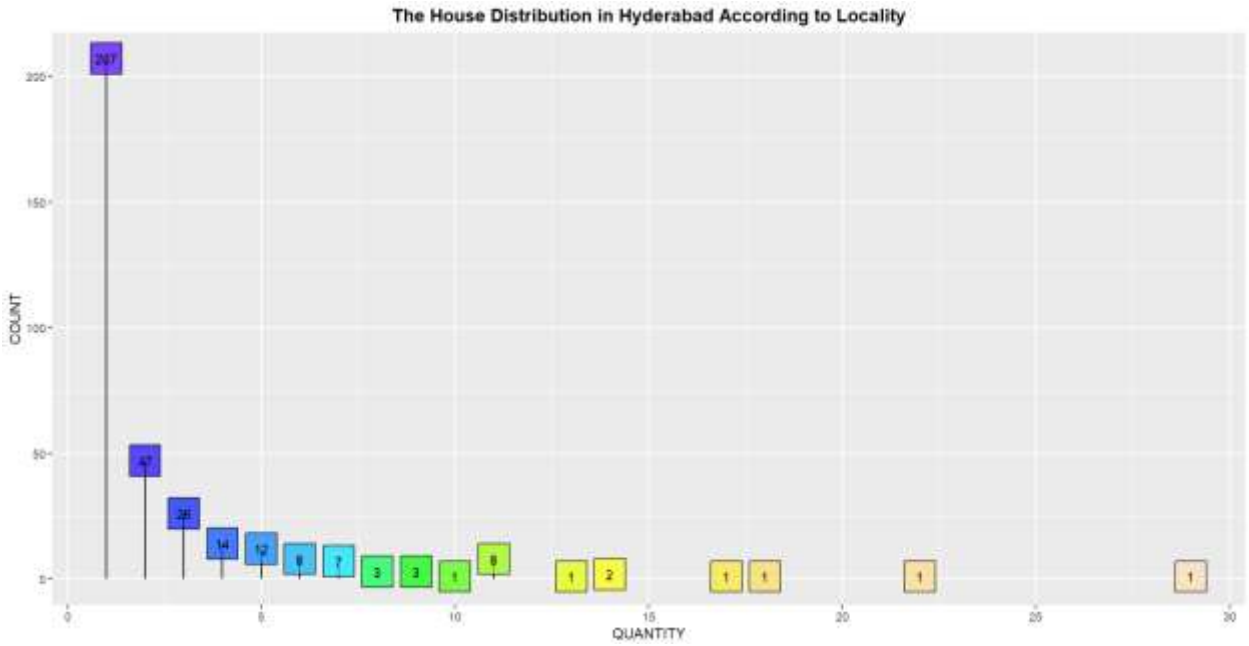

Figure 119: The House Distribution in Hyderabad According to Locality in Lollipop Chart

3.1.3.7 Conclusion for Analysis 1.3

According to each locality in each city, most of the localities has one houses available to rent. However, Bandra West in Mumbai has the highest house available to rent among all the localities and cities, which is 37 houses available.

3.1.4 Analysis 1.4: Average House Rent According to the City

The code below shows the average house rent in each city.

```
#Analysis 1.4: Average House Rent According to City
avg_rent_city <- rent %>%
  group_by(CITY) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_city

#Bar Plot
ggplot(avg_rent_city, aes(x = CITY, y = AVG)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(avg_rent_city$CITY))) +
  ggtitle("Average House Rent According to City") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(CITY, label = AVG), position = position_dodge(width = 0.1))
```

Figure 120: Code to Find the Average House Rent According to the City

Figure 121 shows the average house rent according to the city. Mumbai has the highest average house rent among the six cities. On the other hand, Kolkata has the lowest average house rent. The difference between Mumbai's and Kolkata's rent is 73,676. It could prove that the supply in Mumbai is more popular compared to Kolkata. The four cities' average rent is nearby and does not have big differences compared to others.

```
> avg_rent_city
# A tibble: 6 x 2
  CITY      AVG
  <chr>     <chr>
1 Bangalore 24966.4
2 Chennai   21614.1
3 Delhi     29462.0
4 Hyderabad 20555.0
5 Kolkata   11645.2
6 Mumbai    85321.2
```

Figure 121: Output of the Average House Rent According to the City

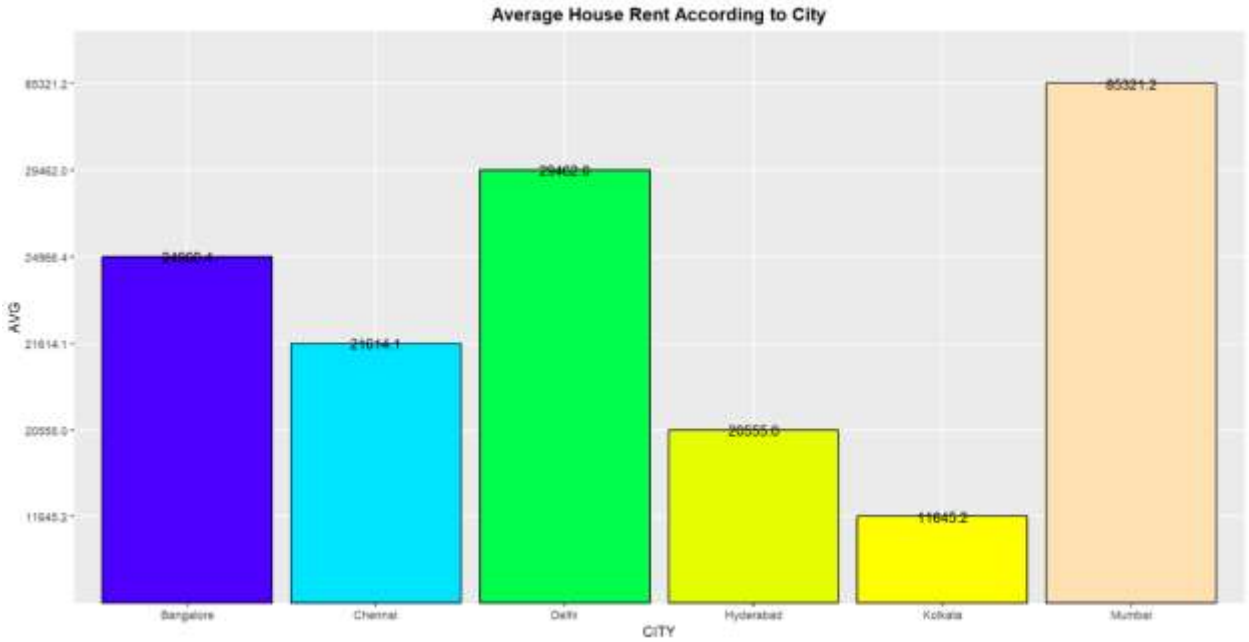

Figure 122: Average House Rent According to the City in Bar Plot

3.1.5 Analysis 1.5: The Top 10 House Availability to Rent According to Area Locality Distribution in Cities

The code below shows the top 10 house availability to rent according to area locality distribution in cities. There is an additional features where is abline() function where it can add lines on the graph. It can customize the value, color and size of the line.

```
#Analysis 1.5: The Top 10 House Availability to Rent According to Area Locality Distribution in Cities
# Suppress summaries info
options(dplyr.summarise.inform = FALSE)

locality_city <- rent %>%
  select(LOCALITY,CITY) %>%
  group_by(LOCALITY,CITY) %>%
  summarise(QUANTITY = n())
locality_city

top10_locality = head(arrange(locality_city,desc(QUANTITY), .group = "drop"),10)
par(mar=c(7,5,5,5))
top10_locality_bar <- barplot(top10_locality$QUANTITY ,ylab = "House Availablity",
                              border=F , names.arg=top10_locality$LOCALITY,
                  las=2 ,
                  col=c("#4C00FF","#0080FF","#00FF4D","#0080FF","#E6FF00",
                        "#E6FF00","#4C00FF","#00FF4D","#FFE53C","#0080FF") ,
                  ylim=c(0,40) ,
                  main="The Top 10 Highest House Availability to Rent According
                  to Area Locality Distribution in Cities" ) +
  abline(v=c(3.7 , 7.3 ,10.9 ) , col="grey")

  top10_locality_bar <- legend("topright", legend = c(unique(top10_locality$CITY)) ,
       col = c("#4C00FF","#0080FF","#00FF4D","#E6FF00","#FFE53C") ,
       bty = "n", pch=20 , pt.cex = 2, cex = 0.5, horiz = FALSE, inset = c(- 0.22, 0))
```

Figure 123: Code to Find the Top 10 House Availability to Rent According to Area Locality Distribution in Cities

From the figure below, Bandra West in Mumbai has the highest house available to rent, followed by Gachibowli in Hyderabad.

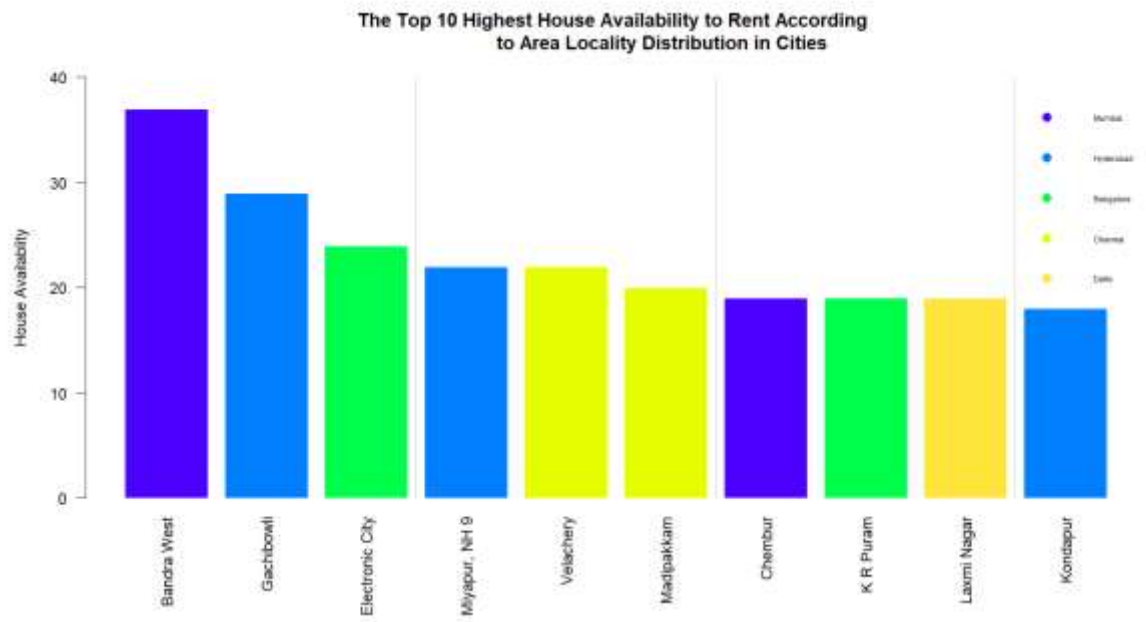

Figure 124: Output of Top 10 House Availability to Rent According to Area Locality Distribution in Cities in Bar Plot

3.1.6 Conclusion of Question 1

From the analysis above, Mumbai has the highest house rent and the average house rent in Mumbai is high compared to the others. If the user prefers the surrounding in Mumbai and willing to pay a higher rent to stay in Mumbai, Mumbai has many houses for rent.

On the other perspective, although Kolkata has lesser rent and the house available to rent is low compared to the others, there may be other conditions which will be discuss in the further analysis where it will attract the user to rent the house.

3.2 Question 2: How does the house condition affect the house rent?

3.2.1 Analysis 2.1: How does the house condition affect the house rent?

3.2.1.1 Analysis 2.1.1: The Quantity of Bedrooms, Halls and Kitchen

The code below shows the quantity of the bedrooms, halls and kitchen.

```
#Question 2: How does the house condition affect the house rent ?
#Analysis 2.1: How does the quantity of BHK affect the house rent ?
#Analysis 2.1.1: The Quantity of Bedrooms, Halls and Kitchen
bhk_num <- rent %>%
  group_by(BHK) %>%
  summarise(QUANTITY = n())
bhk_num

#Bar Plot
ggplot(bhk_num, aes(x = BHK, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black", fill = topo.colors(length(bhk_num$BHK))) +
  ggtitle("The Quantity of Bedrooms, Halls and Kitchen") +
  scale_x_continuous(breaks = seq(from = 0, to = 6, by = 1)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(BHK, label = QUANTITY), position = position_dodge(width = 0.1))
```

Figure 125: Code to Find the Quantity of Bedrooms, Halls and Kitchen

Mostly houses have at least 2 BHK and the highest BHK goes up to 6 BHK.

```
> bhk_num
# A tibble: 6 × 2
    BHK QUANTITY
  <int>    <int>
1     1     1167
2     2     2265
3     3     1098
4     4      189
5     5       19
6     6        8
```

Figure 126: Output of the Quantity of Bedrooms, Halls and Kitchen on the Console

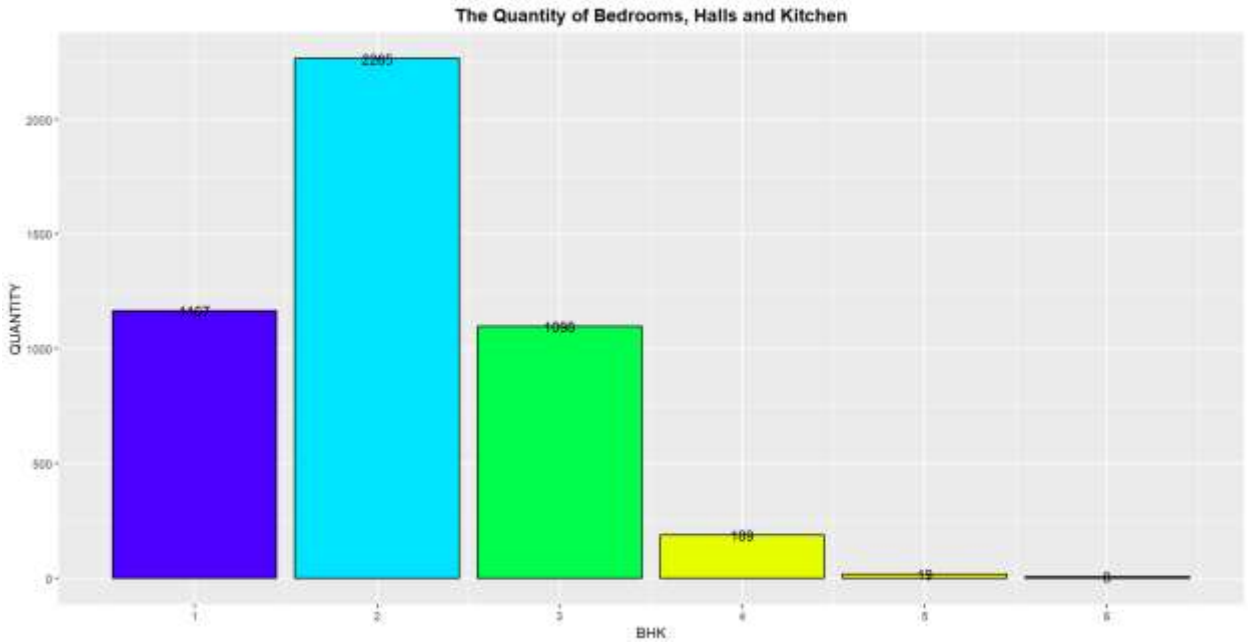

Figure 127: The Quantity of Bedrooms, Halls and Kitchen in Bar Plot

3.2.1.2 Analysis 2.1.2: Average Rent according to BHK

The code below shows the average Rent according to the number of the bedrooms, halls and kitchen.

```
#Analysis 2.1.2: Average Rent according to BHK
avg_rent_bhk <- rent %>%
  group_by(BHK) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_bhk$AVG <- as.integer(avg_rent_bhk$AVG)
avg_rent_bhk <- arrange(avg_rent_bhk, AVG)

ggplot(avg_rent_bhk, aes(x = BHK, y = AVG)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(avg_rent_bhk$BHK))) +
  scale_x_continuous(breaks = seq(from = 0, to = 6, by = 1)) +
  ggtitle("Average House Rent According to BHK") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(BHK, label = AVG), position = position_dodge(width = 0.1))
```

Figure 128: Code to Find the Average Rent according to BHK

Aside from the house that have 6 BHK, if the number of the BHK increase, the higher the house rent will be.

```
> avg_rent_bhk
# A tibble: 6 × 2
    BHK    AVG
  <int>  <int>
1     1  14139
2     2  22113
3     3  55863
4     6  73125
5     4 168864
6     5 297500
```

Figure 129: Output of the Average Rent according to BHK on the Console

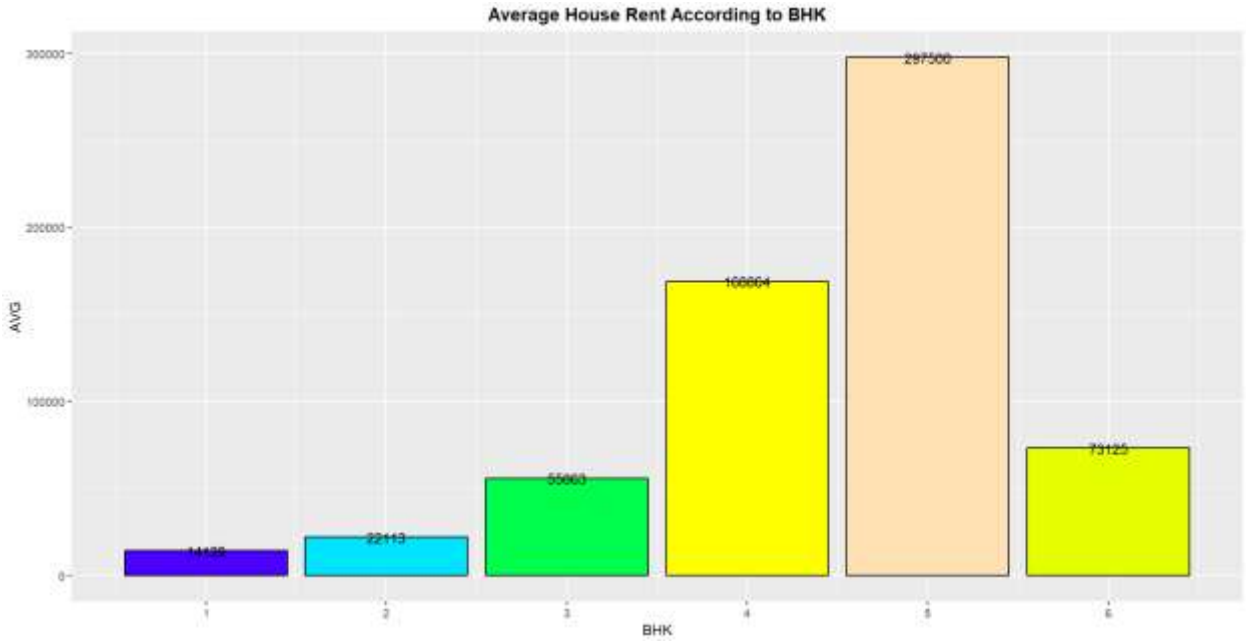

Figure 130: The Average Rent according to BHK in Bar Plot

3.2.2 Analysis 2.2: How does the quantity of Bathroom affect the house rent?

3.2.2.1 Analysis 2.2.1: The Quantity of Bathroom

The code below calculates the quantity of bathroom.

```
#Analysis 2.2: How does the quantity of Bathroom affect the house rent ?
#Analysis 2.2.1: The Quantity of Bathroom
bath_num <- rent %>%
  group_by(BATHROOM) %>%
  summarise(QUANTITY = n())
bath_num

ggplot(bath_num, aes(x = BATHROOM, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(bath_num$BATHROOM))) +
  ggtitle("The Quantity of Bathrooms") +
  scale_x_continuous(breaks = seq(from = 0, to = 10, by = 1)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(BATHROOM, label = QUANTITY), position = position_dodge(width = 0.1))
```

Figure 131: Code to Find the Quantity of Bathroom

Mostly houses come with 2 bathrooms.

```
> bath_num
# A tibble: 8 × 2
  BATHROOM QUANTITY
     <int>    <int>
1        1     1474
2        2     2291
3        3      749
4        4      156
5        5       60
6        6       12
7        7        3
8       10        1
```

Figure 132: Output of the Quantity of Bathroom on the Console

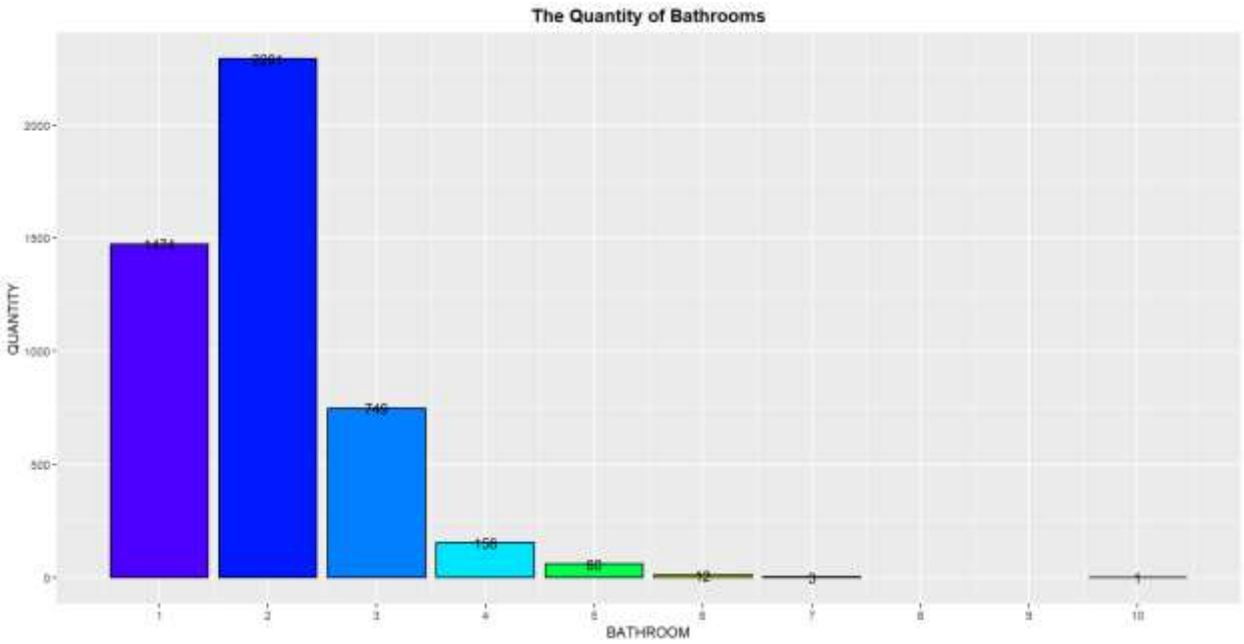

Figure 133: The Quantity of Bathrooms in Bar Plot

3.2.2.2 Analysis 2.2.2: Average Rent according to Bathroom

The code below shows the average rent according to the number of bathrooms.

```
#Analysis 2.2.2: Average Rent according to Bathroom
avg_rent_bath <- rent %>%
  group_by(BATHROOM) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_bath$AVG <- as.integer(avg_rent_bath$AVG)

plot(avg_rent_bath$AVG, type = "o", xlab = "Bathroom", ylab = "Average Rental",
     main = "Average Rent according to Bathroom", col = "black" )
```

Figure 134: Code to Find the Average Rent according to Bathrooms

The house that contains 5 bathrooms has the highest average rent and the houses that come with 1 bathroom has the lowest average rent.

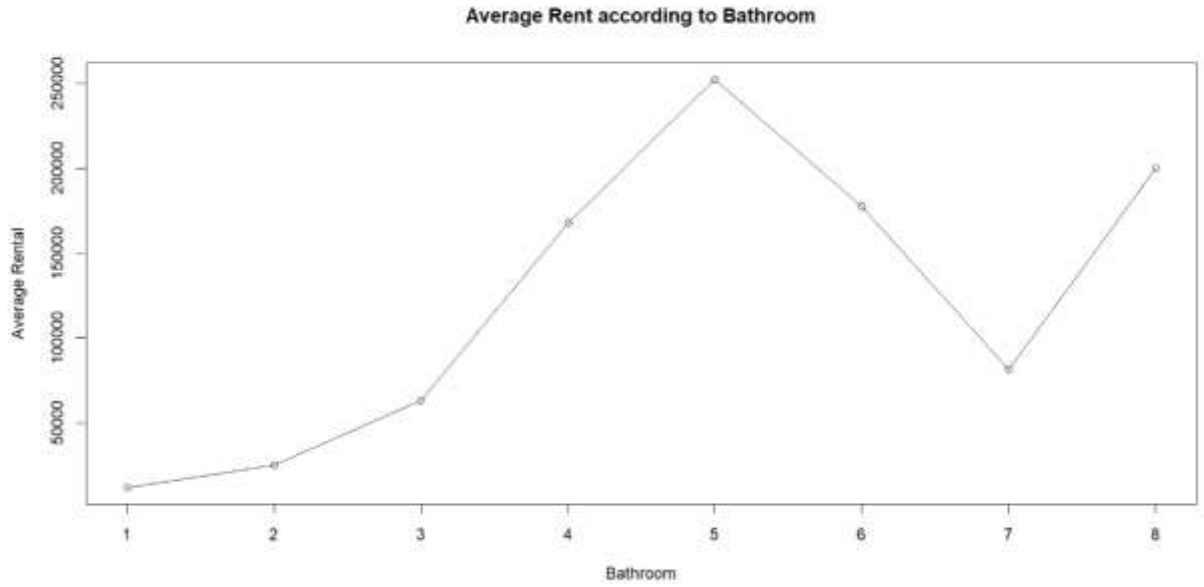

Figure 135: The average rent according to the number of bathrooms

3.2.3 Analysis 2.3: How the numbers of rooms affect the rent?

The code below shows the numbers of room and the average rent according to the rooms available. The plot() function is used to draw the line graph.

```
#Analysis 2.3: How the numbers of rooms affect the rent ?
label <- c("bath", "bhk")
par(mar = c(2, 2, 2, 2))
plot1 <- plot(avg_rent_bath$AVG, type = "l", xlab = "Rooms", ylab = "Average Rental",
              main = "Average Rent according to Rooms Available", col = "blue")
plot1 <- lines(avg_rent_bhk$AVG, type = "l", xlab = "Rooms", ylab = "Average Rental",
              main = "Average Rent according to Rooms Available", col = "green")
plot1 <- legend("topright", legend = label, cex = 0.8, fill = c("blue", "green"))
```

Figure 136: Code to Find the Average Rent according to the Rooms Available

The intersection points of the bathrooms line and bhk line lies on approximately 5.4, where we assume it as 5 rooms in total. The average rent for 5 to 6 rooms is between 200000 to 250000. The house comes with 1 room and 1 bathroom has the lowest average rent among all.

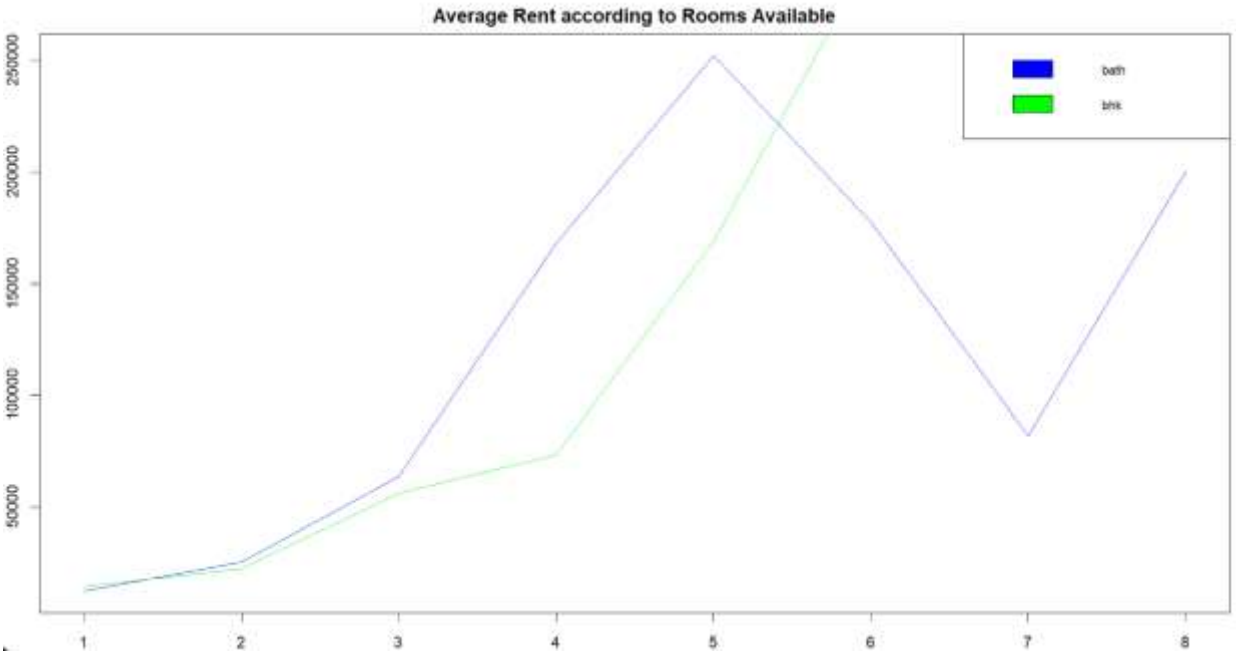

Figure 137: The Average Rent according to the Rooms Available in Line Graph

3.2.4 Analysis 2.4: How does the type of furnishing status affect the house rent?

3.2.4.1 Analysis 2.4.1: The Total Quantity of Each Type of Furnishing Status

The code below illustrates the total quantity of each type of furnishing status.

```
#Analysis 2.4: How does the type of furnishing status affect the house rent ?
#Analysis 2.4.1: The Total Quantity of Each Type of Furnishing Status
house_furnish <- rent %>%
  group_by(FURNISHING_STATUS) %>%
  summarise(QUANTITY = n())
house_furnish

#Pie Chart
pie(house_furnish$QUANTITY, paste(house_furnish$FURNISHING_STATUS,house_furnish$QUANTITY),
    radius = 0.7, main = "House Available to Rent Distribution According to Furnishing Status",
    col = topo.colors(length(house_furnish$FURNISHING_STATUS)), clockwise =  TRUE)
legend("topright", house_furnish$FURNISHING_STATUS, fill =
         topo.colors(length(house_furnish$FURNISHING_STATUS)),
       pt.cex = 2, cex = 0.7, horiz = FALSE, inset = c( -0.3 , 0.35))
par(mar = c(2, 2, 2, 2))

#Bar Chart
ggplot(house_furnish, aes(x = FURNISHING_STATUS, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black", fill =
             topo.colors(length(house_furnish$FURNISHING_STATUS))) +
  ggtitle("The Quantity of Furnishing Status") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(FURNISHING_STATUS, label = QUANTITY), cex = 5, vjust = 1.5)
```

Figure 138: Code to Show the Total Quantity of Each Type of Furnishing Status

Most of the houses is semi-furnished.

```
> house_furnish
# A tibble: 3 × 2
  FURNISHING_STATUS QUANTITY
  <chr>                <int>
1 Furnished              680
2 Semi-Furnished        2251
3 Unfurnished           1815
```

Figure 139: Output of the Total Quantity of Each Type of Furnishing Status on the Console

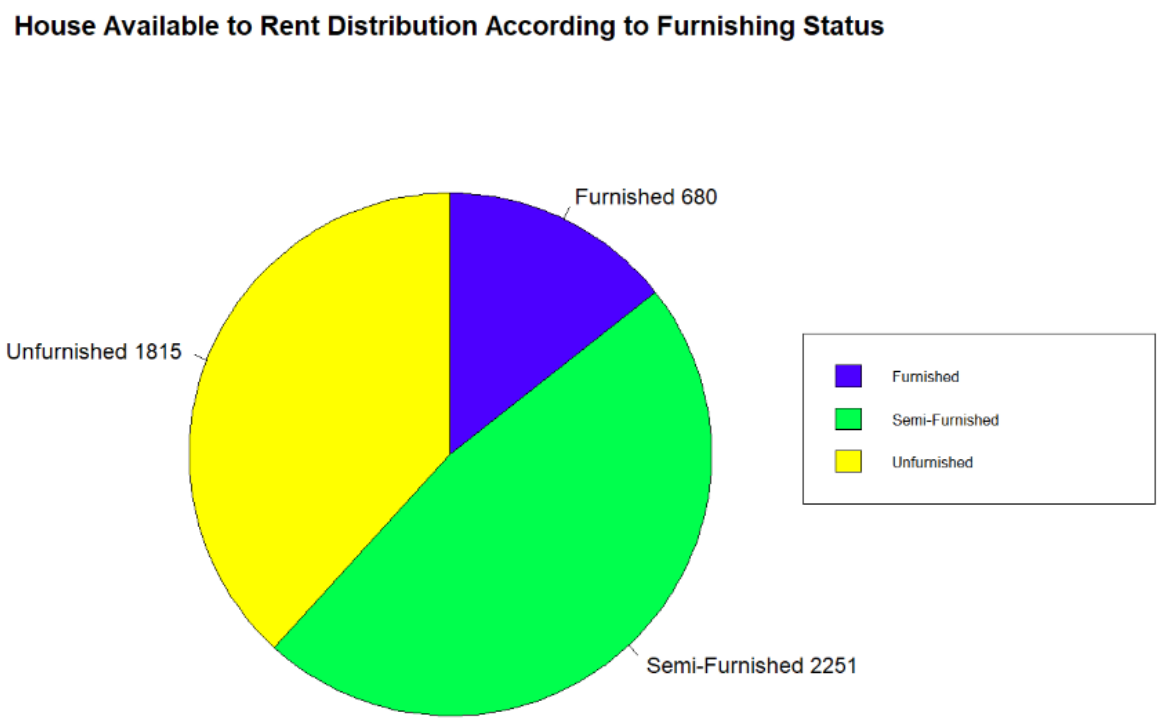

Figure 140: House Available to Rent Distribution According to Furnishing Status in Pie Chart

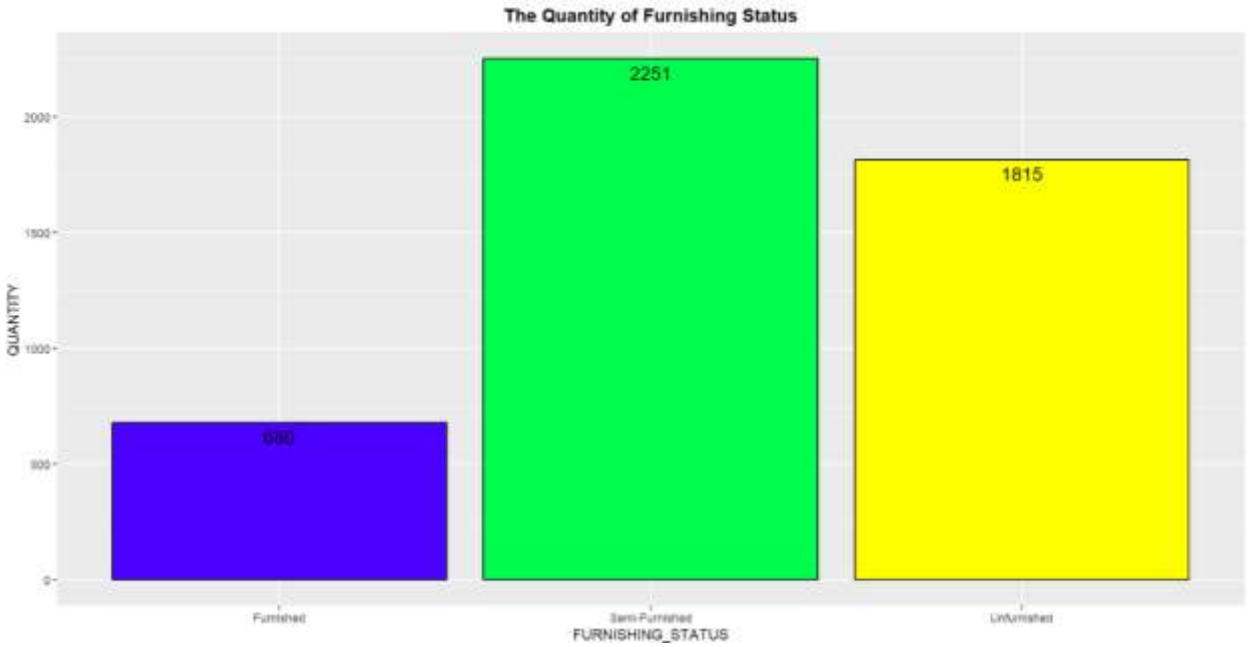

Figure 141: House Available to Rent Distribution According to Furnishing Status in Bar Plot

3.2.4.2 Analysis 2.4.2: Average Rent according to Furnishing Status

```
#Analysis 2.4.2: Average Rent according to Furnishing Status
avg_rent_furnish <- rent %>%
  group_by(FURNISHING_STATUS) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_furnish$AVG <- as.integer(avg_rent_furnish$AVG)
avg_rent_furnish <- arrange(avg_rent_furnish, AVG)

ggplot(avg_rent_furnish, aes(x = FURNISHING_STATUS, y = AVG)) +
  scale_y_continuous(breaks = seq(from = 0, to = 60000,by = 10000)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(avg_rent_furnish$FURNISHING_STATUS))) +
  ggtitle("Average House Rent According to Furnishing Status") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(FURNISHING_STATUS, label = AVG),  cex = 5, vjust = 1.5)
```

Figure 142: Code to Show Average Rent according to Furnishing Status

Furnished house cost higher than the semi-furnished and unfurnished house.

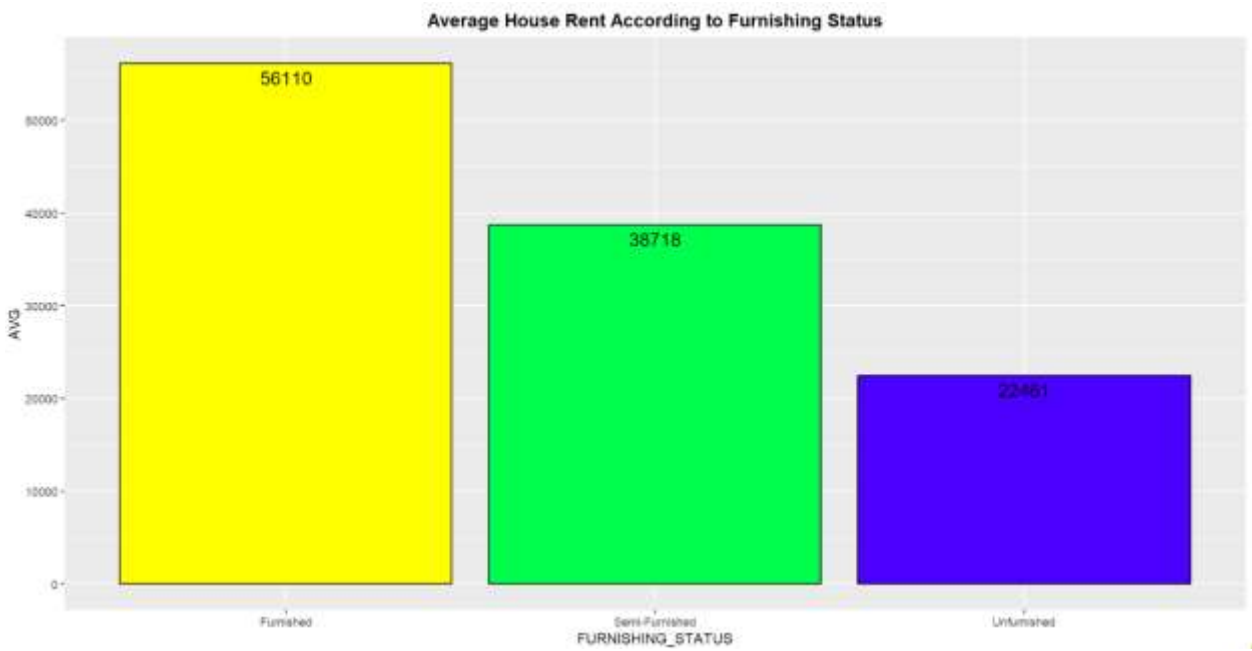

Figure 143: Average House Rent According to Furnishing Status in Bar Plot

3.2.5 Analysis 2.5: How does the number of floor available affect the house rent?

3.2.5.1 Analysis 2.5.1: The Total Quantity of Floor Available

```
#Analysis 2.5: How does the number of floor available affect the house rent ?
#Analysis 2.5.1: The Total Quantity of Floor Available
house_floor_available <- rent %>%
  group_by(FLOOR_AVAILABLE) %>%
  summarise(QUANTITY = n())
house_floor_available

par(mar = c(4.5, 4.5, 3, 3))
plot(house_floor_available$FLOOR_AVAILABLE, house_floor_available$QUANTITY,
     main ="The Total Quantity of Floor Available", xlab="Floor Available",
     ylab="Quantity", col = "#4C00FF",pch = 18, cex = 2)
```

Figure 144: Code to Show the Total Quantity of Floor Available

The supply for first floor houses is higher compared to the others, where the second floor is the second highest along with ground floor comes at the third.

```
> house_floor_available
# A tibble: 54 × 2
   FLOOR_AVAILABLE QUANTITY
             <dbl>    <int>
 1               0      927
 2            0.25       11
 3            0.75       23
 4               1     1161
 5               2      945
 6               3      512
 7               4      272
 8               5      164
 9               6       93
10               7       74
# … with 44 more rows
# i Use `print(n = ...)` to see more rows
```

Figure 145: Output of the Total Quantity of Floor Available on the Console

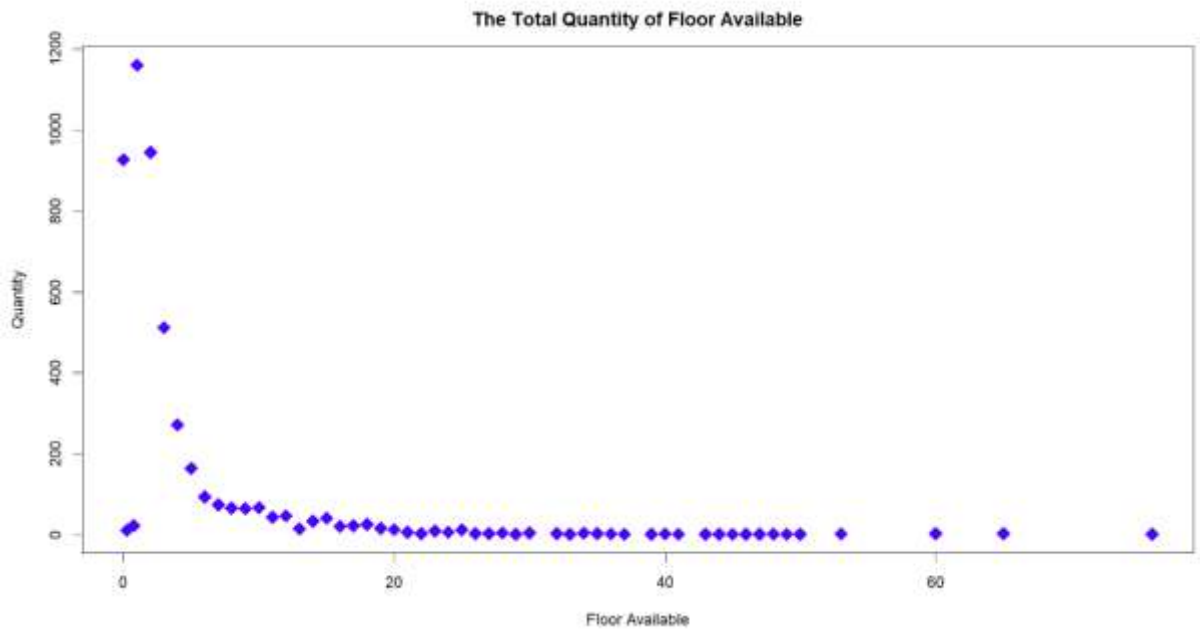

Figure 146: The Total Quantity of Floor Available in Scatterplot

3.2.5.2 Analysis 2.5.2: Average Rent according to Floor Available

```
#Analysis 2.5.2: Average Rent according to Floor Available
avg_rent_floor_available <- rent %>%
  group_by(FLOOR_AVAILABLE) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_floor_available$AVG <- as.integer(avg_rent_floor_available$AVG)

summary(avg_rent_floor_available)

par(mar = c(5, 5, 4, 4))
plot(avg_rent_floor_available$AVG, type = "o", xlab = "Floor Available",
     ylab = "Average Rental",
     main = "Average Rent according to Floor Available", col = "#7845E3",
     pch = 17, cex = 2, lwd = 1.5)
abline(v=c(11.25, 24.50, 39.75) , col="#ABD1FB")
abline(h=c(16958, 71046 , 95516, 195563, 380000) , col="#C9B6D8")
```

Figure 147: Code to Find the Average Rent according to the Floor Available

```
> summary(avg_rent_floor_available)
 FLOOR_AVAILABLE       AVG
 Min.   : 0.00   Min.   : 16958
 1st Qu.:11.25   1st Qu.: 71046
 Median :24.50   Median : 95516
 Mean   :26.28   Mean   :136275
 3rd Qu.:39.75   3rd Qu.:195563
 Max.   :76.00   Max.   :380000
```

Figure 148: Summary of the Average Rent According to the Floor Available on the Console

More houses with lower floor available falls below the mean of the average rental.

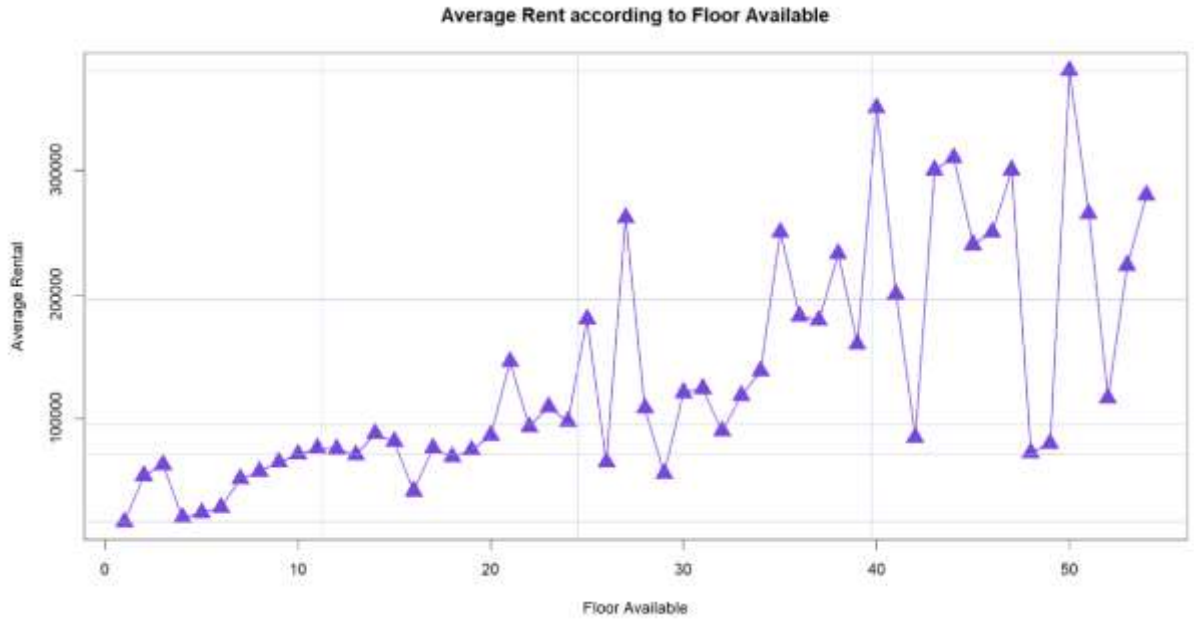

Figure 149: Average Rent According to the Floor Available in Connected Scatterplot

3.2.6 Analysis 2.6: How does the number of total floor available affect the house rent?

3.2.6.1 Analysis 2.6.1: The Total Quantity of Total Floor

```
#Analysis 2.6: How does the number of total floor available affect the house rent ?
#Analysis 2.6.1: The Total Quantity of Total Floor
house_total_floor <- rent %>%
  group_by(TOTAL_FLOOR) %>%
  summarise(QUANTITY = n())
house_total_floor

par(mar = c(4.5, 4.5, 3, 3))
plot(house_total_floor$TOTAL_FLOOR, house_total_floor$QUANTITY,
     main ="The Total Quantity of Total Floor", xlab="Total Floor",
     ylab="Quantity", col = "#0080FF", pch = 9, cex = 2)
```

Figure 150: Code to Find the Total Quantity of Total Floor

The houses with total floor of 2,3,4 are the most popular houses among the all.

```
> house_total_floor
# A tibble: 67 × 2
   TOTAL_FLOOR QUANTITY
         <dbl>    <int>
 1           0        1
 2           1      335
 3           2      868
 4           3      915
 5           4      938
 6           5      422
 7           6       95
 8           7      170
 9           8       87
10           9       40
# … with 57 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figure 151: Output of the Total Quantity of Total Floor on the Console

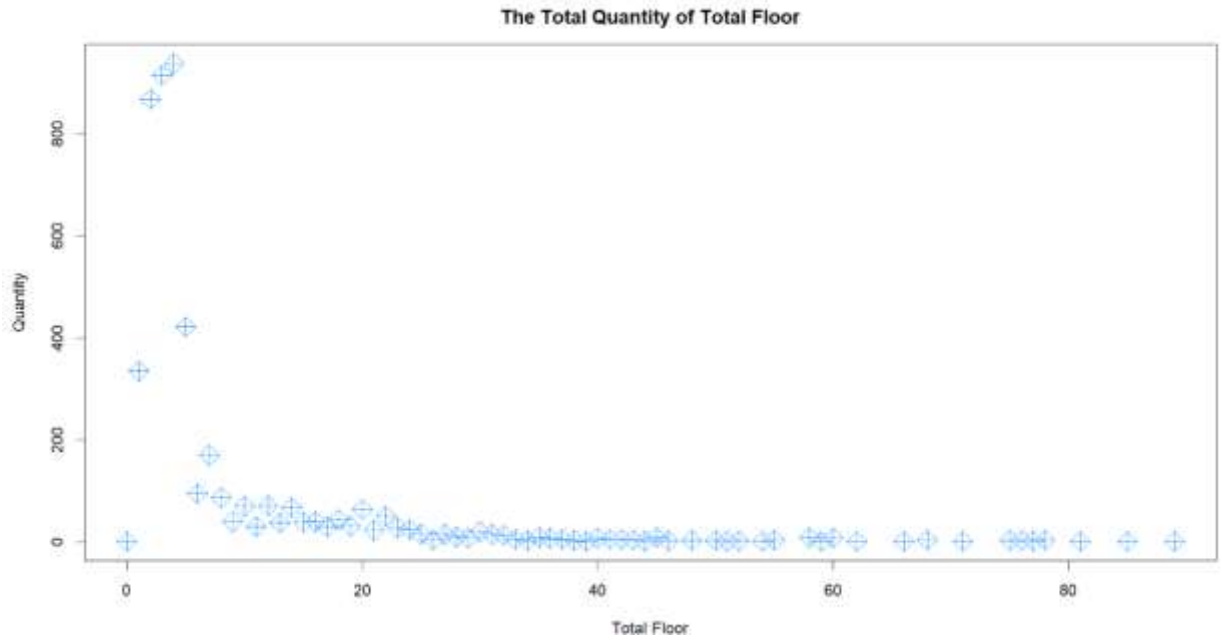

Figure 152: The Total Quantity of Total Floor in Scatterplot

3.2.6.2 Analysis 2.6.2: Average Rent according to Total Floor

```
#Analysis 2.6.2: Average Rent according to Total Floor
avg_rent_total_floor <- rent %>%
  group_by(TOTAL_FLOOR) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_total_floor$AVG <- as.integer(avg_rent_total_floor$AVG)

summary(avg_rent_total_floor)

par(mar = c(5, 5, 4, 4))
plot(avg_rent_total_floor$AVG, type = "o", xlab = "Total Floor",
     ylab = "Average Rental",
     main = "Average Rent according to Total Floor", col = "#4C00FF", pch = 18, cex = 2, lwd = 1.5)
abline(v=c(16.5, 33, 51.5) , col="#F8D8FC")
abline(h=c(15533, 65286 , 82915, 140000, 380000) , col="#BED0FB")
```

Figure 153: Code to Find the Average Rent According to the Total Floor

Most of the houses with lower total floor cost lower rent.

```
> summary(avg_rent_total_floor)
  TOTAL_FLOOR          AVG
 Min.   : 0.00   Min.   : 15533
 1st Qu.:16.50   1st Qu.: 65286
 Median :33.00   Median : 82915
 Mean   :35.76   Mean   :113219
 3rd Qu.:51.50   3rd Qu.:140000
 Max.   :89.00   Max.   :380000
```

Figure 154: The Summary of the Average Rent According to the Total Floor on the Console

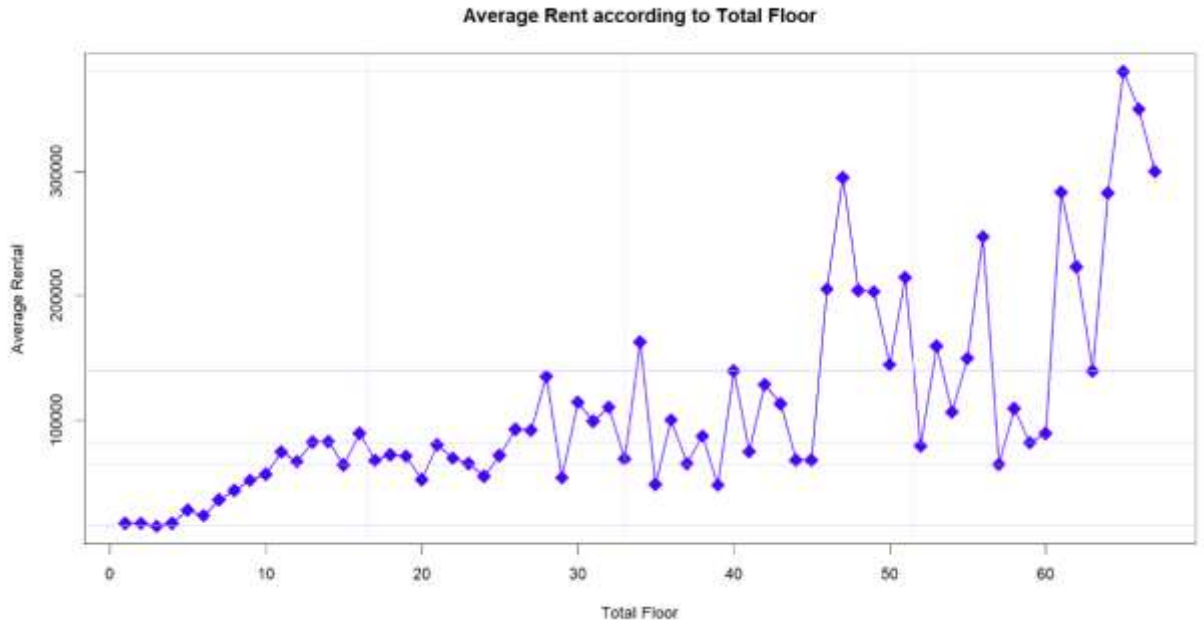

Figure 155: The Average Rent According to the Total Floor in Connected Scatterplot

3.2.7 Analysis 2.7: How does the house size affect the house rent?

3.2.7.1 Analysis 2.7.1: House Size Distribution

```
#Analysis 2.7: How does the house size affect the house rent ?
#Analysis 2.7.1: House Size Distribution
#Bar Plot
ggplot(house_size_range, aes(x = SIZE_RANGE, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(house_size_range$SIZE_RANGE))) +
  ggtitle("House Size Distribution") +
  theme( plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(SIZE_RANGE, label = QUANTITY), vjust = -1, position = position_dodge(width = 0.1))

#Lollipop Chart for Range Between 1000 to 10000
ggplot(house_size_range, aes(x = SIZE_RANGE, y = QUANTITY)) +
  geom_segment( aes(x = SIZE_RANGE, xend = SIZE_RANGE, y = 0, yend = QUANTITY), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(house_size_range$SIZE_RANGE)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = QUANTITY), color = "black", size = 3.5) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("House Size Distribution")
```

Figure 156: Code to Find the House Size Distribution

More than 95% of the houses have the house size of 0 to 2000.

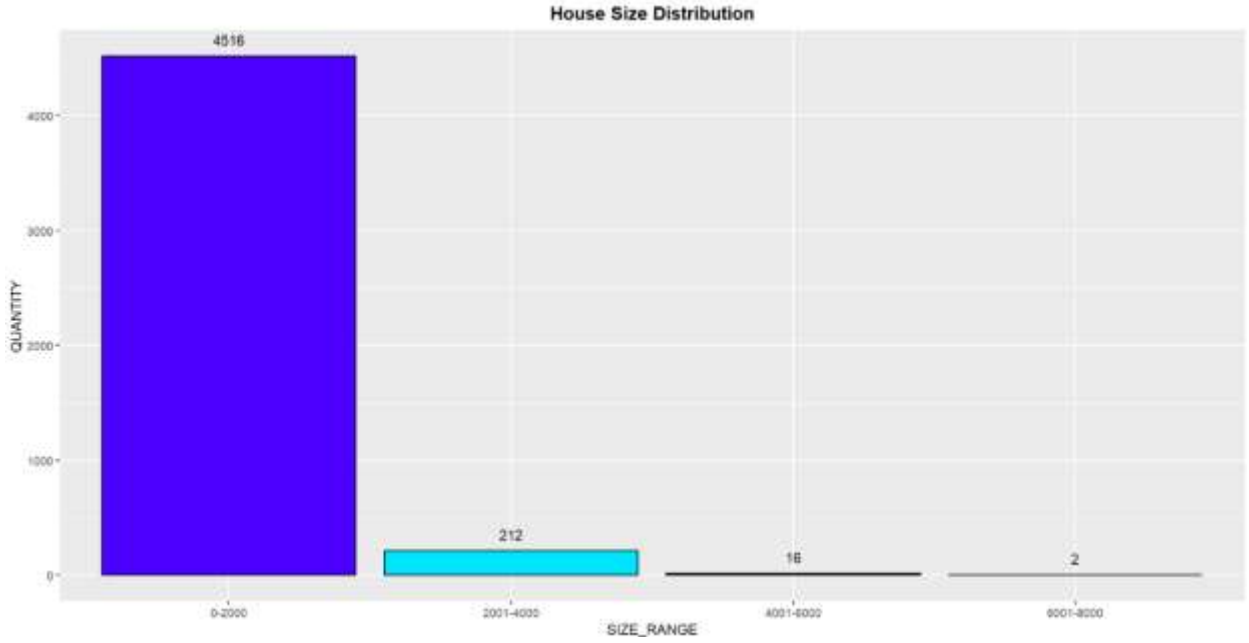

Figure 157: The House Size Distribution in Bar Plot

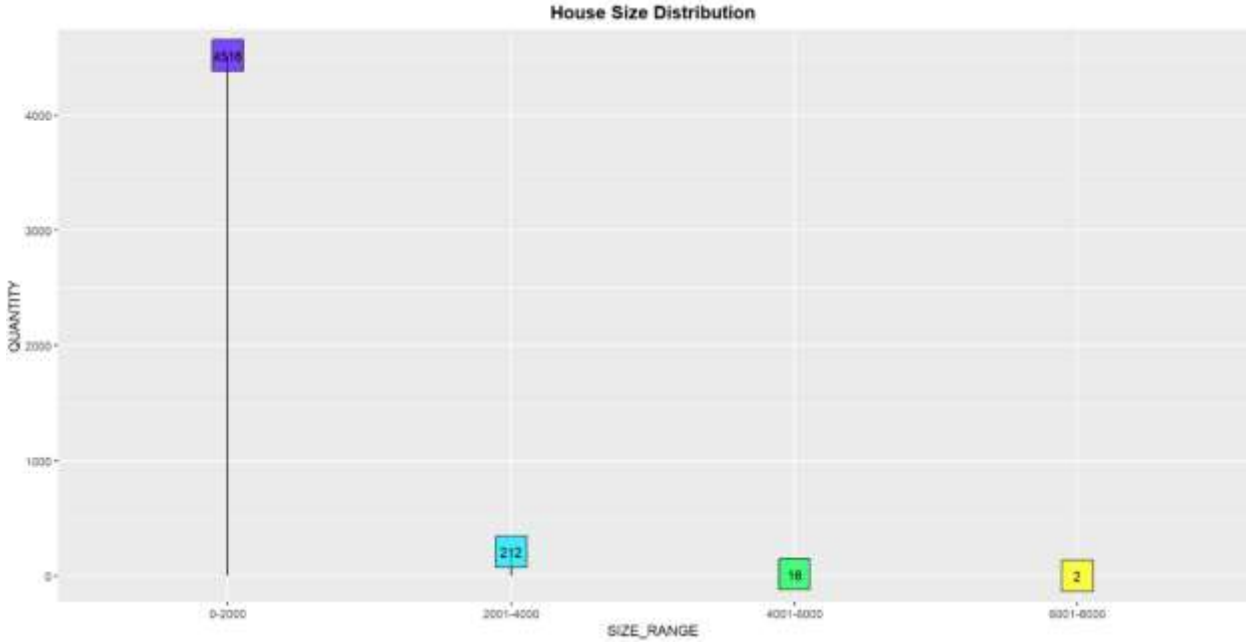

Figure 158: The House Size Distribution in Lollipop Plot

3.2.7.2 Analysis 2.7.2: the Rent Distribution in Each City According to House Size

```
#Analysis 2.7.2: the Rent Distribution in Each City According to House Size
ggplot(rent, aes(x = RENTAL, y= HOUSE_SIZE, shape = CITY, colour = CITY))+
  ggtitle("The Rent Distribution in Each City According to House Size") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_point(size=3, color=topo.colors(length(rent$CITY)))
```

Figure 159: Code to Find the Rent Distribution in Each City According to the House Size

Most of the houses has lower house size and lower rent even though there is a house which cost more than 3500000.

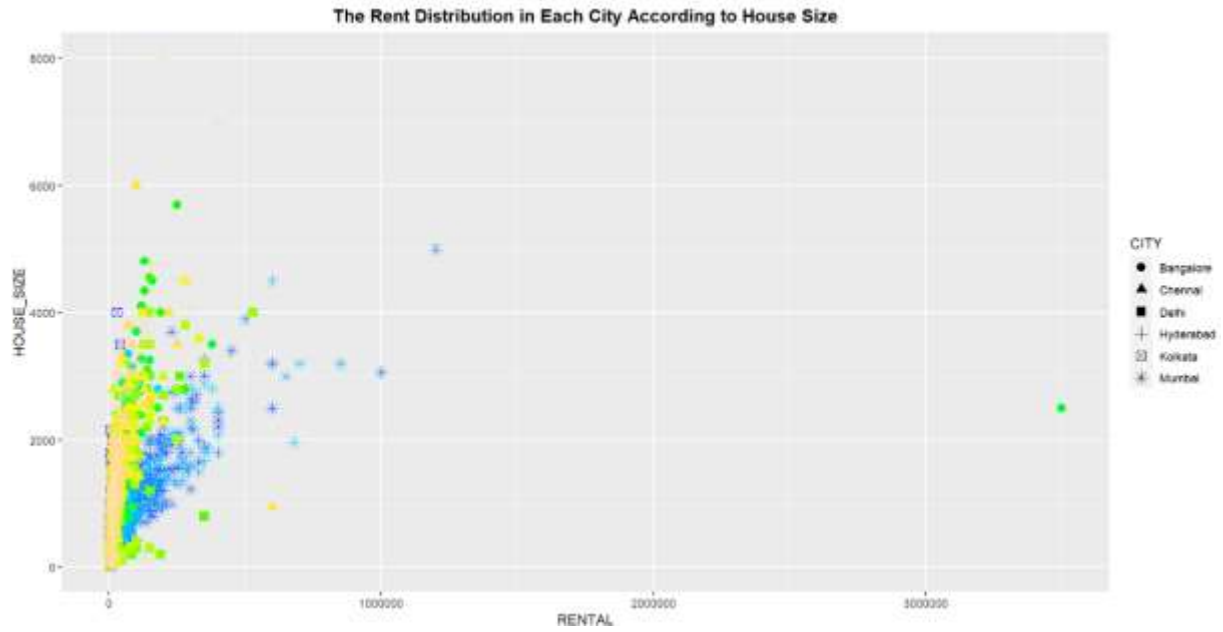

Figure 160: The Rent Distribution in Each City According to the House Size in Scatterplot

3.2.7.3 Analysis 2.7.3: Average Rent according to House Size

```
#Analysis 2.7.3: Average Rent according to House Size
avg_rent_size <- rent %>%
  group_by(HOUSE_SIZE) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_size$AVG <- as.integer(avg_rent_size$AVG)

summary(avg_rent_size)

par(mar = c(5, 5, 4, 4))
plot(avg_rent_size$AVG, type = "p", xlab = "House Size",
     ylab = "Average Rental",
     main = "Average Rent according to House Size",
     col = alpha("#4C00FF",0.2), pch = 16, cex = 3, lwd = 1.5)
abline(v=c(605, 1015, 1566) , col="#F8D8FC")
abline(h=c(4500, 28833, 55972, 1200000) , col="#BED0FB")
```

Figure 161: Code to Find the Average Rent according to the House Size

```
> summary(avg_rent_size)
   HOUSE_SIZE         AVG
 Min.   :  10   Min.   :   4500
 1st Qu.: 605   1st Qu.:  16339
 Median :1015   Median :  28833
 Mean   :1229   Mean   :  56070
 3rd Qu.:1566   3rd Qu.:  55972
 Max.   :8000   Max.   :1200000
```

Figure 162: Output of the Summary of the Average Rent According to the House Size on the Console

Most of the rental lies between 0 – 200000 as the color within the range is deep compared to other range,

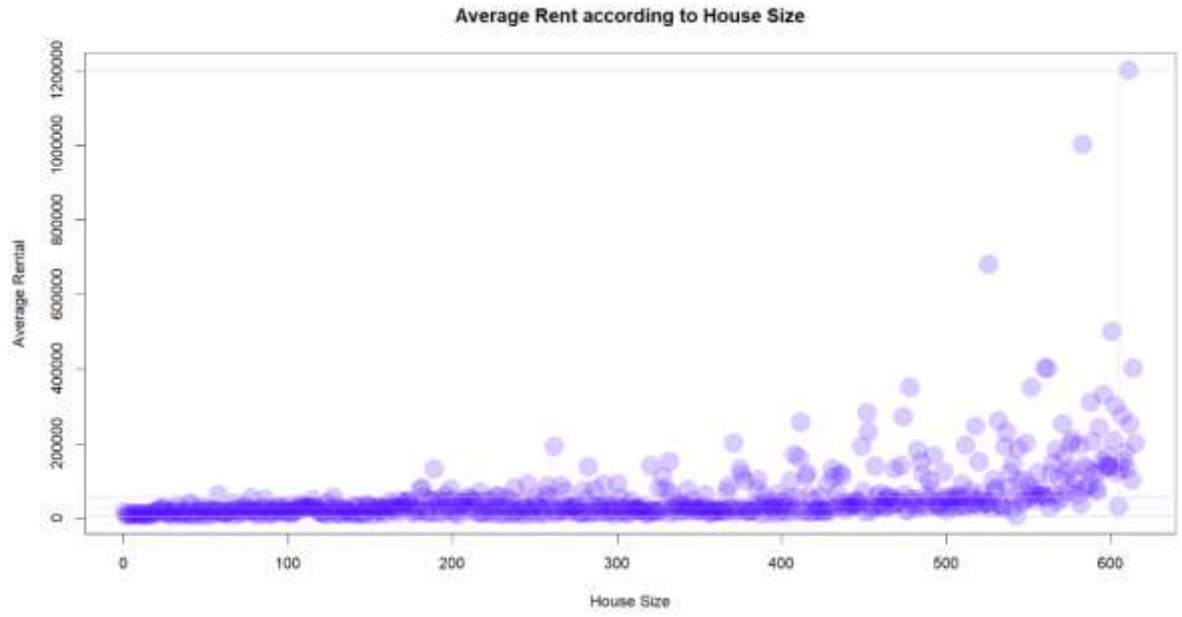

Figure 163: Average Rent According to the House Size in Bubble Plot

3.2.8 Analysis 2.8: How does the area type affect the house rent?

3.2.8.1 Analysis 2.8.1: Area Type Distribution

```
#Analysis 2.8: How does the area type affect the house rent ?
#Analysis 2.8.1: Area Type Distribution
area_type_num <- rent %>%
  group_by(AREA_TYPE) %>%
  summarise(QUANTITY = n())
area_type_num

#Bar Plot
par(mar = c(4, 4, 4, 4))
ggplot(area_type_num, aes(x = AREA_TYPE, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(area_type_num$AREA_TYPE))) +
  ggtitle("The Area Type Distribution") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(AREA_TYPE, label = QUANTITY), vjust = -1, position = position_dodge(width = 0.1))

#Lolipop Chart for Range Between 1000 to 10000
ggplot(area_type_num, aes(x = AREA_TYPE, y = QUANTITY)) +
  geom_segment( aes(x = AREA_TYPE, xend = AREA_TYPE, y = 0, yend = QUANTITY), colour = "Black") +
  geom_point( size = 12, color = "black",
              fill = alpha(topo.colors(length(area_type_num$AREA_TYPE)), 0.3),
              alpha = 0.7, shape = 22, stroke = 1) +
  geom_text(aes(label = QUANTITY), color = "black", size = 3.5) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  ggtitle("The Area Type Distribution")
```

Figure 164: Code to find the Area Type Distribution

```
> area_type_num
# A tibble: 3 × 2
  AREA_TYPE   QUANTITY
  <chr>          <int>
1 Built Area         2
2 Carpet Area     2298
3 Super Area      2446
```

Figure 165: Output of the Area Type Distribution on the Console

Most of the houses are categorized as super area.

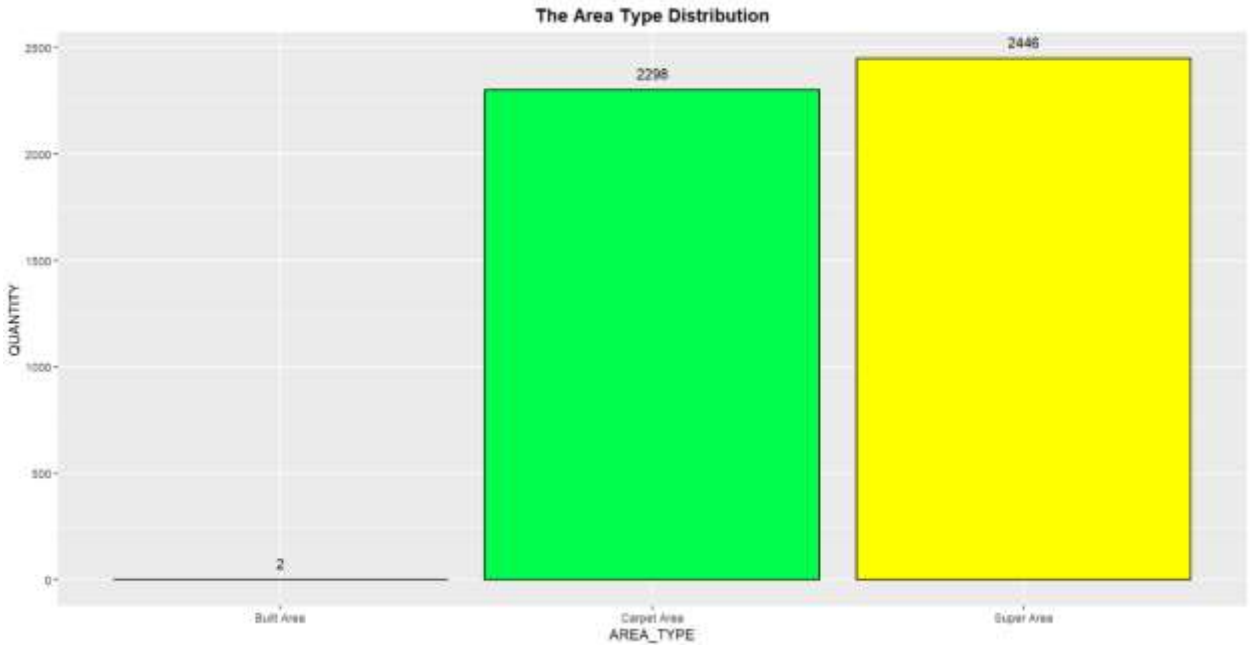

Figure 166: The Area Type Distribution in Bar Plot

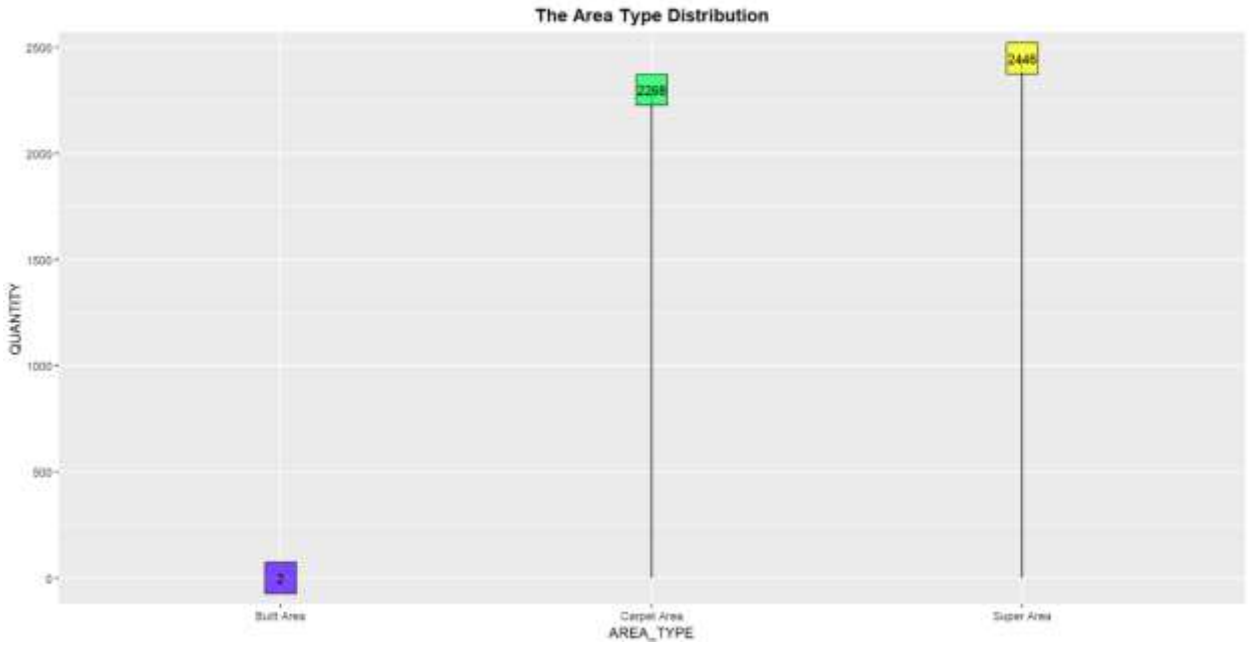

Figure 167: The Area Type Distribution in Lollipop Plot

3.2.8.2 Analysis 2.8.2: Average Rent according to Area Type

```
#Analysis 2.8.2: Average Rent according to Area Type
avg_rent_area <- rent %>%
  group_by(AREA_TYPE) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_area$AVG <- as.integer(avg_rent_area$AVG)

ggplot(avg_rent_area, aes(x = AREA_TYPE, y = AVG)) +
  scale_y_continuous(breaks = seq(from = 0, to = 60000,by = 10000)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(avg_rent_area$AREA_TYPE))) +
  ggtitle("Average Rent according to Area Type") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(AREA_TYPE, label = AVG),  cex = 5, vjust = 1.5)
```

Figure 168: Code to Find the Average Rent According to the Area Type

Carpet area houses cost the highest among all. Thus, carpet is really expensive if the user decided to buy a carpet area house.

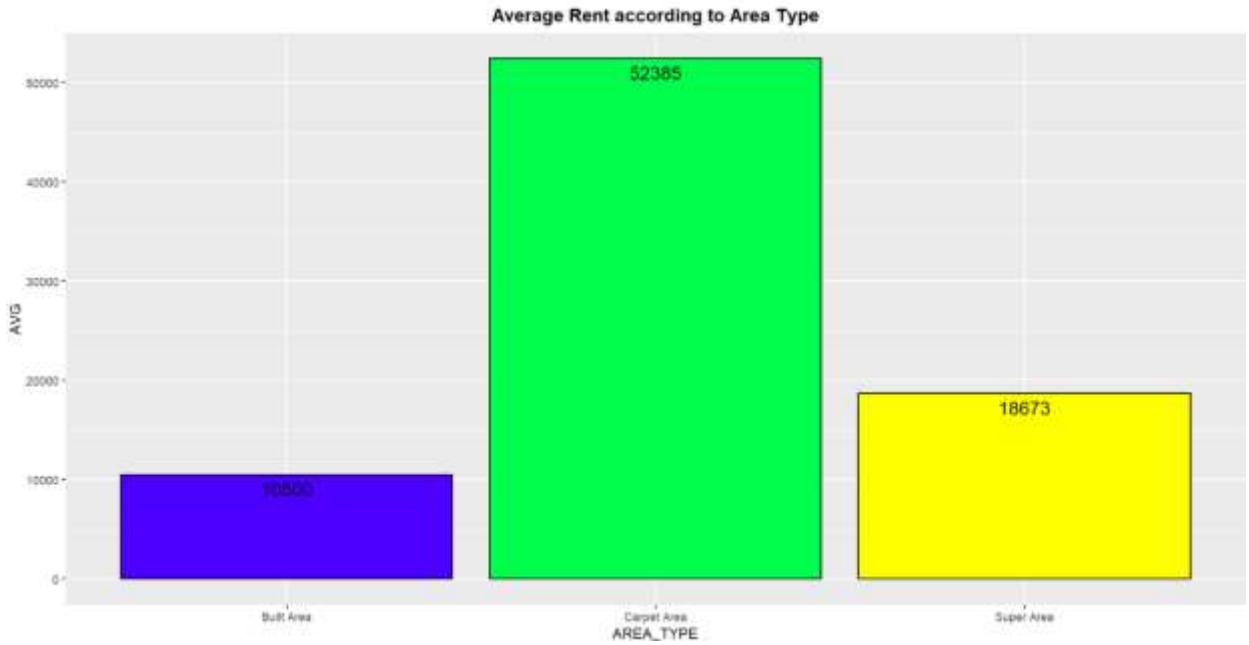

Figure 169: Average Rent according to the Area Type in Bar Plot

3.2.9 Conclusion for Question 2

The house condition affects the rental if the tenant requires better surroundings. However, a bigger group of tenant could enjoy reasonable and affordable price if many spaces is required.

3.3 Question 3: How does the relationship between the tenant and owner affect the house rent?

3.3.1 Analysis 3.1: How does the tenant preferred affect the house rent?

3.3.1.1 Analysis 3.1.1: The Quantity of Each Tenant Preferred

```
#Question 3: How does the relationship between the tenant and owner affect the house rent ?
#Analysis 3.1: How does the tenant preferred affect the house rent ?
#Analysis 3.1.1: The Quantity of Each Tenant Preferred
tenant_num <- rent %>%
  group_by(TENANT_PREFERRED) %>%
  summarise(QUANTITY = n())
tenant_num

#Pie Chart
par(mar = c(2, 2, 2, 2))
pie(tenant_num$QUANTITY, paste(tenant_num$TENANT_PREFERRED,tenant_num$QUANTITY),
    radius = 0.7, main = "The Quantity of Each Type Of Point of Contact",
    col = topo.colors(length(tenant_num$TENANT_PREFERRED)), clockwise =  TRUE)
legend("topright", tenant_num$TENANT_PREFERRED,
       fill = topo.colors(length(tenant_num$TENANT_PREFERRED)),
       pt.cex = 2, cex = 0.7, horiz = FALSE, inset = c( -0.3 , 0.35))

#Bar Plot
par(mar = c(4, 4, 4, 4))
ggplot(tenant_num, aes(x = TENANT_PREFERRED, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(tenant_num$TENANT_PREFERRED))) +
  ggtitle("The Quantity of Each Type Of Point of Contact") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(TENANT_PREFERRED, label = QUANTITY), vjust = -1,
            position = position_dodge(width = 0.1))
```

Figure 170: Code to find the Quantity of Each Tenant Preferred

Bachelors / Family is the most famous tenant preferred.

```
> tenant_num
# A tibble: 3 x 2
  TENANT_PREFERRED QUANTITY
  <chr>                <int>
1 Bachelors              830
2 Bachelors/Family      3444
3 Family                 472
```

Figure 171: Output of the Quantity of Each Tenant Preferred on the Console

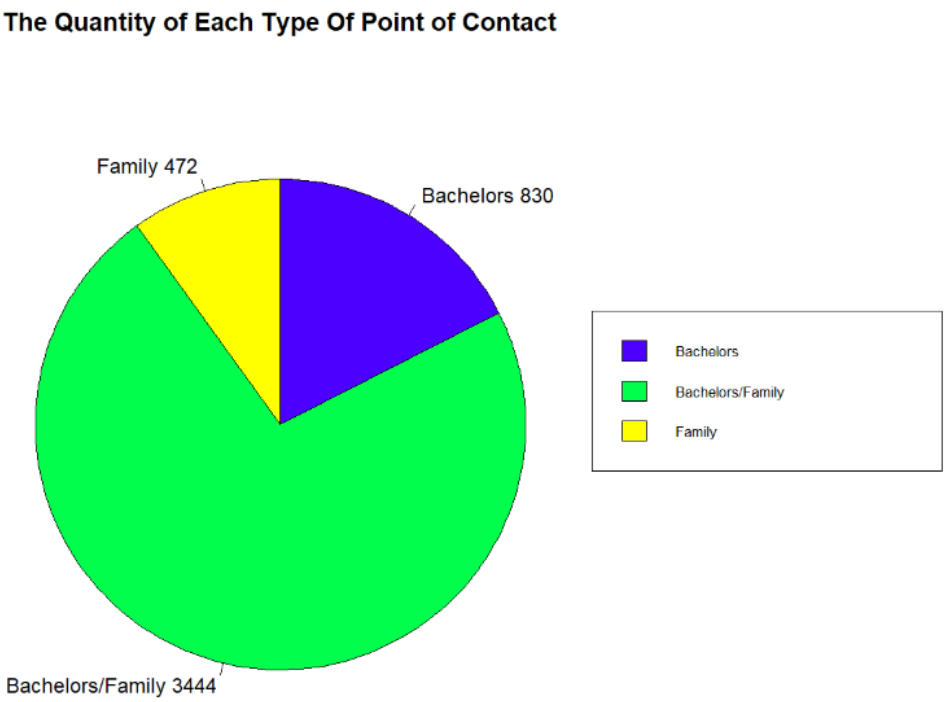

Figure 172: The Quantity of Each Tenant Preferred in Pie Chart

3.3.1.2 Analysis 3.1.2: Average Rent according to Tenant Preferred

```
#Analysis 3.1.2: Average Rent according to Tenant Preferred
avg_rent_tp <- rent %>%
  group_by(TENANT_PREFERRED) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_tp$AVG <- as.integer(avg_rent_tp$AVG)

ggplot(avg_rent_tp, aes(x = TENANT_PREFERRED, y = AVG)) +
  scale_y_continuous(breaks = seq(from = 0, to = 60000,by = 10000)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(avg_rent_tp$TENANT_PREFERRED))) +
  ggtitle(" Average Rent according to Tenant Preferred") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(TENANT_PREFERRED, label = AVG),  cex = 5, vjust = 1.5)
```

Figure 173: Code to find the Average Rent according to the Tenant Preferred

If the tenant preferred is family, the cost of rent will be higher.

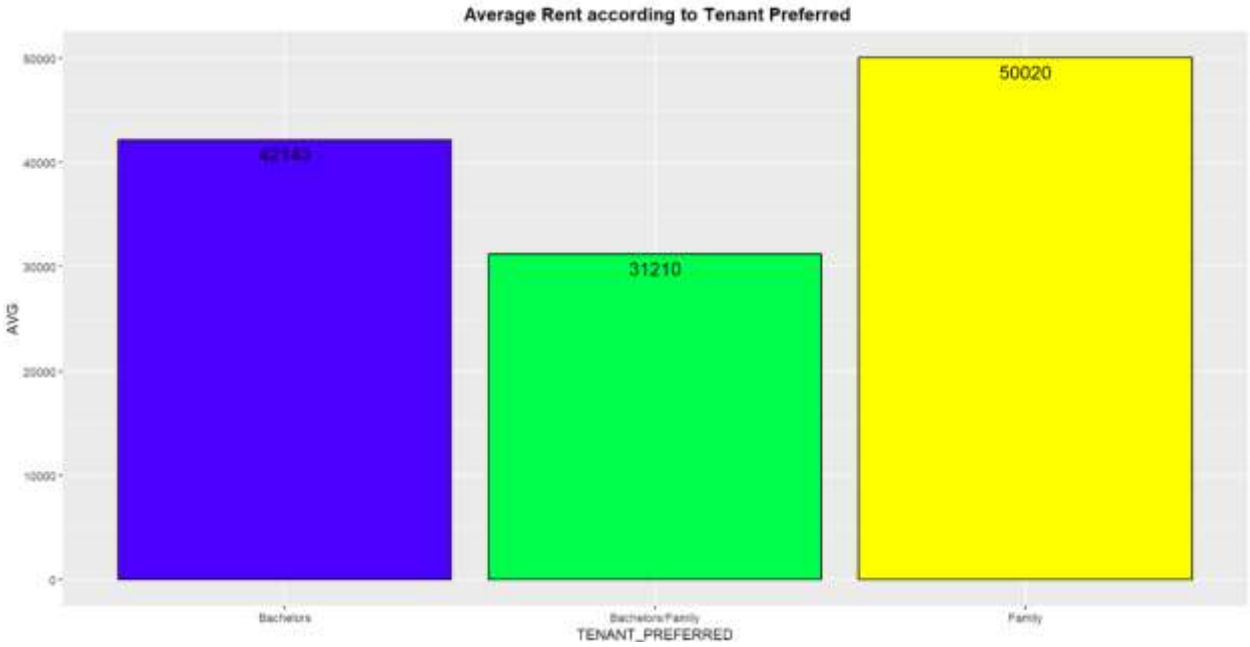

Figure 174: The Average Rent according to the Tenant Preferred in Bar Plot

3.3.2 Analysis 3.2: How does the Point of Contact affect the house rent?

3.3.2.1 Analysis 3.2.1: The Quantity of Each Type of Point of Contact

```
#Analysis 3.2: How does the Point of Contact affect the house rent ?
#Analysis 3.2.1: The Quantity of Each Type Of Point of Contact
contact_num <- rent %>%
  group_by(POINT_OF_CONTACT) %>%
  summarise(QUANTITY = n())
contact_num['POINT_OF_CONTACT'] = NULL
contact_num  <- cbind(POINT_OF_CONTACT, contact_num)
contact_num

#Pie Chart
par(mar = c(2, 2, 2, 2))
pie(contact_num$QUANTITY, paste(contact_num$POINT_OF_CONTACT,contact_num$QUANTITY),
    radius = 0.7, main = "The Quantity of Each Type Of Point of Contact",
    col = topo.colors(length(contact_num$POINT_OF_CONTACT)), clockwise =  TRUE)
legend("topright", contact_num$POINT_OF_CONTACT,
       fill = topo.colors(length(contact_num$POINT_OF_CONTACT)),
       pt.cex = 2, cex = 0.7, horiz = FALSE, inset = c( -0.3 , 0.35))

#Bar Plot
par(mar = c(4, 4, 4, 4))
ggplot(contact_num, aes(x = POINT_OF_CONTACT, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(contact_num$POINT_OF_CONTACT))) +
  ggtitle("The Quantity of Each Type Of Point of Contact") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(POINT_OF_CONTACT, label = QUANTITY), vjust = -1,
            position = position_dodge(width = 0.1))
```

Figure 175: Code to Find the Quantity of Each Type of Point of Contact

Most of the tenant contact owner compared to agent and builder.

```
> contact_num
  POINT_OF_CONTACT QUANTITY
1    Contact Owner     3216
2    Contact Agent     1529
3  Contact Builder        1
```

Figure 176: Output of the Quantity of Each Type of Point of Contact on the Console

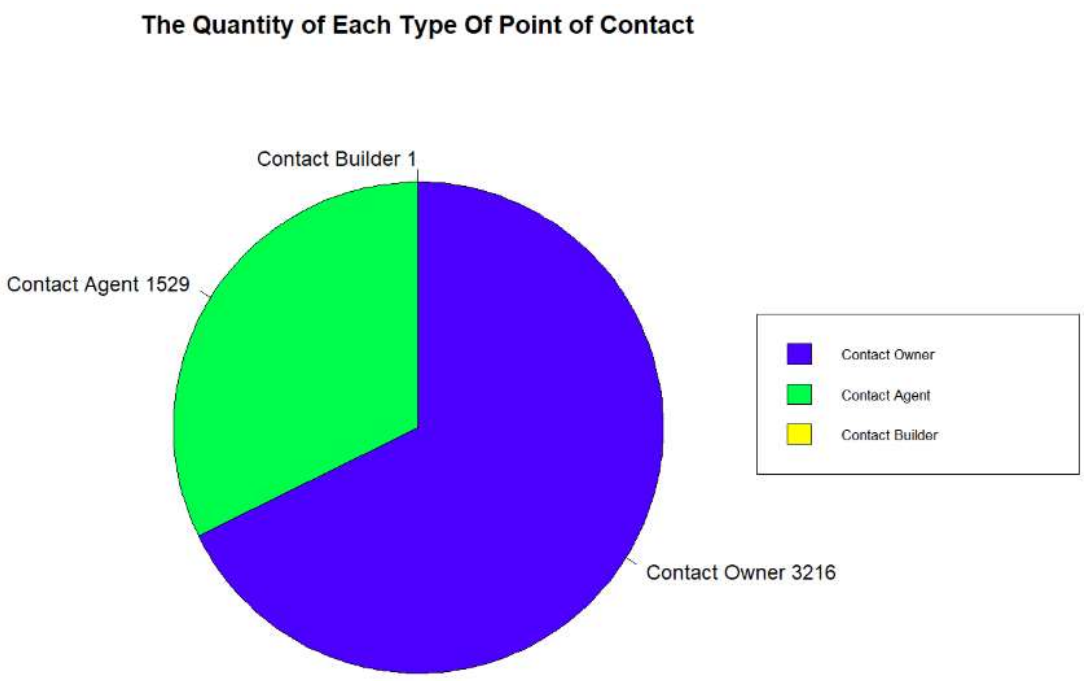

Figure 177: The Quantity of Each Type of Point of Contact in Pie Chart

3.3.2.2 Analysis 3.2.2: Average Rent according to Point of Contact

```
#Analysis 3.2.2: Average Rent according to Point of Contact
avg_rent_poc <- rent %>%
  group_by(POINT_OF_CONTACT) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_poc$AVG <- as.integer(avg_rent_poc$AVG)

ggplot(avg_rent_poc, aes(x = POINT_OF_CONTACT, y = AVG)) +
  scale_y_continuous(breaks = seq(from = 0, to = 80000,by = 10000)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black",
           fill = topo.colors(length(avg_rent_poc$POINT_OF_CONTACT))) +
  ggtitle("Average Rent according to Point of Contact") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(POINT_OF_CONTACT, label = AVG),  cex = 5, vjust = 1.5)
```

Figure 178: Code to find the Average Rent according to the Point of Contact

If the tenant contact through agent, the average rent will be higher as there might be invisible fees include in the rent.

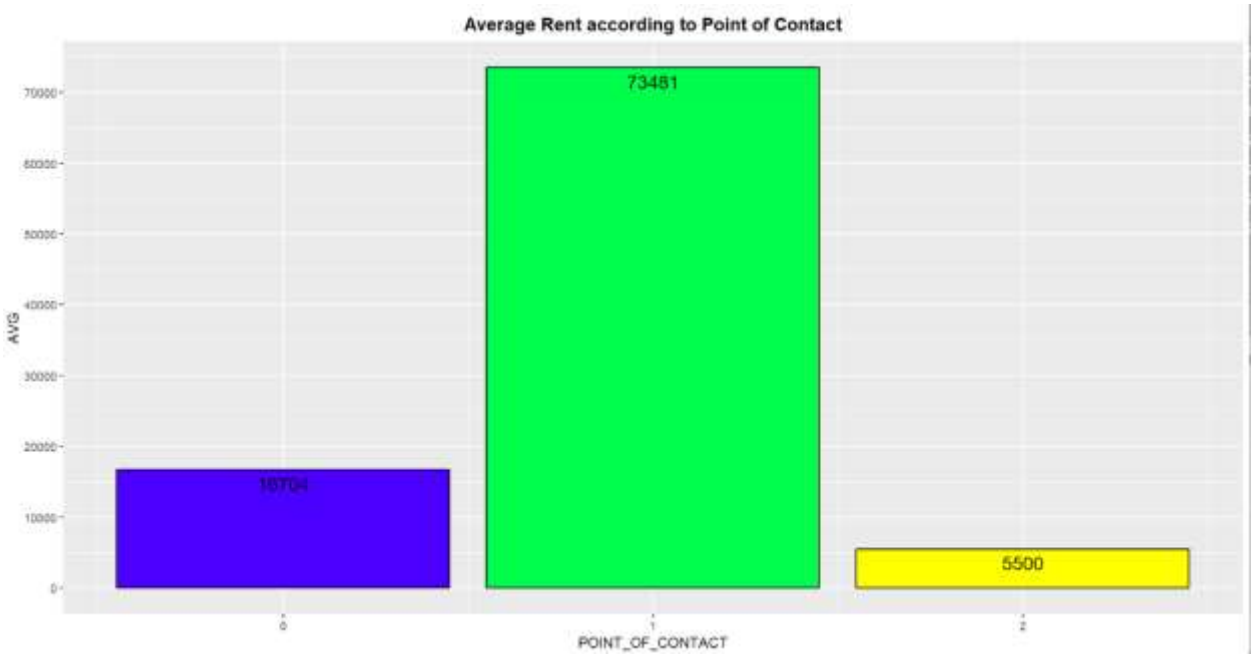

Figure 179: The Average Rent according to Point of Contact ion Bar Plot

3.3.3 Conclusion of Question 3

It is easier to get a cheaper price if the tenant is connected to the owner. It is suggested to contact through the owner as agent may have extra charges.

3.4 Question 4: What is the best day to rent a cheaper house?

3.4.1 Analysis 4.1: How does the day of house rent posted affect the house rent?

3.4.1.1 Analysis 4.1.1: The Quantity House Rent Post on Each Day

```
#Question 4: What is the best day to rent a cheaper house ?
#Analysis 4.1: How does the day of house rent posted affect the house rent ?
#Analysis 4.1.1: The Quantity House Rent Post on Each Day
date_num <- rent %>%
  group_by(DAY) %>%
  summarise(QUANTITY = n())
date_num

summary(date_num)

ggplot(date_num, aes(x = DAY, y = QUANTITY)) +
  geom_line( color= "#69b3a2", linewidth = 2, alpha = 0.9) +
  scale_x_continuous(breaks = seq(from = 1, to = 31,  by = 1)) +
  scale_y_continuous(breaks = seq(from = 0, to = 500,  by = 50)) +
  ggtitle("The Quantity House Rent Post on Each Day") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5))
```

Figure 180: Code to Show the Quantity of the House rent Post on Each Day

```
> summary(date_num)
      DAY          QUANTITY
 Min.   : 1.0   Min.   : 55.0
 1st Qu.: 8.5   1st Qu.: 85.5
 Median :16.0   Median :147.0
 Mean   :16.0   Mean   :153.1
 3rd Qu.:23.5   3rd Qu.:192.0
 Max.   :31.0   Max.   :456.0
```

Figure 181: The Summary of Date Count on the Console

The sixth of every month has the most post compared to the other day.

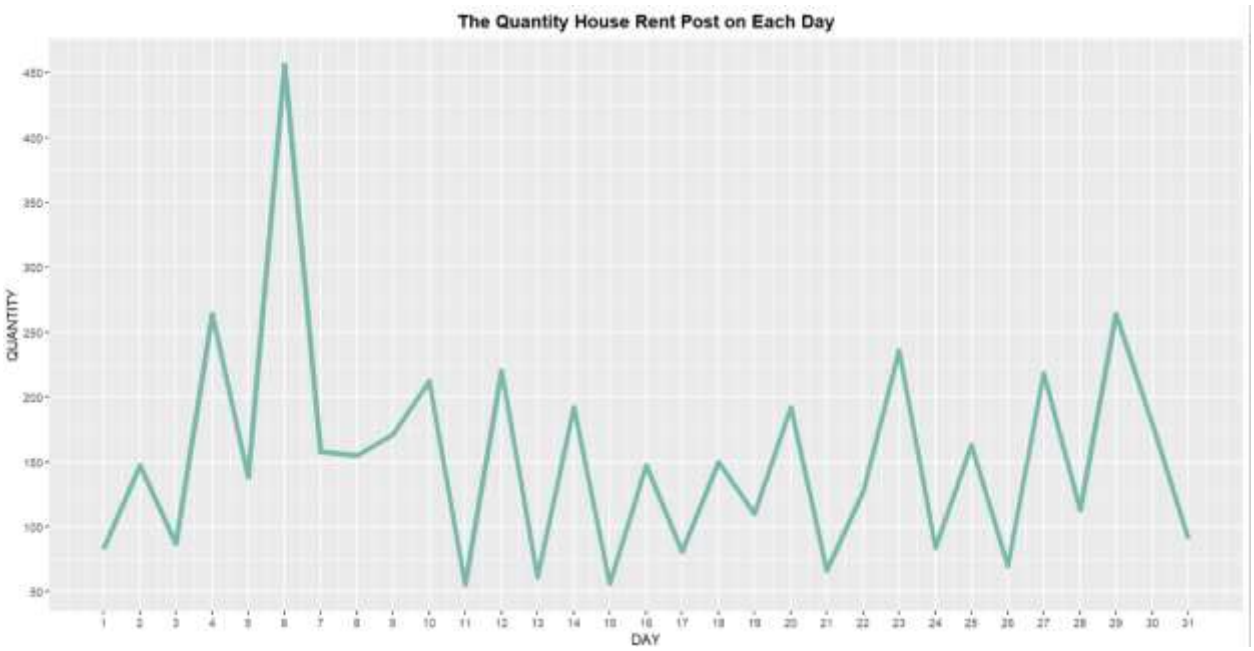

Figure 182: The Quantity of House Rent Post on Each Day in Line Graph

3.4.1.2 Analysis 4.1.2: Average Rent according to Day

```
#Analysis 4.1.2: Average Rent according to Day
avg_rent_day <- rent %>%
    group_by(DAY) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_day$AVG <- as.integer(avg_rent_day$AVG)

ggplot(avg_rent_day, aes(x = DAY, y = AVG)) +
  geom_line( color= "#D27AE7", linewidth = 1.5, alpha = 0.9) +
  scale_x_continuous(breaks = seq(from = 1, to = 31,  by = 1)) +
  scale_y_continuous(breaks = seq(from = 15000, to = 80000,  by = 4000)) +
  ggtitle("The Average Rent according to Day") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_hline(yintercept = mean(avg_rent_day$AVG, na.rm=TRUE), color = "#B4DCDC", lwd = 1, lty = 2)
```

Figure 183: Code to Find the Average Rent according to Day

The rent on 9 of each month is pricey, while the rent on 5 of each month is cheap.

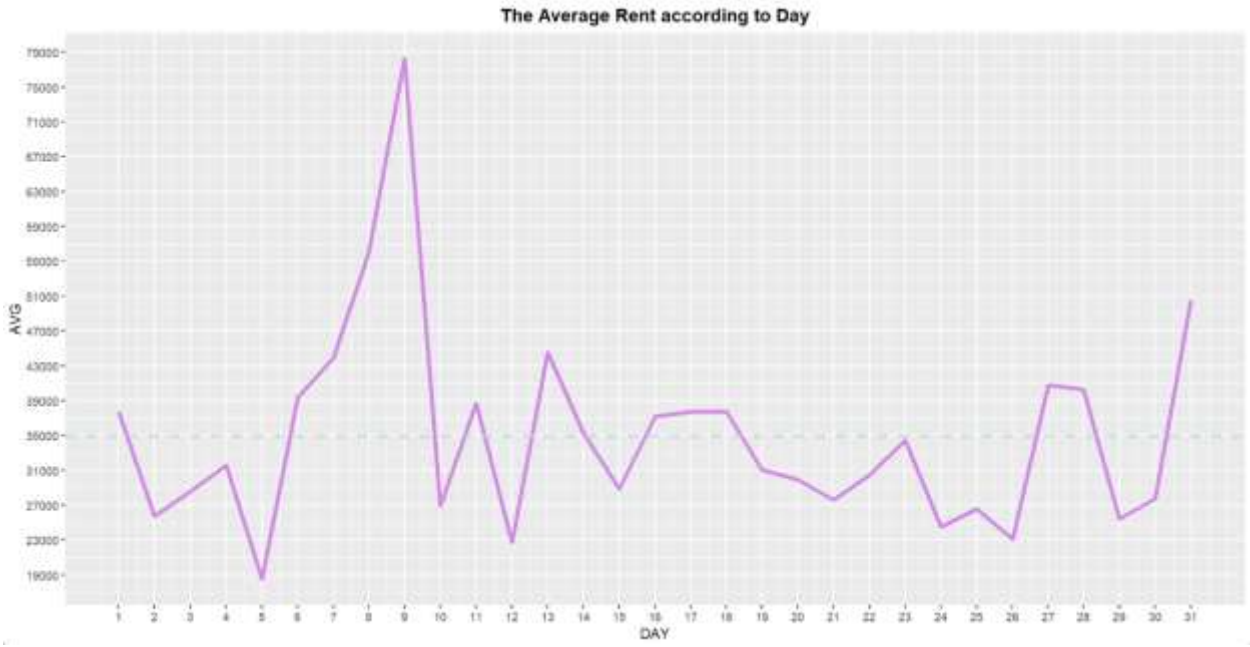

Figure 184: The Average Rent according to Day in Line Graph

3.4.2 Analysis 4.2: How does the month of house rent posted affect the house rent?

3.4.2.1 Analysis 4.2.1: The Quantity House Rent Post on Each Month

```
#Analysis 4.2: How does the month of house rent posted affect the house rent ?
#Analysis 4.2.1: The Quantity House Rent Post on Each Month
month_num <- rent %>%
  group_by(MONTH) %>%
  summarise(QUANTITY = n())
month_num

summary(month_num)

ggplot(month_num, aes(x = MONTH, y = QUANTITY)) +
  geom_line( color= "#69b3a2", linewidth = 2, alpha = 0.9) +
  scale_x_continuous(breaks = seq(from = 0, to = 12,  by = 1)) +
  scale_y_continuous(breaks = seq(from = 0, to = 1900,  by = 100)) +
  ggtitle("The Quantity House Rent Post on Each Month") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5))
```

Figure 185: The Code of the Quantity of House Rent Post on Each Month

```
> summary(month_num)
     MONTH          QUANTITY
 Min.   :4.00   Min.   : 228.0
 1st Qu.:4.75   1st Qu.: 790.5
 Median :5.50   Median :1329.5
 Mean   :5.50   Mean   :1186.5
 3rd Qu.:6.25   3rd Qu.:1725.5
 Max.   :7.00   Max.   :1859.0
```

Figure 186: The Summary of the Month Count on the Console

June has the most post on each month.

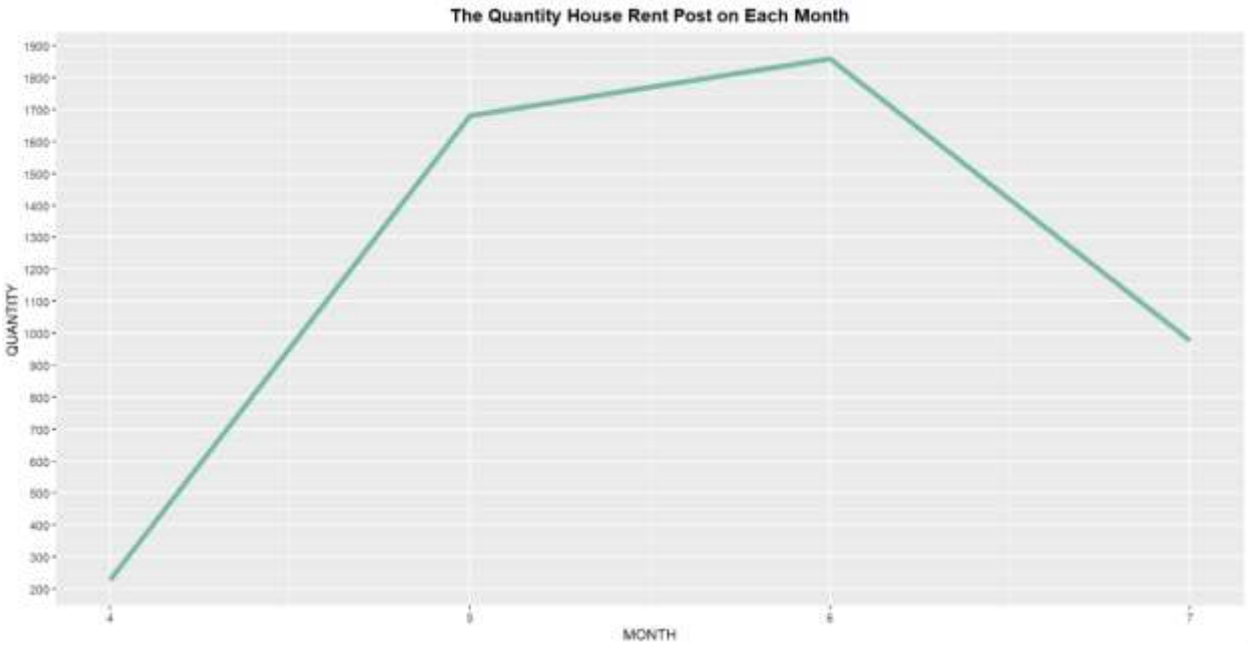

Figure 187: The Quantity of the House Rent Post on Each Month in Line Graph

3.4.2.2 Analysis 4.2.2: Average Rent according to Month

```
#Analysis 4.2.2: Average Rent according to Month
avg_rent_month <- rent %>%
  group_by(MONTH) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_month$AVG <- as.integer(avg_rent_month$AVG)

ggplot(avg_rent_month, aes(x = MONTH, y = AVG)) +
  geom_line( color= "#D27AE7", linewidth = 1.5, alpha = 0.9) +
  scale_x_continuous(breaks = seq(from = 1, to = 31,  by = 1)) +
  scale_y_continuous(breaks = seq(from = 15000, to = 80000,  by = 4000)) +
  ggtitle("The Average Rent according to Month") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_hline(yintercept = mean(avg_rent_month$AVG, na.rm=TRUE),
             color = "#B4DCDC", lwd = 1, lty = 2) +
  geom_label(data = avg_rent_month, aes(x = MONTH, y = AVG, label = AVG),
             color= "#62AAC5",
             size = 3.5, angle = 45, fontface = "bold")
```

Figure 188: Code to Show the Average Rent According to Month

It is more suitable for the people to rent a house in April. The price of rent will start raising after on.

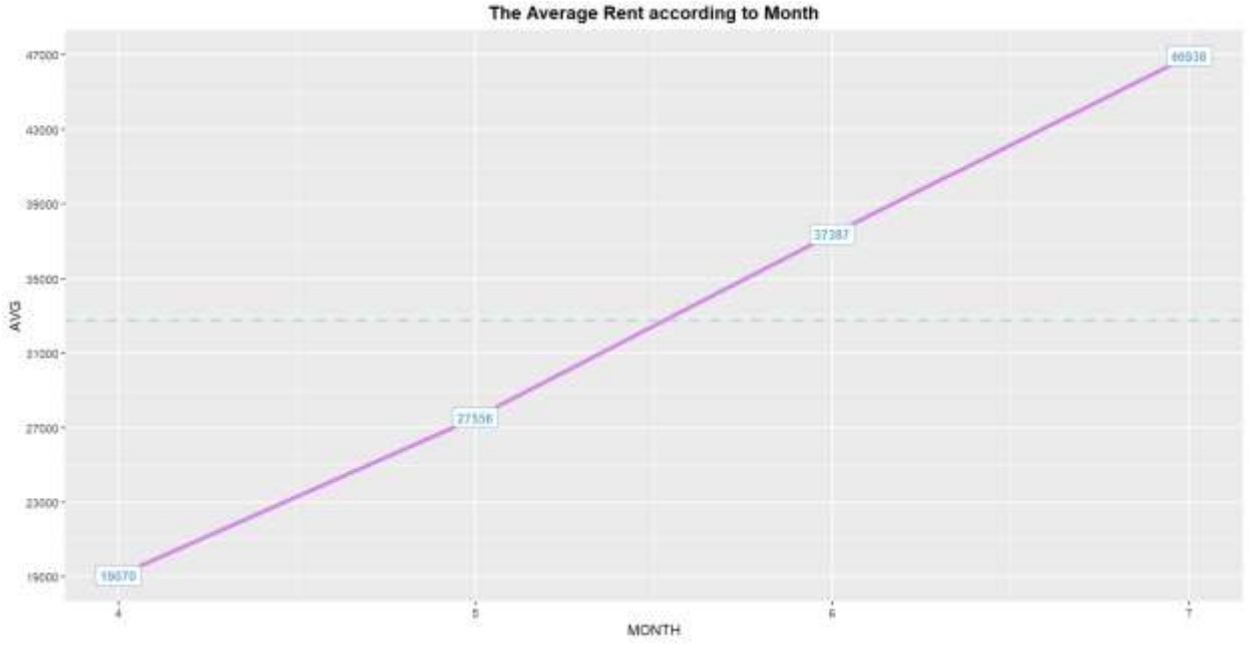

Figure 189: the Average Rent according to Month in Line Graph

3.4.3 Analysis 4.3: How does the date of house rent posted affect the house rent?

3.4.3.1 Analysis 4.3.1: Which Date has the most house rent posted?

```
#Analysis 4.3: How does the date of house rent posted affect the house rent ?
#Analysis 4.3.1: Which Date has the most house rent posted ?
date_num <- rent %>%
  group_by(DATE) %>%
  summarise(QUANTITY = n())
date_num

summary(date_num)

par(mar = c(4, 4, 4, 4))
ggplot(date_num, aes(x = DATE, y = QUANTITY)) +
  geom_line( color= "#69b3a2", linewidth = 0.5, alpha = 0.9) +
  scale_y_continuous(breaks = seq(from = 0, to = 350,  by = 50)) +
  scale_x_date(date_breaks = "1 day", date_labels = "%d %b") +
  ggtitle("The Quantity House Rent Post on Each Date") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  theme(axis.text.x=element_text(size = 7, angle = 60, hjust = 1)) +
  geom_hline(yintercept = mean(date_num$QUANTITY, na.rm=TRUE),
             color = "#B4DCDC", lwd = 1, lty = 2) +
  geom_point( size = 2, color = "black", fill = alpha(topo.colors(length(date_num$DATE)), 0.3),
              alpha = 0.7, shape = 21, stroke = 1)
```

Figure 190: The Code to show the Date with the most House Rent Posted

2022-07-06 is the best day to rent a house as there are many selection for the tenant to select.

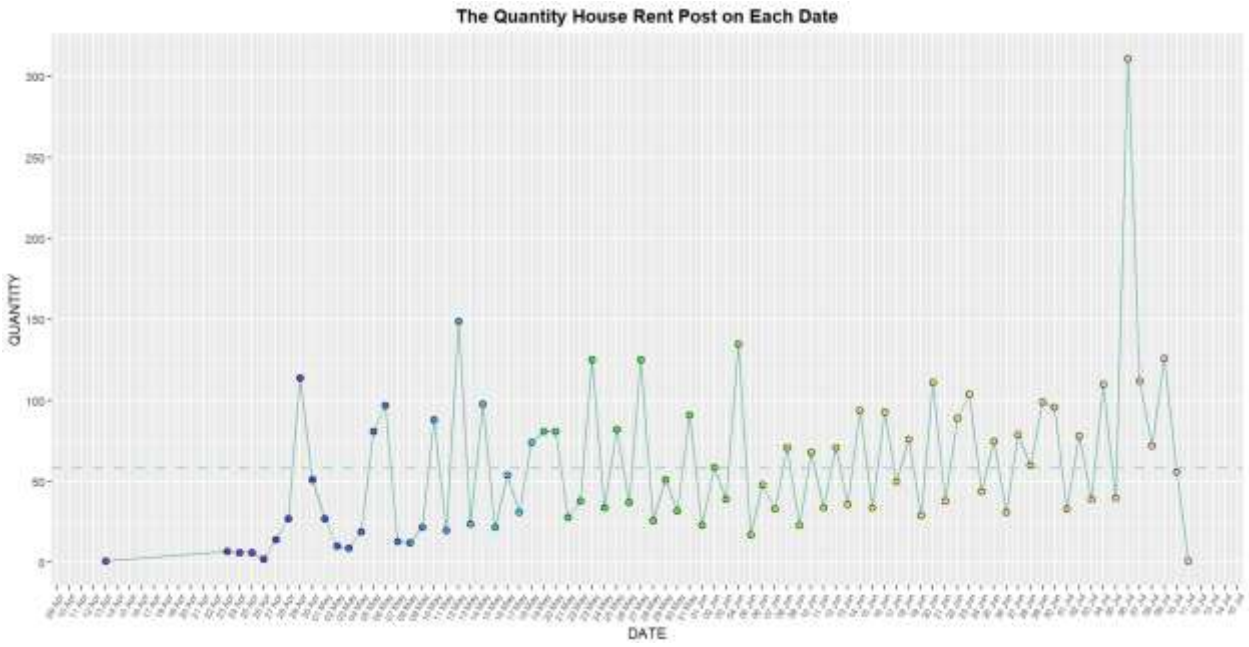

Figure 191: The Quantity of the House Rent Post on Each Date

3.4.3.2 Analysis 4.3.2: The Date with the Lowest Average Rent

```
#Analysis 4.3.2: The Date with the Lowest Average Rent
avg_rent_date <- rent %>%
  group_by(DATE) %>%
  summarise(AVG = format(round(mean(RENTAL),1), nsmall = 1))
avg_rent_date <- arrange(avg_rent_date, AVG)
View(avg_rent_date)
```

Figure 192: Code to Find the Date with the Lowest Average Rent

2022-04-24 has the lowest rent among all and 2022-07-29 has the highest rent among all. The tenant is suggested to rent a house on 2022-04-24.

	DATE	AVG		DATE	AVG		DATE	AVG		DATE	AVG
1	2022-04-24	10000.0	21	2022-05-10	17919.3	41	2022-06-29	27559.6	61	2022-05-28	38673.1
2	2022-05-03	10444.4	22	2022-05-20	18316.7	42	2022-06-02	27854.2	62	2022-05-14	38890.6
3	2022-04-25	11166.7	23	2022-06-26	18983.9	43	2022-06-21	27986.8	63	2022-06-30	39942.7
4	2022-06-19	11656.9	24	2022-05-04	19289.5	44	2022-06-06	28279.2	64	2022-07-10	40642.8
5	2022-07-11	12000.0	25	2022-04-26	20000.0	45	2022-06-09	28391.3	65	2022-05-27	41148.0
6	2022-04-30	12533.3	26	2022-05-25	20704.9	46	2022-05-29	29245.1	66	2022-06-12	41166.2
7	2022-05-08	12958.3	27	2022-04-29	21804.4	47	2022-06-04	29852.4	67	2022-06-11	43691.2
8	2022-05-07	13561.5	28	2022-05-16	22061.1	48	2022-07-01	29921.2	68	2022-06-07	44363.6
9	2022-05-12	13977.9	29	2022-07-05	23484.6	49	2022-06-22	29996.9	69	2022-06-27	44461.6
10	2022-05-01	14370.4	30	2022-05-23	24057.6	50	2022-07-08	30601.4	70	2022-06-16	46000.0
11	2022-04-23	15375.0	31	2022-07-03	24328.2	51	2022-06-14	31469.1	71	2022-06-13	46333.3
12	2022-05-30	15434.4	32	2022-05-24	24450.0	52	2022-05-22	31492.1	72	2022-07-07	47342.0
13	2022-05-09	15477.3	33	2022-06-17	25334.0	53	2022-05-11	31520.0	73	2022-07-06	47996.6
14	2022-05-02	15800.0	34	2022-07-02	25387.2	54	2022-05-13	32812.5	74	2022-06-23	48114.4
15	2022-05-15	15886.4	35	2022-04-13	260000.0	55	2022-06-25	34217.3	75	2022-05-18	49445.9
16	2022-05-05	16329.6	36	2022-06-18	26500.7	56	2022-07-04	35848.2	76	2022-05-31	50551.7
17	2022-04-28	16492.6	37	2022-06-24	26534.1	57	2022-06-03	37089.7	77	2022-06-28	51668.3
18	2022-05-06	17152.2	38	2022-05-26	26713.5	58	2022-06-15	37250.0	78	2022-05-17	57762.8
19	2022-06-05	17170.6	39	2022-05-21	27142.9	59	2022-05-19	37997.5	79	2022-06-01	76239.1
20	2022-04-27	17375.0	40	2022-06-10	27316.5	60	2022-06-20	38335.1	80	2022-06-08	88988.7
81	2022-07-09	98301.6									

Figure 193: Output of the Date with the Average Rent in Table Format

3.4.4 Conclusion of Question 4

2022-04-24 is the best day to rent a house, but it is suggested to rent a house earlier if the user wants to rent in a reasonable and affordable price.

4.0 Extra Features

The additional features are listed above, which are:

1. sapply()

```
#check data type
datatype = data.frame(DATA_TYPE = sapply(rent, class))
datatype
View(datatype)
```

Figure 194: sapply()

2. strsplit()

```
#split the floor available and total floor
floor = data.frame(do.call("rbind", strsplit(as.character(rent$Floor), " out of ", fixed = TRUE)))
names(floor)=c("Floor_Available", "Total_Floor")
floor
```

Figure 195: strsplit()

3. legend()

```
legend("topright", house_city$CITY, fill = topo.colors(length(house_city$CITY)),
       pt.cex = 2, cex = 0.7, horiz = FALSE, inset = c( -0.3 , 0.35)) #Additional Features
```

Figure 196: legend()

4. summarise()

```
date_num <- rent %>%
  group_by(DATE) %>%
  summarise(QUANTITY = n())
```

Figure 197: summarise()

5. geom_hline()

```
ggplot(avg_rent_day, aes(x = DAY, y = AVG)) +
  geom_line( color= "#D27AE7", linewidth = 1.5, alpha = 0.9) +
  scale_x_continuous(breaks = seq(from = 1, to = 31,  by = 1)) +
  scale_y_continuous(breaks = seq(from = 15000, to = 80000,  by = 4000)) +
  ggtitle("The Average Rent according to Day") +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_hline(yintercept = mean(avg_rent_day$AVG, na.rm=TRUE), color = "#B4DCDC", lwd = 1, lty = 2)
```

Figure 198: geom_hline()

6. scale_y_continuos() & scale_x_continuos()

```
ggplot(bhk_num, aes(x = BHK, y = QUANTITY)) +
  geom_bar(stat = "identity", width = 0.9, color = "Black", fill = topo.colors(length(bhk_num$BHK))) +
  ggtitle("The Quantity of Bedrooms, Halls and Kitchen") +
  scale_x_continuous(breaks = seq(from = 0, to = 6, by = 1)) +
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5)) +
  geom_text(aes(BHK, label = QUANTITY), position = position_dodge(width = 0.1))
```

Figure 199: scale_x_continuos()

7. options()

```
#Force full display
options(scipen=999)
options(show.signif.stars=FALSE)
```

Figure 200:options()

8. geom_segment()

```
ggplot(rental_range_1000_10000, aes(x = RENTAL_RANGE, y = QUANTITY)) +
  geom_segment( aes(x = RENTAL_RANGE, xend = RENTAL_RANGE, y = 0, yend = QUANTITY),
                colour = "Black") +
```

Figure 201: geom_statement()

9. abline()

```
abline(v=c(3.7 , 7.3 ,10.9 ) , col="grey")
```

Figure 202: abline()

10. Lollipop Plot

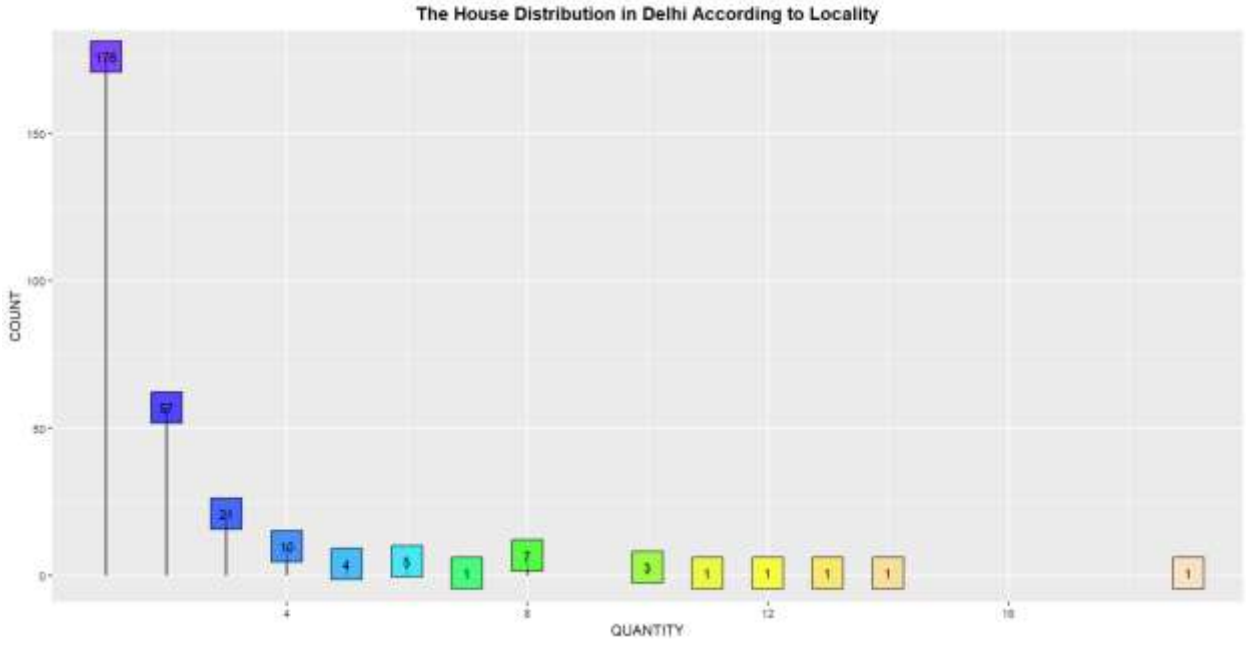

Figure 203: Lollipop Plot

11. unique()

```
#find element in floor
unique(floor$Floor_Available)
```

Figure 204: unique()

12. par()

```
par(mar = c(2, 2, 2, 2))
```

Figure 205: par()

13. is.null() & anyNA()

```
#check null value
is.null(rent)
anyNA(rent)
```

Figure 206: is.null() & anyNa()

14. duplicated()

```
#check duplicated value
duplicated(rent)
```

Figure 207: duplicated()

5.0 Conclusion

In a nutshell, there may be many conditions that may affect the rental, user preference and buying power. The tenant should consider wisely and should beware of the geographical location, human relationship, date, time and house condition. It may be easy to find a cheap house, but it is challenging to find an affordable house that come with better surroundings and environment.

References

GeeksforGeeks. (2020, June 5). *Display the internal Structure of an Object in R Programming - str() Function*. https://www.geeksforgeeks.org/display-the-internal-structure-of-an-object-in-r-programming-str-function/

Versus. (n.d.). *Kolkata vs Mumbai: What is the difference?* VERSUS. https://versus.com/en/kolkata-vs-mumbai