Airbnb & ZILLOW DATA CHALLENGE

OVERVIEW OF BUSINESS PROBLEM AND DATA

Problem Statement: - The problem statement is about you being as a Consulting firm help a real estate agency to understand which are the best zip codes in New York City are to buy and rent property. The real estate agency has already figured company that 2 Bedrooms properties are the best for investment; however, they do not know which zip codes the best are to invest in. The real estate company has engaged your firm to build out a data product and provide your conclusions to help them understand which zip codes would generate the most profit on short term rentals within New York City.

SOURCE OF THE DATA

The data comes from two different sources i.e. Airbnb and Zillow.

1) Zillow Dataset:- a) Zillow provides us an estimate of value for two-bedroom properties and consist information on RegionID, RegionName, City, State, SizeRank and Cost shown between April 1996 to June 2017.

2) Airbnb Dataset:- a) Airbnb dataset contains information on property listings such as location details, number of bedrooms, room types, services, host details, cleaning fee, rent price details, reviews and ratings.

Steps to be Followed: -

A) Understand and Clean the data.
B) Highlight Quality Insights based on Analysis and Data Munging.
C) Perform Data Visualization and obtain useful information through data mining which can help solve the above business problem.

ASSUMPTIONS: -

1) The investor will pay for the property in cash (i.e. no mortgage/interest rate will need to be accounted for).

2) The time value of money discount rate is 0% (i.e. $1 today is worth the same 100 years from now).

3) All properties and all square feet within each locale can be assumed to be homogeneous (i.e. a 1000 square foot property in a locale such as Bronx or Manhattan generates twice the revenue and costs twice as much as any other 500 square foot property within that same locale.)

4) Assuming Occupancy rate is 75% as given in the statement.

5) Assuming the company will put properties on rent throughout the year every day.

6) Calculating the Latest Price of the property by using Auto.ARIMA function for the next year by considering the values of last 7 years.

7) **Approach**:- Keeping the occupancy rate 75% for each scenario(for example:- Daily, Weekly and Monthly) to check which Zip codes are profitable for daily, weekly and for monthly bookings. I assumed this because people mostly book rental places for more than a day because it is cheaper than hotels and people who travel for business book properties for single day. Also, students who come for studies can book the apartment for months.

8) By using the formula, we can segregate the zip codes based upon revenue, which zip code will give better revenue if we put the apartment for daily basis bookings, which is beneficial for weekly and monthly.

DATA LOADING

a) Loading the ZILLOW DATA
b) Zillow data contains 8946 Rows and 262 Columns

```
dfzill <- read.csv("C:/Users/Setia Comp/Downloads/CO/Zip_Zhvi_2bedroom.csv")
```

Checking the dimensions of the dataset.

```
dim(dfzill)
```

```
## [1] 8946  262
```

a) Loading the Airbnb Data.
b) The Airbnb data contains 40753 Rows and 95 Columns.

```
dflist <- read.csv("C:/Users/Setia Comp/Downloads/CO/listings.csv")
```

Checking the dimensions of the dataset.

```
dim(dflist)
```

```
## [1] 40753    95
```

Fetching the column names of Zillow data and Airbnb Data

```
colnames(dfzill)
```

```
##   [1] "RegionID"   "RegionName" "City"       "State"      "Metro"
##   [6] "CountyName" "SizeRank"   "X1996.04"   "X1996.05"   "X1996.06"
##  [11] "X1996.07"   "X1996.08"   "X1996.09"   "X1996.10"   "X1996.11"
##  [16] "X1996.12"   "X1997.01"   "X1997.02"   "X1997.03"   "X1997.04"
##  [21] "X1997.05"   "X1997.06"   "X1997.07"   "X1997.08"   "X1997.09"
##  [26] "X1997.10"   "X1997.11"   "X1997.12"   "X1998.01"   "X1998.02"
##  [31] "X1998.03"   "X1998.04"   "X1998.05"   "X1998.06"   "X1998.07"
##  [36] "X1998.08"   "X1998.09"   "X1998.10"   "X1998.11"   "X1998.12"
##  [41] "X1999.01"   "X1999.02"   "X1999.03"   "X1999.04"   "X1999.05"
```

```
##  [46] "X1999.06"   "X1999.07"   "X1999.08"   "X1999.09"   "X1999.10"
##  [51] "X1999.11"   "X1999.12"   "X2000.01"   "X2000.02"   "X2000.03"
##  [56] "X2000.04"   "X2000.05"   "X2000.06"   "X2000.07"   "X2000.08"
##  [61] "X2000.09"   "X2000.10"   "X2000.11"   "X2000.12"   "X2001.01"
##  [66] "X2001.02"   "X2001.03"   "X2001.04"   "X2001.05"   "X2001.06"
##  [71] "X2001.07"   "X2001.08"   "X2001.09"   "X2001.10"   "X2001.11"
##  [76] "X2001.12"   "X2002.01"   "X2002.02"   "X2002.03"   "X2002.04"
##  [81] "X2002.05"   "X2002.06"   "X2002.07"   "X2002.08"   "X2002.09"
##  [86] "X2002.10"   "X2002.11"   "X2002.12"   "X2003.01"   "X2003.02"
##  [91] "X2003.03"   "X2003.04"   "X2003.05"   "X2003.06"   "X2003.07"
##  [96] "X2003.08"   "X2003.09"   "X2003.10"   "X2003.11"   "X2003.12"
## [101] "X2004.01"   "X2004.02"   "X2004.03"   "X2004.04"   "X2004.05"
## [106] "X2004.06"   "X2004.07"   "X2004.08"   "X2004.09"   "X2004.10"
## [111] "X2004.11"   "X2004.12"   "X2005.01"   "X2005.02"   "X2005.03"
## [116] "X2005.04"   "X2005.05"   "X2005.06"   "X2005.07"   "X2005.08"
## [121] "X2005.09"   "X2005.10"   "X2005.11"   "X2005.12"   "X2006.01"
## [126] "X2006.02"   "X2006.03"   "X2006.04"   "X2006.05"   "X2006.06"
## [131] "X2006.07"   "X2006.08"   "X2006.09"   "X2006.10"   "X2006.11"
## [136] "X2006.12"   "X2007.01"   "X2007.02"   "X2007.03"   "X2007.04"
## [141] "X2007.05"   "X2007.06"   "X2007.07"   "X2007.08"   "X2007.09"
## [146] "X2007.10"   "X2007.11"   "X2007.12"   "X2008.01"   "X2008.02"
## [151] "X2008.03"   "X2008.04"   "X2008.05"   "X2008.06"   "X2008.07"
## [156] "X2008.08"   "X2008.09"   "X2008.10"   "X2008.11"   "X2008.12"
## [161] "X2009.01"   "X2009.02"   "X2009.03"   "X2009.04"   "X2009.05"
## [166] "X2009.06"   "X2009.07"   "X2009.08"   "X2009.09"   "X2009.10"
## [171] "X2009.11"   "X2009.12"   "X2010.01"   "X2010.02"   "X2010.03"
## [176] "X2010.04"   "X2010.05"   "X2010.06"   "X2010.07"   "X2010.08"
## [181] "X2010.09"   "X2010.10"   "X2010.11"   "X2010.12"   "X2011.01"
## [186] "X2011.02"   "X2011.03"   "X2011.04"   "X2011.05"   "X2011.06"
## [191] "X2011.07"   "X2011.08"   "X2011.09"   "X2011.10"   "X2011.11"
## [196] "X2011.12"   "X2012.01"   "X2012.02"   "X2012.03"   "X2012.04"
## [201] "X2012.05"   "X2012.06"   "X2012.07"   "X2012.08"   "X2012.09"
## [206] "X2012.10"   "X2012.11"   "X2012.12"   "X2013.01"   "X2013.02"
## [211] "X2013.03"   "X2013.04"   "X2013.05"   "X2013.06"   "X2013.07"
## [216] "X2013.08"   "X2013.09"   "X2013.10"   "X2013.11"   "X2013.12"
## [221] "X2014.01"   "X2014.02"   "X2014.03"   "X2014.04"   "X2014.05"
## [226] "X2014.06"   "X2014.07"   "X2014.08"   "X2014.09"   "X2014.10"
## [231] "X2014.11"   "X2014.12"   "X2015.01"   "X2015.02"   "X2015.03"
## [236] "X2015.04"   "X2015.05"   "X2015.06"   "X2015.07"   "X2015.08"
## [241] "X2015.09"   "X2015.10"   "X2015.11"   "X2015.12"   "X2016.01"
## [246] "X2016.02"   "X2016.03"   "X2016.04"   "X2016.05"   "X2016.06"
## [251] "X2016.07"   "X2016.08"   "X2016.09"   "X2016.10"   "X2016.11"
## [256] "X2016.12"   "X2017.01"   "X2017.02"   "X2017.03"   "X2017.04"
## [261] "X2017.05"   "X2017.06"
```

```
colnames(dflist)
```

```
##  [1] "id"                                "listing_url"
##  [3] "scrape_id"                         "last_scraped"
##  [5] "name"                              "summary"
```

```
 ##  [7] "space"                                "description"
 ##  [9] "experiences_offered"                  "neighborhood_overview"
## [11] "notes"                                "transit"
## [13] "access"                               "interaction"
## [15] "house_rules"                          "thumbnail_url"
## [17] "medium_url"                           "picture_url"
## [19] "xl_picture_url"                       "host_id"
## [21] "host_url"                             "host_name"
## [23] "host_since"                           "host_location"
## [25] "host_about"                           "host_response_time"
## [27] "host_response_rate"                   "host_acceptance_rate"
## [29] "host_is_superhost"                    "host_thumbnail_url"
## [31] "host_picture_url"                     "host_neighbourhood"
## [33] "host_listings_count"                  "host_total_listings_count"
## [35] "host_verifications"                   "host_has_profile_pic"
## [37] "host_identity_verified"               "street"
## [39] "neighbourhood"                        "neighbourhood_cleansed"
## [41] "neighbourhood_group_cleansed"         "city"
## [43] "state"                                "zipcode"
## [45] "market"                               "smart_location"
## [47] "country_code"                         "country"
## [49] "latitude"                             "longitude"
## [51] "is_location_exact"                    "property_type"
## [53] "room_type"                            "accommodates"
## [55] "bathrooms"                            "bedrooms"
## [57] "beds"                                 "bed_type"
## [59] "amenities"                            "square_feet"
## [61] "price"                                "weekly_price"
## [63] "monthly_price"                        "security_deposit"
## [65] "cleaning_fee"                         "guests_included"
## [67] "extra_people"                         "minimum_nights"
## [69] "maximum_nights"                       "calendar_updated"
## [71] "has_availability"                     "availability_30"
## [73] "availability_60"                      "availability_90"
## [75] "availability_365"                     "calendar_last_scraped"
## [77] "number_of_reviews"                    "first_review"
## [79] "last_review"                          "review_scores_rating"
## [81] "review_scores_accuracy"               "review_scores_cleanliness"
## [83] "review_scores_checkin"                "review_scores_communication"
## [85] "review_scores_location"               "review_scores_value"
## [87] "requires_license"                     "license"
## [89] "jurisdiction_names"                   "instant_bookable"
## [91] "cancellation_policy"                  "require_guest_profile_picture"
## [93] "require_guest_phone_verification"     "calculated_host_listings_count"
## [95] "reviews_per_month"
```

DATA MUNGING

Filtering out th Zillow Dataset

Multiple steps will be executed to perform data munging. These steps are as follows:

a) Creating a data frame and Selecting only relevant columns such as RegionName, City, SizeRank and the cost property from last 7 years to forecast Latest Price of the property.

```
Zillowdf <- dfzill[,c(2,3,7,190:262)]
head(Zillowdf)
```

```
##   RegionName      City SizeRank X2011.06  X2011.07 X2011.08 X2011.09
## 1      10025  New York        1   897700   896300   892300    890400
## 2      60657   Chicago        2   301800   299500   299900    301100
## 3      10023  New York        3  1367900  1365400  1375100   1380400
## 4      60614   Chicago        4   326000   326100   326700    326300
## 5      79936   El Paso        5    83300    83100    82800     82500
## 6      60640   Chicago        6   213200   210900   209600   208200
##    X2011.10 X2011.11 X2011.12 X2012.01 X2012.02 X2012.03 X2012.04  X2012.05
## 1    888600   891700   899500   904400   908200   914000   915100    912300
## 2    300300   298900   298500   298500   297000   296800   298700    299600
## 3   1377000  1375100  1379000  1395200  1414500  1419000  1403100   1383200
## 4    324400   322700   323200   322800   320700   319500   320100    320500
## 5     82300    82200    82300    82200    81900    81700    82100     82600
## 6    205900   204000   203200   202500   200800   199400   199900    201900
##    X2012.06 X2012.07 X2012.08 X2012.09 X2012.10 X2012.11 X2012.12  X2013.01
## 1    914000   921100   923300   917300   915000   922800   929100    937700
## 2    300700   303900   306800   307500   308500   310000   310800    311200
## 3   1376700  1378200  1378700  1375900  1366700  1365500  1382200   1404700
## 4    321800   323600   324300   324100   324700   326000   327600    329800
## 5     82900    83000    83000    82900    82100    81200    80800     80700
## 6    204500   207000   208100   207100   205300   204700   204700    205800
##    X2013.02 X2013.03 X2013.04 X2013.05 X2013.06 X2013.07 X2013.08  X2013.09
## 1    955700   974200   995500  1019500  1035100  1054900  1079900   1092600
## 2    313000   315800   319000   323400   327500   330000   331800    334500
## 3   1428000  1445700  1452900  1460100  1484400  1508400  1522800   1538300
## 4    332600   336800   342300   348100   353600   358900   361900    363900
## 5     81200    81800    81800    81400    81400    81500    81900     82000
## 6    208600   211800   213500   213800   215100   218400   221500    223900
##    X2013.10 X2013.11 X2013.12 X2014.01 X2014.02 X2014.03 X2014.04  X2014.05
## 1   1103500  1118800  1139300  1154600  1144100  1120300  1125500   1136000
## 2    336000   335700   335400   336300   338800   342400   344400    344000
## 3   1568600  1597400  1622900  1654300  1684600  1713000  1728800   1736100
## 4    366200   368300   369800   371400   372400   373200   373800    374800
## 5     81900    81900    82100    82100    81500    80800    80300     80100
```

```
## 6    226100   227900   229100   230200   230600   230400   230000   229000
##    X2014.06 X2014.07 X2014.08 X2014.09 X2014.10 X2014.11 X2014.12 X2015.01
## 1   1135100  1130000  1138200  1153700  1174800  1185400  1188400  1189700
## 2    343900   345100   346100   346900   348000   349700   351200   351700
## 3   1745900  1753800  1736600  1730400  1734500  1728700  1720800  1717700
## 4    376200   376800   376300   374900   373800   373900   374700   375300
## 5     80100    80700    81200    81700    81900    81700    81500    81700
## 6    227600   226100   225700   226200   226500   226500   227100   227800
##    X2015.02 X2015.03 X2015.04 X2015.05 X2015.06 X2015.07 X2015.08 X2015.09
## 1   1193700  1199900  1201400  1202600  1214200  1235200  1258000  1287700
## 2    350700   350400   352000   354300   355900   356500   355200   353800
## 3   1700100  1680400  1676400  1685600  1708100  1730400  1751800  1778300
## 4    375000   374700   376300   378100   378000   377700   378300   380000
## 5     81700    80900    81000    81500    81400    80500    80000    80100
## 6    229400   231800   234100   235400   235100   233900   233700   235300
##    X2015.10 X2015.11 X2015.12 X2016.01 X2016.02 X2016.03 X2016.04 X2016.05
## 1   1307200  1313900  1317100  1327400  1338800  1350400  1356600  1358500
## 2    353700   354600   356200   357800   358200   358500   360300   362400
## 3   1810400  1831600  1844400  1861600  1889600  1901500  1895300  1890200
## 4    383100   385900   388100   389700   391800   393400   394700   394900
## 5     80500    80800    81400    82300    82600    82600    82500    82500
## 6    237200   238500   239300   239600   239500   240200   242700   244900
##    X2016.06 X2016.07 X2016.08 X2016.09 X2016.10 X2016.11 X2016.12 X2017.01
## 1   1364000  1373300  1382600  1374400  1364100  1366300  1354800  1327500
## 2    363700   365200   367100   368600   370200   372300   375300   378700
## 3   1898400  1924500  1967300  1993500  1980700  1960900  1951300  1937800
## 4    395700   396400   397500   398900   401200   403200   405700   408300
## 5     82600    82700    82600    82400    82300    82400    82300    82500
## 6    247700   249500   248800   247000   247300   248700   250800   252800
##    X2017.02 X2017.03 X2017.04 X2017.05 X2017.06
## 1   1317300  1333700  1352100  1390000  1431000
## 2    381400   381800   382100   383300   385100
## 3   1929800  1955000  2022400  2095000  2142300
## 4    408800   408000   410100   412200   412200
## 5     83200    83900    84100    83900    83700
## 6    253800   253800   253400   254100   255100
```

b) Installing the necessary packages. These packages are used for data pre-processing, cleaning, transformation and Visualization.

c)From the selected columns filtering the the cityname which in our case is New York only.

```
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.4.4

##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

Zillowdf <- filter(Zillowdf,City =="New York")

## Warning: package 'bindrcpp' was built under R version 3.4.4

head(Zillowdf)

##    RegionName     City SizeRank X2011.06 X2011.07 X2011.08 X2011.09
## 1       10025 New York         1   897700   896300   892300   890400
## 2       10023 New York         3  1367900  1365400  1375100  1380400
## 3       10128 New York        14   975700   988600   998000  1019700
## 4       10011 New York        15  1426800  1427800  1424600  1432800
## 5       10003 New York        21  1361700  1357800  1364400  1358000
## 6       11201 New York        32   795300   799300   798000   801800
##   X2011.10 X2011.11 X2011.12 X2012.01 X2012.02 X2012.03 X2012.04 X2012.05
## 1   888600   891700   899500   904400   908200   914000   915100   912300
## 2  1377000  1375100  1379000  1395200  1414500  1419000  1403100  1383200
## 3  1045500  1064200  1066000  1057500  1058600  1069600  1068700  1054500
## 4  1456500  1485100  1500200  1509600  1518500  1530800  1538000  1530500
## 5  1329800  1317800  1333200  1348500  1349500  1352200  1354100  1351900
## 6   808000   808200   805600   805100   813000   825500   835700   839100
##   X2012.06 X2012.07 X2012.08 X2012.09 X2012.10 X2012.11 X2012.12 X2013.01
## 1   914000   921100   923300   917300   915000   922800   929100   937700
## 2  1376700  1378200  1378700  1375900  1366700  1365500  1382200  1404700
## 3  1045000  1043400  1050300  1050500  1050700  1059700  1079600  1091600
## 4  1524500  1546500  1574800  1599600  1622500  1639000  1656100  1684600
## 5  1364200  1376600  1384200  1387900  1404200  1419200  1425700  1435300
## 6   836700   836900   843900   852300   857400   859900   858700   857100
##   X2013.02 X2013.03 X2013.04 X2013.05 X2013.06 X2013.07 X2013.08 X2013.09
## 1   955700   974200   995500  1019500  1035100  1054900  1079900  1092600
## 2  1428000  1445700  1452900  1460100  1484400  1508400  1522800  1538300
## 3  1106100  1121700  1139900  1153100  1174400  1182500  1170800  1166000
## 4  1703000  1710000  1734300  1765200  1786000  1810700  1841500  1867600
## 5  1460300  1466500  1458100  1465500  1502300  1563900  1592000  1596200
## 6   856800   863400   875900   891000   909500   937200   965400   992200
##   X2013.10 X2013.11 X2013.12 X2014.01 X2014.02 X2014.03 X2014.04 X2014.05
## 1  1103500  1118800  1139300 1154600   1144100  1120300  1125500  1136000
## 2  1568600  1597400  1622900 1654300   1684600  1713000  1728800  1736100
## 3  1172700  1171800  1173000 1187000   1200200  1209600  1214800  1218800
## 4  1882200  1897000  1917300 1963400   1999200  2003500  2007900  2027700
## 5  1625200  1672300  1699500 1718500   1734300  1748600  1763700  1766700
## 6  1008700  1014300  1020800 1040000   1058000  1074200  1098300  1124500
##   X2014.06 X2014.07 X2014.08 X2014.09 X2014.10 X2014.11 X2014.12 X2015.01
## 1  1135100  1130000  1138200  1153700  1174800  1185400  1188400  1189700
```

```
## 2   1745900  1753800  1736600  1730400  1734500  1728700  1720800  1717700
## 3   1221200  1230500  1243500  1259000  1277400  1296300  1305600  1310800
## 4   2043500  2056300  2064500  2066000  2057900  2031300  1999000  1979200
## 5   1772200  1762700  1736700  1712400  1703700  1702500  1708800  1716300
## 6   1140900  1156900  1182000  1207600  1223800  1231600  1240500  1253600
##    X2015.02 X2015.03 X2015.04 X2015.05 X2015.06 X2015.07 X2015.08 X2015.09
## 1   1193700  1199900  1201400  1202600  1214200  1235200  1258000  1287700
## 2   1700100  1680400  1676400  1685600  1708100  1730400  1751800  1778300
## 3   1313400  1313500  1314500  1328000  1347900  1376100  1409500  1431400
## 4   1982900  2001600  2014700  2023500  2055300  2078300  2083600  2088800
## 5   1720500  1721800  1741800  1775800  1796500  1821500  1870100  1901000
## 6   1264500  1270500  1276300  1289600  1303800  1305300  1298900  1301000
##    X2015.10 X2015.11 X2015.12 X2016.01 X2016.02 X2016.03 X2016.04 X2016.05
## 1   1307200  1313900  1317100  1327400  1338800  1350400  1356600  1358500
## 2   1810400  1831600  1844400  1861600  1889600  1901500  1895300  1890200
## 3   1441600  1453100  1468100  1492000  1518100  1531300  1525300  1509000
## 4   2110600  2127500  2168900  2204700  2216100  2212500  2222600  2231900
## 5   1904900  1914000  1926400  1932200  1936700  1945200  1935600  1911200
## 6   1314200  1322800  1320500  1318800  1325600  1333000  1334800  1333100
##    X2016.06 X2016.07 X2016.08 X2016.09 X2016.10  X2016.11 X2016.12 X2017.01
## 1   1364000  1373300  1382600  1374400  1364100  1366300  1354800  1327500
## 2   1898400  1924500  1967300  1993500  1980700  1960900  1951300  1937800
## 3   1520400  1543900  1547400  1526000  1523700  1527200  1541600  1557800
## 4   2250800  2285200  2329100  2354000  2355500  2352200  2332100  2313300
## 5   1918700  1947600  1951300  1932800  1930400  1937500  1935100  1915700
## 6   1334600  1339000  1343000  1340200  1338700  1350600  1375600  1390200
##    X2017.02 X2017.03 X2017.04 X2017.05 X2017.06
## 1   1317300  1333700  1352100  1390000  1431000
## 2   1929800  1955000  2022400  2095000  2142300
## 3   1582900  1598900  1646100  1720500  1787100
## 4   2319600  2342100  2365900  2419700  2480400
## 5   1916500  1965700  2045300  2109100  2147000
## 6   1398100  1399900  1400500  1407300  1420700
```

d) Changing the column name RegionName to zipcode so we can merge both the dataset by this unique Id.

```
colnames(Zillowdf)[1] <- "zipcode"
colnames(Zillowdf)[1]
```

```
## [1] "zipcode"
```

```
head(Zillowdf)
```

```
##    zipcode     City SizeRank X2011.06 X2011.07 X2011.08 X2011.09 X2011.10
## 1    10025 New York         1   897700   896300   892300   890400   888600
## 2    10023 New York         3  1367900  1365400  1375100  1380400  1377000
## 3    10128 New York        14   975700   988600   998000  1019700  1045500
## 4    10011 New York        15  1426800  1427800  1424600  1432800  1456500
## 5    10003 New York        21  1361700  1357800  1364400  1358000  1329800
## 6    11201 New York        32   795300   799300   798000   801800   808000
```

```
##   X2011.11 X2011.12 X2012.01 X2012.02 X2012.03 X2012.04 X2012.05 X2012.06
 ## 1   891700   899500   904400   908200   914000   915100   912300   914000
 ## 2  1375100  1379000  1395200  1414500  1419000  1403100  1383200  1376700
 ## 3  1064200  1066000  1057500  1058600  1069600  1068700  1054500  1045000
 ## 4  1485100  1500200  1509600  1518500  1530800  1538000  1530500  1524500
 ## 5  1317800  1333200  1348500  1349500  1352200  1354100  1351900  1364200
 ## 6   808200   805600   805100   813000   825500   835700   839100   836700
##   X2012.07 X2012.08 X2012.09 X2012.10 X2012.11 X2012.12 X2013.01 X2013.02
 ## 1   921100   923300   917300   915000   922800   929100   937700   955700
 ## 2  1378200  1378700  1375900  1366700  1365500  1382200  1404700  1428000
 ## 3  1043400  1050300  1050500  1050700  1059700  1079600  1091600  1106100
 ## 4  1546500  1574800  1599600  1622500  1639000  1656100  1684600  1703000
 ## 5  1376600  1384200  1387900  1404200  1419200  1425700  1435300  1460300
 ## 6   836900   843900   852300   857400   859900   858700   857100   856800
##   X2013.03 X2013.04 X2013.05 X2013.06 X2013.07 X2013.08 X2013.09 X2013.10
 ## 1   974200   995500  1019500  1035100  1054900  1079900  1092600  1103500
 ## 2  1445700  1452900  1460100  1484400  1508400  1522800  1538300  1568600
 ## 3  1121700  1139900  1153100  1174400  1182500  1170800  1166000  1172700
 ## 4  1710000  1734300  1765200  1786000  1810700  1841500  1867600  1882200
 ## 5  1466500  1458100  1465500  1502300  1563900  1592000  1596200  1625200
 ## 6   863400   875900   891000   909500   937200   965400   992200  1008700
##   X2013.11 X2013.12 X2014.01 X2014.02 X2014.03 X2014.04 X2014.05 X2014.06
 ## 1  1118800  1139300  1154600  1144100  1120300  1125500  1136000  1135100
 ## 2  1597400  1622900  1654300  1684600  1713000  1728800  1736100  1745900
 ## 3  1171800  1173000  1187000  1200200  1209600  1214800  1218800  1221200
 ## 4  1897000  1917300  1963400  1999200  2003500  2007900  2027700  2043500
 ## 5  1672300  1699500  1718500  1734300  1748600  1763700  1766700  1772200
 ## 6  1014300  1020800  1040000  1058000  1074200  1098300  1124500  1140900
##   X2014.07 X2014.08 X2014.09 X2014.10 X2014.11 X2014.12 X2015.01 X2015.02
 ## 1  1130000  1138200  1153700  1174800  1185400 1188400   1189700  1193700
 ## 2  1753800  1736600  1730400  1734500  1728700 1720800   1717700  1700100
 ## 3  1230500  1243500  1259000  1277400  1296300 1305600   1310800  1313400
 ## 4  2056300  2064500  2066000  2057900  2031300 1999000   1979200  1982900
 ## 5  1762700  1736700  1712400  1703700  1702500 1708800   1716300  1720500
 ## 6  1156900  1182000  1207600  1223800  1231600 1240500   1253600  1264500
##   X2015.03 X2015.04 X2015.05 X2015.06 X2015.07 X2015.08 X2015.09 X2015.10
 ## 1  1199900  1201400  1202600  1214200  1235200 1258000   1287700  1307200
 ## 2  1680400  1676400  1685600  1708100  1730400 1751800   1778300  1810400
 ## 3  1313500  1314500  1328000  1347900  1376100 1409500   1431400  1441600
 ## 4  2001600  2014700  2023500  2055300  2078300 2083600   2088800  2110600
 ## 5  1721800  1741800  1775800  1796500  1821500 1870100   1901000  1904900
 ## 6  1270500  1276300  1289600  1303800  1305300 1298900   1301000  1314200
##   X2015.11 X2015.12 X2016.01 X2016.02 X2016.03 X2016.04 X2016.05 X2016.06
## 1  1313900  1317100  1327400  1338800  1350400  1356600  1358500  1364000
## 2  1831600  1844400  1861600  1889600  1901500  1895300  1890200  1898400
## 3  1453100  1468100  1492000  1518100  1531300  1525300  1509000  1520400
## 4  2127500  2168900  2204700  2216100  2212500  2222600  2231900  2250800
## 5  1914000  1926400  1932200  1936700  1945200  1935600  1911200  1918700
## 6  1322800  1320500  1318800  1325600  1333000  1334800  1333100  1334600
##    X2016.07 X2016.08 X2016.09 X2016.10 X2016.11  X2016.12 X2017.01 X2017.02
```

```
## 1   1373300  1382600  1374400  1364100  1366300  1354800  1327500 1317300
## 2   1924500  1967300  1993500  1980700  1960900  1951300  1937800 1929800
## 3   1543900  1547400  1526000  1523700  1527200  1541600  1557800 1582900
## 4   2285200  2329100  2354000  2355500  2352200  2332100  2313300 2319600
## 5   1947600  1951300  1932800  1930400  1937500  1935100  1915700 1916500
## 6   1339000  1343000  1340200  1338700  1350600  1375600  1390200 1398100
##    X2017.03 X2017.04 X2017.05  X2017.06
## 1   1333700  1352100  1390000  1431000
## 2   1955000  2022400  2095000  2142300
## 3   1598900  1646100  1720500  1787100
## 4   2342100  2365900  2419700  2480400
## 5   1965700  2045300  2109100  2147000
## 6   1399900  1400500  1407300  1420700
```

e) Introducing the new column (Latest price) i.e current price of the properties to our Zillow data frame as forecasted value based upon previous values.

```
Zillowdf$LatestPrice <- NA
colnames(Zillowdf)
```

```
 ##  [1] "zipcode"     "City"        "SizeRank"    "X2011.06"    "X2011.07"
 ##  [6] "X2011.08"    "X2011.09"    "X2011.10"    "X2011.11"    "X2011.12"
## [11] "X2012.01"    "X2012.02"    "X2012.03"    "X2012.04"    "X2012.05"
## [16] "X2012.06"    "X2012.07"    "X2012.08"    "X2012.09"    "X2012.10"
## [21] "X2012.11"    "X2012.12"    "X2013.01"    "X2013.02"    "X2013.03"
## [26] "X2013.04"    "X2013.05"    "X2013.06"    "X2013.07"    "X2013.08"
## [31] "X2013.09"    "X2013.10"    "X2013.11"    "X2013.12"    "X2014.01"
## [36] "X2014.02"    "X2014.03"    "X2014.04"    "X2014.05"    "X2014.06"
## [41] "X2014.07"    "X2014.08"    "X2014.09"    "X2014.10"    "X2014.11"
## [46] "X2014.12"    "X2015.01"    "X2015.02"    "X2015.03"    "X2015.04"
## [51] "X2015.05"    "X2015.06"    "X2015.07"    "X2015.08"    "X2015.09"
## [56] "X2015.10"    "X2015.11"    "X2015.12"    "X2016.01"    "X2016.02"
## [61] "X2016.03"    "X2016.04"    "X2016.05"    "X2016.06"    "X2016.07"
## [66] "X2016.08"    "X2016.09"    "X2016.10"    "X2016.11"    "X2016.12"
## [71] "X2017.01"    "X2017.02"    "X2017.03"    "X2017.04"    "X2017.05"
## [76] "X2017.06"    "LatestPrice"
```

f) We are assuming that there is seasonality in the price and that values depend not only on previous values (Auto Regressive AR) but also on differences between previous values (Moving Average MA) So we apply Auto.ARIMA model to predict the cost of the properties in Zip codes from July 2017 to July 2018.

```
for(i in 1:nrow(Zillowdf)){
  zillts = ts(as.vector(t(Zillowdf[,c(4:76)])[,i]),start =
c(2011,06),frequency = 12)
  ARfit = arima(zillts, order=c(1,1,1),
seasonal=list(order=c(1,0,1),period=NA),
                        method="ML")
```

```
  pred_val = predict(ARfit, n.ahead = 13)

  predval <- pred_val$pred
  Zillowdf$LatestPrice[i] <- as.integer(predval[length(predval)])
}
```

```
## Warning in log(s2): NaNs produced
```

```
head(Zillowdf)
```

```
##    zipcode     City SizeRank X2011.06 X2011.07 X2011.08 X2011.09 X2011.10
## 1    10025 New York         1   897700   896300   892300   890400   888600
## 2    10023 New York         3  1367900  1365400  1375100  1380400  1377000
## 3    10128 New York        14   975700   988600   998000  1019700  1045500
## 4    10011 New York        15  1426800  1427800  1424600  1432800  1456500
## 5    10003 New York        21  1361700  1357800  1364400  1358000  1329800
## 6    11201 New York        32   795300   799300   798000   801800   808000
##   X2011.11 X2011.12 X2012.01 X2012.02 X2012.03 X2012.04 X2012.05 X2012.06
## 1   891700   899500   904400   908200   914000   915100   912300   914000
## 2  1375100  1379000  1395200  1414500  1419000  1403100  1383200  1376700
## 3  1064200  1066000  1057500  1058600  1069600  1068700  1054500  1045000
## 4  1485100  1500200  1509600  1518500  1530800  1538000  1530500  1524500
## 5  1317800  1333200  1348500  1349500  1352200  1354100  1351900  1364200
## 6   808200   805600   805100   813000   825500   835700   839100   836700
##   X2012.07 X2012.08 X2012.09 X2012.10 X2012.11 X2012.12 X2013.01 X2013.02
## 1   921100   923300   917300   915000   922800   929100   937700   955700
## 2  1378200  1378700  1375900  1366700  1365500  1382200  1404700  1428000
## 3  1043400  1050300  1050500  1050700  1059700  1079600  1091600  1106100
## 4  1546500  1574800  1599600  1622500  1639000  1656100  1684600  1703000
## 5  1376600  1384200  1387900  1404200  1419200  1425700  1435300  1460300
## 6   836900   843900   852300   857400   859900   858700   857100   856800
##   X2013.03 X2013.04 X2013.05 X2013.06 X2013.07 X2013.08 X2013.09 X2013.10
## 1   974200   995500  1019500  1035100  1054900  1079900  1092600  1103500
## 2  1445700  1452900  1460100  1484400  1508400  1522800  1538300  1568600
## 3  1121700  1139900  1153100  1174400  1182500  1170800  1166000  1172700
## 4  1710000  1734300  1765200  1786000  1810700  1841500  1867600  1882200
## 5  1466500  1458100  1465500  1502300  1563900  1592000  1596200  1625200
## 6   863400   875900   891000   909500   937200   965400   992200  1008700
##   X2013.11 X2013.12 X2014.01 X2014.02 X2014.03 X2014.04 X2014.05 X2014.06
## 1  1118800  1139300  1154600  1144100  1120300  1125500  1136000  1135100
## 2  1597400  1622900  1654300  1684600  1713000  1728800  1736100  1745900
## 3  1171800  1173000  1187000  1200200  1209600  1214800  1218800  1221200
## 4  1897000  1917300  1963400  1999200  2003500  2007900  2027700  2043500
## 5  1672300  1699500  1718500  1734300  1748600  1763700  1766700  1772200
## 6  1014300  1020800  1040000  1058000  1074200  1098300  1124500  1140900
##   X2014.07 X2014.08 X2014.09 X2014.10 X2014.11 X2014.12 X2015.01 X2015.02
## 1  1130000  1138200  1153700  1174800  1185400  1188400   1189700  1193700
## 2  1753800  1736600  1730400  1734500  1728700  1720800   1717700  1700100
## 3  1230500  1243500  1259000  1277400  1296300  1305600   1310800  1313400
## 4  2056300  2064500  2066000  2057900  2031300  1999000   1979200  1982900
```

```
## 5   1762700  1736700  1712400  1703700  1702500  1708800  1716300  1720500
## 6   1156900  1182000  1207600  1223800  1231600  1240500  1253600  1264500
##    X2015.03 X2015.04 X2015.05 X2015.06 X2015.07 X2015.08 X2015.09 X2015.10
## 1   1199900  1201400  1202600  1214200  1235200  1258000  1287700  1307200
## 2   1680400  1676400  1685600  1708100  1730400  1751800  1778300  1810400
## 3   1313500  1314500  1328000  1347900  1376100  1409500  1431400  1441600
## 4   2001600  2014700  2023500  2055300  2078300  2083600  2088800  2110600
## 5   1721800  1741800  1775800  1796500  1821500  1870100  1901000  1904900
## 6   1270500  1276300  1289600  1303800  1305300  1298900  1301000  1314200
##    X2015.11 X2015.12 X2016.01 X2016.02 X2016.03 X2016.04 X2016.05 X2016.06
## 1   1313900  1317100  1327400  1338800  1350400  1356600  1358500  1364000
## 2   1831600  1844400  1861600  1889600  1901500  1895300  1890200  1898400
## 3   1453100  1468100  1492000  1518100  1531300  1525300  1509000  1520400
## 4   2127500  2168900  2204700  2216100  2212500  2222600  2231900  2250800
## 5   1914000  1926400  1932200  1936700  1945200  1935600  1911200  1918700
## 6   1322800  1320500  1318800  1325600  1333000  1334800  1333100  1334600
##    X2016.07 X2016.08 X2016.09 X2016.10 X2016.11 X2016.12 X2017.01 X2017.02
## 1   1373300  1382600  1374400  1364100  1366300  1354800  1327500  1317300
## 2   1924500  1967300  1993500  1980700  1960900  1951300  1937800  1929800
## 3   1543900  1547400  1526000  1523700  1527200  1541600  1557800  1582900
## 4   2285200  2329100  2354000  2355500  2352200  2332100  2313300  2319600
## 5   1947600  1951300  1932800  1930400  1937500  1935100  1915700  1916500
## 6   1339000  1343000  1340200  1338700  1350600  1375600  1390200  1398100
 ##    X2017.03 X2017.04 X2017.05 X2017.06 LatestPrice
## 1   1333700  1352100  1390000  1431000     1460272
## 2   1955000  2022400  2095000  2142300     2173876
## 3   1598900  1646100  1720500  1787100     1983660
## 4   2342100  2365900  2419700  2480400     2620588
## 5   1965700  2045300  2109100  2147000     2167533
## 6   1399900  1400500  1407300  1420700     1479431
```

g) Extracting the required columns for further Data Analysis and let's look at the top five rows from clean Zillow data.

```
Zillowdf <- Zillowdf[,c(1,2,3,77)]
head(Zillowdf)
```

```
##    zipcode      City SizeRank LatestPrice
## 1    10025 New York         1     1460272
## 2    10023 New York         3     2173876
## 3    10128 New York        14     1983660
## 4    10011 New York        15     2620588
## 5    10003 New York        21     2167533
## 6    11201 New York        32     1479431
```

FILTERING THE LISTING DATA

a) Filter the Listings data to obtain only those data points which correspond to properties having 2 bedrooms.

b) We are selecting columns which are containing relevant information about rent values, column names include (id, zip code, bedrooms, price, weekely_price, monthly_price, cleaning fee, number_of_reviews, review_scores_rating).

```
listdf <- dflist[,c(1,44,56,61:63,65,77,80)]
colnames(listdf)
```

```
 ## [1] "id"                    "zipcode"              "bedrooms"
 ## [4] "price"                 "weekly_price"         "monthly_price"
 ## [7] "cleaning_fee"          "number_of_reviews"    "review_scores_rating"
```

c) From the selected columns filtering the bedrooms = 2 as per the case study.

```
library(dplyr)
listdf <- filter(listdf,bedrooms==2)
head(listdf)
```

```
##          id zipcode bedrooms    price weekly_price monthly_price
## 1   9513511   10462        2  $130.00
## 2   5046189   10469        2  $150.00
## 3   4357134   11102        2  $200.00
## 4 16027061    11102        2  $250.00
## 5 11301089    11105        2   $79.00
## 6 14855080    11105        2  $225.00
##   cleaning_fee number_of_reviews review_scores_rating
 ## 1                              4                  85
 ## 2       $75.00                31                  95
 ## 3                              0                  NA
 ## 4                              0                  NA
 ## 5      $400.00                 1                  60
 ## 6       $95.00                15                  87
```

COMBINING THE ZILLOW AND LISTING DATA

a) Using the merge data to combine both the dataset based on the common unique key- Zipcode.

```
mergedata <- merge(listdf,Zillowdf,by = c("zipcode"))
colnames(mergedata)
```

```
 ##  [1] "zipcode"              "id"                   "bedrooms"
 ##  [4] "price"                "weekly_price"         "monthly_price"
 ##  [7] "cleaning_fee"         "number_of_reviews"    "review_scores_rating"
 ## [10] "City"                 "SizeRank"             "LatestPrice"
```

b) Looking at the top 5 values of merge dataset. Now we will look at the structure and summary of the merge dataset.

```
head(mergedata)
```

```
##    zipcode       id bedrooms    price weekly_price monthly_price
## 1    10003 13561752        2 $450.00
## 2    10003  4942107        2 $989.00
## 3    10003   711635        2 $240.00    $1,365.00     $5,460.00
## 4    10003  4510857        2 $119.00      $850.00     $3,200.00
## 5    10003  3799598        2 $240.00
## 6    10003   568743        2 $159.00    $1,500.00     $6,000.00
##   cleaning_fee number_of_reviews review_scores_rating      City SizeRank
## 1      $150.00                14                    94 New York        21
## 2                             37                   100 New York        21
## 3       $50.00                63                    95 New York        21
## 4       $60.00                 2                    80 New York        21
## 5       $75.00               144                    88 New York        21
## 6       $75.00               137                    83 New York        21
##   LatestPrice
## 1     2167533
## 2     2167533
## 3     2167533
## 4     2167533
## 5     2167533
## 6     2167533
```

c) We can see that above output merged data contains lot of issues in data such as NAs, incorrect data symbols such as $ and wrong data types. So we move to data cleaning tab to clean the data

```
str(mergedata)
```

```
## 'data.frame':    1238 obs. of  12 variables:
##  $ zipcode            : Factor w/ 205 levels "","05340","07310",..: 8 8 8
8 8 8 8 8 8 8 ...
##  $ id                 : int   13561752 4942107 711635 4510857 3799598
```

```
568743 8335547 15094880  7664343  8884228 ...
##  $ bedrooms            : int  2 2 2 2 2 2 2 2  2 2 ...
##  $ price               : Factor w/ 583 levels "$1,000.00","$1,021.00",..:
378 579 194 53 194  96 205 182 120 263 ...
##  $ weekly_price        : Factor w/ 786 levels "","$1,000.00",..: 1 1 110
716 1 147 1 1 2 164 ...
##  $ monthly_price       : Factor w/ 839 levels "","$1,000.00",..: 1 1 646
452 1 675 1 1 1 659 ...
##  $ cleaning_fee        : Factor w/ 172 levels "","$0.00","$10.00",..: 37 1
121 133 148 148 159 121  4 4 ...
##  $ number_of_reviews   : int  14 37 63 2 144 137 1 26 2 72 ...
##  $ review_scores_rating: int  94 100 95 80 88 83 100 95 90 95 ...
##  $ City                : Factor w/ 4684 levels "Aberdeen","Abilene",..:
2702 2702  2702 2702 2702 2702 2702 2702 2702 2702 ...
##  $ SizeRank            : int  21 21 21  21 21 21 21 21 21 21 ...
##  $ LatestPrice         : int  2167533 2167533 2167533 2167533 2167533
2167533 2167533 2167533  2167533 2167533  2167533 ...

summary(mergedata)

##      zipcode           id              bedrooms     price
##  11215   :141   Min.    :   20853   Min.    :2  $250.00: 93
##  10003   :133   1st Qu.: 4218751    1st Qu.:2  $200.00: 72
##  10025   :112   Median : 9410246    Median :2   $300.00: 59
##  10036   :108   Mean    : 9218383   Mean    :2  $150.00: 56
##  10011   :102    3rd Qu.:14476782    3rd Qu.:2  $350.00: 42
##  10014   : 95   Max.    :18508770   Max.    :2  $400.00: 32
##   (Other):547                                  (Other):884
##     weekly_price   monthly_price   cleaning_fee number_of_reviews
 ##           :981            :1058          :236  Min.    :  0.00
 ##  $1,200.00: 14   $4,000.00:  12   $100.00:208  1st Qu.:   1.00
 ##  $1,500.00: 13   $6,000.00:   8   $50.00  : 87  Median :   5.00
 ##  $1,100.00: 10   $3,000.00:   6   $80.00  : 81  Mean    : 17.43
 ##  $1,000.00:  9   $5,000.00:   6   $150.00: 80  3rd Qu.: 20.00
 ##  $1,400.00:  8   $2,500.00:   5   $75.00  : 73  Max.    :306.00
 ## (Other)   :203   (Other)  : 143  (Other):473
 ## review_scores_rating        City        SizeRank        LatestPrice
 ## Min.    :  20.00      New York:1238   Min.    :   1.0  Min.    : 364632
 ## 1st Qu.:  90.00       Aberdeen:  0   1st Qu.:   15.0   1st Qu.:1336690
 ## Median :  95.00       Abilene :  0   Median :   71.0  Median :1904404
 ## Mean    :  93.12      Abingdon:  0   Mean    : 485.9  Mean    :1844515
 ##  3rd Qu.:100.00       Abington:  0   3rd Qu.: 580.0    3rd Qu.:2220092
 ## Max.    :100.00       Acton   :  0   Max.    :4149.0  Max.    :3293975
 ## NA's   :268           (Other) :  0
```

DATA CLEANING

a) Settling the standard names of the columns in merged data.
b) Correcting the number of levels in the data by including only New York city.

```
colnames(mergedata) <-
c("zipcode","id","bedrooms","price","weekly_price","monthly_price","cleaning_
fee","number_of_reviews","review_scores_rating","city","size_rank","LatestPri
ce")
mergedata$city <- factor(mergedata$city, levels=c("New York"))
```

```
head(mergedata)
```

```
##    zipcode       id bedrooms   price weekly_price monthly_price
## 1    10003 13561752        2 $450.00
## 2    10003  4942107        2 $989.00
## 3    10003   711635        2 $240.00    $1,365.00    $5,460.00
## 4    10003  4510857        2 $119.00      $850.00    $3,200.00
## 5    10003  3799598        2 $240.00
## 6    10003   568743        2 $159.00    $1,500.00    $6,000.00
##    cleaning_fee number_of_reviews review_scores_rating      city size_rank
## 1       $150.00                14                   94 New York         21
## 2                              37                  100 New York         21
## 3        $50.00                63                   95 New York         21
## 4        $60.00                 2                   80 New York         21
## 5        $75.00               144                   88 New York         21
## 6        $75.00               137                   83 New York         21
##   LatestPrice
## 12167533
## 22167533
## 32167533
## 42167533
## 52167533
 ## 6     2167533
```

c) Variables price (daily rent), weekly_price, monthly_price and cleaning_fee contain symbols such as "$" attached which would prevent these columns from being used for numerical analysis.

```
colnm <- c("price", "weekly_price",
"monthly_price","cleaning_fee") replacing_dollar <- function(x){
  price <- as.numeric(gsub("[$,]","",x))
  return(price)
}
mergedata[colnm] <- lapply(mergedata[colnm], replacing_dollar)
head(mergedata)
```

```
##    zipcode       id bedrooms price weekly_price monthly_price cleaning_fee
## 1    10003 13561752        2   450           NA            NA          150
## 2    10003  4942107        2   989           NA            NA           NA
## 3    10003   711635        2   240         1365          5460           50
## 4    10003  4510857        2   119          850          3200           60
## 5    10003  3799598        2   240           NA            NA           75
## 6    10003   568743        2   159         1500          6000           75
##    number_of_reviews review_scores_rating      city size_rank LatestPrice
## 1                 14                   94 New York         21     2167533
## 2                 37                  100 New York         21     2167533
## 3                 63                   95 New York         21     2167533
## 4                  2                   80 New York         21     2167533
## 5                144                   88 New York         21     2167533
## 6                137                   83 New York         21     2167533
```

d) Let's check the summary again and the result shows that all the variables have proper data type and consistent values except for missing data which will be imputed soon.

```
summary(mergedata)
```

```
##     zipcode          id              bedrooms     price
##  11215  :141   Min.    :   20853   Min.    :2  Min.    : 28.0
##  10003  :133   1st Qu.: 4218751    1st Qu.:2   1st Qu.:165.0
##  10025  :112   Median : 9410246   Median :2    Median :240.0
##  10036  :108   Mean    : 9218383   Mean    :2  Mean    :278.7
##  10011  :102    3rd Qu.:14476782    3rd Qu.:2   3rd Qu.:325.0
##  10014  : 95   Max.   :18508770   Max.   :2   Max.   :4700.0
##  (Other):547
##   weekly_price  monthly_price    cleaning_fee    number_of_reviews
##  Min.   : 310   Min.   : 1250   Min.   :  0.00   Min.   :   0.00
##  1st Qu.:1000   1st Qu.: 3200   1st Qu.: 60.00   1st Qu.:   1.00
##  Median :1390   Median : 4730   Median : 95.00   Median :   5.00
##  Mean   :1640   Mean   : 5630   Mean   : 94.11   Mean   : 17.43
##  3rd Qu.:2071   3rd Qu.: 7275   3rd Qu.:110.00   3rd Qu.: 20.00
##  Max.   :5950   Max.   :17100   Max.   :350.00   Max.   :306.00
##  NA's   :981    NA's   :1058    NA's    :236
##  review_scores_rating       city        size_rank       LatestPrice
##  Min.   : 20.00       New York:1238   Min.   :    1.0  Min.    : 364632
##  1st Qu.: 90.00                       1st Qu.:   15.0  1st Qu.:1336690
##  Median : 95.00                       Median :   71.0  Median :1904404
##  Mean   : 93.12                       Mean    : 485.9  Mean    :1844515
##  3rd Qu.:100.00                       3rd Qu.: 580.0   3rd Qu.:2220092
##  Max.   :100.00                       Max.    :4149.0  Max.   :3293975
##  NA's   :268
```

CREATING FUNCTION TO NORMALISE THE DATA POINTS AND SCALING THE VARIABLES OF 0-1

```
scaling_data <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}

mergedata["number_of_reviews"] <- lapply(mergedata["number_of_reviews"],
scaling_data)

summary(mergedata$number_of_reviews)

##     Min.  1st Qu.   Median      Mean  3rd Qu.    Max.
## 0.000000 0.003268 0.016340 0.056952 0.065360 1.000000

summary(mergedata)

##     zipcode           id              bedrooms     price
##   11215  :141  Min.    :   20853  Min.    :2  Min.    :  28.0
##   10003  :133  1st Qu.: 4218751    1st Qu.:2  1st Qu.: 165.0
##   10025  :112  Median : 9410246   Median :2   Median : 240.0
##   10036  :108  Mean    : 9218383  Mean    :2  Mean    : 278.7
##   10011  :102   3rd Qu.:14476782   3rd Qu.:2  3rd Qu.: 325.0
##   10014  : 95  Max.    :18508770  Max.    :2  Max.    :4700.0
##  (Other):547
 ##  weekly_price  monthly_price    cleaning_fee    number_of_reviews
 ## Min.    :  310  Min.    : 1250   Min.    :   0.00  Min.    :0.000000
 ##  1st Qu.:1000  1st Qu.: 3200   1st Qu.: 60.00    1st Qu.:0.003268
 ## Median :1390   Median : 4730   Median : 95.00   Median :0.016340
 ## Mean    :1640   Mean    : 5630   Mean    : 94.11  Mean    :0.056952
 ##  3rd Qu.:2071  3rd Qu.: 7275    3rd Qu.:110.00   3rd Qu.:0.065360
 ## Max.    :5950   Max.    :17100   Max.    :350.00  Max.    :1.000000
 ## NA's    :981    NA's    :1058    NA's    :236
 ## review_scores_rating       city        size_rank        LatestPrice
 ## Min.    :  20.00       New York:1238   Min.    :   1.0  Min.    : 364632
 ## 1st Qu.:  90.00                        1st Qu.:   15.0   1st Qu.:1336690
 ## Median :  95.00                        Median :   71.0  Median :1904404
 ## Mean    :  93.12                       Mean    : 485.9  Mean    :1844515
 ##  3rd Qu.:100.00                        3rd Qu.: 580.0    3rd Qu.:2220092
 ## Max.    :100.00                        Max.    :4149.0  Max.    :3293975
##  NA's    :268
```

a) Using the ggplot2 library package to plot variables contain NA's values.

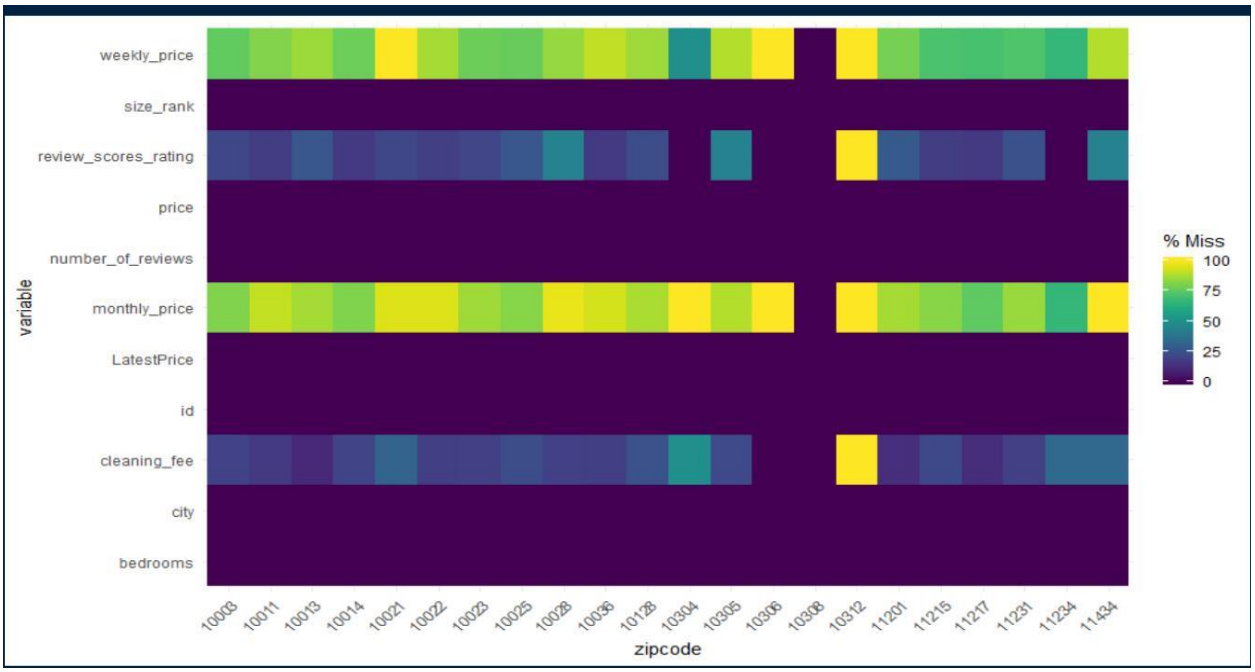
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.4.4

library(naniar)
```

```
## Warning: package 'naniar' was built under R version 3.4.4
```

```
gg_miss_fct(x = mergedata, fct = zipcode)
```



b) Removing the Na values - The NAs are removed below by imputing them using mice package and CART algorithm as follows

```
library(mice)
```

```
## Warning: package 'mice' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
impute_data <- subset(mergedata,select = -c(id,city))
impute_data <- mice(impute_data,m=5, method='cart',printFlag=FALSE)
```

```
## Warning: Number of logged events: 106
```

```
final_data<- complete(impute_data)
```

c) There are no more missing data values in the final data

```
summary(final_data)
```

```
##     zipcode       bedrooms     price          weekly_price  monthly_price
##   11215  :141   Min.   :2   Min.   :  28.0   Min.   : 310   Min.   : 1250
```

```
##  10003  :133  1st Qu.:2   1st Qu.: 165.0   1st Qu.:1012   1st Qu.: 3250
##  10025  :112  Median :2   Median : 240.0   Median :1412   Median : 4900
##  10036  :108  Mean   :2   Mean   :  278.7  Mean   :1727   Mean   :  5660
##  10011  :102  3rd Qu.:2   3rd Qu.:  325.0  3rd Qu.:2118   3rd Qu.:  7000
## 10014  : 95   Max.   :2   Max.   :4700.0    Max.   :5950   Max.   :17100
## (Other):547
##  cleaning_fee    number_of_reviews  review_scores_rating   size_rank
## Min.   :   0.00  Min.   :0.000000   Min.   : 20.00       Min.    :   1.0
## 1st Qu.:  60.00   1st Qu.:0.003268  1st Qu.: 90.00       1st Qu.:   15.0
## Median :  90.00  Median :0.016340   Median : 96.00       Median :   71.0
## Mean   :  93.97  Mean    :0.056952  Mean    : 93.28      Mean    : 485.9
##  3rd Qu.:110.00   3rd Qu.:0.065360   3rd Qu.:100.00      3rd Qu.: 580.0
## Max.   :350.00   Max.    :1.000000  Max.    :100.00      Max.    :4149.0
##
##  LatestPrice
## Min.   : 364632
## 1st Qu.:1336690
## Median :1904404
## Mean   :1844515
## 3rd Qu.:2220092
## Max.   :3293975
##
```

d) Now we have a complete dataset and lets have a look on the summary.

```
a <- subset(mergedata, select =c(id,city))
complete_data <- cbind(final_data,a)
summary(complete_data)
```

```
##    zipcode       bedrooms      price         weekly_price  monthly_price
## 11215  :141   Min.    :2   Min.    :  28.0  Min.    : 310  Min.    : 1250
## 10003  :133    1st Qu.:2    1st Qu.: 165.0  1st Qu.:1012   1st Qu.: 3250
## 10025  :112   Median :2    Median : 240.0  Median :1412   Median : 4900
## 10036  :108   Mean   :2   Mean    : 278.7  Mean    :1727  Mean    : 5660
## 10011  :102    3rd Qu.:2    3rd Qu.: 325.0   3rd Qu.:2118  3rd Qu.: 7000
## 10014  : 95   Max.   :2   Max.   :4700.0   Max.    :5950   Max.   :17100
## (Other):547
##  cleaning_fee    number_of_reviews  review_scores_rating   size_rank
## Min.   :  0.00   Min.   :0.000000   Min.   : 20.00       Min.    :   1.0
## 1st Qu.: 60.00    1st Qu.:0.003268  1st Qu.: 90.00       1st Qu.:   15.0
## Median : 90.00   Median :0.016340   Median : 96.00       Median :   71.0
## Mean   : 93.97   Mean    :0.056952   Mean  : 93.28       Mean    : 485.9
## 3rd Qu.:110.00    3rd Qu.:0.065360  3rd Qu.:100.00       3rd Qu.: 580.0
## Max.   :350.00   Max.    :1.000000   Max.  :100.00       Max.    :4149.0
##
##  LatestPrice            id                 city
## Min.   : 364632   Min.    :   20853  New York:1238
## 1st Qu.:1336690   1st Qu.: 4218751
## Median :1904404   Median : 9410246
## Mean   :1844515   Mean    : 9218383
## 3rd Qu.:2220092    3rd Qu.:14476782
```

```
##  Max.   :3293975   Max.   :18508770
##
```

```
head(complete_data)
```

```
##   zipcode bedrooms price weekly_price monthly_price cleaning_fee
 ## 1   10003        2   450         1900          7000          150
 ## 2   10003        2   989         3500         17100          350
 ## 3   10003        2   240         1365          5460           50
 ## 4   10003        2   119          850          3200           60
 ## 5   10003        2   240         1895          8200           75
 ## 6   10003        2   159         1500          6000           75
##    number_of_reviews review_scores_rating size_rank LatestPrice       id
## 1        0.045751634                   94        21     2167533 13561752
## 2        0.120915033                  100        21     2167533  4942107
## 3        0.205882353                   95        21     2167533   711635
## 4        0.006535948                   80        21     2167533  4510857
## 5        0.470588235                   88        21     2167533  3799598
## 6        0.447712418                   83        21     2167533   568743
##       city
## 1 New York
## 2 New York
## 3 New York
## 4 New York
## 5 New York
## 6 New York
```

```
sum(sapply(complete_data, function(x) { sum(is.na(x)) }))
```

```
## [1] 0
```

```
gg_miss_fct(x = complete_data, fct = zipcode)
```

Now we can see above data is totally clean and there is no missing values. Lets step into further data analysis.

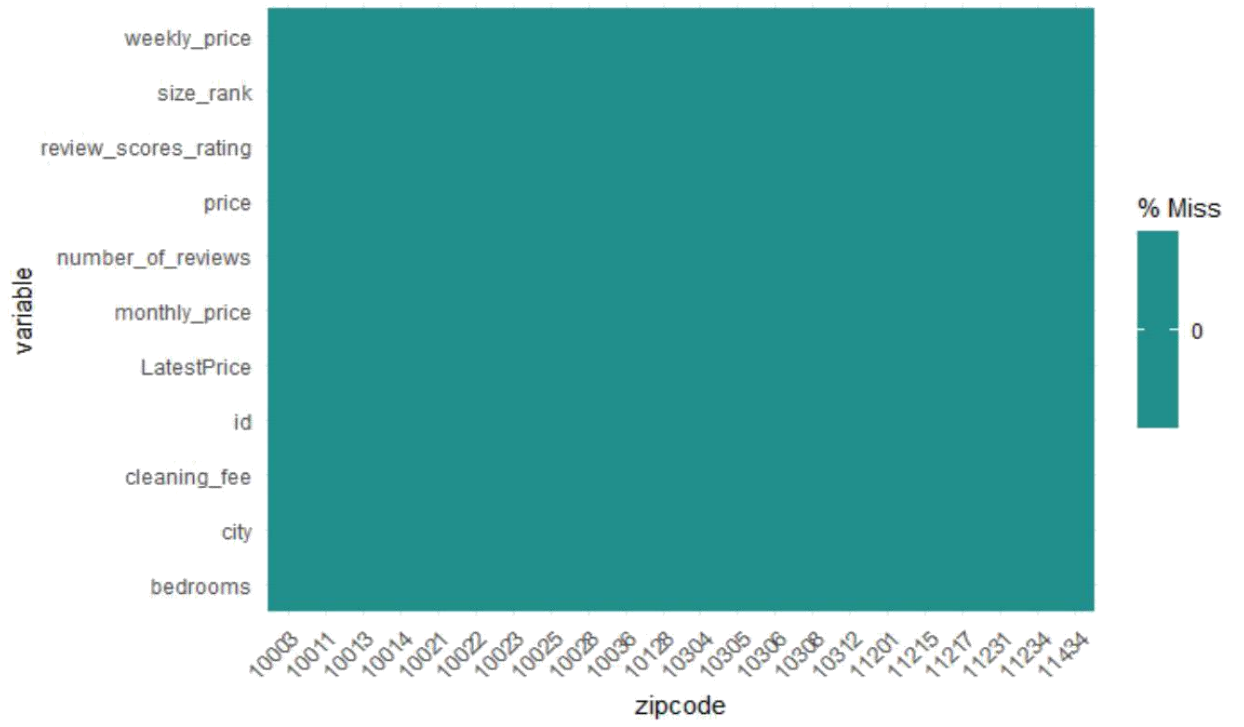
Exploratory Data Analysis and Conclusion: -

Assumption: All properties and all square feet within each locale can be assumed to be homogeneous.

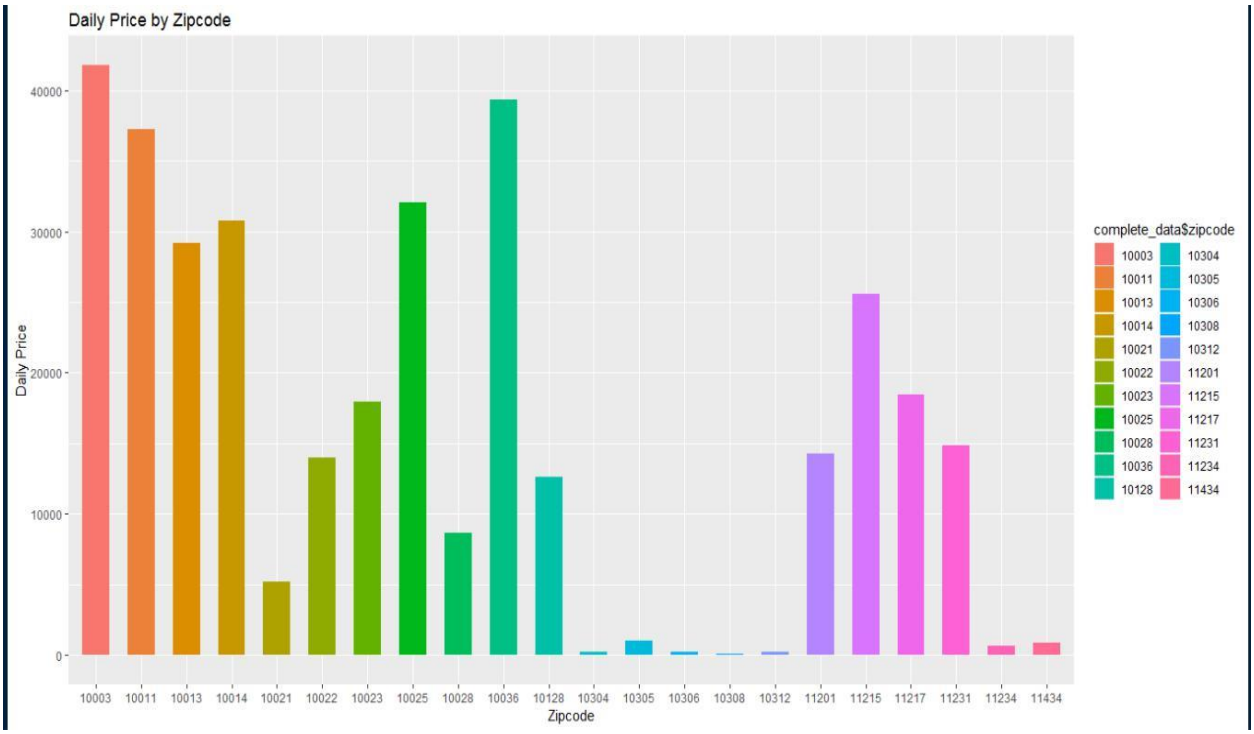
Daily price by Zipcode

a) The graph depicts the daily price of the properties per zipcode.

b) The zip codes (10003,10011,10013,10014,10025,10036,11215) has the highest daily market price by Zip codes.




```
library(ggplot2)
ggplot(complete_data, aes(x=complete_data$zipcode,
y=complete_data$price, fill=complete_data$zipcode )) +
  geom_bar(stat="identity", width = 0.6)+labs(title = "Daily Price
by Zipcode",
x="Zipcode",y="Daily Price")
```

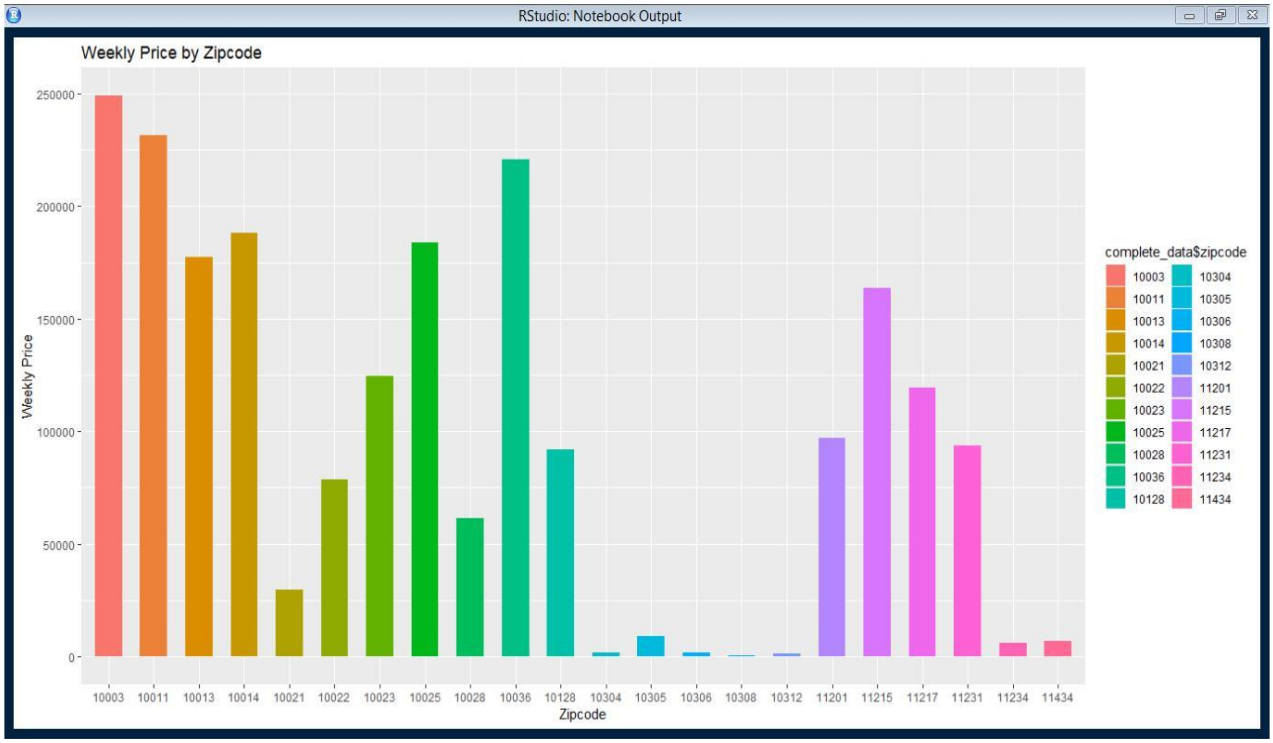


Weekly Price by Zipcode

a) The graph depicts the weekly price of the properties per zip code.

b) The zip codes (10003,10011,10013,10014,10025,10036,11215) has the highest weekly market price by Zip codes.

```
library(ggplot2)
ggplot(complete_data, aes(x=complete_data$zipcode,
y=complete_data$weekly_price, fill=complete_data$zipcode )) +
  geom_bar(stat="identity", width = 0.6)+labs(title = "Weekly Price by
Zipcode",
x="Zipcode",y="Weekly Price")
```

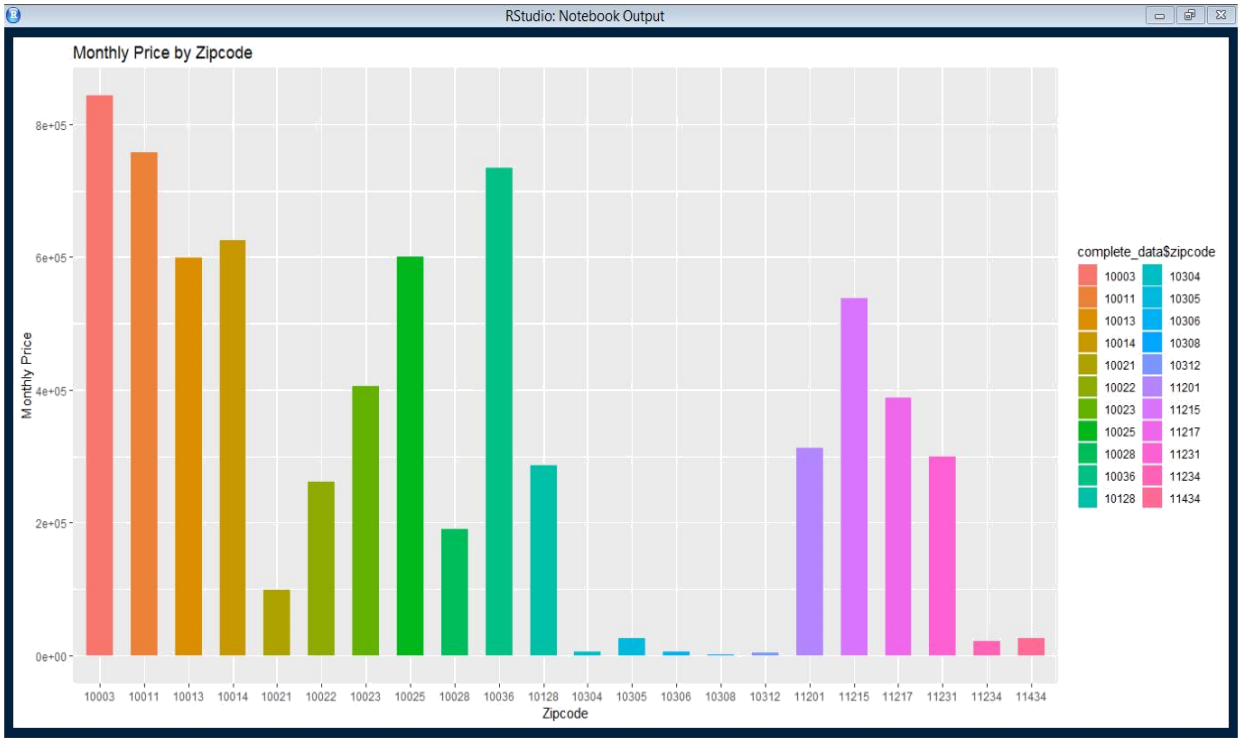


Monthly Price by Zipcode

a) The graph depicts the monthly price of the properties per zip code.

b) The zip codes (10003,10011,10013,10014,10025,10036,11215) has the highest monthly market price by Zip codes.

```
library(ggplot2)
ggplot(complete_data, aes(x=complete_data$zipcode,
y=complete_data$monthly_price, fill=complete_data$zipcode )) +
  geom_bar(stat="identity", width = 0.6)+labs(title = "Monthly Price by
Zipcode",
x="Zipcode",y="Monthly Price")
```

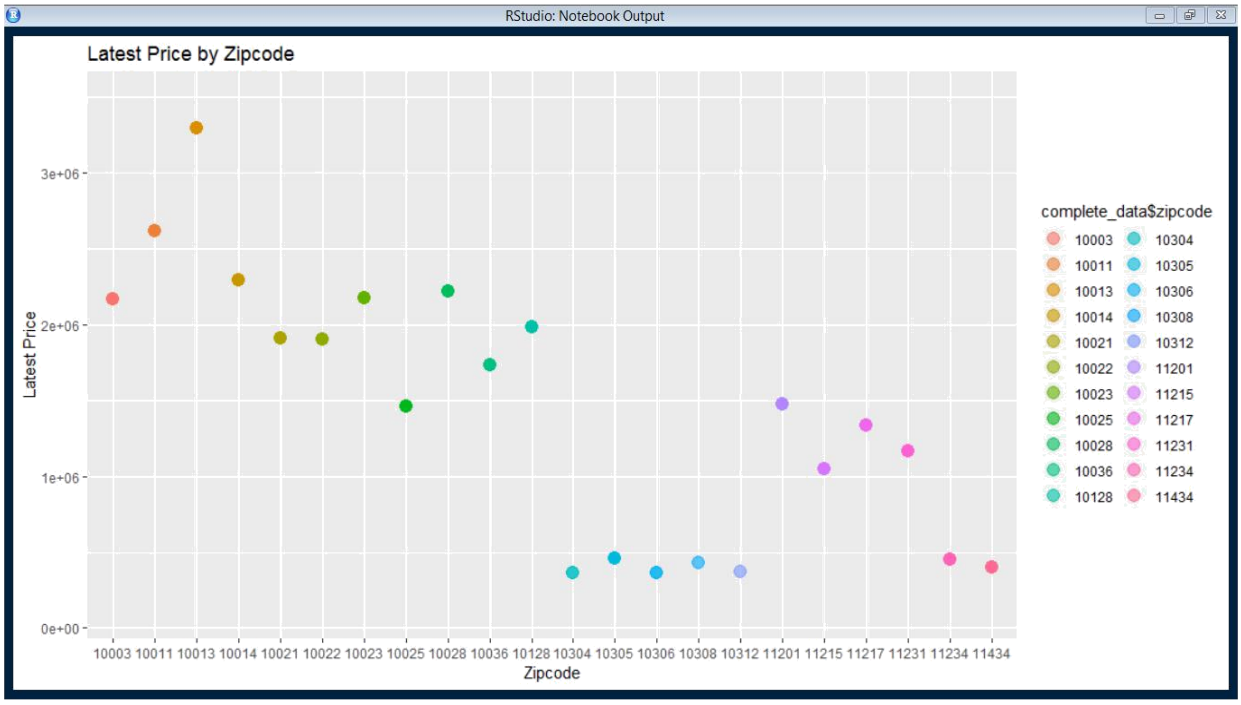


Latest Price of the Properties by Zip codes

a) The graph depicts the Latest price of the properties per zip code.

b) The zip codes (10011,10013,10014,10003) has the highest Latest market price by Zip codes.

```
qplot(data = complete_data, x = complete_data$zipcode, y =
complete_data$LatestPrice, color = complete_data$zipcode, size = I(4), alpha
= I(0.6)) + scale_y_continuous(name="Latest Price",
limits=c(100000, 3500000))+labs(title = "Latest Price by Zipcode",
x="Zipcode",y="Latest Price")
```

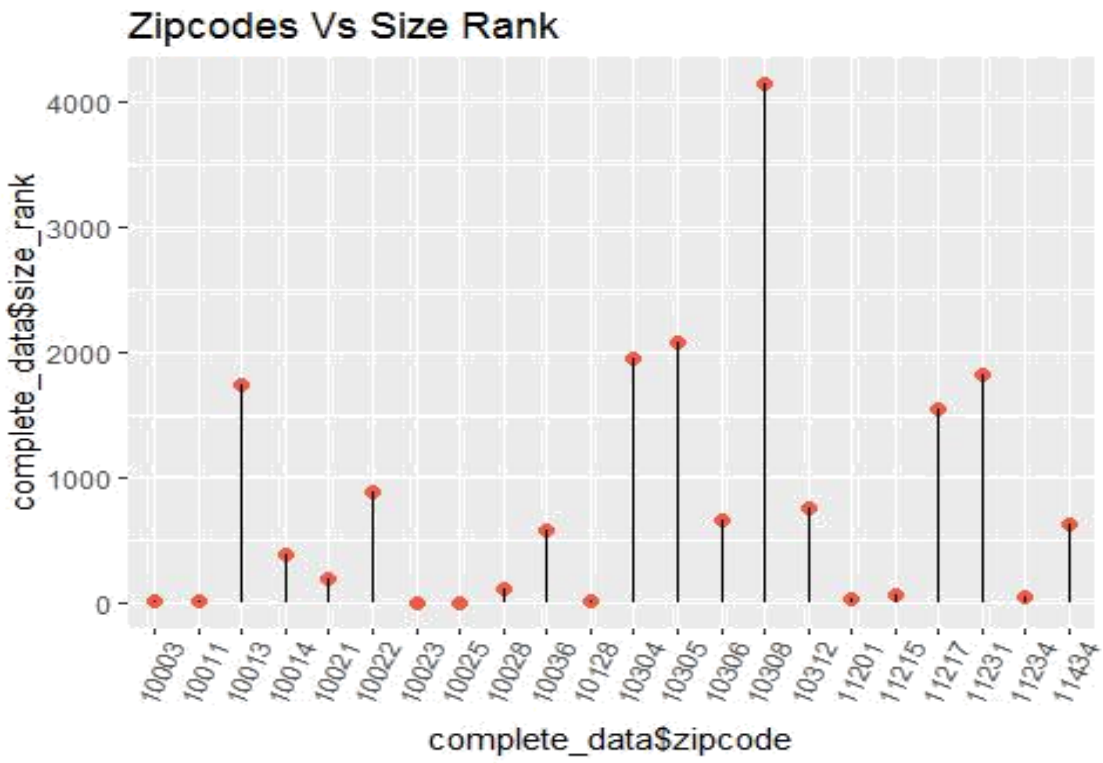


Size Rank of the Properties by Zip codes

a) The graph depicts the Size Rank of the properties per zip code.

b) Higher the size rank, lower the population of the area.

```
ggplot(complete_data, aes(x=complete_data$zipcode,
y=complete_data$size_rank)) +
  geom_point(size=2, col="tomato2") +
  geom_segment(aes(x=complete_data$zipcode,
                   xend=complete_data$zipcode,
                   y=0,
                   yend=complete_data$size_rank)) +
  labs(title="Zip codes Vs Size Rank") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```



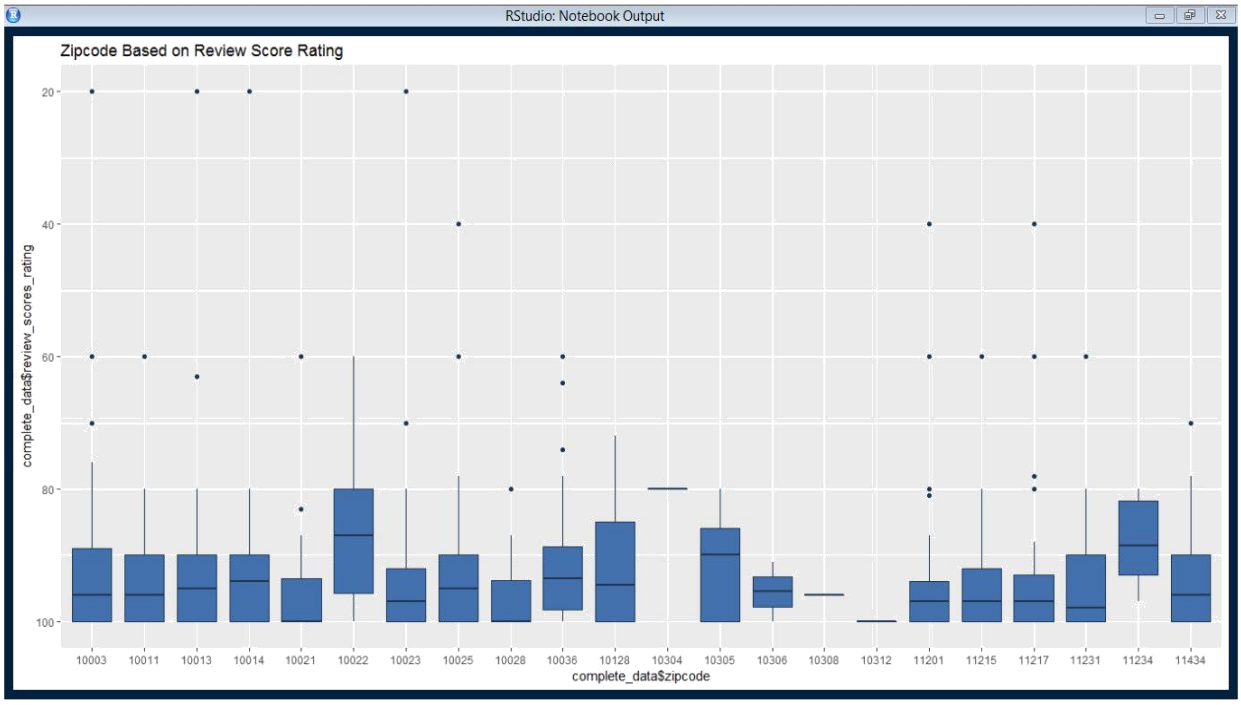
Review Score Rating of the Properties by Zip codes

a) The graph depicts the Review Score Rating of the properties per zipcode.

b) Higher the reviews better the property would be.

```
fill <- "#4271AE"
line <- "#1F3552"

p10 <- ggplot(complete_data, aes(x = complete_data$zipcode, y
= complete_data$review_scores_rating)) +
        geom_boxplot(fill = fill, colour = line) +
        ggtitle("Zipcode Based on Review Score Rating")
p10 + scale_y_reverse()
```



Important information Obtained from the above graphs: -

a) There are 22 Zip codes in New York city having rental properties listed on Airbnb

b) With the above graph we can see the daily, weekly and monthly price of the property as of now and which zip code has the highest market price respectively.

c) Now we use calculative formula to get the daily, weekly and monthly revenue of the properties and gathered the zip codes to see which Zipcode is more profitable for daily rental, weekly rental and monthly rental by revenue.

d) Assuming that Occupancy rate to be 0.75 as given in the case study, Revenue of the property will be calculated by below formula: -

Revenue <- Occupancyrate*Daily/weekly/monthly_price_of_the_property*365/48/12+(Cleaning_fee *occupancy_rate)*12 <- as cleaning fee will be paid by customer in the form of service tax based upon the occupancy rate.

REVENUE- Revenue is the amount of money that a company actually receives during a specific period, including discounts and deductions for returned merchandise. Revenue is calculated by multiplying the price at which goods or services are sold by the number of units or amount sold.

Calculating Revenue Based of the daily rental bookings:-

```
occupancyrate_1<- 0.75
Data1 <- complete_data
Data1$Revenue_daily <-occupancyrate_1*Data1$price*365
+(Data1$cleaning_fee*occupancyrate_1)*12 head(Data1)


##   zipcode bedrooms price weekly_price monthly_price cleaning_fee
 ## 1   10003        2   450         1900          7000          150
 ## 2   10003        2   989         3500         17100          350
 ## 3   10003        2   240         1365          5460           50
 ## 4   10003        2   119          850          3200           60
 ## 5   10003        2   240         1895          8200           75
 ## 6   10003        2   159         1500          6000           75
##    number_of_reviews review_scores_rating size_rank LatestPrice       id
## 1        0.045751634                   94        21     2167533 13561752
## 2        0.120915033                  100        21     2167533  4942107
## 3        0.205882353                   95        21     2167533   711635
## 4        0.006535948                   80        21     2167533  4510857
## 5        0.470588235                   88        21     2167533  3799598
## 6        0.447712418                   83        21     2167533   568743
##       city Revenue_daily
## 1 New York      124537.50
## 2 New York      273888.75
## 3 New York       66150.00
## 4 New York       33116.25
## 5 New York       66375.00
## 6 New York       44201.25
```

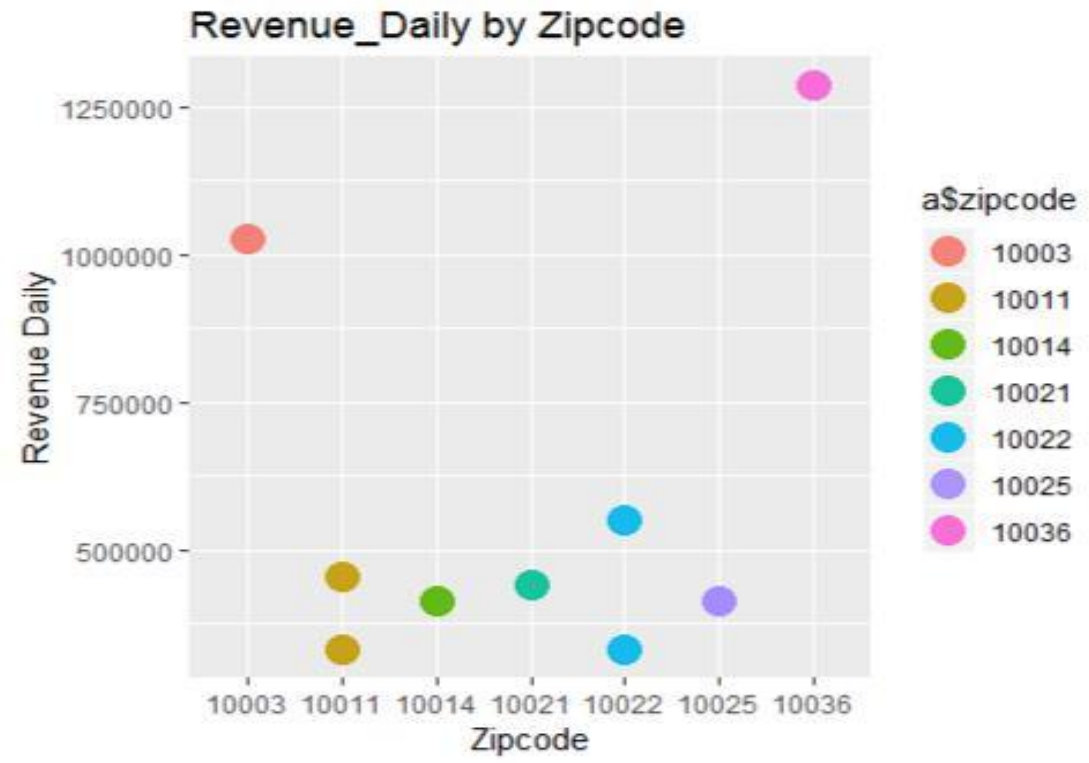
Fetching relevant columns from the data frame

```
Column1 <- c("zipcode","id","city","Revenue_daily")
Daily <- Data1[,Column1]
sort_daily<- Daily[order(-
Daily$Revenue_daily),] head(sort_daily)

##      zipcode       id     city Revenue_daily
## 720    10036 12376888 New York      1287255.0
## 37     10003  2281142 New York      1028362.5
## 464    10022  2266010 New York       549300.0
## 204    10011  2307885 New York       453937.5
## 423    10021  2284454 New York       439800.0
## 569    10025  5649623 New York       413325.0
```

Fetching Top 10 Zipcodes in descending order giving best revenue by daily rental.

```
a <- head(sort_daily,10)
qplot(data =a, x = a$zipcode, y = a$Revenue_daily, color = a$zipcode, size =
I(5), alpha = I(0.9))+ labs(title = "Revenue_Daily by Zipcode",
x="Zipcode",y="Revenue Daily")
```




```
a
```

```
##      zipcode       id     city Revenue_daily
## 720    10036 12376888 New York      1287255.0
## 37     10003  2281142 New York      1028362.5
## 464    10022  2266010 New York       549300.0
## 204    10011  2307885 New York       453937.5
## 423    10021  2284454 New York       439800.0
## 569    10025  5649623 New York       413325.0
## 319    10014  2243984 New York       412425.0
## 631    10025  2243769 New York       411750.0
## 179    10011 17877401 New York       331191.0
## 448    10022 16173564 New York       330300.0
```

Conclusion: -

The Zipcodes with the highest Revenue by daily rentals are 10036, 10003, 10022, 10011 and 10021.

The zip codes with the lowest Revenue by daily rentals are 10014, 10025, 10025 and 10022.

Calculating the Revenue Based of the Weekly rental bookings: -

```
occupancyrate_2 <- 0.75
Data1$Revenue_weekly <-
occupancyrate_2*Data1$weekly_price*4*12+(Data1$cleaning_fee*occupancyrate_2)*
12
head(Data1)
```

```
##   zipcode bedrooms price weekly_price monthly_price cleaning_fee
 ## 1   10003        2   450         1900          7000          150
 ## 2   10003        2   989         3500         17100          350
 ## 3   10003        2   240         1365          5460           50
 ## 4   10003        2   119          850          3200           60
 ## 5   10003        2   240         1895          8200           75
 ## 6   10003        2   159         1500          6000           75
##   number_of_reviews review_scores_rating size_rank LatestPrice        id
## 1        0.045751634                  94       21     2167533  13561752
## 2        0.120915033                 100       21     2167533   4942107
## 3        0.205882353                  95       21     2167533    711635
## 4        0.006535948                  80       21     2167533   4510857
## 5        0.470588235                  88       21     2167533   3799598
## 6        0.447712418                  83       21     2167533    568743
##       city Revenue_daily Revenue_weekly
## 1 New York      124537.50          69750
## 2 New York      273888.75         129150
## 3 New York       66150.00          49590
## 4 New York       33116.25          31140
## 5 New York       66375.00          68895
## 6 New York       44201.25          54675
```

Fetching relevant columns from the data frame

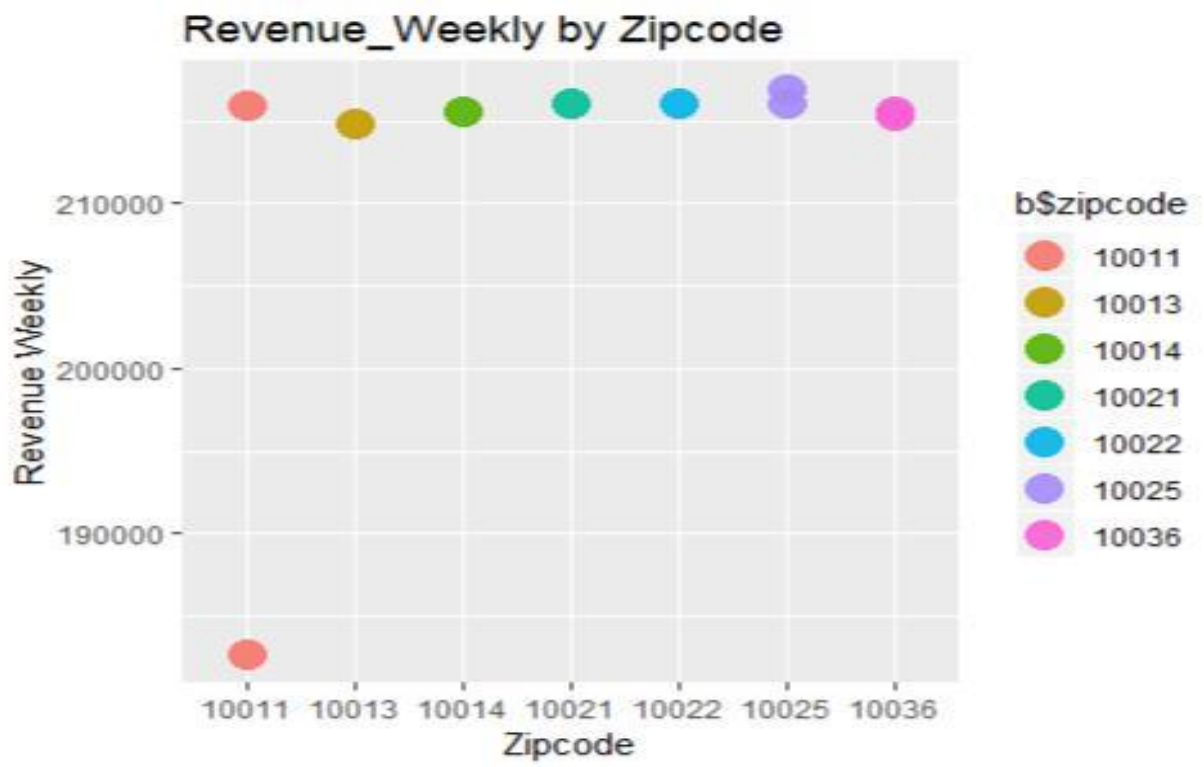
```
Column2 <- c("zipcode","id","city","Revenue_weekly")
weekly <- Data1[,Column2]
sort_weekly<- weekly[order(-
weekly$Revenue_weekly),] head(sort_weekly)
```

```
##      zipcode      id     city Revenue_weekly
## 569    10025  5649623 New York          216900
## 423    10021  2284454 New York          216000
## 464    10022  2266010 New York          216000
## 557    10025  6138423 New York          216000
```

```
##  203   10011 15841618 New York          215910
##  385   10014 14606167 New York          215550
```

Fetching Top 10 Zipcodes in descending order giving best revenue by weekly rental.

```
b <- head(sort_weekly,10)
qplot(data =b, x = b$zipcode, y = b$Revenue_weekly, color = b$zipcode, size =
I(5), alpha = I(0.9))+ labs(title = "Revenue_Weekly by Zipcode",
x="Zipcode",y="Revenue Weekly")
```




```
b
```

```
##      zipcode       id      city Revenue_weekly
## 569    10025  5649623 New York          216900
## 423    10021  2284454 New York          216000
## 464    10022  2266010 New York          216000
## 557    10025  6138423 New York          216000
## 203    10011 15841618 New York          215910
## 385    10014 14606167 New York          215550
## 752    10036  2299633 New York          215550
## 724    10036  1171581 New York          215325
## 247    10013  9753240 New York          214875
## 174    10011  4678742 New York          182691
```

Conclusion: -

The Zipcodes with the highest Revenue by weekly rentals are 10014, 10128, 10011, 10036 and 10011.

The Zipcodes with the lowest Revenue by weekly rentals are 10003, 10011, 10011,10028 and 10011.

Calculating the Revenue Based of the Monthly rental bookings:-

```
occupancyrate_3 <- 0.75
Data1$Revenue_monthly <-
occupancyrate_3*Data1$monthly_price*12+(Data1$cleaning_fee*occupancyrate_3)*1
2
head(Data1)
```

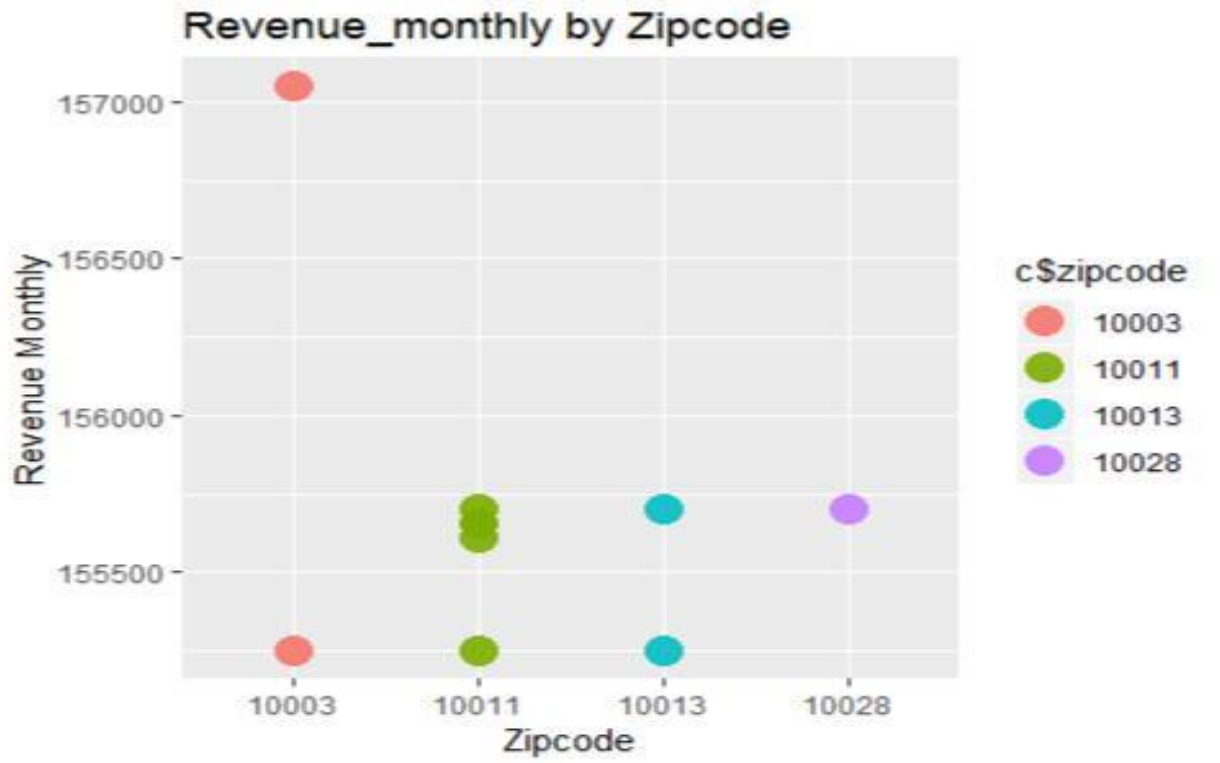
```
##   zipcode bedrooms price weekly_price monthly_price cleaning_fee
 ## 1   10003        2   450         1900          7000          150
 ## 2   10003        2   989         3500         17100          350
 ## 3   10003        2   240         1365          5460           50
 ## 4   10003        2   119          850          3200           60
 ## 5   10003        2   240         1895          8200           75
 ## 6   10003        2   159         1500          6000           75
##    number_of_reviews review_scores_rating size_rank LatestPrice       id
## 1        0.045751634                   94        21     2167533 13561752
## 2        0.120915033                  100        21     2167533  4942107
## 3        0.205882353                   95        21     2167533   711635
## 4        0.006535948                   80        21     2167533  4510857
## 5        0.470588235                   88        21     2167533  3799598
## 6        0.447712418                   83        21     2167533   568743
##       city Revenue_daily Revenue_weekly Revenue_monthly
## 1 New York      124537.50          69750           64350
## 2 New York      273888.75         129150          157050
## 3 New York       66150.00          49590           49590
## 4 New York       33116.25          31140           29340
## 5 New York       66375.00          68895           74475
## 6 New York       44201.25          54675           54675
```

Fetching relevant columns from the data frame

```
Column3 <- c("zipcode","id","city","Revenue_monthly")
monthly <- Data1[,Column3]
sort_monthly<- monthly[order(-monthly$Revenue_monthly),]
```

Fetching Top 10 Zipcodes in descending order giving best revenue by monthly rental.

```
c <- head(sort_monthly,10)
qplot(data =c, x = c$zipcode, y = c$Revenue_monthly, color = c$zipcode, size
= I(5), alpha = I(0.9)) + labs(title = "Revenue_monthly by
Zipcode", x="Zipcode",y="Revenue Monthly")
```




```
c
```

```
##      zipcode       id      city Revenue_monthly
## 2      10003  4942107 New York           157050
## 171    10011 15268598 New York           155700
## 294    10013 14993030 New York           155700
## 653    10028  8498323 New York           155700
## 153    10011  7790526 New York           155655
## 202    10011  1746726 New York           155655
## 190    10011 17108224 New York           155610
## 88     10003 12267522 New York           155250
## 187    10011 17152737 New York           155250
## 296    10013  3530517 New York           155250
```

Conclusion: -

The Zipcodes with the highest Revenue by monthly rentals are 10036, 10011, 10013, 10014 and 10011.

The Zipcodes with the lowest Revenue by monthly rentals are 10013,10028 and 10014.

SUMMARY OF THE ABOVE DATA ANALYSIS:-

Objective of the Case Study: - The purpose of this case study was to analyze the data given by Zillow and Airbnb to identify the best zip codes for the investment in the two bedrooms properties in the City New York. The steps followed to analyze the data are, Loading the data of Zillow & Airbnb. Then Cleaning the data was the second step in which we chose relevant columns like Cityname = New York and bedrooms should be equal to 2. Also removed NA's values and $ sign to clean the data further and analyze which Zipcodes are best for investment.

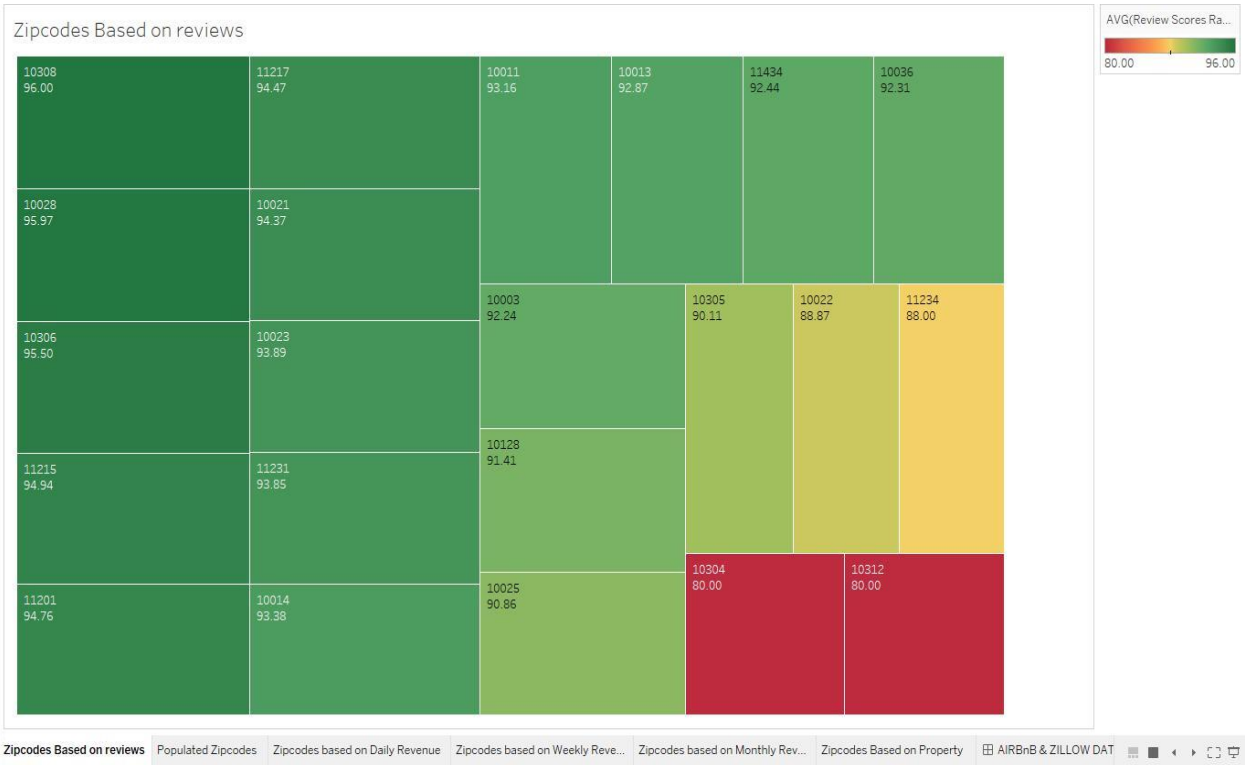
Taking into consideration that some people come to visit New York for 2-3 days, some come for business trip and book an apartment for 1 day because they find it less expensive than hotels, some came for a week and book an apartment for whole week and there are many universities in New York, So there must be students and many international students as well who book an apartment for months as many people can't afford hostel rent.

So, I have segregated the properties into 3 scenarios. For which Zipcodes and ID an agent should invest and put his apartment for daily rental, for weekly rental and for monthly rental, so he can gain more profit out of it.

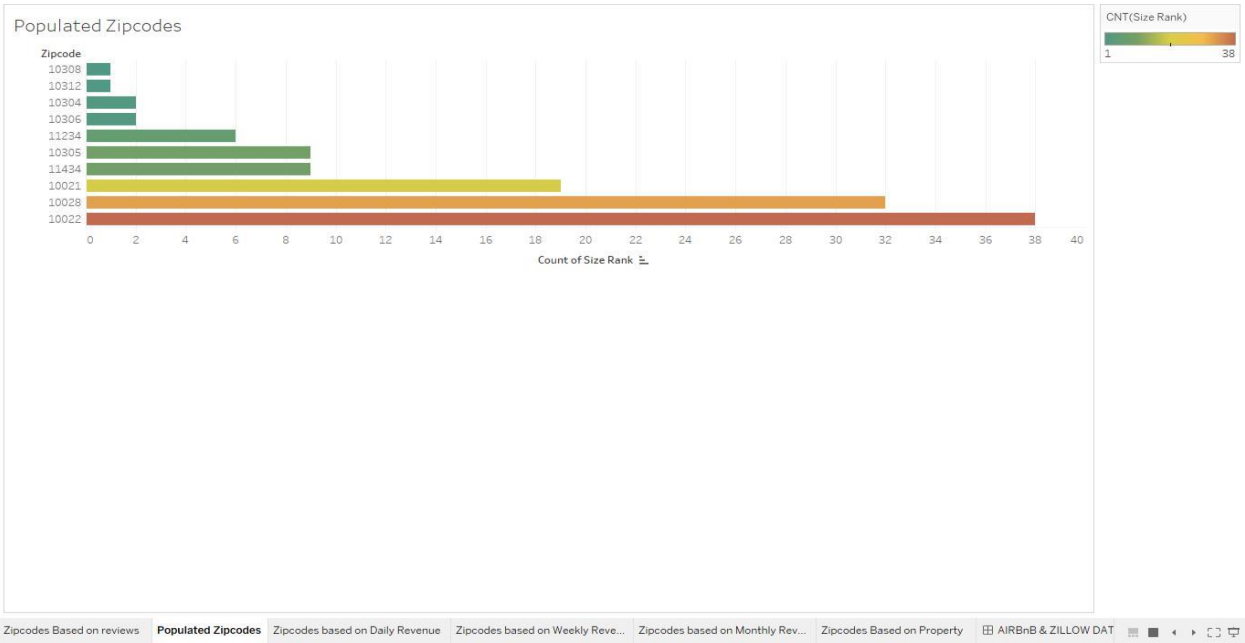
Finally, we can summaries the above analysis and found that the zip codes with the highest **Revenue from daily rental business are 10011, 10003, 10036, 10022, 11201 and 10025. The zip codes with the highest revenue from Weekly and Monthly business are 10011, 10036 and 10013. Hence the zip codes with the highest revenue in any scenario keeping into account the consideration we made the zip codes with the maximum revenue are 10011,10036,10013,10022,11201 and 10025.**

DATA VISUALIZATION GRAPHS USING TABLEAU

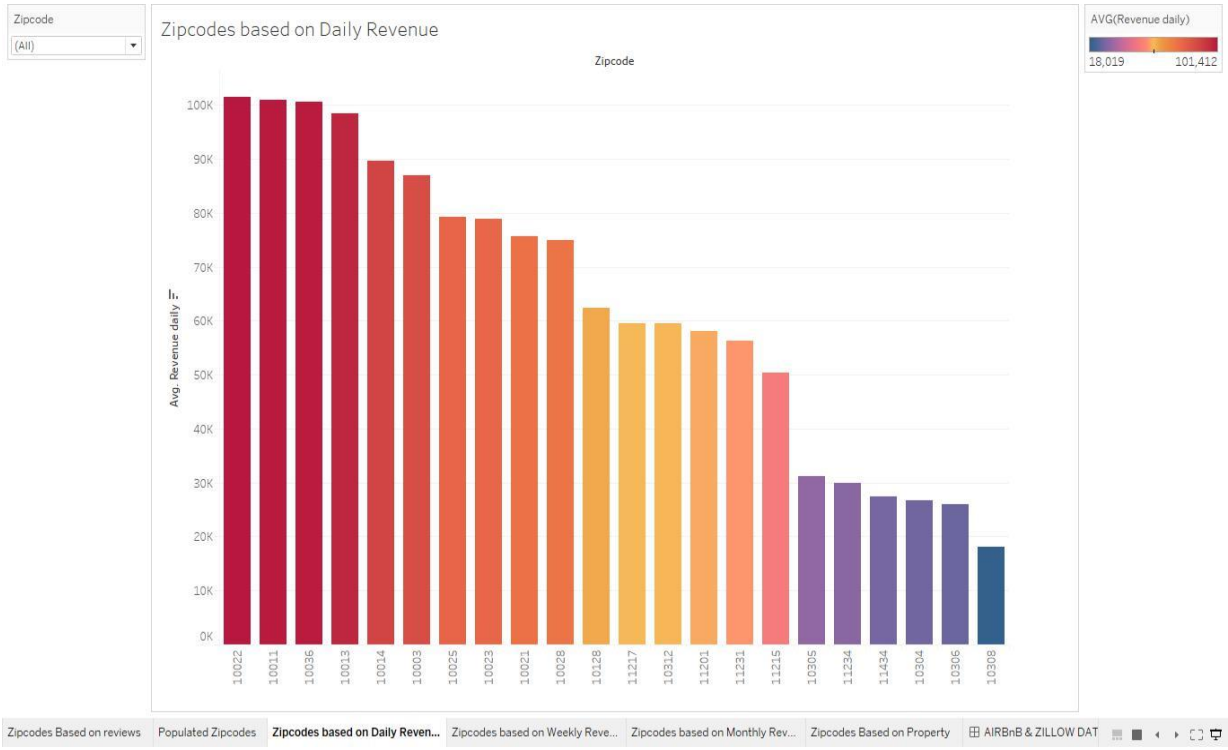
1) Reviews score ratings of the properties by Zip codes.



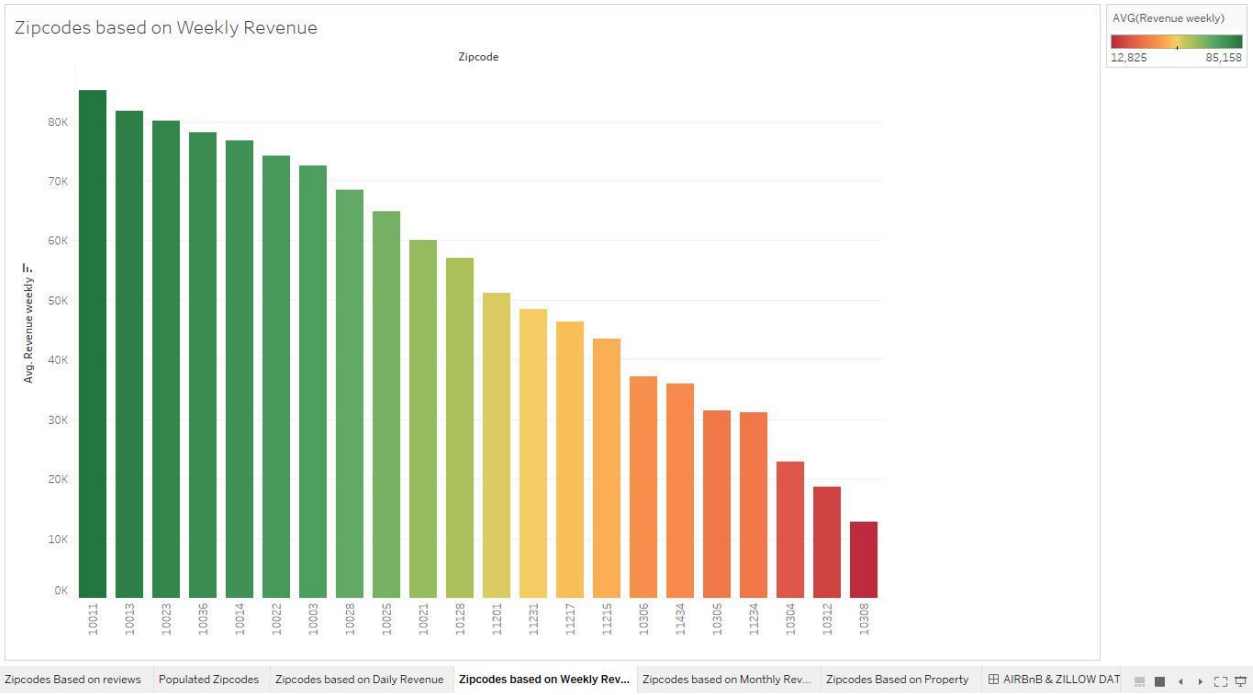
2) Size Rank(Top 10 by ascending order) of the properties by zip codes – Lower the size rank, Higher the population of the area.



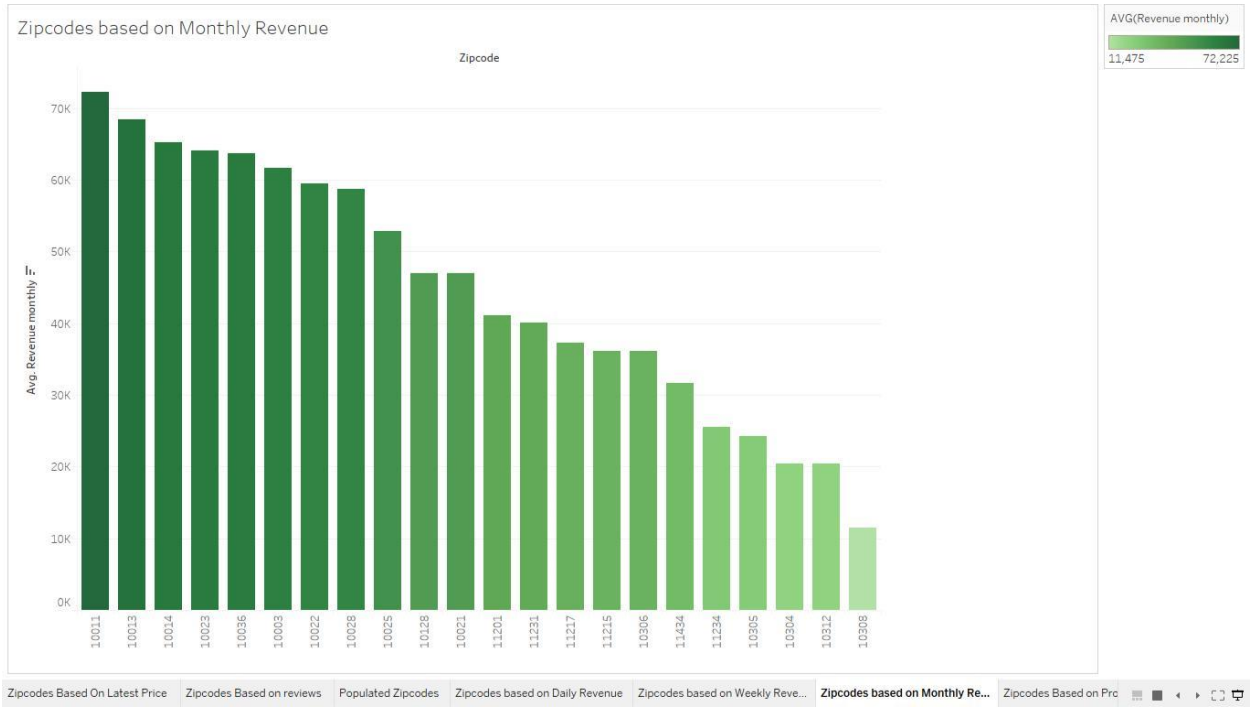
3) Zip codes based on Daily Revenue.



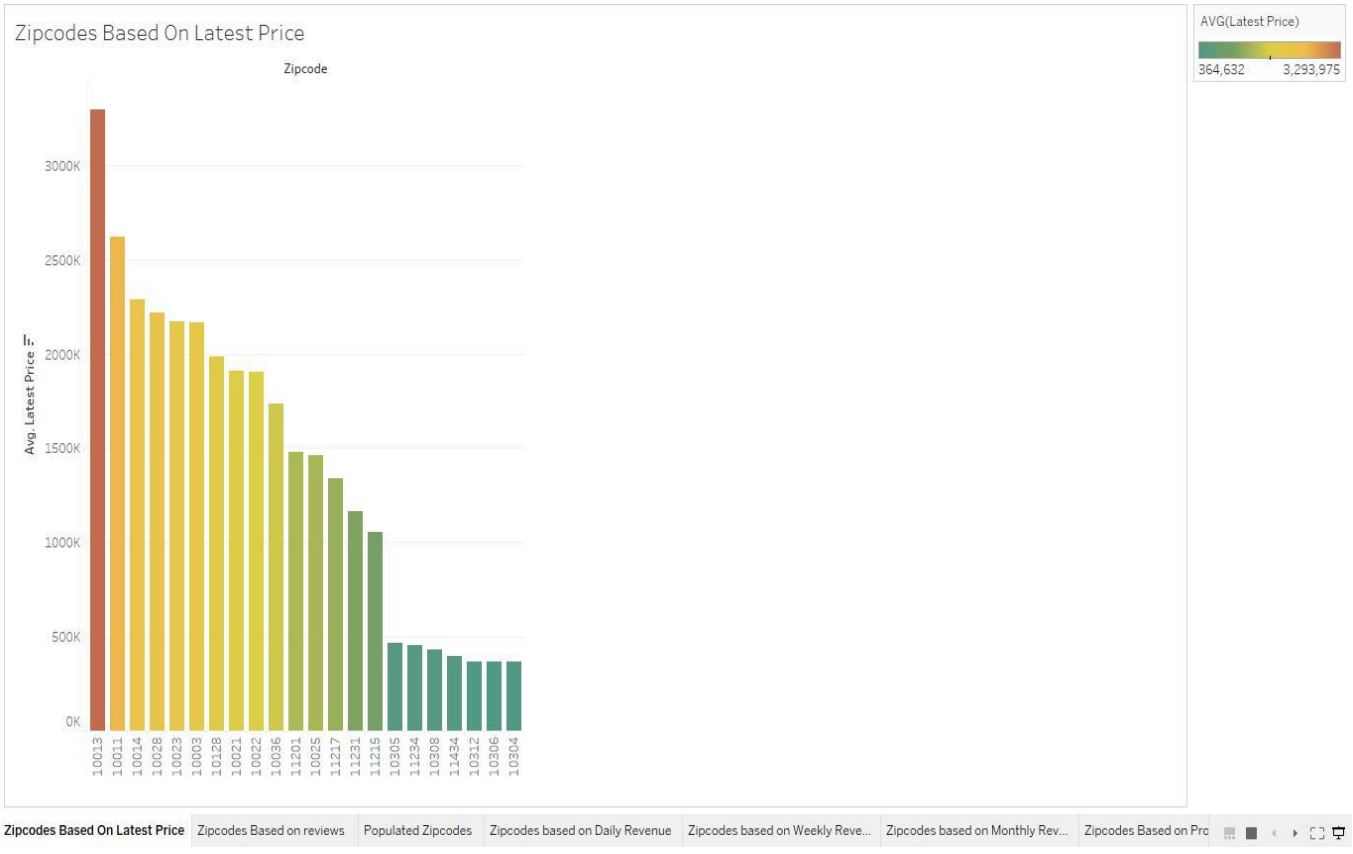
4) Zip codes based on weekly revenue.



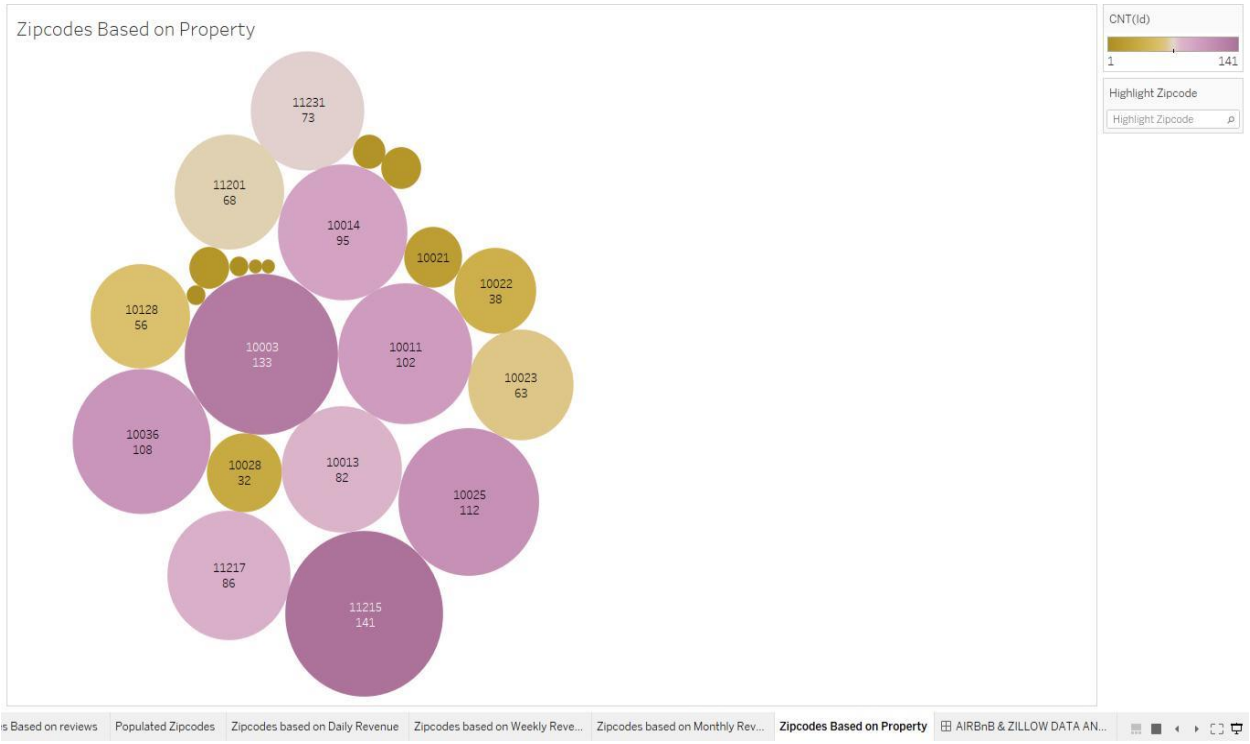
5) Zip codes based on Monthly Revenue.



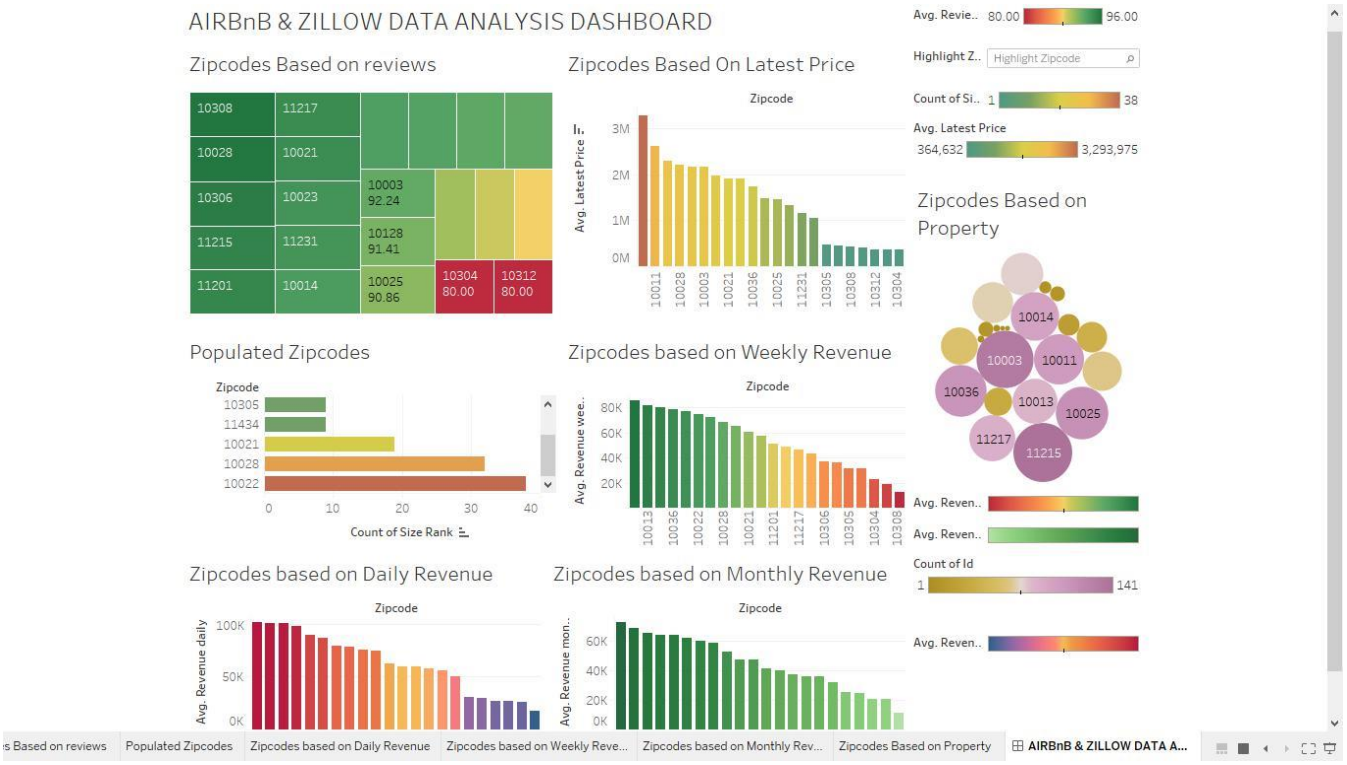
6) Latest Price of the Property by Zip codes.



7) Zip codes based on properties.

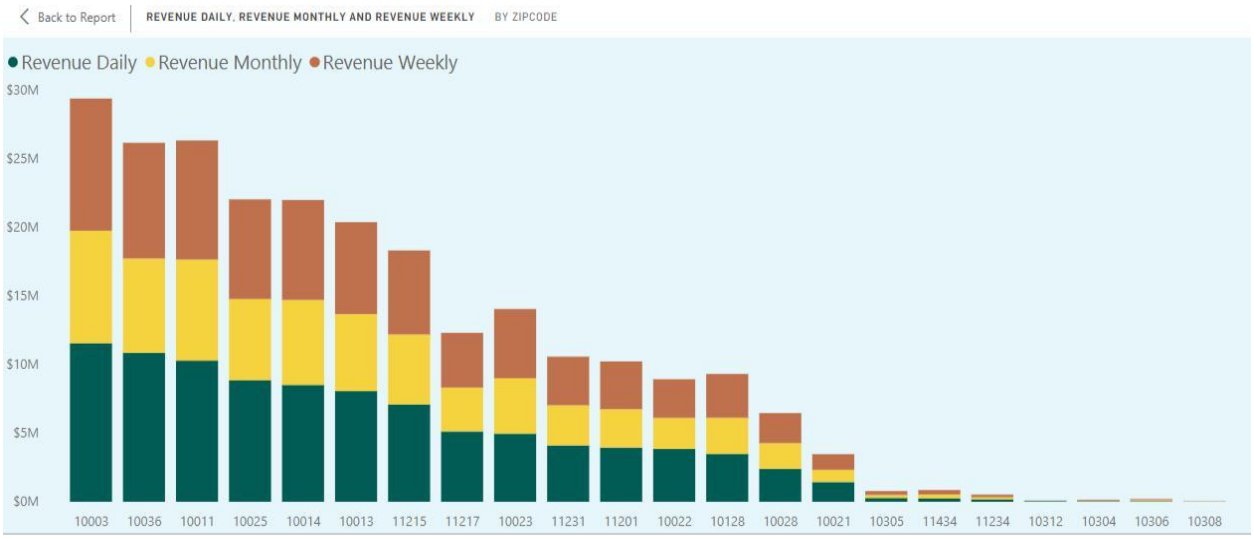


8) Dashboard of Airbnb and Zillow Data Analysis

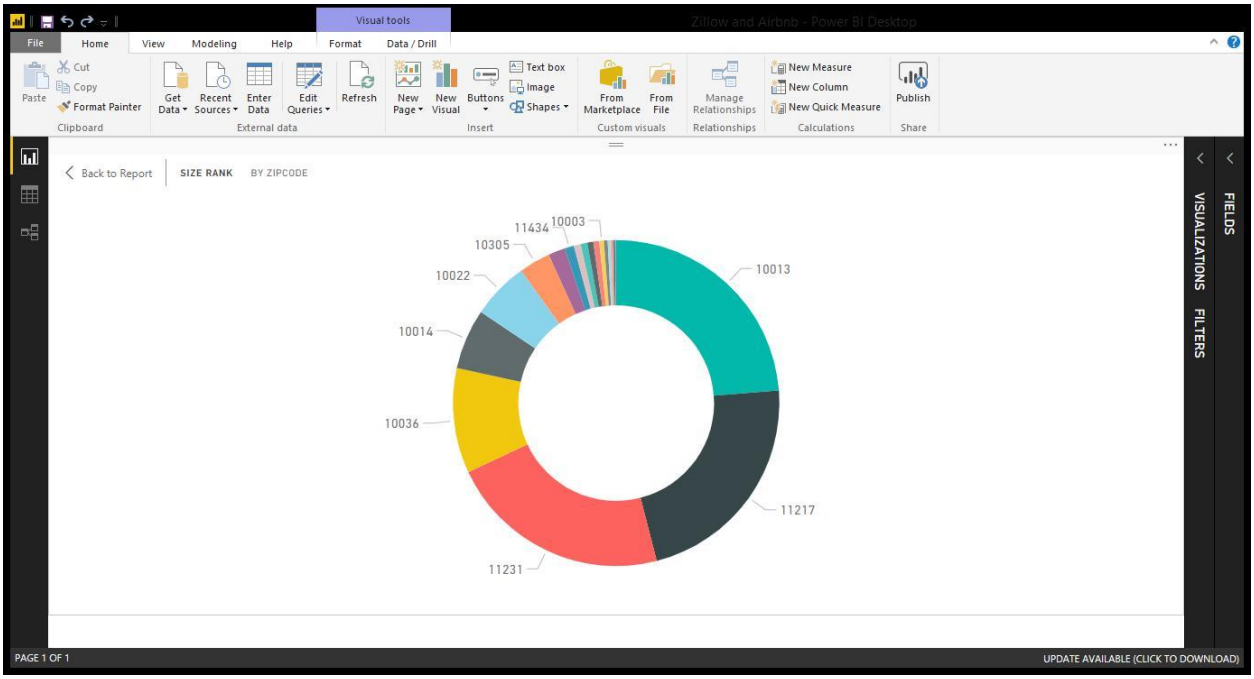


DATA VISUALIZATION GRAPHS USING MICROSOFT POWER BI

1. Daily, Weekly and Monthly Revenue by Zip codes.



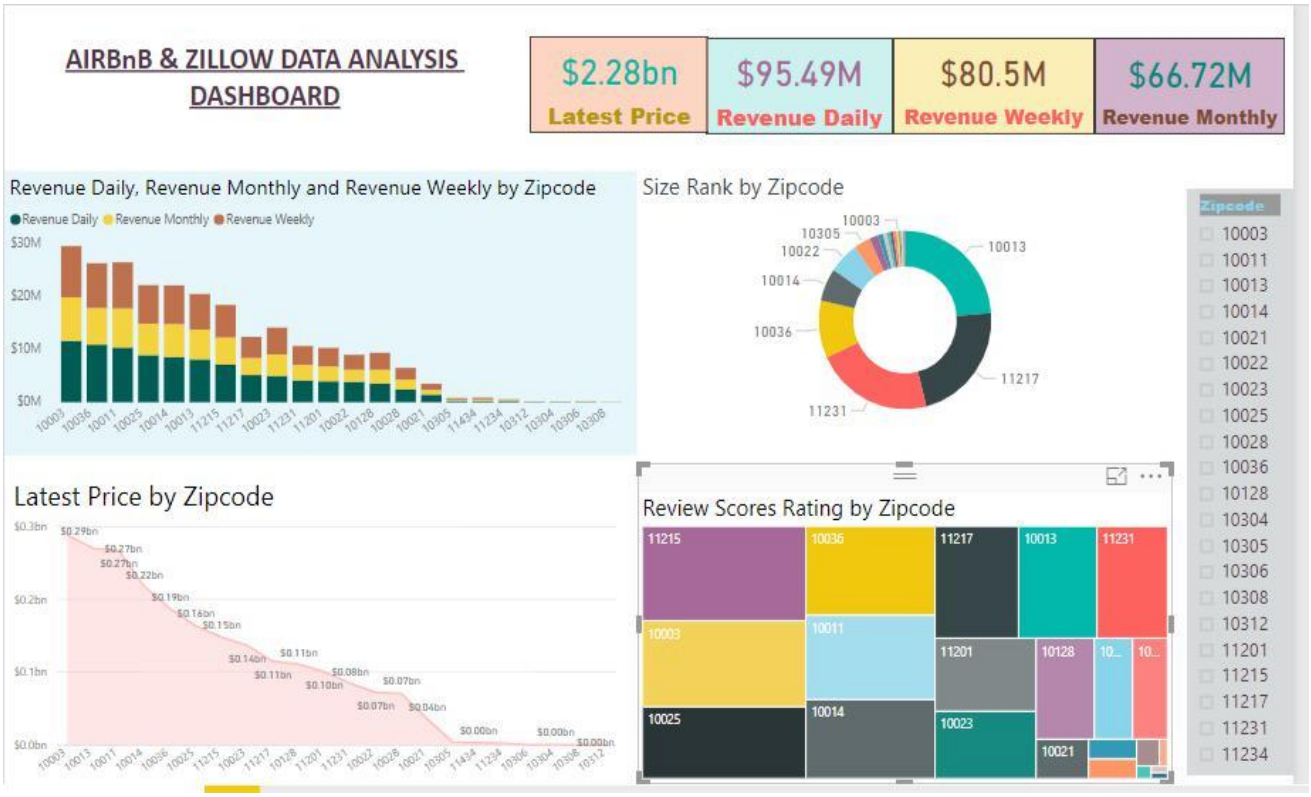
2. Size Rank of the Property by Zip Codes.



3. Latest Price (July 2017 to July 2018) of the properties by Zip codes.

4. Reviews Scores Ratings of the properties by Zip codes.

5. Dashboard of the Airbnb and Zillow Data Analysis.



1. To visit the Dashboard of Airbnb & Zillow in Tableau. Please visit this link.

https://public.tableau.com/profile/aarzoo.dawra#!/vizhome/AarzooZillowandAirbnb/AIRBnBZILLOWDATAANALYSISDASHBOARD