New York City Airbnb Analysis in Python

Dr. Shilpa Balan

CIS 5270 – Business Intelligence

College of Business and Economics

California State University, Los Angeles

By:
Ninelia Talverdi,
Raymond Delgado

May 2021

Contents

Introduction

New York City is the most populous city in the United States with an estimated population of 8.3 million people. It is a city of renters, vacancy rates are at crisis levels, and rents continue to rise. Income levels for the average New Yorker haven't kept pace, and affordability is at record lows. Housing is scarce; homelessness levels are increasing; food insecurity is growing; and economic and racial inequality rates in New York City are near the highest in the United States.

It's at this time that short term rental platforms, dominated by Airbnb, have entered the market, and grown to have listings of tens of thousands of rooms and entire apartments. (Inside Airbnb, 2021)

Since 2008, guests and hosts have used Airbnb to expand on traveling options and present a more “local” way of experiencing the city.

The goal of this project is to provide insights about Airbnb listings to understand the main factors that affect user’s decision to rent a property on Airbnb. We also wanted to see how Covid-19 pandemic has affected on Airbnb. In our analyses, we tried to explore and visualize the data for the following questions:

- How are rentals distributed among the five boroughs of New York City?
- What are the top 5 property types of listings?
- How are the number of listings reviews per year?
- What is the average price distribution by borough and year?
- Which words are used frequently in listings names?

Data Description

The dataset used for this analysis is "*listings.csv*" dataset which is a detailed listings data for New York City Airbnb. The data is scraped and assembled by Inside Airbnb and it is downloaded from Inside Airbnb website. The data behind the Inside Airbnb site is sourced from publicly available information from the Airbnb site. The data has been analyzed, cleansed and aggregated where appropriate to facilitate public discussion.

Dataset URL: http://data.insideairbnb.com/united-states/ny/new-york-city/2021-02-04/data/listings.csv.gz

The size of the dataset is 82.7MB. It consists of in total of 74 columns and 37012 rows.

The listings data includes a large set of attributes about the rental property such as listing name, hostname, property type, room type, price, availability, number of reviews, location of listings within all 5 boroughs of New York City, and etc.

The below table describes the dataset fields that are used in this project:

Field Name	Type	Description	Sample Data
id	int	A unique number identifying an Airbnb listing.	34760
name	object	Name of the listing.	Sunny Room in Old Historical Brooklyn Townhouse
host_name	object	Name of the listing's host. Usually just the first names.	Justin
host_response_rate	object	Indicates the percentage of new inquiries and reservation requests you responded to (by either accepting/pre-approving or declining) within 24 hours in the past 30 days.	100

neighbourhood_group_cleansed	object	Region of the listing - Represents one of the five boroughs in New York City in which a listing resides (Bronx, Staten Island, Queens, Brooklyn and Manhattan)	Brooklyn
latitude	float	latitude coordinates (The angular distance of a location or object north or south of the Earth's celestial equator)	40.69101
longitude	float	longitude coordinates (- The angular distance of a location or object east or west of the meridian)	-73.97312
property_type	object	Indicates the type of housing a listing	Private room in townhouse
room_type	object	Indicates the type of space available (Entire home/apt, Private room, Shared room)	Private room
price	object	Indicates the cost of a listing (in dollars)	74
number_of_reviews	int	Indicates the total number of reviews written about a listing	360
first_review	object	Indicates the date the first review of the listing is posted.	6/1/2017
last_review	object	Indicates the date the latest review of the listing is posted.	8/29/2019

Below are sample screenshots of the data in Spyder:

```
In [4]: runfile('C:/Users/Ninel/Desktop/School/CSULA/CIS 5270/Python Files/NYC_Airbnb.py', wdir='C:/Users/Ninel/Desktop/School/
CSULA/CIS 5270/Python Files')
Detailed Listings row, cols:  (37012, 74)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37012 entries, 0 to 37011
Data columns (total 74 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            37012 non-null  int64
 1   listing_url                                   37012 non-null  object
 2   scrape_id                                     37012 non-null  int64
 3   last_scraped                                  37012 non-null  object
 4   name                                          36999 non-null  object
 5   description                                   35789 non-null  object
 6   neighborhood_overview                         23329 non-null  object
 7   picture_url                                   37012 non-null  object
 8   host_id                                       37012 non-null  int64
 9   host_url                                      37012 non-null  object
 10  host_name                                     36994 non-null  object
 11  host_since                                    36994 non-null  object
 12  host_location                                 36896 non-null  object
 13  host_about                                    22168 non-null  object
 14  host_response_time                            18505 non-null  object
 15  host_response_rate                            18505 non-null  object
 16  host_acceptance_rate                          22379 non-null  object
 17  host_is_superhost                             36994 non-null  object
 18  host_thumbnail_url                            36994 non-null  object
 19  host_picture_url                              36994 non-null  object
 20  host_neighbourhood                            30323 non-null  object
 21  host_listings_count                           36994 non-null  float64
 22  host_total_listings_count                     36994 non-null  float64
 23  host_verifications                            37012 non-null  object
 24  host_has_profile_pic                          36994 non-null  object
 25  host_identity_verified                        36994 non-null  object
 26  neighbourhood                                 23329 non-null  object
 27  neighbourhood_cleansed                        37012 non-null  object
 28  neighbourhood_group_cleansed                  37012 non-null  object
 29  latitude                                      37012 non-null  float64
 30  longitude                                     37012 non-null  float64
 31  property_type                                 37012 non-null  object
 32  room_type                                     37012 non-null  object
 33  accommodates                                  37012 non-null  int64
 34  bathrooms                                     0 non-null      float64
 35  bathrooms_text                                36910 non-null  object
 36  bedrooms                                      33404 non-null  float64
 37  beds                                          36522 non-null  float64
 38  amenities                                     37012 non-null  object
 39  price                                         37012 non-null  object
 40  minimum_nights                                37012 non-null  int64
 41  maximum_nights                                37012 non-null  int64
 42  minimum_minimum_nights                        36894 non-null  float64
```

```
 43  maximum_minimum_nights                        36894 non-null  float64
 44  minimum_maximum_nights                        36894 non-null  float64
 45  maximum_maximum_nights                        36894 non-null  float64
 46  minimum_nights_avg_ntm                        36894 non-null  float64
 47  maximum_nights_avg_ntm                        36894 non-null  float64
 48  calendar_updated                              0 non-null      float64
 49  has_availability                              37012 non-null  object
 50  availability_30                               37012 non-null  int64
 51  availability_60                               37012 non-null  int64
 52  availability_90                               37012 non-null  int64
 53  availability_365                              37012 non-null  int64
 54  calendar_last_scraped                         37012 non-null  object
 55  number_of_reviews                             37012 non-null  int64
 56  number_of_reviews_ltm                         37012 non-null  int64
 57  number_of_reviews_l30d                        37012 non-null  int64
 58  first_review                                  27489 non-null  object
 59  last_review                                   27489 non-null  object
 60  review_scores_rating                          26777 non-null  float64
 61  review_scores_accuracy                        26753 non-null  float64
 62  review_scores_cleanliness                     26764 non-null  float64
 63  review_scores_checkin                         26741 non-null  float64
 64  review_scores_communication                   26755 non-null  float64
 65  review_scores_location                        26740 non-null  float64
 66  review_scores_value                           26740 non-null  float64
 67  license                                       0 non-null      float64
 68  instant_bookable                              37012 non-null  object
 69  calculated_host_listings_count                37012 non-null  int64
 70  calculated_host_listings_count_entire_homes   37012 non-null  int64
 71  calculated_host_listings_count_private_rooms  37012 non-null  int64
 72  calculated_host_listings_count_shared_rooms   37012 non-null  int64
 73  reviews_per_month                             27489 non-null  float64
dtypes: float64(23), int64(17), object(34)
memory usage: 20.9+ MB
None
```

Note: The data type of some of the fields are changed during the “Date Cleaning” process.

Here are screenshots of the first rows of the dataframe using “*head*” method:

```
None
     id                        listing_url       scrape_id last_scraped  \
0  2595  https://www.airbnb.com/rooms/2595  20210204180331   2021-02-05
1  3831  https://www.airbnb.com/rooms/3831  20210204180331   2021-02-05
2  5121  https://www.airbnb.com/rooms/5121  20210204180331   2021-02-05
3  5178  https://www.airbnb.com/rooms/5178  20210204180331   2021-02-05
4  5203  https://www.airbnb.com/rooms/5203  20210204180331   2021-02-06

                                          name  \
0                         Skylit Midtown Castle
1  Whole flr w/private bdrm, bath & kitchen(pls read)
2                                BlissArtsSpace!
3                  Large Furnished Room Near B'way
4                Cozy Clean Guest Room - Family Apt


description  \
0  Beautiful, spacious skylit studio in the heart of Midtown, Manhattan. <br /><br />STUNNING SKYLIT STUDIO / 1 BED
+ SINGLE / FULL BATH / FULL KITCHEN / FIREPLACE / CENTRALLY LOCATED / WiFi + APPLE TV / SHEETS + TOWELS<br /><br />
><b>The space</b><br />- Spacious (500+ft²), immaculate and nicely furnished & designed studio.<br />- Tuck
yourself into the ultra comfortable bed under the skylight. Fall in love with a myriad of bright lights in the city
night sky. <br />- Single-sized bed/convertible floor mattress with luxury bedding (available upon request).<br />-
Gorgeous pyramid skylight with amazing diffused natural light, stunning architectural details, soaring high vaulted
ceilings, exposed brick, wood burning fireplace, floor seating area with natural zafu cushions, modern style mixed
with eclectic art & antique treasures, large full bath, newly renovated kitchen, air conditioning/heat, high speed
WiFi Internet, and Apple TV.<br />- Centrally located in the heart of Midtown Manhattan
1  Enjoy 500 s.f. top floor in 1899 brownstone, w/ wood & ceramic flooring throughout, roomy bdrm, & upgraded
kitchen & bathroom.  This space is unique but one of the few legal AirBnbs with a totally private bedroom, private
full bathroom and private eat-in kitchen, SO PLEASE READ "THE SPACE" CAREFULLY.  It's sunny & loaded with
everything you need! Your floor, and the common staircase/hallway/entryway are cleaned/sanitized per Airbnb's
Enhanced Cleaning Protocol.<br /><br /><b>The space</b><br />We host on the entire top floor of our double-duplex
brownstone in Clinton Hill on Gates near Classon Avenue - (7 blocks to C train, 5 blocks to G train, minutes to
downtown Brooklyn & lower Manhattan).  It is not an apartment in the traditional sense, it is more of an efficiency
set-up and is TOTALLY LEGAL with all short-term rental laws. The top floor for our guests consists of a sizable
bedroom, full bath and eat-in kitchen for your exclusive use - you get the amenities of a private apartment
2  <b>The space</b><br />HELLO EVERYONE AND THANKS FOR VISITING BLISS ART SPACE! <br /><br />Thank you all for your
support. I've traveled a lot in the last year few years, to the  U.K. Germany, Italy and France! Loved Paris,
Berlin and Calabria! Highly recommend all these places. <br /><br /><br />One room available for rent in a 2
bedroom apt in Bklyn. We share a common space with kitchen. I am an artist(painter, filmmaker) and curator who is
working in the film industry while I'm building my art event production businesses.<br /><br />Price above is
nightly for one person. Monthly rates available.  Price is $900 per month for one person. Utilities not included,
they are about 50 bucks, payable when the bill arrives mid month.<br /> <br />Couples rates are slightly more for
monthly and 90$ per night short term. If you are a couple please Iet me know and I’ll give you the monthly rate for
that. Room rental is on a temporary basis, perfect from 2- 6 months - no long term requests please!
3  Please don’t expect the luxury here just a basic room in the center of Manhattan.<br /><br /><b>The space</b><br
/>You will use one large, furnished, private room of a two-bedroom apartment and share a bathroom with the host.
<br /><br />The apartment is located a few blocks away from Central Park between 8th and 9th Avenue.<br />The
closest subway station is Columbus Circle 59th Street.  Great restaurants, Broadway and all transportation are
easily accessible. <br /><br />The cost of the room is $79 per night.  Weekly rate is available.<br />There is a
$12.00 fee per second guest. <br /><br />The apartment also features hardwood floors and a second-floor walk-up.
```

```
$12.00 fee per second guest. <br /><br />The apartment also features hardwood floors and a second-floor walk-up.
<br />There is a full-sized bed,TV, microwave, and a small refrigerator as well as other appliances.  <br />Wired
internet, WIFI, TV, electric heat, bed sheets and towels are included. <br /><br />A kitchen is not available in
the living room. Please ask the host if you need.<br /><br />Basic check in/out time is 10 am. I am
4  Our best guests are seeking a safe, clean, spare room in a family apartment.  They are comfortable being
independent, accommodating of family noise (quiet hours 11pm-7am), and aren't afraid of a friendly two year old
golden lab (dog).  Our guests aren't put off by an old bathroom that while perfectly clean, has some peeling paint.
In short, our guests want to feel like they are staying at their sister's apartment while visiting the city! (only
their sister changed the sheets and cleaned).<br /><br /><b>The space</b><br />Stay in my family's little  guest
room and enjoy privacy, a warm welcome, and security. <br /><br />Your guest room is comfortable and clean.   It is
small but well outfitted, has a single bed and a fabulous mattress which is firm and yet pillowy on top, all at the
same time. The bathroom is shared and immediately across the hall. ("Shared" in the sense it isn't "en suite." The
family will use our second bath while you are staying with us). The bathroom is fully suppl


neighborhood_overview  \
0                                       Centrally located in the heart of Manhattan just a few blocks from all
subway connections in the very desirable Midtown location a few minutes walk to Times Square, the Theater District,
Bryant Park and Herald Square.
1  Just the right mix of urban center and local neighborhood; close to all but enough quiet for a calming walk. 15
to 45 minutes to most parts of Manhattan; 10 to 30 minutes to most Brooklyn points of interest; 45 minutes to 60
minutes to historic Coney Island.
2
NaN
3
Theater district, many restaurants around here.
4
Our neighborhood is full of restaurants and cafes.  There is plenty to do.

                                                                picture_url  \
0  https://a0.muscache.com/pictures/f0813a11-40b2-489e-8217-89a2e1637830.jpg
1  https://a0.muscache.com/pictures/e49999c2-9fd5-4ad5-b7cc-224deac989aa.jpg
2  https://a0.muscache.com/pictures/2090980c-b68e-4349-a874-4818402923e7.jpg
3              https://a0.muscache.com/pictures/12065/f070997b_original.jpg
4             https://a0.muscache.com/pictures/103776/b371575b_original.jpg

   host_id                              host_url    host_name  host_since  \
0     2845  https://www.airbnb.com/users/show/2845     Jennifer  2008-09-09
1     4869  https://www.airbnb.com/users/show/4869  LisaRoxanne  2008-12-07
2     7356  https://www.airbnb.com/users/show/7356        Garon  2009-02-03
3     8967  https://www.airbnb.com/users/show/8967     Shunichi  2009-03-03
4     7490  https://www.airbnb.com/users/show/7490    MaryEllen  2009-02-05

                      host_location  \
0  New York, New York, United States
1  New York, New York, United States
2  New York, New York, United States
3  New York, New York, United States
4  New York, New York, United States
```

```
host_about  \
0
A New Yorker since 2000! My passion is creating beautiful, unique spaces where unforgettable memories are made.
It's my pleasure to host people from around the world and meet new faces. Welcome travelers! \r\n\r\nI am a Sound
Therapy Practitioner and Kundalini Yoga & Meditation teacher. I work with energy and sound for relaxation and
healing, using Symphonic gong, singing bowls, tuning forks, drums, voice and other instruments.
1
Laid-back Native New Yorker (formerly bi-coastal) and AirBnb host of over 6 years and over 400 stays!  Besides
being a long-time and attentive AirBnb host, I am an actor, attorney, professor and group fitness instructor.
2
I am an artist(painter, filmmaker) and curator who is working in the film industry while I'm building my business.
\r\n\r\nI am extremely easy going and would like that you are the laid back\r\nand enjoy life kind of person. I
also ask that you are open, honest\r\nand easy to communicate with as this is how I like to live my life.And of
course creative people are very welcome!\r\n
3
I used to work for a financial industry but now I work at a Japanese food market as an assistant manager.
4  Welcome to family life with my oldest two away at college all the way down to a seventh grader. You may see
everything from lively dinner conversation to a nearly empty apartment with everyone out enjoying the city.  I'm
friendly, leave tea and coffee always available and responsive to a guest's needs.  My family has enjoyed
everything from the guest who tends towards the private as well as the ones who dive in with the science
experiment! \r\nHosting through Airbnb has created a wonderful opportunity to meet people from all over the world,
plot their addresses, and  learn about other places.   I began hosting through Airbnb four years ago as a work-
from-home job. I continue because the whole family entirely grooves on the notion we get to meet people from all
over the world and help them visit our city.

   host_response_time host_response_rate host_acceptance_rate  \
0  within a few hours                93%                 26%
1  within a few hours                98%                 93%
2                 NaN                NaN                 NaN
3        within a day               100%                100%
4                 NaN                NaN                 NaN

  host_is_superhost  \
0                 f
1                 f
2                 f
3                 f
4                 f

                                                                                host_thumbnail_url  \
0  https://a0.muscache.com/im/pictures/user/50fc57af-a6a3-4e88-8f16-efd6cac7c9bc.jpg?aki_policy=profile_small
1          https://a0.muscache.com/im/users/4869/profile_pic/1371927771/original.jpg?aki_policy=profile_small
2  https://a0.muscache.com/im/pictures/user/72a61bea-cfb1-45b6-abbb-85bdbd790b32.jpg?aki_policy=profile_small
3          https://a0.muscache.com/im/users/8967/profile_pic/1265419894/original.jpg?aki_policy=profile_small
4          https://a0.muscache.com/im/users/7490/profile_pic/1409068414/original.jpg?aki_policy=profile_small

                                                                                  host_picture_url  \
0  https://a0.muscache.com/im/pictures/user/50fc57af-a6a3-4e88-8f16-efd6cac7c9bc.jpg?aki_policy=profile_x_medium
1          https://a0.muscache.com/im/users/4869/profile_pic/1371927771/original.jpg?aki_policy=profile_x_medium
2  https://a0.muscache.com/im/pictures/user/72a61bea-cfb1-45b6-abbb-85bdbd790b32.jpg?aki_policy=profile_x_medium
```

```
4          https://a0.muscache.com/im/users/7490/profile_pic/1409068414/original.jpg?aki_policy=profile_x_medium

   host_neighbourhood  host_listings_count  host_total_listings_count  \
0             Midtown                  6.0                        6.0
1        Clinton Hill                  1.0                        1.0
2  Bedford-Stuyvesant                  1.0                        1.0
3      Hell's Kitchen                  1.0                        1.0
4     Upper West Side                  1.0                        1.0

                                                                                                    host_veri
fications  \
0  ['email', 'phone', 'reviews', 'offline_government_id', 'kba', 'selfie', 'government_id', 'identity_manual',
'work_email']
1                                          ['email', 'phone', 'reviews', 'offline_government_id', 'kba',
'government_id']
2           ['email', 'phone', 'facebook', 'reviews', 'offline_government_id', 'selfie', 'government_id',
'identity_manual']
3                                                                                  ['email', 'phone', 'facebook',
'reviews']
4                                           ['email', 'phone', 'facebook', 'google', 'reviews', 'jumio',
'government_id']

  host_has_profile_pic host_identity_verified  \
0                    t                       t
1                    t                       t
2                    t                       t
3                    t                       f
4                    t                       t

                       neighbourhood neighbourhood_cleansed  \
0            New York, United States               Midtown
1  Brooklyn, New York, United States          Clinton Hill
2                                NaN     Bedford-Stuyvesant
3            New York, United States               Midtown
4            New York, United States       Upper West Side

  neighbourhood_group_cleansed  latitude  longitude  \
0                    Manhattan  40.75362  -73.98377
1                     Brooklyn  40.68514  -73.95976
2                     Brooklyn  40.68688  -73.95596
3                    Manhattan  40.76468  -73.98315
4                    Manhattan  40.80178  -73.96723

              property_type        room_type  accommodates  bathrooms  \
0           Entire apartment  Entire home/apt             2        NaN
1         Entire guest suite  Entire home/apt             3        NaN
2  Private room in apartment     Private room             2        NaN
3  Private room in apartment     Private room             2        NaN
4  Private room in apartment     Private room             1        NaN
```

```
amenities  \
0                                                                         ["Refrigerator", "Air conditioning",
"Baking sheet", "Free street parking", "Bathtub", "Kitchen", "Keypad", "Coffee maker", "Oven", "Iron", "Hangers",
"Smoke alarm", "Dedicated workspace", "Fire extinguisher", "Hot water", "Long term stays allowed", "Extra pillows
and blankets", "Hair dryer", "Bed linens", "Essentials", "Dishes and silverware", "TV", "Wifi", "Heating", "Paid
parking off premises", "Cooking basics", "Stove", "Luggage dropoff allowed", "Cleaning before checkout", "Carbon
monoxide alarm", "Ethernet connection"]
1  ["Refrigerator", "Microwave", "Shampoo", "High chair", "Pack \u2019n Play/travel crib", "Air conditioning",
"Free street parking", "Bathtub", "Kitchen", "Coffee maker", "Oven", "Free parking on premises", "Iron", "Hangers",
"Smoke alarm", "Dedicated workspace", "Fire extinguisher", "Hot water", "Children\u2019s books and toys", "Long
term stays allowed", "Extra pillows and blankets", "Lockbox", "Hair dryer", "Bed linens", "Essentials", "Dishes and
silverware", "TV", "Wifi", "Cable TV", "Heating", "Cooking basics", "Stove", "Luggage dropoff allowed", "Baby
safety gates", "Carbon monoxide alarm"]
2
["Wifi", "Heating", "Air conditioning", "Long term stays allowed", "Kitchen"]
3
["Extra pillows and blankets", "Hair dryer", "Bed linens", "Iron", "Essentials", "Hangers", "Smoke alarm", "TV",
"Refrigerator", "Lock on bedroom door", "Shampoo", "Heating", "Wifi", "Microwave", "Paid parking off premises",
"Luggage dropoff allowed", "Air conditioning", "Free street parking"]
4
["Hair dryer", "Breakfast", "Smoke alarm", "Essentials", "Dedicated workspace", "Host greets you", "Wifi",
"Shampoo", "Heating", "Fire extinguisher", "Paid parking off premises", "Elevator", "Free street parking", "Hot
water", "Air conditioning", "Carbon monoxide alarm"]

     price  minimum_nights  maximum_nights  minimum_minimum_nights  \
0  $100.00              30            1125                    30.0
1   $73.00               1             730                     1.0
2   $60.00              30             730                    30.0
3   $79.00               2              14                     2.0
4   $75.00               2              14                     2.0

   maximum_minimum_nights  minimum_maximum_nights  maximum_maximum_nights  \
0                    30.0                  1125.0                  1125.0
1                     1.0                  1125.0                  1125.0
2                    30.0                   730.0                   730.0
3                     2.0                    14.0                    14.0
4                     2.0                    14.0                    14.0

   minimum_nights_avg_ntm  maximum_nights_avg_ntm  calendar_updated  \
0                    30.0                  1125.0               NaN
1                     1.0                  1125.0               NaN
2                    30.0                   730.0               NaN
3                     2.0                    14.0               NaN
4                     2.0                    14.0               NaN

  has_availability  availability_30  availability_60  availability_90  \
0                t               30               60               90
1                t                6               27               55
2                t               30               60               90
```

```
   availability_365 calendar_last_scraped  number_of_reviews  \
0               365           2021-02-05                 48
1               249           2021-02-05                386
2               365           2021-02-05                 50
3               343           2021-02-05                474
4                 0           2021-02-06                118

   number_of_reviews_ltm  number_of_reviews_l30d first_review last_review  \
0                      0                      0   2009-11-21  2019-11-04
1                     80                      3   2014-09-30  2021-01-27
2                      0                      0   2009-05-28  2019-12-02
3                     10                      0   2009-05-06  2020-09-25
4                      0                      0   2009-09-07  2017-07-21

   review_scores_rating  review_scores_accuracy  review_scores_cleanliness  \
0                  94.0                     9.0                        9.0
1                  90.0                    10.0                        9.0
2                  90.0                     8.0                        8.0
3                  84.0                     9.0                        8.0
4                  98.0                    10.0                       10.0

   review_scores_checkin  review_scores_communication  review_scores_location  \
0                   10.0                         10.0                     10.0
1                    9.0                         10.0                     10.0
2                   10.0                         10.0                      9.0
3                    9.0                          9.0                     10.0
4                   10.0                         10.0                     10.0

   review_scores_value  license instant_bookable  \
0                  9.0      NaN                f
1                 10.0      NaN                f
2                  9.0      NaN                f
3                  9.0      NaN                f
4                 10.0      NaN                f

   calculated_host_listings_count  \
0                               2
1                               1
2                               1
3                               1
4                               1

   calculated_host_listings_count_entire_homes  \
0                                            2
1                                            1
2                                            0
3                                            0
4                                            0
```

```
   calculated_host_listings_count_private_rooms  \
0                                             0
1                                             0
2                                             1
3                                             1
4                                             1

   calculated_host_listings_count_shared_rooms  reviews_per_month
0                                            0               0.35
1                                            0               4.99
2                                            0               0.35
3                                            0               3.31
4                                            0               0.85
```

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 26 09:06:43 2021

@author: Ninel
"""

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.patches as mpatches

import seaborn as sns

from matplotlib.pyplot import show

import matplotlib.image as mpimg
map_img = mpimg.imread('map.png')

from wordcloud import WordCloud

import warnings

warnings.filterwarnings('ignore')

#Set the display properties so that we can view the data
pd.set_option("display.max_rows", None, "display.max_columns", None, 'display.max_colwidth', None)

#Reading data from CSV file
data = pd.read_csv('listings.csv')

#Creating a copy of dataframe
df = data.copy()

#Showing number of rows and columns
print("Listings rows, cols: ", df.shape)

#Viewing summary of dataframe
print(df.info())

#Viewing first rows
print(df.head())
```

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.patches as mpatches

import seaborn as sns

from matplotlib.pyplot import show

import matplotlib.image as mpimg

map_img = mpimg.imread('map.png')

from wordcloud import WordCloud

import warnings

warnings.filterwarnings('ignore')


#Set the display properties so that we can view the data

pd.set_option("display.max_rows", None, "display.max_columns", None,
'display.max_colwidth', None)


#Reading data from CSV file

data = pd.read_csv('listings.csv')

#Creating a copy of dataframe

df = data.copy()

#Showing number of rows and columns

print("Listings rows, cols: ", df.shape)

#Viewing summary of dataframe

print(df.info())

#Viewing first rows

print(df.head())
```

Data Cleaning

After observing the raw data, we came out with the below plan for cleaning the data:

- Drop columns that are not relevant to the analysis. Example: URLs, columns with boolean values, etc.
- Find NA/missing values for each column and delete them.
- Convert columns to their correct data type. Example: “host_response_rate”
- Remove unnecessary symbols. Example: $ from “price”
- Split columns to multiple columns. Example: “first_review”

So, we cleaned the data using the following steps in Spyder:

1. Dropping out of scope data

Since our dataset was a large one containing many different types of columns, we decided to remove some columns such as ‘host_url’, ‘last_scraped’, ‘neighborhood_overview’, ‘calendar_last_scraped’, ‘calendar_updated’, 'review_scores_accuracy', ‘scrape_id', etc. from the dataset since they were not giving valuable information and were not that useful for our analysis. We deleted the irrelevant columns using “*drop*” method.

Before cleaning:

```
Shape of the dataset before dropping:  (37012, 74)
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'description',
       'neighborhood_overview', 'picture_url', 'host_id', 'host_url',
       'host_name', 'host_since', 'host_location', 'host_about',
       'host_response_time', 'host_response_rate', 'host_acceptance_rate',
       'host_is_superhost', 'host_thumbnail_url', 'host_picture_url',
       'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'neighbourhood',
       'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'latitude',
       'longitude', 'property_type', 'room_type', 'accommodates', 'bathrooms',
       'bathrooms_text', 'bedrooms', 'beds', 'amenities', 'price',
       'minimum_nights', 'maximum_nights', 'minimum_minimum_nights',
       'maximum_minimum_nights', 'minimum_maximum_nights',
       'maximum_maximum_nights', 'minimum_nights_avg_ntm',
       'maximum_nights_avg_ntm', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'number_of_reviews_ltm', 'number_of_reviews_l30d', 'first_review',
       'last_review', 'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'license', 'instant_bookable',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month'],
      dtype='object')
     id                          listing_url       scrape_id last_scraped  \
0  2595  https://www.airbnb.com/rooms/2595  20210204180331   2021-02-05
1  3831  https://www.airbnb.com/rooms/3831  20210204180331   2021-02-05
2  5121  https://www.airbnb.com/rooms/5121  20210204180331   2021-02-05
3  5178  https://www.airbnb.com/rooms/5178  20210204180331   2021-02-05
4  5203  https://www.airbnb.com/rooms/5203  20210204180331   2021-02-06

                                        name  \
0                          Skylit Midtown Castle
1  Whole flr w/private bdrm, bath & kitchen(pls read)
2                                BlissArtsSpace!
3                  Large Furnished Room Near B'way
4                Cozy Clean Guest Room - Family Apt

                                         description  \
0  Beautiful, spacious skylit studio in the heart of Midtown, Manhattan. <br /><br />STUNNING SKYLIT STUDIO / 1 BED
+ SINGLE / FULL BATH / FULL KITCHEN / FIREPLACE / CENTRALLY LOCATED / WiFi + APPLE TV / SHEETS + TOWELS<br /><br /
><b>The space</b><br />- Spacious (500+ft²), immaculate and nicely furnished & designed studio.<br />- Tuck
yourself into the ultra comfortable bed under the skylight. Fall in love with a myriad of bright lights in the city
night sky. <br />- Single-sized bed/convertible floor mattress with luxury bedding (available upon request).<br />-
Gorgeous pyramid skylight with amazing diffused natural light, stunning architectural details, soaring high vaulted
```

```
#Data Cleaning

#Dropping irrelevant columns

print("Shape of the dataset before dropping: ", df.shape)

print(df.columns)

print(df.head())

drop_cols =
df.drop(['listing_url','scrape_id','last_scraped','description','neighborhood_overview','picture_
url','host_url', 'bedrooms',
'host_location','host_since','host_about','host_is_superhost','host_thumbnail_url',
'bathrooms_text', 'host_picture_url', 'host_response_time', 'host_picture_url',
'host_neighbourhood','host_verifications', 'host_has_profile_pic',
'host_identity_verified','neighbourhood','amenities','has_availability', 'accommodates',
'calendar_last_scraped','instant_bookable','host_listings_count','bathrooms','minimum_minim
um_nights', 'beds', 'review_scores_value', 'host_id',
'maximum_minimum_nights','minimum_maximum_nights','maximum_maximum_nights',
'minimum_nights_avg_ntm', 'minimum_nights', 'maximum_nights_avg_ntm',
'calendar_updated','review_scores_accuracy', 'review_scores_cleanliness', 'maximum_nights',
'review_scores_checkin', 'review_scores_communication','review_scores_location','license',
'neighbourhood_cleansed', 'availability_30', 'availability_60','availability_90',
'availability_365', 'host_listings_count', 'has_availability', 'number_of_reviews_ltm',
'number_of_reviews_l30d','calculated_host_listings_count', 'host_total_listings_count',
'calculated_host_listings_count_entire_homes','calculated_host_listings_count_private_room
s', 'calendar_last_scraped', 'calculated_host_listings_count_shared_rooms',
'review_scores_rating', 'reviews_per_month',], axis = 1, inplace = True)

print("Shape of the dataset after dropping: ", df.shape)

print(df.columns)

print(df.head())
```

```
#Data Cleaning

#Dropping irrelevant columns
print("Shape of the dataset before dropping: ", df.shape)

print(df.columns)

print(df.head())

drop_cols = df.drop(['listing_url','scrape_id','last_scraped','description','neighborhood_overview','picture_url','host_url', 'bedrooms',
                     'host_location','host_since','host_about','host_is_superhost','host_thumbnail_url', 'bathrooms_text', 'review_scores_value',
                     'host_picture_url', 'host_response_time', 'host_picture_url', 'host_neighbourhood','host_verifications', 'host_id',
                     'host_has_profile_pic', 'host_identity_verified','neighbourhood','amenities','has_availability', 'accommodates',
                     'calendar_last_scraped','instant_bookable','host_listings_count','bathrooms','minimum_minimum_nights', 'beds',
                     'maximum_minimum_nights','minimum_maximum_nights','maximum_maximum_nights', 'minimum_nights_avg_ntm', 'minimum_nights',
                     'maximum_nights_avg_ntm', 'calendar_updated','review_scores_accuracy', 'review_scores_cleanliness', 'maximum_nights',
                     'review_scores_checkin', 'review_scores_communication','review_scores_location','license', 'neighbourhood_cleansed',
                     'availability_30', 'availability_60','availability_90', 'availability_365', 'host_listings_count', 'has_availability',
                     'number_of_reviews_ltm', 'number_of_reviews_l30d','calculated_host_listings_count', 'host_total_listings_count',
                     'calculated_host_listings_count_entire_homes','calculated_host_listings_count_private_rooms', 'calendar_last_scraped',
                     'calculated_host_listings_count_shared_rooms', 'review_scores_rating', 'reviews_per_month',], axis = 1, inplace = True)

print("Shape of the dataset after dropping: ", df.shape)

print(df.columns)

print(df.head())
```

The unnecessary columns are deleted from the dataset.

After cleaning:

```
Shape of the dataset after dropping:  (37012, 14)
Index(['id', 'name', 'host_name', 'host_response_rate', 'host_acceptance_rate',
       'neighbourhood_group_cleansed', 'latitude', 'longitude',
       'property_type', 'room_type', 'price', 'number_of_reviews',
       'first_review', 'last_review'],
      dtype='object')
     id                                              name    host_name  \
0  2595                             Skylit Midtown Castle     Jennifer
1  3831  Whole flr w/private bdrm, bath & kitchen(pls read)  LisaRoxanne
2  5121                                   BlissArtsSpace!        Garon
3  5178                Large Furnished Room Near B'way      Shunichi
4  5203              Cozy Clean Guest Room - Family Apt    MaryEllen

  host_response_rate host_acceptance_rate neighbourhood_group_cleansed  \
0                93%                  26%                     Manhattan
1                98%                  93%                      Brooklyn
2                NaN                  NaN                      Brooklyn
3               100%                 100%                     Manhattan
4                NaN                  NaN                     Manhattan

   latitude  longitude             property_type        room_type    price  \
0  40.75362  -73.98377          Entire apartment  Entire home/apt  $100.00
1  40.68514  -73.95976        Entire guest suite  Entire home/apt   $73.00
2  40.68688  -73.95596  Private room in apartment     Private room   $60.00
3  40.76468  -73.98315  Private room in apartment     Private room   $79.00
4  40.80178  -73.96723  Private room in apartment     Private room   $75.00

   number_of_reviews first_review last_review
0                 48   2009-11-21  2019-11-04
1                386   2014-09-30  2021-01-27
2                 50   2009-05-28  2019-12-02
3                474   2009-05-06  2020-09-25
4                118   2009-09-07  2017-07-21
```

2. Removing “NA”/ “NAN”, blank and invalid data

1) There were many blank or “NA” values for some columns. To create consistency in data for analysis, we decided to remove them using “*dropna*” method.

First, we checked the number of blank or “NA” values of each column:

```
Shape of the dataset before dropping:  (37012, 14)
id                                         0
name                                      13
host_name                                 18
host_response_rate                     18507
host_acceptance_rate                   14633
neighbourhood_group_cleansed               0
latitude                                   0
longitude                                  0
property_type                              0
room_type                                  0
price                                      0
number_of_reviews                          0
first_review                            9523
last_review                             9523
dtype: int64
```

```
#Dropping missing or NA/NAN values
print("Shape of the dataset before dropping: ", df.shape)

#Checking missing or NA/NAN values before dropping
print(df.isnull().sum())
```


```
#Dropping missing or NA/NAN values
print("Shape of the dataset before dropping: ", df.shape)

#Checking missing or NA/NAN values before dropping
print(df.isnull().sum())
```


There are many blank values for “name”, “host_resposne_rate”, “first_review”, “last_review” etc.

Before cleaning:

	name	host_response_rate	host_acceptance_rate
3720	Clean and Huge Studio in Safe Area	NaN	NaN
3721	Spacious Artist's Studio	NaN	NaN
3722	NaN	NaN	NaN
3723	Beautiful, Spacious Brooklyn Home	NaN	NaN
3724	Cozy and Vibey Apt in Williamsburg	NaN	NaN
3725	Cozy hideaway	NaN	NaN
3726	Penthouse w/Terrace Williamsburg	100%	75%
3727	Beautiful Place in Bushwick/Ridgewood	100%	91%
3728	Central One Bedroom Manhattan Apt	NaN	NaN
3729	Special OFFER on Airbnb NYC Room!	88%	83%
3730	Clean/Quiet Apt, Great Location, with washer/dryer	95%	99%

```
#Dropping missing or NA/NAN values
print("Shape of the dataset before dropping: ", df.shape)
#Checking missing or NA/NAN values before dropping
print(df.isnull().sum())
#Viewing specific rows and columns for missing or NA/NAN values before dropping
print(df.loc[3720:3730, ['name', 'host_response_rate', 'host_acceptance_rate']])
#Removing missing or NA/NAN values
df.dropna(inplace=True)
#Viewing specific rows and columns for missing or NA/NAN values after dropping
print(df.loc[3720:3730, ['name', 'host_response_rate', 'host_acceptance_rate']])
#Checking missing or NA/NAN values after dropping
print(df.isnull().sum())
print("Shape of the dataset after dropping: ", df.shape)
```

```
#Viewing specific rows and columns for missing or NA/NAN values before dropping
print(df.loc[3720:3730, ['name', 'host_response_rate', 'host_acceptance_rate']])

#Removing missing or NA/NAN values
df.dropna(inplace=True)

#Viewing specific rows and columns for missing or NA/NAN values after dropping
print(df.loc[3720:3730, ['name', 'host_response_rate', 'host_acceptance_rate']])

#Checking missing or NA/NAN values after dropping
print(df.isnull().sum())

print("Shape of the dataset after dropping: ", df.shape)
```

All rows with “NAN” and blank values were removed from the dataset.

After cleaning:

```
                                                  name host_response_rate  \
3726                    Penthouse w/Terrace Williamsburg               100%
3727                Beautiful Place in Bushwick/Ridgewood               100%
3729                    Special OFFER on Airbnb NYC Room!                88%
3730  Clean/Quiet Apt, Great Location, with washer/dryer                 95%

     host_acceptance_rate
3726                  75%
3727                  91%
3729                  83%
3730                  99%
id                                0
name                              0
host_name                         0
host_response_rate                0
host_acceptance_rate              0
neighbourhood_group_cleansed      0
latitude                          0
longitude                         0
property_type                     0
room_type                         0
price                             0
number_of_reviews                 0
first_review                      0
last_review                       0
dtype: int64
Shape of the dataset after dropping:  (13276, 14)
```

3. Deleting duplicate data

We checked to see if there are any duplicate values in the dataset using "*duplicated*" function.

```
Empty DataFrame
Columns: [id, name, host_name, host_response_rate, host_acceptance_rate, neighbourhood_group_cleansed,
latitude, longitude, property_type, room_type, price, number_of_reviews, first_review, last_review]
Index: []
```

```
Checking duplicate values

rint(df[df.duplicated()])
```

```
#Checking duplicate values
print(df[df.duplicated()])
```

There were no duplicate values.

4. Removing symbols

Since we wanted to analyze the statistics of data, we had to change the type of some of the columns. Before doing that, we decided to remove the unnecessary alphabets or symbols from the values.

1) We removed the "$" and "," from the values of "price" column using "*replace*" method.

Before cleaning:

	id	property_type	price
0	2595	Entire apartment	$100.00
1	3831	Entire guest suite	$73.00
3	5178	Private room in apartment	$79.00
5	5803	Private room in townhouse	$83.00
8	6990	Private room in apartment	$62.00
9	7097	Entire apartment	$199.00
11	7801	Entire loft	$299.00
12	8490	Entire loft	$120.00
14	9657	Entire apartment	$150.00
16	10452	Private room in apartment	$70.00
17	10962	Private room in townhouse	$83.00
19	12192	Private room in apartment	$40.00

```
#Removing specific symbols

#Viewing "price" column before removing symbols

print(df.loc[0:20, ['id', 'property_type', 'price']])

#Remove $ from "price"

df['price'] = df['price'].str.replace('$','')

#Remove , from "price"

df['price'] = df['price'].str.replace(',','')

#Viewing "price" column after removing symbols

print(df.loc[0:20, ['id', 'property_type', 'price']])
```


```
#Removing specific symbols

#Viewing "price" column before removing symbols
print(df.loc[0:20, ['id', 'property_type', 'price']])

#Remove $ from "price"
df['price'] = df['price'].str.replace('$','')

#Remove , from "price"
df['price'] = df['price'].str.replace(',','')

#Viewing "price" column after removing symbols
print(df.loc[0:20, ['id', 'property_type', 'price']])
```

After cleaning:

	id	property_type	price
0	2595	Entire apartment	100.00
1	3831	Entire guest suite	73.00
3	5178	Private room in apartment	79.00
5	5803	Private room in townhouse	83.00
8	6990	Private room in apartment	62.00
9	7097	Entire apartment	199.00
11	7801	Entire loft	299.00
12	8490	Entire loft	120.00
14	9657	Entire apartment	150.00
16	10452	Private room in apartment	70.00
17	10962	Private room in townhouse	83.00
19	12192	Private room in apartment	40.00

2) We also removed the "%" from the values of "host_response_rate" and "host_acceptance_rate" columns using "*replace*" method.

Before cleaning:

	host_name	host_response_rate	host_acceptance_rate
0	Jennifer	93%	26%
1	LisaRoxanne	98%	93%
3	Shunichi	100%	100%
5	Laurie	100%	100%
8	Cyn	100%	100%
9	Jane	100%	100%
11	Chaya	100%	62%
12	Nathalie	100%	95%
14	Dana	100%	83%
16	Angela	100%	100%
17	Laurie	100%	100%
19	Edward	75%	81%

```
#Viewing "host_response_rate", "host_acceptance_rate" columns before removing symbols
print(df.loc[0:20, ['host_name', 'host_response_rate', 'host_acceptance_rate']])
#Remove % from "host_response_rate", "host_acceptance_rate"
df['host_response_rate'] = df['host_response_rate'].str.replace('%','')
df['host_acceptance_rate'] = df['host_acceptance_rate'].str.replace('%','')
#Viewing "host_response_rate", "host_acceptance_rate" columns after removing symbols
print(df.loc[0:20, ['host_name', 'host_response_rate', 'host_acceptance_rate']])
```

```
#Viewing "host_response_rate", "host_acceptance_rate" columns before removing symbols
print(df.loc[0:20, ['host_name', 'host_response_rate', 'host_acceptance_rate']])

#Remove % from "host_response_rate", "host_acceptance_rate"
df['host_response_rate'] = df['host_response_rate'].str.replace('%','')

df['host_acceptance_rate'] = df['host_acceptance_rate'].str.replace('%','')

#Viewing "host_response_rate", "host_acceptance_rate" columns after removing symbols
print(df.loc[0:20, ['host_name', 'host_response_rate', 'host_acceptance_rate']])
```

The unnecessary symbols are removed.

After cleaning:

	host_name	host_response_rate	host_acceptance_rate
0	Jennifer	93	26
1	LisaRoxanne	98	93
3	Shunichi	100	100
5	Laurie	100	100
8	Cyn	100	100
9	Jane	100	100
11	Chaya	100	62
12	Nathalie	100	95
14	Dana	100	83
16	Angela	100	100
17	Laurie	100	100
19	Edward	75	81

5. Fixing data types

We had to change the type of some of the variables (“price”, “host_acceptance_rate”, “host_response_rate”, “first_review” and “last_review”) before analyzing the statistics of data and visualization.

Let’s take a look at data types again after cleaning the dataset.

```
object     10
int64       2
float64     2
dtype: int64
{int64: ['id', 'number_of_reviews'], float64: ['latitude', 'longitude'], object: ['name', 'host_name',
'host_response_rate', 'host_acceptance_rate', 'neighbourhood_group_cleansed', 'property_type',
'room_type', 'price', 'first_review', 'last_review']}
```

```
#Checking how the data types are distributed.
print(df.dtypes.value_counts())
print(df.columns.to_series().groupby(df.dtypes).groups)
```

```
#Checking how the data types are distributed.
print(df.dtypes.value_counts())

print(df.columns.to_series().groupby(df.dtypes).groups)
```

We decided to change the data type of "price", host_acceptance_rate", "host_resposne_rate" to *float* and convert "first_review" and "last_review" to *date*.

Converting data types

Before cleaning:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13276 entries, 0 to 36915
Data columns (total 14 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   id                           13276 non-null  int64
 1   name                         13276 non-null  object
 2   host_name                    13276 non-null  object
 3   host_response_rate           13276 non-null  object
 4   host_acceptance_rate         13276 non-null  object
 5   neighbourhood_group_cleansed 13276 non-null  object
 6   latitude                     13276 non-null  float64
 7   longitude                    13276 non-null  float64
 8   property_type                13276 non-null  object
 9   room_type                    13276 non-null  object
 10  price                        13276 non-null  object
 11  number_of_reviews            13276 non-null  int64
 12  first_review                 13276 non-null  object
 13  last_review                  13276 non-null  object
dtypes: float64(2), int64(2), object(10)
memory usage: 2.1+ MB
None
```

We used "*astype*" method to convert object to float type and used "*to_datetime*" method to convert string type to date time.

```
#Fixing data types

#Viewing data types before converting

print(df.info())

#Converting data types

df['price'] = df['price'].astype(float)

df['host_acceptance_rate'] = df['host_acceptance_rate'].astype(float)

df['host_response_rate'] = df['host_response_rate'].astype(float)

df['first_review'] = pd.to_datetime(df['first_review'])

df['last_review'] = pd.to_datetime(df['last_review'])

#Viewing data types after converting

print(df.info())
```

```
#Fixing data types

#Viewing data types before converting
print(df.info())

#Converting data types
df['price'] = df['price'].astype(float)
df['host_acceptance_rate'] = df['host_acceptance_rate'].astype(float)
df['host_response_rate'] = df['host_response_rate'].astype(float)
df['first_review'] = pd.to_datetime(df['first_review'])
df['last_review'] = pd.to_datetime(df['last_review'])

#Viewing data types after converting
print(df.info())
```

All columns are converted to their appropriate types.

After cleaning:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13276 entries, 0 to 36915
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype         
---  ------                        --------------  -----         
 0   id                            13276 non-null  int64         
 1   name                          13276 non-null  object        
 2   host_name                     13276 non-null  object        
 3   host_response_rate            13276 non-null  float64       
 4   host_acceptance_rate          13276 non-null  float64       
 5   neighbourhood_group_cleansed  13276 non-null  object        
 6   latitude                      13276 non-null  float64       
 7   longitude                     13276 non-null  float64       
 8   property_type                 13276 non-null  object        
 9   room_type                     13276 non-null  object        
 10  price                         13276 non-null  float64       
 11  number_of_reviews             13276 non-null  int64         
 12  first_review                  13276 non-null  datetime64[ns]
 13  last_review                   13276 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(5), int64(2), object(5)
memory usage: 2.1+ MB
None
```

6. Splitting a column into multiple columns

Since the “first_review” and “last_review” columns were containing a complete value of dates, we decided to split them into separate ones to be able to use the values separately if needed for our analysis.

First, we changed the type of “first_review” and “last_review” columns from *date* to *string* because we wanted to use “*astype*” method.

```
#Converting to datatype
df['first_review'] = df['first_review'].astype(str)
df['last_review'] = df['last_review'].astype(str)
```

Next, we used the “*split*” function to split “first_review” and “last_review” columns into separate columns.

Before cleaning:

	id	first_review	last_review
0	2595	2009-11-21	2019-11-04
1	3831	2014-09-30	2021-01-27
3	5178	2009-05-06	2020-09-25
5	5803	2009-04-23	2020-10-17
8	6990	2009-10-28	2019-12-09
9	7097	2010-01-16	2021-01-25
11	7801	2009-08-09	2011-12-28
12	8490	2009-08-25	2021-01-11
14	9657	2009-09-08	2020-11-06
16	10452	2010-04-18	2019-10-12
17	10962	2009-09-27	2020-09-04
19	12192	2009-10-27	2020-11-14

```
#Viewing 'first_review' , 'last_review' before splitting

print(df.loc[0:25, ['id', 'first_review', 'last_review']])

#Split columns

df[["first_review_year", "first_review_month", "first_review_day"]] =
df["first_review"].str.split("-", expand = True)

df[["last_review_year", "last_review_month", "last_review_day"]] =
df["last_review"].str.split("-", expand = True)

#Viewing 'first_review' , 'last_review' before splitting

print(df.loc[0:25, ['id', 'first_review_year', 'first_review_month', 'first_review_day',
'last_review_year','last_review_month', 'last_review_day']])
```

```
#Splitting 'first_review' , 'last_review' columns

#Converting to datatype
df['first_review'] = df['first_review'].astype(str)
df['last_review'] = df['last_review'].astype(str)

#Viewing 'first_review' , 'last_review' before splitting
print(df.loc[0:25, ['id', 'first_review', 'last_review']])

#Split columns
df[["first_review_year", "first_review_month", "first_review_day"]] = df["first_review"].str.split("-", expand = True)
df[["last_review_year", "last_review_month", "last_review_day"]] = df["last_review"].str.split("-", expand = True)

#Viewing 'first_review' , 'last_review' before splitting
print(df.loc[0:25, ['id', 'first_review_year', 'first_review_month', 'first_review_day', 'last_review_year','last_review_month', 'last_review_day']])
```

New columns are created in our dataset.

After cleaning:

	id	first_review_year	first_review_month	first_review_day	\
0	2595	2009	11	21	
1	3831	2014	09	30	
3	5178	2009	05	06	
5	5803	2009	04	23	
8	6990	2009	10	28	
9	7097	2010	01	16	
11	7801	2009	08	09	
12	8490	2009	08	25	
14	9657	2009	09	08	
16	10452	2010	04	18	
17	10962	2009	09	27	
19	12192	2009	10	27	

	last_review_year	last_review_month	last_review_day
0	2019	11	04
1	2021	01	27
3	2020	09	25
5	2020	10	17
8	2019	12	09
9	2021	01	25
11	2011	12	28
12	2021	01	11
14	2020	11	06
16	2019	10	12
17	2020	09	04
19	2020	11	14

Let's take a look at the final data after cleaning process using "*head*" method.

```
     id                                                name    host_name  \
0  2595                              Skylit Midtown Castle     Jennifer
1  3831  Whole flr w/private bdrm, bath & kitchen(pls read)  LisaRoxanne
3  5178                     Large Furnished Room Near B'way      Shunichi
5  5803      Lovely Room 1, Garden, Best Area, Legal rental       Laurie
8  6990                             UES Beautiful Blue Room          Cyn

   host_response_rate  host_acceptance_rate neighbourhood_group_cleansed  \
0                93.0                  26.0                    Manhattan
1                98.0                  93.0                     Brooklyn
3               100.0                 100.0                    Manhattan
5               100.0                 100.0                     Brooklyn
8               100.0                 100.0                    Manhattan

   latitude  longitude             property_type        room_type  price  \
0  40.75362  -73.98377          Entire apartment  Entire home/apt  100.0
1  40.68514  -73.95976        Entire guest suite  Entire home/apt   73.0
3  40.76468  -73.98315  Private room in apartment     Private room   79.0
5  40.66829  -73.98779  Private room in townhouse     Private room   83.0
8  40.78962  -73.94802  Private room in apartment     Private room   62.0

   number_of_reviews first_review last_review first_review_year  \
0                 48   2009-11-21  2019-11-04              2009
1                386   2014-09-30  2021-01-27              2014
3                474   2009-05-06  2020-09-25              2009
5                182   2009-04-23  2020-10-17              2009
8                233   2009-10-28  2019-12-09              2009

  first_review_month first_review_day last_review_year last_review_month  \
0                 11                21             2019                11
1                 09                30             2021                01
3                 05                06             2020                09
5                 04                23             2020                10
8                 10                28             2019                12

  last_review_day
0              04
1              27
3              25
5              17
8              09
```

After completing the cleaning process, we decided to save the dataset in a new CSV file to be able to perform analysis and visualization.

We used "*to_csv*" method to save our dataset in a CSV file.

```
#Viewing first rows

print(df.head())

#Save dataframe in a new CSV file

df.to_csv (r'C:\Users\Ninel\Desktop\School\CSULA\CIS 5270\Python
Files\clean_listings.csv', index = False, header=True)
```

Statistical Summary

Statistical Summary

Below we see the statistical summary of our complete dataset using "*describe*" method.

```
                 id  host_response_rate  host_acceptance_rate      latitude  \
count  1.327600e+04        13276.000000          13276.000000  13276.000000
mean   2.616258e+07           90.494577             83.023576     40.727254
std    1.504778e+07           22.121937             23.777037      0.058127
min    2.595000e+03            0.000000              0.000000     40.508680
25%    1.336091e+07           93.000000             77.000000     40.686077
50%    2.833200e+07          100.000000             93.000000     40.724475
75%    4.003776e+07          100.000000            100.000000     40.762110
max    4.797869e+07          100.000000            100.000000     40.910780

          longitude         price  number_of_reviews
count  13276.000000  13276.000000       13276.000000
mean     -73.945511    130.823290          42.332932
std        0.056775    170.066052          63.766155
min      -74.239860     10.000000           1.000000
25%      -73.983990     59.000000           4.000000
50%      -73.953370     94.000000          15.000000
75%      -73.925260    149.000000          54.000000
max      -73.710870   4500.000000         753.000000
```

```
#Viewing basic statistical details

print(df.describe())
```

```
#Viewing basic statistical details
print(df.describe())
```

➢ Statistical summary of price

```
Here is an overview of 'price' column:
count    13276.000000
mean       130.823290
std        170.066052
min         10.000000
25%         59.000000
50%         94.000000
75%        149.000000
max       4500.000000
Name: price, dtype: float64
Mean of price is: 130.82329014763482
Standard deviation of price is: 170.06605240299382
Minimum of price is: 10.0
25th percentile of price is: 59.0
Median of price is: 94.0
75th percentile of price is: 149.0
Maximum of price is: 4500.0
```

```
#Statistics for 'price'
print("Here is an overview of 'price' column:")
print(df['price'].describe())
print('Mean of price is:' , df.price.mean())
print('Standard deviation of price is:' , df.price.std())
print('Minimum of price is:' , df.price.min())
print('25th percentile of price is:' , df.price.quantile(0.25))
print('Median of price is:' , df.price.quantile(0.5))
print('75th percentile of price is:' , df.price.quantile(0.75))
print('Maximum of price is:' , df.price.max())
```

```
#Statistics for 'price'
print("Here is an overview of 'price' column:")
print(df['price'].describe())

print('Mean of price is:' , df.price.mean())
print('Standard deviation of price is:' , df.price.std())
print('Minimum of price is:' , df.price.min())
print('25th percentile of price is:' , df.price.quantile(0.25))
print('Median of price is:' , df.price.quantile(0.5))
print('75th percentile of price is:' , df.price.quantile(0.75))
print('Maximum of price is:' , df.price.max())
```

Based on the statistical summary of price, the min is 10 and max value is 4500. The mean value for the price is 130.82 which means a vast majority of rentals have low prices. The 75% percentile value is 149. This also shows that most of the rentals are not expensive.

Lastly the standard deviation is almost 170 which is greater than the mean. So more of our data is clustered about the mean.

To analyze the price more accurately, we decided to break down the price per borough.

```
                              count        mean         std   min   25%  \
neighbourhood_group_cleansed
Bronx                         528.0   88.952652   83.969677  18.0  45.0
Brooklyn                     5098.0  122.826403  141.791976  17.0  57.0
Manhattan                    5407.0  159.483447  213.966912  10.0  70.0
Queens                       2056.0   89.383268   98.659896  10.0  45.0
Staten Island                 187.0   93.983957   64.666346  24.0  51.5

                               50%     75%     max
neighbourhood_group_cleansed
Bronx                         65.0  100.00   914.0
Brooklyn                      95.0  149.00  4500.0
Manhattan                    100.0  169.00  3900.0
Queens                        65.0  100.25  2293.0
Staten Island                 79.0  118.00   431.0
```

```
#Showing distribution of price in each borough
print(df.groupby('neighbourhood_group_cleansed')['price'].describe())
```

```
#Showing distribution of price in each borough
print(df.groupby('neighbourhood_group_cleansed')['price'].describe())
```

The above table clearly shows the distribution of price in each borough and across all. The number of listings is not even. Most Airbnb listings are in either Manhattan or Brooklyn. However, these two regions have also the highest prices among the five regions. A possible reason is that because the demands in these regions are high, causing more hosts to rent out their rooms or apartments.

➢ Statistical summary of host response rate

```
Here is an overview of 'host_response_rate' column:
count    13276.000000
mean        90.494577
std         22.121937
min          0.000000
25%         93.000000
50%        100.000000
75%        100.000000
max        100.000000
Name: host_response_rate, dtype: float64
Mean of host resposne rate is: 90.49457667972281
Standard deviation of host resposne rate is: 22.1219367049945
Minimum of host resposne rate is: 0.0
25th percentile of host resposne rate is: 93.0
Median of host resposne rate is: 100.0
75th percentile of host resposne rate is: 100.0
Maximum of host resposne rate is: 100.0
```

```
#Statistics for 'host_response_rate'

print("Here is an overview of 'host_response_rate' column:")

print(df['host_response_rate'].describe())


print('Mean of host resposne rate is:' , df.host_response_rate.mean())

print('Standard deviation of host resposne rate is:' , df.host_response_rate.std())

print('Minimum of host resposne rate is:' , df.host_response_rate.min())

print('25th percentile of host resposne rate is:' , df.host_response_rate.quantile(0.25))

print('Median of host resposne rate is:' , df.host_response_rate.quantile(0.5))

print('75th percentile of host resposne rate is:' , df.host_response_rate.quantile(0.75))

print('Maximum of host resposne rate is:' , df.host_response_rate.max())
```

```
#Statistics for 'host_response_rate'
print("Here is an overview of 'host_response_rate' column:")
print(df['host_response_rate'].describe())

print('Mean of host resposne rate is:' , df.host_response_rate.mean())
print('Standard deviation of host resposne rate is:' , df.host_response_rate.std())
print('Minimum of host resposne rate is:' , df.host_response_rate.min())
print('25th percentile of host resposne rate is:' , df.host_response_rate.quantile(0.25))
print('Median of host resposne rate is:' , df.host_response_rate.quantile(0.5))
print('75th percentile of host resposne rate is:' , df.host_response_rate.quantile(0.75))
print('Maximum of host resposne rate is:' , df.host_response_rate.max())
```

Based on the statistical summary, the min value of host response rate is 0 percent and the max value is 100%. Only 404 listings had a response rating value of 0 and 7894 listings had a response rating value of 100 which is a high number out of total number of listings (13276).

The average host response rate is 90 percent which means a large percentage of new inquiries and reservation requests are responded by the hosts.

Lastly the standard deviation is almost 22% which is a low value. This means that the host response rate values are close to the mean of our set.

The below table shows the distribution of host response rate in each borough:

neighbourhood_group_cleansed	count	mean	std	min	25%	50%	75%	max
Bronx	528.0	92.541667	20.941732	0.0	99.75	100.0	100.0	100.0
Brooklyn	5098.0	91.356414	21.196102	0.0	95.00	100.0	100.0	100.0
Manhattan	5407.0	88.875902	23.524516	0.0	90.00	100.0	100.0	100.0
Queens	2056.0	91.786479	20.794994	0.0	94.00	100.0	100.0	100.0
Staten Island	187.0	93.818182	19.322089	0.0	100.00	100.0	100.0	100.0

```
#Showing distribution of host_response_rate in each borough
print(df.groupby('neighbourhood_group_cleansed')['host_response_rate'].describe())
```

```
#Showing distribution of host_response_rate in each borough
print(df.groupby('neighbourhood_group_cleansed')['host_response_rate'].describe())
```

As we can see, Manhattan has the lowest mean comparing to other regions. It might be due to receiving lots of inquiries. The 50th percentile, 75th percentile and maximum have same values which is 100%.

➢ Statistical summary of number of listings

```
Here is an overview of 'number_of_reviews' column:
count    13276.000000
mean        42.332932
std         63.766155
min          1.000000
25%          4.000000
50%         15.000000
75%         54.000000
max        753.000000
Name: number_of_reviews, dtype: float64
Mean of host resposne rate is: 42.332931605905394
Standard deviation of host resposne rate is: 63.76615496636431
Minimum of host resposne rate is: 1
25th percentile of host resposne rate is: 4.0
Median of host resposne rate is: 15.0
75th percentile of host resposne rate is: 54.0
Maximum of host resposne rate is: 753
The range of the 'number_of_reviews' column: 752
```

```
#Statistics for 'number_of_reviews'

print("Here is an overview of 'number_of_reviews' column:")

print(df['number_of_reviews'].describe())


print('Mean of host resposne rate is:' , df.number_of_reviews.mean())

print('Standard deviation of host resposne rate is:' , df.number_of_reviews.std())

print('Minimum of host resposne rate is:' , df.number_of_reviews.min())

print('25th percentile of host resposne rate is:' , df.number_of_reviews.quantile(0.25))

print('Median of host resposne rate is:' , df.number_of_reviews.quantile(0.5))

print('75th percentile of host resposne rate is:' , df.number_of_reviews.quantile(0.75))

print('Maximum of host resposne rate is:' , df.number_of_reviews.max())

print("The range of the 'number_of_reviews' column:", df.number_of_reviews.max()-
df.number_of_reviews.min())
```

```
#Showing distribution of host_response_rate in each borough
print(df.groupby('neighbourhood_group_cleansed')['host_response_rate'].describe())

#Statistics for 'number_of_reviews'
print("Here is an overview of 'number_of_reviews' column:")
print(df['number_of_reviews'].describe())

print('Mean of host resposne rate is:' , df.number_of_reviews.mean())
print('Standard deviation of host resposne rate is:' , df.number_of_reviews.std())
print('Minimum of host resposne rate is:' , df.number_of_reviews.min())
print('25th percentile of host resposne rate is:' , df.number_of_reviews.quantile(0.25))
print('Median of host resposne rate is:' , df.number_of_reviews.quantile(0.5))
print('75th percentile of host resposne rate is:' , df.number_of_reviews.quantile(0.75))
print('Maximum of host resposne rate is:' , df.number_of_reviews.max())
print("The range of the 'number_of_reviews' column:", df.number_of_reviews.max()-df.number_of_reviews.min())

#Showing distribution of host_response_rate in each borough
print(df.groupby(['neighbourhood_group_cleansed'])['number_of_reviews'].describe())
```

Based on the statistical summary, the count of the ‘number_of_reviews’ column is 13276, this means that 13276 individual properties have been reviewed. The mean of this column is 44.33 which means that there is a constant engagement on an individual listing. The standard deviation of the column is 63.75 which shows the individual data points are widely spread, meaning there is a major disparity of numbers of reviews vary greatly by listing. The minimum value is 1 which means there is a listing with only one review. The first quartile of listings maxes out at 4 reviews. The second quartile of listings maxes out at 15 reviews. It must be noted that the second quartile is the same as the median. Thus, the median of this column is 15. This means that one half of listings have less than 15 reviews, but the second half of listings have more than 15 reviews. The third quartile of listings maxes out at 54 reviews. The max value is 753, which means that the listing with the most review had 753 reviews. The range of this column is 752, which is the difference between the minimum (1) and maximum (753). The range of the reviews provide more evidence that data points vary greatly.

Analysis and Visualizations

Let's take a look at the overall distribution of Airbnb listings in New York City before we start the analyses.

Figure 1 – Map showing NYC Airbnb listings distribution

A quick summary of the dataset revealed a total of 13,276 listings in New York City. The above map shows how listings are distributed in 5 boroughs of New York City but it doesn't show exactly which area has the highest number of listings. We will explore this in our first analysis.

Note: Please keep in mind that the below analyses are based on a clean dataset and lots of rows have been removed during data cleaning process.

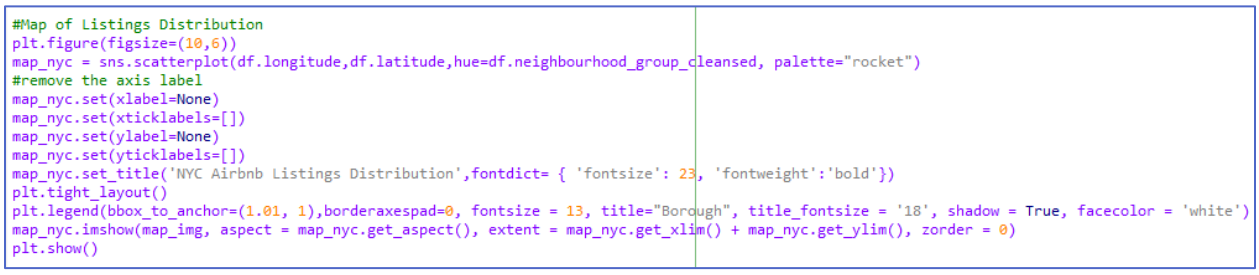
Features used in Python Spyder:

Plot Type: scatter plot on map

Libraries: matplotlib.pyplot, seaborn

Methods: figure(), set(), set_title(), tight_layout(), legend(), imshow(), show()

```
import matplotlib.pyplot as plt

import seaborn as sns


#Map of Listings Distribution

plt.figure(figsize=(10,6))

map_nyc = sns.scatterplot(df.longitude,df.latitude,hue=df.neighbourhood_group_cleansed, palette="rocket")

#remove the axis label

map_nyc.set(xlabel=None)

map_nyc.set(xticklabels=[])

map_nyc.set(ylabel=None)

map_nyc.set(yticklabels=[])

map_nyc.set_title('NYC Airbnb Listings Distribution',fontdict= { 'fontsize': 23, 'fontweight':'bold'})

plt.tight_layout()

plt.legend(bbox_to_anchor=(1.01, 1),borderaxespad=0, fontsize = 13, title="Borough", title_fontsize = '18', shadow = True, facecolor = 'white')

map_nyc.imshow(map_img, aspect = map_nyc.get_aspect(), extent = map_nyc.get_xlim() + map_nyc.get_ylim(), zorder = 0)

plt.show()
```



1. How are rentals distributed among the five boroughs of New York City?

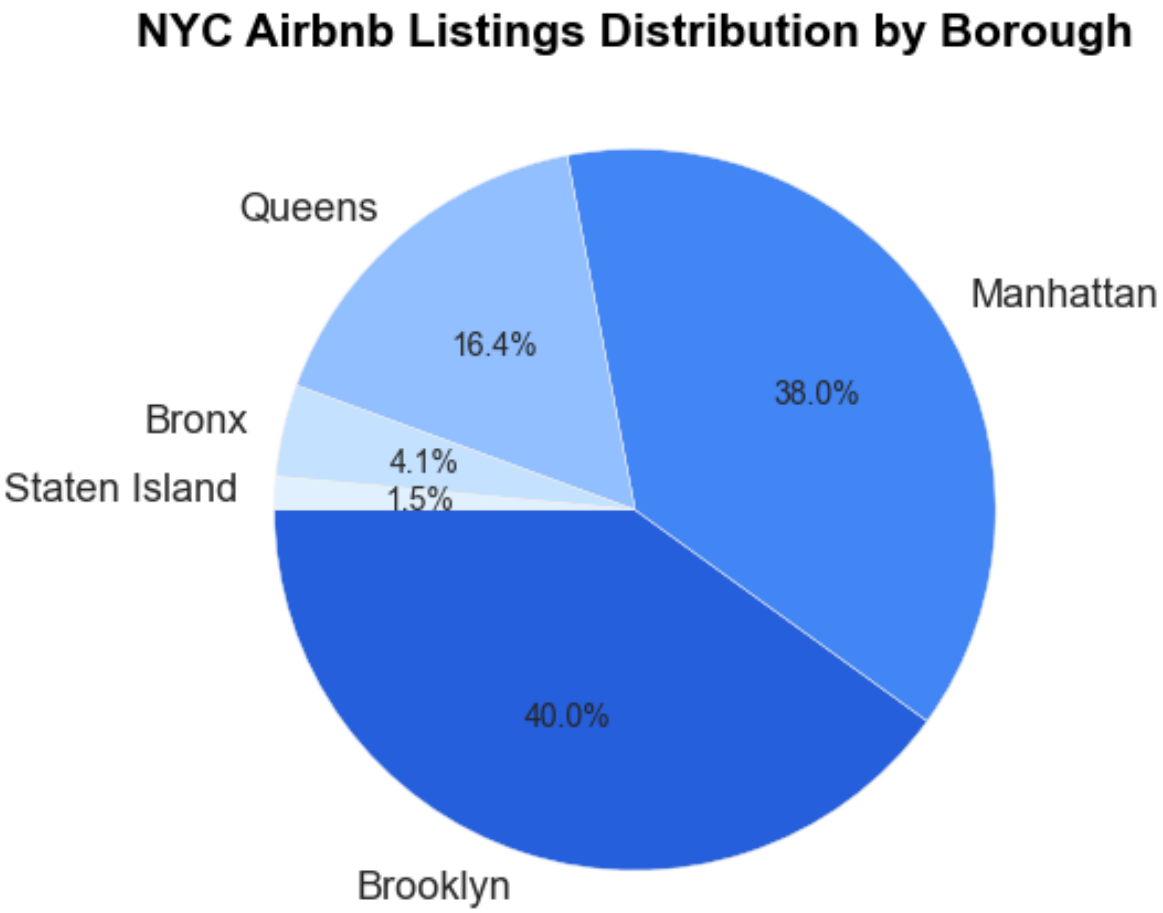


Figure 2 - Pie chart showing percentage of listings distribution by borough

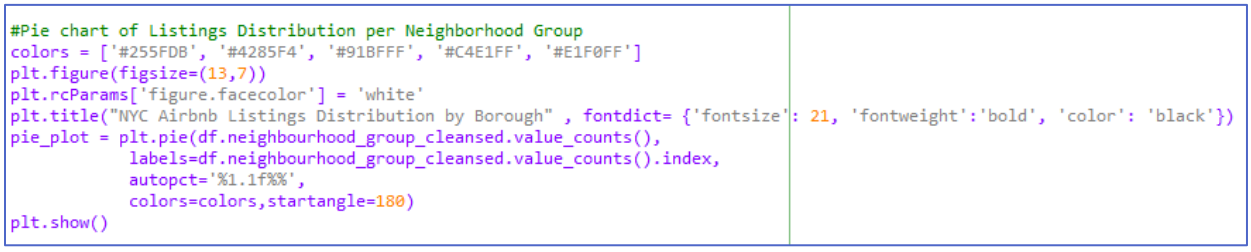
Features used in Python Spyder:

Plot Type: pie plot

Libraries: matplotlib.pyplot

Methods: figure(), rcParams[], title(), pie(), show()

```
import matplotlib.pyplot as plt

#Pie chart of Listings Distribution per Neighborhood Group

colors = ['#255FDB', '#4285F4', '#91BFFF', '#C4E1FF', '#E1F0FF']

plt.figure(figsize=(13,7))

plt.rcParams['figure.facecolor'] = 'white'

plt.title("NYC Airbnb Listings Distribution by Borough" , fontdict= {'fontsize': 21,
'fontweight':'bold', 'color': 'black'})

pie_plot = plt.pie(df.neighbourhood_group_cleansed.value_counts(),
labels=df.neighbourhood_group_cleansed.value_counts().index, autopct='%1.1f%%',
colors=colors,startangle=180)

plt.show()
```



Insights:

Our initial approach to analyze the data was to first see how listings are distributed across different boroughs. As it's reflected in the above visualization, Brooklyn and Manhattan concentrate the majority of the listed rentals on Airbnb, adding up to 78% of the listings. This means that the bulk of visitors of New York stay in properties, rooms or residencies located in these two areas. This might be because Brooklyn and Manhattan are better areas comparing to other regions of New York City since they are well known for arts, entertainment, commuting, dining, and nightlife. So, visitors are more interested to stay in the places located in these two areas. Another reason is that people can find almost any type of property in either of these boroughs even though the prices are

high. After Brooklyn and Manhattan, Queens comes in at third place with 16.4 percentage. Lastly, Staten Island and Bronx have the lowest percentage of listings. This can be because they are best regions for buying a house rather than renting a place. So, there are more houses than apartments in these regions. Moreover, based on Natural Areas Conservancy's map of New York City, Staten Island is New York's greenest borough. 59% of Staten Island is covered in either landscaped or natural greenery. On the other hand, Manhattan and Brooklyn have lower percentages. Only 31% of Brooklyn and 28% of Manhattan are covered with green. This can also be one of the reasons that there are more properties in these areas and thence more listings are available on Airbnb.

2. What are the top 5 property types of listings?

Figure 3 - Vertical bar chart showing top 5 property types of listings

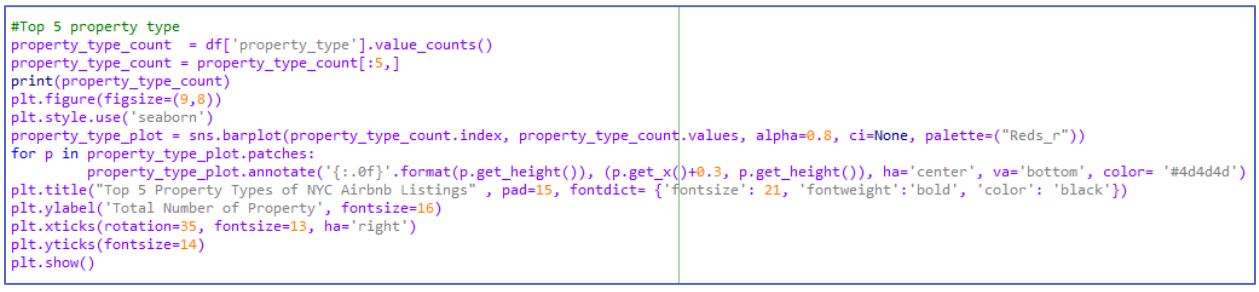
Features used in Python Spyder:

Plot Type: bar plot (vertical)

Libraries: matplotlib.pyplot, seaborn

Methods: value_counts(), figure(), style.use(), barplot(), annotate(), title(), show()

```
import matplotlib.pyplot as plt

import seaborn as sns


#Top 5 property type

property_type_count  = df['property_type'].value_counts()

property_type_count = property_type_count[:5,]

print(property_type_count)

plt.figure(figsize=(9,8))

plt.style.use('seaborn')

property_type_plot = sns.barplot(property_type_count.index, property_type_count.values, alpha=0.8, ci=None, palette=("Reds_r"))

for p in property_type_plot.patches:

        property_type_plot.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.3, p.get_height()), ha='center', va='bottom', color= '#4d4d4d')

plt.title("Top 5 Property Types of NYC Airbnb Listings" , pad=15, fontdict= {'fontsize': 21, 'fontweight':'bold', 'color': 'black'})

plt.ylabel('Total Number of Property', fontsize=16)

plt.xticks(rotation=35, fontsize=13, ha='right')

plt.yticks(fontsize=14)

plt.show()
```



Insights:

Our next thought was to find the top 5 property types that are most popular among users.

As we can see in the above bar chart, among the top 5 property types, the most demanded ones are apartment type with 4755 listings followed by private-room in apartment type with 2874 listings. This can be due to having a high number of listings located in Manhattan and Brooklyn since they have more “Apartment” style properties than “House” style. On the other hand, Staten Island with the lowest number of listings has more ‘House’ style property than ‘Apartment’. So, the house types are less available. In addition, renting an apartment seem to be more affordable than renting a house so people tend to rent apartment type properties more when traveling.

Having privacy also seems to be a popular feature among the top property types. Nowadays, people care more about their privacy and they prefer to rent places that are not shared with other guests either the entire property or a private room in these properties. We assume that’s why hosts rent out these types of properties more on Airbnb.

3. How are the number of listings reviews per year?

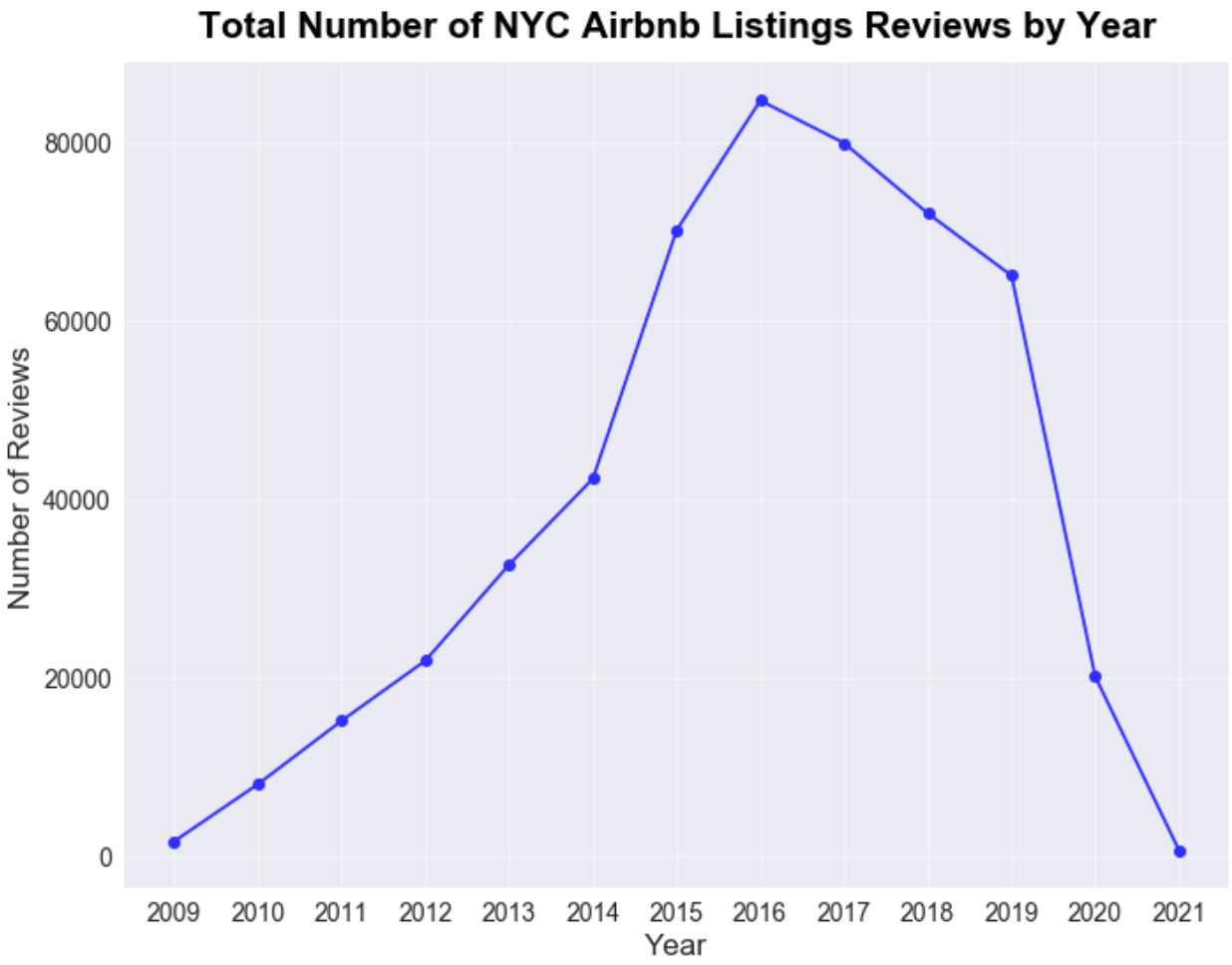


Figure 4 Line chart showing number of listings reviews per year

Features used in Python Spyder:

Plot Type: line plot

Libraries: matplotlib.pyplot

Methods: groupby().sum[], style.use(), figure(), plot(), title()

Please note the dataframe that is used in our line chart is the following:

first_review_year	number_of_reviews
2009	1667
2010	8066
2011	15179
2012	21907
2013	32668
2014	42285
2015	70112
2016	84658
2017	79891
2018	71496
2019	64116
2020	19457
2021	614

```
import matplotlib.pyplot as plt


#To create a new dataframe to hold the data that will be used for the line chart.

time_df = df.groupby('first_review_year').sum()[['number_of_reviews']]

#To see our new dataframe

#print(time_df)

#Convert the index from time_df and assign the output to a new dataframe called 'time_df_fin_ver'

time_df.reset_index(inplace=True)

time_df_fin_ver = time_df.rename(columns = {'index':'group_by_values'})

#To see the new data frame

#print(time_df_fin_ver)
```

```
#Assign values to the variables that are going to store

# The year (x-axis) and total number of revie2s(y-axis)

y_axis = time_df_fin_ver['number_of_reviews']

x_axis = time_df_fin_ver['first_review_year']


#To see the variables for our y and x axis

#print(y_axis)

#print(x_axis)

plt.style.use('seaborn')


# Creates and define the paramaters for the line chart.

plt.figure(figsize=(10,8))

plt.style.use('seaborn')

plt.plot(x_axis, y_axis, marker = 'o', color='#2E2EFF')

plt.xlabel("Year", fontsize=17)

plt.ylabel("Number of Reviews", fontsize=17)

plt.title("Total Number of NYC Airbnb Listings Reviews by Year", pad=15, fontdict=
{'fontsize': 21, 'fontweight':'bold', 'color': 'black'})

plt.xticks(x_axis)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.grid(b=True, linewidth=0.5)

axes = plt.gca()

plt.show()
```

```
#To create a new dataframe to hold the data that will be used for the line chart.
time_df = df.groupby('first_review_year').sum()[['number_of_reviews']]
#Uncomment 22 to see our new dataframe
#print(time_df)

#Convert the index from time_df and assign the output to a new dataframe called 'time_df_fin_ver'
time_df.reset_index(inplace=True)
time_df_fin_ver = time_df.rename(columns = {'index':'group_by_values'})
#To see the new data frame
#print(time_df_fin_ver)

#Assign values to the variables that are going to store
#The year (x-axis) and total number of revie2s(y-axis)
y_axis = time_df_fin_ver['number_of_reviews']
x_axis = time_df_fin_ver['first_review_year']
#To see the variables for our y and x axis
#print(y_axis)
#print(x_axis)

#Creates and define the paramaters for the line chart.
plt.figure(figsize=(10,8))
plt.style.use('seaborn')
plt.plot(x_axis, y_axis, marker = 'o', color='blue')
plt.xlabel("Year", fontsize=17)
plt.ylabel("Number of Reviews", fontsize=17)
plt.title("Total Number of NYC Airbnb Listings Reviews by Year", pad=15, fontdict= {'fontsize': 21, 'fontweight':'bold', 'color': 'black'})
plt.xticks(x_axis)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.grid(b=True, linewidth=0.5)
axes = plt.gca()
plt.show()
```

Insights:

Our next motive was to find out the number of reviews per year. The above line chart shows that there was constant growth in user activity from the year 2009 to its peak in 2016. It must be noted that user activity had a massive 65.8% increase between the years of 2014 to 2015. However, the line chart also shows a gradual decrease in user activity from the 2016 to 2019 and it displays a massive 69.6 % decrease in user activity between the users of 2019 and 2020. The main contributor to lack of user engagement in 2020 is the Covid-19 pandemic. The tourism and hospitality industries were both impacted by the pandemic--Airbnb operates within both of these industries. For instance, in her article in the New York times, Erin Griffith states that Airbnb suffered $1.2 billion decrease in revenue during the first nine months of 2020. Griffith cites the travel restrictions and the consumer uncertainty as reasons for the decrease in revenue (Griffith, 2020).

Please note that the line chart does not provide a complete analysis on the user engagement within 2021 and it only includes data on the first quarter of 2021. It must be noted that travel restrictions are being lifted and many people feel more optimistic about traveling again. For example, in his

article on NPR, Bill Chappell states that local officials will completely reopen New York City on July 1st (Chappell, 2021). Therefore, we expect the user engagement will be greater than what it was in 2020.

4. What is the average price distribution by borough and year?

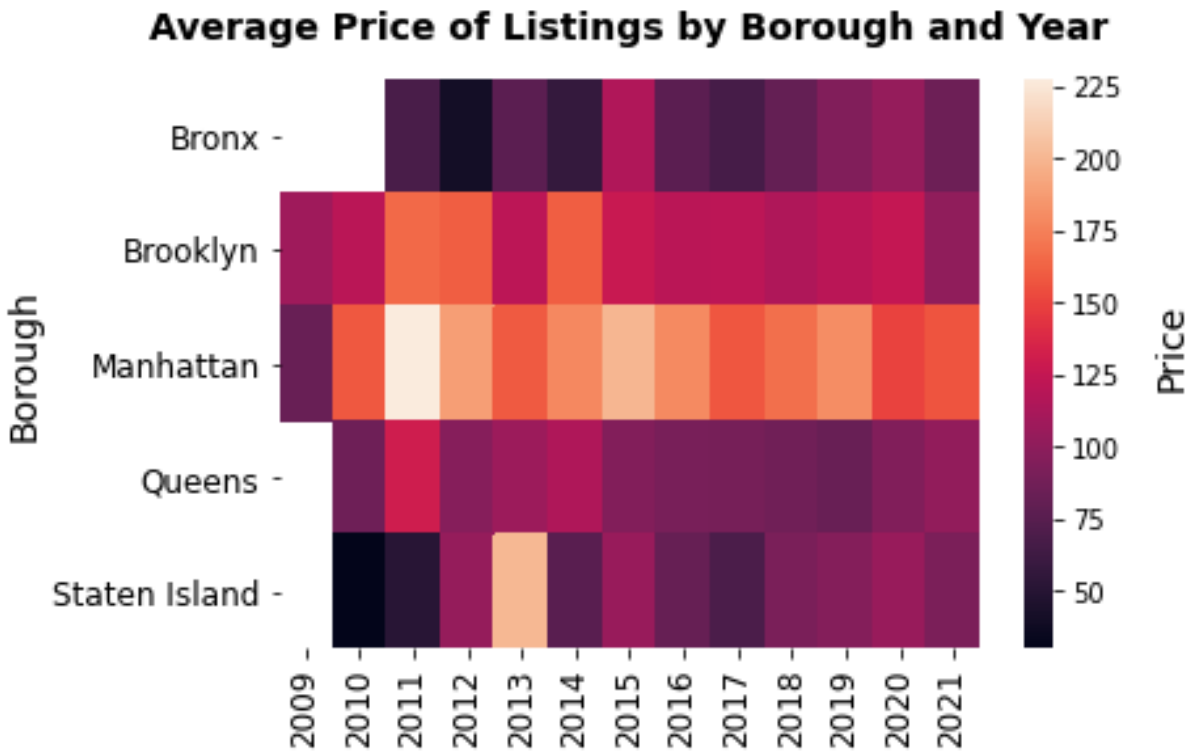


Figure 5 – Heatmap showing distribution of average price of listings by borough and year

Features used in Python Spyder:

Plot Type: heatmap

Libraries: matplotlib.pyplot, seaborn

Methods: groupby().mean[], reset_index(), rename_columns(), sort_values(), pivot(), heatmap(), title(), show()

Note: Due to the final dataframe being pivoted, there are missing values in our heatmap.

```
import matplotlib.pyplot as plt

import seaborn as sns


#Average price by borough and year

#Creates a new dataframe to group the average price of a listing by NYC borough and year

hm_df = df.groupby(['neighbourhood_group_cleansed','first_review_year']).mean()[['price']]

#print(hm_df.head(10))

To reset the index of hd_df and assign the value into a new data frame

hm_df.reset_index(inplace=True)

hm_df_v2 = hm_df.rename(columns = {'index':'group_by_values'})

#print(hm_df_v2.head(10))

#Sort the results by price (High --> Low)

hm_df_v3 = hm_df_v2.sort_values('price',ascending = False)

#print (hm_df_v3.head(10))

#Pivot value of hm_df_v3 and assign it to a new data frame

hm_df_finver= hm_df_v3.pivot('neighbourhood_group_cleansed','first_review_year','price')

#print(hm_df_finver.head(10))

#Creates the heatmap to display the average price of all listings within a certain NYC borough and

Year

ax = sns.heatmap(hm_df_finver, cbar_kws={'label': 'Price'})

ax.figure.axes[-1].yaxis.label.set_size(14)

plt.title("Average Price of Listings by Borough and Year", pad=15, fontdict= {'fontsize': 14,
'fontweight':'bold', 'color': 'black'})

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.xlabel("")

plt.ylabel("Borough", fontsize=14)

plt.show()
```

```
#Creates a new dataframe to group the average price of a listing by NYC borough and year
hm_df = df.groupby(['neighbourhood_group_cleansed','first_review_year']).mean()[['price']]
#print(hm_df.head(10))

#To reset the index of hd_df and assign the value into a new data frame
hm_df.reset_index(inplace=True)
hm_df_v2 = hm_df.rename(columns = {'index':'group_by_values'})
#print(hm_df_v2.head(10))

#Sort the results by price (High --> Low)
hm_df_v3 = hm_df_v2.sort_values('price',ascending = False)
#print (hm_df_v3.head(10))

#Pivot value of hm_df_v3 and assign it to a new data frame
hm_df_finver= hm_df_v3.pivot('neighbourhood_group_cleansed','first_review_year','price')
#print(hm_df_finver.head(10))

#Creates the heatmap to display the average price of all listings within a certain NYC borough and Year

ax = sns.heatmap(hm_df_finver, cbar_kws={'label': 'Price'})
ax.figure.axes[-1].yaxis.label.set_size(14)
plt.title("Average Price of Listings by Borough and Year", pad=15, fontdict= {'fontsize': 14, 'fontweight':'bold', 'color': 'black'})
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel("")
plt.ylabel("Borough", fontsize=14)
plt.show()
```

Insights:

The first insight that we gained from the Heatmap is that Manhattan and Brooklyn are the only boroughs with no records missing. However, the Bronx is the borough with the least number of records--it is missing data for 2009 and 2010. The highest average price was paid by consumers in Manhattan in 2011. Moreover, Manhattan tends to have the highest average price per listing. Conversely, the Bronx tends to have the lowest average price per listing. We believe that Manhattan has the highest average price because of the tourist attractions found within Manhattan. For instance, both Broadway and Wall Street are both found within Manhattan. Unfortunately, the Bronx does have the same amount of tourist attractions. In fact, criminality is a big issue in the Bronx. The rate of violent crime per 1,000 residents is 8.57 which is higher than both the rest of NY State (3.59) and national median (4) (Neighborhood Scout). Overall, the key insight is that the heat map highlights the disparities between the boroughs.

5. Which words are used frequently in listings names?

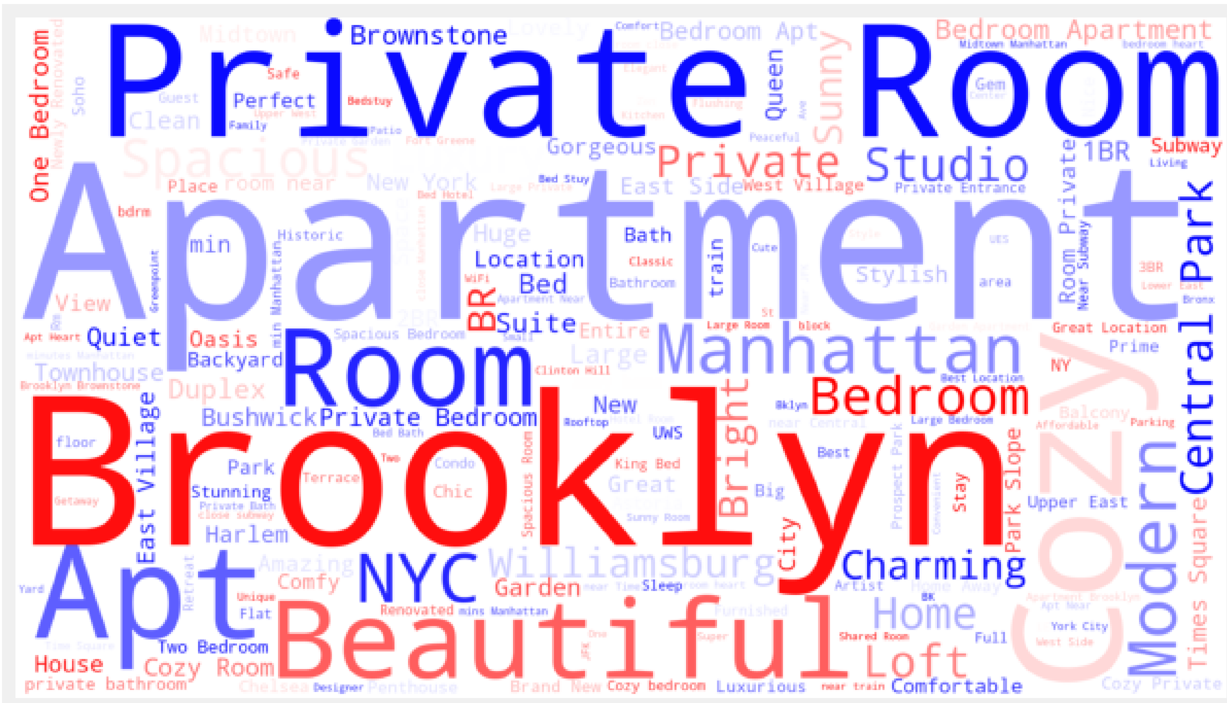


Figure 6 – Word cloud showing most used words in listings names

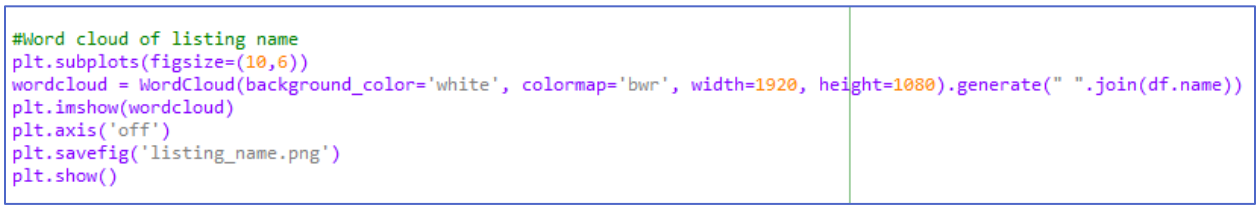
<u>Features used in Python Spyder:</u>

Plot Type: wordcloud

Methods: subplot(), wordcloud(), axis(), imshow(), savefig(), show()

Libraries: matplotlib.pyplot, wordcloud

```
import matplotlib.pyplot as plt

from wordcloud import WordCloud


#Word cloud of listing name

plt.subplots(figsize=(10,6))

wordcloud = WordCloud(background_color='white', colormap='bwr', width=1920,
height=1080).generate(" ".join(df.name))

plt.imshow(wordcloud)

plt.axis('off')

plt.savefig('listing_name.png')

plt.show()
```



Insights:

Our last thought was to find out which words appear the most in the listings' names. For example, the words "Apartment", "Brooklyn", "Private Room" are three of the biggest words in our word cloud. This means that these are the most common phases used in names of many Airbnb listings. Many hosts used the phrase "private room" to entice potential users. Thus, we can deduce that privacy is a desirable quality in a listing. This makes sense because many people would like to stay in a place that is private and secure. The fact that the word "apartment" is common reflects the reality of New York City. According to the New York City department of City Planning, New York City has the highest population density (27,000 people per square mile) of any U.S city (New

York City Department of City Planning, 2021). Thus, New York City has a large number of apartment complexes to house a very large population within a limited space. Furthermore, Brooklyn is a common text because of its fame and centralized location. It must be noted that Brooklyn is near Manhattan, JFK International Airport, and The Statue of Liberty. We can deduce that the hosts who include the word “Brooklyn” are trying to advertise the location of their listing. Overall, the word cloud provides us with insight on what phrases and themes users are looking for when searching for Airbnb listings.

Conclusion

Our goal was to find out main factors that affects user's decision to rent a property on Airbnb. Although the analyses done above was not very deep, we concluded the following:

- Brooklyn and Manhattan have the highest number of listings on Airbnb.
- Apartment type is the most popular property among the property types of listings.
- The number of listings reviews had a massive decrease in 2020 due to Covid-19 pandemic.
- Airbnb listings in Manhattan has the highest average price.
- "Apartment", "Brooklyn", "Private Room" are most used words in listings names.

References

Chappell, B. (2021, April 29). *New York City, Former COVID-19 Epicenter, To 'Fully Reopen' On July 1.* Retrieved from npr: https://www.npr.org/sections/coronavirus-live-updates/2021/04/29/991949214/new-york-city-former-covid-19-epicenter-to-fully-reopen-on-july-1

Clarke, K. (2020, July 31). *Which are New York's Greenest Boroughs?* Retrieved from Maps Mania: http://googlemapsmania.blogspot.com/2020/07/which-are-new-yorks-greenest-boroughs.html

Griffith, E. (2020, November 16). *Airbnb Reveals Falling Revenue, With Travel Hit by Pandemic.* Retrieved from New York Times: https://www.nytimes.com/2020/11/16/technology/airbnb-revenue-prospectus-ipo.html

Inside Airbnb. (2021, May 8). *New York City.* Retrieved from Inside Airbnb: http://insideairbnb.com/new-york-city/index.html?neighbourhood=&filterEntireHomes=true&filterHighlyAvailable=false&filterRecentReviews=true&filterMultiListings=true

Neighborhood Scout. (2021, May 6). *Bronx, NY Crime Rate.* Retrieved from Neighborhood Scout: https://www.neighborhoodscout.com/ny/bronx/crime

New York City Department of City Planning. (2021, May 6). *Population - New York City Population.* Retrieved from The Official Website of the City of New York: https://www1.nyc.gov/site/planning/planning-level/nyc-population/population-facts.page#:~:text=About%201%20in%20every%2038,arrived%20in%202000%20or%20later.