CSP 554 Big Data Technologies
Topic #8 - Team D
Project Report

Analysis of New York AirBnB

By:

1. **Tanmay Akhil Devikar (A20517094)**
2. **Pradeep Pal (A20497070)**
3. **Shekhar Kausal (A20512180)**
4. **Praveen Kumar Turpatiu (A20510960)**

Overview:

In order to investigate and extract insights from some intriguing data sets, we used a variety of big data techniques in this study. Data should be ingested, transformed, profiled, summarized, and visualized.

In order to gather information and provide answers on the listings, hosts, and locations, the goal of this project is to clean up and display the Airbnb New York dataset. Listing ID, host ID, host name, neighborhood group, neighborhood, latitude, longitude, kind of room, price, number of reviews, minimum nights, and availability 365 are among the details included in the collection.

With our data collection, we want to provide answers to and visualizations of a few research topics, such as how pricing is impacted by reviews, which area is most likely to be booked right away, and how highly rated reviews effect Airbnb availability.

I. Introduction:

Based in the US, Airbnb, Inc. is a firm that offers travel services and runs an online marketplace for housing, mostly homestays for holiday rentals. Airbnb receives a commission from each reservation; it does not really own any of the houses that are featured. The business was founded in 2008. The company's entire name, AirBedandBreakfast.com, is abbreviated to Airbnb. Recent years have seen an exponential increase in the popularity of Airbnb, which offers guests distinctive and reasonably priced lodging options while also allowing hosts to make money from their excess space. More than 7 million listings on Airbnb are available in more than 220 countries and regions as of 2022.

The purpose of this investigation is to investigate and purify the data in order to get further insight into the factors influencing Airbnb prices in New York City. We will use data cleaning techniques to eliminate missing values, duplicates, and outliers from the data and extract insightful information from it. Finally, we will display the data to identify patterns and trends within the dataset.

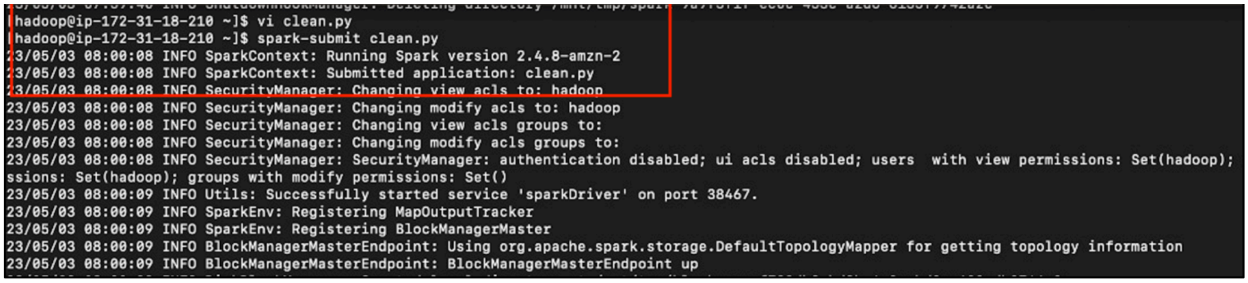
Project Tools:

- Amazon S3 bucket
- Apache Spark
- Jupyter Notebook with Python

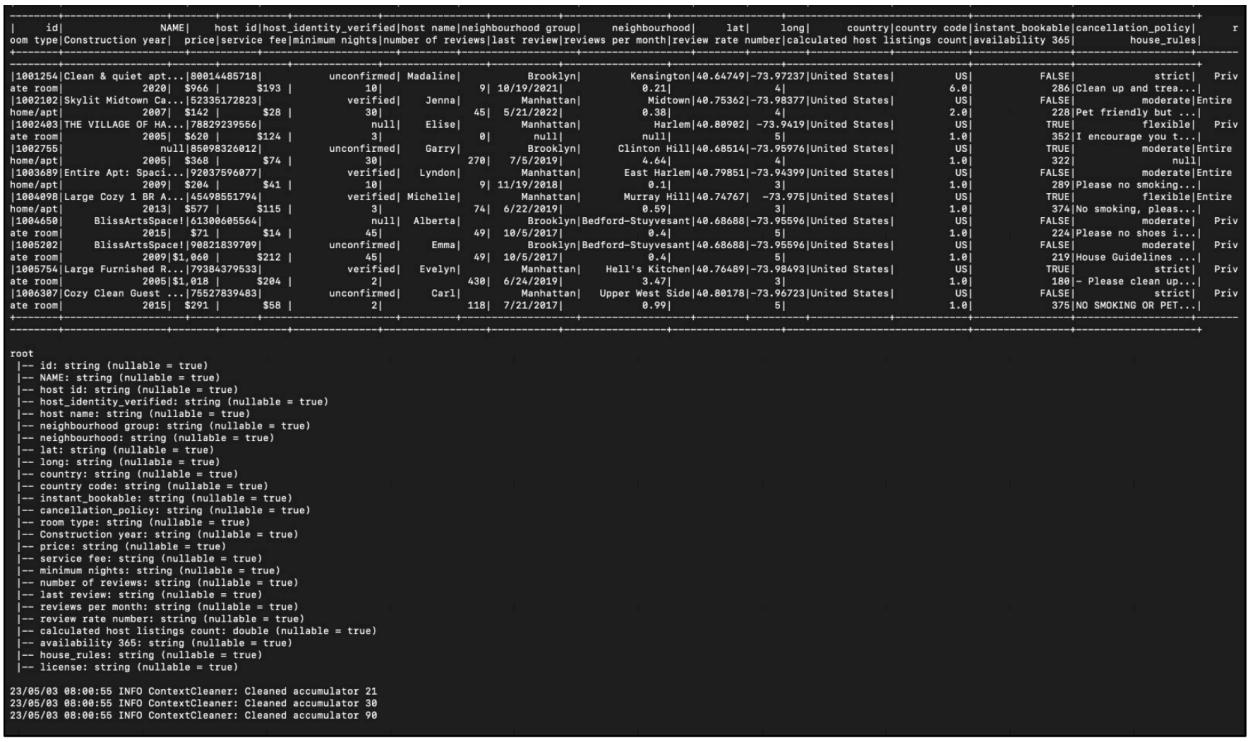
II. Data Operations and Process for Analysis:

- **Data Processing:** Using an Amazon S3 bucket, we converted the data into the necessary fields. After establishing a Spark session, we read the CSV dataset. We are using PySpark to profile the data once it has been read in order to use it for data analysis.

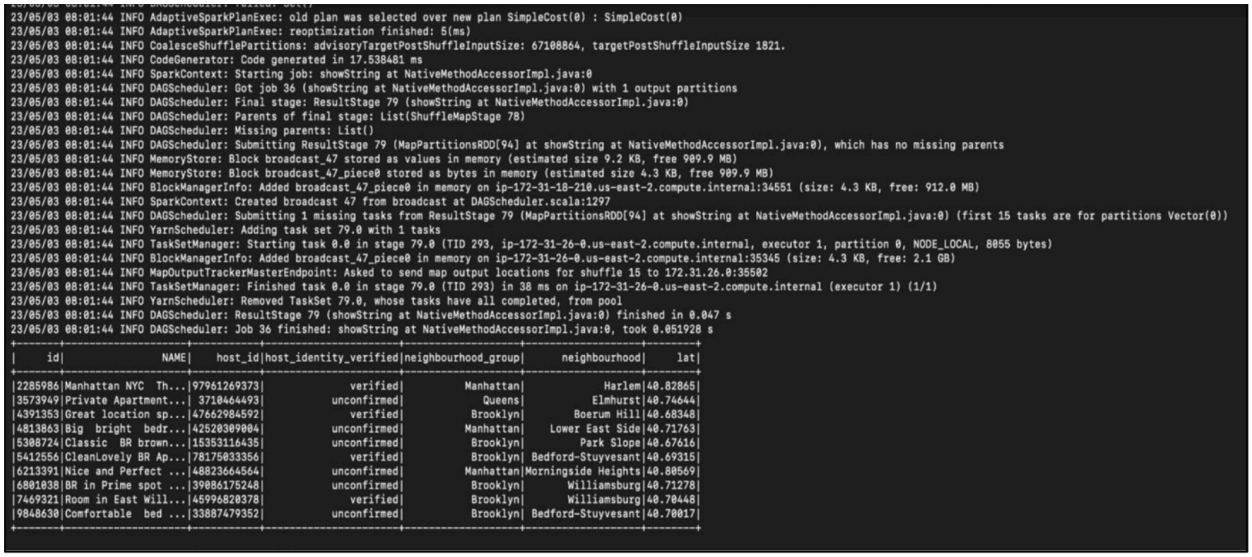
- **Data Cleaning**: The dataset has been cleaned by us. We've eliminated leading and trailing spaces in string columns, eliminated duplication, renamed columns to make them easier to read, and removed digits and punctuation from the 'NAME' column. Changing the $ and, in the pricing and service charge columns, as well as fixing and swapping out incorrect names.



Before cleaning the data:



After cleaning the data:



```
23/05/03 08:01:44 INFO AdaptiveSparkPlanExec: old plan was selected over new plan SimpleCost(0) : SimpleCost(0)
23/05/03 08:01:44 INFO AdaptiveSparkPlanExec: reoptimization finished: 5(ms)
23/05/03 08:01:44 INFO CoalesceShufflePartitions: advisoryTargetPostShuffleInputSize: 67108864, targetPostShuffleInputSize 1821.
23/05/03 08:01:44 INFO CodeGenerator: Code generated in 17.538481 ms
23/05/03 08:01:44 INFO SparkContext: Starting job: showString at NativeMethodAccessorImpl.java:0
23/05/03 08:01:44 INFO DAGScheduler: Got job 36 (showString at NativeMethodAccessorImpl.java:0) with 1 output partitions
23/05/03 08:01:44 INFO DAGScheduler: Final stage: ResultStage 79 (showString at NativeMethodAccessorImpl.java:0)
23/05/03 08:01:44 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 78)
23/05/03 08:01:44 INFO DAGScheduler: Missing parents: List()
23/05/03 08:01:44 INFO DAGScheduler: Submitting ResultStage 79 (MapPartitionsRDD[94] at showString at NativeMethodAccessorImpl.java:0), which has no missing parents
23/05/03 08:01:44 INFO MemoryStore: Block broadcast_47 stored as values in memory (estimated size 9.2 KB, free 909.9 MB)
23/05/03 08:01:44 INFO MemoryStore: Block broadcast_47_piece0 stored as bytes in memory (estimated size 4.3 KB, free 909.9 MB)
23/05/03 08:01:44 INFO BlockManagerInfo: Added broadcast_47_piece0 in memory on ip-172-31-18-210.us-east-2.compute.internal:34551 (size: 4.3 KB, free: 912.0 MB)
23/05/03 08:01:44 INFO SparkContext: Created broadcast 47 from broadcast at DAGScheduler.scala:1297
23/05/03 08:01:44 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 79 (MapPartitionsRDD[94] at showString at NativeMethodAccessorImpl.java:0) (first 15 tasks are for partitions Vector(0))
23/05/03 08:01:44 INFO YarnScheduler: Adding task set 79.0 with 1 tasks
23/05/03 08:01:44 INFO TaskSetManager: Starting task 0.0 in stage 79.0 (TID 293, ip-172-31-26-0.us-east-2.compute.internal, executor 1, partition 0, NODE_LOCAL, 8055 bytes)
23/05/03 08:01:44 INFO BlockManagerInfo: Added broadcast_47_piece0 in memory on ip-172-31-26-0.us-east-2.compute.internal:35345 (size: 4.3 KB, free: 2.1 GB)
23/05/03 08:01:44 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 15 to 172.31.26.0:35502
23/05/03 08:01:44 INFO TaskSetManager: Finished task 0.0 in stage 79.0 (TID 293) in 38 ms on ip-172-31-26-0.us-east-2.compute.internal (executor 1) (1/1)
23/05/03 08:01:44 INFO YarnScheduler: Removed TaskSet 79.0, whose tasks have all completed, from pool
23/05/03 08:01:44 INFO DAGScheduler: ResultStage 79 (showString at NativeMethodAccessorImpl.java:0) finished in 0.047 s
23/05/03 08:01:44 INFO DAGScheduler: Job 36 finished: showString at NativeMethodAccessorImpl.java:0, took 0.051928 s
+-------+--------------------+----------+--------------------+--------------------+-------------+-------+
|     id|                NAME|   host_id|host_identity_verified|neighbourhood_group|neighbourhood|    lat|
+-------+--------------------+----------+--------------------+--------------------+-------------+-------+
|2285986|Manhattan NYC  Th...|97961269373|            verified|           Manhattan|       Harlem|40.82865|
|3573949|Private Apartment...| 3710464493|          unconfirmed|              Queens|     Elmhurst|40.74644|
|4391353|Great location sp...|47662984592|            verified|            Brooklyn|  Boerum Hill|40.68348|
|4813863|Big  bright  bedr...|42620309004|          unconfirmed|           Manhattan|Lower East Side|40.71763|
|5308724|Classic  BR brown...|15353116435|          unconfirmed|            Brooklyn|    Park Slope|40.67616|
|5412556|CleanLovely BR Ap...|78175033356|            verified|            Brooklyn|Bedford-Stuyvesant|40.69315|
|6213391|Nice and Perfect ...|48823664564|          unconfirmed|           Manhattan|Morningside Heights|40.80569|
|6801038|BR in Prime spot ...|39086175248|          unconfirmed|            Brooklyn|   Williamsburg|40.71278|
|7469321|Room in East Will...|45996820378|            verified|            Brooklyn|   Williamsburg|40.70448|
|9848630|Comfortable  bed ...|33887479352|          unconfirmed|            Brooklyn|Bedford-Stuyvesant|40.70017|
+-------+--------------------+----------+--------------------+--------------------+-------------+-------+
```

- **Data Profiling:** To read the column formats in Spark, we changed their forms using data operations. In order to process and analyze the data, we renamed columns in the dataset. To obtain the most accurate results, handle missing or null values by substituting the mean and minimum averages of the data for the null values.

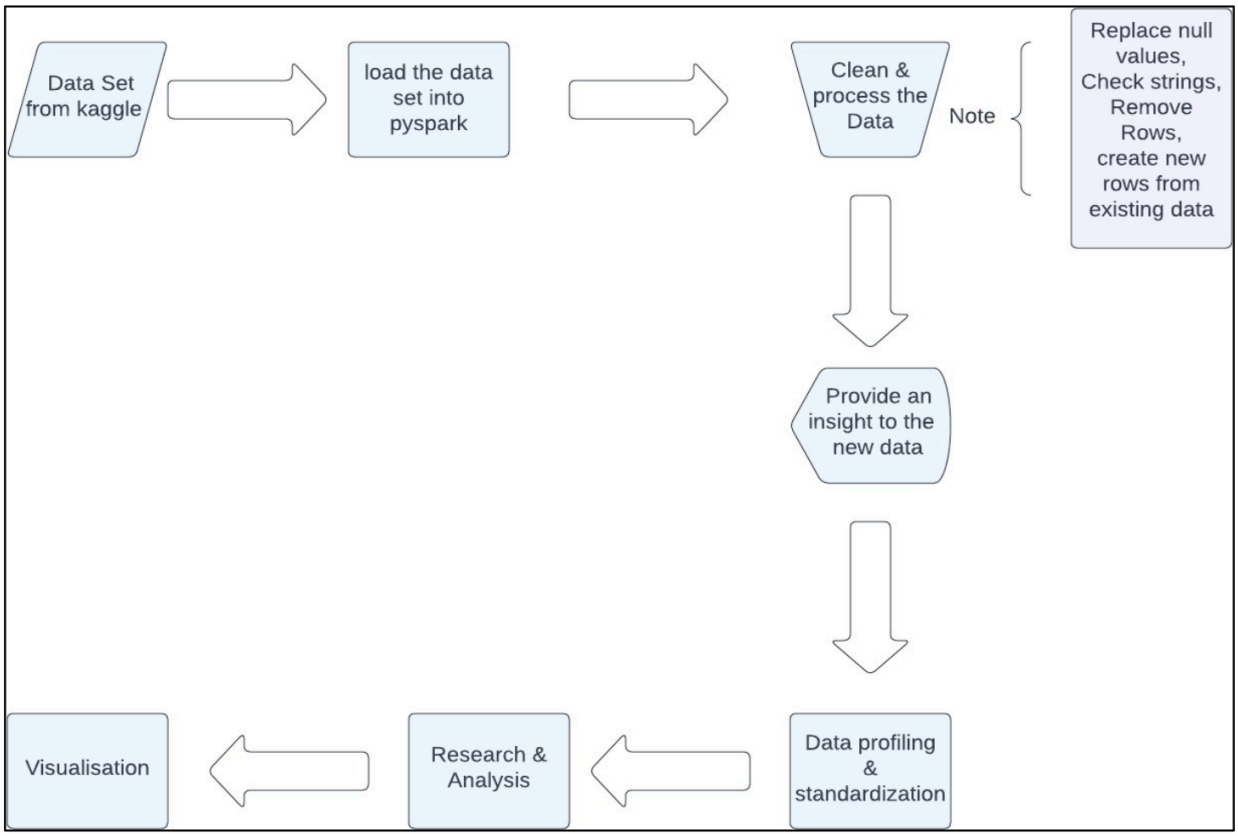
```
+---------+---------+------------------+------------------+
|min_price|max_price|        mean_price|       stddev_price|
+---------+---------+------------------+------------------+
|     50.0|   1200.0|626.0142138794353|331.65217328193256|
+---------+---------+------------------+------------------+

+---------------+---------------+------------------+------------------+
|min_service_fee|max_service_fee|  mean_service_fee|stddev_service_fee|
+---------------+---------------+------------------+------------------+
|            0.0|          240.0|124.87262856671809| 66.53877283181558|
+---------------+---------------+------------------+------------------+
```

```
+---------------+-----------------+
|      room_type|       avg(price)|
+---------------+-----------------+
|    Shared room|631.5299703264095|
|     Hotel room|666.3913043478261|
|Entire home/apt|624.5971085409253|
|   Private room|627.3187095421548|
+---------------+-----------------+
```

```
+-------------------+-----+
|neighbourhood_group|count|
+-------------------+-----+
|             Queens|11262|
|           Brooklyn|35264|
|      Staten Island|  834|
|          Manhattan|35262|
|              Bronx| 2295|
+-------------------+-----+
```

- **Data Analytics:** We carried out several procedures on the fields in order to obtain the necessary analysis. We looked over the information and conducted an analysis on it all.
- **Data Visualization:** To illustrate the historical data and provide an analysis for the New York AirBnB data, which has a high commuter count, we developed interactive charts and graphs. Using PySpark, we generated data frames, which we then utilized to draw graphs for the study.

Process diagram:



III. Comparison Of Tools:

PySpark: An Apache Spark Python API is called PySpark. It enables you to develop Spark applications in Python. Numerous data processing tasks, such as SQL, machine learning, and graph processing, are supported by PySpark.

- PySpark is an Apache Spark API that runs on Python. It enables you to develop Spark applications in Python.
- SQL, machine learning, and graph processing are just a few of the many data processing activities that it offers.
- Even for developers who are not familiar with Java or Scala, it is simple to understand and use.

Advantages:

- PySpark is an effective and adaptable large data processing tool.
- Even for developers who are not familiar with Java or Scala, it is simple to learn and use.
- PySpark has a sizable development community that provides strong support.

Disadvantages:

- Batch processing is one form of activity where PySpark might be sluggish.
- PySpark may use a lot of RAM.

Hive: Hive is an infrastructure for data warehouses developed on top of Apache Hadoop. It offers an interface for accessing Hadoop data that is similar to SQL.

- Hive is an Apache Hadoop-based data warehousing system.
- Hive offers a querying interface for Hadoop data similar to that of SQL.
- Even for developers who are unfamiliar with Hadoop or MapReduce, Hive is simple to understand and utilize.

Advantages:

- The Hive data warehouse infrastructure is strong and expandable.
- Even for developers who are not familiar with Hadoop or MapReduce, it is simple to understand and utilize.
- There is a sizable development community that supports Hive effectively.

Disadvantages:

- Certain tasks, such interactive querying, may cause Hive to lag.
- Hive may use a lot of RAM.

Scala: Scala is a general-purpose programming language with a focus on scalability and fast performance. Because it is compiled, it operates more quickly than interpreted languages like Python. Because Scala is a functional programming language, data processing activities are a good fit for it.

- Scala is a general-purpose programming language with a focus on scalability and fast performance.
- Scala is quicker than interpreted languages like Python since it is a compiled language.
- Because Scala is a functional programming language, data processing activities are a good fit for it.

Advantages:

- Scala is a very strong and flexible programming language.
- It is fast and scalable.
- It is well-suited for data processing tasks.

Disadvantages:

- For developers who are unfamiliar with functional programming, learning Scala might be challenging.
- Scala may use a lot of RAM.

IV. Observations and Analysis:

AirBnB Dataset : We can see the top 10 rows of the dataset with attribute headers and its schema just after loading it through the file.

id	NAME	host_id	host_identity_verified	neighbourhood_group	neighbourhood	lat
1210105	ALL ABOUT A VERY ...	42067422922	verified	Brooklyn	Fort Greene	40.69088
1244348	City Room Semi P...	39254540312	unconfirmed	Manhattan	Harlem	40.81156
1247110	HarlemHamilton He...	58292705877	unconfirmed	Manhattan	Harlem	40.82426
1249319	Entire Apt in Hea...	49315683762	verified	Brooklyn	Williamsburg	40.71534
1252080	Red Room for two ...	43465301578	verified	Brooklyn	Bedford-Stuyvesant	40.6798
1398992	Prime Williamsbur...	93988661910	verified	Brooklyn	Williamsburg	40.72059
1474105	Furnished Bedroom...	67288825352	verified	Manhattan	Harlem	40.80637
1575728	Lovely bdrm in P...	76284357302	verified	Brooklyn	Prospect Heights	40.67763
1638138	Lovely BR Midtow...	73838785546	unconfirmed	Manhattan	Midtown	40.75632
1688950	BR Village day...	60729843423	verified	Manhattan	Greenwich Village	40.7311

Schema of the dataset after we did couple of modifications to it:

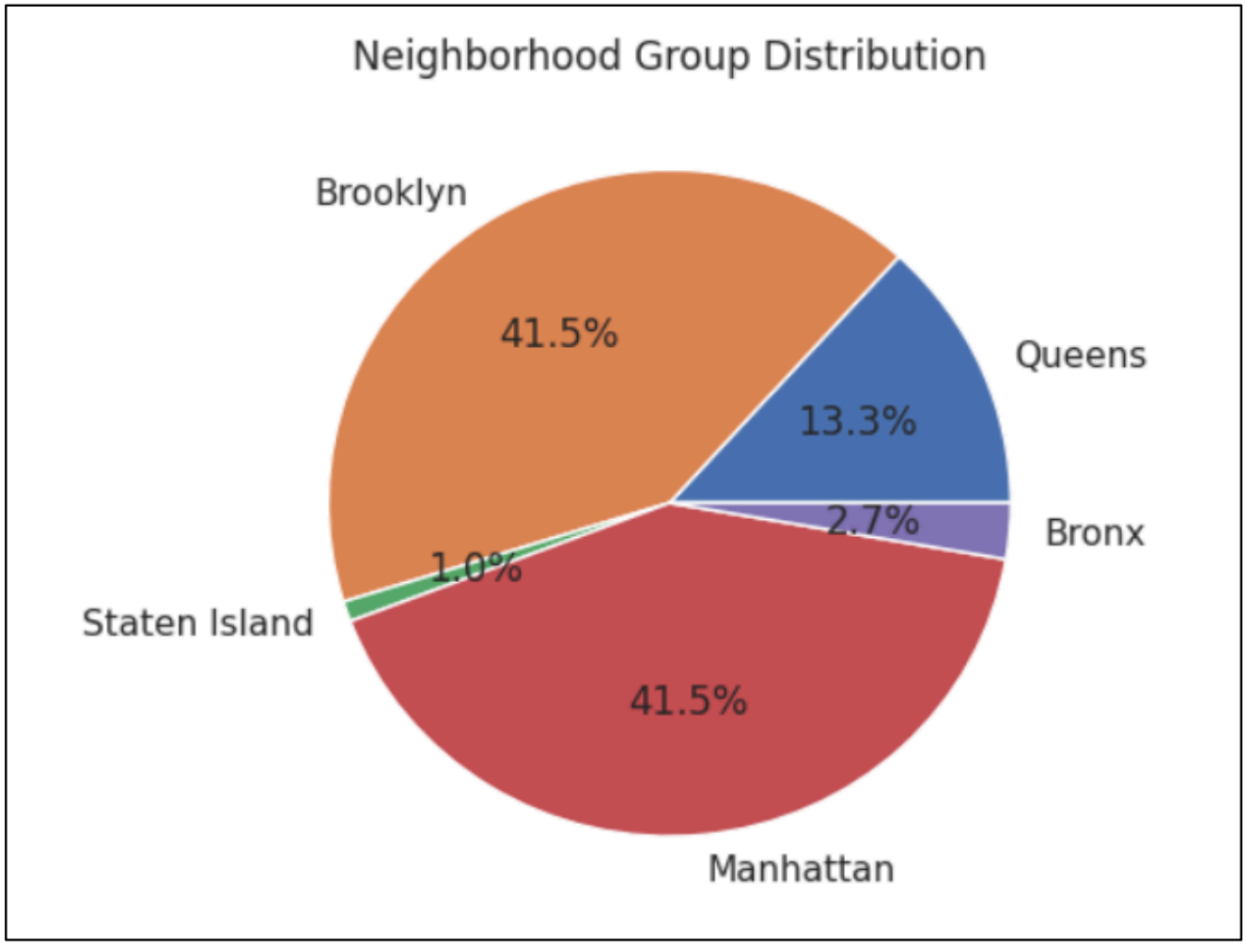
```
root
 |-- id: string (nullable = true)
 |-- NAME: string (nullable = true)
 |-- host_id: string (nullable = true)
 |-- host_identity_verified: string (nullable = true)
 |-- neighbourhood_group: string (nullable = true)
 |-- neighbourhood: string (nullable = true)
 |-- lat: double (nullable = true)
 |-- long: double (nullable = true)
 |-- instant_bookable: string (nullable = true)
 |-- cancellation_policy: string (nullable = true)
 |-- room_type: string (nullable = true)
 |-- construction_year: integer (nullable = true)
 |-- price: double (nullable = true)
 |-- service_fee: double (nullable = false)
 |-- minimum_nights: integer (nullable = false)
 |-- num_reviews: integer (nullable = true)
 |-- last_review: string (nullable = true)
 |-- reviews_per_month: double (nullable = false)
 |-- review_rate_num: double (nullable = false)
 |-- calculated_host_listings_count: double (nullable = false)
 |-- availability_365: integer (nullable = false)
```

V. VISUALIZATION OF THE NYC AirBnB DATASET:

To investigate the connections between the various variables in the dataset, we employed variety of visualization tools. We were able to discern patterns and trends in the data and learn more about the features of the Airbnb industry thanks to these visual aids.

1. Neighborhood group distribution:

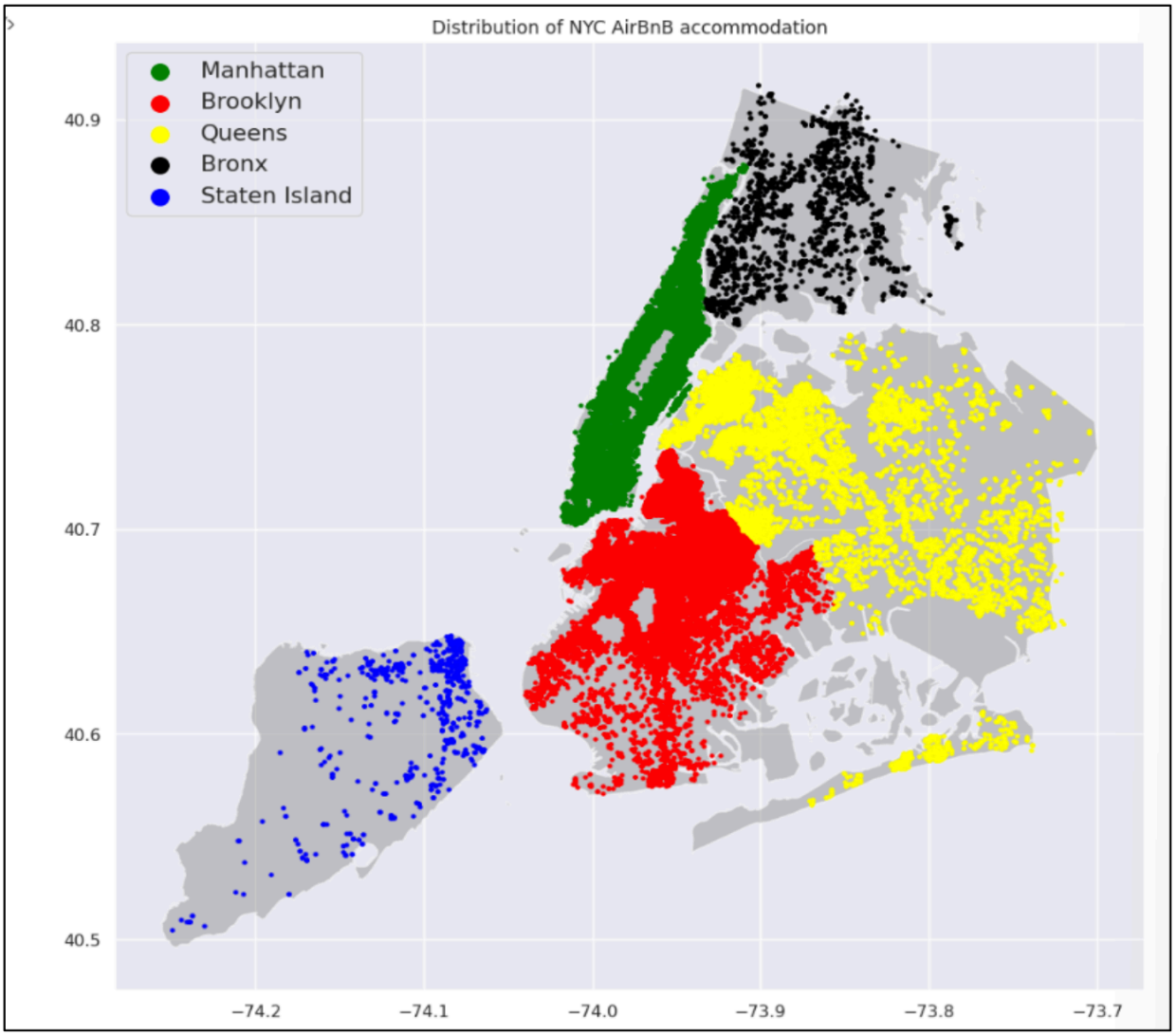
This pie chart labels the percentage of listings in each neighborhood category and display the distribution of listings by group.
Staten Island has the lowest percentage of listings in New York, whereas Brooklyn and Manhattan account for 62% of the total.



2. Density of Airbnb listings across neighborhoods:

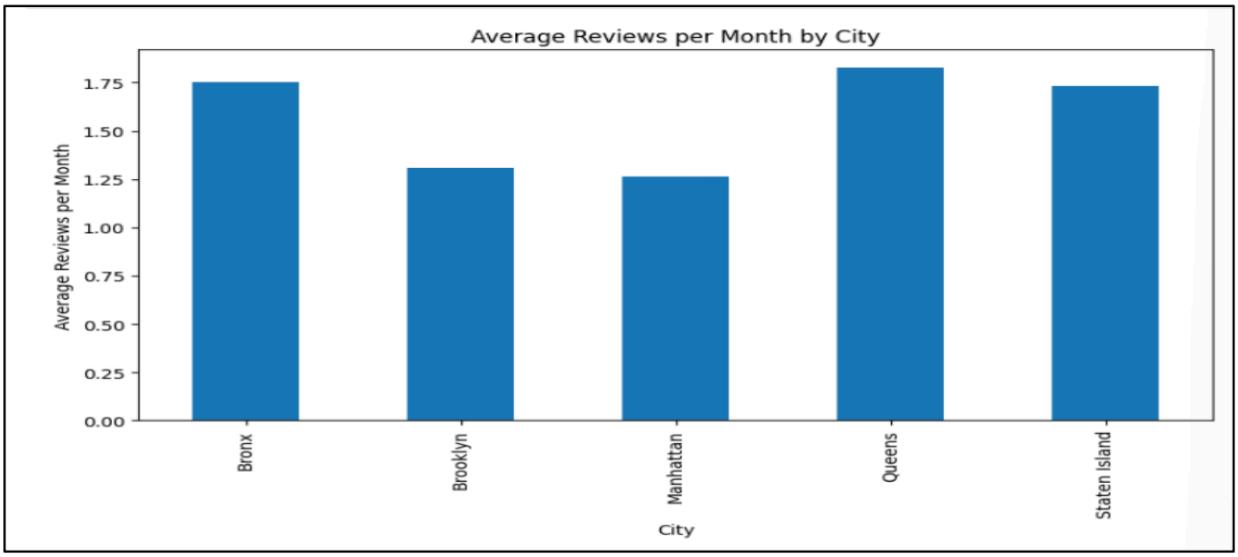
Using the seaborn library, this graph creates a 2D kernel density plot of the latitude and longitude coordinates of Airbnb listings. The resultant figure displays the density of Airbnb listings in the selected city's various neighborhoods, with places with greater densities indicated by warmer hues. The matplotlib.pyplot module is used to show the plot and assign it a title.

The following diagram shows the listing of New York according to the neighborhood type:



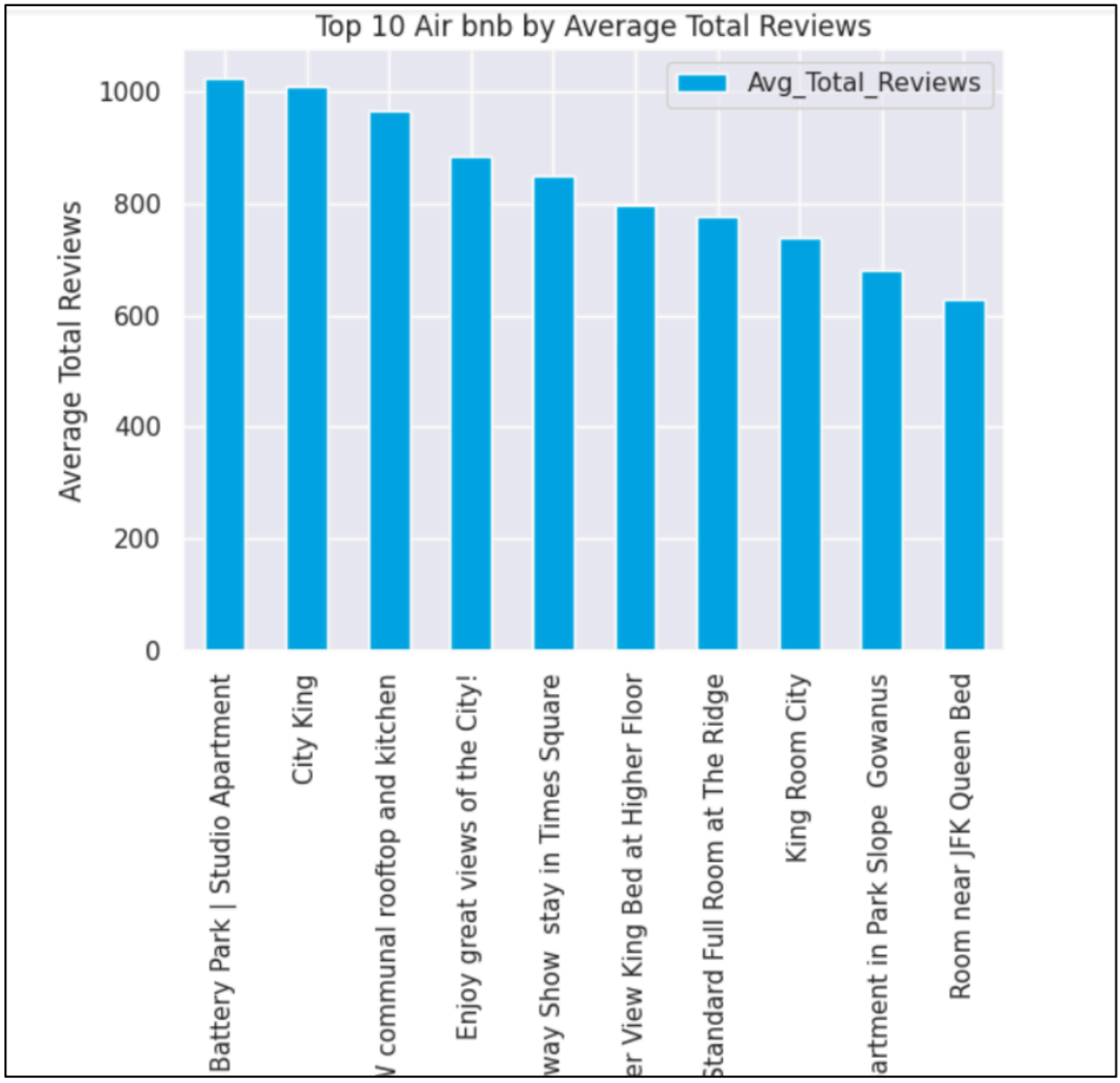
3. Reviews per month by city:

The analysis of the data results in the graph below, which compares the average monthly reviews to the neighborhood group. This shows each city in the dataset's average monthly review count. Knowing whether a listing has been more popular over time rather than just during certain seasons of the year is helpful.



4. Top 10 hosting places vs Average Total Reviews, Top 10 places here are taken based on the average total reviews for each place.

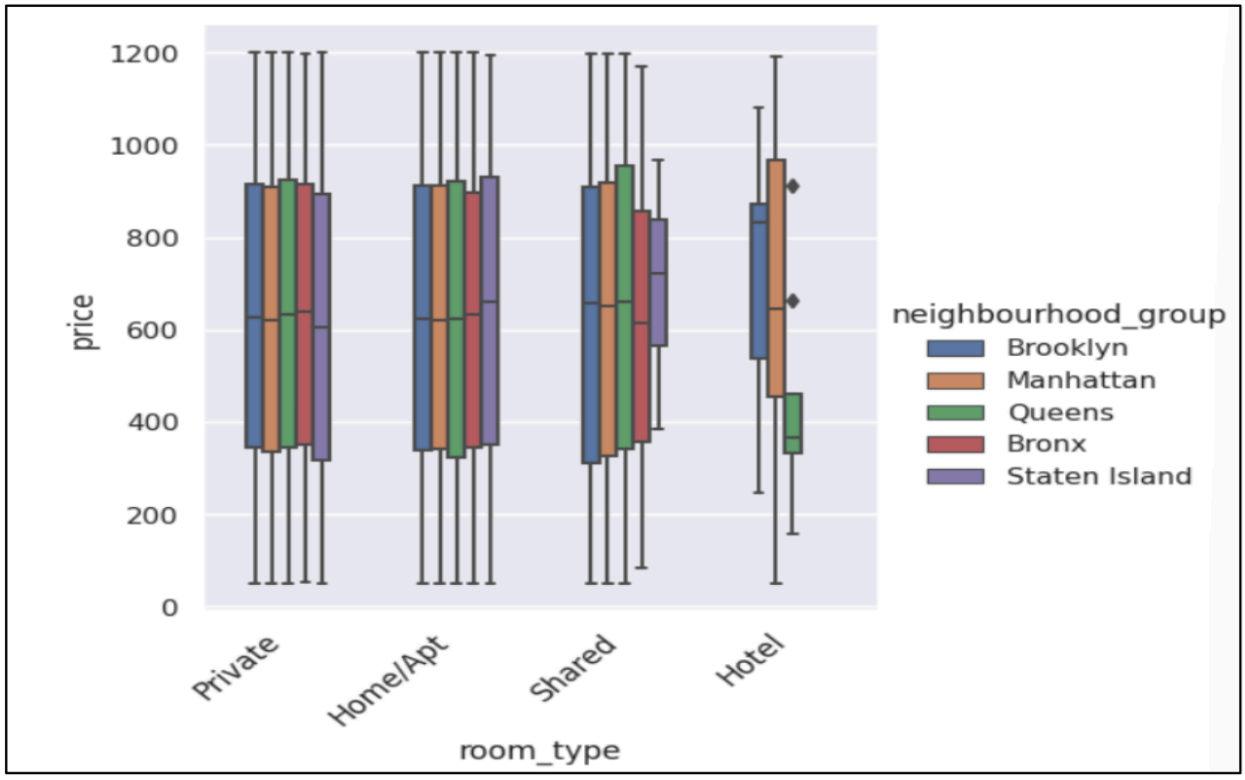
The top 10 hosting locations are shown in the graph below, which is entirely based on average evaluations. The studio unit from Battery Park is the highest. Over a thousand evaluations are averaged overall.



5. Price vs room type for different neighborhood:

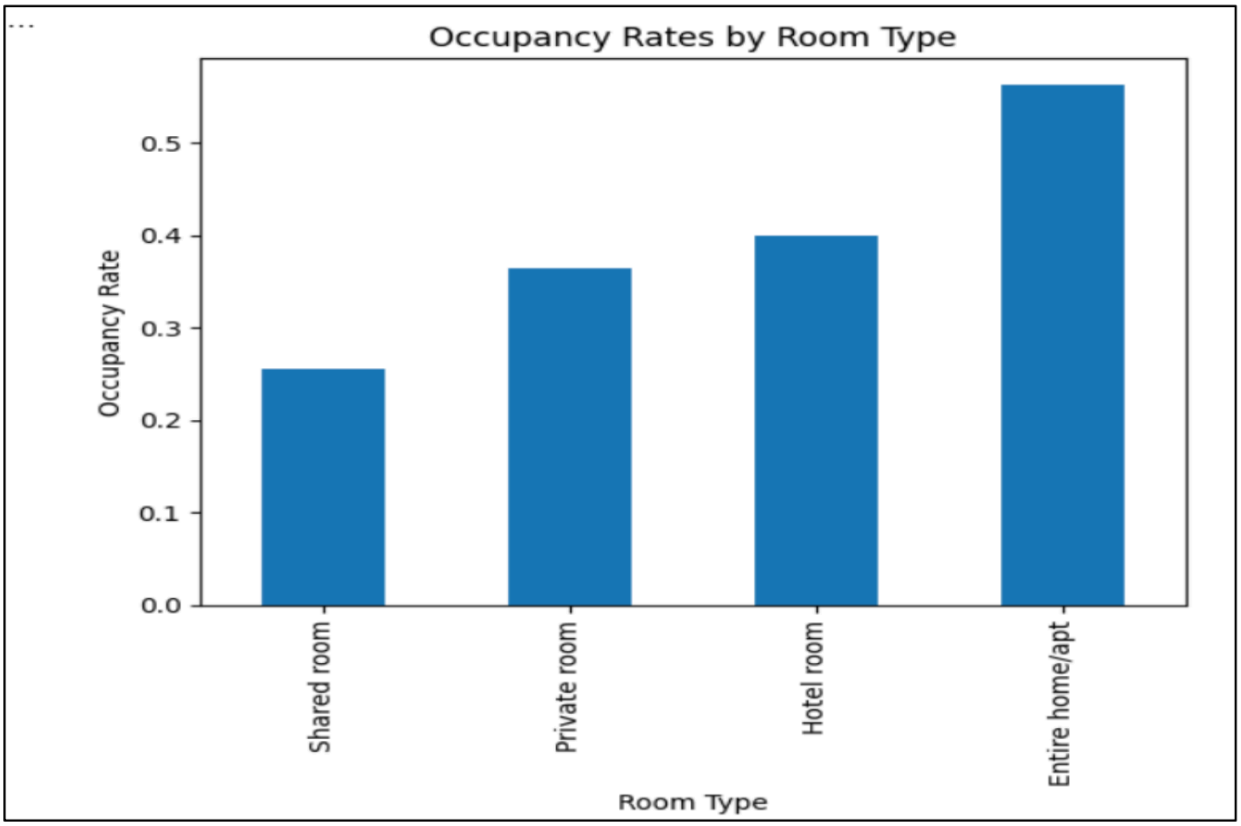
Displaying the price distribution of various room kinds in different neighborhoods within a given dataset. The color parameter organizes the data by neighborhood, the x-axis displays the kind of room, and the y-axis indicates the price.

The data indicates that the minimum and maximum pricing for private and apartment spaces are consistent across all areas. Staten Island, on the other hand, has high ratings from the above radar map but a short range in shared rooms. It follows that queens have the cheapest hotel rooms.



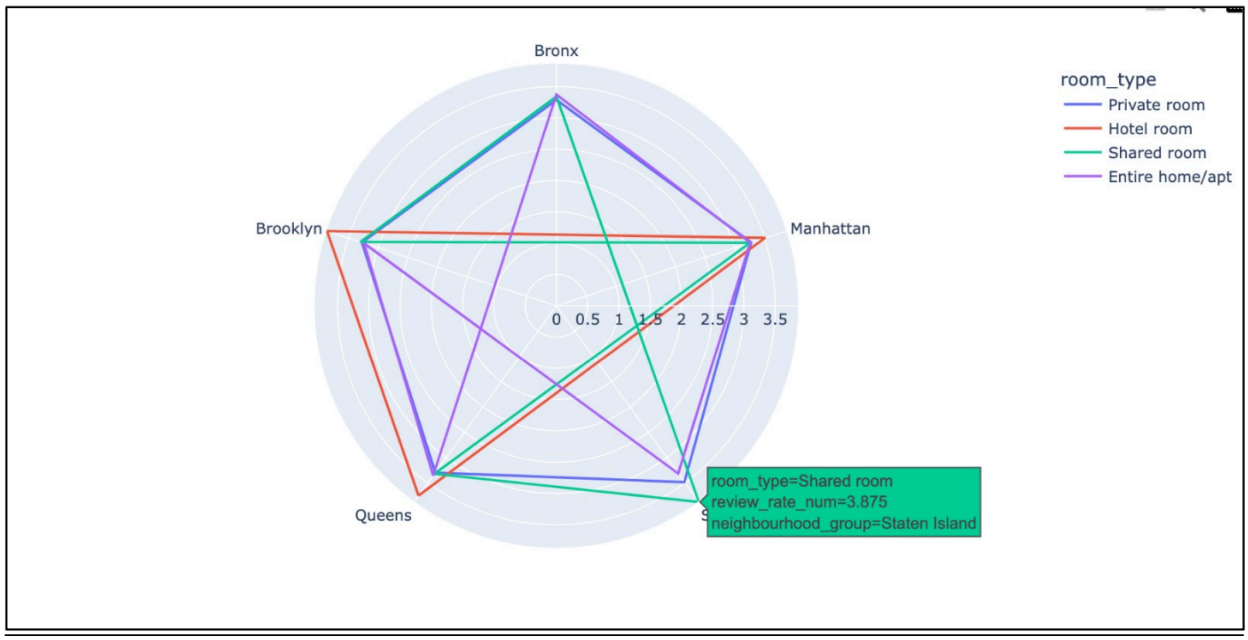
6. Occupancy rates by room type:

The analysis of the data is displayed in terms of occupancy rates for every kind of property in the dataset. It sorts the data by type of property, determines the mean occupancy rate for each category, then computes the occupancy rate for each listing based on the quantity of reviews and nights stayed. Based on the data we have, we may infer that people favor a certain listing based on the number of reviews and nights stayed. The entire house or flat appears to be at the top of this property.



7. Ratings per neighbourhood type:

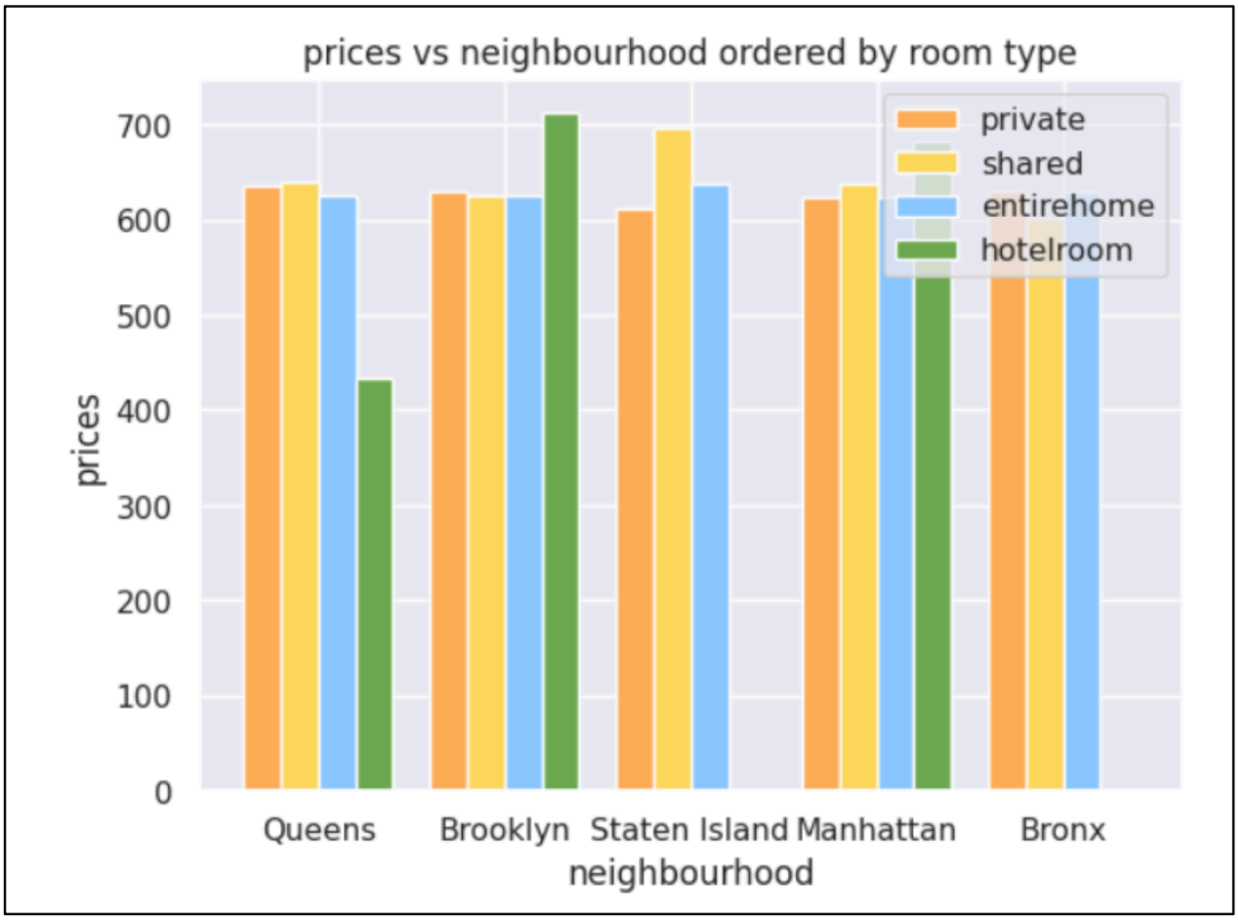
We may calculate ratings according to the kind of neighborhood with the aid of this radar plot. According to the narrative, hotel rooms are often thought to be more popular in Brooklyn and Queens, although shared rooms are highly regarded alone in Staten Island. It appears from the Bronx listings that the average rating for each kind of lodging is 3.0. For hotel rooms, Brooklyn has the highest rating of 4.0.



8. Prices vs neighbourhood ordered by room type:

In the bar plot below, the data displays an analysis of the neighborhood and pricing sorted by kind of accommodation.

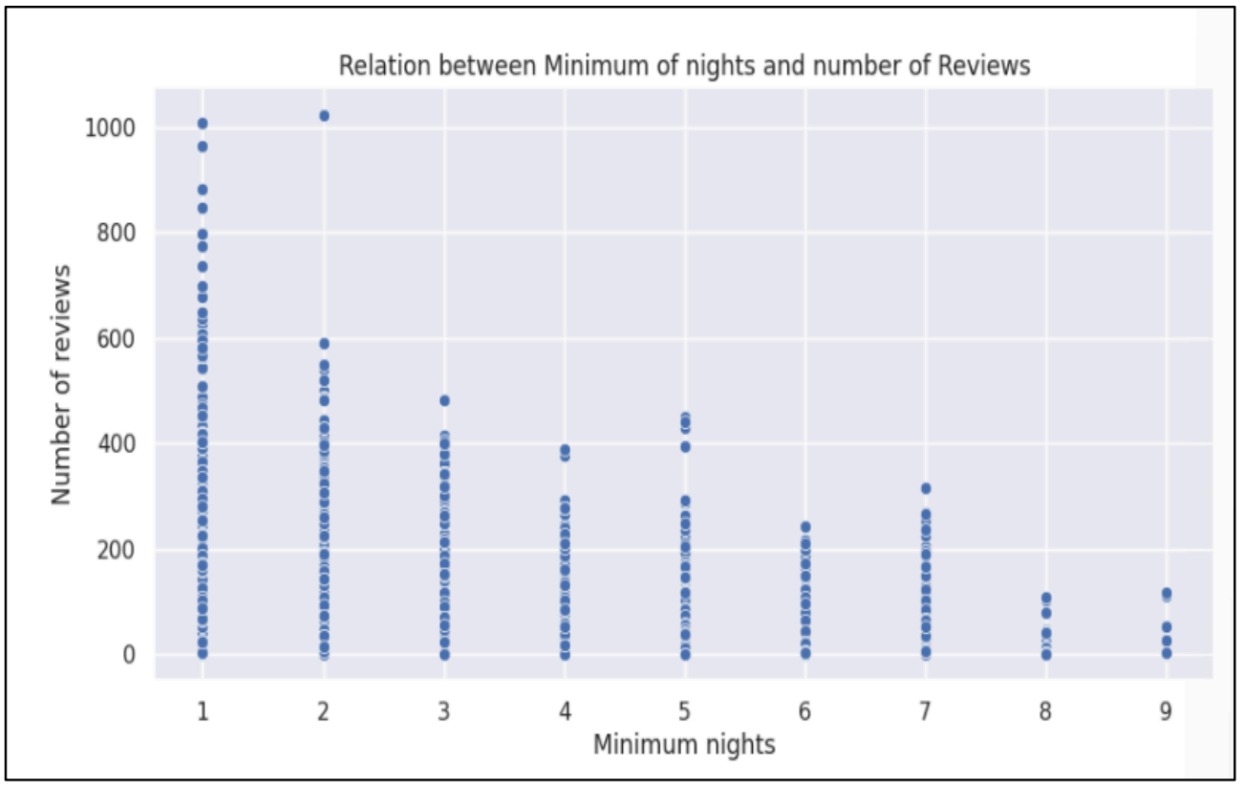
The storyline suggests that hotel rooms in Brooklyn are the most expensive, while those in Queens are the least expensive.



9. Minimum number of nights vs number of reviews:

In order to illustrate the relationship between these two variables, this code first filters values that fall outside the upper and lower bounds of the interquartile range (IQR) from a Pandas Data Frame in order to remove outliers. The remaining data points are then plotted in a scatter plot with'minimum nights' on the x-axis and 'num_reviews' on the y-axis. In addition, the plot has labels for the x, y, and title axes.

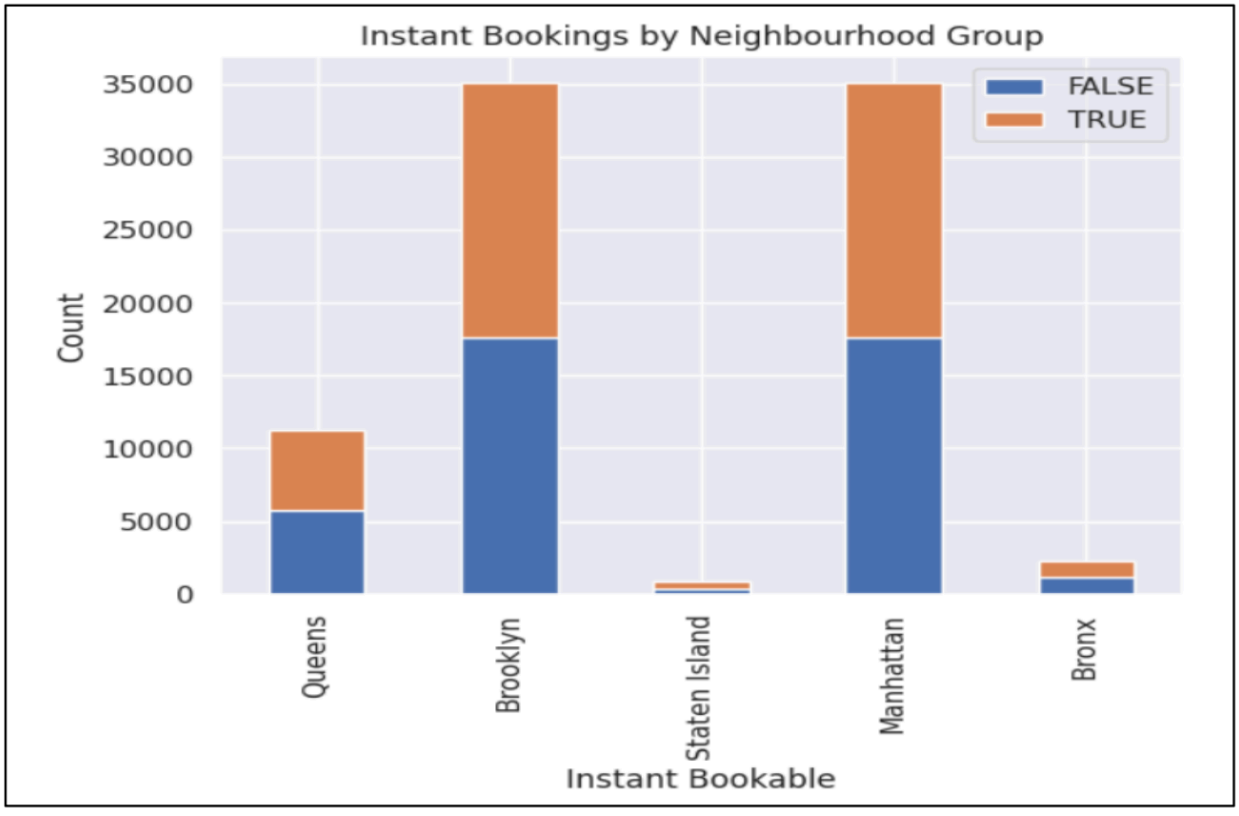
This graph suggests that the majority of rooms are occupied during shorter stays. The highest sort after minimum nights is 1.



10. Instant bookings vs neighborhood group:

This diagram shows and groups data by neighbourhood group and instant bookable as true or false. X axis shows the total no of listings in the neighborhood , Y axis shows different neighborhood types.

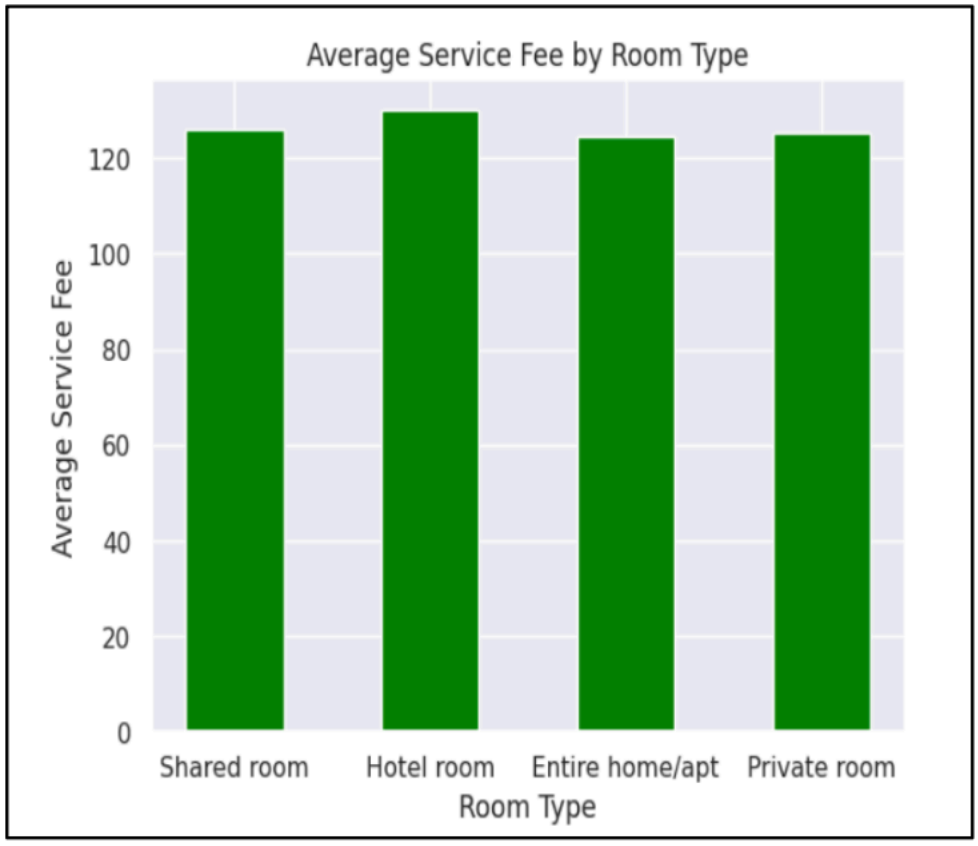
The graph infers out Manhattan and Brooklyn has equal proportion of instant bookable comparatively



11. Average service fee by room type:

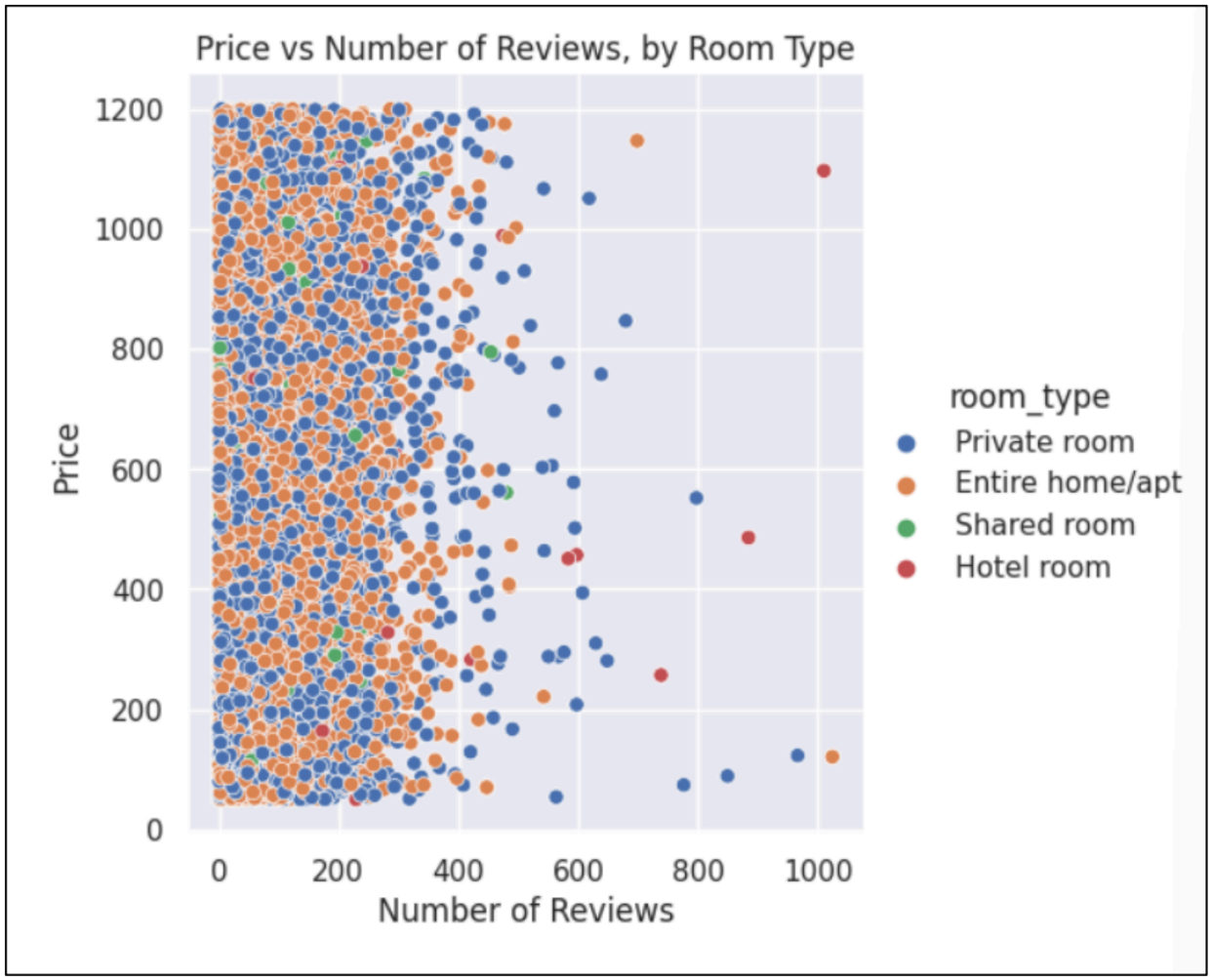
The dataset's first bar chart showing the average review score by neighborhood. The neighborhood group is displayed on the x-axis, while the average review score is displayed on the y-axis.

Given that hotel rooms often have more facilities than other types of rooms, we may assume that the service price is higher there.



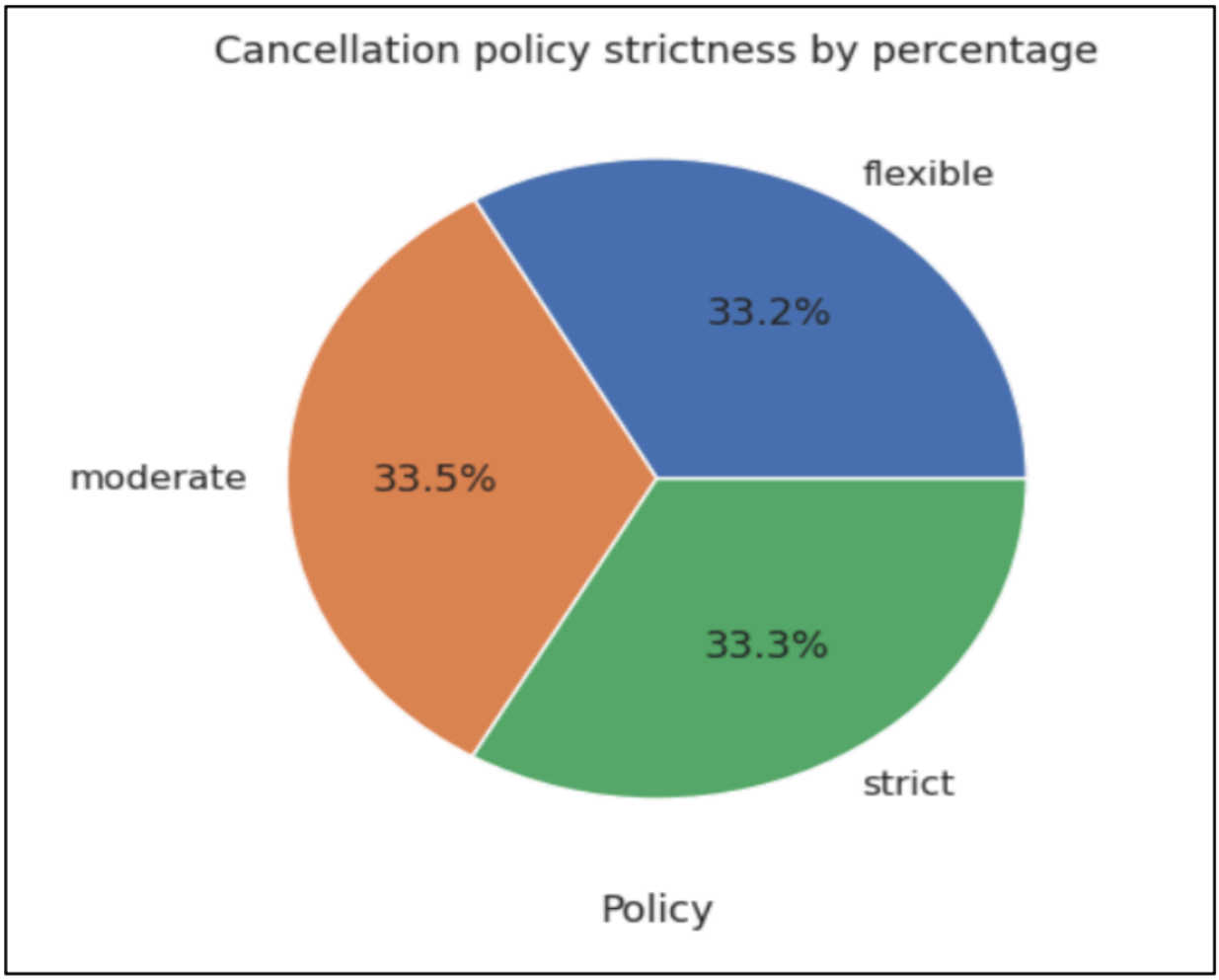
12. Price vs number of reviews by room type:

Below is the price vs number of reviews scatter plot for every room type differentiated based on color.



13. Cancellation policy strictness:

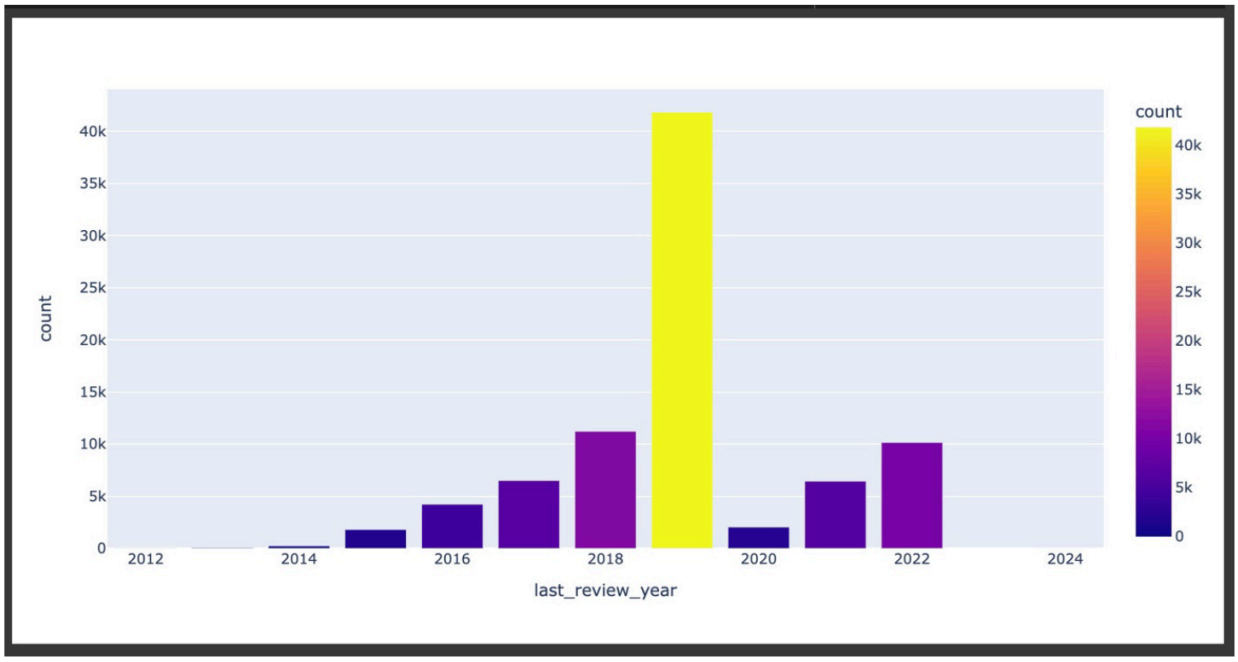
The data is analyzed by 'cancellation_policy' in the pie chart below, which also counts the number of 'id' for each policy. The proportion of each type of insurance is displayed in the resultant chart.



14. Yearly based count :

This graph depicts yearly based number of reviews for the Air BnB's with x-axis as count of reviews and y-axis is year of the review.

We can infer the growing trends in booking the AirBnB until 2019 and with a sudden drop in 2020 probably due to covid and it started picking up in 2022.



VI. <u>Conclusion</u>:

To sum up, this project included cleaning and presenting a dataset of New York-based Airbnb listings. Thanks to the data cleaning process, we were able to handle missing numbers, eliminate duplicates, and correct data discrepancies.

Through exploratory data analysis and data visualization, we were able to discover more about the features of the posts and how they connect to other variables like location, property type, and price. Homes and flats were the two most popular forms of real estate, and we found that prices varied significantly based on the location, number of bedrooms, and amenities offered.

We also found that this project might have a lot more uses in the future, such as sentiment analysis, time-series analysis, user segmentation, and geographical analysis.

All things considered, this research provided helpful information on the various trends and patterns in the Airbnb business. Both hosts and visitors may utilize these facts to enhance their listings and make more informed hotel choices.

VII. Future Scope:

Based on the analysis and data cleansing completed for this project, the following future scopes are possible:

1. **Time series analysis**: To spot patterns and seasonality in the Airbnb market, you may examine the data over an extended period of time. In light of seasonal demand, this can assist hosts in better understanding how to price their properties.

2. **Sentiment analysis**: Analyzing data on customer satisfaction to identify trends and patterns. This might help owners of Airbnb update their listings and improve the overall experience for their guests.

3. **Spatial analysis**: By analyzing the data using geospatial tools, we can determine the relationship between pricing and tourism destinations. Finding Airbnb activity hotspots may be aided by this.

VIII. References:

1. Python : https://docs.python.org/3/library/
2. HDFS Architecture Guide : https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html [3]
3. PySpark : https://spark.apache.org/docs/latest/api/python/
4. AWS S3 : https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html [5]
5. Data set : https://www.kaggle.com/code/thecansin/eda-and-data-visualization-ny-airbnb/

6. Python Library : https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.violinplot.html https://seaborn.pydata.org/generated/seaborn.pairplot.html https://seaborn.pydata.org/generated/seaborn.lmplot.html

7. Plotly : https://plotly.com/python/
8. Scala Documentation : https://www.baeldung.com/scala/scaladoc
9. Hive Documentation : https://cwiki.apache.org/confluence/display/HIVE