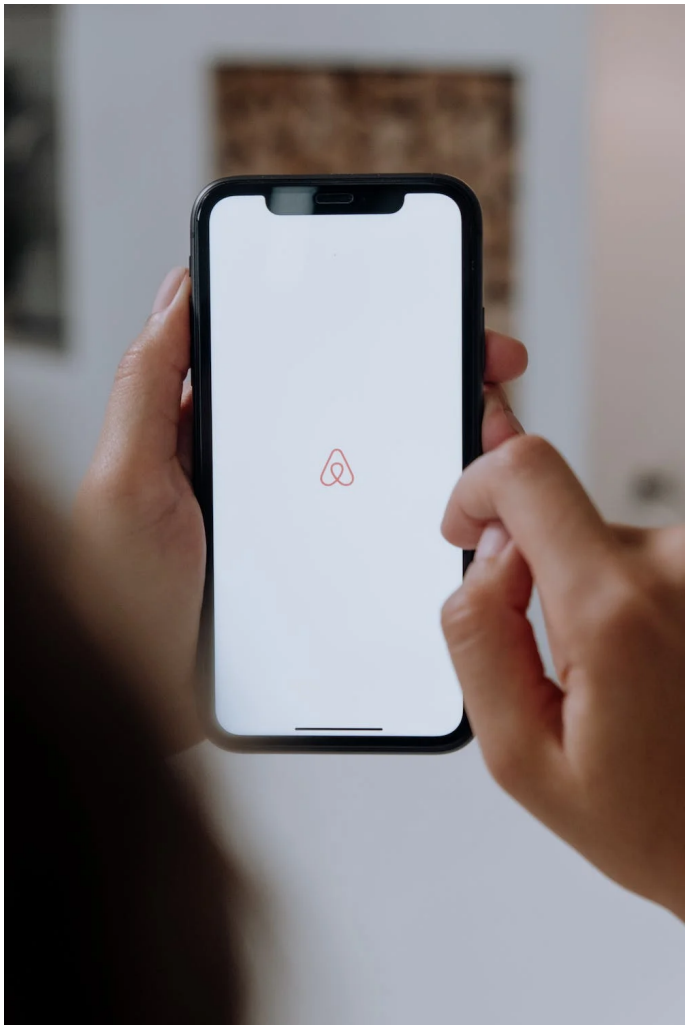
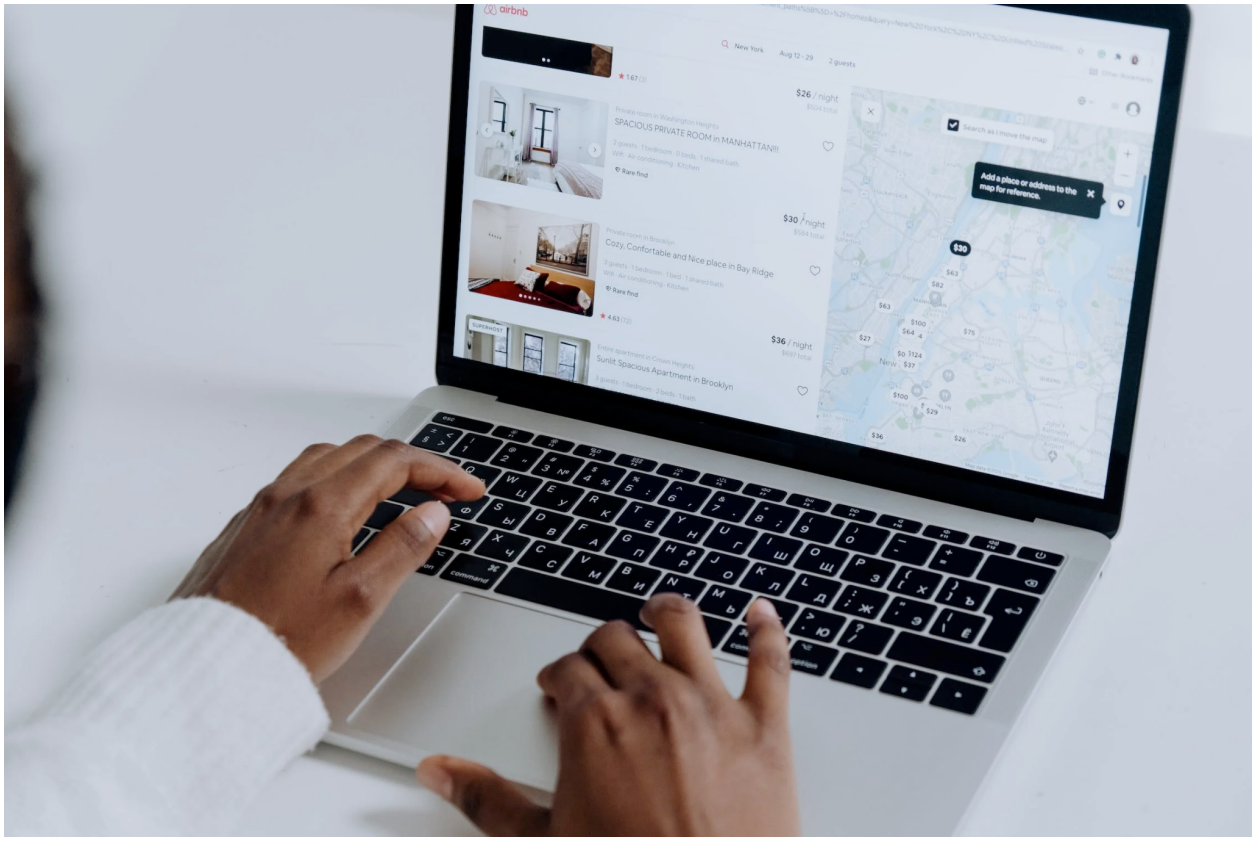


Airbnb: From ETL to Insights

Team Members:

1. EVELYN TAN-ROGERS
2. SEOW KHENG CHUNG
3. GOH BOON HENG

1. Introduction



This report documents the work of our group, **Dream Team**, where we used publicly available data from Inside Airbnb, an advocacy group, to replicate the ETL (Extract, Transform, Load) process typical of data engineering. To draw insights from our data, we also carried out data analysis and visualisation tasks.

1.1 Background

Airbnb is a global platform that allows users to list and discover accommodations and experiences in over 100,000 cities and towns all over the world. They launched in Singapore in 2012; at the time, it was estimated that there were over 600,000 registered members in the country. Despite a strong start, Airbnb soon encountered regulatory challenges, due to Singapore's strict housing and subletting guidelines.

Currently, Airbnb listings within Singapore must reflect the minimum rental period of three consecutive months of stay. However, Singapore-based users are free to use the Airbnb app to book vacation homes elsewhere, for any length of stay.

1.2 Why Airbnb?

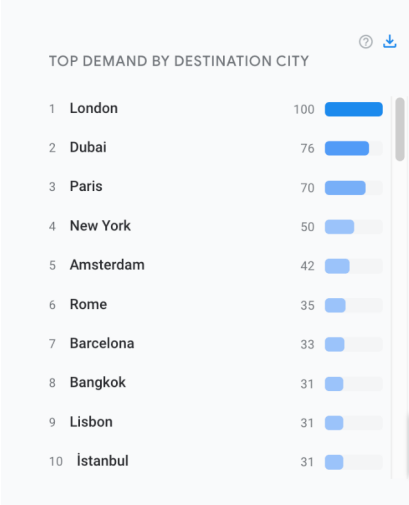
We sought ready-made data for this project, due to the limited time allocated for project work. After a brief discussion about possible options, we opted to use Airbnb data for these reasons:

- Airbnb's data is collected by an advocacy group called Inside Airbnb, and free data sets are available for public use.
- The data files are organised by city, providing a quick way to select preferred cities for study.
- The data is easy to understand—there is a data dictionary for quick reference. Those interested in the methodology can refer to the data assumptions page.
- Working with Airbnb data gives us a taste of what it's like to handle big data for a consumer brand.
- As travellers ourselves, we can formulate insightful questions about the data, which are not only relevant for travellers, but also for Airbnb and its competitors, e.g. hotels.

1.3 Which Cities to Focus On?

Initially, we had planned to collect Airbnb data for 10 cities, but after considering our time limitations, we decided to narrow this to five cities.

For a quick way to filter cities by popularity, we used Google's Destination Insights. At the time of checking, these were the most popular destinations for travellers worldwide:

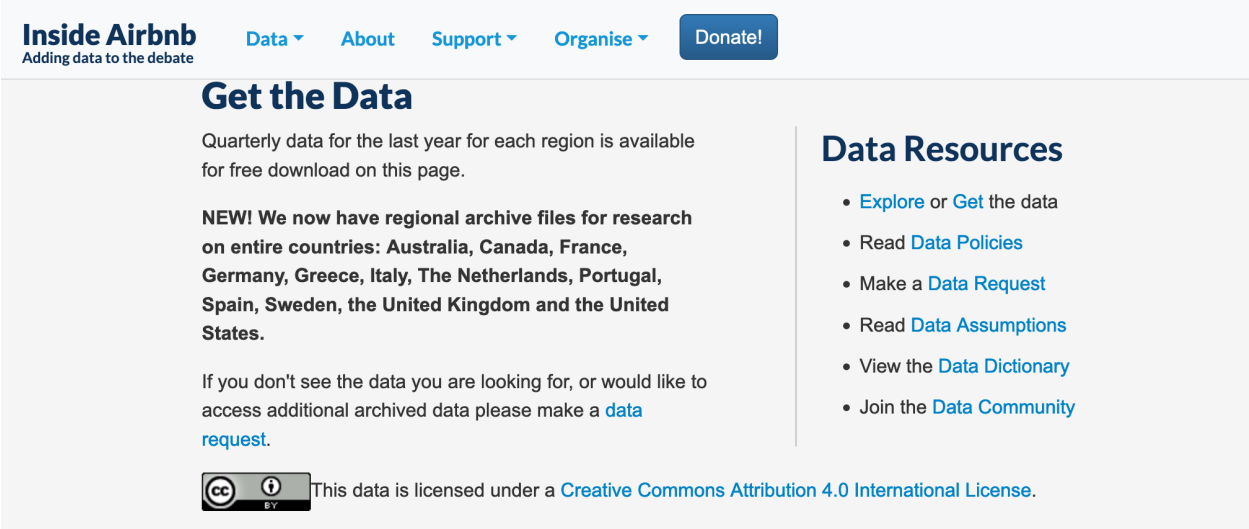


We looked specifically at "Top Demand by Destination City," and checked to see if Inside Airbnb had data for the top five cities listed. As data for the second most popular city (Dubai) was not available, we used data for the following cities:

1. London
2. Paris
3. New York
4. Amsterdam
5. Rome

In the following sections, we will detail the steps that we took to select, clean, transform, and analyse our data.

2. Selecting Data Sets



Before brainstorming for our data questions, we looked at existing projects for ideas. Some examples:

- Airbnb Sentiment Analysis with Python: This project made use of the Python library NLTK (Natural Language Toolkit), which includes various natural language processing (NLP) tools and resources. The author scored the reviews, presented the most frequent words, and highlighted interesting data findings as well as inaccuracies.
- Insights Behind Airbnb Customer Reviews: This project made use of VADER (Valence Aware Dictionary and sEntiment Reasoner), a sentiment analysis tool and library for analysing and categorising text sentiment, primarily in English. VADER is available as a Python library and is part of the NLTK library. The project seeks to answer these questions: Do people tend to leave comments about bad reviews or good reviews? Do sentiments differ by neighbourhoods? Are positive sentiments correlated with other features or metrics?

2.1 What Do We Want to Find Out?

From our discussions, we generated questions that businesses and Airbnb customers would be interested in.

These are our primary questions, which we have used Python to answer:

1. For each city, are reviews more positive or negative?
2. What are the most frequent words for the reviews? What can this tell us about consumer concerns?

3. Is there a correlation between price and review sentiment? (E.g. Do reviews tend to become more positive or more negative when prices increase?)
4. Which city has the highest rating scores—overall, and for specific factors such as cleanliness and location?

These are our secondary questions, which we have used Python (*See appendix I*) and Tableau Public to answer:

1. Do more experienced Airbnb hosts get more positive or negative reviews?
2. For Airbnb Superhosts, are their reviews more positive or negative? ("Airbnb Superhost" is a designation given to experienced and highly rated hosts on the Airbnb platform, who meet certain criteria set by Airbnb.)
3. For Airbnb hosts who don't verify their identity, does this impact their rating?
4. Which types of homes receive the most positive reviews?

2.2 Which Data Fields Are Relevant?

To answer our data questions, we selected two Airbnb data sets for each city, known as "listings" and "reviews".

Of the 75 fields in the "listings" data, this is what we wanted to use:

1. id (the unique id for each listing)
2. name (name of the listing)
3. host_since
4. host_location
5. host_is_superhost
6. host_identity_verified
7. property_type
8. price
9. review_scores_rating
10. review_scores_accuracy (does the actual home match its description?)
11. review_scores_cleanliness (how clean is the home?)
12. review_scores_checkin (how smooth is the check-in process?)
13. review_scores_communication (is it easy to communicate with the host?)
14. review_scores_location (is the location good for travellers?)
15. review_scores_value (is the home value for money?)

In our analysis, we used the **host_location** field as a substitute for a "city, country" field. The reason for this: in our data sets, location information was stored in the form of longitude and latitude coordinates.

There were two limitations to our approach—hosts were not necessarily in the same location as their rental apartments, and host_location data did not always conform to the "city, country" format. Since we had ample data, we decided to drop non-conforming data rows.

Of the 6 fields in the “reviews” data, this is what we wanted to use:

1. listing_id
2. date
3. comments (review text)

3. Data Preparation & Upload To PostgreSQL

This section will detail the steps taken to extract, transform, and load our data.

3.1 Where to host the data?

We did consider using an online platform to host our data, like Supabase. However, due to the restrictions that Supabase had for “Free to use” subscribers, and the advice given by our lecturer Christine, we decided to each host the data locally. This would allow each of us to run our separate codes to clean and load the data, for optimal learning.

3.2 Data cleaning

What we cleaned from the “listings” table:

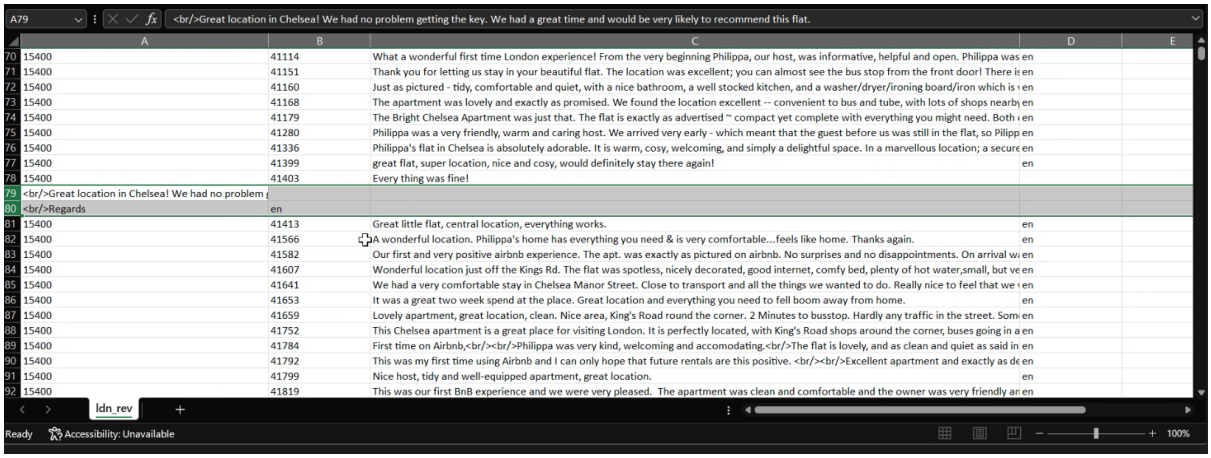
- host_location: remove null
- host_location: remove anything not in the “Amsterdam, Netherlands” format
- name: we only want to keep the listing name, no other additional info

What we cleaned from the “reviews” table:

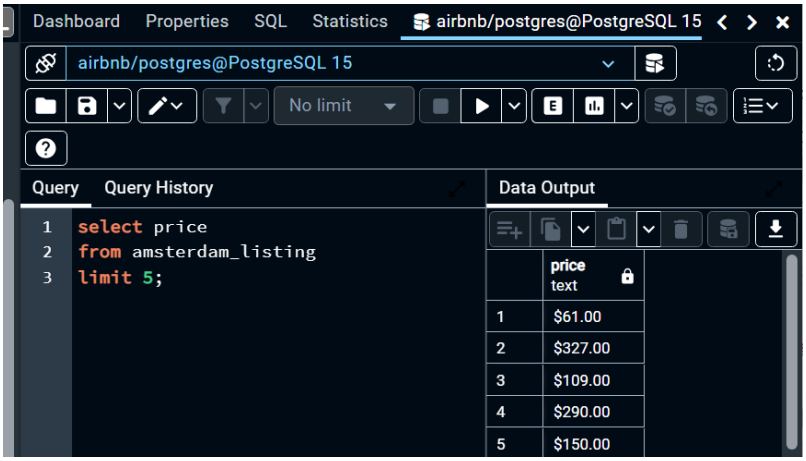
- comments: remove non-English comments

Challenges:

- To remove non-English comments, we used a Python library called Langdetect. However, it took a long time to process even tens of thousands of rows per dataframe.
- Kent looked at how we could process our data sets for the five cities in parallel, to save time.
- There were line breaks in our cleaned data; these were unwanted and we could not get rid of them. Was the issue Langdetect? Kent did a test to find out—he ran his code without Langdetect, and if needed, he would run Langdetect on a small data set to confirm that it was causing the issue.

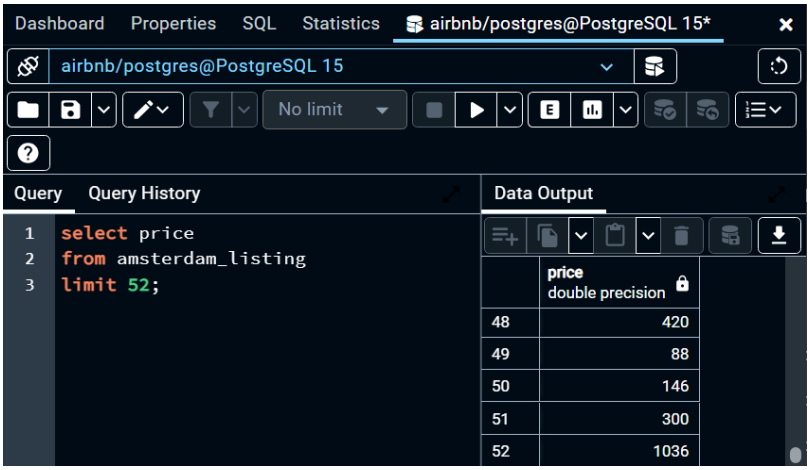


- During the analysing phase, Evelyn found out that the ‘price’ column in the listing data set was a ‘string’ data type, e.g. $1,234.00.



Findings & Actions:

- Langdetect is easier to handle compared to CLD2.
- Used Python’s Multiprocessing module to process the datasets in parallel to save time:
  - Time taken for Langdetect to process the smallest file **without** Multiprocessing: about 1 hour
  - Time taken for Langdetect to process all 5 files **with** Multiprocessing: 1 hour 12 mins
- Langdetect was not the cause of the line break.
  - Must **pre-process** the raw data before reading it into a dataframe with Pandas.
    - In this case, encoding to UTF-8
- Added a line of code to remove ‘$’, ‘,’ and change to the ‘float’ data type.



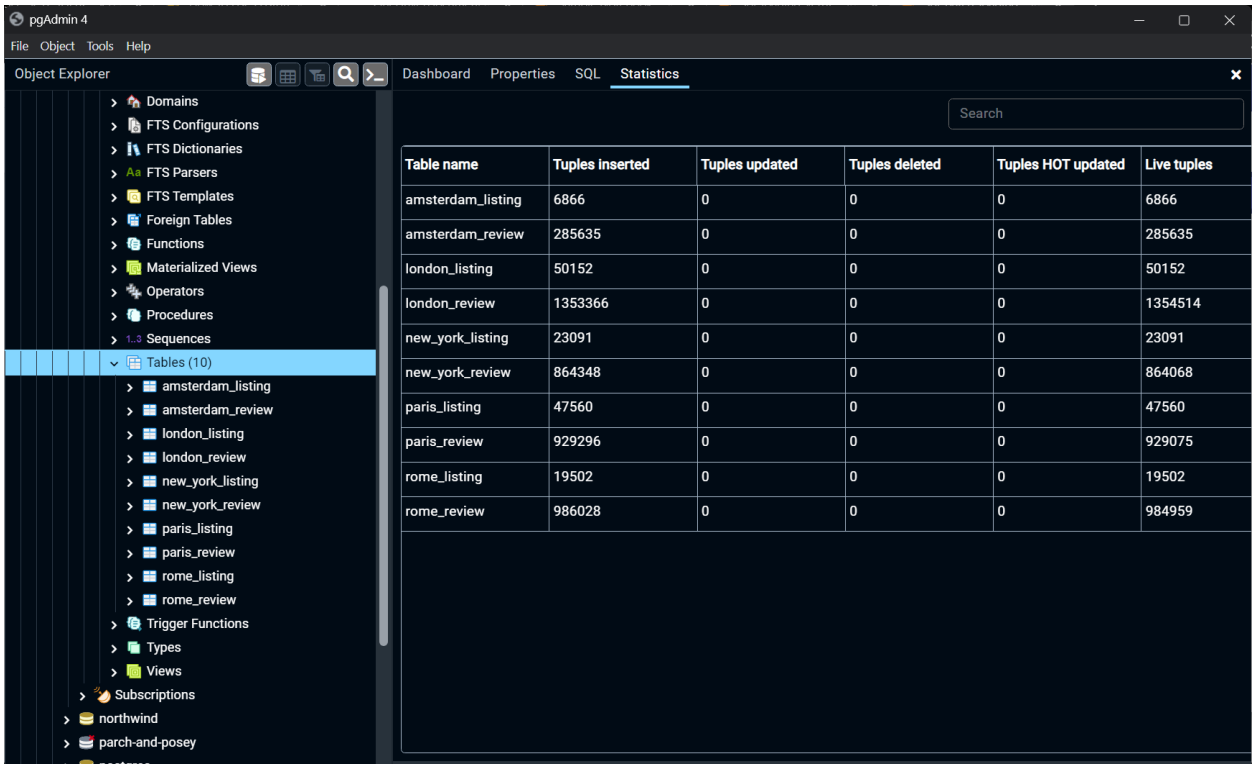
3.3 Upload to PostgreSQL (localhost)

Challenge:

- What was the most efficient way to load our data for all 5 cities?

Findings & Actions:

- Kent wrote a code to batch ingest data to Postgres.
- *Refer to Appendix II - ERD Schema Diagram*



4. Data Analysis

This section will detail how we used Python to answer our primary data questions, as listed in Section 2.1.

4.1 Analysis Approach

Boon Heng and Evelyn explored sentiment analysis in Python—this is the process of determining whether the language in a piece of text is positive, negative, or neutral.

Boon Heng used the Python tool VADER (Valence Aware Dictionary and sEntiment Reasoner), which is part of the Natural Language Toolkit library, better known as NLTK. Evelyn used the Python library Transformers to run a model known as BERT (Bidirectional Encoder Representations from Transformers).

4.1.1. Processing text: tokenisation

- **Similarity between models:** Both VADER and Transformers-based models like BERT start by processing raw text data into a format that can be analysed; this is called tokenisation. Tokenisation involves breaking down text into smaller pieces, called tokens, which can be words or subwords (smaller units than words), depending on the model.
- **Difference between models:** VADER focuses on whole words, while BERT breaks words into subwords. The subwords method helps BERT models to understand a wide range of words, even those not seen before. For example, if there is a new word not in the model's original data, BERT can still make sense of it by splitting it into smaller known pieces and looking at those pieces together. This is different from simpler tools like VADER, which might struggle with new or complicated words.

4.1.2 Scoring words:

- **VADER:** It scores the sentiment of text using a lexicon—a dictionary of sentiment-related words. In VADER's lexicon, each word is assigned a sentiment score, which can be positive, neutral, or negative. The sentiment score of a piece of text is determined by summing the sentiment scores of the recognised words in the text, adjusted according to rules that take into account negation and intensity modifiers. An example of a negation word is “not,” while an example of an intensity modifier is “extremely.”
- **Transformers:** Models like BERT do not score individual words in isolation. Instead, they analyse the text as a whole, considering the context in which words are used. These models have been trained on vast data sets, and have learned to associate words and phrases with sentiment as part of larger patterns in text. They generate a sentiment score based on the entire input sequence.

4.1.3 Calculating overall sentiment:

- **VADER:** It calculates the overall sentiment by considering the intensity and polarity of each word. The final sentiment score is a compound score representing the overall sentiment of the sentence.
- **Transformers:** The sentiment score comes from the model's output layer, which provides the probabilities of each sentiment class—'Positive' or 'Negative' in our use case. The overall sentiment is determined by selecting the class with the highest probability.

For this project, one issue with running the above models was the longer-than-expected processing time. For efficiency, we used sample sizes ranging from 250 to 1,000.

4.2 Were Reviews More Positive or Negative?
According to both our results, reviews for all five cities were mostly positive, for a random sampling of 250 reviews from each city.

Results from Transformers' pipeline function, where we specifically loaded the "distilbert-base-uncased-finetuned-sst-2-english" model, which is designed for sentiment analysis.

```
        # Determine overall sentiment
        positive_count = comments_df['sentiment'].value_counts().get('POSITIVE', 0)
        overall_sentiment = "Neutral"
        if positive_count > len(comments_df) / 2:
            overall_sentiment = "Mostly Positive"
        elif positive_count < len(comments_df) / 2:
            overall_sentiment = "Mostly Negative"

    print(f"Table {table_name} ({'sample' if table_size > THRESHOLD else 'full'}) is {overal
```

```
Table amsterdam_review (sample) is Mostly Positive
Table london_review (sample) is Mostly Positive
Table new_york_review (sample) is Mostly Positive
Table paris_review (sample) is Mostly Positive
Table rome_review (sample) is Mostly Positive
```

Results from VADER:

```
    # Determine overall sentiment
    positive_count = comments_df['type'].value_counts().get('postive', 0)
    percent_count = round((positive_count/SAMPLE_SIZE)*100, 2)
    overall_sentiment = "Neutral"
    if positive_count > len(comments_df) / 2:
        overall_sentiment = "Positive"
    elif positive_count < len(comments_df) / 2:
        overall_sentiment = "Negative"

    print(f"{table_name} ({'sample' if table_size > THRESHOLD else 'full'}) is {percent_count}
```

```
amsterdam_review (sample) is 97.2 % Positive
london_review (sample) is 96.4 % Positive
new_york_review (sample) is 94.4 % Positive
paris_review (sample) is 95.6 % Positive
rome_review (sample) is 98.0 % Positive
```

4.3 What are the most frequent words for the reviews?
In Evelyn's approach, she used ChatGPT to develop a script to analyse customer reviews from the five cities, with a random sample size of 1,000 for each city. Her script extracted the highest frequency words—capped at top 50—to generate a word cloud, as well as a list of top words.

For the above, these were the additional tools imported:

- **collections.Counter:** This class is part of the Collections module. It is used to count the frequency of words in the text, to identify the most common words.
- **matplotlib.pyplot:** This module from the Matplotlib library is used to display the word clouds.
- **nltk:** The Natural Language Toolkit is needed for several operations in the script:
    - **nltk.corpus.stopwords:** This function retrieves a list of 'stop words'; these are common words typically excluded during text processing to focus on more significant words.
    - **nltk.tokenize.word_tokenize:** This function breaks down the text into individual words or 'tokens.'
- **wordcloud.WordCloud:** This class from the Wordcloud library is used to generate a visual word cloud from the provided text.

In summary, the script processes review texts by:

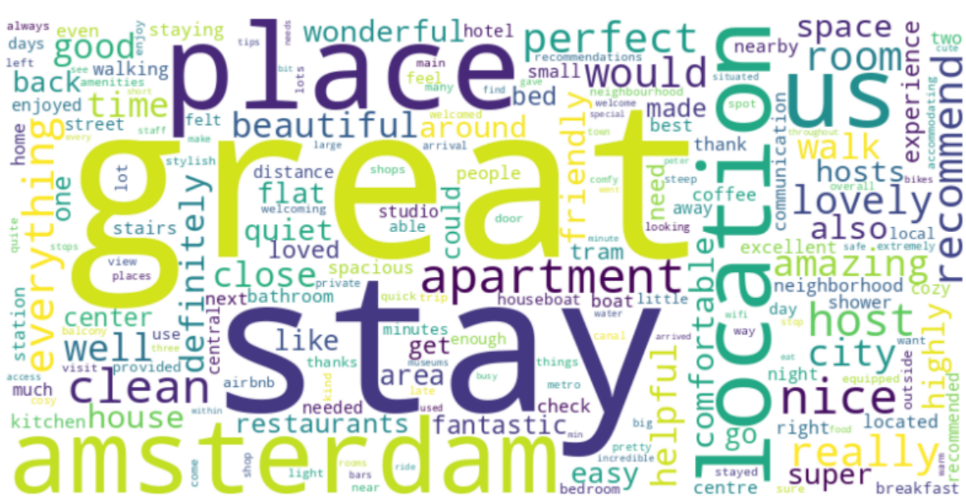
- **Tokenising:** splitting the text into individual words.
- **Lowercasing:** converting all words to lowercase to ensure consistency. This is the typical procedure when using NLTK tools, which are case sensitive. If lowercasing is not carried out, it would unnecessarily increase the size of the analysis data set. For instance, "Dog," "dog," and "DOG" would be considered three separate words.
- **Defining, removing stop words:** Excluding common words that don't carry significant meaning.
- **Filtering non-alphabetic tokens:** Ensuring that symbols or numerics don't skew the word frequency counts.

In Evelyn's final results, the word clouds did not differ much by city. Below is the Amsterdam word cloud—it suggests a positive sentiment, with location and cleanliness being top customer priorities.

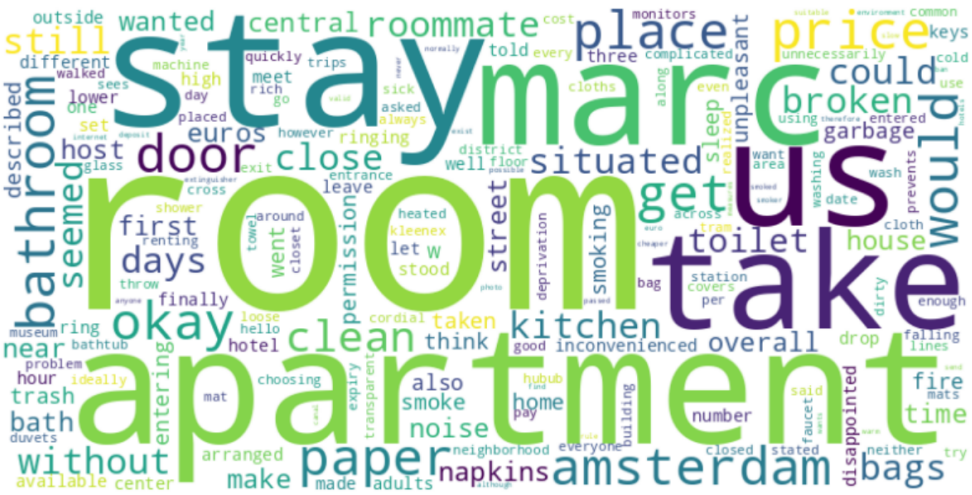
Word Cloud for amsterdam_review

Boon Heng decided to further explore the word cloud by grouping words into 3 categories (“Positive”, “Negative,” and “Neutral”) through sentiment analysis, before generating word clouds. This approach was more targeted, and helped us to get a clearer picture of what customers were happy or dissatisfied with.

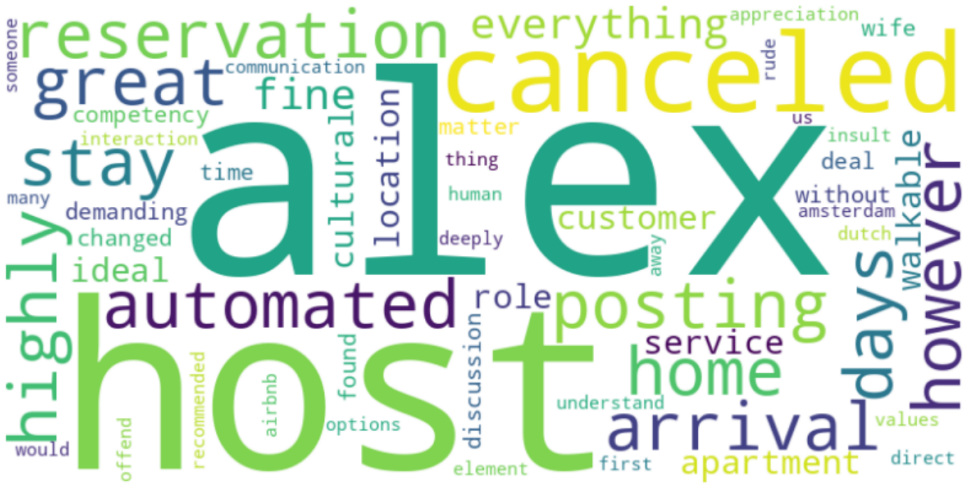
Positive Word Cloud for amsterdam_review



Negative Word Cloud for amsterdam_review



Neutral Word Cloud for amsterdam_review



4.4 Is there a correlation between price and review sentiment?

With the help of ChatGPT, Evelyn produced a script to analyse the correlation between the sentiment scores of property reviews and the price of properties in different cities. She used a sample size of 250 for each city.

In summary, the script processes reviews by:

- **Fetching and preparing data:** Retrieving property listings and their corresponding reviews, and truncating comments to fit the model's maximum input size (512 tokens) for sentiment analysis.
- **Analysing sentiment:** Utilising a pre-trained sentiment analysis model from the Transformers library (distilbert-base-uncased-finetuned-sst-2-english) to evaluate the emotional tone of each review. Appending sentiment scores and labels directly to the existing data for further analysis.

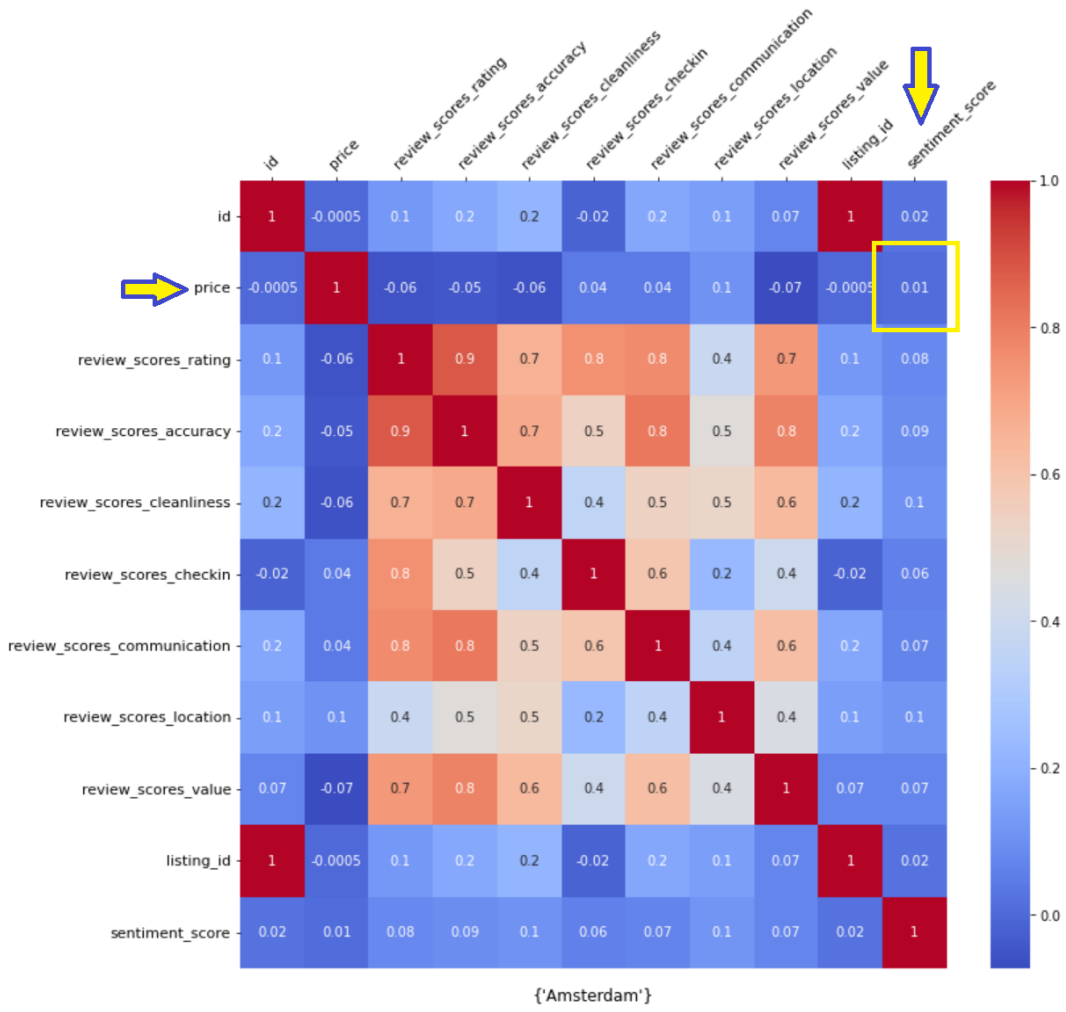
- **Calculating correlation:** Computing the statistical correlation between the sentiment of reviews and the associated property prices.
- **Iterating across cities:** Repeating the entire analysis process for each specified city in the list.

In Evelyn's final results, there was no correlation between price and review sentiment.

```
Correlation between sentiment score and price in amsterdam: 0.01
Correlation between sentiment score and price in london: 0.00
Correlation between sentiment score and price in new_york: 0.00
Correlation between sentiment score and price in paris: 0.02
Correlation between sentiment score and price in rome: -0.10
```

This held true even when she increased the minimum price to 500 for sample selection, with a larger sample size of 500.

To explore and visually represent correlations in our data, Boon Heng used the Seaborn library to create heatmaps for each city. This approach allowed us to observe correlations across multiple data fields, extending beyond the singular connection between review sentiment and pricing. (*For details on heatmaps, see Appendix III.*)



From the heatmap, price has a correlation value of 0.01 with sentimental_score—meaning there is almost no correlation. Price has a slightly higher correlation value with other variables, but nothing above 0.5, i.e. moderate correlation.

5. Conclusion

This was a fulfilling project to work on, and we all gained valuable learning experiences. Below is a summary of our challenges and lessons learned, as well as suggested improvements.

5.1 Project Challenges

During the project, these were our main challenges:

1. When we ran the Langdetect tool, this triggered an error message—the text needed to be encoded in UTF-8 format. Had to encode the original data as UTF-8, and remove </br> tags in the data. These cleaned .csv files were then imported into the data frame.
2. Not a straightforward process to incorporate the reference codes that we found online into our project. We had to do a lot of additional research, as well as modify our codes in order to make everything work.
3. Encountered “AttributeError: 'Engine' object has no attribute 'execute,” when codes had worked a day before. Managed to fix the codes, but still not sure what was the cause.
4. Challenging to document all project developments and findings in one place, as well as keep track of all Python libraries and tools used. To develop a more efficient documentation process in future.

5.2 Reflections and Lessons Learned

These were our main learning points:

1. Data Extraction
    - For this project, we were fortunate to have ready-made data in .csv format, which we were familiar with. In future, if we are dealing with unfamiliar data, we would need to stretch our abilities to find ways to handle the data.
2. Data Cleaning
    - Running our data cleaning codes without incorporating multi-processing will take up too much time.
3. Data Loading
    - Initially, we did not think of making our code “fully automated.” (For example, we still manually went to pgAdmin to create a database.) However, as we progressed, we felt that this “creating database” step could be incorporated into our Python script, hence making the process even more productive.
4. Data Analysing
    - Choosing what to analyse and what Python tools to use can be difficult, since we have not had any experience with this. One way to find out is to do a search on Google to see how others have done their research, and which Python tools are

mentioned more—usually more people will use the same tool if it works well and is relatively easy to apply. Another way is to use ChatGPT to develop codes.

5.3 Areas for Further Exploration

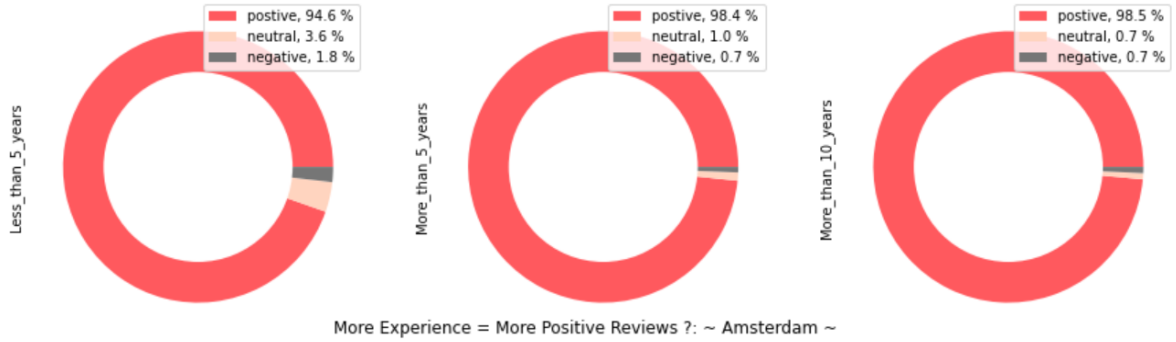
Given more time, these are some areas that we would have liked to investigate further:

1. We would have extracted data from more cities—currently we only have 5 because of the time required to run the codes for each city. (Note: the cleaned tables for each city are mostly over 10,000 rows.)
2. We would have considered appending a “city, country” field to the data, instead of dropping “host_location” rows that didn’t conform to our needs.
3. Given more time, we would have tried to look for a suitable place to host our data; ideally, a scalable and secure platform suitable for hosting PostgreSQL databases—with a free tier.
4. The sentiment analysis tools that we used only apply to English words. For a more complete picture, we would have liked to use sentiment analysis tools that work on other languages as well. (The reviews in our data sets consisted of several languages, but we only processed English reviews.)
5. For sample sizes, we chose arbitrary values—such as 250, 500, and 1,000—for shorter runtimes. Given more time, we would have experimented with Python libraries such as Math and SciPy to derive a mathematically sound sample size for each of our data sets.
6. To have a better sense of script runtimes, we could use the Tqdm library in Python to add progress bars. ("Tqdm" stands for "taqaddum" in Arabic, which means "progress.")

Appendix I

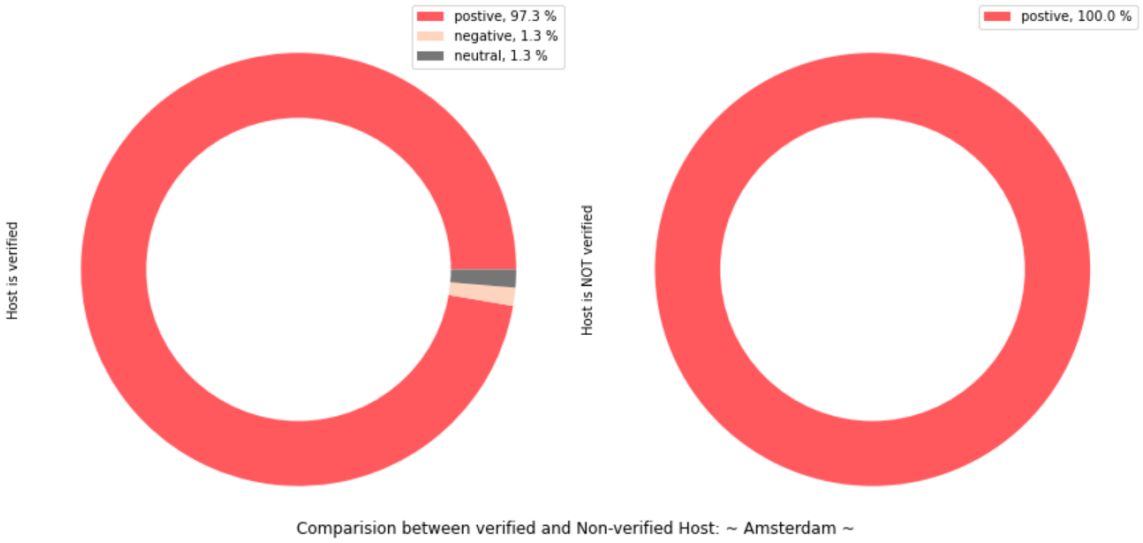
Showing only 1 city.

1. Do more experienced Airbnb hosts get more positive or negative reviews?

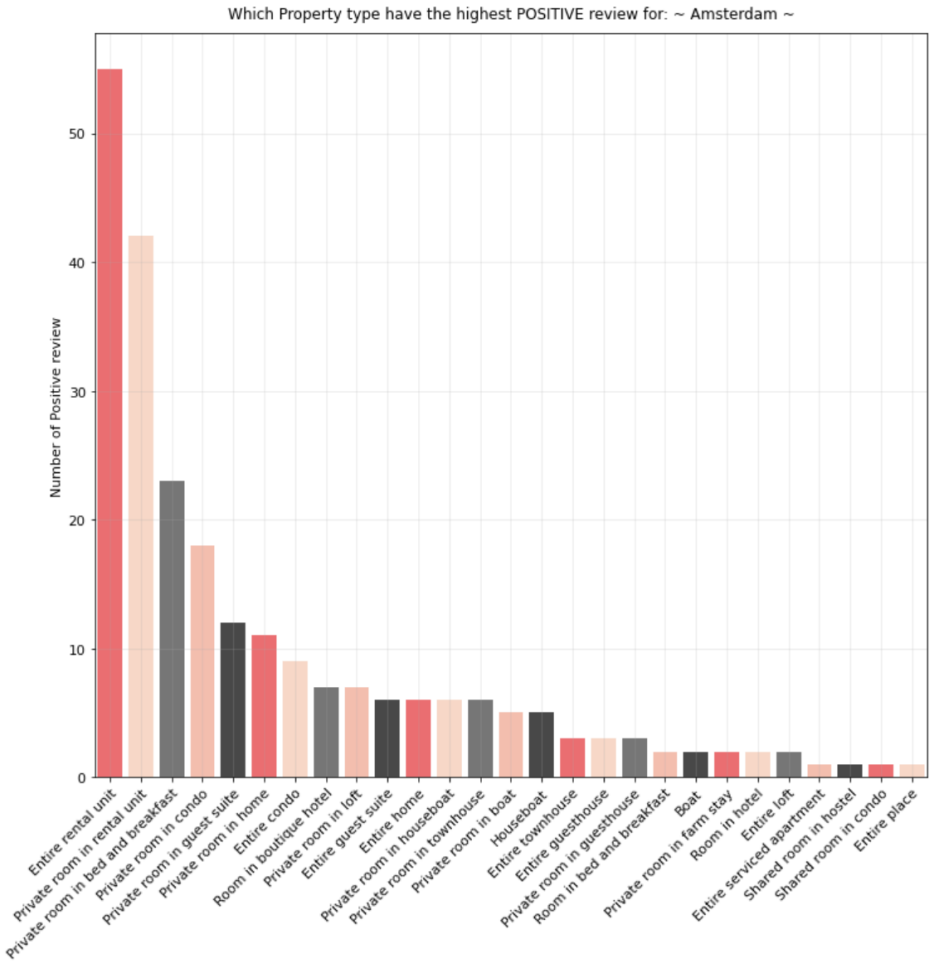


2. For Airbnb Superhosts, are their reviews more positive or negative? (“Airbnb Superhost” is a designation given to experienced and highly rated hosts on the Airbnb platform, who meet certain criteria set by Airbnb.)

3. For Airbnb hosts who don't verify their identity, does this impact their rating?



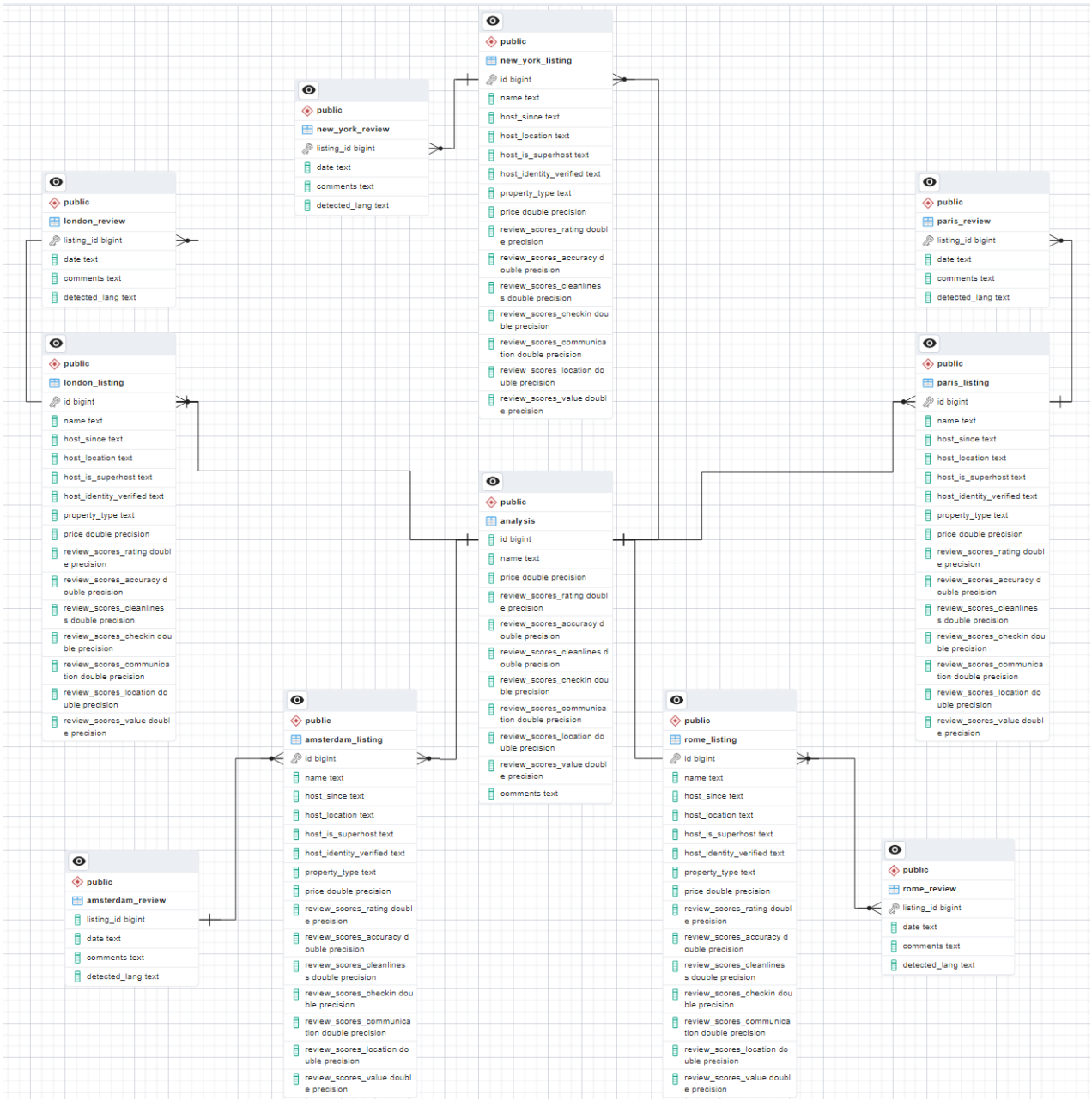
4. Which types of homes receive the most positive reviews?



Appendix II

ERD Schema Diagram A (*This is how our data was organised for this project. This design served the purposes of our project, and we were able to merge the city tables for analysis when needed.*)

ERD Schema Diagram B (*The purpose of this ERD diagram is to represent how the data creators—Inside Airbnb—may have normalised their scraped data. Note that this is just a representation.*)

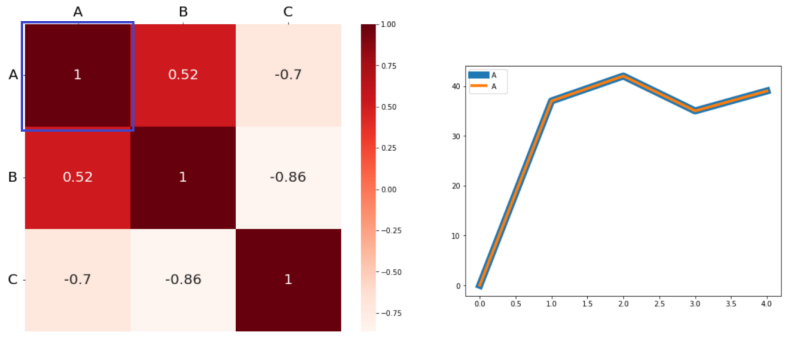


Appendix III

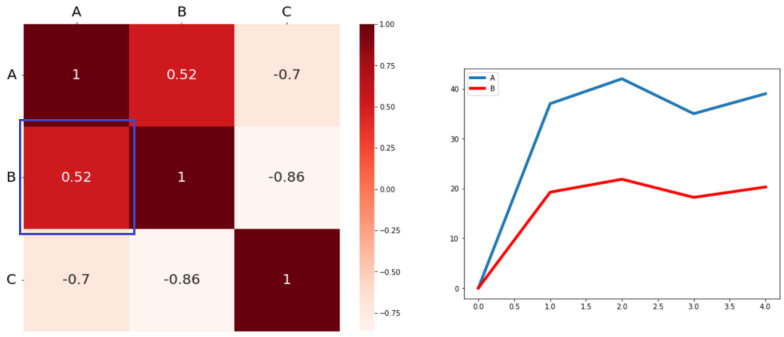
How do we interpret the correlation value on heatmaps?

- Correlation value of 1.0 = Perfect positive correlation (2 variables move in the same direction closely)
- Correlation value of 0 = No relationship at all (1 variable is moving, but the other variable might not be moving)
- Correlation value of -1.0 = Perfect negative correlation (2 variables move in the opposite direction closely)

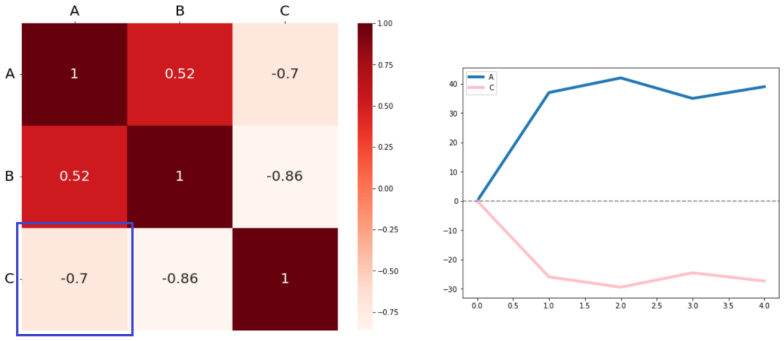
See 3 examples below, and notice how 2 variables react to one another based on the correlation value on the line chart. (Refer to the “Blue” square for each example.)



Correlation value of '1.0' Between A and A



Correlation value of '0.52' Between A and B



Correlation value of '-0.7' Between A and C