Airbnb Insights

Unveiling Host Experience

Group 06
Diogo Faria: up201907014
Sérgio Gama: up201906690
Tiago Rodrigues: up201906807
Valentina Wu: up201907483

Subject Description

In the Airbnb Listings project, the focus is on designing and implementing a data warehouse for Airbnb reviews to offer a comprehensive analysis of users' experiences of a property within the Airbnb platform.

A dataset has been chosen in the city of Porto, Portugal, which will allow this project to be carried out with the necessary abundance of data and not become excessive. This dataset includes information about the review that the user left on the property listed on the platform, as well as details of the property listed, such as the score assigned, and some statistics on the quality of properties based on location, among other relevant attributes.

Assignment Requirements

- The number of facts:
  - The dataset contains around 745,000 facts related to Airbnb reviews for the city of Porto. In addition, there are 12,818 facts for listings and 156 facts for location-based listing statistics. These numbers exceed the specified minimum of 10,000 facts.
- Aggregate facts:
  - In the fact table, Location-based Listing Statistics presents attributes that are considered addictive and semi-addictive. These attributes provide information that allows you to understand statistics at a regional level.
- Dimensional Framework: There are more than 4 dimensions are common in different facts.
  - Review: Date, Host, Property, Reviewer
  - Listing: Location Host, Property
  - Location-Based Listing Statistics: Location

Planning

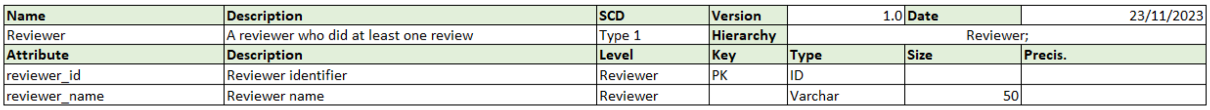
Facts\Dimensions	Location	Date	Host	Property	Reviewer
Review		x	x	x	x
Listing	x		x	x	
Location-Based Listing Statistics	x				

Our project has 5 dimensions and 3 facts. Our fact tables contain information about:

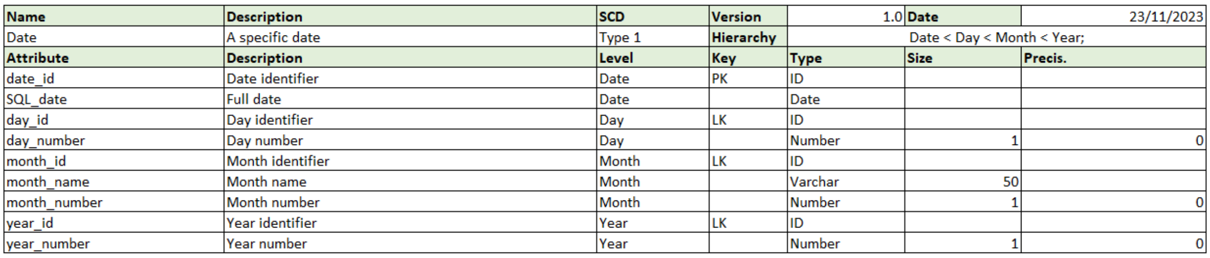
- The reviews left on a particular property: which includes who left the review, when they left it, and who owns the property the review was left on;
- A particular listing: what the property the listing is trying to advertise is, where it is, and the host of that property.

- Statistics around pricing based on locations: the number of listings and price points on a particular location.

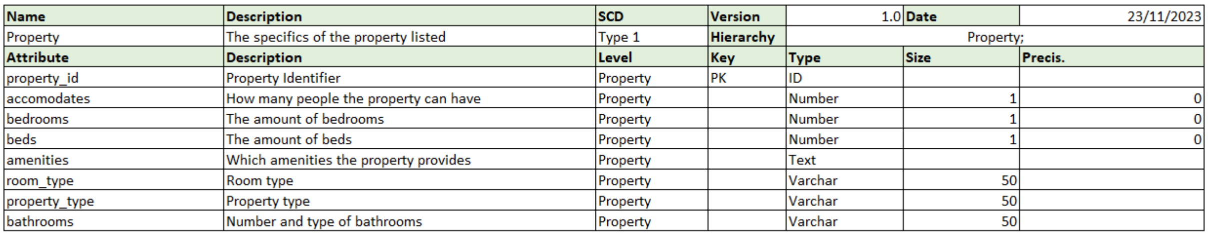
Next, we'll present the dictionaries for our dimensions, followed by the dictionaries for the fact tables:



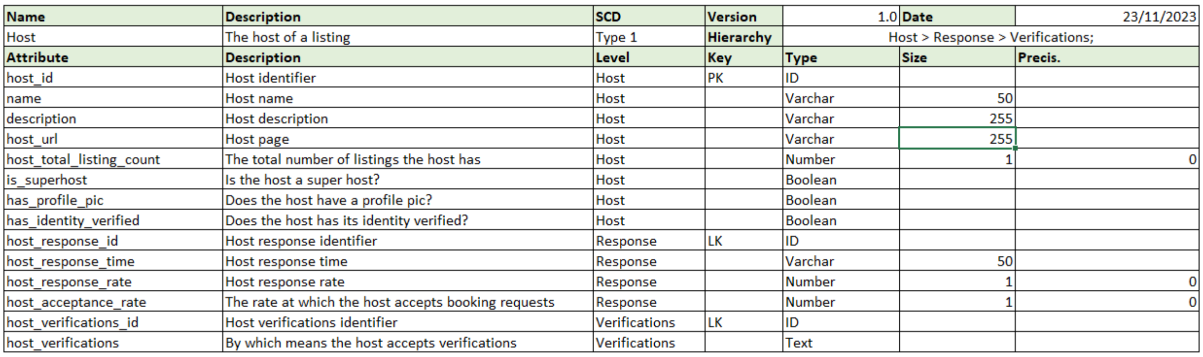
Name	Description	SCD	Version	1.0	Date	23/11/2023
Reviewer	A reviewer who did at least one review	Type 1	Hierarchy	Reviewer;		
Attribute	**Description**	**Level**	**Key**	**Type**	**Size**	**Precis.**
reviewer_id	Reviewer identifier	Reviewer	PK	ID		
reviewer_name	Reviewer name	Reviewer		Varchar	50	



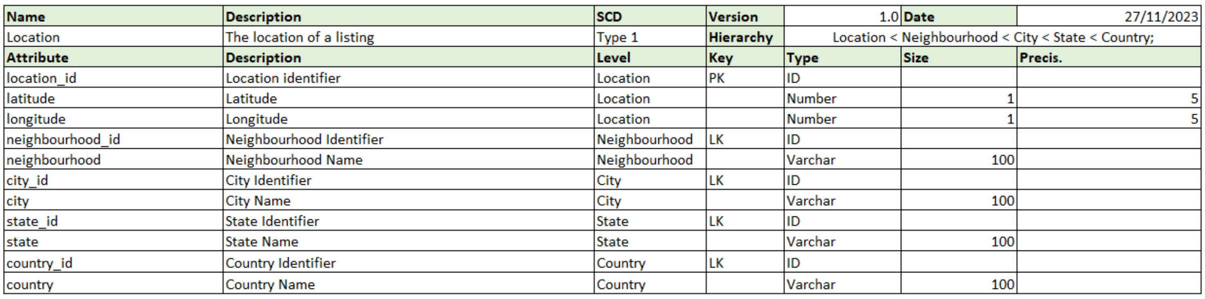
Name	Description	SCD	Version	1.0	Date	23/11/2023
Date	A specific date	Type 1	Hierarchy	Date < Day < Month < Year;		
Attribute	**Description**	**Level**	**Key**	**Type**	**Size**	**Precis.**
date_id	Date identifier	Date	PK	ID		
SQL_date	Full date	Date		Date		
day_id	Day identifier	Day	LK	ID		
day_number	Day number	Day		Number	1	0
month_id	Month identifier	Month	LK	ID		
month_name	Month name	Month		Varchar	50	
month_number	Month number	Month		Number	1	0
year_id	Year identifier	Year	LK	ID		
year_number	Year number	Year		Number	1	0



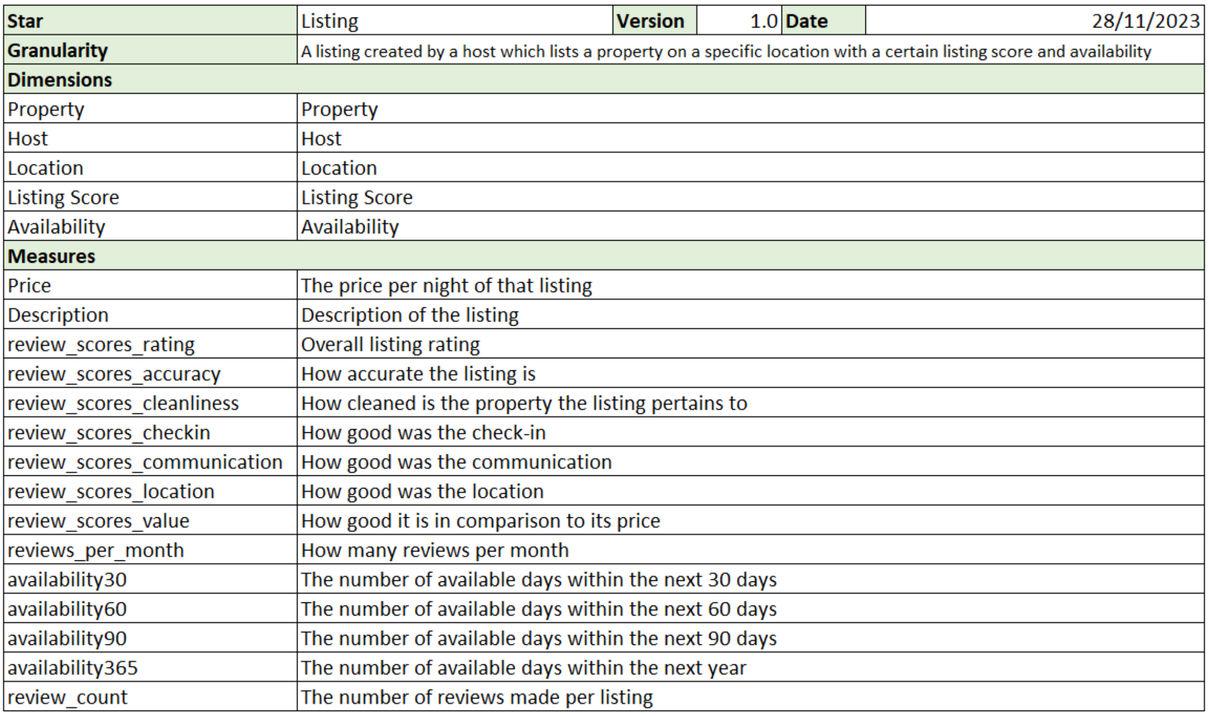
Name	Description	SCD	Version	1.0	Date	23/11/2023
Property	The specifics of the property listed	Type 1	Hierarchy	Property;		
Attribute	**Description**	**Level**	**Key**	**Type**	**Size**	**Precis.**
property_id	Property Identifier	Property	PK	ID		
accomodates	How many people the property can have	Property		Number	1	0
bedrooms	The amount of bedrooms	Property		Number	1	0
beds	The amount of beds	Property		Number	1	0
amenities	Which amenities the property provides	Property		Text		
room_type	Room type	Property		Varchar	50	
property_type	Property type	Property		Varchar	50	
bathrooms	Number and type of bathrooms	Property		Varchar	50	



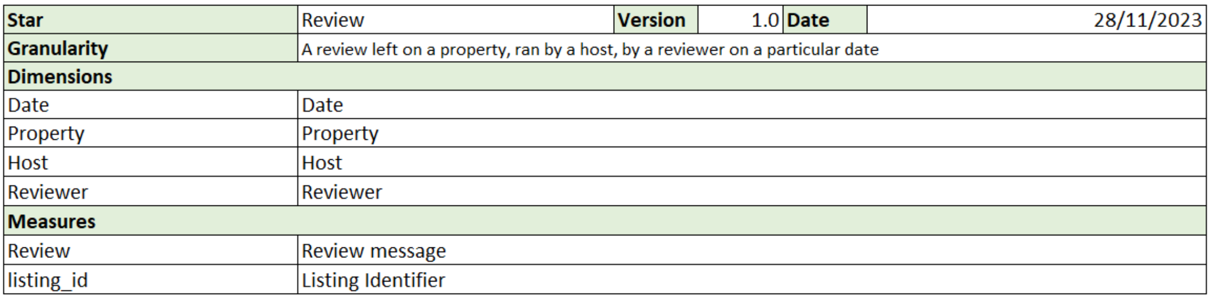
Name	Description	SCD	Version	1.0	Date	23/11/2023
Host	The host of a listing	Type 1	Hierarchy	Host > Response > Verifications;		
Attribute	**Description**	**Level**	**Key**	**Type**	**Size**	**Precis.**
host_id	Host identifier	Host	PK	ID		
name	Host name	Host		Varchar	50	
description	Host description	Host		Varchar	255	
host_url	Host page	Host		Varchar	255	
host_total_listing_count	The total number of listings the host has	Host		Number	1	0
is_superhost	Is the host a super host?	Host		Boolean		
has_profile_pic	Does the host have a profile pic?	Host		Boolean		
has_identity_verified	Does the host has its identity verified?	Host		Boolean		
host_response_id	Host response identifier	Response	LK	ID		
host_response_time	Host response time	Response		Varchar	50	
host_response_rate	Host response rate	Response		Number	1	0
host_acceptance_rate	The rate at which the host accepts booking requests	Response		Number	1	0
host_verifications_id	Host verifications identifier	Verifications	LK	ID		
host_verifications	By which means the host accepts verifications	Verifications		Text		



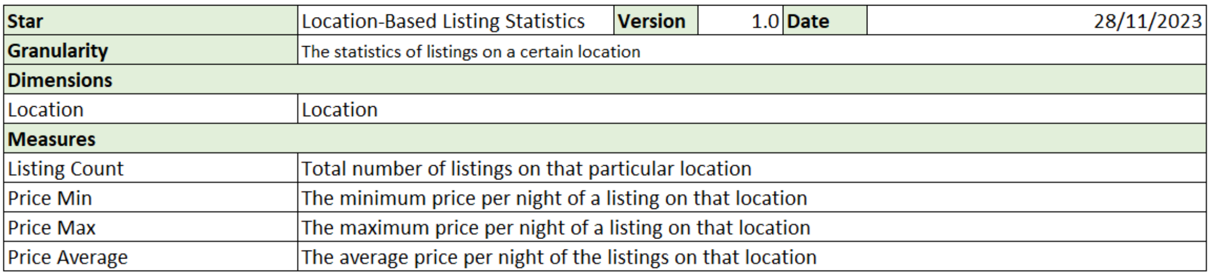
Name	Description	SCD	Version	1.0	Date	27/11/2023
Location	The location of a listing	Type 1	Hierarchy	Location < Neighbourhood < City < State < Country;		
Attribute	**Description**	**Level**	**Key**	**Type**	**Size**	**Precis.**
location_id	Location identifier	Location	PK	ID		
latitude	Latitude	Location		Number	1	5
longitude	Longitude	Location		Number	1	5
neighbourhood_id	Neighbourhood Identifier	Neighbourhood	LK	ID		
neighbourhood	Neighbourhood Name	Neighbourhood		Varchar	100	
city_id	City Identifier	City	LK	ID		
city	City Name	City		Varchar	100	
state_id	State Identifier	State	LK	ID		
state	State Name	State		Varchar	100	
country_id	Country Identifier	Country	LK	ID		
country	Country Name	Country		Varchar	100	



Star	Listing	Version	1.0	Date	28/11/2023
Granularity	A listing created by a host which lists a property on a specific location with a certain listing score and availability				
Dimensions					
Property	Property				
Host	Host				
Location	Location				
Listing Score	Listing Score				
Availability	Availability				
Measures					
Price	The price per night of that listing				
Description	Description of the listing				
review_scores_rating	Overall listing rating				
review_scores_accuracy	How accurate the listing is				
review_scores_cleanliness	How cleaned is the property the listing pertains to				
review_scores_checkin	How good was the check-in				
review_scores_communication	How good was the communication				
review_scores_location	How good was the location				
review_scores_value	How good it is in comparison to its price				
reviews_per_month	How many reviews per month				
availability30	The number of available days within the next 30 days				
availability60	The number of available days within the next 60 days				
availability90	The number of available days within the next 90 days				
availability365	The number of available days within the next year				
review_count	The number of reviews made per listing				

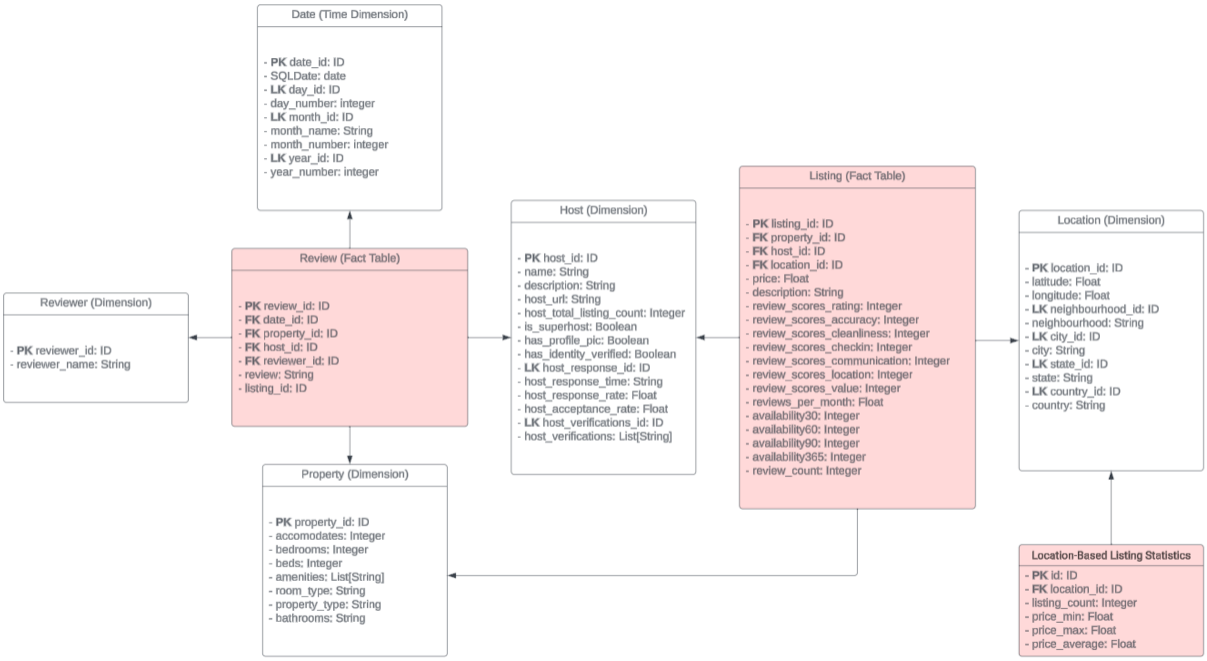


Star	Review	Version	1.0	Date	28/11/2023
Granularity	A review left on a property, ran by a host, by a reviewer on a particular date				
Dimensions					
Date	Date				
Property	Property				
Host	Host				
Reviewer	Reviewer				
Measures					
Review	Review message				
listing_id	Listing Identifier				



Star	Location-Based Listing Statistics	Version	1.0	Date	28/11/2023
Granularity	The statistics of listings on a certain location				
Dimensions					
Location	Location				
Measures					
Listing Count	Total number of listings on that particular location				
Price Min	The minimum price per night of a listing on that location				
Price Max	The maximum price per night of a listing on that location				
Price Average	The average price per night of the listings on that location				

Dimensional data model



The presented dimensional model offers a robust database schema tailored for a property rental platform, Airbnb.

Dimension tables

- **reviewer**: Contains information about individuals providing reviews, including unique identifiers and reviewer names.
- **date**: Stores details related to dates, including day, month, and year components, facilitating time-based analysis.
- **property**: Captures characteristics of listed properties such as accommodations, bedrooms, beds, and amenities, enabling detailed property-level insights.
- **host**: Provides details about property hosts, encompassing host names, descriptions, response metrics, and verification status.
- **location**: Stores geographical coordinates only linked to the neighbourhood table, which enables spatial analysis.
  - **country**: Contains country-specific information, likely including unique identifiers and country codes.
  - **state**: Includes state-specific information, establishing a foreign key relationship with the country table for hierarchical geographic analysis.
  - **city**: Holds city-specific details with a foreign key link to the state table, allowing for localised analysis.
  - **neighbourhood**: Contains information about neighbourhoods, establishing a foreign key relationship with the city table for granular geographic insights.

Fact tables

- **listing**: Serves as the core fact table, capturing essential details about property listings, including price, availability, and review scores. Foreign key relationships link it to the property, host, and location dimension tables.

- **review**: Captures details about individual reviews, linking to dimensions such as date, property, host, and reviewer, providing a holistic view of the review data.

Aggregation tables

In the dimensional model itself, there is only one aggregation table, but in practice, location-based aggregation is implemented with these four tables below:

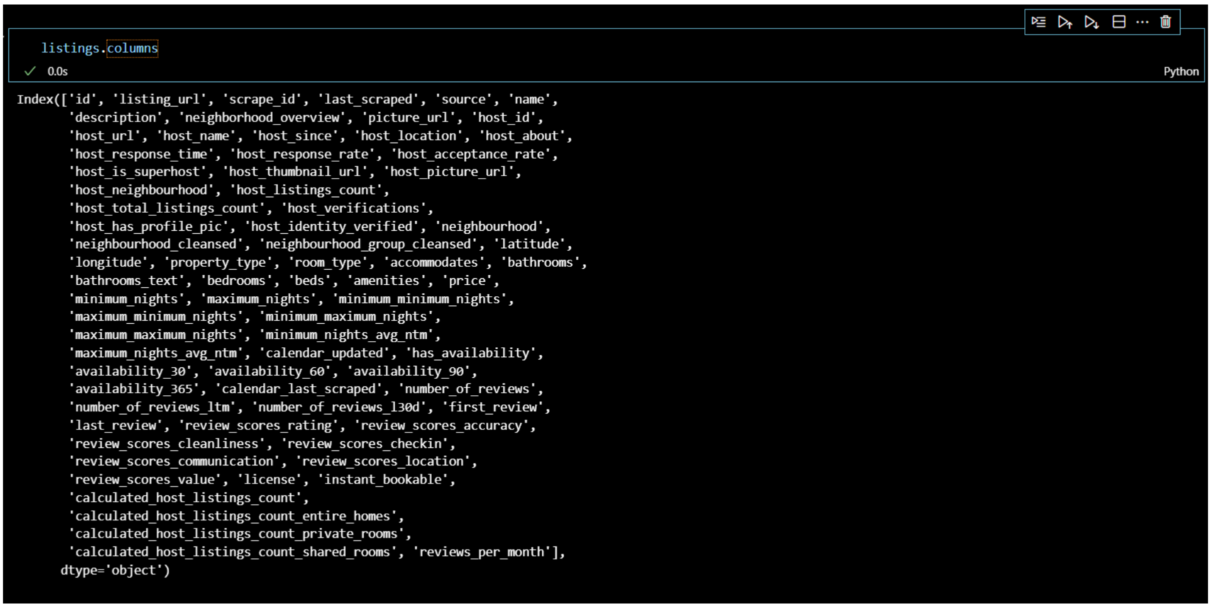
- **neighbourhood_based_listing_statistics**: Contains pre-computed aggregated statistics at the neighbourhood level, including minimum, maximum, and average prices, facilitating efficient querying for localized insights.
- **city_based_listing_statistics**: Offers aggregated statistics at the city level, providing a higher-level perspective on property data.
- **state_based_listing_statistics**: Presents aggregated statistics at the state level, allowing for regional analysis.
- **country_based_listing_statistics**: Contains aggregated statistics at the country level, offering a broad dataset overview.

Data sources selection. Extraction, transformation, and loading

The dataset of Airbnb Listing was collected on OpenDataSoft, a data-sharing platform that enables organizations to publish, share, and collaborate on data.

Regarding the dataset, Airbnb Listing refers to individual properties that hosts make available for short-term rental through the Airbnb platform. These listings include host information, property information such as location, price, property characteristics, availabilities, and some reviews associated with the properties. As the dataset provides a large number of facts, we chose only the data related to Porto City. Thus, we gathered two tables: listings and reviews.

The listings have the following columns:



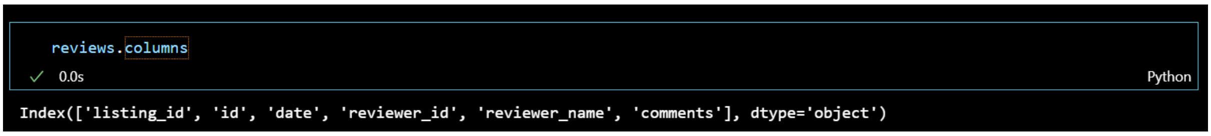
```
listings.columns
```

✓ 0.0s

Python

```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'source', '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')
```

And the reviews:



```
reviews.columns
```

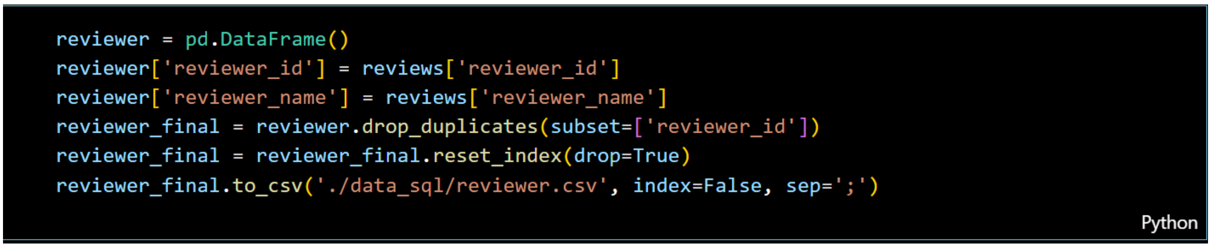
✓ 0.0s

Python

```
Index(['listing_id', 'id', 'date', 'reviewer_id', 'reviewer_name', 'comments'], dtype='object')
```

After gathering the dataset, we used Python (the panda's library) to transform the data in the 7 tables presented in our dimensional model; the names used in the dimensional model are the same ones on the tables created. We transform the table location as a cascade table to facilitate the analysis of the aggregated fact, the Location-Based Listings Statistics.

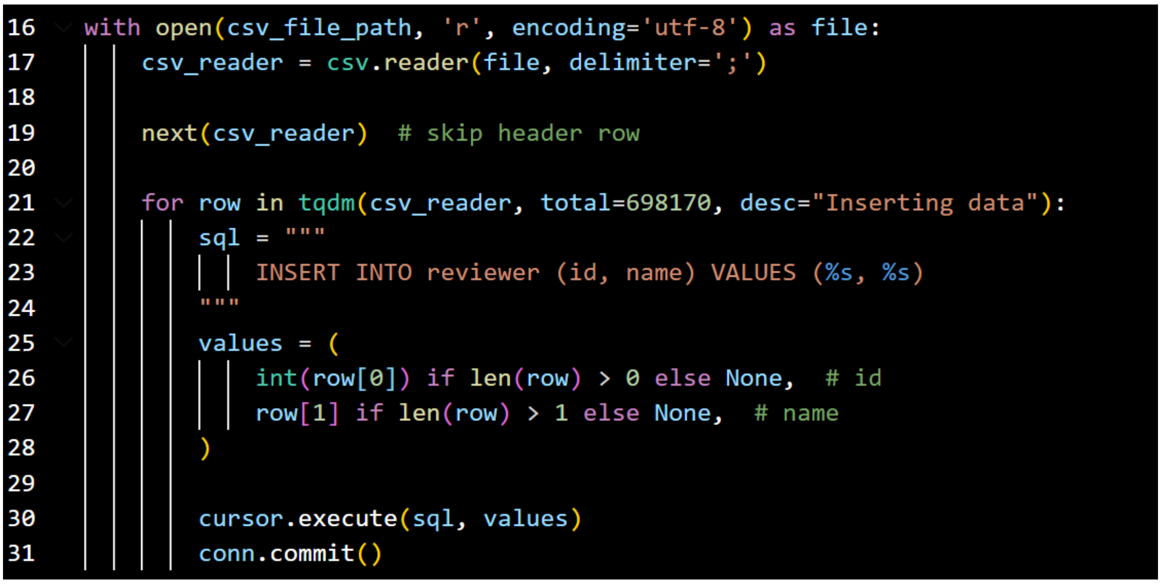
An example of a Python script to get the table reviewer that fetches the essential attributes: the 'reviewer_id' and 'reviewer_name'



```
reviewer = pd.DataFrame()
reviewer['reviewer_id'] = reviews['reviewer_id']
reviewer['reviewer_name'] = reviews['reviewer_name']
reviewer_final = reviewer.drop_duplicates(subset=['reviewer_id'])
reviewer_final = reviewer_final.reset_index(drop=True)
reviewer_final.to_csv('./data_sql/reviewer.csv', index=False, sep=';')
```

Python

To facilitate the storage and retrieval of data, we employ the MySQL database, a widely adopted open-source relational database management system known for its effectiveness in data management. The database is instantiated within a Docker container, providing a portable and scalable environment. To populate each table with processed data, we execute a Python script tailored for this purpose.
The figure below shows how we populate the data about the reviewer into the database:



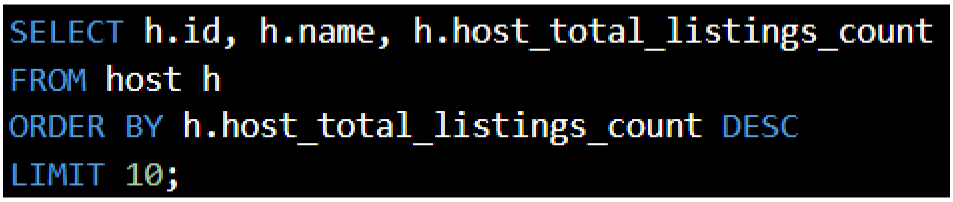
```python
with open(csv_file_path, 'r', encoding='utf-8') as file:
    csv_reader = csv.reader(file, delimiter=';')

    next(csv_reader)  # skip header row

    for row in tqdm(csv_reader, total=698170, desc="Inserting data"):
        sql = """
            INSERT INTO reviewer (id, name) VALUES (%s, %s)
        """
        values = (
            int(row[0]) if len(row) > 0 else None,  # id
            row[1] if len(row) > 1 else None,  # name
        )

        cursor.execute(sql, values)
        conn.commit()
```

Querying and Data Analysis

This section will delve into the queries we've chosen to analyse the data. Each query has been chosen to extract valuable information from the various dimensions of our data warehouse.

1. **Find the top hosts based on the total number of listings they have.**

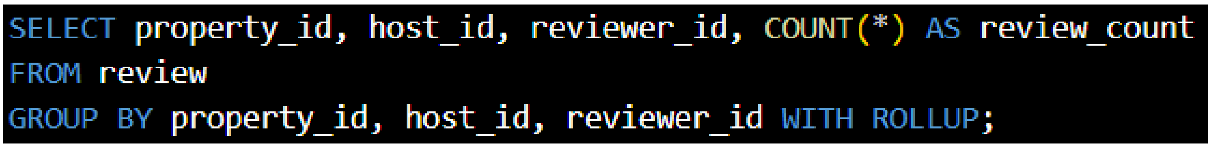
We want to be able to find the top hosts to understand where we should choose by the total number of listings they have.



```sql
SELECT h.id, h.name, h.host_total_listings_count
FROM host h
ORDER BY h.host_total_listings_count DESC
LIMIT 10;
```

2. **Number of reviews for each combination of ‘property_id’, ‘host_id’ and ‘reviewer_id’**

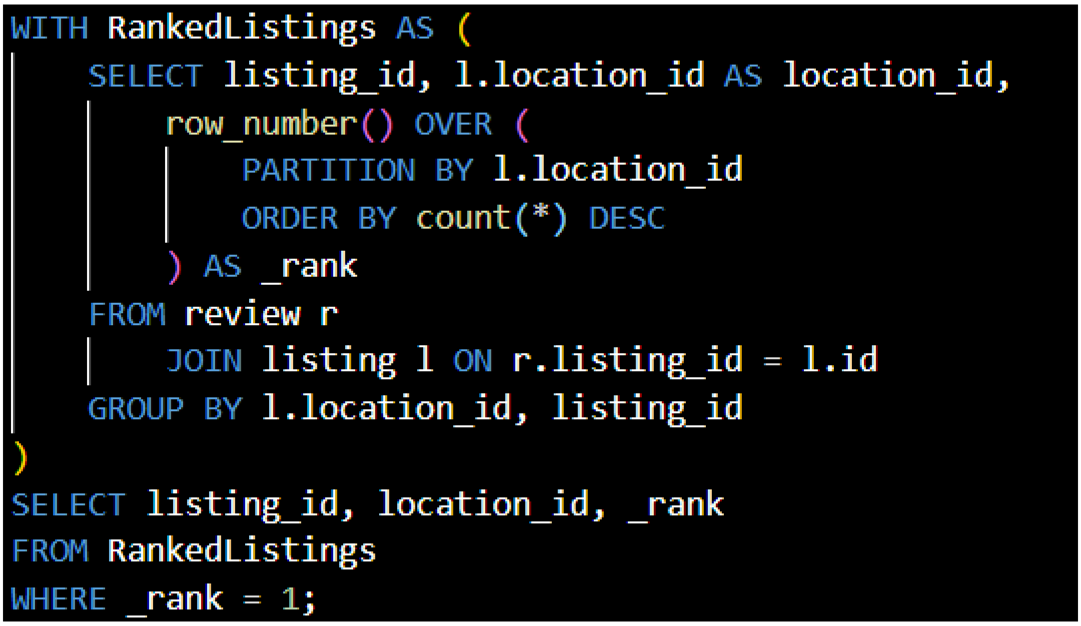
This query provides a detailed insight into the number of reviews for each combination of ‘property_id’, ‘host_id’, and ‘reviewer_id’. This allows us to see how many reviews each specific property receives from individual reviewers and hosted by specific hosts.



```sql
SELECT property_id, host_id, reviewer_id, COUNT(*) AS review_count
FROM review
GROUP BY property_id, host_id, reviewer_id WITH ROLLUP;
```

3. **Listings with the Most Reviews in Each Neighbourhood**

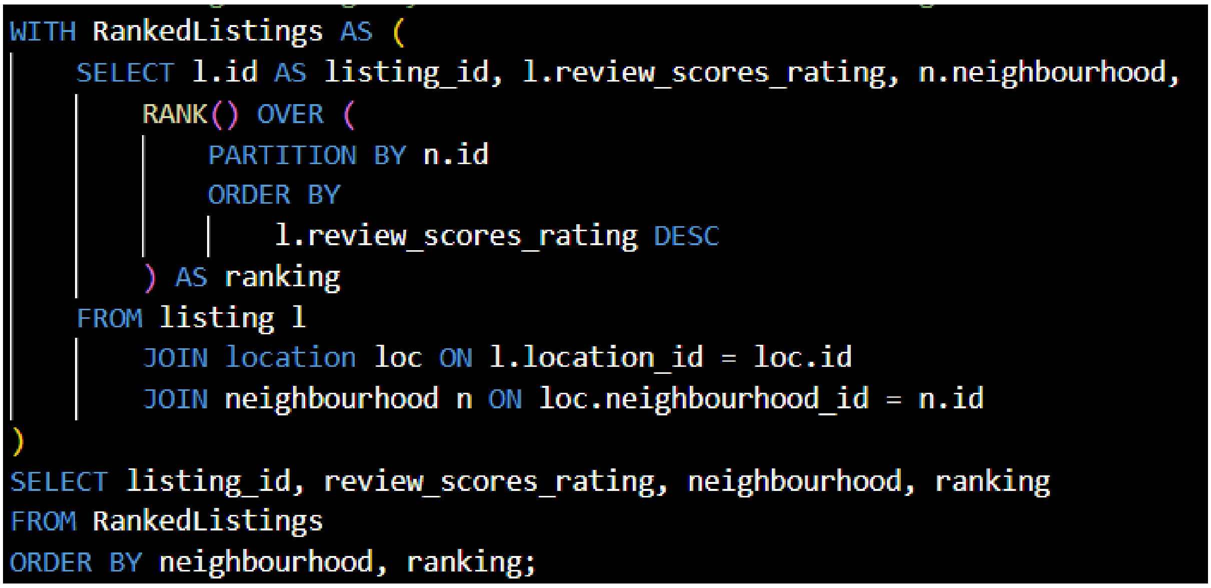
This query uses a Common Table Expression (CTE) to rank listings within each neighbourhood based on the number of reviews they have received. The ‘_rank’ column is assigned using the ‘row_number()’ window function, and then the main query filters out the listings with the highest rank in each neighbourhood.



```sql
WITH RankedListings AS (
    SELECT listing_id, l.location_id AS location_id,
        row_number() OVER (
            PARTITION BY l.location_id
            ORDER BY count(*) DESC
        ) AS _rank
    FROM review r
        JOIN listing l ON r.listing_id = l.id
    GROUP BY l.location_id, listing_id
)
SELECT listing_id, location_id, _rank
FROM RankedListings
WHERE _rank = 1;
```

4. **Ranking Listings by Review Scores Within Each Neighbourhood**

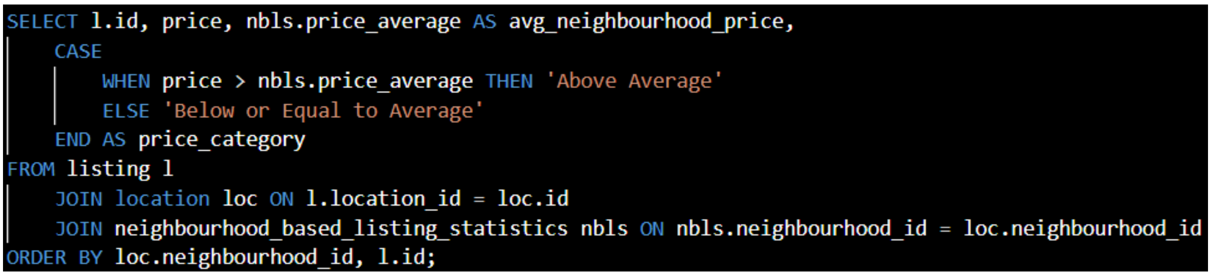
Another CTE ranks listings within each neighbourhood based on their scores. The ‘RANK()' window function sorts the listings by review_scores_rating in descending order for each neighbourhood. The main query selects information such as review_scores_rating, neighbourhood, and ranking.



```sql
WITH RankedListings AS (
    SELECT l.id AS listing_id, l.review_scores_rating, n.neighbourhood,
        RANK() OVER (
            PARTITION BY n.id
            ORDER BY
                l.review_scores_rating DESC
        ) AS ranking
    FROM listing l
        JOIN location loc ON l.location_id = loc.id
        JOIN neighbourhood n ON loc.neighbourhood_id = n.id
)
SELECT listing_id, review_scores_rating, neighbourhood, ranking
FROM RankedListings
ORDER BY neighbourhood, ranking;
```

5. **Identifying Listings with Prices Above the Neighbourhood Average**

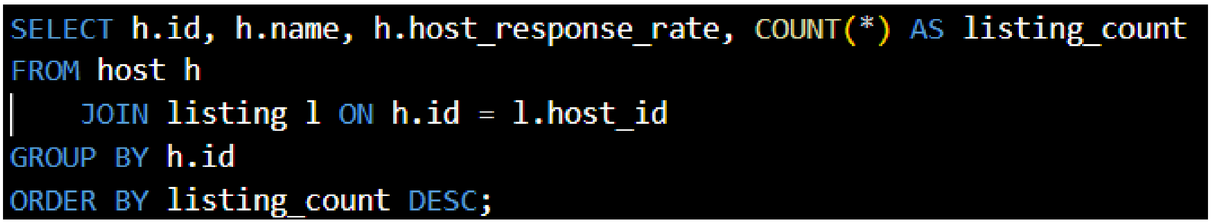
This query identifies listings with above-average prices for each neighbourhood. It joins the listings table with the neighbourhood_based_listing_statistics table to compare the prices of each listing with the average price of the neighbourhood. The result, in addition to including the price and the average price of the neighbourhood, also has a categorical column indicating whether the price is above or below the average.



```
SELECT l.id, price, nbls.price_average AS avg_neighbourhood_price,
    CASE
        WHEN price > nbls.price_average THEN 'Above Average'
        ELSE 'Below or Equal to Average'
    END AS price_category
FROM listing l
    JOIN location loc ON l.location_id = loc.id
    JOIN neighbourhood_based_listing_statistics nbls ON nbls.neighbourhood_id = loc.neighbourhood_id
ORDER BY loc.neighbourhood_id, l.id;
```

6. **Analyse host response rates and their impact on listing popularity**

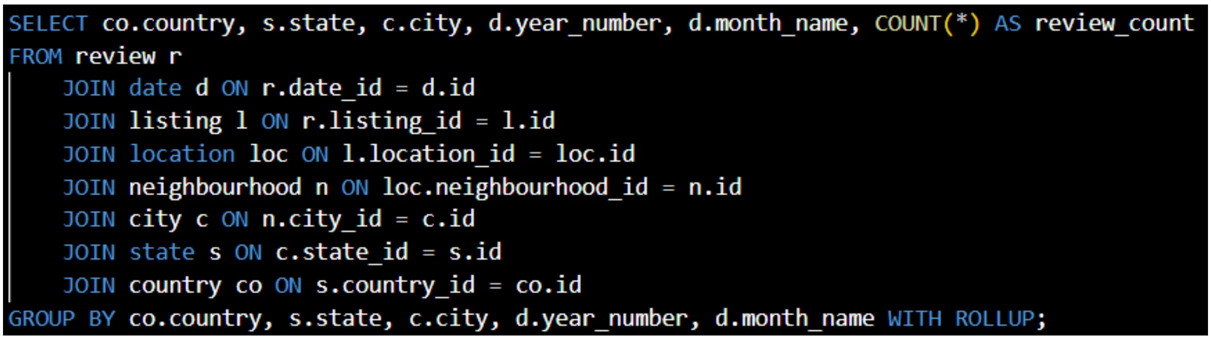
This query analyses the relationship between host_response_rate and its impact on the popularity of the listing. The result makes it possible to understand when there is a correlation between host responsiveness and the number of listings they manage.



```
SELECT h.id, h.name, h.host_response_rate, COUNT(*) AS listing_count
FROM host h
    JOIN listing l ON h.id = l.host_id
GROUP BY h.id
ORDER BY listing_count DESC;
```

7. **Analyse monthly review trends for each city, with subtotals for each year, city, state, and a grand total**

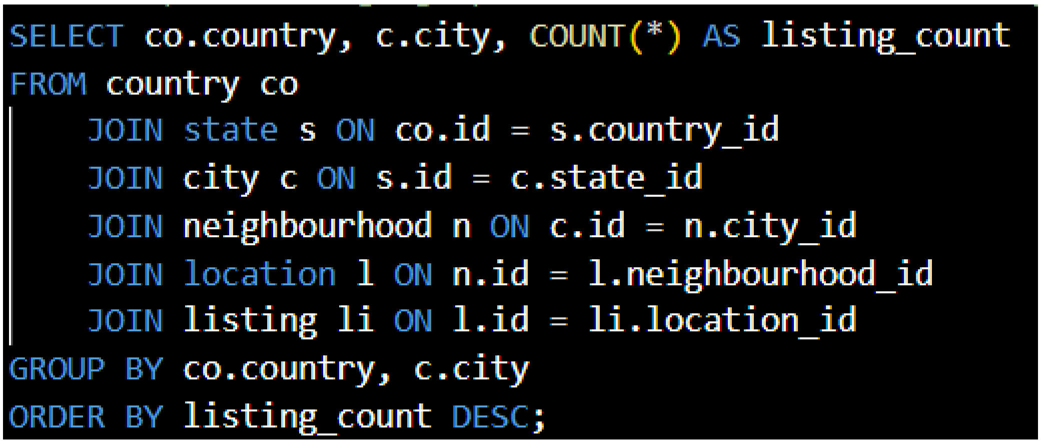
This query offers an analysis of the review's monthly trends. The data is aggregated based on country, state, city, year, and month, using the 'WITH ROLLUP' clause to include summary rows. The result shows the number of reviews for each combination to analyze trends over time and at various geographic levels.



```
SELECT co.country, s.state, c.city, d.year_number, d.month_name, COUNT(*) AS review_count
FROM review r
    JOIN date d ON r.date_id = d.id
    JOIN listing l ON r.listing_id = l.id
    JOIN location loc ON l.location_id = loc.id
    JOIN neighbourhood n ON loc.neighbourhood_id = n.id
    JOIN city c ON n.city_id = c.id
    JOIN state s ON c.state_id = s.id
    JOIN country co ON s.country_id = co.id
GROUP BY co.country, s.state, c.city, d.year_number, d.month_name WITH ROLLUP;
```

8. **Explore the geographical distribution of listings across cities and countries**

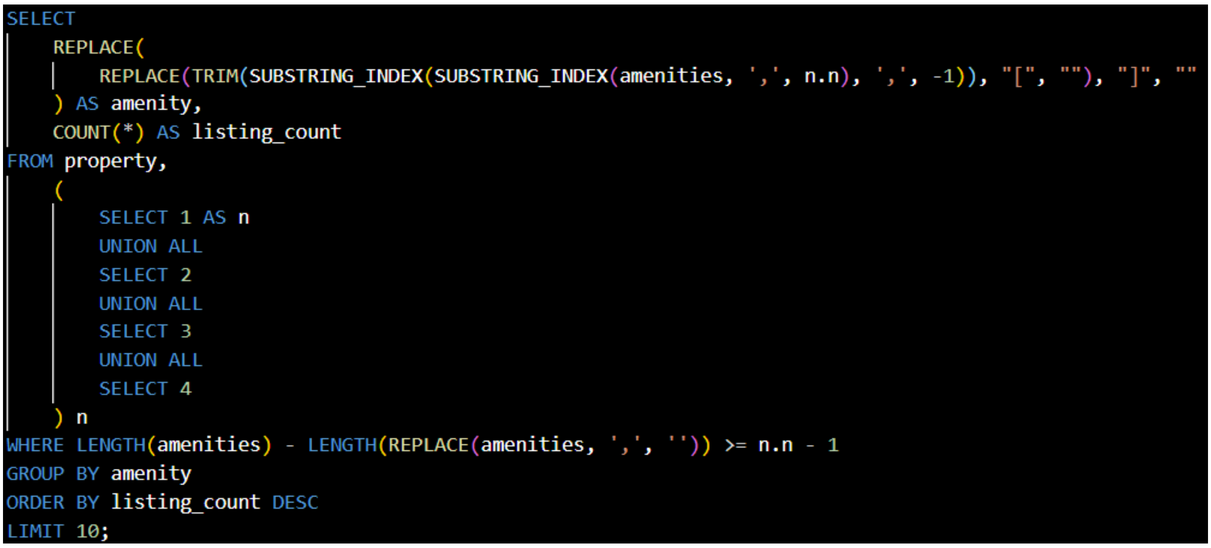
This query gives an overview of the geographical distribution of listings, counting the number of listings for each city and country. The result helps to understand the concentration of different regions, identifying popular cities.



```
SELECT co.country, c.city, COUNT(*) AS listing_count
FROM country co
    JOIN state s ON co.id = s.country_id
    JOIN city c ON s.id = c.state_id
    JOIN neighbourhood n ON c.id = n.city_id
    JOIN location l ON n.id = l.neighbourhood_id
    JOIN listing li ON l.id = li.location_id
GROUP BY co.country, c.city
ORDER BY listing_count DESC;
```

9. **Query to identify the top individual amenities, considering each amenity as a separate entity in the list**

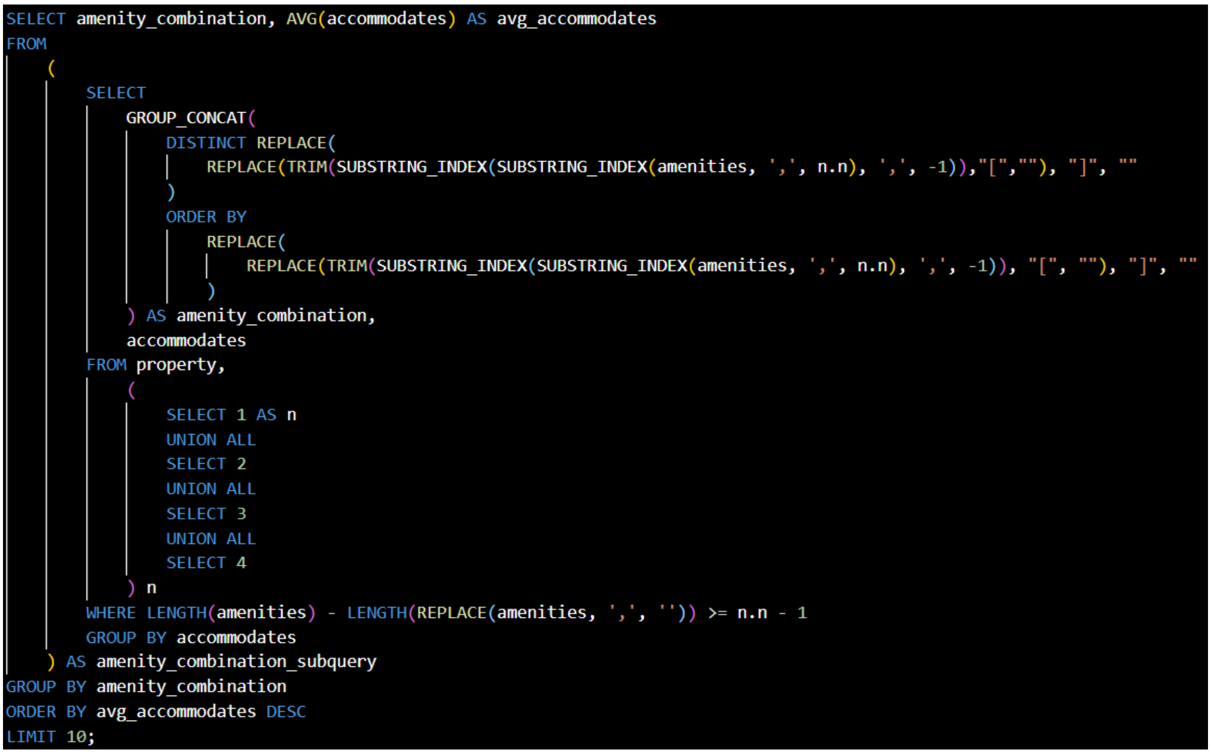
This query divides amenities into individual entities, counting their occurrences and identifying the top amenities. It provides details of the most common amenities among the listings.



```
SELECT
    REPLACE(
        REPLACE(TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(amenities, ',', n.n), ',', -1)), "[", ""), "]", ""
    ) AS amenity,
    COUNT(*) AS listing_count
FROM property,
    (
        SELECT 1 AS n
        UNION ALL
        SELECT 2
        UNION ALL
        SELECT 3
        UNION ALL
        SELECT 4
    ) n
WHERE LENGTH(amenities) - LENGTH(REPLACE(amenities, ',', '')) >= n.n - 1
GROUP BY amenity
ORDER BY listing_count DESC
LIMIT 10;
```

10. **Query to calculate the average accommodates for listings based on combinations of amenities**

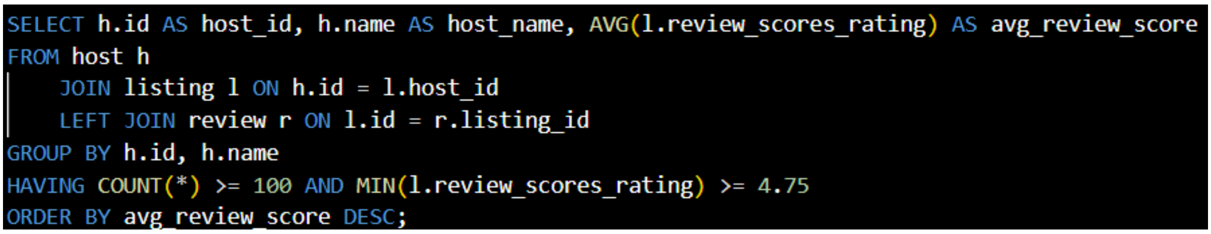
This query explores the relationship between average accommodations value and different combinations of amenities. This analysis can uncover patterns in the number of accommodations associated with a specific combination of amenities.



```sql
SELECT amenity_combination, AVG(accommodates) AS avg_accommodates
FROM
    (
        SELECT
            GROUP_CONCAT(
                DISTINCT REPLACE(
                    REPLACE(TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(amenities, ',', n.n), ',', -1)),"[",""), "]", ""
                )
                ORDER BY
                    REPLACE(
                        REPLACE(TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(amenities, ',', n.n), ',', -1)), "[", ""), "]", ""
                    )
            ) AS amenity_combination,
            accommodates
        FROM property,
            (
                SELECT 1 AS n
                UNION ALL
                SELECT 2
                UNION ALL
                SELECT 3
                UNION ALL
                SELECT 4
            ) n
        WHERE LENGTH(amenities) - LENGTH(REPLACE(amenities, ',', '')) >= n.n - 1
        GROUP BY accommodates
    ) AS amenity_combination_subquery
GROUP BY amenity_combination
ORDER BY avg_accommodates DESC
LIMIT 10;
```

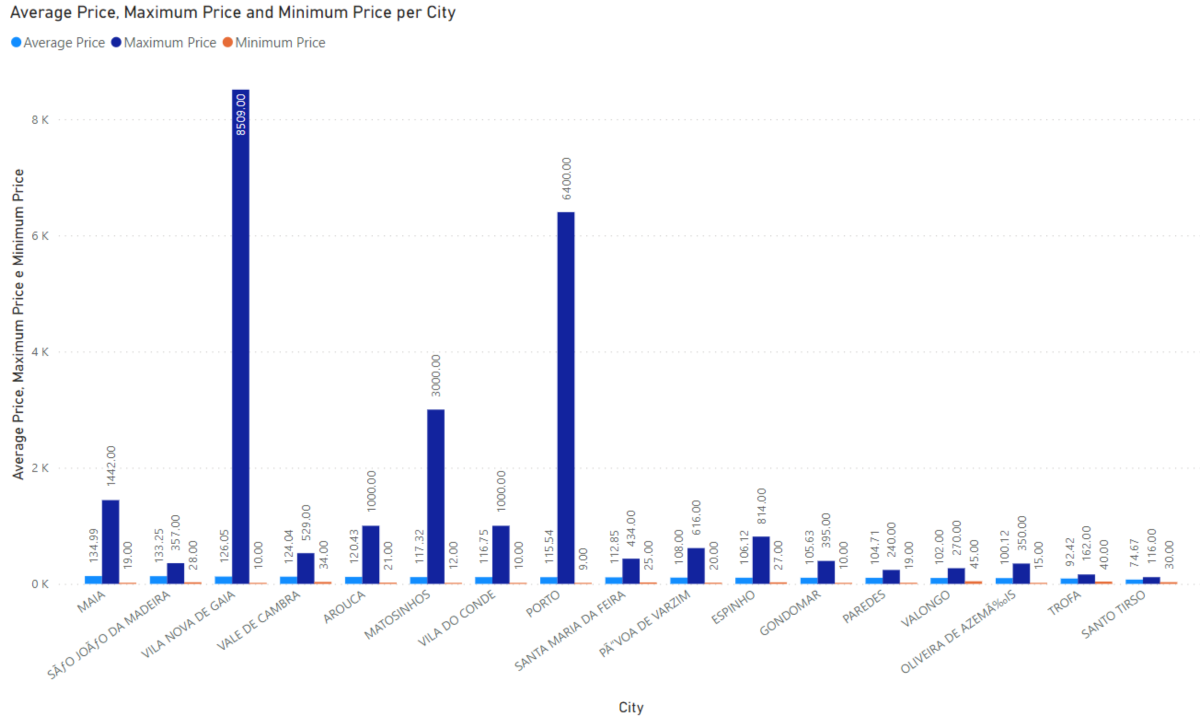
11. Query to identify hosts whose listings consistently receive high review scores

This query identifies hosts whose listings consistently receive high review scores. The 'HAVING' clause ensures that only hosts with a substantial number of reviews (at least 100) and a minimum average review score of 4.75 are included. This query provides valuable insight into the hosts that are most consistent in giving good experiences to guests.

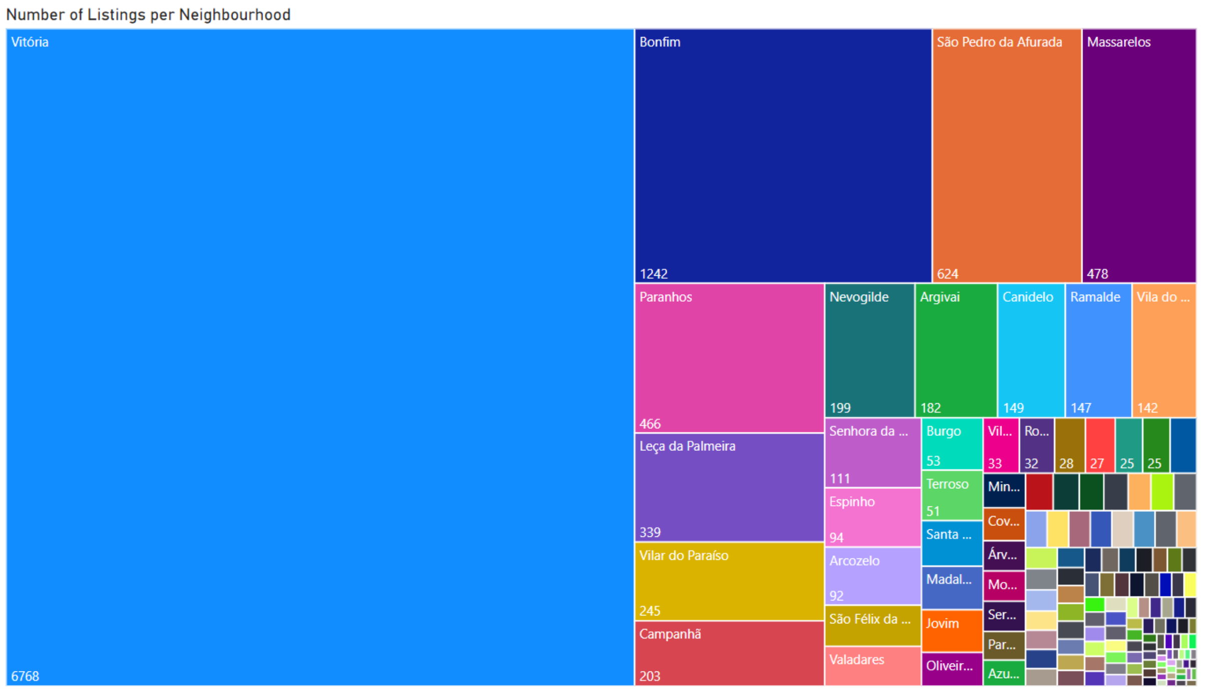


```sql
SELECT h.id AS host_id, h.name AS host_name, AVG(l.review_scores_rating) AS avg_review_score
FROM host h
    JOIN listing l ON h.id = l.host_id
    LEFT JOIN review r ON l.id = r.listing_id
GROUP BY h.id, h.name
HAVING COUNT(*) >= 100 AND MIN(l.review_scores_rating) >= 4.75
ORDER BY avg_review_score DESC;
```

In terms of data analysis, we decided to do 5 visualisations using Power BI that could show various uses of our data:

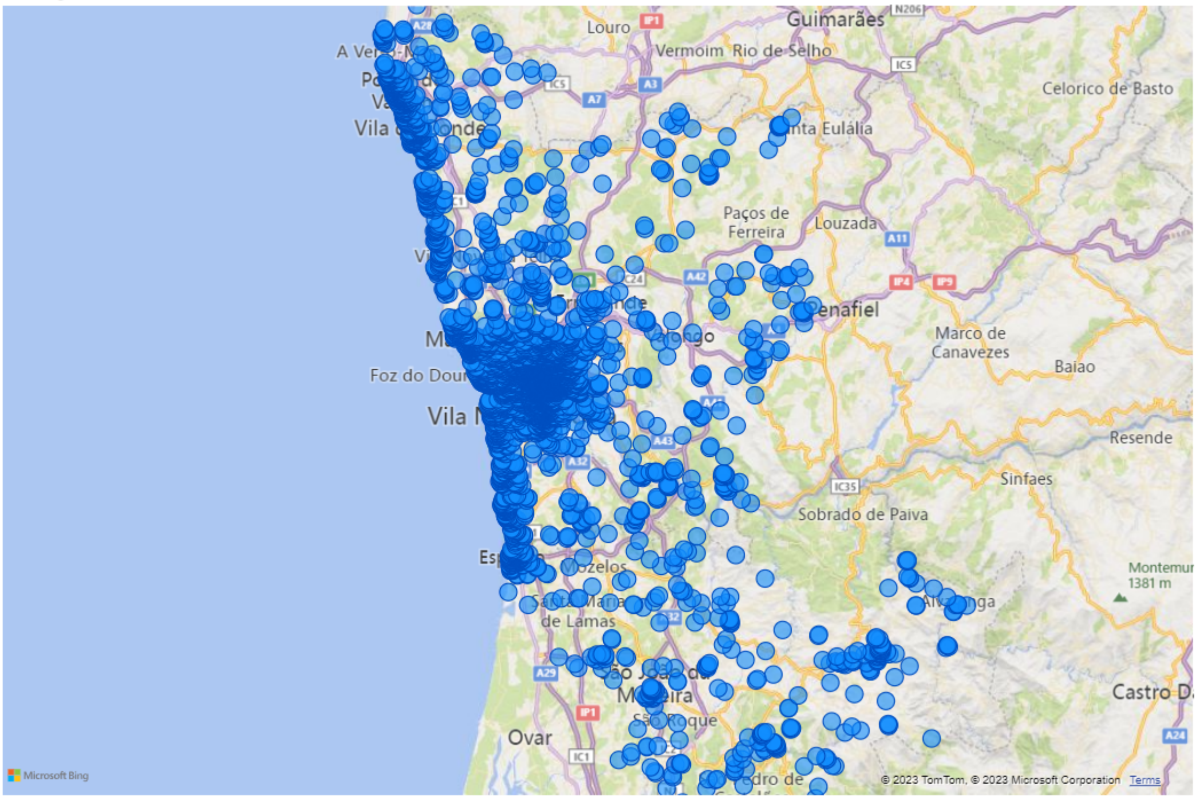

Average Price, Maximum Price and Minimum Price per City

With the first graph (Average Price, Maximum Price and Minimum Price per City), we can see and analyse the range of prices in the cities our data has, coming directly from our aggregated fact table, and can conclude, for example, that while “Vila Nova de Gaia” has the listing with the highest price, on average it isn’t the most expensive city, that one being “Maia”.

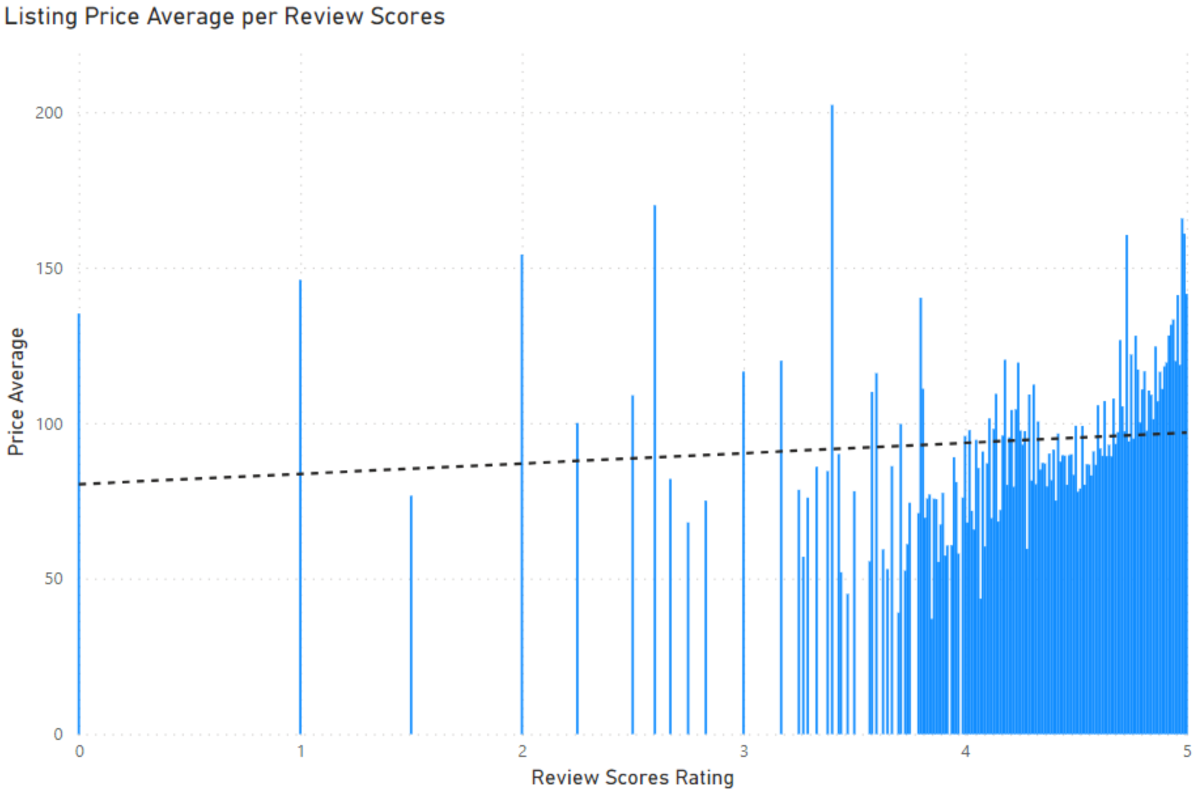

Number of Listings per Neighbourhood

With the treemap Number of Listings per Neighbourhood, we can see the number of listings in the various neighbourhoods present in our data and see how “Vitória” seems to account for around half of our total Airbnb listings, probably indicating that it is a very enticing place for tourists.

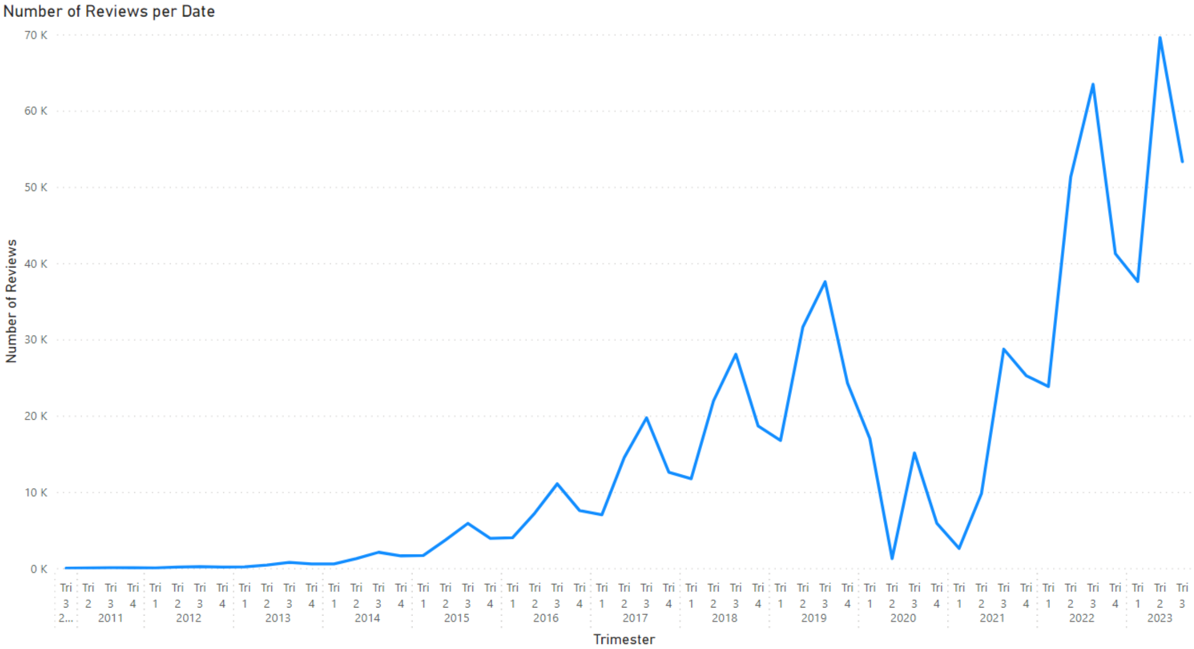
Listings Distribution



This map (Listings Distribution) shows us the distributions of listings around the world, though our data set is limited to a smaller region, and we can see how certain zones have a much higher density of listings when compared to others.



With this graph (Listing Price Average per Review Scores), we tried to see if there was a correlation between prices and ratings, thinking we would find higher-rated listings with a higher price, and while there's a slight trend of it going upward as the ratings increase, it isn't as drastic as we were expecting.



Lastly, even though we don't have data about when listings were created or when a guest is in one, we wanted to see if we could see any trends regarding the popularity of Airbnb over the years. Using the number of reviews by date, we see a continuing trend of the peak of the year being around the 3rd quarter of the year (June, July, and September) and that even though there were a couple of years where the number of reviews went down instead of up, most likely due to COVID, the number of reviews has been steadily increasing, meaning there are more listings and/or guests as the years have gone by. Of course, this is all with the assumption that a higher number of guests/listings would result in a higher number of reviews.

Critical reflection about the advantages and shortcomes with respect to the operational databases

The strength of the dimensional model lies in its ability to support detailed and granular analysis. By including dimensions like property, host, date, and location, the model empowers users to explore specific facets of the dataset with a high level of detail, for example, how many properties there are in a specific time frame and location. This granularity is essential for extracting nuanced insights and understanding various data elements in depth. In addition, the hierarchical representation of geographic dimensions, ranging from the broad level of the country down to the detailed level of neighbourhood, proves instrumental in providing geospatial insights. This structure facilitates a comprehensive analysis of location-based information related to property listings.

Regarding the shortcomings of the dimensional model, the denormalized structure of the model introduces a potential concern regarding data redundancy, particularly notable in tables such as *neighbourhood_based_listing_statistics*. While data redundancy is often intentional for query performance optimization, it raises considerations about storage efficiency. Moreover, the model demonstrates limited support for real-time updates. Its inherent focus on analytical processing makes it less optimal for scenarios requiring immediate and dynamic updates or transactional requirements. In such cases, the model may not align with the need for real-time responsiveness.

Conclusion

In conclusion, with the Airbnb Insights project we have successfully implemented a robust data warehouse tailored for detailed analysis of property rental data within the city of Porto, Portugal. The dimensional model, with its five dimensions (reviewer, date, property, host, location) and three fact tables (listing, review, and location-based listing statistics), offers a structured framework for analysis. The chosen data sources, extraction, transformation, and loading processes using Python and MySQL contribute to a seamless and effective data management strategy.
The queries made for data analysis cover a spectrum of dimensions, offering valuable insights into host performance, review trends, pricing dynamics, and geographical distribution. Furthermore, the integration of Power BI visualizations expands the project's analytical scope.

The complete codebase for the project can be accessed through the link diogof19/AID-PROJ (github.com), and the transformed dataset is available here.