ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MSc in Business Analytics - Part time

Data Management & Business Intelligence

2nd Assignment

"Airbnb Report"

Professor:

Mr. Chatziantoniou Damianos

Gkourioti Panagiota – P2822109

Koletsi Thaleia – P2822120

Table of Contents

1. Introduction

The scope of this report is to design and develop a data warehouse, build a data cube on top of it, develop some OLAP reports and visualize our results based on an existing business model of Airbnb. This was accomplished by using SQL Server Database, SQL Server Analysis Services and Power BI of Microsoft as presented below in detail.

2. Case description

Airbnb is an online community-based hospitality marketplace aiming to connect people who want to rent out their homes with people who are looking for accommodation in that location, without owning any rooms itself.

Airbnb operates as a transaction facilitator between hosts and travelers who are looking for comfortable accommodation at a cheap price. With over 1,500,000 listings in 34,000 cities and 190 countries, its wide coverage enables travelers to rent private homes all over the world. Personal profiles as well as a rating and reviewing system provide information about the host and what is on offer. Profiles and user reviews help to create reputation and trust among participants of the marketplace. Additionally, Airbnb has several revenue streams, all connected to booking stays and experiences.

The company aims to profitably utilize the data in order to improve its business operations, to raise the quality of customer services, and overall, to be more competitive in business industry in the long run.

To achieve this research objective, we make use of various analytical tools and technologies, which help us provide answers to core issues that concern the company. We gained access to data for two cities, Amsterdam and Berlin with house listings information, rental calendars and customer reviews for management and processing.

Starting with designing the BI system and workflow, we proceed to data cleaning and transformation, building and deploying a data cube, and conduct analytical calculations to provide useful statistics about properties. Through the aforementioned analysis we aim to produce dashboards and turn raw data into actionable business insights which will support and successfully navigate the company stakeholders through their professional pathway.

3. BI Process overview

The scope of the project is to answer questions related to the prices and ratings of the properties in the system as well as the hosts, combined with the factors that affect them.

In particular, the results of the project will show the most expensive properties, the ones with the highest scores and their characteristics, average statistics and the most popular/busy hosts.

For the creation of the BI system, specific techniques were applied, differentiated in many stages. At first, the ETL process was followed, which is a type of data integration referring to the three steps (extract, transform, load). The data was extracted in CSV format. These files were then imported into the MS SQL Server and the database was created.

After moving to the staging area and doing the necessary data transformations, we proceed with creating the Data Warehouse. At this point, decisions are made to determine the dimensions of the tables and the fact tables. Once these tables are created, the data types of their columns are defined and the corresponding foreign keys for their connection are assigned.

The next step is to fill the Data Warehouse with the corresponding final data. Once the correct combinations are made and the loading of the Data Warehouse is completed, the next step is the connection with SQL Server Analysis Services for the development of multidimensional cubes. There, we load the cube with the Data Warehouse, we do some calculations, create measures and process the cube.

The final step of visualization is performed with the help of Power BI. Power BI is used to create live dashboards, to analyze and find the answers regarding our questions/assumptions while being connected live to the multidimensional cube.

The expected answers will be displayed in a live dashboard and will be determined according to some characteristics. Those characteristics are going to be used as filters for the display and grouping of top priced / top rated properties and best hosts respectively.

4. Data Source

After the purpose of conducting the analysis had been defined, we proceeded to collect the data needed. The dataset was extracted from Kaggle.com, an online platform that offers access to public datasets and allows users to explore and build models in a web-based data-science environment.

For this project we used the Airbnb dataset, which contains data for two cities, Amsterdam and Berlin with 44,799 house listings in total along with several of their features, user reviews as well as rental calendars with 16,351,640 daily prices. The data was split into four csv files, two for each city. One csv contains each listing's characteristics and the other one contains a calendar with daily prices for all listings and availability information.

4.1. Description of dataset

The dataset consists of 226 columns, 84 of data type string, 65 integers, 22 booleans and other. Specifically, the columns that we included in our analysis are:

- id
- host_id
- host_name
- host_response_rate
- host_acceptance_rate

- host_is_superhost
- host_total_listings_count
- city
- latitude
- longitude
- property_type
- room_type
- accommodates
- bathrooms
- bedrooms
- amenities
- cleaning_fee
- extra_people
- availability_365
- number_of_reviews
- review_scores_rating
- review_scores_accuracy
- review_scores_cleanliness
- review_scores_checkin
- review_scores_communication
- review_scores_location
- review_scores_value
- instant_bookable
- cancellation_policy
- 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
- price
- date
- minimum_nights
- maximum_nights
- availability

5. Data Analysis

After the crucial step of finding and selecting the dataset, our analysis included the following stages:

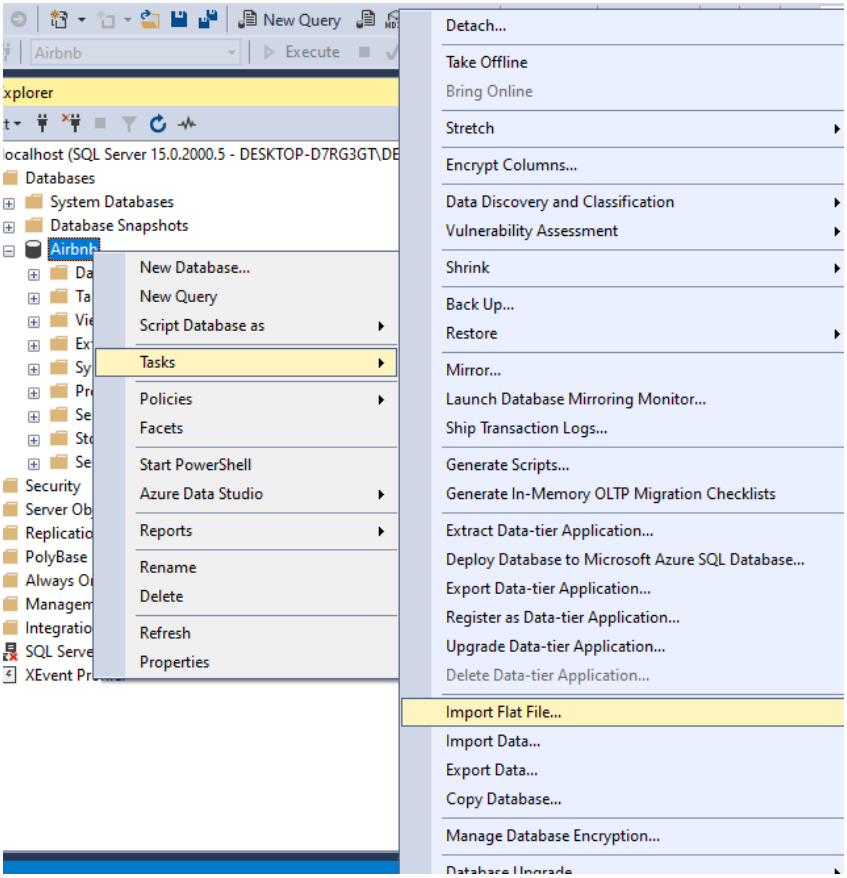
1. Data import and staging
2. Data Cleaning and Transformation
3. Designing and developing a data warehouse
4. Building and deploying a data cube
5. Developing some OLAP reports
6. Visualizing results

5.1. Data import and staging

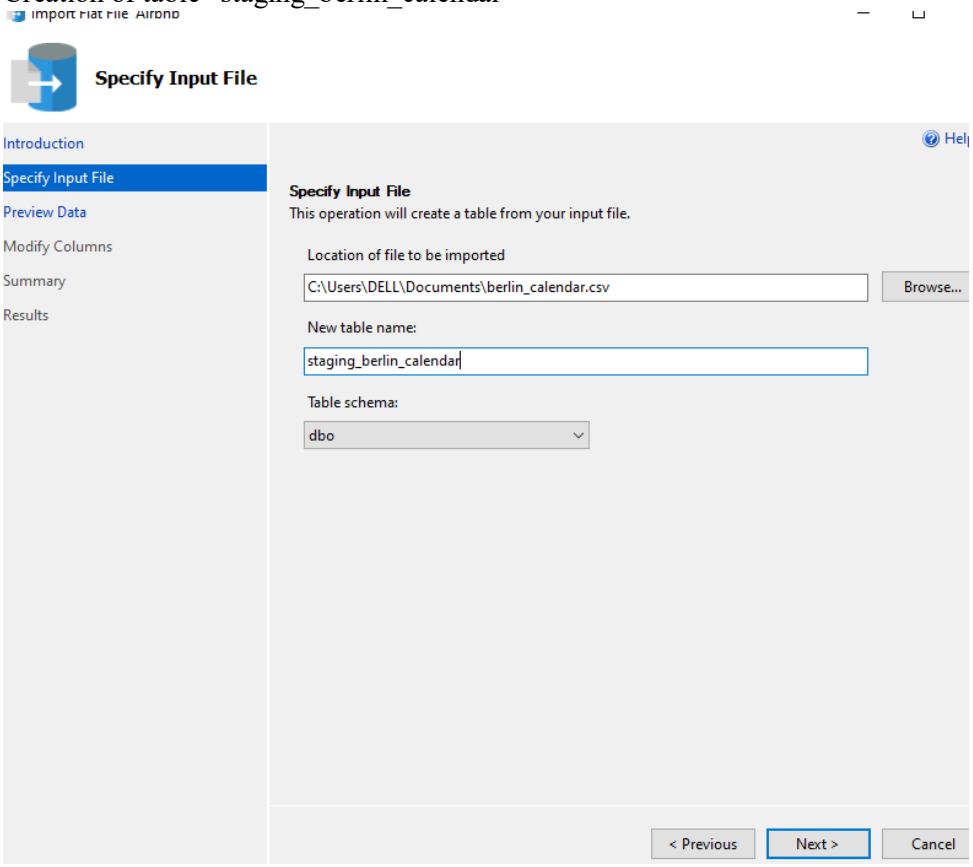
The initial step of our data analysis was to connect to Microsoft SQL Server Management Studio and create a database. We then imported the csv files into the database and created staging tables. The purpose of the staging area is to load data from the data source, modify & cleanse them before we final load them into the Data Warehouse.

This process is described below with illustrative screenshots:

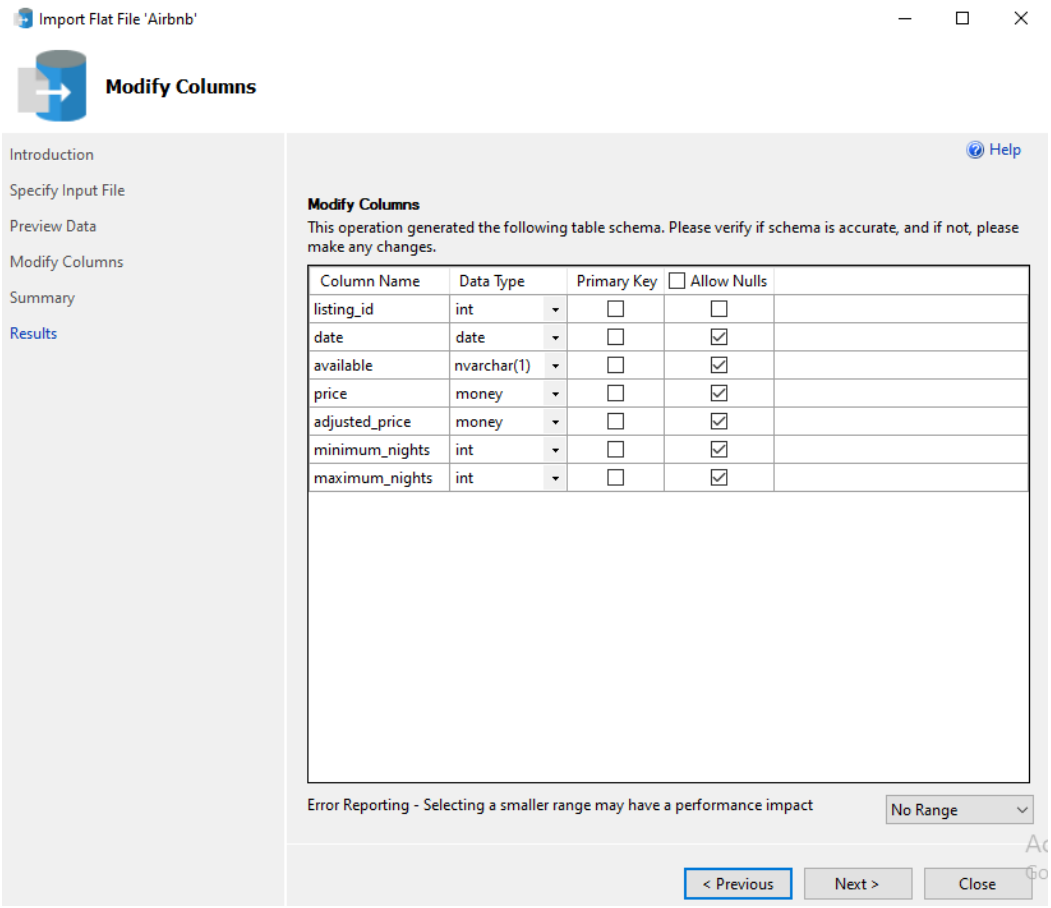
1. Importing csv files



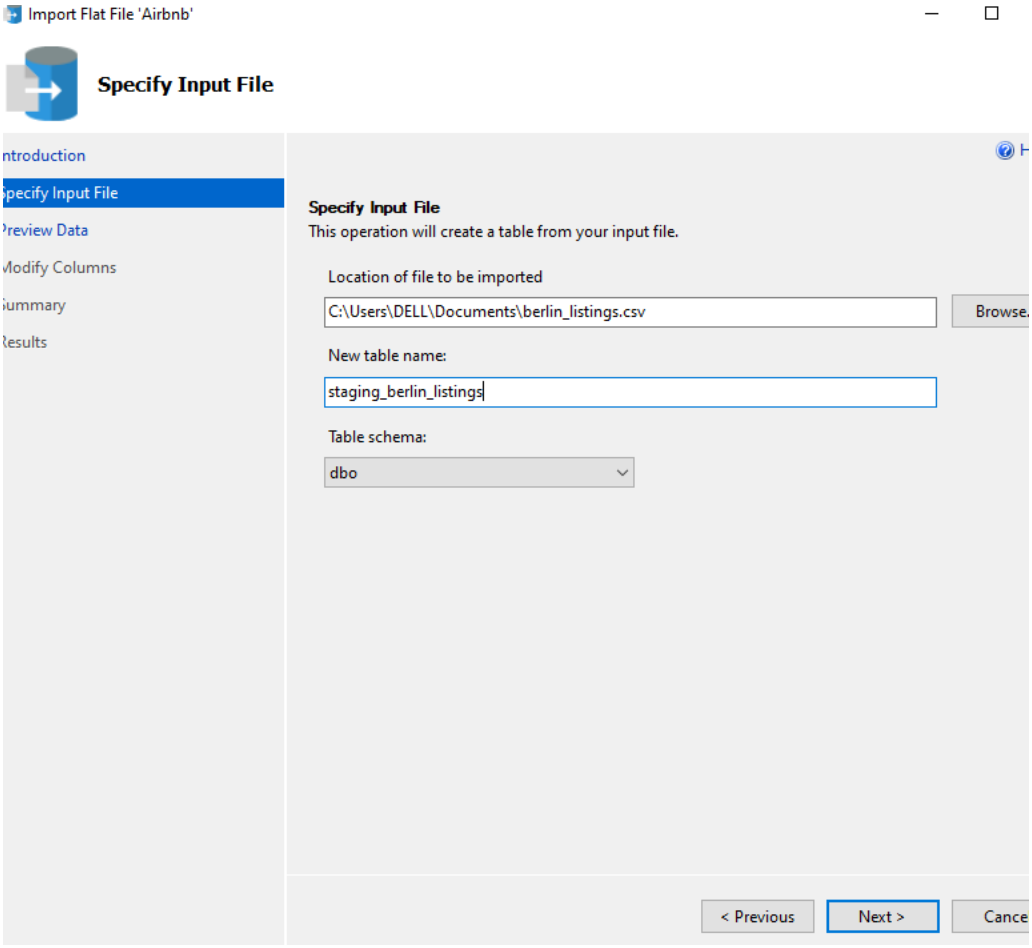
2. Creation of table “staging_berlin_calendar”



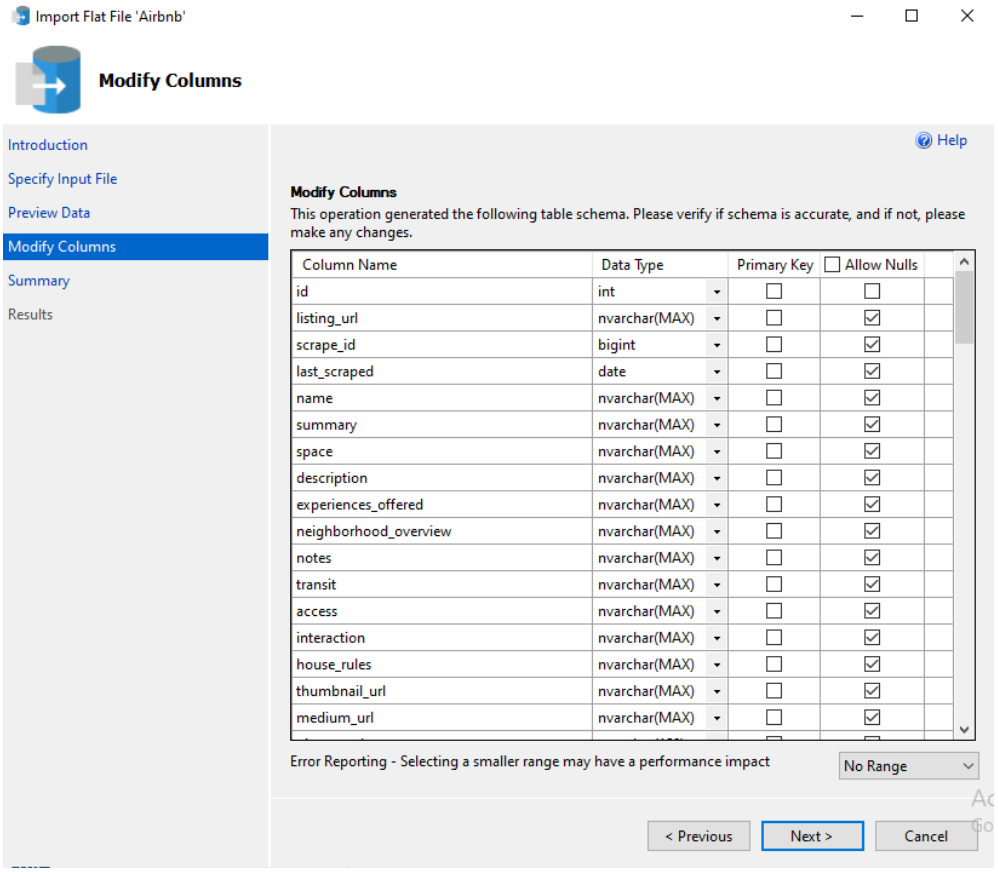
3. Modifying columns and data types

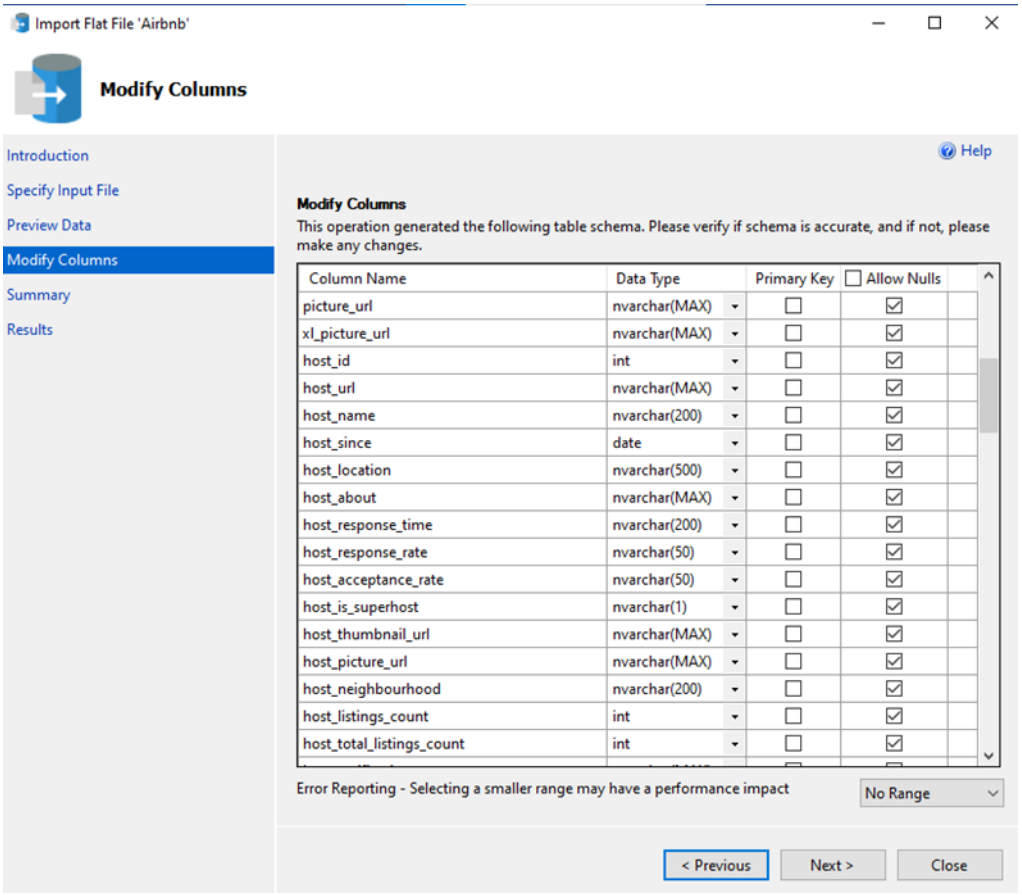


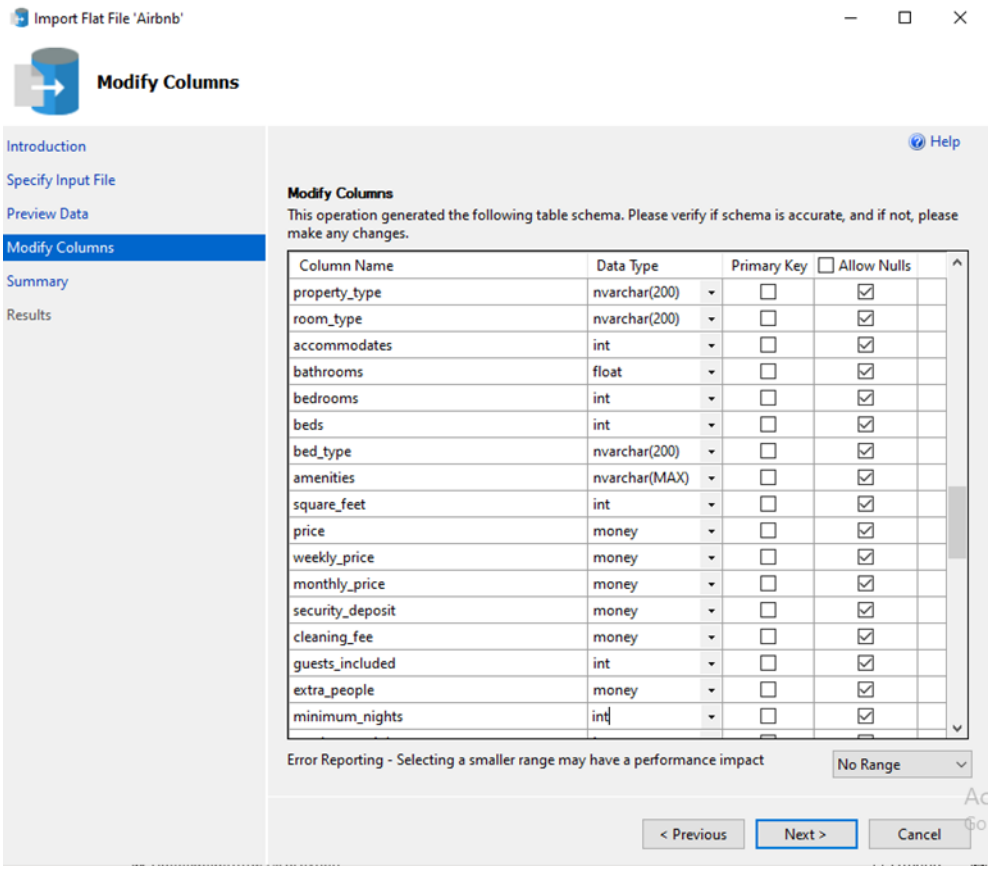
4. Creation of table “staging_berlin_listings”

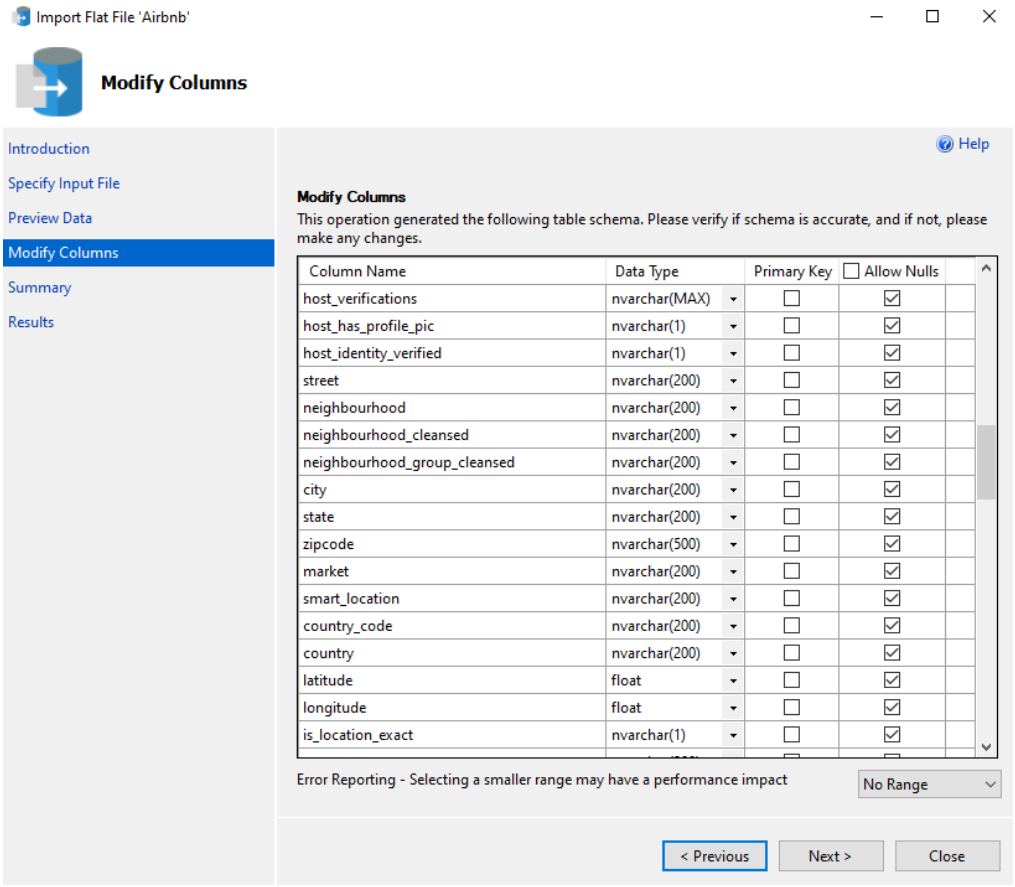


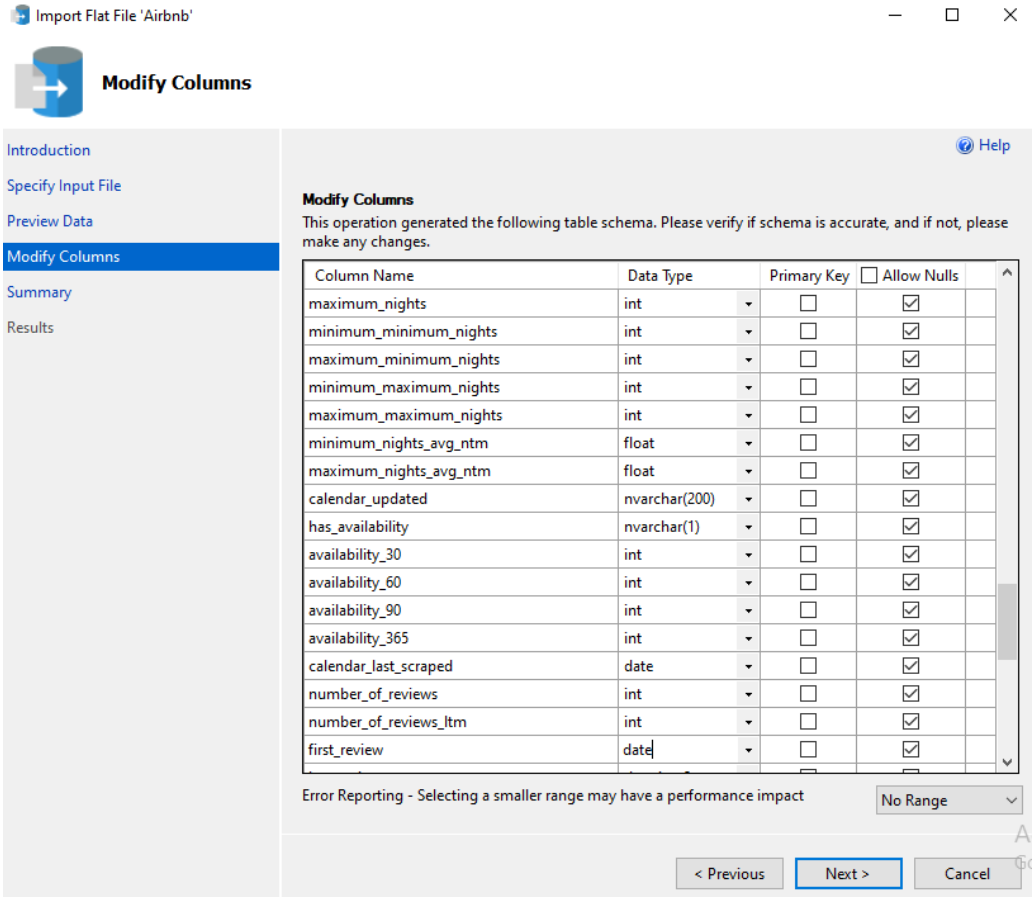
5. Modifying columns and data types

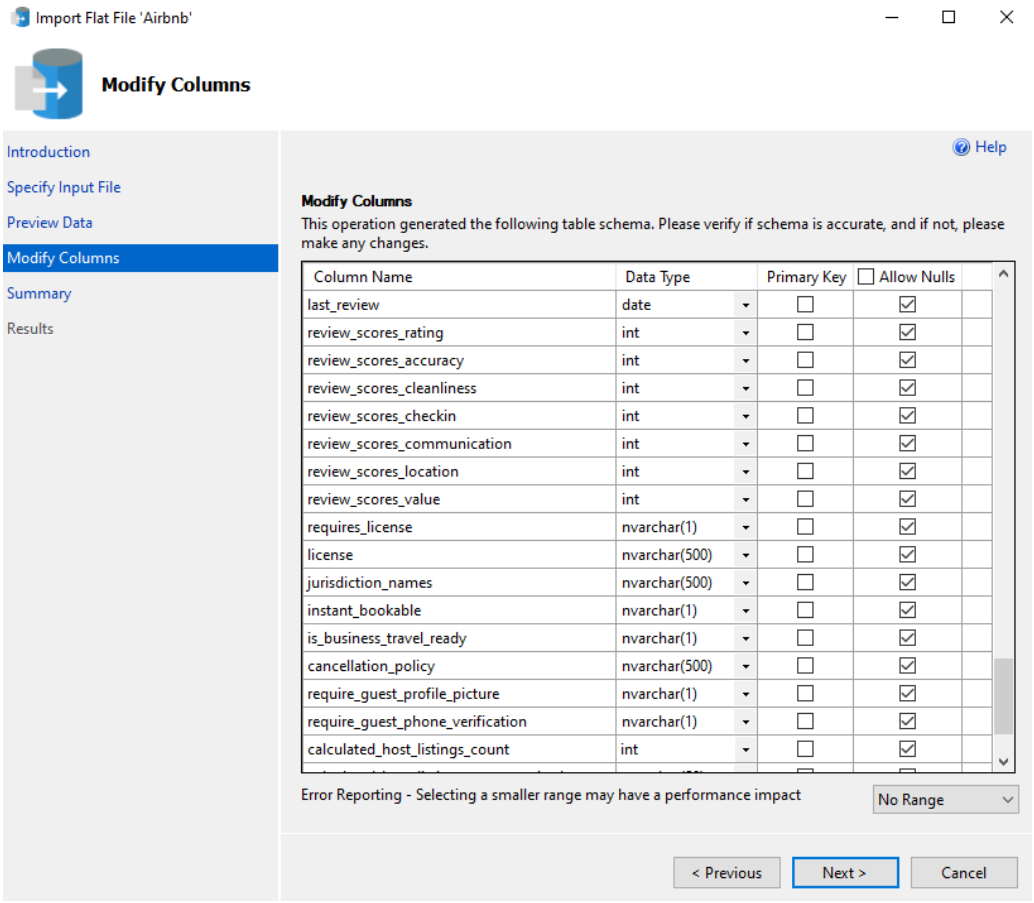


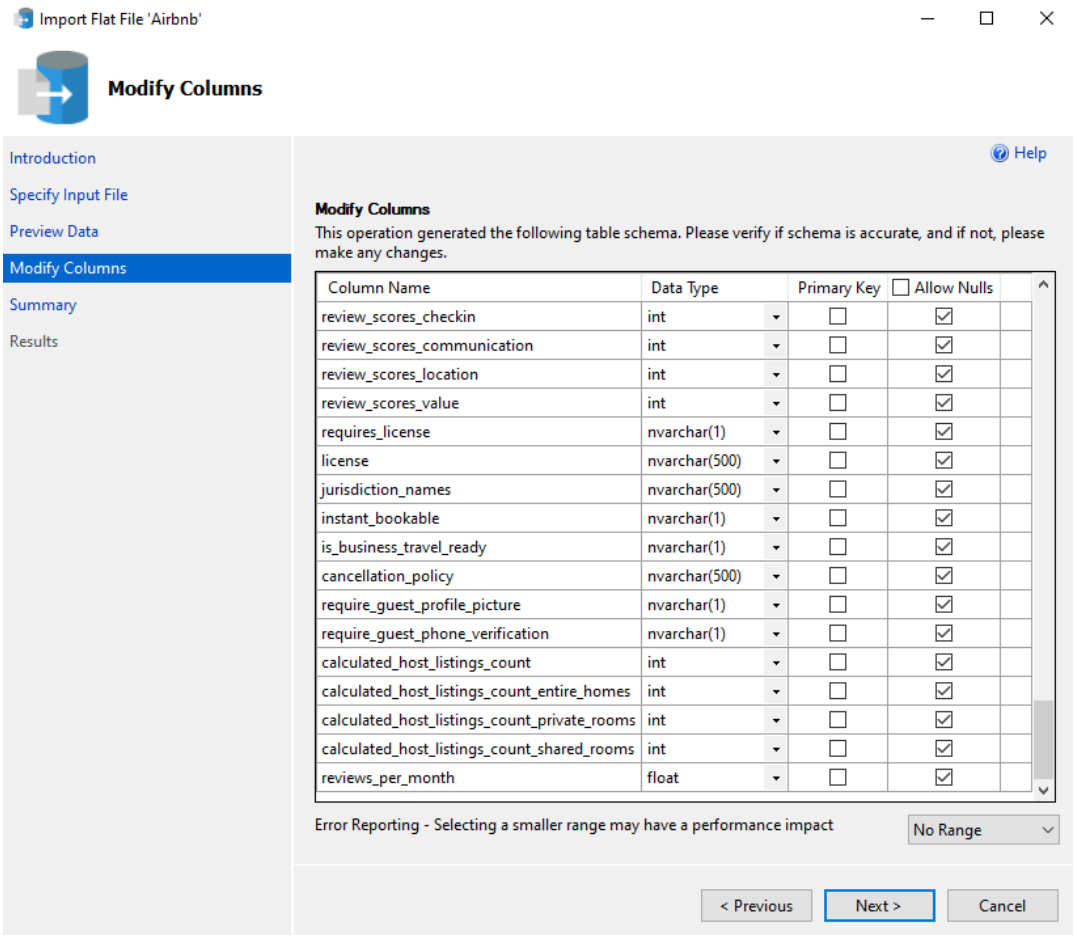

Import Flat File 'Airbnb'
Modify Columns
Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results
Help
Modify Columns
This operation generated the following table schema. Please verify if schema is accurate, and if not, please make any changes.
Column Name | Data Type | Primary Key | Allow Nulls
picture_url nvarchar(MAX)
xl_picture_url nvarchar(MAX)
host_id int
host_url nvarchar(MAX)
host_name nvarchar(200)
host_since date
host_location nvarchar(500)
host_about nvarchar(MAX)
host_response_time nvarchar(200)
host_response_rate nvarchar(50)
host_acceptance_rate nvarchar(50)
host_is_superhost nvarchar(1)
host_thumbnail_url nvarchar(MAX)
host_picture_url nvarchar(MAX)
host_neighbourhood nvarchar(200)
host_listings_count int
host_total_listings_count int
Error Reporting - Selecting a smaller range may have a performance impact
No Range
< Previous
Next >
Close


Import Flat File 'Airbnb'
Modify Columns
Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results
Help
Modify Columns
This operation generated the following table schema. Please verify if schema is accurate, and if not, please make any changes.
Column Name | Data Type | Primary Key | Allow Nulls
property_type nvarchar(200)
room_type nvarchar(200)
accommodates int
bathrooms float
bedrooms int
beds int
bed_type nvarchar(200)
amenities nvarchar(MAX)
square_feet int
price money
weekly_price money
monthly_price money
security_deposit money
cleaning_fee money
guests_included int
extra_people money
minimum_nights int
Error Reporting - Selecting a smaller range may have a performance impact
No Range
< Previous
Next >
Cancel


Import Flat File 'Airbnb'
Modify Columns
Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results
Help
Modify Columns
This operation generated the following table schema. Please verify if schema is accurate, and if not, please make any changes.
Column Name
Data Type
Primary Key
Allow Nulls
host_verifications nvarchar(MAX)
host_has_profile_pic nvarchar(1)
host_identity_verified nvarchar(1)
street nvarchar(200)
neighbourhood nvarchar(200)
neighbourhood_cleansed nvarchar(200)
neighbourhood_group_cleansed nvarchar(200)
city nvarchar(200)
state nvarchar(200)
zipcode nvarchar(500)
market nvarchar(200)
smart_location nvarchar(200)
country_code nvarchar(200)
country nvarchar(200)
latitude float
longitude float
is_location_exact nvarchar(1)
Error Reporting - Selecting a smaller range may have a performance impact
No Range
< Previous
Next >
Close


Import Flat File 'Airbnb'
Modify Columns
Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results
Help
Modify Columns
This operation generated the following table schema. Please verify if schema is accurate, and if not, please make any changes.
Column Name
Data Type
Primary Key
Allow Nulls
maximum_nights int
minimum_minimum_nights int
maximum_minimum_nights int
minimum_maximum_nights int
maximum_maximum_nights int
minimum_nights_avg_ntm float
maximum_nights_avg_ntm float
calendar_updated nvarchar(200)
has_availability nvarchar(1)
availability_30 int
availability_60 int
availability_90 int
availability_365 int
calendar_last_scraped date
number_of_reviews int
number_of_reviews_ltm int
first_review date
Error Reporting - Selecting a smaller range may have a performance impact
No Range
< Previous
Next >
Cancel


Import Flat File 'Airbnb'
Modify Columns
Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results
Help
Modify Columns
This operation generated the following table schema. Please verify if schema is accurate, and if not, please make any changes.
Column Name
Data Type
Primary Key
Allow Nulls
last_review date
review_scores_rating int
review_scores_accuracy int
review_scores_cleanliness int
review_scores_checkin int
review_scores_communication int
review_scores_location int
review_scores_value int
requires_license nvarchar(1)
license nvarchar(500)
jurisdiction_names nvarchar(500)
instant_bookable nvarchar(1)
is_business_travel_ready nvarchar(1)
cancellation_policy nvarchar(500)
require_guest_profile_picture nvarchar(1)
require_guest_phone_verification nvarchar(1)
calculated_host_listings_count int
Error Reporting - Selecting a smaller range may have a performance impact
No Range
< Previous
Next >
Cancel


Import Flat File 'Airbnb'
Modify Columns
Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results
Help
Modify Columns
This operation generated the following table schema. Please verify if schema is accurate, and if not, please make any changes.
Column Name
Data Type
Primary Key
Allow Nulls
review_scores_checkin int
review_scores_communication int
review_scores_location int
review_scores_value int
requires_license nvarchar(1)
license nvarchar(500)
jurisdiction_names nvarchar(500)
instant_bookable nvarchar(1)
is_business_travel_ready nvarchar(1)
cancellation_policy nvarchar(500)
require_guest_profile_picture nvarchar(1)
require_guest_phone_verification nvarchar(1)
calculated_host_listings_count int
calculated_host_listings_count_entire_homes int
calculated_host_listings_count_private_rooms int
calculated_host_listings_count_shared_rooms int
reviews_per_month float
Error Reporting - Selecting a smaller range may have a performance impact
No Range
< Previous
Next >
Cancel

The same procedure is followed for staging_amsterdam_calendar and staging_amsterdam_listings. After loading the data from the sources in the staging area, some modifications and transformations are performed before they are finally loaded into the data warehouse.

5.2. Data Cleaning and Transformation

The process of data cleaning is the middle step of ETL (extract, transform, load) process and is a very crucial part of the analysis. Analyzing information requires structured and accessible data for best results. Data transformation enables us to alter the structure and format of raw data as needed in order to produce more meaningful outcomes. It may be constructive (adding, replicating data), destructive (deleting fields and records), aesthetic and structural (renaming, combining, summarizing columns in a database).

This procedure facilitates the data analysis and yields several benefits. Properly formatted and validated data improves data quality and ensures compatibility between applications, systems, and different types of data. It prevents errors or other issues from occurring, because of null values, unexpected duplicates, incorrect indexing or incompatible formats. Specifically, for our assignment, the following alterations were made to the initial staging tables:

1. We began by removing the unwanted columns from the database, as they included unnecessary information for our study and were of no interest regarding the scenarios we examine.

```
USE Airbnb;

ALTER TABLE staging_amsterdam_listings
DROP COLUMN scrape_id , summary, [space], [description], listing_url,last_scraped,[name],[description],neighborhood_overview,
notes,transit,access,interaction,house_rules,experiences_offered, thumbnail_url, medium_url, xl_picture_url,requires_license,license,jurisdiction_names,
square_feet,calendar_last_scraped,first_review,last_review,host_verifications,host_url,host_since,host_location,host_about,host_response_time,host_thumb
host_picture_url,host_neighbourhood,host_listings_count,host_total_listings_count,host_verifications,host_has_profile_pic,host_identity_verified,
street,neighbourhood_cleansed,neighbourhood_group_cleansed,[state],zipcode,market,smart_location,country_code,is_location_exact,square_feet,price,
weekly_price,monthly_price,security_deposit,guests_included,minimum_nights,maximum_nights,minimum_minimum_nights,maximum_minimum_nights,minimum_maximum_
minimum_nights_avg_ntm,maximum_nights_avg_ntm,calendar_updated,has_availability,availability_30,availability_60,availability_90,number_of_reviews_ltm,
instant_bookable,is_business_travel_ready,require_guest_profile_picture,require_guest_phone_verification
;

ALTER TABLE staging_berlin_listings
DROP COLUMN scrape_id , summary, [space], [description], listing_url,last_scraped,[name],[description],neighborhood_overview,
notes,transit,access,interaction,house_rules,experiences_offered, thumbnail_url, medium_url, xl_picture_url,requires_license,license,jurisdiction_names,
square_feet,calendar_last_scraped,first_review,last_review,host_verifications,host_url,host_since,host_location,host_about,host_response_time,host_thumb
host_picture_url,host_neighbourhood,host_listings_count,host_total_listings_count,host_verifications,host_has_profile_pic,host_identity_verified,
street,neighbourhood_cleansed,neighbourhood_group_cleansed,[state],zipcode,market,smart_location,country_code,is_location_exact,square_feet,price,
weekly_price,monthly_price,security_deposit,guests_included,minimum_nights,maximum_nights,minimum_minimum_nights,maximum_minimum_nights,minimum_maximum_
minimum_nights_avg_ntm,maximum_nights_avg_ntm,calendar_updated,has_availability,availability_30,availability_60,availability_90,number_of_reviews_ltm,
instant_bookable,is_business_travel_ready,require_guest_profile_picture,require_guest_phone_verification
;
```

2. Subsequently, we decided how to handle missing values. The N/A values were replaced with nulls for the continuous variables in order to exclude them from aggregations and the opposite was done for the categorical variables.

```
UPDATE staging_amsterdam_listings SET host_acceptance_rate = NULL
WHERE host_acceptance_rate = 'N/A';

UPDATE staging_amsterdam_listings SET host_response_rate = NULL
WHERE host_response_rate = 'N/A';

UPDATE staging_berlin_listings SET host_acceptance_rate = NULL
WHERE host_acceptance_rate = 'N/A';

UPDATE staging_berlin_listings SET host_response_rate = NULL
WHERE host_response_rate = 'N/A';
```

```
UPDATE staging_amsterdam_listings
SET host_is_superhost='n'
WHERE host_is_superhost is null;

UPDATE staging_amsterdam_listings
SET [host_name]='n'
WHERE [host_name] is null;

UPDATE staging_berlin_listings
SET host_is_superhost='n'
WHERE host_is_superhost is null;

UPDATE staging_berlin_listings
SET [host_name]='n'
WHERE [host_name] is null;
```

3. A required step was to convert acceptance and response rates to numeric data types because the percentages were stored as strings.

```
UPDATE staging_amsterdam_listings SET host_acceptance_rate = REPLACE(host_acceptance_rate, '%', '');

ALTER TABLE staging_amsterdam_listings
ALTER COLUMN host_acceptance_rate int;

UPDATE staging_amsterdam_listings SET host_response_rate = REPLACE(host_response_rate, '%', '');

ALTER TABLE staging_amsterdam_listings
ALTER COLUMN host_response_rate int;

UPDATE staging_berlin_listings SET host_acceptance_rate = REPLACE(host_acceptance_rate, '%', '');

ALTER TABLE staging_berlin_listings
ALTER COLUMN host_acceptance_rate int;

UPDATE staging_berlin_listings SET host_response_rate = REPLACE(host_response_rate, '%', '');

ALTER TABLE staging_berlin_listings
ALTER COLUMN host_response_rate int;
```

4. It was also ensured that all rows from staging listings' tables refer to the corresponding city.

```
UPDATE staging_amsterdam_listings SET city = 'Amsterdam';

UPDATE staging_berlin_listings SET city = 'Berlin';
```

5. To handle data more easily, we combined the staging tables for each city into a unique staging table, one on top of the other.

```
SELECT * INTO staging_calendar
FROM
(   SELECT *
    FROM  staging_amsterdam_calendar
    UNION ALL
    SELECT *
    FROM staging_berlin_calendar
) a

SELECT * INTO  staging_listings
FROM
(   SELECT     *
    FROM  staging_amsterdam_listings
    UNION ALL
    SELECT  *
    FROM staging_berlin_listings
) b

ALTER TABLE staging_listings
ADD CONSTRAINT PK_listings PRIMARY KEY (id);
```

6. To summarize and simplify some data, we added two extra columns:

   One with the categorization regarding the additional cost for extra people, namely whether the properties have extra charge for additional people or not.

```
ALTER TABLE staging_listings
ADD extra_people2 AS
    CASE
      WHEN (staging_listings.extra_people <> 0) THEN 'with charge'
      WHEN (staging_listings.extra_people = 0) THEN 'no charge'
    END
```

   And another one with the property category, which includes a categorization of properties based on the amenities they offer (standard, average, luxury, missing info, other). Specifically:

   - **Standard** property category is defined by having the essential amenities, for example TV, Wifi, heating.
   - **Average** property category includes properties with some extra basic amenities, for example kitchen, washer, breakfast, parking.
   - **Luxury** property category includes properties that have at least one premium feature, for example pool, fireplace, garden, gym.
   - **Missing info** property category includes properties with no specific information of their amenities.
   - **Other** property category includes the properties that do not fit in any of the above categories.

```
ALTER TABLE staging_listings
ADD property_category AS
    CASE
     WHEN ( ((amenities LIKE '%TV%') OR (amenities LIKE '%Wifi%' OR amenities LIKE '%Internet%') OR (amenities LIKE '%Heating%'))
          AND ((amenities NOT LIKE '%kitchen%' AND amenities NOT LIKE '%oven%' ) AND (amenities NOT LIKE '%pool%')
          AND (amenities NOT LIKE '%Hot tub%') AND (amenities NOT LIKE '%Waterfront%')
          AND (amenities NOT LIKE '%parking%') AND (amenities NOT LIKE '%washer%')
          AND (amenities NOT LIKE '%breakfast%') AND (amenities NOT LIKE '%garden%')
          AND (amenities NOT LIKE '%fireplace%') AND (amenities NOT LIKE '%balcony%') AND (amenities NOT LIKE '%Gym%')) ) THEN 'Standard'
     WHEN ( (amenities LIKE '%TV%' OR amenities LIKE '%Wifi%' OR amenities LIKE '%Internet%' OR amenities LIKE '%Heating%'
          OR amenities LIKE '%parking%' OR amenities LIKE '%washer%' OR amenities LIKE '%breakfast%'
          OR amenities LIKE '%kitchen%' OR amenities LIKE '%oven%')
          AND (amenities NOT LIKE '%garden%' AND amenities NOT LIKE '%fireplace%'
          AND amenities NOT LIKE '%balcony%' AND amenities NOT LIKE '%pool%'
          AND amenities NOT LIKE '%Hot tub%' AND amenities NOT LIKE '%Gym%')) THEN 'Average'
     WHEN ( ((amenities LIKE '%TV%') OR (amenities LIKE '%Wifi%' OR amenities LIKE '%Internet%') OR (amenities LIKE '%Heating%'))
          OR ((amenities LIKE '%parking%') OR (amenities LIKE '%washer%')
          OR (amenities LIKE '%breakfast%') OR (amenities LIKE '%kitchen%') OR (amenities LIKE '%oven%'))
          AND ((amenities LIKE '%pool%') OR (amenities LIKE '%Hot tub%') OR (amenities LIKE '%garden%')
          OR (amenities LIKE '%fireplace%') OR (amenities LIKE '%balcony%') OR (amenities LIKE '%Gym%')) ) THEN 'Luxury'
     WHEN amenities ='{}' OR amenities LIKE '%translation%' THEN 'Missing info'
     ELSE 'Other'

    END
```

5.3. Building the data warehouse

Data warehouse is the central repository of information that can be analyzed to make more informed decisions. It pulls together data from many different sources within an organization for reporting and analysis and it can be accessed through business intelligence (BI) tools, SQL clients, and other analytics applications.

5.3.1. Dimensions

Dimension tables are companion tables to the fact table that contain descriptive attributes to be used as query constraining, should be descriptive, wordy, complete and quality assured. They are connected to the fact tables through their primary keys that uniquely identify each element included. They are located at the edges of the schema and can contain hierarchies where is needed (e.g location, date).

For the purpose of our analysis we created 7 dimension tables as described below:

a) PropertiesDim includes all the necessary characteristics to describe a property listed in Airbnb platform:
- Id: represents the primary key of the table which in a later stage will be included in the fact table
- Listing id: is the unique code that each property has in the platform
- Host id: the unique code which identifies the host of the property
- Accommodates: represents the number of guests
- Bathrooms: represents the number of bathrooms
- Bedrooms: represents the number of bedrooms
- Property type: represents the kind of property, whether it is an apartment, a cottage, a hotel, a villa etc.
- Room type: represents the type of the room of the property, if it is a private or shared room, hotel room or an entire home/apartment
- Extra people2: represents whether a property has extra charge for the extra guests
- Property category: is the category that the property belongs (average, standard, luxury)
- Cancelation policy: is the kind of policy that the host of the property follows in case of cancelation (flexible, strict, moderate etc.)
- Avg_price: is an average price added for every listing, calculated by the staging calendar table

Using the below code we created the table and we insert the necessary data for our analysis from staging table.

```
CREATE TABLE PropertiesDim(
      id INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
      listing_id INT NOT NULL,
      host_id INT NULL,
      accommodates INT NULL,
      bathrooms FLOAT NULL,
      bedrooms INT NULL,
      property_type  NVARCHAR (200) NULL,
      room_type NVARCHAR (200) NULL,
      extra_people2 NVARCHAR (200) NULL,
      property_category NVARCHAR (200) NULL,
      cancellation_policy NVARCHAR (200) NULL);


INSERT INTO PropertiesDim(listing_id,host_id,accommodates,bathrooms,bedrooms,property_type,room_type,extra_people2,property_category,cancellation_policy) (
SELECT s.id, s.host_id,s.accommodates,s.bathrooms,s.bedrooms,s.property_type,s.room_type,s.extra_people2,s.property_category,s.cancellation_policy
FROM staging_listings s);

ALTER TABLE PropertiesDim
ADD avg_price float;

UPDATE PropertiesDim  SET avg_price = (SELECT AVG(price) FROM staging_calendar
WHERE PropertiesDim.listing_id = staging_calendar.listing_id GROUP BY listing_id)
```

b) HostDim includes information about the host of each property:
- ID: represents the primary key of the table that will be used as a connection with the fact table
- Host id: is the unique code which identifies the host of the property (used by the Airbnb platform)
- Host name: is the first name of the host

- Host is superhost: represents with T/F/N if the host has the title of superhost or not (N : if we do not have specific info)

Using the below code we created the table, imported the data and removed the duplicate rows in order to have a clear table that contains only the unique values. Finally, we updated the table and replaced the nulls with NAs.

```
CREATE TABLE HostsDim (
    [host_id] INT NOT NULL,
    [host_name] NVARCHAR(MAX) DEFAULT 'N/A' NULL,
    [host_is_superhost] NVARCHAR(1) NULL,
);

INSERT INTO HostsDim ([host_id],[host_name],[host_is_superhost]) SELECT staging_listings.host_id,
                                                staging_listings.host_name,
                                                staging_listings.host_is_superhost
                                                FROM staging_listings;

WITH cte AS (
    SELECT
        [host_id], [host_name],[host_is_superhost],
        ROW_NUMBER() OVER ( PARTITION BY [host_id], [host_name],[host_is_superhost]
        ORDER BY [host_id], [host_name],[host_is_superhost] ) row_num FROM HostsDim)
DELETE FROM cte
WHERE row_num > 1;


ALTER TABLE HostsDim
ADD ID INT IDENTITY (1,1) NOT NULL;

ALTER TABLE HostsDim
ADD CONSTRAINT PK_hosts PRIMARY KEY (ID);


UPDATE HostsDim
SET host_is_superhost='NA'
WHERE host_is_superhost is null;

UPDATE HostsDim
SET host_name='NA'
WHERE host_name is null;
```

c) CityDim contains:

- city id: represents the primary key of the table and it is used also to connect LocationDim table with CityDim
- city name: represents the name of the city where the property is located

Using the following code we created the table and imported the necessary data.

```
CREATE TABLE CityDim(
             city_id INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
             city_name VARCHAR (200));

INSERT INTO CityDim (city_name) SELECT DISTINCT city FROM staging_listings;
```

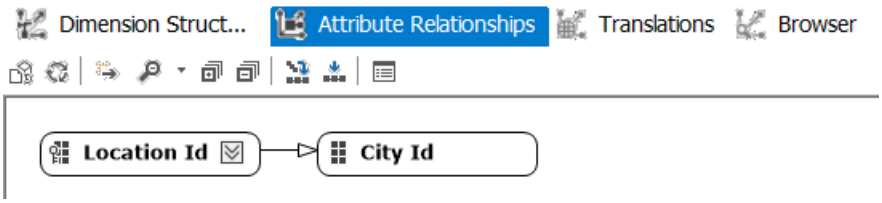
d) LocationDim includes location data and it is connected with the City dimension:

- Location id: represents the primary key of the table which will be used at a later stage to connect this table with the fact tables
- Longitude
- Latitude
- City id: is the foreign key which connects the location dimension table with the city dimension table

Below is the code we used for the table's creation:

```
CREATE TABLE LocationDim (
                latitude FLOAT NOT NULL,
                longitude FLOAT NOT NULL,
                city_id INT NOT NULL,
                FOREIGN KEY (city_id) REFERENCES CityDim (city_id ));

INSERT INTO LocationDim (latitude,longitude, city_id) (SELECT s.latitude,s.longitude, c.city_id
                                                        FROM staging_listings s, CityDim  c
                                                        WHERE s.city = c.city_name);


--delete duplicate rows
WITH cte AS (
    SELECT
        latitude, longitude,city_id,
        ROW_NUMBER() OVER ( PARTITION BY latitude, longitude, city_id  ORDER BY latitude, longitude, city_id ) row_num FROM LocationDim)
DELETE FROM cte
WHERE row_num > 1;


ALTER TABLE LocationDim
ADD location_id INT IDENTITY (1,1) NOT NULL;

ALTER TABLE LocationDim
ADD CONSTRAINT PK_Location PRIMARY KEY (location_id);
```

Important to note here is that we created a hierarchy between LocationDim table and CityDim table from detailed to less-detailed level as shown below:



e) YearDim includes the years for those there is available data for rental fees and availability.

- Year id: is the primary key of the table which in the next step will be used to connect the YearDim table with MonthDim table
- Year: includes the respective year

The following code was used for the table's creation. Data was sourced from staging calendar table.

```
CREATE TABLE YearDim (
          year_id INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
         [year] INT NULL);

INSERT INTO YearDim ([year]) SELECT DISTINCT year([date]) FROM staging_calendar;
```

f) MonthDim includes the months for which there is available data for rental fees and availability.

- Month id: is the primary key, which in the next step will be used to connect the Datedim with MonthDim table
- Month_number: includes the month number
- Month: includes the month name
- Year id: is the foreign key which connects the two tables

Using the below code we created the MonthDim table, inserted the data and connected it with the YearDim table.

```
CREATE TABLE MonthDim (
          month_id INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
          month_number INT  NULL,
          [month] NVARCHAR (50)  NOT NULL,
          year_id INT NOT NULL,
          FOREIGN KEY (year_id) REFERENCES YearDim (year_id));

INSERT INTO MonthDim (month_number,[month],year_id) SELECT DISTINCT month([date]),DATENAME(month, [date]),year_id
                                                    FROM staging_calendar,YearDim  WHERE year(staging_calendar.[date])=YearDim.[year];
```

g) DateDim includes all the available dates for which there is data for rental fees and availability.

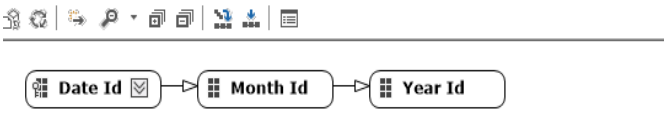
- Date id: is the primary key which connects the date dimension table with the fact table
- Date: includes all the available dates
- Month id: id the foreign key which connects the two tables

Using the below code we created the table, imported the available data and connected it with MonthDim table.

```
CREATE TABLE DateDim (
      date_id INT IDENTITY (1,1) NOT NULL PRIMARY KEY,
      [date] DATE NOT NULL,
      month_id INT NOT NULL
      FOREIGN KEY (month_id) REFERENCES MonthDim (month_id)
      );
```

```
INSERT INTO DateDim ([date],month_id) SELECT DISTINCT staging_calendar.[date],month_id
                                      FROM staging_calendar, MonthDim ,YearDim
                                      WHERE month(staging_calendar.[date])=MonthDim.month_number
                                            AND year(staging_calendar.[date])=YearDim.[year]
                                            AND MonthDim.year_id=YearDim.year_id;
```

For the three latter tables we created a hierarchy from the detailed to the less detailed level creating a snowflake in our schema:



5.3.2 Fact Tables

A fact table consists of facts/events and quantitative information that analysts want to aggregate. It is the central table in a star/snowflake schema of a data warehouse and it is surrounded by dimension tables. A fact table typically has two kinds of columns, those that include the metrics, measurements and those that are foreign keys and connect the table with the dimension tables.

For our project we built three fact tables, thus creating a fact constellation because fact tables have common dimensions. The reason for which we chose to design three facts is that we want to analyze the data from three different perspectives: host perspective, properties (listings) perspective and pricing perspective.

a) **FactListings** contains all the necessary information related with properties listed in Airbnb platform. This fact table is connected with foreign keys with the dimension tables HostDim, LocationDim, ProperiesDim. Specifically, the included columns are:

Foreign keys: *listing_key*: connects the fact table with the PropertiesDim dimension table.
location_id: connects the fact table with LocationDim dimension table.
host_id: connects the fact table with HostsDim dimension table

Metrics:

availabity_365

cleaning_fee

review_scores_rating

review_scores_cleanliness

review_scores_communication

review_scores_value

reviews_per_month

review_scores_accuracy

review_scores_checkin

review_scores_location,

number_of_reviews

Using the following code, we created the fact table and imported all the above data:

```
CREATE TABLE FactListings (
    listing_key INT NOT NULL PRIMARY KEY,
    [host_id] INT NOT NULL,
    location_id INT NOT NULL,
    availabity_365 INT NULL,
    cleaning_fee MONEY NULL,
    review_scores_rating INT  NULL,
    review_scores_accuracy INT  NULL,
    review_scores_cleanliness INT NULL,
    review_scores_checkin INT NULL,
    review_scores_communication INT NULL,
    review_scores_location INT NULL,
    review_scores_value INT NULL,
    number_of_reviews INT NULL,
    reviews_per_month FLOAT NULL,
    FOREIGN KEY (listing_key) REFERENCES PropertiesDim(id),
    FOREIGN KEY (host_id) REFERENCES HostsDim (ID),
    FOREIGN KEY (location_id) REFERENCES LocationDim (location_id)
     );

INSERT INTO FactListings (listing_key, [host_id], location_id, availabity_365,
                          cleaning_fee, review_scores_rating, review_scores_accuracy, review_scores_cleanliness, review_scores_checkin,
                          review_scores_communication, review_scores_location, review_scores_value,
                          number_of_reviews, reviews_per_month)
                          (SELECT P.id, h.ID,
                                  L.location_id,
                                 s.availability_365, s.cleaning_fee, s.review_scores_rating,
                                  s.review_scores_accuracy, s.review_scores_cleanliness, s.review_scores_checkin, s.review_scores_communication,
                                  s.review_scores_location, s.review_scores_value,
                                  s.number_of_reviews, s.reviews_per_month
                           FROM staging_listings  s , HostsDim  h, PropertiesDim  P,LocationDim  L
                           WHERE P.listing_id=s.id AND L.latitude=s.latitude AND L.longitude=s.longitude AND s.host_id = h.host_id );
```

b) **FactHosts** contains all the data related with the hosts of the properties. It is connected through foreign key with the HostsDim dimension table. Specifically included columns are:
Foreign key: *host_id:* foreign key that is primary key in HostsDim table.

Metrics:

host_response_rate	host_acceptance_rate
total_listings	total_entire_homes
total_private_rooms	total_shared_rooms

Below is the used code for table's creation and data entry.

```
CREATE TABLE FactHosts (
    [host_id] INT NOT NULL,
    [host_response_rate] INT NULL,
    [host_acceptance_rate] INT NULL,
    [total_listings] INT NULL,
    [total_entire_homes] INT NULL,
    [total_private_rooms] INT NULL,
    [total_shared_rooms] INT NULL,
    FOREIGN KEY (host_id) REFERENCES HostsDim (ID),
    );

INSERT INTO  FactHosts ( host_id, host_response_rate, host_acceptance_rate, total_listings, total_entire_homes, total_private_rooms, total_shared_rooms)
             (SELECT DISTINCT h.ID, s.host_response_rate, s.host_acceptance_rate, s.calculated_host_listings_count,
                              s.calculated_host_listings_count_entire_homes, s.calculated_host_listings_count_private_rooms,
                              s.calculated_host_listings_count_shared_rooms
              FROM staging_listings  s, HostsDim  h
              WHERE s.host_id = h.host_id);
```

c) **FactCalendar** includes available dates, daily rental price, minimum and maximum nights and availability per day. It is connected with PropertiesDim, LocationDim and DateDim dimension tables. Specifically, it contains the following columns:

Foreign keys: *listing_id*: connects the table with PropertiesDim

location_id: connects the table with LocationDim

date_id: connects the table with DateDim

Metrics:

price

minimum_nights

maximum-nights

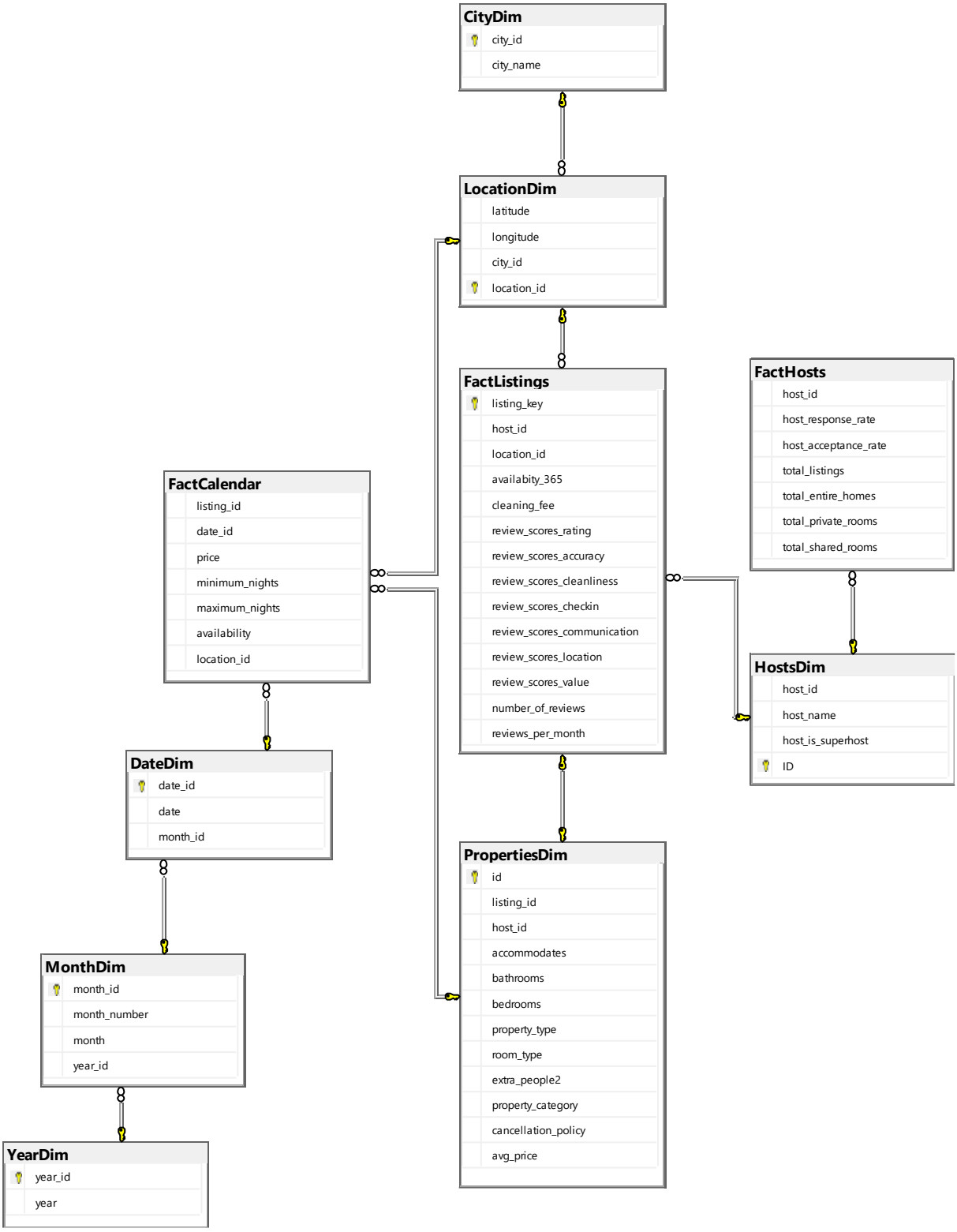
availability

For the creation of the table initially we created a view table which contains the primary key of properties dimension table (id), the primary key of location dimension table (location_id) and the listing_id which is the unique code that the platform uses for each listed property. That step was crucial in order to achieve a connection with the staging calendar table from where we imported the data.

```
CREATE VIEW VR (property_key,location_id,listing_id)
AS
SELECT P.id,L.location_id,s.id
FROM PropertiesDim  P,LocationDim  L, staging_listings  s
WHERE P.listing_id=s.id AND L.latitude=s.latitude AND L.longitude=s.longitude;
GO
```

```
CREATE TABLE FactCalendar (
    listing_id INT NOT NULL,
    date_id INT NOT NULL,
    price MONEY NULL,
    minimum_nights INT NULL,
    maximum_nights INT NULL,
    [availability] NVARCHAR (1) NULL,
    location_id INT NOT NULL,
    FOREIGN KEY (listing_id) REFERENCES PropertiesDim (ID),
    FOREIGN KEY (date_id) REFERENCES DateDim (date_id),
    FOREIGN KEY (location_id) REFERENCES LocationDim (location_id)
    );
```

```
INSERT INTO  FactCalendar (listing_id, date_id, location_id, price, minimum_nights, maximum_nights, [availability])
             (SELECT P.id, d.date_id, L.location_id, sc.price, sc.minimum_nights, sc.maximum_nights, sc.available
              FROM staging_calendar  sc, DateDim  d, PropertiesDim  P, LocationDim  L , VR
              WHERE sc.[date] = d.[date] AND P.id=VR.property_key AND P.listing_id = sc.listing_id AND VR.listing_id=sc.listing_id AND VR.location_id=L.location_id);
```



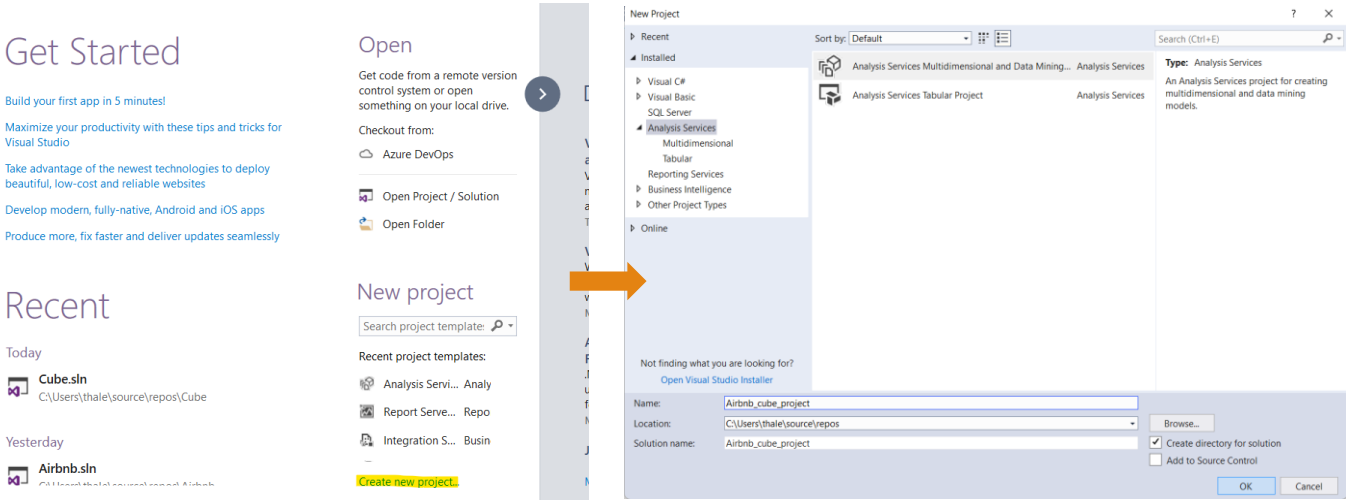
6. Cube

6.1. Cube Creation using Visual Studio

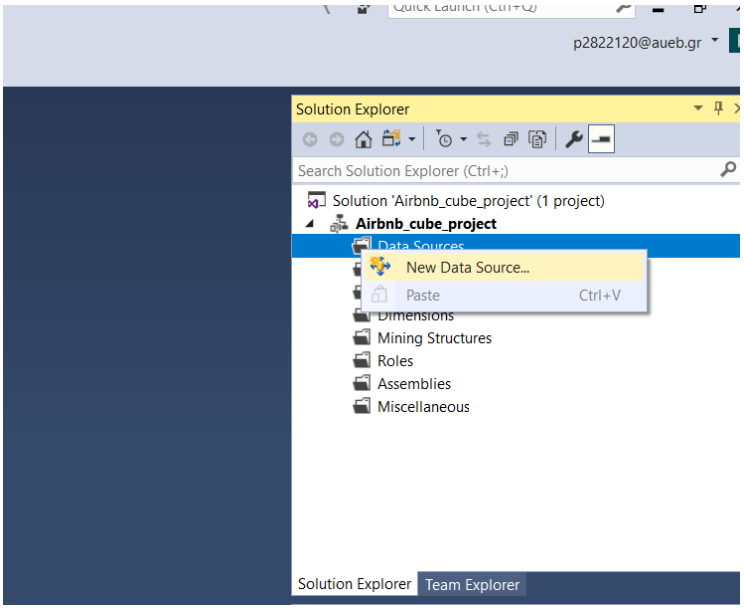
An online analytical processing (OLAP) cube is a core component of data warehousing implementations which provides fast and flexible multidimensional analysis. Cubes are a feature in Service Manager that use an existing data warehouse infrastructure to provide self-service business intelligence capabilities to end users. Cubes can deploy aggregations and calculations in large volumes of data while simultaneously can provide to users access to any data points and measures. With OLAP cube creation, the data can be rolled up, drilled down, sliced and diced as needed to answer the widest variety of questions that are relevant to a user's area of interest. These cubes are stored in SQL Server Analysis Services

(SSAS) and one can use them to analyze the data from multiple perspectives. For this project, we created a multidimensional analysis. Steps are shown below in detail:

Firstly, we open Visual Studio (SSDT) and create a new project and we give a name to our project :

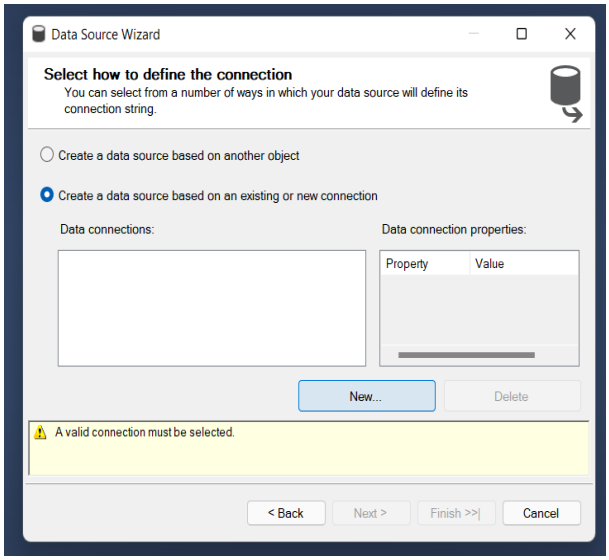


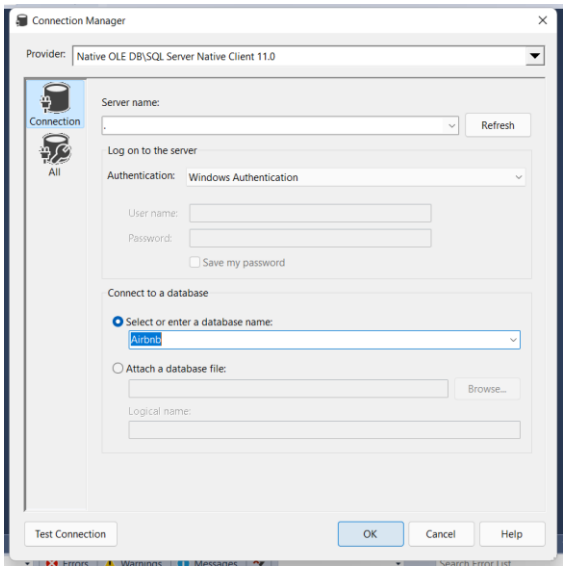
Once we created the project then we should import the datasource from SSMS:



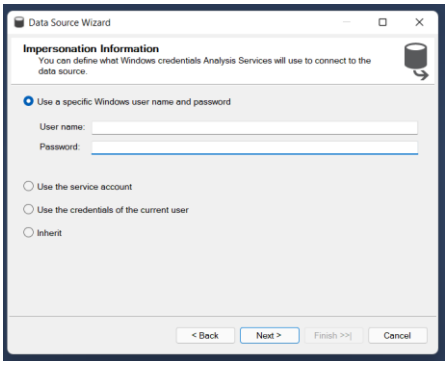
A wizard window pops up:

We select the database that we created in previous steps:

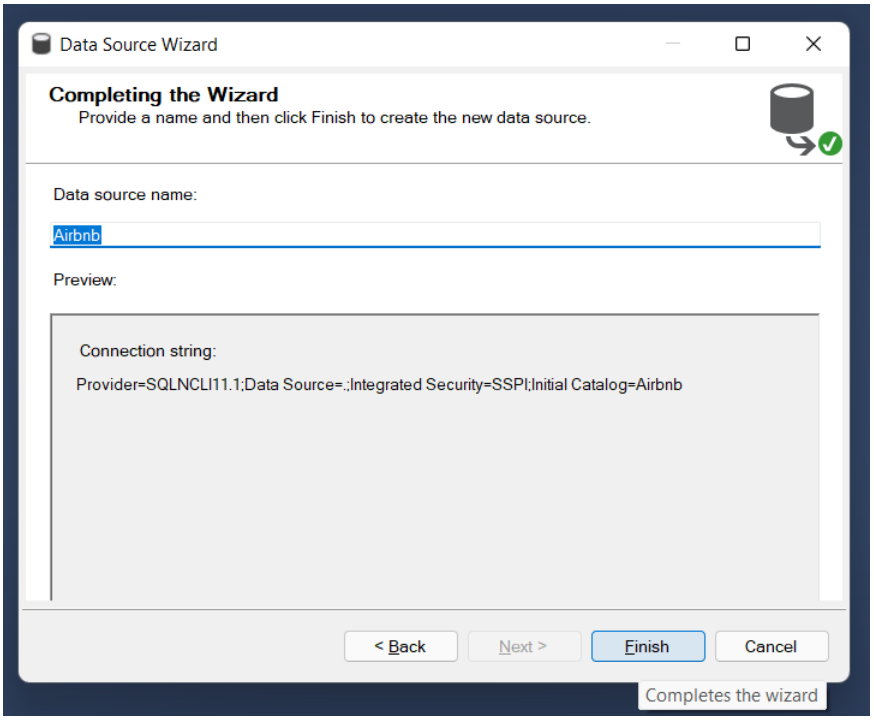




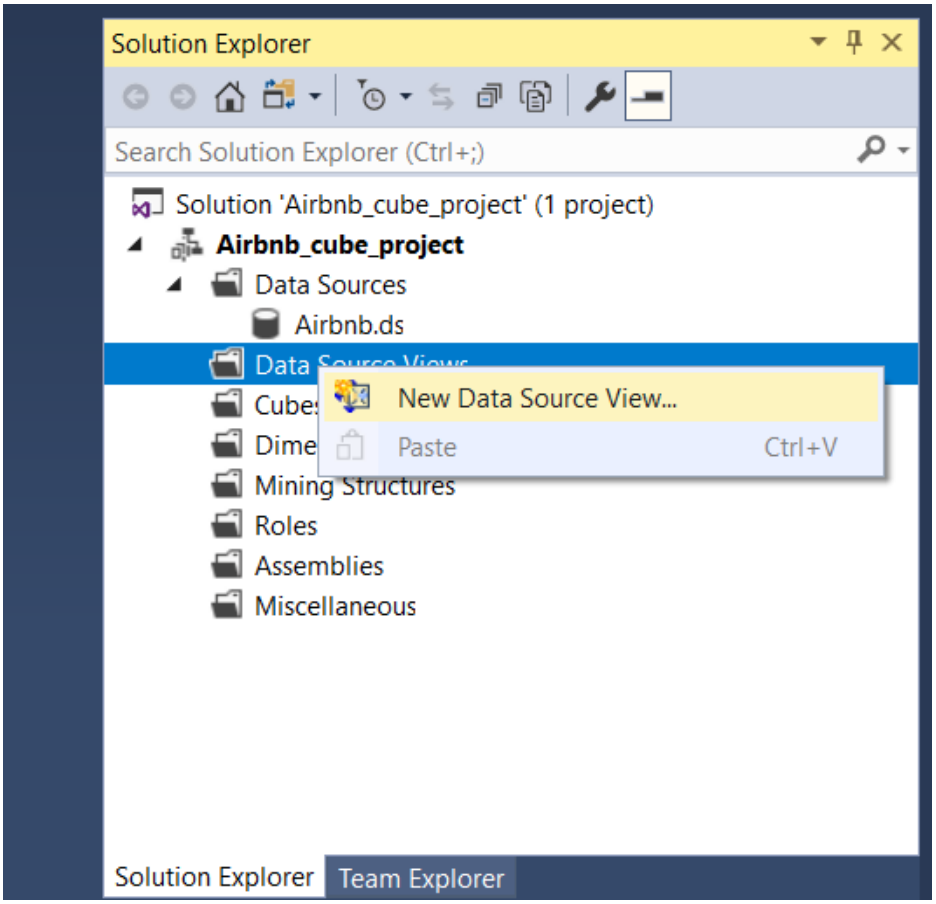
We should populate the PC credentials, an important step for the correct proceess of the cube:



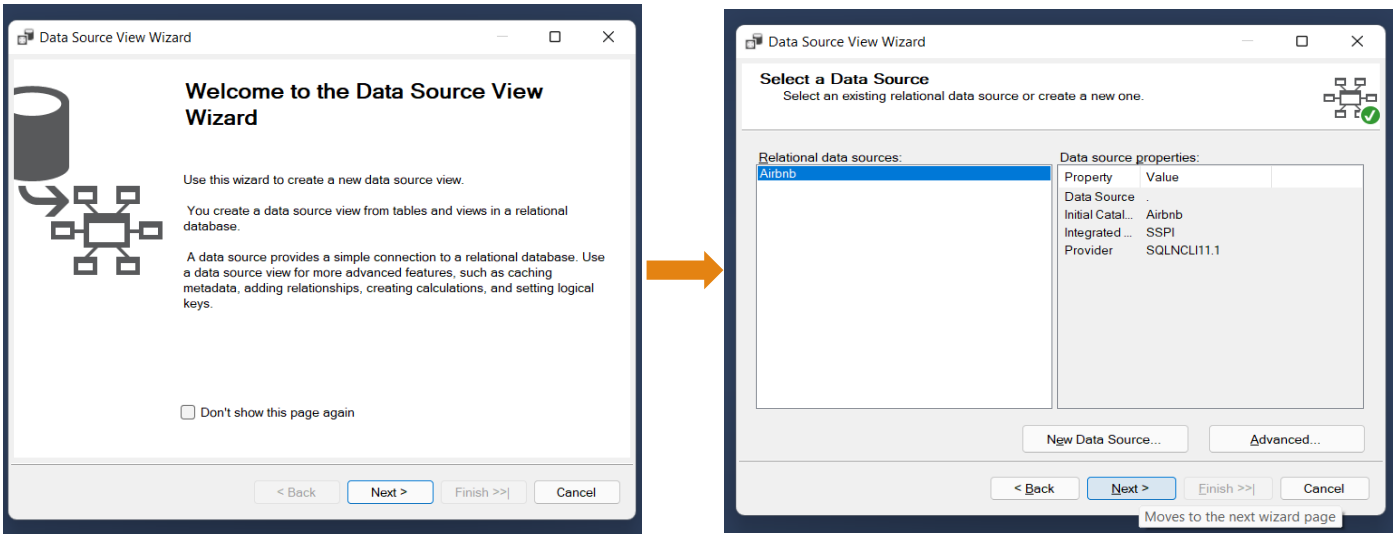
The data source upload procedure has been completed:



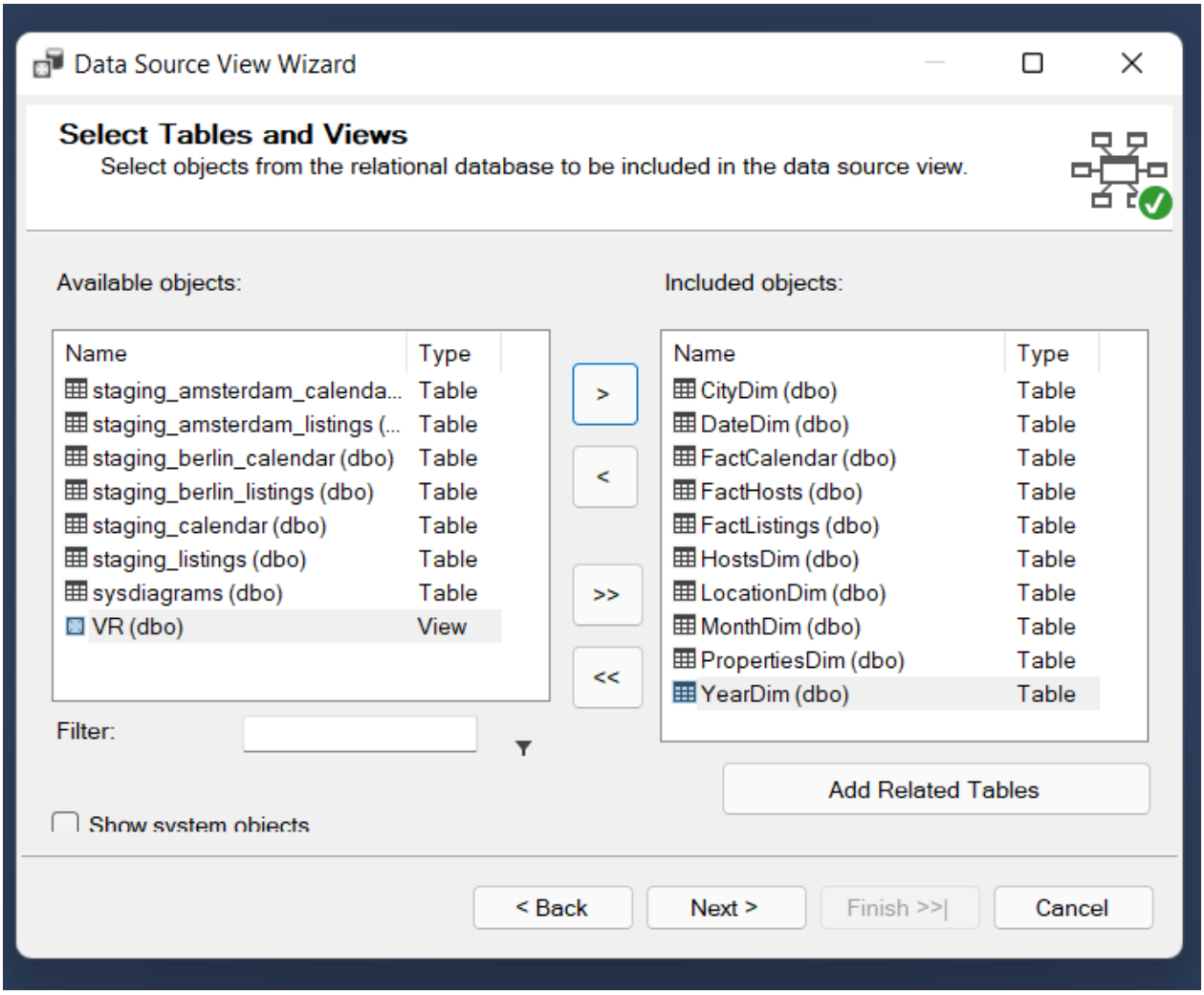
Next step is to import the tables that will be used for relational schema and cube creation:



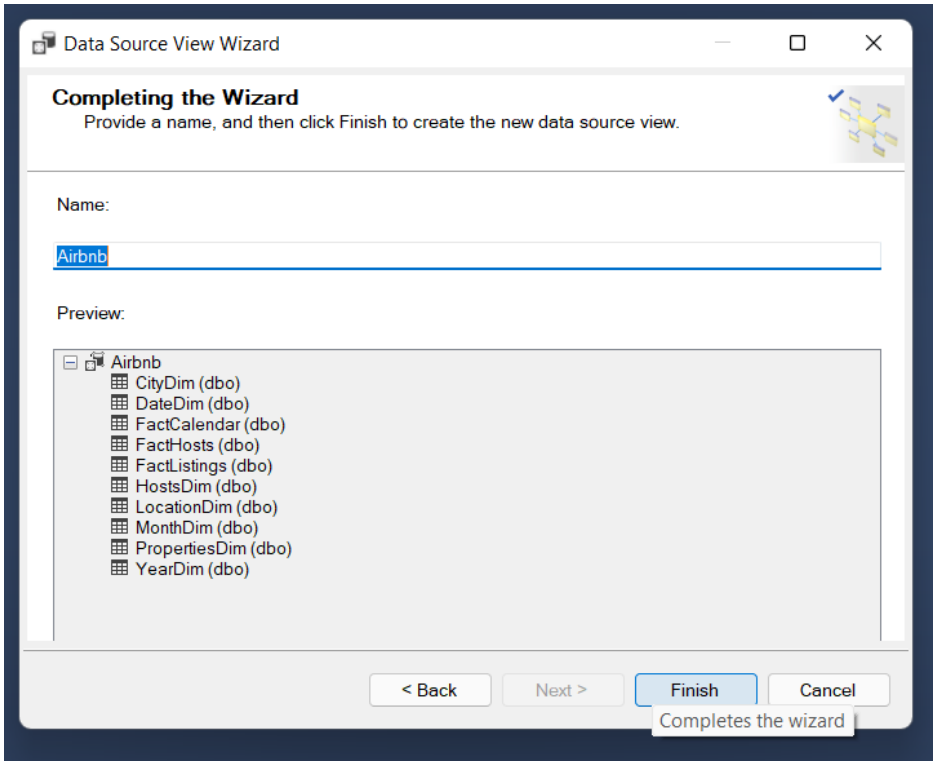
A wizard window pops up again:



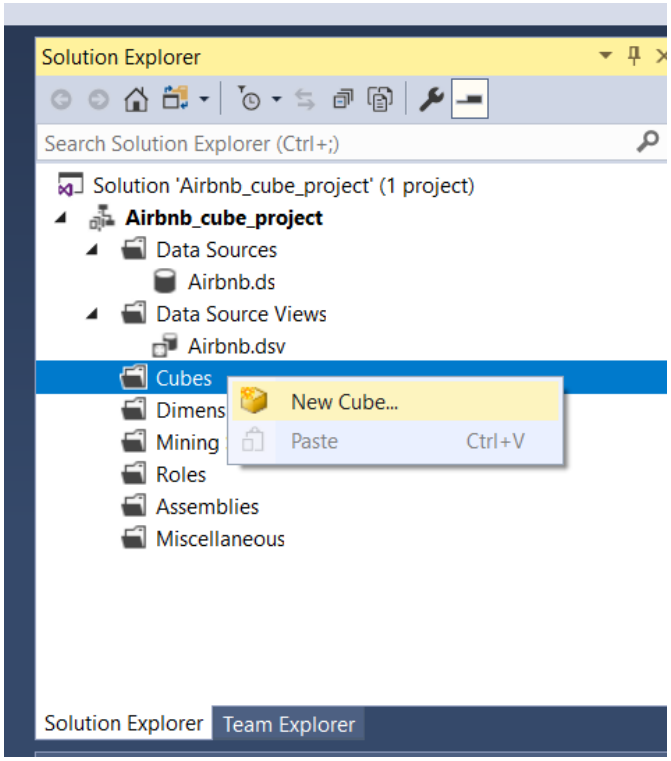
We select all the fact and dimension tables we created in SSMS :



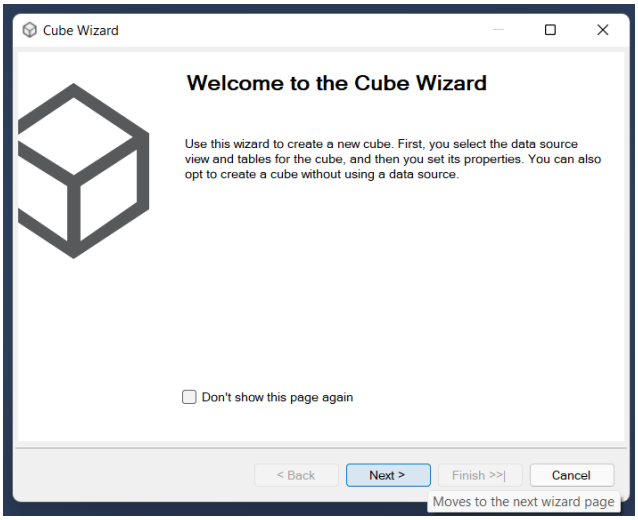
Tables' import has been also completed:

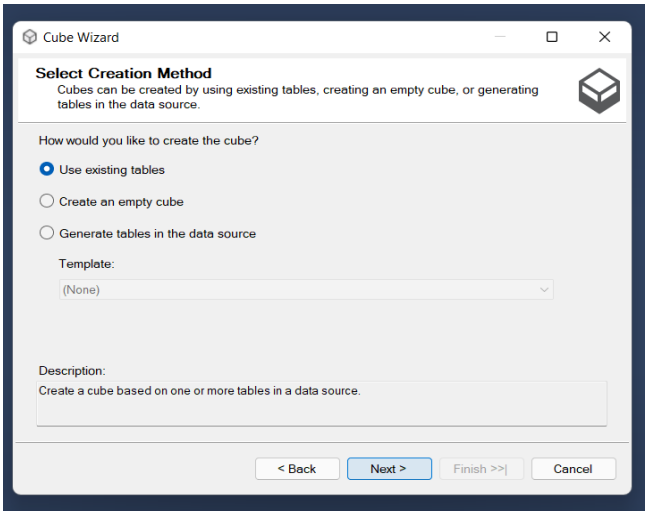


Now since we have import all the necessary data we can start to build the cube:

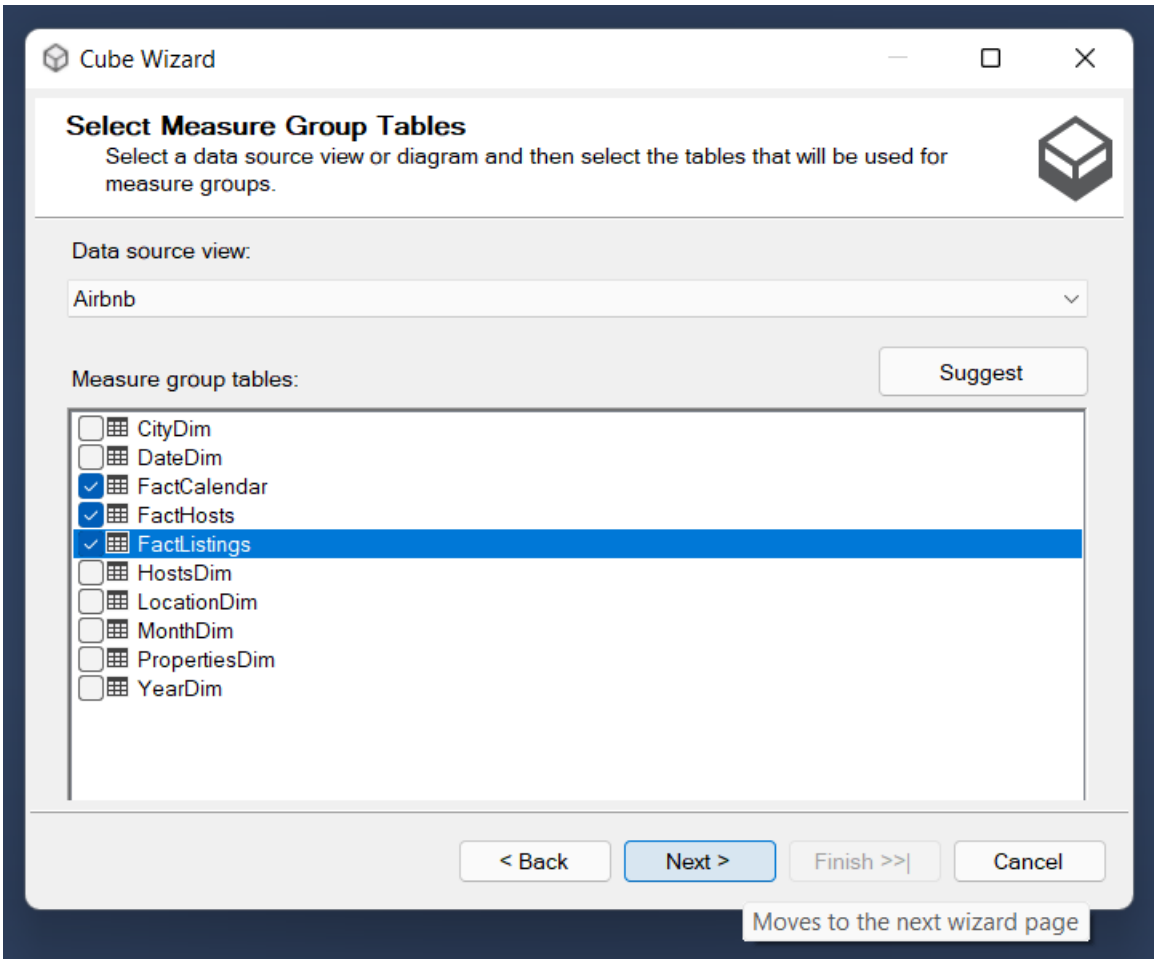


Again a wizard window pops up:

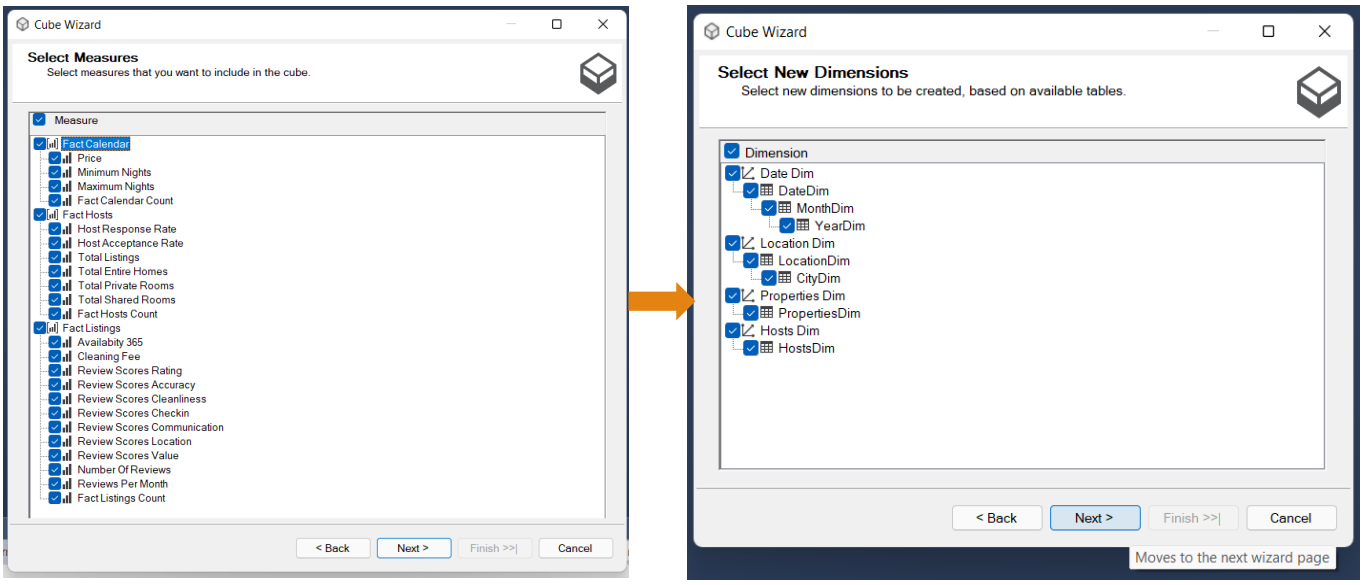




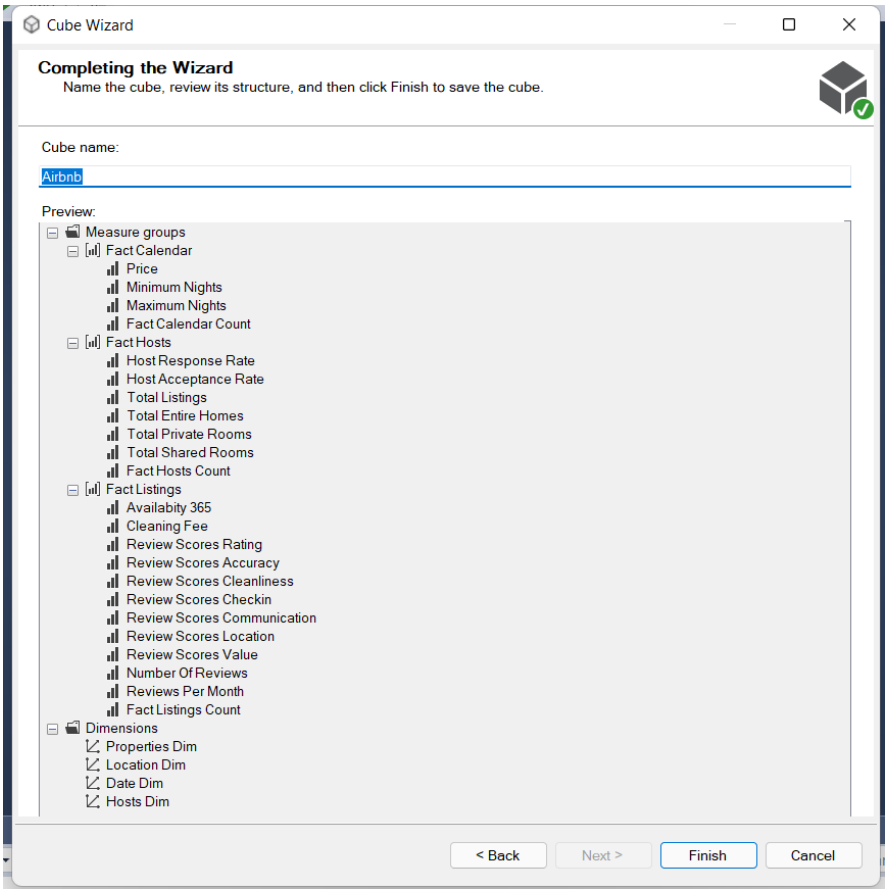
At this stage, we need to select the fact tables of our schema :



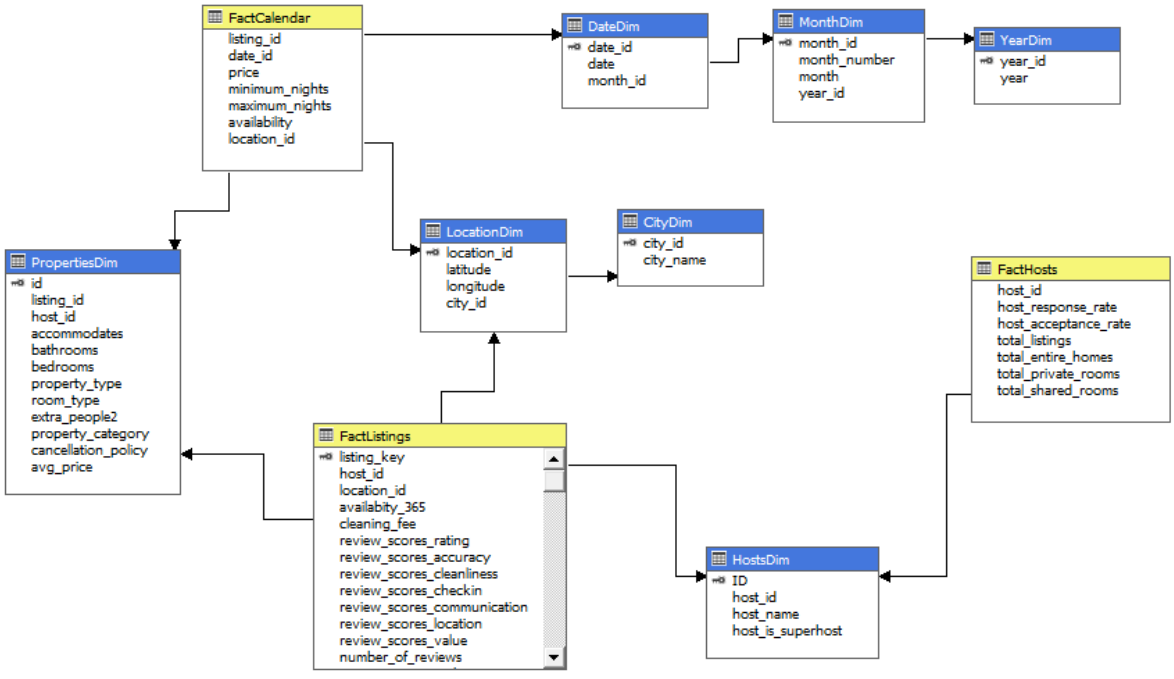
After selecting the fact tables, the SSAS recognizes the metrics and the dimensions:



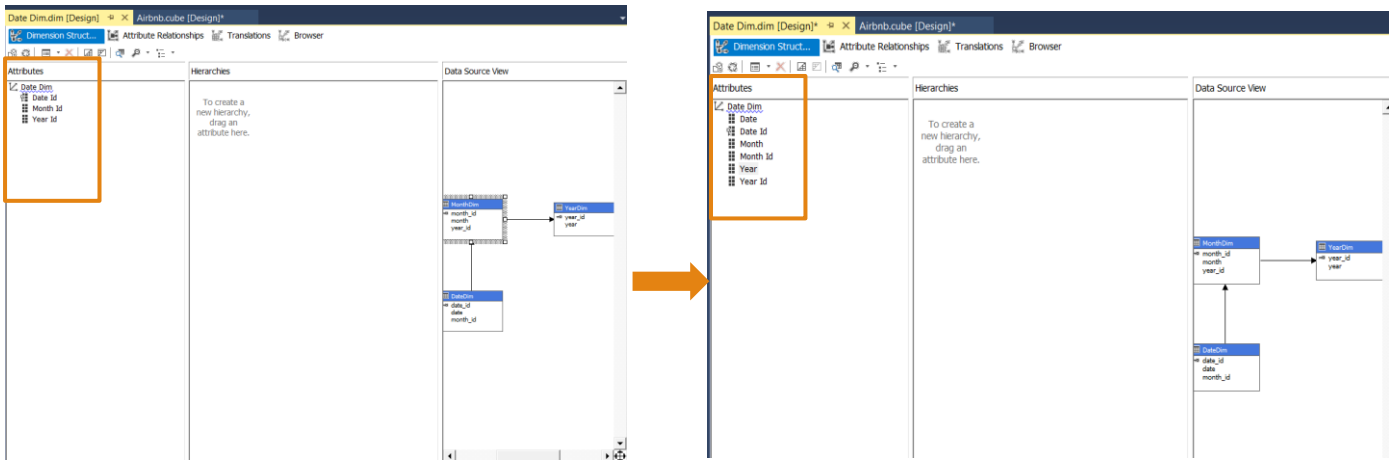
The cube creation process has been completed:

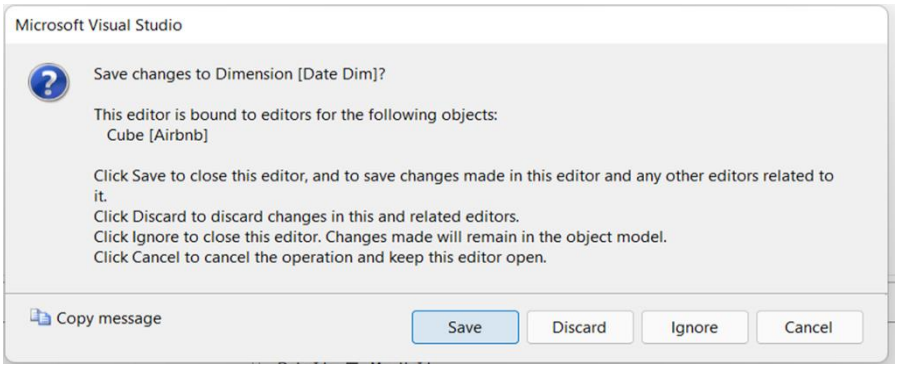


The **snowflake schema** with fact constellation modeling is created:

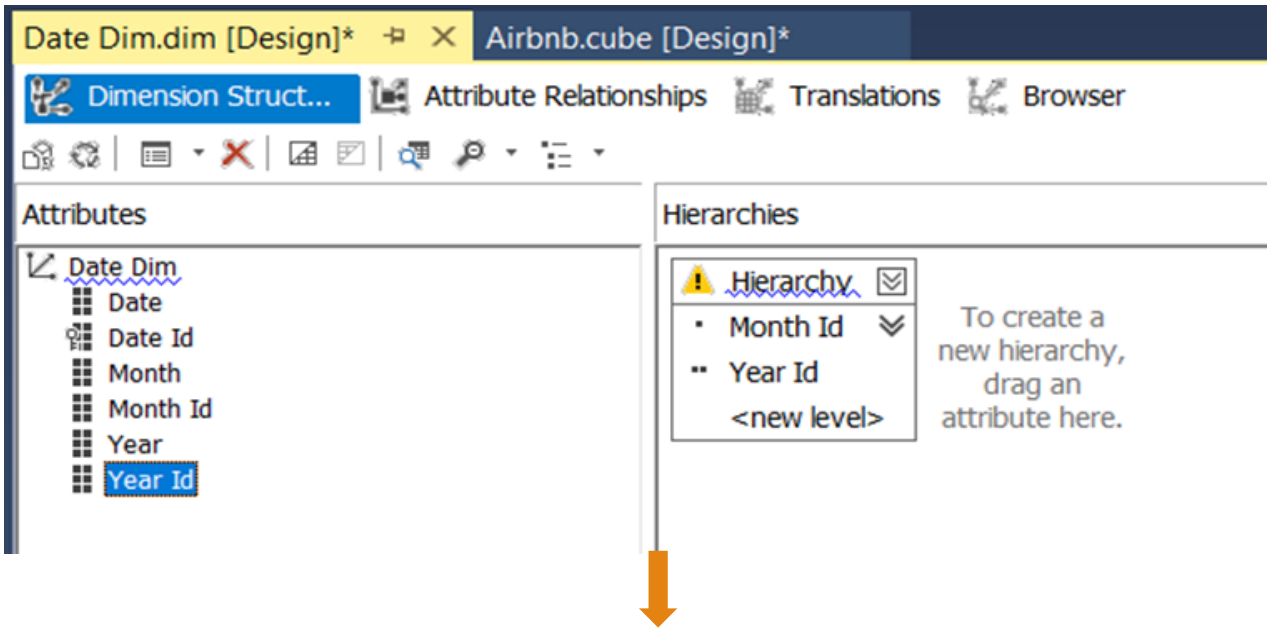


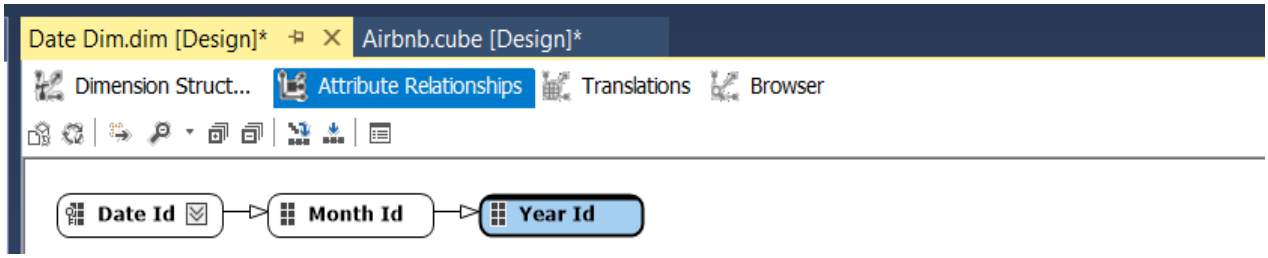
Next step is to edit all the attributes in each dimension table because only IDs were included, as per below:



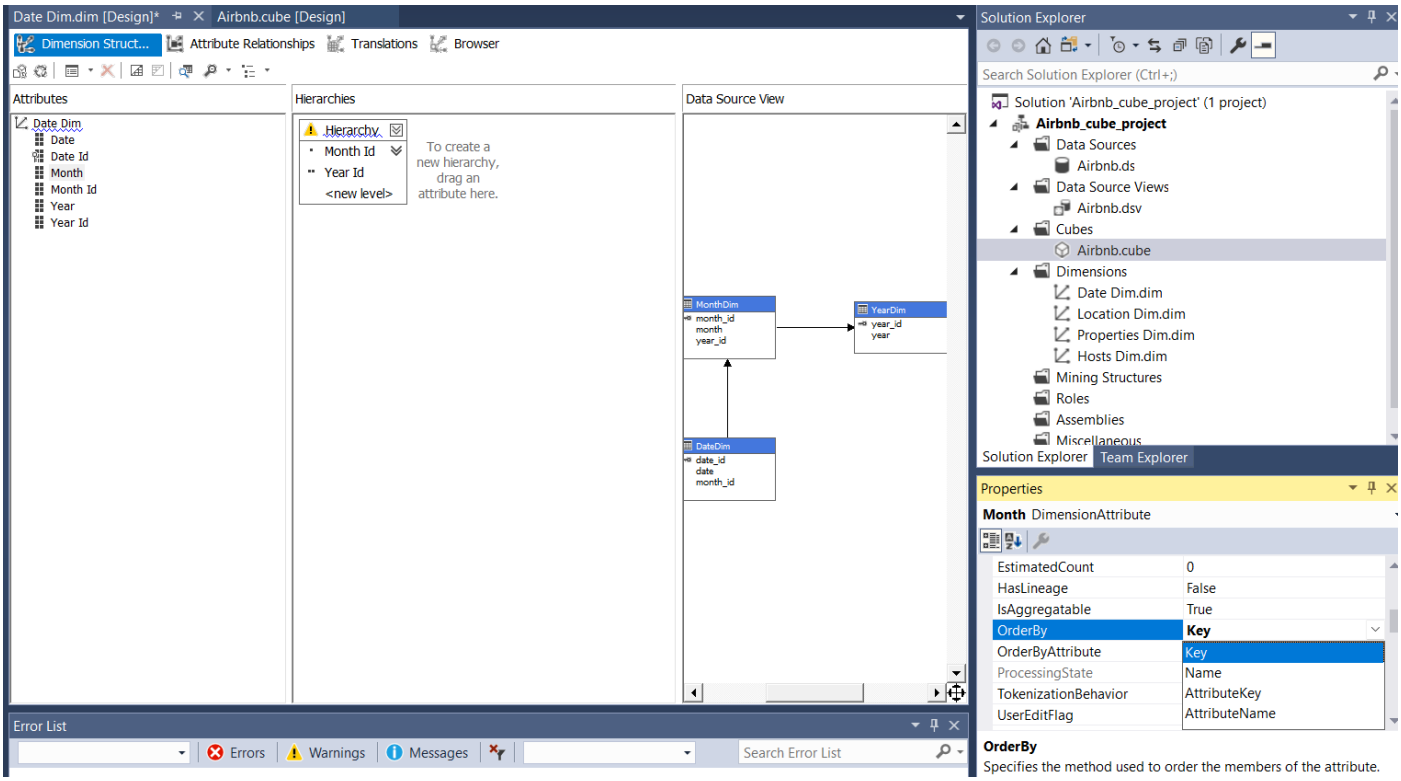


We follow the same process for all the dimension tables, and we create hierarchies where is needed (in this project we created hierarchies for Date and Location tables):

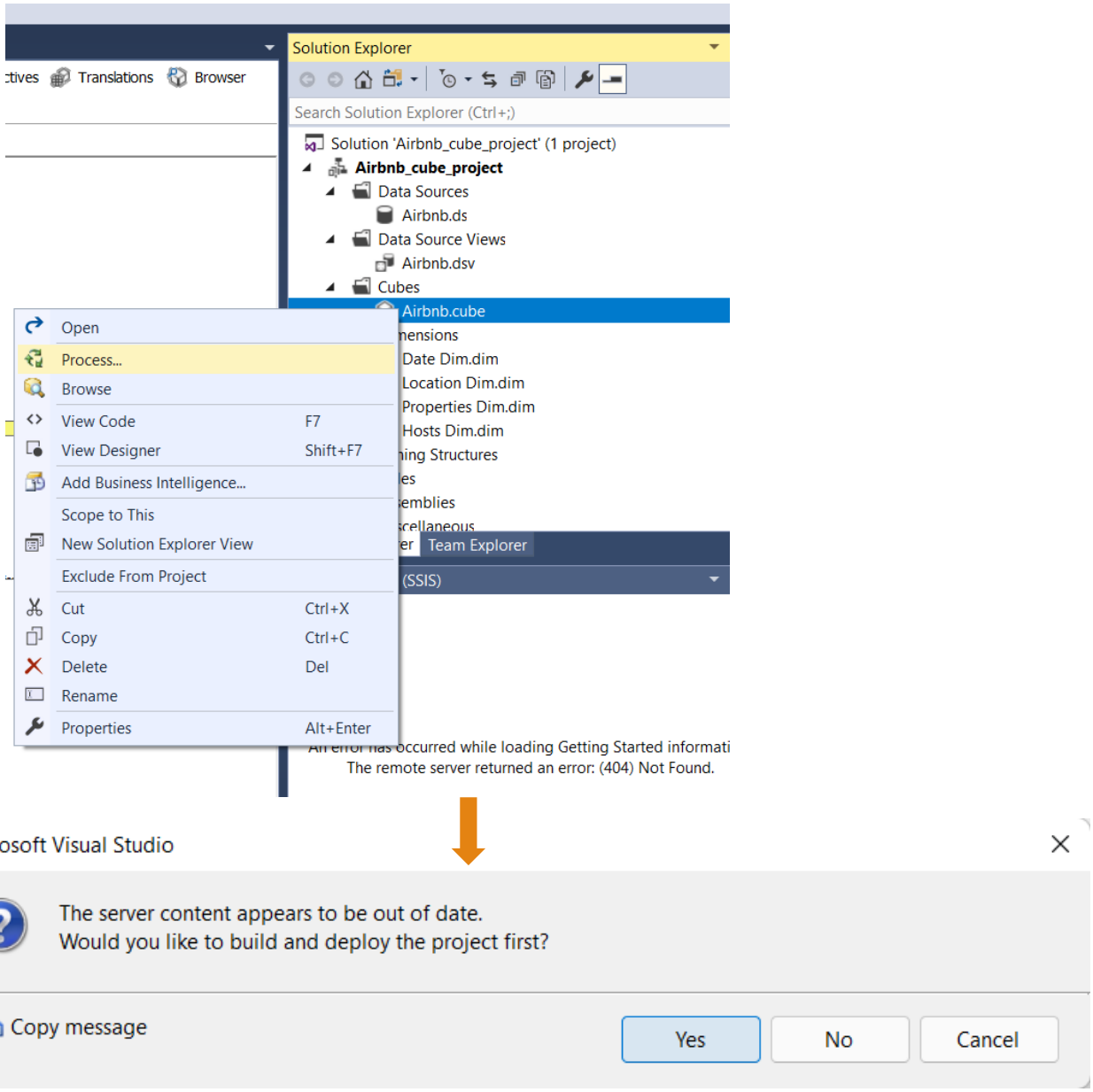




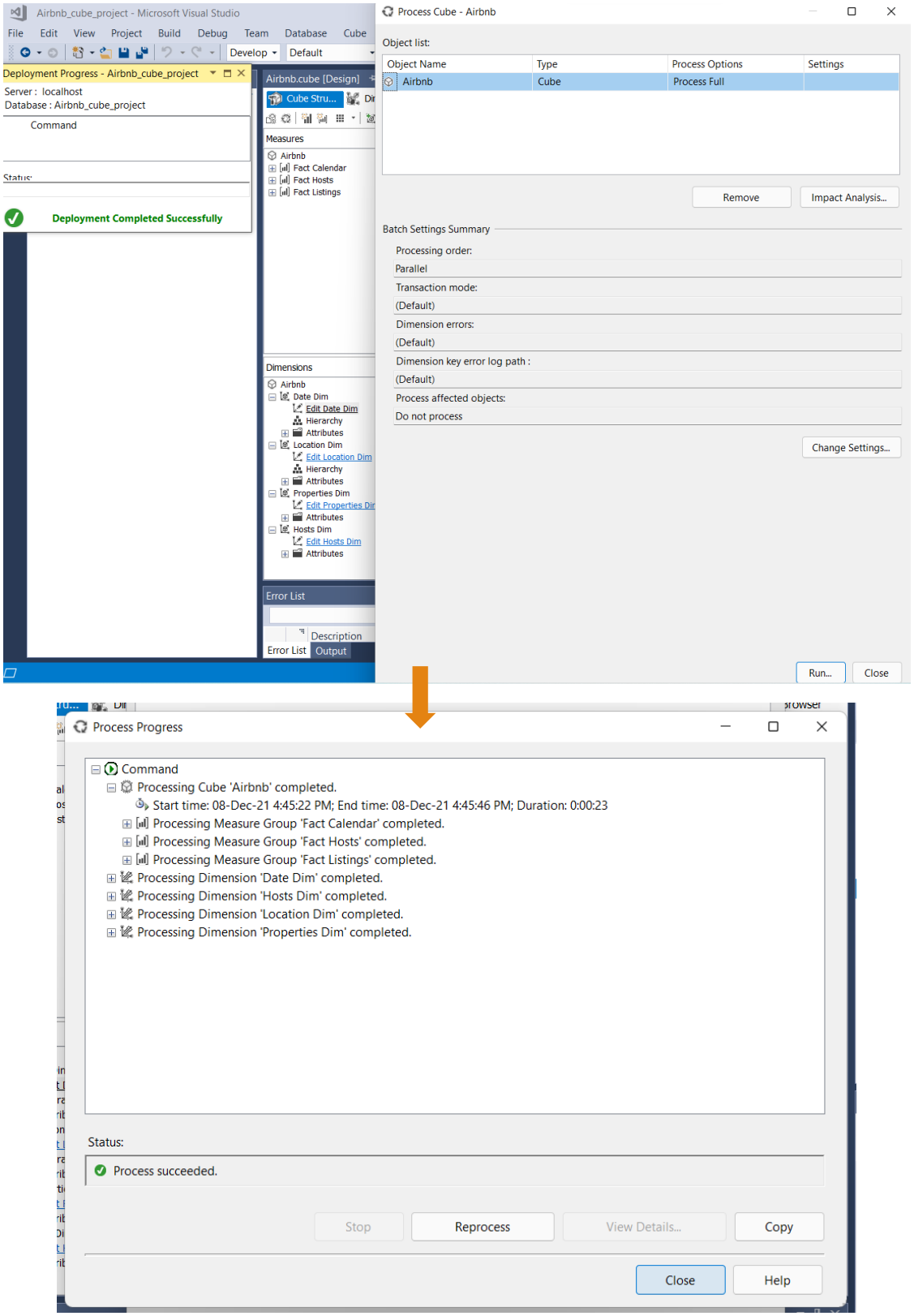
Important to note here that for Date dimension we proceed with the below adjustment in the field 'Order By' in properties, so that Power BI can recognize the data as date type:



After all the above edits cube is ready to be processed:

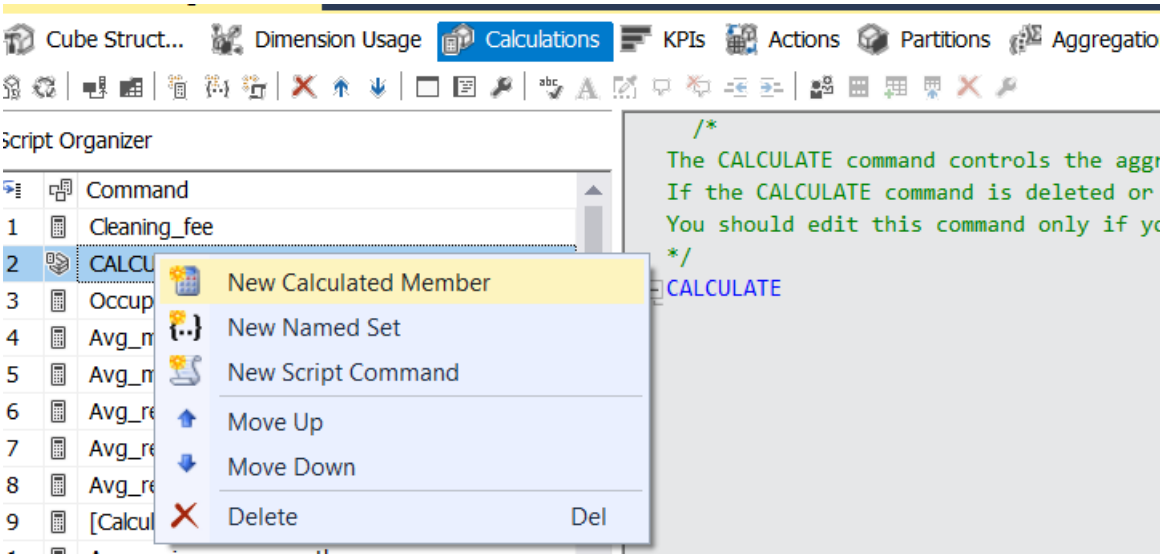

Translations
Browser
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Airbnb_cube_project' (1 project)
Airbnb_cube_project
Data Sources
Airbnb.ds
Data Source Views
Airbnb.dsv
Cubes
Airbnb.cube
Date Dim.dim
Location Dim.dim
Properties Dim.dim
Hosts Dim.dim
Open
Process...
Browse
View Code F7
View Designer Shift+F7
Add Business Intelligence...
Scope to This
New Solution Explorer View
Exclude From Project
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Rename
Properties Alt+Enter
Team Explorer
(SSIS)
The remote server returned an error: (404) Not Found.

Microsoft Visual Studio
The server content appears to be out of date.
Would you like to build and deploy the project first?
Copy message
Yes
No
Cancel



6.2. Calculations

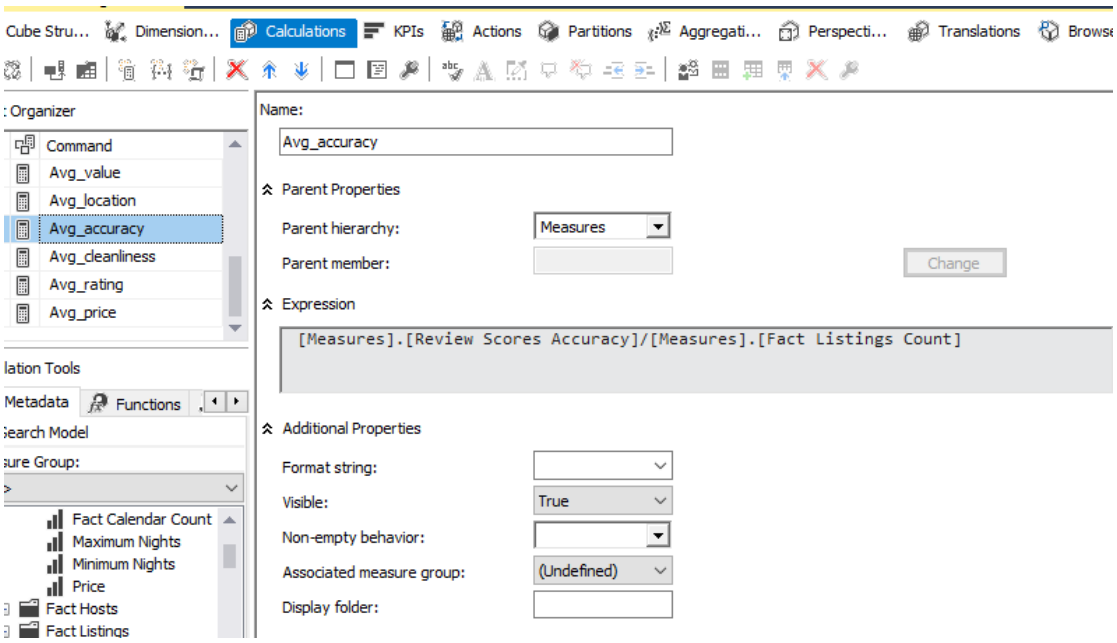
In SSDT Visual Studio a calculation tool is also provided for the end user. Using this tool, we created some extra measures that helped us provide interesting results for our analysis.



1. **Average accuracy**

By creating this measure, we can calculate the average of reviews scores accuracy.

Avg_accuracy = Reviews scores accuracy / Number of total listings



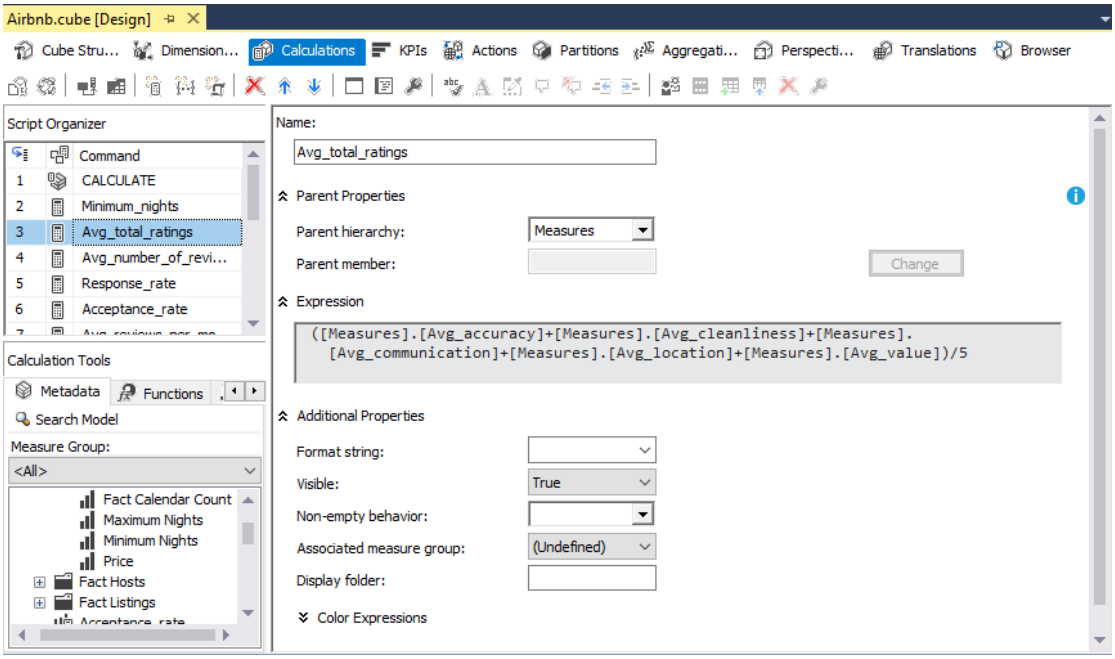
Following the above process, we created also the below measures:

Avg_location, Avg_communication, Avg_value, Avg_cleanliness, Avg_number_of_reviews, Avg_reviews_per_month

2. **Average total ratings**

This measure calculates the total average ratings for each property, taking into account the average rates of accuracy, cleanliness, communication, location and value that we created in a previous step.

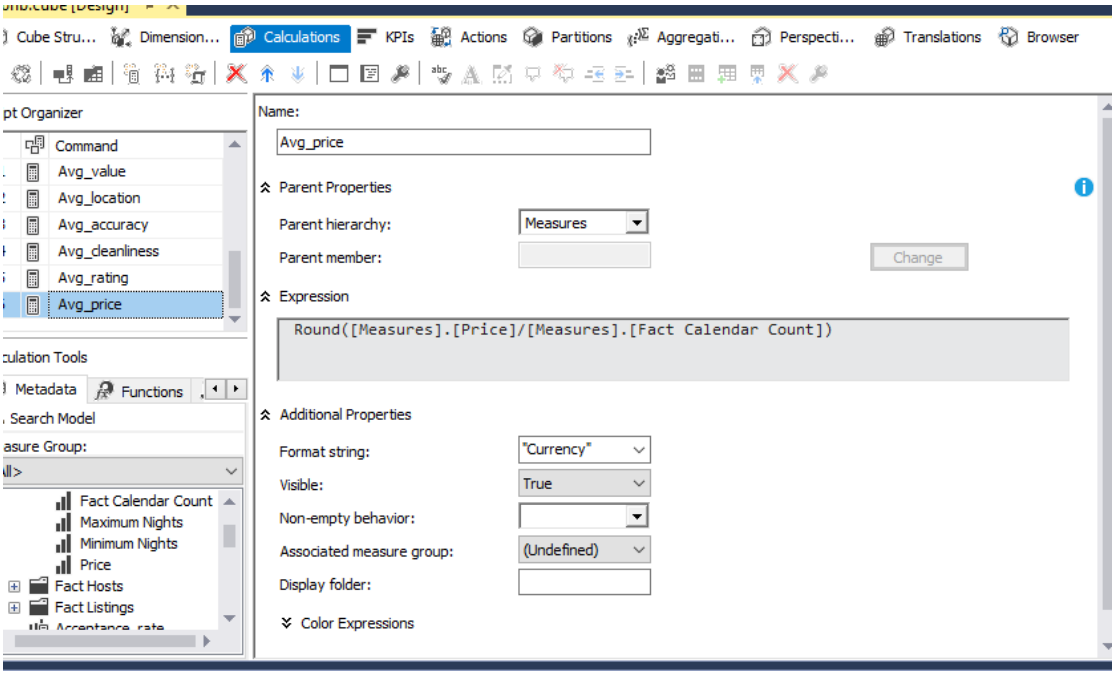
Avg_total_ratings= (Avg_accuracy+Avg_cleanliness+Avg_communication+Avg_location+Avg_value)/5



3. Average price

This measure calculates the rounded average price by using all the available dates.

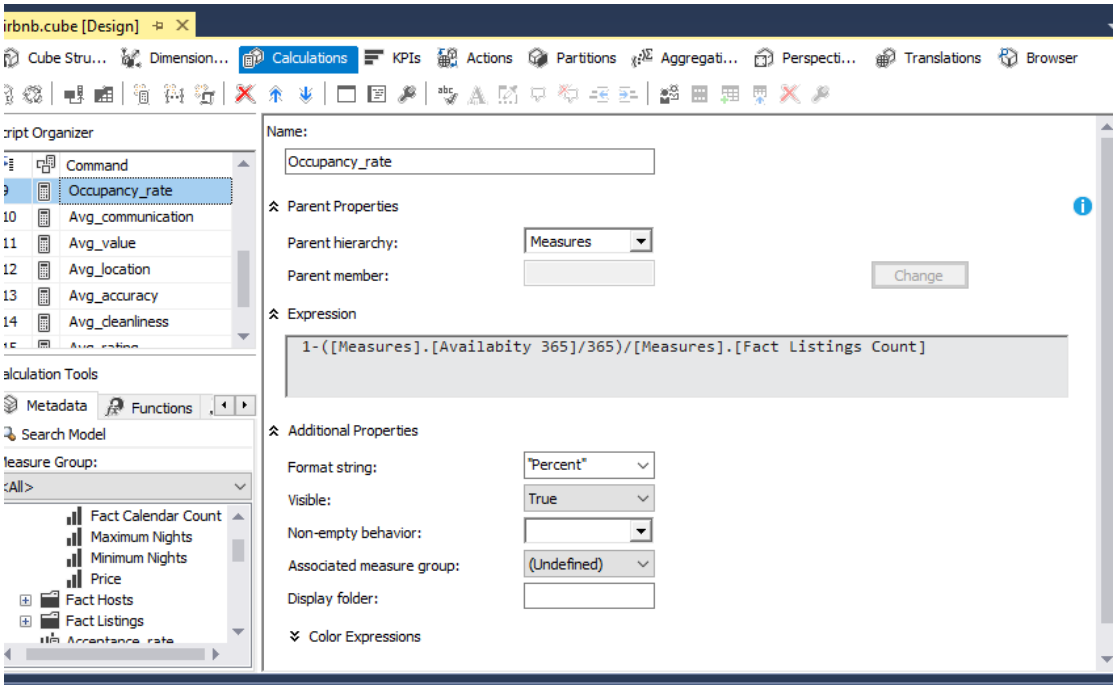
Avg_price= Price / Number of total dates



4. Occupancy rate

This measure calculates the occupation rate for each property. For the calculation we used the column availability 365 which represents the number of days the property was available within the year, so the rest days of the year refer to days when it was occupied.

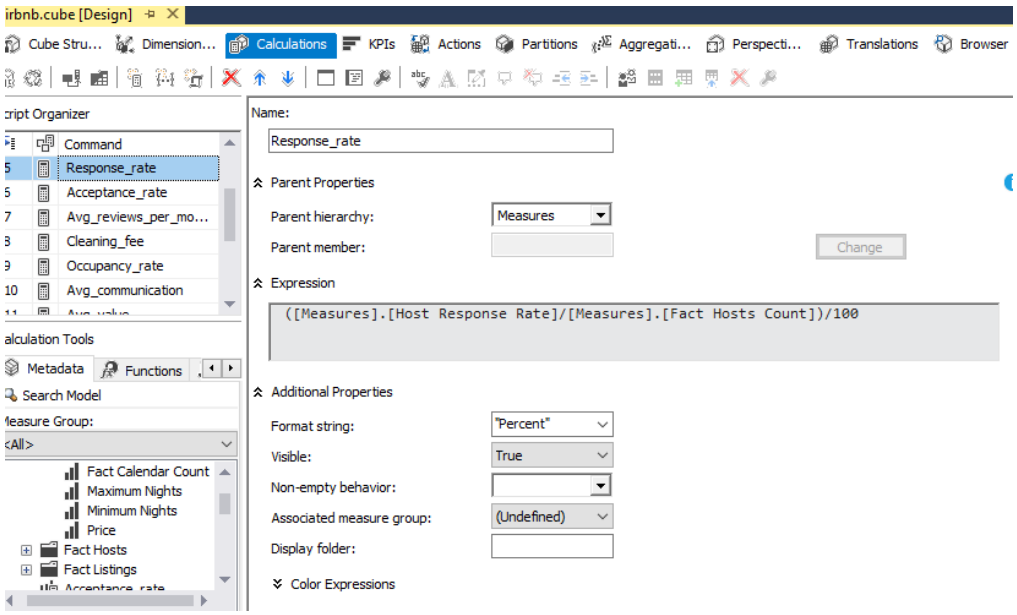
Rate= 1- (availability 365/365) / Number of total listings



5. Response rate

This measure calculates the percentage of response rate for the available number of hosts.

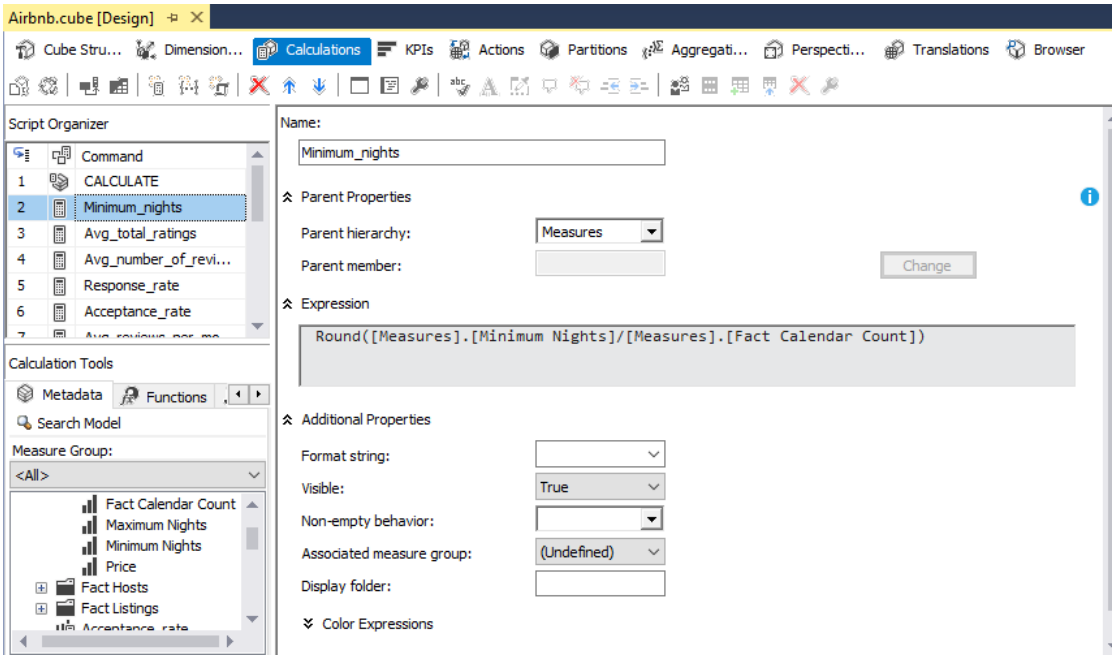
Response rate= Response rate / Number of total hosts



Following the same procedure, we also created the percentage of acceptance rate for the available hosts.

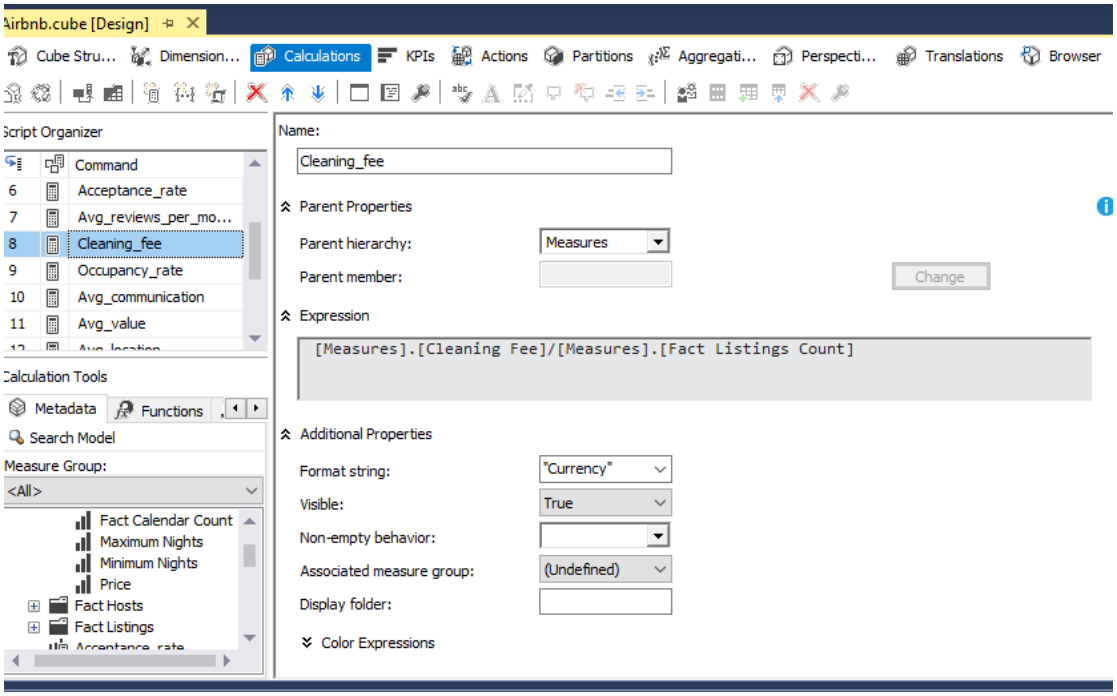
6. Minimum nights

This measure calculates the average minimum nights a property is available. For this calculation we used the fact calendar which includes this detail per day for each of the listed properties.



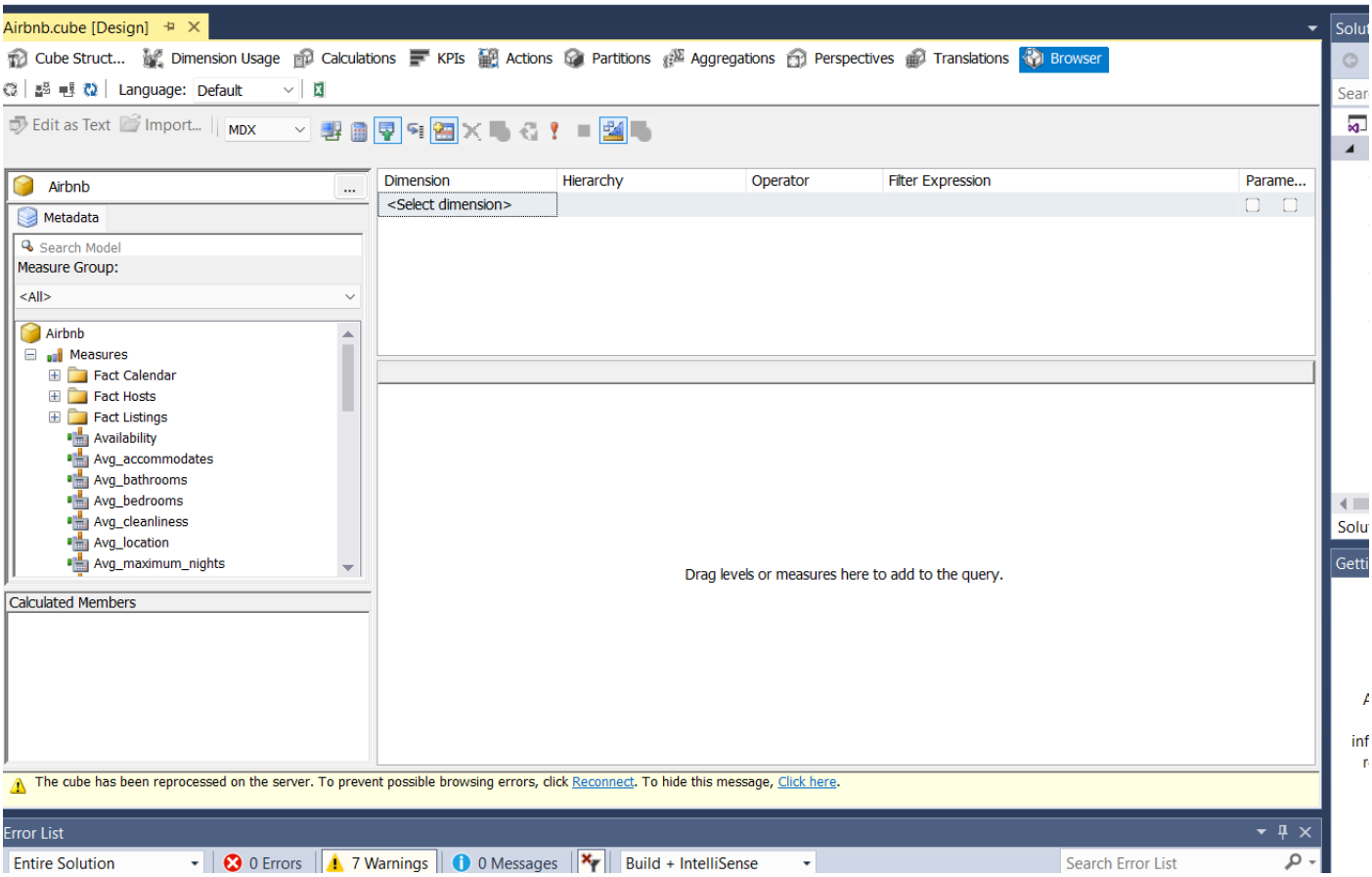
7. Cleaning fee

This measure calculates the average cleaning fee for the listed properties.



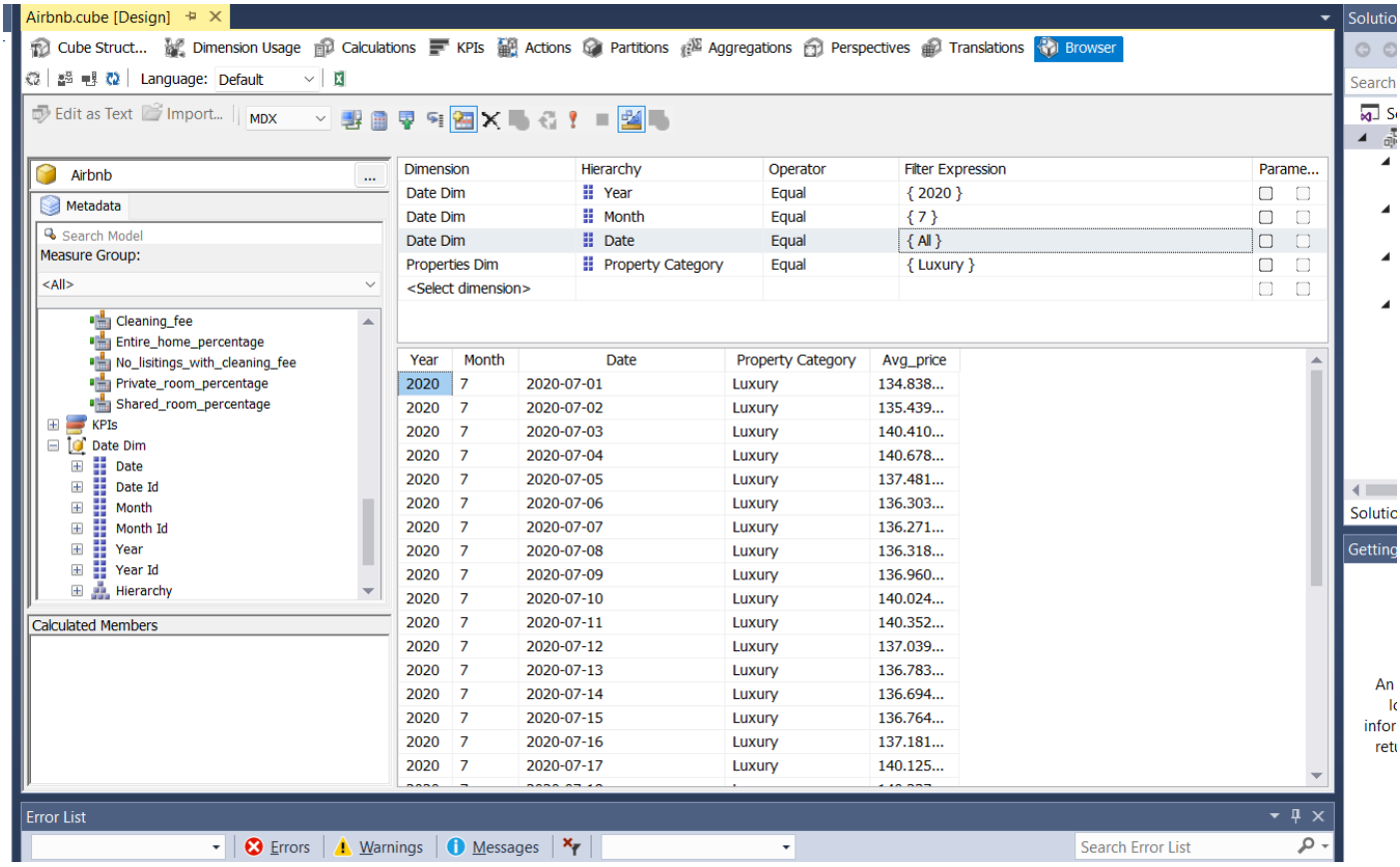
6.3. OLAP Reports

To create the OLAP reports, we used the Browser window in SSAS:



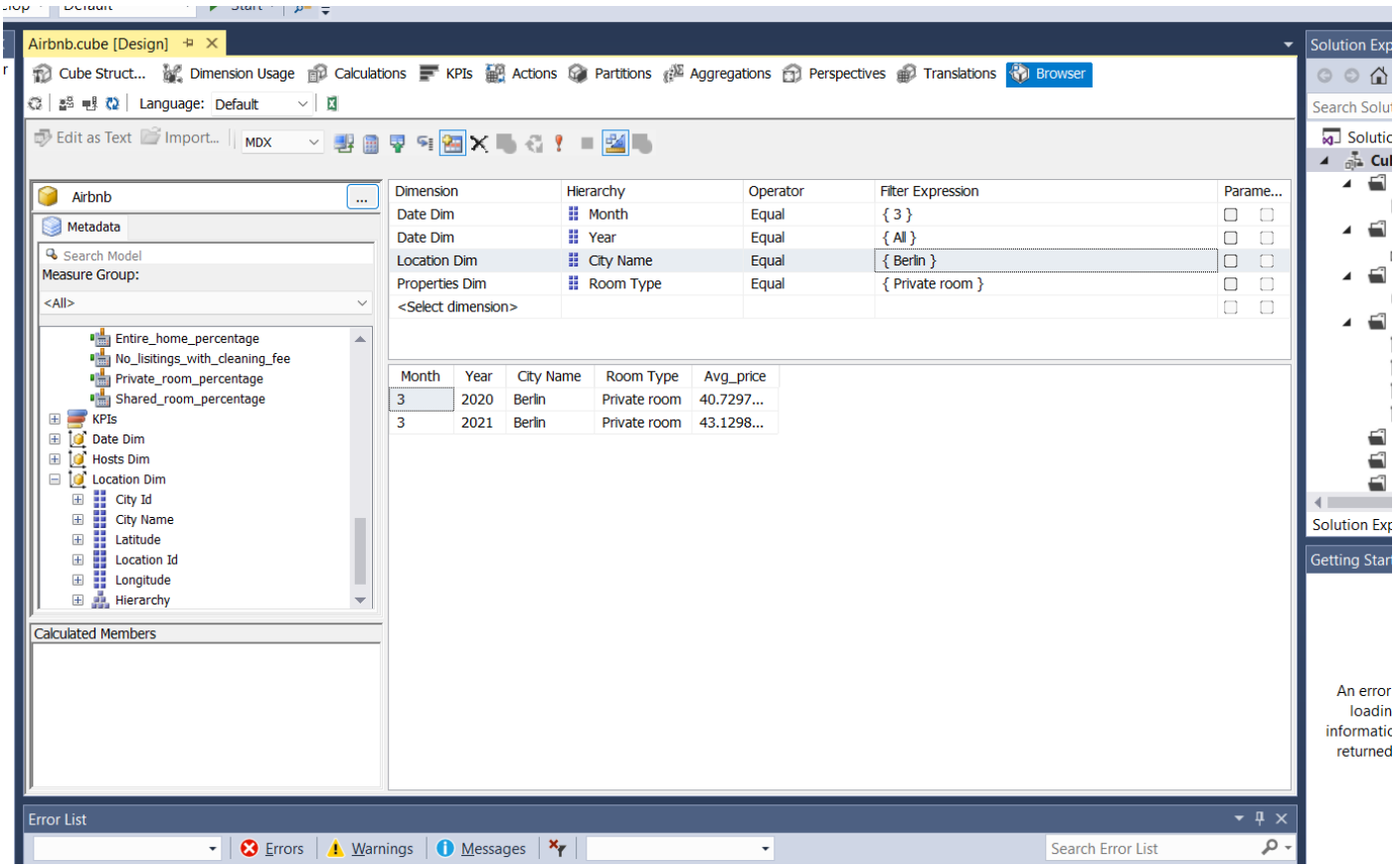
6.3.1. Drill Down

Cubes also have reporting capabilities. The drill-down operation is one of them and converts less-detailed data into more-detailed data through one of two methods: moving down in the concept hierarchy or adding a new dimension to the cube. In this project we created a drill down report, as we are moving from years to month and dates in order to find the average price of luxury properties in all available dates of **July 2020**.



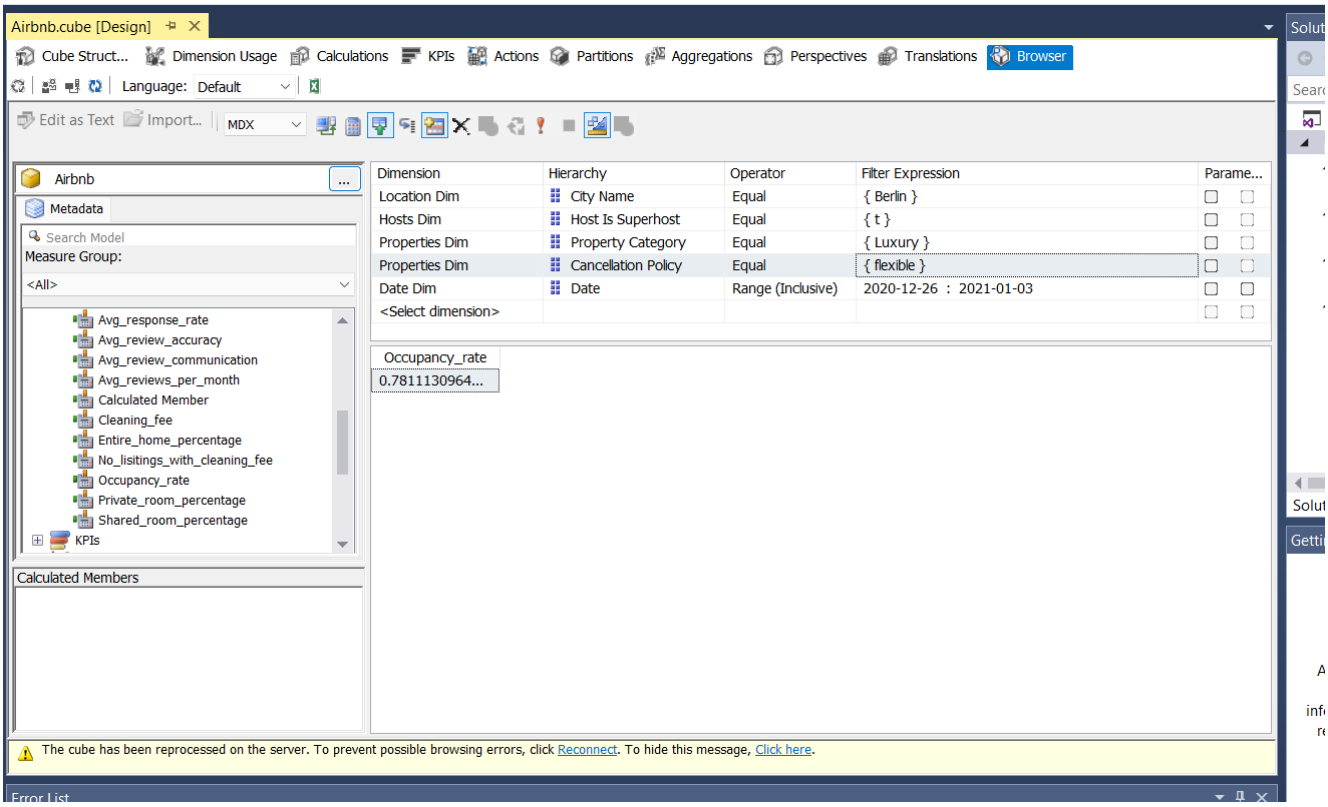
6.3.2. Roll up

Roll up is the opposite of the drill-down function—it aggregates data on an OLAP cube by moving up in the concept hierarchy or by reducing the number of dimensions. For the below analysis we used the roll up operation to show the average price of private rooms in Berlin and in March of both years.



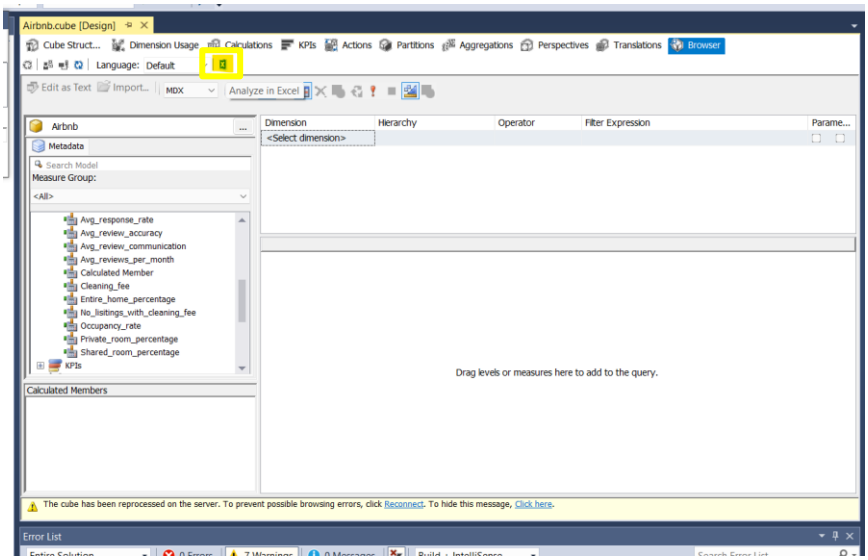
6.3.3. Slice and dice

The slice operation creates a sub-cube by selecting a single dimension from the main OLAP cube. The dice operation emphasizes two or more dimensions from a cube given and suggests a new sub-cube, as well as slice operation does. In this project the slice and dice operation was used to show the occupancy rate of luxury properties in Berlin which belong to superhosts and have flexible cancellation policy during Christmas period (26/12/2020-03/01/2021).

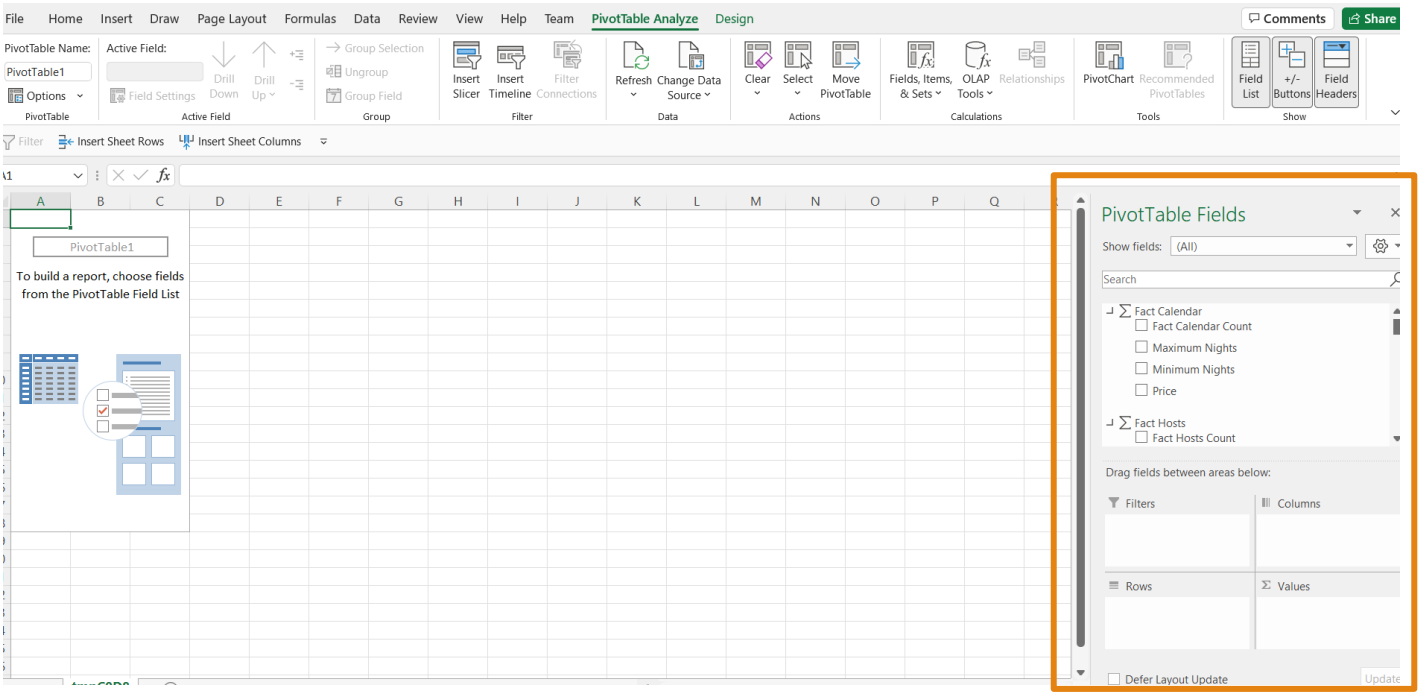


6.3.4. Excel reports

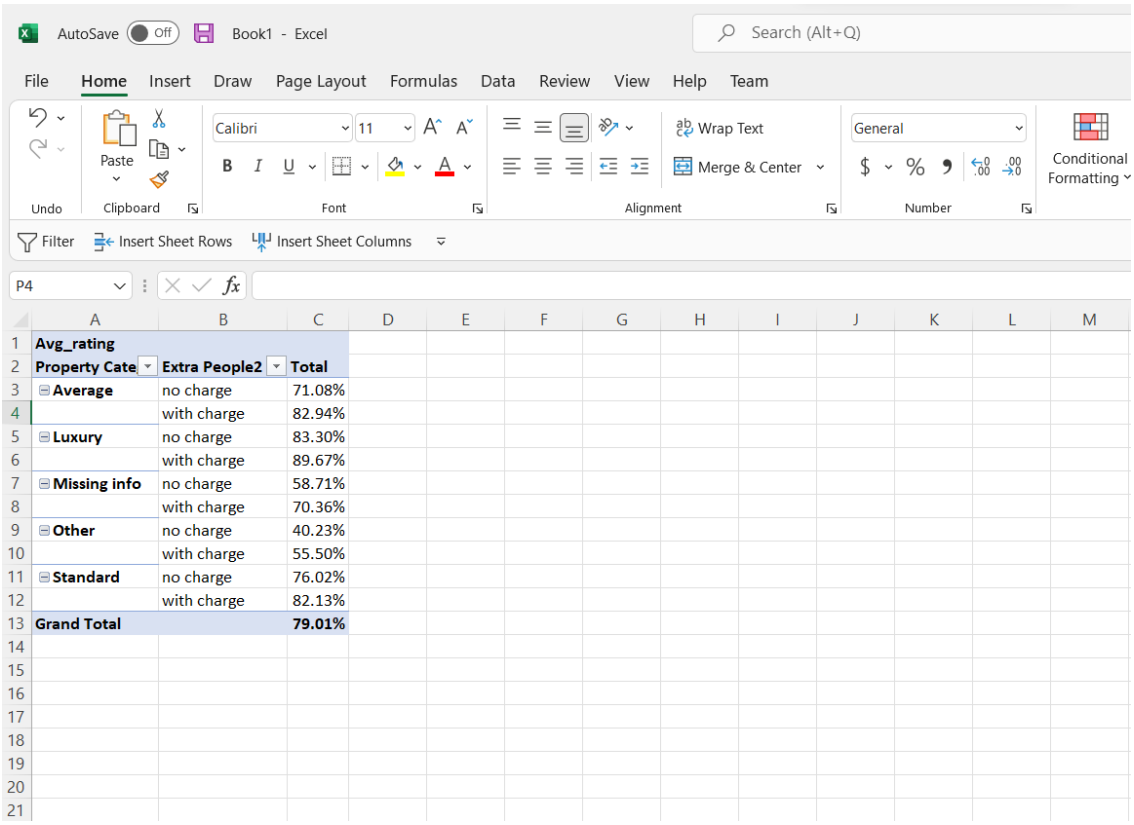
We created also an excel report by exporting the data to excel app as shown below:



Once we extract the data, a pivot table is automatically created in excel workbook and by using the right fields we can build further and detailed analysis.



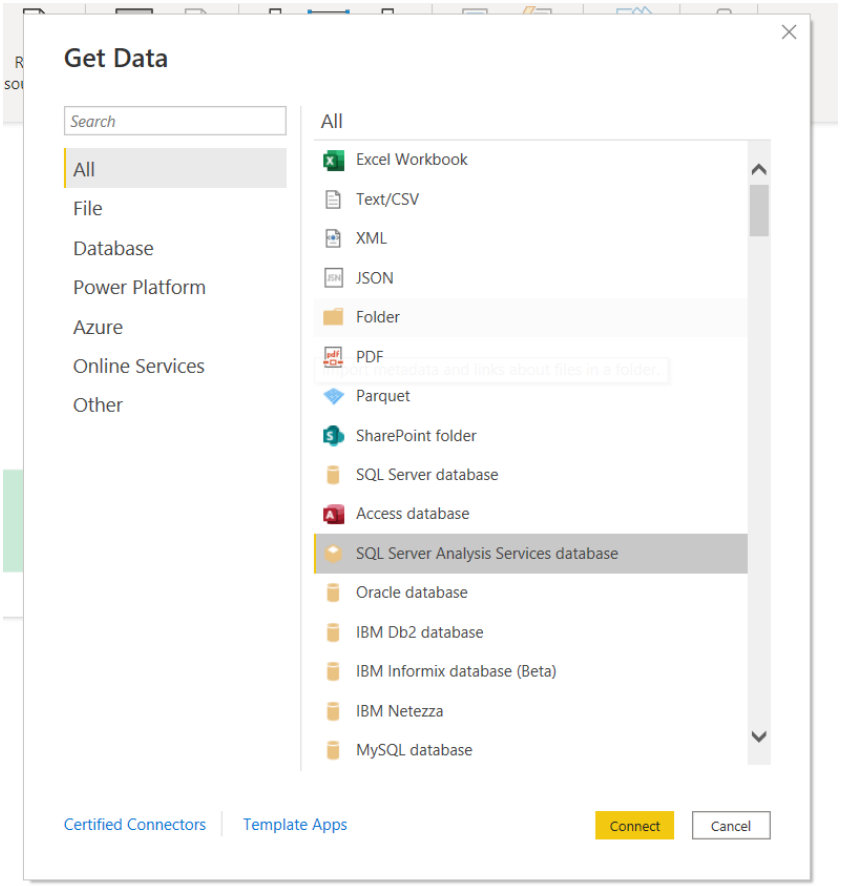
Thus, by selecting property type option and 'Extra people2' category, which represents whether a property has extra charge for the extra guests, we can take a quick report which shows the average rating for each property category and for each kind of extra guests charge per property category.

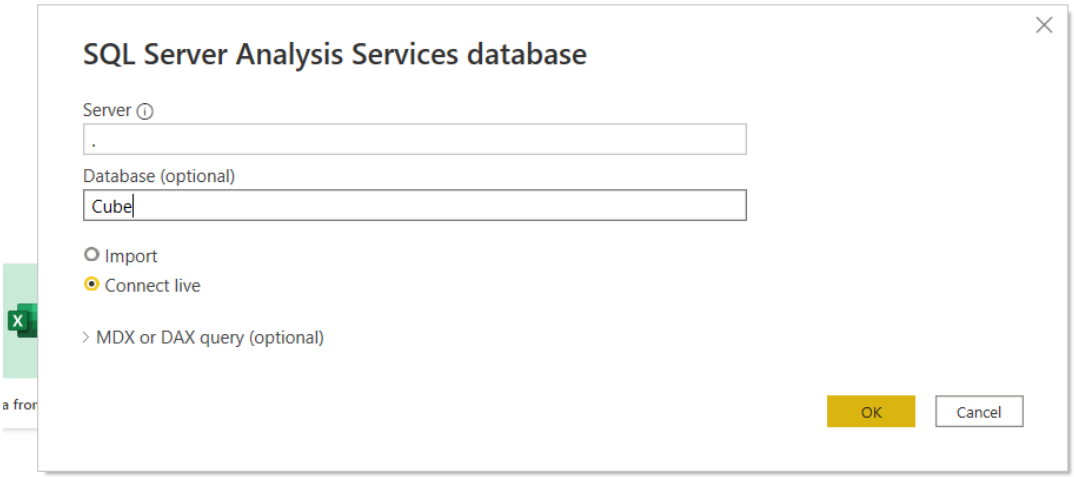


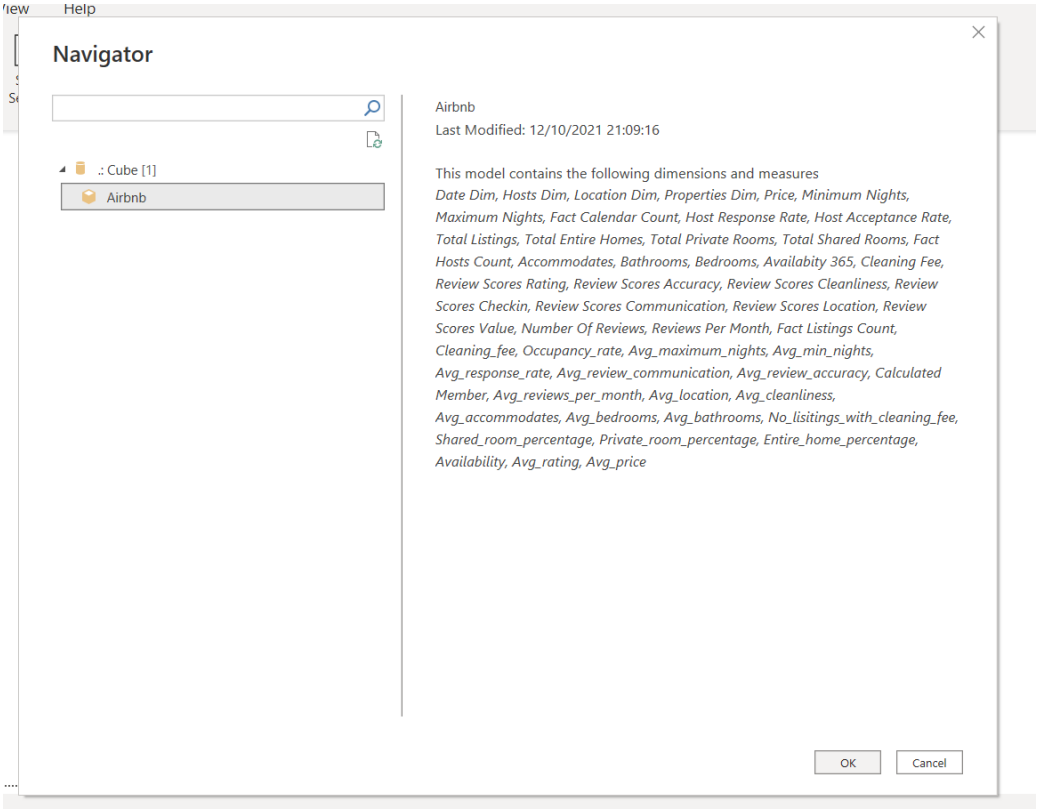
	A	B	C
1	Avg_rating		
2	Property Cate	Extra People2	Total
3	Average	no charge	71.08%
4		with charge	82.94%
5	Luxury	no charge	83.30%
6		with charge	89.67%
7	Missing info	no charge	58.71%
8		with charge	70.36%
9	Other	no charge	40.23%
10		with charge	55.50%
11	Standard	no charge	76.02%
12		with charge	82.13%
13	Grand Total		79.01%

7. Data visualizations

The final step of the analysis that enables us to draw conclusions regarding our inquiries is data visualization. For that purpose, SQL Server Analysis Services is connected live to Power BI Desktop. The process is displayed in below screenshots:







7.1.1. Top priced properties

The following dashboard depicts the top priced listings based on different scenarios. Slicers can be used to examine each scenario by checking different options. Specifically, the slicers used are: city, bedrooms, bathrooms and accommodates. Additionally, we use three charts, namely, two donut charts, a funnel, a bar chart and several cards, in order to produce some statistical inferences.

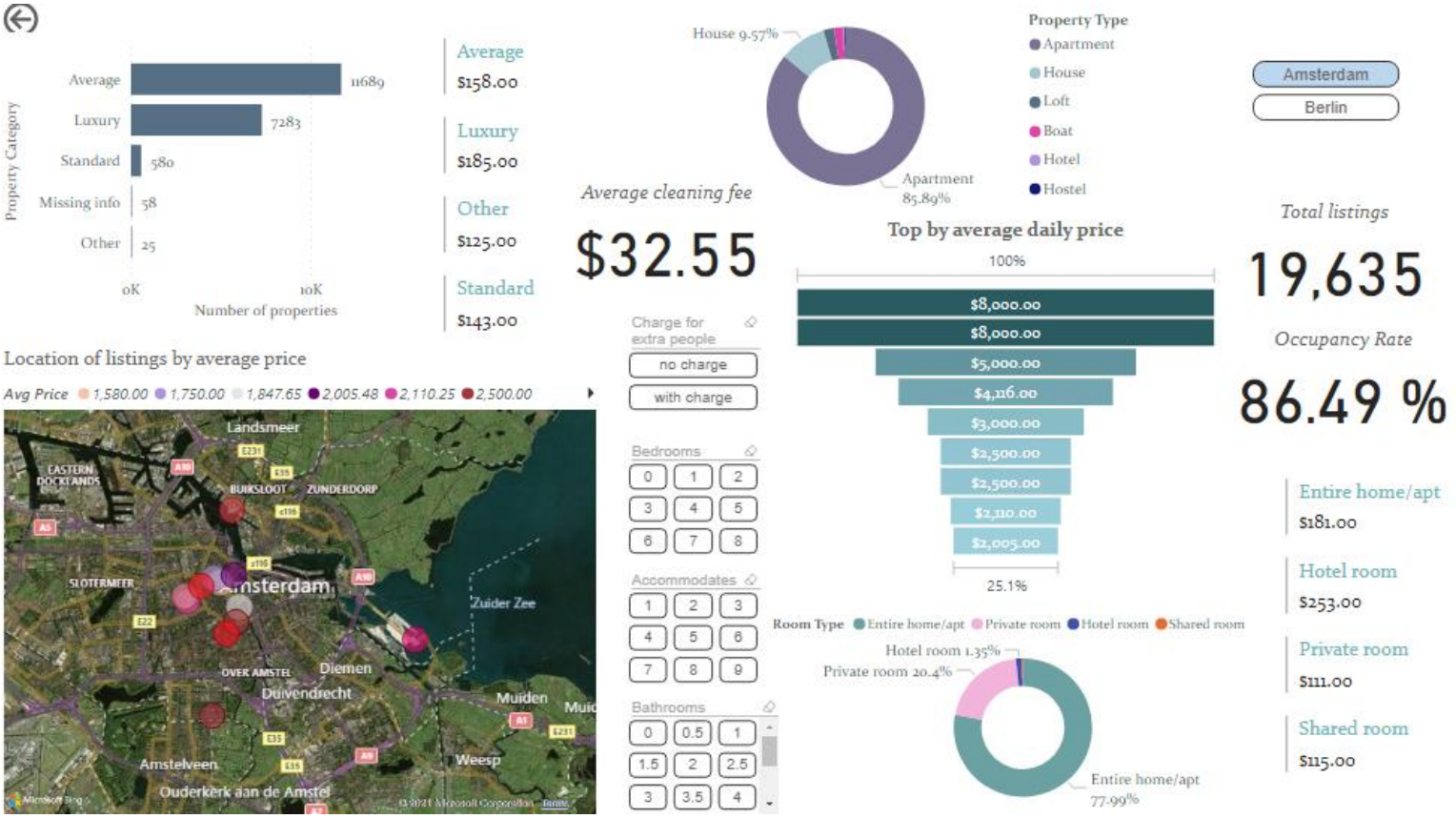
The purpose of putting slicers on the report canvas is to easily access important filters and provide the user with a focused report by placing the slicer right next to essential graphs and charts.

The charts display some general statistics about all kinds of properties and change simultaneously according to selected options. The bar chart shows the listings count for every property category and the multi row card on the side displays the average price for each category. As expected, for both cities, the average price for the luxury properties is the highest, average properties come next and standard/other are the lowest, as they provide fewer amenities.

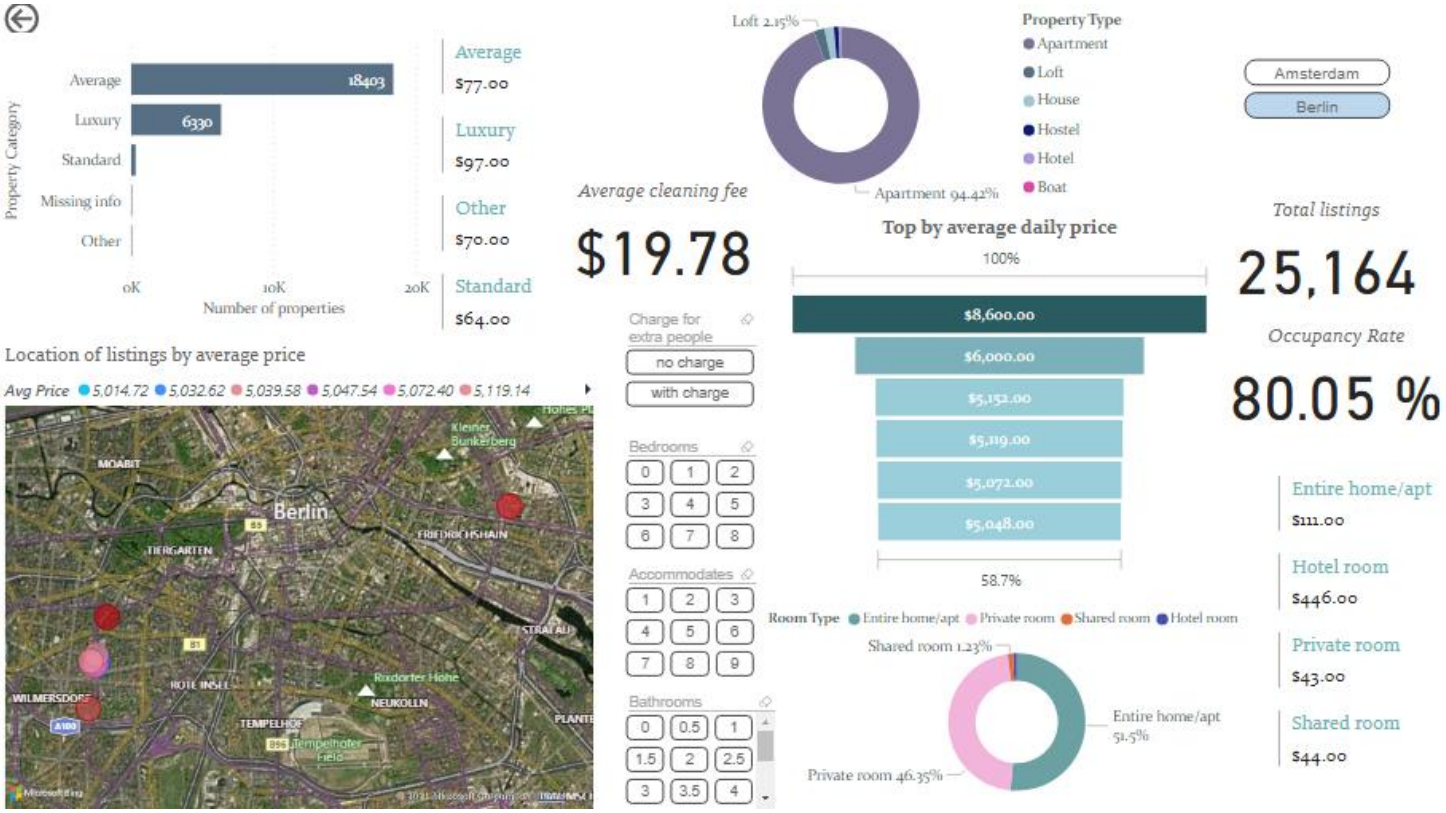
The donut charts illustrate the associated percentage for some property and room types. It can be easily seen that the highest percentage in both cities corresponds to apartments, but the other differ a bit between cities. In Amsterdam, houses have a much larger percentage as well as boats and lofts that also take up a small percentage, while in Berlin houses and lofts are fewer. For the room types, entire homes/apartments are the majority for Amsterdam, while in Berlin private rooms have an almost equal percentage. The average prices for the room types are displayed on the side and show that hotel rooms are by far the most expensive.

The funnel summarizes the top highest priced properties, which cost 8000$ and can be also seen on the map (the ones colored in red are the most expensive) and they are located near the canal. It should be noted that for Berlin, an outlier was excluded because the price was not considered normal (>200,000$). Moreover, the average cleaning fee in Amsterdam is higher and the yearly occupancy rate almost the same.

Amsterdam:



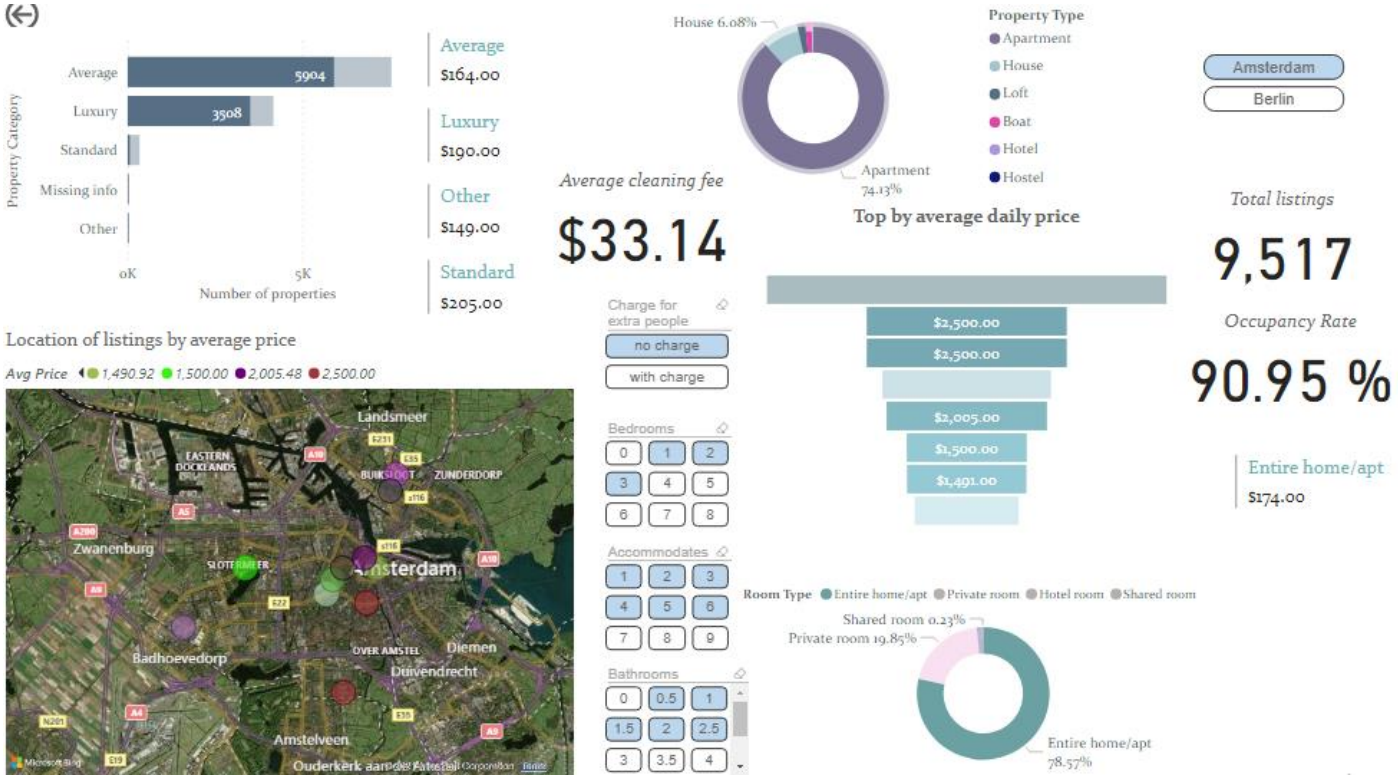
Berlin:



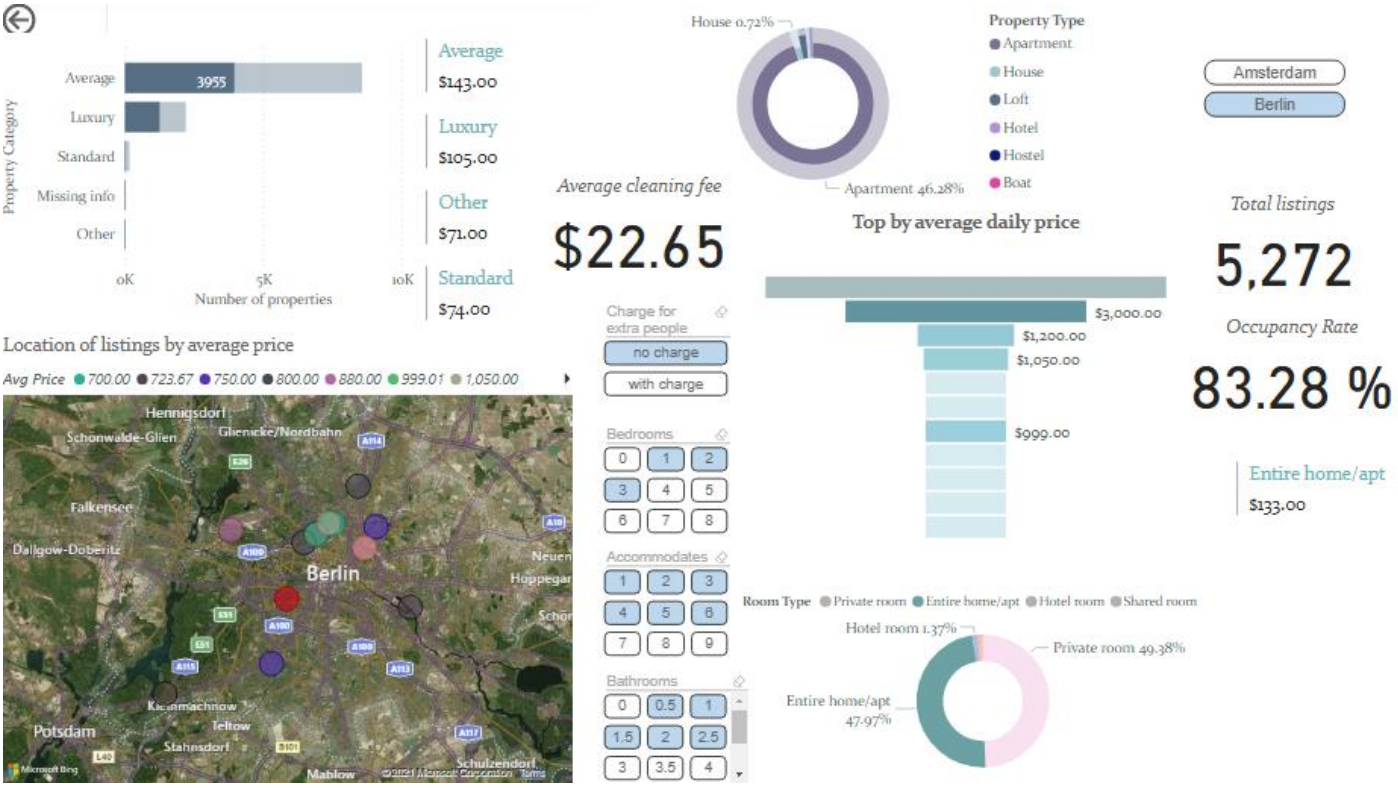
In the following scenarios, we select the properties which have a maximum of three bedrooms, a maximum of six accommodates, a maximum of 2.5 bathrooms, are of type entire home/apartment and have no charge for extra people, because we think they correspond to a representative sample of listings. It can be observed that the highest prices now are at 2,500$(Amsterdam)/3,000$(Berlin), the average price is 174$(Amsterdam),133$(Berlin), but it also depends on the category. The standard property category is more expensive in Amsterdam because it is influenced by the high

average hotel room price. In Berlin the top average prices are lower, as it seen on the map and on average property category prices.

Amsterdam:

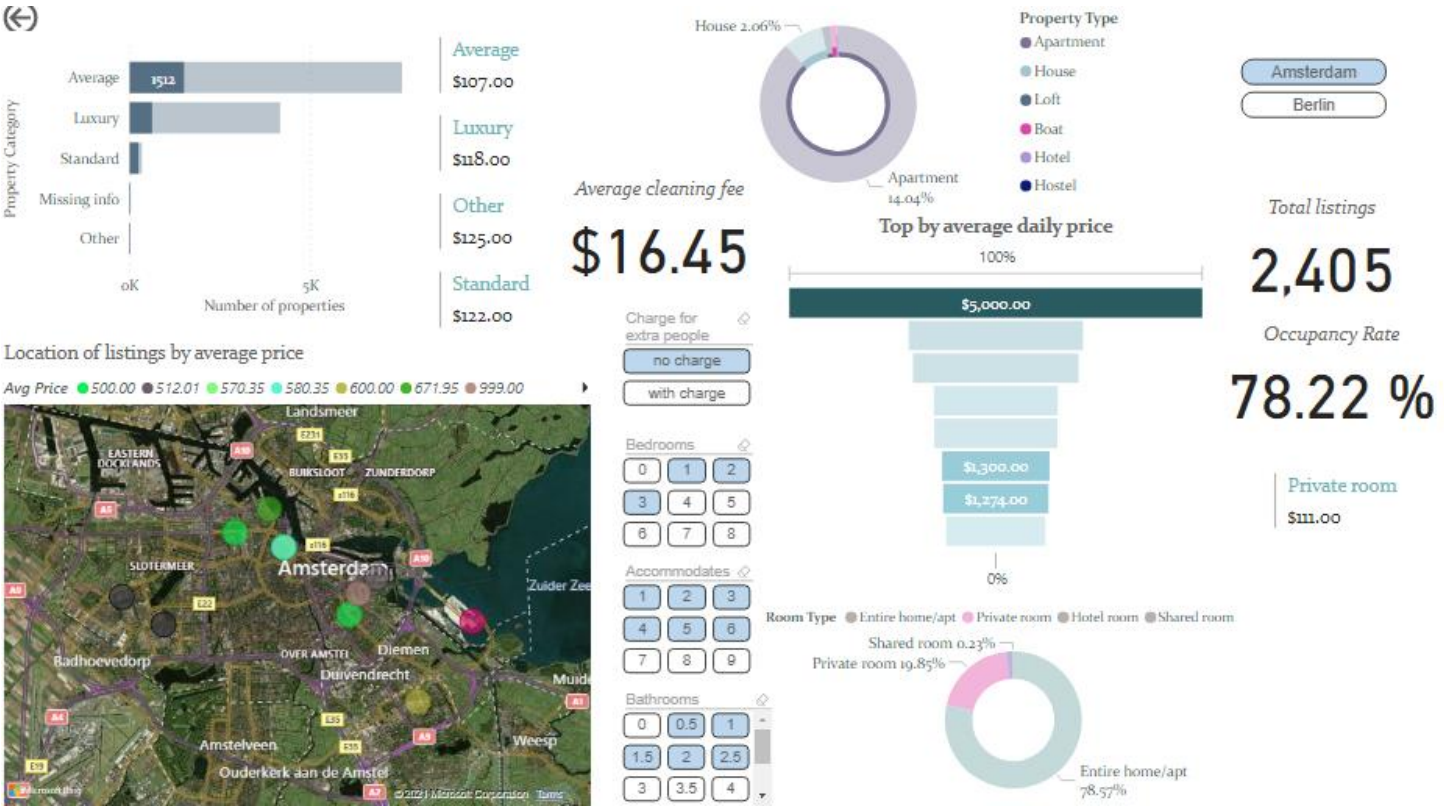


Berlin:

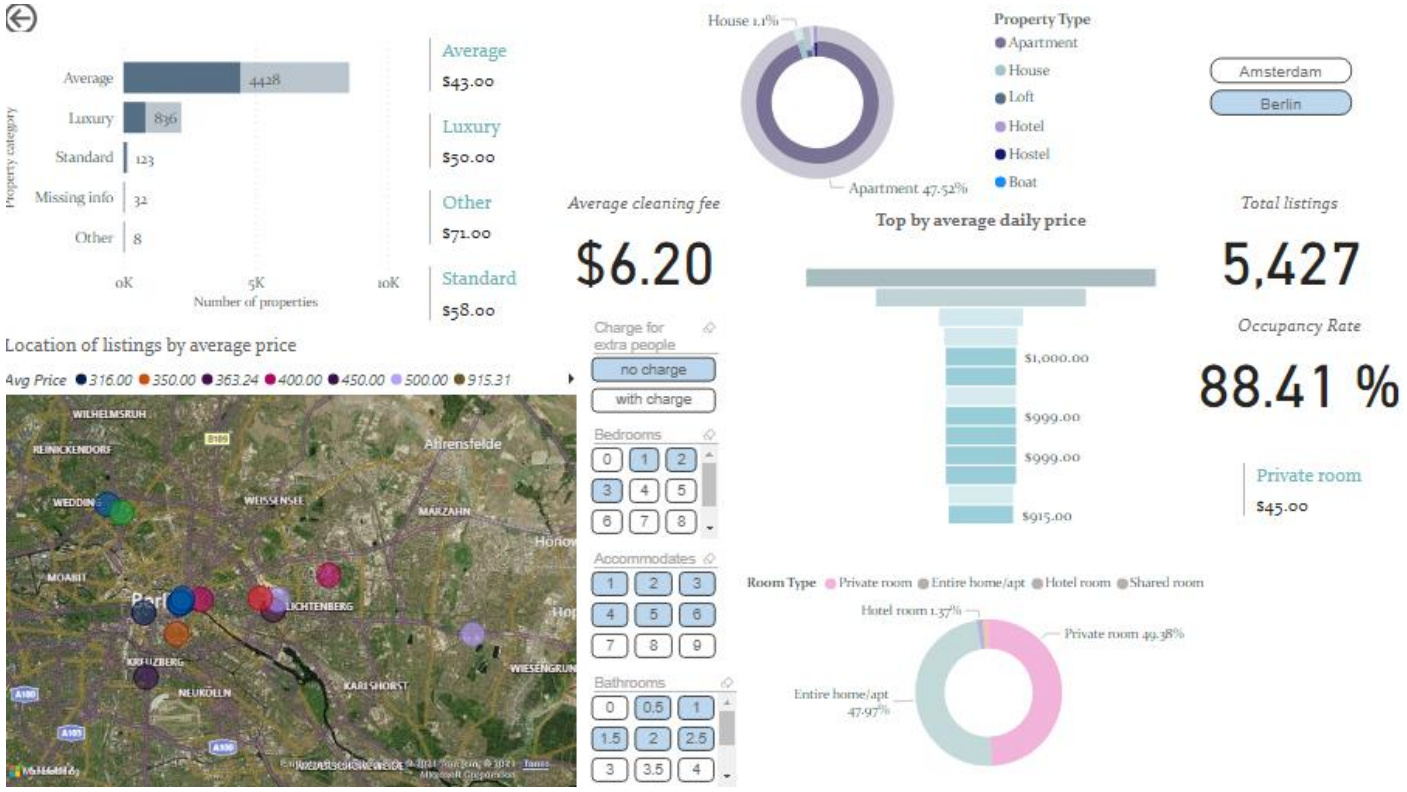


For private room type in Amsterdam and keeping the other filters same, we can see that the highest price is 5000$, but the majority are significantly lower. As depicted on the map, the next top prices are between 500-1300 and the cleaning fee is lower than entire homes, although the occupancy rate is also lower. In Berlin, the top prices and the cleaning fee for private rooms are lower than those in Amsterdam and the average prices for each property category are lower than the entire homes.

Amsterdam:

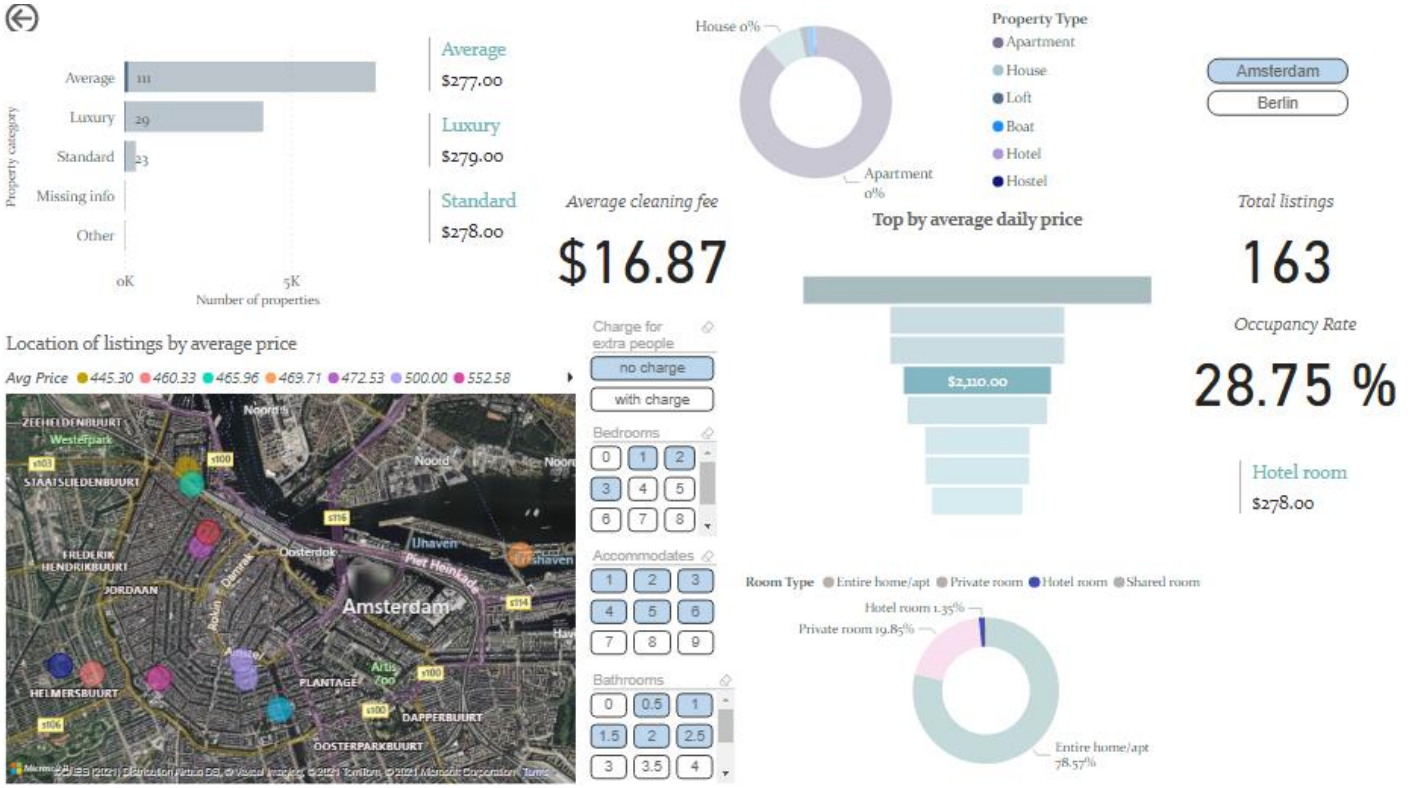


Berlin:



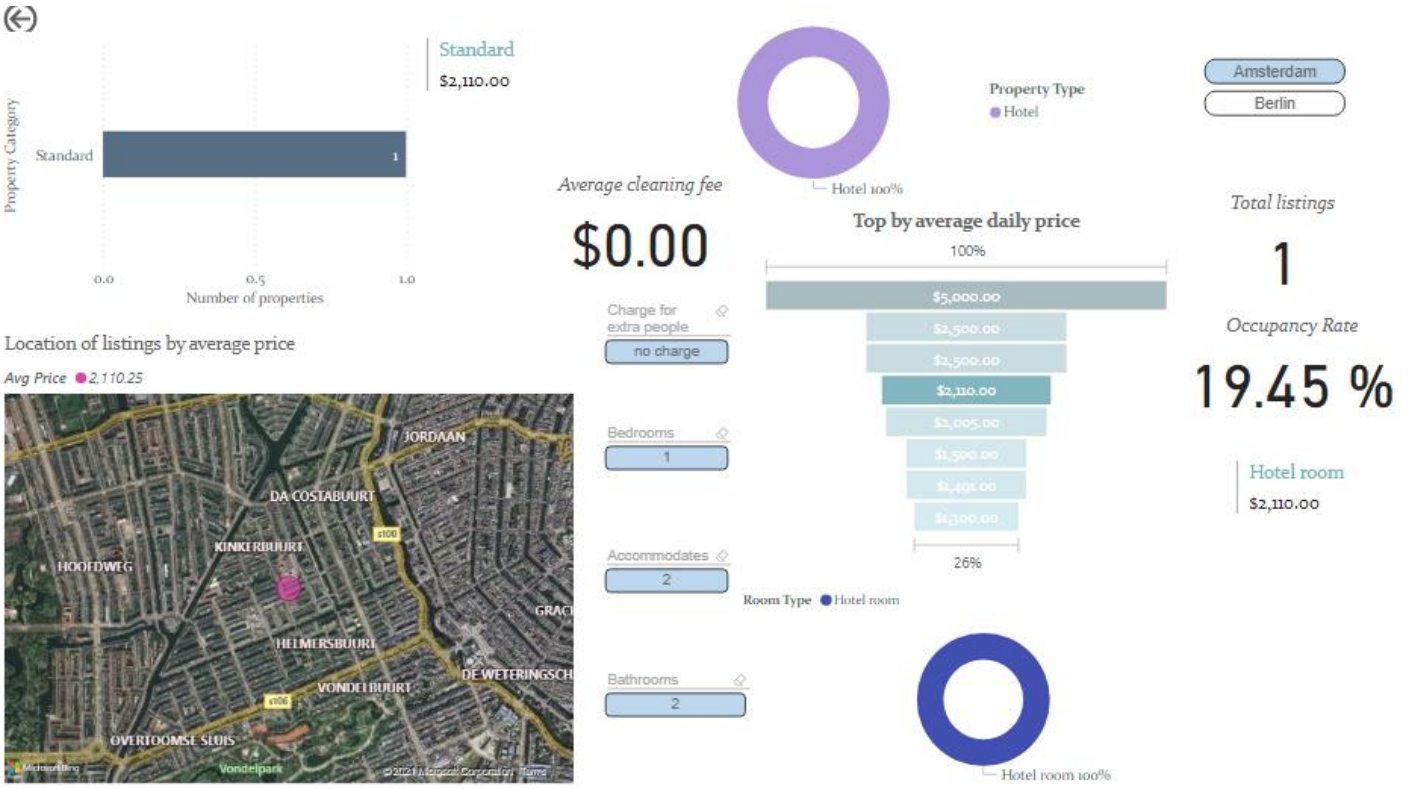
The top priced hotel rooms in Amsterdam are few and have high prices and average cleaning fee, are located in different neighborhoods and the occupancy rate is low which means that they are not very popular due to high prices.

Amsterdam:



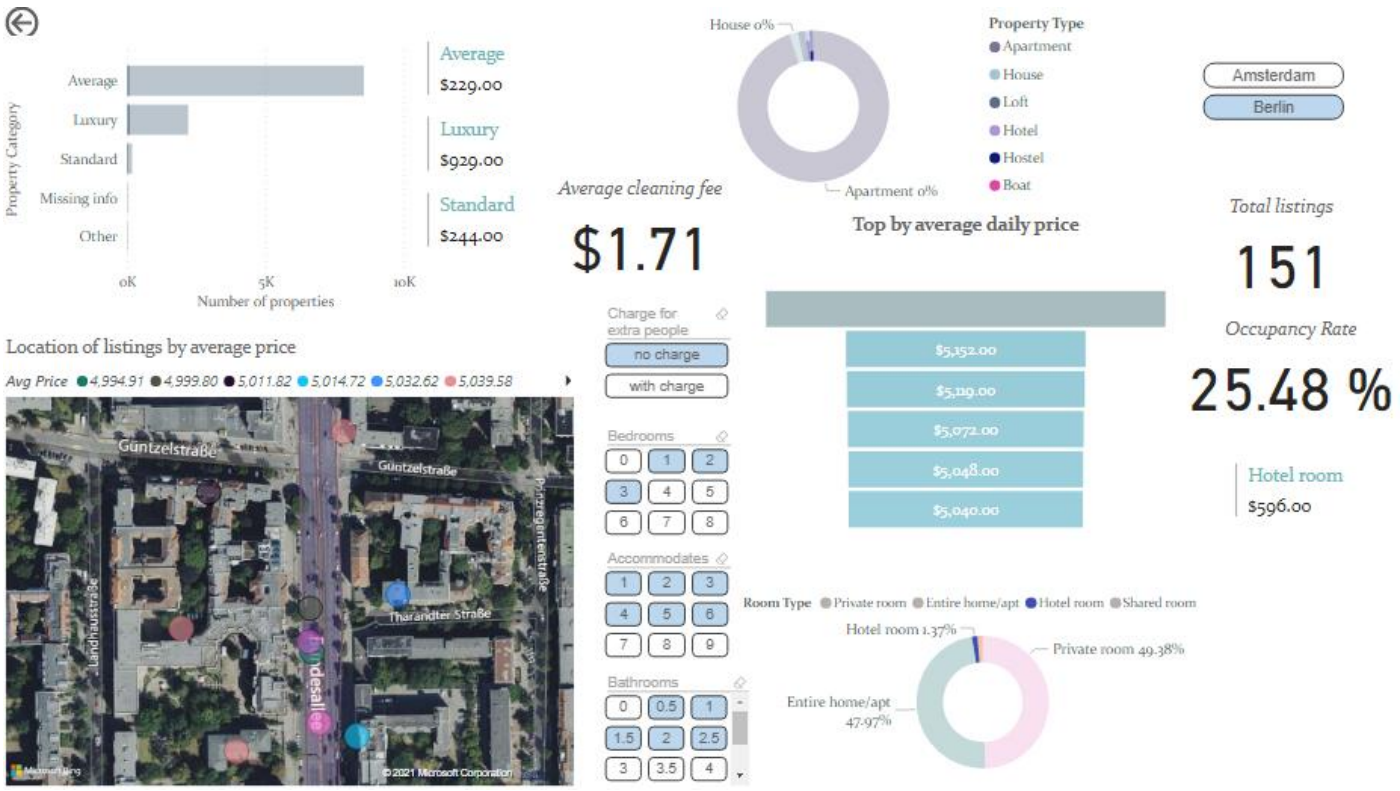
By selecting the top priced hotel room for Amsterdam, we observe its specific characteristics.

The top priced hotel room in Amsterdam is at 2,110$ and is a double room with 2 bathrooms and is of standard property category, as it offers only the basics, but the occupancy rate is very low, which means it is not so popular.



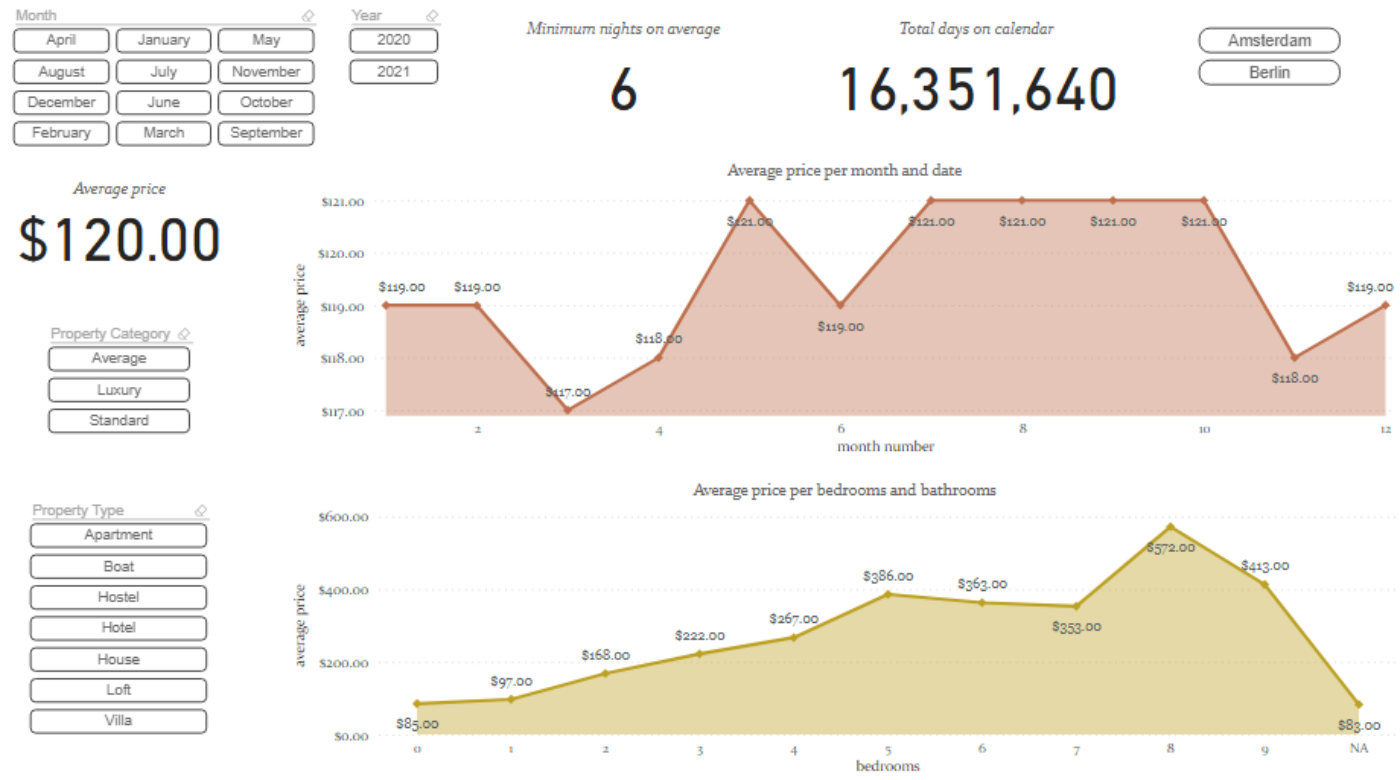
The top priced hotel rooms in Berlin are few and have very high prices, around 5,000$, are of luxury property category and are located in the same neighborhood as seen on the map. But the occupancy rate is low which means that they are not very popular due to high prices.

Berlin:



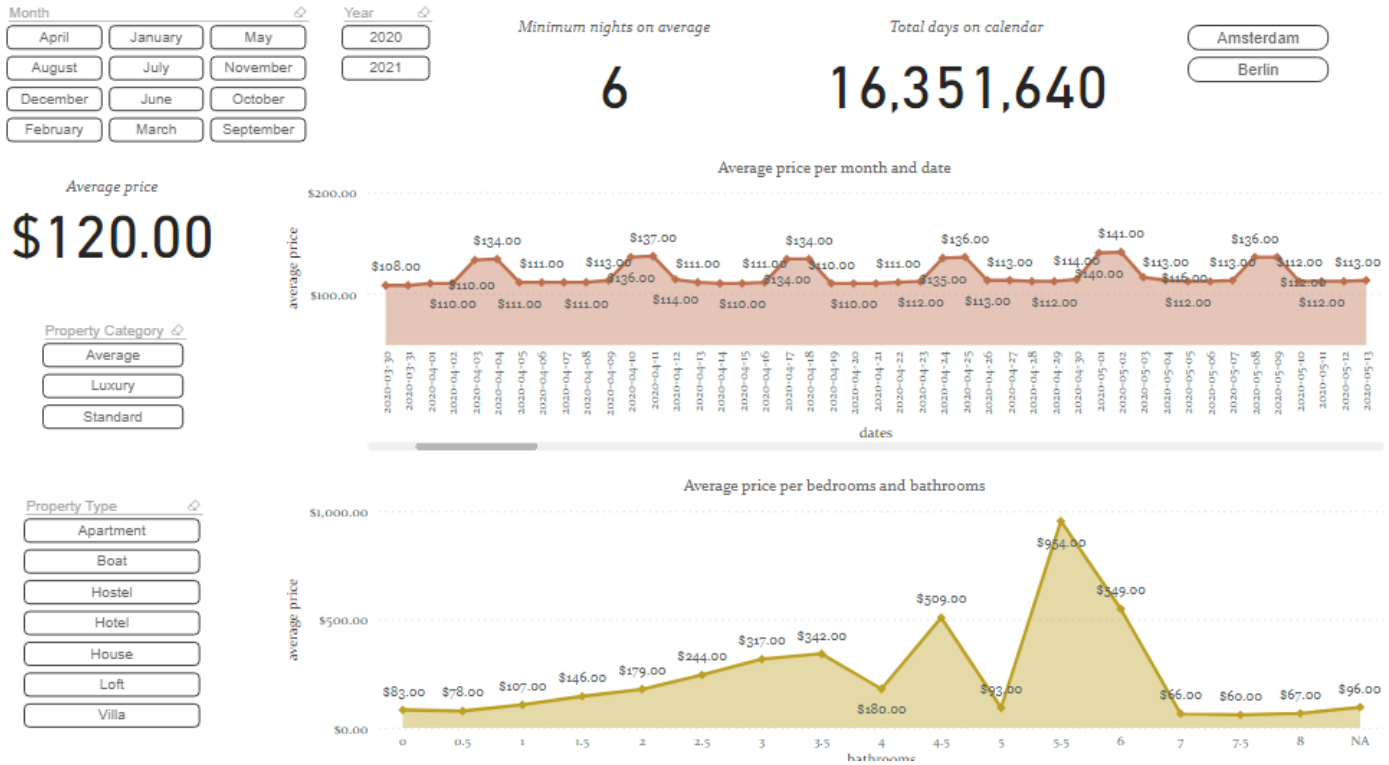
The shared rooms have the lowest prices among all room types for both cities, therefore they are not included in the analysis for top priced properties.

At the area charts provided below, we can observe how prices generally fluctuate during the year and depending on the bedrooms/bathrooms.



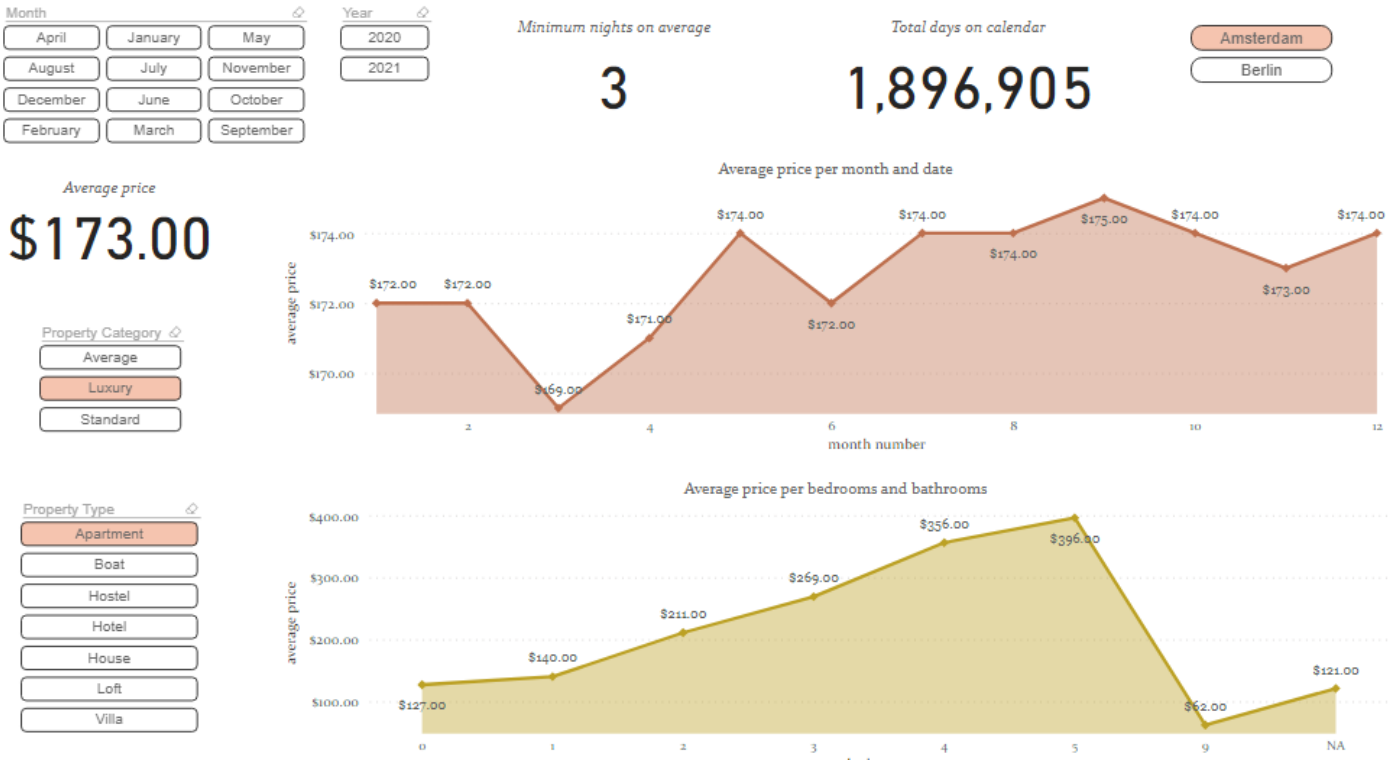
By selecting the slicers on the side, we can filter the results based on city, month, year, property category and property type.

Through a drill-down on the dates, we aim to examine the price trend and we come to the conclusion that there is a sharp increase on the price on Fridays and Saturdays. Additionally, drilling-down on bathrooms, we can see that there is a gradual increase until 3.5 bathrooms and then there are two price spikes at 4.5 and 5.5, which is justified by the increase of demand, while after that the price plummets because the customers are indifferent to more than 7 bathrooms.

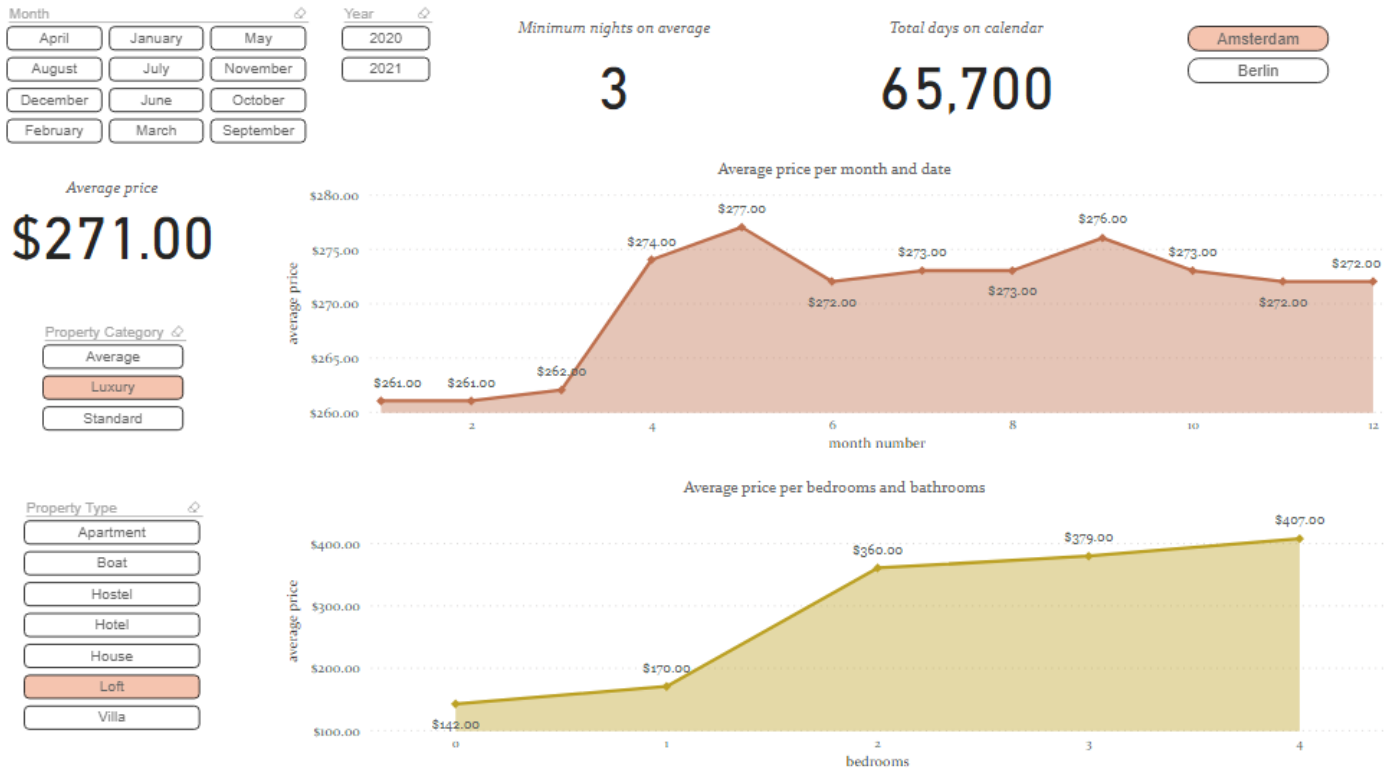


Subsequently, within the context of the top priced properties analysis, we selected the luxury property category in Amsterdam and compared the apartments with the lofts regarding price range and bedrooms. It is observed that the prices are much higher for the lofts, but the available bedrooms for lofts are up to 4, while apartments go up to 9. The price per bedrooms has an upward trend for both types.

<u>Luxury apartments Amsterdam:</u>

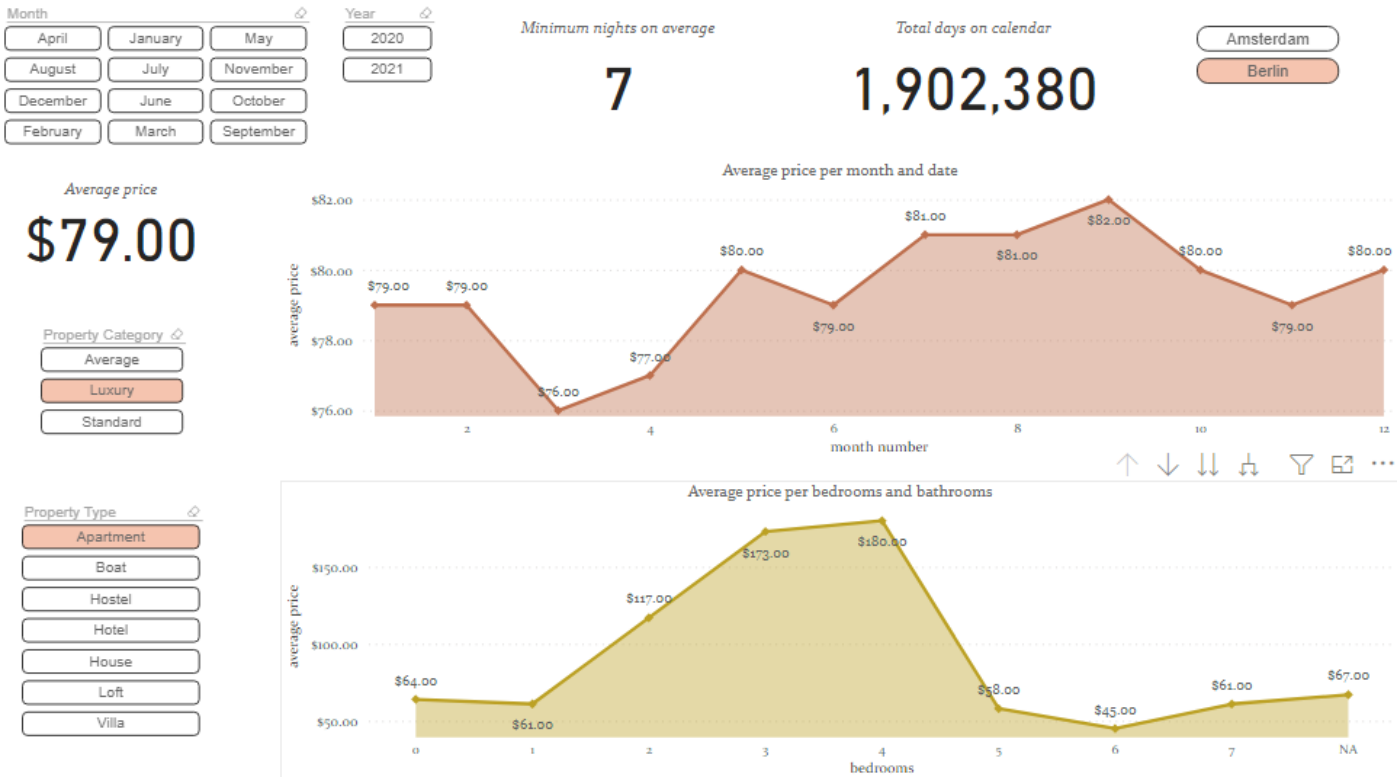


Luxury lofts Amsterdam:

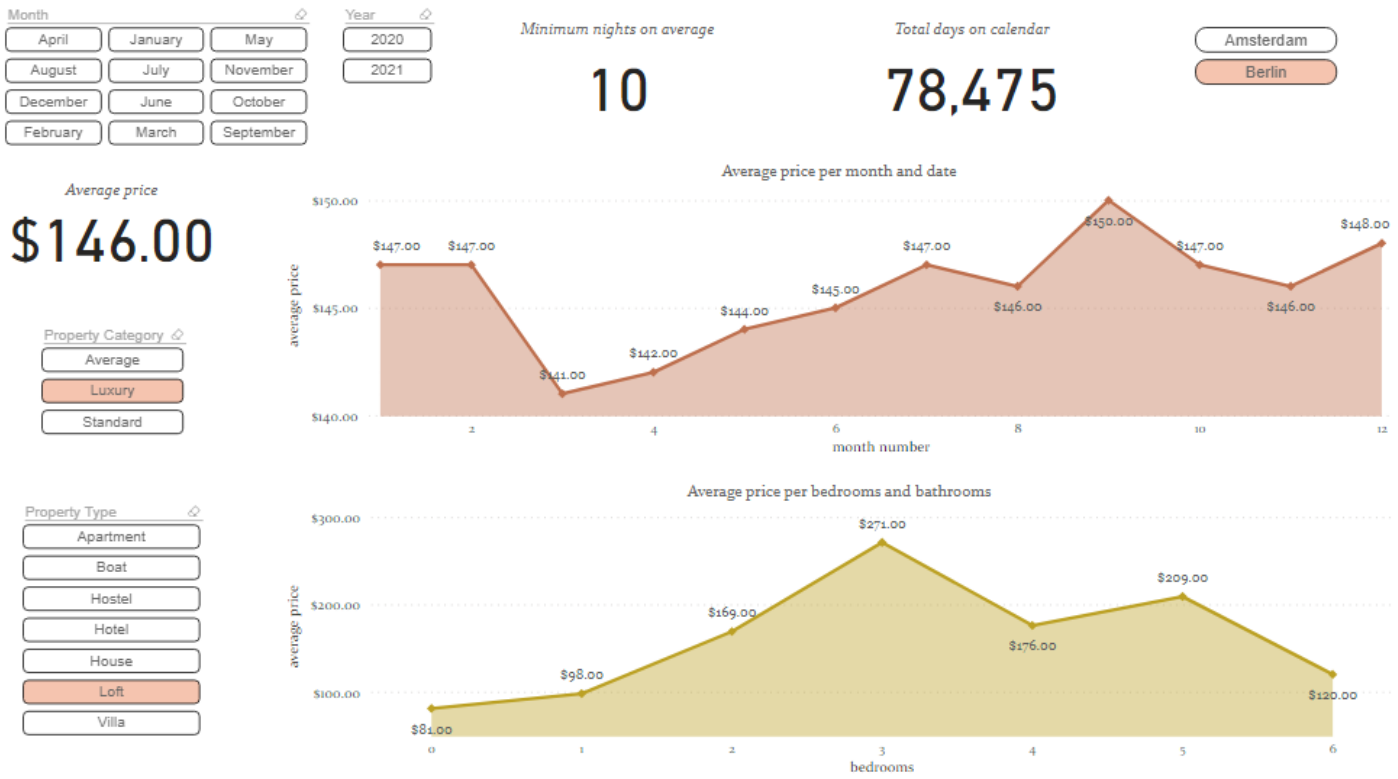


With regards to Berlin, the prices of luxury type are generally lower than those of Amsterdam, but similarly to Amsterdam the lofts are priced higher than apartments. Moreover, luxury lofts in Berlin have more available bedrooms (up to 6).

Luxury apartments Berlin:



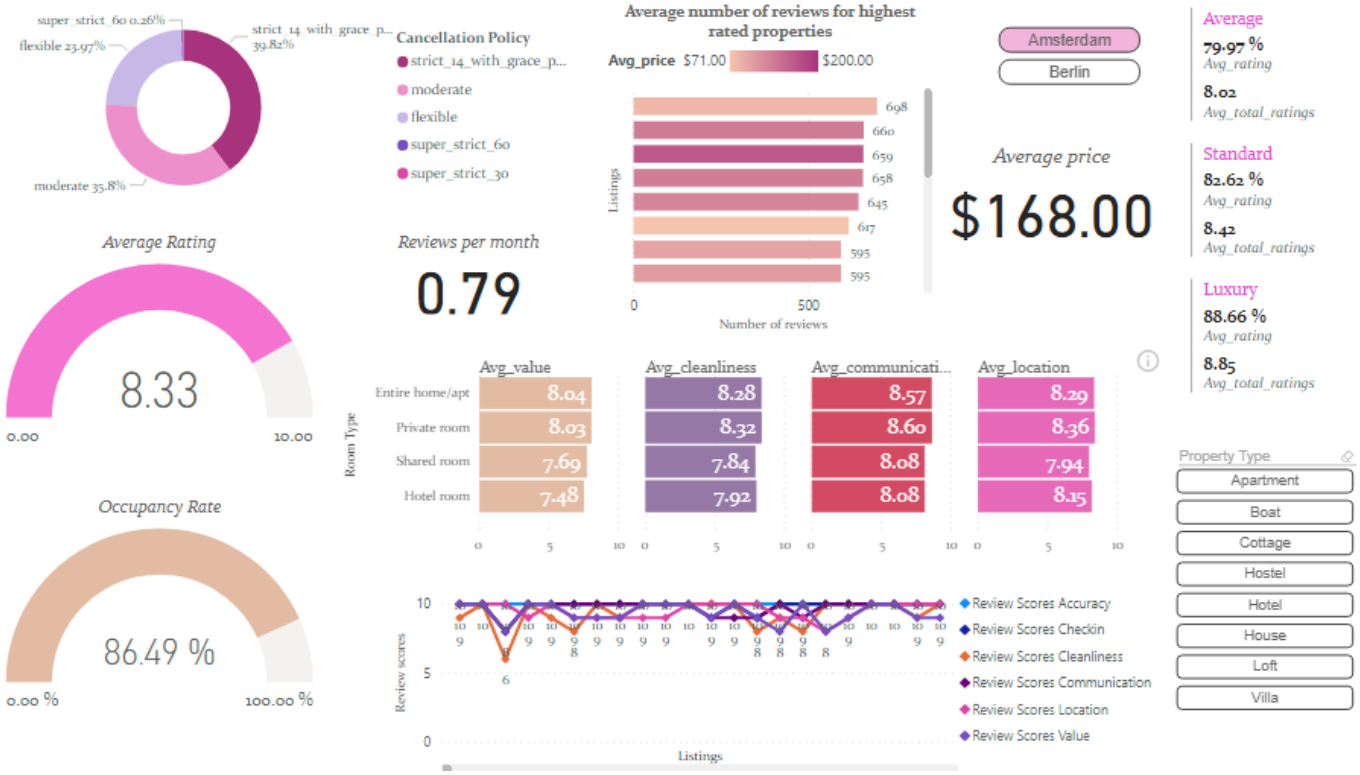
Luxury lofts Berlin:



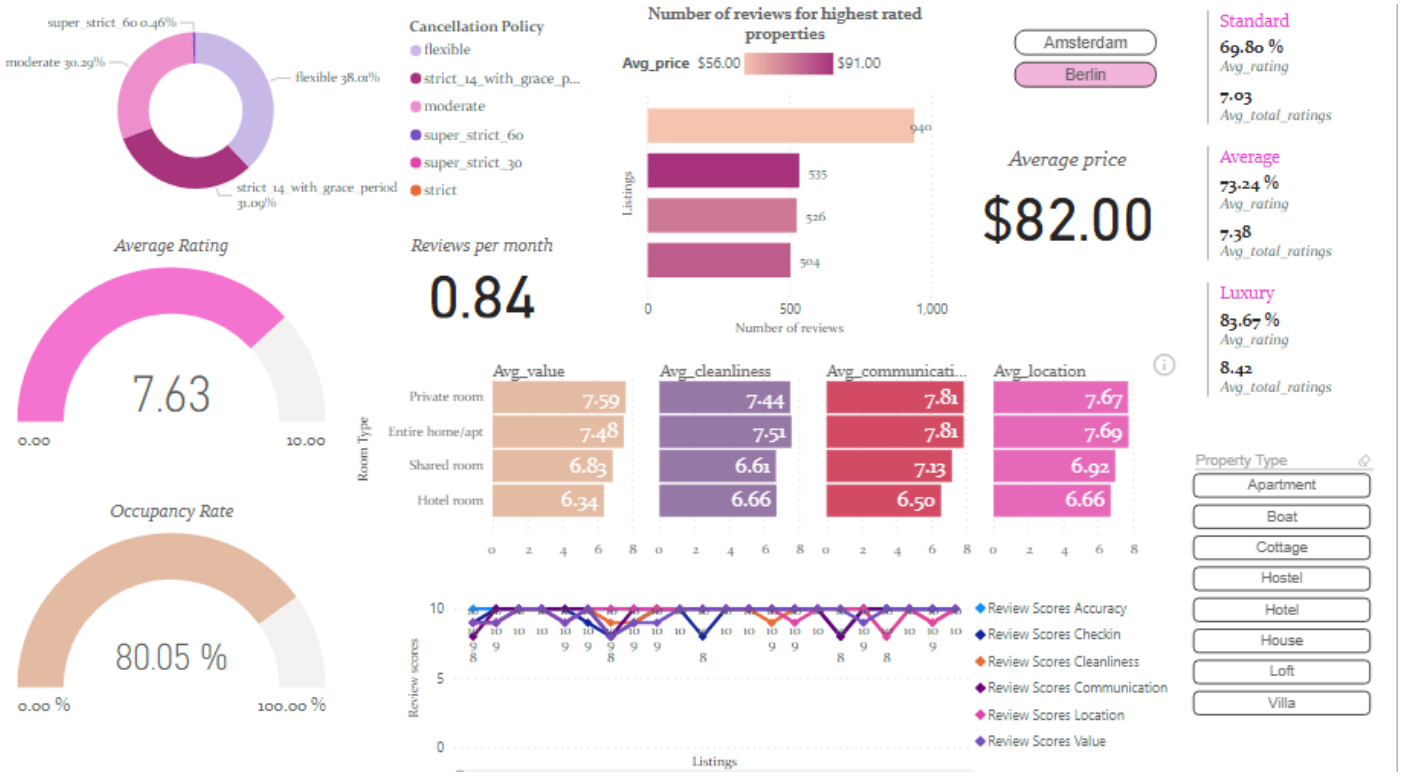
7.1.2. Top Rated Properties

Airbnb reviews play a major role in attracting guests. The more and better reviews the listings have, the more likely it is for people to make reservations and the better the reputation for the host. Therefore, it is useful to analyze them.

<u>Amsterdam:</u>



Berlin:



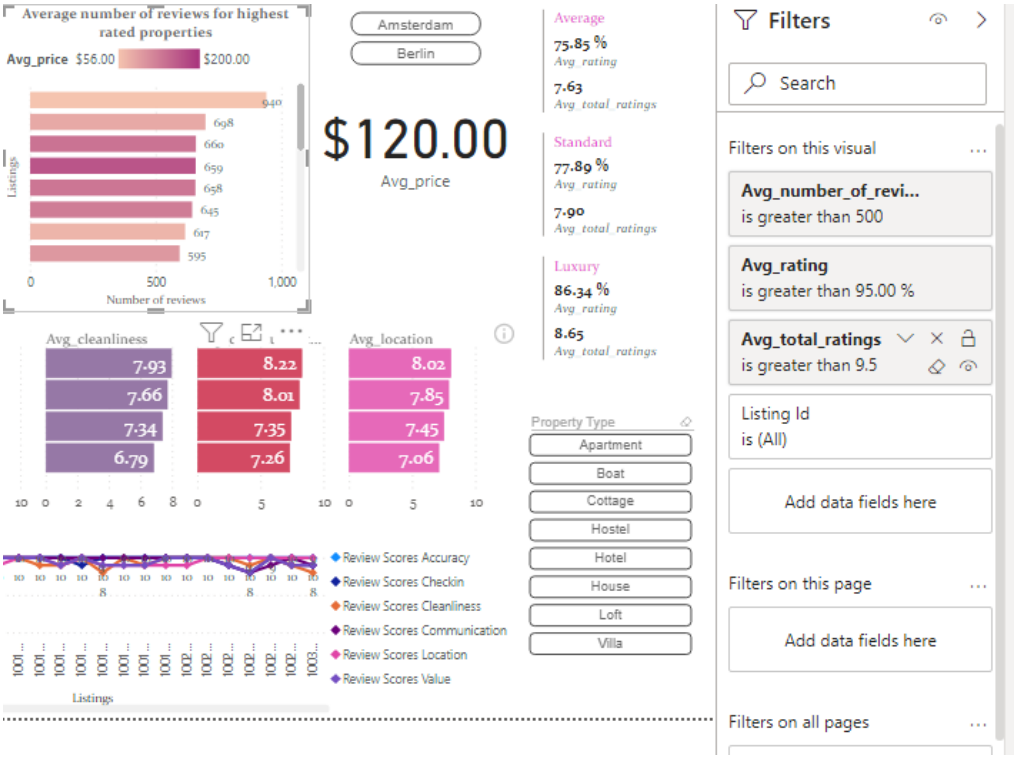
The above dashboards depict the top-rated properties (on top middle bar chart) and some general rating statistics, depending on the number of reviews and the ratings regarding specific categories for the two cities. The general rating is from 0 to 100% and the review scores ratings range from 0 to 10 and refer to overall experience, cleanliness, accuracy, value, communication, location and check-in. Moreover, we used the previously mentioned measure of average total rating which is an average of review scores cleanliness, location, accuracy and value.

The slicers used in this dashboard to examine different cases are city and property type.

The donut chart on the left top of the dashboard shows the percentage share of every cancellation policy type for the total number of listings and it is observed that the most common for both cities are flexible, moderate and strict 14-with grace-period policies. The four bar charts on the middle include the average review scores for each room type. As shown, the entire homes and private rooms have the highest scores. The bottom line chart illustrates how the review scores range for every listing. It seems that the majority of review scores are higher than 6 out of 10. There is also a multi row card that lists the average ratings for every property category, from which the top rated is luxury category, and gauge charts with the average total rating and the occupancy rate.

It can be observed that the average prices and ratings for Amsterdam are higher than those of Berlin.

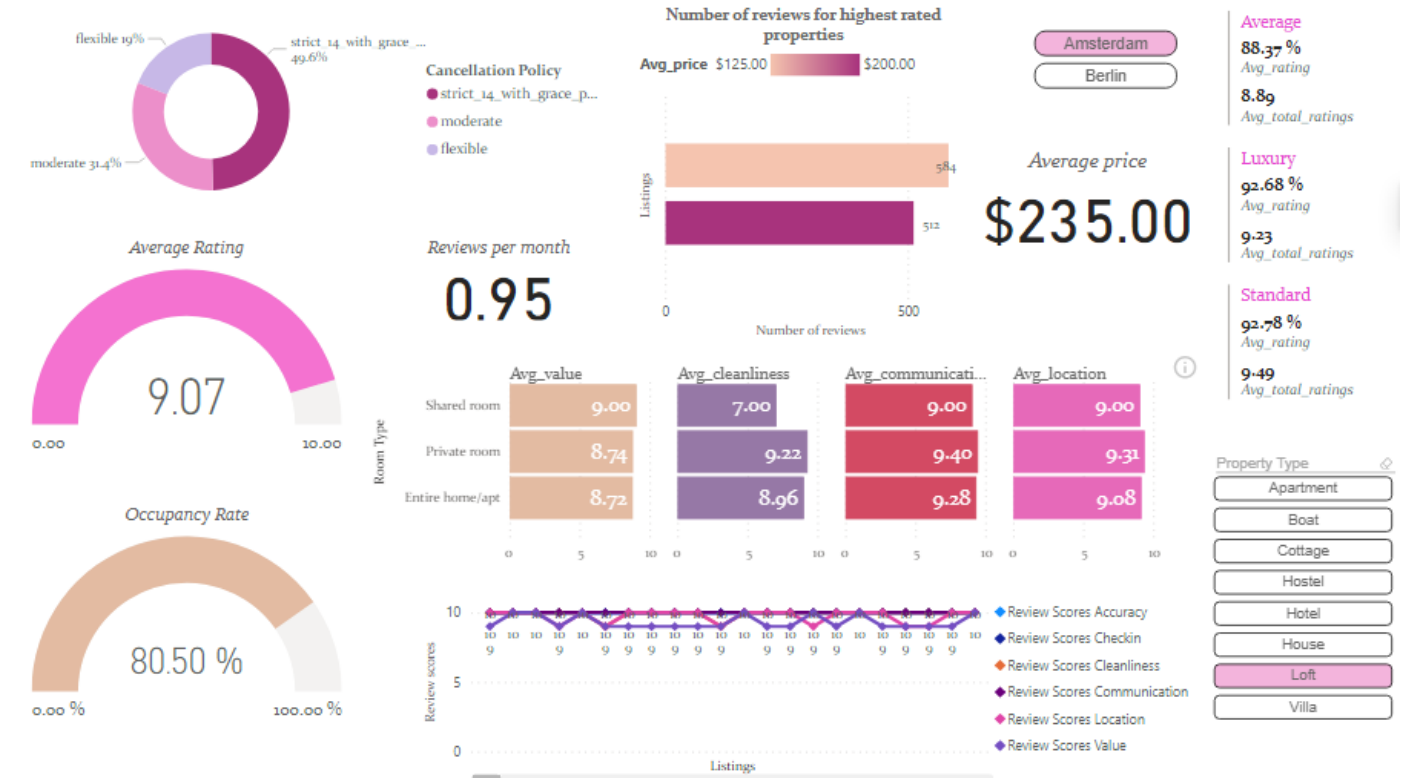
The upper middle bar chart displays the number of reviews for top rated properties and is filtered by the average ratings (greater than 95%), average total ratings (above 9.5) and number of reviews (above 500). In that way, the results are more credible and representative given that we select the properties that have a large number of reviews with high ratings. Furthermore, the gradient color of the bars represents the price, the darker the color, the higher the price. So, we get a hint of whether the ratings are affected by the prices or vice versa.



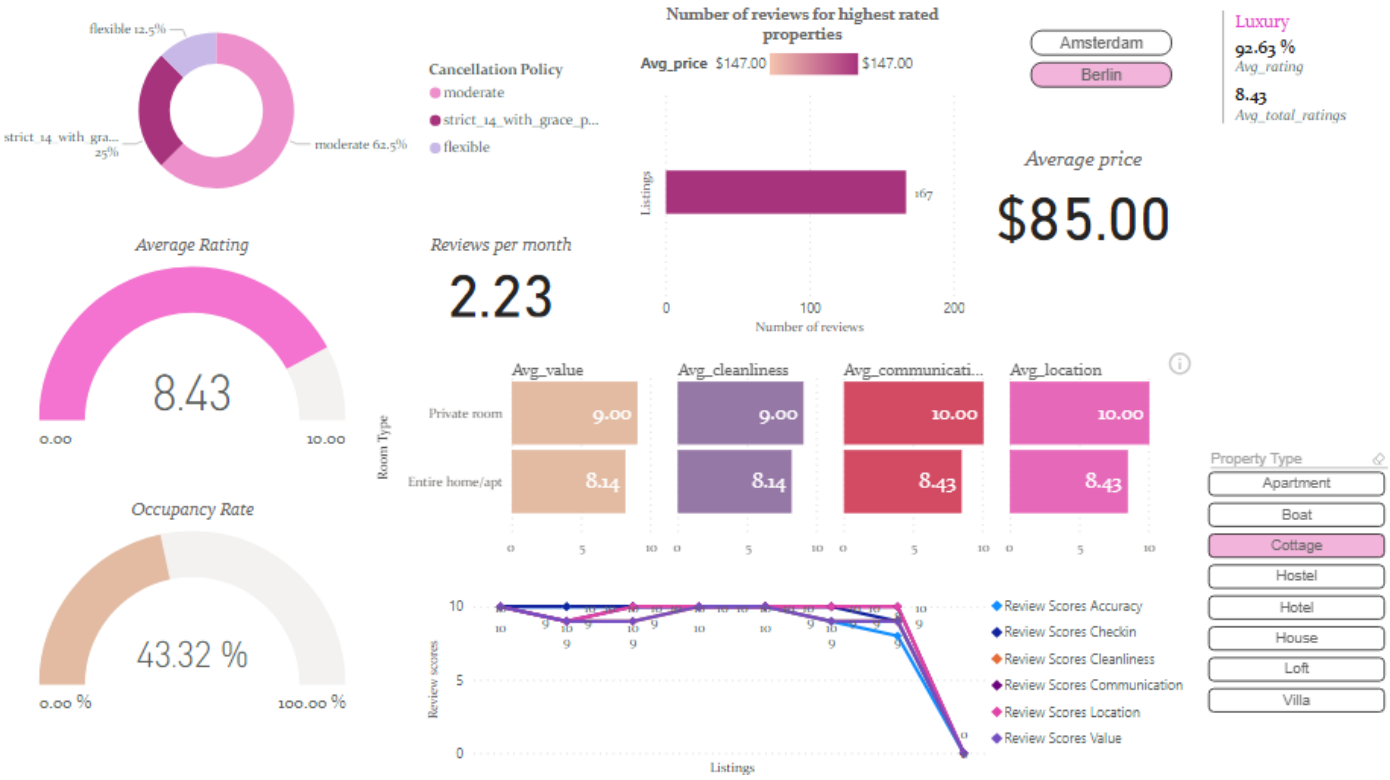
The best combination of highest number of reviews along with top average ratings/scores in Amsterdam belongs to loft property type. The price range is also high and the properties were occupied for the most days within the year. This implies that these high prices are justified by the quality of the listings.

The top rated for Berlin belong to cottage property type, but the filter for the high number of reviews was not satisfied, as it has 167 reviews.

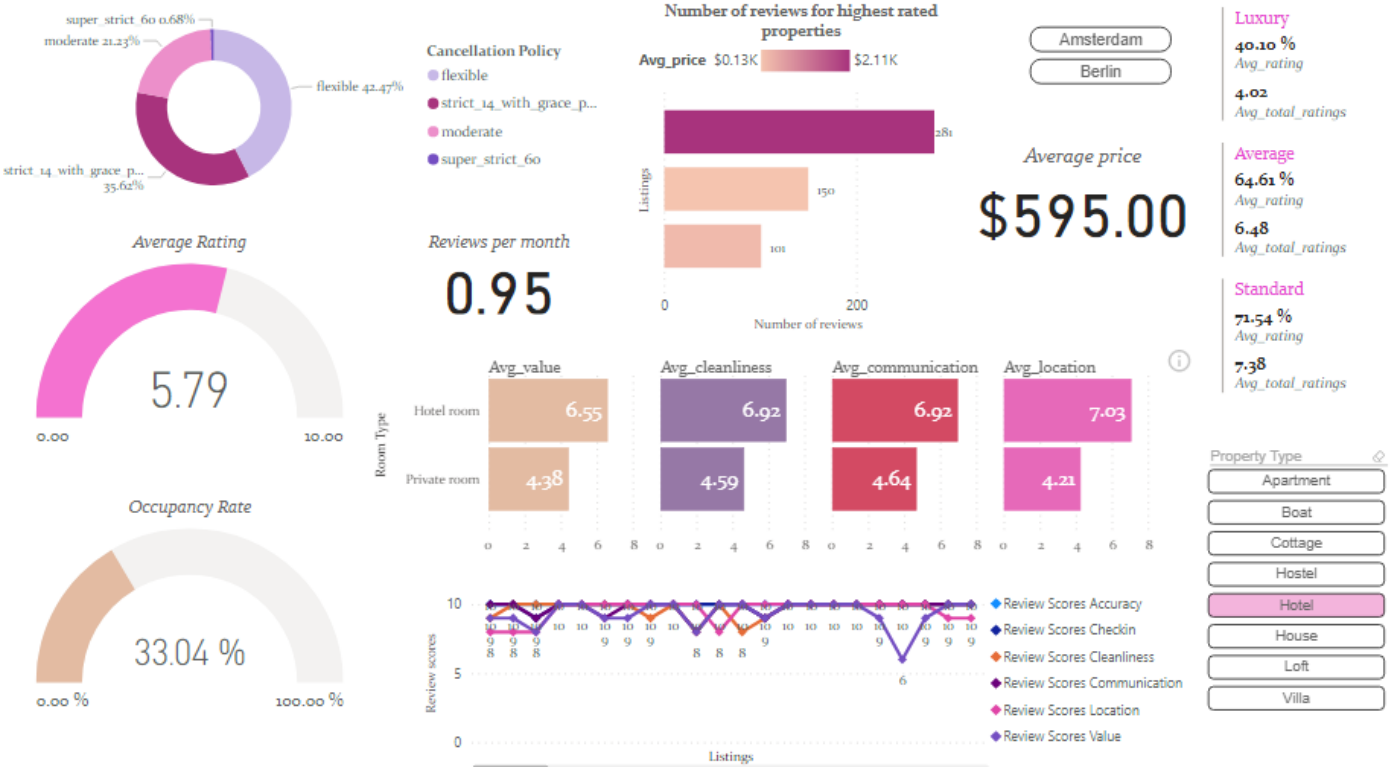
<u>Amsterdam:</u>



Berlin:



On the other hand, hotels, which have considerably higher prices have the lowest ratings among a high number of reviews. In this case, the high prices are not justified.

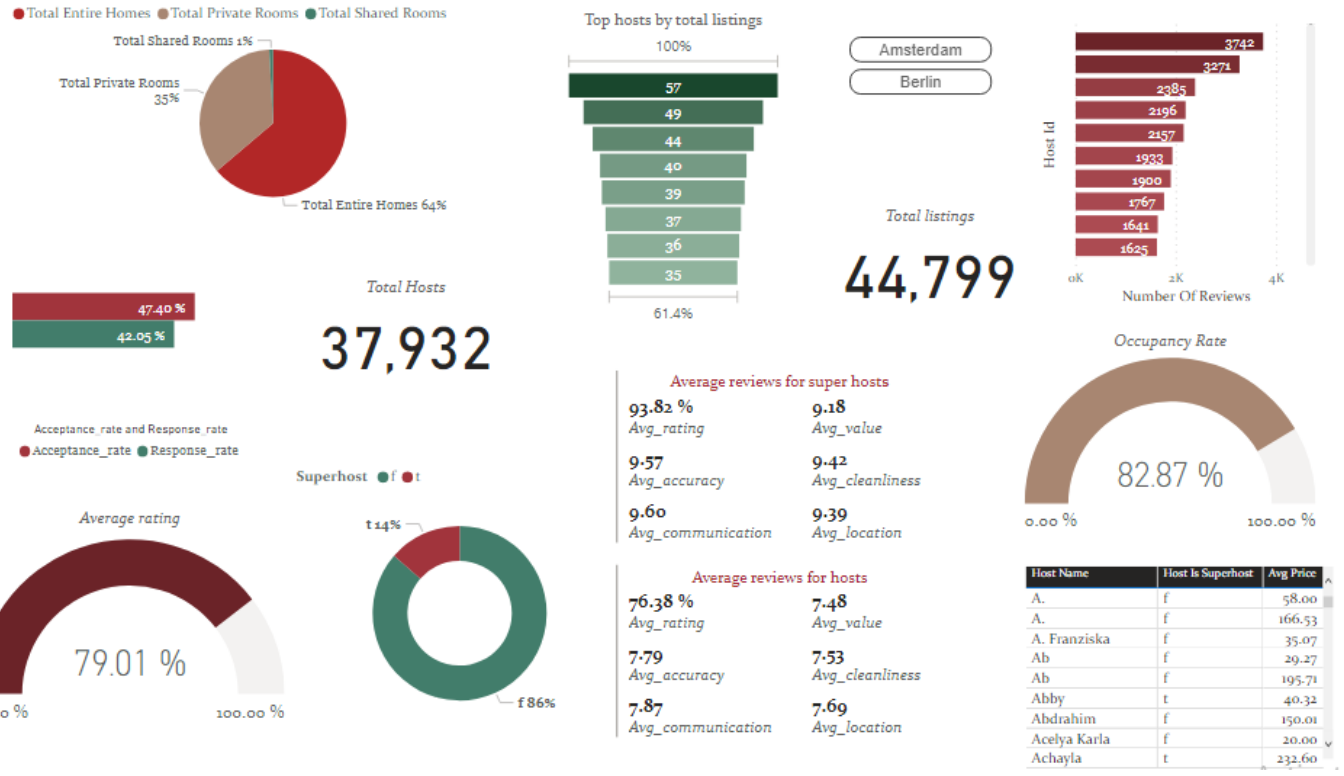


7.1.3. Top Hosts

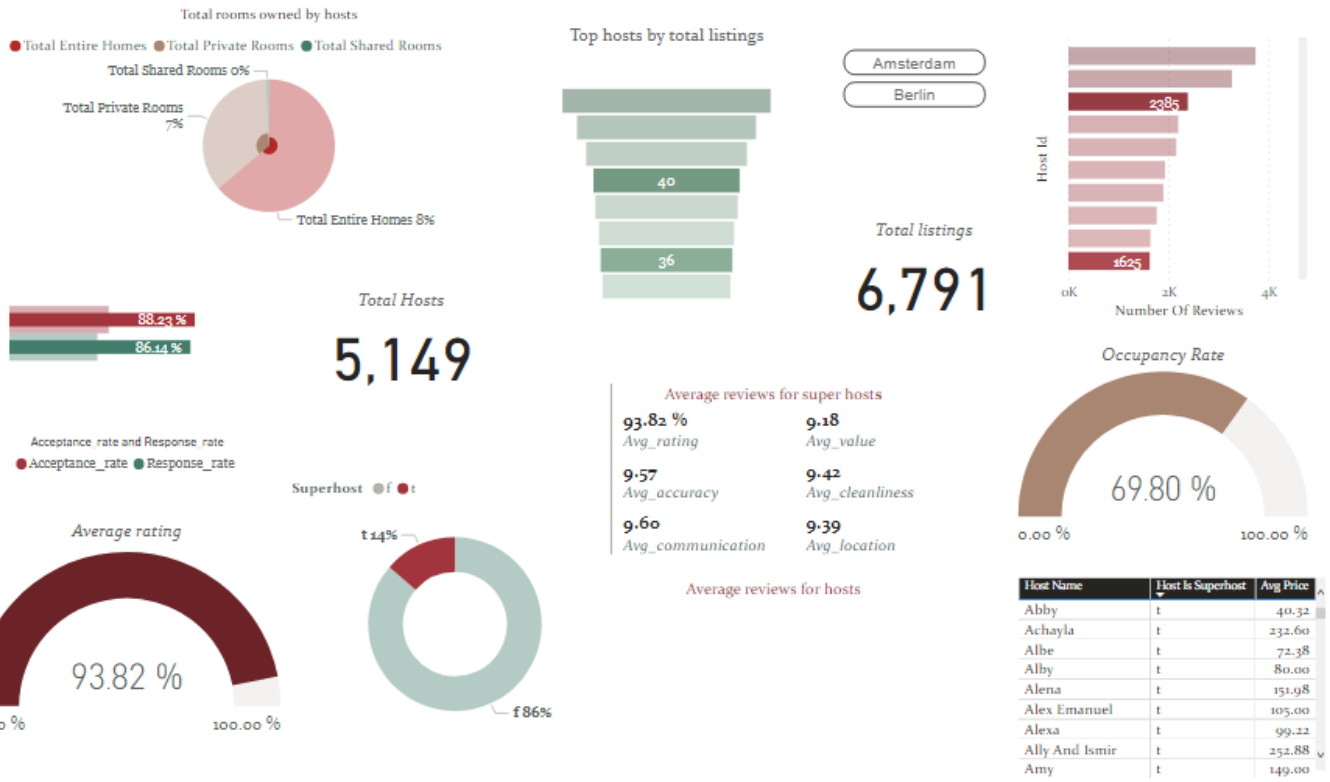
The last dashboard analyzes the top hosts. Top hosts are considered the ones with the largest amount of properties along with the highest average occupancy rate during the year.

The funnel depicts the hosts with the most properties and by combining it with the average occupancy rate during the year, which is shown in the gauge chart, we indicate the busiest hosts. The rating gauge chart also gives us a hint about the quality of their services. The pie chart illustrates the proportion of the total properties for each room type owned by hosts, which shows that the largest percentage belongs to entire homes. The bar chart below displays the respective acceptance and response rates for the hosts, while the bar chart on the top right of the dashboard shows the largest numbers of reviews for properties owned by each host.

The donut chart visualizes the percentage of superhosts among total hosts and the multi row card next to it includes some general statistics about superhosts. Finally, the table on the bottom right corner lists the name of the selected superhost, whether they are superhosts and an average price of their listings. It should be noted that Airbnb recognizes hosts as superhosts when they have a high average rating, response rate and occupancy rate.

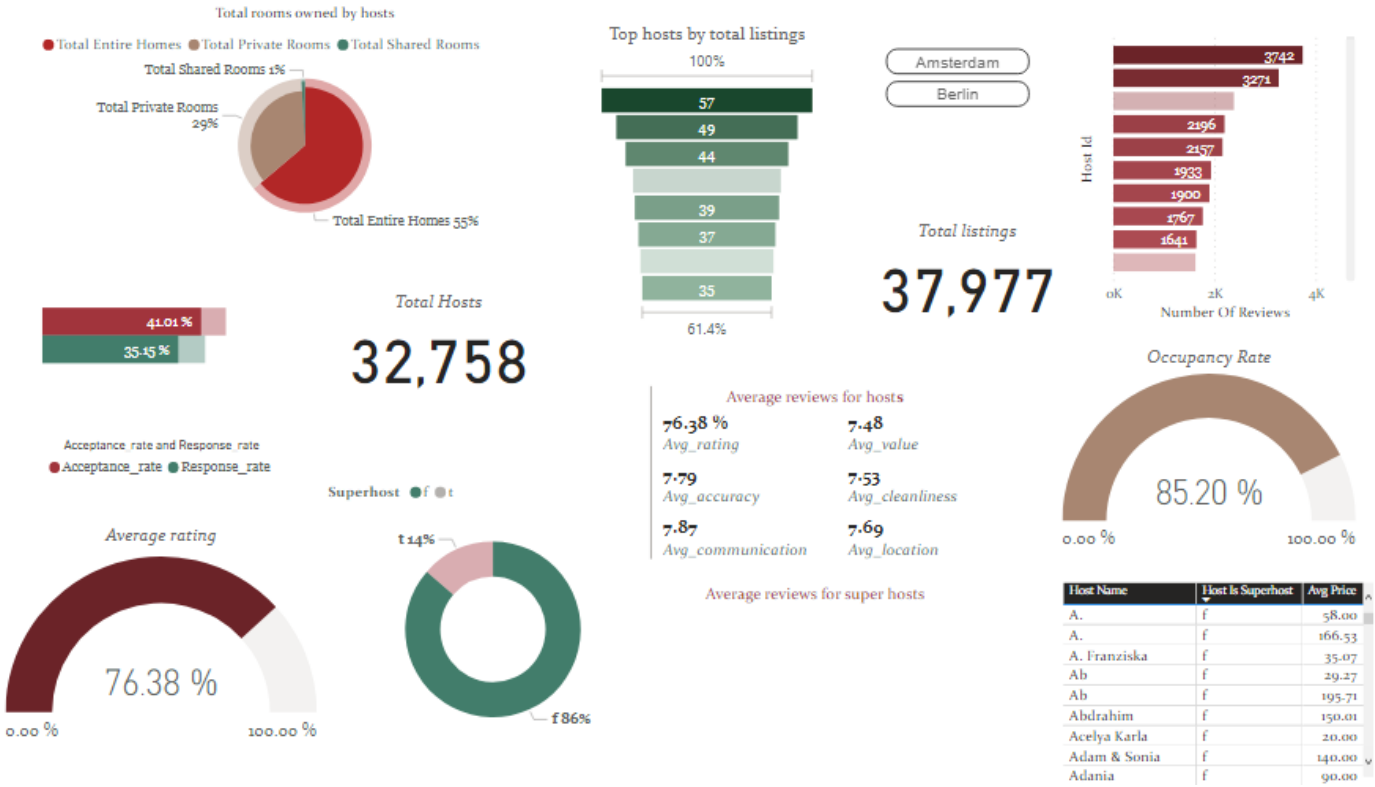


Superhosts:



Host Name	Host Is Superhost	Avg Price
Abby	t	40.32
Achayla	t	232.60
Albe	t	72.38
Alby	t	80.00
Alena	t	151.98
Alex Emanuel	t	105.00
Alexa	t	99.22
Ally And Ismir	t	252.88
Amy	t	149.00

Hosts:

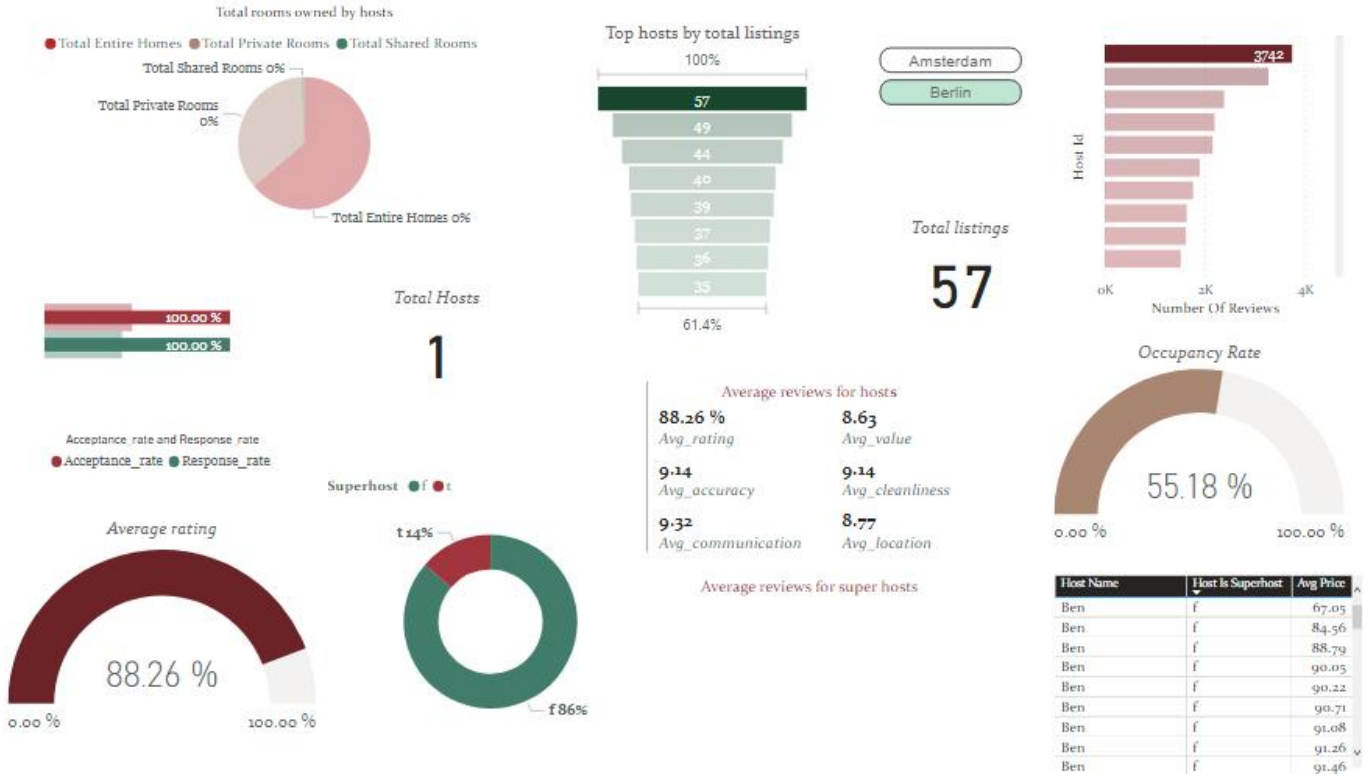


Host Name	Host Is Superhost	Avg Price
A.	f	58.00
A.	f	166.53
A. Franziska	f	35.07
Ab	f	29.27
Ab	f	195.71
Abdrahim	f	150.01
Acelya Karla	f	20.00
Adam & Sonia	f	140.00
Adania	f	90.00

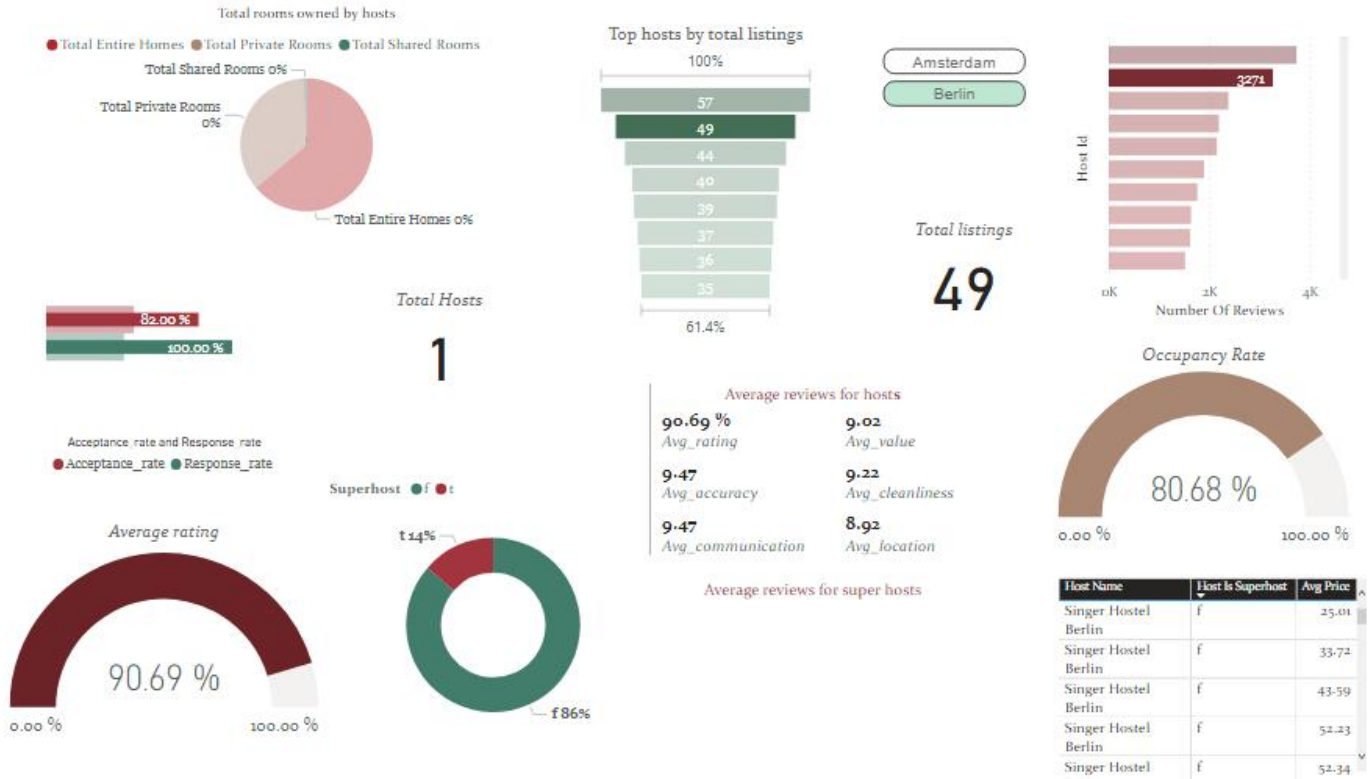
From the above graphs, it can be concluded that the superhosts have much higher ratings and hosts as well as response and acceptance rates, but they own less houses and their occupancy rates are much lower.

It is important to note that there is also a small percentage of hosts, for whom we have no information and it has been excluded from the donut chart and the multi row card as it is insignificant.

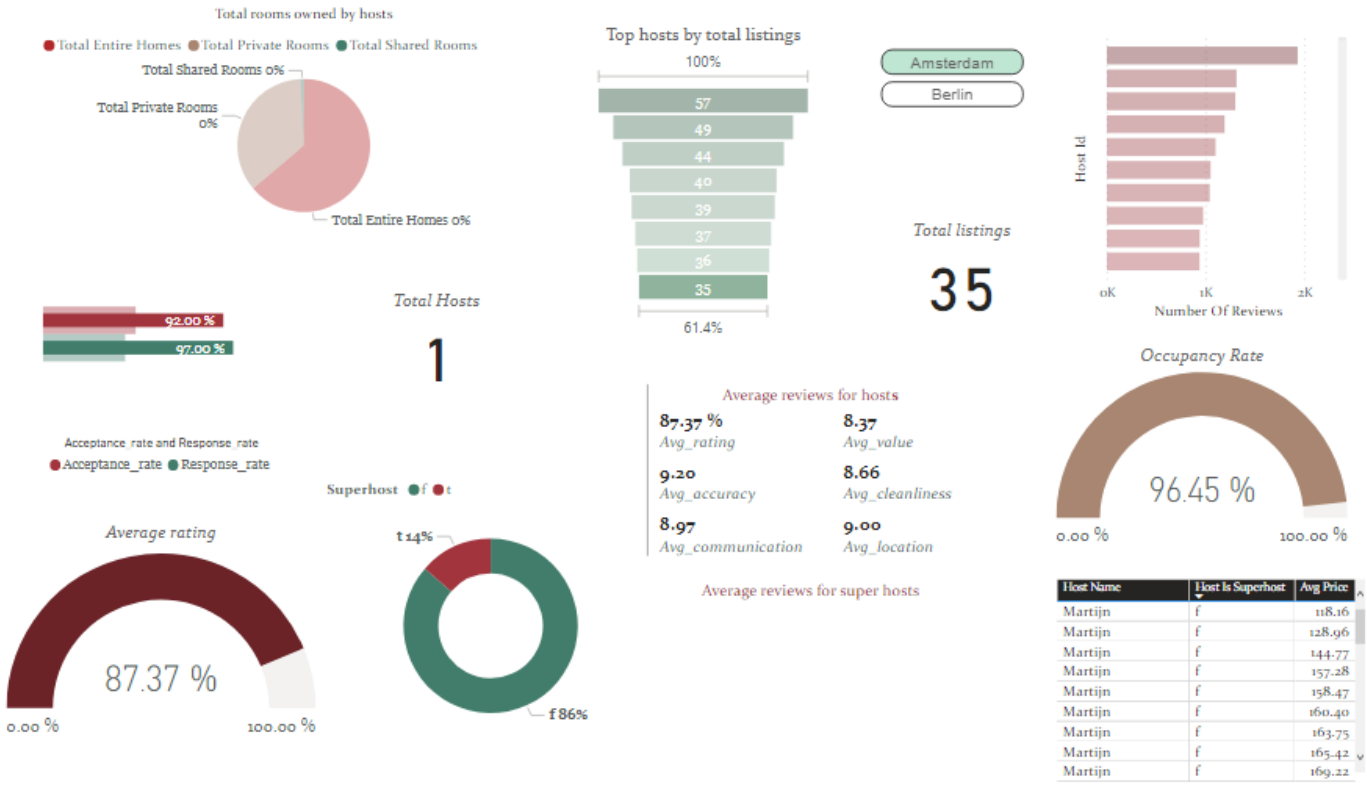
The top host in Berlin is Ben, owing 57 listings, having almost 3750 reviews, an average rating of 88%, 100% acceptance and response rates. Nevertheless, his listings' occupancy rate is 55%, which does not make him a busy host. The low occupancy rate though could be due to properties being available only at specific seasons, for example summer. He is not a superhost and the prices of his listings range from 66$ to 288$.



The next top host is from Berlin as well, their name is "Singer hostel", owning 49 listings and having 3271 reviews and very high ratings/ scores. The occupancy rate is also higher than the previous one and the prices are lower, from 25$ to 143$.



The top host from Amsterdam is Martjin with 35 total listings and generally high average ratings, although the number of reviews is not available and the prices range from 118$ to 365$.



8. Conclusions

To summarize our conclusions, by performing exploratory data analysis on the "Airbnb" dataset, we have gained various new insight about this online marketplace, we identified the top priced and top rated listings, where they are located, what characteristics they have, how the prices and ratings range, as well as the top hosts and some general statistics. From the above dashboards and graphs, the following assumptions could be stated:

- Prices in Berlin are overall lower than those in Amsterdam
- The entire homes and apartments dominate the marketplace, as they are the majority.
- Hotel rooms are the priciest among room types, but they are not accompanied by high ratings.
- Shared rooms and hotel rooms have the lowest review scores and ratings.
- Luxury property category has generally higher prices than the others due to the services offered.
- Lofts are also higher priced than apartments.
- Although people are willing to spend extra money for higher quality services, high prices do not always correspond to high ratings and quality.
- Low prices on the other hand are mostly accompanied with average ratings.
- The distribution of price over time shows that the summer months and weekends have generally high prices.
- The more bedrooms and bathrooms a property has, the higher the price, but from a certain number and above, the price falls.
- Superhosts have higher ratings, scores per category and response/acceptance rates, therefore their title is justified and are more trusted within the community. Nevertheless, they are not considered very busy.
- Hosts with most listings are not usually superhosts.
- Most top hosts are located in Berlin.

In general, a data analysis project could be proven to be very beneficial for the company. It is expected to yield several benefits, such as:

- ✓ Improvement of support services, business processes, staff productivity, or efficiency as a whole.
- ✓ Reduction/minimization of costs.
- ✓ Increase in attraction and generation of higher revenue.
- ✓ Increase supply (and variety) of short-term rentals available to travelers.

9. References

- www.Kaggle.com
- Modern Data Management & Business Intelligence slides
- Multidimensional Databases & Data Warehousing, by Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen.