Data Analytics

Exploring Paris Airbnb Market

Felicia Onwudinjo

June, 2022

Table of content

INTRODUCTION

Business Use Case : In this project, we applied skills such as data importing and cleaning skills to uncover insights about the Airbnb market here in Paris.

We imported data from multiple file types and combine them to answer questions about the Airbnb market in Paris. We used string cleaning and date manipulation skills to extract accurate information from the datasets. The packages and tools used here are utilized by data scientists every day since so much of the world's data is stored in unconventional formats and is not clean or analysis-ready.

Goals : Our goals was to convert untidy data into appropriate formats to analyse, and answer key questions including:

- What is the average price, per night, of an Airbnb listing in Paris?
- How does the average price of an Airbnb listing, per month, compare to the private rental market?
- How many adverts are for private rooms?
- How do Airbnb listing prices compare across the other Parisian suburbs?

DATASETS AND DATA SOURCES
MOST OF OUR DATASET WAS GOTTEN FROM KAGGLE.

Kaggle is a web-based platform that hosts the world's largest data science community, with over 536,000 active participants from 194 countries and approximately 150,000 monthly submissions, as well as sophisticated tools and resources to help achieve all data science achievements. Kaggle gets its data mainly by web scraping and example will be Paris Airbnb which has been scrapped from Airbnb's webpage and also the community has shared a lot of datasets . You'll find everything you need to finish your data science projects inside Kaggle, including access to more than 50,000 public datasets and 400,000 public notebooks.

DATASETS

The datasets consists of information about Airbnb listings in Paris city.

Since 2008, guests and hosts have used Airbnb to travel in a more unique, personalized way. As part of the Airbnb Inside initiative, this dataset describes the listing activity of homestays in Paris.

An initial part of the examination of the data is performed. The data comprises 59881 rows and 96 columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59881 entries, 0 to 59880
Data columns (total 96 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              59881 non-null  int64
 1   listing_url                     59881 non-null  object
 2   scrape_id                       59881 non-null  int64
 3   last_scraped                    59881 non-null  object
 4   name                            59807 non-null  object
 5   summary                         57524 non-null  object
 6   space                           40516 non-null  object
 7   description                     59384 non-null  object
 8   experiences_offered             59881 non-null  object
 9   neighborhood_overview           35647 non-null  object
 10  notes                           18606 non-null  object
```

Content

The following Airbnb activity is included in this Paris dataset:

- Listings, including full descriptions and average review score
- Reviews, including unique id for each reviewer and detailed comments
- Calendar, including listing id and the price and availability for that day

Visualization

Before cleaning our dataset, we did a map visualization of our using coordinates from the longitude and latitude columns to see the concentration of Airbnb listings across Paris.

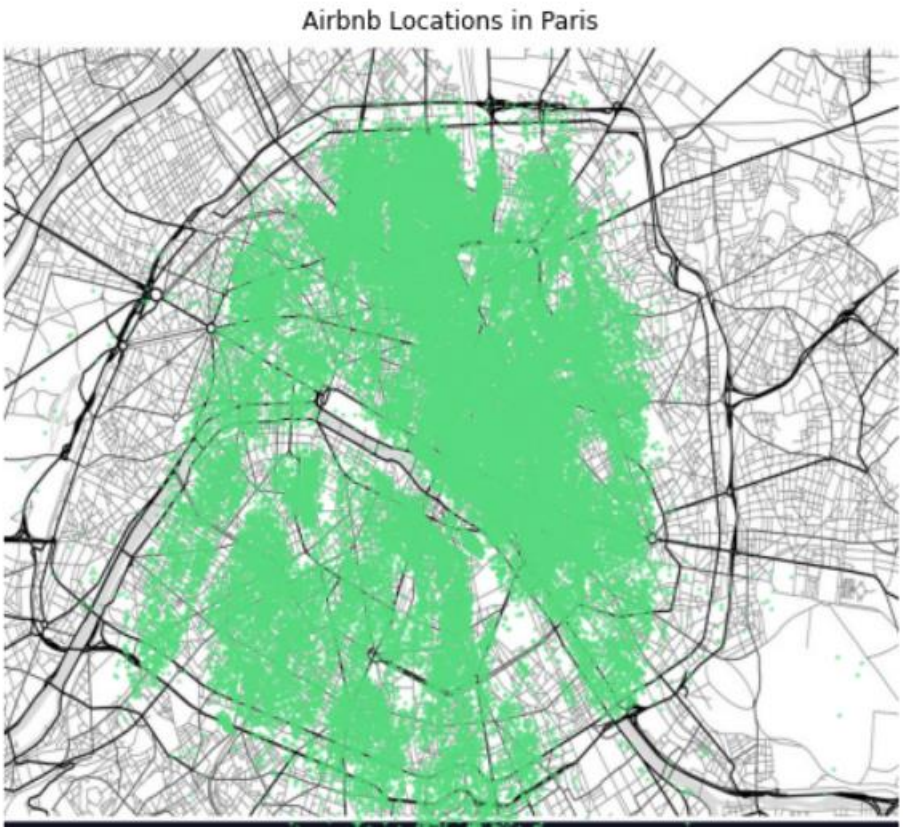


Figure 1: Map of Airbnb listings

From this visuals , we can see there are more listings in Paris center which is Arrondisement 1,2,3 and 4.

Data cleaning and Exploratory data analysis

The datasets is very disorganized, with thousands of unknown values. We made some reasonable assumptions and work from there. Our datasets had 96 columns in which most where not insignificant. So we picked some columns which were significant to our Project as follows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59881 entries, 0 to 59880
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   59881 non-null  int64
 1   room_type            59881 non-null  object
 2   minimum_nights       59881 non-null  int64
 3   number_of_reviews    59881 non-null  int64
 4   host_listings_count  59873 non-null  float64
 5   availability_30      59881 non-null  int64
 6   availability_60      59881 non-null  int64
 7   availability_90      59881 non-null  int64
 8   availability_365     59881 non-null  int64
 9   neighbourhood        57231 non-null  object
dtypes: float64(1), int64(7), object(2)
memory usage: 4.6+ MB
```

Unnecessary data

Most of the columns from our "data all datasets" where considered unnecessary due to the following:

- Uninformative/repetitive data: Some of the columns contained repetitive data and were uninformative, it did not fit the purpose of our project.
- Irrelevant: Some features were not related to the question we are trying to solve in the project, they are irrelevant. So we dd not include them in our new columns, example: scrapping id column.
- Duplicates: We first remove the unique identifier *id* in the dataset. Then checked to find out the number of duplicated rows.

OUTLIERS

We use boxplots and histogram to determine outliers. In descriptive statistics, a box plot or boxplot is a method for graphically depicting groups of numerical data through their quartiles. We first selected columns with numeric dtypes and plotted a boxplot for each of the columns, Lets look at the column minimum nights.

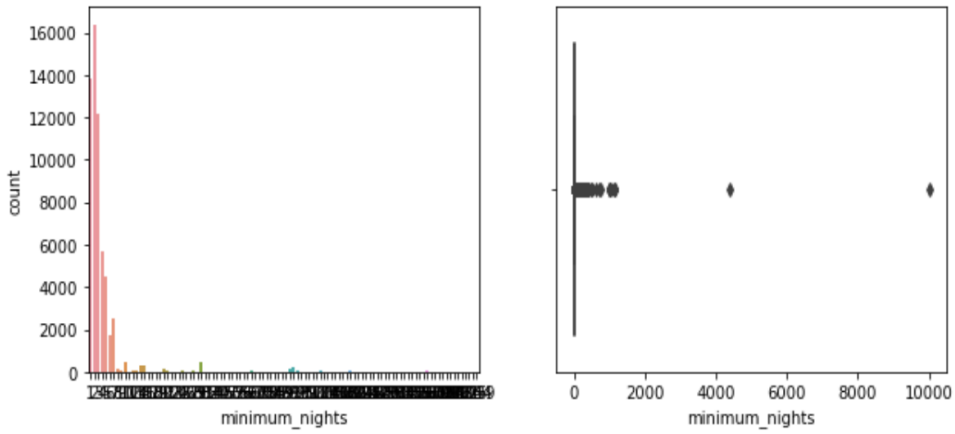
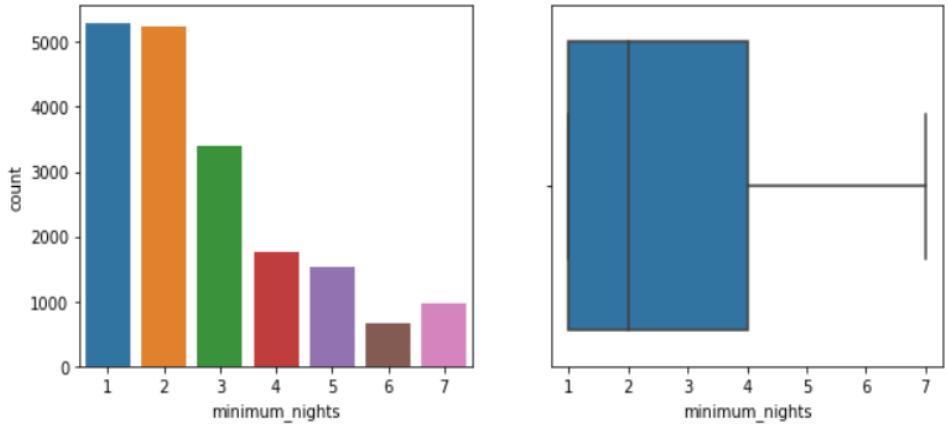


Figure 2. Before cleaning

Here, we can see that there are a lot of outliers, it's not feasible to have minimum nights of 10,000. We clean this by using the standard quantile for taking out outliers.



Figrure 3: After taking out outliers

We can see from the above figure that we can make some analysis clearly, for instance: There are more listings with one and two minimum nights. Same is done for the other numerical columns such as availability_30, number of reviews .

CHANGING DATATYPES

Looking at the price table, the columns “price” , “weekly price”, and “monthly price” has different data types including strings and float which are objects.

```
id                 int64
price             object
weekly_price      object
monthly_price     object
dtype: object
```

We needed to perform some calculations, so we changed the datatype with the following codes:

```
def clean_currency(x):
    """ If the value is a string, then remove currency symbol and delimiters
    otherwise, the value is numeric and can be converted
    """
    if isinstance(x, str):
        return(x.replace('$', '').replace(',', ''))
    return(x)
```

These codes also changed our data to become numerical and we have the following:

```
id                   int64
price              float64
weekly_price       float64
monthly_price      float64
price_type          object
dtype: object
```

MISSING DATA / NULL DATA

Still on the price table we have lots of missing data on both weekly price and monthly price columns:

```
prices.isna().sum()
```

```
id                  0
price               0
weekly_price    47739
monthly_price   52094
dtype: int64
```

We fill the data by multiplying the price by night 7 and replacing the result to the 'nan' null values. We used the same approach for the monthly price column by multiplying the price column by (365/12) for each null row.

```
id              0
price           0
weekly_price    0
monthly_price   0
dtype: int64
```

Now we have no null values

CLEANING ALPHABETS CHARACTERS

Looking at the 'neighborhood' column of the data table, we realized that names of the neighborhoods were written in French alphabets.

```
0                       République
1                           Alésia
2    Saint-Paul - Ile Saint-Louis
3                        Le Marais
4    Gare du Nord - Gare de I'Est
```

We used a lambda functions and encoders to change the type of characters to English characters as below.

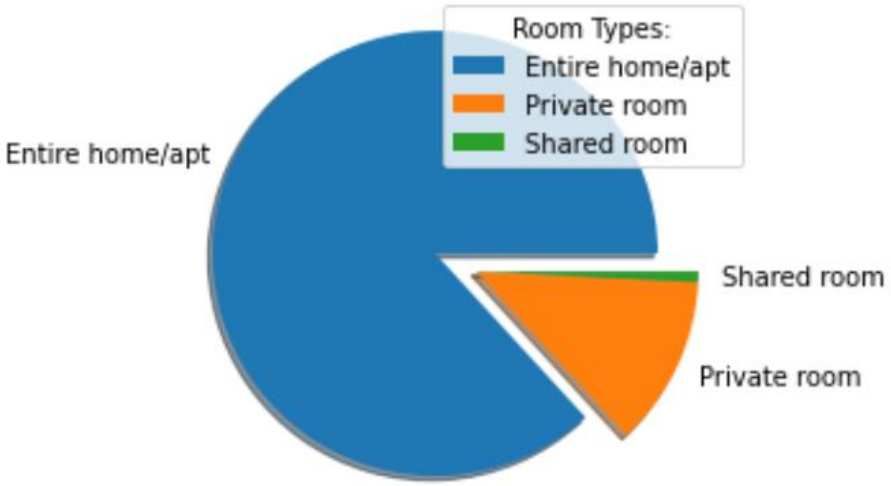
```
0                       Republique
1                           Alesia
2    Saint-Paul - Ile Saint-Louis
3                       Le Marais
4    Gare du Nord - Gare de I'Est
```

This brings us to the end of our data cleaning.

Data Exploration

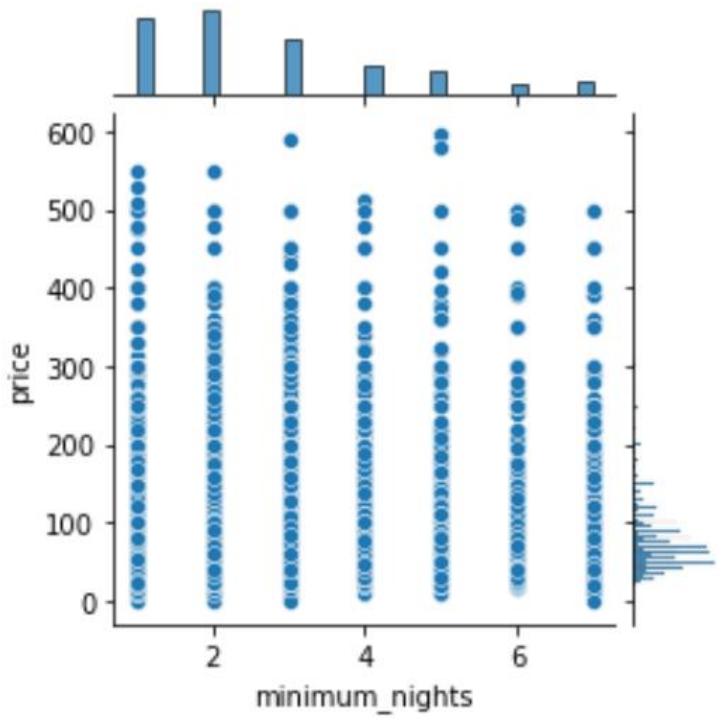
We do some basic visualizations to get some data insights.

1. We would like to know the type of room with the highest number of listing



The above shows pie that show that there are more listings for Entire home/apt than Private room, and private room.

2. We would like to get the relationship between minimum nights and price



It seems that extreme high priced listings tend to ask for a smaller minimum nights of stay

DATABASES STORING

What is a relational database and how does it work?

There are several types of databases, such as the Relational Database and the NoSQL.

In this project we will use the Relational Database because of the following reasons:

- It is SQL based and takes CSV files can be connected by keys
- We can use SQL like languages to query files
- The allow the indexing of columns for faster keys
- It supports the join operations.

As compared to NoSQL which is not structured, Relational database is structured into row and columns.

A relational database's data tables store information about connected objects. Each column includes the data properties, whereas each row holds a record with a unique identifier (known as a key). Each feature is assigned a value in each record, making it simple to establish links between data points.

TABLE CREATION AND INSERTION

We created a database in MySQL called Airbnb and also created tables with identifiers such as the primary and the foreign keys:

```
create database if not exists Airbnb;
use Airbnb;

CREATE TABLE IF NOT EXISTS hosts(
    host_id INTEGER NOT NULL PRIMARY KEY,
    host_name VARCHAR(50) NOT NULL,
    host_since VARCHAR(50) NOT NULL,
    host_location VARCHAR(50) NOT NULL
);
CREATE TABLE IF NOT EXISTS price (
    id INTEGER NOT NULL PRIMARY KEY,
    price float NOT NULL,
    weekly_price float NOT NULL,
    monthly_price float NOT NULL
    );

CREATE table if not exists property(
id INTEGER NOT NULL PRIMARY KEY,
room_type VARCHAR(255) NOT NULL,
minimum_nights INTEGER NOT NULL,
number of reviews INTEGER NOT NULL,
```

We used MySQL Alchemy, which is a library in python to directly our data from python into the created tables on MySQL. We imported the following libraries:

```
import mysql.connector as sql
from sqlalchemy import create_engine
import sqlalchemy as db
import pandas as pd
engine = db.create_engine('mysql://root:Leecia2864@localhost:3306/Airbnb')
```

The ‘engine’ function served as the connector between mySQL data base and python.

What is SQLAlchemy used for ?

SQLAlchemy is a library that facilitates the communication between Python programs and databases . Most of the times, this library is used as an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements.

There are also other ways to insert tables to SQL by:

- Manually inserting row and columns names using the INSERT function in MySQL. This is best for updating data bases
- Using the Import wizard from MySQL to import directly CSV files.

META DATA

We will start by giving an in dept description of our column names and their data types.

PROPERTIES TABLE

	ELEMENT	DATATYPE	PURPOSE
1	ID	INT	Unique identity for each listings or properties
2	HOST_ID	INT	Unique identity for the host of a property
3	PRICE_ID	INT	The unique identity price of each property
4	ROOM_TYPE	VARCHAR	The type of a property listing
5	MINIMUM_NIGHT	INTEGER	The least number of nights of a stay in a property
6	NUMBER OF REVIEWS	INTEGER	Number of reviews of each listing
7	HOST LISTING COUNT	INTEGER	Number of property listed for each host
8	AVAILABILITY_30	INTEGER	Availability of a property in within 30 days

9	AVAILABILITY_60	INTEGER	Availability of a property in within 60 days
10	AVAILABILITY_90	INTEGER	Availability of a property in within 3 months
11	AVAILABILITY_365	INTEGER	Availability of a property in within 30 days
12	NEIGHBOURHOOD	VARCHAR	Availability of a property in within one year

Here **VARCHAR**- variable character and **INT** – Integer

PRICE TABLE

ELEMENT	DATATYPE	PURPOSE
PRICE ID	C	Unique identifier
PRICE	INT	Price of each listing per night
WEEKLY PRICE	INT	Price of each listing per month
MONTHLY PRICE	INT	Price of each listing per year

REVIEWS TABLE

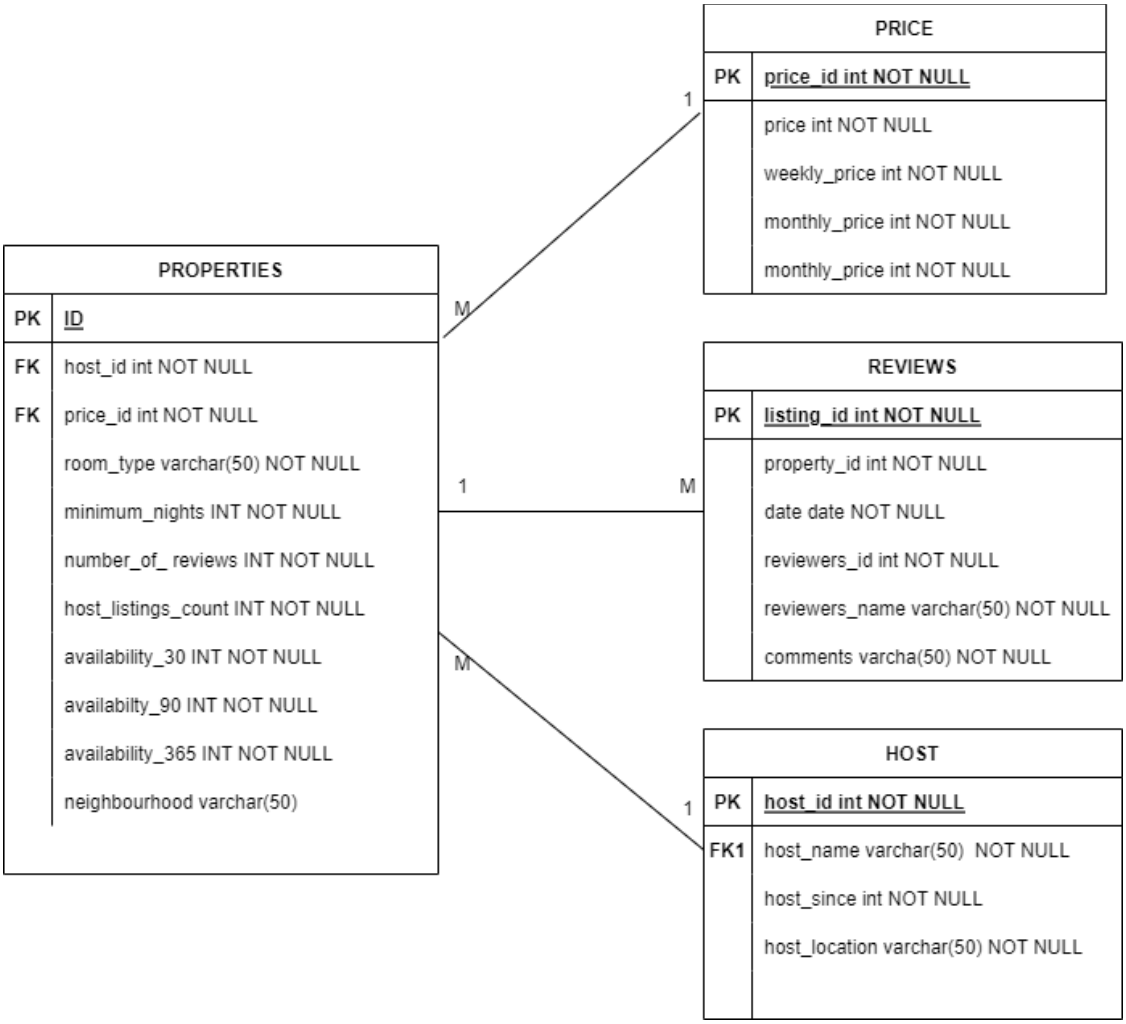
ELEMENT	DATATYPE	PURPOSE
LISTING_ID	INT	Unique id for listing property
PROPERTY_ID	INT	Unique id from from property id
DATE	DATE TYPE	The date of listing
REVIEWERS_ID	INT	Unique id of reviewers
REVIEWERS_NAME	VARCHAR	Name of reviewers
COMMENTS	VARCHAR	Comments of reviewers

HOST TABLE

ELEMENT	DATATYPE	PURPOSE
HOST ID	INT	Unique identity of host
HOST NAME	VARCHAR	Name of host
HOST SINCE	VARCHAR	How long of since being a host
HOST LOCATION	VARCHAR	Location of the host

UML DIAGRAM

Here is the UML diagram shows relationships between entities



SQL QUERIES

We have to make some queries from our cleaned dataset and answer some business problems:

1. What is the average price, per night, of an Airbnb listing in NYC?

From our queries, we see that the average price of an Airbnb in Paris per night cost 111 euros.

	AVG_Price
▶	111

2. How does the average price of an Airbnb listing, per month, compare to the private rental market?

	Average_price_per_month
▶	3243.81

Now we know how much a property costs, on average, per night, but it would be useful to have a benchmark for comparison. According to Numbeo.com , a 1 bedroom apartment in Paris costs, on average, 1200 euros per month. Let's convert the per night prices of our listings into monthly costs, so we can compare to the private market.

The average monthly price of an Airbnb listing is 3243 euros as compared to the normal rental market is quite high.

3. What is the number of listing per room: From our analysis, there are more listings for Entire home and apartment as compared to private rooms and shared rooms

room_type	COUNT(room_type)
Entire home/apt	19552
Private room	2290
Shared room	138

4. Next, we would like to know the average price of the most expensive Airbnb properties in Paris.

neighbourhood	AVG_PRICE
Louvre - Tuileries	161.69767441860466
XIV Arrondissement	145
Madeleine - Vendome	122.1025641025641
Champs-Elysees	120.03286384976526
Chatelet - Les Halles - Beaubourg	116.80081300813008

From our query, we have seen that the most expensive neighbourhood of Airbnb listings are the Louvre, XIV Arrondissement, Madeleine, Champs-Elysees, and Chatelet.

5. Lastly we would like to know the name of the host with the highest number of reviews and the location of the property

HOST	NEIGHBOURHOOD	NUMBER_OF_REVIEWS
Tracy	Bastille	174

Here we can see that Tracy is the Host who hast the most reviews which is located in Bastille.

CONCLUSIONS

From these project, we have seen several ways of cleaning data, by

- Checking for outliers
- Checking for unique values to avoid duplicates
- Changing datatypes, fixing alphabetical characters and lots more.
- We have also giving some graphical visualisations of the map of Airbnb Property distribution across Paris.
- We have shown why we chose the relational database and its advantages
- Showed how we created tables and imported our csv files into MySQL
- Lastly , We did some queries and gave some data driven insights based on our cleaned dataset.

Due to time constraint, we could have done a lot more visualisations to answer a lot more questions such as:

- check the distribution of review numbers
- check how number of reviews variate along with different availability groups

and lots more.