DSA5104 Project by Group 28 (Dice 4) Database System for Airbnb

1. Dataset Overview

Inside Airbnb is a vast dataset capturing detailed information on listings—such as photos, locations, amenities, booking status, and guest feedback—from cities globally. This dataset is characterized by complex data relationships and a diverse range of data types. A database built upon this dataset can aid in recommending suitable listings to guests, assisting hosts in pricing, and bolstering platform marketing strategies. By conducting analysis, we can extract insights into hot travel trends and consumer behaviors. Such insights are instrumental in enhancing the user experience, which in turn can lead to increased bookings and revenue. Our research is primarily concentrated on three major hubs: Singapore, Hong Kong, and New York.

In the Inside Airbnb dataset, data for each city is organized into four distinct tables. The 'Listing' table offers detailed information about the listings, including basic descriptions, host information, and historical ratings. In total, there are over 40,000 listings in these three cities. The 'Calendar' table presents daily specifics for each listing, including availability and pricing. It records the availability of listings from September 2023 through September 2024. The 'Review' table compiles user ratings for listings they have stayed in, providing insights into guest experiences. There is a total of over 80,000 comment data. Lastly, the ‘Neighborhood’ table is a collection of areas in each city, and each listing is located in one of the areas.

2. Data Cleaning and Preprocess

2.1. Host

After we retrieved the Listing table from Airbnb, we observed that it consists of detailed host information, hence, we decided to separate these attributes into another Host table. The attributes of the Host table are:

```
host_combined.columns
```

```
Index(['host_id', 'host_name', 'host_since', 'host_location', 'host_about',
       'host_response_time', 'host_response_rate', 'host_acceptance_rate',
       'host_is_superhost', 'host_neighbourhood', 'host_listings_count',
       'host_verifications', 'host_identity_verified', 'city'],
      dtype='object')
```

We observed that for the *host_verifications*, it consists of a list of verification methods:

```
host_final_df[["host_verifications"]].value_counts()
```

```
host_verifications
['email', 'phone']                    20966
['email', 'phone', 'work_email']       2866
['phone']                              2441
['phone', 'work_email']                  95
['email']                                60
[]                                       27
['email', 'work_email']                   5
Name: count, dtype: int64
```

We decided to separate this information into boolean attributes that are more suitable for querying. Instead of the column *host_verifications*, we will construct three boolean columns *email, phone* and *work_email*, with 1 indicating that the host has this verification method and 0 otherwise. Here is a overview of our processed columns:

```
host_final_df[["email", "phone", "work_email"]]
```

	email	phone	work_email
0	1	1	0
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
...	...	...	...
26460	1	1	0
26461	1	1	0
26462	1	1	0
26463	1	1	0
26464	1	1	0

26465 rows × 3 columns

2.2. Listings

The remaining Listings table contains 62 attributes, with some of them being less significant such as “picture_url”, “scrape_id”, etc. We decided to firstly drop those irrelevant attributes.

After this step, for the remaining Listing attributes, we standardised some of them so that they are more suitable for querying. Firstly, we removed the “%” sign for *host_response_rate* and *host_acceptance_rate* and converted it from string type to a decimal between 0 and 1. This thus allows mathematical comparison during the filtering process.

Moreover, for the attribute *bathrooms_text*, it is originally shown as a string below:

```
hk_listing_raw.bathrooms_text.value_counts()
```

```
bathrooms_text
1 bath                3012
1 private bath        1322
1 shared bath         1231
2 baths                361
1.5 baths              233
2 shared baths         194
1.5 shared baths        78
3 baths                 47
```

We standardised it to a decimal value, for example, 1 represents 1 bath and 2.5 represents 2.5 baths. This thus allows mathematical comparison during the query process.

Besides this, we also observed that the *amenities* attribute consists of an array of amenities available for that listing:

```
listing_combined[["amenities"]]
```

	amenities
0	["TV", "AC - split type ductless system", "Fir...
1	["Patio or balcony", "Fire extinguisher", "Ref...
2	["Smoke alarm", "AC - split type ductless syst...
3	["Wifi", "Kitchen", "Air conditioning", "TV wi...
4	["Lockbox", "Essentials", "Self check-in", "Ha...
...	...
3478	["Shampoo", "Fire extinguisher", "AC - split t...
3479	["Security cameras on property", "TV", "Dedica...
3480	["Security cameras on property", "Smoke alarm"...
3481	["Self check-in", "Bed linens", "Hot water ket...
3482	["Fire extinguisher", "Bed linens", "Hot water...

Instead of putting them into an array, we decided to choose the top 10 most frequent amenities and expand them into 10 boolean columns, with 1 indicating the presence of that specific amenity. This thus gave us the following columns:

	wifi	hair_dryer	shower_gel	shampoo	air_conditioning	tv	Refrigerator	kitchen	self_check_in	pets_allowed
0	1	0	0	0	0	1	1	1	0	1
1	1	0	0	0	1	0	1	1	0	0
2	1	1	0	1	0	0	1	1	1	0
3	1	0	0	0	1	0	0	1	0	0
4	1	1	0	1	1	0	1	1	1	0
...	...	...	...	...	...	...	...	...	...	...
3478	1	1	1	1	0	0	0	1	0	0
3479	1	0	0	0	1	1	0	0	0	0
3480	1	0	0	0	1	1	0	1	0	1
3481	1	1	0	0	1	1	1	1	1	0
3482	1	1	0	0	1	1	1	1	0	0

42243 rows × 10 columns

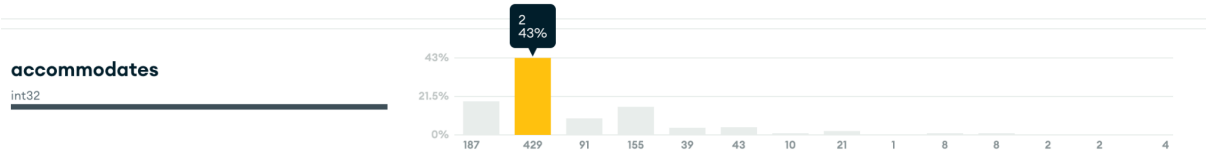
After these preprocessing steps and finalization, these are the columns we kept for the listing table:

```
listing_combined.columns
```

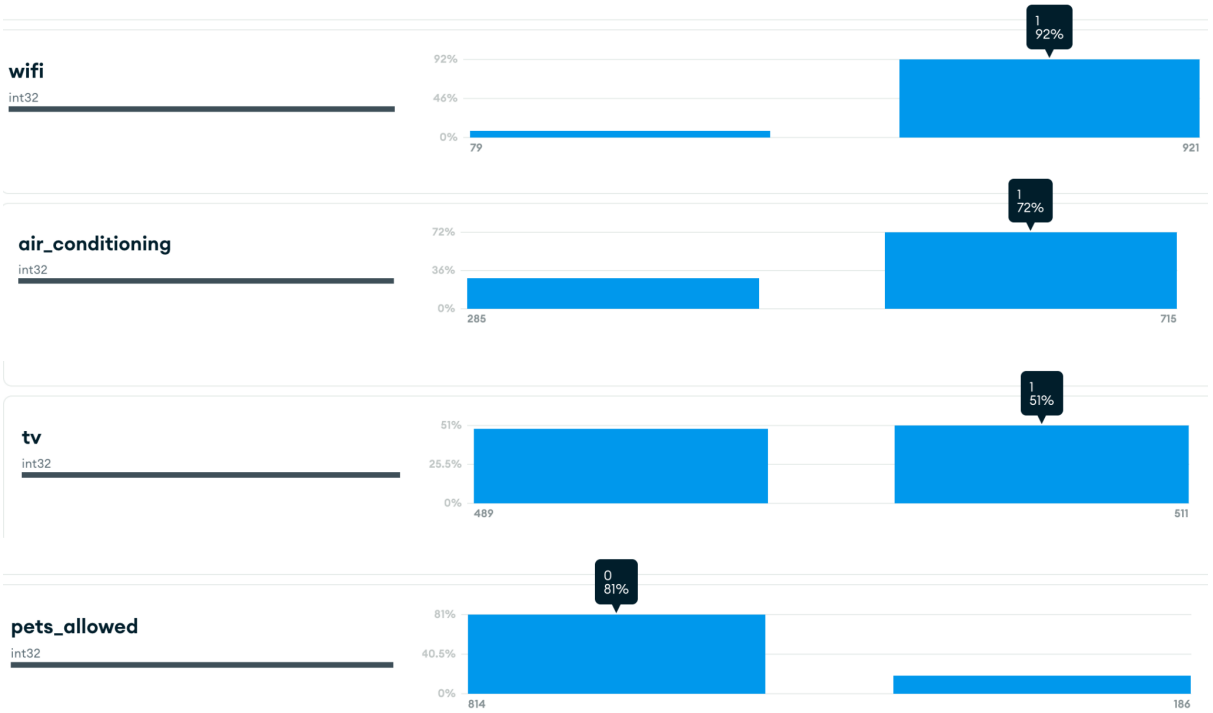
```
Index(['id', 'name', 'description', 'neighbourhood_cleansed',
       'neighborhood_overview', 'host_id', 'latitude', 'longitude',
       'property_type', 'room_type', 'accommodates', 'bedrooms', 'beds',
       'bathrooms', 'number_of_reviews', 'number_of_reviews_l30d',
       'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location', 'license',
       'instant_bookable', 'city', 'wifi', 'hair_dryer', 'shower_gel',
       'shampoo', 'air_conditioning', 'tv', 'Refrigerator', 'kitchen',
       'self_check_in', 'pets_allowed'],
      dtype='object')
```

Note that the attribute *city* here consists of three values: Singapore, Hong Kong, and New York. Each represents the city that the listing is located in.

For the data visualization part, we have analysed the distribution for some of the columns:



From the accommodates attribute, we can observe that 43% of the listings are targeting at couples or small groups of 2. When the accommodates required is greater than 7, the number of suitable listings drop significantly to 10.



Moreover it is also seen that 91% of the listings are equipped with Wifi, 72% of the listings have Air Conditioning and only 51% of the listings have TV. Besides, almost 81% of the listings are not pets allowed.

2.3. Calendar

The original calendar tables retrieved from Airbnb contain attributes as shown below. Each row shows a unique price detail, availability, and length of stay for a specific property at a specific date. The overall data shows the property pricing details from Sep 2023 to Sep 2024.

	listing_id	date	available	price	adjusted_price	minimum_nights	maximum_nights
0	17891	2023-09-17	f	$1,400.00	$1,400.00	60	365
1	17891	2023-09-18	f	$1,400.00	$1,400.00	60	365
2	17891	2023-09-19	f	$1,400.00	$1,400.00	60	365
3	17891	2023-09-20	f	$1,400.00	$1,400.00	60	365
4	17891	2023-09-21	f	$1,400.00	$1,400.00	60	365

After careful evaluation and preprocessing, we first map the columns ‘available’ to binary output ‘1’ and ‘0’ where ‘1’ represents ‘available’ and ‘0’ represents ‘unavailable’. Next, since the unit of price for the three cities is not standardized, we had to make some conversion to change the price to SGD unit and also drop the ‘$’ symbol.

```
calendar.isnull().sum()
```

```
cid              0
listing_id       0
date             0
available        0
price            0
minimum_nights   12
maximum_nights   12
dtype: int64
```

We also check for null values and duplicate data present in the dataset. Since the data is retrieved from the airbnb website, we only observe a few null values and duplicate data in the column ‘minimum_nights’ and ‘maximum_nights’. Therefore, we replace the null values in the ‘minimum_nights’ to be 1 and null values in the ‘maximum_nights’ to be 365 to ensure consistency in the dataset. The duplicate data is also removed by pd.drop_duplicate() function. Therefore, the preprocessed data is then shown as below.

	listing_id	date	available	price	minimum_nights	maximum_nights
0	17891	2023-09-17	0	243.90	60.0	365.0
1	17891	2023-09-18	0	243.90	60.0	365.0
2	17891	2023-09-19	0	243.90	60.0	365.0
3	17891	2023-09-20	0	243.90	60.0	365.0
4	17891	2023-09-21	0	243.90	60.0	365.0
...	...	...	...	...	...	...
17888648	992729511234270784	2024-09-25	0	273.33	30.0	365.0
17888649	992729511234270784	2024-09-26	0	273.33	30.0	365.0
17888650	992729511234270784	2024-09-27	0	273.33	30.0	365.0
17888651	992729511234270784	2024-09-28	0	273.33	30.0	365.0
17888652	992729511234270784	2024-09-29	0	273.33	30.0	365.0

2.4. Neighbourhood

The original dataset lacks corresponding 'neighbourhood_group' data for Hong Kong. To address this, we manually categorized Hong Kong's 18 neighbourhoods into four distinct groups: Hong Kong Island, Kowloon, New Territories and Islands.

2.5. Review

The original review tables retrieved from Airbnb contain attributes as shown below. Each row shows a specific Airbnb user (with unique id) and his comment on an Airbnb listing that he stayed in. Each review is made on a specific date.

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	2595	17857	2009-11-21	50679	Jean	Notre séjour de trois nuits.\r Nous avons ...
1	2595	19176	2009-12-05	53267	Cate	Great experience.
2	2595	19760	2009-12-10	38960	Anita	I've stayed with my friend at the Midtown Cast...
3	2595	34320	2010-04-09	71130	Kai-Uwe	We've been staying here for about 9 nights, en...
4	2595	46312	2010-05-25	117113	Alicia	We had a wonderful stay at Jennifer's charming...

In this table, the “comments” column is open for the user to input any content in any language including special characters. The values in the “comments” column are processed by changing the data type to string and regularising the delimiters and newline characters so that they are capable with SQL and MongoDB applications. The process is shown below:

```
# Convert the 'column_name' to string type
combined_df['comments'] = combined_df['comments'].astype(str)
```

```
type(combined_df['comments'][100])
```

```
str
```

```
import pandas as pd
import re

# Assuming df is your DataFrame and 'comments' is the column with the HTML content
combined_df['comments'] = combined_df['comments'].replace(to_replace=r'<br\s*\/?>', value='\n', regex=True)
```

```
combined_df['comments']
```

```
0          The rooms were clean and tidy. Beds very comfo...
1          Thank you for making me feel at home, it is su...
2          An absolutely amazing place. It’s clean, styli...
3          This location is private yet it is only 400m w...
4          Jeremy is a great host that will make your sta...
                                ...
1114700    先說總體感受，實在是太棒了！無論是房子還是房東 Coco，都是完美的！\n房子位置在油麻地，...
1114701    This apartment is only 3 mins walk from the Ya...
1114702    Awesome stay with Forrest. What a great guy. P...
1114703    卫生：很干净\n实用小贴士：楼下有按摩店累了可以去按摩放松一下\n沟通：房东很友好，阳光帅气...
1114704                                  房主人很好，交代清晰及顺利，推薦👍🏻
Name: comments, Length: 1114705, dtype: object
```

2.5. Data Generation

To construct a robust Airbnb database which is capable of handling diverse and complex queries, we decided to generate more data from the users’ perspective with a mock data generation tool. Two new relational tables are generated, namely the “user” and “transaction” table.

The “user” table contains information about Airbnb registered users with their personal information like username, registration date, email and location. Each user is distinguished by a unique attribute “user_id”. A brief display of the “user” table is shown below:

user_id	username	email	join_date	location
100001332	Alexandra Nicoleta	ebaybutt1321@sfgate.com	19-11-2017	Pakistan
100006379	Khamvilaya	akirrage25s@domainmarket.com	12-08-2017	Philippines
100007175	Philippa	wedwardes2u2@yellowbook.com	22-08-2018	Iran
100012743	Angelica	skaresqzm@cbsnews.com	05-04-2018	Poland
10001292	Susan	gpoundfordy07@skyrock.com	07-03-2016	Colombia

The “transaction” table contains the transaction records when a user booked and made payments for a listing. Every row represents a record of bill information which includes a unique transaction id, the involved user’s id and listing’s id, the booking date, the planned check-in and check-out date, the booked number of nights, the price and payment information, and the status of the booking (whether it is pending, completed or cancelled).

In this table, there are some derived attributes. The number of nights is derived from check-in and check-out date. The total price is derived from the number of nights and price per night. A brief display of the “transaction” table is shown below:

transaction_id	user_id	listing_id	booking_date	check_in_date	check_out_date	nights	price_per_night	total_price	payment_method	status
1	235988461	27618287	10/24/2023	12/15/2023	1/5/2024	21	353.78	7429.38	Mastercard	Pending
10	8833467	453094	9/30/2023	11/8/2023	12/5/2023	27	666.32	17990.64	Mastercard	Pending
100	180219863	27816487	11/27/2022	12/27/2022	12/29/2022	2	149.86	299.72	Visa	Pending
1000	44924067	822794157986992256	7/19/2023	7/21/2023	8/6/2023	16	320.05	5120.8	Visa	Cancelled
10000	11046359	39158146	11/23/2022	11/29/2022	12/14/2022	15	66.12	991.8	Mastercard	Pending
100000	496122025	536578	4/14/2023	5/27/2023	6/16/2023	20	318.15	6363	American Express	Cancelled
10001	466112267	907668348093927095	9/1/2023	10/11/2023	11/7/2023	27	355.47	9597.69	Cash	Pending
10002	3313865	906311410277025031	11/17/2022	11/29/2022	12/26/2022	27	905.78	24456.06	Visa	Pending

3. Data Storage

3.1. Relational Database - MySQL

3.1.1. Requirement Analysis

The Airbnb database needs to meet the needs of both daily and analytical queries. Daily queries may include:

1. Listing information management: It is crucial to first ensure that the dataset pertaining to listings is comprehensive, encompassing specific details about each property. Following this, efficient retrieval mechanisms for listing information must be in place. This feature is pivotal to the user experience on Airbnb, as it allows customers to efficiently find properties that align with their specific requirements. Users can filter their search based on various criteria, including location, price, and the availability of certain amenities. Alternatively, users can search for the highest-rated listings or those nearest to a particular attraction.
2. User information management: The Airbnb database should encompass more than just basic user details like name, birthday, registration time, and email. For hosts, it is essential to include a comprehensive list of their properties, verified identity information, and historical data on property rentals, including guest reviews, information on the hosts' responsiveness to inquiries and their rate of accepting

bookings. This additional information helps in creating a detailed profile or 'portrait' of the host, providing valuable insights to customers when making booking decisions.

3. Booking management: For efficient system operation, the Airbnb database must accurately record bookings and track the status of orders in real-time, including whether they are confirmed or canceled. Additionally, it's essential to update listing availability promptly to reflect these changes. This ensures the system remains up-to-date and reliable for both hosts and guests.
4. Review management: The review information of a property significantly influences customer decision-making during the reservation process. Therefore, the database should meticulously record details like ratings from previous tenants, the time of the review, and the review content. Moreover, considering Airbnb's global user base, reviews can be in multiple languages. To accommodate this diversity, the database requires proper encoding to accurately display these various languages, including the correct rendering of emoji emoticons.

In addition to daily operations, the Airbnb database must support analytical queries for strategic decision-making:

1. Market trend analysis: The Airbnb database should be capable of aggregating data to identify trends in popular destinations, detect seasonal variations in booking, and develop effective pricing strategies. Such capabilities are essential for gaining critical market insights.
2. Customer behavior analytics: Understanding customer preferences and booking habits through data analytics is key to delivering customized marketing and increasing satisfaction.

3.1.2. Conceptual Design - ER Model

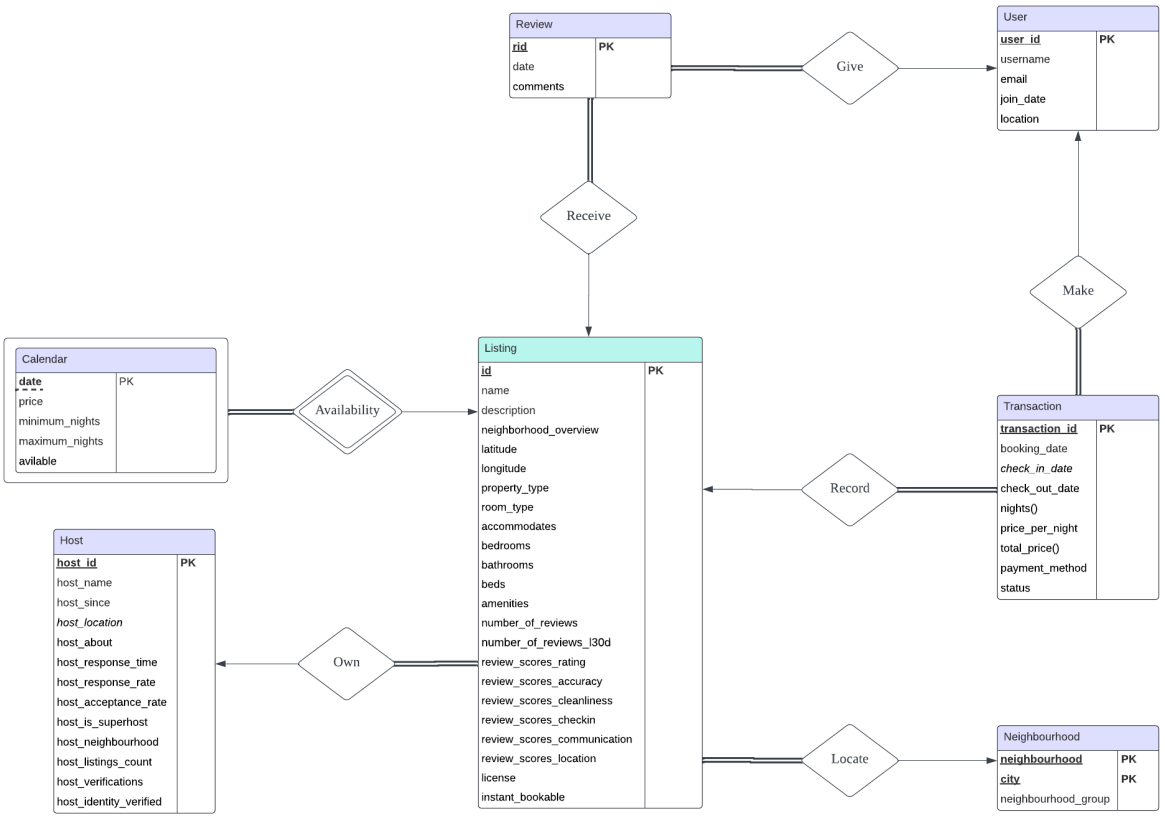
In the ER diagram, there are 6 entities and 1 weak entity in total. The 6 entities are “listing”, “transaction”, “review”, “neighbourhood”, “host” and “user”.

The relationship between “listing” and “review” is called “receive”, which represents the feedbacks provided to the Airbnb listings. The relationship between “user” and “review” is called “give”, since users give feedbacks to listings that they have booked and stayed.

The relationship between “listing” and “transaction” is called “record”, which represents the bill information for listings that are booked or have been booked before. The relationship between “user” and “transaction” is called “make”, since users make transactions when they book the listings.

The relationship between “listing” and “neighbourhood” is called “locate”, which elaborates on the detailed neighbourhood and location information for the Airbnb listings.

The weak entity in our conceptual design is called “calender”. The relationship between “listing” and “calender” is called “availability”, which specifies the availability of Airbnb listings over time. The “calender” is a weak entity because it relies on the “id” attribute in the “listing” relation, thus cannot exist on its own. The detailed ER diagram is shown below:



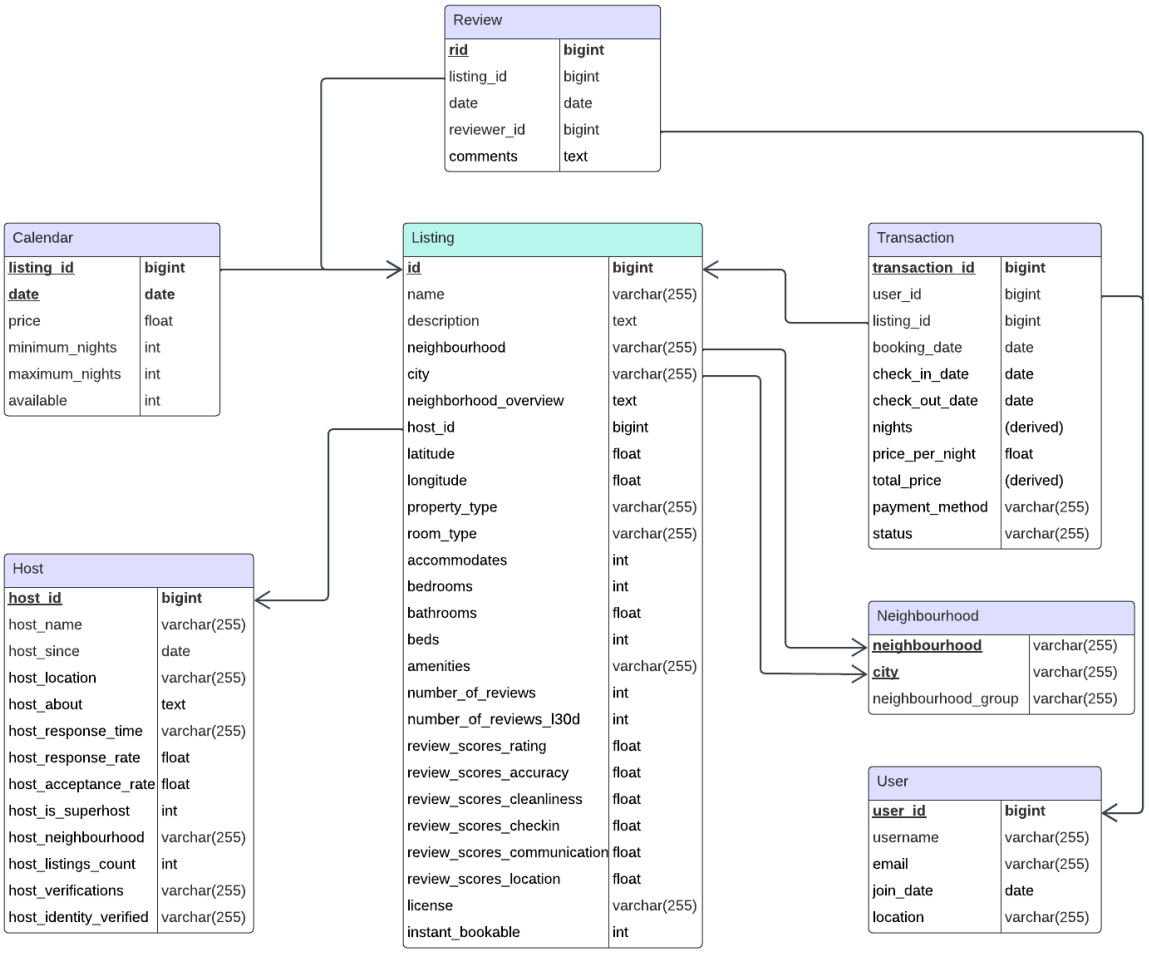
3.1.3. Logical Design - Translate to Relational Model

The ER diagram establishes a conceptual data model, which the relational schema then translates into a structured format optimized for database implementation. The conversion process begins by transforming each entity from the ER diagram into a corresponding table. For example, the 'listing' entity is converted into a 'listing' table, encompassing attributes such as id, name, and description.

Regarding the conversion of weak entities, such as 'calendar', it's necessary to incorporate the primary key 'listing_id' from its associated strong entity 'listing'. This key serves as both a primary and foreign key in the 'calendar' table, establishing a direct link to the 'listing' entity.

The conversion of relationships, particularly many-to-one relationships present in the ER diagram, involves appending the primary key from the 'one' side of the relationship to the table on the 'many' side as a foreign key. For instance, in the 'Make' and 'Record' relationship, this process entails adding the primary keys 'user_id' and 'listing_id' from the User and Listing entities, respectively, to the 'transaction' table. These primary keys are then used as foreign keys in the 'transaction' table to establish a relationship with the User and Listing entities.

The transformed relational model is shown below:



When actually storing the data into the MySQL database, since columns such as comments contain different languages as well as emoticons, we chose the 'utf8mb4' encoding method when creating the database to make all the content display correctly.

3.1.4. Schema Refinement - BCNF

The first step in the process of performing a schema refinement of our relational model to make it conform to the Boyce-Codd Normal Form (BCNF) is to identify all the function dependencies. FDs for each relation are as follows:

1. Calendar: (listing_id, date) -> all other attributes
2. Host: host_id -> all other attributes
3. Review: rid -> all other attributes
4. Listing: id -> all other attributes
5. Transaction: transaction_id -> all other attributes (disregard the derived attributes 'nights' and 'total_price')
6. Neighbourhood: (neighbourhood, city) -> all other attributes
7. User: user_id -> all other attributes

We can see that each table satisfies BCNF, as the determinant in each functional dependency is a superkey of the corresponding table. For ease of querying, we keep the derived attributes 'nights' and 'total_price' when storing the 'Transaction' table.

3.2. Non-relational Database - MongoDB

For a non-relational database system, we decided to use MongoDB. Unlike traditional relational databases that use tables with fixed schemas, MongoDB is document-oriented, storing data in JSON-like documents with dynamic schemas. This means that the structure of data in MongoDB can change over time, allowing for greater flexibility in data modeling.

Calendar

Storage size: 3.17 MB
Documents: 598 K
Avg. document size: 138.00 B
Indexes: 1
Total index size: 1.27 MB

Host

Storage size: 4.80 MB
Documents: 26 K
Avg. document size: 437.00 B
Indexes: 1
Total index size: 278.53 kB

Listing

Storage size: 4.10 kB
Documents: 42 K
Avg. document size: 1.74 kB
Indexes: 1
Total index size: 4.10 kB

Reviews

Storage size: 18.87 MB
Documents: 81 K
Avg. document size: 368.00 B
Indexes: 1
Total index size: 831.49 kB

Transaction

Storage size: 10.13 MB
Documents: 100 K
Avg. document size: 267.00 B
Indexes: 1
Total index size: 1.01 MB

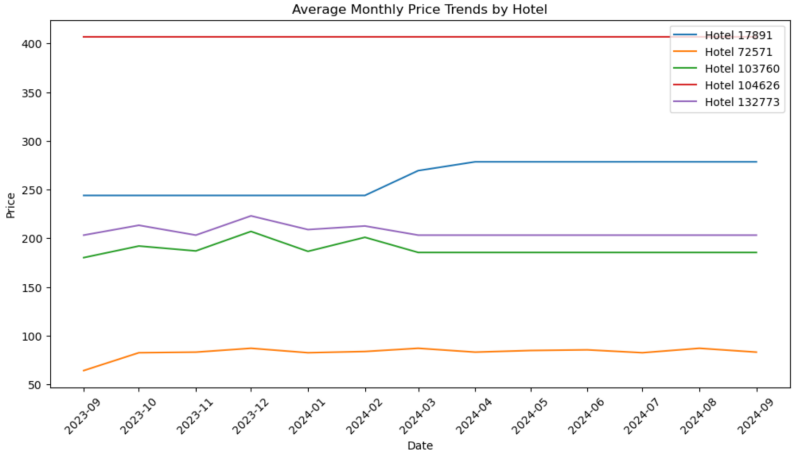
User

Storage size: 5.40 MB
Documents: 74 K
Avg. document size: 139.00 B
Indexes: 1
Total index size: 757.76 kB

However, compared to relational database systems, MongoDB put less emphasis on ACID properties (Atomicity, Consistency, Isolation, Durability). Furthermore, complex queries which require data from different collections can be more challenging to implement and less efficient.

Moreover, as a non-relational database does not require BCNF constraints, we decided to embed the small Neighbourhood collection under the Listing collection to avoid unnecessary lookup operations. Furthermore, we found out that due to the large size of the Calendar table, it is very slow and inefficient to carry out aggregate functions such as join to the Calendar table. Therefore, we make some further preprocessing on the Calendar table.

For data visualisation, we have randomly selected a few hotels to see how their price changes over years. The graph below suggests that the hotel price remains unchanged in one month and varies on a small scale over months.



Therefore, to avoid repeatedness of the data and ensure computational effectiveness, we group the hotel price by months and sum up the availability within a month. The new Calendar table used for MongoDB is then as shown below.

	listing_id	year	month	price	minimum_nights	maximum_nights	available
0	2595	2023	10	400.00	30.0	1125.0	16
1	2595	2023	11	400.00	30.0	1125.0	30
2	2595	2023	12	400.00	30.0	1125.0	31
3	2595	2024	1	400.00	30.0	1125.0	31
4	2595	2024	2	400.00	30.0	1125.0	29
...	...	...	...	...	...	...	...
590154	992729511234270720	2024	5	305.86	30.0	365.0	31
590155	992729511234270720	2024	6	305.66	30.0	365.0	28
590156	992729511234270720	2024	7	273.33	30.0	365.0	0
590157	992729511234270720	2024	8	273.33	30.0	365.0	0
590158	992729511234270720	2024	9	273.33	30.0	365.0	0

590159 rows × 7 columns

Hence, the final 6 collections we designed are – Calendar, Host, Listing, Reviews, Transaction, User. Example of listing data is shown below:


```
_id: ObjectId('65537d5320a377489292209b')
id: 17891
name: "Rental unit in Hong Kong Island · ★4.76 · Studio · 1 bed · 1 bath"
description: "Gorgeous and spacious loft, in the best location. Also available for C…"
neighbourhood: "Central & Western"
neighbourhood_group: "Hong Kong Island"
neighborhood_overview: "Best neighborhood in Hong Kong! A mix of old and new."
host_id: 69063
latitude: 22.28327
longitude: 114.14988
property_type: "Entire rental unit"
room_type: "Entire home/apt"
accommodates: 3
beds: 1
bathrooms: 1
number_of_reviews: 73
number_of_reviews_l30d: 0
review_scores_rating: 4.76
review_scores_accuracy: 4.73
review_scores_cleanliness: 4.51
review_scores_checkin: 4.92
review_scores_communication: 4.93
review_scores_location: 4.9
instant_bookable: false
city: "Hong Kong"
```

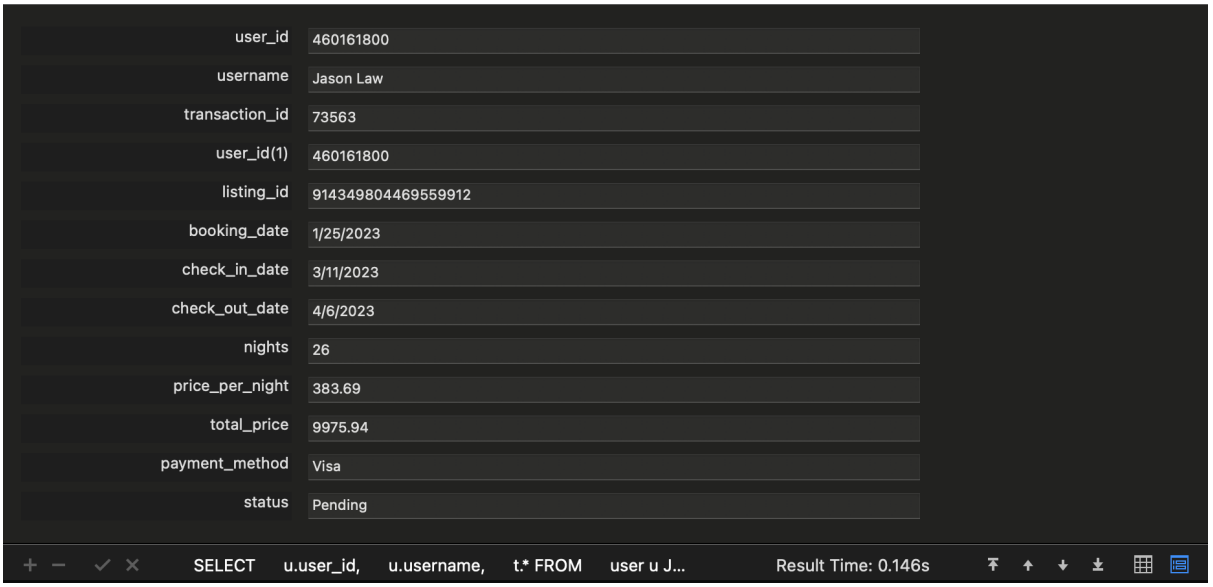
4. Data Access

(The full output is not fully shown in this section due to space constraints, please refer to Section 7 for the full output)

4.1. Find all transactions of the user named ‘Jason Law’. Show his user-id, username and exact his transaction details.

4.1.1. MySQL

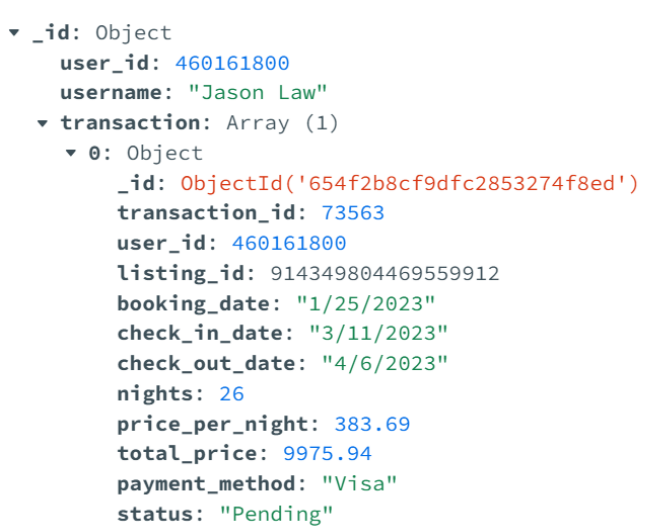
```
SELECT
    u.user_id,
    u.username,
    t.*
FROM
    user u
JOIN
    transaction t ON u.user_id = t.user_id
WHERE
    u.username = 'Jason Law';
```

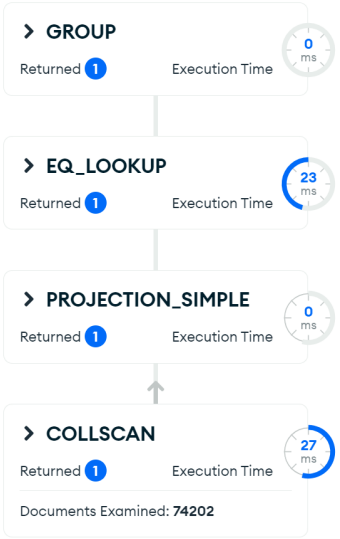


4.1.2. MongoDB

```
db.getCollection('User').aggregate(
 [
  {
   $lookup: {
    from: 'Transaction',
    localField: 'user_id',
    foreignField: 'user_id',
    as: 'transaction'
   }
```

```
    },
    { $match: { username: 'Jason Law' } },
    {
      $group: {
        _id: {
          user_id: '$user_id',
          username: '$username',
          transaction: '$transaction'
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

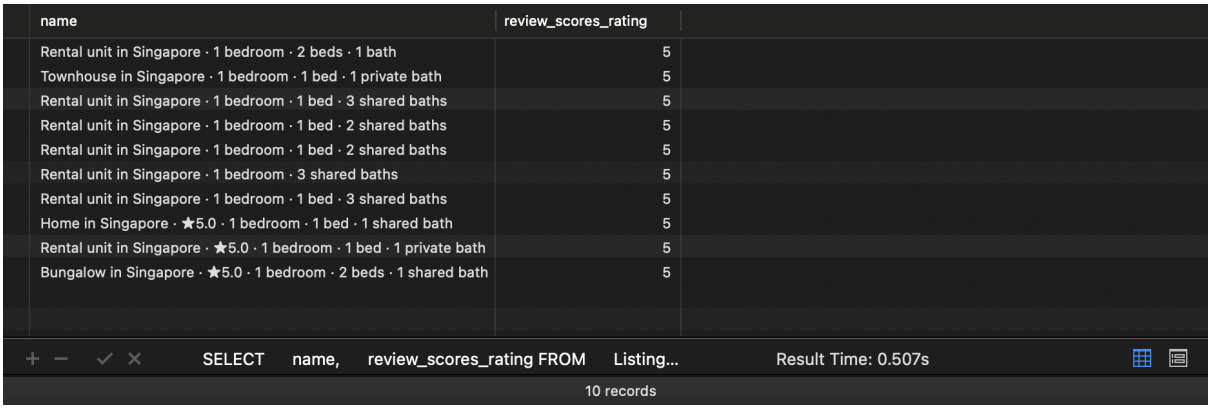




4.2. Find the name and rating scores of top 10 properties in Singapore with the highest review_score_rating.

4.2.1. MySQL

```
SELECT
   name,
   review_scores_rating
FROM
   Listing
WHERE
   city = 'Singapore'
ORDER BY
   review_scores_rating DESC
LIMIT 10;
```



name	review_scores_rating
Rental unit in Singapore · 1 bedroom · 2 beds · 1 bath	5
Townhouse in Singapore · 1 bedroom · 1 bed · 1 private bath	5
Rental unit in Singapore · 1 bedroom · 1 bed · 3 shared baths	5
Rental unit in Singapore · 1 bedroom · 1 bed · 2 shared baths	5
Rental unit in Singapore · 1 bedroom · 1 bed · 2 shared baths	5
Rental unit in Singapore · 1 bedroom · 3 shared baths	5
Rental unit in Singapore · 1 bedroom · 1 bed · 3 shared baths	5
Home in Singapore · ★5.0 · 1 bedroom · 1 bed · 1 shared bath	5
Rental unit in Singapore · ★5.0 · 1 bedroom · 1 bed · 1 private bath	5
Bungalow in Singapore · ★5.0 · 1 bedroom · 2 beds · 1 shared bath	5

4.2.2. MongoDB

```
db.getCollection('Listing').aggregate(
 [
  { $match: { city: 'Singapore' } },
  { $sort: { review_scores_rating: -1 } },
  { $limit: 10 },
  {
   $project: {
    name: 1,
    review_scores_rating: 1
   }
  }
 ],
 { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
_id: ObjectId('6550bacc00de12b981d17d76')
name: "Rental unit in Singapore · 1 bedroom · 1 bed · 1 private bath"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17d96')
name: "Loft in Singapore · ★5.0 · 1 bedroom · 1 bed · 1.5 baths"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17d9e')
name: "Rental unit in Singapore · 1 bedroom · 1 bed · 1 private bath"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17d72')
name: "Rental unit in Singapore · 1 bedroom · 1 bed · 1 private bath"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17d8d')
name: "Condo in Singapore · ★5.0 · 1 bedroom · 1 bed · Shared half-bath"
review_scores_rating: 5
```

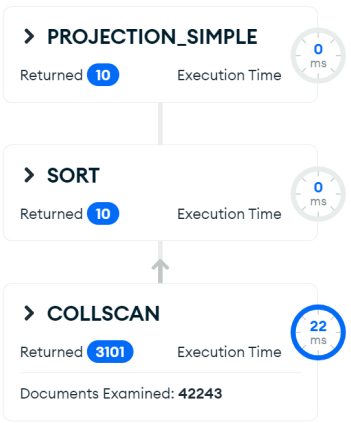
```
_id: ObjectId('6550bacc00de12b981d17d79')
name: "Rental unit in Singapore · ★5.0 · 1 bedroom · 1 bed · 1 private bath"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17da0')
name: "Rental unit in Singapore · 1 bedroom · 1 bed · 2 shared baths"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17d9f')
name: "Rental unit in Singapore · ★5.0 · 1 bedroom · 1 bed · 1 shared bath"
review_scores_rating: 5
```

```
_id: ObjectId('6550bacc00de12b981d17d9c')
name: "Condo in Singapore · 1 bedroom · 1 bed · 1 private bath"
review_scores_rating: 5
```

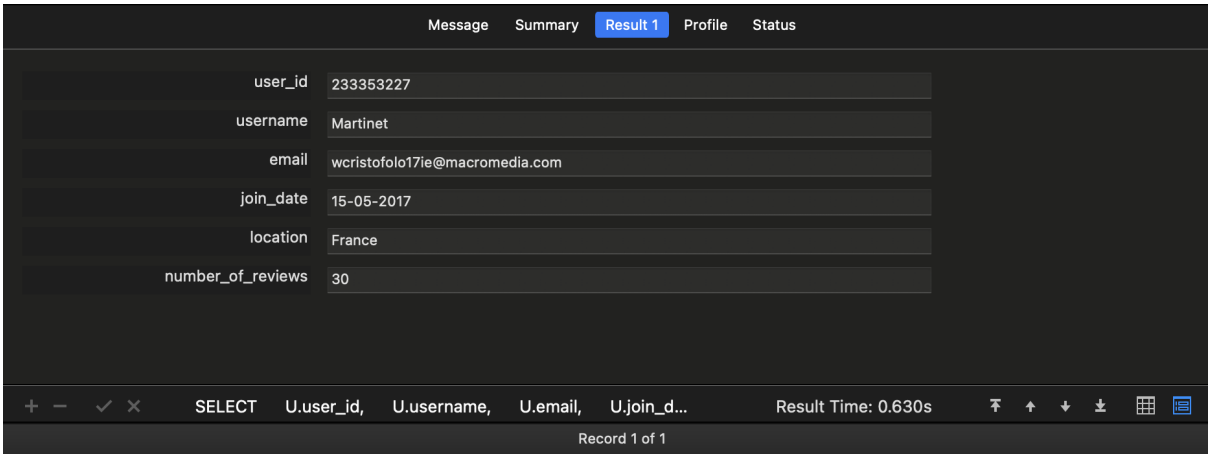
```
_id: ObjectId('6550bacc00de12b981d17d71')
name: "Condo in Singapore · 1 bedroom · 2 beds · 1 private bath"
review_scores_rating: 5
```



4.3. Find the user who made the highest number of reviews, show the detailed user information and the number of reviews he/she made.

4.3.1. MySQL

```
SELECT
    U.user_id,
    U.username,
    U.email,
    U.join_date,
    U.location,
    COUNT(R.reviewer_id) AS number_of_reviews
FROM
    User U
JOIN
    Review R ON U.user_id = R.reviewer_id
GROUP BY
    U.user_id
ORDER BY
    number_of_reviews DESC
LIMIT 1;
```

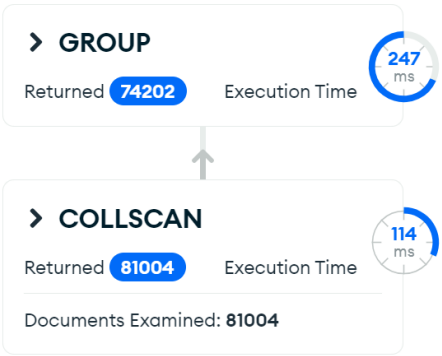


4.3.2. MongoDB

```
db.getCollection('review_preprocessed').aggregate(
 [
  {
   $group: {
    _id: {
     reviewer_id: '$reviewer_id',
     reviewer_name: '$reviewer_name'
    },
    count: { $sum: 1 }
   }
  },
  { $sort: { count: -1 } },
```

```
  {
    $lookup: {
      from: 'user_table',
      localField: '_id.reviewer_id',
      foreignField: 'user_id',
      as: 'user_information'
    }
  },
  { $limit: 1 }
 ],
 { maxTimeMS: 60000, allowDiskUse: true }
);
```

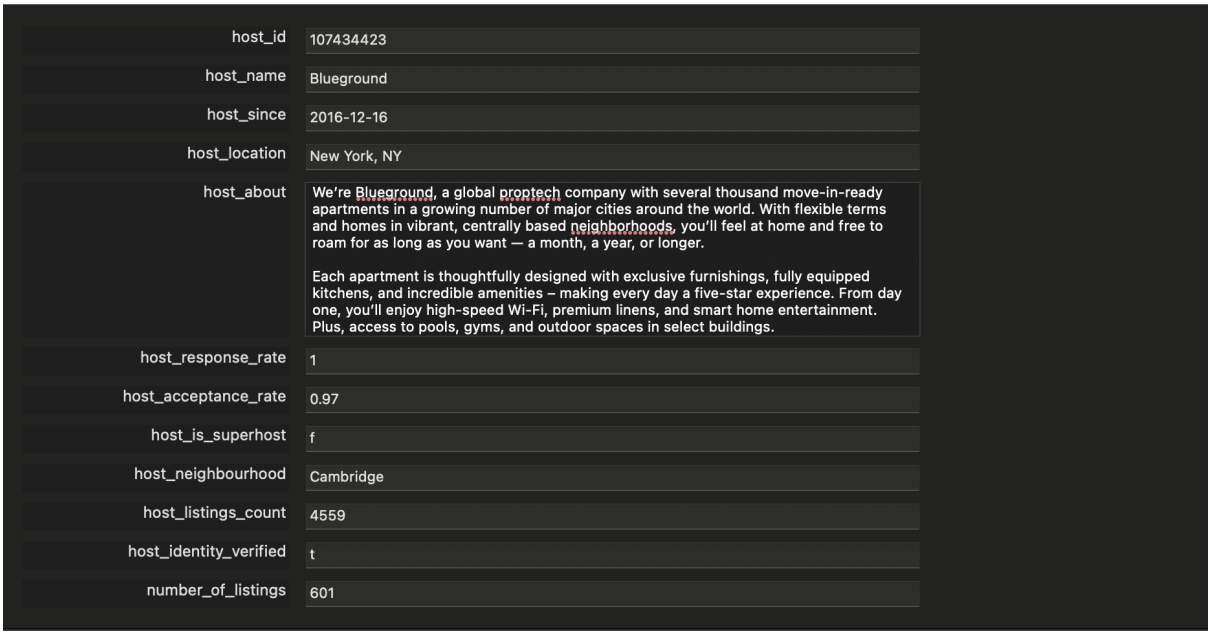
```
▾ _id: Object
    reviewer_id: 233353227
    reviewer_name: "Martinet"
  count: 30
▾ user_information: Array (1)
  ▾ 0: Object
      _id: ObjectId('654f929829f50c5798404189')
      user_id: 233353227
      username: "Martinet"
      email: "wcristofolo17ie@macromedia.com"
      join_date: "15-05-2017"
      location: "France"
```



4.4. Find the host who owns the highest number of listings, show the detailed host information and the number of listings he/she has.

4.4.1. MySQL

```
SELECT
    H.host_id,
    H.host_name,
    H.host_since,
    H.host_location,
    H.host_about,
    H.host_response_rate,
    H.host_acceptance_rate,
    H.host_is_superhost,
    H.host_neighbourhood,
    H.host_listings_count,
    H.host_identity_verified,
    COUNT(L.id) AS number_of_listings
FROM
    Host H
LEFT JOIN
    Listing L ON H.host_id = L.host_id
GROUP BY
    H.host_id,
    H.host_name,
    H.host_since,
    H.host_location,
    H.host_about,
    H.host_response_rate,
    H.host_acceptance_rate,
    H.host_is_superhost,
    H.host_neighbourhood,
    H.host_listings_count,
    H.host_identity_verified
ORDER BY
    number_of_listings DESC
LIMIT 1;
```

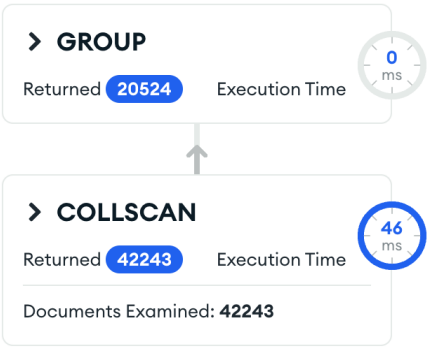


host_id	107434423
host_name	Blueground
host_since	2016-12-16
host_location	New York, NY
host_about	We're Blueground, a global proptech company with several thousand move-in-ready apartments in a growing number of major cities around the world. With flexible terms and homes in vibrant, centrally based neighborhoods, you'll feel at home and free to roam for as long as you want — a month, a year, or longer. Each apartment is thoughtfully designed with exclusive furnishings, fully equipped kitchens, and incredible amenities – making every day a five-star experience. From day one, you'll enjoy high-speed Wi-Fi, premium linens, and smart home entertainment. Plus, access to pools, gyms, and outdoor spaces in select buildings.
host_response_rate	1
host_acceptance_rate	0.97
host_is_superhost	f
host_neighbourhood	Cambridge
host_listings_count	4559
host_identity_verified	t
number_of_listings	601

4.4.2. MongoDB

```
db.getCollection('listing_combined').aggregate(
 [
  {
   $group: {
    _id: '$host_id',
    count: { $sum: 1 }
   }
  },
  { $sort: { count: -1 } },
  { $limit: 1 },
  {
   $lookup: {
    from: 'host_combined',
    localField: '_id',
    foreignField: 'host_id',
    as: 'host_information'
   }
  }
 ],
 { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
_id: 107434423
count: 601
▾ host_information: Array (1)
  ▾ 0: Object
      _id: ObjectId('655087573ff3ad5caa82261c')
      host_id: 107434423
      host_name: "Blueground"
      host_since: 2016-12-16T00:00:00.000+00:00
      host_location: "New York, NY"
      host_about: "We’re Blueground, a global proptech company with several thousand move…"
      host_response_time: "within an hour"
      host_response_rate: 1
      host_acceptance_rate: 0.97
      host_is_superhost: false
      host_neighbourhood: "Cambridge"
      host_listings_count: 4559
      host_verifications: "['email', 'phone', 'work_email']"
      host_identity_verified: true
      city: "New York"
```

4.5. A customer would like to book one of these properties for a trip to Singapore in May 2024: located in the Marina East area, four people allowed, pets allowed, wifi, and priced under S$300 per night. List first 5 results that match (Marina East is located at the central region of Singapore).

4.5.1. MySQL

```
SELECT
    L.id,
    L.name,
    L.description,
    C.price,
    L.accommodates
FROM
    Listing L
JOIN
    Calendar C ON L.id = C.listing_id
JOIN
    Neighbourhood N ON L.neighbourhood_cleansed = N.neighbourhood
WHERE
    N.city = 'Singapore' AND
    N.neighbourhood_group = 'Central Region' AND
    L.accommodates >= 4 AND
    L.pets_allowed = 1 AND
    L.wifi = 1 AND
    C.price < 300 AND
    C.available > 0 AND
    C.year = 2024 AND
    C.month = 5
GROUP BY
    L.id
```

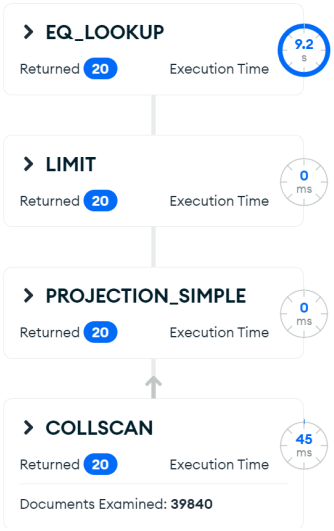
id	name	description	price	accommodates
10399199	Condo in Singapore · ★4.94 · 2 bedrooms · 2 beds · 1 bath	This is a 2bed, 2bath condo apartment 150 meters from Fort Cann	246	8
23874999	Condo in Singapore · ★4.93 · 1 bedroom · 1 bed · 1.5 baths	Bedroom with independent private bathroom, in large condo apart	130	6
25711980	Rental unit in Singapore · 2 bedrooms · 2 beds · 1 bath	Gives easy accessibility to all areas in Singapore, these furnished	299	4
33888989	Serviced apartment in Orchard · ★4.56 · 1 bedroom · 2 beds · 1 bath	Our licensed serviced apartment is set in the heart of city. It is cor	298	4
34569244	Condo in Singapore · 1 bedroom · 1 bed · 1 private bath	Condo bedroom in 2-bedroom condo apartment 150 meters from	137	6
34569884	Condo in Singapore · ★4.50 · 2 bedrooms · 6 beds · 2 baths	This is a 2bed, 2bath condo apartment 150 meters from Fort Cann	254	8
35237622	Condo in Singapore · ★4.80 · 1 bedroom · 1 bed · 1 private bath	Condo bedroom with independent bathroom, in large condo apart	171	6
38395413	Rental unit in Singapore · 2 bedrooms · 3 beds · 2 baths	Conveniently located in Orchard area, Singapore's largest shoppin	250	4
38951604	Condo in Singapore · ★5.0 · 2 bedrooms · 2 beds · 2 baths	This is a 2bed, 2bath condo apartment 150 meters from Fort Cann	273	8
38965851	Condo in Singapore · ★4.67 · 1 bedroom · 1 bed · 1 private bath	Condo bedroom in 2bed, 2bath condo apartment 150 meters from	137	6
41162396	Serviced apartment in Singapore · 2 bedrooms · 2 beds · 2 baths	Integral accommodation solution, providing space, flexibility and a	290	4
41482560	Boutique hotel in Singapore · ★4.95 · 1 bedroom · 2 beds · 1 private bath	Welcome to Singapore's very first "pop-up" shipping container Tir	265.55	4
42462883	Condo in Singapore · 2 bedrooms · 2 beds · 2 baths	An ideal choice for medical tourism, these are spacious and conve	185	4
43129915	Condo in Singapore · 2 bedrooms · 2 beds · 2 baths	Your home away from home is located is within walking distance fr	194	4
43313632	Rental unit in Singapore · 3 bedrooms · 3 beds · 2 baths	It is very near to novena MRT station <b>The space</b	261	6
49133042	Serviced apartment in Tanjong Pagar · ★4.17 · Studio · 1 bed · 1.5 baths	When you stay with us, you will stay right in the heart of downtowr	198	4
50404949	Serviced apartment in Singapore · ★2.50 · 1 bedroom · 2 beds · 1 bath	we welcome you and your pets with open arms. serviced apartmer	208	4
50493546	Condo in Singapore · 2 bedrooms · 2 beds · 2 baths	Located in CBD area, its a spacious 2-bedroom apartment which is	220	4
52545178	Condo in Singapore · 2 bedrooms · 2 beds · 2 baths	Located in downtown and a 12-minutes walk to Orchard MRT provi	290	4
53057197	Hotel in Singapore · Studio · 1 bed · 1 bath	Designed for privacy and spacious, cosy living, our Studio Executi	254	4
53370046	Condo in Singapore · 3 bedrooms · 3 beds · 2 baths	An ideal choice for medical tourism, these are spacious and conve	274	6
8200542	Condo in Singapore · ★4.45 · 3 bedrooms · 4 beds · 2 baths	2 balconies!- Sky Pool at 20th floor rooftop! 5 mins walk to Paya L	265.81	5
8277151	Serviced apartment in Orchard · ★4.71 · Studio · 2 beds · 1 bath	Our serviced apartment is set in the heart of city. It is convenient,	288	4

1 row selected Result Time: 0.619s

Record 1 of 23

4.5.2. MongoDB

```
db.getCollection('Listing').aggregate(
  [
    {
      $match: {
        city: 'Singapore',
        pets_allowed: 1,
        neighbourhood_group: 'Central Region',
        accommodates: { $gte: 4 },
        wifi: 1
      }
    },
    {
      $lookup: {
        from: 'Calendar',
        localField: 'id',
        foreignField: 'listing_id',
        as: 'Calendar'
      }
    },
    {
      $project: {
        id: 1,
        name: 1,
        Price: {
          $filter: {
            input: '$Calendar',
            as: 'calendar',
            cond: {
              $and: [
                {
                  $eq: ['$$calendar.year', 2024]
                },
                { $eq: ['$$calendar.month', 5] },
                { $lt: ['$$calendar.price', 300] }
              ]
            }
          }
        }
      }
    },
    { $limit: 20 },
    { $match: { 'Price.0': { $exists: true } } },
    { $limit: 5 }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```




```
_id: ObjectId('65537d5720a377489292ba16')
id: 8200542
name: "Condo in Singapore · ★4.45 · 3 bedrooms · 4 beds · 2 baths"
▸ Price: Array (1)

_id: ObjectId('65537d5720a377489292ba1a')
id: 8277151
name: "Serviced apartment in Orchard  · ★4.71 · Studio · 2 beds · 1 bath"
▸ Price: Array (1)

_id: ObjectId('65537d5720a377489292ba5c')
id: 10399199
name: "Condo in Singapore · ★4.94 · 2 bedrooms · 2 beds · 1 bath"
▸ Price: Array (1)

_id: ObjectId('65537d5720a377489292ba92')
id: 13454638
name: "Rental unit in Singapore · ★4.29 · 3 bedrooms · 3 beds · 2.5 baths"
▸ Price: Array (1)

_id: ObjectId('65537d5720a377489292bbaf')
id: 22233862
name: "Home in Singapore · ★4.81 · 1 bedroom · 2 beds · 2 shared baths"
▸ Price: Array (1)
```


4.5.3. MySQL: This is a extra type of query where SQL is more capable of handling. When the Airbnb user wants to check the availability of a specific range of dates, for example: A customer would like to book one of these properties for a trip to Singapore from 2024.5.1 to 2024.5.5: located in the Marina East area, four people allowed, pets allowed, wifi, and priced under S$300 per night. List first 5 results that match.

```
SELECT
    L.id,
    L.name,
    L.description,
    L.accommodates
FROM
```

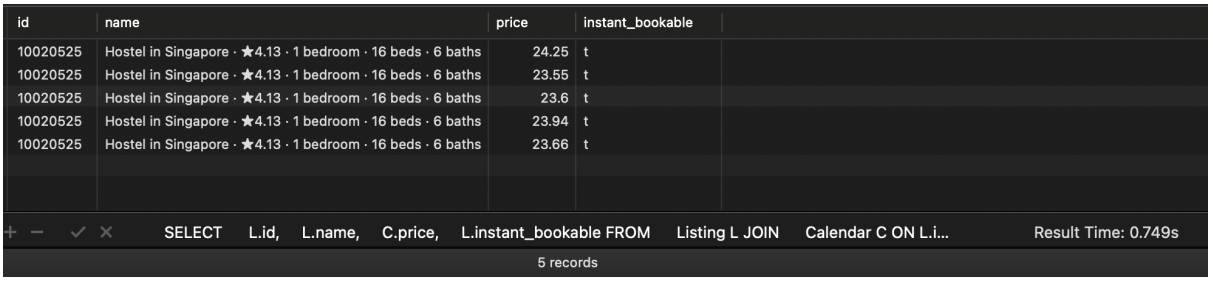
```
    Listing L
JOIN
    calendar_date C ON L.id = C.listing_id
JOIN
    Neighbourhood N ON L.neighbourhood_cleansed = N.neighbourhood
WHERE
    N.neighbourhood_group = 'Central Region' AND
                    N.city = 'Singapore' AND
    L.accommodates >= 4 AND
    L.pets_allowed = 1 AND
    L.wifi = 1 AND
    C.price < 300 AND
    C.available = 1 AND
    C.date BETWEEN '2024-05-01' AND '2024-05-05'
GROUP BY
    L.id
HAVING
    COUNT(C.date) >= 4
Limit 5;
```

id	name	description	accommodates
10399199	Condo in Singapore · ★4.94 · 2 bedrooms · 2 beds · 1 bath	This is a 2bed, 2bath condo apartment 150 meters from Fort Cann	8
23874999	Condo in Singapore · ★4.93 · 1 bedroom · 1 bed · 1.5 baths	Bedroom with independent private bathroom, in large condo apart	6
25711980	Rental unit in Singapore · 2 bedrooms · 2 beds · 1 bath	Gives easy accessibility to all areas in Singapore, these furnished	4
33888989	Serviced apartment in Orchard · ★4.56 · 1 bedroom · 2 beds · 1 bath	Our licensed serviced apartment is set in the heart of city. It is cor	4
34569244	Condo in Singapore · 1 bedroom · 1 bed · 1 private bath	Condo bedroom in 2-bedroom condo apartment 150 meters from l	6

4.6. (Daily Query) Find all the instant bookable listings with price per night less than 50 dollars. Show only the first 5 results.

4.6.1. MySQL

```
SELECT
   L.id,
   L.name,
   C.price,
   L.instant_bookable
FROM
   Listing L
JOIN
   Calendar C ON L.id = C.listing_id
WHERE
   L.instant_bookable = 't' AND
   C.price < 50
GROUP BY
   L.id,
   L.name,
   C.price,
   L.instant_bookable
Limit 5;
```



id	name	price	instant_bookable
10020525	Hostel in Singapore · ★4.13 · 1 bedroom · 16 beds · 6 baths	24.25	t
10020525	Hostel in Singapore · ★4.13 · 1 bedroom · 16 beds · 6 baths	23.55	t
10020525	Hostel in Singapore · ★4.13 · 1 bedroom · 16 beds · 6 baths	23.6	t
10020525	Hostel in Singapore · ★4.13 · 1 bedroom · 16 beds · 6 baths	23.94	t
10020525	Hostel in Singapore · ★4.13 · 1 bedroom · 16 beds · 6 baths	23.66	t

4.6.2. MongoDB

```
db.getCollection('Listing').aggregate(
  [
    { $match: { instant_bookable: true } },
    {
      $lookup: {
        from: 'Calendar',
        localField: 'id',
        foreignField: 'listing_id',
        as: 'Calendar'
      }
    },
    {
      $project: {
        id: 1,
        name: 1,
```

```
      instant_bookable: 1,
      Price: {
        $filter: {
          input: '$Calendar',
          as: 'calendar',
          cond: {
            $lt: ['$$calendar.price', 50]
          }
        }
      }
    }
  },
  { $limit: 20 },
  { $match: { Price: { $ne: [] } } },
  { $limit: 5 }
 ],
 { maxTimeMS: 60000, allowDiskUse: true }
);
```

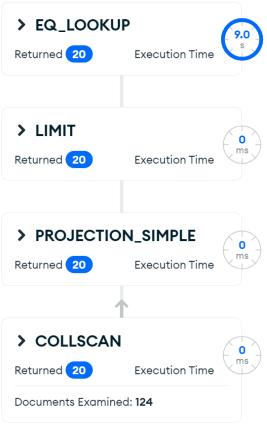
```
_id: ObjectId('65537d5320a37748929220aa')
id: 562683
name: "Rental unit in Wan Chai · ★4.43 · 1 bedroom · 8 beds · 3 shared baths"
instant_bookable: true
▸ Price: Array (6)
```

```
_id: ObjectId('65537d5320a37748929220ab')
id: 562808
name: "Rental unit in Wan Chai · ★4.44 · 1 bedroom · 8 beds · 2 shared baths"
instant_bookable: true
▸ Price: Array (6)
```

```
_id: ObjectId('65537d5320a37748929220ac')
id: 562817
name: "Rental unit in Wan Chai · ★4.43 · 1 bedroom · 8 beds · 3 shared baths"
instant_bookable: true
▸ Price: Array (8)
```

```
_id: ObjectId('65537d5320a37748929220ad')
id: 562840
name: "Rental unit in Wan Chai · ★4.52 · 1 bedroom · 8 beds · 3 shared baths"
instant_bookable: true
▸ Price: Array (6)
```

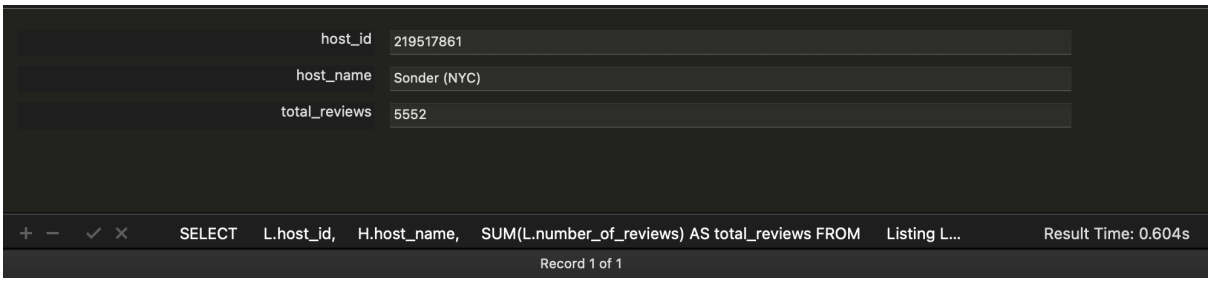
```
_id: ObjectId('65537d5320a37748929220b4')
id: 645404
name: "Hostel in Hong Kong · ★4.64 · 1 bedroom · 1 bed · 2.5 shared baths"
instant_bookable: true
▸ Price: Array (13)
```



4.7. (Analysis Query) Find the host who has received the highest number of reviews on his listings.

4.7.1. MySQL

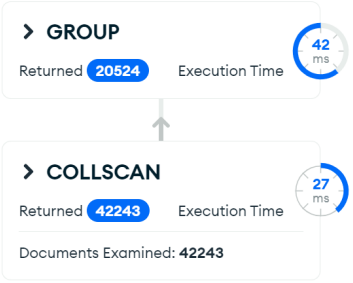
```
SELECT
   L.host_id,
   H.host_name,
   SUM(L.number_of_reviews) AS total_reviews
FROM
   Listing L
JOIN
   Host H ON L.host_id = H.host_id
GROUP BY
   L.host_id, H.host_name
ORDER BY
   total_reviews DESC
LIMIT 1;
```



4.7.2. MongoDB

```
db.getCollection('Listing').aggregate(
  [
    {
      $group: {
        _id: '$host_id',
        total_reviews: {
          $sum: '$number_of_reviews'
        }
      }
    },
    { $sort: { total_reviews: -1 } },
    { $limit: 1 },
    {
      $lookup: {
        from: 'Host',
        localField: '_id',
        foreignField: 'host_id',
        as: 'host'
      }
    },
```

```
  { $unwind: { path: '$host' } },
  {
    $project: {
      host_id: '$_id',
      total_reviews: 1,
      host_name: '$host.host_name'
    }
  }
 ],
 { maxTimeMS: 60000, allowDiskUse: true }
);
```



4.8. Find the first 10 name, description, region, accommodates, and Price details of all hotels located at Central Region in Nov, 2023

4.8.1. MySQL

```
SELECT
    L.name,
    L.description,
    N.neighbourhood_group AS region,
    L.accommodates,
    C.price
FROM
    Listing L
JOIN
    Neighbourhood N ON L.neighbourhood_cleansed = N.neighbourhood
JOIN
    Calendar C ON L.id = C.listing_id
WHERE
    N.neighbourhood_group = 'Central Region' AND
    N.city = 'Singapore' AND
    C.year = 2023 AND
    C.month = 11 AND
    C.available > 0
GROUP BY
    L.id
ORDER BY
    L.id ASC
LIMIT 10;
```

name	description	region	accommodates	price
Hostel in Singapore · ★4.65 · 1 bedroom · 4 beds · 7 sharec	Singapore's 1st Space Themed Luxury Capsule Hostel. Strategically located at Boat Quay, the Hea	Central Region	16	55.53
Hostel in Singapore · ★4.13 · 1 bedroom · 16 beds · 6 baths	Welcome to the Meadows Hostel. A newly-opened hostel superbly located in the heart of Singapor	Central Region	16	23.6
Hostel in Singapore · ★4.21 · 1 bedroom · 2 beds · 8 baths	Welcome to the Meadow Hostel. A newly opened hostel superbly located in the heart of Singapore	Central Region	16	47.2
Hostel in Singapore · ★4.0 · 1 bedroom · 16 beds · 8 shared	Welcome to the Meadows Hostel. A newly-opened hostel superbly located in the heart of Singapor	Central Region	16	47.2
Hostel in Singapore · ★4.18 · 1 bedroom · 16 beds · 6 share	Welcome to the Meadows Hostel. A newly-opened hostel superbly located in the heart of Singapor	Central Region	16	23.6
Hostel in Singapore · ★4.18 · 1 bedroom · 16 beds · 6 baths	Welcome to the Meadows Hostel. A newly opened hostel superbly located in the heart of Singapor	Central Region	16	23.6
Hostel in Singapore · ★4.27 · 1 bedroom · 16 beds · 6 baths	Welcome to the Meadows Hostel. A newly opened hostel superbly located in the heart of Singapor	Central Region	16	23.6
Hostel in Singapore · ★4.09 · 1 bedroom · 16 beds · 6 share	Welcome to the Meadows Hostel. A newly opened hostel superbly located in the heart of Singapor	Central Region	16	23.6
Hostel in Singapore · ★4.0 · 1 bedroom · 16 beds · 6 shared	Welcome to the Meadows Hostel. A newly opened hostel superbly located in the heart of Singapor	Central Region	16	23.6
Hostel in Singapore · ★4.12 · 1 bedroom · 16 beds · 6 baths	Welcome to the Meadows Hostel. A newly opened hostel superbly located in the heart of Singapor	Central Region	16	47.2

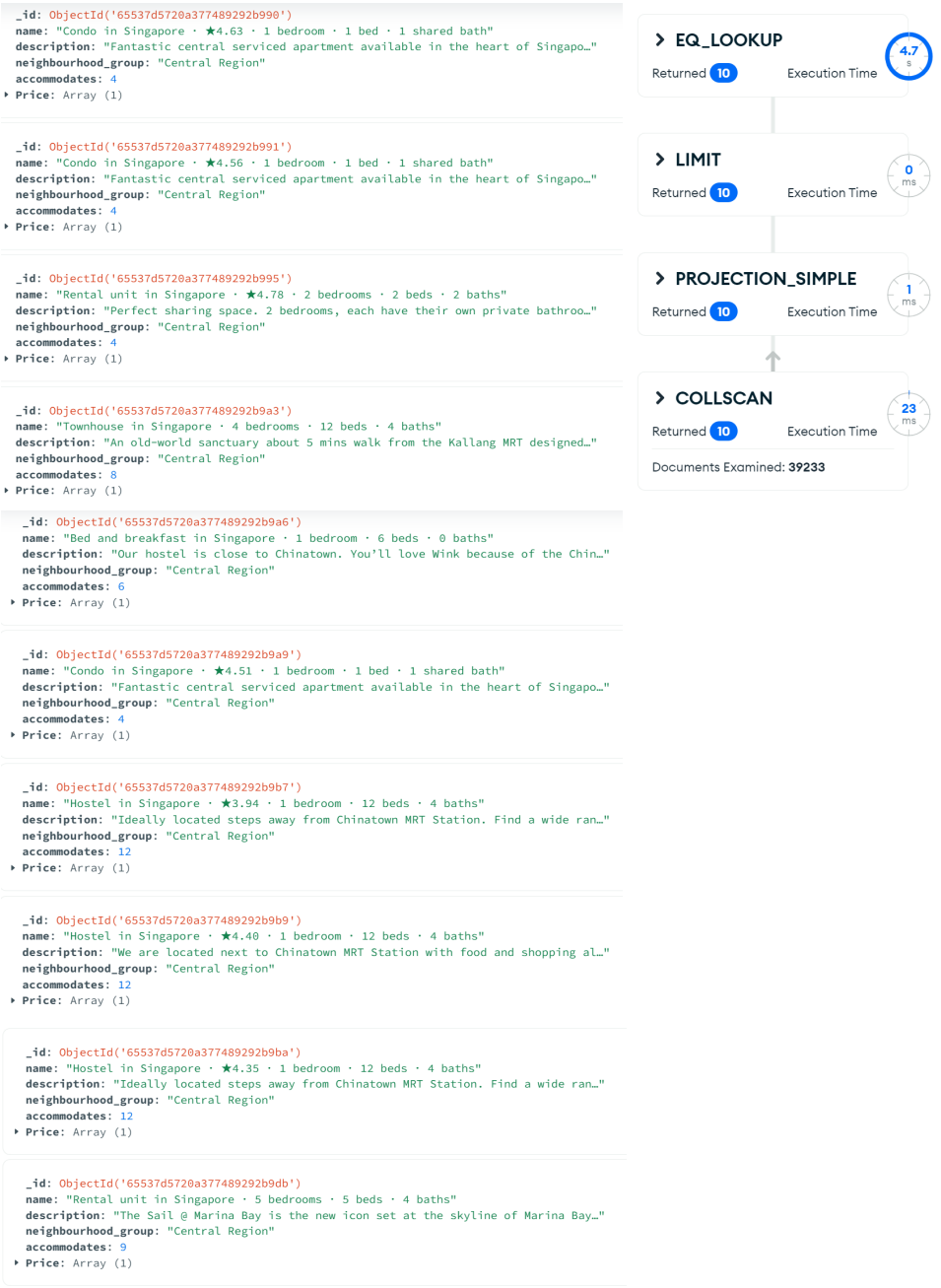
SELECT L.name, L.description, N.neighbourhood_group AS region, L.accommodates, C.price... Result Time: 0.005s

10 records

4.8.2 MongoDB

```
db.getCollection('Listing').aggregate(
  [
    {
      $match: {
        accommodates: { $gte: 4 },
        city: 'Singapore',
        neighbourhood_group: 'Central Region'
      }
    },
```

```
    {
      $lookup: {
        from: 'Calendar',
        localField: 'id',
        foreignField: 'listing_id',
        as: 'Price'
      }
    },
    {
      $project: {
        name: 1,
        description: 1,
        neighbourhood_cleansed: 1,
        neighbourhood_group: 1,
        accommodates: 1,
        Price: {
          $filter: {
            input: '$Price',
            as: 'price',
            cond: {
              $and: [
                { $eq: ['$$price.month', 11] },
                { $eq: ['$$price.year', 2023] }
              ]
            }
          }
        }
      }
    },
    { $limit: 10 }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```



4.9. When there is a new transaction record (e.g. user_id 100001332 booked a listing_id 10001501, on day 2024-04-21, price 31.36) being created, add this information to the transaction table, and update the availability in the calendar table accordingly.

4.9.1. MySQL

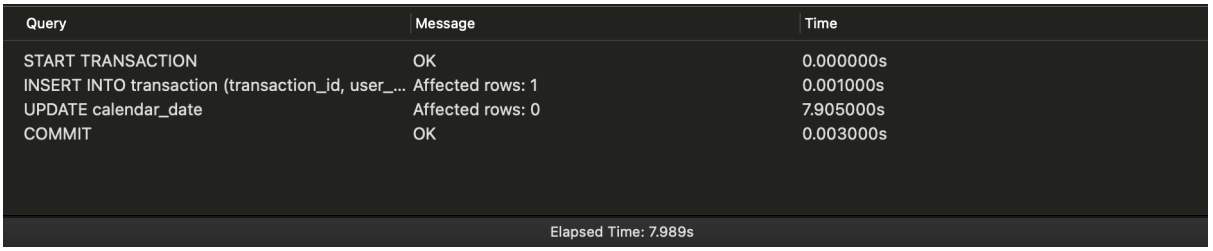
START TRANSACTION;

INSERT INTO transaction (transaction_id, user_id, listing_id, booking_date, check_in_date, price_per_night, total_price)

```
VALUES (1234512345, 100001332, 10001507, CURRENT_DATE(), '2024-04-21', 31.36,
31.36);

UPDATE calendar_date
SET available = 0
WHERE listing_id = 10001501 AND date = '2024-04-21';

COMMIT;
```



Query	Message	Time
START TRANSACTION	OK	0.000000s
INSERT INTO transaction (transaction_id, user_...	Affected rows: 1	0.001000s
UPDATE calendar_date	Affected rows: 0	7.905000s
COMMIT	OK	0.003000s

Elapsed Time: 7.989s

4.9.2. MongoDB

```
db.transaction.insert(
        {
          "transaction_id": 1234512345,
          "user_id": 100001332,
          "listing_id": 10001507,
          "booking_date": "21/04/2023",
          "check_in_date": "21/04/2023",
          "price_per_night": 31.36,
          "total_price": 31.36
        }
);
```

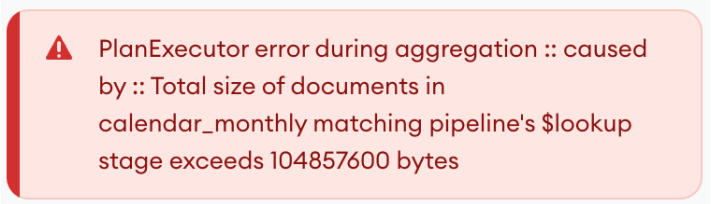
5. Comparison between two database systems

The performance of MongoDB queries compared to SQL (Structured Query Language) queries can vary based on several factors. Here are some reasons why MongoDB queries might be slower than SQL in certain scenarios:

Data Model: MongoDB stores data in BSON (a binary representation of JSON) documents, while MySQL has a more rigid schema requirement. This makes MongoDB more flexible and dynamic and particularly beneficial for handling diverse data. One such scenario is when we want to display all the reviews under a specific listing, MongoDB is able to tidy them in key-value pairs that are easy to analyze.

Query Complexity: MongoDB queries can sometimes be more verbose or complex, especially when dealing with nested documents or arrays. This can lead to slower performance compared to the relatively straightforward query language of SQL. This explains why we have difficulty in computing complex queries in MongoDB.

Aggregation Framework: MongoDB's aggregation framework is powerful but can be slower for complex operations compared to SQL's query capabilities, especially if the operations involve large datasets or complex data transformations.This is true as we encountered runtime error when generating aggregate function in MongoDB.



Joins: SQL databases are typically more efficient at handling joins due to their relational nature. MongoDB supports a form of join with the `$lookup` operator, but it might not be as efficient as SQL joins, particularly for large datasets or complex join conditions.

Caching Mechanisms: The effectiveness of caching mechanisms in the database can affect query performance. SQL databases might have more mature caching strategies compared to MongoDB, depending on the specific database system and configuration.

Data Distribution: In distributed database environments, how data is sharded or replicated can impact query performance. MongoDB's performance can be affected by the chosen shard key and the distribution of data across the cluster.

Read/Write Patterns: MongoDB might be optimized for certain types of read/write patterns, such as heavy write loads. In scenarios with different patterns, such as complex read operations, SQL databases might perform better.

6. Workload distribution

In the first section, each of us was in charge of one table to do the data cleaning and preprocessing. For the data storage section, Anqi and Lizheng were in charge of MySQL relational database while Hongpan and Zhuolin were in charge of MongoDB non-relational database. For the respective queries, Anqi and Lizheng were in charge of writing the query for the MySQL part while Hongpan and Zhuolin were in charge of writing the query for the MongoDB part. In the end, we compared and discussed the pros and cons for the two database systems together.