Big Data storage

AIRBNB

Simulating Airbnb's Database Queries with MongoDB

Group 2

21/05/2023

Stefano Sperti 20222246

Vojtech Kolomaznik 20222224

Pedro Bonifácio 20211652

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

ABSTRACT

MongoDB is a popular document-oriented database that has become the go-to choice for many modern web applications. In this project, we will be using a dataset from Airbnb, a popular online marketplace for vacation rentals, to simulate the queries that the website makes to its central database in a MongoDB format. Through this project, you will gain hands-on experience with MongoDB and learn how it can be used to store, retrieve, and manipulate large amounts of data in a highly scalable and efficient manner. You will also gain a deeper understanding of the architecture and capabilities of MongoDB, which will be valuable in developing robust and performant web applications.

KEYWORDS

Big Data storage, Airbnb, MongoDB

CONTENTS

Introduction 2
MongoDB in the business process 3
Description of the raw dataset 4
MongoDB import 6
Validation of the Database 7
Query in an Airbnb database 9
Story 1: A student 10
Story 2: A married couple 11
Story 3 Big group 13
Optimization of the Query 14
Index 1: Composite index on price and number of beds. 14
Index 2: guest_satisfaction_overall 14
Index 3: Bedrooms 14
Aggregation Query 16
Analysis 1: Comparing host and super-host performances 16
Analysis 2 The differences in average price in the city 17
Analysis 3 Differences in average price between weekends and days of the week 19
Analysis 4 Price distribution impact on rental characteristics and guest satisfaction 19
Analysis 5: Average price by room type 21
References 23

INTRODUCTION

Airbnb is a company founded in 2008, that focuses on providing short and long-term rentals as a service through their website. Data is central to their business model - it contains information about properties, helps determine prices or prevent fraud. The company has experienced dramatic growth, going from 1.5 million available properties in 2016 to 6.6 million properties in 2022. This has put a significant strain on the underlying technical infrastructure, which supports the storage, movement and processing of data. This project will go into how deploying a MongoDB instance may be compatible with the business process of the company and provide readers with relevant information and know-how. We will examine the process of importing the data into MongoDB, validating and verifying it, and optimizing its performance. We will also cover basic and advanced queries in MongoDB, including aggregation queries, and demonstrate how these can be used to extract useful information from the Airbnb dataset.

MONGODB IN THE BUSINESS PROCESS

In today's digital age, companies have access to a wealth of data that can be leveraged to make better business decisions. This is especially true in the hospitality industry, where companies like Airbnb have collected a massive amount of data on people's holiday habits and accommodation preferences. With 1.5 petabytes of data at their disposal, Airbnb's biggest challenge is connecting guests with the right accommodation. This is where big data and MongoDB technology come into play. Big data refers to the vast amount of structured and unstructured data that is generated and collected every day. This data can be analyzed to identify patterns, trends, and insights that can be used to make better decisions. In the case of Airbnb, big data is used to determine the appropriate price of a room or apartment, based on a number of variables such as location, time of year, type of accommodation, transport links, fraud detection and analytics. Using this data, Airbnb created an algorithm called Aerosolve to help their hosts determine the right price for their offering, which is particularly challenging given the sheer range of accommodation available. In general, a datum refers to a piece of information or record of an action or event, typically reflecting a decision made by a person. By reconstructing the sequence of events that led to that decision, one can gain valuable insights into a person's preferences and tendencies. For instance, this can reveal which properties are more appealing to guests or which features are more useful or less useful to them. Such feedback can be a valuable resource for making informed decisions about community growth, product development, and resource allocation. In essence, this involves translating the customer's feedback into a language that is better suited for decision-making purposes. Given the large volume of data generated by this platform, there are several reasons why Airbnb would benefit from using a MongoDB database, including:

- Flexible schema design: Airbnb's business process involves managing a large number of property listings that have varying attributes and features. With MongoDB's flexible schema design, Airbnb can easily add or modify fields in their data models without having to perform complex schema migrations.
- Scalability: As Airbnb's user base grows, the volume of data generated by the platform increases significantly. MongoDB's ability to horizontally scale across multiple servers enables Airbnb to handle this growth in data volume and ensure that their platform remains performant.
- Real-time analytics: Airbnb needs to analyze user behavior and property data in real-time to provide personalized recommendations and pricing. MongoDB's support for real-time analytics and aggregation pipelines enables Airbnb to quickly query and analyze large volumes of data.
- Geographic queries: Airbnb's business process involves managing property listings in multiple locations around the world. MongoDB's support for geospatial queries and indexes enables Airbnb to efficiently perform location-based queries and provide location-based recommendations to its users.

One notable illustration of scalability is demonstrated by the ability to partition city information into separate collections. This approach allows for easy scalability in two scenarios: expanding the platform to new areas or sharding different areas of the same city. By leveraging MongoDB's sharding capabilities, additional collections can be effortlessly added and distributed across the cluster.

DESCRIPTION OF THE RAW DATASET

The dataset provides comprehensive information on Airbnb prices in various popular European cities. It includes features such as the total price of the listing, the type of room, the host's superhost status, and other factors such as whether the listing is for multiple rooms or business purposes. The dataset also includes information on the overall guest satisfaction rating, the number of bedrooms, and the distance from the city center, along with longitude and latitude coordinates for location identification. Overall, this dataset provides valuable insights into the determinants of Airbnb prices in popular European cities, making it a useful resource for researchers and analysts. The raw version of the dataset: https://www.kaggle.com/datasets/thedevastator/airbnb-prices-in-european-cities

realSum	This is the total price of the Airbnb listing that includes all fees and taxes.
room_type	This variable identifies the type of room offered by the host, which could be a private room, shared room, or an entire home/apartment.
room_shared	This variable is a binary indicator that shows if the room offered by the host is shared or not. It is derived from the room_type variable.
room_private	This variable is a binary indicator that shows if the room offered by the host is private or not. It is derived from the room_type variable.
person_capacity	This variable indicates the maximum number of guests that can be accommodated in the Airbnb listing.
host_is_superhost	This variable is a binary indicator that shows if the host is a superhost or not. A superhost is an experienced host who provides exceptional hospitality to guests.
multi	This variable is a binary indicator that shows if the listing is for multiple rooms or not.
biz	This variable is a binary indicator that shows if the listing is used for business purposes or not.
cleanliness_rating	This variable indicates the cleanliness rating of the Airbnb listing based on guest feedback.

guest_satisfaction_overall	This variable represents the overall rating of the Airbnb listing based on guest feedback. It is an average of all the individual ratings given by guests.
bedrooms	This variable indicates the number of bedrooms in the Airbnb listing.
dist	This variable represents the distance from the Airbnb listing to the city center.
metro_dist	This variable represents the distance from the Airbnb listing to the nearest metro station.
attr_index	This variable is a composite index of various tourist attractions near the Airbnb listing.
attr_index_norm	This variable is the normalized version of the attr_index variable.
rest_index	This variable is a composite index of various restaurants and cafes near the Airbnb listing.
rest_index_norm	This variable is the normalized version of the rest_index variable.
lng	This variable represents the longitude of the location of the Airbnb listing.
lat	This variable represents the latitude of the location of the Airbnb listing.
weekend	This is a Boolean cell that indicates whether a feature corresponds to the price for a weekend (True) or the price for the week (False).

In particular, our dataset is divided in ten collections that represent ten different cities:

- Amsterdam
- Athens
- Barcelona
- Berlin

- Budapest
- Lisbon
- London
- Paris
- Rome
- Vienna

MONGODB IMPORT

We stored the database in the folder dump that was created after creating in the memory saving of the database. I establish a connection to the server via the shell and proceed to upload the database utilizing the command:

.\mongorestore.exe --uri
“mongodb+srv://<username>:<password>@<account>.<ID_computer>.mongodb.net"

This command specifies the Uniform Resource Identifier (URI) for connecting to the MongoDB server where the database will be uploaded. It also includes the username and password for the account required to access the server.

VALIDATION OF THE DATABASE

These validation rules help identify potential data issues or inconsistencies by searching for specific conditions or ranges of values in different fields. The count of matching documents provides insights into the data quality or compliance with the defined rules. These validation rules have been applied for all the collections in our database.

Validation rule 1: "guest_satisfaction_overall" field needs to be an integer greater than 0 or less than 100.

Validation rule 2: This rule specifies that the "multi" field is either 0 or 1.

Validation rule 3: "lng" (longitude) needs to be a float and needs to have a value greater than -180 or less than 180. It ensures that the longitude values fall within a valid range.

Validation rule 4: "lat" (latitude) needs to be a float and needs to have a value greater r than -90 or less than -90. It ensures that the latitude values fall within a valid range.

Validation rule 5: Validate "weekend" is a Boolean.

Validation rule 6: Validate "realSum" needs to be a float and needs to have a value greater than zero.

Validation rule 7: Validate "person_capacity" needs to be a float and needs to have a value greater than zero.

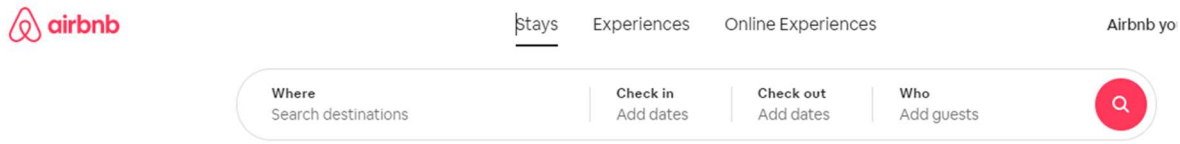
Validation rule 8: Validate "room_type" is a string and it is one of the valid options (private room, shared room, or entire home/apartment):

```
{
 $jsonSchema: {
  bsonType: 'object',
  required: [
   'guest_satisfaction_overall',
   'multi',
   'lng',
   'lat',
   'weekend',
   'realSum',
   'person_capacity',
   'room_type'
  ],
  properties: {
   guest_satisfaction_overall: {
    bsonType: 'int',
    minimum: 0,
    maximum: 100,
    description: 'Guest satisfaction has to be a number between 0 and 100.'
   },
   multi: {
```

```
      bsonType: 'int',
      'enum': [
        0,
        1
      ],
      description: 'Multi must be an either 0 or 1.'
    },
    lng: {
      bsonType: 'number',
      minimum: -180,
      maximum: 180,
      description: 'Longitude must be in [-180, 180].'
    },
    lat: {
      bsonType: 'number',
      minimum: -90,
      maximum: 90,
      description: 'Latitude must be in [-90, 90].'
    },
    weekend: {
      bsonType: 'bool',
      description: 'Weekend must be boolean.'
    },
    realSum: {
      bsonType: 'number',
      description: 'realSum must be greater than zero.'
    },
    person_capacity: {
      bsonType: 'number',
      minimum: 0,
      description: 'person_capacity must be greater than zero.'
    },
    room_type: {
      bsonType: 'string',
      'enum': [
        'Private room',
        'Shared room',
        'Entire home/apt'
      ],
      description: 'Room type must be \'Private room\', \'Shared room\', or \'Entire home/apt\'.'
    }
  }
}}
```

QUERY IN AN AIRBNB DATABASE

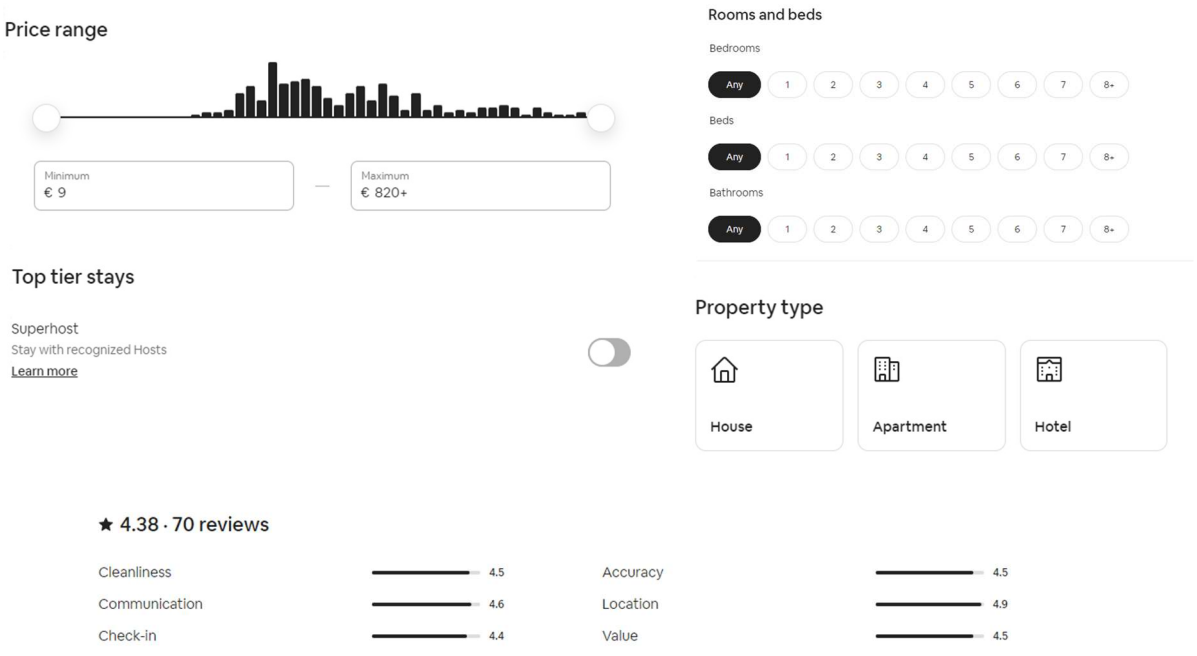
If you open the Airbnb website, you will see a user interface that allows you to search for and book vacation rentals in various locations around the world. The website typically presents a search bar at the top of the page where you can enter the destination, check-in and check-out dates, and the number of guests. Unfortunately, the check-in check out information is missing in our dataset since they are “private” information of the company. Therefore we assume a hypothetical situation in which all the queries that we make are in a period in which all the houses are available. We can only distinguish if they are searching for a reservation during the weekend or during the weekdays and the price is always the daily prices for 1 night of the period.

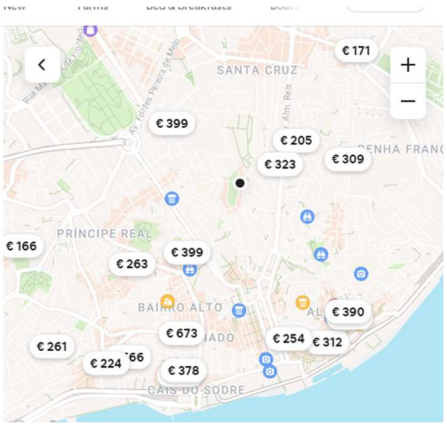


After entering your search criteria, you will be presented with a list of available rentals, along with information about the property, including photos, pricing, and amenities. You can then select a rental that meets your needs and proceed to book it through the website.

After the first research, you can add more requests using the additional filter that allows users to refine their search results based on specific criteria or preferences. After entering the initial search criteria, such as location and dates, the website presents a set of filters on the left-hand side of the screen.

These filters can include options such as price range, property type (e.g. apartment, house, villa), number of bedrooms, and locations. Users can select one or more filters to narrow down their search results and find a rental that meets their specific needs. In the next figures you can see the additional filter in Airbnb website.





All those information are present in our dataset. In particular, the “Destination” is the name of the collection in our database. It is the most important information in the query and for this reason gives the name to the collection for improving efficiency. Each property is a mongoDB document and all the other information will result in additional fields in that document.

STORY 1: A STUDENT

Alex decides to find the perfect Airbnb accommodation for their group of friends (in total 4) for some days during the week in Lisbon. In total they have a maximum of 120 euro for each night and they want to find the best solution.

Query 1: Enter in the website and search for a room for 4 people for weekdays in Lisbon.

```
db.lisbon.find({
  "person_capacity": { $gte: 4 },
  "weekend": false
})
```

Query 2: In their quest for affordability, Alex decided to fix the maximum price to 120 and sort the listings by the "smallest price."

```
db.lisbon.find(
  {
    "person_capacity": { $gte: 4 },
    "weekend": false,
    "realSum": { $lte: 120 },
  },
  {
    "room_shared": 1,
    "room_type": 1,
    "realSum":1
  }
).sort({ "realSum": 1 }).limit(3)
```

Query 3: He discovered that many of the rooms had shared rooms and therefore they decided to include the options to only include private rooms.

```
db.lisbon.find(
 {
  "person_capacity": { $gte: 4 },
  "weekend": false,
  "realSum": { $lte: 120 },
  "room_shared": false,
 },
 {
  "realSum": 1,
  "dist": 1,
  "room_type": 1,
  "dist" :1
 }
).sort({ realSum: 1 }).limit(3)
```

Query 4: While refining their search criteria, Alex realized that they could reduce their budget to 110 euros they decided to prioritize listings that were nearest to the city center.

```
db.lisbon.find({
 person_capacity: { $gte: 4 },
 weekend: false,
 realSum: { $lte: 110 },
 room_shared: false,
})
```

STORY 2: A MARRIED COUPLE

A couple is searching for a room for a weekend in Amsterdam.

Query 1:

```
db.amsterdam.find({
 "weekend": true,
 "person_capacity": { $gte: 2 }
})
```

Query 2: Since they don't have any budget constraints, they decide to search for a house in the city center that offers a private room.

```
db.amsterdam.find({
 "weekend": true,
  "room_shared": false,
```

```
  "person_capacity": { $gte: 2 }
},
 {
   "realSum": 1,
   "dist": 1,
    "room_shared": 1,
    "guest_satisfaction_overall" : 1,
    "cleanliness_rating":1
 }).sort({ "dist" : 1 }).limit(3)
```

Query 3: Additionally, they prioritize finding accommodations with a good rating, particularly focusing on a high cleanliness_rating.

```
db.amsterdam.find({
  "weekend": true,
   "room_shared": false,
  "person_capacity": { $gte: 2 },
   "cleanliness_rating": { $gte: 9 },
   "guest_satisfaction_overall" : { $gte: 85 }
},
 {
   "realSum": 1,
   "dist": 1,
    "room_shared": 1,
    "guest_satisfaction_overall" :1,
    "cleanliness_rating":1
 }).sort({ "dist" : 1 }).limit(5)
```

Query 4: After refining their search criteria and considering their preferences, they ultimately decide to choose the cheapest option that meets their requirements because they still want to be mindful of their expenses.

```
db.amsterdam.find({
  "weekend": true,
   "room_shared": false,
  "person_capacity": { $gte: 2 },
   "cleanliness_rating": { $gte: 9 },
   "guest_satisfaction_overall" : { $gte: 85 },
   "dist" : { $lte: 1 }
},
 {
   "realSum": 1,
   "dist": 1,
   "guest_satisfaction_overall" :1,
   "cleanliness_rating":1
 }).sort({"realSum": 1}).limit(1)
```

STORY 3 BIG GROUP

A big group is attending an event for several days during the week in Vienna. There are 6 members in the group, and they prefer to stay together in the same place.

Query 1: Enter in the website

```
db.vienna.find({
  "weekend": false,
  "person_capacity": { $gte: 6 }
})
```

Query 2: Their primary concern is to minimize expenses, so they are not particular about the city itself. However, they prioritize being close to the metro for convenience.

```
db.vienna.find({
"person_capacity": { $gte: 6 },
"weekend": false
}, { "metro_dist": 1, "realSum": 1 }).sort({ "realSum": 1 }).limit(5)
```

Query 3: After setting their budget to 180 euro for each night, they decide that they prefer not to stay in a shared room and require a minimum of 2 bedrooms in the accommodation.

```
db.vienna.find({
"person_capacity": { $gte: 6 },
"weekend": false,
"room_shared": false,
"bedrooms": { $gte: 2 },
"realSum": { $lte: 180}
}, { "metro_dist": 1, "realSum": 1, "guest_satisfaction_overall": 1 }).sort({
"guest_satisfaction_overall": -1 }).limit(5)
```

Query 4: Ultimately, they choose the cheapest of the listing with the guest satisfaction rating over 80%.

```
db.vienna.find({
"person_capacity": { $gte: 6 },
"weekend": false,
"room_shared": false,
"bedrooms": { $gte: 2 },
"guest_satisfaction_overall": { $gte: 80},
}, { "metro_dist": 1, "realSum": 1, "guest_satisfaction_overall": 1 }).sort({ "realSum": 1 }).limit(1)
```

OPTIMIZATION OF THE QUERY

INDEX 1: COMPOSITE INDEX ON PRICE AND NUMBER OF BEDS.

When looking for an apartment, most people filter by the number of beds and price. This will make the search faster. We use the city of Amsterdam as example.

db.amsterdam.createIndex({"realSum":1, "person_capacity": 1})

To test this index, we used a query that finds a cheap overnight apartment for two people.

db.amsterdam.find({"realSum": {$gte:80 , $lte: 130}, "person_capacity": 2})

Using .explain("executionStats") we can see that we have to examine 2080 documents without the index. With the index, we only examine 2 keys and one document. We don't have to unnecessarily search a large amount of documents.

INDEX 2: GUEST_SATISFACTION_OVERALL

When loading available apartments for a tiled view, guest satisfaction overall is visible (as it helps guests decide which place to book). Higher satisfaction score is preferred. We have created a reverse ordered index to help with loading the data faster. We use the city of Amsterdam as example.

db.amsterdam.createIndex({"guest_satisfaction_overall":-1})

To test this index, we used this query that filters well rated accommodation, but avoids one hundred percent guest satisfaction (as that may indicate that the apartment was only booked once).

db.amsterdam.find({"guest_satisfaction_overall": {$gte: 90, $lte: 95}})

Looking at the execution statistics, we can see that we have to examine 2080 documents without the index. With the index, we only have to examine 610 documents, and the execution time dropped by one millisecond. This will save a lot of time when the database becomes larger and holds more records.

INDEX 3: BEDROOMS

Sometimes, a very large group is looking for accommodation. This does not happen very often, but it is useful for them to instantly find the properties with the largest number of bedrooms. For this instance, we have created a reverse ordered index for bedrooms. We use the city of London as example.

db.london.createIndex({"bedrooms": -1})

To test this index, we used a query that finds apartments with capacity of 6 or more people.

db.london.find({"person_capacity": {$gte:6}})

Looking at the execution statistics, we can see that implementing the index restricted the amount of examined documents from 9993 to 682, and the execution time dropped from 7 to just 2 milliseconds. With the scaling of the data, this will save a lot of time.

AGGREGATION QUERY

With its flexibility and adaptability, the Aggregation Framework empowers data analysts to perform complex analytical tasks tailored to the specific requirements of their business, ultimately extracting valuable information from Airbnb data.

ANALYSIS 1: COMPARING HOST AND SUPER-HOST PERFORMANCES

Through a comparison of metrics between normal hosts and super hosts, Airbnb gains insights into the influence of superhost status on pricing, cleanliness, and guest satisfaction. This analysis helps identify areas where super hosts excel, such as higher guest satisfaction and superior cleanliness ratings. The findings can be used to promote the advantages of becoming a super host, motivating hosts to provide outstanding hospitality and enhance the overall Airbnb experience.

Aggregation query #1: which analyze the differences in price, cleanliness, and guest satisfaction between normal and super hosts in the city - center of Lisbon:

```
db.lisbon.aggregate([
 {
        $match: {
        dist: { $lte: 2 }
        }},
 {
        $group: {
        _id: "$host_is_superhost",
        avgRealSum: { $avg: "$realSum" },
        avgCleanlinessRating: { $avg: "$cleanliness_rating" },
        avgGuestSatisfactionOverall: { $avg: "$guest_satisfaction_overall" },
        count: { $sum: 1 }}}])
```

```
> {_id: false,
  avgRealSum: 246.17622974790896,
  avgCleanlinessRating: 9.239122987324427,
  avgGuestSatisfactionOverall: 89.78622816032887,
  count: 2919}

{_id: true,
  avgRealSum: 278.1615375827691,
  avgCleanlinessRating: 9.853629976580796,
  avgGuestSatisfactionOverall: 95.96604215456675,
  count: 854}
```

Notes: although there is an increase in average price from hosts to superhosts, there is also the corresponding positive effect in both ratings, which suggests that, in this case, the superhost label is accurate.

Airbnb decide to find if all the superhost continue to achieve the minimum satisfaction rating or there is some of them that must demote:

Aggregation query #2: finding the worst rated superhost in Lisbon:

```
db.lisbon.aggregate([
 {$match: {
 host_is_superhost: true}},
 {$sort: { guest_satisfaction_overall: 1 }},
 {$limit: 1 },
 {$project: {
 _id: 0,
 realSum: 1,
 room_type: 1,
 guest_satisfaction_overall: 1,
 cleanliness_rating: 1}}])
```

```
> { realSum: 299.953095684803,
 room_type: 'Entire home/apt',
 cleanliness_rating: 4,
 guest_satisfaction_overall: 20}
```

We can observe that the aggregate could do the same as find. Both the aggregation pipeline and the find query achieve the same result of filtering the documents where host_is_superhost is true, sorting them by guest_satisfaction_overall in ascending order, limiting the result to one document, and projecting only the specified fields.

```
db.lisbon.find(
 { "host_is_superhost": true },
 { "realSum": 1, "room_type": 1, "guest_satisfaction_overall": 1, "cleanliness_rating": 1 }
).sort({ "guest_satisfaction_overall": 1 }).limit(1)
```

Notes: this query shows us that Airbnb might have to demote some superhosts. Let's use the following query to see how many low rated superhosts there are in Lisbon:

```
db.lisbon.countDocuments({
 host_is_superhost: true,
 $or: [
        { guest_satisfaction_overall: { $lt: 70 } },
        { cleanliness_rating: { $lt: 7 } }
]})
```

```
> 5
```

Conclusion: As we only have 5 superhosts with overall rating lower than 70 or cleanliness rating lower than 7, we think Airbnb can afford to analyze them case-by-case, maybe contacting the hosts before making a decision.

ANALYSIS 2 THE DIFFERENCES IN AVERAGE PRICE IN THE CITY

Airbnb decides to analyze average prices in the city of Amsterdam because it aims to identify popular neighborhoods or areas where demand consistently surpasses supply. By understanding these

market dynamics, Airbnb can uncover potential opportunities for expansion or focus on targeted marketing efforts in these high-demand areas.

Aggregation query #3: find the average price of a private room, for 2 persons, in the center of Amsterdam and Rome and compare:

```
db.amsterdam.aggregate([
 {$match: {
        room_type: "Private room",
        person_capacity: 2,
        dist: { $lt: 2 }}},
 {$group: {
        _id: null,
        avgRealSum: { $avg: "$realSum" }}}])

> {_id: null,
 avgRealSum: 406.08091554591687}
```

We now know what's the average price, let's find how many private rooms exist with price lower than 400, using

```
db.amsterdam.countDocuments({
 room_type: "Private room",
 dist: { $lt: 2 },
 person_capacity: 2,
 realSum: { $lt: 400}
})

> 166
```

We repeat the same analysis in Rome:

```
db.rome.aggregate([
 {$match: {
        room_type: "Private room",
        person_capacity: 2,
        dist: { $lt: 2 }}},
 {$group: {
        _id: null,
        avgRealSum: { $avg: "$realSum" }}}])

Result:
{
 _id: null,
 avgRealSum: 157.69711714435016
}
```

Conclusion: there is a huge difference in price between cities.

ANALYSIS 3 DIFFERENCES IN AVERAGE PRICE BETWEEN WEEKENDS AND DAYS OF THE WEEK

Airbnb decides to conduct an analysis of price variations between weekends and weekdays because it aims to enhance its pricing strategies. By understanding these differences, the platform can provide hosts in the city of Paris with more accurate pricing recommendations tailored to specific days of the week.

Aggregation query #4:

```
db.paris.aggregate([
 { $group: {
    _id: "$weekend",
    avgPrice: { $avg: "$realSum" }
 }}])
```

We can observe that the average during the weekends is 398 and during the week is 387, therefore there are a few increases in price during the weekends.

ANALYSIS 4 PRICE DISTRIBUTION IMPACT ON RENTAL CHARACTERISTICS AND GUEST SATISFACTION

Airbnb aims to verify whether an increase in prices corresponds to improved performance or if it is merely speculation. To achieve this, they are considering implementing the following query in the city of Lisbon:

Aggregation query #5: using the $bucket operator, we categorized the documents into four groups based on the price of the rental, with the goal of comparing the distribution of various fields.

The four groups: [0,200[, [200, 400[, [400, 600[and [600, +inf[

```
db.lisbon.aggregate([
 {
  $bucket: {
   groupBy: "$realSum",
   boundaries: [0, 200, 400, 600],
   default: "600+",
   output: {
          count: { $sum: 1 },
          average_person_capacity: { $avg: "$person_capacity" },
          cleanliness_rating: { $avg: "$cleanliness_rating" },
          guest_satisfaction_overall: { $avg: "$guest_satisfaction_overall" },
          bedrooms: { $avg: "$bedrooms" },
         dist: { $avg: "$dist" },
         metro_dist: { $avg: "$metro_dist" }
   }}}])
```

```
>{
  _id: 0,
  count: 2228,
  average_person_capacity: 2.4483842010771992,
  cleanliness_rating: 9.225314183123878,
  guest_satisfaction_overall: 89.64497307001795,
  bedrooms: 1.0309694793536803,
  dist: 2.3188775259743486,
  metro_dist: 0.6635906298727461
}
{
  _id: 200,
  count: 3126,
  average_person_capacity: 3.775111964171465,
  cleanliness_rating: 9.437619961612285,
  guest_satisfaction_overall: 91.75111964171465,
  bedrooms: 1.3464491362763915,
  dist: 1.745283141570629,
  metro_dist: 0.751450245969198
}
{
  _id: 400,
  count: 353,
  average_person_capacity: 4.895184135977337,
  cleanliness_rating: 9.628895184135978,
  guest_satisfaction_overall: 93.8300283286119,
  bedrooms: 2.0169971671388103,
  dist: 1.7333043674550623,
  metro_dist: 0.6842246924979049
}
{
  _id: '600+',
  count: 56,
  average_person_capacity: 5.071428571428571,
  cleanliness_rating: 9.785714285714286,
  guest_satisfaction_overall: 94.80357142857143,
  bedrooms: 2.0535714285714284,
  dist: 1.805946819346038,
  metro_dist: 0.5576016771114136
}
```


Conclusion: Exploring the results of the aggregation query, it is evident that as the rental price increases, there is a corresponding increase in various rental characteristics. Higher-priced rentals exhibit higher average person capacity, more bedrooms, better cleanliness ratings, and overall guest satisfaction. However, the distance to the city center and metro does not show a significant correlation with price. These findings suggest that increasing prices align with improved rental attributes and guest satisfaction, highlighting the value associated with higher-priced accommodations in Lisbon.

References:

https://www.mongodb.com/docs/manual/reference/operator/aggregation/bucket/

ANALYSIS 5: AVERAGE PRICE BY ROOM TYPE

Airbnb decides to conduct an average price analysis by room type in the city of Budapest because it aims to gain insights into pricing patterns and preferences within the dataset. This analysis involves calculating the average prices for private rooms, shared rooms, and entire homes/apartments, allowing Airbnb to understand the variations in pricing and identify the preferences of its users in Budapest.

Aggregation query #6:

```
db.budapest.aggregate([
 {
  $group: {
    _id: "$room_type",
    avgPrice: { $avg: "$realSum" }
}}])
```

Result:

```
>> {  _id: 'Entire home/apt',

 avgPrice: 184.57318020904665}

{  _id: 'Private room',

 avgPrice: 109.13775931813383}

{  _id: 'Shared room',

 avgPrice: 126.83044736435724}
```

Conclusion: The analysis of average prices by room type reveals distinct pricing patterns within the dataset. The results indicate that "Entire home/apt" listings have the highest average price at 184.57, followed by "Shared room" listings at 126.83, and "Private room" listings at 109.14. These findings suggest that the type of accommodation significantly influences the pricing structure, with entire homes/apartments generally commanding higher prices compared to shared or private rooms.

CONCLUSION

In this project, we exploited the capabilities of MongoDB, a widely used document-oriented database, by working with a dataset from Airbnb, a renowned vacation rental marketplace. Through hands-on experience, we learned how MongoDB can efficiently store, retrieve, and manipulate large volumes of data, making it an ideal choice for modern web applications.

We began by understanding the significance of having a non-relational database - like MongoDB in the business process, followed by a detailed description of the raw dataset of Airbnb listings obtained from Kaggle. We then proceeded to implement the database using MongoDB, ensuring the validation of the database structure to maintain data integrity implementing 8 validation rules.

With the database in place, we delved into querying the Airbnb database, exploring different scenarios through engaging stories involving a student, a married couple, and a big group. These stories allowed us to witness the power of MongoDB in providing flexible and efficient query capabilities to meet diverse application needs. We have performed 15 queries along the entire project.

To enhance performance, we explored query optimization techniques, ensuring efficient retrieval of data. Those optimization techniques were tested on criteria such as speed or memory usage, and they outperformed the original database. Additionally, we utilized aggregation queries to perform advanced analysis on the dataset. We examined the performance differences between hosts and superhosts, analyzed variations in average prices across the city, explored price differences between weekends and weekdays, and investigated the impact of price on rental characteristics and guest satisfaction. Finally, we scrutinized the average prices based on room types, uncovering insights into pricing patterns for different accommodation options. For doing that we have implemented 6 different aggregation queries.

Overall, this project has demonstrated that MongoDB can be useful for powering Big Data solutions, which may be of interest to both academics and commercial companies.

REFERENCES

Marr, B. (2016). Big data in practice: how 45 successful companies used big data analytics to deliver extraordinary results. John Wiley & Sons.

Thomas Erl, with Wajid Khattak, Paul Buhler (2016), Big Data Fundamentals Concepts, Drivers Techniques-Prentice Hall

Reference MongoDB:
https://www.mongodb.com/docs/manual/reference/operator/aggregation/bucket/

Sitography

- http://nerds.airbnb.com/scaling-data-science/
- http://thenewstack.io/airbnbs-airpal-reflects-new-ways-to-query-andget-answers-from-hive-and-hadoop/
- http://www.washingtonpost.com/news/wonkblog/wp/2015/08/27/wifihot-tubs-and-big-data-how-airbnb-determines-the-price-of-a-home/
- https://www.mongodb.com/docs/manual/reference/operator/aggregation/bucket/

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa