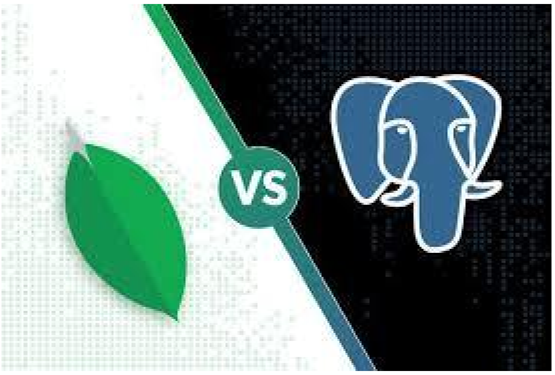
“Ss. Cyril and Methodius” University in Skopje
FACULTY OF COMPUTER
SCIENCE AND ENGINEERING

Comparative Analysis of Performance Metrics of NoSQL and Relational Databases



Author:
Angela Mladenovska

June 2024

Table of Contents

1. Introduction

Data is an extremely significant component in the modern world. Various conclusions are drawn from their manipulation and analysis. They are arranged and kept in a database to ensure effective organization and productive work. Although there are many other kinds of databases, relational and non-relational databases are the most well-known.

The foundation of relational databases is the tabular arrangement of data, which consists of rows that display individual entries and columns that represent features that are present in the table. Only structured data is managed by this kind of database. It is possible to establish relationships that correspond between several tables. Relationships can be formed between different tables, as the name depicts. Most well known relational database management systems are MySQL, Oracle, Microsoft SQL Server, and PostgreSQL. PostgreSQL's performances will be actually taken into consideration throughout this project.

On the other hand, non-relational databases or NoSQL have a different approach from the relational ones: they do not have a classic tabular structure. They can be graph-based, document-based, key-value and column family [1]. The non-relational database management system MongoDB, which stores data in JSON format, a representation of a document-oriented database will be introduced to us in this project.

To clarify, document-oriented databases offer efficient data management, mainly in a semi-structured format.

Different types of databases produce different results when running the same queries. In this project assignment we will compare the timing performance of PostgreSQL and MongoDB.

2. Dataset Selection and Analysis

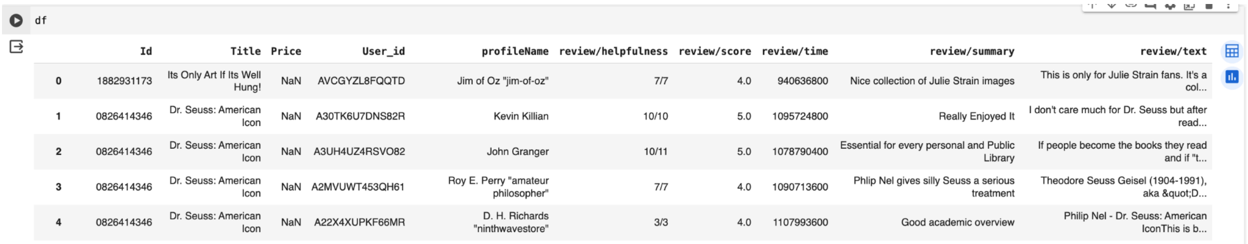
For the purpose of this project, secondary data analysis[2] is performed, which is analysis of data that has been collected from various sources for a particular purpose.

This process consists of several steps: defining a research topic, selecting a dataset, reviewing the selected dataset. We will later discuss the steps that were taken in detail.

It was necessary to select a larger dataset, with several hundreds to a million records. On the link[3] where the selected dataset is located, there are two datasets: books_data.csv and books_rating.csv. For our purposes, we only use the books_rating dataset.

The Books rating dataset contains about 3 million records and consists of 10 columns: Id, Title, Price, User_id, profileName, review/helpfulness, review/score, review/time, review/summary, review/text.

After the selection was defined, the dataset was reviewed, and since it initially contained null values and unnecessarily formatted values, it needed to be processed and analyzed accordingly. The data analysis and data preparation was performed in the Google Colab tool.

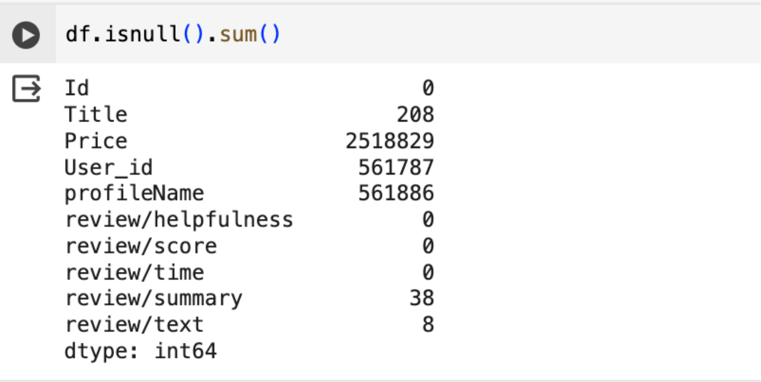


```
df
```

	Id	Title	Price	User_id	profileName	review/helpfulness	review/score	review/time	review/summary	review/text
0	1882931173	Its Only Art If Its Well Hung!	NaN	AVCGYZL8FQQTD	Jim of Oz "jim-of-oz"	7/7	4.0	940636800	Nice collection of Julie Strain images	This is only for Julie Strain fans. It's a col...
1	0826414346	Dr. Seuss: American Icon	NaN	A30TK6U7DNS82R	Kevin Killian	10/10	5.0	1095724800	Really Enjoyed It	I don't care much for Dr. Seuss but after read...
2	0826414346	Dr. Seuss: American Icon	NaN	A3UH4UZ4RSVO82	John Granger	10/11	5.0	1078790400	Essential for every personal and Public Library	If people become the books they read and if "t...
3	0826414346	Dr. Seuss: American Icon	NaN	A2MVUWT453QH61	Roy E. Perry "amateur philosopher"	7/7	4.0	1090713600	Phlip Nel gives silly Seuss a serious treatment	Theodore Seuss Geisel (1904-1991), aka &quot;D...
4	0826414346	Dr. Seuss: American Icon	NaN	A22X4XUPKF66MR	D. H. Richards "ninthwavestore"	3/3	4.0	1107993600	Good academic overview	Philip Nel - Dr. Seuss: American IconThis is b...

Picture 1: Initial Overview of the Dataset

Additionally, a check was made on how many null (nan/null) values exist in each of the columns. This was done in order to properly handle them, because in the general case such values would be unnecessary.



```
df.isnull().sum()
```

```
Id                          0
Title                     208
Price                 2518829
User_id                561787
profileName            561886
review/helpfulness          0
review/score                0
review/time                 0
review/summary             38
review/text                 8
dtype: int64
```

Picture 2: Overview of the presence of null values in the dataset

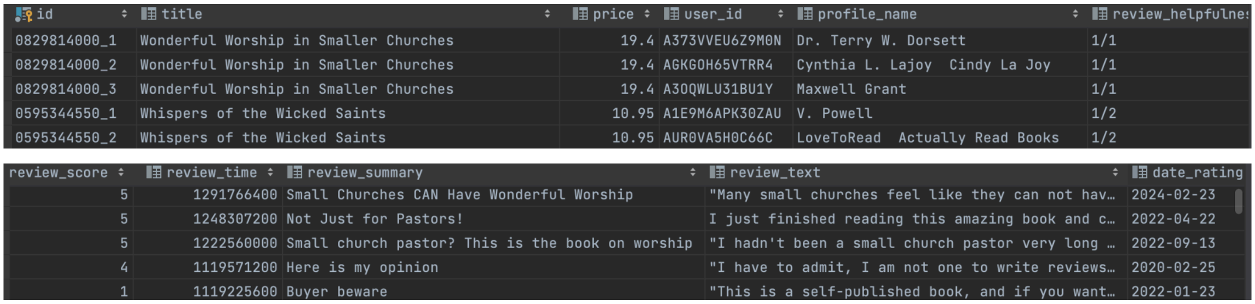
A specific method for handling the null values is selected for each column. For the Title, review/summary, review/text and Price columns we dropped the rows where the values are missing. Null values in the User_id and profileName columns are replaced by the value 'Unknown'.
In order to maintain consistency when importing the data, those values from the columns Title, profileName, review/summary, review/text, which contained the value ';'(semicolon), were deleted, because the separator when saving the dataset after its processing was done to be ';'. This is due to the presence of commas in the values represented in this dataset, and by default the comma is the separator in a csv file.
The values from the Id column were not unique, and in order for it to serve as the primary key in the database table, some changes were made to make those values unique.
Another change that has been made is the removal of unnecessary quotes on values from the profileName column.

Also, in order to have a richer choice for data manipulation, a new date_rated column has been added that represents the dates when each book was rated. The column is filled with random dates using Python functions.
After completing the data cleaning, 385 902 records remained in the dataset.

3. Creating Relational and Aggregation Models for Data Importing

3.1 Relational model

For the relational model, the DataGrip IDE was used, mainly because of the friendly user interface that it offers, for data manipulation. A new Postgres database was created in PgAdmin, where a table was generated for importing the dataset. In DataGrip, a connection was made with the database.

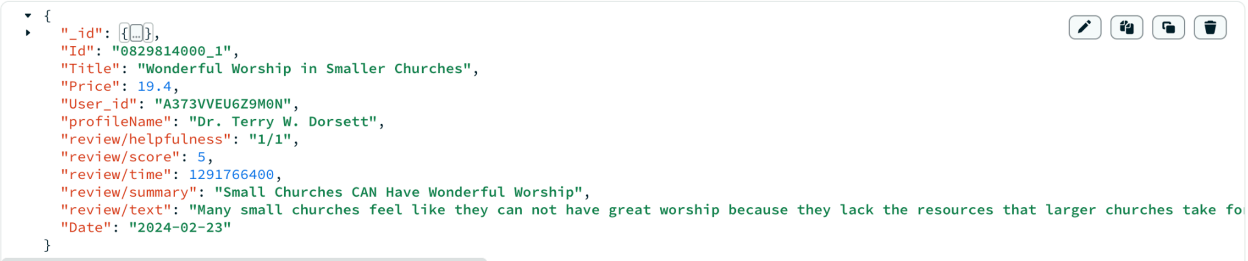


id	title	price	user_id	profile_name	review_helpfulnes	review_score	review_time	review_summary	review_text	date_rating
0829814000_1	Wonderful Worship in Smaller Churches	19.4	A373VVEU6Z9M0N	Dr. Terry W. Dorsett	1/1	5	1291766400	Small Churches CAN Have Wonderful Worship	"Many small churches feel like they can not hav…	2024-02-23
0829814000_2	Wonderful Worship in Smaller Churches	19.4	AGKGOH65VTRR4	Cynthia L. Lajoy Cindy La Joy	1/1	5	1248307200	Not Just for Pastors!	I just finished reading this amazing book and c…	2022-04-22
0829814000_3	Wonderful Worship in Smaller Churches	19.4	A3OQWLU31BU1Y	Maxwell Grant	1/1	5	1222560000	Small church pastor? This is the book on worship	"I hadn't been a small church pastor very long …	2022-09-13
0595344550_1	Whispers of the Wicked Saints	10.95	A1E9M6APK30ZAU	V. Powell	1/2	4	1119571200	Here is my opinion	"I have to admit, I am not one to write reviews…	2020-02-25
0595344550_2	Whispers of the Wicked Saints	10.95	AUR0VA5H0C66C	LoveToRead Actually Read Books	1/2	1	1119225600	Buyer beware	"This is a self-published book, and if you want…	2022-01-23

Picture 3: Overview of the imported dataset in DataGrip

3.2 Aggregation model

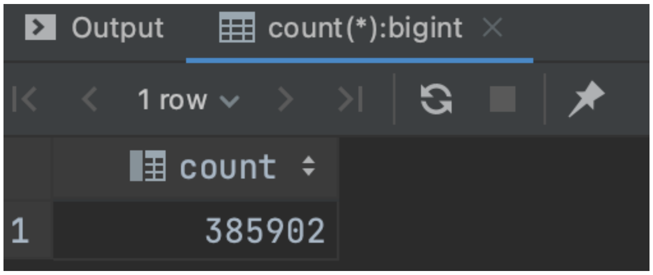
The MongoDB non-relational database management system was chosen for working with the non-relational database. MongoDB was installed locally on the machine, additionally MongoDBCompass IDE was installed where a connection was made with the database. The csv dataset was imported there as well, but its structure was modified in JSON format.
It was explored how MongoDB works, before we moved on to further work. [5]



```
{
  "_id": {…},
  "Id": "0829814000_1",
  "Title": "Wonderful Worship in Smaller Churches",
  "Price": 19.4,
  "User_id": "A373VVEU6Z9M0N",
  "profileName": "Dr. Terry W. Dorsett",
  "review/helpfulness": "1/1",
  "review/score": 5,
  "review/time": 1291766400,
  "review/summary": "Small Churches CAN Have Wonderful Worship",
  "review/text": "Many small churches feel like they can not have great worship because they lack the resources that larger churches take fo
  "Date": "2024-02-23"
}
```

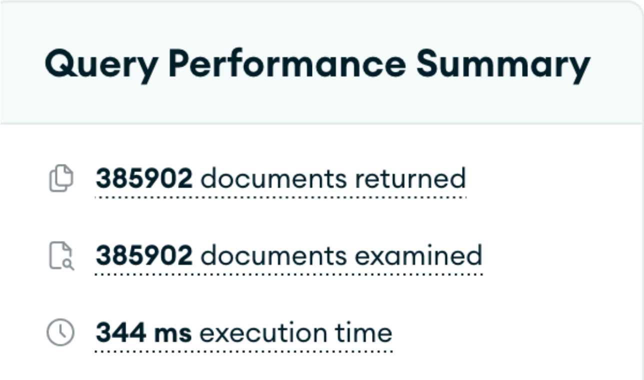
Picture 4: Overview of a single JSON object in MongoDBCompass

Also, a check was made whether the number of the records is equal in both of the databases.



	count
1	385902

Picture 5: Total number of rows in the relational model



Query Performance Summary

385902 documents returned

385902 documents examined

344 ms execution time

Picture 6: Total number of records in NoSQL

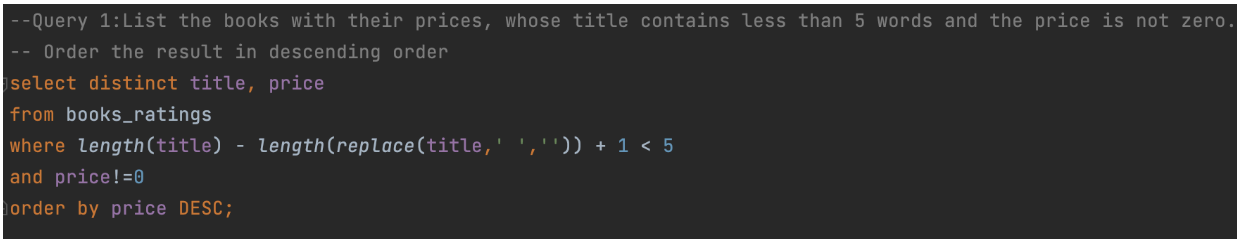
With this, we ensured that the dataset was imported appropriately. This indicates that there were not any issues during the importing of the datasets.

4. Defining queries

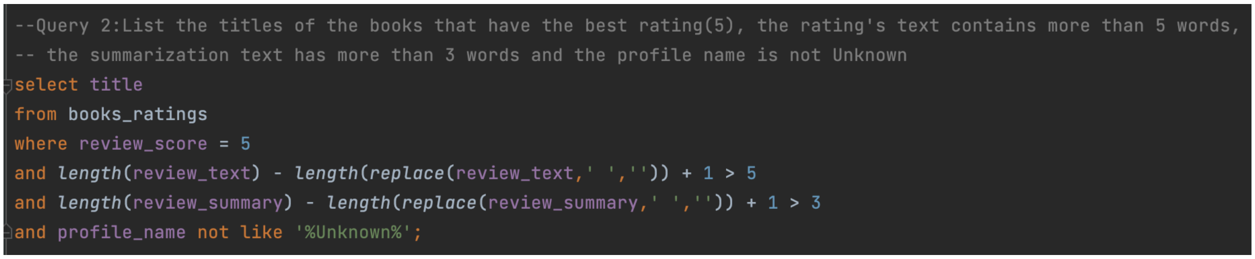
After the models were created, the next step was to define queries of various types: aggregation, filtering, retrieving.

4.1 Queries in PostgreSQL

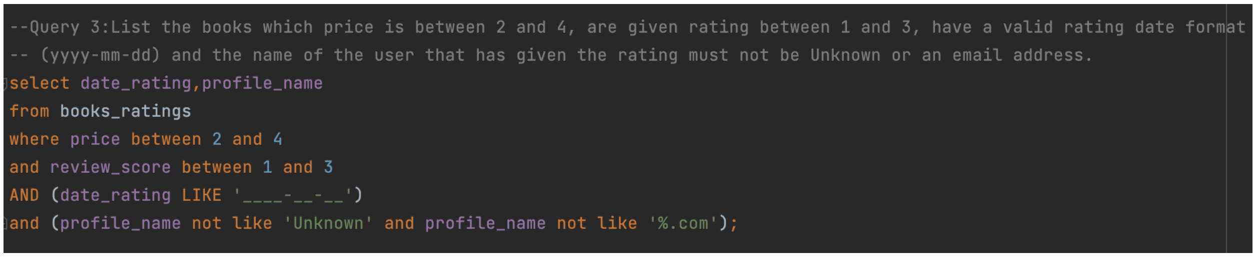
After the successful importing of the data in the database, 10 queries were written which purpose is to be executed in the database. Pictures of the queries can be found in this section, as well as description of the query can be found in the comments on the pictures.



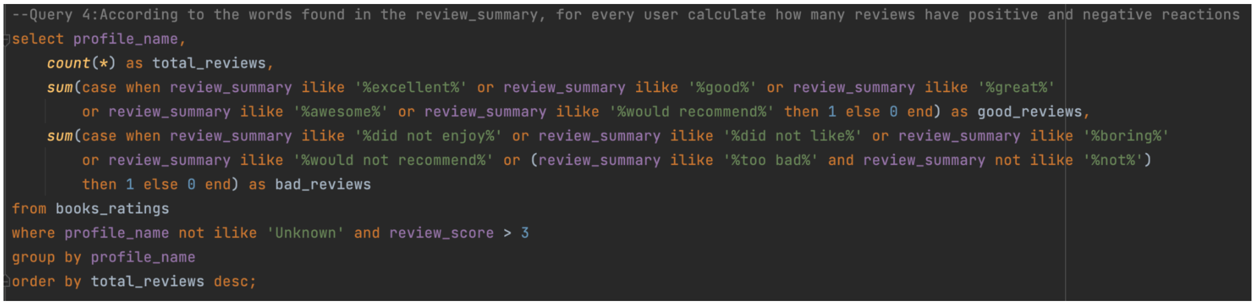
```
--Query 1:List the books with their prices, whose title contains less than 5 words and the price is not zero.
-- Order the result in descending order
select distinct title, price
from books_ratings
where length(title) - length(replace(title,' ','')) + 1 < 5
and price!=0
order by price DESC;
```



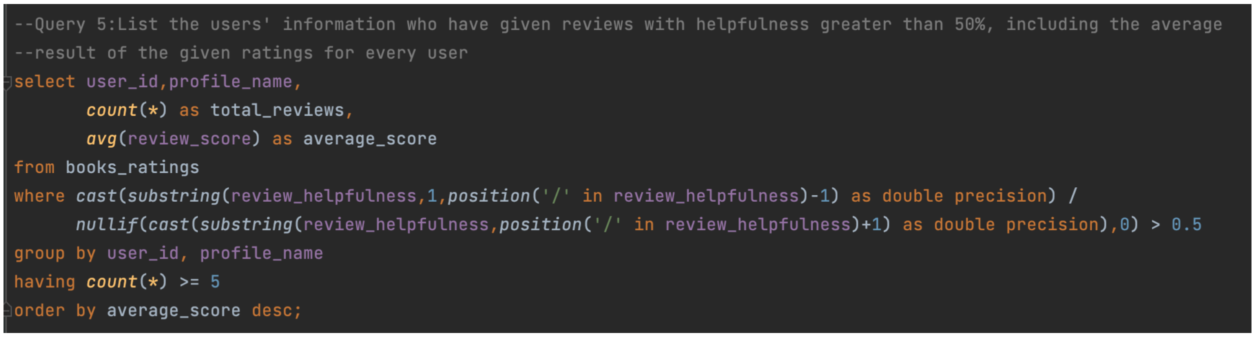
```
--Query 2:List the titles of the books that have the best rating(5), the rating's text contains more than 5 words,
-- the summarization text has more than 3 words and the profile name is not Unknown
select title
from books_ratings
where review_score = 5
and length(review_text) - length(replace(review_text,' ','')) + 1 > 5
and length(review_summary) - length(replace(review_summary,' ','')) + 1 > 3
and profile_name not like '%Unknown%';
```



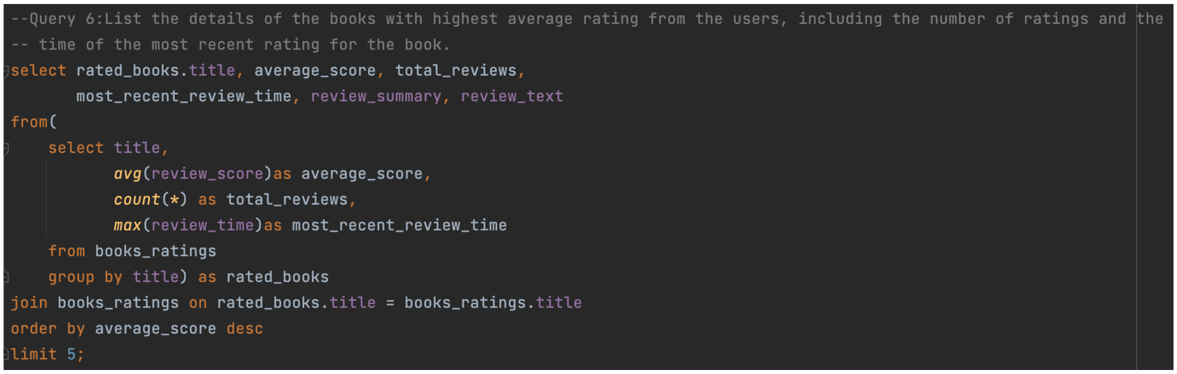
```
--Query 3:List the books which price is between 2 and 4, are given rating between 1 and 3, have a valid rating date format
-- (yyyy-mm-dd) and the name of the user that has given the rating must not be Unknown or an email address.
select date_rating,profile_name
from books_ratings
where price between 2 and 4
and review_score between 1 and 3
AND (date_rating LIKE '____-__-__')
and (profile_name not like 'Unknown' and profile_name not like '%.com');
```



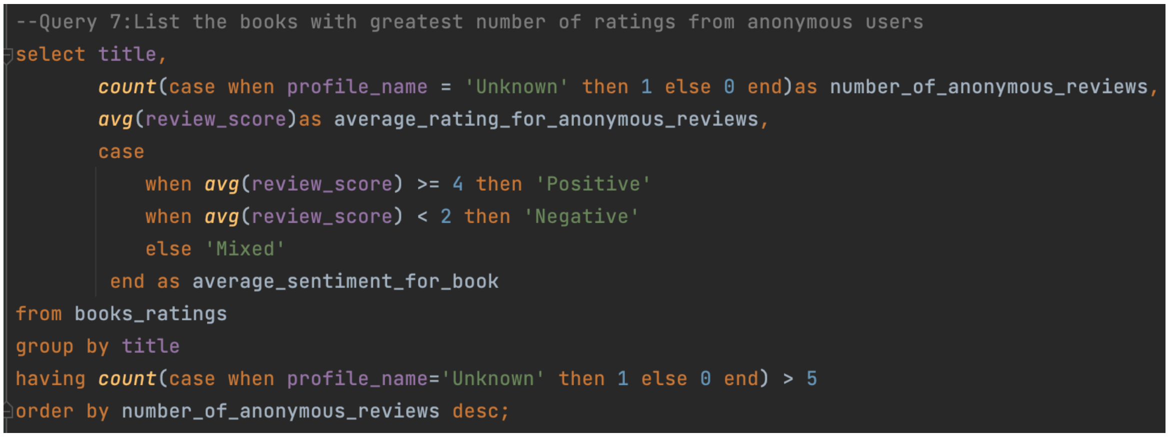
```
--Query 4:According to the words found in the review_summary, for every user calculate how many reviews have positive and negative reactions
select profile_name,
    count(*) as total_reviews,
    sum(case when review_summary ilike '%excellent%' or review_summary ilike '%good%' or review_summary ilike '%great%'
        or review_summary ilike '%awesome%' or review_summary ilike '%would recommend%' then 1 else 0 end) as good_reviews,
    sum(case when review_summary ilike '%did not enjoy%' or review_summary ilike '%did not like%' or review_summary ilike '%boring%'
        or review_summary ilike '%would not recommend%' or (review_summary ilike '%too bad%' and review_summary not ilike '%not%')
        then 1 else 0 end) as bad_reviews
from books_ratings
where profile_name not ilike 'Unknown' and review_score > 3
group by profile_name
order by total_reviews desc;
```



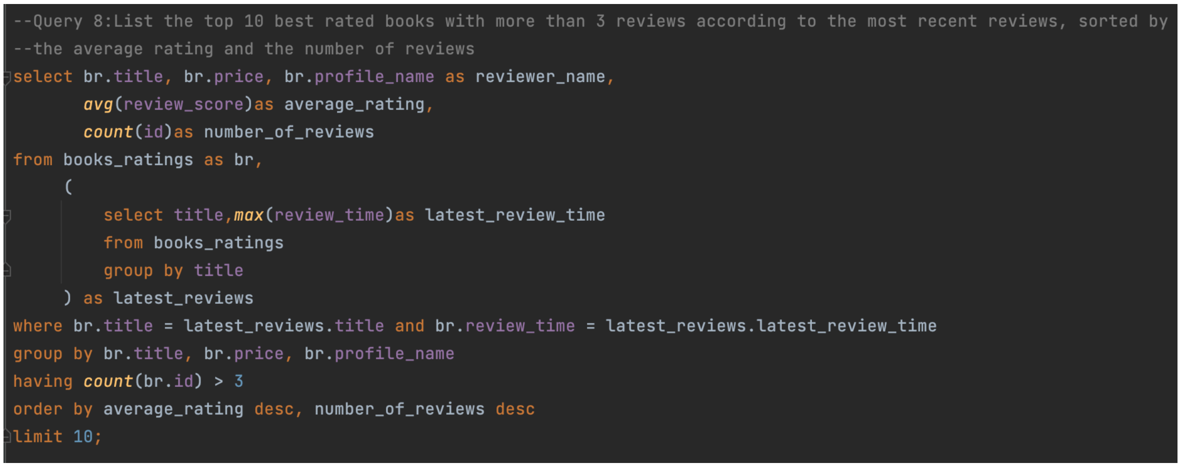
```
--Query 5:List the users' information who have given reviews with helpfulness greater than 50%, including the average
--result of the given ratings for every user
select user_id,profile_name,
       count(*) as total_reviews,
       avg(review_score) as average_score
from books_ratings
where cast(substring(review_helpfulness,1,position('/' in review_helpfulness)-1) as double precision) /
      nullif(cast(substring(review_helpfulness,position('/' in review_helpfulness)+1) as double precision),0) > 0.5
group by user_id, profile_name
having count(*) >= 5
order by average_score desc;
```



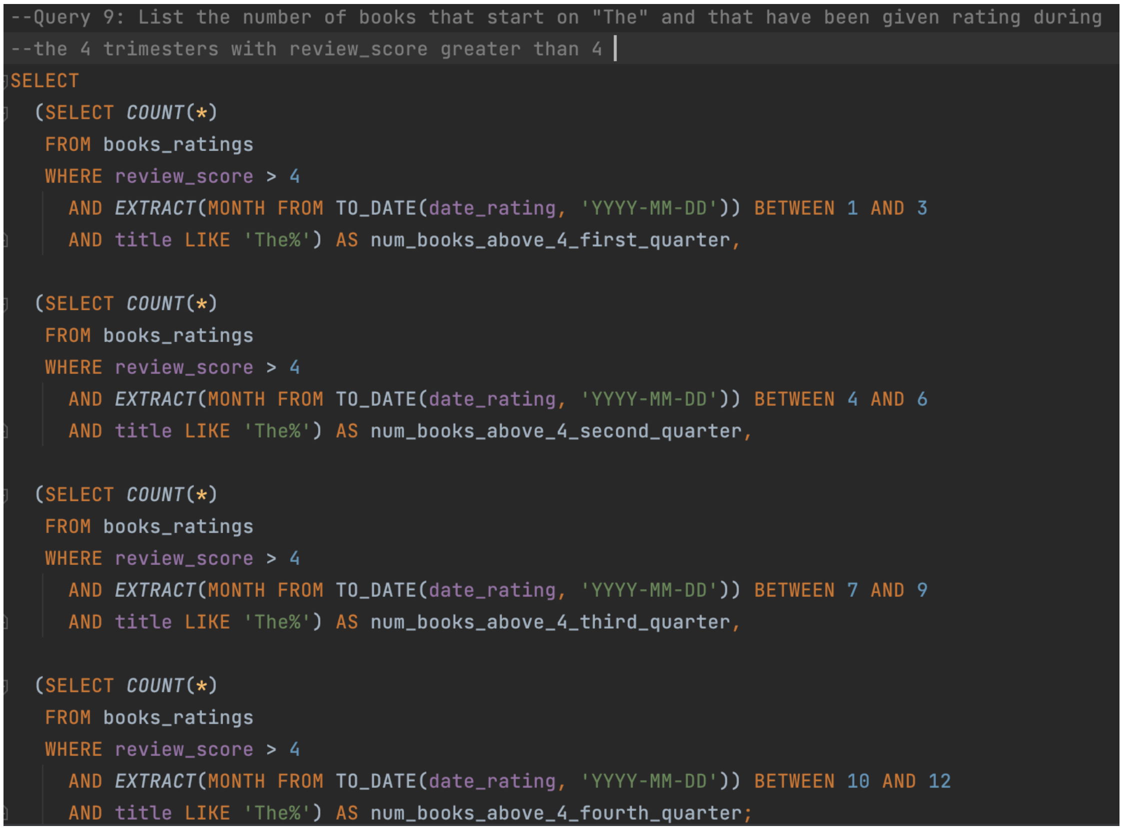
```
--Query 6:List the details of the books with highest average rating from the users, including the number of ratings and the
-- time of the most recent rating for the book.
select rated_books.title, average_score, total_reviews,
       most_recent_review_time, review_summary, review_text
from(
    select title,
           avg(review_score)as average_score,
           count(*) as total_reviews,
           max(review_time)as most_recent_review_time
    from books_ratings
    group by title) as rated_books
join books_ratings on rated_books.title = books_ratings.title
order by average_score desc
limit 5;
```



```
--Query 7:List the books with greatest number of ratings from anonymous users
select title,
       count(case when profile_name = 'Unknown' then 1 else 0 end)as number_of_anonymous_reviews,
       avg(review_score)as average_rating_for_anonymous_reviews,
       case
           when avg(review_score) >= 4 then 'Positive'
           when avg(review_score) < 2 then 'Negative'
           else 'Mixed'
        end as average_sentiment_for_book
from books_ratings
group by title
having count(case when profile_name='Unknown' then 1 else 0 end) > 5
order by number_of_anonymous_reviews desc;
```



```
--Query 8:List the top 10 best rated books with more than 3 reviews according to the most recent reviews, sorted by
--the average rating and the number of reviews
select br.title, br.price, br.profile_name as reviewer_name,
       avg(review_score)as average_rating,
       count(id)as number_of_reviews
from books_ratings as br,
     (
         select title,max(review_time)as latest_review_time
         from books_ratings
         group by title
     ) as latest_reviews
where br.title = latest_reviews.title and br.review_time = latest_reviews.latest_review_time
group by br.title, br.price, br.profile_name
having count(br.id) > 3
order by average_rating desc, number_of_reviews desc
limit 10;
```

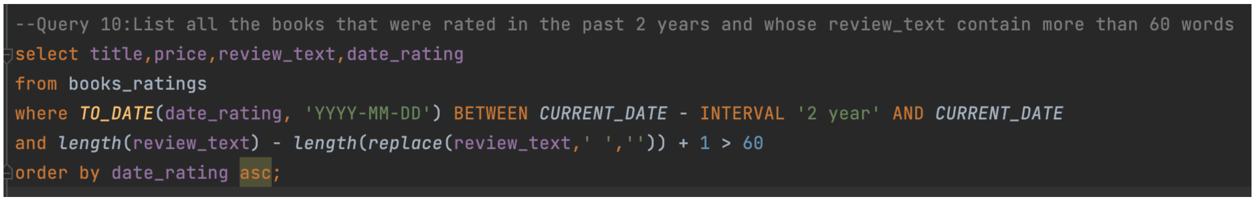


```sql
--Query 9: List the number of books that start on "The" and that have been given rating during
--the 4 trimesters with review_score greater than 4
SELECT
  (SELECT COUNT(*)
   FROM books_ratings
   WHERE review_score > 4
     AND EXTRACT(MONTH FROM TO_DATE(date_rating, 'YYYY-MM-DD')) BETWEEN 1 AND 3
     AND title LIKE 'The%') AS num_books_above_4_first_quarter,

  (SELECT COUNT(*)
   FROM books_ratings
   WHERE review_score > 4
     AND EXTRACT(MONTH FROM TO_DATE(date_rating, 'YYYY-MM-DD')) BETWEEN 4 AND 6
     AND title LIKE 'The%') AS num_books_above_4_second_quarter,

  (SELECT COUNT(*)
   FROM books_ratings
   WHERE review_score > 4
     AND EXTRACT(MONTH FROM TO_DATE(date_rating, 'YYYY-MM-DD')) BETWEEN 7 AND 9
     AND title LIKE 'The%') AS num_books_above_4_third_quarter,

  (SELECT COUNT(*)
   FROM books_ratings
   WHERE review_score > 4
     AND EXTRACT(MONTH FROM TO_DATE(date_rating, 'YYYY-MM-DD')) BETWEEN 10 AND 12
     AND title LIKE 'The%') AS num_books_above_4_fourth_quarter;
```

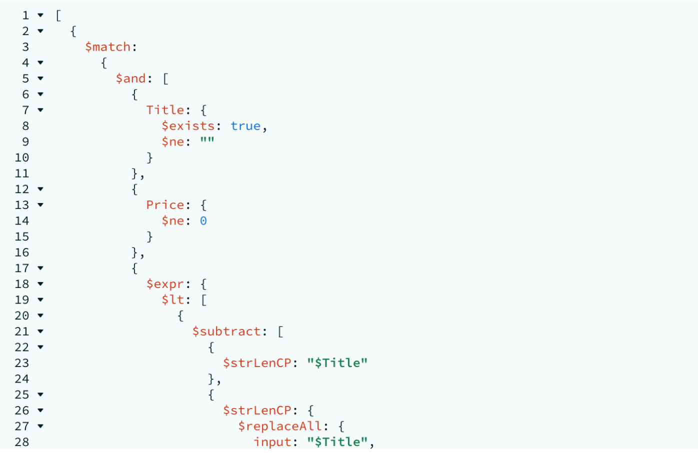


```sql
--Query 10:List all the books that were rated in the past 2 years and whose review_text contain more than 60 words
select title,price,review_text,date_rating
from books_ratings
where TO_DATE(date_rating, 'YYYY-MM-DD') BETWEEN CURRENT_DATE - INTERVAL '2 year' AND CURRENT_DATE
and length(review_text) - length(replace(review_text,' ','')) + 1 > 60
order by date_rating asc;
```

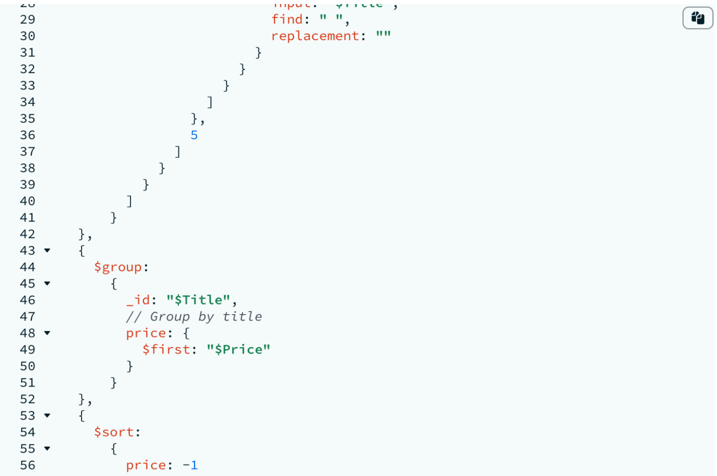
4.2 Queries in MongoDB

The same queries were written for MongoDB. Pictures of the queries can be found in the following section.

Query 1

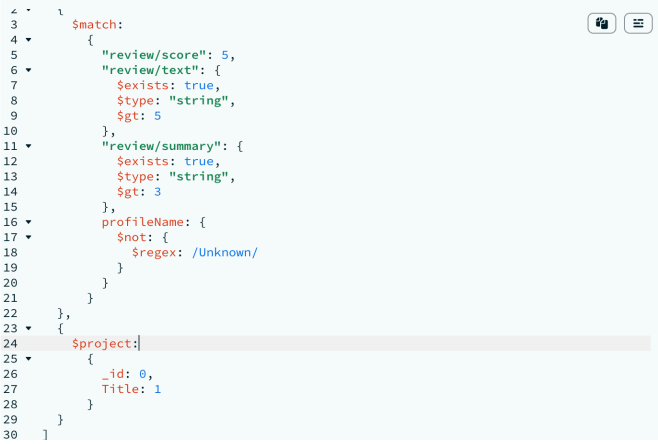


```
[
  {
    $match:
      {
        $and: [
          {
            Title: {
              $exists: true,
              $ne: ""
            }
          },
          {
            Price: {
              $ne: 0
            }
          },
          {
            $expr: {
              $lt: [
                {
                  $subtract: [
                    {
                      $strLenCP: "$Title"
                    },
                    {
                      $strLenCP: {
                        $replaceAll: {
                          input: "$Title",
```



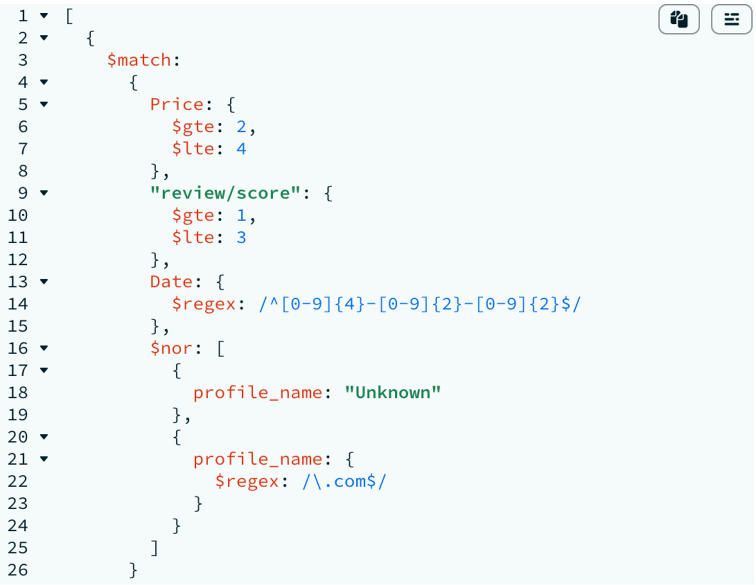
```
                          find: " ",
                          replacement: ""
                        }
                      }
                    }
                  ]
                },
                5
              ]
            }
          }
        ]
      }
  },
  {
    $group:
      {
        _id: "$Title",
        // Group by title
        price: {
          $first: "$Price"
        }
      }
  },
  {
    $sort:
      {
        price: -1
```

Query 2

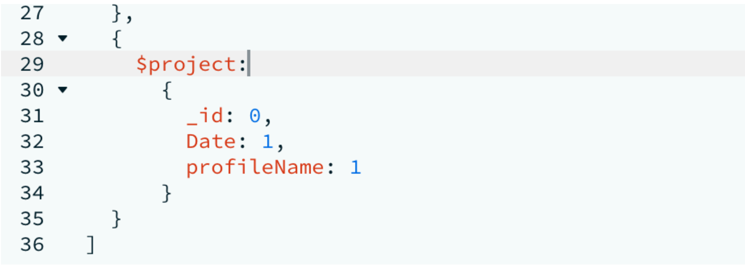


```
  {
    $match:
      {
        "review/score": 5,
        "review/text": {
          $exists: true,
          $type: "string",
          $gt: 5
        },
        "review/summary": {
          $exists: true,
          $type: "string",
          $gt: 3
        },
        profileName: {
          $not: {
            $regex: /Unknown/
          }
        }
      }
  },
  {
    $project:
      {
        _id: 0,
        Title: 1
      }
  }
]
```

Query 3

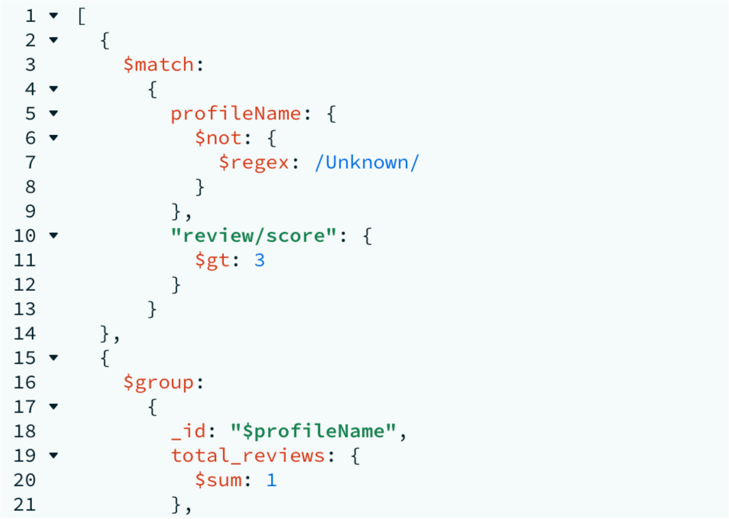


```
[
  {
    $match:
      {
        Price: {
          $gte: 2,
          $lte: 4
        },
        "review/score": {
          $gte: 1,
          $lte: 3
        },
        Date: {
          $regex: /^[0-9]{4}-[0-9]{2}-[0-9]{2}$/
        },
        $nor: [
          {
            profile_name: "Unknown"
          },
          {
            profile_name: {
              $regex: /\.com$/
            }
          }
        ]
      }
```

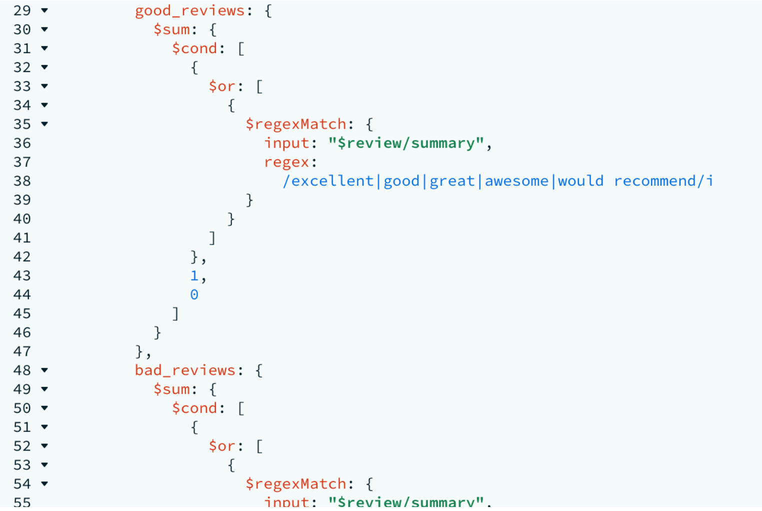


```
  },
  {
    $project:
      {
        _id: 0,
        Date: 1,
        profileName: 1
      }
  }
]
```

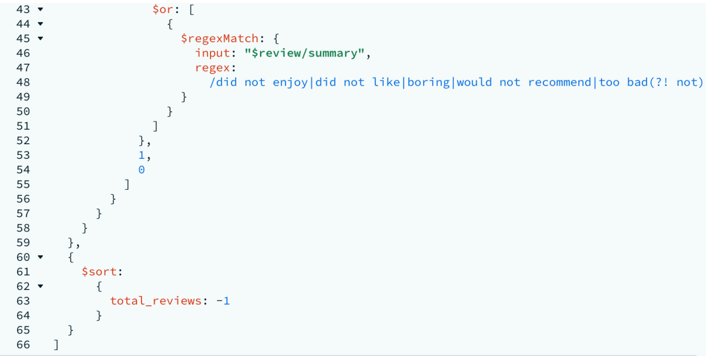
Query 4



```
[
  {
    $match:
      {
        profileName: {
          $not: {
            $regex: /Unknown/
          }
        },
        "review/score": {
          $gt: 3
        }
      }
  },
  {
    $group:
      {
        _id: "$profileName",
        total_reviews: {
          $sum: 1
        },
```

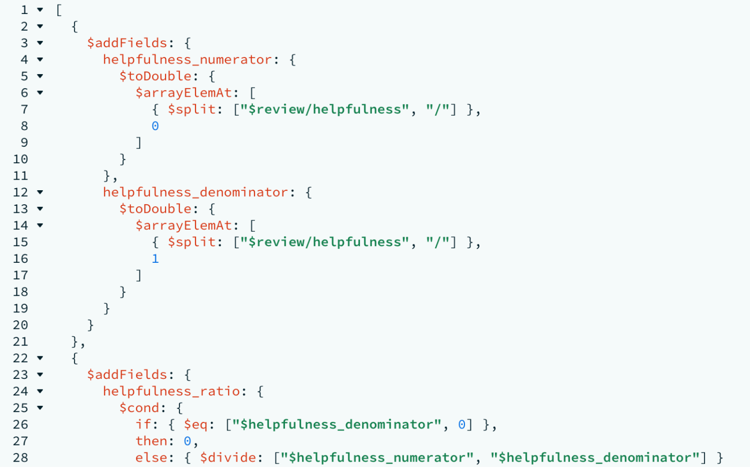


```
        good_reviews: {
          $sum: {
            $cond: [
              {
                $or: [
                  {
                    $regexMatch: {
                      input: "$review/summary",
                      regex:
                        /excellent|good|great|awesome|would recommend/i
                    }
                  }
                ]
              },
              1,
              0
            ]
          }
        },
        bad_reviews: {
          $sum: {
            $cond: [
              {
                $or: [
                  {
                    $regexMatch: {
                      input: "$review/summary",
```

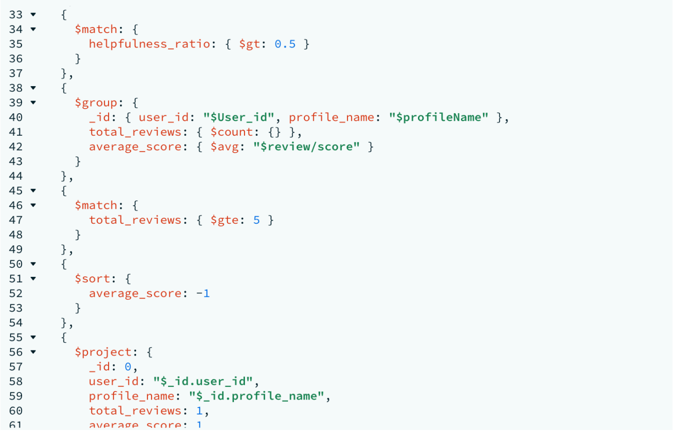


```
              $or: [
                {
                  $regexMatch: {
                    input: "$review/summary",
                    regex:
                      /did not enjoy|did not like|boring|would not recommend|too bad(?! not)
                  }
                }
              ]
            },
            1,
            0
          ]
        }
      }
    }
  },
  {
    $sort:
      {
        total_reviews: -1
      }
  }
]
```

Query 5

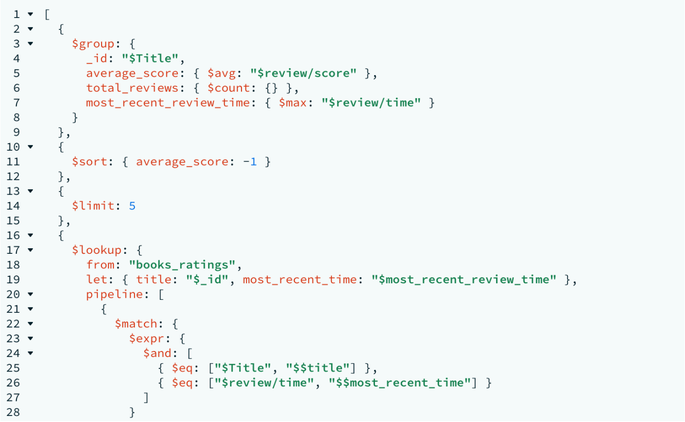


```
[
  {
    $addFields: {
      helpfulness_numerator: {
        $toDouble: {
          $arrayElemAt: [
            { $split: ["$review/helpfulness", "/"] },
            0
          ]
        }
      },
      helpfulness_denominator: {
        $toDouble: {
          $arrayElemAt: [
            { $split: ["$review/helpfulness", "/"] },
            1
          ]
        }
      }
    }
  },
  {
    $addFields: {
      helpfulness_ratio: {
        $cond: {
          if: { $eq: ["$helpfulness_denominator", 0] },
          then: 0,
          else: { $divide: ["$helpfulness_numerator", "$helpfulness_denominator"] }
```

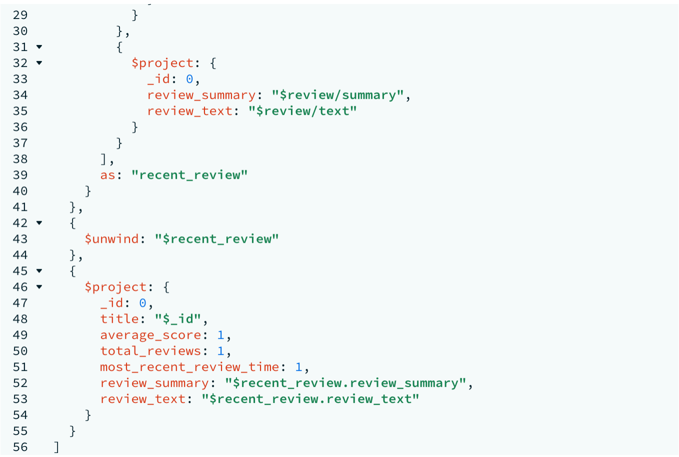


```
  {
    $match: {
      helpfulness_ratio: { $gt: 0.5 }
    }
  },
  {
    $group: {
      _id: { user_id: "$User_id", profile_name: "$profileName" },
      total_reviews: { $count: {} },
      average_score: { $avg: "$review/score" }
    }
  },
  {
    $match: {
      total_reviews: { $gte: 5 }
    }
  },
  {
    $sort: {
      average_score: -1
    }
  },
  {
    $project: {
      _id: 0,
      user_id: "$_id.user_id",
      profile_name: "$_id.profile_name",
      total_reviews: 1,
      average_score: 1
```

Query 6

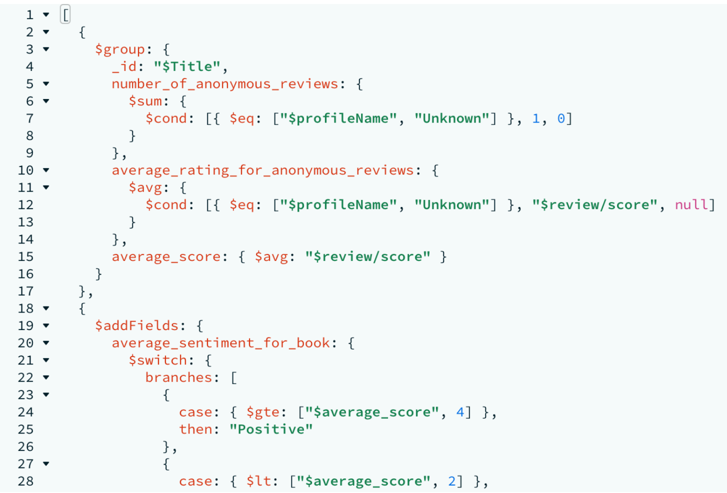


```
[
  {
    $group: {
      _id: "$Title",
      average_score: { $avg: "$review/score" },
      total_reviews: { $count: {} },
      most_recent_review_time: { $max: "$review/time" }
    }
  },
  {
    $sort: { average_score: -1 }
  },
  {
    $limit: 5
  },
  {
    $lookup: {
      from: "books_ratings",
      let: { title: "$_id", most_recent_time: "$most_recent_review_time" },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $eq: ["$Title", "$$title"] },
                { $eq: ["$review/time", "$$most_recent_time"] }
              ]
            }
```

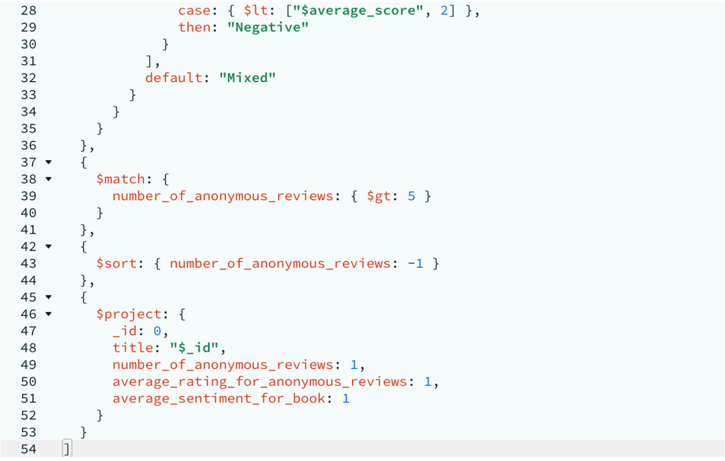


```
          }
        },
        {
          $project: {
            _id: 0,
            review_summary: "$review/summary",
            review_text: "$review/text"
          }
        }
      ],
      as: "recent_review"
    }
  },
  {
    $unwind: "$recent_review"
  },
  {
    $project: {
      _id: 0,
      title: "$_id",
      average_score: 1,
      total_reviews: 1,
      most_recent_review_time: 1,
      review_summary: "$recent_review.review_summary",
      review_text: "$recent_review.review_text"
    }
  }
]
```

Query 7

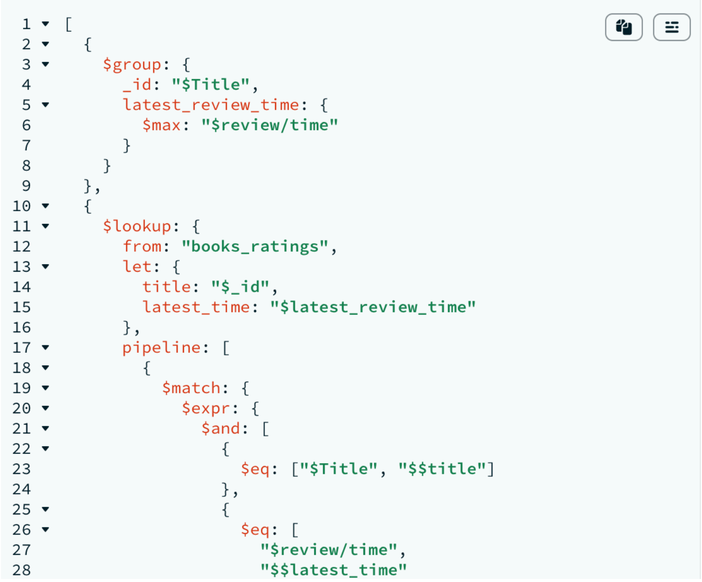


```
[
  {
    $group: {
      _id: "$Title",
      number_of_anonymous_reviews: {
        $sum: {
          $cond: [{ $eq: ["$profileName", "Unknown"] }, 1, 0]
        }
      },
      average_rating_for_anonymous_reviews: {
        $avg: {
          $cond: [{ $eq: ["$profileName", "Unknown"] }, "$review/score", null]
        }
      },
      average_score: { $avg: "$review/score" }
    }
  },
  {
    $addFields: {
      average_sentiment_for_book: {
        $switch: {
          branches: [
            {
              case: { $gte: ["$average_score", 4] },
              then: "Positive"
            },
            {
              case: { $lt: ["$average_score", 2] },
```

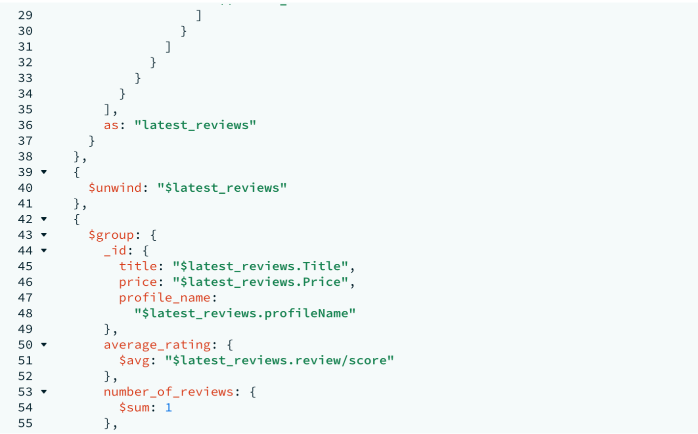


```
              case: { $lt: ["$average_score", 2] },
              then: "Negative"
            }
          ],
          default: "Mixed"
        }
      }
    }
  },
  {
    $match: {
      number_of_anonymous_reviews: { $gt: 5 }
    }
  },
  {
    $sort: { number_of_anonymous_reviews: -1 }
  },
  {
    $project: {
      _id: 0,
      title: "$_id",
      number_of_anonymous_reviews: 1,
      average_rating_for_anonymous_reviews: 1,
      average_sentiment_for_book: 1
    }
  }
]
```

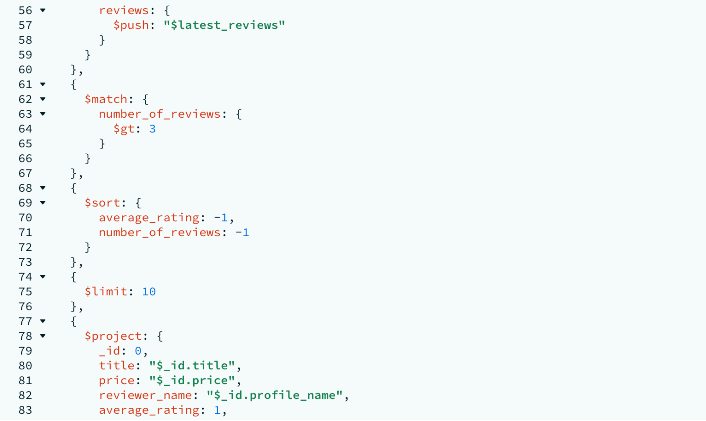
Query 8



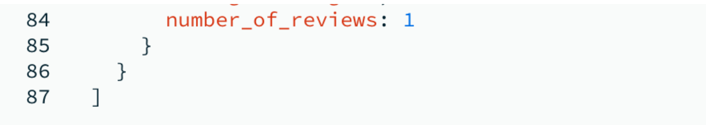
```
[
  {
    $group: {
      _id: "$Title",
      latest_review_time: {
        $max: "$review/time"
      }
    }
  },
  {
    $lookup: {
      from: "books_ratings",
      let: {
        title: "$_id",
        latest_time: "$latest_review_time"
      },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                {
                  $eq: ["$Title", "$$title"]
                },
                {
                  $eq: [
                    "$review/time",
                    "$$latest_time"
```



```
                  ]
                }
              ]
            }
          }
        }
      ],
      as: "latest_reviews"
    }
  },
  {
    $unwind: "$latest_reviews"
  },
  {
    $group: {
      _id: {
        title: "$latest_reviews.Title",
        price: "$latest_reviews.Price",
        profile_name:
          "$latest_reviews.profileName"
      },
      average_rating: {
        $avg: "$latest_reviews.review/score"
      },
      number_of_reviews: {
        $sum: 1
      },
```

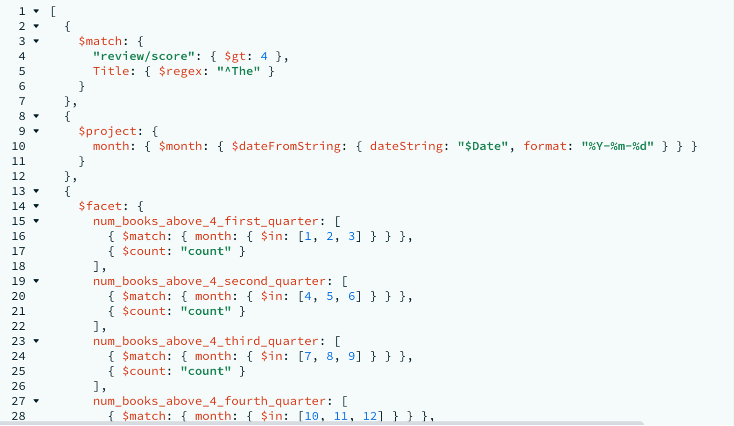


```
      reviews: {
        $push: "$latest_reviews"
      }
    }
  },
  {
    $match: {
      number_of_reviews: {
        $gt: 3
      }
    }
  },
  {
    $sort: {
      average_rating: -1,
      number_of_reviews: -1
    }
  },
  {
    $limit: 10
  },
  {
    $project: {
      _id: 0,
      title: "$_id.title",
      price: "$_id.price",
      reviewer_name: "$_id.profile_name",
      average_rating: 1,
```

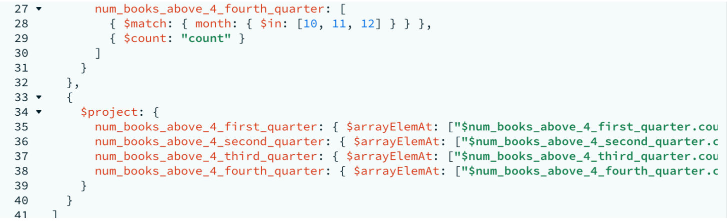


```
      number_of_reviews: 1
    }
  }
]
```

Query 9

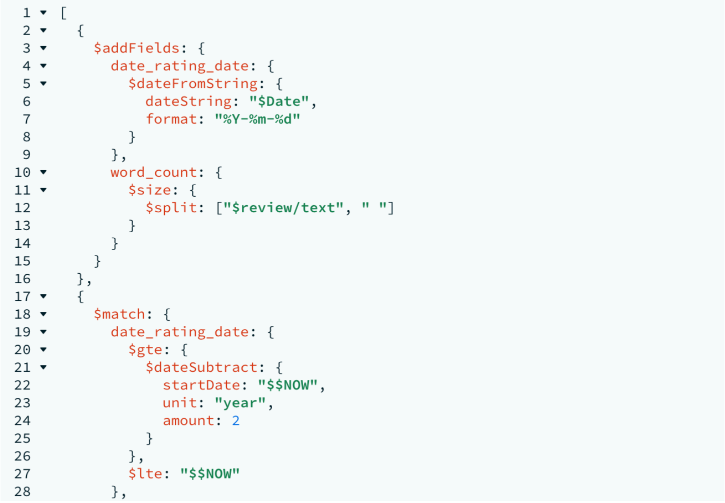


```
[
  {
    $match: {
      "review/score": { $gt: 4 },
      Title: { $regex: "^The" }
    }
  },
  {
    $project: {
      month: { $month: { $dateFromString: { dateString: "$Date", format: "%Y-%m-%d" } } }
    }
  },
  {
    $facet: {
      num_books_above_4_first_quarter: [
        { $match: { month: { $in: [1, 2, 3] } } },
        { $count: "count" }
      ],
      num_books_above_4_second_quarter: [
        { $match: { month: { $in: [4, 5, 6] } } },
        { $count: "count" }
      ],
      num_books_above_4_third_quarter: [
        { $match: { month: { $in: [7, 8, 9] } } },
        { $count: "count" }
      ],
      num_books_above_4_fourth_quarter: [
        { $match: { month: { $in: [10, 11, 12] } } },
```

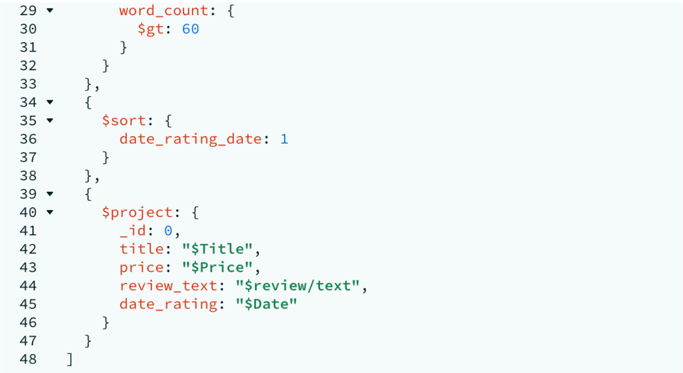


```
      num_books_above_4_fourth_quarter: [
        { $match: { month: { $in: [10, 11, 12] } } },
        { $count: "count" }
      ]
    }
  },
  {
    $project: {
      num_books_above_4_first_quarter: { $arrayElemAt: ["$num_books_above_4_first_quarter.cou
      num_books_above_4_second_quarter: { $arrayElemAt: ["$num_books_above_4_second_quarter.c
      num_books_above_4_third_quarter: { $arrayElemAt: ["$num_books_above_4_third_quarter.cou
      num_books_above_4_fourth_quarter: { $arrayElemAt: ["$num_books_above_4_fourth_quarter.c
    }
  }
]
```

Query 10



```
[
  {
    $addFields: {
      date_rating_date: {
        $dateFromString: {
          dateString: "$Date",
          format: "%Y-%m-%d"
        }
      },
      word_count: {
        $size: {
          $split: ["$review/text", " "]
        }
      }
    }
  },
  {
    $match: {
      date_rating_date: {
        $gte: {
          $dateSubtract: {
            startDate: "$$NOW",
            unit: "year",
            amount: 2
          }
        },
        $lte: "$$NOW"
      },
```



```
      word_count: {
        $gt: 60
      }
    }
  },
  {
    $sort: {
      date_rating_date: 1
    }
  },
  {
    $project: {
      _id: 0,
      title: "$Title",
      price: "$Price",
      review_text: "$review/text",
      date_rating: "$Date"
    }
  }
]
```

5. Generating reports about the time performances for the queries

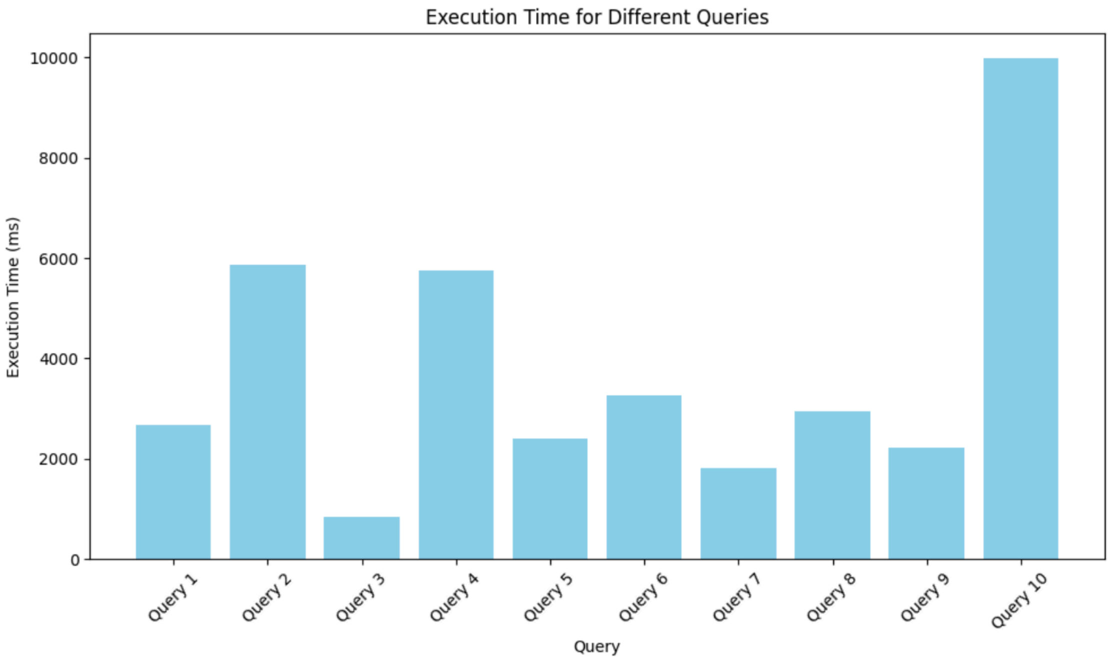
The queries were executed on the following machine: MacBook Air 2019 Intel core i5 8GB RAM. In this section a detailed overview of the time performances are given for each query, in both databases.

5.1 Time performances in PostgreSQL

There are 2 times in the PostgreSQL execution: execution time and fetching time.
Execution time refers to the amount of time it takes for the database server to process a query, while fetching time refers to the time taken to send the result set from the database server to the client application. The total execution time is a sum from both of these times.

Query(Q)	**Execution time of the query(ms)**
Q1	2673
Q2	5859
Q3	836
Q4	5742
Q5	2401
Q6	3263
Q7	1823
Q8	2939
Q9	2215
Q10	9969

Table 1: Overview of the execution time of each query in PostgreSQL

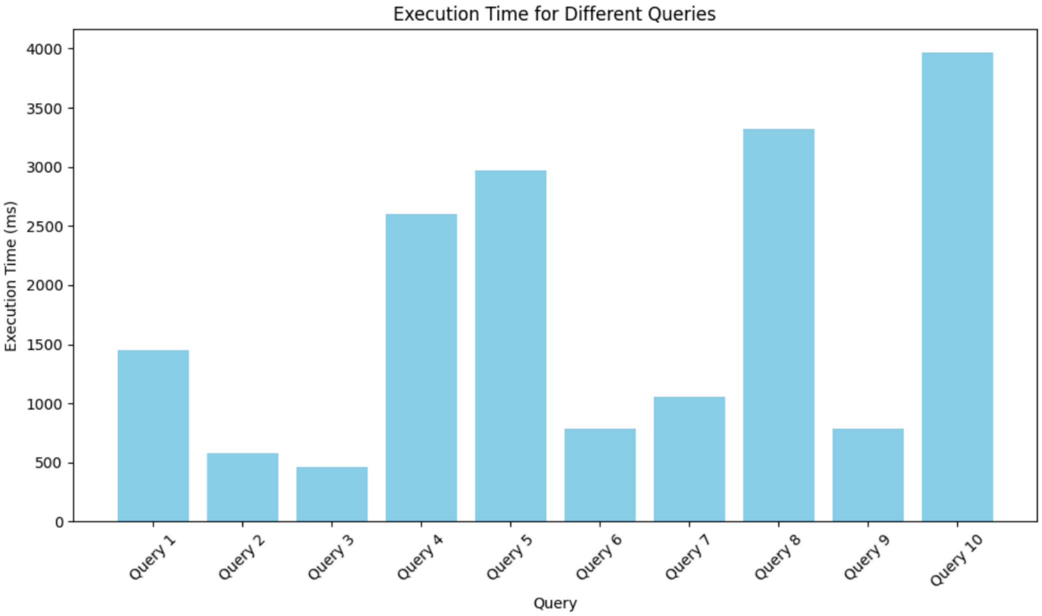


Picture 7: Graphic overview of the execution time of each query in PostgreSQL

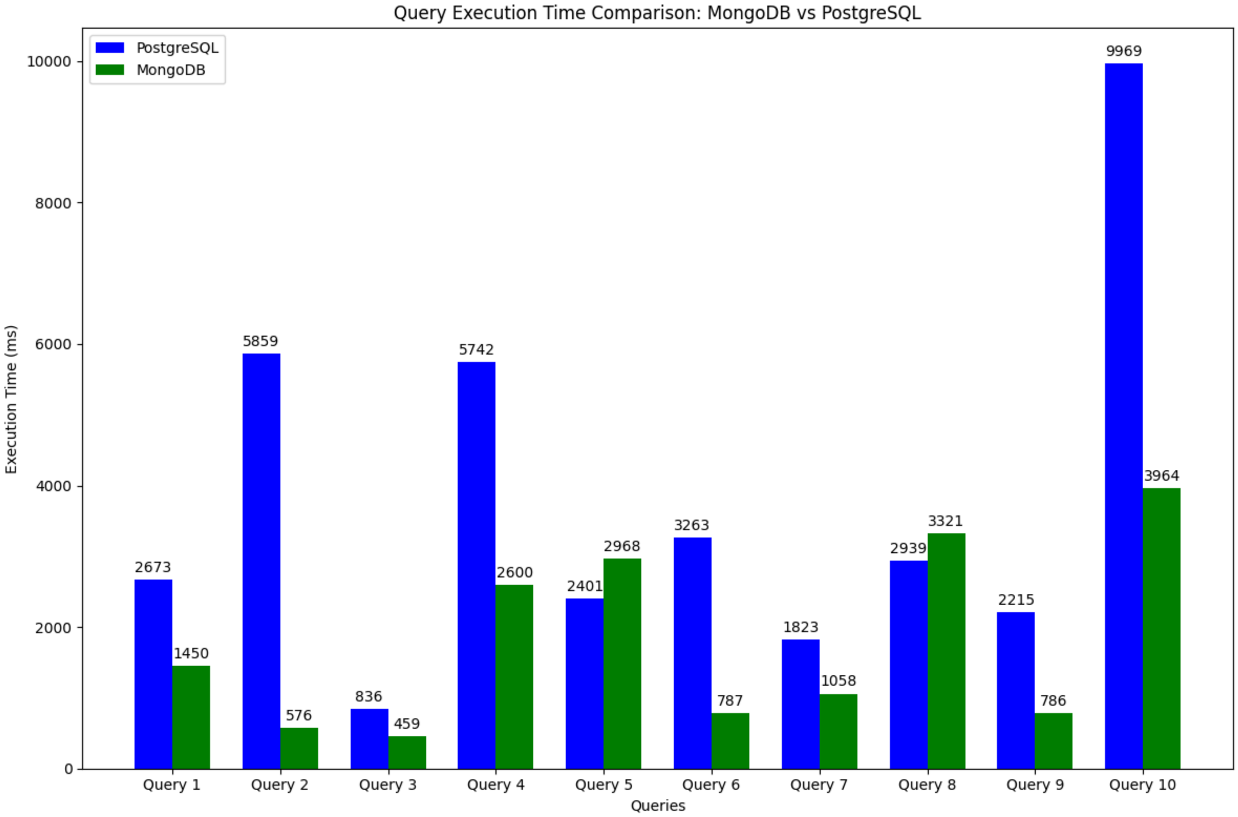
5.2 Time performances in MongoDB

Прашалник(Query-Q)	**Време на извршување на прашалникот(ms)**
Q1	1450
Q2	576
Q3	459
Q4	2600
Q5	2968
Q6	787
Q7	1058
Q8	3321
Q9	786
Q10	3964

Table 2: Overview of the execution time of each query in MongoDB



Picture 8: Graphic overview of the execution time of each query in MongoDB



Picture 9: Comparison of the execution times for each query in both of the databases using a graph

According to the received results, we can conclude that the non-relational database gives better performances than the relational. Eight out of ten queries have shorter execution time in MongoDB.

The possible reasons why the two remaining queries have larger execution time in MongoDB, can be the fact that MongoDB is not very suitable for joining data from multiple instances, as well as working with functions like cast and substring that take more execution time in this type of database.

6. Conclusions

There is often a challenge and dilemma in choosing an appropriate database. A major factor influencing the choice is the nature of the data itself.

Relational databases are known to have schema and take data from fewer sources. They are used when the integrity of the data is preserved and they are normalized. This means that if any data does not match the schema and breaks the constraints, it will not be able to be inserted. This would affect the performance of the database.

On the other hand, non-relational databases take a different approach. They arise mainly as a need for improved performance due to the emergence of huge amounts of data in recent times. Apart from managing unstructured data, they can also manage structured and semi-structured data. They do not maintain a schema like relational databases and have eventual consistency. Specifically in our case, the document-oriented Mongo database creates dynamic schemas, which means that records can be inserted without the structure being created beforehand. MongoDB also supports indexing, which enables fast searching through documents. It is also worth mentioning that this type of database has horizontal scaling, which means that data is distributed across multiple nodes, giving faster performance.[6]

In this project, MongoDB gave better performance. The reasons for this phenomenon are the mentioned characteristics. MongoDB is known to provide better performance for queries involving filtering and aggregations. On the other hand, its weaknesses are working with data joins and nested queries, which is actually inherent to relational databases.

Depending on what we want to achieve and what queries we want to write, strengths and weaknesses should be considered before choosing an appropriate database to work with.

7. References

[1] M. Sharma, V. D. Sharma and M. M. Bundele, "Performance Analysis of RDBMS and No SQL Databases: PostgreSQL, MongoDB and Neo4j," 2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, 2018, pp. 1-5, doi: 10.1109/ICRAIE.2018.8710439.

[2] Smith, A.K., Ayanian, J.Z., Covinsky, K.E. *et al. Conducting High-Value Secondary Dataset Analysis: An Introductory Guide and Resources*. *J GEN INTERN MED* **26**, 920–929 (2011). https://doi.org/10.1007/s11606-010-1621-5

[3] *Amazon Books Reviews*. (2022, September 13). Kaggle. https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews/data?fbclid=IwAR07NneLx8rHJAWxBfAmC2iMHPwXabX5KoAt9Ad6IaGyQ75W8itBlIexIb8

[4] Alex Sington. (2022, October 29). *Import CSV files into PostgreSQL | pgAdmin | Data Analyst Skill Tutorial #3* [Video]. YouTube. https://www.youtube.com/watch?v=UdMj8iKSCoc

[5] *MonGoDB: The Definitive Guide*. (n.d.). Google Books. https://books.google.mk/books?hl=en&lr=&id=pIrCDwAAQBAJ&oi=fnd&pg=PR2&dq=installing+mongodb&ots=bnNziBEsEd&sig=-IzwpZWmOYMWDARjsvpemm9pggQ&redir_esc=y#v=onepage&q=installing%20mongodb&f=false

[6] Dipina Damodaran, B., Salim, S. and Vargese, S.M., 2016. Performance evaluation of MySQL and MongoDB databases. *Int. J. Cybern. Inform.(IJCI)*, *5*, pp.387-394.
Link to paper