HSLU Hochschule Luzern

A DATABASE-DRIVEN BOOK RECOMMENDATION SYSTEM

DATABASE PROJECT REPORT

Database Management for Data Scientists – F23

Maha Hazimé-Zayour - maha.hazime-zayour@stud.hslu.ch

Incheol Jeong - incheol.jeong@stud.hslu.ch

Tejesh Reddy Koki – tejeshreddy.koki@stud.hslu.ch

Vignesh Govindaraj - vignesh.govindaraj@stud.hslu.ch

TABLE OF CONTENTS

INTRODUCTION & CONTENT

This report documents the development of a book recommendation system using database technology. The project aims to apply the principles of database management to support decision-making by providing valuable information through efficient database queries. The system will utilize data analysis to support users, publishers, authors, and booksellers in making informed decisions about book recommendations, thereby enhancing the overall reading experience and benefiting the book industry ecosystem.

MOTIVATION

The motivation behind this project is to leverage the power of data analysis and database technologies to enhance the book selection process for readers, while also supporting publishers, authors, and booksellers. With the abundance of books available today, readers often face challenges in discovering books that align with their preferences and interests. By developing a book recommendation system, we aim to provide users with personalized recommendations based on their reading habits, ultimately improving their reading experience. Simultaneously, the system will support publishers in identifying emerging trends and demand for specific genres or authors, authors in tailoring their writing to engage their target audience, and booksellers in optimizing inventory and driving sales.

PROJECT IDEA & USE CASE

DECISION SUPPORT FOR VALUE GENERATION

The primary goal of the book recommendation system is to support users in making informed decisions about which books to read next. By analyzing user preferences, reading history, and other relevant data, the system will generate tailored book recommendations, enhancing the user's reading journey, increasing engagement, and encouraging exploration of new genres or authors. Additionally, the system will provide valuable insights to publishers, authors, and booksellers, assisting them in strategic decision-making, such as book acquisitions, marketing campaigns, inventory management, and customer satisfaction.

THE ANALYTICAL METHOD USED FOR DECISION SUPPORT:

The analytical method used in this project for decision support combines query-based analysis and visualization techniques.

Queries are used to extract specific information from the database, perform calculations, and generate valuable insights that are relevant to the needs of authors, booksellers, publishers, and readers. By using queries, we can identify top-rated books, average ratings, popular genres, and other factors that impact the success of books.

Visualization on the other hand plays a crucial role in understanding the data by showing trends, patterns, and characteristics of highly reviewed books in a clear and intuitive way. This helps easily interpret the information and make informed decisions about book recommendations.

This combination of query-based analysis and visualization techniques enables us to have the insight to get answers to the questions we need to answer.

DATA SOURCES USED FOR ANALYSIS

To support the book recommendation system, we have chosen Goodreads as the primary data source. Goodreads is a popular online platform where readers can discover, rate, and review books. By utilizing the Goodreads dataset from Kaggle, we gain access to a vast collection of user ratings, reviews, book metadata, and user information. This rich dataset provides valuable insights into readers' preferences, enabling us to generate accurate recommendations for users, publishers, authors, and booksellers.

Additionally, we will combine the Goodreads dataset with genre data to further enhance the recommendation system. The genre data will allow us to consider readers' genre preferences and provide more specific and relevant book recommendations. By combining multiple data sources, we aim to create a comprehensive and diverse recommendation engine that supports decision-making across various stakeholders in the book industry.

Analyzing the data sources reveals potential deficiencies such as incomplete or missing information, data inconsistencies, and possible biases in user ratings. These deficiencies need to be addressed during the data loading and transformation process to ensure the reliability and accuracy of the recommendation system.

DATABASE TECHNOLOGY WITH RATIONALE

For this project, we have chosen to implement the book recommendation system using a SQL database. SQL (Structured Query Language) is a widely used database technology that provides a robust and efficient way to store, retrieve, and manipulate structured data. By leveraging SQL, we can easily perform complex queries and aggregations on the dataset, facilitating the generation of accurate and relevant book recommendations.

Compared to other technologies, SQL databases offer several advantages for this project. Firstly, SQL provides a standardized language for querying and manipulating data, making it easier to write and optimize database queries. Secondly, SQL databases offer strong data consistency and reliability, ensuring that the recommendation system operates with accurate and up-to-date information. Finally, SQL databases are well-suited for relational data models, allowing us to establish meaningful relationships between books, users, genres, and other entities.

However, it's worth noting that SQL databases may have limitations in handling extremely large datasets or unstructured data. In such cases, alternative technologies like NoSQL databases or big data frameworks might be more suitable. Nevertheless, for this particular book recommendation system

DATA MODEL & DATABASE SCHEMA

DATA MODEL ACCORDING TO ER TECHNIQUE V. CHEN (1976)

The Entity-relationship technique, first introduced by Chen in 1976, is a popular and widely used method for designing data models. This technique focuses on identifying and representing the entities, attributes, and relationships within a system. In this section, we will illustrate two different types of entity-relationships diagrams, but they both represent the same concept.

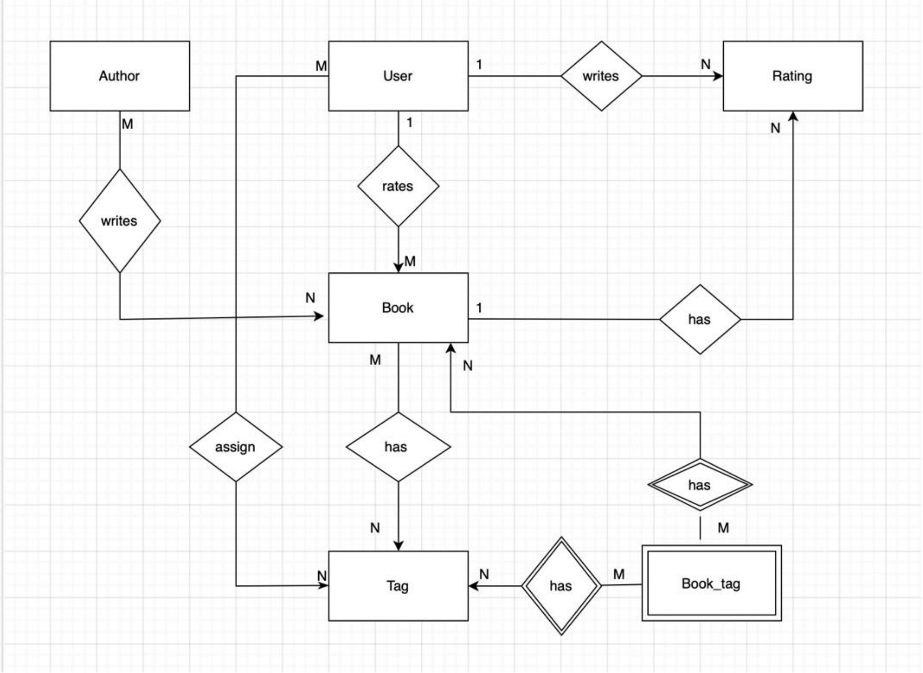


Figure 1: Database schema for Book Recommendation system

The first diagram presents the data model for our book recommendation system using the ER technique v. Chen. In Chen ER diagram, entities are represented as rectangles, relationships are represented as diamonds, and attributes are represented as ovals. The Chen model also includes constructs for weak entities and recursive relationships.

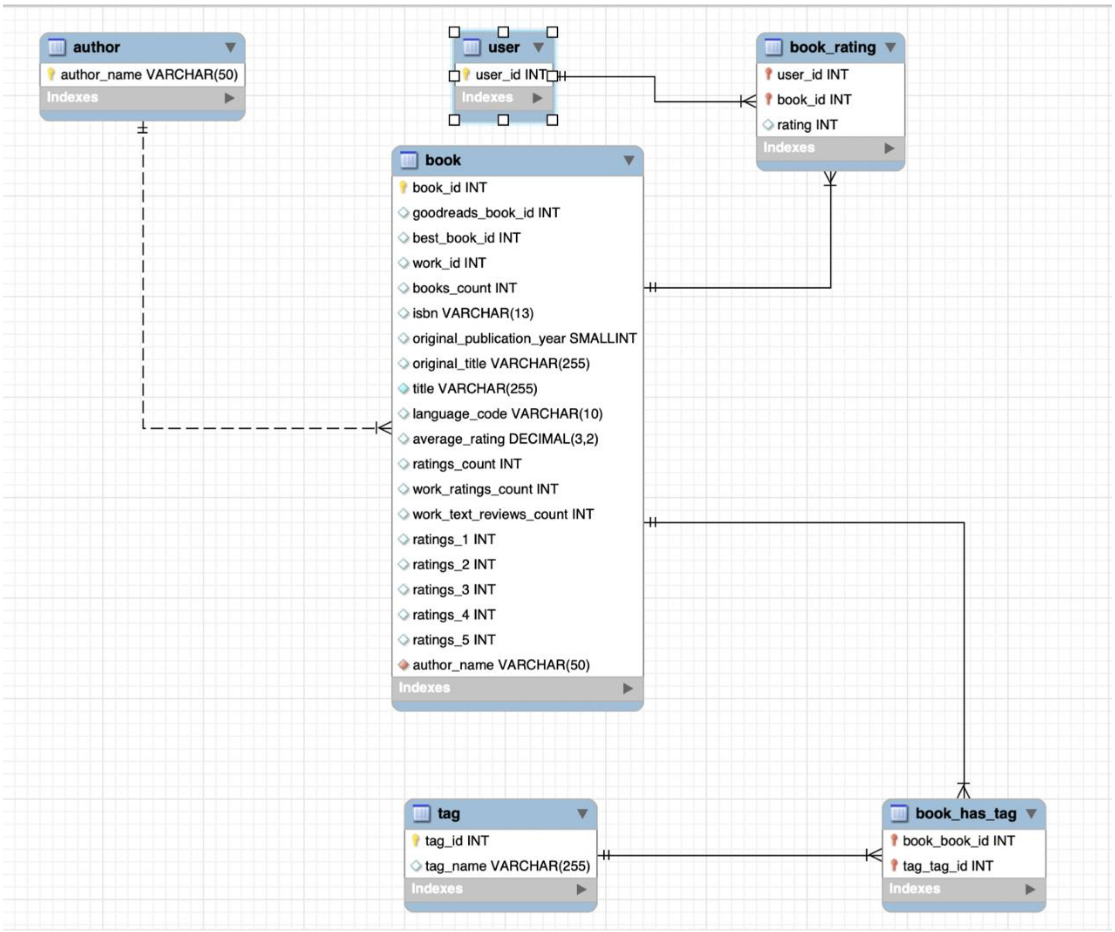


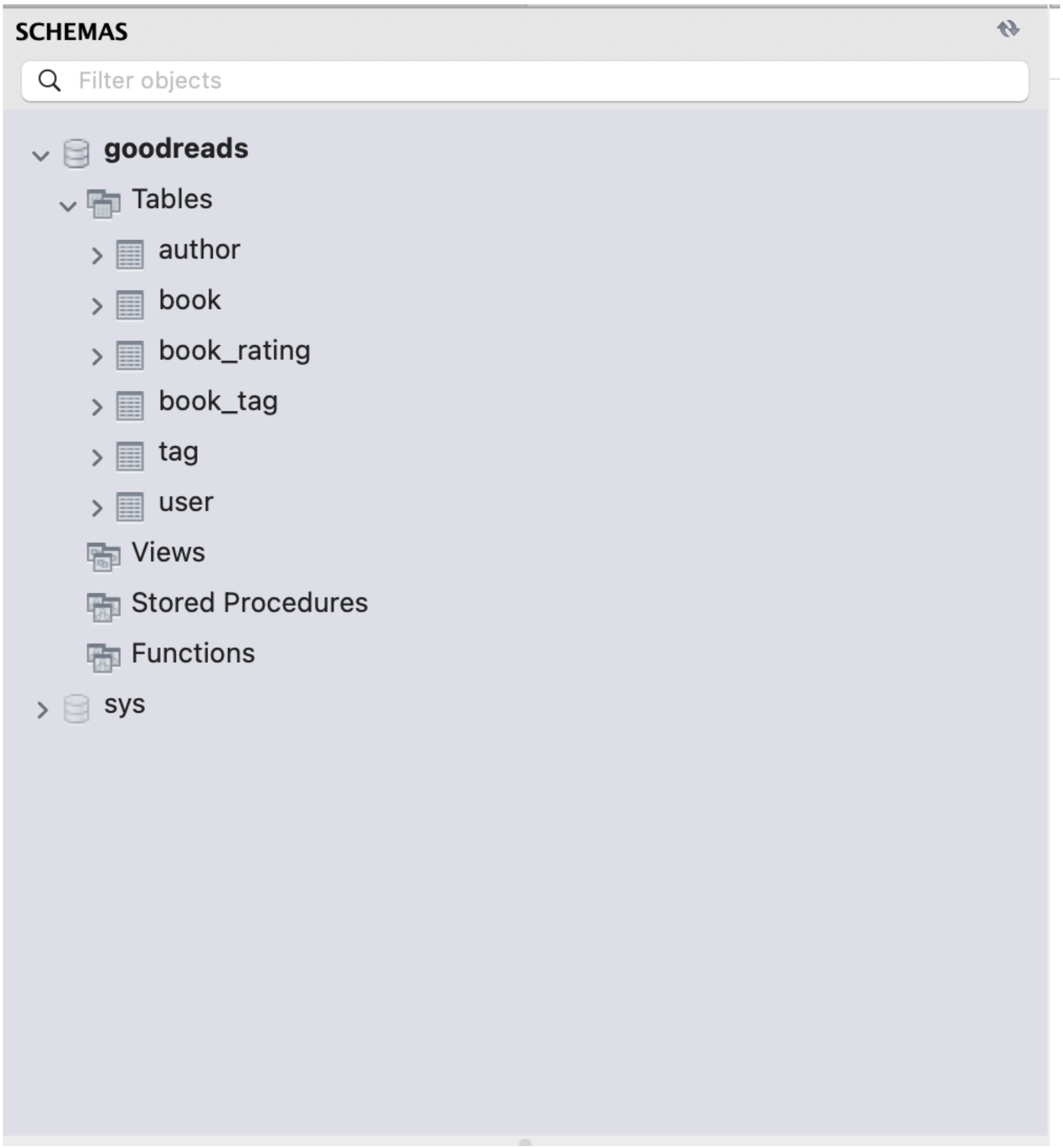
Figure 2: ER diagram for Book Recommendation system

The second ER diagram illustrated in MySQL Workbench. It is based on the Crow's Foot notation, which is like Chen ER diagram but with some differences in the graphical representation of relationships and cardinality. It provides a comprehensive overview of the database design, showing the physical implementation of the database, including the table names, columns, and foreign key constraints. The ER diagram illustrated in MySQL Workbench is a tool used to design and document databases.

As the diagrams above show the entities in our data model represent the main objects in our system, while the relationships between these entities illustrate how they are connected and interact with each other. For example, the Book entity is connected to the Author entity through a many-to-many relationship, which represents the fact that many books can be written by many authors. Similarly, the Book entity is connected to the Tag entity through a many-to-many relationship, which represents the fact that many books can have many tags. Overall, the ER diagram provides a clear and concise visual representation of our data model, which will serve as a foundation for the development of our book recommendation system.

DATABASE SCHEMA IN DATABASE DDL SYNTAX (E.G. SQL)

The provided database schema consists of following tables: Author, Book, Tags, Book_Tag, User, and Rating. Each table represents a specific entity in the Goodreads book ratings dataset.



SCHEMAS

Filter objects

- goodreads
  - Tables
    - author
    - book
    - book_rating
    - book_tag
    - tag
    - user
  - Views
  - Stored Procedures
  - Functions
- sys

Figure 3: DB schema tables from SQL

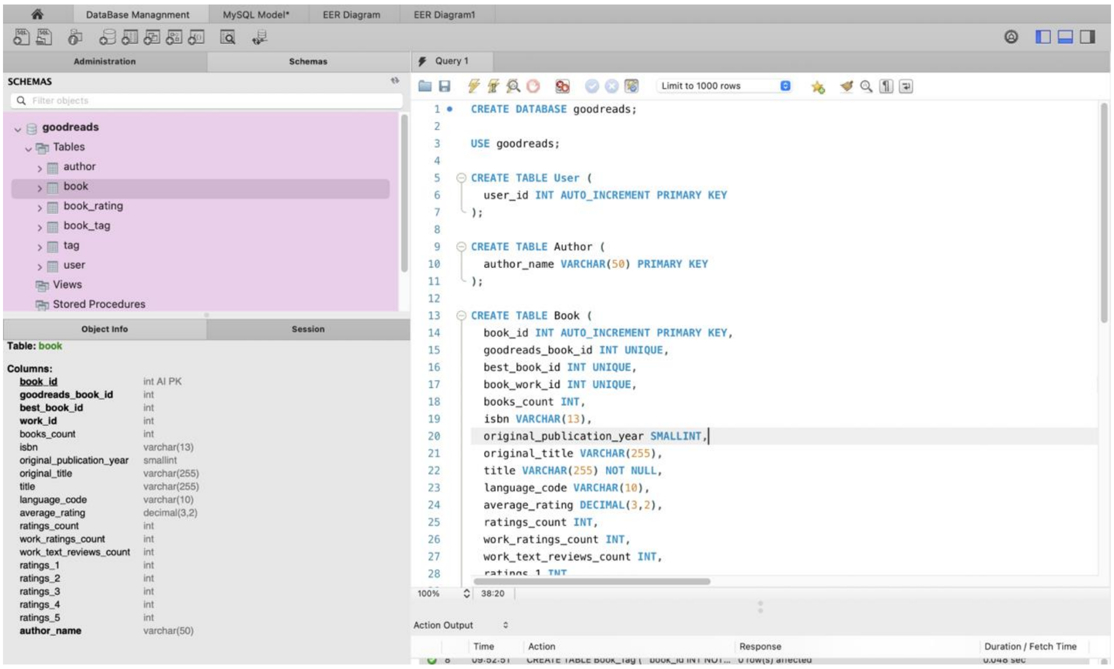


Table: book

Columns:

book_id	int AI PK
goodreads_book_id	int
best_book_id	int
work_id	int
books_count	int
isbn	varchar(13)
original_publication_year	smallint
original_title	varchar(255)
title	varchar(255)
language_code	varchar(10)
average_rating	decimal(3,2)
ratings_count	int
work_ratings_count	int
work_text_reviews_count	int
ratings_1	int
ratings_2	int
ratings_3	int
ratings_4	int
ratings_5	int
author_name	varchar(50)

```sql
CREATE DATABASE goodreads;

USE goodreads;

CREATE TABLE User (
  user_id INT AUTO_INCREMENT PRIMARY KEY
);

CREATE TABLE Author (
  author_name VARCHAR(50) PRIMARY KEY
);

CREATE TABLE Book (
  book_id INT AUTO_INCREMENT PRIMARY KEY,
  goodreads_book_id INT UNIQUE,
  best_book_id INT UNIQUE,
  book_work_id INT UNIQUE,
  books_count INT,
  isbn VARCHAR(13),
  original_publication_year SMALLINT,
  original_title VARCHAR(255),
  title VARCHAR(255) NOT NULL,
  language_code VARCHAR(10),
  average_rating DECIMAL(3,2),
  ratings_count INT,
  work_ratings_count INT,
  work_text_reviews_count INT,
  ratings_1 INT
```

```
8   09:52:51   CREATE TABLE Book_Tag ( book_id INT NOT...   0 row(s) affected   0.048 sec
```

Figure 4: SQL code for tables creation

The Authors table stores information about the authors of the books, including their unique author_id and name.

The Books table stores information about the books themselves, including their unique book_id, goodreads_book_id, best_book_id, work_id, books_count, ISBNs, publication year, titles, language codes, average ratings, and ratings counts. Each book is associated with one or more authors through the Author table and one or more genres through the Book_Tag table.

The Tag table stores information about the different tags/genres of the books, including their unique tag_id and tag_name.

The Book_Tag table establishes a many-to-many relationship between the Book and Tag tables by associating each book with one or more tags.

The Author table establishes a many-to-many relationship between the Book and Author tables by associating each book with one or more authors.

The User table stores information about the users of the Goodreads platform, including their unique user_id and username.

The Rating table stores information about the ratings given by users to the books on the platform, including their unique rating_id, the book_id they are associated with, the user_id who provided the rating, and the actual rating value given. Each rating is associated with one book and one user.

Overall, this schema provides a comprehensive structure to capture the essential information about the books, authors, genres, and ratings data in the Goodreads book ratings dataset. The relationships between the different entities are clearly defined, allowing for easy querying and analysis of the data. By implementing this schema in a database management system, decision-makers such as publishers and authors can make informed decisions about publishing, marketing, and advertising strategies based on the ratings and reviews of the books.

RELATIONSHIP BETWEEN MODEL AND SCHEMA

The connection between the conceptual model and the physical schema is vital as it transforms the high-level representation of entity relationships and attributes into a concrete implementation of data storage and manipulation in the database. Changes made to ER model must correspond to updates in the physical schema to maintain data integrity.

For instance, the Goodreads database demonstrates how the ER model entities such as Author, Book, Tag, User, and Rating with their attributes and relationships are used to create the physical schema of tables, columns, and relationship. The physical schemamay differ from the conceptual model depending on the specific database management system in use.

The relationship between the conceptual model and physical schema is visible in the foreign key constraints that relate different entities in the ER model to their corresponding tables in the physical schema. Maintaining a close relationship between the conceptual model and the database schema is crucial in ensuring that the stored data is consistent, accurate, and meaningful.

Ultimately, the database schema created from the conceptual model is essential for effective data management, enabling informed decision-making by publishers, authors, and other stakeholders. Therefore, understanding the relationship between the conceptual ER model and physical schema is critical for comprehending database design, structure, and functionality.

VERBAL DESCRIPTION OF THE MOST IMPORTANT DATA CLASSES AND ATTRIBUTES

The Goodreads dataset is comprised of various essential data classes and attributes that are vital for analysis, visualization, and decision-making. One of the primary data classes is the "Books" entity, which includes key attributes like book_id, title, author_id, average_rating, and ratings_count. These attributes offer valuable information about the books, such as their unique identifiers, titles, authors, and ratings, enabling users to evaluate the popularity and quality of books on the platform.

Another significant data class is the "Authors" entity, which comprises attributes such as author_name, and average_rating. These attributes provide critical details about the authors, such as their unique identifiers, names, and average ratings, empowering users to analyze the popularity and quality of authors on the platform.

The "Tag" entity is another essential data class that encompasses attributes like tag_id and tag_name. These attributes offer valuable information about the tags/genres of books available on the platform, enabling users to inspect the distribution of books across genres and identify popular genres that may be relevant for marketing and advertising.

The "User" entity is another essential data class in the Goodreads dataset, comprising attributes like user_id. This attribute provides essential details about the platform's users, such as their unique identifiers, names, and locations, enabling users to analyze the demographics and preferences of their user base.

Finally, the "Rating" entity is an essential data class that includes attributes such as user_id, book_id, and rating. These attributes provide valuable information about the ratings given by users to books on the platform, enabling users to assess the popularity and quality of books from the users' perspective.

In conclusion, the Goodreads dataset encompasses various crucial data classes and attributes that are indispensable for analysis, visualization, and decision-making. The Books, Authors, Genres, Users, and Ratings entities, along with their attributes, offer valuable insights into the popularity, quality, and preferences of books and authors on the platform. This data is fundamental for publishers, authors, and advertisers to make informed decisions about publishing, marketing, and advertising strategies.

LOADING & TRANSFORMING THE DATA SYSTEM COMPONENTS, RELATIONSHIPS, AND DATA FLOWS

The system setup involves a MySQL database installation hosted on a dedicated server and cloud-based infrastructure. We utilize MySQL Workbench as our primary client tool for managing and administering the database. Communication between the client machines and the MySQL server is established through network connectivity using the TCP/IP protocol.

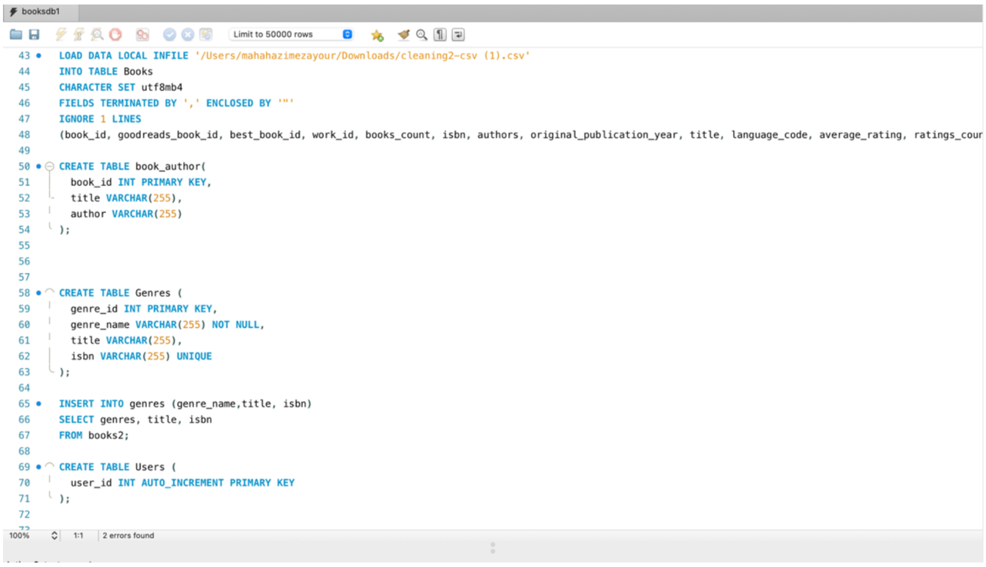
In our data loading and transformation process, we import data from CSV files into the MySQL database. The client machines, where the CSV files are located, initiate the data loading process by executing SQL commands. The data flows from the client machines to the MySQL server over the network connection. The server then processes the data and inserts it into the respective tables within the database.

For the data loading step, we utilize the LOAD DATA LOCAL INFILE command in MySQL to import the CSV files into the appropriate tables. This command allows us to specify the file path, character set, field delimiter, and other parameters to ensure proper data loading.

Once the data is loaded, we perform data transformation to structure and organize it for analysis. This involves creating additional tables, such as BOOK_AUTHOR and Genres, and populating them with relevant data from the existing tables. We establish relationships between tables using foreign key constraints, ensuring data integrity and enabling efficient querying and analysis.

DATA LOADING:

The initial phase of the process involves importing the raw data from external sources, primarily in CSV file format, into the MySQL database.



```sql
LOAD DATA LOCAL INFILE '/Users/mahahazimezayour/Downloads/cleaning2-csv (1).csv'
INTO TABLE Books
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
IGNORE 1 LINES
(book_id, goodreads_book_id, best_book_id, work_id, books_count, isbn, authors, original_publication_year, title, language_code, average_rating, ratings_cour

CREATE TABLE book_author(
  book_id INT PRIMARY KEY,
  title VARCHAR(255),
  author VARCHAR(255)
);



CREATE TABLE Genres (
  genre_id INT PRIMARY KEY,
  genre_name VARCHAR(255) NOT NULL,
  title VARCHAR(255),
  isbn VARCHAR(255) UNIQUE
);

INSERT INTO genres (genre_name,title, isbn)
SELECT genres, title, isbn
FROM books2;

CREATE TABLE Users (
  user_id INT AUTO_INCREMENT PRIMARY KEY
);
```

Figure 5: Loading CSV data in MYSQL server

The provided code snippets demonstrate the data loading steps for various tables within the database.

Loading Authors Data:

The authors table is created with columns for author_id and author_name. The data from the CSV file, 'goodreads_authors_noID.csv,' is loaded into the authors table using the LOAD DATA LOCAL INFILE command. This command specifies the file path, character set, field delimiter (','), and instructs the system to ignore the header row.

Loading Books Data:

The Books table is created with columns representing different book attributes. The data from the CSV file, 'cleaning2-csv (1).csv,' is loaded into the Books table using the LOAD DATA LOCAL INFILE command. The file path, character set, field delimiter (','), and enclosing character ('"') are specified, and the header row is ignored.

Loading Books2 Data:

The Books2 table is created with columns designed to store additional book information. The data from the CSV file, 'goodreads_genrescsv.csv,' is loaded into the Books2 table using the LOAD DATA LOCAL INFILE command. Similar to previous cases, the file path, character set, field delimiter (','), and ignoring the header row are specified.

Loading Ratings Data:

To facilitate the storage of user ratings for books, the rating table is created. The data from the CSV file, 'ratings.csv,' is loaded into the rating table using the LOAD DATA LOCAL INFILE command. As before, the file path, character set, field delimiter (','), and ignoring the header row are specified.

DATA TRANSFORMATION:

Following the data loading phase, data transformation is undertaken to organize and modify the imported data to align it with the desired format for analysis. The provided code includes the creation of additional tables and the application of specific transformations.

Transforming Book-Author Relationship:

The BOOK_AUTHOR table is created to establish a relationship between books and authors. The title and authors columns are populated by selecting the corresponding data from the books table.

Transforming Genre Information:

To capture genre information associated with books, the Genres table is created. The isbn, title, and genres data from the books2 table are inserted into the Genres table, allowing for genre-based analysis.

Establishing Relationships:

In order to establish relationships between tables, foreign key constraints are added to the rating table. These constraints reference the user and Books2 tables, ensuring the integrity of the data. Specifically, the user IDs and book IDs from the rating table are linked to the primary keys in the respective tables.

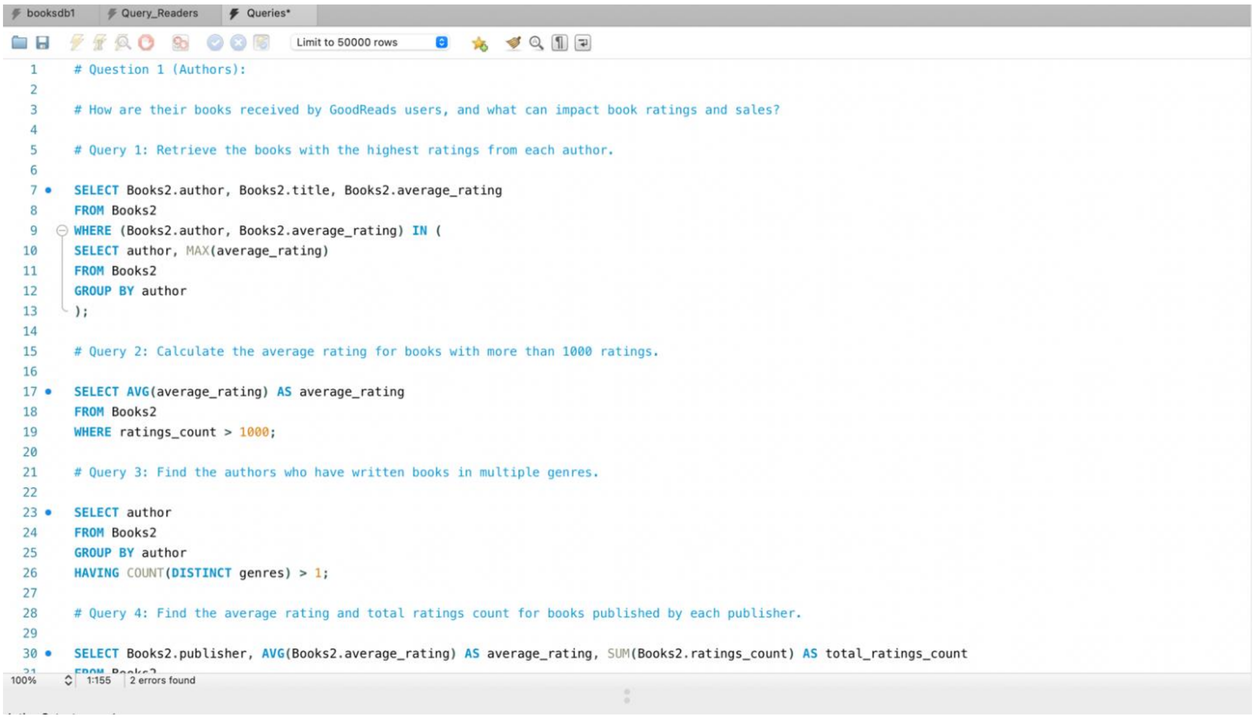
By following these steps, the data is successfully loaded into the MySQL database and transformed to facilitate analysis.

Overall, our system architecture enables seamless communication and data flow between the client machines and the MySQL server. It leverages MySQL Workbench as the client tool and utilizes the power of SQL commands and data loading techniques to import and transform the data, setting the foundation for further analysis and insights.

ANALYZING & EVALUATING DATA

To gain insights and answer the business questions, we utilized various SQL queries on our database. Here, we describe each query and provide an explanation of how it works to calculate the relevant metrics:

QUERIES IN DATABASE SYNTAX (E.G. SQL)



```sql
# Question 1 (Authors):

# How are their books received by GoodReads users, and what can impact book ratings and sales?

# Query 1: Retrieve the books with the highest ratings from each author.

SELECT Books2.author, Books2.title, Books2.average_rating
FROM Books2
WHERE (Books2.author, Books2.average_rating) IN (
SELECT author, MAX(average_rating)
FROM Books2
GROUP BY author
);

# Query 2: Calculate the average rating for books with more than 1000 ratings.

SELECT AVG(average_rating) AS average_rating
FROM Books2
WHERE ratings_count > 1000;

# Query 3: Find the authors who have written books in multiple genres.

SELECT author
FROM Books2
GROUP BY author
HAVING COUNT(DISTINCT genres) > 1;

# Query 4: Find the average rating and total ratings count for books published by each publisher.

SELECT Books2.publisher, AVG(Books2.average_rating) AS average_rating, SUM(Books2.ratings_count) AS total_ratings_count
FROM Books2
```

Figure 6: Querying data

VERBAL DESCRIPTION OF QUERIES

Authors:

Query 1: Retrieves the books with the highest ratings from each author by selecting the author, book title, and average rating where the author and average rating match the maximum values for each author in the Books2 table.

Query 2: Calculates the average rating for books with more than 1000 ratings by selecting the average rating from the Books2 table where the ratings count is greater than 1000.

Booksellers:

Query 6: Retrieves the top-rated book from each publisher by joining the Books2 table with a subquery that finds the maximum average rating for each publisher and then matching the publisher and average rating with the Books2 table.

Query 7: Calculates the average rating for books with more than 1000 ratings using the same query as Query 2 for authors.

Publishers:

Query 11: Finds the average rating, total ratings count, and average number of pages for books in each genre by joining the Books2 and Genres tables on the ISBN and grouping the results by genre.

Query 12: Retrieves the top-rated book and its author from each genre by joining the Books2 and Genres tables on the ISBN and selecting the genre, book title, and author where the average rating is the maximum for each genre.

Query 13: Finds the number of books published per year and the average rating of books published in each year by selecting the publication date, counting the number of books, and

calculating the average rating from the Books2 table grouped by publication date.

Readers:

Query 2: Retrieves the top-rated book and its author from each genre using the same query as Query 12 for publishers and book genres.

Query 6: Retrieves the top-rated book from each publisher using the same query as Query 6 for booksellers.

Query 9: Retrieves the books with the highest ratings from each author using the same query as Query 1 for authors.

Query 15: Finds the average rating, total ratings count, and average number of pages for books in each genre using the same query as Query 11 for publishers.

SHOW CONNECTION BETWEEN QUERIES AND USE CASE:

The queries described above directly address the specific business questions outlined in the report. Each query is designed to extract relevant data from the database and calculate metrics that provide insights into the respective use cases.

For authors, Queries 1 and 2 allow us to analyze how authors' books are received by GoodReads users and identify factors that can impact book ratings and sales.

For booksellers, Queries 6 and 7 help identify the most popular books among GoodReads users and explore factors influencing user reviews and ratings.

For publishers, Queries 11, 12, and 13 provide insights into the popularity of book genres, the impact of different factors on book ratings and sales, and the performance of books published by each publisher.

For readers, Queries 2, 6, 9, and 15 enable them to discover new books and authors based on popularity, genre, and other factors, helping them make informed choices and explore new reading experiences.

By executing these queries and analyzing the results, we are able to generate meaningful insights and metrics that directly address the business questions and support decision-making processes within the respective use cases.

EFFICIENCY & QUERY PERFORMANCE OPTIMIZATION MEASURES USED

1. Number of tables in the database:

This query is used to count total number of tables in the database ‘booksdb’ by quering the information_schema.tables and for tables filtering by using table_schema.

2. Size of each table in the database:

This query is used to retrieve the table_name and calculates the size of each table in ‘booksdb’ by summing the index_length and data_length columns from information_schema.tables, and the size was rounded to two decimal places and returned in megabytes.

3. Number of records in each table:

This query is used to retrieve the table_name and table_rows columns from the information_schema.tables table to get the number of records – rows in each table in the 'booksdb' database. The result is then ordered by using the table_rows column in descending order.

4. Most frequently accessed tables based on queries:

This query is used to join the information_schema.tables and information_schema.columns tables and used to determine the most frequently accessed It will count the total number of occurrences of each table_name and then orders the result by the query_count in descending order.

5. Show slow queries:

These 2 queries will check the status of the slow query in MySQL by the method of retrieving the values of slow_query_log and long_query_time, it will show whether the slow query logging is enabled or not and the duration of threshold for queries to be considered slow.

6. Enable slow query logging:

These 2 queries will enable slow query logging by setting the values of the global variables slow_query_log and long_query_time to 'ON' and 1, respectively.

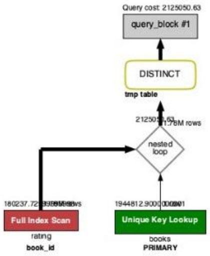
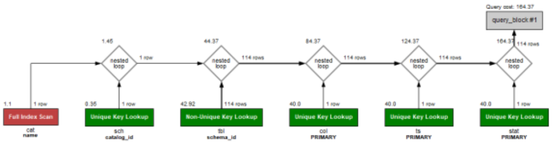
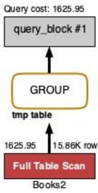
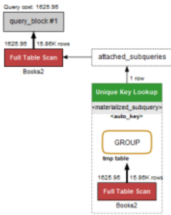
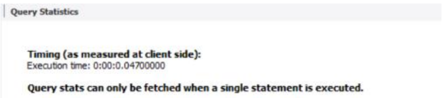
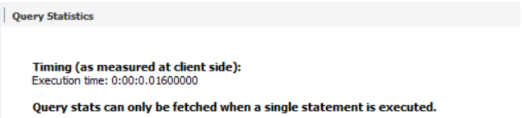
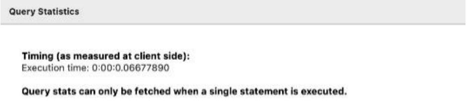
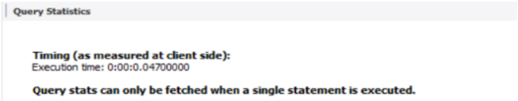
7. View slow query log: This query helps to retrieve the values of the slow_query_log_file variable, by which it specifies the file path where the slow query log is stored. It also shows the location of the slow query log file for the further analysis.

8. Analyze slow queries:

This query will retrieve all the entries from the mysql.slow_log table, which has information about slow queries that executed in the database. It can be used to analyze the details of slow queries and also helps to optimize their performance.

These queries above will provide more useful information about the size, performance and structure of tables in the 'booksdb' database, and also slow query logging, they help in monitoring and optimizing the database for better performance.

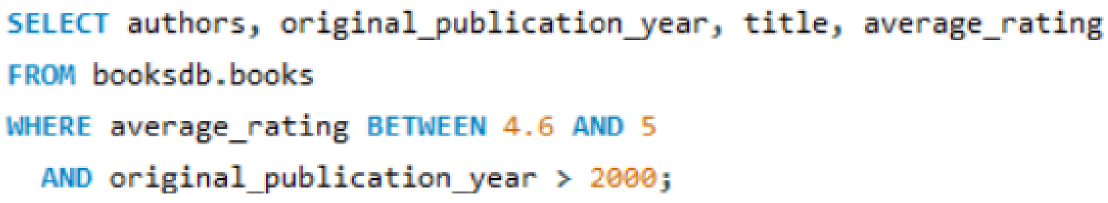
CORRELATION BETWEEN MEASURES AND RUNTIME OPTIMIZATION

	Original Query	With Primary Key
Time	0.04700 secs for Query 1 and 0.066789 for Query 2	0.01600 secs for Optimization 1 and 0.04700 secs for Optimization 2.
Operators	Full Table Scan	Full Table Scan
Execution Plan 1	 Query cost: 2125050.63 query_block #1 DISTINCT tmp table 2125050.63 1.78M rows nested loop 180237.72 Full Index Scan rating book_id 1944812.90 Unique Key Lookup books PRIMARY	 Query cost: 164.37 query_block #1 1.45 44.37 84.37 124.37 164.37 nested loop 1 row 114 rows 1.1 Full Index Scan cat name 0.35 Unique Key Lookup sch catalog_id 42.92 Non-Unique Key Lookup tbl schema_id 40.0 Unique Key Lookup col PRIMARY 40.0 Unique Key Lookup ts PRIMARY 40.0 Unique Key Lookup stat PRIMARY
Execution Plan 2	 Query cost: 1625.95 query_block #1 GROUP tmp table 1625.95 15.86K rows Full Table Scan Books2	 Query cost: 1625.95 query_block #1 1625.95 15.86K rows Full Table Scan Books2 attached_subqueries 1 row Unique Key Lookup <materialized_subquery> <auto_key> GROUP tmp table 1625.95 15.86K rows Full Table Scan Books2
Query status 1 from MySQL	 Query Statistics Timing (as measured at client side): Execution time: 0:00:0.04700000 Query stats can only be fetched when a single statement is executed.	 Query Statistics Timing (as measured at client side): Execution time: 0:00:0.01600000 Query stats can only be fetched when a single statement is executed.
Query status 1 from MySQL	 Query Statistics Timing (as measured at client side): Execution time: 0:00:0.06677890 Query stats can only be fetched when a single statement is executed.	 Query Statistics Timing (as measured at client side): Execution time: 0:00:0.04700000 Query stats can only be fetched when a single statement is executed.

VISUALIZATION & DECISION SUPPORT

VISUALIZATION OF QUERY RESULTS

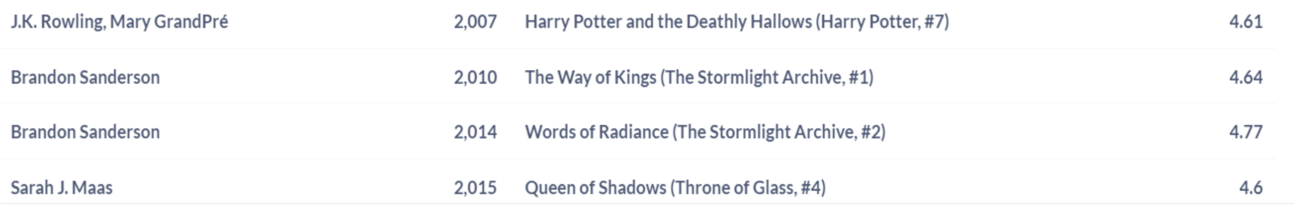
In this chapter, we're going to talk about visualizing data and decision support. We're going to show you how MySQL connects Metabase to visualize the data in the database through queries, and let's see how easy this visualization is to make decisions. First of all, visualizing the data required a target setting for book recommendation first. So we wanted to classify books with average_rating, which is a book that readers have received good reviews, between 4.6 and 5.0. Also, we set a target group with books published after 2000 to organize trendy books. So, we want to work on visualizing the books that readers have highly reviewed among the trendy books published after the 2000s to find out the characteristics of the books. The queries used to do this are as follows.



```
SELECT authors, original_publication_year, title, average_rating
FROM booksdb.books
WHERE average_rating BETWEEN 4.6 AND 5
  AND original_publication_year > 2000;
```

Figure 7: SQLQuery code_1 to fetch author, title with rating between 4.6 and 5 and publication year above 2000

And we want to visualize the data with this query. First, we want to visualize this data as a table.



J.K. Rowling, Mary GrandPré	2,007	Harry Potter and the Deathly Hallows (Harry Potter, #7)	4.61
Brandon Sanderson	2,010	The Way of Kings (The Stormlight Archive, #1)	4.64
Brandon Sanderson	2,014	Words of Radiance (The Stormlight Archive, #2)	4.77
Sarah J. Maas	2,015	Queen of Shadows (Throne of Glass, #4)	4.6

Figure 8: Result of SQLQuery_code1

However, table data is difficult to understand intuitively, so we want to analyze the data intuitively using different kinds of visualization methods. To do that, we used a scatterplot graph.

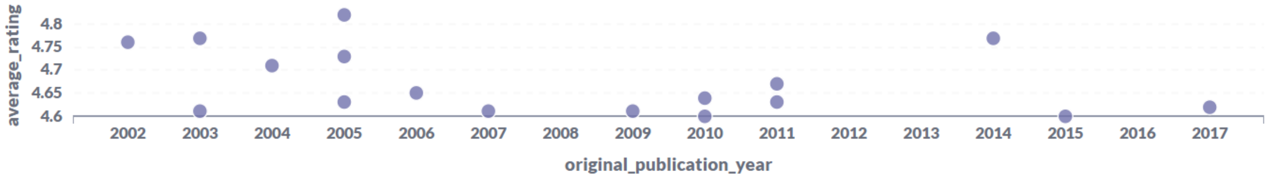
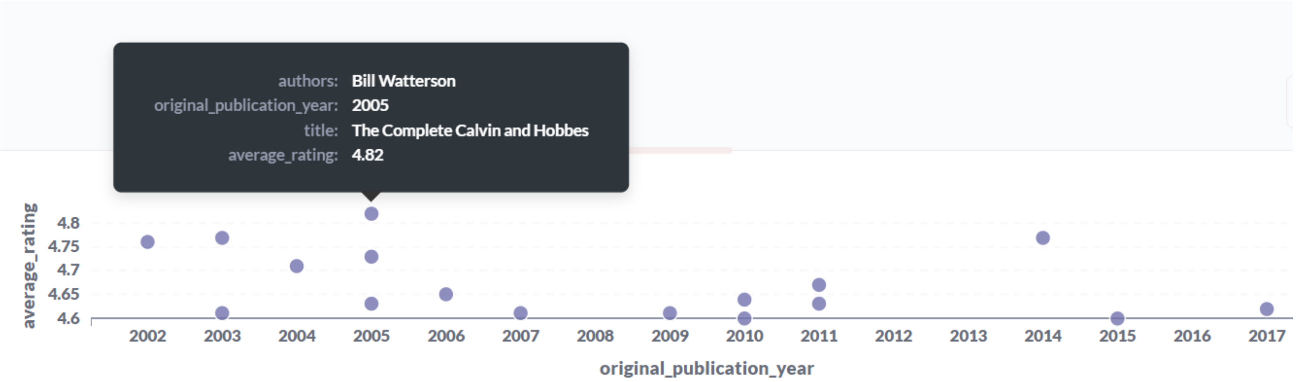


Figure 9: Scatterplot of SQLQuery_code1

This scatterplot graph helps you intuitively analyze which year has the best books. It is also set up so that you can check the information in the relevant book by placing the mouse cursor on that point. So if you want to see the book that has the highest rating among the good books published in 2005, you can just put your cursor up there.

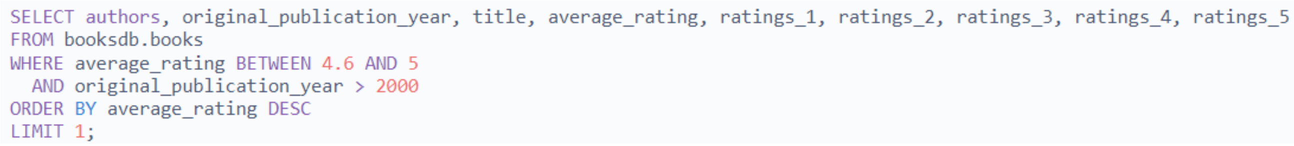


authors: Bill Watterson
original_publication_year: 2005
title: The Complete Calvin and Hobbes
average_rating: 4.82

Figure 10: Detailed view of Scatterplot of SQLQuery_code1

We also visualized each rating score to obtain detailed rating information for this book, which has the highest rating. So we wanted to intuitively see how many percentages of people gave this book five points.

We used this query to check that.



```
SELECT authors, original_publication_year, title, average_rating, ratings_1, ratings_2, ratings_3, ratings_4, ratings_5
FROM booksdb.books
WHERE average_rating BETWEEN 4.6 AND 5
  AND original_publication_year > 2000
ORDER BY average_rating DESC
LIMIT 1;
```

Figure 11: SQLQuery_code2 to get rating information and percentages of people giving 5 points

And result is below:

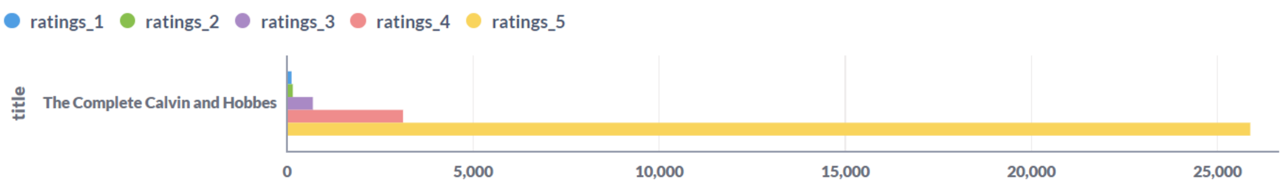


Figure 12: Result of SQLQuery_code2

We can easily check how many people gave this book 5 rating point with above visualized information.

And also, we want to reply our question we made with our queries.

Our question: Calculate the average rating for books with more than 1000 ratings.

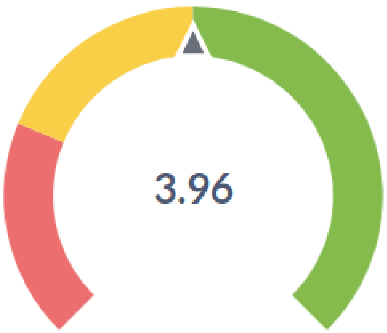


Figure 13: Average ratings of the books

Answer: The average rating for books with more than 1000 ratings is 3.96.

Our question: What genres of books are most popular

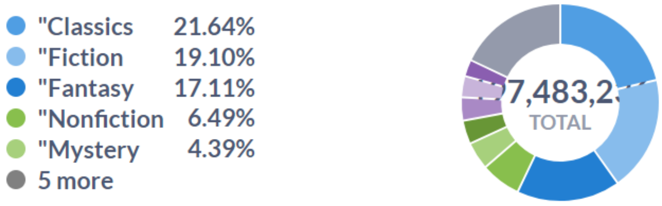


Figure 14: Genres of the book

Answer: As you can see here, the most popular genre is “Classics” account for 21.64%

Our Question: "Which publishers have the highest average (5.0) ratings for their books?

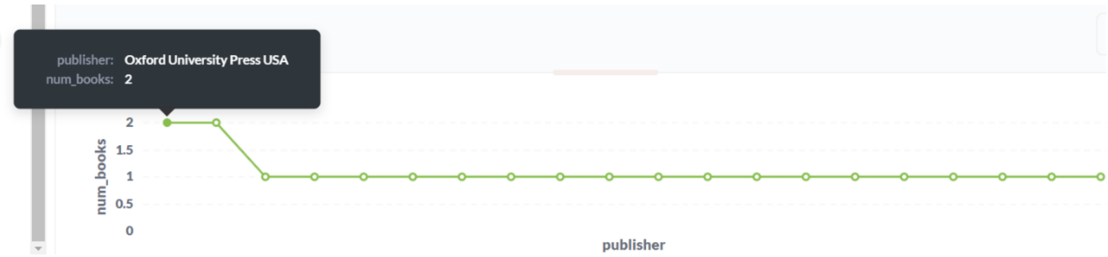


Figure 15: Visualizing plot for publishers of highest average rating -1

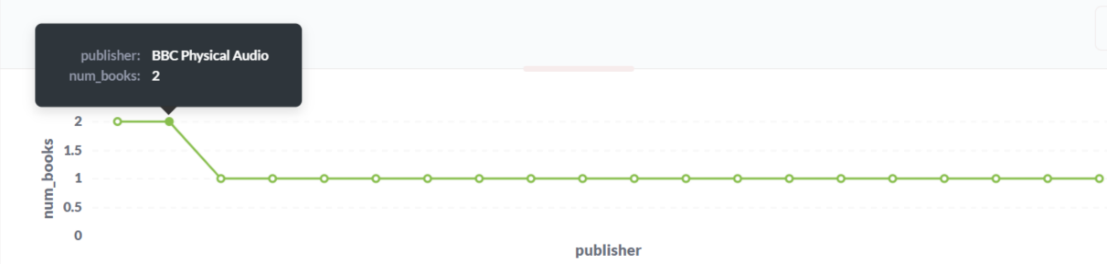


Figure 16: Visualizing plot for publishers of highest average rating-2

Answer: there are only two publishers having the highest average rating (=5.0). The Oxford University Press USA and BBC Physical Audio. So, we can refer this data to choose when we try to buy a book having good average rating.

Our question: Which top 10 authors have the most books in the database

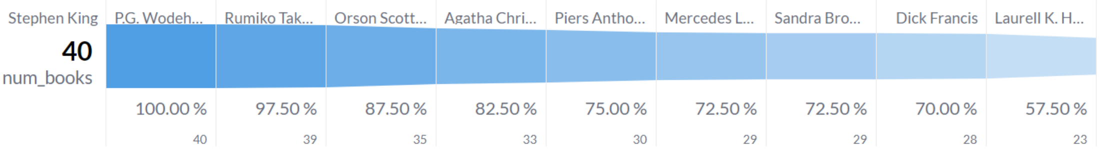


Figure 17: Authors with most books in DB

Answer: As you can see above visualized result, Stephen King and P.G Wodehouse are the top 2 authors having the most books in this database. They have each 40 books in this database.

DECISION RECOMMENDATION

The visualizations above make it very easy for us to decide which book to read. Bill Watterson's The Complete Calvin and Hobbs, published in 2005 with the highest rating of 4.82, is the first book to be recommended. In this next order, as the scatter graph shows, the second highest rating, Brandon Sanderson's 2014 published Words of Radiance, is recommended with a rating of 4.77. In this way, we found it very easy and simple to find recommendations for books.

CONNECTION BETWEEN VISUALIZATION AND ORIGINAL USE CASE

Once connected, with MySQL and Metabase, we learned that data can be easily visualized anytime, anywhere. That's why when we have new information that we want to get, we can easily visualize it through Metabase (for example, the year in which the most books were published, the books with the lowest ratings, the year in which the books to avoid were published, etc.). So, we've learned that this process is very useful for us in interpreting and using data. I'm sure that processing queries to get the data you need and using them to visualize the data in the metabase will be a stepping stone to your growth as a data scientist.

CONCLUSIONS & LESSONS LEARNED

In conclusion, the Database Management module has given us a thorough understanding of numerous strategies such as the ETL (Extract, convert, Load) process, which uses MYSQL to extract, clean, convert, and load data. We can run many types of queries using SQL metadata, such as calculating average ratings for books with more than 1000 ratings, identifying the highest rated books from different publishers, and determining which books have the highest ratings and the author of each book. These searches produce useful data for bookstores, publishers, and readers.

Also, by visualizing the results it helped us to provide excellent recommendations to readers, publishers and booksellers. Finally we evaluated the system’s efficiency and query performance by implementing optimization measures, to ensure the system operates at its maximum capacity. Hands-on experience extracting, processing, and loading data, as well as running queries and visualizing the results, allowed us to gain a better knowledge of and appreciation for the value of good data management. Database. The self learning actually pushed and motivated us so much to go further and do more research on understanding and learning the concepts. There is still a lot to learn in the core of Database management, but this module helped us to create to good base to learn further topics.

APPENDIX-1

Table of Figures

APPENDIX-2

Links to code and Files

Link: booksdb1.sql (db tables Creation only)

Link: GoodReads.sql (db tables and loading data)

Link: Queries (Queries sql code for answering research questions)

Link: Optimization 1.sql (optimization code)

Link: Optimization 2.sql (optimization code)

Link: goodreads_cleaned (Cleaned CSV file from Goodreads)

Link: goodreads_books_extracted (Extracted required columns from metadata for analysis)