Book Recommender

Kaveri Bhattacharya

July 2024

Business Use Case

In the highly competitive book retail industry, customer retention and personalized experiences are key drivers of success. A book recommender system can significantly enhance customer engagement, increase sales, and improve overall customer satisfaction. This system leverages data analytics and machine learning to provide personalized book recommendations based on user preferences. Personally, this project was interesting to me as I am an avid reader but often struggle to find the right titles for me. A book recommendation system that actually helps broaden my book horizons while taking my preferences into account would be something I would really appreciate.

Objectives

1. **Enhance User Experience:** By offering personalized book suggestions, a user can discover new titles that align with their interests, improving their overall book-search experience.
2. **Increase Sales:** Targeted recommendations can lead to higher conversion rates as customers are more likely to purchase books that are curated to their tastes.
3. **Client Retention:** Personalized recommendations foster a deeper connection with the client, encouraging repeat visits and loyalty to the platform.
4. **Predict User Preferences**: The system can predict which books users might enjoy based on their reading history/ preferences, helping users discover new books they are likely to appreciate.

High-level plan:

● Research project topic

● Data collection - API, dataset and scraping

● Project scope

● Project planning in Trello

● Exploratory data analysis in Python (data wrangling,

data cleaning & data visualization)

● Selection and creation of a database using MySQL

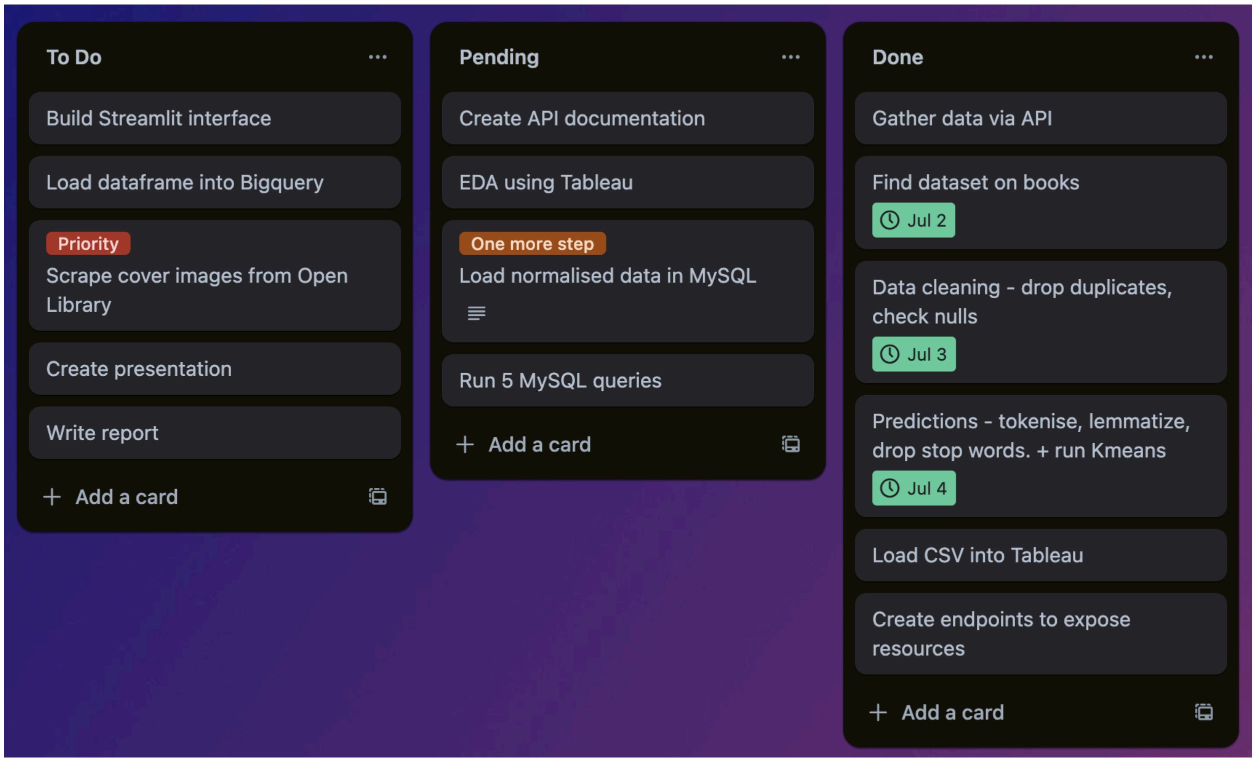
● Adding data to database and creating Entity Relationship Diagram

● Data manipulation in SQL

- Exposing data via API
- Visualization insights in Tableau
- Train and test models

Project Management

Trello board to manage daily project tasks



To Do
- Build Streamlit interface
- Load dataframe into Bigquery
- Priority — Scrape cover images from Open Library
- Create presentation
- Write report

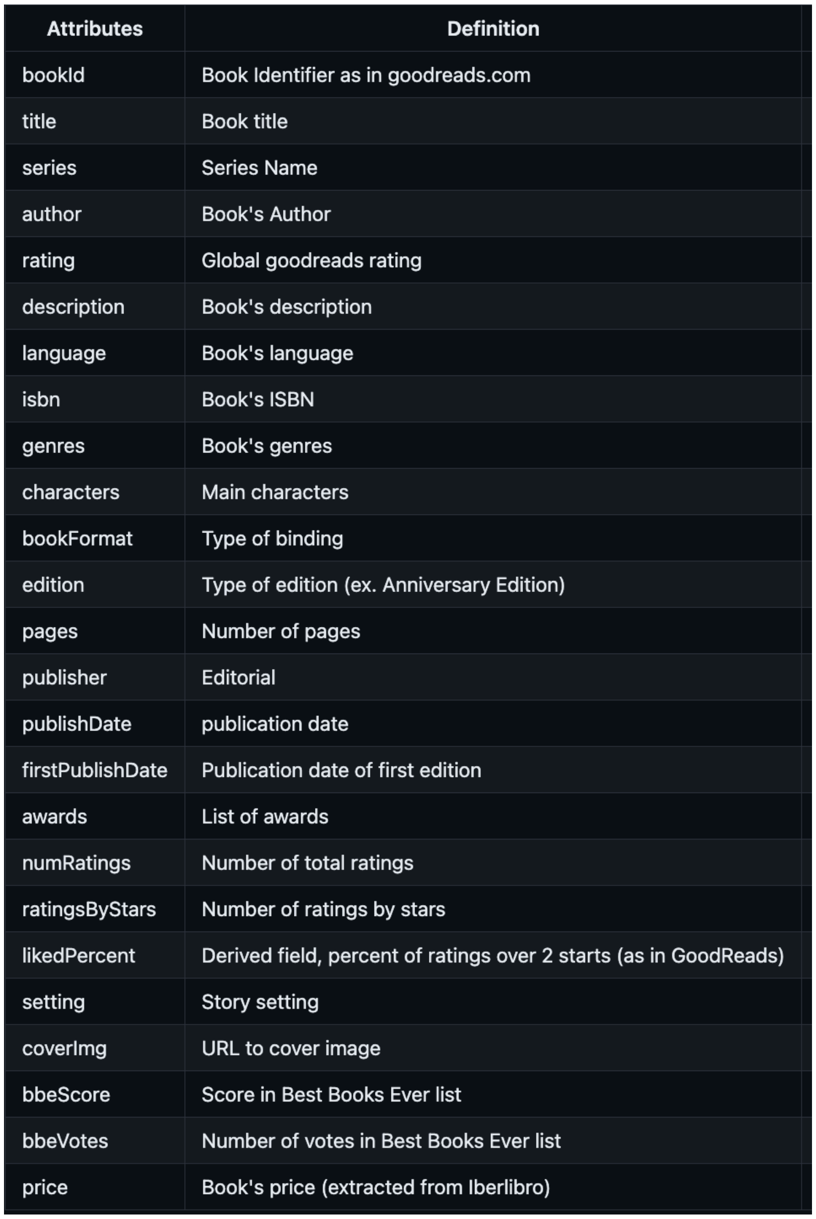
Pending
- Create API documentation
- EDA using Tableau
- One more step — Load normalised data in MySQL
- Run 5 MySQL queries

Done
- Gather data via API
- Find dataset on books — Jul 2
- Data cleaning - drop duplicates, check nulls — Jul 3
- Predictions - tokenise, lemmatize, drop stop words. + run Kmeans — Jul 4
- Load CSV into Tableau
- Create endpoints to expose resources

Data and data sources

1. Flat files

I found the dataset on github- It was a project that was created to retrieve data from the Best Books Ever list on Goodreads.com created by 2 students enrolled in the Master's Degree in Data Science of the Universitat Oberta de Catalunya. From this file I sourced 1 CSV: best_books.csv. It had 52,478 rows of books with 25 columns of data. Here is the metadata for this CSV:



Attributes	Definition
bookId	Book Identifier as in goodreads.com
title	Book title
series	Series Name
author	Book's Author
rating	Global goodreads rating
description	Book's description
language	Book's language
isbn	Book's ISBN
genres	Book's genres
characters	Main characters
bookFormat	Type of binding
edition	Type of edition (ex. Anniversary Edition)
pages	Number of pages
publisher	Editorial
publishDate	publication date
firstPublishDate	Publication date of first edition
awards	List of awards
numRatings	Number of total ratings
ratingsByStars	Number of ratings by stars
likedPercent	Derived field, percent of ratings over 2 starts (as in GoodReads)
setting	Story setting
coverImg	URL to cover image
bbeScore	Score in Best Books Ever list
bbeVotes	Number of votes in Best Books Ever list
price	Book's price (extracted from Iberlibro)

2. API

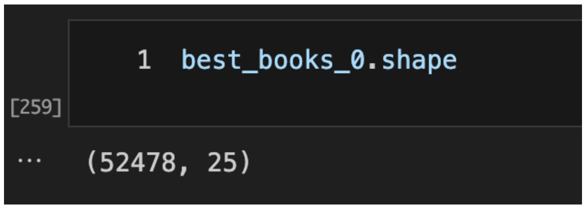
I used the API from Open Library to pull in ISBN, first_sentence of each book and goodreads_id which I thought I might need to work with the goodreads website, but I didn’t end up using as it was actually already in my source database. I got 5,935 rows and 6 columns of data from this data source.

3. Web Scraping:

I scraped cover images for the books in my database from Open Library to make up for the links in my original dataset that didn’t work. Now I have a complete list of cover images for all 52,424 rows (the number of rows I have after dropping duplicates)

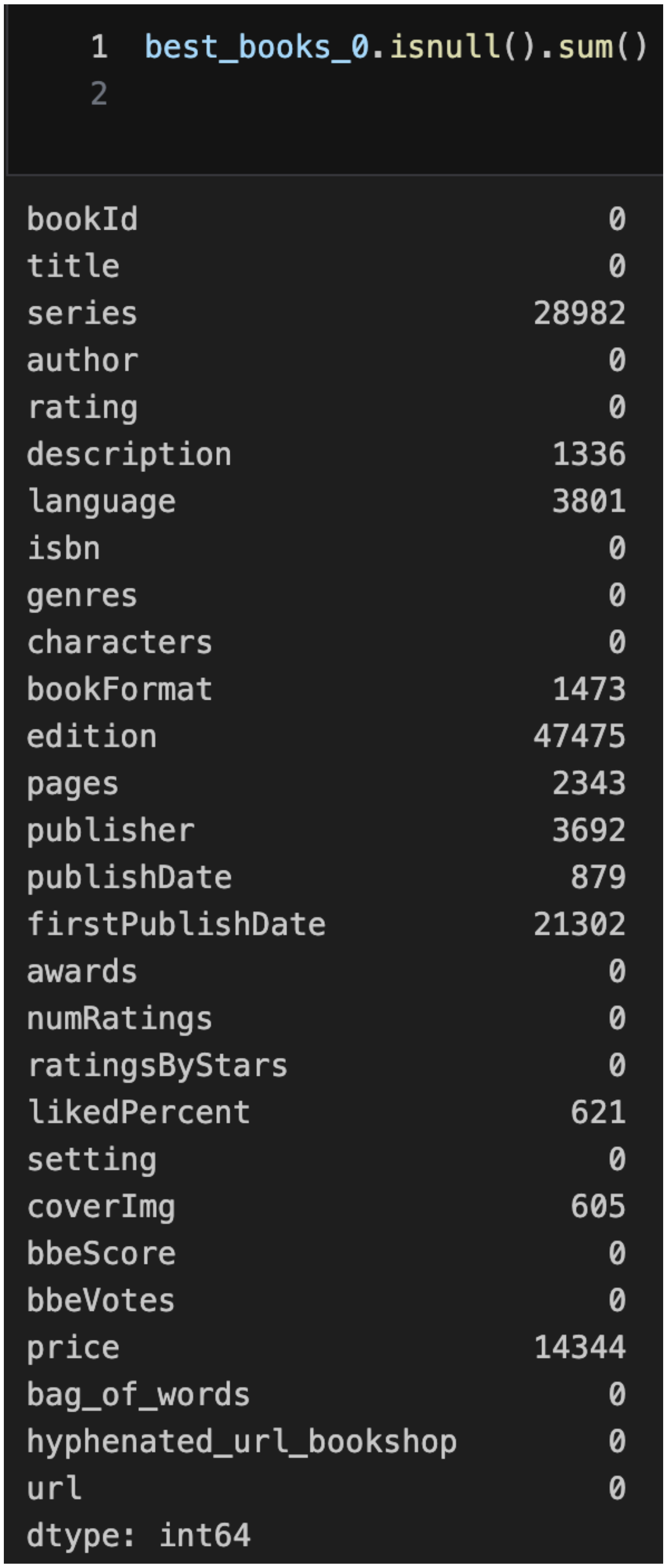
Data cleaning and Exploratory data analysis

Of the 52,478 rows of data from the flat file, there were 54 duplicated book ids, which I dropped. My resulting dataframe had 52,424 rows with 0 duplicates. When I checked for Null values, I found the edition, firstPublishDate and price columns had a number of Null values. I decided to leave them as they were as I am doing NLP where Null values don’t need to be handled. If I were doing a more distance based algorithm for ML, I would have handled Nulls differently. I have included some screenshots to demonstrate the shape of my data and the number of Nulls in my columns below.

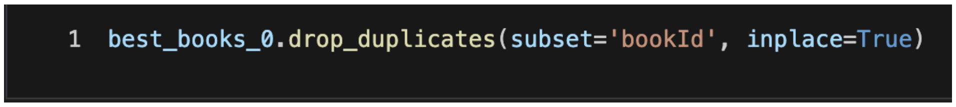


```
[259]
best_books_0.shape
```

```
(52478, 25)
```

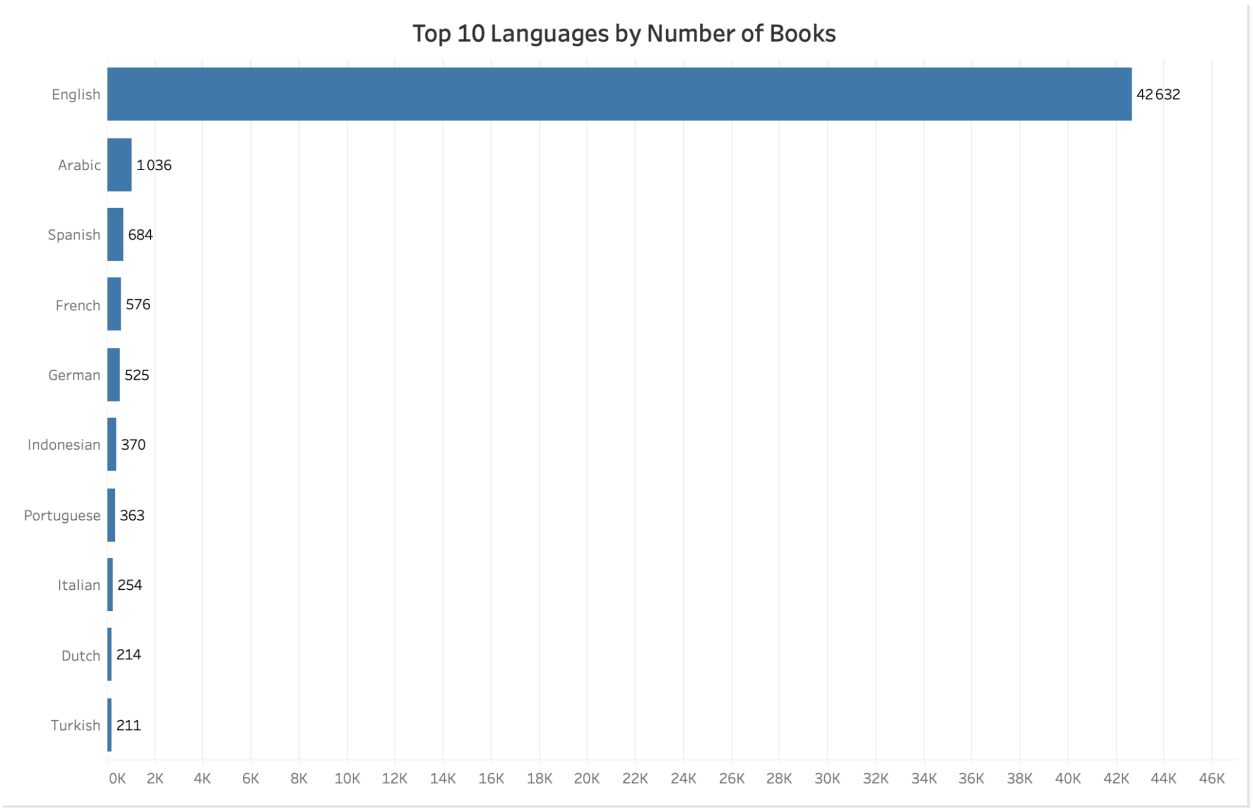

```
best_books_0.isnull().sum()
```

```
bookId                        0
title                         0
series                    28982
author                        0
rating                        0
description                1336
language                   3801
isbn                          0
genres                        0
characters                    0
bookFormat                 1473
edition                   47475
pages                      2343
publisher                  3692
publishDate                 879
firstPublishDate          21302
awards                        0
numRatings                    0
ratingsByStars                0
likedPercent                621
setting                       0
coverImg                    605
bbeScore                      0
bbeVotes                      0
price                     14344
bag_of_words                  0
hyphenated_url_bookshop       0
url                           0
dtype: int64
```

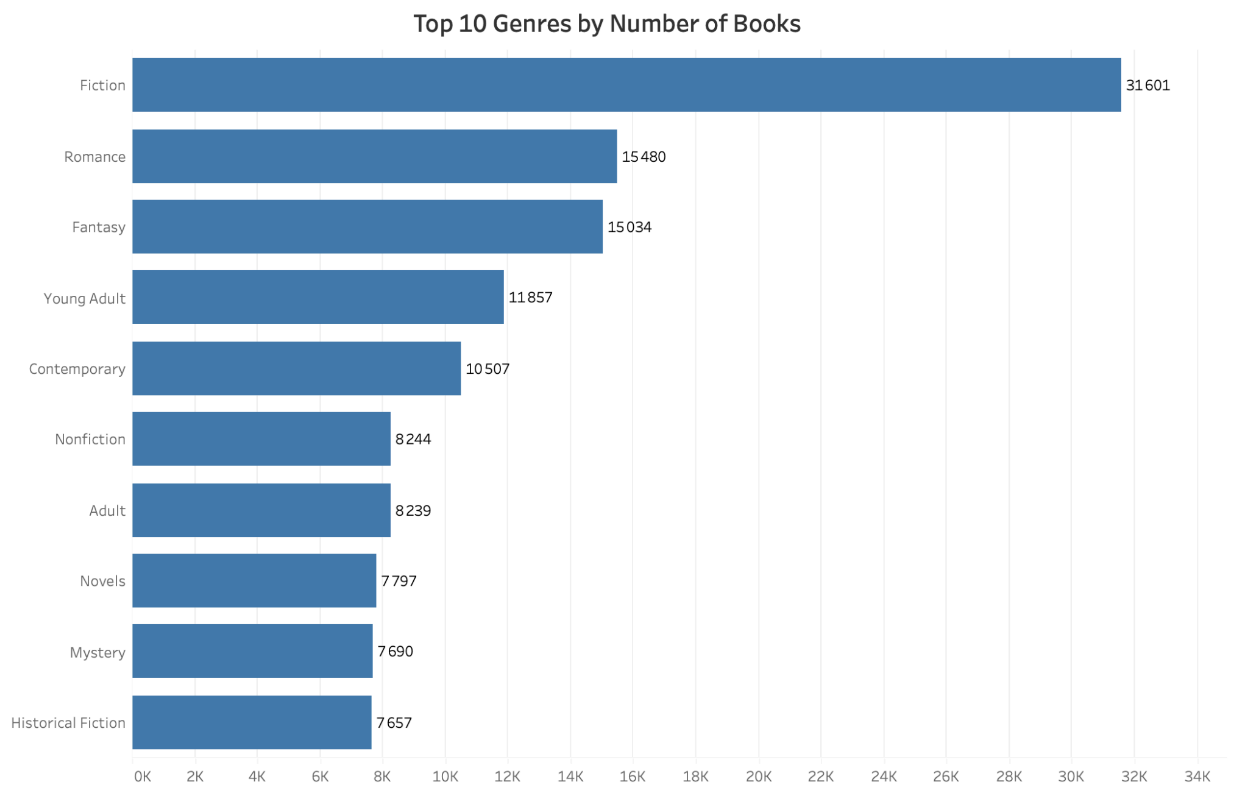


```python
best_books_0.drop_duplicates(subset='bookId', inplace=True)
```

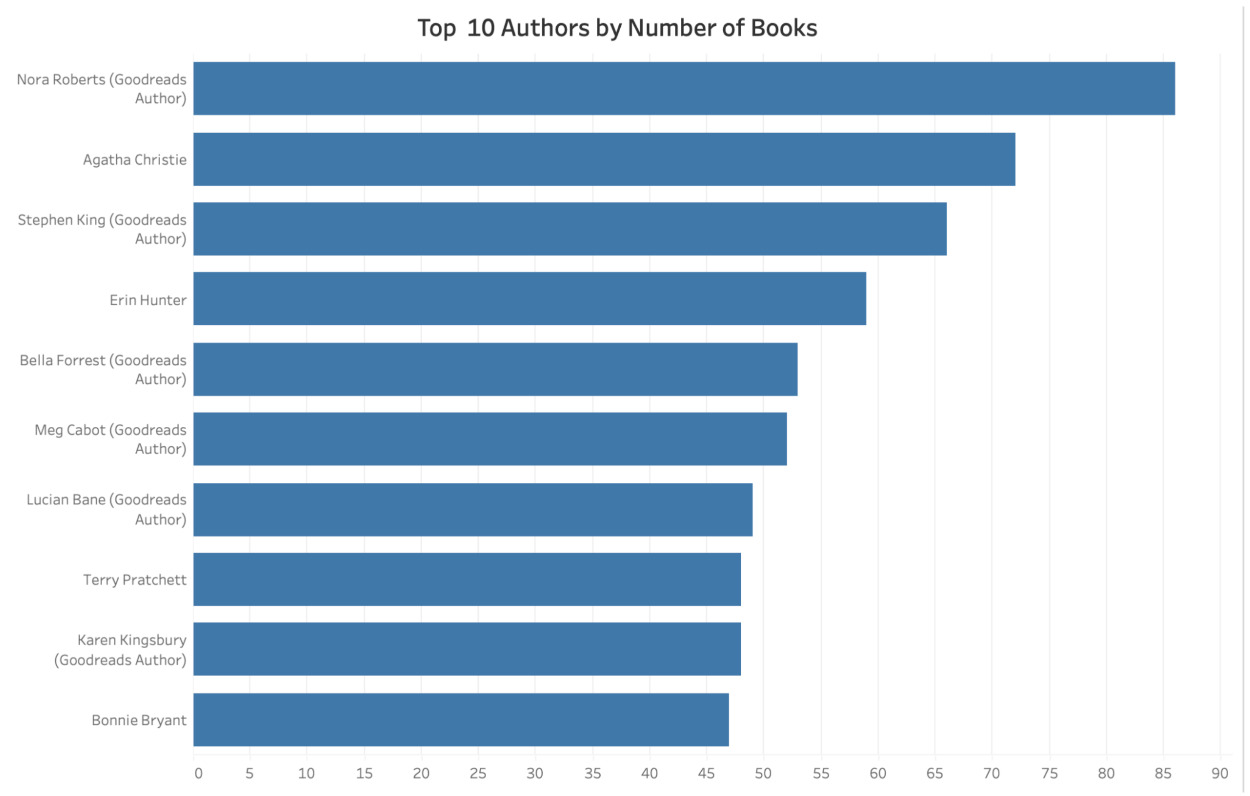
Visualizations using Tableau:



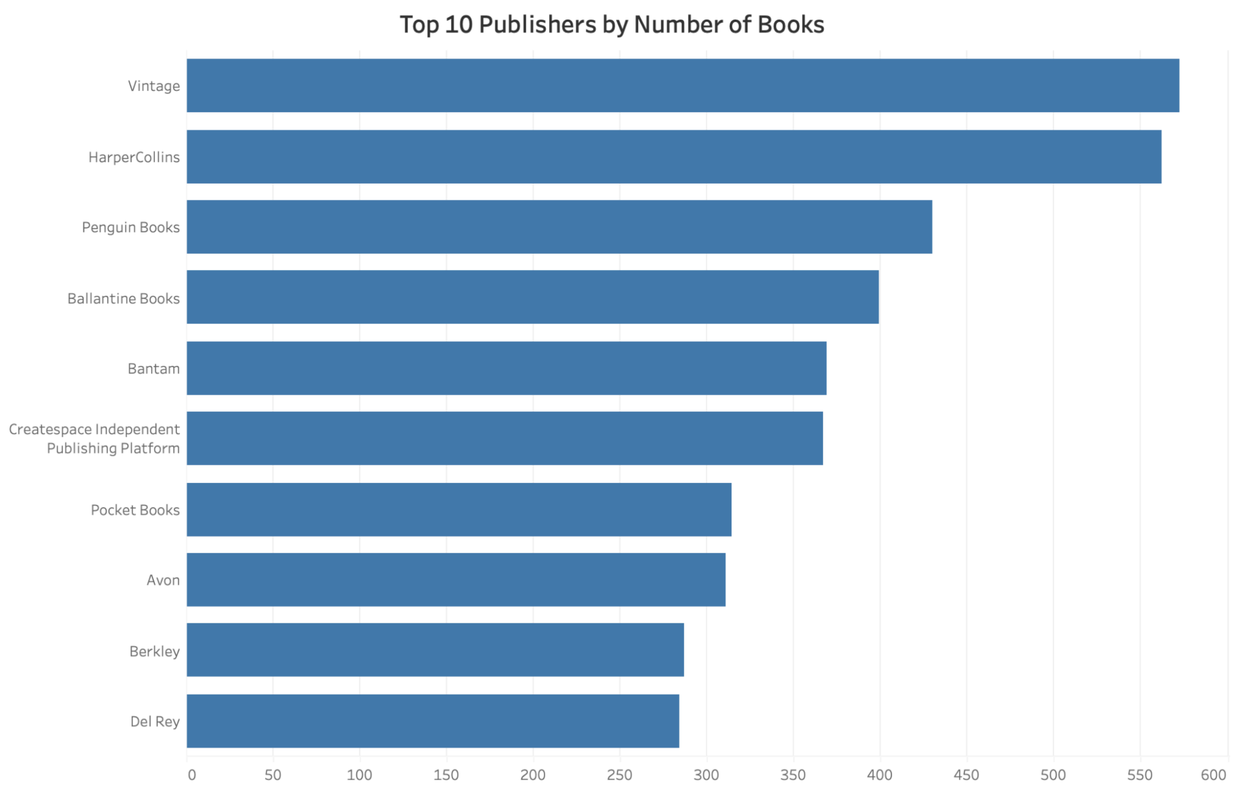
The vast majority of the books in the dataset were in English, which makes sense as this dataset is based on Goodreads data, an American social cataloging website and a subsidiary of Amazon.



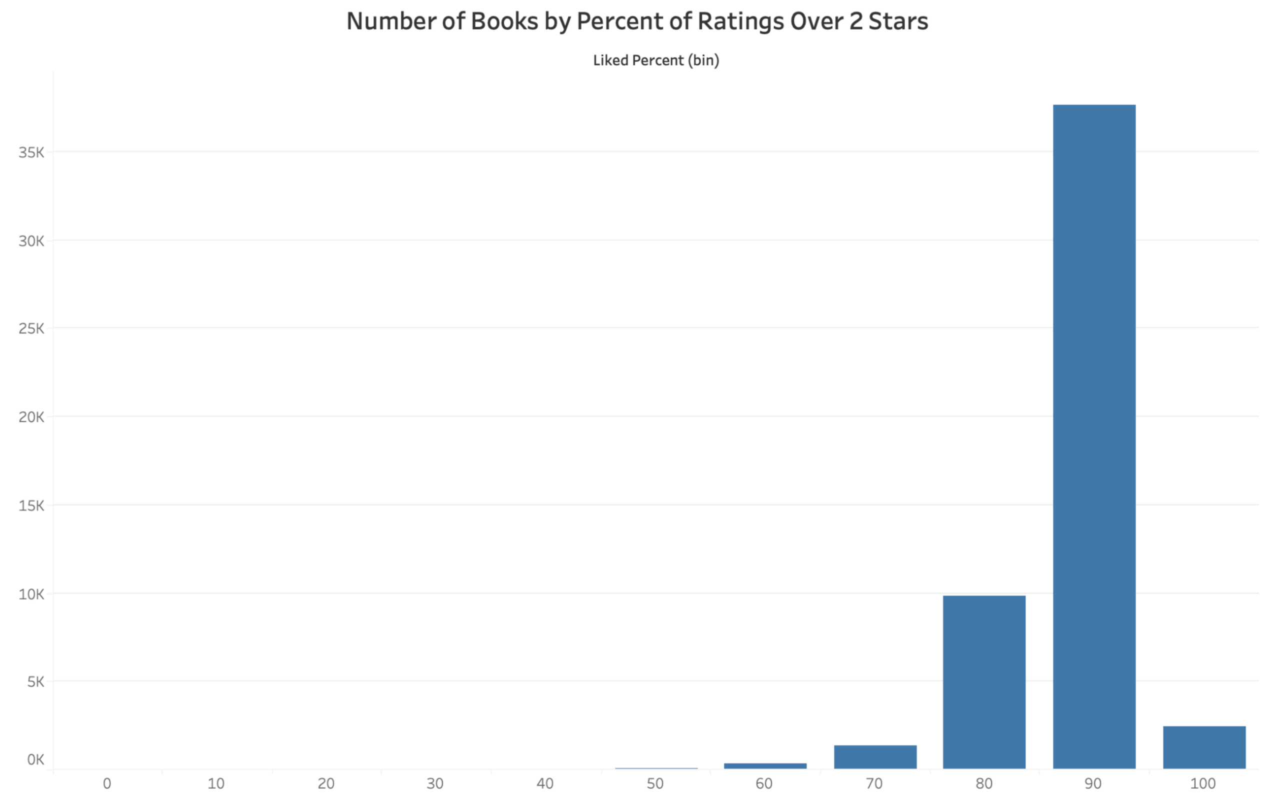
There are 981 distinct genres in the dataset. Books have many different genres, ‘Fiction’ being the most common. I ran a MySQL query to see if all books belonged to either ‘Fiction’, ‘Nonfiction’, ‘Historical Fiction’ or another overarching genre, and I found that 11,218 books do not belong to any genre with the string ‘fiction’ in it.



This chart shows the authors that have the greatest number of books in the database. All these authors are very prolific, for example, Nora Roberts has written 230 books, Agatha Christie has written 75 books, Stephen King, 68 books and Erin Hunter, 152 books.



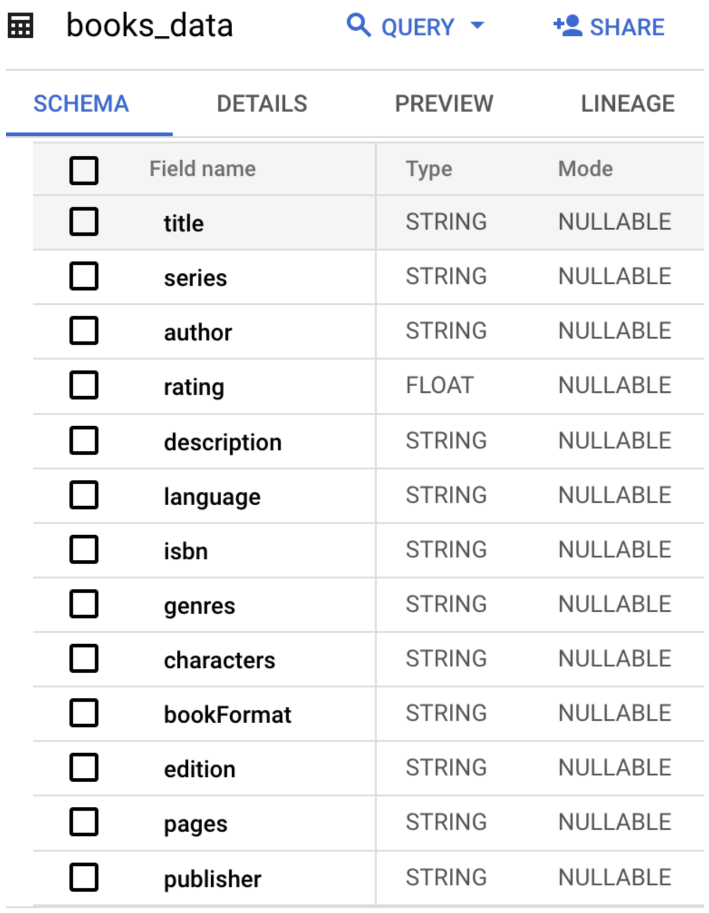
Vintage and Harper Collins are the top publishers of books in this dataset, followed by Penguin and Ballantine Books.



This chart shows the number of books with percent of ratings above 2 stars (out of 5). To help interpret this, it means that over 35K books have 90% of their ratings above the threshold of 2 stars.

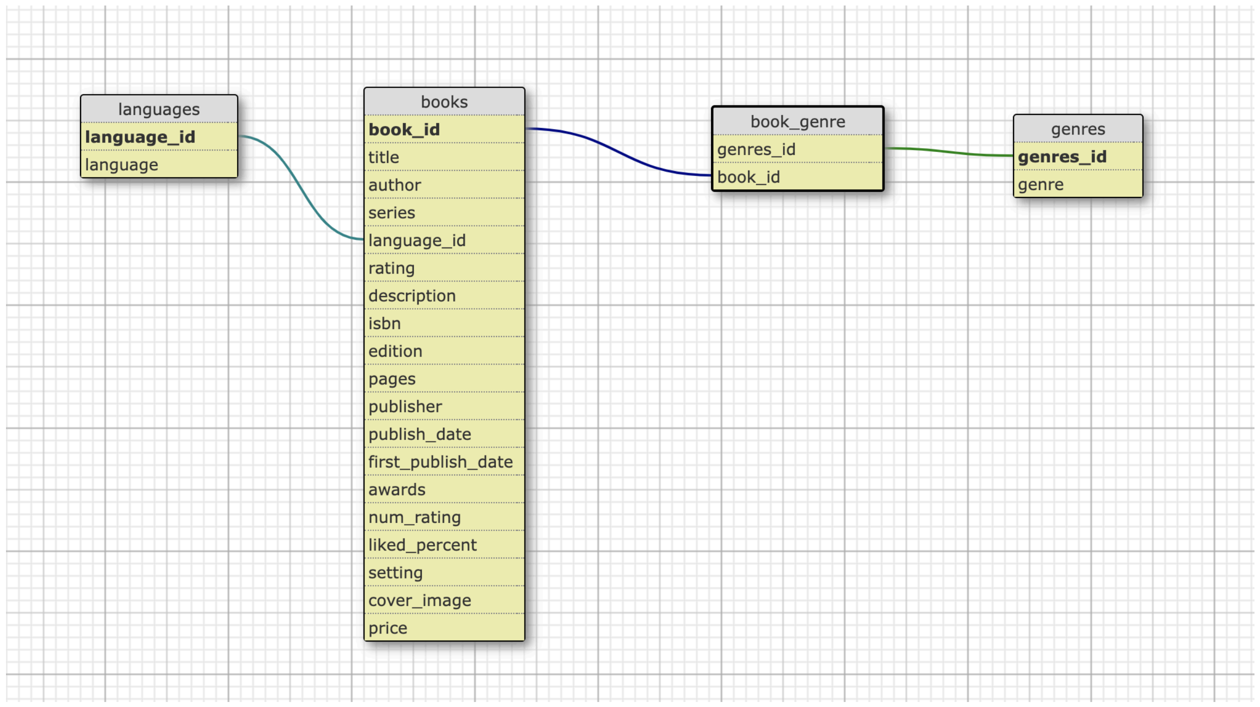
BigQuery

I uploaded my denormalized table to BigQuery. After ensuring my dataset was properly formatted and cleaned, I navigated to the BigQuery web console. From there, I selected my project and created a new dataset to house my table. With everything set up, I proceeded to upload my denormalized table directly from my local machine. BigQuery efficiently handled the upload, automatically detecting schema details such as field names and column types.



Field name	Type	Mode
title	STRING	NULLABLE
series	STRING	NULLABLE
author	STRING	NULLABLE
rating	FLOAT	NULLABLE
description	STRING	NULLABLE
language	STRING	NULLABLE
isbn	STRING	NULLABLE
genres	STRING	NULLABLE
characters	STRING	NULLABLE
bookFormat	STRING	NULLABLE
edition	STRING	NULLABLE
pages	STRING	NULLABLE
publisher	STRING	NULLABLE

Entity Relationship Diagram



API:

API Documentation

Overview

This API provides endpoints to interact with a collection of books stored in my MySQL database.

Endpoints

List Books

Endpoint: `/books`

Method: GET

Description: Retrieves a paginated list of books.

Query Parameters:

- `page`: (optional) Page number (default: 0)
- `page_size`: (optional) Number of books per page (default: 20, max: 20)
- `include_details`: (optional) Flag to include detailed information (default: 0)

Response:

- `books`: List of books with basic details.
- `next_page`: Link to the next page of books (if applicable).
- `last_page`: Link to the last page of books.

Error Responses:

- `500`: Internal Server Error

Get Book Details

Endpoint: `/books/{book_id}`

Method: GET

Description: Retrieves detailed information about a specific book identified by `book_id`.

Path Parameters:

- `book_id`: ID of the book to retrieve

Response:

- Detailed information about the book including title, author, series, rating, description, pages, language, and genres.

Error Responses:

- `404`: Book not found
- `500`: Internal Server Error

Notes

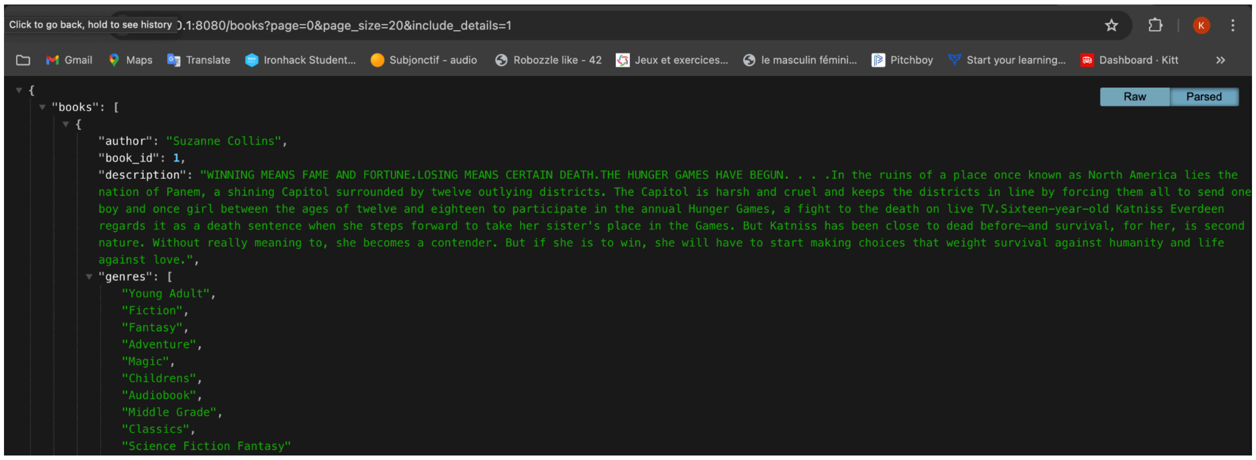
- **Database:** Connects to a MySQL database named `final` on `localhost`.
- **Pagination:** Supports pagination with a default page size of 20 books per page.

- **Error Handling:** Returns appropriate HTTP status codes and error messages.

Usage Instructions

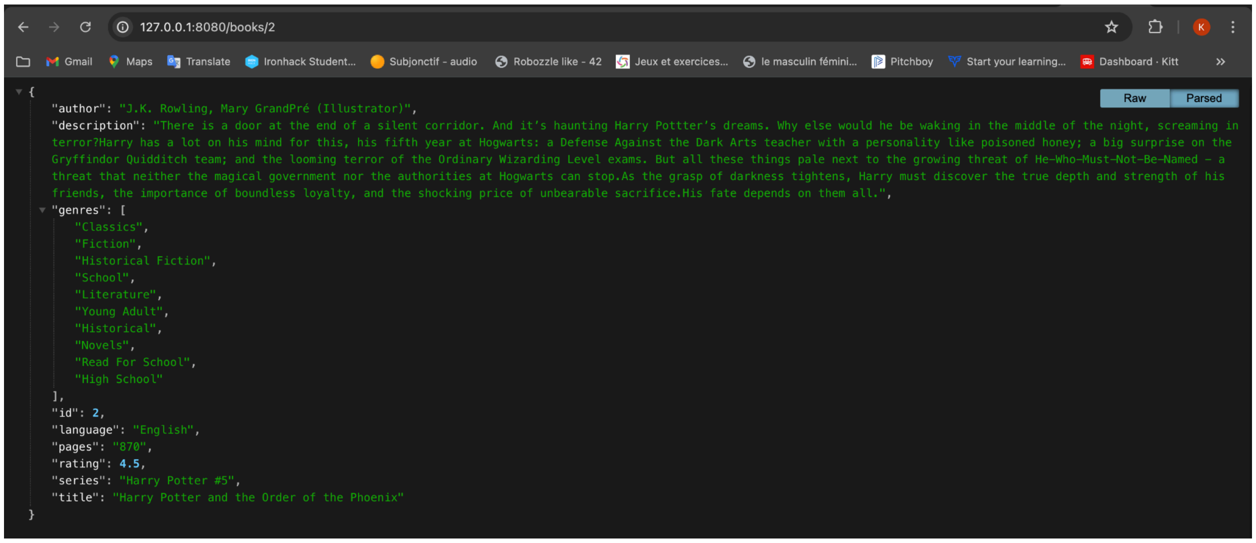
1. **List Books:**
    - Send a GET request to `/books` to retrieve a list of books.
    - Use query parameters `page`, `page_size`, and `include_details` for pagination and additional details.
2. **Get Book Details:**
    - Send a GET request to `/books/{book_id}` where `{book_id}` is the ID of the book you want to retrieve.
    - Replace `{book_id}` with the actual ID of the book.

Screenshot of endpoint: `/books` **with query parameters: page=0&page_size=20&include_details=1**



```
{
  "books": [
    {
      "author": "Suzanne Collins",
      "book_id": 1,
      "description": "WINNING MEANS FAME AND FORTUNE.LOSING MEANS CERTAIN DEATH.THE HUNGER GAMES HAVE BEGUN. . . .In the ruins of a place once known as North America lies the
      nation of Panem, a shining Capitol surrounded by twelve outlying districts. The Capitol is harsh and cruel and keeps the districts in line by forcing them all to send one
      boy and once girl between the ages of twelve and eighteen to participate in the annual Hunger Games, a fight to the death on live TV.Sixteen-year-old Katniss Everdeen
      regards it as a death sentence when she steps forward to take her sister's place in the Games. But Katniss has been close to dead before—and survival, for her, is second
      nature. Without really meaning to, she becomes a contender. But if she is to win, she will have to start making choices that weight survival against humanity and life
      against love.",
      "genres": [
        "Young Adult",
        "Fiction",
        "Fantasy",
        "Adventure",
        "Magic",
        "Childrens",
        "Audiobook",
        "Middle Grade",
        "Classics",
        "Science Fiction Fantasy"
```

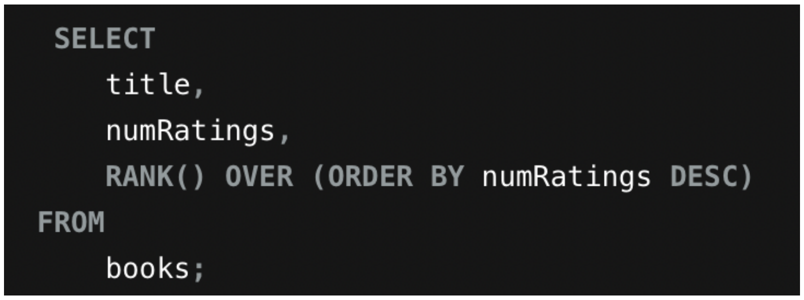
Screenshot of endpoint: `/books/{book_id}` **with path parameters:** `book_id`: 2


127.0.0.1:8080/books/2

```
{
    "author": "J.K. Rowling, Mary GrandPré (Illustrator)",
    "description": "There is a door at the end of a silent corridor. And it’s haunting Harry Pottter’s dreams. Why else would he be waking in the middle of the night, screaming in terror?Harry has a lot on his mind for this, his fifth year at Hogwarts: a Defense Against the Dark Arts teacher with a personality like poisoned honey; a big surprise on the Gryffindor Quidditch team; and the looming terror of the Ordinary Wizarding Level exams. But all these things pale next to the growing threat of He-Who-Must-Not-Be-Named - a threat that neither the magical government nor the authorities at Hogwarts can stop.As the grasp of darkness tightens, Harry must discover the true depth and strength of his friends, the importance of boundless loyalty, and the shocking price of unbearable sacrifice.His fate depends on them all.",
    "genres": [
        "Classics",
        "Fiction",
        "Historical Fiction",
        "School",
        "Literature",
        "Young Adult",
        "Historical",
        "Novels",
        "Read For School",
        "High School"
    ],
    "id": 2,
    "language": "English",
    "pages": "870",
    "rating": 4.5,
    "series": "Harry Potter #5",
    "title": "Harry Potter and the Order of the Phoenix"
}
```

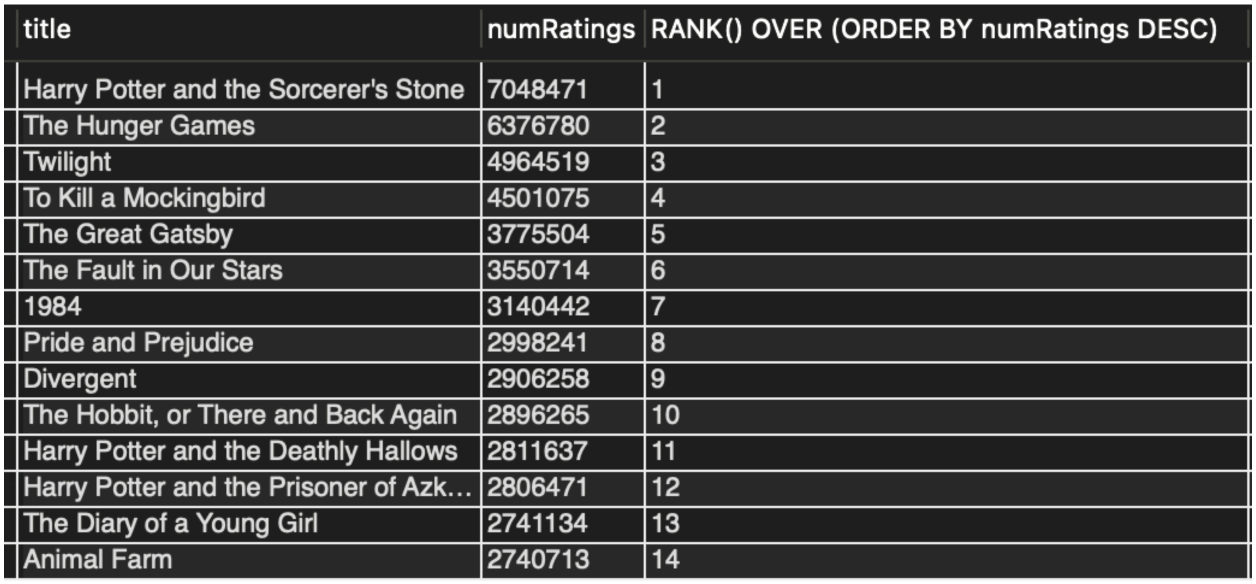
Data manipulation in SQL

Query 1:


```sql
  SELECT
     title,
     numRatings,
     RANK() OVER (ORDER BY numRatings DESC)
 FROM
     books;
```

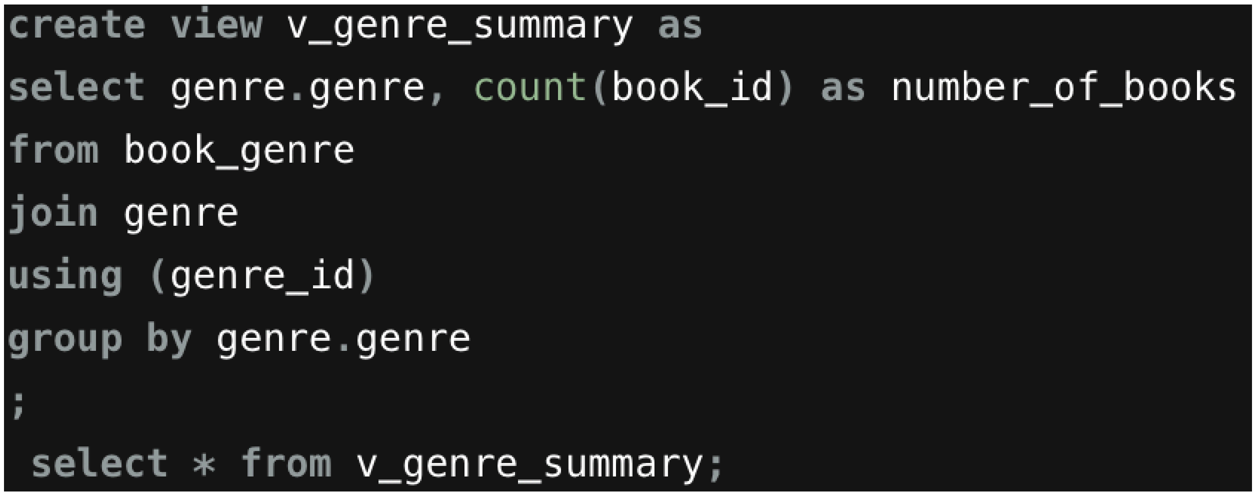
This query selects book titles and their corresponding number of ratings from the books table, while also computing and displaying the rank of each book based on the number of ratings in descending order. This is useful for identifying popular books and ranking items based on the number of ratings.

Result 1:



title	numRatings	RANK() OVER (ORDER BY numRatings DESC)
Harry Potter and the Sorcerer's Stone	7048471	1
The Hunger Games	6376780	2
Twilight	4964519	3
To Kill a Mockingbird	4501075	4
The Great Gatsby	3775504	5
The Fault in Our Stars	3550714	6
1984	3140442	7
Pride and Prejudice	2998241	8
Divergent	2906258	9
The Hobbit, or There and Back Again	2896265	10
Harry Potter and the Deathly Hallows	2811637	11
Harry Potter and the Prisoner of Azk…	2806471	12
The Diary of a Young Girl	2741134	13
Animal Farm	2740713	14

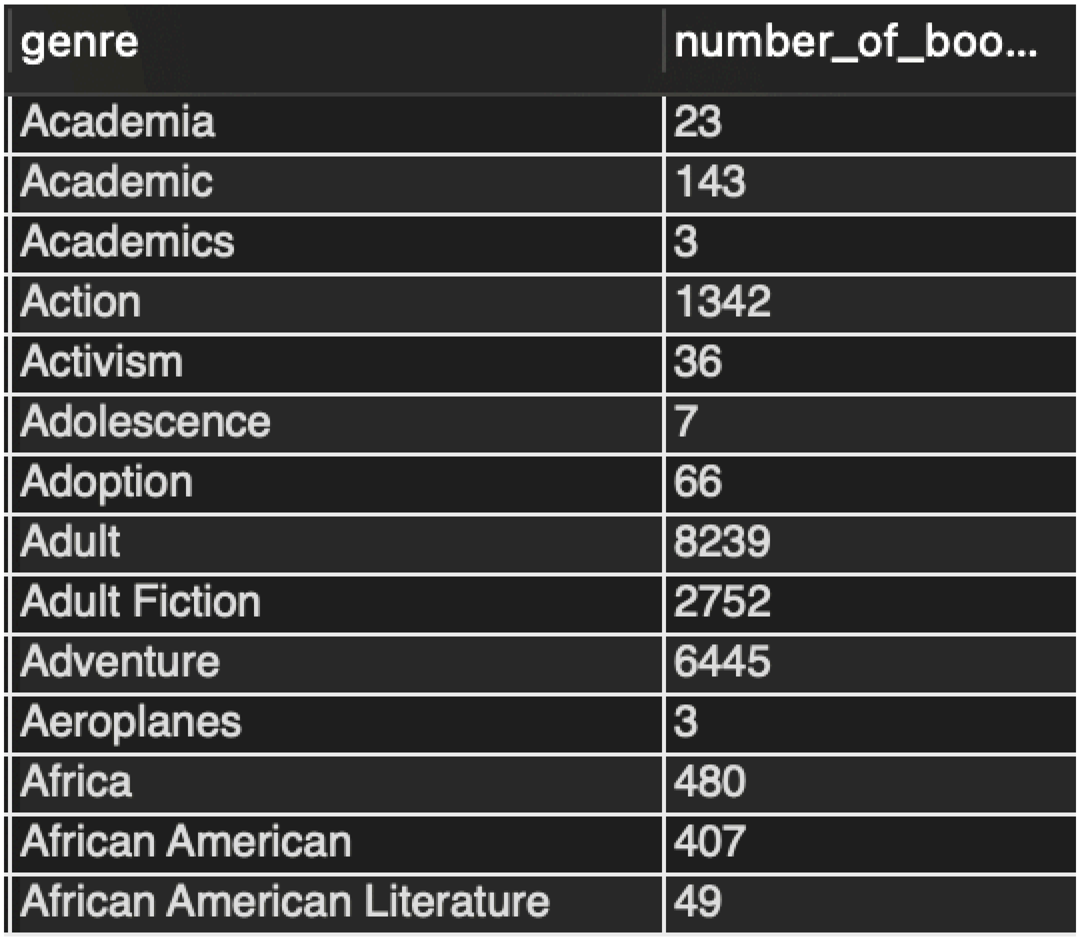
Query 2:



```
create view v_genre_summary as
select genre.genre, count(book_id) as number_of_books
from book_genre
join genre
using (genre_id)
group by genre.genre
;
 select * from v_genre_summary;
```

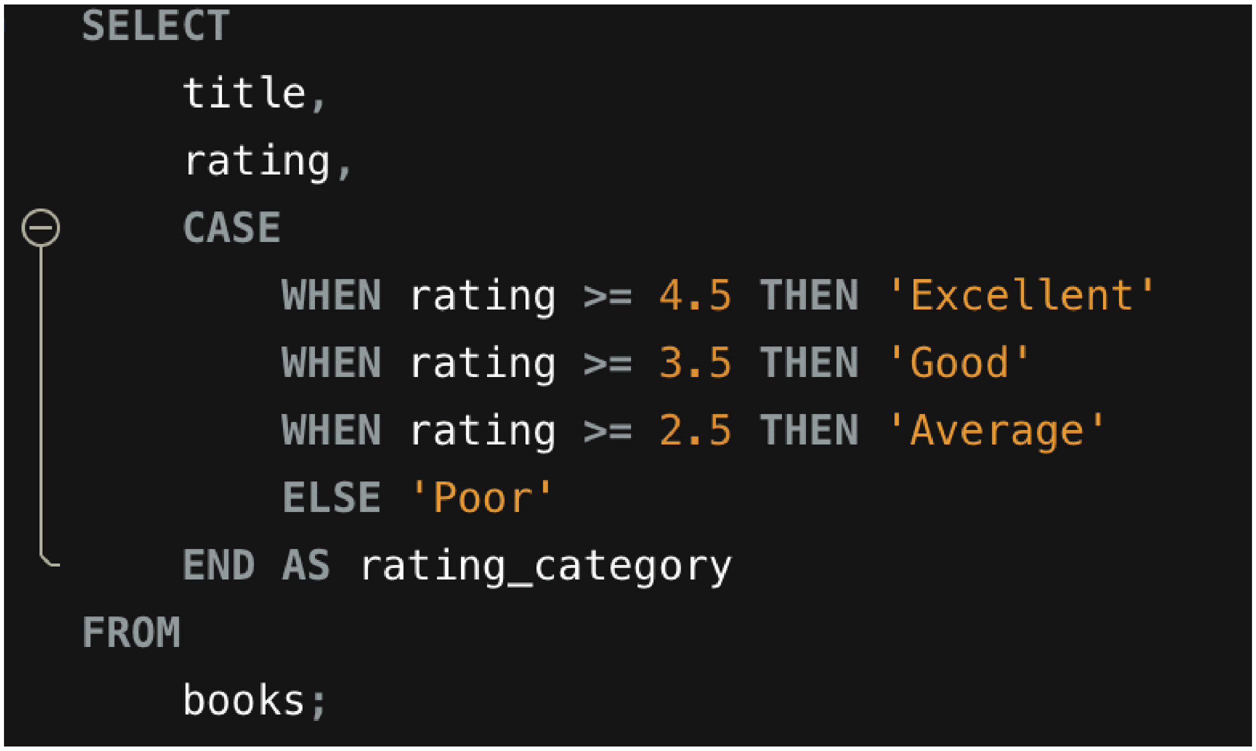
The view created in this query allowed me to quickly retrieve and analyze the distribution of books across different genres without needing to recompute the aggregation each time, enhancing efficiency in querying genre-related statistics.

Result 2:



genre	number_of_boo...
Academia	23
Academic	143
Academics	3
Action	1342
Activism	36
Adolescence	7
Adoption	66
Adult	8239
Adult Fiction	2752
Adventure	6445
Aeroplanes	3
Africa	480
African American	407
African American Literature	49

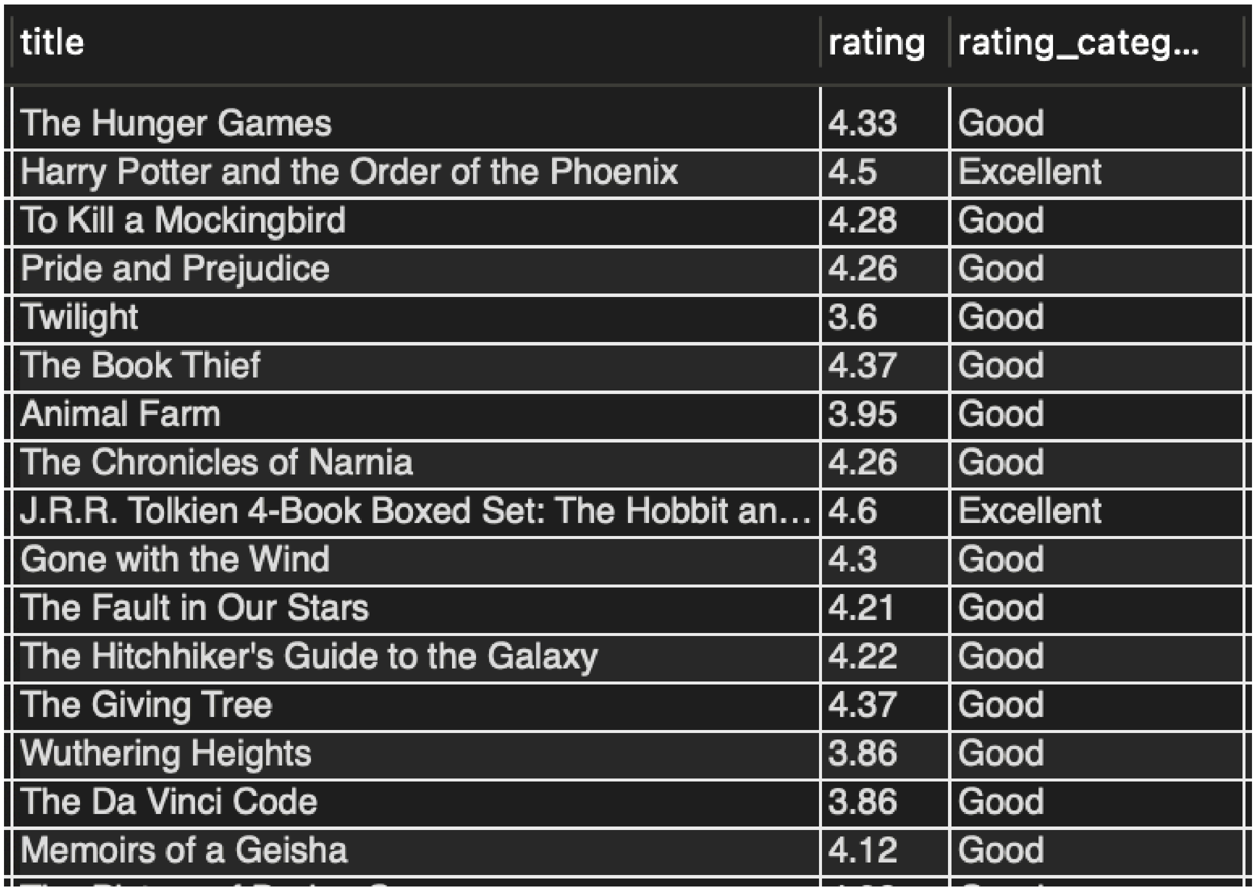
Query 3:



```
SELECT
    title,
    rating,
    CASE
        WHEN rating >= 4.5 THEN 'Excellent'
        WHEN rating >= 3.5 THEN 'Good'
        WHEN rating >= 2.5 THEN 'Average'
        ELSE 'Poor'
    END AS rating_category
FROM
    books;
```

This query provides a qualitative assessment of each book's `rating`, ‘Excellent’, ‘Good’, ‘Average’ or ‘Poor’, allowing for easier interpretation and analysis of rating distributions across the dataset.

Result 3:



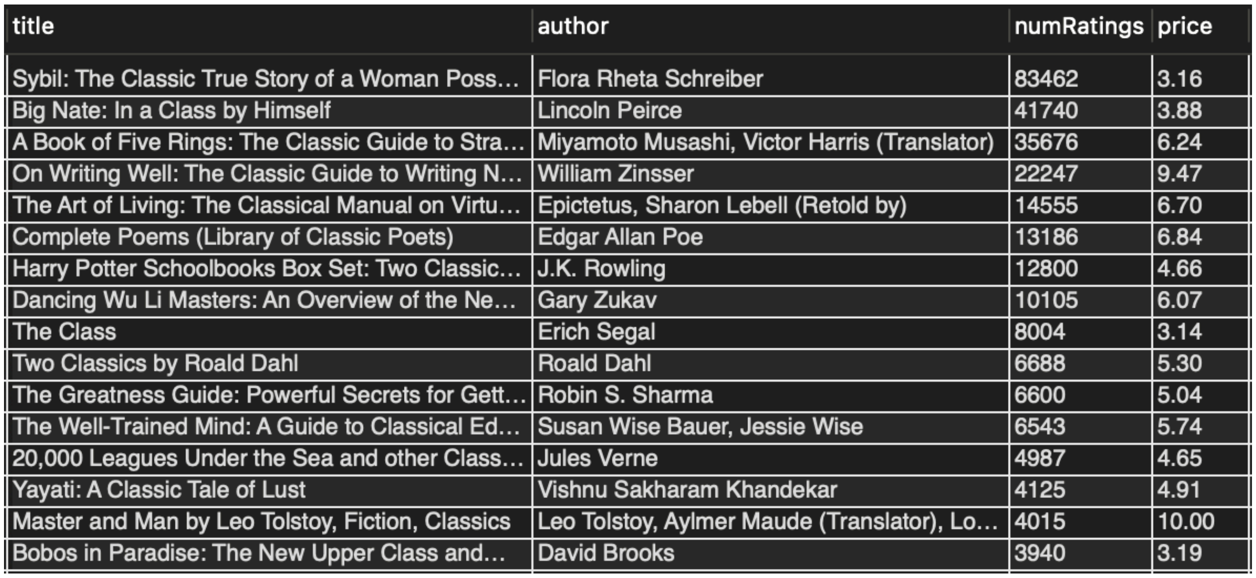
title	rating	rating_categ...
The Hunger Games	4.33	Good
Harry Potter and the Order of the Phoenix	4.5	Excellent
To Kill a Mockingbird	4.28	Good
Pride and Prejudice	4.26	Good
Twilight	3.6	Good
The Book Thief	4.37	Good
Animal Farm	3.95	Good
The Chronicles of Narnia	4.26	Good
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit an...	4.6	Excellent
Gone with the Wind	4.3	Good
The Fault in Our Stars	4.21	Good
The Hitchhiker's Guide to the Galaxy	4.22	Good
The Giving Tree	4.37	Good
Wuthering Heights	3.86	Good
The Da Vinci Code	3.86	Good
Memoirs of a Geisha	4.12	Good

Query 4:

```
SELECT title, author, numRatings, price
FROM books
WHERE numRatings > 1000
  AND price BETWEEN 3 AND 10
  AND title LIKE '%class%'
ORDER BY numRatings DESC;
```

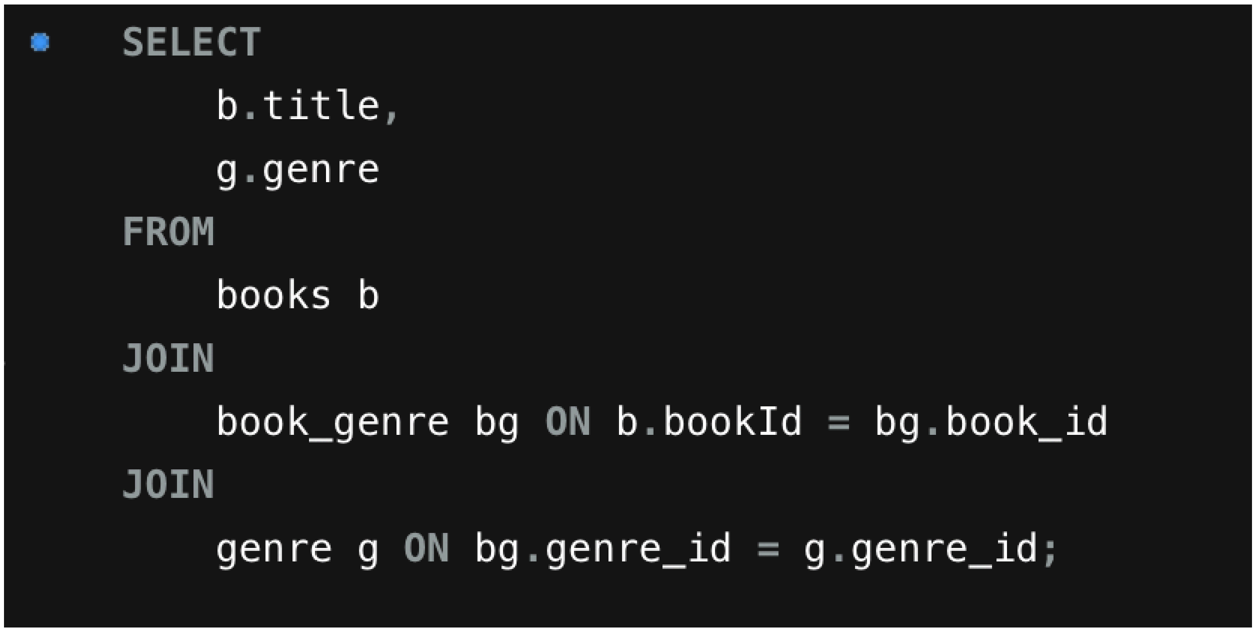
The SQL query retrieves a subset of data from the `books` table that meets specified criteria. It selects columns `title`, `author`, `numRatings`, and `price` where the following conditions are met: `numRatings` is greater than 1000, `price` falls within the range of 3 to 10, and the `title` contains the substring 'class'. The results are sorted in descending order based on `numRatings`, prioritizing books with higher ratings counts first. This query is designed to identify and prioritize books that are popular (`numRatings > 1000`), within a specified price range (`3 <= price <= 10`), and have 'class' in their title, facilitating targeted exploration and analysis of relevant books within the dataset.

Result 4:



title	author	numRatings	price
Sybil: The Classic True Story of a Woman Poss…	Flora Rheta Schreiber	83462	3.16
Big Nate: In a Class by Himself	Lincoln Peirce	41740	3.88
A Book of Five Rings: The Classic Guide to Stra…	Miyamoto Musashi, Victor Harris (Translator)	35676	6.24
On Writing Well: The Classic Guide to Writing N…	William Zinsser	22247	9.47
The Art of Living: The Classical Manual on Virtu…	Epictetus, Sharon Lebell (Retold by)	14555	6.70
Complete Poems (Library of Classic Poets)	Edgar Allan Poe	13186	6.84
Harry Potter Schoolbooks Box Set: Two Classic…	J.K. Rowling	12800	4.66
Dancing Wu Li Masters: An Overview of the Ne…	Gary Zukav	10105	6.07
The Class	Erich Segal	8004	3.14
Two Classics by Roald Dahl	Roald Dahl	6688	5.30
The Greatness Guide: Powerful Secrets for Gett…	Robin S. Sharma	6600	5.04
The Well-Trained Mind: A Guide to Classical Ed…	Susan Wise Bauer, Jessie Wise	6543	5.74
20,000 Leagues Under the Sea and other Class…	Jules Verne	4987	4.65
Yayati: A Classic Tale of Lust	Vishnu Sakharam Khandekar	4125	4.91
Master and Man by Leo Tolstoy, Fiction, Classics	Leo Tolstoy, Aylmer Maude (Translator), Lo…	4015	10.00
Bobos in Paradise: The New Upper Class and…	David Brooks	3940	3.19

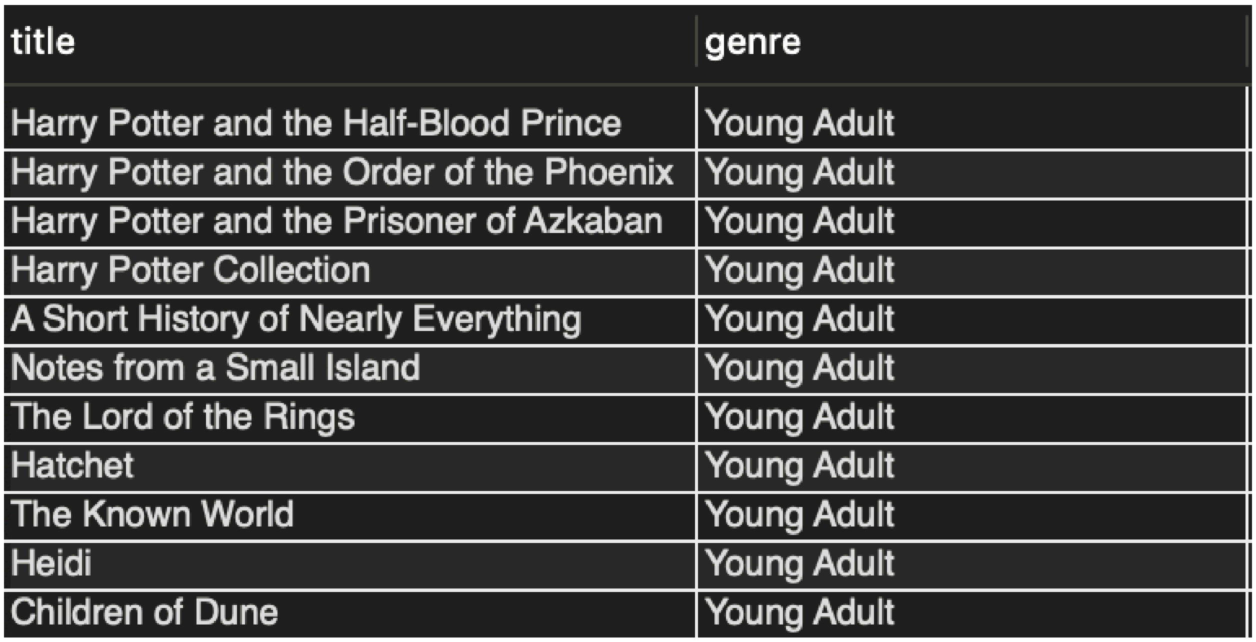
Query 5:



```
SELECT
    b.title,
    g.genre
FROM
    books b
JOIN
    book_genre bg ON b.bookId = bg.book_id
JOIN
    genre g ON bg.genre_id = g.genre_id;
```

This SQL query joins three tables (`books`, `book_genre`, and `genre`) to retrieve specific data elements. It selects columns `b.title` from the `books` table and `g.genre` from the `genre` table. The `JOIN` operations connect these tables based on their relationships: `books` is joined with `book_genre` using `b.bookId = bg.book_id`, linking each book to its associated genres stored in `book_genre`. Then, `book_genre` is joined with `genre` via `bg.genre_id = g.genre_id`, providing the actual genre names corresponding to each book. This query effectively combines information across multiple tables to fetch and display book titles along with their respective genres, facilitating comprehensive data retrieval and analysis based on genre categorization.

Result 5:



title	genre
Harry Potter and the Half-Blood Prince	Young Adult
Harry Potter and the Order of the Phoenix	Young Adult
Harry Potter and the Prisoner of Azkaban	Young Adult
Harry Potter Collection	Young Adult
A Short History of Nearly Everything	Young Adult
Notes from a Small Island	Young Adult
The Lord of the Rings	Young Adult
Hatchet	Young Adult
The Known World	Young Adult
Heidi	Young Adult
Children of Dune	Young Adult

Machine Learning

I conducted a machine learning analysis on the dataset that involved a bag of words approach after concatenating several pertinent columns. Initially, I performed tokenization to break down the text into individual tokens, followed by lemmatization to normalize the words to their base forms. Additionally, I removed stop words to focus only on meaningful content and applied a cleaning procedure to refine the text blob. Subsequently, I applied the KMeans clustering algorithm with specific parameters: KMeans(n_clusters=20, random_state=100). This algorithm partitioned the data into 20 distinct clusters based on the similarity of the text features, facilitating the exploration and identification of patterns within the dataset.

GDPR

After a thorough assessment of the data collected for this project, I can confidently affirm that no personal data has been utilized in any part of the processes. The sources of the data are entirely public, ensuring full transparency and accessibility.
Therefore, this project adheres to the guidelines and principles of the General Data

Protection Regulation (GDPR).

Sources:

https://github.com/scostap/goodreads_bbe_dataset?tab=readme-ov-file#dataset-information

https://en.wikipedia.org/wiki/Goodreads