Bookstore Database Design

Part 3

Manisha Goyal (mg7609)

Rhea Chandok (rc5397)

Sanam Palsule (sp7940)

Table of Contents

Physical Database Design and Deployment

The physical database design is a critical phase in the development of our bookstore's database system. It extends the logical design into a concrete implementation that considers the specific features and limitations of the chosen database management system (DBMS) and the deployment environment. For our bookstore, the physical design is tailored to MySQL hosted on Google Cloud, with an architecture that may also integrate local database capabilities.

Physical Database Design Considerations

Indexing

An index in a database is like a well-organized reference book that helps quickly locate information. We utilize indexing to optimize the performance of the database by reducing the amount of time required to access records. It makes searches faster by providing a roadmap for the database to find specific data without scanning through every page, improving overall performance. In our project, we have created an index for the ISBN of a book to make the book search faster

Query for Creating an Index:

CREATE INDEX idx_isbn ON Book (ISBN);

Query for Searching a book with ISBN:

SELECT * FROM books WHERE isbn = ‘9780446359207’;

Search before creating an index:

Search after creating an index:

Looking at the above screenshots, we can see that, even with a small amount of data, the time it took to run a specific search was different before and after creating an index. This tells us that having an index is crucial because it makes searches faster.

Physical Schema Modifications

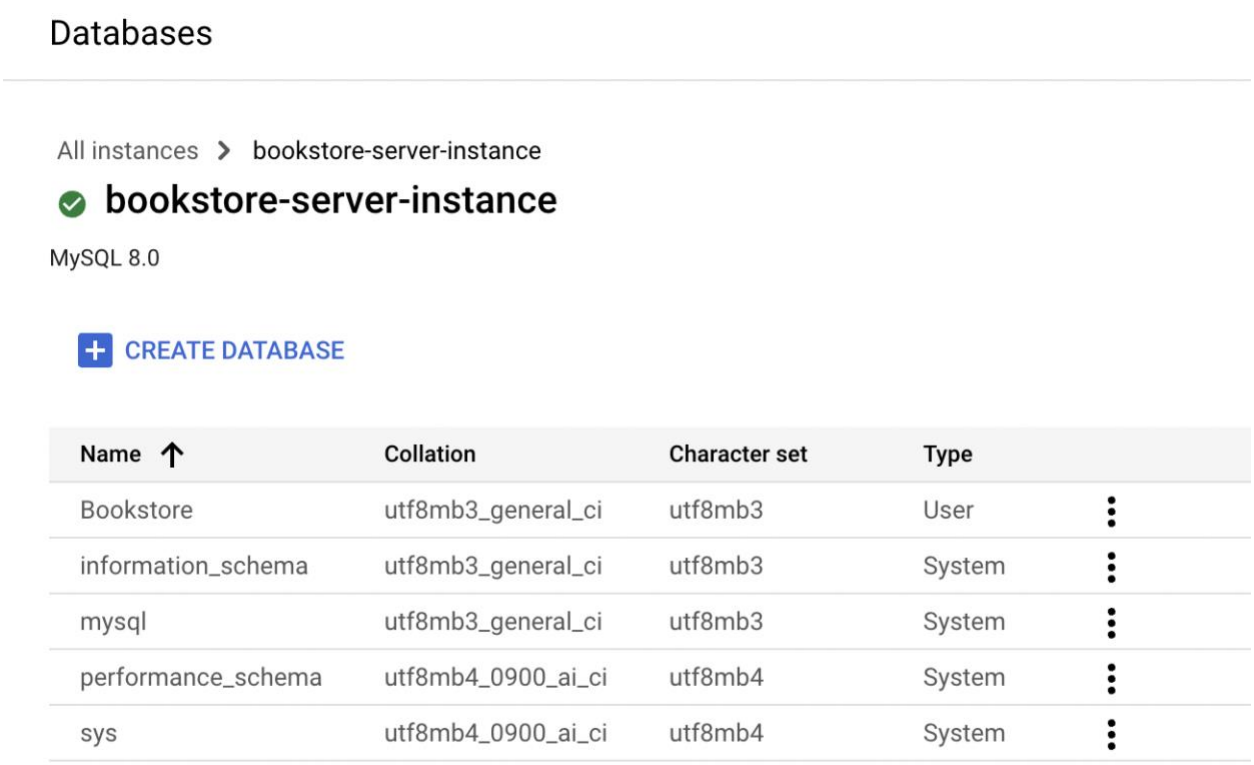
Denormalization: In select cases, denormalization has been applied to optimize read operations for specific queries that are critical to application performance.

Deployment on Google Cloud

Cloud-Based Infrastructure

Deployment: The physical database has been deployed on Google Cloud SQL, a fully managed database service that provides MySQL compatibility.

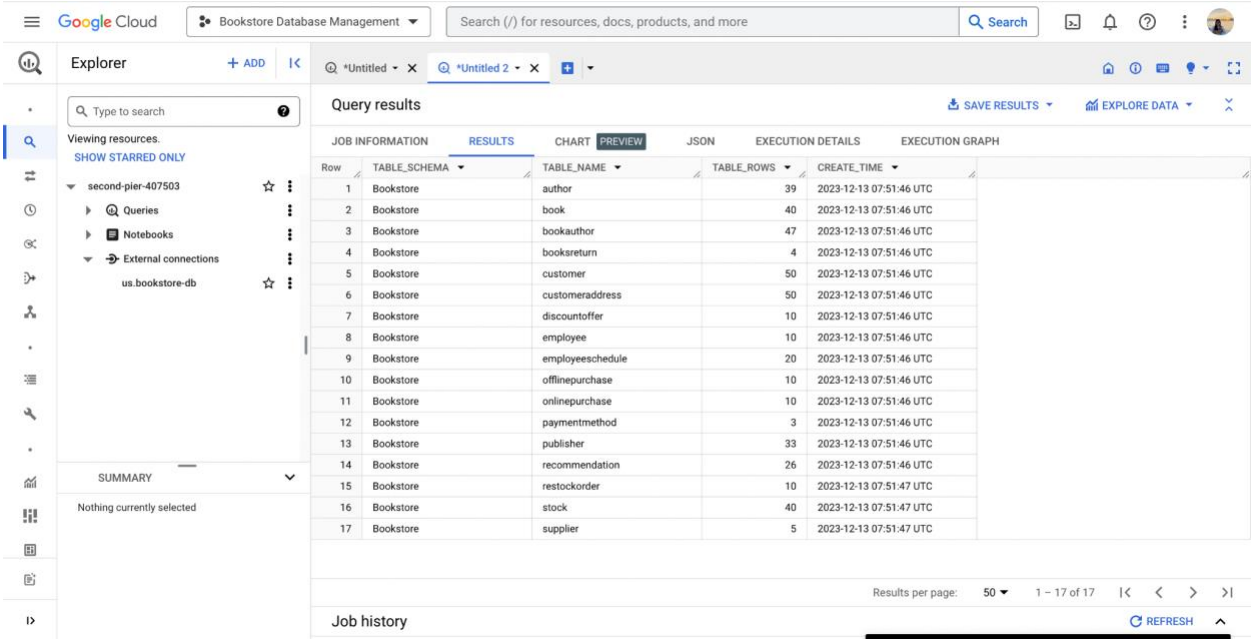
Advantages: This service offers high availability, scalability, and the convenience of managed backups and maintenance.

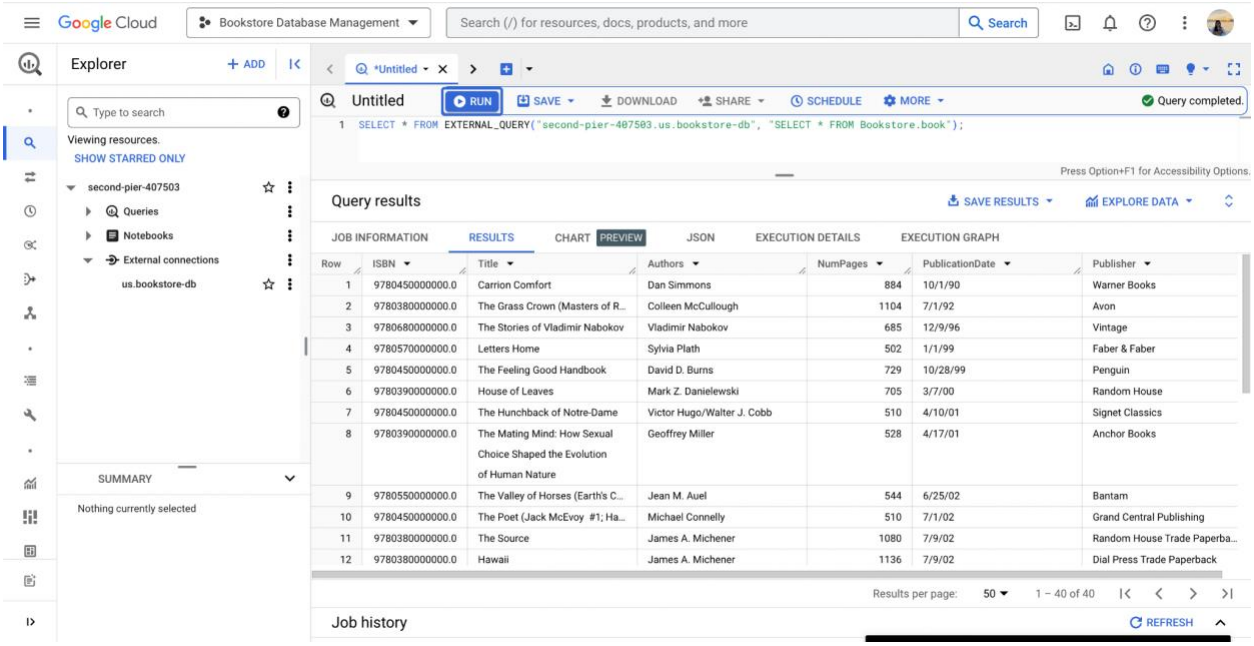


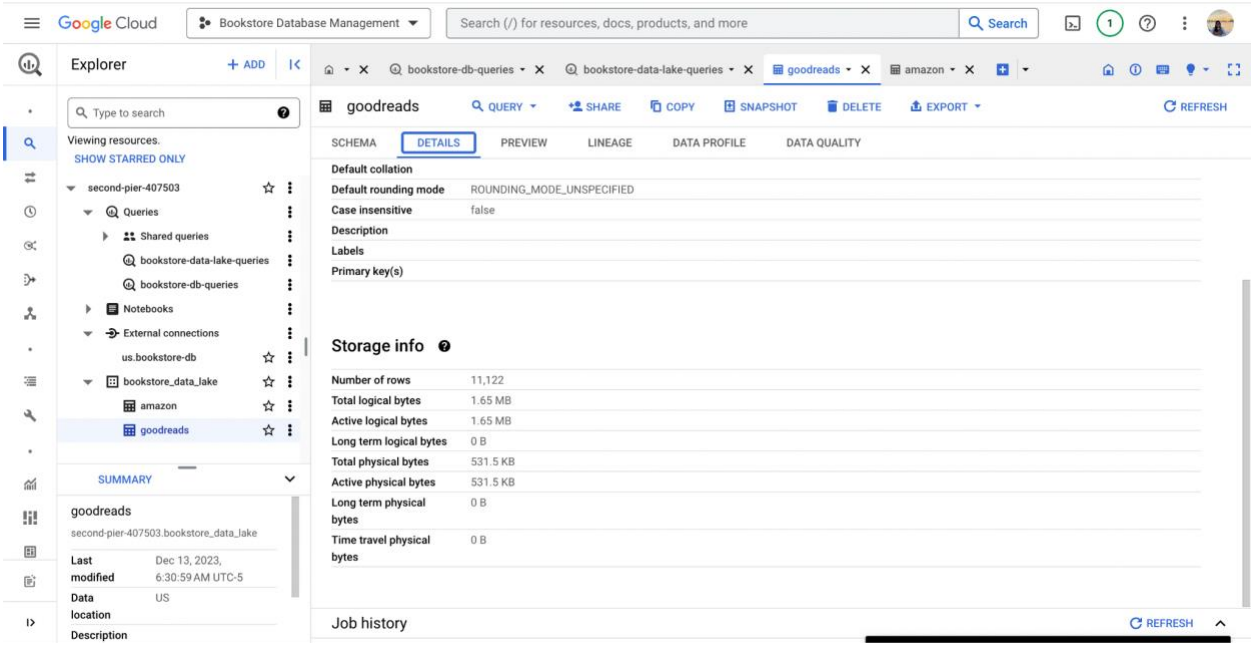
bookstore-db

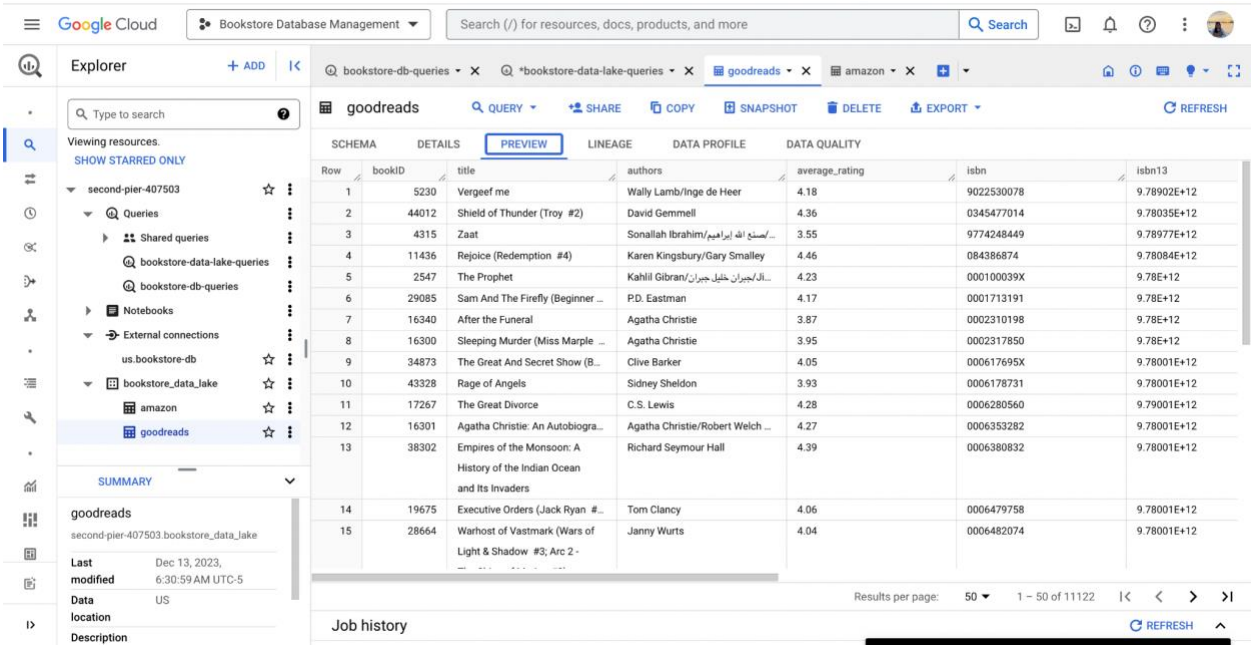
Connection info

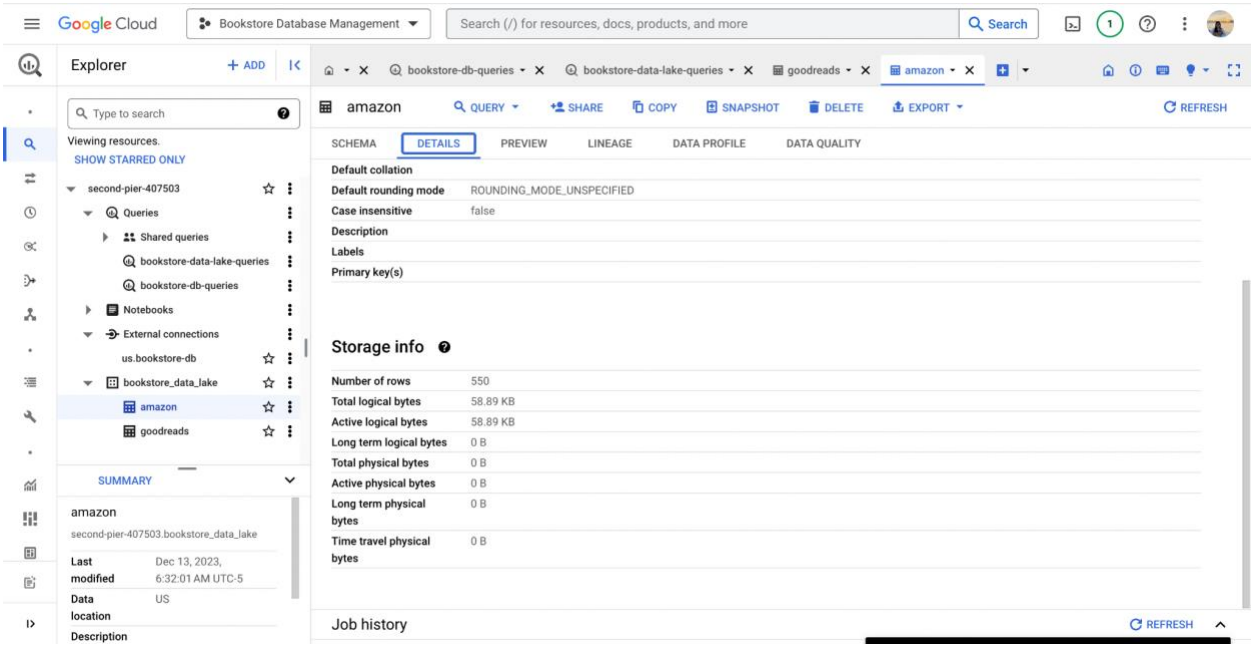
Connection ID	projects/second-pier-407503/locations/us/connections/bookstore-db
Friendly name	
Created	Dec 13, 2023, 3:54:02 AM UTC-5
Last modified	Dec 13, 2023, 3:54:02 AM UTC-5
Data location	us
Description	
Connection type	Cloud SQL - MySQL
Cloud SQL connection name	second-pier-407503:us-central1:bookstore-server-instance
Database name	Bookstore
Has credential	Yes
Service account id	service-417431979409@gcp-sa-bigqueryconnection.iam.gserviceaccount.com

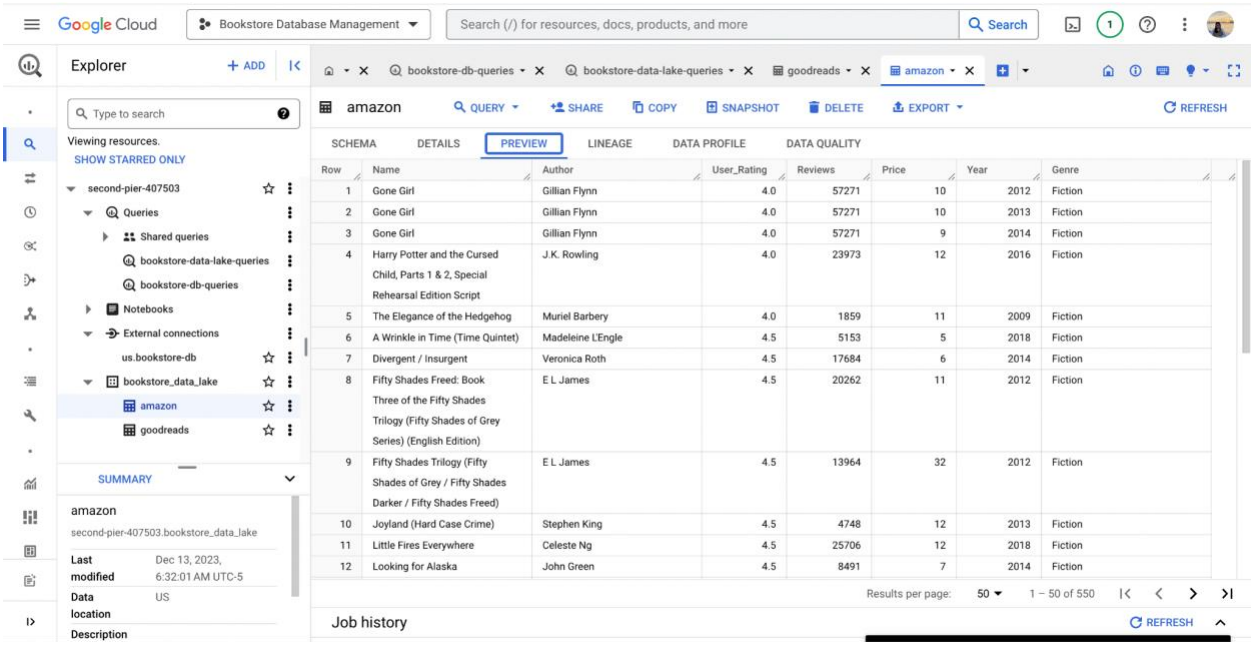



Google Cloud
Bookstore Database Management
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Type to search
Viewing resources.
SHOW STARRED ONLY
second-pier-407503
Queries
Notebooks
External connections
us.bookstore-db
SUMMARY
Nothing currently selected
*Untitled
Untitled
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
MORE
Query completed.
1 SELECT * FROM EXTERNAL_QUERY("second-pier-407503.us.bookstore-db", "SELECT * FROM Bookstore.book");
Press Option+F1 for Accessibility Options.
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
PREVIEW
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row ISBN Title Authors NumPages PublicationDate Publisher
1 9780450000000.0 Carrion Comfort Dan Simmons 884 10/1/90 Warner Books
2 9780380000000.0 The Grass Crown (Masters of R... Colleen McCullough 1104 7/1/92 Avon
3 9780680000000.0 The Stories of Vladimir Nabokov Vladimir Nabokov 685 12/9/96 Vintage
4 9780570000000.0 Letters Home Sylvia Plath 502 1/1/99 Faber & Faber
5 9780450000000.0 The Feeling Good Handbook David D. Burns 729 10/28/99 Penguin
6 9780390000000.0 House of Leaves Mark Z. Danielewski 705 3/7/00 Random House
7 9780450000000.0 The Hunchback of Notre-Dame Victor Hugo/Walter J. Cobb 510 4/10/01 Signet Classics
8 9780390000000.0 The Mating Mind: How Sexual Choice Shaped the Evolution of Human Nature Geoffrey Miller 528 4/17/01 Anchor Books
9 9780550000000.0 The Valley of Horses (Earth's C... Jean M. Auel 544 6/25/02 Bantam
10 9780450000000.0 The Poet (Jack McEvoy #1; Ha... Michael Connelly 510 7/1/02 Grand Central Publishing
11 9780380000000.0 The Source James A. Michener 1080 7/9/02 Random House Trade Paperba...
12 9780380000000.0 Hawaii James A. Michener 1136 7/9/02 Dial Press Trade Paperback
Results per page: 50
1 – 40 of 40
Job history
REFRESH


Google Cloud
Bookstore Database Management
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Type to search
Viewing resources.
SHOW STARRED ONLY
second-pier-407503
Queries
Shared queries
bookstore-data-lake-queries
bookstore-db-queries
Notebooks
External connections
us.bookstore-db
bookstore_data_lake
amazon
goodreads
SUMMARY
goodreads
second-pier-407503.bookstore_data_lake
Last modified Dec 13, 2023, 6:30:59 AM UTC-5
Data location US
Description
bookstore-db-queries
bookstore-data-lake-queries
goodreads
amazon
goodreads
QUERY
SHARE
COPY
SNAPSHOT
DELETE
EXPORT
REFRESH
SCHEMA
DETAILS
PREVIEW
LINEAGE
DATA PROFILE
DATA QUALITY
Default collation
Default rounding mode ROUNDING_MODE_UNSPECIFIED
Case insensitive false
Description
Labels
Primary key(s)
Storage info
Number of rows 11,122
Total logical bytes 1.65 MB
Active logical bytes 1.65 MB
Long term logical bytes 0 B
Total physical bytes 531.5 KB
Active physical bytes 531.5 KB
Long term physical bytes 0 B
Time travel physical bytes 0 B
Job history
REFRESH

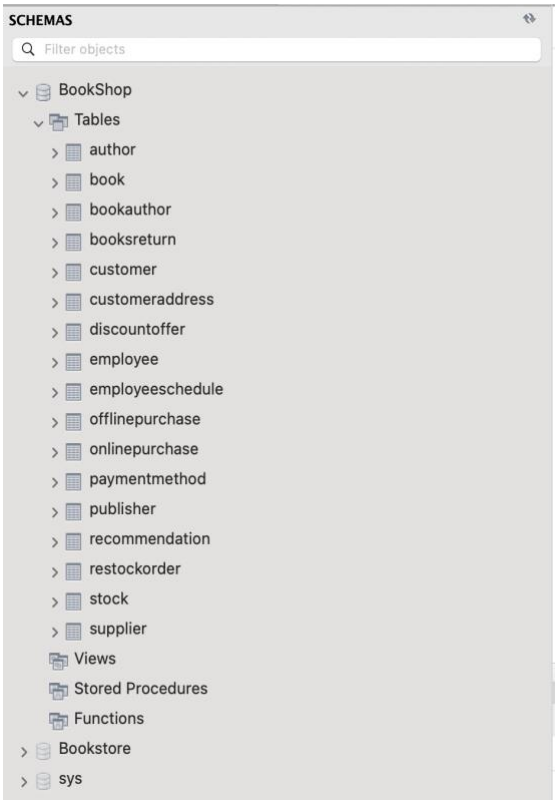

Google Cloud
Bookstore Database Management
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Type to search
Viewing resources.
SHOW STARRED ONLY
second-pier-407503
Queries
Shared queries
bookstore-data-lake-queries
bookstore-db-queries
Notebooks
External connections
us.bookstore-db
bookstore_data_lake
amazon
goodreads
SUMMARY
goodreads
second-pier-407503.bookstore_data_lake
Last modified Dec 13, 2023, 6:30:59 AM UTC-5
Data location US
Description
bookstore-db-queries
*bookstore-data-lake-queries
goodreads
amazon
QUERY
SHARE
COPY
SNAPSHOT
DELETE
EXPORT
REFRESH
SCHEMA
DETAILS
PREVIEW
LINEAGE
DATA PROFILE
DATA QUALITY
Row bookID title authors average_rating isbn isbn13
1 5230 Vergeef me Wally Lamb/Inge de Heer 4.18 9022530078 9.78902E+12
2 44012 Shield of Thunder (Troy #2) David Gemmell 4.36 0345477014 9.78035E+12
3 4315 Zaat Sonallah Ibrahim/إبراهيم/... 3.55 9774248449 9.78977E+12
4 11436 Rejoice (Redemption #4) Karen Kingsbury/Gary Smalley 4.46 084386874 9.78084E+12
5 2547 The Prophet Kahlil Gibran/جبران خليل جبران/... 4.23 000100039X 9.78E+12
6 29085 Sam And The Firefly (Beginner ... P.D. Eastman 4.17 0001713191 9.78E+12
7 16340 After the Funeral Agatha Christie 3.87 0002310198 9.78E+12
8 16300 Sleeping Murder (Miss Marple ... Agatha Christie 3.95 0002317850 9.78E+12
9 34873 The Great And Secret Show (B... Clive Barker 4.05 000617695X 9.78001E+12
10 43328 Rage of Angels Sidney Sheldon 3.93 0006178731 9.78001E+12
11 17267 The Great Divorce C.S. Lewis 4.28 0006280560 9.79001E+12
12 16301 Agatha Christie: An Autobiogra... Agatha Christie/Robert Welch ... 4.27 0006353282 9.78001E+12
13 38302 Empires of the Monsoon: A History of the Indian Ocean and Its Invaders Richard Seymour Hall 4.39 0006380832 9.78001E+12
14 19675 Executive Orders (Jack Ryan #... Tom Clancy 4.06 0006479758 9.78001E+12
15 28664 Warhost of Vastmark (Wars of Light & Shadow #3; Arc 2 - Janny Wurts 4.04 0006482074 9.78001E+12
Results per page: 50 1 – 50 of 11122
Job history
REFRESH


Google Cloud
Bookstore Database Management
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Type to search
Viewing resources.
SHOW STARRED ONLY
second-pier-407503
Queries
Shared queries
bookstore-data-lake-queries
bookstore-db-queries
Notebooks
External connections
us.bookstore-db
bookstore_data_lake
amazon
goodreads
SUMMARY
amazon
second-pier-407503.bookstore_data_lake
Last modified Dec 13, 2023, 6:32:01 AM UTC-5
Data location US
Description
bookstore-db-queries
bookstore-data-lake-queries
goodreads
amazon
QUERY
SHARE
COPY
SNAPSHOT
DELETE
EXPORT
REFRESH
SCHEMA
DETAILS
PREVIEW
LINEAGE
DATA PROFILE
DATA QUALITY
Default collation
Default rounding mode ROUNDING_MODE_UNSPECIFIED
Case insensitive false
Description
Labels
Primary key(s)
Storage info
Number of rows 550
Total logical bytes 58.89 KB
Active logical bytes 58.89 KB
Long term logical bytes 0 B
Total physical bytes 0 B
Active physical bytes 0 B
Long term physical bytes 0 B
Time travel physical bytes 0 B
Job history
REFRESH



Integration with Local Systems

Hybrid Approach: While the primary database is hosted in the cloud, a local MySQL instance is used for development and testing purposes. This ensures that the development cycle is efficient and that changes can be tested locally before deployment.



```
Limit to 1000 rows
1 • select * from bookshop.employee
```

100% 32:1

Result Grid Filter Rows: Search Export:

EmployeeID	FName	LName	DoB	Salary	Email	PhoneNo	JoiningDate	JobRole
1	Libbie	Sergean	11/11/75	$7,382.61	lsergean0@blogger.com	989-567-3031	1/23/12	Salesperson
2	Adeline	Bendan	12/9/22	$8,261.97	abendan1@ucsd.edu	592-424-1618	2/6/15	Salesperson
3	Shaylah	Knudsen	6/9/99	$7,529.10	sknudsen2@ovh.net	898-537-6304	1/24/16	Salesperson
4	Barbee	Brandham	1/1/13	$9,903.27	bbrandham3@parallels.com	794-625-4574	3/18/23	Salesperson
5	Marilee	Ludron	3/6/10	$9,176.01	mludron4@toplist.cz	137-858-9011	9/17/12	Salesperson
6	Jarib	Roj	5/15/21	$7,015.44	jroj5@google.de	717-635-8626	7/23/17	Salesperson
7	Lesley	Readshaw	3/26/86	$8,134.69	lreadshaw6@g.co	428-777-0924	2/3/11	Salesperson
8	Blondell	Dautry	2/8/10	$7,513.66	bdautry7@taobao.com	143-687-6433	4/14/14	Salesperson
9	Hew	Bolger	6/25/92	$9,607.19	hbolger8@ft.com	191-908-4931	11/8/18	Manager
10	Suzette	Falvey	12/25/95	$7,820.63	sfalvey9@hhs.gov	595-118-7946	9/1/13	Cashier

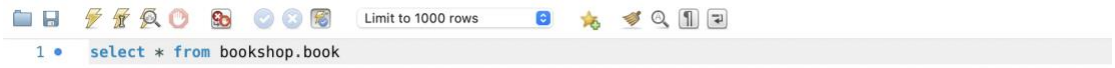
Query 1

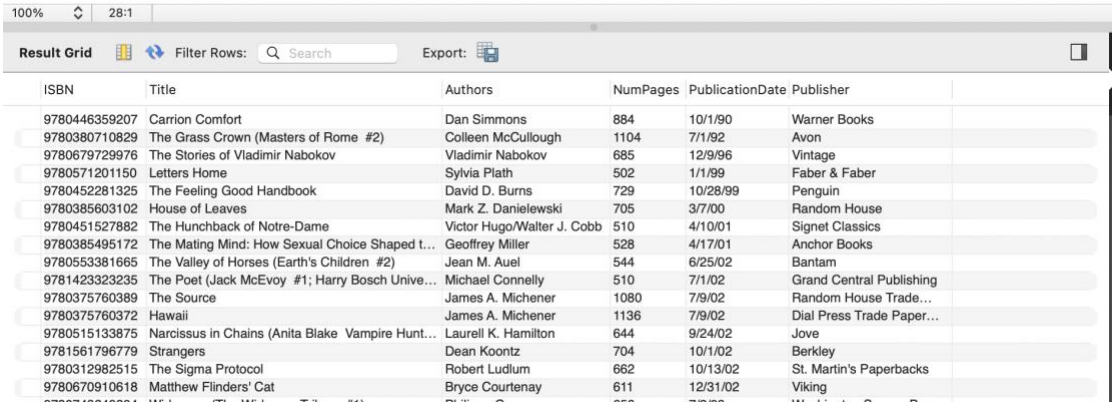
```
Limit to 1000 rows
1 • select * from bookshop.customer
```

100% 32:1

Result Grid Filter Rows: Search Export:

CustomerID	FName	LName	Email	PhoneNo
1	Joe	Scruton	jscruton0@nature.com	936-193-6112
2	Maryrose	Cyphus	mcyphus1@sakura.ne.jp	931-391-5818
3	Simonette	Brownstein	sbrownstein2@chron.com	266-266-4640
4	Paddy	Ismay	pismay3@cbslocal.com	365-880-8115
5	Talyah	Eglinton	teglinton4@angelfire.com	989-397-2845
6	Dame	Garrish	dgarrish5@epa.gov	832-825-1064
7	Abagail	MacCaull	amaccaull6@rakuten.co.jp	746-325-4829
8	Hube	Treherne	htreherne7@uol.com.br	654-611-8260
9	Raleigh	Moncey	rmoncey8@reddit.com	944-681-9709
10	Maxy	Edgeler	medgeler9@amazon.com	169-738-5445
11	Carolus	Buttfield	cbuttfielda@amazon.com	706-409-2409
12	Nan	Burdett	nburdettb@huffingtonpos…	801-519-4752
13	Kearney	Furnell	kfurnellc@sbwire.com	135-717-4055
14	Noellyn	Stout	nstoutd@blog.com	103-303-2150
15	Biddy	Stanlick	bstanlicke@zimbio.com	693-187-5767
16	Kevin	Goreway	kgorewayf@yahoo.com	872-864-2791
17	Colin	Smissen	csmisseng@domainmark…	228-758-0285





100% 28:1

Result Grid Filter Rows: Search Export:

ISBN	Title	Authors	NumPages	PublicationDate	Publisher
9780446359207	Carrion Comfort	Dan Simmons	884	10/1/90	Warner Books
9780380710829	The Grass Crown (Masters of Rome #2)	Colleen McCullough	1104	7/1/92	Avon
9780679729976	The Stories of Vladimir Nabokov	Vladimir Nabokov	685	12/9/96	Vintage
9780571201150	Letters Home	Sylvia Plath	502	1/1/99	Faber & Faber
9780452281325	The Feeling Good Handbook	David D. Burns	729	10/28/99	Penguin
9780385603102	House of Leaves	Mark Z. Danielewski	705	3/7/00	Random House
9780451527882	The Hunchback of Notre-Dame	Victor Hugo/Walter J. Cobb	510	4/10/01	Signet Classics
9780385495172	The Mating Mind: How Sexual Choice Shaped t…	Geoffrey Miller	528	4/17/01	Anchor Books
9780553381665	The Valley of Horses (Earth's Children #2)	Jean M. Auel	544	6/25/02	Bantam
9781423323235	The Poet (Jack McEvoy #1; Harry Bosch Unive…	Michael Connelly	510	7/1/02	Grand Central Publishing
9780375760389	The Source	James A. Michener	1080	7/9/02	Random House Trade…
9780375760372	Hawaii	James A. Michener	1136	7/9/02	Dial Press Trade Paper…
9780515133875	Narcissus in Chains (Anita Blake Vampire Hunt…	Laurell K. Hamilton	644	9/24/02	Jove
9781561796779	Strangers	Dean Koontz	704	10/1/02	Berkley
9780312982515	The Sigma Protocol	Robert Ludlum	662	10/13/02	St. Martin's Paperbacks
9780670910618	Matthew Flinders' Cat	Bryce Courtenay	611	12/31/02	Viking

Security Measures

Data Encryption: Data at rest and in transit is encrypted using Google Cloud's built-in security features.

Access Control: Access to the database is tightly controlled with firewall rules and authentication mechanisms to ensure that only authorized application servers and personnel can interact with the database.

Our physical database design for the bookstore application is meticulously crafted to provide optimal performance, scalability, and security, leveraging the capabilities of MySQL and the strengths of Google Cloud's infrastructure. This design lays a solid foundation for robust database operations and is equipped to handle the demands of our application, including the integration of machine learning model outputs and insights derived from unstructured data.

Business Use Cases for Workflow-Based Application

To support the workflow of our bookstore application, we have identified several business use cases that reflect the typical interactions between the system and its users—both employees and customers. These use cases have been documented to detail the processes and functionalities available through our application.

Business Use Cases Identification

Employee Use Cases

Employee Login: Employees start their interaction with the system by logging in to the application.

Restock Books: Once logged in, employees can update inventory levels. This use case involves identifying stock for refresh, updating the database, and potentially triggering orders for new stock.

Add New Employee: Managers can add new staff members to the system.

Customer Use Cases

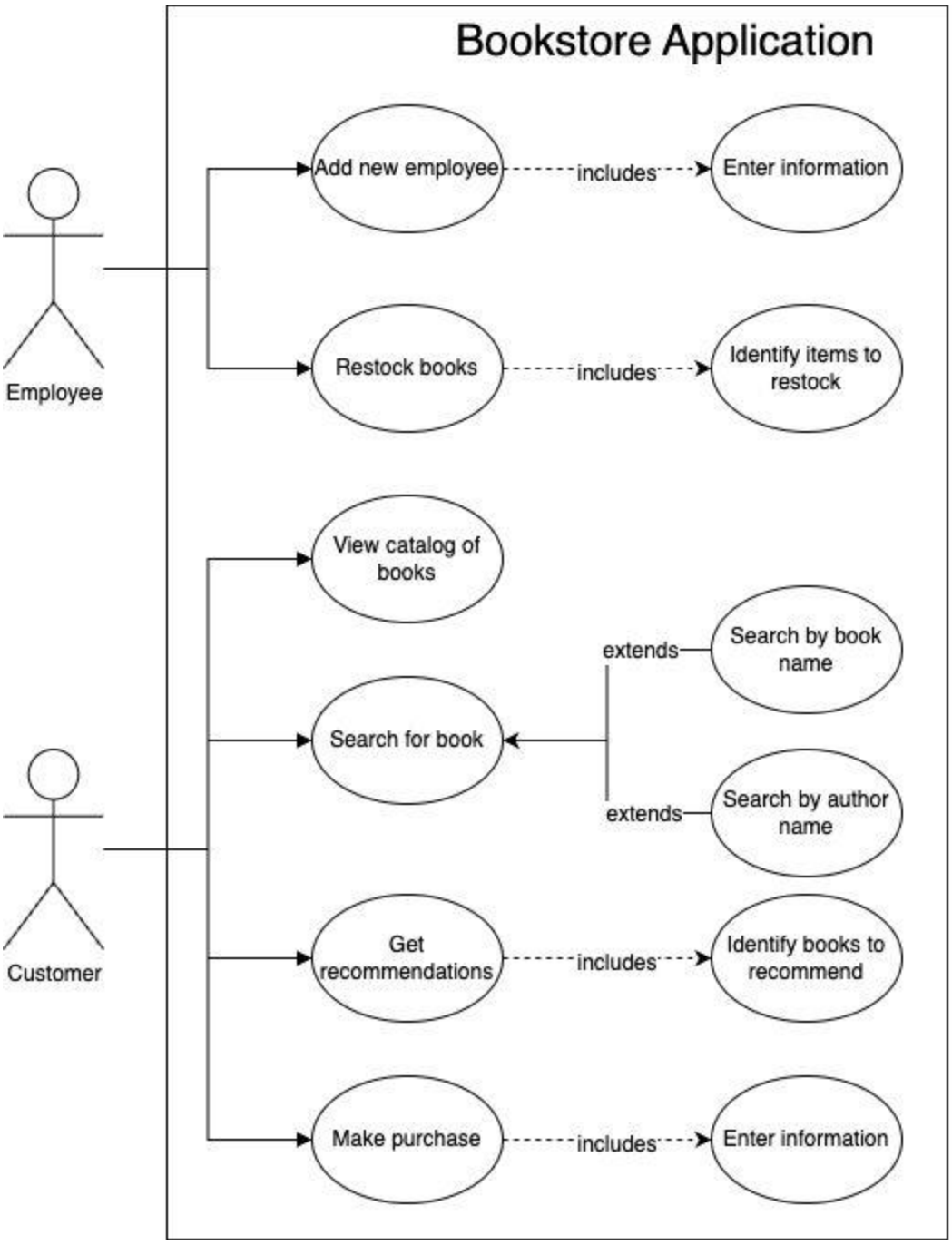
Customer Login: Customers log in to access personalized services, such as book recommendations and online purchases.

Search by Book/Author: Customers can search the bookstore's catalog by book title or author name. This use case involves querying the database and presenting results in a user-friendly manner.

Get Book Recommendations: The system provides personalized book recommendations based on the customer's books of interest, leveraging the machine learning models integrated into the application.

View Catalog of Books: Customers can browse the entire catalog of books. This use case includes filtering and sorting functionalities to enhance the browsing experience.

Checkout and Purchase Books: This critical use case allows customers to select books, add them to a cart, and complete the purchase process, which updates the inventory and records the transaction in the database.



The documented business use cases provide a clear roadmap for the functionalities required by our bookstore application. They serve as guidelines for the development process, ensuring that the application meets the needs of both employees and customers. By outlining these use cases, we ensure that the application's workflows are designed for efficiency, usability, and scalability.

Refinement and Evaluation of Big Data Insights

Building upon the EDA with integrated data lake insights, our bookstore has established a framework to leverage Big Data for strategic advantage. This section refines our approach to extracting and applying these insights and details our methodology for evaluating Big Data ideas with short, medium, and long-term benefits.

Refinement of Data Lake Insights

Enhanced Data Analytics

We have refined our data analytics processes to draw actionable insights more effectively from our data lake. This includes the use of more sophisticated machine learning algorithms and data mining techniques to analyze customer ratings and purchasing patterns.

Integration with Machine Learning Models

The insights from the data lake are now more seamlessly integrated with our machine learning models. This has enhanced our recommendation engine and predictive stock management capabilities, delivering more personalized customer experiences and efficient inventory handling.

Big Data Evaluation Approach

Short-Term Benefits

Quick Wins: Identify and prioritize insights that can be quickly turned into actionable strategies, such as targeted marketing campaigns based on trending genres or authors.

Performance Metrics: Evaluate the impact of these insights on key performance indicators (KPIs) like sales conversion rates, customer engagement, and inventory turnover.

Medium-Term Benefits

Customer Experience Enhancement: Develop features that utilize medium-term insights, such as a "books like this" feature that uses customer behavior to suggest similar titles.

Business Process Optimization: Implement changes to business processes, such as dynamic pricing or inventory restocking, that are informed by predictive analytics.

Long-Term Benefits

Strategic Planning: Use long-term insights to inform broader business strategies, such as market expansion, new genre introduction, or partnership development with publishers.

Innovation: Invest in research and development for new applications of Big Data, like augmented reality book previews or data-driven content creation.

Continuous Improvement Cycle

Feedback Loops: Establish feedback loops where insights from Big Data analytics are regularly reviewed and used to refine the EDA and the bookstore's overall data strategy.

Scalability and Adaptation: Ensure that the architecture is scalable and adaptable, allowing the bookstore to incorporate emerging Big Data technologies and methodologies.

Refining the use of insights within the EDA, coupled with a structured approach to evaluating Big Data initiatives, positions our bookstore to capitalize on data-driven opportunities. By categorizing initiatives into short, medium, and long-term benefits, we can align our Big Data strategies with our overall business objectives, ensuring that we not only respond to current trends but also shape future ones.

Machine Learning Model Selection and Training

In our pursuit of data-driven decision-making, we have selected and trained various machine learning models to extract actionable insights from our bookstore's data. These models are integral to improving user experience and driving organizational excellence across multiple fronts.

Machine Learning Data Selection

Goodreads ratings data was chosen for training our machine learning models due to its rich and diverse user base, often providing more detailed and genre-spanning ratings information compared to Amazon. This platform attracts dedicated book enthusiasts, offering nuanced ratings and reviews that are less influenced by commercial factors and more reflective of content quality. Such depth and consistency in the data make Goodreads an ideal source for training models, especially for tasks like ratings predictions and personalized book recommendations, ensuring our models are attuned to genuine reader preferences and diverse reading habits.

Machine Learning Algorithm Selection

Algorithm Diversity: We have employed a mix of supervised and unsupervised learning algorithms tailored to specific use cases within our bookstore application.

1. Supervised Learning: For predictive tasks, such as rating predictions, we have trained models like Random Forest and Gradient Boosting, chosen for their robustness and accuracy in handling tabular data.
2. Unsupervised Learning: To understand customer segments and book clustering, algorithms such as K-means have been utilized to identify patterns and groupings in the data without labeled outcomes.

Model Training and Insights Extraction

Rating Predictions Model:

- The Random Forest algorithm was trained on historical book ratings data to predict future ratings.
- The model's accuracy suggests it is highly effective in predicting the ratings for unseen data, indicating its potential in assisting with stock management decisions.

Book Recommendation Engine:

- Utilizing techniques like collaborative filtering, we trained models to generate personalized book recommendations for users.
- The recommendation engine is designed to enhance the user experience by suggesting books aligned with individual user preferences.

Model Evaluation and Deployment

Performance Metrics: Models were evaluated using appropriate metrics. For instance, accuracy and F1 score for classification tasks, and mean squared error (MSE) for regression.

Deployment: The trained models have been deployed as part of the application's backend, running on Google Cloud to ensure scalability and reliability.

Continuous Learning: Models are updated periodically with new data by the bookstore, ensuring that the insights they provide remain relevant and accurate over time.

Business Impact and Decisions

Short-Term Impact: In the short term, the models aid in tactical decisions like creating targeted promotions based on predicted popular titles and managing inventory levels based on demand forecasts.

Medium-Term Strategies: Over the medium term, the insights support strategic decisions such as optimizing the book procurement process and refining marketing strategies.

Long-Term Vision: In the long term, the models contribute to overarching goals such as expanding to new markets based on genre popularity and reader demographics.

The careful selection and training of machine learning algorithms underpin our ability to extract meaningful insights from our bookstore's data. These insights are crucial in driving business decisions that not only improve the immediate user experience but also contribute to sustained organizational excellence. As we continue to refine our models and incorporate new data, our capacity for informed decision-making will grow, reinforcing our competitive edge in the marketplace.

Ratings Prediction Model

Goodreads Book Ratings Predictions

Content

1. Exploring Data.
2. Analyze Data through visualizations.
3. Data Preparation e.g.: [Ordinal Encoding, Handling Missing Values].
4. Feature Engineering.
5. Building Multiple Machine Learning Models.
6. Compare models accuracy on training data.
7. Make predictions using each model.
8. Compare models accuracy on test data.
9. Compare training VS test score for each model
10. Extract best model for use in the bookstore to drive decisions.

```
!pip install eli5
```

```
[ ] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import seaborn as sns
    import matplotlib.pyplot as plt
    from joblib import dump
    from google.colab import drive
    from eli5.sklearn import PermutationImportance
    import eli5
```

1. Importing and Exploring Dataset

```
[ ] drive.mount('/content/drive')
    df = pd.read_csv('books.csv', on_bad_lines='skip')
    df.head()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date	publisher
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591	9/16/2006	Scholastic Inc.
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221	9/1/2004	Scholastic Inc.
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244	11/1/2003	Scholastic
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9780439655484	eng	435	2339585	36325	5/1/2004	Scholastic Inc.
4	8	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9780439682589	eng	2690	41428	164	9/13/2004	Scholastic

```
df.describe() # Generate the summary table of the data
```

	bookID	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11123.000000	11123.000000	1.112300e+04	11123.000000	1.112300e+04	11123.000000
mean	21310.856963	3.934075	9.759880e+12	336.405556	1.794285e+04	542.048099
std	13094.727252	0.350485	4.429758e+11	241.152626	1.124992e+05	2576.619589
min	1.000000	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	10277.500000	3.770000	9.780345e+12	192.000000	1.040000e+02	9.000000
50%	20287.000000	3.960000	9.780582e+12	299.000000	7.450000e+02	47.000000
75%	32104.500000	4.140000	9.780872e+12	416.000000	5.000500e+03	238.000000
max	45641.000000	5.000000	9.790008e+12	6576.000000	4.597666e+06	94265.000000

```
[ ]  df.dtypes # Check the data types of all columns
```

```
bookID                          int64
title                          object
authors                        object
average_rating                float64
isbn                           object
isbn13                          int64
language_code                 float64
num_pages                       int64
ratings_count                   int64
text_reviews_count              int64
publication_date      datetime64[ns]
publisher                      object
year                            int64
num_occ                         int64
dtype: object
```

```
[ ]  df.isnull().sum() # Check if there's any missing value
```

```
bookID                0
title                 0
authors               0
average_rating        0
isbn                  0
isbn13                0
language_code         0
  num_pages           0
ratings_count         0
text_reviews_count    0
publication_date      0
publisher             0
dtype: int64
```

2. Data Cleaning & Feature Engineering

```
from sklearn.preprocessing import OrdinalEncoder

encoding = {'language_code':{'en-US': 'eng', 'en-GB': 'eng', 'en-CA': 'eng'}} # Unify the langauge codes
df.replace(encoding, inplace=True)

enc = OrdinalEncoder()
enc.fit(df[['language_code']])
df[['language_code']] = enc.fit_transform(df[['language_code']]) # Apply ordinal encoding on language_code to convert it into numerical column
```

```
df['publication_date'] = pd.to_datetime(df['publication_date'], format='%m/%d/%Y', errors='coerce') # Convert data type of publication_date from object into date type
df[df['publication_date'].isnull()]
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date	publisher
8177	31373	In Pursuit of the Proper Sinner (Inspector Lyn...	Elizabeth George	4.10	0553575104	9780553575101	2.0	718	10608	295	NaT	Bantam Books
11094	45531	Montaillou village occitan de 1294 à 1324	Emmanuel Le Roy Ladurie/Emmanuel Le Roy-Ladurie	3.96	2070323285	9782070323289	4.0	640	15	2	NaT	Folio histoire

Since there are only 2 books with courrpted dates, we googled these 2 books to get the publication dates and put them manually

```
[ ] df.loc[df.bookID == 31373, 'publication_date'] = '1999-10-01 00:00:00'
    df.loc[df.bookID == 45531, 'publication_date'] = '1975-10-01 00:00:00'
```

```
[ ] df['year'] = pd.DatetimeIndex(df['publication_date']).year # Extract year of publication in a separate column

    df.rename(columns = {'  num_pages': 'num_pages'}, inplace=True) # Rename the column to remove leading whitespaces
```

```
[ ] df['num_occ'] = df.groupby('title')['title'].transform('count') # Add a new feature which has the number of occurences of each book
```

Calculating New Features

```
[ ] df['rate_occ'] = df['average_rating'] * df['num_occ']
    df['rate_weight'] = df['average_rating'] * df['text_reviews_count']
    df['rate_weight_2'] = df['average_rating'] * df['ratings_count']
    df['rate_per_pages'] = df['average_rating'] * df['num_pages']
```

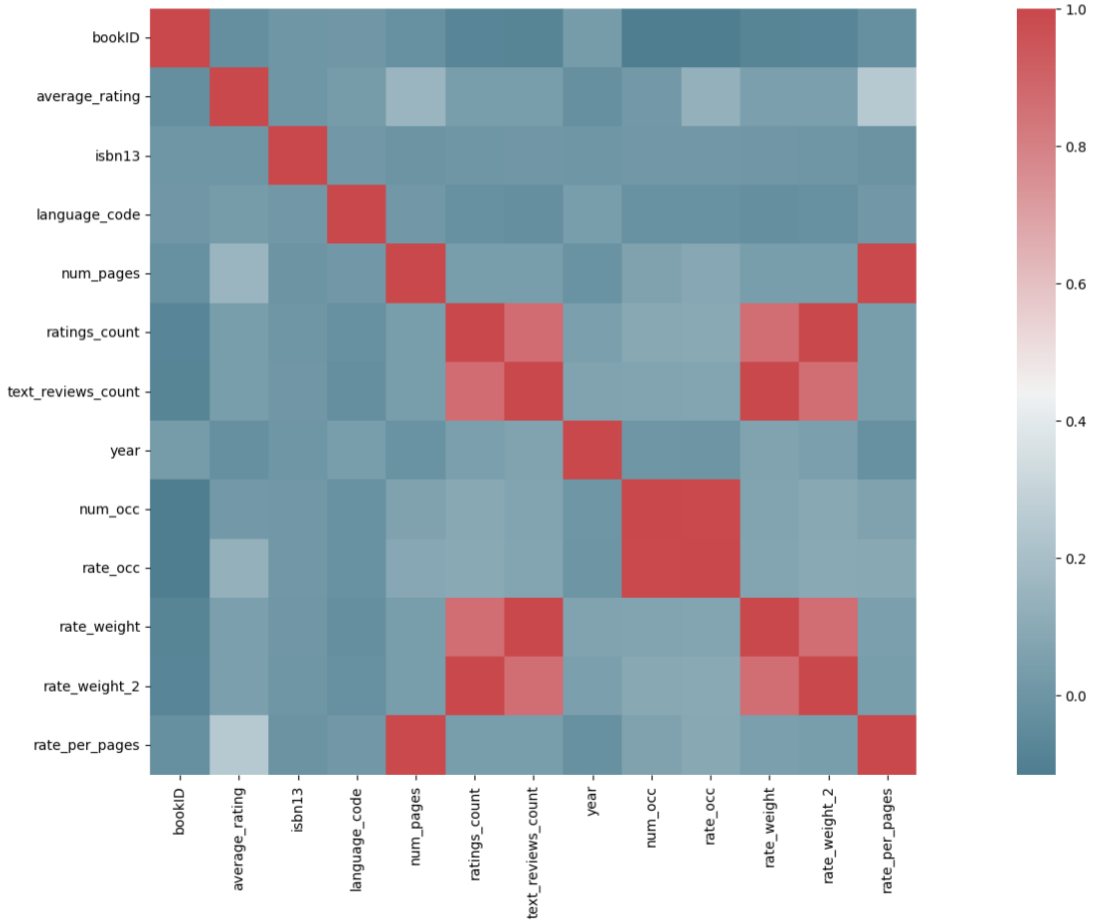
```
[ ] df.columns
```

```
Index(['bookID', 'title', 'authors', 'average_rating', 'isbn', 'isbn13',
       'language_code', 'num_pages', 'ratings_count', 'text_reviews_count',
       'publication_date', 'publisher', 'year', 'num_occ', 'rate_occ',
       'rate_weight', 'rate_weight_2', 'rate_per_pages'],
      dtype='object')
```

3. Data Analysis & Visualizations

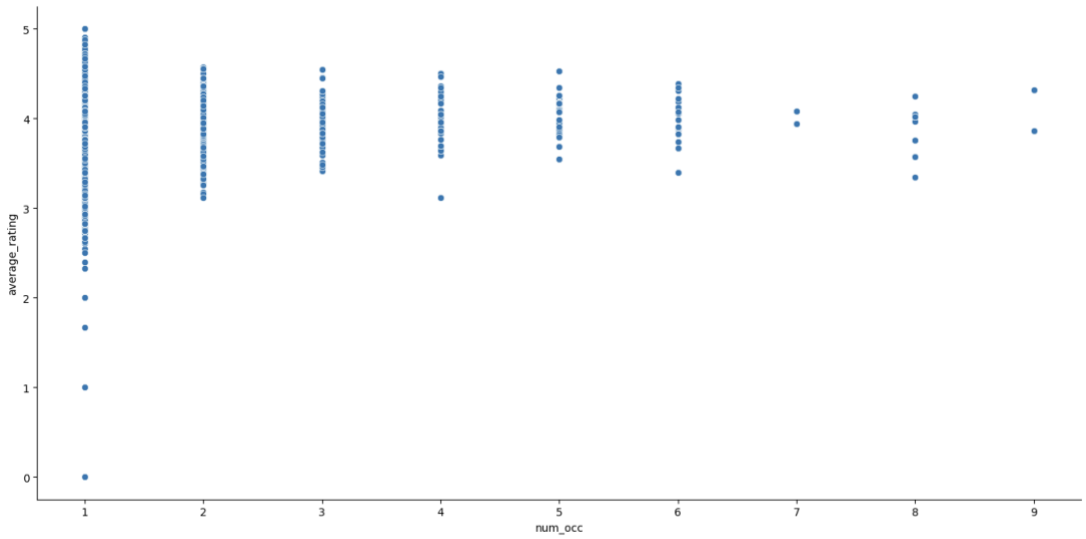
```
fig = plt.gcf()
fig.set_size_inches(26, 10)
corr = df.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True)
```

```
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
<Axes: >
```



```
[ ] sns.relplot(x="num_occ", y="average_rating", data=df, height=7, aspect = 2)
```

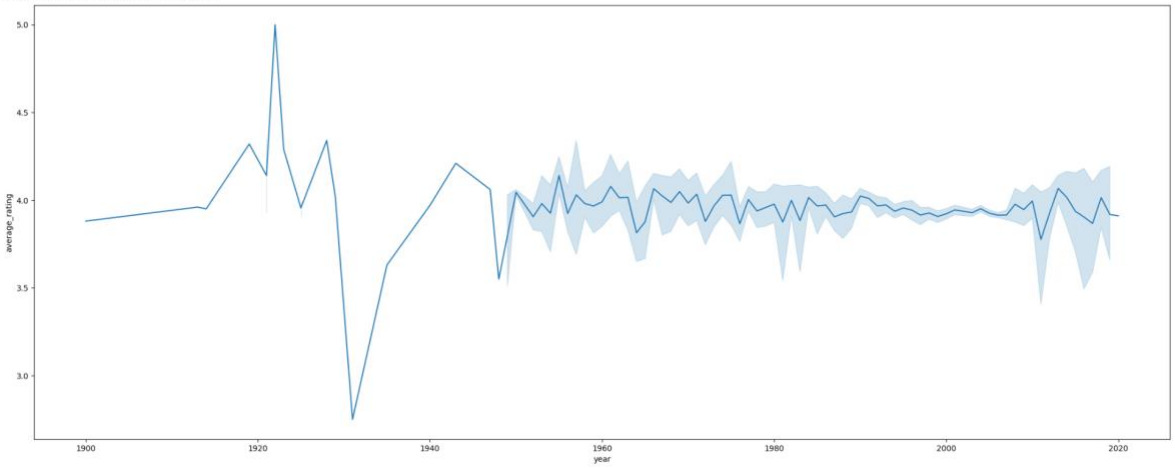
```
<seaborn.axisgrid.FacetGrid at 0x7cae310bcdf0>
```



The upper visual says that any book appeared more than once has a good/high rating

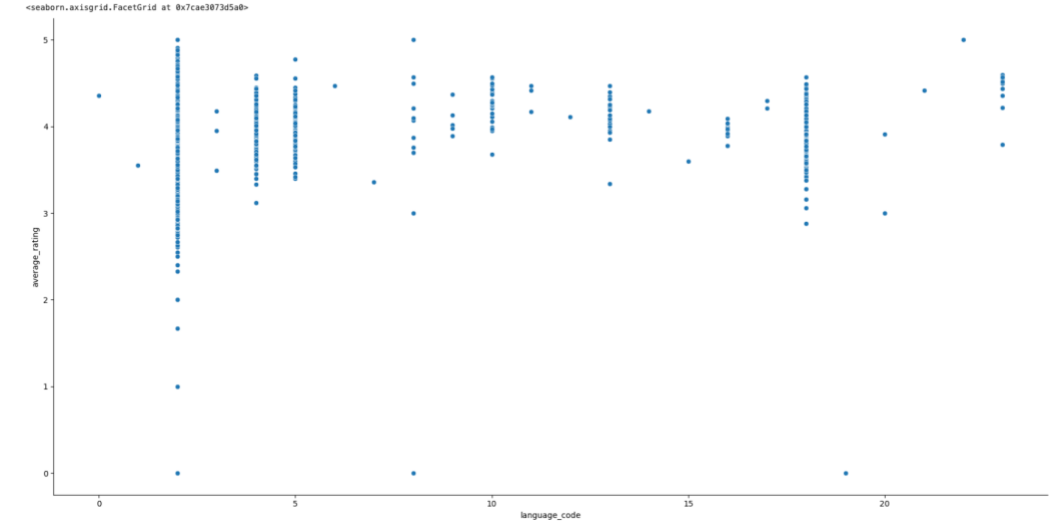
```
fig = plt.gcf()
fig.set_size_inches(26, 10)
sns.lineplot(x="year", y="average_rating", data=df)
```

```
<Axes: xlabel='year', ylabel='average_rating'>
```



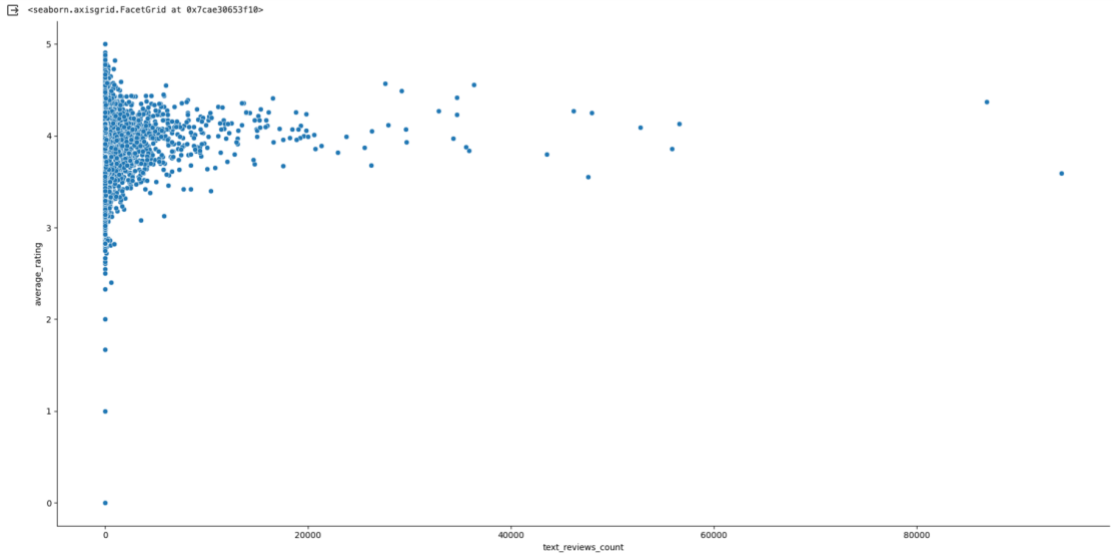
```
[ ] sns.relplot(x="language_code", y="average_rating", data=df, height=9, aspect = 2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7cae3073d5a0>
```



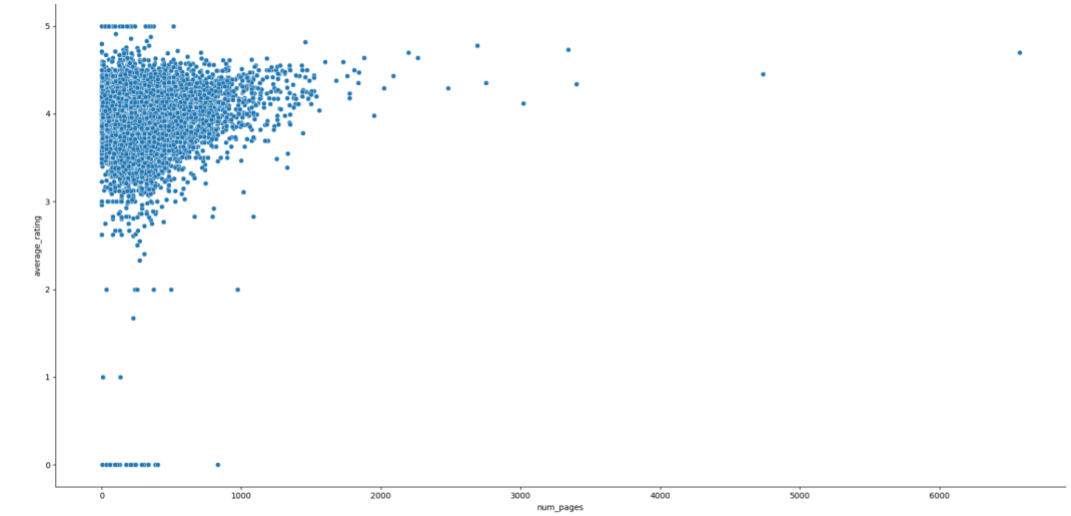

```
sns.relplot(x="text_reviews_count", y="average_rating", data=df, height=9, aspect = 2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7cae30653f10>
```



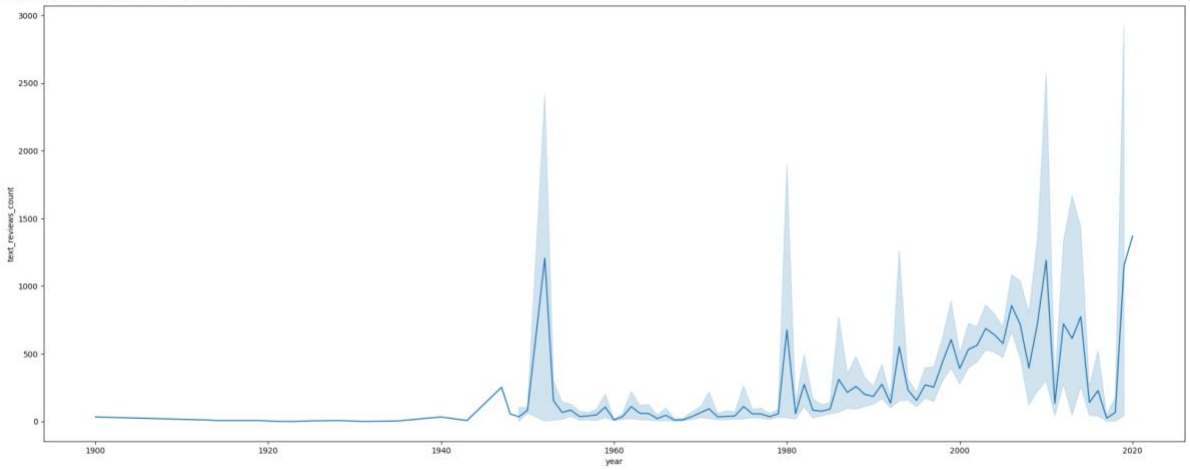
```
sns.relplot(x="num_pages", y="average_rating", data=df, height=9, aspect = 2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7cae3073d660>
```




```
[ ] fig = plt.gcf()
    fig.set_size_inches(26, 10)
    sns.lineplot(x="year", y="text_reviews_count", data=df)
```

```
<Axes: xlabel='year', ylabel='text_reviews_count'>
```



We can see from the visual above that starting from the 80s, the rate/number of reviews is getting higher than before.

This could be attributed to the effect of the internet picking up and people shareing more reviews and thoughts online.

5. Creating Model

We will be training four models and pick one based on performance on the test data:

1. AdaBoost Model
2. Linear Regression Model
3. Ridge Regression Model
4. Random Forest Model

```
[ ] label = df['average_rating'].values
    df.drop(['bookID', 'title', 'authors', 'isbn', 'isbn13', 'publication_date', 'publisher', 'average_rating'], axis=1, inplace=True)
```

```
[ ] # Split the Data into 70% - 30% for train and test
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(df, label, test_size=0.3)
```

```
    from sklearn.model_selection import GridSearchCV
    from sklearn.ensemble import AdaBoostRegressor
    from sklearn.tree import DecisionTreeRegressor

    model = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4))

    parameters = {
        'learning_rate': [0.001, 0.01, 0.02, 0.1, 0.2, 1.0],
        'n_estimators': [10, 50, 100, 200]
    }

    grad_Ada = GridSearchCV(model, parameters, refit=True)
    grad_Ada.fit(X_train, y_train)

    print('Best Score: ', grad_Ada.best_score_*100, '\nBest Parameters: ', grad_Ada.best_params_)
```

```
Best Score:  93.14043713523648
Best Parameters:  {'learning_rate': 0.02, 'n_estimators': 200}
```

```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.model_selection import GridSearchCV

    model  = LinearRegression()

    parameters = {
        'fit_intercept': [True, False],
    }

    grad_Linear = GridSearchCV(model, parameters, refit=True)
    grad_Linear.fit(X_train, y_train)

    print('Best Score: ', grad_Linear.best_score_*100, '\nBest Parameters: ', grad_Linear.best_params_)
```

```
Best Score:  77.27720716829015
Best Parameters:  {'fit_intercept': True}
```

```
[ ] from sklearn.linear_model import Ridge

    model = Ridge()

    parameters = {
        'fit_intercept': [True, False],
        'alpha': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    }

    grad_ridge = GridSearchCV(model, parameters, refit=True)
    grad_ridge.fit(X_train, y_train)

    print('Best Score: ', grad_ridge.best_score_*100, '\nBest Parameters: ', grad_ridge.best_params_)
```

```
Best Score:  77.27719334927319
Best Parameters:  {'alpha': 0.001, 'fit_intercept': True}
```

```
[ ] from sklearn.ensemble import RandomForestRegressor

    model = RandomForestRegressor()

    parameters = {
        'n_estimators': [50, 100, 150, 200],
        'max_depth': [3, 5, 7, 10, 12, 15],
        'min_samples_split': [5, 10, 15],
        'min_samples_leaf': [5, 10, 15]
    }

    grad_rf = GridSearchCV(model, parameters, refit=True, cv=10)
    grad_rf.fit(X_train, y_train)

    print('Best Score: ', grad_rf.best_score_*100, '\nBest Parameters: ', grad_rf.best_params_)
```

```
Best Score:  98.55459891472951
Best Parameters:  {'max_depth': 15, 'min_samples_leaf': 5, 'min_samples_split': 5, 'n_estimators': 200}
```

```
[ ] l = []
    l.append(('AdaBoost', grad_Ada.best_score_*100))
    l.append(('Linear Regression', grad_Linear.best_score_*100))
    l.append(('Ridge Regression', grad_ridge.best_score_*100))
    l.append(('Random Forest', grad_rf.best_score_*100))
    scores = pd.DataFrame(l, columns =['Model', 'Train Score'])
```

6. Make Predictions using the 4 Models

```
[ ] from sklearn.metrics import r2_score, mean_squared_error, accuracy_score
```

AdaBoost Model

```
[ ] # AdaBoost Model
    pred_adaboost = grad_Ada.predict(X_test)

    # Check Model Score
    print("Residual sum of squares: ",  np.mean((pred_adaboost - y_test) ** 2))
    print('RMSE: '+str(np.sqrt(mean_squared_error(y_test, pred_adaboost))))
    print('Model Score on Test Data: ', grad_Ada.score(X_test, y_test))
```

```
Residual sum of squares:  0.006525180197627225
RMSE: 0.08077858749462771
Model Score on Test Data:  0.9588548854748151
```

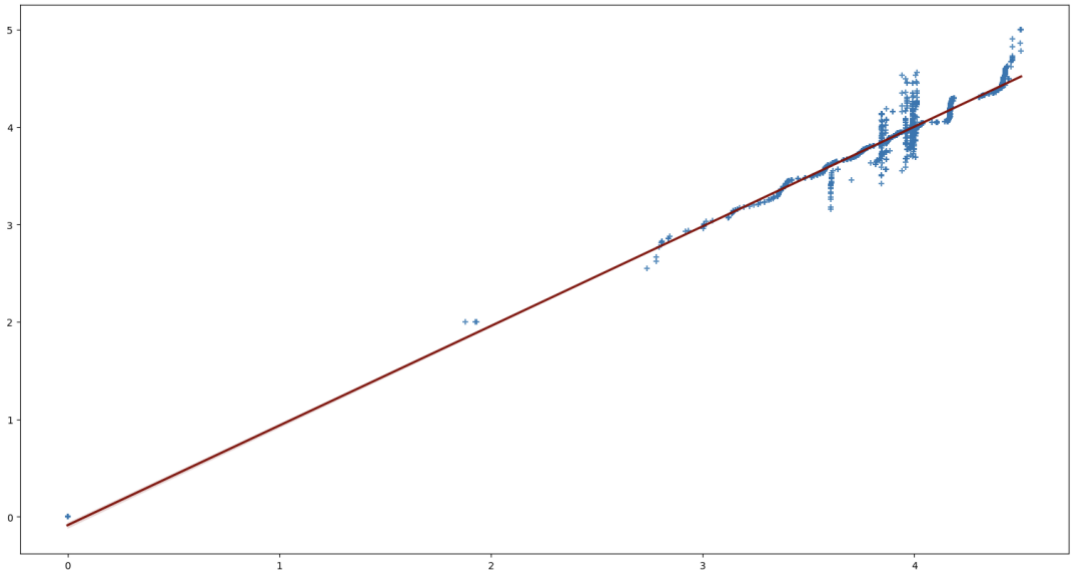
Features Importance

```
[ ] perm = PermutationImportance(grad_Ada.best_estimator_, random_state=1).fit(X_test, y_test)
    eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Weight	Feature
1.6937 ± 0.0472	rate_occ
0.0943 ± 0.0046	num_occ
0.0019 ± 0.0015	rate_per_pages
0.0005 ± 0.0001	ratings_count
0.0000 ± 0.0000	text_reviews_count
0.0000 ± 0.0000	year
0.0000 ± 0.0000	num_pages
0.0000 ± 0.0000	rate_weight_2
0 ± 0.0000	language_code
-0.0000 ± 0.0000	rate_weight

```
plt.figure(figsize=(19,10))
sns.regplot(x=pred_adaboost, y=y_test, marker="+", line_kws={'color':'darkred','alpha':1.0})
```

```
<Axes: >
```



Linear Regression Model

```
[ ] # Linear Regression Model
    pred_lr = grad_Linear.predict(X_test)

    # Check Model Score
    print("Residual sum of squares: ",  np.mean((pred_lr - y_test) ** 2))
    print('RMSE: '+str(np.sqrt(mean_squared_error(y_test, pred_lr))))
    print('Model Score on Test Data: ', grad_Linear.score(X_test, y_test))
```

```
Residual sum of squares:  0.02356871478719784
RMSE: 0.1535210564945338
Model Score on Test Data:  0.8513853350006635
```

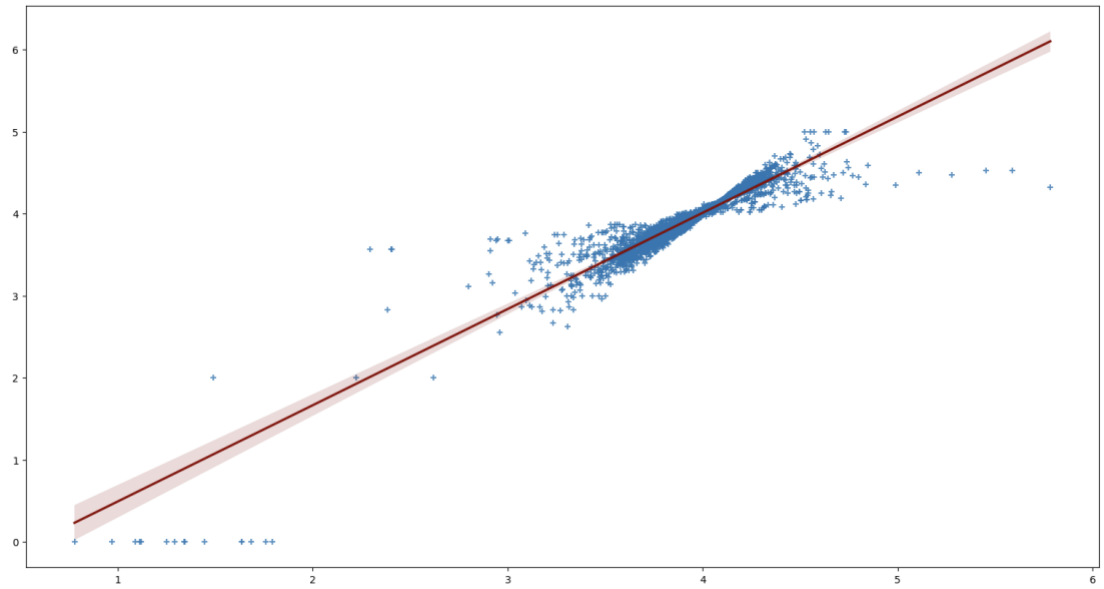
Features Importance

```
perm = PermutationImportance(grad_Linear.best_estimator_, random_state=1).fit(X_test, y_test)
eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Weight	Feature
36.5451 ± 0.7596	rate_occ
35.8743 ± 1.2263	num_occ
6.2787 ± 0.1215	rate_per_pages
5.7062 ± 0.2717	num_pages
0.2712 ± 0.0082	ratings_count
0.2420 ± 0.0057	rate_weight_2
0.2043 ± 0.0102	rate_weight
0.2025 ± 0.0089	text_reviews_count
0.0002 ± 0.0003	language_code
-0.0000 ± 0.0000	year

```
plt.figure(figsize=(19,10))
sns.regplot(x=pred_lr, y=y_test, marker="+", line_kws={'color':'darkred','alpha':1.0})
```

```
<Axes: >
```



Ridge Regression Model

```
# Ridge Regression Model
pred_ridge = grad_ridge.predict(X_test)

# Check Model Score
print("Residual sum of squares: ",  np.mean((pred_ridge - y_test) ** 2))
print('RMSE: '+str(np.sqrt(mean_squared_error(y_test, pred_ridge))))
print('Model Score on Test Data: ', grad_ridge.score(X_test, y_test))
```

```
Residual sum of squares:  0.023568987075049647
RMSE: 0.15352194330143704
Model Score on Test Data:  0.851383618064961
```

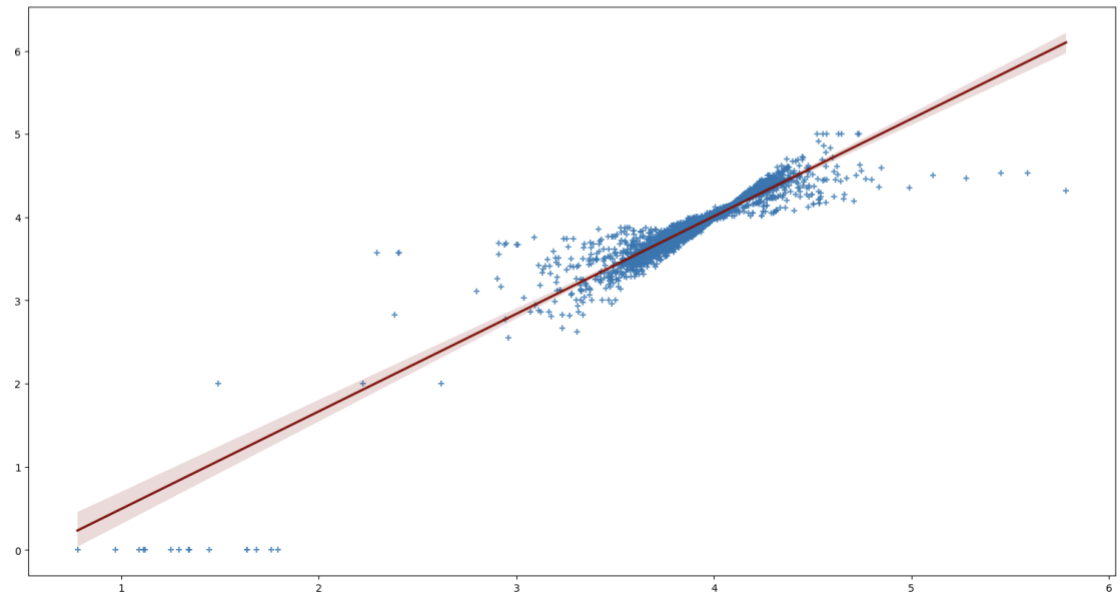
Features Importance

```
[ ] perm = PermutationImportance(grad_ridge.best_estimator_, random_state=1).fit(X_test, y_test)
    eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Weight	Feature
36.5436 ± 0.7596	rate_occ
35.8728 ± 1.2262	num_occ
6.2791 ± 0.1216	rate_per_pages
5.7066 ± 0.2717	num_pages
0.2712 ± 0.0082	ratings_count
0.2420 ± 0.0057	rate_weight_2
0.2043 ± 0.0102	rate_weight
0.2025 ± 0.0089	text_reviews_count
0.0002 ± 0.0003	language_code
-0.0000 ± 0.0000	year

```
plt.figure(figsize=(19,10))
sns.regplot(x=pred_ridge,y=y_test, marker="+", line_kws={'color':'darkred','alpha':1.0})
```

```
<Axes: >
```



Random Forest Model

```
[ ] # Random Forest Model
    pred_rf = grad_rf.predict(X_test)

    # Check Model Score
    print("Residual sum of squares: ",  np.mean((pred_rf - y_test) ** 2))
    print('RMSE: '+str(np.sqrt(mean_squared_error(y_test, pred_rf))))
    print('Model Score on Test Data: ', grad_rf.score(X_test, y_test))
```

```
Residual sum of squares:  0.0017604099531430685
RMSE: 0.04195723957963713
Model Score on Test Data:  0.9888995756531466
```

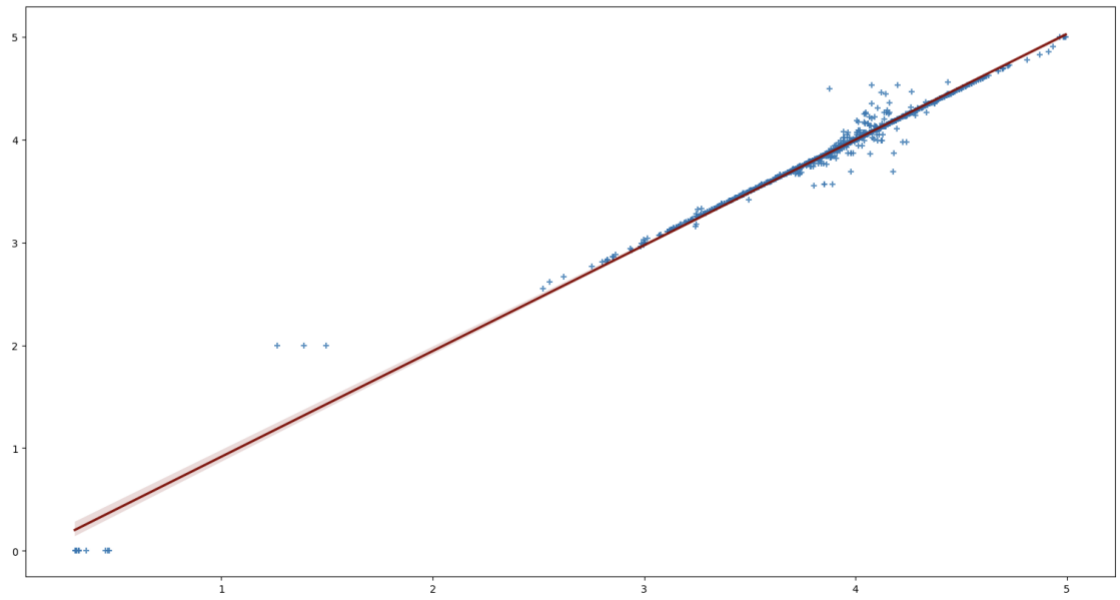
Features Importance

```
perm = PermutationImportance(grad_ridge.best_estimator_, random_state=1).fit(X_test, y_test)
eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Weight	Feature
36.5436 ± 0.7596	rate_occ
35.8728 ± 1.2262	num_occ
6.2791 ± 0.1216	rate_per_pages
5.7066 ± 0.2717	num_pages
0.2712 ± 0.0082	ratings_count
0.2420 ± 0.0057	rate_weight_2
0.2043 ± 0.0102	rate_weight
0.2025 ± 0.0089	text_reviews_count
0.0002 ± 0.0003	language_code
-0.0000 ± 0.0000	year

```
plt.figure(figsize=(19,10))
sns.regplot(x=pred_rf,y=y_test, marker="+", line_kws={'color':'darkred','alpha':1.0})
```

```
<Axes: >
```



```
[ ] l2 = []
    l2.append(('AdaBoost', grad_Ada.score(X_test, y_test)*100))
    l2.append(('Linear Regression', grad_Linear.score(X_test, y_test)*100))
    l2.append(('Ridge Regression', grad_ridge.score(X_test, y_test)*100))
    l2.append(('Random Forest', grad_rf.score(X_test, y_test)*100))

    test_scores = pd.DataFrame(l2, columns =['Model', 'Test Score'])
```

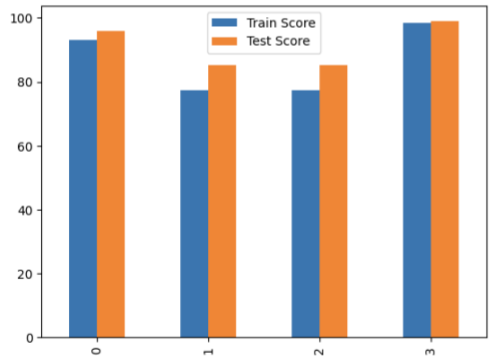
Comparison of results from the four models

```
[ ] scores['Test Score'] = test_scores['Test Score']
    scores
```

	Model	Train Score	Test Score
0	AdaBoost	93.140437	95.885489
1	Linear Regression	77.277207	85.138534
2	Ridge Regression	77.277193	85.138362
3	Random Forest	98.554599	98.889958

```
[ ] scores.plot.bar()
```

```
<Axes: >
```



The Random Forest model has the highest Test Score at approximately 98.8, which indicates it is the best performing model among those tested. This suggests that the Random Forest model is the most accurate in predicting the ratings on unseen data and would be the recommended model to use in the bookstore to derive insights and drive decisions.

```
# Exporting the model
dump(grad_rf, '/content/drive/My Drive/ratings_prediction_model.joblib')
```

```
['/content/drive/My Drive/ratings_prediction_model.joblib']
```

Book Recommendation Engine

Goodreads Ratings Based Book Recommendation Engine

1. Loading required libraries, with installation:
2. Basic Data Exploration
3. Exploratory Data Analysis
    1. Which are the books with most occurances in the list?
    2. Which are the top 10 most rated books?
    3. Which are the authors with most books?
    4. Which are the top 10 highly rated authors?
    5. What is the rating distribution for the books?
    6. Is there relationship between ratings and review counts?
    7. Is there a relationship between number of pages and ratings?
    8. Is there a relationship between ratings and ratings count?
    9. Which are the books with the highest reviews?
4. Topic Modelling
    1. KMeans Clustering
    2. KMeans Clustering with optimisaztion
5. Book Recommendation Engine

Loading libraries with installation

```
[ ]  !pip install urllib3
     !pip install isbnlib
     !pip install newspaper3k

     import numpy as np
     import pandas as pd
     import os
     import seaborn as sns
     import isbnlib
     from newspaper import Article
     import matplotlib.pyplot as plt
     plt.style.use('ggplot')
     from tqdm import tqdm
     from progressbar import ProgressBar
     import re
     from scipy.cluster.vq import kmeans, vq
     from pylab import plot, show
     from matplotlib.lines import Line2D
     import matplotlib.colors as mcolors
     from sklearn.cluster import KMeans
     from sklearn import neighbors
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
```

Topic Modelling

KMeans Clustering

Since our goal is to find groups in data, we will use KMeans clustering which is a type of unsupervised learning which groups unlabelled data.

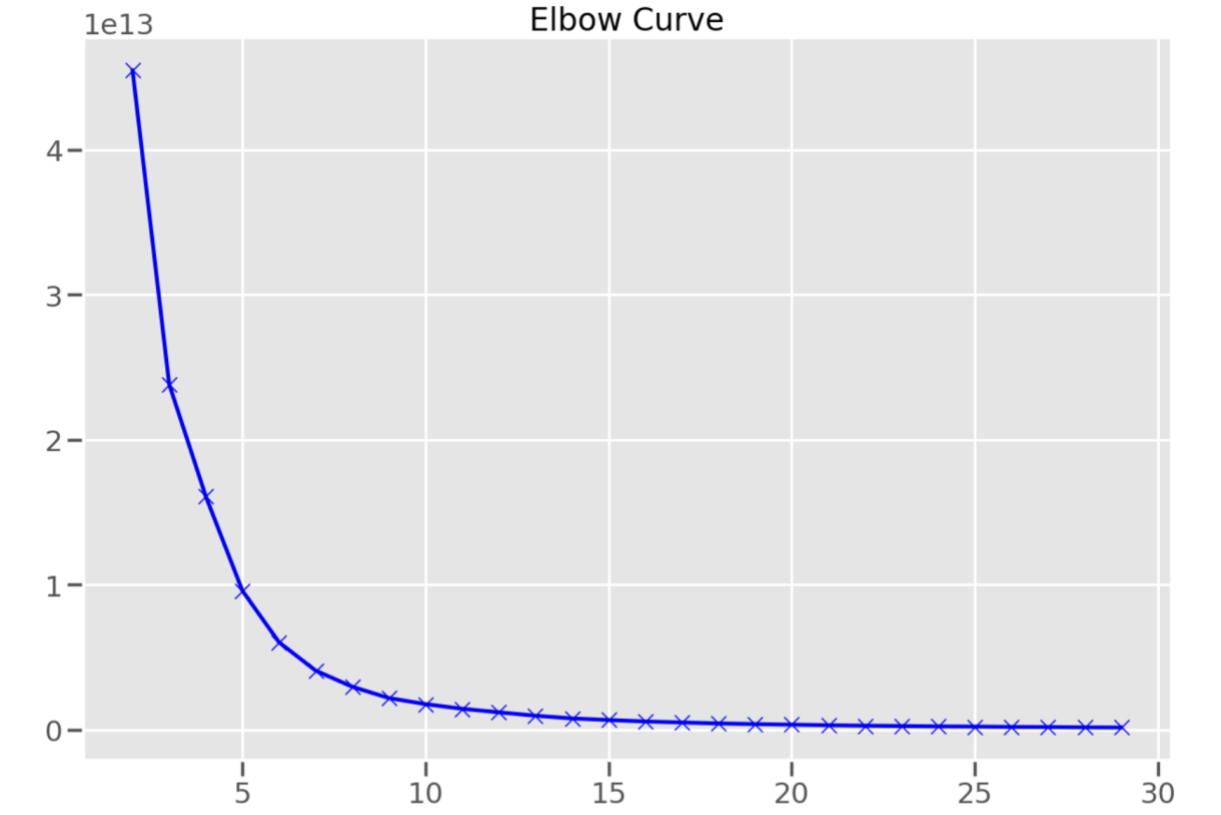
With this, we attempt to find a relationship or groups between the rating count and average rating value.

```
trial = df[['average_rating', 'ratings_count']]
data = np.asarray([np.asarray(trial['average_rating']), np.asarray(trial['ratings_count'])]).T
```

We will use the Elbow Curve method for the best way of finding the number of clusters for the data

```
[ ] X = data
    distortions = []
    for k in range(2,30):
        k_means = KMeans(n_clusters = k)
        k_means.fit(X)
        distortions.append(k_means.inertia_)

    fig = plt.figure(figsize=(15,10))
    plt.plot(range(2,30), distortions, 'bx-')
    plt.title("Elbow Curve")
```

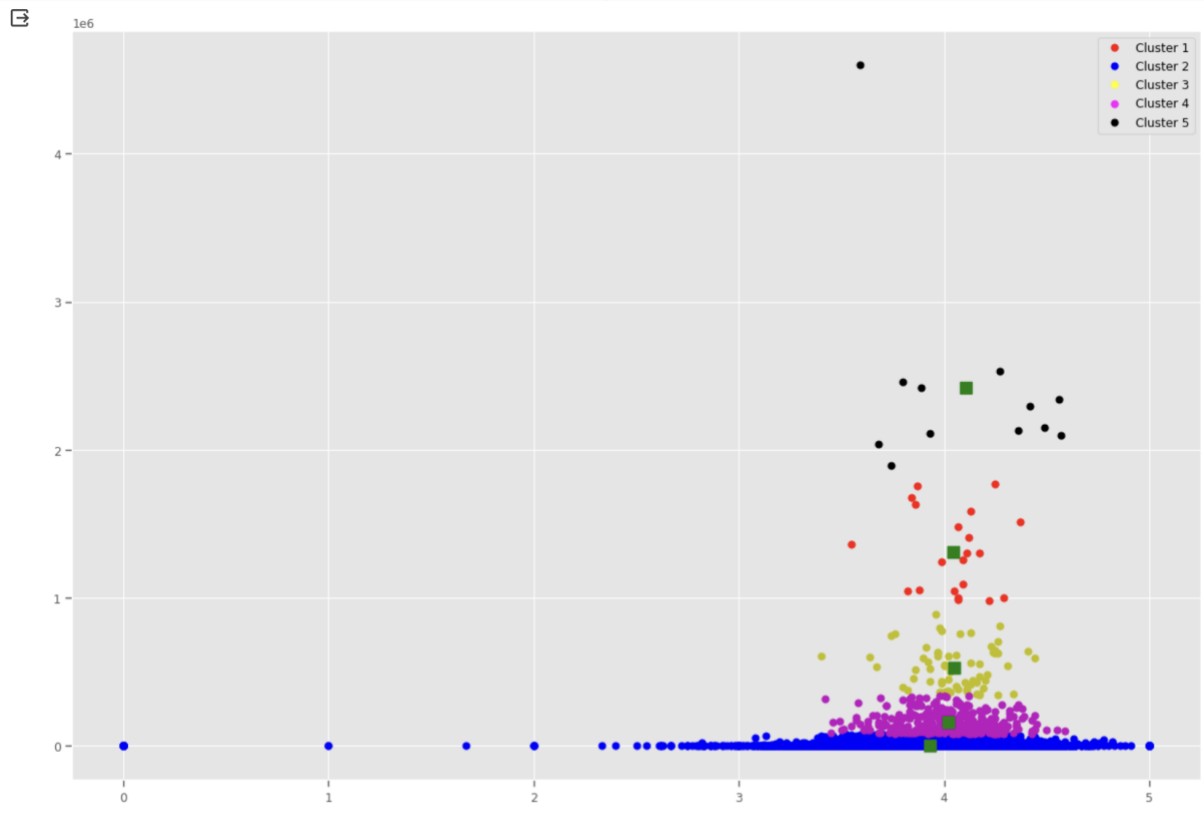
```
Text(0.5, 1.0, 'Elbow Curve')
```



From the above plot, we can see that the elbow lies around the value K=5, so that's what we will attempt it with

```
[ ]  #Computing K means with K = 5, thus, taking it as 5 clusters
     centroids, _ = kmeans(data, 5)

     #assigning each sample to a cluster
     #Vector Quantisation:

     idx, _ = vq(data, centroids)
```

```
     # some plotting using numpy's logical indexing
     sns.set_context('paper')
     plt.figure(figsize=(15,10))
     plt.plot(data[idx==0,0],data[idx==0,1],'or',#red circles
          data[idx==1,0],data[idx==1,1],'ob',#blue circles
          data[idx==2,0],data[idx==2,1],'oy', #yellow circles
          data[idx==3,0],data[idx==3,1],'om', #magenta circles
          data[idx==4,0],data[idx==4,1],'ok',#black circles
     )
     plt.plot(centroids[:,0],centroids[:,1],'sg',markersize=8, )

     circle1 = Line2D(range(1), range(1), color = 'red', linewidth = 0, marker= 'o', markerfacecolor='red')
     circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker= 'o', markerfacecolor='blue')
     circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0,  marker= 'o', markerfacecolor='yellow')
     circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker= 'o', markerfacecolor='magenta')
     circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker= 'o', markerfacecolor='black')

     plt.legend((circle1, circle2, circle3, circle4, circle5)
               , ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5'), numpoints = 1, loc = 0, )


     plt.show()
```



We can see from the above plot, that because of two outliers, the whole clustering algortihm is skewed. Let's remove them and form inferences

KMeans with optimisation

Finding the outliers and then removing them.

```
[ ] trial.idxmax()
```

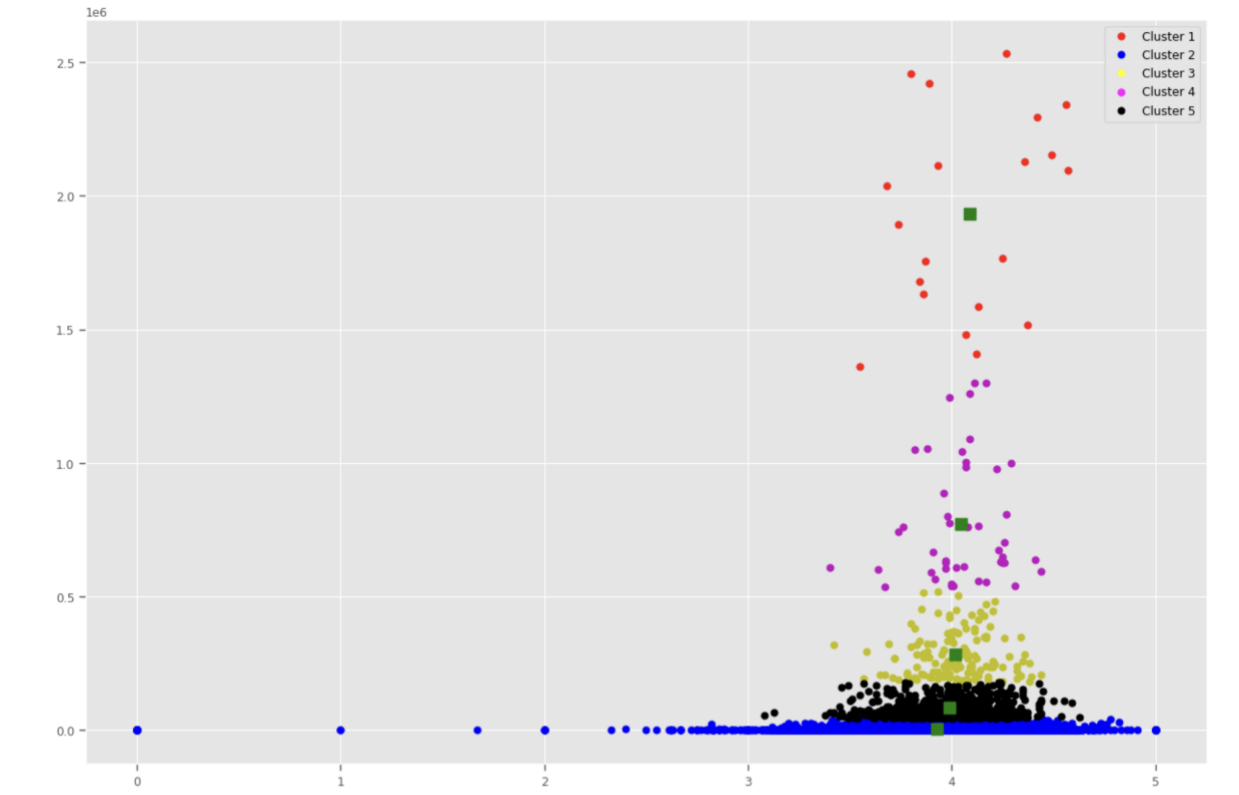
```
average_rating     2034
ratings_count     41865
dtype: int64
```

```
[ ] trial.drop(41865, inplace = True)
```

```
[ ] data = np.asarray([np.asarray(trial['average_rating']), np.asarray(trial['ratings_count'])]).T
```

```
[ ] #Computing K means with K = 8, thus, taking it as 8 clusters
    centroids, _ = kmeans(data, 5)

    #assigning each sample to a cluster
    #Vector Quantisation:

    idx, _ = vq(data, centroids)
```

```
    # some plotting using numpy's logical indexing
    sns.set_context('paper')
    plt.figure(figsize=(15,10))
    plt.plot(data[idx==0,0],data[idx==0,1],'or',#red circles
         data[idx==1,0],data[idx==1,1],'ob',#blue circles
         data[idx==2,0],data[idx==2,1],'oy', #yellow circles
         data[idx==3,0],data[idx==3,1],'om', #magenta circles
         data[idx==4,0],data[idx==4,1],'ok',#black circles
    )
    plt.plot(centroids[:,0],centroids[:,1],'sg',markersize=8, )

    circle1 = Line2D(range(1), range(1), color = 'red', linewidth = 0, marker= 'o', markerfacecolor='red')
    circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker= 'o', markerfacecolor='blue')
    circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0,  marker= 'o', markerfacecolor='yellow')
    circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker= 'o', markerfacecolor='magenta')
    circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker= 'o', markerfacecolor='black')

    plt.legend((circle1, circle2, circle3, circle4, circle5)
               , ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5'), numpoints = 1, loc = 0, )

    plt.show()
```



From the above plot, now we can see that once the whole system *can* be classified into clusters. As the count increases, the rating would end up near the cluster given above. The green squares are the centroids for the given clusters.

As the rating count seems to decrease, the average rating seems to become sparser, with higher volatility and less accuracy.

Recommendation Engine

Having seen the clustering, we can infer that there can be some recommendations which can happen with the relation between Average Rating and Ratings Count.

Taking the Ratings_Distribution (A self created classifying trend), the recommendation system works with the algortihm of K Nearest Neighbors.

Based on a book entered by the user, the nearest neighbours to it would be classified as the books which the user might like.

KNN is used for both classification and regression problems. In classification problems to predict the label of a instance we first find k closest instances to the given one based on the distance metric and based on the majority voting scheme or weighted majority voting(neighbors which are closer are weighted higher) we predict the labels.

In a setting such as this, the unsupervised learning takes place, with the similar neighbors being recommended. For the given list, if I ask recommendations for "The Catcher in the Rye", five books related to it would appear.

Creating a books features table, based on the Ratings Distribution, which classifies the books into ratings scale such as:

- Between 0 and 1
- Between 1 and 2
- Between 2 and 3
- Between 3 and 4
- Between 4 and 5

Broadly, the recommendations then consider the average ratings and ratings cout for the query entered.

```
[ ] books_features = pd.concat([df['Ratings_Dist'].str.get_dummies(sep=","), df['average_rating'], df['ratings_count']], axis=1)
```

```
[ ] books_features.head()
```

bookID	Between 0 and 1	Between 1 and 2	Between 2 and 3	Between 3 and 4	Between 4 and 5	average_rating	ratings_count
1	0	0	0	0	1	4.57	2095690
2	0	0	0	0	1	4.49	2153167
4	0	0	0	0	1	4.42	6333
5	0	0	0	0	1	4.56	2339585
8	0	0	0	0	1	4.78	41428

The min-max scaler is used to reduce the bias which would have been present due to some books having a massive amount of features, yet the rest having less. Min-Max scaler would find the median for them all and equalize it.

```
[ ] min_max_scaler = MinMaxScaler()
    books_features = min_max_scaler.fit_transform(books_features)
```

```
    np.round(books_features, 2)
```

```
array([[0.  , 0.  , 0.  , ..., 1.  , 0.91, 0.46],
       [0.  , 0.  , 0.  , ..., 1.  , 0.9 , 0.47],
       [0.  , 0.  , 0.  , ..., 1.  , 0.88, 0.  ],
       ...,
       [0.  , 0.  , 0.  , ..., 0.  , 0.79, 0.  ],
       [0.  , 0.  , 0.  , ..., 0.  , 0.74, 0.  ],
       [0.  , 0.  , 0.  , ..., 0.  , 0.78, 0.  ]])
```

```
[ ] model = neighbors.NearestNeighbors(n_neighbors=6, algorithm='ball_tree')
    model.fit(books_features)
    distance, indices = model.kneighbors(books_features)
```

Creating specific functions to help in finding the book names:

- Get index from Title
- Get ID from partial name
- Print the similar books from the feature dataset. *(This uses the Indices metric from the nearest neighbors to pick the books.)*

```
[ ] def get_index_from_name(name):
        try:
            return df[df["title"] == name].index.tolist()[0]
        except IndexError:
            return -1  # Or raise an exception

    all_books_names = list(df.title.values)

    def get_id_from_partial_name(partial):
        for name in all_books_names:
            if partial in name:
                print(name,all_books_names.index(name))

    def print_similar_books(query=None,id=None):
      if query:
            found_id = get_index_from_name(query)
            if found_id == -1:
                print("Book not found.")
                return
            id = found_id

      if id is not None and id < len(indices):
            for idx in indices[id][1:]:
                if idx < len(df):
                    print(df.iloc[idx]["title"])
                else:
                    print("Index out of bounds for DataFrame.")
      else:
            print("Invalid book ID.")
```

To test the System, let's try with following examples-

- System by name: The Catcher in the Rye
- System by Name: Agile Web Development with Rails: A Pragmatic Guide
- System by partial name: Harry Potter and the

The Catcher in the Rye:

```
[ ]  print_similar_books("The Catcher in the Rye")
```

```
Hitchhiker's Guide To The Galaxy: The Filming of the Douglas Adams classic
The Peloponnesian War
Henry and June: From the Unexpurgated Diary of Anaïs Nin
Hemingway & Bailey's Bartending Guide to Great American Writers
Liberty Before Liberalism
```

Harry Potter and the:

Since most users won't remember the name for the entire book (especially how it has been entered in the books database), the function to get ID from the partial names helps to choose to ID of the book the user is looking for.

```
[ ]  get_id_from_partial_name("Harry Potter and the ")
```

```
Harry Potter and the Half-Blood Prince (Harry Potter  #6) 0
Harry Potter and the Order of the Phoenix (Harry Potter  #5) 1
Harry Potter and the Chamber of Secrets (Harry Potter  #2) 2
Harry Potter and the Prisoner of Azkaban (Harry Potter  #3) 3
Harry Potter and the Half-Blood Prince (Harry Potter  #6) 0
Harry Potter and the Prisoner of Azkaban (Harry Potter  #3) 3
Harry Potter and the Chamber of Secrets (Harry Potter  #2) 2
Harry Potter and the Sorcerer's Stone (Harry Potter  #1) 8873
Harry Potter and the Philosopher's Stone (Harry Potter  #1) 10674
Harry Potter and the Goblet of Fire (Harry Potter  #4) 10675
```

```
[ ]  print_similar_books(id = 1) #ID for the Book 5
```

```
Harry Potter and the Half-Blood Prince (Harry Potter  #6)
The Fellowship of the Ring (The Lord of the Rings  #1)
Harry Potter and the Chamber of Secrets (Harry Potter  #2)
Harry Potter and the Prisoner of Azkaban (Harry Potter  #3)
The Hobbit  or There and Back Again
```

```
# Exportin the recommendation model
from joblib import dump

dump(model, '/content/drive/My Drive/model.joblib')
dump(indices, '/content/drive/My Drive/indices.joblib')
dump(books_features, '/content/drive/My Drive/books_features.joblib')
```

```
['/content/drive/My Drive/books_features.joblib']
```

Leveraging Google Cloud for Big Data Analytics

Incorporating Big Data Analytics capabilities from Google Cloud, our bookstore application has harnessed powerful tools to extract, filter, analyze, and present data. These capabilities have enabled us to gain deep insights into our datasets, both structured and unstructured, and to visualize the results in a manner that supports informed business decision-making.

Data Analysis and Visualization

Preliminary Data Analysis:

- Initial data analyses were conducted to understand the composition of our datasets, including bestseller books and unstructured book ratings.
- Key metrics such as author prominence were extracted to provide a foundational understanding of our inventory and customer preferences.

Data Visualization:

- We employed various visualization techniques to make the data more accessible and understandable. This included the use of bar charts, line graphs, and heatmaps to reveal patterns and trends within the data.
- These visualizations helped in identifying key areas of focus, such as highly rated authors, which could inform our inventory decisions and marketing strategies.

Leveraging Google Cloud Platform Services

BigQuery for Data Warehousing: Google BigQuery was utilized to store and analyze large volumes of data efficiently. Its powerful data warehousing capabilities allowed us to run complex queries quickly and derive insights from both structured and unstructured datasets.

Machine Learning Services: Google Cloud's machine learning services such as Collaboratory were leveraged to enhance our analytics capabilities. These services provided tools creating the machine learning models.

Insights and Business Impact

Data-Driven Decisions: The insights gained from these analytics have been pivotal in shaping business strategies. For example, stocking decisions are now more data-informed, focusing on books that have high customer ratings.

Enhanced User Experience: Analytics have also played a role in enhancing the customer experience. Personalized book recommendations and dynamic content on the application are direct outcomes of the insights derived from our data analysis.

By leveraging the Big Data Analytics and streaming capabilities of Google Cloud, our bookstore has established a robust framework for data analysis and visualization. This framework not only supports current operational and strategic needs but is also scalable for future growth, ensuring that we continue to derive valuable insights as our datasets expand and evolve.

Continuous Development of Big Data Platform and Data Lake

Our bookstore's commitment to a data-driven future is underscored by the continual development of our Big Data platform and the expansion of our data lake. Designed with scalability and adaptability at its core, the platform is structured to accommodate a growing repository of datasets—from transactional data to customer interactions and book reviews. We are enhancing our data analytics and visualization capabilities to ensure that insights extracted are increasingly sophisticated, actionable, and seamlessly integrated into our Enterprise Data Architecture (EDA). This iterative enhancement allows for real-time business intelligence, supporting informed decision-making and strategic growth, while also enriching the user experience through a deeper understanding of customer needs and market trends.

Reference Architecture

Foundational Principles	- Integration of real-time data processing and machine learning for advanced analytics. - Fostering a data-driven culture for informed business decision-making.
Organising Framework	- Machine Learning Integration: Establishing a machine learning layer to handle complex analytics and model training. - Database Enhancement: Optimizing MySQL for the evolving data needs, including support for machine
Business Solutions	- Advanced Analytics: Utilizing machine learning for predictive insights and personalized recommendations. - Application Workflow Design: Beginning to integrate data-driven insights into operational processes, setting the stage for Part 4.
Data Governance	- Enhanced Data Quality Management: Implementing robust mechanisms to ensure the accuracy and reliability of insights from machine learning models. - Ethical and Responsible Use of Data: Ensuring the ethical application of machine learning, maintaining