CIT550 Project Milestone 2 – Project Outline
Proposal Team Name: Team A+

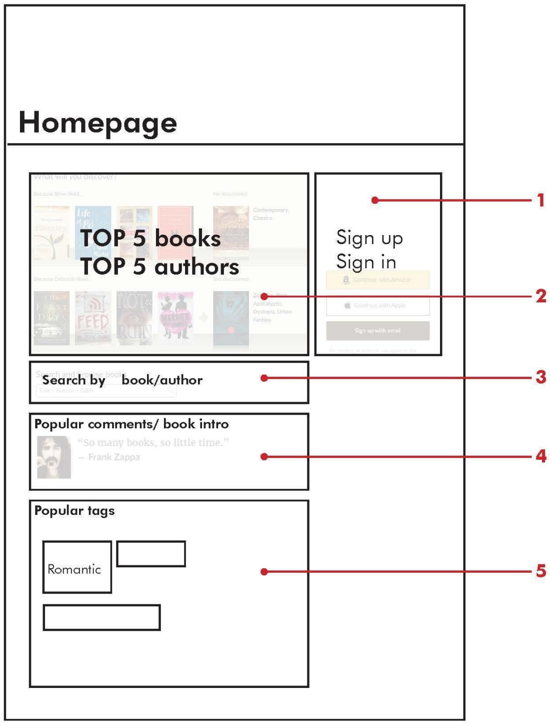
---

1. Motivation for the idea/description of the problem the application solves

Upon emergence in the 1930s, comics have been a major part of pop culture. For our generation, no childhood memory is complete without the Captain, Batman and all the superheroes. The goal of our project is to build a book catalog to document, analyze and give people their customized recommendation of comic memory.

Given how nowadays, we tend to live in a bubble where all the information we receive is what we tend to like and believe, we specifically added the function "surprise me" to encourage our readers to break the bubble and embrace the uncertainty. Inspired by a bookstore that covers books entirely to surprise their readers, we plan to tweak the query of recommendation to make the surprise novel yet agreeable.

2. List of features you will definitely implement in the application and features you might implement in the application, given enough time



1) Sign-up and log in **MUST**

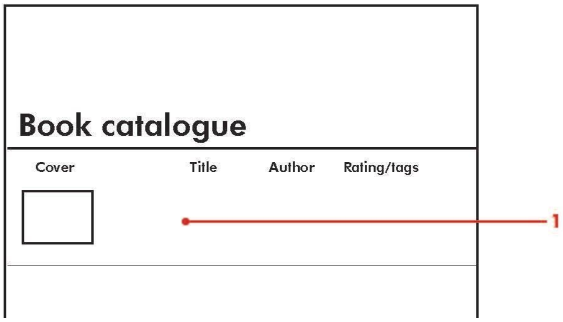
   Separate sign-up and log-in **MAYBE**

2) Showcase top 5 books and authors **MUST**
3) Search by book or author (toggle first, then search) **MUST**
4) Popular comments **MAYBE**
5) Popular tags **MAYBE**

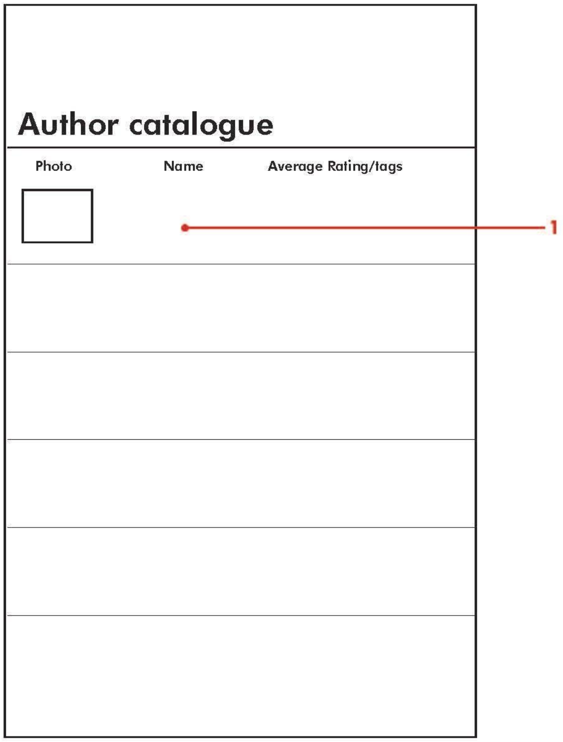
Query used:

a) Find popular book written by author: (author page default)
Given an author name, list all his/her books, order by numbers of read

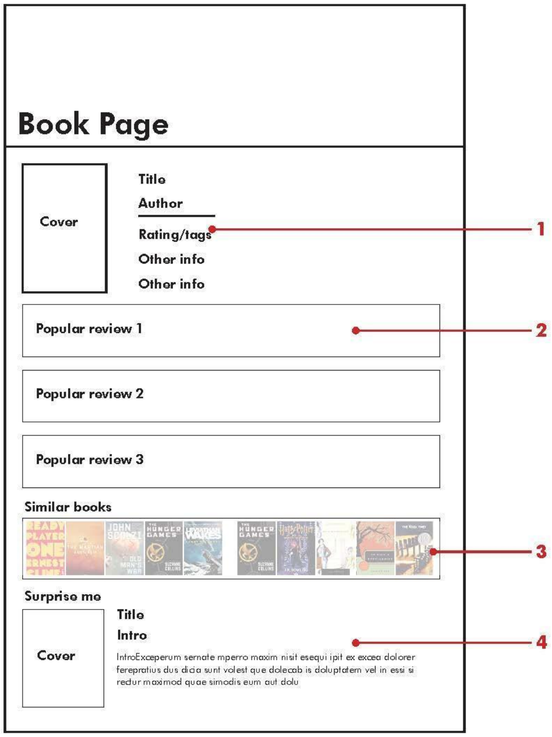
```
SELECT books.title, COUNT(interactions.user_id) AS peopleRead
FROM authors
JOIN books ON authors.authorid = books.authors
LEFT JOIN interactions ON interactions.book_id = books.book_id
WHERE authors.name LIKE '%___%' AND interactions.is_read = 'True'
GROUP BY books.title
ORDER BY COUNT(interactions.user_id) desc
```



1) Cover, title, author, rating/tags **MUST**



1) Photo, name, average rating/tags **MUST**


Book Page
Cover
Title
Author
Rating/tags
Other info
Other info
1
Popular review 1
2
Popular review 2
Popular review 3
Similar books
3
Surprise me
Title
Intro
4
Cover
IntroExceperum sernate mperro maxim nisit esequi ipit ex excea dolorer ferepratius dus dicia sunt volest que dolecab is doluptatem vel in essi si redur maximod quae simodis eum aut dolu

1) Book's basic info **MUST**

2) Top review **MUST**

3) Similar books **MUST**

Here we put covers of 10 books that are similar, and user can use arrow to view more (arrow **MAYBE**)

4) Surprise me **MAYBE**

Here we put a random book that we think could be a pleasant surprise with its cover, title, and introduction

Queries used:

a) Similar books
Given a book_title, list all books that are similar
(book page / home page will implement)

- Approach 1: select book in similar_book list

```
SELECT books.title, books.author, books.average_rating
FROM books
WHERE books.book_id IN (
        SELECT books.simlar_books
        FROM books
        WHERE books.title LIKE '%___%'
)
ORDER BY books.average_rating DESC
```

- Approach 2: Use collaborative filtering

Find book similarity score base on user interaction
Then order by similarity score and take top 10

b) Surprise me

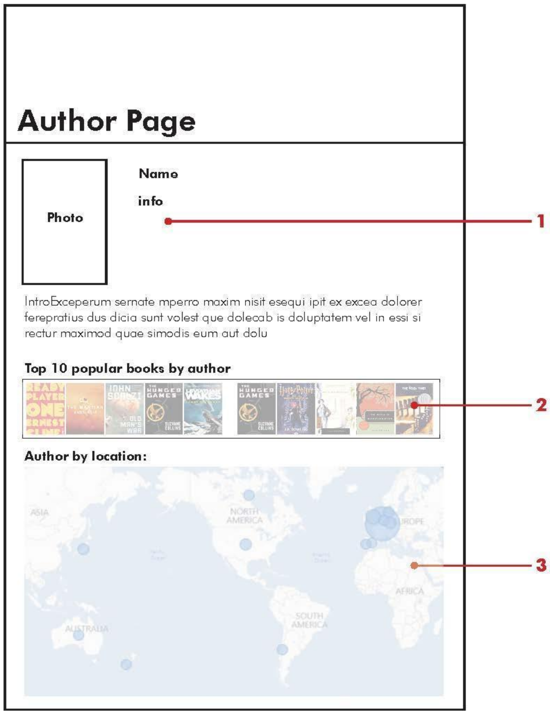
Given a **book_title**, give a list of books based on the user's past book query or popular shelves without revealing information about title or author, only showing blurred covers.

- Approach 1: find similar books’ similar books that are not originally similar

- Approach 2: Use collaborative filtering

  Find book similarity score base on user interaction

  Then order by similarity score and take books that are 60% - 70% similar


Author Page
Photo
Name
info
IntroExceperum sernate mperro maxim nisit esequi ipit ex excea dolorer ferepratius dus dicia sunt volest que dolecab is doluptatem vel in essi si rectur maximod quae simodis eum aut dolu
Top 10 popular books by author
Author by location:
ASIA
NORTH AMERICA
EUROPE
AFRICA
SOUTH AMERICA
AUSTRALIA
1
2
3

1) Basic info of author **MUST**

2) Top ten popular books by author **MUST**

3) Locate author on the world map, which will also show the most popular authors’ location **MAYBE**

3. **List of pages the application will have and a 1-2 sentence description of each page. We expect that the functionality of each page will be meaningfully different from the functionality of the other pages.**

   We plan to design 5 pages, homePage, bookCatalogPage, authorCatalogPage, bookPage and authorPage.

   1. **The homePage** is the main page, which will include sign-up and log in, showcase top 5 books and authors, search by book or author (toggle first, then search), some popular comments and tags.

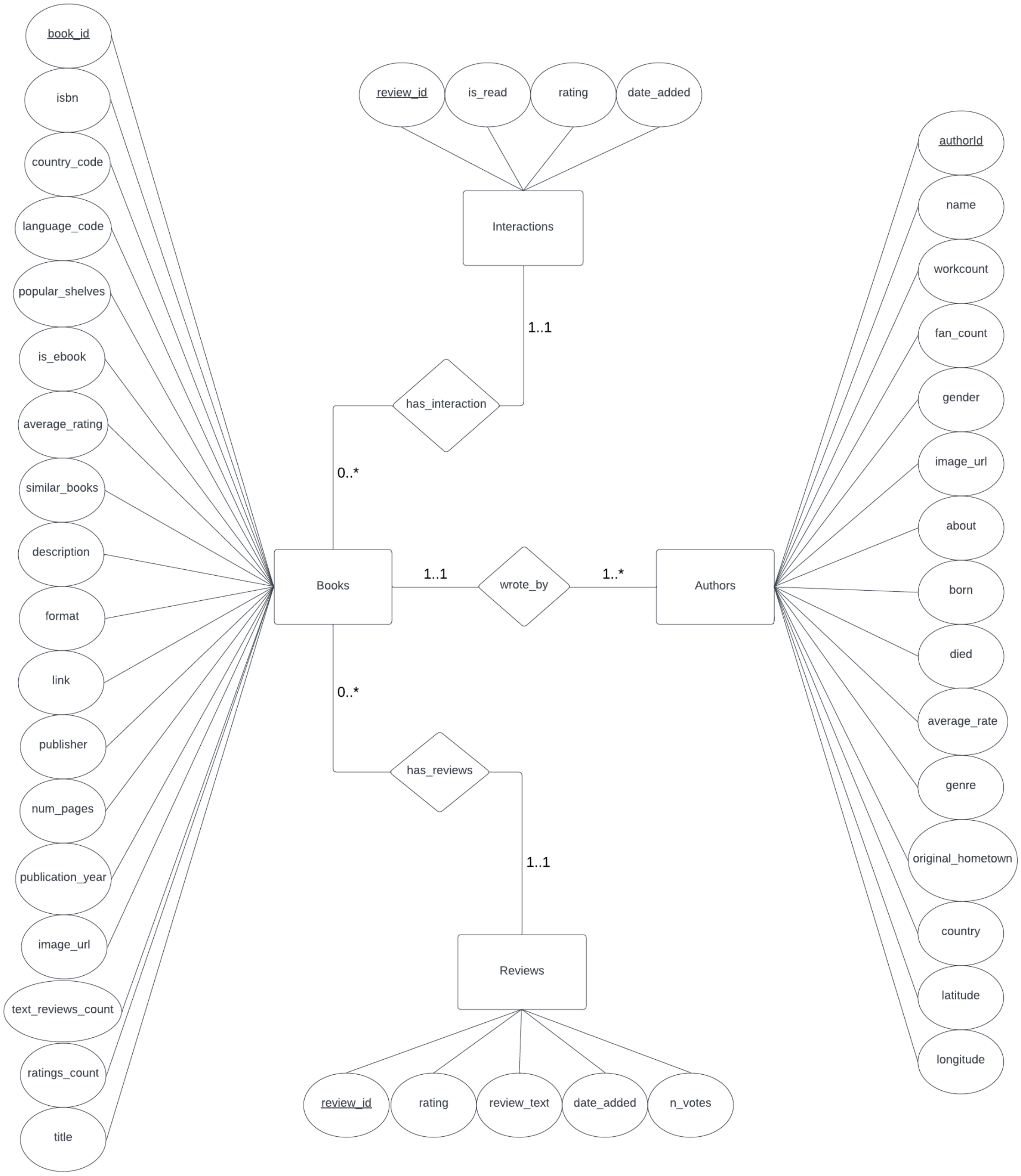
   2. **The bookCatalogPage** is the page after choosing ‘book’ in the toggle, put input in the search bar, and clicking the ‘search’ button. It lists all comic books found given the search keyword. Our users could sort books by title, author, average rating and tags.

   3. **The authorCatalogPage** is the page after choosing ‘author’ in the toggle, put input in the search bar, and clicking the ‘search’ button. It lists all the authors found given the search keyword. Our users could sort their search result by average rating and tags.

   4. **The bookPage** lists detailed information about that book, including the book’s basic info, top review, similar books, and potentially the surprise me section which will randomly showcase 1 book in the bottom section.

   5. **The authorPage** lists the basic info of the author, up to top ten popular books by author and potentially their location on the world map, which will also show the most popular authors’ location.

4. Relational schema as an ER diagram

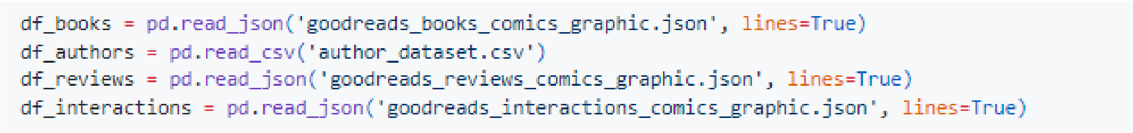


5. SQL DDL for creating the database

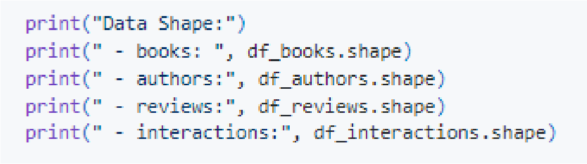
```
CREATE TABLE Authors(
   authorId          INT,
   name              VARCHAR(255),
   workcount          INT(8),
   fan_count          INT(8),
   gender            CHAR(7),
   image_url          VARCHAR(255),
   about             TEXT,
   born              CHAR(10),
   died              CHAR(10),
   average_rate        DOUBLE,
   genre             TEXT,
   original_hometown text  CHAR(50),
   country           CHAR(14),
   latitude          DOUBLE,
   longitude          DOUBLE,
   image_type          CHAR(8),
   PRIMARY KEY (authorId)
);

CREATE TABLE Books
(
   book_id              INT,
   isbn                DOUBLE,
   country_code         CHAR(2),
   language_code         CHAR(5),
   popular_shelves       TEXT,
   is_ebook             CHAR(5),
   average_rating        DOUBLE,
   similar_books         TEXT,
   description           TEXT,
   format               CHAR(50),
   link                VARCHAR(255),
   publisher            VARCHAR(255),
   num_pages             INT,
   publication_year      INT,
   url                 VARCHAR(255),
   image_url             VARCHAR(255),
   title               VARCHAR(255),
   authors          INT,
   PRIMARY KEY (book_id),
   FOREIGN KEY (authors) REFERENCES Authors(authorId)
);


CREATE TABLE Interactions
(
   review_id CHAR(32),
   is_read CHAR(5),
   rating DOUBLE,
   date_added CHAR(10),
   book_id INT,
   PRIMARY KEY (review_id),
   FOREIGN KEY (book_id) REFERENCES Books(book_id)
);


CREATE TABLE Reviews
(
   review_id CHAR(32),
   rating DOUBLE,
   review_text TEXT,
   date_added CHAR(10),
   n_votes INT,
   book_id INT,
   PRIMARY KEY (review_id),
   FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

6. Data Processing

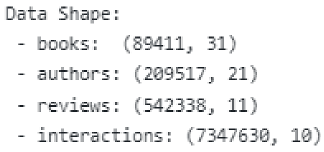
- **First step:** Read all the files into dataframe and inspect the number of rows and attributes for each data frame. The data comes in different formats. The author information table is in csv format while the book information, reviews and interactions come in json.



```
df_books = pd.read_json('goodreads_books_comics_graphic.json', lines=True)
df_authors = pd.read_csv('author_dataset.csv')
df_reviews = pd.read_json('goodreads_reviews_comics_graphic.json', lines=True)
df_interactions = pd.read_json('goodreads_interactions_comics_graphic.json', lines=True)
```



```
print("Data Shape:")
print(" - books: ", df_books.shape)
print(" - authors:", df_authors.shape)
print(" - reviews:", df_reviews.shape)
print(" - interactions:", df_interactions.shape)
```

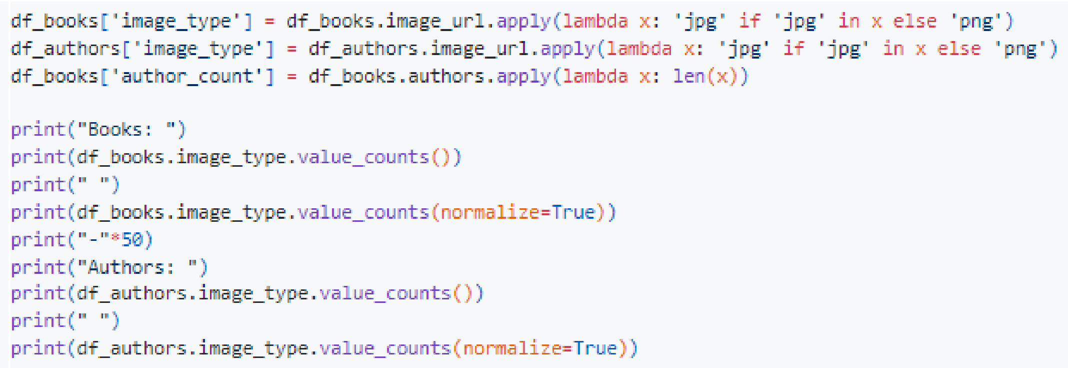


```
Data Shape:
 - books:  (89411, 31)
 - authors: (209517, 21)
 - reviews: (542338, 11)
 - interactions: (7347630, 10)
```

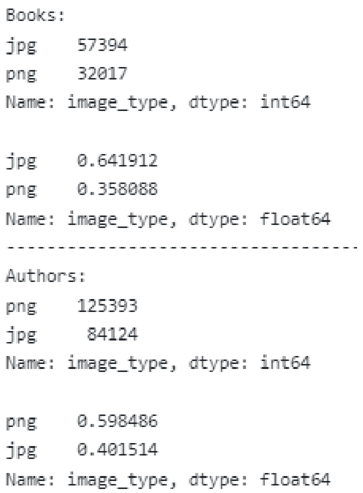
- **Second step**: filter out records with invalid image type

  After inspecting the dataset, we noticed that only jpg format images are valid. And since we used both the images in both the book and author section, we will have to filter out books with both valid author image and valid book cover image.

  o First, we inspected the number of entries and the percentage of book with both valid cover image and valid author image.

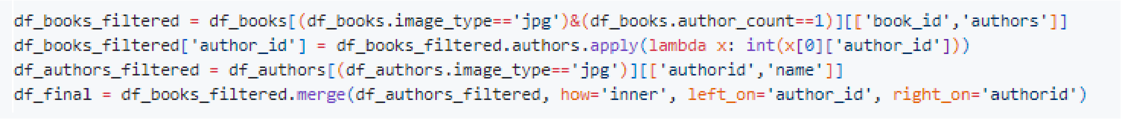


```
df_books['image_type'] = df_books.image_url.apply(lambda x: 'jpg' if 'jpg' in x else 'png')
df_authors['image_type'] = df_authors.image_url.apply(lambda x: 'jpg' if 'jpg' in x else 'png')
df_books['author_count'] = df_books.authors.apply(lambda x: len(x))

print("Books: ")
print(df_books.image_type.value_counts())
print(" ")
print(df_books.image_type.value_counts(normalize=True))
print("-"*50)
print("Authors: ")
print(df_authors.image_type.value_counts())
print(" ")
print(df_authors.image_type.value_counts(normalize=True))
```



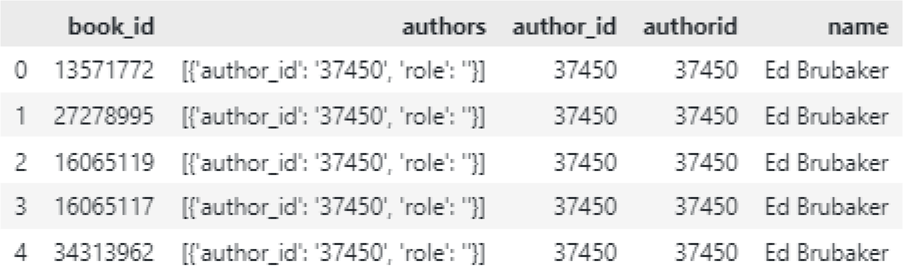
```
Books:
jpg    57394
png    32017
Name: image_type, dtype: int64

jpg    0.641912
png    0.358088
Name: image_type, dtype: float64
--------------------------------------------------
Authors:
png    125393
jpg     84124
Name: image_type, dtype: int64

png    0.598486
jpg    0.401514
Name: image_type, dtype: float64
```

- Then we filter out entries with invalid author image or invalid cover image by doing a inner join of valid entries of author and book

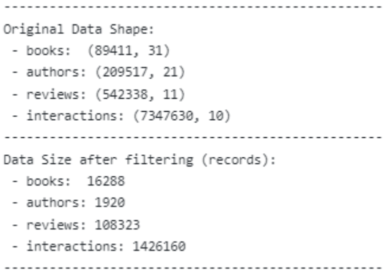


```python
df_books_filtered = df_books[(df_books.image_type=='jpg')&(df_books.author_count==1)][['book_id','authors']]
df_books_filtered['author_id'] = df_books_filtered.authors.apply(lambda x: int(x[0]['author_id']))
df_authors_filtered = df_authors[(df_authors.image_type=='jpg')][['authorid','name']]
df_final = df_books_filtered.merge(df_authors_filtered, how='inner', left_on='author_id', right_on='authorid')
```

Final Result:



	book_id	authors	author_id	authorid	name
0	13571772	[{'author_id': '37450', 'role': ''}]	37450	37450	Ed Brubaker
1	27278995	[{'author_id': '37450', 'role': ''}]	37450	37450	Ed Brubaker
2	16065119	[{'author_id': '37450', 'role': ''}]	37450	37450	Ed Brubaker
3	16065117	[{'author_id': '37450', 'role': ''}]	37450	37450	Ed Brubaker
4	34313962	[{'author_id': '37450', 'role': ''}]	37450	37450	Ed Brubaker



```
--------------------------------------------------
Original Data Shape:
 - books:  (89411, 31)
 - authors: (209517, 21)
 - reviews: (542338, 11)
 - interactions: (7347630, 10)
--------------------------------------------------
Data Size after filtering (records):
 - books:  16288
 - authors: 1920
 - reviews: 108323
 - interactions: 1426160
--------------------------------------------------
```

7. List of technologies you will use.

- Backend: AWS RDS MySQL database
- Data Cleaning: Python(pandas)
- Frontend: Nodejs, React, HTML

8. Description of what each group member will be responsible for

Backend:

- Data cleaning/ Data Processing: Hongri Jia
- AWS RDS Database Setup: Yazhuo Wang
- Database ERD, normalization: Jiameng Chen
- Database queries: Hongri Jia, Yazhuo Wang, Jiameng Chen, Yi Cao

Frontend(divided by pages)

- HomePage
- BookPage
- AuthorPage
- Book catalog page and Author catalog page

(Each team member will work on one of the pages and develop both the server side query and client side hook and display)