Real Time & Big Data Analytics

Professor Tang

Term Project Report

Sachin Malepati (sm9449)
Vipul Goyal (vg2134)
Atharva Moroney (amm9801)
Deepali Chugh (dc4600)

Table of Contents

Abstract

Book Recommendation Systems are widely popular these days and are used by almost all book sellers. Not only are they beneficial for the bibliophiles, but also for the book sellers for a myriad of reasons. A good book recommendation system is a key factor in increasing the sales of a book seller as it increases the visibility of their books and creates a better customer experience. In this project, we aim to build a Book Recommendation System based on the UCSD Goodreads dataset. We also aim to gain valuable insights from the data which can be used for finding the reading trends and publishing trends for a particular year. The tools and technologies used in this project are MapReduce to clean the datasets and Hive to produce relevant results in the form of trends and recommendations.

Introduction

A book recommendation system is important for customer engagement since the more customers read, the more revenue they generate. Undoubtedly, bibliophils are the primary beneficiaries of book recommendation systems. Other beneficiaries include agencies and publishing houses. The better the recommendation system that a book seller uses, the better is their customer engagement and experience. Moreover, such recommendation systems are a key to increasing the visibility of books in book stores, whether online or offline. Online book sellers such as Kindle, Amazon, Goodreads, etc. compete on many factors and the success is majorly dependent on the goodness of their recommendation systems. Motivated by this fact, we decided to build a book recommendation system for this project.

There are many publicly available datasets online on websites such as Kaggle, Google Datasets, etc. We have chosen the UCSD Goodreads dataset for implementing this project, because it has comprehensive information about the books, authors, genres and user-book interactions. The size of this dataset is ~32 GB and the dataset contains more than 230 million interactions, which is excellent for building a good recommendation system. In this project, we aim to provide two types of results - a robust Book Recommendation System, and some insights into reading and publishing trends among the users. While recommending books to a user, we consider user's own ratings, similar user ratings, genre, number of ratings, and negatively reviewed books.

The Book Recommendation System will suggest books to the user based on -

a) top-rated books similar to the ones the user has rated the highest
b) top-rated books in a genre the user has rated the highest
c) top-rated books by other users who have read similar books as this particular user

Apart from recommendations, we also perform some analytics on the data to find the reading and publishing trends among the users. We have observed some interesting trends, which are mentioned in the Results section. The detailed process of obtaining recommendations is described in the Design Diagram and Results sections.

Goodness

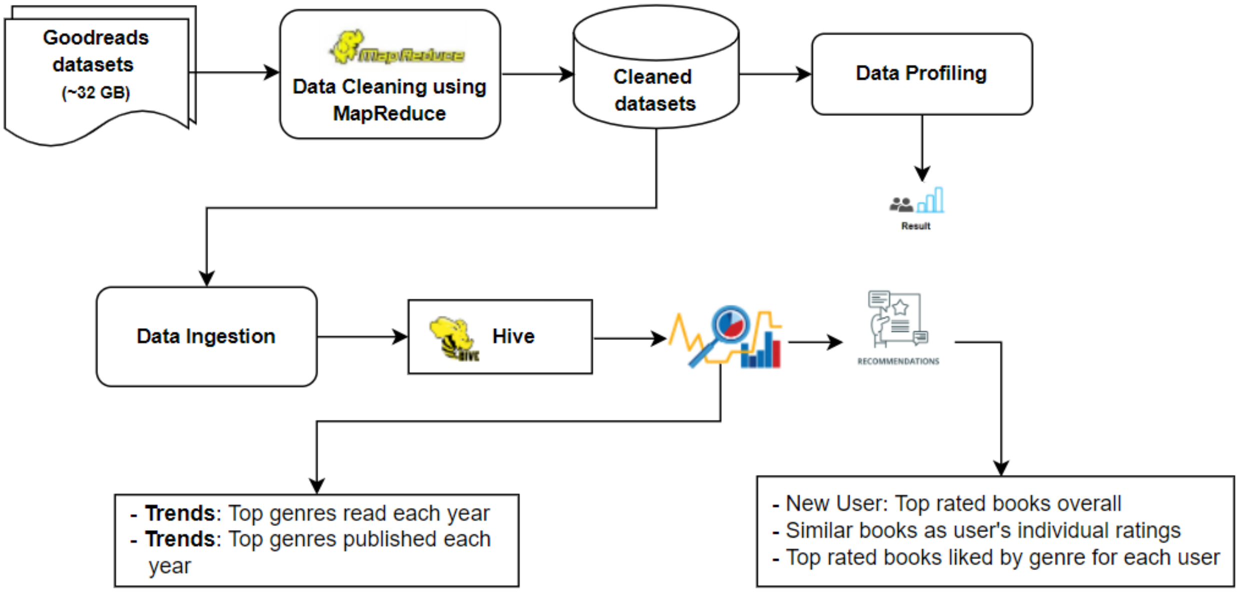
Robustness and accuracy are key factors for the success of any recommendation system. Also, we should have reasonable confidence the trends that we see are indeed legitimate. To ensure these, we take some precautions while building the recommendation system as well as finding the trends. Particularly, to discover the top-rated books overall, we only evaluate books with more ratings than the average number of ratings. This ensures that our results are not biased because of the books which are rated very high or very low but have very few ratings. It is difficult to judge the quality of a book based only on a small number of ratings, hence we disregard such books from our recommendations. Also, we remove interactions in which the books are present on the user's shelf but have not yet been read. This will ensure that only the books which have been read and rated by the user will play a role in recommendations. In terms of our generated insights, we observe that the trends observed in the dataset are also observed in many published trends online.

Datasets

The dataset used in this project is the Goodreads dataset, collected in late 2017 from goodreads.com by UCSD using only users’ public shelves. The cumulative size of all the datasets used in the project is ~32 GB.

We have used **five** datasets of both CSV and JSON format. The first dataset is the **Books** dataset (JSON format) of size 9.2 GB which contains the metadata and information about ~2.36 million books. The second dataset used in this project is the **Authors** dataset of 105 MB containing the metadata about authors. This is followed by the third dataset **Genres** (JSON format) of size 200 MB, which consists of the genres that the different books belong to. In this dataset, it is noticed that a single book may belong to multiple genres and hence, this is processed further during the project. The next dataset used is the **Shelves** dataset (CSV format) of size 4.5 GB that comprises the complete user-book interactions containing ~229 million interactions. The last dataset used in this project is the **Reviews** dataset (JSON format) of size 16.7 GB, comprising 15.7 million user reviews with detailed review texts by the user.

Design Diagram


Goodreads datasets (~32 GB)
MapReduce
Data Cleaning using MapReduce
Cleaned datasets
Data Profiling
Result
Data Ingestion
Hive
RECOMMENDATIONS
- Trends: Top genres read each year
- Trends: Top genres published each year
- New User: Top rated books overall
- Similar books as user's individual ratings
- Top rated books liked by genre for each user

The procedure for this project follows a four step process, beginning with fetching a sufficiently large dataset of ~32 GB which comprised of the meta-data of 2.36 million books, 229 million user-book interactions, 15 million records of user reviews with their detailed review texts as well as different genres that the books belonged to. The first step is to clean the data using MapReduce. The cleaned data is then processed and data profiling is executed resulting in information such as unique users, unique genres and authors while also finding books that have missing genres in their dataset.The next step is data ingestion into Hive where after querying data, two kinds of relevant and insightful results are found: trends and recommendations, which are further elaborated in detail in the Results section.

Data Cleaning and profiling

The data cleaning and profiling step is done to ensure that all the data conforms to a particular standard and can be ingested into Hive tables. The main purpose of the data cleaning step is to eliminate noisy data and malformed records. Data profiling is performed to gain some preliminary information about the size and contents of the data, to determine how the data can be structured into Hive tables and to decide on the next steps required for the main analysis.

We now describe the data cleaning and profiling on each of our 5 data sources.

Books Dataset

In this dataset, each record consists of data related to books. Three MapReduce programs are written to clean and profile the dataset. For cleaning, We have ignored isbn, text_reviews_count, series, popular_shelves, asin, kindle_asin, description, link, publisher, publication_day,isbn13, publication_month, edition_information, url, image_url, work_id, title_without_series. We have replaced any commas in the book title so that it won't be problematic when converting the cleaned data into a csv file. Similar_book_id: which includes book id and similar book id. Because there are several similar books for a specific book, we have created a separate file in which we stored the same book id with each similar book id in order to store the data into the hive. Authors: which includes the author and the book id, Because there are several authors for a specific book, we have created a separate file in which we stored the different author id with a particular book id in order to store the data into the hive. Books: includes country code, language code, is ebook, average rating, format, book id, ratings count, work id, title, and publication_year**.**

Shelves/Interactions dataset

For this dataset, we have written 2 MapReduce programs - one for cleaning the data and one for profiling. In the first program, we adopt the Replicated Join MapReduce pattern because the two mapping files (book_id_map.csv and user_id_map.csv) are quite small in size (~50 MB each), hence can be easily accommodated in the memory. In the setup() method, we read the 2 files from DistributedCache and store in respective hashmaps in memory. In the map() method, we output each record with the respective mapped user_ids and book_ids. As this is a replicated join pattern, there is no reducer in this program. In this program, we discard the records where is_read = 0 because such records are not useful for our purpose of building a book recommendation system. We have also included counters for total number of records accepted, malformed records, and records discorded becauses is_read is 0. In the second program, we perform data profiling. Specifically, we find out the number of unique authors and number of unique books. This will be useful in performing data further analysis using Hive in the second phase, as it will give us an estimate of the number of rows during table joins. We adopt the typical Distinct data summarization MapReduce pattern for this.

Following table profiles the Interactions dataset -

Measure	**Value**
Total Number of records	228,648,342
Records discarded because of is_read=0	116,517,139
Records accepted	112,131,203
Malformed records	0

Number of unique books	2,339,815
Number of unique authors	836,433

Authors dataset

In this dataset, each record consists of data related to a single Author. The columns present in this data are author id, author name, review count, rating and rating count. One MapReduce program is written to clean and profile the dataset. For cleaning, we have ignored the rating count and review count columns as they are not necessary for the analysis. We also had to replace any commas in the name so that it wont be problematic when converting the cleaned data into a csv file. Replaced all the null ratings to the value zero. Ignored all the columns for which the author id is not known.

For the profiling, we have used summarization patterns and used counters to track various attributes. Following are the profiling results obtained after the job is completed.

Measure	**Count**
Empty Author name or ID	5
Average Ratings between 0 and 1	3625
Average Ratings between 1 and 2	3824
Average Ratings between 2 and 3	31787
Average Ratings between 3 and 4	436220
Average Ratings between 4 and 5	325547

Genres dataset

In this dataset, each record consists of data related to a single Book and corresponding Genres. The columns present in this data are Book id, Array of strings containing genre details. One MapReduce program is written to clean and profile the dataset. For cleaning, we skip all the rows with empty book id. Obtain only the first and most relevant genre from the list of genres. Replacing all the commas in the genre name with a space.

For the profiling, we have used summarization patterns and used counters to track various attributes. Following are the profiling results obtained after the job is completed.

Measure	Count
Empty Genre	409513
Crime	164500
Fantasy	31666
Fiction	800781
History	172163
Mystery	164500
Poetry	25068
Romance	658719
Thriller	164500
Total rows	2360655

Reviews Dataset

In case of this dataset, each record consisted of multiple values that did not directly contribute to the recommendation system, thereby redundantly increasing the processing time of the dataset. The initial version of the dataset contains records with column n_votes, n_comments that signify the other user interaction with a particular user's review, date_updated that signify the date of review update and review_text that do not directly contribute towards recommendations in this project. Hence, with the step of data cleaning, the columns user_id, book_id and read_at have been retained. In the read_at column, the format used is Day, Date and Time. In the recommendation system, since the recommendations are unaffected by the time and day of review, the column is restructured and the year in which the respective book is read is retained, removing the remaining elements from the value. Additionally, there were a number of records where the year in which the book was read was missing. Hence, this value was replaced by zero. After data cleaning, the dataset size was reduced to ~8 GB. The next step in the process was data profiling, whereby unique users who reviewed were listed. Each user, having read multiple books, had reviewed multiple times in the dataset. Hence, this step resulted in providing unique ~456k users.

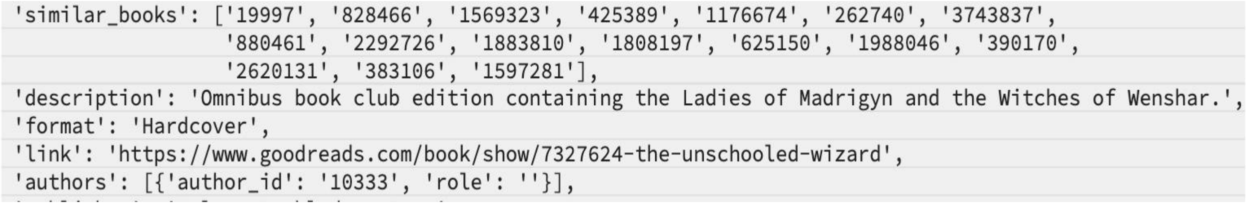
Coding Challenges:

During the implementation of the problem statement, we have encountered many challenges with respect to the data and the tools used in the process. Below, we will try to cover the issues we faced and how we managed to overcome them.

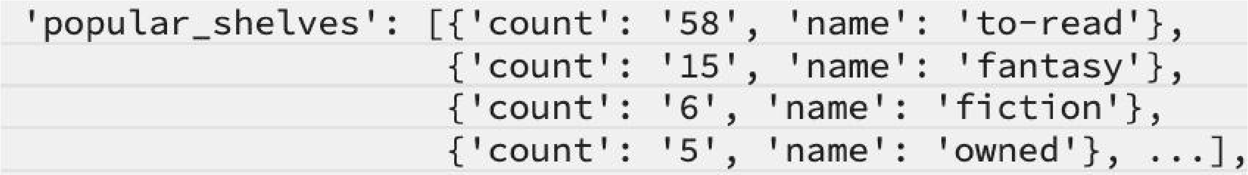
Challenge 1:

In the Book Authors dataset, we came across the structure of the data which consists of multiple nesting of json objects. So, to get the required fields out of the data, we had to parse the input multiple times to convert all the levels of nested json layers.

Below is the example of nesting we encountered:

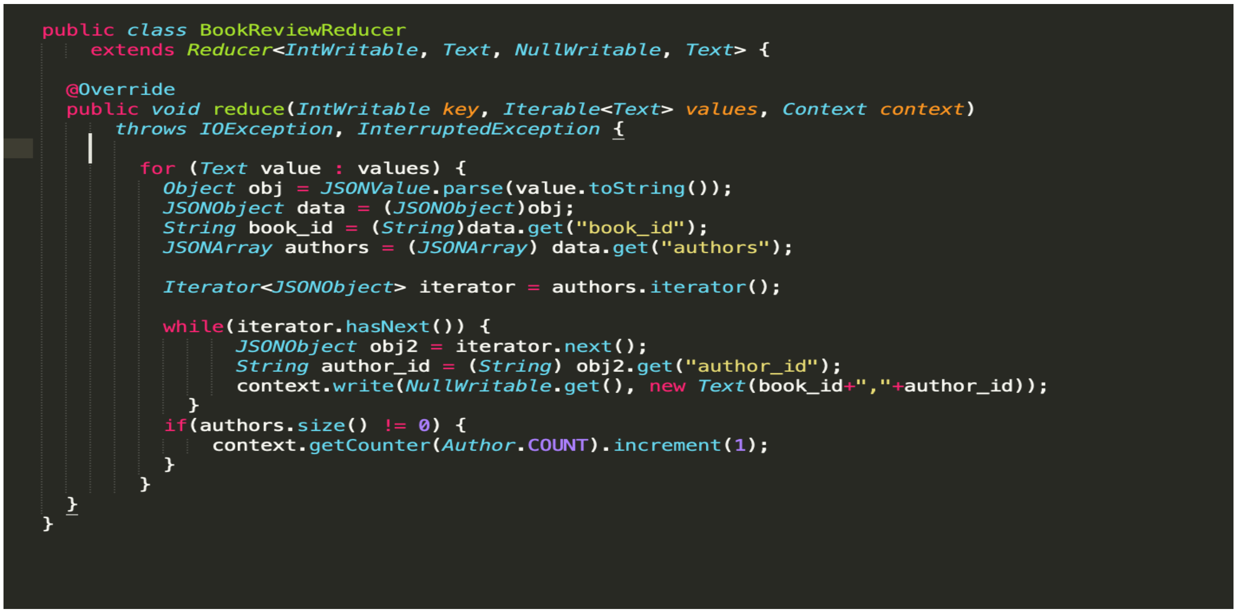


```
'similar_books': ['19997', '828466', '1569323', '425389', '1176674', '262740', '3743837',
                  '880461', '2292726', '1883810', '1808197', '625150', '1988046', '390170',
                  '2620131', '383106', '1597281'],
'description': 'Omnibus book club edition containing the Ladies of Madrigyn and the Witches of Wenshar.',
'format': 'Hardcover',
'link': 'https://www.goodreads.com/book/show/7327624-the-unschooled-wizard',
'authors': [{'author_id': '10333', 'role': ''}],
```



```
'popular_shelves': [{'count': '58', 'name': 'to-read'},
                    {'count': '15', 'name': 'fantasy'},
                    {'count': '6', 'name': 'fiction'},
                    {'count': '5', 'name': 'owned'}, ...],
```

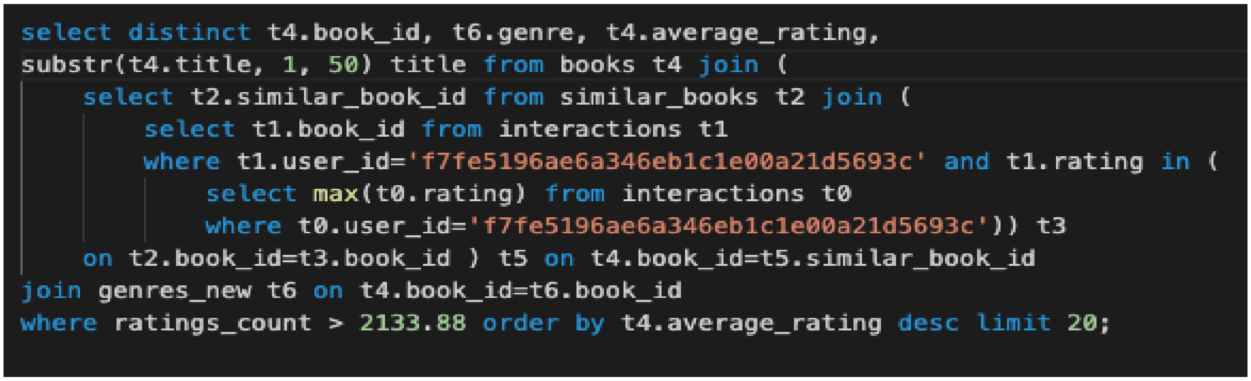
And, we handled the nesting in the following way:



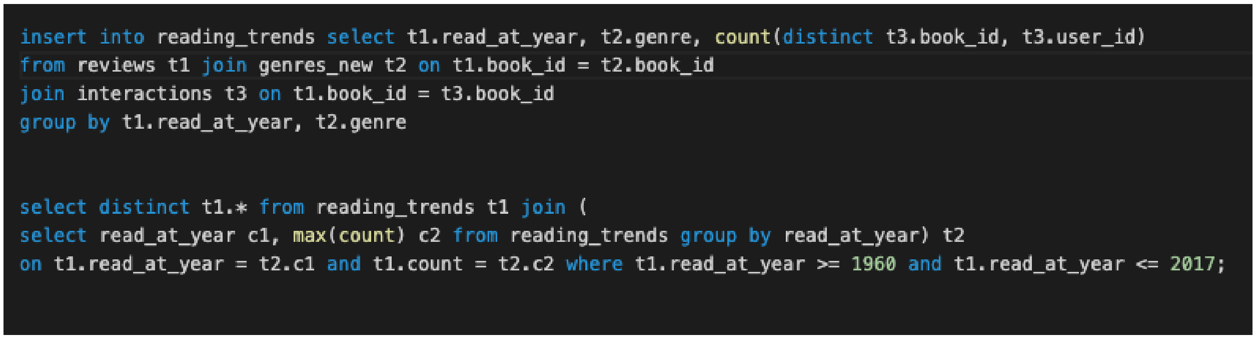
```
public class BookReviewReducer
    extends Reducer<IntWritable, Text, NullWritable, Text> {

  @Override
  public void reduce(IntWritable key, Iterable<Text> values, Context context)
      throws IOException, InterruptedException {

      for (Text value : values) {
        Object obj = JSONValue.parse(value.toString());
        JSONObject data = (JSONObject)obj;
        String book_id = (String)data.get("book_id");
        JSONArray authors = (JSONArray) data.get("authors");

        Iterator<JSONObject> iterator = authors.iterator();

        while(iterator.hasNext()) {
              JSONObject obj2 = iterator.next();
              String author_id = (String) obj2.get("author_id");
              context.write(NullWritable.get(), new Text(book_id+","+author_id));
          }
        if(authors.size() != 0) {
            context.getCounter(Author.COUNT).increment(1);
        }
      }
  }
}
```

Challenge 2:

To deal with multiple tables, we wanted to use nested sub-queries to get the desired output. We were able to write queries with one level of nesting but faced many difficulties in producing multiple levels of nested sub-queries. To handle this, we have generated multiple intermediate tables so that we can achieve a similar result of nested sub-queries. The implementation looks as follows.



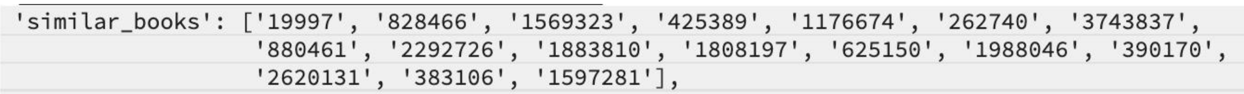
```
select distinct t4.book_id, t6.genre, t4.average_rating,
substr(t4.title, 1, 50) title from books t4 join (
    select t2.similar_book_id from similar_books t2 join (
        select t1.book_id from interactions t1
        where t1.user_id='f7fe5196ae6a346eb1c1e00a21d5693c' and t1.rating in (
            select max(t0.rating) from interactions t0
            where t0.user_id='f7fe5196ae6a346eb1c1e00a21d5693c')) t3
    on t2.book_id=t3.book_id ) t5 on t4.book_id=t5.similar_book_id
join genres_new t6 on t4.book_id=t6.book_id
where ratings_count > 2133.88 order by t4.average_rating desc limit 20;
```



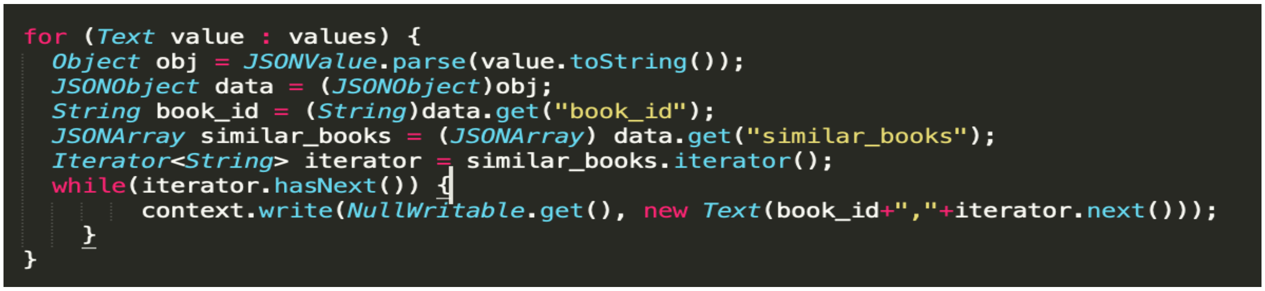
```
insert into reading_trends select t1.read_at_year, t2.genre, count(distinct t3.book_id, t3.user_id)
from reviews t1 join genres_new t2 on t1.book_id = t2.book_id
join interactions t3 on t1.book_id = t3.book_id
group by t1.read_at_year, t2.genre


select distinct t1.* from reading_trends t1 join (
select read_at_year c1, max(count) c2 from reading_trends group by read_at_year) t2
on t1.read_at_year = t2.c1 and t1.count = t2.c2 where t1.read_at_year >= 1960 and t1.read_at_year <= 2017;
```

Challenge 3:

In the Books dataset, there is a column named “similar books” which contained an array of strings with book id’s similar to the given book. To make the queries simple and readable, we had to convert this column into a new table. So, for every book id and a list of similar book id’s, we insert a row in the new table with the current book id and one of the entries of the similar books array. Therefore, for each book there will be k number of entries in the new table given there are k similar books.



```
'similar_books': ['19997', '828466', '1569323', '425389', '1176674', '262740', '3743837',
                  '880461', '2292726', '1883810', '1808197', '625150', '1988046', '390170',
                  '2620131', '383106', '1597281'],
```



```java
for (Text value : values) {
  Object obj = JSONValue.parse(value.toString());
  JSONObject data = (JSONObject)obj;
  String book_id = (String)data.get("book_id");
  JSONArray similar_books = (JSONArray) data.get("similar_books");
  Iterator<String> iterator = similar_books.iterator();
  while(iterator.hasNext()) {
        context.write(NullWritable.get(), new Text(book_id+","+iterator.next()));
    }
}
```

Results

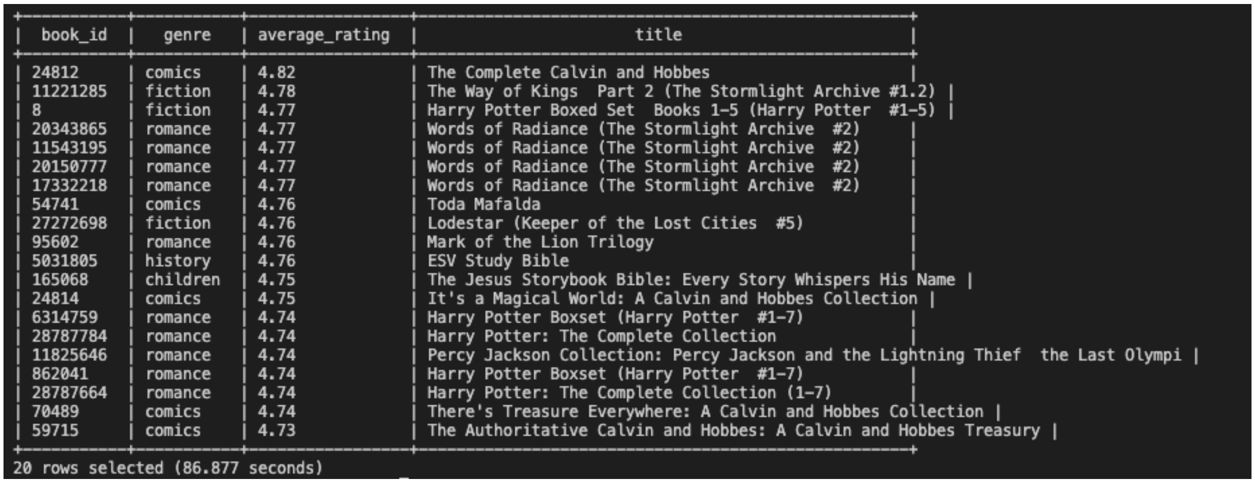
As mentioned in the Introduction, the motivation behind our project was to develop a Book Recommendation System as well as gain some valuable insights and trends from the data. Hence, we provide two types of results in this section - recommendations of 4 types and trends/insights. The insights specifically showcase the reading and publishing trends since the 1990's till 2017.

We now describe the 4 types of recommendations below.

Recommendation Type 1

The first type of recommendation is targeted towards a new user who has just registered on our platform. Clearly, since the user is new, they do not have any reading history on our platform. So we cannot know this user's likes or dislikes, nor do we have any clue about the reading tendencies of the user in terms of genres he prefers to read, any particular authors he likes, etc. Hence it makes sense to just provide such a user with the overall top-rated books across all genres. So in this type of recommendation, we simply find the 20 top rated books and display it to the user. While doing so, we exclude the books which have number of rating lower than average number of ratings.

Below image shows the result of Recommendation Type 1 -



```
+-----------+-----------+-----------------+-----------------------------------------------------+
|  book_id  |   genre   | average_rating  |                        title                        |
+-----------+-----------+-----------------+-----------------------------------------------------+
| 24812     | comics    | 4.82            | The Complete Calvin and Hobbes                      |
| 11221285  | fiction   | 4.78            | The Way of Kings  Part 2 (The Stormlight Archive #1.2) |
| 8         | fiction   | 4.77            | Harry Potter Boxed Set  Books 1-5 (Harry Potter  #1-5) |
| 20343865  | romance   | 4.77            | Words of Radiance (The Stormlight Archive  #2)      |
| 11543195  | romance   | 4.77            | Words of Radiance (The Stormlight Archive  #2)      |
| 20150777  | romance   | 4.77            | Words of Radiance (The Stormlight Archive  #2)      |
| 17332218  | romance   | 4.77            | Words of Radiance (The Stormlight Archive  #2)      |
| 54741     | comics    | 4.76            | Toda Mafalda                                        |
| 27272698  | fiction   | 4.76            | Lodestar (Keeper of the Lost Cities  #5)            |
| 95602     | romance   | 4.76            | Mark of the Lion Trilogy                            |
| 5031805   | history   | 4.76            | ESV Study Bible                                     |
| 165068    | children  | 4.75            | The Jesus Storybook Bible: Every Story Whispers His Name |
| 24814     | comics    | 4.75            | It's a Magical World: A Calvin and Hobbes Collection |
| 6314759   | romance   | 4.74            | Harry Potter Boxset (Harry Potter  #1-7)            |
| 28787784  | romance   | 4.74            | Harry Potter: The Complete Collection               |
| 11825646  | romance   | 4.74            | Percy Jackson Collection: Percy Jackson and the Lightning Thief  the Last Olympi |
| 862041    | romance   | 4.74            | Harry Potter Boxset (Harry Potter  #1-7)            |
| 28787664  | romance   | 4.74            | Harry Potter: The Complete Collection (1-7)         |
| 70489     | comics    | 4.74            | There's Treasure Everywhere: A Calvin and Hobbes Collection |
| 59715     | comics    | 4.73            | The Authoritative Calvin and Hobbes: A Calvin and Hobbes Treasury |
+-----------+-----------+-----------------+-----------------------------------------------------+
20 rows selected (86.877 seconds)
```

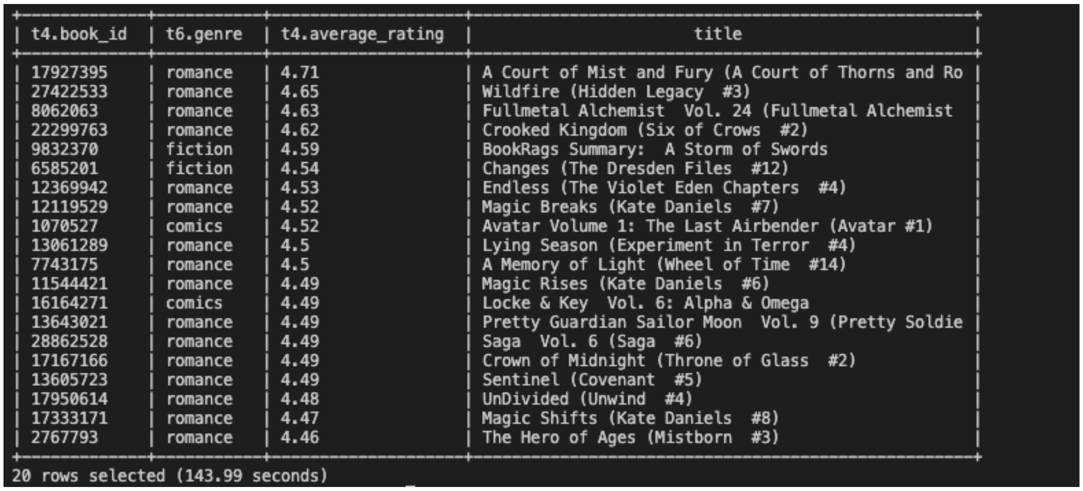
The query for this recommendation is -

```
select t1.book_id book_id, genre, average_rating, substr(title, 1, 10) title
from books t1 left join genres_new t2 on t1.book_id=t2.book_id
where ratings_count > 2133.88 order by average_rating desc limit 20;
```

Recommendation Type 2

Recommendations types 2, 3 & 4 are targeted towards existing users with some reading history. A user who has been on our platform for a while will have some recorded likes and dislikes, which we can leverage to provide recommendations to this user. Particularly, for type 2 recommendation, we recommend books similar to the ones the user has rated the highest. For instance, let’s say a user John has liked books B1, B2 and B3 and has rated them the highest. We go to our database and find the books which are similar to these three books. We then sort them based on their ratings and provide 20 recommendations to the user John.

Below image shows the result of Recommendation Type 2 -



```
+-------------+-----------+--------------------+------------------------------------------------------+
| t4.book_id  | t6.genre  | t4.average_rating  |                        title                         |
+-------------+-----------+--------------------+------------------------------------------------------+
| 17927395    | romance   | 4.71               | A Court of Mist and Fury (A Court of Thorns and Ro   |
| 27422533    | romance   | 4.65               | Wildfire (Hidden Legacy  #3)                         |
| 8062063     | romance   | 4.63               | Fullmetal Alchemist  Vol. 24 (Fullmetal Alchemist    |
| 22299763    | romance   | 4.62               | Crooked Kingdom (Six of Crows  #2)                   |
| 9832370     | fiction   | 4.59               | BookRags Summary:  A Storm of Swords                 |
| 6585201     | fiction   | 4.54               | Changes (The Dresden Files  #12)                     |
| 12369942    | romance   | 4.53               | Endless (The Violet Eden Chapters  #4)               |
| 12119529    | romance   | 4.52               | Magic Breaks (Kate Daniels  #7)                      |
| 1070527     | comics    | 4.52               | Avatar Volume 1: The Last Airbender (Avatar #1)      |
| 13061289    | romance   | 4.5                | Lying Season (Experiment in Terror  #4)              |
| 7743175     | romance   | 4.5                | A Memory of Light (Wheel of Time  #14)               |
| 11544421    | romance   | 4.49               | Magic Rises (Kate Daniels  #6)                       |
| 16164271    | comics    | 4.49               | Locke & Key  Vol. 6: Alpha & Omega                   |
| 13643021    | romance   | 4.49               | Pretty Guardian Sailor Moon  Vol. 9 (Pretty Soldie   |
| 28862528    | romance   | 4.49               | Saga  Vol. 6 (Saga  #6)                              |
| 17167166    | romance   | 4.49               | Crown of Midnight (Throne of Glass  #2)              |
| 13605723    | romance   | 4.49               | Sentinel (Covenant  #5)                              |
| 17950614    | romance   | 4.48               | UnDivided (Unwind  #4)                               |
| 17333171    | romance   | 4.47               | Magic Shifts (Kate Daniels  #8)                      |
| 2767793     | romance   | 4.46               | The Hero of Ages (Mistborn  #3)                      |
+-------------+-----------+--------------------+------------------------------------------------------+
20 rows selected (143.99 seconds)
```

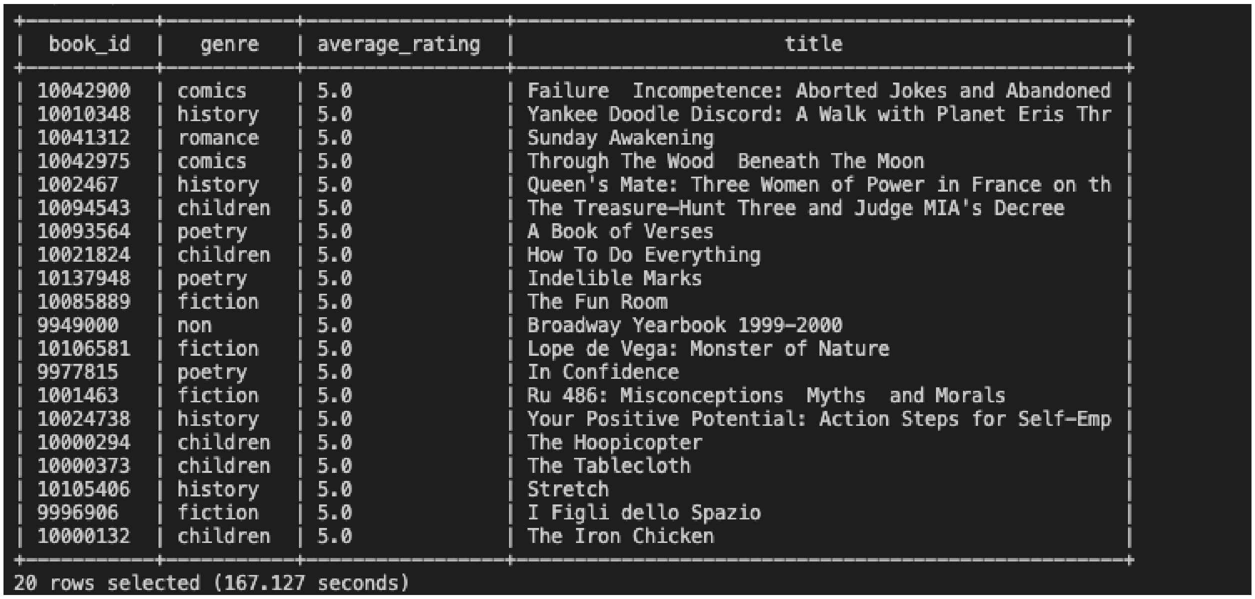
The query for this recommendation is -

```
select distinct t4.book_id, t6.genre, t4.average_rating, substr(t4.title, 1, 50) title
from books t4 join (
        select t2.similar_book_id from similar_books t2 join (
                select t1.book_id from interactions t1
                where t1.user_id='...' and t1.rating in (
                        select max(t0.rating) from interactions t0 where t0.user_id='...')) t3
        on t2.book_id=t3.book_id ) t5 on t4.book_id=t5.similar_book_id
join genres_new t6 on t4.book_id=t6.book_id where ratings_count > 2133.88
order by t4.average_rating desc limit 20;
```

Recommendation Type 3

This type of recommendation takes a little different approach. It is often the case that a particular user likes or prefers to read specific genres. Considering this, it would be beneficial for a user to receive recommendations for genres he likes the most. Type 3 caters to this particular use case. First, we find the books which are higly rated by the user and find their corresponding genres. We take top 2-3 such genres. Then, we find the books which are highest rated in those genres and sort them in descending order based on their average ratings and display them to the user.

Below image shows the result of Recommendation Type 3 -



```
+-----------+-----------+----------------+----------------------------------------------------+
|  book_id  |   genre   | average_rating |                       title                        |
+-----------+-----------+----------------+----------------------------------------------------+
| 10042900  | comics    | 5.0            | Failure  Incompetence: Aborted Jokes and Abandoned |
| 10010348  | history   | 5.0            | Yankee Doodle Discord: A Walk with Planet Eris Thr |
| 10041312  | romance   | 5.0            | Sunday Awakening                                   |
| 10042975  | comics    | 5.0            | Through The Wood  Beneath The Moon                 |
| 1002467   | history   | 5.0            | Queen's Mate: Three Women of Power in France on th |
| 10094543  | children  | 5.0            | The Treasure-Hunt Three and Judge MIA's Decree     |
| 10093564  | poetry    | 5.0            | A Book of Verses                                   |
| 10021824  | children  | 5.0            | How To Do Everything                               |
| 10137948  | poetry    | 5.0            | Indelible Marks                                    |
| 10085889  | fiction   | 5.0            | The Fun Room                                       |
| 9949000   | non       | 5.0            | Broadway Yearbook 1999-2000                        |
| 10106581  | fiction   | 5.0            | Lope de Vega: Monster of Nature                    |
| 9977815   | poetry    | 5.0            | In Confidence                                      |
| 1001463   | fiction   | 5.0            | Ru 486: Misconceptions  Myths  and Morals          |
| 10024738  | history   | 5.0            | Your Positive Potential: Action Steps for Self-Emp |
| 10000294  | children  | 5.0            | The Hoopicopter                                    |
| 10000373  | children  | 5.0            | The Tablecloth                                     |
| 10105406  | history   | 5.0            | Stretch                                            |
| 9996906   | fiction   | 5.0            | I Figli dello Spazio                               |
| 10000132  | children  | 5.0            | The Iron Chicken                                   |
+-----------+-----------+----------------+----------------------------------------------------+
20 rows selected (167.127 seconds)
```

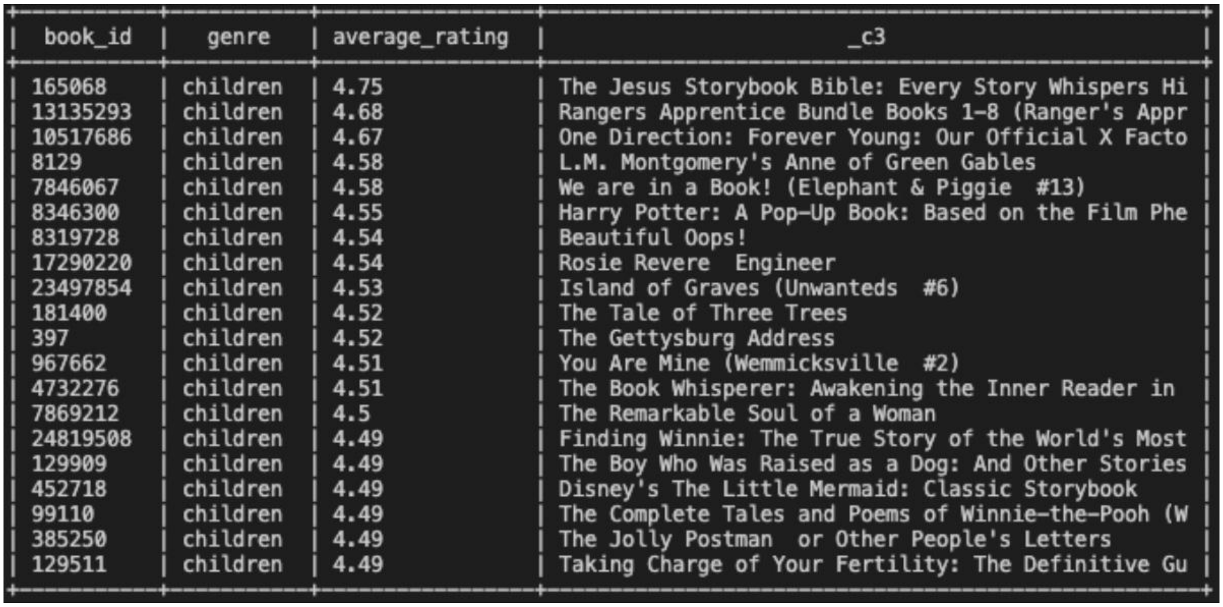
The query for this recommendation is -

```
select distinct t1.book_id book_id, t2.genre genre, t3.average_rating average_rating,
substr(t3.title, 1, 50) title
from interactions t1 join genres_new t2 on t1.book_id = t2.book_id
join books t3 on t1.book_id = t3.book_id
where t1.rating in (select max(rating) from interactions
where user_id='...')
and t3.ratings_count > 2133.88 and t2.genre != ‘’
order by t3.average_rating desc limit 20;
```

Recommendation Type 4

The final type of recommendation lets the user choose the genre which he wants to read. Many times a user is in a mood for reading particular genre. This type caters to such use cases. For instance, let’s say a parent wants to read a book to his or her child. Quite obviously, the parent would want to find a book suitable for children i.e. belonging to “Children” genre. In this case, the user (parent) will provide the genre and our platform finds the top rated books in this particular genre only. It sorts the books in the descending order of their average ratings and presents them to the user. Like in all other types of recommendations, in this type also, we filter out the books which have number of ratings lower than the average number of ratings.

Below image shows the result of Recommendation Type 4 -



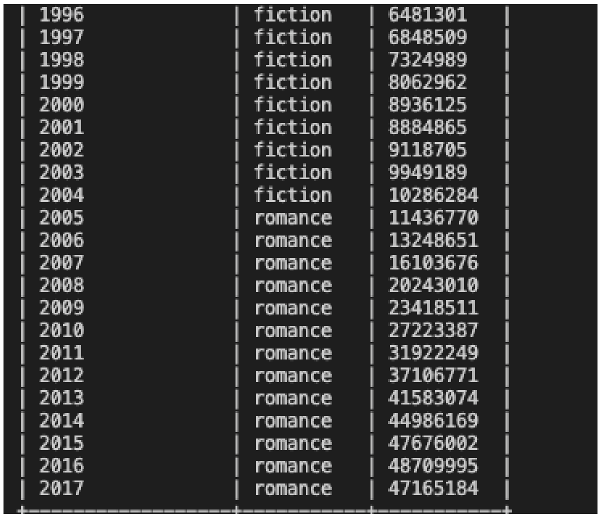
```
+------------+------------+------------------+----------------------------------------------------+
| book_id    | genre      | average_rating   | _c3                                                |
+------------+------------+------------------+----------------------------------------------------+
| 165068     | children   | 4.75             | The Jesus Storybook Bible: Every Story Whispers Hi |
| 13135293   | children   | 4.68             | Rangers Apprentice Bundle Books 1-8 (Ranger's Appr |
| 10517686   | children   | 4.67             | One Direction: Forever Young: Our Official X Facto |
| 8129       | children   | 4.58             | L.M. Montgomery's Anne of Green Gables             |
| 7846067    | children   | 4.58             | We are in a Book! (Elephant & Piggie  #13)         |
| 8346300    | children   | 4.55             | Harry Potter: A Pop-Up Book: Based on the Film Phe |
| 8319728    | children   | 4.54             | Beautiful Oops!                                    |
| 17290220   | children   | 4.54             | Rosie Revere  Engineer                             |
| 23497854   | children   | 4.53             | Island of Graves (Unwanteds  #6)                   |
| 181400     | children   | 4.52             | The Tale of Three Trees                            |
| 397        | children   | 4.52             | The Gettysburg Address                             |
| 967662     | children   | 4.51             | You Are Mine (Wemmicksville  #2)                   |
| 4732276    | children   | 4.51             | The Book Whisperer: Awakening the Inner Reader in  |
| 7869212    | children   | 4.5              | The Remarkable Soul of a Woman                     |
| 24819508   | children   | 4.49             | Finding Winnie: The True Story of the World's Most |
| 129909     | children   | 4.49             | The Boy Who Was Raised as a Dog: And Other Stories |
| 452718     | children   | 4.49             | Disney's The Little Mermaid: Classic Storybook     |
| 99110      | children   | 4.49             | The Complete Tales and Poems of Winnie-the-Pooh (W |
| 385250     | children   | 4.49             | The Jolly Postman  or Other People's Letters       |
| 129511     | children   | 4.49             | Taking Charge of Your Fertility: The Definitive Gu |
+------------+------------+------------------+----------------------------------------------------+
```

The query for this recommendation is -

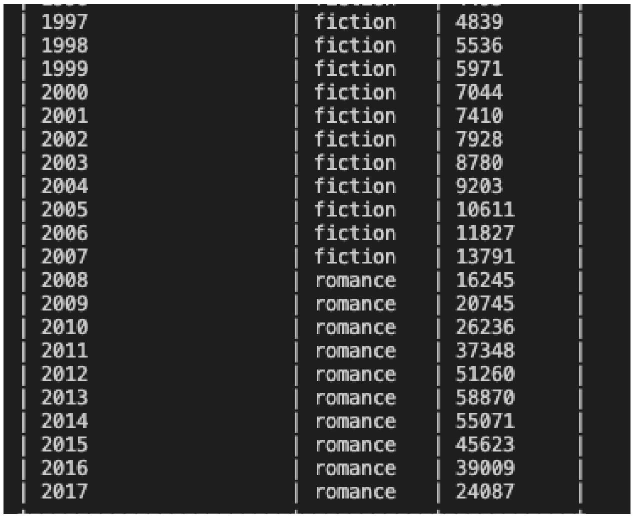
select t1.book_id book_id, genre, average_rating, substr(title, 1, 50) title from books t1 left join genres_new t2 on t1.book_id=t2.book_id where ratings_count > 2133.88 and genre='children' order by average_rating desc limit 20;

The next part of the results describes the insights we gained from the dataset. In particular, we observed two very interesting trends in the time period from 1990’s to 2017 in terms of number of books published and number of books read in particular genres.

Below images show the genres corresponding to each year in which the highest number of books were read and the highest number of books were published, respectively, and the corresponding number.



```
| 1996             | fiction   | 6481301   |
| 1997             | fiction   | 6848509   |
| 1998             | fiction   | 7324989   |
| 1999             | fiction   | 8062962   |
| 2000             | fiction   | 8936125   |
| 2001             | fiction   | 8884865   |
| 2002             | fiction   | 9118705   |
| 2003             | fiction   | 9949189   |
| 2004             | fiction   | 10286284  |
| 2005             | romance   | 11436770  |
| 2006             | romance   | 13248651  |
| 2007             | romance   | 16103676  |
| 2008             | romance   | 20243010  |
| 2009             | romance   | 23418511  |
| 2010             | romance   | 27223387  |
| 2011             | romance   | 31922249  |
| 2012             | romance   | 37106771  |
| 2013             | romance   | 41583074  |
| 2014             | romance   | 44986169  |
| 2015             | romance   | 47676002  |
| 2016             | romance   | 48709995  |
| 2017             | romance   | 47165184  |
+------------------+-----------+-----------+
```



```
| 1997                   | fiction   | 4839       |
| 1998                   | fiction   | 5536       |
| 1999                   | fiction   | 5971       |
| 2000                   | fiction   | 7044       |
| 2001                   | fiction   | 7410       |
| 2002                   | fiction   | 7928       |
| 2003                   | fiction   | 8780       |
| 2004                   | fiction   | 9203       |
| 2005                   | fiction   | 10611      |
| 2006                   | fiction   | 11827      |
| 2007                   | fiction   | 13791      |
| 2008                   | romance   | 16245      |
| 2009                   | romance   | 20745      |
| 2010                   | romance   | 26236      |
| 2011                   | romance   | 37348      |
| 2012                   | romance   | 51260      |
| 2013                   | romance   | 58870      |
| 2014                   | romance   | 55071      |
| 2015                   | romance   | 45623      |
| 2016                   | romance   | 39009      |
| 2017                   | romance   | 24087      |
```

We can observe a very interesting shift here. From 1990’s till mid 2000s, fiction was the dominant genre both in terms of number of books read and number of books published. But since mid-2000s, we observe that the inclination has changed to “romance” genre i.e. romance genre began to dominate in both terms. This is quite interesting to observe because since mid-2000s, we can generally observe that the overall inclination of people has changed to romance genre in terms of movies as well as music. Similar trends can be observed in music and movie industry.

Future Work

We would have worked on the following things given we had more time:

E-Books: Trends in adapting the e-books in the past 20 years.

Trending Authors: Analyze the trending patterns of Authors year-wise.

Customer Segmentation: Segregate all the users into clusters so that for any given user, we can recommend books based on the books liked by the similar users in the cluster.

Conclusion

In this project, we have successfully analyzed the trends in reading patterns and were able to recommend books to the users. We have worked on a total of 5 different data sets suming upto 32 GB of data. Each dataset is cleaned and profiled using map-reduce programs. The resulting data is then ingested to hive for further analysis. We were able to analyze the reading and publishing trends. We extracted top trending books year-wise and genre-wise. Finally, we were also able to implement four different types of recommendations for each user.

Acknowledgements

We wish to thank Professor Tang for continued guidance throughout the project. We also express our sincere gratitude towards the HPC support team for their quick turnaround times on our queries and providing crucial support.

References

- Mengting Wan, Julian McAuley, "Item Recommendation on Monotonic Behavior Chains" , in RecSys'18.
- Mengting Wan, Rishabh Misra, Napa Nakashole, Julian McAuley, "Fine-Grained Spoiler Detection from Large-Scale Review Corpora", In, ACL'19.
- Dataset: https://sites.google.com/enq.ucsd.edu/ucsdbookgraph/home