INFO7250 ENGINEERING BIG DATA SYSTEMS: TEAM 4

Final Project Report

Authors
Fredy Daruwala, Gautam Pawar, Nikita Anand, Tapadyuti Maiti
Date: 24th April 2017
Version 1.0

Table of Contents

Our Team: -

Fredy Daruwala
Gautam Pawar
Nikita Anand
Tapadyuti Maiti

Problem Statement/Objectives: -

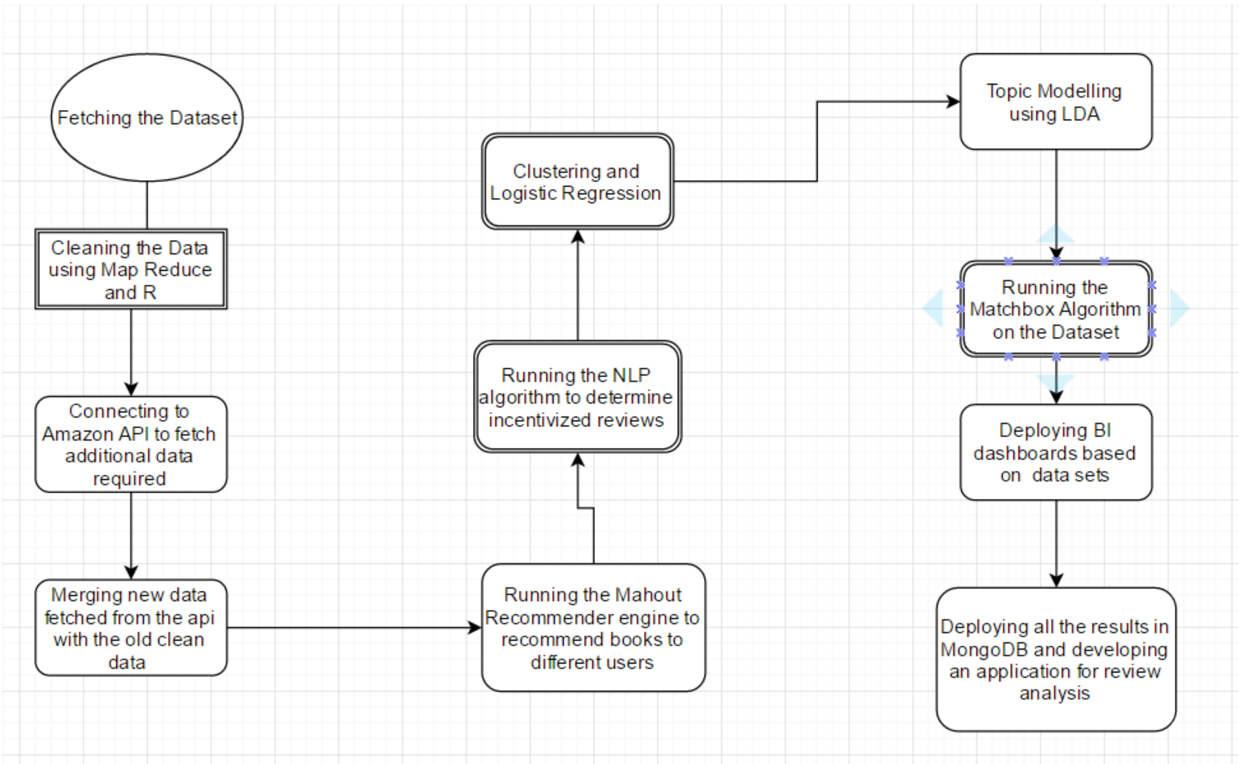
Online reviews play a very important role in information dissemination and play a major role in influencing a user's decision while buying a product. However, a user may only read a limited number of reviews before deciding to purchase an item. An important aspect to the success of a rating and reviews portal for amazon is to identify which reviews to promote as being useful. The main goal of our project is to analyze the Amazon Books Dataset based on the user's ratings and reviews and **host the results on our web application for a publisher** to analyze and determine the usefulness of a review. We are also performing an NLP sentiment analysis on the reviews to find out if a review has been incentivized or not. Sellers incentives for reviews aren't necessarily asking for positive reviews, but in general, they aren't handing out products that they suspect you'll hate. By law, these incentivized reviews must also be disclosed to the public to avoid dishonest sales practices.

For publishers to improve their book sales they need tools to carefully analyze the review data and study market behavior based on the reviews and the ratings users provide. This has made it essential to have enough tools so that the users and the publishers can easily analyze the data and derive meaningful results.

Problem Approach: -

Our main dataset file was an Amazon Books review dataset file which was a total size of 9 GB and is in the json format. The main challenge with this dataset was that since it was in the **Json format** we had to clean the data set and parse it to extract required information for analysis using Map and Reduce framework. It is not easy to parse such a huge data set file using normal Java, Python or R programs as it would take a large amount of time and Python would not be able to parse such a huge data set. We used 3 **Map and Reduce programs** to parse the dataset. Later we used the **Amazon Product Advertisement API** to fetch missing data in the Json file such as title, book author and genre of the books. We did this using the custom uri generated from Amazon to hit the api using a map-reduce program and fetch the additional data. After obtaining the additional data, we merged this data with the cleaned Json data **using R scripts**. Once we had cleaned and merged all the data, we hosted the data on mongodb and performed different recommendations and analysis on the same.

RoadMap: -



Application Use Cases: -

- Analyze the data from the Amazon Books Review dataset and the Amazon Product advertisement API.
- Displaying the top-rated books in the data set using Mongodb Queries.
- Analyze the dataset for recommendations based on user, books and their ratings.
- Show data visualizations for various categories of books using the Tableau server.
- Using NLP algorithms to predict whether a user review is incentivized or non-incentivized.

The application we built is a Spring MVC application with jsp pages as the front end and MongoDb database server as the backend. We staged all the clean data in a Mongodb table and interfaced it with our application. We also interfaced our application with the Amazon Product Advertisement API to fetch live data about the books from the Amazon Server.

Dataset Characteristics: -

Our dataset was the Amazon Books Review dataset. We obtained the dataset from the below URL: -
https://snap.stanford.edu/data/web-Amazon.html

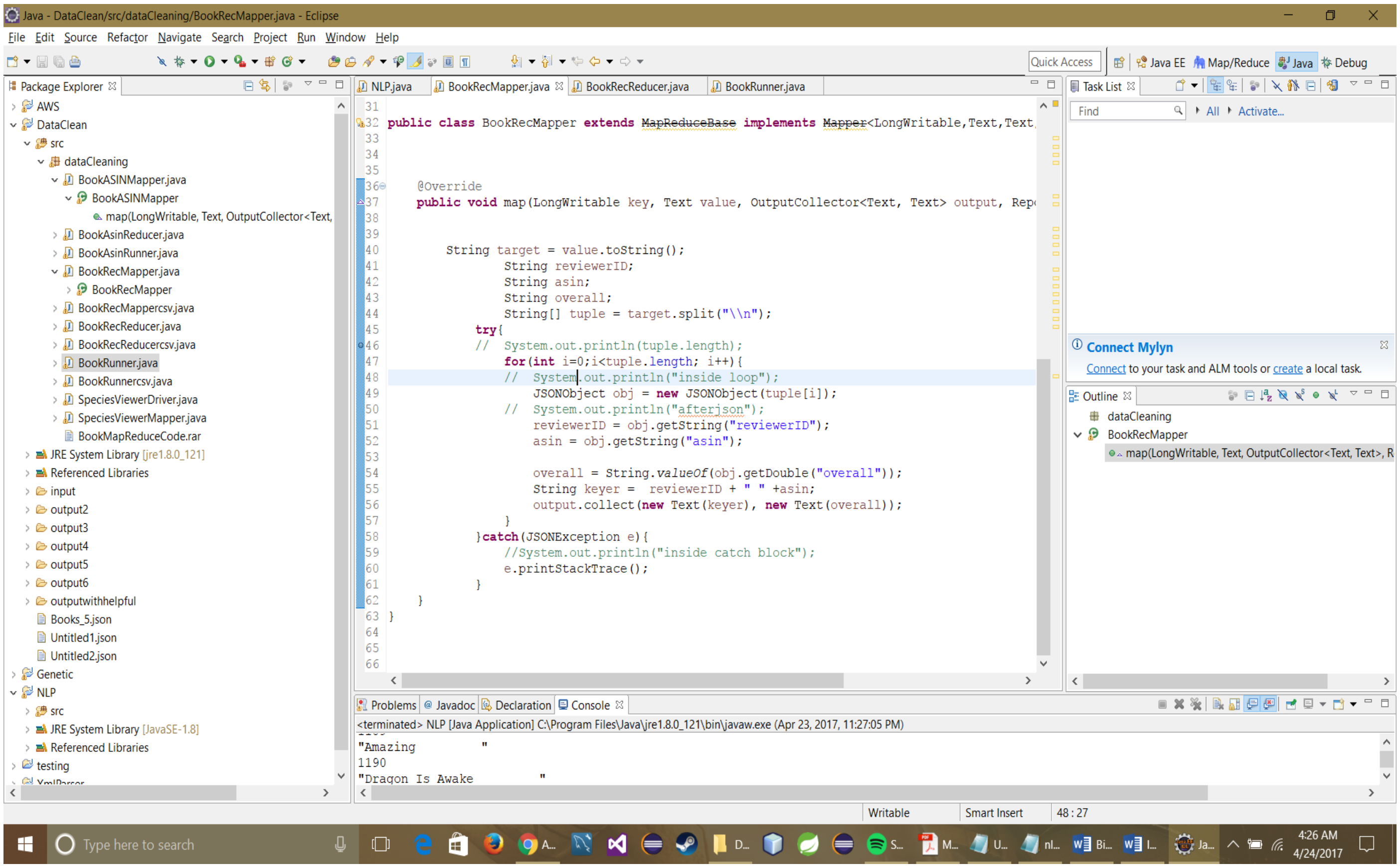
This dataset is a json file containing 9 million reviews and has the following characteristics: -

```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the
piano.  He is having a wonderful time playing these old hymns.
The music  is at times hard to read because we think the book
was published for singing from more than playing from.  Great
purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

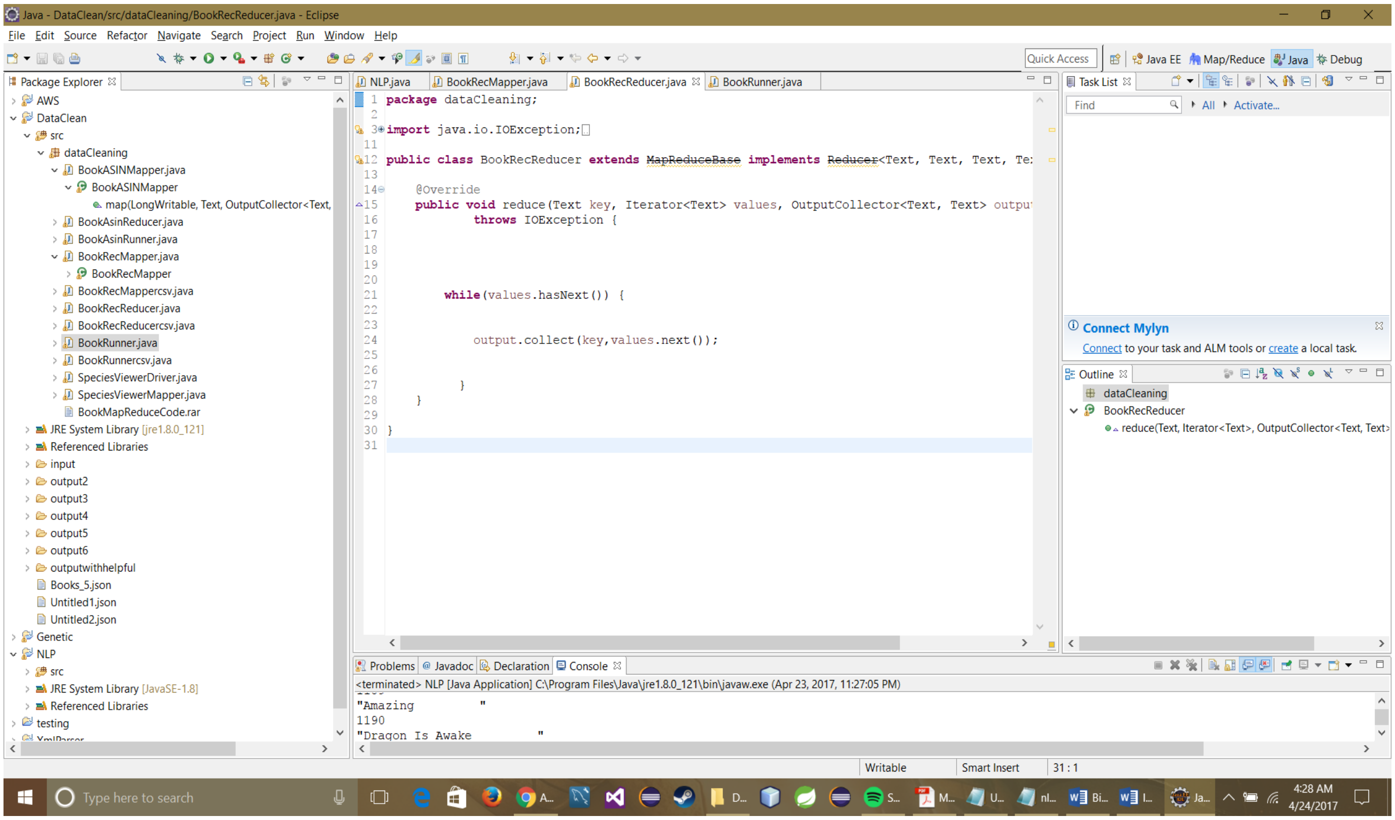
We have used all the above characteristics of the data set for our analysis. There 8898041

Data Cleaning: -

Using the power of Map Reduce we could clean the data. We ran the Map and Reduce algorithms to first convert all the JSON data into rows and columns format. We used the org.json.* library in java to parse the JSON file using Map-Reduce algorithm. We are fetching all the values from the JSON object using the obj.getString().


Java - DataClean/src/dataCleaning/BookRecMapper.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
AWS
DataClean
src
dataCleaning
BookASINMapper.java
BookASINMapper
map(LongWritable, Text, OutputCollector<Text,
BookAsinReducer.java
BookAsinRunner.java
BookRecMapper.java
BookRecMapper
BookRecMappercsv.java
BookRecReducer.java
BookRecReducercsv.java
BookRunner.java
BookRunnercsv.java
SpeciesViewerDriver.java
SpeciesViewerMapper.java
BookMapReduceCode.rar
JRE System Library [jre1.8.0_121]
Referenced Libraries
input
output2
output3
output4
output5
output6
outputwithhelpful
Books_5.json
Untitled1.json
Untitled2.json
Genetic
NLP
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
testing
NLP.java BookRecMapper.java BookRecReducer.java BookRunner.java
public class BookRecMapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,
@Override
public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Rep
String target = value.toString();
String reviewerID;
String asin;
String overall;
String[] tuple = target.split("\\n");
try{
// System.out.println(tuple.length);
for(int i=0;i<tuple.length; i++){
// System.out.println("inside loop");
JSONObject obj = new JSONObject(tuple[i]);
// System.out.println("afterjson");
reviewerID = obj.getString("reviewerID");
asin = obj.getString("asin");
overall = String.valueOf(obj.getDouble("overall"));
String keyer = reviewerID + " " +asin;
output.collect(new Text(keyer), new Text(overall));
}
}catch(JSONException e){
//System.out.println("inside catch block");
e.printStackTrace();
}
}
}
Task List
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
dataCleaning
BookRecMapper
map(LongWritable, Text, OutputCollector<Text, Text>, R
Problems Javadoc Declaration Console
<terminated> NLP [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Apr 23, 2017, 11:27:05 PM)
"Amazing "
1190
"Dragon Is Awake "
Writable Smart Insert 48 : 27
Type here to search
4:26 AM
4/24/2017

BookRecReducer: -

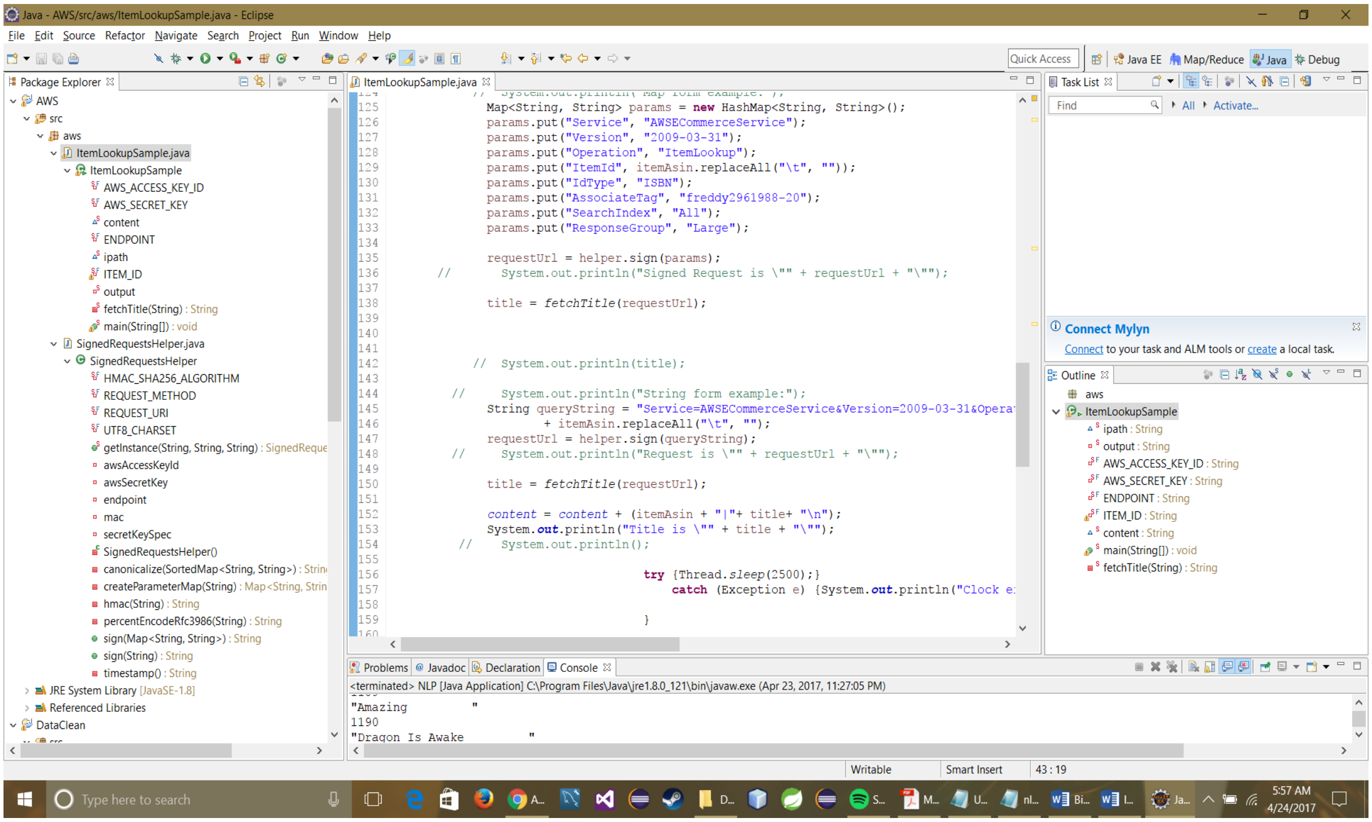


We used a series of mappers and reducers to obtain the final cleaned data. This resulted in the generation of a csv file which contained all the data from the parsed json file.

Fetching Data from the Amazon Product API: -

This was one of our most challenging tasks in the project. This task took a long time as it was not easy to fetch data from the API. The data provided in the Amazon books dataset was not complete as it lacked the book title, book author and the genre of the book. We wrote a program in java which used the unique ASIN ID of the books in our data set and fetched the values from the amazon product API service. We iterated through the entire ASIN data set and fetched all the values.

The main reason as to why the task was challenging is because we could not fetch an API call more than once in a second and running a for loop was doing exactly that. After researching we realized that we had to use the thread function to accomplish the same. The normal for loop was hitting the api more than 13 times a second and hence we were getting error 503.

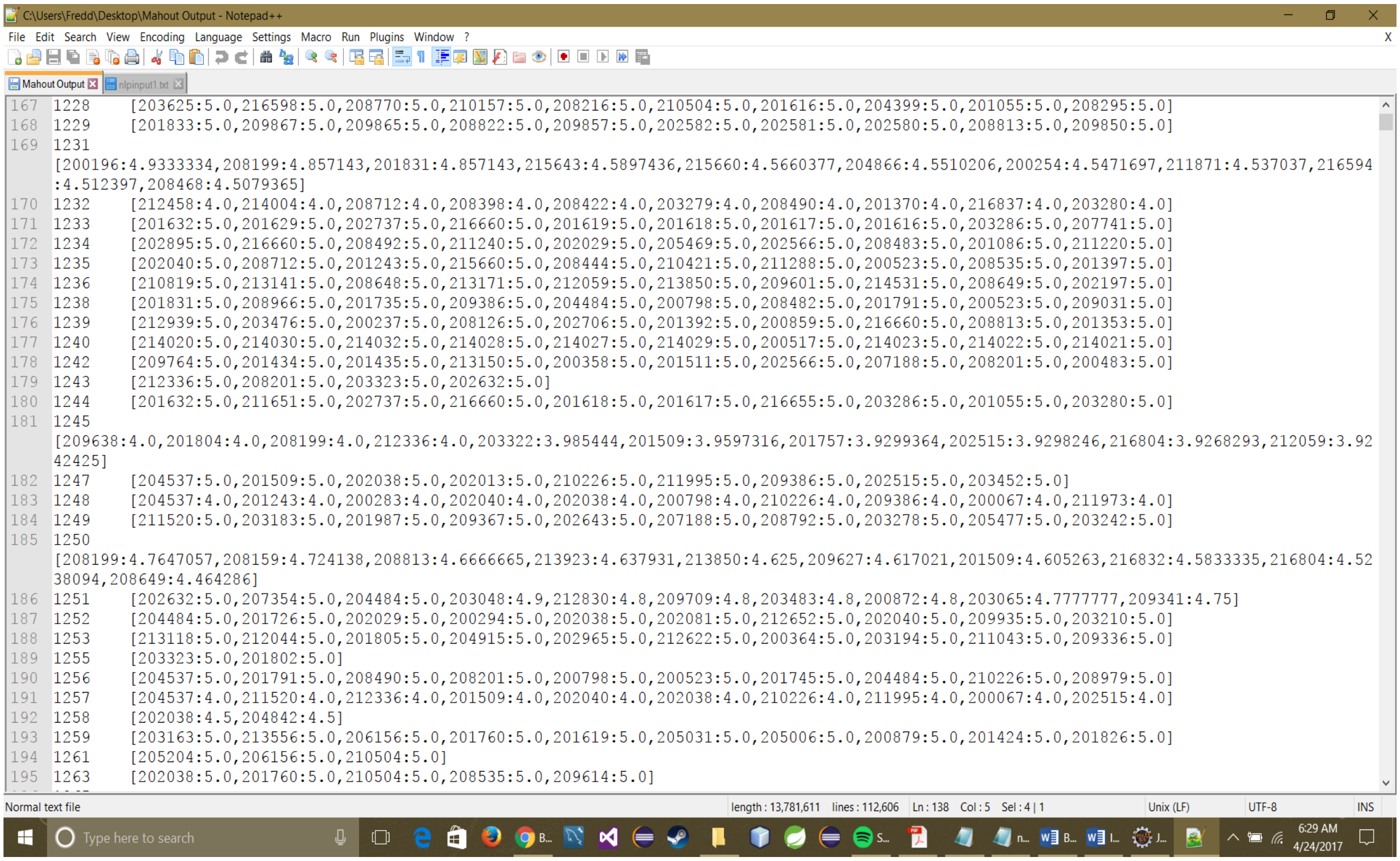


<u>Mahout Recommendation</u>: -

We also derived the input file in the form of Reviewer ID, ASIN ID and Rating to run on Mahout. We derived this by running a Map Reduce implementation on the entire data set. Once we got all the data we ran a Mahout recommendation command: -

bin/Hadoopjarmahout-core-0.7-job.jar org.apache.mahout.cf.taste.hadoop.item.RecommenderJob -s SIMILARITY_COOCCURRENCE --input ./genrerecommender --output ./outputgenrerecommender

This provided us with a recommendation for all the user IDs. We obtained 10 recommendations for each user. The main challenge we faced while running the Mahout Recommendation engine was that the Reviewer ID and the ASIN ID were in the Alphanumeric format and Mahout does not accept alphanumeric values. To do that we had to generate numeric IDs for each alphanumeric ID in the dataset. Below is the output file screenshot for the Mahout job: -



In the above screenshot 1228 is the user ID and 203625 is the book ID recommendation for the user.

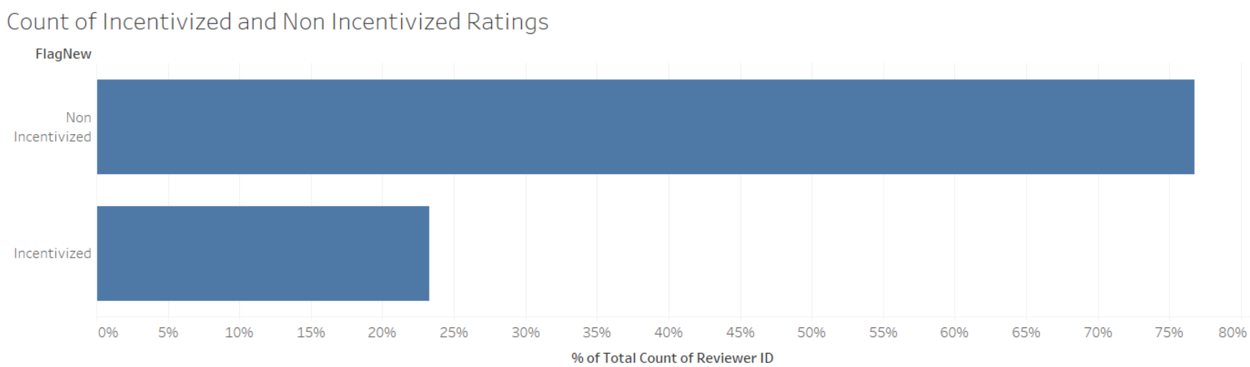
<u>Calculation of Incentivized and Non-Incentivized reviews using Stanford NLP algorithm</u>: -

The goal of this activity is to find out if a review is incentivized or not. A lot of suppliers these days pay their reviewers incentives to give them a good rating and a good review. The way we are doing this is by considering the review column and the summary column in the dataset and assigning a combined score to them. Then we calculate the difference between the actual rating and the calculated rating. If the difference is greater than 2 then the rating is an incentivized rating else the rating is non-incentivized.

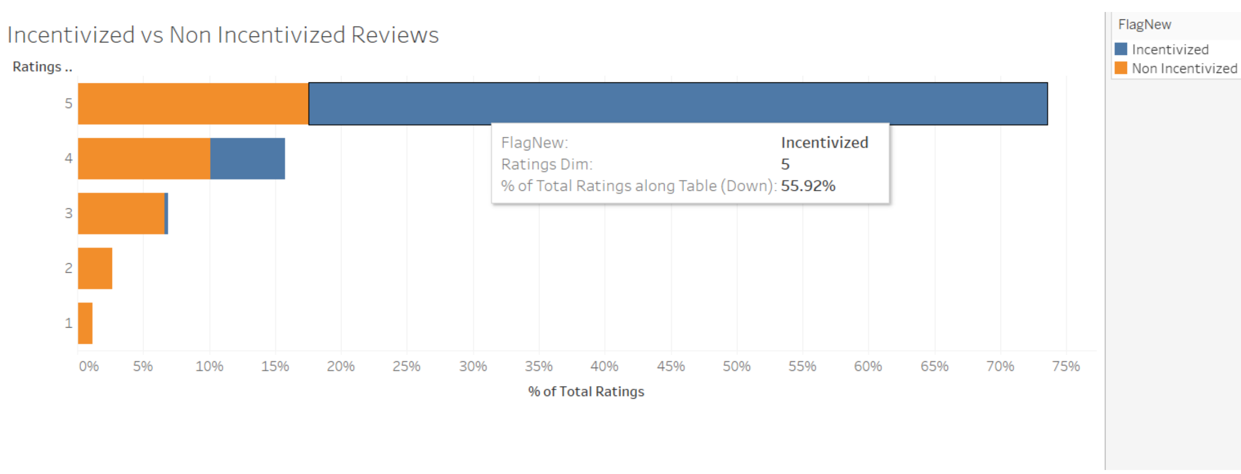
Incentivized Reviews based on Text Analytics:

After performing calculation of Incentivized and Non Incentivized reviews on Stanford NLP algorithm, we have got the data of both kind of reviews. We will perform some interesting visualization on it. Incentivized reviews explains the bias towards the review user shows by writing words like “**Honest**” and “**Unbiased**” and rate those books highly to avail discounts on them.

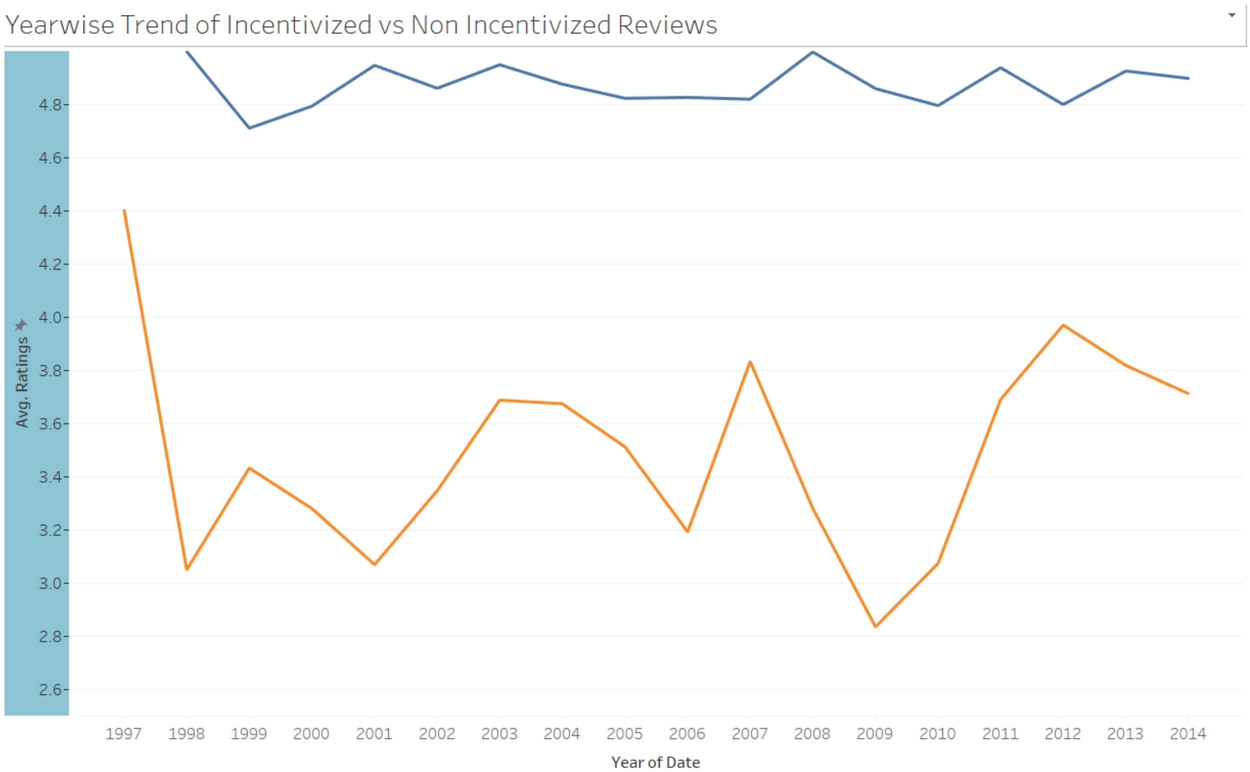
As depicted below, over 25% of the reviews were **incentivized** while rest 75 were non incentivized.



It is clearly observed that most of the non- incentivized reviews pattern cover 55% of the 5-start reviews. i.e. users who tends to give incentivized reviews have given the 5 rating the most as compared to other ratings where percentage of non-incentivized reviews are very less.

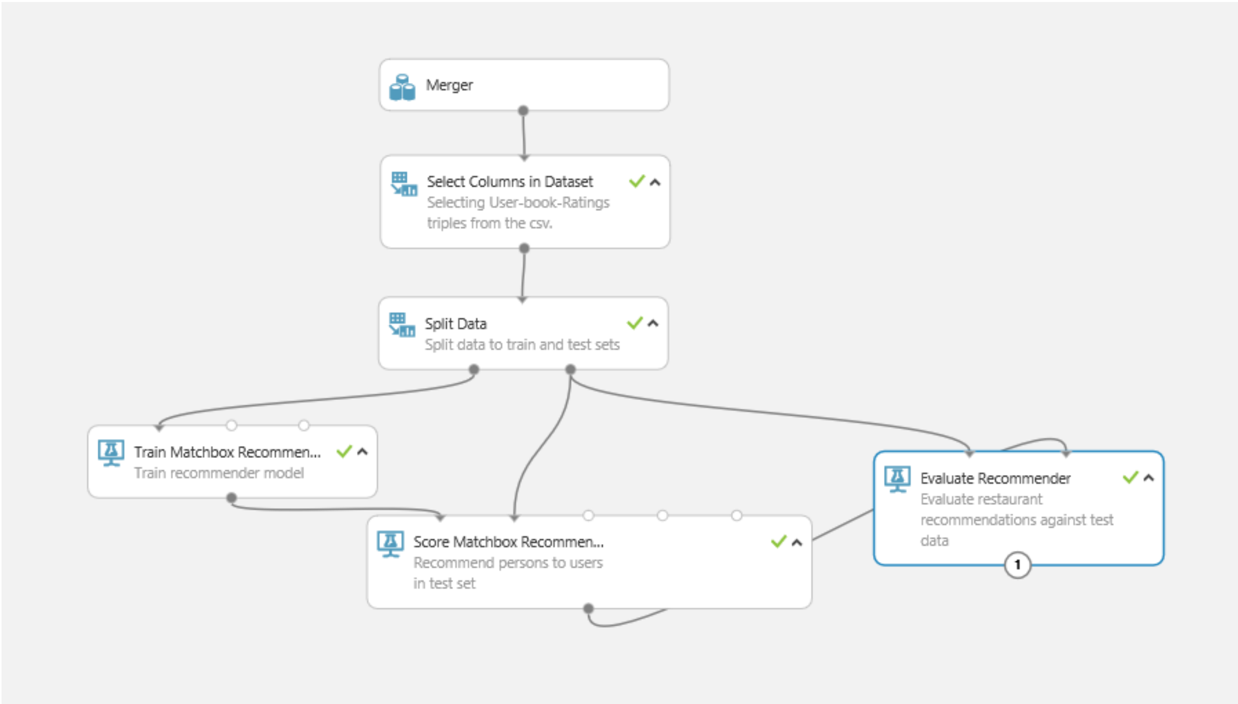


Year wise trend depicts how **average incentivized** reviews range between **4.5 to 5** whereas the average non-incentivized reviews range between **4.5 and 2.8**.



<u>Person Recommendation Using MatchBox Recommendation</u>:

Whenever user visits the homepage, we will be recommending him the related person based on the user's preferences. Those related users will be suggesting the books based on their match with the user. We are using Matchbox Recommendation algorithm that combines collaborative filtering with a content-based approach. It is therefore considered a **hybrid recommender**. It expects user-book-rating triples as input to train the model and outputs the related users based on the ratings provided by them to different books.

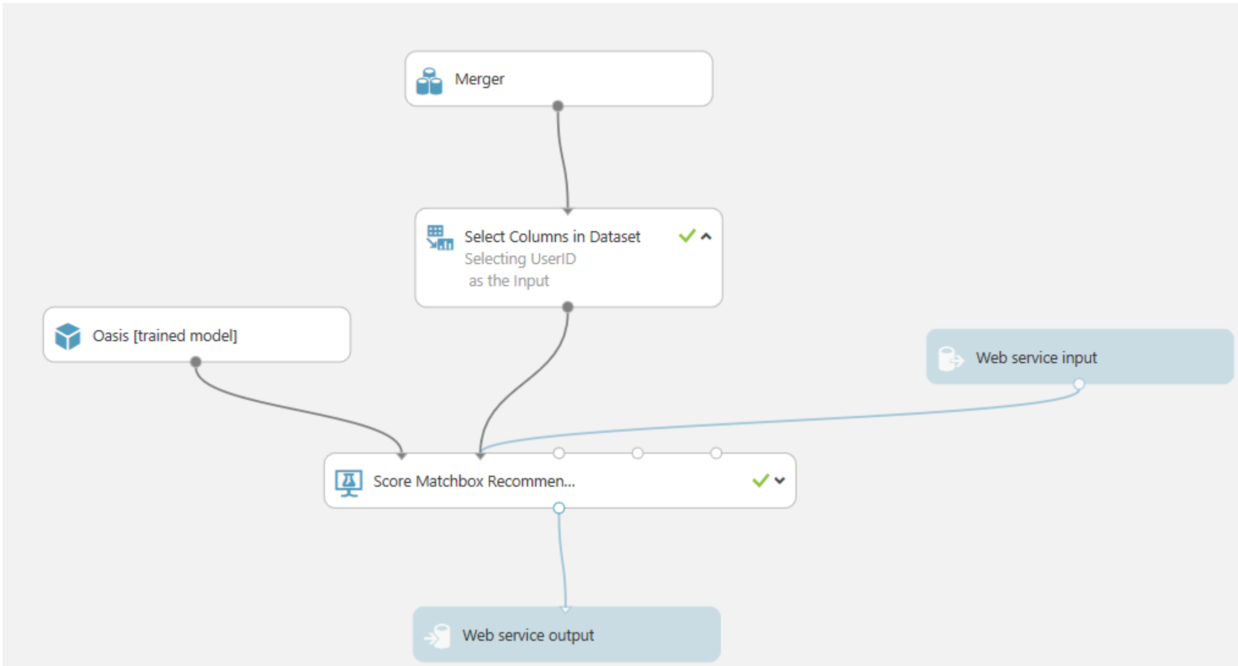


L1 Sim NDCG	L2 Sim NDCG
0.817591	0.803753

Evaluate Recommender computes the average normalized discounted cumulative gain (**NDCG**), based on Manhattan (**L1 Sim NDCG**) and Euclidean (**L2 Sim NDCG**) distances, and returns both values in the output dataset. In our case, both the values are 0.81 and 0.80 that shows a good performance by the model.

Deploying the Web Service using the Trained Recommended Model:

Once we have trained the model, we will deploy it using the web service.



Output (Related Users up to 5): Below is the output of the user ID with the recommended users based on the book rating.

User	Related User 1	Related User 2	Related User 3	Related User 4	Related User 5
A11TYILTAFKPR3	A2INDDW3XYFFV1	A1JAG4YEOMXCKE	A370Z6I5GBWU44	AMKZHBOK7VMQR	A1K1JW1C5CUSUZ
A12W8NRSYR593I	A15ACUAJEJXCS3	AMRZ5G7HF7I03			
A133S8CUIVRFKN	A59LBV682DWGM	A188JKV8QVVOT7	AMALQ15DP0YDJ	A3ASXNNWLR4J4I	AI9AFMD6OVGUG
A1340OFLZBW5NG	AHD101501WCN1	ABFOAYZA2UHD3	A308E1C7MU3462		
A157UENZPTI1TD	A1DYXCF4148PJT	A2GPEV42IO41CI			
A15ACUAJEJXCS3	AJQ1S39GZBKUG	A38AAPXSJN4C5G	A2SHQJP6PNQTLD	A2W6WXEUAVM3E	A12W8NRSYR593I
A1BC4GPH9LBSNR	A2QZQBINBG6B5N	A3R8PXSFGY9MC2	AS5ERWDSXRDNX	AZXGPM8EKSHE9	A34UTL4AVX80MK
A1BM81XB4QHOA3	A1K1JW1C5CUSUZ	A3W43PSHRIG8KV	A1NPNGWBVD9AK3	AHD101501WCN1	A2NHD7LUXVGTD3
A1C9NH907F3RHE	AK1C761G3YD0J	A545U4UQEPT1J	A3H9YD6K9TVKDP	A281NPSIMI1C2R	A1K1JW1C5CUSUZ
A1DW9QEARBGRFO	A281NPSIMI1C2R	A2PH70X2FVDDGH	A3MFKQXPA7PERW	A3UJJGY799F76I	AZTK9T1ECFSX1
A1DYXCF4148PJT	A34K6MOV1NC10E	A1IU7S4HCK1XK0	A1S1O3YT8NS68A	A222LQEPE7O7BV	A2NNPISXOKJ1K4

Below are the list of suggested users if we enter a user id.

<u>Machine Learning Algorithms</u>: -

Using K-Means clustering to predict the helpfulness of a review:

We have analyzed the structure of text the book reviews provided by the amazon users and predicted the how the reviews posted by the users effect the helpfulness of the product.

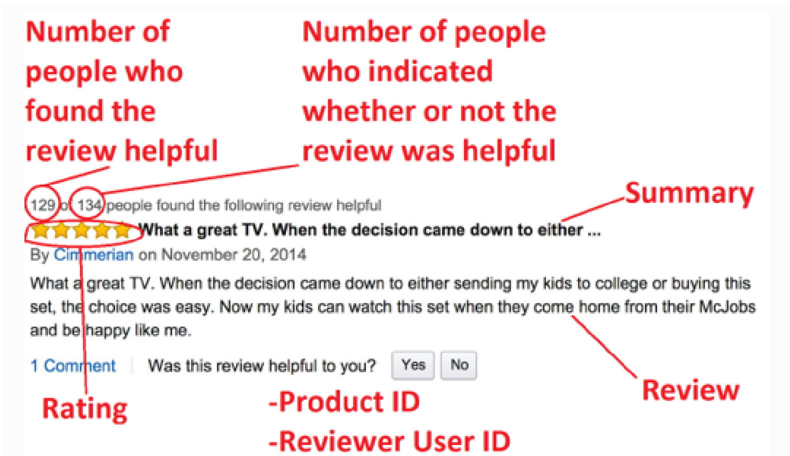
Language and Services used: **Python and Apache Spark 1.6**

Terms Used:

Helpfulness Numerator: Number of people who found the review helpful.

Helpfulness Denominator: Number of people who indicated whether review was helpful or not.

Helpfulness = 1 if Helpfulness Numerator/ Helpfulness Denominator >0.5 else 0



Dataset:

Exploratory Analysis:

We have done feature selection and will focus on review related data and the related columns as required.

	HelpfulNumerator	HelpfulDenomerator	Review_Text	Ratings
0	0	0	Spiritually and mentally inspiring! A book tha...	5.0
1	4	4	When Gibran was first introduced to me I had ...	5.0
2	0	0	As you read Gibran's poetry brings spiritual ...	5.0
3	0	0	Jubran Kahlil Jubran had a teacher make a mist...	5.0
4	0	0	When I first started writing poetry at age 12 ...	5.0

```
#include reviews that have more than 15 helpfulness data point only
df1 = df1[(df1.HelpfulDenomerator > 15)]
```

```
#converting the book reviews to lower case
df1.loc[:, 'Review_Text'] = df1['Review_Text'].str.lower()
df1["Review_Text"].head(10)
```

```
9       this man was a son of a pastor  but worshipped...
35      this is one of the first (literary) books i re...
40      certainly the words are of kahlil gibran  but ...
42      this book was given to me as a gift before i j...
49      gibran gets right down to the bedrock of what ...
59      the prophet is almustafa  called ""the chosen ...
```

Creating the Helpfulness Column(Derived):

```
#transform Helpfulness into a binary variable with 0.50 ratio
df1.loc[:, 'Helpfulness'] = np.where(df1.loc[:, 'HelpfulNumerator'] / df1.loc[:, 'HelpfulDenomerator'] > 0.50, 1, 0)
df1.head(3)
```

	HelpfulNumerator	HelpfulDenomerator	Review_Text	Ratings	Helpfulness
9	0	27	This man was a son of a pastor but worshipped...	1.0	0
35	81	92	This is one of the first (literary) books I re...	5.0	1
42	19	25	This book was given to me as a gift before I j...	5.0	1

Summary Measures and Correlation Matrix:

```
df1.groupby('Helpfulness').count()
```

	HelpfulNumerator	HelpfulDenomerator	Review_Text	Ratings
Helpfulness				
0	10926	10926	10926	10926
1	35981	35981	35981	35981

```
df1.corr()
```

	HelpfulNumerator	HelpfulDenomerator	Ratings	Helpfulness
HelpfulNumerator	1.000000	0.968204	0.085496	0.155268
HelpfulDenomerator	0.968204	1.000000	-0.009486	0.031488
Ratings	0.085496	-0.009486	1.000000	0.449498
Helpfulness	0.155268	0.031488	0.449498	1.000000

Bag of Words Model: Using Bag of Words model, we got less accuracy, so we went for Logistic Regression classification modeling.

```
# ROC/AUC score
#Accuracy of bag of Words Model is 75 Percent
y_score = probas
test2 = np.array(list(df2.Helpfulness))
y_true = test2
roc_auc_score(y_true, y_score[:,1].T)
```

```
0.75004537957503881
```

Used K-Means Clustering and created 5 clusters for the review words. Selecting top 15 words.

```
['quot',
 'new',
 'characters',
 'reading',
 'like',
 'good',
 'books',
 'character',
 'love',
 'life',
 'history',
 'time',
 'work',
 'really',
 'don',
 'american',
 'read',
```

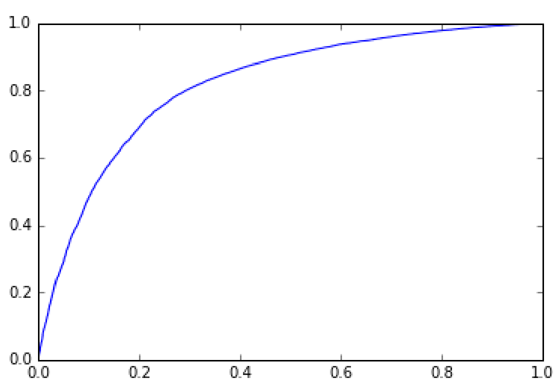
Cluster wise Summary:

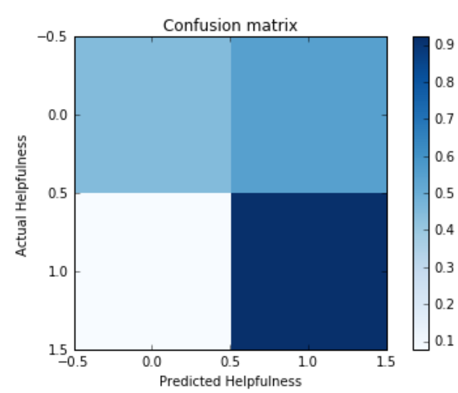
ROC Curve, Confusion Matrix and accuracy using Logistic Regression:

```
roc_auc_score(y_true, y_score[:,1].T)
```

```
0.817285742002576O9
```

```
[[ 4898  6028]
 [ 2808 33173]]
```





Result: Below are the top words that effects helpfulness of a review. Plot, Story , Novel, Character etc words with high coefficients shows that they are contributors to a helpful review posted by a user. Read, Read book and american are negatively correlated but we need to remove more common words to remove. We can use these words to suggest user what all words can led to a positive and a negative review This can be a suggestion or a tip for a healthy and good review. Negative reviews will be rejected.

```
#Displayin the top parameters
words.extend(['score'])
sorted(zip(words,gs.best_estimator_.coef_[0]),key=lambda x:x[1])
```

```
[('read book', -1.4517062337490938),
 ('read', -0.65564660289473486),
 ('american', -0.33220442695913532),
 ('good', 0.097733434820765394),
 ('don', 0.099658412191117013),
 ('people', 0.1688742692035092),
 ('war', 0.24795276487231696),
 ('reading', 0.29325034925531579),
 ('world', 0.40865494487118792),
 ('history', 0.56390817369734958),
 ('score', 0.73301317655666598),
 ('love', 0.74225660023473528),
 ('like', 0.75846148131849778),
 ('just', 0.76507499906274401),
 ('books', 0.92842676539356794),
 ('time', 1.1204680692599365),
 ('really', 1.1350328179437257),
 ('quot', 1.1627607462714684),
 ('ve', 1.299041736152452),
 ('author', 1.4240716916070468),
 ('new', 1.5545441216921103),
 ('characters', 1.5691960894924462),
 ('plot', 1.7299017179279561),
 ('story', 1.7573310939537168),
 ('book', 1.8228550220322288),
 ('novel', 1.8706610916276731),
 ('character', 1.9134522122828601),
 ('work', 2.0474878313886276),
 ('life', 2.3307098581062005)]
```

Topic Modeling using LDA (Latent Dirichlet Allocation) in R:

Latent Dirichlet allocation (LDA) is a topic model that generates topics based on word frequency from a set of documents. LDA is particularly useful for finding reasonably accurate mixtures of topics within a given document set

Packages Used: nltk, numpy, genism, csv.

We have divided the book reviews text posted by users into different Topics that will consist of number of Documents. These topics will describe the structure of the document and contains the most probable documents in it.

Implementation:

1) **Cleaning Data:**

   We will remove the stop words, punctuations and convert the sentences into lower case and append it into a list “texts”
   We will use the **tokenizer** to convert texts into tokens and **stemming** to group(stem) similar words.

```
# clean and tokenize document string
tokens = tokenizer.tokenize(raw_lower)

# remove stop words from tokens
stopped_tokens = [i for i in tokens if not i in en_stop]

# stem tokens
stemmed_tokens = [p_stemmer.stem(i) for i in stopped_tokens]

# add tokens to list
texts.append(stemmed_tokens)
```

2) Implementing the Model:

We will convert the text into a dictionary using doc2bow() . We will build the LDA model using the genism default LdaModel method. We will initially give the **number of topics** as 5. We can change it according to the number of words and depending upon the result.

```
# # convert tokenized documents into a document-term using doc2bow that will produce documents.
corpus = [dictionary.doc2bow(text) for text in texts]

#generate LDA model
lda = gensim.models.ldamodel.LdaModel(corpus, num_topics=5, id2word = dictionary, passes=20)
```

3) Analyze the Result:

As we can analyze below is a topic wise division of a specific book. We can see in each topic there are different documents with their probable score. More the probability, more likely is the correlation of the word in the topic.

In topic 1, all the words **"wife", "husband", "girl", "marriage**" words depict the script od the novel The Gone girl with a high probable score. If a user is seeing the novel for the first time , he can get an idea of the kind of reviews and an overview of the book.

```
#Topic Wise words with probability..
for i in  lda.show_topics():
    print(i[0], i[1])
```

```
0 0.047*"ami" + 0.041*"nick" + 0.034*"s" + 0.010*"wife" + 0.008*"stori" + 0.008*"husband" + 0.007*"gone" + 0.007*"dis
appear" + 0.007*"girl" + 0.007*"marriag"
1 0.053*"book" + 0.041*"read" + 0.022*"twist" + 0.018*"stori" + 0.017*"end" + 0.016*"charact" + 0.015*"turn" + 0.013
*"love" + 0.012*"great" + 0.012*"plot"
2 0.053*"book" + 0.041*"t" + 0.033*"end" + 0.029*"read" + 0.018*"like" + 0.015*"just" + 0.012*"realli" + 0.010*"didn"
+ 0.010*"charact" + 0.010*"stori"
3 0.032*"charact" + 0.011*"person" + 0.010*"stori" + 0.010*"plot" + 0.010*"peopl" + 0.010*"main" + 0.008*"two" + 0.00
8*"narrat" + 0.006*"like" + 0.006*"well"
4 0.025*"s" + 0.017*"flynn" + 0.016*"girl" + 0.015*"gone" + 0.011*"novel" + 0.009*"charact" + 0.008*"one" + 0.008*"ca
n" + 0.006*"marriag" + 0.006*"gillian"
```

Trying with different parameters

```
#Trying with different topic value..
print(lda.print_topics(num_topics=10, num_words=10))
```

```
[(0, '0.073*"book" + 0.054*"read" + 0.045*"t" + 0.024*"twist" + 0.024*"put" + 0.016*"turn" + 0.016*"love" + 0.015*"gr
eat" + 0.015*"end" + 0.015*"couldn"'), (1, '0.018*"movi" + 0.015*"war" + 0.012*"rose" + 0.011*"ben" + 0.010*"affleck"
+ 0.005*"cover" + 0.005*"scott" + 0.005*"1st" + 0.004*"recogn" + 0.004*"21st"'), (2, '0.063*"t" + 0.034*"end" + 0.028
*"book" + 0.025*"like" + 0.020*"charact" + 0.019*"didn" + 0.017*"don" + 0.017*"just" + 0.013*"star" + 0.012*"s"'),
(3, '0.066*"book" + 0.038*"read" + 0.032*"end" + 0.017*"like" + 0.013*"just" + 0.012*"good" + 0.010*"get" + 0.010*"ti
me" + 0.010*"author" + 0.009*"realli"'), (4, '0.019*"s" + 0.018*"charact" + 0.016*"stori" + 0.011*"one" + 0.010*"nove
l" + 0.010*"book" + 0.009*"plot" + 0.009*"two" + 0.009*"gone" + 0.009*"wife"'), (5, '0.010*"sister" + 0.008*"twin" +
0.007*"margo" + 0.007*"bar" + 0.007*"call" + 0.007*"front" + 0.006*"cat" + 0.005*"town" + 0.005*"york" + 0.005*"la
w"'), (6, '0.063*"ami" + 0.055*"nick" + 0.038*"s" + 0.008*"wife" + 0.007*"disappear" + 0.007*"stori" + 0.006*"anniver
sari" + 0.006*"marriag" + 0.006*"dunn" + 0.006*"husband"'), (7, '0.012*"don8217t" + 0.011*"it8217" + 0.009*"8211" +
0.007*"can8217t" + 0.007*"i8217m" + 0.006*"amy8217" + 0.006*"extent" + 0.006*"didn8217t" + 0.005*"nick8217" + 0.004
*"8221"'), (8, '0.031*"s" + 0.027*"girl" + 0.024*"gone" + 0.020*"flynn" + 0.011*"can" + 0.011*"gillian" + 0.011*"t" +
0.009*"know" + 0.008*"one" + 0.008*"read"'), (9, '0.030*"book" + 0.026*"charact" + 0.025*"read" + 0.023*"end" + 0.023
*"flynn" + 0.021*"twist" + 0.018*"stori" + 0.018*"well" + 0.015*"plot" + 0.015*"turn"')]
```

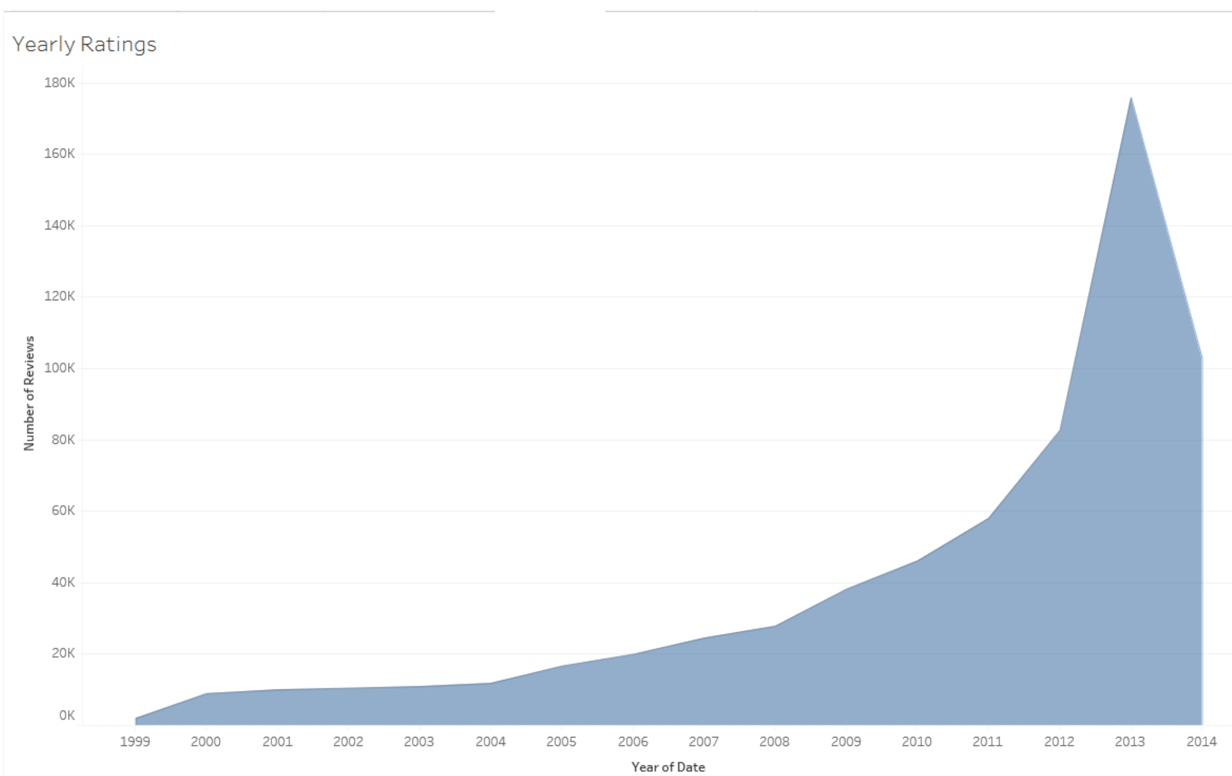
Application and Future Scope:

Whenever user selects a book, we can display the **topic** wise top **documents** so that he would not have to go through all the reviews and check to get the idea of the type of book. He can get an idea in nutshell of the book and it will save time and customer experience.

Exploratory Analysis (Using Tableau):

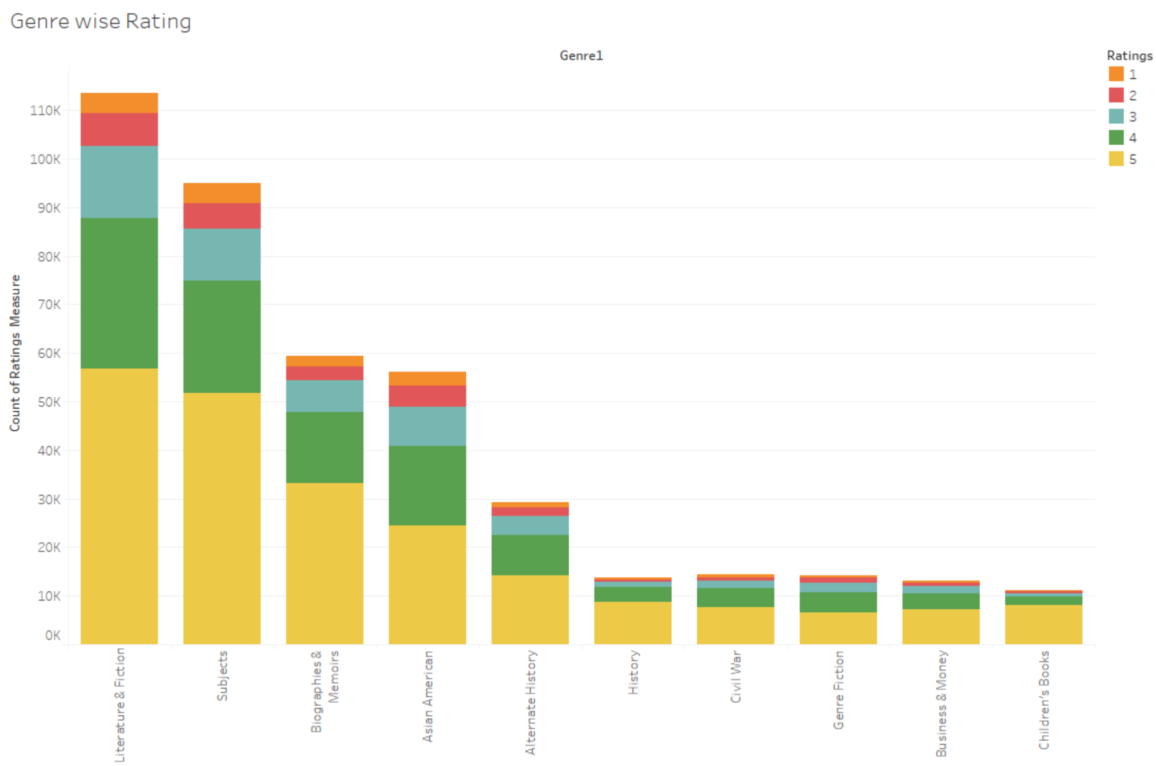
Reviews from users who have used the product in question can give us more context to the product. Each reviewer rates the product from 1 to 5 stars, and provides a text summary of their experiences and opinions about the product. The ratings for each product are averaged together to get an overall product rating.

As depicted below, number of reviewers keeps increasing year by year using the amazon book store.



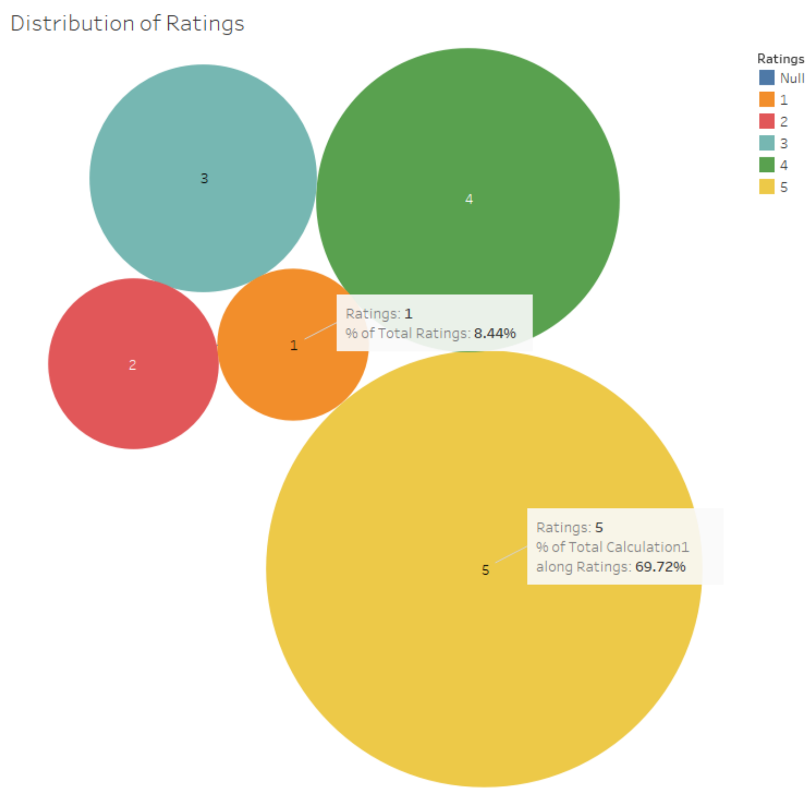
1) **Genre based Ratings Distribution**

Given stacked bar chart shows the distribution of ratings over the genre. Literature leads the race with around 55k 5-star ratings followd by Biographies and memories. This visualization proves that the top categories have a significantly higher percentage of 4/5-star ratings than the bottom categories, and a much a lower proportion of 1/2/3-star ratings. The inverse holds true for the bottom categories.

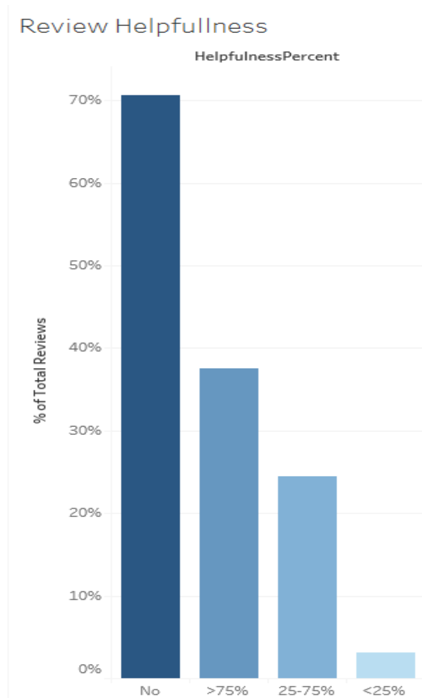


2) **Distribution of Ratings:**

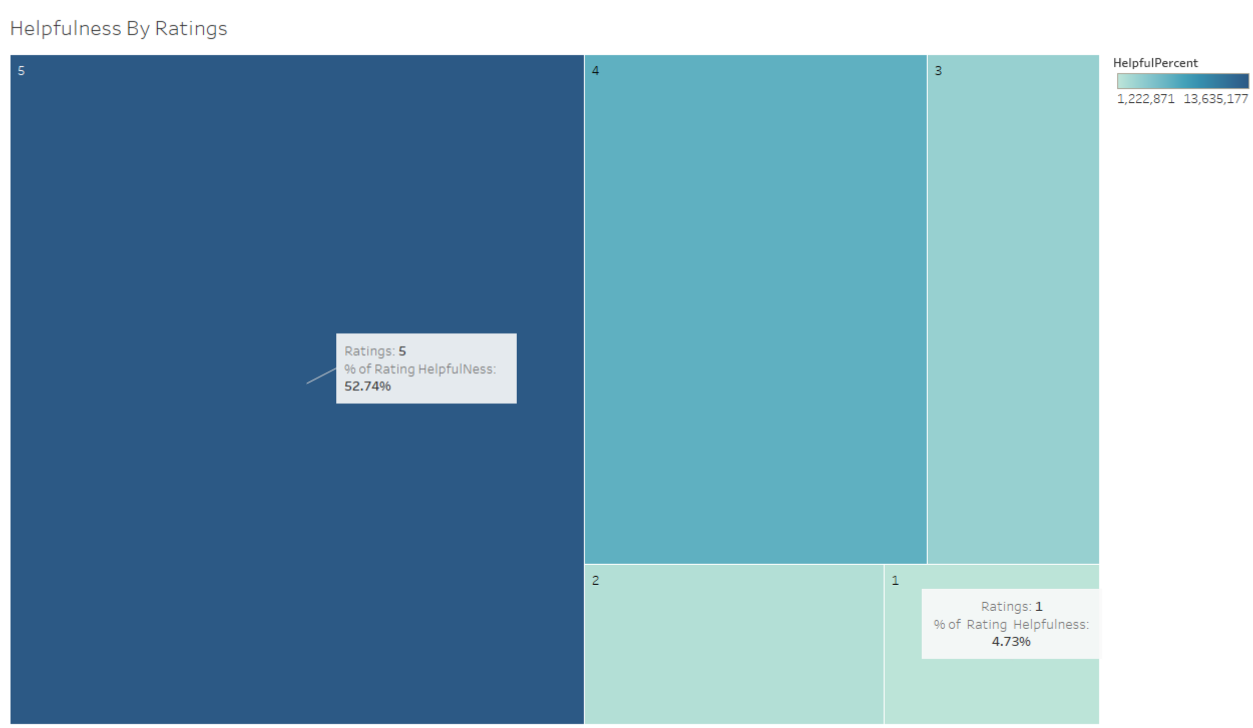
We can see the distribution of the ratings over the bookstore reviews writter. 5-star rating composes the major portion i.e. 69.72% and 1-star lowest with 8.44 %. Rest of the ratings follow with avearge percentages.

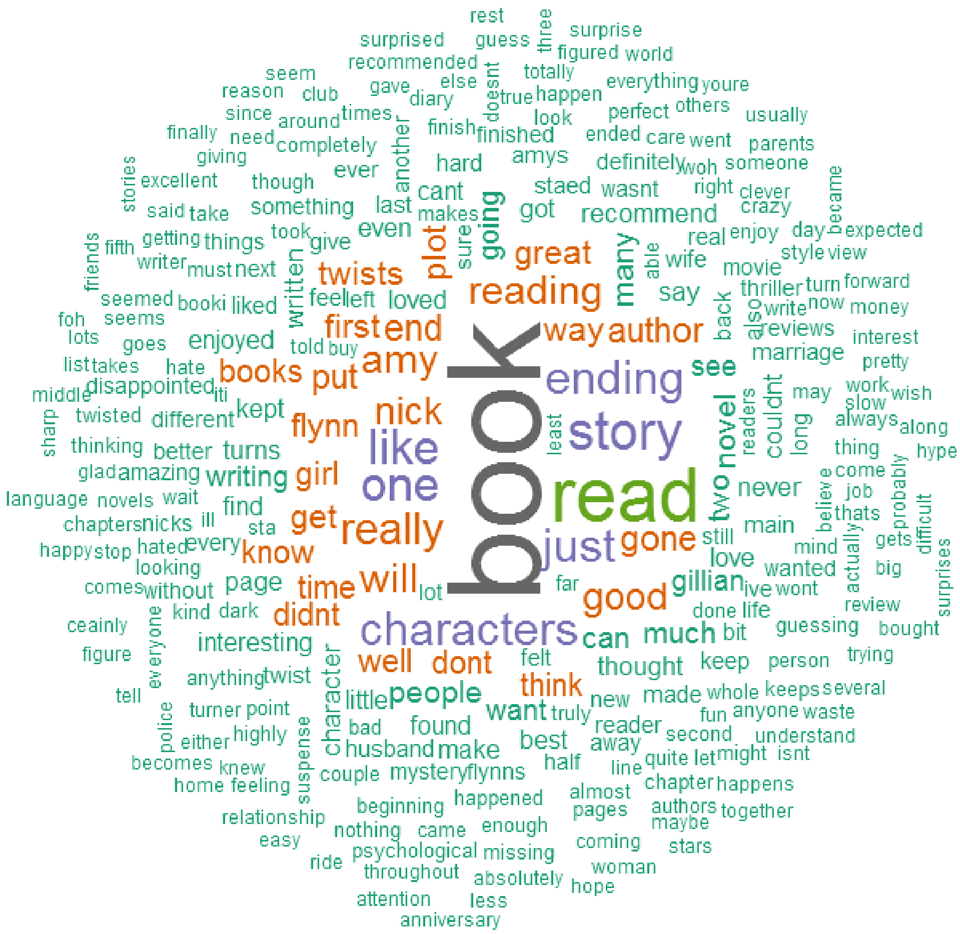


3) **Distribution of Ratings:**
Around 70% reviews are not voted. We have created a derived column Helpfulness by dividing Helpful Numerator by total reviewed ratings. As we can see, around 40 percent reviews are >75 % helpful and the trend goes on as the helpfulness decreases.



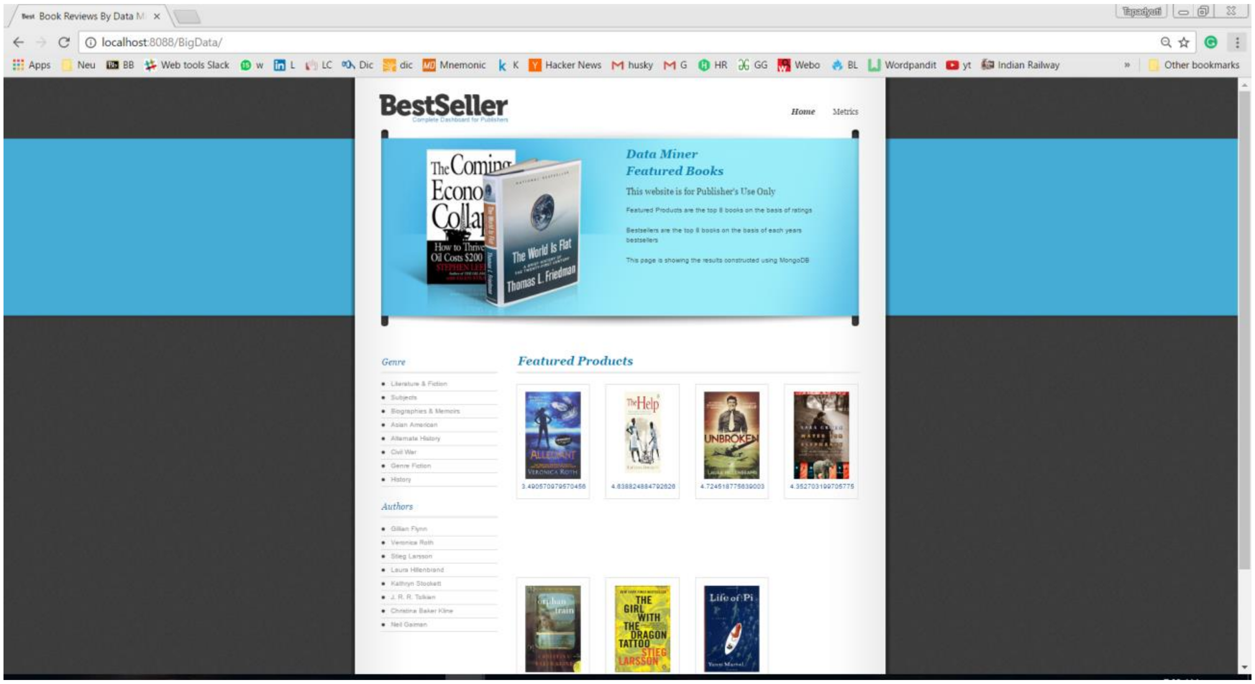
4) **Helpfulness by Ratings :**
As expected, 5 rating reviews were most helpful with 52 percent and the 1 rating reviews were least 4.73 % helpful





Word Cloud for the book Gone Girl.

Web Application/MongoDB connectivity :



Priliminary Step :

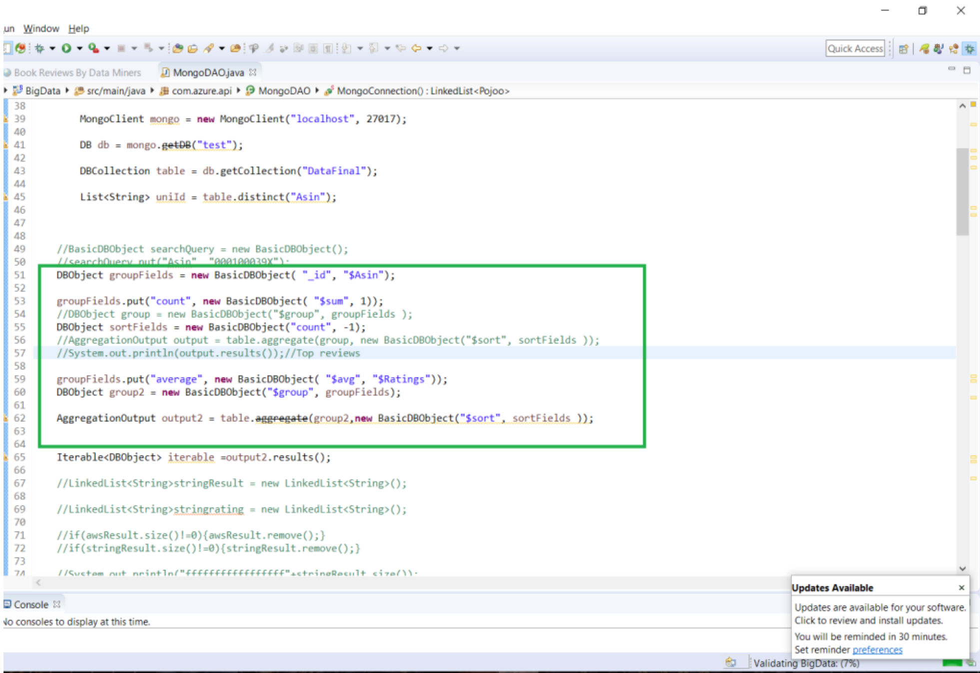
Importing Data from json format to MongoDB 3.4
Following Command has been used:

Test- default Database Schema present in MongoDB

./mongoimport -d **test** -c **DataFinal** -f Asin,ReviewerID,Genre1,HelpfulNumerator,HelpfulDenomerator,Review_Text,Review_Summary,ReviewerName,Date,Ratings,Title,Genre,Author,GenreID,UniqueReviewerID,UniqueASINID,YEAR --file F:\MyFinalData.json

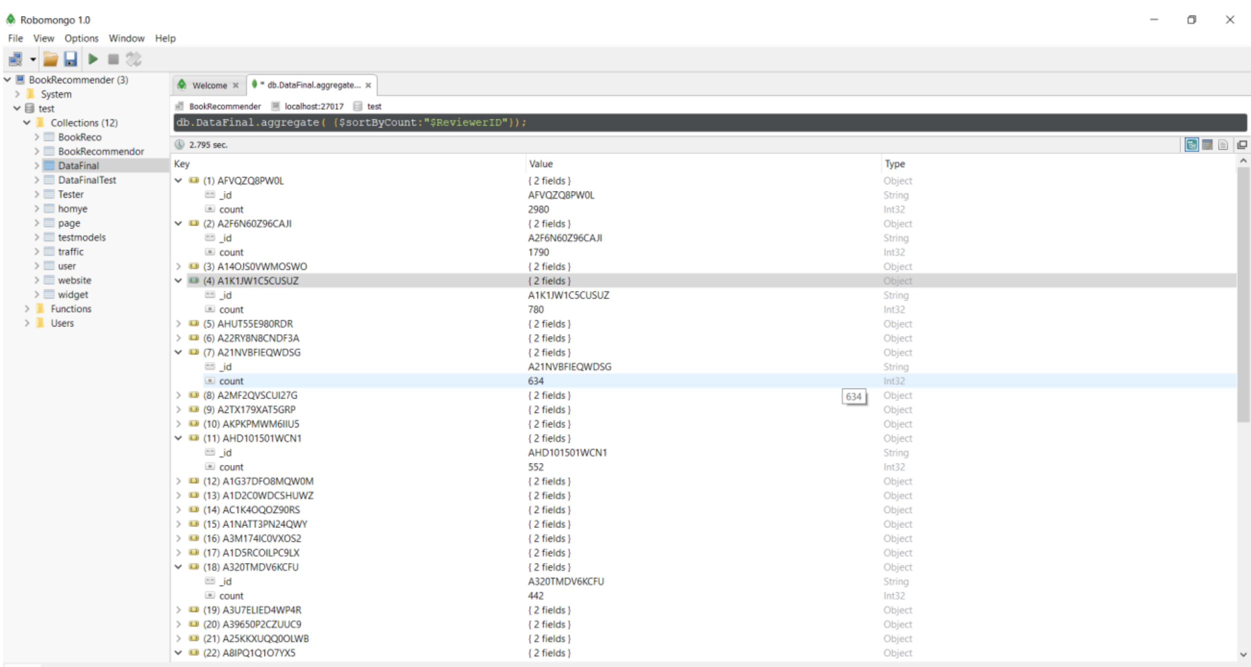
MongoDb Basic Queries:

Steps taken :

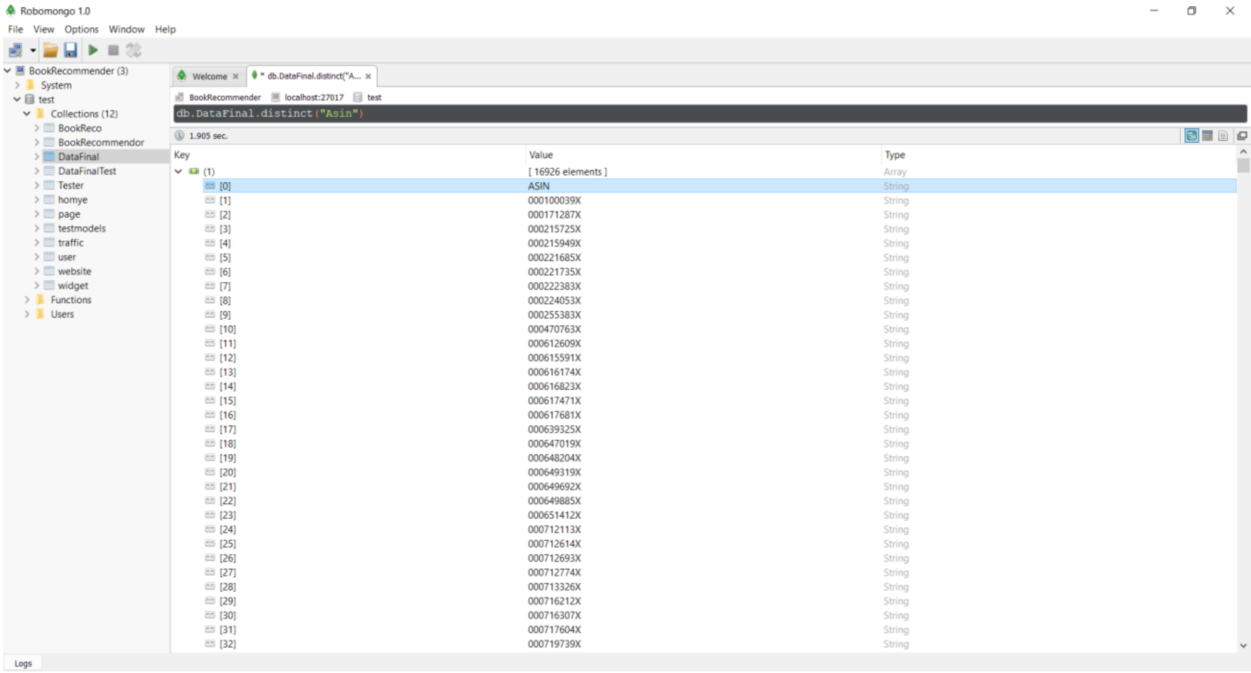


1.
db.DataFinal.aggregate({$sortByCount:"$ReviewerID"});

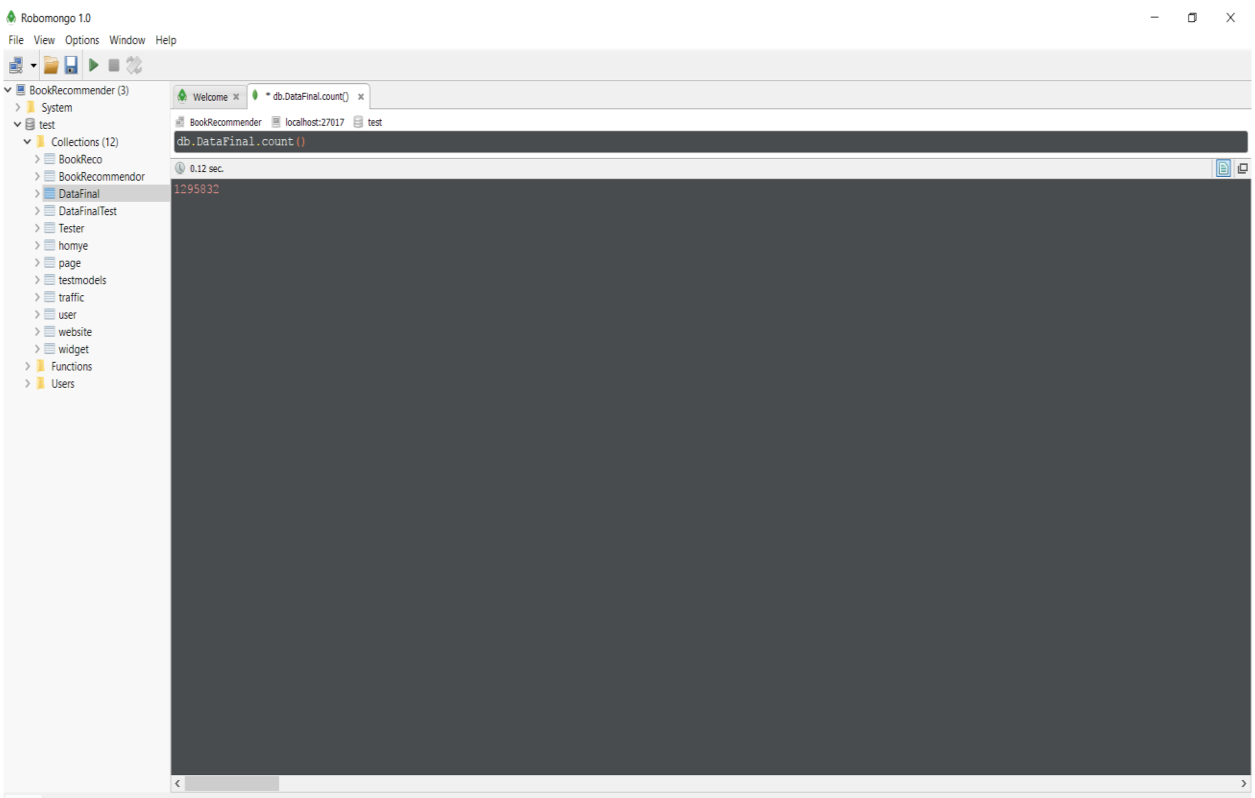
Finding AsinId , sorted by numberof Reviews.



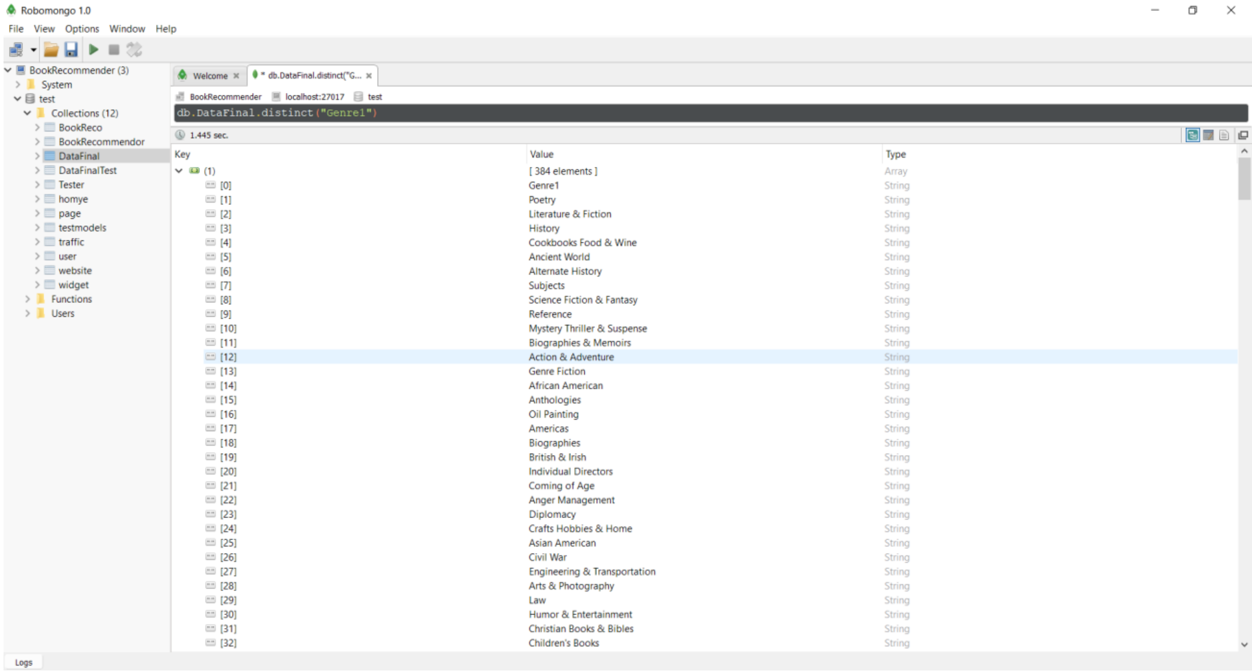
2. **Finding all distinct Asin Id in the document**.



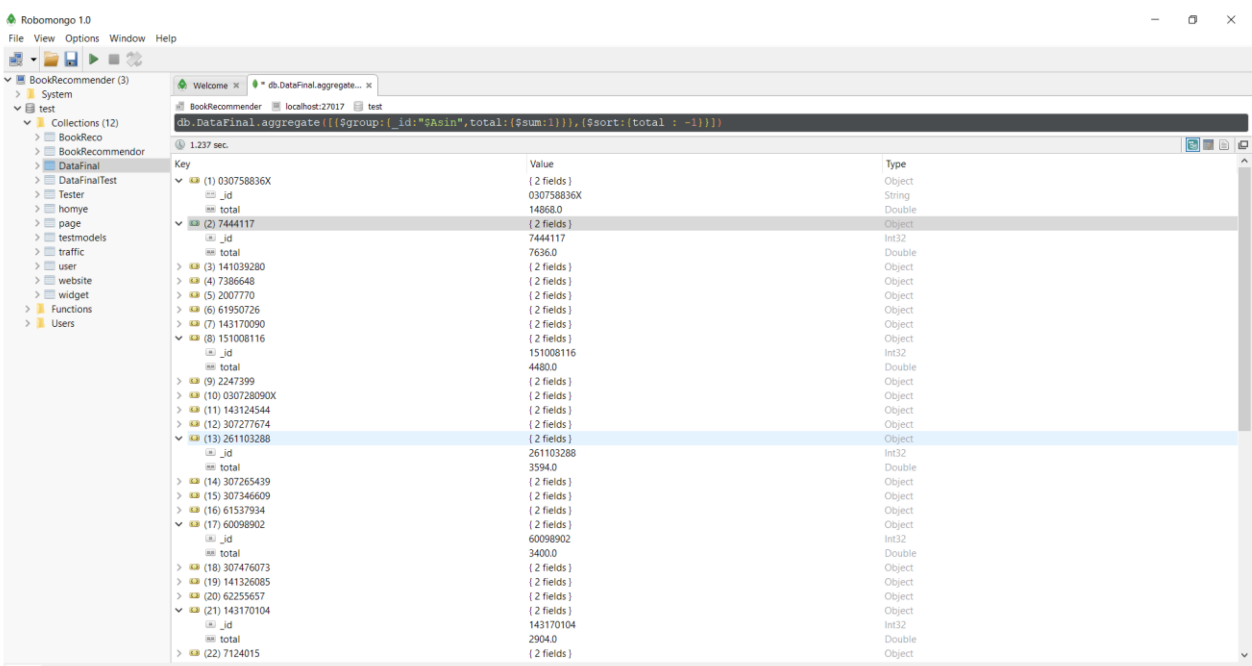
3.**Finding Count of the Document**:



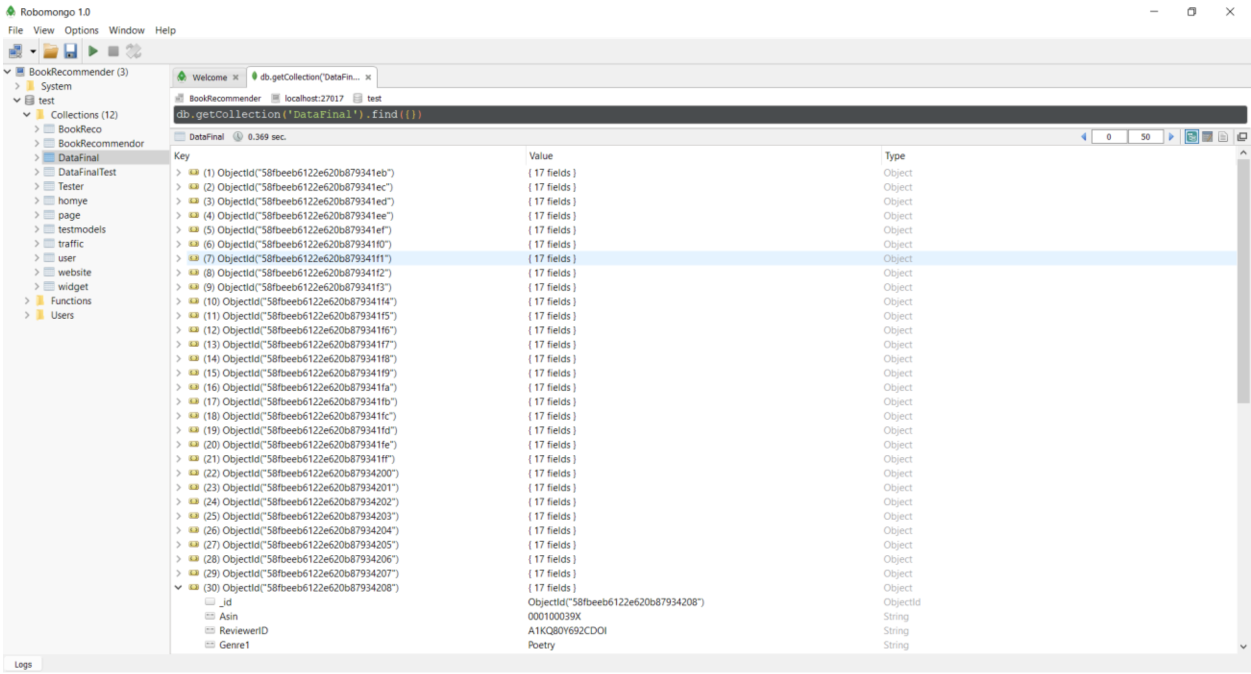
4.Getting all the Distinct Genres



5. Number of Review for each books in Ascending order.



6.Showing all Documents:



7.Changing String format to Int for Rating field.

```
db.DataFinal.find({Ratings: {$exists: true}}).forEach(function(obj) {
   obj.Ratings = new NumberInt(obj.Ratings);
   db.DataFinal.save(obj);
});
```

Contribution by Each Individual: -

As a team, we have worked effectively on all areas of the project in a collaborative manner and contributed equally to each part. As some of the functionalities are dependent to each other we must schedule and allot the tasks sequentially.

Fredy Daruwala: - He is responsible for the data cleaning part of the application using the Map and Reduce operation. He is also responsible for most of the storage and management of the data. He is also responsible for handling the Mahout Recommendation engine and to recommend books to the users.

Gautam Pawar: - He is responsible for running all the machine learning algorithms on the data set. He also is responsible for cleaning the data and running the machine learning
G algorithms on the data.

Nikita Anand: - She is responsible for all the Tableau visualizations, data cleaning activity using R and Python. She is also responsible for building the Web application for performing different analysis on and predict future and study past trends.

Tapadyuti Maiti: - He is responsible for building the web application and running the NLP analysis on reviews data so that we can obtain incentivized and non-incentivized reviews analysis.

Issues and Their Resolution: -

The main issues we faced in the project were the following: -

- Data Cleaning activity which took a long time as there were a lot of missing values on the json file and data would not be loaded because of that. The solution was that we had to use a series of mappers and reducers to obtain the clean data that we could use.
- Another main challenge we faced was connecting to the Amazon Product API to fetch the additional data values as it would not allow many calls to the api in a second. The solution to that was to use multi-threading or use thread.sleep method to hold on for the next iteration.
- Another challenge was to deploy the data into MongoDb and access the same. It was not easy connecting the application to the MongoDb server. There were also issues loading the data into the same.

Conclusion/Result:

- We can recommend users with a variety of books from the Mahout Recommendation engine.
- We can analyze multiple reviews to determine if they are incentivized or not.
- We were successfully able to predict the relationship between helpfulness and the review text.
- We could describe the structure and nature of the review into various topics.
- Also, we could recommend friends of similar user and in turn predict the books they would like to read.

Future Scope : -

We can do real time data analytics of the Amazon Product API as we have used the dataset of the amazon movie review datasets. We can create a module for suggesting modules for helpful reviews to the users and publishers.