

FINAL PROJECT- ENGINEERING OF BIG DATA SYSTEMS

GitHub Source code: <https://github.com/sirijami/BookCrossingProject>

Dataset: **Book-crossing** dataset
<http://www2.informatik.uni-freiburg.de/~cziegler/BX/>

It includes 3 csv files:

BX-Users

- Contains the users. Note that user IDs ('User-ID') have been anonymized and map to integers. Demographic data is provided ('Location', 'Age') if available. Otherwise, these fields contain *NULL*-values.

BX-Books

- Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given ('Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher'), obtained from Amazon Web Services. Note that in case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours ('Image-URL-S', 'Image-URL-M', 'Image-URL-L'), i.e., small, medium, large. These URLs point to the Amazon web site.

BX-Book-Ratings

- Contains the book rating information. Ratings ('Book-Rating') are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

Glimpse of Dataset:

Book Ratings

User-ID	ISBN	Book-Rating
"276725"	"034545104X"	"0"
"276726"	"0155061224"	"5"
"276727"	"0446520802"	"0"
"276729"	"052165615X"	"3"
"276729"	"0521795028"	"6"
"276733"	"2080674722"	"0"
"276736"	"3257224281"	"8"
"276737"	"0600570967"	"6"
"276744"	"038550120X"	"7"
"276745"	"342310538"	"10"
"276746"	"0425115801"	"0"
"276746"	"0449006522"	"0"
"276746"	"0553561618"	"0"
"276746"	"055356451X"	"0"
"276746"	"0786013990"	"0"
"276746"	"0786014512"	"0"
"276747"	"0060517794"	"9"
"276747"	"0451192001"	"0"
"276747"	"0609801279"	"0"
"276747"	"0671537458"	"9"
"276747"	"0679776818"	"8"
"276747"	"0943066433"	"7"
"276747"	"1570231028"	"0"
"276747"	"1885408226"	"7"
"276748"	"0747558167"	"6"
"276748"	"3442437407"	"0"
"276751"	"033390804X"	"0"
"276751"	"3596218098"	"8"
"276754"	"0684867621"	"8"
"276755"	"0451166892"	"5"
"276760"	"8440682697"	"10"
"276762"	"034544003X"	"0"
"276762"	"0380000059"	"0"
"276762"	"0380711524"	"5"
"276762"	"0451167317"	"0"
"276762"	"0451454052"	"0"

Books:

File: BX-Books.csv

1	"ISBN";"Book-Title";"Book-Author";"Year-Of-Publication";"Publisher";"Image-URL-S";"Image-URL-M";"Image-URL-L"
2	"0195153448";"Classical Mythology";"Mark P. O. Morford";"2002";"Oxford University Press";"http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg";"http://0002005018";"Clara Callan";"Richard Bruce Wright";"2001";"HarperFlamingo Canada";"http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg";"http://images.amaz
3	"0060973129";"Decision in Normandy";"Carlo D'Este";"1991";"HarperPerennial";"http://images.amazon.com/images/P/0060973129.01.THUMBZZZ.jpg";"http://images.amaz
4	"0374157065";"Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It";"Gina Bari Kolata";"1999";"Farrar Straus Girc
5	"0393045218";"The Mummys of Urumchi";"E. J. W. Barber";"1999";"W. W. Norton & Company";"http://images.amazon.com/images/P/0393045218.01.THUMBZZZ.jpg";"http://images.amazon.
6	"0399135782";"The Kitchen God's Wife";"Amy Tan";"1991";"Putnam Pub Group";"http://images.amazon.com/images/P/0399135782.01.THUMBZZZ.jpg";"http://images.amazon.
7	"0425176428";"What If?: The World's Foremost Military Historians Imagine What Might Have Been";"Robert Cowley";"2000";"Berkeley Publishing Group";"http://images.amazon.
8	"0671870432";"PLEADING GUILTY";"Scott Turow";"1993";"Audiorworks";"http://images.amazon.com/images/P/0671870432.01.THUMBZZZ.jpg";"http://images.amazon.com/image
9	"0679425608";"Under the Black Flag: The Romance and the Reality of Life Among the Pirates";"David Cordingly";"1996";"Randon House";"http://images.amazon.com/in
10	"074322678X";"Where You'll Find Me: And Other Stories";"Ann Beattie";"2002";"Scribner";"http://images.amazon.com/images/P/074322678X.01.THUMBZZZ.jpg";"http://i
11	"0771074670";"Nights Below Station Street";"David Adams Richards";"1988";"Emblem Editions";"http://images.amazon.com/images/P/0771074670.01.THUMBZZZ.jpg";"http://i
12	"080652121X";"Hitler's Secret Bankers: The Myth of Swiss Neutrality During the Holocaust";"Adam Lebor";"2000";"Citadel Press";"http://images.amazon.com/images/
13	"0887841740";"The Middle Stories";"Sheila Heti";"2004";"House of Anansi Press";"http://images.amazon.com/images/P/0887841740.01.THUMBZZZ.jpg";"http://images.an
14	"1552041778";"Jane Doe";"R. J. Kaiser";"1999";"Mira Books";"http://images.amazon.com/images/P/1552041778.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/15
15	"1558746218";"A Second Chicken Soup for the Woman's Soul (Chicken Soup for the Soul Series)";"Jack Canfield";"1998";"Health Communications";"http://images.amaz
16	"1567407781";"The Witchfinder (Amos Walker Mystery Series)";"Loren D. Estleman";"1998";"Brilliance Audio - Trade";"http://images.amazon.com/images/P/1567407781
17	"1575663937";"More Cunning Than Man: A Social History of Rats and Man";"Robert Hendrickson";"1999";"Kensington Publishing Corp.,"; "http://images.amazon.com/ma
18	"1881320189";"Goodbye to the Buttermilk Sky";"Julia Oliver";"1994";"River City Pub";"http://images.amazon.com/images/P/1881320189.01.THUMBZZZ.jpg";"http://im
19	"0440234743";"The Testament";"John Grisham";"1999";"Dell";"http://images.amazon.com/images/P/0440234743.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/044
20	"0452264464";"Beloved (Plume Contemporary Fiction)";"Toni Morrison";"1994";"Plume";"http://images.amazon.com/images/P/0452264464.01.THUMBZZZ.jpg";"http://image
21	"0609804618";"Our Dumb Century: The Onion Presents 100 Years of Headlines from America's Finest News Source!";"The Onion";"1999";"Three Rivers Press";"http://in
22	"1841721522";"New Vegetarian: Bold and Beautiful Recipes for Every Occasion";"Celia Brooks Brown";"2001";"Ryland Peters & Small Ltd";"http://images.amazon.
23	"1879384493";"If I'd Known Then What I Know Now: Why Not Learn from the Mistakes of Others? : You Can't Afford to Make Them All Yourself";"J. R. Parrish";"2005
24	"0061076031";"Mary-Kate & Ashley Switching Goals (Mary-Kate and Ashley Starring in)";"Mary-Kate & Ashley Olsen";"2000";"HarperEntertainment";"http://in
25	"0439095026";"Tell Me This Isn't Happening";"Robynn Clairday";"1999";"Scholastic";"http://images.amazon.com/images/P/0439095026.01.THUMBZZZ.jpg";"http://images
26	"0689821166";"Flood : Mississippi 1927";"Kathleen Duey";"1998";"Aladdin";"http://images.amazon.com/images/P/0689821166.01.THUMBZZZ.jpg";"http://images.amazon.c
27	"0971880107";"Wild Animus";"Rich Shapero";"2004";"Too Far";"http://images.amazon.com/images/P/0971880107.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/09
28	"0345402871";"Airframe";"Michael Crichton";"1997";"Ballantine Books";"http://images.amazon.com/images/P/0345402871.01.THUMBZZZ.jpg";"http://images.amazon.com/i
29	"0345417623";"Timeline";"MICHAEL CRICHTON";"2000";"Ballantine Books";"http://images.amazon.com/images/P/0345417623.01.THUMBZZZ.jpg";"http://images.amazon.com/i
30	"0684823802";"OUT OF THE SILENT PLANET";"C.S. Lewis";"1996";"Scribner";"http://images.amazon.com/images/P/0684823802.01.THUMBZZZ.jpg";"http://images.amazon.com
31	"0375759778";"Prague : A Novel";"ARTHUR PHILLIPS";"2003";"Randor House Trade Paperbacks";"http://images.amazon.com/images/P/0375759778.01.THUMBZZZ.jpg";"http://
32	"0425163091";"Chocolate Jesus";"Stephan Jaramillo";"1998";"Berkley Publishing Group";"http://images.amazon.com/images/P/0425163091.01.THUMBZZZ.jpg";"http://im
33	"3404921038";"Wie Barney es sieht: Mordecaie Richler";"2002";"Ly?=?bbe";"http://images.amazon.com/images/P/3404921038.01.THUMBZZZ.jpg";"http://images.amazon.c
34	"3442353866";"Der Fluch der Kaiserin. Ein Richter-Di-Roman";"Eleanor Cooney";"2001";"Goldmann";"http://images.amazon.com/images/P/3442353866.01.THUMBZZZ.jpg
35	"3442421065";"Sturmzeit. Roman";"Charlotte Link";"1991";"Goldmann";"http://images.amazon.com/images/P/3442421065.01.THUMBZZZ.jpg";"http://images.amazon.com/in
36	"3442446937";"Tage der Unschuld";"Richard North Patterson";"2000";"Goldmann";"http://images.amazon.com/images/P/3442446937.01.THUMBZZZ.jpg";"http://images.ame
37	"0375406328";"Lying Awake";"Mark Salzman";"2000";"Alfred A. Knopf";"http://images.amazon.com/images/P/0375406328.01.THUMBZZZ.jpg";"http://images.amazon.com/ma
38	"0446310786";"To Kill a Mockingbird";"Harper Lee";"1960";"Little Brown & Company";"http://images.amazon.com/images/P/0446310786.01.THUMBZZZ.jpg";"http://im
39	"0446310786";"To Kill a Mockingbird";"Harper Lee";"1988";"Little Brown & Company";"http://images.amazon.com/images/P/0446310786.01.THUMBZZZ.jpg";"http://im

Users

File: BX-Users.csv

1	"User-ID";"Location";"Age"
2	"1";"nyc, new york, usa";NULL
3	"2";"stockton, california, usa";"18"
4	"3";"moscow, yukon territory, russia";NULL
5	"4";"porto, v.n.gaia, portugal";"17"
6	"5";"farnborough, hants, united kingdom";NULL
7	"6";"santa monica, califonia, usa";"61"
8	"7";"washington, dc, usa";NULL
9	"8";"timmins, ontario, canada";NULL
10	"9";"germantown, tennessee, usa";NULL
11	"10";"albacete, wisconsin, spain";"26"
12	"11";"melbourne, victoria, australia";"14"
13	"12";"fort bragg, california, usa";NULL
14	"13";"barcelona, barcelona, spain";"26"
15	"14";"mediapolis, iowa, usa";NULL
16	"15";"calgary, alberta, canada";NULL
17	"16";"albuquerque, new mexico, usa";NULL
18	"17";"chesapeake, virginia, usa";NULL
19	"18";"rio de janeiro, rio de janeiro, brazil";"25"
20	"19";"weston, ";"14"
21	"20";"langhorne, pennsylvania, usa";"19"
22	"21";"ferrol / spain, alabama, spain";"46"
23	"22";"erfurt, thueringen, germany";NULL
24	"23";"philadelphia, pennsylvania, usa";NULL
25	"24";"cologne, nrw, germany";"19"
26	"25";"oakland, califonia, usa";"55"
27	"26";"bellevue, washington, usa";NULL

Assumption:

- In users file, it contains list of the user who purchased books.
- Rating file include users who have purchased and also given ratings

Relation between the files:

- Book Ratings with user via user-id
- Book Ratings with books via ISBN

Used: Apache hadoop Ecosystem

- Hadoop HDFS (To store the data)
- Hadoop MapReduce (To perform analysis)
- Apache Pig (High level data flow language, to perform analysis much faster)
- Apache Hive (Data Warehouse)
- Apache Mahout(Collaborative filtering-recommendation)

Analysis:

- Frequency of book published each year:(MapReduce)
- Find the list of users who have most active and sent maximum number of ratings.(MapReduce)
- Average Rating of each book (ISBN is unique book id)- (MapReduce)
- Find the book details with meaningful information like ‘Book id, Book title and average rating’(Pig)
- Display top-ten list of books with highest ratings.(Pig)
- Display list of all books published from year 2001 - 2008 and sort the books by year published.(Pig)
- Find all unique list of books with book id and book name.(Pig)
- Display number of books each author published.(case-sensitive, but this sometimes leads to wrong analysis)(pig)
- Used LOWER function to first convert and then group by name to get proper results.(Pig)
- Find the users who bought the books but not yet rated any book. So that we can send notifications to them to ask to rate.(Pig)
- Find the behaviour of users who have given ratings.(Pig)
- Give the list of user in sorted order in user-id.(Hive)
- Find the user with age above 20.(Hive)
- Find users who purchase books are mostly from which location.(Hive)
- Recommend user a book based on his past purchase history.(Mahout)
- User based recommendation using another similarity.(Mahout)
- User based recommendation based on Euclidean similarity(Mahout)

Getting started:

Load the data to hdfs:

```
Sirishas-MacBook-Pro:bin sirishaepari$ hadoop fs -mkdir /BigDataProject
2018-04-16 21:15:39,064 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Sirishas-MacBook-Pro:bin sirishaepari$ hadoop fs -mkdir /BigDataProject/Dataset
2018-04-16 21:15:54,708 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Sirishas-MacBook-Pro:bin sirishaepari$ hadoop fs -mkdir /BigDataProject/OutputData
2018-04-16 21:16:09,914 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Sirishas-MacBook-Pro:bin sirishaepari$ hadoop fs -ls /BigDataProject
2018-04-16 21:16:33,655 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x  - sirishaepari supergroup          0 2018-04-16 21:15 /BigDataProject/Dataset
drwxr-xr-x  - sirishaepari supergroup          0 2018-04-16 21:16 /BigDataProject/OutputData
```



```
Sirishas-MacBook-Pro:bin sirishaepari$ hadoop fs -copyFromLocal /Users/sirishaepari/Desktop/BX-CSV-Dump /BigDataProject/Dataset
2018-04-16 21:17:53,377 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2018-04-16 21:17:53,377 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Sirishas-MacBook-Pro:bin sirishaepari$ hadoop fs -ls /BigDataProject/Dataset/BX-CSV-Dump
2018-04-16 21:18:33,871 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--  1 sirishaepari supergroup  30682276 2018-04-16 21:17 /BigDataProject/Dataset/BX-CSV-Dump/BX-Book-Ratings.csv
-rw-r--r--  1 sirishaepari supergroup  77787439 2018-04-16 21:17 /BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv
-rw-r--r--  1 sirishaepari supergroup 12284157 2018-04-16 21:17 /BigDataProject/Dataset/BX-CSV-Dump/BX-Users.csv
```

Analysis:

Frequency of book published each year

Sol: Use the ‘books’ file and set key as “Year of Published” and value as “1”.

Challenge: Was to split data properly and retrieve correct fields. As the csv format was “col1”,“col2”,“col3”

Java - FrequencyOfBookPublishedEachYear/src/FrequencyOfBookPublishedEachYearDriver.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject

Package Explorer

```

26     job.setOutputKeyClass(Text.class);
27     job.setOutputValueClass(IntWritable.class);
28
29     job.setInputFormatClassTextInputFormat.class);
30     job.setOutputFormatClassTextOutputFormat.class);
31     FileInputFormat.addInputPath(job, new Path(args[0]));
32     FileOutputFormat.setOutputPath(job, new Path(args[1]));
33
34     System.exit(job.waitForCompletion(true) ? 0 : 1);
35
36 }
37
38 public static class FrequencyOfBookPublishedEachYearMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
39
40     @Override
41     protected void map(LongWritable key, Text value,
42                         Mapper<LongWritable, Text, Text, IntWritable>.Context context)
43                         throws IOException, InterruptedException {
44         String[] tokens = value.toString().split("\n");
45         if(!tokens[3].equals("Year-Of-Publication")){
46             context.write(new Text(tokens[3]), new IntWritable(1));
47         }
48     }
49 }
50
51
52 public static class FrequencyOfBookPublishedEachYearReducer extends Reducer<Text, IntWritable,
53 Text, IntWritable>{
54
55     @Override
56     protected void reduce(Text arg0, Iterable<IntWritable> arg1,
57                          Reducer<Text, IntWritable, Text, IntWritable>.Context context)
58                         throws IOException, InterruptedException {
59         int frequency = 0;
60         for(IntWritable val : arg1){
61             frequency = frequency+ val.get();
62         }
63         context.write(arg0, new IntWritable(frequency));
64     }
65 }
66
67 }
68

```

Terminal Shell Edit View Window Help

Wed 6:02 PM Sirisha Epari

mov

1987 6529
1988 7493
1989 7937
1990 8661
1991 9389
1992 9906
1993 10602
1994 11796
1995 13548
1996 14031
1997 14892
1998 15767
1999 17432
2000 17235
2001 17360
2002 17628
2003 14359
2004 5839
2005 46
2006 3
2008 1
2010 2
2011 2
2012 1
2020 3
2021 1
2024 1
2026 1
2030 7
2037 1
2038 1
2050 2

Sirishas-MacBook-Pro:sbin sirishaepari\$

Find the list of users who are most active and sent maximum number of ratings.

Sol: Count number of ratings given by each user and sort them by ratings.

Java - CountRatingsPerUserIdMapReduce/src/CountRatingsPerUserIdDriver.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject

```

 37     job.setInputFormatClass(TextInputFormat.class);
 38     job.setOutputFormatClass(TextOutputFormat.class);
 39
 40     /* Set input & output format */
 41     FileInputFormat.addInputPath(job, new Path(args[0]));
 42     FileOutputFormat.setOutputPath(job, new Path(args[1]));
 43
 44     System.exit(job.waitForCompletion(true) ? 0 : 1);
 45
 46 }
 47
 48 public static class CountRatingsPerUserIdMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
 49
 50     @Override
 51     protected void map(LongWritable key, Text value,
 52                         Mapper<LongWritable, Text, Text, IntWritable>.Context context)
 53                         throws IOException, InterruptedException {
 54         String[] tokens = value.toString().split(";");
 55         if(!tokens[0].equals("User-ID")){
 56             context.write(new Text(tokens[0]), new IntWritable(1));
 57         }
 58     }
 59 }
 60
 61
 62 public static class CountRatingsPerUserIdReducer extends Reducer<Text, IntWritable,
 63                                         Text, IntWritable>{
 64
 65     @Override
 66     protected void reduce(Text arg0, Iterable<IntWritable> arg1,
 67                          Reducer<Text, IntWritable, Text, IntWritable>.Context context)
 68                         throws IOException, InterruptedException {
 69         int sum = 0;
 70         for(IntWritable val : arg1){
 71             sum = sum + val.get();
 72         }
 73         context.write(arg0, new IntWritable(sum));
 74     }
 75 }
 76
 77 }
 78 }
 79

```

"36606",1607
 "226545",1623
 "175003",1670
 "238781",1685
 "23768",1708
 "213350",1718
 "269566",1737
 "242824",1747
 "73394",1804
 "172742",1810
 "177458",1819
 "78783",1879
 "69697",1915
 "189334",1924
 "189835",1973
 "231210",2017
 "135149",2100
 "190925",2154
 "60244",2236
 "129358",2317
 "98741",2317
 "227447",2340
 "232131",2347
 "102967",2352
 "171118",2421
 "185233",2448
 "55492",2459
 "204864",2504
 "245963",2507
 "52584",2512
 "36836",2529
 "234623",2674
 "16795",2948
 "230522",2991
 "235105",3067
 "110973",3100
 "76352",3367
 "278418",4533
 "212898",4785
 "35859",5850
 "98391",5891
 "153662",6109
 "198711",7550
 "11676",13602

Average Rating of each book

Sol: Create a custom **countAverageTuple** writable class. I am implementing Writable class as it's passed as a value.

Hence, i only need to override the read and write methods.

In the mapper class, send the key as book id and value as countAverageTuple.

In reducer, loop through the values and find average.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java - AverageRatingOfEachBook/src/countAverageTuple.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and navigation.
- Package Explorer:** Shows the project structure:
 - AverageRatingOfEachBook
 - CountRatingsPerUserIdMapRe
 - FrequencyOfBookPublishedEac
 - RecommendBooks
 - src
 - (default package)
 - AverageRatingOfEachBook.java
 - countAverageTuple.java
- Editor:** Displays the Java code for `countAverageTuple.java`. The code implements the `Writable` interface with fields `int count` and `float average`. It overrides `readFields` and `write` methods to read/write integers and floats respectively. It also overrides `toString` to return the string representation of the tuple (`count, average`). The code is numbered from 1 to 46.

Java - AverageRatingOfEachBook/src/AverageRatingOfEachBookDriver.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject

Package Explorer

```

protected void map(
    LongWritable key,
    Text value,
    Mapper<LongWritable, Text, Text, countAverageTuple>.Context context)
throws IOException, InterruptedException {
    countAverageTuple outCountAverage = new countAverageTuple();

    String[] temp = value.toString().split(";");
    String rating = temp[2];
    rating = rating.replaceAll("\\\"", "");

    if(!rating.equals("Book-Rating")){
        outCountAverage.setCount(1);
        outCountAverage.setAverage(Float.parseFloat(rating));
        context.write(new Text(temp[1]), outCountAverage);
    }
}

public static class AverageRatingOfEachReducer extends Reducer<Text, countAverageTuple, Text, countAverageTuple>{
    @Override
    protected void reduce(
        Text arg0,
        Iterable<countAverageTuple> arg1,
        Reducer<Text, countAverageTuple, Text, countAverageTuple>.Context context)
    throws IOException, InterruptedException {
        countAverageTuple result = new countAverageTuple();
        float sum = 0;
        int count = 0;

        for(countAverageTuple v: arg1){
            sum += v.getCount() * v.getAverage();
            count = count + v.getCount();
        }
        result.setCount(count);
        result.setAverage(sum/count);

        context.write(arg0, result);
    }
}

```

Terminal Shell Edit View Window Help

sbm — bash — 130x36

```

B00005JKL8 1,10.0
B00005JL3K 1,5.0
B00005JLFV 1,10.0
B00005JLHX 1,10.0
B00005JM02 1,10.0
B00005JM5E 1,5.0
B00005LD7A 1,0.0
B00005LINE 1,10.0
B00005LOUQ 1,10.0
B00005N5J6 1,8.0
B00005NCS7 1,0.0
B00005NR00 1,8.0
B00005OC4X 2,3.0
B00005PJ9D 2,0.0
B00005Q80L 1,0.0
B00005Q8R2 2,0.0
B00005Q0XD 1,9.0
B00005QZ7U 1,10.0
B00005R2BG 2,0.0
B00005RG9 1,10.0
B00005RID3 2,5.0
B00005SNX0 1,0.0
B00005TZWI 1,9.0
B00005U7YK 3,6.0
B00005UMI4 1,10.0
B00005UMK6 2,8.5
B00005UQ9V 1,10.0
B00005V36Z 1,8.0
B00005V7MF 1,0.0
B00005VBCP 1,7.0
B00005VC3X 1,0.0
B00005VC87 1,0.0
B00005VDDED 1,0.0
B00005VFHP 1,0.0
B00005VGK1 1,0.0
B00005VHJ0 1,10.0

```

(hmm but what is first field? What does that mean?)

Find the book details with meaningful information like 'Book id, Book title and average rating'

Sol: Find the average rating of each book (above result) and join with the 'Books' file on book-id and determine the meaningful result. Limit the number of records to 10.

```
AverageRatingPerBook.pig
```

```
1 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv' USING PigStorage('') as (ISBN:chararray,BookTitle:chararray,BookAuthor:chararray,Year:int,Rating:float);
2 B = LOAD '/BigDataProject/OutputData/AverageRatingOfEachBook/part-r-00000' USING PigStorage(',') as (ISBN:chararray,count:int,average:float);
3 C = join A by $0, B by $0;
4 D = FOREACH C GENERATE $0,$1,$10;
5 E = limit D 10;
6 STORE E INTO '/BigDataProject/OutputData/AverageRatingPerBook';
```

```
2018-01-20 10:10:10,0.0 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapReduce - Total input paths to process : 1
("0000913154", "The Way Things Work: An Illustrated Encyclopedia of Technology", 8.0)
("0001010565", "Mog's Christmas", 0.0)
("0001046438", "Liar", 9.0)
("0001046713", "Twopence to Cross the Mersey", 0.0)
("000104687X", "T.S. Eliot Reading \"The Wasteland\" and Other Poems", 6.0)
("0001046934", "The Prime of Miss Jean Brodie", 0.0)
("0001047213", "The Fighting Man", 9.0)
("0001047647", "First Among Equals", 0.0)
("0001047663", "Matter Of Honour", 0.0)
("0001047868", "Kidnapped", 0.0)
```

Display top-ten list of books with highest ratings.

Sol: Sort the above result based on highest rating and display top ten records.

```
OrderByAverage.pig
```

```
1 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv' USING PigStorage('') as (ISBN:chararray,BookTitle:chararray,BookAuthor:chararray,Year:int,Rating:float);
2 B = LOAD '/BigDataProject/OutputData/AverageRatingOfEachBook/part-r-00000' USING PigStorage(',') as (ISBN:chararray,count:int,average:float);
3 C = join A by $0, B by $0;
4 D = FOREACH C GENERATE $0,$1,$10;
5 E = ORDER D BY $2 DESC;
6 F = limit E 10;
7 STORE F INTO 'BigDataProject/OutputData/sortedAverageRatingPerBook';
```

```
("1902418239", "Avebury", 10.0)
("1858683858", "Ultimate Encyclopedia of Science Fiction the De", 10.0)
("1858683688", "The Natural History Museum Book of Dinosaurs", 10.0)
("0759661804", "Life Is an Adventure", 10.0)
("0486424650", "The Communist Manifesto and Other Revolutionary Writings: Marx, Marat, Paine, Mao Tse-Tung, Gandhi and Others", 10.0)
("0619109688", "Microsoft Office XP New Features Guide: Office 2000 To XP Changes", 10.0)
("8525008044", "Pau-Brasil (Obras completas de Oswald de Andrade)", 10.0)
("968130165X", "Como Hacerse Rico Sin Preocupacione", 10.0)
("0307301900", "Barbie & 10.0)
("185863864X", "George: a 20th Century Miniature", 10.0)
```

Display list of all books published from year 2001 - 2008 and sort the books by year published.

Sol: Filter and then sort by books by year published.

```
rangeYearOfPublisher.pig
```

```
1 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv' USING PigStorage('') as (ISBN:chararray,BookTitle:chararray,BookAuthor:chararray,Year:int,Rating:float);
2 B = FOREACH A GENERATE $0,$1,$2,(int)REPLACE($3, '\\\\', ''),$4 ;
3 C = FILTER B By $3 >= 2000 AND $3 <= 2008;
4 D = ORDER C BY $3;
5 DUMP D;
```

("1582343594", "City of Masks : A Cree Black Novel", "Daniel Hecht", 2004, "Bloomsbury USA")
 ("0743257502", "The Glory Cloak : A Novel of Louisa May Alcott and Clara Barton", "Patricia O'Brien", 2004, "Touchstone")
 ("1594140332", "Designer Genes: Tales of the Biotech Revolution", "Brian Stableford", 2004, "Five Star (ME)")
 ("0765304775", "Wind Spirit (Ella Clah)", "Aimee Thurlo", 2004, "Forge Books")
 ("0765310112", "The Journal of Professor Abraham Van Helsing", "Allen C. Kupfer", 2004, "Forge Books")
 ("0849917905", "Black", "Ted Dekker", 2004, "WestBow Press")
 ("0060570563", "Babe in Toyland", "Eugenie Seifer Olson", 2004, "Avon Trade")
 ("082177557X", "Come Summer", "Sandra Steffen", 2004, "Zebra Books")
 ("0345467159", "The Alpine Pursuit (Daheim, Mary)", "MARY DAHEIM", 2004, "Ballantine Books")
 ("0802141358", "The Mammoth Cheese", "Sheri Holman", 2004, "Grove Press")
 ("0312324766", "The Men We Became : My Friendship with John F. Kennedy, Jr.", "Robert T. Littell", 2004, "St. Martin's Press")
 ("0758208022", "The Royal Treatment", "MaryJanice Davidson", 2004, "BRAVA")
 ("0373691637", "Legally Mine (Harlequin Temptation, No. 963)", "Kate Hoffmann", 2004, "Harlequin")
 ("1570722625", "When the Last Magnolia Weeps", "Mary Samps", 2004, "Silver Dagger Mysteries")
 ("0451211707", "Under a Lucky Star (Star Trilogy (Paperback))", "Diane Farr", 2004, "Signet Book")
 ("0736912932", "A Quarter for a Kiss (Clark, Mindy Starns. Million Dollar Mysteries, 4.)", "Mindy Starns Clark", 2004, "Harvest House Publishers")
 ("0765301296", "Nano", "John Robert Marlow", 2004, "Forge Books")
 ("0312865392", "Inventing Memory", "Anne Harris", 2004, "Tor Books")
 ("0881501719", "The Asey Mayo Trio: Three Cape Cod Mysteries", "Phoebe Atwood Taylor", 2005, "Countryman Press")
 ("0743492072", "Crowners Quest : A Crowner John Mystery (Crowner John Mystery)", "Bernard Knight", 2005, "Simon &)
 ("0743484894", "The Winter's Tale (Folger Shakespeare Library)", "William Shakespeare", 2005, "Washington Square Press")
 ("0060541490", "Sexy", "Joyce Carol Oates", 2005, "HarperTempest")
 ("0140144463", "The Cornish Trilogy", "Robertson Davies", 2005, "Penguin Books Ltd")
 ("0140124373", "Red Dwarf", "Grant Naylor", 2005, "Penguin Books Ltd")
 ("014002999", "Thunderball (James Bond 007)", "Ian Fleming", 2005, "Penguin Books Ltd")
 ("0060567341", "Chinese Cinderella and the Secret Dragon Society", "Adeline Yen Mah", 2005, "HarperCollins")
 ("1552782662", "The Shanghai Murders", "David Rotenberg", 2005, "McArthur &)
 ("0060738189", "Serpico", "Peter Maas", 2005, "Perennial")
 ("0743492056", "The Sanctuary Seeker : A Crowner John Mystery", "Bernard Knight", 2005, "Simon &)
 ("0140860363", "On the Road", "Jack Kerouac", 2005, "Penguin Highbridge")
 ("0671878654", "An Oblique Approach", "Eric Flint", 2005, "Baen")
 ("1552782603", "The Lake Ching Murders", "David Rotenberg", 2005, "McArthur &)
 ("0743492064", "The Poisoned Chalice : A Crowner John Mystery (Crowner John Mystery)", "Bernard Knight", 2005, "Simon &)
 ("0374103747", "In My Brother's Shadow", "Uwe Timm", 2005, "Farrar, Straus and Giroux")
 ("0752848291", "A Village Dilemma", "Rebecca Shaw", 2005, "Orion mass market paperback")
 ("0743490258", "Best of Friends", "Cathy Kelly", 2005, "Downtown Press")
 ("0141002980", "Moonraker (James Bond 007)", "Ian Fleming", 2005, "Penguin Books Ltd")
 ("0375829849", "The Golden Goose", "Dick King-Smith", 2005, "Knopf Books for Young Readers")
 ("0385338082", "Can You Keep a Secret?", "SOPHIE KINSELLA", 2005, "Delta")
 ("088150078X", "The Annulet of Gilt: An Asey Mayo Cape Cod Mystery", "Phoebe Atwood Taylor", 2005, "Countryman Press")
 ("0140622454", "A Woman of No Importance (Penguin Popular Classics)", "Oscar Wilde", 2005, "Penguin Books Ltd")
 ("1593100175", "Beauty Is Soul Deep", "Michelle Lee", 2005, "Barbour Publishing")

Find all unique list of books with book id and book name.

Sol: Use DISTINCT to find list of unique list of books from 'Books.csv'

```

UniqueBookList.pig
/usr/local/Cellar/hadoop/3.0.0/libexec/sbin/UniqueBookList.pig .

1 /* Find unique list of books with their ISBN number and Book name */
2 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv' USING PigStorage(',') as (ISBN:chararray,BookTitle:chararray,BookAu
3 B = DISTINCT A;
4 C = FOREACH B GENERATE $0,$1;
5 DUMP C;

```

```

("9999669972", "Being Bullied (Lets Talk About Series)")
("9999980538", "Murphy's Law")
("999984584", "Anointed")
("9999991556", "Impact of New Product Introductions on the Market Value of Firms (Research Program, Working Paper/Report No 89-105)")
("9999999999", "World Directory of Awards and Prizes")
("B00001IVC7", "Desperation")
("B00001U0CP", "Unnatural Exposure")
("B000023VWQ", "House Without A Key")
("B000023VNR", "The Postman Always Rings Twice")
("B000023VWW", "I, The Jury")
("B000023VWY", "The Dragon Murder Case")
("B00004THMF", "My Gal Sunday : Henry and Sunday Stories")
("B000051WXP", "Emma")
("B000051X7T", "The Oval Portrait")
("B000051XGM", "Off on a Comet")
("B000051ZU0", "Triangle")
("B000051ZUT", "The Trellisane Confrontation")
("B0000523SS", "The Covenant of the Crown")
("B0000523SU", "Ghost Ship")
("B0000523SW", "The Children of Hamlin (Star Trek: The Next Generation, Book 3) [DOWNLOAD: MICROSOFT READER]")
("B0000523SY", "Strike Zone (Star Trek: The Next Generation, Book 5) [DOWNLOAD: MICROSOFT READER]")
("B000055ZSS", "The Second Coming of Lucas Brokaw")
("B00005B4LM", "Downward to the Earth")
("B00005BBW4", "Sense and Sensibility")
("B00005BJZ7", "Fierce Eden")
("B00005LDTA", "Believing It All: What My Children Taught Me about Trout Fishing, Jelly Toast and Life")
("B00005N5J6", "Dinosaur Summer")
("B00005NCS7", "Moonlight Becomes You")
("B00005Q80L", "My Gal Sunday")
("B00005Q8R2", "Hell's Kitchen")
("B00005T7WI", "Sandkings")
("B00005U7YK", "The New Gurus -- From Sun-Tzu And Jesus To Machiavelli And Winnie The Pooh")
("B00005UMI4", "Desecration: Left Behind #9")
("B00005UMK6", "Best a Man Can Get")
("B00005V7MF", "The Company: The Story of a Murderer")
("B0000633PU", "The Story of Aladdin and the Wonderful Lamp")
("B00006391H", "Southern Nights (Florida)")
("B000063VMD", "Dying Inside")
("B0000640CT", "For Love of Mother Not")
("B000068RVK", "The Victoria's Secret Catalog Never Stops Coming: And Other Lessons I Learned From Breast Cancer")
("B000069F44", "The Lady in the Lake")
("B00006CRTE", "Devil Knows You're Dead, The: A Matthew Scudder Crime Novel")

```

Display number of books each author published.

```

GroupByAuthors.pig
/usr/local/Cellar/hadoop/3.0.0/libexec/sbin/GroupByAuthors.pig

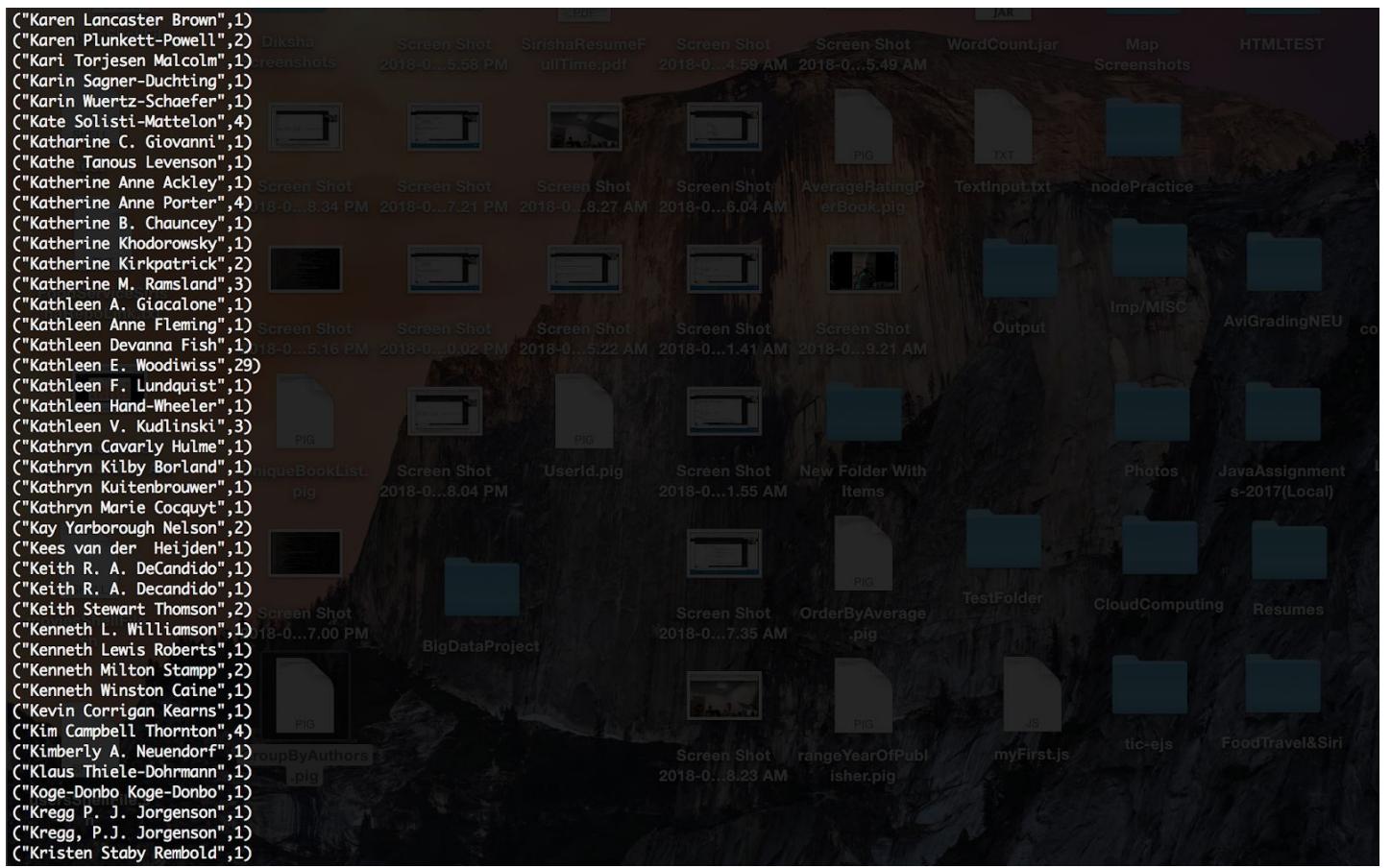
1 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv' USING PigStorage(',') as (ISBN:chararray,BookTitle:chararray,BookAuthor:chararray);
2 B = GROUP A BY $2;
3 C = FOREACH B GENERATE
4   group as author,
5   COUNT(A);
6 DUMP C;
7
8

```

/* When it group by it is case sensitive

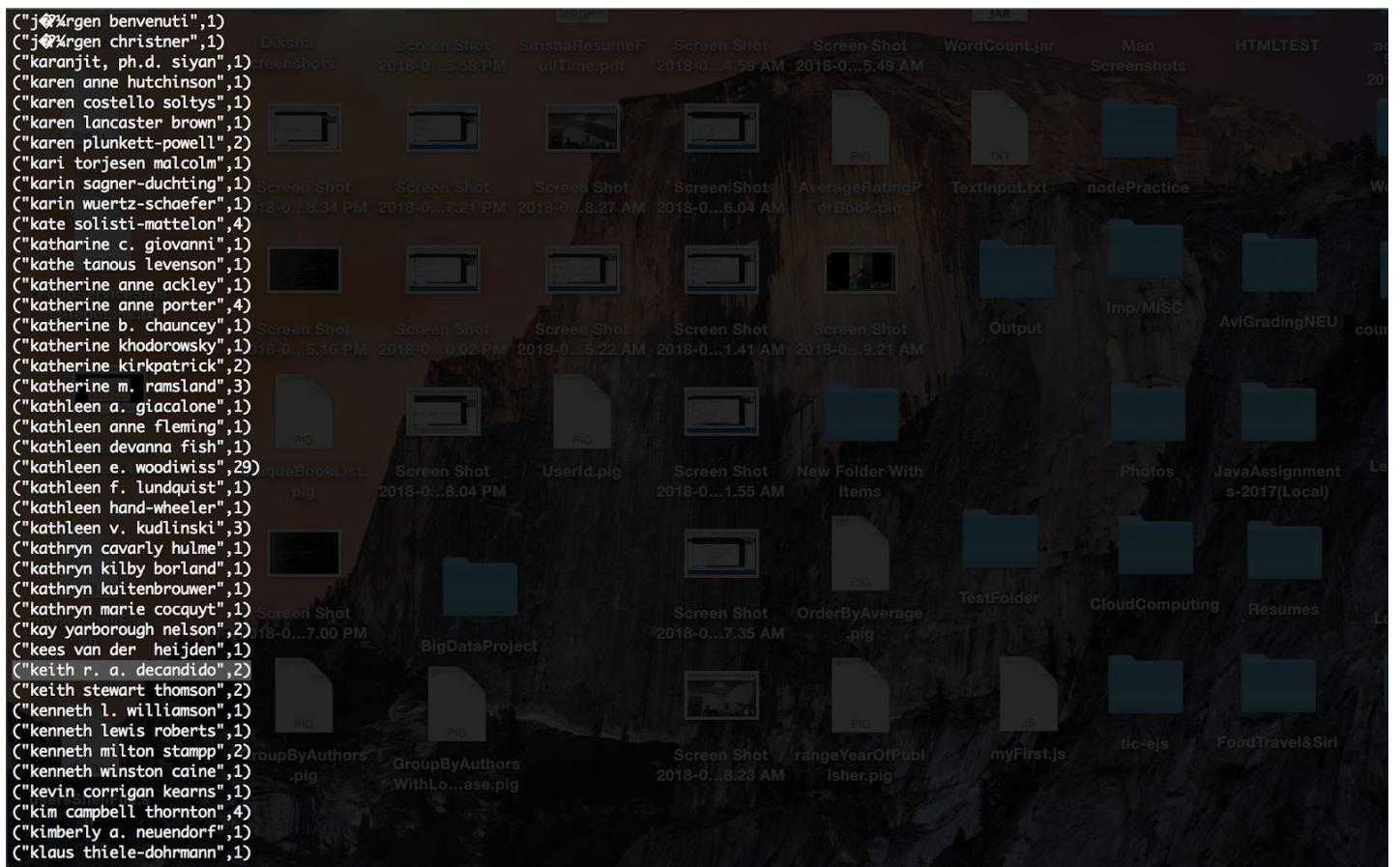
Hence you can see few same author names but grouped differently

Ex: Keith R. A. Decandido */



```
GroupByAuthorsWithLowerCase.pig
/usr/local/Cellar/hadoop/3.0.0/libexec/sbin/GroupByAuthorsWithLowerCase.pig ~
1 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Books.csv' USING PigStorage('') as (ISBN:chararray,BookTitle:chararray,BookAuthor:chararray);
2 B = GROUP A BY LOWER($2);
3 C = FOREACH B GENERATE
4   group as author,
5   COUNT(A);
6 DUMP C;
```

Used LOWER function to first convert and then group by name to get proper results.



Find the users who bought the books but not yet rated any book. So that we can send notifications to them to ask to rate.

Sol: Find the distinct users in “Book-ratings” and distinct users in “users” file and do a outer left join to get the list of the users who have not yet given ratings.

```

LeftJoin.pig
/usr/local/Cellar/hadoop/3.0.0/libexec/sbin/LeftJoin.pig .

1 A = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Users.csv' USING PigStorage(',') as (UserID:chararray,Location:chararray,Age:chararray);
2 B = LOAD '/BigDataProject/Dataset/BX-CSV-Dump/BX-Book-Ratings.csv' USING PigStorage(',') as
3 (UserID:chararray,ISBN:chararray,BookRating:chararray);
4 UniqueUsersA = DISTINCT A ;
5 UniqueUsersB = DISTINCT B ;
6 joinResult = JOIN UniqueUsersA BY $0 LEFT OUTER, UniqueUsersB BY $0;
7 usersWhoHaveNotGivenRatings = FILTER joinResult BY ($3 is null);
8 STORE usersWhoHaveNotGivenRatings INTO '/BigDataProject/OutputData/UsersWhoHaveNotGivenAnyRatings' USING PigStorage(',');
9

```

```

"278703", "emmerich, nordrhein-westfalen, germany", "27", "",  

"278704", "aberdeen, scotland, united kingdom", "49", "",  

"278705", "madrid, madrid, spain", "24", "",  

"278706", "bakersfield, california, usa", NULL, ,  

"278707", "dallas, texas, usa", NULL, ,  

"278708", "tulsa, oklahoma, usa", "55", ,  

"278709", "livemore, california, usa", "18", ,  

"278711", "toronto, ontario, canada", "41", ,  

"278712", "cranbrook, british columbia, canada", NULL, ,  

"278716", "grenoble, rhone, france", NULL, ,  

"278717", "mayaguez, puerto rico", "53", ,  

"278718", "augsburg, bavaria, germany", NULL, ,  

"278719", "bonn, nordrhein-westfalen, germany", "39", ,  

"278722", "miraflores de la sierra, massachusetts, spain", NULL, ,  

"278725", "toronto, ontario, canada", "25", ,  

"278726", "washington, dc, usa", NULL, ,  

"278727", "surrey hills, victoria, australia", NULL, ,  

"278728", "xichang, alaska, china", "15", ,  

"278730", "mankato, minnesota, usa", "24", ,  

"278733", "rayland, ohio, usa", NULL, ,  

"278734", "edgewood, new mexico, usa", NULL, ,  

"278735", "chicago, illinois, usa", "56", ,  

"278736", "douglasville, georgia, usa", "41", ,  

"278737", "germering, bayern, germany", "30", ,  

"278738", "smyrna, georgia, usa", NULL, ,  

"278739", "gillette, pennsylvania, canada", "66", ,  

"278741", "west chester, pennsylvania, usa", "50", ,  

"278742", "beijing, china, china", "22", ,  

"278743", "datong, china, china", "18", ,  

"278744", "victoria, british columbia, canada", NULL, ,  

"278745", "craigieburn, victoria, australia", "21", ,  

"278746", "johannesburg, gauteng, south africa", "55", ,  

"278748", "n.chas, south carolina, usa", "15", ,  

"278749", "oak lawn, illinois, usa", NULL, ,  

"278751", "casalnuovo (na), campania, italy", NULL, ,  

"278753", "dubbo, new south wales, australia", "18", ,  

"278754", "tallahassee, florida, usa", "27", ,  

"278756", "denmark, wisconsin, usa", "43", ,  

"278757", "concordia, missouri, usa", NULL, ,  

"278758", "tampere, finland, finland", NULL, ,  

"278761", "saskatoon, saskatchewan, canada", "39", ,  

"278762", "melbourne, victoria, australia", "38", ,  

"278763", "chester, south carolina, usa", "46", ,  

"278765", "san francisco, california, usa", NULL, ,  

"278768", "braunschweig, niedersachsen, germany", "25", ,

```

Find the behaviour of users who have given ratings.

Sol: Join the 'ratings file' with 'user file' via user id and find the information.

This helps to find user behaviour like location, age etc

```

2018-04-26 12:04:30,371 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
"6543", "strafford, missouri, usa", "34"
"23768", "st. louis, missouri, usa", "45"
"28266", "portland, oregon, usa", NULL
"28523", "springfield, missouri, usa", "24"
"39002", "san jose, ", NULL
"56157", "florissant, missouri, usa", "36"
"59102", "plantation, florida, usa", "35"
"59287", "tyler, texas, usa", NULL
"77940", "melaka, melaka, malaysia", NULL
"81977", "minneapolis, minnesota, usa", "34"
"123981", "phoenix, arizona, usa", "46"
"132081", "cranston, rhode island, usa", "35"
"133235", "addison, texas, usa", "33"
"135045", "spring, texas, usa", "29"
"138543", "mesquite, texas, usa", "21"
"144315", "tempe, arizona, usa", NULL
"144486", "jacksonville, florida, usa", "38"
"145451", "winter haven, florida, usa", "63"
"147799", "saint louis, missouri, usa", NULL
"158506", "woodland hills, california, usa", "52"
"163202", "los angeles, california, usa", "26"
"168816", "santa clara, california, usa", "35"
"173743", "gloucester, massachusetts, usa", "35"
"174834", "portland, oregon, usa", "59"
"195374", "navarre, florida, usa", "51"
"208019", "tomball, texas, usa", NULL
"227106", "st. peters, missouri, usa", "30"
"227447", "kelseyville, california, usa", "38"
"227520", "san jose, california, usa", "33"
"268191", "lewes, delaware, usa", NULL
"271448", "ona, west virginia, usa", "27"
"276725", "tyler, texas, usa", NULL

```

Hive :

Data cleaning: I have removed the first line of the csv file. So, that it will not be processed in hive while executing queries.

Created a directory named /BookCrossingDataset/Users in HDFS.

Used the hdfs put command to copy the file from local file system to hdfs.

In hive created a table named **users**

```
CREATE TABLE users ( userid STRING, location STRING, age STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ","
STORED AS TEXTFILE;
```

```
LOAD DATA INPATH '/.../BX-Users1.csv' OVERWRITE INTO TABLE users;
```

```
hive> show tables;
OK
countratingperuser
demo
users
Time taken: 0.627 seconds, Fetched: 3 row(s)
```

```
hive> describe users;
OK
userid          string
location        string
age             string
Time taken: 0.061 seconds, Fetched: 3 row(s)
```

Give the list of user in sorted order by userid.

```
Time taken: 0.619 seconds
hive> select * from users order by userid;
```

"101333"	"boise, idaho, usa"	NULL
"101334"	"troisdorf, nordrhein-westfalen, germany"	
"101335"	"halifax, nova scotia, canada"	"31"
"101336"	"miami, florida, usa"	"41"
"101337"	"chihuahua, texas, mexico"	NULL
"101338"	"centerville, ohio, usa"	NULL
"101339"	"rowland heights, california, usa"	NULL
"10134"	"herkimer, new york, usa"	"19"
"101340"	"assen, drenthe, netherlands"	NULL
"101341"	"northridge, california, usa"	NULL
"101342"	"whittier, california, usa"	"15"
"101343"	"essen, nordrhein-westfalen, germany"	NULL
"101344"	"montcada i reixac, barcelona, spain"	"29"
"101345"	"fuzhou, alabama, china"	"17"
"101346"	"hamburg, n/a, germany"	NULL
"101347"	"london, england, united kingdom"	NULL
"101348"	"london, london, united kingdom"	NULL
"101349"	"sisters, oregon, usa"	NULL
"10135" "	"proctorville, ohio, usa"	NULL
"101350"	"northfield, ohio, usa"	NULL
"101351"	"aachen, nordrhein-westfalen, germany"	"28"
"101352"	"austrheim, hordaland, norway"	NULL
"101353"	"the hague, zuid-holland, netherlands"	NULL
"101354"	"roma, lazio, italy"	NULL
"101355"	"sacramento, california, usa"	"18"
"101356"	"bruceton, tennessee, usa"	"30"
"101357"	"na, england, united kingdom"	"38"
"101358"	"paris, @le de france, france"	"49"
"101359"	"las vegas, nevada, usa"	NULL
"10136" "	"aveiro, aveiro, portugal"	"25"
"101360"	"hopewell junction, new york, usa"	"57"
"101361"	"lleida, catalunya, spain"	"27"
"101362"	"dubuque, iowa, usa"	"35"
"101363"	"firenze, toscana, italy"	"31"
"101364"	"los angeles, california, usa"	"52"
"101365"	"edinburgh, scotland, united kingdom"	"53"
"101366"	"bluffton, indiana, usa"	NULL
"101367"	"marblehead, massachusetts, usa"	"26"

Find the user with age above 20

Sol: Since there are null values in the list. First filter the data where users age is not null and then filter by where age is greater than 20.

"159303"	"biał	ystok, podlaskie, poland"
"159653"	"považ	sk&bystrica, tren&tany kraj, slovakia"
"163216"	"#379	ory, n/a, poland"
"163757"	"jarosł	aw, n/a, poland"
"174446"	"nede?ć	ina 159, istra, croatia"
"180843"	"#269	akovec, croatia, croatia"
"181814"	"krak&w,	ma?ł opolska, poland"
"192877"	"#321	ż
"195119"	"katowice, #346	lą
"197430"	"katowice, silesia (Ś	lą
"198000"	"#321	ź
"202180"	"gdań	sk, , "
"203402"	"#304	zmİ
"210971"	"new cast	le, colorado, usa"
"224770"	"#304	zmit, n/a, turkey"
"225574"	"#305	cosı
"226045"	"#304	stanbul, İ
"227753"	"adelaide	, south austra ia, australia"
"230148"	"#304	stanbul, İ
"230585"	"gdań	sk, pomorskie, poland"
"233200"	"andrespol, #322	ł, poland"
"233313"	"t#345	inec 6, n/a, czech republic"
"236551"	"garč	in, n/a, croatia"
"239900"	"shanghai	， china, california, china"
"247501"	"westl	ake, ohio, usa"
"249420"	"evesham	worcs, england, united kingdom"
"253804"	"#304	stanbul, turkey, turkey"
"259769"	"wrocł	aw, n/a, poland"
"260445"	"#304	stanbul, İ
"262513"	"savi?č	enta, england, croatia"
"263916"	"#321	ź
"263972"	"l	ondon, england, united kingdom"
"265245"	"al	buquerque, new mexico, usa"
"268506"	"pabianice, woj.	ł ł, poland"
"269315"	"krak&w,	ma?ł opolska, poland"
"271934"	"#304	stanbul, n/a, turkey"
"272685"	"jū	rmala, rī
"275113"	"val	jevo, n/a, yugoslavia"

Find users who purchase books are mostly from which location?

Sol: group by location and count, and then sort the results.

From this data set we understand most of the users are from "london, england, united kingdom".

		Screen Shot	Map	HTMLTEST				
"atlanta, georgia, usa"	582	2018-0...6.57 AM	2018-0...9.25 AM	2018-0...9.52 AM	2018-0...5.47 AM	2018-0...5.21 AM	Screenshots	
"brooklyn, new york, usa"	606							
"minneapolis, minnesota, usa"	609							
"albuquerque, new mexico, usa"	622							
"montreal, quebec, canada"	634							
"adelaide, south australia, australia"	636							
"philadelphia, pennsylvania, usa"	658							
"tucson, arizona, usa"	664							
"perth, western australia, australia"	673							
"st. louis, missouri, usa"	704							
"edmonton, alberta, canada"	716							
"brisbane, queensland, australia"	751							
"victoria, british columbia, canada"	800							
"barcelona, catalunya, spain"	829							
"roma, lazio, italy"	842							
"hamburg, hamburg, germany"	883							
"barcelona, barcelona, spain"	905							
"calgary, alberta, canada"	981							
"austin, texas, usa"	986							
"los angeles, california, usa"	1005							
"milano, lombardia, italy"	1157							
"berlin, berlin, germany"	1173							
"houston, texas, usa"	1187							
"ottawa, ontario, canada"	1243							
"san francisco, california, usa"	1269							
"san diego, california, usa"	1277							
"vancouver, british columbia, canada"	1359							
"madrid, madrid, spain"	1400							
"new york, new york, usa"	1411							
"seattle, washington, usa"	1484							
"chicago, illinois, usa"	1526							
"portland, oregon, usa"	1629							
"melbourne, victoria, australia"	1708							
"sydney, new south wales, australia"	1744							
"toronto, ontario, canada"	2250							
"london, england, united kingdom"	2506							
Time taken:	4.884 seconds, Fetched:	57310 row(s)						

Recommend user a book based on his past purchase history.

Sol: Firstly, I have to format the data.

- Remove all the quotes from the file
- Since, the book id had mix of integer & character, I converted that into Long.
- Stored in a csv file named **formattedBookRating.csv**
- In another class, created the data model, user similarity, user Neighborhood, recommender.
- It suggests user a book based on previous interests.
- Here is choose pearsonCorrelationSimilarity

The format data looks like below:

276726,155061224,5
276727,446520802,0
276729,521795028,6
276733,2080674722,0
276736,3257224281,8
276737,600570967,6
276745,342310538,10
276746,425115801,0
276746,449006522,0
276746,553561618,0
276746,786013990,0
276746,786014512,0
276747,60517794,9
276747,451192001,0
276747,609801279,0
276747,671537458,9
276747,679776818,8
276747,943066433,7
276747,1570231028,0
276747,1885408226,7
276748,747558167,6
276748,3442437407,0
276751,3596218098,8
276754,684867621,8
276755,451166892,5
276760,8440682697,10
276762,380000059,0
276762,380711524,5
276762,451167317,0
276762,451454952,0
276762,843920262,0
276762,3404122879,0
276762,3404182928,0
276762,3404611306,0
276762,342662429,0
276762,3426690179,0
276762,3442424216,0
276762,3442425573,0
276762,3453092007,8
276762,3453157745,0
276762,3453176944,0
276762,3453185137,0
276762,3453185323,0
276762,3453213025,3
276762,3453877241,0
276762,3492226604,0
276762,3517017442,0

Java - RecommendBooks/src/main/java/MahoutRecommendation/RecommendBooks/BookCrossingDataConvert.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject

```

2 import java.io.BufferedReader;
3 import java.io.BufferedWriter;
4 import java.io.FileReader;
5 import java.io.FileWriter;
6 import java.io.IOException;
7
8 public class BookCrossingDataConvert {
9
10
11    public static void main(String[] args) throws IOException {
12        BufferedReader br = new BufferedReader(new FileReader("/Users/sirishaepari/Desktop/BigData/BigDataSets"
13                + "/BX-Csv-Dump/BX-Book-Ratingscopy.csv"));
14        BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/sirishaepari/Desktop/BigData/BigDataSets"
15                + "/BX-Csv-Dump/formatedBookRatings.csv"));
16
17        String record;
18        while((record = br.readLine()) != null){
19            String replaceQuotes = record.replaceAll("\\"", "\"");
20            String[] tokens = replaceQuotes.split(";", -1);
21            try{
22                bw.write(Long.parseLong(tokens[0]) + "," + Long.parseLong(tokens[1]) + "," + Long.parseLong(tokens[2]) + "\n");
23            }catch(Exception e){
24                continue;
25            }
26        }
27        br.close();
28        bw.close();
29    }
30
31
32 }
33

```

Problems Javadoc Declaration Console

<terminated> AppOne [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Apr 27, 2018, 12:18:05 PM)

log4j:WARN No appenders could be found for logger (org.apache.mahout.cf.taste.impl.model.file.FileDataModel).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Recommended Item: RecommendedItem[item:3423202327, value:10.0]

Java - RecommendBooks/src/main/java/MahoutRecommendation/RecommendBooks/AppOne.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject

```

19 public class AppOne {
20
21     public static void main(String[] args) {
22         try {
23             //STEP 1. Create the DataModel
24             DataModel model = new FileDataModel(new File("/Users/sirishaepari/Desktop/BigData/BigDataSets"
25                     + "/BX-Csv-Dump/formatedBookRatings.csv"));
26
27             UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
28
29             //STEP 3. Define neighborhood
30             UserNeighborhood neighborhood = new NearestNUserNeighborhood(276861, similarity, model);
31
32             //STEP 4. Build a Recommender Engine
33             Recommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);
34
35             //STEP 5. Ask the Recommender Engine for Recommendation
36             List<RecommendedItem> recommendations = recommender.recommend(276861, 1);
37
38             for (RecommendedItem item : recommendations)
39                 System.out.println("Recommended Item: " + item);
40         } catch (IOException ex) {
41             System.out.println("EXCEPTION: " + ex.getMessage());
42         } catch (TasteException ex) {
43             System.out.println("EXCEPTION: " + ex.getMessage());
44         }
45     }
46
47
48 }
49
50

```

Problems Javadoc Declaration Console

<terminated> AppOne [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Apr 27, 2018, 12:18:05 PM)

log4j:WARN No appenders could be found for logger (org.apache.mahout.cf.taste.impl.model.file.FileDataModel).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Recommended Item: RecommendedItem[item:3423202327, value:10.0]

User Recommendation using Euclidean Similarity: // Took help from internet

Java - RecommendBooks/src/main/java/MahoutRecommendation/RecommendBooks/App.java - Eclipse - /Users/sirishaepari/Documents/BookCrossingProject

Package Explorer

```

27 }
28     try {
29         //STEP 1. Create the DataModel
30         DataModel model = new FileDataModel(new File("/Users/sirishaepari/Desktop/BigData/BigDataSets"
31             + "/BX-CSV-Dump/formattedBookRatings.csv"));
32         BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/sirishaepari/Desktop/BigData/BigDataSets/booksUserRecomm"
33             bw.write("User based Recommendation Euclidean Similarity" + "\n");
34
35         //STEP 2. Find Similar users
36         UserSimilarity similarity = new org.apache.mahout.cf.taste.impl.similarity.EuclideanDistanceSimilarity(model);
37
38
39         //STEP 3. Define neighborhood
40         UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity, model);
41
42         //STEP 4. Build a Recommender Engine
43         Recommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);
44
45         int x = 1;
46         for(LongPrimitiveArrayIterator users = (LongPrimitiveArrayIterator) model.getUserIDs();
47             users.hasNext()){
48             long userId = users.nextLong();
49             bw.write("for userid " + userId + "\n");
50             List<RecommendedItem> recommendations = recommender.recommend(userId, 5);
51             for(RecommendedItem rec: recommendations){
52                 bw.write(rec.getItemID() + " , " + rec.getValue() + "\n");
53             }
54             x++;
55             if(x > 10){
56                 break;
57             }
58         }
59         bw.close();
60
61     } catch (IOException ex) {
62         System.out.println("EXCEPTION: " + ex.getMessage());
63     } catch (TasteException ex) {
64         System.out.println("EXCEPTION: " + ex.getMessage());
65     }
66 }
67
68

```

booksUserRecommendation.csv

```

1 for userid 2
2 for userid 7
3 for userid 8
4 684865459 ,10.0
5 440977096 ,10.0
6 743454529 ,10.0
7 553587188 ,10.0
8 866254099 ,10.0
9 for userid 9
10 866254099 ,10.0
11 451519167 ,10.0
12 898153883 ,10.0
13 395193958 ,10.0
14 671601075 ,10.0
15 for userid 10
16 8432249303 ,9.0
17 8435015955 ,8.9
18 8445071416 ,8.8
19 8408039369 ,8.666667
20 8483101610 ,8.666667
21 for userid 12
22 for userid 14
23 60975288 ,10.0
24 385600054 ,10.0
25 2277241202 ,10.0
26 60294884 ,10.0
27 761109196 ,10.0
28 for userid 16
29 140440453 ,10.0
30 385121679 ,10.0
31 812512413 ,10.0
32 609803875 ,10.0
33 449225119 ,10.0
34 for userid 17
35 1562470345 ,10.0
36 849912970 ,10.0
37 60953691 ,10.0
38 1853260002 ,10.0
39 385121679 ,10.0
40 for userid 19
41 1566404398 ,10.0
42 609804138 ,10.0
43 515136530 ,10.0
44 684801221 ,10.0
45 439064864 ,10.0
46

```

Things I learnt from this project.

- How to transform the data in cleaner and good format to run analysis.
- How important data cleansing is.
- Look for Case sensitive, null, empty values, symbols like & etc are may give somewhat wrong analysis.
- How fast and simple pig (data flow language)
- Once you have data in HIVE, you can use any visualization tools like Tableau, Heu, Banana etc

<https://wiki.apache.org/hadoop/MountableHDFS>

<https://github.com/d3/d3/wiki/Gallery>

