Northeastern University

FINAL PROJECT REPORT

Topic: Book Crossing Analysis

Academic Term: Fall 2020

INFO 7250 – Engineering Big Data Systems

Professor: Mr. Yusuf Ozbek

BOOK CROSSING DATASET

SUMMARY

This dataset consists of information about books, users and ratings with a size of 115MB

BX_Users.csv contains information about 2,78,858 users with different fields like userId, location and age
BX_Books.csv contains information about 2,71,379 books with different fields like title, author, year of publication, publisher and URL's linking to images are given in 3 different fields.
BX_Book_Ratings.csv contains 11,49,780 ratings from the users with different fields userId, ISBN and bookRating

After the dataset is cleaned, the dataset contains 0.067GB (i.e., 67MB) of data (all the steps to clean the dataset are kept in a new document)

Dataset Link: https://grouplens.org/datasets/book-crossing/

S.No	Analysis Description	Implementation Details
1	To find out the maximum age of user from each country	MongoDB
2	Top 25 explicit rated book details	Apache Pig (Joins)
3	Most popular author based on the books count	Apache Pig
4	Publishers with maximum book count in the year 2003	Apache Pig
5	To bin ratings into categories	Apache Pig
6	To find out number of users per country of age group 16 to 25	Apache Hive
7	To find out number of times each rating has been given for a book	Apache Hive
8	To find out details of user who gave highest rating for book	Apache Hive (Join)
9	Mahout book recommendation system	Mahout
10	To find out books that have rating	Map Reduce (Joins)
11	Binning users based on age	Binning (Organization Pattern)
12	To find out highest rated books	TOP N Filtering Pattern
13	To find out number of books published each year	Counting with Counters
14	To find out list of ratings in descending order, different age groups have given for books	Secondary Sorting, Chaining and Joins
15	Dataset Cleaning and Visualizations	Python, PowerBI

MONGODB

- Imported the dataset using mongoimport utility and created the database

```
C:\Users\kandr>mongoimport --db=booksDB --collection=books --type=csv --headerline --file=C:\Users\kandr\Desktop\books.csv
2020-12-13T22:25:07.220-0800    connected to: mongodb://localhost/
2020-12-13T22:25:10.150-0800    251204 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kandr>mongoimport --db=booksDB --collection=users --type=csv --headerline --file=C:\Users\kandr\Desktop\users.csv
2020-12-13T22:25:41.936-0800    connected to: mongodb://localhost/
2020-12-13T22:25:44.543-0800    278858 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kandr>mongoimport --db=booksDB --collection=ratings --type=csv --headerline --file=C:\Users\kandr\Desktop\ratings.csv
2020-12-13T22:26:16.532-0800    connected to: mongodb://localhost/
2020-12-13T22:26:19.532-0800    [########................] booksDB.ratings        7.32MB/20.3MB (36.0%)
2020-12-13T22:26:22.532-0800    [################.......] booksDB.ratings        14.5MB/20.3MB (71.4%)
2020-12-13T22:26:24.673-0800    [########################] booksDB.ratings        20.3MB/20.3MB (100.0%)
2020-12-13T22:26:24.673-0800    1031174 document(s) imported successfully. 0 document(s) failed to import.
```

```
> show databases;
Lab03                0.000GB
Quiz1_TA             0.000GB
accessLogDB          0.001GB
admin                0.000GB
booksDB              0.067GB
config               0.000GB
contactManagementDB  0.000GB
gameCollectionDB     0.000GB
local                0.000GB
movieLens10M         0.335GB
movieLensDB          0.030GB
mydb                 0.000GB
newNyseDB            0.706GB
test                 0.000GB
testdb               0.000GB
userDB               0.000GB
> use booksDB;
switched to db booksDB
> show collections;
books
ratings
users
```

```
> db.books.find().limit(3).pretty();
{
        "_id" : ObjectId("5fd705434261425e45291ac5"),
        "isbn" : 195153448,
        "book_title" : "Classical Mythology",
        "book_author" : "Mark P. O. Morford",
        "year_of_publication" : 2002,
        "publisher" : "Oxford University Press"
}
{
        "_id" : ObjectId("5fd705434261425e45291ac6"),
        "isbn" : 60973129,
        "book_title" : "Decision in Normandy",
        "book_author" : "Carlo D'Este",
        "year_of_publication" : 1991,
        "publisher" : "HarperPerennial"
}
{
        "_id" : ObjectId("5fd705434261425e45291ac7"),
        "isbn" : 2005018,
        "book_title" : "Clara Callan",
        "book_author" : "Richard Bruce Wright",
        "year_of_publication" : 2001,
        "publisher" : "HarperFlamingo Canada"
}
```

```
> db.users.find().limit(3).pretty();
{
        "_id" : ObjectId("5fd70565fc6c083867f2fb72"),
        "user_id" : 1,
        "city" : "nyc",
        "state" : "new york",
        "country" : "usa",
        "age" : 34
}
{
        "_id" : ObjectId("5fd70565fc6c083867f2fb73"),
        "user_id" : 2,
        "city" : "stockton",
        "state" : "california",
        "country" : "usa",
        "age" : 18
}
{
        "_id" : ObjectId("5fd70565fc6c083867f2fb74"),
        "user_id" : 5,
        "city" : "farnborough",
        "state" : "hants",
        "country" : "united kingdom",
        "age" : 41
}
```

```
> db.ratings.find().limit(3).pretty();
{
        "_id" : ObjectId("5fd70588c7471f3ec0e9918e"),
        "user_id" : 276727,
        "isbn" : 446605239,
        "rating" : 0
}
{
        "_id" : ObjectId("5fd70588c7471f3ec0e9918f"),
        "user_id" : 276726,
        "isbn" : 867210613,
        "rating" : 5
}
{
        "_id" : ObjectId("5fd70588c7471f3ec0e99190"),
        "user_id" : 276729,
        "isbn" : 521795028,
        "rating" : 6
}
```

TEXT SEARCH SCORE IMPLEMENTATION

Text search assigns a score to each document that contains the search term in the indexed fields. The score generated determines the relevance of a document to a given search query.

```
> db.books.createIndex(
... {
... book_title:'text',
... book_author:'text',
... publisher:'text'},
... { weights:{
... book_title:10,
... book_author:5},
... name:'text_index_for_books'
... }
... );
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

```
> db.books.find({$text:{$search:'brain'}},{book_title:1,titleScore:{$meta:"textScore"}}).sort({titleScore:{$meta:"textScore"}}).limit(5).pretty();
{
        "_id" : ObjectId("5fd705454261425e452c4eec"),
        "book_title" : "Marshall Brain's How Stuff Works : How Much Does the Earth Weigh? (Marshall Brain's How Stuff Works)",
        "titleScore" : 12.613636363636363
}
{
        "_id" : ObjectId("5fd705444261425e452b1aed"),
        "book_title" : "Brain's clinical neurology (Oxford medical publications)",
        "titleScore" : 11.458333333333334
}
{
        "_id" : ObjectId("5fd705454261425e452bca5b"),
        "book_title" : "Me and My Little Brain (Great Brain)",
        "titleScore" : 11.25
}
{
        "_id" : ObjectId("5fd705454261425e452b9015"),
        "book_title" : "Me and My Little Brain (Great Brain)",
        "titleScore" : 11.25
}
{
        "_id" : ObjectId("5fd705434261425e452a09ed"),
        "book_title" : "The Great Brain (Great Brain)",
        "titleScore" : 11.25
}
```

```
> db.books.find({$text:{$search:'"chocolate on the brain"'}},{book_title:1,titleScore:{$meta:"textScore"}}).sort({titleScore:{$meta:"textScore"}}).limi
t(5).pretty();
{
        "_id" : ObjectId("5fd705464261425e452ced32"),
        "book_title" : "Chocolate On The Brain : Foolproof Recipes for Unrepentant Chocoholics",
        "titleScore" : 11.666666666666668
}
```

CODE

```
//TEXT SEARCH INDEX
db.books.createIndex(
    {
    book_title:'text',
    book_author:'text',
    publisher:'text'},
{ weights:{
            book_title:10,
            book_author:5},
    name:'text_index_for_books'
}
);

//TEXT SEARCH SCORE

db.books.find({$text:{$search:'brain'}},{book_title:1,titleScore:{$meta:"textScore"}}).sort({titleScore:{$meta:"textScore"}}).limit(5);
```

ANALYSIS

1. **To find out the maximum age of users from each country**

```
> db.MaximumAgeFromEveryCountry.find().sort({"value":-1}).limit(5).pretty();
{ "_id" : { "country" : "usa" }, "value" : { "maximumAge" : 99 } }
{ "_id" : { "country" : "italy" }, "value" : { "maximumAge" : 99 } }
{ "_id" : { "country" : "canada" }, "value" : { "maximumAge" : 99 } }
{
        "_id" : {
                "country" : "australia"
        },
        "value" : {
                "maximumAge" : 97
        }
}
{ "_id" : { "country" : "slovenia" }, "value" : { "maximumAge" : 97 } }
```

CODE

```
//Max age of people from each country

var map3=function(){if(this.country!='n/a'){emit({country:this.country},{age:this.age});}}

var reduce3=function(key,values){var maxAge=0; values.forEach((val)=>{if(val.age>maxAge){maxAge=val.age;}});return {maximumAge:maxAge};}

db.users.mapReduce(map2,reduce2,{out:"MaxAgeFromCountry"})

db.MaxAgeFromCountry.find();
```

APACHE PIG

Steps

- Loaded users, ratings and books dataset using PigStorage

```
grunt> users = LOAD '/home/meghana/Downloads/Project/users.csv' USING PigStorage(',') AS (user_id:chararray, city:chararray, state:chararray
, country:chararray, age:int);
2020-12-16 14:46:47,515 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
grunt> books = LOAD '/home/meghana/Downloads/Project/books.csv' USING PigStorage(',') AS (isbn:chararray,book_title:chararray,book_author:ch
ararray,year_of_publication:int,publisher:chararray);
2020-12-16 14:47:38,056 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
grunt> ratings = LOAD '/home/meghana/Downloads/Project/ratings.csv' USING PigStorage(',') AS (user_id:chararray, isbn:chararray, rating:int)
;
2020-12-16 14:47:58,968 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
```

```
grunt> describe users;
users: {user_id: chararray,city: chararray,state: chararray,country: chararray,age: int}
grunt> describe books;
books: {isbn: chararray,book_title: chararray,book_author: chararray,year_of_publication: int,publisher: chararray}
```

- Displayed few rows of the data to view the structure of loaded data

```
(1,nyc,new york,usa,34)
(2,stockton,california,usa,18)
(3,moscow,yukon territory,russia,38)
(4,porto,v.n.gaia,portugal,17)
(5,farnborough,hants,united kingdom,41)
(6,santa monica,california,usa,61)
(7,washington,dc,usa,50)
(8,timmins,ontario,canada,36)
(9,germantown,tennessee,usa,34)
```

```
(0195153448,Classical Mythology,Mark P. O. Morford,2002,Oxford University Press)
(0002005018,Clara Callan,Richard Bruce Wright,2001,HarperFlamingo Canada)
(0060973129,Decision in Normandy,Carlo D'Este,1991,HarperPerennial)
(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 Giroux)
(0393045218,The Mummies of Urumchi,E. J. W. Barber,1999,W. W. Norton & Company)
(0399135782,The Kitchen God's Wife,Amy Tan,1991,Putnam Pub Group)
(0425176428,What If?: The World's Foremost Military Historians Imagine What Might Have Been,Robert Cowley,2000,Berkley Publishing Group)
(0671870432,PLEADING GUILTY,Scott Turow,1993,Audioworks)
(0679425608,Under the Black Flag: The Romance and the Reality of Life Among the Pirates,David Cordingly,1996,Random House)
```

- Filtering out ratings column for later use

```
grunt> ratings_explicit = filter ratings by (rating!=0);
grunt> describe ratings_explicit;
ratings_explicit: {user_id: chararray,isbn: chararray,rating: int}
```

```
grunt> re = LIMIT ratings_explicit 10;
grunt> dump re;
2020-12-16 14:52:00,809 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: FILTER,LIMIT
2020-12-16 14:52:00,834 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2020-12-16 14:52:00,877 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2020-12-16 14:52:00,933 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699138048 to monitor. collectionUsageThreshold = 489396640, usageThreshold = 489396640
2020-12-16 14:52:01,018 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold:
```

- Displayed the first ten rows of the data to view the structure of loaded data

```
2020-12-16 14:52:06,879 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-16 14:52:06,881 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2020-12-16 14:52:06,881 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 14:52:06,899 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 14:52:06,899 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(276726,0867210613,5)
(276729,052165615X,3)
(276729,0521795028,6)
(276744,044023722X,7)
(276747,0060517794,9)
(276747,0671537458,9)
(276747,0679776818,8)
(276747,0943066433,7)
(276747,1885408226,7)
(276748,0747558167,6)
```

2. TOP 25 EXPLICIT RATED BOOK DETAILS

First, let's calculate top 25 rated books (with explicit rating i.e rating from 1 to 10) and later let's calculate top rated book details

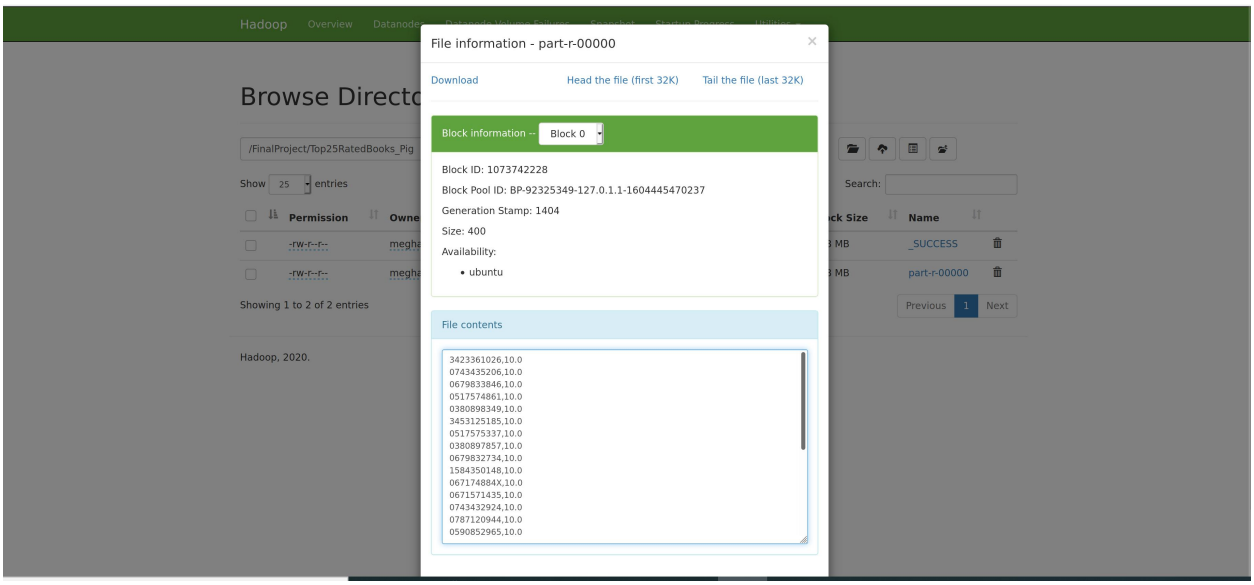
- Load the datasets
- Filtering explicit ratings from ratings
- Grouping explicit ratings by isbn
- Calculating the average of ratings
- Sorting the ratings to descending order
- Restricting the rows to 25 by using LIMIT

```
grunt> history;
1   users = LOAD '/home/meghana/Downloads/Project/users.csv' USING PigStorage(',') AS (user_id:chararray, city:chararray, state:chararray, c
ountry:chararray, age:int);
2   books = LOAD '/home/meghana/Downloads/Project/books.csv' USING PigStorage(',') AS (isbn:chararray,book_title:chararray,book_author:chara
rray,year_of_publication:int,publisher:chararray);
3   ratings = LOAD '/home/meghana/Downloads/Project/ratings.csv' USING PigStorage(',') AS (user_id:chararray, isbn:chararray, rating:int);
4   ratings_explicit = filter ratings by (rating!=0);
5   re = LIMIT ratings_explicit 10;
6   u = limit users 10;
7   b = limit books 10;
8   bookGroup = Group ratings_explicit by isbn;
9   bookRating = FOREACH bookGroup GENERATE group as isbn, AVG(ratings_explicit.rating) as average;
10   sortData = ORDER bookRating by average desc;
11   top_25 = limit sortData 25;
12   store top_25 into 'hdfs://localhost:9000/FinalProject/Top25RatedBooks_Pig' using PigStorage(',');
```

- Output from terminal

```
2020-12-16 16:16:40,733 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 16:16:40,748 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 16:16:40,749 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(3423361026,10.0)
(0743435206,10.0)
(0679833846,10.0)
(0517574861,10.0)
(0380898349,10.0)
(3453125185,10.0)
(0517575337,10.0)
(0380897857,10.0)
(0679832734,10.0)
(1584350148,10.0)
(067174884X,10.0)
(0671571435,10.0)
(0743432924,10.0)
(0787120944,10.0)
(0590852965,10.0)
(0253188709,10.0)
(1584652144,10.0)
(0253203317,10.0)
(0380895897,10.0)
(1584690445,10.0)
(1584690453,10.0)
(1584690461,10.0)
(158469047X,10.0)
(1584790466,10.0)
(0849917077,10.0)
```

- Output from HDFS



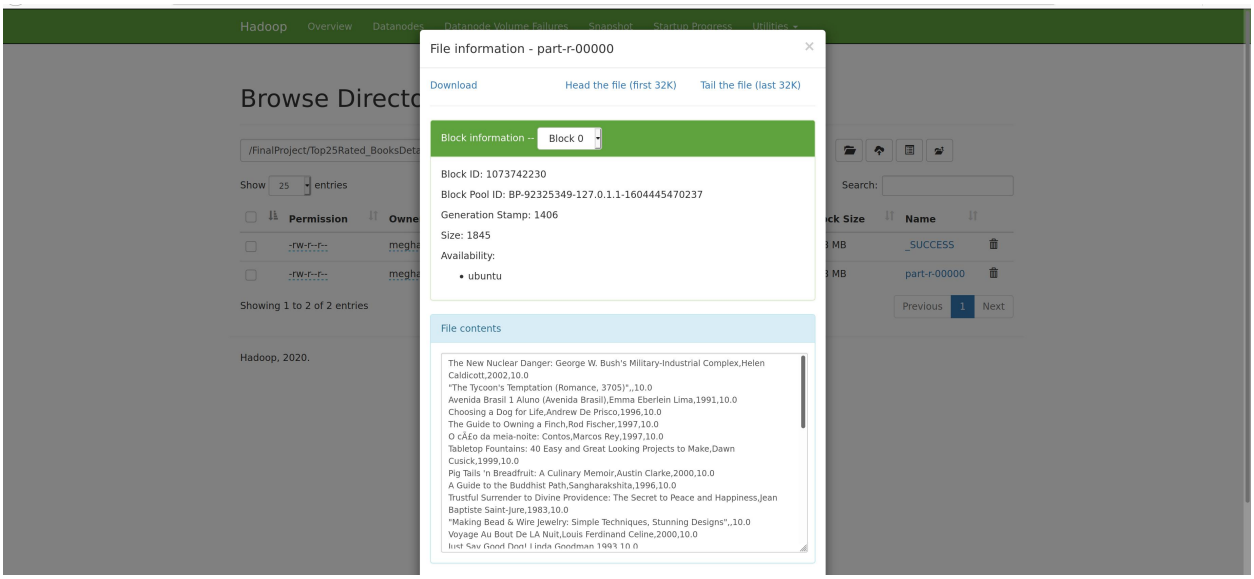
CODE

```
//Highest rated books (explicit ratings only)
bookGroup = Group ratings_explicit by isbn;
describe bookGroup;
bookRating = FOREACH bookGroup GENERATE group as isbn, AVG(ratings_explicit.rating) as average;
sortData = ORDER bookRating by average desc;
top_25 = limit sortData 25;
store top_25 into 'hdfs://localhost:9000/FinalProject/Top25RatedBooks_Pig' using PigStorage(',');
dump top_25;
// joining ratings and books to display book details
explicitRatedBooks = JOIN books BY isbn LEFT OUTER, bookRating BY isbn;
top25_ratings = LIMIT explicitRatedBooks 25;
DUMP top25_ratings;
```

TOP 25 EXPLICIT RATED BOOK DETAILS

```
grunt> describe top25_bookDetails;
top25_bookDetails: {books::book_title: chararray,books::book_author: chararray,books::year_of_publication: int,bookRating::average: double}
```

- Output from hdfs



- Output from the terminal

```
2020-12-16 16:43:54,845 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 16:43:54,845 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(The New Nuclear Danger: George W. Bush's Military-Industrial Complex,Helen Caldicott,2002,10.0)
("The Tycoon's Temptation (Romance, 3705)",,10.0)
(Avenida Brasil 1 Aluno (Avenida Brasil),Emma Eberlein Lima,1991,10.0)
(Choosing a Dog for Life,Andrew De Prisco,1996,10.0)
(The Guide to Owning a Finch,Rod Fischer,1997,10.0)
(O cÃ£o da meia-noite: Contos,Marcos Rey,1997,10.0)
(Tabletop Fountains: 40 Easy and Great Looking Projects to Make,Dawn Cusick,1999,10.0)
(Pig Tails 'n Breadfruit: A Culinary Memoir,Austin Clarke,2000,10.0)
(A Guide to the Buddhist Path,Sangharakshita,1996,10.0)
(Trustful Surrender to Divine Providence: The Secret to Peace and Happiness,Jean Baptiste Saint-Jure,1983,10.0)
("Making Bead & Wire Jewelry: Simple Techniques, Stunning Designs",,10.0)
(Voyage Au Bout De LA Nuit,Louis Ferdinand Celine,2000,10.0)
(Just Say Good Dog!,Linda Goodman,1993,10.0)
("The White Man's Indian: Images of the American Indian, from Columbus to the Present",,10.0)
(The Craft & Art of Bamboo: 30 Elegant Projects to make for Home and Garden,Carol Stangler,2001,10.0)
(Jesus Freak,DC Talk,1997,10.0)
(O menino maluquinho,Ziraldo,1988,10.0)
(Schindler's List / Piano Solos,John Williams,2000,10.0)
(The Pirates of Penzance or the Slave of Duty: Or the Slave of Duty,W. S. Gilbert,1986,10.0)
(May It Please the Court : The First Amendment: Live Recordings and Transcripts of the Oral Arguments Made Before the Supreme Court in Sixte
en Key First Amendment Cases,Peter Irons,1997,10.0)
(The Beatles Complete Scores,Hal Leonard Publishing Corporation,1993,10.0)
(The Great Cat Massacre: And Other Episodes in French Cultural History,Robert Darnton,1985,10.0)
("Substitute Father (Cowboy Grooms Wanted!) (Romance, 3597 : Cowboy Grooms Wanted!)",,10.0)
(Miss Saigon: Piano Vocal Selections,Hal Leonard Publishing Corporation,1990,10.0)
(The stolen child: A first novel,Colin Cheong,1991,10.0)
```

CODE

- Joining the datasets books and bookRating(obtained from above)
- Sorting the explicitRatedBooks by descending order
- Restricting the rows to 25 by using LIMIT
- Looping through all the explicitRatedBooks and generating a new relation alias named top25_bookDetails with all the required book details

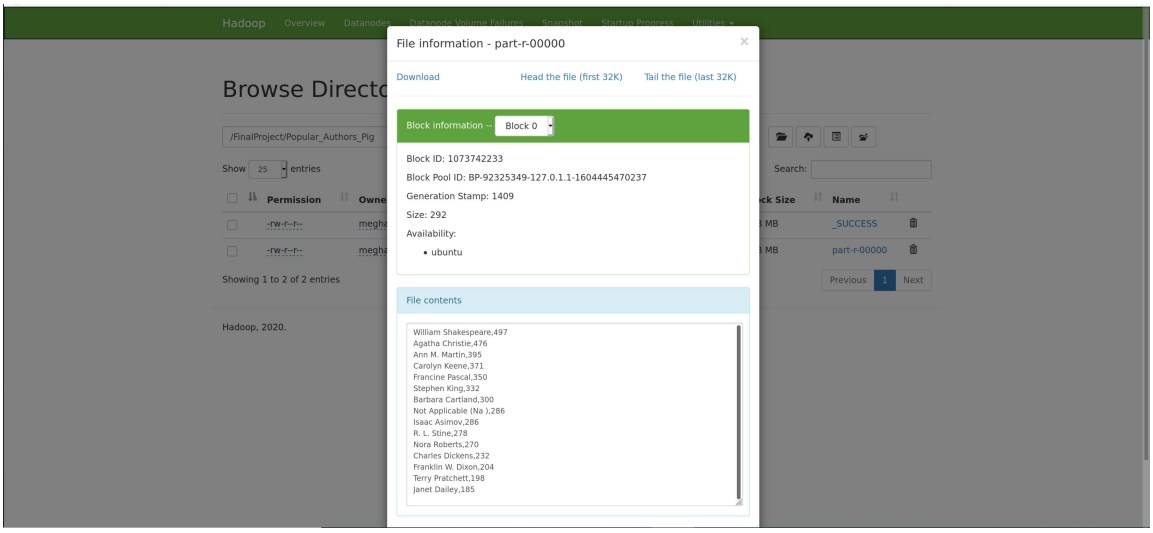
```
// joining ratings and books to display book details
explicitRatedBooks = JOIN books BY isbn LEFT OUTER, bookRating BY isbn;
top25_ratings = LIMIT explicitRatedBooks 25;
DUMP top25_ratings;


//Highest rated books along with their details
order_explicitRatedBooks = ORDER explicitRatedBooks BY average DESC;
top_explicitRatedBooks = LIMIT order_explicitRatedBooks 25;
top25_bookDetails= FOREACH top_explicitRatedBooks generate book_title,book_author,year_of_publication,average;
store top25_bookDetails into 'hdfs://localhost:9000/FinalProject/Top25Rated_BooksDetails_Pig' using PigStorage(',');
DUMP top25_bookDetails;
```

3. Most popular author based on the books count

- **Output from terminal**

```
2020-12-16 21:12:51,470 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-16 21:12:51,471 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
2020-12-16 21:12:51,472 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 21:12:51,482 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 21:12:51,482 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(William Shakespeare,497)
(Agatha Christie,476)
(Ann M. Martin,395)
(Carolyn Keene,371)
(Francine Pascal,350)
(Stephen King,332)
(Barbara Cartland,300)
(Not Applicable (Na ),286)
(Isaac Asimov,286)
(R. L. Stine,278)
(Nora Roberts,270)
(Charles Dickens,232)
(Franklin W. Dixon,204)
(Terry Pratchett,198)
(Janet Dailey,185)
```

- Output from hdfs



CODE

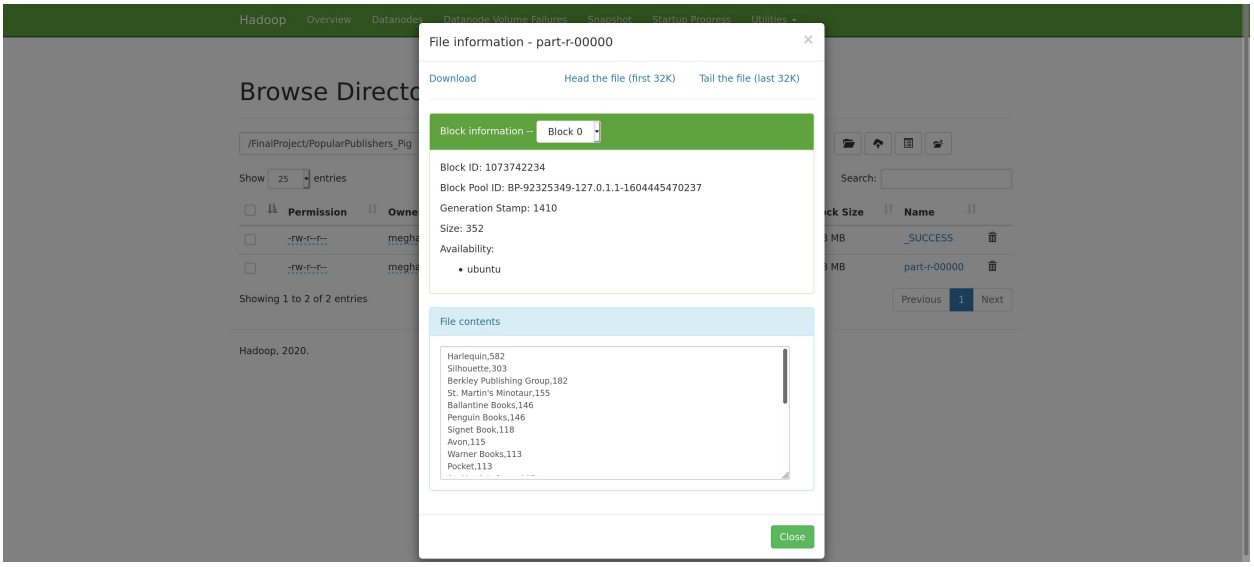
- Grouping books by book_author
- Calculating the count of books
- Sorting the authorBookCount to descending order
- Restricting the rows to 15 by using LIMIT

```
//Most popular author
authorGroup = GROUP books BY book_author;
authorCount = FOREACH authorGroup GENERATE group AS book_author, COUNT($1) AS AuthorBookCount;
DESCRIBE authorCount;
sortAuthors = ORDER authorCount BY authorBookCount DESC;
top_15_authors = LIMIT sortAuthors 15;
DESCRIBE top_15_authors;
store top_15_authors into 'hdfs://localhost:9000/FinalProject/Popular_Authors_Pig' using PigStorage(',');
DUMP top_15_authors;
```

4. **Publishers with maximum book count in the year 2003**
    - Output from terminal

```
2020-12-16 21:32:10,749 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-16 21:32:10,750 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
2020-12-16 21:32:10,751 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 21:32:10,756 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 21:32:10,756 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(Harlequin,582)
(Silhouette,303)
(Berkley Publishing Group,182)
(St. Martin's Minotaur,155)
(Ballantine Books,146)
(Penguin Books,146)
(Signet Book,118)
(Avon,115)
(Warner Books,113)
(Pocket,113)
(St. Martin's Press,107)
(Tor Books,101)
(Goldmann,97)
(Simon & Schuster,95)
(Leisure Books,93)
(HarperCollins,93)
(Zebra Books,89)
(Five Star (ME),86)
(Scholastic Paperbacks,85)
(LÃ?Ã¼bbe,82)
```

Output from HDFS



CODE

- Filtering books by publication year
- Grouping books_2003 by publisher
- Calculating the count
- Sorting the publisherBookCount to descending order
- Restricting the rows to 20 by using LIMIT

```
//Publishers with maximum book count in the year 2003
books_2003 = FILTER books BY year_of_publication == 2003;
publisher_2003 = GROUP books_2003 BY publisher;
publisherGroup = FOREACH publisher_2003 GENERATE group AS publisher, COUNT($1) AS publisherBookCount;
sortData2003 = ORDER publisherGroup BY publisherBookCount DESC;
top_20_2003 = LIMIT sortData2003 20;
store top_20_2003 into 'hdfs://localhost:9000/FinalProject/PopularPublishers_Pig' using PigStorage(',');
DUMP top_20_2003;
```

5. To bin ratings into categories

```
grunt> history;
1   ratings = LOAD '/home/meghana/Downloads/Project/ratings.csv' USING PigStorage(',') AS (user_id:chararray, isbn:chararray, rating:int);
2   SPLIT ratings into worst if(rating>=1 AND rating<=3), bad if(rating>=4 AND rating<=6), good if(rating>=7 AND rating<=9), excellent if(ra
ting==10);
3   worst_all = Group worst All;
4   worst_count = foreach worst_all Generate COUNT (worst.isbn);
```

```
2020-12-16 23:53:33,154 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-16 23:53:33,158 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
2020-12-16 23:53:33,158 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 23:53:33,168 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 23:53:33,168 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(8974)
```

Count of bad ratings

```
2020-12-16 23:56:51,347 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-16 23:56:51,347 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
2020-12-16 23:56:51,348 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 23:56:51,355 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 23:56:51,355 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(84661)
```

Count of good ratings

```
2020-12-16 23:58:22,489 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-16 23:58:22,490 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
2020-12-16 23:58:22,490 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 23:58:22,496 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 23:58:22,497 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(218990)
```

Count of excellent ratings

```
2020-12-16 23:59:53,846 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-16 23:59:53,870 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-16 23:59:53,870 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(71227)
```

Verifying the above count with total count of explicit ratings

```
2020-12-17 00:04:49,667 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-12-17 00:04:49,669 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
2020-12-17 00:04:49,670 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-12-17 00:04:49,680 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2020-12-17 00:04:49,680 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(383852)
```

CODE

- Using SPLIT to divide ratings into categories like worst, bad, good and excellent

```
grunt> history;
1   ratings = LOAD '/home/meghana/Downloads/Project/ratings.csv' USING PigStorage(',') AS (user_id:chararray, isbn:chararray, rating:int);
2   SPLIT ratings into worst if(rating>=1 AND rating<=3), bad if(rating>=4 AND rating<=6), good if(rating>=7 AND rating<=9), excellent if(ra
ting==10);
3   worst_all = Group worst All;
4   worst_count = foreach worst_all Generate COUNT (worst.isbn);
5   bad_all = Group bad All;
6   bad_count = foreach bad_all Generate COUNT (bad.isbn);
7   good_all = Group good All;
8   good_count = foreach good_all Generate COUNT (good.isbn);
9   excellent_all = Group excellent All;
10   excellent_count = foreach excellent_all Generate COUNT (excellent.isbn);
11   explicit = filter ratings by rating!=0;
12   explicit_all = Group explicit All;
13   explicit_count = foreach explicit_all Generate COUNT (explicit.isbn);
```

```
//To categorize ratings into excellent, good, bad and worst
SPLIT ratings into worst if(rating>=1 AND rating<=3), bad if(rating>=4 AND rating<=6), good if(rating>=7 AND rating<=9), excellent
if(rating==10);
```

HIVE

- Creating table and loading the data

```
hive> CREATE TABLE ratings(user_id int,isbn String,rating int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' tblproperties("skip.header.line
.count"="1");
OK
Time taken: 0.069 seconds
```

```
hive> Load Data LOCAL inpath '/home/meghana/Downloads/Project/ratings.csv' OVERWRITE into table ratings;
Loading data to table default.ratings
OK
Time taken: 0.367 seconds
hive> show tables;
OK
filmtable
filmtable2
pokes
ratings
Time taken: 0.051 seconds, Fetched: 4 row(s)
```

```
hive> CREATE TABLE users(user_id int, city String,state String,country String,age int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' tblprop
erties("skip.header.line.count"="1");
OK
Time taken: 0.055 seconds
hive> show tables;
OK
filmtable
filmtable2
pokes
ratings
users
Time taken: 0.039 seconds, Fetched: 5 row(s)
hive> Load Data LOCAL inpath '/home/meghana/Downloads/Project/users.csv' OVERWRITE into table users;
Loading data to table default.users
OK
Time taken: 0.244 seconds
```

```
hive> CREATE TABLE books(isbn String,book_title String,book_author String,year_of_publication int,publisher String) ROW FORMAT DELIMITED FIE
LDS TERMINATED BY ',' tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.066 seconds
hive> Load Data LOCAL inpath '/home/meghana/Downloads/Project/books.csv' OVERWRITE into table books;
Loading data to table default.books
OK
Time taken: 0.27 seconds
hive> show tables;
OK
books
filmtable
filmtable2
pokes
ratings
users
Time taken: 0.043 seconds, Fetched: 6 row(s)
```

```
hive> select count(*) from users;
Query ID = meghana_20201217234224_81782276-f77e-4295-a887-d02490941b78
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1608274458871_0007, Tracking URL = http://ubuntu:8088/proxy/application_1608274458871_0007/
Kill Command = /usr/local/bin/hadoop-3.3.0/bin/mapred job  -kill job_1608274458871_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-12-17 23:42:31,525 Stage-1 map = 0%,  reduce = 0%
2020-12-17 23:42:36,695 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.76 sec
2020-12-17 23:42:40,821 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.37 sec
MapReduce Total cumulative CPU time: 3 seconds 370 msec
Ended Job = job_1608274458871_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.37 sec   HDFS Read: 10115523 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 370 msec
OK
278860
```

```
hive> select count(*) from ratings;
Query ID = meghana_20201217234645_02473528-0bc6-41f1-ac5a-e3b188411dcb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1608274458871_0009, Tracking URL = http://ubuntu:8088/proxy/application_1608274458871_0009/
Kill Command = /usr/local/bin/hadoop-3.3.0/bin/mapred job  -kill job_1608274458871_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-12-17 23:46:52,410 Stage-1 map = 0%,  reduce = 0%
2020-12-17 23:46:57,592 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2020-12-17 23:47:03,815 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.53 sec
MapReduce Total cumulative CPU time: 3 seconds 530 msec
Ended Job = job_1608274458871_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.53 sec   HDFS Read: 21342780 HDFS Write: 107 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 530 msec
OK
1031175
Time taken: 19.13 seconds, Fetched: 1 row(s)
```

```
hive> select count(*) from books;
Query ID = meghana_20201217234551_dd6d42a3-808a-4d91-bcef-361c6f4da275
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1608274458871_0008, Tracking URL = http://ubuntu:8088/proxy/application_1608274458871_0008/
Kill Command = /usr/local/bin/hadoop-3.3.0/bin/mapred job  -kill job_1608274458871_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-12-17 23:45:57,543 Stage-1 map = 0%,  reduce = 0%
2020-12-17 23:46:02,710 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.66 sec
2020-12-17 23:46:08,903 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.21 sec
MapReduce Total cumulative CPU time: 3 seconds 210 msec
Ended Job = job_1608274458871_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.21 sec   HDFS Read: 22252558 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 210 msec
OK
251205
Time taken: 18.49 seconds, Fetched: 1 row(s)
```

CODE

```
CREATE TABLE books(isbn String,book_title String,book_author String,year_of_publication int,publisher String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
Load Data LOCAL inpath '/home/meghana/Downloads/Project/books.csv' OVERWRITE into table books;

CREATE TABLE users(user_id int, city String,state String,country String,age int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

Load Data LOCAL inpath '/home/meghana/Downloads/Project/users.csv' OVERWRITE into table users;

CREATE TABLE ratings(user_id int,isbn String,rating int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

Load Data LOCAL inpath '/home/meghana/Downloads/Project/ratings.csv' OVERWRITE into table ratings;
```

6. To find out number of users per country of age group 16 to 25

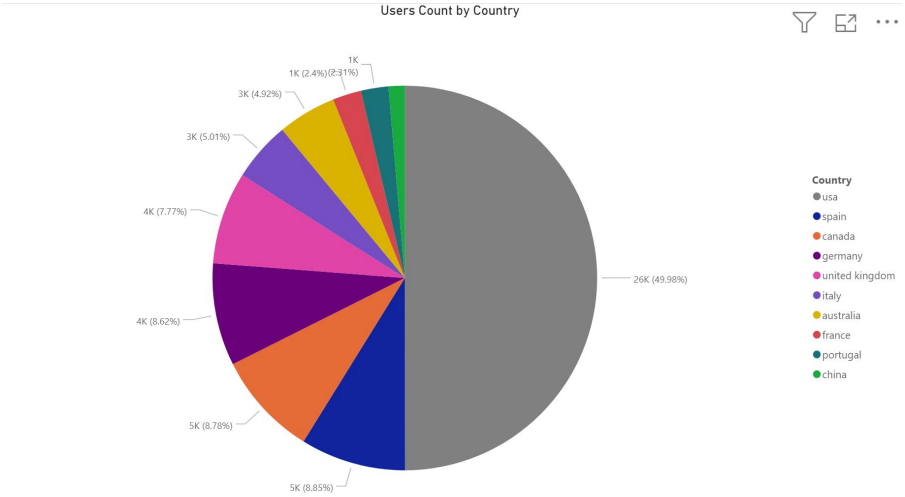
```
hive> select country,count(user_id) as UsersCount from users where age BETWEEN 16 AND 25 GROUP BY country SORT BY UsersCount DESC limit 10;
Query ID = meghana_20201218011210_4ad67d1d-de4e-4877-a8ba-942dcde5f483
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1608274458871_0038, Tracking URL = http://ubuntu:8088/proxy/application_1608274458871_0038/
Kill Command = /usr/local/bin/hadoop-3.3.0/bin/mapred job  -kill job_1608274458871_0038
```

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.08 sec   HDFS Read: 10117351 HDFS Write: 14755 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.55 sec   HDFS Read: 21525 HDFS Write: 374 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.33 sec   HDFS Read: 7844 HDFS Write: 335 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 960 msec
OK
country userscount
usa     25631
spain   4536
canada  4500
germany 4420
united kingdom  3982
italy   2571
australia       2524
france  1231
portugal        1185
china   699
```

CODE

```
select country,count(user_id) as UsersCount from users where age BETWEEN 16 AND 25 GROUP BY country SORT BY UsersCount DESC limit 10;
```

VISUALIZATION



7. To find out number of times each rating has been given for a book

Calculating the count of ratings for each book

```
hive> select isbn,count(rating) as ratingsCount from ratings GROUP BY isbn SORT BY ratingsCount DESC limit 10;
Query ID = meghana_20201218003231_d3fd19ed-d302-48b8-ad31-37085cb9bb94
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

OUTPUT

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.84 sec   HDFS Read: 21342913 HDFS Write: 7153752 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.08 sec   HDFS Read: 7160513 HDFS Write: 406 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.6 sec   HDFS Read: 7873 HDFS Write: 359 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 520 msec
OK
isbn    ratingscount
0971880107      2502
0316666343      1295
0385504209      898
044023722X      838
0312278586      828
0330332775      815
0142001740      774
0060928336      740
0312195516      723
0671027360      670
```

CODE

```
select isbn,count(rating) as ratingsCount from ratings GROUP BY isbn SORT BY ratingsCount DESC limit 10;
```

From the above output choosing isbn 0316666343 for further analysis

```
hive> select rating,count(*) as count, isbn from ratings where isbn=0316666343 group by isbn,rating having rating>0;
Query ID = meghana_20201218153718_4cd2c4af-6058-4b84-900e-98c0cd46ae8e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
```

OUTPUT

```
Total MapReduce CPU Time Spent: 4 seconds 860 msec
OK
1       1       0316666343
2       3       0316666343
3       7       0316666343
4       4       0316666343
5       29      0316666343
6       33      0316666343
7       115     0316666343
8       188     0316666343
9       179     0316666343
10      148     0316666343
```

CODE

```
select rating,count(*) as count, isbn from ratings where isbn=0316666343 group by isbn,rating having rating>0;
```

VISUALIZATION

- From the above visualization it is clear that isbn '0316666343' has got 5.5 rating and many of the users have rated this book by 8

8. To find out details of user who gave highest rating for book

- Using joins, rank() and partition by to find out the details of user who gave highest rating for book

```
hive> select user_id,book_title,rating, book_author,country from (select u.user_id,book_title,r.rating, book_author,country,rank() over(part
ition by book_title order by rating desc) as Rank from books b inner join ratings r on b.isbn=r.isbn inner join users u on r.user_id=u.user_
id )temp where rank=1 limit 5;
No Stats for default@books, Columns: book_title, isbn, book_author
No Stats for default@ratings, Columns: user_id, isbn, rating
No Stats for default@users, Columns: country, user_id
Query ID = meghana_20201218015911_268d5259-fc91-4bb9-b95d-cd47a65d658f
Total jobs = 7
Stage-14 is selected by condition resolver.
Stage-15 is filtered out by condition resolver.
Stage-1 is filtered out by condition resolver.
Execution completed successfully
MapredLocal task succeeded
```

```
user_id book_title      rating  book_author     country
96448    A Light in the Storm: The Civil War Diary of Amelia Martin Fenwick Island Delaware 1861 (Dear America) 9       Karen Hesse     usa
172742   Always Have Popsicles  0       Rebecca Harvin  usa
198711   Apple Magic (The Collector's series)   0       Martina Boudreau        usa
269557   Ask Lily (Young Women of Faith: Lily Series Book 5)    8       Nancy N. Rue    usa
11601    Beyond IBM: Leadership Marketing and Finance for the 1990s     0       Lou Mobley      usa
Time taken: 77.744 seconds, Fetched: 5 row(s)
```

CODE

```
select user_id,book_title,rating, book_author,country from (
    select u.user_id,book_title,r.rating, book_author,country,rank() over(partition by book_title order by rating desc) as Rank
    from books b
    inner join ratings r
    on b.isbn=r.isbn
    inner join users u
    on r.user_id=u.user_id )temp
    where rank=1 limit 5;
```

MAHOUT

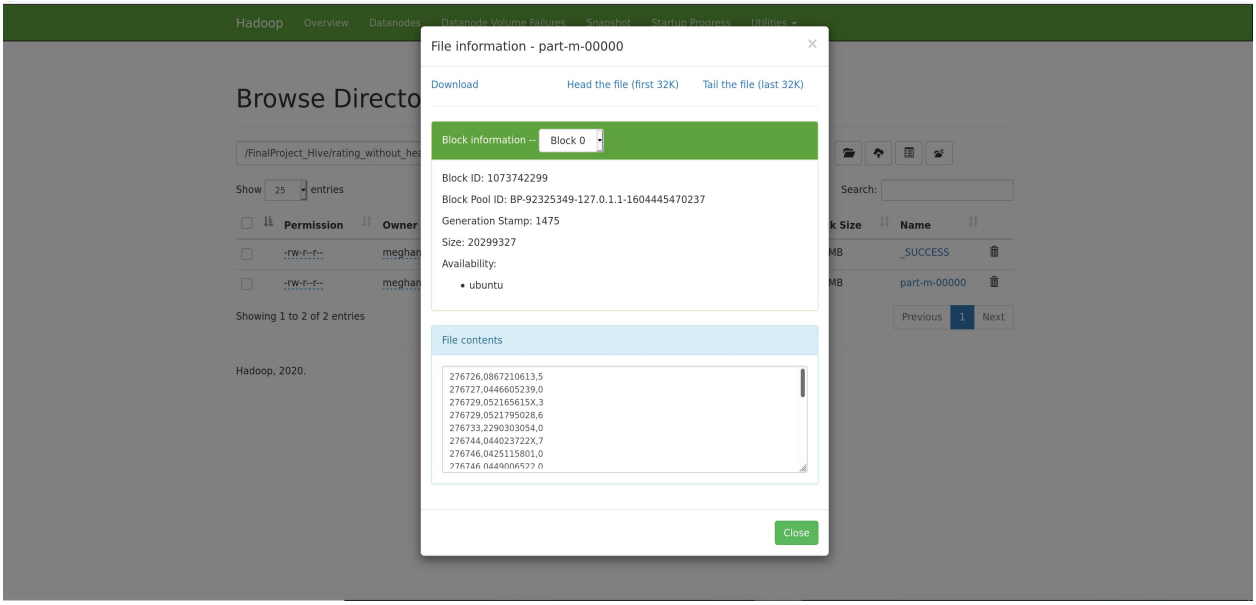
9. Book Recommendation System

- Removing the headers of ratings.csv file using Pig

```
grunt> REGISTER '/home/meghana/Downloads/piggybank-0.17.0.jar';
2020-12-17 15:27:36,043 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
```

```
grunt> rating_wo_headers = LOAD 'hdfs://localhost:9000/FinalProject_Hive/ratings.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage
(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');
2020-12-17 15:33:17,970 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use df
s.bytes-per-checksum
```

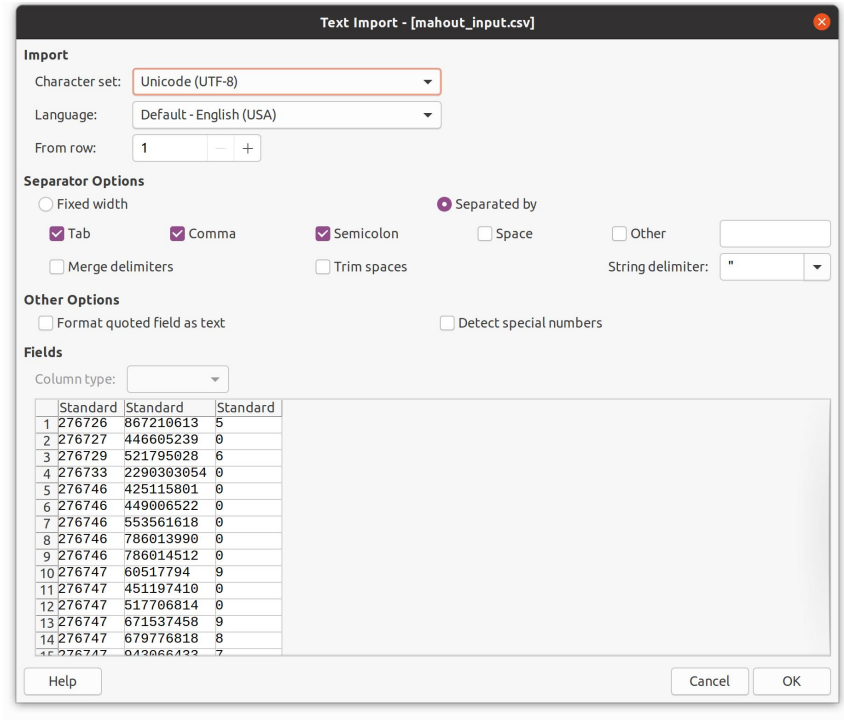
- Loading the file to hdfs



- Copying the file from hdfs to local file system

```
meghana@ubuntu:/usr/local/bin/hadoop-3.3.0/bin$ hadoop fs -copyToLocal hdfs://localhost:9000/FinalProject_Hive/rating_without_headers/part-m-00000 /home/meghana/Downloads/Project/
2020-12-17 15:56:19,761 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

The mapper file obtained is converted to mahout compatable data type and generated file screenshot is added below



RESULT

```
For User Id: 1343
recommended book is: 552998486 with preference value of 9.5
recommended book is: 142001740 with preference value of 8.5
recommended book is: 140063838 with preference value of 7.5
recommended book is: 552998001 with preference value of 5.0
recommended book is: 330244078 with preference value of 4.5
For User Id: 1344
There are no recommendations
For User Id: 1345
There are no recommendations
For User Id: 1348
There are no recommendations
For User Id: 1355
There are no recommendations
For User Id: 1359
There are no recommendations
For User Id: 1361
There are no recommendations
```

CODE

- Converting the data to mahout compatible data type

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class MahoutDataConversion {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader( fileName: "/home/meghana/Downloads/Project/mahout_ratings.csv"));
        BufferedWriter bw = new BufferedWriter(new FileWriter( fileName: "/home/meghana/Downloads/Project/mahout_input.csv"));
        String line;
        while((line = br.readLine()) != null) {
            String[] input = line.split( regex: ",", limit: -1);// -1 specifies no limit
            try{
                bw.write( str: Long.parseLong(input[0]) + "," + Long.parseLong(input[1]) + "," + Long.parseLong(input[2]) + "\n");
                //converted to long data type in order to be compatible with Mahout
            }catch(Exception e){
                continue;
            }
        }
        br.close();
        bw.close();
    }
}
```

Recommender Code

```
import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.impl.common.LongPrimitiveIterator;
import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.impl.neighborhood.NearestNUserNeighborhood;
import org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;
import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;
import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.Recommender;
import org.apache.mahout.cf.taste.similarity.UserSimilarity;
import java.io.File;
import java.io.IOException;
import java.util.List;

public class BookRecommendation {
    public static void main(String[] args) throws IOException, TasteException {
        try {
            DataModel model = new FileDataModel(new File( pathname: "/home/meghana/Downloads/Project/mahout_input.csv"));
            UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
            UserNeighborhood neighborhood = new NearestNUserNeighborhood( n: 2, similarity, model);
            // verifies 2 of the nearest neighbours
            Recommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);
            int count=1;
            // getUserIDs takes first column as userID
            for (LongPrimitiveIterator iterator = model.getUserIDs(); iterator.hasNext(); ) {
                long userId = iterator.nextLong();
                System.out.println("For User Id: " + userId);
                List<RecommendedItem> recommendedItemList = recommender.recommend(userId,  i: 5);
                // Generates five recommendations for each user
                if (recommendedItemList.isEmpty()) {
                    System.out.println("There are no recommendations");
                }
                else {
                    for (RecommendedItem recommendation : recommendedItemList) {
                        System.out.println("recommended book is: " + recommendation.getItemID() + " with preference value of " + recommendation.getValue());
                    }
                    count++;
                    if (count > 147) {
                        break;
                    }
                }
            }
        }
        catch (IOException e) {
            System.out.println("Oops there is an ioexception");
            e.printStackTrace();
        } catch (TasteException e) {
            System.out.println("Oops there is a taste Exception");
            e.printStackTrace();
        }
    }
}
```

pom.xml

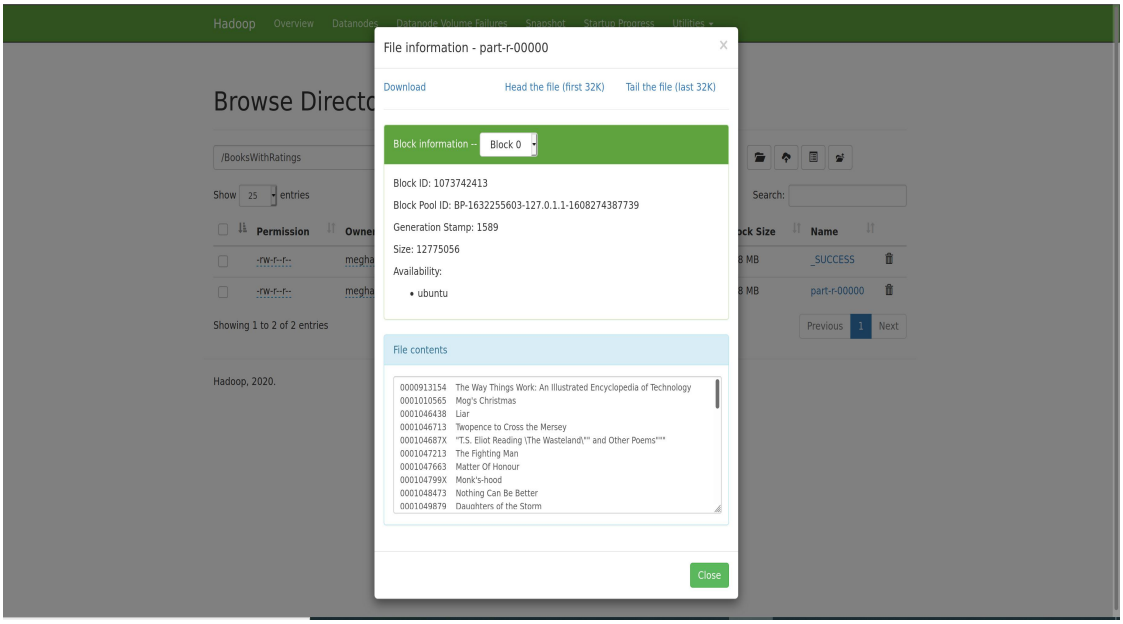
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>
<artifactId>RecommenderSystem</artifactId>
<version>1.0-SNAPSHOT</version>


<dependencies>
    <!-- https://mvnrepository.com/artifact/org.apache.mahout/mahout-core -->
    <dependency>
        <groupId>org.apache.mahout</groupId>
        <artifactId>mahout-core</artifactId>
        <version>0.9</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common -->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.3.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core -->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>3.3.0</version>
    </dependency>
</dependencies>
</project>
```

MAPREDUCE

10. To find out books that have rating(Map Reduce)

```
meghana@ubuntu:/usr/local/bin/hadoop-3.3.0/bin$ hadoop jar /home/meghana/Desktop/IntelliJ_Projects/Analysis-1/target/A_1-1.0-SNAPSHOT.jar Dr
iver /Project/ratings.csv /Project/books.csv /BooksWithRatings
2020-12-18 02:27:40,123 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
2020-12-18 02:27:40,773 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2020-12-18 02:27:41,215 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
2020-12-18 02:27:41,225 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/meghana/.staging/job
_1608274458871_0050
```

- Output from HDFS



CODE

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class RatingsMapper extends Mapper<Object, Text, Text, Text> {
    private Text outkey = new Text();
    private Text outvalue = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String[] input = value.toString().split( regex: ",");

        if (input[0].equals("user_id"))
        {
            return;
        }

        // Exclude zero rating
        if (input[2].equals("0"))
        {
            return;
        }

        // isbn
        outkey.set(input[1]);
        outvalue.set("R"+value.toString());
        context.write(outkey,outvalue);
    }
}
```

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class BooksMapper extends Mapper<Object, Text, Text, Text> {
    private Text outkey = new Text();
    private Text outvalue = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String[] input = value.toString().split( regex: ",");

        if (input[0].equals("isbn"))
        {
            return;
        }
        //isbn
        outkey.set(input[0]);
        outvalue.set("B"+input[1].toString());
        context.write(outkey,outvalue);
    }
}
```

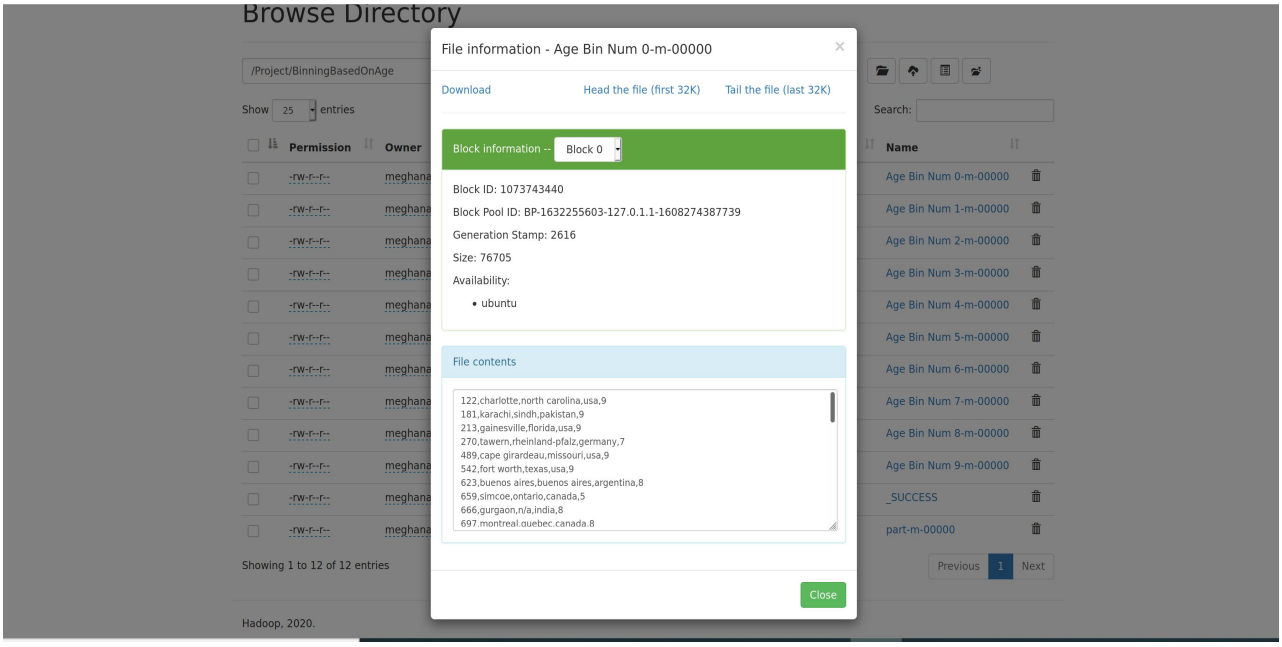
```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

public class BooksRatingsReducer extends Reducer<Text, Text, Text, Text> {
    private boolean rated = false;
    private Text bookTitle = null;
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
        for(Text val : values)
        {
            if(val.charAt(0)=='R')
                rated = true;
            else if(val.charAt(0)=='B') {
                bookTitle = new Text(val.toString().substring(1));
            }
        }

        if (rated)
        {
            context.write(key, bookTitle);
        }
        else
        {
            context.write(new Text(), new Text());
        }
    }
}
```

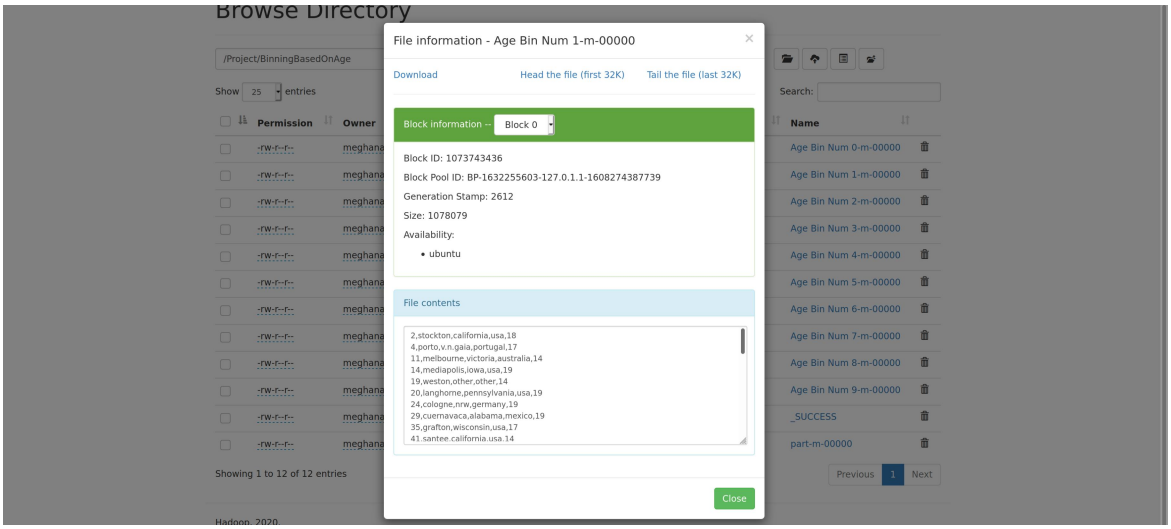
```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Driver {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Driver.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        MultipleInputs.addInputPath(job,new Path(args[0]), TextInputFormat.class,RatingsMapper.class);
        MultipleInputs.addInputPath(job,new Path(args[1]), TextInputFormat.class,BooksMapper.class);
        job.setReducerClass(BooksRatingsReducer.class);

        FileOutputFormat.setOutputPath(job, new Path(args[2]));
        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[2]),  b: true);
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

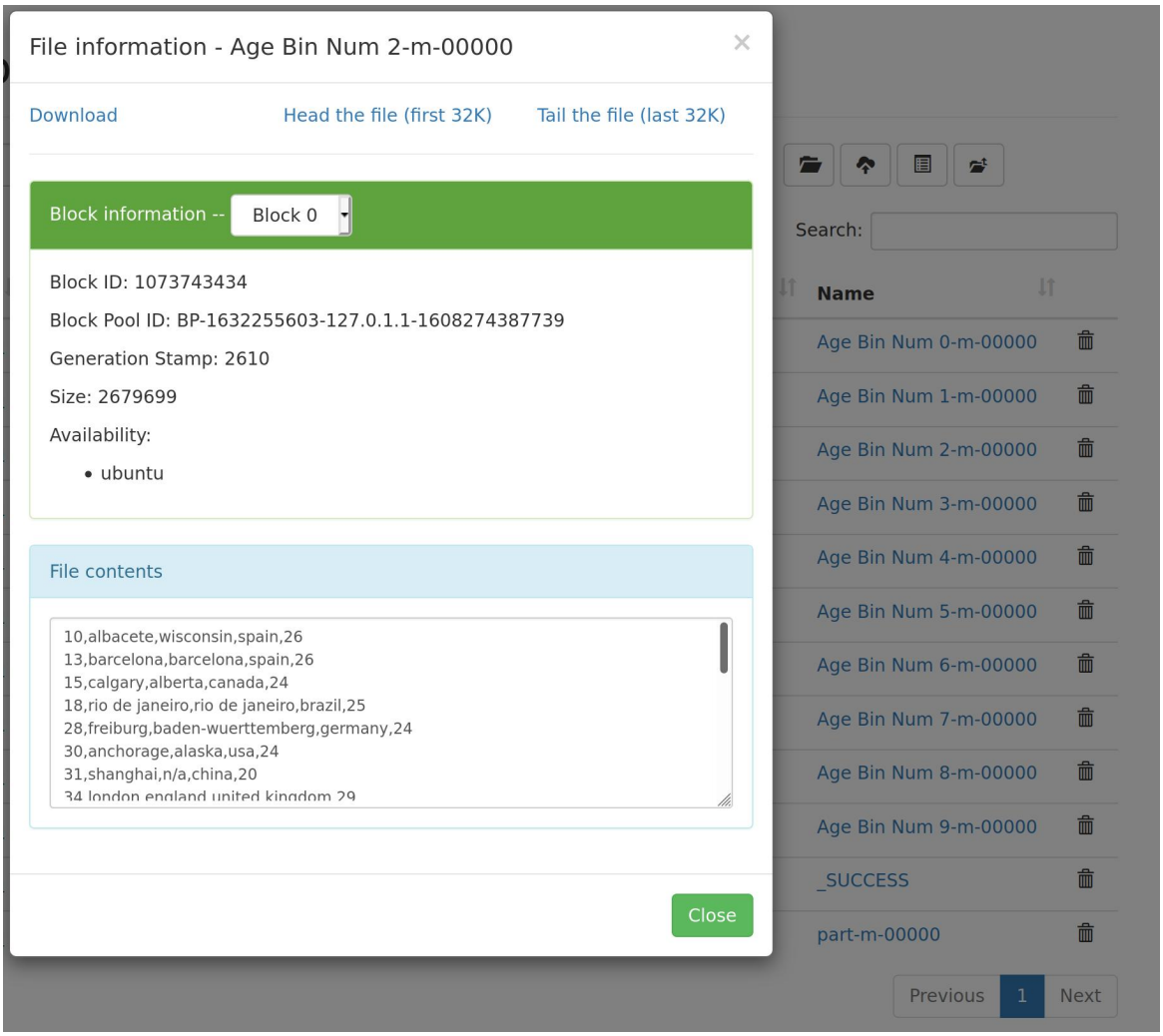
11. Binning users based on age (Organization Pattern)

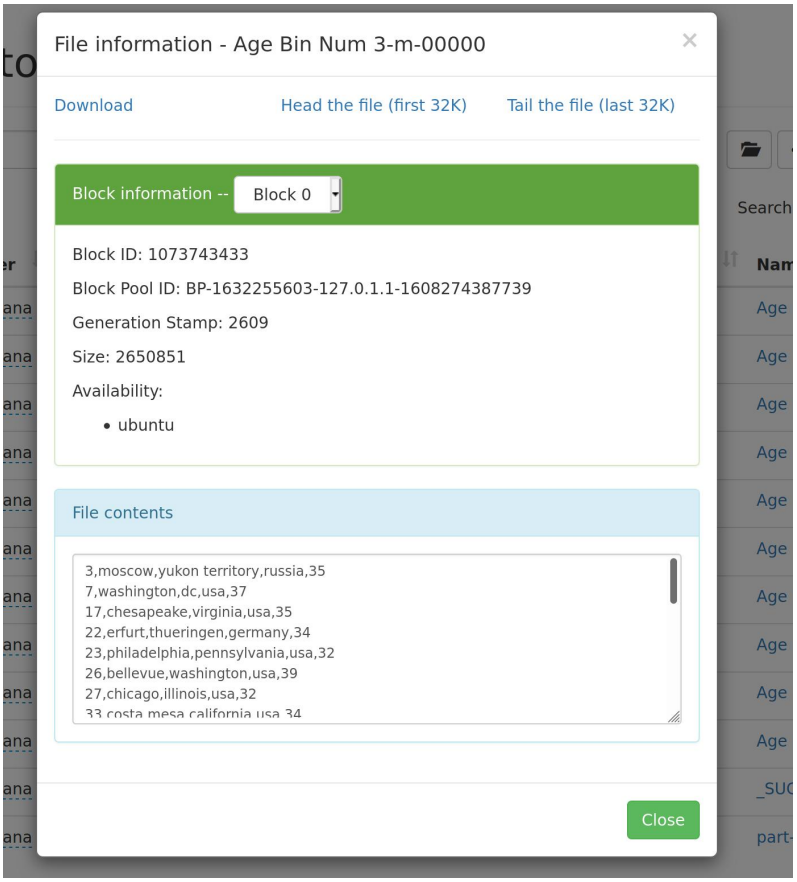
- Since the maximum age of users is 99, I am dividing the users into 10 bins

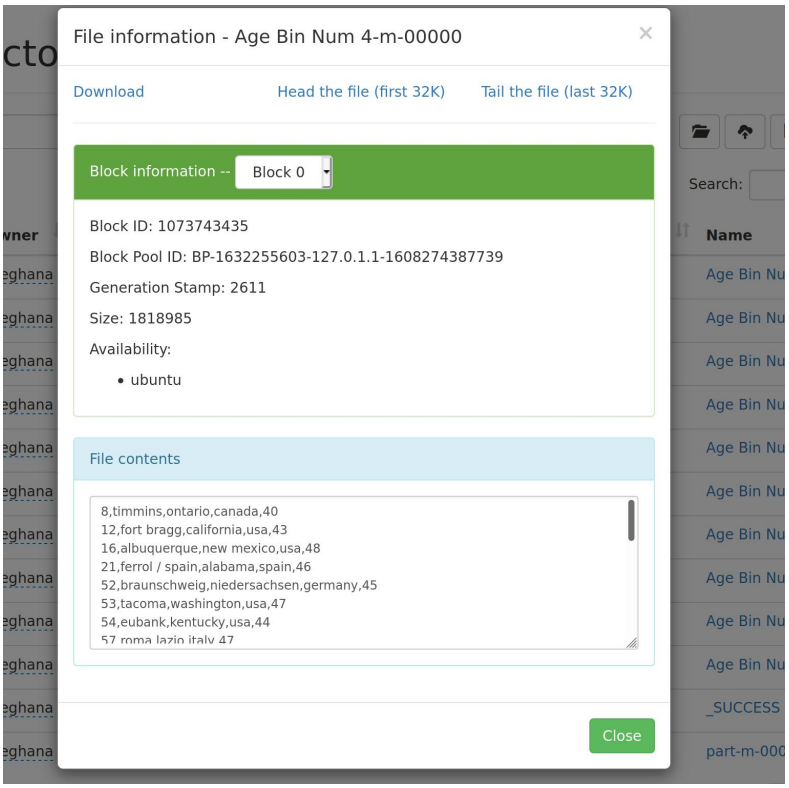
OUTPUT

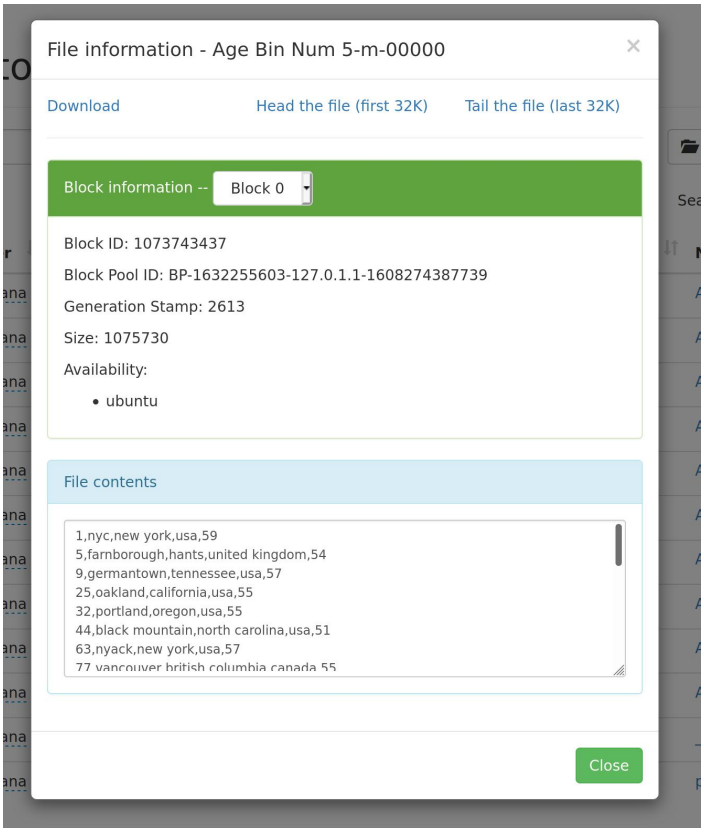


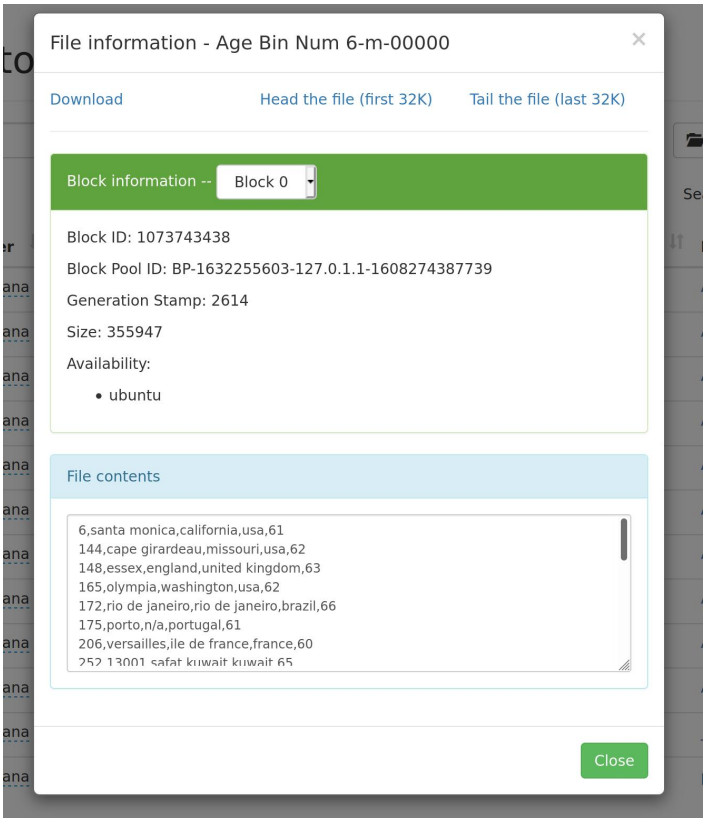

Browse Directory
File information - Age Bin Num 1-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073743436
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2612
Size: 1078079
Availability:
ubuntu
File contents
2,stockton,california,usa,18
4,porto,v.n.gaia,portugal,17
11,melbourne,victoria,australia,14
14,mediapolis,iowa,usa,19
19,weston,other,other,14
20,langhorne,pennsylvania,usa,19
24,cologne,nrw,germany,19
29,cuernavaca,alabama,mexico,19
35,grafton,wisconsin,usa,17
41,santee,california,usa,14
Close
Search:
Name
Age Bin Num 0-m-00000
Age Bin Num 1-m-00000
Age Bin Num 2-m-00000
Age Bin Num 3-m-00000
Age Bin Num 4-m-00000
Age Bin Num 5-m-00000
Age Bin Num 6-m-00000
Age Bin Num 7-m-00000
Age Bin Num 8-m-00000
Age Bin Num 9-m-00000
_SUCCESS
part-m-00000
Previous
1
Next

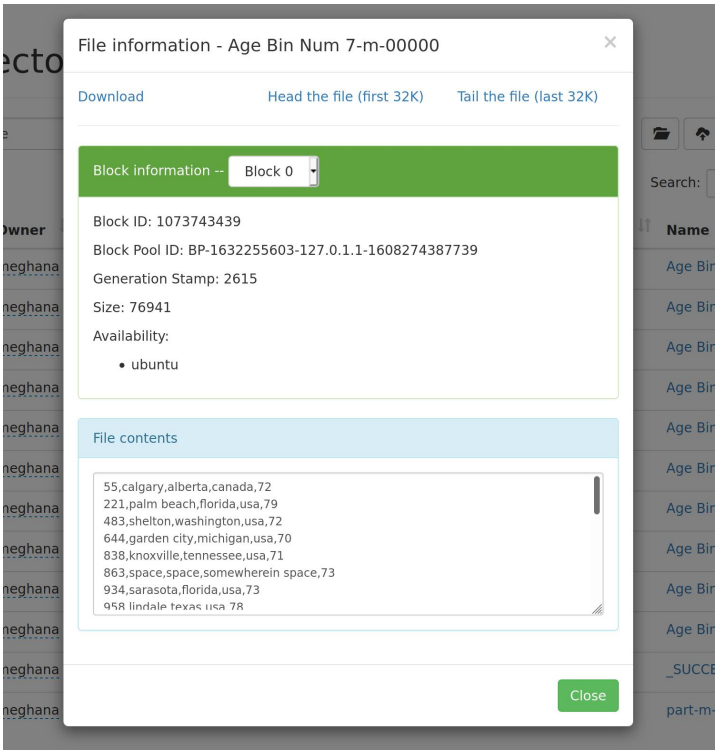

File information - Age Bin Num 2-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073743434
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2610
Size: 2679699
Availability:
ubuntu
File contents
10,albacete,wisconsin,spain,26
13,barcelona,barcelona,spain,26
15,calgary,alberta,canada,24
18,rio de janeiro,rio de janeiro,brazil,25
28,freiburg,baden-wuerttemberg,germany,24
30,anchorage,alaska,usa,24
31,shanghai,n/a,china,20
34,london,england,united kingdom,29
Close
Search:
Name
Age Bin Num 0-m-00000
Age Bin Num 1-m-00000
Age Bin Num 2-m-00000
Age Bin Num 3-m-00000
Age Bin Num 4-m-00000
Age Bin Num 5-m-00000
Age Bin Num 6-m-00000
Age Bin Num 7-m-00000
Age Bin Num 8-m-00000
Age Bin Num 9-m-00000
_SUCCESS
part-m-00000
Previous
1
Next

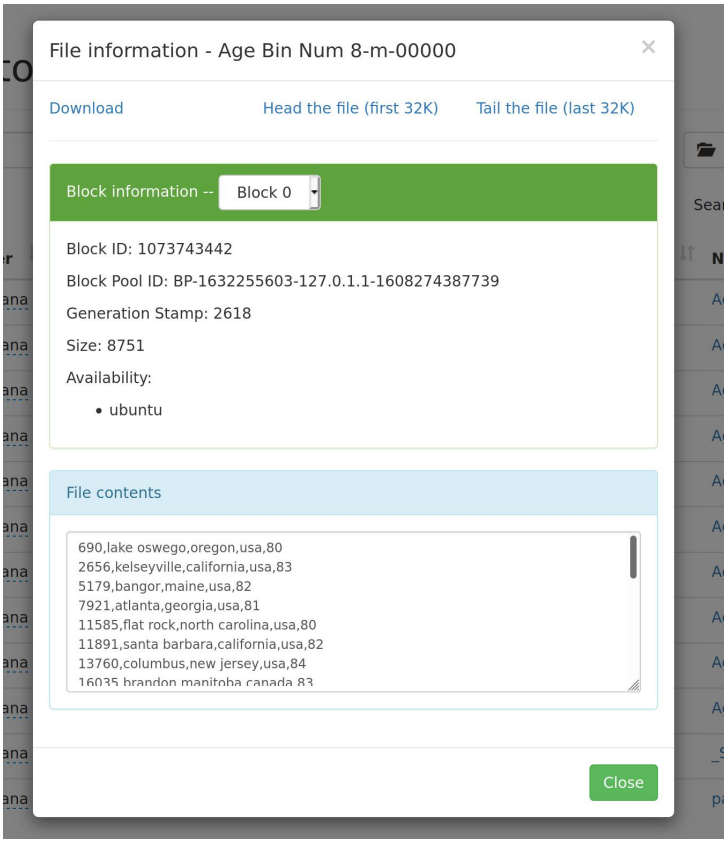

File information - Age Bin Num 3-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073743433
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2609
Size: 2650851
Availability:
ubuntu
File contents
3,moscow,yukon territory,russia,35
7,washington,dc,usa,37
17,chesapeake,virginia,usa,35
22,erfurt,thueringen,germany,34
23,philadelphia,pennsylvania,usa,32
26,bellevue,washington,usa,39
27,chicago,illinois,usa,32
Close

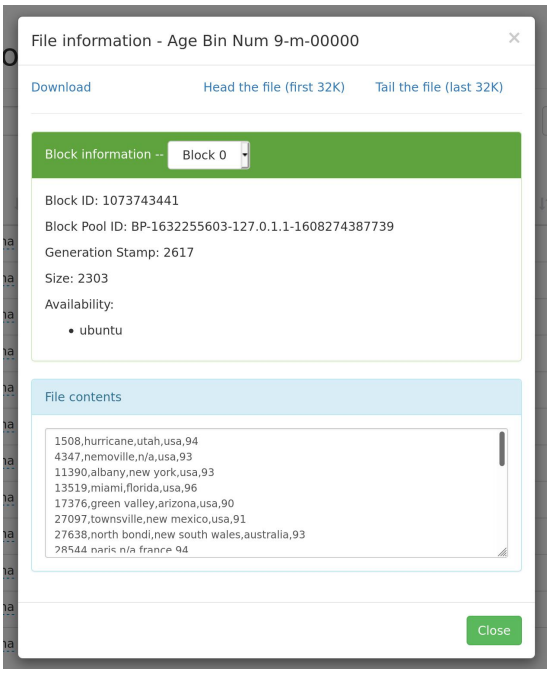

File information - Age Bin Num 4-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073743435
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2611
Size: 1818985
Availability:
ubuntu
File contents
8,timmins,ontario,canada,40
12,fort bragg,california,usa,43
16,albuquerque,new mexico,usa,48
21,ferrol / spain,alabama,spain,46
52,braunschweig,niedersachsen,germany,45
53,tacoma,washington,usa,47
54,eubank,kentucky,usa,44
Close


File information - Age Bin Num 5-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073743437
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2613
Size: 1075730
Availability:
ubuntu
File contents
1,nyc,new york,usa,59
5,farnborough,hants,united kingdom,54
9,germantown,tennessee,usa,57
25,oakland,california,usa,55
32,portland,oregon,usa,55
44,black mountain,north carolina,usa,51
63,nyack,new york,usa,57
77 vancouver british columbia canada 55
Close


File information - Age Bin Num 6-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073743438
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2614
Size: 355947
Availability:
ubuntu
File contents
6,santa monica,california,usa,61
144,cape girardeau,missouri,usa,62
148,essex,england,united kingdom,63
165,olympia,washington,usa,62
172,rio de janeiro,rio de janeiro,brazil,66
175,porto,n/a,portugal,61
206,versailles,ile de france,france,60
252 13001 safat kuwait kuwait 65
Close


File information - Age Bin Num 7-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information --
Block 0
Block ID: 1073743439
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2615
Size: 76941
Availability:
ubuntu
File contents
55,calgary,alberta,canada,72
221,palm beach,florida,usa,79
483,shelton,washington,usa,72
644,garden city,michigan,usa,70
838,knoxville,tennessee,usa,71
863,space,space,somewherein space,73
934,sarasota,florida,usa,73
Close


File information - Age Bin Num 8-m-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information --
Block 0
Block ID: 1073743442
Block Pool ID: BP-1632255603-127.0.1.1-1608274387739
Generation Stamp: 2618
Size: 8751
Availability:
ubuntu
File contents
690,lake oswego,oregon,usa,80
2656,kelseyville,california,usa,83
5179,bangor,maine,usa,82
7921,atlanta,georgia,usa,81
11585,flat rock,north carolina,usa,80
11891,santa barbara,california,usa,82
13760,columbus,new jersey,usa,84
Close



CODE

```java
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;
import java.io.IOException;

public class UserCountryBinMapper extends Mapper<Object, Text, Text, NullWritable> {
    private MultipleOutputs<Text, NullWritable> multipleOutputs = null;
    @Override
    protected void setup(Context context) { multipleOutputs = new MultipleOutputs(context); }

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        try {
            String[] input = value.toString().split( regex: ",");

            if (input[0].equals("user_id")) {
                return;
            }

            int age = Integer.parseInt(input[4]);
            int bin = age / 10;

            multipleOutputs.write( namedOutput: "UserCountryBins", value, NullWritable.get(), baseOutputPath: "Age Bin Num " + bin);
        }
        catch(Exception ex)
        {
        }
    }
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        multipleOutputs.close();
    }
}
```

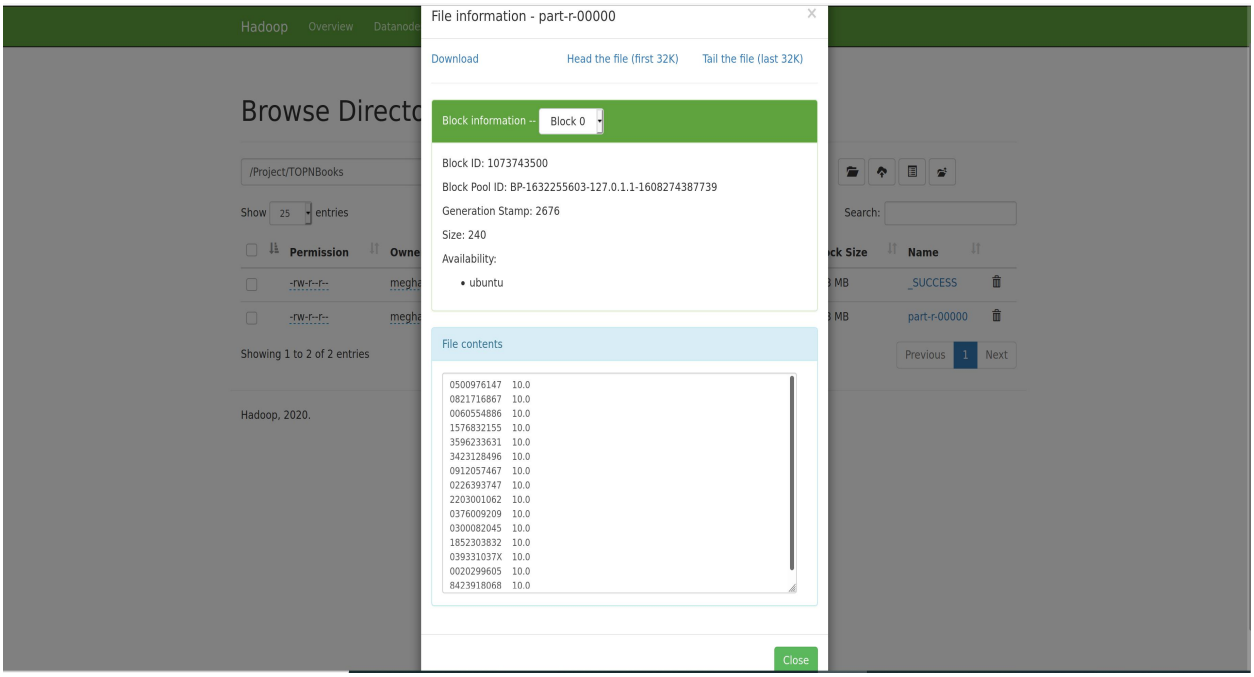
```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.io.IOException;

public class Driver {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Driver.class);
        job.setMapperClass(UserCountryBinMapper.class);
        job.setNumReduceTasks(0);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(NullWritable.class);
        TextInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[1]), b: true);
        MultipleOutputs.addNamedOutput(job, namedOutput: "UserCountryBins", TextOutputFormat.class,
                Text.class, NullWritable.class);
        MultipleOutputs.setCountersEnabled(job, enabled: true);

        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

12. Highest rated books(TOP N FILTERING PATTERN)

```
meghana@ubuntu:/usr/local/bin/hadoop-3.3.0/bin$ hadoop jar /home/meghana/Desktop/IntelliJ_Projects/Analysis-3/target/A_3-1.0-SNAPSHOT.jar To
pDriver /Project/ratings.csv /Project/TOPNBooks 15
2020-12-18 12:43:52,249 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
2020-12-18 12:43:52,833 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2020-12-18 12:43:53,230 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
2020-12-18 12:43:53,283 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/meghana/.staging/job
_1608304187440_0004
2020-12-18 12:43:53,507 INFO input.FileInputFormat: Total input files to process : 1
2020-12-18 12:43:53,584 INFO mapreduce.JobSubmitter: number of splits:1
2020-12-18 12:43:53,507 INFO input.FileInputFormat: Total input files to process : 1
2020-12-18 12:43:53,584 INFO mapreduce.JobSubmitter: number of splits:1
2020-12-18 12:43:53,722 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1608304187440_0004
2020-12-18 12:43:53,723 INFO mapreduce.JobSubmitter: Executing with tokens: []
```

- Number of books needed can be specified from terminal as an argument

- Output from HDFS



CODE

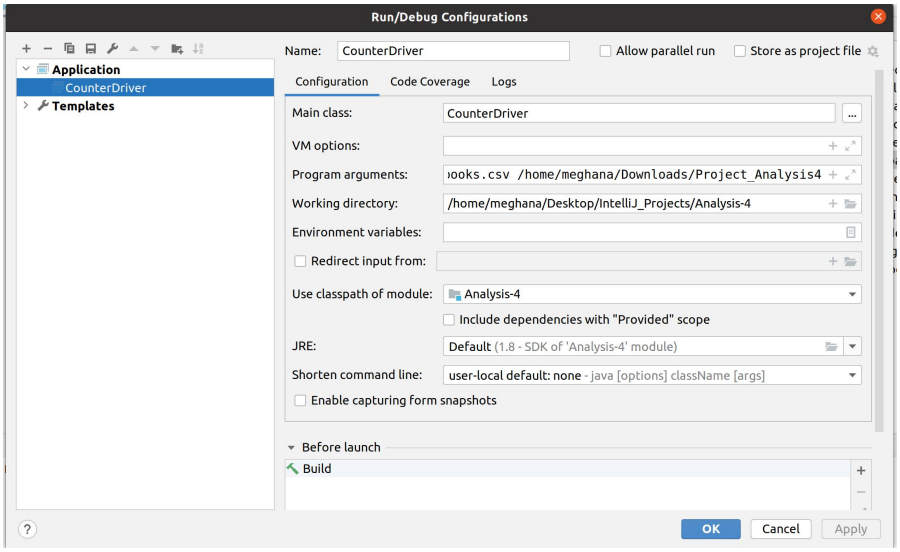
```
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class RatingsMapper extends Mapper<Object, Text, Text, FloatWritable> {
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        try {
            String[] tokens = value.toString().split( regex: ",");

            if (tokens[0].equals("user_id")) {
                return;
            }
            Float rating = Float.parseFloat(tokens[2]);
            // <Isbn, rating>
            context.write(new Text(tokens[1].toString()), new FloatWritable(rating));
        }
        catch(Exception e){
        }
    }
}
```

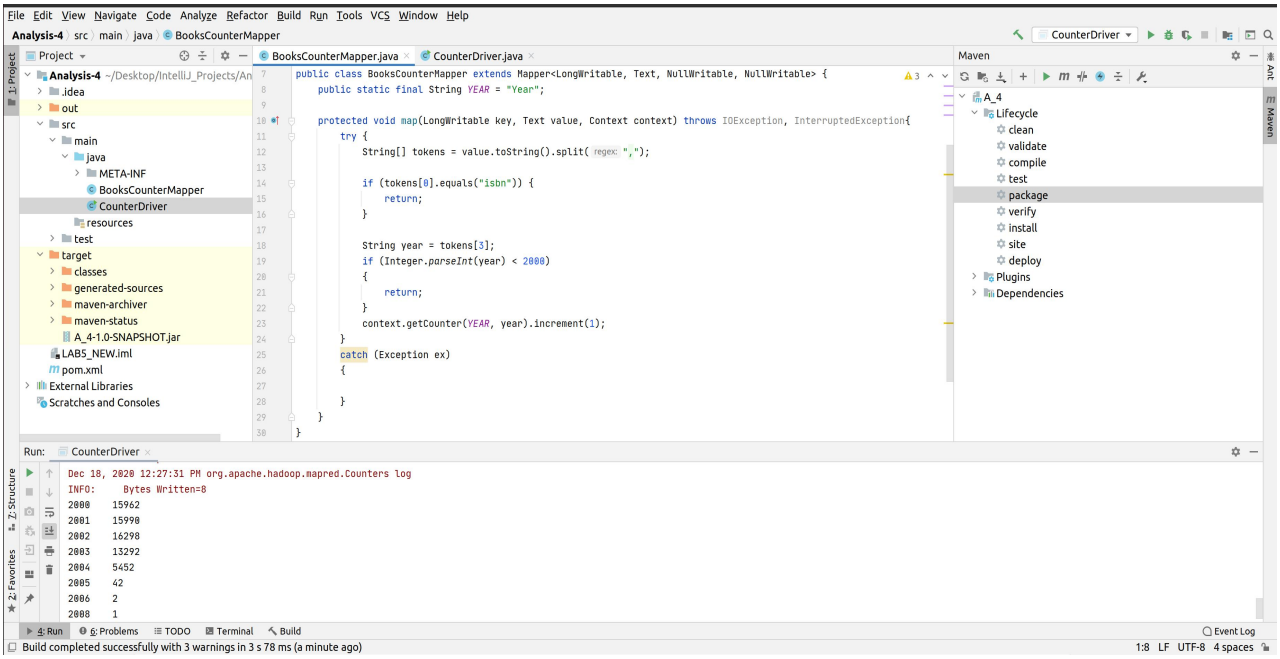
```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.stream.Collectors;

public class TopReducer extends Reducer<Text, FloatWritable, Text, FloatWritable> {
    private Map<String, Float> isbnRating;

    protected void setup(Context context)
            throws IOException,
            InterruptedException
    {
        isbnRating = new HashMap<String, Float>();
    }

    protected void reduce(Text key, Iterable<FloatWritable> values, Context context)
            throws IOException,
            InterruptedException
    {
        float totalRating = 0;
        int count = 0;
        for (FloatWritable val : values)
        {
            totalRating += val.get();
            count++;
        }
        // Average rating of each isbn is stored into map
        isbnRating.put(key.toString(), totalRating/count);
    }


    protected void cleanup(Context context)
            throws IOException,
            InterruptedException
    {
        Configuration config = context.getConfiguration();
        int topN = Integer.parseInt(config.get( name: "topNBooks",  defaultValue: "20"));
        // sorting hashmap based on values
        Map<String,Float> sorted = isbnRating
                .entrySet() Set<Map<K, V>.Entry<String, Float>>
                .stream() Stream<Map<K, V>.Entry<String, Float>>
                .sorted(Map.Entry.<~> comparingByValue().reversed())
                .limit(topN)
                .collect(Collectors.toMap(
                        Map.Entry::getKey, Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::new));

        sorted.forEach((key,value) -> {
            try {
                context.write(new Text(key), new FloatWritable(value));
            } catch (IOException e) {
                e.printStackTrace();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });
    }
}
```

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class TopDriver {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        conf.set("topNBooks",args[2]);
        Job job = Job.getInstance(conf);
        job.setJarByClass(TopDriver.class);
        job.setNumReduceTasks(1);// So that we get top books wrt average rating
        job.setMapperClass(RatingsMapper.class);
        job.setReducerClass(TopReducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(FloatWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(FloatWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[1]), b: true);
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

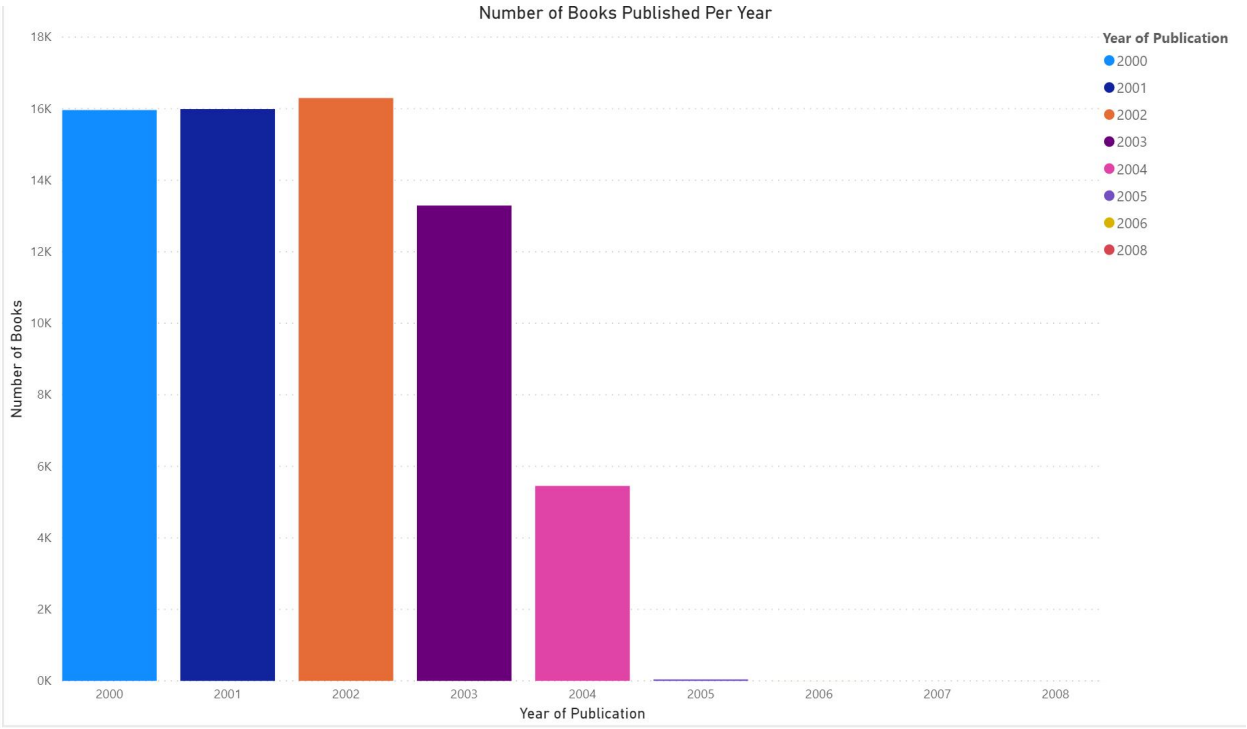
13. To find out number of books published each year(counting with counters)

OUTPUT





VISUALIZATION



CODE

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class BooksCounterMapper extends Mapper<LongWritable, Text, NullWritable, NullWritable> {
    public static final String YEAR = "Year";

    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException{
        try {
            String[] tokens = value.toString().split( regex: ",");

            if (tokens[0].equals("isbn")) {
                return;
            }

            String year = tokens[3];
            if (Integer.parseInt(year) < 2000)
            {
                return;
            }
            context.getCounter(YEAR, year).increment(1);
        }
        catch (Exception ex)
        {

        }
    }
}
```

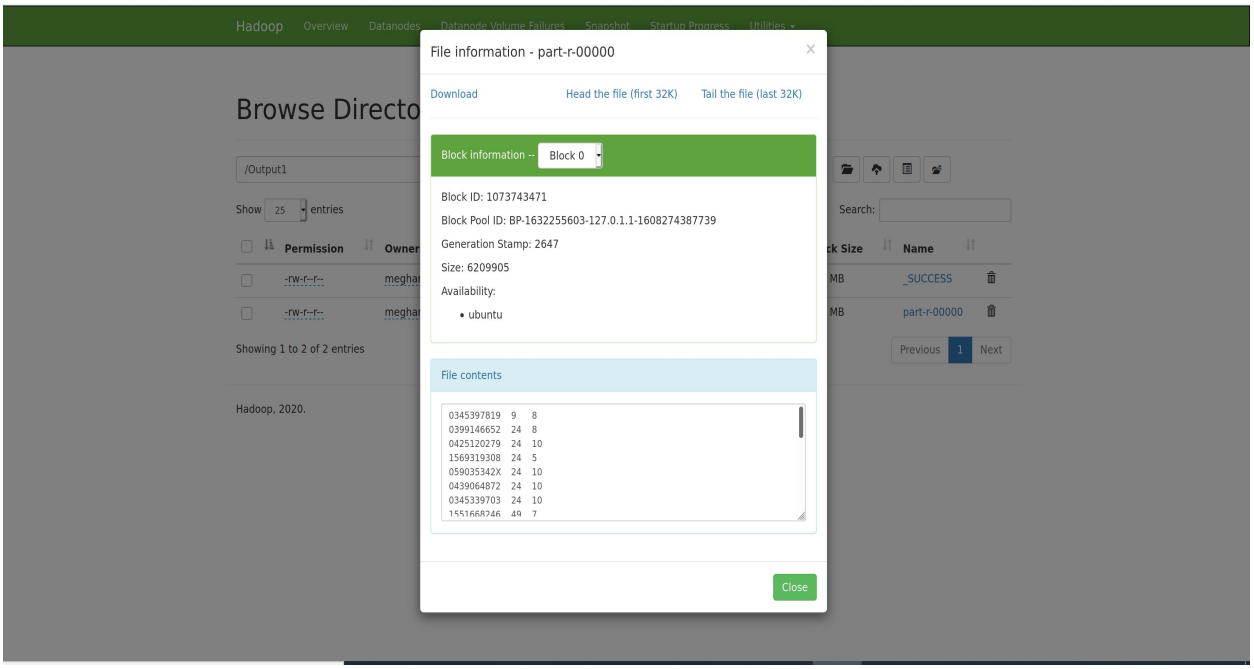
```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;

public class CounterDriver {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(CounterDriver.class);
        job.setMapperClass(BooksCounterMapper.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[1]), b: true);
        int code = job.waitForCompletion( verbose: true) ? 0 : 1;
        if(code==0)
        {
            for(Counter counter: job.getCounters().getGroup(BooksCounterMapper.YEAR)){
                System.out.println(counter.getDisplayName()+"\t"+counter.getValue());
            }
        }
        fs.get(conf).delete(new Path(args[1]), b: true);
        System.exit(code);
    }
}
```

14. To find out list of ratings in descending order, different age groups have given for books <u>(Secondary Sorting and Chaining)</u>

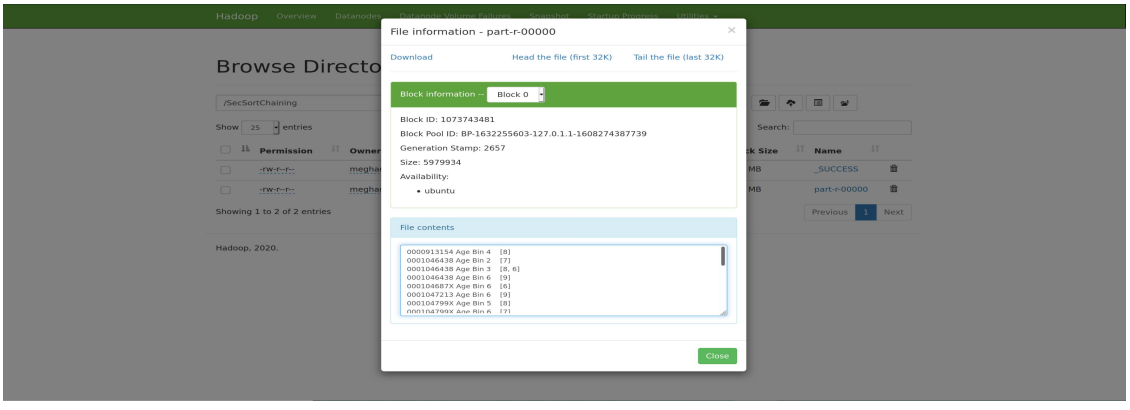
- Running the map reduce program using hadoop jar path

```
meghana@ubuntu:/usr/local/bin/hadoop-3.3.0/bin$ hadoop jar /home/meghana/Desktop/IntelliJ_Projects/Analysis-5/target/A_5-1.0-SNAPSHOT.jar Dr
iver /Project/ratings.csv /Project/users.csv /Output1 /SecSortChaining
2020-12-18 11:10:24,219 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
2020-12-18 11:10:24,861 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2020-12-18 11:10:25,327 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
2020-12-18 11:10:25,407 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/meghana/.staging/job
_1608304187440_0001
2020-12-18 11:10:26,386 INFO input.FileInputFormat: Total input files to process : 1
2020-12-18 11:10:26,429 INFO input.FileInputFormat: Total input files to process : 1
2020-12-18 11:10:26,507 INFO mapreduce.JobSubmitter: number of splits:2
```

- Output from HDFS



- Final Output after chaining and secondary sorting



CODE

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class UsersMapper extends Mapper<Object, Text, Text, Text> {
    private Text outkey = new Text();
    private Text outvalue = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String[] input = value.toString().split( regex: ",");

        if (input[0].equals("user_id"))
        {
            return;
        }

        outkey.set(input[0]);
        outvalue.set("U"+value.toString());//to indicate that it is coming from users table 'U' is used
        context.write(outkey, outvalue);
    }
}
```

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class RatingsMapper extends Mapper<Object, Text, Text, Text> {
    private Text outkey = new Text();
    private Text outvalue = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String[] input = value.toString().split( regex: ",");

        if (input[0].equals("user_id"))
        {
            return;
        }

        // Exclude zero rating
        if (input[2].equals("0"))
        {
            return;
        }

        // user_id as the key
        outkey.set(input[0]);
        outvalue.set("R"+value.toString());
        context.write(outkey,outvalue);
    }
}
```

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
public class UsersRatingsReducer extends Reducer<Text, Text, Text, Text> {
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
        try {
            String age = "";
            List<String> ratings = new ArrayList<String>();
            for (Text val : values) {
                if (val.charAt(0) == 'U') {
                    age = val.toString().substring(1).split( regex: ",")[4];
                    continue;
                }

                ratings.add(val.toString().substring(1));
            }
            if (age.isEmpty()) {
                //context.write(new Text(), new Text());
                return;
            }
            for (String rating : ratings) {
                    String[] input = rating.split( regex: ",");
                    context.write(new Text( string: input[1] + "\t" + age), new Text(input[2]));
            }
            }
        catch(Exception ex)
        {
        }
    }
}
```

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class SecUserBookRatingMapper extends Mapper<LongWritable, Text, UserBookRatingKey, Text> {
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException{
        String line  = value.toString();
        String[] tokens = line.split( regex: "\t");
        String isbn = null;
        String age = null;
        String rating = null;

        try {
            isbn = tokens[0];
            age = tokens[1];
            rating = tokens[2];
        }

        catch (Exception e)
        {

        }

        if(isbn!=null && age!=null && rating != null)
        {
            UserBookRatingKey uk = new UserBookRatingKey(isbn, age, rating);
            context.write(uk, new Text(rating));
        }

    }

}
```

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class SecUserBookRatingReducer extends Reducer<UserBookRatingKey, Text, Text, Text> {
// sorting ratings and writing the output
    protected void reduce(UserBookRatingKey key, Iterable<Text> values, Context context)
            throws IOException,
            InterruptedException
    {
        List<Integer> ratings = new ArrayList<~>();
       for(Text v: values)
       {
           ratings.add(Integer.parseInt(v.toString()));
       }

        Collections.sort(ratings, Collections.reverseOrder());
        String output = Arrays.toString(ratings.toArray());

        int ageBin = Integer.parseInt(key.getAge())/10;
       context.write(new Text( string: key.getIsbn() + " Age Bin " + ageBin),new Text(output));
    }
}
```

```java
import org.apache.hadoop.io.WritableComparable;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class UserBookRatingKey implements WritableComparable<UserBookRatingKey> {
    private String isbn;
    private String rating;
    private String age;
    public UserBookRatingKey() { super(); }
    public String getIsbn() { return isbn.trim(); }
    public void setIsbn(String isbn) { this.isbn = isbn; }
    public String getRating() { return rating; }
    public void setRating(String rating) { this.rating = rating; }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public UserBookRatingKey(String isbn, String age, String rating)
    {
        super();
        this.isbn=isbn;
        this.rating=rating;
        this.age=age;
    }
```

```
    public int compareTo(UserBookRatingKey o) {
        int result = this.isbn.compareTo(o.isbn);
        if(result==0){
            int result1 = this.rating.compareTo(o.rating);
            if (result1 == 0)
            {
                return this.age.compareTo(o.age);
            }
        }
        return result;
    }

    public void write(DataOutput out) throws IOException {
        out.writeUTF(isbn);
        out.writeUTF(rating);
        out.writeUTF(age);
    }

    public void readFields(DataInput in) throws IOException {
        isbn=in.readUTF();
        rating=in.readUTF();
        age = in.readUTF();
    }

    @Override
    public String toString()
    {
        return isbn + "," +rating+ ","+age;
    }
}


import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class SecSortGroupComparator extends WritableComparator {
    public SecSortGroupComparator() { super(UserBookRatingKey.class, createInstances: true); }
//if isbn and age are same we are telling framework to consider it as same key
    @Override
    public int compare(WritableComparable a, WritableComparable b)
    {
        UserBookRatingKey uk1 = (UserBookRatingKey)a;
        UserBookRatingKey uk2 = (UserBookRatingKey)b;

        int result = uk1.getIsbn().compareTo(uk2.getIsbn());

        if(result==0)
        {
            Integer age1Bin = Integer.parseInt(uk1.getAge())/10;
            Integer age2Bin = Integer.parseInt(uk2.getAge())/10;

            return age1Bin.compareTo(age2Bin);
        }

        return result;
    }

}
```

```java
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class SecSortComparator extends WritableComparator {
    public SecSortComparator() { super(UserBookRatingKey.class,  createInstances: true); }
    //sort output from mapper based on isbn and age
    public int compare(WritableComparable a, WritableComparable b)
    {
        UserBookRatingKey uk1 = (UserBookRatingKey)a;
        UserBookRatingKey uk2 = (UserBookRatingKey)b;

        int result = uk1.getIsbn().compareTo(uk2.getIsbn());

        if(result==0)
        {
            Integer age1Bin = Integer.parseInt(uk1.getAge())/10;
            Integer age2Bin = Integer.parseInt(uk2.getAge())/10;

            return age1Bin.compareTo(age2Bin);
            /*
            if (result1 == 0)
            {
                Integer rating1 = Integer.parseInt(uk1.getRating());
                Integer rating2 = Integer.parseInt(uk2.getRating());

                return -1*rating1.compareTo(rating2);
            }*/
        }

        return result;
    }
}
```

```java
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Partitioner;

public class KeyPartitioner extends Partitioner<UserBookRatingKey, Text> {

    @Override
    public int getPartition(UserBookRatingKey key, Text value, int numPartitions){
        return key.getIsbn().hashCode()%numPartitions;
    }
}
```

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
//import org.apache.hadoop.mapreduce.lib.input.CombineTextInputFormat;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.io.IOException;

public class Driver {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Driver.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        MultipleInputs.addInputPath(job,new Path(args[0]), TextInputFormat.class,RatingsMapper.class);
        MultipleInputs.addInputPath(job,new Path(args[1]), TextInputFormat.class,UsersMapper.class);
        job.setReducerClass(UsersRatingsReducer.class);
        FileOutputFormat.setOutputPath(job, new Path(args[2]));
        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[2]), b: true);
        boolean result = job.waitForCompletion( verbose: true);


        if(result)
        {
            Job job1 = Job.getInstance();
            job1.setJarByClass(Driver.class);
            job1.setMapperClass(SecUserBookRatingMapper.class);
            job1.setReducerClass(SecUserBookRatingReducer.class);
            job1.setMapOutputKeyClass(UserBookRatingKey.class);
            job1.setMapOutputValueClass(Text.class);
            job1.setOutputKeyClass(Text.class);
            job1.setOutputValueClass(Text.class);
            TextInputFormat.addInputPath(job1, new Path(args[2]));
            Path finalOutputDirectory = new Path(args[3]);
            TextOutputFormat.setOutputPath(job1, finalOutputDirectory);
            FileSystem fs2 = FileSystem.get(job1.getConfiguration());

            if(fs2.exists(finalOutputDirectory))
            {
                fs2.delete(finalOutputDirectory, b: true);
            }

            job1.setGroupingComparatorClass(SecSortGroupComparator.class);
            job1.setSortComparatorClass(SecSortComparator.class);
            job1.setPartitionerClass(KeyPartitioner.class);
            job1.waitForCompletion( verbose: true);

        }
    }
}
```

APPENDIX

All the source code is added under respective analysis