POC - Analyzing Book- Crossing Data

Contents

[Data Set 2](#_Toc393680017)

[BX-Users: 2](#_Toc393680018)

[BX-Books: 2](#_Toc393680019)

[BX-Book-Ratings: 2](#_Toc393680020)

[Problem 2](#_Toc393680021)

[Solution 2](#_Toc393680022)

[Find out the frequency of books published each year. 2](#_Toc393680023)

[Language used 2](#_Toc393680024)

[Source Code 2](#_Toc393680025)

[Results 6](#_Toc393680026)

[Find out in which year maximum number of books were published 6](#_Toc393680027)

[Language used 6](#_Toc393680028)

[Source Code 6](#_Toc393680029)

[Results 7](#_Toc393680030)

[Find out how many book were published based on ranking in the year 2002. 7](#_Toc393680031)

[Language used 7](#_Toc393680032)

[Source Code 7](#_Toc393680033)

[Results 8](#_Toc393680034)

# Data Set

The Book-Crossing dataset consists of 3 tables.

## BX-Users:

This file contains the list of the users, their age and where they are collected. If that data is unavailable for any field then it is filled with NULL.

## BX-Books:

It gives us the details about the book such as Book-Title, Book-Author, Year-Of-Publication, Publisher, Image-URL and ISBN. Here ISBN will act as a unique code for a book. Invalid ISBNs have already been removed from the dataset. URLs linking to cover images are also given, appearing in three different flavors (`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:

It contains the book rating information. Ratings are either explicitly expressed on a scale from 1-10 (higher values denoting higher appreciation) or implicitly expressed by 0.

# Problem

* Find out the frequency of books published each year. (Hint: Use Boooks.csv file for this)
* Find out in which year maximum number of books were published
* Find out how many book were published based on ranking in the year 2002.­ ( Hint: Use Book.csv and Book-Ratings.csv)

# Solution

## Find out the frequency of books published each year.

### Language used

Java

### Source Code

#### BookPublishFreqDriver.java

**public** **class** BookPublishFreqDriver {

**public** **static** **void** main(String[] args) **throws** IOException {

// **TODO** Auto-generated method stub

//Creating a JobConf object and assigning a job name for identification purposes

JobConf conf = **new** JobConf(BookPublishFreqDriver.**class**);

conf.setJobName("BookPublishFreq");

// Setting configuration object with the Data Type of output Key and

// Value

conf.setOutputKeyClass(Text.**class**);

conf.setOutputValueClass(IntWritable.**class**);

// Providing the mapper and reducer class names

conf.setMapperClass(BookPublishFreqMap.**class**);

conf.setReducerClass(BookPublishFreqReducer.**class**);

// Setting format of input and output

conf.setInputFormat(TextInputFormat.**class**);

conf.setOutputFormat(TextOutputFormat.**class**);

// The hdfs input and output directory to be fetched from the command

// line

FileInputFormat.*setInputPaths*(conf, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(conf, **new** Path(args[1]));

// Running the job

JobClient.*runJob*(conf);

}

#### BookPublishFreqMap.java

**public** **class** BookPublishFreqMap **extends** MapReduceBase **implements**

Mapper<LongWritable, Text, Text, IntWritable> {

**private** Text bookYear = **new** Text();

@Override

**public** **void** map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)

**throws** IOException {

//Converting the record (single line) to String and storing it in a String variable line

String line = value.toString();

//StringTokenizer is breaking the record (line) into words

StringTokenizer tokenizer = **new** StringTokenizer(line, ";");

**while** (tokenizer.hasMoreTokens()) {

tokenizer.nextElement();

tokenizer.nextElement();

tokenizer.nextElement();

String yearData = tokenizer.nextElement().toString();

**int** year = 0;

// In some cases parts[2] is empty filed and no year.

**try** {

**if**(!yearData.isEmpty())

{

String temp = yearData.replace("\"", "");

bookYear.set(temp + ',');

**try** {

year = Integer.*parseInt*(temp);

}

**catch** (NumberFormatException e) {

tokenizer.nextElement();

yearData = tokenizer.nextElement().toString();

}

}

**else**

{

tokenizer.nextElement();

yearData = tokenizer.nextElement().toString();

String temp = yearData.replace("\"", "");

bookYear.set(temp + ',');

year = Integer.*parseInt*(temp);

}

// make sure year is valid #

**if**(year > 1800)

{

output.collect(bookYear, **new** IntWritable(1));

}

**else**

{

//bookYear.set(yearData);

bookYear.set("Invalid year");

//output.collect(bookYear, new IntWritable(1));

}

} **catch** (NumberFormatException e) {

bookYear.set("Invalid year");

}

**break**;

}

}

}

#### BookPublishFreqReduce.java

**public** **class** BookPublishFreqReducer **extends** MapReduceBase **implements**

Reducer<Text, IntWritable, Text, IntWritable> {

@Override

**public** **void** reduce(Text key, Iterator<IntWritable> values,

OutputCollector<Text, IntWritable> output, Reporter reporter)

**throws** IOException {

//Defining a local variable sum of type int

**int** sum = 0;

/\*

\* Iterates through all the values available with a key and adds them together

\* and give the final result as the key and sum of its values.

\*/

**while** (values.hasNext()) {

sum += values.next().get();

}

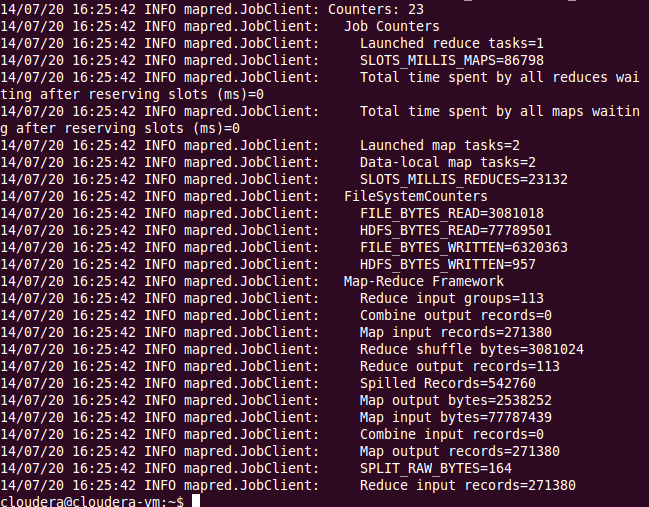
//Dumping the output

output.collect(key, **new** IntWritable(sum));

}

}

### Results



Result data set provided as zip files.

## Find out in which year maximum number of books were published

### Language used

Pig

### Source Code (Book\_year\_max.pig )

**NOTE : /BookPublish1\_28/part-00000 is from Problem 1 output.**

bookdata\_year\_max = load '/BookPublish1\_28/part-00000' using PigStorage (',') as (year: chararray, NoOfBooks: int);

-------- convert them to Tuple to get max value -----------------------

bookdata\_year\_max\_grp = group bookdata\_year\_max all;

---------Get max value -------------------------------

bookdata\_year\_max\_res = foreach bookdata\_year\_max\_grp generate MAX(bookdata\_year\_max.NoOfBooks) as maximum;

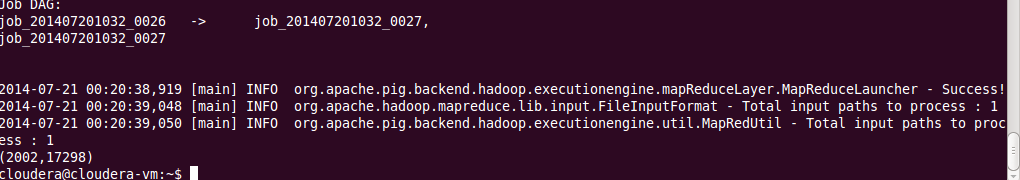
--- output year of max sales ------------------------------------------------------

res = filter bookdata\_year\_max by NoOfBooks == bookdata\_year\_max\_res.maximum ;

dump res ;

### Results

2002 17298



## Find out how many book were published based on ranking in the year 2002.

### Language used

Pig

### Source Code (Book\_rating.pig )

bookdata = load '/BX-Books.csv' using PigStorage (';') as (ISBN:chararray, Booktitle: chararray, BookAuthor: chararray, Year: chararray, Publisher: chararray, Image\_URL\_S: chararray, Image\_URL\_M: chararray, Image\_URL\_L: chararray);

bookrating = load '/BX-Book-Ratings.csv' using PigStorage (';') as (User: chararray, rating\_ISBN: chararray, BookRating: chararray);

--------- Filter data by ISBN and Year ---------------------

bookdataResult = foreach bookdata generate ISBN, Year;

--------- Filter data by ISBN and Rating ---------------------

ratingResult = foreach bookrating generate rating\_ISBN, BookRating;

--------- Filter data by year by 2002---------------------

bookdataFilter = FILTER bookdataResult by (Year == '"2002"');

bookdataISBN = foreach bookdataFilter generate ISBN;

--------- Join 2 DBs---------------------

bookdata\_rating = join bookdataISBN by ISBN , bookratingResult by rating\_ISBN;

--------- Filter ISBN and BookRating ---------------------

bookdataISBNRes = foreach bookdata\_rating generate ISBN, BookRating;

BookdataRatingGroup = group bookdataISBNRes by BookRating;

--------- Count books ---------------------

cnt = foreach BookdataRatingGroup generate group , COUNT(bookdataISBNRes);

dump cnt ;

### Results

