Big Data Engineering Course INFO 7250

Amazon Dataset Analysis

Nikhita Methwani

001492785

Problem Statement

Implement various Big Data Technologies such as Hadoop Map reduce, HIVE, MongoDB, Mahout, Apache Pig on Amazon Dataset to analyze various aspects of dataset

Dataset

Dataset: Amazon Customer Reviews on Health Care products

https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt

Fields Description:

DATA COLUMNS:
marketplace - 2 letter country code of the marketplace where the review was written.
customer_id - Random identifier that can be used to aggregate reviews written by a single author.
review_id - The unique ID of the review.
product_id - The unique Product ID the review pertains to. In the multilingual dataset the reviews for the same product in different countries can be grouped by the same product_id.
product_parent - Random identifier that can be used to aggregate reviews for the same product.
product_title - Title of the product.
product_category - Broad product category that can be used to group reviews (also used to group the dataset into coherent parts).
star_rating - The 1-5 star rating of the review.
helpful_votes - Number of helpful votes.
total_votes - Number of total votes the review received.
vine - Review was written as part of the Vine program.
verified_purchase - The review is on a verified purchase.
review_headline - The title of the review.
review_body - The review text.
review_date - The date the review was written.

Data Format:
Tab ('\t') separated text file, without quote or escape characters.
First line in each file is header; 1 line corresponds to 1 record.

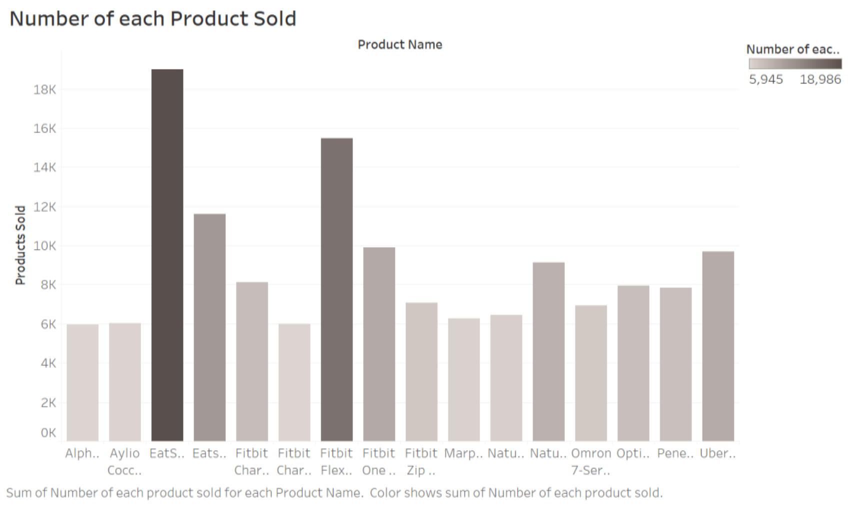
Analysis

1) Number of each Product Sold
2) Analyze top 10 products for each rating
3) Analyze each customer's product list along with the count of products purchased
4) Total Ratings count in the entire data
5) Verified Products along with their minimum and maximum ratings
6) Verified /non-verified purchase of the overall products

Hadoop Map reduce

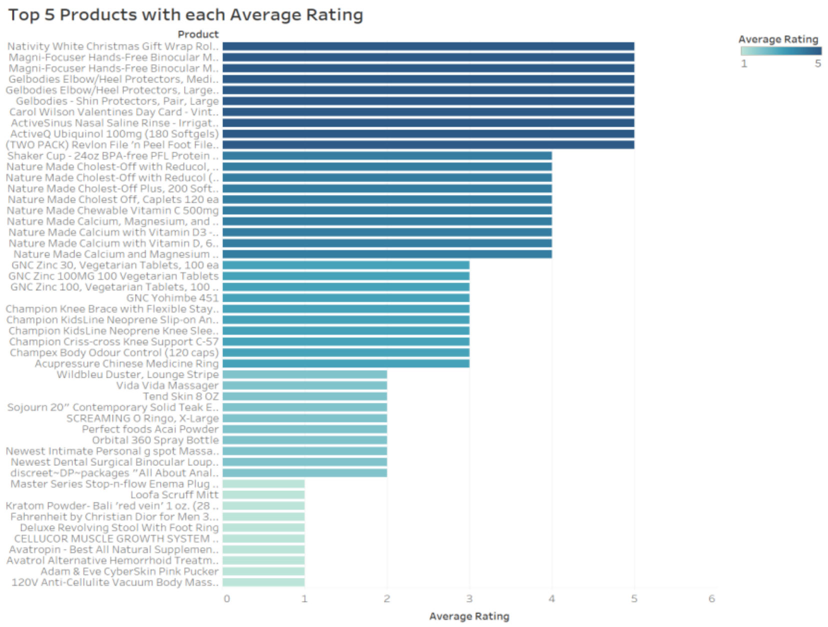
1) Number of each Product Sold

Analysis of number of products sold
It determines the products which are of high demand



2) Analyze top 10 products for each rating

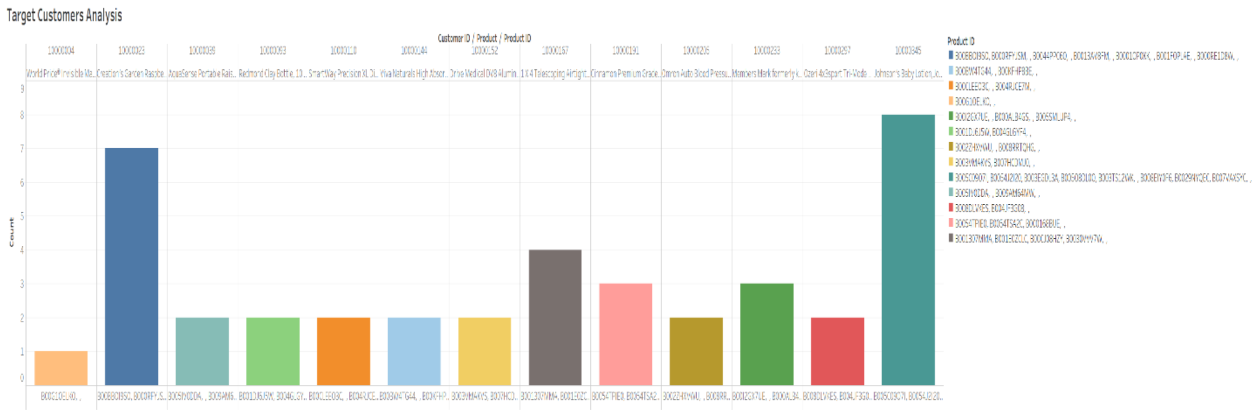
This evaluation will help the business to understand the trend of products along with the satisfaction of users as per the ratings [0-5]



Sum of Average Rating for each Product. Color shows sum of Average Rating. The view is filtered on sum of Average Rating, which ranges from 1 to 5.

3) Analyze each customer's product list along with the count of products purchased

The Analysis determines the list of products along with total products purchased by each customer. It helps to target customers according to their product purchase history.

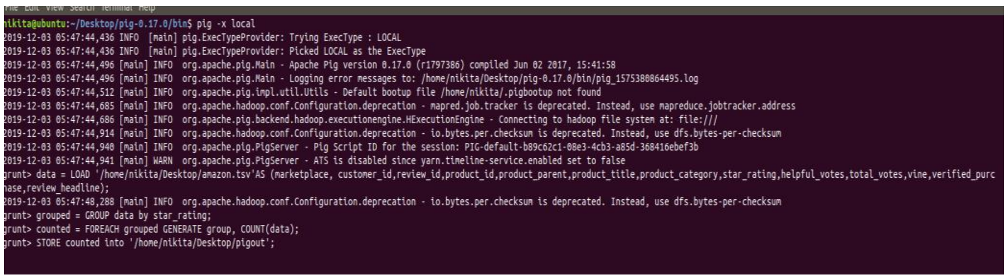


Sum of Count for each Product ID broken down by Customer ID and Product. Color shows details about Product ID.

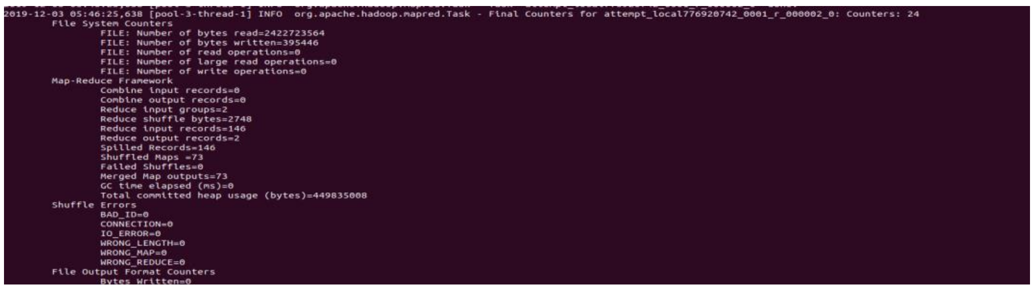
PIG

Total Ratings count in the entire data

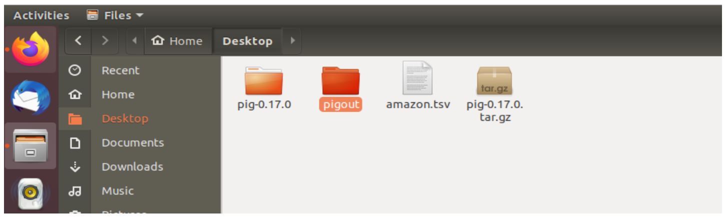
This analysis can determine the business about the service they are providing with their products based on overall ratings count [5 → in a healthy state]

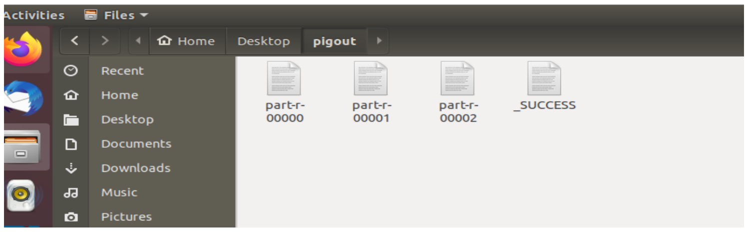


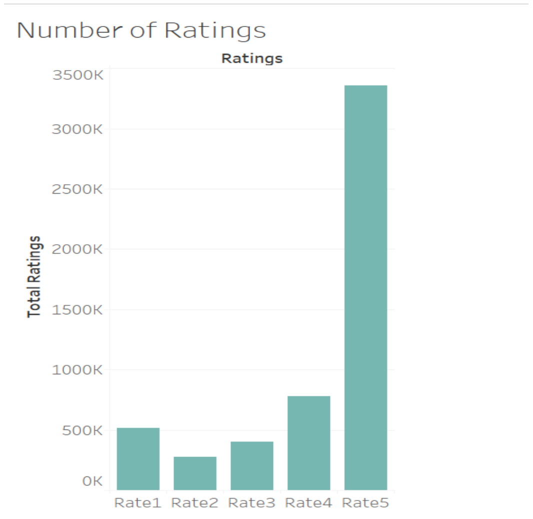
```
nikita@ubuntu:~/Desktop/pig-0.17.0/bin$ pig -x local
2019-12-03 05:47:44,436 INFO  [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2019-12-03 05:47:44,436 INFO  [main] pig.ExecTypeProvider: Picked LOCAL as the ExecType
2019-12-03 05:47:44,496 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2019-12-03 05:47:44,496 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/nikita/Desktop/pig-0.17.0/bin/pig_1575380864495.log
2019-12-03 05:47:44,512 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/nikita/.pigbootup not found
2019-12-03 05:47:44,685 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2019-12-03 05:47:44,686 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2019-12-03 05:47:44,914 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2019-12-03 05:47:44,940 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-b89c62c1-08e3-4cb3-a85d-368416ebef3b
2019-12-03 05:47:44,941 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> data = LOAD '/home/nikita/Desktop/amazon.tsv'AS (marketplace, customer_id,review_id,product_id,product_parent,product_title,product_category,star_rating,helpful_votes,total_votes,vine,verified_purc
hase,review_headline);
2019-12-03 05:47:48,288 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> grouped = GROUP data by star_rating;
grunt> counted = FOREACH grouped GENERATE group, COUNT(data);
grunt> STORE counted into '/home/nikita/Desktop/pigout';
```



```
2019-12-03 05:46:25,638 [pool-3-thread-1] INFO  org.apache.hadoop.mapred.Task - Final Counters for attempt_local776920742_0001_r_000002_0: Counters: 24
        File System Counters
                FILE: Number of bytes read=2422723564
                FILE: Number of bytes written=395446
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=2748
                Reduce input records=146
                Reduce output records=2
                Spilled Records=146
                Shuffled Maps =73
                Failed Shuffles=0
                Merged Map outputs=73
                GC time elapsed (ms)=0
                Total committed heap usage (bytes)=449835008
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Output Format Counters
                Bytes Written=0
```



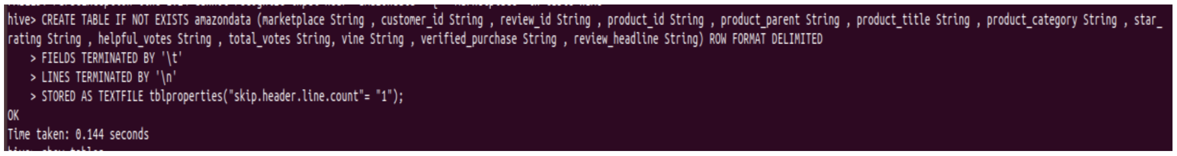

Activities
Files
Home
Desktop
pigout
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
part-r-00000
part-r-00001
part-r-00002
_SUCCESS


Number of Ratings
Ratings
Total Ratings
3500K
3000K
2500K
2000K
1500K
1000K
500K
0K
Rate1
Rate2
Rate3
Rate4
Rate5

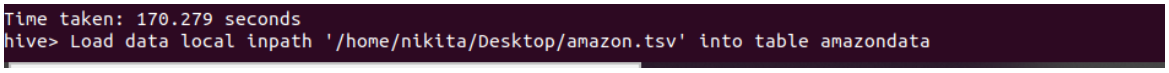
Hive

Verified Products along with their minimum and maximum ratings

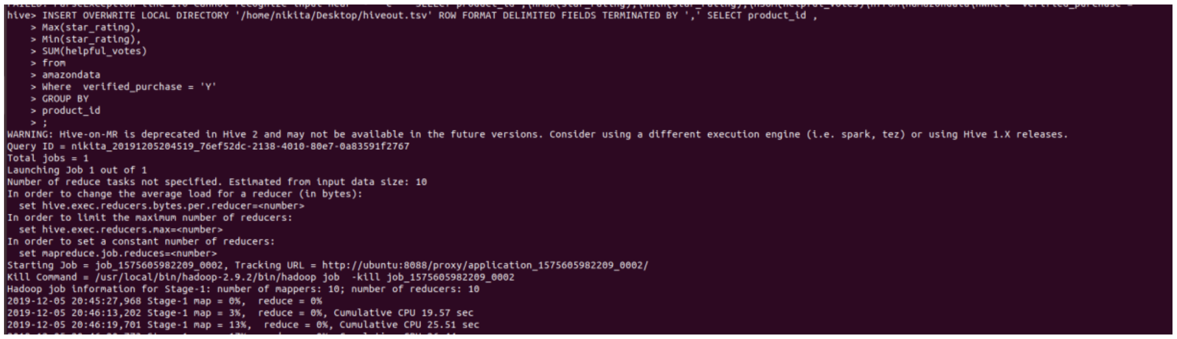
This Analysis will help to evaluate each verified product that is sold with its minimum rating and maximum rating and the helpful count determines the number of user found the reviews for the related product helpful.



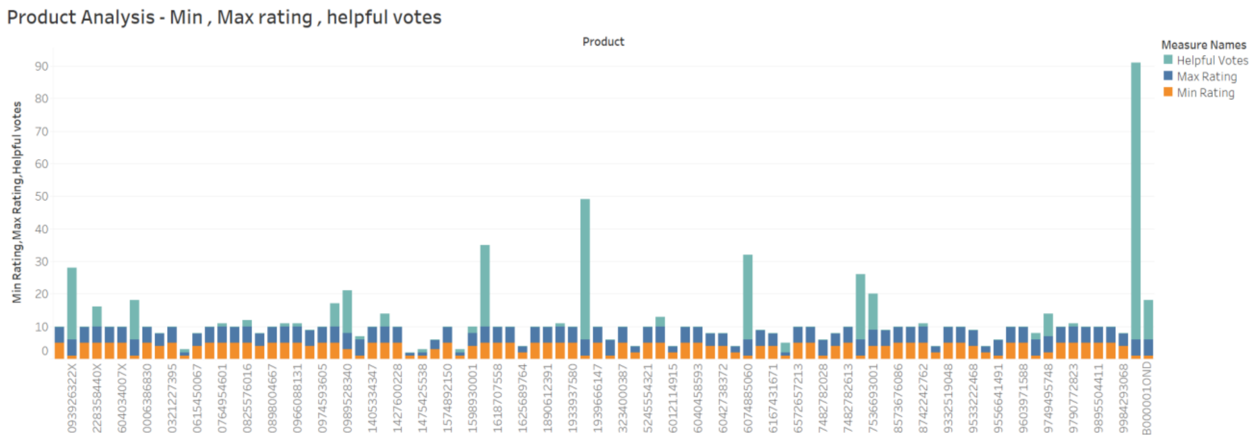
```
hive> CREATE TABLE IF NOT EXISTS amazondata (marketplace String , customer_id String , review_id String , product_id String , product_parent String , product_title String , product_category String , star_
rating String , helpful_votes String , total_votes String, vine String , verified_purchase String , review_headline String) ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '\t'
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE tblproperties("skip.header.line.count"= "1");
OK
Time taken: 0.144 seconds
```



```
Time taken: 170.279 seconds
hive> Load data local inpath '/home/nikita/Desktop/amazon.tsv' into table amazondata
```



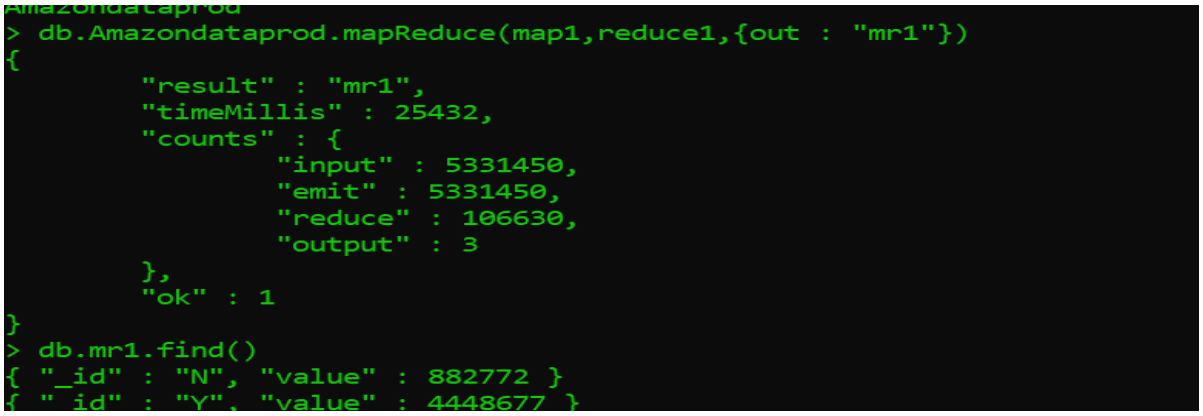
```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/home/nikita/Desktop/hiveout.tsv' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' SELECT product_id ,
    > Max(star_rating),
    > Min(star_rating),
    > SUM(helpful_votes)
    > from
    > amazondata
    > Where  verified_purchase = 'Y'
    > GROUP BY
    > product_id
    > ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = nikita_20191205204519_76ef52dc-2138-4010-80e7-0a83591f2767
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 10
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_1575605982209_0002, Tracking URL = http://ubuntu:8088/proxy/application_1575605982209_0002/
Kill Command = /usr/local/bin/hadoop-2.9.2/bin/hadoop job  -kill job_1575605982209_0002
Hadoop job information for Stage-1: number of mappers: 10; number of reducers: 10
2019-12-05 20:45:27,968 Stage-1 map = 0%,  reduce = 0%
2019-12-05 20:46:13,202 Stage-1 map = 3%,  reduce = 0%, Cumulative CPU 19.57 sec
2019-12-05 20:46:19,701 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 25.51 sec
```



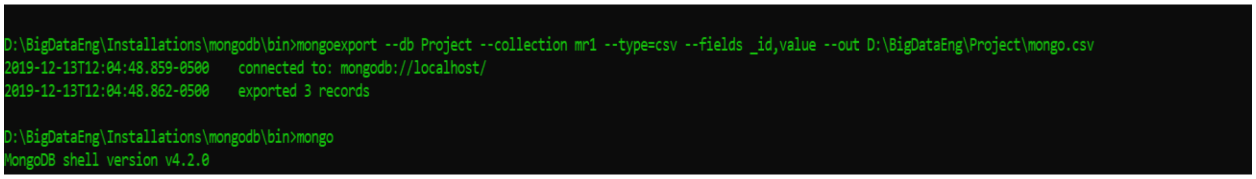
Helpful Votes, Max Rating and Min Rating for each Product. Color shows details about Helpful Votes, Max Rating and Min Rating.

Mongo dB – Map Reduce

Verified /non-verified purchase of the overall products

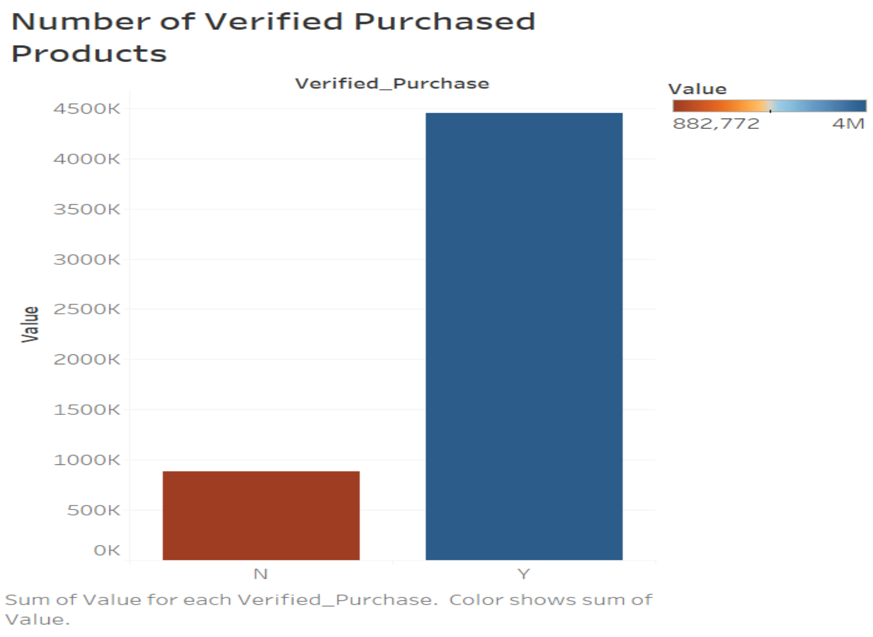


```
Amazondataprod
> db.Amazondataprod.mapReduce(map1,reduce1,{out : "mr1"})
{
        "result" : "mr1",
        "timeMillis" : 25432,
        "counts" : {
                "input" : 5331450,
                "emit" : 5331450,
                "reduce" : 106630,
                "output" : 3
        },
        "ok" : 1
}
> db.mr1.find()
{ "_id" : "N", "value" : 882772 }
{ "_id" : "Y", "value" : 4448677 }
```



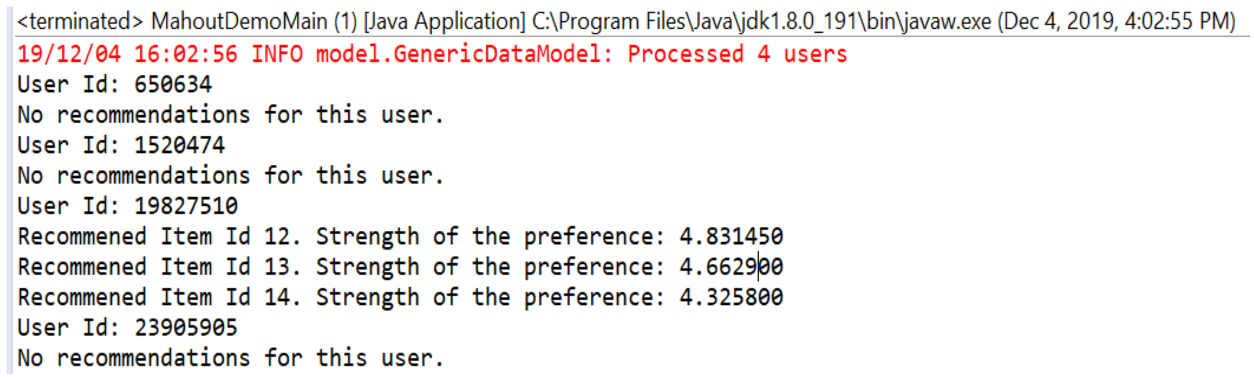
```
D:\BigDataEng\Installations\mongodb\bin>mongoexport --db Project --collection mr1 --type=csv --fields _id,value --out D:\BigDataEng\Project\mongo.csv
2019-12-13T12:04:48.859-0500    connected to: mongodb://localhost/
2019-12-13T12:04:48.862-0500    exported 3 records

D:\BigDataEng\Installations\mongodb\bin>mongo
MongoDB shell version v4.2.0
```



Mahout

Recommend Customers with Products along with strength of preference



```
<terminated> MahoutDemoMain (1) [Java Application] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (Dec 4, 2019, 4:02:55 PM)
19/12/04 16:02:56 INFO model.GenericDataModel: Processed 4 users
User Id: 650634
No recommendations for this user.
User Id: 1520474
No recommendations for this user.
User Id: 19827510
Recommened Item Id 12. Strength of the preference: 4.831450
Recommened Item Id 13. Strength of the preference: 4.662900
Recommened Item Id 14. Strength of the preference: 4.325800
User Id: 23905905
No recommendations for this user.
```

Appendix

1) MongoDB

Upload Data:

```
package BDE_Assignment1;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

import org.bson.Document;
import org.bson.types.ObjectId;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class MovieLens_MongoDb {

    public static void main(String[] args)
    {
            MongoClient mongoClient = new MongoClient("localhost",27017);
            MongoDatabase database = mongoClient.getDatabase("Project");
            MongoCollection<Document> collection = database.getCollection("Amazondataprod");
            Document document = new Document();
            int count = 0;
            BufferedReader objReader = null;
             try {
              String strCurrentLine;
              String[] amazondata;


              objReader = new BufferedReader(new
FileReader("D:\\BigDataEng\\Project\\amazon\\amazon.tsv"));

              while ((strCurrentLine = objReader.readLine()) != null) {
```

```
                                                amazondata = strCurrentLine.split("\\t");

                                                ObjectId id = new ObjectId();
                                        document.put("_id", id);
    //                                          document.put("product_id",amazondata[3]);
    //                      document.put("star_rating",amazondata[7]);
                                        document.put("product_id",amazondata[4]);
                            document.put("review",amazondata[12]);
                            document.put("verified_purchase",amazondata[11]);
                            collection.insertOne(document);
                            System.out.println(document.toJson());
                            count++;
                      }
                    System.out.println(count);
                     } catch (IOException e) {

                      e.printStackTrace();

                     } finally {

                      try {
                       if (objReader != null)
                        objReader.close();
                      } catch (IOException ex) {
                       ex.printStackTrace();
                      }
                     }
                    }
    }
```

<u>Mapper</u>:

```
    function()
{
emit(this.verified_purchase,1);
}
```

<u>Reducer</u>:

```
function(key,value)
{
var count = 0 ;
for(var i = 0 ; i<value.length ; i++)
{
count ++;
}
return count ;
```

}

2) Mahout

```
package m1.mahout;

import java.io.*;
import java.util.*;

import org.apache.mahout.cf.taste.impl.common.LongPrimitiveIterator;
import org.apache.mahout.cf.taste.impl.model.file.*;
import org.apache.mahout.cf.taste.impl.neighborhood.*;
import org.apache.mahout.cf.taste.impl.recommender.*;
import org.apache.mahout.cf.taste.impl.similarity.*;
import org.apache.mahout.cf.taste.model.*;
import org.apache.mahout.cf.taste.neighborhood.*;
import org.apache.mahout.cf.taste.recommender.*;
import org.apache.mahout.cf.taste.similarity.*;

public class MahoutDemoMain {

  public static void main(String[] args) throws Exception {
     // Create a data source from the CSV file
     File userPreferencesFile = new File("D:\\BigDataEng\\Project\\mahout.csv");
     DataModel dataModel = new FileDataModel(userPreferencesFile);

     UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(dataModel);
//      UserNeighborhood userNeighborhood = new NearestNUserNeighborhood(2,
userSimilarity, dataModel);
     UserNeighborhood userNeighborhood = new ThresholdUserNeighborhood(0.2,
userSimilarity, dataModel);

     // Create a generic user based recommender with the dataModel, the
userNeighborhood and the userSimilarity
     Recommender genericRecommender =  new
GenericUserBasedRecommender(dataModel, userNeighborhood, userSimilarity);

     // Recommend 5 items for each user
     for (LongPrimitiveIterator iterator = dataModel.getUserIDs(); iterator.hasNext();)
     {
        long userId = iterator.nextLong();

        // Generate a list of 5 recommendations for the user
        List<RecommendedItem> itemRecommendations =
genericRecommender.recommend(userId, 3);

        System.out.format("User Id: %d%n", userId);
```

```
          if (itemRecommendations.isEmpty())
          {
             System.out.println("No recommendations for this user.");
          }
          else
          {
             // Display the list of recommendations
             for (RecommendedItem recommendedItem : itemRecommendations)
             {
                System.out.format("Recommened Item Id %d. Strength of the preference: 
%f%n", recommendedItem.getItemID(), recommendedItem.getValue());
             }
          }
       }
   }
}
```

3)Map reduce:

Use Case1: Number of Products Sold

```
package project.P1;

import java.io.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;

public class Map extends Mapper<LongWritable, Text, Text, IntWritable> {

	// Called once for each key/value pair in the input split
	IntWritable count = new IntWritable(1);
	String prodtitle ;
	Text prod_title = new Text();


	@Override
	public void map(LongWritable key, Text value, Context context) throws IOException, 
InterruptedException {
		String[] tokens = value.toString().split("\t");
		if (tokens[5].equals("product_title")) {
			return;
		} else {
			prodtitle = tokens[5];
		}
```

```
                        prod_title.set(prodtitle);

                        context.write(prod_title,count);

            }

}
```

Reduce:

```
package project.P1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

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

                        IntWritable c = new IntWritable();
                        int count = 0;



                        for (IntWritable val : values) {
                                    count+= val.get();
                        }

                        c.set(count);
                        context.write(key ,c);
            }
}
```

Driver:

```
package project.P1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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.jobcontrol.ControlledJob;
import org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Driver {

        public static void main(String[] args) throws Exception {
//              JobControl jobControl = new JobControl("jobChain");
                Configuration conf = new Configuration();

                Job job = Job.getInstance(conf, "Project_P1");
                job.setJarByClass(Driver.class);

                job.setMapperClass(Map.class);
                job.setCombinerClass(Reduce.class);
                job.setReducerClass(Reduce.class);

                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(IntWritable.class);

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

                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));


//              Thread jobControlThread = new Thread(jobControl);
//              jobControlThread.start();

                FileSystem fs = FileSystem.get(conf);
                fs.delete(new Path(args[1]), true);

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

        }
}
```

Use Case2:

Map:

```
package project.P1;
```

```
import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class Map extends Mapper<LongWritable, Text, Text, CompositeKey> {

	Text prod_cat = new Text();
	CompositeKey ck = new CompositeKey();

	@Override
	protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

		String[] tokens = value.toString().split("\t");
		if (tokens[5].equals("product_title")) {
			return;
		} else {
			prod_cat.set(tokens[5].trim());
		}

		if (tokens[7].contains("star_rating")) {
			return;

		} else {
			long rate = Long.parseLong(tokens[7].trim());

			ck.setCount(1);
			ck.setRating_avg(rate);

		}

		context.write(prod_cat, ck);
	}
}
```

Reduce:

```
package project.P1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class Reduce extends Reducer<Text, CompositeKey, Text, CompositeKey> {


	CompositeKey ck = new CompositeKey();
	@Override
	protected void reduce(Text key, Iterable<CompositeKey> values, Context context)
			throws IOException, InterruptedException {

		long sum = 0;
		long count = 0;
		long avg = 0;

		for (CompositeKey val : values)
		{
			sum += val.getRating_avg() * val.getCount() ;
			count = count + val.getCount();
		}

		avg = sum/count;

		ck.setCount(count);
		ck.setRating_avg(avg);

		context.write(key, ck);

	}

}
```
Map2:
```
package project.P1;

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

public class Map2 extends Mapper<LongWritable, Text, LongWritable, Text> {

	LongWritable avg = new LongWritable();
	public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

		String line = value.toString();
		String[] tokens = line.split("\t");

		Text prd = new Text(tokens[0]);

		long rating_avg = Long.parseLong(tokens[2]);

		avg.set(rating_avg);

		context.write(avg, prd);

	}

}
```

Reduce2:

```
package project.P1;

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

public class Reduce2 extends Reducer<LongWritable, Text, LongWritable, Text> {

	LongWritable avg = new LongWritable();

	protected void reduce(LongWritable key, Iterable<Text> value, Context context)
			throws IOException, InterruptedException {
		int counter = 0;
		for (Text val : value) {
		counter++;
		if(counter<= 10)
		{
				context.write(key, val);
			}

		}

	}
```

```
}

Driver:
package project.P1;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.jobcontrol.ControlledJob;
import org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {
	public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException
	{
		// First MapReduce
		JobControl jobControl = new JobControl("jobChain");
		Configuration cnf1 = new Configuration();

		Job job1 = Job.getInstance(cnf1);
		job1.setJarByClass(Driver.class);
		job1.setJobName("MR1");

		FileInputFormat.setInputPaths(job1, new Path(args[0]));
		FileOutputFormat.setOutputPath(job1, new Path(args[1] + "/temp"));

		job1.setMapperClass(Map.class);
		job1.setReducerClass(Reduce.class);
		 job1.setCombinerClass(Reduce.class);

//		 job1.setNumReduceTasks(4);

		job1.setOutputKeyClass(Text.class);
		job1.setOutputValueClass(CompositeKey.class);
```

```
                        job1.setInputFormatClass(TextInputFormat.class);
                        job1.setOutputFormatClass(TextOutputFormat.class);

                        ControlledJob controlledJob1 = new ControlledJob(cnf1);
                        controlledJob1.setJob(job1);
                        jobControl.addJob(controlledJob1);

                        // Second MapReduce

                        Configuration cnf2 = new Configuration();

                        Job job2 = Job.getInstance(cnf2);
                        job2.setJarByClass(Driver.class);
                        job2.setJobName("MR2");

//                      job2.setNumReduceTasks(4);
                        job2.setMapperClass(Map2.class);
                        job2.setReducerClass(Reduce2.class);
                        job2.setCombinerClass(Reduce2.class);

//                      job2.setPartitionerClass(CustomPartiton.class);

                        job2.setMapOutputKeyClass(LongWritable.class);
                        job2.setMapOutputValueClass(Text.class);

                        job2.setOutputKeyClass(LongWritable.class);
                        job2.setOutputValueClass(Text.class);

                        job2.setInputFormatClass(TextInputFormat.class);
                        job2.setOutputFormatClass(TextOutputFormat.class);


                        ControlledJob controlledJob2 = new ControlledJob(cnf2);
                        controlledJob2.setJob(job2);

                        FileInputFormat.setInputPaths(job2, new Path(args[1] + "/temp"));
                        FileOutputFormat.setOutputPath(job2, new Path(args[1] + "/final"));
//
                        FileSystem fs = FileSystem.get(cnf1);
                        fs.delete(new Path(args[1]), true);
//
//                      // make job2 dependent on job1
                        controlledJob2.addDependingJob(controlledJob1);
```

```
//			// add the job to the job control
			jobControl.addJob(controlledJob2);

			Thread jobControlThread = new Thread(jobControl);
			jobControlThread.start();

			System.exit(job2.waitForCompletion(true) ? 0 : 1);

		}
}
```

CompositeKey:

```
package project.P1;

import org.apache.hadoop.io.Writable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class CompositeKey implements Writable {
		private long count;
		private long rating_avg;


		public long getCount() {
			return count;
		}

		public void setCount(long count) {
			this.count = count;
		}

		public long getRating_avg() {
			return rating_avg;
		}

		public void setRating_avg(long rating_avg) {
			this.rating_avg = rating_avg;
		}



		public CompositeKey() {
```

```
		super();
	}

	public CompositeKey(long count, long rating_avg) {
		super();
		this.count = count;
		this.rating_avg = rating_avg;
	}

	public void readFields(DataInput in) throws IOException {
		count = in.readLong();
		rating_avg = in.readLong();
	}

	public void write(DataOutput out) throws IOException {
		out.writeLong(count);
		out.writeLong(rating_avg);
	}

	@Override
	public String toString() {
		return count + "\t" + rating_avg;
	}

}
```

CustomPartition:

```
package project.P1;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Partitioner;

public class CustomPartiton extends Partitioner<LongWritable, Text> {
	public int getPartition(LongWritable key, Text value, int numReduceTasks) {
		if (numReduceTasks == 0)
			return 0;
		if (key.equals(1))
			return 0;
		if (key.equals(2))
			return 1;
		if (key.equals(3))
			return 2;
		if (key.equals(4))
```

```
                    return 3;
            else
                    return 4;
    }

}
```

Use Case 3: Customer → Products Purchased
Map:

```
package project.P1;

import java.io.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;

public class Map extends Mapper<LongWritable, Text, Text, CompositeKey> {

    // Called once for each key/value pair in the input split
    IntWritable count = new IntWritable(1);

    Text customerid = new Text();
    CompositeKey ck = new CompositeKey();

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
            String[] tokens = value.toString().split("\t");

            String customer_id = tokens[1];
            String product_id = tokens[3];
            String product_title = tokens[5];

            customerid.set(customer_id);


            ck.setProductid(product_id);
            ck.setProduct_title(product_title);
            ck.setCount(1);


            context.write(customerid, ck);
```

```
	}

}
```

Reduce:

```
package project.P1;

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

public class Reduce extends Reducer<Text, CompositeKey, Text,CompositeKey> {



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

		String productid = " ";
		String producttitle = " ";
		String out = " ";
		int count = 0;
		Text proddetails = new Text();
		CompositeKey ck = new CompositeKey();
		for (CompositeKey val : values)
		{
//			out = val.getProductid() + val.getProduct_title();
			productid = val.getProductid() + ", " + productid;
			producttitle = val.getProduct_title() +"," +  producttitle;
			count+= val.getCount();
		}

//		out = out + count;
		ck.setCount(count);
		ck.setProductid(productid);
		ck.setProduct_title(producttitle);
//		proddetails.set(out);

		context.write(key ,ck);
	}
}
```

CompositeKey

```
package project.P1;
```

```
import org.apache.hadoop.io.Writable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class CompositeKey implements Writable {
	private String productid;
	private String product_title;
	private long count;


	public String getProductid() {
		return productid;
	}

	public void setProductid(String productid) {
		this.productid = productid;
	}

	public String getProduct_title() {
		return product_title;
	}

	public void setProduct_title(String product_title) {
		this.product_title = product_title;
	}

	public long getCount() {
		return count;
	}

	public void setCount(long count) {
		this.count = count;
	}

	public CompositeKey() {
		super();
	}



	public CompositeKey(String productid, String product_title, long count) {
		super();
		this.productid = productid;
		this.product_title = product_title;
		this.count = count;
	}

	public void readFields(DataInput in) throws IOException {
		productid = in.readUTF();
		product_title = in.readUTF();
		count = in.readLong();

	}

	public void write(DataOutput out) throws IOException {
		out.writeUTF(productid);
		out.writeUTF(product_title);
		out.writeLong(count);

	}

	@Override
	public String toString() {
		return productid + "\t" + product_title + "\t" + count;
	}

}
```

Driver:

```
package project.P1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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.jobcontrol.ControlledJob;
import org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Driver {
	public static void main(String[] args) throws Exception {
//		JobControl jobControl = new JobControl("jobChain");
		Configuration conf = new Configuration();

		Job job = Job.getInstance(conf, "Project_P1");
		job.setJarByClass(Driver.class);

		job.setMapperClass(Map.class);
		job.setCombinerClass(Reduce.class);
		job.setReducerClass(Reduce.class);

		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(CompositeKey.class);

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

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));


//		Thread jobControlThread = new Thread(jobControl);
//		jobControlThread.start();

		FileSystem fs = FileSystem.get(conf);
		fs.delete(new Path(args[1]), true);

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

	}
}
```