Engineering of Big Data Systems

Analysis of Amazon Customer Review's Dataset

Aditya Joshi

NUID 001837740

Aditya Joshi
001837740

Contents:

Aditya Joshi
001837740

1. **About the dataset**:
   The Amazon Customer Reviews Dataset is a large dataset with size > 20GB. However, for this analysis, we've used a subset of this dataset named "amazon_reviews_us_Home_Entertainment_v1_00".
   This dataset is of 500 MB in size.

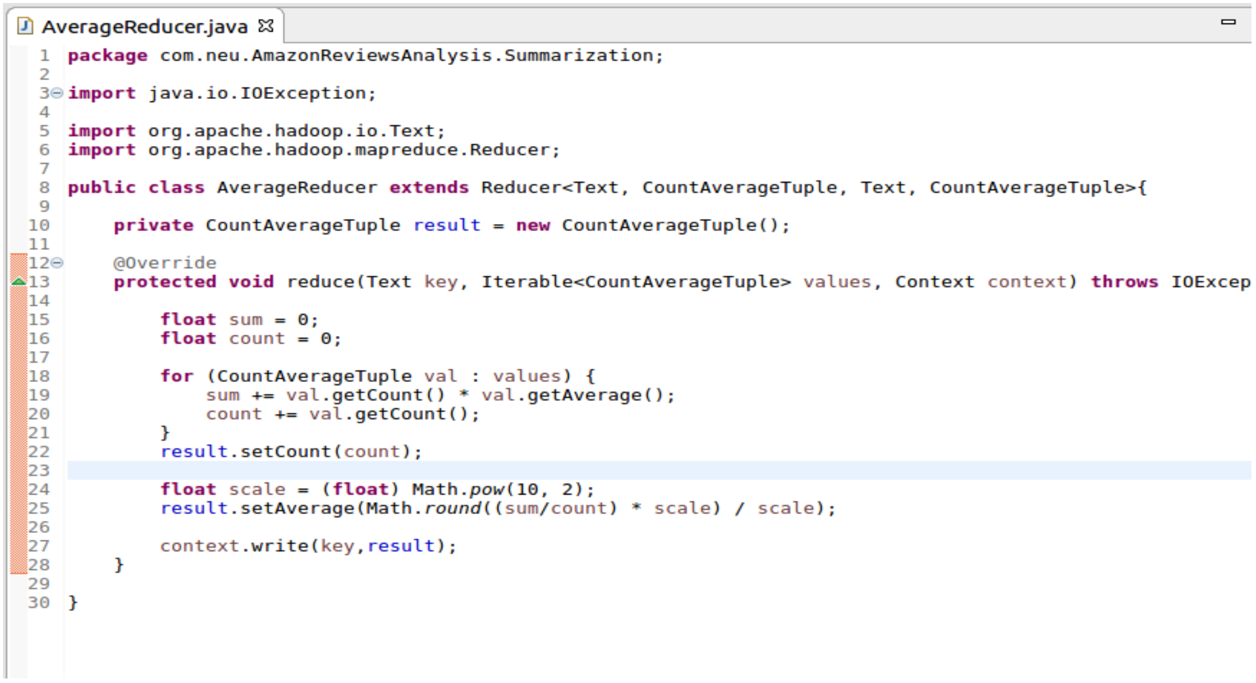
   Here's the detailed description of dataset and its contents.

   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.

   Link for dataset:
   https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Home_Entertainment_v1_00.tsv.gz

Aditya Joshi
001837740

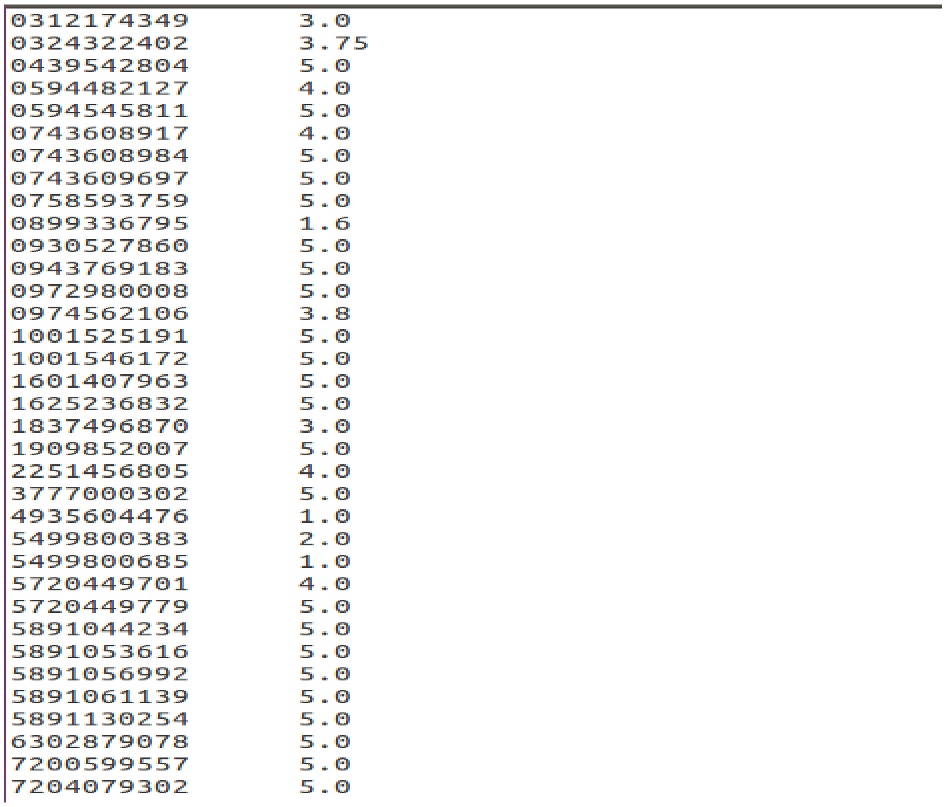
2. **Map Reduce by Summarization**:
   Average Values:
   This map reduce job is to calculate the average rating for each reviewed product. Reducer is also used as Combiner.



AverageReducer.java

```java
package com.neu.AmazonReviewsAnalysis.Summarization;

import java.io.IOException;

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

public class AverageReducer extends Reducer<Text, CountAverageTuple, Text, CountAverageTuple>{

    private CountAverageTuple result = new CountAverageTuple();

    @Override
    protected void reduce(Text key, Iterable<CountAverageTuple> values, Context context) throws IOExcep

        float sum = 0;
        float count = 0;

        for (CountAverageTuple val : values) {
            sum += val.getCount() * val.getAverage();
            count += val.getCount();
        }
        result.setCount(count);

        float scale = (float) Math.pow(10, 2);
        result.setAverage(Math.round((sum/count) * scale) / scale);

        context.write(key,result);
    }

}
```

Output:

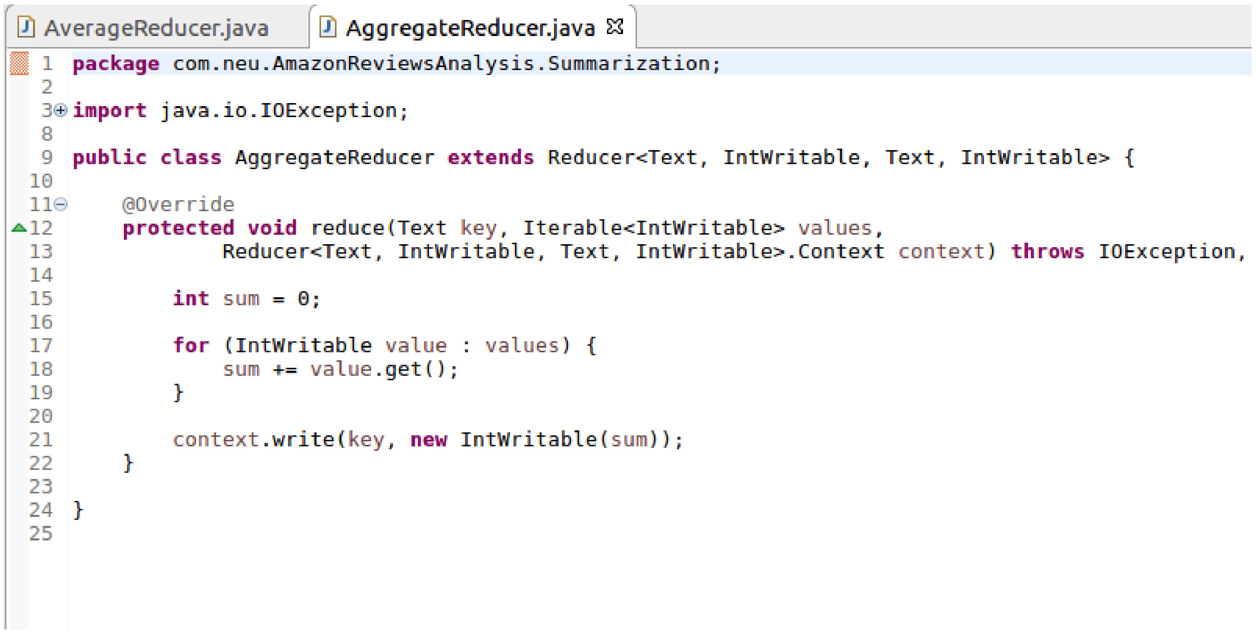


```
0312174349      3.0
0324322402      3.75
0439542804      5.0
0594482127      4.0
0594545811      5.0
0743608917      4.0
0743608984      5.0
0743609697      5.0
0758593759      5.0
0899336795      1.6
0930527860      5.0
0943769183      5.0
0972980008      5.0
0974562106      3.8
1001525191      5.0
1001546172      5.0
1601407963      5.0
1625236832      5.0
1837496870      3.0
1909852007      5.0
2251456805      4.0
3777000302      5.0
4935604476      1.0
5499800383      2.0
5499800685      1.0
5720449701      4.0
5720449779      5.0
5891044234      5.0
5891053616      5.0
5891056992      5.0
5891061139      5.0
5891130254      5.0
6302879078      5.0
7200599557      5.0
7204079302      5.0
```

Aditya Joshi
001837740

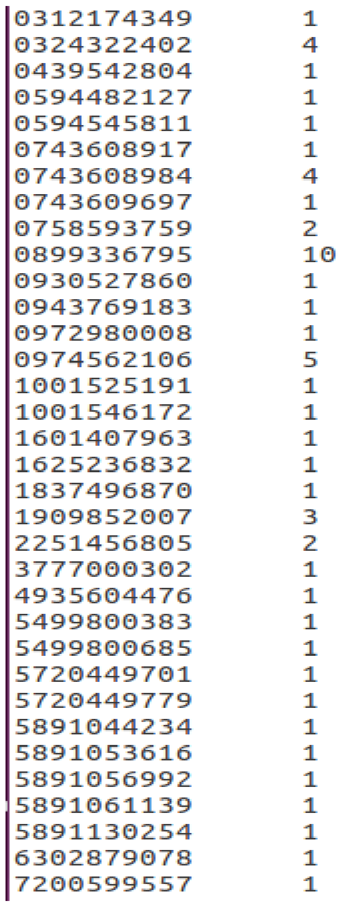
Aggregate Values

This map reduce job calculates the number of reviews per product.



```
package com.neu.AmazonReviewsAnalysis.Summarization;

import java.io.IOException;

public class AggregateReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException,

        int sum = 0;

        for (IntWritable value : values) {
            sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }

}
```

Output:

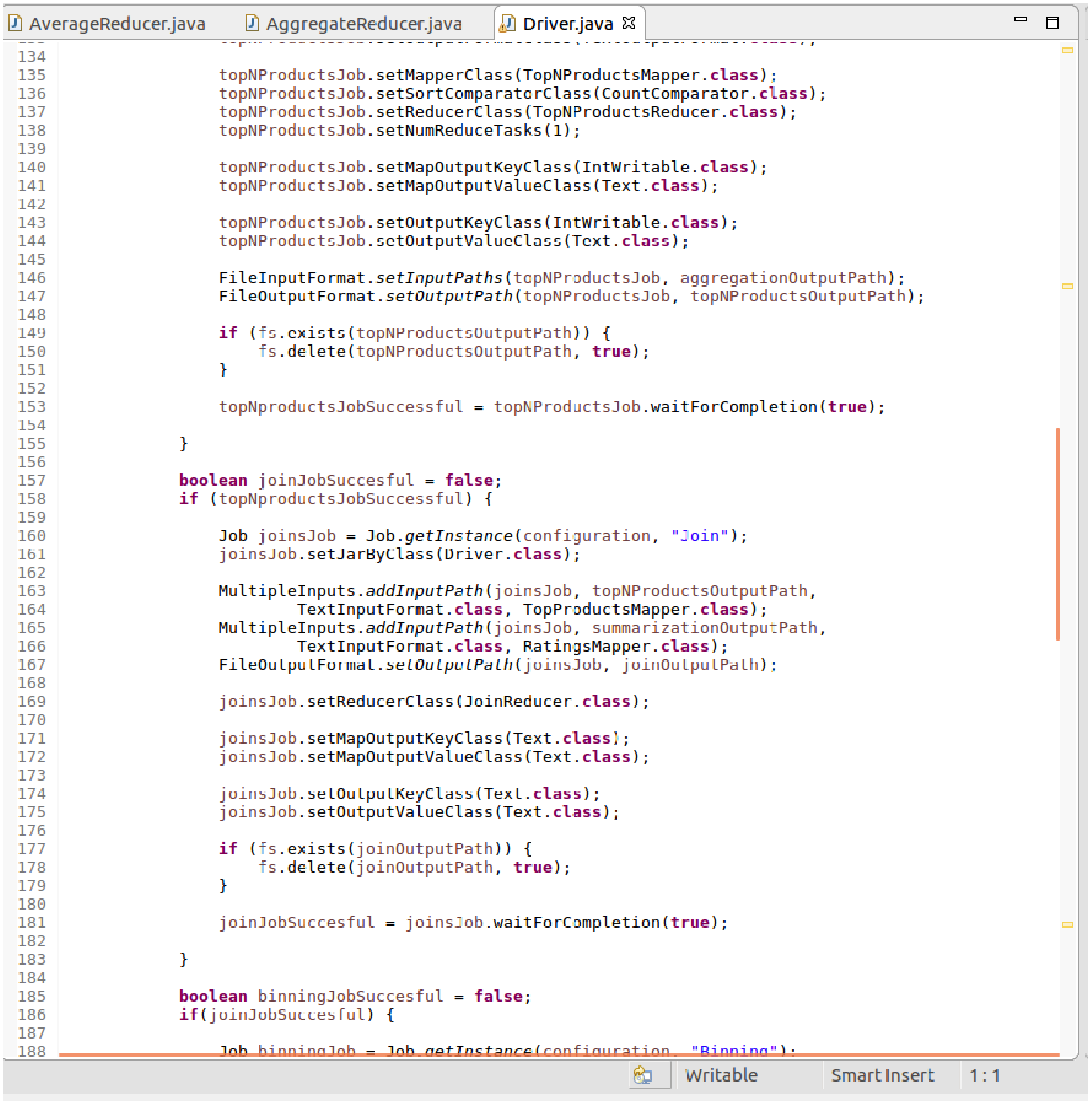


```
0312174349      1
0324322402      4
0439542804      1
0594482127      1
0594545811      1
0743608917      1
0743608984      4
0743609697      1
0758593759      2
0899336795      10
0930527860      1
0943769183      1
0972980008      1
0974562106      5
1001525191      1
1001546172      1
1601407963      1
1625236832      1
1837496870      1
1909852007      3
2251456805      2
3777000302      1
4935604476      1
5499800383      1
5499800685      1
5720449701      1
5720449779      1
5891044234      1
5891053616      1
5891056992      1
5891061139      1
5891130254      1
6302879078      1
7200599557      1
```

Aditya Joshi
001837740

3. Job Chaining

The MR job uses the chaining where the output of one MapReduce job goes to the other before finally outputting it to the user. Job chaining has been used for all the jobs in this analysis.

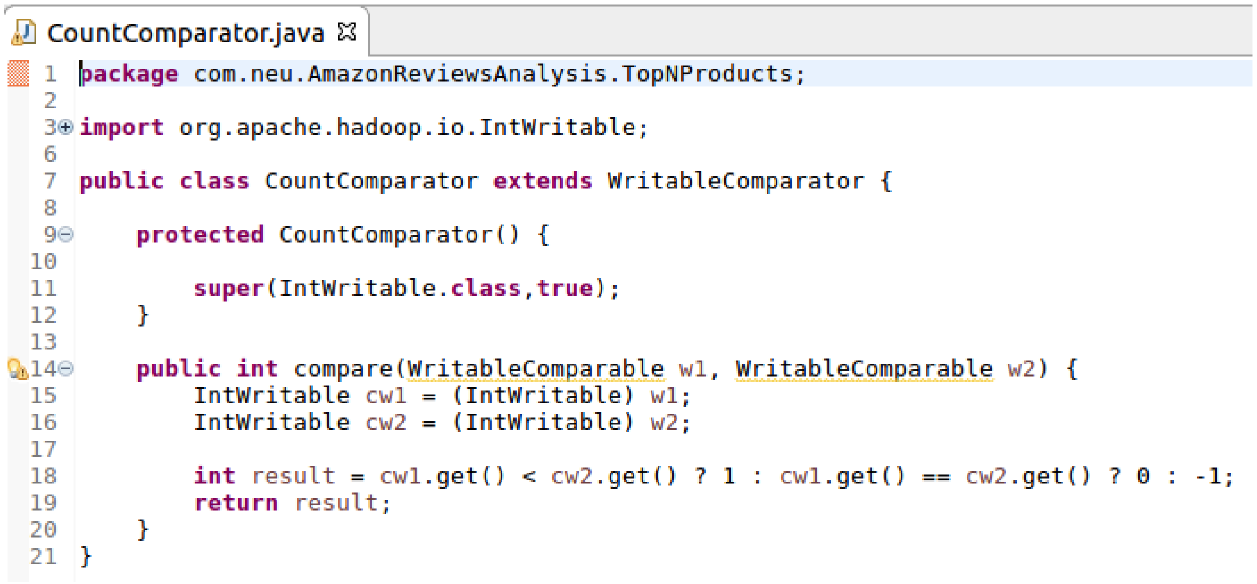


```java
            topNProductsJob.setMapperClass(TopNProductsMapper.class);
            topNProductsJob.setSortComparatorClass(CountComparator.class);
            topNProductsJob.setReducerClass(TopNProductsReducer.class);
            topNProductsJob.setNumReduceTasks(1);

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

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

            FileInputFormat.setInputPaths(topNProductsJob, aggregationOutputPath);
            FileOutputFormat.setOutputPath(topNProductsJob, topNProductsOutputPath);

            if (fs.exists(topNProductsOutputPath)) {
                fs.delete(topNProductsOutputPath, true);
            }

            topNproductsJobSuccessful = topNProductsJob.waitForCompletion(true);

        }

        boolean joinJobSuccesful = false;
        if (topNproductsJobSuccessful) {

            Job joinsJob = Job.getInstance(configuration, "Join");
            joinsJob.setJarByClass(Driver.class);

            MultipleInputs.addInputPath(joinsJob, topNProductsOutputPath,
                    TextInputFormat.class, TopProductsMapper.class);
            MultipleInputs.addInputPath(joinsJob, summarizationOutputPath,
                    TextInputFormat.class, RatingsMapper.class);
            FileOutputFormat.setOutputPath(joinsJob, joinOutputPath);

            joinsJob.setReducerClass(JoinReducer.class);

            joinsJob.setMapOutputKeyClass(Text.class);
            joinsJob.setMapOutputValueClass(Text.class);

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

            if (fs.exists(joinOutputPath)) {
                fs.delete(joinOutputPath, true);
            }

            joinJobSuccesful = joinsJob.waitForCompletion(true);

        }

        boolean binningJobSuccesful = false;
        if(joinJobSuccesful) {

            Job binningJob = Job.getInstance(configuration, "Binning");
```

Aditya Joshi
001837740

4. Map reduce for Top N values using secondary sort.

Here secondary sort is implemented by using a comparator class.

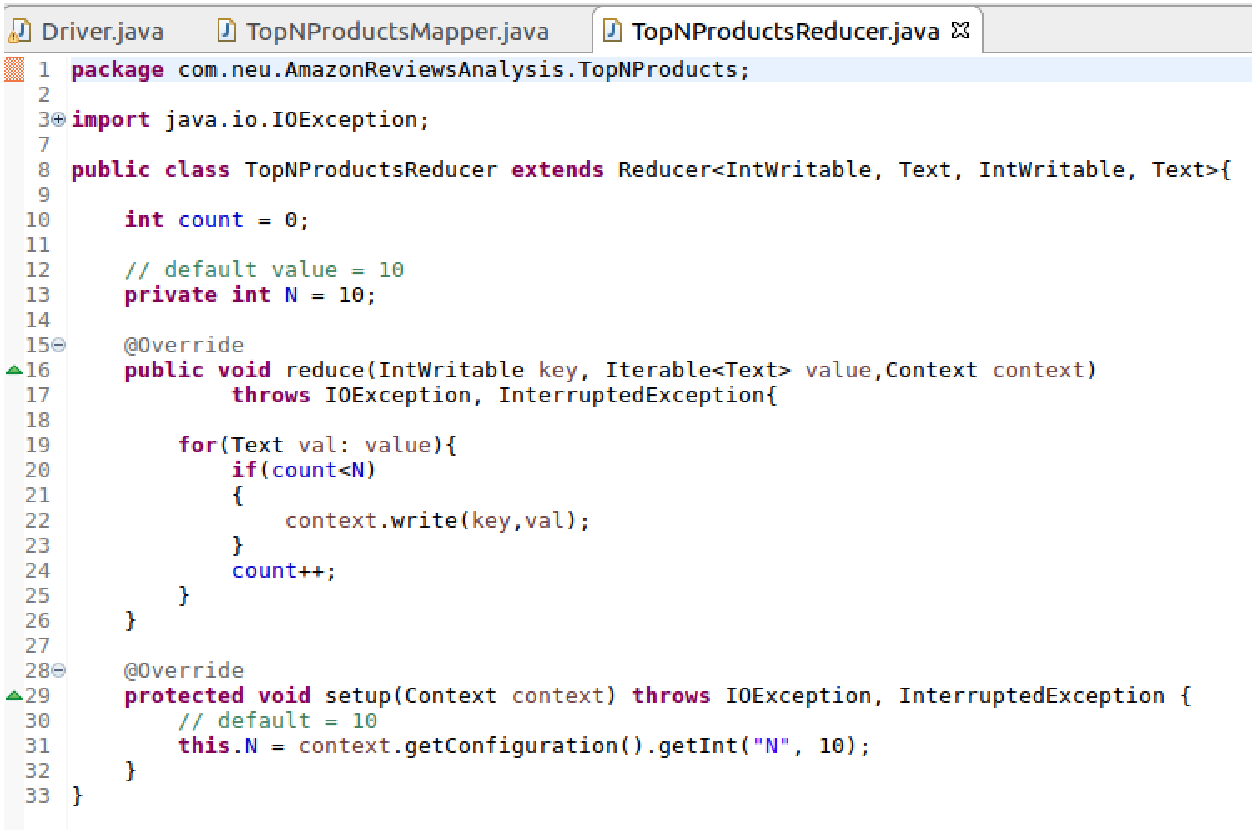

CountComparator.java

```java
package com.neu.AmazonReviewsAnalysis.TopNProducts;

import org.apache.hadoop.io.IntWritable;

public class CountComparator extends WritableComparator {

    protected CountComparator() {

        super(IntWritable.class,true);
    }

    public int compare(WritableComparable w1, WritableComparable w2) {
        IntWritable cw1 = (IntWritable) w1;
        IntWritable cw2 = (IntWritable) w2;

        int result = cw1.get() < cw2.get() ? 1 : cw1.get() == cw2.get() ? 0 : -1;
        return result;
    }
}
```

```java
boolean topNproductsJobSuccessful = false;
if (issummarizationJobSuccessful) {

    Job topNProductsJob = Job.getInstance(configuration, "Top N Rated Products");
    topNProductsJob.setJarByClass(Driver.class);

    int N = 200;
    topNProductsJob.getConfiguration().setInt("N", N);

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

    topNProductsJob.setMapperClass(TopNProductsMapper.class);
    topNProductsJob.setSortComparatorClass(CountComparator.class);
    topNProductsJob.setReducerClass(TopNProductsReducer.class);
    topNProductsJob.setNumReduceTasks(1);

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

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

    FileInputFormat.setInputPaths(topNProductsJob, aggregationOutputPath);
    FileOutputFormat.setOutputPath(topNProductsJob, topNProductsOutputPath);

    if (fs.exists(topNProductsOutputPath)) {
        fs.delete(topNProductsOutputPath, true);
    }

    topNproductsJobSuccessful = topNProductsJob.waitForCompletion(true);

}
```

Aditya Joshi
001837740

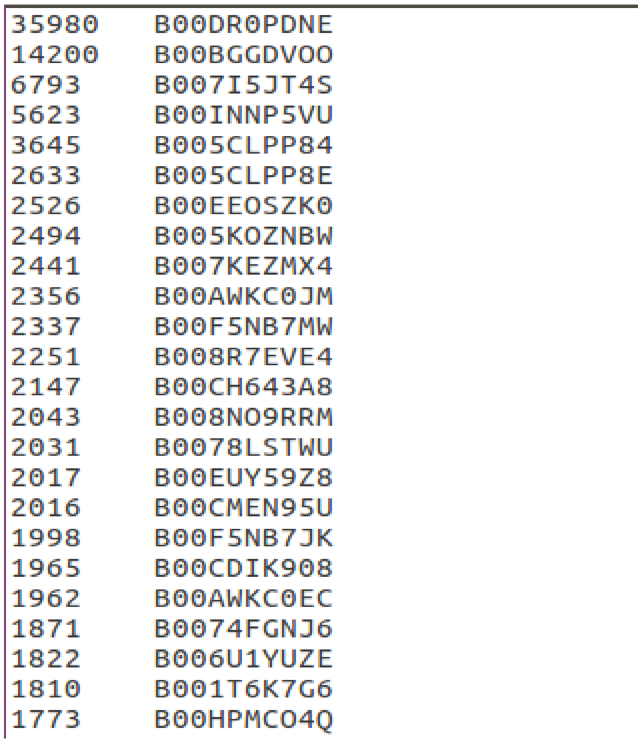
Reducer:



Driver.java   TopNProductsMapper.java   TopNProductsReducer.java

```java
package com.neu.AmazonReviewsAnalysis.TopNProducts;

import java.io.IOException;

public class TopNProductsReducer extends Reducer<IntWritable, Text, IntWritable, Text>{

    int count = 0;

    // default value = 10
    private int N = 10;

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

        for(Text val: value){
            if(count<N)
            {
                context.write(key,val);
            }
            count++;
        }
    }

    @Override
    protected void setup(Context context) throws IOException, InterruptedException {
        // default = 10
        this.N = context.getConfiguration().getInt("N", 10);
    }
}
```

Output: Top reviewed products sorted by count along with product ID.



```
35980   B00DR0PDNE
14200   B00BGGDVOO
6793    B007I5JT4S
5623    B00INNP5VU
3645    B005CLPP84
2633    B005CLPP8E
2526    B00EEOSZK0
2494    B005KOZNBW
2441    B007KEZMX4
2356    B00AWKC0JM
2337    B00F5NB7MW
2251    B008R7EVE4
2147    B00CH643A8
2043    B008NO9RRM
2031    B0078LSTWU
2017    B00EUY59Z8
2016    B00CMEN95U
1998    B00F5NB7JK
1965    B00CDIK908
1962    B00AWKC0EC
1871    B0074FGNJ6
1822    B006U1YUZE
1810    B001T6K7G6
1773    B00HPMCO4Q
```

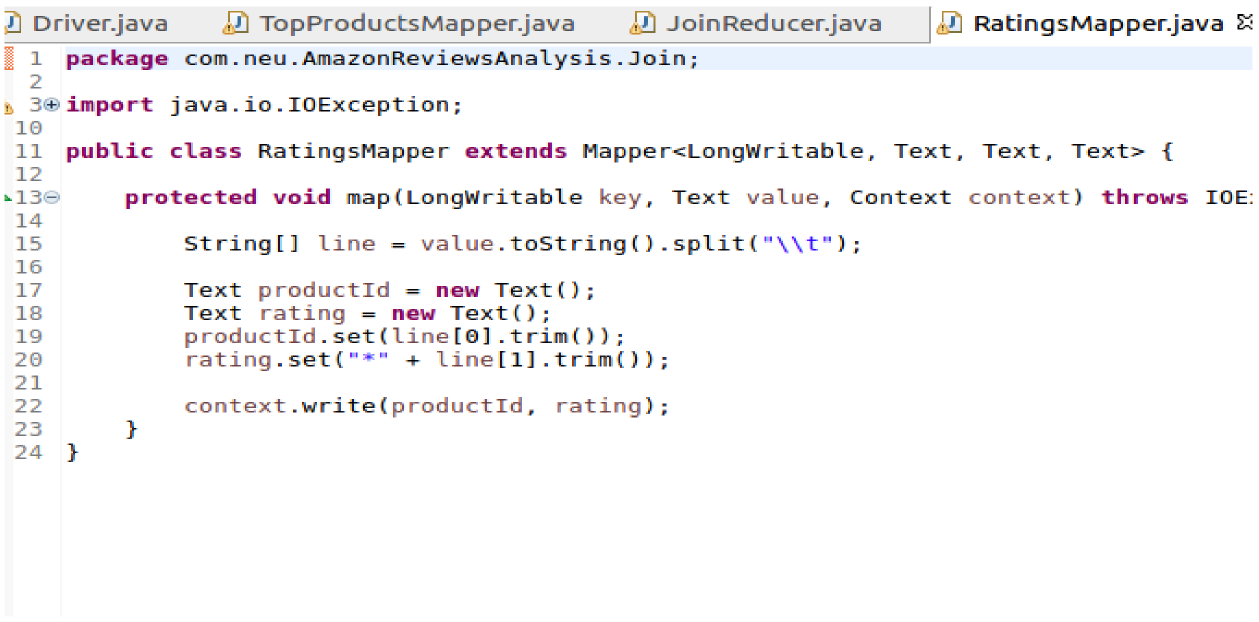
Aditya Joshi
001837740

5. Joins

An inner join is used to get product ID, review count and average rating.

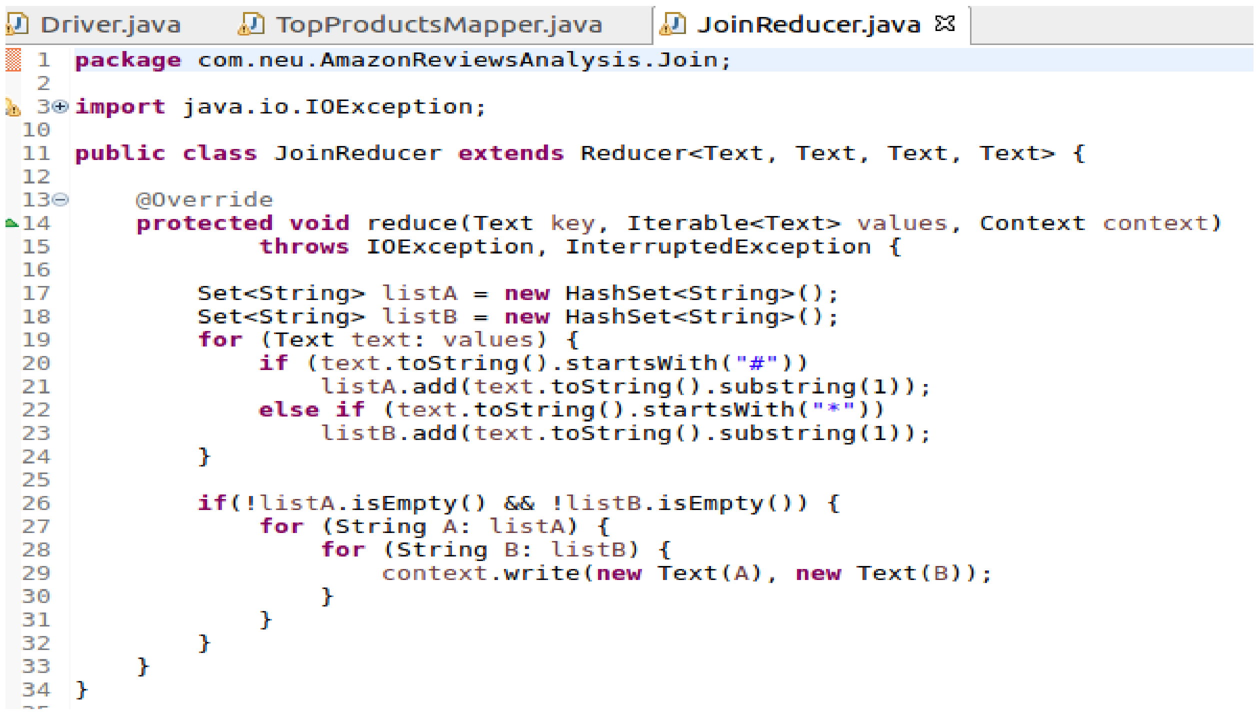
Multiple inputs are passed for join.

```
boolean joinJobSuccesful = false;
if (topNproductsJobSuccessful) {

    Job joinsJob = Job.getInstance(configuration, "Join");
    joinsJob.setJarByClass(Driver.class);

    MultipleInputs.addInputPath(joinsJob, topNProductsOutputPath,
            TextInputFormat.class, TopProductsMapper.class);
    MultipleInputs.addInputPath(joinsJob, summarizationOutputPath,
            TextInputFormat.class, RatingsMapper.class);
    FileOutputFormat.setOutputPath(joinsJob, joinOutputPath);

    joinsJob.setReducerClass(JoinReducer.class);

    joinsJob.setMapOutputKeyClass(Text.class);
    joinsJob.setMapOutputValueClass(Text.class);

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

    if (fs.exists(joinOutputPath)) {
        fs.delete(joinOutputPath, true);
    }

    joinJobSuccesful = joinsJob.waitForCompletion(true);

}
```



```
package com.neu.AmazonReviewsAnalysis.Join;

import java.io.IOException;

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

    protected void map(LongWritable key, Text value, Context context) throws IOE

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

        Text productId = new Text();
        Text rating = new Text();
        productId.set(line[0].trim());
        rating.set("*" + line[1].trim());

        context.write(productId, rating);
    }
}
```

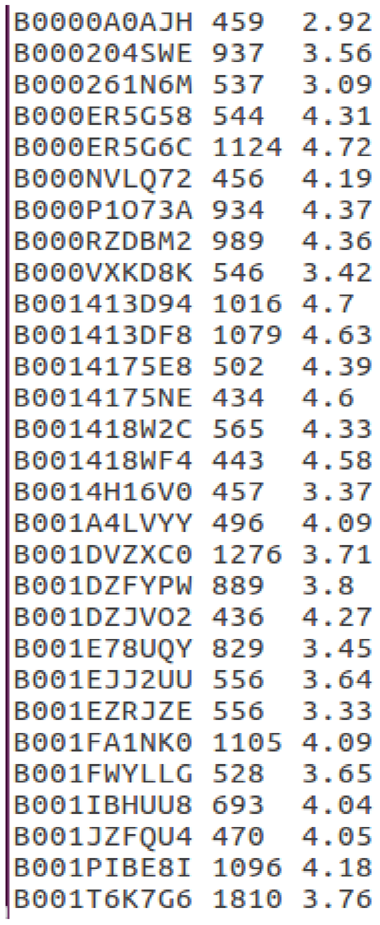
Aditya Joshi
001837740



Driver.java TopProductsMapper.java JoinReducer.java

```java
package com.neu.AmazonReviewsAnalysis.Join;

import java.io.IOException;

public class JoinReducer extends Reducer<Text, Text, Text, Text> {

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

        Set<String> listA = new HashSet<String>();
        Set<String> listB = new HashSet<String>();
        for (Text text: values) {
            if (text.toString().startsWith("#"))
                listA.add(text.toString().substring(1));
            else if (text.toString().startsWith("*"))
                listB.add(text.toString().substring(1));
        }

        if(!listA.isEmpty() && !listB.isEmpty()) {
            for (String A: listA) {
                for (String B: listB) {
                    context.write(new Text(A), new Text(B));
                }
            }
        }
    }
}
```

Output:



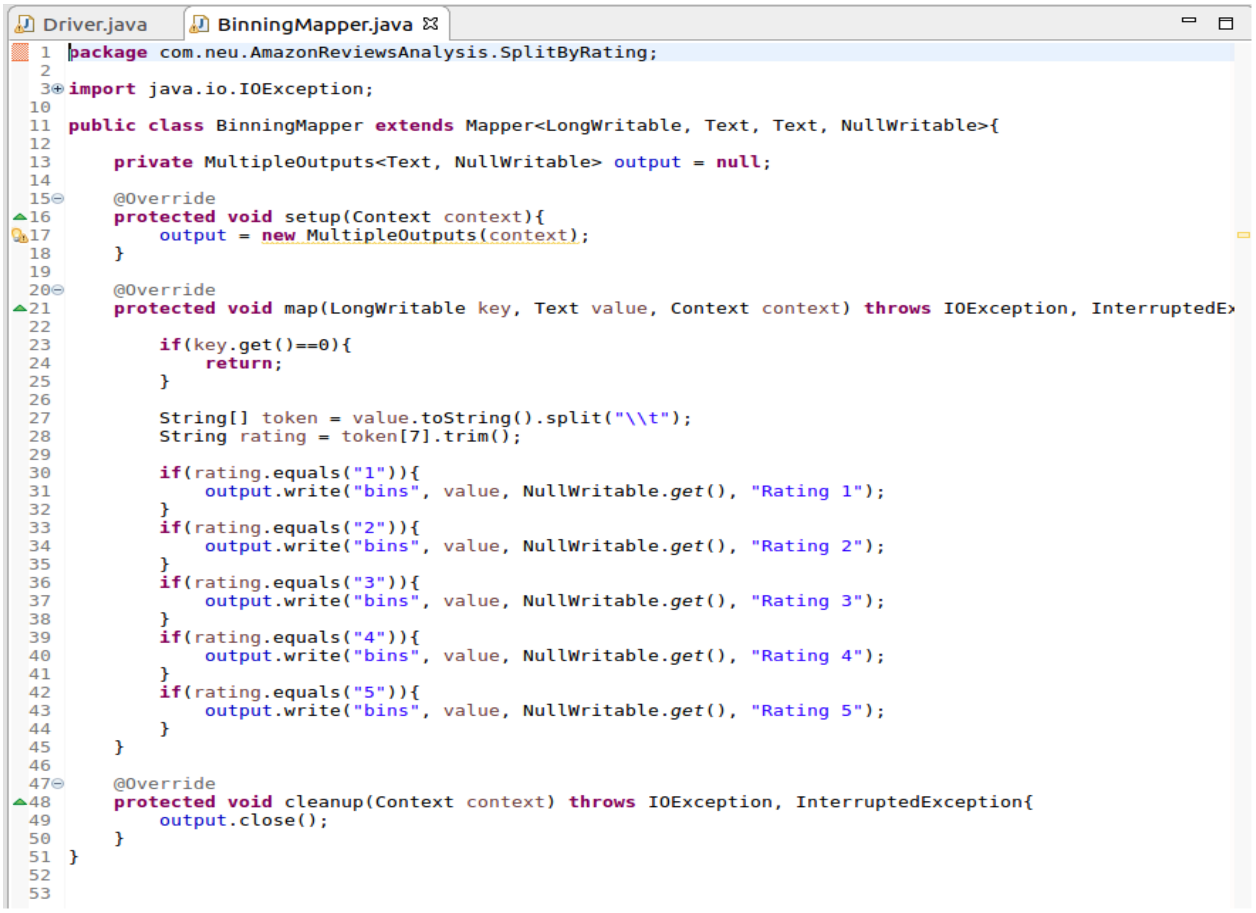
```
B0000A0AJH 459  2.92
B000204SWE 937  3.56
B000261N6M 537  3.09
B000ER5G58 544  4.31
B000ER5G6C 1124 4.72
B000NVLQ72 456  4.19
B000P1O73A 934  4.37
B000RZDBM2 989  4.36
B000VXKD8K 546  3.42
B001413D94 1016 4.7
B001413DF8 1079 4.63
B0014175E8 502  4.39
B0014175NE 434  4.6
B001418W2C 565  4.33
B001418WF4 443  4.58
B0014H16V0 457  3.37
B001A4LVYY 496  4.09
B001DVZXC0 1276 3.71
B001DZFYPW 889  3.8
B001DZJVO2 436  4.27
B001E78UQY 829  3.45
B001EJJ2UU 556  3.64
B001EZRJZE 556  3.33
B001FA1NK0 1105 4.09
B001FWYLLG 528  3.65
B001IBHUU8 693  4.04
B001JZFQU4 470  4.05
B001PIBE8I 1096 4.18
B001T6K7G6 1810 3.76
```

Aditya Joshi
001837740

6. Binning Pattern.

Here binning is used to split the reviews based on their ratings in separate bins. Only mapper is used to implement this. Multiple outputs are generated for each bin.

```
Job binningJob = Job.getInstance(configuration, "Binning");
binningJob.setJarByClass(Driver.class);

binningJob.setMapperClass(BinningMapper.class);
binningJob.setMapOutputKeyClass(Text.class);
binningJob.setMapOutputValueClass(NullWritable.class);

//No combiner, partitioner or reducer is used in this pattern!
binningJob.setNumReduceTasks(1);

FileInputFormat.setInputPaths(binningJob, inputPath);
FileOutputFormat.setOutputPath(binningJob, binningOutputPath);

if (fs.exists(binningOutputPath)) {
    fs.delete(binningOutputPath, true);
}

MultipleOutputs.addNamedOutput(binningJob, "bins", TextOutputFormat.class,
        Text.class, NullWritable.class);
MultipleOutputs.setCountersEnabled(binningJob, true);

binningJobSuccesful = binningJob.waitForCompletion(true);
```

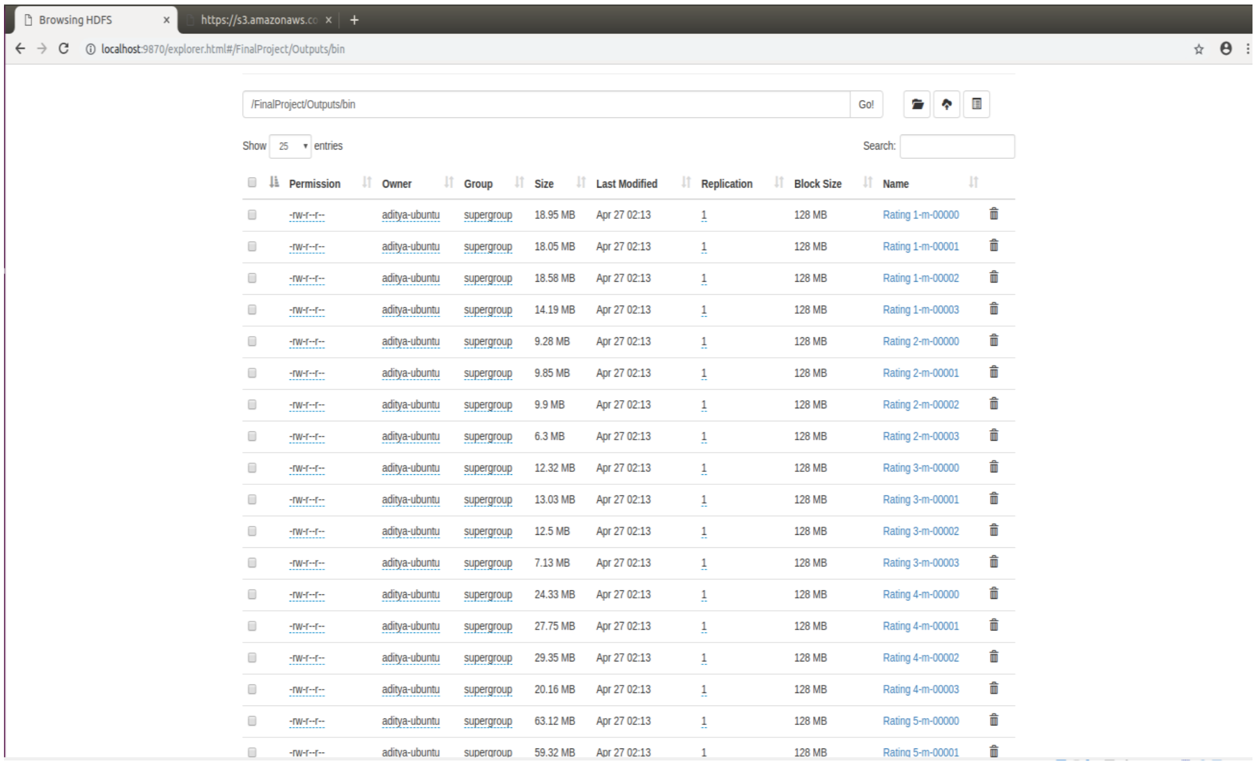


Driver.java | BinningMapper.java

```
package com.neu.AmazonReviewsAnalysis.SplitByRating;

import java.io.IOException;

public class BinningMapper extends Mapper<LongWritable, Text, Text, NullWritable>{

    private MultipleOutputs<Text, NullWritable> output = null;

    @Override
    protected void setup(Context context){
        output = new MultipleOutputs(context);
    }

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

        if(key.get()==0){
            return;
        }

        String[] token = value.toString().split("\\t");
        String rating = token[7].trim();

        if(rating.equals("1")){
            output.write("bins", value, NullWritable.get(), "Rating 1");
        }
        if(rating.equals("2")){
            output.write("bins", value, NullWritable.get(), "Rating 2");
        }
        if(rating.equals("3")){
            output.write("bins", value, NullWritable.get(), "Rating 3");
        }
        if(rating.equals("4")){
            output.write("bins", value, NullWritable.get(), "Rating 4");
        }
        if(rating.equals("5")){
            output.write("bins", value, NullWritable.get(), "Rating 5");
        }
    }

    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException{
        output.close();
    }
}
```

Aditya Joshi
001837740

Outputs:



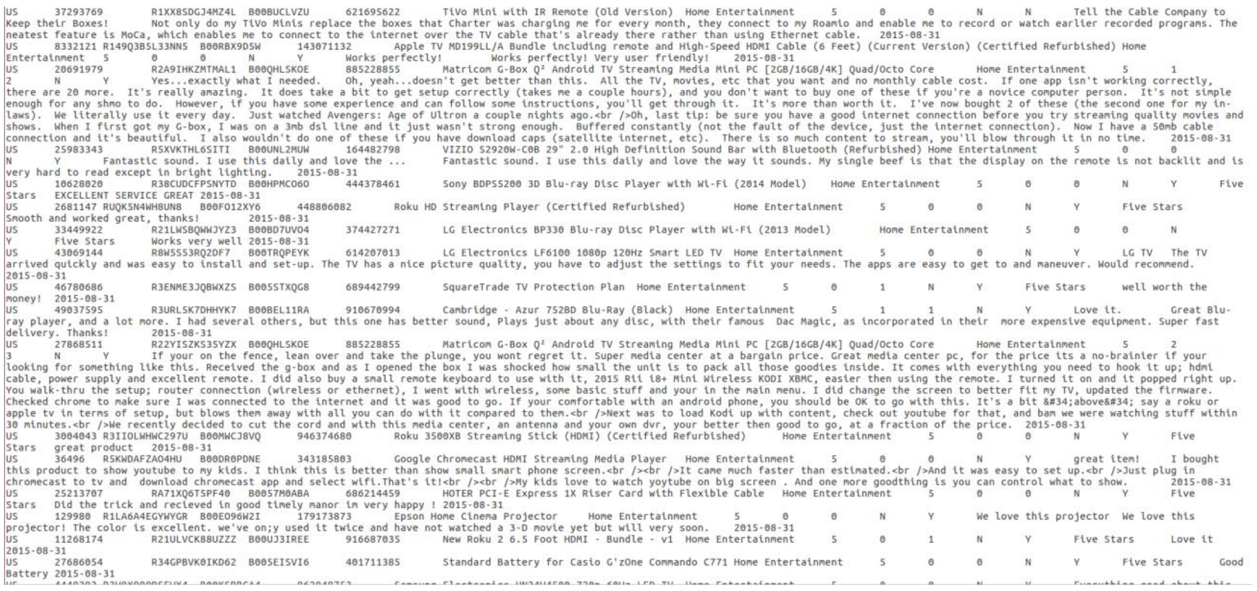
Browsing HDFS | https://s3.amazonaws.co

localhost:9870/explorer.html#/FinalProject/Outputs/bin

/FinalProject/Outputs/bin Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
	-rw-r--r--	aditya-ubuntu	supergroup	18.95 MB	Apr 27 02:13	1	128 MB	Rating 1-m-00000
	-rw-r--r--	aditya-ubuntu	supergroup	18.05 MB	Apr 27 02:13	1	128 MB	Rating 1-m-00001
	-rw-r--r--	aditya-ubuntu	supergroup	18.58 MB	Apr 27 02:13	1	128 MB	Rating 1-m-00002
	-rw-r--r--	aditya-ubuntu	supergroup	14.19 MB	Apr 27 02:13	1	128 MB	Rating 1-m-00003
	-rw-r--r--	aditya-ubuntu	supergroup	9.28 MB	Apr 27 02:13	1	128 MB	Rating 2-m-00000
	-rw-r--r--	aditya-ubuntu	supergroup	9.85 MB	Apr 27 02:13	1	128 MB	Rating 2-m-00001
	-rw-r--r--	aditya-ubuntu	supergroup	9.9 MB	Apr 27 02:13	1	128 MB	Rating 2-m-00002
	-rw-r--r--	aditya-ubuntu	supergroup	6.3 MB	Apr 27 02:13	1	128 MB	Rating 2-m-00003
	-rw-r--r--	aditya-ubuntu	supergroup	12.32 MB	Apr 27 02:13	1	128 MB	Rating 3-m-00000
	-rw-r--r--	aditya-ubuntu	supergroup	13.03 MB	Apr 27 02:13	1	128 MB	Rating 3-m-00001
	-rw-r--r--	aditya-ubuntu	supergroup	12.5 MB	Apr 27 02:13	1	128 MB	Rating 3-m-00002
	-rw-r--r--	aditya-ubuntu	supergroup	7.13 MB	Apr 27 02:13	1	128 MB	Rating 3-m-00003
	-rw-r--r--	aditya-ubuntu	supergroup	24.33 MB	Apr 27 02:13	1	128 MB	Rating 4-m-00000
	-rw-r--r--	aditya-ubuntu	supergroup	27.75 MB	Apr 27 02:13	1	128 MB	Rating 4-m-00001
	-rw-r--r--	aditya-ubuntu	supergroup	29.35 MB	Apr 27 02:13	1	128 MB	Rating 4-m-00002
	-rw-r--r--	aditya-ubuntu	supergroup	20.16 MB	Apr 27 02:13	1	128 MB	Rating 4-m-00003
	-rw-r--r--	aditya-ubuntu	supergroup	63.12 MB	Apr 27 02:13	1	128 MB	Rating 5-m-00000
	-rw-r--r--	aditya-ubuntu	supergroup	59.32 MB	Apr 27 02:13	1	128 MB	Rating 5-m-00001

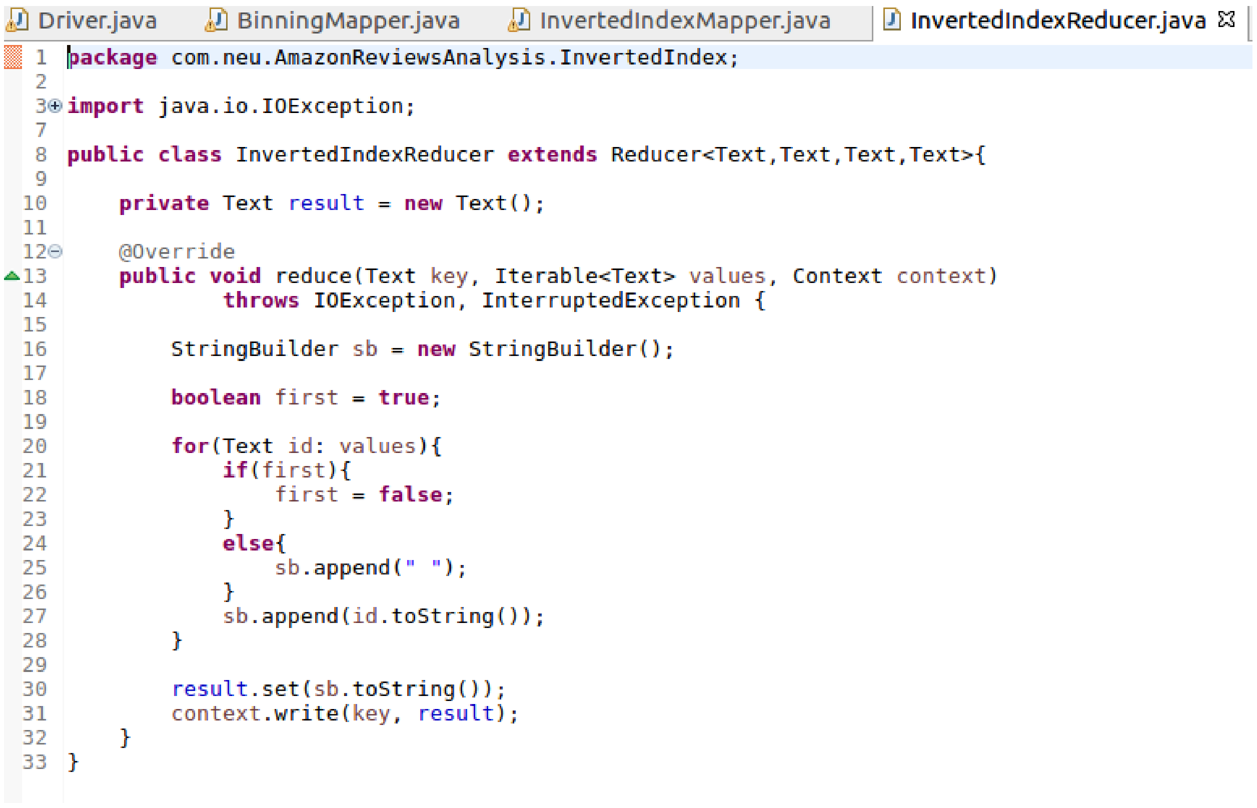


```
US      37293769        R1XX8SDGJ4MZ4L  B00BUCLVZU      621695622       TiVo Mini with IR Remote (Old Version)  Home Entertainment      5       0       0       N       N       Tell the Cable Company to
Keep their Boxes!       Not only do my TiVo Minis replace the boxes that Charter was charging me for every month, they connect to my Roamio and enable me to record or watch earlier recorded programs. The
neatest feature is MoCa, which enables me to connect to the internet over the TV cable that's already there rather than using Ethernet cable.   2015-08-31
US      8332121 R149Q3B5L33NN5  B00RBX9D5W      143071132       Apple TV MD199LL/A Bundle including remote and High-Speed HDMI Cable (6 Feet) (Current Version) (Certified Refurbished) Home
Entertainment   5       0       0       N       Y       Works perfectly!        Works perfectly! Very user friendly!    2015-08-31
US      20691979        R2A9IHKZMTMAL1  B00QHLSKOE      885228855       Matricom G-Box Q² Android TV Streaming Media Mini PC [2GB/16GB/4K] Quad/Octo Core       Home Entertainment      5       1
2       N       Y       Yes...exactly what I needed.    Oh, yeah...doesn't get better than this.  All the TV, movies, etc that you want and no monthly cable cost.  If one app isn't working correctly,
there are 20 more.  It's really amazing.  It does take a bit to get setup correctly (takes me a couple hours), and you don't want to buy one of these if you're a novice computer person.  It's not simple
enough for any shmo to do.  However, if you have some experience and can follow some instructions, you'll get through it.  It's more than worth it.  I've now bought 2 of these (the second one for my in-
laws).  We literally use it every day.  Just watched Avengers: Age of Ultron a couple nights ago.<br />Oh, last tip: be sure you have a good internet connection before you try streaming quality movies and
shows.  When I first got my G-box, I was on a 3mb dsl line and it just wasn't strong enough.  Buffered constantly (not the fault of the device, just the internet connection).  Now I have a 50mb cable
connection and it's beautiful.  I also wouldn't do one of these if you have download caps (satellite internet, etc).  There is so much content to stream, you'll blow through it in no time.    2015-08-31
US      25983343        R5XVKTHL6SITI   B00UNL2MUW      164482798       VIZIO S2920W-C0B 29" 2.0 High Definition Sound Bar with Bluetooth (Refurbished) Home Entertainment      5       0       0
N       Y       Fantastic sound. I use this daily and love the ...      Fantastic sound. I use this daily and love the way it sounds. My single beef is that the display on the remote is not backlit and is
very hard to read except in bright lighting.    2015-08-31
US      10628020        R38CUDCFPSNYTD  B00HPMCO6O      444378461       Sony BDPS5200 3D Blu-ray Disc Player with Wi-Fi (2014 Model)    Home Entertainment      5       0       0       N       Y       Five
Stars   EXCELLENT SERVICE GREAT 2015-08-31
US      2681147 RUQK5N4WH8UN8   B00FO12XY6      448806082       Roku HD Streaming Player (Certified Refurbished)        Home Entertainment      5       0       0       N       Y       Five Stars
Smooth and worked great, thanks!        2015-08-31
US      33449922        R21LWSBQWWJYZ3  B00BD7UVO4      374427271       LG Electronics BP330 Blu-ray Disc Player with Wi-Fi (2013 Model)        Home Entertainment      5       0       0       N
Y       Five Stars      Works very well 2015-08-31
US      43069144        R8W5S53RQ2DF7   B00TRQPEYK      614207013       LG Electronics LF6100 1080p 120Hz Smart LED TV  Home Entertainment      5       0       0       N       Y       LG TV   The TV
arrived quickly and was easy to install and set-up. The TV has a nice picture quality, you have to adjust the settings to fit your needs. The apps are easy to get to and maneuver. Would recommend.
2015-08-31
US      46780686        R3ENME3JQBWXZS  B005STXQG8      689442799       SquareTrade TV Protection Plan  Home Entertainment      5       0       1       N       Y       Five Stars      well worth the
money!  2015-08-31
US      49037595        R3URL5K7DHHYK7  B00BEL11RA      910670994       Cambridge - Azur 752BD Blu-Ray (Black)  Home Entertainment      5       1       1       N       Y       Love it.        Great Blu-
ray player, and a lot more. I had several others, but this one has better sound, Plays just about any disc, with their famous  Dac Magic, as incorporated in their  more expensive equipment. Super fast
delivery. Thanks!       2015-08-31
US      27868511        R22YISZKS35YZX  B00QHLSKOE      885228855       Matricom G-Box Q² Android TV Streaming Media Mini PC [2GB/16GB/4K] Quad/Octo Core       Home Entertainment      5       2
3       N       Y       If your on the fence, lean over and take the plunge, you wont regret it. Super media center at a bargain price. Great media center pc, for the price its a no-brainier if your
looking for something like this. Received the g-box and as I opened the box I was shocked how small the unit is to pack all those goodies inside. It comes with everything you need to hook it up; hdmi
cable, power supply and excellent remote. I did also buy a small remote keyboard to use with it, 2015 Rii i8+ Mini Wireless KODI XBMC, easier then using the remote. I turned it on and it popped right up.
You walk-thru the setup; router connection (wireless or ethernet), I went with wireless, some basic stuff and your in the main menu. I did change the screen to better fit my TV, updated the firmware.
Checked chrome to make sure I was connected to the internet and it was good to go. If your comfortable with an android phone, you should be OK to go with this. It's a bit &#34;above&#34; say a roku or
apple tv in terms of setup, but blows them away with all you can do with it compared to them.<br />Next was to load Kodi up with content, check out youtube for that, and bam we were watching stuff within
30 minutes.<br />We recently decided to cut the cord and with this media center, an antenna and your own dvr, your better then good to go, at a fraction of the price.  2015-08-31
US      3004043 R3IIOLWHWC297U  B00MWCJ8VQ      946374680       Roku 3500XB Streaming Stick (HDMI) (Certified Refurbished)      Home Entertainment      5       0       0       N       Y       Five
Stars   great product   2015-08-31
US      36496   R5KWDAFZAO4HU   B00DR0PDNE      343185803       Google Chromecast HDMI Streaming Media Player   Home Entertainment      5       0       0       N       Y       great item!     I bought
this product to show youtube to my kids. I think this is better than show small smart phone screen.<br /><br />It came much faster than estimated.<br />And it was easy to set up.<br />Just plug in
chromecast to tv and  download chromecast app and select wifi.That's it!<br /><br />My kids love to watch yoytube on big screen . And one more goodthing is you can control what to show.      2015-08-31
US      25213707        RA71XQ6T5PF40   B0057M0ABA      686214459       HOTER PCI-E Express 1X Riser Card with Flexible Cable   Home Entertainment      5       0       0       N       Y       Five
Stars   Did the trick and recieved in good timely manor im very happy ! 2015-08-31
US      129980  R1LA6A4EGYWYGR  B00EO96W2I      179173873       Epson Home Cinema Projector     Home Entertainment      5       0       0       N       Y       We love this projector  We love this
projector! The color is excellent. we've on;y used it twice and have not watched a 3-D movie yet but will very soon.    2015-08-31
US      11268174        R21ULVCK88UZZZ  B00UJ3IREE      916687035       New Roku 2 6.5 Foot HDMI - Bundle - v1  Home Entertainment      5       0       1       N       Y       Five Stars      Love it
2015-08-31
US      27686054        R34GPBVK0IKD62  B005EISVI6      401711385       Standard Battery for Casio G'zOne Commando C771 Home Entertainment      5       0       0       N       Y       Five Stars      Good
Battery 2015-08-31
```

Aditya Joshi
001837740

7. Inverted Index Pattern

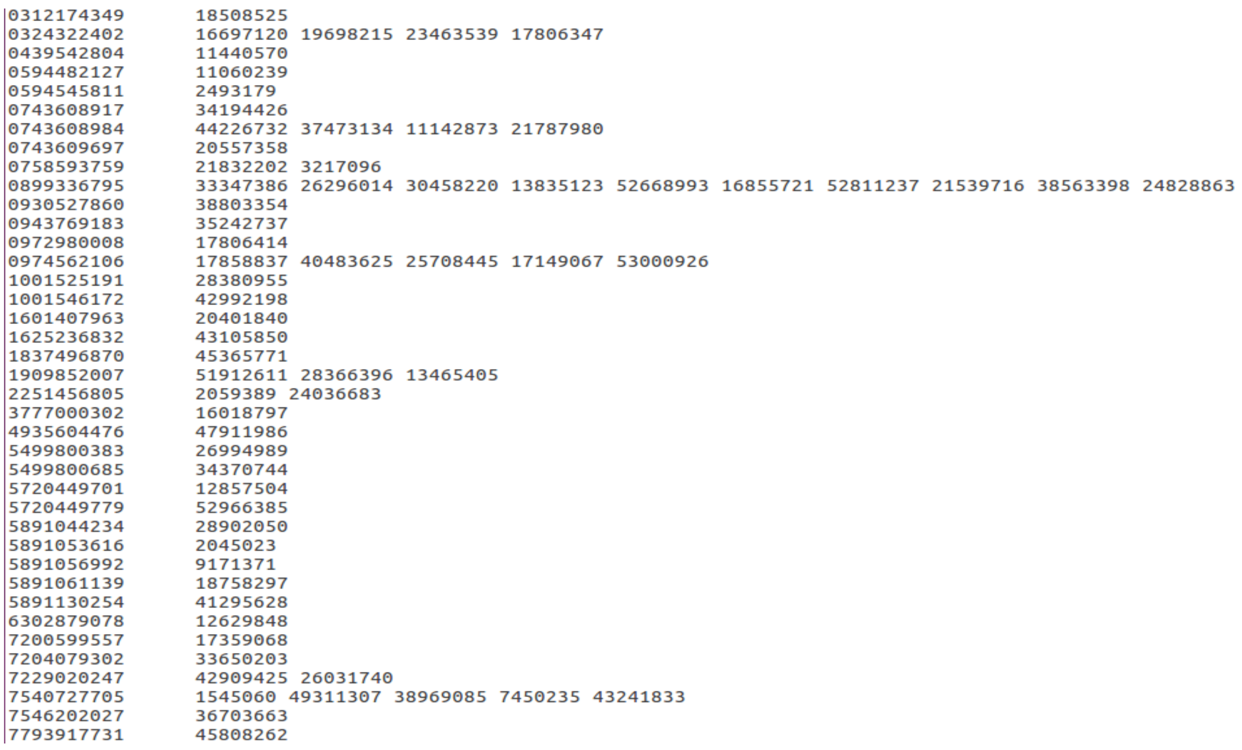
Used inverted index pattern to find each user who has reviewed the product.



Driver.java   BinningMapper.java   InvertedIndexMapper.java   InvertedIndexReducer.java

```java
package com.neu.AmazonReviewsAnalysis.InvertedIndex;

import java.io.IOException;

public class InvertedIndexReducer extends Reducer<Text,Text,Text,Text>{

    private Text result = new Text();

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

        StringBuilder sb = new StringBuilder();

        boolean first = true;

        for(Text id: values){
            if(first){
                first = false;
            }
            else{
                sb.append(" ");
            }
            sb.append(id.toString());
        }

        result.set(sb.toString());
        context.write(key, result);
    }
}
```

Output:



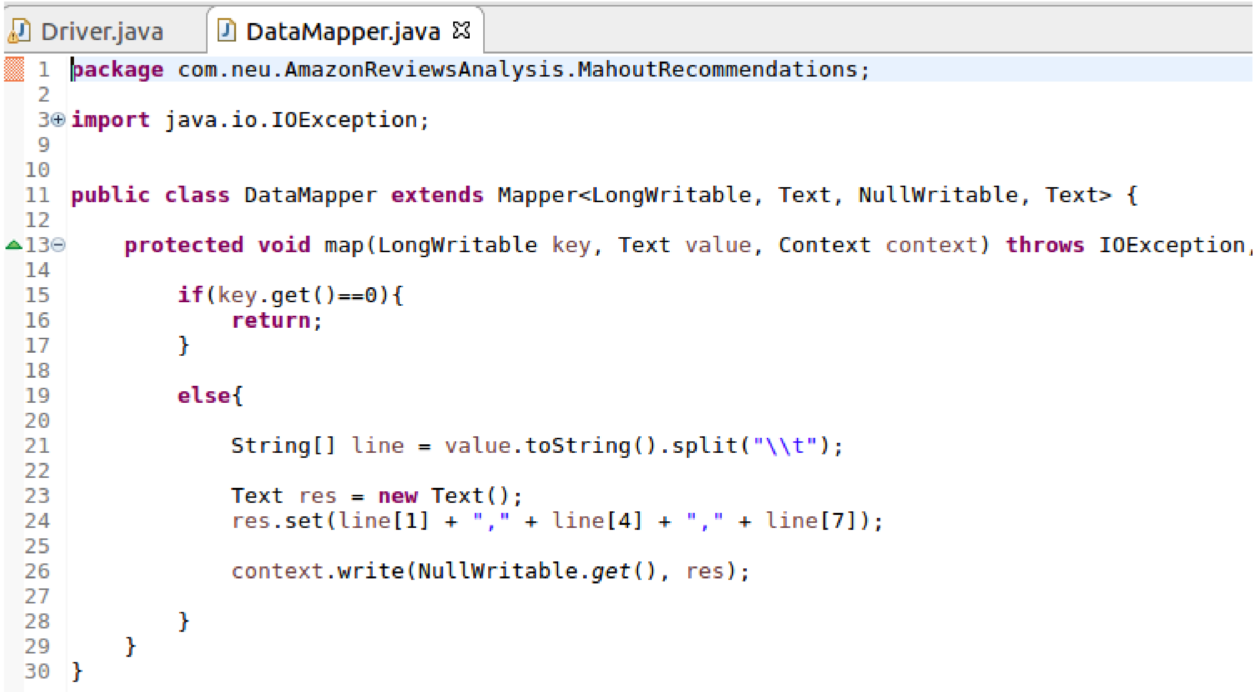
```
0312174349        18508525
0324322402        16697120 19698215 23463539 17806347
0439542804        11440570
0594482127        11060239
0594545811        2493179
0743608917        34194426
0743608984        44226732 37473134 11142873 21787980
0743609697        20557358
0758593759        21832202 3217096
0899336795        33347386 26296014 30458220 13835123 52668993 16855721 52811237 21539716 38563398 24828863
0930527860        38803354
0943769183        35242737
0972980008        17806414
0974562106        17858837 40483625 25708445 17149067 53000926
1001525191        28380955
1001546172        42992198
1601407963        20401840
1625236832        43105850
1837496870        45365771
1909852007        51912611 28366396 13465405
2251456805        2059389 24036683
3777000302        16018797
4935604476        47911986
5499800383        26994989
5499800685        34370744
5720449701        12857504
5720449779        52966385
5891044234        28902050
5891053616        2045023
5891056992        9171371
5891061139        18758297
5891130254        41295628
6302879078        12629848
7200599557        17359068
7204079302        33650203
7229020247        42909425 26031740
7540727705        1545060 49311307 38969085 7450235 43241833
7546202027        36703663
7793917731        45808262
```

Aditya Joshi
001837740

8. Mahout Recommendations

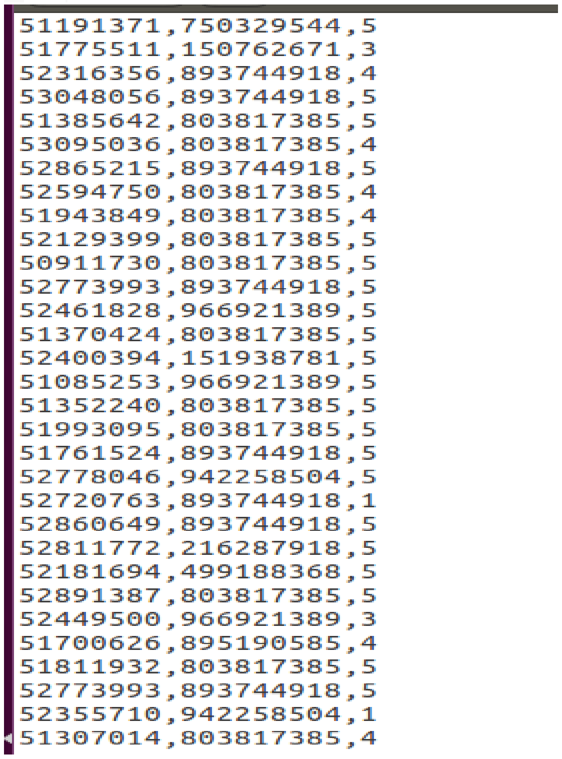
Data Cleaning

For mahout recommendation systems, we need limited data. We need User ID, Product ID and rating. We'll be using mahout's user recommender to recommend products to users by creating a model.



Driver.java | DataMapper.java

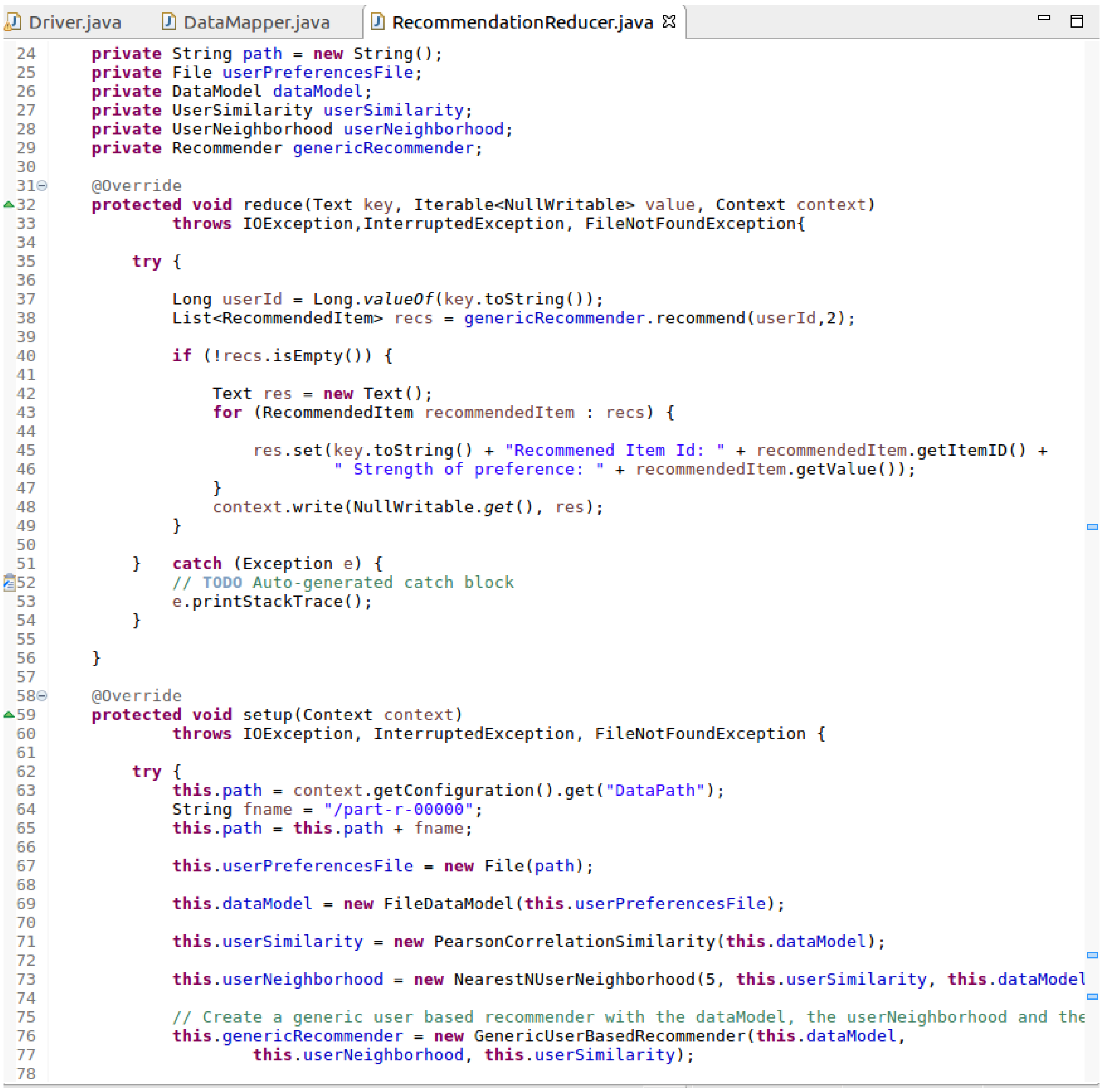
```java
package com.neu.AmazonReviewsAnalysis.MahoutRecommendations;

import java.io.IOException;


public class DataMapper extends Mapper<LongWritable, Text, NullWritable, Text> {

    protected void map(LongWritable key, Text value, Context context) throws IOException,

        if(key.get()==0){
            return;
        }

        else{

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

            Text res = new Text();
            res.set(line[1] + "," + line[4] + "," + line[7]);

            context.write(NullWritable.get(), res);

        }
    }
}
```



```
51191371,750329544,5
51775511,150762671,3
52316356,893744918,4
53048056,893744918,5
51385642,803817385,5
53095036,803817385,4
52865215,893744918,5
52594750,803817385,4
51943849,803817385,4
52129399,803817385,5
50911730,803817385,5
52773993,893744918,5
52461828,966921389,5
51370424,803817385,5
52400394,151938781,5
51085253,966921389,5
51352240,803817385,5
51993095,803817385,5
51761524,893744918,5
52778046,942258504,5
52720763,893744918,1
52860649,893744918,5
52811772,216287918,5
52181694,499188368,5
52891387,803817385,5
52449500,966921389,3
51700626,895190585,4
51811932,803817385,5
52773993,893744918,5
52355710,942258504,1
51307014,803817385,4
```

Aditya Joshi
001837740

Mahout user recommender

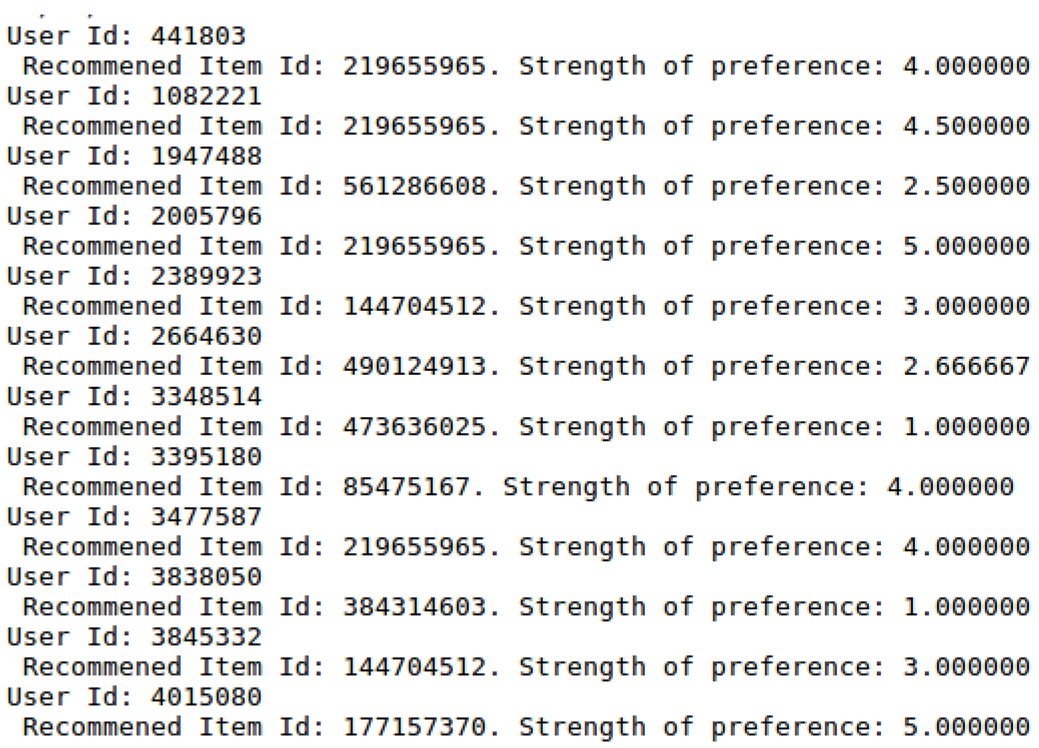

Driver.java DataMapper.java RecommendationReducer.java

```
    private String path = new String();
    private File userPreferencesFile;
    private DataModel dataModel;
    private UserSimilarity userSimilarity;
    private UserNeighborhood userNeighborhood;
    private Recommender genericRecommender;

    @Override
    protected void reduce(Text key, Iterable<NullWritable> value, Context context)
            throws IOException,InterruptedException, FileNotFoundException{

        try {

            Long userId = Long.valueOf(key.toString());
            List<RecommendedItem> recs = genericRecommender.recommend(userId,2);

            if (!recs.isEmpty()) {

                Text res = new Text();
                for (RecommendedItem recommendedItem : recs) {

                    res.set(key.toString() + "Recommened Item Id: " + recommendedItem.getItemID() +
                            " Strength of preference: " + recommendedItem.getValue());
                }
                context.write(NullWritable.get(), res);
            }

        }   catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    @Override
    protected void setup(Context context)
            throws IOException, InterruptedException, FileNotFoundException {

        try {
            this.path = context.getConfiguration().get("DataPath");
            String fname = "/part-r-00000";
            this.path = this.path + fname;

            this.userPreferencesFile = new File(path);

            this.dataModel = new FileDataModel(this.userPreferencesFile);

            this.userSimilarity = new PearsonCorrelationSimilarity(this.dataModel);

            this.userNeighborhood = new NearestNUserNeighborhood(5, this.userSimilarity, this.dataModel

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

```

Aditya Joshi
001837740

Outputs:

Here we see a recommended product ID and strength of preference for users.



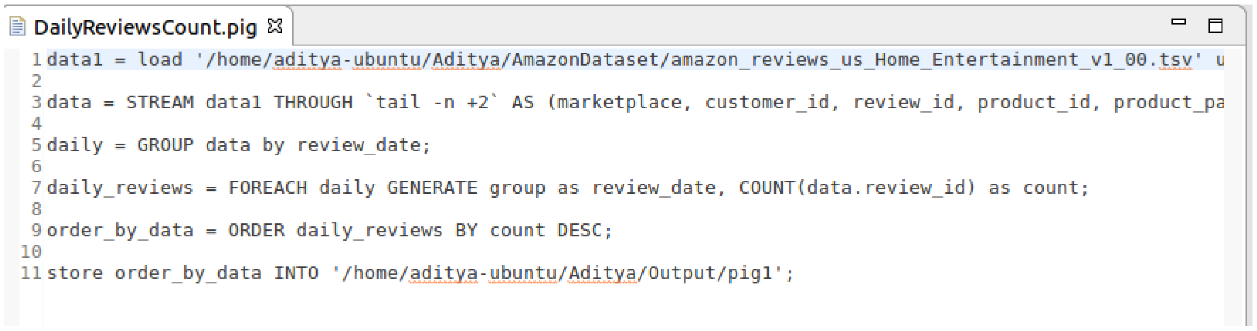
```
User Id: 441803
 Recommened Item Id: 219655965. Strength of preference: 4.000000
User Id: 1082221
 Recommened Item Id: 219655965. Strength of preference: 4.500000
User Id: 1947488
 Recommened Item Id: 561286608. Strength of preference: 2.500000
User Id: 2005796
 Recommened Item Id: 219655965. Strength of preference: 5.000000
User Id: 2389923
 Recommened Item Id: 144704512. Strength of preference: 3.000000
User Id: 2664630
 Recommened Item Id: 490124913. Strength of preference: 2.666667
User Id: 3348514
 Recommened Item Id: 473636025. Strength of preference: 1.000000
User Id: 3395180
 Recommened Item Id: 85475167. Strength of preference: 4.000000
User Id: 3477587
 Recommened Item Id: 219655965. Strength of preference: 4.000000
User Id: 3838050
 Recommened Item Id: 384314603. Strength of preference: 1.000000
User Id: 3845332
 Recommened Item Id: 144704512. Strength of preference: 3.000000
User Id: 4015080
 Recommened Item Id: 177157370. Strength of preference: 5.000000
```

Aditya Joshi
001837740

9. Pig Analysis Script

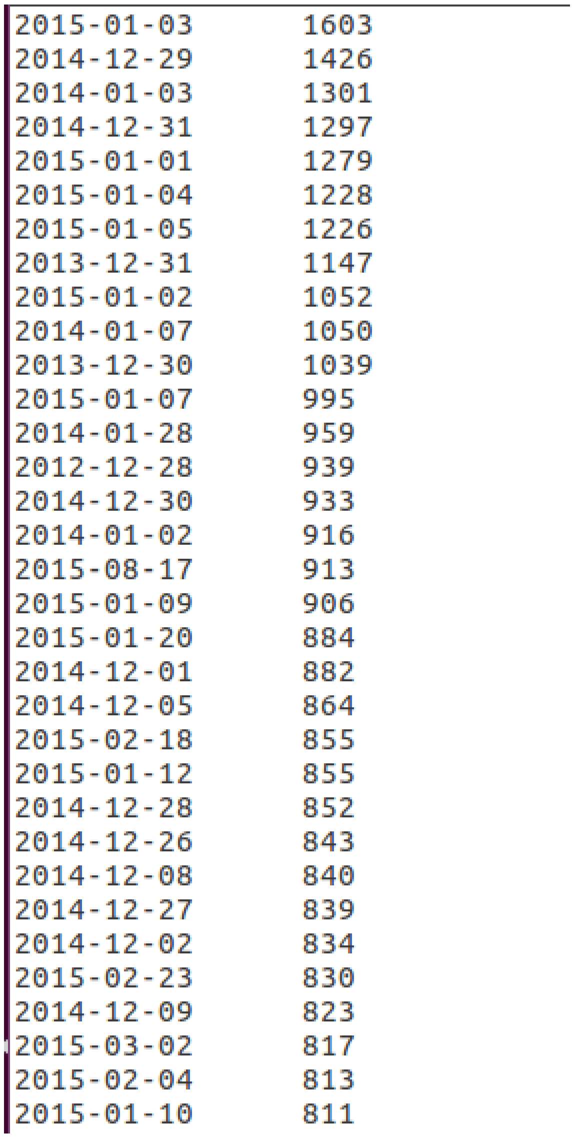
Daily reviews count

Pig script to get a count of reviews daily:


DailyReviewsCount.pig

```
data1 = load '/home/aditya-ubuntu/Aditya/AmazonDataset/amazon_reviews_us_Home_Entertainment_v1_00.tsv' u

data = STREAM data1 THROUGH `tail -n +2` AS (marketplace, customer_id, review_id, product_id, product_pa

daily = GROUP data by review_date;

daily_reviews = FOREACH daily GENERATE group as review_date, COUNT(data.review_id) as count;

order_by_data = ORDER daily_reviews BY count DESC;

store order_by_data INTO '/home/aditya-ubuntu/Aditya/Output/pig1';
```

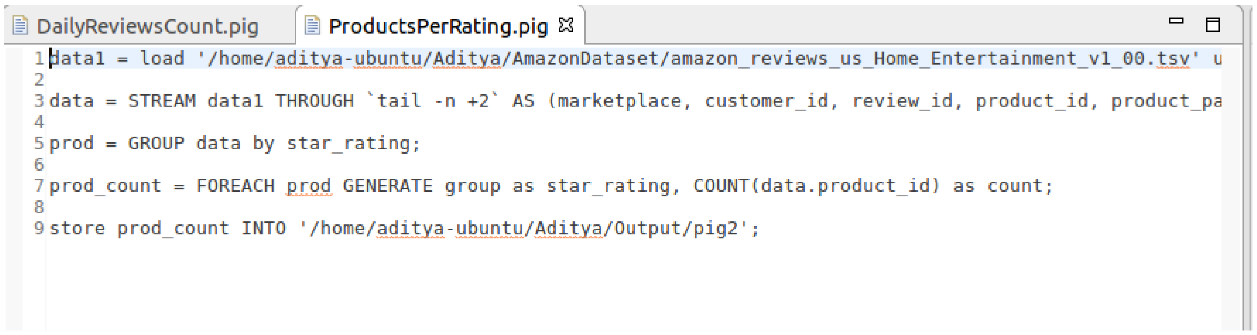
Output:



```
2015-01-03      1603
2014-12-29      1426
2014-01-03      1301
2014-12-31      1297
2015-01-01      1279
2015-01-04      1228
2015-01-05      1226
2013-12-31      1147
2015-01-02      1052
2014-01-07      1050
2013-12-30      1039
2015-01-07      995
2014-01-28      959
2012-12-28      939
2014-12-30      933
2014-01-02      916
2015-08-17      913
2015-01-09      906
2015-01-20      884
2014-12-01      882
2014-12-05      864
2015-02-18      855
2015-01-12      855
2014-12-28      852
2014-12-26      843
2014-12-08      840
2014-12-27      839
2014-12-02      834
2015-02-23      830
2014-12-09      823
2015-03-02      817
2015-02-04      813
2015-01-10      811
```

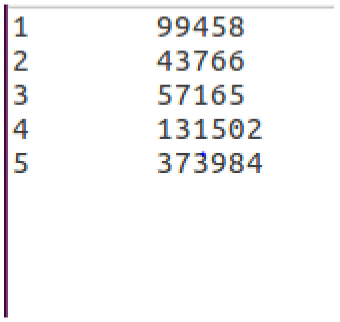
Aditya Joshi
001837740

Products per ratings



```
data1 = load '/home/aditya-ubuntu/Aditya/AmazonDataset/amazon_reviews_us_Home_Entertainment_v1_00.tsv' u

data = STREAM data1 THROUGH `tail -n +2` AS (marketplace, customer_id, review_id, product_id, product_pa

prod = GROUP data by star_rating;

prod_count = FOREACH prod GENERATE group as star_rating, COUNT(data.product_id) as count;

store prod_count INTO '/home/aditya-ubuntu/Aditya/Output/pig2';
```

Output:



```
1         99458
2         43766
3         57165
4         131502
5         373984
```

Aditya Joshi
001837740

Source Code:

Driver.java

```
package com.neu.AmazonReviewsAnalysis;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
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.NullWritable;
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.MultipleInputs;
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 com.neu.AmazonReviewsAnalysis.InvertedIndex.InvertedIndexMapper;
import com.neu.AmazonReviewsAnalysis.InvertedIndex.InvertedIndexReducer;
import com.neu.AmazonReviewsAnalysis.Join.JoinReducer;
import com.neu.AmazonReviewsAnalysis.Join.RatingsMapper;
import com.neu.AmazonReviewsAnalysis.Join.TopProductsMapper;
import com.neu.AmazonReviewsAnalysis.MahoutRecommendations.DataMapper;
import
com.neu.AmazonReviewsAnalysis.MahoutRecommendations.ProductRecommendation;
import
com.neu.AmazonReviewsAnalysis.MahoutRecommendations.RecommendationMapper;
import
com.neu.AmazonReviewsAnalysis.MahoutRecommendations.RecommendationReducer;
import com.neu.AmazonReviewsAnalysis.SplitByRating.BinningMapper;
import com.neu.AmazonReviewsAnalysis.Summarization.AggregateMapper;
import com.neu.AmazonReviewsAnalysis.Summarization.AggregateReducer;
import com.neu.AmazonReviewsAnalysis.Summarization.AverageMapper;
import com.neu.AmazonReviewsAnalysis.Summarization.AverageReducer;
import com.neu.AmazonReviewsAnalysis.Summarization.CountAverageTuple;
import com.neu.AmazonReviewsAnalysis.TopNProducts.CountComparator;
```

Aditya Joshi
001837740

```
import com.neu.AmazonReviewsAnalysis.TopNProducts.TopNProductsMapper;
import com.neu.AmazonReviewsAnalysis.TopNProducts.TopNProductsReducer;



/**
 * @author aditya
 *
 */
public class Driver  {

    private static final Log logger = LogFactory.getLog(Driver.class);

    public static void main( String[] args ) {

            if (args.length != 9) {
        logger.error("Usage: com.neu.AmazonReviewsAnalysis.Driver path arguments
missing");
        System.exit(1);
     }

            try {
                    Configuration configuration = new Configuration();
                    FileSystem fs = FileSystem.get(configuration);

                    // Input/Output Path for MapReduce Jobs
                    Path inputPath = new Path(args[0]);
                    Path summarizationOutputPath = new Path(args[1]);
                    Path aggregationOutputPath = new Path(args[2]);
                    Path topNProductsOutputPath = new Path(args[3]);
                    Path joinOutputPath = new Path(args[4]);
                    Path binningOutputPath = new Path(args[5]);
                    Path invertedIndexOutputPath = new Path(args[6]);
                    Path dataCleanOutputPath = new Path(args[7]);
                    Path recommedationOutputPath = new Path(args[8]);

                    // Define MapReduce Job
                    Job summarizationJob = Job.getInstance(configuration, "Product Ratings
Average");
                    summarizationJob.setJarByClass(Driver.class);

                    // Set Input and Output locations for summarizationJob Job
```

```
			FileInputFormat.setInputPaths(summarizationJob, inputPath);
			FileOutputFormat.setOutputPath(summarizationJob,
summarizationOutputPath);

			// Set Input and Output formats for summarizationJob Job
			summarizationJob.setInputFormatClass(TextInputFormat.class);
			summarizationJob.setOutputFormatClass(TextOutputFormat.class);

			// Set Mapper/Combiner/Reducer classes for MeanStdHelpfulReviews
Job
			summarizationJob.setMapperClass(AverageMapper.class);
			summarizationJob.setReducerClass(AverageReducer.class);
			summarizationJob.setCombinerClass(AverageReducer.class);

			// Set key/values classes
			summarizationJob.setOutputKeyClass(Text.class);
			summarizationJob.setOutputValueClass(CountAverageTuple.class);

			// Check if the output path is available or not
			if (fs.exists(summarizationOutputPath)) {
				fs.delete(summarizationOutputPath, true);
			}
			boolean issummarizationJobSuccessful =
(summarizationJob.waitForCompletion(true));

			boolean isAggregationJobSuccesful = false;
			if (issummarizationJobSuccessful) {

				Job aggregationJob = Job.getInstance(configuration, "Review
count");
				aggregationJob.setJarByClass(Driver.class);

				aggregationJob.setMapperClass(AggregateMapper.class);
				aggregationJob.setCombinerClass(AggregateReducer.class);
				aggregationJob.setReducerClass(AggregateReducer.class);

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

				aggregationJob.setOutputKeyClass(Text.class);
				aggregationJob.setOutputValueClass(IntWritable.class);
				FileInputFormat.addInputPath(aggregationJob, inputPath);
				FileOutputFormat.setOutputPath(aggregationJob,
aggregationOutputPath);

				if (fs.exists(aggregationOutputPath)) {
					fs.delete(aggregationOutputPath, true);
				}

				isAggregationJobSuccesful =
aggregationJob.waitForCompletion(true);

			}

			boolean topNproductsJobSuccessful = false;
			if (issummarizationJobSuccessful) {

				Job topNProductsJob = Job.getInstance(configuration, "Top N
Rated Products");
				topNProductsJob.setJarByClass(Driver.class);

				int N = 200;
				topNProductsJob.getConfiguration().setInt("N", N);

				topNProductsJob.setInputFormatClass(TextInputFormat.class);

	topNProductsJob.setOutputFormatClass(TextOutputFormat.class);

				topNProductsJob.setMapperClass(TopNProductsMapper.class);

	topNProductsJob.setSortComparatorClass(CountComparator.class);
				topNProductsJob.setReducerClass(TopNProductsReducer.class);
				topNProductsJob.setNumReduceTasks(1);

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

				topNProductsJob.setOutputKeyClass(IntWritable.class);
				topNProductsJob.setOutputValueClass(Text.class);
```

```
                        FileInputFormat.setInputPaths(topNProductsJob,
aggregationOutputPath);
                        FileOutputFormat.setOutputPath(topNProductsJob,
topNProductsOutputPath);

                        if (fs.exists(topNProductsOutputPath)) {
                            fs.delete(topNProductsOutputPath, true);
                        }

                        topNproductsJobSuccessful =
topNProductsJob.waitForCompletion(true);

                    }

                    boolean joinJobSuccesful = false;
                    if (topNproductsJobSuccessful) {

                        Job joinsJob = Job.getInstance(configuration, "Join");
                        joinsJob.setJarByClass(Driver.class);

                        MultipleInputs.addInputPath(joinsJob, topNProductsOutputPath,
                                TextInputFormat.class, TopProductsMapper.class);
                  MultipleInputs.addInputPath(joinsJob, summarizationOutputPath,
                        TextInputFormat.class, RatingsMapper.class);
                  FileOutputFormat.setOutputPath(joinsJob, joinOutputPath);

                        joinsJob.setReducerClass(JoinReducer.class);

                        joinsJob.setMapOutputKeyClass(Text.class);
                        joinsJob.setMapOutputValueClass(Text.class);

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

                        if (fs.exists(joinOutputPath)) {
                            fs.delete(joinOutputPath, true);
                        }

                        joinJobSuccesful = joinsJob.waitForCompletion(true);

                    }
```

```
			boolean binningJobSuccesful = false;
			if(joinJobSuccesful) {

				Job binningJob = Job.getInstance(configuration, "Binning");
				binningJob.setJarByClass(Driver.class);

				binningJob.setMapperClass(BinningMapper.class);
				binningJob.setMapOutputKeyClass(Text.class);
				binningJob.setMapOutputValueClass(NullWritable.class);

				//No combiner, partitioner or reducer is used in this pattern!
				binningJob.setNumReduceTasks(1);

				FileInputFormat.setInputPaths(binningJob, inputPath);
				FileOutputFormat.setOutputPath(binningJob,
binningOutputPath);

				if (fs.exists(binningOutputPath)) {
					fs.delete(binningOutputPath, true);
				}

				MultipleOutputs.addNamedOutput(binningJob, "bins",
TextOutputFormat.class,
						Text.class, NullWritable.class);
				MultipleOutputs.setCountersEnabled(binningJob, true);

				binningJobSuccesful = binningJob.waitForCompletion(true);
			}

			boolean invertedIndexJobSuccesful = false;
			if (binningJobSuccesful) {

				Job invertedIndexJob = Job.getInstance(configuration, "Inverted
Index");
				invertedIndexJob.setJarByClass(Driver.class);

				invertedIndexJob.setMapperClass(InvertedIndexMapper.class);
				invertedIndexJob.setReducerClass(InvertedIndexReducer.class);

				invertedIndexJob.setInputFormatClass(TextInputFormat.class);
```

```
	invertedIndexJob.setOutputFormatClass(TextOutputFormat.class);
				invertedIndexJob.setMapOutputKeyClass(Text.class);
				invertedIndexJob.setMapOutputValueClass(Text.class);
				invertedIndexJob.setOutputKeyClass(Text.class);
				invertedIndexJob.setOutputValueClass(Text.class);

				FileInputFormat.addInputPath(invertedIndexJob, inputPath);
				FileOutputFormat.setOutputPath(invertedIndexJob,
invertedIndexOutputPath);

				if (fs.exists(invertedIndexOutputPath)) {
					fs.delete(invertedIndexOutputPath, true);
				}

				invertedIndexJobSuccesful =
invertedIndexJob.waitForCompletion(true);

			}

			boolean mahoutDataCleanJobSuccesful = false;
			if(invertedIndexJobSuccesful) {

				Job dataCleanJob = Job.getInstance(configuration, "Data clean");
				dataCleanJob.setJarByClass(Driver.class);

				dataCleanJob.setMapperClass(DataMapper.class);
				dataCleanJob.setMapOutputKeyClass(NullWritable.class);
				dataCleanJob.setMapOutputValueClass(Text.class);
				dataCleanJob.setNumReduceTasks(1);

				FileInputFormat.setInputPaths(dataCleanJob, inputPath);
				FileOutputFormat.setOutputPath(dataCleanJob,
dataCleanOutputPath);

				if (fs.exists(dataCleanOutputPath)) {
					fs.delete(dataCleanOutputPath, true);
				}

				mahoutDataCleanJobSuccesful =
dataCleanJob.waitForCompletion(true);
```

```
				}

				boolean recommendationJobSuccesful = false;
				if(mahoutDataCleanJobSuccesful) {

					Job recommendationJob = Job.getInstance(configuration,
	"Recommendation");
					String path = dataCleanOutputPath.toString();
					recommendationJob.getConfiguration().set("DataPath", path);

					recommendationJob.setJarByClass(Driver.class);
					FileInputFormat.setInputPaths(recommendationJob,
	dataCleanOutputPath);
					FileOutputFormat.setOutputPath(recommendationJob,
	recommedationOutputPath);

		recommendationJob.setMapperClass(RecommendationMapper.class);

		recommendationJob.setReducerClass(RecommendationReducer.class);
					recommendationJob.setNumReduceTasks(1);

					recommendationJob.setMapOutputKeyClass(Text.class);

		recommendationJob.setMapOutputValueClass(NullWritable.class);

					recommendationJob.setOutputKeyClass(NullWritable.class);
					recommendationJob.setOutputValueClass(Text.class);

					if(fs.exists(recommedationOutputPath)) {
						fs.delete(recommedationOutputPath, true);
					}
					recommendationJobSuccesful =
	recommendationJob.waitForCompletion(true);
					//ProductRecommendation.Recommend(dataCleanOutputPath);
				}
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
```

Aditya Joshi
001837740

Summarization

AverageMapper.java

```
package com.neu.AmazonReviewsAnalysis.Summarization;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class AverageMapper extends Mapper<LongWritable, Text, Text,
CountAverageTuple> {

    private Text text = new Text();
    private CountAverageTuple outCountAverage = new CountAverageTuple();

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

            if(key.get()==0){
                return;
            }

            else{

                String[] line = value.toString().split("\\t");
                String productId = line[3].trim();
                text.set(productId);
                outCountAverage.setCount(1);
                outCountAverage.setAverage(Float.valueOf(line[7].trim()));

                context.write(text, outCountAverage);

            }
    }
}
```

AverageReducer.java

```
package com.neu.AmazonReviewsAnalysis.Summarization;
```

Aditya Joshi
001837740

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

public class AverageReducer extends Reducer<Text, CountAverageTuple, Text,
CountAverageTuple>{

    private CountAverageTuple result = new CountAverageTuple();

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

            float sum = 0;
            float count = 0;
            for (CountAverageTuple val : values) {
                    sum += val.getCount() * val.getAverage();
                  count += val.getCount();
            }
            result.setCount(count);

     float scale = (float) Math.pow(10, 2);
     result.setAverage(Math.round((sum/count) * scale) / scale);

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

CountAverageTupple.java

```
package com.neu.AmazonReviewsAnalysis.Summarization;

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

import org.apache.hadoop.io.Writable;

public class CountAverageTuple implements Writable {

    private float count = 0;
```

```
	private float average = 0;
	public float getCount() {
		return count;
	}
	public void setCount(float count) {
		this.count = count;
	}
	public float getAverage() {
		return average;
	}
	public void setAverage(float average) {
		this.average = average;
	}
	public void write(DataOutput out) throws IOException {
		// TODO Auto-generated method stub
		out.writeFloat(average);
		out.writeFloat(count);

	}
	public void readFields(DataInput in) throws IOException {
		// TODO Auto-generated method stub
		average = in.readFloat();
		count = in.readFloat();
	}

	public String toString() {
	 return String.valueOf(average);
       }

    }
```

Aditya Joshi
001837740

Aggregate Map Reduce

AggregateMapper.java

```
package com.neu.AmazonReviewsAnalysis.Summarization;

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.Mapper;

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

    private static final IntWritable one = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context)
                throws IOException, InterruptedException {

        if(key.get()==0){
            return;

        } else {

            String[] line = value.toString().split("\\t");
            String productId = line[3].trim();

            context.write(new Text(productId), one);

        }
    }

}
```

Aditya Joshi
001837740

AggregateReducer.java

```
package com.neu.AmazonReviewsAnalysis.Summarization;

import java.io.IOException;

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

public class AggregateReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
                    Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws
IOException, InterruptedException {

        int sum = 0;

        for (IntWritable value : values) {
            sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }

}
```

Aditya Joshi
001837740

Top N Products

CountComparator.java

```
package com.neu.AmazonReviewsAnalysis.TopNProducts;

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

public class CountComparator extends WritableComparator {

    protected CountComparator() {

            super(IntWritable.class,true);
    }

    public int compare(WritableComparable w1, WritableComparable w2) {
            IntWritable cw1 = (IntWritable) w1;
            IntWritable cw2 = (IntWritable) w2;

            int result = cw1.get() < cw2.get() ? 1 : cw1.get() == cw2.get() ? 0 : -1;
            return result;
    }
}
```

TopNProductsMapper.java

```
package com.neu.AmazonReviewsAnalysis.TopNProducts;

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

public class TopNProductsMapper extends Mapper<LongWritable, Text, IntWritable, Text>
{
    public void map(LongWritable key, Text value,Context context){

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

            String productId = row[0].trim();
```

```
                int count = Integer.parseInt(row[1].trim());

                try{

                        Text id = new Text(productId);
                        IntWritable prodRating = new IntWritable(count);
                        context.write(prodRating, id);

                }catch(Exception e){

                }
        }
    }

    TopNProductsReducer.java
    package com.neu.AmazonReviewsAnalysis.TopNProducts;

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

    public class TopNProductsReducer extends Reducer<IntWritable, Text, IntWritable, Text>{

        int count = 0;

        // default value = 10
        private int N = 10;

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

                for(Text val: value){
                        if(count<N)
                        {
                                context.write(key,val);
                        }
                        count++;
                }
        }
```

```
		@Override
		protected void setup(Context context) throws IOException, InterruptedException {
			// default = 10
			this.N = context.getConfiguration().getInt("N", 10);
		}
	}
```

Aditya Joshi
001837740

Joins

TopProductsMapper.java

```
package com.neu.AmazonReviewsAnalysis.Join;

import java.io.IOException;

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

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

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

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

            Text productId = new Text();
            Text count = new Text();
            productId.set(line[1].trim());
            count.set("#"+ line[1] + " "+ line[0].trim());

            context.write(productId, count);
    }
}
```

RatingsMapper.java

```
package com.neu.AmazonReviewsAnalysis.Join;

import java.io.IOException;

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 org.apache.hadoop.mapreduce.Mapper.Context;

public class RatingsMapper extends Mapper<LongWritable, Text, Text, Text> {
    protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException{

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

            Text productId = new Text();
            Text rating = new Text();
            productId.set(line[0].trim());
            rating.set("*" + line[1].trim());

            context.write(productId, rating);
    }
}
```

JoinReducer.java

```
package com.neu.AmazonReviewsAnalysis.Join;

import java.io.IOException;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;

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

public class JoinReducer extends Reducer<Text, Text, Text, Text> {

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

     Set<String> listA = new HashSet<String>();
      Set<String> listB = new HashSet<String>();
      for (Text text: values) {
         if (text.toString().startsWith("#"))
            listA.add(text.toString().substring(1));
         else if (text.toString().startsWith("*"))
            listB.add(text.toString().substring(1));
      }
```

```
            if(!listA.isEmpty() && !listB.isEmpty()) {
                  for (String A: listA) {
                     for (String B: listB) {
                        context.write(new Text(A), new Text(B));
                     }
                  }
            }
         }
      }
```

Aditya Joshi
001837740

Binning Pattern

BinningMapper.java

```
package com.neu.AmazonReviewsAnalysis.SplitByRating;

import java.io.IOException;

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 org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;

public class BinningMapper extends Mapper<LongWritable, Text, Text, NullWritable>{

    private MultipleOutputs<Text, NullWritable> output = null;

    @Override
    protected void setup(Context context){
            output = new MultipleOutputs(context);
    }

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

            if(key.get()==0){
                    return;
            }

            String[] token = value.toString().split("\\t");
            String rating = token[7].trim();

            if(rating.equals("1")){
                    output.write("bins", value, NullWritable.get(), "Rating 1");
            }
            if(rating.equals("2")){
                    output.write("bins", value, NullWritable.get(), "Rating 2");
            }
				if(rating.equals("3")){
					output.write("bins", value, NullWritable.get(), "Rating 3");
				}
				if(rating.equals("4")){
					output.write("bins", value, NullWritable.get(), "Rating 4");
				}
				if(rating.equals("5")){
					output.write("bins", value, NullWritable.get(), "Rating 5");
				}
		}

		@Override
		protected void cleanup(Context context) throws IOException, InterruptedException{
			output.close();
		}
	}
```

Aditya Joshi
001837740

Inverted Index Pattern

InvertedIndexMapper.java

```
package com.neu.AmazonReviewsAnalysis.InvertedIndex;

import java.io.IOException;

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

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

    private Text productId = new Text();
    private Text userId = new Text();

    public void map(LongWritable key, Text values, Context context) throws
InterruptedException{

            if(key.get()==0){
                    return;
            }

            try{
                    String[] tokens = values.toString().split("\\t");
                    userId.set(tokens[1]);
                    productId.set(tokens[3]);
                    context.write(productId, userId);
            }
            catch(IOException  ex){
                    System.out.println("Error in Mapper" + ex.getMessage());
            }
    }
}
```

Aditya Joshi
001837740

InvertedIndexReducer.java

```
package com.neu.AmazonReviewsAnalysis.InvertedIndex;

import java.io.IOException;

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

public class InvertedIndexReducer extends Reducer<Text,Text,Text,Text>{

    private Text result = new Text();

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

        StringBuilder sb = new StringBuilder();

        boolean first = true;

        for(Text id: values){
            if(first){
                first = false;
            }
            else{
                sb.append(" ");
            }
            sb.append(id.toString());
        }

        result.set(sb.toString());
        context.write(key, result);
    }
}
```

Aditya Joshi
001837740

Mahout Recommendation

Data Cleaning

DataMapper.java

```
package com.neu.AmazonReviewsAnalysis.MahoutRecommendations;

import java.io.IOException;

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


public class DataMapper extends Mapper<LongWritable, Text, NullWritable, Text> {

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

            if(key.get()==0){
                return;
            }

            else{

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

                Text res = new Text();
                res.set(line[1] + "," + line[4] + "," + line[7]);

                context.write(NullWritable.get(), res);

            }
    }
}
```

Aditya Joshi
001837740

User Recommendation with Mahout

RecommendationMapper.java

```
package com.neu.AmazonReviewsAnalysis.MahoutRecommendations;

import java.io.IOException;

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 org.apache.hadoop.mapreduce.Mapper.Context;

public class RecommendationMapper extends Mapper<LongWritable, Text, Text,
NullWritable> {

    private Text text = new Text();

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


            String[] line = value.toString().split(",");
            String userId = line[0].trim();
            text.set(userId);

            context.write(text, NullWritable.get());
    }
}
```

RecommendationReducer.java

```
package com.neu.AmazonReviewsAnalysis.MahoutRecommendations;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;

import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.mahout.cf.taste.common.TasteException;
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;

public class RecommendationReducer extends Reducer<Text, NullWritable, NullWritable,
Text>{

    private String path = new String();
    private File userPreferencesFile;
    private DataModel dataModel;
    private UserSimilarity userSimilarity;
    private UserNeighborhood userNeighborhood;
    private Recommender genericRecommender;

    @Override
    protected void reduce(Text key, Iterable<NullWritable> value, Context context)
                throws IOException,InterruptedException, FileNotFoundException{

            try {

                Long userId = Long.valueOf(key.toString());
                List<RecommendedItem> recs =
genericRecommender.recommend(userId,2);

        if (!recs.isEmpty()) {

                    Text res = new Text();
                    for (RecommendedItem recommendedItem : recs) {

                        res.set(key.toString() + "Recommened Item Id: " +
recommendedItem.getItemID() +
                                                          " Strength of preference: " +
recommendedItem.getValue());
                                }
                                context.write(NullWritable.get(), res);
                        }

                }       catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

        @Override
        protected void setup(Context context)
                        throws IOException, InterruptedException, FileNotFoundException {

                try {
                        this.path = context.getConfiguration().get("DataPath");
                        String fname = "/part-r-00000";
                        this.path = this.path + fname;

                        this.userPreferencesFile = new File(path);

                        this.dataModel = new FileDataModel(this.userPreferencesFile);

                        this.userSimilarity = new PearsonCorrelationSimilarity(this.dataModel);

                        this.userNeighborhood = new NearestNUserNeighborhood(5,
this.userSimilarity, this.dataModel);

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

                } catch (FileNotFoundException ex) {

                        System.out.println("Exception: " + ex.getMessage());
                }       catch (TasteException e) {
```

```
				// TODO Auto-generated catch block
				e.printStackTrace();
			}	catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

		}

	}
```

Analysis using Pig

DailyReviewsCount.pig

```
data1 = load '/home/aditya-ubuntu/Aditya/AmazonDataset/amazon_reviews_us_Home_Entertainment_v1_00.tsv' using PigStorage('\t') AS (marketplace, customer_id, review_id, product_id, product_parent, product_title, product_category, star_rating, helpful_votes, total_votes, vine, verified_purchase, review_headline, review_body, review_date);

data = STREAM data1 THROUGH `tail -n +2` AS (marketplace, customer_id, review_id, product_id, product_parent, product_title, product_category, star_rating, helpful_votes, total_votes, vine, verified_purchase, review_headline, review_body, review_date);

daily = GROUP data by review_date;

daily_reviews = FOREACH daily GENERATE group as review_date, COUNT(data.review_id) as count;

order_by_data = ORDER daily_reviews BY count DESC;

store order_by_data INTO '/home/aditya-ubuntu/Aditya/Output/pig1';
```

Aditya Joshi
001837740

ProductsPerRating.pig

```
data1 = load '/home/aditya-
ubuntu/Aditya/AmazonDataset/amazon_reviews_us_Home_Entertainment_v1_00.tsv'
using PigStorage('\t') AS (marketplace, customer_id, review_id, product_id, product_parent,
product_title, product_category, star_rating, helpful_votes, total_votes, vine,
verified_purchase, review_headline, review_body, review_date);

data = STREAM data1 THROUGH `tail -n +2` AS (marketplace, customer_id, review_id,
product_id, product_parent, product_title, product_category, star_rating, helpful_votes,
total_votes, vine, verified_purchase, review_headline, review_body, review_date);

prod = GROUP data by star_rating;

prod_count = FOREACH prod GENERATE group as star_rating, COUNT(data.product_id) as
count;

store prod_count INTO '/home/aditya-ubuntu/Aditya/Output/pig2';
```