INFO 7250: ENGINEERING BIG-DATA SYSTEMS FALL 2019

ANALYSIS OF AMAZON CUSTOMER REVIEWS

NAME: AMI GANDHI
NUID: 001446230

Abstract:

Analysis of amazon is very crucial part when it comes to find an efficient way of getting insights on customer reviews about different products. Hence, this project is mainly aimed to analyze big data and produce an informative result about the customer reviews for the product Camera present on Amazon using Hadoop architecture, Mahout and Pig

Dataset:

Dataset Link:
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Camera_v1_00.tsv.gz

Accessing the data:
Data is present in the Amazon S3 bucket. It can be accessed as mentioned in below link –
https://s3.amazonaws.com/amazon-reviews-pds/readme.html

About the data:
DATA FORMAT -
Tab ('\t') separated text file, without quote or escape characters. This dataset is 1GB in size. First line in the file is header; 1 line corresponds to 1 record.

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 Analysis:

1. Find the total number of products present in the dataset
2. Find the average product rating reviews for each product
3. Find the topN reviewed products sorted by count
4. Find total product rating for all those products which are present in the topN reviewed products
5. Find all the users who has reviewed each product
6. Find all the records partitioned by the date in which the product was reviewed
7. Find all the product information for each unique star rating of the product by dividing it into different categories/bins
8. Recommend the products to the user based on the star rating
9. Find the count of the reviews grouped by date for each product
10. Find the count of products for each product star rating

Techniques/Technologies Used:

1. Hadoop MapReduce
2. Summarization Pattern – Numerical Summarization, Inverted Index
3. Joins – Reduce Side Inner Join
4. Partitioning
5. Secondary Sorting
6. MapReduce Chaining
7. Filtering Pattern – TopN filtering pattern
8. Binning Pattern
9. Mahout Recommendation
10. Apache Pig

Additional Notes:

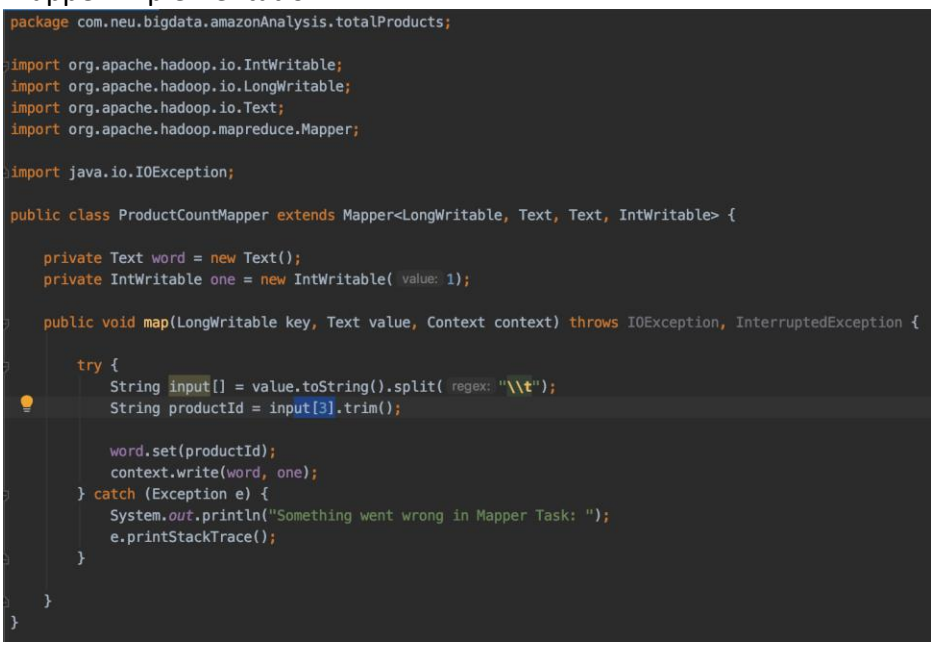
1. Dataset was taken from Amazon S3 bucket on cloud using AWS-CLI
2. Implemented logging using log4j in almost all analysis
3. Implemented MultipleInputs in Reduce Side Join Pattern
4. Implemented MultipleOutputs in Binning pattern
5. Used a Custom Partitioner
6. Used to Combiner in TopN filtering pattern
7. Implemented a WritableComparator class while performing secondary sorting
8. Used MapReduce Chaining to chain various job in analysis
9. Ran pig script in local mode and used the grunt shell

Data Analysis Approach:

1. **Find the total number of products present in the dataset:**

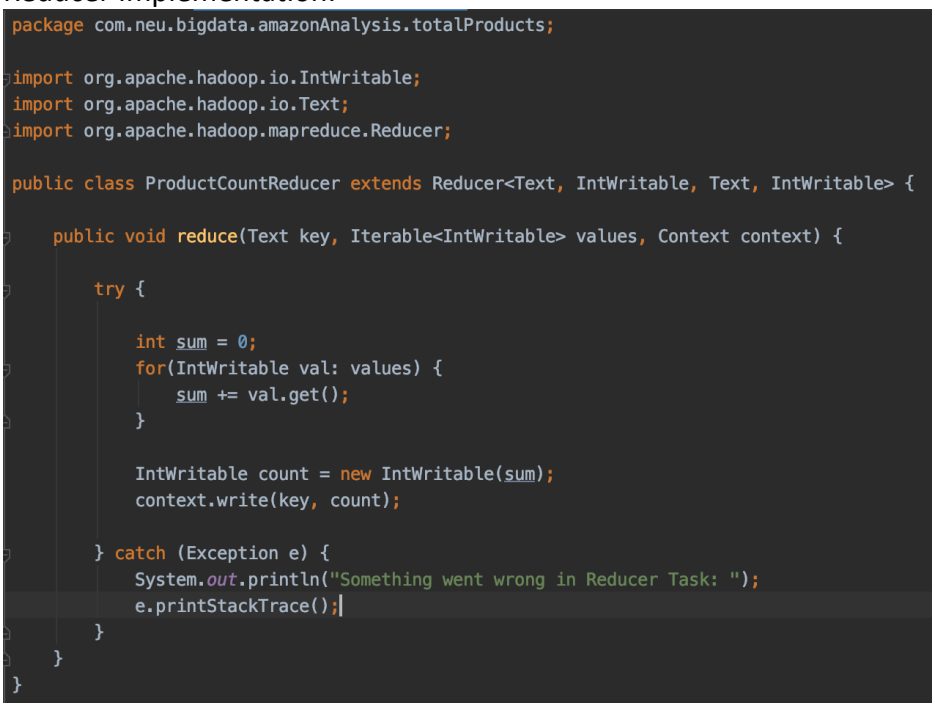
Hadoop MapReduce is used to calculate the total number of products present in dataset.
Final Output: ProductID, Product Count

Mapper Implementation:



```
package com.neu.bigdata.amazonAnalysis.totalProducts;

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

import java.io.IOException;

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

    private Text word = new Text();
    private IntWritable one = new IntWritable( value: 1);

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

        try {
            String input[] = value.toString().split( regex: "\\t");
            String productId = input[3].trim();

            word.set(productId);
            context.write(word, one);
        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }

    }
}
```

Reducer Implementation:



```
package com.neu.bigdata.amazonAnalysis.totalProducts;

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

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

    public void reduce(Text key, Iterable<IntWritable> values, Context context) {

        try {

            int sum = 0;
            for(IntWritable val: values) {
                sum += val.get();
            }

            IntWritable count = new IntWritable(sum);
            context.write(key, count);

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

Main Implementation:

```java
public class ProductCountMain {

    private final static Logger logger = Logger.getLogger(com.neu.bigdata.amazonAnalysis.totalProducts.ProductCountMain.class);
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        Configuration conf = new Configuration();
        try {
            long startTime = System.currentTimeMillis();

            Job job = new Job(conf,  jobName: "ProductCount");
            job.setJarByClass(ProductCountMain.class);

            job.setMapperClass(ProductCountMapper.class);
            job.setReducerClass(ProductCountReducer.class);

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

            FileInputFormat.addInputPath(job, new Path(args[1]));
            FileOutputFormat.setOutputPath(job, new Path(args[2]));

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

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

            job.setNumReduceTasks(1);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);
            System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/AmazonAnalysis_jar/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.totalProducts.ProductCountMain /project/dataset /project/output/ProductCount/result

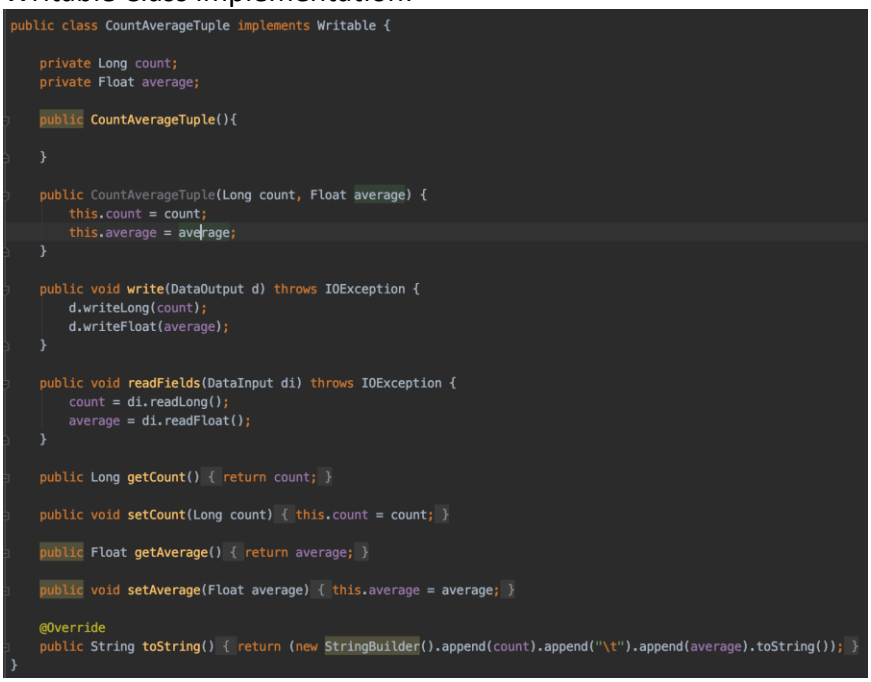
Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/ProductCount/result/part-r-00000 | head
Password:
2019-12-14 18:42:33,186 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform..
0011300000      1
094339676X      1
0974096512      1
0981602940      1
0983947600      5
0984445129      2
0984445145      1
0984445161      5
0984920242      2
1123000034      3
```

2. **Find the average product rating reviews for each product:**

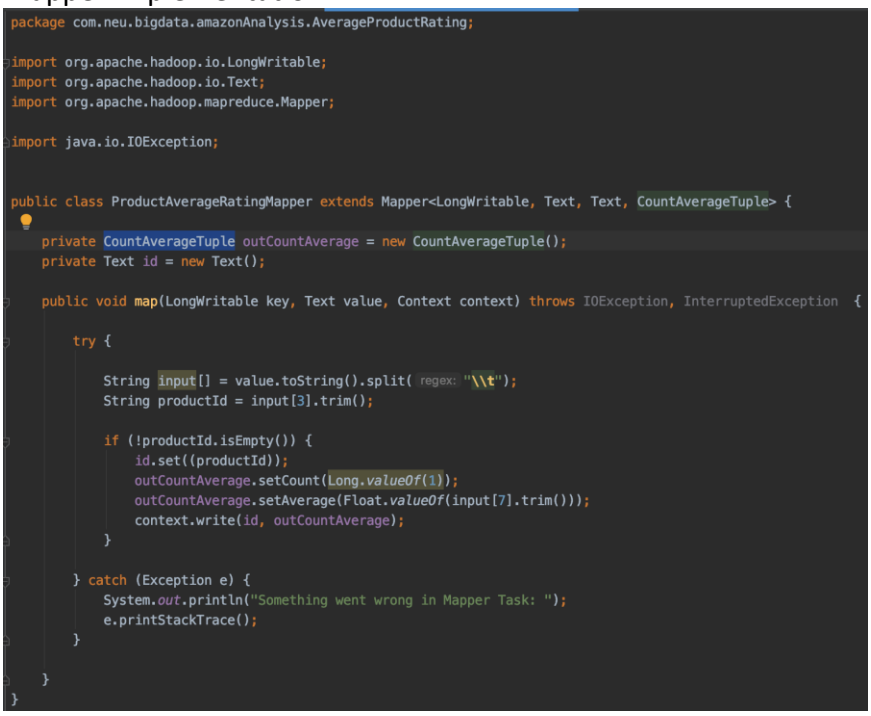
Hadoop MapReduce is used to calculate the total number of products present in dataset. Here, reducer is also used as combiner. Writable class is implemented to calculate average.
Output: ProductID, Product Count, Product Average Rating

Writable Class implementation:



```
public class CountAverageTuple implements Writable {

    private Long count;
    private Float average;

    public CountAverageTuple(){

    }

    public CountAverageTuple(Long count, Float average) {
        this.count = count;
        this.average = average;
    }

    public void write(DataOutput d) throws IOException {
        d.writeLong(count);
        d.writeFloat(average);
    }

    public void readFields(DataInput di) throws IOException {
        count = di.readLong();
        average = di.readFloat();
    }

    public Long getCount() { return count; }

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

    public Float getAverage() { return average; }

    public void setAverage(Float average) { this.average = average; }

    @Override
    public String toString() { return (new StringBuilder().append(count).append("\t").append(average).toString()); }
}
```

Mapper Implementation:



```
package com.neu.bigdata.amazonAnalysis.AverageProductRating;

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

import java.io.IOException;


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

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

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

        try {

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

            if (!productId.isEmpty()) {
                id.set((productId));
                outCountAverage.setCount(Long.valueOf(1));
                outCountAverage.setAverage(Float.valueOf(input[7].trim()));
                context.write(id, outCountAverage);
            }

        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }

    }
}
```

Reducer Implementation:

```java
package com.neu.bigdata.amazonAnalysis.AverageProductRating;

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

import java.io.IOException;

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

    private CountAverageTuple result = new CountAverageTuple();

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

        try {
            long count = 0;
            float sum = 0;

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

            result.setCount(count);
            result.setAverage(sum/count);
            context.write(key, result);

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }

}
```

Main Implementation:

```java
public class ProductAverageRatingMain {

    final static Logger logger = Logger.getLogger(ProductAverageRatingMain.class);

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException
    {
        try {
            long startTime = System.currentTimeMillis();
            Job job = Job.getInstance();
            job.setJarByClass(ProductAverageRatingMain.class);

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

            job.setMapperClass(ProductAverageRatingMapper.class);
            job.setReducerClass(ProductAverageRatingReducer.class);
            job.setCombinerClass(ProductAverageRatingReducer.class);

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

            job.setNumReduceTasks(1);

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

            job.waitForCompletion( verbose: true);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }
    }
}
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/AmazonAnalysis_jar/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.AverageProducts.ProductAverageVotesMain /project/dataset /project/output/ProductAverageVote

Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/ProductAverageVote/part-r-00000 | head
2019-12-14 19:58:27,528 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
0011300000      1       3.0
094339676X      1       5.0
0974096512      1       4.0
0981602940      1       5.0
0983947600      5       4.2
0984445129      2       5.0
0984445145      1       5.0
0984445161      5       5.0
0984920242      2       5.0
1123000034      3       3.6666667
```

3. Find the topN reviewed products sorted by count:

Hadoop MapReduce is used to find topN reviewed products. Output file of 1st analysis performed on product count is sued as input to this analysis. Secondary sorting is performed by extending WritableComparator class to get top 10 products in descending order. TopN filtering pattern is used to find top 10 products.
Final Output: Product Count, ProductID

Count Comparator Implementation:

```java
package com.neu.bigdata.amazonAnalysis.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, createInstances: 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;
    }
}
```

Mapper Implementation:

```java
package com.neu.bigdata.amazonAnalysis.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( regex: "\\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){
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }
    }
}
```

Reducer Implementation:

```java
package com.neu.bigdata.amazonAnalysis.topNProducts;

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

import java.io.IOException;

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

    int count = 0;
    private int N = 10;

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

    @Override
    public void reduce(IntWritable key, Iterable<Text> value, Context context)
            throws IOException, InterruptedException{
        try {
            for(Text val: value){
                if(count<N)
                {
                    context.write(key,val);
                }
                count++;
            }

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

Main Implementation:

```
final static Logger logger = Logger.getLogger(TopNProductsMain.class);
public static void main(String[] args) throws IOException {

    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(conf);

    try {
        long startTime = System.currentTimeMillis();
        Job topNProductsJob = Job.getInstance(conf,  jobName: "Top N Rated Products");
        topNProductsJob.setJarByClass(com.neu.bigdata.amazonAnalysis.topNProducts.TopNProductsMain.class);

        int N = 10;
        topNProductsJob.getConfiguration().setInt( name: "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, new Path(args[0]));  // Output file path of totalProducts - MR chaining
        FileOutputFormat.setOutputPath(topNProductsJob, new Path(args[1]));
        if (fs.exists(new Path(args[1]))) {
            fs.delete(new Path(args[1]),  b: true);
        }

        topNProductsJob.waitForCompletion( verbose: true);
        long endTime = System.currentTimeMillis();
        logger.info("Time taken in milliseconds : " + (endTime - startTime));
        logger.info("Time taken in seconds : " + (endTime - startTime)/1000);
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/TopNProducts/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.topNProducts.TopNProductsMain /project/output/ProductCount/result/ /project/output/TopNProducts

Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/TopNProducts/part-r-00000 | head
Password:
2019-12-14 20:29:11,408 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
4654    B006ZP8UOW
4399    B00007E7JU
3619    B0039BPG1A
3565    B002VPE1WK
3177    B0050R67U0
2358    B00AAIPT76
2317    B00009R6TA
2269    B00007EDZG
2244    B00F9FCW7K
2234    B004TJ6JH6
```

Time taken in milliseconds : 5873
Time taken in seconds : 5

4. Find total product rating for all products that are present in topN reviewed products:

Hadoop MapReduce is used to calculate the total number of products present in dataset. Output of analysis 2 and analysis 3 of topN products is used as 2 inputs (MultipleInputs) for this analysis and a Reduce Side Inner Join technique is used to get intersected products.

Output: ProductID, Product Count, Product Average

Top products Mapper Implementation:

```java
package com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin;

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

import java.io.IOException;

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

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

        Text productId = new Text();
        Text count = new Text();

        try {
            String[] input = value.toString().split( regex: "\\t");
            productId.set(input[1].trim());
            count.set("#"+ input[1] + " "+ input[0].trim());

            context.write(productId, count);

        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper 1 Task: ");
            e.printStackTrace();
        }
    }
}
```

Ratings Mapper Implementation:

```java
package com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin;

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

import java.io.IOException;

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

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

        Text productId = new Text();
        Text rating = new Text();

        try {
            String[] input = value.toString().split( regex: "\\t");
            productId.set(input[0].trim());
            rating.set("*" + input[2].trim());

            context.write(productId, rating);

        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper 2 Task: ");
            e.printStackTrace();
        }
    }
}
```

Reducer Implementation:

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

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

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

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

        try {
            Set<String> listA = new HashSet<>();
            Set<String> listB = new HashSet<>();

            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));
                    }
                }
            }
        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

Main Implementation:

```java
public class JoinMain {

    final static Logger logger = Logger.getLogger(JoinMain.class);
    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        try {
            long startTime = System.currentTimeMillis();
            Job joinsJob = Job.getInstance(conf,  jobName: "Join");
            joinsJob.setJarByClass(com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin.JoinMain.class);

            Path topNProductsOutputPath = new Path(args[0]);
            Path summarizationOutputPath = new Path(args[1]);
            MultipleInputs.addInputPath(joinsJob, topNProductsOutputPath, TextInputFormat.class, TopProductsMapper.class);
            MultipleInputs.addInputPath(joinsJob, summarizationOutputPath, TextInputFormat.class, RatingsMapper.class);
            FileOutputFormat.setOutputPath(joinsJob, new Path(args[2]));

            joinsJob.setReducerClass(JoinReducer.class);

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

            if (fs.exists(new Path(args[2]))) {
                fs.delete(new Path(args[2]),  b: true);
            }

            joinsJob.waitForCompletion( verbose: true);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/ReduceSideInnerJoin/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin.JoinMain /project/output/TopNProducts/ /project/output/ProductAverageVote /project/output/ReduceSideInnerJoin

Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/ReduceSideInnerJoin/part-r-00000 | head
Password:
2019-12-14 20:43:49,300 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
B00004THD0 651  4.290323
B00004WCGF 989  4.48635
B00004WCID 785  4.5312104
B00004ZCJG 978  4.214724
B00004ZCJI 889  4.302587
B00005K47X 678  4.6091447
B00005LEN4 1417 4.642202
B00006I5J7 668  4.718563
B00006JN3G 1039 4.381136
B00007E7JU 4399 4.5810413
```

Time taken in milliseconds : 5883
Time taken in seconds : 5

5. **Find all the users who has reviewed each product.**

Hadoop MapReduce is used to get all the users who has reviewed each product. Here, Inverted Index summarization pattern is used to get user information.
Output: ProductID, User ID (Space separated)

Mapper Implementation:

```java
package com.neu.bigdata.amazonAnalysis.invertedIndexPattern;

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

import java.io.IOException;

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

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

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

        if(key.get()==0){
            return;
        }
        try{
            String[] tokens = value.toString().split( regex: "\\t");
            userId.set(tokens[1]);
            productId.set(tokens[3]);
            context.write(productId, userId);

        } catch(Exception  e){
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }
    }
}
```

Reducer Implementation:

```java
package com.neu.bigdata.amazonAnalysis.invertedIndexPattern;

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

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 {

        try {
            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);

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
```

Main Implementation:

```java
    final static Logger logger = Logger.getLogger(InvertedIndexMain.class);
    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        try {
            long startTime = System.currentTimeMillis();
            Job invertedIndexJob = Job.getInstance(conf,  jobName: "Inverted Index");
            invertedIndexJob.setJarByClass(com.neu.bigdata.amazonAnalysis.invertedIndexPattern.InvertedIndexMain.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, new Path(args[0]));
            FileOutputFormat.setOutputPath(invertedIndexJob, new Path(args[1]));
            if (fs.exists(new Path(args[1]))) {
                fs.delete(new Path(args[1]),  b: true);
            }

            invertedIndexJob.waitForCompletion( verbose: true);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/InvertedIndex/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.invertedIndexPattern.InvertedIndexMain /project/dataset /project/output/InvertedIndex

Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/InvertedIndex/part-r-00000 | head
Password:
2019-12-14 20:51:46,609 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
0011300000      4218977
094339676X      14079644
0974096512      14514479
0981602940      26601318
0983947600      26556637 34591948 45419585 16476153 20125622
0984445129      37980273 53057681
0984445145      50935372
0984445161      13798903 36373601 26460354 40524550 20308830
0984920242      53086236 5357804
1123000034      23388040 6349032 30886376
```

Time taken in milliseconds : 25617
Time taken in seconds : 25

6. **Find all the records partitioned by date for each product.**

Hadoop MapReduce is used to find all records partitioned by date for each product. A custom partitioner class is extended to partition and implement this analysis.
Final Output: Entire data partitioned into separate partitions

Custom Partitioner class:

```
public static class YearPartitionPartitioner extends Partitioner<Text, Text> {
    @Override
    public int getPartition(Text key, Text value, int numPartitions){
        int n=1;
        if(numPartitions==0){
            return 0;
        }
        else if(key.equals(("99"))){
            return n % numPartitions;
        }
        else if(key.equals(new Text( string: "00"))){
            return 2 % numPartitions;
        }
        else if(key.equals(new Text( string: "01"))){
            return 3 % numPartitions ;
        }
        else if(key.equals(new Text( string: "02"))){
            return 4 % numPartitions;
        }
        else if(key.equals(new Text( string: "03"))){
            return 5 % numPartitions;
        }
        else if(key.equals(new Text( string: "04"))){
            return 6 % numPartitions;
        }
        else if(key.equals(new Text( string: "05"))){
            return 7 % numPartitions;
        }
        else if(key.equals(new Text( string: "06"))){
            return 8 % numPartitions;
        }
        else if(key.equals(new Text( string: "07"))){
            return 9 % numPartitions;
        }
        else if(key.equals(new Text( string: "08"))){
```

Mapper Implementation:

```
public static class YearPartitionMapper extends Mapper<LongWritable, Text, Text, Text> {

    private Text inputRec = new Text();
    private Text year = new Text();

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

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

        String[] line = value.toString().split( regex: "\\t");
        String[] yearPart = line[14].split( regex: "=");
        String yearVal = yearPart[2].trim();

        year.set(yearVal);
        inputRec.set(value);

        context.write(year, inputRec);
    }
}
```

Reducer Implementation:

```
public static class YearPartitionReducer extends Reducer<Text, Text, Text, NullWritable> {

    protected void reduce(Text key, Iterable<Text> values, Reducer.Context context) throws IOException, InterruptedException{
        for(Text t: values){

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

Main Implementation:

```
public class YearPartitioner {

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

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf, jobName: "Partitioning");

        job.setJarByClass(com.neu.bigdata.amazonAnalysis.YearPartitioner.class);

        job.setMapperClass(YearPartitionMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        //Custom Partitioner:
        job.setPartitionerClass(YearPartitionPartitioner.class);

        job.setReducerClass(YearPartitionReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);
        job.setNumReduceTasks(14);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        if (fs.exists(new Path(args[1]))) {
            fs.delete(new Path(args[1]),  b: true);
        }

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

Command to run mapreduce:

sudo hadoop jar
/Users/agandhi/Desktop/bigdata/project/out/artifacts/YearPartitioner/AmazonAnalysis.jar
com.neu.bigdata.amazonAnalysis.YearPartitioner /project/dataset
/project/output/YearPartitioner

Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -ls /project/output/YearPartitioner/
2019-12-14 21:02:46,485 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 15 items
-rw-r--r--   1 root wheel          0 2019-12-13 08:51 /project/output/YearPartitioner/_SUCCESS
-rw-r--r--   1 root wheel  705235035 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00000
-rw-r--r--   1 root wheel          0 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00001
-rw-r--r--   1 root wheel          0 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00002
-rw-r--r--   1 root wheel   33875890 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00003
-rw-r--r--   1 root wheel   36334003 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00004
-rw-r--r--   1 root wheel   37128247 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00005
-rw-r--r--   1 root wheel   36091013 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00006
-rw-r--r--   1 root wheel   35640976 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00007
-rw-r--r--   1 root wheel   36641863 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00008
-rw-r--r--   1 root wheel   37398983 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00009
-rw-r--r--   1 root wheel   36250216 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00010
-rw-r--r--   1 root wheel   37889057 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00011
-rw-r--r--   1 root wheel   36829760 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00012
-rw-r--r--   1 root wheel   36260675 2019-12-13 08:51 /project/output/YearPartitioner/part-r-00013
```

Head of output file of 1st partition:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/YearPartitioner/part-r-00000 | head
2019-12-14 21:03:02,821 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
12      US      36584609        R1UUGOVVI59BSS  B00006371B      461416707       Celestron Nexstar 11 GPS Telescope (Telescope Only)     Camera  5       26      2
9       N       N       4th Celestron Telescope that I have owned       For the past 15 years I have used more telescopes than I can possibly remember.  During
 the same period of time,  I have only &amp;quot;owned&amp;quot; one brand; CELESTRON.  <BR>  <BR>The optics are sometimes a toss up with other manufacturers but one t
hing always brings me back to purchase and that &amp;quot;one thing&amp;quot; is the steadiness &amp;amp; quality in CELESTRON products.  They offer the best mounts, drive
s &amp;amp; tripods in business.  <br />Their Nexstar G11 is a telescope designed with excellent optics and a &amp;quot;Go To&amp;quot; system that would satisfy even the
most demanding astronomy enthusiast.  <br />This system allows the operator the most &amp;quot;visual time&amp;quot; of any significant object due to the accuracy of
its GPS system.  <br />When you ask for a specific object the telescope responds with stunning accuracy.  CCD imaging is a pleasure with this instrument.  <br
/>I want to thank all of those who had a part in the design and production of this &amp;quot;state of the art&amp;quot; instrument.  The performance of this telescope
will not be rivaled for some time.  But when this finally occurs, CELESTRON will have moved forward several steps.  I can not imagine what lies ahead for what
I consider the best telescope manufacturer on this planet. <BR>mja      2002-09-12
12      US      53027936        R1LUB36RW3L789  B0000691LX      190587822       SiPix StyleCam Blink Digital Camera     Camera  3       3       4       N       P
retty cool, but Macintosh users beware  Can't beat the price and the package is mighty tiny.  Warnings about the image quality are correct, but for [$$] I thin
k the image quality is just fine.  What zapped me, however, is the lack of Macintosh support.  The device is USB and TWAIN, so it can't be very hard to support
 the Mac.  It works on my PC but not on my Mac.  There goes my plan to give one to my kid to use on his Mac.    2002-09-12
12      US      51208206        RL2UXE9IFW9K2   B00004SGAZ      284419744       Kodak KB18 35mm Camera  Camera  4       1       1       N       N       Great c
amera   This is a great cheap camera for everyday use.  I have had mine for several years and have not had any problems yet.  The pictures come out great.  I g
ot this camera to replace another Kodak camera I had that broke after about 10 years.  I swore I would never buy another kind of camera...      2002-09-12
12      US      26199104        R3JZPXYR4U47BZ  B0000BZG0P      130425040       Casio Exilim EX-Z4U 4 MP Digital Camera w/ 3x Optical Zoom and Dock     Camera1
9       51      N       Y       Casio!What a good camera        Great l;ittle camera..Sent back optio 4 no way the quality equals the casio!!But like the other
 buyer said Read the book!Oh yes you must also use the CD because it has the manuel on it..I did'nt care for the Casio software!!But &amp;quot;you get a Kodak also
 and I found it to be 100% more easy to use...Trust me on this!I bought a card reader (SD) you just put the memory card in and you save the wear on camera!The
optio did'nt come near the quality of the Casio but,pics were very good...      2003-11-12
12      US      24305047        R3RGB4YM6FSQOM  B00006IR39      121513177       Fujifilm FinePix 2650 2MP Digital Camera w/ 3x Optical Zoom     Camera  5       T
he Greatest Camera for It's Price!!!!   I bought this as my first digital camera last Christmas. It was so easy to use and set up. The picture quality is great
.  It was recently stolen and I'm going to buy another one just like it.  I would recommend this camera to anyone who doesn't like to read directions.  2003-11
-12
```

7. **Find all the product information for each unique star rating of the product by dividing it into different categories/bins**

Hadoop MapReduce is used to all the product information for each unique star rating of the product by dividing it into different categories/bins. Binning pattern is used to split data into 5

bins for each rating category. This is map-side only. No reducer or combiner is used in this pattern. MultipleOutputs is used to output 5 bins for each rating category.
Final Output: Output split into 5 output files with data respective to each rating category.

Mapper Implementation:

```
public class BinningProductReviewRatingsMapper 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{

        try {
            if(key.get()==0) { return; }
            String[] token = value.toString().split( regex: "\\t");
            String rating = token[7].trim();
            if(rating.equals("1")){
                output.write( namedOutput: "bins", value, NullWritable.get(),  baseOutputPath: "Rating1"); }
            if(rating.equals("2")){
                output.write( namedOutput: "bins", value, NullWritable.get(),  baseOutputPath: "Rating2"); }
            if(rating.equals("3")){
                output.write( namedOutput: "bins", value, NullWritable.get(),  baseOutputPath: "Rating3"); }
            if(rating.equals("4")){
                output.write( namedOutput: "bins", value, NullWritable.get(),  baseOutputPath: "Rating4"); }
            if(rating.equals("5")){
                output.write( namedOutput: "bins", value, NullWritable.get(),  baseOutputPath: "Rating5"); }
        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace(); }
    }

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

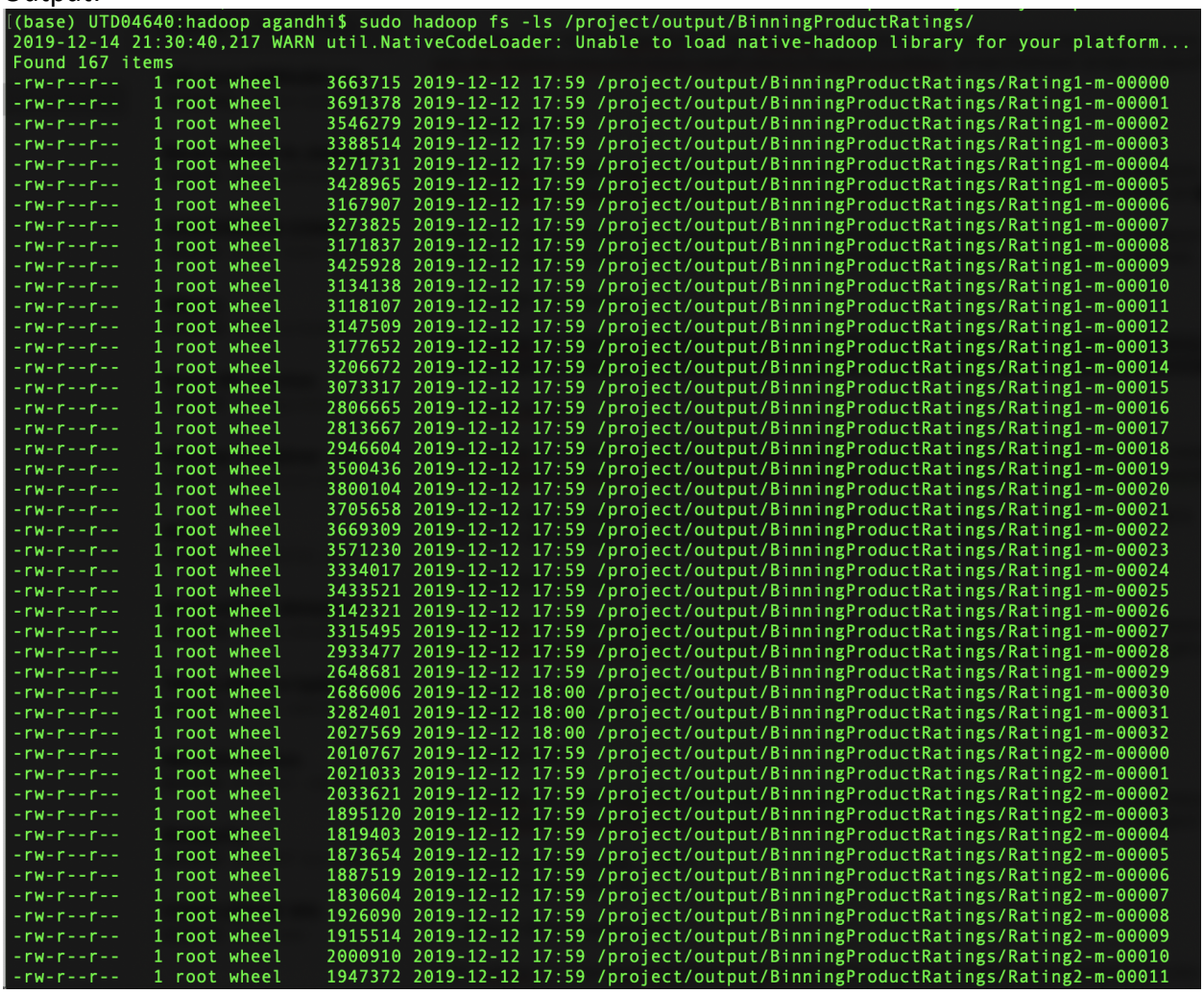
Main Implementation:

```
public class BinningProductReviewRatingsMain {

    final static Logger logger = Logger.getLogger(com.neu.bigdata.amazonAnalysis.BinReviewRatings.BinningProductReviewRatingsMain.class);
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        try {
            long startTime = System.currentTimeMillis();

            Job binningJob = Job.getInstance(conf,  jobName: "Binning Pattern");
            binningJob.setJarByClass(com.neu.bigdata.amazonAnalysis.BinReviewRatings.BinningProductReviewRatingsMain.class);

            binningJob.setMapperClass(BinningProductReviewRatingsMapper.class);
            binningJob.setMapOutputKeyClass(Text.class);
            binningJob.setMapOutputValueClass(NullWritable.class);
            binningJob.setNumReduceTasks(1);

            FileInputFormat.setInputPaths(binningJob, new Path(args[0]));
            FileOutputFormat.setOutputPath(binningJob, new Path(args[1]));
            if (fs.exists(new Path(args[1]))) {
                fs.delete(new Path(args[1]),  b: true);
            }

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

            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);
            System.exit(binningJob.waitForCompletion( verbose: true) ? 0 : 1);

        } catch (Exception e ) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/BinningProductReviewRatings/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.BinReviewRatings.BinningProductReviewRatingsMain /project/dataset /project/output/BinningProductRatings

Output:



```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -ls /project/output/BinningProductRatings/
2019-12-14 21:30:40,217 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 167 items
-rw-r--r--   1 root wheel    3663715 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00000
-rw-r--r--   1 root wheel    3691378 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00001
-rw-r--r--   1 root wheel    3546279 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00002
-rw-r--r--   1 root wheel    3388514 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00003
-rw-r--r--   1 root wheel    3271731 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00004
-rw-r--r--   1 root wheel    3428965 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00005
-rw-r--r--   1 root wheel    3167907 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00006
-rw-r--r--   1 root wheel    3273825 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00007
-rw-r--r--   1 root wheel    3171837 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00008
-rw-r--r--   1 root wheel    3425928 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00009
-rw-r--r--   1 root wheel    3134138 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00010
-rw-r--r--   1 root wheel    3118107 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00011
-rw-r--r--   1 root wheel    3147509 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00012
-rw-r--r--   1 root wheel    3177652 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00013
-rw-r--r--   1 root wheel    3206672 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00014
-rw-r--r--   1 root wheel    3073317 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00015
-rw-r--r--   1 root wheel    2806665 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00016
-rw-r--r--   1 root wheel    2813667 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00017
-rw-r--r--   1 root wheel    2946604 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00018
-rw-r--r--   1 root wheel    3500436 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00019
-rw-r--r--   1 root wheel    3800104 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00020
-rw-r--r--   1 root wheel    3705658 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00021
-rw-r--r--   1 root wheel    3669309 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00022
-rw-r--r--   1 root wheel    3571230 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00023
-rw-r--r--   1 root wheel    3334017 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00024
-rw-r--r--   1 root wheel    3433521 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00025
-rw-r--r--   1 root wheel    3142321 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00026
-rw-r--r--   1 root wheel    3315495 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00027
-rw-r--r--   1 root wheel    2933477 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00028
-rw-r--r--   1 root wheel    2648681 2019-12-12 17:59 /project/output/BinningProductRatings/Rating1-m-00029
-rw-r--r--   1 root wheel    2686006 2019-12-12 18:00 /project/output/BinningProductRatings/Rating1-m-00030
-rw-r--r--   1 root wheel    3282401 2019-12-12 18:00 /project/output/BinningProductRatings/Rating1-m-00031
-rw-r--r--   1 root wheel    2027569 2019-12-12 18:00 /project/output/BinningProductRatings/Rating1-m-00032
-rw-r--r--   1 root wheel    2010767 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00000
-rw-r--r--   1 root wheel    2021033 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00001
-rw-r--r--   1 root wheel    2033621 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00002
-rw-r--r--   1 root wheel    1895120 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00003
-rw-r--r--   1 root wheel    1819403 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00004
-rw-r--r--   1 root wheel    1873654 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00005
-rw-r--r--   1 root wheel    1887519 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00006
-rw-r--r--   1 root wheel    1830604 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00007
-rw-r--r--   1 root wheel    1926090 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00008
-rw-r--r--   1 root wheel    1915514 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00009
-rw-r--r--   1 root wheel    2000910 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00010
-rw-r--r--   1 root wheel    1947372 2019-12-12 17:59 /project/output/BinningProductRatings/Rating2-m-00011
```

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/BinningProductRatings/Rating1-m-00000 | head
2019-12-14 21:32:43,867 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
US      27885926        R1YND4BS823GN5  B00HRXSSRA      708418657       ChiliPower DMW-BLC12, DMW-BLC12E, DMW-BLC12PP 1450mAh Battery for Panasonic Lumix DMC-F
Z200, DMC-G5, DMC-G6, DMC-G6K, DMC-GH2  Camera  1       0       0       N       Y       Sucky.  Lasted a few hours with its first charge, which is fine. But wi
th subswquent charges it only held its charge for about 30 minutes or less before dying. Sucky. 2015-08-31
US      6465510 R1QNYFW6G31R5T  B00EDCZKJ2      258297575       ZINK Phone Photo & Labels Wireless Printer. Wi-Fi Enabled. Print Directly from IOS & Android Sm
art Phones, Tablets. Includes FREE Arts & Crafts App.   Camera  1       1       3       N       N       Never Agian.    If I could leave a 0 I would, I was so
excited about using this product. I could not wait for it to come in, and when I tested the product on the .5 inch roll it comes with it worked perfectly. I al
so tested it on different wifi connections, with the 2 inch roll. Everything worked fine. (My plan was to use this for a wedding photobooth) during the rehears
al it worked perfectly again, got some test shots in with the bride. But morning of the wedding day it wouldn't work.... turned it off, on, kept it on the same
 wifi, tried a different wifi.... nothing was working. I even tried a different ipad.... still nothing..... I have no idea what happened, but I do know it was
not worth the headache. Good luck to all other buyers, hopefully you won't have my luck.       2015-08-31
US      11402392        RA77423YFRUVF   B00XZGBW44      236367800       Sricam SP720 Plug & Play Wireless IP Camera 720p HD - Video Monitoring, Home Surveillan
ce, Baby Monitor - Two Way Audio, Night Vision, Motion Sensor – Monitor On iPhone, iPad, and Android    Camera  1       0       0       N       Y       BAD     N
ot usable due to its a new product and not recognized by softwear...since only support is in JAPAN it is impossible to get help.  Do not buy!!  2015-08-31
US      22125044        R2DTIYMTXQPAJV  B0046C8BWM      992311641       57" Tripod Kit For The Canon SX510 HS, SX520 HS, SX720 HS, S120, SX500 IS, SX280 HS, SX
150 IS, SX400 IS, G12, G1 X, G1X, SX50 HS, G15, G16, SX60 HS, G3 X, G1X Mark II, G9 X, G5 X, G7 X, G7 X Mark II Digital Camera  Camera  1       0       0       D
on't get suckered       The second or third time I used it one leg bent and now won't retract.  The other legs however won't stay extended -- the locks don't w
ork half the time.  Racheting the top post up doesn't work either, it won't stay put and keeps falling.  Complete waste of money.       2015-08-31
US      52859599        R12KFKM60SHKF2  B002GUF90I      509362904       Nikon ML-L3 Remote Control for Nikon D40, D40x, D60, D80 & D90 Digital SLR Cameras + Ze
ikos 75 Inch Full Size Tripod with Exclusive FREE Complimentary EXCLUSIVELY DESIGNED BLUE BENDY FLEXPOD Trademarked By ClearMax Camera  1       0       0       D
ud      It never worked.  I should have sent it back.   2015-08-31
US      29739494        RB6NEA7FPIARO   B00TXM8CWY      148531063       Panasonic Lumix DMC-GH4 4K Micro Four Thirds Digital Camera Body with 64GB Card + Case
+ Flash + 2 Batteries + Microphone & LED Light + Stabilizer Kit Camera  1       13      14      N       Y       Did not get what was promised! This is a ...  D
id not get what was promised! This is a scam, be aware! My friend ordered the same package deal and was scammed as well. Amazon needs to put an end to these ty
pes of empty promises! The ad looks to food to be true with over $300 worth of extras, but when you get your package it is all generic except the GH4. SCAM!  2
015-08-31
US      28334098        R30OJ7ZIFGUFF1  B005M226Y4      297607479       Alzo Multi-Mount for Attaching Video Gear- Incl. Microphones & Lights To Dslr Or Camcor
ders Or Video Cameras   Camera  1       2       2       N       Y       It looked cool, and it also looked like it was just ... I dont recomend this product at
 all. Each of the sliders are different sizes! they are just slightly off so they dont fit every mount. There are also no stoppers on the mount slides at all.
So if the mount becomes loose, and your camera shifts forward, all the equipment on each of the mounts will plumit to the ground. The center bracket is so smal
l it does not fit anything im wanting to put there. Not really sure what the poiint of it is.<br /><br />I was really excited about this product. It looked coo
l, and it also looked like it was just what I needed. But that was not the case.<br /><br />Really un happy with this, and will ship it back    2015-08-31
US      45046643        R1PRCVFTGK54HZ  B0046710G6      480886069       Foscam FI8918W Wireless/Wired Pan & Tilt IP/Network Camera with 8 Meter Night Vision an
d 3.6mm Lens (67° Viewing Angle)        Camera  1       1       1       N       Y       No continuous streaming Ended up returning this item as it did not corr
ectly describe that it does not support continuous streaming.   2015-08-31
US      1589319 R185ZLPNVU1QEJ  B00A8MP3GQ      559200732       dCables Canon PowerShot A4000 IS Battery Charger - Wall & Travel Charger for PowerShot A4000 IC
amera   1       0       0       N       Y       not satisfied   The device didn't work .        2015-08-31
US      34839788        R2OM6OYRFC1EX5  B000BNY64C      853708552       STK's Canon BP-511 2200mAh Battery for Select Canon Cameras (2 Pack)    Camera  1       S
pam     Buyer Beware: After purchasing this, I begun getting spam email from SterlingTek.       2015-08-31
cat: Unable to write to output stream.
(base) UTD04640:hadoop agandhi$
```

8. Recommend the products to the user based on the star rating:

Apache Mahout is used to Recommend the products to the user based on the star rating. Here, data is first cleaned to get the userID, productID and star_rating. UserID from this cleaned data is then used as input to get recommended products on the basis of star rating. MapReduce chaining is used for this recommender implementation.

Final Output: UserID, ProductID, start rating

Recommendation Data Mapper Implementation:

```java
package com.neu.bigdata.amazonAnalysis.mahoutRecommendation;

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

import java.io.IOException;

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

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

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

            else{
                String[] line = value.toString().split( regex: "\\t");
                Text output = new Text();
                output.set(line[1] + "," + line[4] + "," + line[7]); // 1: userID, 4:productID, 7:Rating

                context.write(NullWritable.get(), output);
            }

        } catch(Exception  e){
            System.out.println("Something went wrong in Recommendation Data Mapper Task: ");
            e.printStackTrace();
        }
    }
}
```

User Data Mapper Implementation:

```java
package com.neu.bigdata.amazonAnalysis.mahoutRecommendation;

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

import java.io.IOException;

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

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

        try {
            String[] line = value.toString().split( regex: ",");
            String userId = line[0].trim();
            context.write(new Text(userId), NullWritable.get());

        } catch (Exception e) {
            System.out.println("Something went wrong in User Mapper Task: ");
            e.printStackTrace();
        }
    }
}
```

Reducer Implementation:

```java
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 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( n: 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) {
            e.printStackTrace();
        }   catch (IOException e) {
            e.printStackTrace();
        }
```

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

        try {

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

            if (!recs.isEmpty()) {

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

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

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

Main Implementation:

```java
public class RecommendationMain {

    final static Logger logger = Logger.getLogger(RecommendationMain.class);

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

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        boolean mahoutGetDataJobSuccesful = false;
        Path inputPath = new Path(args[0]);
        Path dataOutputPath = new Path(args[1]);
        Path recommedationOutputPath = new Path(args[2]);
        long startTime = System.currentTimeMillis();

        try {

            Job getDataJob = Job.getInstance(conf,  jobName: "Get recommendation data");
            getDataJob.setJarByClass(com.neu.bigdata.amazonAnalysis.mahoutRecommendation.RecommendationMain.class);

            getDataJob.setMapperClass(RecommendationDataMapper.class);
            getDataJob.setMapOutputKeyClass(NullWritable.class);
            getDataJob.setMapOutputValueClass(Text.class);
            getDataJob.setOutputKeyClass(NullWritable.class);
            getDataJob.setOutputValueClass(Text.class);
            getDataJob.setNumReduceTasks(1);

            FileInputFormat.setInputPaths(getDataJob, inputPath);
            FileOutputFormat.setOutputPath(getDataJob, dataOutputPath);

            if (fs.exists(dataOutputPath)) {
                fs.delete(dataOutputPath,  b: true);
            }

            mahoutGetDataJobSuccesful = getDataJob.waitForCompletion( verbose: true);

            if(mahoutGetDataJobSuccesful) {
```

```
        if(mahoutGetDataJobSuccesful) {

            Job recommendationJob = Job.getInstance(conf, jobName: "Recommendation");
            String path = dataOutputPath.toString();
            recommendationJob.getConfiguration().set("DataPath", path);

            recommendationJob.setJarByClass(com.neu.bigdata.amazonAnalysis.mahoutRecommendation.RecommendationMain.class);
            FileInputFormat.setInputPaths(recommendationJob, dataOutputPath);
            FileOutputFormat.setOutputPath(recommendationJob, recommedationOutputPath);

            recommendationJob.setMapperClass(UserDataMapper.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, b: true);
            }

            recommendationJob.waitForCompletion( verbose: true);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        }
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Something went wrong in main recommendation: ");
        e.printStackTrace();
    }

}
}
```

Command to run mapreduce:

sudo hadoop jar /Users/agandhi/Desktop/bigdata/project/out/artifacts/MahoutRecommendation/AmazonAnalysis.jar com.neu.bigdata.amazonAnalysis.mahoutRecommendation.RecommendationMain /project/dataset /project/output/MahoutRecommendation/data /project/output/MahoutRecommendation/recommendation

Output:

```
(base) UTD04640:hadoop agandhi$ sudo hadoop fs -cat /project/output/MahoutRecommendation/data/part-r-00000 | h
2019-12-14 21:45:42,547 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... 
52430974,411935344,5
52859048,411935344,5
52267609,532198566,5
51306730,532198566,5
51967237,532198566,5
51709592,531961783,3
52133593,601562672,5
50905673,130421062,4
53075717,531961783,2
52995474,532198566,4
```

9. **Find the count of the reviews partitioned by date for each product.**

Apache Pig is used to do the above analysis and find the count of reviews grouped by date for each product.

Pig Script:

```
raw_data = load '/Users/agandhi/Desktop/bigdata/final_project/data/amazon_reviews_us_Camera_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 raw_data 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 '/Users/agandhi/Desktop/bigdata/final_project/pig';
```

10. Find the count of products for each product star rating.

Apache Pig is used to do the above analysis and find the count of products for each product star rating.

Pig Script:

```
raw_data = load '/Users/agandhi/Desktop/bigdata/final_project/data/amazon_reviews_us_Camera_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 raw_data 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 '/Users/agandhi/Desktop/bigdata/final_project/pig';
```

Appendix (Source Code for all above analysis):

1. **Find the total number of products present in the dataset**

```
package com.neu.bigdata.amazonAnalysis.totalProducts;

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

import java.io.IOException;

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

    private Text word = new Text();
```

```
    private IntWritable one = new IntWritable(1);

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

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

            word.set(productId);
            context.write(word, one);
        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }

    }
}
```

```
package com.neu.bigdata.amazonAnalysis.totalProducts;

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

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

    public void reduce(Text key, Iterable<IntWritable> values, Context context) {

        try {

            int sum = 0;
            for(IntWritable val: values) {
                sum += val.get();
            }

            IntWritable count = new IntWritable(sum);
            context.write(key, count);

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.totalProducts;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
```

```java
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.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;
import org.apache.log4j.Logger;

// Count total number of products

public class ProductCountMain {

    private final static Logger logger =
Logger.getLogger(com.neu.bigdata.amazonAnalysis.totalProducts.ProductCountMain.class);
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        Configuration conf = new Configuration();
        try {
            long startTime = System.currentTimeMillis();

            Job job = new Job(conf, "ProductCount");
            job.setJarByClass(ProductCountMain.class);

            job.setMapperClass(ProductCountMapper.class);
            job.setReducerClass(ProductCountReducer.class);

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

            FileInputFormat.addInputPath(job, new Path(args[1]));
            FileOutputFormat.setOutputPath(job, new Path(args[2]));

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

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

            job.setNumReduceTasks(1);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);
            System.exit(job.waitForCompletion(true) ? 0 : 1);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }

    }
}
```

2. Find the average product rating reviews for each product

```
package com.neu.bigdata.amazonAnalysis.AverageProductRating;

import org.apache.hadoop.io.Writable;

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

public class CountAverageTuple implements Writable {

    private Long count;
    private Float average;

    public CountAverageTuple(){

    }

    public CountAverageTuple(Long count, Float average) {
        this.count = count;
        this.average = average;
    }

    public void write(DataOutput d) throws IOException {
        d.writeLong(count);
        d.writeFloat(average);
    }

    public void readFields(DataInput di) throws IOException {
        count = di.readLong();
        average = di.readFloat();
    }

    public Long getCount() {
        return count;
    }

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

    public Float getAverage() {
        return average;
    }

    public void setAverage(Float average) {
        this.average = average;
    }

    @Override
    public String toString() {
```

```
        return (new StringBuilder().append(count).append("\t").append(average).toString());
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.AverageProductRating;

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

import java.io.IOException;


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

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

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

        try {

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

            if (!productId.isEmpty()) {
                id.set((productId));
                outCountAverage.setCount(Long.valueOf(1));
                outCountAverage.setAverage(Float.valueOf(input[7].trim()));
                context.write(id, outCountAverage);
            }

        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }

    }
}
```

```
package com.neu.bigdata.amazonAnalysis.AverageProductRating;

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

import java.io.IOException;

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

    private CountAverageTuple result = new CountAverageTuple();

    public void reduce(Text key, Iterable<CountAverageTuple> value, Context context)
```

```
        throws IOException, InterruptedException {

        try {
            long count = 0;
            float sum = 0;

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

            result.setCount(count);
            result.setAverage(sum/count);
            context.write(key, result);

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }

}
```

```
package com.neu.bigdata.amazonAnalysis.AverageProductRating;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.io.Text;
import java.io.IOException;
import org.apache.log4j.Logger;

// Find the average product rating reviews for each product

public class ProductAverageRatingMain {

    final static Logger logger = Logger.getLogger(ProductAverageRatingMain.class);

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException
    {
        try {
            long startTime = System.currentTimeMillis();
            Job job = Job.getInstance();
            job.setJarByClass(ProductAverageRatingMain.class);

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

            job.setMapperClass(ProductAverageRatingMapper.class);
            job.setReducerClass(ProductAverageRatingReducer.class);
```

```
        job.setCombinerClass(ProductAverageRatingReducer.class);

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

        job.setNumReduceTasks(1);

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

        job.waitForCompletion(true);
        long endTime = System.currentTimeMillis();
        logger.info("Time taken in milliseconds : " + (endTime - startTime));
        logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

      } catch (Exception e) {
        System.out.println("Something went wrong in main class: ");
        e.printStackTrace();
      }
   }
}
```

3. Find the topN reviewed products sorted by count

```
package com.neu.bigdata.amazonAnalysis.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;
   }
}
```

```
package com.neu.bigdata.amazonAnalysis.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){
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.topNProducts;

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

import java.io.IOException;

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

    int count = 0;
    private int N = 10;

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

    @Override
    public void reduce(IntWritable key, Iterable<Text> value, Context context)
            throws IOException, InterruptedException{
        try {
            for(Text val: value){
                if(count<N)
                {
                    context.write(key,val);
                }
                count++;
            }

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
```

```
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.topNProducts;

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.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.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.log4j.Logger;

import java.io.IOException;

// Find the topN reviewed products sorted by count
// Here, secondary sorting technique is used with implementation of comparator class

public class TopNProductsMain {

    final static Logger logger = Logger.getLogger(TopNProductsMain.class);
    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        try {
            long startTime = System.currentTimeMillis();
            Job topNProductsJob = Job.getInstance(conf, "Top N Rated Products");
            topNProductsJob.setJarByClass(com.neu.bigdata.amazonAnalysis.topNProducts.TopNProductsMain.class);

            int N = 10;
            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, new Path(args[0]));  // Output file path of totalProducts - MR
chaining
```

```
            FileOutputFormat.setOutputPath(topNProductsJob, new Path(args[1]));
            if (fs.exists(new Path(args[1]))) {
                fs.delete(new Path(args[1]), true);
            }

            topNProductsJob.waitForCompletion(true);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();

        }
    }
}
```

4. **Find total product rating for all those products which are present in the topN reviewed products**

```
package com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin;

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

import java.io.IOException;

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

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

        Text productId = new Text();
        Text count = new Text();

        try {
            String[] input = value.toString().split("\\t");
            productId.set(input[1].trim());
            count.set("#"+ input[1] + " "+ input[0].trim());

            context.write(productId, count);

        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper 1 Task: ");
            e.printStackTrace();
        }
    }
}
```

```java
package com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin;

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

import java.io.IOException;

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

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

        Text productId = new Text();
        Text rating = new Text();

        try {
            String[] input = value.toString().split("\\t");
            productId.set(input[0].trim());
            rating.set("*" + input[2].trim());

            context.write(productId, rating);

        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper 2 Task: ");
            e.printStackTrace();
        }
    }
}
```

```java
package com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin;

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

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

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

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

        try {
            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));
                    }
                }
            }
        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.log4j.Logger;

import java.io.IOException;

// Find average product rating for all those products which are present in the topN reviewed products
// An inner join is used to Product id, review count and average rating
// Multiple input is passed as input for join to be performed

public class JoinMain {

    final static Logger logger = Logger.getLogger(JoinMain.class);
    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        try {
            long startTime = System.currentTimeMillis();
            Job joinsJob = Job.getInstance(conf, "Join");
            joinsJob.setJarByClass(com.neu.bigdata.amazonAnalysis.reduceSideInnerJoin.JoinMain.class);

            Path topNProductsOutputPath = new Path(args[0]);
            Path summarizationOutputPath = new Path(args[1]);
            MultipleInputs.addInputPath(joinsJob, topNProductsOutputPath, TextInputFormat.class,
TopProductsMapper.class);
            MultipleInputs.addInputPath(joinsJob, summarizationOutputPath, TextInputFormat.class, RatingsMapper.class);
            FileOutputFormat.setOutputPath(joinsJob, new Path(args[2]));
```

```
        joinsJob.setReducerClass(JoinReducer.class);

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

        if (fs.exists(new Path(args[2]))) {
            fs.delete(new Path(args[2]), true);
        }

        joinsJob.waitForCompletion(true);
        long endTime = System.currentTimeMillis();
        logger.info("Time taken in milliseconds : " + (endTime - startTime));
        logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

    } catch (Exception e) {
        System.out.println("Something went wrong in main class: ");
        e.printStackTrace();
    }

  }
}
```

5. Find all the users who has reviewed each product

```
package com.neu.bigdata.amazonAnalysis.invertedIndexPattern;

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

import java.io.IOException;

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

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

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

      if(key.get()==0){
         return;
      }
      try{
         String[] tokens = value.toString().split("\\t");
         userId.set(tokens[1]);
         productId.set(tokens[3]);
         context.write(productId, userId);

      } catch(Exception  e){
```

```
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace();
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.invertedIndexPattern;

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

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 {

        try {
            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);

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.invertedIndexPattern;

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

```java
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.log4j.Logger;

import java.io.IOException;

//This pattern is used to find each user who has reviewed the product

public class InvertedIndexMain {

    final static Logger logger = Logger.getLogger(InvertedIndexMain.class);
    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        try {
            long startTime = System.currentTimeMillis();
            Job invertedIndexJob = Job.getInstance(conf, "Inverted Index");

invertedIndexJob.setJarByClass(com.neu.bigdata.amazonAnalysis.invertedIndexPattern.InvertedIndexMain.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, new Path(args[0]));
            FileOutputFormat.setOutputPath(invertedIndexJob, new Path(args[1]));
            if (fs.exists(new Path(args[1]))) {
                fs.delete(new Path(args[1]), true);
            }

            invertedIndexJob.waitForCompletion(true);
            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        } catch (Exception e) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }
    }
}
```

6. **Find all the records partitioned by the date in which the product was reviewed**

```java
package com.neu.bigdata.amazonAnalysis;

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.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

// Find all the records partitioned by the year in which the product was reviewed

public class YearPartitioner {

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

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf,"Partitioning");

        job.setJarByClass(com.neu.bigdata.amazonAnalysis.YearPartitioner.class);

        job.setMapperClass(YearPartitionMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        //Custom Partitioner:
        job.setPartitionerClass(YearPartitionPartitioner.class);

        job.setReducerClass(YearPartitionReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);
        job.setNumReduceTasks(14);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        if (fs.exists(new Path(args[1]))) {
            fs.delete(new Path(args[1]), true);
        }

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

    public static class YearPartitionMapper extends Mapper<LongWritable, Text, Text, Text> {

        private Text inputRec = new Text();
        private Text year = new Text();
```

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

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

            String[] line = value.toString().split("\\t");
            String[] yearPart = line[14].split("-");
            String yearVal = yearPart[2].trim();

            year.set(yearVal);
            inputRec.set(value);

            context.write(year, inputRec);
        }
    }

    public static class YearPartitionPartitioner extends Partitioner<Text, Text> {
        @Override
        public int getPartition(Text key, Text value, int numPartitions){
            int n=1;
            if(numPartitions==0){
                return 0;
            }
            else if(key.equals(("99"))){
                return n % numPartitions;
            }
            else if(key.equals(new Text("00"))){
                return 2 % numPartitions;
            }
            else if(key.equals(new Text("01"))){
                return 3 % numPartitions ;
            }
            else if(key.equals(new Text("02"))){
                return 4 % numPartitions;
            }
            else if(key.equals(new Text("03"))){
                return 5 % numPartitions;
            }
            else if(key.equals(new Text("04"))){
                return 6 % numPartitions;
            }
            else if(key.equals(new Text("05"))){
                return 7 % numPartitions;
            }
            else if(key.equals(new Text("06"))){
                return 8 % numPartitions;
            }
            else if(key.equals(new Text("07"))){
                return 9 % numPartitions;
            }
            else if(key.equals(new Text("08"))){
```

```
                return 10 % numPartitions;
            }
            else if (key.equals(new Text("09"))){
                return 11 % numPartitions;
            }
            else if (key.equals(new Text("10"))){
                return 12 % numPartitions;
            }
            else if (key.equals(new Text("11"))){
                return 13 % numPartitions;
            }
            else
            {
                return 14 % numPartitions;
            }
        }
    }

    public static class YearPartitionReducer extends Reducer<Text, Text, Text, NullWritable> {

        protected void reduce(Text key, Iterable<Text> values, Reducer.Context context) throws IOException,
InterruptedException{
            for(Text t: values){

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

7. **Find all the product information for each unique star rating of the product by dividing it into different categories/bins**

```
package com.neu.bigdata.amazonAnalysis.BinReviewRatings;

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;

import java.io.IOException;

public class BinningProductReviewRatingsMapper 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{
```

```
        try {
            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(), "Rating1"); }
            if(rating.equals("2")){
                output.write("bins", value, NullWritable.get(), "Rating2"); }
            if(rating.equals("3")){
                output.write("bins", value, NullWritable.get(), "Rating3"); }
            if(rating.equals("4")){
                output.write("bins", value, NullWritable.get(), "Rating4"); }
            if(rating.equals("5")){
                output.write("bins", value, NullWritable.get(), "Rating5"); }
        } catch (Exception e) {
            System.out.println("Something went wrong in Mapper Task: ");
            e.printStackTrace(); }
    }

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

```
package com.neu.bigdata.amazonAnalysis.BinReviewRatings;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
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 org.apache.log4j.Logger;

import java.io.IOException;

//Binning pattern is used to divide the reviews based on the ratings in separate bins.
//This is map-side pattern
//Multiple outputs are generated for each bin

public class BinningProductReviewRatingsMain {

    final static Logger logger =
Logger.getLogger(com.neu.bigdata.amazonAnalysis.BinReviewRatings.BinningProductReviewRatingsMain.class);
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        Configuration conf = new Configuration();
```

```
        FileSystem fs = FileSystem.get(conf);

        try {
            long startTime = System.currentTimeMillis();

            Job binningJob = Job.getInstance(conf, "Binning Pattern");

binningJob.setJarByClass(com.neu.bigdata.amazonAnalysis.BinReviewRatings.BinningProductReviewRatingsMain.class)
;

            binningJob.setMapperClass(BinningProductReviewRatingsMapper.class);
            binningJob.setMapOutputKeyClass(Text.class);
            binningJob.setMapOutputValueClass(NullWritable.class);
            binningJob.setNumReduceTasks(1);

            FileInputFormat.setInputPaths(binningJob, new Path(args[0]));
            FileOutputFormat.setOutputPath(binningJob, new Path(args[1]));
            if (fs.exists(new Path(args[1]))) {
                fs.delete(new Path(args[1]), true);
            }

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

            long endTime = System.currentTimeMillis();
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);
            System.exit(binningJob.waitForCompletion(true) ? 0 : 1);

        } catch (Exception e ) {
            System.out.println("Something went wrong in main class: ");
            e.printStackTrace();
        }
    }
}
```

8. Recommend the products to the user based on the star rating

```
package com.neu.bigdata.amazonAnalysis.mahoutRecommendation;

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

import java.io.IOException;

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

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

        try {
            if(key.get()==0){
```

```
            return;
        }

        else{
            String[] line = value.toString().split("\\t");
            Text output = new Text();
            output.set(line[1] + "," + line[4] + "," + line[7]); // 1: userID, 4:productID, 7:Rating

            context.write(NullWritable.get(), output);
        }

    } catch(Exception  e){
        System.out.println("Something went wrong in Recommendation Data Mapper Task: ");
        e.printStackTrace();
    }
  }
}
```

```
package com.neu.bigdata.amazonAnalysis.mahoutRecommendation;

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

import java.io.IOException;

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

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

    try {
        String[] line = value.toString().split(",");
        String userId = line[0].trim();
        context.write(new Text(userId), NullWritable.get());

    } catch (Exception e) {
        System.out.println("Something went wrong in User Mapper Task: ");
        e.printStackTrace();
    }
  }
}
```

```
package com.neu.bigdata.amazonAnalysis.mahoutRecommendation;

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;

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

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 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) {
         e.printStackTrace();
      }  catch (IOException e) {
         e.printStackTrace();
      }

   }

   @Override
```

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

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

        } catch (Exception e) {
            System.out.println("Something went wrong in Reducer Task: ");
            e.printStackTrace();
        }
    }
}
```

```
package com.neu.bigdata.amazonAnalysis.mahoutRecommendation;


import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.log4j.Logger;

import java.io.IOException;

// This is used to recommend the products based on rating to the user
// UserID, ProductID and star_rating consists of data for recommendation

public class RecommendationMain {

    final static Logger logger = Logger.getLogger(RecommendationMain.class);

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

        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        boolean mahoutGetDataJobSuccesful = false;
```

```
        Path inputPath = new Path(args[0]);
        Path dataOutputPath = new Path(args[1]);
        Path recommedationOutputPath = new Path(args[2]);
        long startTime = System.currentTimeMillis();

        try {

            Job getDataJob = Job.getInstance(conf, "Get recommendation data");

getDataJob.setJarByClass(com.neu.bigdata.amazonAnalysis.mahoutRecommendation.RecommendationMain.class);

            getDataJob.setMapperClass(RecommendationDataMapper.class);
            getDataJob.setMapOutputKeyClass(NullWritable.class);
            getDataJob.setMapOutputValueClass(Text.class);
            getDataJob.setOutputKeyClass(NullWritable.class);
            getDataJob.setOutputValueClass(Text.class);
            getDataJob.setNumReduceTasks(1);

            FileInputFormat.setInputPaths(getDataJob, inputPath);
            FileOutputFormat.setOutputPath(getDataJob, dataOutputPath);

            if (fs.exists(dataOutputPath)) {
                fs.delete(dataOutputPath, true);
            }
            mahoutGetDataJobSuccesful = getDataJob.waitForCompletion(true);

            if(mahoutGetDataJobSuccesful) {

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


recommendationJob.setJarByClass(com.neu.bigdata.amazonAnalysis.mahoutRecommendation.RecommendationMain.
class);
                FileInputFormat.setInputPaths(recommendationJob, dataOutputPath);
                FileOutputFormat.setOutputPath(recommendationJob, recommedationOutputPath);

                recommendationJob.setMapperClass(UserDataMapper.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);
                }

                recommendationJob.waitForCompletion(true);
                long endTime = System.currentTimeMillis();
```

```
            logger.info("Time taken in milliseconds : " + (endTime - startTime));
            logger.info("Time taken in seconds : " + (endTime - startTime)/1000);

        }
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Something went wrong in main recommendation: ");
        e.printStackTrace();
    }

  }
}
```

9. Find the count of the reviews partitioned by date for each product

```
raw_data = load '/Users/agandhi/Desktop/bigdata/final_project/data/amazon_reviews_us_Camera_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 raw_data 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 '/Users/agandhi/Desktop/bigdata/final_project/pig';
```

11. Find the count of products for each product star rating

```
raw_data = load '/Users/agandhi/Desktop/bigdata/final_project/data/amazon_reviews_us_Camera_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 raw_data 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 '/Users/agandhi/Desktop/bigdata/final_project/pig';
```