USING

- Kaushik Umesh Patil
001056490

Technologies Used

1. Hadoop MapReduce

2. Apache Hive

3. Apache Pig

4. Tableau

Dataset

Dataset Link ->

https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Sports_v1_00.tsv.gz

The data is Amazon Sports available in TSV files in the amazon-reviews-pdf S3 bucket in AWS US East Region. Each line in the data files corresponds to an individual review (tab delimited, with no quote and escape characters).

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 spelter 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.

Dataset Screenshots

marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine
US	48945260	R1WBPB8MDCCN8F	B012P7UPSM	409940130	Chicago Blackhawks Adult Cuff Knit Beanie w/ Pom One Size Fits All NHL Authentic Hat Cap - OSFA	Sports	5	0	0	N
US	5782091	R32M0YEWV77XG8	B001GQ3VHG	657746679	Copag Poker Size Regular Index 1546 Playing Cards 2 decks (Black Gold Setup)	Sports	5	1	1	N
US	45813853	RR8V7WR27NXJ5	B008VS8M58	962442336	Baoer 223 5.56x45mm Caliber Cartridge Laser Bore Sighter Boresighter	Sports	1	0	0	N
US	1593730	R1MHO5V9Z932AY	B005F06F4U	74305227	All Terrain Tackle Jig - Grass Master - June Bug - 3/4oz- 2 Pack	Sports	5	0	0	N
US	29605511	R16PD71086BD2V	B010T4IE2C	787185588	Swim Cap - 3 Pack (Blue, Black & Red)	Sports	5	0	1	N
US	11112959	RIZBIFGWTRWXT6	B004RKJGL5	94127483	adidas Men's Sport Performance Climalite Boxer Brief, Two-Pack	Sports	3	0	0	N
US	108031	R3AUMSHAW73HWN	B005V3DCBU	526977496	Nike Men's Performance Cotton Cushioned Crew Socks	Sports	4	2	3	N
US	13981540	R2KWDWFOHGX6FL	B00MHT9WN8	26521265	Green Bay Packers NFL Team Apparel Women's Forward Progress Shirt Plus Sizes	Sports	5	0	0	N
US	37993909	R3H9543FWBWFBU	B001CSIRQ8	652431165	Isokinetics Inc. Adjustable Balance Board - 16.5" Round - 2 Levels of Difficulty - For Exercise and Physical Therapy	Sports	5	1	1	N
US	26040213	RUANXOQ9W3OU5	B001KZ3NOO	635861713	Aottop Quality Electronic Digital Vernier Caliper	Sports	5	0	0	N
US	34657602	R31673RTGEZSW7	B00005RCQ5	72099763	Everlast 2'x6' Folding Mat	Sports	5	2	2	N
US	14346192	R22OQLFSH42RCM	B00FA7RWVI	757354022	OGIO Men's Shredder Stand Bag	Sports	5	1	1	N
US	38782687	R12LEL4F3TSZUJ	B000MMH2HU	272402599	Blue Sea Systems A-Series Double Blank Mounting Panel	Sports	5	2	2	N
US	27138575	R2L9XWD03072NI	B00WJOATDS	375070208	Zengi AR15 45 Degree Offset Front and Rear Flip up Iron Sights, Back-up Iron Sights B.I.U.S	Sports	5	1	2	N
US	11838771	R2K0U91HIACANO	B00BOM2JNY	639666785	Black Mountain Products Static Strength Exercise Stability Ball with Pump, 2000 lb/55cm, Blue	Sports	5	0	0	N
US	535800	R29SP6MBT7MXG7	B00OD5GAFM	38715442	Trijicon VCOG 1-6x24 Riflescope Ballistic Reticle	Sports	4	3	3	N
US	23156579	R31XREAAMATEPY	B00CAHDC1K	57088652	Naruto Headband for Ninja Shinobi - Sand Village (Sunagakure)	Sports	5	0	0	N
US	48107879	R2116AVB87SO38	B004NLHXLG	114125984	Under Armour Men's Resistor Low-Cut Socks (6 Pack)	Sports	5	0	0	N
US	27260960	R3RDVBB6O0X3HW	B00L7OANWI	883962979	Yes4All Deep Tissue Massage AccuPoint Roller (Clearance Sale)	Sports	5	0	0	N
US	39537314	R3LW5T149LKKQM	B0085PPSIQ	691479969	Lansky PS-MED01 BladeMedic	Sports	4	0	0	N
US	18457385	R1THNVUD55C27K	B003WWQO94	567685590	Champion Sports Colored Lacrosse Balls: Official Size for Professional, College & Grade School Games - NCAA, NFHS, Certified	Sports	5	0	0	N
US	28689758	R2V1NIGCEDQMWR	B00APSXAS4	356387565	Callaway Men's RAZR X NG Iron Set	Sports	5	0	0	N
US	35214079	RIJXY4FJE2U7O	B00ZPTA41G	511703314	NCAA Hover Helmet - Collectible Levitating Football Helmet with Electromagnetic Stand	Sports	5	0	0	N
US	7572409	R3OI78MWLX18YH	B00AAHIRKS	191288137	Gerber 31-001098 Remix Tactical	Sports	1	1	2	N
US	41582860	R1NP0HLBWDXTG3	B0099WGZRW	521874476	inForce WML, Multifunction Weapon Mounted Light, White and Infrared LED, 200 INF-WML-S-W-IR	Sports	1	0	0	N
US	38139271	RFZBXOMZWEYPV	B00K5503O8	40527220	Fixxxer Rear Cover Plate fits Springfield XDs, Tactical Skull design	Sports	5	0	0	N
US	106639	R1XW7YNDY1HHV2	B00GV6J8WI	896379297	Wilson NFL MVP Football	Sports	5	0	0	N
US	22885356	RXQWVQ6XDQ4L6	B00NSWHZCS	396708923	Rukket 10x7ft Haack Golf Net | Practice Driving Indoor and Outdoor | Golfing at Home Swing Training Aids | By SEC Coach Chris Haack	Sports	5	1	2	N
US	38468354	RPCITKSJ62BBH	B013F06TCO	273588985	Rodut (TM) Adjustable Right Handed Tactical Leg Holster For Pistol,	Sports	3	0	0	N
US	11228537	R10H8KSR2XWCRO	B00OY5EH1M	101613319	Kidder Infant Flexback Biolite Life Jacket	Sports	5	0	0	N
US	15342235	R1F6HW2IAXFZOE	B00NAHA6HC	330547128	Ultimate Arms Gear 5 Pack 357 Magnum 38 Special Full Moon Clips 7 Round Smith & Wesson S&W Model 686 & 7 Shot L Frames Revolvers	Sports	5	0	0	N
US	28088591	R34TV89J7GOPWJ	B001CJOA6	728149528	Under Armour Men's HeatGear No Show Socks (3 Pair)	Sports	5	0	0	N
US	20093436	RD8O39QNCLDJN	B002TSPE5C	942225482	Sitting Stepper	Sports	5	0	0	N
US	22750110	RHYA42XJUM2TE	B00545UBYQ	904724650	Leather Front Pocket Holster For COLT POCKETLITE .380	Sports	5	1	1	N
US	44801274	R1DUQZYOS0NA7Y	B00169CU66	605693953	Total Gym XLS ,Â Universal Home Gym for Total Body Workout	Sports	4	0	1	N
US	1454979	R2INQ2834TESLU	B005DUUL9W	191246478	BEP Battery Switches	Sports	5	0	0	N
US	36539492	RTWRRMR2Q5BO5	B00T4XAVKS	972354716	Black & Denver Orange Spirit Pom Poms	Sports	3	0	0	N
US	21391837	R2EJXGTZLFDBFS	B00806KGY6	647250004	Covert Code Black Wireless Game Camera Mossy Oak	Sports	4	1	1	N
US	23536197	R3EE10IYRRQG98	B001CF3L80	3390781	Jugs Bucket of Small-Balls (4 dozen)	Sports	5	0	1	N
US	44877916	R2QL1B82PSWQD0	B00162UHEA	839287489	Lee Precision 3 Jaw Chuck	Sports	5	0	0	N
US	41703431	R1XZYB1K4IANYR	B000IDRRY6	385405694	Rothco Vintage Paratrooper Fatigues	Sports	5	0	0	N

amazon_reviews_us_Sports_v1_00

```
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$ hadoop fs -copyFromLocal amazon_reviews_us_Sports_v1_00.tsv /AmazonSports
2020-07-31 04:51:06,169 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-07-31 04:51:06,914 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:07,596 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:08,074 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:08,425 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:08,842 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:09,243 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:09,640 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:10,055 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:10,432 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:10,786 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:11,088 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:11,422 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:11,762 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:12,144 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-31 04:51:12,695 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$ hadoop fs -ls /AmazonSports
2020-07-31 04:51:56,082 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 kaushikpatil supergroup 2007024927 2020-07-31 04:51 /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$ hadoop fs -count /AmazonSports
2020-07-31 04:52:21,703 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
           1            1         2007024927 /AmazonSports
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$ hadoop fs -head /AmazonSports
2020-07-31 04:52:39,592 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
head: `/AmazonSports': Is a directory
(base) Kaushiks-MacBook-Pro:AmazonSportsReviews kaushikpatil$
```

hadoop fs -copyFromLocal amazon_reviews_us_Sports_v1_00.tsv /AmazonSports

hadoop fs -ls /AmazonSports

The Big Data Analysis of Amazon Sports Review will be done using Hadoop MapReduce, Apache Hive and Apache Pig and Tableau.

MapReduce: It is the core component for data processing in Hadoop framework. In layman's term MapReduce helps to split the input data set into a number of parts and run a program on all data parts parallel at once. The term MapReduce refers to two separate and distinct tasks. The first is the map operation, takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce operation combines those data tuples based on the key and accordingly modifies the value of the key.

Apache Hive: It is a platform used to develop SQL type scripts to do MapReduce operations.

Apache Pig: It is a procedural language platform used to develop a script for MapReduce operations.

Tableau: Tableau is the fastest growing data visualization tool that aims to help people see and understand data. In other words, it simply converts raw data into a very easily understandable format.

Deliverables

Total 15+ unique operations achieved.

Hadoop

1. Overall Average Rating of all Products (Basic MapReduce)
2. Average Rating of all Products based on the Year (SecondorySorting Technique)
3. Number of Products per Rating (Counter using Numerical Summarization Pattern)
4. Segregation of all Products based on their rating (Binning Organization Pattern)
5. Top 25 Best Products based on average ratings (Filtering Pattern and Job Chaining)
6. List of customers for each product who have given review (Distinct or Inverted Index Pattern)
7. Legit Customer Review Analysis (Numerical Summarization Pattern)
8. Average of ratings and list of customers reviewed (Using Inner Join)
9. Amazon Word Count

Hive

1. Top 10 Best Products based on average ratings
2. Top 3 Customer based on the number of products
3. Top 3 Popular Product based on number of products sold
4. Number of products sold per day
5. Number of products per rating

Pig

1. Count of Daily Reviews
2. Reviews per Rating

Hadoop

Overall Average Rating of all Products (Basic MapReduce)

I have started the project with basic MapReduce for calculating overall average rating for all products.

It will show how many times the rating has been given to a product and its average rating along with product title.

There are two writable – ProductsWritable (Details of the products) and AverageCountWritable (Average and Count values).

Outputs

hadoop jar 1.OverallAverageRating-1.0-SNAPSHOT.jar com.neu.edu.overallrating.Driver /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part1/result1

```
2020-07-31 04:57:09,500 INFO mapred.LocalJobRunner: Finishing task: attempt_local1269047607_0001_r_000000_0
2020-07-31 04:57:09,500 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-07-31 04:57:10,342 INFO mapreduce.Job:  map 100% reduce 100%
2020-07-31 04:57:10,343 INFO mapreduce.Job: Job job_local1269047607_0001 completed successfully
2020-07-31 04:57:10,365 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=353756877
                FILE: Number of bytes written=1942230297
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=18107456062
                HDFS: Number of bytes written=130851438
                HDFS: Number of read operations=339
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=18
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=4850361
                Map output records=4850360
                Map output bytes=352135804
                Map output materialized bytes=176695081
                Input split bytes=2010
                Combine input records=4850360
                Combine output records=2435166
                Reduce input groups=1227079
                Reduce shuffle bytes=176695081
                Reduce input records=2435166
                Reduce output records=1227079
                Spilled Records=4870332
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=197
                Total committed heap usage (bytes)=6210715648
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=130851438
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part1/result
2020-07-31 06:15:58,619 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-07-31 06:15 /AmazonSports/Part1/result/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup   93654141 2020-07-31 06:15 /AmazonSports/Part1/result/part-r-00000
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -tail /AmazonSports/Part1/result/part-r-00000
2020-07-31 06:16:27,389 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-07-31 06:16:28,023 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
1 Average Rating=5.0
Product = B014GB81AY            Rating =1       Overall Average Rating=5.0
Product = B014GGEKFO            Rating =1       Overall Average Rating=5.0
Product = B014GLZCLU            Rating =1       Overall Average Rating=5.0
Product = B014GNO5JI            Rating =1       Overall Average Rating=5.0
Product = B014H2P150            Rating =1       Overall Average Rating=5.0
Product = B014K11D2S            Rating =1       Overall Average Rating=5.0
Product = B014KW48HY            Rating =1       Overall Average Rating=5.0
Product = B014L9YYGQ            Rating =1       Overall Average Rating=5.0
Product = B014LN3T5Y            Rating =1       Overall Average Rating=5.0
Product = B014PX90WQ            Rating =2       Overall Average Rating=5.0
Product = B014Q2SGSA            Rating =2       Overall Average Rating=5.0
Product = B014RCREPA            Rating =1       Overall Average Rating=5.0
Product = B018D6XMYI            Rating =1       Overall Average Rating=5.0
Product = B01C4N7OQ8            Rating =1       Overall Average Rating=5.0
Product = B01C69JNCS            Rating =1       Overall Average Rating=5.0
Product = B01G316GQG            Rating =1       Overall Average Rating=5.0
Product = B01MTN5XWB            Rating =2       Overall Average Rating=5.0
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
```

Code

RatingMapper

```
package com.neu.edu.overallrating;



import java.io.IOException;

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

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

import org.apache.hadoop.mapreduce.Mapper;

/**

 *

 * @author kaushikpatil

 */

public class RatingMapper extends Mapper <Object, Text, ProductsWritable,
AverageCountWritable> {



    private ProductsWritable productsWritable = new ProductsWritable();

    private AverageCountWritable averageCountWritable = new
AverageCountWritable();



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



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



            if(values[3] != null && values[7] != null &&
!values[0].equals("marketplace")) {

                String productID = values[3];

                float productRating = Float.parseFloat(values[7]);



                productsWritable.setProductsSymbol(productID);

                productsWritable.setratings(productRating);
```

```
                averageCountWritable.setCount(1);

                averageCountWritable.setRatingAvg(productRating);

                context.write(productsWritable, averageCountWritable);

            }

    }

}
```

RatingReducer

```
package com.neu.edu.overallrating;



import java.io.IOException;



import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;



/**

 *

 * @author kaushikpatil

 */

public class RatingReducer extends Reducer <ProductsWritable,
AverageCountWritable, ProductsWritable, AverageCountWritable> {



    private AverageCountWritable res = new AverageCountWritable();



    public void reduce(ProductsWritable key, Iterable <AverageCountWritable>
values, Context context) throws IOException, InterruptedException {
```

```
        float sum = 0;
        int count = 0;

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

        res.setCount(count);
        res.setRatingAvg(sum/count);
        context.write(key, res);
    }
}
```

Driver

```
package com.neu.edu.overallrating;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
/**
 *
 * @author kaushikpatil
 */
public class Driver {
    public static void main(String[] args) throws Exception{

      Path inputPath = new Path(args[0]);
      Path outputDir = new Path(args[1]);

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf);
      job.setJarByClass(RatingMapper.class);
      job.setPartitionerClass(AveragePartitioner.class);

      job.setMapperClass(RatingMapper.class);
      job.setReducerClass(RatingReducer.class);
      job.setCombinerClass(RatingReducer.class);

      job.setNumReduceTasks(1);

      job.setMapOutputKeyClass(ProductsWritable.class);
      job.setMapOutputValueClass(AverageCountWritable.class);
```

```
        job.setOutputKeyClass(ProductsWritable.class);
        job.setOutputValueClass(AverageCountWritable.class);

        FileInputFormat.addInputPath(job, inputPath);
        job.setInputFormatClass(TextInputFormat.class);
        FileOutputFormat.setOutputPath(job, outputDir);

        FileSystem hdfs = FileSystem.get(conf);

              if (hdfs.exists(outputDir)){
                  hdfs.delete(outputDir, true);
              }

              int code = job.waitForCompletion(true) ? 0 : 1;
              System.exit(code);
    }
}
```

ProductsWritable

```
package com.neu.edu.overallrating;

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

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

import org.apache.hadoop.io.WritableComparable;



/**

 *

 * @author kaushikpatil

 */

public class ProductsWritable implements Writable,
WritableComparable<ProductsWritable>{



	private String productID;

	private float ratings;



	public ProductsWritable() {

		super();

	}



	public ProductsWritable(String ProductsSymbol, float ratings) {

		super();



		this.ratings = ratings;

		this.productID = ProductsSymbol;

	}



	public String getProductsSymbol() {
```

```
		return productID;

	}



	public void setProductsSymbol(String ProductsSymbol) {

		this.productID = ProductsSymbol;

	}



	public float getrating() {

		return ratings;

	}



	public void setratings(float ratings) {

		this.ratings = ratings;

	}







	  @Override

	public int hashCode() {

		final int prime = 31;

		int result = 1;

		result = prime * result + ((productID == null) ? 0 :
productID.hashCode());

		result = prime * result + Float.floatToIntBits(ratings);

		return result;
```

```
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		ProductsWritable other = (ProductsWritable) obj;
		if (productID == null) {
			if (other.productID != null)
				return false;
		} else if (!productID.equals(other.productID))
			return false;
		if (Float.floatToIntBits(ratings) !=
Float.floatToIntBits(other.ratings))
			return false;
		return true;
	}

	  public void readFields(DataInput in) throws IOException {
		ratings = in.readInt();
		productID = in.readUTF();
```

```
}

  public void write(DataOutput out) throws IOException {
      out.writeFloat(ratings);
      out.writeUTF(productID);

}

  public int compareTo(ProductsWritable o) {
      int result = -1;
          if(ratings >  o.ratings) {
              result = 1;
          }
          if(ratings == o.ratings) {
              result = 0;
          }
          if(result == 0) {
            result  = productID.compareTo(o.productID);
       }
      return result;
  }

@Override
public String toString() {
```

```
            return "Product = " + productID + "\t";
      }
}
```

AverageCountWritable

```
package com.neu.edu.overallrating;

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

import org.apache.hadoop.io.Writable;

/**
 *
 * @author kaushikpatil
 */
public class AverageCountWritable implements Writable{

    public float RatingAvg;
    public int count;

    public AverageCountWritable (float RatingAvg, int count) {
        this.RatingAvg = RatingAvg;
        this.count = count;
```

```
}

  public AverageCountWritable() {
        super();
    }

  public float getRatingAvg() {
        return RatingAvg;
  }

  public void setRatingAvg(float RatingAvg) {
        this.RatingAvg = RatingAvg;
  }

  public int getCount() {
        return count;
  }

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

```
        public void readFields(DataInput in) throws IOException {
            RatingAvg = in.readFloat();
            count = in.readInt();
      }



      public void write(DataOutput out) throws IOException {
            out.writeFloat(RatingAvg);
            out.writeInt(count);
      }

      @Override
      public String toString() {
            return "Rating =" + count+ "\t" + "Overall Average Rating=" +
RatingAvg;
      }
}
```

AveragePartitioner

```
package com.neu.edu.overallrating;

import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Partitioner;
/**
 *
 * @author kaushikpatil
```

```
 */

public class AveragePartitioner extends Partitioner <ProductsWritable,
NullWritable>{

    @Override

    public int getPartition(ProductsWritable arg0, NullWritable arg1, int
numOfReducerTasks) {

        int hash = arg0.hashCode();

        int partition = hash % numOfReducerTasks;

        return partition;

    }
```

Rating of all Products based on the Year (SecondorySorting Technique)

It's important to see the year wise rating of a particular product to analyze the business of the product.

I have used Secondary Sorting for getting the year wise order for all the products.

There are two writable - CompositeKeyWritable and AverageCountWritable. I have also used comparator and partitioner for comparing two key value pair and segregating for the average rating.

Outputs

hadoop jar 2.RatingPerYear-1.0-SNAPSHOT.jar com.neu.edu.ratingperyear.Driver /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part2/result2

```
2020-08-01 02:49:22,483 INFO mapred.LocalJobRunner: Finishing task: attempt_local210403173_0001_r_000000_0
2020-08-01 02:49:22,484 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-08-01 02:49:23,304 INFO mapreduce.Job:  map 100% reduce 100%
2020-08-01 02:49:23,305 INFO mapreduce.Job: Job job_local210403173_0001 completed successfully
2020-08-01 02:49:23,324 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=291409183
                FILE: Number of bytes written=1601715028
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=18107456062
                HDFS: Number of bytes written=113221746
                HDFS: Number of read operations=339
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=18
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=4850361
                Map output records=4850360
                Map output bytes=135810080
                Map output materialized bytes=145510890
                Input split bytes=2010
                Combine input records=0
                Combine output records=0
                Reduce input groups=1585938
                Reduce shuffle bytes=145510890
                Reduce input records=4850360
                Reduce output records=1585938
                Spilled Records=9700720
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=192
                Total committed heap usage (bytes)=6552551424
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=113221746
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part2/result
2020-08-01 02:52:36,451 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-01 02:49 /AmazonSports/Part2/result/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup  113221746 2020-08-01 02:49 /AmazonSports/Part2/result/part-r-00000
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -tail /AmazonSports/Part2/result/part-r-00000
2020-08-01 02:52:52,981 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-01 02:52:53,608 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
g for Year=5.0
ProductId=B00003G4JR,   Year=1999              Rated=2 ,Average Rating for Year=5.0
ProductId=B00003G1U1,   Year=1999              Rated=1 ,Average Rating for Year=5.0
ProductId=B00003G1TZ,   Year=1999              Rated=1 ,Average Rating for Year=5.0
ProductId=B00001LDC4,   Year=1999              Rated=1 ,Average Rating for Year=3.0
ProductId=B00000K4CC,   Year=1999              Rated=31        ,Average Rating for Year=4.419354838709677
ProductId=B00000K3SK,   Year=1999              Rated=3 ,Average Rating for Year=4.0
ProductId=B00000K2GX,   Year=1999              Rated=5 ,Average Rating for Year=5.0
ProductId=B00000ISXB,   Year=1999              Rated=2 ,Average Rating for Year=5.0
ProductId=B00000IRP9,   Year=1999              Rated=4 ,Average Rating for Year=5.0
ProductId=B00000GBX9,   Year=1999              Rated=1 ,Average Rating for Year=5.0
ProductId=1892524031,   Year=1999              Rated=2 ,Average Rating for Year=3.0
ProductId=0395911737,   Year=1999              Rated=3 ,Average Rating for Year=4.333333333333333
ProductId=0679771611,   Year=1998              Rated=1 ,Average Rating for Year=2.0
ProductId=1570340439,   Year=1997              Rated=1 ,Average Rating for Year=4.0
```

Code

YearMapper

```
package com.neu.edu.ratingperyear;



import java.io.IOException;
```

```
import java.text.SimpleDateFormat;

import java.util.Date;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;



/**

 *

 * @author kaushikpatil

 */

public class YearMapper extends Mapper <Object, Text, CompositeKeyWritable,
AverageCountWritable>{



    private AverageCountWritable averageCountRating = new
AverageCountWritable();



    @Override

        public void map(Object key, Text value, Mapper<Object, Text,
CompositeKeyWritable, AverageCountWritable>.Context context) throws
IOException, InterruptedException {



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

            String productId = null;



                int year = 0;

                double rating = 0.0D;
```

```
                if(values[5] != null && values[7] != null &&
!values[0].equals("marketplace")) {

                    try {

                        productId = values[3];

                        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd");

                        Date reviewDate = sdf.parse(values[14]);

                        year = Integer.parseInt(reviewDate.toString().split("
")[5]);

                        rating = Double.parseDouble(values[7]);



                        }catch(Exception e) {

                            e.printStackTrace();

                    }



                if(year != 0 && productId != null) {



                    averageCountRating.setCount(1);

                    averageCountRating.setRatingAvg(rating);

                    CompositeKeyWritable ck = new CompositeKeyWritable(year,
productId);



                    try {

                            context.write(ck, averageCountRating);

                    }catch (Exception e) {

                            e.printStackTrace();
```

```
                    }
                }
                }
      }
}
```

YearReducer

```
package com.neu.edu.ratingperyear;


import java.io.IOException;


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


/**
 *
 * @author kaushikpatil
 */
public class YearReducer extends Reducer <CompositeKeyWritable,
AverageCountWritable, CompositeKeyWritable, AverageCountWritable> {


    private AverageCountWritable res = new AverageCountWritable();


    @Override
        public void reduce(CompositeKeyWritable arg0,
Iterable<AverageCountWritable> arg1, Reducer<CompositeKeyWritable,
```

```
            AverageCountWritable, CompositeKeyWritable,
AverageCountWritable>.Context arg2) throws IOException, InterruptedException
{



            double sum = 0;

            int count = 0;



                for(AverageCountWritable val: arg1) {

                    sum += val.getCount() * val.getRatingAvg();

                    count += val.getCount();

                }



                res.setCount(count);

                res.setRatingAvg(sum/count);

                arg2.write(arg0, res);

        }

}
```

Driver

```
package com.neu.edu.ratingperyear;




import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 *
 * @author kaushikpatil
 */
public class Driver {
    public static void main(String[] args) throws Exception{

        Path inputPath = new Path(args[0]);
        Path outputDir = new Path(args[1]);

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(YearMapper.class);

        job.setPartitionerClass(YearPartitioner.class);
        job.setGroupingComparatorClass(YearComparator.class);
        job.setSortComparatorClass(SecSortingComparator.class);

        job.setMapperClass(YearMapper.class);
        job.setReducerClass(YearReducer.class);
```

```
        job.setNumReduceTasks(1);

        job.setMapOutputKeyClass(CompositeKeyWritable.class);
        job.setMapOutputValueClass(AverageCountWritable.class);

        job.setOutputKeyClass(CompositeKeyWritable.class);
        job.setOutputValueClass(AverageCountWritable.class);

        FileInputFormat.addInputPath(job, inputPath);
        job.setInputFormatClass(TextInputFormat.class);
        FileOutputFormat.setOutputPath(job, outputDir);

        FileSystem hdfs = FileSystem.get(conf);

        if (hdfs.exists(outputDir)){
            hdfs.delete(outputDir, true);
        }

        int code = job.waitForCompletion(true) ? 0 : 1;
            System.exit(code);
    }
}
```

CompositeKeyWritable

```
package com.neu.edu.ratingperyear;
```

```
import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;



import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;



/**

 *

 * @author kaushikpatil

 */

public class CompositeKeyWritable implements Writable,
WritableComparable<CompositeKeyWritable>{



    private String productId;

    private int year;



    public CompositeKeyWritable() {

        super();

    }



    public CompositeKeyWritable(int year, String productId) {

      super();

      this.year = year;
```

```
  this.productId = productId;
}

public String getProductId() {
  return productId;
}

public void setProductId(String productId) {
  this.productId = productId;
}

public int getYear() {
    return year;
}

public void setYear(int year) {
    this.year = year;
}

public void readFields(DataInput in) throws IOException {
    productId = in.readUTF();
    year = in.readInt();
}
```

```
public void write(DataOutput out) throws IOException {
    out.writeUTF(productId);
    out.writeInt(year);
}

public int compareTo(CompositeKeyWritable o) {

    int result = -1;

    if(year > o.year) {
        result = 1;
    }

    else if (year == o.year) {
        result = 0;
    }

    else {
        result = -1;
    }

    if(result == 0) {
        result = productId.compareTo(o.productId);
    }
```

```
            return result;
    }

    @Override
      public String toString() {
            return "ProductId=" + productId + ",\t"+ "Year=" + year + "\t";
        }
}
```

AverageCountWritable

```
package com.neu.edu.ratingperyear;

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

import org.apache.hadoop.io.Writable;

/**
 *
 * @author kaushikpatil
 */
public class AverageCountWritable implements Writable{

      public double RatingAvg;
```

```
  public int count;

  public AverageCountWritable(double RatingAvg, int count) {
            super();
            this.RatingAvg = RatingAvg;
            this.count = count;
  }

  public AverageCountWritable() {
        super();
  }

public double getRatingAvg() {
  return RatingAvg;
}

public void setRatingAvg(double RatingAvg) {
  this.RatingAvg = RatingAvg;
}

public int getCount() {
  return count;
}
```

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

    public void readFields(DataInput in) throws IOException {
        RatingAvg = in.readDouble();
        count = in.readInt();
    }

    public void write(DataOutput out) throws IOException {
        out.writeDouble(RatingAvg);
        out.writeInt(count);
    }

    @Override
    public String toString() {
      return "Rated=" + count+  "\t"+",Average Rating for Year=" + RatingAvg;
    }
}
```

YearPartitioner

```
package com.neu.edu.ratingperyear;

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

```
import org.apache.hadoop.mapreduce.Partitioner;



/**

 *

 * @author kaushikpatil

 */

public class YearPartitioner extends Partitioner <CompositeKeyWritable,
NullWritable>{



    @Override

      public int getPartition(CompositeKeyWritable arg0, NullWritable arg1,
int numOfReducerTasks) {

            int hash = arg0.getProductId().hashCode();

            int partition = hash % numOfReducerTasks;

            return partition;

        }

}
```

YearComparator

```
package com.neu.edu.ratingperyear;



import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;



/**

 *
```

```
 * @author kaushikpatil
 */
public class YearComparator extends WritableComparator{

    protected YearComparator() {
        super(CompositeKeyWritable.class, true);
    }

    @Override
      public int compare(WritableComparable a, WritableComparable b) {

            CompositeKeyWritable ck1 = (CompositeKeyWritable) a;
            CompositeKeyWritable ck2 = (CompositeKeyWritable) b;
            return ck1.getProductId().compareTo(ck2.getProductId());
        }
}
```

SecSortingComparator

```
package com.neu.edu.ratingperyear;

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

/**
 *
```

```
 * @author kaushikpatil
 */
public class SecSortingComparator extends WritableComparator{

	protected SecSortingComparator() {
		super(CompositeKeyWritable.class, true);
	}

	@Override
	public int compare(WritableComparable a, WritableComparable b) {

		CompositeKeyWritable ck1 = (CompositeKeyWritable) a;
		CompositeKeyWritable ck2 = (CompositeKeyWritable) b;

		int result = 0;

		if(ck1.getYear() > ck2.getYear()) {
		    result = 1;
		}

		else if (ck1.getYear() == ck2.getYear()) {
		    result = 0;
		}
```

```
            else {
                result = -1;
            }

            if(result == 0) {
                result = ck1.getProductId().compareTo(ck2.getProductId());
            }
      return result*(-1);
        }
}
```

Number of Products per Rating (Counter using Numerical Summarization Pattern)

How can we know which products are good or bad? Which products are liked by customers?

So, we can determine this by knowing the ratings of the products and then dividing them on a scale of 5.

I have used counter for the ratings to count the number of products per rating using Numerical Summarization Pattern. We have a class called Rating and a Mapper.

Outputs

hadoop jar 3.ProdPerRating-1.0-SNAPSHOT.jar com.neu.edu.prodperrating.Rating /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part3/result3

```
2020-08-04 13:10:13,137 INFO mapred.LocalJobRunner: Finishing task: attempt_local632964561_0001_r_000000_0
2020-08-04 13:10:13,137 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-08-04 13:10:13,272 INFO mapreduce.Job:  map 100% reduce 100%
2020-08-04 13:10:13,273 INFO mapreduce.Job: Job job_local632964561_0001 completed successfully
2020-08-04 13:10:13,297 INFO mapreduce.Job: Counters: 41
        File System Counters
                FILE: Number of bytes read=307775
                FILE: Number of bytes written=8557200
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=18107456062
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=323
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=34
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=4850361
                Map output records=0
                Map output bytes=0
                Map output materialized bytes=90
                Input split bytes=2010
                Combine input records=0
                Combine output records=0
                Reduce input groups=0
                Reduce shuffle bytes=90
                Reduce input records=0
                Reduce output records=0
                Spilled Records=0
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=100
                Total committed heap usage (bytes)=6068109312
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        State
                1=362984
                2=229384
                3=380499
                4=836773
                5=3040720
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=0
```

```
Products with Rating 1 = 362984 products.
Products with Rating 2 = 229384 products.
Products with Rating 3 = 380499 products.
Products with Rating 4 = 836773 products.
Products with Rating 5 = 3040720 products.
(base) Kaushiks-MBP:documents kaushikpatil$
```

Code

Rating

```
package com.neu.edu.prodperrating;


import java.io.IOException;
```

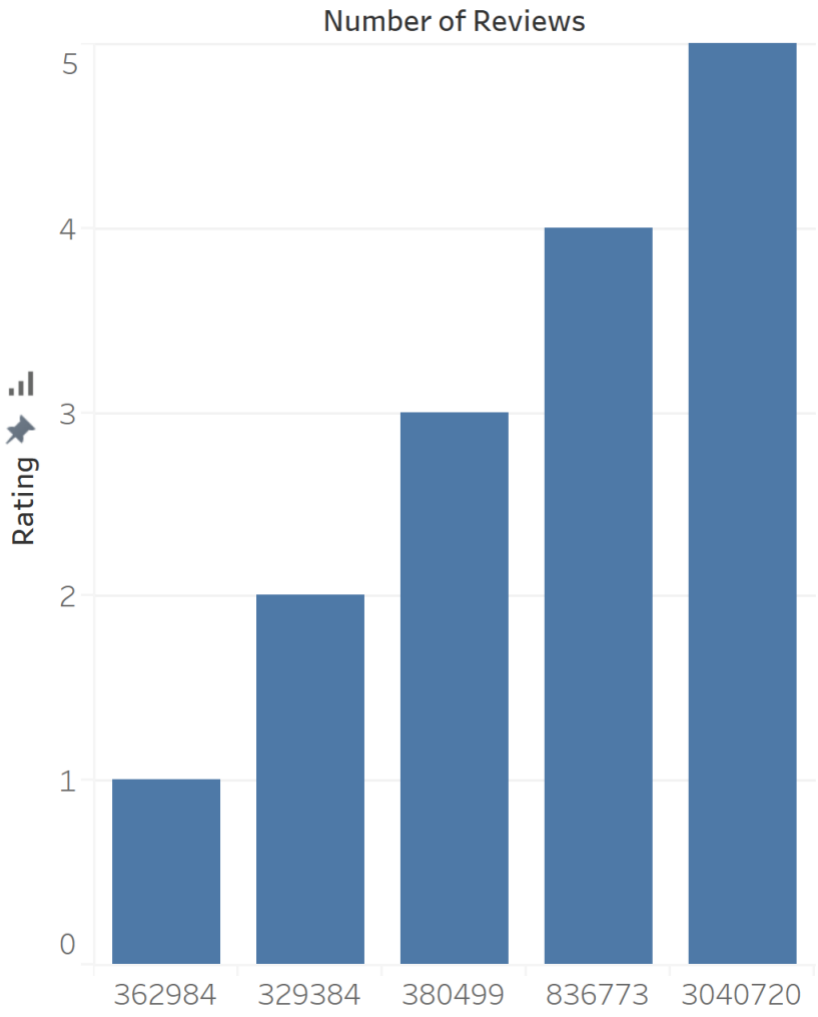
```
import java.util.Arrays;

import java.util.HashSet;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Counter;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


/**

 *

 * @author kaushikpatil

 */

public class Rating {


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


        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Products per Rating");
```

```
        job.setJarByClass(Rating.class);

        job.setMapperClass(ProdMapper.class);

        job.setMapOutputKeyClass(Text.class);

        job.setMapOutputValueClass(Text.class);



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

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



        Path out = new Path(args[1]);

        FileSystem.get(conf).delete(out, true);



        int code = job.waitForCompletion(true) ? 0 : 1;



       if(code==0){

            for(Counter counter :
job.getCounters().getGroup(ProdMapper.STATE_COUNTER_GROUP)){

                {

                     System.out.println("Products with Rating
"+counter.getDisplayName()+" = "+counter.getValue()+" products.");

                }

            }

        }

        System.exit(code);

    }
```

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



        public static final String STATE_COUNTER_GROUP = "State";

        public static final String UNKNOWN_COUNTER = "Unknown";

        public static final String NULL_OR_EMPTY_COUNTER = "Null or
empty";



        private String[] statesArray = new String[] {"1", "2", "3", "4",
"5"};



        private HashSet<String> states = new
HashSet<String>(Arrays.asList(statesArray));



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



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

                return;

            }



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

            String stateName = token[7].trim();



            if(stateName != null && !stateName.isEmpty()){

                String[] stateTokens =
stateName.toUpperCase().split("\\s");
```

```
                boolean unknown = true;

                for(String state: stateTokens){

                    if(state.contains(state)){



                        context.getCounter(STATE_COUNTER_GROUP,
state).increment(1);

                        unknown = false;

                        break;

                    }

                }



            if(unknown){

                context.getCounter(STATE_COUNTER_GROUP,
UNKNOWN_COUNTER).increment(1);

            }

            } else{

                context.getCounter(STATE_COUNTER_GROUP,
NULL_OR_EMPTY_COUNTER).increment(1);

            }

        }

    }

}
```

Data Visualization



Segregation of all Products based on their rating (Binning Organization Pattern)

We have segregated all the products and their details based on the rating into small bins which consists of products of that specific reviews and ratings only

This is achieved by Binning Organization Pattern. We have used Driver and Mapper as this method doesn't require any reducer, partitioner etc.

Outputs

hadoop jar 4.RatingBinning-1.0-SNAPSHOT.jar com.neu.edu.ratingbinning.RatingBinning /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part4/result4

```
2020-08-04 14:45:41,968 INFO mapred.LocalJobRunner: Finishing task: attempt_local418592101_0001_m_000014_0
2020-08-04 14:45:41,969 INFO mapred.LocalJobRunner: map task executor complete.
2020-08-04 14:45:42,658 INFO mapreduce.Job: Job job_local418592101_0001 completed successfully
2020-08-04 14:45:42,678 INFO mapreduce.Job: Counters: 26
        File System Counters
                FILE: Number of bytes read=274570
                FILE: Number of bytes written=8039235
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=16100373791
                HDFS: Number of bytes written=16099886391
                HDFS: Number of read operations=1365
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=1455
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=4850361
                Map output records=0
                Input split bytes=2010
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=220
                Total committed heap usage (bytes)=1557135360
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=0
        org.apache.hadoop.mapreduce.lib.output.MultipleOutputs
                Rating_1=362984
                Rating_2=229384
                Rating_3=380499
                Rating_4=836773
                Rating_5=3040720
```

```
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part4/result
2020-08-04 14:57:08,559 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 91 items
-rw-r--r--   1 kaushikpatil supergroup   10793055 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00000
-rw-r--r--   1 kaushikpatil supergroup   10749034 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00001
-rw-r--r--   1 kaushikpatil supergroup   10011329 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00002
-rw-r--r--   1 kaushikpatil supergroup    9799250 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00003
-rw-r--r--   1 kaushikpatil supergroup   10407177 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00004
-rw-r--r--   1 kaushikpatil supergroup   10400363 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00005
-rw-r--r--   1 kaushikpatil supergroup   11157894 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00006
-rw-r--r--   1 kaushikpatil supergroup   10932486 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00007
-rw-r--r--   1 kaushikpatil supergroup   11154610 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00008
-rw-r--r--   1 kaushikpatil supergroup   10352877 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00009
-rw-r--r--   1 kaushikpatil supergroup    9765818 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00010
-rw-r--r--   1 kaushikpatil supergroup   10910610 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00011
-rw-r--r--   1 kaushikpatil supergroup   13256096 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00012
-rw-r--r--   1 kaushikpatil supergroup   12375406 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00013
-rw-r--r--   1 kaushikpatil supergroup   13299972 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_1-m-00014
-rw-r--r--   1 kaushikpatil supergroup    6904285 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00000
-rw-r--r--   1 kaushikpatil supergroup    6954935 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00001
-rw-r--r--   1 kaushikpatil supergroup    6799085 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00002
-rw-r--r--   1 kaushikpatil supergroup    6693439 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00003
-rw-r--r--   1 kaushikpatil supergroup    7194979 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00004
-rw-r--r--   1 kaushikpatil supergroup    7122608 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00005
-rw-r--r--   1 kaushikpatil supergroup    7501319 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00006
-rw-r--r--   1 kaushikpatil supergroup    7329004 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00007
-rw-r--r--   1 kaushikpatil supergroup    7722948 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00008
-rw-r--r--   1 kaushikpatil supergroup    7683127 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00009
-rw-r--r--   1 kaushikpatil supergroup    7075762 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00010
-rw-r--r--   1 kaushikpatil supergroup    7528085 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00011
-rw-r--r--   1 kaushikpatil supergroup    8463804 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00012
-rw-r--r--   1 kaushikpatil supergroup    8198742 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00013
-rw-r--r--   1 kaushikpatil supergroup    7985350 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_2-m-00014
-rw-r--r--   1 kaushikpatil supergroup   10946411 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00000
-rw-r--r--   1 kaushikpatil supergroup   11055051 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00001
-rw-r--r--   1 kaushikpatil supergroup   11170772 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00002
-rw-r--r--   1 kaushikpatil supergroup   11286773 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00003
-rw-r--r--   1 kaushikpatil supergroup   11819080 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00004
-rw-r--r--   1 kaushikpatil supergroup   11849521 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00005
-rw-r--r--   1 kaushikpatil supergroup   12033187 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00006
-rw-r--r--   1 kaushikpatil supergroup   12223592 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00007
-rw-r--r--   1 kaushikpatil supergroup   12817639 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00008
-rw-r--r--   1 kaushikpatil supergroup   12621177 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00009
-rw-r--r--   1 kaushikpatil supergroup   12335226 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00010
-rw-r--r--   1 kaushikpatil supergroup   12596755 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00011
-rw-r--r--   1 kaushikpatil supergroup   12384707 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00012
-rw-r--r--   1 kaushikpatil supergroup   12977796 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00013
-rw-r--r--   1 kaushikpatil supergroup   12012161 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_3-m-00014
-rw-r--r--   1 kaushikpatil supergroup   22434601 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00000
-rw-r--r--   1 kaushikpatil supergroup   22864138 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00001
-rw-r--r--   1 kaushikpatil supergroup   23041638 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00002
-rw-r--r--   1 kaushikpatil supergroup   23229984 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00003
-rw-r--r--   1 kaushikpatil supergroup   23792458 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00004
-rw-r--r--   1 kaushikpatil supergroup   24405521 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00005
-rw-r--r--   1 kaushikpatil supergroup   24908681 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00006
-rw-r--r--   1 kaushikpatil supergroup   25929343 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00007
-rw-r--r--   1 kaushikpatil supergroup   26786375 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00008
-rw-r--r--   1 kaushikpatil supergroup   27914186 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00009
-rw-r--r--   1 kaushikpatil supergroup   27780516 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00010
-rw-r--r--   1 kaushikpatil supergroup   27225889 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00011
-rw-r--r--   1 kaushikpatil supergroup   27802617 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00012
-rw-r--r--   1 kaushikpatil supergroup   30103000 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00013
-rw-r--r--   1 kaushikpatil supergroup   29900025 2020-08-04 14:45 /AmazonSports/Part4/result/Rating_4-m-00014
```

```
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -head /AmazonSports/Part4/result/Rating_5-m-00002
2020-08-04 16:16:58,271 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-04 16:16:59,038 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
US      33190470        R34GHM7VLNEMBU  B00HBIVNDC      713005168       Thrive on Wellness Thick Exercise Mat with Carry Strap - BEST Comfort on Hips, Knees, Spine and Joints, 72" x 24" x 1/2" EXTRA Long
Yoga Mats for P90X, Pilates, Yoga, Strength and Stretch Workouts        Sports  5       0       1       N       Y       Five Stars      wife uses  and loves it 2015-04-26
US      2144895 RPLJZ6WC2C387   B00FNRC9WM      467658862       IZZO Swami Watch Golf GPS (New and Improved)    Sports  5       0       0       N       Y       Five Stars      Excellent       2015-04-26
US      37578680        R3L915KQ7YOMXO  B00755KZB2      415674961       Grambling State University - Teardrop Keychain - Gold   Sports  5       0       0       N       Y       Five Stars      Bought as a
gift and they loved it! 2015-04-26
US      19331207        RXJHBTJSXLV7Y   B003QORJFQ      750204989       Adidas Men's ClimaLite 3 Stripes Cuff Polo Shirt, XXX-Large, UNIVERSITY RED/WHITE       Sports  5       0       0       N       Y  F
ive Stars       Nice fit, and I love the cool feel. Would buy another.  2015-04-26
US      7646445 R2FTMGJ5SI4ZDS  B00ARBF05C      462891002       uxcell Men Cycling Sports Protecting Blue Elastic Neoprene Single Shoulder Brace Support       Sports  5       0       0       N       Y  I
t worked very well. My(base) Kaushiks-MBP:documents kaushikpatil$
```

Code

RatingBinner

```
package com.neu.edu.ratingbinning;


import java.io.IOException;
```

```
import org.apache.hadoop.conf.Configuration;

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.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


/**

 *

 * @author kaushikpatil

 */

public class RatingBinning {


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


        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Rating Binning");


        job.setJarByClass(RatingBinning.class);
```

```
        job.setMapperClass(RatingBinningMapper.class);

        job.setMapOutputKeyClass(Text.class);

        job.setMapOutputValueClass(NullWritable.class);



        job.setNumReduceTasks(0);



        TextInputFormat.addInputPath(job, new Path(args[0]));

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



        MultipleOutputs.addNamedOutput(job, "bins", TextOutputFormat.class,
Text.class, NullWritable.class);

        MultipleOutputs.setCountersEnabled(job, true);



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

    }



public static class RatingBinningMapper extends Mapper<LongWritable, Text,
Text, NullWritable> {



    private MultipleOutputs<Text, NullWritable> mos = null;



    protected void setup(Context context) {

        mos = new MultipleOutputs(context);

    }
```

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



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

        return;

        }



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

    {

        String rating = token[7].trim();



        if (rating.equals("1")) {

            mos.write("bins", value, NullWritable.get(), "Rating_1");

        }



        if (rating.equals("2")) {

            mos.write("bins", value, NullWritable.get(), "Rating_2");

        }



        if (rating.equals("3")) {

            mos.write("bins", value, NullWritable.get(), "Rating_3");

        }



        if (rating.equals("4")) {

            mos.write("bins", value, NullWritable.get(), "Rating_4");
```

```
        }



        if (rating.equals("5")) {

            mos.write("bins", value, NullWritable.get(), "Rating_5");

        }

    }

    }



    protected void cleanup(Context context) throws IOException,
InterruptedException {

    mos.close();

    }

}

}
```

Top 25 Best Products based on average ratings (Filtering Pattern)

We should know the products which are doing well for customer satisfaction for keeping the track of demand and supply.

We can use filtering pattern for getting the top 25 Best products based on their average ratings from the dataset.

Here we will use two mappers, two reducers, comparator and driver. The first map reduce will get the average of all reviews for all products. The second map reduce will get only the top 25 best products using the counter.

Outputs

hadoop jar 5.Top25Products-1.0-SNAPSHOT.jar com.neu.edu.topproducts.Driver /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part5/result1 /AmazonSports/Part5/result2

```
2020-08-04 18:44:20,495 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local137776519_0001_m_000007_0 decomp: 5119093 len: 5119097 to MEMORY
2020-08-04 18:44:20,498 INFO reduce.InMemoryMapOutput: Read 5119093 bytes from map-output for attempt_local137776519_0001_m_000007_0
2020-08-04 18:44:20,498 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 5119093, inMemoryMapOutputs.size() -> 13, commitMemory -> 67084846, usedMemory ->72203939
2020-08-04 18:44:20,500 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local137776519_0001_m_000001_0 decomp: 6808893 len: 6808897 to MEMORY
2020-08-04 18:44:20,504 INFO reduce.InMemoryMapOutput: Read 6808893 bytes from map-output for attempt_local137776519_0001_m_000001_0
2020-08-04 18:44:20,504 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 6808893, inMemoryMapOutputs.size() -> 14, commitMemory -> 72203939, usedMemory ->79012832
2020-08-04 18:44:20,507 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local137776519_0001_m_000014_0 decomp: 3443318 len: 3443322 to MEMORY
2020-08-04 18:44:20,508 INFO reduce.InMemoryMapOutput: Read 3443318 bytes from map-output for attempt_local137776519_0001_m_000014_0
2020-08-04 18:44:20,508 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 3443318, inMemoryMapOutputs.size() -> 15, commitMemory -> 79012832, usedMemory ->82456150
2020-08-04 18:44:20,508 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
2020-08-04 18:44:20,509 INFO mapred.LocalJobRunner: 15 / 15 copied.
2020-08-04 18:44:20,509 INFO reduce.MergeManagerImpl: finalMerge called with 15 in-memory map-outputs and 0 on-disk map-outputs
2020-08-04 18:44:20,521 INFO mapred.Merger: Merging 15 sorted segments
2020-08-04 18:44:20,521 INFO mapred.Merger: Down to the last merge-pass, with 15 segments left of total size: 82455955 bytes
2020-08-04 18:44:22,072 INFO reduce.MergeManagerImpl: Merged 15 segments, 82456150 bytes to disk to satisfy reduce memory limit
2020-08-04 18:44:22,072 INFO reduce.MergeManagerImpl: Merging 1 files, 82456126 bytes from disk
2020-08-04 18:44:22,073 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
2020-08-04 18:44:22,073 INFO mapred.Merger: Merging 1 sorted segments
2020-08-04 18:44:22,074 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 82456109 bytes
2020-08-04 18:44:22,074 INFO mapred.LocalJobRunner: 15 / 15 copied.
2020-08-04 18:44:22,266 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2020-08-04 18:44:22,375 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-08-04 18:44:24,230 INFO mapred.Task: Task:attempt_local137776519_0001_r_000000_0 is done. And is in the process of committing
2020-08-04 18:44:24,233 INFO mapred.LocalJobRunner: 15 / 15 copied.
2020-08-04 18:44:24,233 INFO mapred.Task: Task attempt_local137776519_0001_r_000000_0 is allowed to commit now
2020-08-04 18:44:24,270 INFO output.FileOutputCommitter: Saved output of task 'attempt_local137776519_0001_r_000000_0' to hdfs://localhost:9000/AmazonSports/Part5/result1
2020-08-04 18:44:24,271 INFO mapred.LocalJobRunner: reduce > reduce
2020-08-04 18:44:24,271 INFO mapred.Task: Task 'attempt_local137776519_0001_r_000000_0' done.
2020-08-04 18:44:24,272 INFO mapred.Task: Final Counters for attempt_local137776519_0001_r_000000_0: Counters: 30
        File System Counters
                FILE: Number of bytes read=164941587
                FILE: Number of bytes written=165451714
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2007082271
                HDFS: Number of bytes written=16459643
                HDFS: Number of read operations=38
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=3
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Combine input records=0
                Combine output records=0
                Reduce input groups=1046150
                Reduce shuffle bytes=82456210
                Reduce input records=4850360
                Reduce output records=1046150
                Spilled Records=4850360
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=5
                Total committed heap usage (bytes)=238026752
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Output Format Counters
                Bytes Written=16459643
```

```
2020-08-04 18:44:27,443 INFO mapred.LocalJobRunner: Finishing task: attempt_local1211668060_0002_r_000000_0
2020-08-04 18:44:27,443 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-08-04 18:44:27,996 INFO mapreduce.Job:  map 100% reduce 100%
2020-08-04 18:44:27,997 INFO mapreduce.Job: Job job_local1211668060_0002 completed successfully
2020-08-04 18:44:28,002 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=365469324
                FILE: Number of bytes written=385334924
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=4047083828
                HDFS: Number of bytes written=32919661
                HDFS: Number of read operations=91
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=14
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=1046150
                Map output records=1046150
                Map output bytes=15692250
                Map output materialized bytes=17784556
                Input split bytes=126
                Combine input records=0
                Combine output records=0
                Reduce input groups=7633
                Reduce shuffle bytes=17784556
                Reduce input records=1046150
                Reduce output records=25
                Spilled Records=2092300
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=26
                Total committed heap usage (bytes)=432013312
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=16459643
        File Output Format Counters
                Bytes Written=375
```

```
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part5
2020-08-04 18:44:35,435 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x   - kaushikpatil supergroup          0 2020-08-04 18:44 /AmazonSports/Part5/result1
drwxr-xr-x   - kaushikpatil supergroup          0 2020-08-04 18:44 /AmazonSports/Part5/result2
(base) Kaushiks-MBP:documents kaushikpatil$
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part5/result1
2020-08-04 18:44:46,691 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-04 18:44 /AmazonSports/Part5/result1/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup   16459643 2020-08-04 18:44 /AmazonSports/Part5/result1/part-r-00000
(base) Kaushiks-MBP:documents kaushikpatil$
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part5/result2
2020-08-04 18:44:55,119 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-04 18:44 /AmazonSports/Part5/result2/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup        375 2020-08-04 18:44 /AmazonSports/Part5/result2/part-r-00000
```

```
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -tail /AmazonSports/Part5/result1/part-r-00000
2020-08-04 18:45:20,105 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-04 18:45:20,732 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
G       5.0
B01419ZX5C      5.0
B0141BG62E      4.6666665
B0141GN0A0      4.0
B0141GSGJ0      5.0
B0141H4IO6      5.0
B0141N8DI2      1.0
B0141T37MI      5.0
B014269N6O      5.0
B01429RYCQ      5.0
B0142B5SZY      5.0
B0143IVO2M      5.0
B0143J860Y      5.0
B0143JKI8M      5.0
B0143LCFTK      3.0
B0143MIRK0      5.0
B0143RAPR8      5.0
B0143RD3OK      5.0
B01444LKRO      5.0
B01444SHRK      5.0
B014455IE4      5.0
B0145QM6Q0      5.0
B0145QONLG      5.0
B0145XJ43G      5.0
B0145YBOW4      5.0
B0145YW7NY      5.0
B01460ASM4      5.0
B0146179L6      5.0
B01461KRX8      5.0
B014620O6W      5.0
B01462HANM      5.0
B014639E8A      5.0
B0146C85N6      5.0
B0147L5TDU      5.0
B0147LHI8Y      1.0
B0147M421Y      5.0
B0147Q0UM0      4.75
B01482J7D6      5.0
B014AYH8PQ      5.0
B014B94BEQ      5.0
B014BBEPD6      5.0
B014C3P7LM      5.0
B014CGWRJY      5.0
B014CUCIOO      5.0
B014EJBPJC      5.0
B014FVKH8Y      5.0
B014G4BSBK      5.0
B014G7RG9K      5.0
B014GB81AY      5.0
B014GGEKFO      5.0
B014GLZCLU      5.0
B014GNO5JI      5.0
B014GUHTEE      1.0
B014H2P150      5.0
B014JG97LS      1.0
B014K11D2S      5.0
B014KW48HY      5.0
B014L9YYGQ      5.0
B014LN3T5Y      5.0
B014PX90WQ      5.0
```

```
(base) Kaushiks-MBP:documents kaushikpatil$ hadoop fs -tail /AmazonSports/Part5/result2/part-r-00000
2020-08-04 18:45:38,749 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-04 18:45:39,367 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
5.0     B004WWWV0E
5.0     B004WWWUYG
5.0     B004VQXWU4
5.0     B004WWTWCO
5.0     B004VQXX4Y
5.0     B004VQYGJU
5.0     B004VQYGQ8
5.0     B004VQZ3EW
5.0     B004VQZ3OW
5.0     B004WWTSO6
5.0     B004VR04JU
5.0     B004WWTSGO
5.0     B004VR0NSW
5.0     B004VRAQ48
5.0     B004WWTP6W
5.0     B004WWR9P6
5.0     B004WWP2ZA
5.0     B004WWP1YC
5.0     B004VRAQQQ
5.0     B004WWN0QI
5.0     B004VRAS5K
5.0     B004WWL8CQ
5.0     B004VRASGY
5.0     B004WWL5D8
5.0     B004VRB7KA
```

Code

TopMapper1

```
package com.neu.edu.topproducts;



import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;



/**

 *

 * @author kaushikpatil

 */

public class TopMapper1 extends Mapper<LongWritable, Text, Text,
FloatWritable> {



    private Text text = new Text();

    private FloatWritable score = new FloatWritable();



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



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

            return;
```

```
        }


        else{


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

            String productid = line[3].trim();

            float rating = Float.valueOf(line[7].trim());

            text.set(productid);

            score.set(rating);

            context.write(text,score);

        }

    }

}
```

TopMapper2

```
package com.neu.edu.topproducts;


import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/**

 *

 * @author kaushikpatil

 */
```

```
public class TopMapper2 extends Mapper<LongWritable, Text, FloatWritable,
Text> {



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



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

        String productid = row[0].trim();

        float prodrating = Float.parseFloat(row[1].trim());

        try {

            Text id = new Text(productid);

            FloatWritable prod = new FloatWritable(prodrating);

            context.write(prod, id); }

            catch (Exception e) {

            }

    }

}
```

TopReducer1

```
package com.neu.edu.topproducts;



import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;



/**
```

```
 *

 * @author kaushikpatil

 */

public class TopReducer1 extends Reducer<Text, FloatWritable, Text,
FloatWritable>{



    private FloatWritable result = new FloatWritable();



    @Override

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



        float sum = 0;

        int count = 0;



        for(FloatWritable val:values){

            sum+=val.get();

            count = count+1;

        }



        float average = sum/count;

        result.set(average);

        context.write(key,result);

    }

}
```

TopReducer2

```
package com.neu.edu.topproducts;



import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;



/**

 *

 * @author kaushikpatil

 */

public class TopReducer2 extends Reducer<FloatWritable, Text, FloatWritable,
Text> {



    int count = 0;



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



        for (Text val : value) {

            if (count < 25) {

                context.write(key, val);

            }

            count++;
```

```
        }
    }
}
```

TopComparator

```
package com.neu.edu.topproducts;


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


/**
 *
 * @author kaushikpatil
 */
public class TopComparator extends WritableComparator {


    protected TopComparator() {
        super(FloatWritable.class, true);
    }


    public int compare(WritableComparable f1, WritableComparable f2) {


        FloatWritable fw1 = (FloatWritable) f1;
        FloatWritable fw2 = (FloatWritable) f2;
```

```
        int result = fw1.get() < fw2.get() ? 1 : fw1.get() == fw2.get() ? 0 :
-1;

        return result;

    }

}
```

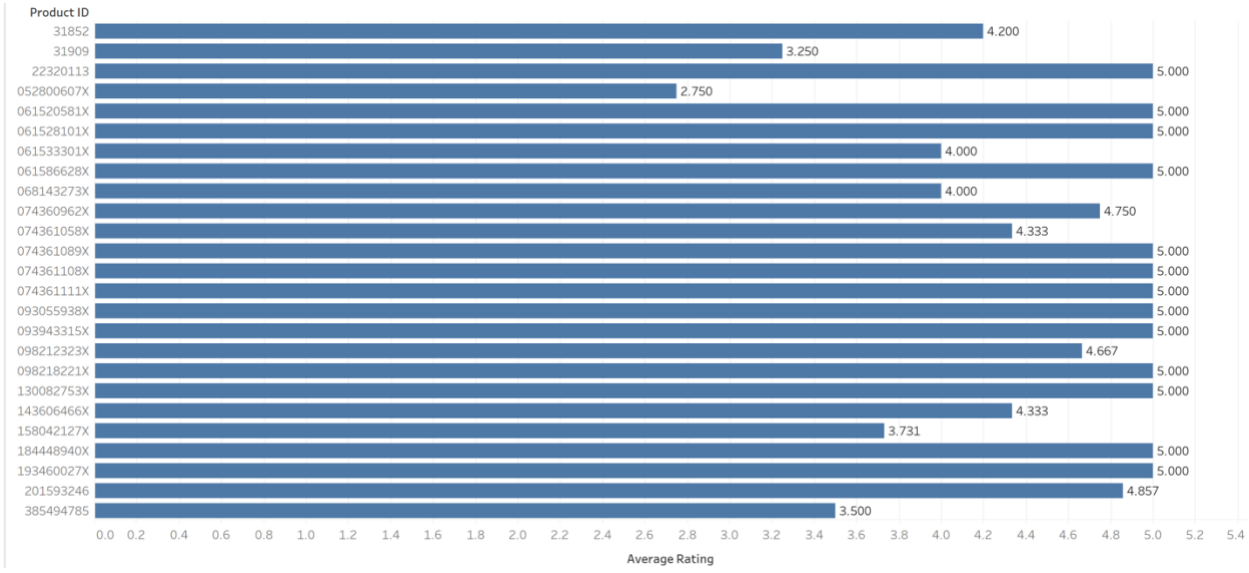
Driver

```
package com.neu.edu.topproducts;



import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;



/**

 *

 * @author kaushikpatil

 */

public class Driver {



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

```
Configuration conf1 = new Configuration();
Job job1 = Job.getInstance(conf1, "Best 25 Products");

job1.setJarByClass(Driver.class);
job1.setMapperClass(TopMapper1.class);
job1.setMapOutputKeyClass(Text.class);
job1.setMapOutputValueClass(FloatWritable.class);
job1.setReducerClass(TopReducer1.class);
job1.setOutputKeyClass(Text.class);
job1.setOutputValueClass(FloatWritable.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));
FileOutputFormat.setOutputPath(job1, new Path(args[1]));

boolean complete = job1.waitForCompletion(true);
Configuration conf2 = new Configuration();

if (complete) {

    Job job2 = Job.getInstance(conf2, "Best 25 Products");
    job2.setJarByClass(Driver.class);
    job2.setMapperClass(TopMapper2.class);
    job2.setMapOutputKeyClass(FloatWritable.class);
```

```
            job2.setMapOutputValueClass(Text.class);

            job2.setSortComparatorClass(TopComparator.class);

            job2.setReducerClass(TopReducer2.class);

            job2.setOutputKeyClass(FloatWritable.class);

            job2.setOutputValueClass(Text.class);


            FileInputFormat.addInputPath(job2, new Path(args[1]));

            FileOutputFormat.setOutputPath(job2, new Path(args[2]));


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

        }

    }

}
```

Data Visualization



Top 25 Products..	
B004VQXWU4	5.000
B004VQXX4Y	5.000
B004VQYGJU	5.000
B004VQYGQ8	5.000
B004VQZ3EW	5.000
B004VQZ3OW	5.000
B004VR0NSW	5.000
B004VR04JU	5.000
B004VRAQ48	5.000
B004VRAQQQ	5.000
B004VRAS5K	5.000
B004VRASGY	5.000
B004VRB7KA	5.000
B004WWL5D8	5.000
B004WWL8CQ	5.000
B004WWN0QI	5.000
B004WWP1YC	5.000
B004WWP2ZA	5.000
B004WWR9P6	5.000
B004WWTP6W	5.000
B004WWTSGO	5.000
B004WWTSO6	5.000
B004WWTWCO	5.000
B004WWWUYG	5.000
B004WWWV0E	5.000

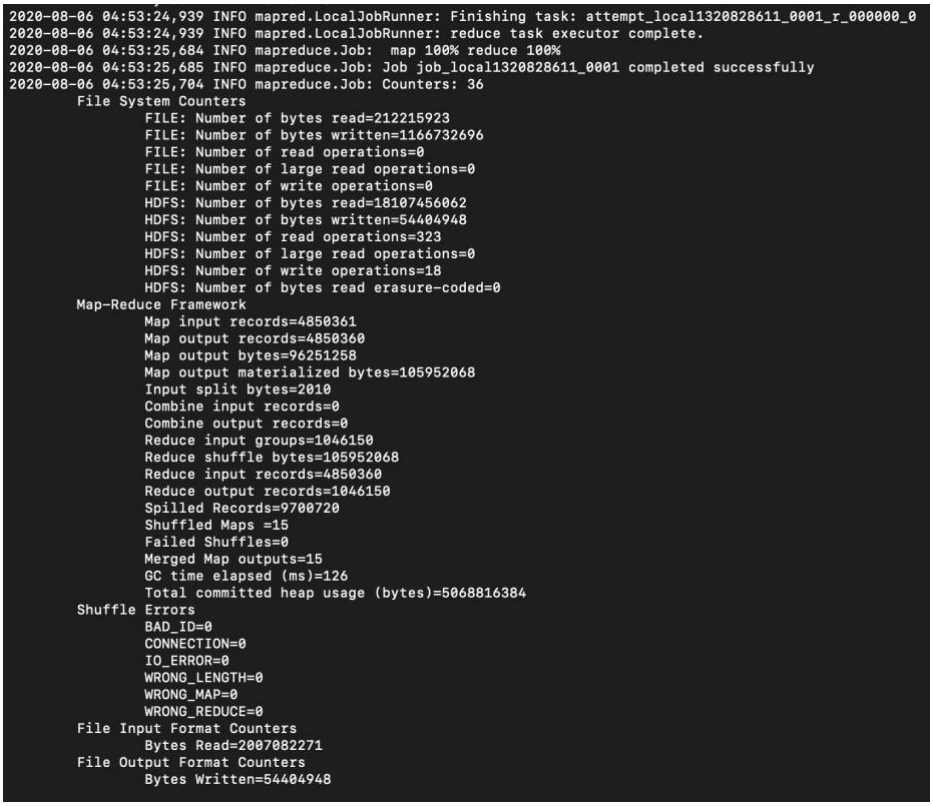
List of customers for each product who have given review (Distinct or Inverted Index Pattern)

We should know which customer has given reviews to which products so that we can keep track of the customers who value the products often.

This can be done by Distinct or Inverted Index Pattern MapReduce Algorithm.

Outputs

hadoop jar 6.DistinctCustomerReviews-1.0-SNAPSHOT.jar com.neu.edu.distinctcustomerreviews.Driver /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part6/result6



```
2020-08-06 04:53:24,939 INFO mapred.LocalJobRunner: Finishing task: attempt_local1320828611_0001_r_000000_0
2020-08-06 04:53:24,939 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-08-06 04:53:25,684 INFO mapreduce.Job:  map 100% reduce 100%
2020-08-06 04:53:25,685 INFO mapreduce.Job: Job job_local1320828611_0001 completed successfully
2020-08-06 04:53:25,704 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=212215923
                FILE: Number of bytes written=1166732696
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=18107456062
                HDFS: Number of bytes written=54404948
                HDFS: Number of read operations=323
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=18
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=4850361
                Map output records=4850360
                Map output bytes=96251258
                Map output materialized bytes=105952068
                Input split bytes=2010
                Combine input records=0
                Combine output records=0
                Reduce input groups=1046150
                Reduce shuffle bytes=105952068
                Reduce input records=4850360
                Reduce output records=1046150
                Spilled Records=9700720
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=126
                Total committed heap usage (bytes)=5068816384
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=54404948
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part6/result
2020-08-06 04:53:59,740 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-06 04:53 /AmazonSports/Part6/result/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup   54404948 2020-08-06 04:53 /AmazonSports/Part6/result/part-r-00000
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -tail /AmazonSports/Part6/result/part-r-00000
2020-08-06 04:54:31,133 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-06 04:54:31,822 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
179L6   5738241 5133685 26125803 31427662
B01461KRX8      51535117 16762727
B014620O6W      5105514
B01462HANM      17853721 14748016 52626196
B014639E8A      26164063 14067534 27093029 33560486
B0146C85N6      10437167
B0147L5TDU      845788 46169977 24623395 822270 20086820 52617800 29095902 37226003 12831018
B0147LHI8Y      41453032
B0147M421Y      41535494
B0147Q0UM0      15381326 13542931 28769280 10595079
B01482J7D6      1692747
B014AYH8PQ      42301608
B014B94BEQ      18160835
B014BBEPD6      30068081
B014C3P7LM      26091241
B014CGWRJY      1847163
B014CUCIOO      22585367
B014EJBPJC      1648307
B014FVKH8Y      20615390
B014G4BSBK      18354793
B014G7RG9K      24059960 42196028
B014GB81AY      49513584
B014GGEKFO      18354793
B014GLZCLU      30068081
B014GNO5JI      9992186
B014GUHTEE      37532836
B014H2P150      30734873
B014JG97LS      16191179
B014K11D2S      14334408
B014KW4BHY      3943705
B014L9YYGQ      41499011
B014LN3T5Y      431745
B014PX90WQ      25836507 45202200
B014Q2SGSA      137711 18967095
B014RCREPA      33659915
B018D6XMYI      43012151
B01C4N7OQ8      25012877
B01C69JNCS      17850781
B01G316GQG      20654459
B01MTN5XWB      28938840 16412121 35614767
B06X9L3MVQ      11506095
```

Code

CustomerMapper

```
package com.neu.edu.distinctcustomerreviews;


import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


/**

 *
```

```
 * @author kaushikpatil

 */

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



    private Text product = new Text();

    private Text customerid = new Text();



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

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

            return;

        }



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

        customerid.set(tokens[1]);

        product.set(tokens[3]);



        try {

            context.write(product, customerid);

        }

        catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}
```

CustomerReducer

```
package com.neu.edu.distinctcustomerreviews;



import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/**

 *

 * @author kaushikpatil

 */

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



    private Text result = new Text();



    @Override

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



        StringBuilder sb = new StringBuilder();

        boolean first = true;

        for (Text id : values) {

            if (first) {

                first = false;
```

```
            } else { sb.append(" ");

            }

            sb.append(id.toString());

        } result.set(sb.toString());

        context.write(key, result);

    }

}
```

Driver

```
package com.neu.edu.distinctcustomerreviews;


import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

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;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


/**

 *

 * @author kaushikpatil

 */

public class Driver {
```

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



        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf,"Customer Details");



        job.setJarByClass(Driver.class);

        job.setMapperClass(CustomerMapper.class);

        job.setReducerClass(CustomerReducer.class);

        job.setMapOutputKeyClass(Text.class);

        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(Text.class);



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

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



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

    }

}
```

Legit Customer Review Analysis (Numerical Summarization Pattern)

Customer Product Helpful Ratio can be calculated by taking two columns – helpful_votes and total_votes in the dataset.

This is to be calculated to get the percentage of actual legit and helpful customer reviews which will be useful in customer product analysis.

Outputs

hadoop jar 7.LegitCustomerAnalysis-1.0-SNAPSHOT.jar com.neu.edu.legitcustomeranalysis.Driver /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part7/result7

```
2020-08-06 05:21:16,957 INFO mapred.LocalJobRunner: Finishing task: attempt_local1909018568_0001_r_000000_0
2020-08-06 05:21:16,957 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-08-06 05:21:17,184 INFO mapreduce.Job:  map 100% reduce 100%
2020-08-06 05:21:17,184 INFO mapreduce.Job: Job job_local1909018568_0001 completed successfully
2020-08-06 05:21:17,197 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=165230463
                FILE: Number of bytes written=911343270
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=18107456062
                HDFS: Number of bytes written=18372006
                HDFS: Number of read operations=323
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=18
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=4850361
                Map output records=4850360
                Map output bytes=72755400
                Map output materialized bytes=82456210
                Input split bytes=2010
                Combine input records=0
                Combine output records=0
                Reduce input groups=1046150
                Reduce shuffle bytes=82456210
                Reduce input records=4850360
                Reduce output records=1046150
                Spilled Records=9700720
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=183
                Total committed heap usage (bytes)=4325376000
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=18372006
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part7/result
2020-08-06 05:22:08,106 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-06 05:21 /AmazonSports/Part7/result/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup   18372006 2020-08-06 05:21 /AmazonSports/Part7/result/part-r-00000
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -head /AmazonSports/Part7/result/part-r-00000
2020-08-06 05:23:07,519 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-06 05:23:08,257 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
50.0    % 0000031852
0.0     % 0000031909
0.0     % 0022320113
0.0     % 0201593246
25.0    % 0385494785
42.86   % 0395911737
0.0     % 0500615594
0.0     % 052800607X
0.0     % 0531904822
33.33   % 0578113171
0.0     % 0607968699
66.67   % 0615154077
0.0     % 0615203868
0.0     % 061520581X
100.0   % 0615247180
0.0     % 0615276156
50.0    % 061528101X
16.67   % 0615295797
9.09    % 0615329020
0.0     % 061533301X
0.0     % 0615375790
100.0   % 0615389953
0.0     % 0615397255
0.0     % 0615541003
0.0     % 0615614604
14.29   % 0615637620
71.43   % 0615715397
33.33   % 0615793940
50.0    % 0615801153
60.0    % 061586628X
0.0     % 0641649975
0.0     % 0641649983
0.0     % 0641869169
0.0     % 0646518178
0.0     % 0679771611
0.0     % 068143273X
50.0    % 0692250573
0.0     % 0736060332
33.33   % 0736065768
7.32    % 0743273575
44.44   % 0743294815
0.0     % 0743608658
0.0     % 0743608666
0.0     % 074360962X
0.0     % 0743609700
0.0     % 0743609840
0.0     % 0743610016
50.0    % 0743610431
66.67   % 074361058X
```

Code

HelpfulMapper

```
package com.neu.edu.legitcustomeranalysis;


import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


/**
```

```
 *

 * @author kaushikpatil

 */

public class HelpfulMapper extends Mapper<LongWritable, Text, Text,
FloatWritable> {



    private Text text = new Text();

    private FloatWritable score = new FloatWritable();



    @Override

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

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

            return;

        } else {

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

            String productid = line[3].trim();

            int num = Integer.parseInt(line[8].trim());

            int deno = Integer.parseInt(line[9].trim());

            float rat;



            if (deno != 0) {

                rat = num / deno;

            } else {

                rat = (float) 0.0;
```

```
            }


            text.set(productid);

            score.set(rat);

            context.write(text, score);

        }

    }

}
```

HelpfulReducer

```
package com.neu.edu.legitcustomeranalysis;


import java.io.IOException;

import java.text.DecimalFormat;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;


/**

 *

 * @author kaushikpatil

 */

public class HelpfulReducer extends Reducer<Text, FloatWritable,
FloatWritable, Text> {


    private FloatWritable result = new FloatWritable();
```

```
    @Override

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



        float sum = 0;

        int count = 0;



        for (FloatWritable val : values) {

            sum += val.get();

            count = count + 1;

        }



        float average = (sum / count) * 100;

        DecimalFormat df = new DecimalFormat();

        df.setMaximumFractionDigits(2);

        float avg = Float.parseFloat(df.format(average));

        key.set("%" + " " + key);

        result.set(avg);

        context.write(result, key);

    }}
```

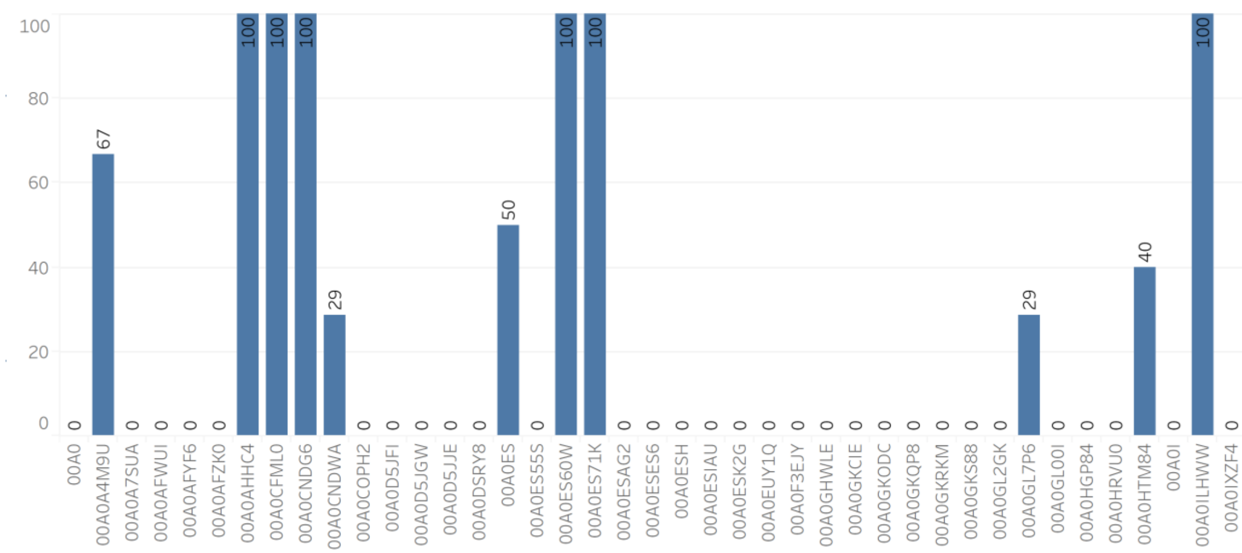
Driver

```
package com.neu.edu.legitcustomeranalysis;



import java.io.IOException;
```

```
import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;



/**

 *

 * @author kaushikpatil

 */

public class Driver {



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



        Configuration conf = new Configuration();



        Job job = Job.getInstance(conf, "Legit Customer Analysis");

        job.setJarByClass(Driver.class);

        job.setMapperClass(HelpfulMapper.class);

        job.setMapOutputKeyClass(Text.class);

        job.setMapOutputValueClass(FloatWritable.class);

        job.setReducerClass(HelpfulReducer.class);
```

```java
        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(FloatWritable.class);


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

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


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

    }

}
```

Data Visualization



Average of ratings and list of customers reviewed (Using Inner Join)

If we want to join two outputs in one, we will use Inner Join. Here we want to concatenate average of rating and customers who reviewed that product in dataset.

So, we will use two mappers and one reducer to get separate outputs and then join them with Inner Join Method.

Here, we can see the product's average rating and customers in one output making it convenient for product analysis.

Outputs

hadoop jar 8.RatingReviewJoin-1.0-SNAPSHOT.jar com.neu.edu.ratingreviewjoin.Driver /AmazonSports/Part5/result1 /AmazonSports/Part6/result6 /AmazonSports/Part8/result8

```
2020-08-06 06:42:29,751 INFO mapred.LocalJobRunner: Finishing task: attempt_local2018251499_0001_r_000000_0
2020-08-06 06:42:29,751 INFO mapred.LocalJobRunner: reduce task executor complete.
2020-08-06 06:42:29,977 INFO mapreduce.Job:  map 100% reduce 100%
2020-08-06 06:42:29,978 INFO mapreduce.Job: Job job_local2018251499_0001 completed successfully
2020-08-06 06:42:29,990 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=200663092
                FILE: Number of bytes written=371789458
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=196134130
                HDFS: Number of bytes written=70864478
                HDFS: Number of read operations=30
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=5
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=2092300
                Map output records=2092298
                Map output bytes=96052730
                Map output materialized bytes=100319203
                Input split bytes=567
                Combine input records=0
                Combine output records=0
                Reduce input groups=1046149
                Reduce shuffle bytes=100319203
                Reduce input records=2092298
                Reduce output records=1046149
                Spilled Records=4184596
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=60
                Total committed heap usage (bytes)=852492288
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=70864478
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part8/InnerJoinOutput
2020-08-06 06:42:54,340 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-06 06:42 /AmazonSports/Part8/InnerJoinOutput/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup   70864478 2020-08-06 06:42 /AmazonSports/Part8/InnerJoinOutput/part-r-00000
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -head /AmazonSports/Part8/InnerJoinOutput/part-r-00000
2020-08-06 06:43:53,522 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-08-06 06:43:54,152 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
0000031909      3.25    0000031909      50110877 39138559 45176396 49005643
0022320113      5.0     0022320113      45533436
0201593246      4.857143        0201593246      15031543 48812301 49404030 6248111 15265695 47311518 26197195
0385494785      3.5     0385494785      14241996 52475148 16095194 46404058
0395911737      4.571429        0395911737      51881165 53041254 52675938 35478205 52799315 52752188 52532820
0500615594      2.0     0500615594      22485530
052800607X      2.75    052800607X      843853 2869597 3324294 568606 1735807 11479803 821662 1817311
0531904822      5.0     0531904822      36455708
0578113171      5.0     0578113171      51629964 25113250 48291098 29746557 9908692 27708164
0607968699      3.0     0607968699      14323852
0615154077      5.0     0615154077      32355433 45157772 46388199
0615203868      2.0     0615203868      17338934 34860726
061520581X      5.0     061520581X      50399755 44467309 29904472
0615247180      5.0     0615247180      12612319
0615276156      5.0     0615276156      15489487
061528101X      5.0     061528101X      12705262 29941766
0615295797      5.0     0615295797      41191769 18136261 12508175 53019381 50644652 15203272
```

Code

RatingReviewMapper1

```
package com.neu.edu.ratingreviewjoin;


import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


/**

 *

 * @author kaushikpatil

 */

public class RatingReviewMapper1 extends Mapper<LongWritable, Text, Text,
Text> {


    private Text k = new Text();

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

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

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

            return;

        }



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

        String productid = Input[0].trim();

        if (productid == null) {

            return;

        }



        k.set(productid);

        v.set("*" + value.toString());

        context.write(k, v);

    }

}
```

RatingReviewMapper2

```
package com.neu.edu.ratingreviewjoin;



import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;
```

```
/**

 *

 * @author kaushikpatil

 */

public class RatingReviewMapper2 extends Mapper<LongWritable, Text, Text,
Text> {



    private Text k = new Text();

    private Text v = new Text();



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

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

            return;

        }



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

        String productid = Input[0].trim();

        if (productid == null) {

            return;

        }



        k.set(productid);

        v.set("#" + value.toString());
```

```
        context.write(k, v);
    }
}
```

RatingReviewReducer

```
package com.neu.edu.ratingreviewjoin;


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


/**
 *
 * @author kaushikpatil
 */
public class RatingReviewReducer extends Reducer<Text, Text, Text, Text> {


    private static final Text EMPTY_TEXT = new Text("");
    private Text tmp = new Text();
    private ArrayList<Text> listA = new ArrayList<Text>();
    private ArrayList<Text> listB = new ArrayList<Text>();
    private String joinType = null;


    public void setup(Reducer.Context context) {
```

```
        joinType = context.getConfiguration().get("join.type");

    }



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



        listA.clear();

        listB.clear();

        while (values.iterator().hasNext()) {

            tmp = values.iterator().next();

            if (Character.toString((char) tmp.charAt(0)).equals("*")) {

                listA.add(new Text(tmp.toString().substring(1)));

            }

            if (Character.toString((char) tmp.charAt(0)).equals("#")) {

                listB.add(new Text(tmp.toString().substring(1)));

            }

        }



        executeJoinLogic(context);

    }



    private void executeJoinLogic(Context context) throws IOException,
InterruptedException {

        if (joinType.equalsIgnoreCase("Inner")) {

            if (!listA.isEmpty() && !listB.isEmpty()) {
```

```
                for (Text A : listA) {
                    for (Text B : listB) {
                        context.write(A, B);
                    }
                }
            }
        }
    }
}
```

Driver

```
package com.neu.edu.ratingreviewjoin;


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


/**
 *
 * @author kaushikpatil
```

```
 */

public class Driver {



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



        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Inner Join");

        job.setJarByClass(Driver.class);

        MultipleInputs.addInputPath(job, new Path(args[0]),
TextInputFormat.class, RatingReviewMapper1.class);

        MultipleInputs.addInputPath(job, new Path(args[1]),
TextInputFormat.class, RatingReviewMapper2.class);

        job.getConfiguration().set("join.type", "Inner");

        job.setReducerClass(RatingReviewReducer.class);

        job.setOutputFormatClass(TextOutputFormat.class);

        TextOutputFormat.setOutputPath(job, new Path(args[2]));

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(Text.class);



        System.exit(job.waitForCompletion(true) ? 0 : 2);

    }

}
```

Amazon Word Count

We are implementing this analysis for getting the word frequency i.e. the number of times the word has occurred in the review. This helps in knowing that particular review is positive, negative or neutral.

Outputs

hadoop jar 9.WordCount-1.0-SNAPSHOT.jar com.neu.edu.wordcount.Driver /AmazonSports/amazon_reviews_us_Sports_v1_00.tsv /AmazonSports/Part9/result9

```
2020-08-11 09:47:44,342 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=12822347902
                FILE: Number of bytes written=18340053612
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2007084281
                HDFS: Number of bytes written=5334466309
                HDFS: Number of read operations=50
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Launched map tasks=16
                Launched reduce tasks=1
                Data-local map tasks=16
                Total time spent by all maps in occupied slots (ms)=1059206
                Total time spent by all reduces in occupied slots (ms)=373671
                Total time spent by all map tasks (ms)=1059206
                Total time spent by all reduce tasks (ms)=373671
                Total vcore-milliseconds taken by all map tasks=1059206
                Total vcore-milliseconds taken by all reduce tasks=373671
                Total megabyte-milliseconds taken by all map tasks=1084626944
                Total megabyte-milliseconds taken by all reduce tasks=382639104
        Map-Reduce Framework
                Map input records=4850361
                Map output records=237363179
                Map output bytes=5038750298
                Map output materialized bytes=5513477796
                Input split bytes=2010
                Combine input records=0
                Combine output records=0
                Reduce input groups=1046151
                Reduce shuffle bytes=5513477796
                Reduce input records=237363179
                Reduce output records=103971702
                Spilled Records=789265717
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=3894
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=6466568192
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2007082271
        File Output Format Counters
                Bytes Written=5334466309
```

```
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -ls /AmazonSports/Part9/result9
2020-08-11 09:47:55,275 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 kaushikpatil supergroup          0 2020-08-11 09:47 /AmazonSports/Part9/result9/_SUCCESS
-rw-r--r--   1 kaushikpatil supergroup 5334466309 2020-08-11 09:47 /AmazonSports/Part9/result9/part-r-00000
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$ hadoop fs -tail /AmazonSports/Part9/result9/part-r-00000
2020-08-11 09:48:21,887 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

B06X9L3MVQ      WordFrequency [word=outside, frequency=1]
B06X9L3MVQ      WordFrequency [word=pepper, frequency=1]
B06X9L3MVQ      WordFrequency [word=pink, frequency=1]
B06X9L3MVQ      WordFrequency [word=products., frequency=1]
B06X9L3MVQ      WordFrequency [word=pushed, frequency=1]
B06X9L3MVQ      WordFrequency [word=red, frequency=1]
B06X9L3MVQ      WordFrequency [word=second, frequency=1]
B06X9L3MVQ      WordFrequency [word=spray, frequency=1]
B06X9L3MVQ      WordFrequency [word=sprays, frequency=1]
B06X9L3MVQ      WordFrequency [word=surprised, frequency=1]
B06X9L3MVQ      WordFrequency [word=test, frequency=3]
B06X9L3MVQ      WordFrequency [word=that, frequency=1]
B06X9L3MVQ      WordFrequency [word=the, frequency=4]
B06X9L3MVQ      WordFrequency [word=then, frequency=1]
B06X9L3MVQ      WordFrequency [word=this, frequency=2]
B06X9L3MVQ      WordFrequency [word=to, frequency=3]
B06X9L3MVQ      WordFrequency [word=was, frequency=1]
B06X9L3MVQ      WordFrequency [word=with, frequency=1]
B06X9L3MVQ      WordFrequency [word=you, frequency=2]
product_id      WordFrequency [word=review_body, frequency=1]
(base) Kaushiks-MacBook-Pro:documents kaushikpatil$
```

Code

WordCountMapper

```
package com.neu.edu.wordcount;



import java.io.IOException;

import java.util.Arrays;

import java.util.HashSet;

import java.util.Set;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;



/**

 *

 * @author kaushikpatil

 */

public class WordCountMapper extends Mapper<Object,Text,Text,WordFrequency>{
```

```
    @Override

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

      InterruptedException {



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



            String product = values[3].toString();

            String review = values[13].toString();



            Text prod = new Text();

            prod.set(product);



            for(String str : review.split(" ")) {



                WordFrequency wf = new WordFrequency(1, str);

                context.write(prod,wf);

            }

    }

}
```

WordCountReducer

```
package com.neu.edu.wordcount;



import java.io.IOException;
```

```
import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;



/**

 *

 * @author kaushikpatil

 */

public class WordCountReducer extends Reducer<Text, WordFrequency, Text,WordFrequency>{



    @Override



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



        Map<String, Integer> frequencyMap = new TreeMap<String,Integer>();



        for(WordFrequency wf : values) {



            if(!frequencyMap.containsKey(wf.getWord())) {

                frequencyMap.put(wf.getWord(), wf.getFrequency());

            }

            else {
```

```
                int freq = frequencyMap.get(wf.getWord());
                frequencyMap.put(wf.getWord(), freq + 1);
            }
        }

        for(Map.Entry<String, Integer> entry : frequencyMap.entrySet()) {

            WordFrequency wf1 = new WordFrequency();
            wf1.setFrequency(entry.getValue());
            wf1.setWord(entry.getKey());
            context.write(key, wf1);
        }
    }
}
```

WordFrequency

```
package com.neu.edu.wordcount;

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

```
/**
 *
 * @author kaushikpatil
 */
public class WordFrequency implements Writable, WritableComparable<WordFrequency>{

    private String word;
    private int frequency;

    public WordFrequency() {
        super();
    }

    public WordFrequency(int frequency, String word) {
        super();
        this.frequency = frequency;
        this.word = word;
    }

    @Override

    public String toString() {
        return "WordFrequency [word=" + word + ", frequency=" + frequency + "]";
```

```
}

public String getWord() {
    return word;
}

public void setWord(String word) {
    this.word = word;
}

public int getFrequency() {
    return frequency;
}

public void setFrequency(int frequency) {
    this.frequency = frequency;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
  result = prime * result + frequency;
  result = prime * result + ((word == null) ? 0 : word.hashCode());
```

```
  return result;
}

@Override
public boolean equals(Object obj) {

    if (this == obj)
        return true;

    if (obj == null)
        return false;

    if (getClass() != obj.getClass())
        return false;

    WordFrequency other = (WordFrequency) obj;
    if (frequency != other.frequency)
        return false;

    if (word == null) {
        if (other.word != null)
            return false;

    } else if (!word.equals(other.word))
```

```
        return false;

    return true;
}

public void readFields(DataInput in) throws IOException {

    frequency = in.readInt();
  word = in.readUTF();
}

public void write(DataOutput out) throws IOException {

    out.writeInt(frequency);
    out.writeUTF(word);
}

public int compareTo(WordFrequency o) {

    int result = -1;

    if(frequency >  o.frequency) {
        result = 1;
    }
```

```
        if(frequency == o.frequency) {
            result = 0;
        }

        if(result == 0) {
            result  = word.compareTo(o.word);
        }

        return result;
    }
}
```

Driver

```
package com.neu.edu.wordcount;

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;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
/**
 *
 * @author kaushikpatil
 */
public class Driver {

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

        if (args.length != 2) {
            System.err.println("Usage: MaxSubmittedCharge <input path> <output
path>");
            System.exit(-1);
        }

      Path inputPath = new Path(args[0]);
      Path outputDir = new Path(args[1]);

      Configuration conf = new Configuration(true);

      Job job = Job.getInstance(conf);
      job.setJarByClass(WordCountMapper.class);

      job.setMapperClass(WordCountMapper.class);
      job.setReducerClass(WordCountReducer.class);
      job.setNumReduceTasks(1);
```

```
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(WordFrequency.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(WordFrequency.class);

        FileInputFormat.addInputPath(job, inputPath);
        job.setInputFormatClass(TextInputFormat.class);

        FileOutputFormat.setOutputPath(job, outputDir);

        FileSystem hdfs = FileSystem.get(conf);
          if (hdfs.exists(outputDir)) {
              hdfs.delete(outputDir, true);
          }

          int code = job.waitForCompletion(true) ? 0 : 1;
          System.exit(code);
    }
}
```

Stemmer

```
package com.neu.edu.wordcount;

class Stemmer
```

```
{  private char[] b;
   private int i,     /* offset into b */
               i_end, /* offset to end of stemmed word */
               j, k;
   private static final int INC = 50;
                    /* unit of size whereby b is increased */
   public Stemmer()
   {  b = new char[INC];
      i = 0;
      i_end = 0;
   }

   /**
    * Add a character to the word being stemmed.  When you are finished
    * adding characters, you can call stem(void) to stem the word.
    */

   public void add(char ch)
   {  if (i == b.length)
      {  char[] new_b = new char[i+INC];
         for (int c = 0; c < i; c++) new_b[c] = b[c];
         b = new_b;
      }
      b[i++] = ch;
```

```
}


/** Adds wLen characters to the word being stemmed contained in a portion
 * of a char[] array. This is like repeated calls of add(char ch), but
 * faster.
 */

public void add(char[] w, int wLen)
{  if (i+wLen >= b.length)
   {  char[] new_b = new char[i+wLen+INC];
      for (int c = 0; c < i; c++) new_b[c] = b[c];
      b = new_b;
   }
   for (int c = 0; c < wLen; c++) b[i++] = w[c];
}


/**
 * After a word has been stemmed, it can be retrieved by toString(),
 * or a reference to the internal buffer can be retrieved by getResultBuffer
 * and getResultLength (which is generally more efficient.)
 */
public String toString() { return new String(b,0,i_end); }
```

```
/**
 * Returns the length of the word resulting from the stemming process.
 */
public int getResultLength() { return i_end; }

/**
 * Returns a reference to a character buffer containing the results of
 * the stemming process.  You also need to consult getResultLength()
 * to determine the length of the result.
 */
public char[] getResultBuffer() { return b; }

/* cons(i) is true <=> b[i] is a consonant. */

private final boolean cons(int i)
{  switch (b[i])
   {  case 'a': case 'e': case 'i': case 'o': case 'u': return false;
      case 'y': return (i==0) ? true : !cons(i-1);
      default: return true;
   }
}

/* m() measures the number of consonant sequences between 0 and j. if c is
   a consonant sequence and v a vowel sequence, and <..> indicates arbitrary
```

```
   presence,
      <c><v>       gives 0
      <c>vc<v>     gives 1
      <c>vcvc<v>   gives 2
      <c>vcvcvc<v> gives 3
      ....
*/

private final int m()
{  int n = 0;
   int i = 0;
   while(true)
   {  if (i > j) return n;
      if (! cons(i)) break; i++;
   }
   i++;
   while(true)
   {  while(true)
      {  if (i > j) return n;
            if (cons(i)) break;
            i++;
      }
      i++;
      n++;
```

```
      while(true)
      {  if (i > j) return n;
         if (! cons(i)) break;
         i++;
      }
      i++;
    }
}

/* vowelinstem() is true <=> 0,...j contains a vowel */

private final boolean vowelinstem()
{  int i; for (i = 0; i <= j; i++) if (! cons(i)) return true;
   return false;
}

/* doublec(j) is true <=> j,(j-1) contain a double consonant. */

private final boolean doublec(int j)
{  if (j < 1) return false;
   if (b[j] != b[j-1]) return false;
   return cons(j);
}
```

```
/* cvc(i) is true <=> i-2,i-1,i has the form consonant - vowel - consonant
   and also if the second c is not w,x or y. this is used when trying to
   restore an e at the end of a short word. e.g.
      cav(e), lov(e), hop(e), crim(e), but
      snow, box, tray.
*/

private final boolean cvc(int i)
{  if (i < 2 || !cons(i) || cons(i-1) || !cons(i-2)) return false;
   {  int ch = b[i];
      if (ch == 'w' || ch == 'x' || ch == 'y') return false;
   }
   return true;
}

private final boolean ends(String s)
{  int l = s.length();
   int o = k-l+1;
   if (o < 0) return false;
   for (int i = 0; i < l; i++) if (b[o+i] != s.charAt(i)) return false;
   j = k-l;
   return true;
}
```

```
/* setto(s) sets (j+1),...k to the characters in the string s, readjusting
   k. */

private final void setto(String s)
{  int l = s.length();
   int o = j+1;
   for (int i = 0; i < l; i++) b[o+i] = s.charAt(i);
   k = j+l;
}

/* r(s) is used further down. */

private final void r(String s) { if (m() > 0) setto(s); }

/* step1() gets rid of plurals and -ed or -ing. e.g.
       caresses  ->  caress
       ponies    ->  poni
       ties      ->  ti
       caress    ->  caress
       cats      ->  cat
       feed      ->  feed
       agreed    ->  agree
       disabled  ->  disable
       matting   ->  mat
```

```
        mating    ->  mate

        meeting   ->  meet

        milling   ->  mill

        messing   ->  mess

        meetings  ->  meet

*/


private final void step1()

{  if (b[k] == 's')

   {  if (ends("sses")) k -= 2; else

      if (ends("ies")) setto("i"); else

      if (b[k-1] != 's') k--;

   }

   if (ends("eed")) { if (m() > 0) k--; } else

   if ((ends("ed") || ends("ing")) && vowelinstem())

   {  k = j;

      if (ends("at")) setto("ate"); else

      if (ends("bl")) setto("ble"); else

      if (ends("iz")) setto("ize"); else

      if (doublec(k))

      {  k--;

         {  int ch = b[k];

            if (ch == 'l' || ch == 's' || ch == 'z') k++;

         }
```

```
        }
        else if (m() == 1 && cvc(k)) setto("e");
      }
   }

   /* step2() turns terminal y to i when there is another vowel in the stem. */

   private final void step2() { if (ends("y") && vowelinstem()) b[k] = 'i'; }

   /* step3() maps double suffices to single ones. so -ization ( = -ize plus
      -ation) maps to -ize etc. note that the string before the suffix must
give
      m() > 0. */

   private final void step3() { if (k == 0) return; /* For Bug 1 */ switch (b[k-
1])
   {
       case 'a': if (ends("ational")) { r("ate"); break; }
                 if (ends("tional")) { r("tion"); break; }
                 break;
       case 'c': if (ends("enci")) { r("ence"); break; }
                 if (ends("anci")) { r("ance"); break; }
                 break;
       case 'e': if (ends("izer")) { r("ize"); break; }
                 break;
```

```
    case 'l': if (ends("bli")) { r("ble"); break; }
              if (ends("alli")) { r("al"); break; }
              if (ends("entli")) { r("ent"); break; }
              if (ends("eli")) { r("e"); break; }
              if (ends("ousli")) { r("ous"); break; }
              break;
    case 'o': if (ends("ization")) { r("ize"); break; }
              if (ends("ation")) { r("ate"); break; }
              if (ends("ator")) { r("ate"); break; }
              break;
    case 's': if (ends("alism")) { r("al"); break; }
              if (ends("iveness")) { r("ive"); break; }
              if (ends("fulness")) { r("ful"); break; }
              if (ends("ousness")) { r("ous"); break; }
              break;
    case 't': if (ends("aliti")) { r("al"); break; }
              if (ends("iviti")) { r("ive"); break; }
              if (ends("biliti")) { r("ble"); break; }
              break;
    case 'g': if (ends("logi")) { r("log"); break; }
} }

/* step4() deals with -ic-, -full, -ness etc. similar strategy to step3. */
```

```
private final void step4() { switch (b[k])
{
    case 'e': if (ends("icate")) { r("ic"); break; }
              if (ends("ative")) { r(""); break; }
              if (ends("alize")) { r("al"); break; }
              break;
    case 'i': if (ends("iciti")) { r("ic"); break; }
              break;
    case 'l': if (ends("ical")) { r("ic"); break; }
              if (ends("ful")) { r(""); break; }
              break;
    case 's': if (ends("ness")) { r(""); break; }
              break;
} }

/* step5() takes off -ant, -ence etc., in context <c>vcvc<v>. */

private final void step5()
{   if (k == 0) return; /* for Bug 1 */ switch (b[k-1])
    {  case 'a': if (ends("al")) break; return;
       case 'c': if (ends("ance")) break;
                 if (ends("ence")) break; return;
       case 'e': if (ends("er")) break; return;
       case 'i': if (ends("ic")) break; return;
```

```
            case 'l': if (ends("able")) break;
                      if (ends("ible")) break; return;
            case 'n': if (ends("ant")) break;
                      if (ends("ement")) break;
                      if (ends("ment")) break;
                      /* element etc. not stripped before the m */
                      if (ends("ent")) break; return;
            case 'o': if (ends("ion") && j >= 0 && (b[j] == 's' || b[j] == 't'))
break;
                                      /* j >= 0 fixes Bug 2 */
                      if (ends("ou")) break; return;
                      /* takes care of -ous */
            case 's': if (ends("ism")) break; return;
            case 't': if (ends("ate")) break;
                      if (ends("iti")) break; return;
            case 'u': if (ends("ous")) break; return;
            case 'v': if (ends("ive")) break; return;
            case 'z': if (ends("ize")) break; return;
            default: return;
        }
        if (m() > 1) k = j;
    }


    /* step6() removes a final -e if m() > 1. */
```

```
private final void step6()
{  j = k;
   if (b[k] == 'e')
   {  int a = m();
      if (a > 1 || a == 1 && !cvc(k-1)) k--;
   }
   if (b[k] == 'l' && doublec(k) && m() > 1) k--;
}


/** Stem the word placed into the Stemmer buffer through calls to add().
 * Returns true if the stemming process resulted in a word different
 * from the input.  You can retrieve the result with
 * getResultLength()/getResultBuffer() or toString().
 */
public void stem()
{  k = i - 1;
   if (k > 1) { step1(); step2(); step3(); step4(); step5(); step6(); }
   i_end = k+1; i = 0;
}


public String stem(String s) {
      for(char c : s.toCharArray()) {
            add(c);
      }
```

```
        stem();
        return toString();
    }
}
```

Stopwords

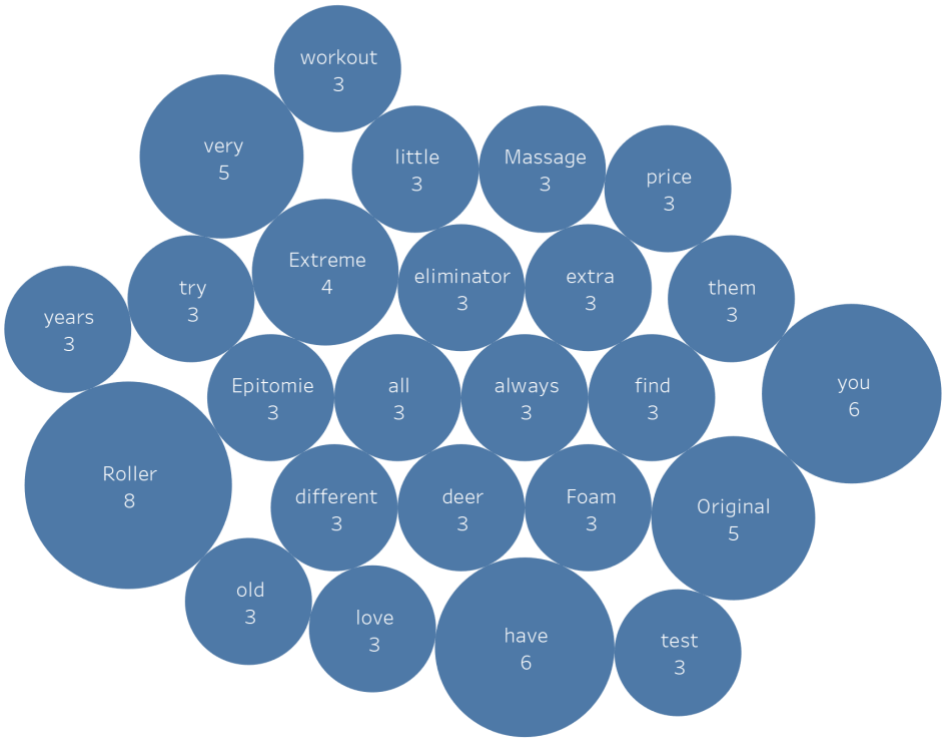
```
package com.neu.edu.wordcount;

import java.util.HashSet;
import java.util.Set;
import java.util.Arrays;

/**
 *
 * @author kaushikpatil
 */

public class Stopwords {

    public static String[] stopwords = {"B06X9L3MVQ"};
    public static Set<String> stopWordSet = new
HashSet<String>(Arrays.asList(stopwords));
    public static Set<String> stemmedStopWordSet = stemStringSet(stopWordSet);

    public static boolean isStopword(String word) {
```

```
        if(word.length() < 2) return true;

        if(word.charAt(0) >= '0' && word.charAt(0) <= '9')

            return true; //remove numbers, "25th", etc

        if(stopWordSet.contains(word)) return true;

        else return false;

    }


    public static boolean isStemmedStopword(String word) {

        if(word.length() < 2) return true;

        if(word.charAt(0) >= '0' && word.charAt(0) <= '9') return true; //remove
numbers, "25th", etc

        String stemmed = stemString(word);

        if(stopWordSet.contains(stemmed))

            return true;

        if(stemmedStopWordSet.contains(stemmed))

            return true;

      if(stopWordSet.contains(word))

            return true;

      if(stemmedStopWordSet.contains(word))

            return true;

        else

            return false;

      }


    public static String removeStopWords(String string) {
```

```
        String result = "";
        String[] words = string.split("\\s+");
      for(String word : words) {
            if(word.isEmpty()) continue;
            if(isStopword(string)) continue; //remove stopwords
            result += (word+" ");
        }
        return result;
    }

    public static String removeStemmedStopWords(String string) {

        String result = "";
        String[] words = string.split("\\s+");
        for(String word : words) {
            if(word.isEmpty()) continue;
            if(isStemmedStopword(word)) continue;
            if(word.charAt(0) >= '0' && word.charAt(0) <= '9') continue;
//remove numbers, "25th", etc
            result += (word+" ");
        }
        return result;
    }
```

```
    public static String stemString(String string) {
        return new Stemmer().stem(string);
    }

    public static Set<String> stemStringSet(Set<String> stringSet) {
        Stemmer stemmer = new Stemmer();
        Set<String> results = new HashSet<String>();
        for(String string : stringSet) {
            results.add(stemmer.stem(string));
        }
        return results;
    }
}
```

Data Visualization



Hive

```
hive> create schema AmazonSports;
OK
Time taken: 7.33 seconds
```

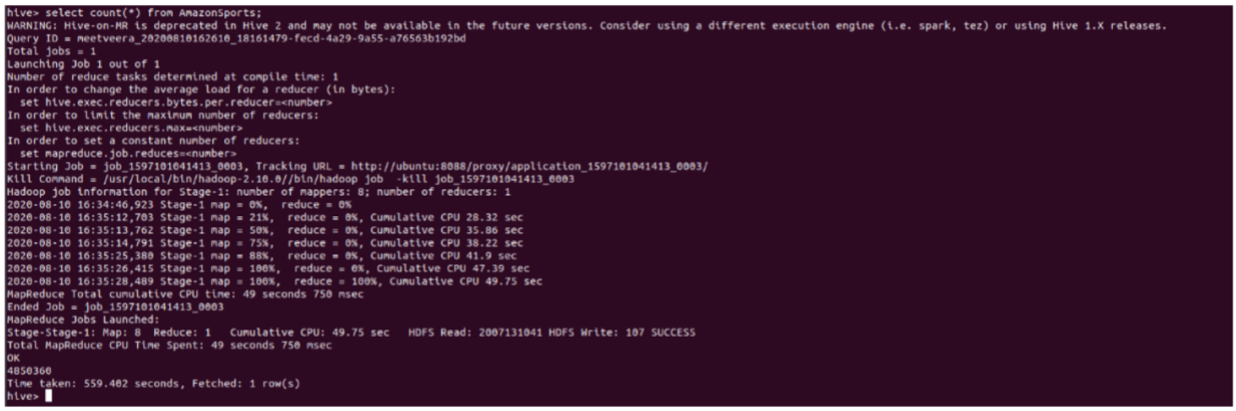
```
hive> use AmazonSports;
OK
Time taken: 0.053 seconds
```

```
hive> CREATE TABLE IF NOT EXISTS AmazonSports ( marketplace String, customer_id int, review_id String, product_id String, product_parent String, product_title String, product_category String, star_rating
float, helpful_votes float, total_votes int, Vine String, verified_purchase String, review_headline String, review_body String, review_date String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TER
MINATED BY '\n' tblproperties ("skip.header.line.count"="1");
OK
Time taken: 1.341 seconds
hive>
```

```
hive> LOAD DATA INPATH '/AmazonSports' OVERWRITE INTO TABLE AmazonSports;
Loading data to table amazonsports.amazonsports
OK
Time taken: 1.765 seconds
hive>
```

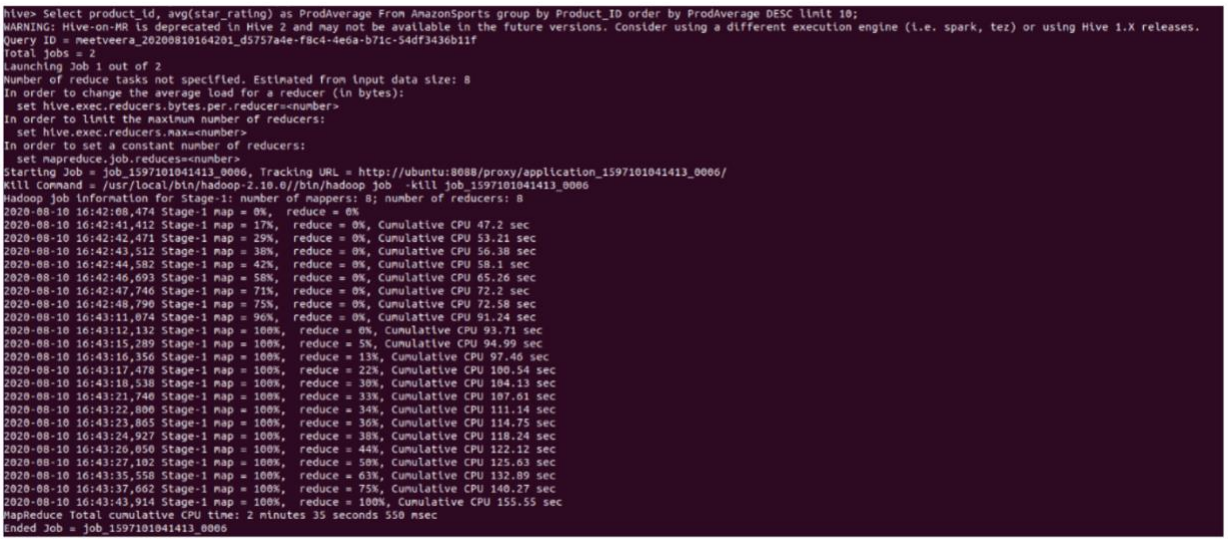
```
CREATE TABLE IF NOT EXISTS AmazonSports (
marketplace String,
customer_id int,
review_id String,
product_id String,
product_parent String,
product_title String,
product_category String,
```

```
star_rating float,
helpful_votes float,
total_votes int,
Vine String,
verified_purchase String,
review_headline String,
review_body String,
review_date String)
```
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' _LINES TERMINATED BY '\n'
tblproperties ("skip.header.line.count"="1");



Top 10 Best Products based on average ratings

Select product_id, avg(star_rating) as ProdAverage
From AmazonSports
group by Product_ID
order by ProdAverage DESC limit 10;



```
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0007, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0007/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0007
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-08-10 16:43:57,711 Stage-2 map = 0%,  reduce = 0%
2020-08-10 16:44:09,088 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 9.63 sec
2020-08-10 16:44:16,338 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 12.11 sec
MapReduce Total cumulative CPU time: 12 seconds 110 msec
Ended Job = job_1597101041413_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 8  Reduce: 8   Cumulative CPU: 155.55 sec   HDFS Read: 2007165285 HDFS Write: 38035708 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 12.11 sec   HDFS Read: 38043268 HDFS Write: 357 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 47 seconds 660 msec
OK
074361111X      5.0
0977943593      5.0
B01C69JNCS      5.0
0743610997      5.0
074361089X      5.0
074361108X      5.0
0615801153      5.0
0974632007      5.0
B014H2P150      5.0
0615715397      5.0
Time taken: 135.772 seconds, Fetched: 10 row(s)
hive>
```

Top 3 Customer based on the number of products

```
select customer_id, count(product_id) as totalProducts
from AmazonSports
group by customer_id
order by totalProducts desc limit 3;
```

```
hive> select customer_id, count(product_id) as totalProducts from AmazonSports group by customer_id order by totalProducts desc limit 3;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = meetveera_20200810163739_c054ad8b-2429-47da-8347-fe99ac74b09c
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 8
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0004, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0004/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0004
Hadoop job information for Stage-1: number of mappers: 8; number of reducers: 8
2020-08-10 16:37:46,867 Stage-1 map = 0%,  reduce = 0%
2020-08-10 16:38:12,555 Stage-1 map = 4%,  reduce = 0%, Cumulative CPU 24.76 sec
2020-08-10 16:38:13,624 Stage-1 map = 8%,  reduce = 0%, Cumulative CPU 37.58 sec
2020-08-10 16:38:18,919 Stage-1 map = 42%,  reduce = 0%, Cumulative CPU 50.32 sec
2020-08-10 16:38:19,986 Stage-1 map = 58%,  reduce = 0%, Cumulative CPU 57.17 sec
2020-08-10 16:38:22,132 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 64.97 sec
2020-08-10 16:38:41,143 Stage-1 map = 88%,  reduce = 0%, Cumulative CPU 73.51 sec
2020-08-10 16:38:42,203 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 81.16 sec
2020-08-10 16:38:46,443 Stage-1 map = 100%,  reduce = 8%, Cumulative CPU 83.53 sec
2020-08-10 16:38:49,638 Stage-1 map = 100%,  reduce = 17%, Cumulative CPU 87.57 sec
2020-08-10 16:38:50,683 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 93.39 sec
2020-08-10 16:38:51,733 Stage-1 map = 100%,  reduce = 36%, Cumulative CPU 98.76 sec
2020-08-10 16:38:52,779 Stage-1 map = 100%,  reduce = 37%, Cumulative CPU 102.4 sec
2020-08-10 16:38:53,832 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 103.56 sec
2020-08-10 16:38:54,869 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 109.75 sec
2020-08-10 16:39:06,473 Stage-1 map = 100%,  reduce = 63%, Cumulative CPU 117.09 sec
2020-08-10 16:39:07,518 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 124.51 sec
2020-08-10 16:39:11,726 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 138.85 sec
MapReduce Total cumulative CPU time: 2 minutes 18 seconds 850 msec
Ended Job = job_1597101041413_0004
```

```
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0005, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0005/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0005
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-08-10 16:39:24,629 Stage-2 map = 0%,  reduce = 0%
2020-08-10 16:39:38,041 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 13.85 sec
2020-08-10 16:39:44,219 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 15.85 sec
MapReduce Total cumulative CPU time: 15 seconds 850 msec
Ended Job = job_1597101041413_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 8  Reduce: 8   Cumulative CPU: 138.85 sec   HDFS Read: 2007161813 HDFS Write: 64458747 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 15.85 sec   HDFS Read: 64466289 HDFS Write: 162 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 34 seconds 700 msec
OK
50820654        552
37651511        294
26955164        272
Time taken: 125.4 seconds, Fetched: 3 row(s)
hive>
```

Top 3 Popular Product based on number of products sold

select product_id, count(product_id) as numberofprods
from AmazonSports
group by product_id
order by numberofprods desc limit 3;

```
hive> select product_id, count(product_id) as numberofprods from AmazonSports group by product_id order by numberofprods desc limit 3;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = meetveera_20200810165007_44c76e92-827a-4524-a53c-fec8bfd97c7a
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 8
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0010, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0010/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0010
Hadoop job information for Stage-1: number of mappers: 8; number of reducers: 8
2020-08-10 16:50:14,162 Stage-1 map = 0%,  reduce = 0%
2020-08-10 16:50:40,558 Stage-1 map = 4%,  reduce = 0%, Cumulative CPU 6.47 sec
2020-08-10 16:50:41,625 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 30.9 sec
2020-08-10 16:50:46,932 Stage-1 map = 17%,  reduce = 0%, Cumulative CPU 40.36 sec
2020-08-10 16:50:47,966 Stage-1 map = 54%,  reduce = 0%, Cumulative CPU 53.06 sec
2020-08-10 16:50:49,018 Stage-1 map = 58%,  reduce = 0%, Cumulative CPU 57.2 sec
2020-08-10 16:50:50,077 Stage-1 map = 71%,  reduce = 0%, Cumulative CPU 61.39 sec
2020-08-10 16:50:51,133 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 63.62 sec
2020-08-10 16:51:09,993 Stage-1 map = 88%,  reduce = 0%, Cumulative CPU 70.7 sec
2020-08-10 16:51:11,044 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 79.84 sec
2020-08-10 16:51:17,315 Stage-1 map = 100%,  reduce = 17%, Cumulative CPU 87.47 sec
2020-08-10 16:51:18,365 Stage-1 map = 100%,  reduce = 35%, Cumulative CPU 95.92 sec
2020-08-10 16:51:21,493 Stage-1 map = 100%,  reduce = 42%, Cumulative CPU 100.13 sec
2020-08-10 16:51:22,540 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 105.47 sec
2020-08-10 16:51:32,962 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 118.64 sec
2020-08-10 16:51:37,074 Stage-1 map = 100%,  reduce = 88%, Cumulative CPU 125.03 sec
2020-08-10 16:51:38,087 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 132.0 sec
MapReduce Total cumulative CPU time: 2 minutes 12 seconds 0 msec
Ended Job = job_1597101041413_0010
```

```
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0011, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0011/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0011
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-08-10 16:51:50,043 Stage-2 map = 0%,  reduce = 0%
2020-08-10 16:51:59,301 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 8.15 sec
2020-08-10 16:52:05,484 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 10.19 sec
MapReduce Total cumulative CPU time: 10 seconds 190 msec
Ended Job = job_1597101041413_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 8  Reduce: 8   Cumulative CPU: 132.0 sec   HDFS Read: 2007161437 HDFS Write: 30647298 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 10.19 sec   HDFS Read: 30654843 HDFS Write: 171 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 22 seconds 190 msec
OK
B001HBHNHE      7405
7245456313      3693
B00FX0S4DC      3051
Time taken: 120.218 seconds, Fetched: 3 row(s)
hive>
```

Number of products sold per day

Select review_date , count(product_id) as ProductsInDays
from AmazonSports
group by review_date
order by ProductsInDays desc limit 10;

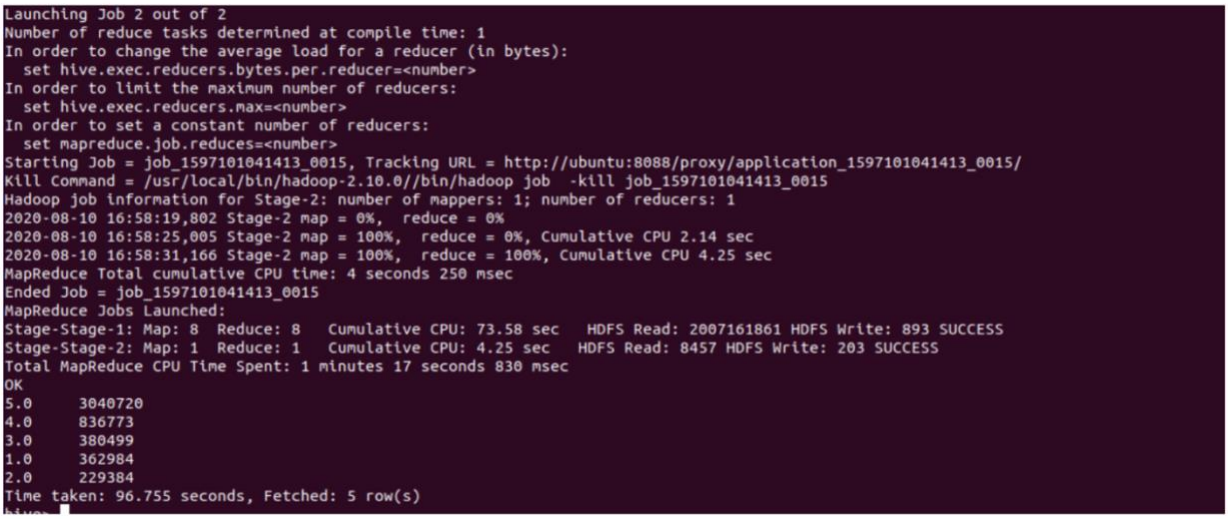
```
hive> Select review_date , count(product_id) as ProductsInDays from AmazonSports group by review_date order by ProductsInDays desc limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = meetveera_20200810165328_10f8d092-6f46-4f9b-aaa3-2e27e72647f4
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 8
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0012, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0012/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0012
Hadoop job information for Stage-1: number of mappers: 8; number of reducers: 8
2020-08-10 16:53:36,003 Stage-1 map = 0%,  reduce = 0%
2020-08-10 16:54:02,362 Stage-1 map = 4%,  reduce = 0%, Cumulative CPU 6.05 sec
2020-08-10 16:54:03,431 Stage-1 map = 38%,  reduce = 0%, Cumulative CPU 24.69 sec
2020-08-10 16:54:04,475 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 40.1 sec
2020-08-10 16:54:22,445 Stage-1 map = 88%,  reduce = 0%, Cumulative CPU 44.88 sec
2020-08-10 16:54:23,519 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 50.8 sec
2020-08-10 16:54:27,721 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 61.44 sec
2020-08-10 16:54:35,062 Stage-1 map = 100%,  reduce = 63%, Cumulative CPU 64.25 sec
2020-08-10 16:54:37,132 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 67.08 sec
2020-08-10 16:54:39,212 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 72.56 sec
MapReduce Total cumulative CPU time: 1 minutes 12 seconds 560 msec
Ended Job = job_1597101041413_0012
```

```
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0013, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0013/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0013
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-08-10 16:54:51,820 Stage-2 map = 0%,  reduce = 0%
2020-08-10 16:54:57,989 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.92 sec
2020-08-10 16:55:05,161 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.9 sec
MapReduce Total cumulative CPU time: 4 seconds 900 msec
Ended Job = job_1597101041413_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 8  Reduce: 8   Cumulative CPU: 72.56 sec   HDFS Read: 2007161741 HDFS Write: 151418 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.9 sec   HDFS Read: 158986 HDFS Write: 371 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 17 seconds 460 msec
OK
2015-01-03      10908
2015-01-05      10516
2015-06-03      10244
2014-12-29      10107
2015-01-07      9996
2015-01-09      9719
2015-01-04      9520
2015-08-04      9461
2015-01-12      9019
2015-08-18      8871
Time taken: 97.617 seconds, Fetched: 10 row(s)
hive>
```

Number of products per rating

select star_rating, count(product_id) as ProductsInRating
from AmazonSports
group by star_rating
order by ProductsInRating desc limit 5;

```
hive> select star_rating, count(product_id) as ProductsInRatingfrom AmazonSports group by star_rating order by ProductsInRating desc limit 5;
FAILED: ParseException line 1:62 missing EOF at 'AmazonSports' near 'ProductsInRatingfrom'
hive> select star_rating, count(product_id) as ProductsInRating from AmazonSports group by star_rating order by ProductsInRating desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = meetveera_20200810165655_e20a880d-0c30-4cd5-bce1-ec83c824598a
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 8
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1597101041413_0014, Tracking URL = http://ubuntu:8088/proxy/application_1597101041413_0014/
Kill Command = /usr/local/bin/hadoop-2.10.0//bin/hadoop job  -kill job_1597101041413_0014
Hadoop job information for Stage-1: number of mappers: 8; number of reducers: 8
2020-08-10 16:57:02,891 Stage-1 map = 0%,  reduce = 0%
2020-08-10 16:57:30,211 Stage-1 map = 25%,  reduce = 0%, Cumulative CPU 25.05 sec
2020-08-10 16:57:31,249 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 39.28 sec
2020-08-10 16:57:32,294 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 42.08 sec
2020-08-10 16:57:50,179 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 53.85 sec
2020-08-10 16:57:54,355 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 63.53 sec
2020-08-10 16:58:04,833 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 68.59 sec
2020-08-10 16:58:06,883 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 73.58 sec
MapReduce Total cumulative CPU time: 1 minutes 13 seconds 580 msec
Ended Job = job_1597101041413_0014
```



Pig

data1 = load '/user/hadoop/AmazonSports.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);

Count of Daily Reviews

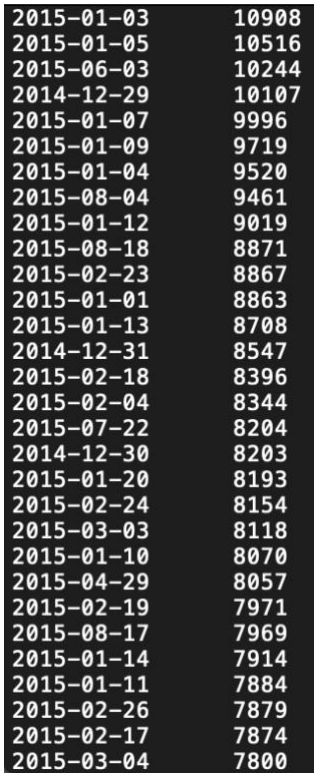
dailydata = GROUP data by review_date;

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

orderdata = ORDER daily_reviews BY count DESC;

store orderdata INTO '/pig1';

hadoop fs -ls /pig1



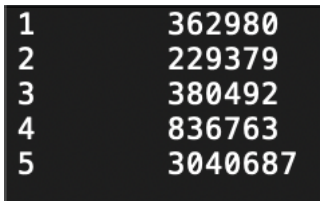
```
2015-01-03      10908
2015-01-05      10516
2015-06-03      10244
2014-12-29      10107
2015-01-07      9996
2015-01-09      9719
2015-01-04      9520
2015-08-04      9461
2015-01-12      9019
2015-08-18      8871
2015-02-23      8867
2015-01-01      8863
2015-01-13      8708
2014-12-31      8547
2015-02-18      8396
2015-02-04      8344
2015-07-22      8204
2014-12-30      8203
2015-01-20      8193
2015-02-24      8154
2015-03-03      8118
2015-01-10      8070
2015-04-29      8057
2015-02-19      7971
2015-08-17      7969
2015-01-14      7914
2015-01-11      7884
2015-02-26      7879
2015-02-17      7874
2015-03-04      7800
```

Reviews per Rating

product = GROUP data by star_rating;

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

store prod_count INTO '/pig2';



```
1       362980
2       229379
3       380492
4       836763
5       3040687
```

Conclusion

Hence, we have successfully implemented all the deliverables in Amazon Sports Reviews Data Analysis using Big Data Tools such as Hadoop MapReduce, Apache Hive and Apache Pig along with some data visualization using Tableau.

----------x----------