AMAZON CUSTOMER REVIEW ANALYSIS

Name: MARIA MATHEW
NUID: 001388143

Under the guidance of:
Prof. Yusuf Ozbek

CONTENTS

PROBLEM STATEMENT

a) **Identify the top 5 products in Amazon's "Personal Care Appliances Department" and "Digital Software Department" based on customer ratings.**

b) **Compare their performance using Apache Hadoop MapReduce, Apache Hive and Apache pig.**

DATASET

This project uses Amazon Customer Reviews Dataset which provide insight into opinions of customers on various Amazons products.
The url for the dataset is given below.

https://s3.amazonaws.com/amazon-reviews-pds/readme.html

DATA COLUMNS

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

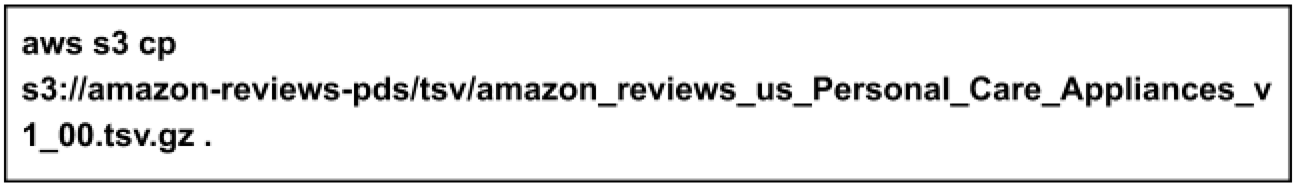
DATA FORMAT

The dataset is available in tsv format. The first line in each file is header.

DOWNLOADING DATA INTO LOCAL SYSTEM

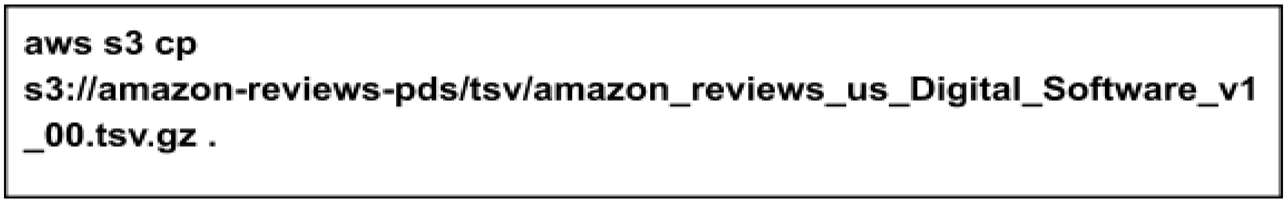
Install AWS CLI and execute the below commands in command prompt to download the dataset into local system. The combined file size is around 95 MB.

amazon_reviews_us_Personal_Care_Appliances_v1_00.tsv.gz



```
aws s3 cp
s3://amazon-reviews-pds/tsv/amazon_reviews_us_Personal_Care_Appliances_v
1_00.tsv.gz .
```

amazon_reviews_us_Digital_Software_v1_00.tsv.gz



```
aws s3 cp
s3://amazon-reviews-pds/tsv/amazon_reviews_us_Digital_Software_v1
_00.tsv.gz .
```

TOP 5 ANALYSIS USING HADOOP MAPREDUCE

MapReduce is a programming model for processing big data in parallel. This method include chaining of MapReduce jobs where the output of first MapReduce job is passed as an input of second MapReduce job.

PREREQUISITES

Step 1: Creating an input folder in HDFS

```
./hadoop fs -mkdir /AmazonInput
```

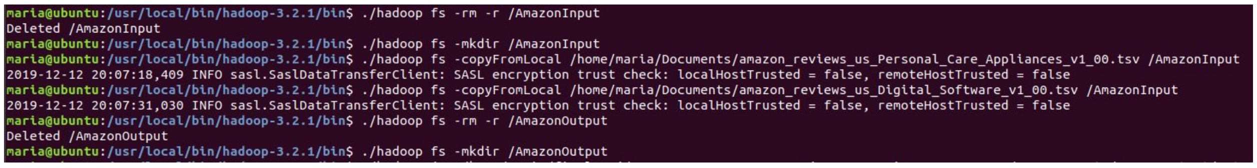
Step 2: Create an output folder in HDFS

```
./hadoop fs -mkdir /AmazonOutput
```

Step 3: Copy the data from local system into Hadoop file system

```
./hadoop fs -copyFromLocal
/home/maria/Documents/amazon_reviews_us_Personal_Care_Appliances_v1_00.tsv
/AmazonInput
```

```
./hadoop fs -copyFromLocal
/home/maria/Documents/amazon_reviews_us_Digital_Software_v1_00.tsv
/AmazonInput
```

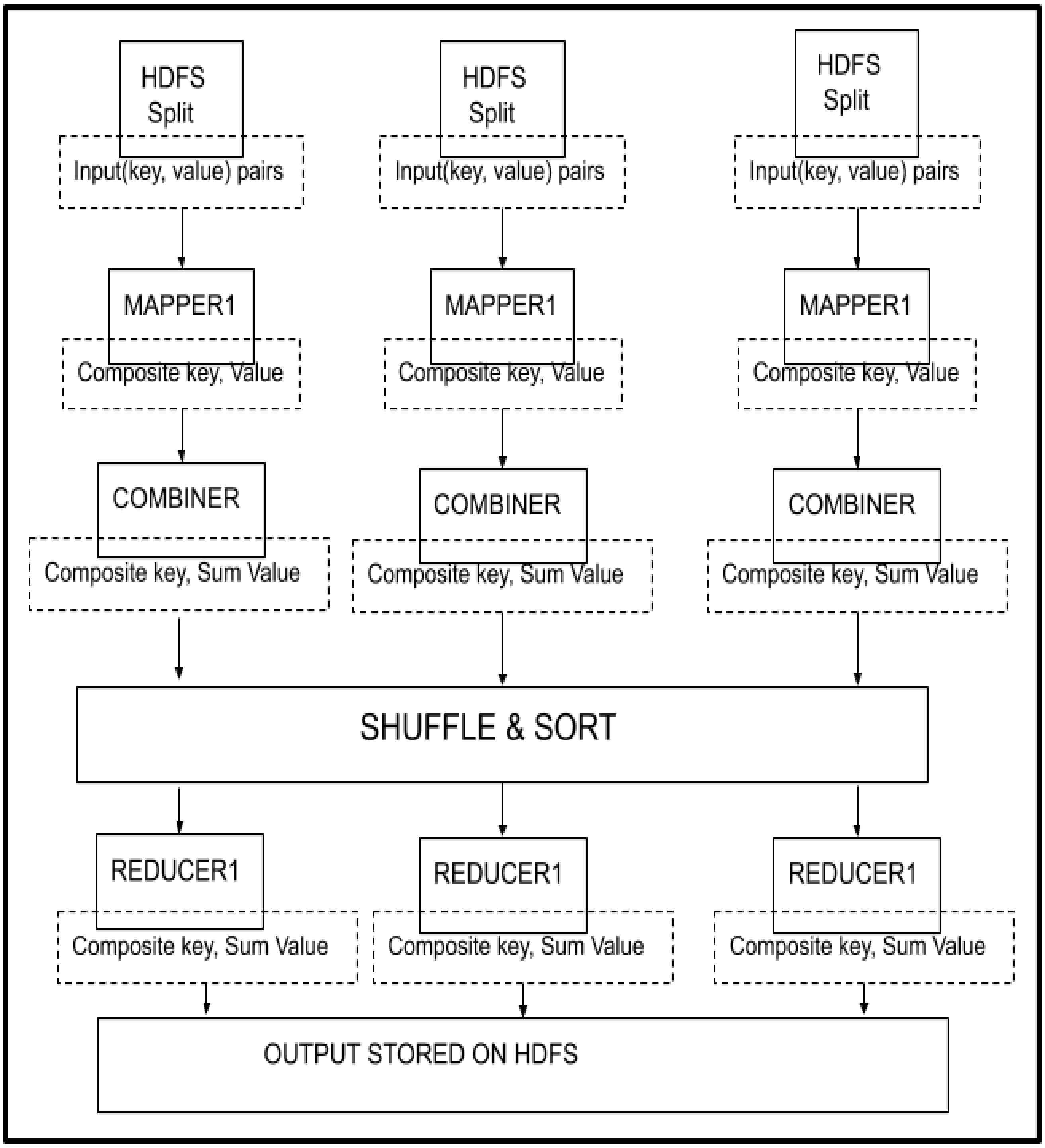


```
maria@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -rm -r /AmazonInput
Deleted /AmazonInput
maria@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -mkdir /AmazonInput
maria@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -copyFromLocal /home/maria/Documents/amazon_reviews_us_Personal_Care_Appliances_v1_00.tsv /AmazonInput
2019-12-12 20:07:18,409 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
maria@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -copyFromLocal /home/maria/Documents/amazon_reviews_us_Digital_Software_v1_00.tsv /AmazonInput
2019-12-12 20:07:31,030 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
maria@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -rm -r /AmazonOutput
Deleted /AmazonOutput
maria@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -mkdir /AmazonOutput
```

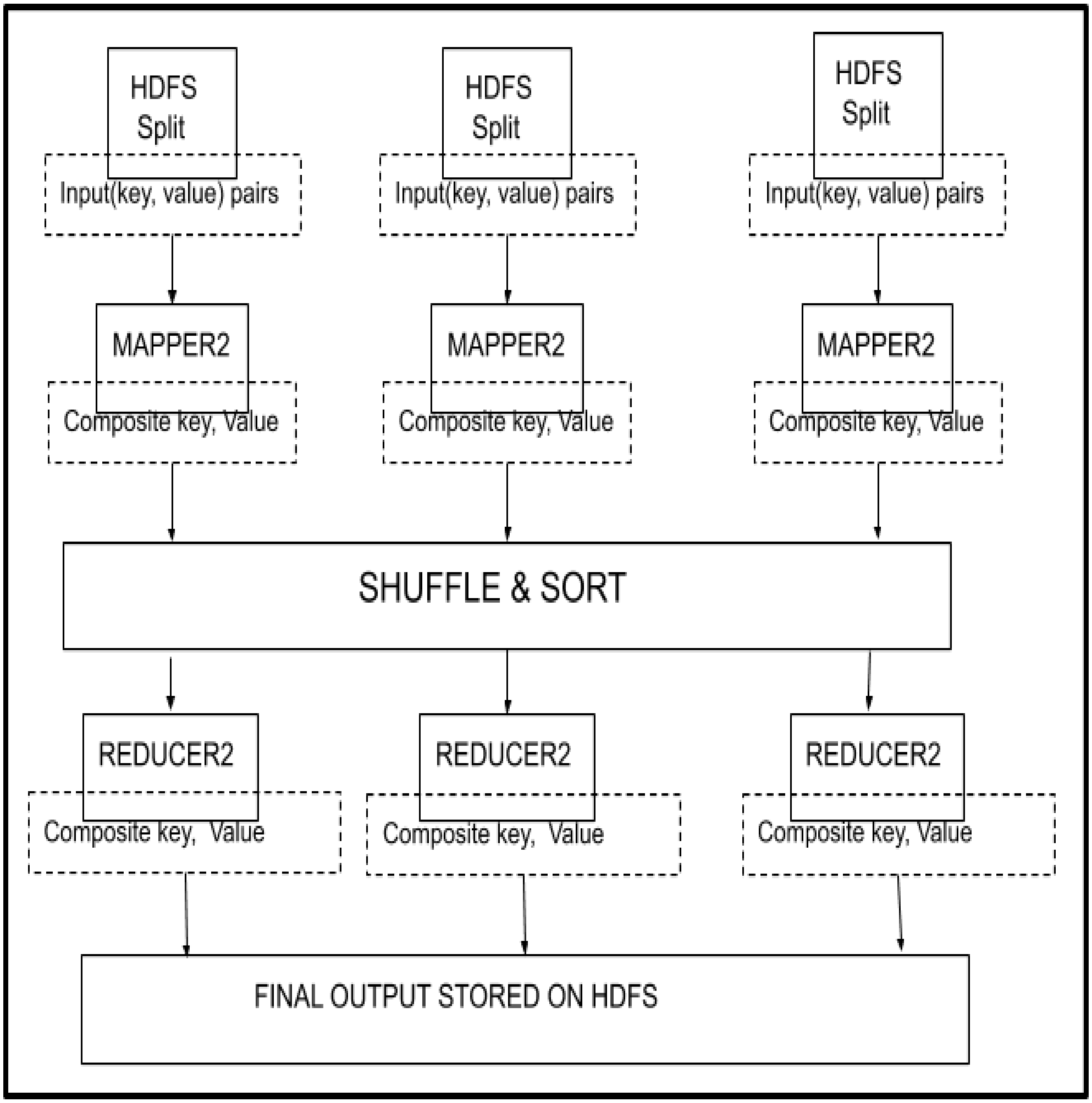
JAR File

```
./hadoop jar /home/maria/finalProj/Amazon8-0.0.1-SNAPSHOT.jar
com.maria.Amazon8.App /AmazonInput /AmazonOutput/out /AmazonOutput/Top10/out
```

DATAFLOW PART A



DATA FLOW PART B



App.java - This is the driver class which performs MapReduce chaining.

ProductMapper.java - This class reads the input file. Here textinputformat is used. Keys are the position in the file, and values are the line of text. The value is split based on tab delimiter. The output key of this class is a composite key which is Product ID and Product Category. The output value is ratings.

CompositeKeyWritable.java - This class creates a composite key for the mapper class ProductMapper.java. The composite key is a combination of Product ID and Product Category.

NaturalKeyPartitioner.java - This class performs partition on composite key based on Product ID.

NaturalKeyGroupComparator.java -This class groups the composite key based on Product ID.

ProductReducer.java - This class calculates the sum of values for each key generated from ProductMapper.java. It is also reused as combiner class to calculate local sum of values of each key generated from each mapper. The output is saved in HDFS.

TopProductMapper.java - This class reads the intermediate output generated by ProductReducer.java. Here textinputformat is used. Keys are the position in the file, and values are the line of text. The value is split based on space delimiter. The output key of this class is a composite key which is Product Category and sum of ratings. The output value is Product ID.

CompositeKeyWritableTop.java - This class creates a composite key for the mapper class TopProductMapper.java. The composite key is a combination of Product Category and sum of ratings .

NaturalKeyPartitionerTop.java - This class performs partition on composite key based on Product Category.

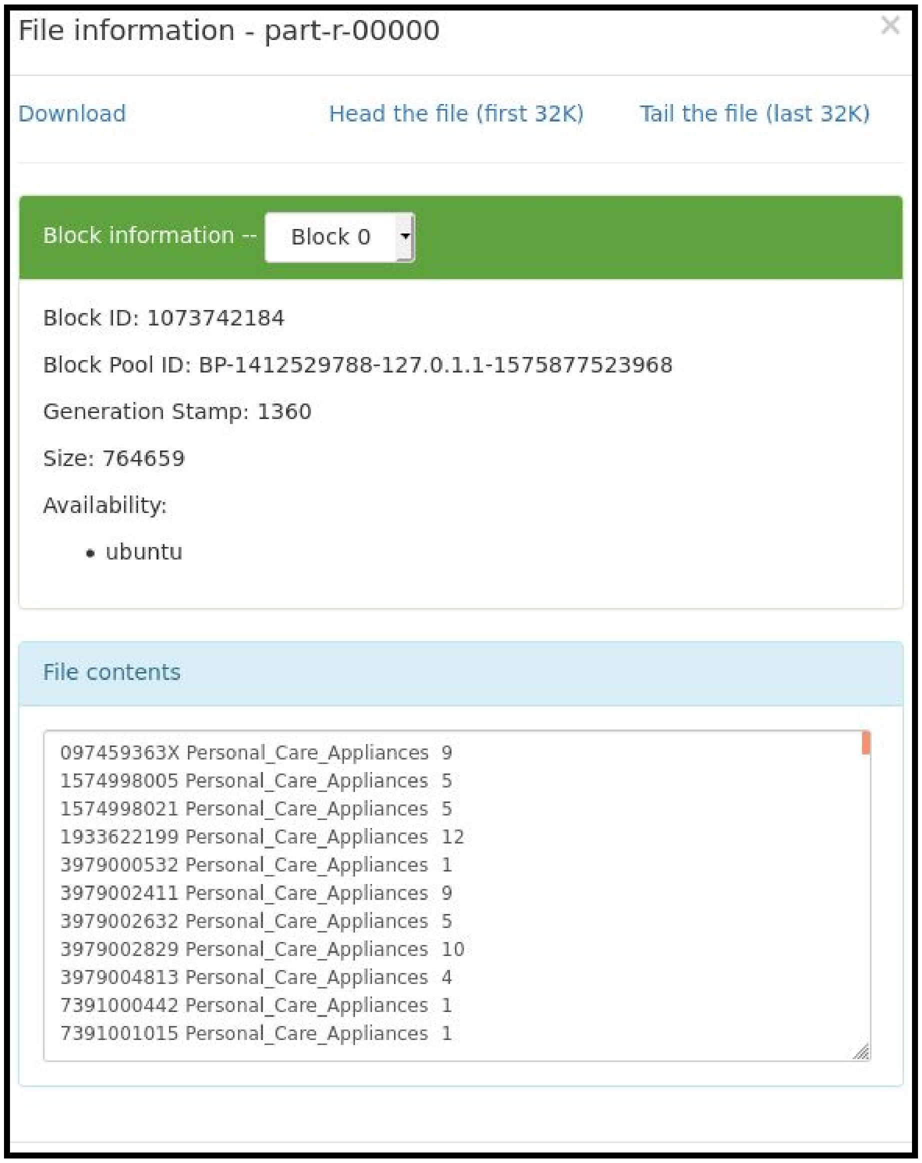
NaturalKeyGroupComparatorTop.java - This class groups the composite key based on rating.

SecondarySortComparatorTop.java - This class performs sorting on composite key based on product rating. Sorting is done in descending order.

TopProductReducer.java - This class identify the top 5 product ID from Digital Software department and Personal Care Appliances department based on ratings. Rank is generated for each product ID.

Output of first MapReduce job

The output format of first MapReduce job:- *Product ID: Product Category: Rating*

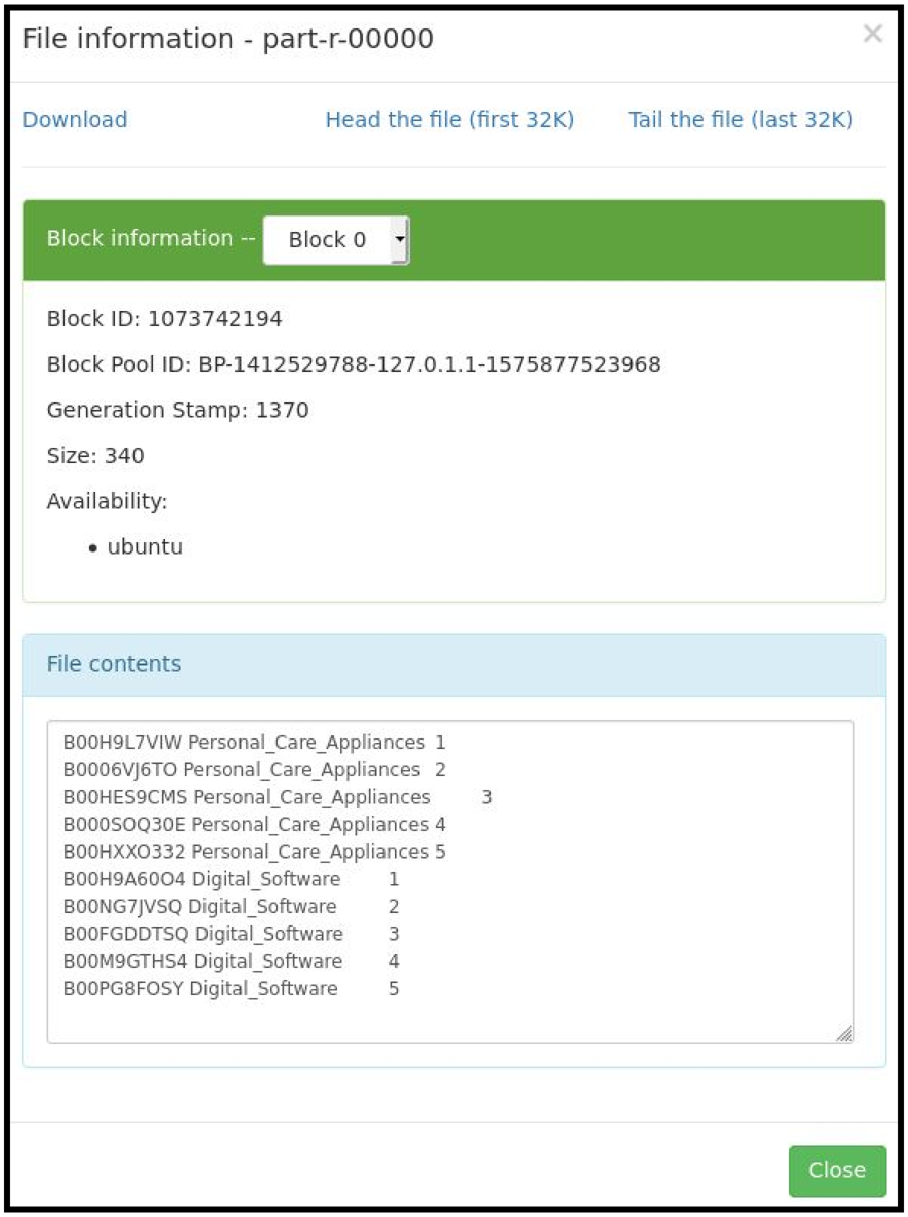


```
Block ID: 1073742184
Block Pool ID: BP-1412529788-127.0.1.1-1575877523968
Generation Stamp: 1360
Size: 764659
Availability:
  • ubuntu
```

```
097459363X Personal_Care_Appliances  9
1574998005 Personal_Care_Appliances  5
1574998021 Personal_Care_Appliances  5
1933622199 Personal_Care_Appliances  12
3979000532 Personal_Care_Appliances  1
3979002411 Personal_Care_Appliances  9
3979002632 Personal_Care_Appliances  5
3979002829 Personal_Care_Appliances  10
3979004813 Personal_Care_Appliances  4
7391000442 Personal_Care_Appliances  1
7391001015 Personal_Care_Appliances  1
```

Output of second MapReduce job

The output format of second MapReduce job:- *Product ID: Product Category: Rank*


File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0

Block ID: 1073742194

Block Pool ID: BP-1412529788-127.0.1.1-1575877523968

Generation Stamp: 1370

Size: 340

Availability:

- ubuntu

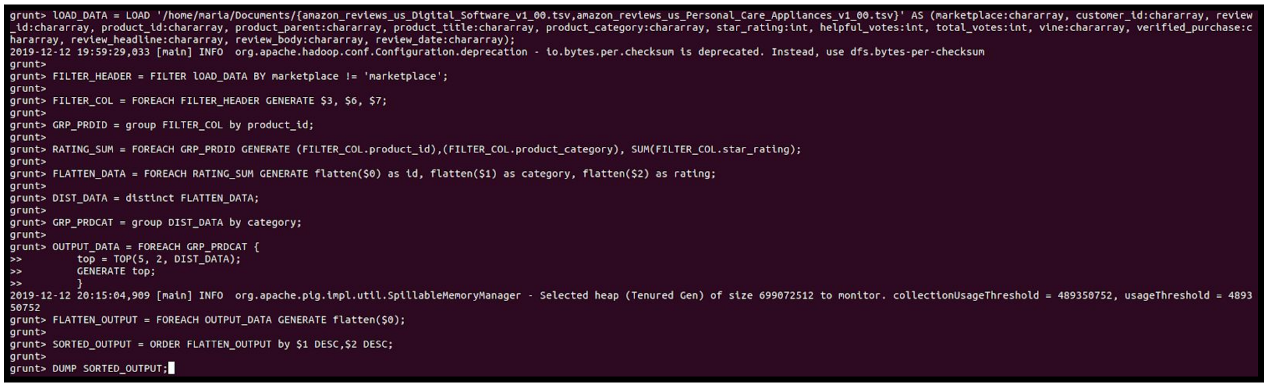
```
B00H9L7VIW Personal_Care_Appliances 1
B0006VJ6TO Personal_Care_Appliances 2
B00HES9CMS Personal_Care_Appliances     3
B000SOQ30E Personal_Care_Appliances 4
B00HXXO332 Personal_Care_Appliances 5
B00H9A60O4 Digital_Software     1
B00NG7JVSQ Digital_Software     2
B00FGDDTSQ Digital_Software     3
B00M9GTHS4 Digital_Software     4
B00PG8FOSY Digital_Software     5
```

TOP 5 ANALYSIS USING APACHE PIG

Apache Pig is a high-level platform for creating programs that run on Apache Hadoop. The language for this platform is called Pig Latin.

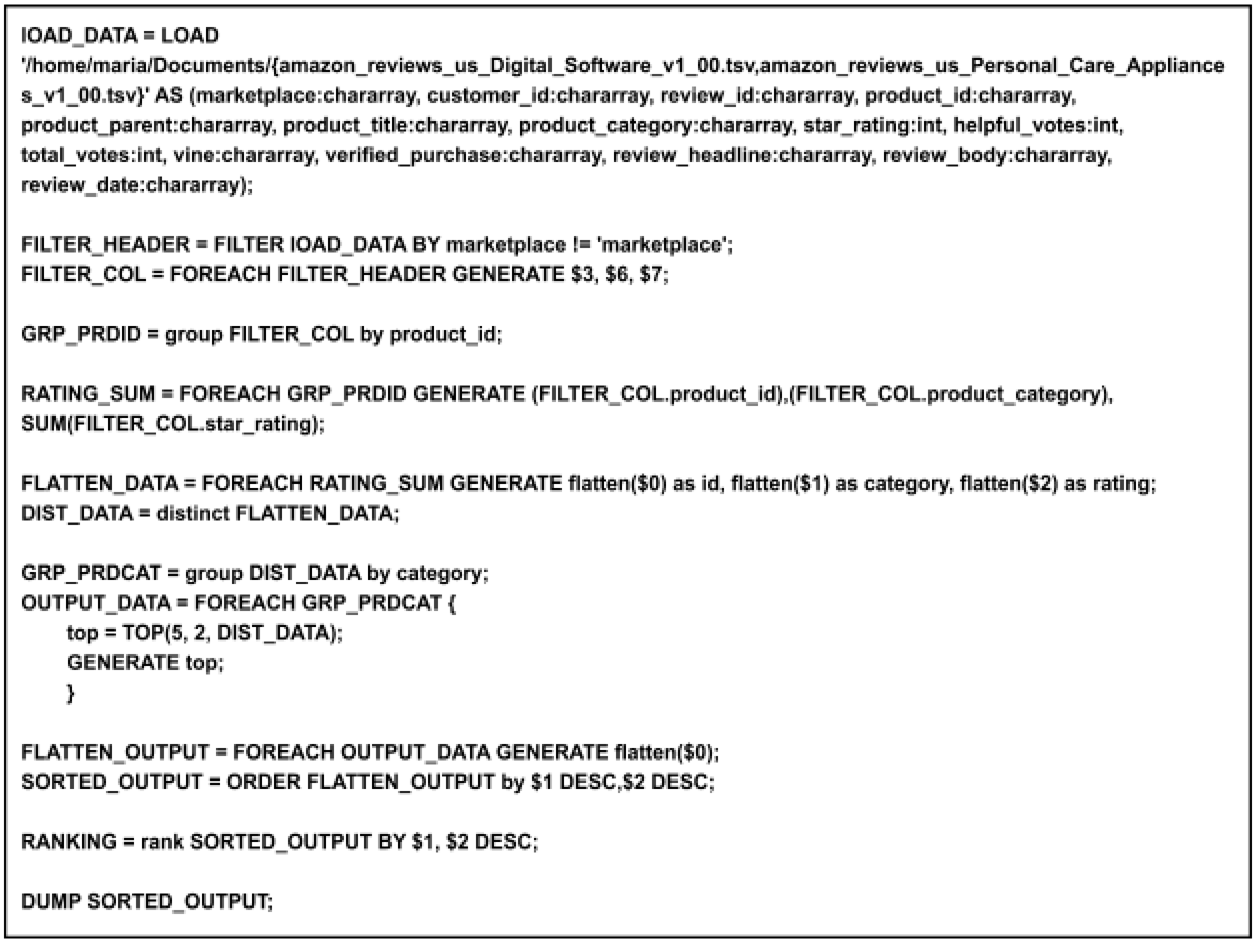
DATA FLOW

Step 1: Load data into Pig relation
Step 2: Remove header from loaded data
Step 3: Filter out the columns and keep only the required column which are Product ID, Product Category and Product Rating.
Step 4: Group the data based on Product ID
Step 5: Calculate the sum of Ratings for each product ID
Step 6: Flatten the bag
Step 7: Filter out duplicate data
Step 8: Group data based on Product Category
Step 9: Calculate the top 5 products.
Step 10: Flatten the bag
Step 11: Sort the data based on product category and Ratings in descending order
Step 12: Display the output



```
grunt> lOAD_DATA = LOAD '/home/maria/Documents/{amazon_reviews_us_Digital_Software_v1_00.tsv,amazon_reviews_us_Personal_Care_Appliances_v1_00.tsv}' AS (marketplace:chararray, customer_id:chararray, review
_id:chararray, product_id:chararray, product_parent:chararray, product_title:chararray, product_category:chararray, star_rating:int, helpful_votes:int, total_votes:int, vine:chararray, verified_purchase:c
hararray, review_headline:chararray, review_body:chararray, review_date:chararray);
2019-12-12 19:59:29,033 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt>
grunt> FILTER_HEADER = FILTER lOAD_DATA BY marketplace != 'marketplace';
grunt>
grunt> FILTER_COL = FOREACH FILTER_HEADER GENERATE $3, $6, $7;
grunt>
grunt> GRP_PRDID = group FILTER_COL by product_id;
grunt>
grunt> RATING_SUM = FOREACH GRP_PRDID GENERATE (FILTER_COL.product_id),(FILTER_COL.product_category), SUM(FILTER_COL.star_rating);
grunt>
grunt> FLATTEN_DATA = FOREACH RATING_SUM GENERATE flatten($0) as id, flatten($1) as category, flatten($2) as rating;
grunt>
grunt> DIST_DATA = distinct FLATTEN_DATA;
grunt>
grunt> GRP_PRDCAT = group DIST_DATA by category;
grunt>
grunt> OUTPUT_DATA = FOREACH GRP_PRDCAT {
>>         top = TOP(5, 2, DIST_DATA);
>>         GENERATE top;
>>         }
2019-12-12 20:15:04,909 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (Tenured Gen) of size 699072512 to monitor. collectionUsageThreshold = 489350752, usageThreshold = 4893
50752
grunt> FLATTEN_OUTPUT = FOREACH OUTPUT_DATA GENERATE flatten($0);
grunt>
grunt> SORTED_OUTPUT = ORDER FLATTEN_OUTPUT by $1 DESC,$2 DESC;
grunt>
grunt> DUMP SORTED_OUTPUT;
```

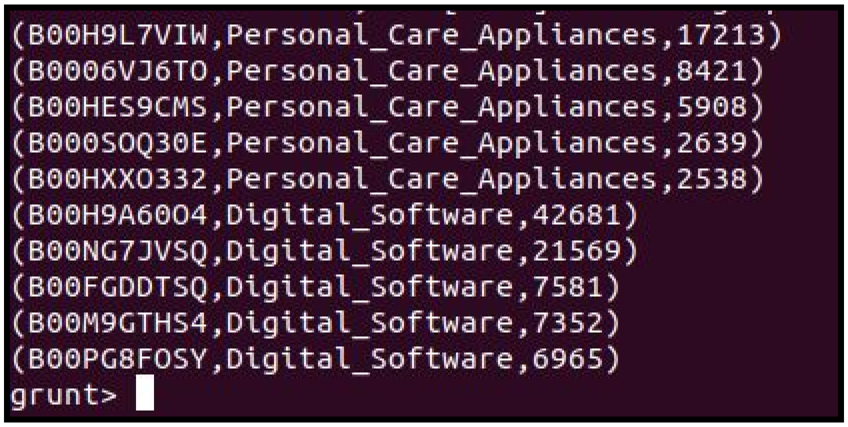
PIG SCRIPT



```
lOAD_DATA = LOAD
'/home/maria/Documents/{amazon_reviews_us_Digital_Software_v1_00.tsv,amazon_reviews_us_Personal_Care_Appliance
s_v1_00.tsv}' AS (marketplace:chararray, customer_id:chararray, review_id:chararray, product_id:chararray,
product_parent:chararray, product_title:chararray, product_category:chararray, star_rating:int, helpful_votes:int,
total_votes:int, vine:chararray, verified_purchase:chararray, review_headline:chararray, review_body:chararray,
review_date:chararray);

FILTER_HEADER = FILTER lOAD_DATA BY marketplace != 'marketplace';
FILTER_COL = FOREACH FILTER_HEADER GENERATE $3, $6, $7;

GRP_PRDID = group FILTER_COL by product_id;

RATING_SUM = FOREACH GRP_PRDID GENERATE (FILTER_COL.product_id),(FILTER_COL.product_category),
SUM(FILTER_COL.star_rating);

FLATTEN_DATA = FOREACH RATING_SUM GENERATE flatten($0) as id, flatten($1) as category, flatten($2) as rating;
DIST_DATA = distinct FLATTEN_DATA;

GRP_PRDCAT = group DIST_DATA by category;
OUTPUT_DATA = FOREACH GRP_PRDCAT {
      top = TOP(5, 2, DIST_DATA);
      GENERATE top;
      }

FLATTEN_OUTPUT = FOREACH OUTPUT_DATA GENERATE flatten($0);
SORTED_OUTPUT = ORDER FLATTEN_OUTPUT by $1 DESC,$2 DESC;

RANKING = rank SORTED_OUTPUT BY $1, $2 DESC;

DUMP SORTED_OUTPUT;
```

FINAL OUTPUT

The output is generated based on total ratings of each Product ID sorted in descending order. The output format is *Product ID:Product Category: total Ratings.*



```
(B00H9L7VIW,Personal_Care_Appliances,17213)
(B0006VJ6TO,Personal_Care_Appliances,8421)
(B00HES9CMS,Personal_Care_Appliances,5908)
(B000SOQ30E,Personal_Care_Appliances,2639)
(B00HXXO332,Personal_Care_Appliances,2538)
(B00H9A60O4,Digital_Software,42681)
(B00NG7JVSQ,Digital_Software,21569)
(B00FGDDTSQ,Digital_Software,7581)
(B00M9GTHS4,Digital_Software,7352)
(B00PG8FOSY,Digital_Software,6965)
grunt>
```

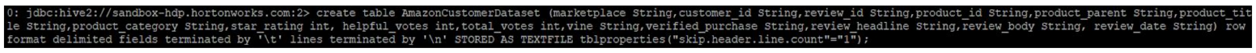
TOP 5 ANALYSIS USING APACHE HIVE

Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. Hive gives a SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop

DATA FLOW

Step 1: Create table AmazonCustomerDataset

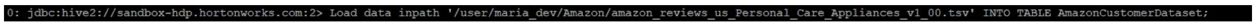
```
create table AmazonCustomerDataset (marketplace String,customer_id
String,review_id String,product_id String,product_parent
String,product_title String,product_category String,star_rating int,
helpful_votes int,total_votes int,vine String,verified_purchase
String,review_headline String,review_body String, review_date String)
row format delimited fields terminated by '\t' lines terminated by '\n'
STORED AS TEXTFILE tblproperties("skip.header.line.count"="1");
```



```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> create table AmazonCustomerDataset (marketplace String,customer_id String,review_id String,product_id String,product_parent String,product_tit
le String,product_category String,star_rating int, helpful_votes int,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' STORED AS TEXTFILE tblproperties("skip.header.line.count"="1");
```

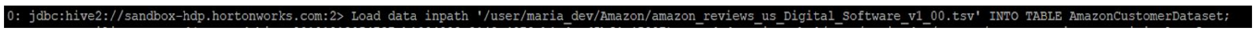
Step 2: Load data into AmazonCustomerDataset

```
Load data inpath
'/user/maria_dev/Amazon/amazon_reviews_us_Personal_Care_Appliances_v1
_00.tsv' INTO TABLE AmazonCustomerDataset;
```



```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> Load data inpath '/user/maria_dev/Amazon/amazon_reviews_us_Personal_Care_Appliances_v1_00.tsv' INTO TABLE AmazonCustomerDataset;
```

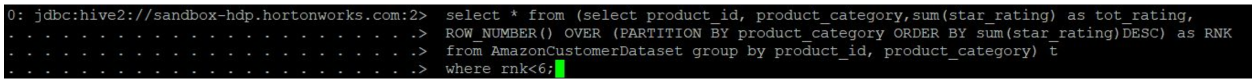
```
Load data inpath
'/user/maria_dev/Amazon/amazon_reviews_us_Digital_Software_v1_00.tsv' INTO
TABLE AmazonCustomerDataset;
```



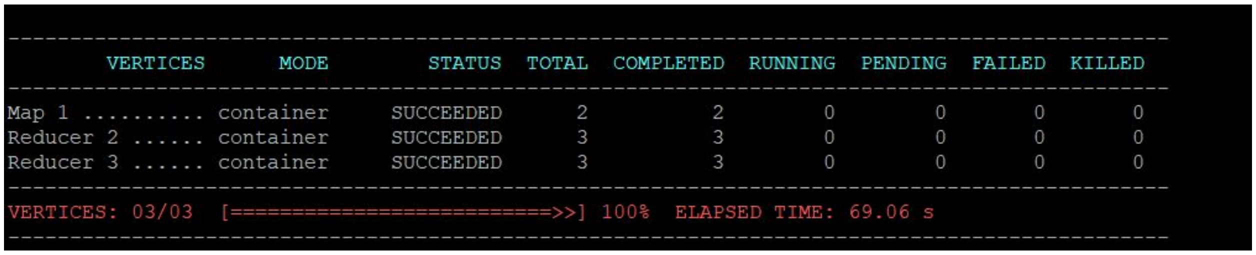
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> Load data inpath '/user/maria_dev/Amazon/amazon_reviews_us_Digital_Software_v1_00.tsv' INTO TABLE AmazonCustomerDataset;
```

Step 3: Selecting top 5 product Id based on total ratings

```
 select * from (select product_id, product_category,sum(star_rating) as tot_rating,
 ROW_NUMBER() OVER (PARTITION BY product_category ORDER BY
sum(star_rating)DESC) as RNK
 from AmazonCustomerDataset group by product_id, product_category) t
 where rnk<6;
```



```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2>  select * from (select product_id, product_category,sum(star_rating) as tot_rating,
. . . . . . . . . . . . . . . . . . . . . . .>  ROW_NUMBER() OVER (PARTITION BY product_category ORDER BY sum(star_rating)DESC) as RNK
. . . . . . . . . . . . . . . . . . . . . . .>  from AmazonCustomerDataset group by product_id, product_category) t
. . . . . . . . . . . . . . . . . . . . . . .>  where rnk<6;
```

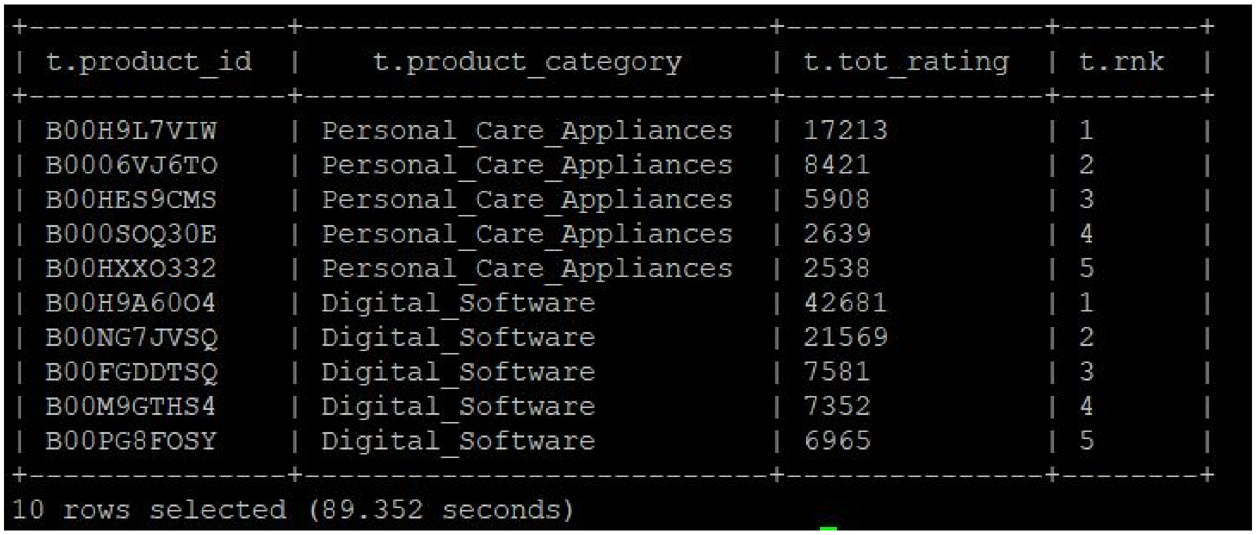


```
----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      3          3        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      3          3        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 69.06 s
----------------------------------------------------------------------------------------------
```

FINAL OUTPUT

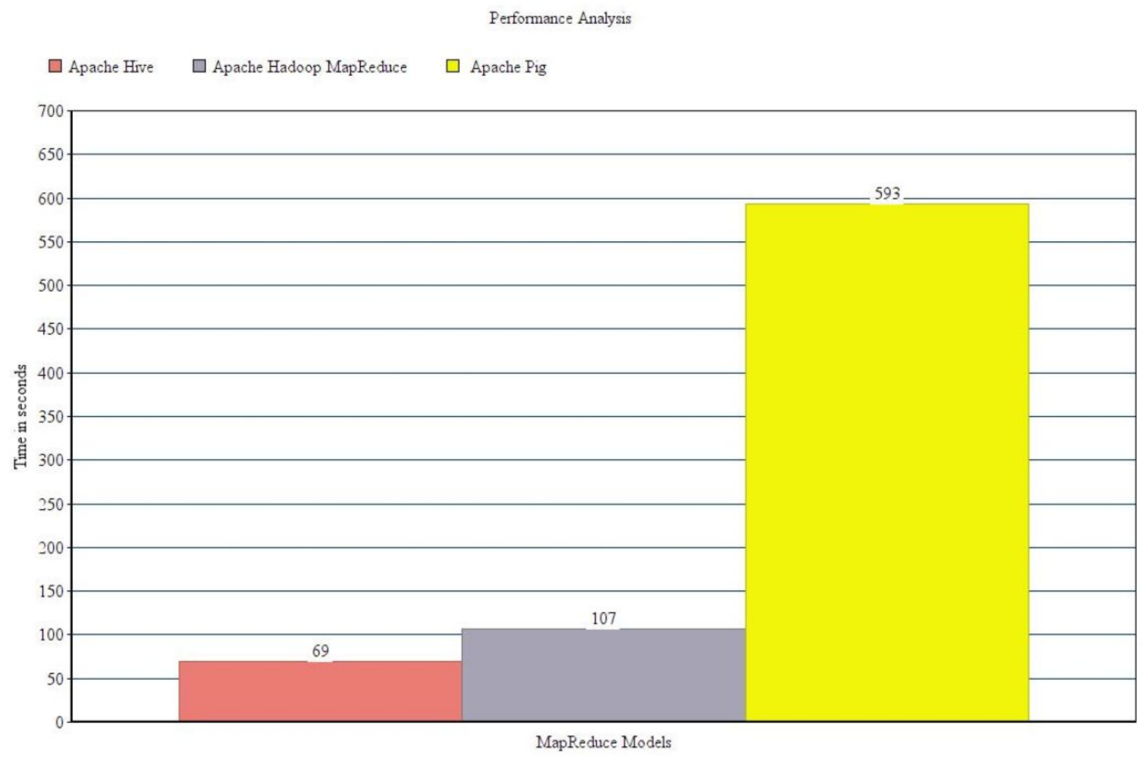
Hive query is executed on the dataset to identify the top 5 products from each department.

The output format is:- *ProductID: Product Category: Total Rating: Rank of ProductID*



```
+--------------+---------------------------+--------------+--------+
| t.product_id |    t.product_category     | t.tot_rating | t.rnk  |
+--------------+---------------------------+--------------+--------+
| B00H9L7VIW   | Personal_Care_Appliances  | 17213        | 1      |
| B0006VJ6TO   | Personal_Care_Appliances  | 8421         | 2      |
| B00HES9CMS   | Personal_Care_Appliances  | 5908         | 3      |
| B000SOQ30E   | Personal_Care_Appliances  | 2639         | 4      |
| B00HXXO332   | Personal_Care_Appliances  | 2538         | 5      |
| B00H9A60O4   | Digital_Software          | 42681        | 1      |
| B00NG7JVSQ   | Digital_Software          | 21569        | 2      |
| B00FGDDTSQ   | Digital_Software          | 7581         | 3      |
| B00M9GTHS4   | Digital_Software          | 7352         | 4      |
| B00PG8FOSY   | Digital_Software          | 6965         | 5      |
+--------------+---------------------------+--------------+--------+
10 rows selected (89.352 seconds)
```

COMPARISON BETWEEN HADOOP MAPREDUCE Vs APACHE HIVE Vs APACHE PIG



APPENDIX

JAVA CODE FOR MAPREDUCE

****

App.java

****

```
package com.maria.Amazon8;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class App {
	public static void main(String[] args) {
		Configuration conf = new Configuration();
		try {
			Job job = Job.getInstance(conf, " TOP 5 PRODUCTS FROM 'DIGITAL SOFTWARE' and
'PERSONAL CARE' DEPARTMENT");

			// set driver class
			job.setJarByClass(App.class);

			// set Natural Key PartitionerClass
			job.setPartitionerClass(NaturalKeyPartitioner.class);
			// set Natural Key GroupingComparatorClass
			job.setGroupingComparatorClass(NaturalKeyGroupComparator.class);



			// set MapperClass
			job.setMapperClass(ProductMapper.class);
			// set CombinerClass
			job.setCombinerClass(ProductReducer.class);
			// set ReducerClass
			job.setReducerClass(ProductReducer.class);

			// set InputFormatClass
			job.setInputFormatClass(TextInputFormat.class);
			// set OutputFormatClass
			job.setOutputFormatClass(TextOutputFormat.class);

			// set OutputKeyClass
			job.setOutputKeyClass(CompositeKeyWritable.class);
			// set OutputValueClass
			job.setOutputValueClass(IntWritable.class);
			// set path
			FileInputFormat.addInputPath(job, new Path(args[0]));
			FileOutputFormat.setOutputPath(job, new Path(args[1]));
			try {
				job.waitForCompletion(true);
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		} catch (IOException e) {

			e.printStackTrace();
		}

		Configuration conf1 = new Configuration();
		try {
			Job job1 = Job.getInstance(conf1, " TOP 5 PRODUCTS FROM 'DIGITAL SOFTWARE' and
'PERSONAL CARE' DEPARTMENT");

			// set driver class
			job1.setJarByClass(App.class);

			// set Natural Key PartitionerClass
			job1.setPartitionerClass(NaturalKeyPartitionerTop.class);
			// set Natural Key GroupingComparatorClass
			job1.setGroupingComparatorClass(NaturalKeyGroupComparatorTop.class);
			// set SortComparatorClass
			job1.setSortComparatorClass(SecondarySortComparatorTop.class);

			// set MapperClass
			job1.setMapperClass(TopProductMapper.class);
			// set ReducerClass
			job1.setReducerClass(TopProductReducer.class);

			// set InputFormatClass
			job1.setInputFormatClass(TextInputFormat.class);
			// set OutputFormatClass
			job1.setOutputFormatClass(TextOutputFormat.class);

			// set OutputKeyClass
			job1.setOutputKeyClass(CompositeKeyWritableTop.class);
			// set setOutputValueClass
			job1.setOutputValueClass(Text.class);

			// set path
			FileInputFormat.addInputPath(job1, new Path(args[1]));
			FileOutputFormat.setOutputPath(job1, new Path(args[2]));
			try {
				System.exit(job1.waitForCompletion(true) ? 0 : 1);
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		} catch (IOException e) {
```

```
                                e.printStackTrace();
                        }
                }
        }
```

****

CompositeKeyWritable.java

****

```
package com.maria.Amazon8;

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

import org.apache.hadoop.io.WritableComparable;

public class CompositeKeyWritable implements WritableComparable {

        public CompositeKeyWritable() {

        }

        public CompositeKeyWritable(String product_ID, String product_Category) {
                super();
                PRD_ID = product_ID;
                PRD_CAT = product_Category;
        }

        String PRD_ID;
        String PRD_CAT;

        // get Product ID
        public String getProductID() {
                return PRD_ID;
        }

        // set Product ID
        public void setProductID(String product_id) {
                PRD_ID = product_id;
        }

        // get Product Category
        public String getProductCategory() {
                return PRD_CAT;
        }

        // set Product Category
        public void setProductCategory(String product_cat) {
                PRD_CAT = product_cat;
        }

        public void readFields(DataInput in) throws IOException {
```

```
		PRD_ID = in.readUTF();
		PRD_CAT = in.readUTF();
	}

	public void write(DataOutput out) throws IOException {
		out.writeUTF(PRD_ID);
		out.writeUTF(PRD_CAT);

	}

	@Override
	public String toString() {

		return PRD_ID + " " + PRD_CAT;
	}

	public int compareTo(Object o) {
		CompositeKeyWritable ck = (CompositeKeyWritable) o;
		String thisvalue = this.getProductID();
		String othervalue = ck.getProductID();
		int result = thisvalue.compareTo(othervalue);
		return (result < 0 ? -1 : (result == 0 ? 0 : 1));

	}

}
```

****

NaturalKeyPartitioner.java

****

```
package com.maria.Amazon8;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class NaturalKeyPartitioner extends Partitioner<CompositeKeyWritable, IntWritable> {

	// partitioning based on Product ID
	public int getPartition(CompositeKeyWritable key, IntWritable value, int numPartitions) {

		return key.getProductID().hashCode() % numPartitions;
	}

}
```

****

NaturalKeyGroupComparator.java

****

```
package com.maria.Amazon8;

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

public class NaturalKeyGroupComparator extends WritableComparator {

        public NaturalKeyGroupComparator() {
                super(CompositeKeyWritable.class, true);
        }

        // grouping based on Product ID
        public int compare(WritableComparable a, WritableComparable b) {
                CompositeKeyWritable ck1 = (CompositeKeyWritable) a;
                CompositeKeyWritable ck2 = (CompositeKeyWritable) b;

                int result = ck1.getProductID().compareTo(ck2.getProductID());

                return result;
        }

}
```

****

ProductMapper.java

****

```
package com.maria.Amazon8;

import java.io.IOException;

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

public class ProductMapper extends Mapper<LongWritable, Text, CompositeKeyWritable, IntWritable> {

        // The map function process each line of tsv file and emit a composite
        // key('product_id' and 'product_category') and rating of each Product ID.
        @Override
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

                // Convert text to string
                String str = value.toString();
                // Skipping header
                if (!(value.toString().contains("marketplace"))) {
                    String[] word = str.split("\\t");

                    if (word.length == 15) {

                              IntWritable rating = new IntWritable();

                              rating.set(Integer.parseInt(word[7]));

                              // Compositekey consist of 'product_id' and 'product_category'
                              CompositeKeyWritable obj = new CompositeKeyWritable(word[3], word[6]);

                              context.write(obj, rating);
                    }
          }

     }
}
```

****

ProductReducer.java

****

```
package com.maria.Amazon8;

import java.io.IOException;

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

public class ProductReducer extends Reducer<CompositeKeyWritable, IntWritable, CompositeKeyWritable, IntWritable> {

     // The reduce function emits a composite key('product_id' and
     // 'product_category') and sum of ratings of each Product ID.
     @Override
     public void reduce(CompositeKeyWritable key, Iterable<IntWritable> values, Context context)
                    throws IOException, InterruptedException {

          int sum = 0;

          for (IntWritable i : values) {

                    sum += i.get();

          }

          IntWritable count = new IntWritable(sum);

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

****

CompositeKeyWritableTop.java

****

```
package com.maria.Amazon8;

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

import org.apache.hadoop.io.WritableComparable;

public class CompositeKeyWritableTop implements WritableComparable {

	public CompositeKeyWritableTop() {

	}

	public CompositeKeyWritableTop(String product_Category, String Total_Ratings) {
		super();
		RATINGS = Total_Ratings;
		PRD_CAT = product_Category;
	}

	String PRD_CAT;
	String RATINGS;

	// get Rating
	public String getRating() {
		return RATINGS;
	}

	// set Rating
	public void setRating(String Total_rating) {
		RATINGS = Total_rating;
	}

	// get Product Category
	public String getProductCategory() {
		return PRD_CAT;
	}

	// set Product Category
	public void setProductCategory(String product_cat) {
		PRD_CAT = product_cat;
	}

	public void readFields(DataInput in) throws IOException {

		RATINGS = in.readUTF();
		PRD_CAT = in.readUTF();
	}

	public void write(DataOutput out) throws IOException {
		out.writeUTF(RATINGS);
		out.writeUTF(PRD_CAT);
	}

	@Override
	public String toString() {

		return PRD_CAT + " " + RATINGS;
	}

	public int compareTo(Object o) {
		CompositeKeyWritableTop ck = (CompositeKeyWritableTop) o;
		String thisvalue = this.getProductCategory();
		String othervalue = ck.getProductCategory();
		int result = thisvalue.compareTo(othervalue);
		return (result < 0 ? -1 : (result == 0 ? 0 : 1));

	}

}
```

****

NaturalKeyPartitionerTop.java

****

```
package com.maria.Amazon8;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class NaturalKeyPartitionerTop extends Partitioner<CompositeKeyWritableTop, IntWritable> {

	// partitioning based on Product Category
	public int getPartition(CompositeKeyWritableTop key, IntWritable value, int numPartitions) {

		return key.getProductCategory().hashCode() % numPartitions;
	}

}
```

****

NaturalKeyGroupComparatorTop.java

****

```
package com.maria.Amazon8;

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

public class NaturalKeyGroupComparatorTop extends WritableComparator {

	public NaturalKeyGroupComparatorTop() {
		super(CompositeKeyWritableTop.class, true);
	}

	// grouping based on Product Rating
	public int compare(WritableComparable a, WritableComparable b) {
		CompositeKeyWritableTop ck1 = (CompositeKeyWritableTop) a;
		CompositeKeyWritableTop ck2 = (CompositeKeyWritableTop) b;

		Integer num1 = Integer.parseInt(ck1.getRating());
		Integer num2 = Integer.parseInt(ck2.getRating());

		int result = num1.compareTo(num2);

		return result;
	}

}
```

****

SecondarySortComparatorTop.java

****

```
package com.maria.Amazon8;

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

public class SecondarySortComparatorTop extends WritableComparator {

	public SecondarySortComparatorTop() {
		super(CompositeKeyWritableTop.class, true);
	}

	// The composite key is sorted in descending order based on total rating
	public int compare(WritableComparable a, WritableComparable b) {

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

		Integer num1 = Integer.parseInt(ck1.getRating());
		Integer num2 = Integer.parseInt(ck2.getRating());

		int result = -1 * num1.compareTo(num2);

		return result;
	}
}
```

****

TopProductMapper.java

****

```
package com.maria.Amazon8;

import java.io.IOException;
import java.util.TreeMap;

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

public class TopProductMapper extends Mapper<LongWritable, Text, CompositeKeyWritableTop, Text> {

	// tree map for finding local top5 products from each mapper in each department
	private TreeMap<Long, String> tmap_PersonalCare;
	private TreeMap<Long, String> tmap_Software;

	// Initializing two tree map. One for personal care department and another for
	// Digital Software department
	@Override
	public void setup(Context context) throws IOException, InterruptedException {

		tmap_PersonalCare = new TreeMap<Long, String>();
		tmap_Software = new TreeMap<Long, String>();

	}

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

		String str = value.toString();

		String[] word = str.split("\\s+");
		if (word.length == 3) {

			// Finding local top 5 from each mapper in Personal Care Appliance department
			if (word[1].equals("Personal_Care_Appliances")) {

				long total_rating = Long.parseLong(word[2]);

				if (!tmap_PersonalCare.containsKey(total_rating)) {

					tmap_PersonalCare.put(total_rating, word[0]);

				} else {

					String product_list = tmap_PersonalCare.get(total_rating);
					product_list += " " + word[0];
					tmap_PersonalCare.put(total_rating, product_list);
				}

				if (tmap_PersonalCare.size() > 5) {
```

```
                tmap_PersonalCare.remove(tmap_PersonalCare.firstKey());
            }

        }
        // Finding local top 5 from each mapper in Digital Software department
        else if (word[1].equals("Digital_Software")) {

            long total_rating = Long.parseLong(word[2]);

            if (!tmap_Software.containsKey(total_rating)) {

                tmap_Software.put(total_rating, word[0]);

            } else {

                String product_list = tmap_Software.get(total_rating);
                product_list += " " + word[0];
                tmap_Software.put(total_rating, product_list);

            }

            if (tmap_Software.size() > 5) {

                tmap_Software.remove(tmap_Software.firstKey());
            }

        }
    }

}

@Override
public void cleanup(Context context) throws IOException, InterruptedException {
    Text products = new Text();

    // Emitting local top 5 key value from each mapper in Personal_Care_Appliances
    // department
    int counter = 0;


    for (int ptr = 5; ptr > 0; ptr--) {

        if (tmap_PersonalCare.size() == 0) {
            break;
        }

        long mykey = tmap_PersonalCare.lastKey();
        String prd_list = tmap_PersonalCare.get(mykey);

        String[] productlist_array = prd_list.split(" ");

        for (String product : productlist_array) {

            if (counter < 5) {

                products.set(product);
```

```
                                        String category = "Personal_Care_Appliances";
                                        String prd_ratings = "" + mykey;

                                        CompositeKeyWritableTop obj = new CompositeKeyWritableTop(category,
prd_ratings);

                                        context.write(obj, products);

                                        counter++;

                                } else {
                                        break;
                                }

                        }
                        if (counter >= 5) {
                                break;
                        } else {
                                tmap_PersonalCare.remove(tmap_PersonalCare.lastKey());
                        }
                }

                // Emitting local top 5 key value from each mapper in Digital Software department
                counter = 0;

                for (int ptr = 5; ptr > 0; ptr--) {

                        if (tmap_Software.size() == 0) {
                                break;
                        }

                        long mykey = tmap_Software.lastKey();
                        String prd_list = tmap_Software.get(mykey);

                        String[] productlist_array = prd_list.split(" ");

                        for (String product : productlist_array) {

                                if (counter < 5) {

                                        products.set(product);

                                        String category = "Digital_Software";
                                        String prd_ratings = "" + mykey;

                                        CompositeKeyWritableTop obj = new CompositeKeyWritableTop(category,
prd_ratings);

                                        context.write(obj, products);

                                        counter++;

                                } else {
                                        break;
                                }

                        }
```

```
                    if (counter >= 5) {
                        break;
                    } else {
                        tmap_Software.remove(tmap_Software.lastKey());
                    }
                }
            }

}
```

****

TopProductReducer.java

****

```
package com.maria.Amazon8;

import java.io.IOException;
import java.util.Map;
import java.util.TreeMap;

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

public class TopProductReducer extends Reducer<CompositeKeyWritableTop, Text, CompositeKeyWritable,
LongWritable> {

    // tree map for finding top5 products from each department
    private TreeMap<Long, String> tmap_PersonalCare;
    private TreeMap<Long, String> tmap_Software;

    // Initializing two tree map. One for personal care department and another for
    // digital software department
    @Override
    public void setup(Context context) throws IOException, InterruptedException {

        tmap_PersonalCare = new TreeMap<Long, String>();
        tmap_Software = new TreeMap<Long, String>();
    }

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

        for (Text product_id : values) {

            long total_rating = Long.parseLong(key.getRating());

            // Finding top 5 product id from Personal Care Appliance department
            if (key.getProductCategory().equals("Personal_Care_Appliances")) {

                if (!tmap_PersonalCare.containsKey(total_rating)) {

                    tmap_PersonalCare.put(total_rating, product_id.toString());
                    } else {

                            String product_list = tmap_PersonalCare.get(total_rating);
                            product_list += " " + product_id.toString();
                            tmap_PersonalCare.put(total_rating, product_list);

                    }

                    if (tmap_PersonalCare.size() > 5) {

                            tmap_PersonalCare.remove(tmap_PersonalCare.firstKey());

                    }

                    // Finding top 5 product id from Digital Software department
            } else if (key.getProductCategory().equals("Digital_Software")) {

                    if (!tmap_Software.containsKey(total_rating)) {

                            tmap_Software.put(total_rating, product_id.toString());

                    } else {

                            String product_list = tmap_Software.get(total_rating);
                            product_list += " " + product_id.toString();
                            tmap_Software.put(total_rating, product_list);

                    }

                    if (tmap_Software.size() > 5) {

                            tmap_Software.remove(tmap_Software.firstKey());
                    }

            }
    }

}

@Override
public void cleanup(Context context) throws IOException, InterruptedException {

    Text products = new Text();
    LongWritable FinalRank = new LongWritable();

    // Emitting top 5 product id and its ratings from Personal_Care_Appliances
    // department

    int counter = 0;
    long rank =1;

    for (int ptr = 5; ptr > 0; ptr--) {

            if (tmap_PersonalCare.size() == 0) {
                    break;
            }
```

```
                long mykey = tmap_PersonalCare.lastKey();
                String prd_list = tmap_PersonalCare.get(mykey);

                String[] productlist_array = prd_list.split(" ");

                for (String product : productlist_array) {

                    if (counter < 5) {

                        products.set(product);
                        FinalRank.set(rank++);

                        String Product_Id = products.toString();
                        String Product_Category = "Personal_Care_Appliances";

                        CompositeKeyWritable obj = new CompositeKeyWritable(Product_Id,
Product_Category);

                        context.write(obj, FinalRank);

                        counter++;

                    } else {

                        break;
                    }

                }
                if (counter >= 5) {

                    break;
                } else {

                    tmap_PersonalCare.remove(tmap_PersonalCare.lastKey());
                }
            }

            // Emitting top 5 product id and its ratings from Digital Software department
            counter = 0;
            rank =1;

            for (int ptr = 5; ptr > 0; ptr--) {

                if (tmap_Software.size() == 0) {
                    break;
                }

                long mykey = tmap_Software.lastKey();
                String prd_list = tmap_Software.get(mykey);

                String[] productlist_array = prd_list.split(" ");

                for (String product : productlist_array) {

                    if (counter < 5) {
```

```
                                        products.set(product);
                                        FinalRank.set(rank++);

                                        String Product_Id = products.toString();
                                        String Product_Category = "Digital_Software";

                                        CompositeKeyWritable obj = new CompositeKeyWritable(Product_Id,
Product_Category);

                                        context.write(obj, FinalRank);

                                        counter++;

                                } else {

                                        break;
                                }

                        }
                        if (counter >= 5) {

                                break;
                        } else {

                                tmap_Software.remove(tmap_Software.lastKey());
                        }
                }
        }

}
```

APACHE PIG SCRIPT FOR MAPREDUCE

```
lOAD_DATA = LOAD
'/home/maria/Documents/{amazon_reviews_us_Digital_Software_v1_00.tsv,amazon_reviews_u
s_Personal_Care_Appliances_v1_00.tsv}' AS (marketplace:chararray, customer_id:chararray,
review_id:chararray, product_id:chararray, product_parent:chararray, product_title:chararray,
product_category:chararray, star_rating:int, helpful_votes:int, total_votes:int, vine:chararray,
verified_purchase:chararray, review_headline:chararray, review_body:chararray,
review_date:chararray);

FILTER_HEADER = FILTER lOAD_DATA BY marketplace != 'marketplace';
FILTER_COL = FOREACH FILTER_HEADER GENERATE $3, $6, $7;

GRP_PRDID = group FILTER_COL by product_id;

RATING_SUM = FOREACH GRP_PRDID GENERATE
(FILTER_COL.product_id),(FILTER_COL.product_category),
SUM(FILTER_COL.star_rating);

FLATTEN_DATA = FOREACH RATING_SUM GENERATE flatten($0) as id, flatten($1) as
category, flatten($2) as rating;
DIST_DATA = distinct FLATTEN_DATA;

GRP_PRDCAT = group DIST_DATA by category;
OUTPUT_DATA = FOREACH GRP_PRDCAT {
        top = TOP(5, 2, DIST_DATA);
        GENERATE top;
        }

FLATTEN_OUTPUT = FOREACH OUTPUT_DATA GENERATE flatten($0);
SORTED_OUTPUT = ORDER FLATTEN_OUTPUT by $1 DESC,$2 DESC;

RANKING = rank SORTED_OUTPUT BY $1, $2 DESC;

DUMP SORTED_OUTPUT;
```

APACHE HIVE QUERY FOR MAPREDUCE

```
 select * from (select product_id, product_category,sum(star_rating) as tot_rating,
 ROW_NUMBER() OVER (PARTITION BY product_category ORDER BY
sum(star_rating)DESC) as RNK
 from AmazonCustomerDataset group by product_id, product_category) t
 where rnk<6;
```