Big Data Analytics | Lab 5  
Market Basket Analysis Using MapReduce

Raj Panchal  
17CEUBG104

**Market Basket Analysis introduction :**

MBA is one of the Data Mining approaches to analyze the association of data sets.

The basic idea to find the associated pairs of items in a store when there are transaction data sets as below.

|  |
| --- |
| 1,bigbazaar,shirt:pant:tie  2,wallmart,eraser:sharpner:earphone  3,iskcon,toothbrush:gel:brazer:tie  4,dmart,blanket:blazer:lighter:perfume  5,bigbazaar,socks:shirt:blazer:helmet  6,central,scale:pencil:eraser:sharpner  7,wallmart,socks:shirt:pant:helmet  8,iskcon,pant:tie:blazer:gel  9,central,pen:eraser:helmet:blazer  10,dmart,toothbrush:toothpaste:helmet:lighter  11,bigbazaar,shirt:pant:tie:shoes  12,dmart,pen:pencil:sharpner:eraser  13,central,perfume:blazer:shoes  14,wallmart,toothbrush:toothpaste:bathsoap  15,iscon,shoes:blazer:blanket |

# Here first columns represent ID,second columns represent mall names,third columns represent items that customers have bought.

If store owners list a pair of items that occur frequently , he could control the stocks more efficiently through item arrangement on shelves or concurrent promotion of items. Thus he would increase the opportunity to make a profit by controlling the order of products and marketing.

For example. People have built and run MBA codes that compute the top frequently occurring pairs of items on as above data, when customers buy a shirt, they purchase a pants as well which happens more frequently. Thus , the owner can refer to the data to run the store.

**Solution :**

**Step 1 :**

**Collect data for input as shown above and upload on hdfs using put command.**

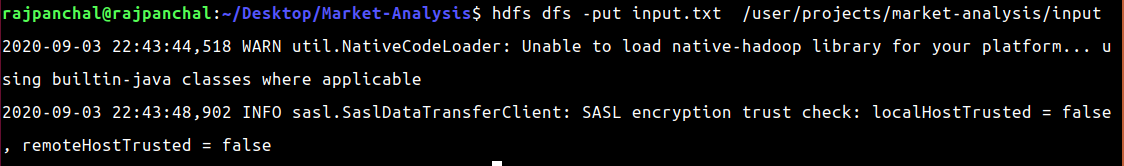
1. Create an input.txt file in the local system and store the above data.
2. Now create a directory in hdfs for this problem.



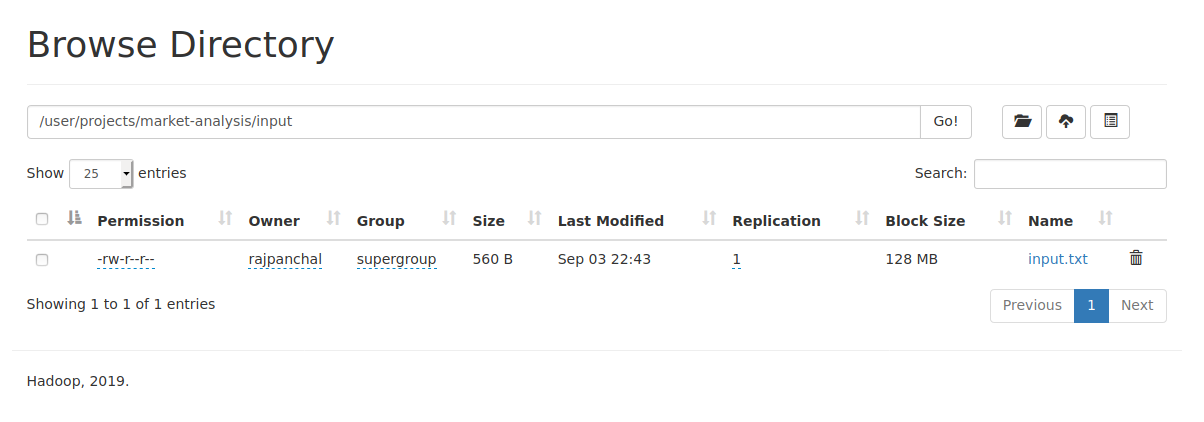
1. Create input sub directory in market-analysis folder.



1. Put input.txt file in input directory.



1. Confirm presence of input file in hdfs.



**Step 2 :**

**Implement MapReduce code.**

1. Write a MapReduce program for MBA.

Code :

|  |
| --- |
| import java.io.IOException;  import java.util.StringTokenizer;  import java.util.\*;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.\*;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.Mapper;  import org.apache.hadoop.mapreduce.Reducer;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public class MarketAnalysis  {  public static class Item implements Comparable<Item>  {  private String name;  private int occurance;    public String getName()  {  return name;  }    public int getOccurance()  {  return occurance;  }    public int compareTo(Item item2)  {  return -(this.occurance-item2.occurance);  }    public Item(String name, int occurance)  {  this.name=name;  this.occurance=occurance;  }    public String toString()  {  return "Item [ name = " + name + ", occurance = " + occurance + "]";  }  }    public static class MarketAnalysisMapper extends Mapper < LongWritable,  Text, Text, IntWritable >  {  public void map (LongWritable key, Text value, Context context) throws IOException, InterruptedException  {  String data[]=value.toString().split(",");  String items=data[data.length-1];  String itemarray[] = items.split(":");    for(int i=0;i<itemarray.length;i++)  {  for(int j=i+1;j<itemarray.length;j++)  {  String word1 = itemarray[i];  String word2 = itemarray[j];  if(word2.compareTo(word1)>0)  {  context.write(new Text(word2 + "," + word1),new IntWritable(1));  }  else  {  context.write(new Text(word1 + "," + word2),new IntWritable(1));  }  }  }  }  }    public static class SumReducer extends Reducer < Text, IntWritable,  Text, IntWritable >  {  List<Item> itemList = new ArrayList<>();  public void reduce(Text key, Iterable < IntWritable > values, Context context) throws  IOException, InterruptedException  {  int sum=0;  for(IntWritable value:values)  {  sum+=value.get();  }  Item item = new Item(key.toString(),sum);  itemList.add(item);  }    public void cleanup(Context context) throws IOException, InterruptedException  {  Collections.sort(itemList);  System.out.println(itemList);  for (Item item : itemList)  {  context.write(new Text(item.getName()),new IntWritable(item.getOccurance()));  }  }  }    public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "MarketAnalysis");  job. setJarByClass(MarketAnalysis.class);  job. setMapperClass(MarketAnalysisMapper.class);  job. setCombinerClass(SumReducer.class);  job. setReducerClass(SumReducer.class);  job. setOutputKeyClass(Text.class);  job. setOutputValueClass(IntWritable.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  System. exit (job.waitForCompletion(true) ? 0 : 1);  }  } |

1. Save file as named MarketAnalysis.java.



1. UnderStanding of different phases of code.

Mapper class (MarketAnalysismapper)

This class creates a pair of items and maps them with a key-value pair.

If there is duplication of pairs then omit the duplicate pair.

And finally map like below.

( [ pant shirt ], 1 )

( [ pant tie ], 1 )

( [ eraser sharpener ], 1) ……

Reducer class (SumReducer)

This class takes input as a key-value pair as shown below.

( [ pant shirt ], ( 1, 1, 1 ) )

( [ pant tie ], ( 1, 1 ) )

( [ tie shoes ] , (1 ,1, 1, 1) ).......

And produce a sum of the particular pair of item as output.

( [ pant shirt ], 3 )

( [ pant tie ], 2 )

( [ tie shoes ] , 4 ).......

There is a also function cleanup that sort the output based on the occurrence of pair of items.

Drive class (main)

In the driver class, we set the configuration of our Mapreduce job to run in Hadoop.

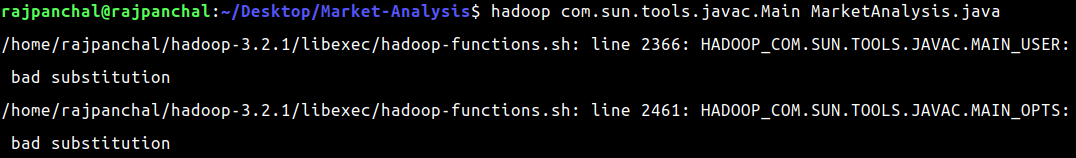
It specifies

* the name of the job, the data type of input/output of the mapper and reducer.
* the names of the mapper and reducer classes.
* Path of the input and output folder.

**Step 3 :**

**Run the MapReduce program successfully.**

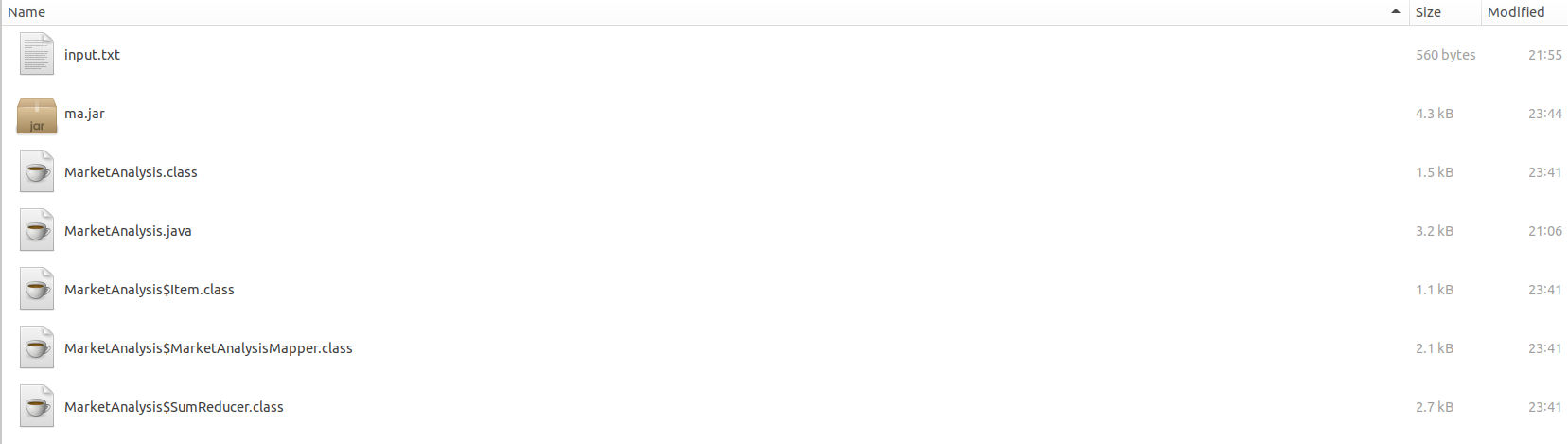
1. Compile the code.



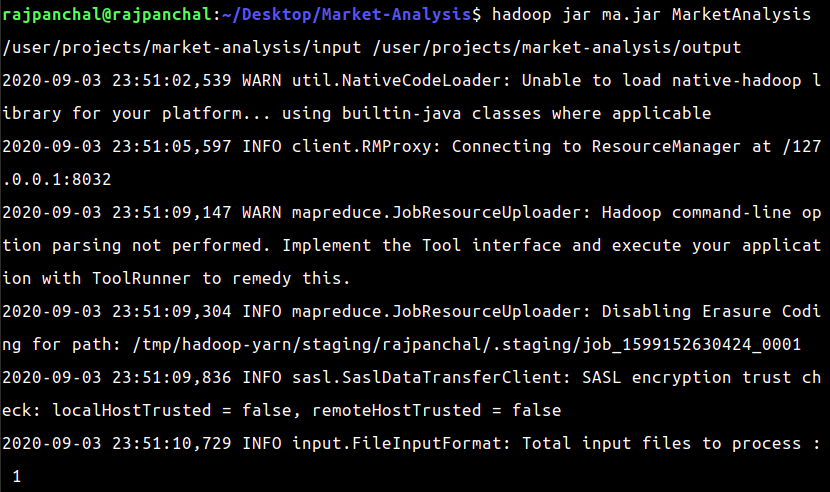
1. Create a jar file named ma.jar in the local system.



1. Locate the jar file in its folder.



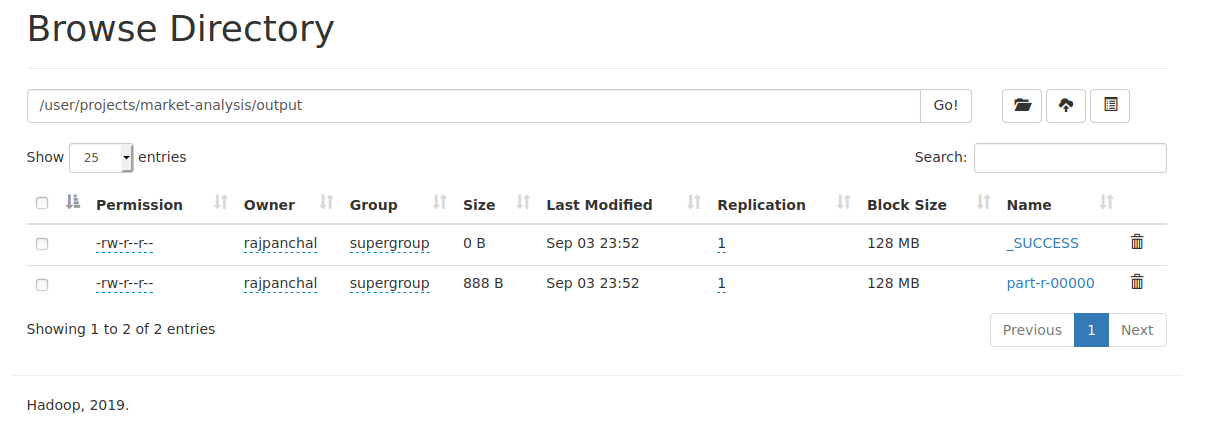
1. Run jar file with input and output argument.



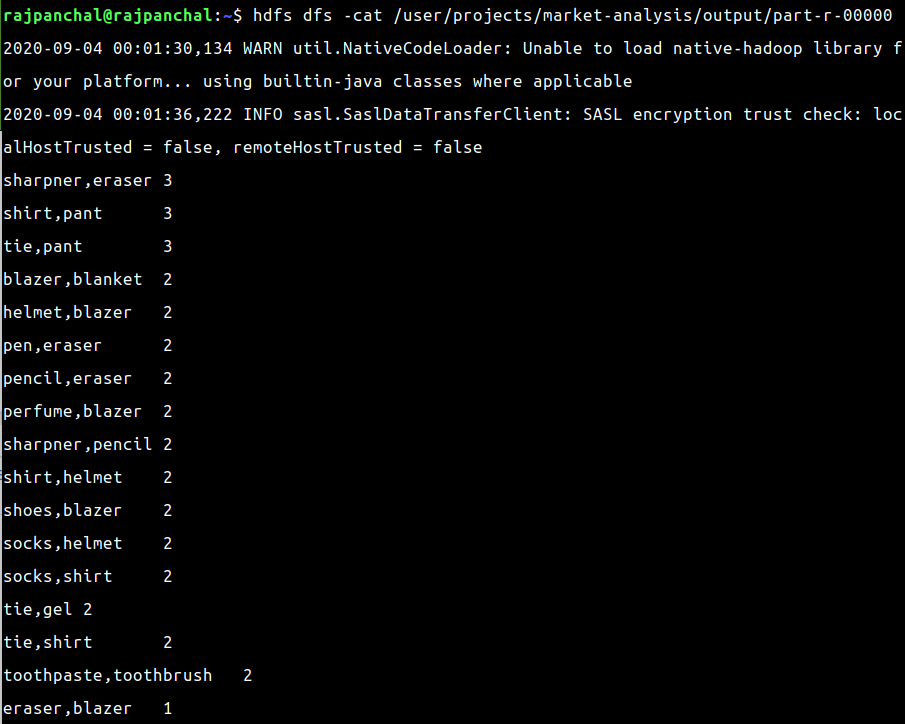
**Step 4 :**

**Observe the output**.

1. Locate the output directory in hdfs.



1. Print output on terminal.



We can conclude from this output that if customers buy sharpeners then they would also go for erases as well and this is also true for shirt and pant pairs.

Using this information the owner could arrange the item in such a way that its next item is also to be likely to buy. Hence this problem helps shopping malls to increase their profits.

DocumentLink: [BDA\_CE74\_LAB5](https://docs.google.com/document/d/1HjpaTTIfYABAkIkmjt-P6Io4e1IG_ieaDBslgMhOvFA/edit?usp=sharing)

Code :[MarketAnalysis.java](https://drive.google.com/file/d/1juT4p1RzGmGKP-ZJ737H5Hq8C_tVw2b5/view?usp=sharing)