# BIG DATA HADOOP AND SPARK DEVLOPMENT

# ASSIGNMENT 5

Table of Contents:

# BIG DATA HADOOPAND SPARK DEVELOPMENT

## 1. Introduction

In this assignment, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

## 2. Objective

This Assignment consolidates the deeper understanding of the Session – 5 ADVANCE MAP REDUCE AND INTRODUCTION TO UNIX CONCEPTS

## 3. Associated Data Files

DATASET
Dataset is sample data of songs heard by users on an online streaming platform. The Description of data set attached in musicdata.txt is as follows: -
1st Column - UserId
2nd Column - TrackId
3rd Column - Songs Share status (1 for shared, 0 for not shared)
4th Column - Listening Platform (Radio or Web - 0 for radio, 1 for web)
5th Column - Song Listening Status (0 for skipped, 1 for fully heard)

## 4. Problem Statement

**Write Map Reduce program for following tasks.**

- Task 1
  Find the number of unique listeners in the data set.

- Task 2
  What are the number of times a song was heard fully?

- Task 3
  What are the number of times a song was shared?

5. Expected Output

- Task 1

Find the number of unique listeners in the data set.

## UniqueListenerMapper.java (Mapper class)

```
package uniqueListenersPackage;

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 UniqueListenerMapper extends Mapper<LongWritable,
Text,IntWritable,IntWritable>{

IntWritable trackId = new IntWritable();
IntWritable userId = new IntWritable();

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

      String parts [] = value.toString().split("\\|");
      System.out.println(parts.length);

      trackId.set(Integer.parseInt(parts [1]));
      userId.set(Integer.parseInt(parts [0]));

      if(parts.length == 5)
      {
            context.write(trackId, userId);
      }
      }
}
```

Java codes for the Mapper class is written above to find the number of unique listeners in the given data set musicdata.txt

By using

```
      trackId.set(Integer.parseInt(parts [1]));
      userId.set(Integer.parseInt(parts [0]));
```

we set the values in the data set as trackId and userId

By using If condition the length of data given is compared to 5 if true then

By using

**context.write(trackId,userId)** we write the output trackId and userId in the file.

## UniqueListnersReducer (Reducer Class)

```
package uniqueListenersPackage;

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

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

public class UniqueListenerReducer extends
Reducer<IntWritable,IntWritable,IntWritable,IntWritable>
{
      public void reduce(
                  IntWritable trackId,
                  Iterable<IntWritable> userIds,

      Reducer<IntWritable,IntWritable,IntWritable,IntWritable>.Context
context)
                  throws IOException,InterruptedException {

            Set<Integer> userIdSet = new HashSet<Integer>();
            for(IntWritable userId : userIds)
            {
                  userIdSet.add(userId.get());
            }
            IntWritable size = new IntWritable(userIdSet.size());
            context.write(trackId,size);
            }
      }
```

Here userIdSet is created as HashSet<Integer> ();

Here we use for iteration to add up the userIdSet by getting the value of userId

Here we create size as object of userIdSet.size() and pass it in the **context.write** to write in the file.

The TrackId and size is written in the output file.

## UniqueListeners (Driver class)

```
package uniqueListenersPackage;

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.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 UniqueListeners {
      public static void main(String[] args) throws ClassNotFoundException,
IOException,InterruptedException,ArrayIndexOutOfBoundsException
      {
      if(args.length != 2)
      {
      System.err.println("Usage: UniqueListeners <inputpath><output path>");
      System.exit(-1);
      }
      //Job Related Configurations
      Configuration conf = new Configuration();
      @SuppressWarnings("deprecation")
      Job job = new ~~Job~~(conf,"No.of Unique Listeners");
      job.setJarByClass(UniqueListeners.class);

      //Specify the number of reducer to 1
      job.setNumReduceTasks(1);

      //Provide paths to pick the input file for the Job
      FileInputFormat.setInputPaths(job, new Path(args[0]));

//Provide paths for the output file, and delete it if it is already present
      Path outputPath = new Path(args[1]);
      FileOutputFormat.setOutputPath(job,outputPath);
      outputPath.getFileSystem(conf).delete(outputPath,true);

    //To set mapper and Reducer for the Job
    job.setMapperClass(uniqueListenersPackage.UniqueListenerMapper.class);
    job.setReducerClass(uniqueListenersPackage.UniqueListenerReducer.class);

      //Set Input and Output Format classes
      job.setInputFormatClass(TextInputFormat.class);
      job.setOutputFormatClass(TextOutputFormat.class);

      //Set the Output key and value classes
      job.setOutputKeyClass(IntWritable.class);
      job.setOutputValueClass(IntWritable.class);

      //Execute the job
      System.exit(job.waitForCompletion(true)?0:1);
      }
}
```

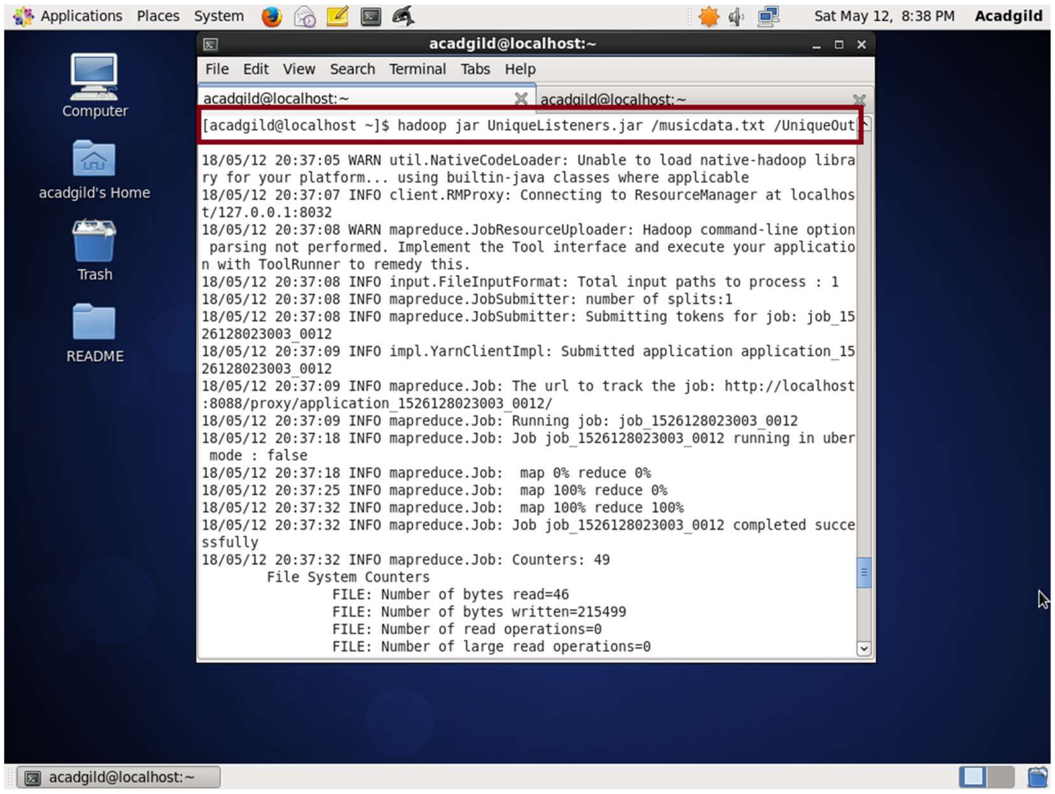
**Job Related Configurations**

```
Configuration conf = new Configuration();
@SuppressWarnings("deprecation")
Job job = new ~~Job~~(conf,"No.of Unique Listeners");
job.setJarByClass(UniqueListeners.class);
```

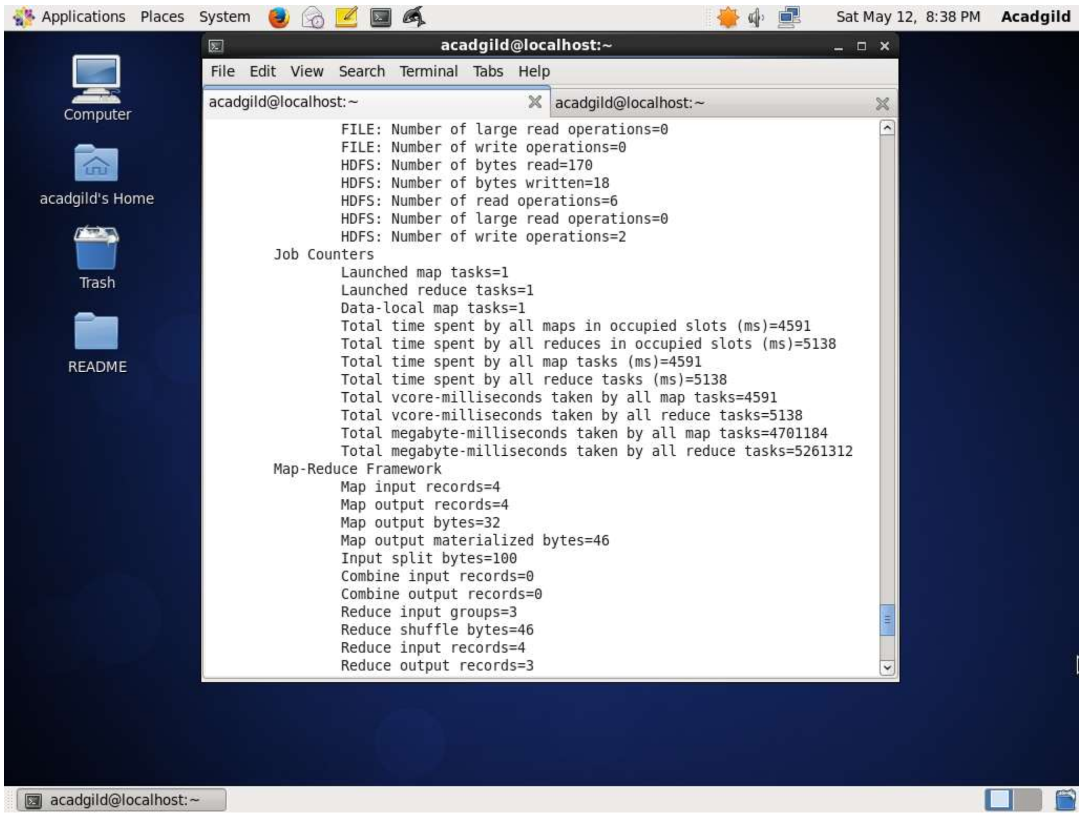
Object is created for configuration and new job object is created by passing them as parameters

By using job.setJarByClass the driver class is passed over here. So that the Compiler reads the code from driver class(UniqueListeners.class)

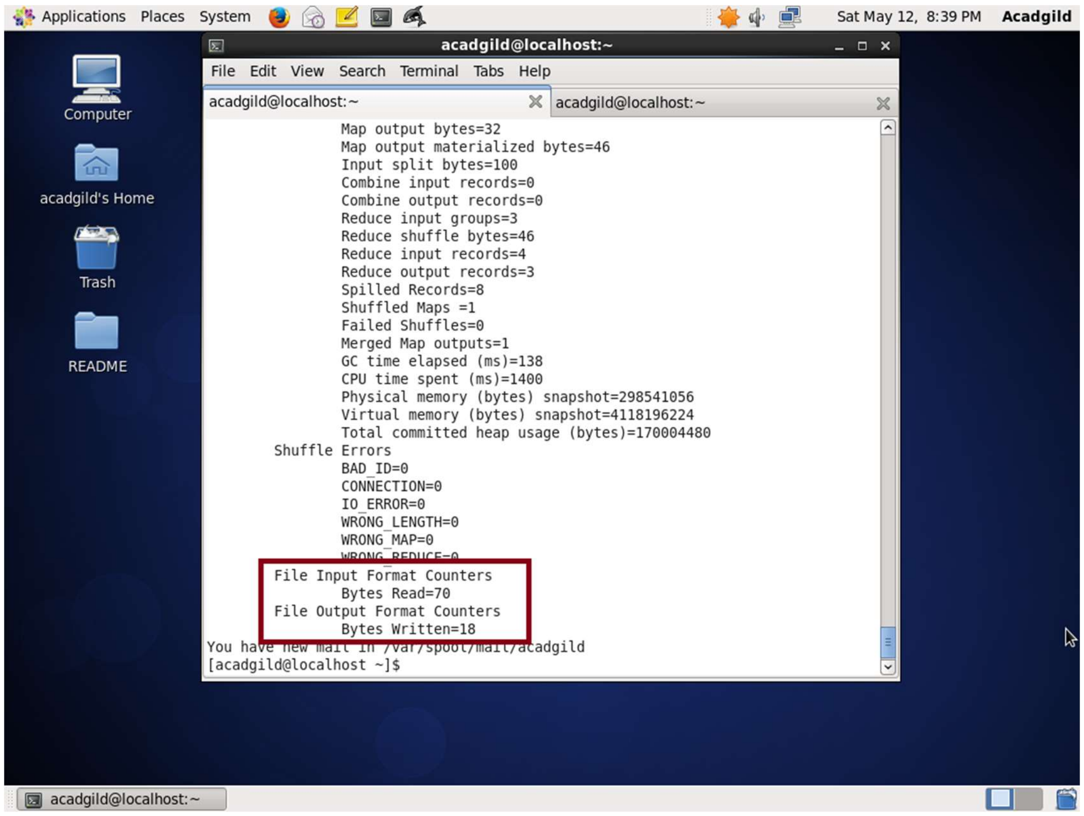
Following are Screenshot to find the Number of Unique Listeners in the data set(musicdata.txt)



```
[acadgild@localhost ~]$ hadoop jar UniqueListeners.jar /musicdata.txt /UniqueOut
18/05/12 20:37:05 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
18/05/12 20:37:07 INFO client.RMProxy: Connecting to ResourceManager at localhos
t/127.0.0.1:8032
18/05/12 20:37:08 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
18/05/12 20:37:08 INFO input.FileInputFormat: Total input paths to process : 1
18/05/12 20:37:08 INFO mapreduce.JobSubmitter: number of splits:1
18/05/12 20:37:08 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
26128023003_0012
18/05/12 20:37:09 INFO impl.YarnClientImpl: Submitted application application_15
26128023003_0012
18/05/12 20:37:09 INFO mapreduce.Job: The url to track the job: http://localhost
:8088/proxy/application_1526128023003_0012/
18/05/12 20:37:09 INFO mapreduce.Job: Running job: job_1526128023003_0012
18/05/12 20:37:18 INFO mapreduce.Job: Job job_1526128023003_0012 running in uber
 mode : false
18/05/12 20:37:18 INFO mapreduce.Job:  map 0% reduce 0%
18/05/12 20:37:25 INFO mapreduce.Job:  map 100% reduce 0%
18/05/12 20:37:32 INFO mapreduce.Job:  map 100% reduce 100%
18/05/12 20:37:32 INFO mapreduce.Job: Job job_1526128023003_0012 completed succe
ssfully
18/05/12 20:37:32 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=46
                FILE: Number of bytes written=215499
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
```



```
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=170
                HDFS: Number of bytes written=18
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=4591
                Total time spent by all reduces in occupied slots (ms)=5138
                Total time spent by all map tasks (ms)=4591
                Total time spent by all reduce tasks (ms)=5138
                Total vcore-milliseconds taken by all map tasks=4591
                Total vcore-milliseconds taken by all reduce tasks=5138
                Total megabyte-milliseconds taken by all map tasks=4701184
                Total megabyte-milliseconds taken by all reduce tasks=5261312
        Map-Reduce Framework
                Map input records=4
                Map output records=4
                Map output bytes=32
                Map output materialized bytes=46
                Input split bytes=100
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=46
                Reduce input records=4
                Reduce output records=3
```



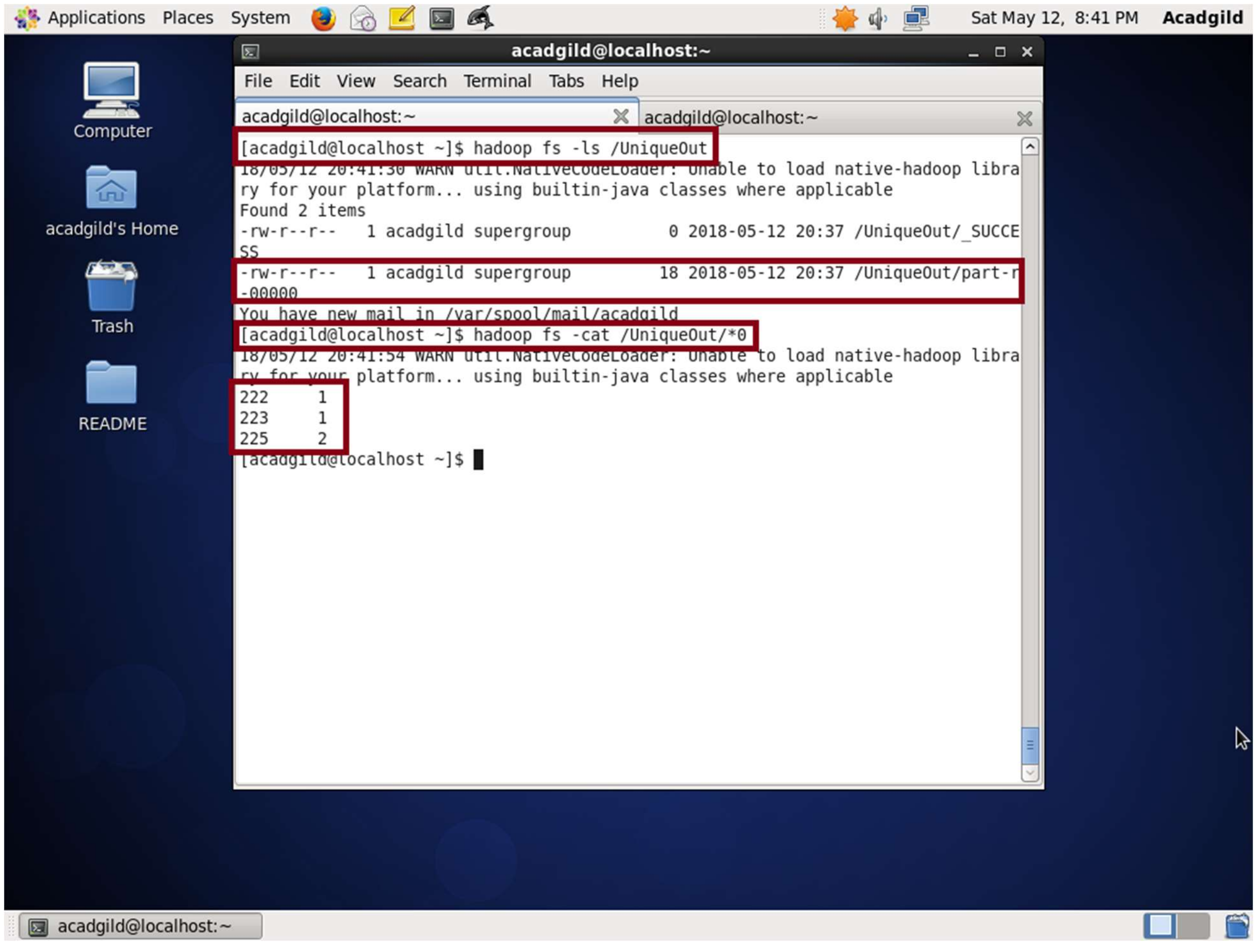
```
                Map output bytes=32
                Map output materialized bytes=46
                Input split bytes=100
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=46
                Reduce input records=4
                Reduce output records=3
                Spilled Records=8
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=138
                CPU time spent (ms)=1400
                Physical memory (bytes) snapshot=298541056
                Virtual memory (bytes) snapshot=4118196224
                Total committed heap usage (bytes)=170004480
        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=70
        File Output Format Counters
                Bytes Written=18
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

The Output is saved in the UniqueOut directory

Which can be listed by Hadoop fs –ls / command and

By using the following command, the data in the file is shown

**hadoop fs –cat /UniqueOut/*0**


Applications Places System
Sat May 12, 8:41 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
Computer
acadgild's Home
Trash
README

```
[acadgild@localhost ~]$ hadoop fs -ls /UniqueOut
18/05/12 20:41:30 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2018-05-12 20:37 /UniqueOut/_SUCCE
SS
-rw-r--r--   1 acadgild supergroup         18 2018-05-12 20:37 /UniqueOut/part-r
-00000
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -cat /UniqueOut/*0
18/05/12 20:41:54 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
222     1
223     1
225     2
[acadgild@localhost ~]$
```

- Task 2

What are the number of times a song was heard fully?

## HeardFully.java (Driver class)

```
package HeardFullyPackage;

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.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 HeardFully {

      public static void main(String[] args) throws ClassNotFoundException,
IOException, InterruptedException{
            if (args.length != 2) {
         System.err.println("Usage: HeardFully <input path> <output path>");
              System.exit(-1);
            }
            //Job Related Configurations
            Configuration conf = new Configuration();
            Job job = new ~~Job~~(conf, "No. of songs heard fully");
            job.setJarByClass(HeardFully.class);
            // Specify the number of reducer to 1
          job.setNumReduceTasks(1);
          //Provide paths to pick the input file for the job
          FileInputFormat.setInputPaths(job, new Path(args[0]));

          //Provide paths to output file for the job, and delete it if
already present
            Path outputPath = new Path(args[1]);
            FileOutputFormat.setOutputPath(job, outputPath);
            outputPath.getFileSystem(conf).delete(outputPath, true);
            //To set the mapper and reducer of this job
          job.setMapperClass(HeardFullyPackage.HeardFullyMapper.class);
          job.setReducerClass(HeardFullyPackage.HeardFullyReducer.class);

          //set the input and output format class
          job.setInputFormatClass(TextInputFormat.class);
          job.setOutputFormatClass(TextOutputFormat.class);

          //set up the output key and value classes
          job.setOutputKeyClass(IntWritable.class);
          job.setOutputValueClass(IntWritable.class);
          //execute the job
          System.exit(job.waitForCompletion(true) ? 0: 1);
      }
}
```

In Driver class we specify the input path and output path for the job is configured. Format of the same are determined.

Mapper class and reducer class are set by using **setMapperClass** and **setReducerclass** for the job.

Configuration is made by creating the object **conf** and New object job is created for Job by calling **conf** object.

Output file path is provided to write the output in the file. In case, the file is already present then the file will be deleted and file will be created freshly.

## HeardFullyMapper.java (Mapper Class)

```
package HeardFullyPackage;

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 HeardFullyMapper extends Mapper<LongWritable, Text,
IntWritable, IntWritable>{

      IntWritable trackId = new IntWritable();
    IntWritable heardFull = new IntWritable();

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

            String record = value.toString();
            String parts[] = record.split("\\|");

            trackId.set(Integer.parseInt(parts[1]));
            heardFull.set(Integer.parseInt(parts[4]));

            if (parts.length == 5) {
                  context.write(trackId, heardFull);
            }
      }
}
```

Java codes for the Mapper class is written above to find the number of times the song was heard in musicdata.txt

By using

trackId.set(Integer.parseInt(parts [1]));

heardFull.set(Integer.parseInt(parts [4]));

we set the values in the data set as trackId and heardFull

By using If condition the length of data given is compared to 5 if true then

By using

context.write(trackId,heardFull) we write the output trackId and heardFull in the file.

## HeardFullyReducer.java

```
package HeardFullyPackage;

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

public class HeardFullyReducer extends Reducer<IntWritable, IntWritable,
IntWritable, IntWritable> {

        public void reduce(
                    IntWritable trackId,
                    Iterable<IntWritable> heardFull,
                    Reducer<IntWritable, IntWritable, IntWritable,
IntWritable>.Context context)
                  throws IOException, InterruptedException {

              List<Integer> heardFullyList = new ArrayList<Integer>();

              for(IntWritable songsHeard: heardFull) {

                    if(songsHeard.get() == 1) {
                          heardFullyList.add(songsHeard.get());
                    }
              }
              IntWritable size = new IntWritable(heardFullyList.size());
              context.write(trackId, size);
             }
}
```

Here List<Integer> heardFullyList is created as ArrayList<Integer>()

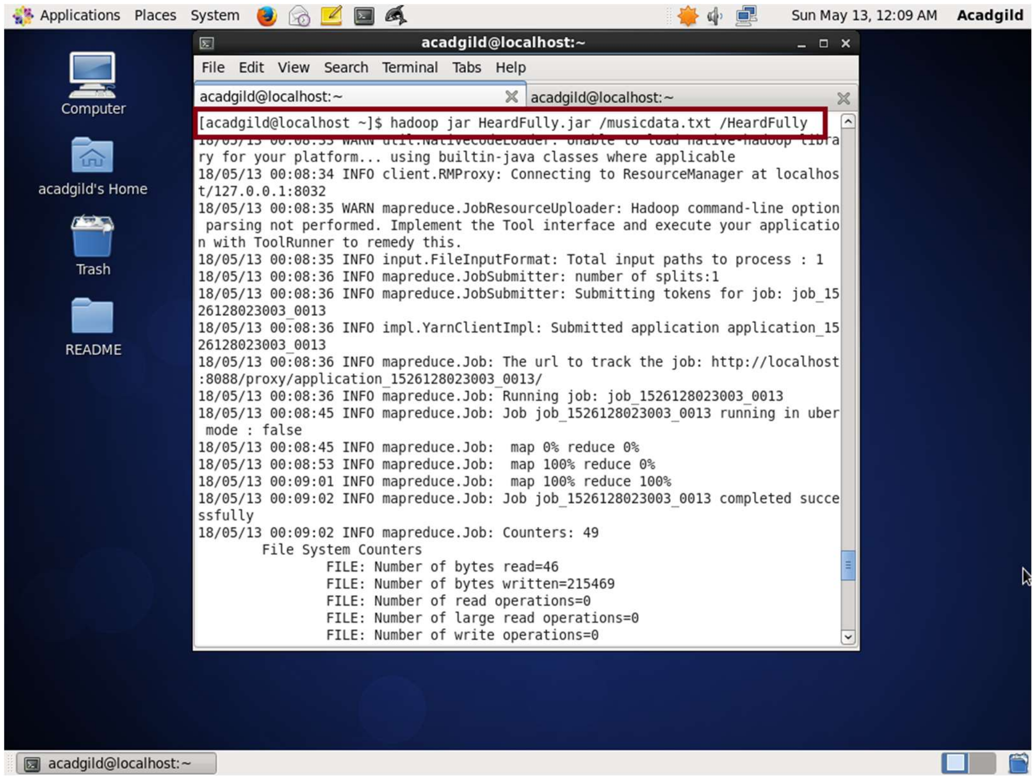
Here we use for iteration to add up the heardFullyList by getting the value of songsHeard

Here we create size as object of heardFullyList.size() and pass it in the context.write to write in the file.

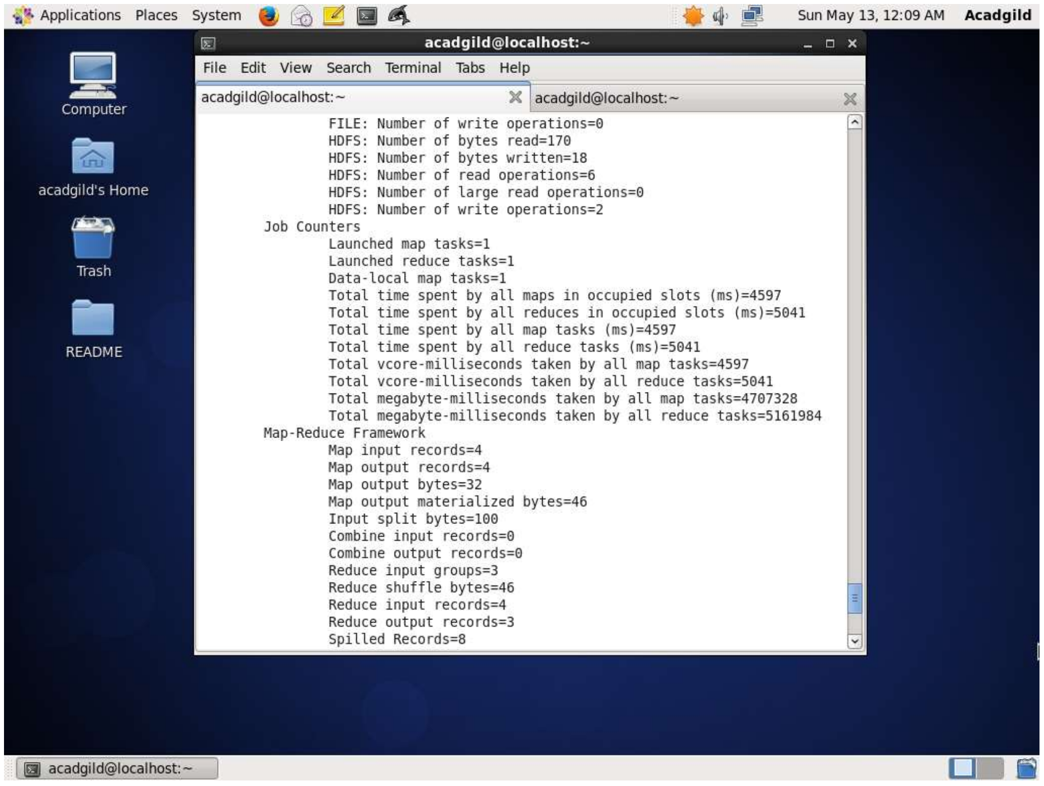
The TrackId and size is written in the output file.

The following are the screenshot of the number of times a song was heard fully

hadoop jar HeardFully.jar /musicdata.txt /HeardFully



```
[acadgild@localhost ~]$ hadoop jar HeardFully.jar /musicdata.txt /HeardFully
18/05/13 00:08:33 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
18/05/13 00:08:34 INFO client.RMProxy: Connecting to ResourceManager at localhos
t/127.0.0.1:8032
18/05/13 00:08:35 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
18/05/13 00:08:35 INFO input.FileInputFormat: Total input paths to process : 1
18/05/13 00:08:36 INFO mapreduce.JobSubmitter: number of splits:1
18/05/13 00:08:36 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
26128023003_0013
18/05/13 00:08:36 INFO impl.YarnClientImpl: Submitted application application_15
26128023003_0013
18/05/13 00:08:36 INFO mapreduce.Job: The url to track the job: http://localhost
:8088/proxy/application_1526128023003_0013/
18/05/13 00:08:36 INFO mapreduce.Job: Running job: job_1526128023003_0013
18/05/13 00:08:45 INFO mapreduce.Job: Job job_1526128023003_0013 running in uber
 mode : false
18/05/13 00:08:45 INFO mapreduce.Job:  map 0% reduce 0%
18/05/13 00:08:53 INFO mapreduce.Job:  map 100% reduce 0%
18/05/13 00:09:01 INFO mapreduce.Job:  map 100% reduce 100%
18/05/13 00:09:02 INFO mapreduce.Job: Job job_1526128023003_0013 completed succe
ssfully
18/05/13 00:09:02 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=46
                FILE: Number of bytes written=215469
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
```



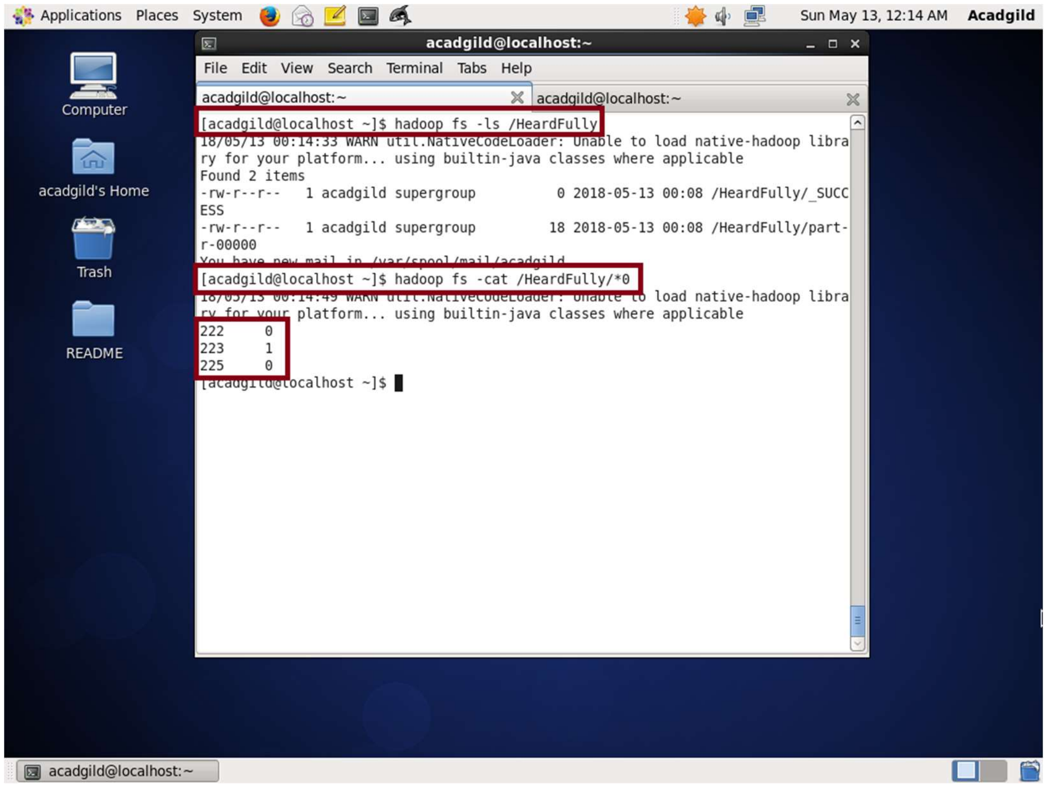
```
                FILE: Number of write operations=0
                HDFS: Number of bytes read=170
                HDFS: Number of bytes written=18
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=4597
                Total time spent by all reduces in occupied slots (ms)=5041
                Total time spent by all map tasks (ms)=4597
                Total time spent by all reduce tasks (ms)=5041
                Total vcore-milliseconds taken by all map tasks=4597
                Total vcore-milliseconds taken by all reduce tasks=5041
                Total megabyte-milliseconds taken by all map tasks=4707328
                Total megabyte-milliseconds taken by all reduce tasks=5161984
        Map-Reduce Framework
                Map input records=4
                Map output records=4
                Map output bytes=32
                Map output materialized bytes=46
                Input split bytes=100
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=46
                Reduce input records=4
                Reduce output records=3
                Spilled Records=8
```

The Output is saved in the HeardFully directory

Which can be listed by hadoop fs –ls / command and

By using the following command, the data in the file is shown

hadoop fs –cat /HeardFully/*0



```
[acadgild@localhost ~]$ hadoop fs -ls /HeardFully
18/05/13 00:14:33 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2018-05-13 00:08 /HeardFully/_SUCC
ESS
-rw-r--r--   1 acadgild supergroup         18 2018-05-13 00:08 /HeardFully/part-
r-00000
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -cat /HeardFully/*0
18/05/13 00:14:49 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
222     0
223     1
225     0
[acadgild@localhost ~]$
```

- Task 3

What are the number of times a song was shared?

### SharedSongsMapper.java (Mapper class)

```java
package SharedSongPackage;

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 SharedSongsMapper extends Mapper<LongWritable, Text,
IntWritable, IntWritable>{

      IntWritable trackId = new IntWritable();
    IntWritable songShared = new IntWritable();

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

            String record = value.toString();
            String parts[] = record.split("\\|");

            trackId.set(Integer.parseInt(parts[1]));
            songShared.set(Integer.parseInt(parts[2]));

            if (parts.length == 5) {
                  context.write(trackId, songShared);
            }
      }
}
```

Java codes for the Mapper class is written above to find the number of times songs were shared in musicdata.txt

By using

trackId.set(Integer.parseInt(parts [1]));

songShared.set(Integer.parseInt(parts [2]));

we set the values in the data set as trackId and songShared

By using If condition the length of data given is compared to 5 if true then

By using

context.write(trackId,songShared) we write the output trackId and songShared in the file.

## SharedSongsReducer (Reducer class)

```
package SharedSongPackage;

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

public class SharedSongsReducer extends Reducer<IntWritable, IntWritable,
IntWritable, IntWritable> {

        public void reduce(
                    IntWritable trackId,
                    Iterable<IntWritable> songShared,
                    Reducer<IntWritable, IntWritable, IntWritable,
IntWritable>.Context context)
                  throws IOException, InterruptedException {

              List<Integer> songSharedList = new ArrayList<Integer>();

              for(IntWritable shared: songShared) {

                    if(shared.get() == 1) {
                          songSharedList.add(shared.get());
                    }
              }
              IntWritable size = new IntWritable(songSharedList.size());
              context.write(trackId, size);
               }
}
```

Java codes for the Reducer class is written above to find the number of times songs were shared in musicdata.txt

List<Integer> songSharedList is created by ArrayList<Integer>();

If shared.get() is 1 then

songSharedList is added with shared.get value

Else

Size is the object created for songSharedList.size()

songSharedList.size() and trackId are written in the Output file.

## SharedSongs (Driver class)

```
package SharedSongPackage;

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.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 SharedSongs {

      public static void main(String[] args) throws ClassNotFoundException,
IOException, InterruptedException{
      if (args.length != 2) {
        System.err.println("Usage: HeardFully <input path> <output path>");
              System.exit(-1);
            }
            //Job Related Configurations
            Configuration conf = new Configuration();
            Job job = new Job(conf, "No. of times a song was shared");
            job.setJarByClass(SharedSongs.class);

            // Specify the number of reducer to 1
          job.setNumReduceTasks(1);

          //Provide paths to pick the input file for the job
          FileInputFormat.setInputPaths(job, new Path(args[0]));

          //Provide paths to pick the output file for the job, and delete it
if already present
           Path outputPath = new Path(args[1]);
           FileOutputFormat.setOutputPath(job, outputPath);
           outputPath.getFileSystem(conf).delete(outputPath, true);

          //To set the mapper and reducer of this job
          job.setMapperClass(SharedSongPackage.SharedSongsMapper.class);
          job.setReducerClass(SharedSongPackage.SharedSongsReducer.class);
          //set the input and output format class
          job.setInputFormatClass(TextInputFormat.class);
          job.setOutputFormatClass(TextOutputFormat.class);
          //set up the output key and value classes
          job.setOutputKeyClass(IntWritable.class);
          job.setOutputValueClass(IntWritable.class);
          //execute the job
          System.exit(job.waitForCompletion(true) ? 0 : 1);
      }

}
```

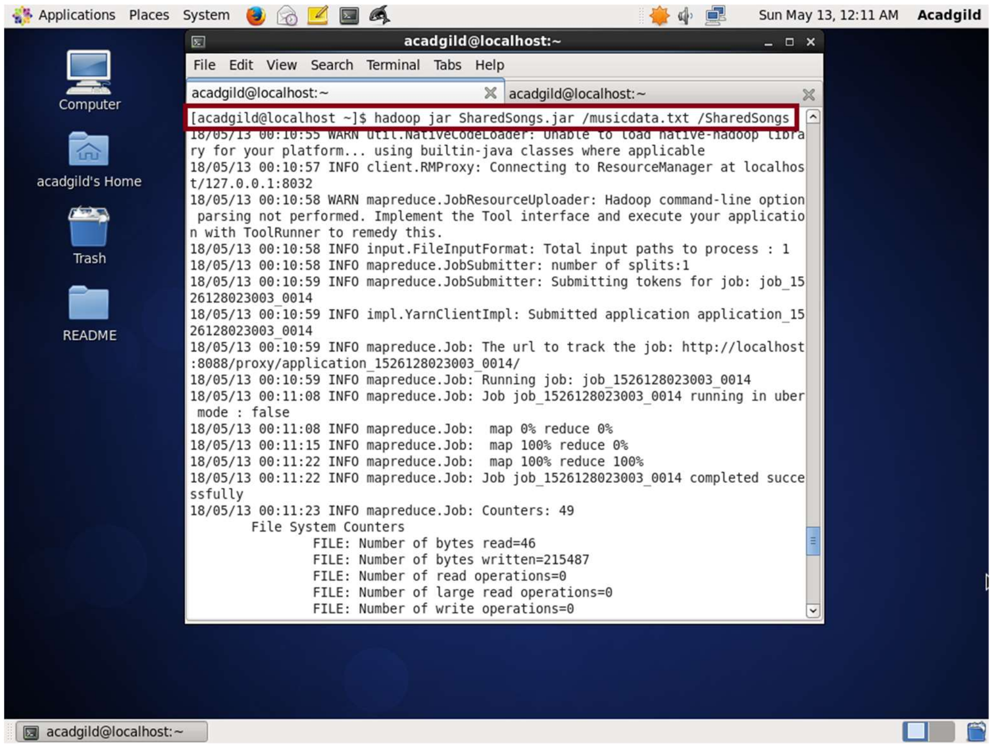
In Driver class we specify the input path and output path for the job is configured. Format of the input and Output formats are Textinputformat and Textoutputformat.

Mapper class and reducer class are set by using **setMapperClass** and **setReducerclass** for the job.

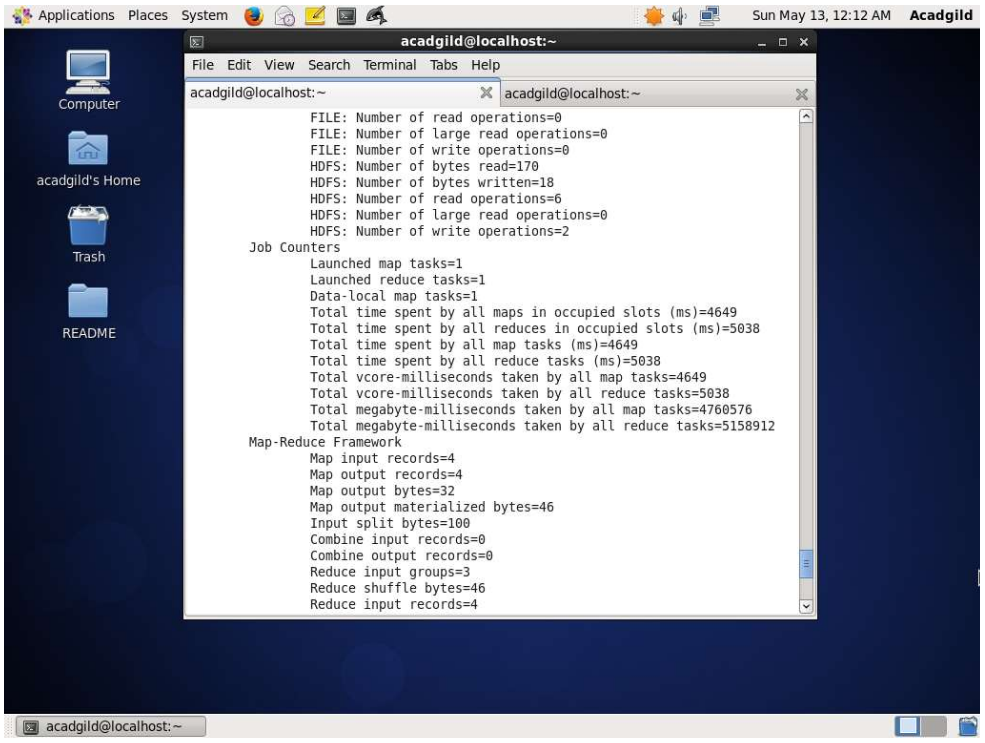
Configuration is made by creating the object **conf** and New object job is created for Job by calling **conf** object.

Output file path is provided to write the output in the file. In case, the file is already present then the file will be deleted and file will be created freshly.

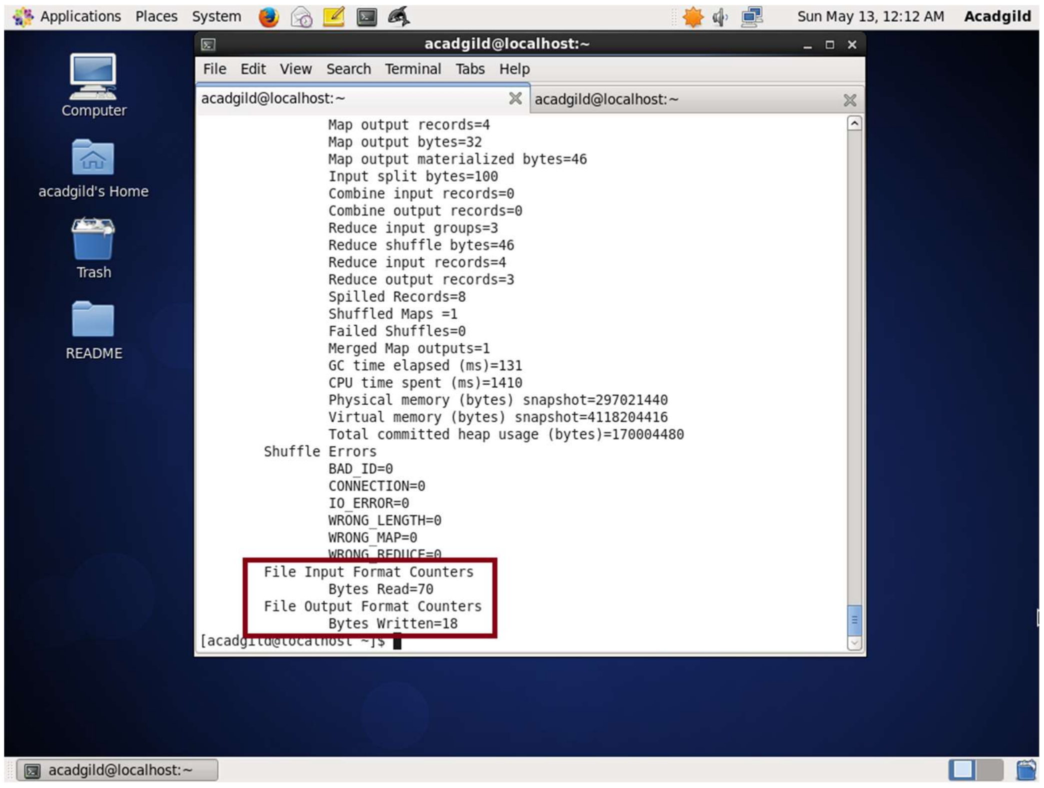
Screen shots for the number of times the song was shared are shown below.



```
[acadgild@localhost ~]$ hadoop jar SharedSongs.jar /musicdata.txt /SharedSongs
18/05/13 00:10:55 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
18/05/13 00:10:57 INFO client.RMProxy: Connecting to ResourceManager at localhos
t/127.0.0.1:8032
18/05/13 00:10:58 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
18/05/13 00:10:58 INFO input.FileInputFormat: Total input paths to process : 1
18/05/13 00:10:58 INFO mapreduce.JobSubmitter: number of splits:1
18/05/13 00:10:59 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
26128023003_0014
18/05/13 00:10:59 INFO impl.YarnClientImpl: Submitted application application_15
26128023003_0014
18/05/13 00:10:59 INFO mapreduce.Job: The url to track the job: http://localhost
:8088/proxy/application_1526128023003_0014/
18/05/13 00:10:59 INFO mapreduce.Job: Running job: job_1526128023003_0014
18/05/13 00:11:08 INFO mapreduce.Job: Job job_1526128023003_0014 running in uber
 mode : false
18/05/13 00:11:08 INFO mapreduce.Job:  map 0% reduce 0%
18/05/13 00:11:15 INFO mapreduce.Job:  map 100% reduce 0%
18/05/13 00:11:22 INFO mapreduce.Job:  map 100% reduce 100%
18/05/13 00:11:22 INFO mapreduce.Job: Job job_1526128023003_0014 completed succe
ssfully
18/05/13 00:11:23 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=46
                FILE: Number of bytes written=215487
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
```


Applications Places System
Sun May 13, 12:12 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
Computer
acadgild's Home
Trash
README

```
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=170
                HDFS: Number of bytes written=18
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=4649
                Total time spent by all reduces in occupied slots (ms)=5038
                Total time spent by all map tasks (ms)=4649
                Total time spent by all reduce tasks (ms)=5038
                Total vcore-milliseconds taken by all map tasks=4649
                Total vcore-milliseconds taken by all reduce tasks=5038
                Total megabyte-milliseconds taken by all map tasks=4760576
                Total megabyte-milliseconds taken by all reduce tasks=5158912
        Map-Reduce Framework
                Map input records=4
                Map output records=4
                Map output bytes=32
                Map output materialized bytes=46
                Input split bytes=100
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=46
                Reduce input records=4
```


Applications Places System
Sun May 13, 12:12 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
Computer
acadgild's Home
Trash
README

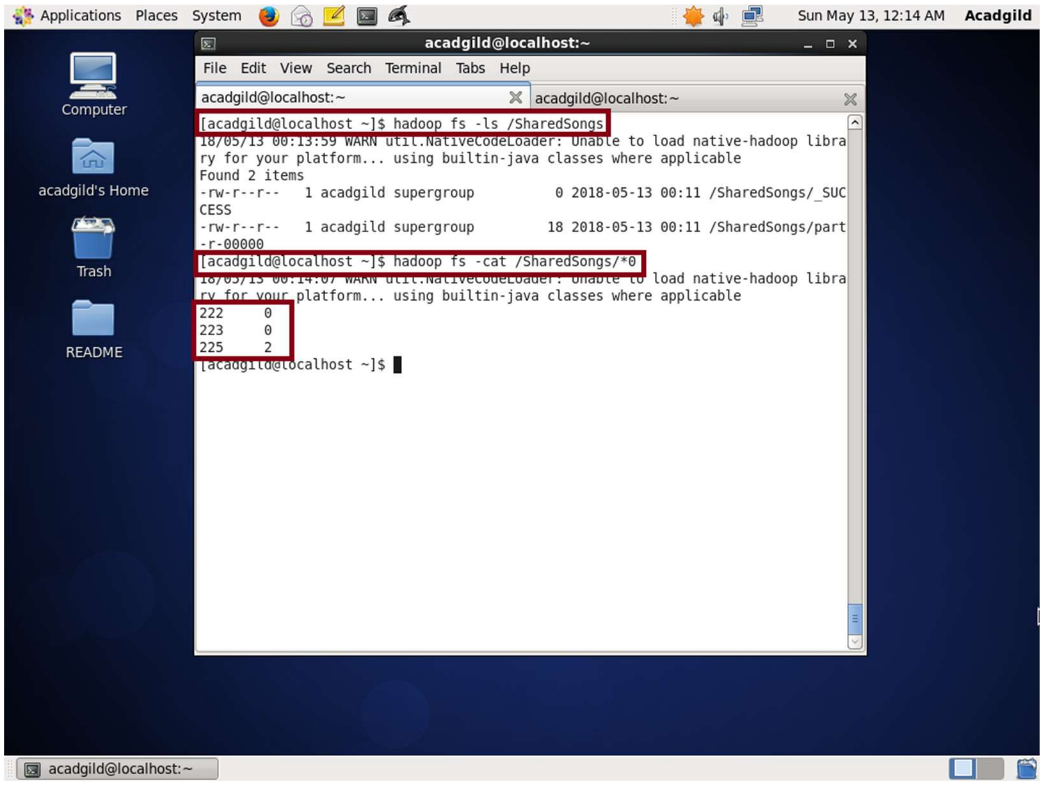
```
                Map output records=4
                Map output bytes=32
                Map output materialized bytes=46
                Input split bytes=100
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=46
                Reduce input records=4
                Reduce output records=3
                Spilled Records=8
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=131
                CPU time spent (ms)=1410
                Physical memory (bytes) snapshot=297021440
                Virtual memory (bytes) snapshot=4118204416
                Total committed heap usage (bytes)=170004480
        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=70
        File Output Format Counters
                Bytes Written=18
[acadgild@localhost ~]$
```

The Output is saved in the SharedSongs directory

Which can be listed by hadoop fs –ls / command and

By using the following command, the data in the file is shown

hadoop fs –cat /SharedSongs/*0


Applications Places System
Sun May 13, 12:14 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
Computer
acadgild's Home
Trash
README

```
[acadgild@localhost ~]$ hadoop fs -ls /SharedSongs
18/05/13 00:13:59 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2018-05-13 00:11 /SharedSongs/_SUC
CESS
-rw-r--r--   1 acadgild supergroup         18 2018-05-13 00:11 /SharedSongs/part
-r-00000
[acadgild@localhost ~]$ hadoop fs -cat /SharedSongs/*0
18/05/13 00:14:07 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
222     0
223     0
225     2
[acadgild@localhost ~]$ 
```