Assignment-5

1). Find the number of unique listeners in the data set.

Map Reduce for the above task is as below:

```
package Tasks;
```

// imported all the built-in packages required for the task

```
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

      public class Uniquelisteners
      {
         //Mapper class
         public static class Map extends Mapper <Object ,/*Input key
Type */
         Text,                /*Input value Type*/
         IntWritable,                /*Output key Type*/
         IntWritable>        /*Output value Type*/
         {
            //Map function
            public void map(Object key, Text value,
            Context context) throws IOException,
InterruptedException
            {
              IntWritable trackid = new IntWritable();
```

// Given data is splitted into an array based on the delimiter present in the data.

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

               trackid.set(Integer.parseInt(parts[1]));
```

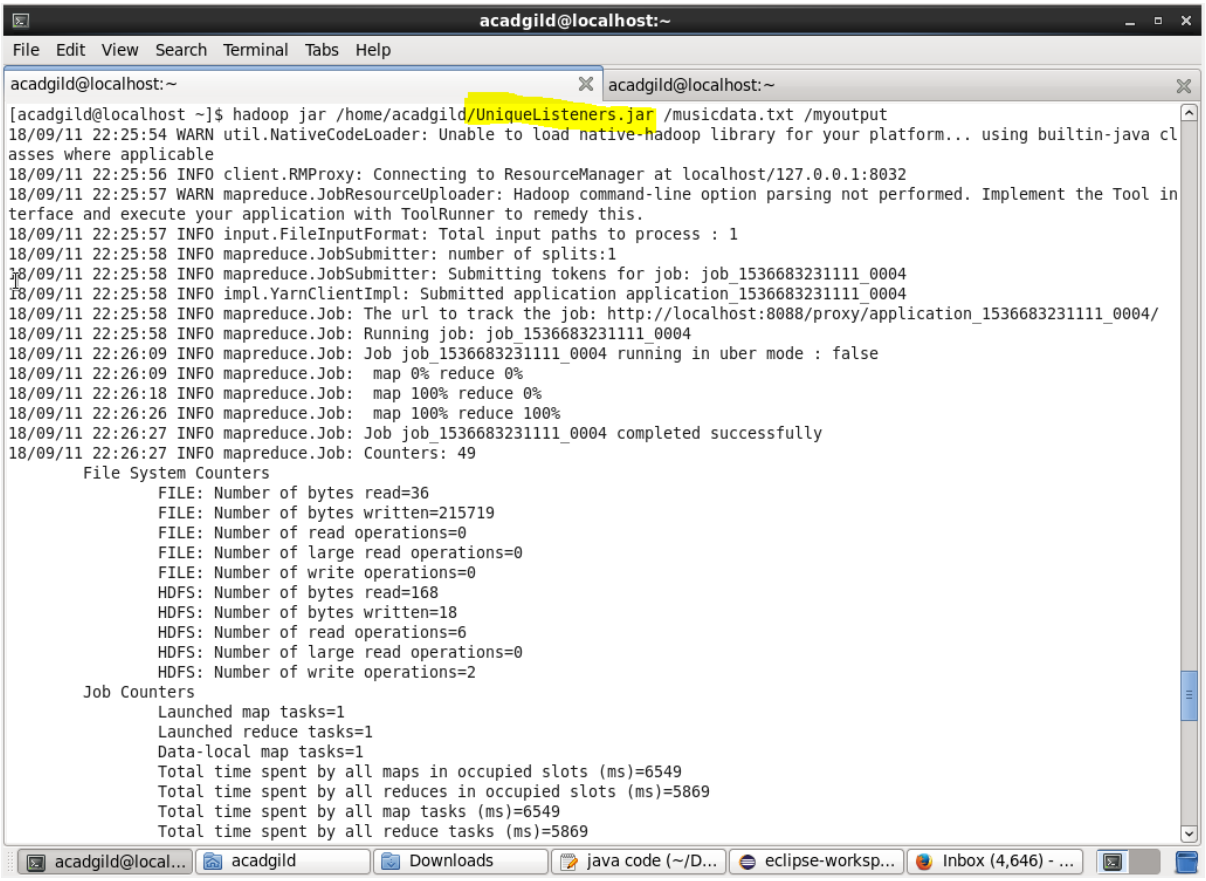
//The value(song id) in the second position of parts is set to the variable (trackid)and it sent to the output as key. The value to the key in the output of map passed as "1".

```
               context.write(trackid, new IntWritable(1));
```

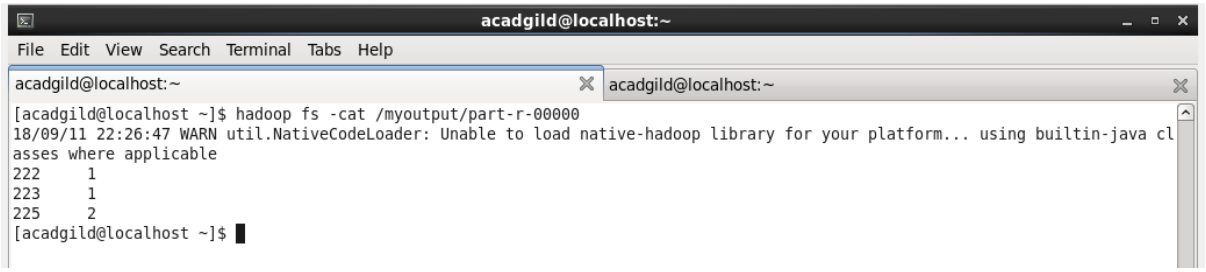
```
        System.out.println("trackId and userId" + trackid + userid);
              }
         }
         //Reducer class (Received output from the mapper as IntWritable
and intWritable)
         public static class Reduce extends Reducer< IntWritable,
IntWritable, IntWritable, IntWritable >
         {
            //Reduce function
            public void reduce( IntWritable trackid,
Iterable<IntWritable> values,Context context) throws IOException,
InterruptedException
               {
                  int sum =0;
      //the below step will check for the each value of a key.

                  for(IntWritable val:values) {

      //each value will be added to the variable "sum"

                       sum += val.get();
                  }
                  System.out.println("size of userid" + sum);

                  context.write( trackid, new IntWritable(sum));

      //Now the output will be with key (trackid) and value (count)

               }
      }

         //Main function
         public static void main(String args[])throws Exception
         {
              Configuration conf = new Configuration();


            Job job = Job.getInstance(conf, "wordcount");

            job.setJarByClass(Uniquelisteners.class);
            job.setOutputKeyClass(IntWritable.class);
            job.setOutputValueClass(IntWritable.class);
            job.setMapperClass(Map.class);
            job.setCombinerClass(Reduce.class);
            job.setReducerClass(Reduce.class);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

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

            Path outputPath = new Path(args[1]);
            outputPath.getFileSystem(conf).delete(outputPath, true);

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

```
        }
    }
```

Executed the code **UniqueListeners.jar** with input file **musicdata.txt**



Got the required output as shown in the below screenshot.



2). What are the number of times a song was heard fully.

Map Reduce for the above task is as below:

```
package Tasks;
```

// imported all the built-in packages required for the task

```
import java.io.IOException;
import java.util.ArrayList;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

     public class FullSong
     {
        //Mapper class
        public static class Map extends Mapper <LongWritable
,/*Input key Type */
        Text,                /*Input value Type*/
        IntWritable,                /*Output key Type*/
        IntWritable>        /*Output value Type*/
        {
           //Map function
           public void map(LongWritable key, Text value,
           Context context) throws IOException,
InterruptedException
           {
            IntWritable trackid = new IntWritable();
            IntWritable fullSong = new IntWritable();
```

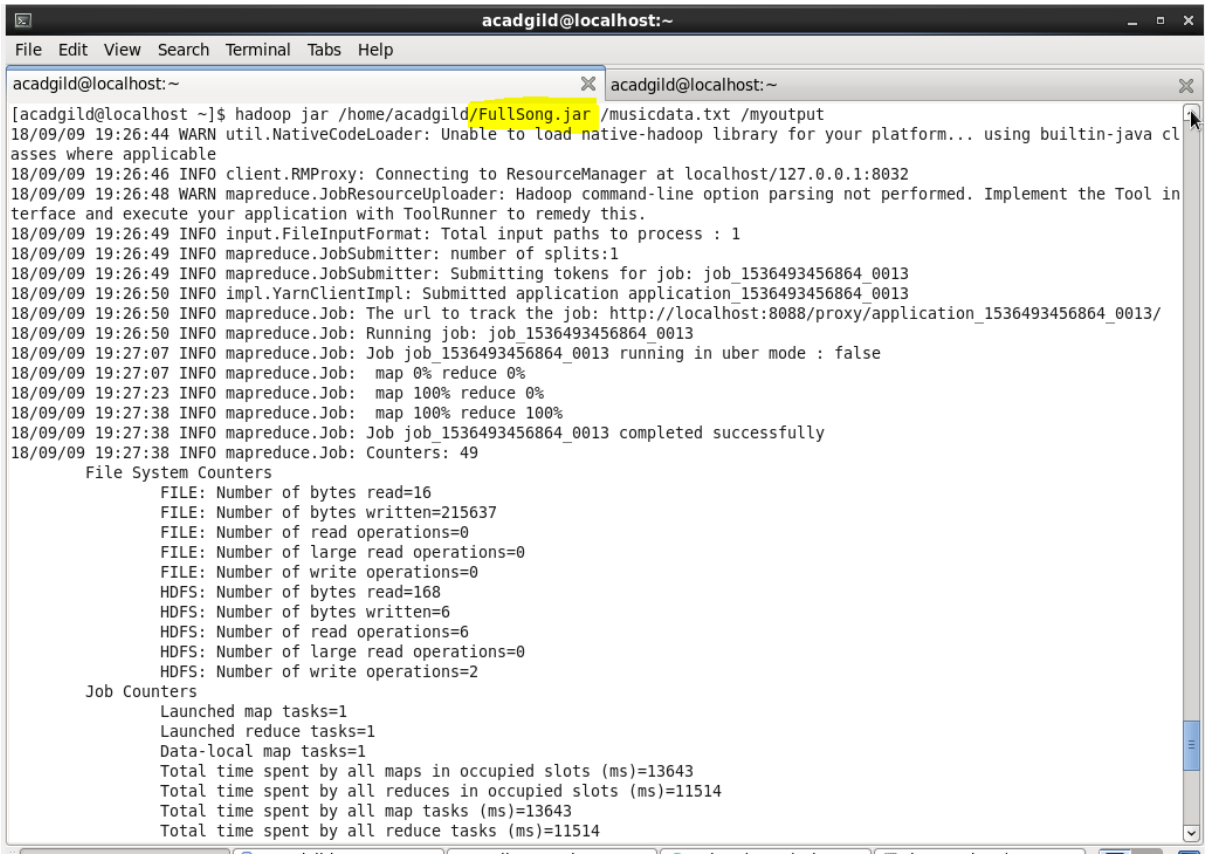
// Given data is splitted into an array based on the delimiter present in the data.

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

              if(!parts[4].equals("0")) {
```

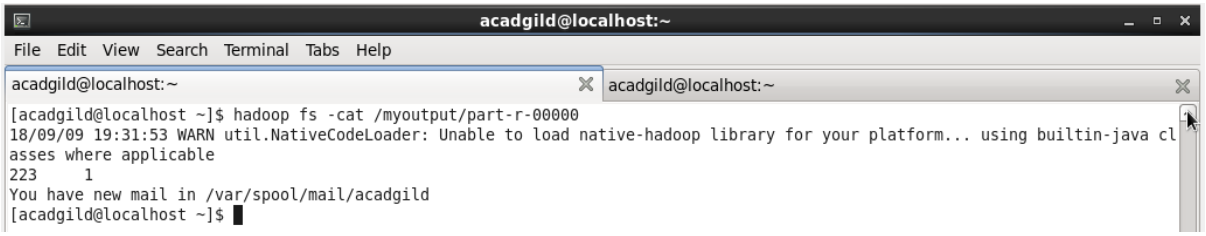
```
     //If the Full song status is not equals "0" satisfies then it
will perform the following steps.

               trackid.set(Integer.parseInt(parts[1]));
               fullSong.set(Integer.parseInt(parts[4]));
//Trackid and full song is passed as key and value to the output
respectively.
               context.write(trackid, fullSong);
               System.out.println("trackId and fullSong" +
trackid + fullSong);

            }
         }
         }

         //Reducer class (Received output from the mapper as
IntWritable and intWritable)
         public static class Reduce extends Reducer<
IntWritable, IntWritable, IntWritable, IntWritable >
         {
            //Reduce function
            public void reduce( IntWritable trackid,
Iterable<IntWritable> fullSongs,Context context) throws
IOException, InterruptedException
               {
                  int sum =0;

           //the below step will check for the each value of a key.

                  for(IntWritable fullsong:fullSongs) {

           //each value will be added to the variable "sum"
                     sum +=fullsong.get();
                }
                  System.out.println("size of userid" + sum);
                 context.write( trackid, new IntWritable(sum));
           //Now the output will be with key (trackid) and value
      (count)
               }
         }

         //Main function
         public static void main(String args[])throws Exception
         {
             Configuration conf = new Configuration();
            Job job = Job.getInstance(conf, "wordcount");

            job.setJarByClass(FullSong.class);
            job.setOutputKeyClass(IntWritable.class);
            job.setOutputValueClass(IntWritable.class);
            job.setMapperClass(Map.class);
            job.setCombinerClass(Reduce.class);
```

```
            job.setReducerClass(Reduce.class);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.addInputPath(job, new
Path(args[0]));
            FileOutputFormat.setOutputPath(job, new
Path(args[1]));
            Path outputPath = new Path(args[1]);
            outputPath.getFileSystem(conf).delete(outputPath,
true);
            System.exit(job.waitForCompletion(true)? 0 :1);
        }
    }
```

Executed the code **FullSong.jar** with input file **musicdata.txt**



Got the required output as shown in the below screenshot.



3). What are the number of times a song was shared.

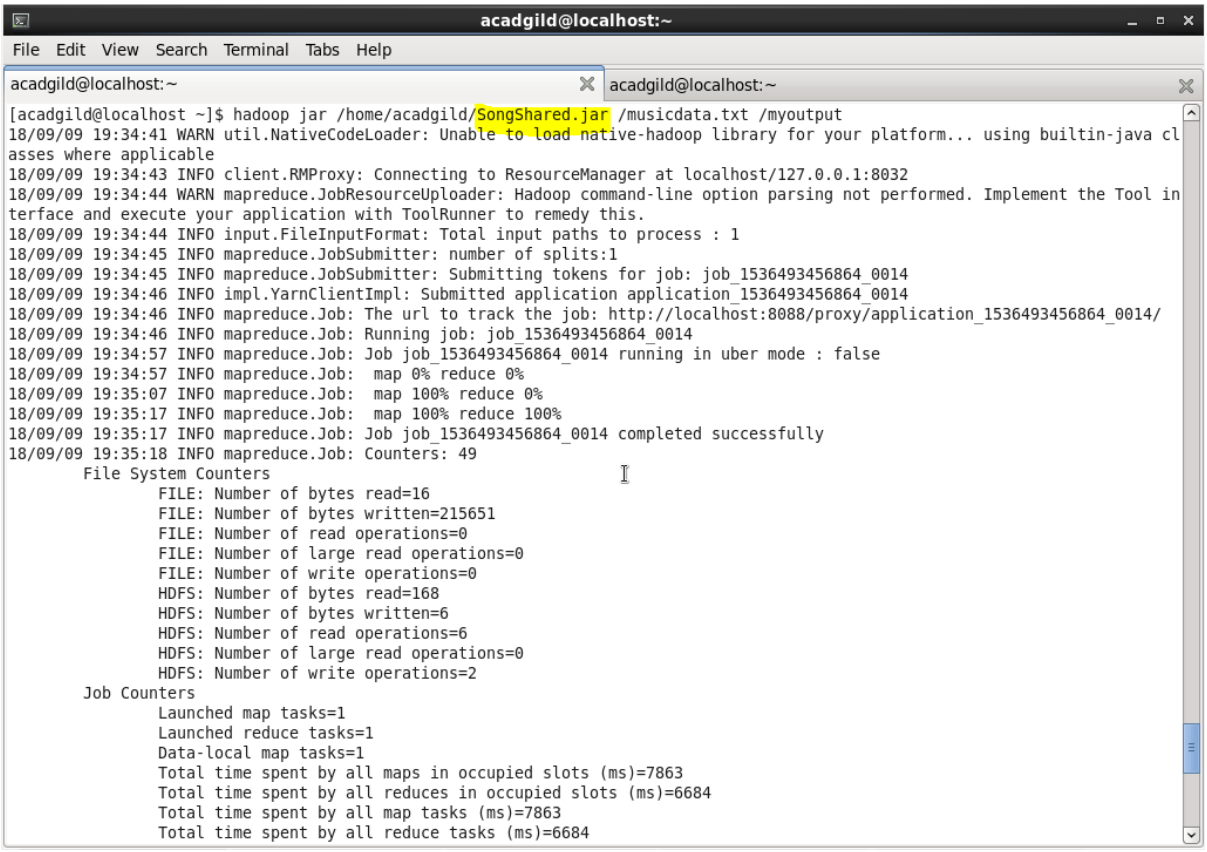
Map Reduce code for the above task is as below:

```
package Tasks;

// imported all the built-in packages required for the task

import java.io.IOException;
import java.util.ArrayList;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

     public class SongShared
     {
        //Mapper class
        public static class Map extends Mapper <LongWritable
,/*Input key Type */
        Text,                /*Input value Type*/
        IntWritable,                /*Output key Type*/
        IntWritable>        /*Output value Type*/
        {
           //Map function
           public void map(LongWritable key, Text value,
           Context context) throws IOException,
InterruptedException
           {
            IntWritable trackid = new IntWritable();
            IntWritable songShared = new IntWritable();

     // Given data is splitted into an array based on the delimiter
     present in the data.

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

              if(!parts[2].equals("0")) {

     //If the song shared status is not equals "0" satisfies then
     it will perform the following steps.
```

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

                 //Trackid and song share status is passed as key and
     value to the output respectively.

                context.write(trackid, songShared);
                System.out.println("trackId and fullSong" +
trackid + songShared);

            }
        }
        }

        //Reducer class (Received output from the mapper as
IntWritable and intWritable)
        public static class Reduce extends Reducer<
IntWritable, IntWritable, IntWritable, IntWritable >
        {
           //Reduce function
           public void reduce( IntWritable trackid,
Iterable<IntWritable> songShare,Context context) throws
IOException, InterruptedException
              {
                 int sum =0;

          //the below step will check for the each value.

                 for(IntWritable songShared:songShare) {

                //each value will be added to the variable "sum"

                    sum +=songShared.get();
                }
                 System.out.println("size of userid" + sum);
                context.write( trackid, new IntWritable(sum));

          //Now the output will be with key (trackid) and value
     (count)
              }
        }

        //Main function
        public static void main(String args[])throws Exception
        {
            Configuration conf = new Configuration();
           Job job = Job.getInstance(conf, "SongShared");

           job.setJarByClass(SongShared.class);
           job.setOutputKeyClass(IntWritable.class);
           job.setOutputValueClass(IntWritable.class);
```

```
            job.setMapperClass(Map.class);
            job.setCombinerClass(Reduce.class);
            job.setReducerClass(Reduce.class);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

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

            Path outputPath = new Path(args[1]);
        outputPath.getFileSystem(conf).delete(outputPath, true);

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

Executed the code **SongShared.jar** with input file **musicdata.txt**



Got the required output as shown in the below screenshot.