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)

Write Map Reduce program for following tasks.

Task 1

Find the number of unique listeners in the data set.

Put musicdata file to Hadoop file system

For that we can use this command

Hadoop fs –put musicdata.txt /musicdata.txt

To run Task using Hadoop we can use

Hadoop jar <Name of the jar> /location of input file /destination

To view the output we can use

Hadoop fs –cat /output/part-r-00000

Driver Class-:

```
package com.hem.hadoop.Assignment5;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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 DriverClass {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: Music Text File <input path> <output path>");
      System.exit(-1);
    }

      //Job Related Configurations
      Configuration conf = new Configuration();
      Job job = new ~~Job~~(conf, "Music Text File");
      job.setJarByClass(DriverClass.class);


    // Specify the number of reducer to 2
    //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(FullyHeardMapper.class);
      job.setMapperClass(UniqueListenersMapper.class);
      //job.setMapperClass(SongSharedMapper.class);
      //job.setReducerClass(FullyHeardReduce.class);
      job.setReducerClass(UniqueListenerReducer.class);
      //job.setReducerClass(SongSharedReducer.class);

    //Set the combiner
    //job.setCombinerClass(SongSharedReducer.class);
    //job.setCombinerClass(FullyHeardReduce.class);
    job.setCombinerClass(UniqueListenerReducer.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(Text.class);
    job.setOutputValueClass(IntWritable.class);

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

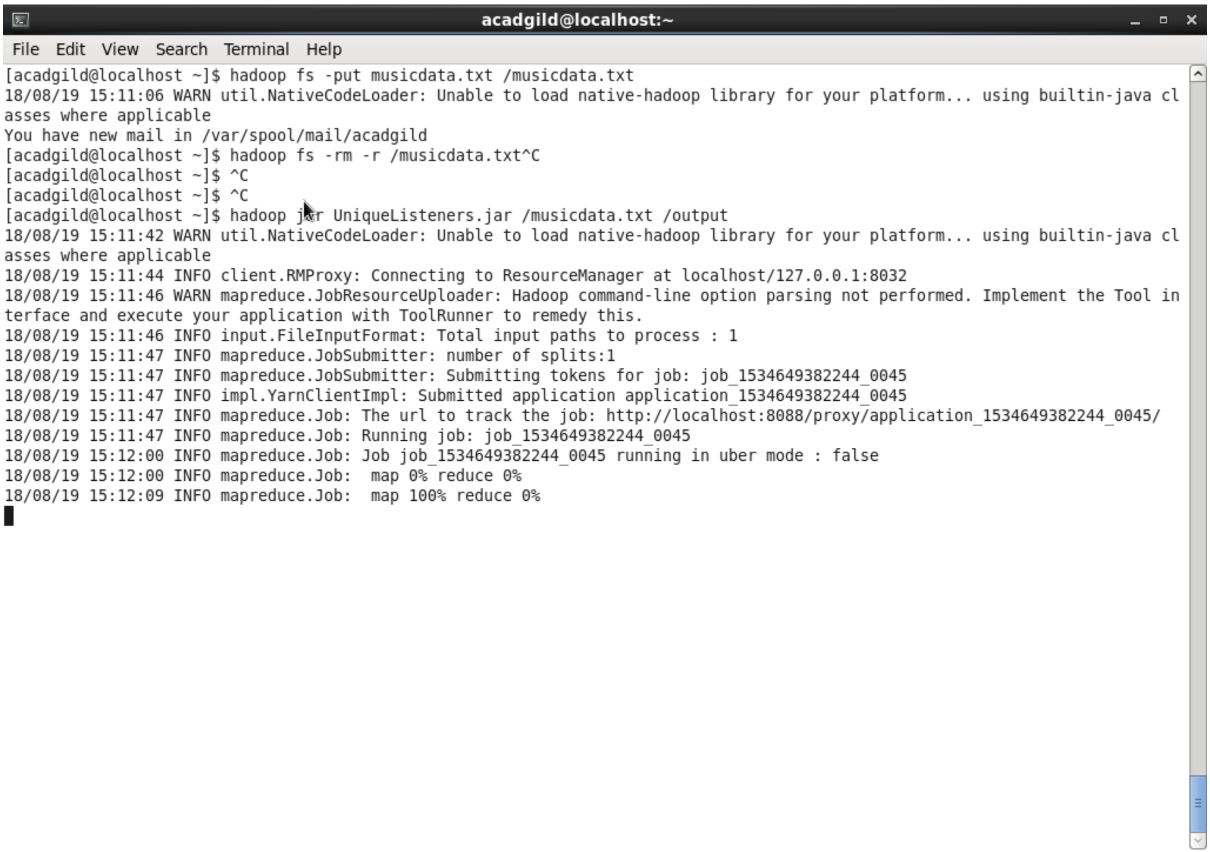
Mapper Class-:

```
package com.hem.hadoop.Assignment5;

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

	private final static IntWritable one = new IntWritable(1);
	private Text word = new Text();

	@Override
	public void map(LongWritable key, Text value, Context context) throws 
IOException, InterruptedException {
		String line = value.toString();
		System.out.println("This is output to mapper:" + key.toString());
		String words[] = line.split("\n");
		for (String wordSplit : words) {
			String tempWord[] = wordSplit.split("\\|");
			word.set(tempWord[0]);
			context.write(word, one);

		}

	}
}
```

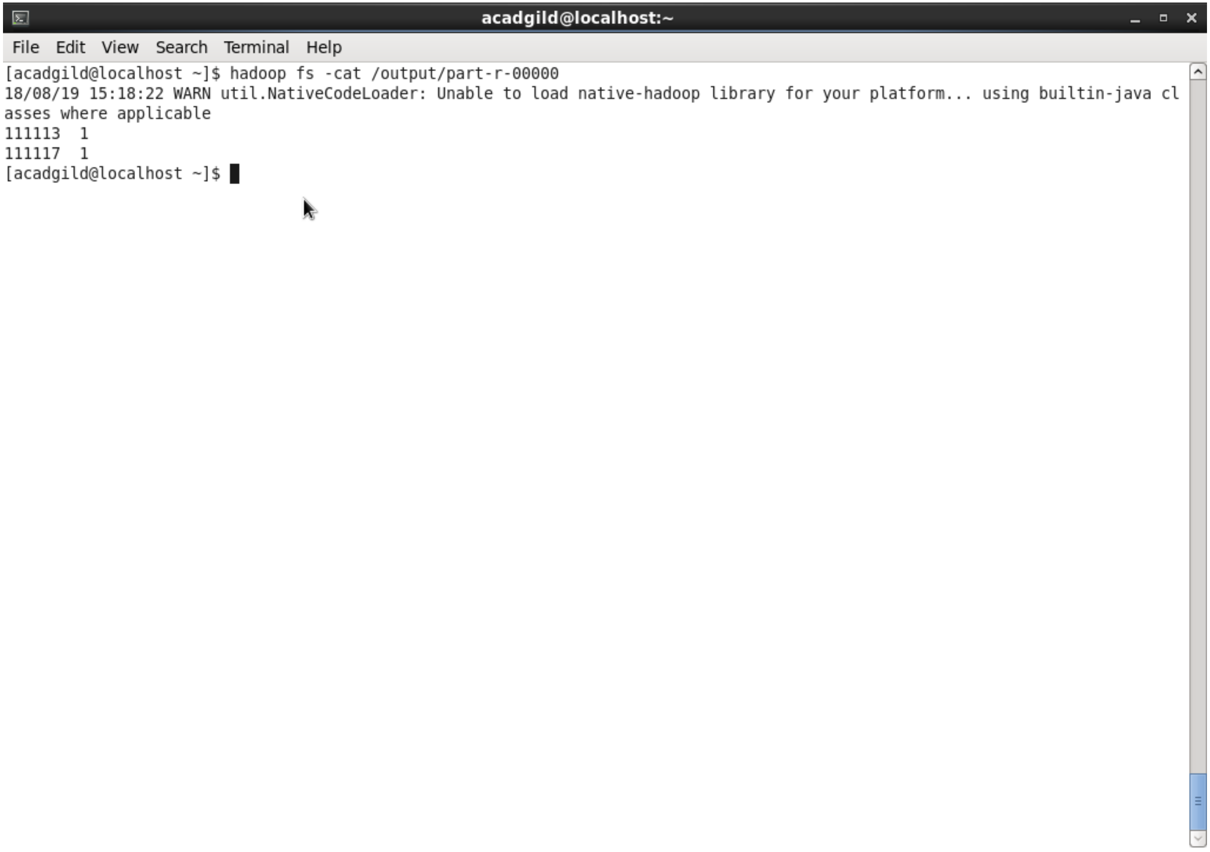
```
Reducer Class-:
package com.hem.hadoop.Assignment5;



import java.io.IOException;

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


public class UniqueListenerReducer
		extends Reducer<Text, IntWritable, Text, IntWritable> {

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

```
            System.out.println("From The Reducer=>" + key);

            int sum = 0;
            for (IntWritable value : values) {
                   sum += value.get();
            }
            if(sum<=1){
            context.write(key, new IntWritable(sum));
            }
      }
}
```

Output-:



```
[acadgild@localhost ~]$ hadoop fs -put musicdata.txt /musicdata.txt
18/08/19 15:11:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -rm -r /musicdata.txt^C
[acadgild@localhost ~]$ ^C
[acadgild@localhost ~]$ ^C
[acadgild@localhost ~]$ hadoop jar UniqueListeners.jar /musicdata.txt /output
18/08/19 15:11:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/08/19 15:11:44 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/08/19 15:11:46 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool in
terface and execute your application with ToolRunner to remedy this.
18/08/19 15:11:46 INFO input.FileInputFormat: Total input paths to process : 1
18/08/19 15:11:47 INFO mapreduce.JobSubmitter: number of splits:1
18/08/19 15:11:47 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1534649382244_0045
18/08/19 15:11:47 INFO impl.YarnClientImpl: Submitted application application_1534649382244_0045
18/08/19 15:11:47 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1534649382244_0045/
18/08/19 15:11:47 INFO mapreduce.Job: Running job: job_1534649382244_0045
18/08/19 15:12:00 INFO mapreduce.Job: Job job_1534649382244_0045 running in uber mode : false
18/08/19 15:12:00 INFO mapreduce.Job:  map 0% reduce 0%
18/08/19 15:12:09 INFO mapreduce.Job:  map 100% reduce 0%
```



```
[acadgild@localhost ~]$ hadoop fs -cat /output/part-r-00000
18/08/19 15:18:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
111113  1
111117  1
[acadgild@localhost ~]$ 
```

Task 2:

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

Driver Class-:

```
package com.hem.hadoop.Assignment5;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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 DriverClass {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: Music Text File <input path> <output path>");
      System.exit(-1);
    }

      //Job Related Configurations
      Configuration conf = new Configuration();
      Job job = new ~~Job~~(conf, "Music Text File");
      job.setJarByClass(DriverClass.class);


    // Specify the number of reducer to 2
    //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(FullyHeardMapper.class);
      //job.setMapperClass(UniqueListenersMapper.class);
      //job.setMapperClass(SongSharedMapper.class);
      job.setReducerClass(FullyHeardReduce.class);
      //job.setReducerClass(UniqueListenerReducer.class);
      //job.setReducerClass(SongSharedReducer.class);

    //Set the combiner
    //job.setCombinerClass(SongSharedReducer.class);
    job.setCombinerClass(FullyHeardReduce.class);
     // job.setCombinerClass(UniqueListenerReducer.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(Text.class);
    job.setOutputValueClass(IntWritable.class);

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

Mapper Class-:

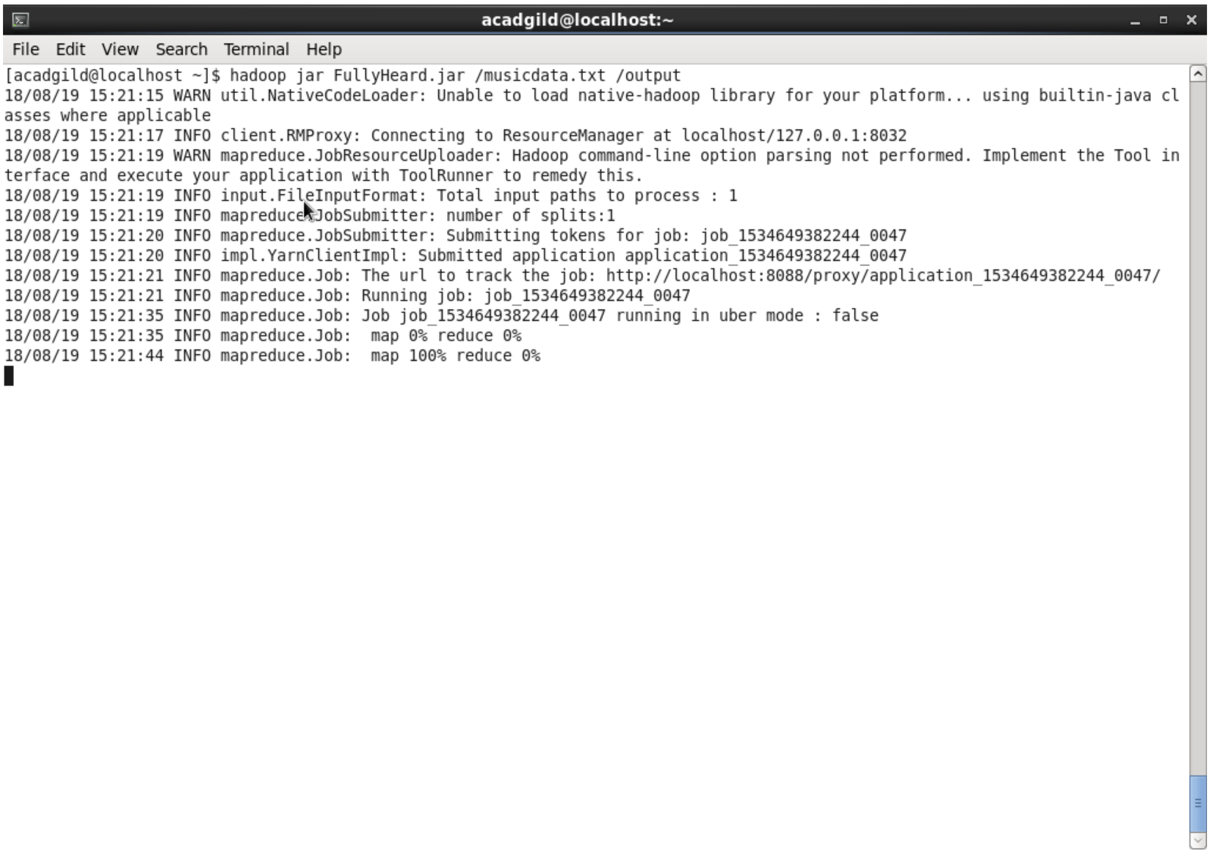
```
package com.hem.hadoop.Assignment5;


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


	private final static IntWritable one = new IntWritable(1);

	private Text word = new Text();


	@Override

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

		String line = value.toString();

		System.out.println("This is output to mapper:" + key.toString());

		String words[] = line.split("\n");

		for (String wordSplit : words) {
```

```
            String tempWord[] = wordSplit.split("\\|");
            if("1".equalsIgnoreCase(tempWord[4])){
                word.set(tempWord[0]);
                context.write(word, one);
            }


        }

    }
}
```

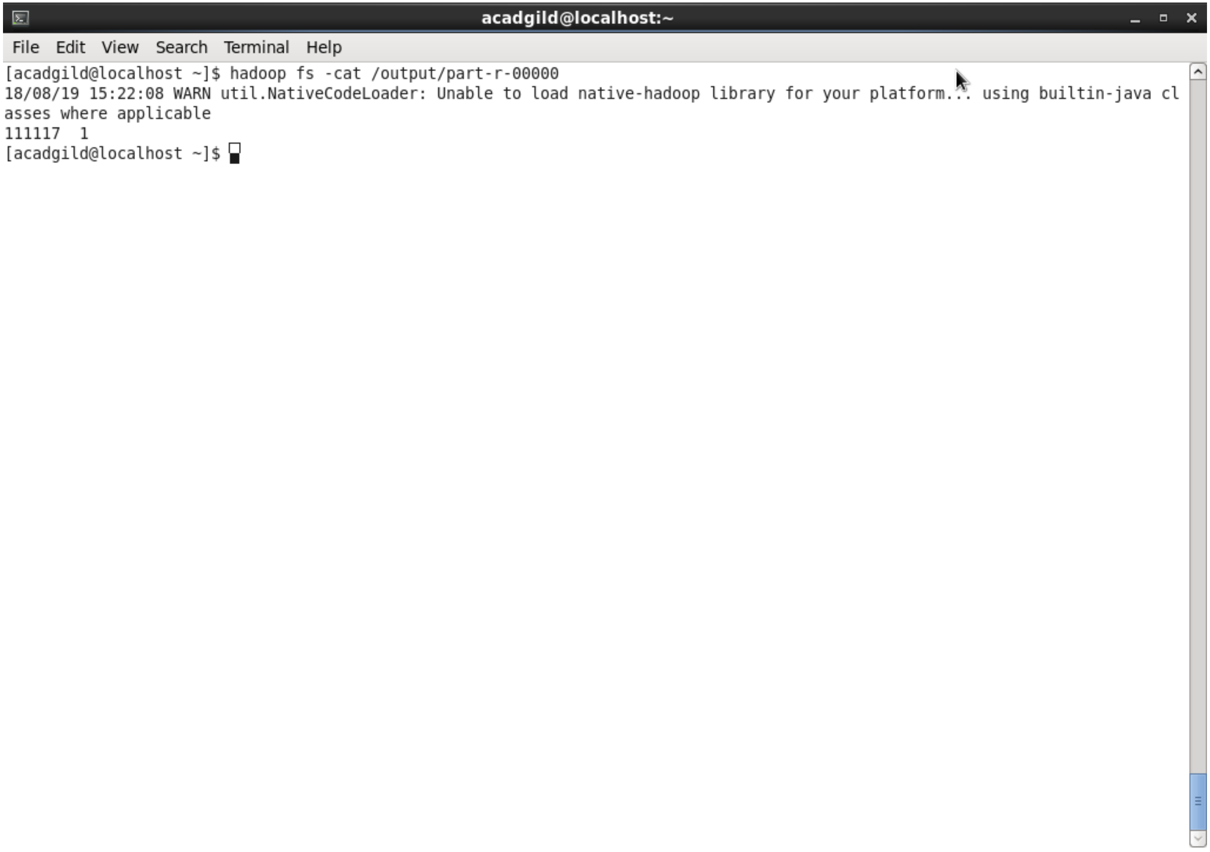
Reducer Class-:

```
package com.hem.hadoop.Assignment5;



import java.io.IOException;

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


public class FullyHeardReduce
        extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
        System.out.println("From The Reducer=>" + key);


        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }


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


    }
}
```

Output-:



```
[acadgild@localhost ~]$ hadoop jar FullyHeard.jar /musicdata.txt /output
18/08/19 15:21:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/08/19 15:21:17 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/08/19 15:21:19 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool in
terface and execute your application with ToolRunner to remedy this.
18/08/19 15:21:19 INFO input.FileInputFormat: Total input paths to process : 1
18/08/19 15:21:19 INFO mapreduce.JobSubmitter: number of splits:1
18/08/19 15:21:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1534649382244_0047
18/08/19 15:21:20 INFO impl.YarnClientImpl: Submitted application application_1534649382244_0047
18/08/19 15:21:21 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1534649382244_0047/
18/08/19 15:21:21 INFO mapreduce.Job: Running job: job_1534649382244_0047
18/08/19 15:21:35 INFO mapreduce.Job: Job job_1534649382244_0047 running in uber mode : false
18/08/19 15:21:35 INFO mapreduce.Job:  map 0% reduce 0%
18/08/19 15:21:44 INFO mapreduce.Job:  map 100% reduce 0%
```



```
[acadgild@localhost ~]$ hadoop fs -cat /output/part-r-00000
18/08/19 15:22:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
111117  1
[acadgild@localhost ~]$
```

Task 3:

What are the number of times a song was shared.

Driver Class-:

```
package com.hem.hadoop.Assignment5;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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 DriverClass {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: Music Text File <input path> <output path>");
      System.exit(-1);
    }

      //Job Related Configurations
      Configuration conf = new Configuration();
      Job job = new ~~Job~~(conf, "Music Text File");
      job.setJarByClass(DriverClass.class);


    // Specify the number of reducer to 2
    //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(FullyHeardMapper.class);
      //job.setMapperClass(UniqueListenersMapper.class);
      job.setMapperClass(SongSharedMapper.class);
      //job.setReducerClass(FullyHeardReduce.class);
      //job.setReducerClass(UniqueListenerReducer.class);
      job.setReducerClass(SongSharedReducer.class);

    //Set the combiner
    job.setCombinerClass(SongSharedReducer.class);
    //job.setCombinerClass(FullyHeardReduce.class);
```

```
     // job.setCombinerClass(UniqueListenerReducer.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(Text.class);
    job.setOutputValueClass(IntWritable.class);

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

Mapper Class-:

```
package com.hem.hadoop.Assignment5;


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


	private final static IntWritable one = new IntWritable(1);

	private Text word = new Text();


	@Override

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

		String line = value.toString();

		System.out.println("This is output to mapper:" + key.toString());

		String words[] = line.split("\n");

		for (String wordSplit : words) {

			String tempWord[] = wordSplit.split("\\|");

			if("1".equalsIgnoreCase(tempWord[2])){
```

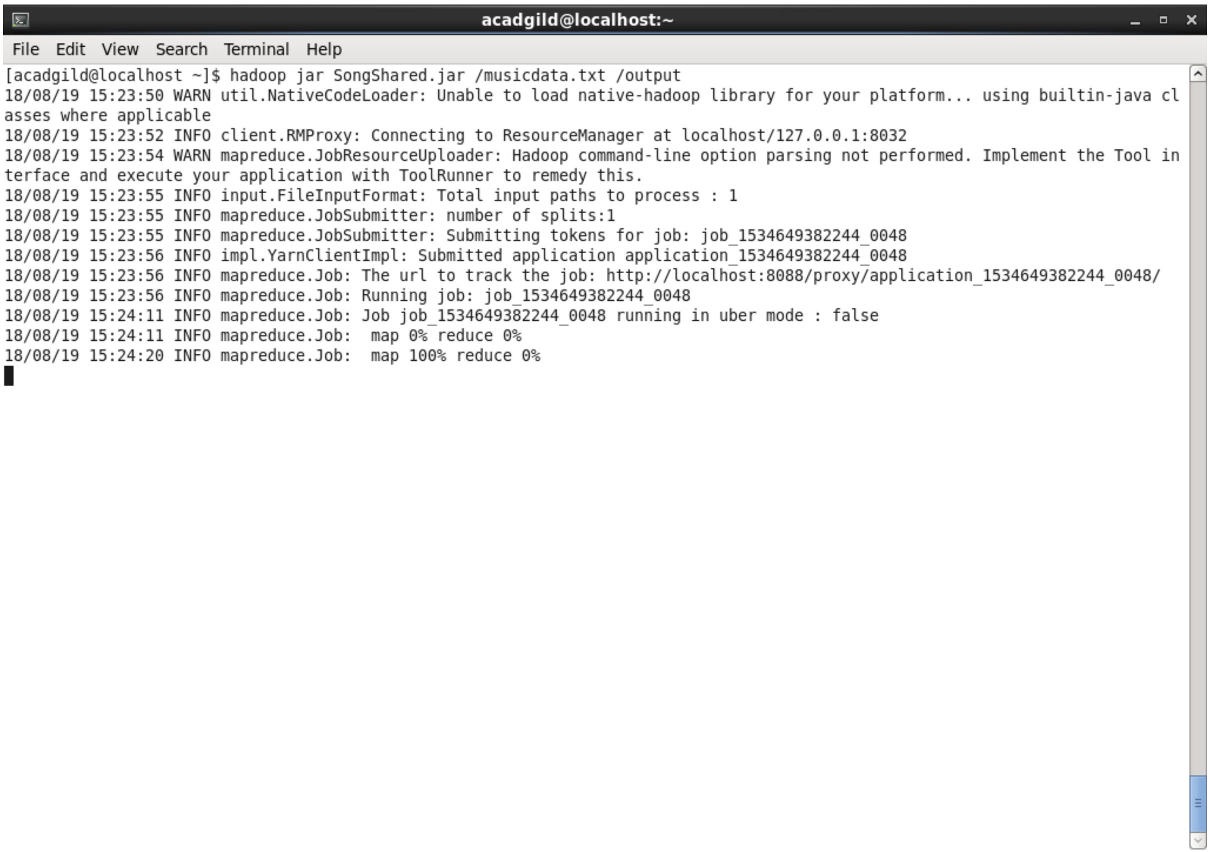
```
                    word.set(tempWord[0]);
                    context.write(word, one);
                }


            }

        }
}
```

Reducer Class-:

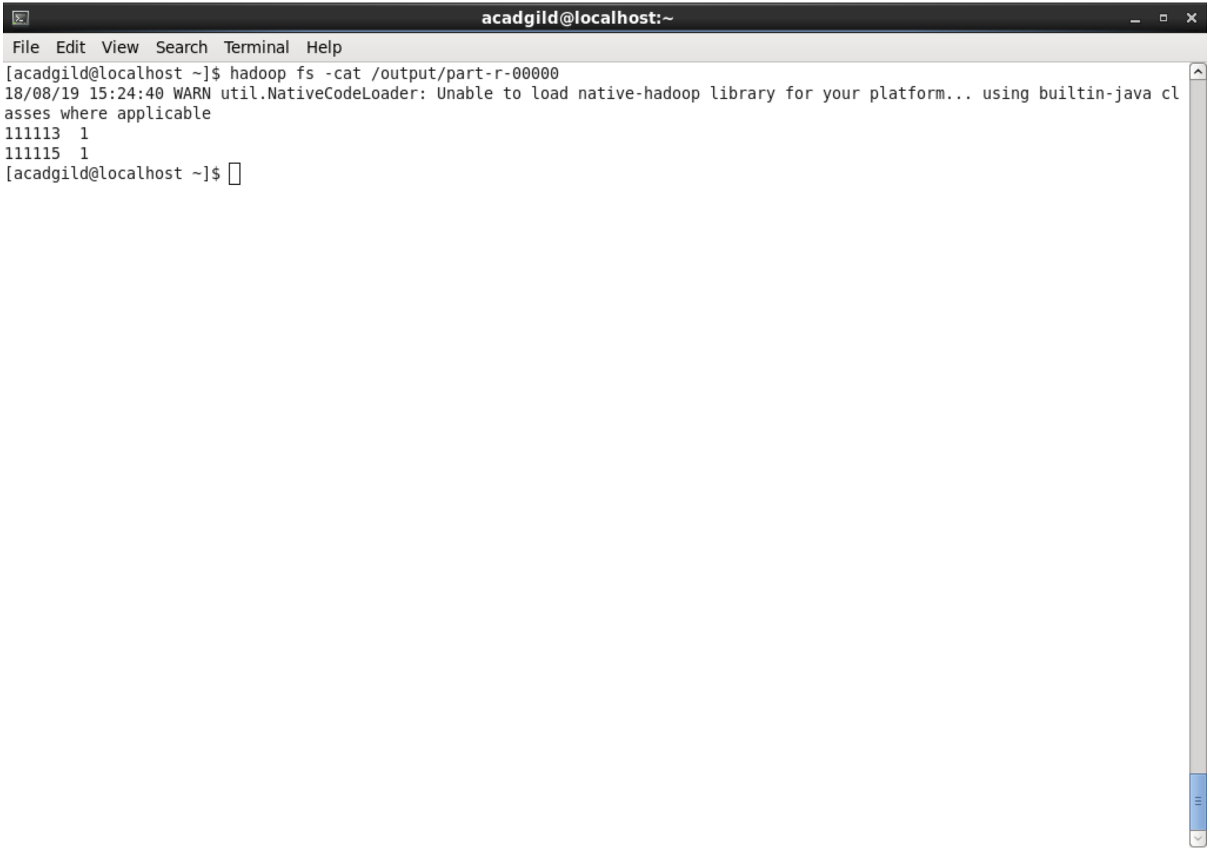
```
package com.hem.hadoop.Assignment5;



import java.io.IOException;

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


public class SongSharedReducer
                extends Reducer<Text, IntWritable, Text, IntWritable> {

        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                        throws IOException, InterruptedException {
                System.out.println("From The Reducer=>" + key);

                int sum = 0;
                for (IntWritable value : values) {
```

```
                sum += value.get();
            }

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

        }
}
```

Output-:



```
[acadgild@localhost ~]$ hadoop jar SongShared.jar /musicdata.txt /output
18/08/19 15:23:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/08/19 15:23:52 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/08/19 15:23:54 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool in
terface and execute your application with ToolRunner to remedy this.
18/08/19 15:23:55 INFO input.FileInputFormat: Total input paths to process : 1
18/08/19 15:23:55 INFO mapreduce.JobSubmitter: number of splits:1
18/08/19 15:23:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1534649382244_0048
18/08/19 15:23:56 INFO impl.YarnClientImpl: Submitted application application_1534649382244_0048
18/08/19 15:23:56 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1534649382244_0048/
18/08/19 15:23:56 INFO mapreduce.Job: Running job: job_1534649382244_0048
18/08/19 15:24:11 INFO mapreduce.Job: Job job_1534649382244_0048 running in uber mode : false
18/08/19 15:24:11 INFO mapreduce.Job:  map 0% reduce 0%
18/08/19 15:24:20 INFO mapreduce.Job:  map 100% reduce 0%
```



```
[acadgild@localhost ~]$ hadoop fs -cat /output/part-r-00000
18/08/19 15:24:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
111113  1
111115  1
[acadgild@localhost ~]$ 
```