Assignment 5

1. Map-reduce program to list the unique listeners

User ID	Track ID	SHARED status	Platform	Skip status
111115	222	0	1	0
111113	225	1	0	0
111117	223	0	1	1
111115	225	1	0	0

```
import java.io.IOExecption; // Importing all the packages for path, config,
                                mapreduce &IO formats


public class uniqueListeners {          // main class

 public class SongConstants {  // defining some constants to move through records

            public static final int userid = 0;
            public static final int Trackid = 1;
            public static final int shared = 2;
            public static final int platform  = 3;
            public static final int Skipped = 4;

                 // 0 represents user ID
            }
```

```
      public static class uniqueListenersMapper extends   //mapper class
              Mapper<object, Text, IntWritable, Intwritable>
      {
            Intwritable Trackid = new Intwritable();
            Intwritable userid = new Intwritable();
// defining the type of input

Public void map(Object key, Text value,
              Mapper<Object, Text, Intwritable, Intwritable>.Context context)
              Throws IOException, Interrupted Exception {
String[] parts = value.toString().split("[|]");
//splitting the string because it separated by"|" //
TrackID.set(Integer.parseInt(parts[SongConstants.Trackid]));
TrackID.set(Integer.parseInt(parts[SongConstants.userid]));
Context.write(trackid,userid);
// dumping userid & trackid in the mapper function//
}
}
Public static class UniqueListnersReduce extends  //reducer class//
   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);
}
}
```

// moving through every userid for a particular track id and adding it to the hashset which copies only one copy of every id, so the duplicates wont exisit. The size will give the count of users for the respective trackid//

```
Public static void main (String[] args) throws Exception {
Configuration conf = new Configuration();
Job job =  Job.getInstance(conf, “UniqueListeners”);
Job.setMapperClass(Map.class);          //set mapper classs//
Job.setReducerClass(Reduce.class);       // set reducer class//

Job.setOutputKeyClass(IntWritable.class);
Job.setOutputvalueClass(IntWritable.class);
//setting outputkey and value format//

Job.setInputFormatKeyClass(ObjectInputFormatclass);
Job.setOutputFormatKeyClass(IntwritableInputFormatclass);
//setting inputkey and value format//

Path outputpath = new Path (args[1]);

FileInputFormat.addInputpath(job, new Path(args[0]));
FileOutputFormat.setoutputpath(job, new Path (args[1]));
System.exit(Job.waitForCompletion(true) ? 0 : 1);
```

```
}}
```