🔧 **Step 1: Create Working Directory**

bash

Copy code

```
mkdir ~/musicstats
cd ~/musicstats
```

📄 **Step 2: Create Java Files**

Create the following three Java files:

---

1. MusicMapper.java

java

Copy code

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;


public class MusicMapper extends Mapper<LongWritable, Text, Text, Text> {
   public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      if (key.get() == 0 && value.toString().contains("UserId")) return; // Skip header


      String[] fields = value.toString().split(",");
      if (fields.length >= 3) {
         String userId = fields[0].trim();
         String trackId = fields[1].trim();
```

```
            String shared = fields[2].trim();

            context.write(new Text("U_" + trackId), new Text(userId)); // For unique listeners
            if (shared.equals("1")) {
                context.write(new Text("S_" + trackId), new Text("1")); // For shared count
            }
        }
    }
}
```

---

2. MusicReducer.java

java

Copy code

```
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MusicReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
        String prefix = key.toString().substring(0, 2);
        String trackId = key.toString().substring(2);

        if (prefix.equals("U_")) {
```

```
            Set<String> uniqueUsers = new HashSet<>();
            for (Text val : values) {
                uniqueUsers.add(val.toString());
            }
            context.write(new Text(trackId), new Text("UniqueListeners: " + uniqueUsers.size()));
        } else if (prefix.equals("S_")) {
            int count = 0;
            for (Text val : values) {
                count += Integer.parseInt(val.toString());
            }
            context.write(new Text(trackId), new Text("SharedCount: " + count));
        }
    }
}
```

---

3. MusicDriver.java

java

Copy code

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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.output.FileOutputFormat;

public class MusicDriver {
```

```
    public static void main(String[] args) throws Exception {
        if (args.length < 2) {
            System.out.println("Usage: MusicDriver <input path> <output path>");
            System.exit(-1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Music Stats");

        job.setJarByClass(MusicDriver.class);
        job.setMapperClass(MusicMapper.class);
        job.setReducerClass(MusicReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

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

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

⚙️ **Step 3: Compile Java Code**

bash

Copy code

javac -classpath `hadoop classpath` -d . Music*.java

📦 Step 4: Create a JAR File

bash

Copy code

jar cf musicstats.jar *.class

📁 Step 5: Prepare Input Dataset

Create a sample CSV and upload to HDFS:

bash

Copy code

cat <<EOF > music_dataset.csv

UserId,TrackId,Shared,Radio,Skip

111115,222,0,1,0

111113,225,1,0,0

111117,223,0,1,1

111115,225,1,0,0

111116,225,0,1,0

111117,225,1,1,0

111118,225,0,1,0

111119,225,0,1,0

111119,225,1,0,1

111120,225,0,1,0

111117,222,1,1,0

111121,225,0,1,0

111122,226,0,1,1

111117,223,0,1,1

111118,226,0,0,1

111119,223,1,1,0

111120,226,1,0,0

111121,223,1,1,0

111122,224,1,0,0

111123,228,0,1,0

EOF

hadoop fs -mkdir -p /music/input

hadoop fs -put music_dataset.csv /music/input/

🏃 Step 6: Run the MapReduce Job

bash

Copy code

hadoop jar musicstats.jar MusicDriver /music/input /music/output

---

📊 Step 7: View Output

bash

Copy code

hadoop fs -cat /music/output/part-r-00000

✅ Expected Output Format

yaml

Copy code

222 UniqueListeners: 2

223 UniqueListeners: 3

223 SharedCount: 2

224 UniqueListeners: 1

224 SharedCount: 1

225 UniqueListeners: 7

225 SharedCount: 3

226 UniqueListeners: 3

226 SharedCount: 1

228 UniqueListeners: 1

🔁 **Rerun Job? Clean Output Folder**

bash

Copy code

hadoop fs -rm -r /music/output

📝 **Notes**

- Each reducer key is prefixed (U_ or S_) to distinguish listener and share data.
- The reducer strips the prefix and returns counts accordingly.