📄 Step 1: Create Working Directory (Optional)

bash

Copy code

```
mkdir ~/music_csv_dataset

cd ~/music_csv_dataset
```

📂 Step 2: Prepare Input Data

Create a CSV dataset in a file named music_data.csv.

bash

Copy code

```
echo -e "UserId,TrackId,Shared,Radio,Skip\n111115,222,0,1,0\n111113,225,1,0,0\n111117,223,0,1,1\n111115,225,1,0,0\n111116,225,0,1,0\n111117,225,1,1,0" > music_data.csv
```

Upload it to HDFS:

bash

Copy code

```
hadoop fs -mkdir -p /music/input

hadoop fs -put music_data.csv /music/input/
```

🔧 Step 3: Create Java Files for CSV Parsing

We will now update the Mapper class to correctly parse CSV data.

1. MusicMapper.java

java

Copy code

```
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.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class MusicMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
   public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException {
      // Ignore the header row
      String line = value.toString();
      if (line.contains("UserId")) return; // Skip header

      String[] fields = line.split(",");

      // Extract TrackId, Radio and Skip columns
      String trackId = fields[1];
      int radio = Integer.parseInt(fields[3]);
      int skip = Integer.parseInt(fields[4]);

      // Emit (trackId_radio) and (trackId_skip)
      output.collect(new Text(trackId + "_radio"), new IntWritable(radio));
      output.collect(new Text(trackId + "_skip"), new IntWritable(skip));
   }
}
```

2. MusicReducer.java

java

Copy code

```
import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;


public class MusicReducer extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {

   public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable>
output, Reporter rep) throws IOException {

      int count = 0;

      while (values.hasNext()) {

         count += values.next().get();

      }

      output.collect(key, new IntWritable(count));

   }

}
```

3. MusicDriver.java

java

Copy code

```
import java.io.IOException;
```

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class MusicDriver extends Configured implements Tool {
   public int run(String args[]) throws IOException {
      if (args.length < 2) {
         System.out.println("Please provide input and output paths");
         return -1;
      }

      JobConf conf = new JobConf(MusicDriver.class);
      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));
      conf.setMapperClass(MusicMapper.class);
      conf.setReducerClass(MusicReducer.class);
      conf.setMapOutputKeyClass(Text.class);
      conf.setMapOutputValueClass(IntWritable.class);
      conf.setOutputKeyClass(Text.class);
```

```
        conf.setOutputValueClass(IntWritable.class);


        JobClient.runJob(conf);
        return 0;
    }

    public static void main(String args[]) throws Exception {
        int exitCode = ToolRunner.run(new MusicDriver(), args);
        System.out.println(exitCode);
    }
}
```

⚖️ Step 4: Compile Java Files

Compile the Java files with Hadoop classpath:

bash

Copy code

```
javac -classpath `hadoop classpath` -d . MusicMapper.java MusicReducer.java MusicDriver.java
```

🗓️ Step 5: Create a JAR File

Create a JAR file for the MapReduce job:

bash

Copy code

```
jar cf music_processing_csv.jar *.class
```

🏃 Step 6: Run the MapReduce Job

Run the MapReduce job:

bash

Copy code

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

---

📊 Step 7: View the Output

After the job finishes, view the output:

bash

Copy code

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

Expected Output:

Copy code

222_radio 1

223_radio 2

225_radio 3

225_skip 1

225_skip 1

This output will show the number of times each track was listened to on the radio and the number of times it was skipped.

✅ Notes

- Make sure the Hadoop daemons (HDFS, MapReduce) are running.
- If you need to re-run the job, remove the output directory first:

bash

Copy code

hadoop fs -rm -r /music/output

End of Music Dataset Processing with CSV Assignment - Code + Commands