Session 5: ADVANCE MAP REDUCE AND INTRODUCTION TO UNIX CONCEPTS

Assignment 5

Task 1:

- Find the number of unique listeners in the data set.

  CODE written for the resolution:

```
package task1_Assignment5;


import java.io.IOException;

import java.util.HashSet;

import java.util.Set;


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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class UniqueListeners {

        private enum COUNTERS {

                INVALID_RECORD_COUNT

        }


        public static void main(String[] args) throws Exception {
```

```java
Configuration conf = new Configuration();
if (args.length != 2) {
        System.err.println("Usage: uniquelisteners <in> <out>");
        System.exit(2);
}
Job job = new Job(conf, "Unique listeners per track");
job.setJarByClass(UniqueListeners.class);
job.setMapperClass(UniqueListenersMapper.class);
job.setReducerClass(UniqueListenersReducer.class);
job.setOutputKeyClass(IntWritable.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
org.apache.hadoop.mapreduce.Counters counters = job.getCounters();
System.out.println("No. of Invalid Records :"
                + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)
                        .getValue());
}


public static class UniqueListenersReducer extends
                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) {
```

```java
                        userIdSet.add(userId.get());

                }

                IntWritable size = new IntWritable(userIdSet.size());

                context.write(trackId, size);

        }

}


public static class UniqueListenersMapper extends

                Mapper<Object, Text, IntWritable, IntWritable> {


        IntWritable trackId = new IntWritable();

        IntWritable userId = new IntWritable();


        public void map(Object key, Text value,

                        Mapper<Object, Text, IntWritable, IntWritable>.Context context)

                        throws IOException, InterruptedException {


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

                trackId.set(Integer.parseInt(parts[1]));

                userId.set(Integer.parseInt(parts[0]));


                if (parts.length == 5) {

                        context.write(trackId, userId);

                } else {

                        // add counter for invalid records

context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);

                }


        }

    }

}
```

Output

Command used to run the user-count.jar created:

```
Tou nave new mait in /var/spoot/mait/acaagitu
[acadgild@localhost ~]$ hadoop jar /home/acadgild/user_count.jar /user/acadgild/
hadoop/musicdata.txt /user/acadgild/hadoop/user_countpertrackoutput1
18/12/12 05:00:07 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
```

Output of the jar file execution seen as contents of file ''part-r-00000''of the directory "user_countpertrackoutput1" :

```
You nave new mail in /var/spool/mail/acaagild
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/hadoop/user_countpertracko
utput1/part-r-00000
18/12/12 05:00:56 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
222     1
223     1
225     2
[acadgild@localhost ~]$ █
```

Task 2:

- What are the number of times a song was heard fully?

Code written for the task:

package task1_Assignment5;

import java.io.IOException;

import java.util.HashSet;

import java.util.Set;

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

```java
public class SongFullyHeard {

  private enum COUNTERS {

    INVALID_RECORD_COUNT

  }


  public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();

    Job job = new Job(conf, "No. of times fully heard per track");

    job.setJarByClass(SongFullyHeard.class);

    job.setMapperClass(SongFullyHeardMapper.class);

    job.setReducerClass(SongFullyHeardReducer.class);

    job.setOutputKeyClass(IntWritable.class);

    job.setOutputValueClass(IntWritable.class);

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

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

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

    org.apache.hadoop.mapreduce.Counters counters = job.getCounters();

    System.out.println("No. of Invalid Records :"

        + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)

            .getValue());

  }


  public static class SongFullyHeardReducer extends

      Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {


    public void reduce(

        IntWritable trackId,

        Iterable<IntWritable> heardStatuses,

        Reducer<IntWritable, IntWritable, IntWritable, IntWritable>.Context context)

        throws IOException, InterruptedException {
```

```java
        Set<Integer> heardStatusSet = new HashSet<Integer>();

        for (IntWritable heardStatus : heardStatuses) {

            if(heardStatus.equals(new IntWritable(1))) {

                heardStatusSet.add(heardStatus.get());

            }

        }

        IntWritable size = new IntWritable(heardStatusSet.size());

        context.write(trackId, size);

    }

}


public static class SongFullyHeardMapper extends

        Mapper<Object, Text, IntWritable, IntWritable> {


    IntWritable trackId = new IntWritable();

    IntWritable heardStatus = new IntWritable();


    public void map(Object key, Text value,

            Mapper<Object, Text, IntWritable, IntWritable>.Context context)

            throws IOException, InterruptedException {


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

        trackId.set(Integer.parseInt(parts[1]));

        heardStatus.set(Integer.parseInt(parts[4]));


        if (parts.length == 5) {

            context.write(trackId, heardStatus);

        } else {

            // add counter for invalid records

            context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);

        }
```

```
    }

  }

}
```

OUTPUT:

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/hadoop/heard_pertrack1/par
t-r-00000
18/12/12 06:34:36 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
222     0
223     1
225     0
```

Task 3:

- What are the number of times a song was shared?

Code written for execution of task:

package task1_Assignment5;

import java.io.IOException;

import java.util.HashSet;

import java.util.Set;

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class SongShareCount {

```java
private enum COUNTERS {

  INVALID_RECORD_COUNT

}


public static void main(String[] args) throws Exception {

  Configuration conf = new Configuration();


  Job job = new Job(conf, "No. of times per track is shared");

  job.setJarByClass(SongShareCount.class);

  job.setMapperClass(SongShareCountMapper.class);

  job.setReducerClass(SongShareCountReducer.class);

  job.setOutputKeyClass(IntWritable.class);

  job.setOutputValueClass(IntWritable.class);

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

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

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

  org.apache.hadoop.mapreduce.Counters counters = job.getCounters();

  System.out.println("No. of Invalid Records :"

      + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)

          .getValue());

}


public static class SongShareCountReducer extends

    Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {


  public void reduce(

      IntWritable trackId,

      Iterable<IntWritable> sharedStatuses,

      Reducer<IntWritable, IntWritable, IntWritable, IntWritable>.Context context)

      throws IOException, InterruptedException {
```

```java
      Set<Integer> sharedStatusSet = new HashSet<Integer>();

      for (IntWritable sharedStatus : sharedStatuses) {

        if(sharedStatus.equals(new IntWritable(1))) {

          sharedStatusSet.add(sharedStatus.get());

        }

      }

      IntWritable size = new IntWritable(sharedStatusSet.size());

      context.write(trackId, size);

    }

}


public static class SongShareCountMapper extends

      Mapper<Object, Text, IntWritable, IntWritable> {


  IntWritable trackId = new IntWritable();

  IntWritable sharedStatus = new IntWritable();


  public void map(Object key, Text value,

      Mapper<Object, Text, IntWritable, IntWritable>.Context context)

      throws IOException, InterruptedException {


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

    trackId.set(Integer.parseInt(parts[1]));

    sharedStatus.set(Integer.parseInt(parts[2]));


    if (parts.length == 5) {

      context.write(trackId, sharedStatus);

    } else {

      // add counter for invalid records

      context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);

    }
```

```
        }

    }

}
```

OUTPUT:

```
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/hadoop/shared_pertrack/par
t-r-00000
18/12/12 06:55:43 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
222     0
223     0
225     1
```