DSBDA MOCK

Name : Apurva Anil Bodake
Roll No. : 33307

Program :

```
package PackageDemo;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;
import org.apache.hadoop.util.GenericOptionsParser;

public class LoginTimeMinMax {
   public static void main(String[] args) throws Exception {
      Configuration c = new Configuration();
      String[] files = new GenericOptionsParser(c, args).getRemainingArgs();
      Path input = new Path(files[0]);
      Path output = new Path(files[1]);
      Job j = Job.getInstance(c, "logInOutTime");
      j.setJarByClass(LoginTimeMinMax.class);
      j.setMapperClass(MapForWordCount.class);
      j.setReducerClass(ReduceForWordCount.class);
      j.setOutputKeyClass(Text.class);
      j.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(j, input);
      FileOutputFormat.setOutputPath(j, output);
      System.exit(j.waitForCompletion(true) ? 0 : 1);
   }

   public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable> {
      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         String alllines = value.toString();
         String[] lines = alllines.split("\n");
         for (String line : lines) {
            String[] word = line.split(",");
            if (word.length > 7) {
               String login = word[5];
               String logout = word[7];
               String ip = word[1];
               String[] loginPart = login.split(" ");
               String[] logoutPart = logout.split(" ");
```

```
                if (loginPart.length > 1 && logoutPart.length > 1) {
                    String[] loginTimePart = loginPart[1].split(":");
                    String[] logoutTimePart = logoutPart[1].split(":");
                    int loginHour = Integer.parseInt(loginTimePart[0]);
                    int loginMinute = Integer.parseInt(loginTimePart[1]);
                    int logoutHour = Integer.parseInt(logoutTimePart[0]);
                    int logoutMinute = Integer.parseInt(logoutTimePart[1]);
                    int loginTotalMinutes = loginHour * 60 + loginMinute;
                    int logoutTotalMinutes = logoutHour * 60 + logoutMinute;
                    int differenceInSeconds = (logoutTotalMinutes - loginTotalMinutes) * 60;
                    Text outputKey = new Text(ip.toUpperCase().trim());
                    IntWritable outputValue = new IntWritable(differenceInSeconds);
                    context.write(outputKey, outputValue);
                }
            }
        }
    }

    public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
        private int maxTime = Integer.MIN_VALUE;
        private int minTime = Integer.MAX_VALUE;
        private Text maxIp = new Text();
        private Text minIp = new Text();

        public void reduce(Text word, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
            int localMax = Integer.MIN_VALUE;
            int localMin = Integer.MAX_VALUE;
            for (IntWritable value : values) {
                int loginTime = value.get();
                if (loginTime > localMax) {
                    localMax = loginTime;
                }
                if (loginTime < localMin) {
                    localMin = loginTime;
                }
            }
            if (localMax > maxTime) {
                maxTime = localMax;
                maxIp.set(word);
            }
            if (localMin < minTime) {
                minTime = localMin;
                minIp.set(word);
            }
        }

        protected void cleanup(Context context) throws IOException, InterruptedException {
            context.write(new Text("Max IP: " + maxIp.toString()), new IntWritable(maxTime));
            context.write(new Text("Min IP: " + minIp.toString()), new IntWritable(minTime));
```

```
        }
    }
}
```

Output :

```
[cloudera@quickstart 33307]$ hadoop fs -ls MockDemo2Dir1Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2025-03-26 00:27 MockDemo2Dir1/_SUCCESS
-rw-r--r--   1 cloudera cloudera          0 2025-03-26 00:27 MockDemo2Dir1/part-r-00000
[cloudera@quickstart 33307]$ hadoop fs -cat MockDemo2Dir1/part-r-00000
Max IP: 10.10.15.204  24467
Min IP: 10.10.15.117  2
```