**Big Data Analytics**

**BAD601**

**Experiment 3:**

Develop a Map Reduce program that mines weather data and displays appropriate messages indicating

the weather conditions of the day.

**Open Notepad and write**

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

import java.io.IOException;

public class WeatherAnalysis {

// Mapper Class

public static class WeatherMapper extends Mapper<Object, Text, Text, Text> {

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

// Skip header line

if (line.startsWith("Date")) {

return;

}

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

if (parts.length < 3) {

return; // Ignore malformed lines

}

String date = parts[0]; // Date column

String maxTempStr = parts[2]; // Max temperature column

try {

context.write(new Text(date), new Text(maxTempStr));

} catch (NumberFormatException e) {

// Skip invalid temperature values

}

}

}

// Reducer Class

public static class WeatherReducer extends Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

for (Text val : values) {

float maxTemp = Float.parseFloat(val);

String condition;

// Determine weather condition

if (maxTemp > 30) {

condition = "Hot Day";

} else if (maxTemp < 10) {

condition = "Cold Day";

} else {

condition = "Normal Day";

}

context.write(key, condition);

}

}

}

// Driver Class (Main Method)

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

if (args.length != 2) {

System.err.println("Usage: WeatherAnalysis <input path> <output path>");

System.exit(-1);

}

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Weather Condition Analysis");

job.setJarByClass(WeatherAnalysis.class);

job.setMapperClass(WeatherMapper.class);

job.setReducerClass(WeatherReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

**Save the Program:**

Click of Save and save the program with the name WeatherMapper.java

**Make a CSV file and save as weather\_data.csv:**

Date,Location,MaxTemperature,MinTemperature

2025-03-01,NewYork,32,20

2025-03-02,NewYork,8,-2

2025-03-03,NewYork,25,15

**Change the directory from cmd:**

Go to the directory where the code is saved using cd

**Get the Hadoop class path:**

hadoop classpath

**Compile the Java File**

Now, compile your Java program using the copied classpath:

javac -classpath "PASTE\_HADOOP\_CLASSPATH\_HERE" -d . WeatherMapper.java

**Create a JAR File**

Once compilation is successful, create the JAR file:

jar cf weather.jar WeatherAnalysis\*.class

**Upload Input Files to HDFS**

start-dfs.cmd

start-yarn.cmd

hdfs dfs -mkdir -p /user/hadoop/weather

hdfs dfs -put weather\_data.csv /user/hadoop/weather/

**Run the Hadoop Job**

hadoop jar weather.jar WeatherAnalysis /user/hadoop/weather/weather\_data.csv /user/hadoop/weather\_output

**View the Output**

hdfs dfs -cat /user/hadoop/weather\_output/part-\*