EXPERIMENT - 3

Aim:

Develop a MapReduce program to find the maximum temperature in each year given a weather data file.

Theory:

Weather datasets often contain massive logs of temperature readings indexed by date. The problem of finding the maximum temperature for every year is a classic aggregation problem. In the Map phase, the program parses each line of the dataset to extract the 'Year' (Key) and the 'Temperature' (Value). The Shuffle and Sort phase groups these values by Year. In the Reduce phase, the reducer iterates through the list of temperatures for each specific year, identifies the maximum value, and writes the final (Year, MaxTemperature) pair.

Description:

The program uses a Mapper class to read raw text input. It takes the line offset as the key and the line string as the value. It parses the string (assuming NCDC format or similar csv) to extract the year and the air temperature, emitting <Year, Temperature>. The Reducer class receives <Year, List<Temperatures>>. It iterates through the list, compares values to find the maximum, and writes the result to the output file.

Code:

```
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;

public class MaxTemperature {

    // 1. Mapper Class
    public static class MaxTemperatureMapper extends Mapper<LongWritable,
Text, Text, IntWritable> {

        private static final int MISSING = 9999;

        @Override
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
```

```
            String line = value.toString();
            // Assuming data format: Year (0-4), Temp (last part) or Space
separated
            // For simplicity, let's assume input is: "1950 25"
            String[] parts = line.split("\s+");

            if (parts.length >= 2) {
                String year = parts[0];
                int airTemperature = Integer.parseInt(parts[1]);

                if (airTemperature != MISSING) {
                    context.write(new Text(year), new
IntWritable(airTemperature));
                }
            }
        }
    }

    // 2. Reducer Class
    public static class MaxTemperatureReducer extends Reducer<Text,
IntWritable, Text, IntWritable> {

        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
            int maxValue = Integer.MIN_VALUE;

            for (IntWritable value : values) {
                maxValue = Math.max(maxValue, value.get());
            }

            context.write(key, new IntWritable(maxValue));
        }
    }

    // 3. Driver / Main Method
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: MaxTemperature <input path> <output
path>");
            System.exit(-1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Max Temperature");

        job.setJarByClass(MaxTemperature.class);
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);

        job.setOutputKeyClass(Text.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);
    }
}
```

Input:

A text file named input.txt containing the following text (Year Temperature):

```
root@DESKTOP-0MNJIUH:~/maxtemp# hdfs dfs -cat /user/root/maxtemp/input/temp.txt
2019 35
2019 42
2019 38
2020 30
2020 45
2020 41
2021 33
2021 47
2021 40
```

Output:

The content of the part-r-00000 file generated by Hadoop:

```
root@DESKTOP-0MNJIUH:~/maxtemp# hdfs dfs -cat /user/root/maxtemp/output/part-r-00000
2019    42
2020    45
2021    47
```

Commands:

Description	Command
Create input directory in HDFS:	hdfs dfs -mkdir /weather_input
Upload the local text file to HDFS:	hdfs dfs -put input.txt /weather_input
Compile the Java program:	javac -classpath $(hadoop classpath) -d . MaxTemperature.java
Create the JAR file:	jar -cvf maxtemp.jar *.class
Run the Hadoop Job:	hadoop jar maxtemp.jar MaxTemperature /weather_input /weather_output
View the Output:	hdfs dfs -cat /weather_output/part-r-00000

Conclusion:

In this experiment, we successfully developed and executed a MapReduce program to calculate the maximum temperature per year. We understood how the Mapper extracts numerical data associated with a key (Year), and how the Reducer aggregates this data to find the maximum value.