# EXPERIMENT - 7

## Aim:

Develop a MapReduce program to classify days as 'Sunny' or 'Cool' based on a temperature threshold.

## Theory:

Data classification is a fundamental task in Big Data processing. In this experiment, we utilize the MapReduce framework to filter and label data records based on a specific condition. The 'Map' phase processes individual weather records, extracts the temperature, and applies a conditional threshold (e.g., Temperature > 25°C). Based on this comparison, the data is tagged with a label ('Sunny' or 'Cool'). The 'Reduce' phase aggregates these records to produce a final list of dates and their corresponding weather classification.

## Description:

The program reads a text file containing Date and Temperature (in Celsius). We define a threshold of 25°C. The Mapper emits the Date as the key. For the value, it checks if the temperature is greater than 25. If yes, it sets the value as 'Sunny'; otherwise, it sets it as 'Cool'. The Reducer simply writes the Date and the Classification to the output file.

## Code:

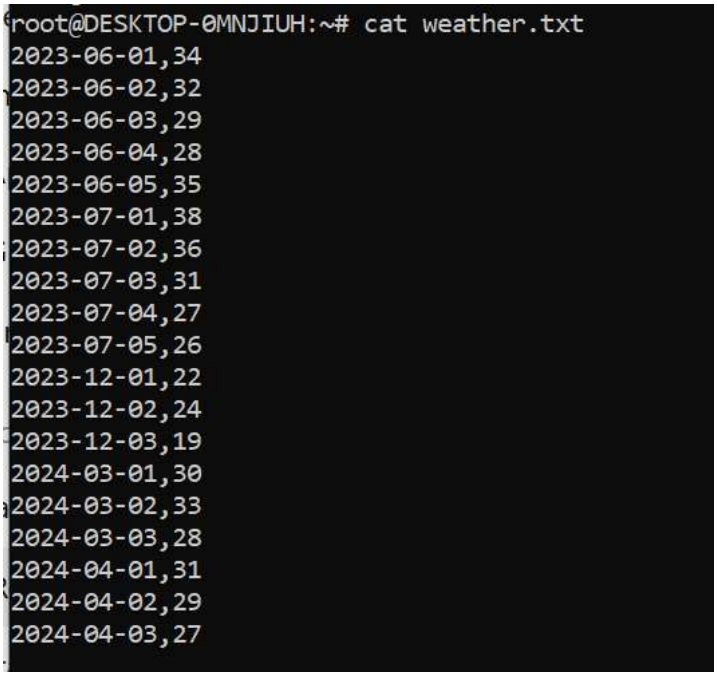
```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WeatherClassification {

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

        private Text date = new Text();
        private Text classification = new Text();

        // Threshold set to 25 degrees
        private static final int THRESHOLD = 25;

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

```
            String line = value.toString();
            // Input Format: Date,Temperature (e.g., "2024-01-01, 15")
            String[] parts = line.split(",");

            if (parts.length >= 2) {
                String dateStr = parts[0].trim();
                try {
                    int temp = Integer.parseInt(parts[1].trim());
                    date.set(dateStr);

                    if (temp > THRESHOLD) {
                        classification.set("Sunny");
                    } else {
                        classification.set("Cool");
                    }

                    context.write(date, classification);

                } catch (NumberFormatException e) {
                    // Handle bad data
                }
            }
        }
    }

    // 2. 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 {
            // Since there is only one classification per date, we just write
the first one found.
            for (Text val : values) {
                context.write(key, val);
            }
        }
    }

    // 3. Driver Method
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Weather Classification");

        job.setJarByClass(WeatherClassification.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);
    }
}
```

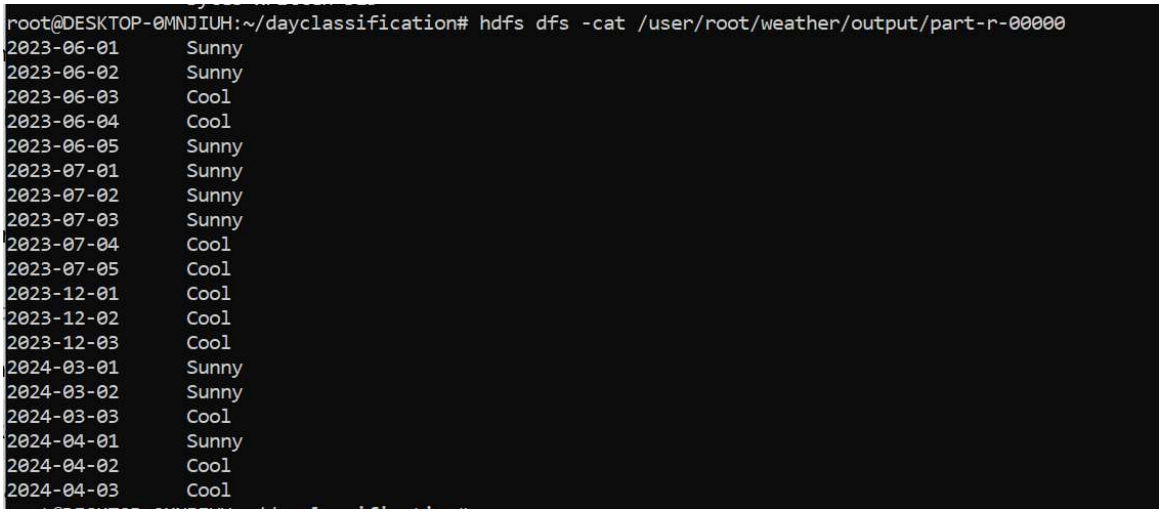
## Input:

A text file named weather_data.txt containing (Date, Temp):




## Output:

The content of the output file generated by Hadoop:



## Commands:

| Description | Command |
|---|---|
| Create input directory: | hdfs dfs -mkdir /weather_input |
| Upload input file: | hdfs dfs -put weather_data.txt /weather_input |
| Compile Java program: | javac -classpath $(hadoop classpath) -d . WeatherClassification.java |
| Create JAR file: | jar -cvf weather.jar *.class |
| Run Hadoop Job: | hadoop jar weather.jar WeatherClassification /weather_input /weather_output |
| View Output: | hdfs dfs -cat /weather_output/part-r-00000 |

## Conclusion:

In this experiment, we successfully implemented a MapReduce program to classify weather data. We learned how to use conditional logic within the Mapper to assign labels to data based on numerical thresholds (Temperature), demonstrating the capability of Hadoop for simple data transformation and classification tasks.