LAB Exercise 5

Aim: Write a Map Reduce program that mines weather data for analysis with MapReduce, since it is semi structured and record oriented. Find average, max and min temperature for each year in NCDC data set?

Dataset:

Reference: https://www.data.gov.in/catalog/all-india-seasonal-and-annual-temperature-series?page=2

Data is collected from **India Meteorological Department (IMD).** The India Meteorological Department (IMD) is the principal agency in India responsible for meteorological observations, weather forecasting, and seismology. Founded in 1875, IMD operates under the Ministry of Earth Sciences and provides critical data and analyses for weather forecasting, climate monitoring, and research. IMD's extensive network of meteorological stations across the country helps gather reliable data on temperature, rainfall, humidity, and other atmospheric parameters, supporting agriculture, disaster management, aviation, and general weather information for public awareness and planning.

Code:

WeatherDriver.java

```java
package org.ankit;

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

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

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

      job.setJarByClass(WeatherDriver.class);
      job.setMapperClass(WeatherMapper.class);
      job.setReducerClass(WeatherReducer.class);
      job.setOutputKeyClass(IntWritable.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);
   }
}
```

WeatherMapper.java

```java
package org.ankit;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class WeatherMapper extends Mapper<Object, Text, IntWritable, Text> {
   private IntWritable year = new IntWritable();
   private Text tempData = new Text();

   @Override    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
      String line = value.toString();
      String[] fields = line.split("\t");

      if (fields.length == 13) { // Ensure data row has 13 fields (year + 12 months)
         try {
            year.set(Integer.parseInt(fields[0])); // Parse year
            StringBuilder temperatures = new StringBuilder();

            // Collect temperatures of each month
            for (int i = 1; i <= 12; i++) {
               temperatures.append(fields[i]).append(",");
            }

            // Remove trailing comma and write year and temperature data to context
            tempData.set(temperatures.toString().replaceAll(",$", ""));
            context.write(year, tempData);
         } catch (NumberFormatException e) {
            // Log and ignore if there's any parsing error
         }
      }
   }
}
```

WeatherReducer.java

```java
package org.ankit;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class WeatherReducer extends Reducer<IntWritable, Text, IntWritable, Text> {
   private Text result = new Text();

   @Override    protected void reduce(IntWritable key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
      double minTemp = Double.MAX_VALUE;
      double maxTemp = Double.MIN_VALUE;
      double sumTemp = 0.0;
      int count = 0;

      for (Text val : values) {
         String[] temps = val.toString().split(",");
         for (String temp : temps) {
            double tempValue = Double.parseDouble(temp);
            minTemp = Math.min(minTemp, tempValue);
            maxTemp = Math.max(maxTemp, tempValue);
            sumTemp += tempValue;
            count++;
         }
      }
```

```
        double avgTemp = sumTemp / count;
        result.set("Min: " + minTemp + ", Max: " + maxTemp + ", Avg: " + avgTemp);
        context.write(key, result);
    }
}
```

Output:

```
Select Administrator: Command Prompt

C:\hadoop\sbin>hadoop fs -put "D:\NITJ\BDA\EXP 5\dataset.txt" /weather

C:\hadoop\sbin>hadoop fs -cat /weather/output/part-r-00000
1901    Min: 17.99, Max: 28.6, Avg: 24.224166666666665
1902    Min: 18.78, Max: 28.68, Avg: 24.326666666666664
1903    Min: 18.29, Max: 28.41, Avg: 23.980833333333333
1904    Min: 17.77, Max: 27.85, Avg: 23.849999999999998
1905    Min: 17.4, Max: 28.69, Avg: 23.9325
1906    Min: 17.5, Max: 29.06, Avg: 24.099999999999998
1907    Min: 18.46, Max: 27.66, Avg: 24.007499999999997
1908    Min: 18.15, Max: 28.49, Avg: 23.95
1909    Min: 17.79, Max: 27.97, Avg: 23.778333333333336
1910    Min: 18.05, Max: 28.36, Avg: 23.760833333333338
1911    Min: 18.52, Max: 28.55, Avg: 23.954166666666666
1912    Min: 18.44, Max: 28.53, Avg: 24.105833333333337
1913    Min: 18.2, Max: 27.95, Avg: 23.882500000000004
1914    Min: 18.73, Max: 28.46, Avg: 24.034166666666668
1915    Min: 17.93, Max: 29.17, Avg: 24.288333333333338
1916    Min: 18.17, Max: 28.16, Avg: 24.075000000000003
1917    Min: 18.16, Max: 27.41, Avg: 23.57083333333333
1918    Min: 17.25, Max: 28.27, Avg: 23.825833333333332
1919    Min: 18.58, Max: 28.3, Avg: 24.012500000000003
1920    Min: 18.27, Max: 27.76, Avg: 23.90833333333333
1921    Min: 18.61, Max: 30.78, Avg: 24.584166666666672
1922    Min: 18.3, Max: 29.8, Avg: 24.22416666666666
1923    Min: 18.23, Max: 28.95, Avg: 24.0475
1924    Min: 18.06, Max: 28.6, Avg: 24.140833333333337
1925    Min: 17.53, Max: 27.82, Avg: 23.95
1926    Min: 18.65, Max: 28.97, Avg: 24.055000000000003
```

```
Select Administrator: Command Prompt
1996    Min: 18.96, Max: 28.37, Avg: 24.657500000000002
1997    Min: 17.86, Max: 28.33, Avg: 24.254166666666666
1998    Min: 18.84, Max: 29.18, Avg: 24.694999999999997
1999    Min: 18.32, Max: 28.13, Avg: 24.66083333333333
2000    Min: 18.87, Max: 28.92, Avg: 24.576666666666668
2001    Min: 18.5, Max: 29.46, Avg: 24.81416666666667
2002    Min: 18.78, Max: 29.56, Avg: 25.03166666666667
2003    Min: 18.6, Max: 29.01, Avg: 24.7
2004    Min: 18.83, Max: 28.12, Avg: 24.799166666666668
2005    Min: 19.07, Max: 28.99, Avg: 24.576666666666668
2006    Min: 19.96, Max: 28.82, Avg: 25.0125
2007    Min: 19.24, Max: 28.84, Avg: 24.7675
2008    Min: 18.49, Max: 28.42, Avg: 24.6125
2009    Min: 19.79, Max: 28.77, Avg: 25.1125
2010    Min: 19.15, Max: 29.19, Avg: 25.189166666666665
2011    Min: 18.32, Max: 28.92, Avg: 24.710833333333326
2012    Min: 18.25, Max: 28.91, Avg: 24.678333333333338
2013    Min: 18.88, Max: 29.06, Avg: 24.819999999999997
2014    Min: 18.81, Max: 29.42, Avg: 24.732499999999998
2015    Min: 19.02, Max: 28.82, Avg: 24.912499999999998
2016    Min: 20.92, Max: 30.41, Avg: 26.454166666666662
2017    Min: 20.59, Max: 30.47, Avg: 26.291666666666668
2018    Min: 19.9, Max: 29.99, Avg: 25.896666666666665
2019    Min: 19.62, Max: 30.27, Avg: 25.857499999999998

C:\hadoop\sbin>
```

Conclusion:

In this experiment we learnt about the manipulation of text and csv data. And how to apply MapReduce technique on this type of data.