DBMSProject(skila001,skila001@ucr.edu,Srikar Kilambi)

First I joined the Weatherstationlocations.csv file with the year txt file in the hdfs.I did this join with state as the key and date, temperature as the value.It took about 1.5 minutes.

```
SuppressWarnings("deprecation")
ublic class AvgMonthly {
  //JOB1
  public static class MapJoinMapper extends Mapper<LongWritable, Text, Text, Text>{
      Text outkey = new Text();
      Text outvalue = new Text();
      HashMap<Integer,String> userdata=new HashMap<Integer,String>();
      HashMap<String,String> userdata1=new HashMap<String,String>();
      protected void setup(Context context) throws IOException{
          Path[] files=DistributedCache.getLocalCacheFiles(context.getConfiguration());
          for(Path file : files) {
              if(file.getName().equals("WeatherStationLocations.csv")) {
                  BufferedReader reader =new BufferedReader(new FileReader(file.toString()));
                  String line =reader.readLine();
                  int id1=0;
                  while(line != null)
                  {
                      if(line.contains("\"USAF\",\"WBAN\",\"STATION NAME\",\"CTRY\",\"STATE\",\"LAT\",\"LON\",\"ELEV(M)\",\"BEGIN\",\"END\""))
                      {
                          line=reader.readLine();
                          continue;
                      }
                      else {
                          String[] line1= line.toString().split(",");
                          //String cols[] = str.split(",");
                          String id=line1[0].replaceAll("\"","");
                          if(id.substring(0,1).contains("A"))
                              break;
```

In the next job I computed the average for each month in each state. This took about 1 seconds

```
                    reader.close();
                }
            }
        }
    public void map(LongWritable key ,Text value, Context context) throws IOException, InterruptedException {
        String value1=value.toString().replaceAll("\\s+"," ");
        if(value1.contains("STN--- WBAN YEARMODA TEMP DEWP SLP STP VISIB WDSP MXSPD GUST MAX MIN PRCP SNDP FRSHTT"))
            return;
        else
        {
        String cols[]=value1.toString().split(" ");
        int id= Integer.parseInt(cols[0]);

        String date=cols[2].substring(4,6);
        String temp=cols[3];
        //String place= userdata.get(useerid);
        String state=userdata.get(id);
        if(state==null) {
            return;
            }
        outkey.set(state+" "+date);
        outvalue.set(temp);
        context.write(outkey,outvalue);
        }

    }
    }

    public static class mapReducer extends Reducer <Text,Text,Text,Text >
     {
    public void reduce(Text key,  Iterable<Text>values, Context context) throws IOException,InterruptedException

    {
```

```
    public void reduce(Text key,  Iterable<Text>values, Context context) throws IOException,InterruptedException
 99
100     {
101
102         Text outvalue=new Text();
103
104         //outvalue.set(key);
105         //outvalue1.set(values);
106         Text outkey = new Text();
107         //String val1=null;
108         double val3=0;
109         //Iterator<Text> iterator = values.iterator(); //Iterating
110         int count=0;
111         //double value = 0;
112         for(Text value: values) {
113             String value1=value.toString();
114             //String val1=value1[1];
115             String val2=value1;
116             //val1=val2;
117             //val3=Double.parseDouble(val1);
118             val3=val3+Double.parseDouble(val2);
119             count=count+1;
120         }
121         double avg=val3/count;
122         outvalue.set(String.valueOf(avg));
123         outkey.set(key);
124         context.write(outkey, outvalue);
125         /*  if(val>max){ //Finding max value
126
127         max = val;
128         a1=val2;}}
129         while (iterator.hasNext()) {
130         //hash.put(value, key1[1]);
131         //hash1.put(value, key1[1]);
132         value=value+iterator.next().get();
```

Then I found the maximum and minimum averages and printed them with their corresponding month values as required.Followed by the difference between them.This took about 1 minute.

```
public static class AverageMapper extends Mapper <LongWritable, Text, Text,Text>
{
    Text outkey = new Text();
    //DoubleWritable outvalue = new DoubleWritable();
    Text t1 = new Text();
    //static HashMap<Double,Integer> last = new HashMap<Double,Integer>();

public void map(LongWritable key, Text value, Context context) throws IOException,InterruptedException
            {
                        //Text outvalue=new Text();
                        String value1=value.toString().replaceAll("\\s+"," ");
                        String[] line = value1.split(" ");
                            t1.set(line[0]);
                            context.write(t1,new Text(line[1]+" "+line[2]));


            }
//private static HashMap<Double,Integer> getoutput(){
    //return last;
//}
}

public static class maxminReducer extends Reducer <Text, Text,Text,Text >
 {
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,InterruptedException

{


        Text outvalue=new Text();
        Text outkey=new Text();
        outkey.set(key);
        //outvalue.set(values);
```

```
        Text outkey=new Text();
        outkey.set(key);
        //outvalue.set(values);

        double min = Integer.MAX_VALUE;
        double max = 0;
        double val=0;
        String a = null;
        String a1 = null;
        //Iterator<DoubleWritable> iterator = values.iterator(); //Iterating
        //hash.put(value, key1[1]);
        //hash1.put(value, key1[1]);
        for(Text value: values) {
            String[] value1=value.toString().split(" ");
            String val1=value1[1];
            String val2=value1[0];
            val=Double.parseDouble(val1);
            if (val < min) { //Finding min value

        min = val;
        a=val2;

        }

            if(val>max){ //Finding max value

        max = val;
        a1=val2;

        }}
        //AvgMonthly obj= new AvgMonthly();
        //String s=obj.hash.get(max);s
        //String s1=obj.hash.get(min);
         String number1=String.valueOf(max);
        String number2=String.valueOf(min);
```

In the final job I sorted the difference to find the state with stable temperature.This took about 30 seconds

```
public static class FinalMapper extends Mapper <LongWritable, Text, DoubleWritable,Text>
{
    Text outkey = new Text();
    Text outvalue = new Text();
    DoubleWritable t1 = new DoubleWritable();


public void map(LongWritable key, Text value, Context context) throws IOException,InterruptedException
        {
                String value1=value.toString().replaceAll("\\s+"," ");
                String[] line = value1.split(" ");
                //AvgMonthly obj= new AvgMonthly();
                //obj.hash.put(Double.parseDouble(line[2]), line[1]);
                //hash1.put(Double.parseDouble(line[2]), line[1]);


                //t1.set(new DoubleWritable(Double.parseDouble(line[5])));
                    outvalue.set(line[0]+" "+line[1]+" "+line[2]);
                    context.write(new DoubleWritable(Double.parseDouble(line[3])),outvalue);
                    //context.write(t1, new DoubleWritable(Double.parseDouble(line[3])));

        }
}

public static class FinalReducer extends Reducer <Text,DoubleWritable,DoubleWritable,Text >
 {
public void reduce(DoubleWritable key, Text values, Context context) throws IOException,InterruptedException

{

    context.write(key, values);

    /*Text outkey = new Text();
    //String[] key1=key.toString().split(" ");
    //outkey.set(key1[0]);
```