1.create java project and save with appropriate project name CSEXXX_TempAnalysis

2.Right click on project and create 3 java class , name it as
a. MinTemp-main class
b. MinTempmapper-mapper class
c. MinTempreducer-reducer class

3.Add Jar files-right click on project ,
a.choose build path option
b.choose configure build path
c.click on add external jars
d.path - Desktop/hadoop-java-jars - select all by pressing ctrl A , click ok

4.Add implementation code in all three java files

5.commands for execution - open new terminal

6.hadoop dfs -mkdir /input - create directory to store all dataset files in HDFS-
  hadoop dfs -ls /input   -displays files inside input directory

7.hadoop dfs -put local file system path /input/ - copies file from local file system to HDFS

8.hadoop jar path-to-jarfile mainclass /path-to-inputfile /output

copy paste path wherever required

9.to check output there are 2 ways :

a.check output from browser --- open firefox browser type in url   localhost:50070- browse filesystem choose output directory and open part-00000 file

b.use cat command

   hadoop dfs -ls /output --displays contents of output directory
   hadoop dfs -cat /output/part-00000 - displays contents of this file

Hint: In main Class change

class name in JobConf() function
class name in setMapperClass() function
class name in setreducerClass() function
main method change in run() with constructor name

MinTemp Main Class:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class MinTemp extends Configured implements Tool {

	public int run(String[] args) throws Exception {
		if(args.length<2)
		{
			System.out.println("Plz Give Input Output Directory Correctly");
			return -1;
		}
		JobConf conf = new JobConf(MinTemp.class);
		FileInputFormat.setInputPaths(conf,new Path(args[0]));
		FileOutputFormat.setOutputPath(conf, new Path(args[1]));
		conf.setMapperClass(MinTempmapper.class);
		conf.setReducerClass(MinTempreducer.class);
		conf.setMapOutputKeyClass(Text.class);
		conf.setMapOutputValueClass(FloatWritable.class);
		conf.setOutputKeyClass(Text.class);
		conf.setOutputValueClass(FloatWritable.class);
		JobClient.runJob(conf);
		return 0;
		}

	public static void main(String[] args) throws Exception {
		int exitcode = ToolRunner.run(new MinTemp(), args);
		System.exit(exitcode);
		}
}
```

MinTempmapper Class:

```
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.Mapper;

public class MinTempmapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text, FloatWritable>
{

        @Override
        public void map(LongWritable key, Text value,
                        OutputCollector<Text, FloatWritable> output, Reporter r)
                                        throws IOException {

                String line = value.toString();
                String[] items = line.split(",");

                String stock = items[3];
                if(!items[1].isEmpty() && !items[1].equals(null))
                {
                        Float closePrice = Float.parseFloat(items[1]);
                   output.collect(new Text(stock), new FloatWritable(closePrice));
                }
}
}
```

MinTempreducer Class:

```
import java.io.IOException;
 import java.util.Iterator;
 import org.apache.hadoop.io.FloatWritable;
```

```
 import org.apache.hadoop.io.Text;
 import org.apache.hadoop.mapred.MapReduceBase;
 import org.apache.hadoop.mapred.OutputCollector;
 import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

 public class MinTempreducer extends MapReduceBase implements
Reducer<Text,FloatWritable,Text,FloatWritable>
 {

        @Override
        public void reduce(Text key, Iterator<FloatWritable> values,
                        OutputCollector<Text, FloatWritable> output, Reporter r)
                        throws IOException {

                float MinTemp=Float.MAX_VALUE;
                 //Iterate all and calculate minimum
                 while (values.hasNext()) {
                        FloatWritable i = values.next();
                        MinTemp = Math.min(MinTemp, i.get());
                }

                //Write output
                output.collect(key, new FloatWritable(MinTemp));
        }
 }
```