Program :

```
package PackageDemo;

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;
import org.apache.hadoop.util.GenericOptionsParser;



public class ProcessUnits {
   public static void main(String[] args) throws Exception {
      Configuration c = new Configuration();
      String[] files = new GenericOptionsParser(c, args).getRemainingArgs();
      Path input = new Path(files[0]);
      Path output = new Path(files[1]);
      Job j = Job.getInstance(c, "maxconsumption");
      j.setJarByClass(ProcessUnits.class);
      j.setMapperClass(MapForMaxConsumption.class);
      j.setReducerClass(ReduceForMaxConsumption.class);
      j.setOutputKeyClass(Text.class);
      j.setOutputValueClass(Text.class);
      FileInputFormat.addInputPath(j, input);
      FileOutputFormat.setOutputPath(j, output);
      System.exit(j.waitForCompletion(true) ? 0 : 1);
 }

   public static class MapForMaxConsumption extends Mapper<LongWritable, Text, Text, Text> {
      public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
         String line = value.toString();
         String[] tokens = line.split(",");
         if (tokens.length != 13) {
            System.err.println("Skipping malformed line: " + line);
            return; // Skip this line
         }
         String year = tokens[0];
         int maxConsumption = Integer.MIN_VALUE;
         String maxMonth = "";
         for (int i = 1; i <= 12; i++) {
            try {
               int consumption = Integer.parseInt(tokens[i]);
               if (consumption > maxConsumption) {
                  maxConsumption = consumption;
```

```
                maxMonth = getMonthName(i);
            }
          } catch (NumberFormatException e) {
            System.err.println("Skipping invalid consumption value: " + tokens[i] + " for year " +
year);
          }
        }
        if (maxMonth.isEmpty()) {
          return; // Skip this entry if no valid max month was found
        }
        con.write(new Text(year), new Text(maxMonth + ":" + maxConsumption));
      }
      private String getMonthName(int monthIndex) {
        String[] months = {
          "January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"
        };
        return months[monthIndex - 1]; // Adjusting for 1-based month index
      }
    }
    public static class ReduceForMaxConsumption extends Reducer<Text, Text, Text, Text> {
        private boolean headerWritten = false; // Flag to check if header is written

          public void reduce(Text year, Iterable<Text> values, Context con) throws IOException,
InterruptedException {
            String maxMonth = "";
            int maxConsumption = Integer.MIN_VALUE;
            for (Text value : values) {
              String[] monthAndConsumption = value.toString().split(":");
              if (monthAndConsumption.length != 2) continue; // Skip if not properly formatted

              String month = monthAndConsumption[0];
              int consumption = Integer.parseInt(monthAndConsumption[1]);
              if (consumption > maxConsumption) {
                maxConsumption = consumption;
                maxMonth = month;
              }
            }
            if (!maxMonth.isEmpty()) {
              if (!headerWritten) {
                // Write the header only once
                con.write(new Text("Year"), new Text("Month"));
                con.write(new Text("Max Consumption"), new Text(""));
                headerWritten = true;
              }
              String outputLine = String.format("Year: %-10s Month: %-15s Max
Consumption: %d", year.toString(), maxMonth, maxConsumption);
              con.write(new Text(""), new Text(outputLine)); // Write formatted output
            }
          }
        }
}
```

Output :

```
[cloudera@quickstart 33316]$ hadoop jar Ass2.jar PackageDemo.ProcessUnits Sample.txt
Assig2Dir1
25/01/21 23:44:05 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
25/01/21 23:44:06 INFO input.FileInputFormat: Total input paths to process : 1
25/01/21 23:44:07 INFO mapreduce.JobSubmitter: number of splits:1
25/01/21 23:44:07 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1737525146805_0003
25/01/21 23:44:07 INFO impl.YarnClientImpl: Submitted application
application_1737525146805_0003
25/01/21 23:44:07 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1737525146805_0003/
25/01/21 23:44:07 INFO mapreduce.Job: Running job: job_1737525146805_0003
25/01/21 23:44:18 INFO mapreduce.Job: Job job_1737525146805_0003 running in uber mode :
false
25/01/21 23:44:18 INFO mapreduce.Job:  map 0% reduce 0%
25/01/21 23:44:26 INFO mapreduce.Job:  map 100% reduce 0%
25/01/21 23:44:36 INFO mapreduce.Job:  map 100% reduce 100%
25/01/21 23:44:37 INFO mapreduce.Job: Job job_1737525146805_0003 completed successfully
25/01/21 23:44:37 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=91
                FILE: Number of bytes written=220899
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=326
                HDFS: Number of bytes written=333
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=6624
                Total time spent by all reduces in occupied slots (ms)=7778
                Total time spent by all map tasks (ms)=6624
                Total time spent by all reduce tasks (ms)=7778
                Total vcore-seconds taken by all map tasks=6624
                Total vcore-seconds taken by all reduce tasks=7778
                Total megabyte-seconds taken by all map tasks=6782976
                Total megabyte-seconds taken by all reduce tasks=7964672
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Map output bytes=75
                Map output materialized bytes=91
                Input split bytes=121
                Combine input records=0
                Combine output records=0
```

```
		Reduce input groups=5
		Reduce shuffle bytes=91
		Reduce input records=5
		Reduce output records=7
		Spilled Records=10
		Shuffled Maps =1
		Failed Shuffles=0
		Merged Map outputs=1
		GC time elapsed (ms)=135
		CPU time spent (ms)=1200
		Physical memory (bytes) snapshot=357376000
		Virtual memory (bytes) snapshot=3007225856
		Total committed heap usage (bytes)=226365440
	Shuffle Errors
		BAD_ID=0
		CONNECTION=0
		IO_ERROR=0
		WRONG_LENGTH=0
		WRONG_MAP=0
		WRONG_REDUCE=0
	File Input Format Counters
		Bytes Read=205
	File Output Format Counters
		Bytes Written=333
[cloudera@quickstart 33316]$ hadoop fs -cat Assig2Dir1/part-r-00000
	Year: 1979	Month: September	Max Consumption: 68
	Year: 1980	Month: August	Max Consumption: 35
	Year: 1981	Month: May	Max Consumption: 39
	Year: 1984	Month: December	Max Consumption: 48
	Year: 1985	Month: June
```