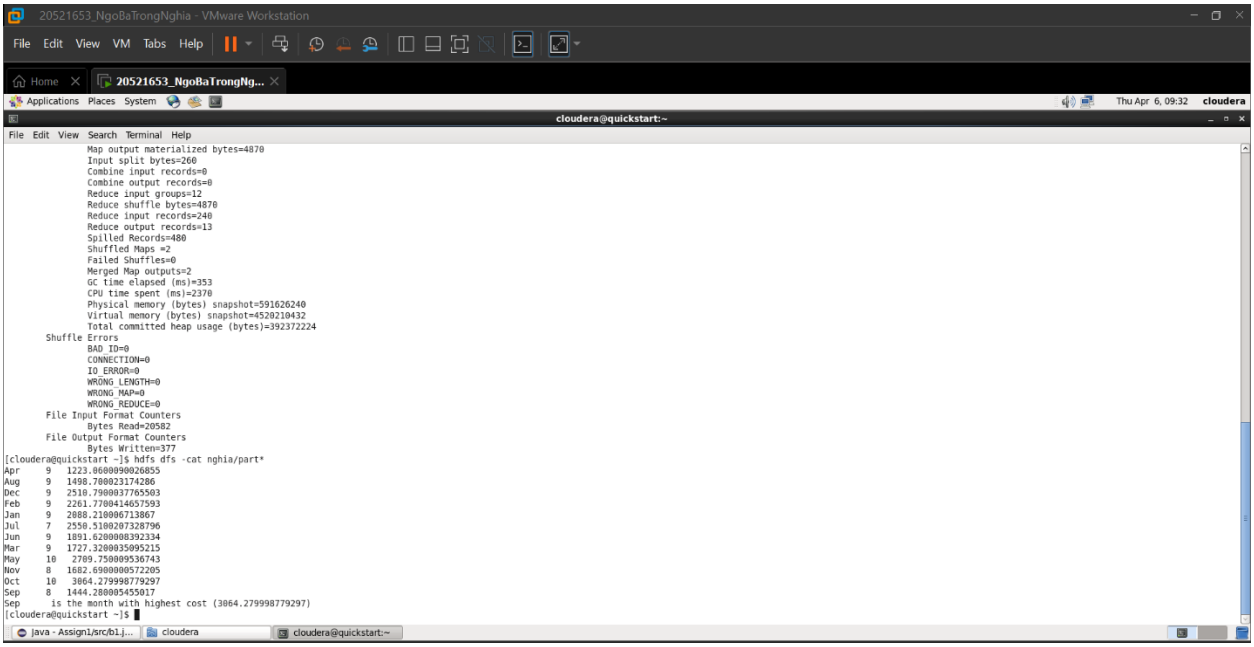
Tên: Ngô Bá Trọng Nghĩa

MSSV: 20521653

Ex1:



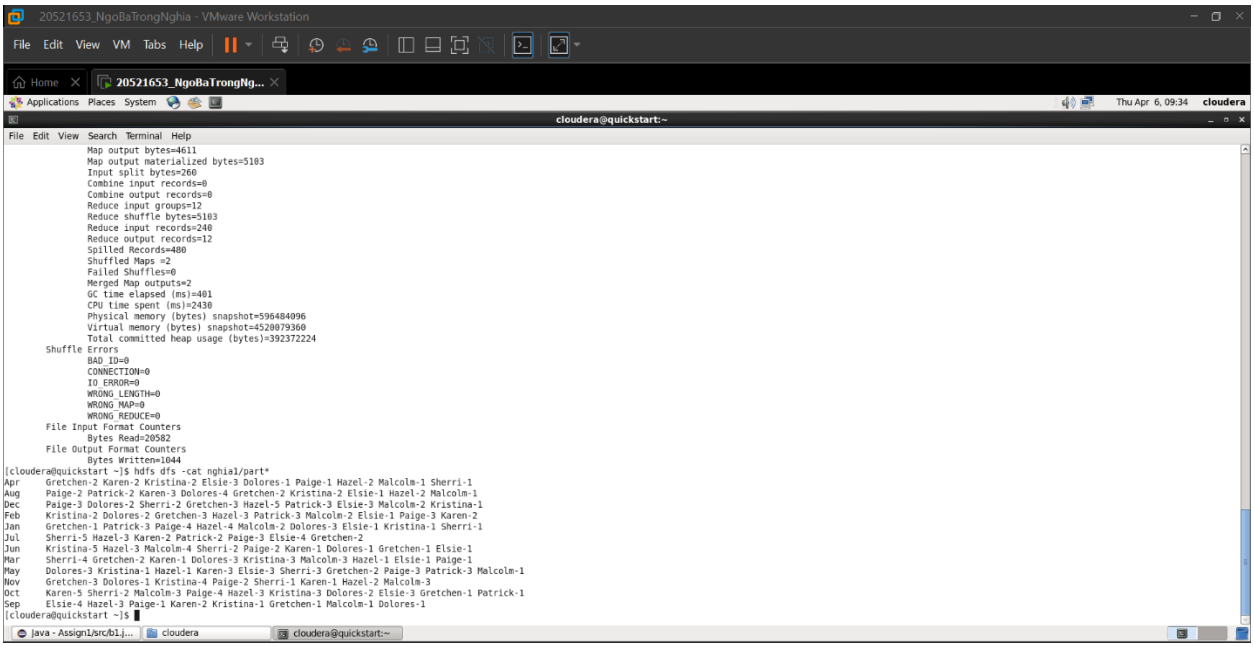
Code java:

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

 public class Ex1{

        //read trans file and show the list of: Game_type Total_amount
```

```
public static class TransMapper extends Mapper <Object, Text, Text, Text>
{
public void map(Object key, Text value, Context context)
throws IOException, InterruptedException
{
String record = value.toString().trim();
String[] parts = record.split(",");

String date = parts[1];
String id = parts[2];
String amount = parts[3];

String month = date.split("-")[0];

context.write(new Text(month), new Text(id + " " + amount));
}
}

public static class TransReducer extends Reducer <Text, Text, Text, Text>
{

	public Double maxCost = 0.0;
	public Text maxMonth;
public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException
{
	List<String> IDs = new ArrayList<String>();
	double total = 0.0;

	for (Text t : values)
	{
		String[] parts = t.toString().trim().split(" ");

		total += Float.parseFloat(parts[1]);
		String id = parts[0];

		if (!IDs.contains(id))
		{
			IDs.add(id);
		}

		if (total > maxCost) {
				maxMonth = key;
```

```
                    maxCost = total;
                }
        }
        context.write(key, new Text(IDs.size() + "   " + Double.toString(total)));
}

@Override
public void cleanup(Context context) throws IOException, InterruptedException {
//write the month with highest cost
context.write(new Text(maxMonth), new Text(" is the month with highest cost (" +
String.valueOf(maxCost) +")") );
}
}

public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "Trans analysis 3");
job.setJarByClass(Ex1.class);
job.setMapperClass(TransMapper.class);
job.setReducerClass(TransReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
//job.setNumReduceTasks(0);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Ex2:



Code java:

```
import java.io.*;
import java.util.*;
import java.net.URI;

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

public class Assignment2 {

        public static class TransMapper extends Mapper<Object, Text, Text, Text> {
                // user map to keep the userId-userName
                private Map<Integer, String> userMap = new HashMap<>();

                public void setup(Context context) throws IOException,
                                InterruptedException {
                        try (BufferedReader br = new BufferedReader(new FileReader(
```

```
                        "cust.txt"))) {
                    String line;
                    while ((line = br.readLine()) != null) {
                        String columns[] = line.split(",");
                        String id = columns[0];
                        String name = columns[1];
                        userMap.put(Integer.parseInt(id), name);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            public void map(Object key, Text value, Context context)
                    throws IOException, InterruptedException {
                String record = value.toString().trim();
                String[] parts = record.split(",");

                String date = parts[1];
                String id = parts[2];
                String name = userMap.get(Integer.parseInt(id));

                String month = date.split("-")[0];

                context.write(new Text(month), new Text(id + "," + name));
            }
        }

        public static class TransReducer extends Reducer<Text, Text, Text, Text> {

            public void reduce(Text key, Iterable<Text> values, Context context)
                    throws IOException, InterruptedException {
                List<String> Names = new ArrayList<String>();
                Map<String, Integer> userCount = new HashMap<String,
Integer>();

                for (Text t : values) {
                    String[] parts = t.toString().trim().split(",");
                    String name = parts[1];

                    if (!Names.contains(name)) {
                        Names.add(name);
                        userCount.put(name, 1);
```

```
                } else {
                        userCount.put(name, userCount.get(name) + 1);
                }

        }

        String str = "";

        for (String name : Names) {
                str += name + "-" + userCount.get(name) + " ";
        }

        context.write(key, new Text(str));
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "q2");
    job.setJarByClass(Assignment2.class);
    job.setMapperClass(TransMapper.class);
    job.setReducerClass(TransReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    // Setting reducer to zero
    // job.setNumReduceTasks(0);

    try {

        job.addCacheFile(new URI(
                        "hdfs://localhost:8020/user/cloudera/cust.txt"));
    } catch (Exception e) {
        System.out.println("File Not Added");
        System.exit(1);
    }

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```