Output:

1. **User Count program**

```
package user_frequency_count;

import java.io.IOException;
import java.util.StringTokenizer;

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



public class user_frequency_count {

        public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString(), ",");
                  while (itr.hasMoreTokens()) {
                        word.set(itr.nextToken());
```

```
                                String validation_string = word.toString();
                                StringTokenizer validation = new StringTokenizer(validation_string,
"."); 
                                int count = 0;
                                while(validation.hasMoreTokens())
                                {
                                        count += 1;
                                        validation.nextToken();
                                }
                                if (count == 4)
                                {
                                        context.write(word, one);
                                }
                                itr.nextToken();
                                itr.nextToken();
                                itr.nextToken();
                        }
                }
        }

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

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
```

```
    result.set(sum);
    context.write(key, result);
  }
 }

 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(user_frequency_count.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }

}
```

2. Extract maximum frequency user program

```
package maximum_user_frequency;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
```

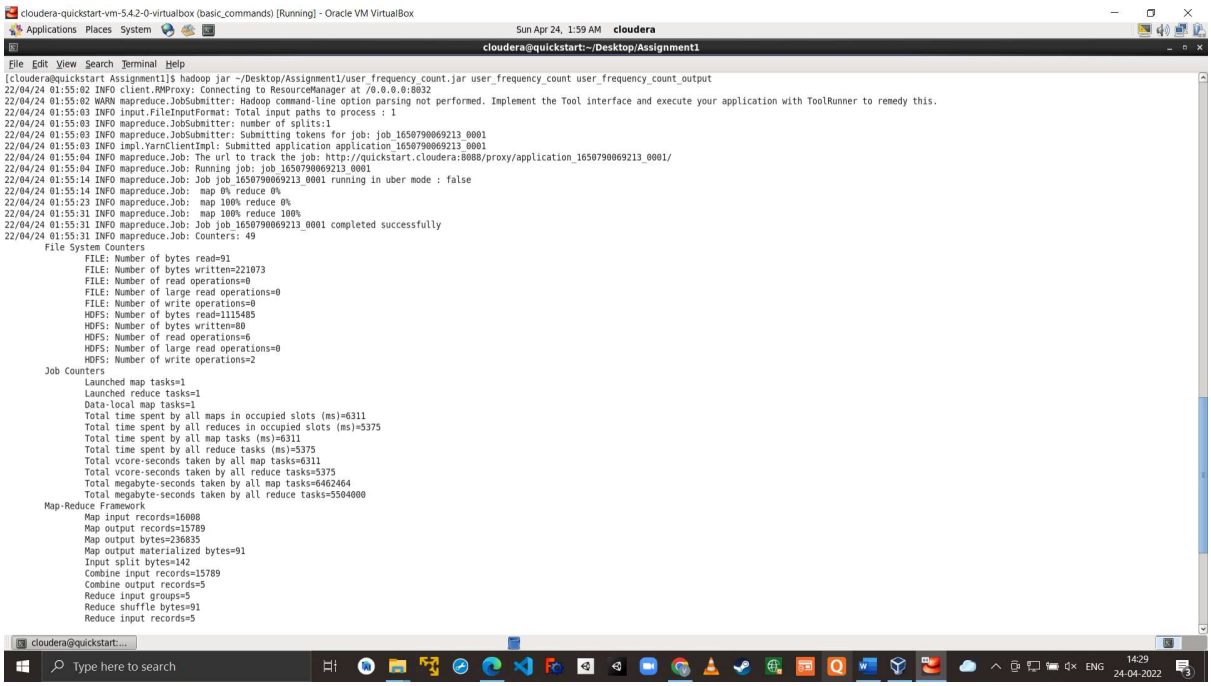
```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaximumUserFrequency{

	static class User implements Writable, WritableComparable<User> {
		private String name = "";
		private Integer freq = new Integer(0);

		public void setName(String n)
		{
			name = n;
		}

		public void setFreq(Integer f)
		{
			freq = f;
		}

		public String getName()
		{
			return name;
		}

		public Integer getFreq()
```

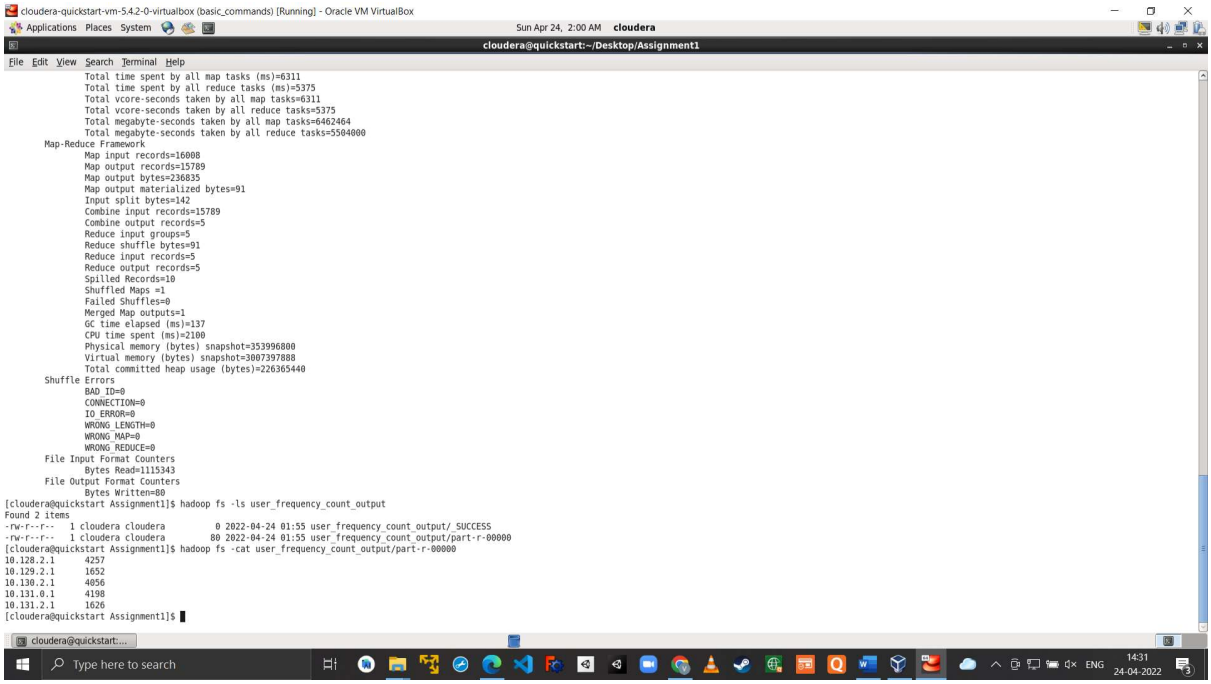
```
	{
		return freq;
	}

	@Override
	public void readFields(DataInput in) throws IOException {
		freq = in.readInt();
		name = in.readLine();
	}

	@Override
	public void write(DataOutput out) throws IOException {
		out.writeInt(freq);
		out.writeBytes(name);
	}

	@Override
	public int compareTo(User o) {
		// TODO Auto-generated method stub
		int result = this.freq.compareTo(o.freq);
		return result;
	}
}

private static User u = new User();

static class UserMaxCountMapper extends Mapper<Object, Text, Text, User> {
	private Text user = new Text();

	@Override
```

```
        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
                user.set("user");
                u.setName(value.toString().split(",")[0]);
                u.setFreq(Integer.parseInt(value.toString().split(",")[1]));
                context.write(user, u);
        }
    }

    static class UserMaxCountReducer extends Reducer<Text, User, Text, IntWritable> {
        private User result = new User();

        public void reduce(Text key, Iterable<User> values, Context context)
                throws IOException, InterruptedException {
                result.setFreq(null);
                result.setName("");

                for (User value : values)
                {
                        if (result.getFreq() == null || (value.getFreq() > result.getFreq())) {
                                result.setFreq(value.getFreq());
                                result.setName(value.getName());
                        }
              }

                key.set(result.getName());
            context.write(key, new IntWritable(result.getFreq()));
        }
    }
```

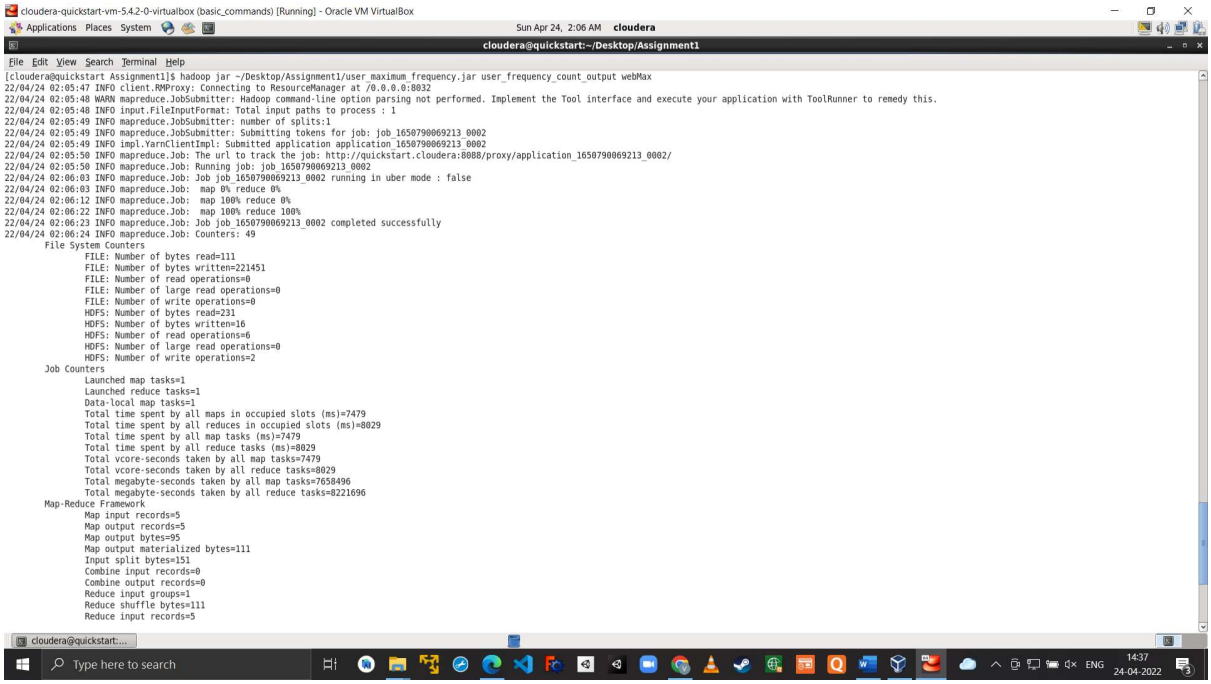
```
        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                Job job = Job.getInstance(conf);
                job.setJarByClass(MaximumUserFrequency.class);
                job.setJobName("find_max_user_count");
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                job.setMapperClass(UserMaxCountMapper.class);
                job.setReducerClass(UserMaxCountReducer.class);
                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(User.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

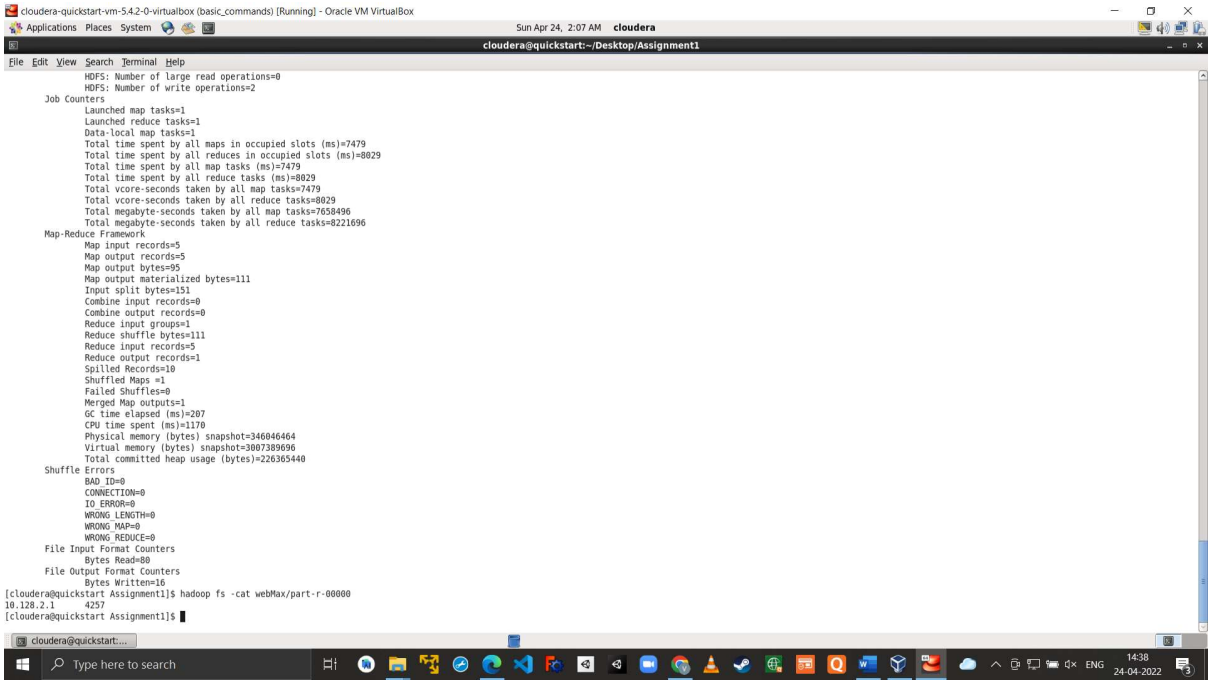
3. User Count program output





4. Maximum User frequency program output




cloudera-quickstart-vm-5.4.2-0-virtualbox (basic_commands) [Running] - Oracle VM VirtualBox
Applications Places System
Sun Apr 24, 2:07 AM cloudera
cloudera@quickstart:~/Desktop/Assignment1
File Edit View Search Terminal Help
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)=7479
Total time spent by all reduces in occupied slots (ms)=8029
Total time spent by all map tasks (ms)=7479
Total time spent by all reduce tasks (ms)=8029
Total vcore-seconds taken by all map tasks=7479
Total vcore-seconds taken by all reduce tasks=8029
Total megabyte-seconds taken by all map tasks=7658496
Total megabyte-seconds taken by all reduce tasks=8221696
Map-Reduce Framework
Map input records=5
Map output records=5
Map output bytes=95
Map output materialized bytes=111
Input split bytes=151
Combine input records=0
Combine output records=0
Reduce input groups=1
Reduce shuffle bytes=111
Reduce input records=5
Reduce output records=1
Spilled Records=10
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=207
CPU time spent (ms)=1170
Physical memory (bytes) snapshot=346046464
Virtual memory (bytes) snapshot=3007389696
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=80
File Output Format Counters
Bytes Written=16
[cloudera@quickstart Assignment1]$ hadoop fs -cat webMax/part-r-00000
10.128.2.1 4257
[cloudera@quickstart Assignment1]$
cloudera@quickstart:...
Type here to search
ENG
14:38
24-04-2022