Part – 1: To find the air craft model with highest number of survivors:

```
package basics.bda;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
public class testing {
        public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>
        {
        //private final static IntWritable one = new IntWritable(1);
        IntWritable one = new IntWritable(1);

        public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
            String[] line = value.toString().split(",");
            IntWritable n = new IntWritable(Integer.parseInt(line[11]));
                context.write(new Text(line[4]),n);
            }
                }
         public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
                int max_sum=0;
                Text max_key=new Text("The  aircraft with highest number of surviours: ");
                Text max_occured_key = new Text();
                 public void reduce(Text key, Iterable<IntWritable> values, Context context)
                 throws IOException, InterruptedException {
                        int sum = 0;
                   for (IntWritable val : values) {
                      sum += val.get();
                   }
                   if(sum > max_sum) {
                        max_sum = sum;
                        max_occured_key.set(key);
                        }
                context.write(key, new IntWritable(sum));

                 }
                  @Override
                 protected void cleanup(Context context) throws IOException, InterruptedException {
                        context.write(max_key, new IntWritable(max_sum));
                 context.write(max_occured_key, new IntWritable(max_sum));
}
                 }
         public static void main(String[] args) throws Exception {
                  Configuration conf = new Configuration();
               Job job = new Job(conf, "wordcount");
               job.setJarByClass(testing.class);
```

```
                job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
                job.setMapperClass(Map.class);
                job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
                job.waitForCompletion(true);
                }
}
```

Part – 2: To find the aircraft with more system failures:

```
package bda.project;

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

import org.apache.hadoop.mapreduce.Mapper.Context;

import org.apache.hadoop.mapreduce.lib.input.*;

import org.apache.hadoop.mapreduce.lib.output.*;

public class takeone {

        public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>

        {

        //private final static IntWritable one = new IntWritable(1);

        IntWritable one = new IntWritable(1);


        public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

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

                String sur=(line[12]).toString();

                if(sur.equals("systems failure"))
```

```
        {
                context.write(new Text(line[4]),one);
        }

    }
  }

public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

          int max_s=0,max=0;
          Text max_key=new Text("The aircraft model in with more system failure : ");
          Text m_key=new Text();
           public void reduce(Text key, Iterable<IntWritable> values, Context context)
           throws IOException, InterruptedException {
                int s=0,c=0,avg=0;
              for (IntWritable val : values) {
                 s+= val.get();
              }
              if(s>max)
              {
                   max=s;
                   m_key.set(key);
              }
          context.write(key, new IntWritable(s));

           }
            @Override
            public void cleanup(Context context) throws IOException, InterruptedException
            {
            context.write(max_key, new IntWritable(max));
            context.write(m_key, new IntWritable(max));
```

```
    }
            }
    public static void main(String[] args) throws Exception
    {
            Configuration conf = new Configuration();
            Job job = new Job(conf, "Aviation Crash Analysis");
            job.setJarByClass(takeone.class);
            job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
            job.setMapperClass(Map.class);
            job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.waitForCompletion(true);
            }
}
```