Analyze impact of different mapper and reducer functions
Write a MapReduce program in Java to find the list of document files in which a book title is mentioned.
Build an inverted index of the form:
BookTitle -> file name
This index helps quickly locate which library logs or sales documents mention a specific book.
Three sample data files
book_sales_1.csv
book_sales_2.csv
book_sales_3.csv
● Build an inverted index mapping each Book Title to the set of files where it appears.
● Ensure no duplicate file names are listed for a book.
Sample Output:
Harry Potter book_sales_1.csv book_sales_2.csv
Inferno book_sales_1.csv book_sales_2.csv
The Hobbit book_sales_1.csv book_sales_3.csv
To Kill a Mockingbird book_sales_2.csv
The Alchemist book_sales_3.csv

```
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

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

public class InvertedIndex {

   // ---------- Mapper ----------
   public static class InvertedIndexMapper extends Mapper<Object, Text, Text, Text>
{
      private Text bookTitle = new Text();
      private Text fileName = new Text();

      @Override
      protected void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {

         // Get the file name from the input split
         String fileNameStr = ((FileSplit) context.getInputSplit()).getPath().getName();
         fileName.set(fileNameStr);
```

```
        // Split the line to get book title
        String[] parts = value.toString().split(",", 2);
        if (parts.length > 0) {
            bookTitle.set(parts[0].trim());
            context.write(bookTitle, fileName);
        }
    }
}

// ---------- Reducer ----------
public static class InvertedIndexReducer extends Reducer<Text, Text, Text, Text> {
    private Text result = new Text();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {

        Set<String> fileSet = new HashSet<>();
        for (Text val : values) {
            fileSet.add(val.toString());
        }

        // Convert Set to space-separated String
        StringBuilder fileList = new StringBuilder();
        for (String file : fileSet) {
            if (fileList.length() > 0) fileList.append(" ");
            fileList.append(file);
        }

        result.set(fileList.toString());
        context.write(key, result);
    }
}

// ---------- Driver ----------
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Inverted Index");

    job.setJarByClass(InvertedIndex.class);
    job.setMapperClass(InvertedIndexMapper.class);
    job.setReducerClass(InvertedIndexReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
```

```
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```