CSE488: Big Data Analytics
[SPRING 2023]

<u>Lab-6</u>
MapReduce program to find the occurrences of each page, most visited page, least visited page, and frequency of pairs.

Submitted by:

Student ID: 2019-3-60-046
Student Name: Mohsenul Kabir Mithun

Creating a Hadoop project in Eclipse:

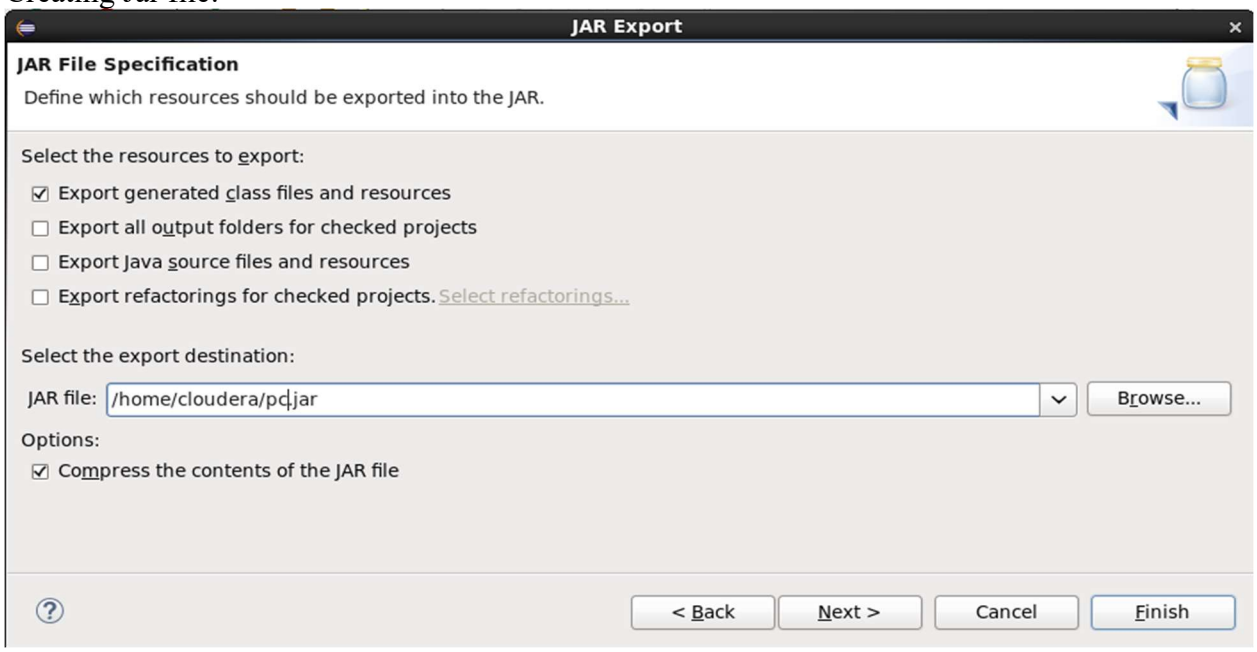
1. Find the occurrences of each page (numbered as 1, 2, 3, and so on).

Source Code SS: (Mapper and Reducer Class)

PageCount.java | MostVisited.java | LeastVisited.java | PairFrequency.java

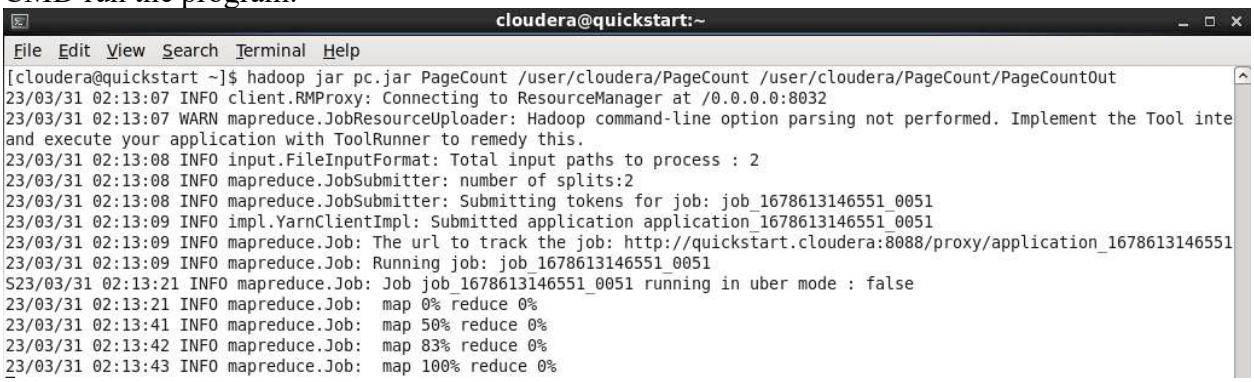
```java
public class PageCount {

    public static class PageCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable ONE = new IntWritable(1);
        private Text page = new Text();

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String[] line = value.toString().split("\\s+");
            for (String pageString : line) {
                page.set(pageString);
                context.write(page, ONE);
            }
        }
    }

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

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

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job1 = Job.getInstance(conf, "Page Count");
        job1.setJarByClass(PageCount.class);
        job1.setMapperClass(PageCountMapper.class);
        job1.setReducerClass(PageCountReducer.class);
        job1.setOutputKeyClass(Text.class);
        job1.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job1, new Path(args[0]));
        FileOutputFormat.setOutputPath(job1, new Path(args[1]));
        System.exit(job1.waitForCompletion(true) ? 0 : 1);
    }
}
```

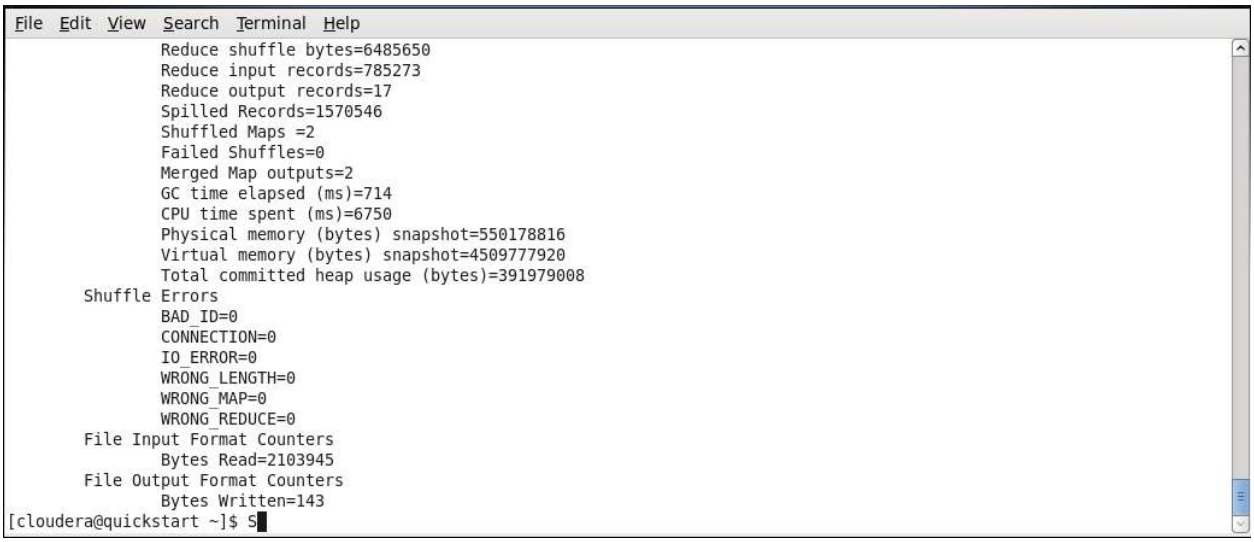
Creating Jar file:



Creating Input file:

CMD run the program:





Output:

Home Page 1 of 1

/ user / cloudera / PageCount / PageCountOut / **part-r-00000**

```
1       157807
10      22559
11      16380
12      45173
13      35768
14      66621
15      10999
16      3122
17      2832
2       75962
3       33788
```

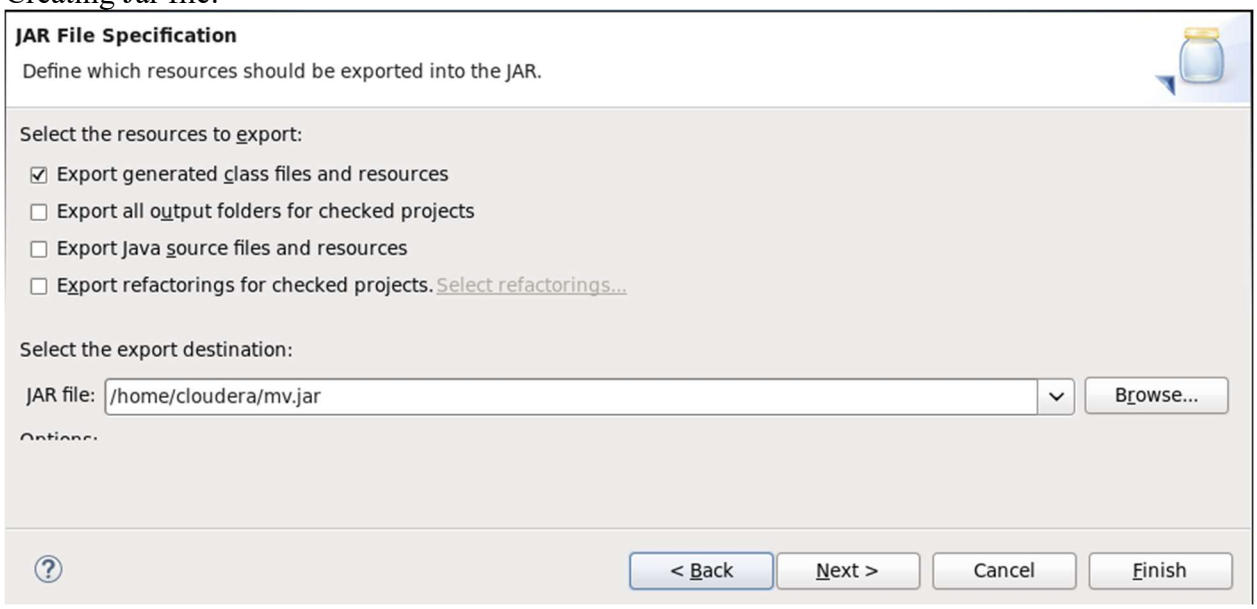
2. Find the most visited page. You may use the output file generated from Question 1 as an input file to solve this problem.

Source Code SS: (Mapper and Reducer Class)

PageCount.java | MostVisited.java | LeastVisited.java | PairFrequency.java

```java
public class MostVisited {
    public static class MostVisitedMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String[] line = value.toString().split("\\s+");
            int count = Integer.parseInt(line[1]);
            context.write(new Text(line[0]), new IntWritable(count));
        }
    }
    public static class MostVisitedReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable(Integer.MIN_VALUE);
        private Text MostVisitedPage = new Text();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
            int maxCount = Integer.MIN_VALUE;
            for (IntWritable value : values) {
                int count = value.get();
                if (count > maxCount) {
                    maxCount = count;
                }
            }
            if (maxCount > result.get()) {
                MostVisitedPage.set(key);
                result.set(maxCount);
            }
        }
        @Override
        protected void cleanup(Context context) throws IOException, InterruptedException {
            context.write(MostVisitedPage, result);
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job1 = Job.getInstance(conf, "Least Visited Page");
        job1.setJarByClass(MostVisited.class);
        job1.setMapperClass(MostVisitedMapper.class);
        job1.setReducerClass(MostVisitedReducer.class);
        job1.setOutputKeyClass(Text.class);
        job1.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job1, new Path(args[0]));
        FileOutputFormat.setOutputPath(job1, new Path(args[1]));
        job1.waitForCompletion(true);
    }
}
```

Creating Jar file:



Creating Input file:

CMD run the program:

```
[cloudera@quickstart ~]$ hadoop jar mv.jar MostVisited /user/cloudera/PageCount/PageCountOut /user/cloudera/PageCount/M
ostVisitedOut
23/03/30 21:09:20 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/03/30 21:09:20 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the T
ool interface and execute your application with ToolRunner to remedy this.
23/03/30 21:09:21 INFO input.FileInputFormat: Total input paths to process : 1
23/03/30 21:09:21 INFO mapreduce.JobSubmitter: number of splits:1
23/03/30 21:09:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1678613146551_0008
23/03/30 21:09:21 INFO impl.YarnClientImpl: Submitted application application_1678613146551_0008
23/03/30 21:09:21 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_16786
13146551_0008/
23/03/30 21:09:21 INFO mapreduce.Job: Running job: job_1678613146551_0008
23/03/30 21:09:32 INFO mapreduce.Job: Job job_1678613146551_0008 running in uber mode : false
23/03/30 21:09:32 INFO mapreduce.Job:  map 0% reduce 0%
```

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
                Reduce shuffle bytes=150
                Reduce input records=17
                Reduce output records=1
                Spilled Records=34
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=194
                CPU time spent (ms)=1410
                Physical memory (bytes) snapshot=366968832
                Virtual memory (bytes) snapshot=3008630784
                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=143
        File Output Format Counters
                Bytes Written=9
[cloudera@quickstart ~]$
```

Output:

Home Page 1 of 1

/ user / cloudera / PageCount / MostVisitedOut / **part-r-00000**

```
1       157807
```

3. Find the least visited page. You may use the output file generated from Question 1 as an input file to solve this problem.

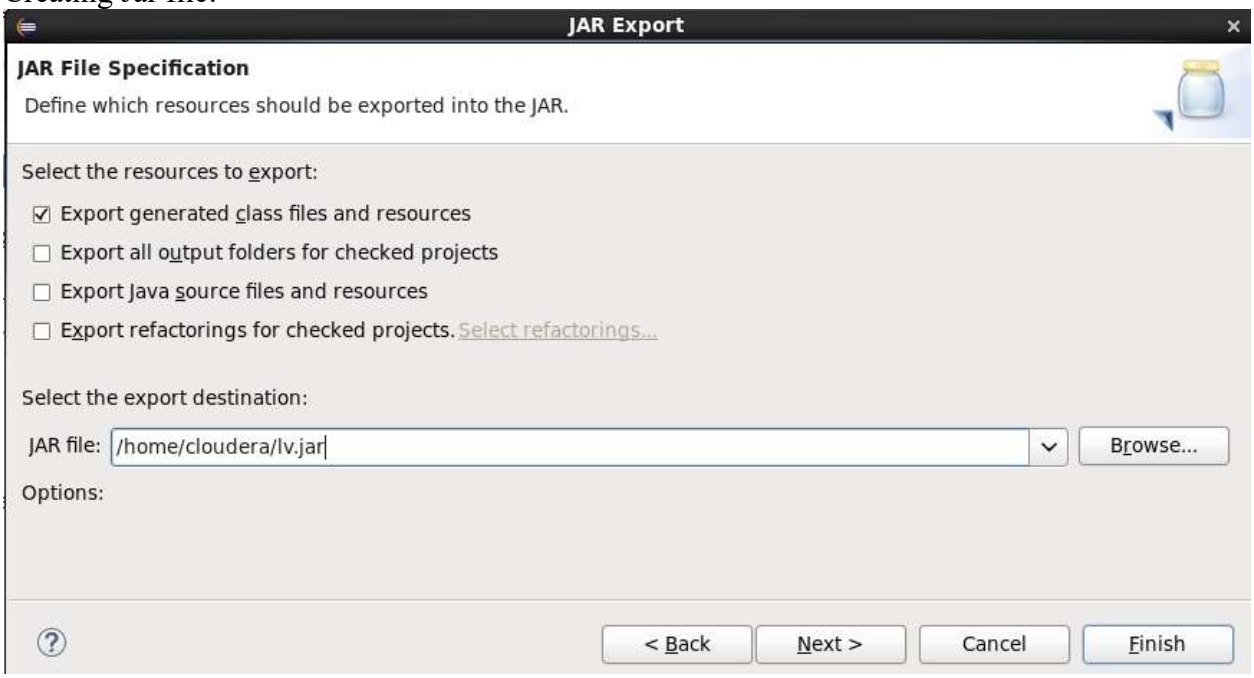
Source Code SS: (Mapper and Reducer Class)

PageCount.java | MostVisited.java | LeastVisited.java | PairFrequency.java

```java
public class LeastVisited {
    public static class LeastVisitedMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String[] line = value.toString().split("\\s+");
            int count = Integer.parseInt(line[1]);
            context.write(new Text(line[0]), new IntWritable(count));
        }
    }
    public static class LeastVisitedReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable(Integer.MAX_VALUE);
        private Text leastVisitedPage = new Text();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
            int minCount = Integer.MAX_VALUE;
            for (IntWritable value : values) {
                int count = value.get();
                if (count < minCount) {
                    minCount = count;
                }
            }
            if (minCount < result.get()) {
                leastVisitedPage.set(key);
                result.set(minCount);
            }
        }
        @Override
        protected void cleanup(Context context) throws IOException, InterruptedException {
            context.write(leastVisitedPage, result);
        }
    }
```

```java
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job1 = Job.getInstance(conf, "Least Visited Page");
        job1.setJarByClass(LeastVisited.class);
        job1.setMapperClass(LeastVisitedMapper.class);
        job1.setReducerClass(LeastVisitedReducer.class);
        job1.setOutputKeyClass(Text.class);
        job1.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job1, new Path(args[0]));
        FileOutputFormat.setOutputPath(job1, new Path(args[1]));
        job1.waitForCompletion(true);
    }
}
```

Creating Jar file:



Creating Input file:

CMD run the program:

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hadoop jar lv.jar LeastVisited /user/cloudera/PageCount/PageCountOut /user/cloudera/PageCount/LeatVisi
tedOut
23/03/31 02:30:05 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/03/31 02:30:06 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inte
rface and execute your application with ToolRunner to remedy this.
23/03/31 02:30:07 INFO input.FileInputFormat: Total input paths to process : 1
23/03/31 02:30:07 INFO mapreduce.JobSubmitter: number of splits:1
23/03/31 02:30:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1678613146551_0054
23/03/31 02:30:08 INFO impl.YarnClientImpl: Submitted application application_1678613146551_0054
23/03/31 02:30:08 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1678613146551
_0054/
23/03/31 02:30:08 INFO mapreduce.Job: Running job: job_1678613146551_0054
23/03/31 02:30:18 INFO mapreduce.Job: Job job_1678613146551_0054 running in uber mode : false
```

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
                Reduce shuffle bytes=150
                Reduce input records=17
                Reduce output records=1
                Spilled Records=34
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=266
                CPU time spent (ms)=1570
                Physical memory (bytes) snapshot=364797952
                Virtual memory (bytes) snapshot=3008925696
                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=143
        File Output Format Counters
                Bytes Written=8
[cloudera@quickstart ~]$
```

Output:

Home Page 1 of 1

/ user / cloudera / PageCount / LeatVisitedOut / **part-r-00000**

17	2832

4. Find the frequency of pairs. In this case, remove the duplicate numbers from each transaction as a data pre-processing step. Use the concept of Java set for removing duplicates.

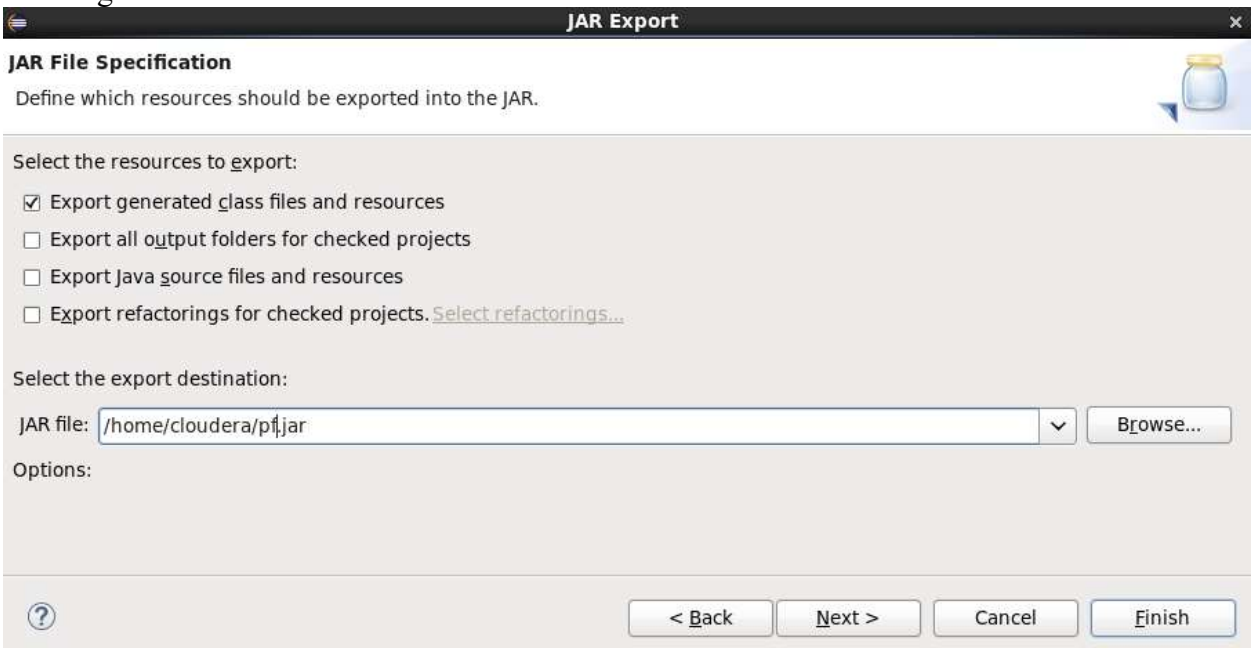
Source Code SS: (Mapper and Reducer Class)

PageCount.java | MostVisited.java | LeastVisited.java | *PairFrequency.java

```java
public class PairFrequency {
  public static class SumMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
      String[] fields = value.toString().split(" ");
      Set<String> uniqueFields = new HashSet<>(Arrays.asList(fields));
      String[] uniqueFieldsArray = uniqueFields.toArray(new String[uniqueFields.size()]);
      Arrays.sort(uniqueFieldsArray);
      int j = 0;
      String f = "";
      for (int i = 0; i < uniqueFieldsArray.length; i++) {
        for (j = i + 1; j < uniqueFieldsArray.length; j++) {
          if (uniqueFieldsArray[i].equals(uniqueFieldsArray[j]) || f.equals(uniqueFieldsArray[j])) {
            continue;
          }
          f = uniqueFieldsArray[j];
          String a = uniqueFieldsArray[i].toString().trim() + " " + uniqueFieldsArray[j].toString().trim();
          context.write(new Text(a), new IntWritable(1));
        }
      }
    }
  }
  public static class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      context.write(key, new IntWritable(sum));
    }
```

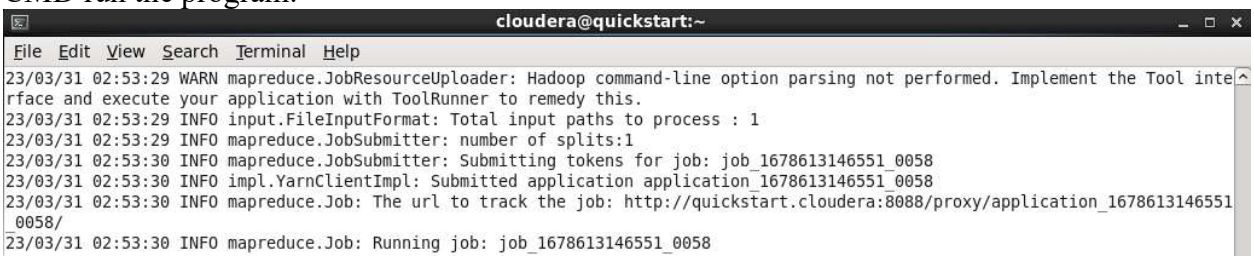
```java
  public static void main(String[] args) throws Exception {
    Job job = Job.getInstance();
    job.setJarByClass(PairFrequency.class);
    job.setJobName("Pair Frequency Count");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(SumMapper.class);
    job.setReducerClass(SumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Creating Jar file:



Creating Input file:

CMD run the program:



Output:


Home
Page 1 of 1
/ user / cloudera / PageCount / PairFrequencyOut / part-r-00000
1 10 4287
1 11 5494
1 12 7138
1 13 661
1 14 6526
1 15 1512
1 16 191
1 17 1262
1 2 12477
1 3 5497
1 4 6578