**Big Data Frameworks CSE3120**

Lab – 5 Map Reduce Join Experiment

**Name:** Naveen Nidadavolu

**Roll No:** 22MIA1049

**Map Reduce Join**

**Aim:** To perform a join operation using MapReduce on two datasets—cus.txt and trs.txt—to combine bo with their respective marks based on ID.

**Procedure**

1. Create input text files and upload them to HDFS.
2. Run the MapReduce join program to process the input files.
3. The Mapper reads and tags records from both files based on ID.
4. The Reducer merges records with the same ID to produce the final joined output.
5. The output is stored in HDFS.
6. Retrieve and display the results.

**Program**

import java.io.\*; import java.net.URI;

import java.util.HashMap; import java.util.Map;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.filecache.DistributedCache; import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

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

public class msp2 {

public static class JoinMapper extends Mapper<LongWritable, Text, Text, Text> { private Map<String, String> userMap = new HashMap<>();

private Text outputKey = new Text(); private Text outputValue = new Text();

@Override

protected void setup(Context context) throws IOException, InterruptedException {

// Read the distributed cache file

Configuration conf = context.getConfiguration();

Path[] cacheFiles = DistributedCache.getLocalCacheFiles(conf); if (cacheFiles != null && cacheFiles.length > 0) {

BufferedReader reader = new BufferedReader(new FileReader(cacheFiles[0].toString())); String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split(",");

userMap.put(parts[0], parts[1]); // UserID -> UserName

}

reader.close();

System.err.println("deb"+ userMap.size() + "users.");

} else {

throw new IOException("dis cah is missing");}

}

@Override

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

String[] transaction = value.toString().split(","); String userId = transaction[0];

String amount = transaction[1];

if (userMap.containsKey(userId)) { outputKey.set(userMap.get(userId)); // UserName

outputValue.set(amount); context.write(outputKey, outputValue);

}

}

}

public static void main(String[] args) throws Exception { if(args.length<3){

System.err.println("err"); System.exit(1);

}

Configuration conf = new Configuration(); DistributedCache.addCacheFile(new URI(args[2]),conf); Job job = new Job(conf);

job.setJarByClass(msp2.class); job.setMapperClass(JoinMapper.class);

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

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

// Add users.txt to the Distributed Cache job.setNumReduceTasks(0); // No reducer required for mapside join System.exit(job.waitForCompletion(true) ? 0 : 1);

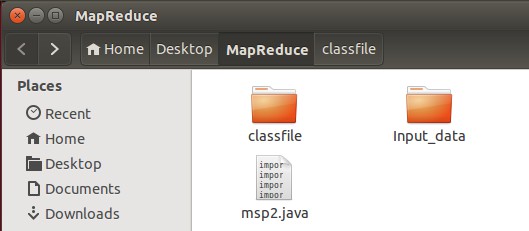
}

}

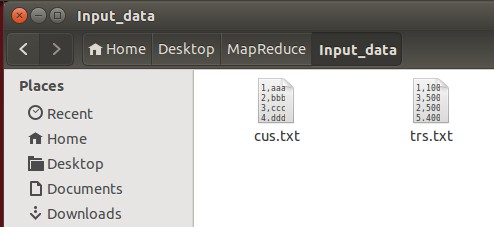
**Input files**

* File 1 (cus.txt): ID Name
* File 2 (trs.txt): ID Price

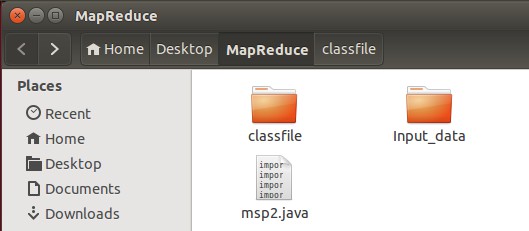
1. Download the text and program files, and place in a **separate folder in desktop**
2. Open the terminal and run basic commands to start the Hadoop services and check its version.

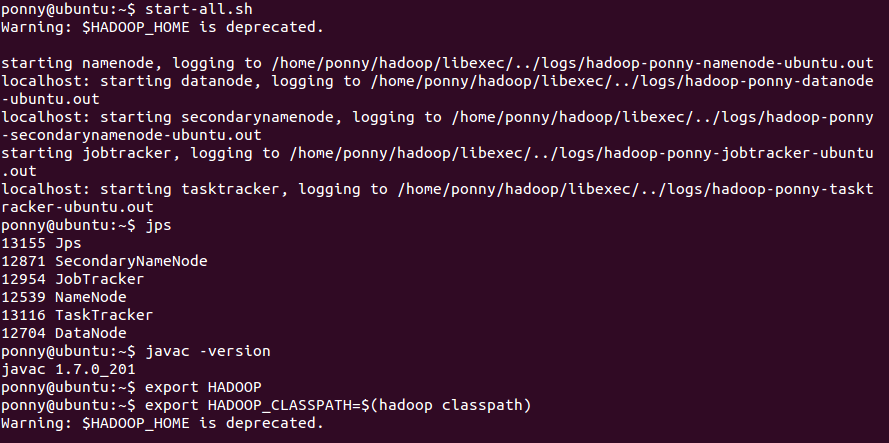


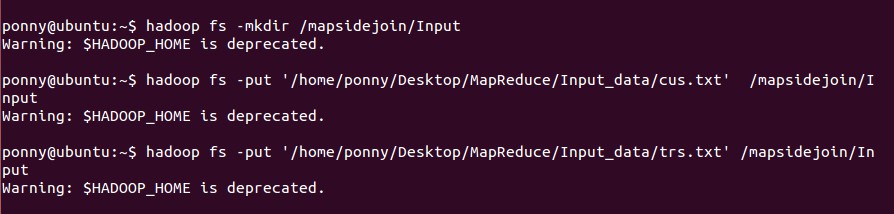
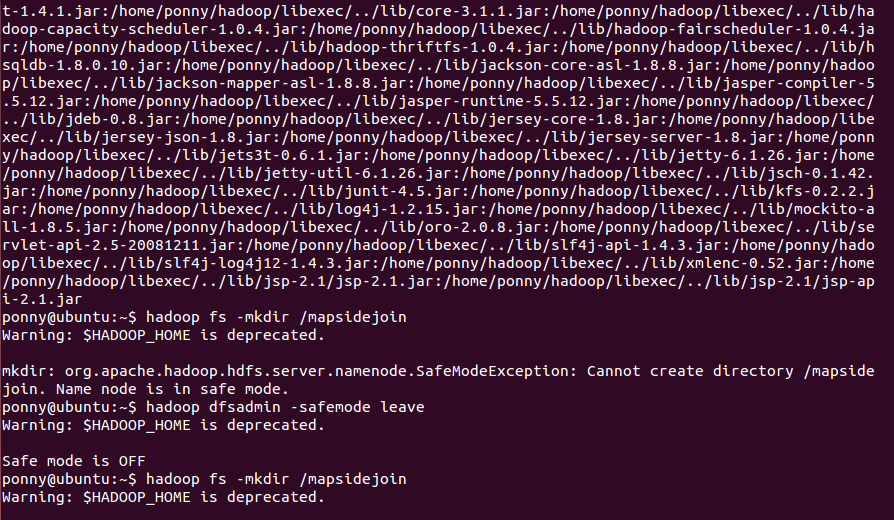
1. In the desktop, create a folder, For ex: here the folder name is given as **MapReduce**. Inside this folder create an **Input\_data** folder and place the input text files (**cus.txt and trs.txt**).

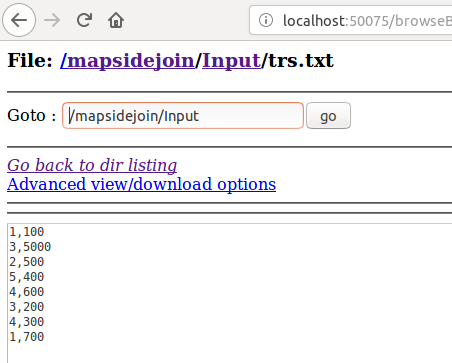
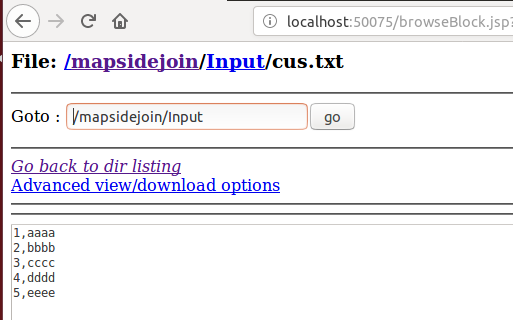
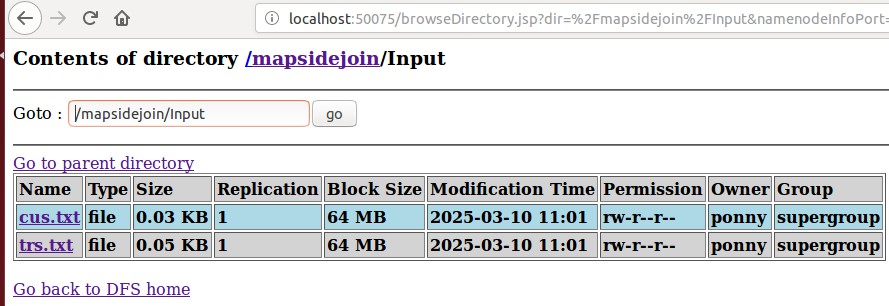
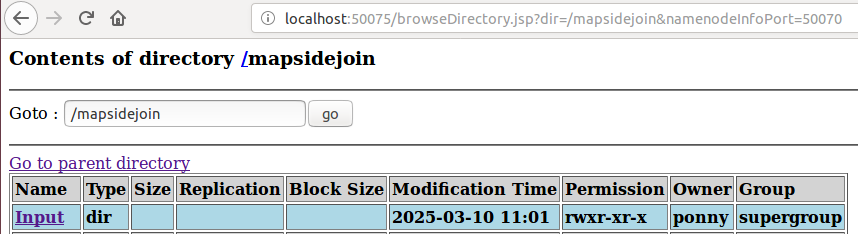


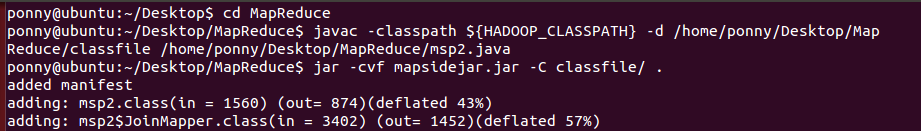
1. Inside the MapReduce folder create an empty folder name classfile.

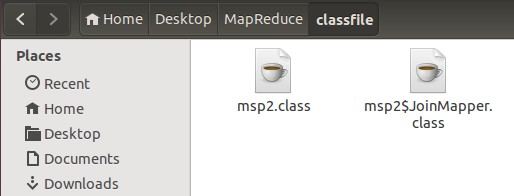


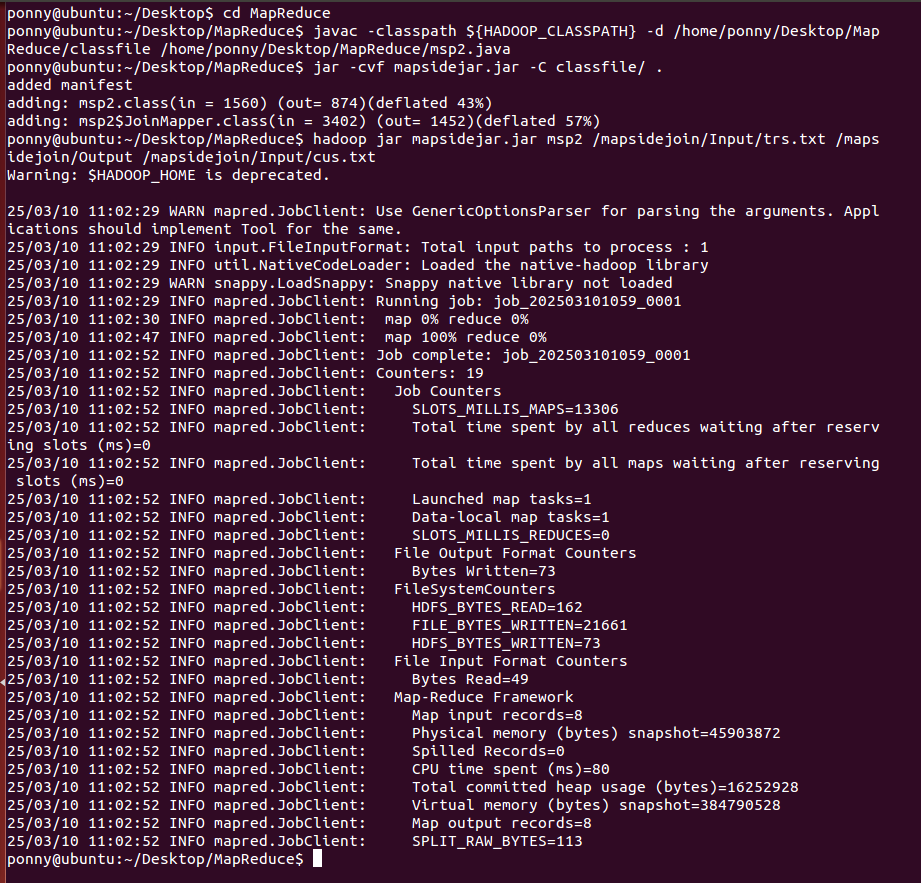
1. Refer the screenshots below

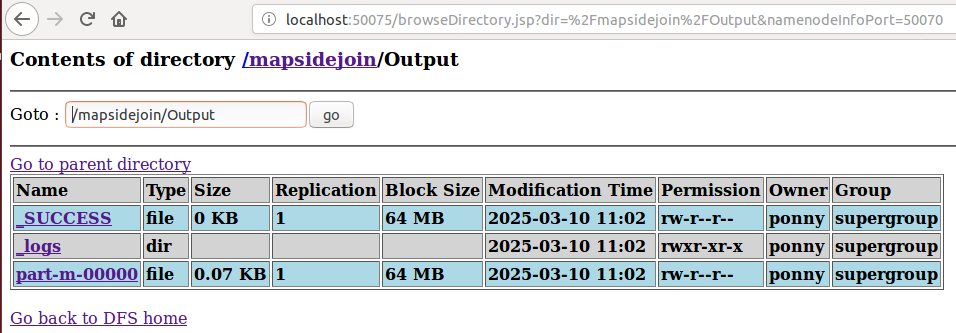
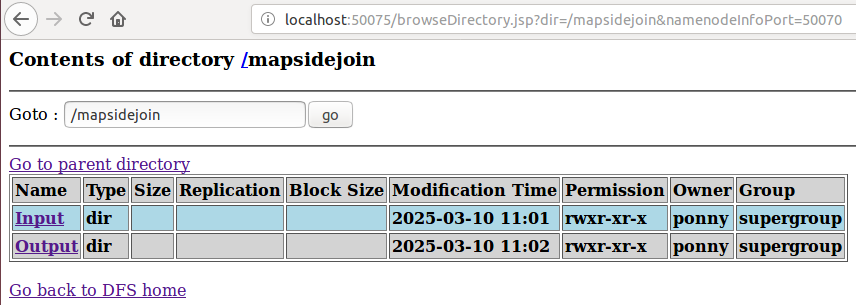




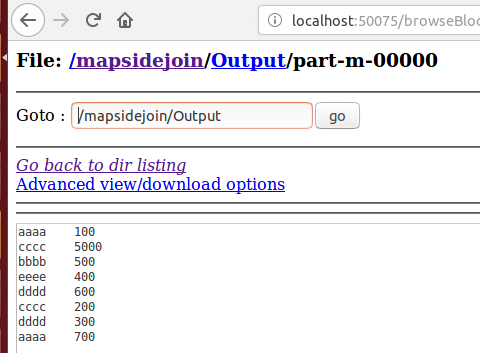








**Output:**



**Result:** Successfully performed a join operation using MapReduce, merging customer details with their transactions based on ID.