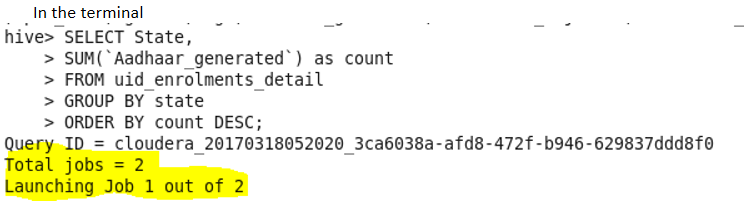
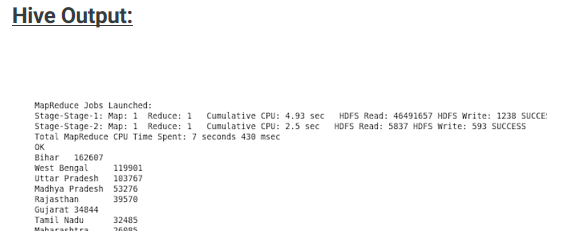
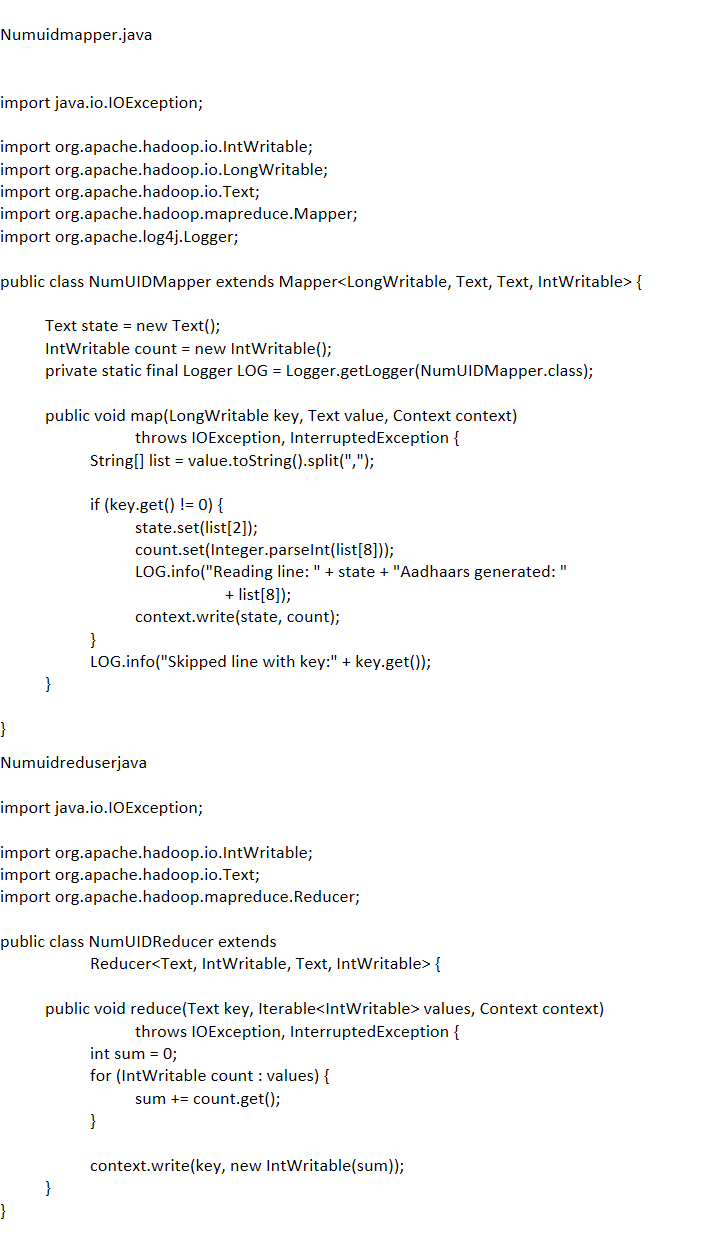
Creating the hive table

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | CREATE TABLE  registrar string,  enrolment_agency string,  state string,  district string,  sub_district string,  pin_code bigint,  gender strings  age int,  aadhaar_generated int,  enrolment_rejected int,  int,  int)  ROW FORMAT DELIMITED  FIELDS TERMINATED BY  TBLPROPERTIES ( •skip. header. |  |  |
|  |  | C:\Users\akash\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image002.png |
|  |  |  |





Job1



Job2

Sortmapper

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class sortMapper extends Mapper<LongWritable, Text, IntWritable, Text> {

Text state = new Text();

IntWritable count = new IntWritable();

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] splits = value.toString().split("\\|");

// Swap key and value i.e. set count as key and state as value

state.set(splits[0].trim());

count.set(Integer.parseInt(splits[1].trim()));

context.write(count, state);

}

}

Sort comparator

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

public class sortComparator extends WritableComparator {

@Override

public int compare(WritableComparable k1, WritableComparable k2) {

IntWritable v1 = (IntWritable) k1;

IntWritable v2 = (IntWritable) k2;

return v1.get() < v2.get() ? 1 : v1.get() == v2.get() ? 0 : -1;

}

protected sortComparator() {

super(IntWritable.class, true);

}

}

Sort reducer

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class sortReducer extends Reducer<IntWritable, Text, Text, IntWritable> {

public void reduce(IntWritable key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

for (Text val : values) {

context.write(val, key);

}

}

}

DRIVER

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.FileSystem;

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.lib.input.FileInputFormat;

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

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class Driver extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception {

if (args.length != 3) {

System.out.println("Usage: [input] [output1] [output2]");

System.exit(-1);

}

Job stateWiseCount = Job.getInstance(getConf());

stateWiseCount.setJobName("Aadhaar Data Analysis");

stateWiseCount.setJarByClass(Driver.class);

/\* Field separator for reducer output\*/

stateWiseCount.getConfiguration().set("mapreduce.output.textoutputformat.separator", " | ");

stateWiseCount.setMapperClass(NumUIDMapper.class);

stateWiseCount.setReducerClass(NumUIDReducer.class);

stateWiseCount.setInputFormatClass(TextInputFormat.class);

stateWiseCount.setMapOutputKeyClass(Text.class);

stateWiseCount.setMapOutputValueClass(IntWritable.class);

stateWiseCount.setOutputKeyClass(Text.class);

stateWiseCount.setOutputValueClass(IntWritable.class);

Path inputFilePath = new Path(args[0]);

Path outputFilePath = new Path(args[1]);

FileInputFormat.addInputPath(stateWiseCount, inputFilePath);

FileOutputFormat.setOutputPath(stateWiseCount, outputFilePath);

FileSystem fs = FileSystem.newInstance(getConf());

if (fs.exists(outputFilePath)) {

fs.delete(outputFilePath, true);

}

stateWiseCount.waitForCompletion(true);

Job sort = Job.getInstance(getConf());

sort.setJobName("Sorting States on Num Aadhaars generated");

sort.setJarByClass(Driver.class);

sort.setOutputKeyClass(Text.class);

sort.setOutputValueClass(IntWritable.class);

sort.setMapperClass(sortMapper.class);

sort.setReducerClass(sortReducer.class);

sort.setSortComparatorClass(sortComparator.class);

sort.setMapOutputKeyClass(IntWritable.class);

sort.setMapOutputValueClass(Text.class);

FileInputFormat.addInputPath(sort, new Path(args[1]));

FileOutputFormat.setOutputPath(sort, new Path(args[2]));

if (fs.exists(new Path(args[2]))) {

fs.delete(new Path(args[2]), true);

}

return sort.waitForCompletion(true) ? 0 : 1;

}

public static void main(String[] args) throws Exception {

ToolRunner.run(new Driver(), args);

}

}

**Build jar and execute**

**output**

