**Project Report**

This is Boston airbnb dataset which I took from kaggle.com. There are 3 tables, listings, reviews and calendar respectively.

Analysis performed:

* Total reviews collected per listings.
* Minimum and maximum prices per listings according to date.
* Cheapest and costliest listings per neighborhood.
* Optimizing the above code by including combiner.
* Bloom filters for catching positive comments for every listing
* Use of partitioner to segregate the listings data according to last review date.
* Segregate data according to the size of bedrooms each apartment is having.
* Taking out top listings having maximum rating score
* Join the listing and review tables.
* Positive and total comments ratio/ percentage.

Dataset Link:

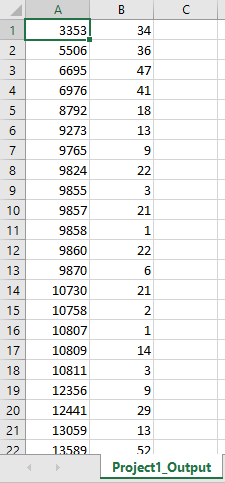
<https://www.kaggle.com/airbnb/boston>

Analysis:

1. Part 1: Analysis of reviews per listing id. It is analyzed to find out for every listing/ apartment how many reviews have been recorded so far. This process has further been optimized by using combiner.

Without combiner time: 9.87 sec

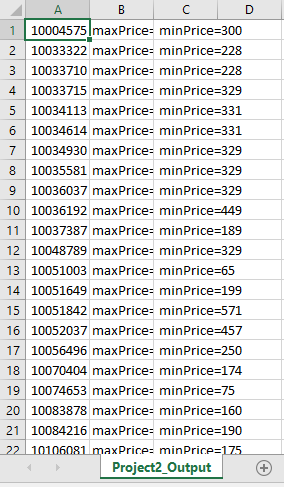
With combiner time: 7.74 sec



1. Part2: Found out minimum and maximum apartment price for each apartment from the calendar table. This analysis can be found used to analyze on which date what was the price for apartment. This code was further optimized using combiner.

Without combiner time: 7.02 sec

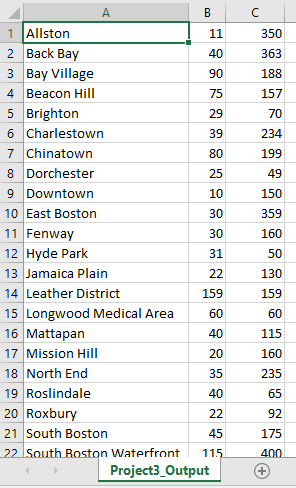
With combiner time: 5.37 sec



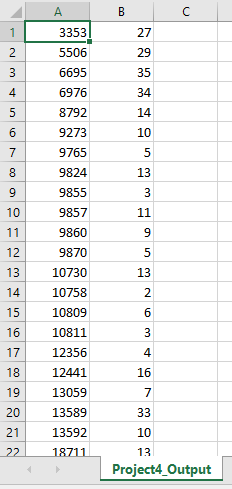
1. For each neighborhood, minimum and maximum rent for apartments are calculated to see what are the trends going for each area. Combiner optimization is also done.

Without combiner time: 7.74 sec

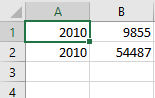
With combiner time: 6.39 sec

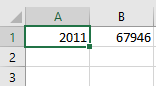


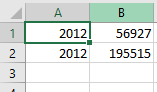
1. Bloom filters are used to find out the positive comments for every listing. Certain hot values are predefined in the filters and every token for the review is compared with the filter values. If any of the values matches, then those listing id’ are sent in the results from reducer. These results are further used to calculate positive comment percentages for every apartment.

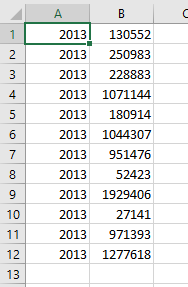


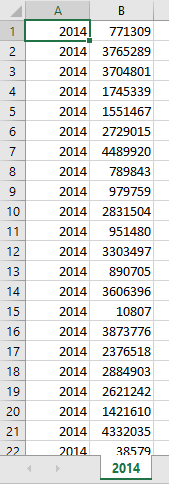
1. Apartments are partitioned according to the last review year. So we can analyze to see the latest comments dates for every apartment. If there are no comments in past days, it can be assumed that there is some problem with that apartment.

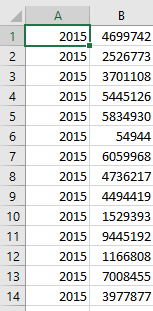


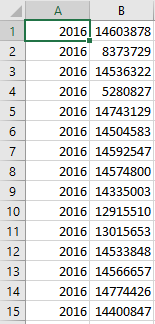




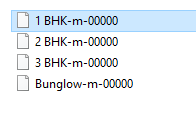




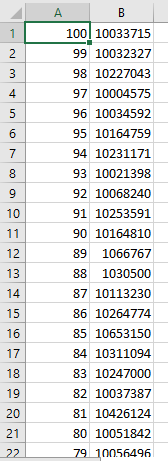




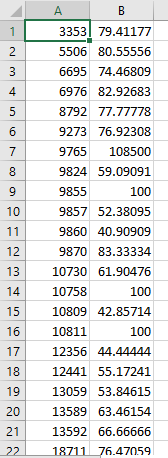
1. Binning pattern is implemented to find out which apartments belong to which category based on bedroom numbers like 1BHK, 2 BHK, 3BHK, Bungalow etc.



1. Top apartments are found out based on their rating score. We use chaining pattern here to combine 2 map-reduce jobs one by another.



1. Listing and reviews table are joined using inner join method using listingId in common. This view can be used to analyze the overall idea in one data table.
2. Positive comment percentage is calculated using previous 2 outputs as inputs. This is the analysis for every apartment which tells us the quality of apartment according to our criteria.



Appendix:

**Project1:**

**Mapper:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project1;

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;

/\*\*

\*

\* @author mugdha

\*/

public class ReviewsCountMapper extends Mapper<Object, Text, LongWritable, IntWritable> {

private long listingId;

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

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

listingId = Long.parseLong(txt);

context.write(new LongWritable(listingId),new IntWritable(1));

}

}

**Combiner:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class ReviewsCountCombiner extends Reducer<LongWritable, IntWritable, LongWritable, IntWritable> {

private IntWritable result = new IntWritable();

protected void reduce(LongWritable key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int counter = 0;

for (IntWritable val : values) {

counter++;

}

result.set(counter);

context.write(key, result);

}

}

**Reducer:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class ReviewsCountReducer extends Reducer<LongWritable, IntWritable, LongWritable, IntWritable> {

// private IntWritable result = new IntWritable();

protected void reduce(LongWritable key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

//int counter =0;

for (IntWritable val : values) {

context.write(key, val);

}

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project1;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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.output.FileOutputFormat;

/\*\*

\*

\* @author mugdha

\*/

public class Project1 {

/\*\*

\* @param args the command line arguments

\*/

private static final String input = "/mugdha/reviews";

private static final String output = "/mugdha/output1";

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(Project1.class);

job.setMapperClass(ReviewsCountMapper.class);

job.setCombinerClass(ReviewsCountCombiner.class);

job.setReducerClass(ReviewsCountReducer.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output));

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

}

}

**Project2:**

**MinMaxAptPrice:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project2;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.Date;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author mugdha

\*/

public class MinMaxAptPrice implements Writable {

private Integer maxPrice;

private Integer minPrice;

public Integer getMaxPrice() {

return maxPrice;

}

public void setMaxPrice(Integer maxPrice) {

this.maxPrice = maxPrice;

}

public Integer getMinPrice() {

return minPrice;

}

public void setMinPrice(Integer minPrice) {

this.minPrice = minPrice;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeInt(maxPrice);

d.writeInt(minPrice);

}

@Override

public void readFields(DataInput di) throws IOException {

minPrice = di.readInt();

maxPrice = di.readInt();

}

@Override

public String toString() {

return "maxPrice=" + maxPrice + ", minPrice=" + minPrice;

}

}

**Mapper:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project2;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mugdha

\*/

public class MinMaxPriceMapper extends Mapper<Object, Text, Text, MinMaxAptPrice> {

private Text area = new Text();

private Integer min;

private Integer max;

private static final SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

private MinMaxAptPrice outPut = new MinMaxAptPrice();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

try {

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

area.set(campaignFields[0]);

min = Integer.parseInt(campaignFields[3]);

max = Integer.parseInt(campaignFields[3]);

char avail = value.toString().split(",")[2].charAt(0);

Date date = sdf.parse(campaignFields[1]);

if (area == null || min == null || max == null) {

return;

}

outPut.setMinPrice(min);

outPut.setMaxPrice(max);

// outPut.setMinDate(date);

// outPut.setMaxDate(date);

if (avail == 't') {

context.write(area, outPut);

}

} catch (ParseException ex) {

Logger.getLogger(MinMaxPriceMapper.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

**Combiner:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project2;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class MinMaxPriceCombiner extends Reducer<Text, MinMaxAptPrice, Text, MinMaxAptPrice> {

private MinMaxAptPrice result = new MinMaxAptPrice();

@Override

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

//result.setMinDate(null);

//result.setMaxDate(null);

result.setMinPrice(null);

result.setMaxPrice(null);

for (MinMaxAptPrice val : values) {

if (result.getMinPrice() == null || val.getMinPrice().compareTo(result.getMinPrice()) < 0) {

result.setMinPrice(val.getMinPrice());// result.setMinDate(val.getMinDate());

}

result.setMaxPrice(val.getMaxPrice());

}

context.write(key, result);

}

}

**Reducer:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project2;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class MinMaxPriceReducer extends Reducer<Text, MinMaxAptPrice, Text, MinMaxAptPrice> {

@Override

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

for (MinMaxAptPrice val : values) {

context.write(key, val);

}

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project2;

import java.io.IOException;

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.lib.input.FileInputFormat;

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

/\*\*

\*

\* @author mugdha

\*/

public class Project2 {

/\*\*

\* @param args the command line arguments

\*/

private static final String input = "/mugdha/calendar";

private static final String output= "/mugdha/output2";

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(Project2.class);

job.setMapperClass(MinMaxPriceMapper.class);

job.setReducerClass(MinMaxPriceReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(MinMaxAptPrice.class);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output));

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

**Project 3:**

**MinMaxTuple:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project3;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author mugdha

\*/

public class MinMaxTuple implements Writable {

Integer minRent;

Integer maxRent;

public MinMaxTuple() {

minRent = 0;

maxRent = 0;

}

void setMinDuration(Integer duration) {

this.minRent = duration;

}

void setMaxDuration(Integer duration) {

this.maxRent = duration;

}

Integer getMinDuration() {

return minRent;

}

Integer getMaxDuration() {

return maxRent;

}

@Override

public void write(DataOutput out) throws IOException {

out.writeInt(minRent);

out.writeInt(maxRent);

}

@Override

public void readFields(DataInput in) throws IOException {

minRent = new Integer(in.readInt());

maxRent = new Integer(in.readInt());

}

public String toString() {

return minRent + "\t" + maxRent;

}

}

**Mapper:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project3;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mugdha

\*/

public class NeighbourhoodRentMapper extends Mapper<Object, Text, Text, MinMaxTuple> {

private Text area= new Text();

private Integer min;

private Integer max;

private MinMaxTuple outPut= new MinMaxTuple();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

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

area.set(campaignFields[11]);

min=Integer.parseInt(campaignFields[19]);

max=Integer.parseInt(campaignFields[19]);

if (area == null || min == null || max== null) {

return;

}

outPut.setMinDuration(min);

outPut.setMaxDuration(max);

context.write(area,outPut);

}

}

**Combiner:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project3;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class NeighbourhoodRentCombiner extends Reducer<Text, MinMaxTuple, Text, MinMaxTuple> {

private MinMaxTuple resultRow = new MinMaxTuple();

public void reduce(Text key, Iterable<MinMaxTuple> values,

Context context

) throws IOException, InterruptedException {

Integer minduration = 0;

Integer maxduration = 0;

resultRow.setMinDuration(null);

resultRow.setMaxDuration(null);

for (MinMaxTuple val : values) {

minduration = val.getMinDuration();

maxduration = val.getMaxDuration();

// get min score

if (resultRow.getMinDuration() == null || minduration.compareTo(resultRow.getMinDuration()) < 0) {

resultRow.setMinDuration(minduration);

}

resultRow.setMaxDuration(maxduration);

}

context.write (key, resultRow);

}

}

**Reducer:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project3;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class NeighbourhoodRentReducer extends Reducer<Text, MinMaxTuple, Text, MinMaxTuple> {

public void reduce(Text key, Iterable<MinMaxTuple> values, Context context) throws IOException, InterruptedException {

for (MinMaxTuple val : values) {

context.write(key, val);

}

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project3;

import java.io.IOException;

import java.sql.Timestamp;

import org.apache.hadoop.conf.Configuration;

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.output.FileOutputFormat;

/\*\*

\*

\* @author mugdha

\*/

public class Project3 {

/\*\*

\* @param args the command line arguments

\*/

private static final String input = "/mugdha/listings";

private static final String output= "/mugdha/output3\_combiner";

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(Project3.class);

job.setMapperClass(NeighbourhoodRentMapper.class);

job.setCombinerClass(NeighbourhoodRentCombiner.class);

job.setReducerClass(NeighbourhoodRentReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(MinMaxTuple.class);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output));

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

}}

**Project4:**

**Filter:**

package project4;

public class Filter {

String c;

Filter(String category){

this.c = category;

}

}

**Mapper:**

package project4;

import com.google.common.base.Charsets;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

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.lib.input.FileInputFormat;

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import com.google.common.hash.BloomFilter;

import com.google.common.hash.Funnel;

import com.google.common.hash.Sink;

import org.apache.hadoop.io.IntWritable;

public class Project4 extends Configured implements Tool {

public static class BloomFilterMapper extends Mapper<Object, Text, IntWritable, IntWritable> {

Funnel<Filter> p = new Funnel<Filter>() {

@Override

public void funnel(Filter mc, Sink into) {

into.putString(mc.c, Charsets.UTF\_8);

}

};

private BloomFilter<Filter> filterList = BloomFilter.create(p, 500, 0.1);

@Override

public void setup(Mapper.Context context) throws IOException, InterruptedException {

Filter p1 = new Filter("nice");

Filter p2 = new Filter("cool");

Filter p3 = new Filter("good");

Filter p4 = new Filter("great");

Filter p5 = new Filter("pleasant");

Filter p6 = new Filter("excellent");

Filter p7 = new Filter("comfortable");

Filter p8 = new Filter("beautiful");

Filter p9 = new Filter("perfect");

Filter p10 = new Filter("friendly");

ArrayList<Filter> catList = new ArrayList<>();

catList.add(p1);

catList.add(p2);

catList.add(p3);

catList.add(p4);

catList.add(p5);

catList.add(p6);

catList.add(p7);

catList.add(p8);

catList.add(p9);

catList.add(p10);

for (Filter pr : catList) {

filterList.put(pr);

}

}

@Override

public void map(Object key, Text value, Mapper.Context context) throws IOException, InterruptedException {

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

String id = values[0];

int listingId = Integer.parseInt(id);

int count = 0;

String val[] = values[5].split(" ");

for (String val1 : val) {

Filter mc = new Filter(val1.trim());

if (filterList.mightContain(mc)) {

count++;

}

}

if(count > 0){

context.write(new IntWritable(listingId), new IntWritable(1));

}

}

}

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

int res = ToolRunner.run(new Configuration(), new Project4(), args);

System.exit(res);

}

private static final String input = "/mugdha/reviews";

private static final String output = "/mugdha/output4";

@Override

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Bloom Filter");

job.setJarByClass(Project4.class);

job.setMapperClass(BloomFilterMapper.class);

job.setReducerClass(BloomFilterDistinctReducer.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output));

boolean success = job.waitForCompletion(true);

System.out.println(success);

return success ? 0 : 1;

}

}

**Reducer:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project4;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class BloomFilterDistinctReducer extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {

@Override

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

int count = 0;

for(IntWritable val : values){

count++;

}

context.write(key, new IntWritable(count));

}

}

**Project 5:**

**Mapper:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project5;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mugdha

\*/

public class LastReviewMapper extends Mapper<Object, Text, IntWritable, IntWritable> {

IntWritable outKey = new IntWritable();

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

int id = Integer.parseInt(value.toString().split(",")[0]);

String lastReview = value.toString().split(",")[21];

if (lastReview.equals("0")) {

}

else{

String y = lastReview.split("/")[2];

int year = Integer.parseInt(y);

context.write(new IntWritable(year), new IntWritable(id));

}

}

}

**Partitioner:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project5;

import org.apache.hadoop.conf.Configurable;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Partitioner;

/\*\*

\*

\* @author mugdha

\*/

public class LastReviewPartitioner extends Partitioner<IntWritable, IntWritable> implements Configurable{

public static final String lastAccess = "last.access";

private Configuration conf = null;

private int minLastAccessDateYear = 0;

@Override

public int getPartition(IntWritable key, IntWritable value, int i) {

return key.get() - minLastAccessDateYear;

}

public Configuration getConf(){

return conf;

}

public void setConf(Configuration conf){

this.conf = conf;

minLastAccessDateYear = conf.getInt(lastAccess, 0);

}

public static void setMinLastAccessDateYear(Job job, int minLastAccessDateYear) {

job.getConfiguration().setInt(lastAccess, minLastAccessDateYear);

}

}

**Reducer:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project5;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class LastReviewReducer extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {

@Override

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

for(IntWritable val : values){

context.write(key, val);

}

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project5;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

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

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

/\*\*

\*

\* @author mugdha

\*/

public class Project5 {

/\*\*

\* @param args the command line arguments

\*/

private static final String INPUT\_PATH = "/mugdha/listings";

private static final String OUTPUT\_PATH = "/mugdha/output5";

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(Project5.class);

job.setMapperClass(LastReviewMapper.class);

LastReviewPartitioner.setMinLastAccessDateYear(job, 2010);

job.setNumReduceTasks(7);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(INPUT\_PATH));

FileOutputFormat.setOutputPath(job, new Path(OUTPUT\_PATH));

job.waitForCompletion(true);

}

}

Project 6:

**Mapper:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project6;

import java.io.IOException;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

/\*\*

\*

\* @author mugdha

\*/

public class BedRoomBins extends Mapper<Object, Text, Text, NullWritable> {

private MultipleOutputs<Text, NullWritable> mulOutput = null;

@Override

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

mulOutput = new MultipleOutputs(context);

}

@Override

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

String bedroom = value.toString().split(",")[17];

if(bedroom.equals("1")){

mulOutput.write("bins", value, NullWritable.get(), "1 BHK");

}

if(bedroom.equals("2")){

mulOutput.write("bins", value, NullWritable.get(), "2 BHK");

}

if(bedroom.equals("3")){

mulOutput.write("bins", value, NullWritable.get(), "3 BHK");

}

if(bedroom.equals("0")){

}

else{

mulOutput.write("bins", value, NullWritable.get(), "Bunglow");

}

}

@Override

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

mulOutput.close();

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project6;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

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.output.FileOutputFormat;

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

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

/\*\*

\*

\* @author mugdha

\*/

public class Project6 {

/\*\*

\* @param args the command line arguments

\*/

private static final String input = "/mugdha/listings";

private static final String output = "/mugdha/output6";

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Binning");

job.setJarByClass(Project6.class);

job.setMapperClass(BedRoomBins.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(NullWritable.class);

MultipleOutputs.addNamedOutput(job, "bins", TextOutputFormat.class, Text.class, NullWritable.class);

MultipleOutputs.setCountersEnabled(job, true);

job.setNumReduceTasks(0);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output));

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

}

}

**Project 7:**

**Mapper1:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project8;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mugdha

\*/

public class Mapper1 {

static Map<Text, FloatWritable> countMap = new HashMap<>();

public static class TokenizerMapper1

extends Mapper<Object, Text, Text, FloatWritable> {

private String listingId;

private int rating;

public void map(Object key, Text value, Mapper.Context context

) throws IOException, InterruptedException {

listingId = value.toString().split(",")[0];

rating = Integer.parseInt(value.toString().split(",")[22]);

context.write(new Text(listingId), new FloatWritable(rating));

}

}

}

**Reducer1:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project8;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class Reducer1 {

static Map<Text, Float> countMap = new HashMap<Text, Float>();

public static class FloatSumReducer1

extends Reducer<Text, FloatWritable, Text, FloatWritable> {

private FloatWritable result = new FloatWritable();

public void reduce(Text key, Iterable<FloatWritable> values,

Context context

) throws IOException, InterruptedException {

for (FloatWritable val : values) {

context.write(key, val);

}

}

}

}

**Mapper2:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project8;

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mugdha

\*/

public class Mapper2 {

public static class TokenizerMapper2

extends Mapper<Object, Text, FloatWritable, Text> {

static TreeMap<Float, Text> countMap = new TreeMap<>();

private float fl = (float) 1.0;

FloatWritable one = new FloatWritable(fl);

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

context.write(one, value);

}

@Override

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

for (Text val : countMap.values()) {

context.write(one, val);

}

}

}

}

**Reducer2:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project8;

import java.io.IOException;

import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class Reducer2 {

public static class FloatSumReducer2

extends Reducer<FloatWritable, Text, FloatWritable, Text> {

String listingId;

Float rating;

public void reduce(FloatWritable key, Iterable<Text> values,

Context context

) throws IOException, InterruptedException {

TreeMap<Float, Text> countMap = new TreeMap<>();

for (Text value : values) {

listingId = value.toString().split("\t")[0];

rating = Float.parseFloat(value.toString().split("\t")[1]);

countMap.put(rating, new Text(listingId));

if (countMap.size() > 50) {

countMap.remove(countMap.firstKey());

}

}

for (Map.Entry<Float, Text> entry : countMap.descendingMap().entrySet()) {

Text value = entry.getValue();

context.write(new FloatWritable(entry.getKey()), new Text(value));

}

for (Text t : countMap.descendingMap().values()) {

String listingId = t.toString().split("\t")[0];

}

}

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project8;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

import project8.Mapper1.TokenizerMapper1;

import project8.Mapper2.TokenizerMapper2;

import project8.Reducer1.FloatSumReducer1;

import project8.Reducer2.FloatSumReducer2;

/\*\*

\*

\* @author mugdha

\*/

public class Project8 {

/\*\*

\* @param args the command line arguments

\*/

private static final String input = "/mugdha/listings";

private static final String middle = "/mugdha/output\_middle";

private static final String output = "/mugdha/output8";

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

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Job job = new Job(conf, "Job1");

job.setJarByClass(Project8.class);

job.setMapperClass(TokenizerMapper1.class);

job.setReducerClass(FloatSumReducer1.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(FloatWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

TextInputFormat.addInputPath(job, new Path(input));

TextOutputFormat.setOutputPath(job, new Path(middle));

job.waitForCompletion(true);

/\*

\* Job 2

\*/

Job job2 = new Job(conf, "Job 2");

job2.setJarByClass(Project8.class);

job2.setMapperClass(TokenizerMapper2.class);

job2.setReducerClass(FloatSumReducer2.class);

job2.setOutputKeyClass(FloatWritable.class);

job2.setOutputValueClass(Text.class);

job2.setInputFormatClass(TextInputFormat.class);

job2.setOutputFormatClass(TextOutputFormat.class);

TextInputFormat.addInputPath(job2, new Path(middle));

TextOutputFormat.setOutputPath(job2, new Path(output));

job2.waitForCompletion(true);

}

}

**Project 8:**

**Mapper & Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project9\_listingreviewjoin;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

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.MultipleInputs;

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

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

import org.apache.hadoop.util.Tool;

/\*\*

\*

\* @author mugdha

\*/

public class Project9\_ListingReviewJoin extends Configured implements Tool {

public static class ListingJoinMapper extends Mapper<Object, Text, Text, Text> {

private Text keyOut = new Text();

private Text valueOut = new Text();

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

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

String listingId = separatedInput[0];

if (listingId == null) {

return;

}

keyOut.set(listingId);

valueOut.set("L" + value.toString());

context.write(keyOut, valueOut);

}

}

public static class ReviewJoinMapper extends Mapper<Object, Text, Text, Text> {

private Text keyOut = new Text();

private Text valueOut = new Text();

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

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

String listingId = separatedInput[0];

if (listingId == null) {

return;

}

keyOut.set(listingId);

valueOut.set("R" + value.toString());

context.write(keyOut, valueOut);

}

}

public static class ListingReviewJoinReducer extends Reducer<Text, Text, Text, Text> {

private static final Text EMPTY\_TEXT = new Text("");

private Text tmp = new Text();

private ArrayList<Text> listings = new ArrayList<>();

private ArrayList<Text> reviews = new ArrayList<>();

private String joinType = null;

public void setup(Context context) {

joinType = context.getConfiguration().get("join.type");

}

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

listings.clear();

reviews.clear();

while (values.iterator().hasNext()) {

tmp = values.iterator().next();

if (Character.toString((char) tmp.charAt(0)).equals("L")) {

listings.add(new Text(tmp.toString().substring(1)));

}

if (Character.toString((char) tmp.charAt(0)).equals("R")) {

reviews.add(new Text(tmp.toString().substring(1)));

}

}

System.out.println(reviews.size());

performJoin(context);

}

private void performJoin(Context context) throws IOException, InterruptedException {

if (joinType.equalsIgnoreCase("inner")) {

if (!listings.isEmpty() && !reviews.isEmpty()) {

for (Text A : listings) {

for (Text B : reviews) {

context.write(A, B);

}

}

}

}

}

}

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

System.exit(new Project9\_ListingReviewJoin().run(args));

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "LabReduceSideJoin");

job.setJarByClass(Project9\_ListingReviewJoin.class);

MultipleInputs.addInputPath(job, new Path("/mugdha/reviews"), TextInputFormat.class, ListingJoinMapper.class); //"/home/mayur/NetBeansProjects/labReduceSideJoin/src/input1"

MultipleInputs.addInputPath(job, new Path("/mugdha/listings"), TextInputFormat.class, ReviewJoinMapper.class);//"/Users/austingnanarajnoah/Documents/workspace-sts/ReduceSideJoin/src/main/java/input2"

job.getConfiguration().set("join.type", "inner");

job.setReducerClass(ListingReviewJoinReducer.class);

job.setOutputFormatClass(TextOutputFormat.class);

TextOutputFormat.setOutputPath(job, new Path("/mugdha/output8"));

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

return job.waitForCompletion(true) ? 0 : 2;

}

}

**Project 9:**

**Mapper**:

package project10\_commentsratio;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mugdha

\*/

public class RatioMapper extends Mapper<Object, Text, LongWritable, FloatWritable> {

private long listingId;

@Override

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

String txt = value.toString().split("\t")[0];

String total= value.toString().split("\t")[1];

String hot= value.toString().split("\t")[3];

float t = Float.parseFloat(total);

float h = Float.parseFloat(hot);

listingId = Long.parseLong(txt);

float ratio = (h\*100)/t;

context.write(new LongWritable(listingId),new FloatWritable(ratio));

}

}

**Reducer:**

package project10\_commentsratio;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mugdha

\*/

public class RatioReducer extends Reducer<LongWritable, FloatWritable, LongWritable, FloatWritable>{

@Override

protected void reduce(LongWritable key, Iterable<FloatWritable> values, Context context) throws IOException, InterruptedException {

for(FloatWritable val : values){

context.write(key, val);

}

}

}

**Main:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package project10\_commentsratio;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

/\*\*

\*

\* @author mugdha

\*/

public class Project10\_CommentsRatio {

public static class TotalJoinMapper extends org.apache.hadoop.mapreduce.Mapper<Object, Text, Text, Text> {

private Text keyOut = new Text();

private Text valueOut = new Text();

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

String[] separatedInput = value.toString().split("\t");

String listingId = separatedInput[0];

if (listingId == null) {

return;

}

keyOut.set(listingId);

valueOut.set("T" + value.toString());

context.write(keyOut, valueOut);

}

}

public static class HotJoinMapper extends org.apache.hadoop.mapreduce.Mapper<Object, Text, Text, Text> {

private Text keyOut = new Text();

private Text valueOut = new Text();

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

String[] separatedInput = value.toString().split("\t");

String listingId = separatedInput[0];

if (listingId == null) {

return;

}

keyOut.set(listingId);

valueOut.set("P" + value.toString());

context.write(keyOut, valueOut);

}

}

public static class JoinReducer extends Reducer<Text, Text, Text, Text> {

private Text tmp = new Text();

private ArrayList<Text> total = new ArrayList<>();

private ArrayList<Text> positive = new ArrayList<>();

private String joinType = null;

public void setup(Context context) {

joinType = context.getConfiguration().get("join.type");

}

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

total.clear();

positive.clear();

while (values.iterator().hasNext()) {

tmp = values.iterator().next();

if (Character.toString((char) tmp.charAt(0)).equals("T")) {

total.add(new Text(tmp.toString().substring(1)));

}

if (Character.toString((char) tmp.charAt(0)).equals("P")) {

positive.add(new Text(tmp.toString().substring(1)));

}

}

System.out.println(positive.size());

executeJoinLogic(context);

}

private void executeJoinLogic(Reducer.Context context) throws IOException, InterruptedException {

if (joinType.equalsIgnoreCase("inner")) {

if (!total.isEmpty() && !positive.isEmpty()) {

System.out.println("here");

for (Text A : total) {

for (Text B : positive) {

context.write(A, B);

}

}

}

}

}

}

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

System.exit(new Project10\_CommentsRatio().run(args));

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "LabReduceSideJoin");

job.setJarByClass(Project10\_CommentsRatio.class);

MultipleInputs.addInputPath(job, new Path("/mugdha/p1out"), TextInputFormat.class, TotalJoinMapper.class); //"/home/mayur/NetBeansProjects/labReduceSideJoin/src/input1"

MultipleInputs.addInputPath(job, new Path("/mugdha/p4out"), TextInputFormat.class, HotJoinMapper.class);//"/Users/austingnanarajnoah/Documents/workspace-sts/ReduceSideJoin/src/main/java/input2"

job.getConfiguration().set("join.type", "inner");

job.setReducerClass(JoinReducer.class);

job.setOutputFormatClass(TextOutputFormat.class);

TextOutputFormat.setOutputPath(job, new Path("/mugdha/output9\_part")); // "TextOutputFormat.setOutputPath(job, new Path()); //"

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.waitForCompletion(true);

Job job2 = new Job(conf, "Job 2");

job2.setJarByClass(Project10\_CommentsRatio.class);

job2.setMapperClass(RatioMapper.class);

job2.setReducerClass(RatioReducer.class);

job2.setOutputKeyClass(LongWritable.class);

job2.setOutputValueClass(FloatWritable.class);

job2.setInputFormatClass(TextInputFormat.class);

job2.setOutputFormatClass(TextOutputFormat.class);

TextInputFormat.addInputPath(job2, new Path("/mugdha/output9\_part"));

TextOutputFormat.setOutputPath(job2, new Path("/mugdha/output9"));

return job2.waitForCompletion(true) ? 0 : 2;

}

}