**Project Report:**

**Dataset:**

United States Cancer Statistics Data

1999-2013 Dataset

**Dataset Link:**

<https://www.cdc.gov/cancer/npcr/uscs/download_data.htm>

**Dataset Description:**

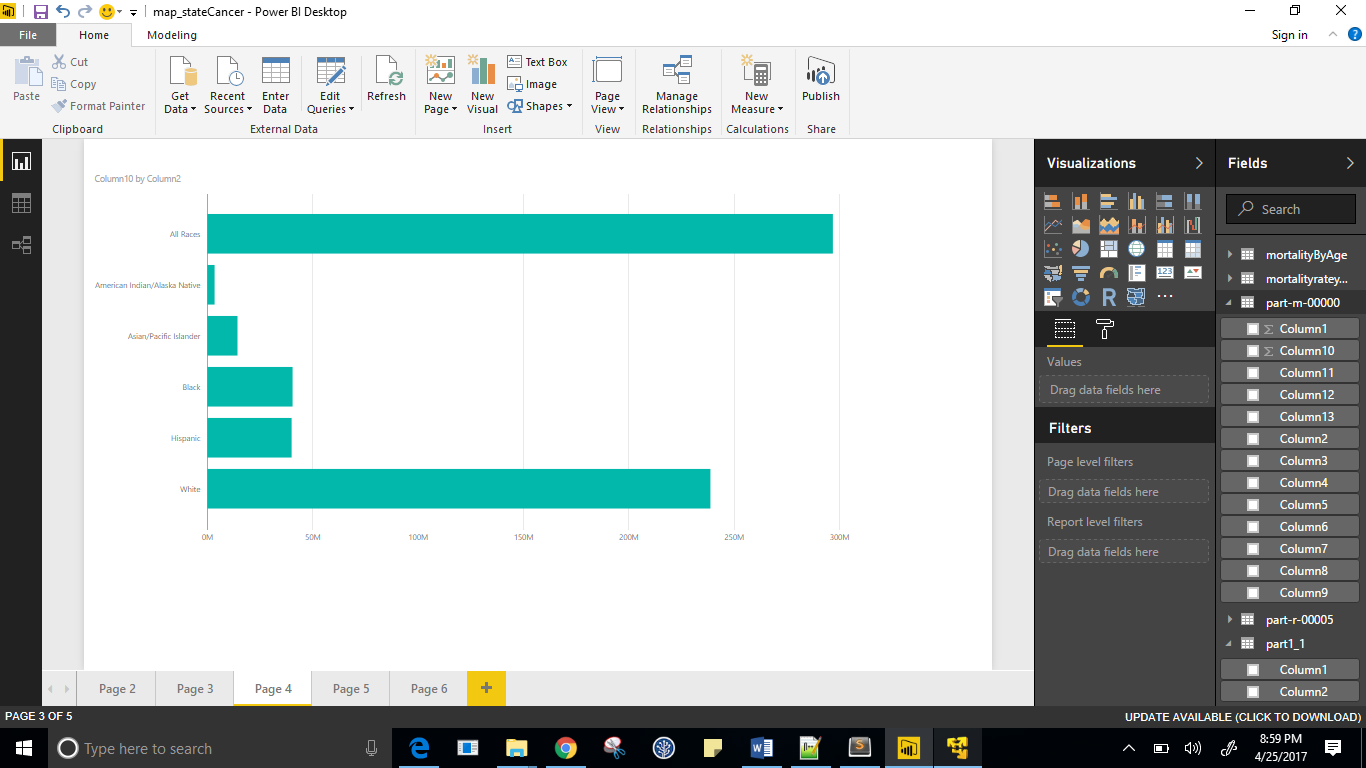
The dataset has data on cancer in United States. For project, I will be taking 5 files of the dataset. The data has different files for cancer in adult, cancer in children, by area, by site, etc. The files have around 1 million records. Each file helps in analyzing cancer mortality rates, occurrence of different type of cancer in different states in different races, to identify highest occurring cancer types among children. Below are few of these analysis that I have performed:

**Analysis (Data Visualization in PowerBI Tool):**

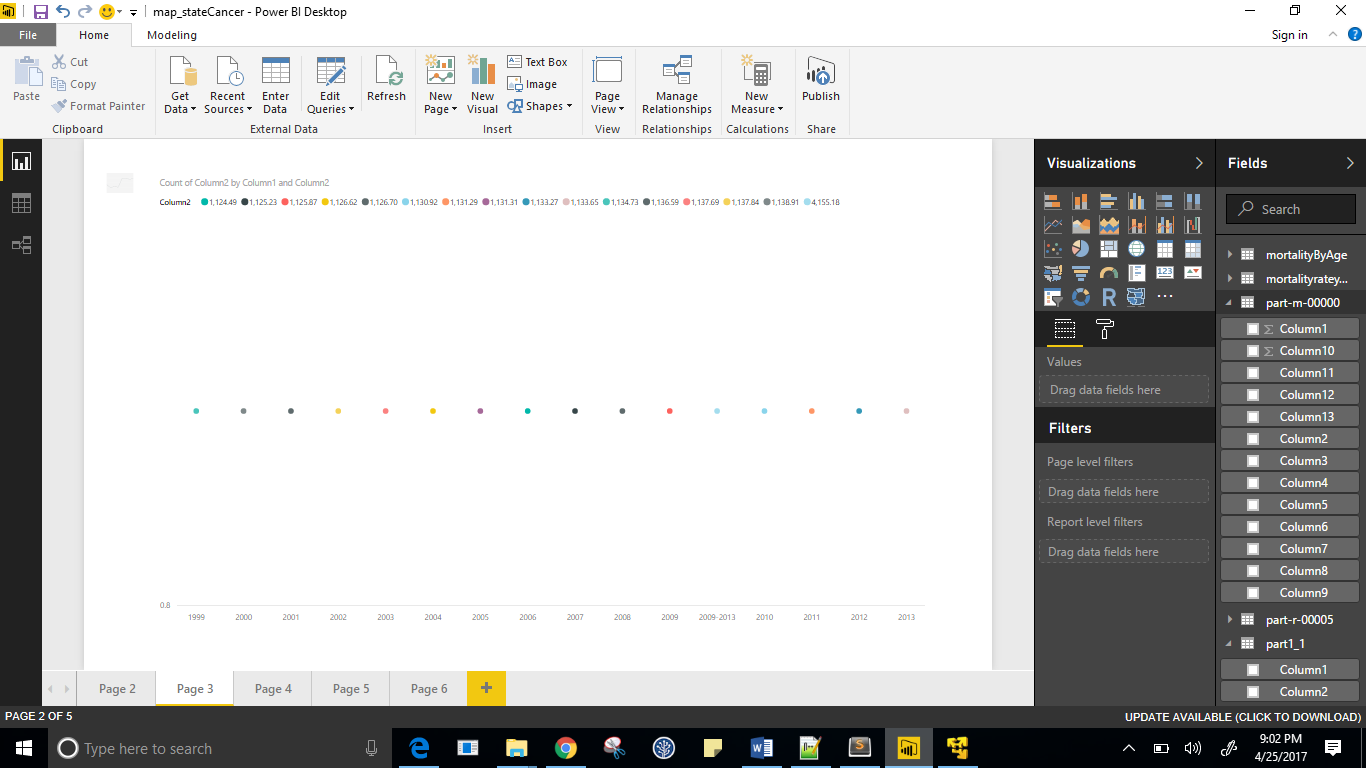
**PowerBI File:**

****

1. Bloom filter for particular year, gender and type of cancer.

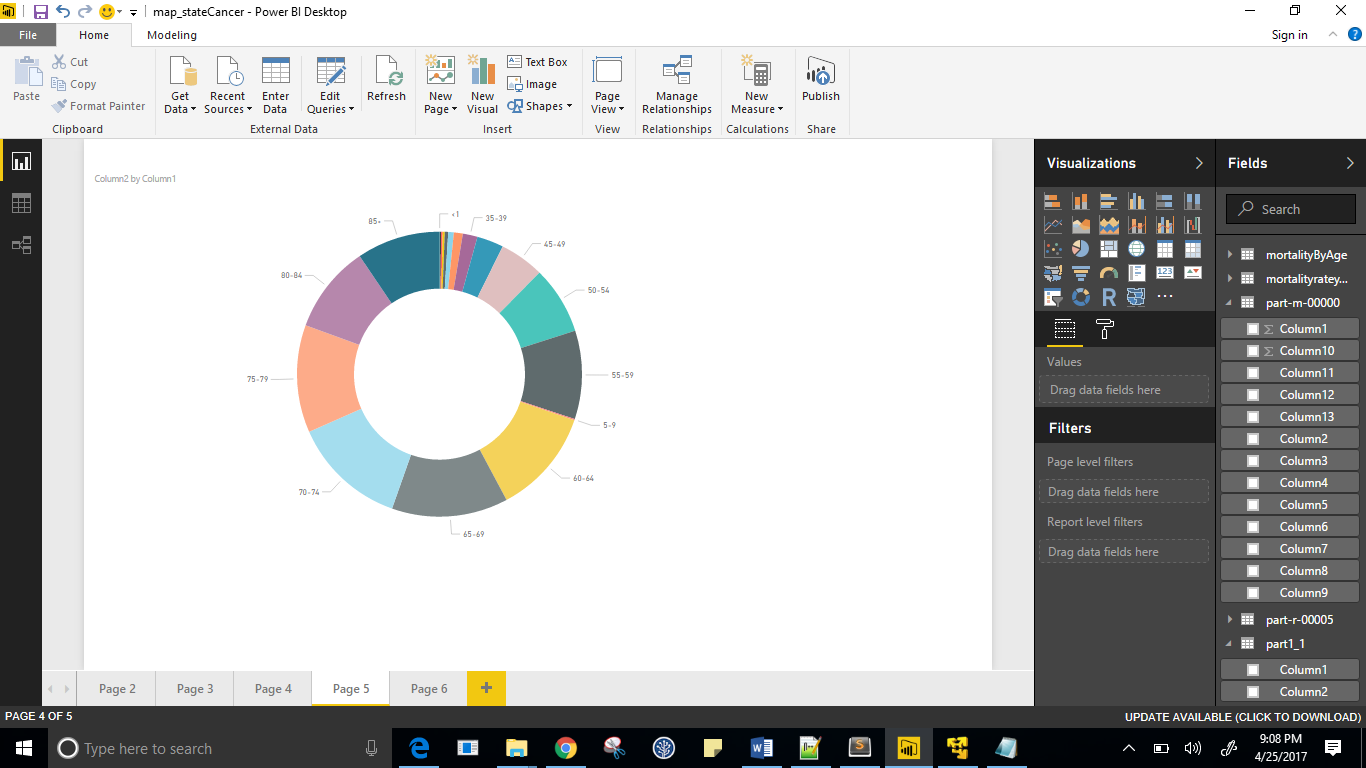


1. Average mortality rate over the years : summarization pattern

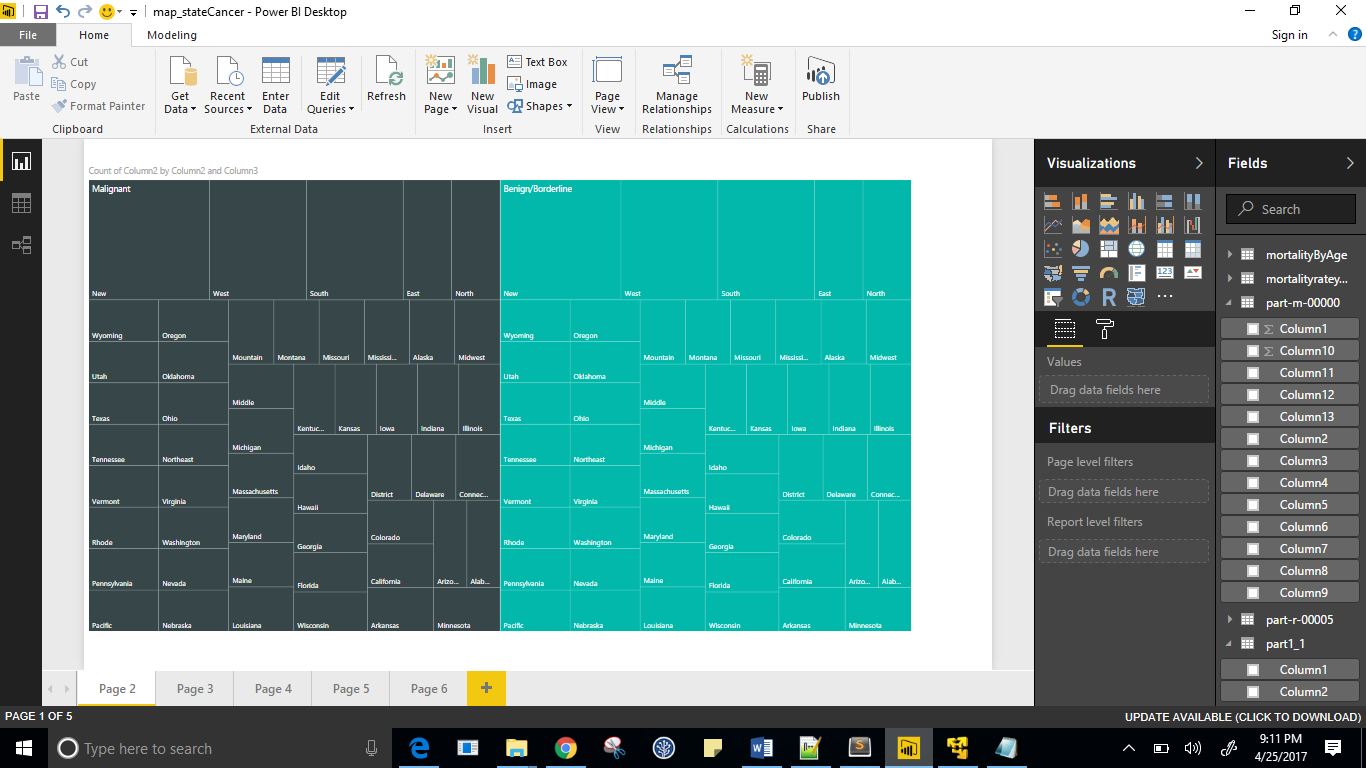


1. Binning /Partitioning pattern for every age group

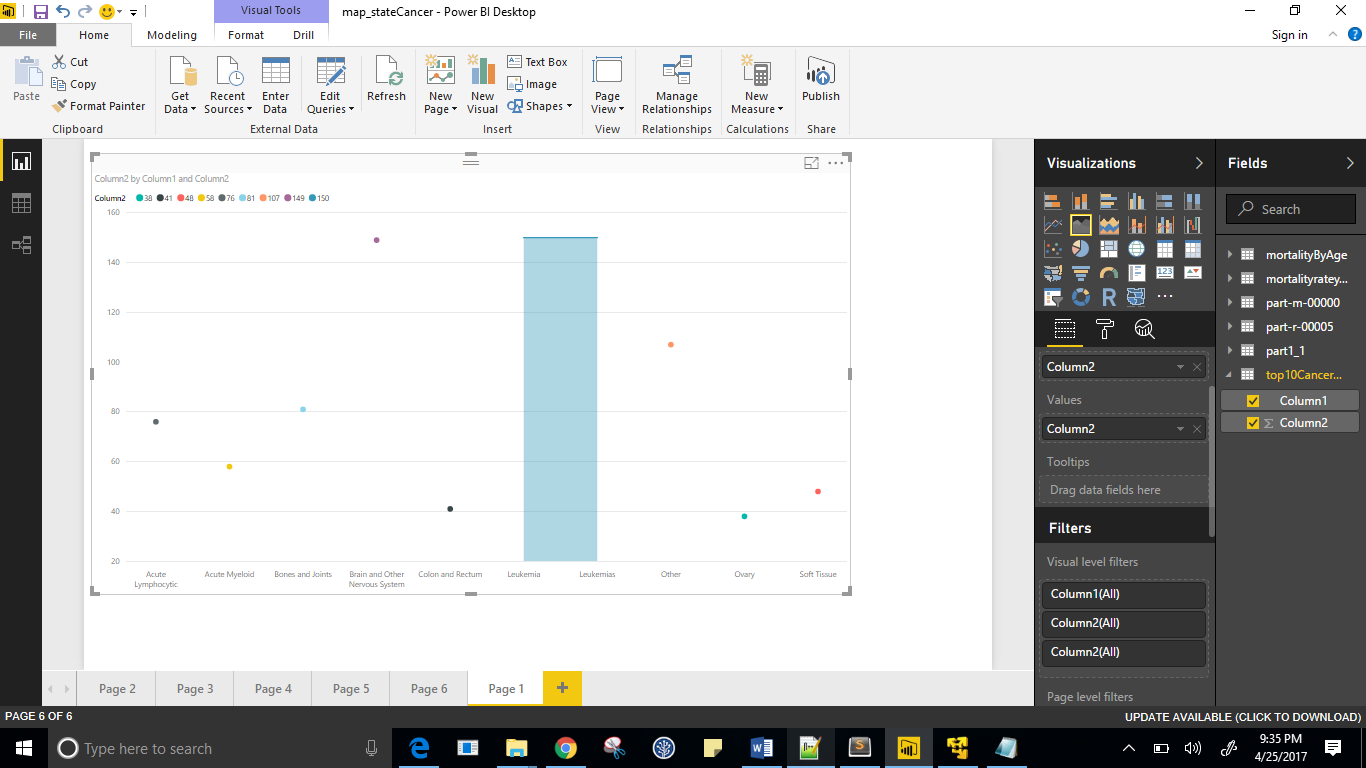
1. Job chaining – Mortality rate by age



1. For every year, getting cancer type and cancer region in each state



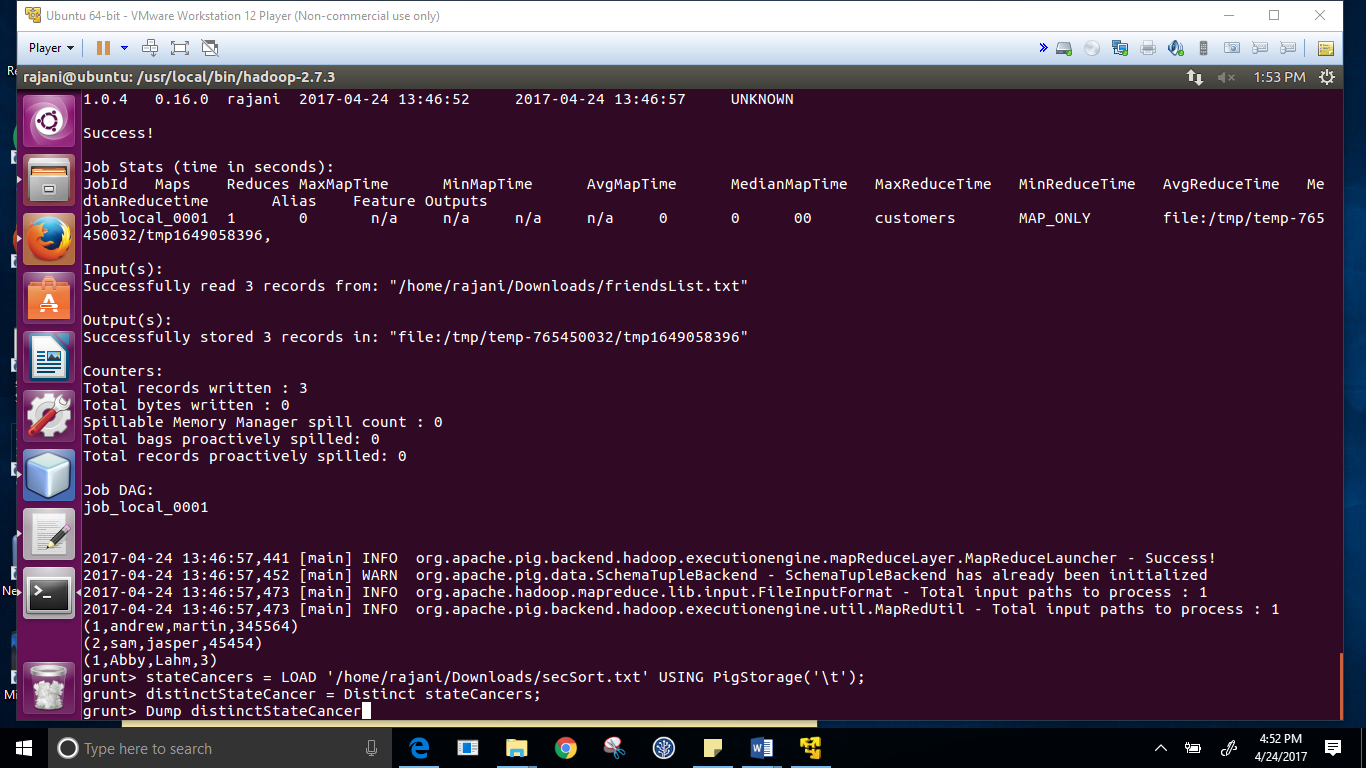
1. Top 10 cancer types in children

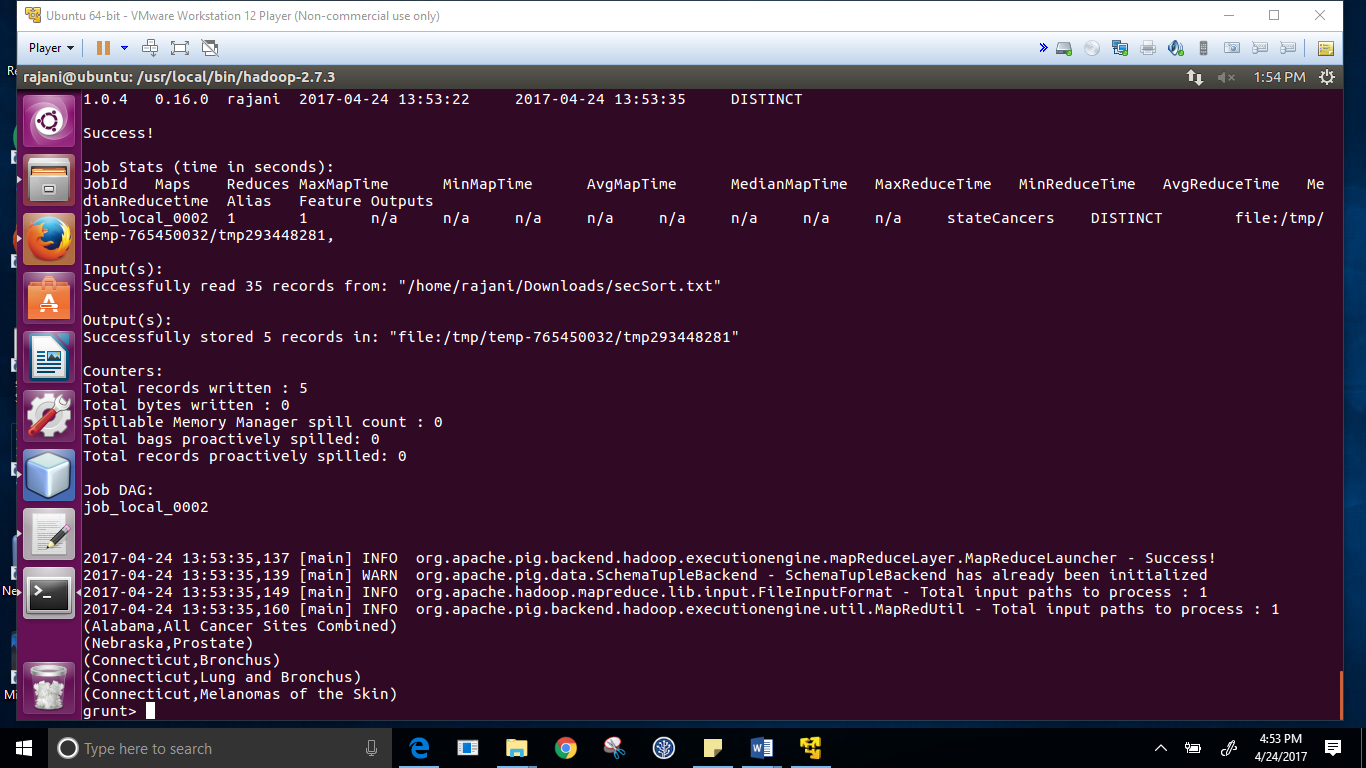


1. Distinct cancer types that has existed in USA from 1999-2013
2. Secondary Sorting to sort State and Cancer types in each state.

**PIG Implementation:**

The analysis 8 - State sorting gives repeated values in output, so I used PIG DISTINCT command in grunt to get distinct values. Below is the screen print of the same.





**Results/Conclusion/Analysis:**

1. From the data we observe that Race – White has more occurrence of cancer. *(http://www.cancerresearchuk.org/health-professional/cancer-statistics/incidence/ethnicity)*
2. We observe that every year the overall average mortality rate has remained constant, it has not increased which is good.
3. As age increases, mortality rate also increases. *(*[*https://medlineplus.gov/ency/article/004008.htm*](https://medlineplus.gov/ency/article/004008.htm) *- Resistance power of body decreases with age)*
4. New Jersey has highest number of Malignant and benign cases
5. Leukemia is the highest cause of cancer among children

**Source Code:**

1. **Ace\_RaceRegion**

/\*

\* 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 hw5\_4;

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.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.output.MultipleOutputs;

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

/\*\*

\*

\* @author rajani

\*/

public class HW5\_4 {

/\*\*

\* @param args the command line arguments

\*/

public static class BinningMapper extends Mapper<Object, Text, Text, NullWritable> {

private MultipleOutputs<Text, NullWritable> mos = null;

protected void setup(Context context) {

mos = new MultipleOutputs(context);

}

protected void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

try {

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

String hours = columns[0].trim();

String raceRegion = columns[0] + " " + columns[6] + " " + columns[9];

if (hours.equals("<1")) {

mos.write("bins", raceRegion, NullWritable.get(), "<1");

}

if (hours.equals("1-4")) {

mos.write("bins", raceRegion, NullWritable.get(), "1-4");

}

if (hours.equals("5-9")) {

mos.write("bins", raceRegion, NullWritable.get(), "5-9");

}

if (hours.equals("10-14")) {

mos.write("bins", raceRegion, NullWritable.get(), "10-14");

}

if (hours.equals("15-19")) {

mos.write("bins", raceRegion, NullWritable.get(), "15-19");

}

if (hours.equals("20-24")) {

mos.write("bins", raceRegion, NullWritable.get(), "20-24");

}

if (hours.equals("25-29")) {

mos.write("bins", raceRegion, NullWritable.get(), "25-29");

}

if (hours.equals("30-34")) {

mos.write("bins", raceRegion, NullWritable.get(), "30-34");

}

if (hours.equals("35-39")) {

mos.write("bins", raceRegion, NullWritable.get(), "35-39");

}

if (hours.equals("40-44")) {

mos.write("bins", raceRegion, NullWritable.get(), "40-44");

}

if (hours.equals("45-49")) {

mos.write("bins", raceRegion, NullWritable.get(), "45-49");

}

if (hours.equals("50-54")) {

mos.write("bins", raceRegion, NullWritable.get(), "50-54");

}

if (hours.equals("55-59")) {

mos.write("bins", raceRegion, NullWritable.get(), "55-59");

}

if (hours.equals("60-64")) {

mos.write("bins", raceRegion, NullWritable.get(), "60-64");

}

if (hours.equals("65-69")) {

mos.write("bins", raceRegion, NullWritable.get(), "65-69");

}

if (hours.equals("70-74")) {

mos.write("bins", raceRegion, NullWritable.get(), "70-74");

}

if (hours.equals("75-79")) {

mos.write("bins", raceRegion, NullWritable.get(), "75-79");

}

if (hours.equals("80-84")) {

mos.write("bins", raceRegion, NullWritable.get(), "80-84");

}

if (hours.equals("85+")) {

mos.write("bins", raceRegion, NullWritable.get(), "85+");

}

} catch (Exception e) {

e.printStackTrace();

}

}

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

mos.close();

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(HW5\_4.class);

job.setNumReduceTasks(0);

//job.setOutputKeyClass(Text.class);

//job.setOutputValueClass(NullWritable.class);

job.setMapperClass(BinningMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(NullWritable.class);

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

MultipleOutputs.setCountersEnabled(job, true);

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

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

//job.setNumReduceTasks(8);

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

}

}

1. **Brain\_JobChaining**

/\*

\* 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 lab3\_empchaining;

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.io.LongWritable;

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;

/\*\*

\*

\* @author rajani

\*/

public class Lab3\_EmpChaining {

/\*\*

\* @param args the command line arguments

\*/

public static class Map1 extends Mapper<LongWritable, Text, Text, IntWritable>{

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

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

if((!row[0].equals("~")) && (!row[3].equals("~"))){

String employeeId = row[0];

String salesAmount = row[3].trim();

try {

IntWritable sales = new IntWritable(Integer.parseInt(salesAmount));

context.write(new Text(employeeId), sales);

} catch (Exception e) {

}

}

}

}

public static class Reduce1 extends Reducer<Text, IntWritable, Text, IntWritable>{

private IntWritable totalSales = new IntWritable();

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

int sum = 0;

for(IntWritable val: values){

sum += val.get();

}

totalSales.set(sum);

context.write(key, totalSales);

}

}

public static class Map2

extends Mapper<LongWritable, Text, IntWritable, Text>{

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

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

Text employeeId = new Text(row[0]);

String salesAmount = row[1].trim();

try {

IntWritable count = new IntWritable(Integer.parseInt(salesAmount));

context.write(count, employeeId);

} catch (Exception e) {

}

}

}

public static class Reduce2 extends Reducer<IntWritable, Text, Text, IntWritable>{

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

for(Text val : value){

context.write(val, key);

}

}

}

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

Configuration conf1 = new Configuration();

Job job1 = Job.getInstance(conf1, "chaining");

job1.setJarByClass(Lab3\_EmpChaining.class);

job1.setMapperClass(Map1.class);

job1.setMapOutputKeyClass(Text.class);

job1.setMapOutputValueClass(IntWritable.class);

job1.setReducerClass(Reduce1.class);

job1.setOutputKeyClass(Text.class);

job1.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));

FileOutputFormat.setOutputPath(job1, new Path(args[1]));

boolean complete = job1.waitForCompletion(true);

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "chaining");

if(complete){

job2.setJarByClass(Lab3\_EmpChaining.class);

job2.setMapperClass(Map2.class);

job2.setMapOutputKeyClass(IntWritable.class);

job2.setMapOutputValueClass(Text.class);

job2.setReducerClass(Reduce2.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[2]));

System.exit(job2.waitForCompletion(true) ? 0 : 1);

}

}

}

1. **CancerYear\_BloomFilter**

/\*

\* 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 bloomfilter\_lab6;

import com.google.common.base.Charsets;

import com.google.common.hash.BloomFilter;

import com.google.common.hash.Funnel;

import com.google.common.hash.Sink;

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.NullWritable;

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;

/\*\*

\*

\* @author rajani

\*/

public class BloomFilter\_Lab6 {

/\*\*

\* @param args the command line arguments

\*/

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

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

@Override

public void funnel(Person person, Sink into) {

into.putInt(person.id).putString(person.firstName, Charsets.UTF\_8)

.putString(person.lastName, Charsets.UTF\_8)

//.putInt(person.birthYear)

;

}

};

private BloomFilter<Person> friends = BloomFilter.create(p, 500, 0.1);

@Override

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

// Person p1 = new Person(1999, "Female", "Acute Lymphocytic", 1593);

// Person p2 = new Person(2008, "Female", "Acute Lymphocytic", 2147);

Person p1 = new Person(1999, "Female", "Acute Lymphocytic");

Person p2 = new Person(2008, "Female", "Acute Lymphocytic");

ArrayList<Person> friendList = new ArrayList<Person>();

friendList.add(p1);

friendList.add(p2);

for(Person pr : friendList){

friends.put(pr);

}

}

@Override

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

try{

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

// Person p = new Person(Integer.parseInt(values[0]), values[2], values[3], Integer.parseInt(values[8]));

Person p = new Person(Integer.parseInt(values[0]), values[2], values[3]);

if((!values[0].equals("~")) && (!values[2].equals("~")) && (!values[3].equals("~")) && (!values[4].equals("Incidence")) && friends.mightContain(p)){

//if(friends.mightContain(p)){

context.write(value, NullWritable.get());

}

}catch(Exception e){

}

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(BloomFilter\_Lab6.class);

job.setMapperClass(BloomFilterMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(NullWritable.class);

job.setNumReduceTasks(0);

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

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

boolean success = job.waitForCompletion(true);

System.out.println(success);

}

}

/\*

\* 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 bloomfilter\_lab6;

/\*\*

\*

\* @author rajani

\*/

public class Person {

int id;

String firstName;

String lastName;

//int birthYear;

// Person(int id, String firstName, String lastName, int birthYear){

Person(int id, String firstName, String lastName){

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

//this.birthYear = birthYear;

}

}

1. **DistinctCancerType**

/\*

\* 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 hw5\_2;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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;

/\*\*

\*

\* @author rajani

\*/

public class HW5\_2 {

/\*\*

\* @param args the command line arguments

\*/

public static class DistinctPatternMapper extends Mapper<Object, Text, Text, NullWritable> {

@Override

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

LongWritable a = (LongWritable) key;

if(a.get() == 0)

{return;}

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

if(!columns[3].equals("~")){

String ipAdd = columns[3].trim();

context.write(new Text(ipAdd), NullWritable.get());

}

}

}

public static class DistinctPatternReducer extends Reducer<Text, NullWritable, Text, NullWritable>{

@Override

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

context.write(key, NullWritable.get());

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(HW5\_2.class);

job.setMapperClass(DistinctPatternMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(NullWritable.class);

job.setReducerClass(DistinctPatternReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(NullWritable.class);

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

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

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

}

}

1. **StateCancerType\_Join**

/\*

\* 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 lab7\_join1;

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.util.Tool;

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.ToolRunner;

/\*\*

\*

\* @author rajani

\*/

public class Lab7\_Join1 extends Configured implements Tool{

public static class JoinMapper1 extends Mapper<Object, Text, Text, Text>{

private Text outKey = new Text();

private Text outValue = new Text();

@Override

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

try{

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

String id = separatedInput[9];

if(id == null){

return;

}

outKey.set(id);

outValue.set("A" + separatedInput[9] + " " + separatedInput[4]);

context.write(outKey, outValue);

}catch(Exception e){

}

}

}

public static class JoinMapper2 extends Mapper<Object, Text, Text, Text>{

private Text outKey = new Text();

private Text outValue = new Text();

@Override

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

try{

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

String id = separatedInput[10].trim();

//String id = value.toString().split("[|]")[10].trim();

if(id == null){

return;

}

if(!separatedInput[5].equals("Incidence")){

outKey.set(id);

outValue.set("B" + separatedInput[0] + " " + separatedInput[9]);

context.write(outKey, outValue);

}

}catch(Exception e){

}

}

}

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

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

private Text tmp = new Text();

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

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

private String joinType = null;

public void setup(Context context){

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

}

@Override

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

listA.clear();

listB.clear();

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

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

if(tmp.charAt(0) == 'A'){

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

}else if(tmp.charAt(0) == 'B'){

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

}

}

executeJoinLogic(context);

}

private void executeJoinLogic(Context context){

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

for(Text A: listA){

if(!listB.isEmpty()){

for(Text B : listB){

try{

context.write(A, B);

}catch(Exception e){

}

}

}else{

try{

context.write(A, EMPTY\_TEXT);

}catch(Exception e){

}

}

}

}else if(joinType.equalsIgnoreCase("rightouter")){

for(Text B: listB){

if(!listA.isEmpty()){

for(Text A: listA){

try{

context.write(A, B);

}catch(Exception e){

}

}

}else{

try{

context.write(EMPTY\_TEXT, B);

}catch(Exception e){

}

}

}

}else if(joinType.equalsIgnoreCase("fullouter")){

if(!listA.isEmpty()){

for(Text A: listA){

if(listB.isEmpty()){

for(Text B: listB){

try{

context.write(A, B);

}catch(Exception e){

}

}

}else{

try{

context.write(A, EMPTY\_TEXT);

}catch(Exception e){

}

}

}

}else{

for(Text B: listB){

try{

context.write(EMPTY\_TEXT, B);

}catch(Exception e){

}

}

}

}else if(joinType.equalsIgnoreCase("anti")){

if(listA.isEmpty() ^ listB.isEmpty()){

for(Text A: listA){

try{

context.write(A, EMPTY\_TEXT);

}catch(Exception e){

}

}

for(Text B : listB){

try{

context.write(EMPTY\_TEXT, B);

}catch(Exception e){

}

}

}

}

}

}

public static void main(String[] args) {

try{

int res = ToolRunner.run(new Configuration(), new Lab7\_Join1(), args);

}catch(Exception e){

}

}

@Override

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

Configuration conf = new Configuration();

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

job.setJarByClass(Lab7\_Join1.class);

MultipleInputs.addInputPath(job, new Path(strings[0]), TextInputFormat.class, JoinMapper1.class);

MultipleInputs.addInputPath(job, new Path(strings[1]), TextInputFormat.class, JoinMapper2.class);

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

job.setReducerClass(JoinReducer.class);

job.setOutputFormatClass(TextOutputFormat.class);

TextOutputFormat.setOutputPath(job, new Path(strings[2]));

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

boolean success = job.waitForCompletion(true);

return success ? 0 : 2;

}

}

1. **MortalityByYear**

/\*

\* 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 mortalitybyyear;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author rajani

\*/

public class CountAverageTuple implements Writable {

private double count;

private double average;

public double getCount() {

return count;

}

public void setCount(double count) {

this.count = count;

}

public double getAverage() {

return average;

}

public void setAverage(double average) {

this.average = average;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeDouble(count);

d.writeDouble(average);

}

@Override

public void readFields(DataInput di) throws IOException {

count = di.readDouble();

average = di.readDouble();

}

public String toString(){

return (this.getAverage()+ "\t" + this.getCount());

}

}

/\*

\* 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 mortalitybyyear;

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

/\*\*

\*

\* @author rajani

\*/

public class MortalityByYear {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

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

job.setJarByClass(MortalityByYear.class);

//job.setNumReduceTasks(0);

job.setMapperClass(Averagemapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CountAverageTuple.class);

job.setCombinerClass(AverageReducer.class);

job.setReducerClass(AverageReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CountAverageTuple.class);

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

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

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

}

public static class Averagemapper extends Mapper<Object, Text, Text, CountAverageTuple> {

CountAverageTuple outCountAverage = new CountAverageTuple();

@Override

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

try{

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

String stockDate = row[10];

//String stockDate = row[0];

if((row[5].equals("Mortality")) && (!row[4].equals("~"))){

outCountAverage.setAverage(Double.parseDouble(row[4]));

outCountAverage.setCount(1);

context.write(new Text(stockDate), outCountAverage);

//context.write(new Text(stockDate), new Text(stockDate));

}

}catch(Exception e){

}

}

}

public static class AverageReducer extends Reducer<Text, CountAverageTuple, Text, CountAverageTuple> {

private CountAverageTuple result = new CountAverageTuple();

@Override

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

double sum = 0;

double count = 0;

for (CountAverageTuple val : values){

sum += val.getCount()\*val.getAverage();

count += val.getCount();

}

result.setCount(count);

result.setAverage(sum/count);

context.write(key, result);

}

}

}

1. **State\_SecondarySort**

/\*

\* 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 lab2\_sorting;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author rajani

\*/

class CompositeKeyWritable implements Writable, WritableComparable<CompositeKeyWritable> {

private String deptNo;

private String lastName;

public CompositeKeyWritable(){

}

public CompositeKeyWritable(String d, String l){

this.deptNo = d;

this.lastName = l;

}

@Override

public void write(DataOutput d) throws IOException {

WritableUtils.writeString(d, deptNo);

WritableUtils.writeString(d, lastName);

}

@Override

public void readFields(DataInput di) throws IOException {

deptNo = WritableUtils.readString(di);

lastName = WritableUtils.readString(di);

}

@Override

public int compareTo(CompositeKeyWritable o) {

int result = deptNo.compareTo(o.deptNo);

if(result==0){

result = lastName.compareTo(o.lastName);

}

return (-1)\*result;

}

public String getDeptNo() {

return deptNo;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public void setDeptNo(String deptNo) {

this.deptNo = deptNo;

}

public String toString(){

return (new StringBuilder().append(deptNo).append("\t").append(lastName).toString());

}

}

/\*

\* 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 lab2\_sorting;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author rajani

\*/

public class GroupingComparator extends WritableComparator{

protected GroupingComparator(){

super(CompositeKeyWritable.class, true);

}

@Override

public int compare(WritableComparable w1, WritableComparable w2) {

CompositeKeyWritable cw1 = (CompositeKeyWritable)w1;

CompositeKeyWritable cw2 = (CompositeKeyWritable)w2;

return cw1.getDeptNo().compareTo(cw2.getDeptNo());

}

}

/\*

\* 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 lab2\_sorting;

import java.io.IOException;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author rajani

\*/

public class Lab2\_Mapper extends Mapper<Object, Text, CompositeKeyWritable, NullWritable>{

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

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

if((!values[0].equals("~")) && (!values[9].equals("~"))){

CompositeKeyWritable cw = new CompositeKeyWritable(values[0], values[9]);

try{

context.write(cw, NullWritable.get());

//context.write(cw, new Text(values[10]));

}catch(IOException | InterruptedException ex){

System.out.println("Error Message" + ex.getMessage());

}

}

}

}

/\*

\* 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 lab2\_sorting;

import java.io.IOException;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author rajani

\*/

public class Lab2\_Reducer extends Reducer<CompositeKeyWritable, NullWritable, CompositeKeyWritable, NullWritable> {

public void reduce(CompositeKeyWritable key, Iterable<NullWritable> values, Context context){

for(NullWritable val : values){

try{

context.write(key, NullWritable.get());

}catch(IOException | InterruptedException ex){

System.out.println("Error Message" + ex.getMessage());

}

}

}

}

/\*

\* 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 lab2\_sorting;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

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.input.TextInputFormat;

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

/\*\*

\*

\* @author rajani

\*/

public class Lab2\_Sorting {

/\*\*

\* @param args the command line arguments

\*/

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

try{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf , "secondary sort");

job.setJarByClass(Lab2\_Sorting.class);

job.setMapperClass(Lab2\_Mapper.class);

job.setReducerClass(Lab2\_Reducer.class);

job.setMapOutputKeyClass(CompositeKeyWritable.class);

job.setMapOutputValueClass(NullWritable.class);

job.setGroupingComparatorClass(GroupingComparator.class);

job.setOutputKeyClass(CompositeKeyWritable.class);

job.setOutputValueClass(NullWritable.class);

job.setInputFormatClass(TextInputFormat.class);

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

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

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

System.out.println("sorting-driver");

}catch(IOException | ClassNotFoundException ex){

Logger.getLogger(Lab2\_Sorting.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

1. **TopTenCancerType**

/\*

\* 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 hw3\_q5;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

/\*\*

\*

\* @author rajani

\*/

public class HW3\_Q5 {

/\*\*

\* @param args the command line arguments

\*/

public static class Map1 extends Mapper<Object, Object, Object, Object>{

public void map(Object key, Object value, Context context){

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

if((!row[5].equals("Incidence")) && (!row[7].equals("All Cancer Sites Combined")) && (!row[4].equals("~"))){

String cancerType = row[7];

String patientMortality = row[4];

try {

DoubleWritable mortality = new DoubleWritable(Double.parseDouble(patientMortality));

context.write(new Text(cancerType), mortality);

} catch (Exception e) {

}

}

}

}

public static class Reduce1 extends Reducer<Text, DoubleWritable, Text, DoubleWritable>{

private DoubleWritable totalSales = new DoubleWritable();

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

int sum = 0;

int count = 0;

for(DoubleWritable val: values){

sum += val.get();

count++;

}

totalSales.set(sum/count);

context.write(key, totalSales);

}

}

public static class Map2

extends Mapper<LongWritable, Text, DoubleWritable, Text>{

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

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

Text employeeId = new Text(row[0]);

String salesAmount = row[1].trim();

try {

DoubleWritable count = new DoubleWritable(Double.parseDouble(salesAmount));

context.write(count, employeeId);

} catch (Exception e) {

}

}

}

// sort in descending , make rating as key and movie id as value

public static class Reduce2 extends Reducer<DoubleWritable, Text, Text, DoubleWritable>{

int i =0;

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

for(Text val : value){

if(i<10){

context.write(val, key);

i++;

}else{

return;

}

}

}

}

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

Configuration conf1 = new Configuration();

Job job1 = Job.getInstance(conf1, "chainingHw");

job1.setJarByClass(HW3\_Q5.class);

job1.setMapperClass(Map1.class);

//job1.setMapOutputKeyClass(Text.class);

//job1.setMapOutputValueClass(IntWritable.class);

job1.setReducerClass(Reduce1.class);

job1.setOutputKeyClass(Text.class);

job1.setOutputValueClass(DoubleWritable.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));

FileOutputFormat.setOutputPath(job1, new Path(args[1]));

boolean complete = job1.waitForCompletion(true);

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "chainingH");

if(complete){

job2.setJarByClass(HW3\_Q5.class);

job2.setMapperClass(Map2.class);

job2.setMapOutputKeyClass(DoubleWritable.class);

job2.setMapOutputValueClass(Text.class);

job2.setSortComparatorClass(SortComparator.class);

job2.setReducerClass(Reduce2.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(DoubleWritable.class);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[2]));

System.exit(job2.waitForCompletion(true) ? 0 : 1);

}

}

}

/\*

\* 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 hw3\_q5;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author rajani

\*/

public class SortComparator extends WritableComparator{

protected SortComparator() {

super(DoubleWritable.class, true);

}

@Override

public int compare(WritableComparable a, WritableComparable b) {

DoubleWritable key1 = (DoubleWritable) a;

DoubleWritable key2 = (DoubleWritable) b;

// sorting in descending order

int result = key1.get() < key2.get() ? 1 : key1.get() == key2.get() ? 0 : -1;

return result;

}

}