## horizontal line



Analysis on Gun Violence Data from 1980-2014

**Final Project**

**─**

Ashwin Lakshman

NUID: 001353233

**INFO7250 Engineering of Big Data**

Northeastern University

# Overview of Data Set

Dataset Used:<https://www.kaggle.com/murderaccountability/homicide-reports>

I have chosen to use this dataset because of its relevance to current society, as well as for its potential for further analyses and studies.

The dataset structure consists of the following columns/variables:

1. **Record ID**
2. **Agency Code**
3. **Agency Name**
4. **Agency Type**
5. **City**
6. **State**
7. **Year**
8. **Month**
9. **Incident**
10. **Crime Type**
11. **Crime Solved**
12. **Victim Sex**
13. **Victim Age**
14. **Victim Race**
15. **Victim Ethnicity**
16. **Perpetrator Sex**
17. **Perpetrator Age**
18. **Perpetrator Race**
19. **Perpetrator Ethnicity**
20. **Relationship**
21. **Weapon**
22. **Victim Count**
23. **Perpetrator Count**
24. **Record Source**

# Goals

1. To Perform various MapReduce tasks implementing different Techniques.
2. To perform multiple analyses on the dataset and derived data using PIG.
3. To load the data and perform multiple queries and aggregations of the dataset using HIVE.

# Source Code Repository

The entire source code, dataset, report and relevant documentation can be found at:

<https://github.com/ashwinlakshman1/BigDataFinalProject>

# Data Cleaning

The rows with erroneous or missing data were removed using a simple MapReduce task on Hadoop to obtain a fully usable Dataset.

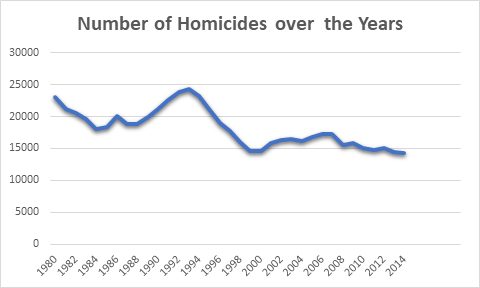
# Analysis of Homicide Data using MapReduce on Hadoop

## Simple MapReduce to Count Data

|  |
| --- |
| Total Count |
|  |

## Count Number of Homicides By Year

|  |
| --- |
| Homicides by Year |
|  |



## Count Number of Homicides By City

|  |
| --- |
| Homicide by city |
|  |

## Top 10: Estimating Safest City using Data from Step 3

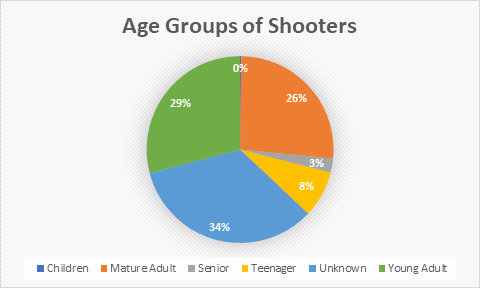
|  |
| --- |
| Safest city |
|  |

## Top 10: Estimating Most Violent City with Data from Step 3

|  |
| --- |
| City with Most Violence |
|  |

## Classification and counting of Perpetrators by Age Group.

|  |
| --- |
| Perps by Age Group: |
|  |



## Number of Shooting Events for each Weapon

|  |
| --- |
| Homicide events By Weapon |
|  |

## TOP 10: Most Frequently Used Weapons using Data From Step 7

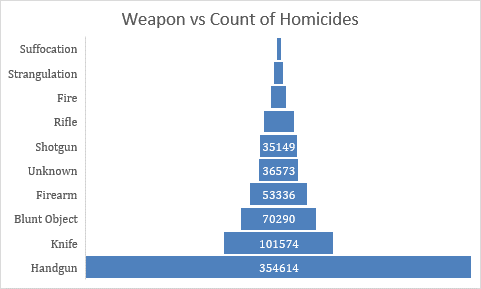
|  |
| --- |
| Top 10 Frequent Weapons |
|  |

## Number of Deaths per Weapon

|  |
| --- |
| Victims/Deaths by Weapon |
|  |

## Deadliest Weapons by Number of Deaths using Data from Step 9

|  |
| --- |
| Deadliest Weapon |
|  |



## INNER-JOIN: Joining Data from Step 7 and Step 9 to calculate Ratio of Deaths: Events of each weapon to calculate the “Deadliness” of the weapon.

|  |
| --- |
| INNER JOIN: Deadliness of a weapon ( Ratio of number of deaths per weapon vs number of times the weapon was used |
|  |

## INVERTED INDEX: Looking at Primary Target Victim Race for each Perpetrator Race and Counting the events

|  |
| --- |
| INVERTED INDEX: Perpetrator Race vs Victim Race |
|  |

## SUMMARIZATION PATTERNS: Calculating the Average Age of the Perpetrator in each State

|  |
| --- |
| SUMMARIZATION: State vs Average Age |
|  |

## SECONDARY SORT: Calculating the Cities in each State with most Homicide Events. Uses Input from Step 3.

|  |
| --- |
| SECONDARY SORTING: Each State with Its list of Cities in descending order of homicide events |
|  |

# 

# Analysis of Homicide Report Data using Apache PIG

## Top 20: Years with most number of Homicide Events

|  |
| --- |
| -- 1. Load data from teh dataset  RAW\_DATA = LOAD 'A\_homicide\_by\_year' USING PigStorage('\t') AS (year: int, deaths: int);  -- 2. Top 10 Years with most homicides  TOPYEAR = FOREACH RAW\_DATA GENERATE year AS y, deaths AS d;  -- 3. Grouping by year,  GROUP\_TOPYEAR = GROUP TOPYEAR BY (y,d);  -- 4. Aggregate, flatten  COUNT\_TOPYEAR = FOREACH GROUP\_TOPYEAR GENERATE FLATTEN(group), COUNT(TOPYEAR) as count;  -- 5.5 Aggregate over year  GROUP\_COUNT\_TOPYEAR = GROUP COUNT\_TOPYEAR BY y;  -- 6. UDF to compute top 10  topDeathYear = FOREACH GROUP\_COUNT\_TOPYEAR { result = TOP(20, 2, COUNT\_TOPYEAR);  GENERATE FLATTEN(result);  }  -- dump  STORE topDeathYear INTO 'PigOutput1' USING PigStorage(','); |

|  |
| --- |
|  |

## Number of Homicides by Race, Per Year

|  |
| --- |
| -- 1. Load data from the dataset  RAW\_DATA = LOAD 'cleaned' USING PigStorage(',') AS (state: chararray, city: chararray, year: int, solved: chararray, vicsex: chararray, vicage: int, vicrace: chararray, perpsex: chararray, perpage: int, perprace: chararray);  PERPRACE\_DATA = FOREACH RAW\_DATA GENERATE year AS y, perprace AS p;  GROUP\_PERPRACE = GROUP PERPRACE\_DATA BY (y,p);  COUNT\_PERPRACE = FOREACH GROUP\_PERPRACE GENERATE FLATTEN(group), (COUNT(PERPRACE\_DATA)) AS freq;  STORE COUNT\_PERPRACE INTO 'PerpRaceCount' USING PigStorage(','); |

|  |
| --- |
|  |

## Proportion of Solved : Total Cases by Year and State

|  |
| --- |
| -- 1. Load data from the dataset  RAW\_DATA = LOAD 'cleaned' USING PigStorage(',') AS (state: chararray, city: chararray, year: int, solved: chararray, vicsex: chararray, vicage: int, vicrace: chararray, perpsex: chararray, perpage: int, perprace: chararray);  A = FOREACH RAW\_DATA GENERATE year AS y, state AS st, solved AS s;  B = GROUP A BY (y,st);  COUNT\_TOTAL = FOREACH B { C = FILTER A BY (s=='Yes');  GENERATE group, COUNT(A) AS tot, COUNT(C) AS del, (float) COUNT(C)/COUNT(A) AS frac;}  dump COUNT\_TOTAL;  STORE COUNT\_TOTAL INTO 'SolvedProportion' USING PigStorage(','); |

|  |
| --- |
|  |

## Count of each combination of Perpetrator Race: Victim Race

|  |
| --- |
| -- 1. Load data from the dataset  RAW\_DATA = LOAD 'cleaned' USING PigStorage(',') AS (state: chararray, city: chararray, year: int, solved: chararray, vicsex: chararray, vicage: int, vicrace: chararray, perpsex: chararray, perpage: int, perprace: chararray);  A = FOREACH RAW\_DATA GENERATE perprace as p, vicrace as v;  B = GROUP A by (p,v);  COUNTA = FOREACH B GENERATE group, COUNT(A);  STORE COUNTA INTO 'PerpVicRaceCount' USING PigStorage(','); |

|  |
| --- |
|  |

# Appendix

|  |
| --- |
| Data Cleaning |
| package com.ashwin.cleaner;  import com.ashwin.event\_count.\*;  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.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;  import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(CountMapper.class);  job.setReducerClass(CountReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(Text.class);  job.setMapOutputKeyClass(NullWritable.class);  job.setMapOutputValueClass(Text.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.cleaner;  import com.ashwin.event\_count.\*;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class CountMapper extends Mapper<LongWritable, Text, NullWritable, Text> {  Text word = new Text();  IntWritable one = new IntWritable(1);  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String city, state, year, solved, vicsex, vicage, vicrace, perpsex, perpage, perprace,newLine;  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  state = tokens[5];  city = tokens[4];  year = tokens[6];  solved = tokens[10];  vicsex = tokens[11];  vicage = tokens[12];  vicrace = tokens[13];  perpsex = tokens[15];  perpage = tokens[16];  perprace = tokens[17];  if(state.equals(" ")||city.equals(" ")||year.equals(" ")||solved.equals(" ")||vicage.equals(" ")||vicrace.equals(" ")||vicsex.equals(" ")||perpage.equals(" ")||perprace.equals(" ")||perpsex.equals(" ")){  return;  }  newLine=state+","+city+","+year+","+solved+","+vicsex+","+vicage+","+vicrace+","+perpsex+","+perpage+","+perprace;  context.write(NullWritable.get(), new Text(newLine));  }  }  package com.ashwin.cleaner;  import com.ashwin.event\_count.\*;  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;  public class CountReducer extends Reducer<NullWritable, Text, NullWritable, Text> {  @Override  protected void reduce(NullWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (Text val : values) {  context.write(key, val);  }  }  } |

|  |
| --- |
| * + 1. EVENT COUNT |
| package com.ashwin.event\_count;  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.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;  import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  /\*\*  \* Hello world!  \*  \*/  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(CountMapper.class);  job.setReducerClass(CountReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setMapOutputKeyClass(NullWritable.class);  job.setMapOutputValueClass(IntWritable.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.event\_count;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class CountMapper extends Mapper<LongWritable, Text, NullWritable, IntWritable> {  Text word = new Text();  IntWritable one = new IntWritable(1);  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  context.write(NullWritable.get(), one);  }  }  package com.ashwin.event\_count;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.mapreduce.Reducer;  public class CountReducer extends Reducer<NullWritable, IntWritable, NullWritable, IntWritable> {  @Override  protected void reduce(NullWritable key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  IntWritable count = new IntWritable(sum);  context.write(key, count);  }  } |

|  |
| --- |
| * + 1. HOMICIDE BY YEAR |
| package com.ashwin.homicide\_by\_city;  import com.ashwin.homicide\_by\_year.\*;  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.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.mapreduce.lib.output.TextOutputFormat;  /\*\*  \* Hello world!  \*  \*/  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(ByCityMapper.class);  job.setReducerClass(ByCityReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(IntWritable.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.homicide\_by\_city;  import com.ashwin.homicide\_by\_year.\*;  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 ByCityMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  Text word = new Text();  IntWritable one = new IntWritable(1);  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  String City = tokens[4]+", "+ tokens[5];  word.set(City);  context.write(word, one);  }  }  package com.ashwin.homicide\_by\_city;  import com.ashwin.homicide\_by\_year.\*;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class ByCityReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  IntWritable count = new IntWritable(sum);  context.write(key, count);  }  } |

|  |
| --- |
| * + 1. HOMICIDE BY CITY |
| package com.ashwin.homicide\_by\_city;  import com.ashwin.homicide\_by\_year.\*;  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.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.mapreduce.lib.output.TextOutputFormat;  /\*\*  \* Hello world!  \*  \*/  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(ByCityMapper.class);  job.setReducerClass(ByCityReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(IntWritable.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.homicide\_by\_city;  import com.ashwin.homicide\_by\_year.\*;  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 ByCityMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  Text word = new Text();  IntWritable one = new IntWritable(1);  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  String City = tokens[4]+", "+ tokens[5];  word.set(City);  context.write(word, one);  }  }  package com.ashwin.homicide\_by\_city;  import com.ashwin.homicide\_by\_year.\*;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class ByCityReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  IntWritable count = new IntWritable(sum);  context.write(key, count);  }  } |

|  |
| --- |
| * + 1. SAFEST CITY |
| package com.ashwin.safest\_city;  import com.ashwin.violent\_city.\*;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  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;  import org.apache.hadoop.util.GenericOptionsParser;  public class App {  public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "top 10");  job.setJarByClass(App.class);  job.setMapperClass(SafestCityMapper.class);  job.setReducerClass(SafestCityReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(LongWritable.class);  job.setOutputKeyClass(LongWritable.class);  job.setOutputValueClass(Text.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  FileSystem fs = FileSystem.get(conf);  fs.delete(new Path(args[1]),true);    System.exit(job.waitForCompletion(true) ? 0 : 1);  }  }  package com.ashwin.safest\_city;  import com.ashwin.violent\_city.\*;  import java.io.\*;  import java.util.\*;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.mapreduce.Mapper;  public class SafestCityMapper extends Mapper<Object, Text, Text, LongWritable> {  private TreeMap<Long, String> tmap;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap = new TreeMap<Long, String>(Collections.reverseOrder());  }  @Override  public void map(Object key, Text value,  Context context) throws IOException,  InterruptedException {  String[] tokens = value.toString().split("\t");  String city = tokens[0];  long no\_of\_homicides = Long.parseLong(tokens[1]);  tmap.put(no\_of\_homicides, city);  if (tmap.size() > 10) {  tmap.remove(tmap.firstKey());  }  }  //Called once at the end of the task  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap.entrySet()) {  long count = entry.getKey();  String city\_name = entry.getValue();  context.write(new Text(city\_name), new LongWritable(count));  }  }  }  package com.ashwin.safest\_city;  import com.ashwin.violent\_city.\*;  import java.io.IOException;  import java.util.Map;  import java.util.TreeMap;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class SafestCityReducer extends Reducer<Text, LongWritable, LongWritable, Text> {  private TreeMap<Long, String> tmap2;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap2 = new TreeMap<Long, String>();  }  @Override  public void reduce(Text key, Iterable<LongWritable> values,  Context context) throws IOException, InterruptedException {  String name = key.toString();  long count = 0;  for (LongWritable val : values) {  count = val.get();  }  tmap2.put(count, name);  if (tmap2.size() > 10) {  tmap2.remove(tmap2.firstKey());  }  }  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap2.entrySet()) {  long count = entry.getKey();  String city\_name = entry.getValue();  context.write(new LongWritable(count), new Text(city\_name));  }  }  } |

|  |
| --- |
| * + 1. MOST VIOLENT CITY |
| package com.ashwin.violent\_city;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  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;  import org.apache.hadoop.util.GenericOptionsParser;  public class App {  public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "top 10");  job.setJarByClass(App.class);  job.setMapperClass(ViolentCityMapper.class);  job.setReducerClass(ViolentCityReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(LongWritable.class);  job.setOutputKeyClass(LongWritable.class);  job.setOutputValueClass(Text.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  FileSystem fs = FileSystem.get(conf);  fs.delete(new Path(args[1]),true);    System.exit(job.waitForCompletion(true) ? 0 : 1);  }  }  package com.ashwin.violent\_city;  import java.io.\*;  import java.util.\*;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.mapreduce.Mapper;  public class ViolentCityMapper extends Mapper<Object, Text, Text, LongWritable> {  private TreeMap<Long, String> tmap;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap = new TreeMap<Long, String>();  }  @Override  public void map(Object key, Text value,  Context context) throws IOException,  InterruptedException {  String[] tokens = value.toString().split("\t");  String city = tokens[0];  long no\_of\_homicides = Long.parseLong(tokens[1]);  tmap.put(no\_of\_homicides, city);  if (tmap.size() > 10) {  tmap.remove(tmap.firstKey());  }  }  //Called once at the end of the task  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap.entrySet()) {  long count = entry.getKey();  String city\_name = entry.getValue();  context.write(new Text(city\_name), new LongWritable(count));  }  }  }  package com.ashwin.violent\_city;  import java.io.IOException;  import java.util.Map;  import java.util.TreeMap;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class ViolentCityReducer extends Reducer<Text, LongWritable, LongWritable, Text> {  private TreeMap<Long, String> tmap2;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap2 = new TreeMap<Long, String>();  }  @Override  public void reduce(Text key, Iterable<LongWritable> values,  Context context) throws IOException, InterruptedException {  String name = key.toString();  long count = 0;  for (LongWritable val : values) {  count = val.get();  }  tmap2.put(count, name);  if (tmap2.size() > 10) {  tmap2.remove(tmap2.firstKey());  }  }  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap2.entrySet()) {  long count = entry.getKey();  String city\_name = entry.getValue();  context.write(new LongWritable(count), new Text(city\_name));  }  }  } |

|  |
| --- |
| * + 1. HOMICIDE BY WEAPON |
| package com.ashwin.homicide\_by\_weapon;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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.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.mapreduce.lib.output.TextOutputFormat;  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(ByWeaponMapper.class);  job.setReducerClass(ByWeaponReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(IntWritable.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.homicide\_by\_weapon;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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 ByWeaponMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  Text word = new Text();  IntWritable one = new IntWritable(1);  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  String Weapon = tokens[20];  word.set(Weapon);  context.write(word, one);  }  }  package com.ashwin.homicide\_by\_weapon;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class ByWeaponReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  IntWritable count = new IntWritable(sum);  context.write(key, count);  }  } |

|  |
| --- |
| * + 1. MOST FREQUENT WEAPON |
| package com.ashwin.frequent\_weapons;  import com.ashwin.violent\_city.\*;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  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;  import org.apache.hadoop.util.GenericOptionsParser;  public class App {  public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "top 10");  job.setJarByClass(App.class);  job.setMapperClass(FreqWeaponMapper.class);  job.setReducerClass(FreqWeaponReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(LongWritable.class);  job.setOutputKeyClass(LongWritable.class);  job.setOutputValueClass(Text.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  FileSystem fs = FileSystem.get(conf);  fs.delete(new Path(args[1]),true);    System.exit(job.waitForCompletion(true) ? 0 : 1);  }  }  package com.ashwin.frequent\_weapons;  import com.ashwin.violent\_city.\*;  import java.io.\*;  import java.util.\*;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.mapreduce.Mapper;  public class FreqWeaponMapper extends Mapper<Object, Text, Text, LongWritable> {  private TreeMap<Long, String> tmap;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap = new TreeMap<Long, String>();  }  @Override  public void map(Object key, Text value,  Context context) throws IOException,  InterruptedException {  String[] tokens = value.toString().split("\t");  String weapon = tokens[0];  long no\_of\_homicides = Long.parseLong(tokens[1]);  tmap.put(no\_of\_homicides, weapon);  if (tmap.size() > 10) {  tmap.remove(tmap.firstKey());  }  }  //Called once at the end of the task  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap.entrySet()) {  long count = entry.getKey();  String weapon\_name = entry.getValue();  context.write(new Text(weapon\_name), new LongWritable(count));  }  }  }  package com.ashwin.frequent\_weapons;  import com.ashwin.violent\_city.\*;  import java.io.IOException;  import java.util.Map;  import java.util.TreeMap;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class FreqWeaponReducer extends Reducer<Text, LongWritable, LongWritable, Text> {  private TreeMap<Long, String> tmap2;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap2 = new TreeMap<Long, String>();  }  @Override  public void reduce(Text key, Iterable<LongWritable> values,  Context context) throws IOException, InterruptedException {  String name = key.toString();  long count = 0;  for (LongWritable val : values) {  count = val.get();  }  tmap2.put(count, name);  if (tmap2.size() > 10) {  tmap2.remove(tmap2.firstKey());  }  }  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap2.entrySet()) {  long count = entry.getKey();  String weapon\_name = entry.getValue();  context.write(new LongWritable(count), new Text(weapon\_name));  }  }  } |

|  |
| --- |
| * + 1. NUMBER OF DEATHS PER WEAPON |
| package com.ashwin.victim\_by\_weapon;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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.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.mapreduce.lib.output.TextOutputFormat;  /\*\*  \* Hello world!  \*  \*/  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(ByWeaponMapper.class);  job.setReducerClass(ByWeaponReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(IntWritable.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.victim\_by\_weapon;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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 ByWeaponMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  Text word = new Text();  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  if(tokens[21].length()<4){  IntWritable victim = new IntWritable(Integer.parseInt(tokens[21])+1);  String Weapon = tokens[20];  word.set(Weapon);  context.write(word, victim);  }  }  }  package com.ashwin.victim\_by\_weapon;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class ByWeaponReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  IntWritable count = new IntWritable(sum);  context.write(key, count);  }  } |

|  |
| --- |
| * + 1. DEADLIEST WEAPON |
| package com.ashwin.deadliest\_weapons;  import com.ashwin.frequent\_weapons.\*;  import com.ashwin.violent\_city.\*;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  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;  import org.apache.hadoop.util.GenericOptionsParser;  public class App {  public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "top 10");  job.setJarByClass(App.class);  job.setMapperClass(DeadliestWeaponMapper.class);  job.setReducerClass(DeadliestWeaponReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(LongWritable.class);  job.setOutputKeyClass(LongWritable.class);  job.setOutputValueClass(Text.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  FileSystem fs = FileSystem.get(conf);  fs.delete(new Path(args[1]),true);    System.exit(job.waitForCompletion(true) ? 0 : 1);  }  }  package com.ashwin.deadliest\_weapons;  import com.ashwin.frequent\_weapons.\*;  import com.ashwin.violent\_city.\*;  import java.io.\*;  import java.util.\*;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.mapreduce.Mapper;  public class DeadliestWeaponMapper extends Mapper<Object, Text, Text, LongWritable> {  private TreeMap<Long, String> tmap;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap = new TreeMap<Long, String>();  }  @Override  public void map(Object key, Text value,  Context context) throws IOException,  InterruptedException {  String[] tokens = value.toString().split("\t");  String weapon = tokens[0];  long no\_of\_homicides = Long.parseLong(tokens[1]);  tmap.put(no\_of\_homicides, weapon);  if (tmap.size() > 10) {  tmap.remove(tmap.firstKey());  }  }  //Called once at the end of the task  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap.entrySet()) {  long count = entry.getKey();  String weapon\_name = entry.getValue();  context.write(new Text(weapon\_name), new LongWritable(count));  }  }  }  package com.ashwin.deadliest\_weapons;  import com.ashwin.frequent\_weapons.\*;  import com.ashwin.violent\_city.\*;  import java.io.IOException;  import java.util.Map;  import java.util.TreeMap;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class DeadliestWeaponReducer extends Reducer<Text, LongWritable, LongWritable, Text> {  private TreeMap<Long, String> tmap2;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap2 = new TreeMap<Long, String>();  }  @Override  public void reduce(Text key, Iterable<LongWritable> values,  Context context) throws IOException, InterruptedException {  String name = key.toString();  long count = 0;  for (LongWritable val : values) {  count = val.get();  }  tmap2.put(count, name);  if (tmap2.size() > 10) {  tmap2.remove(tmap2.firstKey());  }  }  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<Long, String> entry : tmap2.entrySet()) {  long count = entry.getKey();  String weapon\_name = entry.getValue();  context.write(new LongWritable(count), new Text(weapon\_name));  }  }  } |

|  |
| --- |
| * + 1. TOP DEADLIEST WEAPONS |
| package com.ashwin.top\_deadliest;  import com.ashwin.frequent\_weapons.\*;  import com.ashwin.violent\_city.\*;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  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;  import org.apache.hadoop.util.GenericOptionsParser;  public class App {  public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "top 10");  job.setJarByClass(App.class);  job.setMapperClass(TopDeadliestMapper.class);  job.setReducerClass(TopDeadliestReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(Text.class);  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(Text.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  FileSystem fs = FileSystem.get(conf);  fs.delete(new Path(args[1]),true);    System.exit(job.waitForCompletion(true) ? 0 : 1);  }  }  package com.ashwin.top\_deadliest;  import java.io.\*;  import java.util.\*;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class TopDeadliestMapper extends Mapper<Object, Text, Text, Text> {  private TreeMap<String, String> tmap;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap = new TreeMap<String, String>();  }  @Override  public void map(Object key, Text value,  Context context) throws IOException,  InterruptedException {  String[] tokens = value.toString().split("\t");  String ratio = tokens[0];  String details = tokens[1];  tmap.put(ratio, details);  if (tmap.size() > 10) {  tmap.remove(tmap.firstKey());  }  }  //Called once at the end of the task  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<String, String> entry : tmap.entrySet()) {  String rat = entry.getKey();  String w\_details = entry.getValue();  context.write(new Text(rat), new Text(w\_details));  }  }  }  package com.ashwin.top\_deadliest;  import com.ashwin.frequent\_weapons.\*;  import com.ashwin.violent\_city.\*;  import java.io.IOException;  import java.util.Map;  import java.util.TreeMap;  import org.apache.hadoop.io.FloatWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class TopDeadliestReducer extends Reducer<Text, Text, FloatWritable, Text> {  private TreeMap<String, String> tmap2;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  tmap2 = new TreeMap<String, String>();  }  @Override  public void reduce(Text key, Iterable<Text> values,  Context context) throws IOException, InterruptedException {  String ratio = key.toString();  String details = "";  for (Text val : values) {  ratio = val.toString();  }  tmap2.put(ratio, details);  if (tmap2.size() > 10) {  tmap2.remove(tmap2.firstKey());  }  }  @Override  public void cleanup(Context context) throws IOException,  InterruptedException {  for (Map.Entry<String, String> entry : tmap2.entrySet()) {  String ratiox = entry.getKey();  String detailsx = entry.getValue();  context.write(new FloatWritable(Float.parseFloat(ratiox)), new Text(detailsx));  }  }  } |

|  |
| --- |
| * + 1. PERPETRATOR BY AGE GROUP |
| package com.ashwin.perp\_age\_group;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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.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.mapreduce.lib.output.TextOutputFormat;  /\*\*  \* Hello world!  \*  \*/  public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(PerpAgeGroupMapper.class);  job.setReducerClass(PerpAgeGroupReducer.class);    //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(IntWritable.class);    //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));    //Set the number of reducers  // job.setNumReduceTasks(2);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    System.out.println( "SUCCESS" );  }  }  package com.ashwin.perp\_age\_group;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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 PerpAgeGroupMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  Text word = new Text();  IntWritable one = new IntWritable(1);  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  // word.set(tokens[16]);  if (tokens[16].length() < 4 && !tokens[16].equals(" ")) {  int age = 0;    age = Integer.parseInt(tokens[16]);  String group = "";  if (age > 0 && age <= 12) {  group = "Children";  } else if (age > 12 && age <= 18) {  group = "Teenager";  } else if (age > 18 && age < 30) {  group = "Young Adult";  } else if (age >= 30 && age < 60) {  group = "Mature Adult";  } else if (age >= 60) {  group = "Senior";  }  else group="Unknown";  word.set(group);  } else {  word.set("Unknown");  }  context.write(word, one);  }  }  package com.ashwin.perp\_age\_group;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class PerpAgeGroupReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  IntWritable count = new IntWritable(sum);  context.write(key, count);  }  } |

|  |
| --- |
| * + 1. INNER JOIN: RATIO OF DEATHS TO SHOOTING EVENTS FOR EACH GUN TYPE |
| package com.ashwin.inner\_join\_weapon\_deadliness;  import com.ashwin.deadliest\_weapons.\*;  import com.ashwin.frequent\_weapons.\*;  import com.ashwin.violent\_city.\*;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  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.input.MultipleInputs;  import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  import org.apache.hadoop.util.GenericOptionsParser;  public class App {  public static void main(String[] args) throws Exception  {  Configuration conf = new Configuration();  Job job = new Job(conf, "Inner Join-Weapon Deadliness");  job.setJarByClass(App.class);  job.setMapperClass(FrequentMapper.class);  job.setReducerClass(DeadlinessReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(Text.class);  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(Text.class);  MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class, DeadliestMapper.class);  MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class, FrequentMapper.class);  Path outputPath = new Path(args[2]);  FileOutputFormat.setOutputPath(job, outputPath);  System.exit(job.waitForCompletion(true) ? 0 : 1);    }  }  package com.ashwin.inner\_join\_weapon\_deadliness;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  import java.io.IOException;  public class DeadliestMapper extends Mapper<LongWritable, Text, Text, Text> {  // hadoop datatype  Text word = new Text();  IntWritable one = new IntWritable(1);  @Override  protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens= line.split("\t");  String newKey = tokens[0];  word.set(newKey);  String outValue= "A"+tokens[1];  context.write(word,new Text(outValue));  }  }  package com.ashwin.inner\_join\_weapon\_deadliness;  import java.io.IOException;  import java.util.ArrayList;  import java.util.Iterator;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class DeadlinessReducer 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;  @Override  public void setup(Context context) throws IOException,  InterruptedException {  joinType = "inner";  }  @Override  public void reduce(Text key, Iterable<Text> values,  Context context) throws IOException, InterruptedException {  listA.clear();  listB.clear();  Iterator<Text> itr = values.iterator();  while (itr.hasNext()) {  tmp = itr.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)));  }  }  //JOIN LOGIC  for (Text textA : listA) {  for (Text textB : listB) {  float dd = Float.parseFloat(textA.toString()) / Float.parseFloat(textB.toString());  String x = "Weapon : " + key.toString() + " Deaths : " + textA.toString() + " Incidents : " + textB.toString();  Text t1 = new Text(x);  String dds = dd + " ";  Text ddscore = new Text(dds);  context.write(ddscore, t1);  }  }  }  }  package com.ashwin.inner\_join\_weapon\_deadliness;  import java.io.\*;  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 FrequentMapper extends Mapper<LongWritable, Text, Text, Text> {  Text word = new Text();  IntWritable one = new IntWritable(1);    @Override  public void map(LongWritable key, Text value,  Context context) throws IOException,  InterruptedException {  String line = value.toString();  String[] tokens= line.split("\t");  String newKey = tokens[0];  word.set(newKey);  String outValue= "B"+tokens[1];  context.write(word,new Text(outValue));  }  } |

|  |
| --- |
| * + 1. INVERTED INDEX: PERPETRATOR RACE VS VICTIM RACE COMPARISON |
| package com.ashwin.inverted\_index\_race\_comparison;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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.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.mapreduce.lib.output.TextOutputFormat;  /\*\*  \* Hello world!  \*  \*/  public class App {  public static void main(String[] args) {  Configuration conf = new Configuration();  try { //create a new job  Job job = Job.getInstance(conf, "logcount");  job.setJarByClass(App.class);  //Set Mappers and Reducer classes  job.setMapperClass(RaceCompMapper.class);  job.setReducerClass(RaceCompReducer.class);  //Set InputFormat and OutputFormat class OK  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);  //Set the output key and value types OK  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(Text.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(Text.class);  //Set the input and output paths OK  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  //Set the number of reducers  // job.setNumReduceTasks(2);  try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  System.out.println("SUCCESS");  }  }  package com.ashwin.inverted\_index\_race\_comparison;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  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 RaceCompMapper extends Mapper<LongWritable, Text, Text, Text> {  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String perp\_race;  String vic\_race;  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  if(tokens[18].equals("Hispanic")){  perp\_race=tokens[18];  }  else perp\_race=tokens[17];    if(tokens[14].equals("Hispanic")){  vic\_race=tokens[14];  }  else vic\_race=tokens[13];  context.write(new Text(perp\_race+":"), new Text(vic\_race));  }  }  package com.ashwin.inverted\_index\_race\_comparison;  import com.ashwin.homicide\_by\_weapon.\*;  import com.ashwin.homicide\_by\_city.\*;  import com.ashwin.homicide\_by\_year.\*;  import java.io.IOException;  import java.util.HashSet;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class RaceCompReducer extends Reducer<Text, Text, Text, Text> {  HashSet<String> hs = new HashSet<String>();  @Override  protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  hs.clear();  int sum = 0;  // for (IntWritable val : values) {  // sum += val.get();  // }  StringBuffer sb = new StringBuffer("");  for(Text v: values){  hs.add(v.toString());  sum+=1;  }  for(String v: hs){  sb.append(v);  sb.append(" ");  }  context.write(key, new Text(sb.toString()+ " events:"+ Integer.toString(sum)));  }  } |

|  |
| --- |
| * + 1. SUMMARIZATION: AVERAGE AGE OF PERPETRATOR BY STATE |
| package com.ashwin.summarization\_state\_to\_avgage;  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.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.mapreduce.lib.output.TextOutputFormat;    public class App  {  public static void main( String[] args )  {  Configuration conf = new Configuration();  try {  Job job = Job.getInstance(conf, "nysestock");  job.setJarByClass(App.class);    //Set Mappers and Reducer classes  job.setMapperClass(StockMapper.class);  job.setReducerClass(StockReducer.class);  job.setCombinerClass(StockCombiner.class);    TextInputFormat.addInputPath(job, new Path(args[0]));  //Set InputFormat and OutputFormat class  job.setInputFormatClass(TextInputFormat.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the output key and value types  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(StockTuple.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(StockTuple.class);  job.setOutputFormatClass(TextOutputFormat.class);    //Set the input and output paths  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));      //Set the number of reducers  job.setNumReduceTasks(1);    try {  System.exit(job.waitForCompletion(true) ? 0 : 1);  } catch (ClassNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    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 com.ashwin.summarization\_state\_to\_avgage;  import org.apache.hadoop.io.Text;  import java.io.IOException;  import org.apache.hadoop.mapreduce.Reducer;  /\*\*  \*  \* @author ashwi  \*/  public class StockCombiner extends Reducer<Text, StockTuple, Text, StockTuple>{    StockTuple stTuple= new StockTuple();    @Override  protected void reduce(Text key, Iterable<StockTuple> values, Context context)throws IOException,InterruptedException  {  long count=0;  long sum\_price=0;  for(StockTuple st:values){  count+=st.getCount();  sum\_price+=st.getAge();    }  stTuple.setCount(count);  stTuple.setAge(sum\_price);  context.write(key, stTuple);  }  }  package com.ashwin.summarization\_state\_to\_avgage;  import java.io.IOException;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.logging.Level;  import java.util.logging.Logger;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class StockMapper extends Mapper<Object, Text, Text, StockTuple> {  private StockTuple stTuple = new StockTuple();  //private final static SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");  @Override  protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {  String line = value.toString();  String[] tokens = line.split(",");  if (tokens[0].equals("Record ID")) {  return;  }  String city = tokens[5];  String age = tokens[16];  int intage = 28;  if (age.length() < 3 && !age.equals(" ")) {  intage = Integer.parseInt(age);  if (intage == 0) {  intage = 28;  }  }    stTuple.setCount(1);  stTuple.setAge(intage);  context.write(new Text(city), stTuple);  }  }  package com.ashwin.summarization\_state\_to\_avgage;  import java.io.IOException;  import java.util.Date;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  public class StockReducer extends Reducer<Text,StockTuple,Text,StockTuple>{  private StockTuple stTuple = new StockTuple();  protected void reduce(Text key, Iterable<StockTuple> values,Context context) throws IOException, InterruptedException {  // TODO Auto-generated method stub  long count=0;  long total\_age=0;  for(StockTuple st:values)  {  count+=st.getCount();  total\_age=st.getAge();  }  stTuple.setAge(total\_age/count);  stTuple.setCount(count);  context.write(key, stTuple);  }  }  /\*  \* 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 com.ashwin.summarization\_state\_to\_avgage;  import java.io.DataInput;  import java.io.DataOutput;  import java.io.EOFException;  import java.io.IOException;  import java.text.SimpleDateFormat;  import java.util.Date;  import org.apache.hadoop.io.Writable;  /\*\*  \*  \* @author ashwi  \*/  public class StockTuple implements Writable  {  public long getAge() {  return Age;  }  public void setAge(long Age) {  this.Age = Age;  }    public long getCount() {  return count;  }  public void setCount(long count) {  this.count = count;  }  private long Age;  private long count;    @Override  public void readFields(DataInput in) throws IOException {  Age = in.readLong();  count = in.readLong();  }  @Override  public void write(DataOutput out) throws IOException {  out.writeLong(Age);  out.writeLong(count);  }  @Override  public String toString() {  return " average Age: "+Age +" Count: "+count;  }  } |

|  |
| --- |
| * + 1. SECONDARY SORT: |
| /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  /\*\*  \*  \* @author ashwi  \*/  public class App {    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException  {  Job job = Job.getInstance();    job.setJarByClass(App.class);    job.setGroupingComparatorClass(GroupComparator.class);  job.setSortComparatorClass(SecondarySortComparator.class);  job.setPartitionerClass(KeyPartition.class);    FileInputFormat.addInputPath(job, new Path(args[0]));  Path outDir = new Path(args[1]);  FileOutputFormat.setOutputPath(job, outDir);    job.setMapperClass(SecondarySortMapper.class);  job.setReducerClass(SecondarySortReducer.class);    job.setNumReduceTasks(1);    job.setOutputKeyClass(CompositeKey.class);  job.setOutputValueClass(Text.class);    FileSystem fs = FileSystem.get(job.getConfiguration());  if(fs.exists(outDir)) {  fs.delete(outDir, true);  }    job.waitForCompletion(true);  }  }  /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  import org.apache.hadoop.io.WritableComparable;  import java.io.DataInput;  import java.io.DataOutput;  import java.io.IOException;  /\*\*  \*  \* @author ashwi  \*/  public class CompositeKey implements WritableComparable<CompositeKey>{  private String state;  private String deaths;  public CompositeKey() {  super();  }  public String getState() {  return state;  }  public void setState(String state) {  this.state = state;  }  public String getDeaths() {  return deaths;  }  public void setDeaths(String deaths) {  this.deaths = deaths;  }  public CompositeKey(String state, String deaths) {  this.state = state;  this.deaths = deaths;  }  @Override  public void write(DataOutput d) throws IOException {  d.writeUTF(state);  d.writeUTF(deaths);  }  @Override  public void readFields(DataInput di) throws IOException {  state = di.readUTF();  deaths = di.readUTF();  }  @Override  public int compareTo(CompositeKey o) {  int result = this.state.compareTo(o.getState());  if(result==0){  String c1 = this.deaths;  Integer n1 = Integer.valueOf(c1);;  String c2 = o.getDeaths();  Integer n2 = Integer.valueOf(c2);  return n1.compareTo(n2);  }    return result;  }  @Override  public String toString() {  return state + " , Homicides: " + deaths;  }  }  /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  /\*\*  \*  \* @author ashwi  \*/  import org.apache.hadoop.io.WritableComparable;  import org.apache.hadoop.io.WritableComparator;  public class GroupComparator extends WritableComparator{  protected GroupComparator() {  super(CompositeKey.class, true);  }  @SuppressWarnings("rawtypes")  @Override  public int compare(WritableComparable a, WritableComparable b) {    CompositeKey ckw1 = (CompositeKey)a;  CompositeKey ckw2 = (CompositeKey)b;    return ckw1.getState().compareTo(ckw2.getState());  }  }    /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  /\*\*  \*  \* @author ashwi  \*/  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Partitioner;  public class KeyPartition extends Partitioner<CompositeKey, Text>{  @Override  public int getPartition(CompositeKey key, Text value, int numPartitions) {  int hash=key.getState().hashCode();  int part=hash%numPartitions;  return part;    }  }  /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  /\*\*  \*  \* @author ashwi  \*/  import org.apache.hadoop.io.WritableComparable;  import org.apache.hadoop.io.WritableComparator;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.logging.Level;  import java.util.logging.Logger;  public class SecondarySortComparator extends WritableComparator {      protected SecondarySortComparator() {  super(CompositeKey.class, true);  }  @Override  public int compare(WritableComparable a, WritableComparable b) {      CompositeKey ck1 = (CompositeKey) a;  CompositeKey ck2 = (CompositeKey) b;  int result = ck1.getState().compareTo(ck2.getState());  if (result == 0) {  String c1 = ck1.getDeaths();  Integer n1 = Integer.valueOf(c1);  String c2 = ck2.getDeaths();  Integer n2 = Integer.valueOf(c2);  result = -1\*n1.compareTo(n2);    }  return result;  }      }  /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  /\*\*  \*  \* @author ashwi  \*/  public class SecondarySortMapper extends Mapper<LongWritable, Text, CompositeKey, Text>{  IntWritable one = new IntWritable(1);    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  //To change body of generated methods, choose Tools | Templates.  String line = value.toString();  String[] tokens = line.split(",");  String city = tokens[0];  String []details= tokens[1].split("\t");  String state= details[0];  String deaths=details[1];  CompositeKey coKey = new CompositeKey(state, deaths);  context.write(coKey, new Text(city));  }    }      /\*  \* 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 com.ashwin.sec\_sort\_homicidecity;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.mapreduce.Reducer;  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  /\*\*  \*  \* @author ashwi  \*/  public class SecondarySortReducer extends Reducer<CompositeKey, Text, CompositeKey, Text> {  @Override  protected void reduce(CompositeKey key, Iterable<Text> values, Context context) throws IOException, InterruptedException {  //To change body of generated methods, choose Tools | Templates.  int sum = 0;  for (Text v : values) {    context.write(key, v);  }  }  } |

|  |
| --- |
| * + 1. TOP YEARS WITH MOST HOMICIDES |
| -- 1. Load data from teh dataset  RAW\_DATA = LOAD 'A\_homicide\_by\_year' USING PigStorage('\t') AS (year: int, deaths: int);  -- 2. Top 10 Years with most homicides  TOPYEAR = FOREACH RAW\_DATA GENERATE year AS y, deaths AS d;  -- 3. Grouping by year,  GROUP\_TOPYEAR = GROUP TOPYEAR BY (y,d);  -- 4. Aggregate, flatten  COUNT\_TOPYEAR = FOREACH GROUP\_TOPYEAR GENERATE FLATTEN(group), COUNT(TOPYEAR) as count;  -- 5.5 Aggregate over year  GROUP\_COUNT\_TOPYEAR = GROUP COUNT\_TOPYEAR BY y;  -- 6. UDF to compute top 10  topDeathYear = FOREACH GROUP\_COUNT\_TOPYEAR { result = TOP(20, 2, COUNT\_TOPYEAR);  GENERATE FLATTEN(result);  }  -- dump  STORE topDeathYear INTO 'PigOutput1' USING PigStorage(','); |

|  |
| --- |
| * + 1. NUMBER OF HOMICIDES BY RACE PER YEAR |
| -- 1. Load data from the dataset  RAW\_DATA = LOAD 'cleaned' USING PigStorage(',') AS (state: chararray, city: chararray, year: int, solved: chararray, vicsex: chararray, vicage: int, vicrace: chararray, perpsex: chararray, perpage: int, perprace: chararray);  PERPRACE\_DATA = FOREACH RAW\_DATA GENERATE year AS y, perprace AS p;  GROUP\_PERPRACE = GROUP PERPRACE\_DATA BY (y,p);  COUNT\_PERPRACE = FOREACH GROUP\_PERPRACE GENERATE FLATTEN(group), (COUNT(PERPRACE\_DATA)) AS freq;  STORE COUNT\_PERPRACE INTO 'PerpRaceCount' USING PigStorage(','); |

|  |
| --- |
| * + 1. PROPORTION OF SOLVED TO TOTAL CASES: EFFECTIVENESS OF JUSTICE |
| -- 1. Load data from the dataset  RAW\_DATA = LOAD 'cleaned' USING PigStorage(',') AS (state: chararray, city: chararray, year: int, solved: chararray, vicsex: chararray, vicage: int, vicrace: chararray, perpsex: chararray, perpage: int, perprace: chararray);  A = FOREACH RAW\_DATA GENERATE year AS y, state AS st, solved AS s;  B = GROUP A BY (y,st);  COUNT\_TOTAL = FOREACH B { C = FILTER A BY (s=='Yes');  GENERATE group, COUNT(A) AS tot, COUNT(C) AS del, (float) COUNT(C)/COUNT(A) AS frac;}  dump COUNT\_TOTAL;  STORE COUNT\_TOTAL INTO 'SolvedProportion' USING PigStorage(','); |

|  |
| --- |
| * + 1. PERPETRATOR RACE VS VICTIM RACE COUNTER |
| -- 1. Load data from the dataset  RAW\_DATA = LOAD 'cleaned' USING PigStorage(',') AS (state: chararray, city: chararray, year: int, solved: chararray, vicsex: chararray, vicage: int, vicrace: chararray, perpsex: chararray, perpage: int, perprace: chararray);  A = FOREACH RAW\_DATA GENERATE perprace as p, vicrace as v;  B = GROUP A by (p,v);  COUNTA = FOREACH B GENERATE group, COUNT(A);  STORE COUNTA INTO 'PerpVicRaceCount' USING PigStorage(','); |