Project Document **Advance Database Management Systems** **Manasi Dalvi**

Datasets

The project contains two datasets US-pollution dataset and US-drought dataset spanning from year 2000 to 2016 and an intermediary dataset for join.

Problem statement: Find co-relation between drought type and pollutant type

US-drought columns:

The data contains weekly observations about the extent and severity of drought in each county of the United States.

- county: the county name
- state: the state the county is in
- NONE: percentage of the county that is not in drought
- D0: percentage of the county that is in abnormally dry conditions
- D1: percentage of the county that is in moderate drought
- D2: percentage of the county that is in severe drought
- D3: percentage of the county that is in extreme drought
- D4: percentage of the county that is in exceptional drought
- validStart: the starting date of the week that these observations represent
- validEnd: the ending date of the week that these observations represent

US-pollution:

- State : State of monitoring site
- County : County of monitoring site
- City : City of the monitoring site
- Date Local : Date of monitoring

The four pollutants (NO2, O3, SO2 and O3) each has 5 specific columns. For instance, for NO2:

- NO2 Units : The units measured for NO2
- NO2 Mean : The arithmetic mean of concentration of NO2 within a given day
- NO2 AQI : The calculated air quality index of NO2 within a given day
- NO2 1st Max Value : The maximum value obtained for NO2 concentration in a given day
- NO2 1st Max Hour : The hour when the maximum NO2 concentration was recorded in a given day.

Dataset Link:

US-Pollution data: https://www.kaggle.com/sogun3/uspollution

US-drought data: https://www.kaggle.com/us-drought-monitor/united-states-droughts-by-county

Analysis:

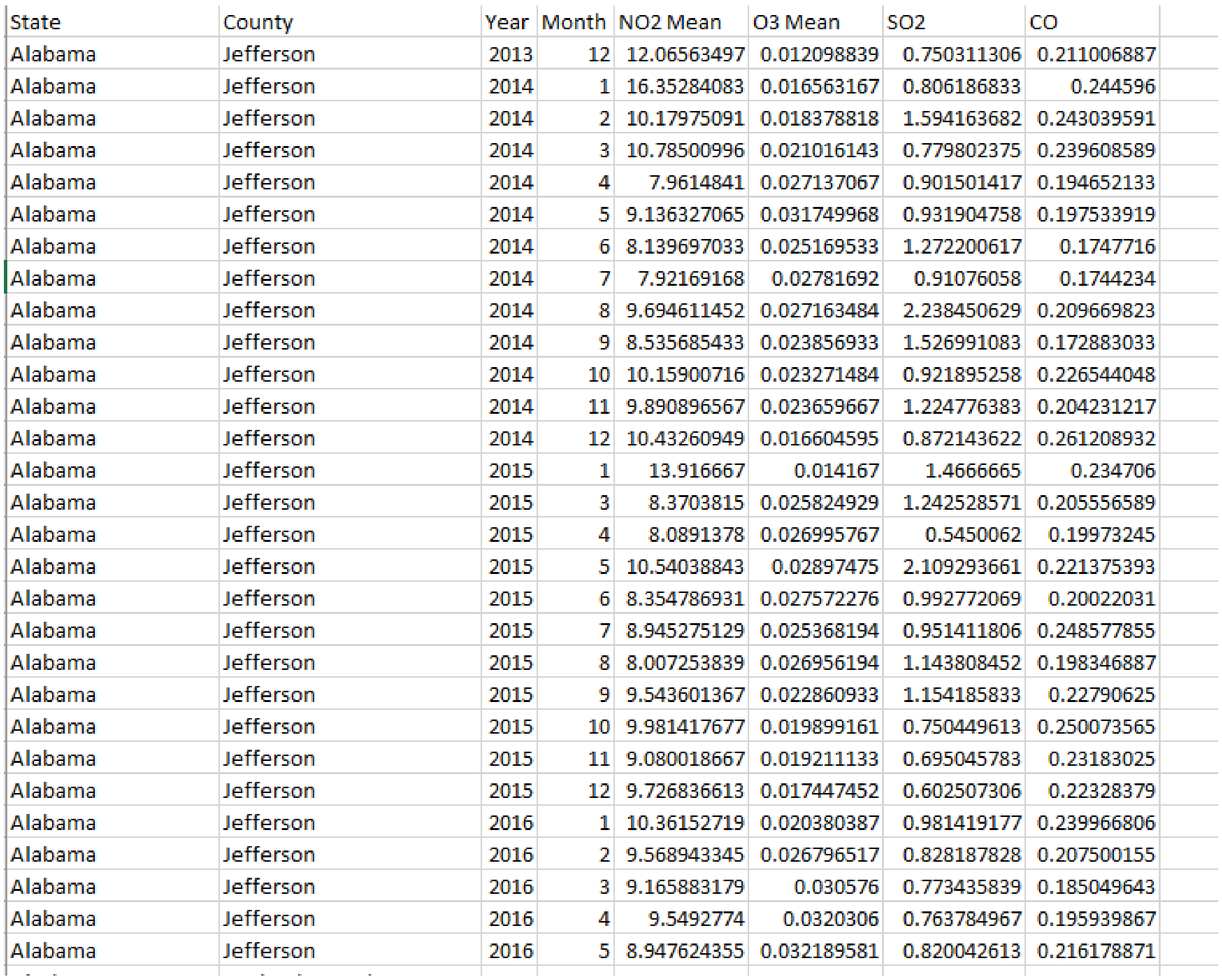
Individual analysis on pollution and drought datasets

Pollution data set:

1. Grouped data by state and year, month.

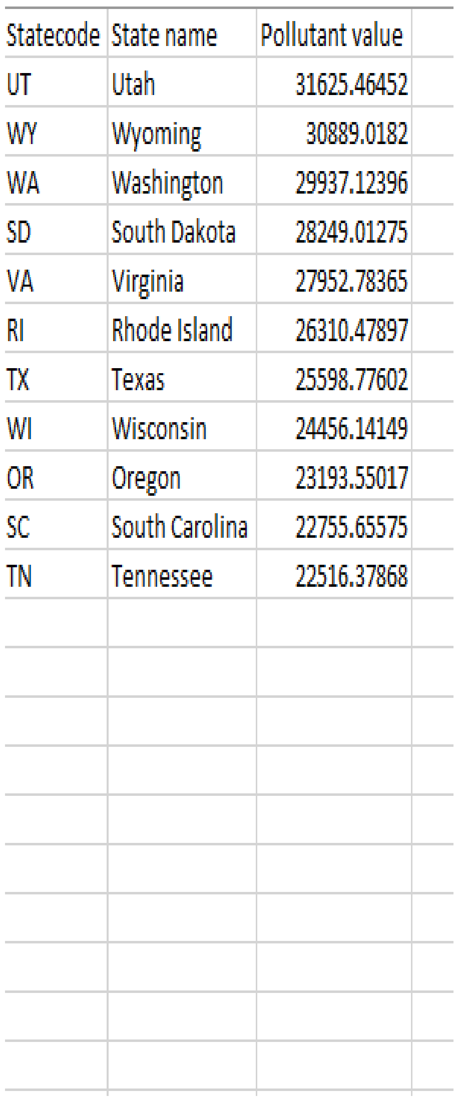
Performed aggregation on four pollutant columns to find out mean value per year.

Output file of pollution dataset analysis consisted of State name, Year, Month, and a single value for each year for each of the four pollutants.

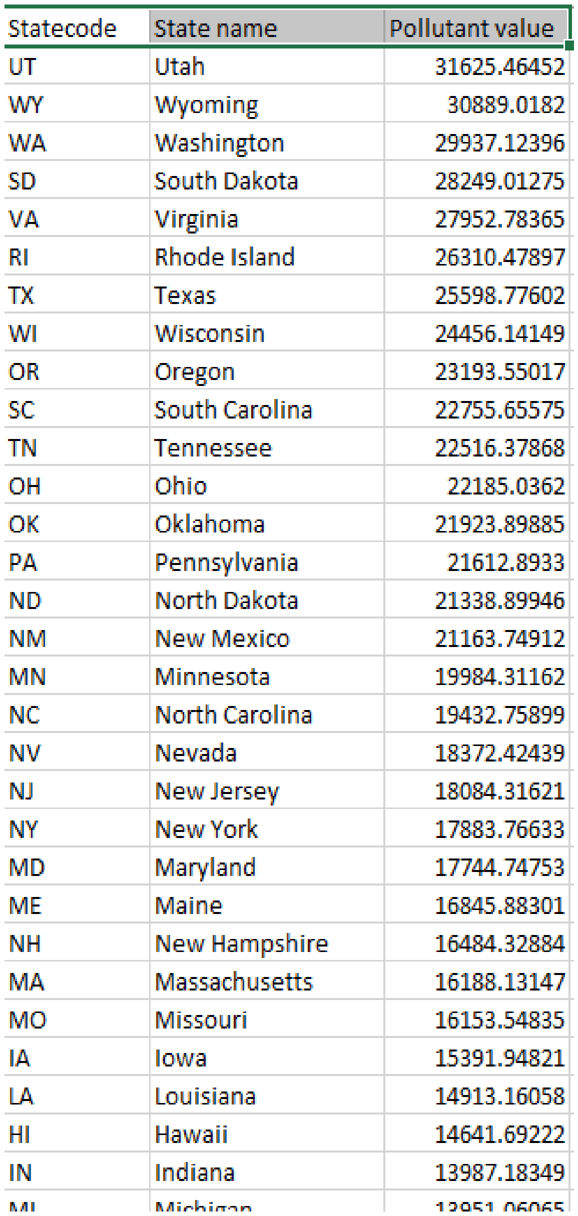


State	County	Year	Month	NO2 Mean	O3 Mean	SO2	CO
Alabama	Jefferson	2013	12	12.06563497	0.012098839	0.750311306	0.211006887
Alabama	Jefferson	2014	1	16.35284083	0.016563167	0.806186833	0.244596
Alabama	Jefferson	2014	2	10.17975091	0.018378818	1.594163682	0.243039591
Alabama	Jefferson	2014	3	10.78500996	0.021016143	0.779802375	0.239608589
Alabama	Jefferson	2014	4	7.9614841	0.027137067	0.901501417	0.194652133
Alabama	Jefferson	2014	5	9.136327065	0.031749968	0.931904758	0.197533919
Alabama	Jefferson	2014	6	8.139697033	0.025169533	1.272200617	0.1747716
Alabama	Jefferson	2014	7	7.92169168	0.02781692	0.91076058	0.1744234
Alabama	Jefferson	2014	8	9.694611452	0.027163484	2.238450629	0.209669823
Alabama	Jefferson	2014	9	8.535685433	0.023856933	1.526991083	0.172883033
Alabama	Jefferson	2014	10	10.15900716	0.023271484	0.921895258	0.226544048
Alabama	Jefferson	2014	11	9.890896567	0.023659667	1.224776383	0.204231217
Alabama	Jefferson	2014	12	10.43260949	0.016604595	0.872143622	0.261208932
Alabama	Jefferson	2015	1	13.916667	0.014167	1.4666665	0.234706
Alabama	Jefferson	2015	3	8.3703815	0.025824929	1.242528571	0.205556589
Alabama	Jefferson	2015	4	8.0891378	0.026995767	0.5450062	0.19973245
Alabama	Jefferson	2015	5	10.54038843	0.02897475	2.109293661	0.221375393
Alabama	Jefferson	2015	6	8.354786931	0.027572276	0.992772069	0.20022031
Alabama	Jefferson	2015	7	8.945275129	0.025368194	0.951411806	0.248577855
Alabama	Jefferson	2015	8	8.007253839	0.026956194	1.143808452	0.198346887
Alabama	Jefferson	2015	9	9.543601367	0.022860933	1.154185833	0.22790625
Alabama	Jefferson	2015	10	9.981417677	0.019899161	0.750449613	0.250073565
Alabama	Jefferson	2015	11	9.080018667	0.019211133	0.695045783	0.23183025
Alabama	Jefferson	2015	12	9.726836613	0.017447452	0.602507306	0.22328379
Alabama	Jefferson	2016	1	10.36152719	0.020380387	0.981419177	0.239966806
Alabama	Jefferson	2016	2	9.568943345	0.026796517	0.828187828	0.207500155
Alabama	Jefferson	2016	3	9.165883179	0.030576	0.773435839	0.185049643
Alabama	Jefferson	2016	4	9.5492774	0.0320306	0.763784967	0.195939867
Alabama	Jefferson	2016	5	8.947624355	0.032189581	0.820042613	0.216178871

2. **Top ten polluted cites in the year 2015**
   Total pollution value for all states in the year 2015 with top ten values.
   Using job chaining and secondary sorting.

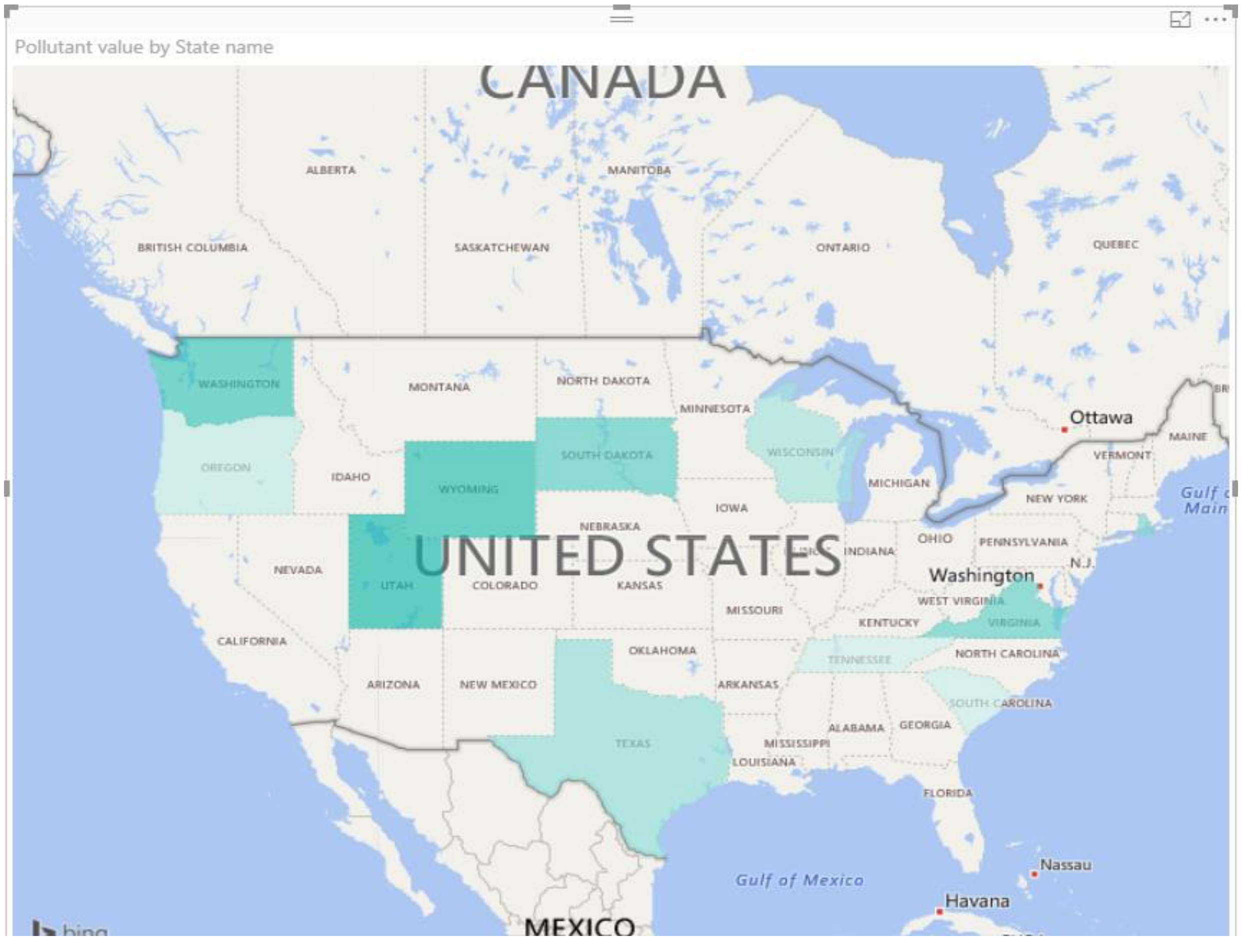


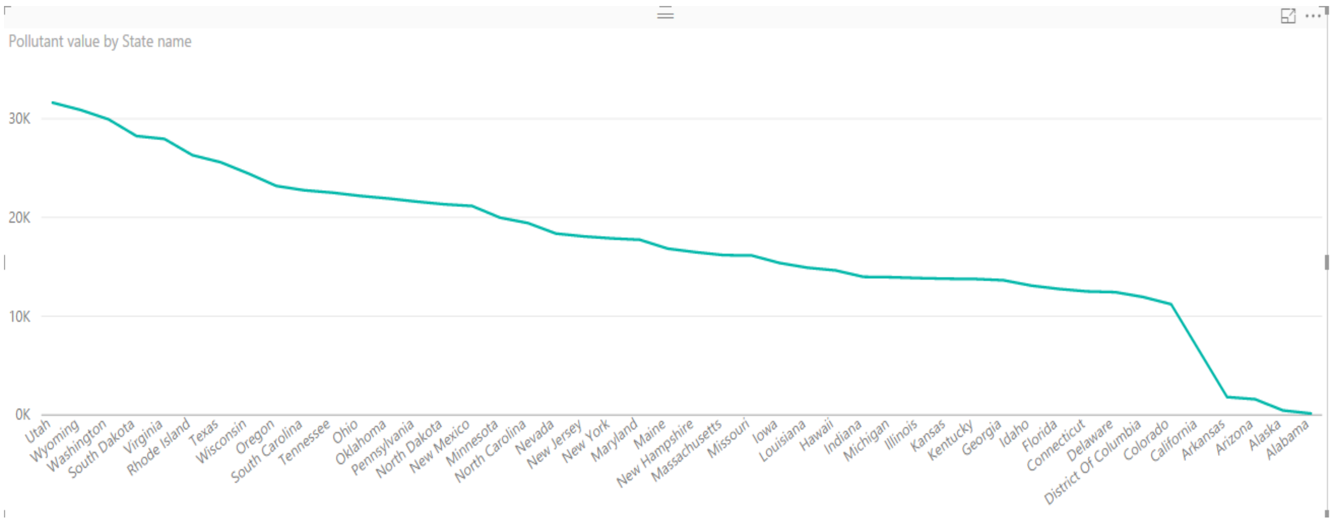
Statecode	State name	Pollutant value
UT	Utah	31625.46452
WY	Wyoming	30889.0182
WA	Washington	29937.12396
SD	South Dakota	28249.01275
VA	Virginia	27952.78365
RI	Rhode Island	26310.47897
TX	Texas	25598.77602
WI	Wisconsin	24456.14149
OR	Oregon	23193.55017
SC	South Carolina	22755.65575
TN	Tennessee	22516.37868



Statecode	State name	Pollutant value
UT	Utah	31625.46452
WY	Wyoming	30889.0182
WA	Washington	29937.12396
SD	South Dakota	28249.01275
VA	Virginia	27952.78365
RI	Rhode Island	26310.47897
TX	Texas	25598.77602
WI	Wisconsin	24456.14149
OR	Oregon	23193.55017
SC	South Carolina	22755.65575
TN	Tennessee	22516.37868
OH	Ohio	22185.0362
OK	Oklahoma	21923.89885
PA	Pennsylvania	21612.8933
ND	North Dakota	21338.89946
NM	New Mexico	21163.74912
MN	Minnesota	19984.31162
NC	North Carolina	19432.75899
NV	Nevada	18372.42439
NJ	New Jersey	18084.31621
NY	New York	17883.76633
MD	Maryland	17744.74753
ME	Maine	16845.88301
NH	New Hampshire	16484.32884
MA	Massachusetts	16188.13147
MO	Missouri	16153.54835
IA	Iowa	15391.94821
LA	Louisiana	14913.16058
HI	Hawaii	14641.69222
IN	Indiana	13987.18349
MI	Michigan	13951.06065

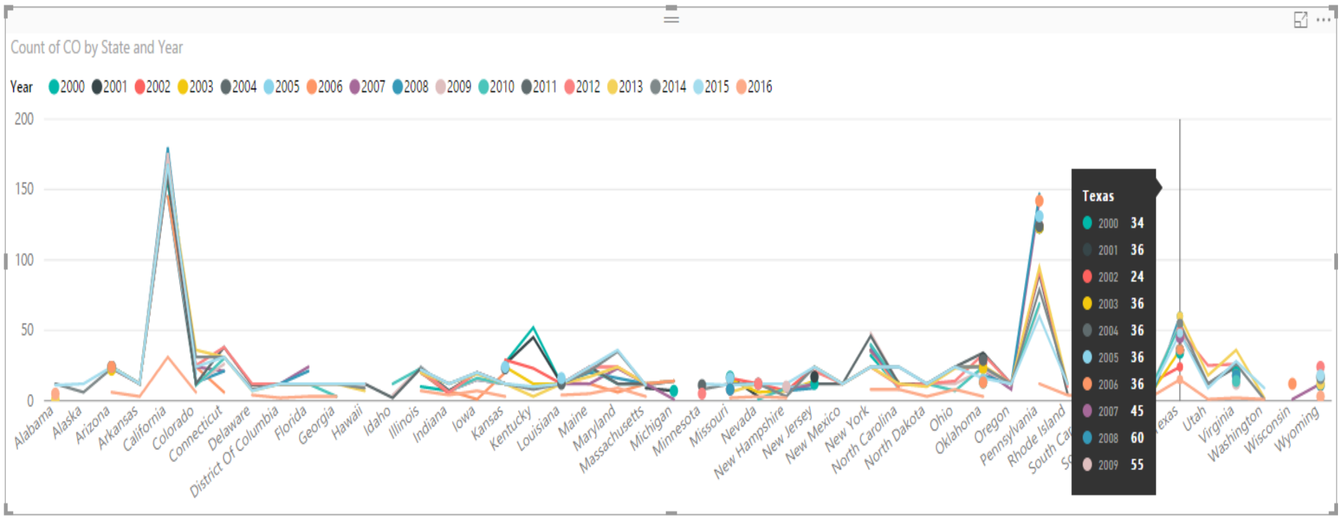
Visualization: Top 10 Polluted cities



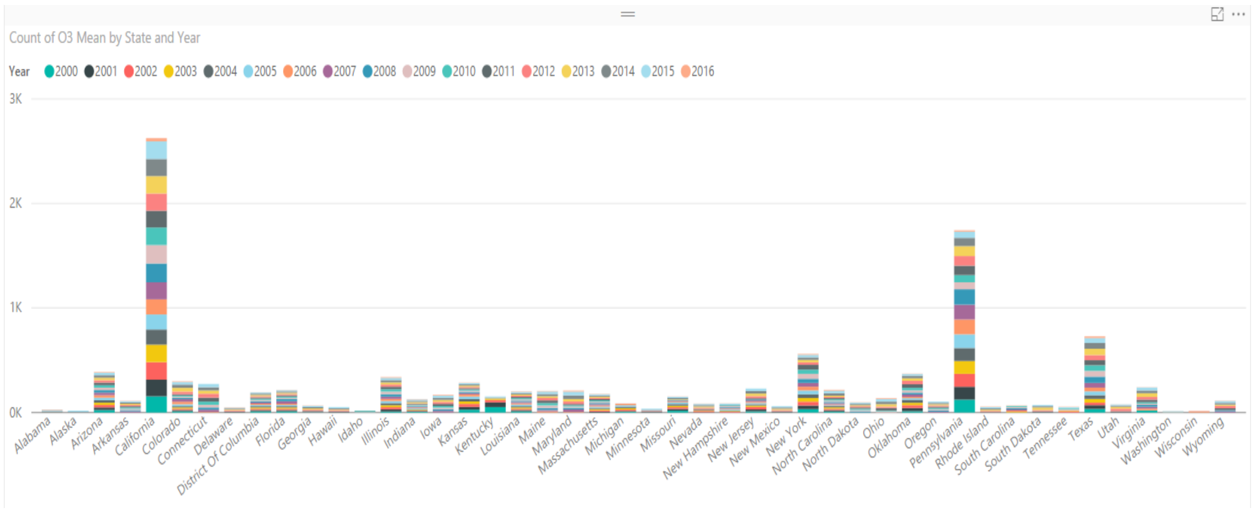


Pollution by state in descending order

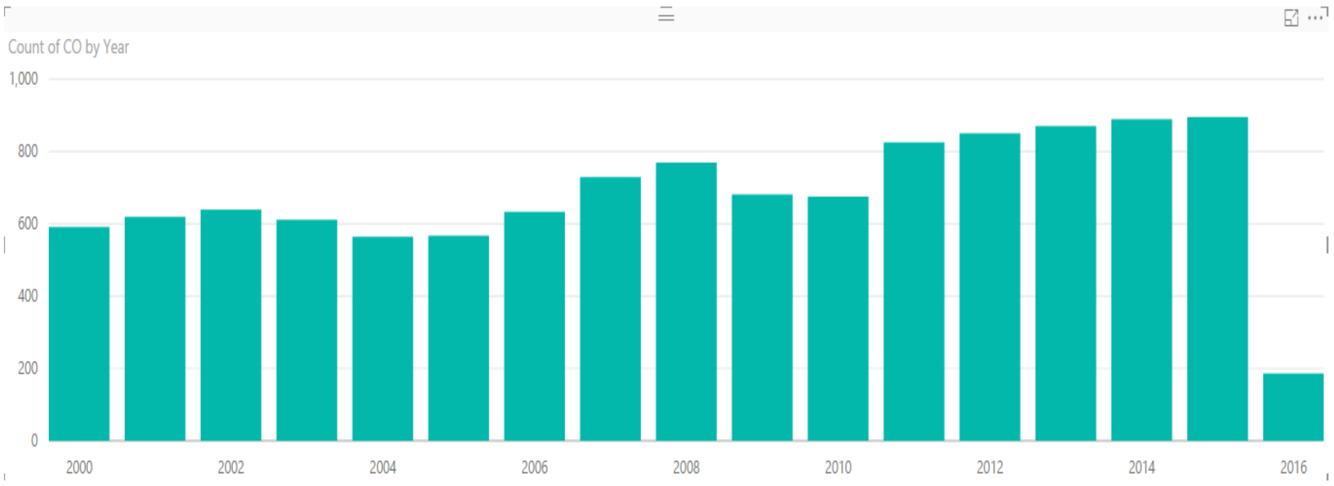
Graphs build on pollution dataset using PowerBI



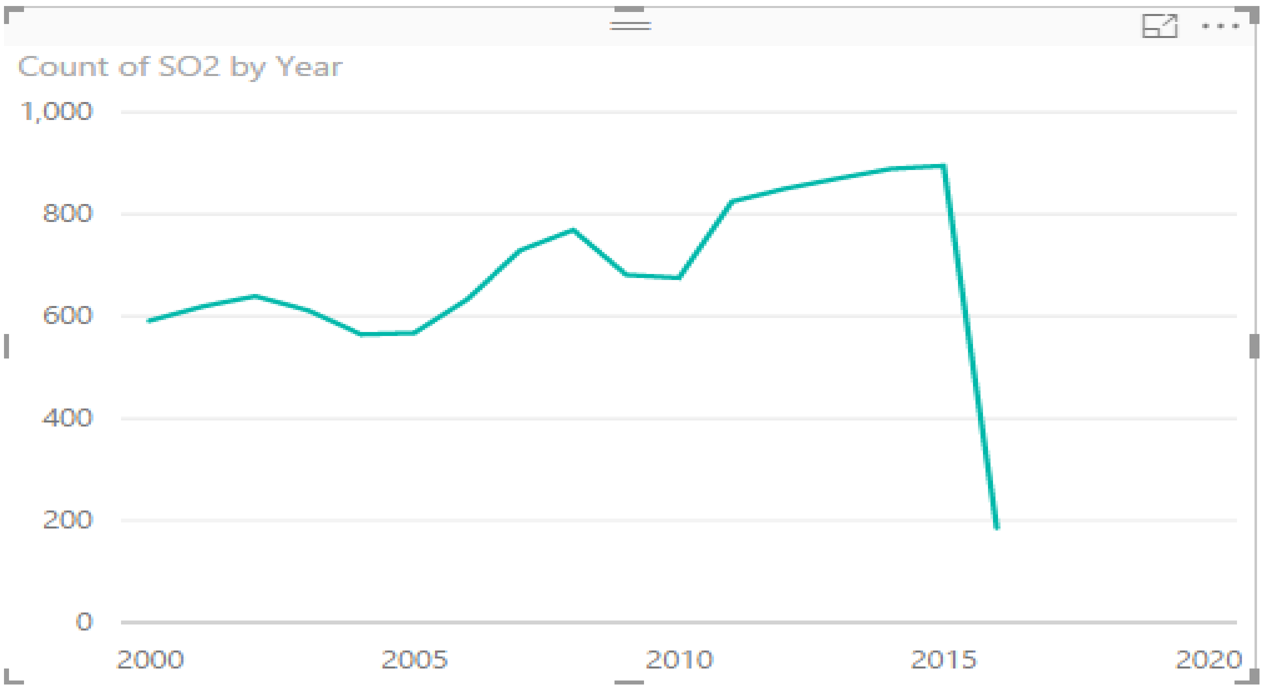
Count of CO by State and Year



Count of O3 by State and Year



CO distribution by year



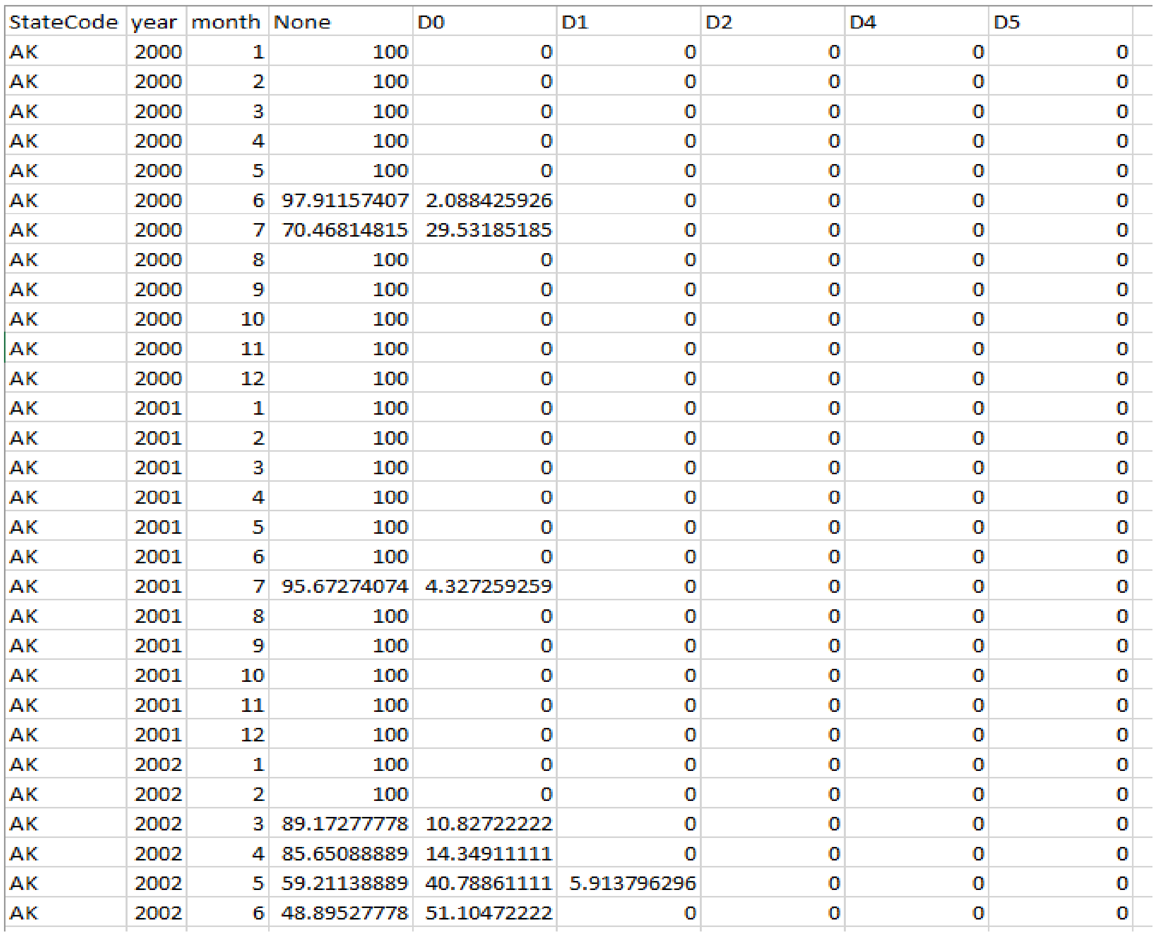
Count of SO2 by year

Drought Dataset Analysis

Dataset contains the percentage of area under a particular type of drought (None, D0,D1,D2,D3,D4)

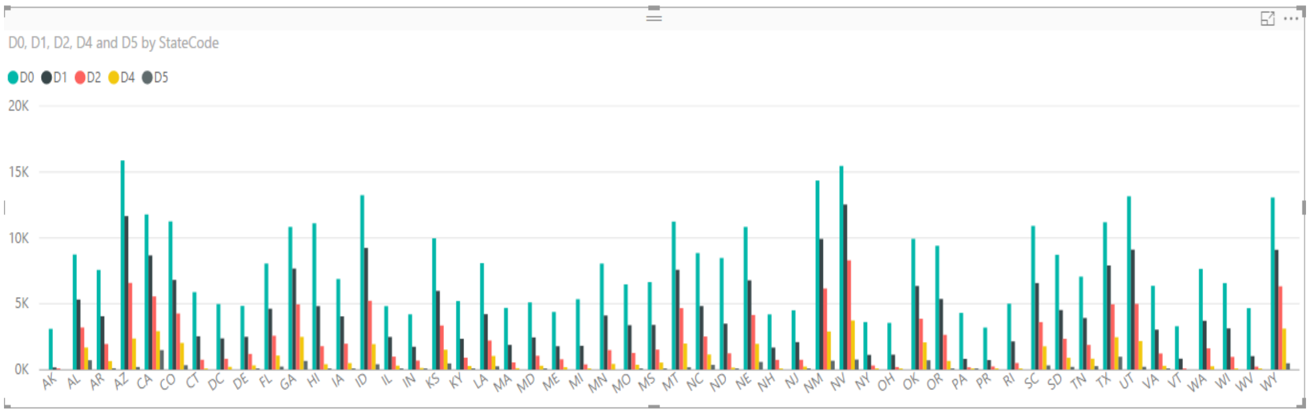
Performed numerical summarization on the data to group the data by Year and Month.

The output file consisted of State code Year-Month and the average of drought for that month-year.

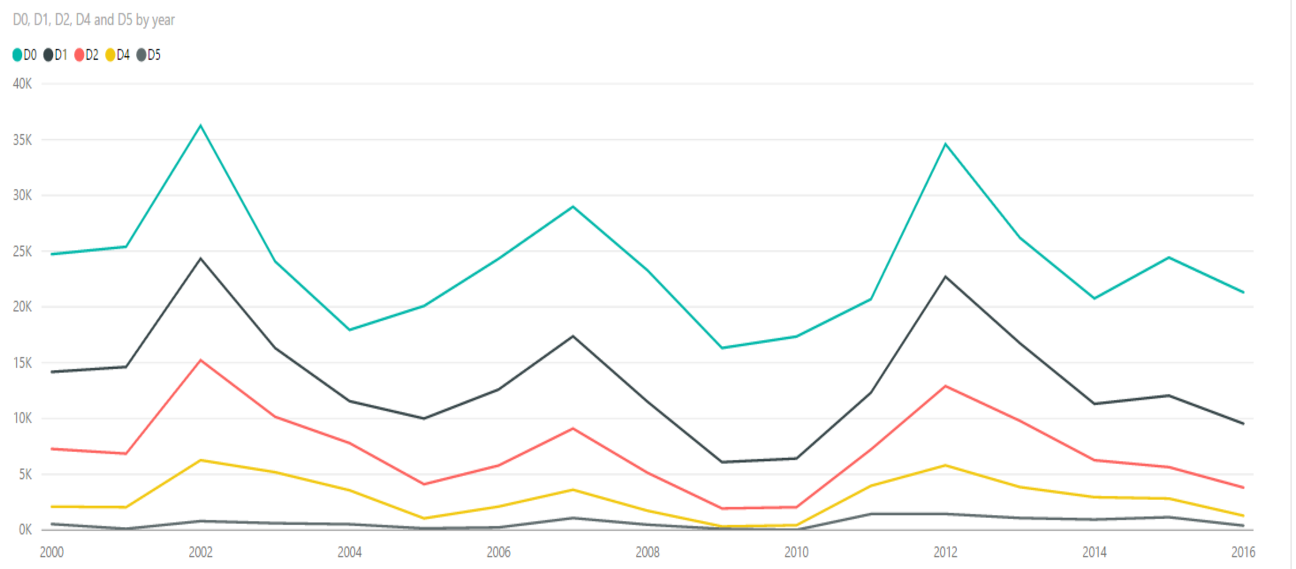


StateCode	year	month	None	D0	D1	D2	D4	D5
AK	2000	1	100	0	0	0	0	0
AK	2000	2	100	0	0	0	0	0
AK	2000	3	100	0	0	0	0	0
AK	2000	4	100	0	0	0	0	0
AK	2000	5	100	0	0	0	0	0
AK	2000	6	97.91157407	2.088425926	0	0	0	0
AK	2000	7	70.46814815	29.53185185	0	0	0	0
AK	2000	8	100	0	0	0	0	0
AK	2000	9	100	0	0	0	0	0
AK	2000	10	100	0	0	0	0	0
AK	2000	11	100	0	0	0	0	0
AK	2000	12	100	0	0	0	0	0
AK	2001	1	100	0	0	0	0	0
AK	2001	2	100	0	0	0	0	0
AK	2001	3	100	0	0	0	0	0
AK	2001	4	100	0	0	0	0	0
AK	2001	5	100	0	0	0	0	0
AK	2001	6	100	0	0	0	0	0
AK	2001	7	95.67274074	4.327259259	0	0	0	0
AK	2001	8	100	0	0	0	0	0
AK	2001	9	100	0	0	0	0	0
AK	2001	10	100	0	0	0	0	0
AK	2001	11	100	0	0	0	0	0
AK	2001	12	100	0	0	0	0	0
AK	2002	1	100	0	0	0	0	0
AK	2002	2	100	0	0	0	0	0
AK	2002	3	89.17277778	10.82722222	0	0	0	0
AK	2002	4	85.65088889	14.34911111	0	0	0	0
AK	2002	5	59.21138889	40.78861111	5.913796296	0	0	0
AK	2002	6	48.89527778	51.10472222	0	0	0	0

Graphs based on Drought dataset using PowerBI



Drought severity by State



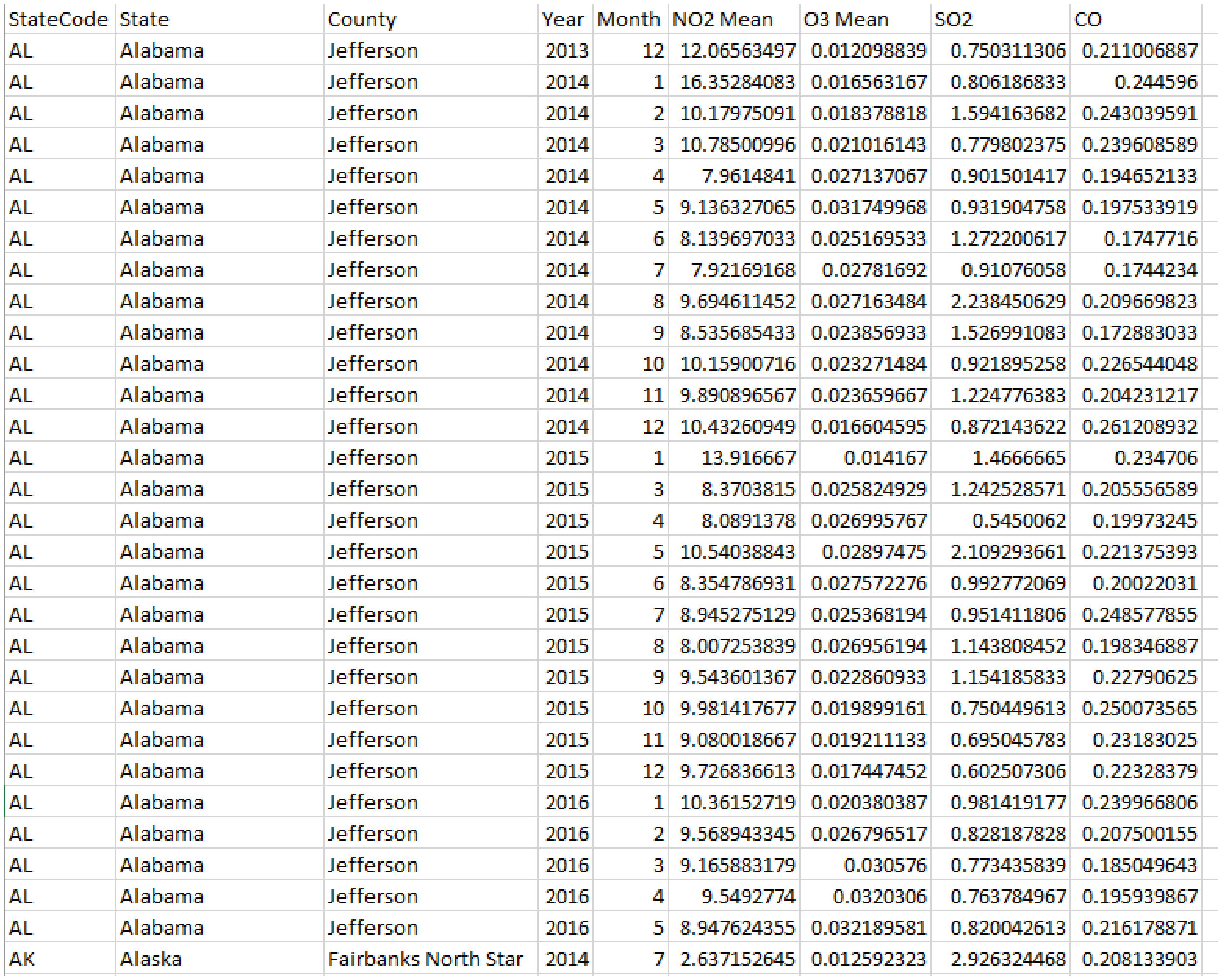
Drought severity by year

Finding correlation between pollutant type and draught type:

Input: Aggregate files obtained from the above analysis: pollution-aggregate, drought-aggregate

To join the two files an intermediary file statecode.csv, is used which contains the state names and codes is used.

Performed a map-side join on the statecode.csv and pollution-aggregate.csv to obtain common key column of state-code.



StateCode	State	County	Year	Month	NO2 Mean	O3 Mean	SO2	CO
AL	Alabama	Jefferson	2013	12	12.06563497	0.012098839	0.750311306	0.211006887
AL	Alabama	Jefferson	2014	1	16.35284083	0.016563167	0.806186833	0.244596
AL	Alabama	Jefferson	2014	2	10.17975091	0.018378818	1.594163682	0.243039591
AL	Alabama	Jefferson	2014	3	10.78500996	0.021016143	0.779802375	0.239608589
AL	Alabama	Jefferson	2014	4	7.9614841	0.027137067	0.901501417	0.194652133
AL	Alabama	Jefferson	2014	5	9.136327065	0.031749968	0.931904758	0.197533919
AL	Alabama	Jefferson	2014	6	8.139697033	0.025169533	1.272200617	0.1747716
AL	Alabama	Jefferson	2014	7	7.92169168	0.02781692	0.91076058	0.1744234
AL	Alabama	Jefferson	2014	8	9.694611452	0.027163484	2.238450629	0.209669823
AL	Alabama	Jefferson	2014	9	8.535685433	0.023856933	1.526991083	0.172883033
AL	Alabama	Jefferson	2014	10	10.15900716	0.023271484	0.921895258	0.226544048
AL	Alabama	Jefferson	2014	11	9.890896567	0.023659667	1.224776383	0.204231217
AL	Alabama	Jefferson	2014	12	10.43260949	0.016604595	0.872143622	0.261208932
AL	Alabama	Jefferson	2015	1	13.916667	0.014167	1.4666665	0.234706
AL	Alabama	Jefferson	2015	3	8.3703815	0.025824929	1.242528571	0.205556589
AL	Alabama	Jefferson	2015	4	8.0891378	0.026995767	0.5450062	0.19973245
AL	Alabama	Jefferson	2015	5	10.54038843	0.02897475	2.109293661	0.221375393
AL	Alabama	Jefferson	2015	6	8.354786931	0.027572276	0.992772069	0.20022031
AL	Alabama	Jefferson	2015	7	8.945275129	0.025368194	0.951411806	0.248577855
AL	Alabama	Jefferson	2015	8	8.007253839	0.026956194	1.143808452	0.198346887
AL	Alabama	Jefferson	2015	9	9.543601367	0.022860933	1.154185833	0.22790625
AL	Alabama	Jefferson	2015	10	9.981417677	0.019899161	0.750449613	0.250073565
AL	Alabama	Jefferson	2015	11	9.080018667	0.019211133	0.695045783	0.23183025
AL	Alabama	Jefferson	2015	12	9.726836613	0.017447452	0.602507306	0.22328379
AL	Alabama	Jefferson	2016	1	10.36152719	0.020380387	0.981419177	0.239966806
AL	Alabama	Jefferson	2016	2	9.568943345	0.026796517	0.828187828	0.207500155
AL	Alabama	Jefferson	2016	3	9.165883179	0.030576	0.773435839	0.185049643
AL	Alabama	Jefferson	2016	4	9.5492774	0.0320306	0.763784967	0.195939867
AL	Alabama	Jefferson	2016	5	8.947624355	0.032189581	0.820042613	0.216178871
AK	Alaska	Fairbanks North Star	2014	7	2.637152645	0.012592323	2.926324468	0.208133903

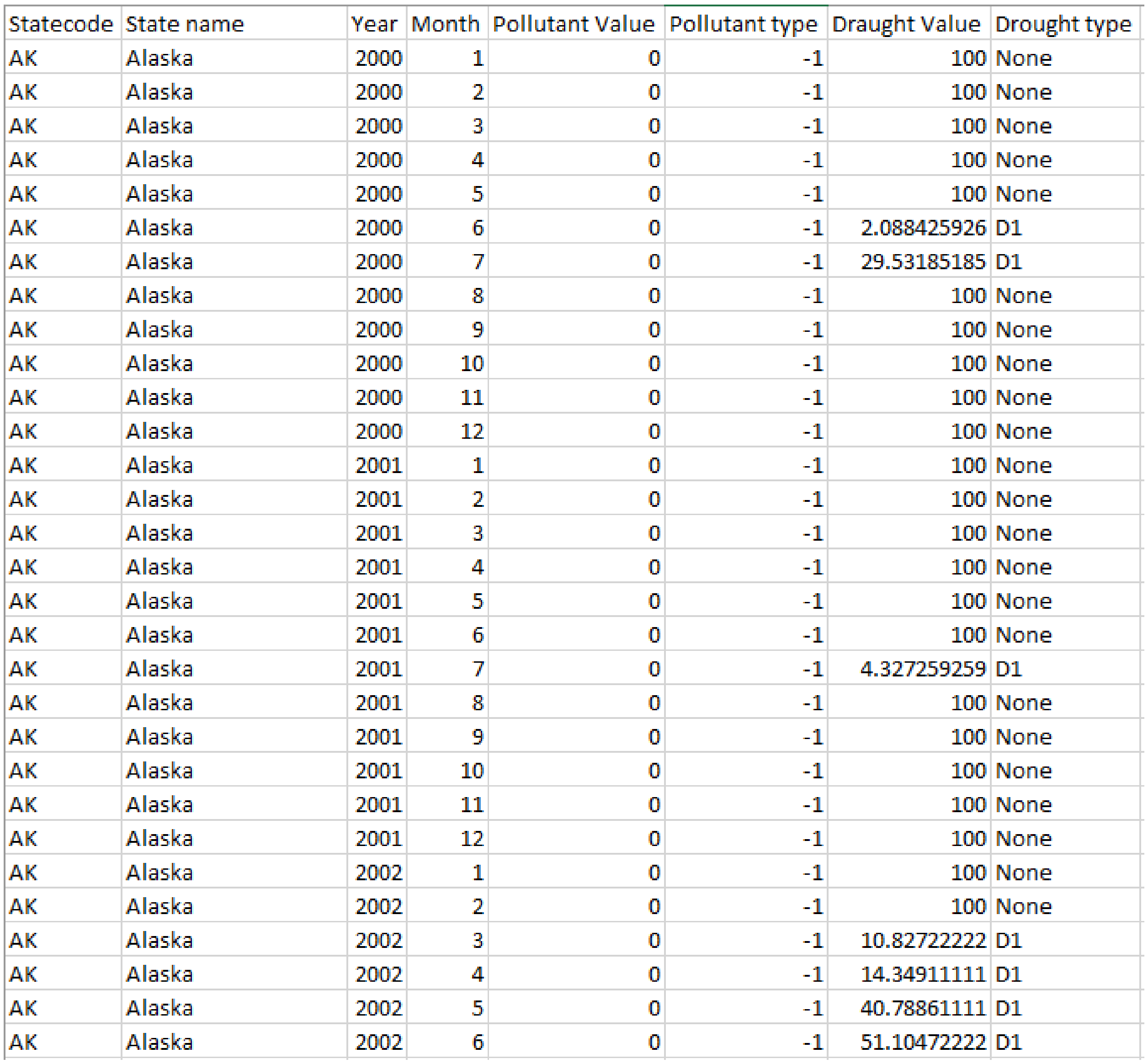
A second Map-side join was performed on Draught-aggregate file with statecodes.csv to insert state name in it.

One more aggregation was done on the draught file to find the maximum type of drought.

The final reduce side join was performed on the two data sets to obtain the final value containing

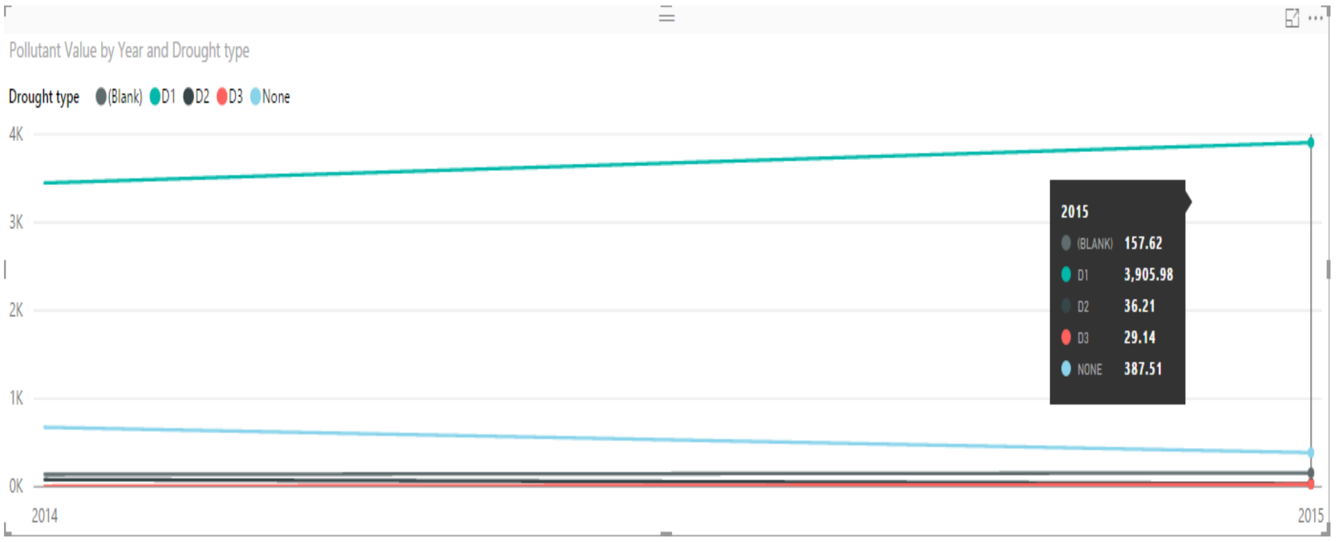
State code, state name, year, month, max pollutant, average draught type for the year.

Final output:



Statecode	State name	Year	Month	Pollutant Value	Pollutant type	Draught Value	Drought type
AK	Alaska	2000	1	0	-1	100	None
AK	Alaska	2000	2	0	-1	100	None
AK	Alaska	2000	3	0	-1	100	None
AK	Alaska	2000	4	0	-1	100	None
AK	Alaska	2000	5	0	-1	100	None
AK	Alaska	2000	6	0	-1	2.088425926	D1
AK	Alaska	2000	7	0	-1	29.53185185	D1
AK	Alaska	2000	8	0	-1	100	None
AK	Alaska	2000	9	0	-1	100	None
AK	Alaska	2000	10	0	-1	100	None
AK	Alaska	2000	11	0	-1	100	None
AK	Alaska	2000	12	0	-1	100	None
AK	Alaska	2001	1	0	-1	100	None
AK	Alaska	2001	2	0	-1	100	None
AK	Alaska	2001	3	0	-1	100	None
AK	Alaska	2001	4	0	-1	100	None
AK	Alaska	2001	5	0	-1	100	None
AK	Alaska	2001	6	0	-1	100	None
AK	Alaska	2001	7	0	-1	4.327259259	D1
AK	Alaska	2001	8	0	-1	100	None
AK	Alaska	2001	9	0	-1	100	None
AK	Alaska	2001	10	0	-1	100	None
AK	Alaska	2001	11	0	-1	100	None
AK	Alaska	2001	12	0	-1	100	None
AK	Alaska	2002	1	0	-1	100	None
AK	Alaska	2002	2	0	-1	100	None
AK	Alaska	2002	3	0	-1	10.82722222	D1
AK	Alaska	2002	4	0	-1	14.34911111	D1
AK	Alaska	2002	5	0	-1	40.78861111	D1
AK	Alaska	2002	6	0	-1	51.10472222	D1

Graphs for co-relation on final dataset:



Pollutant value by year and drought type

Appendix:

Top ten Polluted cities
Pollution Analysis
Drought Analysis
Pollution state code join
Pollution-drought reduce side join

1. Top Ten Polluted:

Driver Class

```
/*
 * 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 totalpollutiontop10;
```

```
import java.net.URI;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.RunningJob;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapreduce.Job;


import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 *
 * @author Manasi
 */
public class TotalPollutionTop10 {

    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(TotalPollutionTop10.class);
        conf.setJobName("totalpollution");

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

        conf.setMapperClass(TotalPollutionMapper.class);
        conf.setReducerClass(TotalPollutionReducer.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[2]));
```

```
          try{
            DistributedCache.addCacheFile(new URI(args[1]), conf);
        }
        catch(Exception e){
            System.out.println(e);
        }

          RunningJob rj = JobClient.runJob(conf);
          boolean success = rj.isComplete();

          if(success){
              JobConf confNew = new JobConf(TotalPollutionTop10.class);
          confNew.setJobName("totalpollutionsort");

          confNew.setOutputKeyClass(DoubleWritable.class);
          confNew.setOutputValueClass(Text.class);

          confNew.setMapperClass(ReducerSorterMapper.class);
          confNew.setReducerClass(ReducerSorterReducer.class);

          confNew.setInputFormat(TextInputFormat.class);
          confNew.setOutputFormat(TextOutputFormat.class);

          FileInputFormat.setInputPaths(confNew, new Path(args[2] + "/part-00000"));
          FileOutputFormat.setOutputPath(confNew, new Path(args[3]));




          confNew.setOutputKeyComparatorClass(TotalPollutionComparator.class);

          JobClient.runJob(confNew);

              }
          else{
              System.out.println("First Job NOt successful");
          }

      }


  }
```

Mapper1 – TotalPollutionMapper

```
/*
 * 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 totalpollutiontop10;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

//import org.apache.hadoop.mapreduce.Reducer.Context;

/**
 *
 * @author Manasi
 */
public class TotalPollutionMapper extends MapReduceBase implements
Mapper<LongWritable, Text, Text, Text> {

        private Map<String,String> stateCodes = new HashMap<String,String>();

    @Override
    public void configure(JobConf job)  {
      // Get the cached archives/files
      File f = new File("statecodenames.csv");
```

```
            System.out.println("filenme = "+ f.getName());


            try{
                BufferedReader reader = new BufferedReader(new FileReader(f));
                String line = reader.readLine();
                while(line!=null){
                    String [] tokens = line.split(",");
                    stateCodes.put(tokens[1].trim().toUpperCase(), tokens[0].trim());
                    System.out.println("State name = "+ tokens[1].trim() +" : Value =
"+stateCodes.get(tokens[1].trim().toUpperCase()));

                    line = reader.readLine();
                }
            }
            catch(FileNotFoundException e){
                System.out.println("File Not found -"+e);

            }
            catch(IOException e){
                System.out.println("IO - " + e);
            }

        }


    @Override
    public void map(LongWritable key, Text value, OutputCollector<Text, Text>
output,Reporter reporter)
            throws IOException {

    if (value.toString().length() > 0) {
            //System.out.println(value);

            String[] attarray = value.toString().split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)", -1);

            if(!attarray[0].equals("")){

               String dateattarray[] = attarray[8].split("-"); //2000-01-01
               String val=attarray[10]+"\t"+attarray[15]+"\t"+attarray[20]+"\t"+attarray[25];
            try{


               String stateCode = stateCodes.get(attarray[5].toUpperCase());

               if(stateCode != null){




double NO2 = 0.0;
        long valueCount=0;
       double SO2 = 0.0;

       double O3 = 0.0;
       double CO = 0.0;
             try{
                NO2= Double.parseDouble(attarray[0]);
             }
             catch(NumberFormatException e){
                NO2=0.0;
             }
             try{
                O3= Double.parseDouble(attarray[1]);
             }
             catch(NumberFormatException e){
                O3=0.0;
             }
             try{
                SO2= Double.parseDouble(attarray[2]);
             }
             catch(NumberFormatException e){
                SO2=0.0;
             }try{
                CO= Double.parseDouble(attarray[3]);
             }
             catch(NumberFormatException e){
                CO=0.0;
             }


             double totalPollution =  (NO2 + O3 + SO2 + CO)/4;
```

```
                //System.out.println("Mapper Key = "+stateCode.toUpperCase() +"\t"+
attarray[5] + " - value - " + totalPollution);
                output.collect(new Text(stateCode.toUpperCase() +"\t"+ attarray[5]), new
Text("" + totalPollution));


                  }

               }
               catch(ArrayIndexOutOfBoundsException e){
                   try{
                   System.out.println("Error = "+ attarray[0] + ","+attarray[5]+"," + attarray[6]);
                   }
                   catch(ArrayIndexOutOfBoundsException ex){
                       System.out.println("Error still exists = "+attarray[0]);
                   }
               }


                //System.out.print("MApper"+c1.getCountymonthyear());
            }
    }
    }
}
```

Reducer 1-TotalPollutionReducer

```
package totalpollutiontop10;

import java.io.IOException;
import java.util.Iterator;

//import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class TotalPollutionReducer extends MapReduceBase implements
   Reducer<Text, Text, Text, Text > {
    @Override
    public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text>
output, Reporter report) throws IOException {

        double pollution = 0.0;
         long valueCount=0;



       //System.out.print("REducer"+key.toString()+"value"+values);
        while(values.hasNext()) {

                Text value = values.next();
                System.out.print("REducer"+key.toString()+"value"+value.toString());
                String v=value.toString();

                try{
                    pollution+= Double.parseDouble(v.trim());
                }
                catch(NumberFormatException e){
                    pollution+=0.0;
                }




            valueCount++;




        }
       if(valueCount!= 0){
        pollution /= valueCount;
       }
       String keypart[] = key.toString().split("\t");
       String stateCode = keypart[0];
       String stateName = keypart[1];

        output.collect(new Text(stateCode+"\t"+stateName), new Text("" + pollution));

    }

}
```

Mapper 2:

```
package totalpollutiontop10;

import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

/**
 *
 * @author manasi dalvi
 */
public class ReducerSorterMapper extends MapReduceBase implements
   Mapper<LongWritable,/*Input key Type */
   Text,                /*Input value Type*/
   DoubleWritable ,                /*Output key Type*/
   Text>        /*Output value Type*/
   {


     @Override
    public void map(LongWritable key, Text value, OutputCollector<DoubleWritable,
Text> output,Reporter reporter)
            throws IOException {

        String line[] = value.toString().split("\t");
        double val = Double.parseDouble(line[2].trim());
        //System.out.println("Value = " + "" + val + "\nvalue = "+line[0] + "\t" + line[1]);
        output.collect(new DoubleWritable(val),new Text(line[0] + "\t" + line[1]));

    }


}
```

Reducer 2: ReducerSorterReducer

```
package totalpollutiontop10;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;


  public  class ReducerSorterReducer extends MapReduceBase implements
        Reducer<DoubleWritable, Text, Text, Text> {
     public static long counter = 0;
     @Override
     public void reduce(DoubleWritable key, Iterator<Text> values,
           OutputCollector<Text, Text> output, Reporter reporter)
           throws IOException {

        while (values.hasNext()) {
           String parts[] = values.next().toString().split("\t");
           String stateCode = parts[0];
           String stateName = parts[1];
           //System.out.println("Reducer : "+stateCode + "\t"+stateName + " : Value = "+
key.get());
           if(counter <= 10){
           output.collect(new Text(stateCode+"\t"+stateName), new Text(""+ key.get()));
           counter++;
           }
          else{
             return;
           }
        }
        }
 }
```

Comparator: TotalPollutionComparator

```
package totalpollutiontop10;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

  public class TotalPollutionComparator extends WritableComparator {
    protected TotalPollutionComparator() {
        super(DoubleWritable.class, true);
    }
    @SuppressWarnings("rawtypes")
    @Override
    public int compare(WritableComparable w1, WritableComparable w2) {
        DoubleWritable key1 = (DoubleWritable) w1;
        DoubleWritable key2 = (DoubleWritable) w2;
        return -1 * key1.compareTo(key2);
    }
}
```

Composite Key

```
package totalpollutiontop10;

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;


public class CompositeKey implements Writable,
            WritableComparable<CompositeKey>{


    private String state;
   // private String county;
    private String countymonthyear;
   public CompositeKey(){

   }

   public CompositeKey(String state, String countymonthyear){
       this.state=state;
      // this.county=county;
       this.countymonthyear=countymonthyear;
   }


   @Override
          public String toString() {
                    return (new StringBuilder().append(state).append("\t")
                                        .append("\t").append(countymonthyear)).toString();
          }




          public void readFields(DataInput dataInput) throws IOException {
                    state = WritableUtils.readString(dataInput);
                    countymonthyear = WritableUtils.readString(dataInput);


          }

          public void write(DataOutput dataOutput) throws IOException {
                    WritableUtils.writeString(dataOutput, state);
                    //WritableUtils.writeString(dataOutput, county);
             WritableUtils.writeString(dataOutput, countymonthyear);
          }

       public int compareTo(CompositeKey objKeyPair) {



             if (objKeyPair == null)
                return 0;
 int intcnt = state.compareTo(objKeyPair.state);

 return intcnt;
```

```
        }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCountymonthyear() {
        return countymonthyear;
    }

    public void setCountymonthyear(String countymonthyear) {
        this.countymonthyear = countymonthyear;
    }
}
```

2. **Pollution Analysis:**

Driver Class:

```
/*
 * 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 pollutionanalysisone;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 *
 * @author Manasi
 */
public class PollutionAnalysisOne extends Configured implements Tool {

    @Override
    public int run(String[] strings) throws Exception {
        if (strings.length != 2) {
                              System.out.printf("Two parameters are required for
SecondarySortBasicDriver- <input dir> <output dir>\n");
                              return -1;
                    }


        JobConf conf = new JobConf(PollutionAnalysisOne.class);

      Job job = Job.getInstance(conf, "PollutionAnalysisOne");
        conf.setJobName("PollutionAnalysisOne");
      conf.setNumReduceTasks(1);
      conf.setNumMapTasks(3);
      conf.setOutputKeyClass(CompositeKey.class);
      conf.setOutputValueClass(Text.class);

      conf.setMapperClass(SecondarySortMapper.class);
      conf.setReducerClass(SecondarySortReducer.class);

      conf.setPartitionerClass(Partitioner.class);

      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(strings[0]));
        FileOutputFormat.setOutputPath(conf, new Path(strings[1]));


        JobClient.runJob(conf);
        //return (job.waitForCompletion(true) ? 0 : 1) == 0?true:false;
```

```
                boolean success = job.waitForCompletion(true);
                return success ? 0 : 1;
        }

   public static void main(String[] args) throws Exception {
                int exitCode = ToolRunner.run(new Configuration(),
                                new PollutionAnalysisOne(), args);
                System.exit(exitCode);
        }

 }
```

Mapper Class

```
/*
 * 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 pollutionanalysisone;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

//import org.apache.hadoop.mapreduce.Reducer.Context;

/**
 *
 * @author Manasi
 */
public class SecondarySortMapper extends MapReduceBase implements
Mapper<LongWritable, Text, CompositeKey, Text> {

   @Override
   public void map(LongWritable key, Text value, OutputCollector<CompositeKey,
Text> output,Reporter reporter)
        throws IOException {
```

```
       // if (value.toString().length() > 0) { //System.out.println(value); String attarray[] =
value.toString().split(","); if(!attarray[0].equals("")){ String dateattarray[] =
attarray[8].split("-"); //2000-01-01 context.write( new CompositeKey( attarray[5],
(attarray[6] + "\t" + dateattarray[0] + "\t" + dateattarray[1])), NullWritable.get()); } }

    if (value.toString().length() > 0) {
            //System.out.println(value);

            String[] attarray = value.toString().split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)", -1);
            //String attarray[] = value.toString().split(",");
            if(!attarray[0].equals("")){

               String dateattarray[] = attarray[8].split("-"); //2000-01-01
               String val=attarray[10]+"\t"+attarray[15]+"\t"+attarray[20]+"\t"+attarray[25];
               //CompositeKey c1=new CompositeKey(attarray[5],dateattarray[0] + "\t" +
dateattarray[1]);

               try{
                CompositeKey c1=new
CompositeKey(attarray[5]+"\t"+attarray[6]+"\t"+dateattarray[0] + "\t" +
dateattarray[1],"");
                output.collect(c1, new Text(val));
               }
               catch(ArrayIndexOutOfBoundsException e){
                   try{
                   System.out.println("Error = "+ attarray[0] + ","+attarray[5]+"," + attarray[6]);
                   }
                   catch(ArrayIndexOutOfBoundsException ex){
                       System.out.println("Error still exists = "+attarray[0]);
                   }
               }



                //System.out.print("MApper"+c1.getCountymonthyear());
            }
    }
    }
}
```

Reducer Class

```
package pollutionanalysisone;

import java.io.IOException;
import java.util.Iterator;

//import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import pollutionanalysisone.CompositeKey;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class SecondarySortReducer extends MapReduceBase implements
   Reducer<CompositeKey, Text, CompositeKey, Text > {

    @Override
    public void reduce(CompositeKey key, Iterator<Text> values,
OutputCollector<CompositeKey, Text> output, Reporter report) throws IOException {

        double NO2 = 0.0;
         long valueCount=0;
        double SO2 = 0.0;

        double O3 = 0.0;
        double CO = 0.0;

       //System.out.print("REducer"+key.toString()+"value"+values);
        while(values.hasNext()) {
            //System.out.print("REducer"+key.toString()+"value"+value.toString());

                Text value = values.next();
                String v[]=value.toString().split("\t");

                try{
                    NO2+= Double.parseDouble(v[0]);
                }
                catch(NumberFormatException e){
                    NO2+=0.0;
                }
                try{
                    O3+= Double.parseDouble(v[1]);
                }
                catch(NumberFormatException e){
                    O3+=0.0;
                }
                try{
                    SO2+= Double.parseDouble(v[2]);
                }
                catch(NumberFormatException e){
                    SO2+=0.0;
                }try{
                    CO+= Double.parseDouble(v[3]);
                }
                catch(NumberFormatException e){
                    CO+=0.0;
                }


            valueCount++;
        }

        NO2 /= valueCount;
        O3 /= valueCount;
        SO2 /= valueCount;
        CO /= valueCount;

        output.collect(key, new Text(NO2+"\t"+O3+"\t"+SO2+"\t"+CO));

    }

}
```

Composite Key class

```
/*
 * 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 pollutionanalysisone;

/**
 *
 * @author Manasi
 */

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;


public class CompositeKey implements Writable,
        WritableComparable<CompositeKey>{


    private String state;
   // private String county;
    private String countymonthyear;


   public CompositeKey(){

   }

   public CompositeKey(String state, String countymonthyear){
       this.state=state;
      // this.county=county;
       this.countymonthyear=countymonthyear;
   }

//    CompositeKey(String toString, String string) {
//     //   throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
//    }
//
   @Override
        public String toString() {
                return (new StringBuilder().append(state).append("\t")
                                .append("\t").append(countymonthyear)).toString();
        }


//        public String toString() {
```

```
//                      return (new StringBuilder().append(deptNo).append("\t")
//                                              .append(lNameEmpIDPair)).toString();
//          }

            public void readFields(DataInput dataInput) throws IOException {
                        state = WritableUtils.readString(dataInput);
                        countymonthyear = WritableUtils.readString(dataInput);
               // monthyear = WritableUtils.readString(dataInput);

            }

            public void write(DataOutput dataOutput) throws IOException {
                        WritableUtils.writeString(dataOutput, state);
                        //WritableUtils.writeString(dataOutput, county);
                WritableUtils.writeString(dataOutput, countymonthyear);
            }

        public int compareTo(CompositeKey objKeyPair) {
                        // TODO:
                        /*
                         * Note: This code will work as it stands; but when
CompositeKeyWritable
                         * is used as key in a map-reduce program, it is de-serialized into an
                         * object for comapareTo() method to be invoked;
                         *
                         * To do: To optimize for speed, implement a raw comparator - will
                         * support comparison of serialized representations
                         */
                /*
                        int result = state.compareTo(objKeyPair.state);
               // int finalresult;
                        if (0 == result) {
                                    result =
countymonthyear.compareTo(objKeyPair.countymonthyear);
              }
                        return result;

*/



                if (objKeyPair == null)
                    return 0;
int intcnt = state.compareTo(objKeyPair.state);
```

```
//int retVal = intcnt == 0 ? countymonthyear.compareTo(objKeyPair.countymonthyear) :
intcnt;
//System.out.println("comparetoResult "+retVal);
return intcnt;

            }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCountymonthyear() {
        return countymonthyear;
    }

    public void setCountymonthyear(String countymonthyear) {
        this.countymonthyear = countymonthyear;
    }

//    @Override
//    public void write(DataOutput d) throws IOException {
//        throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
//    }
//
//    @Override
//    public void readFields(DataInput di) throws IOException {
//        throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
//    }
//
}
```

Partitioner class

```
/*
 * 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 pollutionanalysisone;

//import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;

/**
 *
 * @author Manasi
 */
public class Partitioner extends MapReduceBase implements
        org.apache.hadoop.mapred.Partitioner<CompositeKey, Text> {

        @Override
        public int getPartition(CompositeKey key, Text value,
                        int numReduceTasks) {

                return (key.getState().hashCode() % numReduceTasks);
        }{

}
}
```

3. Drought analysis

Driver Class

```
/*
 * 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 draughtanalysis;

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.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 *
 * @author manasi
 */
public class DraughtAnalysis extends Configured implements Tool{

    @Override
    public int run(String[] strings) throws Exception {
        if (strings.length != 2) {
                                System.out.printf("Two parameters are required for
SecondarySortBasicDriver- <input dir> <output dir>\n");
                                return -1;
                        }


        JobConf conf = new JobConf(DraughtAnalysis.class);

      Job job = Job.getInstance(conf, "DraughtAnalysis");
        conf.setJobName("DraughtAnalysis");
      conf.setNumReduceTasks(1);
      conf.setNumMapTasks(3);
      conf.setOutputKeyClass(DraughtCompositeKey.class);
      conf.setOutputValueClass(Text.class);

      conf.setMapperClass(DraughtMapper.class);
      conf.setReducerClass(DraughtReducer.class);

      conf.setPartitionerClass(DraughtPartitioner.class);

      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(strings[0]));
        FileOutputFormat.setOutputPath(conf, new Path(strings[1]));


        JobClient.runJob(conf);

       boolean success = job.waitForCompletion(true);
```

```
                return success ? 0 : 1;
}
public static void main(String[] args) throws Exception {
                int exitCode = ToolRunner.run(new Configuration(),
                                new DraughtAnalysis(), args);
                System.exit(exitCode);
        }
}
```

Mapper Class

```
/*
 * 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 draughtanalysis;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

/**
 *
 * @author manasi
 */
public class DraughtMapper extends MapReduceBase implements
Mapper<LongWritable,Text,DraughtCompositeKey,Text>{

   @Override
   public void map(LongWritable k1, Text v1, OutputCollector<DraughtCompositeKey,
Text> oc, Reporter rprtr) throws IOException {

      if(v1.toString().length()>0){
         String []darray=v1.toString().split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)", -1);
         if(!darray[0].equalsIgnoreCase("releaseDate")){
            String dateattarray[] = darray[10].split("-"); //2000-01-01
```

```
                String
val=darray[4]+"\t"+darray[5]+"\t"+darray[6]+"\t"+darray[7]+"\t"+darray[8]+"\t"+darray[
9];

                try{
                 DraughtCompositeKey c=new
DraughtCompositeKey(darray[3]+"\t"+dateattarray[0] + "\t" + dateattarray[1],"");
                 oc.collect(c, new Text(val));
                }
                catch(ArrayIndexOutOfBoundsException e){
                    try{
                    System.out.println("Error = "+ darray[3]);
                    }
                    catch(ArrayIndexOutOfBoundsException ex){
                        System.out.println("Error still exists = "+darray[0]);
                    }
                }
            }
        }
    }

}
```

Reducer Class

```
/*
 * 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 draughtanalysis;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

/**
 *
 * @author manasi
 */
```

```
public class DraughtReducer extends MapReduceBase implements
   Reducer<DraughtCompositeKey, Text, DraughtCompositeKey, Text >  {

    @Override
    public void reduce(DraughtCompositeKey k2, Iterator<Text> itrtr,
OutputCollector<DraughtCompositeKey, Text> oc, Reporter rprtr) throws IOException {
       // throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
   double none = 0.0;
        double d0 = 0.0;
        double d1 = 0.0;
        double d2 = 0.0;
        double d3 = 0.0;
        double d4 = 0.0;
      long valueCount=0;


        while(itrtr.hasNext()){
            Text value = itrtr.next();
                String v[]=value.toString().split("\t");

                try{
                    none+= Double.parseDouble(v[0]);
                }
                catch(NumberFormatException e){
                    none+=0.0;
                }
                try{
                    d0+= Double.parseDouble(v[1]);
                }
                catch(NumberFormatException e){
                    d0+=0.0;
                }
                try{
                    d1+= Double.parseDouble(v[2]);
                }
                catch(NumberFormatException e){
                    d1+=0.0;
                }try{
                    d2+= Double.parseDouble(v[3]);
                }
                catch(NumberFormatException e){
                    d2+=0.0;
            }try{
               d3+= Double.parseDouble(v[4]);
            }
            catch(NumberFormatException e){
               d3+=0.0;
            }try{
               d4+= Double.parseDouble(v[5]);
            }
            catch(NumberFormatException e){
               d4+=0.0;
            }
             valueCount++;

      }
    none /= valueCount;
      d0 /= valueCount;
      d1 /= valueCount;
      d2 /= valueCount;
      d3 /= valueCount;
      d4 /= valueCount;

      oc.collect(k2, new Text(none+"\t"+d0+"\t"+d1+"\t"+d2+"\t"+d3+"\t"+d4));

   }

}
```

Partitioner

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package draughtanalysis;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;

/**
 *
 * @author manasi
 */
public class DraughtPartitioner extends MapReduceBase implements
```

```
        org.apache.hadoop.mapred.Partitioner<DraughtCompositeKey, Text> {

    @Override
    public int getPartition(DraughtCompositeKey k2, Text v2, int i) {



                    return (k2.getState().hashCode() % i);
            }

// throw new UnsupportedOperationException("Not supported yet."); //To change body
of generated methods, choose Tools | Templates.
    }
```

Composite key

```
/*
 * 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 draughtanalysis;

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 manasi
 */
class DraughtCompositeKey implements Writable,
        WritableComparable<DraughtCompositeKey> {

    private String state;
    private String monthyear;

    public DraughtCompositeKey() {

    }
    DraughtCompositeKey(String state, String monthyear) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
        this.state = state;
        this.monthyear = monthyear;
    }

    @Override
    public String toString() {
        return (new StringBuilder().append(state).append("\t")
                .append("\t").append(monthyear)).toString();
    }

    @Override
    public void write(DataOutput d) throws IOException {
        // throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
        WritableUtils.writeString(d, state);

        WritableUtils.writeString(d, monthyear);
    }

    @Override
    public void readFields(DataInput di) throws IOException {
        //throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
        state = WritableUtils.readString(di);
        monthyear = WritableUtils.readString(di);
    }

    @Override
    public int compareTo(DraughtCompositeKey o) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.

        if (o == null) {
            return 0;
        }
        int intcnt = state.compareTo(o.state);
        int retVal = intcnt == 0 ? monthyear.compareTo(o.monthyear) : intcnt;
//int retVal = intcnt == 0 ? monthyear.compareTo(objKeyPair.countymonthyear) : intcnt;
//System.out.println("comparetoResult "+retVal);
```

```
        return retVal;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getMonthyear() {
        return monthyear;
    }

    public void setMonthyear(String monthyear) {
        this.monthyear = monthyear;
    }

}
```

4. **Pollution- State-code csv mapside join**

```
package pollutionstatecodes;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.*;

import java.io.IOException;
import java.io.IOException;
import java.net.URI;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapreduce.lib.map.WrappedMapper;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.util.*;
import java.io.File;
import java.io.FileNotFoundException;



public class PollutionStateCodes
{

   //Mapper class


   public static class E_EMapper extends MapReduceBase implements
   Mapper<LongWritable ,/*Input key Type */
   Text,                /*Input value Type*/
   Text,                /*Output key Type*/
   Text>        /*Output value Type*/
   {

       private Map<String,String> stateCodes = new HashMap<String,String>();

       @Override
       public void configure(JobConf job)  {
         // Get the cached archives/files
         File f = new File("statecodenames.csv");

         //Path[] files = DistributedCache.getLocalCacheArchives(job.);
           //System.out.println("length of files = "+ files.length);
           //for(Path p : files){
               System.out.println("filenme = "+ f.getName());
               //if(p.getName().equals("statecodenames.csv")){

               try{
                   BufferedReader reader = new BufferedReader(new FileReader(f));
                   String line = reader.readLine();
                   while(line!=null){
                       String [] tokens = line.split(",");
                       stateCodes.put(tokens[1].trim().toUpperCase(), tokens[0].trim());
                       //System.out.println("State name = "+ tokens[1].trim() +" : Value =
"+stateCodes.get(tokens[1].trim()));

                       line = reader.readLine();
                   }
               }
```

```
        catch(FileNotFoundException e){
            System.out.println("File Not found -"+e);

        }
        catch(IOException e){
            System.out.println("IO - " + e);
        }

 }


//Map function
public void map(LongWritable key, Text value,
OutputCollector<Text, Text> output,
Reporter reporter) throws IOException
{

   String line = value.toString();
   //System.out.print(line);
   //String lasttoken = null;
  String mapFileToken[] = line.split("\t");
  String opVal = "";
  String stateName = "";

  String stateCode = "";
  if(line != null){
  if(mapFileToken[0].equals("")){
     opVal =  line;
     stateCode="";
  }
  else{
      stateName = mapFileToken[0].trim();
       stateCode = stateCodes.get(stateName.toUpperCase());

       //tokens[1] = stateCode;

   opVal = StringUtils.join(",", mapFileToken);
  }
  }
  if(stateCode != null)
   output.collect(new Text(stateCode), new Text(line));
  else{
      System.out.println("Statename == "+stateName);
```

```
            System.out.println("stateCode =" + stateCodes.get(stateName));
          }

        }
      }



      //Main function
      public static void main(String args[])throws Exception
      {
        JobConf conf = new JobConf(PollutionStateCodes.class);

        conf.setJobName("Mapside Join");
        conf.setNumReduceTasks(0);
        conf.setNumMapTasks(1);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        conf.setMapperClass(E_EMapper.class);

        //conf.setCombinerClass(E_EReduce.class);
        //conf.setReducerClass(E_EReduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[2]));

        try{
           DistributedCache.addCacheFile(new URI(args[1]), conf);
        }
        catch(Exception e){
           System.out.println(e);
        }

        JobClient.runJob(conf);
      }
    }
```

5. Reduce side 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.
 */
            Driver Class
package reducejoinpollutionstate;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.*;

import java.io.IOException;
import java.io.IOException;
import java.net.URI;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.map.WrappedMapper;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.util.*;

public class ReduceJoinPollutionState {





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

        JobConf conf = new JobConf(ReduceJoinPollutionState.class);

        conf.setJobName("Reduce Side Join");
        conf.setNumReduceTasks(1);
        conf.setNumMapTasks(2);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        conf.setMapperClass(PollutionMapper.class);

        //conf.setCombinerClass(E_EReduce.class);
        conf.setReducerClass(E_EReduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        //FileInputFormat.setInputPaths(conf, new Path(args[1]));
        MultipleInputs.addInputPath(conf, new Path(args[0]), TextInputFormat.class,
DroughtMapper.class);
        MultipleInputs.addInputPath(conf, new Path(args[1]), TextInputFormat.class,
PollutionMapper.class);

        FileOutputFormat.setOutputPath(conf, new Path(args[3]));
try{
          DistributedCache.addCacheFile(new URI(args[2]), conf);
      }
      catch(Exception e){
          System.out.println(e);
      }


        JobClient.runJob(conf);
    }
}
```

DroughtMapper

```
/*
 * 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 reducejoinpollutionstate;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

/**
 *
 * @author manasi
 */

 public class DroughtMapper extends MapReduceBase implements
        Mapper<LongWritable,/*Input key Type */ Text, /*Input value Type*/ Text,
/*Output key Type*/ Text> /*Output value Type*/ {

    private Map<String, String> stateCodes = new HashMap<String, String>();

    //@Override
    public void configure(JobConf job) {
        // Get the cached archives/files
        File f = new File("statecodenames.csv");



        try {
            BufferedReader reader = new BufferedReader(new FileReader(f));
            String line = reader.readLine();
            while (line != null) {
                String[] tokens = line.split(",");
                stateCodes.put(tokens[0].trim().toUpperCase(), tokens[1].trim());
                //System.out.println("State code = " + tokens[0].trim() + " : Value = " +
stateCodes.get(tokens[0].trim()));

                line = reader.readLine();
            }
        } catch (FileNotFoundException e) {
            System.out.println("File Not found -" + e);

        } catch (IOException e) {
            System.out.println("IO - " + e);
        }

    }

    //Map function
    public void map(LongWritable key, Text value,
            OutputCollector<Text, Text> output,
            Reporter reporter) throws IOException {
        String line = value.toString();
        String mapFileToken[] = value.toString().split("\t");

        // String opVal = "";
        if (line != null) {

            String stateName = "";
            String stateCode = mapFileToken[0];
            if (mapFileToken[0].equals("")) {
                //opVal = line;
                stateCode = "";
            } else {
                stateCode = mapFileToken[0].trim();
                stateName = stateCodes.get(stateCode.toUpperCase());
                if (stateName!=null && stateCode != null) {

                    String droughtkey = mapFileToken[0] + "\t" + stateName + "\t" +
mapFileToken[1] + "\t" + mapFileToken[2];

                    double[] droughtValues = new double[6];

                    int j = 5;

                    for (int i = 0; i < 6; i++) {

                        try {
                            droughtValues[i] = Double.parseDouble(mapFileToken[j]);
                            j++;
                        } catch (NumberFormatException e) {
                            droughtValues[i] = 0.0;
                        }

                    }
                    double maxdrought = 0.0;
                    int maxdroughtindex = 0;
                    if (droughtValues[0] < 100.0) {
                        //maxdroughtindex = 1;
                        int i = 1;
                        while (i < 6) {
                            if (droughtValues[i] >= maxdrought) {
                                maxdroughtindex = i;
                                maxdrought = droughtValues[i];
                            }
                            i++;
                        }
                    } else {
                        maxdrought = droughtValues[0];
                        maxdroughtindex = 0;

                    }

                    String drtype = maxdroughtindex == 0 ? "None" : "D" +
(maxdroughtindex);

                    //System.out.println("Draought mapper key = "+droughtkey+"\nVaue = "
+ "draught\t" + maxdrought + "\t" + drtype);

                    output.collect(new Text(droughtkey), new Text("draught\t" + maxdrought
+ "\t" + drtype));

                }
                else {
                    System.out.println("StateCode == " + stateCode);
                    System.out.println("StateName =" + stateCodes.get(stateCode));
                }
                //
            }

        }
    }
}
```

PollutionMapper

```
/*
 * 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 reducejoinpollutionstate;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

/**
 *
 * @author manasi
 */



   //Mapper class
   public class PollutionMapper extends MapReduceBase implements
         Mapper<LongWritable,/*Input key Type */ Text, /*Input value Type*/ Text,
/*Output key Type*/ Text> /*Output value Type*/ {

      //key --- code state year  value -----no,so,co,o3
      //Map function
      public void map(LongWritable key, Text value,
            OutputCollector<Text, Text> output,
            Reporter reporter) throws IOException {
         String line = value.toString();
         String tokens[] = line.split("\t");
         // String linePart[] = tokens[1].split(",");
         String keyCode = tokens[0] + "\t" + tokens[1]+ "\t" + tokens[3] + "\t" + tokens[4];
         //System.out.println("key = " + keyCode);
         //System.out.println("value = " + "poll\t" + tokens[7] + "\t" + tokens[8] + "\t" +
tokens[9] + "\t" + tokens[10]);
```

```
            output.collect(new Text(keyCode), new Text("poll\t" + tokens[7] + "\t" +
tokens[8] + "\t" + tokens[9] + "\t" + tokens[10   ]));
        }
    }
```

Reducer class

```
/*
 * 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 reducejoinpollutionstate;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

/**
 *
 * @author manasi
 */
public class E_EReduce extends MapReduceBase implements
   Reducer< Text, Text, Text, Text > {


    //Reduce function
    public void reduce(Text key, Iterator<Text> values,
            OutputCollector<Text, Text> output, Reporter reporter) throws IOException {

        double maxPollutant = 0.0;
        long count = 0;
        int type = -1;
        String pollutantType = "";
        Double maxdrought = 0.0;
        String drtype = "";
        String droughtType = "";
        String droughtValue = "";
        double pollutants[] = new double[4];
        //System.out.println("key = "+ key.toString());
        while (values.hasNext()) {
            //output.collect(key, new Text("KEY = " + key.toString() + " \nvalues = " +
values.next().toString()));

            String parts[] = values.next().toString().split(("\t"));

            // double pollutants[] = new double[4];
            if (parts[0].equals("poll")) {
                count++;
                for (int i = 0; i < 4; i++) {

                    try {
                        pollutants[i] += Double.parseDouble(parts[i + 1]);

                    } catch (NumberFormatException e) {
                        pollutants[i] = 0.0;
                    }

                }

            } else if (parts[0].equals("draught")) {
                try {
                    maxdrought = Double.parseDouble(parts[1]);
                } catch (Exception e) {
                    maxdrought = -1.0;
                }
                drtype = parts[2];
            }

        }

        for (int i = 0; i < 4; i++) {
            if(count!=0){
            pollutants[i] /= count;
            }
            if (pollutants[i] >= maxPollutant) {
                maxPollutant = pollutants[i];
                type = i;
            }

        }

        if(maxPollutant == 0.0){
                type = -1;
            }

        if(type==0) {
            pollutantType = "NO2";
        }
        else if(type==1) {
            pollutantType = "O3";
        }
        else if(type==2) {
            pollutantType = "SO2";
        }
        else if(type==3) {
            pollutantType = "CO";
        }
        else{
            pollutantType = "-1";
        }

                output.collect(key, new Text(maxPollutant + "\t" + pollutantType + "\t" +
maxdrought + "\t" + drtype));
        }

    }
```