ENGINEERING BIG DATA SYSTEMS

ROAD ACCIDENTS IN THE USA

Satwik Kashyap
kashyap.sa@northeastern.edu

Introduction

This is a countrywide auto collision dataset, which covers 49 states of the USA. The mishap information is gathered from February 2016 to June 2020, utilizing two APIs that give streaming traffic occurrence (or occasion) information. These APIs broadcast traffic information caught by an assortment of substances, for example, the US and state divisions of transportation, law implementation offices, traffic cameras, and traffic sensors inside the street organizations. Right now, there are about 3.5 million mishap records in this dataset. In this project, we are focused on performing Map Reduce with Java in a single standalone cluster on virtual machine

Data Set

This dataset has been collected in real-time, using multiple Traffic APIs. Currently, it contains accident data that are collected from February 2016 to June 2020 for the Contiguous United States.

The columns we have considered for the map reduce analysis are listed below:

ID	Distance(mi)	Timezone	Precipitation(in)	Station
Source	Description	Airport_Code	Weather_Condition	Stop
TMC	Number	Weather_Timestamp	Amenity	Traffic_Calming
Severity	Street	Temperature(F)	Bump	Traffic_Signal
Start_Time	Side	Wind_Chill(F)	Crossing	Turning_Loop
End_Time	City	Humidity(%)	Give_Way	Sunrise_Sunset
Start_Lat	County	Pressure(in)	Junction	Civil_Twilight
Start_Lng	State	Visibility(mi)	No_Exit	Nautical_Twilight
End_Lat	Zipcode	Wind_Direction	Railway	Astronomical_Twilight
End_Lng	Country	Wind_Speed(mph)	Roundabout	

HADOOP ANALYSIS

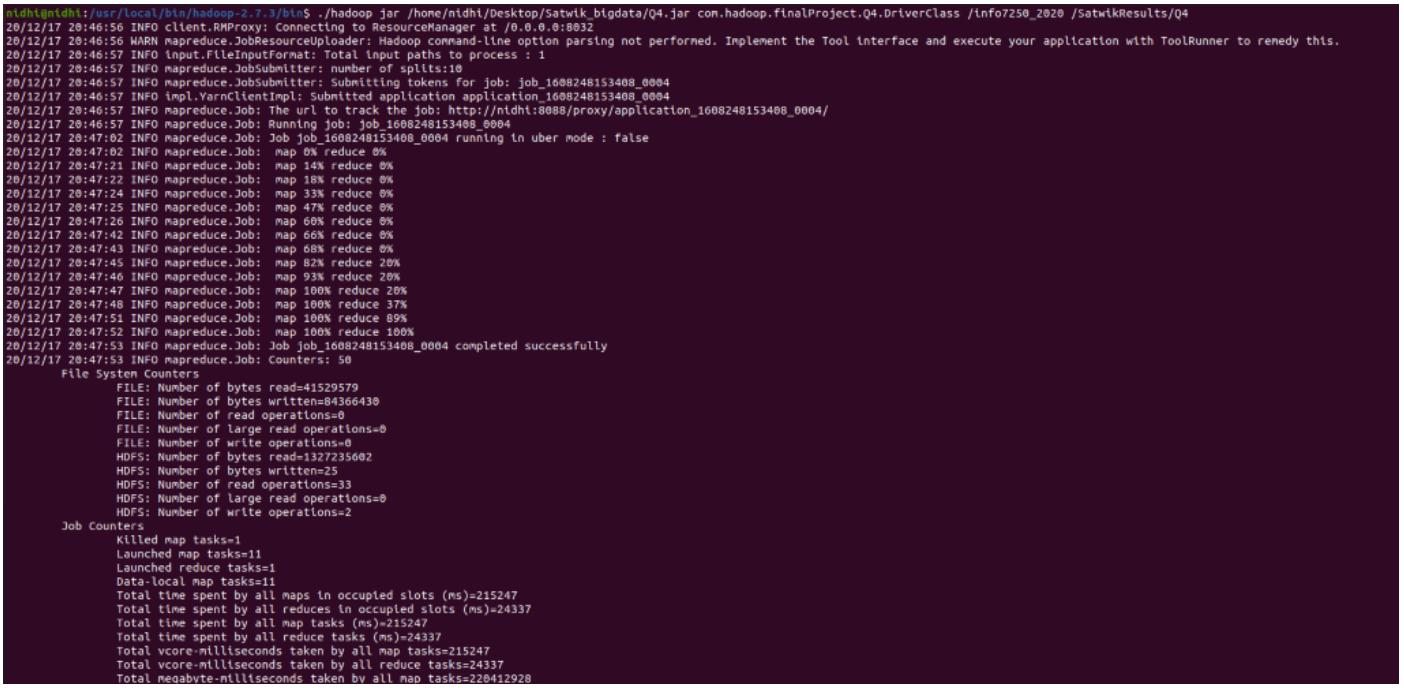
Importing the Data into HDFS from Local

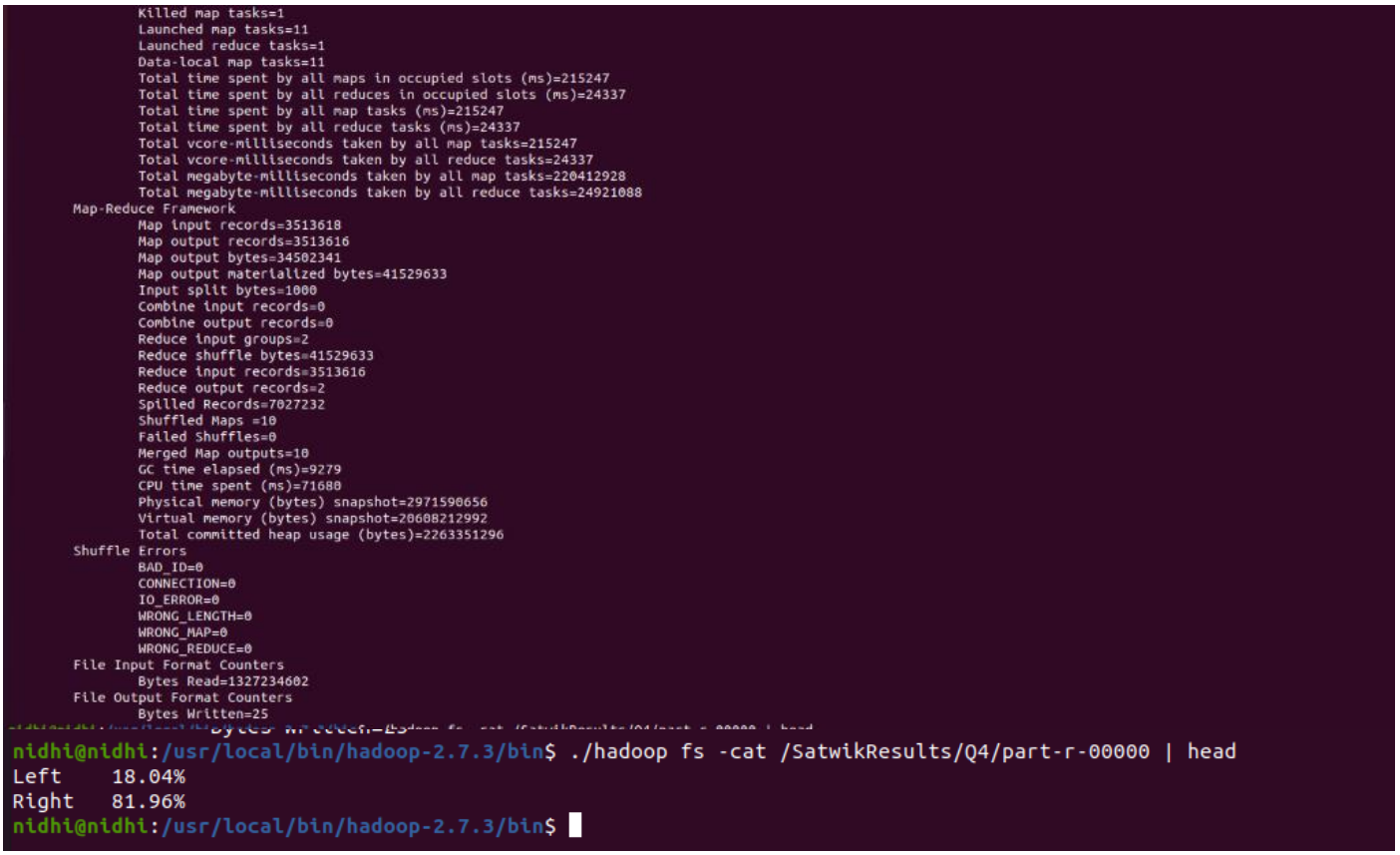
./hdfs dfs -copyFromLocal /home/nidhi/Desktop/Satwik_bigdata/usaccident/US_Accidents_June20.csv /info7250_2020

-rw-r--r--	nidhi	supergroup	1.24 GB	12/17/2020, 5:56:25 PM	1	128 MB	info7250_2020
drwxr-xr-x	nidhi	supergroup	0 B	10/15/2020, 10:11:37 PM	0	0 B	numbers
-rw-r--r--	nidhi	supergroup	8.54 KB	12/17/2020, 3:17:16 PM	1	128 MB	prod
-rw-r--r--	nidhi	supergroup	11.65 KB	12/17/2020, 3:03:49 PM	1	128 MB	product
-rw-r--r--	nidhi	supergroup	9.4 KB	12/17/2020, 2:55:45 PM	1	128 MB	products
-rw-r--r--	nidhi	supergroup	68 B	12/17/2020, 3:22:41 PM	1	128 MB	productzz
drwx------	nidhi	supergroup	0 B	10/31/2020, 2:28:41 AM	0	0 B	tmp

Q1 : Total Number of accidents on Left and Right side

It is a Map Reduce program to calculate the total number of accident on each side of the road





MAPPER

```
package com.mycompany.assignment_4.Q1TotalNumberofaccidentsonLeftandRightside;

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

   Text sideOfRoad = new Text();
   IntWritable one = new IntWritable(1);

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

      String []tokens =  value.toString().split(",");
      if(!tokens[0].equals("ID")) {
         String side = tokens[14]; // 14th column in the data
         if(!side.equals(" ")) {

            if(side.equals("L")) {
               side = "Left";
            } else {
               side = "Right";
            }

            sideOfRoad.set(side);
            context.write(sideOfRoad, one);
         }
      }
```

```
    }
}
```

REDUCER

```
package com.mycompany.assignment_4.Q1TotalNumberofaccidentsonLeftandRightside;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

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

    Text percentage = new Text();

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

        int count = 0;

        for(IntWritable v : values) {
            count += v.get();
        }

        int total = 3513617;  // total number of records in the data

        double perc = ((double) count / total) * 100;
        percentage.set(String.format("%.2f", perc) + "%");

        context.write(key, percentage);
    }
}
```

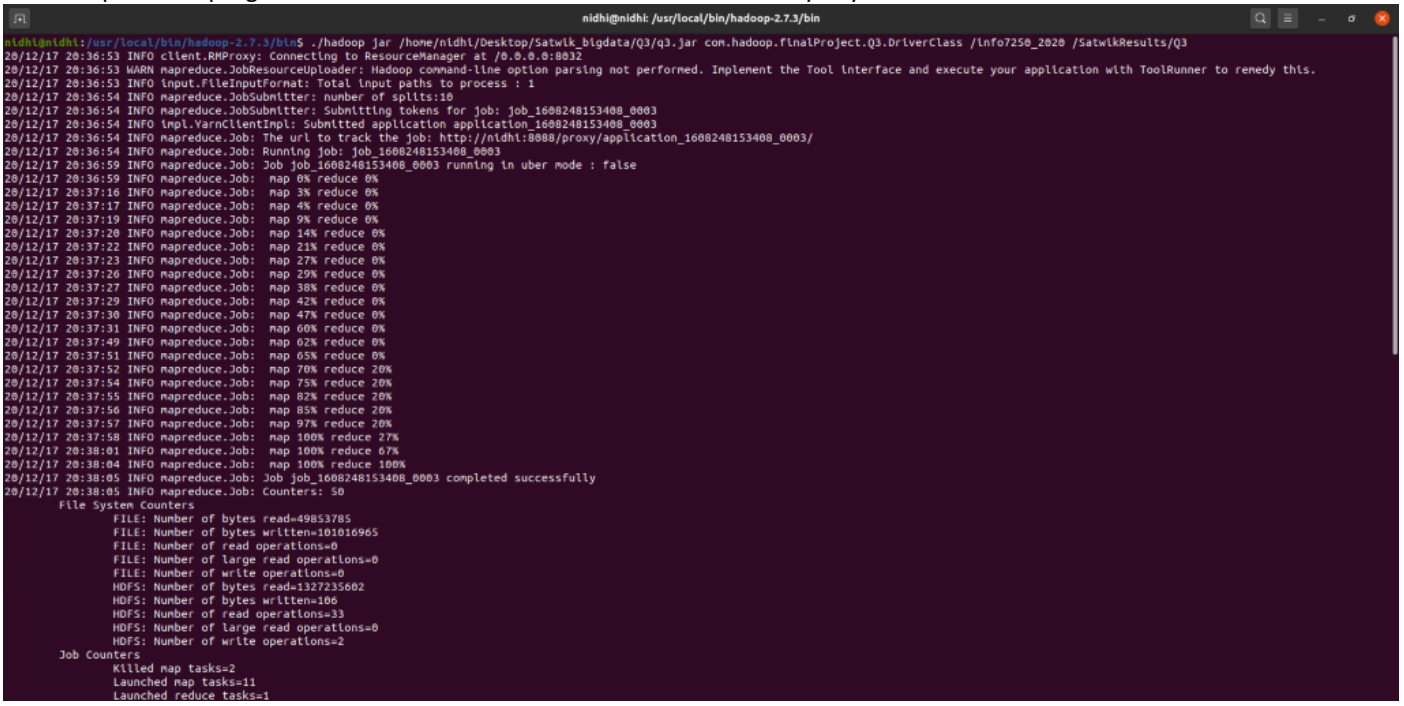
DRIVER:

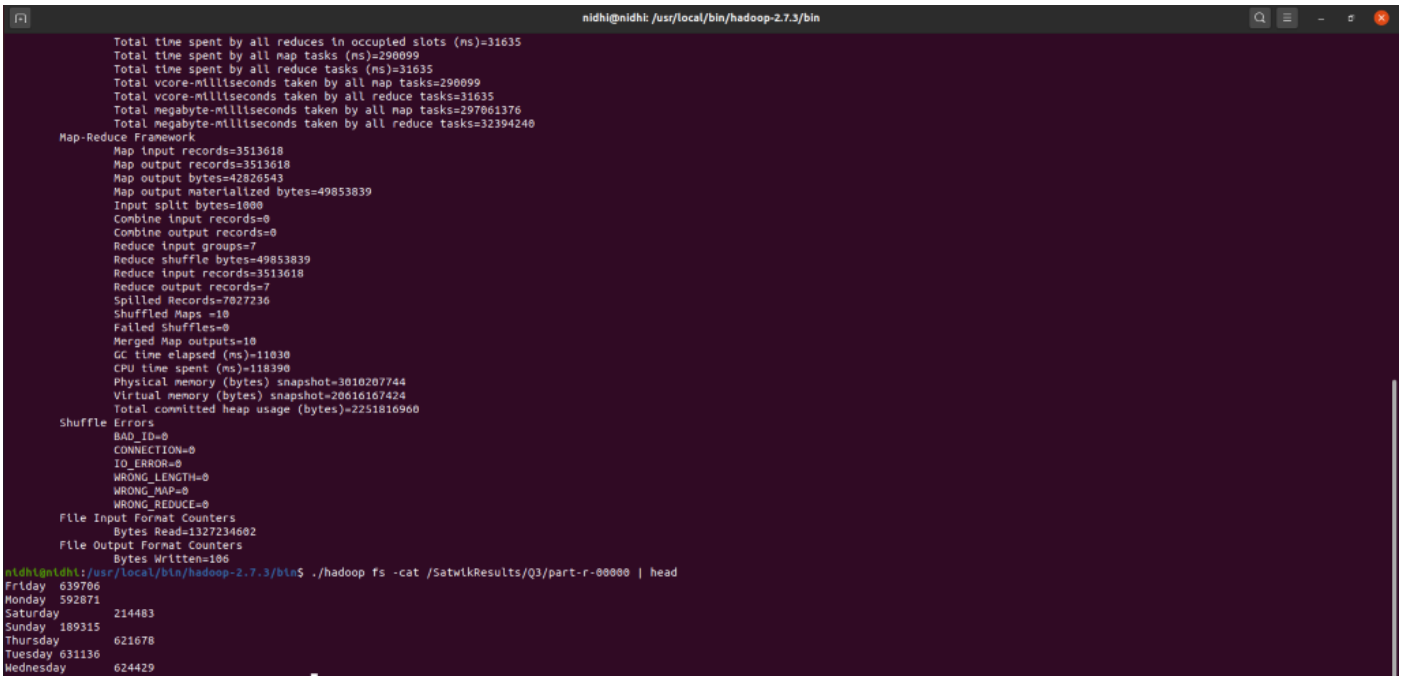
```
package com.mycompany.assignment_4.Q1TotalNumberofaccidentsonLeftandRightside;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;


// Defining a Driver Class
public class DriverClass {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
        Configuration conf = new Configuration();

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

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
    // Driver Class
    job.setJarByClass(DriverClass.class);

    job.setInputFormatClass(TextInputFormat.class);

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

    // Mapper Class
    job.setMapperClass(MapperClass.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);

    // Reducer Class
    job.setReducerClass(ReducerClass.class);

    // Using it to create one reducer
    job.setNumReduceTasks(1);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
       fs.delete(outDir, true);
    }
    // SUbmitting the job and waiting for it to complete
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

}
```

Q2. Weekday Analysis : Total Number of accidents

It is a Map Reduce program to calculate the total number of accident every day in a week





MAPPER

```
package com.mycompany.assignment_4.Q2WeekdayAnalysisTotalNumberofaccidents;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

   public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
      Configuration conf = new Configuration();

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

      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      //job.getConfiguration().set("mapreduce.output.textoutputformat.recordseparator","\t");

      // Driver Class
      job.setJarByClass(DriverClass.class);

      job.setInputFormatClass(TextInputFormat.class);
```

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

    // Comparator
    job.setSortComparatorClass(ComparatorClass.class);

    // Mapper
    job.setMapperClass(MapperClass.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);

    // Reducer
    job.setReducerClass(ReducerClass.class);

    // Create only 1 reducer
    job.setNumReduceTasks(1);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
      fs.delete(outDir, true);
    }


    // Submit the job, then poll for progress until the job is complete
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

}
```

REDUCER

```
package com.mycompany.assignment_4.Q2WeekdayAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<Text, IntWritable, Text, LongWritable> {

  LongWritable sum = new LongWritable();

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

    long count = 0;

    for(IntWritable v : values) {
      count += v.get();
    }

    sum.set(count);

    context.write(key, sum);
```

```
    }
}
```

COMPARATOR

```
package com.mycompany.assignment_4.Q2WeekdayAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparator;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class ComparatorClass extends WritableComparator {

    SimpleDateFormat parser = new SimpleDateFormat("EEEE");

    protected ComparatorClass() {
        super(Text.class, true);
    }

    @Override
    public int compare(Object a, Object b) {
        Text m1 = (Text) a;
        Text m2 = (Text) b;

        Date d1 = new Date();
        Date d2 = new Date();

        try {
            d1 = parser.parse(m1.toString());
            d2 = parser.parse(m2.toString());
        } catch (ParseException e) {
            e.printStackTrace();
        }

        return d1.compareTo(d2);
    }
}
```

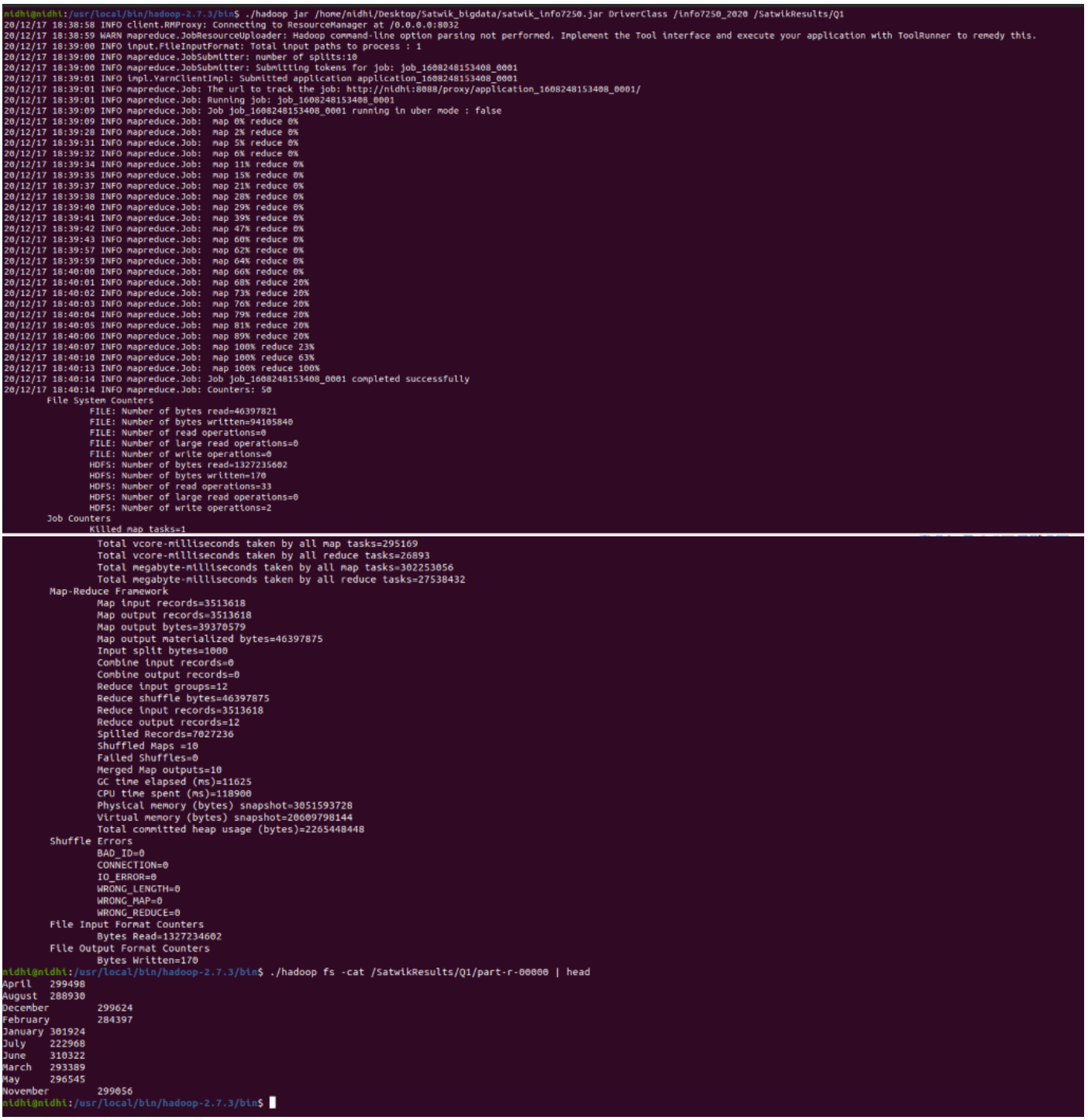
DRIVER

```
package com.mycompany.assignment_4.Q2WeekdayAnalysisTotalNumberofaccidents;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "NumberOfAccidentsPerWeekday");
```

```
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    //job.getConfiguration().set("mapreduce.output.textoutputformat.recordseparator","\t");

    // Driver Class
    job.setJarByClass(DriverClass.class);

    job.setInputFormatClass(TextInputFormat.class);

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

    // Comparator
    job.setSortComparatorClass(ComparatorClass.class);

    // Mapper
    job.setMapperClass(MapperClass.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);

    // Reducer
    job.setReducerClass(ReducerClass.class);

    // Create only 1 reducer
    job.setNumReduceTasks(1);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
       fs.delete(outDir, true);
    }


    // Submit the job, then poll for progress until the job is complete
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

}
```

Q3. Monthly Analysis : Total Number of accidents

It is a Map Reduce program to calculate the total number of accident in a given month



MAPPER

```
package com.mycompany.assignment_4.Q3MonthlyAnalysisTotalNumberofaccidents;

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;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MapperClass extends Mapper<LongWritable, Text, Text, IntWritable> {
```

```
  Text month = new Text();
  IntWritable one = new IntWritable(1);

  SimpleDateFormat parser = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
  SimpleDateFormat format = new SimpleDateFormat("MMMMM"); //taking the month name

  @Override
  protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

    String []tokens =  value.toString().split(",");
    Date date = new Date();

    String timeStamp = tokens[4];

    try {
       date = parser.parse(timeStamp);
       } catch (ParseException e) {
       }


    month.set(format.format(date));

    context.write(month, one);

  }
}
```

REDUCER

```
package com.mycompany.assignment_4.Q3MonthlyAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<Text, IntWritable, Text, LongWritable> {

  LongWritable sum = new LongWritable();

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

    long count = 0;

    for(IntWritable v : values) {
       count += v.get();
    }

    sum.set(count);

    context.write(key, sum);
  }
}
```

DRIVER

```
package com.mycompany.assignment_4.Q3MonthlyAnalysisTotalNumberofaccidents;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

   public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
      Configuration conf = new Configuration();

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

      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      job.getConfiguration().set("mapreduce.output.textoutputformat.recordseparator","\t");

      // Driver Class
      job.setJarByClass(DriverClass.class);

      job.setInputFormatClass(TextInputFormat.class);

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

      // Comparator
      job.setSortComparatorClass(ComparatorClass.class);

      // Mapper
      job.setMapperClass(MapperClass.class);

      job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(IntWritable.class);

      // Reducer
      job.setReducerClass(ReducerClass.class);

      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      // Create only 1 reducer
      job.setNumReduceTasks(1);

      FileSystem fs = FileSystem.get(job.getConfiguration());

      if(fs.exists(outDir)){
         fs.delete(outDir, true);
      }


      // Submit the job, then poll for progress until the job is complete
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }

}
```

COMPARATOR

```
package com.mycompany.assignment_4.Q3MonthlyAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class ComparatorClass extends WritableComparator {

   SimpleDateFormat parser = new SimpleDateFormat("MMMMM");

   protected ComparatorClass() {
      super(Text.class, true);
   }

   @Override
   public int compare(Object a, Object b) {
      Text m1 = (Text) a;
      Text m2 = (Text) b;

      Date d1 = new Date();
      Date d2 = new Date();

      try {
         d1 = parser.parse(m1.toString());
         d2 = parser.parse(m2.toString());
      } catch (ParseException e) {
         e.printStackTrace();
      }

      return d1.compareTo(d2);
   }
}
```

Q4. State Analysis : Total Number of accidents – top 10 prone

It is a Map Reduce program to calculate the top 10 accident prone states

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop jar /home/nidhi/Desktop/Satwik_bigdata/Q6.jar com.hadoop.finalProject.Q6.DriverClass /info7250_2020 /SatwikResults/Q6
20/12/17 21:01:19 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/12/17 21:01:20 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/12/17 21:01:20 INFO input.FileInputFormat: Total input paths to process : 1
20/12/17 21:01:20 INFO mapreduce.JobSubmitter: number of splits:10
20/12/17 21:01:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1608248153408_0006
20/12/17 21:01:21 INFO impl.YarnClientImpl: Submitted application application_1608248153408_0006
20/12/17 21:01:21 INFO mapreduce.Job: The url to track the job: http://nidhi:8088/proxy/application_1608248153408_0006/
20/12/17 21:01:21 INFO mapreduce.Job: Running job: job_1608248153408_0006
20/12/17 21:01:27 INFO mapreduce.Job: Job job_1608248153408_0006 running in uber mode : false
20/12/17 21:01:27 INFO mapreduce.Job:  map 0% reduce 0%
20/12/17 21:01:46 INFO mapreduce.Job:  map 17% reduce 0%
20/12/17 21:01:47 INFO mapreduce.Job:  map 32% reduce 0%
20/12/17 21:01:49 INFO mapreduce.Job:  map 35% reduce 0%
20/12/17 21:01:50 INFO mapreduce.Job:  map 42% reduce 0%
20/12/17 21:01:52 INFO mapreduce.Job:  map 47% reduce 0%
20/12/17 21:01:53 INFO mapreduce.Job:  map 60% reduce 0%
20/12/17 21:02:10 INFO mapreduce.Job:  map 67% reduce 0%
20/12/17 21:02:11 INFO mapreduce.Job:  map 74% reduce 0%
20/12/17 21:02:12 INFO mapreduce.Job:  map 87% reduce 0%
20/12/17 21:02:13 INFO mapreduce.Job:  map 90% reduce 0%
20/12/17 21:02:14 INFO mapreduce.Job:  map 93% reduce 17%
20/12/17 21:02:15 INFO mapreduce.Job:  map 100% reduce 17%
20/12/17 21:02:17 INFO mapreduce.Job:  map 100% reduce 23%
20/12/17 21:02:20 INFO mapreduce.Job:  map 100% reduce 27%
```

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -tail /SatwikResults/Q6/part-r-00000 | head
California 329284
Texas 258002
Florida 258002
South Carolina 173277
North Carolina 165958
New York 160817
Pennsylvania 106787
Illinois 99692
Virginia 96075
Michigan 95983nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$
```

MAPPER

```
package com.mycompany.assignment_4.Q4StateAnalysisTotalNumberofaccidentstop10prone;

import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

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

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

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

REDUCER

```
package com.mycompany.assignment_4.Q4StateAnalysisTotalNumberofaccidentstop10prone;

import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.Comparator;
import java.util.TreeMap;

public class ReducerClass extends Reducer<NullWritable, Text, NullWritable, Text> {

   static class DescOrder implements Comparator<Integer> {

      @Override
      public int compare(Integer o1, Integer o2) {
         return o2.compareTo(o1);
      }
   }

   private TreeMap<Integer, Text> countMap = new TreeMap<>(new DescOrder());

   @Override
   protected void reduce(NullWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
      for (Text value : values) {

         String[] tokens = value.toString().split("\t");
         int countOfStates = Integer.parseInt(tokens[1]);

         countMap.put(countOfStates, new Text(value));
```

```
        if (countMap.size() > 10)
            countMap.remove(countMap.lastKey());
        }

        for (Text t : countMap.values())
            context.write(NullWritable.get(), t);
    }
}
```

DRIVER

```
package com.mycompany.assignment_4.Q4StateAnalysisTotalNumberofaccidentstop10prone;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DriverClass.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));


        job.setInputFormatClass(TextInputFormat.class);

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

        job.setMapOutputKeyClass(NullWritable.class);
        job.setMapOutputValueClass(Text.class);


        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(Text.class);

        // Create only 1 reducer
        job.setNumReduceTasks(1);

        FileSystem fs = FileSystem.get(job.getConfiguration());
```

```
    if(fs.exists(outDir)){
      fs.delete(outDir, true);
    }

    // Submit the job, then poll for progress until the job is complete
    System.exit(job.waitForCompletion(true)?0:1);

  }
}
```

Q5. Hourly Analysis : Total Number of accidents

It is a Map Reduce program to calculate the total number of accident in a given hour

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop jar /home/nidhi/Desktop/Satwik_bigdata/Q2/BigDataFinalProject.jar com.hadoop.finalProject.Q2.DriverClass /info7250_2020 /SatwikResults/Q2
20/12/17 20:28:35 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/12/17 20:28:35 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/12/17 20:28:35 INFO input.FileInputFormat: Total input paths to process : 1
20/12/17 20:28:36 INFO mapreduce.JobSubmitter: number of splits:10
20/12/17 20:28:36 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1608248153408_0002
20/12/17 20:28:36 INFO impl.YarnClientImpl: Submitted application application_1608248153408_0002
20/12/17 20:28:36 INFO mapreduce.Job: The url to track the job: http://nidhi:8088/proxy/application_1608248153408_0002/
20/12/17 20:28:36 INFO mapreduce.Job: Running job: job_1608248153408_0002
20/12/17 20:28:41 INFO mapreduce.Job: Job job_1608248153408_0002 running in uber mode : false
20/12/17 20:28:41 INFO mapreduce.Job:  map 0% reduce 0%
20/12/17 20:29:00 INFO mapreduce.Job:  map 3% reduce 0%
20/12/17 20:29:01 INFO mapreduce.Job:  map 9% reduce 0%
20/12/17 20:29:03 INFO mapreduce.Job:  map 12% reduce 0%
20/12/17 20:29:04 INFO mapreduce.Job:  map 23% reduce 0%
20/12/17 20:29:06 INFO mapreduce.Job:  map 26% reduce 0%
20/12/17 20:29:07 INFO mapreduce.Job:  map 39% reduce 0%
20/12/17 20:29:08 INFO mapreduce.Job:  map 46% reduce 0%
20/12/17 20:29:09 INFO mapreduce.Job:  map 60% reduce 0%
20/12/17 20:29:23 INFO mapreduce.Job:  map 62% reduce 0%
20/12/17 20:29:25 INFO mapreduce.Job:  map 64% reduce 0%
20/12/17 20:29:26 INFO mapreduce.Job:  map 70% reduce 0%
20/12/17 20:29:27 INFO mapreduce.Job:  map 70% reduce 20%
20/12/17 20:29:28 INFO mapreduce.Job:  map 73% reduce 20%
20/12/17 20:29:29 INFO mapreduce.Job:  map 82% reduce 20%
20/12/17 20:29:31 INFO mapreduce.Job:  map 88% reduce 20%
20/12/17 20:29:32 INFO mapreduce.Job:  map 97% reduce 20%
20/12/17 20:29:33 INFO mapreduce.Job:  map 100% reduce 30%
20/12/17 20:29:36 INFO mapreduce.Job:  map 100% reduce 67%
20/12/17 20:29:38 INFO mapreduce.Job:  map 100% reduce 100%
20/12/17 20:29:39 INFO mapreduce.Job: Job job_1608248153408_0002 completed successfully
20/12/17 20:29:39 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=31622568
                FILE: Number of bytes written=64556269
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1327235602
                HDFS: Number of bytes written=230
                HDFS: Number of read operations=33
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=249094
```

```
                Total vcore-milliseconds taken by all map tasks=249094
                Total vcore-milliseconds taken by all reduce tasks=27439
                Total megabyte-milliseconds taken by all map tasks=255072256
                Total megabyte-milliseconds taken by all reduce tasks=28097536
        Map-Reduce Framework
                Map input records=3513618
                Map output records=3513618
                Map output bytes=24595326
                Map output materialized bytes=31622622
                Input split bytes=1000
                Combine input records=0
                Combine output records=0
                Reduce input groups=24
                Reduce shuffle bytes=31622622
                Reduce input records=3513618
                Reduce output records=24
                Spilled Records=7027236
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=10512
                CPU time spent (ms)=100180
                Physical memory (bytes) snapshot=2968145920
                Virtual memory (bytes) snapshot=20614549504
                Total committed heap usage (bytes)=2197815296
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1327234602
        File Output Format Counters
                Bytes Written=230
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -cat /SatwikResults/Q2/part-r-00000 | head
00      29189
01      22401
02      23120
03      22702
04      64706
05      99244
06      198679
07      316352
08      326257
09      202664
```

MAPPER

```
package com.mycompany.assignment_4.Q5HourlyAnalysisTotalNumberofaccidents;

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;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

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

   Text hour = new Text();
   IntWritable one = new IntWritable(1);

   SimpleDateFormat parser = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
   SimpleDateFormat format = new SimpleDateFormat("HH");

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

      String []tokens =  value.toString().split(",");
      Date date = new Date();

      if(!tokens[0].equals("ID")) {
         String timeStamp = tokens[4];

         try {
            date = parser.parse(timeStamp);
         } catch (ParseException e) {
            e.printStackTrace();
         }
      }

      hour.set(format.format(date));

      context.write(hour, one);

   }
}
```

REDUCER

```
package com.mycompany.assignment_4.Q5HourlyAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<Text, IntWritable, Text, LongWritable> {

   LongWritable sum = new LongWritable();
```

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

        long count = 0;

        for(IntWritable v : values) {
            count += v.get();
        }

        sum.set(count);

        context.write(key, sum);
    }
}
```

COMPARATOR

```
package com.mycompany.assignment_4.Q5HourlyAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparator;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class ComparatorClass extends WritableComparator {

    SimpleDateFormat parser = new SimpleDateFormat("HH");

    protected ComparatorClass() {
        super(Text.class, true);
    }

    @Override
    public int compare(Object a, Object b) {
        Text m1 = (Text) a;
        Text m2 = (Text) b;

        Date d1 = new Date();
        Date d2 = new Date();

        try {
            d1 = parser.parse(m1.toString());
            d2 = parser.parse(m2.toString());
        } catch (ParseException e) {
            e.printStackTrace();
        }

        return d1.compareTo(d2);
    }
}
```

DRIVER

```
package com.mycompany.assignment_4.Q5HourlyAnalysisTotalNumberofaccidents;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

   public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
      Configuration conf = new Configuration();

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

      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      job.getConfiguration().set("mapreduce.output.textoutputformat.recordseparator","\t");

      // Driver Class
      job.setJarByClass(DriverClass.class);

      job.setInputFormatClass(TextInputFormat.class);

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

      // Comparator
      job.setSortComparatorClass(ComparatorClass.class);

      // Mapper
      job.setMapperClass(MapperClass.class);

      job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(IntWritable.class);

      // Reducer
      job.setReducerClass(ReducerClass.class);

      // Create only 1 reducer
      job.setNumReduceTasks(1);

      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      FileSystem fs = FileSystem.get(job.getConfiguration());
      if(fs.exists(outDir)){
         fs.delete(outDir, true);
      }


      // Submit the job, then poll for progress until the job is complete
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }

}
```

Q6. State Analysis : Total Number of accidents

It is a Map Reduce program to calculate the total number of accident per state in the USA

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop jar /home/nidhi/Desktop/Satwik_bigdata/Q5.jar com.hadoop.finalProject.Q5.DriverClass /info7250_2020 /SatwikResults/Q5
20/12/17 20:53:09 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/12/17 20:53:09 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/12/17 20:53:10 INFO input.FileInputFormat: Total input paths to process : 1
20/12/17 20:53:10 INFO mapreduce.JobSubmitter: number of splits:10
20/12/17 20:53:11 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1608248153408_0005
20/12/17 20:53:11 INFO impl.YarnClientImpl: Submitted application application_1608248153408_0005
20/12/17 20:53:11 INFO mapreduce.Job: The url to track the job: http://nidhi:8088/proxy/application_1608248153408_0005/
20/12/17 20:53:11 INFO mapreduce.Job: Running job: job_1608248153408_0005
20/12/17 20:53:18 INFO mapreduce.Job: Job job_1608248153408_0005 running in uber mode : false
20/12/17 20:53:18 INFO mapreduce.Job:  map 0% reduce 0%
20/12/17 20:53:36 INFO mapreduce.Job:  map 8% reduce 0%
20/12/17 20:53:37 INFO mapreduce.Job:  map 23% reduce 0%
20/12/17 20:53:39 INFO mapreduce.Job:  map 26% reduce 0%
20/12/17 20:53:40 INFO mapreduce.Job:  map 37% reduce 0%
20/12/17 20:53:41 INFO mapreduce.Job:  map 40% reduce 0%
20/12/17 20:53:42 INFO mapreduce.Job:  map 60% reduce 0%
20/12/17 20:53:59 INFO mapreduce.Job:  map 74% reduce 0%
20/12/17 20:54:02 INFO mapreduce.Job:  map 87% reduce 20%
20/12/17 20:54:03 INFO mapreduce.Job:  map 100% reduce 20%
20/12/17 20:54:06 INFO mapreduce.Job:  map 100% reduce 67%
20/12/17 20:54:08 INFO mapreduce.Job:  map 100% reduce 100%
20/12/17 20:54:08 INFO mapreduce.Job: Job job_1608248153408_0005 completed successfully
20/12/17 20:54:08 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=55504286
                FILE: Number of bytes written=112315866
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1327235602
                HDFS: Number of bytes written=761
                HDFS: Number of read operations=33
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=218410
                Total time spent by all reduces in occupied slots (ms)=22500
                Total time spent by all map tasks (ms)=218410
                Total time spent by all reduce tasks (ms)=22500
                Total vcore-milliseconds taken by all map tasks=218410
                Total vcore-milliseconds taken by all reduce tasks=22500
                Total megabyte-milliseconds taken by all map tasks=223651840
                Total megabyte-milliseconds taken by all reduce tasks=23040000
```

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -cat /SatwikResults/Q5/part-r-00000 | head
        1
Alabama 44625
Arizona 78584
Arkansas        2012
California      816825
Colorado        49731
Connecticut     25901
Delaware        5739
District Of Columbia    4820
Florida 258002
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$
```

MAPPER

```
package com.mycompany.assignment_4.Q6StateAnalysisTotalNumberofaccidents;

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

   Text state = new Text();
   IntWritable one = new IntWritable(1);

   StateMap statesMap = new StateMap();

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

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

     String stateString = "";
```

```
    if(!tokens[0].equals("ID")) {
      stateString = statesMap.getStateFromAbbr(tokens[17]);
    }

    state.set(stateString);


    context.write(state, one);

  }
}
```

REDUCER

```
package com.mycompany.assignment_4.Q6StateAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<Text, IntWritable, Text, LongWritable> {

  LongWritable sum = new LongWritable();

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

    long count = 0;

    for(IntWritable v : values) {
      count += v.get();
    }

    sum.set(count);

    context.write(key, sum);
  }
}
```

DRIVER

```
package com.mycompany.assignment_4.Q6StateAnalysisTotalNumberofaccidents;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {
```

```
  public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
    Configuration conf = new Configuration();

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

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    //job.getConfiguration().set("mapreduce.output.textoutputformat.recordseparator","\t");

    // Driver Class
    job.setJarByClass(DriverClass.class);

    job.setInputFormatClass(TextInputFormat.class);

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

    // Mapper
    job.setMapperClass(MapperClass.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);

    // Reducer
    job.setReducerClass(ReducerClass.class);
    job.setNumReduceTasks(1);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
      fs.delete(outDir, true);
    }


    // Submit the job, then poll for progress until the job is complete
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

}
```

STATEMAP

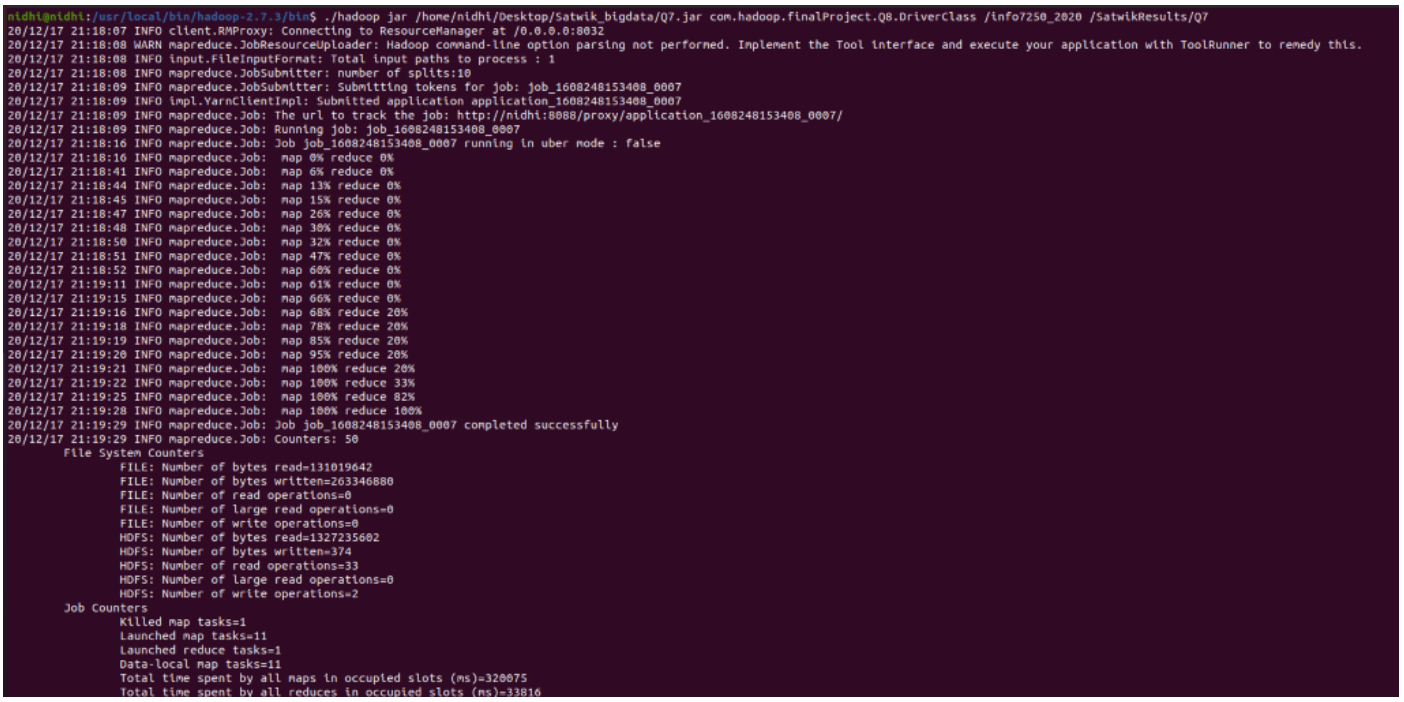
```
package com.mycompany.assignment_4.Q6StateAnalysisTotalNumberofaccidents;

import java.util.HashMap;
import java.util.Map;

public class StateMap {

  public static final Map<String, String> STATE_MAP;

  static {
    STATE_MAP = new HashMap<String, String>();
    STATE_MAP.put("AL", "Alabama");
    STATE_MAP.put("AK", "Alaska");
    STATE_MAP.put("AB", "Alberta");
    STATE_MAP.put("AZ", "Arizona");
```

```
STATE_MAP.put("AR", "Arkansas");
STATE_MAP.put("BC", "British Columbia");
STATE_MAP.put("CA", "California");
STATE_MAP.put("CO", "Colorado");
STATE_MAP.put("CT", "Connecticut");
STATE_MAP.put("DE", "Delaware");
STATE_MAP.put("DC", "District Of Columbia");
STATE_MAP.put("FL", "Florida");
STATE_MAP.put("GA", "Georgia");
STATE_MAP.put("GU", "Guam");
STATE_MAP.put("HI", "Hawaii");
STATE_MAP.put("ID", "Idaho");
STATE_MAP.put("IL", "Illinois");
STATE_MAP.put("IN", "Indiana");
STATE_MAP.put("IA", "Iowa");
STATE_MAP.put("KS", "Kansas");
STATE_MAP.put("KY", "Kentucky");
STATE_MAP.put("LA", "Louisiana");
STATE_MAP.put("ME", "Maine");
STATE_MAP.put("MB", "Manitoba");
STATE_MAP.put("MD", "Maryland");
STATE_MAP.put("MA", "Massachusetts");
STATE_MAP.put("MI", "Michigan");
STATE_MAP.put("MN", "Minnesota");
STATE_MAP.put("MS", "Mississippi");
STATE_MAP.put("MO", "Missouri");
STATE_MAP.put("MT", "Montana");
STATE_MAP.put("NE", "Nebraska");
STATE_MAP.put("NV", "Nevada");
STATE_MAP.put("NB", "New Brunswick");
STATE_MAP.put("NH", "New Hampshire");
STATE_MAP.put("NJ", "New Jersey");
STATE_MAP.put("NM", "New Mexico");
STATE_MAP.put("NY", "New York");
STATE_MAP.put("NF", "Newfoundland");
STATE_MAP.put("NC", "North Carolina");
STATE_MAP.put("ND", "North Dakota");
STATE_MAP.put("NT", "Northwest Territories");
STATE_MAP.put("NS", "Nova Scotia");
STATE_MAP.put("NU", "Nunavut");
STATE_MAP.put("OH", "Ohio");
STATE_MAP.put("OK", "Oklahoma");
STATE_MAP.put("ON", "Ontario");
STATE_MAP.put("OR", "Oregon");
STATE_MAP.put("PA", "Pennsylvania");
STATE_MAP.put("PE", "Prince Edward Island");
STATE_MAP.put("PR", "Puerto Rico");
STATE_MAP.put("QC", "Quebec");
STATE_MAP.put("RI", "Rhode Island");
STATE_MAP.put("SK", "Saskatchewan");
STATE_MAP.put("SC", "South Carolina");
STATE_MAP.put("SD", "South Dakota");
STATE_MAP.put("TN", "Tennessee");
STATE_MAP.put("TX", "Texas");
STATE_MAP.put("UT", "Utah");
STATE_MAP.put("VT", "Vermont");
STATE_MAP.put("VI", "Virgin Islands");
STATE_MAP.put("VA", "Virginia");
STATE_MAP.put("WA", "Washington");
STATE_MAP.put("WV", "West Virginia");
```

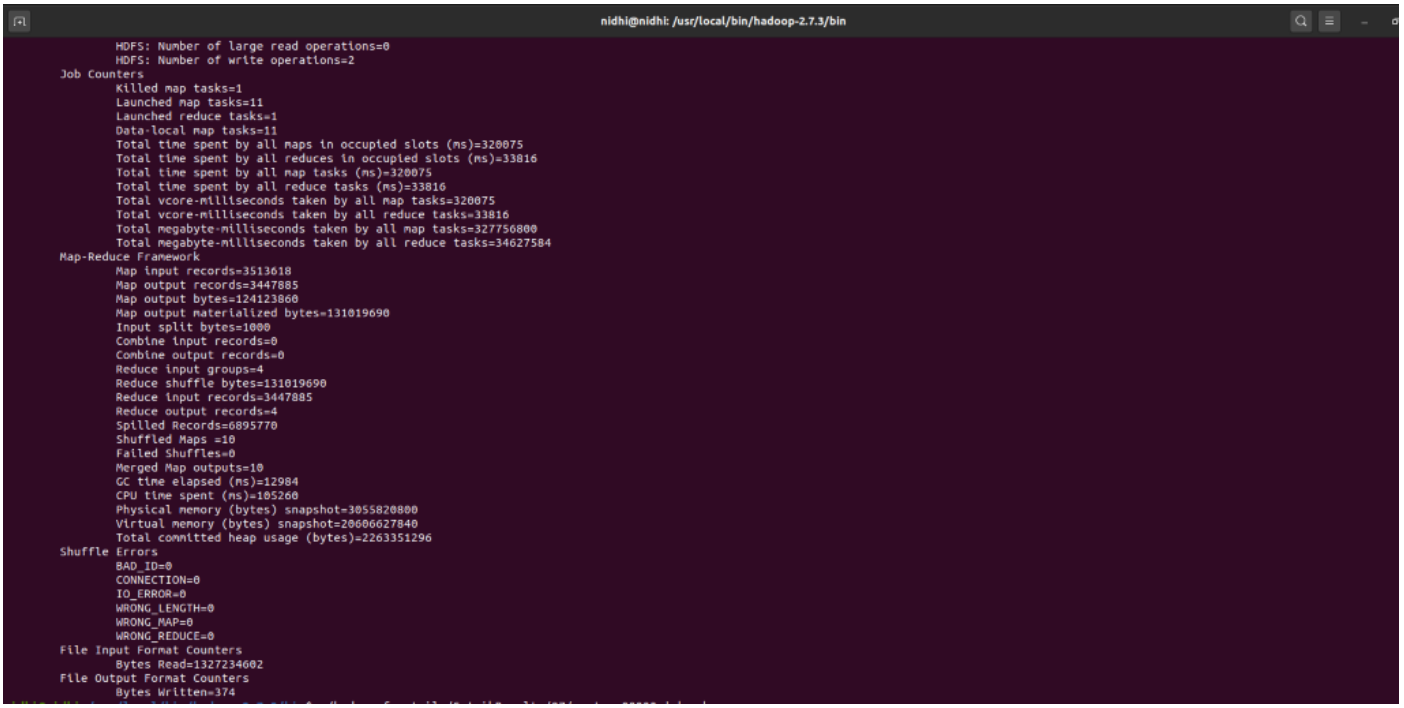
```
    STATE_MAP.put("WI", "Wisconsin");
    STATE_MAP.put("WY", "Wyoming");
    STATE_MAP.put("YT", "Yukon Territory");
  }

  public String getStateFromAbbr(String abbr) {
    return STATE_MAP.get(abbr);
  }

}
```

Q7. Temperature Analysis

This MapReduce Numerical Summarization job performs analysis to determine the minimum, maximum and average temperature (F) per severity (range from 1 to 4) of accident.



```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop jar /home/nidhi/Desktop/Satwik_bigdata/Q7.jar com.hadoop.finalProject.Q8.DriverClass /info7250_2020 /SatwikResults/Q7
20/12/17 21:18:07 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/12/17 21:18:08 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/12/17 21:18:08 INFO input.FileInputFormat: Total input paths to process : 1
20/12/17 21:18:08 INFO mapreduce.JobSubmitter: number of splits:10
20/12/17 21:18:09 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1608248153408_0007
20/12/17 21:18:09 INFO impl.YarnClientImpl: Submitted application application_1608248153408_0007
20/12/17 21:18:09 INFO mapreduce.Job: The url to track the job: http://nidhi:8088/proxy/application_1608248153408_0007/
20/12/17 21:18:09 INFO mapreduce.Job: Running job: job_1608248153408_0007
20/12/17 21:18:16 INFO mapreduce.Job: Job job_1608248153408_0007 running in uber mode : false
20/12/17 21:18:16 INFO mapreduce.Job:  map 0% reduce 0%
20/12/17 21:18:41 INFO mapreduce.Job:  map 6% reduce 0%
20/12/17 21:18:44 INFO mapreduce.Job:  map 13% reduce 0%
20/12/17 21:18:45 INFO mapreduce.Job:  map 15% reduce 0%
20/12/17 21:18:47 INFO mapreduce.Job:  map 26% reduce 0%
20/12/17 21:18:48 INFO mapreduce.Job:  map 30% reduce 0%
20/12/17 21:18:50 INFO mapreduce.Job:  map 32% reduce 0%
20/12/17 21:18:51 INFO mapreduce.Job:  map 47% reduce 0%
20/12/17 21:18:52 INFO mapreduce.Job:  map 60% reduce 0%
20/12/17 21:19:11 INFO mapreduce.Job:  map 61% reduce 0%
20/12/17 21:19:15 INFO mapreduce.Job:  map 66% reduce 0%
20/12/17 21:19:16 INFO mapreduce.Job:  map 68% reduce 20%
20/12/17 21:19:18 INFO mapreduce.Job:  map 78% reduce 20%
20/12/17 21:19:19 INFO mapreduce.Job:  map 85% reduce 20%
20/12/17 21:19:20 INFO mapreduce.Job:  map 95% reduce 20%
20/12/17 21:19:21 INFO mapreduce.Job:  map 100% reduce 20%
20/12/17 21:19:22 INFO mapreduce.Job:  map 100% reduce 33%
20/12/17 21:19:25 INFO mapreduce.Job:  map 100% reduce 82%
20/12/17 21:19:28 INFO mapreduce.Job:  map 100% reduce 100%
20/12/17 21:19:29 INFO mapreduce.Job: Job job_1608248153408_0007 completed successfully
20/12/17 21:19:29 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=131019642
                FILE: Number of bytes written=263346880
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1327235602
                HDFS: Number of bytes written=374
                HDFS: Number of read operations=33
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=320075
                Total time spent by all reduces in occupied slots (ms)=33816
```



```
nidhi@nidhi: /usr/local/bin/hadoop-2.7.3/bin
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=320075
                Total time spent by all reduces in occupied slots (ms)=33816
                Total time spent by all map tasks (ms)=320075
                Total time spent by all reduce tasks (ms)=33816
                Total vcore-milliseconds taken by all map tasks=320075
                Total vcore-milliseconds taken by all reduce tasks=33816
                Total megabyte-milliseconds taken by all map tasks=327756800
                Total megabyte-milliseconds taken by all reduce tasks=34627584
        Map-Reduce Framework
                Map input records=3513618
                Map output records=3447885
                Map output bytes=124123860
                Map output materialized bytes=131019690
                Input split bytes=1000
                Combine input records=0
                Combine output records=0
                Reduce input groups=4
                Reduce shuffle bytes=131019690
                Reduce input records=3447885
                Reduce output records=4
                Spilled Records=6895770
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=12984
                CPU time spent (ms)=105260
                Physical memory (bytes) snapshot=3055820800
                Virtual memory (bytes) snapshot=20606627840
                Total committed heap usage (bytes)=2263351296
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1327234602
        File Output Format Counters
                Bytes Written=374
```

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -tail /SatwikResults/Q7/part-r-00000 | head
1       Max Temperature = 111.0 | Min Temperature = 3.9 | Average Temperature = 70.74177685233381
2       Max Temperature = 170.6 | Min Temperature = -89.0 | Average Temperature = 61.99493106361987
3       Max Temperature = 167.0 | Min Temperature = -89.0 | Average Temperature = 61.85957242491674
4       Max Temperature = 117.0 | Min Temperature = -40.0 | Average Temperature = 59.02189844306117
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$
```

MAPPER

```
package com.mycompany.assignment_4.Q7TemperatureAnalysis;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MapperClass extends Mapper<Object, Text, IntWritable, TempWritable> {

    IntWritable sev = new IntWritable();
    TempWritable tempWritable = new TempWritable();

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

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

        if (!tokens[0].equals("ID")) {

            if(tokens[3].isEmpty() || tokens[23].isEmpty() || tokens[3].equals(" ") || tokens[26].equals(" ")) {
                return;
            }

            int severity = Integer.parseInt(tokens[3]);
            double temperature = Double.parseDouble(tokens[23]);

            tempWritable.setAverageTemp(temperature);
            tempWritable.setMaxTemp(temperature);
            tempWritable.setMinTemp(temperature);
            tempWritable.setCount(1);

            sev.set(severity);

            context.write(sev, tempWritable);
```

```
        }
      }
}
```

REDUCER

```
package com.mycompany.assignment_4.Q7TemperatureAnalysis;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<IntWritable, TempWritable, IntWritable, TempWritable> {

    TempWritable newTuple = new TempWritable();

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

        double maxTemp = Double.MIN_VALUE;
        double minTemp = Double.MAX_VALUE;
        long count = 0;
        double sumOfTemps = 0.0;

        for (TempWritable tuple : values) {

            sumOfTemps += tuple.getAverageTemp() * tuple.getCount();
            count += tuple.getCount();

            if(tuple.getMinTemp() < minTemp) {
                minTemp = tuple.getMinTemp();
            }

            if(tuple.getMaxTemp() > maxTemp) {
                maxTemp = tuple.getMaxTemp();
            }
        }

        newTuple.setAverageTemp(sumOfTemps / count);
        newTuple.setMaxTemp(maxTemp);
        newTuple.setMinTemp(minTemp);
        newTuple.setCount(count);
```

```
            context.write(key, newTuple);
        }
    }
```

TEMPWRITABLE

```
package com.mycompany.assignment_4.Q7TemperatureAnalysis;

import org.apache.hadoop.io.Writable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class TempWritable implements Writable {

    private double maxTemp;
    private double averageTemp;
    private double minTemp;

    public double getMinTemp() {
        return minTemp;
    }

    public void setMinTemp(double minTemp) {
        this.minTemp = minTemp;
    }

    private long count;

    public long getCount() {
        return count;
    }

    public void setCount(long count) {
        this.count = count;
    }

    public double getMaxTemp() {
        return maxTemp;
    }
```

```
    public void setMaxTemp(double maxTemp) {
        this.maxTemp = maxTemp;
    }

    public double getAverageTemp() {
        return averageTemp;
    }

    public void setAverageTemp(double averageTemp) {
        this.averageTemp = averageTemp;
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeDouble(averageTemp);
        dataOutput.writeDouble(maxTemp);
        dataOutput.writeDouble(minTemp);
        dataOutput.writeLong(count);
    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {
        averageTemp = dataInput.readDouble();
        maxTemp = dataInput.readDouble();
        minTemp = dataInput.readDouble();
        count = dataInput.readLong();
    }

    @Override
    public String toString() {
        return "Max Temperature = " + maxTemp +
                " | Min Temperature = " + minTemp +
                " | Average Temperature = " + averageTemp;
    }
}
```

DRIVER

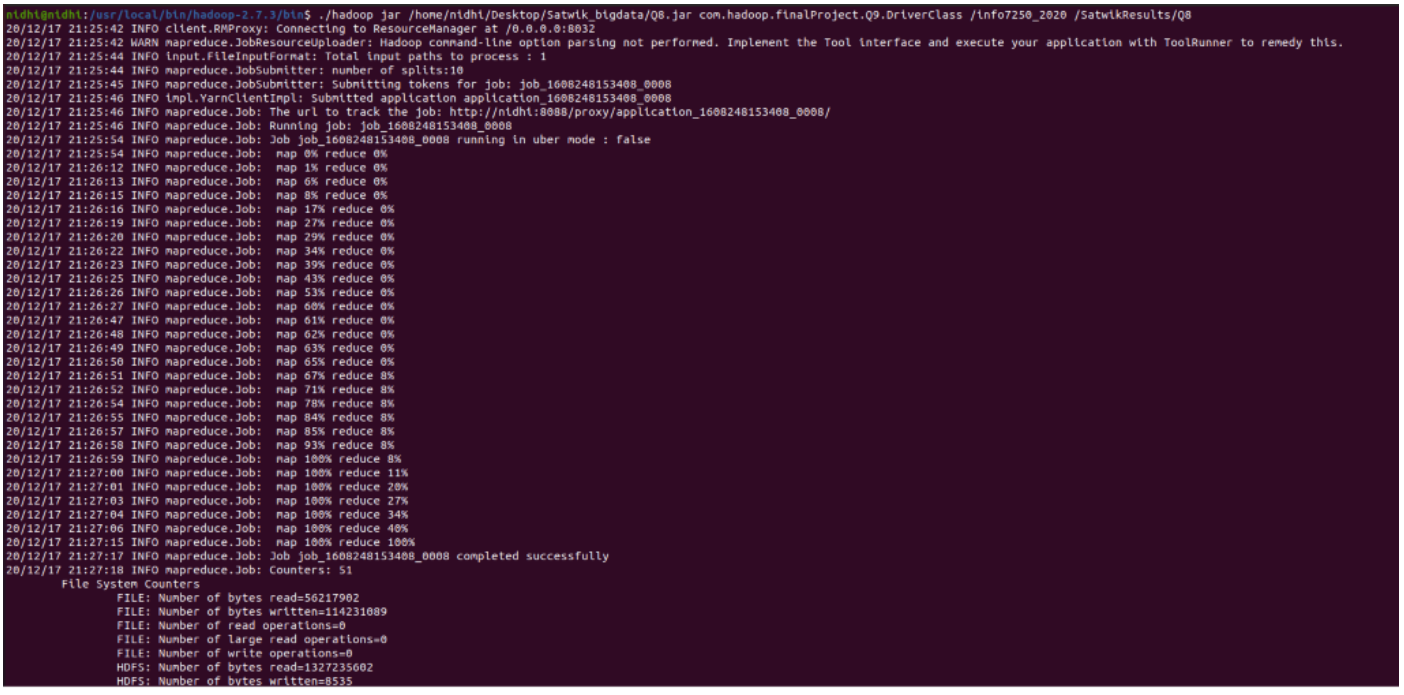
```
package com.mycompany.assignment_4.Q7TemperatureAnalysis;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
```

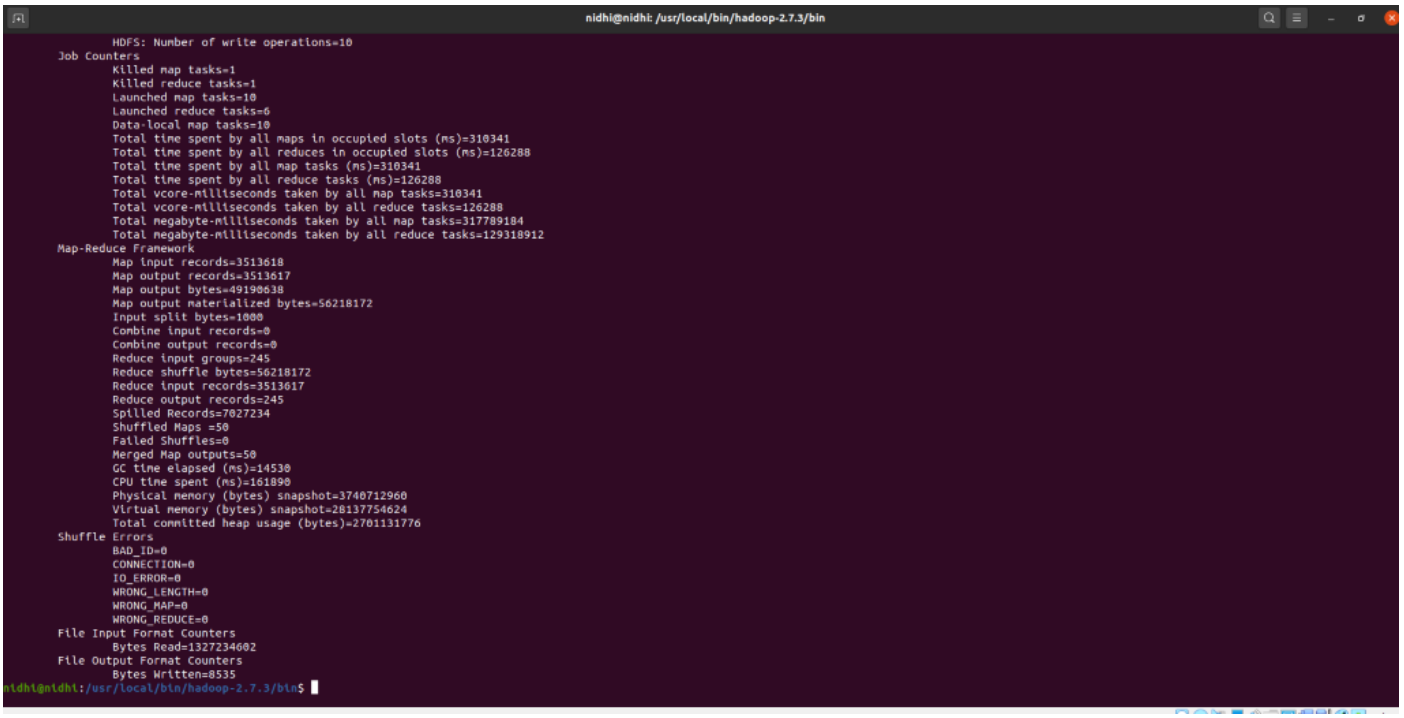
```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
        Configuration conf = new Configuration();

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

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // Driver Class
        job.setJarByClass(DriverClass.class);

        job.setInputFormatClass(TextInputFormat.class);

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

        // Mapper
        job.setMapperClass(MapperClass.class);

        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(TempWritable.class);

        // Reducer
        job.setReducerClass(ReducerClass.class);

        // Create only 1 reducer
        job.setNumReduceTasks(1);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(TempWritable.class);
```

```
        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)){
            fs.delete(outDir, true);
        }


        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

}
```

Q8. Year and State Analysis : Total Number of accidents

This MapReduce Numerical Summarization job performs the number of accidents per state per job partitioned per year






```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -cat /SatwikResults/Q8/part-r-00000 | head
Alabama - in Year - 2020 :      8251
Arkansas - in Year - 2020 :     263
Arizona - in Year - 2020 :      16257
California - in Year - 2020 :   153560
Colorado - in Year - 2020 :     9604
Connecticut - in Year - 2020 :  3097
District Of Columbia - in Year - 2020 :         1168
Delaware - in Year - 2020 :     1305
Florida - in Year - 2020 :      34251
Georgia - in Year - 2020 :      9990
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$
```

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -tail /SatwikResults/Q8/part-r-00000 | head
020 :   178
Michigan - in Year - 2020 :     7294
Minnesota - in Year - 2020 :    19131
Missouri - in Year - 2020 :     4629
Mississippi - in Year - 2020 :  625
Montana - in Year - 2020 :      8
North Carolina - in Year - 2020 :       23494
North Dakota - in Year - 2020 :         1
Nebraska - in Year - 2020 :     1464
New Hampshire - in Year - 2020 :        920
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$
```

CompositeKeyComparator

```
package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

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

public class CompositeKeyComparator extends WritableComparator {

    protected CompositeKeyComparator() {
        super(CompositeKeyWritable.class, true);
    }
```

```
    @Override
    public int compare(WritableComparable a, WritableComparable b) {

        CompositeKeyWritable w1 = (CompositeKeyWritable) a;
        CompositeKeyWritable w2 = (CompositeKeyWritable) b;

        int result = w1.getYear().compareTo(w2.getYear());

        if(result == 0) {
            return w1.getState().compareTo(w2.getState());
        }

        return result;
    }

}
```

CompositeKeyWritable

```
package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.WritableComparable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class CompositeKeyWritable implements WritableComparable<CompositeKeyWritable> {

    private String state;
    private String year;

    @Override
    public String toString() {
        return "CompositeKeyWritable{" +
                "state='" + state + '\'' +
                ", year='" + year + '\'' +
                '}';
    }

    public String getState() {
        return state;
```

```
}

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

public String getYear() {
    return year;
}

public void setYear(String year) {
    this.year = year;
}

public CompositeKeyWritable(String state, String weatherCondition) {
    super();
    this.state = state;
    this.year = weatherCondition;
}

public CompositeKeyWritable() {
    super();
}

@Override
public int compareTo(CompositeKeyWritable o) {
    int result = this.state.compareTo(o.state);
    if(result == 0) {
        return this.year.compareTo(o.year);
    }
    return result;
}

@Override
public void write(DataOutput dataOutput) throws IOException {
    dataOutput.writeUTF(state);
    dataOutput.writeUTF(year);
}

@Override
public void readFields(DataInput dataInput) throws IOException {
```

```
        state = dataInput.readUTF();
        year = dataInput.readUTF();
    }
}
```

DriverClass

```
package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

    import org.apache.hadoop.conf.Configuration;
    import org.apache.hadoop.fs.FileSystem;
    import org.apache.hadoop.fs.Path;
    import org.apache.hadoop.io.IntWritable;
    import org.apache.hadoop.io.Text;
    import org.apache.hadoop.mapreduce.Job;
    import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
    import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
    import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
    import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

    import java.io.IOException;

    public class DriverClass {

        public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
            Configuration config = new Configuration();

            Job job = Job.getInstance(config, "");

            job.setJarByClass(DriverClass.class);

            job.setGroupingComparatorClass(NaturalKeyComparator.class);
            job.setSortComparatorClass(CompositeKeyComparator.class);
            job.setPartitionerClass(NaturalKeyPartitioner.class);

            job.setMapOutputKeyClass(CompositeKeyWritable.class);
            job.setMapOutputValueClass(IntWritable.class);

            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            job.setOutputKeyClass(Text.class);
```

```
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);

        // Set Number of Reduce Tasks
        job.setNumReduceTasks(5);

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if (fs.exists(outDir)) {
            fs.delete(outDir, true);
        }

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

}
```

MapperClass

```
package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;

public class MapperClass extends Mapper<Object, Text, CompositeKeyWritable, IntWritable> {

    SimpleDateFormat parser = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    SimpleDateFormat formater = new SimpleDateFormat("yyyy");

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

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

            String stateString = tokens[17];
            String year = null;
            try {
                year = formater.format(parser.parse(tokens[4]));
            } catch (ParseException e) {
                e.printStackTrace();
            }

            CompositeKeyWritable cKW = new CompositeKeyWritable(stateString, year);

            context.write(cKW, new IntWritable(1));
        }
}
```

NaturalKeyComparator

```
    package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

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

    public class NaturalKeyComparator extends WritableComparator {

        public NaturalKeyComparator() {
            super(CompositeKeyWritable.class, true);
        }

        @Override
        public int compare(WritableComparable a, WritableComparable b) {
            CompositeKeyWritable w1 = (CompositeKeyWritable) a;
            CompositeKeyWritable w2 = (CompositeKeyWritable) b;

            return w1.getState().compareTo(w2.getState());

        }
}
```

NaturalKeyPartitioner

```
    package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;
```

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class NaturalKeyPartitioner extends Partitioner<CompositeKeyWritable, IntWritable> {

    @Override
    public int getPartition(CompositeKeyWritable compositeKeyWritable, IntWritable intWritable, int i) {
        return Integer.parseInt(compositeKeyWritable.getYear()) % i;
    }
}
```

ReducerClass

```
package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<CompositeKeyWritable, IntWritable, Text, IntWritable> {

    Text text = new Text();
    IntWritable count = new IntWritable();
    StateMap stateMap = new StateMap();

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

        int sum = 0;

        for(IntWritable v : values) {
            sum += v.get();
        }

        text.set(stateMap.getStateFromAbbr(key.getState())+ " - in Year - " + key.getYear() + " : ");
        count.set(sum);

        context.write(text, count);
```

```
    }
}
```

StateMap

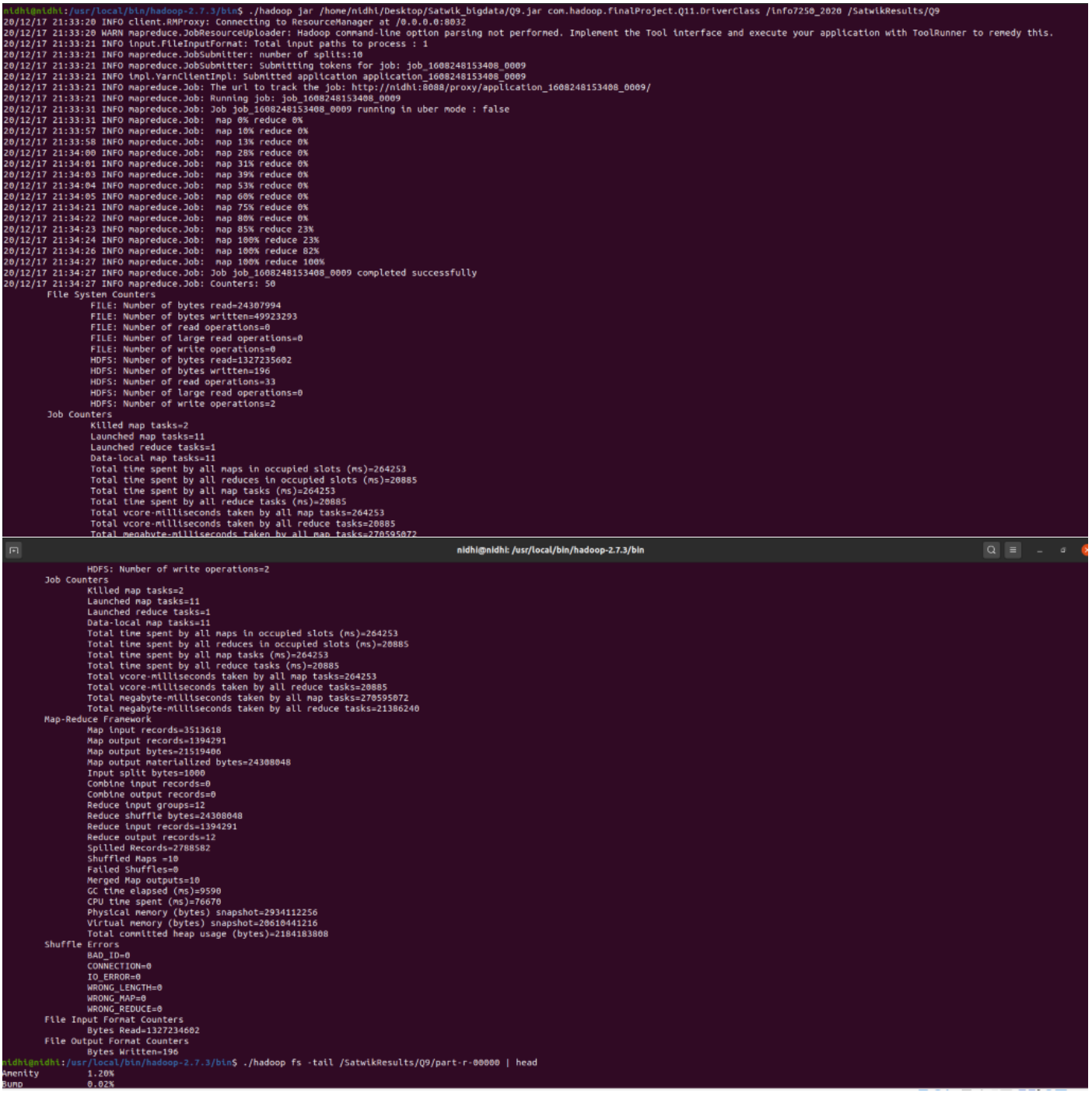
```
package com.mycompany.assignment_4.Q8YearandStateAnalysisTotalNumberofaccidents;

import java.util.HashMap;
import java.util.Map;

public class StateMap {

    public static final Map<String, String> STATE_MAP;

    static {
        STATE_MAP = new HashMap<String, String>();
        STATE_MAP.put("AL", "Alabama");
        STATE_MAP.put("AK", "Alaska");
        STATE_MAP.put("AB", "Alberta");
        STATE_MAP.put("AZ", "Arizona");
        STATE_MAP.put("AR", "Arkansas");
        STATE_MAP.put("BC", "British Columbia");
        STATE_MAP.put("CA", "California");
        STATE_MAP.put("CO", "Colorado");
        STATE_MAP.put("CT", "Connecticut");
        STATE_MAP.put("DE", "Delaware");
        STATE_MAP.put("DC", "District Of Columbia");
        STATE_MAP.put("FL", "Florida");
        STATE_MAP.put("GA", "Georgia");
        STATE_MAP.put("GU", "Guam");
        STATE_MAP.put("HI", "Hawaii");
        STATE_MAP.put("ID", "Idaho");
        STATE_MAP.put("IL", "Illinois");
        STATE_MAP.put("IN", "Indiana");
        STATE_MAP.put("IA", "Iowa");
        STATE_MAP.put("KS", "Kansas");
        STATE_MAP.put("KY", "Kentucky");
        STATE_MAP.put("LA", "Louisiana");
        STATE_MAP.put("ME", "Maine");
        STATE_MAP.put("MB", "Manitoba");
        STATE_MAP.put("MD", "Maryland");
        STATE_MAP.put("MA", "Massachusetts");
```

```
STATE_MAP.put("MI", "Michigan");
STATE_MAP.put("MN", "Minnesota");
STATE_MAP.put("MS", "Mississippi");
STATE_MAP.put("MO", "Missouri");
STATE_MAP.put("MT", "Montana");
STATE_MAP.put("NE", "Nebraska");
STATE_MAP.put("NV", "Nevada");
STATE_MAP.put("NB", "New Brunswick");
STATE_MAP.put("NH", "New Hampshire");
STATE_MAP.put("NJ", "New Jersey");
STATE_MAP.put("NM", "New Mexico");
STATE_MAP.put("NY", "New York");
STATE_MAP.put("NF", "Newfoundland");
STATE_MAP.put("NC", "North Carolina");
STATE_MAP.put("ND", "North Dakota");
STATE_MAP.put("NT", "Northwest Territories");
STATE_MAP.put("NS", "Nova Scotia");
STATE_MAP.put("NU", "Nunavut");
STATE_MAP.put("OH", "Ohio");
STATE_MAP.put("OK", "Oklahoma");
STATE_MAP.put("ON", "Ontario");
STATE_MAP.put("OR", "Oregon");
STATE_MAP.put("PA", "Pennsylvania");
STATE_MAP.put("PE", "Prince Edward Island");
STATE_MAP.put("PR", "Puerto Rico");
STATE_MAP.put("QC", "Quebec");
STATE_MAP.put("RI", "Rhode Island");
STATE_MAP.put("SK", "Saskatchewan");
STATE_MAP.put("SC", "South Carolina");
STATE_MAP.put("SD", "South Dakota");
STATE_MAP.put("TN", "Tennessee");
STATE_MAP.put("TX", "Texas");
STATE_MAP.put("UT", "Utah");
STATE_MAP.put("VT", "Vermont");
STATE_MAP.put("VI", "Virgin Islands");
STATE_MAP.put("VA", "Virginia");
STATE_MAP.put("WA", "Washington");
STATE_MAP.put("WV", "West Virginia");
STATE_MAP.put("WI", "Wisconsin");
STATE_MAP.put("WY", "Wyoming");
STATE_MAP.put("YT", "Yukon Territory");
```

```
    }

    public String getStateFromAbbr(String abbr) {
        return STATE_MAP.get(abbr);
    }

}
```

Q9. Percentage Accidents vs Proximity

This MapReduce Numerical Summarization performs the percentage of accidents vs. the proximity to traffic object.



```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop jar /home/nidhi/Desktop/Satwik_bigdata/Q9.jar com.hadoop.finalProject.Q11.DriverClass /info7250_2020 /SatwikResults/Q9
20/12/17 21:33:20 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/12/17 21:33:20 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/12/17 21:33:21 INFO input.FileInputFormat: Total input paths to process : 1
20/12/17 21:33:21 INFO mapreduce.JobSubmitter: number of splits:10
20/12/17 21:33:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1608248153408_0009
20/12/17 21:33:21 INFO impl.YarnClientImpl: Submitted application application_1608248153408_0009
20/12/17 21:33:21 INFO mapreduce.Job: The url to track the job: http://nidhi:8088/proxy/application_1608248153408_0009/
20/12/17 21:33:21 INFO mapreduce.Job: Running job: job_1608248153408_0009
20/12/17 21:33:31 INFO mapreduce.Job: Job job_1608248153408_0009 running in uber mode : false
20/12/17 21:33:31 INFO mapreduce.Job:  map 0% reduce 0%
20/12/17 21:33:57 INFO mapreduce.Job:  map 10% reduce 0%
20/12/17 21:33:58 INFO mapreduce.Job:  map 13% reduce 0%
20/12/17 21:34:00 INFO mapreduce.Job:  map 28% reduce 0%
20/12/17 21:34:01 INFO mapreduce.Job:  map 31% reduce 0%
20/12/17 21:34:03 INFO mapreduce.Job:  map 39% reduce 0%
20/12/17 21:34:04 INFO mapreduce.Job:  map 53% reduce 0%
20/12/17 21:34:05 INFO mapreduce.Job:  map 60% reduce 0%
20/12/17 21:34:21 INFO mapreduce.Job:  map 75% reduce 0%
20/12/17 21:34:22 INFO mapreduce.Job:  map 80% reduce 0%
20/12/17 21:34:23 INFO mapreduce.Job:  map 85% reduce 23%
20/12/17 21:34:24 INFO mapreduce.Job:  map 100% reduce 23%
20/12/17 21:34:26 INFO mapreduce.Job:  map 100% reduce 82%
20/12/17 21:34:27 INFO mapreduce.Job:  map 100% reduce 100%
20/12/17 21:34:27 INFO mapreduce.Job: Job job_1608248153408_0009 completed successfully
20/12/17 21:34:27 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=24307994
                FILE: Number of bytes written=49923293
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1327235602
                HDFS: Number of bytes written=196
                HDFS: Number of read operations=33
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=2
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=264253
                Total time spent by all reduces in occupied slots (ms)=20885
                Total time spent by all map tasks (ms)=264253
                Total time spent by all reduce tasks (ms)=20885
                Total vcore-milliseconds taken by all map tasks=264253
                Total vcore-milliseconds taken by all reduce tasks=20885
                Total megabyte-milliseconds taken by all map tasks=270595072
```

```
nidhi@nidhi: /usr/local/bin/hadoop-2.7.3/bin
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=2
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=264253
                Total time spent by all reduces in occupied slots (ms)=20885
                Total time spent by all map tasks (ms)=264253
                Total time spent by all reduce tasks (ms)=20885
                Total vcore-milliseconds taken by all map tasks=264253
                Total vcore-milliseconds taken by all reduce tasks=20885
                Total megabyte-milliseconds taken by all map tasks=270595072
                Total megabyte-milliseconds taken by all reduce tasks=21386240
        Map-Reduce Framework
                Map input records=3513618
                Map output records=1394291
                Map output bytes=21519406
                Map output materialized bytes=24308048
                Input split bytes=1000
                Combine input records=0
                Combine output records=0
                Reduce input groups=12
                Reduce shuffle bytes=24308048
                Reduce input records=1394291
                Reduce output records=12
                Spilled Records=2788582
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=9590
                CPU time spent (ms)=76670
                Physical memory (bytes) snapshot=2934112256
                Virtual memory (bytes) snapshot=20610441216
                Total committed heap usage (bytes)=2184183808
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1327234602
        File Output Format Counters
                Bytes Written=196
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -tail /SatwikResults/Q9/part-r-00000 | head
Amenity         1.20%
Bump            0.02%
```

```
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$ ./hadoop fs -tail /SatwikResults/Q9/part-r-00000 | head
Amenity         1.20%
Bump            0.02%
Crossing                7.81%
Give_Way                0.27%
Junction                8.10%
No_Exit         0.12%
Railway         0.89%
Roundabout              0.01%
Station         2.00%
Stop            1.48%
nidhi@nidhi:/usr/local/bin/hadoop-2.7.3/bin$
```

Driver

```
package com.mycompany.assignment_4.Q9PercentageAccidentsvsProximity;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
        Configuration conf = new Configuration();

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

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // Driver Class
        job.setJarByClass(DriverClass.class);

        job.setInputFormatClass(TextInputFormat.class);

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

```
        // Mapper
        job.setMapperClass(MapperClass.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        // Reducer
        job.setReducerClass(ReducerClass.class);

        // Create only 1 reducer
        job.setNumReduceTasks(1);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)){
            fs.delete(outDir, true);
        }


        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

}
```

Mapper

```
package com.mycompany.assignment_4.Q9PercentageAccidentsvsProximity;

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

    Text proximity = new Text();
```

```
    IntWritable one = new IntWritable(1);

    RowHeaderClass rhc = new RowHeaderClass();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String[] tokens = value.toString().split(",");
        if (!tokens[0].equals("ID")) {

            for(int i = 32; i < 45; i++) {
                if(tokens[i].equals("True")) {
                    proximity.set(rhc.getRowName(i));
                    context.write(proximity, one);
                }
            }
        }
    }

}
```

Reducer

```
package com.mycompany.assignment_4.Q9PercentageAccidentsvsProximity;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

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

    Text percentage = new Text();

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

        int count = 0;

        for(IntWritable v : values) {
            count += v.get();
```

```
        }

        int total = 3513617;  // total number of records

        double perc = ((double) count / total) * 100;
        percentage.set("\t" + String.format("%.2f", perc) + "%");

        context.write(key, percentage);
    }
}
```

RowHeaders

```
package com.mycompany.assignment_4.Q9PercentageAccidentsvsProximity;

import java.util.HashMap;
import java.util.Map;

public class RowHeaderClass {

    Map<Integer, String> rowNames = new HashMap<>();
    String []headerNames =
{"Amenity","Bump","Crossing","Give_Way","Junction","No_Exit","Railway","Roundabout","Station","Stop","Traffic_Cal
ming","Traffic_Signal","Turning_Loop"};

    public RowHeaderClass() {

        for(int i = 32; i < 45; i++) {
            rowNames.put(i, headerNames[i-32]);
        }

    }

    public String getRowName(int rowNumber) {
        return rowNames.get(rowNumber);
    }
}
```

APACHE HIVE ANALYSIS

Q1. Number of Accidents per Time Zone

Q2. Weather Conditions leading to accidents

References

https://www.kaggle.com/sobhanmoosavi/us-accidents

https://stackoverflow.com/

https://american-cse.org/csci2015/data/9795a392.pdf

https://towardsdatascience.com/usa-accidents-data-analysis-d130843cde02