

Big Data Analysis using Hadoop

-By Shubham Mehta

Objective: The objective of this project is to demonstrate the ability and usability of the Hadoop framework for analyzing large volumes of data (Big Data).

Architecture

Operating System	Stand-alone Ubuntu-server-12.04 running in VMWare on Windows 10, 64-bit
Memory	2GB
IDE	Eclipse
Java	1.6
Hadoop Release	1.0.3
Other Tools	Putty, WinSCP

Source of Data: <https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/xx67-kt59>

Size of Data: 155 MB (Approximately 448,115 records)

Work Objective

Use MapReduce to do the following

- 1) Summarize all the restaurants with critical violations
- 2) Count the number of restaurants with critical violations, non-critical violations and no violations at all per major area (e.g Queens, Brooklyn ,etc)
- 3) Find the pizza places in each major area that have not committed a critical violation
- 4) Find the zip code per major area with the least number of critical violations
- 5) Based on results from objective #3, find the safest area to eat

Solution

1. Copy file from Windows to Ubuntu using WinSCP
2. For file ingestion, run the command: `hadoop fs -copyFromLocal /home/notroot/lab/data/DOHMH_New_York_City_Restaurant_Inspection_Results.txt /input`
3. Verify that the file has been copied:

```
notroot@ubuntu:~$ hadoop fs -ls /input
Warning: $HADOOP_HOME is deprecated.

Found 12 items
-rw-r--r-- 1 notroot supergroup 1777167 2016-07-15 08:17 /input/1901_C
-rw-r--r-- 1 notroot supergroup 814 2016-07-15 10:02 /input/1901_S
-rw-r--r-- 1 notroot supergroup 162325734 2016-08-23 07:30 /input/DOHMH_New_York_City_Restaurant_Inspection_Results.csv
-rw-r--r-- 1 notroot supergroup 161877619 2016-08-24 14:32 /input/DOHMH_New_York_City_Restaurant_Inspection_Results.txt
-rw-r--r-- 1 notroot supergroup 134 2016-07-01 10:42 /input/c.txt
-rw-r--r-- 1 notroot supergroup 391355 2016-06-30 13:40 /input/custs
-rw-r--r-- 1 notroot supergroup 67456 2016-07-02 20:06 /input/exchange-rate1.txt
-rw-r--r-- 1 notroot supergroup 157 2016-07-21 11:46 /input/partitiondata.txt
-rw-r--r-- 1 notroot supergroup 99271 2016-08-06 09:18 /input/sample.log
-rw-r--r-- 1 notroot supergroup 167 2016-07-05 08:50 /input/sample1.txt
-rw-r--r-- 1 notroot supergroup 88328787 2016-06-30 13:39 /input/txns
-rw-r--r-- 1 notroot supergroup 24 2016-08-06 06:51 /input/words
notroot@ubuntu:~$
```

OR using Web UI

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
1901_C	file	1.69 MB	1	64 MB	2016-07-15 08:17	rw-r--r--	notroot	supergroup
1901_S	file	0.79 KB	1	64 MB	2016-07-15 10:02	rw-r--r--	notroot	supergroup
DOHMH_New_York_City_Restaurant_Inspection_Results.csv	file	154.81 MB	1	64 MB	2016-08-23 07:30	rw-r--r--	notroot	supergroup
DOHMH_New_York_City_Restaurant_Inspection_Results.txt	file	154.38 MB	1	64 MB	2016-08-24 14:32	rw-r--r--	notroot	supergroup
c.txt	file	0.13 KB	1	64 MB	2016-07-01 10:42	rw-r--r--	notroot	supergroup
custs	file	382.18 KB	1	64 MB	2016-06-30 13:40	rw-r--r--	notroot	supergroup
exchange-rate1.txt	file	65.88 KB	1	64 MB	2016-07-02 20:06	rw-r--r--	notroot	supergroup
partitiondata.txt	file	0.15 KB	1	64 MB	2016-07-21 11:46	rw-r--r--	notroot	supergroup
sample.log	file	96.94 KB	1	64 MB	2016-08-06 09:18	rw-r--r--	notroot	supergroup
sample1.txt	file	0.16 KB	1	64 MB	2016-07-05 08:50	rw-r--r--	notroot	supergroup
txns	file	84.24 MB	1	64 MB	2016-06-30 13:39	rw-r--r--	notroot	supergroup

4. MapReduce Code for summarizing hotels with critical violations

```
package shubham;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
//import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import java.io.IOException;
import java.util.StringTokenizer;

public class CriticalViolations {
    public static class MyMapper extends Mapper<LongWritable, Text, Text,
Text>{

        public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException{
            String line=value.toString();
            String[] arr=line.split("\t");

            String s1=arr[12];
            if(s1.equals("Critical")){
                String x=arr[1]+" (" +arr[3]+"," +arr[4]+"," +arr[2]+")";
                String small="";
                String shorten=arr[11];
                for(int i=0;i<shorten.length();i++){
                    small+=shorten.charAt(i);
                    if(shorten.charAt(i)=='.' )
                        break;
                }
            }
        }
    }
}
```

```

        context.write(new Text(x), new Text(small));
    }
}

public static class MyReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context
context) throws IOException, InterruptedException{
        for(Text text: values){
            context.write(key, text);
        }
    }
}

public static void main(String[] args) throws Exception{
    Configuration conf= new Configuration();
    String[] otherArgs = new
GenericOptionsParser(conf, args).getRemainingArgs();
    Job job = new Job(conf, "CriticalViolations");

    job.setJarByClass(CriticalViolations.class);
    job.setMapperClass(MyMapper.class);
    job.setReducerClass(MyReducer.class);

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

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

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

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

```

Testing Class:

```

package shubham;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.MapDriver;
import org.apache.hadoop.mapreduce.MapReduceDriver;
import org.apache.hadoop.mapreduce.ReduceDriver;
import org.junit.Before;
import org.junit.Test;

import shubham.CriticalViolations.MyMapper;
import shubham.CriticalViolations.MyReducer;

```

```

import java.util.*;

public class CriticalTests {

    MapReduceDriver<LongWritable,Text,Text,Text,Text,Text> mapReduceDriver;
    MapDriver<LongWritable,Text,Text,Text> mapDriver;
    ReduceDriver<Text,Text,Text,Text> reduceDriver;

    @Before
    public void setUp() {
        MyMapper mapper = new MyMapper();
        MyReducer reducer = new MyReducer();

        mapDriver = new MapDriver<LongWritable,Text,Text,Text>();
        mapDriver.setMapper(mapper);

        reduceDriver = new ReduceDriver<Text,Text,Text,Text>();
        reduceDriver.setReducer(reducer);

        mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text,
Text, Text, Text>();
        mapReduceDriver.setMapper(mapper);
        mapReduceDriver.setReducer(reducer);

    }

    @Test
    public void testMapper(){
        mapDriver.withInput(new LongWritable(0), new Text("41571350 THE
MET GRILL/DOUBLE TREE HOTEL MANHATTAN 569 LEXINGTON AVENUE 10022
2123506081 American 4/8/2016 Violations were cited in the
following area(s). 09C Food contact surface not properly maintained.
Critical 33 8/21/2016 Cycle Inspection / Initial
Inspection"));

        mapDriver.withOutput(new Text("THE MET GRILL/DOUBLE TREE HOTEL
(569,LEXINGTON AVENUE,MANHATTAN)"),new Text("Food contact surface not
properly maintained."));
        mapDriver.runTest();
    }

    @Test
    public void testReducer(){
        List<Text> values = new ArrayList<Text>();
        values.add(new Text("Food contact surface not properly
maintained."));

        reduceDriver.withInput(new Text("THE MET GRILL/DOUBLE TREE HOTEL
(569,LEXINGTON AVENUE,MANHATTAN)"), values);
        reduceDriver.withOutput(new Text("THE MET GRILL/DOUBLE TREE HOTEL
(569,LEXINGTON AVENUE,MANHATTAN)"), new Text("Food contact surface not
properly maintained."));

    }
}

```

```

@Test
public void testMapReduceSingleProduct(){
    mapReduceDriver.withInput(new LongWritable(0), new Text("41571350
    THE MET GRILL/DOUBLE TREE HOTEL      MANHATTAN    569    LEXINGTON AVENUE
    10022 2123506081    American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
    Critical    33    8/21/2016    Cycle Inspection / Initial
Inspection"));
    mapReduceDriver.addOutput(new Text("THE MET GRILL/DOUBLE TREE
HOTEL (569,LEXINGTON AVENUE,MANHATTAN)"), new Text("Food contact surface not
properly maintained."));
    mapReduceDriver.runTest();
}
}

```

Executing the job:

```

notroot@ubuntu:~/lab/programs$ hadoop jar CriticalViolations.jar shubham.CriticalViolations /input/DOHMH_New_York_City_Restaurant_Inspection_Resu
lts.txt /output/CriticalViolations
Warning: $HADOOP_HOME is deprecated.

16/08/24 20:56:35 INFO input.FileInputFormat: Total input paths to process : 1
16/08/24 20:56:35 INFO util.NativeCodeLoader: Loaded the native-hadoop library

```

Confirming Output:

File: [/output/CriticalViolations/part-r-00000](#)

Goto:

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```

"1,001 NIGHTS" (35,NEPTUNE AVENUE,BROOKLYN)    "Food contact surface not properly washed, rinsed and sanitized after each use and following any activity when contamination may have
occurred.
"1,001 NIGHTS" (35,NEPTUNE AVENUE,BROOKLYN)    "Food not cooled by an approved method whereby the internal product temperature is reduced from 140° F to 70° F or less within 2 hours,
and from 70° F to 41° F or less within 4 additional hours.

```

```

notroot@ubuntu:~/lab/programs$ hadoop fs -ls /output/CriticalViolations
Warning: $HADOOP_HOME is deprecated.

```

```

Found 3 items
-rw-r--r--  1 notroot supergroup          0 2016-08-24 20:57 /output/CriticalViolations/_SUCCESS
drwxr-xr-x  - notroot supergroup          0 2016-08-24 20:56 /output/CriticalViolations/_logs
-rw-r--r--  1 notroot supergroup 36993332 2016-08-24 20:57 /output/CriticalViolations/part-r-00000

```

- 6) MapReduce Code for counting the number of hotels with critical violations, non-critical violations and no violations at all per major area (e.g Queens, Brooklyn ,etc)

```

package shubham;
import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
//import org.apache.hadoop.mapreduce.Mapper.Context;

```

```

import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;
import java.util.StringTokenizer;

public class CountArea {
public static class MyMapper extends Mapper<LongWritable, Text, Text, Text>{

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

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

        if(arr.length!=17 && arr.length!=18){
            System.out.println("EXEC" + arr.length + " " +
line);
        }

        context.write(new Text(arr[2]),new Text(line));

    }

}

public static class MyReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException{

        int critical=0;
        int nonC=0;
        int noV=0;

        for(Text t:values){

            String line =t.toString();
            String[] arr = line.split("\t");

            if(arr.length==17){
                noV++;
            }
            else if(arr[12].equals("Not Applicable")){

                noV++;
            }
            else if(arr[12].equals("Critical")){
                critical++;
            }
            else if(arr[12].equals("Not Critical")){
                nonC++;
            }

        }

    }
}

```

```

                if(!key.toString().equals("BORO") &&
!key.toString().equals("Missing"))
                    context.write(key, new Text("Critical Violation: " + critical + "
Non Critical Violations: " + nonC + "   No Violations: " + noV));
            }
        }

public static void main(String[] args) throws Exception{
    Configuration conf= new Configuration();
    String[] otherArgs = new
GenericOptionsParser(conf,args).getRemainingArgs();
    Job job = new Job(conf, "Count Per Area");

    job.setJarByClass(CriticalViolations.class);
    job.setMapperClass(MyMapper.class);
    job.setReducerClass(MyReducer.class);

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

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

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

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

}

```

Testing Class:

```

package shubham;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.ReduceDriver;
import org.apache.hadoop.mapreduce.ReduceDriver;
import org.junit.Before;
import org.junit.Test;

import shubham.CountArea.MyMapper;
import shubham.CountArea.MyReducer;

import java.util.*;

public class CountAreaTesting {

    MapReduceDriver<LongWritable,Text,Text,Text,Text,Text> mapReduceDriver;
    MapDriver<LongWritable,Text,Text,Text> mapDriver;
    ReduceDriver<Text,Text,Text,Text> reduceDriver;

```

```

@Before
public void setUp() {
    MyMapper mapper = new MyMapper();
    MyReducer reducer = new MyReducer();

    mapDriver = new MapDriver<LongWritable,Text,Text,Text>();
    mapDriver.setMapper(mapper);

    reduceDriver = new ReduceDriver<Text,Text,Text,Text>();
    reduceDriver.setReducer(reducer);

    mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text,
Text, Text, Text>();
    mapReduceDriver.setMapper(mapper);
    mapReduceDriver.setReducer(reducer);

}

@Test
public void testMapper(){
    mapDriver.withInput(new LongWritable(0), new Text("41571350 THE
MET GRILL/DOUBLE TREE HOTEL    MANHATTAN    569    LEXINGTON AVENUE    10022
2123506081    American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
Critical    33    8/21/2016    Cycle Inspection / Initial
Inspection"));
    mapDriver.withOutput(new Text("MANHATTAN"),new Text("41571350
THE MET GRILL/DOUBLE TREE HOTEL    MANHATTAN    569    LEXINGTON AVENUE
10022 2123506081    American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
Critical    33    8/21/2016    Cycle Inspection / Initial
Inspection"));
    mapDriver.runTest();
}

@Test
public void testReducer(){
    List<Text> values = new ArrayList<Text>();
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American
4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained.    Critical    33
8/21/2016    Cycle Inspection / Initial Inspection"));
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American
4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained.    Not Critical    33
8/21/2016    Cycle Inspection / Initial Inspection"));
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American
4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained.    Not Applicable    33
8/21/2016    Cycle Inspection / Initial Inspection"));
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American

```



```

        4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained. 33                        8/21/2016    Cycle
Inspection / Initial Inspection"));
        reduceDriver.withInput(new Text("MANHATTAN"), values);
        reduceDriver.withOutput(new Text("MANHATTAN"), new Text("Critical
Violation: 1 Non Critical Violations: 1 No Violations: 2"));
    }

    @Test
    public void testMapReduceSingleProduct(){
        mapReduceDriver.withInput(new LongWritable(0), new Text("41571350
THE MET GRILL/DOUBLE TREE HOTEL    MANHATTAN    569    LEXINGTON AVENUE
10022 2123506081 American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
Critical    33                        8/21/2016    Cycle Inspection / Initial
Inspection"));
        mapReduceDriver.addOutput(new Text("MANHATTAN"), new
Text("Critical Violation: 1 Non Critical Violations: 0 No Violations: 0"));
        mapReduceDriver.runTest();
    }
}

```

File: [/output/CountPerArea/part-r-00000](#)

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

```

BRONX    Critical Violation: 21481 Non Critical Violations: 17552 No Violations: 806
BROOKLYN    Critical Violation: 58839 Non Critical Violations: 47618 No Violations: 2331
MANHATTAN    Critical Violation: 98700 Non Critical Violations: 76563 No Violations: 3301
QUEENS    Critical Violation: 58260 Non Critical Violations: 45843 No Violations: 1949
STATEN ISLAND    Critical Violation: 8295 Non Critical Violations: 6290 No Violations: 252

```

- 7) MapReduce code for finding the pizza places in each major area that have not committed a critical violation

```
package shubham;
```

```
import java.io.IOException;
```

```

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

```

```

import org.apache.hadoop.util.GenericOptionsParser;

public class PartitionMapper extends Mapper<LongWritable,Text,Text,Text> {
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException{
    String line = value.toString();
    String[] arr=line.split("\t");

    if(arr.length!=17 && arr.length!=18){
        System.out.println("EXEC" + arr.length + " " + line);
    }
    if(!arr[2].equals("BORO") && !arr[2].equals("Missing")){
        if(arr[7].equals("Pizza") || arr[7].equals("Pizza/Italian")){
            context.write(new Text(arr[7]),new Text(line));
        }
    }
}
}

```

```

package shubham;

```

```

import org.apache.hadoop.io.Text;

```

```

import org.apache.hadoop.mapreduce.Partitioner;

```

```

public class AreaPartitioner extends Partitioner<Text,Text> {
@Override
public int getPartition(Text key, Text value, int numReduceTasks){
    String[] arr=value.toString().split("\t");
    String area = arr[2];

    if(numReduceTasks == 0){
        return 0;
    }

    if(area.equals("MANHATTAN")){
        return 0;
    }
    if(area.equals("BRONX")){
        return 1;
    }
    if(area.equals("BROOKLYN")){
        return 2;
    }
    if(area.equals("QUEENS")){
        return 3;
    }
    return 4;
}
}

```

```

package shubham;

```

```

import java.io.IOException;

```

```

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

public class PartitionReducer extends Reducer<Text,Text,Text,Text> {
    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws
        IOException, InterruptedException{

        for(Text val: values){
            String[] arr = val.toString().split("\t");
            if(arr.length==17 || arr[12].equals("Not Critical") ||
arr[12].equals("Not Applicable")){
                context.write( new Text(arr[2]),new Text(arr[1] + " (" +
arr[4]+")" ) );
            }
        }
    }
}

package shubham;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.GenericMRLoadGenerator.SampleMapper;
import org.apache.hadoop.mapred.GenericMRLoadGenerator.SampleReducer;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class SafePizza {

    public static void main(String[] args){
        try{
            Configuration conf = new Configuration();
            String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
            Job job = new Job(conf, "Pizza Places");
            job.setJarByClass(SafePizza.class);
            job.setMapperClass(PartitionMapper.class);
            job.setReducerClass(PartitionReducer.class);
            job.setPartitionerClass(AreaPartitioner.class);
            job.setNumReduceTasks(5);

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

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

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

```

```

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

        System.exit(job.waitForCompletion(true)?0:1);
    }
    catch (Exception e){e.printStackTrace();}

}
}

```

Output:

File: [/output/Pizza22/part-r-00000](#)

Goto : go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

MANHATTAN	LA CROSTA RESTAURANT (EAST 72 STREET)
MANHATTAN	DONA BELLA PIZZA (CHURCH STREET)
MANHATTAN	A-1 PIZZA SHOP (GRAND STREET)
MANHATTAN	ROSA'S PIZZA (W 14TH ST)
MANHATTAN	DOMINO'S (WEST 181 STREET)
MANHATTAN	PAPA JOHN'S (BROADWAY)
MANHATTAN	PICK & PAY GYRO & PIZZA (LEXINGTON AVE)
MANHATTAN	RAY'S PIZZA (2 AVENUE)
MANHATTAN	F & F 99 CENTS PIZZA (3 AVENUE)
MANHATTAN	MEDITERRANEO PIZZA (SAINT NICHOLAS AVE)
MANHATTAN	GINO'S PIZZA (CATHERINE STREET)
MANHATTAN	VTNNV VTNCNM7 (1 AVENUE)

File: [/output/Pizza22/part-r-00002](#)

Goto : go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

BROOKLYN	PAPA JOHN'S (FULTON STREET)
BROOKLYN	DOMINO'S (CLARKSON AVENUE)
BROOKLYN	DOMINO'S (FULTON STREET)
BROOKLYN	PIZZA PARTY (IRVING AVE)
BROOKLYN	TRI-COLOR RESTAURANT AND PIZZERIA (GRAHAM AVENUE)
BROOKLYN	BELLA PIZZA & RESTAURANT (LIVINGSTON ST)
BROOKLYN	LENNY'S PIZZA (86 STREET)
BROOKLYN	LITTLE MUNCHIES (EVERGREEN AVENUE)
BROOKLYN	PIZZA PRINCE (HASSAU AVENUE)
BROOKLYN	"BRICK PIZZERIA, NATHAN'S FAMOUS HOT DOGS" (CLARKSON AVENUE)
BROOKLYN	DOMINO'S (PITKIN AVENUE)
BROOKLYN	THE ORIGNAL PIZZA OF 4TH AVE (4 AVENUE)

File: [/output/Pizza22/part-r-00001](#)

Goto : go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

BRONX	M & R PIZZA (E 208TH ST)
BRONX	PABLO'S PIZZA (HUNTS POINT AVENUE)
BRONX	CESTRA'S PIZZA (EAST TREMONT AVENUE)
BRONX	ARTURO'S PIZZA (BROADWAY)
BRONX	LUKE'S PIZZA (SOUTHERN BOULEVARD)
BRONX	GOLDEN PIZZA (EAST 138 STREET)
BRONX	NICK'S PIZZA (EAST GUN HILL ROAD)
BRONX	PUGSLEY PIZZA (EAST 191 STREET)
BRONX	ANTHONY'S PIZZA (MORRIS PARK AVE)
BRONX	DOMINO'S (BOSTON ROAD)
BRONX	LJ PIZZA (JEROME AVENUE)

File: [/output/Pizza22/part-r-00003](#)

Goto : go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

QUEENS	SINGAS FAMOUS PIZZA (108 STREET)
QUEENS	CORONA PIZZA (108 STREET)
QUEENS	FRATELLI PIZZA (162 STREET)
QUEENS	FRESH MEADOW'S PIZZA & RESTAURANT (69 AVENUE)
QUEENS	ANGELO'S PIZZA (103 STREET)
QUEENS	MARIO'S PIZZERIA (101 AVENUE)
QUEENS	PAPA JOHN'S (ROCKAWAY BOULEVARD)
QUEENS	ALBA'S PIZZA (DITMARS BOULEVARD)
QUEENS	PAPA JOHN'S (GREENPOINT AVENUE)

File: [/output/Pizza22/part-r-00004](#)

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

STATEN ISLAND	CASA NINO (ARTHUR KILL ROAD)
STATEN ISLAND	ZZ'S PIZZA & GRILL (RICHMOND TERRACE)
STATEN ISLAND	TI AMO PIZZA (VICTORY BLVD)
STATEN ISLAND	TI AMO PIZZA (VICTORY BLVD)
STATEN ISLAND	BARI'S PIZZA & PASTA RESTAURANT (BAY STREET)
STATEN ISLAND	ALL STAR PIZZA (ARTHUR KILL ROAD)
STATEN ISLAND	SHARKEY'S SQUARE (HYLAN BLVD)
STATEN ISLAND	PI PIZZERIA (JEWETT AVE)
STATEN ISLAND	MICHELANGELO'S PIZZA (POST AVENUE)
STATEN ISLAND	PI PIZZERIA (JEWETT AVE)

- 8) MapReduce code for finding the zip code, per major area, with the least number of restaurants that have committed a critical violation

```
package shubham;
import org.apache.hadoop.conf.Configuration;

import java.util.ArrayList;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
//import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;
import java.util.StringTokenizer;
public class SafestZipCodes {

    public static class MyMapper extends Mapper<LongWritable, Text,Text, Text>{

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

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

            if(arr.length!=17 && arr.length!=18){
                System.out.println("EXEC" + arr.length + " " +
line);
            }
        }
    }
}
```

```

        if(!arr[2].equals("BORO") && !arr[2].equals("Missing"))
            context.write(new Text(arr[2]),new Text(line));
    }

}

public static class MyReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException{

        ArrayList<Integer> list = new ArrayList<Integer>();
        int[] store = new int[100000];
        for(Text val: values){
            String[] arr = val.toString().split("\t");
            String zipcode = arr[5];
            if(arr[12].equals("Critical")){
                int code = Integer.parseInt(zipcode);
                int find = contains(list,code);
                if(find!=-1){
                    list.add(code);
                    list.add(1);
                }
                else{
                    int freq=list.get(find+1);
                    list.set(find+1, freq+1);
                }
            }
        }

        int min=Integer.MAX_VALUE;
        int storeCode=0;
        int i=1;
        int sum=0;
        while(i<list.size()){
            int num=list.get(i);
            sum+=num;
            // System.out.println(num);
            if(num<min && num!=0){
                min=num;
                storeCode=list.get(i-1);
            }
            i=i+2;
        }
        // System.out.println(key.toString() + " " + sum);
        context.write(key, new Text(storeCode+"    Number of restaurants
with critical violations: " + min));
    }

    private int contains(ArrayList<Integer> list, int code){
        int i=0;
        while(i<list.size()){
            if(list.get(i)==code)
                return i;
            else
                i=i+2;
        }
    }
}

```

```

        return -1;
    }
}

public static void main(String[] args) throws Exception{
    Configuration conf= new Configuration();
    String[] otherArgs = new
GenericOptionsParser(conf,args).getRemainingArgs();
    Job job = new Job(conf, "Safest Zip Codes Per Area");

    job.setJarByClass(SafestZipCodes.class);
    job.setMapperClass(MyMapper.class);
    job.setReducerClass(MyReducer.class);

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

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

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

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

}

```

Testing:

```

package shubham;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.ReduceDriver;
import org.junit.Before;
import org.junit.Test;

import shubham.SafestZipCodes.MyMapper;
import shubham.SafestZipCodes.MyReducer;
import java.util.*;

public class ZipCodeTesting {

    MapReduceDriver<LongWritable,Text,Text,Text,Text,Text> mapReduceDriver;
    Mapper<LongWritable,Text,Text,Text> mapDriver;
    ReduceDriver<Text,Text,Text,Text> reduceDriver;

    @Before
    public void setUp() {

```

```

MyMapper mapper = new MyMapper();
MyReducer reducer = new MyReducer();

mapDriver = new MapDriver<LongWritable,Text,Text,Text>();
mapDriver.setMapper(mapper);

reduceDriver = new ReduceDriver<Text,Text,Text,Text>();
reduceDriver.setReducer(reducer);

mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text,
Text, Text, Text>();
mapReduceDriver.setMapper(mapper);
mapReduceDriver.setReducer(reducer);

}

@Test
public void testMapper(){

    mapDriver.withInput(new LongWritable(0), new Text("41571350 THE
MET GRILL/DOUBLE TREE HOTEL    MANHATTAN    569    LEXINGTON AVENUE    10022
2123506081    American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
Critical    33    8/21/2016    Cycle Inspection / Initial
Inspection"));

    mapDriver.withOutput(new Text("MANHATTAN"),new Text("41571350
THE MET GRILL/DOUBLE TREE HOTEL    MANHATTAN    569    LEXINGTON AVENUE
10022 2123506081    American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
Critical    33    8/21/2016    Cycle Inspection / Initial
Inspection"));

    mapDriver.runTest();

}

@Test
public void testReducer(){

    List<Text> values = new ArrayList<Text>();
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American
4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained.    Critical    33
8/21/2016    Cycle Inspection / Initial Inspection"));
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American
4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained.    Not Critical    33
8/21/2016    Cycle Inspection / Initial Inspection"));
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American
4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained.    Not Applicable    33
8/21/2016    Cycle Inspection / Initial Inspection"));
    values.add(new Text("41571350 THE MET GRILL/DOUBLE TREE HOTEL
MANHATTAN    569    LEXINGTON AVENUE    10022 2123506081    American

```



```

        4/8/2016    Violations were cited in the following area(s). 09C    Food
contact surface not properly maintained. 33                                8/21/2016    Cycle
Inspection / Initial Inspection"));

        reduceDriver.withInput(new Text("MANHATTAN"), values);
        reduceDriver.addOutput(new Text("MANHATTAN"), new Text(10022+"
Number of restaurants with critical violations: " + 1));

    }

    @Test
    public void testMapReduceSingleProduct(){

        mapReduceDriver.withInput(new LongWritable(0), new Text("41571350
THE MET GRILL/DOUBLE TREE HOTEL    MANHATTAN    569    LEXINGTON AVENUE
10022 2123506081    American    4/8/2016    Violations were cited in the
following area(s).    09C    Food contact surface not properly maintained.
Critical    33                                8/21/2016    Cycle Inspection / Initial
Inspection"));
        mapReduceDriver.addOutput(new Text("MANHATTAN"), new Text(10022+"
Number of restaurants with critical violations: " + 1));
        mapReduceDriver.runTest();
    }
}

```

Output:

File: [/output/ZipCodes/part-r-00000](#)

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

BRONX	11370	Number of restaurants with critical violations: 53
BROOKLYN	11256	Number of restaurants with critical violations: 5
MANHATTAN	10178	Number of restaurants with critical violations: 1
QUEENS	11451	Number of restaurants with critical violations: 21
STATEN ISLAND	10317	Number of restaurants with critical violations: 5

9) Finding the safest area to eat

- Let us use the output file (/output/CountPerArea/part-r-0000) to find the safest area to eat. The safest area to eat will essentially be the area where the probability of running into a restaurant that has committed a critical violation is the least.
- To copy the file into our input folder we will use the command : `hadoop fs -cp /output/CountPerArea/part-r-0000 /input`

MapReduce code for finding the safest area to eat and displaying the percentage of restaurants out of the total that have committed critical violations for that area (rounded to the nearest whole percentage)

```
package shubham;

import org.apache.hadoop.conf.Configuration;

import java.util.ArrayList;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import shubham.SafestZipCodes.MyMapper;
import shubham.SafestZipCodes.MyReducer;

import java.io.IOException;
import java.util.StringTokenizer;

public class SafestArea {

    public static class MyMapper extends Mapper<LongWritable, Text, Text, Text>{

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

            String line=value.toString();
                                int flag=0;

            String store ="";
            String s="";

            for(int i=0;i<line.length();i++){

                int a=line.charAt(i);
                if(line.charAt(i)==':' && flag==0){
                    flag = 1;
                }
                else if(line.charAt(i)==' ' && flag==1){
                    flag = 0;

                    store+=s+" ";

                    s="";

                    flag=1;
                }
                else if(a>=48 && a<=57){
```

```

        s+=line.charAt(i);
    }
}

store+=s;
String[] arr = line.split("\t");
System.out.println(arr[0]+" "+store +"print1");
context.write(new Text("Area"), new Text(arr[0] + " "
+store ));
    }
}

public static class MyReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
    throws IOException, InterruptedException{

        String area="";
        double min = Double.MAX_VALUE;
        for(Text val:values){
            System.out.println(val.toString()+ "print");
            String line = val.toString();
            String[] arr = line.split(" ");
            String loc="";
            int critical = 0;
            int nonC = 0;
            int noV = 0;
            if(arr.length==5){
                System.out.println("exec");
                loc=arr[0]+" "+arr[1];
                critical = Integer.parseInt(arr[2]);
                nonC = Integer.parseInt(arr[3]);
                noV = Integer.parseInt(arr[4]);
            }
            else{
                loc = arr[0];
                critical = Integer.parseInt(arr[1]);
                nonC = Integer.parseInt(arr[2]);
                noV = Integer.parseInt(arr[3]);
            }

            double total = (double)(critical + nonC + noV);
            double percent = ((double)(critical))/(double)(total))*100;

            if(percent<min){
                min = percent;
                area = loc;
            }
        }
        context.write(new Text(area), new Text(Math.round(min) + " "));
    }
}

public static void main(String[] args) throws Exception{
    Configuration conf= new Configuration();
    String[] otherArgs = new
    GenericOptionsParser(conf,args).getRemainingArgs();

```

```

Job job = new Job(conf, "Safest Area");

job.setJarByClass(SafestArea.class);
job.setMapperClass(MyMapper.class);
job.setReducerClass(MyReducer.class);

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

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

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

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

}

```

Testing Class:

```

package shubham;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.*;
import org.junit.Before;
import org.junit.Test;

import shubham.SafestArea.MyMapper;
import shubham.SafestArea.MyReducer;

import java.util.*;

public class SafestAreaTesting {

    MapReduceDriver<LongWritable,Text,Text,Text,Text,Text> mapReduceDriver;
    MapDriver<LongWritable,Text,Text,Text> mapDriver;
    ReduceDriver<Text,Text,Text,Text> reduceDriver;

    @Before
    public void setUp() {
        MyMapper mapper = new MyMapper();
        MyReducer reducer = new MyReducer();

        mapDriver = new MapDriver<LongWritable,Text,Text,Text>();
        mapDriver.setMapper(mapper);

        reduceDriver = new ReduceDriver<Text,Text,Text,Text>();
        reduceDriver.setReducer(reducer);
    }
}

```

```

        mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text,
Text, Text, Text>();
        mapReduceDriver.setMapper(mapper);
        mapReduceDriver.setReducer(reducer);

    }

    @Test
    public void testMapper(){
        mapDriver.withInput(new LongWritable(0), new Text("BRONX
Critical Violation: 21481  Non Critical Violations: 17552  No
Violations: 806"));
        mapDriver.withOutput(new Text("Area"),new Text("BRONX 21481 17552
806"));
        mapDriver.runTest();
    }
}

```

Output

File: [/output/percentage10/part-r-00000](#)

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

BRONX	54
-------	----