A Final Project Report on Analysis of ‘Chicago Crime’ Dataset

Bhavik Shah

Northeastern University

MS-IS (NU Id: 001825444)

<u>Summary</u>

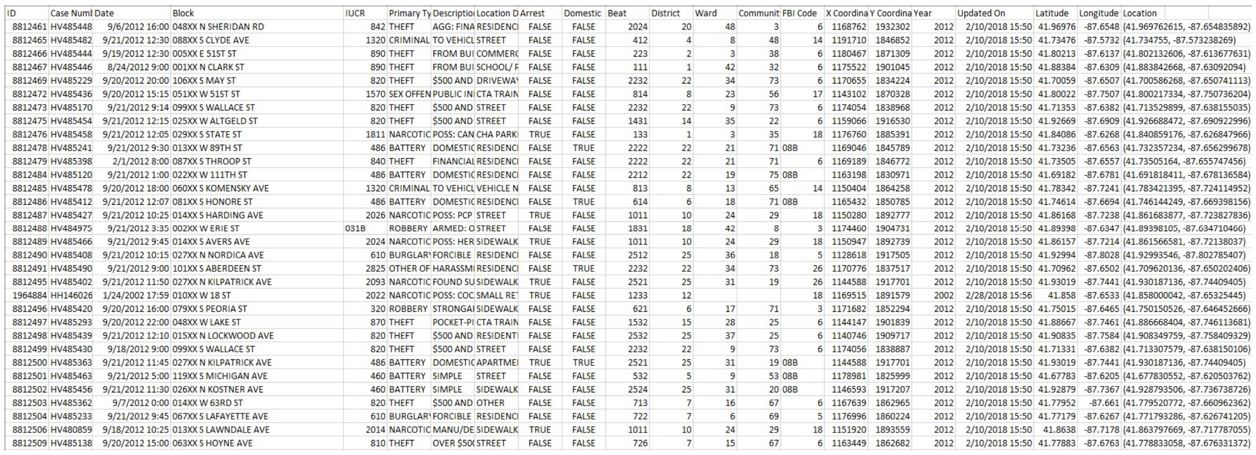
This project is based on the Analysis of the 'Chicago Crime' dataset. The raw dataset consists of crime incidents taking place over Chicago. The data span a period of more than 17 years, including all ~100,0000 crime incidents from 2001 up to 2018. Crime Incident includes crimes category, Location of category, FBI code, cases resolved, and a plaintext review.

In the project, first the analysis of the raw dataset using Power BI is done. The **MapReduce** analysis is performed using several MapReduce patterns such as Partitioning, Binning, Job Chaining etc. Also, in this project, **Apache pig, Apache Hive and Apache Mahout** is implemented.

The analysis on the dataset can help to determine and predict the pattern of crime happening over the years and awareness and improvement of public safety.

Dataset Link: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

<u>Chicago Crime Dataset:</u>



ID	Case Numl	Date	Block	IUCR	Primary Ty	Descriptioi	Location D	Arrest	Domestic	Beat	District	Ward	Communit	FBI Code	X Coordina	Y Coordina	Year	Updated On	Latitude	Longitude	Location
8812461	HV485448	9/6/2012 16:00	048XX N SHERIDAN RD	842	THEFT	AGG: FINA	RESIDENCI	FALSE	FALSE	2024	20	48	3	6	1168762	1932302	2012	2/10/2018 15:50	41.96976	-87.6548	(41.969762615, -87.654835892)
8812465	HV485482	9/21/2012 12:30	088XX S CLYDE AVE	1320	CRIMINAL	TO VEHICL	STREET	FALSE	FALSE	412	4	8	48	14	1191710	1846852	2012	2/10/2018 15:50	41.73476	-87.5732	(41.734755, -87.573238269)
8812466	HV485444	9/19/2012 12:30	005XX E 51ST ST	890	THEFT	FROM BUI	COMMERC	FALSE	FALSE	223	2	3	38	6	1180467	1871309	2012	2/10/2018 15:50	41.80213	-87.6137	(41.802132606, -87.613677631)
8812467	HV485446	8/24/2012 9:00	001XX N CLARK ST	890	THEFT	FROM BUI	SCHOOL/ P	FALSE	FALSE	111	1	42	32	6	1175522	1901045	2012	2/10/2018 15:50	41.88384	-87.6309	(41.883842668, -87.63092094)
8812469	HV485229	9/20/2012 20:00	106XX S MAY ST	820	THEFT	$500 AND	DRIVEWAY	FALSE	FALSE	2232	22	34	73	6	1170655	1834224	2012	2/10/2018 15:50	41.70059	-87.6507	(41.700586268, -87.650741113)
8812472	HV485436	9/20/2012 15:15	051XX W 51ST ST	1570	SEX OFFEN	PUBLIC INI	CTA TRAIN	FALSE	FALSE	814	8	23	56	17	1143102	1870328	2012	2/10/2018 15:50	41.80022	-87.7507	(41.800217334, -87.750736204)
8812473	HV485170	9/21/2012 9:14	099XX S WALLACE ST	820	THEFT	$500 AND	STREET	FALSE	FALSE	2232	22	9	73	6	1174054	1838968	2012	2/10/2018 15:50	41.71353	-87.6382	(41.713529899, -87.638155035)
8812475	HV485454	9/21/2012 12:15	025XX W ALTGELD ST	820	THEFT	$500 AND	STREET	FALSE	FALSE	1431	14	35	22	6	1159066	1916530	2012	2/10/2018 15:50	41.92669	-87.6909	(41.926688472, -87.690922996)
8812476	HV485458	9/21/2012 12:05	029XX S STATE ST	1811	NARCOTIC	POSS: CAN	CHA PARKI	TRUE	FALSE	133	1	3	35	18	1176760	1885391	2012	2/10/2018 15:50	41.84086	-87.6268	(41.840859176, -87.626847966)
8812478	HV485241	9/21/2012 9:30	013XX W 89TH ST	486	BATTERY	DOMESTIC	RESIDENCI	FALSE	TRUE	2222	22	21	71	08B	1169046	1845789	2012	2/10/2018 15:50	41.73236	-87.6563	(41.732357234, -87.656299678)
8812479	HV485398	2/1/2012 8:00	087XX S THROOP ST	840	THEFT	FINANCIAL	RESIDENCI	FALSE	FALSE	2222	22	21	71	6	1169189	1846772	2012	2/10/2018 15:50	41.73505	-87.6557	(41.73505164, -87.655747456)
8812484	HV485120	9/21/2012 1:00	022XX W 111TH ST	486	BATTERY	DOMESTIC	RESIDENCI	FALSE	FALSE	2212	22	19	75	08B	1163198	1830971	2012	2/10/2018 15:50	41.69182	-87.6781	(41.691818411, -87.678136584)
8812485	HV485478	9/20/2012 18:00	060XX S KOMENSKY AVE	1320	CRIMINAL	TO VEHICL	VEHICLE N	FALSE	FALSE	813	8	13	65	14	1150404	1864258	2012	2/10/2018 15:50	41.78342	-87.7241	(41.783421395, -87.724114952)
8812486	HV485412	9/21/2012 12:07	081XX S HONORE ST	486	BATTERY	DOMESTIC	RESIDENCI	FALSE	TRUE	614	6	18	71	08B	1165432	1850785	2012	2/10/2018 15:50	41.74614	-87.6694	(41.746144249, -87.669398156)
8812487	HV485427	9/21/2012 10:25	014XX S HARDING AVE	2026	NARCOTIC	POSS: PCP	STREET	TRUE	FALSE	1011	10	24	29	18	1150280	1892777	2012	2/10/2018 15:50	41.86168	-87.7238	(41.861683877, -87.723827836)
8812488	HV484975	9/21/2012 3:35	002XX W ERIE ST	031B	ROBBERY	ARMED: O	STREET	FALSE	FALSE	1831	18	42	8	3	1174460	1904731	2012	2/10/2018 15:50	41.89398	-87.6347	(41.89398105, -87.634710466)
8812489	HV485466	9/21/2012 9:45	014XX S AVERS AVE	2024	NARCOTIC	POSS: HER	SIDEWALK	TRUE	FALSE	1011	10	24	29	18	1150947	1892739	2012	2/10/2018 15:50	41.86157	-87.7214	(41.861566581, -87.72138037)
8812490	HV485408	9/21/2012 10:15	027XX N NORDICA AVE	610	BURGLARY	FORCIBLE	RESIDENCI	FALSE	FALSE	2512	25	36	18	5	1128618	1917505	2012	2/10/2018 15:50	41.92994	-87.8028	(41.92993546, -87.802785407)
8812491	HV485490	9/21/2012 9:00	101XX S ABERDEEN ST	2825	OTHER OF	HARASSMI	RESIDENCI	FALSE	TRUE	2232	22	34	73	26	1170776	1837517	2012	2/10/2018 15:50	41.70962	-87.6502	(41.709620136, -87.650202406)
8812495	HV485402	9/21/2012 11:50	027XX N KILPATRICK AVE	2093	NARCOTIC	FOUND SU	SIDEWALK	TRUE	FALSE	2521	25	31	19	26	1144588	1917701	2012	2/10/2018 15:50	41.93019	-87.7441	(41.930187136, -87.74409405)
1964884	HH146026	1/24/2002 17:59	010XX W 18 ST	2022	NARCOTIC	POSS: COC	SMALL RET	TRUE	FALSE	1233	12			18	1169515	1891579	2002	2/28/2018 15:56	41.858	-87.6533	(41.858000042, -87.65325445)
8812496	HV485420	9/20/2012 16:00	079XX S PEORIA ST	320	ROBBERY	STRONGAI	SIDEWALK	FALSE	FALSE	621	6	17	71	3	1171682	1852294	2012	2/10/2018 15:50	41.75015	-87.6465	(41.750150526, -87.646452666)
8812497	HV485293	9/20/2012 22:00	048XX W LAKE ST	870	THEFT	POCKET-PI	CTA TRAIN	FALSE	FALSE	1532	15	28	25	6	1144147	1901839	2012	2/10/2018 15:50	41.88667	-87.7461	(41.886668404, -87.746113681)
8812498	HV485439	9/21/2012 12:10	015XX N LOCKWOOD AVE	820	THEFT	$500 AND	RESIDENTI	FALSE	FALSE	2532	25	37	25	6	1140746	1909717	2012	2/10/2018 15:50	41.90835	-87.7584	(41.908349759, -87.758409329)
8812499	HV485430	9/18/2012 9:00	099XX S WALLACE ST	820	THEFT	$500 AND	STREET	FALSE	FALSE	2232	22	9	73	6	1174056	1838887	2012	2/10/2018 15:50	41.71331	-87.6382	(41.713307579, -87.638150106)
8812500	HV485363	9/21/2012 11:45	027XX N KILPATRICK AVE	486	BATTERY	DOMESTIC	APARTMEI	TRUE	TRUE	2521	25	31	19	08B	1144588	1917701	2012	2/10/2018 15:50	41.93019	-87.7441	(41.930187136, -87.74409405)
8812501	HV485463	9/21/2012 5:00	119XX S MICHIGAN AVE	460	BATTERY	SIMPLE	STREET	FALSE	FALSE	532	5	9	53	08B	1178981	1825999	2012	2/10/2018 15:50	41.67783	-87.6205	(41.677830552, -87.620503762)
8812502	HV485456	9/21/2012 11:30	026XX N KOSTNER AVE	460	BATTERY	SIMPLE	SIDEWALK	FALSE	FALSE	2524	25	31	20	08B	1146593	1917207	2012	2/10/2018 15:50	41.92879	-87.7367	(41.928793506, -87.736738726)
8812503	HV485362	9/7/2012 0:00	014XX W 63RD ST	820	THEFT	$500 AND	OTHER	FALSE	FALSE	713	7	16	67	6	1167639	1862965	2012	2/10/2018 15:50	41.77952	-87.661	(41.779520772, -87.660962362)
8812504	HV485233	9/21/2012 9:45	067XX S LAFAYETTE AVE	610	BURGLARY	FORCIBLE	RESIDENCI	FALSE	FALSE	722	7	6	69	5	1176996	1860224	2012	2/10/2018 15:50	41.77179	-87.6267	(41.771793286, -87.626741205)
8812506	HV480859	9/18/2012 10:25	013XX S LAWNDALE AVE	2014	NARCOTIC	MANU/DE	SIDEWALK	TRUE	FALSE	1011	10	24	29	18	1151920	1893559	2012	2/10/2018 15:50	41.8638	-87.7178	(41.863797669, -87.717787055)
8812509	HV485138	9/20/2012 15:00	063XX S HOYNE AVE	810	THEFT	OVER $50(	STREET	FALSE	FALSE	726	7	15	67	6	1163449	1862682	2012	2/10/2018 15:50	41.77883	-87.6763	(41.778833058, -87.676331372)

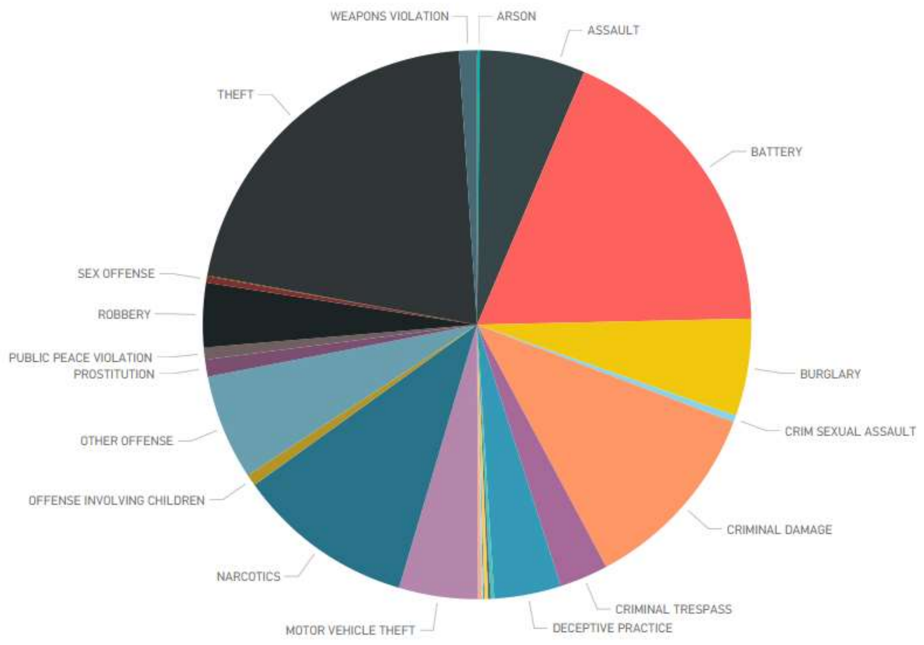
I have used MapReduce, Hive, Pig and with various algorithm as stated below:

a. Counter
b. Average
c. Percentage
d. Inverted Index
e. Secondary Sorting
f. Binning
g. Partitioning Pruning
h. Job Chaining
i. Top K filter
j. Bloom Filter
k. Normal Regex Filter

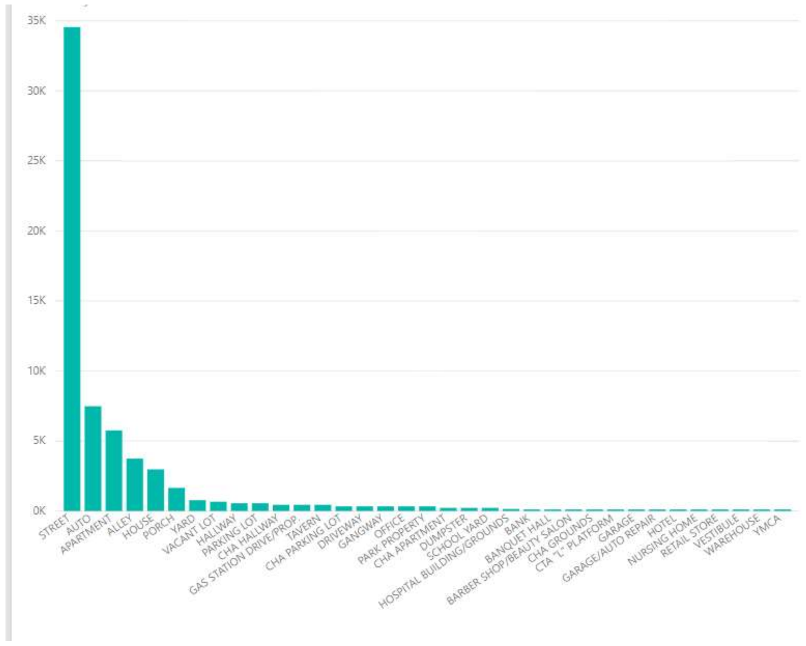
<u>Data Analysis using PowerBi:</u>

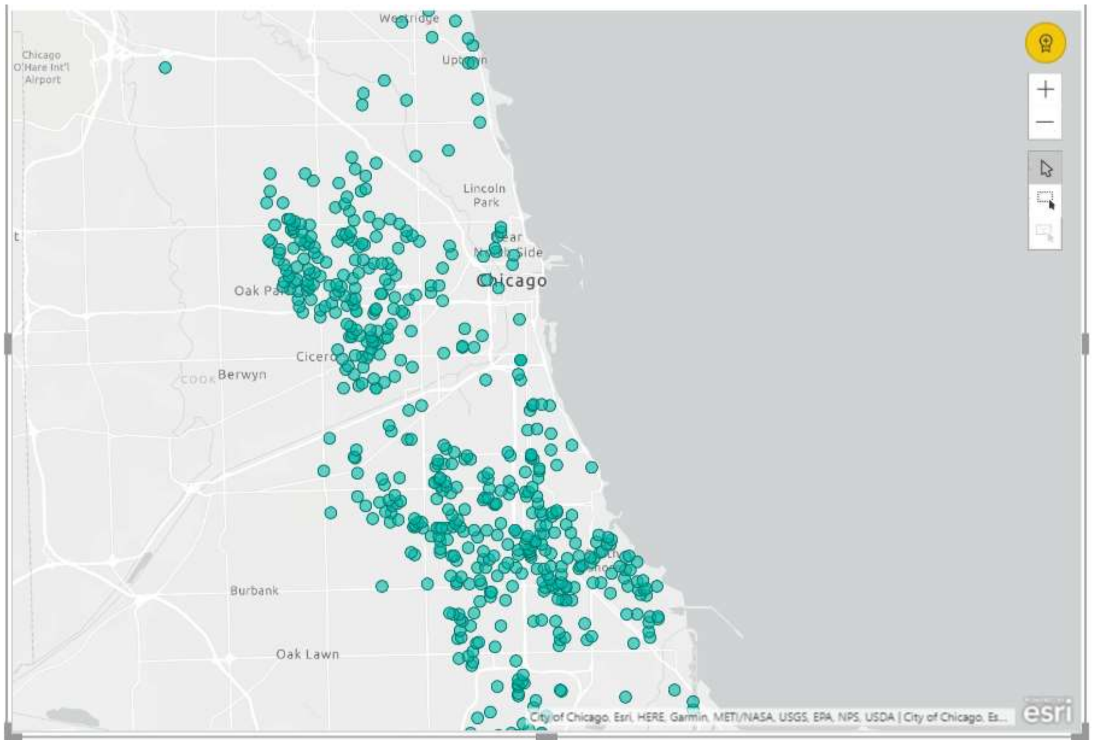
Please find all the implementation code in appendix section.

1. How many different types of crimes are reported?
   This analysis has been achieved through summarization algorithm in mapreduce.

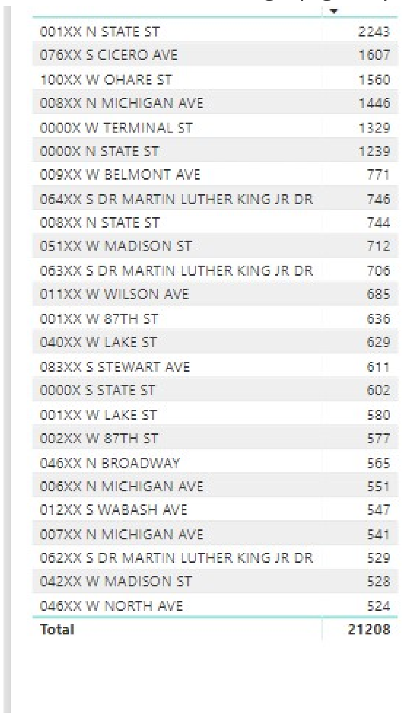


2. Filters based on crime: Analysis of crime Homicide in different areas of Chicago:
   This analysis is performed using mapregex filter in mapreduce



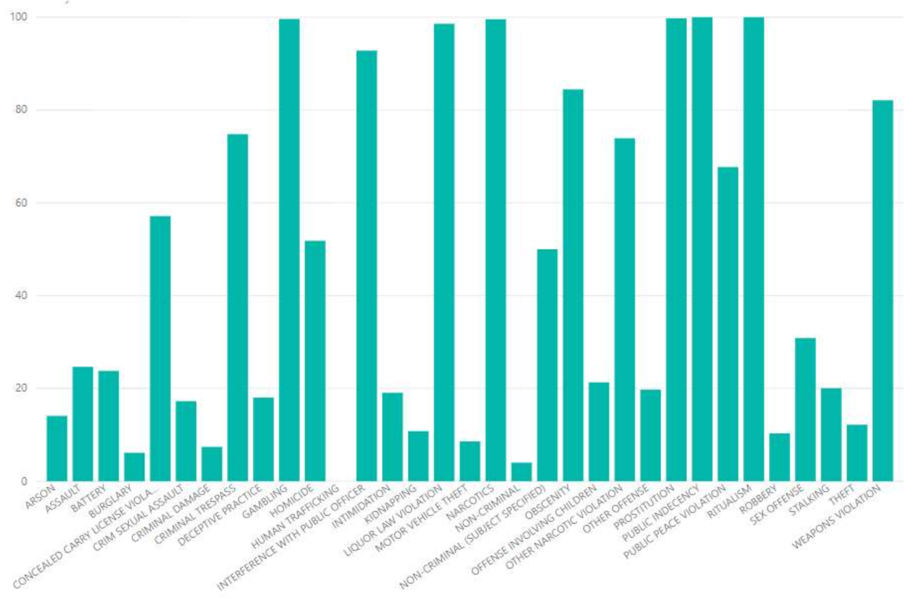

Chicago O'Hare Int'l Airport
Westridge
Uptown
Lincoln Park
Near North Side
Chicago
Oak Park
Cicero
Berwyn
COOK
Burbank
Oak Lawn
City of Chicago, Esri, HERE, Garmin, METI/NASA, USGS, EPA, NPS, USDA | City of Chicago, Es...
esri

3. Top 25 blocks that has highest number of crime incidents.
   This is achieved through pig script.

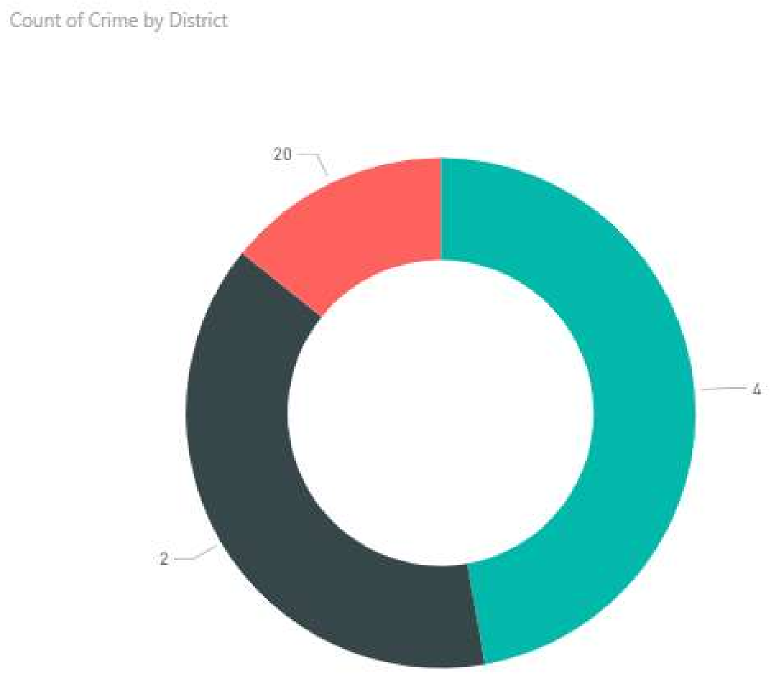


001XX N STATE ST	2243
076XX S CICERO AVE	1607
100XX W OHARE ST	1560
008XX N MICHIGAN AVE	1446
0000X W TERMINAL ST	1329
0000X N STATE ST	1239
009XX W BELMONT AVE	771
064XX S DR MARTIN LUTHER KING JR DR	746
008XX N STATE ST	744
051XX W MADISON ST	712
063XX S DR MARTIN LUTHER KING JR DR	706
011XX W WILSON AVE	685
001XX W 87TH ST	636
040XX W LAKE ST	629
083XX S STEWART AVE	611
0000X S STATE ST	602
001XX W LAKE ST	580
002XX W 87TH ST	577
046XX N BROADWAY	565
006XX N MICHIGAN AVE	551
012XX S WABASH AVE	547
007XX N MICHIGAN AVE	541
062XX S DR MARTIN LUTHER KING JR DR	529
042XX W MADISON ST	528
046XX W NORTH AVE	524
Total	**21208**

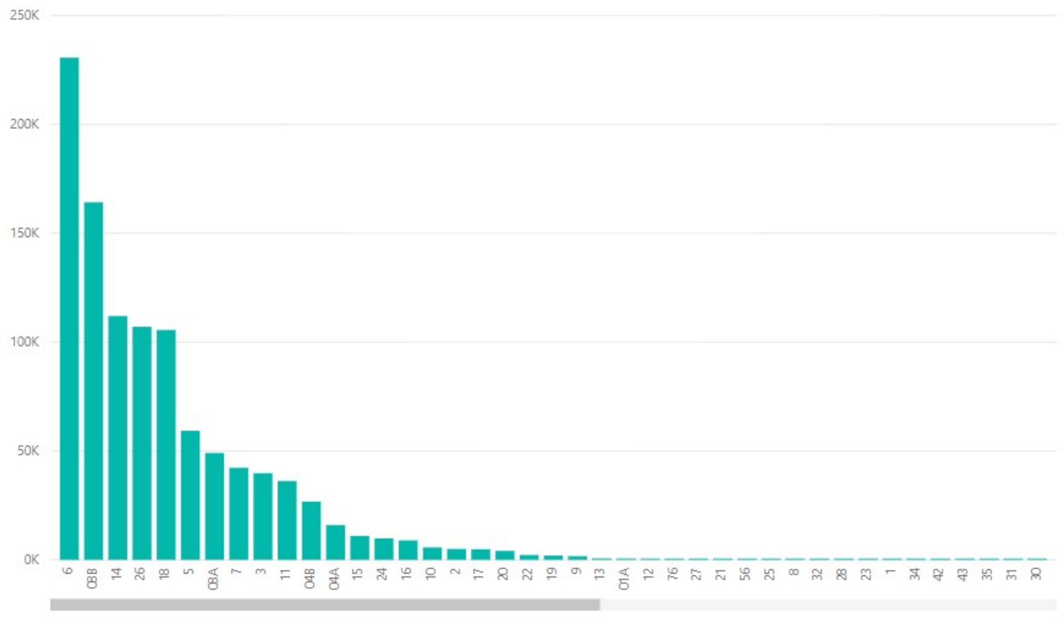
4. Percentage of cases resolved under each crime categories (arrest percentage).
   This is achieved through custom writable class in mapreduce calculating percentage of arrest



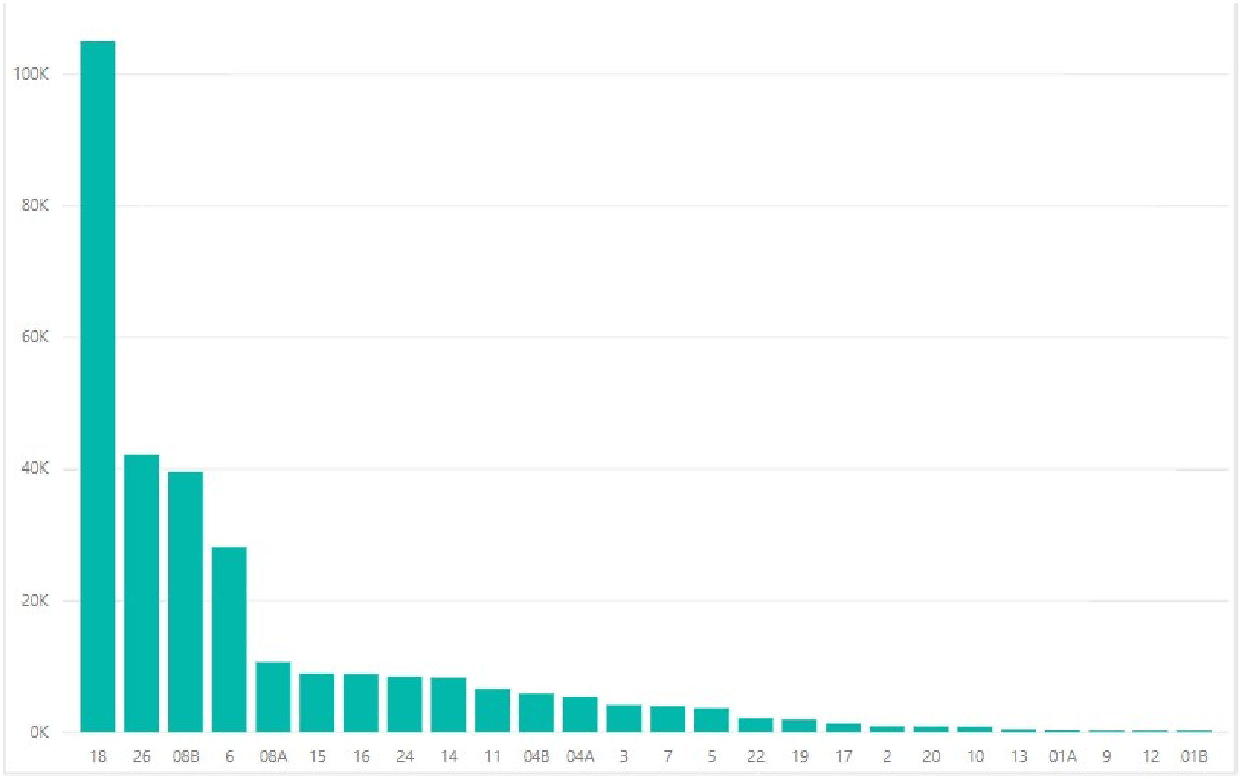
5. Count of crime filtered through district.
   This is achieved using bloom filter where 3 district value has been passed to bloom filter



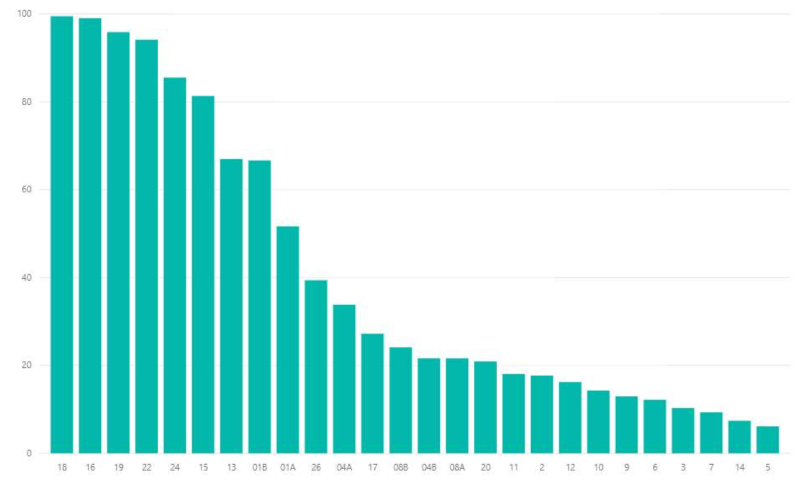
6. Total cases with FBI.
   This is achieved in hive where we analyzed total cases with FBI.



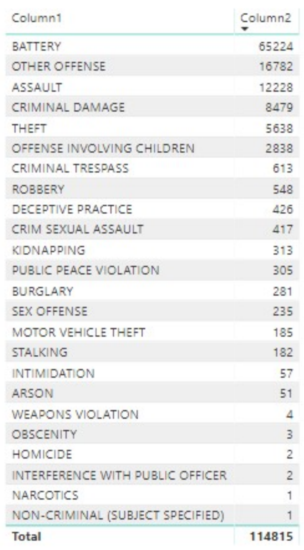
7. Total cases resolved by FBI overall
   This is achieved in hive using filter on cases resolved value = true.



8. Cases solved in percentage:
   This analysis uses job chaining algorithm where first job calculates the percentage of arrest and 2nd job sorts the data in descending order.

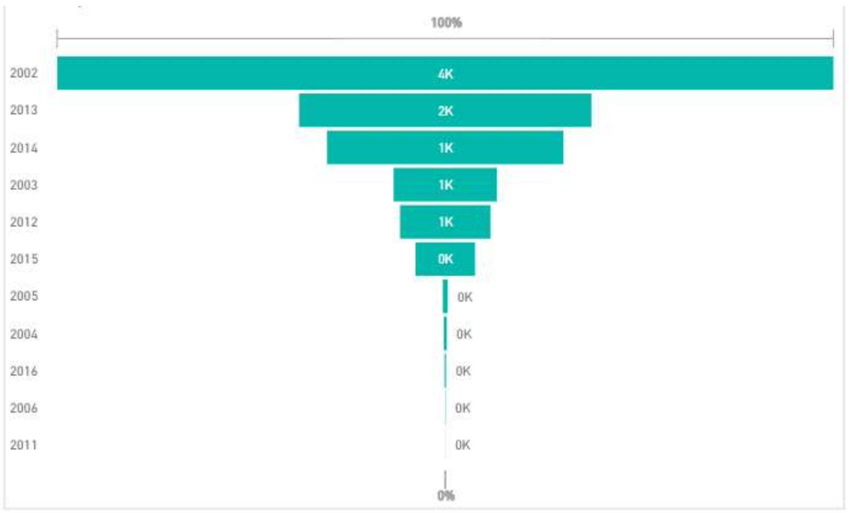


9. Domestic cases which are not resolved
   This is achieved in Hive script using filters on domestic values and arrest values.



Column1	Column2
BATTERY	65224
OTHER OFFENSE	16782
ASSAULT	12228
CRIMINAL DAMAGE	8479
THEFT	5638
OFFENSE INVOLVING CHILDREN	2838
CRIMINAL TRESPASS	613
ROBBERY	548
DECEPTIVE PRACTICE	426
CRIM SEXUAL ASSAULT	417
KIDNAPPING	313
PUBLIC PEACE VIOLATION	305
BURGLARY	281
SEX OFFENSE	235
MOTOR VEHICLE THEFT	185
STALKING	182
INTIMIDATION	57
ARSON	51
WEAPONS VIOLATION	4
OBSCENITY	3
HOMICIDE	2
INTERFERENCE WITH PUBLIC OFFICER	2
NARCOTICS	1
NON-CRIMINAL (SUBJECT SPECIFIED)	1
Total	**114815**

10. Crime analysis based on year
    This is achieved in pig through secondary sorting algorithm.
    First data is grouped with crime category and year and then flattening the group parameter to achieve sorting first on crime category and then on year.



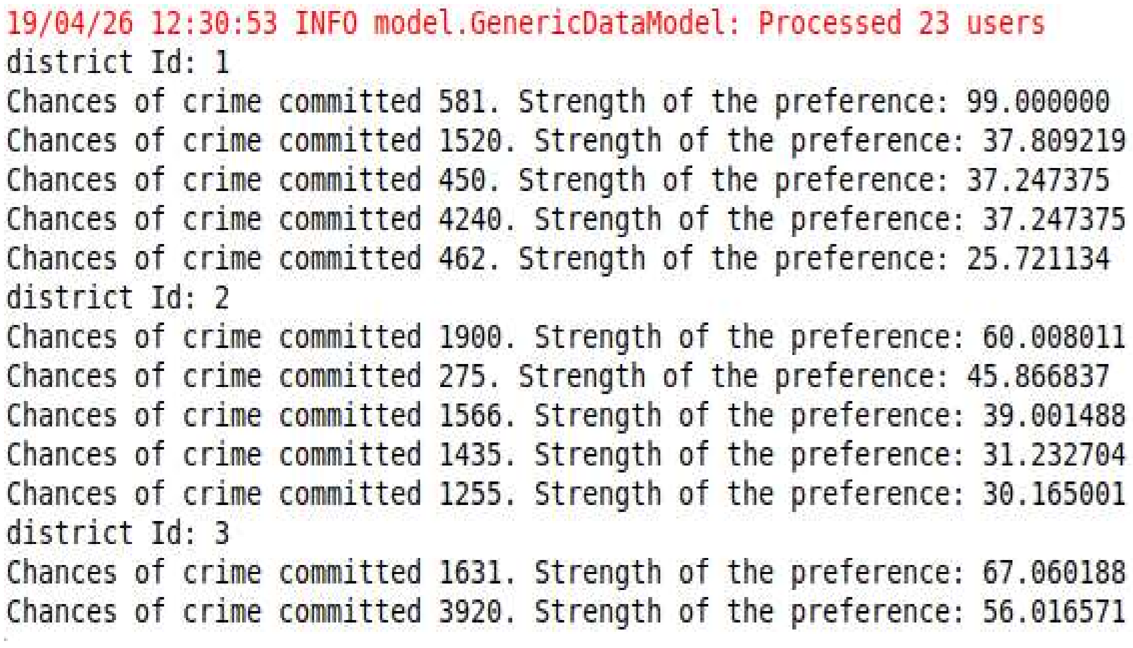
Above analysis has been achieved through MapReduce, Hive, Pig and with various algorithm as stated below:

l. Counter
m. Average
n. Percentage
o. Inverted Index
p. Secondary Sorting
q. Binning
r. Partitioning Pruning
s. Job Chaining
t. Top K filter
u. Bloom Filter
v. Normal Regex Filter

I have implemented all the above algorithm using Map-reduce pig and hive.

Apart from these algorithms, mahout has also been used to predict crime that can happen in different districts based on **mahout's recommendation** algorithm framework.

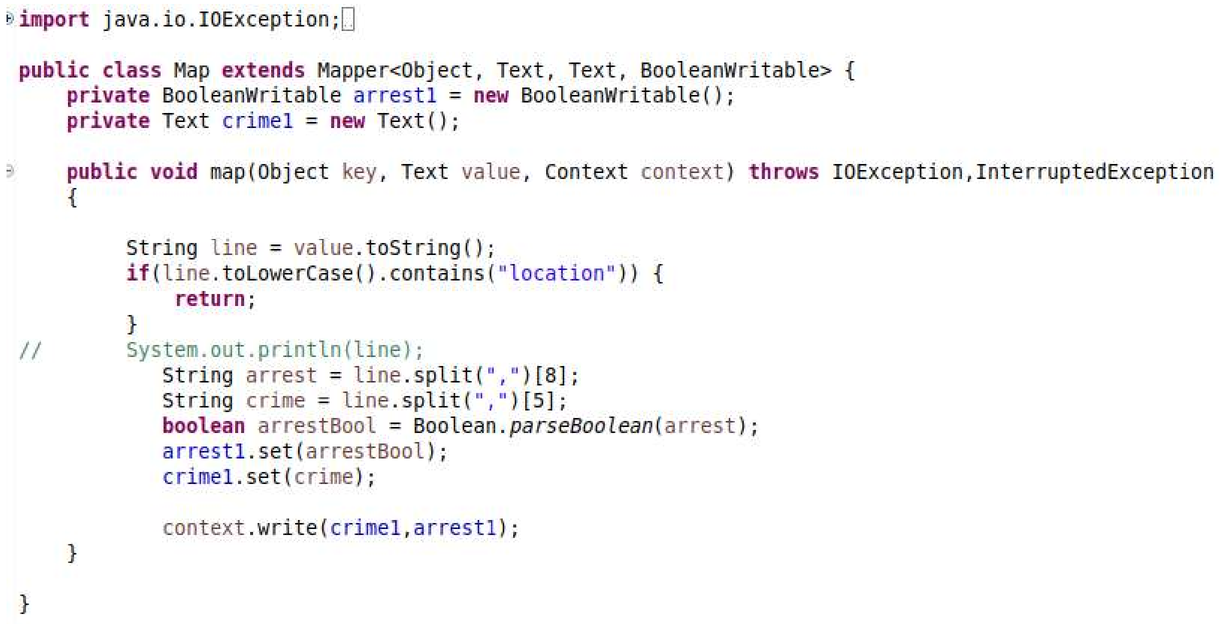
Below is the snapshot which says district 1 has 99% of chance that its unsafe for Kidnapping and should avoid playing in night in that area.


```
19/04/26 12:30:53 INFO model.GenericDataModel: Processed 23 users
district Id: 1
Chances of crime committed 581. Strength of the preference: 99.000000
Chances of crime committed 1520. Strength of the preference: 37.809219
Chances of crime committed 450. Strength of the preference: 37.247375
Chances of crime committed 4240. Strength of the preference: 37.247375
Chances of crime committed 462. Strength of the preference: 25.721134
district Id: 2
Chances of crime committed 1900. Strength of the preference: 60.008011
Chances of crime committed 275. Strength of the preference: 45.866837
Chances of crime committed 1566. Strength of the preference: 39.001488
Chances of crime committed 1435. Strength of the preference: 31.232704
Chances of crime committed 1255. Strength of the preference: 30.165001
district Id: 3
Chances of crime committed 1631. Strength of the preference: 67.060188
Chances of crime committed 3920. Strength of the preference: 56.016571
```

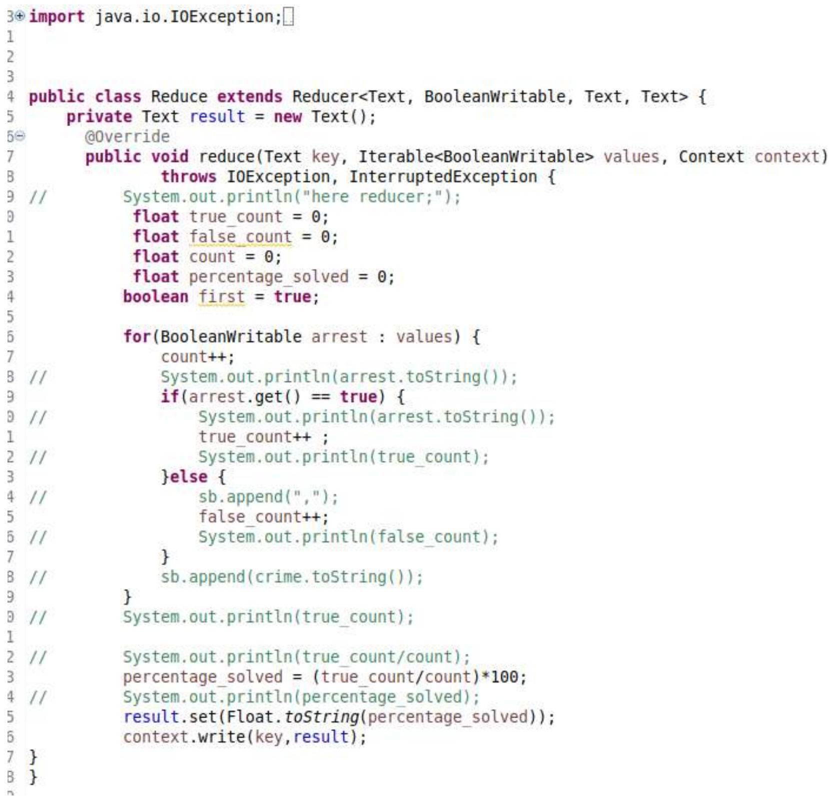
Appendix: Implementation Code:

Analysis using MapReduce:

1. **Percentage of Arrest:**
   a. Map Class


```
import java.io.IOException;

public class Map extends Mapper<Object, Text, Text, BooleanWritable> {
    private BooleanWritable arrest1 = new BooleanWritable();
    private Text crime1 = new Text();

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

        String line = value.toString();
        if(line.toLowerCase().contains("location")) {
            return;
        }
//      System.out.println(line);
           String arrest = line.split(",")[8];
           String crime = line.split(",")[5];
           boolean arrestBool = Boolean.parseBoolean(arrest);
           arrest1.set(arrestBool);
           crime1.set(crime);

           context.write(crime1,arrest1);
    }

}
```

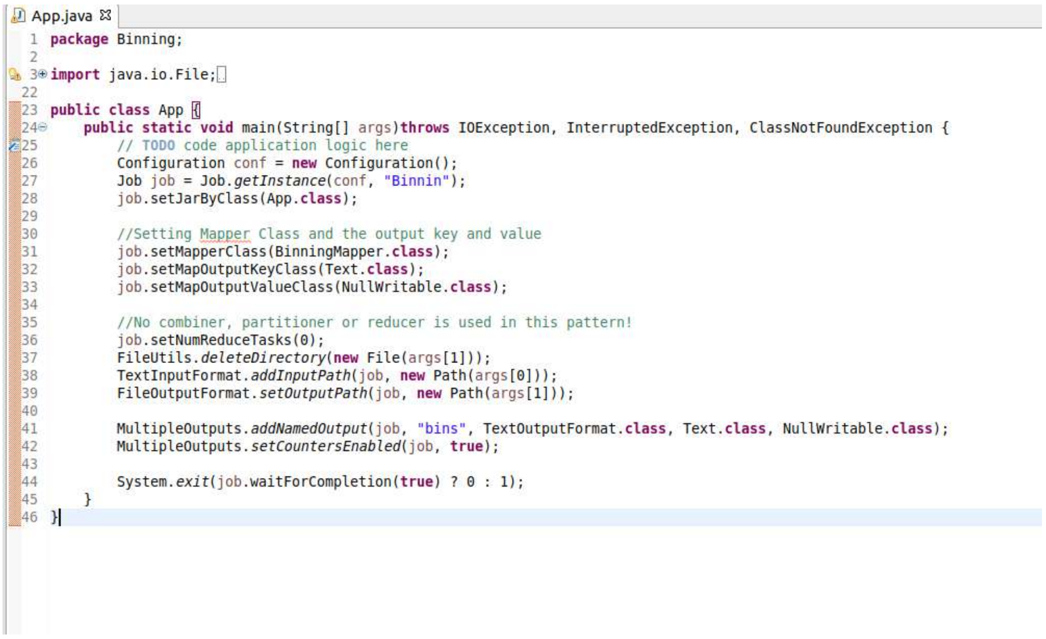
b. Reduce Class



```java
import java.io.IOException;



public class Reduce extends Reducer<Text, BooleanWritable, Text, Text> {
    private Text result = new Text();
      @Override
      public void reduce(Text key, Iterable<BooleanWritable> values, Context context)
              throws IOException, InterruptedException {
//          System.out.println("here reducer;");
             float true_count = 0;
             float false_count = 0;
             float count = 0;
             float percentage_solved = 0;
            boolean first = true;

            for(BooleanWritable arrest : values) {
                count++;
//              System.out.println(arrest.toString());
                if(arrest.get() == true) {
//                  System.out.println(arrest.toString());
                    true_count++ ;
//                  System.out.println(true_count);
                }else {
//                  sb.append(",");
                    false_count++;
//                  System.out.println(false_count);
                }
//              sb.append(crime.toString());
            }
//          System.out.println(true_count);

//          System.out.println(true_count/count);
            percentage_solved = (true_count/count)*100;
//          System.out.println(percentage_solved);
            result.set(Float.toString(percentage_solved));
            context.write(key,result);
}
}
```

2. **Binning Class based on different crimes category:**

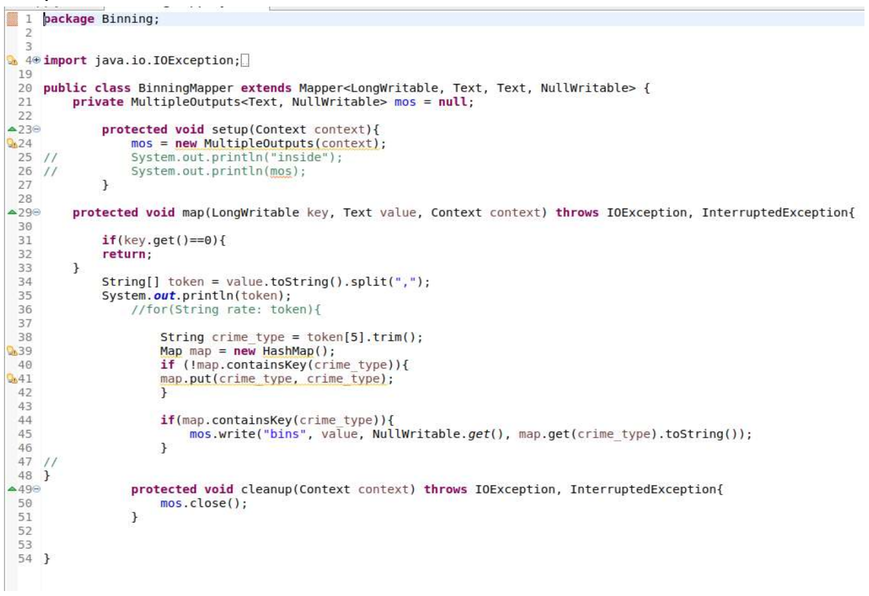
a. Driver class:



App.java

```java
package Binning;

import java.io.File;

public class App {
    public static void main(String[] args)throws IOException, InterruptedException, ClassNotFoundException {
        // TODO code application logic here
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Binnin");
        job.setJarByClass(App.class);

        //Setting Mapper Class and the output key and value
        job.setMapperClass(BinningMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(NullWritable.class);

        //No combiner, partitioner or reducer is used in this pattern!
        job.setNumReduceTasks(0);
        FileUtils.deleteDirectory(new File(args[1]));
        TextInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

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

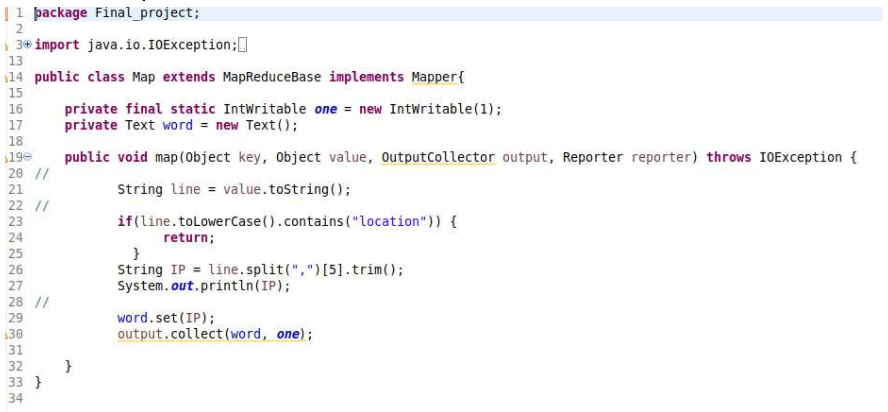
b. Map Class:



```
package Binning;


import java.io.IOException;

public class BinningMapper extends Mapper<LongWritable, Text, Text, NullWritable> {
    private MultipleOutputs<Text, NullWritable> mos = null;

        protected void setup(Context context){
            mos = new MultipleOutputs(context);
//          System.out.println("inside");
//          System.out.println(mos);
        }

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

        if(key.get()==0){
        return;
    }
        String[] token = value.toString().split(",");
        System.out.println(token);
            //for(String rate: token){

                String crime_type = token[5].trim();
                Map map = new HashMap();
                if (!map.containsKey(crime_type)){
                map.put(crime_type, crime_type);
                }

                if(map.containsKey(crime_type)){
                    mos.write("bins", value, NullWritable.get(), map.get(crime_type).toString());
                }
//
}
            protected void cleanup(Context context) throws IOException, InterruptedException{
                mos.close();
            }



}
```

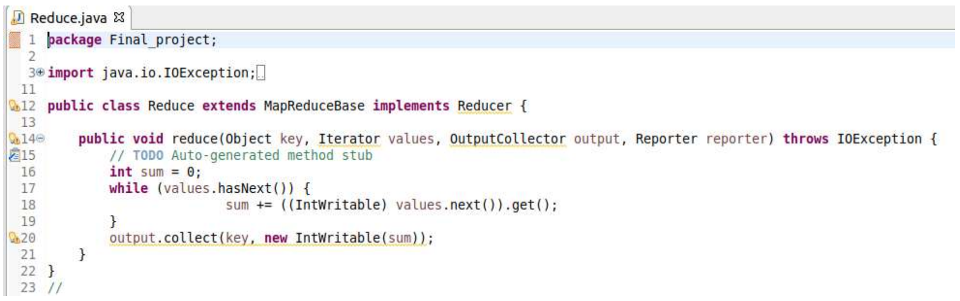
3. Count of different crime categories:

a. Map Class



```
package Final_project;

import java.io.IOException;

public class Map extends MapReduceBase implements Mapper{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Object value, OutputCollector output, Reporter reporter) throws IOException {
//
           String line = value.toString();
//
           if(line.toLowerCase().contains("location")) {
                 return;
             }
           String IP = line.split(",")[5].trim();
           System.out.println(IP);
//
           word.set(IP);
           output.collect(word, one);

    }
}

```

b. Reduce Class:

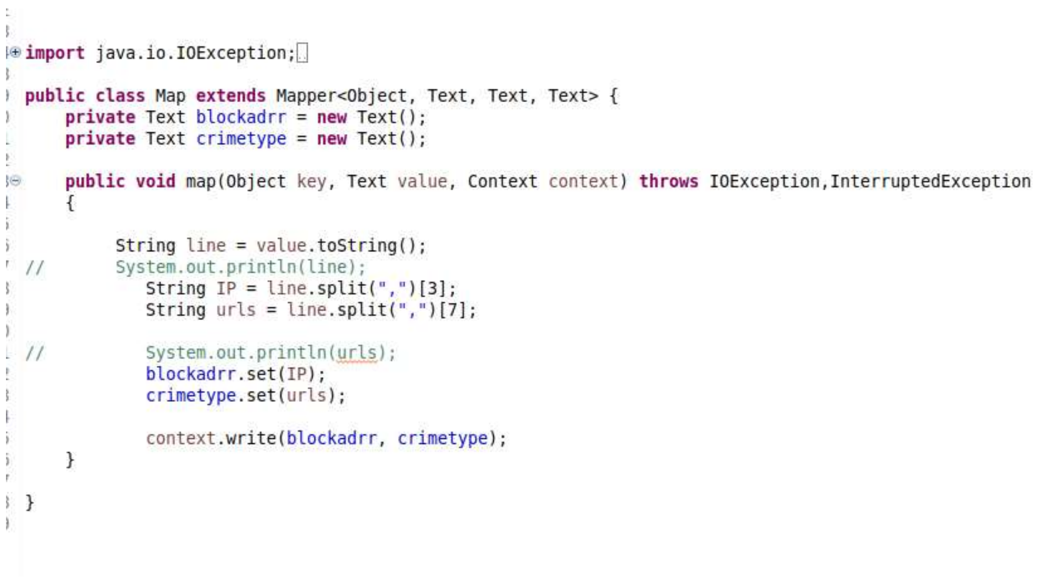


Reduce.java

```
package Final_project;

import java.io.IOException;

public class Reduce extends MapReduceBase implements Reducer {

    public void reduce(Object key, Iterator values, OutputCollector output, Reporter reporter) throws IOException {
        // TODO Auto-generated method stub
        int sum = 0;
        while (values.hasNext()) {
                       sum += ((IntWritable) values.next()).get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
//
```

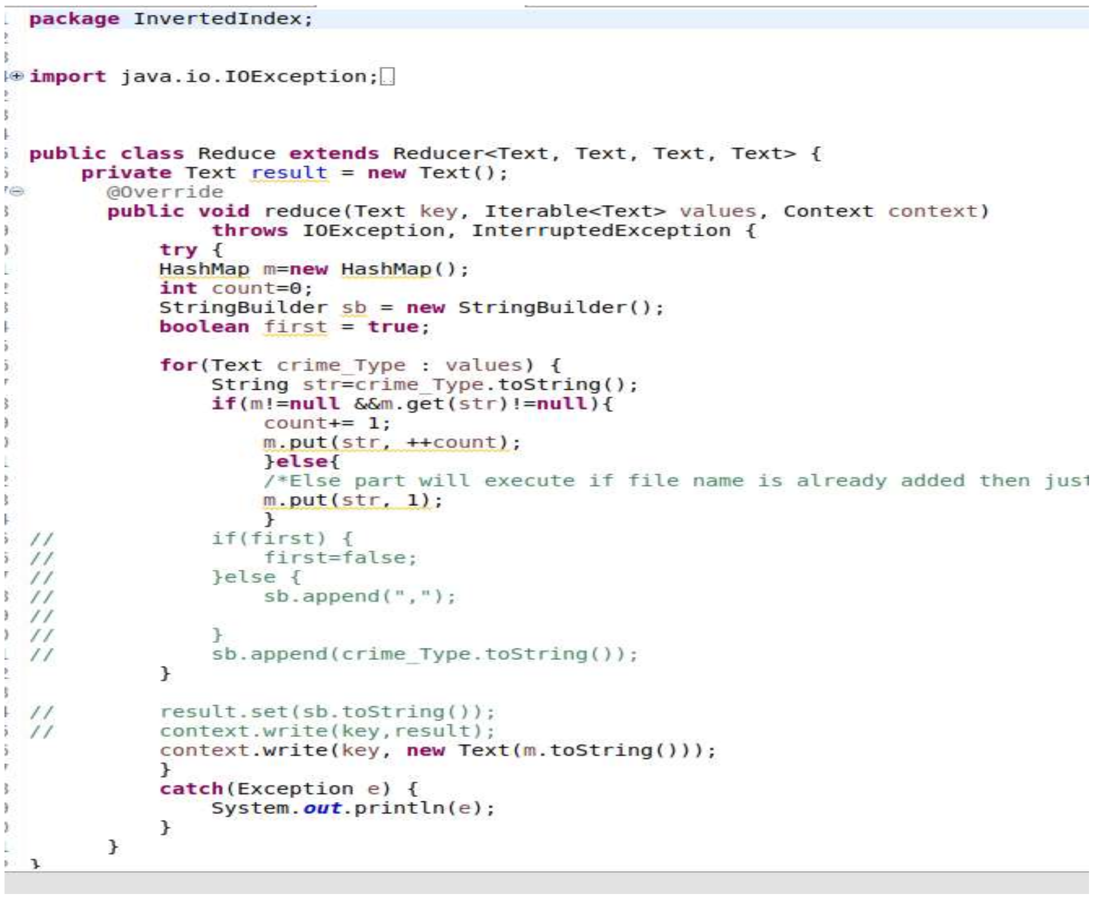
4. Inverted Index for street vs location search

a. Map Class:



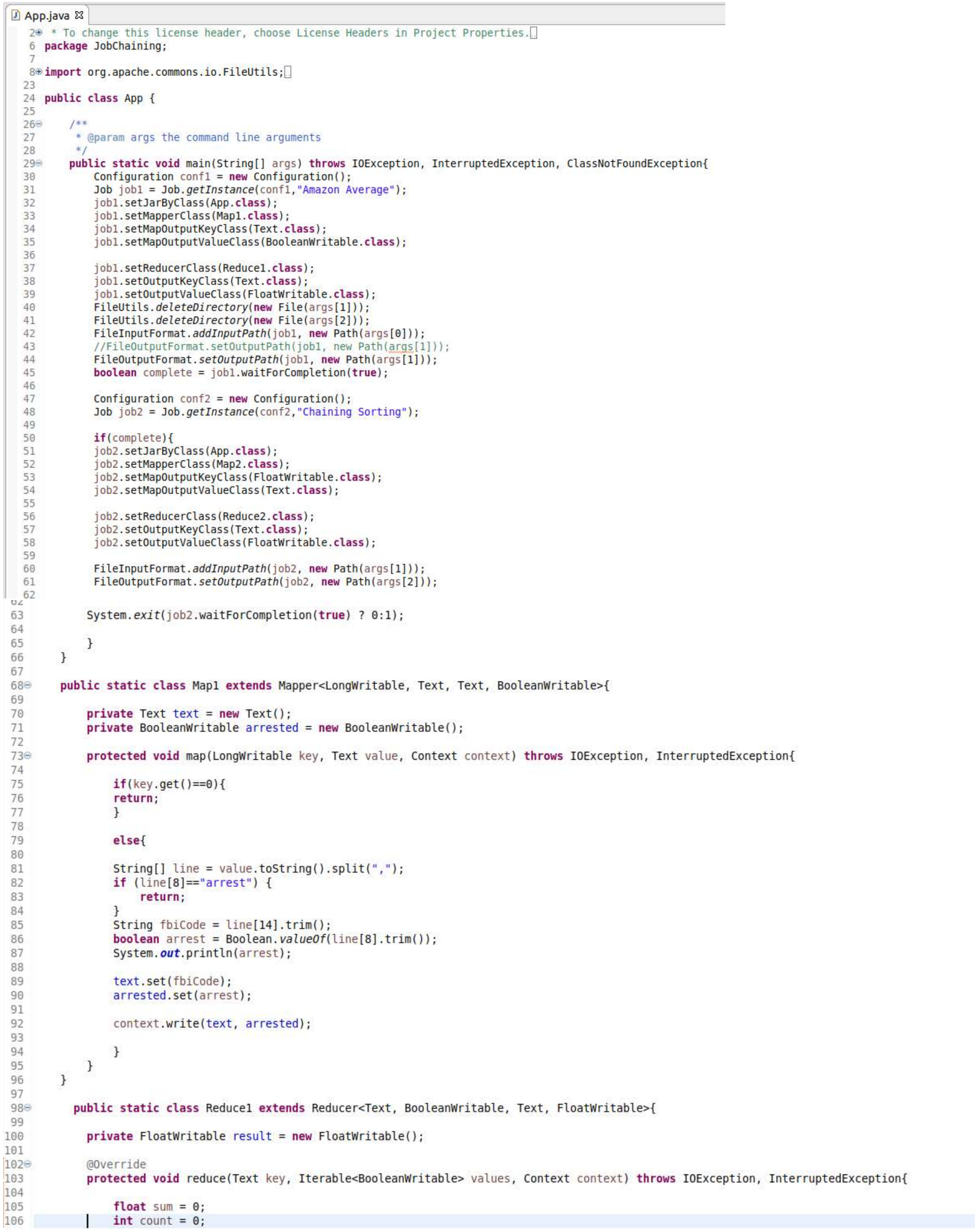
```java
import java.io.IOException;

public class Map extends Mapper<Object, Text, Text, Text> {
    private Text blockadrr = new Text();
    private Text crimetype = new Text();

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

        String line = value.toString();
//      System.out.println(line);
            String IP = line.split(",")[3];
            String urls = line.split(",")[7];

//          System.out.println(urls);
            blockadrr.set(IP);
            crimetype.set(urls);

            context.write(blockadrr, crimetype);
    }

}
```

b. Reducer class:



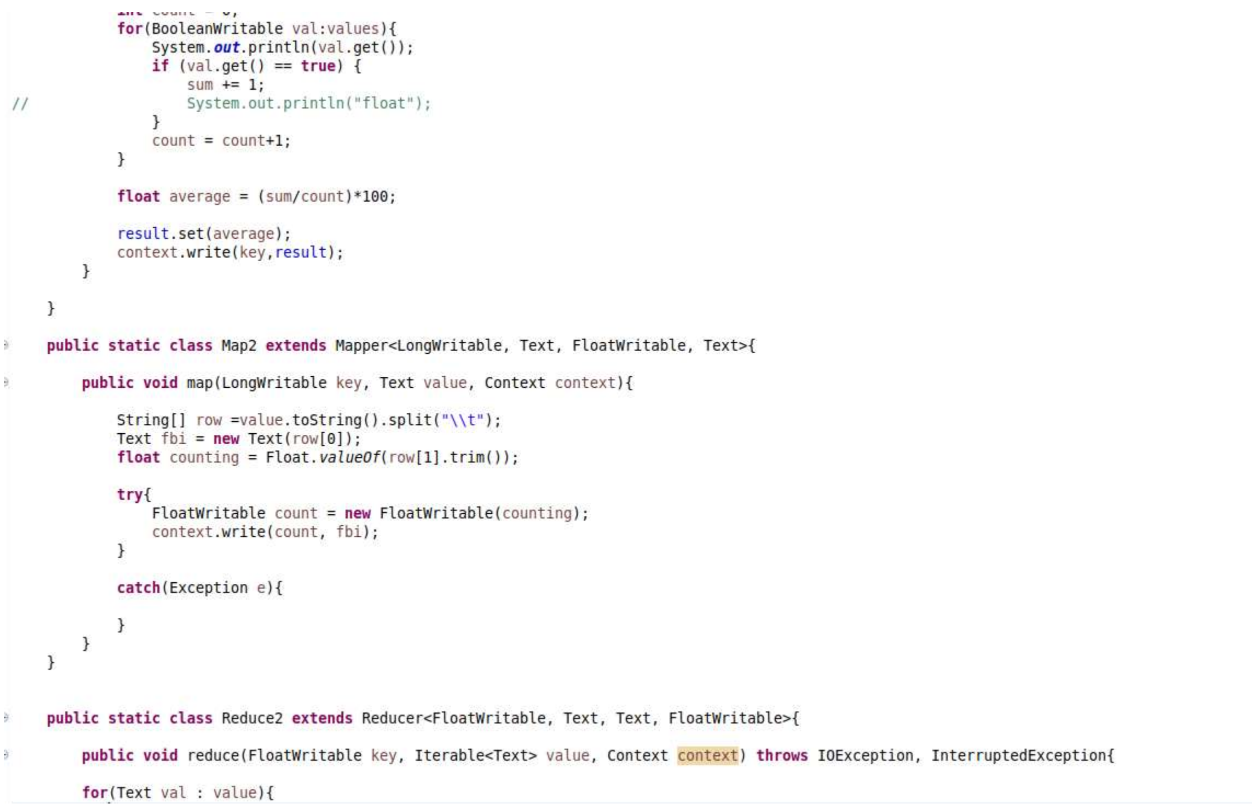
```java
package InvertedIndex;

import java.io.IOException;


public class Reduce extends Reducer<Text, Text, Text, Text> {
    private Text result = new Text();
      @Override
      public void reduce(Text key, Iterable<Text> values, Context context)
              throws IOException, InterruptedException {
          try {
          HashMap m=new HashMap();
          int count=0;
          StringBuilder sb = new StringBuilder();
          boolean first = true;

          for(Text crime_Type : values) {
              String str=crime_Type.toString();
              if(m!=null &&m.get(str)!=null){
                  count+= 1;
                  m.put(str, ++count);
                  }else{
                  /*Else part will execute if file name is already added then just
                  m.put(str, 1);
                  }
//            if(first) {
//                first=false;
//            }else {
//                sb.append(",");
//
//            }
//            sb.append(crime_Type.toString());
          }

//        result.set(sb.toString());
//        context.write(key,result);
          context.write(key, new Text(m.toString()));
          }
          catch(Exception e) {
              System.out.println(e);
          }
      }
}
```

5.Job Chaining:

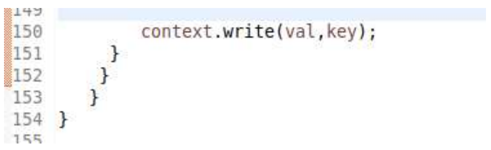


App.java

```java
 * To change this license header, choose License Headers in Project Properties.
package JobChaining;

import org.apache.commons.io.FileUtils;

public class App {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException{
        Configuration conf1 = new Configuration();
        Job job1 = Job.getInstance(conf1,"Amazon Average");
        job1.setJarByClass(App.class);
        job1.setMapperClass(Map1.class);
        job1.setMapOutputKeyClass(Text.class);
        job1.setMapOutputValueClass(BooleanWritable.class);

        job1.setReducerClass(Reduce1.class);
        job1.setOutputKeyClass(Text.class);
        job1.setOutputValueClass(FloatWritable.class);
        FileUtils.deleteDirectory(new File(args[1]));
        FileUtils.deleteDirectory(new File(args[2]));
        FileInputFormat.addInputPath(job1, new Path(args[0]));
        //FileOutputFormat.setOutputPath(job1, new Path(args[1]));
        FileOutputFormat.setOutputPath(job1, new Path(args[1]));
        boolean complete = job1.waitForCompletion(true);

        Configuration conf2 = new Configuration();
        Job job2 = Job.getInstance(conf2,"Chaining Sorting");

        if(complete){
        job2.setJarByClass(App.class);
        job2.setMapperClass(Map2.class);
        job2.setMapOutputKeyClass(FloatWritable.class);
        job2.setMapOutputValueClass(Text.class);

        job2.setReducerClass(Reduce2.class);
        job2.setOutputKeyClass(Text.class);
        job2.setOutputValueClass(FloatWritable.class);

        FileInputFormat.addInputPath(job2, new Path(args[1]));
        FileOutputFormat.setOutputPath(job2, new Path(args[2]));

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

        }
    }

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

        private Text text = new Text();
        private BooleanWritable arrested = new BooleanWritable();

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

            if(key.get()==0){
            return;
            }

            else{

            String[] line = value.toString().split(",");
            if (line[8]=="arrest") {
                return;
            }
            String fbiCode = line[14].trim();
            boolean arrest = Boolean.valueOf(line[8].trim());
            System.out.println(arrest);

            text.set(fbiCode);
            arrested.set(arrest);

            context.write(text, arrested);

            }
        }
    }

      public static class Reduce1 extends Reducer<Text, BooleanWritable, Text, FloatWritable>{

        private FloatWritable result = new FloatWritable();

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

            float sum = 0;
            int count = 0;
```



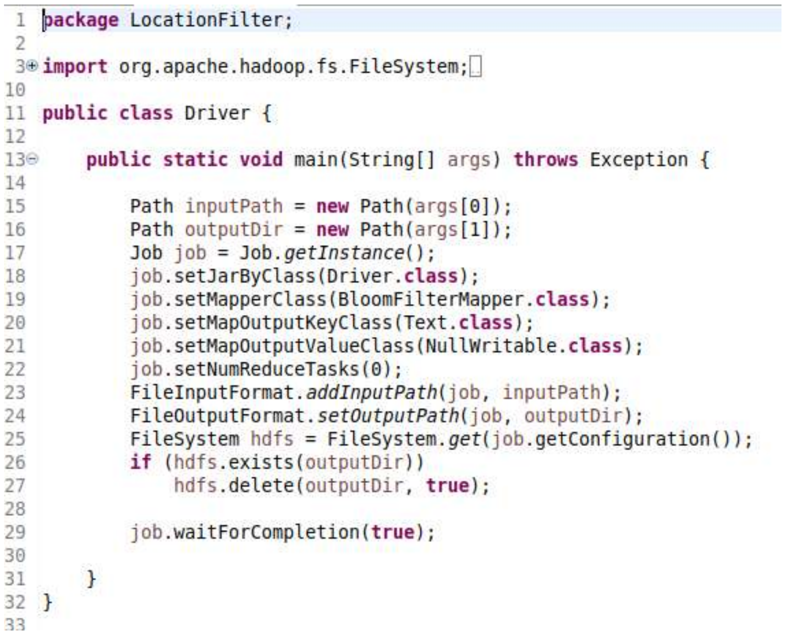
```
            for(BooleanWritable val:values){
                System.out.println(val.get());
                if (val.get() == true) {
                    sum += 1;
//                  System.out.println("float");
                }
                count = count+1;
            }

            float average = (sum/count)*100;

            result.set(average);
            context.write(key,result);
        }

    }

    public static class Map2 extends Mapper<LongWritable, Text, FloatWritable, Text>{

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

            String[] row =value.toString().split("\\t");
            Text fbi = new Text(row[0]);
            float counting = Float.valueOf(row[1].trim());

            try{
                FloatWritable count = new FloatWritable(counting);
                context.write(count, fbi);
            }

            catch(Exception e){

            }
        }
    }


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

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

        for(Text val : value){
```



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

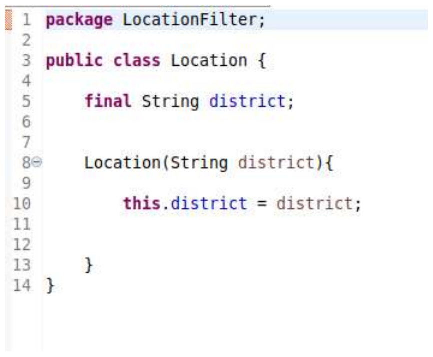
6.Bloom Filter:

a. Driver Class



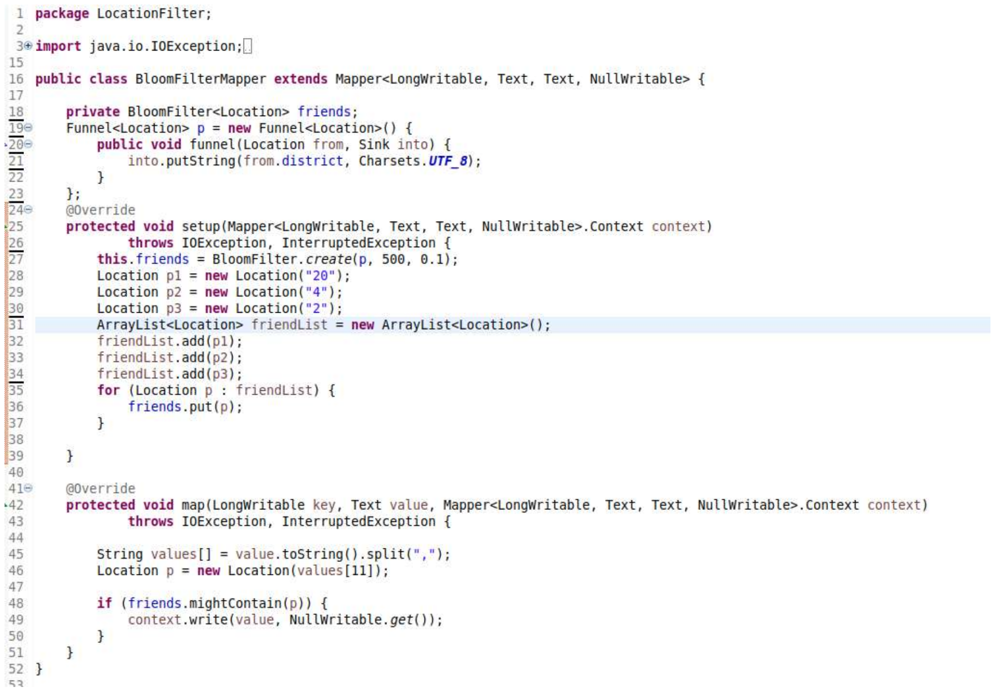
```
package LocationFilter;

import org.apache.hadoop.fs.FileSystem;

public class Driver {

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

        Path inputPath = new Path(args[0]);
        Path outputDir = new Path(args[1]);
        Job job = Job.getInstance();
        job.setJarByClass(Driver.class);
        job.setMapperClass(BloomFilterMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(NullWritable.class);
        job.setNumReduceTasks(0);
        FileInputFormat.addInputPath(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputDir);
        FileSystem hdfs = FileSystem.get(job.getConfiguration());
        if (hdfs.exists(outputDir))
            hdfs.delete(outputDir, true);

        job.waitForCompletion(true);

    }
}
```

b. Location pojo filter:



```java
package LocationFilter;

public class Location {

    final String district;


    Location(String district){

        this.district = district;


    }
}
```

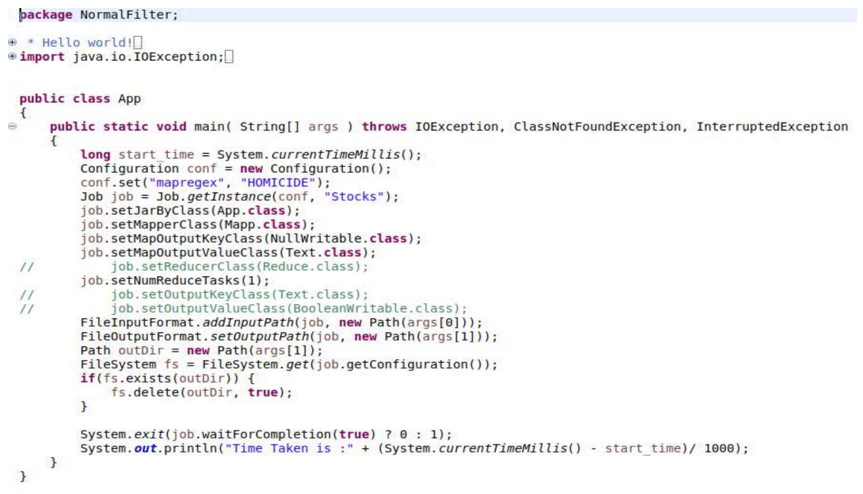
c. Bloom Filter:



```java
package LocationFilter;

import java.io.IOException;

public class BloomFilterMapper extends Mapper<LongWritable, Text, Text, NullWritable> {

    private BloomFilter<Location> friends;
    Funnel<Location> p = new Funnel<Location>() {
        public void funnel(Location from, Sink into) {
            into.putString(from.district, Charsets.UTF_8);
        }
    };
    @Override
    protected void setup(Mapper<LongWritable, Text, Text, NullWritable>.Context context)
            throws IOException, InterruptedException {
        this.friends = BloomFilter.create(p, 500, 0.1);
        Location p1 = new Location("20");
        Location p2 = new Location("4");
        Location p3 = new Location("2");
        ArrayList<Location> friendList = new ArrayList<Location>();
        friendList.add(p1);
        friendList.add(p2);
        friendList.add(p3);
        for (Location p : friendList) {
            friends.put(p);
        }

    }

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, NullWritable>.Context context)
            throws IOException, InterruptedException {

        String values[] = value.toString().split(",");
        Location p = new Location(values[11]);

        if (friends.mightContain(p)) {
            context.write(value, NullWritable.get());
        }
    }
}
```

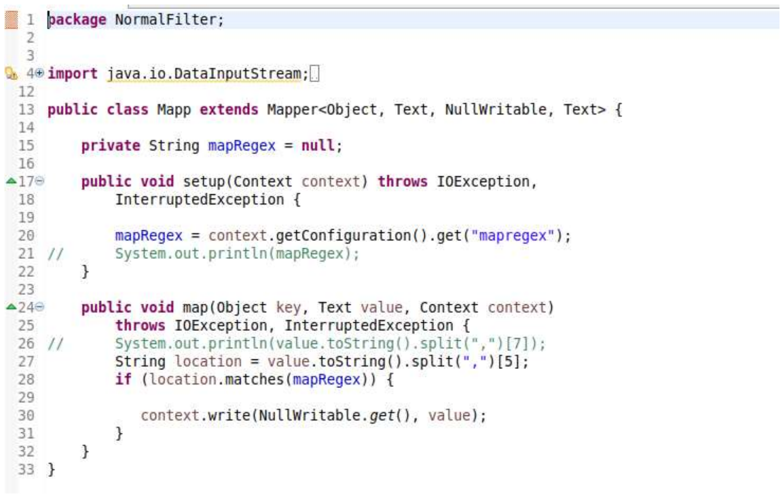
7. Regex Filter to filter crime category:

Driver Class:



```java
package NormalFilter;

 * Hello world!
import java.io.IOException;


public class App
{
    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException
    {
        long start_time = System.currentTimeMillis();
        Configuration conf = new Configuration();
        conf.set("mapregex", "HOMICIDE");
        Job job = Job.getInstance(conf, "Stocks");
        job.setJarByClass(App.class);
        job.setMapperClass(Mapp.class);
        job.setMapOutputKeyClass(NullWritable.class);
        job.setMapOutputValueClass(Text.class);
//          job.setReducerClass(Reduce.class);
        job.setNumReduceTasks(1);
//          job.setOutputKeyClass(Text.class);
//          job.setOutputValueClass(BooleanWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        Path outDir = new Path(args[1]);
        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)) {
            fs.delete(outDir, true);
        }

        System.exit(job.waitForCompletion(true) ? 0 : 1);
        System.out.println("Time Taken is :" + (System.currentTimeMillis() - start_time)/ 1000);
    }
}
```

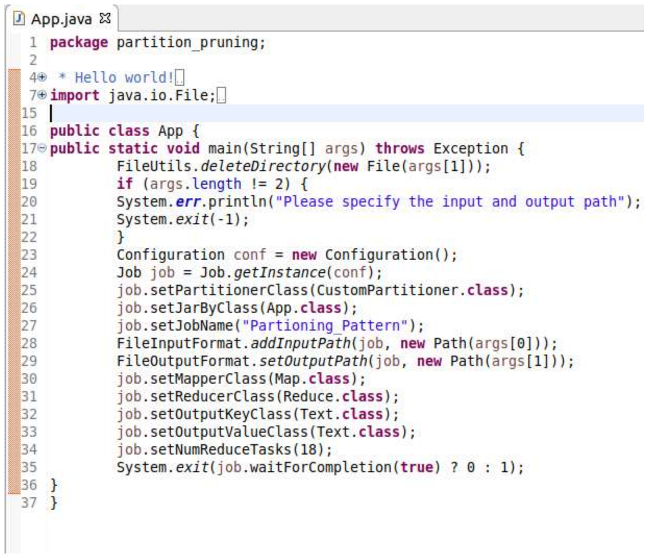
Map Class:



```
package NormalFilter;


import java.io.DataInputStream;

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

    private String mapRegex = null;

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

        mapRegex = context.getConfiguration().get("mapregex");
//      System.out.println(mapRegex);
    }

    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
//      System.out.println(value.toString().split(",")[7]);
        String location = value.toString().split(",")[5];
        if (location.matches(mapRegex)) {

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

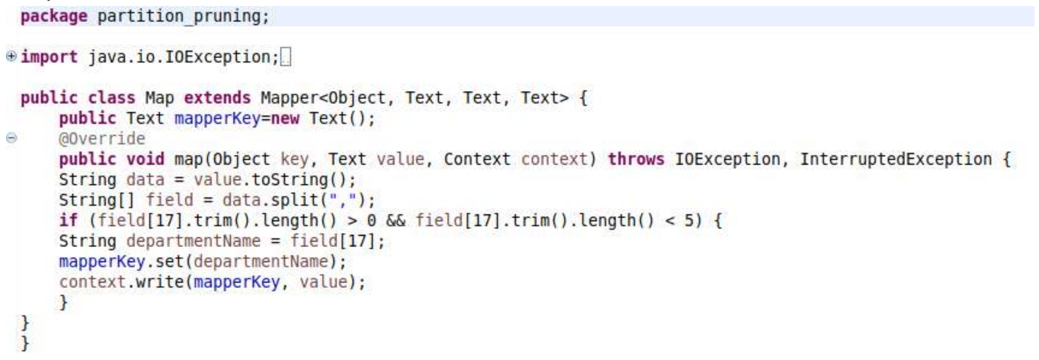
8.Partition Pruning to partition data based on years:

Driver Class:



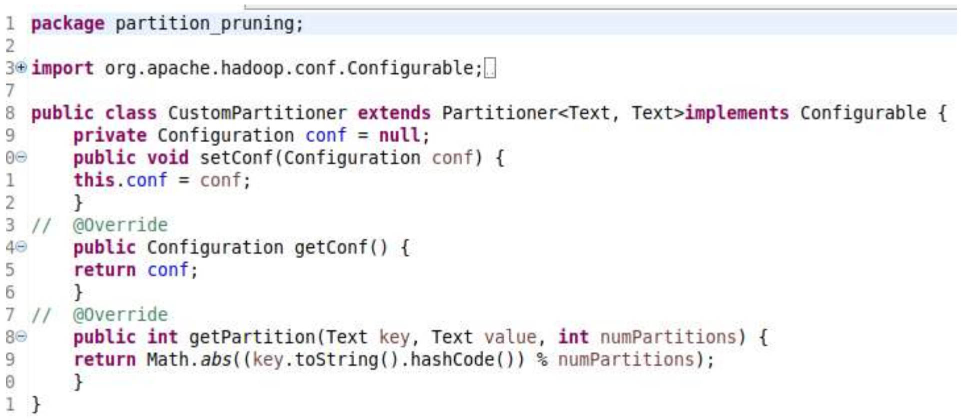
```
App.java

package partition_pruning;

 * Hello world!
import java.io.File;

public class App {
public static void main(String[] args) throws Exception {
        FileUtils.deleteDirectory(new File(args[1]));
        if (args.length != 2) {
        System.err.println("Please specify the input and output path");
        System.exit(-1);
        }
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setPartitionerClass(CustomPartitioner.class);
        job.setJarByClass(App.class);
        job.setJobName("Partioning_Pattern");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setNumReduceTasks(18);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Map Class:



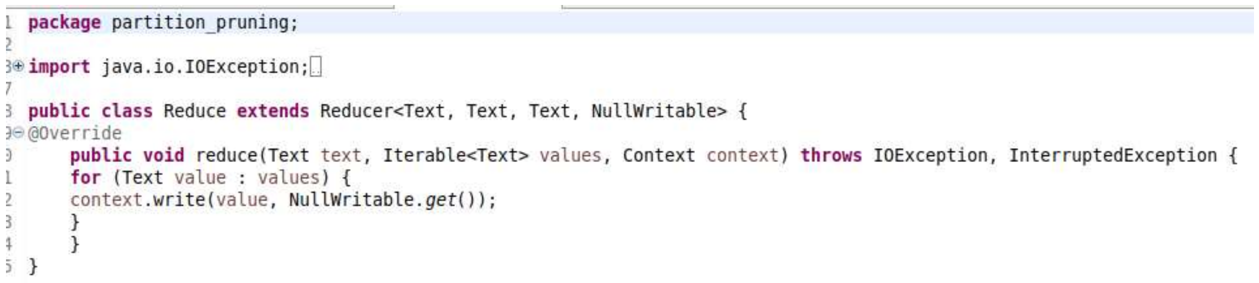
```
package partition_pruning;

import java.io.IOException;

public class Map extends Mapper<Object, Text, Text, Text> {
     public Text mapperKey=new Text();
     @Override
     public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
     String data = value.toString();
     String[] field = data.split(",");
     if (field[17].trim().length() > 0 && field[17].trim().length() < 5) {
     String departmentName = field[17];
     mapperKey.set(departmentName);
     context.write(mapperKey, value);
     }
}
}
```

Custom Partitioner Class:



```
package partition_pruning;

import org.apache.hadoop.conf.Configurable;

public class CustomPartitioner extends Partitioner<Text, Text>implements Configurable {
    private Configuration conf = null;
    public void setConf(Configuration conf) {
    this.conf = conf;
    }
//  @Override
    public Configuration getConf() {
    return conf;
    }
//  @Override
    public int getPartition(Text key, Text value, int numPartitions) {
    return Math.abs((key.toString().hashCode()) % numPartitions);
    }
}
```

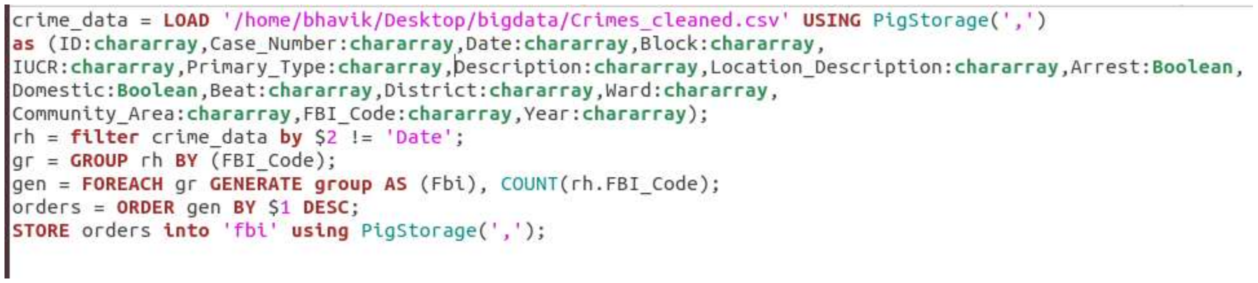
Reducer Class:



```
package partition_pruning;

import java.io.IOException;

public class Reduce extends Reducer<Text, Text, Text, NullWritable> {
@Override
    public void reduce(Text text, Iterable<Text> values, Context context) throws IOException, InterruptedException {
    for (Text value : values) {
    context.write(value, NullWritable.get());
    }
    }
}
```

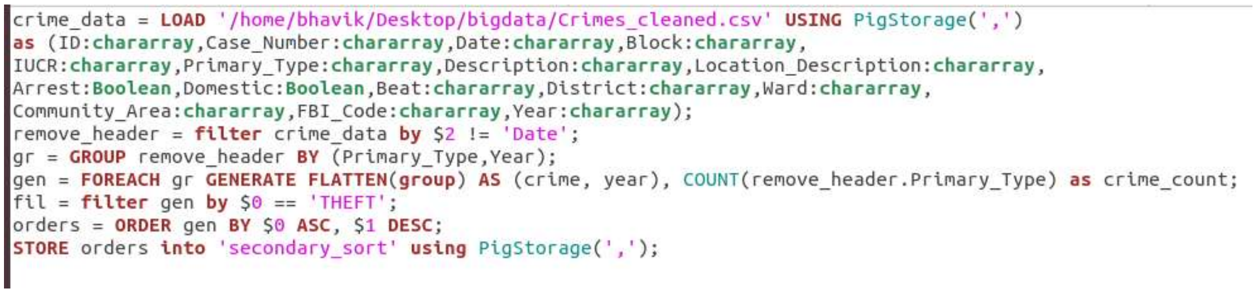
Analysis through Pig Script:

1. Count of cases with FBI:



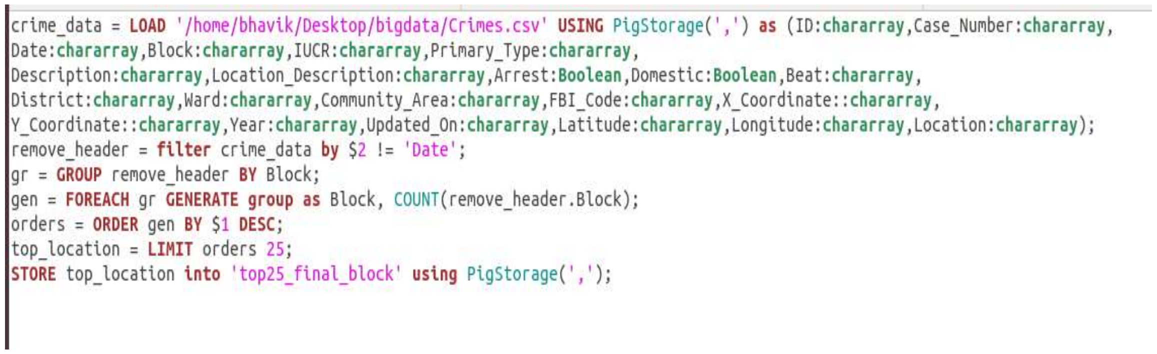
```
crime_data = LOAD '/home/bhavik/Desktop/bigdata/Crimes_cleaned.csv' USING PigStorage(',')
as (ID:chararray,Case_Number:chararray,Date:chararray,Block:chararray,
IUCR:chararray,Primary_Type:chararray,Description:chararray,Location_Description:chararray,Arrest:Boolean,
Domestic:Boolean,Beat:chararray,District:chararray,Ward:chararray,
Community_Area:chararray,FBI_Code:chararray,Year:chararray);
rh = filter crime_data by $2 != 'Date';
gr = GROUP rh BY (FBI_Code);
gen = FOREACH gr GENERATE group AS (Fbi), COUNT(rh.FBI_Code);
orders = ORDER gen BY $1 DESC;
STORE orders into 'fbi' using PigStorage(',');
```

2. Secondary sorting separating data based on year and categories of crime:



```
crime_data = LOAD '/home/bhavik/Desktop/bigdata/Crimes_cleaned.csv' USING PigStorage(',')
as (ID:chararray,Case_Number:chararray,Date:chararray,Block:chararray,
IUCR:chararray,Primary_Type:chararray,Description:chararray,Location_Description:chararray,
Arrest:Boolean,Domestic:Boolean,Beat:chararray,District:chararray,Ward:chararray,
Community_Area:chararray,FBI_Code:chararray,Year:chararray);
remove_header = filter crime_data by $2 != 'Date';
gr = GROUP remove_header BY (Primary_Type,Year);
gen = FOREACH gr GENERATE FLATTEN(group) AS (crime, year), COUNT(remove_header.Primary_Type) as crime_count;
fil = filter gen by $0 == 'THEFT';
orders = ORDER gen BY $0 ASC, $1 DESC;
STORE orders into 'secondary_sort' using PigStorage(',');
```

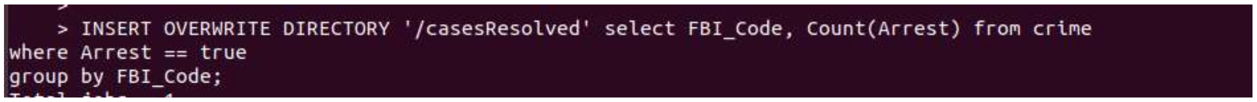
3. Top 25 blocks where crimes are reported:



```
crime_data = LOAD '/home/bhavik/Desktop/bigdata/Crimes.csv' USING PigStorage(',') as (ID:chararray,Case_Number:chararray,
Date:chararray,Block:chararray,IUCR:chararray,Primary_Type:chararray,
Description:chararray,Location_Description:chararray,Arrest:Boolean,Domestic:Boolean,Beat:chararray,
District:chararray,Ward:chararray,Community_Area:chararray,FBI_Code:chararray,X_Coordinate::chararray,
Y_Coordinate::chararray,Year:chararray,Updated_On:chararray,Latitude:chararray,Longitude:chararray,Location:chararray);
remove_header = filter crime_data by $2 != 'Date';
gr = GROUP remove_header BY Block;
gen = FOREACH gr GENERATE group as Block, COUNT(remove_header.Block);
orders = ORDER gen BY $1 DESC;
top_location = LIMIT orders 25;
STORE top_location into 'top25_final_block' using PigStorage(',');
```

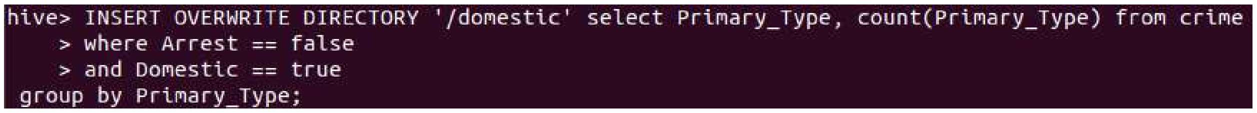
Analysis through Hive

1. Percentage of crime solved by FBI:



```
    > INSERT OVERWRITE DIRECTORY '/casesResolved' select FBI_Code, Count(Arrest) from crime
where Arrest == true
group by FBI_Code;
```

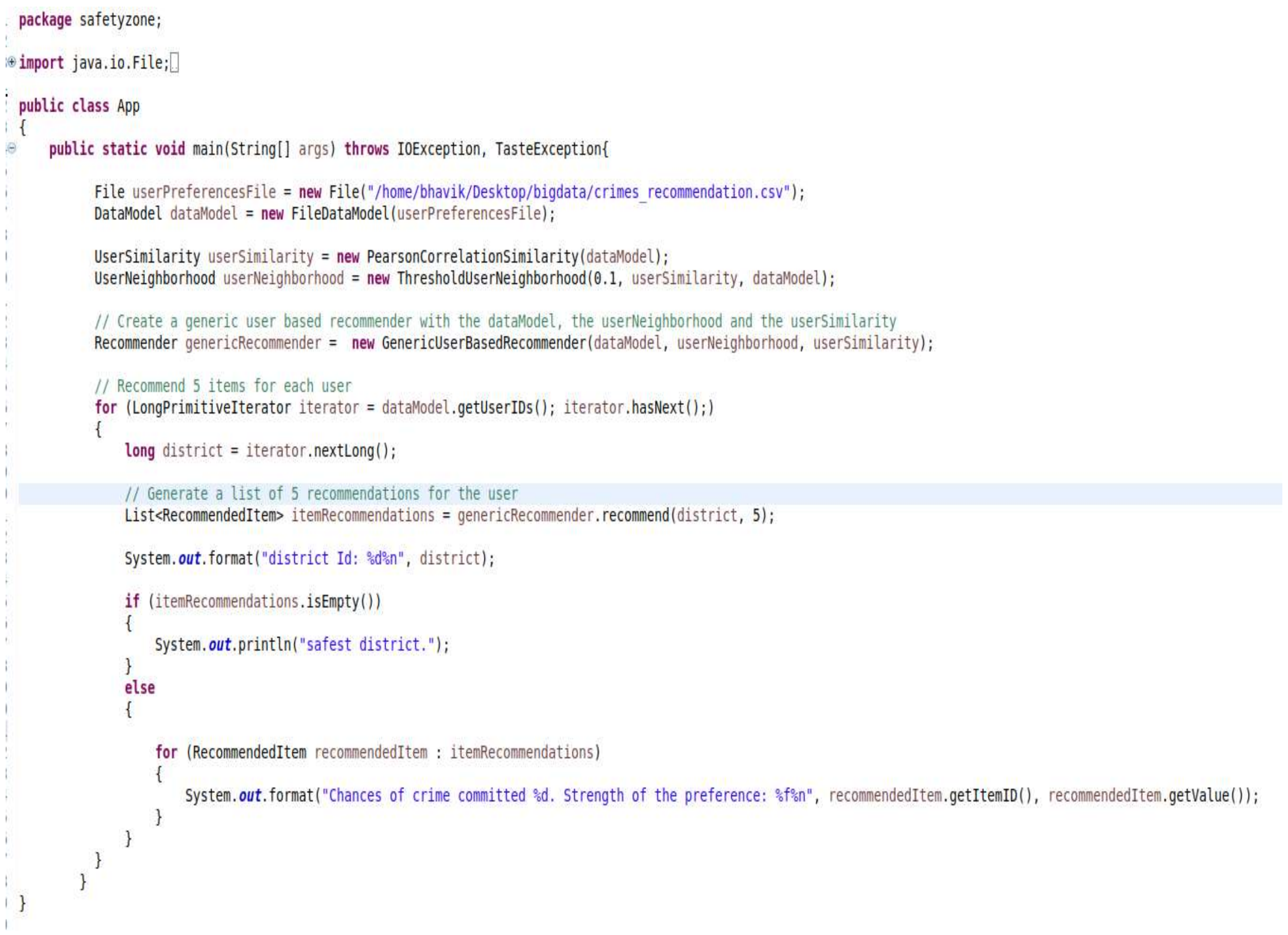
2. Domestic cases which are not resolved:



```
hive> INSERT OVERWRITE DIRECTORY '/domestic' select Primary_Type, count(Primary_Type) from crime
    > where Arrest == false
    > and Domestic == true
 group by Primary_Type;
```

Mahout Recommendation Algorithm

Algorithm to predict crime that can happen in different districts:



```java
package safetyzone;

import java.io.File;

public class App
{
    public static void main(String[] args) throws IOException, TasteException{

        File userPreferencesFile = new File("/home/bhavik/Desktop/bigdata/crimes_recommendation.csv");
        DataModel dataModel = new FileDataModel(userPreferencesFile);

        UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(dataModel);
        UserNeighborhood userNeighborhood = new ThresholdUserNeighborhood(0.1, userSimilarity, dataModel);

        // Create a generic user based recommender with the dataModel, the userNeighborhood and the userSimilarity
        Recommender genericRecommender =  new GenericUserBasedRecommender(dataModel, userNeighborhood, userSimilarity);

        // Recommend 5 items for each user
        for (LongPrimitiveIterator iterator = dataModel.getUserIDs(); iterator.hasNext();)
        {
            long district = iterator.nextLong();

            // Generate a list of 5 recommendations for the user
            List<RecommendedItem> itemRecommendations = genericRecommender.recommend(district, 5);

            System.out.format("district Id: %d%n", district);

            if (itemRecommendations.isEmpty())
            {
                System.out.println("safest district.");
            }
            else
            {

                for (RecommendedItem recommendedItem : itemRecommendations)
                {
                    System.out.format("Chances of crime committed %d. Strength of the preference: %f%n", recommendedItem.getItemID(), recommendedItem.getValue());
                }
            }
        }
    }
}
```