Managing Information Technology and Systems

Big Data Project

MCDA5570

Submitted By:

Aman Sharma (A00429273)
Maninder Kaur (A00429943)
Simon Al Achkar (A00424630)

Course Administrator:

Nikita Neveditsin

AGE QUOD AGIS
SAINT MARY'S
UNIVERSITY SINCE 1802

One University. One World. Yours.

Contents

Project Objective 2

Description of Data 2

Understanding the Data 3

Setting up The Data 4

ETL Using Pig 4

Insights Gathered/Results 5

Objective 1: For Each Year And District, The % Of Homicide That Lead To An Arrest. 5

Objective 2: Display The Number Of Thefts By Month, All Years Combined. 7

Objective 3: See The Trend Of All Kinds Of Crime Through The Years 8

Objective 4: See The Trend Of Specific Kind Of Crimes Though The Years 9

Objective 5: Top 3 Day Of The Week Most Crime Are Committed For Each Year 10

Figure 1: Final Zepellin Notebook 12

Comments 13

References 14

Project Objective

In this Project, we will use Apache Zeppelin. With Zeppelin, we will do a number of data analysis by answering some questions on the crime dataset using Hive, Spark and Pig. We will prepare some chart to better represent our results and finally share them.

On completing this big data project using zeppelin, we will have know what Zeppelin is, gained the ability to install new interpreters, use Zeppelin for performing data analysis, sharing results.

Description of Data

Data Set Name: Chicago's Crimes - 2001 to present

Data Set Link: https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD

Description of Data:

This dataset reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to present, minus the most recent seven days.

Data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. In order to protect the privacy of crime victims, addresses are shown at the block level only and specific locations are not identified. The dataset contains more than 65,000 records/rows of data.

Understanding the Data

S. No.	Field Name	Field Description
1.	ID	Unique identifier for the record.
2.	Case Number	The Chicago Police Department RD Number (Records Division Number), which is unique to the incident.
3.	Date	Date when the incident occurred. this is sometimes a best estimate.
4.	Block	The partially redacted address where the incident occurred, placing it on the same block as the actual address.
5.	IUCR	The Illinois Uniform Crime Reporting code. This is directly linked to the Primary Type and Description.
6.	Primary Type	The primary description of the IUCR code.
7.	Description	The secondary description of the IUCR code, a subcategory of the primary description.
8.	Location Description	Description of the location where the incident occurred.
9.	Arrest	Indicates whether an arrest was made.
10.	Domestic	Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act.
11.	Beat	Indicates the beat where the incident occurred. A beat is the smallest police geographic area – each beat has a dedicated police beat car. Three to five beats make up a police sector, and three sectors make up a police district. The Chicago Police Department has 22 police districts
12.	District	Indicates the police district where the incident occurred.
13.	Ward	The ward (City Council district) where the incident occurred.
14.	Community Area	Indicates the community area where the incident occurred. Chicago has 77 community areas.
15.	FBI Code	Indicates the crime classification as outlined in the FBI's National Incident-Based Reporting System (NIBRS).
16.	X Coordinate	The x coordinate of the location where the incident occurred in State Plane Illinois East NAD 1983 projection. This location is shifted from the actual location for partial redaction but falls on the same block.
17.	Y Coordinate	The y coordinate of the location where the incident occurred in State Plane Illinois East NAD 1983 projection. This location is shifted from the actual location for partial redaction but falls on the same block.
18.	Year	Year the incident occurred.
19.	Updated On	Date and time the record was last updated.
20.	Latitude	The latitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.
21.	Longitude	The longitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.
22.	Location	The location where the incident occurred in a format that allows for creation of maps and other geographic operations on this data portal. This location is shifted from the actual location for partial redaction but falls on the same block.

Setting up The Data

Make Directory:

```
hdfs dfs -mkdir -p /user/project/rawdata/crime/Chicago
```

Downloading File:

```
wget -O crime_chicago.csv https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD
```

Transferring File from local

```
hdfs dfs -put crime_chicago.csv /user/project/rawdata/crime/Chicago
```

ETL Using Pig

- **Register The Third Party Piggybank Jar (To read CSV File)**

```
register hdfs:///user/project/rawdata/crime/piggybank-0.16.0.jar;
```

- **Read The File Content From hdfs**

```
raw_data_chicago = LOAD '/user/project/rawdata/crime/chicago' USING org.apache.pig.piggybank.storage.CSVLoader() AS (id: chararray, case_no: chararray, date: chararray, block: chararray, iucr: chararray, primary_type: chararray, description: chararray, location_description: chararray, arrest: chararray, domestic: chararray, beat: chararray, district: chararray, ward: chararray, community_area: chararray, fbi_code: chararray, coordinate_x: chararray, coordinat_y: chararray, year: chararray, updated_on: chararray, latitude: chararray, longitude: chararray, location: chararray);
```

- **Remove The Header**

```
raw_data_headless_chicago = FILTER raw_data_chicago BY id != 'ID';
```

- **Select The Required Columns**

```
data_chicago = FOREACH raw_data_headless_chicago GENERATE date, block, primary_type, description, location_description, arrest, domestic, district, year;
```

Insights Gathered/Results

Objective 1: For Each Year And District, The % Of Homicide That Lead To An Arrest.

%pig.query

```
homicide_group = FILTER data_chicago BY primary_type == 'HOMICIDE';
```

```
homicide_grouping = GROUP homicide_group BY (year, district);
```

```
homicide_year_dist_count = FOREACH homicide_grouping GENERATE group.year, group.district, CONCAT(group.year, group.district) AS key, COUNT(homicide_group) as total_count;
```

Filter Out Homicide That Lead To Arrest

```
filtered_data = FILTER homicide_group BY arrest == 'true';
```

Group Records By Year And District Again

```
filtered_grouping = GROUP filtered_data BY (year, district);
```

Project The Aggregate Here

```
filtered_year_dist_count = FOREACH filtered_grouping GENERATE group.year, group.district, CONCAT(group.year, group.district) AS key, COUNT(filtered_data) as filt_count;
```

Join The Two Records Together And Calculate For Percentage

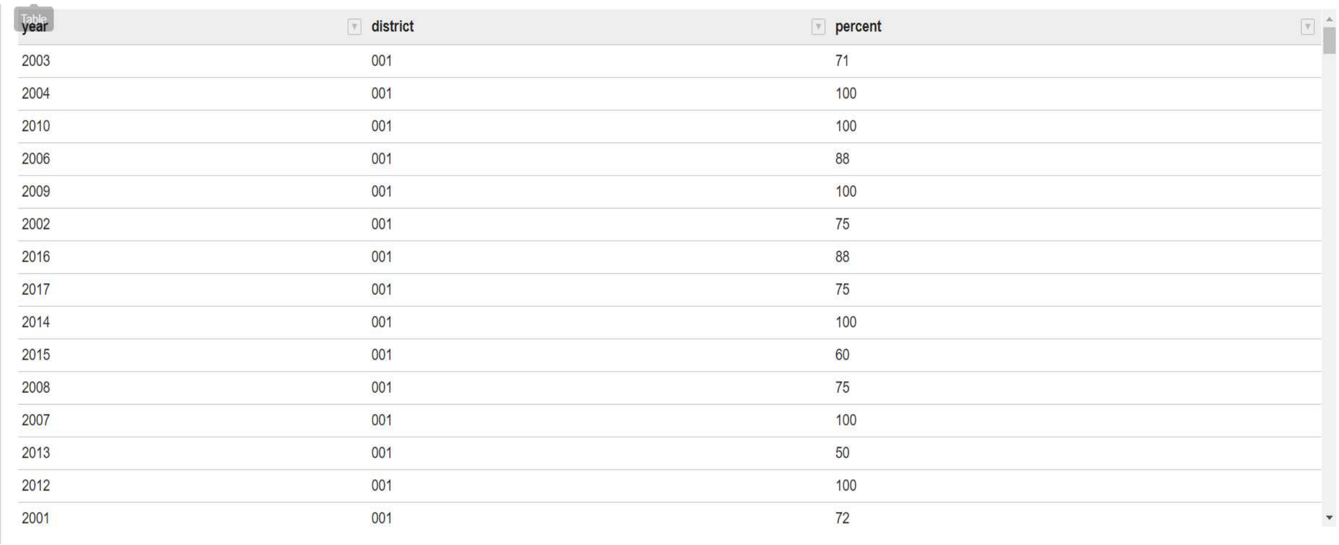
```
join_records = JOIN homicide_year_dist_count BY key, filtered_year_dist_count BY key;
```

Project Result

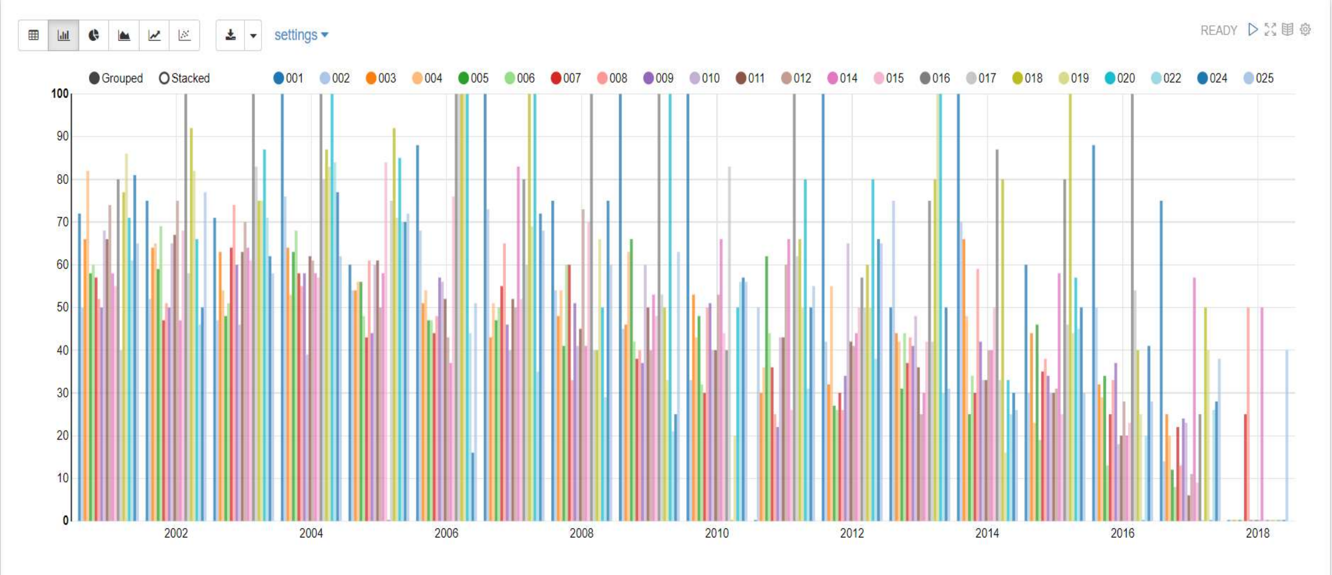
```
aggregation = FOREACH join_records GENERATE  homicide_year_dist_count::year as year, homicide_year_dist_count::district as district,  (int)(((double)filtered_year_dist_count::filt_count / (double)homicide_year_dist_count::total_count) * 100) AS percent;
```

```
foreach aggregation generate year, district, percent;
```

Output:



year	district	percent
2003	001	71
2004	001	100
2010	001	100
2006	001	88
2009	001	100
2002	001	75
2016	001	88
2017	001	75
2014	001	100
2015	001	60
2008	001	75
2007	001	100
2013	001	50
2012	001	100
2001	001	72



Objective 2: Display The Number Of Thefts By Month, All Years Combined.

%pig.query

```
filtered_data = FILTER data_chicago BY  (INDEXOF(UPPER(primary_type), 'THEFT', 0) > -1)
AND (arrest == 'true');
```

Select Month And Primary Type

```
projected_data = FOREACH filtered_data GENERATE GetMonth(ToDate(date, 'MM/dd/yyyy
hh:mm:ss a')) AS month, primary_type;
```

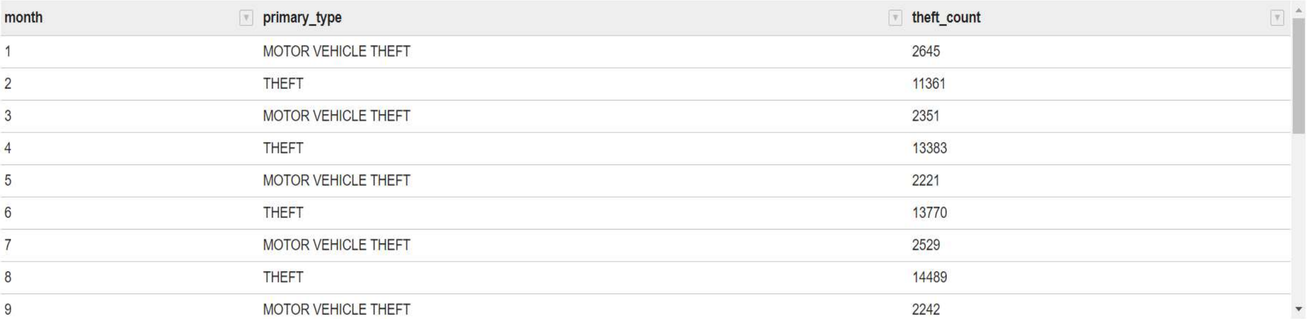
Group By Month And Primary Type

```
grouped_data = GROUP projected_data BY (month, primary_type);
```

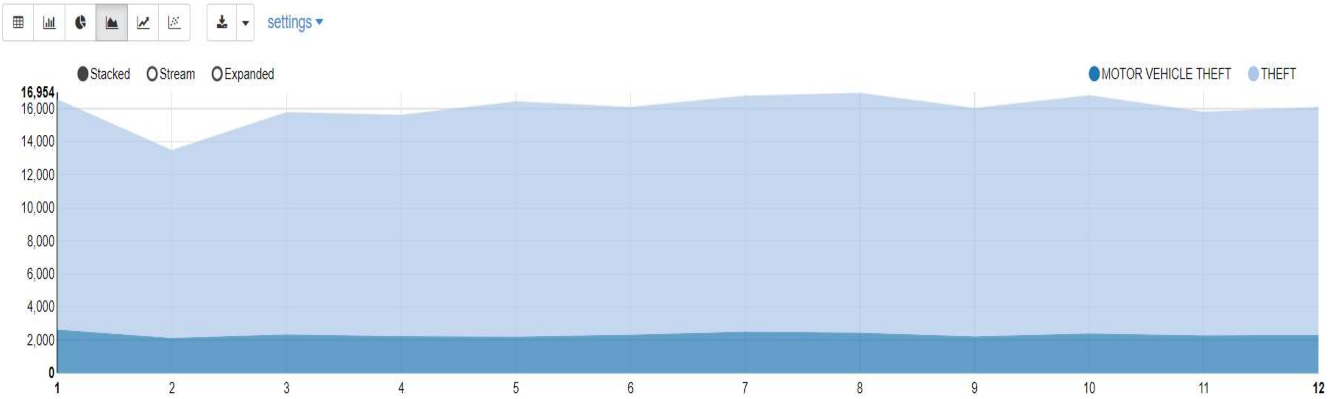
Aggregation

```
FOREACH grouped_data GENERATE group.month AS month, group. primary_type as
primary_type, COUNT(projected_data) AS theft_count;
```

Output:



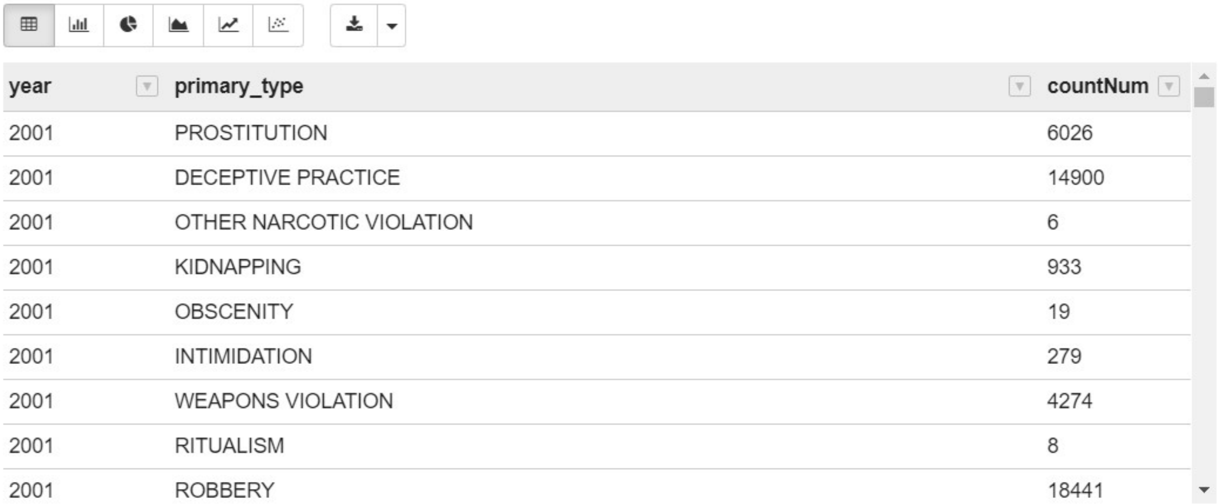
month	primary_type	theft_count
1	MOTOR VEHICLE THEFT	2645
2	THEFT	11361
3	MOTOR VEHICLE THEFT	2351
4	THEFT	13383
5	MOTOR VEHICLE THEFT	2221
6	THEFT	13770
7	MOTOR VEHICLE THEFT	2529
8	THEFT	14489
9	MOTOR VEHICLE THEFT	2242



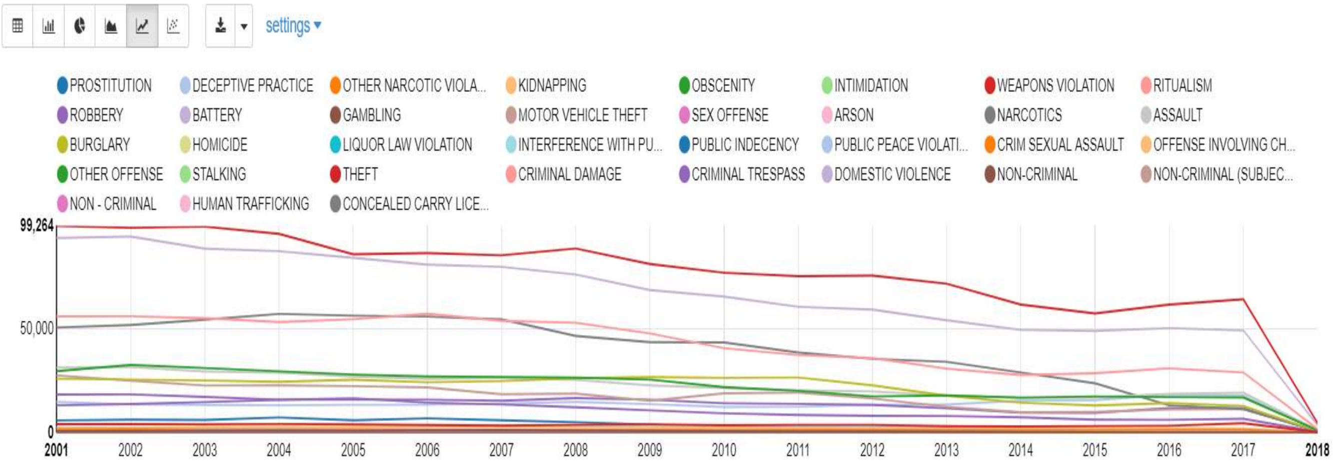
Objective 3: See The Trend Of All Kinds Of Crime Through The Years

```
%hive

select year, primary_type, count(1) as countNum from chicago_crime_raw where year =
'${year=2001,2001|2002|2003|2004|2005|2006|2007|2008|2009|2010|2011|2012|2013|2014|2015|
2016|2017|2018}' and primary_type IN
('${checkbox:Crime=PROSTITUTION|BATTERY|THEFT|ROBBERY|ASSAULT}')

and group by year, primary_type order by year;
```

Output:



year	primary_type	countNum
2001	PROSTITUTION	6026
2001	DECEPTIVE PRACTICE	14900
2001	OTHER NARCOTIC VIOLATION	6
2001	KIDNAPPING	933
2001	OBSCENITY	19
2001	INTIMIDATION	279
2001	WEAPONS VIOLATION	4274
2001	RITUALISM	8
2001	ROBBERY	18441

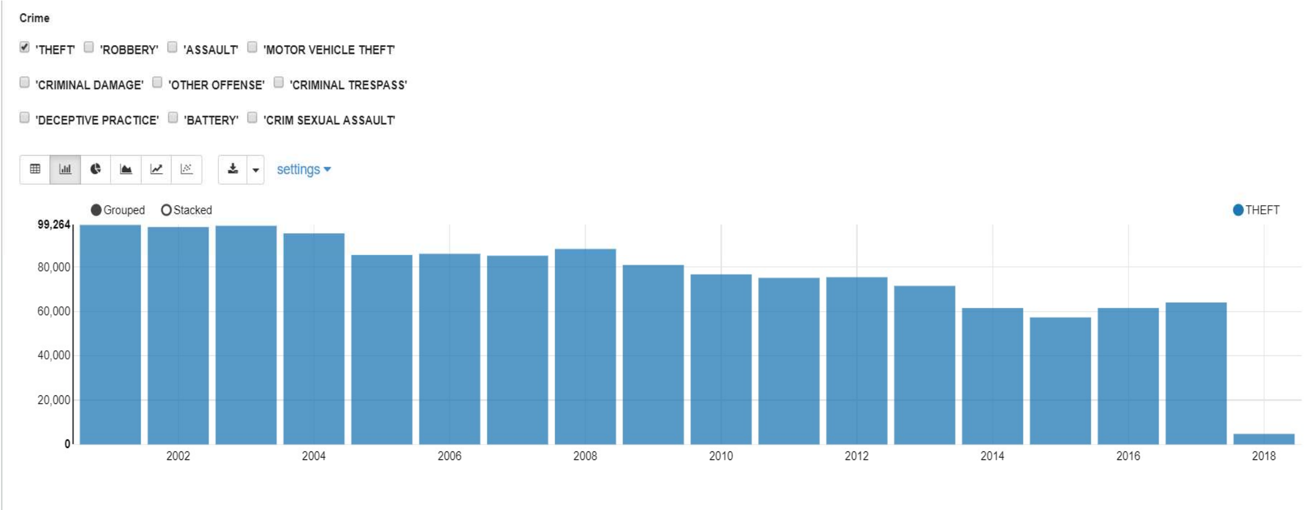


Objective 4: See The Trend Of Specific Kind Of Crimes Though The Years

%hive

```
select year, primary_type, count(1) as knt from chicago_crime_raw where primary_type IN (${checkbox:Crime='THEFT'|'ROBBERY'|'ASSAULT','THEFT'|'ROBBERY'|'ASSAULT'|'MOTOR VEHICLE THEFT'|'CRIMINAL DAMAGE'|'OTHER OFFENSE'|'CRIMINAL TRESPASS'|'DECEPTIVE PRACTICE'|'BATTERY'|'CRIM SEXUAL ASSAULT'}) group by year, primary_type;
```

Output:



Crime

☑ 'THEFT' ☐ 'ROBBERY' ☐ 'ASSAULT' ☐ 'MOTOR VEHICLE THEFT'

☐ 'CRIMINAL DAMAGE' ☐ 'OTHER OFFENSE' ☐ 'CRIMINAL TRESPASS'

☐ 'DECEPTIVE PRACTICE' ☐ 'BATTERY' ☐ 'CRIM SEXUAL ASSAULT'

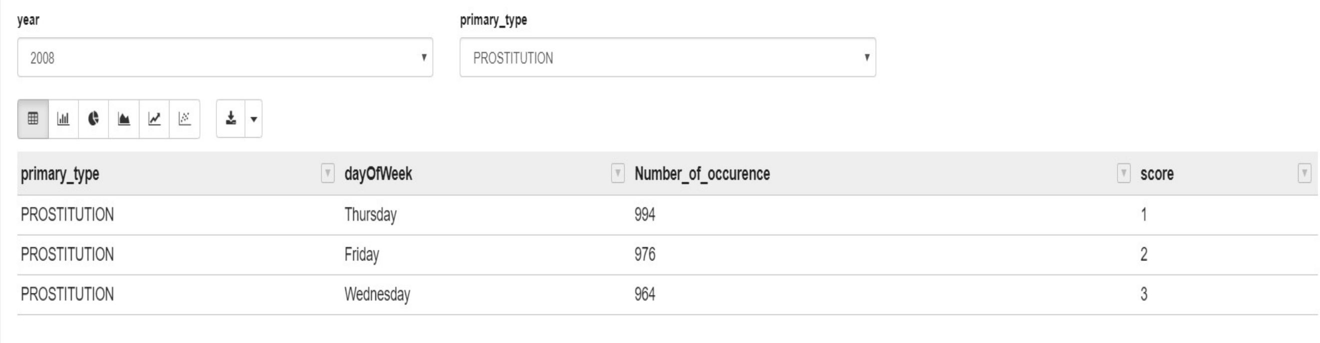
Objective 5: Top 3 Day Of The Week Most Crime Are Committed For Each Year

```
%spark2

import java.text.SimpleDateFormat
import org.apache.spark.sql.functions._

val dayofweek = udf((dt: String) => {
   if (dt == null) null
   else {
      val sdf1 = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss a")
      val date = sdf1.parse(dt)
      val sdf2 = new SimpleDateFormat("EEEEEE")
      sdf2.format(date)
   }
})

val tableDF = sqlContext.sql("select crime_date, year, primary_type from chicago_crime_raw
where year != 'Year'").
   withColumn("dayOfWeek", dayofweek(col("crime_date"))).
   select("primary_type", "dayOfWeek", "year").cache

tableDF.registerTempTable("crime")
```

%spark2.sql

```
SELECT *
FROM
  (SELECT primary_type,
          dayOfWeek,
          cnt as Number_of_occurence,
          rank() over (partition BY primary_type
                      ORDER BY cnt DESC) score
   FROM
     (SELECT primary_type,
             dayOfWeek,
             count(1) cnt
      FROM crime
      WHERE YEAR =
${year=2001,2001|2002|2003|2004|2005|2006|2007|2008|2009|2010|2011|2012|2013|2014|2015|
2016|2017|2018}
      AND primary_type = '${primary_type=THEFT,THEFT|ROBBERY|ASSAULT|MOTOR
VEHICLE THEFT|CRIMINAL DAMAGE|OTHER OFFENSE|CRIMINAL
TRESPASS|DECEPTIVE PRACTICE|BATTERY|CRIM SEXUAL
ASSAULT|PROSTITUTION}'
      GROUP BY primary_type,
               dayOfWeek) AS v) AS w
WHERE score <= 3
```

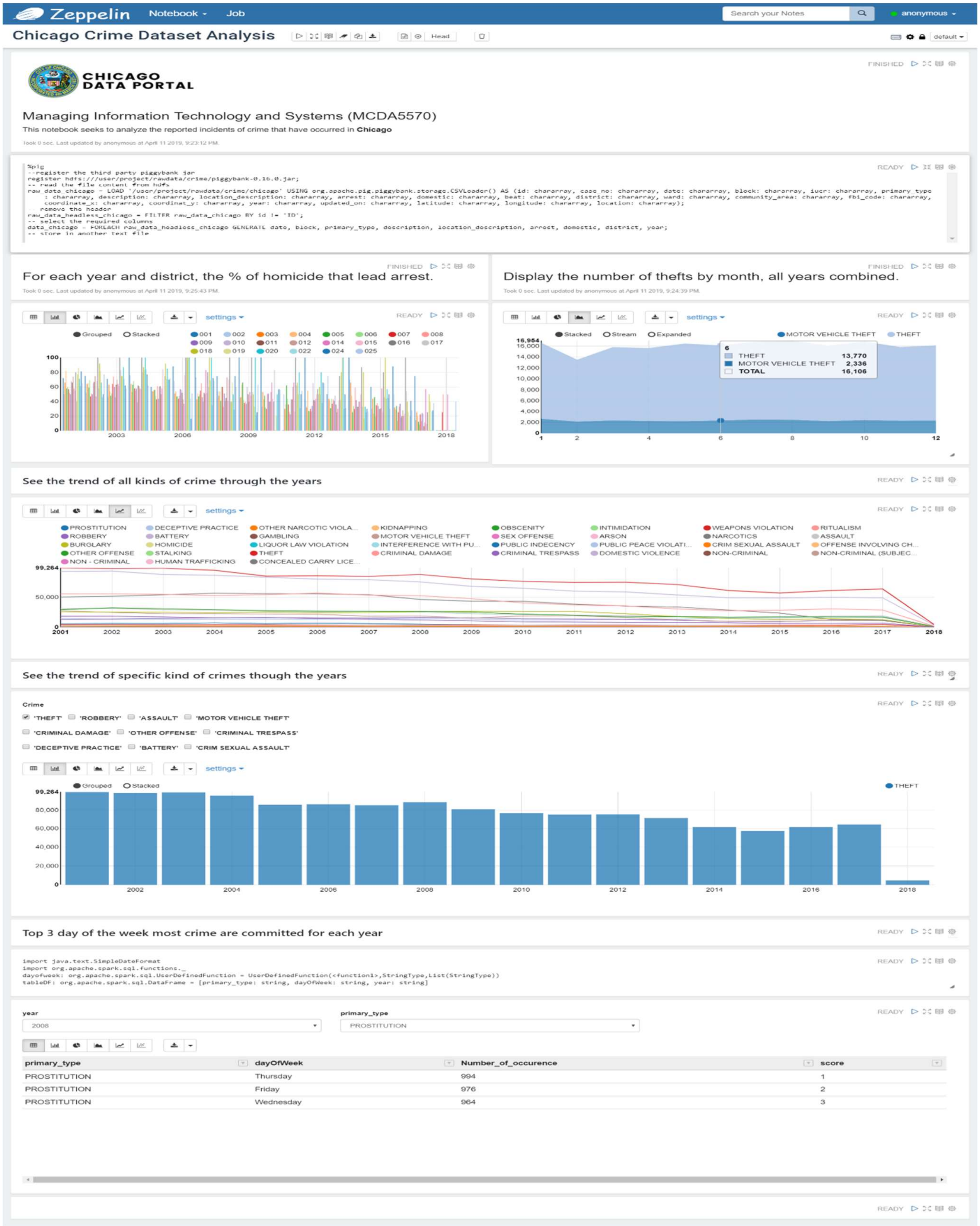
Output



year: 2008

primary_type: PROSTITUTION

primary_type	dayOfWeek	Number_of_occurence	score
PROSTITUTION	Thursday	994	1
PROSTITUTION	Friday	976	2
PROSTITUTION	Wednesday	964	3



Chicago Crime Dataset Analysis

Managing Information Technology and Systems (MCDA5570)

This notebook seeks to analyze the reported incidents of crime that have occurred in **Chicago**

Took 0 sec. Last updated by anonymous at April 11 2019, 9:23:12 PM.

```
%pig
--register the third party piggybank jar
register hdfs:///user/project/rawdata/crime/piggybank-0.16.0.jar;
-- read the file content from hdfs
raw_data_chicago = LOAD '/user/project/rawdata/crime/chicago' USING org.apache.pig.piggybank.storage.CSVLoader() AS (id: chararray, case_no: chararray, date: chararray, block: chararray, iucr: chararray, primary_type
    : chararray, description: chararray, location_description: chararray, arrest: chararray, domestic: chararray, beat: chararray, district: chararray, ward: chararray, community_area: chararray, fbi_code: chararray,
    coordinate_x: chararray, coordinat_y: chararray, year: chararray, updated_on: chararray, latitude: chararray, longitude: chararray, location: chararray);
-- remove the header
raw_data_headless_chicago = FILTER raw_data_chicago BY id != 'ID';
-- select the required columns
data_chicago = FOREACH raw_data_headless_chicago GENERATE date, block, primary_type, description, location_description, arrest, domestic, district, year;
-- store in another text file
```

For each year and district, the % of homicide that lead arrest.

Took 0 sec. Last updated by anonymous at April 11 2019, 9:25:43 PM.

Display the number of thefts by month, all years combined.

Took 0 sec. Last updated by anonymous at April 11 2019, 9:24:39 PM.

See the trend of all kinds of crime through the years

See the trend of specific kind of crimes though the years

Crime

☑ 'THEFT' ☐ 'ROBBERY' ☐ 'ASSAULT' ☐ 'MOTOR VEHICLE THEFT'

☐ 'CRIMINAL DAMAGE' ☐ 'OTHER OFFENSE' ☐ 'CRIMINAL TRESPASS'

☐ 'DECEPTIVE PRACTICE' ☐ 'BATTERY' ☐ 'CRIM SEXUAL ASSAULT'

Top 3 day of the week most crime are committed for each year

```
import java.text.SimpleDateFormat
import org.apache.spark.sql.functions._
dayofweek: org.apache.spark.sql.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,List(StringType))
tableDF: org.apache.spark.sql.DataFrame = [primary_type: string, dayOfWeek: string, year: string]
```

year: 2008

primary_type: PROSTITUTION

primary_type	dayOfWeek	Number_of_occurence	score
PROSTITUTION	Thursday	994	1
PROSTITUTION	Friday	976	2
PROSTITUTION	Wednesday	964	3

Figure 1: Final Zepellin Notebook

Comments

What was difficult to understand?

In general, we had to work with lower level API's in Hadoop Ecosystem.

Simple data transformation and ingestion tasks required significant effort from time to time.

Lastly, integrating multiple technologies was a challenge but instructive task.

What was too easy to understand?

Understanding working with the tools designed to provide conventional work flows was relatively easy.

For example, Hive uses similar syntax to conventional SQL structure.

Zeppelin uses the modern notebook style development interface, which is versatile and easy to follow.

Accompanying, Hadoop Ecosystem and different tools with the Linux classes made it more comfortable working with several different tools using Terminal interface.

Maybe something was not covered but you'd like it to be covered?

Managing Information Tech & Systems is a hybrid course bringing management and IT aspects together. The only thing I could come up as a shortcoming was that the understanding of live data analysis which in very advantageous in today's date science generation.

Other Comments:

I seriously recommend Nikita to teach this course to upcoming batches as well as he taught the whole concept clearly and in detail and provided challenging yet simple examples & in class exercise covering every topic individually.

References

- The Apache Software Foundation (2019) *Apache Zeppelin 0.8.0 Documentation: Interpreter in Apache Zeppelin,* Available at: http://zeppelin.apache.org/docs/0.8.0/usage/interpreter/overview.html.

- The Apache Software Foundation (2019) *Pig Interpreter for Apache Zeppelin,* Available at: http://zeppelin.apache.org/docs/0.8.0/interpreter/pig.html

- The Apache Software Foundation (2019) *Hive Interpreter for Apache Zeppelin,* Available at: http://zeppelin.apache.org/docs/0.8.0/interpreter/hive.html

- The Apache Software Foundation (2019) *Markdown Interpreter for Apache Zeppelin,* Available at: http://zeppelin.apache.org/docs/0.8.0/interpreter/markdown.html

- The Apache Software Foundation (2019) *Zeppelin Tutorial,* Available at: https://zeppelin.apache.org/docs/0.5.5-incubating/tutorial/tutorial.html

- The Apache Software Foundation (2019) *Spark Interpreter,* Available at: https://zeppelin.apache.org/docs/0.5.5-incubating/interpreter/spark.html