

Exploratory Data Analysis of Crimes in Chicago

Final Project Group 7

Tianlu Chen	512471	Kaydee Wang	512566
Hanye Wang	510752	Zachary Li	511614
Chenwei Lin	512551		

Executive Summary

The above-national-average crime rates in the City of Chicago have drawn our attention. Therefore, an exploratory data analysis of crime incidents reported from 2008-2016 was conducted on the dataset extracted from the Chicago Police Department to investigate trends and traits of crimes in terms of time, location, and crime type. After data preprocessing utilizing Linux and pySpark, other tools including Hive, Mapreduce, and Tableau were also used in further analysis.

The results reveal a downward trend in the overall crime incidents during the given period, with seasonal ups in summer and daily peaks at noon and 7 pm. As for location and crime type, relatively fewer incidences happened downtown and on the outer edges of Chicago. Several crime types such as criminal damage and weapons violation had similar distributions and were more common in central areas. While the theft, which had top cumulative frequency, was more concentrated on the east coast. And more than half of the top 15 crime types had an offender arrested rate lower than 25%. Corresponding with the whole picture, almost all communities and the top 15 crime types had a decreasing number of crime incidents over this period. District 25 and the streets were the most dangerous place and scenes to go to.

1. Description of the data

Dataset name: Crimes in Chicago—An extensive dataset of crimes in Chicago (2001-2017), by City of Chicago(*resource:*<https://www.kaggle.com/datasets/currie32/crimes-in-chicago>)

Dataset content: 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 2017.

Where the data comes from: Data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system.

Dataset size: We just extract the data records from 2008 from 2017, the size of which is 1.04GB with 22 feature columns and 4,145,425 Rows.

Column information:

ID - *Unique identifier for the record.*

Case Number - *The Chicago Police Department RD Number (Records Division Number), which is unique to the incident.*

Date - Date when the incident occurred. this is sometimes the best estimate.

Block - The partially redacted address where the incident occurred, placing it on the same block as the actual address.

IUCR - The Illinois Uniform Crime Reporting code. This is directly linked to the Primary Type and Description. See the list of IUCR codes at <https://data.cityofchicago.org/d/c7ck-438e>.

Primary Type - The primary description of the IUCR code.

Description - The secondary description of the IUCR code, a subcategory of the primary description.

Location Description - Description of the location where the incident occurred.

Arrest - Indicates whether an arrest was made.

Domestic - Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act.

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. See the beats at <https://data.cityofchicago.org/d/aerh-rz74>.

District - Indicates the police district where the incident occurred. See the districts at <https://data.cityofchicago.org/d/fthy-xz3r>.

Ward - The ward (City Council district) where the incident occurred. See the wards at <https://data.cityofchicago.org/d/sp34-6z76>.

Community Area - Indicates the community area where the incident occurred. Chicago has 77 community areas. See the community areas at <https://data.cityofchicago.org/d/cauq-8yn6>.

FBI Code - Indicates the crime classification as outlined in the FBI's National Incident-Based Reporting System (NIBRS). See the Chicago Police Department listing of these classifications at http://gis.chicagopolice.org/clearmap_crime_sums/crime_types.html.

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.

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.

Year - Year the incident occurred.

Updated On - Date and time the record was last updated.

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.

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.

Location - The location where the incident occurred in a format that allows for the 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.

2. Problem Statement

Chicago is a city known throughout the world for its high crime rate. Its crime rate, especially violent crime, is consistently well above the national average. Chicago was responsible for nearly half of the increase in homicides in the United States in recent years, although the national crime rate has remained low.

The members of our group are concerned about the city of Chicago. We hope to drill down through the database of Chicago crime records to study the city's crime since 2008 and summarize the patterns of crime time, place, and type of crime.

Our research questions include: what are the changing patterns in the number of crimes per year in Chicago from 2008-2016? What is the pattern of time of day distribution? Which locations have the highest crime rates in Chicago? What communities are they in? What is the distribution of types of crime in Chicago? And so on, which will be elaborated on in Part 4.

3. Why is this Big Data?

The reason that we selected this data set is that it reflects reported incidents of crime (except murders) that occurred in the City of Chicago from 2008-2016. The data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. It's a structured data set.

It is big data, because

- **Volume:** The original dataset is 1.04 gigabytes, with 22 features and more than 4 million rows.

- **Velocity:** The data is generated fast so that can be helpful in instant decisions and seizing opportunities.
- **Variety:** It contains different kinds of crime types in various timing and location through the nine years.

4. Method and Results

4.1 Data Preprocessing

4.1.1 Getting Integrated Data

Problem: Since the original data are saved in a different csv file, we need to first put all the data records into only one CSV file in order to analyze them more easily.

Method: We used the Linux system to tackle this problem. Firstly, upload those data files into the Linux system respectively and then write them into one file. Last, we downloaded this file into our own system, named ChicagoCrime.csv.

4.1.2 Data Cleansing

We choose to use pySpark to do data cleaning tasks with the ChicagoCrime.csv file, containing over 4 million rows. PySpark could handle such a large file fastly and efficiently. We first created a PySpark data frame, named df, to store the data from ChicagoCrime.csv.

Step 1:

Drop duplicate rows: Duplicate data takes up unnecessary storage space and slows down our processing process. Also, duplicate data can skew analysis results and threaten the integrity of the data set. Thus, we use dropDuplicates function to remove the duplicate rows.

Step 2 :

Drop the column unmatched with our analysis: There are 22 columns in the original dataset; however, many of these columns are useless throughout our analysis process. Therefore, we used the drop function to remove the column Case_Number, ID, _c0, IUCR, FBI_Code, XCoordinate, Y_Coordinate, Updated_On, and Location.

Step 3:

Rename column name for analysis convenience: Because the column names in the original dataset contain lots of space, we in this step try to replace the space with an underline. After doing that, it would be more convenient and easier for us to do the SQL query in Hive and PySpark.

Step 4:

Deal with the comma in the dataset: Since in this dataset we have some text information columns, there are lots of commas existing in it. Therefore, we need to remove all the commas, which we confirmed are unnecessary, in order to successfully write the final csv file into Hue to conduct Hive SQL queries.

Step 5:

Get timestamp from a string: In the original dataset, the date and time of the records are stored in a string column, in order to do some analysis on the crime time we need to form a new timestamp column, which contains the date and time of the criminal record.

Step 6:

Get the cleaned dataset: After all the steps above we could get a well-processed dataset, which we can do much research on. And we write the processed dataset into a new csv file called Chicago_Crime_Clean.csv. Also, we also get a csv file to further conduct research on major crime types, which contains records only of the top 15 frequent crime types, named TOP15_CRIME.csv.

4.2 Time analysis

4.2.1 Methods and Tools Used

The tools used in this part of the time analysis are Hive, Linux, and Pyspark.

After processing the dataset, we intend to first analyze it in the temporal dimension in order to make a deeper analysis of the crime in Chicago. Before the analysis, we asked several questions specific to time. First, what is the trend in the total number of crimes in Chicago over the interval years included in the dataset, whether it is increasing or decreasing, and what is the rate of increase or decrease? Second, is the distribution of the number of cases in Chicago related to the month? Are there certain months when crime frequency is frequent? Is there a seasonal distribution of crime incidence? Third, is the distribution of the number of crime cases in Chicago related to the hourly distribution? What time of day do you need to be especially safe? When is it appropriate to go out and when is it inappropriate to go out? Fourth, after exploring the number of crimes in each time period examine whether there are any

crime types that are more frequent. Are there particular crimes that occur at specific times of the day?

4.2.2 The Total Number of Crimes Between 2008 and 2016

In this first graph(4.2.2), the x-axis represents the year and the y-axis represents the number of crimes. From the graph, it can be seen that from 2008 to 2016, the total number of crimes is on a downward trend. The rate of decline has been calculated to be about thirty-eight percent. This shows to some extent that the security situation in Chicago is getting better year by year.

4.2.3 Number of Crimes per Month

In the second table(4.2.3), the conclusion from running the SQL statement through hive shows the annual and monthly distribution of the number of crimes. We can see the number of crimes for each month of the year. In the graph, a darker color means a higher number of crimes. For example, in July and August 2008, one can see almost 40,000 crimes. It can also be seen that the number of crimes is higher in summer than in other months, so there is a seasonal pattern in the number of crimes. Therefore we can deduce that Chicago has a high crime rate in the summer and that if you arrive in Chicago during the summer, especially in July and August, you need to be more careful about safety.

4.2.4 The Number of Crimes in Each Period

In the third table(4.2.4), the hourly distribution of the number of crimes is shown by summing the total number of crimes for each hour. the x-axis represents the number of crimes and the vertical represents the hour. We can see from the graph that the number of crimes is low between 1:00 a.m. and 7:00 a.m., increases rapidly from 8:00 a.m., and peaks at noon and night, with the highest number of crimes at 12:00 noon and 7:00 p.m. Therefore, people need to pay more attention to safety during this period.

4.2.5 The Number of Different Crime Types at Different Times of the Day

In the fourth graph(4.2.5), we can see the distribution of the number of crimes of different crime types at different times of the day. In the graph, we can see that there are more theft crimes during the day and a lot of violence at night. It is also worth noting that there are almost no narcotics during the day, but they are more frequent at night. This suggests a temporal distribution pattern for different offender types of crime. This gives us the insight that if we go out during the day in Chicago, we need to be careful about being burglarized and avoid conflicts with others at night.

4.3 Location analysis

To make better use of this data set, and learn deeper about how to reduce Chicago's crime rate in an effective manner, location analysis is necessary.

4.3.1 Methods and Tools used

As there is latitude and longitude information in our dataset, after cleansing, it's efficient to use Tableau to transfer these two into the geographic role and draw the crime occurrence heat map from different aspects from 2008 to 2016. We Use shades of red and blue on the graph to indicate the density of crime occurring at that location.

Also, we filter the Top 15 Crime Types by occurrence using PySpark, to find more details about the most frequent crime types.

4.3.2 Crime Frequency Distribution

Overall, after drawing the crime frequency distribution (chart 4.3.2), we found that crime frequency is high in all but the downtown and outermost edges of Chicago. Specifically, more crime occurs on the west and south sides of the city.

Because this is the cumulative frequency distribution of crime occurring in Chicago over a nine-year period, it reflects the overall citywide crime situation over that time.

4.3.3 Top 15 Crime Types Frequency Distribution

Then we use tableau to draw these graphs by crime type and divide them into two categories.

The first eight types of crime distribution (chart 4.3.3 high concentration) concentration is relatively high, color shades represent the distribution density. We can find that the distribution patterns of criminal damage, Weapons Violation, battery, and motor vehicle theft are very similar, mostly in the central area, on both sides of the river.

The next seven crime types (chart 4.3.3 low concentration), are scattered throughout Chicago, with occasional concentrations of one or two. For example, you can see the Crime type of Theft located evenly throughout the city, with only one exception of a point on the east coast. Living in these areas of high concentration requires us to be extra cautious

4.3.4 Crime Frequency by Community

For statistical and planning purposes, the City of Chicago is grouped into 77 community areas. The borders of these areas typically do not change, so it is

possible to compare statistics from one period to the other. So we draw a trend bar chart with a trendline to find the crime frequency trend by the community.

The overall graph is very large, containing all 77 communities. It's hard and meaningless to show them all. Therefore, we choose some communities with comparably high counts (chart 4.3.4). For example, the 25th Community, in the westernmost part of Chicago, whose bar chart is covered with dark blue, shows a gradual downward trend throughout these nine years.

These analyses help us focus on security by the community and facilitate the allocation of police and government resources.

4.4 Crime Type Analysis

4.4.1 Methods and Tools used

We analyzed the data and results we wanted by using pySpark and MapReduce for the type of crime from 2008 to 2016. Then, using tableau to get the images of our data export, clearly showing some crime-type data. For the last section 4.4.6, we used MapReduce to get the result we wanted.

4.4.2 Top 15 Crime Types

The diagram (chart 4.4.2) shows the top 15 crime types in the period of 2008 - 2016. We can clearly see that theft is the type of crime that has occurred the most over the years, with a total of 650,753 cases. This is followed by battery and criminal damage, each with 533,945 and 334,005 cases. The fewest type of crime is only 21,379.

4.4.3 Top 15 Crime Types Offenders Arrested

According to the graph in 4.4.3, only five of the most frequent types of crimes have an arrest rate of more than 60%, which are prostitution, narcotics, weapon violation, public peace violation, and criminal trespass. The arrest rates for prostitution and narcotics exceeded 99%. All other crimes have an arrest rate of no more than 25%. This set of arrest rates reflects, to some extent, the importance that the Chicago Police Department places on different types of crimes.

4.4.4 Crime Scenes for Top 10 Crime Types

Graph 4.4.4 illustrates the locations where the most crimes occurred in the top ten crime types. It is clear to see that the top four locations, street, residence, apartment, and sidewalk, have the highest number of crimes, all reaching over 300,000, with a staggering 708,000 crimes occurring in the street. This means that between 2008 and 2016, the crime risk level of the street is much higher than the other locations.

4.4.5 Top 15 Crime Type Trends

For the top 15 crime types trends, we also made a graph to show (in alphabetical order). From 4.4.5 we can see that battery, theft, and narcotics have the most obvious downward trend, from 75.9k to 50.2k a year, 46.5k to 12.4k a year, and 88.4k to 61.2k a year, respectively. This also indicates that by and large crime in Chicago is trending downward and the overall number of crimes is decreasing.

4.4.6 Frequency of Arrests per District

We also used MapReduce to derive the total number of criminal arrests per district between 2008 and 2016. The results show that District 8 has the highest number of arrests, 207,597, followed by District 11, District 7, District 6, and District 25, all with over 177,000 arrests. The least is District 13 and District 23, both only once. The results are shown in the MapReduce section (steps 6 and 7).

5. Conclusion

Through the analysis of the dataset by time, location, and crime type. The decreasing trend of crime incidents was shown in the whole picture over 2008-2016 as well as in communities and the top 15 crime type. During the explored period, “summer time”, “District 25”, “theft” and “streets” were the top “dangerous” words. District 25 which was indicated as the most dangerous community in the dataset, only ranked the 5th place in the frequency of arrests per district. Considering the facts above, the Chicago Police Department could put more effort into emphasizing the safety issue in District 25.

Appendix

I. Pyspark code

#to write the dataset into PySpark dataframe named df

```
df = spark.read.option('header','true').csv('ChicagoCrime.csv',inferSchema=True)
```

#make sure we do not use the column with too many null values

```
df.summary("count").show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|summary|_c0|ID|Case Number|Date|Block|IUCR|Primary Type|Description|Location Description|Beat|District|
|Ward|Community Area|FBI Code|X Coordinate|Y Coordinate|Year|Updated On|Latitude|Longitude|Location|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|count|4145425|4145425|4145418|4145425|4145425|4145425|4145425|4145425|4145425|4143476|4145425|414534
1|4145348|4143930|4145425|4079697|4079697|4145425|4145425|4079697|4079697|4079697|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

#to drop duplicate rows

```
df=df.dropDuplicates()
```

#to see how many rows are in the dataset after drop duplicate rows

```
df.summary("count").show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|summary|_c0|ID|Case Number|Date|Block|IUCR|Primary Type|Description|Location Description|Beat|District|
|Ward|Community Area|FBI Code|X Coordinate|Y Coordinate|Year|Updated On|Latitude|Longitude|Location|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|count|2997930|2997930|2997926|2997930|2997930|2997930|2997930|2997930|2997930|2995998|2997930|299788
7|2997877|2997061|2997930|2945194|2945194|2997930|2997930|2945194|2945194|2945194|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

#to replace the space with underline in column names

```
from pyspark.sql import functions as F
```

```
df = df.select([F.col(col).alias(col.replace(' ', '_')) for col in df.columns])
```

#to drop unnecessary columns

```
df=df.drop('Case_Number','ID','_c0','IUCR','FBI_Code','X_Coordinate','Y_Coordinate','Updated_
On','Location')
```

#to get the timestamp column from string column which contains time and date of crime record

```
import pyspark.sql.functions as F
```

```
df=df.withColumn("new_time", F.to_timestamp("Date","MM/dd/yyyy hh:mm:ss a"))
```

#to get month and hour column in new dataframe Chicago_Crime_Clean

```
from pyspark.sql import SQLContext
```

```
sqlContext = SQLContext(sc)
```

```
df.registerTempTable('t')
```

```
Chicago_Crime_Clean=sqlContext.sql(
```

```
'select *, month(new_time) as month ,hour(new_time) as hour from t ')
```

#to remove all the commas in the dataframe

```
from pyspark.sql.functions import *
```

```
for name in Chicago_Crime_Clean.columns:
```

```
    Chicago_Crime_Clean = Chicago_Crime_Clean.withColumn(name, regexp_replace(name, ',', ' '))
```

```
#to output Chicago_Crime_Clean into one csv file for further analysis in linux and hive
```

```
Chicago_Crime_Clean.coalesce(1).write.option('header',True).csv('Chicago_Crime_Clean')
```

```
# analyze the change in the total number of crimes from 2008-2017
```

```
sqlContext.sql('select year(new_time) as year, count(Date) as Total_Case from t WHERE  
year(new_time) != 2017 group by year(new_time) order by year ').show(20)
```

year	Total_Case
2008	426964
2009	392556
2010	370141
2011	351555
2012	335670
2013	306703
2014	274527
2015	262995
2016	265462

```
#analyze the change in the total number of crimes from 2008-01 to 2017-01
```

```
sqlContext.sql('select year(new_time) as year, month(new_time) as month, count(Date) as  
Total_Case from t group by year(new_time), month(new_time) order by year,month ').show()
```

year	month	Total_Case
2008	1	33342
2008	2	29037
2008	3	33970
2008	4	35589
2008	5	38058
2008	6	37691
2008	7	40480
2008	8	40531
2008	9	37435
2008	10	37874
2008	11	33553
2008	12	29404
2009	1	30202
2009	2	28220
2009	3	33674
2009	4	32555
2009	5	35231
2009	6	34257
2009	7	35665
2009	8	35812

```
only showing top 20 rows
```

#analyze the total number of crimes from 0:00 to 12:00

```
sqlContext.sql('select hour(new_time) as hour, count(Date) as Total_Case from t group by
hour(new_time) order by hour ').show(24)
```

hour	Total_Case
0	162002
1	93111
2	78680
3	64141
4	47834
5	40004
6	48753
7	69209
8	103689
9	135588
10	129943
11	135674
12	170017

#analyze the top 3 crime type in each hour of the day

```
sqlContext.sql('SELECT hour, Primary_Type, Total_Case\
FROM \
(select hour(new_time) as hour,Primary_Type, count(Date) as Total_Case , \
rank() OVER(Partition by hour(new_time) ORDER BY count(DATE) DESC) as rk \
from t group by hour(new_time),Primary_Type order by Total_Case desc) x\
where rk<=3 ORDER BY hour').show(50)
```

hour	Primary_Type	Total_Case
0	CRIMINAL DAMAGE	23567
0	BATTERY	25960
0	THEFT	35820
1	BATTERY	25818
1	THEFT	15178
1	CRIMINAL DAMAGE	14057
2	CRIMINAL DAMAGE	12188
2	BATTERY	23454
2	THEFT	12156
3	BATTERY	19334
3	CRIMINAL DAMAGE	10482
3	THEFT	9870
4	BATTERY	13377
4	CRIMINAL DAMAGE	8136
4	THEFT	7447

(only part of the result)

#find the top 15 most frequent crime type

```
sqlContext.sql(
'select Primary_Type, count(Date) as Case_Total \
from t \
```

group by Primary_Type order by Case_Total DESC LIMIT 15').show()

Primary_Type	Case_Total
THEFT	650753
BATTERY	533945
CRIMINAL DAMAGE	334005
NARCOTICS	307289
BURGLARY	189423
OTHER OFFENSE	182209
ASSAULT	181543
MOTOR VEHICLE THEFT	133916
DECEPTIVE PRACTICE	129023
ROBBERY	118250
CRIMINAL TRESPASS	78133
WEAPONS VIOLATION	32852
PUBLIC PEACE VIOL...	25915
PROSTITUTION	21623
OFFENSE INVOLVING...	21379

#find the top 15 most frequent crime location

```
sqlContext.sql(
  'select Location_Description, count(Date) as Case_Total \
  from t \
  group by Location_Description order by Case_Total DESC LIMIT 15').show()
```

Location_Description	Case_Total
STREET	719284
RESIDENCE	491148
APARTMENT	361167
SIDEWALK	336545
OTHER	108417
PARKING LOT/GARAG...	85835
ALLEY	66557
SCHOOL, PUBLIC, B...	58949
RESIDENCE-GARAGE	57190
SMALL RETAIL STORE	54523
RESIDENTIAL YARD ...	52135
RESIDENCE PORCH/H...	51556
VEHICLE NON-COMME...	49273
RESTAURANT	47501
DEPARTMENT STORE	39482

#find the 5 crime type in top10 crime location

```
sqlContext.sql('SELECT Location_Description,Primary_Type, Total_Case\
FROM \
(select Location_Description,Primary_Type, count(Date) as Total_Case , \
rank() OVER(Partition by Location_Description ORDER BY count(DATE) DESC) as rk \
from t group by Location_Description,Primary_Type ) x\
where rk<=5 \')
```

and Location_Description in (SELECT Location_Description From t GROUP BY Location_Description \ ORDER by COUNT(Count) DESC LIMIT 10) ORDER BY Location_Description').show(50)

Location_Description	Primary_Type	Total_Case
ALLEY	NARCOTICS	20988
ALLEY	THEFT	5752
ALLEY	BATTERY	11219
ALLEY	CRIMINAL DAMAGE	5466
ALLEY	ROBBERY	8354
APARTMENT	BURGLARY	62119
APARTMENT	BATTERY	124572
APARTMENT	CRIMINAL DAMAGE	40257
APARTMENT	THEFT	36387
APARTMENT	OTHER OFFENSE	29565
OTHER	CRIMINAL DAMAGE	11093
OTHER	THEFT	35208
OTHER	DECEPTIVE PRACTICE	12482
OTHER	BATTERY	10609
OTHER	OTHER OFFENSE	9853
PARKING LOT/GARAG...	THEFT	31424
PARKING LOT/GARAG...	CRIMINAL DAMAGE	13901
PARKING LOT/GARAG...	MOTOR VEHICLE THEFT	8950
PARKING LOT/GARAG...	NARCOTICS	8364
PARKING LOT/GARAG...	BATTERY	7770

only part of the result)

#output a csv file which only contains the crime records of top15 frequent crime types

TOP15_CRIME=sqlContext.sql(

'select * \n from t \n

WHERE Primary_Type in (SELECT Primary_Type FROM t GROUP BY Primary_Type ORDER BY count(Date) DESC LIMIT 15) ')

TOP15_CRIME.coalesce(1).write.option('header',True).csv('TOP15_CRIME')

II.Linux code

#upload two files from local to linux

cd ~/Desktop/ChicagoCrime

chmod 400 S_keypair_1.pem

scp -i S_keypair_1.pem -P1422 Chicago_Crimes_2012_to_2017.csv chenwei.l@107.22.41.91:ChicagoCrime_2.csv

scp -i S_keypair_1.pem -P1422 Chicago_Crimes_2008_to_2011.csv chenwei.l@107.22.41.91:ChicagoCrime.csv

#put two files into one csv file called ChicagoCrime.csv

tail -n+2 ChicagoCrime_2.csv >> ChicagoCrime.csv

#download ChicagoCrime.csv from linux to local desktop

cd ~/Downloads

chmod 400 S_keypair.pem

scp -i S_keypair.pem -P1422 chenwei.l@107.22.41.91:ChicagoCrime.csv ~/Desktop

rm ChicagoCrime.csv

#download Chicago_Crime_Clean.csv from local to linux

cd ~/Desktop/ChicagoCrime

chmod 400 S_keypair_1.pem

**scp -i S_keypair_1.pem -P1422 Chicago_Crime_Clean.csv
chenwei.l@107.22.41.91:ChicagoCrime.csv**

#put Chicago_Crime_Clean.csv through linux to hadoop distributed file system

hdfs dfs -put ChicagoCrime.csv (have already revised the file name to ChicagoCrime before putting it to hadoop)

#Count the number of columns and print the name of each column

awk -F"," '{print NF}' ChicagoCrime.csv | head -1

head -1 Chicagocrime.csv | tr "," "\n"

```
[li.zhanyu@ip-172-31-21-73 ~]$ awk -F"," '{print NF}' ChicagoCrime.csv | head -1
17
[li.zhanyu@ip-172-31-21-73 ~]$ head -1 ChicagoCrime.csv | tr "," "\n"
Date
Block
Primary_Type
Description
Location_Description
Arrest
Domestic
Beat
District
Ward
Community_Area
Year
Latitude
Longitude
new_time
month
hour
```

#Count the most three locations and blocks where crimes happened in terms of frequency.

awk -F"," '{print \$4}' ChicagoCrime.csv | sort | uniq -c | sort -rn | head -3

```
[li.zhanyu@ip-172-31-21-73 ~]$ awk -F"," '{print $4}' ChicagoCrime.csv | sort | uniq -c | sort -rn | head -3
719284 STREET
491148 RESIDENCE
361167 APARTMENT
```

awk -F"," '{print \$2}' ChicagoCrime.csv | sort | uniq -c | sort -rn | head -3

```
[li.zhanyu@ip-172-31-21-73 ~]$ awk -F"," '{print $2}' ChicagoCrime.csv | sort | uniq -c | sort -rn | head -3
5778 001XX N STATE ST
4755 008XX N MICHIGAN AVE
4352 076XX S CICERO AVE
```


III.Hive SQL Code

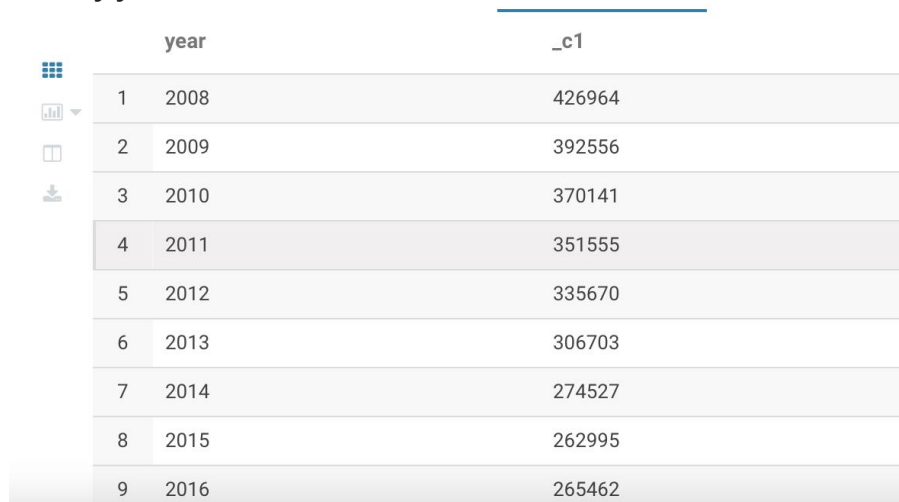
#write the ChicagoCrime.csv into Hue to do Hive SQL Query

```
CREATE TABLE chicago_crime (Date_ string, Block string, Primary_Type string,  
Description string,Location_Description string, Arrest boolean,  
Domestic boolean, Beat int, District double, Ward double, Community_Area double,  
Year int, Latitude double, Longitude string,  
new_time timestamp, month int,hour int)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","  
stored as TEXTFILE TBLPROPERTIES("skip.header.line.count"="1");
```

```
load data inpath '/user/hanye.w/ChicagoCrime.csv' into table chicago_crime;
```

#Change of the total number of crimes from 2008 to 2017

```
select year,count(year)  
from chicago_crime  
group by year  
order by year
```



	year	_c1
1	2008	426964
2	2009	392556
3	2010	370141
4	2011	351555
5	2012	335670
6	2013	306703
7	2014	274527
8	2015	262995
9	2016	265462

Total number of crimes from January 2008 to December 2017

```
select year,month,count(year) as numofcrime  
from chicago_crime  
group by year,month  
order by year,month
```

	year	month	numofcrime
1	2008	1	33342
2	2008	2	29037
3	2008	3	33970
4	2008	4	35589
5	2008	5	38058
6	2008	6	37691
7	2008	7	40480
8	2008	8	40531
9	2008	9	37435
10	2008	10	37874
11	2008	11	33553
12	2008	12	29404
13	2009	1	30202
14	2009	2	28220
15	2009	3	33674
16	2009	4	32555
17	2009	5	35231
18	2009	6	34257

#Total number of crimes from 0 to 12

```
select hour,count(year) as numofcrime
from chicago_crime
group by hour
order by hour
```

	hour	numofcrime
1	0	162002
2	1	93111
3	2	78680
4	3	64141
5	4	47834
6	5	40004
7	6	48753
8	7	69209
9	8	103689
10	9	135588
11	10	129943
12	11	135674
13	12	170017
14	13	142146
15	14	152037
16	15	162844
17	16	152665

#Statistics of crime types at each point from 0 to 12

```
select hour,primary_type,count(year) as numofcrime
```

from chicago_crime
group by hour,primary_type
order by hour

	hour	primary_type	numofcrime
1	0	STALKING	95
2	0	OFFENSE INVOLVING CHILDREN	2851
3	0	NON-CRIMINAL	4
4	0	INTERFERENCE WITH PUBLIC OFFICER	556
5	0	LIQUOR LAW VIOLATION	371
6	0	OTHER OFFENSE	10127
7	0	OBSCENITY	36
8	0	NON - CRIMINAL	1
9	0	KIDNAPPING	44
10	0	HUMAN TRAFFICKING	8
11	0	ARSON	370
12	0	HOMICIDE	335
13	0	DECEPTIVE PRACTICE	14400
14	0	CRIMINAL TRESPASS	2677
15	0	SEX OFFENSE	1128
16	0	PROSTITUTION	748
17	0	OTHER NARCOTIC VIOLATION	2

IV.MapReduce Code

#We first uploaded the dataset to HDFS.

hdfs dfs -put ChicagoCrime.csv

#Then, we created a mapper file. The code for the mapper file is shown in the screenshot.

nano cc_map.py

```

GNU nano 2.3.1                                File: cc_map.py
#!/usr/bin/python

import sys

for line in sys.stdin:
    line = line.strip()
    info = line.split(',')
    dis = info[8]
    arr = info[5]
    arr1 = 'FALSE'
    num = 0
    if dis == "District":
        continue
    if arr == arr1:
        num = 0
    else:
        num = 1
    print "%s, %s" % (dis, num)

```

#We created a reducer file. The code for the reducer file is shown in the screenshot.

nano cc_red.py

```
GNU nano 2.3.1 File: cc_red.py

import sys
count = {}

for line in sys.stdin:
    line = line.strip()
    dis, cnumber = line.split(' ', ' ')
    try:
        cnumber = int(cnumber)
    except:
        continue
    try:
        count[dis] = count[dis] + cnumber
    except:
        count[dis] = cnumber
count1 = sorted(count.items(), key = lambda x: x[1], reverse = True)
for dis in count1:
    print '%s, %s' % (dis[0], dis[1])
```

#Next, we created a bash file. The code is shown in the screenshot.

nano cc_runmr.sh

```
GNU nano 2.3.1 File: cc_runmr.sh

#!/bin/bash
hadoop jar /opt/cloudera/hadoop-streaming-3.0.0-cdh6.2.0.jar \
    -Dmapred.reduce.tasks=1 \
    -input /user/li.zhanyu/ChicagoCrime.csv \
    -output /user/li.zhanyu/output \
    -file cc_map.py \
    -file cc_red.py \
    -mapper "python cc_map.py" \
    -reducer "python cc_red.py"
```

#Run the MapReduce bash cc_runmr.sh

```
[li.zhanyu@ip-172-31-21-73 ~]$ bash cc_runmr.sh
WARNING: Use 'yarn jar' to launch applications.
22/12/05 19:49:17 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [cc_map.py, cc_red.py] [/opt/cloudera/parcels/CDH-6.2.0-1.cdh6.2.0.p0.967373/jars/hadoop-streaming-3.0.0-cdh6.2.0.jar] /tmp/streamjob8761921
47634546319.jar tmpDir=null
22/12/05 19:49:18 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-21-73.ec2.internal/172.31.21.73:8032
22/12/05 19:49:18 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-21-73.ec2.internal/172.31.21.73:8032
22/12/05 19:49:18 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /user/li.zhanyu/.staging/job_1670216450401_0026
22/12/05 19:49:19 INFO mapred.FileInputFormat: Total input files to process : 1
22/12/05 19:49:19 INFO mapreduce.JobSubmitter: number of splits:4
22/12/05 19:49:19 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
22/12/05 19:49:19 INFO Configuration.deprecation: yarn.resourceanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-pub
lisher.enabled
22/12/05 19:49:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1670216450401_0026
22/12/05 19:49:19 INFO mapreduce.JobSubmitter: Executing with tokens: []
22/12/05 19:49:19 INFO conf.Configuration: resource-types.xml not found
22/12/05 19:49:19 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/12/05 19:49:19 INFO impl.YarnClientImpl: Submitted application application_1670216450401_0026
22/12/05 19:49:19 INFO mapreduce.Job: The url to track the job: http://ip-172-31-21-73.ec2.internal:8888/proxy/application_1670216450401_0026/
22/12/05 19:49:19 INFO mapreduce.Job: Running job: job_1670216450401_0026
22/12/05 19:49:26 INFO mapreduce.Job: Job job_1670216450401_0026 running in uber mode : false
22/12/05 19:49:26 INFO mapreduce.Job: map 0% reduce 0%
22/12/05 19:49:34 INFO mapreduce.Job: map 25% reduce 0%
22/12/05 19:49:40 INFO mapreduce.Job: map 50% reduce 0%
22/12/05 19:49:46 INFO mapreduce.Job: map 75% reduce 0%
22/12/05 19:49:53 INFO mapreduce.Job: map 100% reduce 0%
22/12/05 19:50:02 INFO mapreduce.Job: map 100% reduce 100%
22/12/05 19:50:02 INFO mapreduce.Job: Job job_1670216450401_0026 completed successfully
22/12/05 19:50:02 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read:1487841
  FILE: Number of bytes written:4094714
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read:493187076
  HDFS: Number of bytes written:331
  HDFS: Number of read operations=17
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=4
  Launched reduce tasks=1
  Data-local map tasks=4
  Total time spent by all maps in occupied slots (ms)=121360
  Total time spent by all reduces in occupied slots (ms)=43860
  Total time spent by all map tasks (ms)=20228
  Total time spent by all reduce tasks (ms)=7310
  Total vcore-milliseconds taken by all map tasks=20228
  Total vcore-milliseconds taken by all reduce tasks=7310
  Total megabyte-milliseconds taken by all map tasks=124280832
  Total megabyte-milliseconds taken by all reduce tasks=44912640
Map-Reduce Framework
  Map input records=2997931
  Map output records=2997930
  Map output bytes=25562098
  Map output materialized bytes=1488418
  Input split bytes=496
  Combine input records=0
  Combine output records=0
  Reduce input groups=26
  Reduce shuffle bytes=1488418
  Reduce input records=2997930
  Reduce output records=26
  Spilled Records=5995860
  Shuffled Maps =4
  Failed Shuffles=0
  Merged Map outputs=4
  GC time elapsed (ms)=306
  CPU time spent (ms)=18200
  Physical memory (bytes) snapshot=2253758464
  Virtual memory (bytes) snapshot=35245842432
  Total committed heap usage (bytes)=2306575744
  Peak Map Physical memory (bytes)=531791872
  Peak Map Virtual memory (bytes)=7046733824
  Peak Reduce Physical memory (bytes)=237924352
  Peak Reduce Virtual memory (bytes)=7859963904
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=403185580
File Output Format Counters
  Bytes Written=331
22/12/05 19:50:02 INFO streaming.StreamJob: Output directory: /user/li.zhanyu/Final
```

Last, we get the output hdfs dfs -cat Final/part-*

```
[li.zhanyu@ip-172-31-21-73 ~]$ hdfs dfs -cat Final/part-*
8.0, 207597
11.0, 196138
7.0, 179098
6.0, 178319
25.0, 177262
4.0, 176278
3.0, 153422
9.0, 148155
12.0, 142076
5.0, 134594
15.0, 133750
19.0, 133305
10.0, 130935
18.0, 127199
2.0, 126085
1.0, 115638
14.0, 113370
16.0, 102006
22.0, 99742
17.0, 87327
24.0, 86600
20.0, 48898
31.0, 91
"", 43
13.0, 1
23.0, 1
```

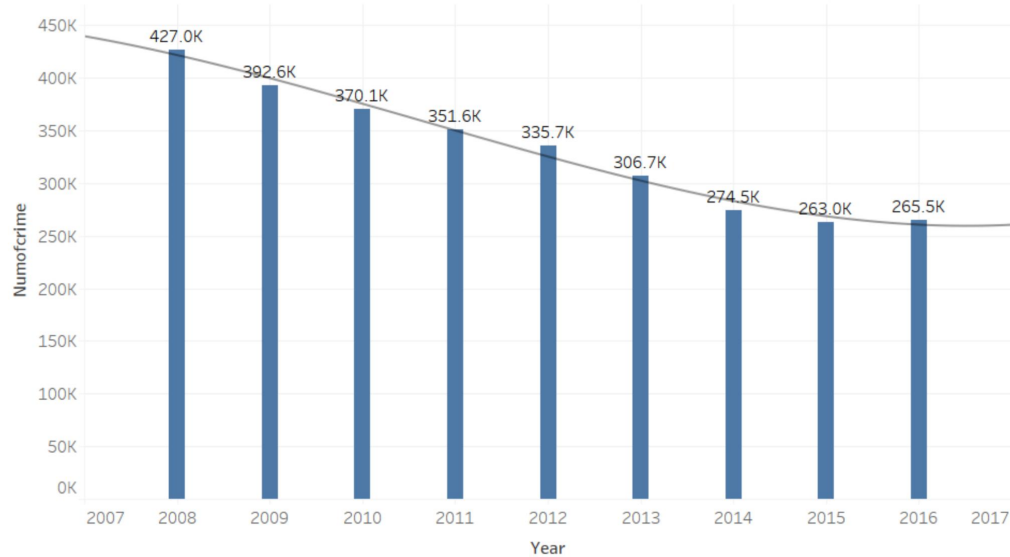
#The top five districts that had the most arrests

```
[li.zhanyu@ip-172-31-21-73 ~]$ cat ChicagoCrime.csv | python cc_map.py | python cc_red.py | head -5
8.0, 207597
11.0, 196138
7.0, 179098
6.0, 178319
25.0, 177262
```

V.Charts

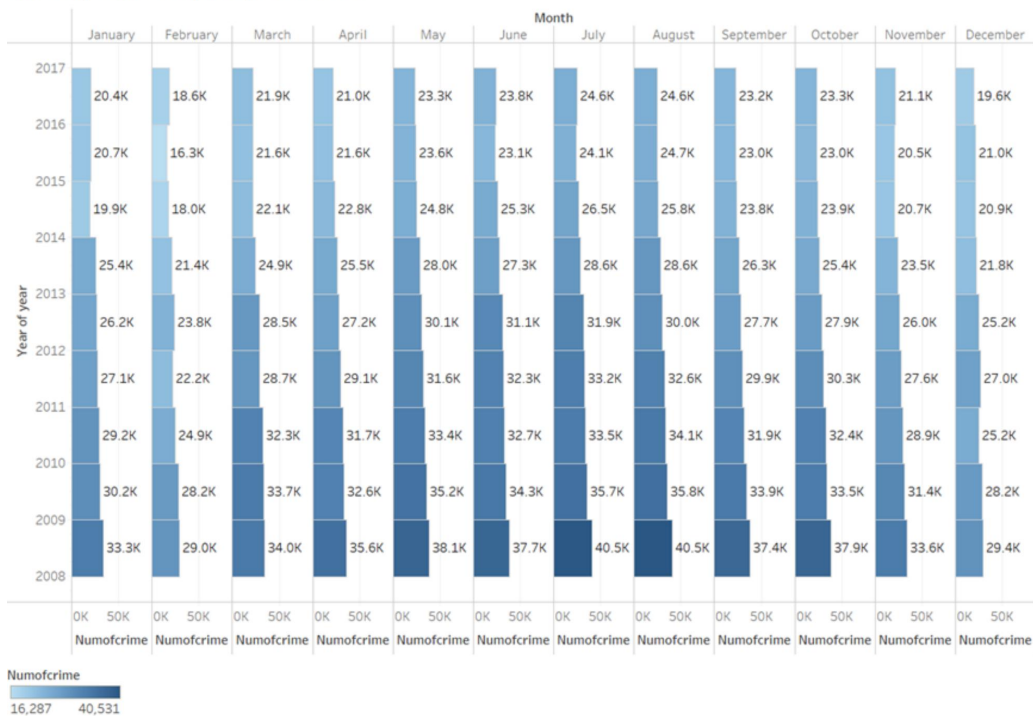
4.2.2 The total number of crimes from 2008-2016

the Change in the total number of crimes from 2008-2017



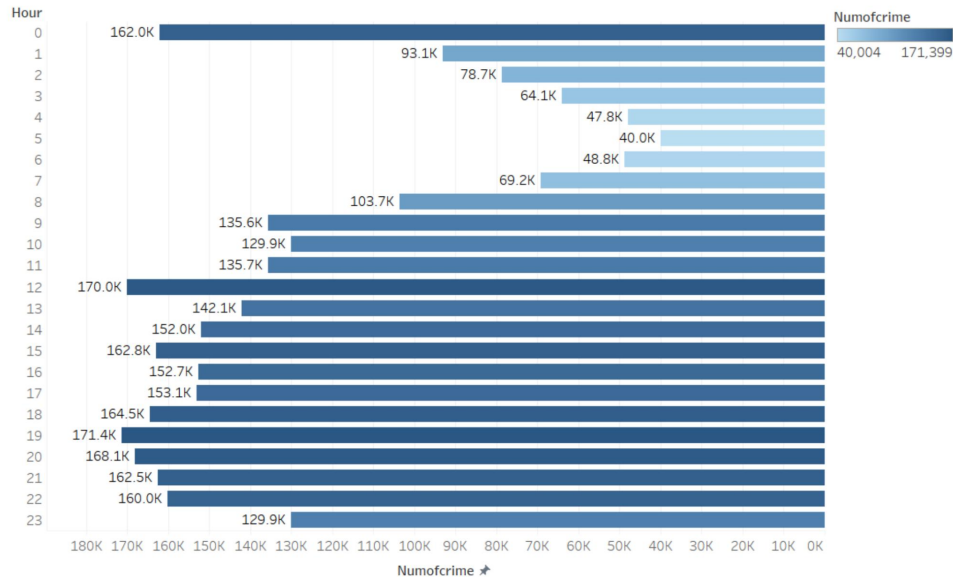
4.2.3 Monthly crime frequency 2008-2016

Monthly crime frequency 2008-2016



4.2.4 Number of crimes by

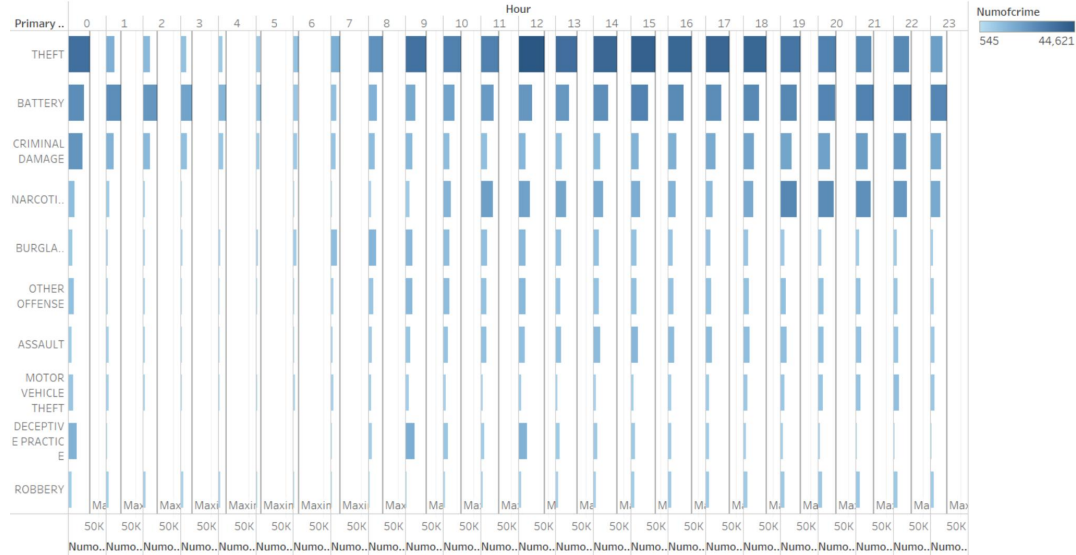
Num of Crime by Hour



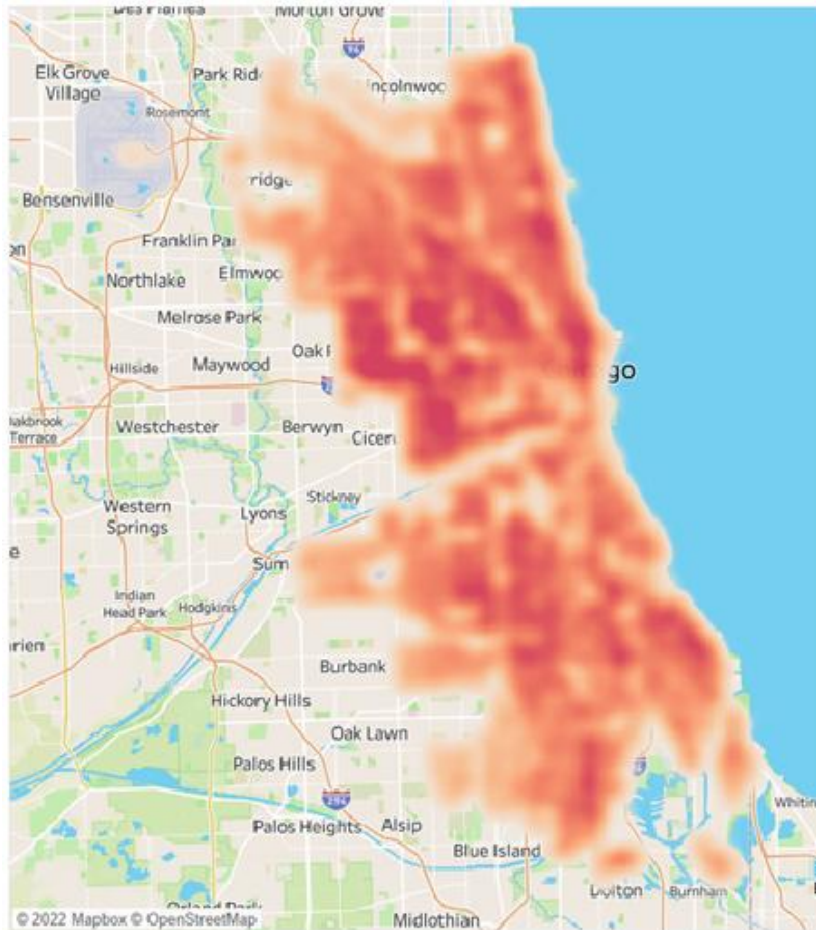
hour

4.2.5 Frequency of Top 10 Crime Types According to Time of a Day

Distribution of crime types according to time of day

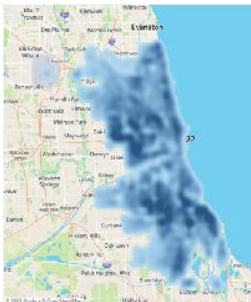


4.3.2 Crime Frequency Heat Map (2008-2016)

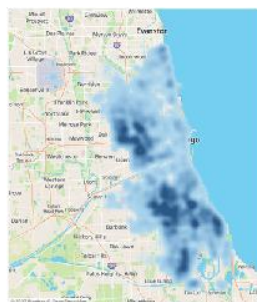


4.3.3 Top 15 Crime Types Heat Map (High concentration)

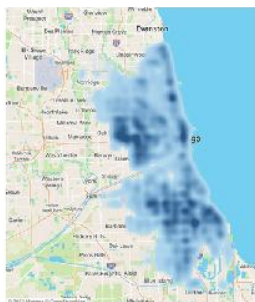
Criminal Damage



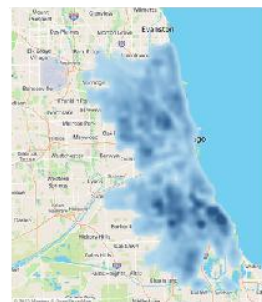
Weapons Violation



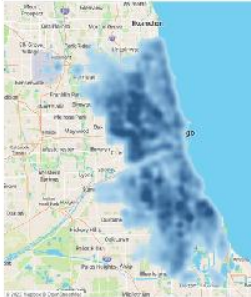
Robbery



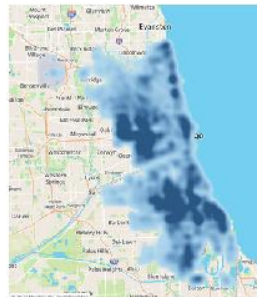
Burglary



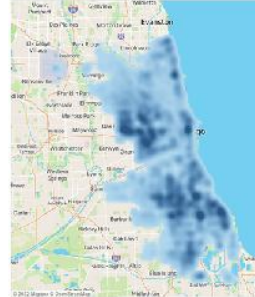
Motor Vehicle Theft



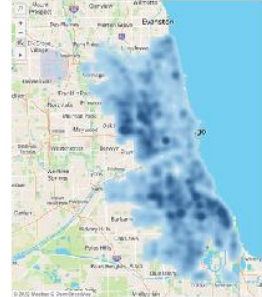
Battery



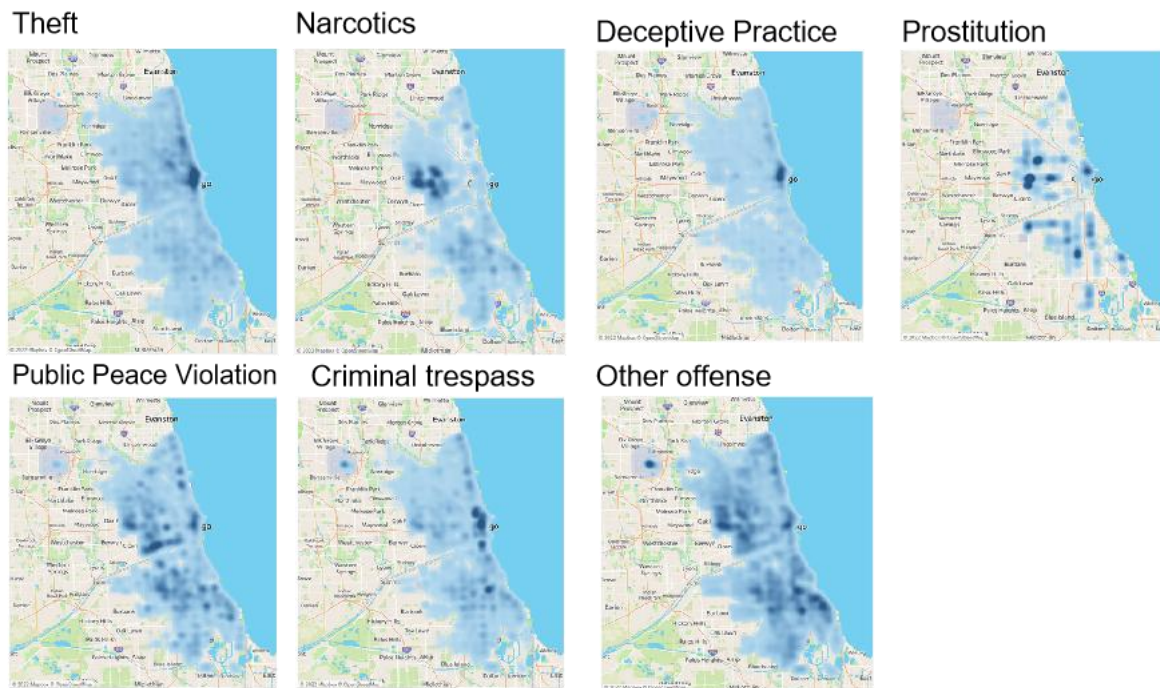
Assault



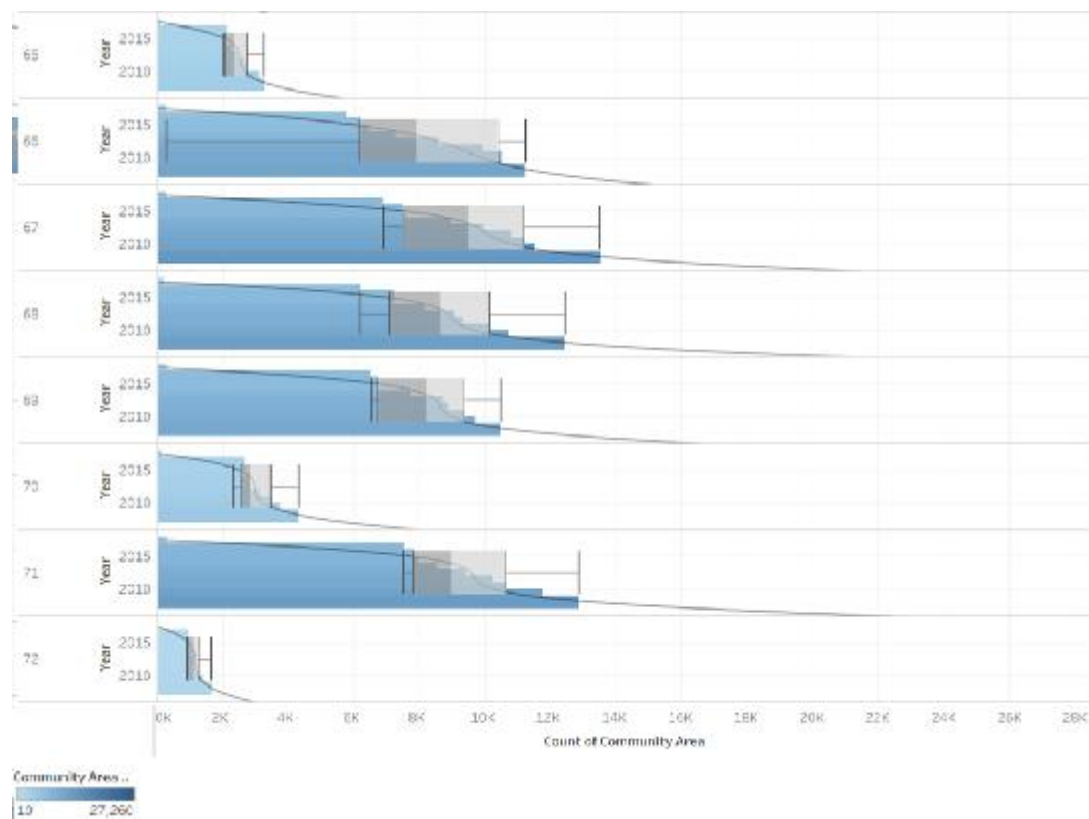
Offensive Involving Children

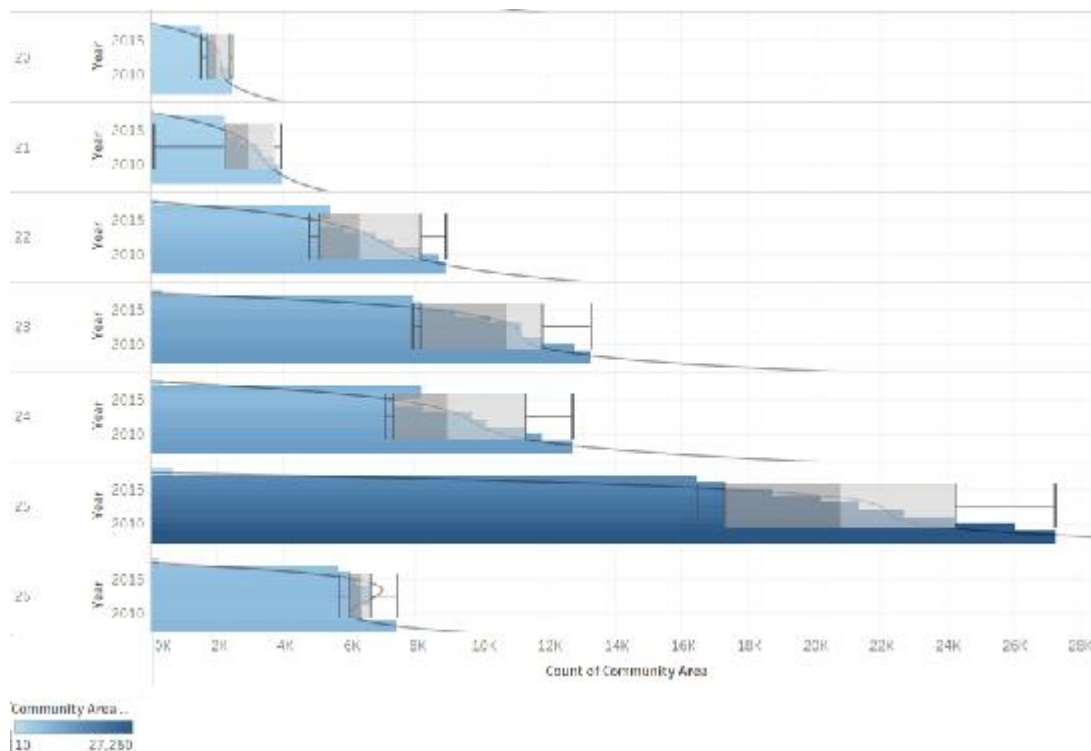


4.3.3 Top 15 Crime Types Heat Map (Low concentration)

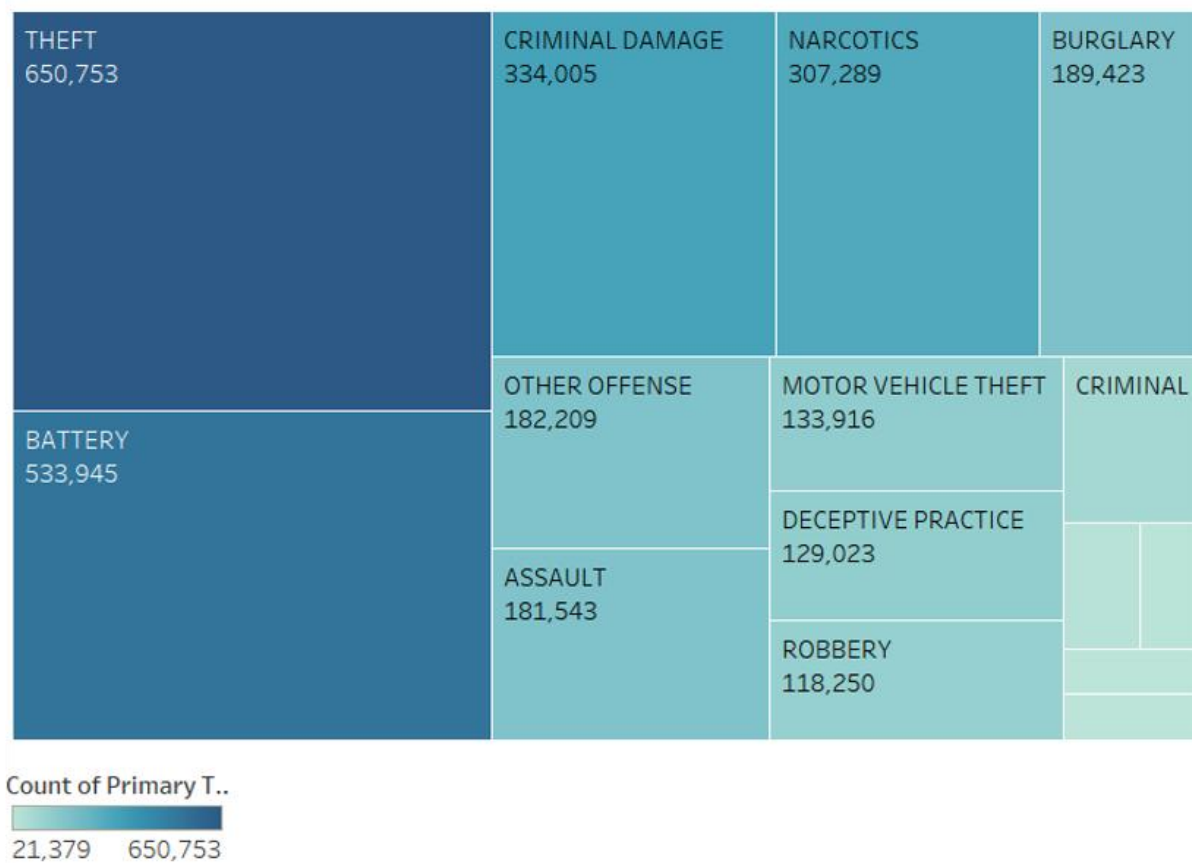


4.3.4 Crime Frequency by Community (Showing only part of the whole picture with the highest frequency of crime)

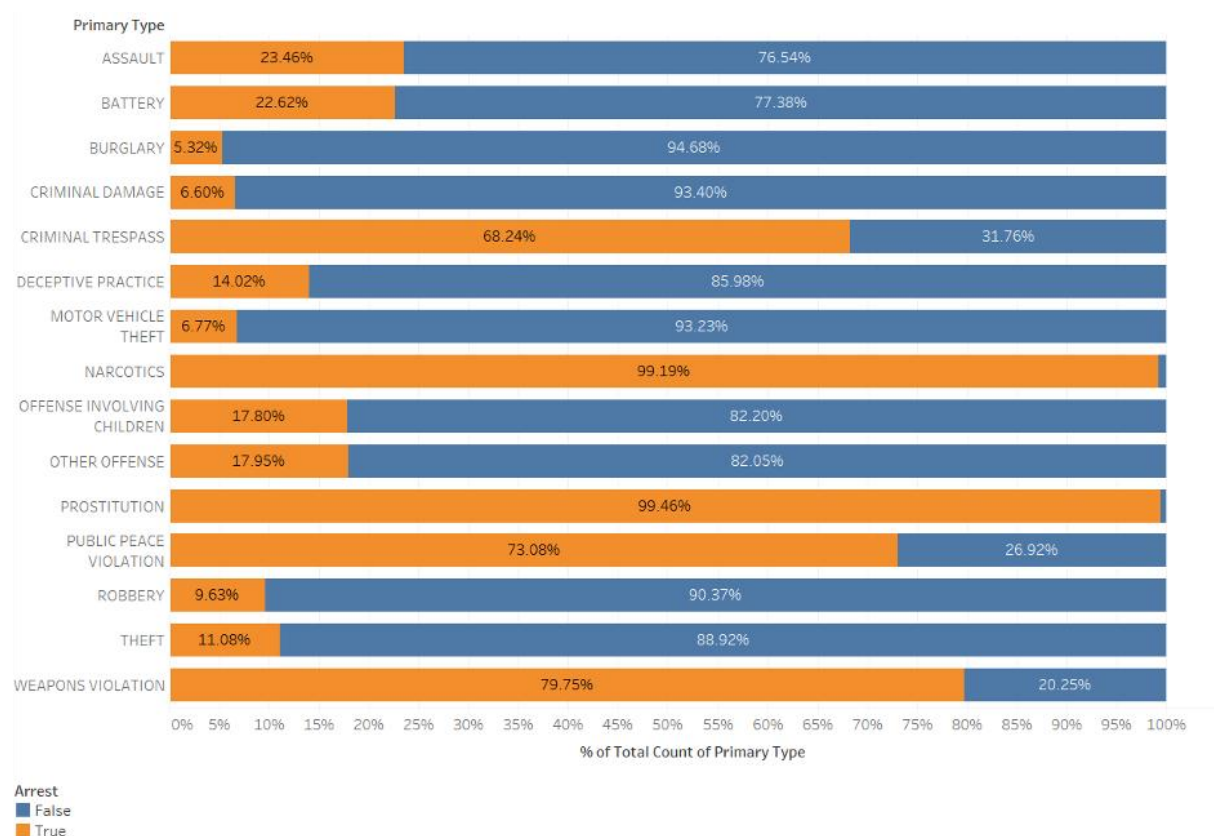




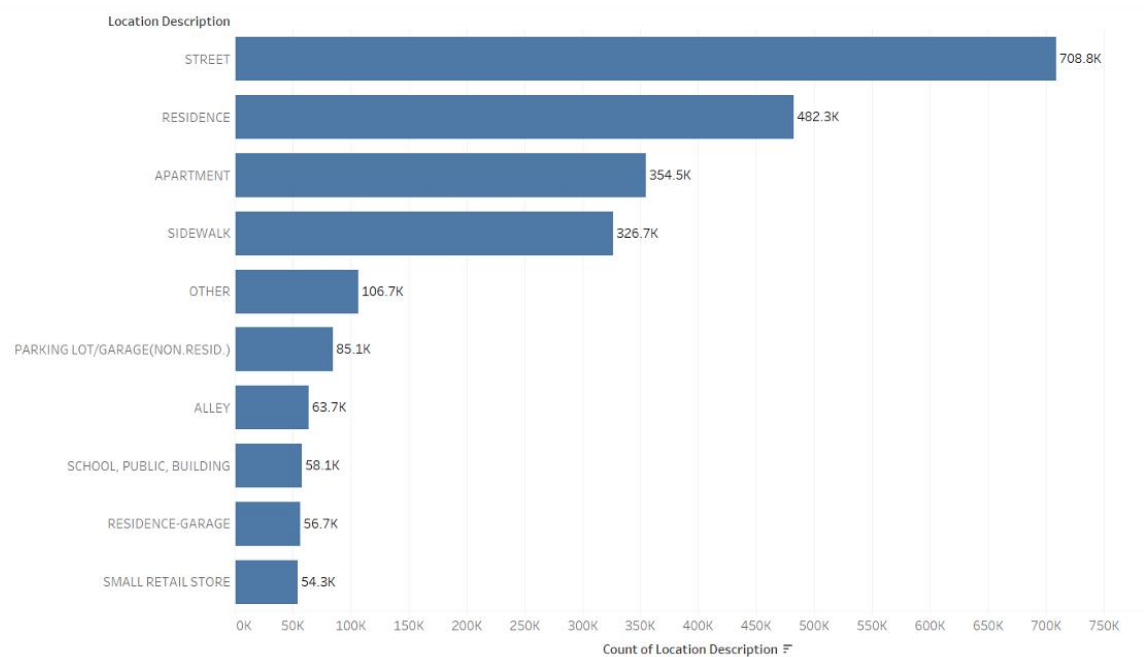
4.4.2 Top 15 Crime Types



4.4.3 Top 15 Crime Types Offenders Arrested in Percentage



4.4.4 Crime Scenes for Top 10 Crime Types:



4.4.5 Top 15 Crime Type Trends:

