Kalind Joshi | Ayush Dodal | Nirmiti Patil

002752274 | 002770479 | 002103927

2023

Big Data Architecture & Governance

Northeastern University

1. Contents

2. Assignment..... 2
2.1. Case..... 2
2.2. Assignment Goals..... 2
2.2.1. Visualization Deliverables..... 3
2.2.2. OTHER DELIVERABLES..... 3
3. Documentation..... 4
3.1. Vision Diagram..... 4
3.2. Data Wrangling and Cleansing..... 5
3.3. Database Installation..... 6
3.4. Data Mapping and Integration..... 7
3.5. Data Validation and Data Visualization..... 8
3.6. System Integration and User Acceptance Testing..... 9
3.7. Challenges Encountered..... 10
3.8. End User Instructions..... 11

2. Assignment

2.1. Case

Each team should select a dataset to analyze and build an analytical dashboard as a Proof-of-concept to illustrate the value of data-driven analytics. You need to present your dataset.

.

2.2. Assignment Goals

To work with datasets, Perform/Create:

- Create your group assignment project in Velero:
  - Project
  - Project Plan
  - Resource Allocation
  - Timesheet
  - Issues & Risks.
  - You are required to report on your team's progress every week

- **Data Profiling** – Using the Python profiling library, describe your understanding of the data.

- **Data Wrangling and Cleansing** – Use a Python program to:
  - Filtering and aggregating if needed.
  - Missing value handling.
  - Deriving additional columns from existing datasets if needed.
  - Cleaning (removing blank spaces, formatting dates, Capitalizing, etc.).

- **Database Installation:** Install the NEO4J database.

- **Business and Technical Metadata** – develop a business term list describing all the data elements available in the file.
- **Data Validation** – Validate the data using python data libraries.

- **Data Visualization** – Create a presentation dashboard to reflect your understanding of the data, you may use python visualization libraries, Power BI, or Tableau
- **System Integration and User Acceptance Testing** - Test Cases – describe your validation & testing process.
- **Risks/Issues** – identify risks and issues related to your project and describe how you resolved them.
- **End User Instructions (Steps to run your Dashboard)** – provide a full description of how to run your process:
    - Database Creation and load.
    - Visualization interpretation - describe information regarding your findings.

2.2.1. Visualization Deliverables

Once you wrangle/clean/join/integrate the data, import the data into **NEO4J** and illustrate how to use the appropriate graph to illustrate various aspects of analysis. Provide a complete explanation of the dashboard and usability.

Questions to answer:

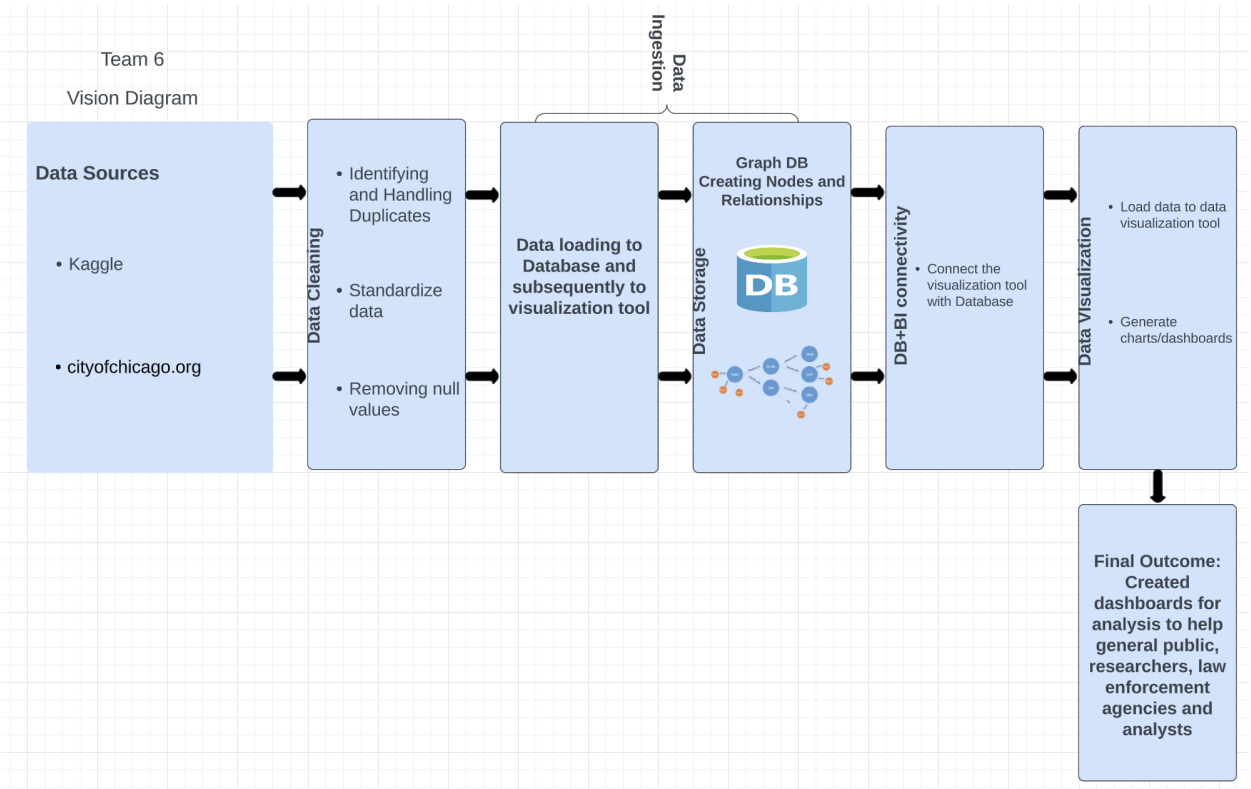
- Who is using this dashboard and how do they benefit from your dashboard?
- What value would be generated using this dashboard?
- What data are used for dimensions and columns are used for measurement?
- Did you generate any new dimensions?

2.2.2. OTHER DELIVERABLES

- Presentation of the entire work from the first step to the dashboards including the Velero screenshots.
- Business and technical metadata presentation – Identifying all available business terms and extracting related technical metadata.
- Complete instructions on how to implement and run the database load, technical metadata extraction, and dashboard.

3. Documentation

3.1. Vision Diagram



3.2. Data Wrangling and Cleansing

We began pre-processing by loading our dataset into python and studying it to understand all of its attributes and interconnections. The Chicago Crime Dataset has almost 8 million records in total, spanning from 2001 to 2017. They are divided into 4 files, each file covering 4 years.

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
nRowsRead = 1000 # specify 'None' if want to read whole file
# Chicago_Crimes_2001_to_2004.csv may have more rows in reality, but we are only loading/previewing the first 1000 rows
df1 = pd.read_csv('Chicago_Crimes_2001_to_2004.csv', error_bad_lines=False)#''', delimiter=',', nrows = nRowsRead'''
df2 = pd.read_csv('Chicago_Crimes_2005_to_2007.csv', error_bad_lines=False) #delimiter=',', nrows = nRowsRead)
df3 = pd.read_csv('Chicago_Crimes_2008_to_2011.csv', error_bad_lines=False) #delimiter=',', nrows = nRowsRead)
df4 = pd.read_csv('Chicago_Crimes_2012_to_2017.csv', error_bad_lines=False) #delimiter=',', nrows = nRowsRead)
df1.dataframeName = 'Chicago_Crimes_2001_to_2004.csv'
df2.dataframeName = 'Chicago_Crimes_2005_to_2007.csv'
df3.dataframeName = 'Chicago_Crimes_2008_to_20011.csv'
df4.dataframeName = 'Chicago_Crimes_2012_to_2017.csv'
df = pd.concat([df1, df2, df3,df4], ignore_index=False, axis=0)
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
b'Skipping line 1513591: expected 23 fields, saw 24\n'
D:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (17,20) have mixed types.Speci
pe option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
b'Skipping line 533719: expected 23 fields, saw 24\n'
b'Skipping line 1149094: expected 23 fields, saw 41\n'
```

```
There are 7941282 rows and 23 columns
```

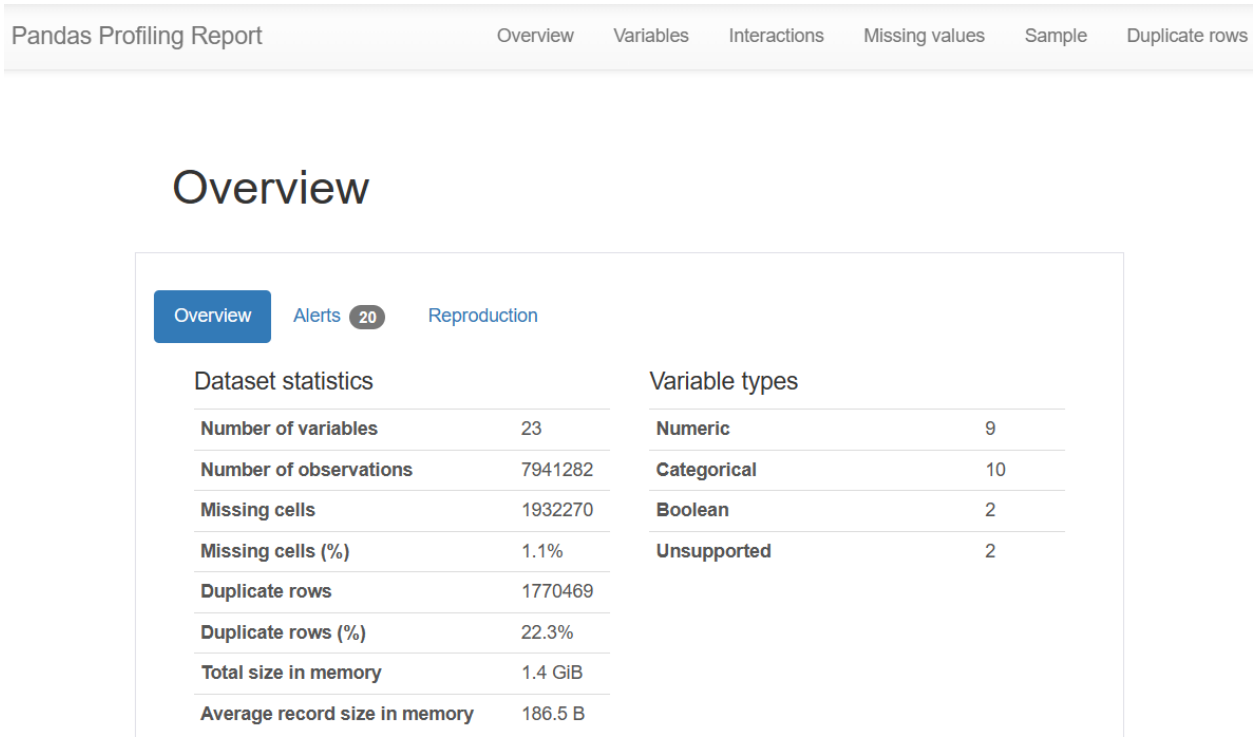
Data Profiling:-

There are 22 fields and we did a deep dive into which attributes are important and which ones do not have a major impact.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7941282 entries, 0 to 1456713
Data columns (total 23 columns):
 #   Column                Dtype
---  ------                -----
 0   Unnamed: 0            int64
 1   ID                    int64
 2   Case Number           object
 3   Date                  object
 4   Block                 object
 5   IUCR                  object
 6   Primary Type          object
 7   Description           object
 8   Location Description  object
 9   Arrest                bool
 10  Domestic              bool
 11  Beat                  int64
 12  District              float64
 13  Ward                  float64
 14  Community Area        float64
 15  FBI Code              object
 16  X Coordinate          float64
 17  Y Coordinate          object
 18  Year                  float64
 19  Updated On            object
 20  Latitude              object
 21  Longitude             float64
 22  Location              object
dtypes: bool(2), float64(6), int64(3), object(12)
memory usage: 1.3+ GB
```

Here we can see that there are minimal missing values in the dataset. However, the duplicate rows are high in number and need to be removed. They can reduce data quality and result in inaccurate data visualizations and reports.



Pandas Profiling Report | Overview | Variables | Interactions | Missing values | Sample | Duplicate rows

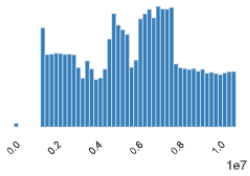
Variables

ID

ID

Real number (ℝ)

Distinct	6170812	**Minimum**	634
Distinct (%)	77.7%	**Maximum**	10827880
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	5926070.9	**Memory size**	121.2 MiB



More details

Pandas Profiling Report | Overview | Variables | Interactions | Missing values | Sample | Duplicate rows

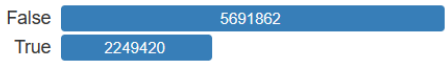
Variables

Arrest

Arrest

Boolean

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	68.2 MiB



More details

Data Cleaning:-

Got rid of duplicate values.

```
#Dropping Duplicates
df.drop_duplicates()
```

	Unnamed: 0	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arrest	...	Ward	Communit Are
0	879	4786321	HM399414	01/01/2004 12:01:00 AM	082XX S COLES AVE	0840	THEFT	FINANCIAL ID THEFT: OVER $300	RESIDENCE	False	...	7.0	46
1	2544	4676906	HM278933	03/01/2003 12:00:00 AM	004XX W 42ND PL	2825	OTHER OFFENSE	HARASSMENT BY TELEPHONE	RESIDENCE	False	...	11.0	61
2	2919	4789749	HM402220	06/20/2004 11:00:00 AM	025XX N KIMBALL AVE	1752	OFFENSE INVOLVING CHILDREN	AGG CRIM SEX ABUSE FAM MEMBER	RESIDENCE	False	...	35.0	22
3	2927	4789765	HM402058	12/30/2004 08:00:00 PM	045XX W MONTANA ST	0840	THEFT	FINANCIAL ID THEFT: OVER $300	OTHER	False	...	31.0	20
4	3302	4677901	HM275615	05/01/2003 01:00:00 AM	111XX S NORMAL AVE	0841	THEFT	FINANCIAL ID THEFT:$300 &UNDER	RESIDENCE	False	...	34.0	49

Here we are cleaning the column ‘Block’ by extracting the street name and dropping out the unnecessary data.

```
# Splitting Block column by House no, Street  and Unit to extract just the street name from the column
df['house no.'] = df.Block.apply(lambda x: x.split(' ')[0])
df['xyz'] = df.Block.apply(lambda x: x.split(' ')[1])
df['street'] = df.Block.apply(lambda x: x.split(' ')[2])
df['street end'] = df['Block'].str.extract(r'(\w+)$')
df["Address block"] = df['street'].astype(str) +" "+ df["street end"]

#Drooping out other parameters apart from street name
df = df.drop('xyz',axis = 1)
df = df.drop('street',axis = 1)
df = df.drop('street end',axis = 1)
df = df.drop('house no.',axis = 1)
df = df.drop('Block',axis = 1)

#Remaining extracted Street name by Block
df=df.rename(columns={'Address block':'Block'})
```

Another important change we made was the addition of a new attribute : UniqueID.

We added this because the format of ‘Location’ is not compatible with Neo4j.

More details are mentioned in the Challenges Encountered section.

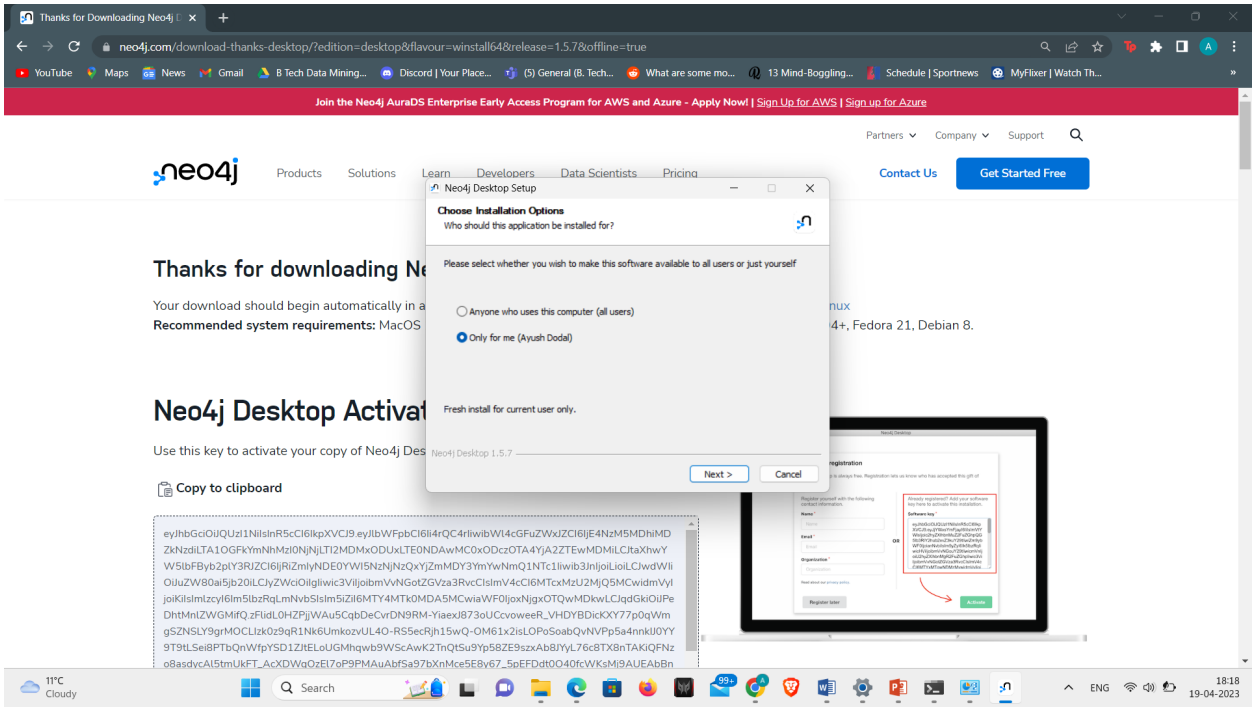
```
# create a new column 'UniqueID' and assign a unique ID to each (Latitude, Longitude) pair
df3['UniqueID'] = df3['Location'].apply(lambda x: hash(x))

# print the dataframe to verify the results
df3.head()
```

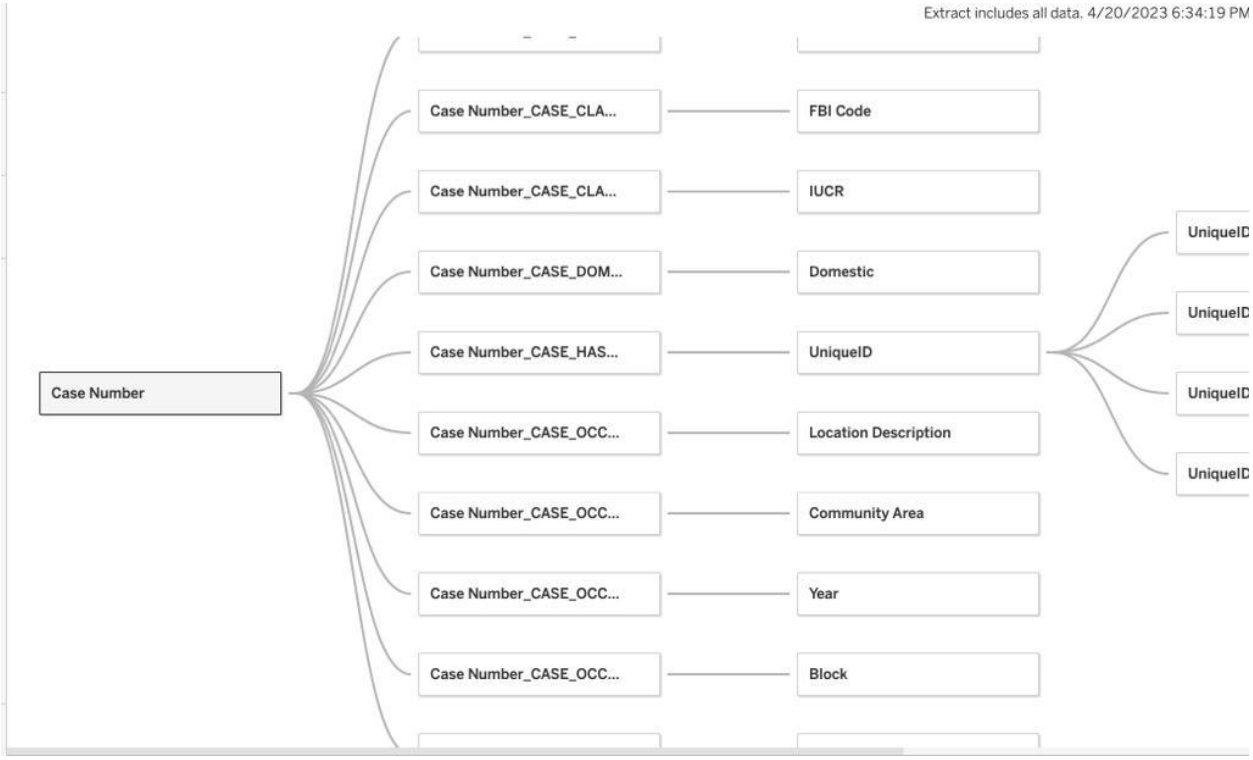
imary Type	Description	Location Description	Arrest	...	District	Ward	Community Area	FBI Code	X Coordinate	Y Coordinate	Year	Updated On	Location	UniqueID
IICIDE	FIRST DEGREE MURDER	ALLEY	True	...	3.0	6.0	69.0	01A	1178207.0	1855308.0	2008	08/17/2015 03:03:40 PM	(41.758275857, -87.622451031)	386106639535259970
IICIDE	FIRST DEGREE MURDER	STREET	True	...	15.0	24.0	25.0	01A	1144200.0	1895857.0	2008	08/17/2015 03:03:40 PM	(41.87025207, -87.746069362)	-623691163198600792
IICIDE	FIRST DEGREE MURDER	PARK PROPERTY	False	...	8.0	18.0	66.0	01A	1157314.0	1859778.0	2008	08/17/2015 03:03:40 PM	(41.770990476, -87.698901469)	-4740121907779104319
IICIDE	FIRST DEGREE MURDER	RESTAURANT	False	...	15.0	37.0	25.0	01A	1141065.0	1904824.0	2008	08/17/2015 03:03:40 PM	(41.894916924, -87.757358147)	7580557972628098219
IICIDE	FIRST DEGREE MURDER	GARAGE	False	...	10.0	22.0	30.0	01A	1154123.0	1886297.0	2008	08/17/2015 03:03:40 PM	(41.843826272, -87.709893465)	7798137727751833469

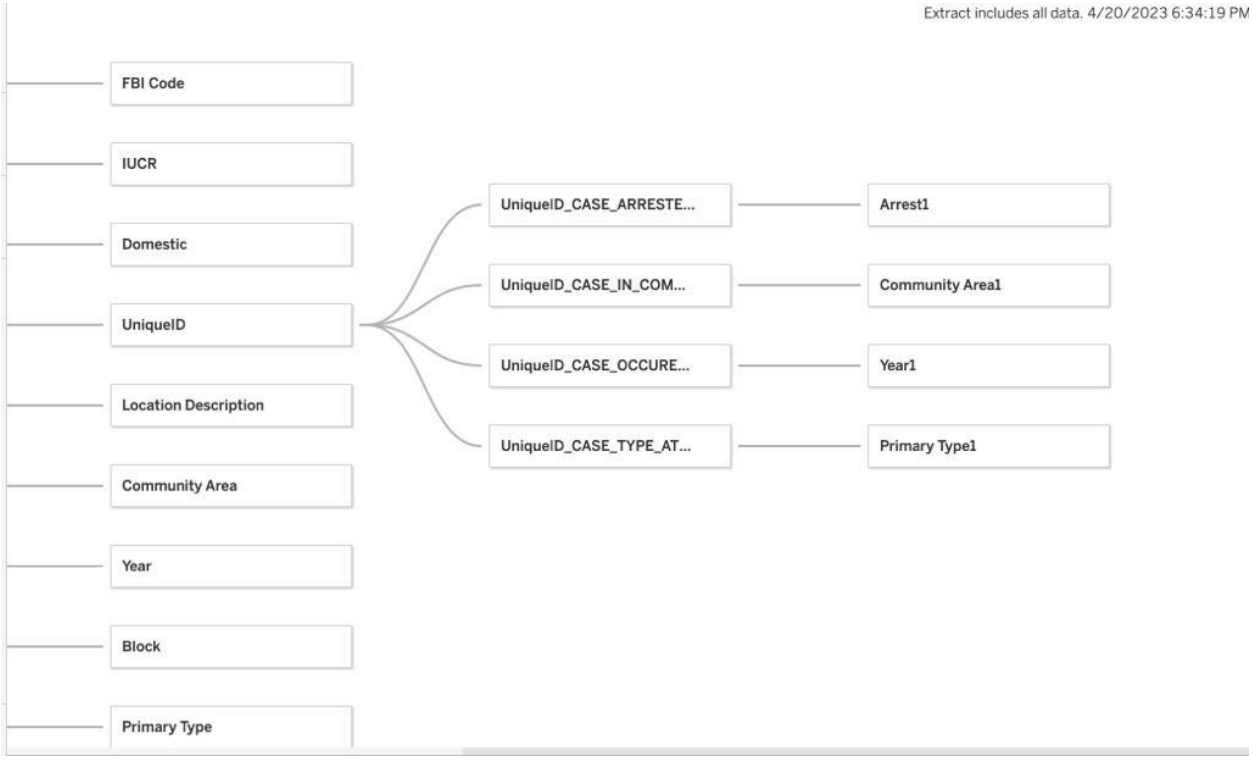
3.3. Database Installation

Downloaded the .exe file from the neo4j website and installed the application on our device.

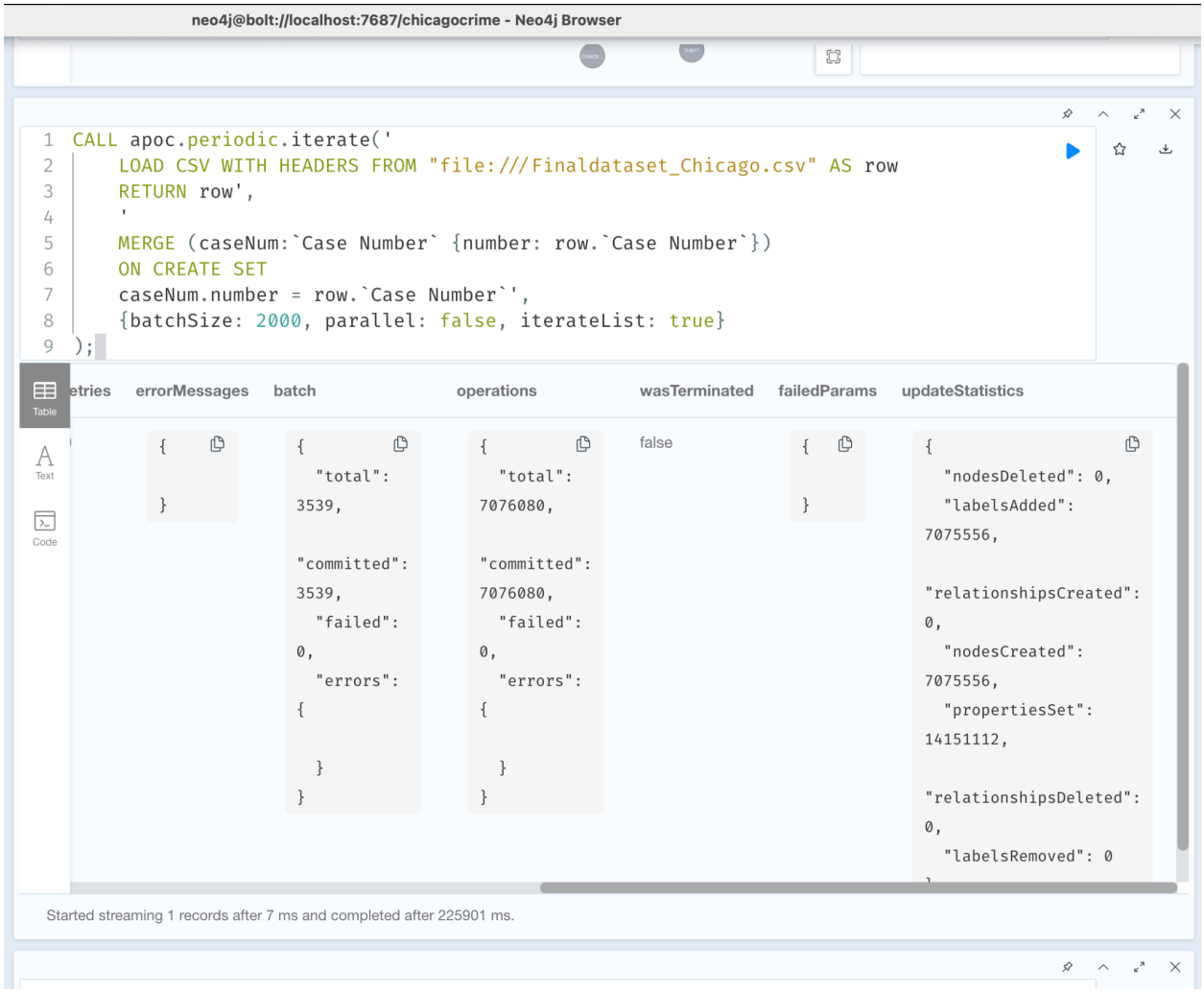


3.4. Data Mapping and Integration

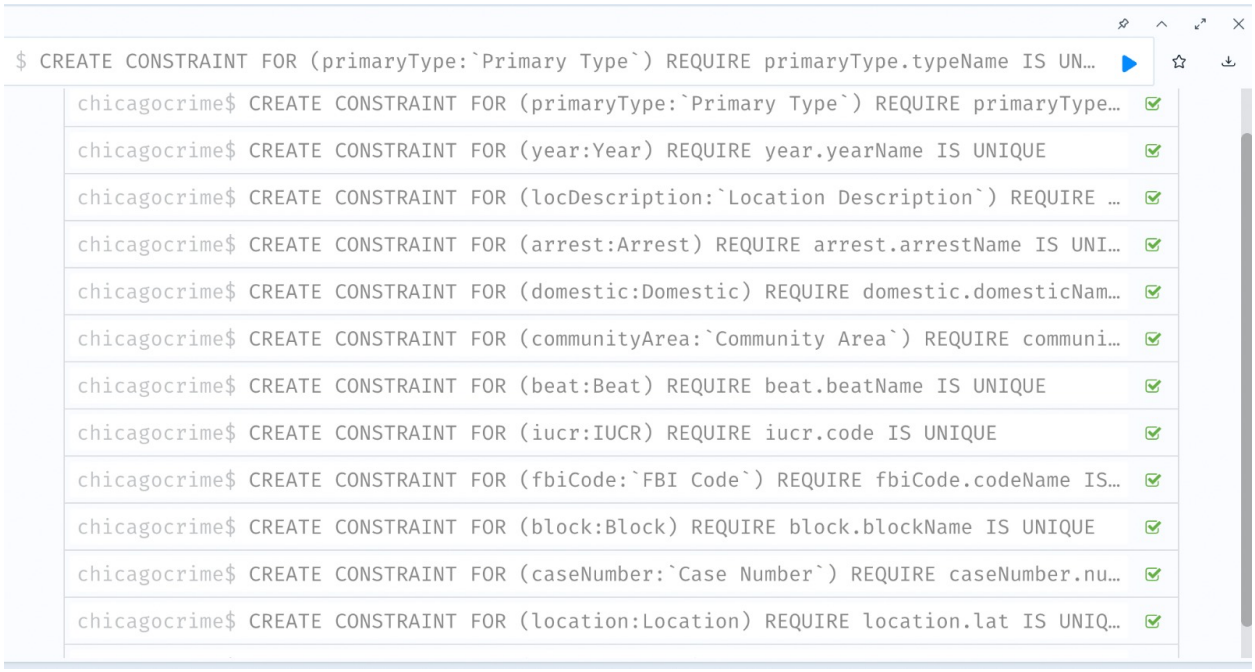




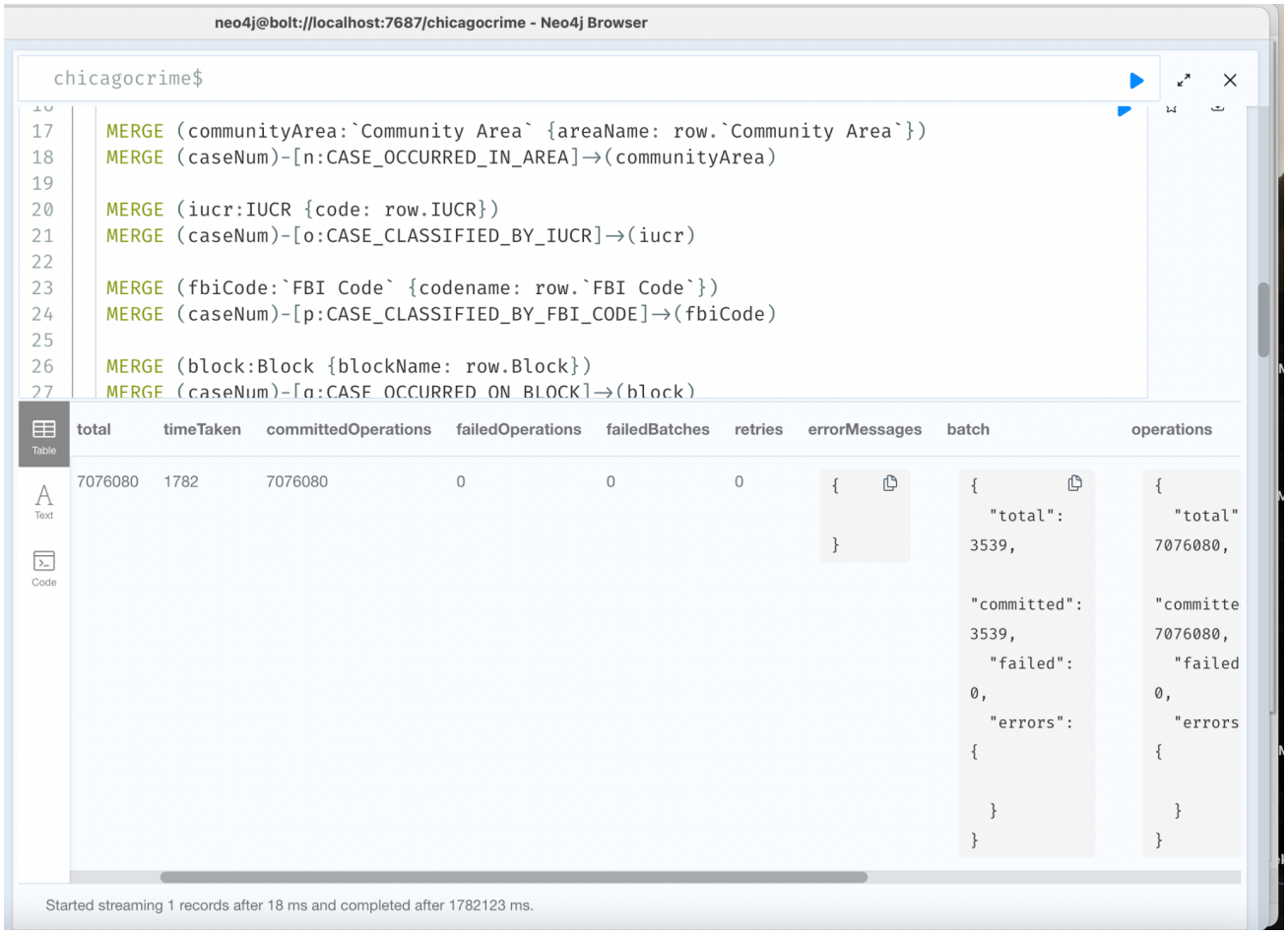
- After cleaning the dataset, we created nodes. The following is an example of creation of node ‘Case Number’:



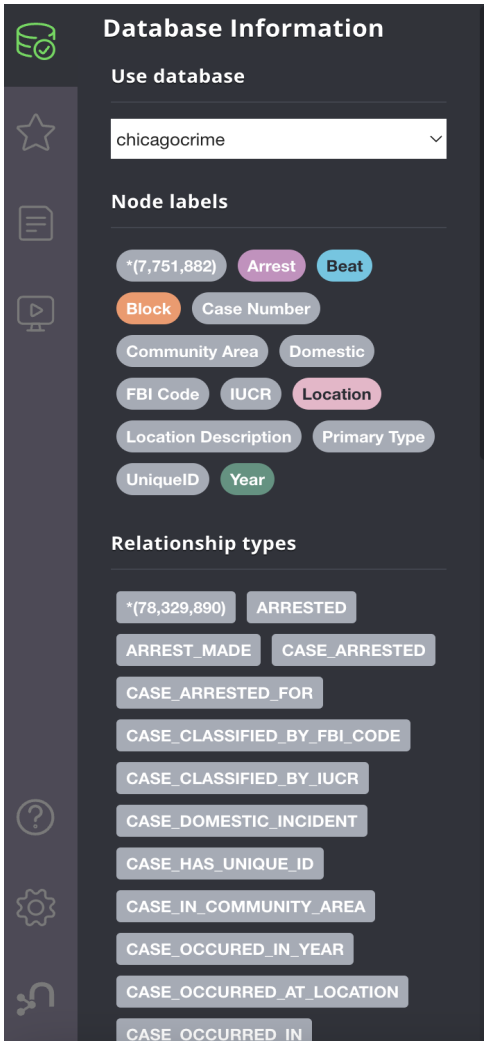
- Creating nodes and constraints for each nodes:



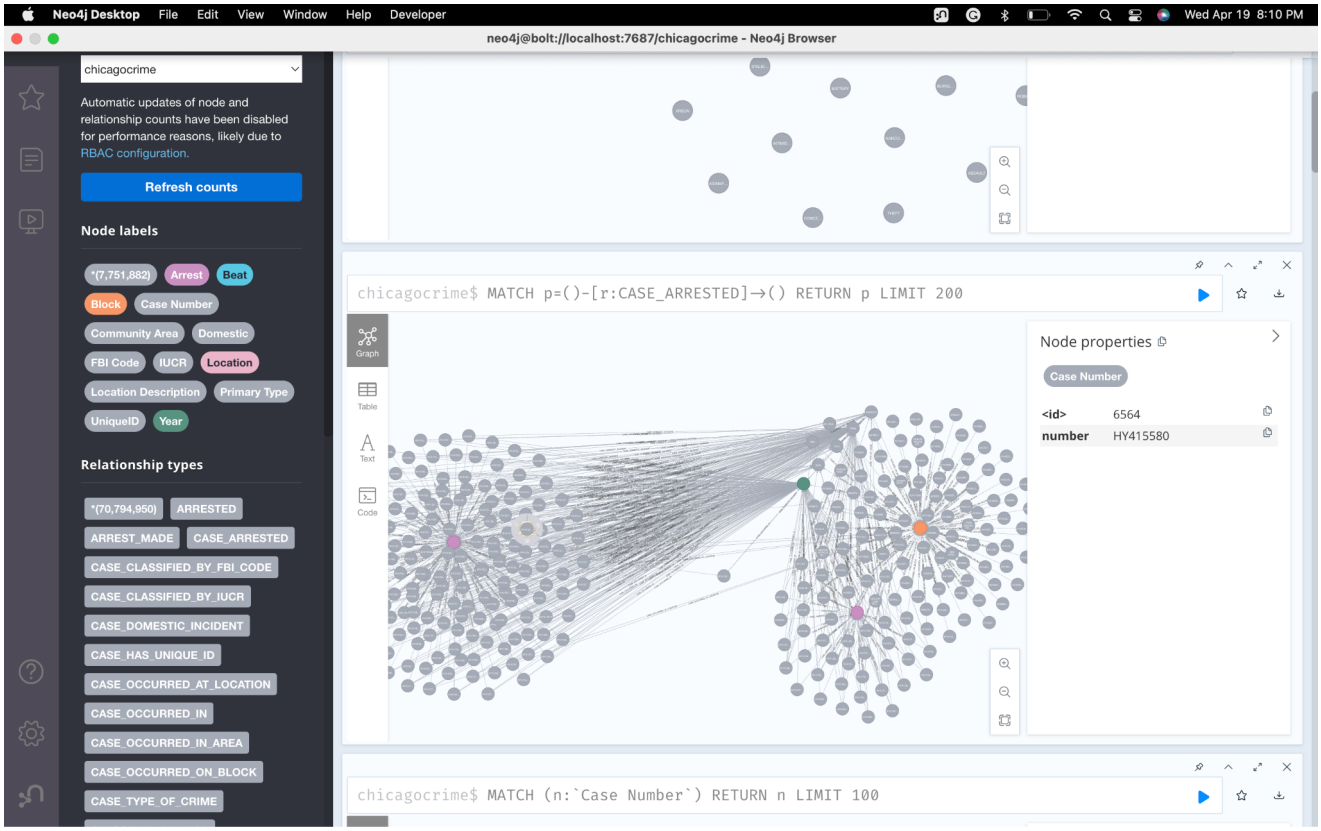
- After creating all the nodes and uploading their related data, we created all the relationships as follows:



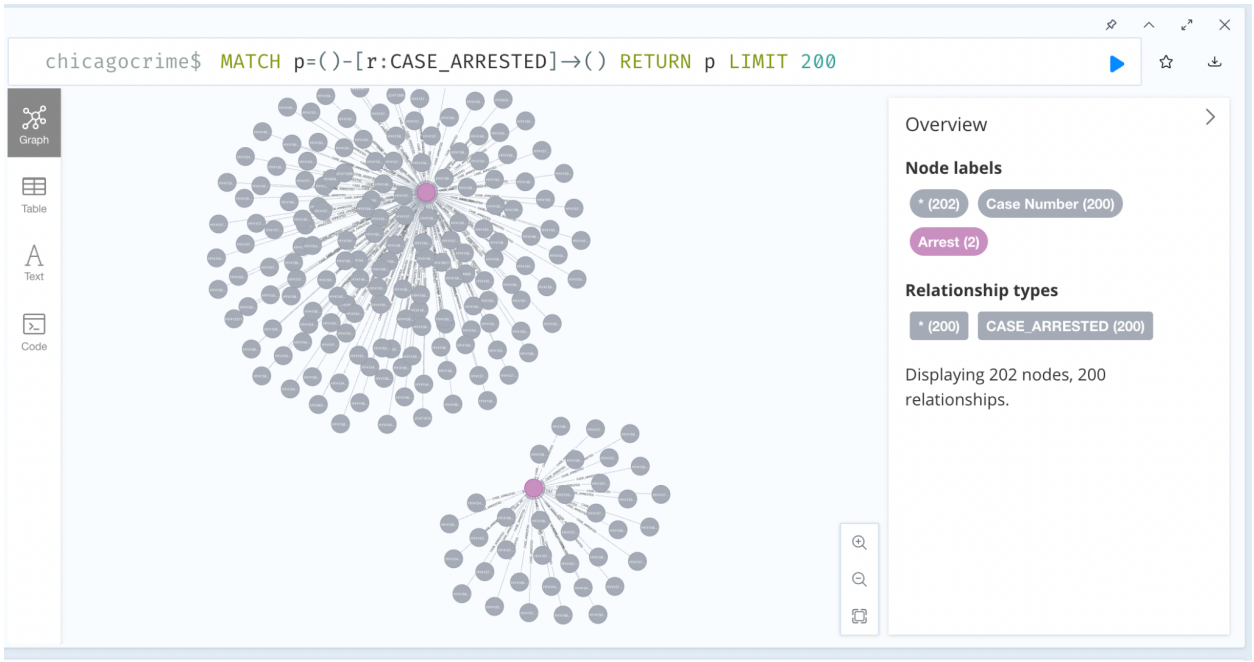
- For data integration, we have successfully created 12 nodes and established 26 relationships in our Neo4j database. This can be visualized through the uploaded data, which showcases a total of 7.7 million records and 75 million combinations of relationships.



- After creating the nodes and relations according to our database model, the graph looks like follows for ‘Case Arrested’:

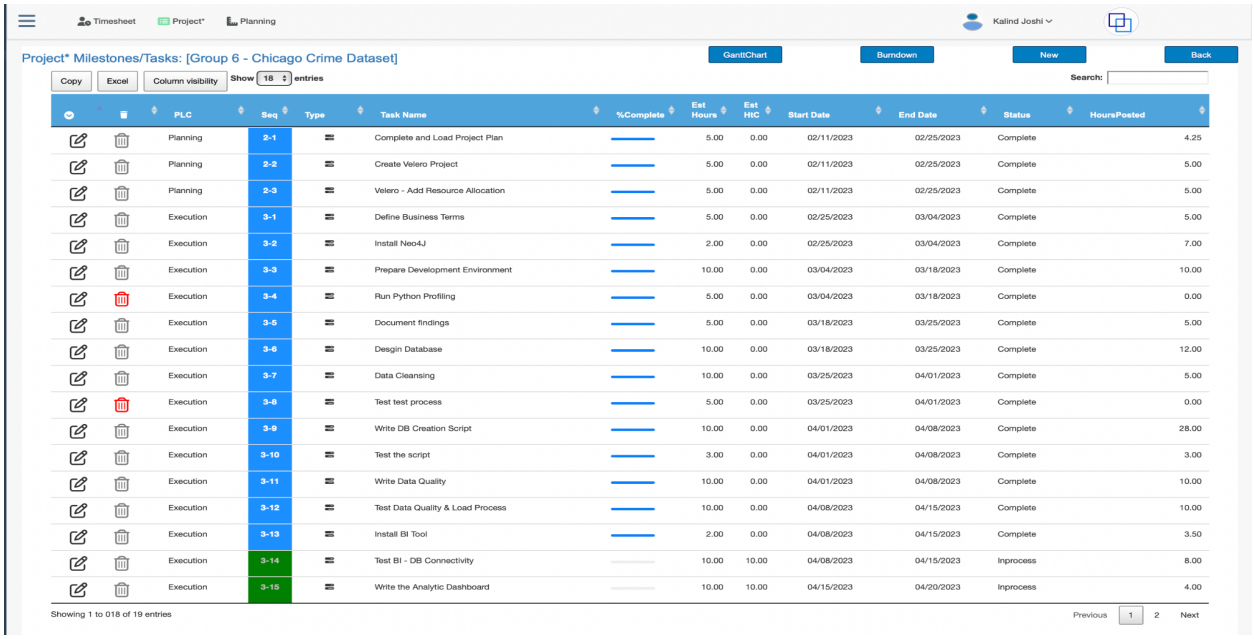


- Example of data mapping: Graph showing ‘Case Number’ for which arrest was made [TRUE] or not [FALSE] (with limit to 200)



Velero ETP Screenshots:

- Plan for the project:



Timesheet | Project* | Planning | Kalind Joshi

Project* Milestones/Tasks: [Group 6 - Chicago Crime Dataset]

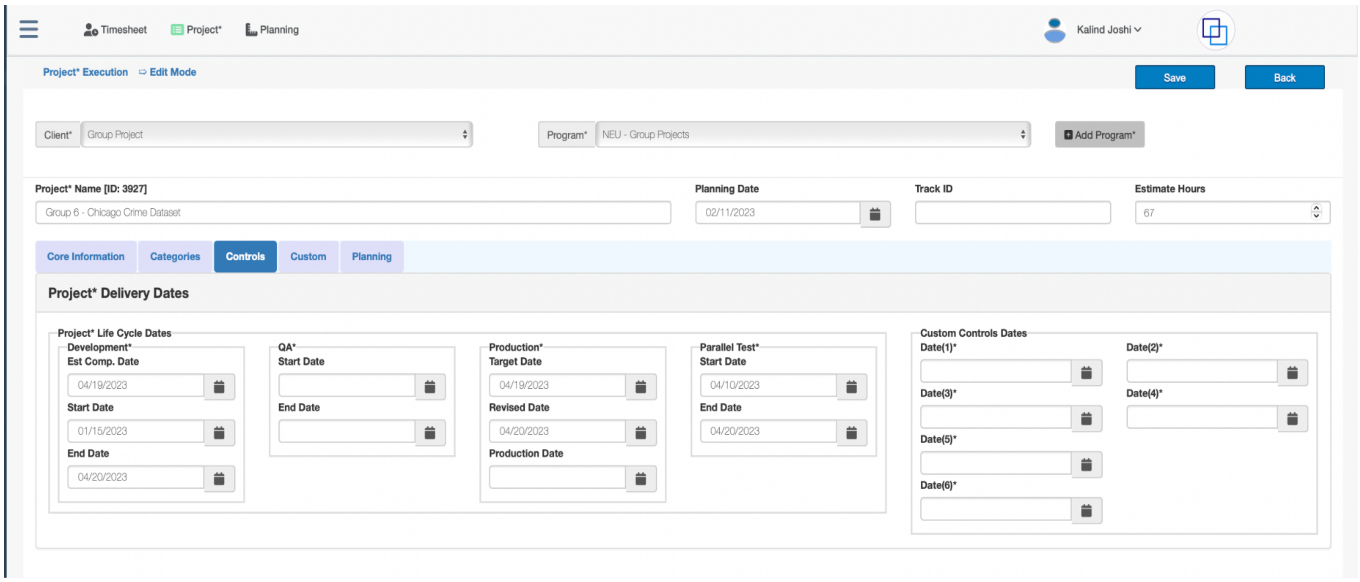
GanttChart | Burndown | New | Back

Copy | Excel | Column visibility | Show 18 entries | Search:

PLC	Seq	Type	Task Name	%Complete	Est Hours	Est HtC	Start Date	End Date	Status	HoursPosted
Planning	2-1		Complete and Load Project Plan		5.00	0.00	02/11/2023	02/25/2023	Complete	4.25
Planning	2-2		Create Velero Project		5.00	0.00	02/11/2023	02/25/2023	Complete	5.00
Planning	2-3		Velero - Add Resource Allocation		5.00	0.00	02/11/2023	02/25/2023	Complete	5.00
Execution	3-1		Define Business Terms		5.00	0.00	02/25/2023	03/04/2023	Complete	5.00
Execution	3-2		Install Neo4J		2.00	0.00	02/25/2023	03/04/2023	Complete	7.00
Execution	3-3		Prepare Development Environment		10.00	0.00	03/04/2023	03/18/2023	Complete	10.00
Execution	3-4		Run Python Profiling		5.00	0.00	03/04/2023	03/18/2023	Complete	0.00
Execution	3-5		Document findings		5.00	0.00	03/18/2023	03/25/2023	Complete	5.00
Execution	3-6		Desgin Database		10.00	0.00	03/18/2023	03/25/2023	Complete	12.00
Execution	3-7		Data Cleansing		10.00	0.00	03/25/2023	04/01/2023	Complete	5.00
Execution	3-8		Test test process		5.00	0.00	03/25/2023	04/01/2023	Complete	0.00
Execution	3-9		Write DB Creation Script		10.00	0.00	04/01/2023	04/08/2023	Complete	28.00
Execution	3-10		Test the script		3.00	0.00	04/01/2023	04/08/2023	Complete	3.00
Execution	3-11		Write Data Quality		10.00	0.00	04/01/2023	04/08/2023	Complete	10.00
Execution	3-12		Test Data Quality & Load Process		10.00	0.00	04/08/2023	04/15/2023	Complete	10.00
Execution	3-13		Install BI Tool		2.00	0.00	04/08/2023	04/15/2023	Complete	3.50
Execution	3-14		Test BI - DB Connectivity		10.00	10.00	04/08/2023	04/15/2023	Inprocess	8.00
Execution	3-15		Write the Analytic Dashboard		10.00	10.00	04/15/2023	04/20/2023	Inprocess	4.00

Showing 1 to 018 of 19 entries | Previous | 1 | 2 | Next

- Dates planned for progress and completion:



- Timesheet hours posted:

Date / Project*	activity	Hours
Wednesday, 02/22/2023	Daily Total	5.00
Group 6 - Chicago Crime Dataset	Project Planning	5.0
Thursday, 02/23/2023	Daily Total	5.00
Group 6 - Chicago Crime Dataset	Project Planning	5.0
Monday, 02/27/2023	Daily Total	5.00
Group 6 - Chicago Crime Dataset	Project Planning	5.0
Wednesday, 03/01/2023	Daily Total	5.00
Group 6 - Chicago Crime Dataset	Project Planning	5.0
Thursday, 03/02/2023	Daily Total	2.00
Group 6 - Chicago Crime Dataset	Project Planning	2.0
Wednesday, 03/15/2023	Daily Total	1.50
Group 6 - Chicago Crime Dataset	Code Development	1.5
Sunday, 03/19/2023	Daily Total	3.00
Group 6 - Chicago Crime Dataset	Code Development	3.0
Saturday, 03/25/2023	Daily Total	2.00
Group 6 - Chicago Crime Dataset	Code Development	2.0
Thursday, 04/06/2023	Daily Total	4.00
Group 6 - Chicago Crime Dataset	Project Planning	4.0
Wednesday, 04/12/2023	Daily Total	4.00
Group 6 - Chicago Crime Dataset	Project Planning	4.0
Friday, 04/14/2023	Daily Total	3.00
Group 6 - Chicago Crime Dataset	Code Development	3.0
Wednesday, 04/19/2023	Daily Total	5.00
Group 6 - Chicago Crime Dataset	Code Development	5.0
Thursday, 04/20/2023	Daily Total	4.00
Group 6 - Chicago Crime Dataset	Data Analysis	4.0
Total Hours Posted:		48.50

3.5. Data Validation and Data Visualization

- Who will be using this dashboard and how do they benefit from this dashboard?

The visualization dashboard on Tableau for Chicago crimes can provide valuable insights to a wide range of users, including law enforcement agencies, city officials, researchers, analysts, and the general public, to make informed decisions and take appropriate actions to address crime-related issues effectively.

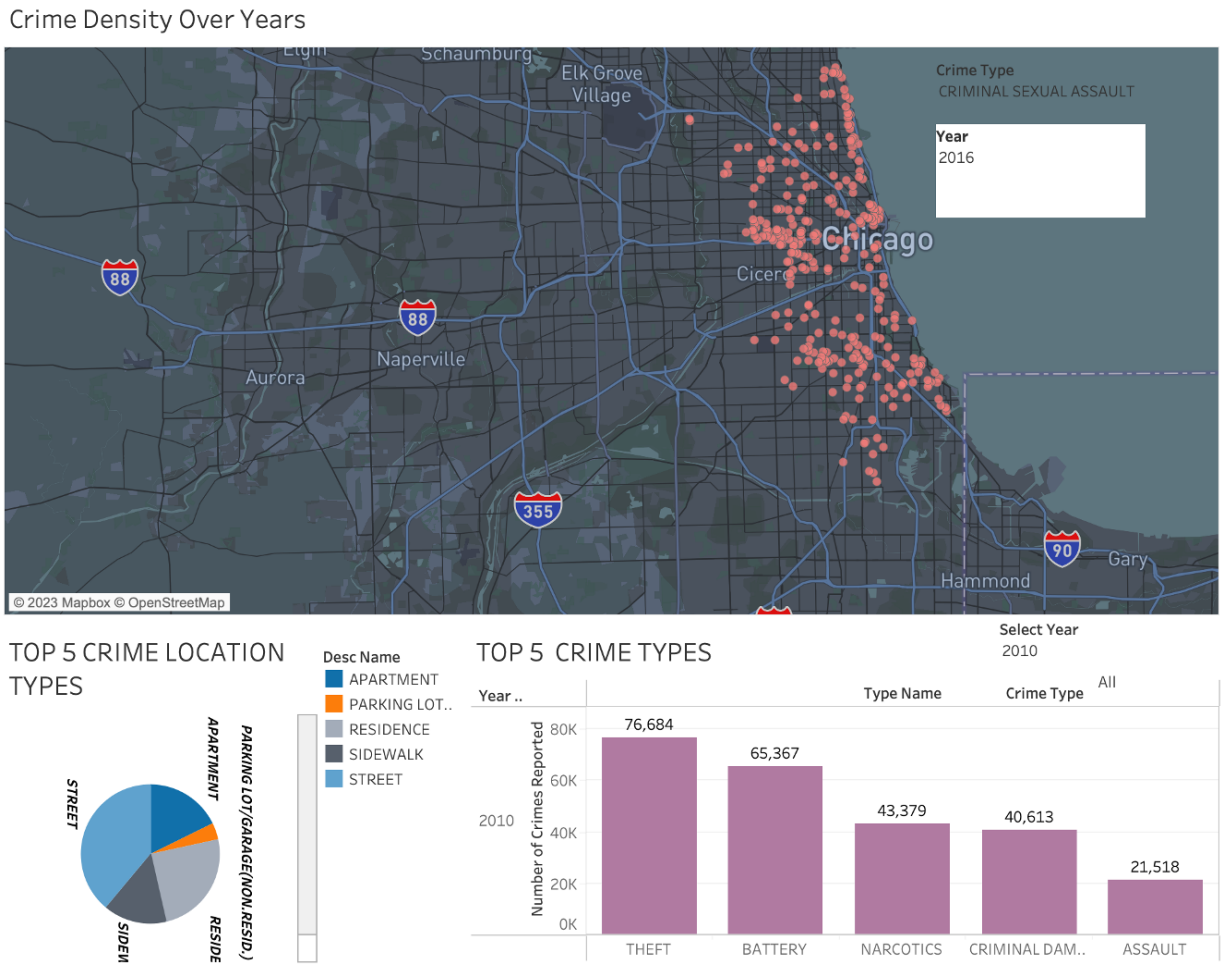
- What value would be generated using this dashboard?

The visualization dashboard on Tableau for Chicago crimes can generate value by providing stakeholders with data-driven insights, enhancing decision-making, enabling proactive crime prevention, optimizing resource allocation, and promoting transparency and accountability. These benefits can contribute to more effective crime management strategies, improved public safety, and stronger relationships between law enforcement agencies and communities.

- Did you generate any new dimensions?

Yes we created a new dimension named 'UniqueID'

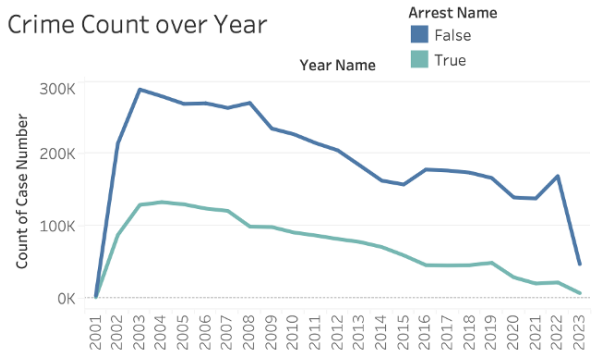
Data Visualization:-

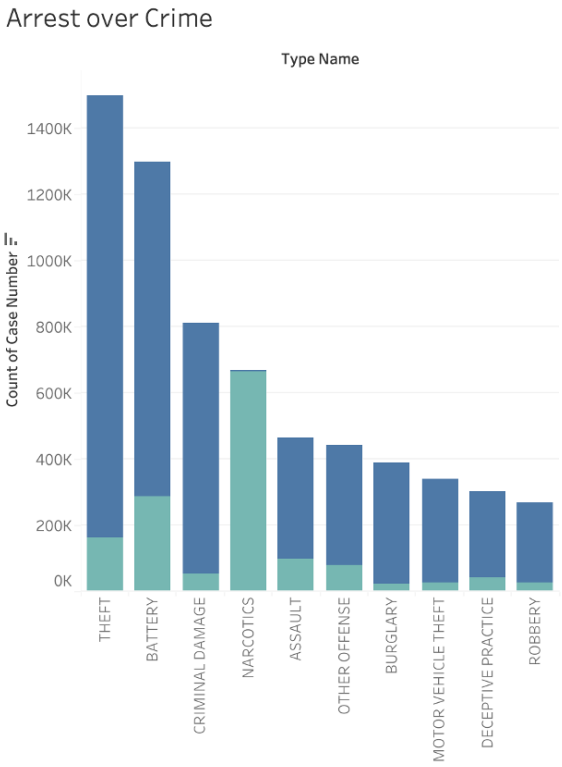


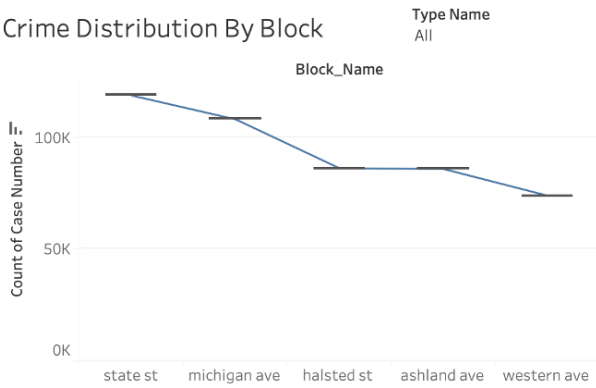
The primary visualization on the dashboard is a Map that utilizes latitude and longitude as the axes. This map displays the geographical locations where crimes have occurred, categorized by years. It allows users to visually assess the areas with the highest crime rates and gain insights into the density of crimes in different regions.

The pie chart displays the top 5 crime location types for each year. Upon analysis, it is evident that in 2016, the majority of crimes occurred on streets, followed by incidents that took place in residential areas.

The bar chart presented displays the top five crime types reported annually, ranked in descending order. The most frequently reported crime type is Theft, followed by Battery.







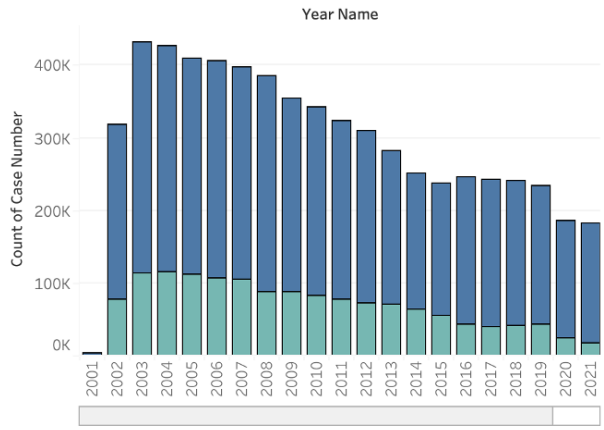
The line graph in the top left corner of the dashboard displays a trend of crimes committed over the years, as well as the number of arrests made in relation to those crimes. Notably, there is a noticeable spike in cases registered from 2001 to 2002. However, there is also a significant disparity between the number of cases registered and the arrests made, indicating that not all crimes have resulted in arrests.

The second graph in the report is a bar graph that presents data on cases registered and arrests made, categorized by crime type. From the graph, we can observe that the category of Narcotics has the highest ratio of cases registered to arrests made, whereas Robbery and Burglary show lower ratios compared to other crime types.

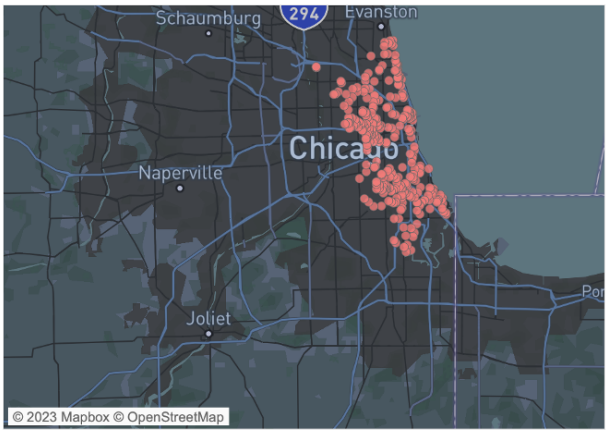
The third graph is a line graph that provides insights on the number of arrests in different blocks by name. It allows users to analyze and compare the arrests data for various blocks, identifying which block has the highest and lowest number of arrests among others in between. According

to the graph, State Street has the highest number of registered cases, while Western Ave has the lowest.

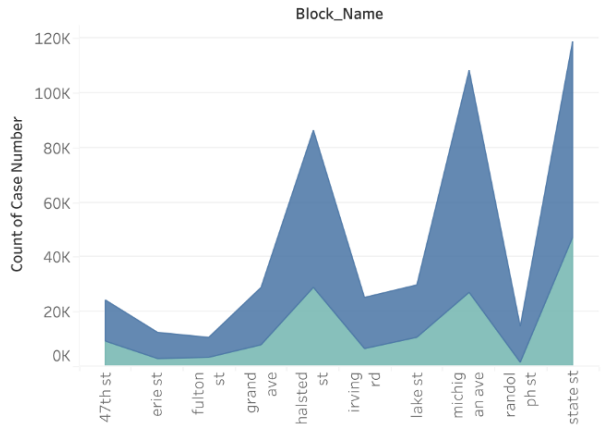
Number of Arrests over the years



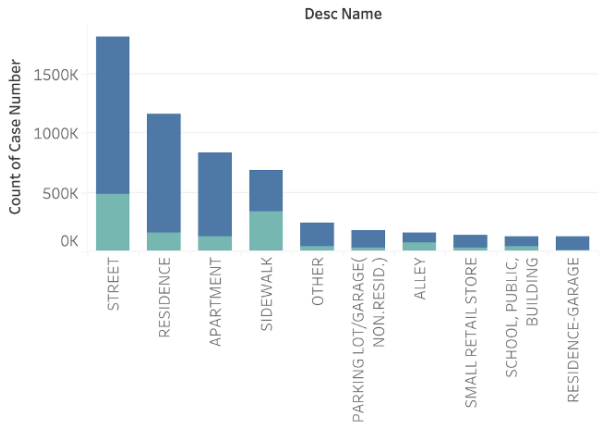
Crime Density Over Years



Blocks with Maximum Arrests



Locations Types with Maximum Arrests



The first bar graph illustrates the progression of crime rates over the years, showing a consistent downward trend in both the number of crimes committed and the arrest rates.

This indicates a potential decrease in overall crime in the given area.

The map provides valuable geographical insights, enabling users to access crime data for various localities in Chicago. This allows for a better understanding of the distribution of crime across different neighborhoods or areas in the city.

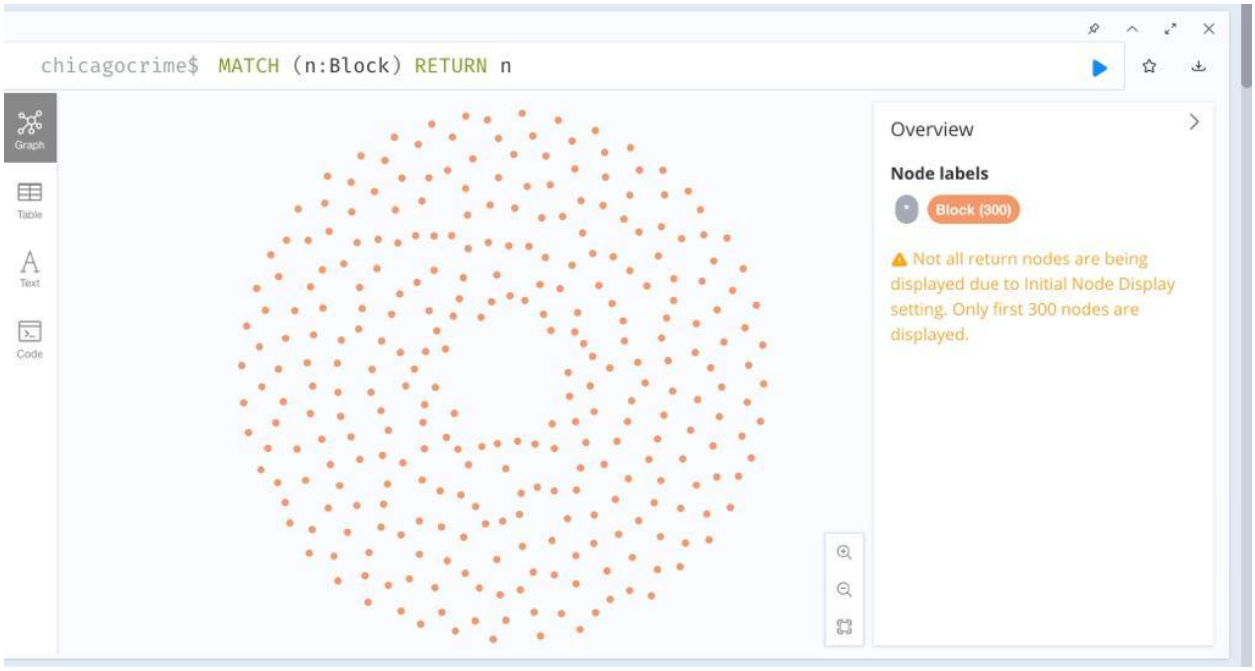
The area graph presents a visual representation of arrest rates at the block level, offering insights into how arrest rates vary across different blocks or areas within the

locality. This information can help identify specific areas with higher or lower arrest rates, potentially revealing patterns or trends in law enforcement activities.

Lastly, the final bar graph displays the types of locations with the highest crime and arrest rates. This information can provide insights into the specific types of locations that are more prone to crime and where law enforcement activities may be focused.

3.6. System Integration and User Acceptance Testing

The dataset we processed in python has been successfully integrated into neo4j as shown below.



```
df['Block']
```

```
0              082XX S COLES AVE
1                004XX W 42ND PL
2            025XX N KIMBALL AVE
3             045XX W MONTANA ST
4             111XX S NORMAL AVE
                    ...
1456709          026XX W 23RD PL
1456710      073XX S HARVARD AVE
1456711          024XX W 63RD ST
1456712     082XX S EXCHANGE AVE
1456713          001XX E 75TH ST
Name: Block, Length: 7941282, dtype: object
```

3.7. Challenges Encountered

1. Problem: While working with the Chicago crime dataset, the "Location" column was encountered. This column contains values in the format of (latitude, longitude) and is not compatible with Neo4j due to the long floating values, which occupy a lot of time. As a result, the query gets terminated.

   Solution: To resolve the problem of loading the "Location" column in Neo4j, a new column named "UniqueID" was created in the pandas dataframe. The purpose of this column is to contain a unique value for every combination of (latitude, longitude). By doing this, while loading data in Neo4j, this data is loaded by taking the "UniqueID" column into consideration and keeping (latitude, longitude) as their property.

   By using this approach, the "Location" column data was loaded in Neo4j without any errors and in a much shorter time. This solution provides a workaround to the problem of loading long floating values into Neo4j.

   Conclusion: In conclusion, the problem of loading the "Location" column data in Neo4j was addressed by creating a new column named "UniqueID" in the pandas dataframe. This approach allowed the data to be loaded in Neo4j without any errors and in a much shorter time. By using this solution, the dataset can be efficiently loaded in Neo4j, and the data can be analyzed further.

In [25]:
```
# create a new column 'UniqueID' and assign a unique ID to each (Latitude, Longitude) pair
df3['UniqueID'] = df3['Location'].apply(lambda x: hash(x))

# print the dataframe to verify the results
df3.head()
```

Out[25]:

nary Type	Description	Location Description	Arrest	...	District	Ward	Community Area	FBI Code	X Coordinate	Y Coordinate	Year	Updated On	Location	UniqueID
CIDE	FIRST DEGREE MURDER	ALLEY	True	...	3.0	6.0	69.0	01A	1178207.0	1855308.0	2008	08/17/2015 03:03:40 PM	(41.758275857, -87.622451031)	386106639535259970
CIDE	FIRST DEGREE MURDER	STREET	True	...	15.0	24.0	25.0	01A	1144200.0	1895857.0	2008	08/17/2015 03:03:40 PM	(41.87025207, -87.746069362)	-623691163198600792
CIDE	FIRST DEGREE MURDER	PARK PROPERTY	False	...	8.0	18.0	66.0	01A	1157314.0	1859778.0	2008	08/17/2015 03:03:40 PM	(41.770990476, -87.698901469)	-4740121907779104319
CIDE	FIRST DEGREE MURDER	RESTAURANT	False	...	15.0	37.0	25.0	01A	1141065.0	1904824.0	2008	08/17/2015 03:03:40 PM	(41.894916924, -87.757358147)	7580557972628098219
CIDE	FIRST DEGREE MURDER	GARAGE	False	...	10.0	22.0	30.0	01A	1154123.0	1886297.0	2008	08/17/2015 03:03:40 PM	(41.843826272, -87.709893465)	7798137727751833469

2. Problem: The "Case Number" column in the Chicago crime dataset contained 7 million rows, which were too large to process and resulted in memory errors when trying to load them into Neo4j. This posed a significant challenge as the data in this column was crucial for creating relationships between nodes.

Solution: After conducting extensive research, we found a solution to load the "Case Number" column into Neo4j using the apoc.periodic.iterate method. This method enabled us to load the data in batches of 2000 rows, which made it possible to load the entire dataset without causing memory errors.

Furthermore, the "Case Number" column was used to create relationships with every other node in the dataset, resulting in over 70 million combinations of relationships in total. To ensure that the loading process did not result in memory errors, we had to increase the heap space from the neo4j.config file from 1GB to 4GB.

The apoc.periodic.iterate method made it possible to load the "Case Number" column without causing memory errors, and also enabled the creation of the necessary relationships between nodes. This approach made it possible to analyze the dataset in Neo4j and gain insights into the patterns of crime in Chicago.

Conclusion: In conclusion, the challenge of loading the "Case Number" column in Neo4j was addressed by using the apoc.periodic.iterate method. This method allowed the data to be loaded in batches of 2000 rows, which made it possible to load the entire dataset without causing memory errors. The "Case Number" column was used to create relationships between nodes, resulting in over 70 million combinations of relationships in total. By using this approach, the dataset was efficiently loaded in Neo4j, and the data could be analyzed further to gain insights into the patterns of crime in Chicago.

3.8. End User Instructions

Our dataset, titled 'Chicago Crime', contains approximately 7 million rows of data. This dataset includes information such as the Primary Type, Location Description, community area ETC corresponding to the crime committed, the type of crime, whether an arrest was made in connection with the crime.

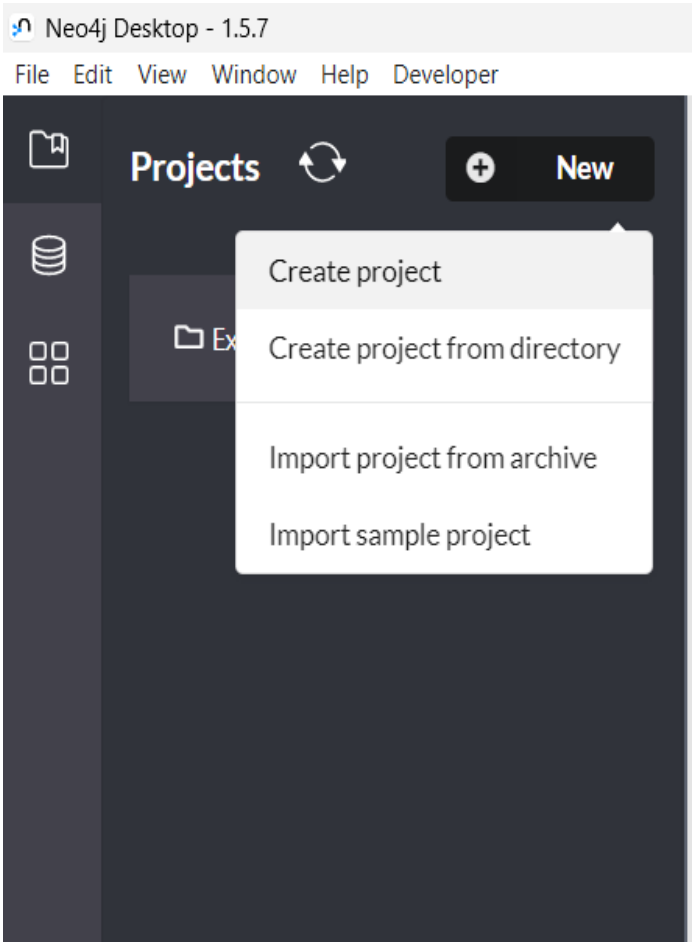
The business terms of the dataset is as follows:

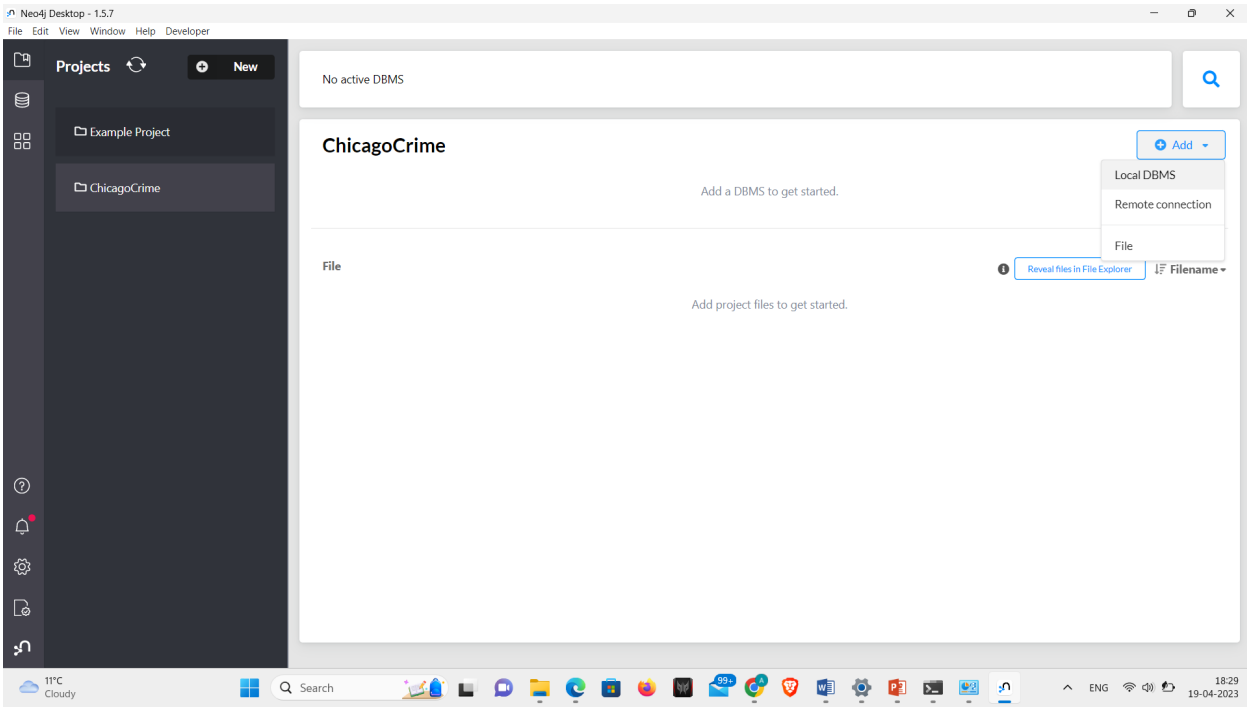
1. **ID -** Unique identifier for the record.
2. **Case Number -** The Chicago Police Department RD Number (Records Division Number), which is unique to the incident.
3. **Date -** Date when the incident occurred. This is sometimes the best estimate.
4. **Block -** The partially redacted address where 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 (small region) where the incident occurred.
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.
15. **FBI Code -** Indicates the crime classification as outlined in the FBI's National Incident- Based Reporting System.
16. **Year -** Year the incident occurred.
17. **Location -** The location where the incident occurred.

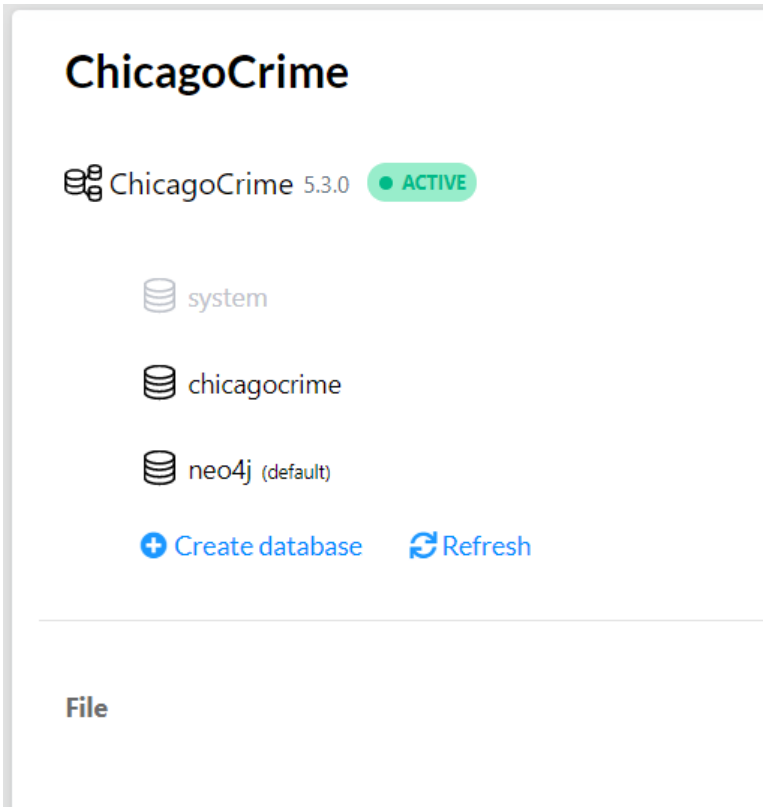
I. Neo4j Desktop:

For our project, we obtained the Neo4j Database software from the official website and utilized the generated key to activate our account.

Next, we conducted testing on the provided movie database and subsequently set up a new DBMS with a database named 'chicagocrime' within it.







To begin working with the chicagocrime database, initiate a connection by clicking on the start button located on the right side of the interface. This will

allow us to establish a connection and start utilizing the database for our project..

II. Neo4j Browser:

After confirming that the connection is active and the database is running, access the 'Neo4j Browser' from the 'Graph Apps' option in the left vertical menu. This will allow you to work on the database by creating nodes and relationships.

Loading data into Neo4j:

a. Start the ChicagoCrime project by pressing start
b. Open cypher queries, copy the code
c. Once the database is active, open the database, paste the query in front of the $ sign and execute it.
d. Wait for all data to be loaded into Neo4j.

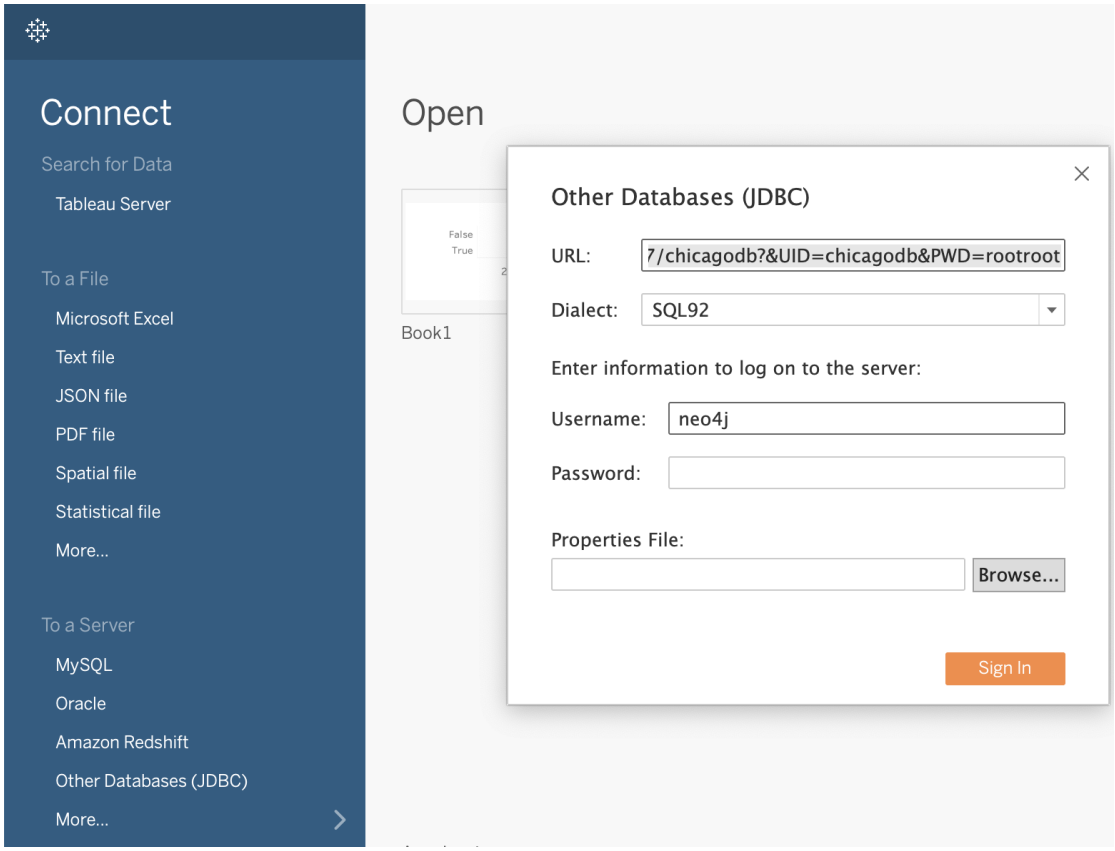
As part of our data integration process, we utilized the Neo4j Browser to create unique nodes and constraints to ensure data integrity. Subsequently, we established relationships between every node to finalize the graph database model, and successfully uploaded data into it.

Finally, we have 13 nodes and 78 Million relationships.

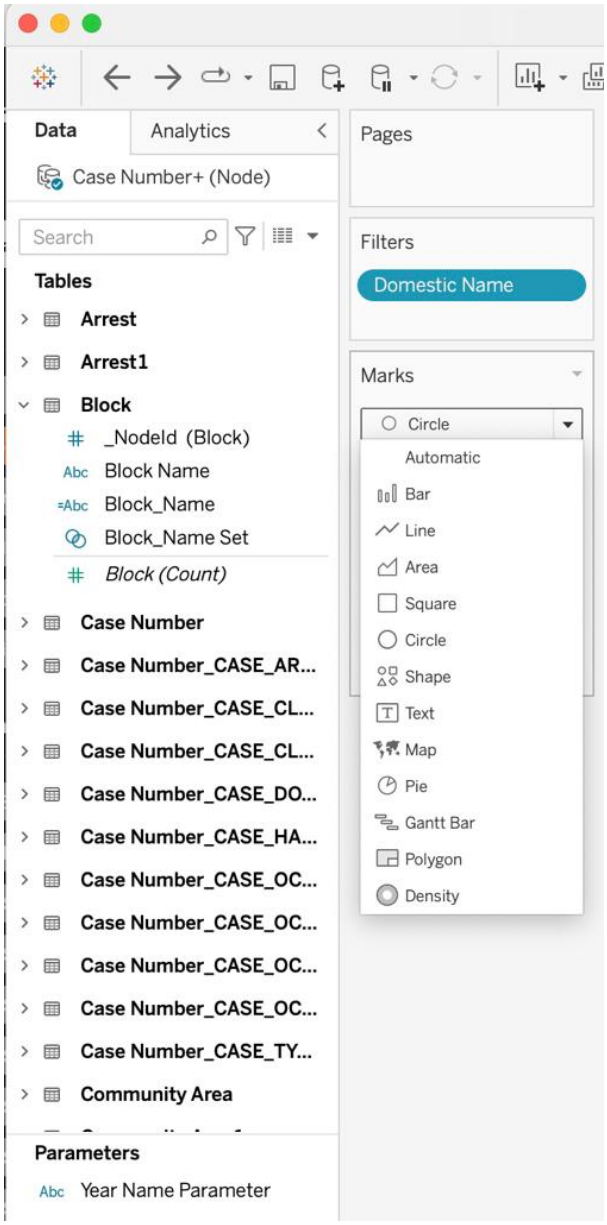
III. Neo4j + Tableau connectivity:

Subsequent to uploading the data and establishing the necessary nodes and relationships in Neo4j, we successfully established a connection between Tableau and Neo4j utilizing the JDBC connector.

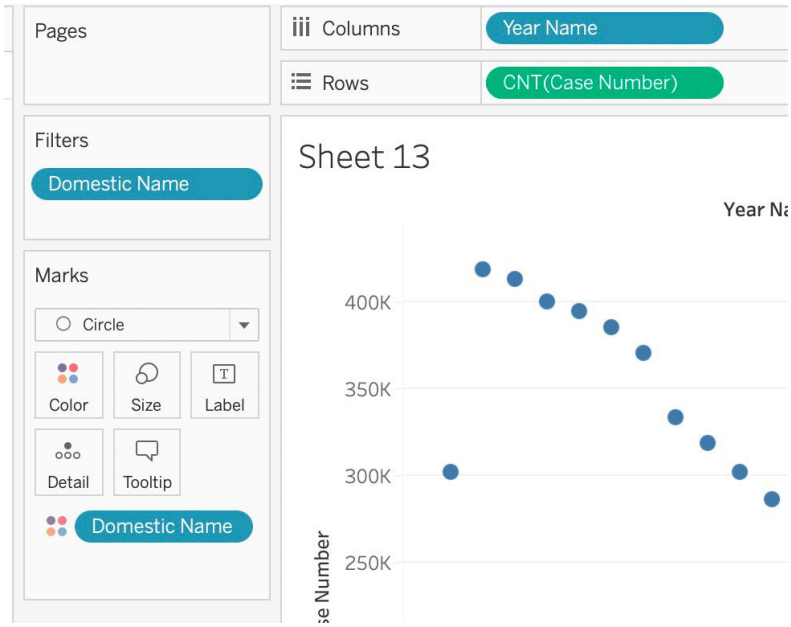
In order to connect Tableau with Neo4j, we downloaded the JDBC connector and placed it in the Tableau folder. Next, we opened Tableau and selected the "connect to a server using JDBC connector" option. We then entered the localhost URL along with the DBMS name and password to establish a connection with Neo4j.


Connect
Search for Data
Tableau Server
To a File
Microsoft Excel
Text file
JSON file
PDF file
Spatial file
Statistical file
More...
To a Server
MySQL
Oracle
Amazon Redshift
Other Databases (JDBC)
More...
Open
False
True
Book1
Other Databases (JDBC)
URL:
7/chicagodb?&UID=chicagodb&PWD=rootroot
Dialect:
SQL92
Enter information to log on to the server:
Username:
neo4j
Password:
Properties File:
Browse...
Sign In

IV. Tableau Visualisation:



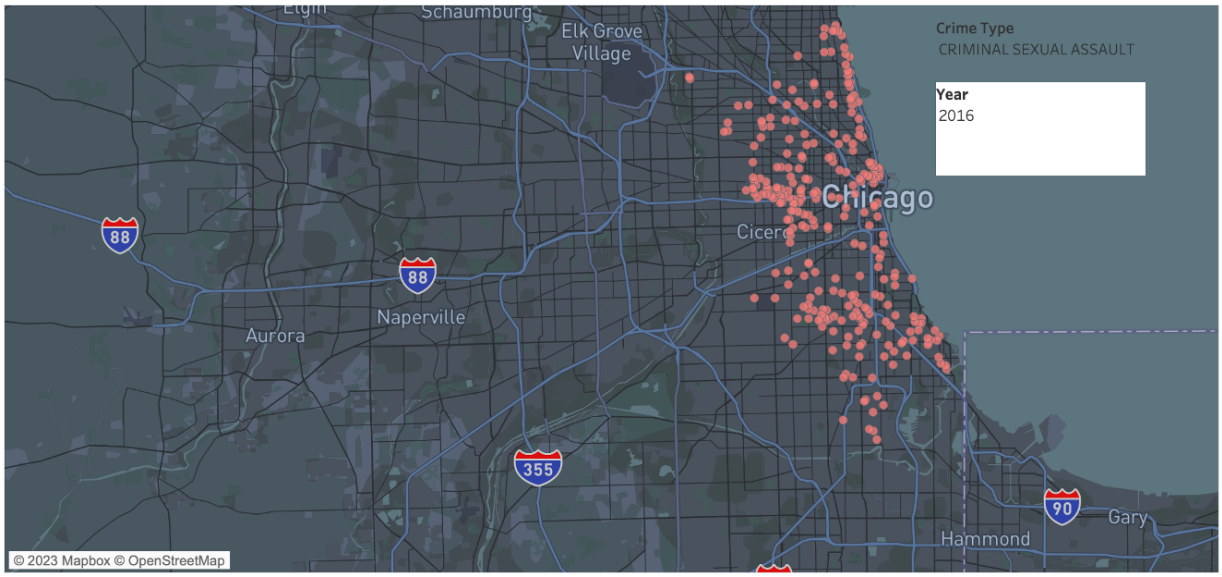
To create a new visualization, select the row and column from the list in the image above which are the nodes and relations in our graph. The type of graph can be selected from the dropdown on the right side.



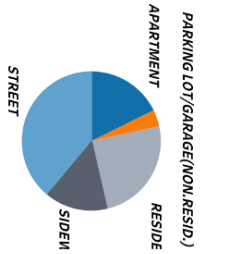
Then, drag them to the column and rows area as shown above to create a visualization.

Dashboard 1:-

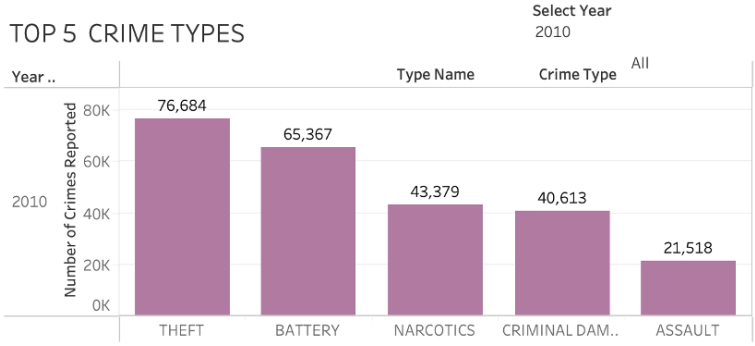
Crime Density Over Years



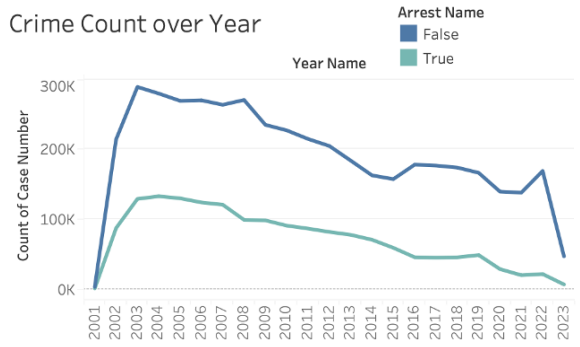
TOP 5 CRIME LOCATION TYPES

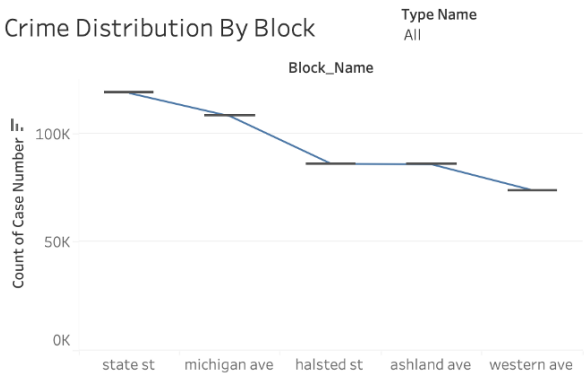


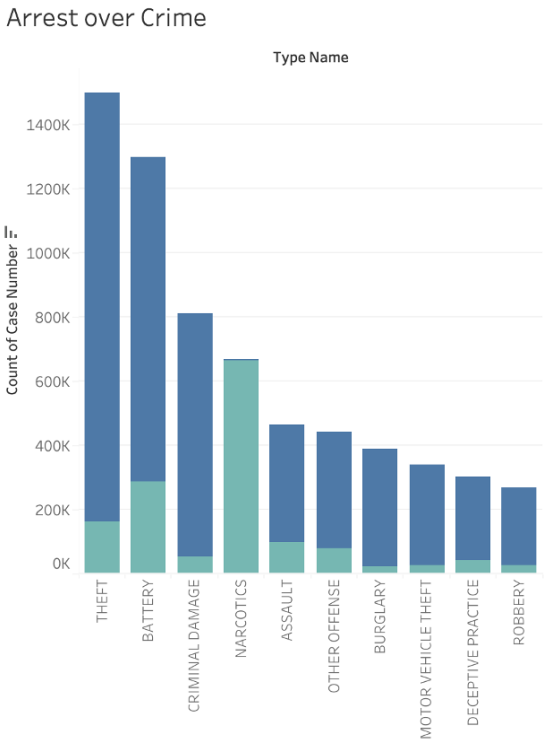
TOP 5 CRIME TYPES



Dashboard 2:-

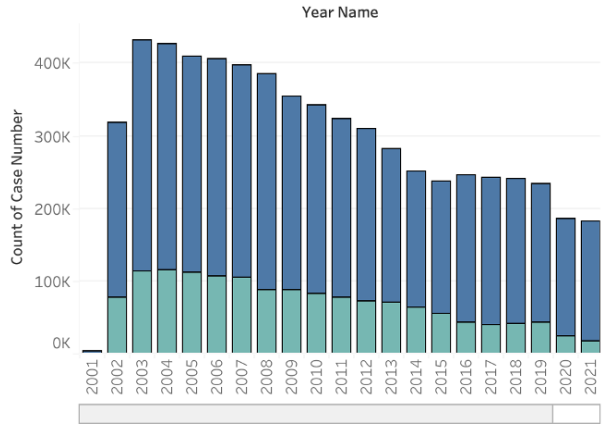

Crime Count over Year
Arrest Name
False
True
Year Name
Count of Case Number
300K
200K
100K
0K
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023


Crime Distribution By Block
Type Name
All
Block_Name
Count of Case Number
100K
50K
0K
state st
michigan ave
halsted st
ashland ave
western ave

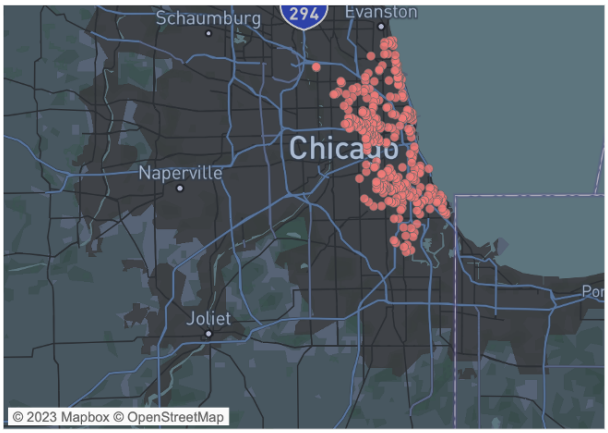

Arrest over Crime
Type Name
Count of Case Number
1400K
1200K
1000K
800K
600K
400K
200K
0K
THEFT
BATTERY
CRIMINAL DAMAGE
NARCOTICS
ASSAULT
OTHER OFFENSE
BURGLARY
MOTOR VEHICLE THEFT
DECEPTIVE PRACTICE
ROBBERY

Dashboard 3:

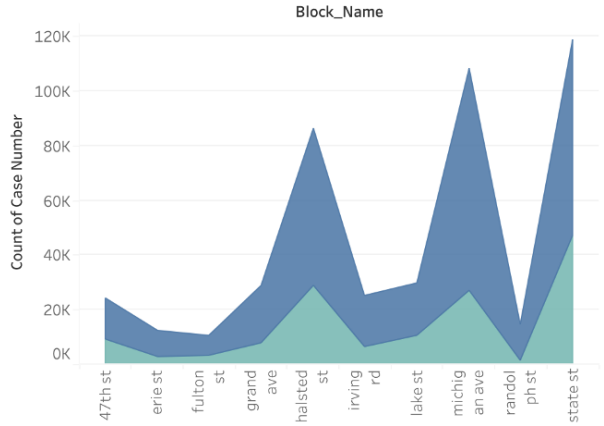
Number of Arrests over the years



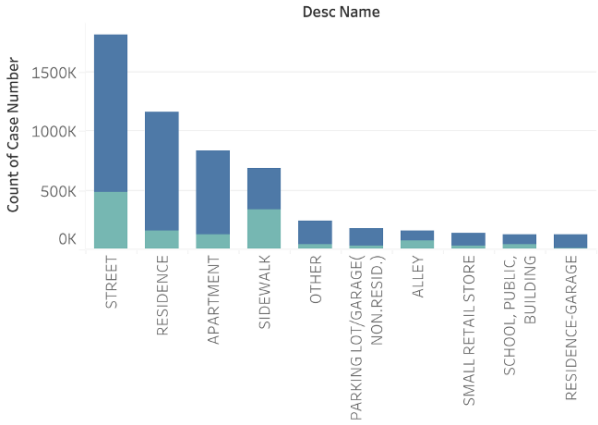
Crime Density Over Years



Blocks with Maximum Arrests



Locations Types with Maximum Arrests



Using Dashboard 1, the user can analyze which crime types are most committed and which locations are most susceptible to these particular crimes. Users can drill down by selecting a specific crime on the map and look at the respective crime type and location where it has been committed.

Dashboard 2 can be used to identify the trend of crime rate over the years in Chicago, and compare it with block wise crime occurrence.

Dashboard 3 can be used to gain insights about arrest rate in different localities, and compare crime density and arrest rate over the years.