Big Data Management

Implementing Big data Concepts for Visualizing Chicago Crime Data & Game of Thrones(GOT)

Divya Dasaraju, Neeharika Mallineni, Siddharth Chittoor

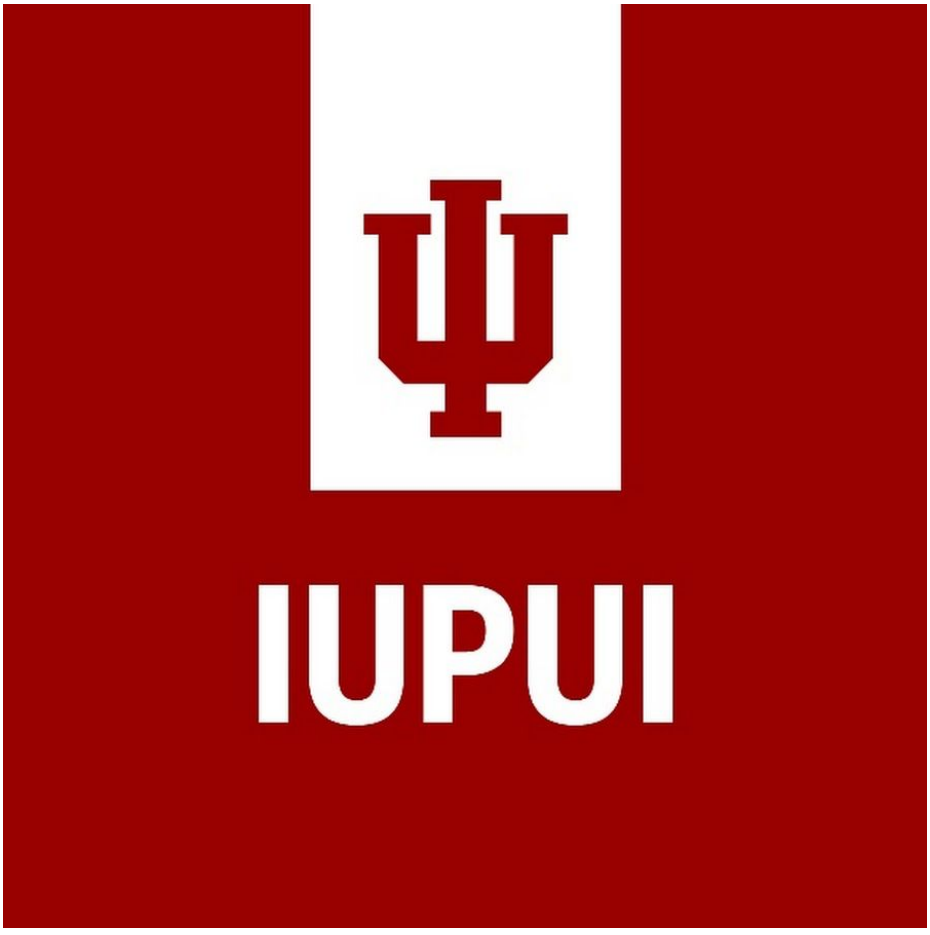


Table of Contents

Abstract :

The main aim of our project is to understand and implement the Big data technologies on Chicago Crime Dataset for Relational Part and GOT for Non-Relational Part. For relational part, MYSQL and Hive is used as query language to extract the answers from the dataset. For non relational database we used Neo4j with Apoc,GraphQL andGraph Algorithms plugin.

Introduction:

Chicago is one of the busiest and most populated cities in the United States, which is a reason for crimes happening in the city. Crimes in Chicago is an interesting topic to work with Big data tools and Data Science projects. Chicago crime dataset is obtained from Kaggle website. The main objective of this project is to visualize the dataset according to the query results using Tableau.

Game of Thrones dataset is complex as there are a lot of people with a lot of relationships between them. We chose this because we thought it would be a good way to represent the data in a graph format. In this dataset we found out how many people have wifes, the shortest path between two people among other things, maximum and average number of allies for a person.

Data Processing:

The Chicago crime dataset is obtained from Kaggle website which is a huge repository for datasets. Initially it had 22 feature attributes or columns in the dataset. After processing the dataset, there are some feature attributes that are not useful for the implementation of Big data Pipeline for Visualization, so we removed the columns Block(it is not that useful since location gives better description about the area), Domestic, column Primary type has crimes related to Domestic violence and Case number, community area etc. For final model, 13 feature attributes are listed for the data pipeline model. Efficiency of the data pipeline increases when there are attributes that contributes most to the model than the feature attributes that are not used any more. For half queries , Hive is used as query language and executed with the help of AWS S3 and Aws EMR and for remaining queries , Crime dataset is imported to MYSQL to work with the queries.

For NonSQL part we did not have to perform any pre processing as the data we found was clean.

I . Relational Database - Chicago Crime Dataset

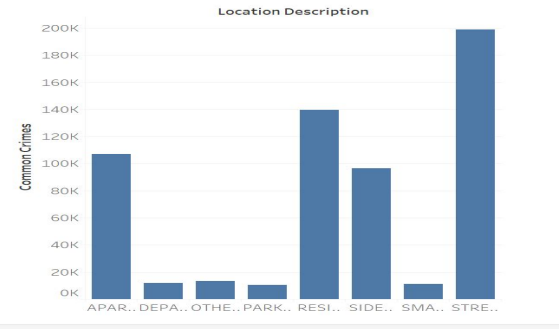
Once the data is uploaded successfully, now analyze insights about the data. Chicago crime dataset is all about different types of crimes happening in and around the city.

Q1: Most Common types Crimes:

In the first query we can see the distinct types of crimes and their locations which occurred mostly in the chicago.

```
13 •  select Primary_Type, Location_Description , count(*) as common_crimes from chicago_crimes
14    group by Primary_Type
15    order by common_crimes DESC
16    Limit 25;
```

Primary_Type	Location_Description	common_crimes
THEFT	RESIDENCE	31233
BATTERY	APARTMENT	27631
NARCOTICS	STREET	17616
CRIMINAL DAMAGE	APARTMENT	15853
ASSAULT	SIDEWALK	9613
OTHER OFFENSE	STREET	9403
BURGLARY	RESIDENCE-GARAGE	8215
MOTOR VEHICLE THEFT	STREET	6784
DECEPTIVE PRACTICE	RESIDENCE	6533
ROBBERY	SIDEWALK	5232
CRIMINAL TRESPASS	DRUG STORE	4039
WEAPONS VIOLATION	VEHICLE NON-COM...	1932



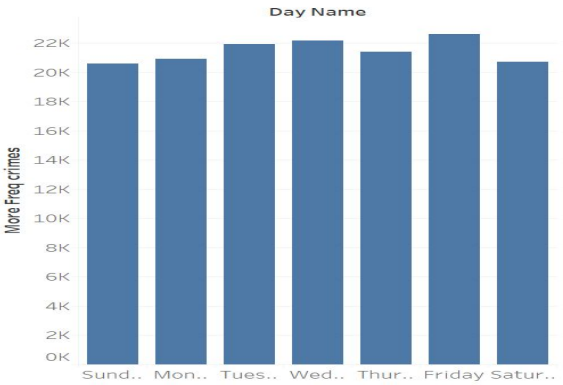
We can see from the visualization, that crime type Theft has occurred in the Residency areas with more count, which is then followed by crime type Battery in Apartments and Narcotics in the street areas.

Q2. Frequently happening crimes, list accordingly with weekends/weekdays

Crimes happening in the days like Fridays and Wednesdays are more. Crimes happening on Saturdays and Sundays are comparatively less.

```
3 •  create TEMPORARY TABLE  temp as (select dayname(crime_date)
4     as day_name ,crime_date from chicago_crimes);
5 •  select  day_name, count(day_name) as more_freq from temp
6     group by day_name order by more_freq desc limit 7;
```

day_name	more_freq
Friday	22616
Wednesday	22161
Tuesday	21927
Thursday	21421
Monday	20906
Saturday	20730
Sunday	20605



We can see from the visualization that most number of crimes happened on friday compared to weekends.

Q3: Top location crimes

Most of the crimes that happens primarily depends on the location. Query is executed to find the top locations where the crimes are happening frequently.

```
6     select primary_type , location_description, count(*) as more_crimes
7     from chicago_crimes group by primary_type, location_description
8     order by more_crimes desc
9     limit 25;
```

primary_type	location_description	more_crimes
THEFT	STREET	7507
BATTERY	APARTMENT	7153
NARCOTICS	SIDEWALK	5756
CRIMINAL DAMAGE	STREET	5714
NARCOTICS	STREET	5643
MOTOR VEHICLE THEFT	STREET	5503
BATTERY	RESIDENCE	5402
BATTERY	SIDEWALK	4019
OTHER OFFENSE	RESIDENCE	3812
BATTERY	STREET	3536
THEFT	RESIDENCE	3006

By the above query, we can see that locations like streets, apartments , residence areas and sidewalk are the more prone areas to the crimes happening in the city.

Q4: Morning and Night crimes

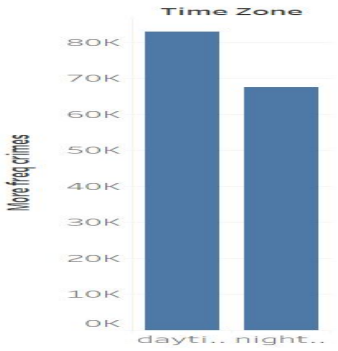
Following MYSQL query is executed successfully showing respective Day time and Morning crimes in the city of Chicago. Total crimes that happens in the day time are greater than that happened in the night times as shown in the visualization.

```
create temporary table temp1 select time(crime_date) as crime_time ,
crime_date from chicago_crimes;

create temporary table temp2 select  case
when crime_time between '06:00:00' and '18:00:00' then 'daytime'
else 'nighttime'
end  as time_zone from temp1;
select time_zone, count(time_zone) as more_freq from temp2
group by time_zone order by more_freq desc limit 2;
```

Filter Rows: | Export: | Wrap Cell Content:

time_zone	more_freq
daytime	82896
nighttime	67470



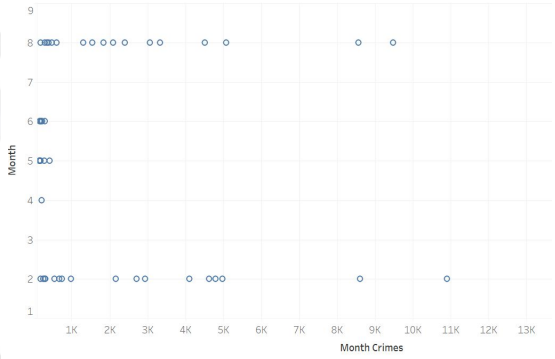
Q5: In which months crimes occurred mostly

Crimes happening in different months are listed below with a successfully executed query . Highest crimes happened in February month with crime type Theft, followed my Battery and Narcotics in the same month.

```
select Primary_Type, month(updated_on) , count(updated_on) as month_crimes
 from chicago_crimes where month(updated_on) between 1 and 12
group by Primary_Type , updated_on
order by month_crimes desc limit 50;
```

Filter Rows: | Export: | Wrap Cell Content:

Primary_Type	month(updated_on)	month_crimes
THEFT	2	16850
BATTERY	2	14382
NARCOTICS	2	10910
THEFT	8	9487
CRIMINAL DAMAGE	2	8615
BATTERY	8	8560
NARCOTICS	8	5080
BURGLARY	2	4983
ASSAULT	2	4790
OTHER OFFENSE	2	4677

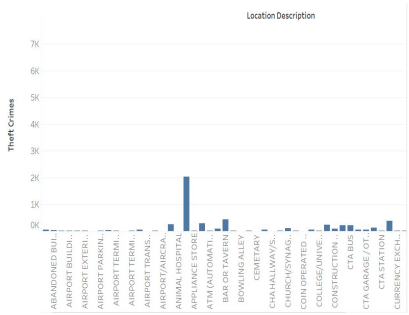


Q6: Frequently occurred theft crimes in Chicago

In the Chicago city, Theft is one of the serious problems. Theft crime type is occurred mostly in Streets of Chicago, followed by residency areas and apartments. Least updated thefts are in federal building , forest preserve areas.

```
select Location_Description, count(*) as theft_crimes from chicago_crimes where
Primary_Type = 'Theft' group by Location_Description
order by theft_crimes desc ;
```

Location_Description	theft_crimes
STREET	7507
RESIDENCE	3006
APARTMENT	2031
DEPARTMENT STORE	1794
OTHER	1787
SMALL RETAIL STORE	1615
PARKING LOT/GARAGE(NON.RESID.)	1527
GROCERY FOOD STORE	1247
SIDEWALK	1038
RESTAURANT	991
RESIDENTIAL YARD (FRONT/BACK)	866

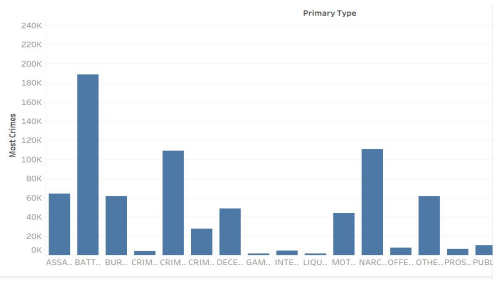


Q7: Most reported Crimes

In the city of Chicago, most reported crime is the Theft and major number of counts. And we have seen in the precious queries that Theft is mostly happened on apartments, streets and residency areas.

```
select primary_type , Updated_On, count(*) as most_crimes
from chicago_crimes group by primary_type, Updated_On
order by most_crimes desc
limit 20;
```

primary_type	Updated_On	most_crimes
THEFT	2016/2/4 6:33	16850
BATTERY	2016/2/4 6:33	14382
NARCOTICS	2016/2/4 6:33	10910
THEFT	2015/8/17 15:03	9487
CRIMINAL DAMAGE	2016/2/4 6:33	8615
BATTERY	2015/8/17 15:03	8560
NARCOTICS	2015/8/17 15:03	5080
BURGLARY	2016/2/4 6:33	4983
ASSAULT	2016/2/4 6:33	4790
OTHER OFFENSE	2016/2/4 6:33	4622
CRIMINAL DAMAGE	2015/8/17 15:03	4517

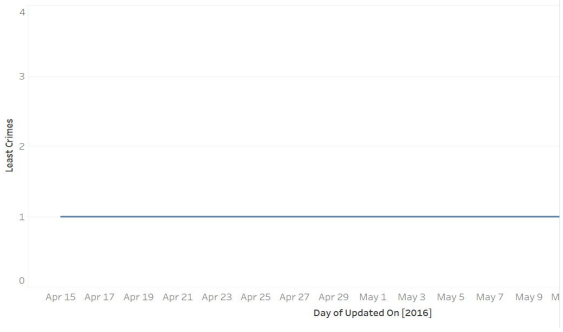


Q8: Least reported Crimes

Likewise, crimes that are reported and updated least are Criminal Trespass followed by criminal damage and so on. The following query is illustrated and executed successfully.

```
select primary_type , Updated_On, count(*) as least_crimes from chicago_crimes
group by primary_type, Updated_On
order by least_crimes asc
limit 10;
```

primary_type	Updated_On	least_crimes
CRIMINAL TRESPASS	2016/5/10 15:56	1
CRIMINAL DAMAGE	2016/5/11 15:48	1
ARSON	2016/5/13 15:49	1
ARSON	2016/5/15 15:47	1
KIDNAPPING	2016/4/15 9:20	1
OFFENSE INVOLVING CHILDREN	2016/5/17 15:46	1
NARCOTICS	2016/5/16 15:48	1
HOMICIDE	2016/5/16 15:48	1
NON-CRIMINAL	2016/5/16 15:50	1
SEX OFFENSE	2016/5/17 15:46	1

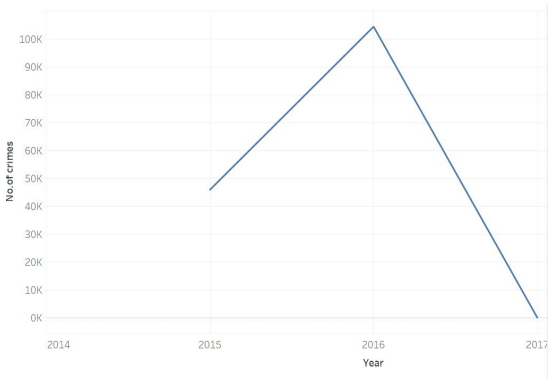


Q9. In which year crimes happened mostly and name the Crimes

The dataset used for this project contains from 2015 to 2017. Out of these years, 2016 has the most recorded and updated crimes in the city, below query is illustrated.

```
select Primary_Type, YEAR(Updated_On) , count(Updated_On) as more_c from
chicago_crimes where year(Updated_On) between 2015 and 2017
group by Primary_Type , Updated_On
order by more_c desc ;
```

Primary_Type	YEAR(Updated_On)	more_c
THEFT	2016	16850
BATTERY	2016	14382
NARCOTICS	2016	10910
THEFT	2015	9487
CRIMINAL DAMAGE	2016	8615
BATTERY	2015	8560
NARCOTICS	2015	5080
BURGLARY	2016	4983
ASSAULT	2016	4790
OTHER OFFENSE	2016	4622
CRIMINAL DAMAGE	2015	4517

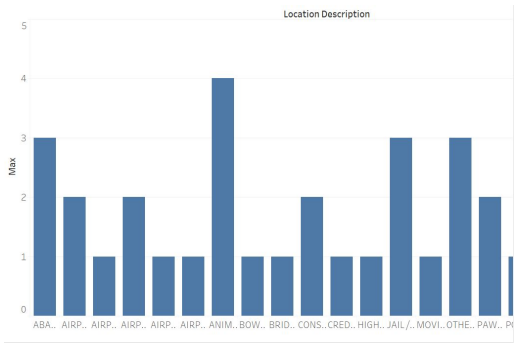


Q10. Which location is dangerous in terms of Assault?

The serious crimes that happened in the city are Theft and Assault. It is interesting to note that assault has been reported mostly in the streets of Chicago. The smaller number of assaults happened in the Airport terminal lower level, movie theatres, and so on.

```
select Location_Description, Primary_Type, count(*) as max from chicago_crimes
where Primary_Type='ASSAULT' group by Location_Description, Primary_Type
order by max desc limit 20;
```

Location_Description	Primary_Type	max
STREET	ASSAULT	1664
APARTMENT	ASSAULT	1511
RESIDENCE	ASSAULT	1419
SIDEWALK	ASSAULT	1260
SCHOOL, PUBLIC, BUILDING	ASSAULT	650
OTHER	ASSAULT	375
RESIDENCE PORCH/HALLWAY	ASSAULT	349
PARKING LOT/GARAGE(NON.RESID.)	ASSAULT	210
RESTAURANT	ASSAULT	209
ALLEY	ASSAULT	189
RESIDENTIAL YARD (FRONT/BACK)	ASSAULT	185



Q11. Locations(description) with highest crime reported

There are some crimes that are reported in high number as well as violated in more number. In this Theft and battery ranked high in the city that are reported.

```
select Location_Description, Primary_Type, count(*) as max
from chicago_crimes group by Location_Description , Primary_Type
order by max desc limit 20;
```

Location_Description	Primary_Type	max
STREET	THEFT	7507
APARTMENT	BATTERY	7153
SIDEWALK	NARCOTICS	5756
STREET	CRIMINAL DAMAGE	5714
STREET	NARCOTICS	5643
STREET	MOTOR VEHICLE THEFT	5503
RESIDENCE	BATTERY	5402
SIDEWALK	BATTERY	4019
RESIDENCE	OTHER OFFENSE	3812
STREET	BATTERY	3536
RESIDENCE	THEFT	3006

II. Non Relational Database - GOT

About the dataset : Our dataset contains two important nodes. We got the dataset from https://raw.githubusercontent.com/joakimskoog/AnApiOfIceAndFire/master/data/houses.json

https://raw.githubusercontent.com/joakimskoog/AnApiOfIceAndFire/master/data/characters.json

1. Characters : In our dataset we have almost 2200 people. Each person has 17 attributes. They are :
    a. Id : A unique id for each character
    b. Name : Name of the character
    c. isFemale : A boolean value to represent the gender of the person
    d. Culture: The culture person belongs to
    e. Aliases : The aliases that person has
    f. Born : When was the person born
    g. Died : When did the person die
    h. Father : The father of the person
    i. Mother : The mother of the person
    j. Spouse: The wife of the person
    k. Children : The name(s) of the person's children
    l. Allegiances : The allegiances of the person
    m. Books : The books in which the person appeared
    n. PlayedBy : Who played the character in the show
    o. TvSeries : The TvSeries person played in
2. Houses : This contains the names of various houses in Game of Thrones. They contain the following attributes :
    a. Id : The id of the house
    b. Name : Name of the house
    c. Seat : Seats the house has
    d. Region : The region the house is in
    e. CoatOfArms : Description of the house logo

f. Words : The motto of the house
g. Titles : The title of the house
h. CurrentLord : The name of the lord of the house
i. Founder : The name of the founder
j. Founded : The data of the house founded
k. Heir : The next heir to the house
l. DiedOut : The date when the house perished
m. AncestralWeapons : List of AncestralWeapons

Why did we choose this dataset?: Game of Thrones is a very complex book/tv show with a lot of characters and multiple relationships between each other, so we thought it would be a good way to represent a Graph DB.

Schema:

There are 10 relationships among 4 nodes in our schema. The 4 nodes are :

1. Person
2. House
3. Seat
4. Region

The 10 relationships are

1. Heir_to House - (Person to house)
2. Allied_with - (Person to house)
3. Led_by - (Person to house)
4. Founded_By - (Person to house)
5. Seat_of - (Seat to house)
6. In_Region - (House to region)
7. Sworn_To - (House to house)
8. Branch_of - (House to house)
9. Spouse - (Person to person)
10. Parent_of - (Person to person)

In the below figure you can see our schema.

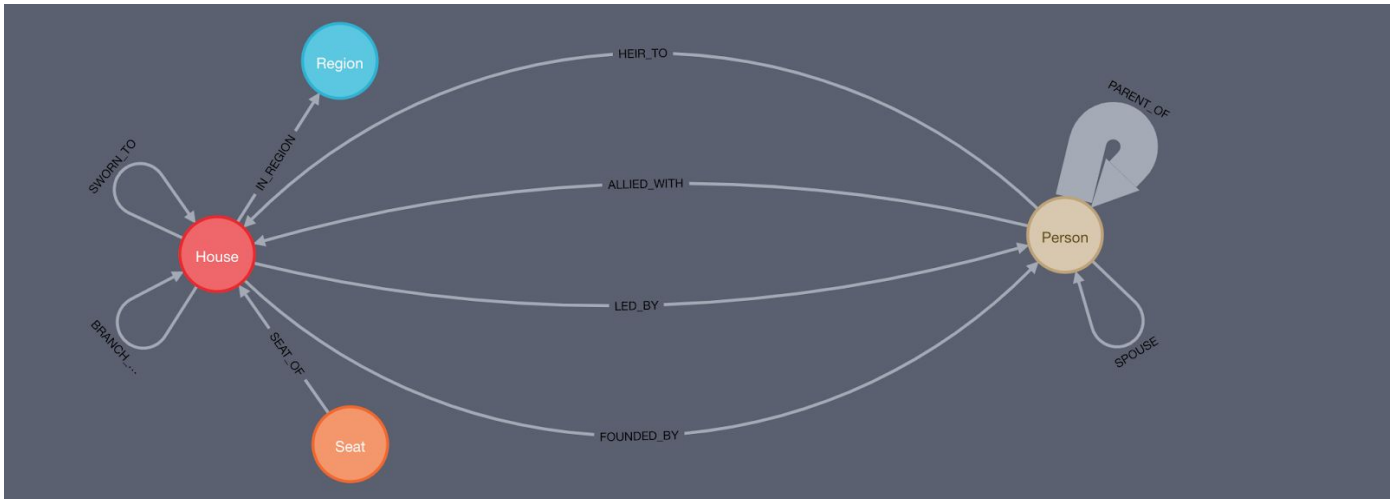


Fig 2 : Schema of our database

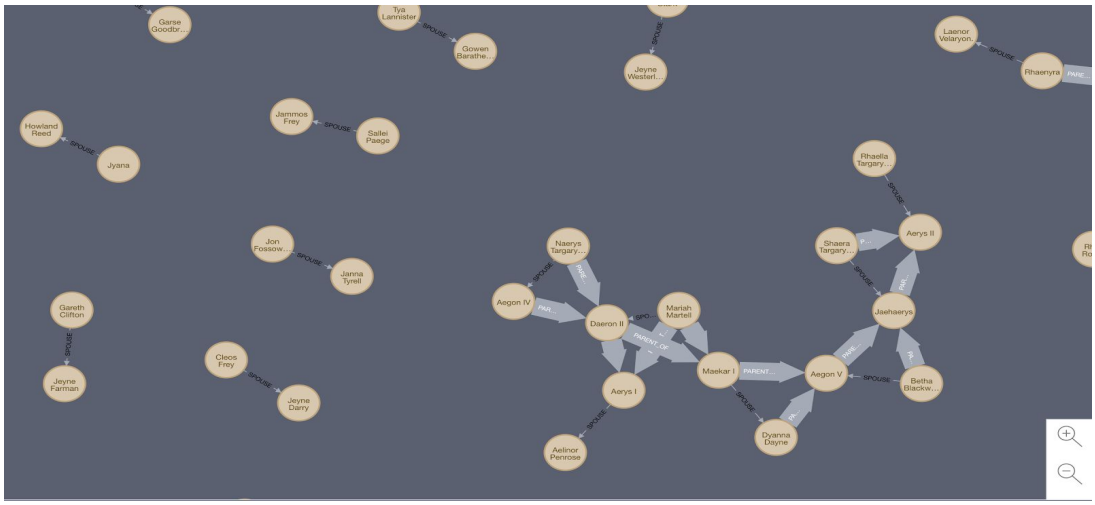
Building schema and data importation :

Cipher query was used for querying the data in Neo4j browser. We used Apoc library to load the data in the form of a Json file (url link). We converted the data to a map using map() in apoc and stored the value. The entire code of schema building is included in our report. We used Apoc, GraphQL and Graph Algorithms plugin.

Queries :

Q1. Displaying all persons with spouses :

This query fetches all the married couples and shows the relationship between them



Q2. Allies :

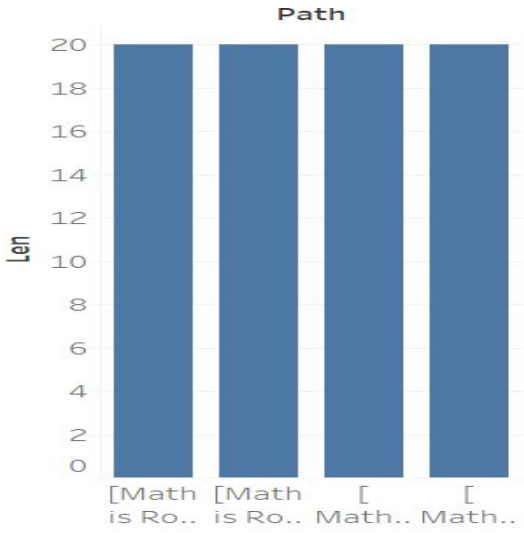
This query returns the maximum allies, minimum allies and average allies a person has.

min_allies	max_allies	avg_allies
1	4	1.1404657933042206

Q3. Shortest path :

Apoc function is used to find the shortest path between two people.

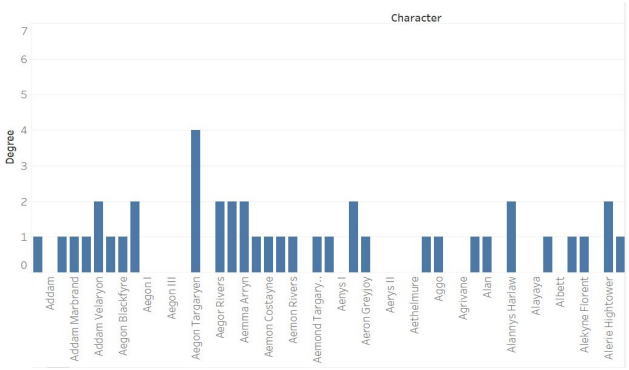
len	path
20	["Mathis Rowan", "House Rowan of Goldengrove", "Bethany Redwyne", "House Redwyne of the Arbor", "Mina Tyrell", "House Tyrell of Highgarden", "Alerie Hightower", "House Hightower of the Hightower", "Rhaena Targaryen", "House Targaryen of King's Landing", "Pycelle", "House Lannister of Casterly Rock", "Tion Frey", "House Frey of the Crossing", "Unknown", "House Butterwell", "Tommard Heddle", "House Blackfyre of King's Landing", "Aegor Rivers", "House Bracken of Stone Hedge", "Barba Bracken"]
20	["Mathis Rowan", "House Rowan of Goldengrove", "Bethany Redwyne", "House Redwyne of the Arbor", "Mina Tyrell", "House Tyrell of Highgarden", "Alerie Hightower", "House Hightower of the Hightower", "Rhaena Targaryen", "House Targaryen of King's Landing", "Jeyne Westerling", "House Westerling of the Crag", "Jeyne Westerling", "House Stark of Winterfell", "Myriame Manderly", "House Manderly of White Harbor", "Donella Hornwood", "House Hornwood of Hornwood", "Berena Hornwood", "House Tallhart of Torrhen's Square", "Benfred Tallhart"]
20	["Mathis Rowan", "House Rowan of Goldengrove", "Bethany Redwyne", "House Redwyne of the Arbor", "Mina Tyrell", "House Tyrell of Highgarden", "Alerie Hightower", "House Hightower of the Hightower", "Rhaena Targaryen", "House Targaryen of King's Landing", "Pycelle", "House Lannister of Casterly Rock", "Tion Frey", "House Frey of the Crossing", "Unknown", "House Butterwell", "Tommard Heddle", "House Blackfyre of King's Landing", "Aegor Rivers", "House Bracken of Stone Hedge", "Bethany Bracken"]
20	["Mathis Rowan", "House Rowan of Goldengrove", "Bethany Redwyne", "House Redwyne of the Arbor", "Mina Tyrell", "House Tyrell of Highgarden", "Alerie



Degree :

Q4. To find the degree of each person allied_with and displaying in descending order

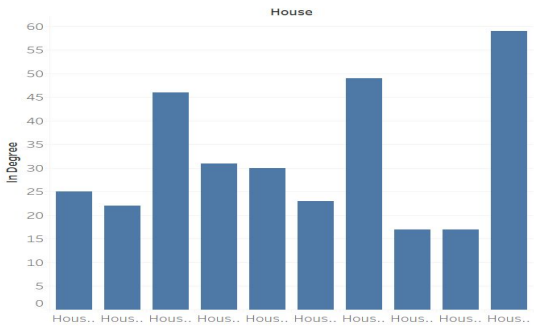
character	degree
"Elaena Targaryen"	4
"Alyssa Velaryon"	3
"Brienne of Tarth"	3
"Ellyn Reyne"	3
"Jeyne Westerling"	3
"Lysa Arryn"	3
"Rhaena Targaryen"	3
"Pycelle"	3
"Rohanne Webber"	3



In degree :

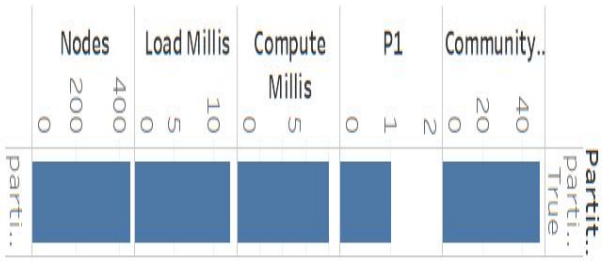
Q5. Finding the in degree for each person, that is the number of connections exist

house	in_degree
"House Tyrell of Highgarden"	59
"House Lannister of Casterly Rock"	49
"House Baratheon of King's Landing"	46
"House Baratheon of Storm's End"	31
"House Bolton of the Dreadfort"	30
"House Arryn of the Eyrie"	25
"House Greyjoy of Pyke"	23
"House Baelish of Harrenhal"	22
"House Stark of Winterfell"	17



Q6. It displays information about each node present performing union find algorithm from the apoc. algo library.

loadMillis	computeMillis	writeMillis	postProcessingMillis	nodes	communityCount	setCount	p1	p5	p10	p25	p50	p75	p90	p95	p99	p10
41	11	2	2	444	48	48	1	1	1	1	1	1	1	1	393	393



Conclusion and Future Work:

Relational - MYSQL

Firstly, the dataset that was obtained from kaggle has lot of unclean data and then the data is cleaned and built a SQL database.Chicago Crime dataset has many attributes that can be queried in many ways. In working with the Chicago crime dataset, it gave good insight about the crimes that happened in the chicago. By analyzing the queries and their output we can build a machine learning efficient model to predict and recommend to the people about the type of crimes , locations and time of the crime. Respective data visualizations can speak a lot more than that.

Non Relational - Neo4j

The data that is analyzed can be used to recommend to Non Relational – Neo4J. The data that is analyzed is used to find the number of people, aliases, founders, regions of the houses. Using different libraries for performing different apoc algorithms. For visualization we used tableau.

Contributions:

Name	Tasks
Divya Dasararaju	Dataset collection, Data Cleaning, Pre-processing, Relational part Database Building, Query Creation and analysis and Report Writing
Neeharika Mallineni	Queries on Neo4J involving apoc and different algorithms,Data visualization for both Relational and Non Relational Databases.
Siddharth Chittoor	Neo4j data collection, pre-processing, schema building and few queries.