SECURITY ON PUBLIC TRANSPORTATION IN CHICAGO

GROUP 6 | Jainam Mehta, Julian Kleindiek, Lola Johnston, Peter Eusebio

AGENDA

1. Executive Summary & Business Case
2. Data Overview & Preparation
3. Database Design
4. Solution Architecture
5. Data Analysis & Analytics Dashboard
6. Conclusions

EXECUTIVE SUMMARY

- Crime on public transportation and in its surroundings is severe in Chicago, posing a threat to consumers and a challenge to law enforcement authorities
- Many of the UChicago students are frequently using public transportation and hence affected by this issue
- The objective of this project is to **discover patterns in the occurrence of crime on and around public transportation**
- The insights derived will be used to:

  Give consumers guidance on the risk related to taking the route that they are anticipating to take

  Help authorities understand what factors contribute to spikes in crime and give recommendations on how to provide more safety on public transportation

BUSINESS USE CASE

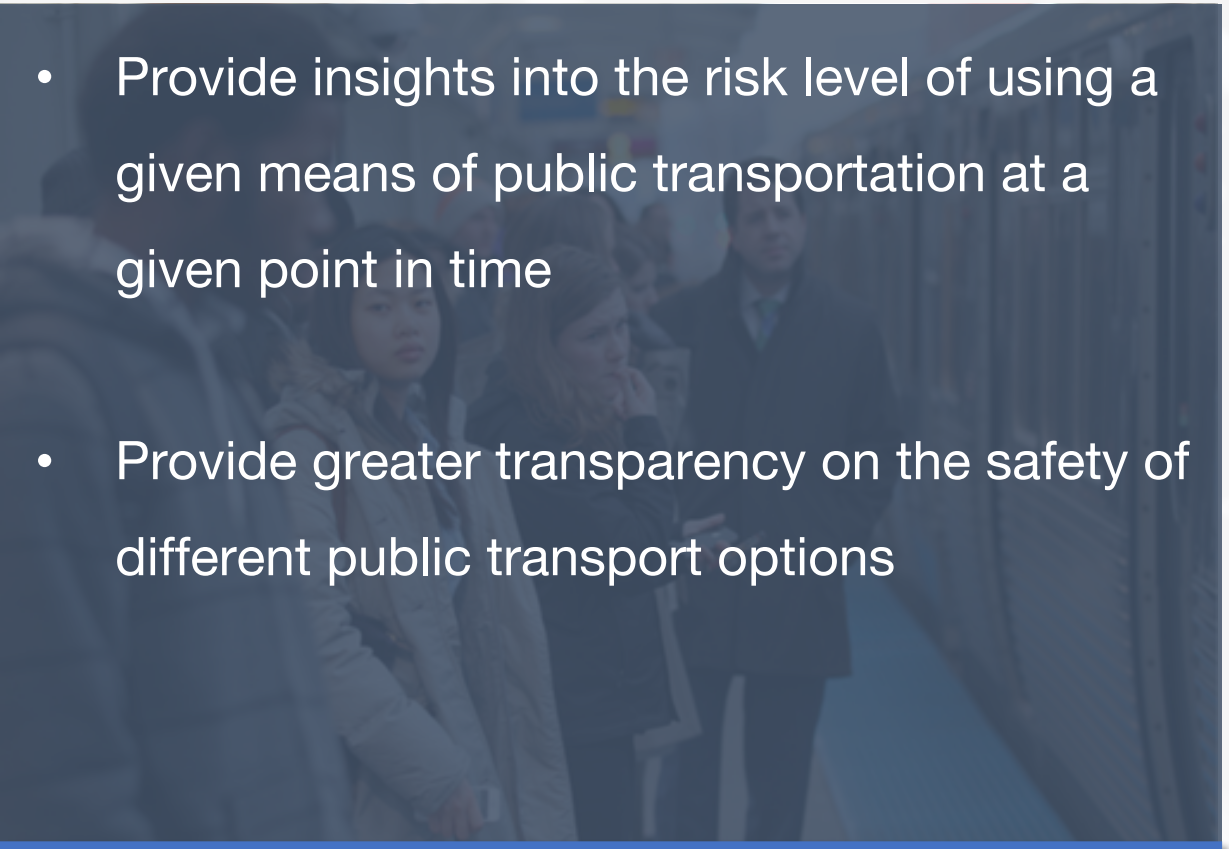
- Exposed to crime on public transportation
- Lack of information on the safety of taking public transportation
- Avoiding dangerous routes as a tourist
- Trade off between safety and cost for transportation
- Lack of safer options

Any person using public transportation in Chicago, including commuters, residents and tourists

- Chicago’s reputation as one of the most dangerous cities in the US
- Inability to ensure safety and prevent crime on public transportation
- Lack of information on how external factors influence crime on public transportation
- Lack of information on where to increase staffing and presence

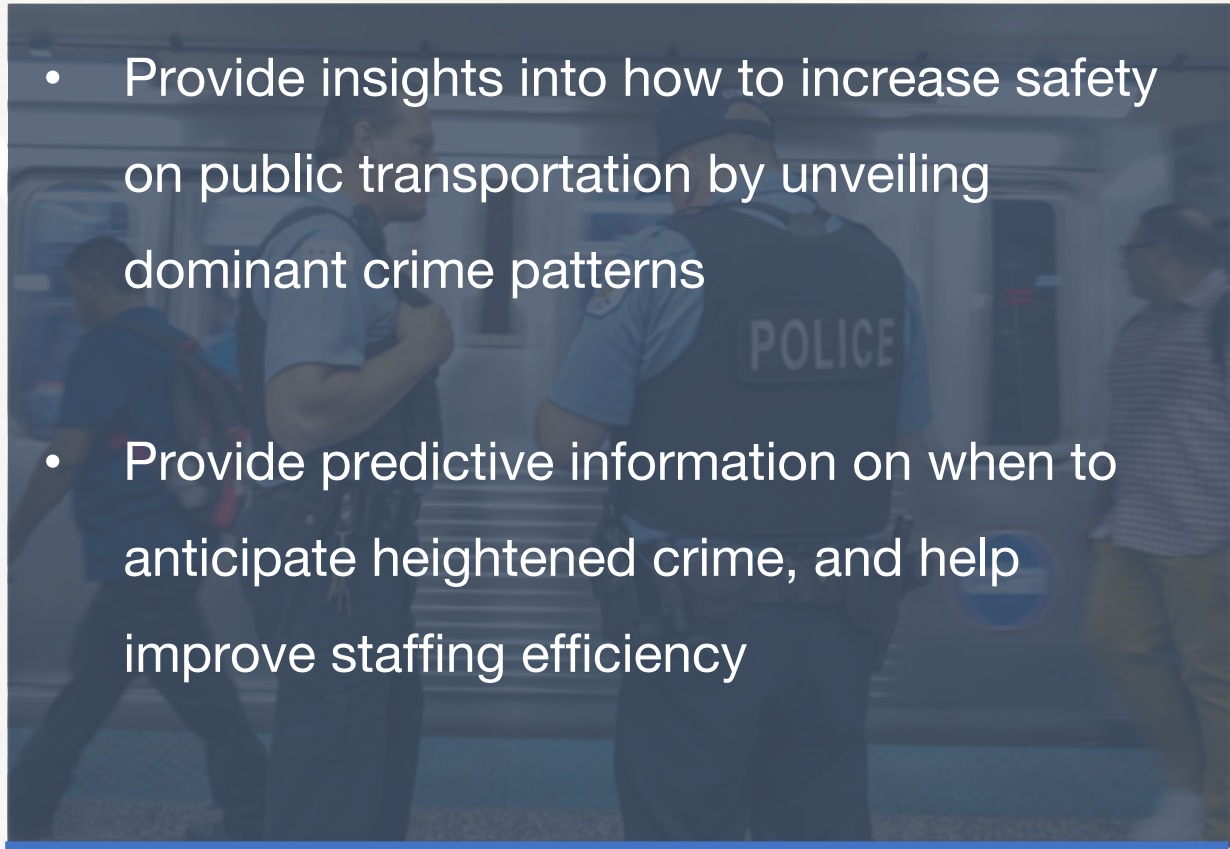
Any organization responsible for providing safety on public transportation in Chicago e.g. CPD, private security

SOLUTION AND IMPACT



- Provide insights into the risk level of using a given means of public transportation at a given point in time
- Provide greater transparency on the safety of different public transport options

CONSUMER



- Provide insights into how to increase safety on public transportation by unveiling dominant crime patterns
- Provide predictive information on when to anticipate heightened crime, and help improve staffing efficiency

AUTHORITIES

DATA OVERVIEW & PREPARATION

DATA OVERVIEW | CRIME

CHICAGO DATA PORTAL

We accessed the crime statistics published here by the City of Chicago via an API, filtering for all CTA-related crime cases. We then cleaned the data in Python according to our needs.

API Call

```
In [3]: # Select date to filter crime dataset for
        date = '2019-11-12T00:00:00.000'

In [4]: # prepare where statement of the API call
        statement = "date <= '" + date + "' AND location_description = 'CTA PLATFORM' OR date <= '" + date
        + "' AND location_description = 'CTA BUS' OR date <= '" + date + "' AND location_description = 'CT
        A TRAIN' OR date <= '" + date + "' AND location_description = 'CTA BUS STOP' OR date <= '" + date
        + "' AND location_description = 'CTA GARAGE / OTHER PROPERTY'"
        statement

Out[4]: "date <= '2019-11-12T00:00:00.000' AND location_description = 'CTA PLATFORM' OR date <= '2019-11-1
        2T00:00:00.000' AND location_description = 'CTA BUS' OR date <= '2019-11-12T00:00:00.000' AND loca
        tion_description = 'CTA TRAIN' OR date <= '2019-11-12T00:00:00.000' AND location_description = 'CT
        A BUS STOP' OR date <= '2019-11-12T00:00:00.000' AND location_description = 'CTA GARAGE / OTHER PR
        OPERTY'"

In [5]: ## WARNING: this query takes approx. 3 minutes to run; don't run it everytime you run this script

        # Pull all crime data for a given date and for crimes with a location description related to CTA
        # API instructions https://dev.socrata.com/foundry/data.cityofchicago.org/ijzp-q8t2

        # Authenticate client (needed for non-public datasets):
        client = Socrata("data.cityofchicago.org",
                         "QtMhXqaTTglPlVSJAC6PEQQxD", username = "juli.kleindiek@gmail.com", password =
        "DEPA_2019")

        # Limit to 1000 rows for test purposes
        results = client.get("ijzp-q8t2",
                             where = statement,
                             limit = 200000)

In [6]: # Convert results to pandas DataFrame
        crime_dirty = pd.DataFrame.from_records(results)
```

Resulting crime Data Frame

```
In [78]: crime.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 102338 entries, 11694436 to 9999890
Data columns (total 23 columns):
caseNumber              102338 non-null object
datetime                102338 non-null datetime64[ns]
block                   102338 non-null object
iucr                    102338 non-null object
primaryType             102338 non-null object
description             102338 non-null object
locationDescription     102338 non-null object
arrest                  102338 non-null bool
domestic                102338 non-null bool
beat                    102338 non-null int64
district                102338 non-null int64
ward                    95593 non-null float64
communityArea           95626 non-null float64
fbiCode                 102338 non-null object
xCoordinate             102338 non-null float64
yCoordinate             102338 non-null float64
year                    102338 non-null object
updatedOn               102338 non-null datetime64[ns]
latitude                102338 non-null float64
longitude               102338 non-null float64
date                    102338 non-null datetime64[ns]
time                    102338 non-null object
gridId                  102338 non-null int64
dtypes: bool(2), datetime64[ns](3), float64(6), int64(3), object(9)
memory usage: 17.4+ MB
```

DATA OVERVIEW | PUBLIC TRANSPORTATION

Chicago Transit Authority

We download the data on Train Stops here and on Bus Stops here and then imported it into Python where we clean it according to our needs.

Resulting BusStops Data Frame

```
In [82]: BusStops.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14036 entries, 0 to 10894
Data columns (total 14 columns):
stopID         14036 non-null object
systemStop     14036 non-null object
street         14036 non-null object
crossSt        14036 non-null object
dir            14036 non-null object
pos            14036 non-null object
routesStpg     14036 non-null object
owlRoutes      14036 non-null object
city           14036 non-null object
status         14036 non-null bool
publicNam      14036 non-null object
latitude       14036 non-null float64
longitude      14036 non-null float64
gridId         14036 non-null int64
dtypes: bool(1), float64(2), int64(1), object(10)
memory usage: 1.5+ MB
```

Resulting TrainStops Data Frame

```
In [83]: TrainStops.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 292 entries, 0 to 299
Data columns (total 19 columns):
stopID                  292 non-null int64
directionID             292 non-null object
stopName                292 non-null object
stationName             292 non-null object
stationDescriptiveName  292 non-null object
mapID                   292 non-null int64
ada                     292 non-null bool
red                     292 non-null bool
blue                    292 non-null bool
g                       292 non-null bool
brn                     292 non-null bool
p                       292 non-null bool
pExp                    292 non-null bool
y                       292 non-null bool
pnk                     292 non-null bool
o                       292 non-null bool
latitude                292 non-null float64
longitude               292 non-null float64
gridId                  292 non-null int64
dtypes: bool(10), float64(2), int64(3), object(4)
memory usage: 25.7+ KB
```

DATA OVERVIEW | WEATHER AND HOLIDAYS

The Weather dataset contains various metrics (temperature, precipitation) from 2001 to present. The holidays dataset contains a list of US holidays and their corresponding dates.

Information on the weather data

```
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6884 entries, 0 to 6883
Data columns (total 11 columns):
Date          6884 non-null datetime64[ns]
Elevation     6884 non-null float64
Latitude      6884 non-null float64
Longitude     6884 non-null float64
Prcp          6884 non-null float64
Station       6884 non-null object
Tavg          3867 non-null float64
Tmax          6884 non-null float64
Tmin          6884 non-null float64
Tobs          0 non-null float64
Tsun          1085 non-null float64
dtypes: datetime64[ns](1), float64(9), object(1)
memory usage: 591.7+ KB
```

Information on the holiday data

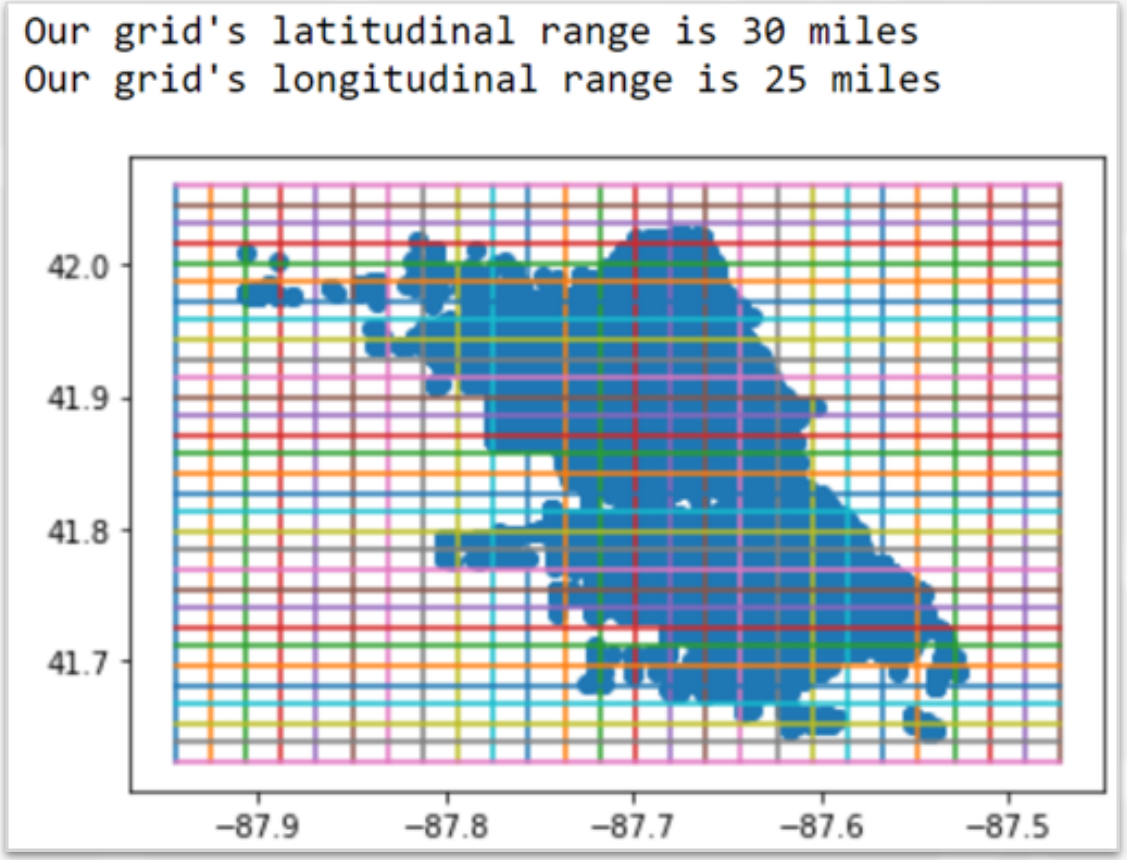
```
hday.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 189 entries, 0 to 188
Data columns (total 2 columns):
Date       189 non-null datetime64[ns]
Holiday    189 non-null object
dtypes: datetime64[ns](1), object(1)
memory usage: 3.0+ KB
```

DATA OVERVIEW | GRID TABLE

A grid of 1x1 mile squares was generated around the crime report locations.

Map of crime reports inside the grid



Code for grid assignment

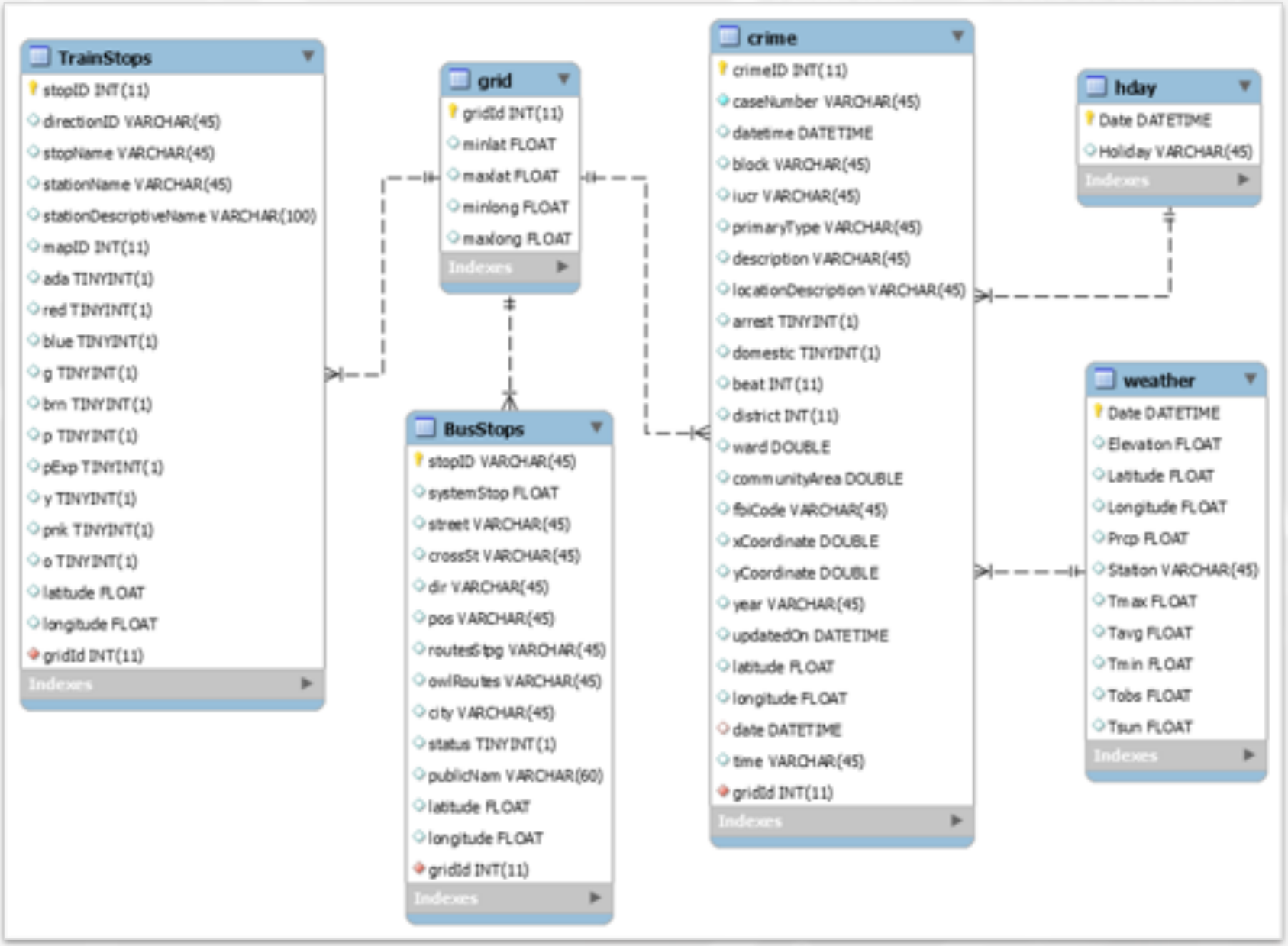
```
#gives gridId corresponding to a lat long pair

def gridsort(lat,long):

    xin = np.nan
    yin = np.nan

    for i in range(0,len(x)-1):
        if (x[i] <= long) & (long < x[i+1]):
            xin = float(i)
            break

    for i in range(0,len(y)-1):
        print(i)
        if (y[i] <= lat) & (lat < y[i+1]):
            yin = float(i)
            break

    gridId = int((xin + 1) + (len(x)-1)*(yin))

    return gridId
```

DATABASE DESIGN

DESIGN CONSIDERATIONS

- The key to setting up a functioning database was the creation of a grid table that enabled us to join the crime table with the bus and train stop tables
- Data is organized in an Enhanced Entity Relationship (EER) diagram
- A star schema was not created because:
  - The database doesn't have many tables, making joins relatively easy and computationally inexpensive to perform
  - The crime data itself is updated daily but by a relatively low amount of new rows (approx. 14 on average)
  - The EER best communicates the relationships between the distinct data sources
  - Revealing the relationship between these data sources is the purpose of our project

Enhanced Entity Relationship diagram



- Grid table allows joins of the crime table with the BusStops and TrainStops tables using gridId as foreign key
- Holiday and weather tables connect directly into the crime table using date as foreign key

SOLUTION ARCHITECTURE

PLATFORM CONSIDERATIONS

PREPARATION

Alternatives: OpenRefine

Programmatic tool compared to OpenRefine which requires GUI

Wider availability of packages (pandas, numpy)

Ability to directly interface with SQL server

DATABASE

Alternatives: AWS, Azure

Greater familiarity with GCP compared to other platforms

Ease of access through shell directly within Chrome

Availability of free credits

VISUALIZATION

Alternatives: PowerBI, Metabase

Better performance on high volume datasets

PowerBI workspace has a limit of 10GB data

Greater flexibility in connecting with database sources and servers

SOLUTION ARCHITECTURE

Data → python → Google Cloud → tableau

- Crime
- TrainStops
- BusStops
- GridId
- Holiday
- Weather

Pull

Clean

Prepare

Process

Cloud SQL
stores all cleaned data

Bucket
stores .py script for daily API call of crime data

Compute Engine
executes the .py script daily at 8:00 AM CT and appends new data to Cloud SQL

Risk Score Development

Data Visualization

DATA ANALYSIS & ANALYTICS DASHBOARD

INSIGHTS | QUERY

Most number of crimes occured on CTA Platforms

```
# Find the number of crimes ocurring by Location type
SELECT locationDescription, COUNT(*) AS numberOfCrimes
FROM crime
GROUP BY locationDescription
ORDER BY COUNT(*) DESC;
```

locationDescription	numberOfCrimes
CTA PLATFORM	37213
CTA TRAIN	26012
CTA BUS	22490
CTA GARAGE / OTHER PROPERTY	10148
CTA BUS STOP	6684

Finding the top 5 types of crime

```
# Find the top 5 types of crime
SELECT primaryType, COUNT(*) AS numberOfCrimes
FROM crime
GROUP BY primaryType
ORDER BY COUNT(*) DESC
LIMIT 5;
```

primaryType	numberOfCrimes
THEFT	31794
DECEPTIVE PRACTICE	16839
BATTERY	15028
CRIMINAL DAMAGE	9329
ROBBERY	8852

INSIGHTS | QUERY

Bus Stops with the most crime

```
# Bus stops with the most crime occurences
SELECT routesStpg AS route, publicNam AS stopName, COUNT(*)
FROM BusStops
INNER JOIN crime USING (gridId)
WHERE locationDescription = "CTA BUS STOP"
GROUP BY stopID
ORDER BY COUNT(*) DESC
LIMIT 5;
```

route	stopName	numberOfCrimes
24	Wentworth & Marquette Rd	164
24	Wentworth & 69th Street	164
67	69th Street & Perry	164
24	Yale & 63rd Street	164
63	63rd Street & Wentworth	164

24 - Wentworth

Holiday with the most crime (2019)

Average Crimes per day: 14.85

```
# Average number of crimes per day
SELECT COUNT(*) / (DATEDIFF(MAX(date), MIN(date))) AS avgCrime
from crime;

# Top 5 Holidays with the most crime occurences in 2019
SELECT Holiday, DATE(Date) as Date, COUNT(*) as numberOfCrimes
FROM hday
INNER JOIN crime USING (Date)
WHERE YEAR(Date) = 2019
GROUP BY Holiday , Date
ORDER BY COUNT(*) DESC
LIMIT 5;
```

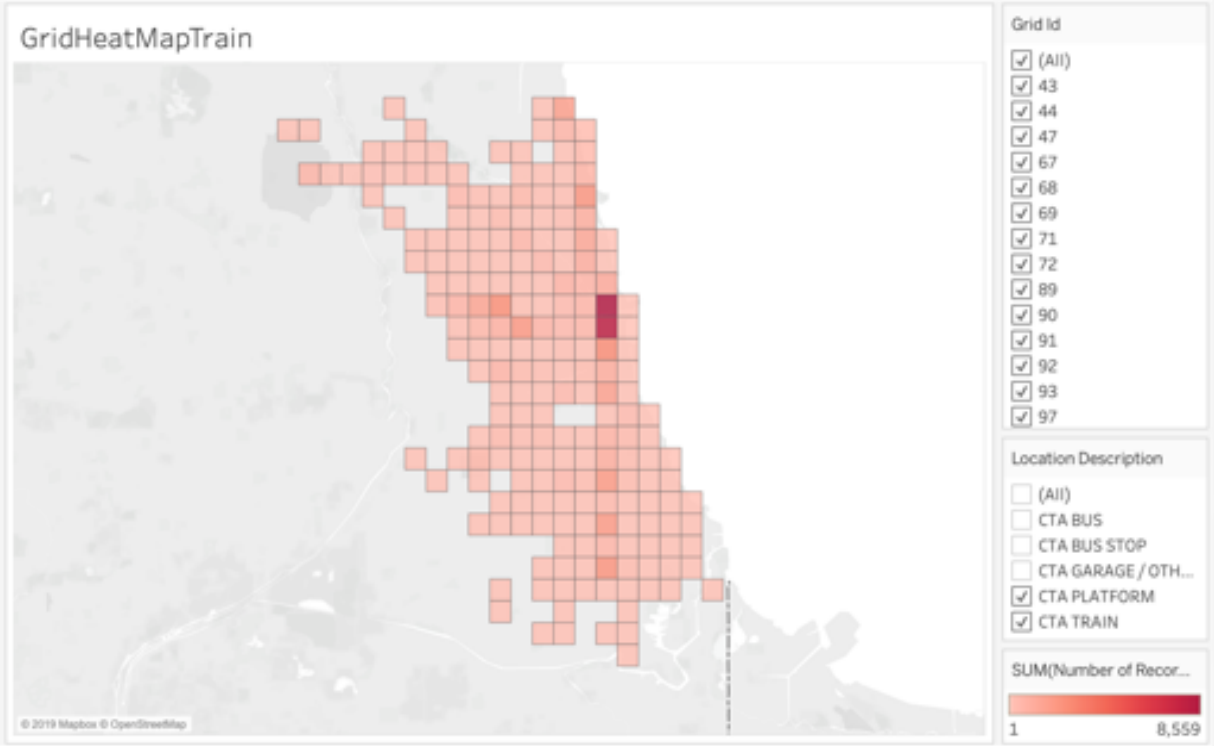
Holiday	Date	numberOfCrimes
Memorial Day	2019-05-27	20
Columbus Day	2019-10-14	18
New Year's Day	2019-01-01	15
Independence Day	2019-07-04	11
Washington's Birthday	2019-02-18	9

Memorial Day

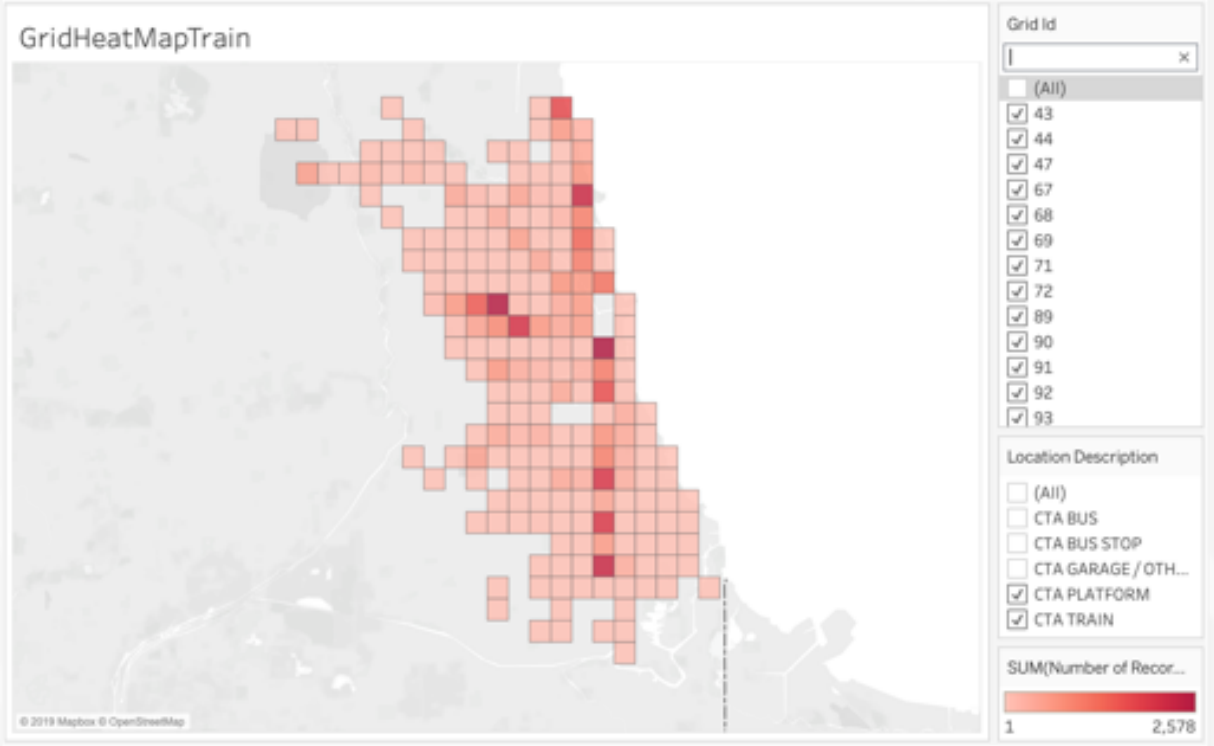
INSIGHTS | VISUAL Taking the train in Chicago

Riskiest areas for Train Riders: The **downtown area, the south and west side**

All train-related crimes



Train-related crimes excluding downtown (outlier)



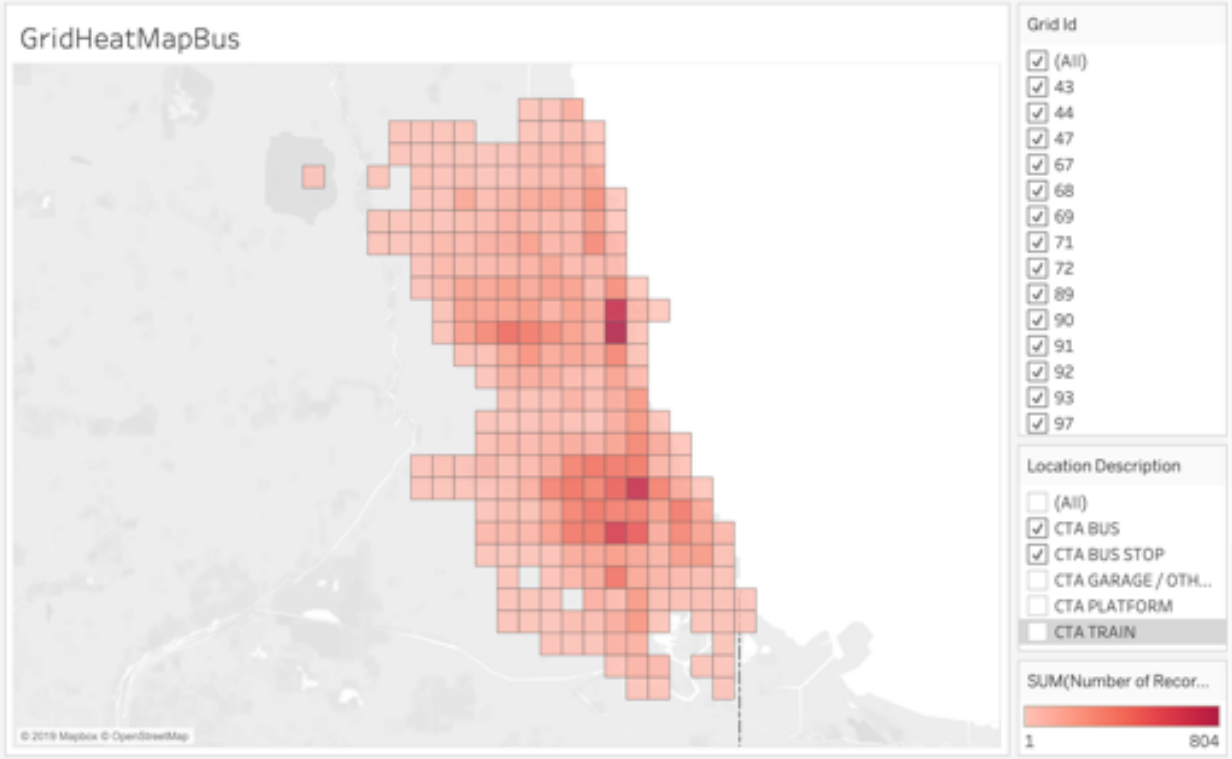
Crime (2001 - present)

INSIGHTS | VISUAL Taking the bus in Chicago

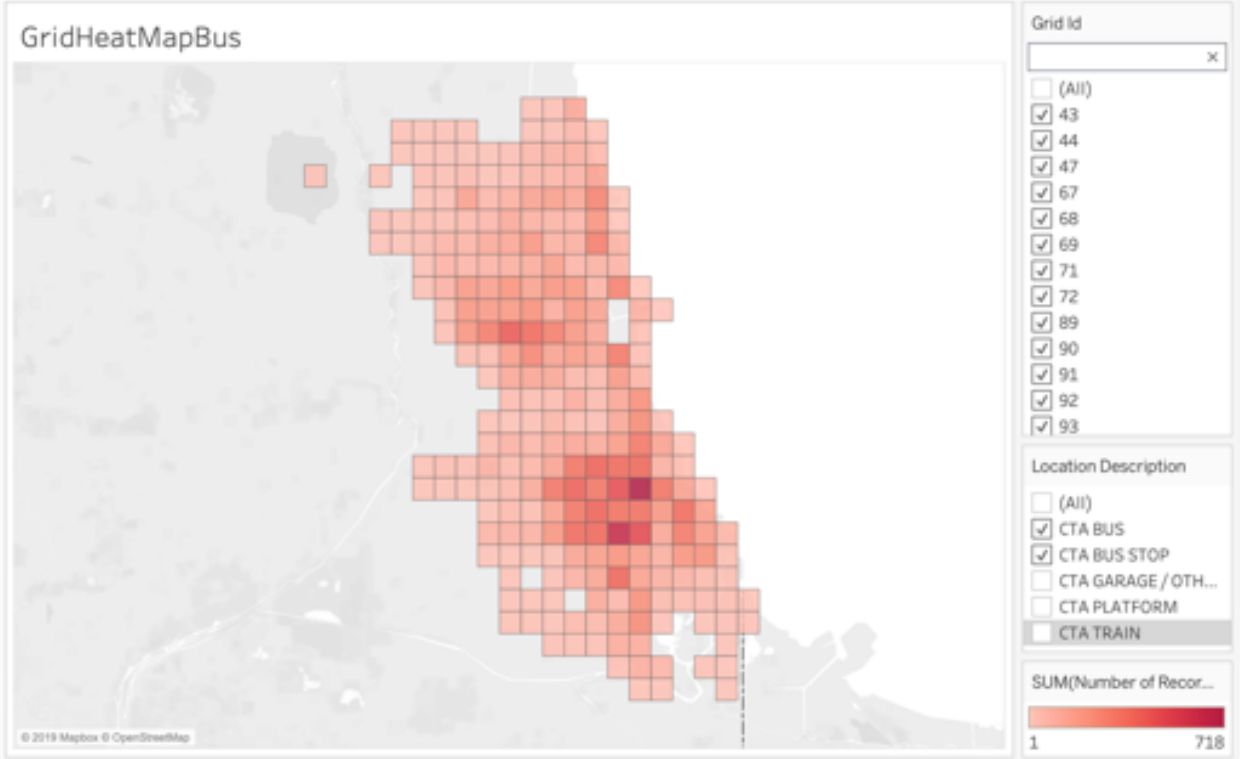
Riskiest areas for Bus Riders: **Downtown and the South Side**

There are much more crimes on trains (max. # of crimes on trains: 8,599 vs. 804 on buses)

All bus related crimes



Bus-related crimes excluding downtown (outlier)

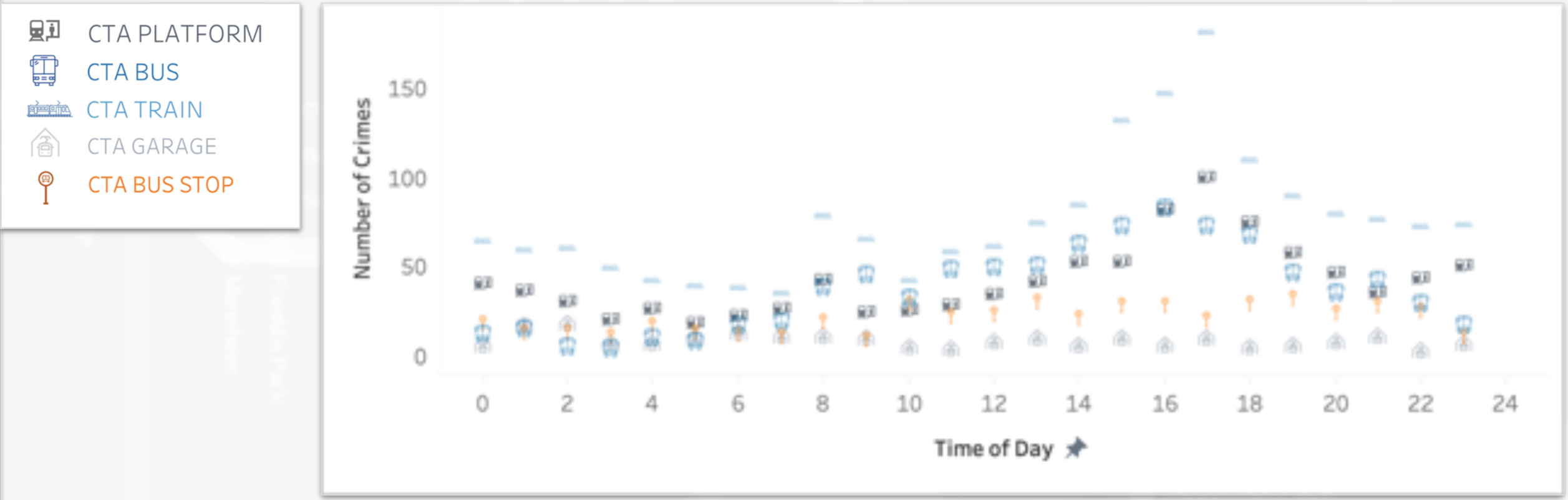


Crime (2001 - present)

INSIGHTS | VISUAL Crime & Time

Crimes occur most frequently during rush hour -- highest in the evening.

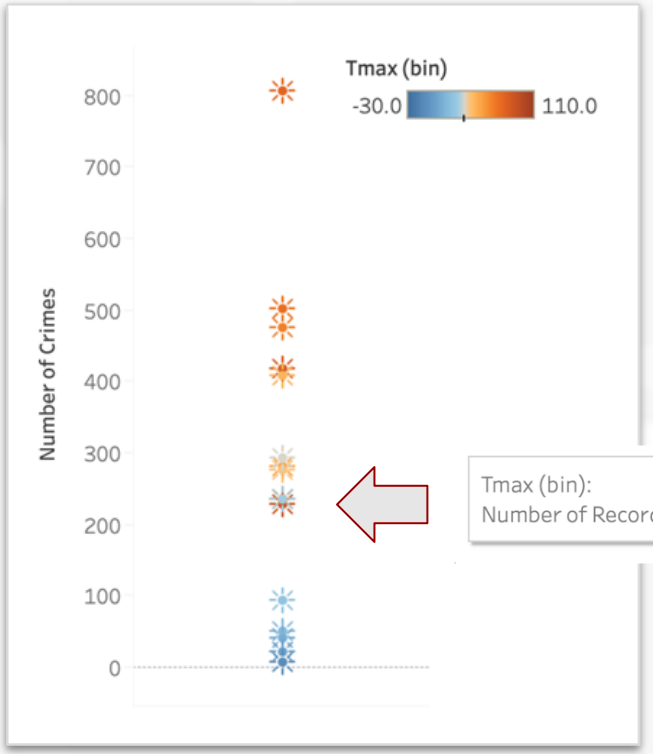
Trains are especially risky between 2pm - 7pm

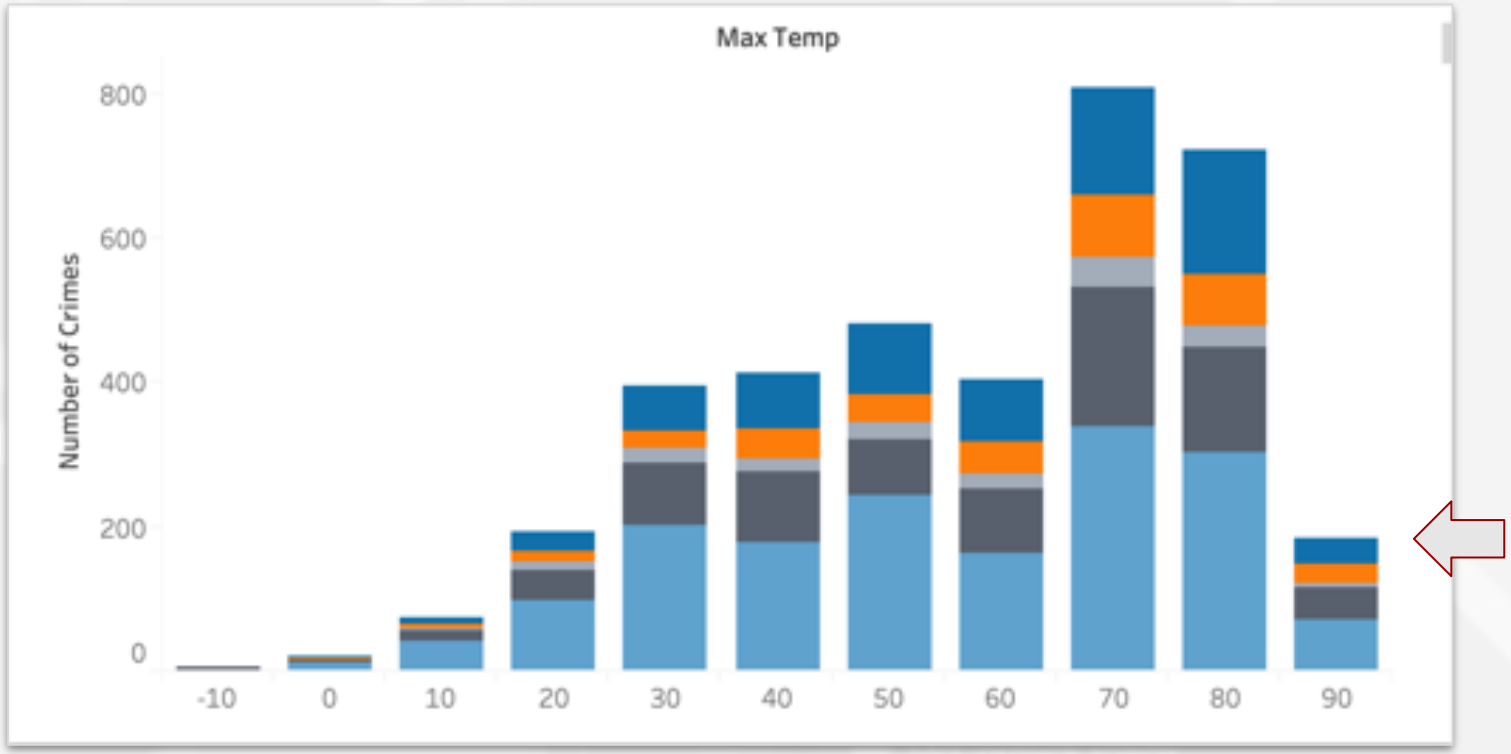


Crime by the hour by location type (2019 - present)

INSIGHTS | VISUAL Crime & Temperature

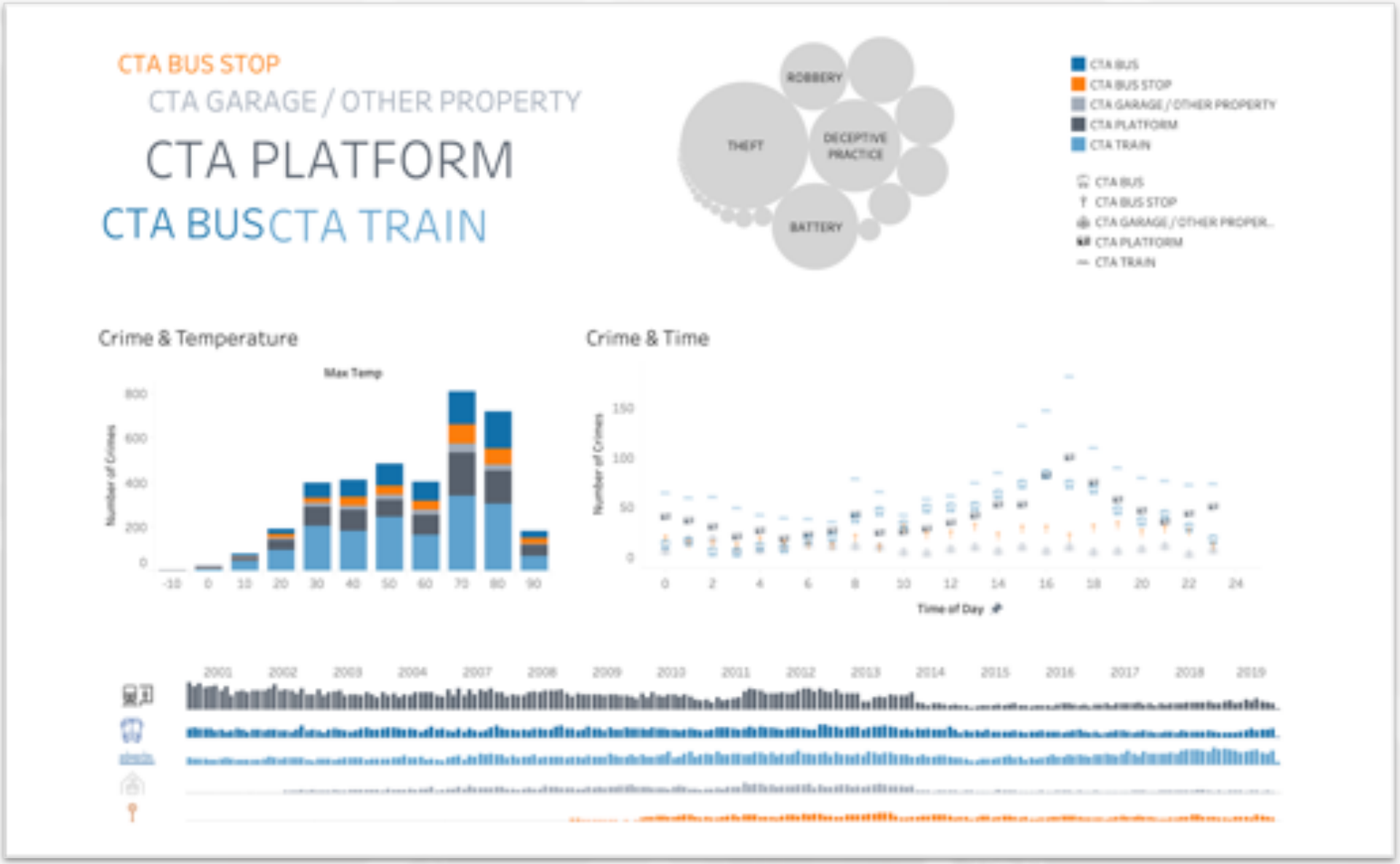
Crimes increase with warmer weather -- until temperatures hit 90+ degrees



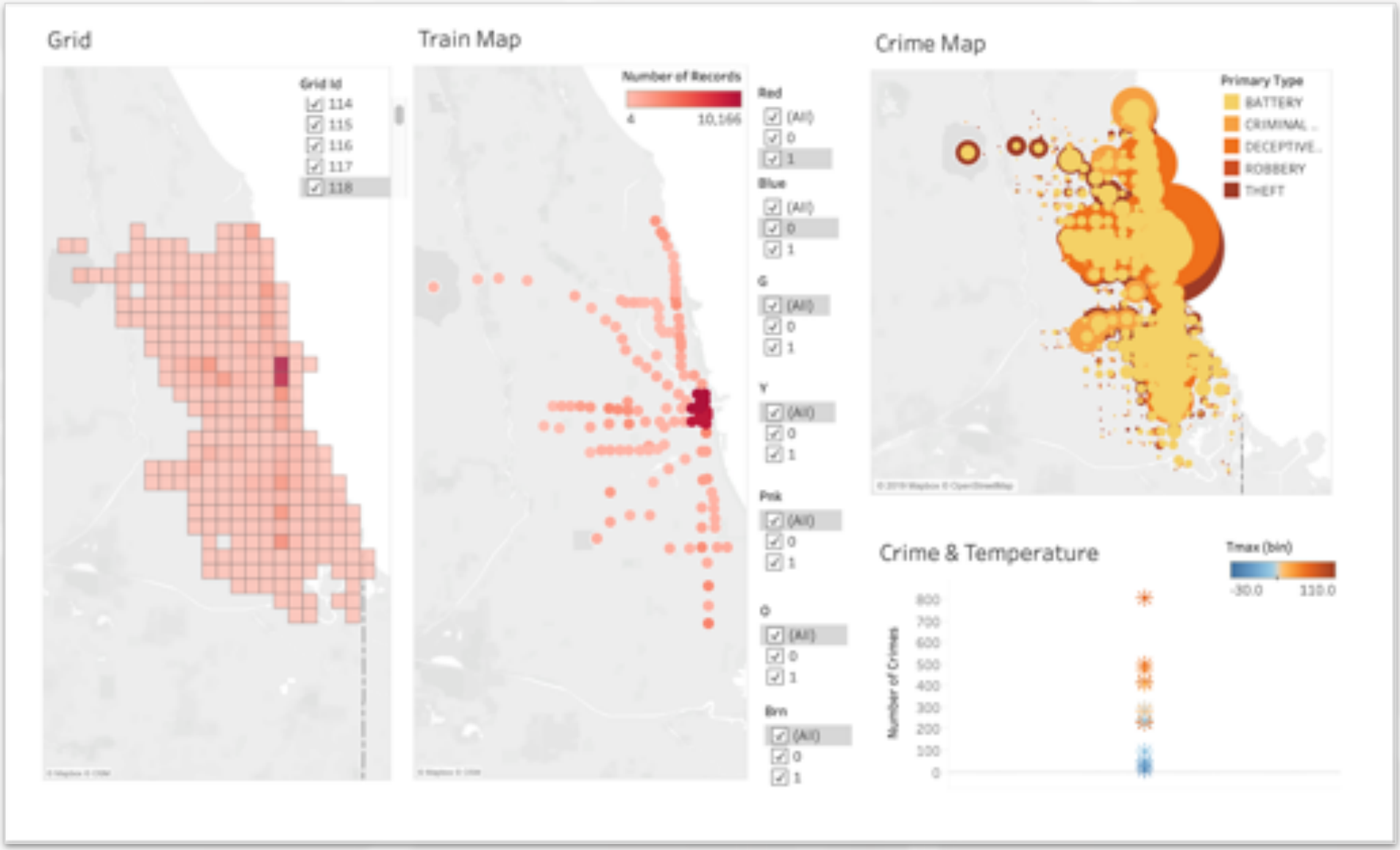


Crimes by temperature (2019 - present)

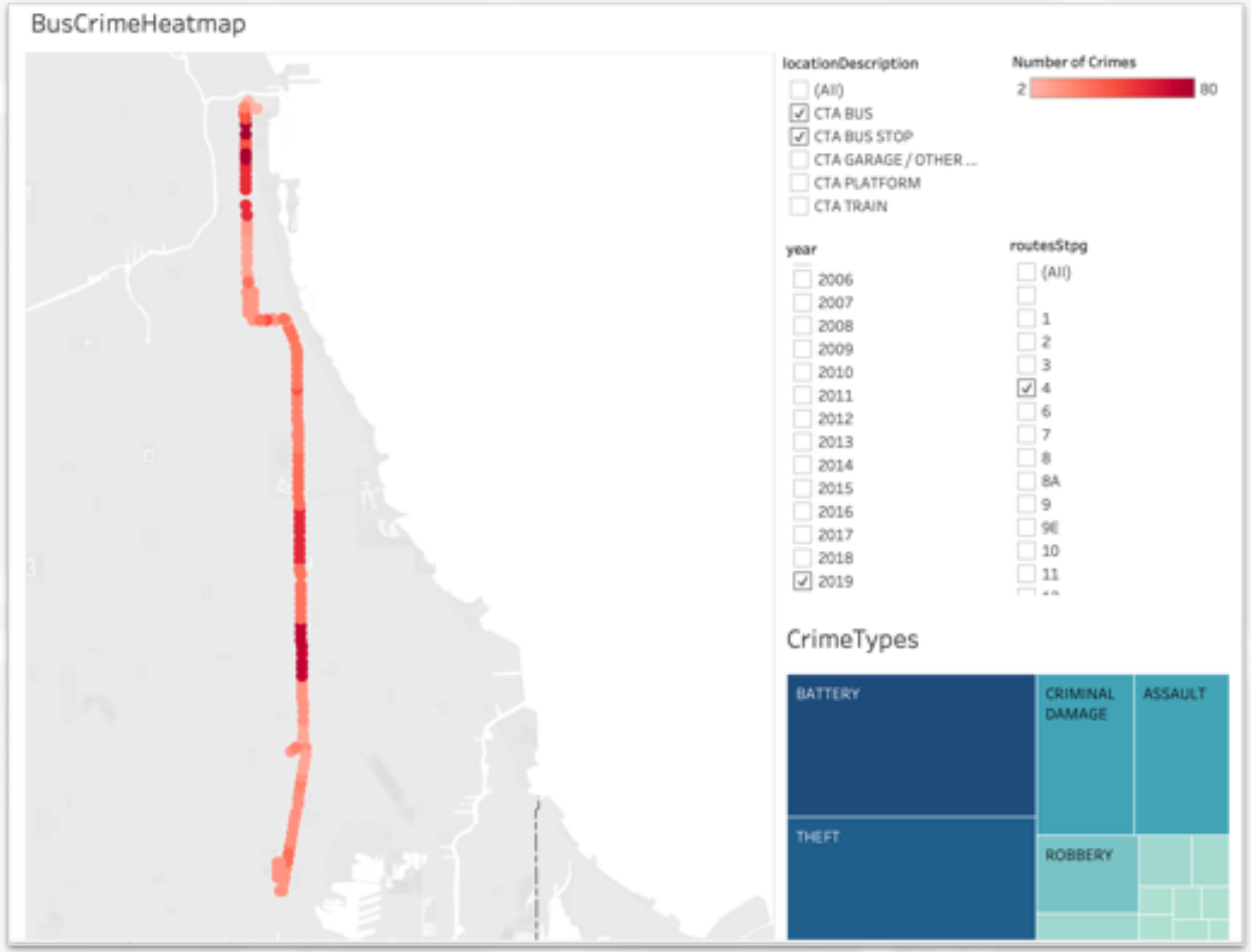
DASHBOARD | SUMMARY



DASHBOARD | MAPS I



DASHBOARD | MAPS II



RECOMMENDATIONS

SIX SAFETY TIPS! When riding the CTA...

1. Take the bus and avoid the train
2. Mind your personal effects, there's a lot of theft
3. Going Downtown or the Southside are both risky rides
4. If you must go west, buses are best
5. Peak hours for the criminal profession are in the PM between 2 and 7
6. When the weather is nice, think twice

CONCLUSION

LESSONS LEARNED AND SCOPE FOR IMPROVEMENT

- **The better you know the data, the more accurate the analysis**: We realized during the project that we needed ridership statistics to scale our metrics; some statistics weren't recorded until 2008/2013.
- **Collaboration constraints**: We primarily collaborated using GitHub, which worked fine except when multiple people worked on the same code simultaneously. Google Colab can be a useful alternative for simultaneous Jupyter Notebook editing.
- **Do it right from the start:** Designing the database to be compatible with our analytics use-case would have saved us time. While performing analyses, we discovered problems with the availability and structure of data that forced us to alter our DDL and data preparation scripts.
- **Pitch your error**: We all got stuck several times throughout the project and what helped us was to pitch the steps we took and the error we got to group members to reflect on the code and find a solution together.