Section 2: Data

Data Description

In this section, I will describe the data used to solve the problem as described previously.

As noted below in the Further Development Section, it is possible to attempt quite complex and sophisticated scenarios when approaching this problem. However, given the size of the project and for simplicity only the following scenario will be addressed:

1. Query the FourSqaure website for the top sites in Chicago
2. Use the FourSquare API to get supplemental geographical data about the top sites
3. Use the FourSquare API to get top restaurent recommendations closest to each of the top site
4. Use open source Chicago Crime data to provide the user with additional crime data

Top Sites from FourSquare Website

Although FourSquare provides a comprehensive API, one of the things that API does not easily support is a mechanism to directly extract the top N sites / venues in a given city. This data, however, is easily available directly from the FourSquare Website. To do this simply go to www.foursquare.com, enter the city of your choise and select Top Picks from *I'm Looking For* selection field.

Using BeautifulSoup and Requests the results of the Top Pick for Chicago was retrieved. A sample venue is shown below:

```
<div class="venueDetails">
    <div class="venueName">
        <h2>
    <a href="/v/millennium-park/42b75880f964a52090251fe3" target="_blank">Millennium
Park
    </a>
</h2>
    </div>
    <div class="venueMeta">
```

```
        <div class="venueScore positive" style="background-color: #00B551;"
title="9.7/10 - People like this place">9.7</div>
        <div class="venueAddressData">
            <div class="venueAddress">201 E Randolph St (btwn Columbus Dr &amp;
Michigan Ave), Chicago</div>
            <div class="venueData"><span class="venueDataItem"><span
class="categoryName">Park</span><span class="delim"> • </span></span>
            </div>
        </div>
    </div>
</div>
```

From this HTML the following data can be extracted:

- Venue Name
- Venue Score
- Venue Category
- Venue HREF
- Venue ID [Extracted from the HREF]

A sample of the extracted data is given below:

id	score	category	name	href
42b75880f964a52090251fe3	9.7	Park	Millennium Park	/v/millennium-park/42b75880f964a52090251fe3
4b9511c7f964a520f38d34e3	9.6	Trail	Chicago Lakefront Trail	/v/chicago-lakefront-trail/4b9511c7f964a520f38...
49e9ef74f964a52011661fe3	9.6	Art Museum	The Art Institute of Chicago	/v/the-art-institute-of-chicago/49e9ef74f964a5...
4f2a0d0ae4b0837d0c4c2bc3	9.6	Deli / Bodega	Publican Quality Meats	/v/publican-quality-meats/4f2a0d0ae4b0837d0c4c...
4aa05f40f964a520643f20e3	9.6	Theater	The Chicago Theatre	/v/the-chicago-theatre/4aa05f40f964a520643f20e3

We will have a closer look at this data gather later on when the supplemental geographical data has been added.

Supplemental Geographical Data

Using the `id` field extracted from the HTML it is then possible to get further supplemental geographical details about each of the top sites from FourSquare using the following sample API call:

```
# Get the properly formatted address and the latitude and longitude
url =
'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(
    venue_id,
    cfg['client_id'],
    cfg['client_secret'],
    cfg['version'])

result = requests.get(url).json()
result['response']['venue']['location']
```

The requests returns a JSON object which can then be queried for the details required. The last line in the sample code above returns the following sample JSON:


```
{
   "city":"Chicago",
   "lng":-87.62323915831546,
   "crossStreet":"btwn Columbus Dr & Michigan Ave",
   "neighborhood":"The Loop",
   "postalCode":"60601",
   "cc":"US",
   "formattedAddress":[
      "201 E Randolph St (btwn Columbus Dr & Michigan Ave)",
      "Chicago, IL 60601",
      "United States"
   ],
   "state":"IL",
   "address":"201 E Randolph St",
   "lat":41.8826616030636,
   "country":"United States"
}
```


From this the following attributes are extracted:

- Venue Address
- Venue Postalcode
- Venue City
- Venue Latitude
- Venue Longitude

Final FourSquare Top Sites Data

A sample of the final FourSquare Top Sites data is shown below:

id	score	category	name	address	postalcode	city	href	latitude	longitude
42b75880f964a52090251fe3	9.7	Park	Millennium Park	201 E Randolph St	60601	Chicago	/v/millennium-park/42b75880f964a52090251fe3	41.882662	-87.623239
4b9511c7f964a520f38d34e3	9.6	Trail	Chicago Lakefront Trail	Lake Michigan Lakefront	60611	Chicago	/v/chicago-lakefront-trail/4b9511c7f964a520f38...	41.967053	-87.646909
49e9ef74f964a52011661fe3	9.6	Art Museum	The Art Institute of Chicago	111 S Michigan Ave	60603	Chicago	/v/the-art-institute-of-chicago/49e9ef74f964a5...	41.879665	-87.623630
4f2a0d0ae4b0837d0c4c2bc3	9.6	Deli / Bodega	Publican Quality Meats	825 W Fulton Market	60607	Chicago	/v/publican-quality-meats/4f2a0d0ae4b0837d0c4c...	41.886642	-87.648718
4aa05f40f964a520643f20e3	9.6	Theater	The Chicago Theatre	175 N State St	60601	Chicago	/v/the-chicago-theatre/4aa05f40f964a520643f20e3	41.885578	-87.627286

Data Analysis and Visualisation

An initial look at the data shows that there are 30 rows of data [as expected] each with 10 attributes. The variable types are all correct except the Venue Rating or Score which will be converted to a float. After converting the score column to a float it can clearly be seen that we have the top venues with a mean of `9.532`.

```
df_top_venues.shape
(30, 10)

df_top_venues.dtypes
id              object
score           object
category        object
name            object
address         object
postalcode      object
city            object
href            object
latitude       float64
```

```
longitude     float64
dtype: object

df_top_venues.score.describe()
count    30.000000
mean      9.523333
std       0.072793
min       9.400000
25%       9.500000
50%       9.500000
75%       9.600000
max       9.700000
Name: score, dtype: float64
```

We are now ready to get the top restaurents within 500 meters of each of the top sites.

FourSquare Restaurent Recommendation Data

Using the the list of all `id` values in the Top Sites DataFrame and the FourSquare `categoryID` that represents all food venues we now search for restaurants within a 500 meter radius.

```
# Configure additional Search parameters
categoryId = '4d4b7105d754a06374d81259'
radius = 500
limit = 15

url =
'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v
={}&categoryId={}&radius={}&limit={}'.format(
    cfg['client_id'],
    cfg['client_secret'],
    ven_lat,
    ven_long,
    cfg['version'],
    categoryId,
    radius,
    limit)

results = requests.get(url).json()
```

The requests returns a JSON object which can then be queried for the restaurant details required. A sample restaurnt from the results returned is shown below:


```
{
    "referralId":"v-1538424503",
    "hasPerk":"False",
    "venuePage":{
        "id":"135548807"
    },
    "id":"55669b9b498ee34e5249ea61",
    "location":{
        "labeledLatLngs":[
            {
```

```
                "label":"display",
                "lng":-87.62460021795313,
                "lat":41.88169538551873
            }
        ],
        "crossStreet":"btwn E Madison & E Monroe St",
        "postalCode":"60603",
        "formattedAddress":[
            "12 S Michigan Ave (btwn E Madison & E Monroe St)",
            "Chicago, IL 60603",
            "United States"
        ],
        "distance":155,
        "city":"Chicago",
        "lng":-87.62460021795313,
        "neighborhood":"The Loop",
        "cc":"US",
        "state":"IL",
        "address":"12 S Michigan Ave",
        "lat":41.88169538551873,
        "country":"United States"
    },
    "name":"Cindy's",
    "categories":[
        {
            "pluralName":"Gastropubs",
            "id":"4bf58dd8d48988d155941735",
            "name":"Gastropub",
            "primary":"True",
            "icon":{
                "prefix":"https://ss3.4sqi.net/img/categories_v2/food/gastropub_",
                "suffix":".png"
            },
            "shortName":"Gastropub"
        }
    ]
},
```

From this JSON the following attributes are extraced and added to the Dataframe:

- Restaurant ID
- Restaurant Category Name
- Restaurant Category ID
- Restaurant Nest_name
- Restaurant Address
- Restaurant Postalcode
- Restaurant City
- Restaurant Latitude
- Restaurant Longitude
- Venue Name

- Venue Latitude
- Venue Longitude

The only piece of data that is missing is the Score or Rating of the Restaurant. To get this we need to make another FourSquare API query using the id of the Restaurant:

```
# Get the restaurant score and href
rest_url =
'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(
    rest_id,
    cfg['client_id'],
    cfg['client_secret'],
    cfg['version'])

result = requests.get(rest_url).json()
rest_score = result['response']['venue']['rating']
```

Using just the data in this DataFrame we will be able to generate maps displaying the chosen Top List Venue and the best scored surrounding restaurants. A sample of this data is shown below:

id	score	category	categoryID	name	address	postal code	city	latitude	longitude	venue_name	venue_latitude	venue_longitude
55669b9b498ee34e5249ea61	9.2	Gastropubs	4bf58dd8d48988d155941735	Cindy's	12 S Michigan Ave	60603	Chicago	41.881695	-87.624600	Millennium Park	41.882662	-87.623239
556509d6498e726bdec19fe9	8.4	Burger Joints	4bf58dd8d48988d16c941735	Shake Shack	12 S Michigan Ave	60603	Chicago	41.881673	-87.624455	Millennium Park	41.882662	-87.623239
49e749fbf964a52086641fe3	9.1	Gastropubs	4bf58dd8d48988d155941735	The Gage	24 S Michigan Ave	60603	Chicago	41.881319	-87.624642	Millennium Park	41.882662	-87.623239
4e879cdc93adfd051d6d609e	9.2	Breakfast Spots	4bf58dd8d48988d143941735	Wildberry Pancakes & Cafe	130 E Randolph St	60601	Chicago	41.884599	-87.623203	Millennium Park	41.882662	-87.623239
49d8159cf96	8.5	Pubs	4bf58dd8d489	Miller's	134 S Wab	60603	Chic	41.87	-87.62	Millennium	41.88	-87.62323

id	score	category	categoryID	name	address	postalcode	city	latitude	longitude	venue_name	venue_latitude	venue_longitude
4a520a05d1fe3			88d11b941735	Pub	ash Ave		ago	9911	5972	Park	2662	9

Looking at the data we get an interesting insight into the range of restuarants that are included. From a list of 30 top venues only 28 actually had more than 10 to provide the user with a real choice. In total there were 387 restaurants found of which 240 were unique occuring only once in the data. There were 72 categories of restaurants. The mean score of all the restaurants wa `8.23` with a manimum value of `9.5` and a minimum value of `5.3`.
Coffee Shops (52) and Pizza Places (29) were the top two most frequently occurring categories but Pie Shops (9.4000) and French Restaurants (9.4000) were the restaurant categories with the highest average score.

```
# What is the shape of the Restaurants DataFrame
df_restaurant.shape
(387, 13)

# Get a count of the top venues that had more than 10 restaurant within 500 meters
#  The number of unique restaurants
#  The number of unique restaurant categories
df_restaurant.venue_name.nunique()
28
df_restaurant.name.nunique()
240
df_restaurant.category.nunique()
72

# Look at the data types
df_restaurant.dtypes
id                    object
score                float64
category              object
categoryID            object
name                  object
address               object
postalcode            object
city                  object
latitude             float64
longitude            float64
venue_name            object
```

```
venue_latitude     float64
venue_longitude    float64
dtype: object

# Describe the Score attribute
df_restaurant.score.describe()
count    387.000000
mean       8.286563
std        0.930138
min        5.300000
25%        7.800000
50%        8.500000
75%        9.000000
max        9.500000
Name: score, dtype: float64

df_restaurant.groupby('category')['name'].count().sort_values(ascending=False)[:10]
category
Coffee Shops                     52
Pizza Places                     29
Cafés                            24
Bakeries                         15
Burger Joints                    15
Gastropubs                       15
New American Restaurants         15
Mexican Restaurants              14
Breakfast Spots                  13
Fast Food Restaurants            13


df_restaurant.groupby('category')['score'].mean().sort_values(ascending=False)[:10]
category
Pie Shops                        9.4000
French Restaurants               9.4000
Molecular Gastronomy Restaurants    9.3000
Filipino Restaurants             9.2000
Cuban Restaurants                9.1000
Ice Cream Shops                  9.0625
Mediterranean Restaurants        9.0600
Korean Restaurants               9.0000
Latin American Restaurants       9.0000
Fish & Chips Shops               9.0000
```

Chicago Crime Data

This dataset can be download from the Chicago Data Portal and reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago in the last year, minus the most recent seven days. A full desription of the data is available on the site.

Data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. In order to protect the privacy of crime victims, addresses are shown at the block level only and specific locations are not identified.

Column Name	**Type**	**Description**
CASE#	Plain Text	The Chicago Police Department RD Number (Records Division Number), which is unique to the incident.
DATE OF OCCURRENCE	Date & Time	Date when the incident occurred. this is sometimes a best estimate.
BLOCK	Plain Text	The partially redacted address where the incident occurred, placing it on the same block as the actual address.
IUCR	Plain Text	The Illinois Unifrom 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 DESCRIPTION	Plain Text	The primary description of the IUCR code.
SECONDARY DESCRIPTION	Plain Text	The secondary description of the IUCR code, a subcategory of the primary description.
LOCATION DESCRIPTION	Plain Text	Description of the location where the incident occurred.
ARREST	Plain Text	Indicates whether an arrest was made.
DOMESTIC	Plain Text	Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act.

Column Name	Type	Description
BEAT	Plain Text	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.
WARD	Number	The ward (City Council district) where the incident occurred. See the wards at https://data.cityofchicago.org/d/sp34-6z76.
FBI CD	Plain Text	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	Plain Text	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	Plain Text	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.
LATITUDE	Number	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	Number	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	Location	The location where the incident occurred in a format that allows for creation of maps and other geographic operations on this data portal. This location is shifted from the actual location for partial redaction

Column Name	Type	Description
		but falls on the same block.

Not all of the attributes are required so on the following data was imported:

- Date of Occurance
- Block
- Primary Description
- Ward
- Latitude
- Longitude

A sample of the imported data is shown.

CASE#	DATE OF OCCURRENCE	BLOCK	PRIMARY DESCRIPTION	WARD	LATITUDE	LONGITUDE
JB241987	04/28/2018 10:05:00 PM	009XX N LONG AVE	NARCOTICS	37.0	41.897895	-87.760744
JB241350	04/28/2018 08:00:00 AM	008XX E 53RD ST	CRIMINAL DAMAGE	5.0	41.798635	-87.604823
JB245397	04/28/2018 09:00:00 AM	062XX S MICHIGAN AVE	THEFT	20.0	41.780946	-87.621995
JB241444	04/28/2018 12:15:00 PM	046XX N ELSTON AVE	THEFT	39.0	41.965404	-87.736202
JB241667	04/28/2018 04:28:00 PM	022XX S KENNETH AVE	ARSON	22.0	41.850673	-87.735597

This data was then processed as follows:

1. Move September 2017 dates to September 2018 The extract of data used was taken mid September which meant that there was half a months data for September 2017 and half a months data for September 2018. These were combined to create a single month.
2. Clean up the column names:
    i. Strip leading & trailing whitespace

    ii. Replace multiple spaces with a single space
    iii. Remove # characters
    iv. Replace spaces with _
    v. Convert to lowercase
3. Change the date of occurrence field to a date / time object
4. Add new columns for:
    i. Hour
    ii. Day
    iii. Month
    iv. Year
    v. etc.
5. Split Block into zip_code and street
6. Verify that all rows have valid data

Data Analysis and Visualisation

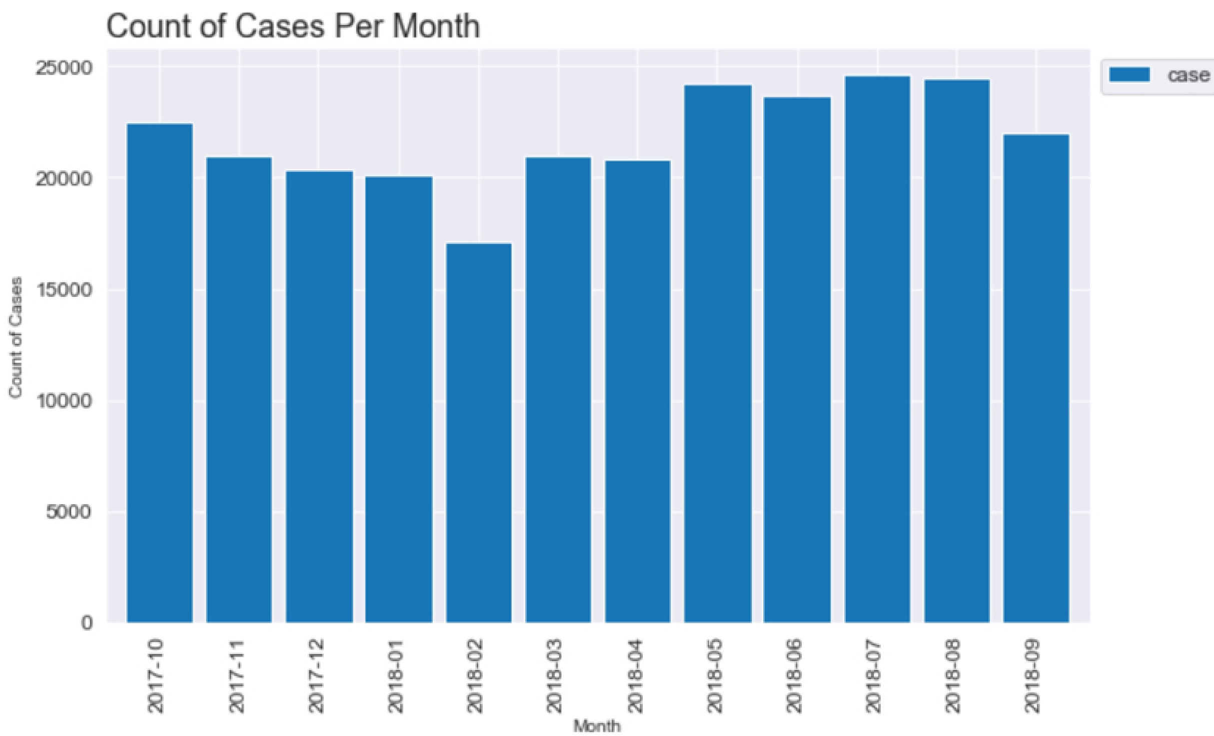
Now let's look at some of the attributes and statistics of the crime dataset.

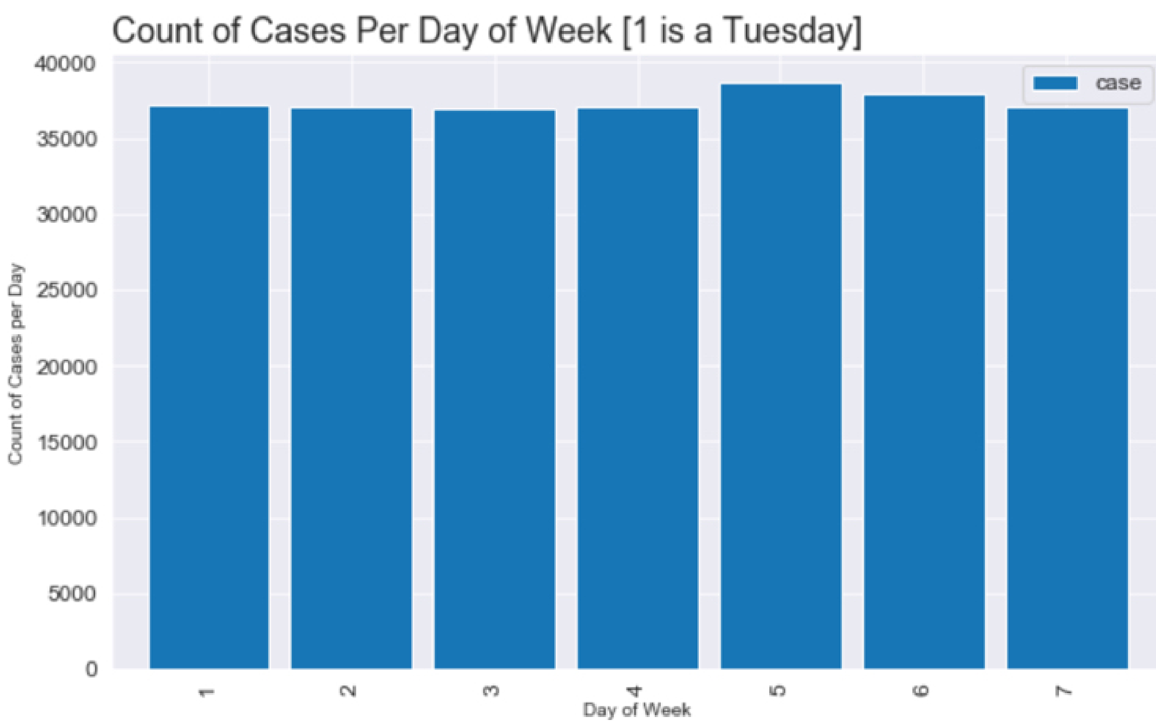
We will start by looking at the top three crimes and a total count for each crime type:

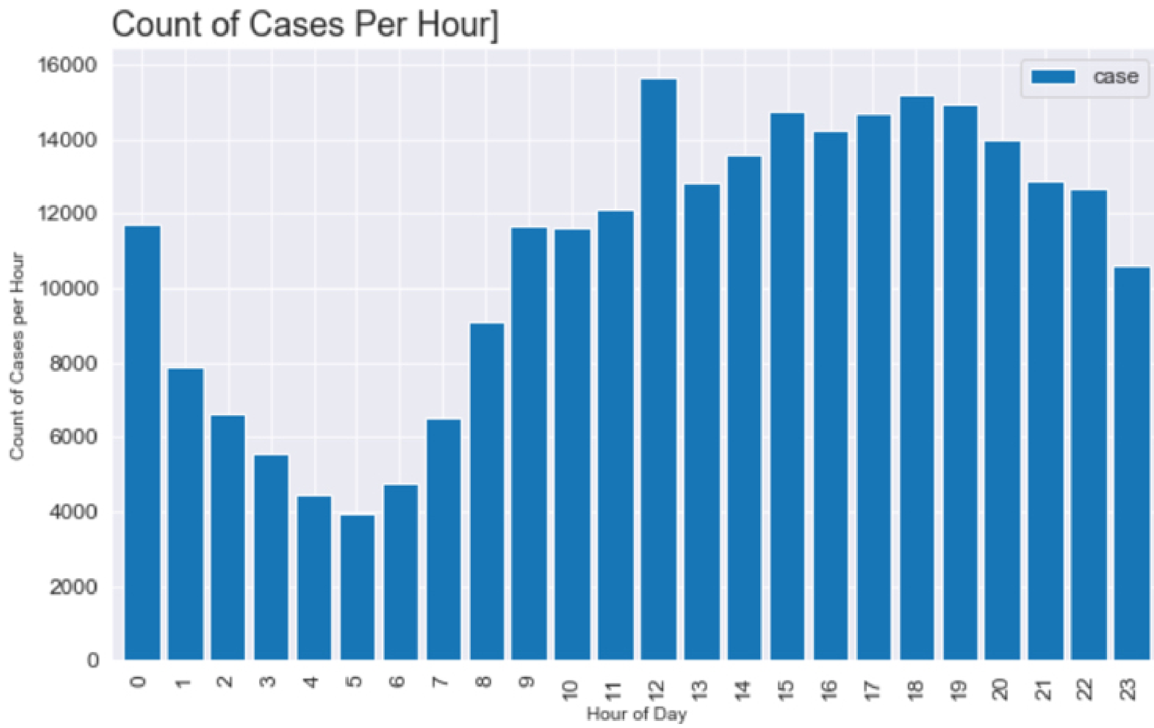
```
# What Crimes are the 3 most commonly occuring ones
df[['primary_description', 'case']].groupby(
    ['primary_description'], as_index=False).count().sort_values(
    'case', ascending=False).head(3)
```

primary_description	case
THEFT	63629
BATTERY	49498
CRIMINAL DAMAGE	27980

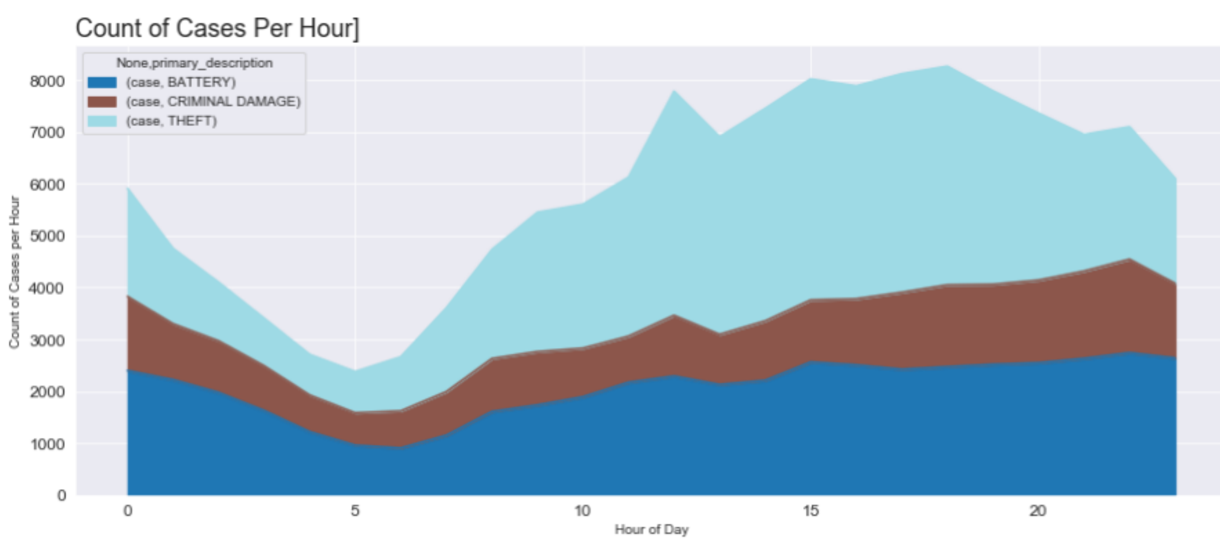
To get a better understanding of the data we will now visualise it. The number of crimes per month, day and hour were calculated:


Count of Cases Per Month
case
Count of Cases
Month
0
5000
10000
15000
20000
25000
2017-10
2017-11
2017-12
2018-01
2018-02
2018-03
2018-04
2018-05
2018-06
2018-07
2018-08
2018-09


Count of Cases Per Day of Week [1 is a Tuesday]
case
Count of Cases per Day
Day of Week
0
5000
10000
15000
20000
25000
30000
35000
40000
1
2
3
4
5
6
7

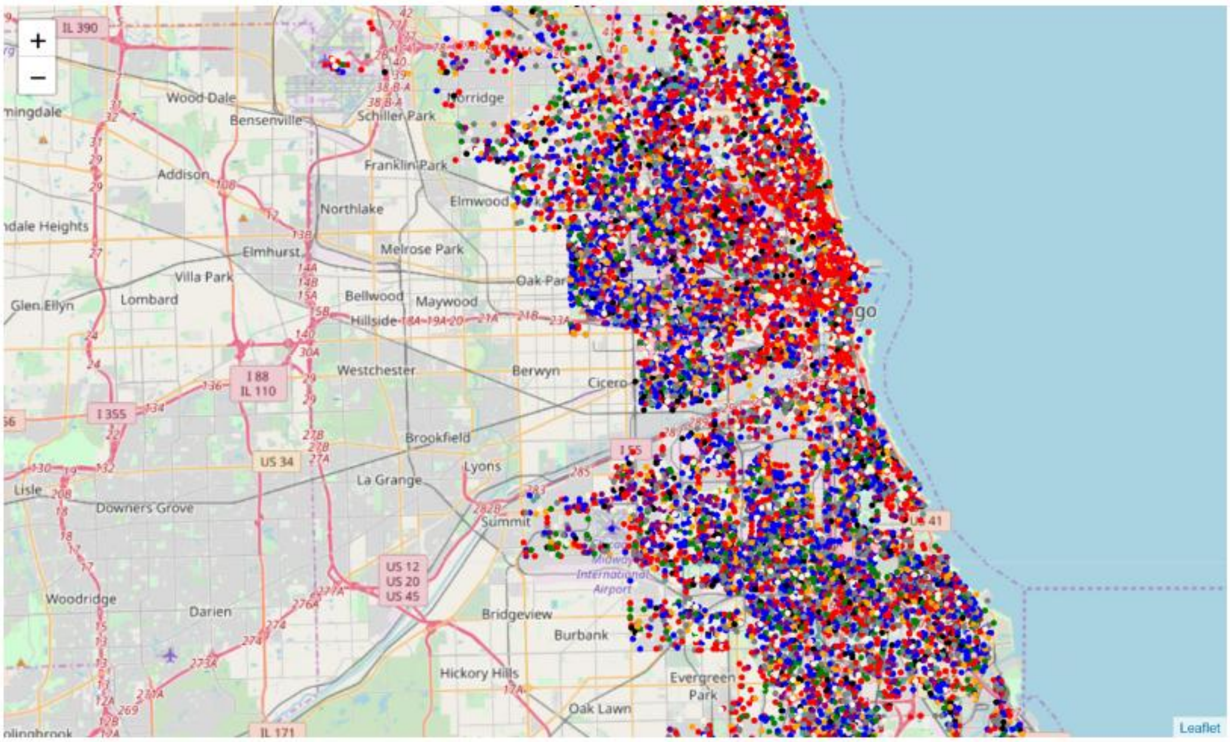


Looking at the top three crimes it is clearly visible that the occurrences of theft rise greatly during daylight hours and particularly between the hours of 3:00 pm and 5:00 pm.



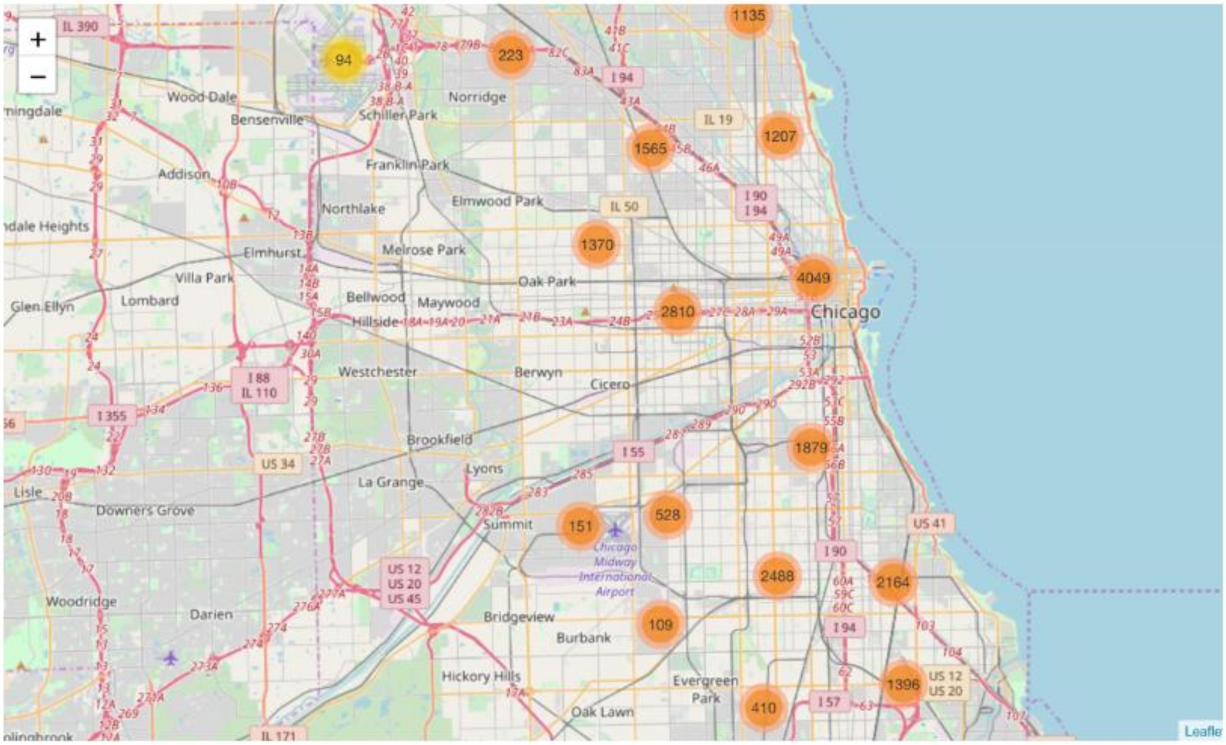
Unsurprisingly there little obvious variation in the number of crimes committed per month other than an apparent drop-off in February. There is a small increase in crime reported at the weekend, Saturday and Sunday, but nothing that could be considered significant. There is an expected fall-off in reported crime rates after midnight and before eight in the morning.

Finally the crimes data for a single month, August, was super-imposed over a map of Chicago to visualise the distribution of that data:



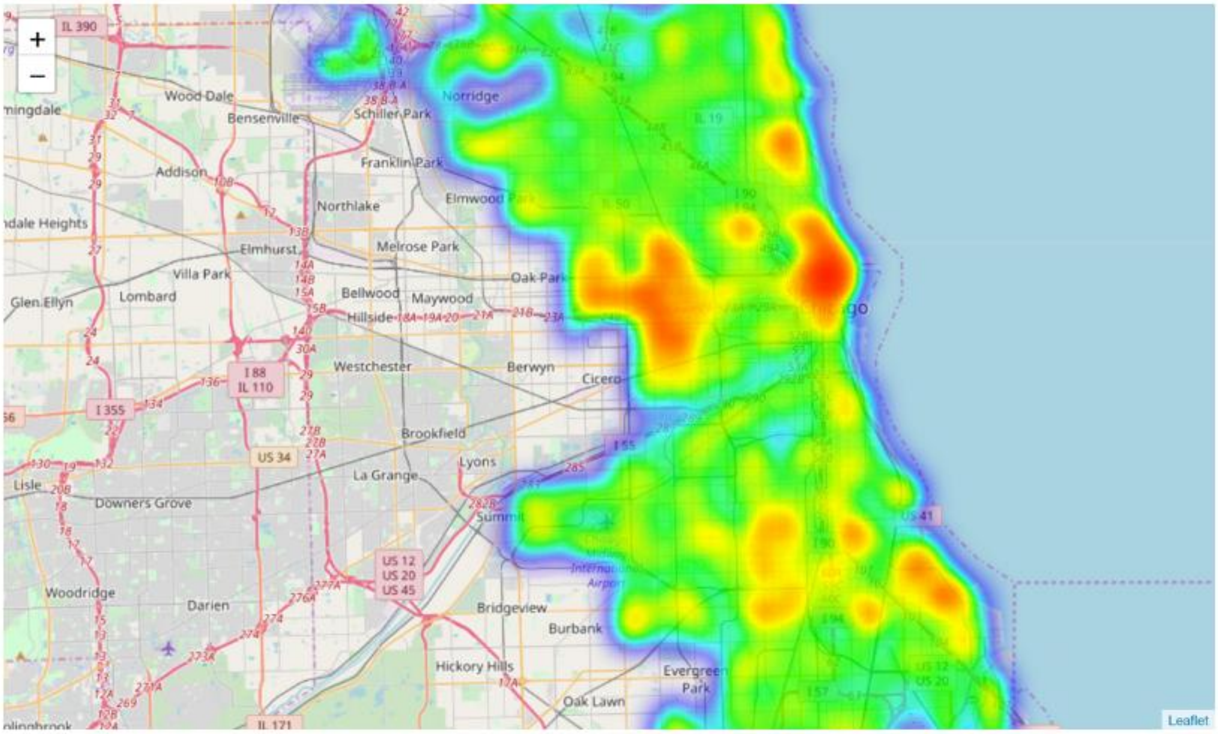
The higher frequency of the top two crimes can be easily seen. Red for Theft and Blue for Battery.

Next the crimes were clustered:



Several obvious clusters of crime locations were visible, particularly in the center of Chicago.

Finally a heat map of the August crimes was created:



This reinforces the cluster chart where it can clearly be seen that the center of Chicago and the area around Oak Park have a high crime rate occurrence. It will be interesting to see later if there is a high probability of crime in these areas if one of the top listed venues are located in these areas.