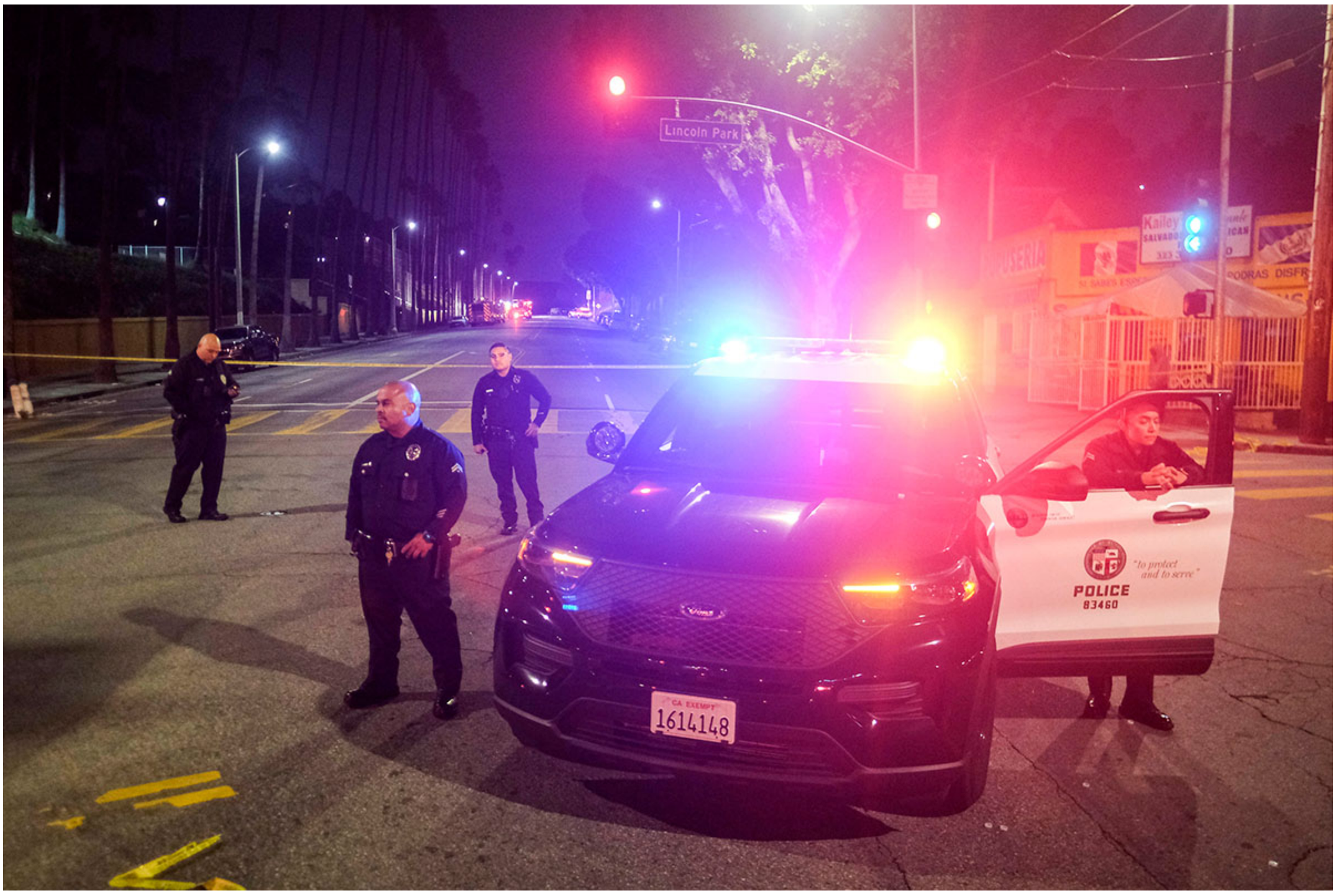


LAPD Crime Data Analysis

Table of Contents

1 Summary

For this analysis I used LAPD api. The dataset reflects incidents of crime in the City of Los Angeles dating back to 2020. The documents I stored in collection crimes in MongoDB. Here I retrieving crime records in batches of 500 records through API.

2 Requirements & Configuration

```
import pymongo
import pprint as pp
import pandas as pd
import requests
import json
import time
import string
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
```

```
pd.set_option('display.precision', 2)
pd.set_option('display.max_rows', 30)
pd.set_option('display.max_colwidth', 25)
```

The data from LAPD api is retrieved. The dataset reflects incidents of crime in the City of Los Angeles dating back to 2020. ([https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8 (https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8)](https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8))

```
API_URL = 'https://data.lacity.org/resource/2nrs-mtv8.json'
CNX_STR = 'mongodb+srv://bahramkhanlarov:OGxVy6YspCC1O32J@bk1820.h3zn6yg.mongodb.net/'
DB_NAME = 'LAPDdb'
COLL_NAME = 'crimes'
```

3 ELT Process

3.1 DB Setup

We will do a setup for each of the collection:

```
client = pymongo.MongoClient(CNX_STR)
db = client[DB_NAME]
crimes = db[COLL_NAME]
```

```
crimes.drop()
crimes.count_documents({})
```

0

```
client.list_database_names()
```

```
['My_Project',
 'Premierleaguedb',
 'db_test',
 'sample_airbnb',
 'sample_analytics',
 'sample_geospatial',
 'sample_guides',
 'sample_mflix',
 'sample_restaurants',
 'sample_supplies',
 'sample_training',
 'sample_weatherdata',
 'test',
 'admin',
 'local']
```

3.2 Extract

```
url = f'{API_URL}'
r = requests.get(url)
crime= r.json()
crime[0]
```


```
{'dr_no': '010304468',
 'date_rptd': '2020-01-08T00:00:00.000',
 'date_occ': '2020-01-08T00:00:00.000',
 'time_occ': '2230',
 'area': '03',
 'area_name': 'Southwest',
 'rpt_dist_no': '0377',
 'part_1_2': '2',
 'crm_cd': '624',
 'crm_cd_desc': 'BATTERY - SIMPLE ASSAULT',
 'mocodes': '0444 0913',
 'vict_age': '36',
 'vict_sex': 'F',
 'vict_descent': 'B',
 'premis_cd': '501',
 'premis_desc': 'SINGLE FAMILY DWELLING',
 'weapon_used_cd': '400',
 'weapon_desc': 'STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)',
 'status': 'AO',
 'status_desc': 'Adult Other',
 'crm_cd_1': '624',
 'location': '1100 W  39TH                         PL',
 'lat': '34.0141',
 'lon': '-118.2978'}
```


Here's a brief description of some of the fields: (https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8 (https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8))

- **dr_no**: The divisional record number, which is a unique identifier for the crime.
- **date_rptd**: The date the crime was reported.
- **date_occ**: The date the crime occurred.
- **time_occ**: The time the crime occurred.
- **area**: The LAPD area where the crime occurred.
- **area_name**: The name of the LAPD area where the crime occurred.
- **rpt_dist_no**: The reporting district number.
- **crm_cd**: The crime code.
- **crm_cd_desc**: A description of the crime code.
- **mocodes**: Modus operandi codes, which describe how the crime was committed.
- **vict_age**: The age of the victim.
- **vict_sex**: The sex of the victim.
- **vict_descent**: The descent of the victim.
- **premis_cd**: The premise code, which describes the type of location where the crime occurred.
- **premis_desc**: A description of the premise code.
- **weapon_used_cd**: The weapon used code, which describes the type of weapon used in the crime (if any).
- **weapon_desc**: A description of the weapon used code.
- **status**: The status of the case.
- **status_desc**: A description of the status of the case.
- **crm_cd_1**: The primary crime code (if multiple crimes were committed).
- **location**: The location where the crime occurred.
- **lat and lon**: The latitude and longitude coordinates of where the crime occurred.

```python
#function to extract values from a single crime
def extract_crime_values(crime):
    keys = ['dr_no', 'date_rptd', 'date_occ', 'time_occ', 'area', 'area_name', 'rpt_dist_no', 'crm_cd',
            'crm_cd_desc', 'mocodes', 'vict_age', 'vict_sex', 'vict_descent', 'premis_cd', 'premis_desc',
            'weapon_used_cd', 'weapon_desc', 'status', 'status_desc', 'crm_cd_1', 'location', 'lat', 'lon']

    values = {}
    for key in keys:
        value = crime.get(key)
        if value is not None:
            values[key] = value

    return values
```

```python
# retrieving crime records in batches of 500 records through API
# extract the relevant fields from each crime record, appending to the list_crimes

import requests
import pprint

list_crimes = []

def retrieve_records(num_records, list_crimes):
    API_URL = 'https://data.lacity.org/resource/2nrs-mtv8.json'
    limit = 500
    offset = 0
    while len(list_crimes) < num_records:
        url = f'{API_URL}?$limit={limit}&$offset={offset}'
        r = requests.get(url)
        crimes = r.json()
        for crime in crimes:
            values = extract_crime_values(crime)
            list_crimes.append(values)
        offset += limit


retrieve_records(500, list_crimes)
pprint.pprint(list_crimes[0])
```

```
{'area': '03',
 'area_name': 'Southwest',
 'crm_cd': '624',
 'crm_cd_1': '624',
 'crm_cd_desc': 'BATTERY - SIMPLE ASSAULT',
 'date_occ': '2020-01-08T00:00:00.000',
 'date_rptd': '2020-01-08T00:00:00.000',
 'dr_no': '010304468',
 'lat': '34.0141',
 'location': '1100 W  39TH                         PL',
 'lon': '-118.2978',
 'mocodes': '0444 0913',
 'premis_cd': '501',
 'premis_desc': 'SINGLE FAMILY DWELLING',
 'rpt_dist_no': '0377',
 'status': 'AO',
 'status_desc': 'Adult Other',
 'time_occ': '2230',
 'vict_age': '36',
 'vict_descent': 'B',
 'vict_sex': 'F',
 'weapon_desc': 'STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)',
 'weapon_used_cd': '400'}
```

3.3 Load

Insert multiple documents in the MongodB collection named 'crimes'

```python
#Load
crimes.insert_many(list_crimes);
```

```python
# count number of documents inserted
print(crimes.count_documents({}))
```

```
500
```

```python
collection_names = db.list_collection_names()
num_collections = len(collection_names)

print(f'The database has {num_collections} collections.')
```

```
The database has 1 collections.
```

```python
# get one crime from MongoDB

from pprint import pprint
pprint(crimes.find_one())
```

```
{'_id': ObjectId('6461dd769e0bfb78fd664f7b'),
 'area': '03',
 'area_name': 'Southwest',
 'crm_cd': '624',
 'crm_cd_1': '624',
 'crm_cd_desc': 'BATTERY - SIMPLE ASSAULT',
 'date_occ': '2020-01-08T00:00:00.000',
 'date_rptd': '2020-01-08T00:00:00.000',
 'dr_no': '010304468',
 'lat': '34.0141',
 'location': '1100 W  39TH                         PL',
 'lon': '-118.2978',
 'mocodes': '0444 0913',
 'premis_cd': '501',
 'premis_desc': 'SINGLE FAMILY DWELLING',
 'rpt_dist_no': '0377',
 'status': 'AO',
 'status_desc': 'Adult Other',
 'time_occ': '2230',
 'vict_age': '36',
 'vict_descent': 'B',
 'vict_sex': 'F',
 'weapon_desc': 'STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)',
 'weapon_used_cd': '400'}
```

```python
# get 5 crimes from MongoDB and display as dataframe
r_crimes = crimes.aggregate([
      {'$limit': 5},
    {'$project': {'_id': 0}},
])
pd.DataFrame(r_crimes)
```

	dr_no	date_rptd	date_occ	time_occ	area	area_name	rpt_dist_no	crm_cd	crm_cd_desc	mocodes	...	premis_(
0	010304468	2020-01-08T00:00:00.000	2020-01-08T00:00:00.000	2230	03	Southwest	0377	624	BATTERY - SIMPLE ASSAULT	0444 0913	...	5(
1	190101086	2020-01-02T00:00:00.000	2020-01-01T00:00:00.000	0330	01	Central	0163	624	BATTERY - SIMPLE ASSAULT	0416 1822 1414	...	1(
2	200110444	2020-04-14T00:00:00.000	2020-02-13T00:00:00.000	1200	01	Central	0155	845	SEX OFFENDER REGISTRA...	1501	...	7:
3	191501505	2020-01-01T00:00:00.000	2020-01-01T00:00:00.000	1730	15	N Hollywood	1543	745	VANDALISM - MISDEAMEA...	0329 1402	...	5(
4	191921269	2020-01-01T00:00:00.000	2020-01-01T00:00:00.000	0415	19	Mission	1998	740	VANDALISM - FELONY ($...	0329	...	4(

5 rows × 23 columns

3.4 Transform

We do some quality checks:

```
# checking the data type of all fields in a MongoDB collection
pipeline = [
    {
        "$project": {
            "field_types": {
                "$map": {
                    "input": {"$objectToArray": "$$ROOT"},
                    "as": "field",
                    "in": {
                        "field_name": "$$field.k",
                        "field_type": {"$type": "$$field.v"}
                    }
                }
            }
        }
    }
]



doc = db.crimes.find_one()

field_types = [
    {
        "field_name": field_name,
        "field_type": type(field_value).__name__
    }
    for field_name, field_value in doc.items()
]

pprint(field_types)
```

```
[{'field_name': '_id', 'field_type': 'ObjectId'},
 {'field_name': 'dr_no', 'field_type': 'str'},
 {'field_name': 'date_rptd', 'field_type': 'str'},
 {'field_name': 'date_occ', 'field_type': 'str'},
 {'field_name': 'time_occ', 'field_type': 'str'},
 {'field_name': 'area', 'field_type': 'str'},
 {'field_name': 'area_name', 'field_type': 'str'},
 {'field_name': 'rpt_dist_no', 'field_type': 'str'},
 {'field_name': 'crm_cd', 'field_type': 'str'},
 {'field_name': 'crm_cd_desc', 'field_type': 'str'},
 {'field_name': 'mocodes', 'field_type': 'str'},
 {'field_name': 'vict_age', 'field_type': 'str'},
 {'field_name': 'vict_sex', 'field_type': 'str'},
 {'field_name': 'vict_descent', 'field_type': 'str'},
 {'field_name': 'premis_cd', 'field_type': 'str'},
 {'field_name': 'premis_desc', 'field_type': 'str'},
 {'field_name': 'weapon_used_cd', 'field_type': 'str'},
 {'field_name': 'weapon_desc', 'field_type': 'str'},
 {'field_name': 'status', 'field_type': 'str'},
 {'field_name': 'status_desc', 'field_type': 'str'},
 {'field_name': 'crm_cd_1', 'field_type': 'str'},
 {'field_name': 'location', 'field_type': 'str'},
 {'field_name': 'lat', 'field_type': 'str'},
 {'field_name': 'lon', 'field_type': 'str'}]
```

```
pipeline = [
    {
        "$project": {
            "vict_sex": { "$cond": { "if": { "$ne": [ "$vict_sex", None ] }, "then": 0, "else": 1 } },
            "premis_desc": { "$cond": { "if": { "$ne": [ "$premis_desc", None ] }, "then": 0, "else": 1 } },
            "crm_cd_desc": { "$cond": { "if": { "$ne": [ "$crm_cd_desc", None ] }, "then": 0, "else": 1 } },
            "weapon_desc": { "$cond": { "if": { "$ne": [ "$weapon_desc", None ] }, "then": 0, "else": 1 } },
            "location": { "$cond": { "if": { "$ne": [ "$location", None ] }, "then": 0, "else": 1 } }
        }
    },
    {
        "$group": {
            "_id": "null",
            "vict_sex": { "$sum": "$vict_sex" },
            "premis_desc": { "$sum": "$premis_desc" },
            "crm_cd_desc": { "$sum": "$crm_cd_desc" },
            "weapon_desc": { "$sum": "$weapon_desc" },
            "location": { "$sum": "$location" }
        }
    }
]
result = list(crimes.aggregate(pipeline))[0]
df = pd.DataFrame.from_dict(result, orient='index', columns=['Missing Values'])
df.index.name = 'Field'
print(df)
```

```
            Missing Values
Field
_id                   null
vict_sex                 0
premis_desc              0
crm_cd_desc              0
weapon_desc              0
location                 0
```

```
# counting the number of unique entries for each attribute/column in collection

pipeline = [
    {
        "$project": {
            "attributes": {
                "$objectToArray": "$$ROOT"
            }
        }
    },
    {
        "$unwind": "$attributes"
    },
    {
        "$group": {
            "_id": "$attributes.k",
            "unique_entries": {
                "$addToSet": "$attributes.v"
            }
        }
    },
    {
        "$project": {
            "_id": 0,
            "Attribute": "$_id",
            "uniqueEntries": {
                "$size": "$unique_entries"
            }
        }
    },
    {
        "$sort": {
            "Attribute": 1
        }
    }
]

result = list(crimes.aggregate(pipeline))
df = pd.DataFrame(result)
print(df)
```

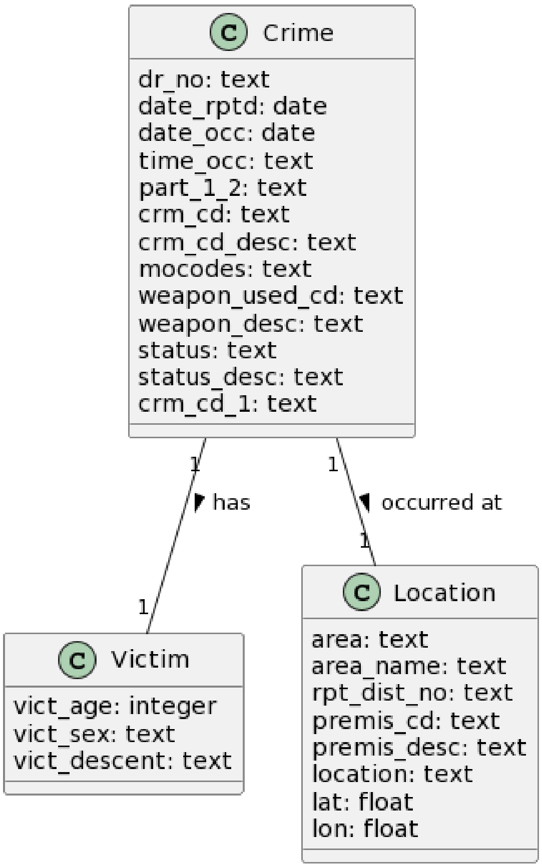
```
         Attribute  uniqueEntries
0              _id            500
1             area             21
2        area_name             21
3           crm_cd             46
4         crm_cd_1             47
5      crm_cd_desc             46
6         date_occ            109
7        date_rptd            124
8            dr_no            500
9              lat            230
10        location            316
11             lon            265
12         mocodes            394
13       premis_cd             65
14     premis_desc             65
15     rpt_dist_no            149
16          status              4
17     status_desc              4
18        time_occ            196
19        vict_age             65
20    vict_descent              9
21        vict_sex              3
22     weapon_desc             24
23  weapon_used_cd             24
```

3.5 Datastructure

This diagram defines three classes, Crime, Victim, and Location, with the attributes specified. It also shows the relationships between the classes using arrows and labels. For example, a Crime has one Victim and occurred at one Location.

The relationship between Crime and Victim is a one-to-one relationship, meaning that each instance of the Crime class is associated with one and only one instance of the Victim class. The arrow points from Crime to Victim, indicating the direction of the relationship. The label “has” describes the nature of the relationship: a Crime has a Victim.

The relationship between Crime and Location is also a one-to-one relationship, meaning that each instance of the Crime class is associated with one and only one instance of the Location class. The arrow points from Crime to Location, indicating the direction of the relationship. The label “occurred at” describes the nature of the relationship: a Crime occurred at a Location.



4 Data Analysis

4.1 The most common types of crimes reported

The output of the following code shows an overview of the 10 common types of crimes, sorted by count numbers

```python
pipeline = [
    {
        '$group': {
            '_id': '$crm_cd_desc',
            'count': {'$sum': 1}
        }
    },
    {
        '$sort': {'count': -1}
    },
    {
        '$limit': 10
    },
    {
        '$project': {
            '_id': 0,
            'crime_type': '$_id',
            'count': 1
        }
    }
]

result = list(db.crimes.aggregate(pipeline))
df = pd.DataFrame(result)
print(df)
```

```
   count                crime_type
0     66      BURGLARY FROM VEHICLE
1     61   BATTERY - SIMPLE ASSAULT
2     38   THEFT PLAIN - PETTY (...
3     29   VANDALISM - FELONY ($...
4     26   SHOPLIFTING - PETTY T...
5     25          THEFT OF IDENTITY
6     24   THEFT-GRAND ($950.01 ...
7     23   ASSAULT WITH DEADLY W...
8     23            VEHICLE - STOLEN
9     17   INTIMATE PARTNER - SI...
```

4.2 Top 15 most common weapons used in crimes

307 cases in the dataset where the weapon used in the crime was not known or not recorded, Strong arm(hands,fist,feet or bodily force) comes on the top of the list

```python
pipeline = [
    {"$group": {"_id": "$weapon_desc", "count": {"$sum": 1}}},
    {"$sort": {"count": -1}},
    {"$limit": 15}
]

results = db.crimes.aggregate(pipeline)

top_weapons = {}

for result in results:
    top_weapons[result['_id']] = result['count']

df = pd.DataFrame.from_dict(top_weapons, orient='index', columns=['count'])
df.index.name = 'weapon'
print(df)
```

```
                           count
weapon
None                         307
STRONG-ARM (HANDS, FIS...    106
UNKNOWN WEAPON/OTHER W...     35
HAND GUN                       8
VERBAL THREAT                  8
KNIFE WITH BLADE 6INCH...      4
OTHER KNIFE                    3
SCISSORS                       3
SEMI-AUTOMATIC PISTOL          3
FOLDING KNIFE                  2
MACHETE                        2
BLUNT INSTRUMENT               2
BOTTLE                         2
ROCK/THROWN OBJECT             2
OTHER CUTTING INSTRUMENT       2
```

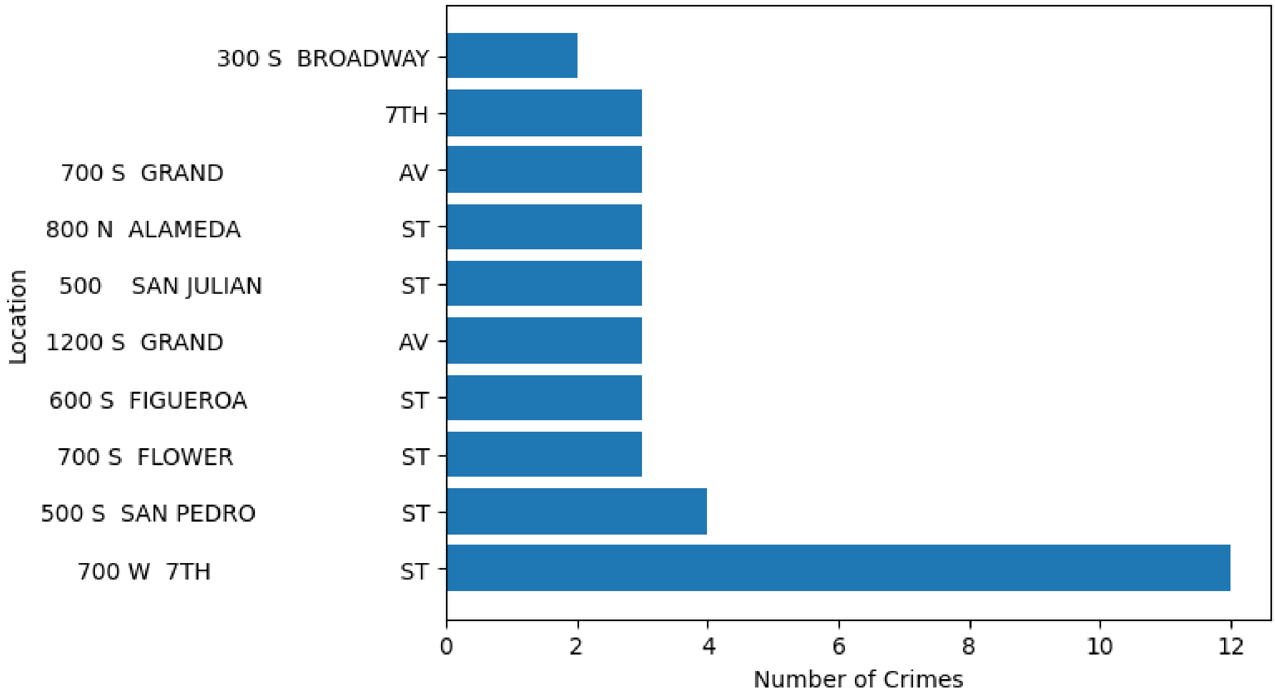
4.3 Location and crime types occurance

The result shows the top 10 crimes in Los Angeles, along with the location and the number of times they occurred. The result is sorted in descending order based on the number of occurrences of each crime type and is limited to the top 10. For example, the most common crime in the dataset is "SHOPLIFTING - PETTY THEFT ($950 & UNDER)" which occurred 12 times in the location "700 W 7TH". The second most common crime is "BATTERY - SIMPLE ASSAULT" which occurred four times in the location "500 S SAN PEDRO".

```python
pipeline = [
    {"$group": {"_id": {"crime_type": "$crm_cd_desc", "location": "$location"}, "count": {"$sum": 1}}},
    {"$sort": {"count": -1}},
    {"$limit": 10},
    {"$project": {"_id": 0, "Crime Type": "$_id.crime_type", "Location": "$_id.location", "Count": "$count"}}
]

results = list(db.crimes.aggregate(pipeline))
df = pd.DataFrame(results)
print(df)
```

```
              Crime Type                  Location  Count
0  SHOPLIFTING - PETTY T...  700 W  7TH          ...     12
1  BATTERY - SIMPLE ASSAULT  500 S  SAN PEDRO    ...      4
2  SHOPLIFTING - PETTY T...  700 S  FLOWER       ...      3
3     BURGLARY FROM VEHICLE  1200 S  GRAND       ...      3
4                   ROBBERY                     7TH      3
5     BURGLARY FROM VEHICLE  700 S  GRAND        ...      3
6  BATTERY - SIMPLE ASSAULT  500    SAN JULIAN   ...      3
7  BATTERY - SIMPLE ASSAULT  600 S  FIGUEROA     ...      3
8                TRESPASSING  800 N  ALAMEDA      ...      3
9  BATTERY - SIMPLE ASSAULT  800 N  ALAMEDA      ...      2
```

```
import matplotlib.pyplot as plt

pipeline = [
    {"$group": {"_id": {"crime_type": "$crm_cd_desc", "location": "$location"}, "count": {"$sum": 1}}},
    {"$sort": {"count": -1}},
    {"$limit": 10},
    {"$project": {"_id": 0, "Crime Type": "$_id.crime_type", "Location": "$_id.location", "Count": "$count"}}
]

results = list(db.crimes.aggregate(pipeline))
df = pd.DataFrame(results)

plt.barh(df['Location'], df['Count'])
plt.xlabel('Number of Crimes')
plt.ylabel('Location')
plt.show()
```



4.4 Number of victims per Descent

I use here sortByCount stage to group the documents by vict_descent and count the number of documents in each group.

The results show the count of crimes reported based on the victim's descent. The abbreviations represent different racial or ethnic categories:

- H: Hispanic or Latino
- W: White
- B: Black or African American
- X: Unknown
- O: Other
- None: Descent not provided or recorded
- A: Asian
- C: Chinese
- K: Korean
- F: Filipino

The results suggest that crimes involving Hispanic or Latino victims are the most frequently reported, followed by crimes involving White and Black or African American victims.

```
pipeline = [{'$sortByCount': '$vict_descent'},{'$limit': 10}]

result = db.crimes.aggregate(pipeline)
for doc in result:
    print(doc['_id'], doc['count'])
```

```
H 144
W 108
B 106
X 47
O 43
None 25
A 20
C 3
K 3
F 1
```

4.5 Victims per gender

The first line is counting the number of documents where the value of the vict_sex field is equal to "F", and where the field exists and is not equal to None. This means it's counting the number of crimes where the victim's gender is female. The second line is similar, but it's counting the number of documents where the vict_sex field is equal to "M", meaning it's counting the number of crimes where the victim's gender is male.

```
crimes.count_documents({"vict_sex": "F", "vict_sex": {"$exists": True, "$ne": None}})
crimes.count_documents({"vict_sex": "M", "vict_sex": {"$exists": True, "$ne": None}})
```

```
475
```

```
crimes.count_documents({"vict_sex": "M"})
```

```
260
```

```
crimes.count_documents({"vict_sex": "F"})
```

```
170
```

The query criteria is defined as an $or operator that specifies two conditions:"vict_sex": "F","vict_sex": "M"

```
crimes.count_documents({"$or": [{"vict_sex": "F"}, {"vict_sex": "M"}]})
```

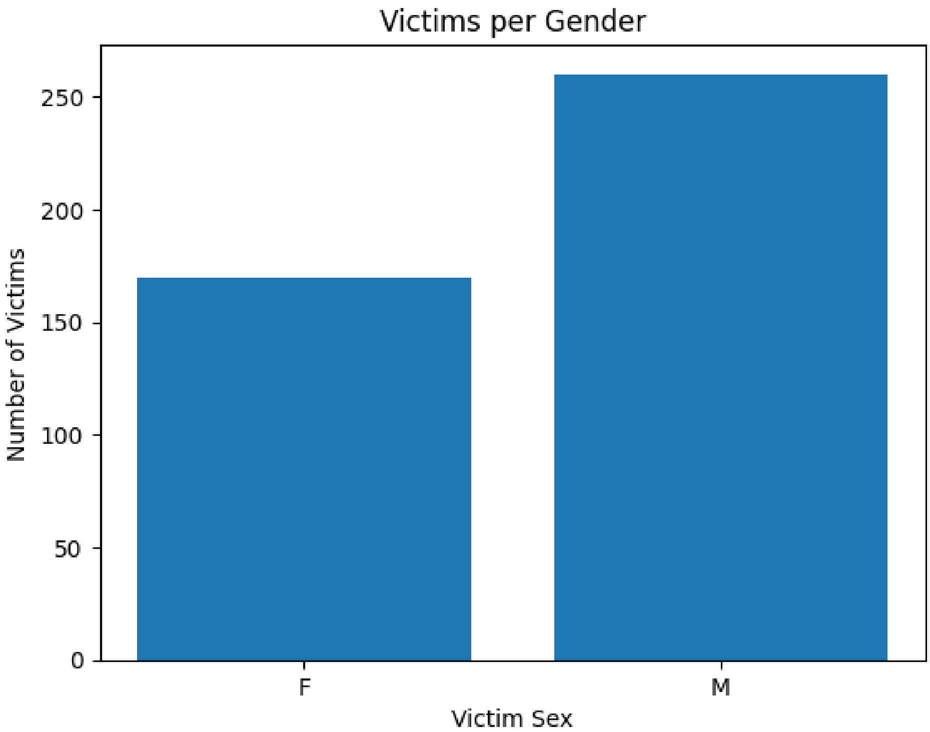
```
430
```

M has a count of 260 which indicates the number of occurrences where vict_sex is equal to "M". F has a count of 169 which indicates the number of occurrences where vict_sex is equal to "F". None has a count of 29 which indicates the number of occurrences where vict_sex is not defined or is null. X has a count of 42 which indicates the number of occurrences where vict_sex has a value other than "M", "F", or null.

```
#Want to check distribution of gender
pipeline = [{'$group':
             {'_id': '$vict_sex',
              'count': {'$sum': 1}}}]

results = crimes.aggregate(pipeline)

for result in results:
    print(result['_id'], result['count'])
```

```
X 45
M 260
None 25
F 170
```

```python
import matplotlib.pyplot as plt

# define the pipeline to group the crimes by victim sex
pipeline = [
    {
        '$match': {
            'vict_sex': {'$in': ['F', 'M']}
        }
    },
    {
        '$group': {
            '_id': '$vict_sex',
            'count': {'$sum': 1}
        }
    }
]

# execute the pipeline and create a bar chart
results = crimes.aggregate(pipeline)
x = []
y = []
for result in results:
    x.append(result['_id'])
    y.append(result['count'])
plt.bar(x, y)
plt.xlabel('Victim Sex')
plt.ylabel('Number of Victims')
plt.title('Victims per Gender')
plt.show()
```



4.6 The time of the day the most crimies occur

According to the dataset, the time at which the most crimes occurred is at 12:00 (noon), and the number of crimes that occurred at that time is 20. It could suggest that there might be some pattern or reason for this, such as more people being out and about during lunchtime or a higher number of businesses being open during that time, among other factors. However, this is just speculation and would require further investigation and analysis to draw any meaningful conclusions.

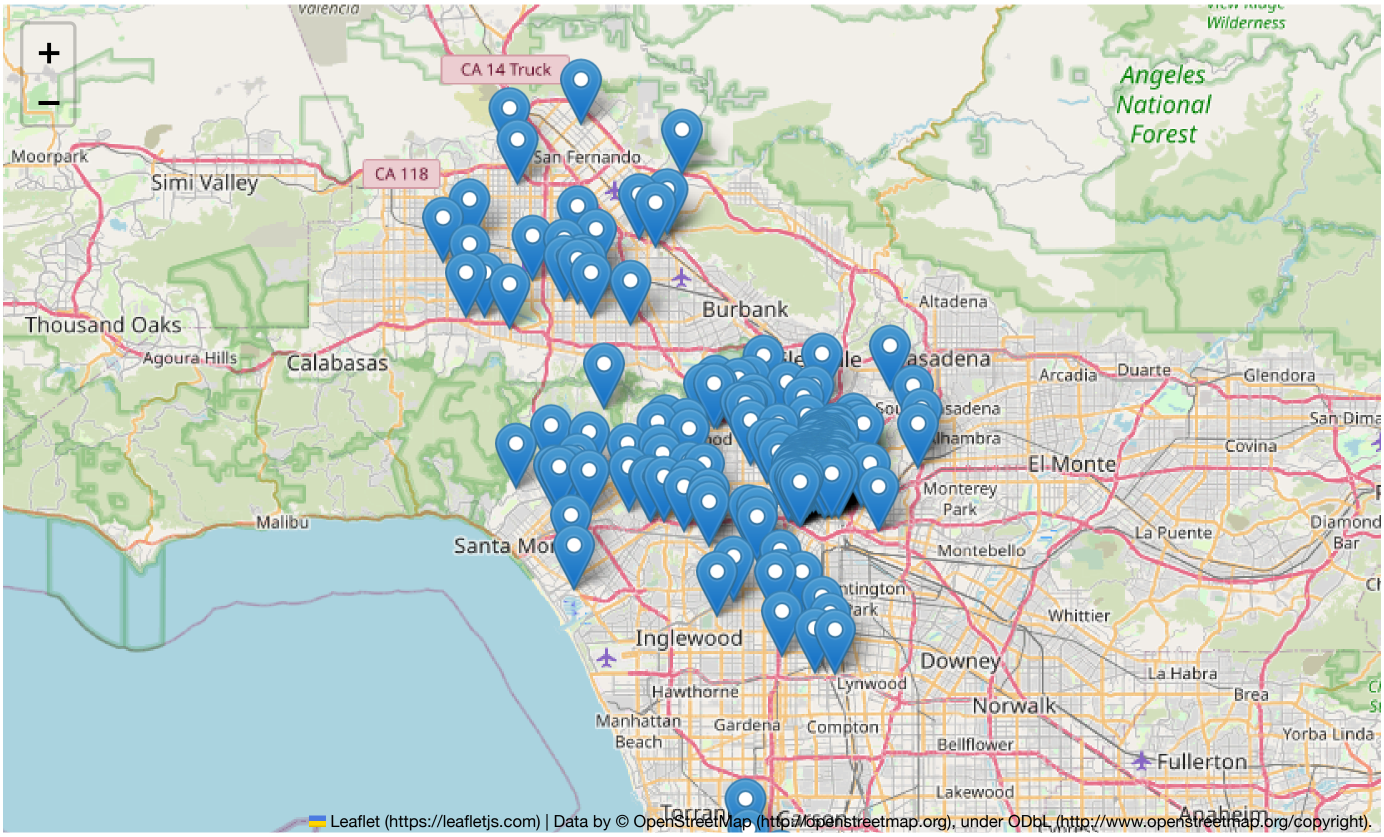
```python
pipeline = [
    {
        "$group": {
            "_id": "$time_occ",
            "count": { "$sum": 1 }
        }
    },
    {
        "$sort": { "count": -1 }
    },
    {
        "$limit": 1
    }
]

result = list(crimes.aggregate(pipeline))[0]
print("The most common time for crimes is {}:00, with {} crimes.".format(result['_id'][:2], result['count']))
```

```
The most common time for crimes is 12:00, with 20 crimes.
```

4.7 Heat map with the concentration of crime incidents based on their geographic locations

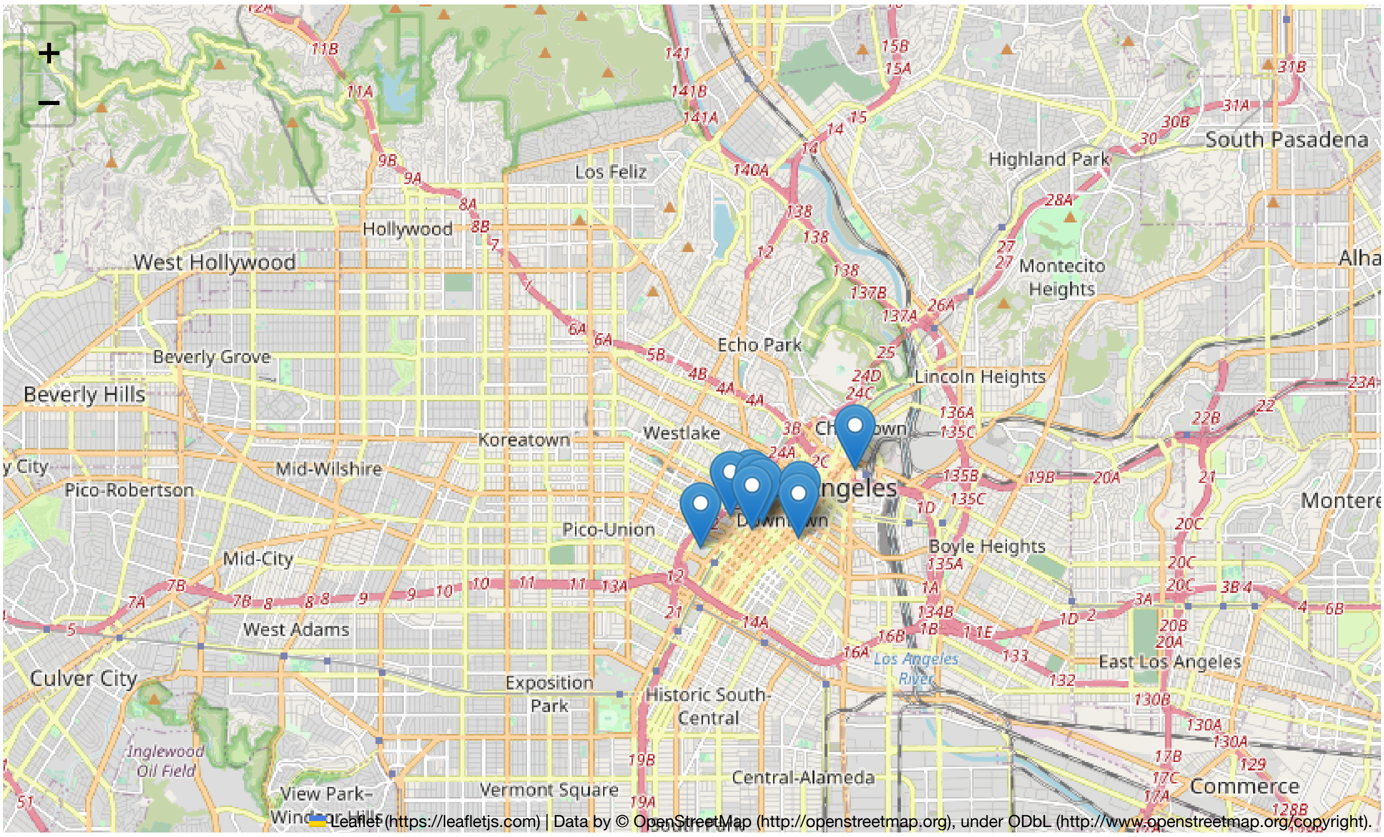
This code creates a map of Los Angeles using the Folium library and displays crime incidents as markers on the map. The pipeline used groups the crimes by their location and counts the number of incidents at each location. The resulting count for each location is used to set the popup of the marker placed at that location. This allows us to visually see areas with high concentration of crimes as locations with more markers. Based on the heatmap, we can see that areas in the central and eastern parts of Los Angeles, such as Downtown LA and Boyle Heights, have higher density of crime incidents compared to other areas of the city. The areas in the west, such as Santa Monica and Beverly Hills, have relatively lower density of crime incidents. However, it's important to note that this is just a general observation based on the data and visualizations, and there may be variations in crime patterns across different neighborhoods and over time.

```python
#!pip install folium
import folium

# set up the map
LA_COORDINATES = (34.0522, -118.2437)
crime_map = folium.Map(location=LA_COORDINATES, zoom_start=10)

# define the pipeline to group the crimes by their location
pipeline = [
    {
        '$group': {
            '_id': {'lat': '$lat', 'lon': '$lon'},
            'count': {'$sum': 1}
        }
    }
]

# execute the pipeline and add markers to the map
results = crimes.aggregate(pipeline)
for result in results:
    lat, lon = result['_id']['lat'], result['_id']['lon']
    count = result['count']
    folium.Marker(location=[lat, lon], popup=f'Crime count: {count}').add_to(crime_map)

# display the map
crime_map
```



We filter out to show the crime points with more than 5 incidents

```python
import folium

# set up the map
LA_COORDINATES = (34.0522, -118.2437)
crime_map = folium.Map(location=LA_COORDINATES, zoom_start=12)

# define the pipeline to group the crimes by their location
pipeline = [
    {
        '$group': {
            '_id': {'lat': '$lat', 'lon': '$lon'},
            'count': {'$sum': 1}
        }
    },
    {
        '$match': {
            'count': {'$gt': 5}
        }
    }
]

# execute the pipeline and add markers to the map
results = crimes.aggregate(pipeline)
for result in results:
    lat, lon = result['_id']['lat'], result['_id']['lon']
    count = result['count']
    folium.Marker(location=[lat, lon], popup=f'Crime count: {count}').add_to(crime_map)

# display the map
crime_map
```



5 Conclusions

In the scope of this project, the analysis is based on retrieved crime records in batches of 500 records through API from Los Angeles Police Department.Therefore, the result and generalisations to the population should be handled carefully due to small size of data subset. I have been only once in LA as tourist and the outcome of the analysis helped me to look at city from different perspective. The findings of this study can have practical implications for different stakeholders:

Los Angeles Police Department:

- They will be able to manage the workforce efficiently by knowing which time of the day and district of the city most of crimes are happening
- What to expect in terms of the crimes- common type of weapon used committing crimes

Tourism Department, Hotels, University:

- Crime Density map will visually show the crime hotspots in the city of Los Angeles which could be avoided by new expats, students, tourists who has no clue about city and who has never been before here
- What time of the day is safer for their activities around the city?

6 Learnings

Firstly, I take opportunity to thank professor for his well structured course content, plenty of handful examples and for his fun and enjoyable teaching style. Even though I am coming from social sciences and it takes me usually more time to grasp the technical knowledge about new technologies, this time with he examples provided by the lecturer and the resources from Datacamp were significantly useful to fully understand Mongodb and how to work with aggregation pipelines, collections, documents. Additionally, fetching data through API, working in jupyter notebook and make presentable pdf at the end, added new skillset for me thanks to this course.

```
%%HTML
<style>
/* display:none  -> hide In/Out column */
/* display:block -> show In/Out column */
.code_cell .run_this_cell {
    display: none;
}
div.prompt {display:none}
</style>
```