

# Shooting Project

Name: Preeyanuch Suphasueb

student ID: 61070306

Adviser: Teerapong Leelanupab

```
In [1]: import json
import urllib
import urllib.request
from pandas.io.json import json_normalize
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install folium
import folium
```

```
Requirement already satisfied: folium in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (0.11.0)
Requirement already satisfied: branca>=0.3.0 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from folium) (0.4.1)
Requirement already satisfied: numpy in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from folium) (1.18.5)
Requirement already satisfied: requests in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from folium) (2.24.0)
Requirement already satisfied: jinja2>=2.9 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from folium) (2.11.2)
Requirement already satisfied: idna<3,>=2.5 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from requests->folium) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from requests->folium) (2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from requests->folium) (1.25.9)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\piyapat wongsawat\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (1.1.1)
```

## Choose Open Web API

ใน assignment นี้ได้เลือกมาข้อมูลจาก <https://opendata.cityofnewyork.us/> ที่เป็นแหล่ง Web API ซึ่งให้ข้อมูลประวัติเหตุการณ์การยิงกันที่เกิดขึ้นภายใน Newyork City ตั้งแต่ปีค.ศ. 2006 จนถึงปีค.ศ. 2018 (NYPD Shooting Incident Data (Historic)-<https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8>) ข้อมูลที่ได้มาเป็นรูปแบบข้อความของเหตุการณ์การยิงกันที่เกิดขึ้นในเมืองนิวยอร์กอย่างล้วนๆ ปีค.ศ. 2006 จนถึงปีค.ศ. 2018 ข้อมูลนี้จะถูกตีบอกราคาทุกไตรมาส และได้รับการตรวจสอบโดยสำนักงานเคราะห์ และวางแผนการจัดการก่อนที่จะโพสต์บนเว็บไซต์ NYPD(New York City Police Department) บันทึกแต่ละรายการ แสดงถึงเหตุการณ์การยิงกันที่นิวยอร์ก และรวมถึงข้อมูลเกี่ยวกับเหตุการณ์สถานที่และเวลาที่เกิดขึ้น นอกจากนี้ยังรวมข้อมูลที่เกี่ยวข้องกับข้อมูลประชากรผู้ต้องสงสัยและเหยื่อ ประชาชื่น

## Collect Data From My Web API

ในส่วนของข้อมูลได้ทำการนำ URL ที่ให้ข้อมูลมา จาก <https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8> และใช้วิธีการ urllib request เพื่อดาวน์โหลดข้อมูลจาก HTTP URL

```
In [2]: url = "https://data.cityofnewyork.us/resource/833y-fsy8.json" # url variable เป็นตัวแปรที่เก็บข้อมูลของ url ที่ต้องการจะใช้ในการดึงข้อมูล
response = urllib.request.urlopen(url) # shooting_json variable เป็นตัวแปรที่เก็บค่าลิงก์ไฟล์ข้อมูลจาก response variable
shooting_json = response.read().decode("utf-8") # response variable เป็นตัวแปรที่เก็บค่าลิงก์ urllib request เพื่อใช้ดาวน์โหลดข้อมูลจาก HTTP URL
#print(shooting_json) #แสดงข้อมูลที่ดาวน์โหลดมาได้จาก shooting_json variable
```

หลังจากดึงข้อมูลจาก Web API เรียบร้อยแล้วทำการ save file เก็บไว้ใน folder ที่ชื่อว่า \data\ เพื่อทำการ Data Staging

```
In [3]: # เมื่อกำ救 file shooting_json ในใน folder data ตัวยาสั่ง open('data\shooting_json', 'w') เพื่อเขียนไฟล์ลงใน folder data
with open('data\shooting.json', 'w') as outfile:
    json.dump(shooting_json, outfile)
```

## Pre Processing

เมื่อทำการ load ข้อมูลมาใน folder data และทำการ load ไฟล์ข้อมูลเข้ามาใช้ในการวิเคราะห์ด้วยคำสั่งด้านล่างนี้

```
In [4]: shooting = json.loads(shooting_json) # คำสั่งใช้ในการ load ข้อมูลมาใช้จาก folder data
#shooting
```

หลังจากที่ทำการ load ข้อมูลมาใช้เรียบร้อยแล้วทำการ normalize ข้อมูลเพื่อให้อยู่ในรูปของ Dataframe เพื่อนำไปใช้ในการวิเคราะห์ได้ง่ายมากขึ้น

```
In [5]: df_shooting = json_normalize(shooting) #ใช้ฟังก์ชัน json_normalize จาก semi-structured JSON data ไปเป็น flat table หรือ dataframe
#เพื่อให้อ่านข้อมูลได้สะดวกมากขึ้น และเก็บ dataframe ไว้ในชื่อ df_shooting
df_shooting.head(10) #ใช้ฟังก์ชัน df_shooting.head(10) ในการแสดงข้อมูล 10 ตัวแรก
```

<ipython-input-5-a8715584698c>:1: FutureWarning: pandas.io.json.json\_normalize is deprecated, use pandas.json\_normalize instead  
df\_shooting = json\_normalize(shooting) #ใช้ฟังก์ชัน json\_normalize จาก semi-structured JSON data ไปเป็น flat table หรือ dataframe

Out[5]:

	incident_key	occur_date	occur_time	boro	precinct	jurisdiction_code	statistical_murder_flag	vic_age_group	vic_sex	vic_race	...	:@compute
0	74146165	2010-08-14T00:00:00.000	3:11:00	QUEENS	113	0	False	25-44	M	BLACK	...	
1	66928846	2009-10-17T00:00:00.000	18:03:00	BROOKLYN	67	0	True	45-64	M	BLACK	...	
2	29114164	2007-05-18T00:00:00.000	23:00:00	BROOKLYN	75	0	False	25-44	M	BLACK	...	
3	85180336	2012-06-09T00:00:00.000	17:15:00	BROOKLYN	81	0	False	25-44	M	BLACK	...	
4	73405770	2010-06-27T00:00:00.000	4:14:00	BRONX	47	0	False	25-44	M	BLACK	...	
5	33397043	2007-07-26T00:00:00.000	23:05:00	QUEENS	110	0	False	18-24	M	BLACK	...	
6	185483181	2018-07-21T00:00:00.000	4:15:00	QUEENS	114	0	False	25-44	M	BLACK	...	
7	137804320	2014-07-20T00:00:00.000	1:40:00	QUEENS	113	0	False	18-24	M	BLACK	...	
8	82323802	2011-12-25T00:00:00.000	3:50:00	QUEENS	113	0	False	18-24	M	BLACK	...	
9	74568517	2010-09-11T00:00:00.000	21:40:00	BRONX	43	0	False	18-24	M	BLACK	...	

10 rows × 25 columns

ผลลัพธ์จาก output ค่าอนุมัติแสดงให้เห็นว่าข้อมูลยังคงถูก clean ก่อนนำไปใช้วิเคราะห์ทั้งข้อมูลที่ต้องรีเซ็ต ค่า NAN และค่า value ในแต่ละคอลัมน์ที่ต้องรีเซ็ต

```
In [6]: df_shooting.info() #ศาสตร์ในการดูข้อมูลที่ฐานข้อมูล dataframe
#เพื่อดูรายการของข้อมูล จำนวนข้อมูลที่มีและ คอลัมน์ที่บันทึก ประเภทของข้อมูลที่บันทึก
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 25 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   incident_key    1000 non-null  object 
 1   occur_date      1000 non-null  object 
 2   occur_time      1000 non-null  object 
 3   boro            1000 non-null  object 
 4   precinct         1000 non-null  object 
 5   jurisdiction_code 1000 non-null  object 
 6   statistical_murder_flag 1000 non-null  bool  
 7   vic_age_group   1000 non-null  object 
 8   vic_sex          1000 non-null  object 
 9   vic_race          1000 non-null  object 
 10  x_coord_cd     1000 non-null  object 
 11  y_coord_cd     1000 non-null  object 
 12  latitude         1000 non-null  object 
 13  longitude        1000 non-null  object 
 14  :@computed_region_efsh_h5xi 999 non-null  object 
 15  :@computed_region_f5dn_yrer 1000 non-null  object 
 16  :@computed_region_yejl_bk3q 1000 non-null  object 
 17  :@computed_region_92fq_4b7q 1000 non-null  object 
 18  :@computed_region_sbqj_enih 1000 non-null  object 
 19  geocoded_column.type 1000 non-null  object 
 20  geocoded_column.coordinates 1000 non-null  object 
 21  perp_age_group   650 non-null   object 
 22  perp_sex          652 non-null   object 
 23  perp_race          652 non-null   object 
 24  location_desc    421 non-null   object 
dtypes: bool(1), object(24)
memory usage: 188.6+ KB
```

output จากค่าอนุมัติได้ว่ามีข้อมูลจำนวนทั้งหมด 1000 rows และมีคอลัมน์ทั้งหมด 25 columns ประเภทข้อมูลเป็น object เกือบทั้งหมด และมี bool อยู่อีก 1 คอลัมน์ เป็นการดูภาพรวมที่ไว้วิเคราะห์ในการเลือก clean data

## clean data

เริ่มท่าความสะอาดข้อมูลโดยรีเม็งจากเปลี่ยนชื่อคอลัมน์บางคอลัมน์ที่ไม่มีความหมาย และเข้าใจยาก ให้สามารถเข้าใจได้ง่ายมากขึ้น

```
In [7]: #ดำเนินการ rename columns ให้ชื่อของคอลัมน์ใน dataframe
# inplace = True เมื่อการพิมพ์ในเพื่อไม่ต้องป้อน again / nothing is returned
df_shooting.rename(columns = {'Old Column Name': 'New Column Name'})
```

```
In [8]: df_shooting.columns.values.tolist() # ให้เราดึงชื่อทุกชื่อใน list ของคอลัมน์ทั้งหมดที่ส่งจาก google spreadsheet ที่มีผลลัพธ์เรียบร้อย
```

```
Out[8]: ['incident_key',
 'occur_date',
 'occur_time',
 'borough',
 'police_precinct',
 'jurisdiction_code',
 'statistical_murder_flag',
 'vic_age_group',
 'vic_sex',
 'vic_race',
 'X_plane',
 'Y_plane',
 'latitude',
 'longitude',
 'zip_codes',
 'community_districts',
 'borough_boundaries',
 'city_council_districts',
 'precinct',
 'geocoded_type',
 'geocoded_coordinates',
 'perp_age_group',
 'perp_sex',
 'perp_race',
 'location_desc']
```

### ความหมายของแอร์โอลอจีสติกส์

incident key ลើក លេខ ID ឃញសំខាន់ដោយគារកិច្ចកម្មធមួយ incident key ដែលបានផ្តល់នូវកីឡា រួមទាំង

incident\_key អង្គភាព ID និងឈ្មោះឯកសារទាំងអស់  
occurred\_date តើដៃ វាយពីកិច្ចការរាយក្រារ

occur\_date ឬចុះពាណិជ្ជកម្មនៃការបង្កើត

**borough** ที่ เนื้อที่เกิดผลการ์ฟาร์กส์

police precinct គីឡូ ដែកលេខគម្រោង (បៀនីរសសងមុខពីកគម្រោងទាំងអស់) នៃ NYPD <https://www1.nyc.gov/site/nypd/>

**jurisdiction\_code** คือ เขตอำนาจศาลที่เกิดเหตุการณ์ยิง (รหัสเขตอำนาจศาล 0 = ตรรกะวน, 1 = การแข่งสั่ง, 2 = ที่อยู่อาศัย และ

statistical murder flag คือ เป็นการฆ

vic age group คือ ช่วงอายุ

vic\_sex คือ เพศของเหยื่อ

vic race คือ ผ่านพันธุ์ของเหยื่อ

`x_plane` คือ พิกัดของเซลล์คูปเพอร์ในแกน x

`y_plane` คือ พิกัดของเฉลี่กอปเพอร์ในแกน y

latitude គឺ ជីវិកតាល់តីរុទ្ធគង់ពីបុនភេកម៉ែនកៅដឹងទេ

longitude គឺ ជីកតាល់ខ្លួនទិន្នន័យបែងក្រោមផ្លូវការ

`zip_codes` คือ รหัสไปรษณีย์ของพื้นที่เกิดเหตุ

`community_districts` คือ รหัสของย่านชุมชนที่เกิดเหตุ

## borough\_boundaries គឺ ខុស

**city\_council\_districts** คือ เขตเทศบาลเมืองที่เกิดเหตุ

precinct គឺ នៅក្នុងរាជធានីភ្នំពេញ ដែលមានចារាប់ខ្លួន។

perp\_sex ตือ เพศของคนร้าย  
perp\_race ตือ เชื้อชาติของคนร้าย  
location\_desc ตือ สถานที่ที่เกิดเหตุ

In [9]: df\_shooting

Out[9]:

	incident_key	occur_date	occur_time	borough	police_precinct	jurisdiction_code	statistical_murder_flag	vic_age_group	vic_sex	vic_race	...	ci
0	74146165	2010-08-14T00:00:00.000	3:11:00	QUEENS	113	0	False	25-44	M	BLACK	...	...
1	66928846	2009-10-17T00:00:00.000	18:03:00	BROOKLYN	67	0	True	45-64	M	BLACK	...	...
2	29114164	2007-05-18T00:00:00.000	23:00:00	BROOKLYN	75	0	False	25-44	M	BLACK	...	...
3	85180336	2012-06-09T00:00:00.000	17:15:00	BROOKLYN	81	0	False	25-44	M	BLACK	...	...
4	73405770	2010-06-27T00:00:00.000	4:14:00	BRONX	47	0	False	25-44	M	BLACK	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...
995	82157674	2011-12-13T00:00:00.000	10:09:00	QUEENS	101	0	False	18-24	M	BLACK	...	...
996	85127160	2012-06-06T00:00:00.000	12:48:00	QUEENS	113	0	False	25-44	M	BLACK	...	...
997	85910307	2012-07-24T00:00:00.000	19:33:00	BROOKLYN	77	0	False	18-24	M	BLACK	...	...
998	91744329	2013-07-21T00:00:00.000	22:25:00	BROOKLYN	77	0	False	25-44	M	BLACK	...	...
999	183308090	2018-05-30T00:00:00.000	23:45:00	BROOKLYN	75	2	True	45-64	M	BLACK	...	...

1000 rows x 25 columns

จาก output จะเห็นว่า index ของ dataframe นี้ค่อนข้างที่จะไม่เป็นระเบียบเลยเลือกที่จะเปลี่ยนชื่อ index ดูสวยงามเข้าใจได้ง่าย

In [10]: index\_sort = [i for i in range(1000)]  
df\_shooting.index = index\_sort  
df\_shooting.index

Out[10]: Int64Index([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,  
...,  
990, 991, 992, 993, 994, 995, 996, 997, 998, 999],  
dtype='int64', length=1000)

ต่อไปจะทำการลอกคอลัมน์ precinct, borough\_boundaries, x\_plane, y\_plane, zip\_codes, geocoded\_type, jurisdiction\_code, community\_districts, city\_council\_districts, geocoded\_coordinates, police\_precinct เนื่องจากทั้งคอลัมน์ทั้งหมดนี้ไม่ใช่ในการวิเคราะห์ข้อมูลในภายหลัง

In [11]: # ทำการลอกคอลัมน์ไม่ได้ใช้คำสั่ง df\_shooting.drop(['Column Name'], axis=1)  
# โดยใช้ axis=1 นือจะลบต่อการลบ column ถ้าหากต้องการลบ row ทีนี้ใช้คำสั่ง axis=0  
df\_shooting = df\_shooting.drop(['precinct','borough\_boundaries','x\_plane','y\_plane','zip\_codes','geocoded\_type','jurisdiction\_code','community\_districts','city\_council\_districts','geocoded\_coordinates','police\_precinct'], axis=1)

หลังจากนั้นจะทำการ clean ข้อมูลที่เป็น Missing Data(Null Data)

In [12]: df\_shooting.isnull().sum() # ใช้คำสั่ง df\_shooting.isnull().sum() เพื่อหาจำนวนของค่า null ในแต่ละคอลัมน์ภายใน dataframe

Out[12]: incident\_key 0  
occur\_date 0  
occur\_time 0  
borough 0  
statistical\_murder\_flag 0  
vic\_age\_group 0  
vic\_sex 0  
vic\_race 0  
latitude 0  
longitude 0  
perp\_age\_group 350  
perp\_sex 348  
perp\_race 348  
location\_desc 579  
dtype: int64

จาก output ด้านบนจะพบว่าคอลัมน์ location\_desc, perp\_age\_group, perp\_sex และ perp\_race มีค่า null อุบัติจันทร์มาหาก โดยเฉพาะคอลัมน์ location\_desc เป็นจังหวะค่า null มากเกินไป และในช่วงเวลาเดียวกันนี้มีค่า null ในคอลัมน์ location\_desc โดยใช้คำสั่ง df\_shooting.drop(['Column Name'], axis=1)

```
In [13]: df_shooting = df_shooting.drop(['location_desc'], axis=1) #ทำการลอกคอลัมน์ location_desc โดยใช้คำสั่ง df_shooting.drop(['Column Name'], axis=1)
```

ผลลัพธ์ในคอลัมน์ perp\_age\_group, perp\_sex และ perp\_race ยังคงมีค่า null อุบัติจันทร์มาหาก แต่ลักษณะของ missing value เหล่านี้ลอกก่อนท่านไปเคราะห์

```
In [14]: #ทำการลบ missing value ที่บันทึกภายในคอลัมน์ perp_age_group, perp_sex และ perp_race ด้วยคำสั่ง df_shooting.dropna()
```

```
In [15]: df_shooting.isnull().sum() #เรียกใช้คำสั่งนี้เพื่อแสดงค่า null ภายในแต่ละคอลัมน์
```

```
Out[15]: incident_key      0  
occur_date      0  
occur_time      0  
borough        0  
statistical_murder_flag  0  
vic_age_group   0  
vic_sex         0  
vic_race        0  
latitude        0  
longitude       0  
perp_age_group  0  
perp_sex         0  
perp_race        0  
dtype: int64
```

จาก output ด้านบนจะไม่พบค่า null ภายในคอลัมน์ perp\_age\_group, perp\_sex และ perp\_race แล้ว

```
In [16]: df_shooting['occur_date'].head(100) #ใช้คำสั่ง df_shooting['ColName'].head(num) เพื่อเรียกดูข้อมูลภายใน dataframe 100 ตัวแรก
```

```
Out[16]: 6    2018-07-21T00:00:00.000  
14   2010-10-03T00:00:00.000  
21   2010-07-02T00:00:00.000  
22   2014-03-08T00:00:00.000  
24   2010-08-09T00:00:00.000  
...  
163  2009-07-04T00:00:00.000  
164  2009-08-02T00:00:00.000  
165  2013-12-10T00:00:00.000  
166  2013-04-16T00:00:00.000  
168  2006-03-18T00:00:00.000  
Name: occur_date, Length: 100, dtype: object
```

จาก output ด้านบนจะพบว่าข้อมูลในคอลัมน์ occur\_date จะเป็นมีอุบัติจันทร์ที่มีเวลา timestamp เพิ่มมากขึ้นโดยจะแบ่งเป็นตัวที่ 0-9 ตัวแรกเป็นข้อมูลที่เราต้องการคือ วัน เดือนปีที่เกิดเหตุ และตัวที่ 10 เป็นต้นไปคือ timestamp ที่จะลงทำการตัดที่ตั้งนี้ออกจากคอลัมน์ occur\_time อุบัติจันทร์ และ timestamp นั้นมีค่าเป็น 0 ทั้งหมด

```
In [17]: occur_date2 = [] #สร้าง list ใหม่ที่ชื่อว่า occur_date2 เพื่อต้องการเก็บข้อมูลเดือนปีที่เหลือจาก timestamp ที่ง่ายกว่า  
for i in df_shooting['occur_date']: #ทำการวนลูป สำหรับที่ i เป็นข้อมูลแต่ละตัวที่อยู่ในคอลัมน์ occur_date  
    date = i[0:10] #นำตัวแรกของ date ออก คือ ข้อมูลเดือนปีที่อยู่ในคอลัมน์ occur_date ที่มีตัวที่ 0-9  
    occur_date2.append(date) #นำ date เข้าไปเก็บไว้ใน list ที่ชื่อว่า occur_date2 ด้วยคำสั่ง listname.append()  
df_shooting['occur_date2'] = occur_date2 #ทำการสร้างคอลัมน์ใหม่ใน dataframe ที่ชื่อว่า occur_date2 ด้วยข้อมูลจาก occur_date2 เมื่อ list ที่เราทำการเก็บข้อมูลรันแล้วมีไว้
```

หลังจากทำการตัด timestamp ที่ง่ายกว่าข้อมูลเดือนปีไว้ในคอลัมน์ occur\_date2 เรายังคงข้อมูลภายในคอลัมน์ occur\_date

```
In [18]: #เมื่อจากเราได้ออกการใช้ชื่อข้อมูลภายในคอลัมน์ occur_date แล้วเริ่งทำการลบคอลัมน์นี้ทิ้ง  
df_shooting = df_shooting.drop(['occur_date'], axis=1) #โดยใช้คำสั่ง df_shooting.drop(['Column Name'], axis=1)
```

```
In [19]: df_shooting['occur_date2'].head(5) #เช็คกุญแจ 5 ตัวแรกในคอลัมน์ occur_date2
```

```
Out[19]: 6    2018-07-21  
14   2010-10-03  
21   2010-07-02  
22   2014-03-08  
24   2010-08-09  
Name: occur_date2, dtype: object
```

จาก output ด้านบนจะเห็นได้ว่าข้อมูลภายในคอลัมน์ occur\_date2 ไม่มีเวลา timestamp แล้วแต่ยังต้องการแบ่งข้อมูลออกเป็น 3 คอลัมน์เพื่อให้ง่ายต่อการวิเคราะห์มากขึ้นโดยแบ่งคอลัมน์เด่นๆ Year, Month, Day

```
In [20]: #ทำการแบ่งคอลัมน์ occur_date2 ออกเป็น 3 คอลัมน์คือ Year, Month, Day
#เนื่องจากข้อมูลมีภาษาไทยคอลัมน์ occur_date2 มีค่าเป็น YYYY-MM-DD จึงทำการตัดชื่อมูลออกเป็น 3 คอลัมน์ด้วย '-'
df_shooting[['Year','Month','Day']] = df_shooting['occur_date2'].str.split("-",expand=True)

#สร้างคอลัมน์ใหม่ที่มีชื่อ Year, Month, Day โดยให้มีค่าเท่ากับคอลัมน์ occur_date2 ที่ทำการตัดชื่อมาแล้ว ด้วยคำสั่ง df_shooting['ColName'].str.split("-",expand=True)
# expand=True คือ เมื่อเราทำการ split string และเป็นผลลัพธ์คอลัมน์แล้วต้องการให้ return ผลลัพธ์เป็น dataframe
# expand=False คือ เมื่อเราทำการ split string และเป็นผลลัพธ์คอลัมน์แล้วต้องการให้ return ผลลัพธ์เป็น series
```

```
In [21]: df_shooting[['Year','Month','Day']] #แสดงข้อมูลในคอลัมน์ Year, Month, Day
```

Out[21]:

	Year	Month	Day
6	2018	07	21
14	2010	10	03
21	2010	07	02
22	2014	03	08
24	2010	08	09
...	...	...	...
993	2011	07	18
994	2017	07	11
995	2011	12	13
996	2012	06	06
999	2018	05	30

650 rows × 3 columns

จาก output จะพบว่าข้อมูล รันเดือนปีถูกแบ่งออกในแต่ละคอลัมน์เรียบร้อยแล้ว

ต่อไปจะทำการรีเคราะห์ value ในแต่ละคอลัมน์มาว่ามีค่า null หรือ ค่าที่คิดปกติหรือไม่เพื่อจากข้อมูลที่เราต้องการใช้เป็น categorical

เริ่มจากการรีเคราะห์ value ภายในคอลัมน์ perp\_age\_group

```
In [22]: df_shooting['perp_age_group'].value_counts() #ใช้คำสั่ง df_shooting['ColName'].value_counts() เพื่อเรียกดู value ของคอลัมน์ perp_age_group
```

Out[22]:

18-24	243
25-44	176
UNKNOWN	147
<18	64
45-64	17
65+	2
1020	1

Name: perp\_age\_group, dtype: int64

จากการแสดง output ด้านบนทำให้เห็นว่าคอลัมน์ perp\_age\_group มีค่า value ที่ด้องลงทึ้ง คือ 'UNKNOWN' และ '1020' ที่มีค่า null และมีค่าซ่างจากที่คิดปกติ ไม่เป็นความจริง(outlier) ด้วยคำสั่งด้านล่างนี้

```
In [23]: perp_age = df_shooting[df_shooting['perp_age_group'] == '1020'].index
#ให้ตั้งเปร parp_age นี่คือท่านที่ index ที่ value ภาษาไทยคอลัมน์ perp_age_group = '1020' ด้วยคำสั่ง df[df['ColName']== Value].index
df_shooting.drop(perp_age, inplace = True) #ทำการลบ index ที่มี value ในคอลัมน์ perp_age_group = '1020'
perp_age2 = df_shooting[df_shooting['perp_age_group'] == 'UNKNOWN'].index
#ให้ตั้งเปร perp_age2 นี่คือท่านที่ index ที่ value ภาษาไทยคอลัมน์ perp_age_group = 'UNKNOWN' ด้วยคำสั่ง df[df['ColName']== Value].index
df_shooting.drop(perp_age2, inplace = True)
#ทำการลบ index ที่มี value ในคอลัมน์ perp_age_group = 'UNKNOWN'
```

```
In [24]: df_shooting['perp_age_group'].value_counts() #เรียกดู value ภายในคอลัมน์ perp_age_group
```

Out[24]:

18-24	243
25-44	176
<18	64
45-64	17
65+	2

Name: perp\_age\_group, dtype: int64

จะเห็นได้ว่า output เมื่อทำการเรียกดู value ของคอลัมน์ perp\_age\_group ไม่มี value 'UNKNOWN' และ '1020' แล้ว

ต่อไปจะทำการรีเคราะห์ value ภายในคอลัมน์ perp\_sex

```
In [25]: df_shooting['perp_sex'].value_counts() #ใช้คำสั่ง df_shooting['Column Name'].value_counts() เพื่อทำการเรียกดู value ของคอลัมน์ perp_sex
```

```
Out[25]: M    491  
F     10  
U     1  
Name: perp_sex, dtype: int64
```

จะเห็นได้ว่าจาก output เมื่อทำการเรียกดู value ของคอลัมน์ perp\_sex จะพบ value 'U' ที่ไม่ใช่ค่าที่แสดงเพศชาย หรือ หญิงซึ่งต้องทำการลบทิ้ง

```
In [26]: perp_sex = df_shooting[df_shooting['perp_sex'] == 'U'].index  
#ให้รวม perp_sex มีค่าที่เป็น index ตัวที่ value ภายในคอลัมน์ perp_sex = 'U' ด้วยคำสั่ง df[df['ColumnName'] == Value].index  
df_shooting.drop(perp_sex, inplace = True)  
#ทำการลบ index ที่มี value ในคอลัมน์ perp_sex = 'U'
```

```
In [27]: df_shooting['perp_sex'].value_counts() #เรียกดู value ภายในคอลัมน์ perp_sex
```

```
Out[27]: M    491  
F     10  
Name: perp_sex, dtype: int64
```

จะเห็นได้ว่า output เมื่อทำการเรียกดู value ของคอลัมน์ perp\_sex 'ไม่มี value 'U' แล้ว

ต่อไปจะทำการรีเคราะห์ value ภายในคอลัมน์ perp\_race

```
In [28]: df_shooting['perp_race'].value_counts() #ใช้คำสั่ง df_shooting['Column Name'].value_counts() เพื่อทำการเรียกดู value ของคอลัมน์ perp_race
```

```
Out[28]: BLACK      367  
WHITE HISPANIC    72  
BLACK HISPANIC    39  
WHITE             12  
UNKNOWN           10  
ASIAN / PACIFIC ISLANDER 1  
Name: perp_race, dtype: int64
```

จาก output เมื่อทำการเรียกดู value ของคอลัมน์ perp\_race ทำให้พบ value ที่มีค่า 'UNKNOWN' อยู่ซึ่งจะต้องทำการลบทิ้งเนื่องจากเป็นค่า null

```
In [29]: perp_race = df_shooting[df_shooting['perp_race'] == 'UNKNOWN'].index  
#ให้รวม perp_race มีค่าที่เป็น index ตัวที่ value ภายในคอลัมน์ perp_race = 'UNKNOWN' ด้วยคำสั่ง df[df['ColumnName'] == Value].index  
df_shooting.drop(perp_race, inplace = True)  
#ทำการลบ index ที่มี value ในคอลัมน์ perp_race = 'UNKNOWN'
```

```
In [30]: df_shooting['perp_race'].value_counts() #เรียกดู value ภายในคอลัมน์ perp_race
```

```
Out[30]: BLACK      367  
WHITE HISPANIC    72  
BLACK HISPANIC    39  
WHITE             12  
ASIAN / PACIFIC ISLANDER 1  
Name: perp_race, dtype: int64
```

จะเห็นได้ว่า output เมื่อทำการเรียกดู value ของคอลัมน์ perp\_race 'ไม่มี value 'UNKNOWN' แล้ว

ต่อไปจะทำการรีเคราะห์ value ภายในคอลัมน์ vic\_age\_group

```
In [31]: df_shooting['vic_age_group'].value_counts() #ใช้คำสั่ง df_shooting['Column Name'].value_counts() เพื่อทำการเรียกดู value ของคอลัมน์ vic_age_group
```

```
Out[31]: 18-24    202  
25-44    192  
<18      53  
45-64    38  
65+      4  
UNKNOWN    2  
Name: vic_age_group, dtype: int64
```

จาก output จะเห็นว่าคอลัมน์ vic\_age\_group มีค่า value 'UNKNOWN' ที่ต้องลบทิ้งเนื่องจากเป็นค่า null

```
In [32]: vic_age = df_shooting[df_shooting['vic_age_group'] == 'UNKNOWN'].index  
#ให้รวม vic_age มีค่าที่เป็น index ตัวที่ value ภายในคอลัมน์ vic_age_group = 'UNKNOWN' ด้วยคำสั่ง df[df['ColumnName'] == Value].index  
df_shooting.drop(vic_age, inplace = True)  
#ทำการลบ index ที่มี value ในคอลัมน์ vic_age_group = 'UNKNOWN'
```

```
In [33]: df_shooting['vic_age_group'].value_counts() #เรียกดู value ภายในคอลัมน์ vic_age_group
```

```
Out[33]: 18-24    202  
25-44    192  
<18      53  
45-64    38  
65+      4  
Name: vic_age_group, dtype: int64
```

จะเห็นได้ว่า output เมื่อทำการเรียกดู value ของคอลัมน์ vic\_age\_group ไม่มี value 'UNKNOWN' แล้ว

ต่อไปจะทำการเรียกดู value ภายในคอลัมน์ vic\_race

```
In [34]: df_shooting['vic_race'].value_counts() #ใช้คำสั่ง df_shooting['Column Name'].value_counts() เพื่อทำการเรียกดู value ของคอลัมน์ vic_race
```

```
Out[34]: BLACK           320  
WHITE HISPANIC        89  
BLACK HISPANIC        52  
WHITE                 20  
ASIAN / PACIFIC ISLANDER   8  
Name: vic_race, dtype: int64
```

จาก output ด้านบนเมื่อทำการเรียกดู value ของคอลัมน์ vic\_race จะพบว่าค่า value 'UNKNOWN' ต้องทำการลบซึ่งเนื่องจากเป็นค่า null

```
In [35]: vic_race = df_shooting[df_shooting['vic_race'] == 'UNKNOWN'].index  
#ให้ดูว่ามี vic_race ที่ค่าเท่ากับ index ตัวใด value ภายในคอลัมน์ vic_race = 'UNKNOWN' ด้วยคำสั่ง df[df['ColName'] == Value].index  
df_shooting.drop(vic_race, inplace = True)  
#ทำการลบ index ที่มี value ในคอลัมน์ vic_race = 'UNKNOWN'
```

```
In [36]: df_shooting['vic_race'].value_counts() #เรียกดู value ภายในคอลัมน์ vic_race
```

```
Out[36]: BLACK           320  
WHITE HISPANIC        89  
BLACK HISPANIC        52  
WHITE                 20  
ASIAN / PACIFIC ISLANDER   8  
Name: vic_race, dtype: int64
```

จะเห็นได้ว่า output เมื่อทำการเรียกดู value ของคอลัมน์ vic\_race ไม่มี value 'UNKNOWN' แล้ว

หลังจาก วิเคราะห์ value แต่ละคอลัมน์เรียบร้อยแล้วจึงใช้คำสั่ง df.info() เพื่อเรียกดูข้อมูลที่น่าสนใจของ dataframe สำหรับเพื่อ check ว่าข้อมูลนั้นถูก clean พร้อมใช้งานหรือไม่แล้ว

```
In [37]: df_shooting.info() #เรียกดูข้อมูลที่น่าสนใจของ df_shooting
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 489 entries, 6 to 999  
Data columns (total 16 columns):  
 #   Column          Non-Null Count  Dtype     
---    
 0   incident_key    489 non-null   object    
 1   occur_time     489 non-null   object    
 2   borough         489 non-null   object    
 3   statistical_murder_flag 489 non-null   bool     
 4   vic_age_group  489 non-null   object    
 5   vic_sex         489 non-null   object    
 6   vic_race        489 non-null   object    
 7   latitude        489 non-null   object    
 8   longitude       489 non-null   object    
 9   perp_age_group 489 non-null   object    
 10  perp_sex        489 non-null   object    
 11  perp_race       489 non-null   object    
 12  occur_date2    489 non-null   object    
 13  Year            489 non-null   object    
 14  Month           489 non-null   object    
 15  Day             489 non-null   object    
dtypes: bool(1), object(15)  
memory usage: 61.6+ KB
```

## 5. Visualization table+graph

1. วิเคราะห์เห็นแต่ละเพศเมียช่วงอายุในช่วงไหน

2. วิเคราะห์เห็นแต่ละเพศเมียชื่อชาติอะไร

- 3.วิเคราะห์คนร้ายแต่ละเพศเมื่อช่วงอายุอยู่ในช่วงไหน
- 4.วิเคราะห์คนร้ายแต่ละเพศเมื่อเชื้อชาติอะไร
- 5.เพศของเหยื่อที่เสียชีวิตมากที่สุด
- 6.ช่วงอายุของเหยื่อที่เสียชีวิตมากที่สุด
- 7.เชื้อชาติของเหยื่อที่เสียชีวิตมากที่สุด
- 8.เขตใน เมือง NYC ที่มีจำนวนการเสียชีวิตมากที่สุด
- 9.ที่ก่อของผู้รอดชีวิตในแต่ละเขตของเมือง NYC
- 10.พื้นที่ก่อของผู้เสียชีวิตในแต่ละเขตของเมือง NYC
- 11.กราฟจำนวนการเกิดเหตุการณ์ใน NYC ของแต่ละปี
- 12.วิเคราะห์แต่ละปีเมื่อจำนวนการเสียชีวิตเป็นอย่างไร

## 1.วิเคราะห์เหยื่อแต่ละเพศเมื่อช่วงอายุในช่วงไหน

ต้องการวิเคราะห์ว่าเหยื่อที่ถูกยิงแต่ละเพศเมื่อช่วงอายุไหนที่ถูกยิงมากที่สุด และน้อยที่สุดด้วย heat map

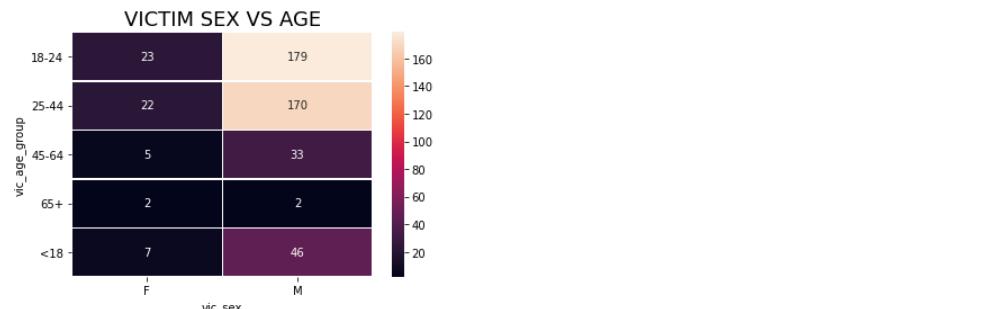
```
In [38]: # ใน vic_sex_age เป็นตัวมายกที่นับ crossstab ระหว่างคอลัมน์ vic_age_group และ vic_sex
vic_sex_age = pd.crosstab(df_shooting['vic_age_group'], df_shooting['vic_sex']) #ใช้คำสั่ง pd.crosstab(df[col1],df[col2])

#สร้าง heatmap จาก crossstab ระหว่างคอลัมน์ vic_age_group และ vic_sex
sns.heatmap(pd.crosstab(df_shooting['vic_age_group'], df_shooting['vic_sex']), annot=True, cbar=True, fmt="d", linewidth=0.3)
#สร้าง heatmap โดยใช้ seaborn 作為 sns คำสั่ง sns.heatmap(pd.crosstab(df[col1],df[col2]))
# annot = True คือ ให้แสดงค่า data value ที่ count ได้
#fmt="d" คือ ให้ data value ที่มีพื้นค่าเป็น int
#cbar=True คือ ต้องการให้มีส่วนแยกตัวข้างขวาด้วย

# ticks
yticks = [i.upper() for i in vic_sex_age.index] #ใน yticks เก็บชื่อชุด list of i ตัวในชุดที่หันหน้าออกที่ i เป็น index ของชื่อชุดภาษาไทยใน vic_sex_age
xticks = [i.upper() for i in vic_sex_age.columns] #ใน xticks เก็บชื่อชุด list of i ตัวในชุดที่หันหน้าออกที่ i เป็น columns ของชื่อชุดภาษาไทยใน vic_sex_age
#ใช้คำสั่ง plt.yticks สำหรับการตั้งค่า position ticks
#labels=yticks คือ ให้ label เป็น list ของชื่อชุดที่เราต้องการแสดงคือ yticks
#rotation=0 คือ กำหนดให้ label ที่มีมุม = 0 คือไม่ต้องเอียง
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)

# title
title = 'VICTIM SEX VS AGE' #ให้ตั้งเป็น title ก็จะขึ้น title ของ graph นี้
plt.title(title, loc='center', fontsize=18) #ใช้คำสั่ง plt.title() และชื่อ title โดยมี location ที่ center ขนาดฟอนต์ = 18
```

Out[38]: Text(0.5, 1.0, 'VICTIM SEX VS AGE')



จากผลลัพธ์ของกราฟ heatmap และลงในเห็นว่า

- เหยื่อที่ถูกยิงมากเป็นเพศชายมากกว่าเพศหญิงโดยๆ ได้จากกลุ่ม F คือ มีจำนวนสูงมากกว่าสิ่งที่
- ในช่วงของเพศชาย มีจำนวนการถูกยิงอยู่ในช่วงอายุ 18-24 ปีมากที่สุด คือ 179 cases และช่วงอายุที่สำคัญ คือ 65 ปีขึ้นไปมีเพียง 2 cases
- ในส่วนของเพศหญิง มีจำนวนการถูกยิงอยู่ในช่วงอายุ 18-24 ปีมากที่สุด คือ 22 cases และช่วงอายุที่สำคัญ คือ 65 ปีขึ้นไปมีเพียง 2 cases
- สังเกตเห็นได้ว่าช่วงอายุของเหยื่อทั้งเพศชาย และหญิงที่ถูกยิงมากที่สุด คือ 18-24 ปี หรือ ช่วงวัยรุน และอยู่ที่สุด คือ 65 ปีขึ้นไป หรือ คนชรา อายุในช่วงอายุเหล่านี้รวมกันมากที่สุดในจำนวนการเสียชีวิตที่มาจากการถูกยิงมากกว่าเพศชาย และหญิงจากมาในปีน้อยลงที่ 18-24 > 25-44 > 45-64 > 65 ปีขึ้นไป แต่อย่างไร ก็ตามเพศชายมีจำนวนการถูกยิงที่มากกว่าเพศหญิงอยู่มาก

## 2.วิเคราะห์เหยื่อแต่ละเพศมีเชื้อชาติอะไร

ต้องการรู้ว่าคนที่ถูกฆ่านี้ที่ถูกฆ่านี้แต่ละเพศมีเชื้อชาติไหนที่ถูกฆามากที่สุด และน้อยที่สุดด้วย heat map

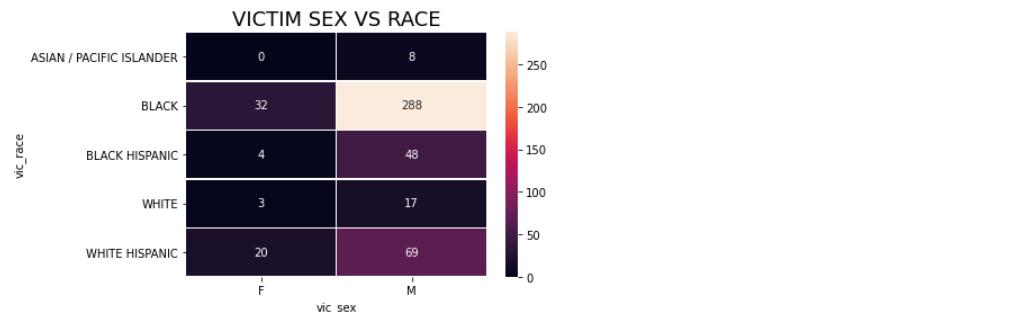
```
In [39]: # ให้ vic_sex_race เป็นตัวแปรกับ crosstab ระหว่างคอลัมน์ vic_race และ vic_sex
vic_sex_race = pd.crosstab(df_shooting['vic_race'], df_shooting['vic_sex'])

#สร้าง heatmap จาก crosstab ระหว่างคอลัมน์ vic_race และ vic_sex
sns.heatmap(pd.crosstab(df_shooting['vic_race'], df_shooting['vic_sex']), annot=True, cbar=True, fmt="d", linewidth=0.3)

# ticks
yticks = [i.upper() for i in vic_sex_race.index] #ให้ yticks เก็บข้อมูล list ของ index ทั้งหมดโดยที่ i เป็น index ของข้อมูลภายใน vic_sex_race
xticks = [i.upper() for i in vic_sex_race.columns] #ให้ xticks เก็บข้อมูล list ของ columns ของข้อมูลภายใน vic_sex_race
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)

# title
title = 'VICTIM SEX VS RACE'
plt.title(title, loc='center', fontsize=18)
```

Out[39]: Text(0.5, 1.0, 'VICTIM SEX VS RACE')



จากผลลัพธ์ของกราฟ heatmap แสดงให้เห็นว่า

- ในส่วนของเพศชาย มีจำนวนการถูกฆ่าที่เชื้อชาติ BLACK มากที่สุด คือ 288 cases และเชื้อชาติ ASIAN/PACIFIC ISLANDER น้อยที่สุดโดยมีจำนวน 8 cases
- ในส่วนของเพศหญิง มีจำนวนการถูกฆ่าที่เชื้อชาติ BLACK น้อยที่สุด คือ 32 cases และเชื้อชาติ ASIAN/PACIFIC ISLANDER น้อยที่สุดโดยมีจำนวน 0 cases
- สังเกตเห็นได้ว่าเชื้อชาติของเหยื่อทั้งเพศชาย และหญิงที่ถูกฆามากที่สุด คือ BLACK และอยู่ที่สุด คือ ASIAN/PACIFIC ISLANDER เพียงเท่านั้น
- จำนวนเหยื่อที่ถูกฆ่าทั้งเพศชาย และหญิงมีลักษณะจากภาพที่สูงไปโลดโผนกันส์ คือ BLACK -> WHITE HISPANIC -> BLACK HISPANIC -> WHITE -> ASIAN/PACIFIC ISLANDER ซึ่งจะเห็นได้ว่าเชื้อชาติ BLACK มีจำนวนการถูกฆ่าที่สูง

## 3.วิเคราะห์คนร้ายแต่ละเพศมีช่วงอายุอยู่ในช่วงไหน

ต้องการรู้ว่าคนร้ายแต่ละเพศมีช่วงอายุใหมมากที่สุด และน้อยที่สุดด้วย heat map

```
In [40]: # ให้ perp_sex_age เป็นตัวแปรกับ crosstab ระหว่างคอลัมน์ perp_age_group และ perp_sex
perp_sex_age = pd.crosstab(df_shooting['perp_age_group'], df_shooting['perp_sex'])

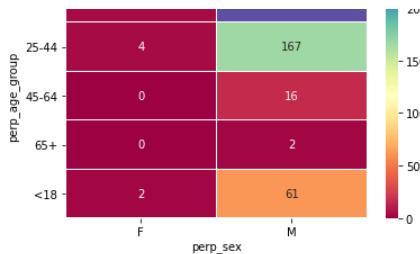
#สร้าง heatmap จาก crosstab ระหว่างคอลัมน์ vic_race และ vic_sex
sns.heatmap(pd.crosstab(df_shooting['perp_age_group'], df_shooting['perp_sex']), annot=True, cbar=True, fmt="d", linewidth=0.3, cmap = "Spectral")

# ticks
yticks = [i.upper() for i in perp_sex_age.index] #ให้ yticks เก็บข้อมูล list ของ index ทั้งหมดโดยที่ i เป็น index ของข้อมูลภายใน perp_sex_age
xticks = [i.upper() for i in perp_sex_age.columns] #ให้ xticks เก็บข้อมูล list ของ columns ของข้อมูลภายใน perp_sex_age
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)

# title
title = 'PERPETRATOR SEX VS AGE'
plt.title(title, loc='center', fontsize=18)
```

Out[40]: Text(0.5, 1.0, 'PERPETRATOR SEX VS AGE')





จากผลลัพธ์ของกราฟ heatmap แสดงให้เห็นว่า

- ค่าร้ายเป็นเพศชายมากกว่าเพศหญิงโดยได้จากการสื้าเรียนรวมมั่ง และ value มีจำนวนสูงกว่าเพศหญิง
- ในสังขของเพศชาย มีจำนวนคนที่ร้ายในช่วงอายุ 18-24 ปีมากที่สุด ต่อ 233 cases และ 65 ปีขึ้นไปอยู่ที่สุดโดยมีจำนวน 2 cases
- ในสังขของเพศหญิง มีจำนวนคนที่ร้ายในช่วงอายุ 18-24 และ 25-44 ปีมากที่สุด ต่อ 4 cases ช่วงอายุ 45-64 และ 65 ปีขึ้นไปอยู่ที่สุดโดยมีจำนวน 0 cases
- สังเกตเห็นได้ว่าช่วงอายุของคนที่ร้ายที่ 18-24 ปีมีจำนวนมากที่สุดที่เพศชาย และหญิง หรือ คนร้ายยังอยู่ในช่วงรุ่น ยังเด็กในระดับมหาวิทยาลัยอยู่ รองลงมาจะเป็น ช่วงอายุ 25-44 ปี หรือ วัยทำงาน

#### 4. วิเคราะห์คนร้ายแต่ละเพศเมื่อเข้าชาติอะไร

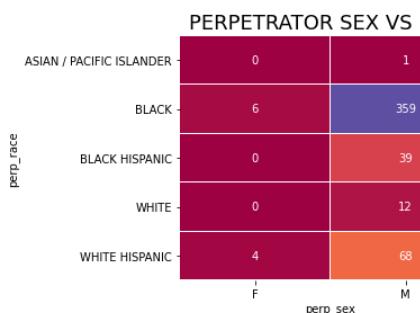
ต้องการวิเคราะห์จำนวนร้ายแต่ละเพศเมื่อเข้าชาติอะไรมากที่สุด และว้อยที่สุดด้วย heatmap

```
# ใน perp_race เป็นตัวแปรที่มี crossstab ระหว่าง colums ที่ perp_race และ perp_sex
perp_sex_race = pd.crosstab(df_shooting['perp_race'], df_shooting['perp_sex'])

sns.heatmap(pd.crosstab(df_shooting['perp_race'], df_shooting['perp_sex']), annot=True, cbar=True, fmt="d", linewidth=0.3, cmap = "Spectral")
# ticks
yticks = [i.upper() for i in perp_sex_race.index] # ให้ yticks เท่ากับมูล list of i ตัวในทุกห้องแม่โดยที่ i เป็น index ของชื่อคลาสใน perp_sex_race
xticks = [i.upper() for i in perp_sex_race.columns] # ให้ xticks เท่ากับมูล list of i ตัวในทุกห้องแม่โดยที่ i เป็น columns ของชื่อคลาสใน perp_sex_race
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)

# title
title = 'PERPETRATOR SEX VS RACE'
plt.title(title, loc='center', fontsize=18)
```

Out[41]: Text(0.5, 1.0, 'PERPETRATOR SEX VS RACE')



จากผลลัพธ์ของกราฟ heatmap แสดงให้เห็นว่า

- ในสังขของเพศชาย มีจำนวนการยิงของคนร้ายที่มีเข้าชาติ BLACK มากที่สุด ต่อ 359 cases และเข้าชาติ ASIAN/PACIFIC ISLANDER น้อยที่สุดโดยมีจำนวน 1 cases และมีสำคัญเข้าชาติของคนร้ายจากมากที่สุดไปน้อยที่สุดคือ BLACK > WHITE HISPANIC > BLACK HISPANIC > WHITE > ASIAN/PACIFIC ISLANDER
- ในสังขของเพศหญิง มีจำนวนการยิงของคนร้ายที่มีเข้าชาติ BLACK มากที่สุด ต่อ 6 cases และเข้าชาติ ASIAN/PACIFIC ISLANDER , BLACK HISPANIC และ WHITE น้อยที่สุดโดยมีจำนวน 0 cases และมีสำคัญเข้าชาติของคนร้ายจากมากที่สุดไปน้อยที่สุดคือ BLACK > WHITE HISPANIC > BLACK HISPANIC = WHITE = ASIAN/PACIFIC ISLANDER
- สังเกตเห็นได้ว่าเข้าชาติของคนร้ายที่เพศชาย และหญิงมากที่สุด ต่อ BLACK

สรุปผลจากการวิเคราะห์ 4 กราฟที่ผ่านมา

เห็นว่า

- จำนวนคนที่ถูกฆ่าที่สำคัญที่สุดอยู่ในช่วง 18-24 ปี หรือ ในช่วงรุ่นเรียนมหาวิทยาลัย มากที่สุดจะเป็นเพศชาย และเพศหญิง

- เข็มขัดเป็น black หรือ คนเก้าีสี มีจำนวนมากที่สุดของทั้งเพศชาย และเพศหญิง
- จำนวนเหยื่อเพศชาย มากกว่า เพศหญิง

#### คนร้าย

- คนร้ายมีอายุในช่วง 18-24 ปี หรือ ในช่วงวัยนรน(เรียนมหาวิทยาลัย) และรองลงมาจะเป็นช่วงอายุ 25-44 ปี หรือ วัยทำงานของทั้งเพศชาย และเพศหญิง
- เข็มขัดของคนร้ายมากที่สุด คือ black หรือ คนผิวสี และ WHITE HISPANIC เป็นเข็มขัดที่มีจำนวนมากของทั้งเพศชาย และหญิง
- จำนวนคนร้ายเพศชาย มากกว่า เพศหญิง

จะสังเกตเห็นได้ว่า ทั้งหมดและคนร้ายมีจำนวนเพศชายที่มากกว่าเพศหญิงอยู่ในช่วงอายุ 18-24 ปี และนี่เป็นข้อเดียวกันที่สุด แต่รองลงมาคือเข็มขัด WHITE HISPANIC ทำให้คาดการณ์ได้ว่าคนที่ดีความชัดแย้งระหว่างกันภายในเชื้อชาติ หรือ อาจเป็นความชัดแย้งกันระหว่างคนผิวสีกับคนขาว เนื่องจากมีกฎหมายชัดเจนเรื่องการเมืองเช่นเดียวกันในช่วงวัยรุ่น และเพศชาย ทำให้การชัดแย้งจนแรงถึงขั้นถึงกันได้ในหมู่เยาวชนชายมากกว่าในช่วงอายุเชื้อชาติอื่น

## 5. เพศของเหยื่อที่เสียชีวิตมากที่สุด

ต้องการวิเคราะห์ว่าเพศของเหยื่อที่เสียชีวิตมากที่สุด ด้วย heat map

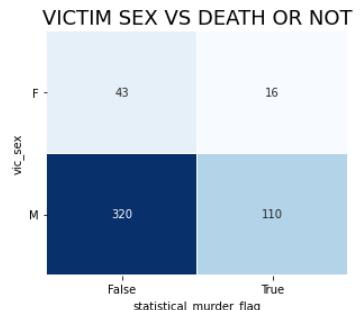
```
In [42]: # ใช้ death_sex เป็นตัวแปรที่ df Crosstab ระหว่าง columm vic_sex และ statistical_murder_flag
death_sex = pd.crosstab(df_shooting['vic_sex'],df_shooting['statistical_murder_flag'])

sns.heatmap(pd.crosstab(df_shooting['vic_sex'], df_shooting['statistical_murder_flag']), annot=True, cbar=True, fmt="d", linewidth=0.3, cmap = "Blues")

# ticks
yticks = [i.upper() for i in death_sex.index] # ให้ yticks เป็นชื่อชุด list of i ที่ใหญ่ที่สุดโดยที่ i เป็น index ของชื่อชุดภายใน perp_sex_race
xticks = [i for i in death_sex.columns] # ให้ xticks เป็นชื่อชุด list of i โดยที่ i เป็น columns ของชื่อชุดภายใน perp_sex_race
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)

# title
title = 'VICTIM SEX VS DEATH OR NOT'
plt.title(title, loc='center', fontsize=18)
```

Out[42]: Text(0.5, 1.0, 'VICTIM SEX VS DEATH OR NOT')



จากผลลัพธ์ของกราฟ heatmap และแสดงให้เห็นว่า

- เหยื่อเพศชาย มีจำนวนการเสียชีวิตมากกว่าเพศหญิง คือ จำนวนการเสียชีวิตชาย = 110 และฝ่ายหญิง = 16
- เหยื่อเพศชาย มีจำนวนการระดับชีวิตมากกว่าเพศหญิง คือ จำนวนการเสียชีวิตชาย = 320 และฝ่ายหญิง = 43
- เหยื่อทั้งเพศชาย และเพศหญิงมีจำนวนการระดับชีวิตมากกว่าเสียชีวิต

## 6. ช่วงอายุของเหยื่อที่เสียชีวิตมากที่สุด

ต้องการวิเคราะห์ว่าช่วงอายุของเหยื่อที่เสียชีวิตมากที่สุด ด้วย heat map

```
In [43]: # ใช้ death_age เป็นตัวแปรที่ df Crosstab ระหว่าง columm vic_age_group และ statistical_murder_flag
death_age = pd.crosstab(df_shooting['vic_age_group'],df_shooting['statistical_murder_flag'])

sns.heatmap(pd.crosstab(df_shooting['vic_age_group'], df_shooting['statistical_murder_flag']), annot=True, cbar=True, fmt="d", linewidth=0.3, cmap = "Blues")

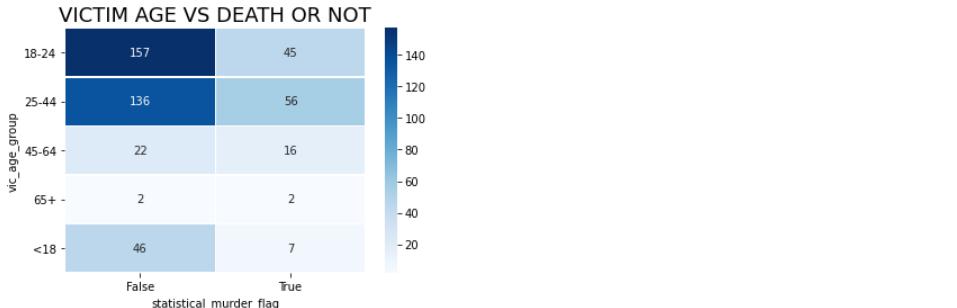
# ticks
yticks = [i.upper() for i in death_age.index] # ให้ yticks เป็นชื่อชุด list of i ที่ใหญ่ที่สุดโดยที่ i เป็น index ของชื่อชุดภายใน death_age
xticks = [i for i in death_age.columns] # ให้ xticks เป็นชื่อชุด list of i โดยที่ i เป็น columns ของชื่อชุดภายใน death_age
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
```

```

plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)
# title
title = 'VICTIM AGE VS DEATH OR NOT'
plt.title(title, loc='center', fontsize=18)

```

Out[43]: Text(0.5, 1.0, 'VICTIM AGE VS DEATH OR NOT')



จากผลลัพธ์ของกราฟ heatmap แสดงให้เห็นว่า

- ช่วงอายุ 18-24 ปี มีจำนวนการอุดช่องสูดที่สุดในทุกช่วงอายุมีจำนวน 157 cases รองลงมาคือ ช่วงอายุ 25-44 ปี และต่ำสุดคือ 65 ปีขึ้นไป
- ช่วงอายุ 25-44 ปี มีจำนวนการเสียชีวิตสูงที่สุดในทุกช่วงอายุมีจำนวน 136 cases รองลงมาคือ ช่วงอายุ 18-24 ปี และต่ำสุดคือ 65 ปีขึ้นไป
- ช่วงอายุของเหยื่อมีจำนวนการอุดช่องสูดกว่าที่สูงกว่าเสียชีวิตอย่างมาก สังเกตได้จากสีที่เข้ม(มีจำนวนมาก) กับ สีที่อ่อน(มีจำนวนน้อย)

## 7. เชื้อชาติของเหยื่อที่เสียชีวิตมากที่สุด

ต้องการรู้เคราะห์ว่าเชื้อชาติของเหยื่อที่เสียชีวิตมากที่สุด ด้วย heatmap

```

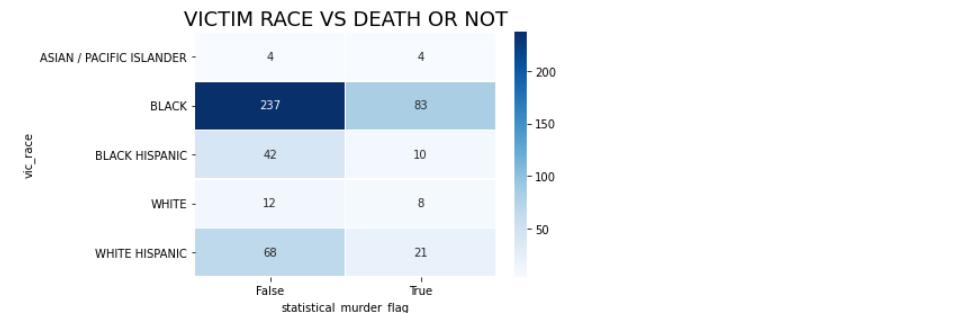
In [44]: # ใน death_race เป็นตัวแปรเดิม crosstab ระหว่างคอลัมน์ vic_race และ statistical_murder_flag
death_race = pd.crosstab(df_shooting['vic_race'], df_shooting['statistical_murder_flag'])

sns.heatmap(pd.crosstab(df_shooting['vic_race'], df_shooting['statistical_murder_flag']),
            annot=True, cbar=True, fmt="d", linewidth=0.3, cmap = "Blues")

# ticks
yticks = [i.upper() for i in death_race.index] # ให้ yticks เป็นชื่อ默 list of i ด้วยทั้งหมดที่ i เป็น index ของข้อมูลภายใน death_race
xticks = [i for i in death_race.columns] # ให้ xticks เป็นชื่อ默 list of i ด้วยที่ i เป็น columns ของข้อมูลภายใน death_race
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)
# title
title = 'VICTIM RACE VS DEATH OR NOT'
plt.title(title, loc='center', fontsize=18)

```

Out[44]: Text(0.5, 1.0, 'VICTIM RACE VS DEATH OR NOT')



จากผลลัพธ์ของกราฟ heatmap แสดงให้เห็นว่า

- เชื้อชาติ BLACK มีจำนวนการอุดช่องสูดที่สุดมีจำนวน 237 cases รองลงมาคือ WHITE-HISPANIC และต่ำสุดคือ ASIAN/PACIFIC ISLANDER
- เชื้อชาติ BLACK มีจำนวนการเสียชีวิตสูงที่สุดมีจำนวน 83 cases รองลงมาคือ WHITE-HISPANIC และต่ำสุดคือ ASIAN/PACIFIC
- เชื้อชาติของเหยื่อมีจำนวนการอุดช่องสูดที่สูงกว่าเสียชีวิตอย่างมาก สังเกตได้จากสีที่เข้ม(มีจำนวนมาก) กับ สีที่อ่อน(มีจำนวนน้อย)

- เผือกเชื้อชาติ 黑人 มีจำนวนการจดชื่อว่า แลและเสียชีวิตมากกว่าเผือกเชื้อชาติ แต่มีจำนวนการจด มากกว่า เสียชีวิต
- เมืองที่จดชื่อว่าสูงที่สุดคือชาวอาชญากร 18-24 ปี รองลงมาคือ ชาวอาชญากร 25-44 ปี และต่ำสุดคือ 65 ปีขึ้นไป
- เมืองที่เสียชีวิตสูงที่สุดคือชาวอาชญากร 25-44 ปี รองลงมาคือ ชาวอาชญากร 18-24 ปี และต่ำสุดคือ 65 ปีขึ้นไป
- เมืองที่มีเชื้อชาติ BLACK มีจำนวนการจด แลและเสียชีวิตสูงที่สุด รองลงมาคือ WHITE-HISPANIC และต่ำสุดคือ ASIAN/PACIFIC ISLANDER ตั้งหนึ่งจะแสดงได้ว่า เผือกเชื้อชาติ 黑人 มีชาวอาชญากร 18-24 ปี และเมืองที่มีเชื้อชาติ 黑人 มีจำนวนการเสียชีวิตที่มากกว่า ชาวอาชญากร 18-24 ปี และเมืองที่มีเชื้อชาติ 黑人 มีชาวอาชญากร 25-44 ปี และเมืองที่มีเชื้อชาติ 黑人 มีจำนวนการเสียชีวิตที่มากกว่า ชาวอาชญากร 25-44 ปี และในเรื่องของสภาวะแทรกซ้อนต่อการได้รับบาดเจ็บที่มีมากกว่าเมืองอาชญากร สูงขึ้น

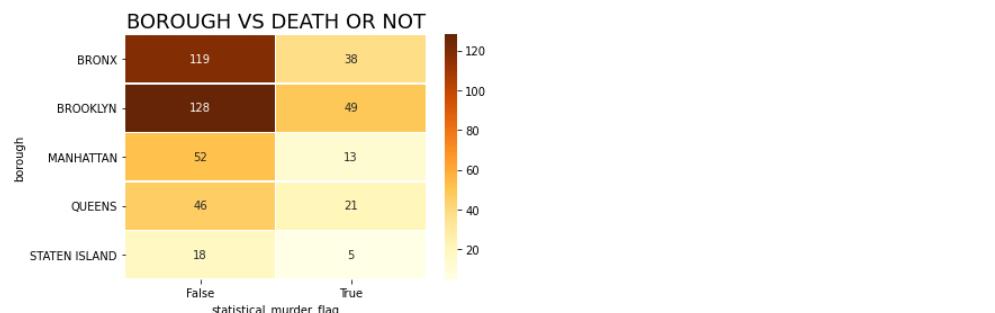
## 8.เขตในเมือง NYC ที่มีจำนวนการเสียชีวิตมากที่สุด

ต้องการวิเคราะห์ว่าแต่ละเขตใน New York City เขตไหนจำนวนการเสียชีวิตมากที่สุด

```
In [45]: # ใน death_borough เป็นตัวแปรเก็บ crosstab ระหว่าง columm 'borough' และ 'statistical_murder_flag'
death_borough = pd.crosstab(df_shooting['borough'], df_shooting['statistical_murder_flag'])

sns.heatmap(pd.crosstab(df_shooting['borough'], df_shooting['statistical_murder_flag']),
            annot=True, cbar=True, fmt="d", linewidth=0.3, cmap = 'YlOrBr')
# ticks
yticks = [i.upper() for i in death_borough.index] # ใน yticks ให้เป็น list of i ตัวใหญ่ทั้งหมดโดยที่ i เป็น index ของข้อมูลภายใน death_borough
xticks = [i for i in death_borough.columns] # ใน xticks ให้เป็น list of i โดยที่ i เป็น columns ของข้อมูลภายใน death_borough
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)
# title
title = 'BOROUGH VS DEATH OR NOT'
plt.title(title, loc='center', fontsize=18)
```

Out[45]: Text(0.5, 1.0, 'BOROUGH VS DEATH OR NOT')



จากผลลัพธ์ของกราฟ heatmap แสดงให้เห็นว่า

- เขต Brooklyn มีจำนวนการจดชื่อว่าสูงที่สุดคือ 128 cases รองลงมาคือ Bronx และต่ำสุดคือ Staten Island
- Brooklyn มีจำนวนการเสียชีวิตที่สูงที่สุดคือ 49 cases รองลงมาคือ Bronx และต่ำสุดคือ Staten Island
- ในแต่ละเขตมีจำนวนการจดชื่อว่ามากกว่าเสียชีวิต จะสังเกตเห็นได้ว่าเขต Brooklyn มีจำนวนการจด แลและเสียชีวิตมากที่สุด เมื่อจากเขตที่มีขนาดของประชากร หนาแน่นที่สุดในเมือง NYC จึงทำให้มีจำนวนผู้ถูกยิงมากกว่าเขตอื่นๆ

## 9.พิกัดของผู้จดชื่อว่าในแต่ละเขตของเมือง NYC

ต้องการสร้างแผนที่แสดงพิกัดทางภูมิศาสตร์เพื่อว่าพิกัดตำแหน่งของผู้จดชื่อว่าในแต่ละเขตเพื่อเป็นการอุ่นภาพรวม

```
In [46]: #Map
# ใน map_hooray เป็น map ที่มี location คือ ละติจูด และลองจิจูด ที่เป็น center ของแผนที่ของสถานที่ที่ต้องการด้วยคำสั่ง folium.Map()
map_hooray = folium.Map(location=[40.730610, -73.935242], zoom_start=10)

#Marker
#สร้าง marker ของแต่ละเขต โดยนำที่ location เป็นละติจูด และลองจิจูดของเขตนั้นๆ ด้วยคำสั่ง folium.Marker()
#และทำการเพิ่มเข้าไปใน map_hooray ด้วยคำสั่ง .add_to(map_hooray)
#popup คือ ต้องการที่ Marker ที่มีหน้าที่ในการคลิกบน Marker นั้น
#icon=folium.Icon(color='color') ก็หมายความว่า Marker นั้น
folium.Marker([40.650002, -73.949997], popup=(('Brooklyn'+ '\n' + 'Number 1 survival'), icon=folium.Icon(color='green')).add_to(map_hooray)

folium.Marker([40.837048, -73.865433], popup=(('Bronx'+ '\n' + 'Number 2 survival'), icon=folium.Icon(color='red')).add_to(map_hooray)

folium.Marker([40.742054, -73.769417], popup=(('Queens'+ '\n' + 'Number 4 survival'), icon=folium.Icon(color='pink')).add_to(map_hooray)
```

```

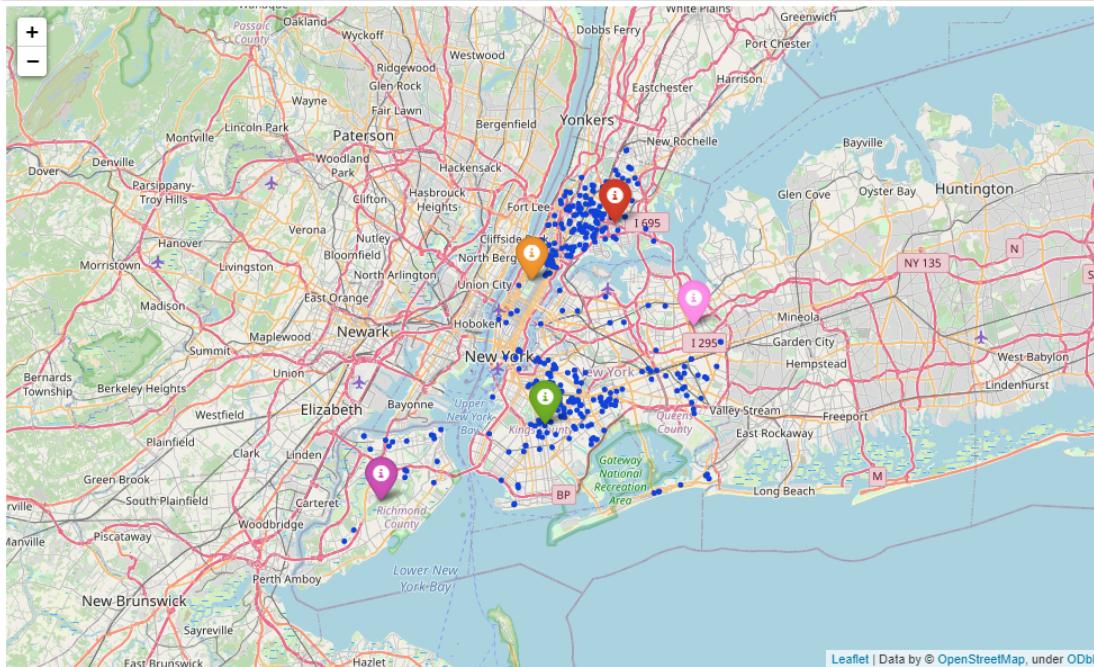
folium.Marker([40.78343, -73.96625], popup='Manhattan'+'\n' + 'Number 3 survival', icon=folium.Icon(color='orange')).add_to(map_hooray)

folium.Marker([40.579021, -74.151535], popup='Staten Island'+'\n' + 'Number 5 survival', icon=folium.Icon(color='purple')).add_to(map_hooray)

# ໃຊ້ນຸ່ມ for ໃປໃນ datafram df_shooting ຄວ່າມີ statistical_murder_flag ທີ່ມີຄວນເປັນ false ອ່ອຍ ຮອດຫຼືກ
for (index, row) in df_shooting[df_shooting['statistical_murder_flag'] == False].iterrows():
    folium.Circle(location=[row['latitude'], row['longitude']], radius=10,color = "#1142D8", fill_color="#1142D8").add_to(map_hooray)
# ໃປເລີ່ມທີ່ມີຄວນການມີ map_hooray ດ້ວຍຄວ່າມີ folium.Circle()
# ໃນ location = row ຂອງຄວ່າມີ latitude ແລະ longitude
# radius ຕີ້ວ່າ ຂາດຂອງການລົມພິທີ່, color ຕີ້ວ່າ ສ້າງການລົມ fill_color ຕີ້ວ່າ ຈະໄສສີ່ຫຼາມໃນວັກລົມທ່ານໄສສີ່ຈະນີເປົ້າແລ້ວຢັນແປງ
map_hooray

```

Out[46]:



ຈາກຈຸດ Marker ສີ່ຈຳຕຸ ແລ້ວໃຫ້ເກີດແຕລະບົດໃນ NYC ຕັ້ງນີ້

- ສີ່ຈຳ ແຂດ Bronx
- ສີ່ຈຳ ແຂດ Manhattan
- ສີ່ຈຳ ແຂດ Queens
- ສີ່ຈຳ ແຂດ Brooklyn
- ສີ່ຈຳ ແຂດ Staten Island

ໂດຍຈຸດສີ່ຈຳເງິນແຕລະຈະແສດງທີ່ກົດຂອງຄວນທີ່ອຳນວຍໃນເຂດແຕລະບົດເກີດໃນ NYC ຂຶ້ງຈາກຄລສັບໃນກາຣີເຄຣະເກົກຂ້າງຂໍ້ວ່າ 8 ແລ້ວໃຫ້ນວ່າ ເກີດ Brooklyn ອ່ອຍ Marker ສີ່ເກີດ ມີຈຳເນັນຖຸເສີຍຫຼືມາກທີ່ສຸດ ຮອງລົມມາເຖີ່ມ Bronx Manhattan Queens ແລະ Staten Island ດາວລາຕົ້ນ

## 10.ພິກັດຂອງຜູ້ເສີຍຫຼືມາກໃນແຕລະບົດຂອງເມືອງ NYC

ຕ່ອງກາຮ່າງແນວທີ່ແສດງພິກັດທາງກົມືສາດຮ່ວມທີ່ກົດຕ່າແໜ່ງຂອງຜູ້ເສີຍຫຼືມາກໃນແຕລະບົດເກີດເປົ້າເປັນກາພຽງ

In [47]:

```

#Map
map_hooray = folium.Map(location=[40.730610, -73.935242], zoom_start=10)

#Marker
folium.Marker([40.650002, -73.949997],('Brooklyn'+'\n' + 'Number 1 death'),icon=folium.Icon(color='green')).add_to(map_hooray)

folium.Marker([40.837048, -73.865433],('Bronx'+'\n' + 'Number 2 death'),icon=folium.Icon(color='red')).add_to(map_hooray)

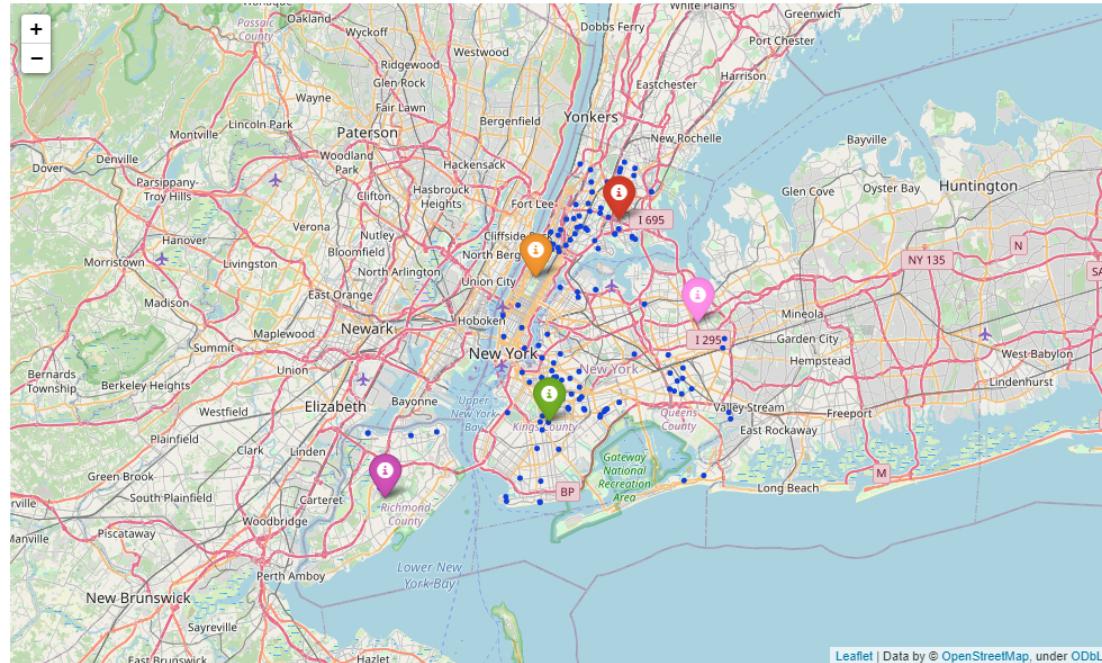
folium.Marker([40.742054, -73.769417],('Queens'+'\n' + 'Number 3 death'),icon=folium.Icon(color='pink')).add_to(map_hooray)

folium.Marker([40.78343, -73.96625],('Manhattan'+'\n' + 'Number 4 death'),icon=folium.Icon(color='orange')).add_to(map_hooray)

```

```
# ให้รันบล็อก for ไปใน dataframe df_shooting คอลัมน์ statistical_murder_flag ที่มีค่าเป็น true หรือ เสียชีวิต
for (index, row) in df_shooting[df_shooting['statistical_murder_flag'] == True].iterrows():
    folium.Circle(location=[row['latitude'], row['longitude']], radius=10, color='red', fill_color='red').add_to(map_hooray)
map_hooray
```

Out[47]:



จากจด Marker สีต่างๆ แสดงให้เห็นถึงเขตแต่ละเขตใน NYC ดังนี้

- สีแดง เชด Bronx
  - สีส้ม เชด Manhattan
  - สีเขียว เชด Queens
  - สีเขียว เชด Brooklyn
  - สีม่วง เชด Staten Island

โดยเจลสีน้ำเงินแต่ละจุดจะแสดงที่เกิดของคนที่เสียชีวิตในเขตแต่ละเขตของเมือง NYC ซึ่งจากผลลัพธ์ในการเรียก救急จากห้าข้อ 8 และลงให้เห็นว่า เมือง Brooklyn หรือ Bronx ที่เสียชีวิตมากที่สุด รองลงมาคือ Bronx Queens Manhattan และ Staten Island ตามลำดับ

#### 11. กราฟจำนวนการเกิดเหตุการณ์เมืองใน NYC ของแต่ละปี

ผู้คนจำนวนมากใน NYC ยังคงเดินทางไปทำงานและใช้ชีวิตอย่างปกติ ไม่คำนึงถึงสถานการณ์การแพร่ระบาดที่รุนแรง

```
In [48]: df_shooting.groupby('Year').count() #ใช้คำสั่ง df_groupby('column name').count()  
#เพื่อที่การรวมข้อมูลภายใต้แต่ละปีของคอลัมน์ Year
```

Out[48]:

2014	39	39	39	39	39	39	39	39	39	39	39	39
2015	28	28	28	28	28	28	28	28	28	28	28	28
2016	22	22	22	22	22	22	22	22	22	22	22	22
2017	30	30	30	30	30	30	30	30	30	30	30	30
2018	28	28	28	28	28	28	28	28	28	28	28	28

จากผลลัพธ์ด้านบนทุก colum มีค่าเท่ากันจึงเลือกมา 1 colum ก่อนเพื่อไปงาน plot กราฟ

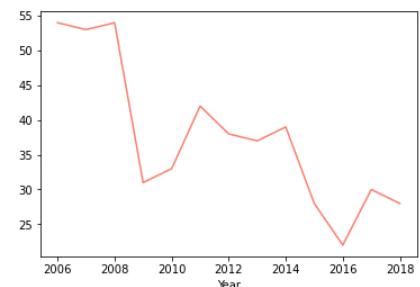
```
In [49]: #ทำการเลือก colum นี้ incident_key มา plot ด้วยคำสั่ง df.groupby('column name1')[['column name2']].count()
df_shooting.groupby('Year')[['incident_key']].count()
```

Out[49]: Year

2006	54
2007	53
2008	54
2009	31
2010	33
2011	42
2012	38
2013	37
2014	39
2015	28
2016	22
2017	30
2018	28

Name: incident\_key, dtype: int64

```
In [50]: #ใช้คำสั่ง df.groupby('column name1')[['column name2']].count().plot() เพื่อ plot ให้มีกราฟเส้น
df_shooting.groupby('Year')[['statistical_murder_flag']].count().plot(color='salmon');
```



จากราฟที่แสดงด้านบน จะสังเกตเห็นได้ว่าในปี 2006 และ 2008 มีจำนวนของการเกิดเหตุการณ์ตายที่สูงโดยมีจำนวน 54 cases และค่อยๆลดลงมาเรื่อยๆจนถึงปี 2009 และค่อยๆเพิ่มสูงขึ้นในปี 2011 โดยมีจำนวน 42 cases และค่อยๆลดลงและเพิ่มสูงขึ้นอีกในปี 2014 มีจำนวน 39 cases ซึ่งน้อยกว่าปีที่ผ่านมา จากนั้นมีการลดจำนวน การเกิดเหตุลงจนถึงปี 2016 มีจำนวนทั้งสิ้น 22 cases และมีจำนวนสูงขึ้นอีกในปี 2017 มีเพียง 30 cases และลดลงในปี 2018

## 12.วิเคราะห์แต่ละปีมีจำนวนการเสียชีวิตเป็นอย่างไร

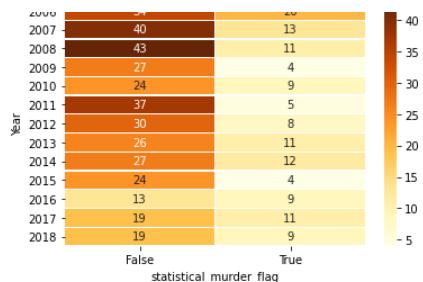
ต้องการวิเคราะห์ว่าแต่ละปีเกิดเหตุการณ์ตายมีจำนวนกี่ครั้งบ้าง และรอดชีวิตเป็นอย่างไร

```
In [51]: # ให้ year_death เป็นชื่อมบที่ crossstab จะหาผลลัพธ์ Year และ statistical_murder_flag
year_death = pd.crosstab(df_shooting['Year'],df_shooting['statistical_murder_flag'])

sns.heatmap(pd.crosstab(df_shooting['Year'], df_shooting['statistical_murder_flag']),
            annot=True, cbar=True,fmt="d", linewidth=0.5, cmap = 'YlOrBr')
# ticks
yticks = [i.upper() for i in year_death.index] # ให้ yticks เก็บข้อมูล list of i ที่ใหญ่ที่สุดมดโดยที่ i เป็น index ของข้อมูลภายใน year_death
xticks = [i for i in year_death.columns] # ให้ xticks เก็บข้อมูล list of i โดยที่ i เป็น columns ของข้อมูลภายใน year_death
plt.yticks(plt.yticks()[0], labels=yticks, rotation=0)
plt.xticks(plt.xticks()[0], labels=xticks, rotation=0)
# title
title = 'YEAR VS DEATH OR NOT'
plt.title(title, loc='center', fontsize=18)
```

Out[51]: Text(0.5, 1.0, 'YEAR VS DEATH OR NOT')

YEAR VS DEATH OR NOT

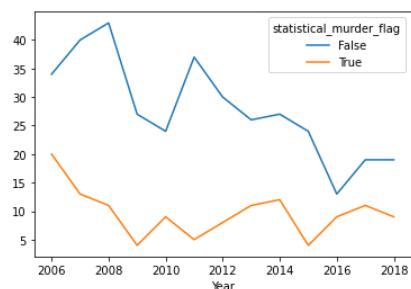


จากผลลัพธ์กราฟ heat map จะพบว่า

- ปี 2008 มีจำนวนผู้รอดชีวิตมากที่สุด รองลงมาคือปี 2007 ต่ำสุดในปี 2013
- ปี 2006 มีจำนวนผู้เสียชีวิตมากที่สุด รองลงมาคือปี 2006 ต่ำสุดในปี 2009 และ 2015
- แต่ทุกปีมีจำนวนของผู้รอดชีวิตมากกว่าเสียชีวิต

```
In [52]: year_death.plot() #ทำการสร้างฟาร์มาเพื่อนของตัวแปร year_death ที่เก็บ crosstab ระหว่างคอลัมน์ Year และ statistical_murder_flag
```

```
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1c1cf9ed5e0>
```



จากผลลัพธ์ของกราฟเห็นได้ว่าจำนวนของผู้รอดชีวิตมีจำนวนสูงที่สุดในปี 2008 ในขณะที่จำนวนผู้เสียชีวิตในปี 2006 มีจำนวนสูงที่สุด และสังเกตได้ว่าในปีหลังๆ ทั้งจำนวนของผู้รอดชีวิต และเสียชีวิตมีจำนวนลดลงจากปีก่อนๆ หรือ มองได้ว่าจำนวนการเกิดเหตุการณ์เป็นกันค่อยๆ ลดลง สำหรับจำนวนผู้รอดชีวิตค่อนข้างเริ่มคงที่อยู่ที่ 19 cases และจำนวนผู้เสียชีวิตมีแนวโน้มลดลงใน 2 ปีหลังคือปี 2017 และ 2018

```
In [ ]:
```

```
In [ ]:
```