NYC Crime Data Analysis

CS 673 Scalable Databases - Fall 2024 - Midterm Presentation
Instructor: Professor Anthony Escalona
Github: https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

Group Members

- Joshua Gottlieb (jg05394n@pace.edu)
- Carlos Correa (cc95353n@pace.edu)
- Anchal Rai (ar65667n@pace.edu)
- Martin Chunsen (cs83339n@pace.edu)

Presentation Structure

Objectives

- Identify the most common crimes and crime types in NYC
- Uncover possible correlation between complaints and arrests
- Gain insight into crime rate demographics to help identify most likely suspect/victim profiles to help inform citizens and police
- Develop an understanding of the safety of the boroughs of NYC and the effectiveness of precincts to help determine which boroughs are the safest or most dangerous, and where possible increased police funding is required.

Data Sets

NYPD Historic Arrest, Complaints and Shooting Data (2006 to 2023):

- https://data.cityofnewyork.us/Public-Safety/NYPD-Arrests-Data-Historic-/8h9b-rp9u/about_data
- https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i/about_data
- https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8/about_data

New York City Borough Population Data (Real and Projected, 1950-2040)

- https://data.cityofnewyork.us/City-Government/New-York-City-Population-by-Borough-1950-2040/xywu-7bv9/about_data

After cleaning, data was put into a SQLite database for built-in compatibility with Pandas, allowing us to directly extract the results of queries into dataframes.

Challenges Faced

Data Cleaning:

- Validating bad data and normalizing data
  - Bad values such as negative ages, multiple descriptions per offense code, year values from before 2006 (data starts in 2006 according to documentation)
  - Imputing missing data such as race, sex, and age information so as not to bias data
- Lack of information available about data
  - No names for precincts, only codes
- Lack of funding data

Data Sharing and GitHub limits:

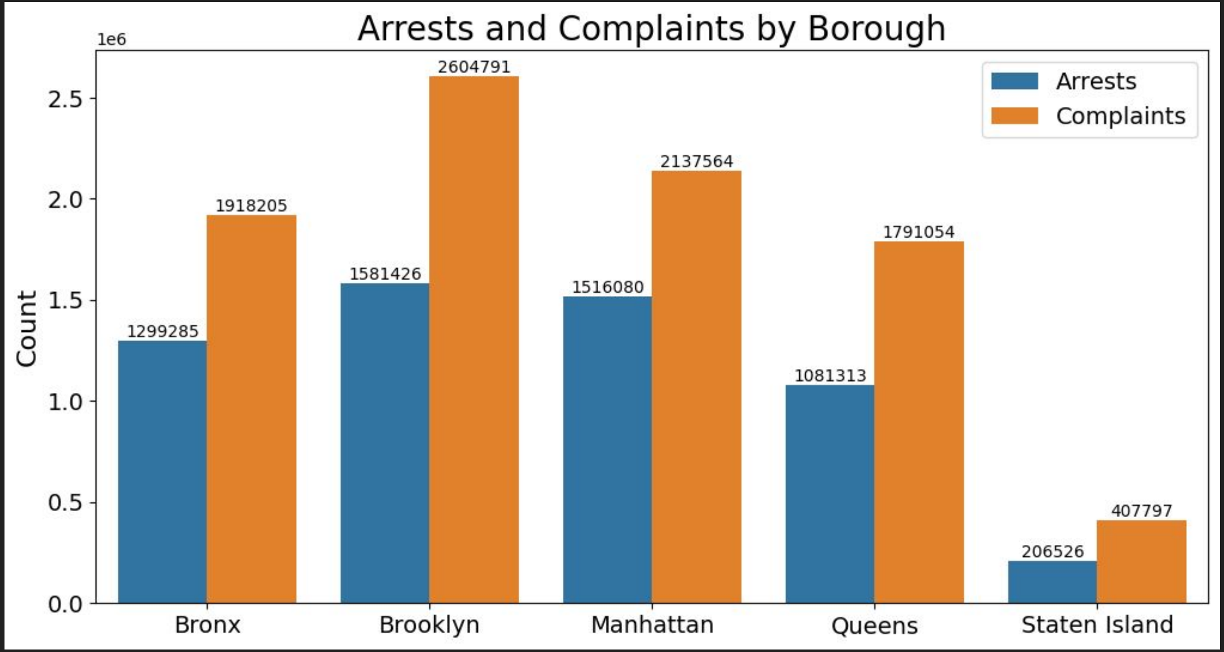
- Data files and SQLite database too large for GitHub
  - GitHub has a 100 MB file size limit - initial data files and constructed database were too large for GitHub, so we used Google Drive to share data that was too large

Borough Statistics - Most Common Crime

	borough	crime_type	crime_count	rank
0	B	PETIT LARCENY	264063	1
1	K	PETIT LARCENY	425510	1
2	M	PETIT LARCENY	487041	1
3	Q	PETIT LARCENY	311016	1
4	S	HARRASSMENT 2	79403	1

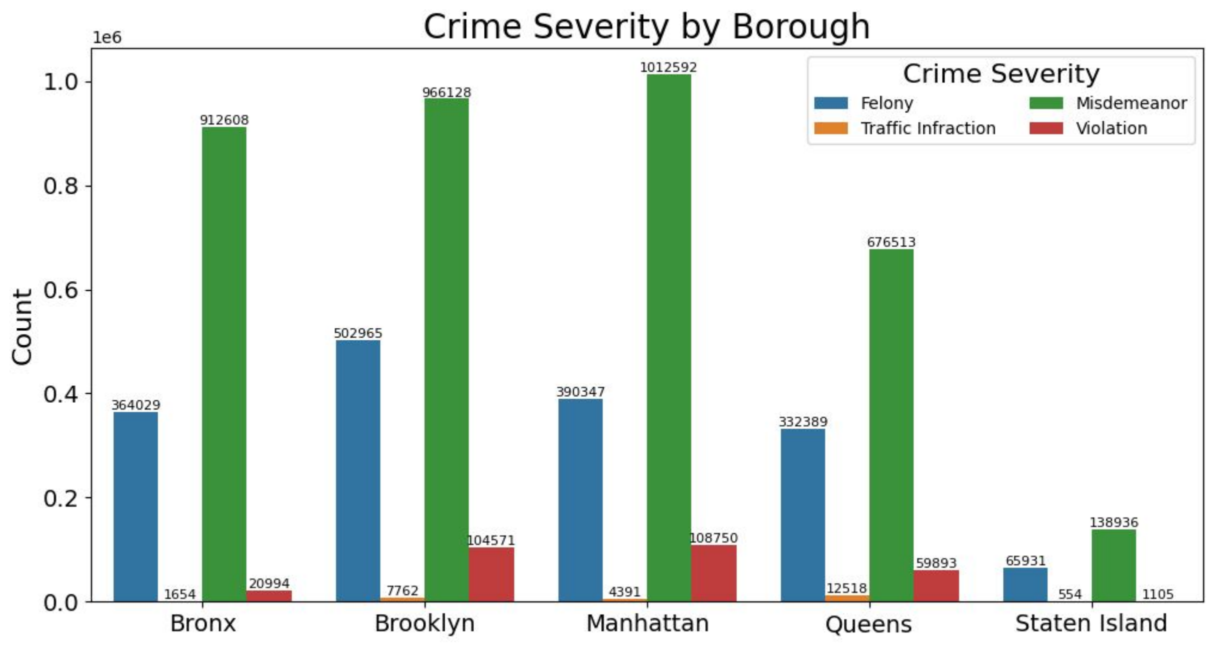
In 4 of the 5 Boroughs (Bronx (B), Brooklyn (K), Manhattan (M), Queens (Q)) Petit Larceny is the most common crime, while in Staten Island (S) Harassment 2 is the most common crime.

Borough Statistics - Arrests and Complaints



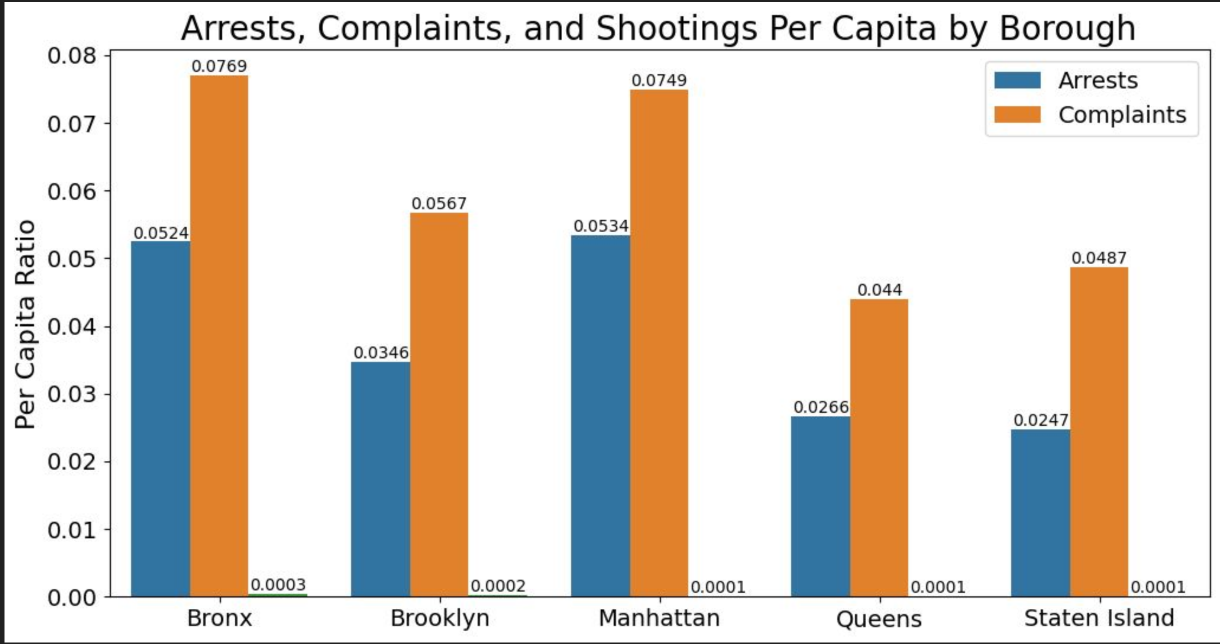
Manhattan has the highest ratio of arrests:complaints with 70.92%, followed by the Bronx with 67.73%. Brooklyn and Manhattan are the most active boroughs.

Borough Statistics - Crime Severity



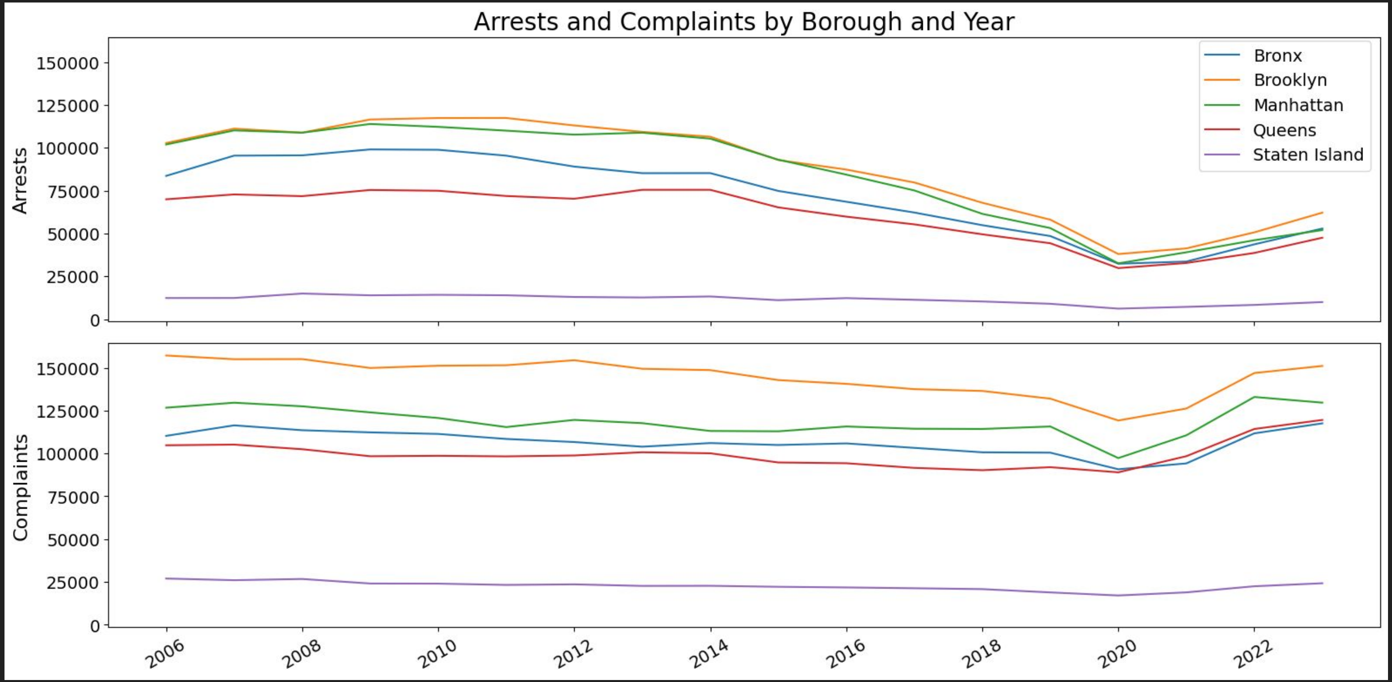
The majority of the crimes at the boroughs are misdemeanors followed by felonies. Brooklyn has the highest felony rate.

Borough Statistics - Per Capita Statistics



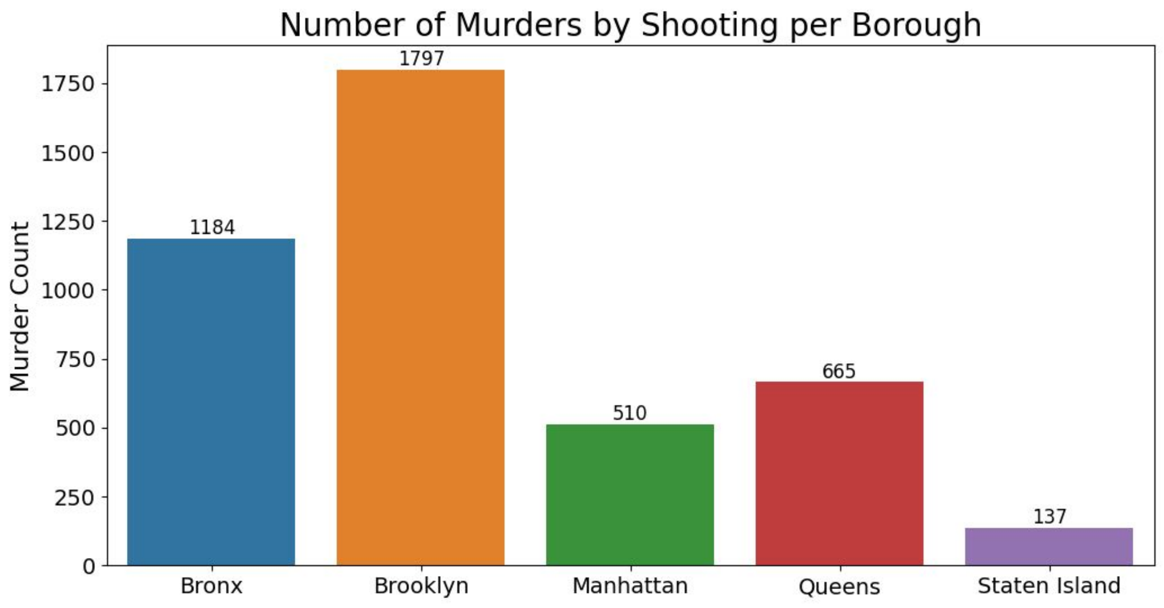
Manhattan and the Bronx have the highest crime rates per capita.

Borough Statistics - Arrests and Complaints by Year



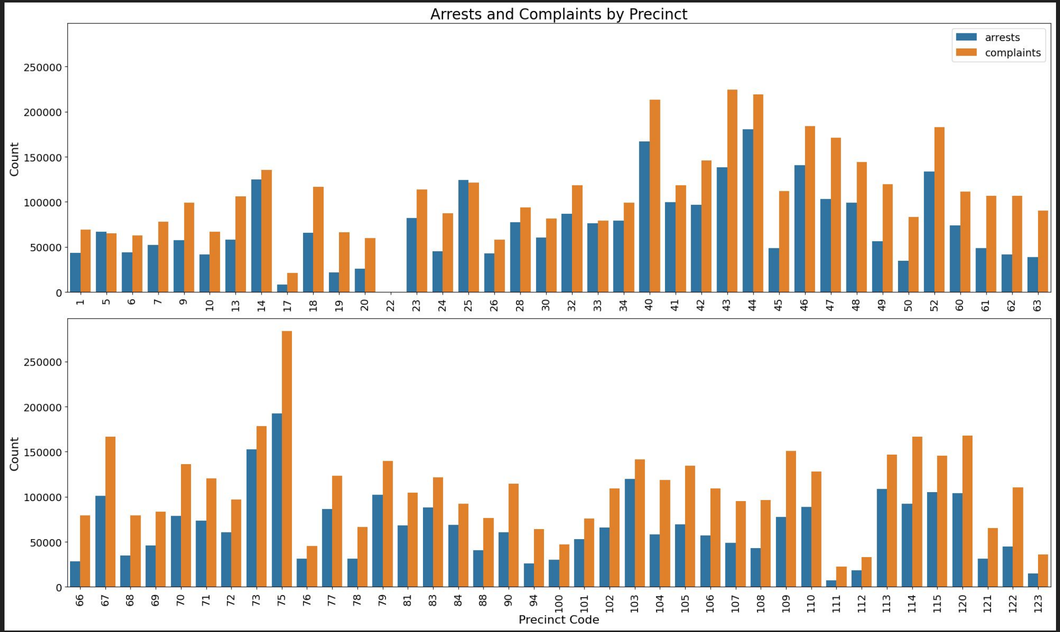
There is a noticeable decline in arrest rates post-2018, but the complaint rate keeps growing. Crimes are not leading to arrests - perhaps the police are underfunded or understaffed.

Borough Statistics - Murders by Shootings



Although there are not many shootings in New York City, Brooklyn has had the most murders by shooting in the span of the data (2006-2023).

Precinct Statistics - Arrests and Complaints, All Precincts



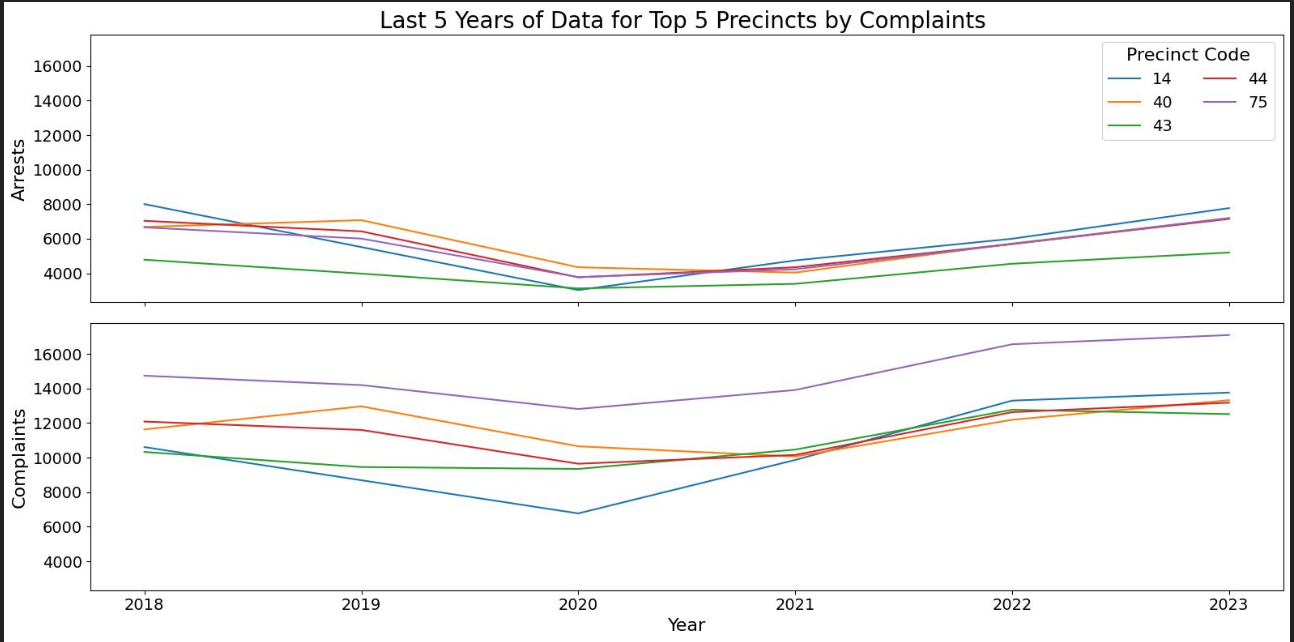
The majority of the precincts show a higher number of complaints than actual arrests. The exception is precinct 5 (Manhattan-Chinatown) which has more arrest than complaints.

Precinct Statistics - Top 5 Most Active Precincts

Precinct Boroughs:

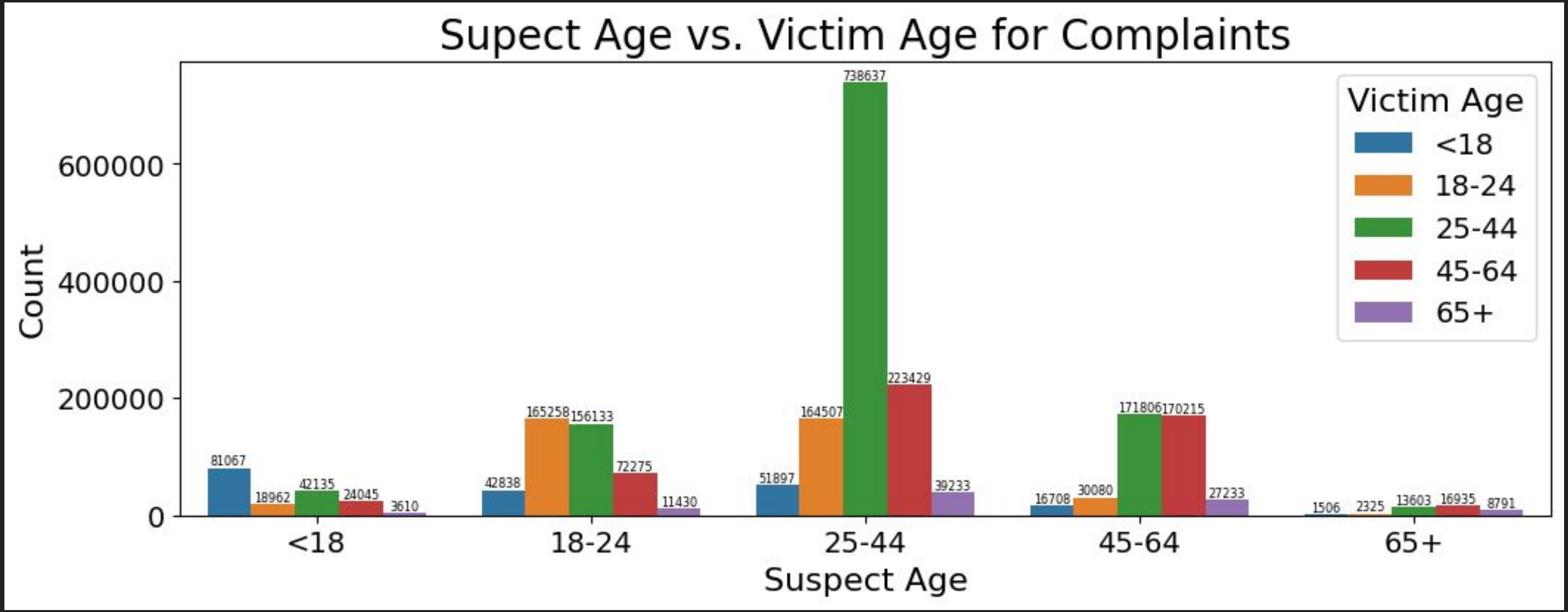
- 14: Manhattan
- 40: Bronx
- 43: Bronx
- 44: Bronx
- 75: Brooklyn

(NYPD, Precincts, n.d.)



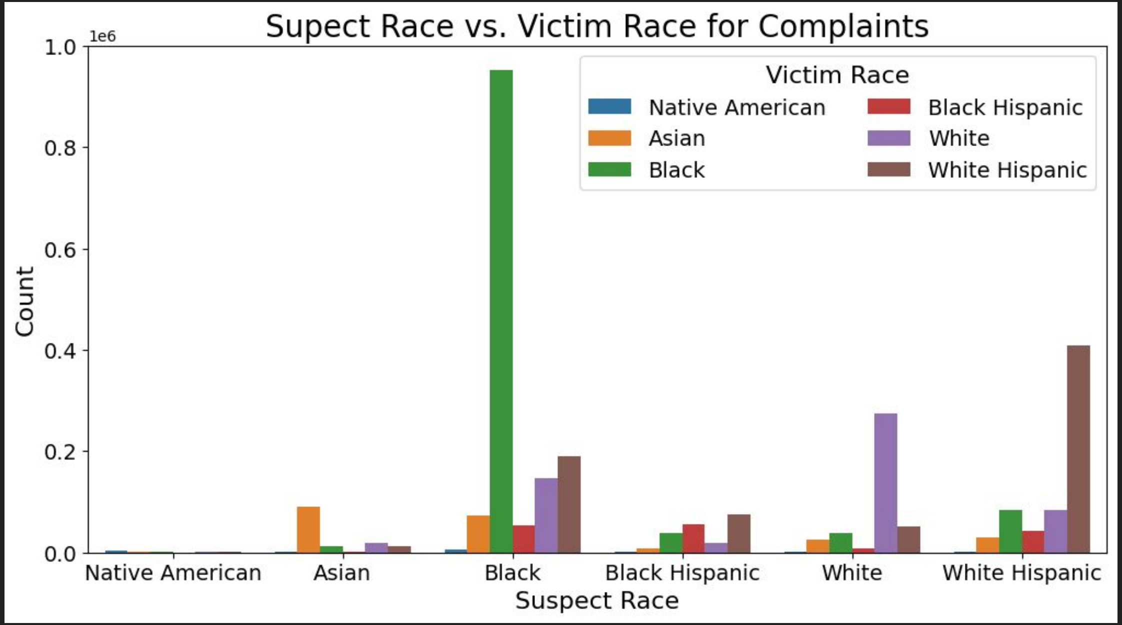
These 5 precincts have had a noticeable increase in complaints, but a lower rate of arrests, indicating they may be lacking in manpower.

Crime Demographics: Age Profiles



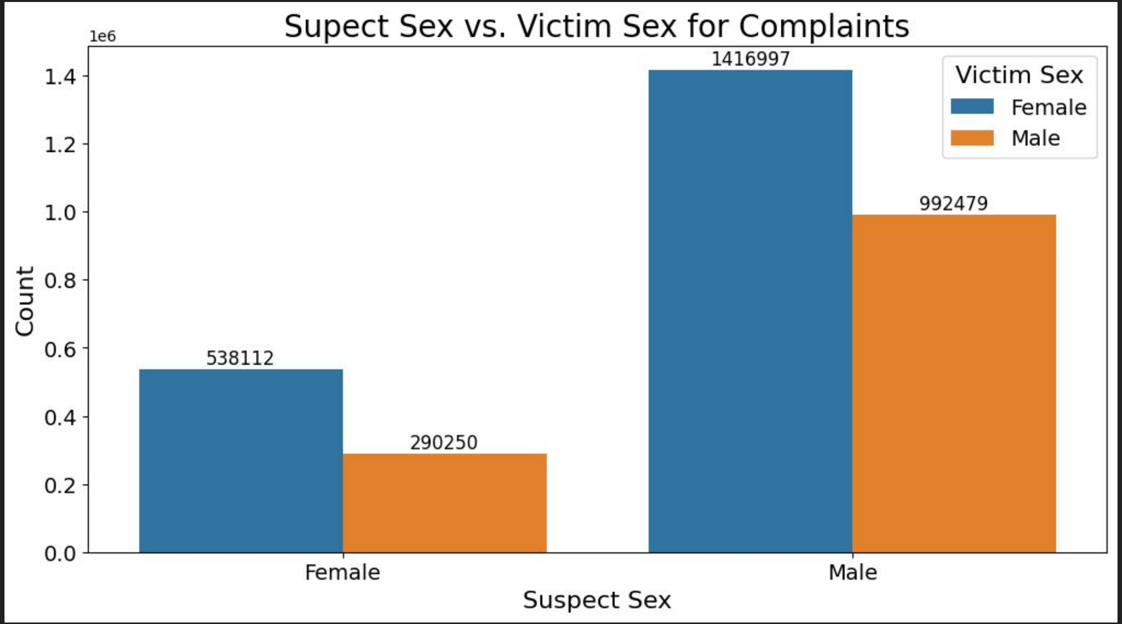
Suspects tend to choose victims within the same age group, or one age group away. The majority of complaints fall within the 24-44 age group.

Crime Demographics: Race Profiles



In most cases, the suspect and victim are of the same race.

Crime Demographics: Sex Profiles



Women are more likely to be victims. 64% of female suspects choose female victims, compared to 41% of males choosing male victims.

Conclusions

- Most of the crime around the five boroughs is petit larceny which means that the stolen property had a value of less than $1,000 (applies for New York).
  - This type of crime will result in a criminal record, with a maximum of up to one year of jail time, but most of the time does not result in jail time for 1st time offenders. (Bassett J., n.d.)
- Most offenders shared the same racial and age demographics as their victims. Most offenders are male, and females are more frequently victims of both male and female offenders. These findings coincide with studies done by the U.S. Department of Justice (Morgan & Thompson, 2022, Tapp & Thompson, 2023).
- Brooklyn is the least safe borough by crime severity (most felonies). The Bronx and Manhattan are the least safe based on per capita crime rates.

Future Steps

- It would be beneficial to find a dataset showing each precinct's funding to help diagnose whether the funding is effective or whether certain precincts are under-funded. Funding data could be beneficial for answering why some precincts have a higher complaint-arrest ratio.
- Currently, all felonies/misdemeanors/etc. are grouped together. There is a large difference between fraud felony charges and murder felony charges, so further cleaning to create sub-categories would be useful for more comprehensive analysis.
- Performing second and third level relationship analysis on demographic trends could yield deeper insights (age + sex vs. age + sex, for example).
- Perform modeling on the data, such as logistic regression or decision tree modeling to try to predict different aspects of crimes based on other features present in the data.

References - Data Sets

Department of City Planning. (2022, May 9). New york city population by borough, 1950-2040. NYC OpenData. Retrieved October 14, 2024 from https://data.cityofnewyork.us/City-Government/New-York-City-Population-by-Borough-1950-2040/xywu-7bv9/about_data

New York Police Department. (2024, April 23). NYPD arrests data historic. NYC OpenData. Retrieved October 14, 2024 from https://data.cityofnewyork.us/Public-Safety/NYPD-Arrests-Data-Historic-/8h9b-rp9u/about_data

New York Police Department. (2024, April 23). NYPD complaints data historic. NYC OpenData. Retrieved October 14, 2024 from https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i/about_data

New York Police Department. (2024, April 23). NYPD shooting incident data historic. NYC OpenData. Retrieved October 14, 2024 from https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8/about_data

References - External

Basset, J. (n.d.). What is the difference between petit and grand larceny in new york? JBassettLaw.com. Retrieved November 11, 2024 from https://jbassettlaw.com/what-is-the-difference-between-petit-and-grand-larceny-in-new-york/

Morgan, R. & Thompson, A. (2022). Criminal victimization, 2020 - Supplemental statistical tables. U.S. Department of Justice. Retrieved November 11, 2024 from https://bjs.ojp.gov/content/pub/pdf/cv20sst.pdf

New York City Police Department. (n.d.). Precincts landing page. NYC.gov. Retrieved November 11, 2024 from https://www.nyc.gov/site/nypd/bureaus/patrol/precincts-landing.page

Tapp., S & Thompson, A. (2023). Criminal victimization, 2022. U.S. Department of Justice. Retrieved November 11, 2024 from https://bjs.ojp.gov/document/cv22.pdf

Types of criminal cases. (n.d.). NYCourts.gov. Retrieved November 11, 2024 from https://www.nycourts.gov/courthelp/criminal/typesCriminalCases.shtml

Code Snippets Cleaning https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
# Check for missing columns
# Highly missing: > 90% missingness
# Moderately missing: 1-90% missingness
# Low missing: < 1% missingness
highly_missing, moderately_missing, low_missing = cut.calculate_missing_ratios(df)

# 5 columns with high missingness
highly_missing
```

```
PARKS_NM            0.995787
HADEVELOPT          0.996450
HOUSING_PSA         0.924798
TRANSIT_DISTRICT    0.977783
STATION_NAME        0.977783
dtype: float64
```

```
# Drop these columns that are mostly null, as the data is irrecoverable
df.drop([*[x for x in df.columns if x in highly_missing.index.tolist()]], axis = 1, inplace = True)
```

```
def calculate_missing_ratios(df):
    missing_ratios = df.isnull().sum() / len(df.index)
    highly_missing = missing_ratios[missing_ratios > 0.9]
    moderately_missing = missing_ratios[(missing_ratios > 0.01) & (missing_ratios < 0.9)]
    low_missing = missing_ratios[(missing_ratios <= 0.01) & (missing_ratios > 0)]

    return highly_missing, moderately_missing, low_missing
```

Dealing with Missing Values

```
# Some columns with moderate missingness
# All of these columns are age, sex and race columns
moderately_missing
```

```
SUSP_AGE_GROUP    0.514144
SUSP_RACE         0.413587
SUSP_SEX          0.428546
VIC_AGE_GROUP     0.182123
dtype: float64
```

```
# Validate age, sex, and race columns, converting invalid values to UNKNOWN or U
# Validate all columns while we are at it
df = cut.validate_age(df, ['SUSP_AGE_GROUP', 'VIC_AGE_GROUP'])
df = cut.validate_sex(df, ['SUSP_SEX', 'VIC_SEX'])
df = cut.validate_race(df, ['SUSP_RACE', 'VIC_RACE'])
```

```
# Rather than introduce biases via imputation, we will drop these records
low_missing
```

```
ADDR_PCT_CD         0.000034
OFNS_DESC           0.002118
PD_CD               0.000842
PD_DESC             0.000842
CRM_ATPT_CPTD_CD    0.000019
BORO_NM             0.000884
VIC_RACE            0.000075
VIC_SEX             0.000035
dtype: float64
```

Code Snippets Cleaning https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
def validate_age(df, cols):
    valid_age_ranges = ['UNKNOWN', '<18', '18-24', '25-44', '45-64', '65+']

    for c in cols:
        df[c] = df[c].apply(lambda x: 'UNKNOWN' if not any(x == y for y in valid_age_ranges) else x)

    return df

def validate_sex(df, cols):
    valid_sexes = ['M', 'F', 'U']

    for c in cols:
        df[c] = df[c].apply(lambda x: 'U' if not any(x == y for y in valid_sexes) else x)

    return df

def validate_race(df, cols):
    valid_races = ['BLACK', 'WHITE', 'WHITE HISPANIC', 'BLACK HISPANIC',
                   'ASIAN / PACIFIC ISLANDER', 'UNKNOWN', 'AMERICAN INDIAN/ALASKAN NATIVE']

    for c in cols:
        df[c] = df[c].apply(lambda x: 'UNKNOWN' if not any(x == y for y in valid_races) else x)

    return df
```

```
def validate_dates_and_times(df, date_cols, time_cols = None):
    for c in date_cols:
        df = df.loc[df[c].apply(lambda x: int(x[-4:])) >= 2006]
        df[c] = pd.to_datetime(df[c])

    if time_cols is not None:
        for c in time_cols:
            df[c] = pd.to_datetime(df[c], format = '%H:%M:%S').dt.time

    return df
```

Functions to validate age, sex, and race data,
and to ensure data is in valid range (2006 and later)

Code Snippets Cleaning https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
In [7]: cols = ['KY_CD', 'OFNS_DESC']
        offense_code_df = pd.concat([arrest_df[cols], complaint_df[cols]]).value_counts().reset_index()[cols].sort_values(cols[0])
```

```
In [8]: # Multiple descriptions for different codes
        multi_dict = multiple_descriptions(offense_code_df, cols[0], cols [1])
        multi_dict
```

```
{103: ['HOMICIDE-NEGLIGENT,UNCLASSIFIE', 'HOMICIDE-NEGLIGENT,UNCLASSIFIED'],
 121: ['CRIMINAL MISCHIEF & RELATED OF',
  'CRIMINAL MISCHIEF & RELATED OFFENSES'],
 125: ['NYS LAWS-UNCLASSIFIED FELONY', 'VEHICLE AND TRAFFIC LAWS'],
 233: ['FORCIBLE TOUCHING', 'SEX CRIMES'],
 343: ['OTHER OFFENSES RELATED TO THEFT', 'THEFT OF SERVICES'],
 347: ['INTOXICATED & IMPAIRED DRIVING', 'INTOXICATED/IMPAIRED DRIVING'],
 349: ['DISRUPTION OF A RELIGIOUS SERV', 'DISRUPTION OF A RELIGIOUS SERVICE'],
 351: ['CRIMINAL MISCHIEF & RELATED OF',
  'CRIMINAL MISCHIEF & RELATED OFFENSES'],
 359: ['OFFENSES AGAINST PUBLIC ADMINI',
  'OFFENSES AGAINST PUBLIC ADMINISTRATION'],
 361: ['HARASSMENT', 'OFF. AGNST PUB ORD SENSBLTY &'],
 362: ['OFFENSES AGAINST MARRIAGE UNCL',
  'OFFENSES AGAINST MARRIAGE UNCLASSIFIED'],
 455: ['UNLAWFUL POSS. WEAP. ON SCHOOL',
  'UNLAWFUL POSS. WEAP. ON SCHOOL GROUNDS'],
 571: ['LOITERING/GAMBLING (CARDS, DIC',
  'LOITERING/GAMBLING (CARDS, DICE, ETC)'],
 577: ['UNDER THE INFLUENCE OF DRUGS', 'UNDER THE INFLUENCE, DRUGS'],
 672: ['LOITERING', 'PROSTITUTION & RELATED OFFENSES'],
 677: ['NYS LAWS-UNCLASSIFIED VIOLATION', 'OTHER STATE LAWS']}
```

```
In [9]: # Replace differing descriptions with normalized version, then drop duplicates
        # Each row is now unique
        index_map = [1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0]
        replace_description(offense_code_df, multi_dict, index_map, cols[0], cols[1])
        offense_code_df = offense_code_df.drop_duplicates()
        offense_code_df
```

	KY_CD	OFNS_DESC
46	101	MURDER & NON-NEGL. MANSLAUGHTER
75	102	HOMICIDE-NEGLIGENT-VEHICLE
77	103	HOMICIDE-NEGLIGENT,UNCLASSIFIED
40	104	RAPE
7	105	ROBBERY
...	...	...
78	685	ADMINISTRATIVE CODES
54	880	MOVING INFRACTIONS
22	881	OTHER TRAFFIC INFRACTION
79	882	PARKING OFFENSES
90	995	F.C.A. P.I.N.O.S.

79 rows × 2 columns

Dealing with multiple descriptions for the same offense codes (most were similar, just used different formatting based on the user)

Code Snippets Cleaning https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

Normalizing Borough Names

```
In [18]: arrest_df.ARREST_BORO.unique()

['B', 'K', 'S', 'Q', 'M']
Categories (5, object): ['B', 'K', 'M', 'Q', 'S']
```

```
In [20]: complaint_df.BORO_NM.unique()

['MANHATTAN', 'BROOKLYN', 'BRONX', 'QUEENS', 'STATEN ISLAND']
Categories (5, object): ['BRONX', 'BROOKLYN', 'MANHATTAN', 'QUEENS', 'STATEN ISLAND']
```

```
In [21]: # Need to replace long-form names with codes and rename column
complaint_df.BORO_NM = complaint_df.BORO_NM.cat.rename_categories(
    {'BRONX': 'B', 'BROOKLYN': 'K', 'MANHATTAN': 'M', 'QUEENS': 'Q', 'STATEN ISLAND': 'S'}
)
complaint_df.rename({'BORO_NM': 'BORO'}, axis = 1, inplace = True)
```

```
In [22]: shooting_df.BORO.unique()

['MANHATTAN', 'BRONX', 'QUEENS', 'BROOKLYN', 'STATEN ISLAND']
Categories (5, object): ['BRONX', 'BROOKLYN', 'MANHATTAN', 'QUEENS', 'STATEN ISLAND']
```

```
In [23]: # Need to replace long-form names with codes
shooting_df.BORO = shooting_df.BORO.cat.rename_categories(
    {'BRONX': 'B', 'BROOKLYN': 'K', 'MANHATTAN': 'M', 'QUEENS': 'Q', 'STATEN ISLAND': 'S'}
)
```

Code Snippets Cleaning https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
In [26]: # Standardizing naming conventions
         arrest_df.rename({'AGE_GROUP': 'PERP_AGE_GROUP',
                           'ARREST_PRECINCT': 'PRECINCT_CD',
                           'Latitude': 'LATITUDE',
                           'Longitude': 'LONGITUDE'}, axis = 1, inplace = True)
         complaint_df.rename({'CMPLNT_NUM': 'CMPLNT_KEY',
                              'CMPLNT_FR_DT': 'CMPLNT_DATE',
                              'CMPLNT_FR_TM': 'CMPLNT_TIME',
                              'ADDR_PCT_CD': 'PRECINCT_CD',
                              'Latitude': 'LATITUDE',
                              'Longitude': 'LONGITUDE'}, axis = 1, inplace = True)
         shooting_df.rename({'PRECINCT': 'PRECINCT_CD',
                             'Latitude': 'LATITUDE',
                             'Longitude': 'LONGITUDE'}, axis = 1, inplace = True)
```

Standardizing column naming conventions across data sets.

Code Snippets Tables https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```python
def create_table(cursor, table_name, df, primary_key, foreign_keys = {}):
    command = f'CREATE TABLE IF NOT EXISTS {table_name}('
    for i, c in enumerate(df.columns):
        python_ctype = str(df[c].dtype)

        # SQLite has limited data types
        # It does not have datetime types, and supports everything as text
        if any([python_ctype == 'object',
                python_ctype == 'datetime64[ns]',
                python_ctype == 'category']):
            sql_ctype = 'TEXT'
        # SQLite does not have boolean types, and stores as integer
        elif any([python_ctype == 'int64',
                  python_ctype == 'bool']):
            sql_ctype = 'INTEGER'
        elif python_ctype == 'float64':
            sql_ctype = 'REAL'
        else:
            print(f'Unable to determine SQLite dtype for column {c}. Table was not created.')

            return False

        # If a column name is a number, forcefully turn it into a string
        if c.isnumeric():
            c = f'"{c}"'
```

```python
        command += f'{(c)} {sql_ctype}'

        if c == primary_key:
            command += ' PRIMARY KEY'

        if i != len(df.columns) - 1:
            command += ', '


    if len(foreign_keys) != 0:
        command += ', '

        for j, k in enumerate(foreign_keys.keys()):
            command += f'FOREIGN KEY ({k}) REFERENCES {foreign_keys[k][0]}({foreign_keys[k][1]})'

            if j != len(foreign_keys.keys()) - 1:
                command += ', '

    command += ');'

    print(command)

    cursor = cursor.execute(command)

    print(f'Table {table_name} successfully created.')

    return cursor
```

Function to construct create table SQL statement.

Code Snippets Tables https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```python
# Create internal code descriptions table and populate
res = create_table(cursor = cur, table_name = 'INTERNAL_CODES', df = internal_code_df, primary_key = 'PD_CD')
internal_code_df.to_sql('INTERNAL_CODES', conn, if_exists = 'replace', index = False)
```

```
CREATE TABLE IF NOT EXISTS INTERNAL_CODES(PD_CD INTEGER PRIMARY KEY, PD_DESC TEXT);
Table INTERNAL_CODES successfully created.


475
```

```python
# Create offense code descriptions table and populate
res = create_table(cursor = cur, table_name = 'OFFENSE_CODES', df = offense_code_df, primary_key = 'KY_CD')
offense_code_df.to_sql('OFFENSE_CODES', conn, if_exists = 'replace', index = False)
```

```
CREATE TABLE IF NOT EXISTS OFFENSE_CODES(KY_CD INTEGER PRIMARY KEY, OFNS_DESC TEXT);
Table OFFENSE_CODES successfully created.


79
```

```python
# Create borough table and populate
res = create_table(cursor = cur, table_name = 'BOROUGHS', df = population_df, primary_key = 'BORO_CD')
population_df.to_sql('BOROUGHS', conn, if_exists = 'replace', index = False)
```

```
CREATE TABLE IF NOT EXISTS BOROUGHS(BORO_CD TEXT PRIMARY KEY, BORO_DESC TEXT, "1950" INTEGER, "1960" INTEGER, "1970" INTEGER, "1980" INTEGER, "1990" INTEGER,
"2000" INTEGER, "2010" INTEGER, "2020" INTEGER, "2030" INTEGER, "2040" INTEGER);
Table BOROUGHS successfully created.


6
```

Table Creation Function Calls
(# = Rows Written)

```python
# Create arrest table and populate
res = create_table(cursor = cur, table_name = 'ARRESTS', df = arrest_df,
                   primary_key = 'ARREST_KEY',
                   foreign_keys = {
                       'PD_CD': ('INTERNAL_CODES', 'PD_CD'),
                       'KY_CD': ('OFFENSE_CODES', 'KY_CD')})
arrest_df.to_sql('ARRESTS', conn, if_exists = 'replace', index = False)
```

```
CREATE TABLE IF NOT EXISTS ARRESTS(ARREST_KEY INTEGER PRIMARY KEY, ARREST_DATE TEXT, KY_CD INTEGER, PD_CD INTEGER, LAW_CAT_CD TEXT, LAW_CODE TEXT, PRECINCT_CD
INTEGER, JURISDICTION_CODE INTEGER, BORO TEXT, PERP_AGE_GROUP TEXT, PERP_SEX TEXT, PERP_RACE TEXT, LATITUDE REAL, LONGITUDE REAL, FOREIGN KEY (PD_CD) REFERENCE
S INTERNAL_CODES(PD_CD), FOREIGN KEY (KY_CD) REFERENCES OFFENSE_CODES(KY_CD));
Table ARRESTS successfully created.


5684630
```

```python
# Create complaints table and populate
res = create_table(cursor = cur, table_name = 'COMPLAINTS', df = complaint_df,
                   primary_key = 'CMPLNT_KEY',
                   foreign_keys = {
                       'PD_CD': ('INTERNAL_CODES', 'PD_CD'),
                       'KY_CD': ('OFFENSE_CODES', 'KY_CD')})
complaint_df.to_sql('COMPLAINTS', conn, if_exists = 'replace', index = False)
```

```
CREATE TABLE IF NOT EXISTS COMPLAINTS(CMPLNT_KEY INTEGER PRIMARY KEY, CMPLNT_DATE TEXT, CMPLNT_TIME TEXT, RPT_DT TEXT, KY_CD INTEGER, PD_CD INTEGER, LAW_CAT_CD
TEXT, PRECINCT_CD INTEGER, JURISDICTION_CODE INTEGER, JURIS_DESC TEXT, BORO TEXT, SUSP_AGE_GROUP TEXT, SUSP_RACE TEXT, SUSP_SEX TEXT, VIC_AGE_GROUP TEXT, VIC_R
ACE TEXT, VIC_SEX TEXT, LATITUDE REAL, LONGITUDE REAL, FOREIGN KEY (PD_CD) REFERENCES INTERNAL_CODES(PD_CD), FOREIGN KEY (KY_CD) REFERENCES OFFENSE_CODES(KY_C
D));
Table COMPLAINTS successfully created.


8859411
```

```python
# Create shootings table and populate
res = create_table(cursor = cur, table_name = 'SHOOTINGS', df = shooting_df, primary_key = 'INCIDENT_KEY')
shooting_df.to_sql('SHOOTINGS', conn, if_exists = 'replace', index = False)
```

```
CREATE TABLE IF NOT EXISTS SHOOTINGS(INCIDENT_KEY INTEGER PRIMARY KEY, OCCUR_DATE TEXT, OCCUR_TIME TEXT, PRECINCT_CD INTEGER, JURISDICTION_CODE REAL, BORO TEX
T, STATISTICAL_MURDER_FLAG INTEGER, PERP_AGE_GROUP TEXT, PERP_SEX TEXT, PERP_RACE TEXT, VIC_AGE_GROUP TEXT, VIC_SEX TEXT, VIC_RACE TEXT, LATITUDE REAL, LONGITU
DE REAL);
Table SHOOTINGS successfully created.


28501
```

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

Top Crimes Per Borough

```
q = '''
SELECT
    borough,
    crime_type,
    crime_count,
    rank
FROM (
    SELECT
        c.BORO AS borough,
        o.OFNS_DESC AS crime_type,
        COUNT(*) AS crime_count,
        ROW_NUMBER() OVER (PARTITION BY c.BORO ORDER BY COUNT(*) DESC) AS rank
    FROM
        COMPLAINTS c
    JOIN
        OFFENSE_CODES o ON c.KY_CD = o.KY_CD
    GROUP BY
        c.BORO, o.OFNS_DESC
) AS ranked_crimes
WHERE rank in (1, 2, 3, 4, 5)
ORDER BY rank, borough;
'''

top5_crimes = execute_chunk_query(q, conn, verbosity = 1)
top5_crimes
```

	borough	crime_type	crime_count	rank
0	B	PETIT LARCENY	264063	1
1	K	PETIT LARCENY	425510	1
2	M	PETIT LARCENY	487041	1
3	Q	PETIT LARCENY	311016	1
4	S	HARRASSMENT 2	79403	1
5	B	HARRASSMENT 2	262470	2
6	K	HARRASSMENT 2	354103	2
7	M	GRAND LARCENY	294527	2
8	Q	HARRASSMENT 2	251340	2
9	S	PETIT LARCENY	63448	2
10	B	ASSAULT 3 & RELATED OFFENSES	234031	3
11	K	ASSAULT 3 & RELATED OFFENSES	283927	3
12	M	HARRASSMENT 2	238009	3
13	Q	CRIMINAL MISCHIEF & RELATED OFFENSES	202066	3
14	S	CRIMINAL MISCHIEF & RELATED OFFENSES	54896	3
15	B	CRIMINAL MISCHIEF & RELATED OFFENSES	183838	4
16	K	CRIMINAL MISCHIEF & RELATED OFFENSES	266871	4
17	M	ASSAULT 3 & RELATED OFFENSES	185233	4
18	Q	ASSAULT 3 & RELATED OFFENSES	191560	4
19	S	ASSAULT 3 & RELATED OFFENSES	39459	4
20	B	DANGEROUS DRUGS	164891	5
21	K	GRAND LARCENY	203596	5
22	M	CRIMINAL MISCHIEF & RELATED OFFENSES	167647	5
23	Q	GRAND LARCENY	150070	5
24	S	HARASSMENT	28457	5

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
q = '''
SELECT
    sq1.*,
    sq2.complaints,
    sq3.shootings
FROM (
    SELECT
        A.BORO AS borough,
        CAST(SUBSTR(A.ARREST_DATE, 1, INSTR(A.ARREST_DATE, '-') - 1) AS INT) AS year,
        COUNT(A.ARREST_KEY) AS arrests
    FROM ARRESTS A
    GROUP BY year, A.BORO
) sq1
JOIN (
    SELECT
        C.BORO as borough,
        CAST(SUBSTR(C.CMPLNT_DATE, 1, INSTR(C.CMPLNT_DATE, '-') - 1) AS INT) AS year,
        COUNT(C.CMPLNT_KEY) AS complaints
    FROM COMPLAINTS C
    GROUP BY year, C.BORO
) sq2 ON sq1.borough = sq2.borough AND sq1.year = sq2.year
JOIN (
    SELECT
        S.BORO as borough,
        CAST(SUBSTR(S.OCCUR_DATE, 1, INSTR(S.OCCUR_DATE, '-') - 1) AS INT) AS year,
        COUNT(S.INCIDENT_KEY) AS shootings
    FROM SHOOTINGS S
    GROUP BY year, S.BORO
) sq3 on sq1.borough = sq3.borough AND sq1.year = sq3.year
ORDER BY sq1.year ASC;
'''
```

	borough	year	arrests	complaints	shootings
0	B	2006	83642	110243	568
1	K	2006	102820	157146	850
2	M	2006	101957	126680	288
3	Q	2006	69969	104756	296
4	S	2006	12365	26981	53
...	...	...	...	...	...
85	B	2023	52905	117546	426
86	K	2023	62178	151040	402
87	M	2023	52004	129634	178
88	Q	2023	47553	119537	171
89	S	2023	10004	24227	24

90 rows × 5 columns

Extracting data grouped by borough and year.

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
# Query to extract shootings where a murder occurred by borough
q = '''
SELECT BORO AS borough, COUNT(DISTINCT INCIDENT_KEY) AS murder_count
FROM SHOOTINGS
WHERE STATISTICAL_MURDER_FLAG = TRUE
GROUP BY BORO;
'''

murder_shootings_data = execute_chunk_query(q, conn, verbosity = 1)
murder_shootings_data
```

	borough	murder_count
0	B	1184
1	K	1797
2	M	510
3	Q	665
4	S	137

Extracting murders by shootings from the data set.

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
q = '''
SELECT
    COUNT(CMPLNT_KEY) AS complaints,
    SUSP_AGE_GROUP AS susp_age,
    VIC_AGE_GROUP as vic_age
FROM COMPLAINTS
WHERE SUSP_AGE_GROUP <> "UNKNOWN" AND VIC_AGE_GROUP <> "UNKNOWN"
GROUP BY SUSP_AGE_GROUP, VIC_AGE_GROUP
ORDER BY SUSP_AGE_GROUP DESC;
'''
```

	complaints	susp_age	vic_age
0	18962	<18	18-24
1	42135	<18	25-44
2	24045	<18	45-64
3	3610	<18	65+
4	81067	<18	<18
5	2325	65+	18-24
6	13603	65+	25-44
7	16935	65+	45-64
8	8791	65+	65+
9	1506	65+	<18
10	30080	45-64	18-24
11	171806	45-64	25-44
12	170215	45-64	45-64
13	27233	45-64	65+
14	16708	45-64	<18
15	164507	25-44	18-24
16	738637	25-44	25-44
17	223429	25-44	45-64
18	39233	25-44	65+
19	51897	25-44	<18
20	165258	18-24	18-24
21	156133	18-24	25-44
22	72275	18-24	45-64
23	11430	18-24	65+
24	42838	18-24	<18

Extracting age-related complaints data

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
q = '''
SELECT
    COUNT(CMPLNT_KEY) AS complaints,
    SUSP_SEX AS susp_sex,
    VIC_SEX as vic_sex
FROM COMPLAINTS
WHERE SUSP_SEX <> "U" AND VIC_SEX <> "U"
GROUP BY SUSP_SEX, VIC_SEX
ORDER BY SUSP_SEX DESC;
'''
```

	complaints	susp_sex	vic_sex
0	1416997	M	F
1	992479	M	M
2	538112	F	F
3	290250	F	M

Extracting sex-related complaints data

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
q = '''
SELECT
    COUNT(CMPLNT_KEY) AS complaints,
    SUSP_RACE AS susp_race,
    VIC_RACE AS vic_race
FROM COMPLAINTS
WHERE SUSP_RACE <> "UNKNOWN" AND VIC_RACE <> "UNKNOWN"
GROUP BY SUSP_RACE, VIC_RACE
ORDER BY SUSP_RACE DESC;
'''
```

Extracting race-related complaints data

	complaints	susp_race	vic_race
0	2249	WHITE HISPANIC	AMERICAN INDIAN/ALASKAN NATIVE
1	30232	WHITE HISPANIC	ASIAN / PACIFIC ISLANDER
2	84109	WHITE HISPANIC	BLACK
3	42378	WHITE HISPANIC	BLACK HISPANIC
4	82877	WHITE HISPANIC	WHITE
5	407770	WHITE HISPANIC	WHITE HISPANIC
6	2023	WHITE	AMERICAN INDIAN/ALASKAN NATIVE
7	26052	WHITE	ASIAN / PACIFIC ISLANDER
8	38054	WHITE	BLACK
9	7006	WHITE	BLACK HISPANIC
10	274944	WHITE	WHITE
11	51397	WHITE	WHITE HISPANIC
12	673	BLACK HISPANIC	AMERICAN INDIAN/ALASKAN NATIVE
13	7578	BLACK HISPANIC	ASIAN / PACIFIC ISLANDER
14	37287	BLACK HISPANIC	BLACK
15	56222	BLACK HISPANIC	BLACK HISPANIC
16	19687	BLACK HISPANIC	WHITE
17	75570	BLACK HISPANIC	WHITE HISPANIC
18	6379	BLACK	AMERICAN INDIAN/ALASKAN NATIVE
19	72679	BLACK	ASIAN / PACIFIC ISLANDER
20	952564	BLACK	BLACK
21	53282	BLACK	BLACK HISPANIC
22	146210	BLACK	WHITE
23	189512	BLACK	WHITE HISPANIC
24	984	ASIAN / PACIFIC ISLANDER	AMERICAN INDIAN/ALASKAN NATIVE
25	91102	ASIAN / PACIFIC ISLANDER	ASIAN / PACIFIC ISLANDER
26	11625	ASIAN / PACIFIC ISLANDER	BLACK
27	1887	ASIAN / PACIFIC ISLANDER	BLACK HISPANIC
28	18021	ASIAN / PACIFIC ISLANDER	WHITE
29	12381	ASIAN / PACIFIC ISLANDER	WHITE HISPANIC
30	4464	AMERICAN INDIAN/ALASKAN NATIVE	AMERICAN INDIAN/ALASKAN NATIVE
31	1620	AMERICAN INDIAN/ALASKAN NATIVE	ASIAN / PACIFIC ISLANDER
32	1800	AMERICAN INDIAN/ALASKAN NATIVE	BLACK
33	294	AMERICAN INDIAN/ALASKAN NATIVE	BLACK HISPANIC
34	2208	AMERICAN INDIAN/ALASKAN NATIVE	WHITE
35	1568	AMERICAN INDIAN/ALASKAN NATIVE	WHITE HISPANIC

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
q = '''
SELECT
    BORO AS borough,
    LAW_CAT_CD AS crime_severity,
    COUNT(ARREST_KEY) AS arrests
FROM ARRESTS
GROUP BY BORO, LAW_CAT_CD
ORDER BY BORO DESC;
'''
```

	borough	crime_severity	arrests
0	S	F	65931
1	S	I	554
2	S	M	138936
3	S	V	1105
4	Q	F	332389
5	Q	I	12518
6	Q	M	676513
7	Q	V	59893
8	M	F	390347
9	M	I	4391
10	M	M	1012592
11	M	V	108750
12	K	F	502965
13	K	I	7762
14	K	M	966128
15	K	V	104571
16	B	F	364029
17	B	I	1654
18	B	M	912608
19	B	V	20994

Extracting crime severity data by borough

Code Snippets Queries https://github.com/JoshuaGottlieb/Scalable-Databases-Midterm

```
q = '''
SELECT
    sq1.*,
    sq2.complaints,
    sq3.shootings
FROM (
    SELECT
        A.PRECINCT_CD AS precinct,
        CAST(SUBSTR(A.ARREST_DATE, 1, INSTR(A.ARREST_DATE, '-') - 1) AS INT) AS year,
        COUNT(A.ARREST_KEY) AS arrests
    FROM ARRESTS A
    GROUP BY year, A.PRECINCT_CD
) sq1
JOIN (
    SELECT
        C.PRECINCT_CD as precinct,
        CAST(SUBSTR(C.CMPLNT_DATE, 1, INSTR(C.CMPLNT_DATE, '-') - 1) AS INT) AS year,
        COUNT(C.CMPLNT_KEY) AS complaints
    FROM COMPLAINTS C
    GROUP BY year, C.PRECINCT_CD
) sq2 ON sq1.precinct = sq2.precinct AND sq1.year = sq2.year
JOIN (
    SELECT
        S.PRECINCT_CD as precinct,
        CAST(SUBSTR(S.OCCUR_DATE, 1, INSTR(S.OCCUR_DATE, '-') - 1) AS INT) AS year,
        COUNT(S.INCIDENT_KEY) AS shootings
    FROM SHOOTINGS S
    GROUP BY year, S.PRECINCT_CD
) sq3 on sq1.precinct = sq3.precinct AND sq1.year = sq3.year
ORDER BY sq1.year ASC;
'''
```

	precinct	year	arrests	complaints	shootings
0	5	2006	5027	4150	3
1	6	2006	3860	5582	1
2	7	2006	3085	3820	8
3	9	2006	4145	6360	3
4	10	2006	3608	4952	15
...	...	...	...	...	...
1269	115	2023	3584	9484	6
1270	120	2023	4902	9196	19
1271	121	2023	2510	6724	1
1272	122	2023	1573	5415	2
1273	123	2023	1019	2890	2

1274 rows × 5 columns

Extracting data grouped by precinct and year.