IST 652 Project

An attempt by:

Aishwarya Dhanda
Devesh Singh
Karan Bharadwaj
Thejaswini Ram

NYPD Arrest Data Analysis - 2021 and a comparison with its LAPD counterpart data

Chosen Dataset

The chosen dataset represents NYPD crime and arrest data for the year of 2021. It basically contains a breakdown of every arrest effected in NYC by the NYPD for that particular year.

Each record represents an arrest effected in NYC by the NYPD and includes information about the type of crime, the location, the time of enforcement, perpetrator's sex, age, and race, level of offense etc.

This data has been published by The New York Police Department (NYPD) and we obtained it from 'NYC OpenData' website (this is a website which contains free public data published by NYC agencies).

The exact link is given below

https://data.cityofnewyork.us/Public-Safety/NYPD-Arrest-Data-Year-to-Date-/uip8-fykc/data (https://data.cityofnewyork.us/Public-Safety/NYPD-Arrest-Data-Year-to-Date-/uip8-fykc/data)

Description of the Dataset

This dataset contains 156,000 rows and 19 columns. An important part of understanding the dataset includes getting familiarized with the various columns of data which has been explained below.

Data Dictionary - Column Information	
Column Name	**Column Description**
ARREST_KEY	Randomly generated persistent ID for each arrest
ARREST_DATE	Exact date of arrest for the reported event
PD_CD	Three-digit internal classification code (more granular than Key Code)
PD_DESC	Description of internal classification corresponding with PD code (more granular than Offense Description)
KY_CD	Three digit internal classification code (more general category than PD code)
OFNS_DESC	Description of internal classification corresponding with KY code (more general category than PD description)
LAW_CODE	Law code charges corresponding to the NYS Penal Law, VTL and other various local laws
LAW_CAT_CD	Level of offense: felony, misdemeanor, violation

ARREST_BORO	Borough of arrest. B(Bronx), S(Staten Island), K(Brooklyn), M(Manhattan), Q(Queens)
ARREST_PRECINCT	Precinct where the arrest occurred
JURISDICTION_CODE	Jurisdiction responsible for arrest. Jurisdiction codes 0(Patrol), 1(Transit) and 2(Housing) represent NYPD whilst codes 3 and more represent non NYPD jurisdictions
AGE_GROUP	Perpetrator's age within a category
PERP_SEX	Perpetrator's sex description
PERP_RACE	Perpetrator's race description
X_COORD_CD	Midblock X-coordinate for New York State Plane Coordinate System, Long Island Zone, NAD 83, units feet (FIPS 3104)
Y_COORD_CD	Midblock Y-coordinate for New York State Plane Coordinate System, Long Island Zone, NAD 83, units feet (FIPS 3104)
Latitude	Latitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)
Longitude	Longitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)

Importing packages

In [11]:

```
%matplotlib inline

import pandas as pd
import numpy as np
import requests
from io import StringIO
from io import BytesIO
from zipfile import ZipFile
#Add additional libraries below this line
```

In [12]:
```
#Defining the url for the dataset
urlds="https://gitlab.gitlab.svc.cent-su.org/ccaicedo/652public/-/raw/maste

#Access to datasets via URLs is usually easy (see command below) but we hav
csvdata=requests.get(urlds,verify=False).content  #this will generate a war

zf = ZipFile(BytesIO(csvdata),'r')  #The dataset is being accessed from a z
#It might take a while for all of the data to be accessed. Be patient.
```

```
/opt/conda/lib/python3.9/site-packages/urllib3/connectionpool.py:1013: In
secureRequestWarning: Unverified HTTPS request is being made to host 'git
lab.gitlab.svc.cent-su.org'. Adding certificate verification is strongly
advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.htm
l#ssl-warnings (https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.h
tml#ssl-warnings)
  warnings.warn(
```

In [13]:
```
#Opening the dataset file and reading it into a data frame called "data"
data=pd.read_csv(zf.open("Team9_NYPD_Arrest_Data__Year_to_Date_.csv"))
```

Initial Data Exploration

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 155507 entries, 0 to 155506
Data columns (total 19 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ARREST_KEY                155507 non-null  int64  
 1   ARREST_DATE               155507 non-null  object 
 2   PD_CD                     155478 non-null  float64
 3   PD_DESC                   155404 non-null  object 
 4   KY_CD                     155404 non-null  float64
 5   OFNS_DESC                 155404 non-null  object 
 6   LAW_CODE                  155507 non-null  object 
 7   LAW_CAT_CD                154114 non-null  object 
 8   ARREST_BORO               155507 non-null  object 
 9   ARREST_PRECINCT           155507 non-null  int64  
 10  JURISDICTION_CODE         155507 non-null  int64  
 11  AGE_GROUP                 155507 non-null  object 
 12  PERP_SEX                  155507 non-null  object 
 13  PERP_RACE                 155507 non-null  object 
 14  X_COORD_CD                155507 non-null  int64  
 15  Y_COORD_CD                155507 non-null  int64  
 16  Latitude                  155507 non-null  float64
 17  Longitude                 155507 non-null  float64
 18  New Georeferenced Column  155507 non-null  object 
dtypes: float64(4), int64(5), object(10)
memory usage: 22.5+ MB
```

In [15]:
```
data.shape #checking the column and row count
```

Out[15]: (155507, 19)

In [16]:
```
data.describe()   #summary statistics
```

Out[16]:

	ARREST_KEY	PD_CD	KY_CD	ARREST_PRECINCT	JURISDICTION_CODE	X_C
count	1.555070e+05	155478.000000	155404.000000	155507.000000	155507.000000	1.5
mean	2.304676e+08	407.828066	244.962974	62.850322	0.912486	1.0
std	4.628028e+06	275.739138	150.334545	35.258605	7.894204	2.1
min	2.224711e+08	0.000000	101.000000	1.000000	0.000000	9.1
25%	2.263289e+08	113.000000	111.000000	34.000000	0.000000	9.9
50%	2.306202e+08	339.000000	235.000000	62.000000	0.000000	1.0
75%	2.344524e+08	705.000000	344.000000	101.000000	0.000000	1.0
max	2.385139e+08	997.000000	995.000000	123.000000	97.000000	1.0

In [17]:
```
data.head(5) # Viewing the data
```

Out[17]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_
0	238013474	12/18/2021	157.0	RAPE 1	104.0	RAPE	PL 1303501	
1	236943583	11/25/2021	263.0	ARSON 2,3,4	114.0	ARSON	PL 1501500	
2	234938876	10/14/2021	594.0	OBSCENITY 1	116.0	SEX CRIMES	PL 2631100	
3	234788259	10/11/2021	263.0	ARSON 2,3,4	114.0	ARSON	PL 1501001	
4	234188790	09/28/2021	578.0	NaN	NaN	NaN	PL 2223001	

In [18]:
```
data_dev=data # making a copy
```

Data cleaning for the NYPD dataset

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

Since the process of data cleaning is crucial in obtaining sound and accurate analysis, we have performed the below cleaning activities aimed at achieving the best results for our analysis.

In [19]:
```
data_dev['ARREST_DATE']=pd.to_datetime(data_dev['ARREST_DATE'])
```

We have converted the date column ARREST_DATE to datetime to help us in further analysis

In [20]:
```
data_dev.columns= data_dev.columns.str.strip().str.lower()
```

Next, as a part of data cleaning we converted the column names to lower case and removed white spaces if any.

In [21]:
```
data_dev.head(2)
```

Out[21]:

	arrest_key	arrest_date	pd_cd	pd_desc	ky_cd	ofns_desc	law_code	law_cat_cd	arrest_boro
0	238013474	2021-12-18	157.0	RAPE 1	104.0	RAPE	PL 1303501	F	Q
1	236943583	2021-11-25	263.0	ARSON 2,3,4	114.0	ARSON	PL 1501500	F	K

In [22]:
```
#Renaming some columns for ease of operations in future
data_dev.rename(columns={'x_coord_cd': 'x_coord', 'y_coord_cd': 'y_coord','
```

The above piece of code helps rename some of our columns into easily readable names that helps in ease of reference as needed further.

In [23]:
```
data_dev['arrest_boro'] = data_dev['arrest_boro'].replace({'Q': 'Queens', '
data_dev['perp_sex'] = data_dev['perp_sex'].replace({'M': 'Male', 'F': 'Fem
data_dev['law_cat_cd'] = data_dev['law_cat_cd'].replace({'F': 'Felony', 'M'
data_dev['jurisdiction_code'] = data_dev['jurisdiction_code'].replace({0: '
```

We further decided to replace some abbreviations with their actual names to help in ease of understanding. The boroughs which were coded as Q,M,S,B and K were replaced with their actual names. The 'law_cat_cd' column which mentions the level of offence was also expanded to show the actual names which are felony, misdemeanour, violation and Traffic Infraction. We also replaced Jurisdiction codes with the actual Jurisdiction details.

In [24]:
```
data_dev = data_dev.set_index('arrest_date') #setting datetime as index
```

The datetime was set as an index as we wanted to analyse the arrests across months and seasons.

By doing so, we could come up with meaningful analysis and visualizations.

In [25]:
```
np.count_nonzero(data_dev.isnull())  #checking for null values
```

Out[25]: 1731

The above command checks for the number of null values which we found to be 1731.

In [26]:
```
data_dev[data_dev.isnull().any(axis=1)] #checking the dataframe rows with n
```

Out[26]:

	arrest_key	pd_cd	pd_desc	ky_cd	ofns_desc	law_code	law_cat_cd	arr
arrest_date								
2021-09-28	234188790	578.0	NaN	NaN	NaN	PL 2223001	Misdemeanor	
2021-09-18	233755503	579.0	NaN	NaN	NaN	PL 2224002	Felony	
2021-09-10	233381184	578.0	NaN	NaN	NaN	PL 2223001	Misdemeanor	
2021-05-29	228849706	NaN	NaN	NaN	NaN	PL 2650022	Misdemeanor	
2021-01-24	223489005	NaN	NaN	NaN	NaN	PL 2650022	Misdemeanor	
...	...	...	...	...	...	...	...	
2021-01-25	223521347	49.0	U.S. CODE UNCLASSIFIED	995.0	FOR OTHER AUTHORITIES	FOA9000049	NaN	
2021-01-11	222919919	49.0	U.S. CODE UNCLASSIFIED	995.0	FOR OTHER AUTHORITIES	FOA9000049	NaN	
2021-02-11	224264917	49.0	U.S. CODE UNCLASSIFIED	995.0	FOR OTHER AUTHORITIES	FOA9000049	NaN	
2021-02-17	224492743	49.0	U.S. CODE UNCLASSIFIED	995.0	FOR OTHER AUTHORITIES	FOA9000049	NaN	
2021-02-18	224526582	49.0	U.S. CODE UNCLASSIFIED	995.0	FOR OTHER AUTHORITIES	FOA9000049	NaN	M

1496 rows × 18 columns

We then further went on to diplay all the rows which contained NaN values.

In [27]:
```
# dropping the unrequired columns
data_dev = data_dev.drop('pd_cd', 1)
data_dev = data_dev.drop('law_code', 1)
data_dev = data_dev.drop('ky_cd', 1)
```

```
/tmp/ipykernel_54/588038671.py:2: FutureWarning: In a future version of p
andas all arguments of DataFrame.drop except for the argument 'labels' wi
ll be keyword-only
  data_dev = data_dev.drop('pd_cd', 1)
/tmp/ipykernel_54/588038671.py:3: FutureWarning: In a future version of p
andas all arguments of DataFrame.drop except for the argument 'labels' wi
ll be keyword-only
  data_dev = data_dev.drop('law_code', 1)
/tmp/ipykernel_54/588038671.py:4: FutureWarning: In a future version of p
andas all arguments of DataFrame.drop except for the argument 'labels' wi
ll be keyword-only
  data_dev = data_dev.drop('ky_cd', 1)
```

We decided to drop the 3 columns mentioned in the above code as we found it to be very legally inclined and not helpful in our line of analysis.

In [28]:
```
data_dev[(data_dev.ofns_desc.isna()) & (data_dev['law_cat_cd'] =='Misdemean
```

Out[28]:

arrest_date	arrest_key	pd_desc	ofns_desc	law_cat_cd	arrest_boro	arrest_precinct	jurisdiction_c
2021-09-28	234188790	NaN	NaN	Misdemeanor	Bronx	44	Pa
2021-09-10	233381184	NaN	NaN	Misdemeanor	Queens	114	Pa
2021-05-29	228849706	NaN	NaN	Misdemeanor	Queens	113	
2021-01-24	223489005	NaN	NaN	Misdemeanor	Bronx	40	Pa
2021-11-22	236791704	NaN	NaN	Misdemeanor	Manhattan	28	Pa

Checking for NA values in ofns_desc where the corresponding law_cat_cd is Misdemeanor.

In [29]:
```
d1=data_dev
d1.loc[(data_dev.ofns_desc.isna()) & (data_dev['law_cat_cd'] =='Misdemeanor
```

We replaced the NA values in ofns_desc with 'Misbehaviour'. We noticed a lot of NA values under ofns_desc and we realised we could not afford to lose any data as it was critical for our further analysis. Hence, we replaced NA values under the mentioned column with 'Misbehaviour' as a sort of a dummy value.

In [30]:
```
d1[(data_dev.ofns_desc.isna()) & (data_dev['law_cat_cd'] =='Felony')]
```

Out[30]:

arrest_date	arrest_key	pd_desc	ofns_desc	law_cat_cd	arrest_boro	arrest_precinct	jurisdiction_coc
2021-09-18	233755503	NaN	NaN	Felony	Queens	106	Patı
2021-11-27	236996404	NaN	NaN	Felony	Queens	113	Patı
2021-12-03	237291769	NaN	NaN	Felony	Queens	115	Patı
2021-12-15	237844150	NaN	NaN	Felony	Brooklyn	77	Patı
2021-12-06	237432502	NaN	NaN	Felony	Staten Island	122	Patı

Next we check for NA values in ofns_desc where the corresponding law_cat_cd is Felony.

In [31]:
```
d1.loc[(data_dev.ofns_desc.isna()) & (data_dev['law_cat_cd'] =='Felony'), '
```

Similar to the previous instance where we use dummy values to retain our data in order to extract maximum insights, we again go about repeating the same process. We now replace it with 'Unclassified crime' since we do not know what kind of Felony is being specified here.

In [32]:
```
d1[(data_dev['law_cat_cd'] =='Felony')].head(6)
```

Out[32]:

arrest_date	arrest_key	pd_desc	ofns_desc	law_cat_cd	arrest_boro	arrest_precinct	jurisdictic
2021-12-18	238013474	RAPE 1	RAPE	Felony	Queens	105	
2021-11-25	236943583	ARSON 2,3,4	ARSON	Felony	Brooklyn	69	
2021-10-14	234938876	OBSCENITY 1	SEX CRIMES	Felony	Brooklyn	61	
2021-10-11	234788259	ARSON 2,3,4	ARSON	Felony	Bronx	42	
2021-09-27	234117071	RAPE 1	RAPE	Felony	Brooklyn	84	
2021-09-18	233755503	NaN	Unclassified Crime	Felony	Queens	106	

In [33]:
```
print(data.shape)
print(data_dev.shape)
```

```
(155507, 19)
(155507, 15)
```

This concludes our attempt at data cleaning and data preparation. Our primary motive was to get rid of junk characters, unnecessary white spaces and effectively handle NA values. In our opinion the approach one takes in treating NA values goes a long way in shaping the course of data analysis. In our case, after closely examining the nature of our dataset, and after careful consideration of our objectives we decided to take an approach wherein we handle NA values without loosing vital and relevant data needed for our line of data analysis.

Data Analysis and Visualization

Our initial exploration with NYPD dataset gave us a few pointers on the kind of questions we could ask this dataset. Further we decided that we would split our analysis into two parts: a. Analysing NYPD data as our primary focus b. Bringing in the arrest data from a city comparable to NYC in scale and magnitude in order to add an element of meaningful comparison (LAPD Arrest Data).

We employ matplotlib, seaborn and folium as our go-to libraries for data visualization.

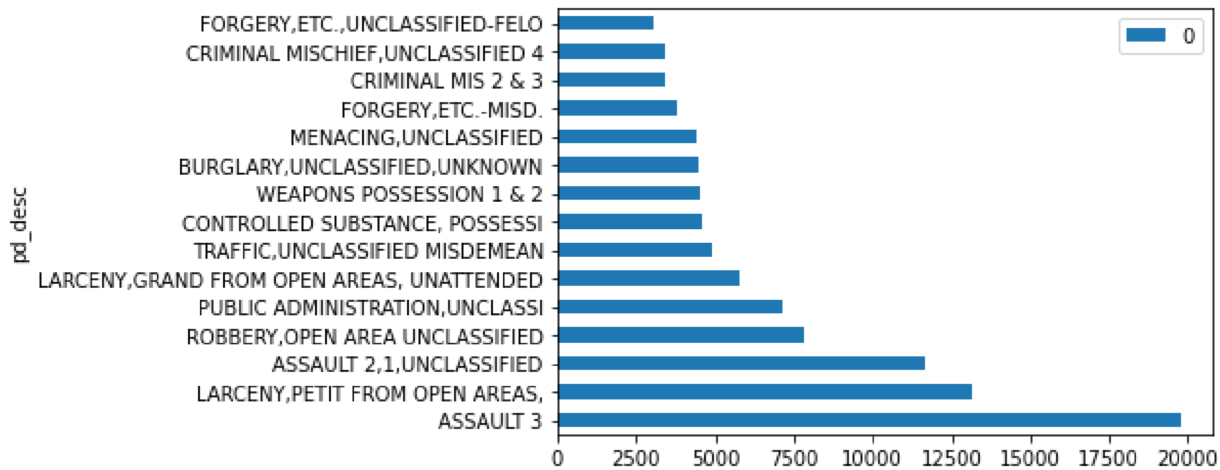
Some of the business questions we would answer are as follows:

1. Which are the most common types of crimes occurring in NYC that led to arrest?
2. How do the distribution of arrests made by NYPD vary monthly for the year of 2021?

3. Which are the most dangerous areas (or boroughs) with highest crime rate within NYC according to the arrests made by NYPD?
4. Which precincts have recorded the most number of arrests in NYC?
5. How do the top 5 crimes in NYC compare to the top 5 crimes in LA which lead to arrest?
6. How do the arrests made in NYC and LA vary according to age and sex?

In [34]:
```
crimetype=pd.DataFrame(data_dev.groupby(['pd_desc']).size())
crimetype.sort_values(0,ascending=False).head(15).plot.barh()
```

Out[34]: <AxesSubplot:ylabel='pd_desc'>



The above plot gives a count of the various types of crimes committed. This is just a small analysis to get things started.

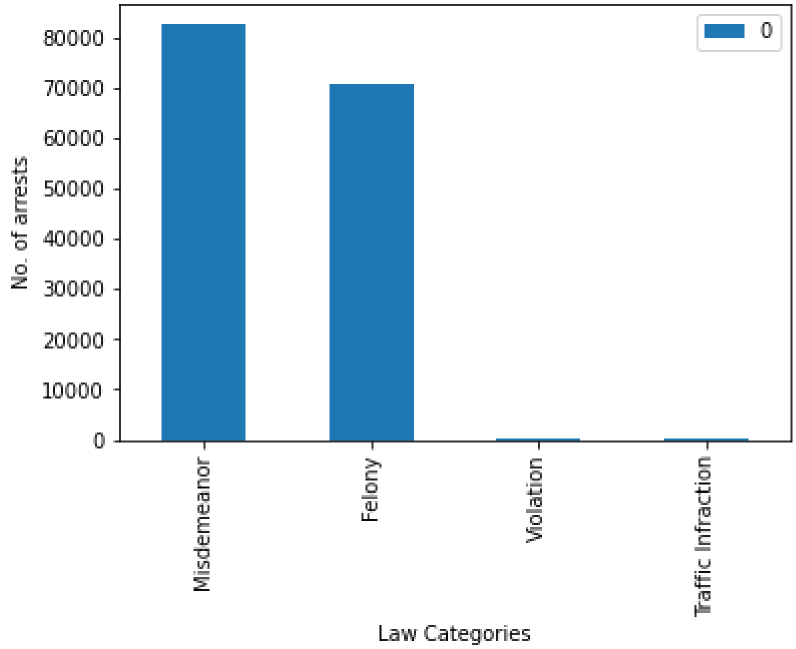
Business Question 1

Which are the most common types of crimes occurring in NYC that led to arrest?

We performed a group by operation on law_cat_cd column and counted the number of different types of offense that led to arrests and used matplotlib library to plot it.

In [24]:
```
law_category=pd.DataFrame(data_dev.groupby(['law_cat_cd']).size())
law_cat_bar=law_category.sort_values(0,ascending=False).head(5).plot.bar()
law_cat_bar.set_xlabel("Law Categories")
law_cat_bar.set_ylabel("No. of arrests")
```

Out[24]:
```
Text(0, 0.5, 'No. of arrests')
```

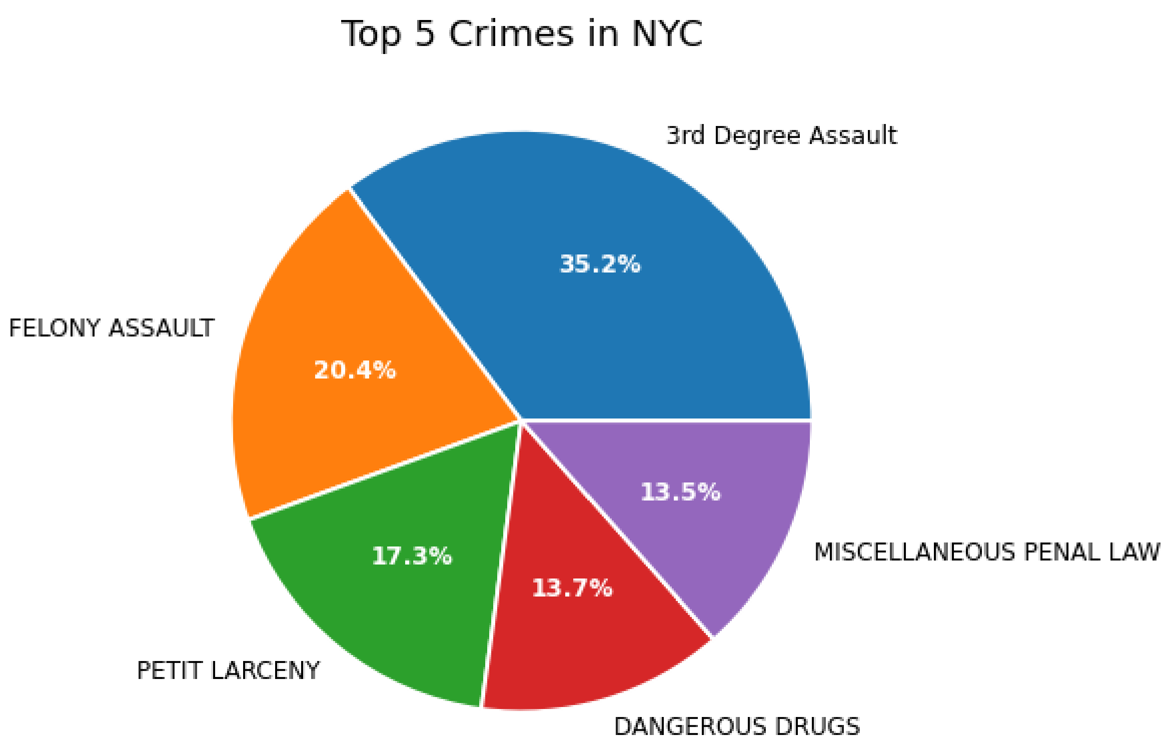


The above bar plot displays the count of the arrests made by the major classification of offense. Misdemeanor is found to be the biggest cause of arrest and Traffic Infraction the least. We owe such a low number for Traffic Infraction to the very popular subway transportation network of New York.

In [38]:
```
import matplotlib.pyplot as plt

x_nyc=data_dev['ofns_desc'].value_counts().head(5)
labels=['3rd Degree Assault','FELONY ASSAULT','PETIT LARCENY','DANGEROUS DR
fig, ax = plt.subplots(figsize=(8, 8))

# Capture each of the return elements.
patches, texts, pcts = ax.pie(
    x_nyc,labels=labels ,autopct='%.1f%%',
    wedgeprops={'linewidth': 2.0, 'edgecolor': 'white'},
    textprops={'size': 'large'})
# Style just the percent values.
plt.setp(pcts, color='white', fontweight='bold')
ax.set_title('Top 5 Crimes in NYC', fontsize=18)
plt.tight_layout()
```



The above visualization shows the proportion of arrests made for more granular details of offense. We infer 3rd degree assault as the most common crime for the year of 2021. This shall throw up interesting finds when we contrast this to its LA counterpart further down the line.

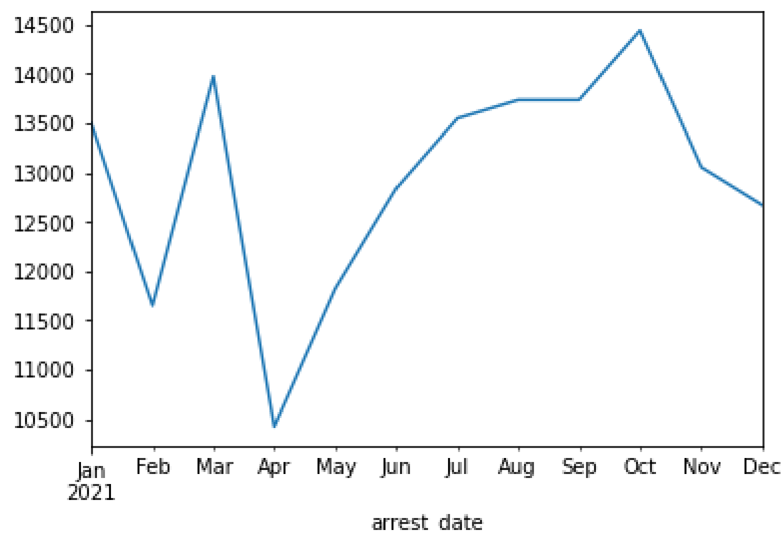
In [39]:
```
data_dev['law_cat_cd'].describe()
```

Out[39]:
```
count          154114
unique              4
top       Misdemeanor
freq            82632
Name: law_cat_cd, dtype: object
```

Business Question 2

How do the distribution of arrests made by NYPD vary monthly for the year of 2021?

In [30]:
```
import matplotlib.pyplot as plt
monthly_crimes = data_dev['pd_desc'].resample('M').count()  #resample, coun
monthly_crimes.sort_index(inplace=True)
time_plot=monthly_crimes.plot().get_figure()
time_plot
time_plot.savefig('test.pdf')
```



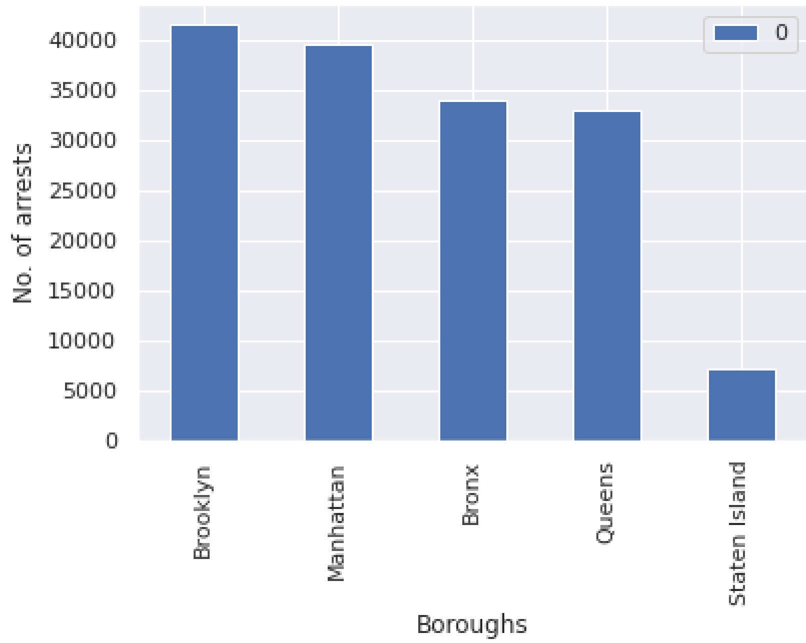
The above visualization displays the distribution of arrests made by NYPD across the 12 months of 2021. The findings reveal April as the month with the lowest number of arrests and October as the month with the highest number of arrests. We attribute the April low to a wave of Covid-19 shutting down a major chunk of commercial activities. We attribute the October high to the full fledged reopening of businesses and the restoration of life akin to the pre-pandemic era.

Business Question 3

Which are the most dangerous areas (or boroughs) with highest crime rate within NYC according to the arrests made by NYPD?

In [44]:
```
law_category=pd.DataFrame(data_dev.groupby(['arrest_boro']).size())
law_cat_bar=law_category.sort_values(0,ascending=False).head(5).plot.bar()
law_cat_bar.set_xlabel("Boroughs")
law_cat_bar.set_ylabel("No. of arrests")
```

Out[44]:
```
Text(0, 0.5, 'No. of arrests')
```



The above bar chart shows the distribution of arrest counts by the boroughs of New York City. Our findings reveal Brooklyn to be the most dangerous borough for the year of 2021 and Staten Island to be the safest borough of 2021 purely by considering the number of arrests made.

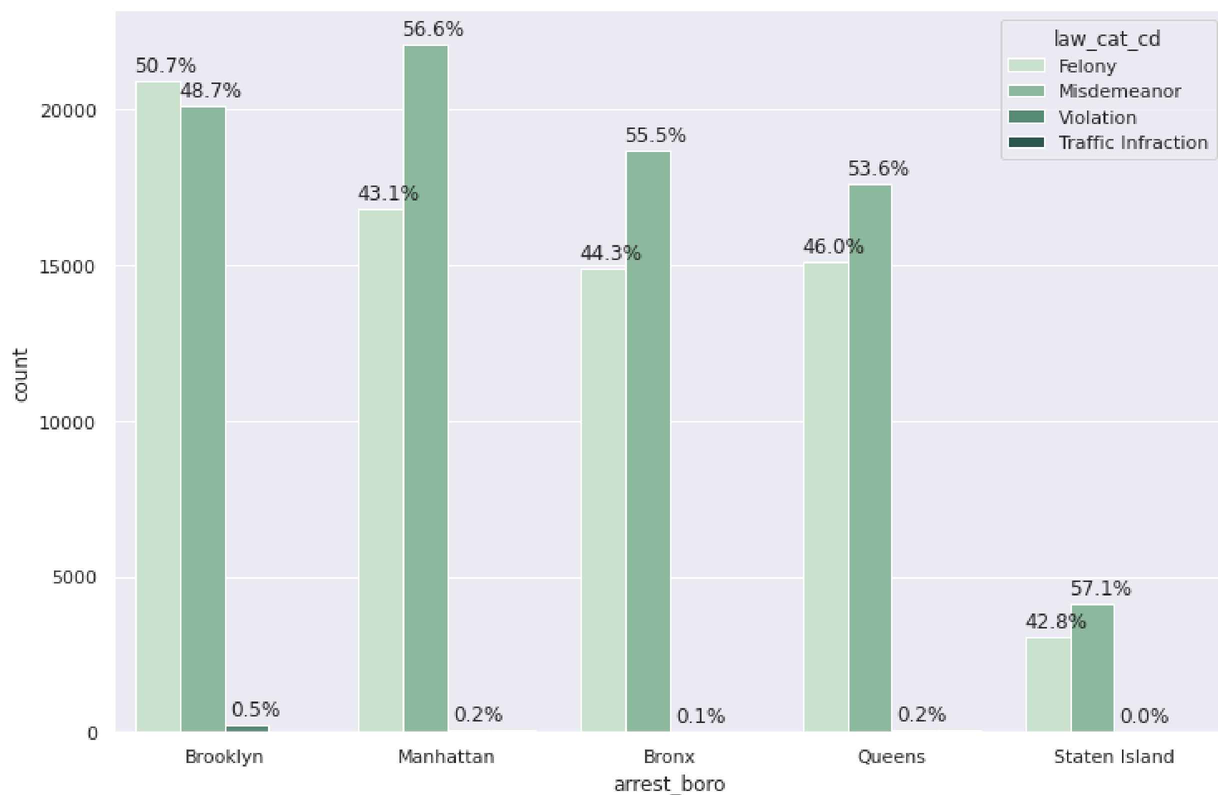
In [42]:
```
import scipy as sp
import seaborn as sns
sns.set(style="darkgrid")
import matplotlib.pyplot as plt
%matplotlib inline
```

In [43]:
```
# tabulate a two way table with variables as boroughs and level of crime
two_way_table = pd.crosstab(index=data["arrest_boro"], columns=data["law_ca
print(two_way_table)

boro_name = ['MANHATTAN', 'BROOKLYN', 'QUEENS', 'BRONX', 'STATEN ISLAND']
crime_level = ['VIOLATION', 'MISDEMEANOR', 'FELONY']
subtotal_boro = data.groupby('arrest_boro')['law_cat_cd'].agg('count').sort
fig = plt.figure(figsize=[12,8])
ax = sns.countplot(x="arrest_boro", hue="law_cat_cd",
                   data=data[['arrest_boro', 'law_cat_cd']],
                   order = subtotal_boro.index,
                  palette = "ch:2.5,-.2,dark=.3")

boro_num2 = [val for val in range(0, 5)]*3 #[0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,
for p, i in zip(ax.patches, boro_num2):
    percent = p.get_height()/subtotal_boro[i]
    ax.annotate('{:.1f}%'. format(percent*100), (p.get_x()+0.138, p.get_hei
```

```
law_cat_cd     Felony  Misdemeanor  Traffic Infraction  Violation     All
arrest_boro
Bronx           14907        18685                  31         40   33663
Brooklyn        20947        20132                  36        223   41338
Manhattan       16822        22097                  88         67   39074
Queens          15111        17610                  69         52   32842
Staten Island    3080         4108                   6          3    7197
All             70867        82632                 230        385  154114
```



In the above visualization, we made an attempt to further shed light upon not only the count of arrests made across boroughs but also the spread of the types of offense contributing to these numbers across these boroughs. An interesting find here was, in Brooklyn, Felony dominated Misdemeanor to emerge as the major offense category whereas we see a reversal of this across all the other boroughs.

Business Question 4

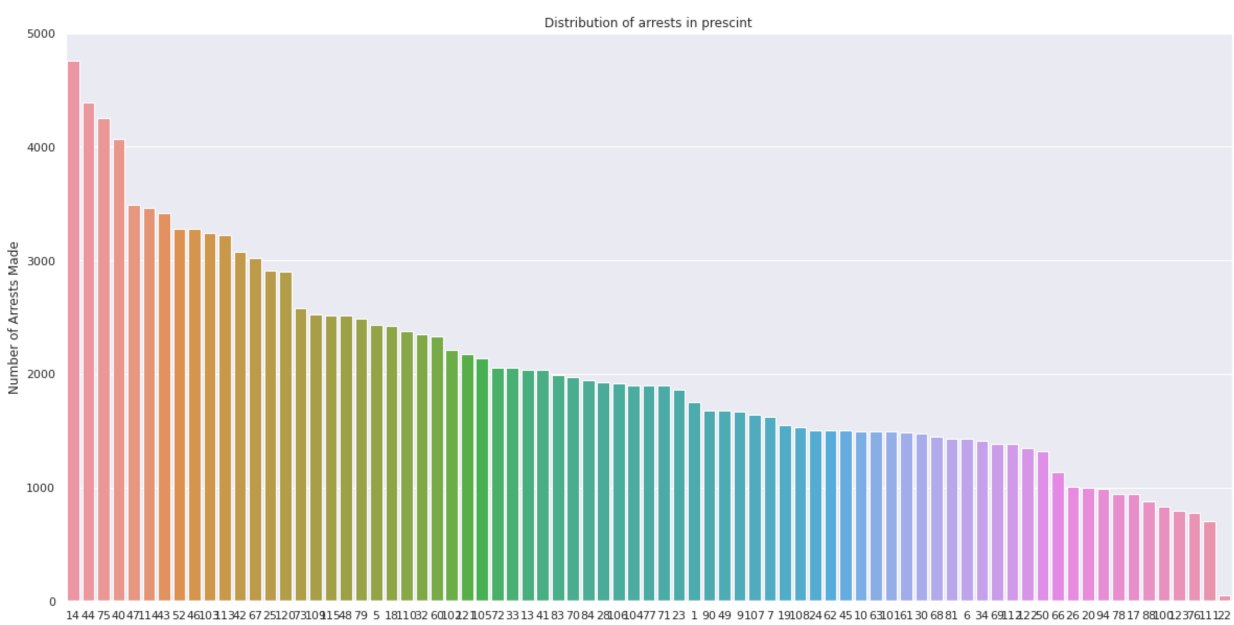
Which precincts have recorded the most number of arrests in NYC?

In [53]:
```
presc_arrest=data.groupby(data['arrest_precinct']).agg({'arrest_key': 'coun
presc_arrest. rename(columns = {'arrest_key':'Total Arrests Made'}, inplace
presc_arrest=presc_arrest.sort_values(by=['Total Arrests Made'], ascending=
presc_arrest=presc_arrest.reset_index()
presc_arrest['arrest_precinct']=presc_arrest['arrest_precinct'].apply(str)


plt.figure(figsize=(20, 10))
plt.title('Distribution of arrests in prescint')

ax = sns.barplot(x="arrest_precinct", y="Total Arrests Made", data=presc_ar

ax.set_ylabel('Number of Arrests Made')
ax.set_xlabel('Prescint Number')
plt.show()
```



The above bar plot shows the distribution of arrests made by the NYPD in 2021 across the various precincts. We infer from the above graph that the precinct 14 has recorded the most number of arrests. On doing the quick look up, we found that precinct 14 corresponds to the midtown south area of Manhattan.Upon further research we realized that this area encompasses Time Square and its surrounding high footfall commercial neighbourhoods.

In [45]:
```
conda install -c conda-forge folium
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.10.3
  latest version: 4.12.0

Please update conda by running

    $ conda update -n base conda



## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - folium


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    branca-0.5.0               |     pyhd8ed1ab_0          26 KB  conda-f
orge
    ca-certificates-2021.10.8  |       ha878542_0         139 KB  conda-f
orge
    certifi-2021.10.8          |   py39hf3d152e_2         145 KB  conda-f
orge
    folium-0.12.1.post1        |     pyhd8ed1ab_1          64 KB  conda-f
orge
    openssl-1.1.1o             |       h166bdaf_0         2.1 MB  conda-f
orge
    ------------------------------------------------------------
                                           Total:         2.5 MB

The following NEW packages will be INSTALLED:

  branca             conda-forge/noarch::branca-0.5.0-pyhd8ed1ab_0
  folium             conda-forge/noarch::folium-0.12.1.post1-pyhd8ed1ab_1

The following packages will be UPDATED:

  ca-certificates                      2021.5.30-ha878542_0 --> 2021.10.8
-ha878542_0
  certifi                          2021.5.30-py39hf3d152e_0 --> 2021.10.8
-py39hf3d152e_2
  openssl                                 1.1.1l-h7f98852_0 --> 1.1.1o-h1
66bdaf_0
```

```
Downloading and Extracting Packages
openssl-1.1.1o       | 2.1 MB    | ####################################
| 100%
ca-certificates-2021 | 139 KB    | ####################################
| 100%
folium-0.12.1.post1  | 64 KB     | ####################################
| 100%
certifi-2021.10.8    | 145 KB    | ####################################
| 100%
branca-0.5.0         | 26 KB     | ####################################
| 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [46]:

```
import folium
from folium import plugins
from folium.plugins import HeatMap
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 12})



data = data[pd.notnull(data['latitude'])]
data = data[pd.notnull(data['longitude'])]
m = folium.Map(location=[40.7221, -73.9198], zoom_start=11)
```

In [47]:

```
# Ensure you're handing it floats
data['latitude'] = data['latitude'].astype(float)
data['longitude'] = data['longitude'].astype(float)

# Filter the DF for rows, then columns, then remove NaNs
#heat_df = data[data['ARREST_DATE']=='2015-04-27'] # Reducing data size so
#heat_df = data[data['OFNS_DESC']=='Homicide'] # Reducing data size so it r
hm_pol = data[data['jurisdiction_code']=='Patrol']
#heat_df = heat_df.dropna(axis=0, subset=['Latitude','Longitude'])

# List comprehension to make out list of lists
heat_data = [[row['latitude'],row['longitude']] for index, row in hm_pol.it

# Plot it on the map
HeatMap(heat_data).add_to(m)

# Display the map
m
```

Out[47]: Make this Notebook Trusted to load map: File -> Trust Notebook

We thought it would be interesting to generate a heat map of the arrests made in NYC so that we could visually better understand the distribution of arrests and also zoom in to the midtown south area corresponding to the precinct 14 as a supplement to the above analysis.

Importing LA dataset

We now get in the LAPD arrest dataset as we need it for our further analysis.

Commands to access second data set start here

In [40]:
```
#Defining the url for the dataset
urlds2="https://gitlab.gitlab.svc.cent-su.org/ccaicedo/652public/-/raw/mast
#Access to datasets via URLs is usually easy (see command below) but we hav
csvdata2=requests.get(urlds2,verify=False).content  #this will generate a w

zf2 = ZipFile(BytesIO(csvdata2),'r')  #The dataset is being accessed from a
#It might take a while for all of the data to be accessed. Be patient.
```

```
/opt/conda/lib/python3.9/site-packages/urllib3/connectionpool.py:1013: In
secureRequestWarning: Unverified HTTPS request is being made to host 'git
lab.gitlab.svc.cent-su.org'. Adding certificate verification is strongly
advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.htm
l#ssl-warnings (https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.h
tml#ssl-warnings)
  warnings.warn(
```

Reading and Cleaning the LA dataset

In [41]:
```
#Opening the dataset file and reading it into a data frame called "data2"
data2=pd.read_csv(zf2.open("Arrest_Data_from_2020_to_Present.csv"))
```

In [42]:
```
data2.head()
```

Out[42]:

	Report ID	Report Type	Arrest Date	Time	Area ID	Area Name	Reporting District	Age	Sex Code	Descent Code	...
0	6115382	BOOKING	01/29/2021 12:00:00 AM	2035.0	1	Central	176	36	M	H	...
1	6303598	BOOKING	01/06/2022 12:00:00 AM	2345.0	6	Hollywood	646	27	M	B	...
2	211218835	RFC	09/01/2021 12:00:00 AM	1230.0	12	77th Street	1207	22	M	H	...
3	211611663	RFC	09/20/2021 12:00:00 AM	1735.0	16	Foothill	1675	24	M	H	...
4	211911576	RFC	07/27/2021 12:00:00 AM	850.0	19	Mission	1961	51	F	H	...

5 rows × 25 columns

In [43]:
```
data_la=data2 #making a copy of the dataset
```

In [44]:
```
data_la.isna().any() #checking for columns with NaN values
```

Out[44]:
```
Report ID                    False
Report Type                  False
Arrest Date                  False
Time                          True
Area ID                      False
Area Name                    False
Reporting District           False
Age                          False
Sex Code                     False
Descent Code                 False
Charge Group Code             True
Charge Group Description      True
Arrest Type Code              True
Charge                       False
Charge Description            True
Disposition Description       True
Address                      False
Cross Street                  True
LAT                          False
LON                          False
Location                     False
Booking Date                  True
Booking Time                  True
Booking Location              True
Booking Location Code         True
dtype: bool
```

In [45]:
```
data_la.isnull().sum()/len(data_la)*100 #checking nan percentages for each
```

Out[45]:
```
Report ID                     0.000000
Report Type                   0.000000
Arrest Date                   0.000000
Time                          0.005681
Area ID                       0.000000
Area Name                     0.000000
Reporting District            0.000000
Age                           0.000000
Sex Code                      0.000000
Descent Code                  0.000000
Charge Group Code             7.440546
Charge Group Description      7.457589
Arrest Type Code              0.000710
Charge                        0.000000
Charge Description            7.440546
Disposition Description       7.564815
Address                       0.000000
Cross Street                 49.388239
LAT                           0.000000
LON                           0.000000
```

Data related to booking are all irrelevant to us so we are dropping these columns, as well as the cross street column.

In [46]:
```
data_la.shape
```
Out[46]:
```
(140823, 25)
```

Checking the dimensions, of the rows with na values

In [47]:
```
print(data_la[(data_la.Time.isna())].shape)
print(data_la[(data_la['Charge Group Code'].isna())].shape)
print(data_la[(data_la['Disposition Description'].isna())].shape)
print(data_la[(data_la['Cross Street'].isna())].shape)
print(data_la[(data_la.Time.isna())].shape)
print(data_la[(data_la.Time.isna())].shape)
```
```
(8, 25)
(10478, 25)
(10653, 25)
(69550, 25)
(8, 25)
(8, 25)
```

In [48]:
```
df2=data_dev #making an extra copy for reference
```

We only need the data from 2021 as we need to contrast it with the NYPD dataset. Hence, we are retaining only this and deleting all other records.

In [49]:
```
data_la['Arrest Date']=pd.to_datetime(data_la['Arrest Date'])
mask=(data_la['Arrest Date'] >= '2021-1-1') & (data_la['Arrest Date'] <= '2
df_la=data_la.loc[mask]
```

In [50]:
```
#la dataset date range
data_la['Arrest Date']=pd.to_datetime(data_la['Arrest Date'])
```

In [51]:
```
data_la = data_la.set_index('Arrest Date') #setting the date time as index
```

In [52]:
```
df_la.shape
```
Out[52]:
```
(66951, 25)
```

In [53]:
```
df_la.head()
```

Out[53]:

	Report ID	Report Type	Arrest Date	Time	Area ID	Area Name	Reporting District	Age	Sex Code	Descent Code	...	D D
0	6115382	BOOKING	2021-01-29	2035.0	1	Central	176	36	M	H	...	CC
2	211218835	RFC	2021-09-01	1230.0	12	77th Street	1207	22	M	H	...	MISDI CC
3	211611663	RFC	2021-09-20	1735.0	16	Foothill	1675	24	M	H	...	MISDI CC
4	211911576	RFC	2021-07-27	850.0	19	Mission	1961	51	F	H	...	MISDI CC
5	6270449	BOOKING	2021-10-30	1300.0	2	Rampart	246	31	M	H	...	CC

5 rows × 25 columns

In [54]:
```
df_la = df_la.drop(columns=['Booking Date','Booking Time','Booking Location
```

In [55]:
```
df_la = df_la.drop(columns=['Charge Group Description', 'Charge Group Code'
```

Renaming some columns

In [56]:
```
#Renaming some columns for ease of operations in future
df_la.rename(columns={'Charge Description': 'Charge_Description', 'Disposit
```

Replacing NA values in the 'Charge Description' columns by 'Misbehaviour' wherever it is 'Misdemeanor' in the Disposition Description

In [57]:
```
df_la[(df_la.Charge_Description.isna()) & (df_la['Disposition_Description']
```

Out[57]:

	Report ID	Report Type	Arrest Date	Time	Area ID	Area_Name	Reporting_District	Age	Sex_Code	D
2	211218835	RFC	2021-09-01	1230.0	12	77th Street	1207	22	M	
3	211611663	RFC	2021-09-20	1735.0	16	Foothill	1675	24	M	
4	211911576	RFC	2021-07-27	850.0	19	Mission	1961	51	F	
7	210216097	RFC	2021-10-03	2210.0	2	Rampart	246	60	M	
10	212115329	RFC	2021-10-17	1210.0	21	Topanga	2125	45	F	

In [58]:
```
df_la.loc[(df_la.Charge_Description.isna()) & (df_la['Disposition_Descripti
```

Replacing NA values for charge description with 'UNKNOWN' for thos entries where Disposition_Description is NA too.

In [59]:
```
df_la.loc[(df_la.Charge_Description.isna()) & (df_la.Disposition_Descriptio
```

We perform the LAPD dataset cleaning in a similar approach to what we employ with NYPD dataset. The major difference was we had to extract the 2021 year related value for the LAPD dataset as opposed to the NYPD dataset which was already defined for 2021.

Replacing the NA values for charge description with 'UNKNOWN' for all the remaining rows with NA values.

In [60]:
```
df_la.loc[(df_la.Charge_Description.isna()),'Charge_Description']='UNKNOWN'
```

In [61]:
```
sum(df_la.Charge_Description.isna()) #charge description column is cleaned.
```

Out[61]: 0

In [62]:
```
df_la.head()
```

Out[62]:

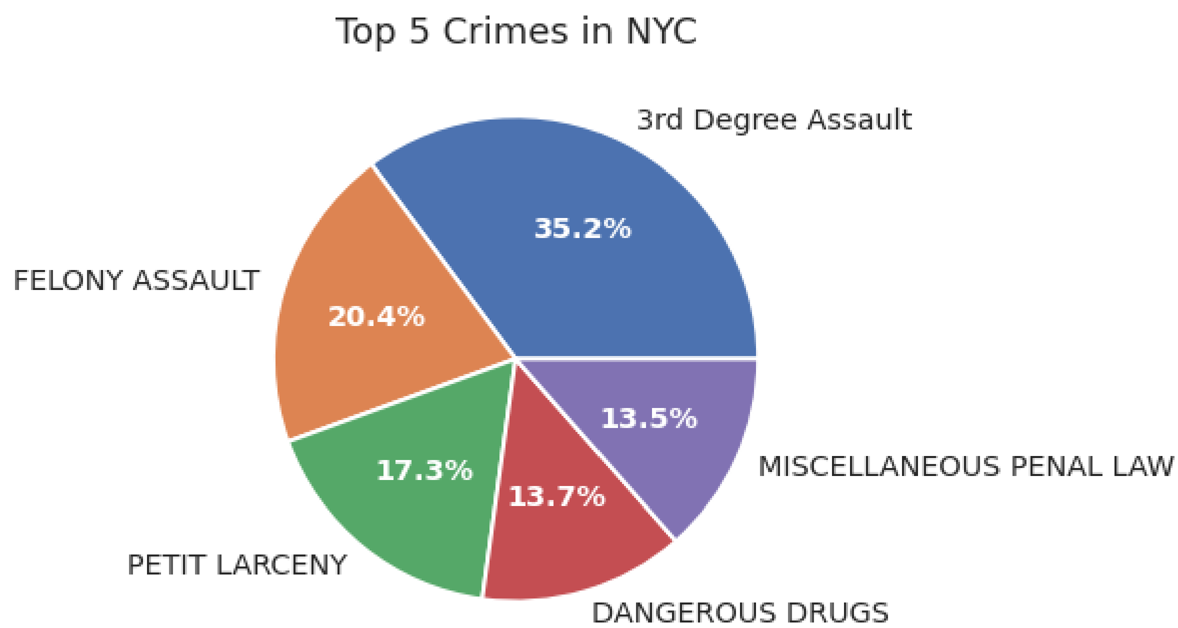
	Report ID	Report Type	Arrest Date	Time	Area ID	Area_Name	Reporting_District	Age	Sex_Code	Des
0	6115382	BOOKING	2021-01-29	2035.0	1	Central	176	36	M	
2	211218835	RFC	2021-09-01	1230.0	12	77th Street	1207	22	M	
3	211611663	RFC	2021-09-20	1735.0	16	Foothill	1675	24	M	
4	211911576	RFC	2021-07-27	850.0	19	Mission	1961	51	F	
5	6270449	BOOKING	2021-10-30	1300.0	2	Rampart	246	31	M	

Business Question 5

How do the top 5 crimes in NYC compare to the top 5 crimes in LA which lead to arrest?

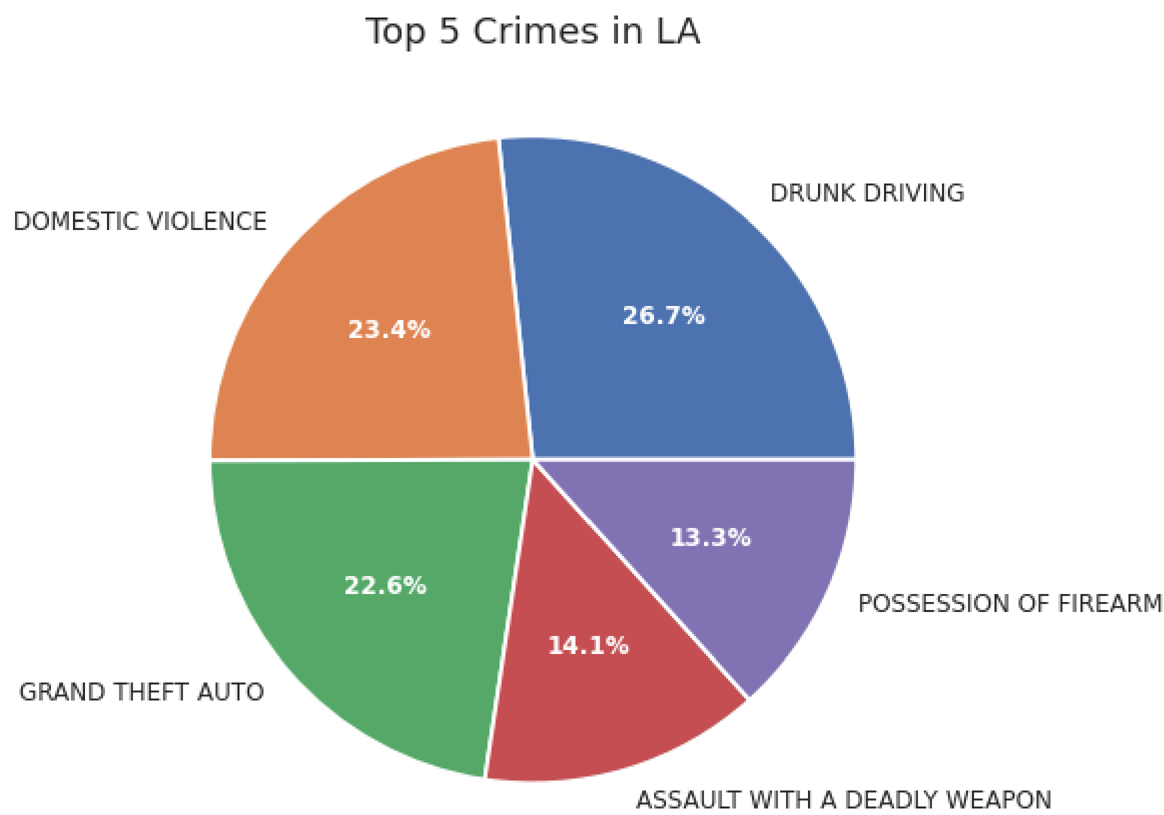
In [55]:
```
x_nyc=data_dev['ofns_desc'].value_counts().head(5)
labels=['3rd Degree Assault','FELONY ASSAULT','PETIT LARCENY','DANGEROUS DR
fig, ax = plt.subplots(figsize=(8, 8))

# Capture each of the return elements.
patches, texts, pcts = ax.pie(
    x_nyc,labels=labels ,autopct='%.1f%%',
    wedgeprops={'linewidth': 2.0, 'edgecolor': 'white'},
    textprops={'size': 'large'})
# Style just the percent values.
plt.setp(pcts, color='white', fontweight='bold')
ax.set_title('Top 5 Crimes in NYC', fontsize=18)
plt.tight_layout()
```



In [63]:
```
x=df_la['Charge_Description'].value_counts().head(5)
labels=['DRUNK DRIVING','DOMESTIC VIOLENCE','GRAND THEFT AUTO','ASSAULT WIT
fig, ax = plt.subplots(figsize=(8, 8))

# Capture each of the return elements.
patches, texts, pcts = ax.pie(
    x,labels=labels, autopct='%.1f%%',
    wedgeprops={'linewidth': 2.0, 'edgecolor': 'white'},
    textprops={'size': 'large'})
# Style just the percent values.
plt.setp(pcts, color='white', fontweight='bold')
ax.set_title('Top 5 Crimes in LA', fontsize=18)
plt.tight_layout()
```



The above two pie charts was plotted with an intension to compare the top 5 crimes in NYC to the top 5 crimes in LA which led to arrest. This comparison threw up surprising yet interesting results. Even though LA and NYC are comparable cities in terms of scale they vastly differ in terms of the nature of crimes committed. The top crime in LA is drunk driving which doesn't even feature in top 5 crimes of NYC. We attribute this to well connected public transportation network of NYC as oppose to the comparably poorer network of LA. Naturally the proportion of ownership of cars in LA is much higher than that in NYC, which obviously has a cascading effect on drunk driving arrests made.

Business Question 6

How do the arrests made in NYC and LA vary according to age and sex?

A. NYPD Data

In [57]:
```
test_df=data_dev #making a new copy to use for pivotting the original dataf
```

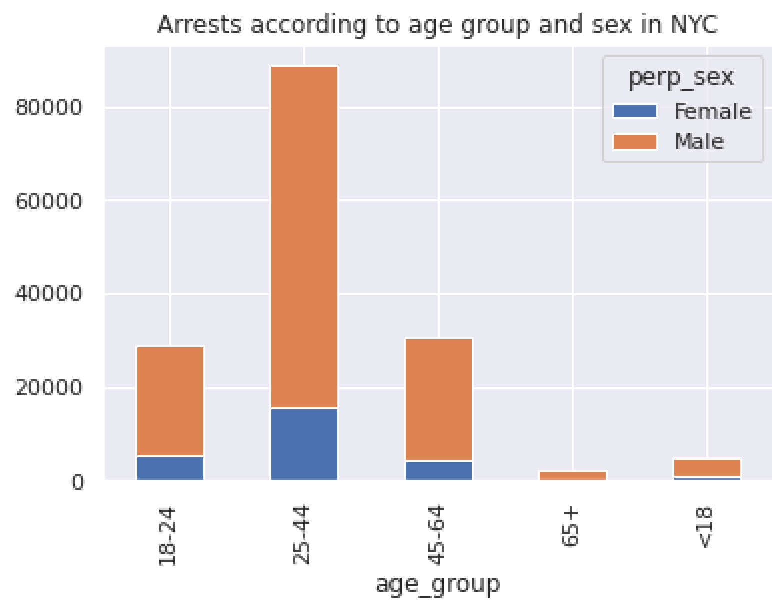
In [58]:
```
test_1_df=test_df.groupby(['age_group','perp_sex']).count() #grouping by ag
```

In [59]:
```
stacked_df=test_1_df.iloc[:,0:1] #subsetting the dataframe above
```

In [60]:
```
pivot_stacked = pd.pivot_table(data=stacked_df, index=['age_group'], column
```

In [61]:
```
pivot_stacked.plot.bar(y='arrest_key',stacked=True,title='Arrests according
```

Out[61]:
```
<AxesSubplot:title={'center':'Arrests according to age group and sex in N
YC'}, xlabel='age_group'>
```



The above visualization is a stacked bar plot that shows the distribution of arrests made by age group and sex in NYC. Males dominate in terms of the arrest count. The age group of 25-44 dominate the arrest count.

B. LAPD Data

In [70]:
```
import seaborn as sns
sns.violinplot(data2['Age'], data2['Sex Code'],invert=False) #Variable Plot
sns.despine()
```

```
/opt/conda/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureW
arning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing othe
r arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```

We experimented with something known as the violin plot to demonstrate the distribution of arrests made by sex and age for the LAPD dataset.

In general, violin plots are a method of plotting numeric data and can be considered a combination of the box plot with a kernel density plot. In the violin plot, we can find the same information as in the box plots:

1. median (a white dot on the violin plot)
2. interquartile range (the black bar in the center of violin)
3. the lower/upper adjacent values (the black lines stretched from the bar) — defined as first quartile — 1.5 IQR and third quartile + 1.5 IQR respectively. These values can be used in a simple outlier detection technique (Tukey's fences) — observations lying outside of these "fences" can be considered outliers.

The unquestionable advantage of the violin plot over the box plot is that aside from showing the abovementioned statistics it also shows the entire distribution of the data. This is of interest, especially when dealing with multimodal data, i.e., a distribution with more than one peak.

One interesting observation in the LAPD dataset was that the median age of female convicts is lower than the median age of male convicts.

A similarity with the NYPD data is the major age group of people arrested remain roughly the same (young adults).

Recommendations and Conclusion

1. Improve patrolling in those precincts which have recorded a large number of crimes such as Precinct 14, 44 and 75.
2. Improve surveillance in Brooklyn which emerged as the most dangerous borough for 2021.
3. Provide better support for financially affected lower income section of society due to COVID-19 which accounts for a spike in crime rate once the pandemic took over.
4. We also observed a major variation in the nature of crimes between LA and NYC. While drunk driving tops the LA charts, third degree assault tops the NYC charts (drunk driving does not even feature in the top 5 of NYC arrests). We suggest ramping up the public transport network in LA.

References

1. https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.html (https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.html)
2. https://seaborn.pydata.org/ (https://seaborn.pydata.org/)
3. https://pandas.pydata.org/ (https://pandas.pydata.org/)
4. https://scipy.org/ (https://scipy.org/)
5. https://python-visualization.github.io/folium/ (https://python-visualization.github.io/folium/)