Full Data Description:-

- Provided by - NYC Open Data
- Agency - Police Department (NYPD)
- Views - 76.8K
- Downloads - 16K
- Rows - 257K
- Columns - 36
- Each row is a Complaint
- Source:- https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Current-Year-To-Date-/5uac-w243

About This Dataset:-

> This Dataset Includes All Valid Felony, Misdemeanour, And Violation Crimes Reported To The New York City Police Department (NYPD) For All Complete Quarters Of The Year (2019)
>
> Background:- In Many Other Cities, Open Data Is A Technical Policy Or An Executive Order. In New York City, It's The Law. On March 7, 2012, Former Mayor Bloomberg Signed Local Law 11 Of 2012, More Commonly Known As The "Open Data Law," Which Amended The New York City Administrative Code To Mandate That All Public Data Be Made Available On A Single Web Portal By The End Of 2018. In November 2015, January 2016, And December 2017, Mayor De Blasio Approved Several Amendments To The Open Data Law. These Laws, Which Include Stronger Requirements On Data Dictionaries And Data Retention, Response Timelines For Public Requests, And An Extension Of The Open Data Mandate Into Perpetuity, Help Make It Easier For New Yorkers To Access City Data Online And Anchor The City's Transparency Initiatives Around Open Data

Update Frequency - Quarterly
Automation - No Date Made Public - 11/1/2018

For Additional Details, Please See The Attached Source:-https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Current-Year-To-Date-/5uac-w243

Columns In Dataset :-

1. CMPLNT_NUM - Randomly generated persistent ID for each complaint
1. ADDR_PCT_CD - The precinct in which the incident occurred
1. BORO_NM - The name of the borough in which the incident occurred
1. CMPLNT_FR_DT - Exact date of occurrence for the reported event (or starting date of occurrence, if CMPLNT_TO_DT exists
1. CMPLNT_FR_TM - Exact time of occurrence for the reported event (or starting time of occurrence, if CMPLNT_TO_TM exists)
1. CMPLNT_TO_DT - Ending date of occurrence for the reported event, if the exact time of occurrence is unknown
1. CMPLNT_TO_TM - Ending time of occurrence for the reported event, if the exact time of occurrence is unknown
1. CRM_ATPT_CPTD_CD - Indicator of whether the crime was successfully completed or attempted, but failed or was interrupted prematurely
1. HADEVELOPT - Name of NYCHA housing development of occurrence, if applicable
1. HOUSING_PSA - Development Level Code
1. JURISDICTION_CODE - Jurisdiction responsible for the incident. Either internal, like Police(0), Transit(1), and Housing(2); or external(3), like Correction, Port Authority, etc.
1. JURIS_DESC - Description of the jurisdiction code
1. KY_CD - Three-digit offence classification code
1. LAW_CAT_CD - Level of offence: felony, misdemeanour, violation
1. LOC_OF_OCCUR_DESC - Specific location of occurrence in or around the premises; inside, opposite of, front of, rear of
1. OFNS_DESC - Description of offence corresponding with key code
1. PARKS_NM - Name of NYC park, playground or greenspace of occurrence, if applicable (state parks are not included)
1. PATROL_BORO - The name of the patrol borough in which the incident occurred
1. PD_CD - Three-digit internal classification code (more granular than Key Code)
1. PD_DESC -Description of internal classification corresponding with PD code (more granular than Offense Description)
1. PREM_TYP_DESC - Specific description of premises; grocery store, residence, street, etc.
1. RPT_DT - Date event was reported to police
1. STATION_NAME - Transit station name
1. SUSP_AGE_GROUP - Suspect's Age Group
1. SUSP_RACE - Suspect's Race Description
1. SUSP_SEX - Suspect's Sex Description
1. TRANSIT_DISTRICT - Transit district in which the offence occurred.
1. VIC_AGE_GROUP - Victim's Age Group
1. VIC_RACE - Victim's Race Description
1. VIC_SEX - Victim's Sex Description

1. X_COORD_CD X - coordinate for New York State Plane Coordinate System, Long Island Zone, NAD 83, units feet (FIPS 3104)

1. Y_COORD_CD Y - coordinate for New York State Plane Coordinate System, Long Island Zone, NAD 83, units feet (FIPS 3104)

1. Latitude - Midblock Latitude coordinates for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)

1. Longitude - Midblock Longitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)

1. Lat_Lon - Latitude and Longitude combined

1. New Georeferenced Column- This is a Geographical point coordinate

Importing Libraries And Loading Data

> *In This Section We Will Load The Necessary Libraries And Load Data:-*

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import squarify
import plotly.express as plx
%matplotlib inline
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")
# Setting Max Rows Display to 1000
pd.set_option('display.max_columns', 1000)
# Setting Max Columns Display to 1000
pd.set_option('display.max_rows', 1000)
```

In [3]:
```
# I Have Found That Null Values Are Disguised In Many Other Names Such As Example [No Clu
# So We Need To Include Them As Well And Drop When Necessary
nan_values = ['NO CLUE', 'N/A', np.nan,"NaT",'("null")',"not available",pd.NaT,"(null)",
              "UNKNOWN","U","E","D"]
df = pd.read_csv(r"C:\Users\abdul\Downloads\NYPD_Complaint_Data_Current__Year_To_Date_.cs
```

In [4]:
```
# Checking If These Two Columns Are Equal Or Not
df['CMPLNT_FR_TM'].equals(df['CMPLNT_TO_TM'])
```

Out[4]:
```
False
```

Cleaning Data

- *Looking Out For Missing/Null Values*
- *Structuring The Data In Proper Format*
- _Droping Duplicate Values__
- *Spelling Correction, Renaming Columns/Rows*

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 130199 entries, 0 to 130198
Data columns (total 36 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   CMPLNT_NUM                130199 non-null  object 
 1   ADDR_PCT_CD               130185 non-null  float64
 2   BORO_NM                   129986 non-null  object 
 3   CMPLNT_FR_DT              130199 non-null  object 
 4   CMPLNT_FR_TM              130199 non-null  object 
 5   CMPLNT_TO_DT              121143 non-null  object 
 6   CMPLNT_TO_TM              121206 non-null  object 
 7   CRM_ATPT_CPTD_CD          130199 non-null  object 
 8   HADEVELOPT                464 non-null     object 
 9   HOUSING_PSA               8781 non-null    float64
 10  JURISDICTION_CODE         130199 non-null  int64  
 11  JURIS_DESC                130199 non-null  object 
 12  KY_CD                     130199 non-null  int64  
 13  LAW_CAT_CD                130199 non-null  object 
 14  LOC_OF_OCCUR_DESC         105318 non-null  object 
 15  OFNS_DESC                 130196 non-null  object 
 16  PARKS_NM                  427 non-null     object 
 17  PATROL_BORO               130199 non-null  object 
 18  PD_CD                     130083 non-null  float64
 19  PD_DESC                   130083 non-null  object 
 20  PREM_TYP_DESC             128509 non-null  object 
 21  RPT_DT                    130199 non-null  object 
 22  STATION_NAME              3159 non-null    object 
 23  SUSP_AGE_GROUP            62706 non-null   object 
 24  SUSP_RACE                 77422 non-null   object 
 25  SUSP_SEX                  83040 non-null   object 
 26  TRANSIT_DISTRICT          3159 non-null    float64
 27  VIC_AGE_GROUP             91044 non-null   object 
 28  VIC_RACE                  88069 non-null   object 
 29  VIC_SEX                   93582 non-null   object 
 30  X_COORD_CD                130198 non-null  float64
 31  Y_COORD_CD                130198 non-null  float64
 32  Latitude                  130198 non-null  float64
 33  Longitude                 130198 non-null  float64
 34  Lat_Lon                   130198 non-null  object 
 35  New Georeferenced Column  130198 non-null  object 
dtypes: float64(8), int64(2), object(26)
memory usage: 35.8+ MB
```

In [6]:
```
df
```

Out[6]:

	CMPLNT_NUM	ADDR_PCT_CD	BORO_NM	CMPLNT_FR_DT	CMPLNT_FR_TM	CMPLNT_TO_DT	C
0	261548672	43.0	BRONX	01/07/2023	21:30:00	NaN	
1	265065361	69.0	BROOKLYN	06/12/2022	01:55:00	NaN	
2	262612530	110.0	QUEENS	09/09/2022	00:55:00	09/09/2022	
3	261233867	123.0	STATEN ISLAND	01/02/2023	02:00:00	01/02/2023	
4	264778583	30.0	MANHATTAN	06/29/2021	21:00:00	06/29/2021	
...	...	...	...	...	...	...	
130194	265938590	83.0	BROOKLYN	03/30/2023	20:55:00	NaN	
130195	265988197	1.0	MANHATTAN	03/31/2023	22:46:00	03/31/2023	
130196	265966526	7.0	MANHATTAN	03/31/2023	03:55:00	03/31/2023	
130197	265958921	48.0	BRONX	03/31/2023	08:10:00	03/31/2023	
130198	265951015	103.0	QUEENS	03/30/2023	13:30:00	03/30/2023	

130199 rows × 36 columns

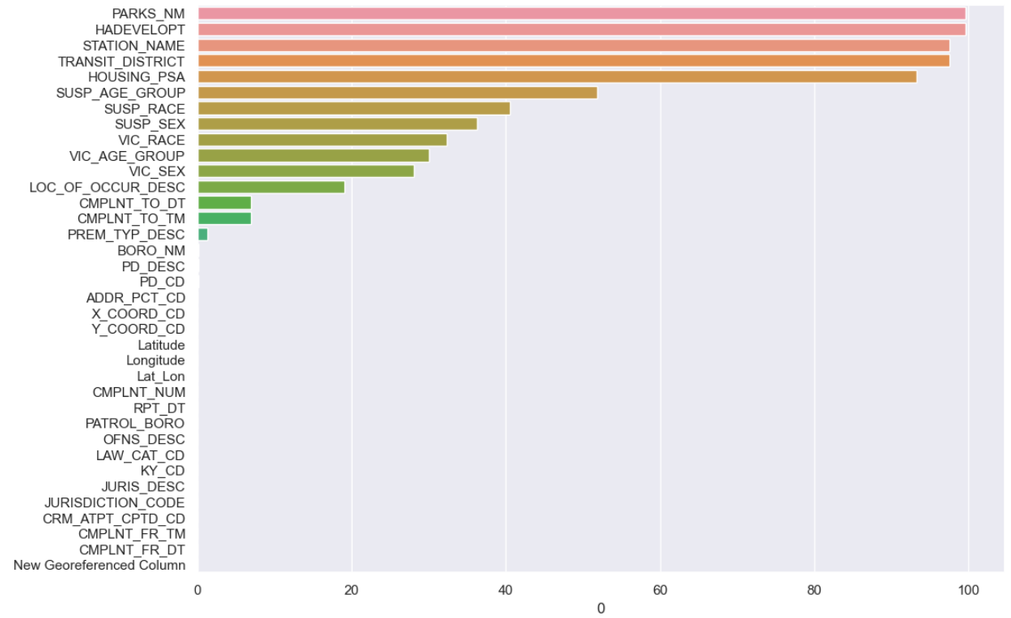
In [7]:
```
# The Code Below Calculates The Percentage Of Null Values From Every Columns
missing_values = df.isnull().mean().round(4).mul(100).sort_values(ascending=False)
missing_values = pd.DataFrame(missing_values)
missing_values
```

Out[7]:

	0
PARKS_NM	99.67
HADEVELOPT	99.64
STATION_NAME	97.57
TRANSIT_DISTRICT	97.57
HOUSING_PSA	93.26
SUSP_AGE_GROUP	51.84
SUSP_RACE	40.54
SUSP_SEX	36.22
VIC_RACE	32.36
VIC_AGE_GROUP	30.07
VIC_SEX	28.12
LOC_OF_OCCUR_DESC	19.11
CMPLNT_TO_DT	6.96
CMPLNT_TO_TM	6.91
PREM_TYP_DESC	1.30
BORO_NM	0.16
PD_DESC	0.09
PD_CD	0.09
ADDR_PCT_CD	0.01
X_COORD_CD	0.00
Y_COORD_CD	0.00
Latitude	0.00
Longitude	0.00
Lat_Lon	0.00
CMPLNT_NUM	0.00
RPT_DT	0.00
PATROL_BORO	0.00
OFNS_DESC	0.00
LAW_CAT_CD	0.00
KY_CD	0.00
JURIS_DESC	0.00
JURISDICTION_CODE	0.00
CRM_ATPT_CPTD_CD	0.00
CMPLNT_FR_TM	0.00
CMPLNT_FR_DT	0.00
New Georeferenced Column	0.00

In [8]:
```
# I Think It Would Be Nice To Plot A Graph Of Null Values Of Every Column, Isn't ?
sns.set(rc={'figure.figsize':(11.7,8.27)})
```

```
sns.barplot(data=missing_values, y=missing_values.index, x=0);
```



In [9]:
```
# More Than 97% Of Rows Are Null Of This Column, There's No Point To Keeping It, So Let's
df["STATION_NAME"].value_counts()
```

Out[9]:

BROADWAY-EAST NEW YORK	89
125 STREET	88
BROADWAY-EASTERN PKWY	75
42 ST.-PORT AUTHORITY BUS TERM	70
42 ST.-TIMES SQUARE	62
161 ST.-YANKEE STADIUM	54
W. 4 STREET	50
14 STREET	49
59 ST.-COLUMBUS CIRCLE	47
34 STREET	46
34 ST.-PENN STATION	42
168 ST.-WASHINGTON HTS.	42
42 ST.-GRAND CENTRAL	41
96 STREET	39
FRANKLIN AVENUE	38
KINGSBRIDGE ROAD	38
145 STREET	34
ATLANTIC AVENUE	34
ROCKAWAY AVENUE	31
34 ST.-HERALD SQ.	30
UTICA AVE.-CROWN HEIGHTS	30
DEKALB AVENUE	30
STILLWELL AVENUE-CONEY ISLAND	28
149 ST.-GRAND CONCOURSE	28
170 STREET	28
ROOSEVELT AVE.-JACKSON HEIGHTS	27
23 STREET	26
86 STREET	25
59 STREET	25
FULTON STREET	24
NOSTRAND AVENUE	23
MYRTLE AVENUE	22
HUNTS POINT AVENUE	22
QUEENSBORO PLAZA	22
3 AVENUE-149 STREET	21
36 STREET	21
CANAL STREET	21
42 STREET	21
FORDHAM ROAD	20
EUCLID AVENUE	20
JAY STREET-BOROUGH HALL	19
PACIFIC STREET	19
74 ST.-BROADWAY	18
116 STREET	18
EAST 180 STREET	18
HOYT-SCHERMERHORN	17
LIVONIA AVENUE	17
167 STREET	17
SIMPSON STREET	17
28 STREET	16
135 STREET	16
NEW LOTS AVENUE	16
71 AVE.-FOREST HILLS	15
CHURCH AVENUE	15
MYRTLE/WYCKOFF AVENUES	15
SUTTER AVENUE	14
CHAMBERS STREET	14
111 STREET	14
JUNCTION BLVD.	14
5 AVENUE	14
UNION SQUARE	14
WOODLAWN	13
PARSONS BLVD.	13
82 ST.-JACKSON HEIGHTS	13
14 ST.-UNION SQUARE	13
181 STREET	13
SUTPHIN BLVD.-ARCHER AVE.	13
GUN HILL ROAD	12
VAN SICLEN AVENUE	12
103 ST.-CORONA PLAZA	12
MAIN ST.-FLUSHING	12
FLATBUSH AVE.-BROOKLYN COLLEGE	12
47-50 STS./ROCKEFELLER CTR.	12
DELANCEY STREET	12
72 STREET	12
BROOK AVENUE	12
TREMONT AVENUE	12
50 STREET	12
BROADWAY/NASSAU	11
BOROUGH HALL	11
HOYT STREET	11
ESSEX STREET	11
ROCKAWAY PKWY-CANARSIE	11
61 ST.-WOODSIDE	11
KINGS HIGHWAY	11
LEXINGTON AVE.	11
49 STREET	11
CHAMBERS ST.-WORLD TRADE CENTE	11
MARCY AVENUE	10
BRONX PARK EAST	10
PRESIDENT STREET	10
BROOKLYN BRIDGE-CITY HALL	10
110 ST.-CENTRAL PARK NORTH	10
205 ST.-NORWOOD	10
148 ST.-HARLEM	10
BEDFORD PK. BLVD.	9
CITY HALL	9
57 STREET	9
FLUSHING AVENUE	9
LEXINGTON AVENUE	9
BURNSIDE AVENUE	9
GRAND AVE.-NEWTON	9
BEDFORD AVENUE	9
PARSONS/ARCHER-JAMAICA CENTER	9
LORIMER STREET	9
200 ST.-DYCKMAN ST.	9
215 STREET	9
BROADWAY/LAFAYETTE	9
WEST 34 STREET/HUDSON YARDS	8
ROCKAWAY BLVD.	8
46 STREET	8
GRANT AVENUE	8
1 AVENUE	8
KINGSTON-THROOP AVENUES	8
68 ST.-HUNTER COLLEGE	8
25 STREET	8
NEVINS STREET	8
PELHAM BAY PARK	8
HIGH STREET	8
EAST BROADWAY	8
103 STREET	8
JACKSON AVENUE	8
FAR ROCKAWAY-MOTT AVE.	8
RALPH AVENUE	7
66 ST.-LINCOLN CENTER	7
GRAND ARMY PLAZA	7
2 AVENUE	7
METROPOLITAN AVENUE	7
UTICA AVENUE	7
77 STREET	7
PELHAM PKWY.	7
CLINTON-WASHINGTON AVENUES	7
HOUSTON STREET	7
8 AVENUE	7

BEVERLY ROAD	6
157 STREET	6
110 ST.-CATHEDRAL PKWY.	6
WILLETS POINT-SHEA STADIUM	6
96TH STREET	6
33 STREET	6
183 STREET	6
174-175 STREETS	6
KOSCIUSKO STREET	6
207 ST.-INWOOD	6
137 ST.-CITY COLLEGE	6
NEWKIRK AVENUE	6
GRAND STREET	6
SMITH-9 STREETS	6
3 AVENUE-138 STREET	6
SARATOGA AVENUE	6
WINTHROP STREET	5
BUSHWICK AVE.-ABERDEEN ST.	5
PROSPECT AVENUE	5
FREEMAN STREET	5
BOWLING GREEN	5
SOUNDVIEW AVENUE	5
53 STREET	5
169 STREET	5
EAST 174 STREET	5
BOTANIC GARDEN	5
18 AVENUE	5
INTERVALE AVENUE	5
SPRING STREET	5
182-183 STREETS	5
9 AVENUE	5
EAST 105 STREET	5
138 ST.-GRAND CONCOURSE	5
BERGEN STREET	5
PROSPECT PARK	5
51 STREET	5
CHAUNCEY STREET	5
VERNON BLVD.-JACKSON AVE.	5
179 ST.-JAMAICA	5
155 STREET	5
52 STREET	5
ELDER AVENUE	5
BAY RIDGE AVENUE	5
HALSEY STREET	5
SUTTER AVENUE-RUTLAND ROAD	5
MORGAN AVENUE	5
BEDFORD PK. BLVD.-LEHMAN COLLE	5
90 ST.-ELMHURST AVE.	5
DITMARS BLVD.-ASTORIA	5
176 STREET	5
FRANKLIN STREET	4
242 ST.-VAN CORTLANDT PARK	4
39 AVENUE	4
LEFFERTS BLVD.	4
EAST 177 ST.-PARKCHESTER	4
238 ST.-NEREID AVE.	4
3 AVENUE	4
231 STREET	4
CHRISTOPHER ST.-SHERIDAN SQ.	4
BEACH 67 STREET	4
COURT SQUARE	4
COURT STREET	4
HEWES STREET	4
CYPRESS AVENUE	4
VAN WYCK BLVD.-BRIARWOOD	4
ROOSEVELT ISLAND	4
HOWARD BEACH-JFK AIRPORT	4
25 AVENUE	4
WILSON AVENUE	4
6 AVENUE	4
LAFAYETTE AVENUE	4
BRIGHTON BEACH	4
BROAD CHANNEL	4
95 STREET-BAY RIDGE	4
MT. EDEN AVENUE	4
69 STREET	4
JEFFERSON STREET	4
FORT HAMILTON PKWY	4
STEINWAY ST.	4
RECTOR STREET	4
LIBERTY AVENUE	4
QUEENS PLAZA	4
9TH STREET	4
20 AVENUE	4
AQUEDUCT-RACETRACK	4
NORTHERN BLVD.	4
CORTLANDT STREET	3
DISTRICT 33 OFFICE	3
CENTRAL AVENUE	3
NEW UTRECHT AVENUE	3
116 ST.-COLUMBIA UNIVERSITY	3
AVENUE "J"	3
BROADWAY	3
EAST TREMONT AV.-WESTCHESTER S	3
72ND STREET	3
191 STREET	3
SENECA AVENUE	3
4 AVENUE-9 STREET	3
ASTORIA BLVD.	3
SOUTH FERRY	3
GRAHAM AVENUE	3
YORK STREET	3
241 ST.-WAKEFIELD	3
WHITEHALL ST.-SOUTH FERRY	3
CRESCENT STREET	3
BEACH 25 STREET	3
ROCKAWAY PARK-BEACH 116 ST.	3
110 STREET	3
EAST 149 STREET	3
40 STREET	3
EAST TREMONT AVE.-WEST FARMS S	3
7TH AVENUE	3
102 STREET	3
NASSAU AVENUE	3
MOSHOLU PKWY.	3
LAWRENCE STREET	3
CARROLL STREET	3
21 ST.-QUEENSBRIDGE	3
EASTERN PKWY-BROOKLYN MUSEUM	2
NORWOOD AVENUE	2
PARKSIDE AVENUE	2
67 AVENUE	2
BROAD STREET	2
CORTELYOU ROAD	2
ALLERTON AVENUE	2
PENNSYLVANIA AVENUE	2
DYRE AVE.-EASTCHESTER	2
ST. LAWRENCE AVENUE	2
GREENPOINT AVENUE	2
WOODHAVEN BLVD.	2
UNION TURNPIKE-KEW GARDENS	2
SUTPHIN BLVD.	2
DISTRICT 4 OFFICE	2
UNION STREET	2

```
AVENUE "I"                          2
AVENUE "H"                          2
45 ROAD-COURT HOUSE SQUARE          2
81 ST.-MUSEUM OF NATURAL HISTO      2
WALL STREET                         2
PRINCE STREET                       2
75 AVENUE                           2
79 STREET                           2
HUNTERS POINT AVENUE                2
MONTROSE AVENUE                     2
63 DRIVE-REGO PARK                  2
18 STREET                           2
FRESH POND ROAD                     2
JAMAICA-VAN WYCK                    2
NEPTUNE AVENUE                      2
SHEPHERD AVENUE                     2
7 AVENUE                            2
BOWERY                              2
45 STREET                           2
175 STREET                          2
MORRIS PARK                         2
80 STREET                           2
ELMHURST AVE.                       2
23 STREET-ELY AVENUE                2
BLEECKER STREET                     2
WYCKOFF AVENUE                      2
BEDFORD-NOSTRAND AVENUES            2
163 ST.-AMSTERDAM AVE.              1
8 ST.-NYU                           1
CYPRESS HILLS                       1
NECK ROAD                           1
AVENUE "M"                          1
207 STREET                          1
71 STREET                           1
225 ST.-MARBLE HILL                 1
BEACH 60 STREET                     1
BEACH 98 STREET                     1
CLEVELAND STREET                    1
219 STREET                          1
15 ST.-PROSPECT PARK                1
233 STREET                          1
ASTOR PLACE                         1
BAY PARKWAY                         1
86TH STREET                         1
SHEEPSHEAD BAY                      1
MYRTLE-WILLOUGHBY AVENUES           1
WEST 8 STREET-NY AQUARIUM           1
AVENUE "U"                          1
85 ST.-FOREST PKWY.                 1
GATES AVENUE                        1
BEACH 44 STREET                     1
225 STREET                          1
JUNIUS STREET                       1
BEVERLEY ROAD                       1
KINGSTON AVENUE                     1
BURKE AVENUE                        1
104 STREET                          1
62 STREET                           1
CLARK STREET                        1
75 ST.-ELDERTS LANE                 1
BEACH 36 STREET                     1
65 STREET                           1
DISTRICT 32 OFFICE                  1
88 STREET                           1
FOREST AVENUE                       1
55 STREET                           1
AQUEDUCT-NORTH CONDUIT AVE.         1
DISTRICT 34 OFFICE                  1
ALABAMA AVENUE                      1
190 STREET                          1
Name: STATION_NAME, dtype: int64
```

In [10]:
```
# Checking If These Two Columns Are Duplicates
# I'm Not So Good With DateTime Datatypes,
# But If I Code Cleanly With Awareness, We Will Definitely Find Some Unique Insights
df['CMPLNT_TO_DT'].equals(df["CMPLNT_FR_DT"])
```

Out[10]:
```
False
```

In [11]:
```
# These Are Mostly Null Columns Or Their Rows Are So Less It's Not Worth Investigating.
# So I Think It's Best To Drop Them.
df.drop(["HADEVELOPT",
         "JURISDICTION_CODE",
         "TRANSIT_DISTRICT",
         "PARKS_NM",
         "STATION_NAME",
         "TRANSIT_DISTRICT",
         "HOUSING_PSA","X_COORD_CD","Y_COORD_CD"], axis=1, inplace=True)
```

In [12]:
```
# Checking For Duplicates
df[df.duplicated()]
# There Are No Duplicate Rows
```

Out[12]:

CMPLNT_NUM	ADDR_PCT_CD	BORO_NM	CMPLNT_FR_DT	CMPLNT_FR_TM	CMPLNT_TO_DT	CMPLNT_T

In [13]:
```
# After A Lot Of Cleaning There Still Remains Tons Of Null Values
# If We Drop Them All We Will Have An Insignificant Dataset. It Will Be Inaccurate To Rea
# So We Will Analyse Each Column Carefully From Now On.

df.isnull().sum().sort_values(ascending=False)
```

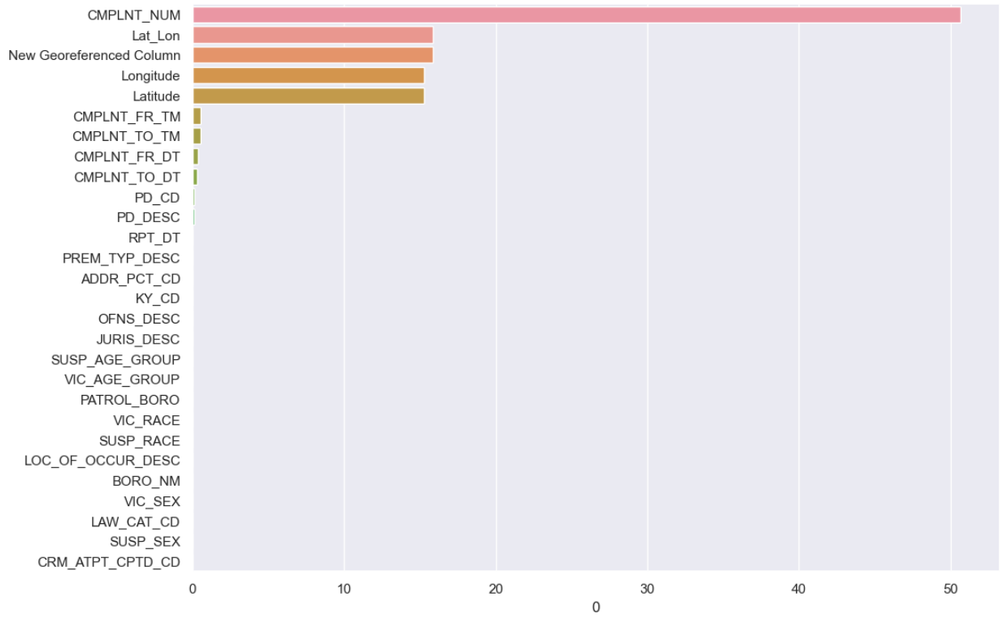
Out[13]:
```
SUSP_AGE_GROUP              67493
SUSP_RACE                   52777
SUSP_SEX                    47159
VIC_RACE                    42130
VIC_AGE_GROUP               39155
VIC_SEX                     36617
LOC_OF_OCCUR_DESC           24881
CMPLNT_TO_DT                 9056
CMPLNT_TO_TM                 8993
PREM_TYP_DESC                1690
BORO_NM                       213
PD_DESC                       116
PD_CD                         116
ADDR_PCT_CD                    14
OFNS_DESC                       3
Longitude                       1
Latitude                        1
Lat_Lon                         1
New Georeferenced Column        1
RPT_DT                          0
PATROL_BORO                     0
LAW_CAT_CD                      0
KY_CD                           0
JURIS_DESC                      0
CRM_ATPT_CPTD_CD                0
CMPLNT_FR_TM                    0
CMPLNT_FR_DT                    0
CMPLNT_NUM                      0
dtype: int64
```

In [14]:
```
# The Code Below Calculates The Percentage Of Unique Values From Every Columns
unique  = df.nunique().sort_values(ascending=False) /256797 *100
unique = pd.DataFrame(unique)
unique
```

Out[14]:

	0
CMPLNT_NUM	50.690623
Lat_Lon	15.854547
New Georeferenced Column	15.854547
Longitude	15.256019
Latitude	15.253293
CMPLNT_FR_TM	0.560754
CMPLNT_TO_TM	0.560754
CMPLNT_FR_DT	0.387076
CMPLNT_TO_DT	0.278041
PD_CD	0.129674
PD_DESC	0.126170
RPT_DT	0.035047
PREM_TYP_DESC	0.031932
ADDR_PCT_CD	0.029985
KY_CD	0.024533
OFNS_DESC	0.023365
JURIS_DESC	0.007399
SUSP_AGE_GROUP	0.005841
VIC_AGE_GROUP	0.005452
PATROL_BORO	0.003115
VIC_RACE	0.002336
SUSP_RACE	0.002336
LOC_OF_OCCUR_DESC	0.001947
BORO_NM	0.001947
VIC_SEX	0.001168
LAW_CAT_CD	0.001168
SUSP_SEX	0.000779
CRM_ATPT_CPTD_CD	0.000779

In [15]:
```
# Plotting Unique Using Barplot
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.barplot(data=unique, y=unique.index, x=0);
```



Region-wise Crime Analysis

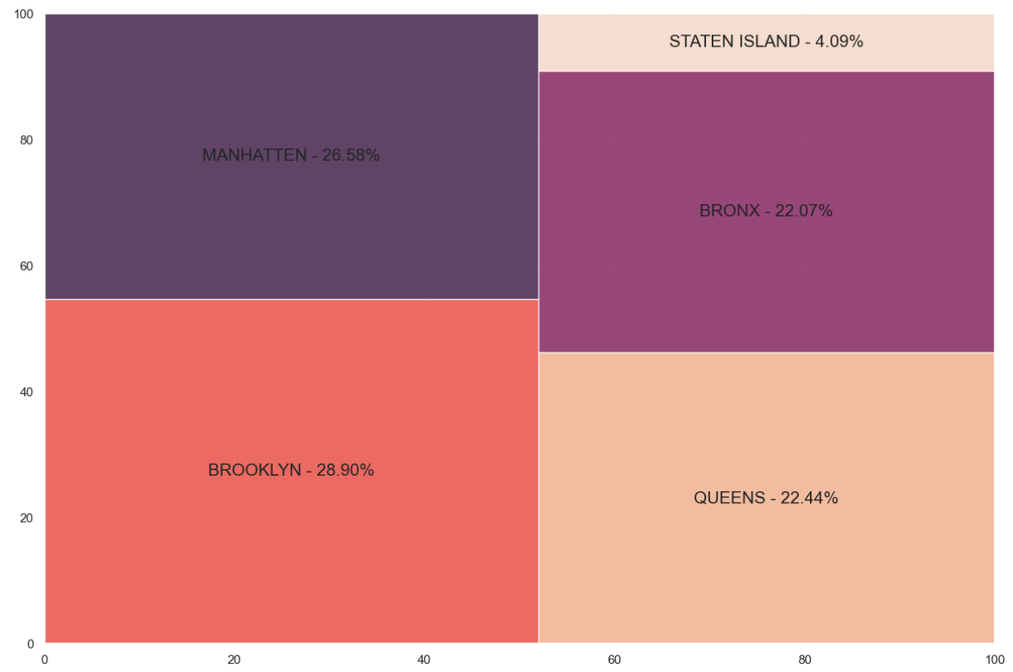
> *In This Section, I Will Analyse Crime By Region, Certain Regions Are Prone To High Crime Activity While There Could Be Many Reasons For That Happening To Present An Accurate/precise Picture, We Need To Be Unbiased Of Anything Like (race, Religion, Sex, Origin Of The Person, Typical Stereotypes) And Then Only We'll Get A Precise/real Picture*

Insights:-

> - Brooklyn Contains The Highest Share Of Crime In New York City At 28.90%
> - The Lowest Would Be Staten Island At 4.09%
> - The 75th Precint Of Brooklyn Has The Highest Count Of Crime At 8834

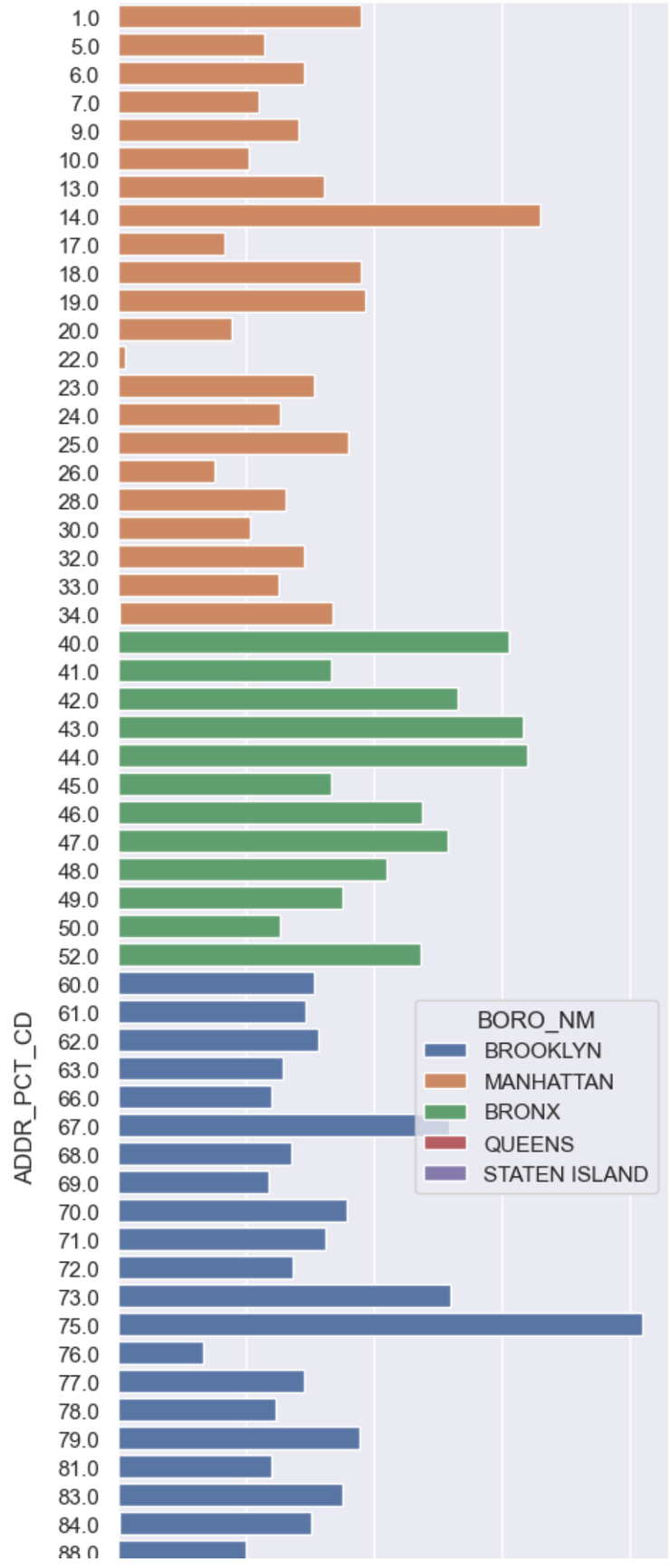
In [16]:
```
# Plotting A Treemap Using Sqaurify
# As I Have Mentioned Above We Have To Drop Nulls For Every Column "individually"
# The Colour Of This Tree Map Will Change Randomly Every Time You Run The Code, It's A Go

br = df["BORO_NM"].value_counts().head(5)
br = pd.DataFrame(br)
br.dropna(inplace=True)
lbl = [str('{:5.2f}'.format(i/br['BORO_NM'].sum()*100)) + "%" for i in br['BORO_NM']]
plt.figure(figsize=(15,10))
squarify.plot(sizes=br["BORO_NM"],
              label=["BROOKLYN - 28.90%",
                     "MANHATTEN - 26.58%",
```

```
                "QUEENS - 22.44%",
                "BRONX - 22.07%",
                "STATEN ISLAND - 4.09%"],
          alpha=0.8,
          text_kwargs={'fontsize':15});
```



In [17]:

```
# Plotting Barplot Of Precincts In NYC City
precint = df.iloc[:,[1,2]]
precint = precint.value_counts(ascending=False)
precint = pd.DataFrame(precint)
precint.reset_index(inplace=True)
precint.drop(precint[precint['BORO_NM'] == "(null)"].index, inplace = True)
sns.set(rc={'figure.figsize':(5,20)})
sns.barplot(data=precint, y="ADDR_PCT_CD",x=0, orient="h", hue ='BORO_NM',dodge=False);
```



In [18]:
```
# Converting These Columns To DateTime
df["CMPLNT_FR_DT"] = pd.to_datetime(df["CMPLNT_FR_DT"], errors = 'coerce')
df["CMPLNT_TO_DT"] = pd.to_datetime(df["CMPLNT_TO_DT"], errors = 'coerce')
df["CMPLNT_FR_TM"] = pd.to_datetime(df["CMPLNT_FR_TM"])
df["CMPLNT_TO_TM"] = pd.to_datetime(df["CMPLNT_TO_TM"])
df["RPT_DT"] = pd.to_datetime(df["RPT_DT"])


# This Column ["VIC_AGE_GROUP"] Gave Me A Lot Of Trouble, I Have To Replace All Those Bog
# You Will See At The End Of The Notebook There's A Beautiful Map Of NYC Filtered By Age

df['VIC_AGE_GROUP']= df['VIC_AGE_GROUP'].replace(['-11','-4',
                                                  '-40','-5',
                                                  '-65','-934',
                                                  '-955','964',
                                                  '-960','-959',
                                                  '-963','-970',
                                                  '-964','-971'],("UNKNOWN"))
df["VIC_AGE_GROUP"].fillna("UNKNOWN", inplace=True)
```

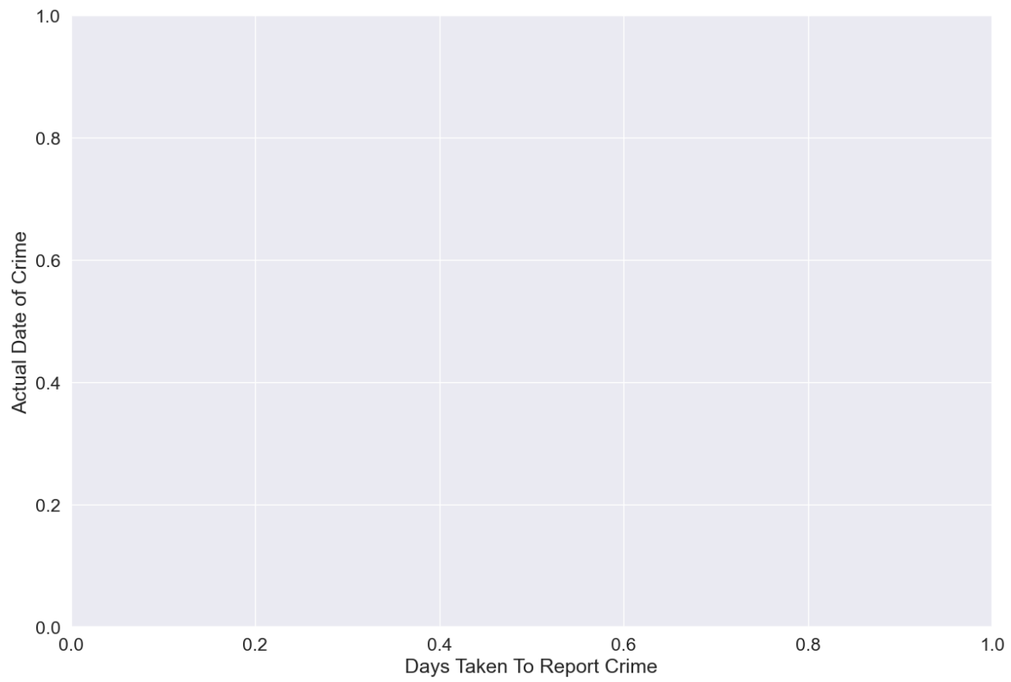
Time Series Analysis

> A Time Series Is A Series Of Data Points Indexed (or Listed Or Graphed) In Time Order. Most Commonly, A Time Series Is A Sequence Taken At Successive Equally Spaced Points In Time. A Time Series Is Very Frequently Plotted Via A Run Chart (which Is A Temporal Line Chart). Time Series Are Used In Statistics, Signal Processing, Pattern Recognition, Econometrics, Mathematical Finance, Weather Forecasting, Earthquake Prediction, Electroencephalography, Control Engineering, Astronomy, Communications Engineering, And Largely In Any Domain Of Applied Science And Engineering Which Involves Temporal Measurements.

Insights:-

> - We Can See A Trend From 2021-07 To Till Now That The Time Between Crime Occurred And Crime Reported To Police Has Shrunk Significantly
> - From 2021 We See That NYPD Was Responding To Complaints Much Faster That Years Before That
> - At Df.loc[2], The Person Of This Complaint Took More Than 1440 Days To Report A Crime, Run This Command On Your Own, I Think There's A Sad Story With This
> - Most Common Time For Crimes Are 12pm, 3pm, 5pm

In [19]:
```
# Plotting A Lineplot Type Graph That Tells The Time Difference Between The Actual Date O
# This Variable X_86 Has An Interesting Story, So Numpy Was Returning Time Difference In
# But I Wanted Time Difference In Days, So If You Divide "nanosecond..values",
#By X_86 You Get The Result In Days Rather Than Nanoseconds
x_86 = 86_400_000_000_000
rpt = df[(df["CMPLNT_FR_DT"] > "2018-01-01") & (df["CMPLNT_FR_DT"] < "2022-08-14") & (df[
rpt = rpt[["CMPLNT_FR_DT","RPT_DT"]]
rpt["nw"] =  rpt["RPT_DT"] - rpt["CMPLNT_FR_DT"]
sns.set(rc={'figure.figsize':(15,10)})
sns.set_style('darkgrid')
sns.set(font_scale=1.5)
pxx = sns.lineplot(data=rpt,
                   y="CMPLNT_FR_DT",
                   x=rpt["nw"]/x_86,
                    color="#34282C",
                    ci=None,
                   linewidth=.07,
                   marker="o",
                   markersize=4,
                  alpha=1)
pxx.set(ylabel='Actual Date of Crime', xlabel='Days Taken To Report Crime')
plt.show()
```



In [20]:
```
# This Victim Took More Than 1400 Days To Report Her Crime, I'm Sad For Her
df.loc[2]
```

Out[20]:

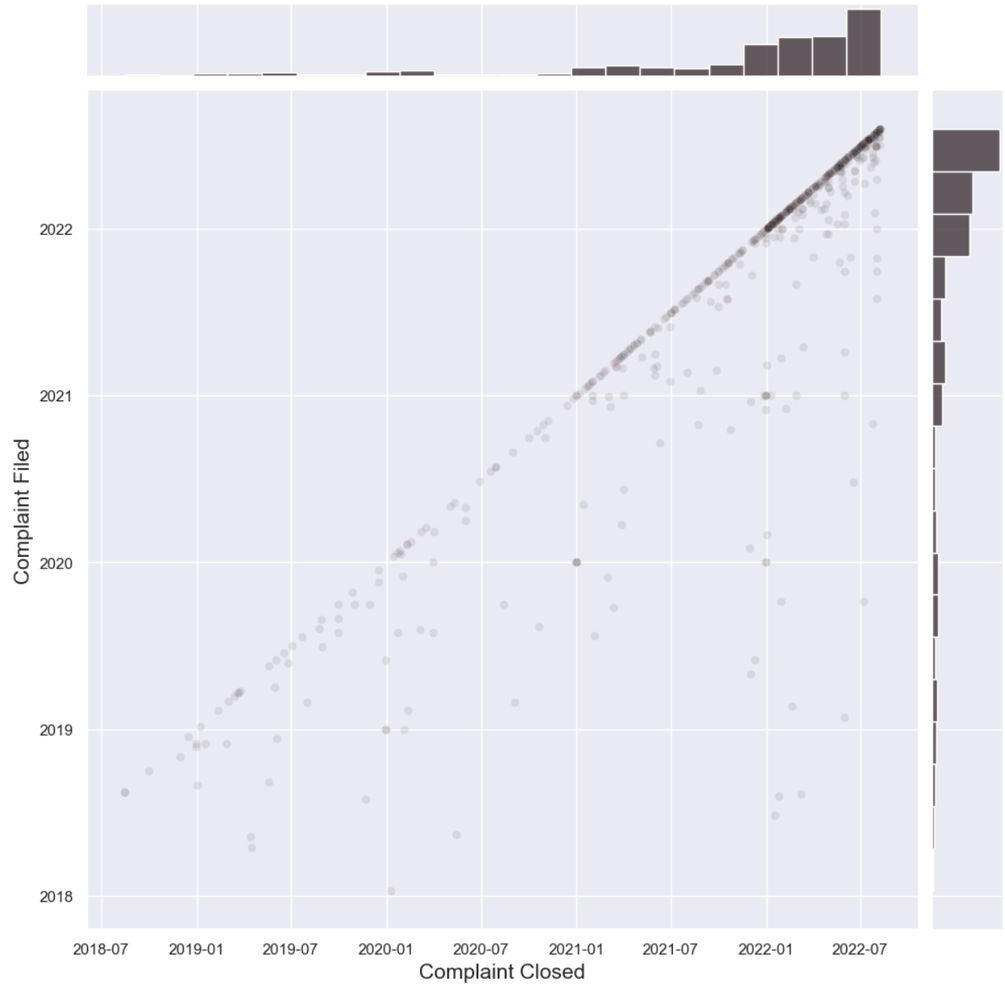
CMPLNT_NUM	262612530
ADDR_PCT_CD	110.0
BORO_NM	QUEENS
CMPLNT_FR_DT	2022-09-09 00:00:00
CMPLNT_FR_TM	2023-05-10 00:55:00
CMPLNT_TO_DT	2022-09-09 00:00:00
CMPLNT_TO_TM	2023-05-10 01:19:00
CRM_ATPT_CPTD_CD	COMPLETED
JURIS_DESC	N.Y. POLICE DEPT
KY_CD	578
LAW_CAT_CD	VIOLATION
LOC_OF_OCCUR_DESC	NaN
OFNS_DESC	HARRASSMENT 2
PATROL_BORO	PATROL BORO QUEENS NORTH
PD_CD	638.0
PD_DESC	HARASSMENT,SUBD 3,4,5
PREM_TYP_DESC	STREET
RPT_DT	2023-01-20 00:00:00
SUSP_AGE_GROUP	25-44
SUSP_RACE	BLACK HISPANIC
SUSP_SEX	M
VIC_AGE_GROUP	25-44
VIC_RACE	WHITE HISPANIC
VIC_SEX	F
Latitude	40.743481
Longitude	-73.874004
Lat_Lon	(40.7434812638841, -73.8740035373971)
New Georeferenced Column	POINT (-73.8740035373971 40.7434812638841)

Name: 2, dtype: object

In [21]:
```
# Plotting A Joint Grid That Tells When A Complaint Is Filed On Y-axis,
# And When A Complaint Closed On The X-axis

pw = df[(df["CMPLNT_FR_DT"] > "2018-01-01") & (df["CMPLNT_FR_DT"] < "2022-08-10") &(df["C
pw = pw[["CMPLNT_FR_DT","CMPLNT_TO_DT"]]
pw.reset_index(inplace=True,drop=True)
sns.set_style("darkgrid")
```

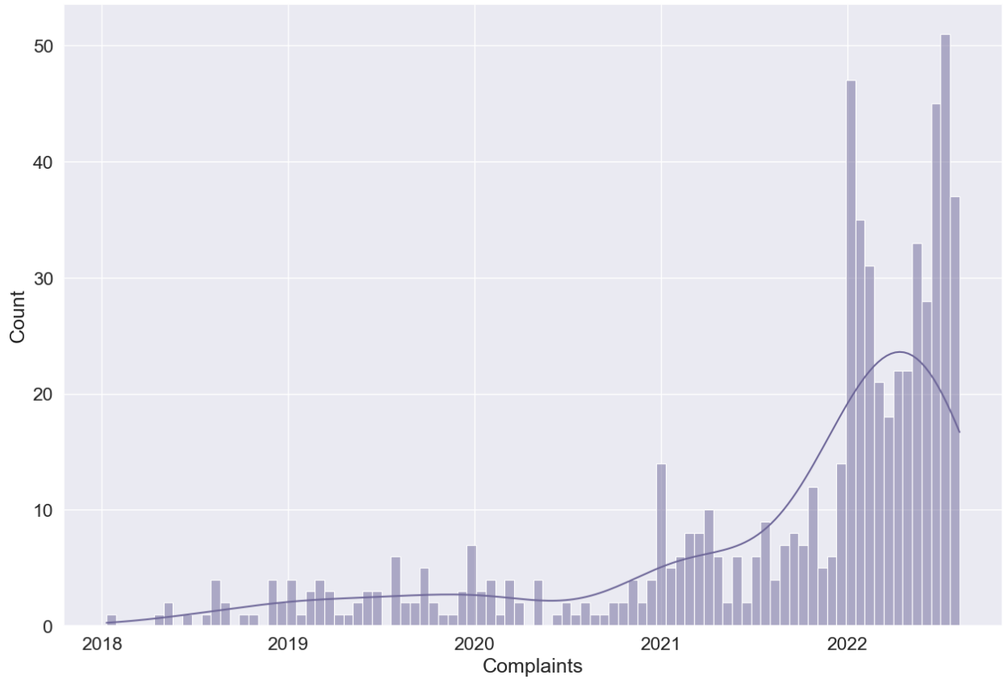
```
sns.set(font_scale=1)
pwc = sns.jointplot(data=pw,y="CMPLNT_FR_DT",
                    x='CMPLNT_TO_DT'
                    ,height=10
                    ,ratio=10
                    ,alpha=0.1
                    ,color="#34282C")
pwc.set_axis_labels('Complaint Closed','Complaint Filed',  fontsize=15)
```

Out[21]: <seaborn.axisgrid.JointGrid at 0x29fd34fca00>

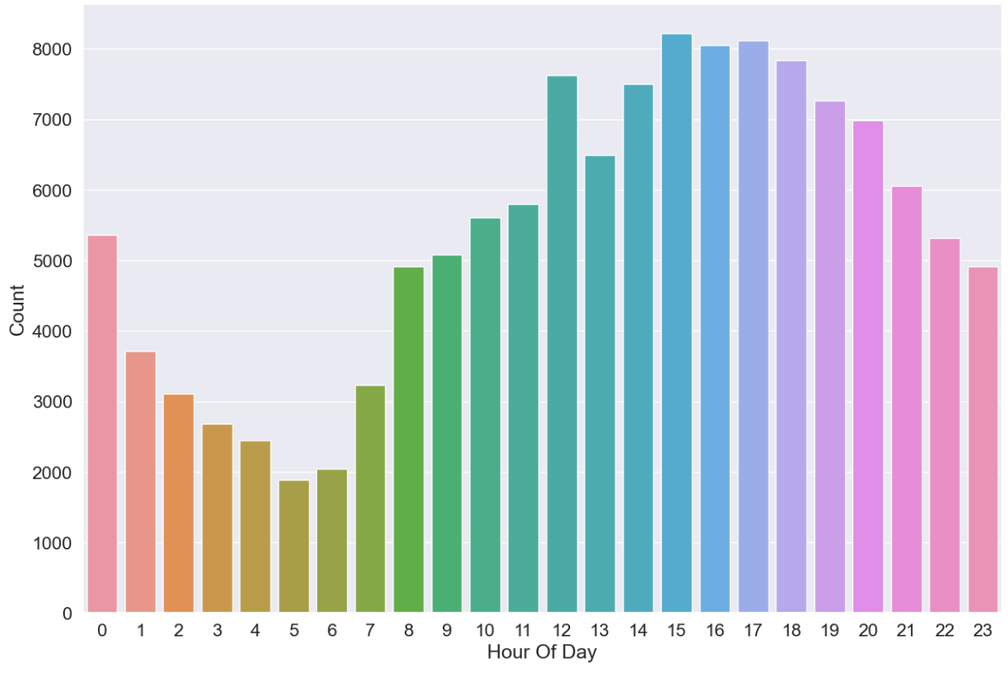


In [22]:
```
# well, self-explanatory.
sns.set(rc={'figure.figsize':(15,10)})
sns.set(font_scale=1.5)
ct=sns.histplot(data=pw,x="CMPLNT_FR_DT",bins=90 ,color="#6e6799",multiple="dodge",kde=Tr
ct.set(xlabel='Complaints')
```

Out[22]: [Text(0.5, 0, 'Complaints')]



In [23]:
```
# # Plotting Graph For The Most Common Hour For Crimes To Occur.
fr = df["CMPLNT_FR_TM"].dt.hour
fr = pd.DataFrame(fr)
fr.reset_index(inplace=True,drop=True)
frr=sns.countplot(data=fr,x="CMPLNT_FR_TM");
sns.set(rc={'figure.figsize':(12,8)})
sns.set(font_scale=1)
frr.set(xlabel='Hour Of Day', ylabel='Count');
```



Actual Crime Analysis

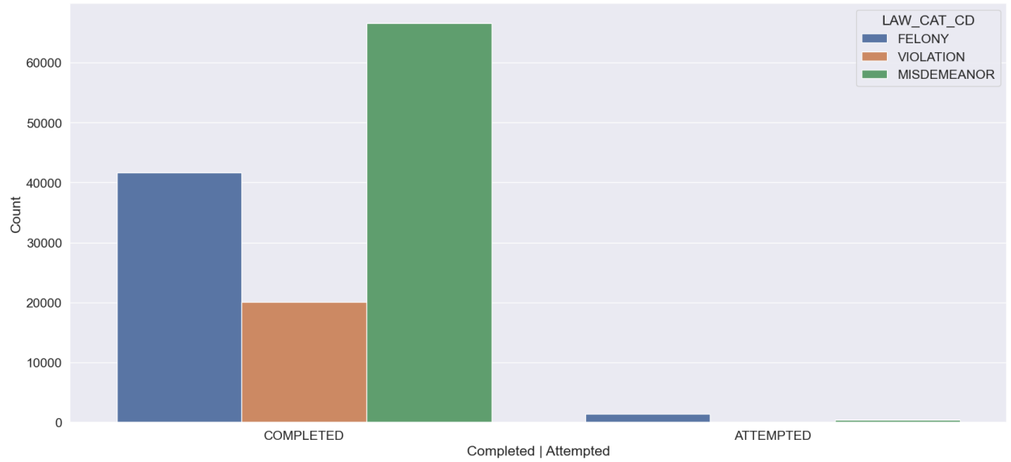
> *Crime Analysis Is A Law Enforcement Function That Involves Systematic Analysis For Identifying And Analyzing Patterns And Trends In Crime And Disorder. Information On Patterns Can Help Law Enforcement Agencies Deploy Resources In A More Effective Manner, And Assist Detectives In Identifying And Apprehending Suspects. Crime Analysis Also Plays A Role In Devising Solutions To Crime Problems And Formulating Crime Prevention Strategies. Quantitative Social Science Data Analysis Methods Are Part Of The Crime Analysis Process, Though Qualitative Methods Such As Examining Police Report Narratives Also Play A Role*

Insights:-

- Most Common Law Classification Is A Misdemeanour
- surprising People Commit More Felonies Than Violation
- NYPD Report Crimes Far-far Above Than Any Other Policing Authority In NYC, Next Is The NY Housing Police
- Most Crimes Occur Inside Of A Structure, Contrary To Popular Belief Of Crime Occuring Outside
- Petit Larceny, Harassment And Assault Are The Most Common Crimes According To Public
- HARASSMENT-SUBDIVISION(3,4,5), LARCENY, PETIT FROM STORE-SHOPL, ASSAULT - 3, Here I Wanted To Show How Police Departments Are More Articulate/proficient When Writing Descriptions Of Crime Than Public
- People Are Likely To Commit Crimes Against People Of Their Races

In [24]:
```
sns.set(rc={'figure.figsize':(20,9)})
sns.set(font_scale=1.4)
crm = sns.countplot(x="CRM_ATPT_CPTD_CD" ,data= df, hue="LAW_CAT_CD");
crm.set(xlabel='Completed | Attempted', ylabel='Count')
```

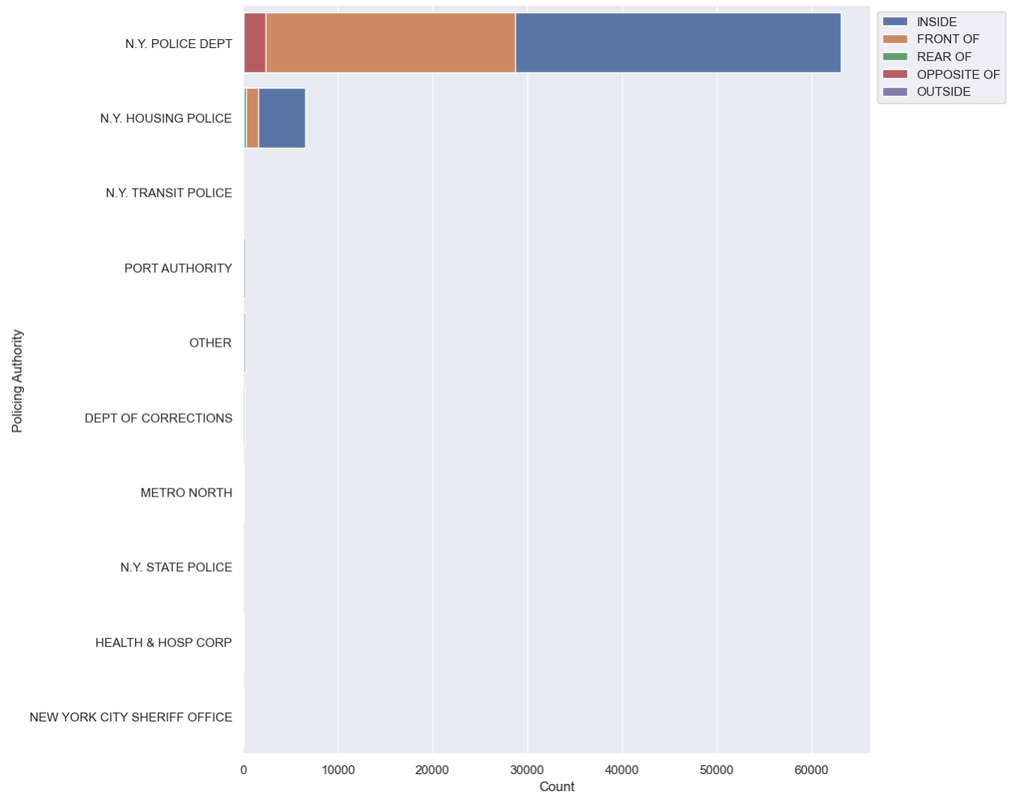
Out[24]:
```
[Text(0.5, 0, 'Completed | Attempted'), Text(0, 0.5, 'Count')]
```



In [25]:
```
# People Don't Commit Violations, I Guess
df[(df["LAW_CAT_CD"]=="VIOLATION")  & (df["CRM_ATPT_CPTD_CD"]=="ATTEMPTED")].count()
```

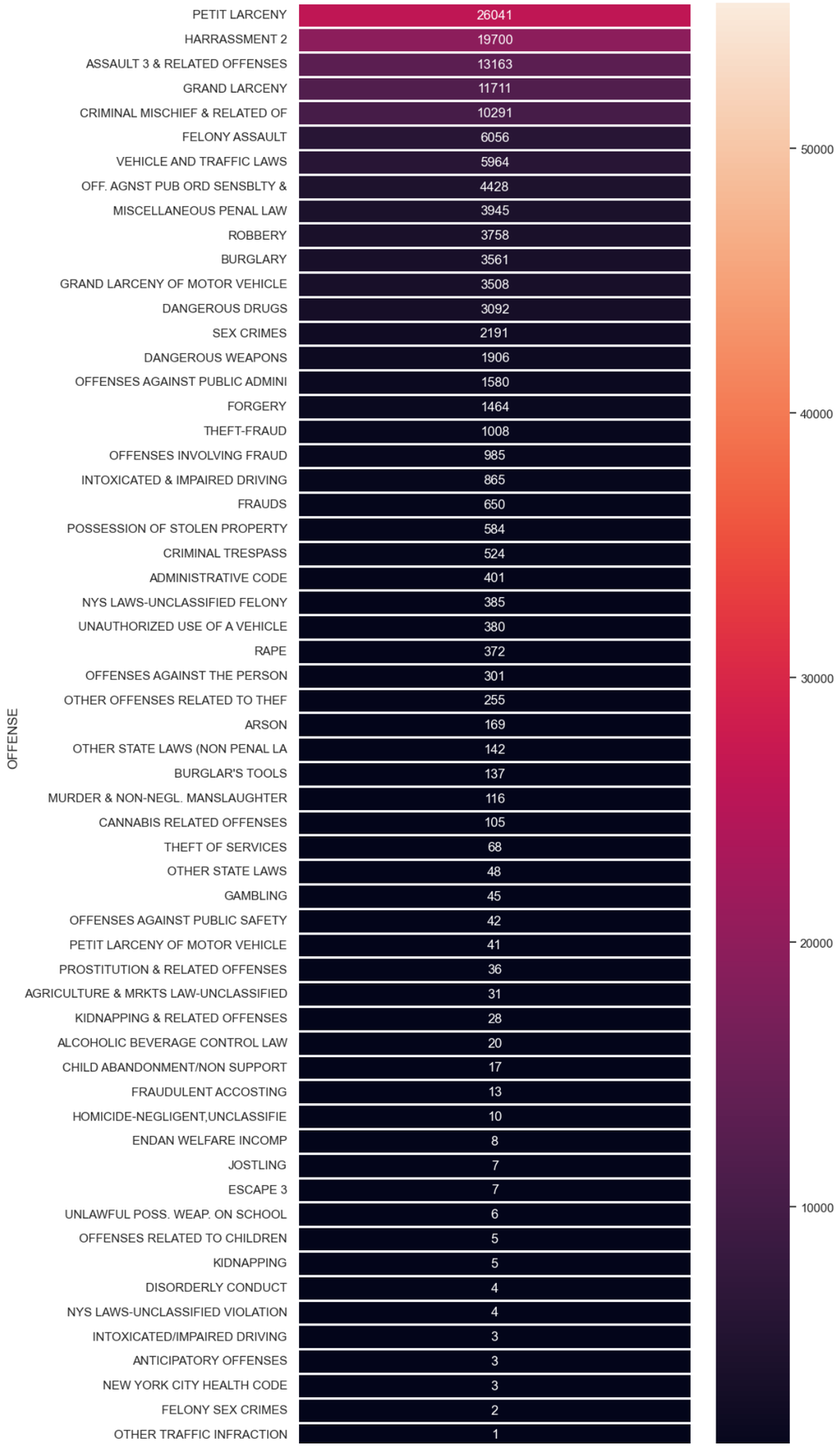
Out[25]:
```
CMPLNT_NUM                53
ADDR_PCT_CD               53
BORO_NM                   53
CMPLNT_FR_DT              53
CMPLNT_FR_TM              53
CMPLNT_TO_DT              50
CMPLNT_TO_TM              50
CRM_ATPT_CPTD_CD          53
JURIS_DESC                53
KY_CD                     53
LAW_CAT_CD                53
LOC_OF_OCCUR_DESC         45
OFNS_DESC                 53
PATROL_BORO               53
PD_CD                     53
PD_DESC                   53
PREM_TYP_DESC             51
RPT_DT                    53
SUSP_AGE_GROUP            29
SUSP_RACE                 38
SUSP_SEX                  41
VIC_AGE_GROUP             53
VIC_RACE                  45
VIC_SEX                   50
Latitude                  53
Longitude                 53
Lat_Lon                   53
New Georeferenced Column  53
dtype: int64
```

In [26]:
```
# Plotting A Countplot, Policing Authority On The Y-axis, Count On The X-axis, And Locati
sns.set(rc={'figure.figsize':(10,12)})
jur = sns.countplot(y=df["JURIS_DESC"],
                    data= df,hue="LOC_OF_OCCUR_DESC", dodge=False,
       order = df['JURIS_DESC'].value_counts().iloc[:10].index)
plt.legend(loc = 2, bbox_to_anchor = (1,1))
jur.set(xlabel='Count', ylabel='Policing Authority'),
jur = df[["JURIS_DESC","LOC_OF_OCCUR_DESC"]]
```

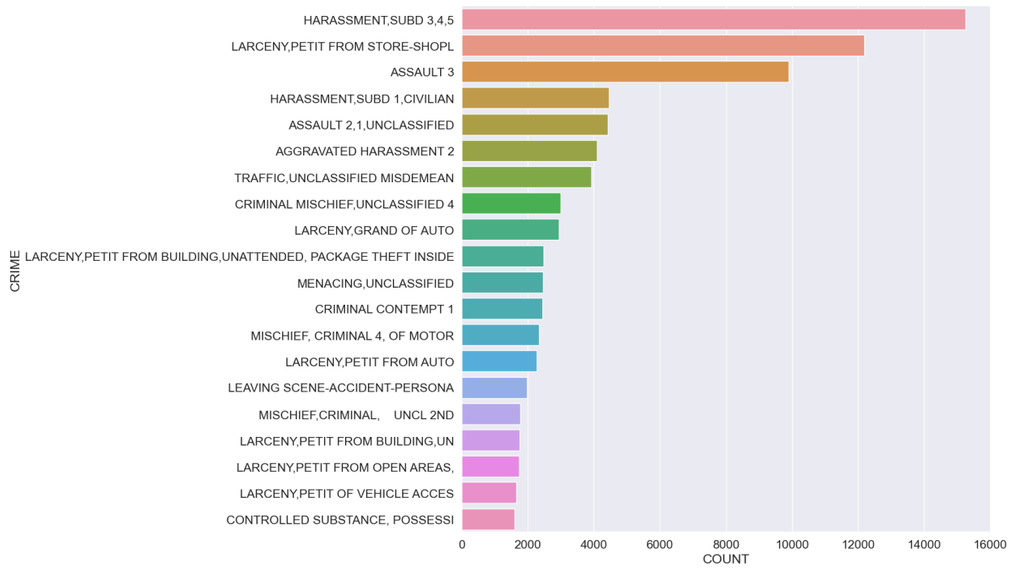


In [27]:
```
# Plotting Heatmap Of Crime Committed According To Public Description,
# See How Police Are So Articulate When Writing Descriptions Of A Crime
ofs = df.OFNS_DESC.value_counts()
ofs = pd.DataFrame(ofs)
label = ofs.index
sns.set(rc={'figure.figsize':(8,24)})
sof = sns.heatmap(ofs,
            vmin=0,
            vmax=55500,
            annot=True,
            linewidths=2,
            linecolor='white',
            fmt='g')
sof.set(xlabel="COUNT",ylabel="OFFENSE")
```

Out[27]:
```
[Text(0.5, 231.10937499999994, 'COUNT'),
 Text(67.24999999999999, 0.5, 'OFFENSE')]
```



In [28]:
```
# Plotting Top 20 Crimes According To Police,
# See How These Are Articulated Comparing Them To The Public Description.
pdd = df.PD_DESC.value_counts()
pdd = pd.DataFrame(pdd).head(20)
sns.set(rc={'figure.figsize':(12,12)})
sns.set(font_scale=1.4)
pddd = sns.barplot(data=pdd,x='PD_DESC',y=pdd.index);
pddd.set(xlabel="COUNT",ylabel="CRIME"),pdd;
```



Race-wise Crime Analysis

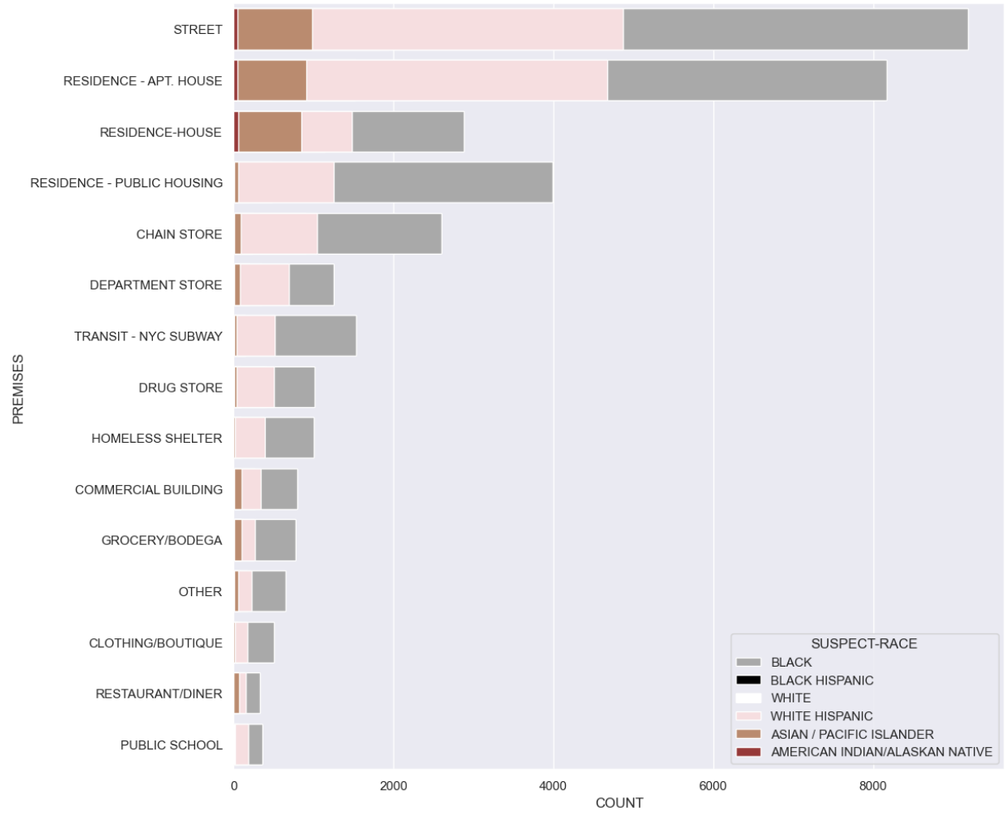
> *Research Suggests That Police Practices, Such As Racial Profiling, Over-policing In Areas Populated By Minorities And In-group Bias May Result In Disproportionately High Numbers Of Racial Minorities Among Crime Suspects. Research Also Suggests That There May Be Possible Discrimination By The Judicial System, Which Contributes To A Higher Number Of Convictions For Racial Minorities. I Will Try To Figure Out If It's True*

Insights:-

> - African Americans Are Suspects Has The Highest Share In The Top 15 Crimes, And Are More Likely In Others
> - Apartment, Street, Residence House, Chain Store, Drug Store Are The Top Places Where Crimes Are Most Likely To Occur
> - Females Are Very Much Likely To Be Victims Of African American Suspects. Way More Than Any Other Race
> - people are likely to commit crimes to the people of thier races

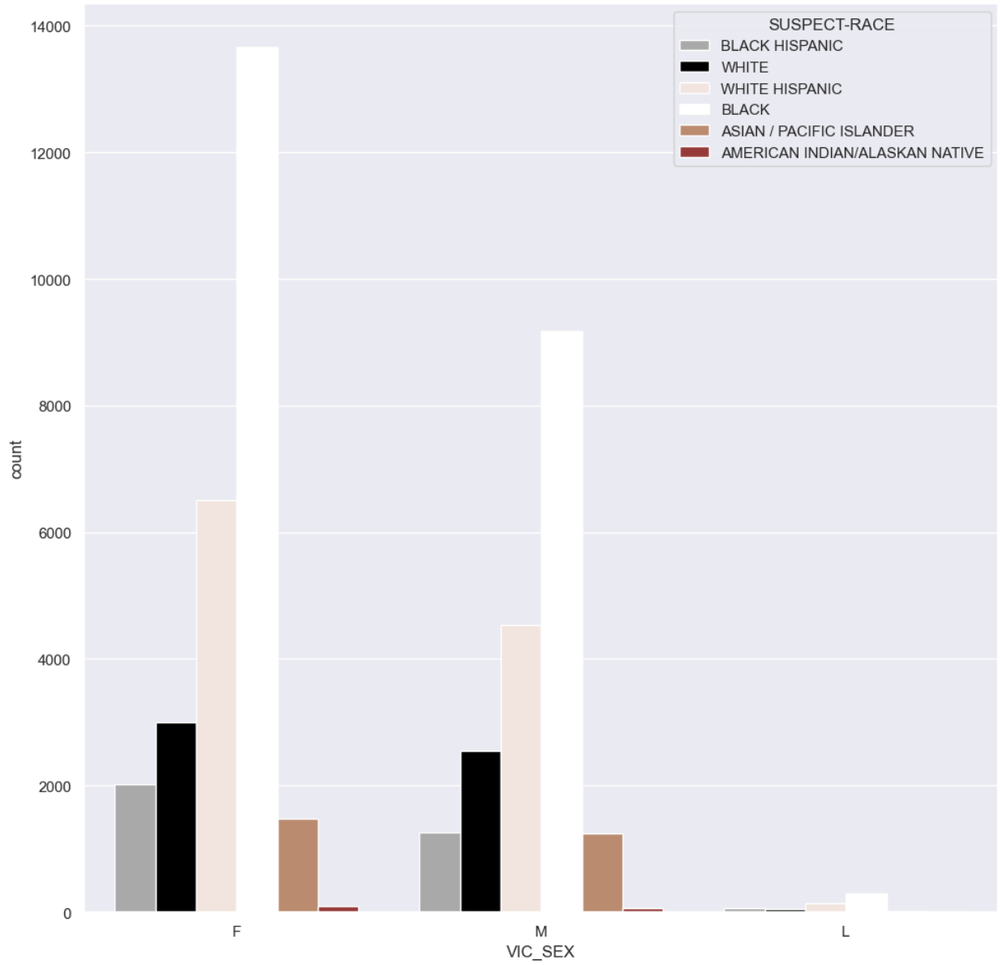
In [29]:
```
# Plotting  Premises Where Crimes Occurred By Race Hue
pre = df[["PREM_TYP_DESC",'SUSP_RACE']]
pre.dropna(inplace=True)
pre.reset_index(drop=True,inplace=True)
sns.set(rc={'figure.figsize':(12,12)})
pre = sns.countplot(data=pre, y=pre['PREM_TYP_DESC'], dodge=False,
              hue="SUSP_RACE",
              order = pre['PREM_TYP_DESC'].value_counts().iloc[0:15].index,
              palette=(sns.color_palette(["#A9A9A9",
                                          "#000000",
                                          "#FFFFFF",
                                          "#fadadd",
                                          "#C68863",
                                          "#A52A2A"],
                                         n_colors=6)))
pre.set(xlabel="COUNT",ylabel="PREMISES")
plt.legend(title="SUSPECT-RACE")
```

Out[29]:
```
<matplotlib.legend.Legend at 0x29fd22ec130>
```



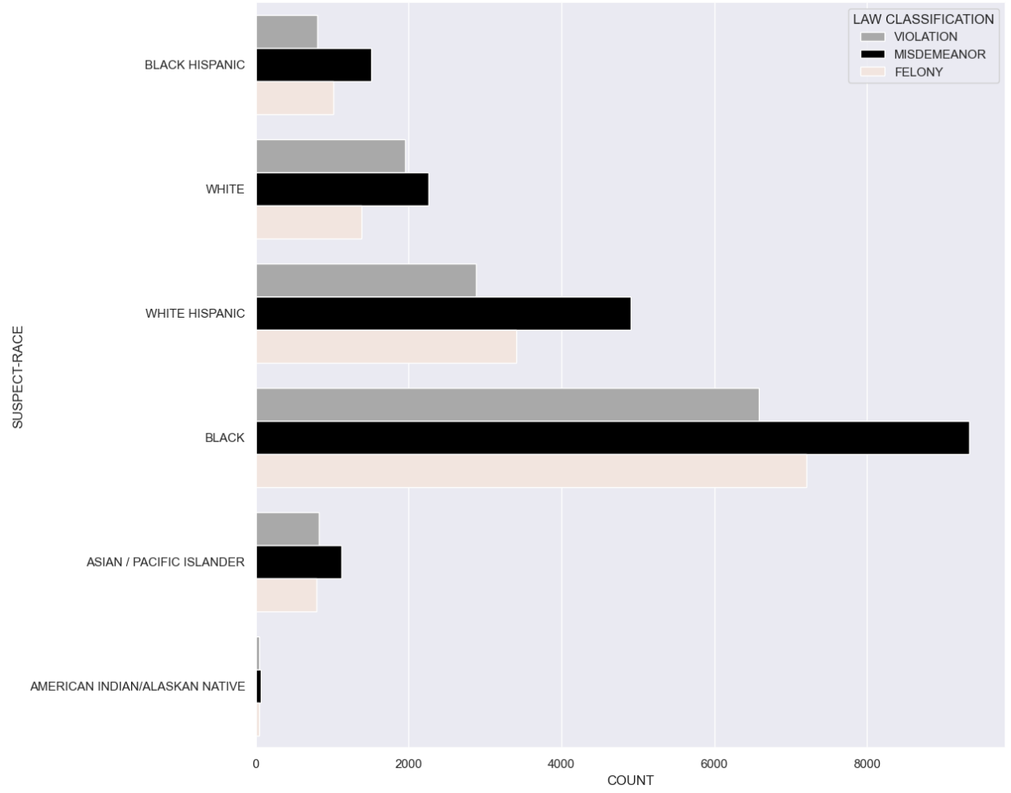
In [30]:
```
vic=df[['VIC_SEX','SUSP_RACE',"LAW_CAT_CD", "VIC_RACE","SUSP_SEX"]]
vic.dropna(inplace=True)
sns.countplot(data=vic, x="VIC_SEX", hue="SUSP_RACE",
              palette=(sns.color_palette(["#A9A9A9","#000000","#F5E4DC",
                                          "#FFFFFF","#C68863","#A52A2A"], n_colors=6)))
plt.legend(title="SUSPECT-RACE")
```

Out[30]:
```
<matplotlib.legend.Legend at 0x29fd260a820>
```



In [31]:
```
lc = sns.countplot(data=vic,
                   y="SUSP_RACE",
                   hue="LAW_CAT_CD",
                   palette=(sns.color_palette(["#A9A9A9","#000000","#F5E4DC"],n_colors=3)))
plt.legend(title="LAW CLASSIFICATION")
lc.set(xlabel="COUNT",ylabel="SUSPECT-RACE")
```

Out[31]:
```
[Text(0.5, 0, 'COUNT'), Text(0, 0.5, 'SUSPECT-RACE')]
```



In [32]:
```
# People Are Likely To Commit Crimes Against People Of Their Races
sns.set(rc={'figure.figsize':(12,18)})
sns.set(font_scale=1.4)
vic_sus = sns.countplot(data=vic,
                        y="VIC_RACE",
                        hue="SUSP_RACE",
                        palette=(sns.color_palette(["#A9A9A9","#000000","#F5E4DC",
                                                    "#FFFFFF","#C68863","#A52A2A"],n_colors=6)))
vic_sus.set(xlabel="COUNT",ylabel="VICTIM-RACE");
plt.legend(title="SUSPECT-RACE" ,bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.01);
```

In [33]:
```
# This Is A Very Interesting Map Of NYC City Just Hover Over It
nod = df[["VIC_AGE_GROUP","Longitude","Latitude","PD_DESC"]]

mpx = plx.scatter_mapbox(nod,lon=nod["Longitude"],
                         lat=nod["Latitude"],
                         zoom=10,
                         width=890,
                         height=800,
                         hover_name ="PD_DESC",
                         opacity = 0.3,color="VIC_AGE_GROUP",
                         color_discrete_sequence=["red","white",
                                                  '#90ee90'," #00008B",
                                                 "#FFFFE0"]);

mpx.update_layout(mapbox_style="carto-darkmatter")
mpx.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
mpx.show()
```