NEW YORK

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

Importing Libraries And Loading Data

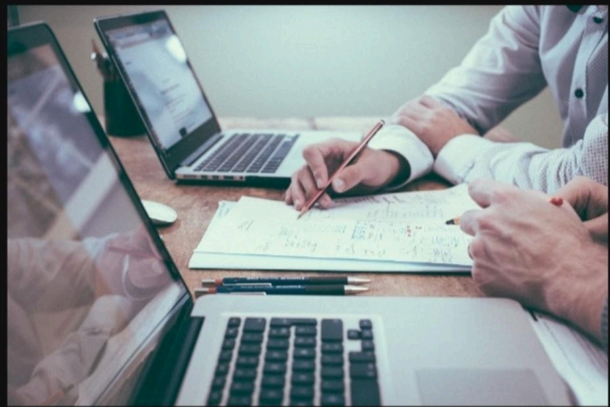
> *In This Section We Will Load The Necessary Libraries And Load Data:-*

n [1]:
```
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)
```

n [2]:
```
# I Have Found That Null Values Are Disguised In Many Other Names Such A
# So We Need To Include Them As Well And Drop When Necessary
nan_values = ['NO CLUE', 'N/A', np.nan,"NaT",'("null")',"not available"
              "UNKNOWN","U","E","D"]
df = pd.read_csv("NYPD_Complaint_Data_Current__Year_To_Date_.csv",na_va
```

n [3]:
```
# Checking If These Two Columns Are Equal Or Not
df['CMPLNT_FR_TM'].equals(df['CMPLNT_TO_TM'])
```

Out[3]: False



Cleaning Data

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

n [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 256797 entries, 0 to 256796
Data columns (total 36 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   CMPLNT_NUM              256797 non-null  object
 1   ADDR_PCT_CD             256797 non-null  int64
 2   BORO_NM                 256356 non-null  object
 3   CMPLNT_FR_DT            256797 non-null  object
 4   CMPLNT_FR_TM            256797 non-null  object
 5   CMPLNT_TO_DT            237423 non-null  object
 6   CMPLNT_TO_TM            237478 non-null  object
 7   CRM_ATPT_CPTD_CD        256797 non-null  object
 8   HADEVELOPT              815 non-null     object
```

```
 8   HADEVELOPT              815 non-null     object
 9   HOUSING_PSA             16985 non-null   float64
 10  JURISDICTION_CODE       256797 non-null  int64
 11  JURIS_DESC              256797 non-null  object
 12  KY_CD                   256797 non-null  int64
 13  LAW_CAT_CD              256797 non-null  object
 14  LOC_OF_OCCUR_DESC       215731 non-null  object
 15  OFNS_DESC               256791 non-null  object
 16  PARKS_NM                1421 non-null    object
 17  PATROL_BORO             256793 non-null  object
 18  PD_CD                   256528 non-null  float64
 19  PD_DESC                 256528 non-null  object
 20  PREM_TYP_DESC           256690 non-null  object
 21  RPT_DT                  256797 non-null  object
 22  STATION_NAME            5708 non-null    object
 23  SUSP_AGE_GROUP          109037 non-null  object
 24  SUSP_RACE               143020 non-null  object
 25  SUSP_SEX                153472 non-null  object
 26  TRANSIT_DISTRICT        5708 non-null    float64
 27  VIC_AGE_GROUP           184109 non-null  object
 28  VIC_RACE                179145 non-null  object
 29  VIC_SEX                 188130 non-null  object
 30  X_COORD_CD              256797 non-null  int64
 31  Y_COORD_CD              256797 non-null  int64
 32  Latitude                256797 non-null  float64
 33  Longitude               256797 non-null  float64
 34  Lat_Lon                 256797 non-null  object
 35  New Georeferenced Column 256797 non-null  object
dtypes: float64(5), int64(5), object(26)
memory usage: 70.5+ MB
```

n [5]:
```
df
```

Out[5]:

	CMPLNT_NUM	ADDR_PCT_CD	BORO_NM	CMPLNT_FR_DT	CMPLNT_FR_TM	CMPL
0	242955164	79	BROOKLYN	04/22/2021	08:00:00	
1	238863220	25	MANHATTAN	12/31/2021	21:45:00	
2	242870934	46	BRONX	03/16/2018	12:00:00	
3	247126072	44	BRONX	11/10/2019	17:00:00	
4	242181297	84	BROOKLYN	03/16/2000	00:00:00	
...	...	...	...	...	...	
256792	247399166	33	MANHATTAN	06/30/2022	22:10:00	
256793	247392181	113	QUEENS	06/30/2022	19:26:00	
256794	247383348	90	BROOKLYN	06/30/2022	12:36:00	
256795	247434393	109	QUEENS	06/30/2022	14:30:00	
256796	247376531	18	MANHATTAN	06/30/2022	12:57:00	

256797 rows × 36 columns

n [6]:
```
# The Code Below Calculates The Percentage Of Null Values From Every Col
missing_values = df.isnull().mean().round(4).mul(100).sort_values(ascen
missing_values = pd.DataFrame(missing_values)
missing_values
```

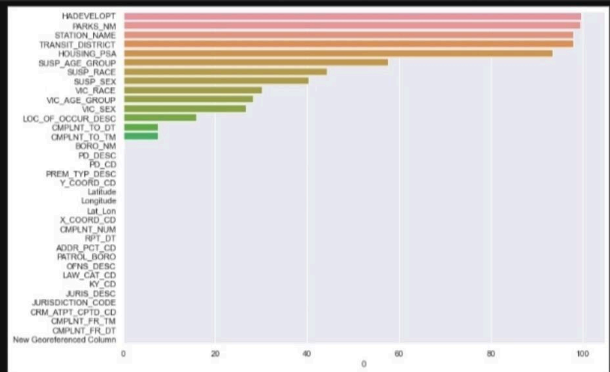
Out[6]:

	0
HADEVELOPT	99.68
PARKS_NM	99.45
STATION_NAME	97.78
TRANSIT_DISTRICT	97.78

TRANSIT_DISTRICT	97.78
HOUSING_PSA	93.39
SUSP_AGE_GROUP	57.54
SUSP_RACE	44.31
SUSP_SEX	40.24
VIC_RACE	30.24
VIC_AGE_GROUP	28.31
VIC_SEX	26.74
LOC_OF_OCCUR_DESC	15.99
CMPLNT_TO_DT	7.54
CMPLNT_TO_TM	7.52
BORO_NM	0.17
PD_DESC	0.10
PD_CD	0.10
PREM_TYP_DESC	0.04
Y_COORD_CD	0.00
Latitude	0.00
Longitude	0.00
Lat_Lon	0.00
X_COORD_CD	0.00
CMPLNT_NUM	0.00
RPT_DT	0.00
ADDR_PCT_CD	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

n [7]:

```
# I Think It Would Be Nice To Plot A Graph Of Null Values Of Every Colum
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.barplot(data=missing_values, y=missing_values.index, x=0);
```



n [8]:

```
# More Than 97% Of Rows Are Null Of This Column, There's No Point To Kee
df["STATION_NAME"].value_counts()
```

Out[8]:

```
125 STREET                       195
34 ST.-PENN STATION              149
42 ST.-PORT AUTHORITY BUS TERM   146
14 STREET                        119
```

[illegible] STREET	[illegible]
59 ST.-COLUMBUS CIRCLE	113
42 ST.-TIMES SQUARE	108
145 STREET	77
168 ST.-WASHINGTON HTS.	73
BROADWAY-EASTERN PKWY	72
FORDHAM ROAD	70
42 ST.-GRAND CENTRAL	69
96 STREET	68
BROADWAY-EAST NEW YORK	65
W. 4 STREET	65
JAY STREET-BOROUGH HALL	63
UTICA AVE.-CROWN HEIGHTS	62
CANAL STREET	57
23 STREET	55
110 ST.-CENTRAL PARK NORTH	52
NOSTRAND AVENUE	51
116 STREET	50
MYRTLE AVENUE	50
KINGSBRIDGE ROAD	50
86 STREET	50
EUCLID AVENUE	49
59 STREET	48
34 ST.-HERALD SQ.	48
149 ST.-GRAND CONCOURSE	48
161 ST.-YANKEE STADIUM	48
3 AVENUE-149 STREET	47
47-50 STS./ROCKEFELLER CTR.	42
CHURCH AVENUE	41
FRANKLIN AVENUE	40
3 AVENUE-138 STREET	39
ATLANTIC AVENUE	39
STILLWELL AVENUE-CONEY ISLAND	38
MAIN ST.-FLUSHING	37
HUNTS POINT AVENUE	36
181 STREET	36
WOODLAWN	35
PARSONS/ARCHER-JAMAICA CENTER	34
72 STREET	33
ROOSEVELT AVE.-JACKSON HEIGHTS	32
HOYT-SCHERMERHORN	32
50 STREET	31
UNION SQUARE	31
103 ST.-CORONA PLAZA	31
PROSPECT AVENUE	30
FULTON STREET	30
170 STREET	30
SUTPHIN BLVD.-ARCHER AVE.	30
EAST 180 STREET	29
34 STREET	29
PACIFIC STREET	29
135 STREET	29
FLATBUSH AVE.-BROOKLYN COLLEGE	29
167 STREET	28
71 AVE.-FOREST HILLS	28
36 STREET	27
ESSEX STREET	27
BURNSIDE AVENUE	27
BROOKLYN BRIDGE-CITY HALL	27
QUEENSBORO PLAZA	26
ROCKAWAY AVENUE	26
LEXINGTON AVE.	26
137 ST.-CITY COLLEGE	26
CHAMBERS ST.-WORLD TRADE CENTE	26
CHAMBERS STREET	25
241 ST.-WAKEFIELD	25
MYRTLE/WYCKOFF AVENUES	23
DEKALB AVENUE	23
BROADWAY/LAFAYETTE	23
EAST BROADWAY	23
JUNCTION BLVD.	23
WEST 34 STREET/HUDSON YARDS	22
179 ST.-JAMAICA	22
MARCY AVENUE	22
42 STREET	22
SIMPSON STREET	22
28 STREET	21
7 AVENUE	20
14 ST.-UNION SQUARE	20
NEW LOTS AVENUE	19
FLUSHING AVENUE	19
QUEENS PLAZA	19
PROSPECT PARK	19
KINGS HIGHWAY	18
191 STREET	18

```
Name: STATION_NAME, dtype: int64
```

In [9]:
```
# 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 Uni
df['CMPLNT_TO_DT'].equals(df["CMPLNT_FR_DT"])
```

Out[9]: False

In [10]:
```
# These Are Mostly Null Columns Or Their Rows Are So Less It's Not Worth
# 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 [11]:
```
# Checking For Duplicates
df[df.duplicated()]
# There Are No Duplicate Rows
```

Out[11]:

CMPLNT_NUM	ADDR_PCT_CD	BORO_NM	CMPLNT_FR_DT	CMPLNT_FR_TM	CMPLNT_TO_DT

In [12]:
```
# 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
# So We Will Analyse Each Column Carefully From Now On.

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

Out[12]:
```
SUSP_AGE_GROUP              147760
SUSP_RACE                   113777
SUSP_SEX                    103325
VIC_RACE                     77652
VIC_AGE_GROUP                72688
VIC_SEX                      68667
LOC_OF_OCCUR_DESC            41066
CMPLNT_TO_DT                 19374
CMPLNT_TO_TM                 19319
BORO_NM                        441
PD_DESC                        269
PD_CD                          269
PREM_TYP_DESC                  107
OFNS_DESC                        6
PATROL_BORO                      4
Longitude                        0
Latitude                         0
Lat_Lon                          0
CMPLNT_NUM                       0
RPT_DT                           0
ADDR_PCT_CD                      0
LAW_CAT_CD                       0
KY_CD                            0
JURIS_DESC                       0
CRM_ATPT_CPTD_CD                 0
CMPLNT_FR_TM                     0
CMPLNT_FR_DT                     0
New Georeferenced Column         0
dtype: int64
```

In [13]:
```
# The Code Below Calculates The Percentage Of Unique Values From Every C
unique  = df.nunique().sort_values(ascending=False) /256797 *100
unique = pd.DataFrame(unique)
unique
```

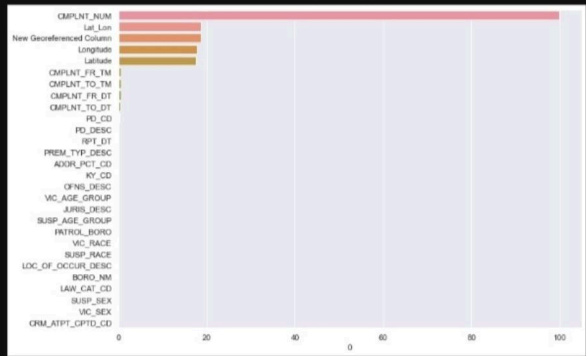
Out[13]:

	0
CMPLNT_NUM	99.974688
Lat_Lon	18.647025
New Georeferenced Column	18.647025
Longitude	17.724117
Latitude	17.634162
CMPLNT_FR_TM	0.560754
CMPLNT_TO_TM	0.560754
CMPLNT_FR_DT	0.559197
CMPLNT_TO_DT	0.383961
PD_CD	0.130843

PD_DESC	0.126949
RPT_DT	0.070484
PREM_TYP_DESC	0.032711
ADDR_PCT_CD	0.029985
KY_CD	0.023754
OFNS_DESC	0.022975
VIC_AGE_GROUP	0.007009
JURIS_DESC	0.006620
SUSP_AGE_GROUP	0.005062
PATROL_BORO	0.003115
VIC_RACE	0.002336
SUSP_RACE	0.002336
LOC_OF_OCCUR_DESC	0.001947
BORO_NM	0.001947
LAW_CAT_CD	0.001168
SUSP_SEX	0.000779
VIC_SEX	0.000779
CRM_ATPT_CPTD_CD	0.000779

In [14]:

```
# 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:-

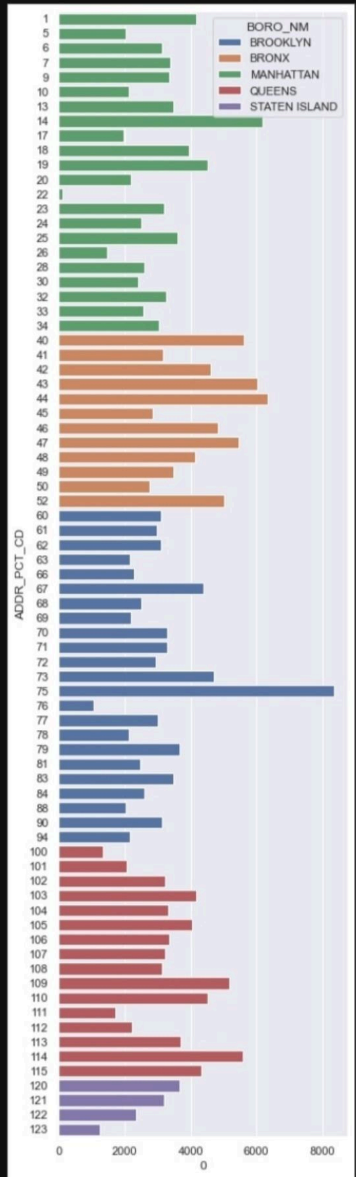
> - Brooklyn Contains The Highest Share Of Crime In New York City At 28.90%
> - The Lowest Would Be Staten Island At 4.09%
> - The 75th Precint Of Brooklyn Has The Highest Count Of Crime At 8834

In [15]:
```
# Plotting A Treemap Using Sqaurify
# As I Have Mentioned Above We Have To Drop Nulls For Every Column "i
# The Colour Of This Tree Map Will Change Randomly Every Time You Run

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 i
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 [16]:
```
# 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 =
sns.set(rc={'figure.figsize':(5,20)})
```

```
sns.set(rc={'figure.figsize':(5,20)})
sns.barplot(data=precint, y="ADDR_PCT_CD",x=0, orient="h", hue ='BORO_
```



In [17]:

```
# Converting These Columns To DateTime
df["CMPLNT_FR_DT"] = pd.to_datetime(df["CMPLNT_FR_DT"], errors = 'coe
df["CMPLNT_TO_DT"] = pd.to_datetime(df["CMPLNT_TO_DT"], errors = 'coe
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 R
# You Will See At The End Of The Notebook There's A Beautiful Map Of

df['VIC_AGE_GROUP']= df['VIC_AGE_GROUP'].replace(['-11','-4',
                                                   '-40','-5',
```

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

Time Series Analysis

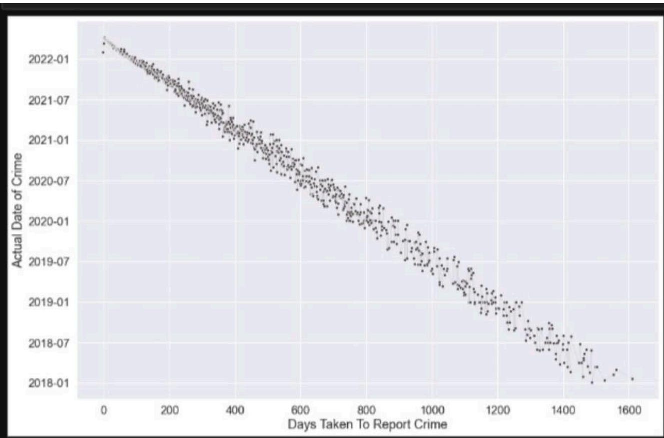
> A Time Series Is A Series Of Data Points Indexed (or Listed Or Graphed) In Time Order. Most Commonly, A Time Series Is A Sequence Taken At Successive Equally Spaced Points In Time. A Time Series Is Very Frequently Plotted Via A Run Chart (which Is A Temporal Line Chart). Time Series Are Used In Statistics, 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 [18]:

```
# Plotting A Lineplot Type Graph That Tells The Time Difference Betwe
# This Variable X_86 Has An Interesting Story, So Numpy Was Returning
# But I Wanted Time Difference In Days, So If You Divide "nanosecond.
#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"] <
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 C
plt.show()
```



In [19]:

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

Out[19]:

```
CMPLNT_NUM                                    242870934
ADDR_PCT_CD                                          46
BORO_NM                                           BRONX
CMPLNT_FR_DT                        2018-03-16 00:00:00
CMPLNT_FR_TM                        2022-08-18 12:00:00
CMPLNT_TO_DT                        2022-03-30 00:00:00
CMPLNT_TO_TM                        2022-08-18 22:57:00
CRM_ATPT_CPTD_CD                              COMPLETED
JURIS_DESC                              N.Y. POLICE DEPT
KY_CD                                               112
LAW_CAT_CD                                       FELONY
LOC_OF_OCCUR_DESC                                INSIDE
OFNS_DESC                                   THEFT-FRAUD
PATROL_BORO                           PATROL BORO BRONX
PD_CD                                             739.0
PD_DESC                        FRAUD,UNCLASSIFIED-FELONY
PREM_TYP_DESC                     RESIDENCE - APT. HOUSE
RPT_DT                              2022-03-30 00:00:00
SUSP_AGE_GROUP                                      NaN
SUSP_RACE                                           NaN
SUSP_SEX                                            NaN
VIC_AGE_GROUP                                     25-44
VIC_RACE                                 WHITE HISPANIC
VIC_SEX                                               F
Latitude                                      40.846629
Longitude                                     -73.91605
Lat_Lon                            (40.846629, -73.91605)
New Georeferenced Column    POINT (-73.91605 40.846629)
Name: 2, dtype: object
```

In [20]:

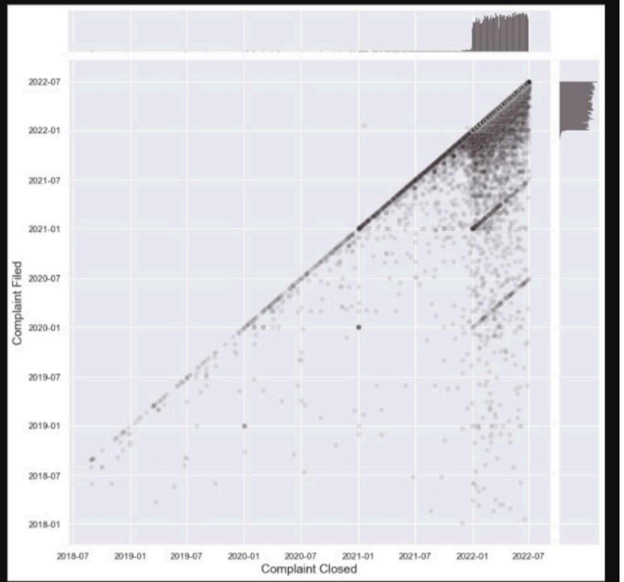
```
# Plotting A Joint Grid That Tells When A Complaint Is Filed On Y-axi
# And When A Complaint Closed On The X-axis

pw = df[(df["CMPLNT_FR_DT"] > "2018-01-01") & (df["CMPLNT_FR_DT"] < ":
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=1
```

Out[20]:

```
<seaborn.axisgrid.JointGrid at 0x17e81164ee0>
```

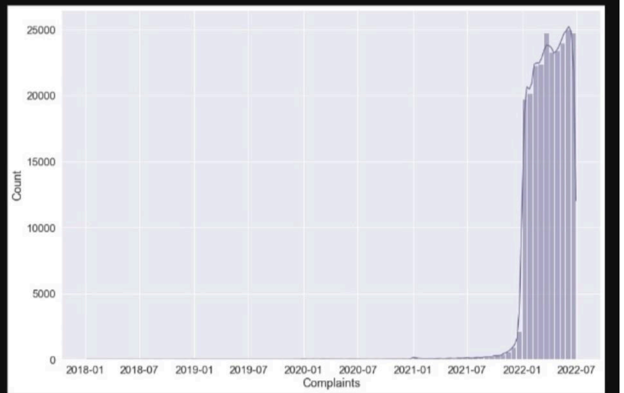
ut[20]: <seaborn.axisgrid.JointGrid at 0x17e81164ee0>



In [21]:

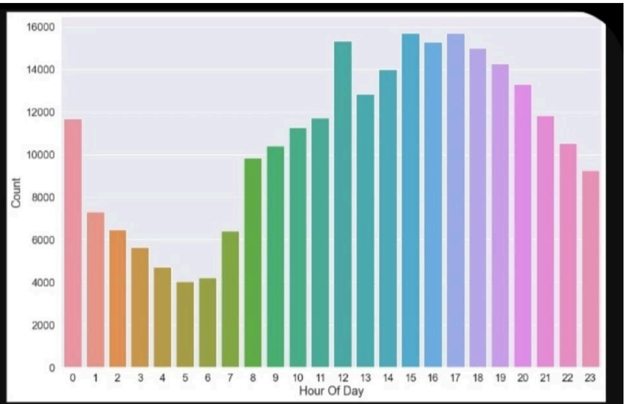
```python
# well, self-explanatory.
sns.set(rc={'figure.figsize':(15,10)})
sns.set(font_scale=1.5)
ct=sns.histplot(data=pw,x="CMPLNT_FR_DT",bins=90 ,color="#6e6799",mul
ct.set(xlabel='Complaints')
```

Out[21]: [Text(0.5, 0, 'Complaints')]



In [22]:

```python
# # 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:-

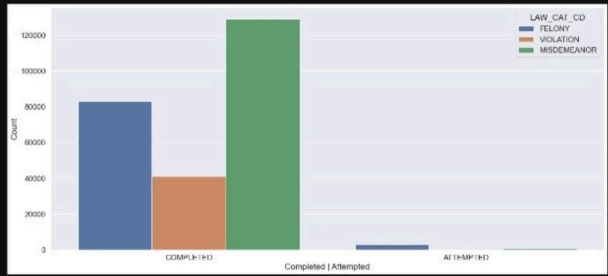
> - Most Common Law Classification Is A Misdemeanour
> - surprising People Commit More Felonies Than Violation
> - NYPD Report Crimes Far-far Above Than Any Other Policing Authority In NYC, Next Is The NY Housing Police
> - Most Crimes Occur Inside Of A Structure, Contrary To Popular Belief Of Crime 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 [23]:
```
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[23]:
```
[Text(0.5, 0, 'Completed | Attempted'), Text(0, 0.5, 'Count')]
```

In [23]:
```python
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')
```

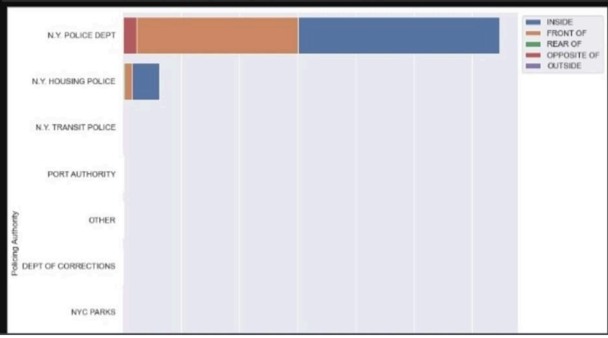
Out[23]:
```
[Text(0.5, 0, 'Completed | Attempted'), Text(0, 0.5, 'Count')]
```



In [24]:
```python
# People Don't Commit Violations, I Guess
df[(df["LAW_CAT_CD"]=="VIOLATION") & (df["CRM_ATPT_CPTD_CD"]=="ATTEMI
```

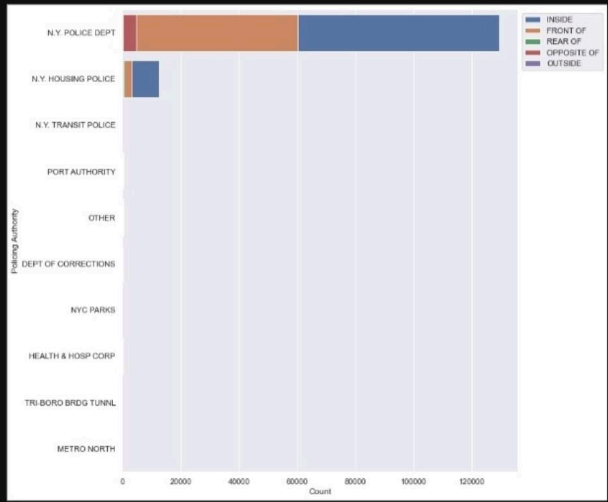
Out[24]:
```
CMPLNT_NUM                  87
ADDR_PCT_CD                 87
BORO_NM                     87
CMPLNT_FR_DT                87
CMPLNT_FR_TM                87
CMPLNT_TO_DT                81
CMPLNT_TO_TM                81
CRM_ATPT_CPTD_CD            87
JURIS_DESC                  87
KY_CD                       87
LAW_CAT_CD                  87
LOC_OF_OCCUR_DESC           73
OFNS_DESC                   87
PATROL_BORO                 87
PD_CD                       87
PD_DESC                     87
PREM_TYP_DESC               87
RPT_DT                      87
SUSP_AGE_GROUP              36
SUSP_RACE                   59
SUSP_SEX                    67
VIC_AGE_GROUP               87
VIC_RACE                    72
VIC_SEX                     84
Latitude                    87
Longitude                   87
Lat_Lon                     87
New Georeferenced Column    87
dtype: int64
```

In [25]:
```python
# Plotting A Countplot, Policing Authority On The Y-axis, Count On Th
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 [25]:

```
# Plotting A Countplot, Policing Authority On The Y-axis, Count On Th
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 [26]:

```
# Plotting Heatmap Of Crime Committed According To Public Description
# See How Police Are So Articulate When Writing Descriptions Of A Cri
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")
```

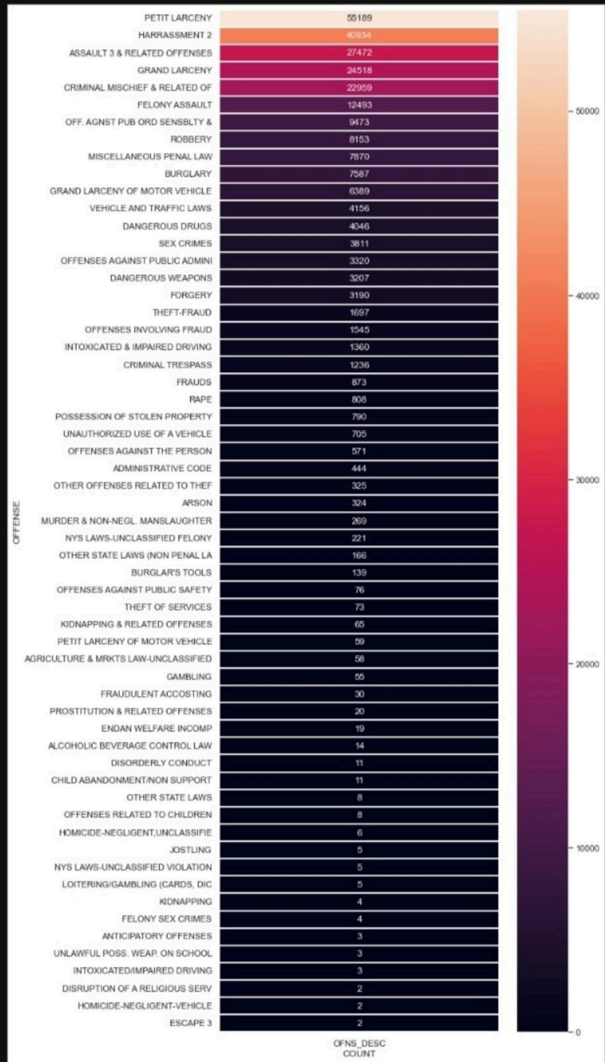
Out[26]: [Text(0.5, 192.5, 'COUNT'), Text(48.499999999999986, 0.5, 'OFFENSE')]

. [26]:

```
# Plotting Heatmap Of Crime Committed According To Public Description
# See How Police Are So Articulate When Writing Descriptions Of A Cri
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")
```

t[26]:

```
[Text(0.5, 192.5, 'COUNT'), Text(48.499999999999986, 0.5, 'OFFENSE')]
```

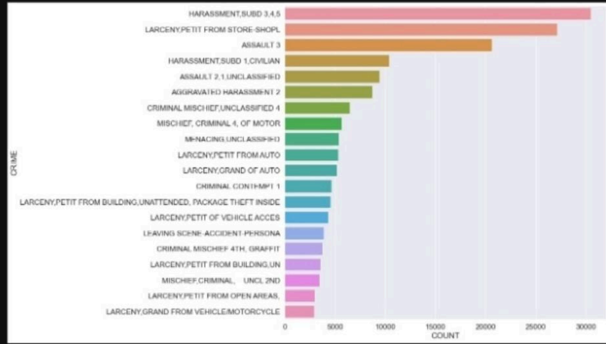


n [27]:

```
# Plotting Top 20 Crimes According To Police,
# See How These Are Articulated Comparing Them To The Public Descript
pdd = df.PD_DESC.value_counts()
pdd = pd.DataFrame(pdd).head(20)
sns.set(rc={'figure.figsize':(12,12)})
```

[27]:

```python
# Plotting Top 20 Crimes According To Police,
# See How These Are Articulated Comparing Them To The Public Descript
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

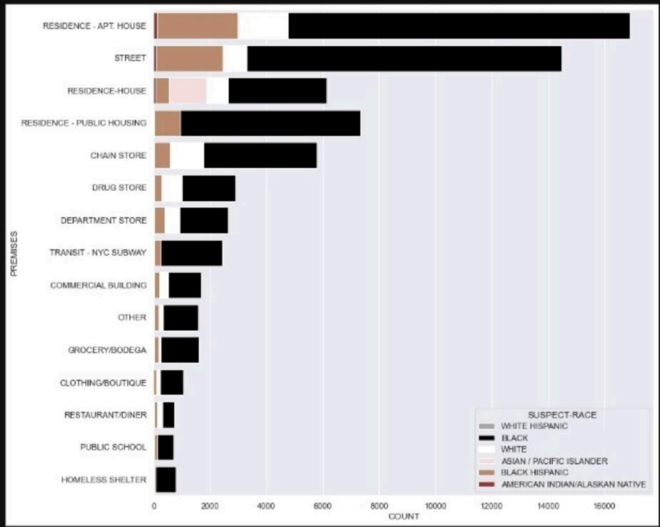
In [28]:

```python
# 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"
```

In [28]:

```python
# 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].
              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[28]: <matplotlib.legend.Legend at 0x17e812a5820>



In [29]:

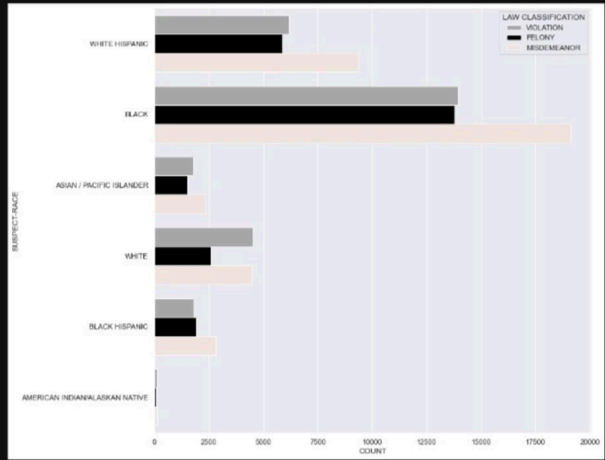
```python
vic=df[['VIC_SEX','SUSP_RACE',"LAW_CAT_CD", "VIC_RACE","SUSP_SEX"]]
vic.dropna(inplace=True)
sns.countplot(data=vic, x="VIC_SEX", hue="SUSP_RACE",
              palette=(sns.color_palette(["#A9A9A9","#000000","#F5E4D(
                                          "#FFFFFF","#C68863","#A52A2/
plt.legend(title="SUSPECT-RACE")
```

Out[29]: <matplotlib.legend.Legend at 0x17e82c93940>

In [30]:

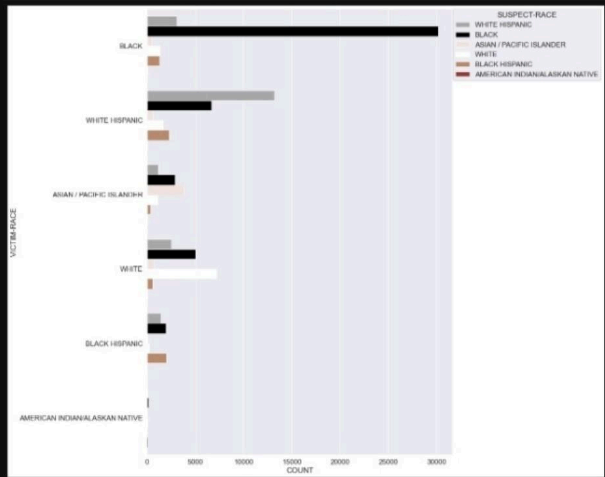
```python
lc = sns.countplot(data=vic,
              y="SUSP_RACE",
              hue="LAW_CAT_CD",
              palette=(sns.color_palette(["#A9A9A9","#000000","#F5E4D(
plt.legend(title="LAW CLASSIFICATION")
lc.set(xlabel="COUNT",ylabel="SUSPECT-RACE")
```

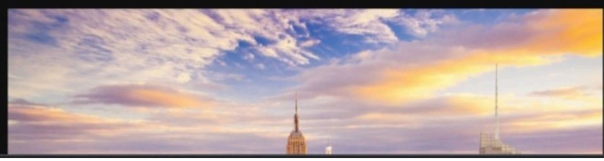
Out[30]: [Text(0.5, 0, 'COUNT'), Text(0, 0.5, 'SUSPECT-RACE')]



In [31]:

```python
# 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","#F5E4D(
                                          "#FFFFFF","#C68863","#A52A2/
vic_sus.set(xlabel="COUNT",ylabel="VICTIM-RACE");
plt.legend(title="SUSPECT-RACE" ,bbox_to_anchor=(1, 1), loc=2, border:
```



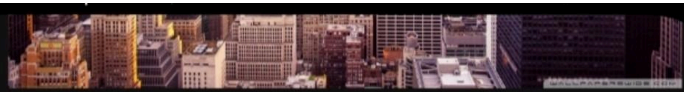


In [32]:

```
# 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()
```

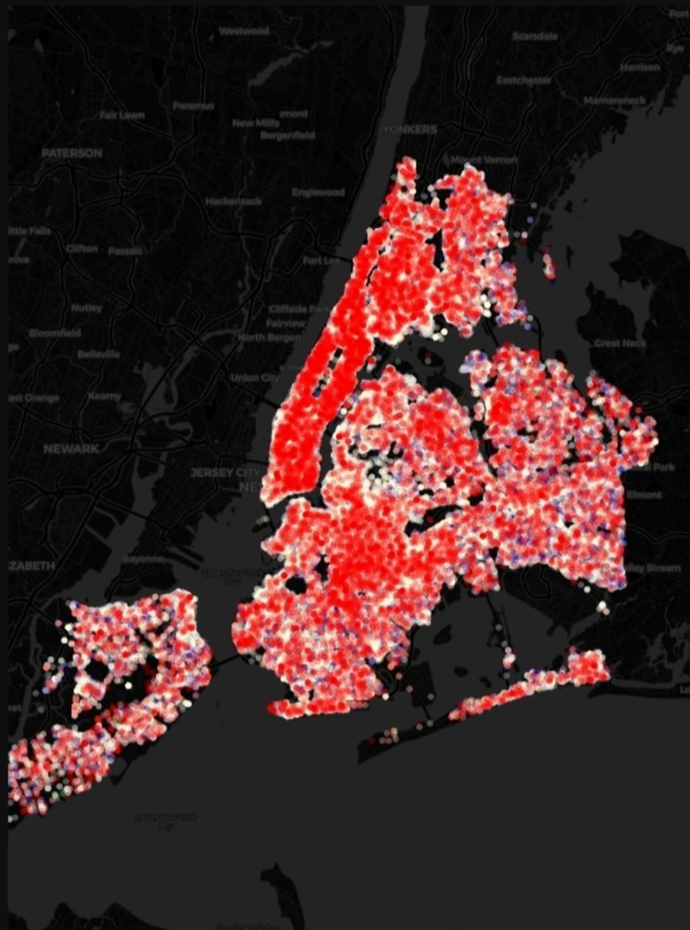


In [32]:

```
# 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()
```



In [ ]: