Housing Maintenance Violation Code Analysis

By Amy Chen, Jeffrey Liu, Claudia Ye

This is an analysis of the data reported by the Department of Housing Preservation and Development (HPD). HPD issues violations to rental dwelling units that have violated either Housing Maintenance Code or the New York State Multiple Dwelling Law.

Tenants can directly consult their landlord or file an official complaint if an issue is discovered in their apartment. These complaints are directed to the HPD, who in turn contacts the building's managing agent informing them of the complaint. The HPD will then follow up the complaint to ensure that the issue has been resolved and thus would close complaint. If the issue continues to be unresolved, the HPD would send a Code Inspector to check for housing or safety violations. The complaints are categorized into either class A, B, C, and I, with class A being least sever and class C being most severe, such as problems with heating and hot water (Class I cases are unique and indicate a serious hazard). The violations persist on file and the complaint cases remain open until the HPD can confirm that the owner has sufficiently corrected the condition.

Set Up

Downloading Packages

In [1]:

```
import pandas as pd
import geopandas as gpd
import matplotlib
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import datetime
```

In [2]:

```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
matplotlib.style.use(['seaborn-talk', 'seaborn-ticks', 'seaborn-whitegrid'])
```

Retrieving Data

Here is a look at the housing violation data. As of when the file was downloaded, there are 4,955,054 housing violations and the dataset is continuously getting updated. We will take a sample of that dataset and work with that data

In [9]:

```
#housing = pd.read_csv('Housing_Maintenance_Code_Violations.csv')
housing = pd.read_csv('sample_housing.csv')
```

```
/Users/claudia/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2785:
DtypeWarning: Columns (28,29,31) have mixed types. Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [10]:

```
sample = housing.sample(frac=0.9)
sample.to_csv('sample_housing.csv')
```

In [11]:

```
#housing = housing.drop(['Unnamed: 0', 'Unnamed: 0.1', 'Unnamed: 0.1.1', 'Unnamed:
0.1.1.1','Unnamed: 0.1.1.2'], axis=1)
```

Our sample consists of 157,434 violations (which is approximately 3.1% of the complete dataset)

In [12]:

```
len(housing)
```

Out[12]:

```
127522
```

In [13]:

```
housing.columns
```

Out[13]:

```
Index(['Unnamed: 0', 'Unnamed: 0.1', 'Unnamed: 0.1.1', 'Unnamed: 0.1.1.1',
       'Unnamed: 0.1.1.1.1', 'Unnamed: 0.1.1.1.1.1', 'Unnamed: 0.1.1.2',
       'ViolationID', 'BuildingID', 'RegistrationID', 'BoroID', 'Borough',
       'HouseNumber', 'LowHouseNumber', 'HighHouseNumber', 'StreetName',
       'StreetCode', 'Postcode', 'Apartment', 'Story', 'Block', 'Lot', 'Class',
       'InspectionDate', 'ApprovedDate', 'OriginalCertifyByDate',
       'OriginalCorrectByDate', 'NewCertifyByDate', 'NewCorrectByDate',
       'CertifiedDate', 'OrderNumber', 'NOVID', 'NOVDescription',
       'NOVIssuedDate', 'CurrentStatusID', 'CurrentStatus',
       'CurrentStatusDate', 'NovType', 'ViolationStatus', 'Latitude',
       'Longitude', 'CommunityBoard', 'CouncilDistrict', 'CensusTract', 'BIN',
       'BBL', 'NTA'],
      dtype='object')
```

In [14]:

```
housing
```

Out[14]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Unnamed: 0.1.1.1.1.1	Unnamed: 0.1.1.2	ViolationID	BuildingID	Registratio
0	73268	36347	17661	232856	347837	2189550	2189550	10030393	355783	340516
1	137952	25037	56077	206447	262704	4531680	4531680	12476430	52532	217021
2	67483	84736	128296	266062	84345	3136794	3136794	6341165	784320	0
3	11065	73782	82409	60760	102037	3471871	3471871	9070188	866183	379416
4	28215	113791	172	65818	324952	3616925	3616925	9510434	164978	368528
5	91214	156974	127337	171439	293901	1960736	1960736	872794	119284	203406
6	7503	7486	223315	63457	86977	1563314	1563314	11868349	107776	215685
7	139695	54155	207867	159084	113536	1207402	1207402	11488849	119069	200823
8	68320	9357	195666	283481	236901	1049424	1049424	11323370	109445	204361
9	110497	156328	109894	268729	210612	4784818	4784818	12268346	357887	330679
10	102530	113464	105549	86216	229649	4428162	4428162	9051338	347134	312957

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Unnamed: 0.1.1.1.1.1	Unnamed: 0.1.1.2	7135009 ViolationID	230006 BuildingID	357812 Registratio
12	9600	94279	158196	247914	14813	2826811	2826811	4503642	76141	219406
13	1821	108271	220087	201667	175601	2597627	2597627	2511536	13199	136203
14	59051	82601	71795	77665	20856	4948845	4948845	12205606	9195	108725
15	94168	43438	144437	181017	176022	1296781	1296781	12424961	18981	119780
16	82157	23475	117563	282493	111928	3190	3190	10011180	209858	366042
17	79913	100670	88777	271313	169792	4172164	4172164	12188773	49352	211090
18	119069	41110	56800	133520	189058	3195887	3195887	12533657	135073	323317
19	37494	131344	214555	293020	253045	2721626	2721626	3450868	639858	0
20	111444	122373	97176	303710	223762	2621739	2621739	2670209	24410	124980
21	42239	16484	56385	190800	28158	1452309	1452309	12687947	144356	336404
22	46817	112740	85435	112603	260029	4021455	4021455	10299544	145628	338109
23	131712	57012	184407	134997	315648	1519503	1519503	11818032	42324	101207
24	122577	49339	174832	72260	133555	2923404	2923404	2251828	396242	368566
25	46557	139962	157665	82458	213597	1854207	1854207	7338606	505305	914666
26	132024	104834	42631	46769	351744	516139	516139	10737660	217335	338345
27	75368	81826	141458	305475	24867	85328	85328	10240258	357290	344157
28	37678	130981	24923	152591	213268	1667196	1667196	2786013	3175	132981
29	126841	66114	27675	117386	45065	2695723	2695723	3194918	42452	116308
...	...	...	...	...	...	...	...	...	...	...
127492	99718	26906	18770	260763	110302	1129618	1129618	11407661	342014	338874
127493	78344	18366	210872	78690	192610	1532966	1532966	11831983	287557	327163
127494	22306	67806	73158	176254	289228	3415152	3415152	8824071	277551	0
127495	60535	20087	53200	58182	280408	4513526	4513526	12029888	41115	104317
127496	91457	71050	194151	278227	343212	1615304	1615304	1685306	327477	329087
127497	125288	36190	129469	161590	41633	3155319	3155319	7248738	531509	0

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Unnamed: 0.1.1.1.1.1	Unnamed: 0.1.1.2	ViolationID	BuildingID	Registratio
127498	20399	1956	57584	207343	249119	929478	929478	11197176	62821	200794
127499	135037	92152	69349	262762	30403	1917769	1917769	8243808	338075	350294
127500	58961	21000	214449	75726	243383	1363488	1363488	11652109	8024	108839
127501	130532	152496	133450	98995	248878	4245997	4245997	12131208	346496	300401
127502	70765	92928	172375	152645	49529	2188901	2188901	10029313	89745	211139
127503	110235	33886	54853	206287	36552	4875972	4875972	9968816	19561	114044
127504	45535	71719	119553	52528	315554	3671643	3671643	9593317	62702	202748
127505	21286	81125	148844	100216	353162	212063	212063	12541435	100943	222230
127506	113575	25537	217290	306004	213830	3227000	3227000	7760674	1641	107410
127507	78988	86464	22156	265117	257753	4681863	4681863	12077467	145678	342141
127508	58046	133338	146141	300179	132536	2423316	2423316	139165	417099	411956
127509	24180	9796	50014	132152	98640	3717553	3717553	9654698	222337	356017
127510	133670	19723	179686	34522	354975	2126204	2126204	983288	136078	353138
127511	110187	5527	91316	309515	173745	4792676	4792676	9049630	21306	107514
127512	129006	59798	5536	121213	157541	2983158	2983158	577250	73929	211030
127513	71875	146482	74803	235803	15051	4899054	4899054	12516322	287672	335232
127514	18299	38903	12256	44584	151937	1836843	1836843	7042301	303054	342778
127515	39818	81833	151964	196279	21225	270055	270055	10469268	114469	201083
127516	87365	131086	168206	62155	152682	414055	414055	10624910	27343	128637
127517	86700	69957	8540	22023	291041	3315924	3315924	8322328	628926	406870
127518	117705	116573	211667	135386	347007	4515796	4515796	3997947	803948	136234
127519	93411	16977	121946	63542	81432	2963341	2963341	5617066	183108	332130
127520	133480	48707	84838	176715	152208	3770329	3770329	9724185	808394	374406
127521	30865	56564	167454	219704	320967	1922392	1922392	8299554	102634	221034

127522 rows × 47 columns

There are many dates that are being used in this dataset. We will convert the inspection date and approval date into datetime since they are currently strings. By converting them, we will create a new column called "Time Until Approval" to see how long it takes the Department of Building to approve. Within this dataset, 65% of the violations are still open.

To find the percentage of violations that are still open, we create a variable grouped by the column "ViolationStatus", and then count the amount of open and closed violations. Then we divide the number of open violations by the total.

In [15]:

```
open_close = housing.groupby('ViolationStatus').count()
open_close['ViolationID'].iloc[1] / open_close['ViolationID'].iloc[0]
#Shouldn't it be divided by the total
```

Out[15]:

```
0.6526314425307466
```

Here we are converting inspection and approval date into datetime.

In [20]:

```
housing['InspectionDate'] = pd.to_datetime(housing['InspectionDate'], format="%m/%d/%Y", errors = '
coerce')
housing['ApprovedDate'] = pd.to_datetime(housing['ApprovedDate'], format="%m/%d/%Y", errors = 'coer
ce')
```

In [21]:

```
housing['Time_Until_Approval'] = (abs(housing['ApprovedDate'] - housing['InspectionDate']))
housing['Time_Until_Approval'] = (housing['Time_Until_Approval'] / np.timedelta64(1, 'D')).fillna(0
).astype(int)
```

We will look at data starting from the 1980s since there aren't many data points in the file for anything before then. Within the last 30-40 years, there seems to be an exponential growth in housing violations. The graph that is displayed shows the change over time in years.

In [22]:

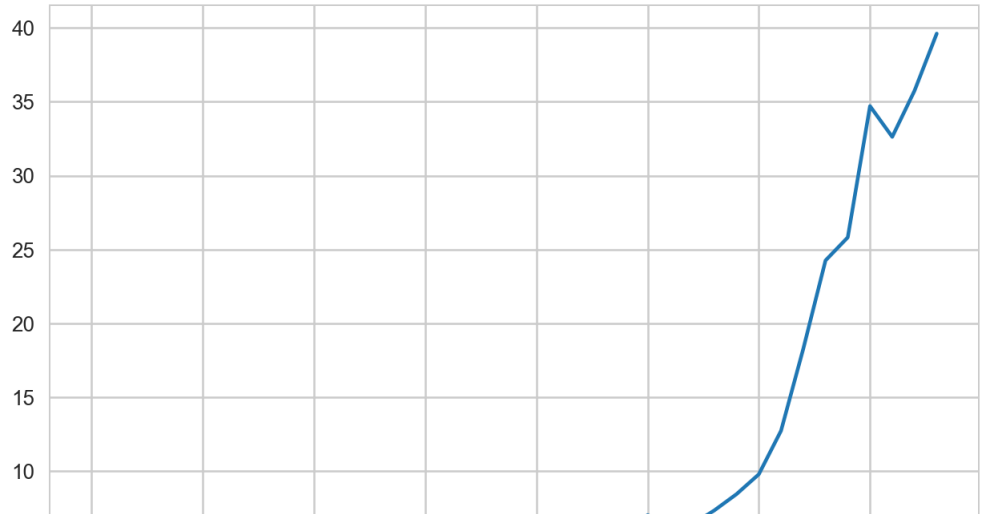
```
housing_past_1980 = housing[(housing.InspectionDate > '1980-01-01')]
```

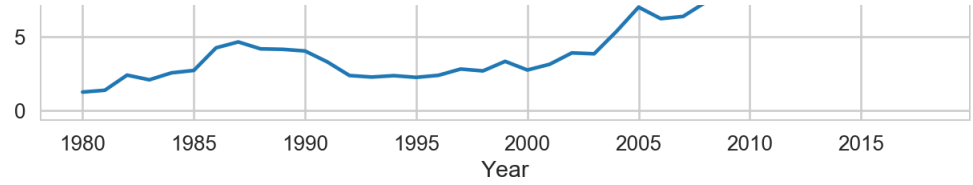
In [23]:

```
matplotlib.pyplot.xlabel("Year")
housing_past_1980.InspectionDate.value_counts().sort_index().resample('AS').mean().plot()
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a171d2c18>
```





We will now look at how many unique violations there are. The order number references to the abstract description of the violation condition which cites a specific section of the law which is in violation. From the code below, we find that there are 389 unique violation. That's 4955054:389 individual violations to violation code. 508 is the most popular order number.

1. **508: REPAIR THE BROKEN OR DEFECTIVE PLASTERED SURFACES AND PAINT IN A UNIFORM COLOR**
2. **501: REPAIR THE BROKEN OR DEFECTIVE**
3. **502: PROPERLY REPAIR WITH SIMILAR MATERIAL THE BROKEN OR DEFECTIVE**
4. **780: OWNER FAILED TO FILE A VALID REGISTRATION STATEMENT WITH THE DEPARTMENT AS REQUIRED BY ADM CODE §27-2097 AND IS THEREFORE SUBJECT TO CIVIL PENALTIES, PROHIBITED FROM CERTIFYING VIOLATIONS, AND DENIED THE RIGHT TO RECOVER POSSESSION OF PREMISES FOR NONPAYMENT OF RENT UNTIL A VALID REGISTRATION STATEMENT IS FILED.**
5. **556: PAINT WITH LIGHT COLORED PAINT TO THE SATISFACTION OF THIS DEPARTMENT**
6. **702: REPAIR OR REPLACE THE SMOKE DETECTOR**
7. **510: 309 M/D LAW ABATE THE NUISANCE CONSISTING OF**
8. **505: REPLACE WITH NEW THE BROKEN OR DEFECTIVE**
9. **583: PROPERLY REPAIR THE SOURCE AND ABATE THE EVIDENCE OF A WATER LEAK**
10. **550: TRACE AND REPAIR THE SOURCE AND ABATE THE NUISANCE CONSISTING OF MOLD ...**

To find the number of unique orders, we converted the order in the column "OrderNumber" into a string.

In [20]:

```
housing['OrderNumber'] = [str(order) for order in housing['OrderNumber']]
print(housing['OrderNumber'].nunique())
```

```
298
```

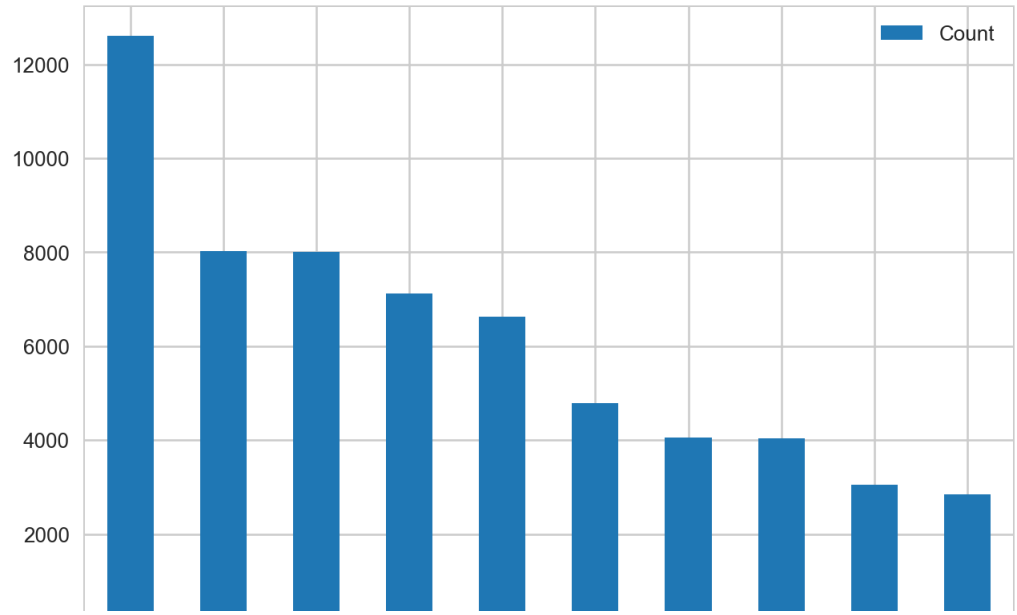
This graph displays the ten most popular violations by their order numbers.

In [21]:

```
orderNumberViolation = housing[['ViolationID',
'OrderNumber']].groupby('OrderNumber').count().rename(columns={"ViolationID":"Count"})
orderNumberViolation.sort_values('Count', ascending=False).head(10).plot(kind='bar')
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a214047f0>
```



We will now look at the number of violations in each borough. Here, Brooklyn has the most violation. The violations are also plotted on the NYC map which I got from geojson. Data without a latitude and longitude are dropped since they were causing errors. The lighter the purple, the less concentration of violation.

In [22]:

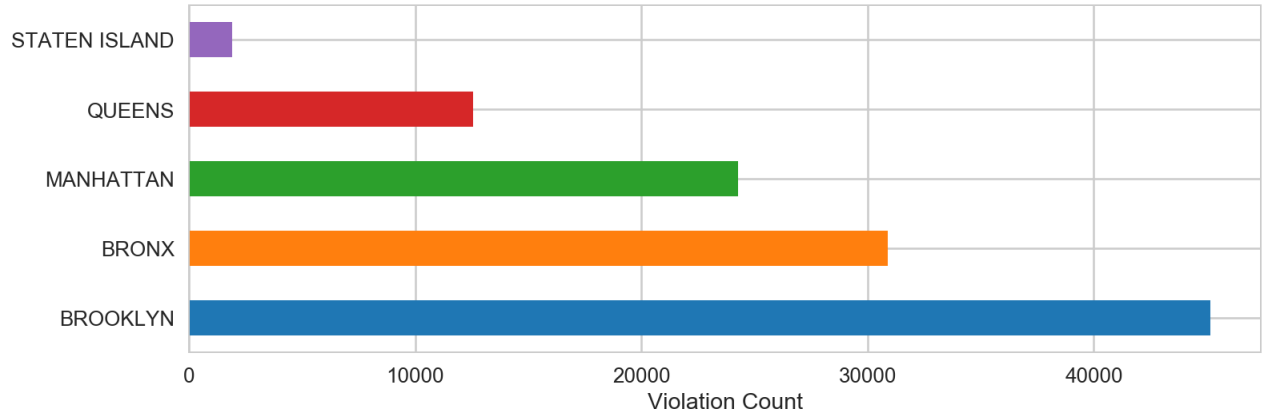
```
housing = housing.dropna(subset=['Latitude', 'Longitude'])
```

In [16]:

```
matplotlib.pyplot.xlabel("Violation Count")
housing.Borough.value_counts().plot(kind='barh', figsize=(12,4))
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a14a48390>
```



In [24]:

```
!curl 'https://data.cityofnewyork.us/api/geospatial/cpf4-rkhq?method=export&format=GeoJSON' -o nyc
-neighborhoods.geojson
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 4067k    0 4067k    0     0  2895k      0 --:--:--  0:00:01 --:--:-- 2895k
```

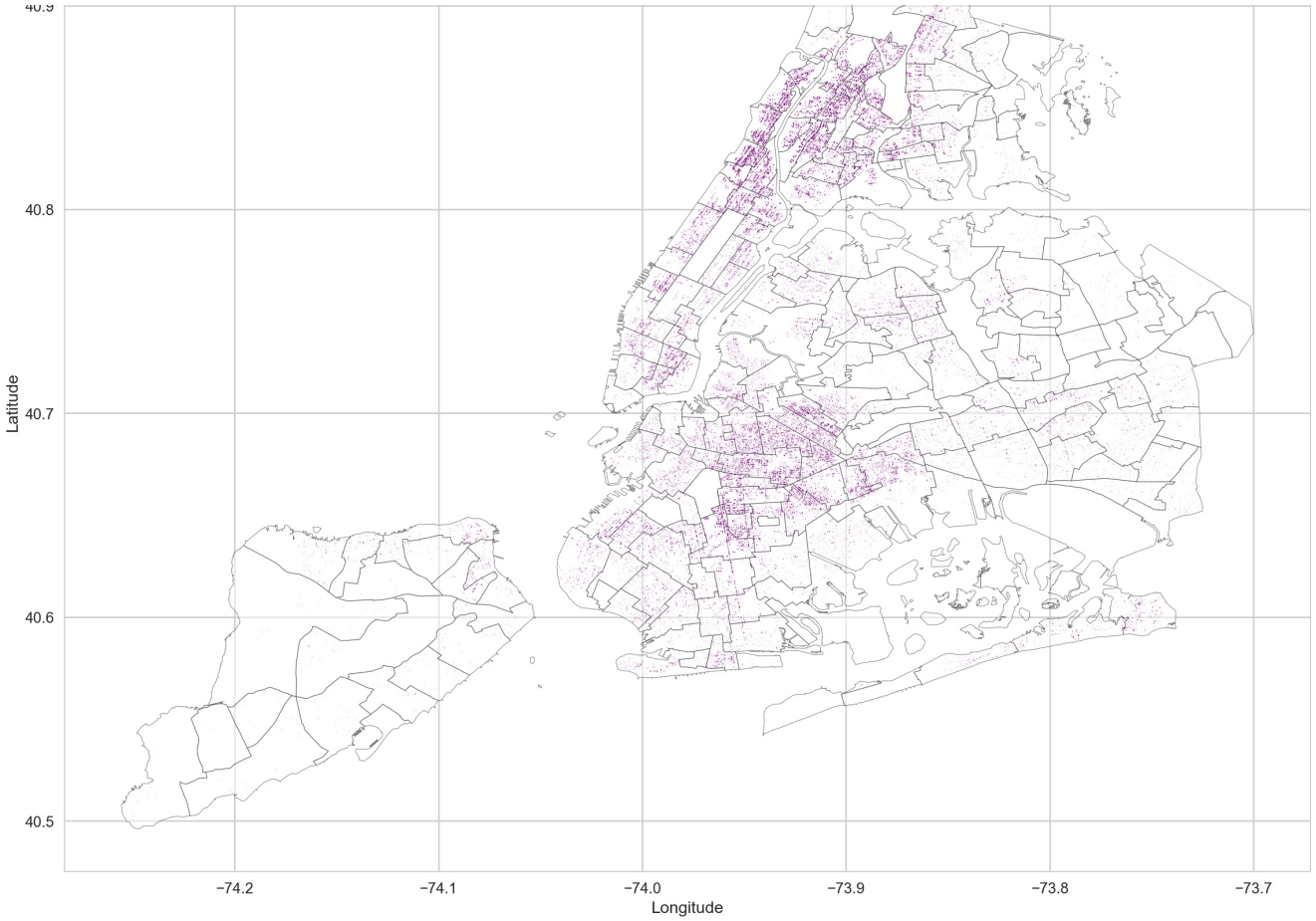
In [25]:

```
df_nyc = gpd.GeoDataFrame.from_file('nyc-neighborhoods.geojson')
```

In [26]:

```
ny_map = df_nyc.plot(linewidth=0.5, color='White', edgecolor = 'Black', figsize = (20,15), alpha=0.
5)

housing_loc = housing.sample(frac=0.7).plot.scatter(
    x="Longitude",
    y="Latitude",
    figsize=(20,15),
    s=0.3,
    color='purple',
    alpha=0.1,
    ax=ny_map
)
```



The following graph below takes 10% of the sample data and plots a KDE map from it. The map may be a bit inaccurate, but generally speaking, it should be okay because higher concentrations of violation will still be mapped out.

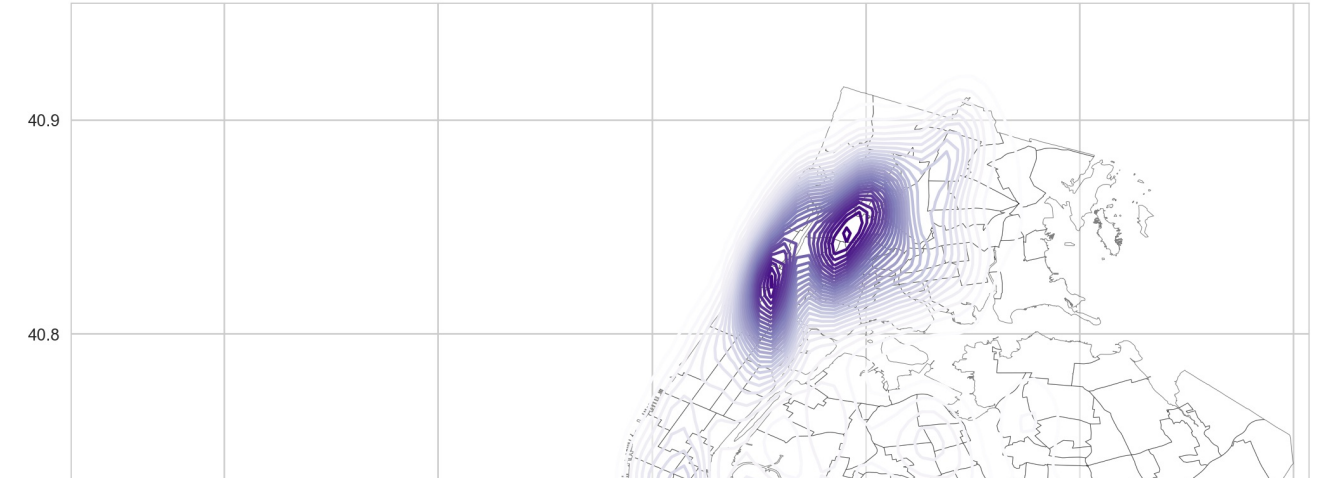
In [27]:

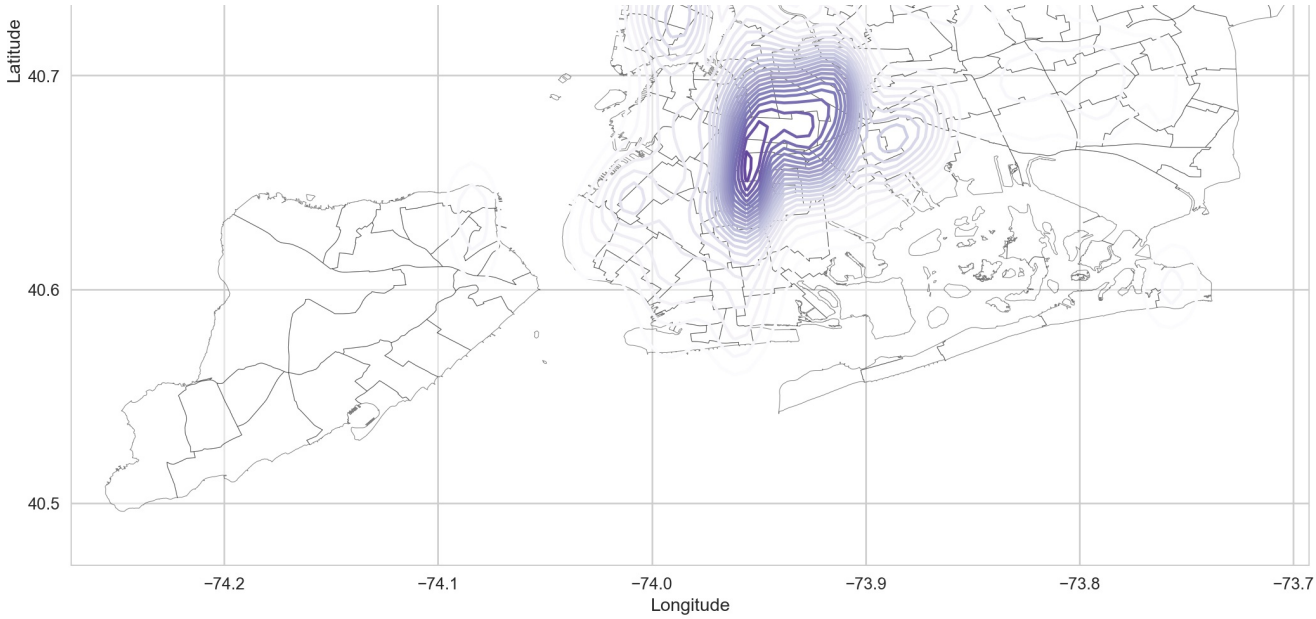
```
ny_map = df_nyc.plot(linewidth=0.5, color='White',edgecolor = 'Black', figsize = (20,15), alpha=0.5
)

sample = housing.sample(frac=0.1)
sns.kdeplot(
    sample.Longitude, sample.Latitude,
    gridsize=100,
    cmap=plt.cm.Purples,
    shade_lowest=False,
    n_levels=30,
    ax=ny_map
)
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a254f7d68>
```





The following graph represents the time it takes for the department of building to approve a violation depending on the class of the violation.

***A* class violation is the least severe and the *C* class is the most severe. *I* class is considered hazardous.**

We will disregard Staten Island and the Bronx here because of they have less cases of buildings with housing violations.

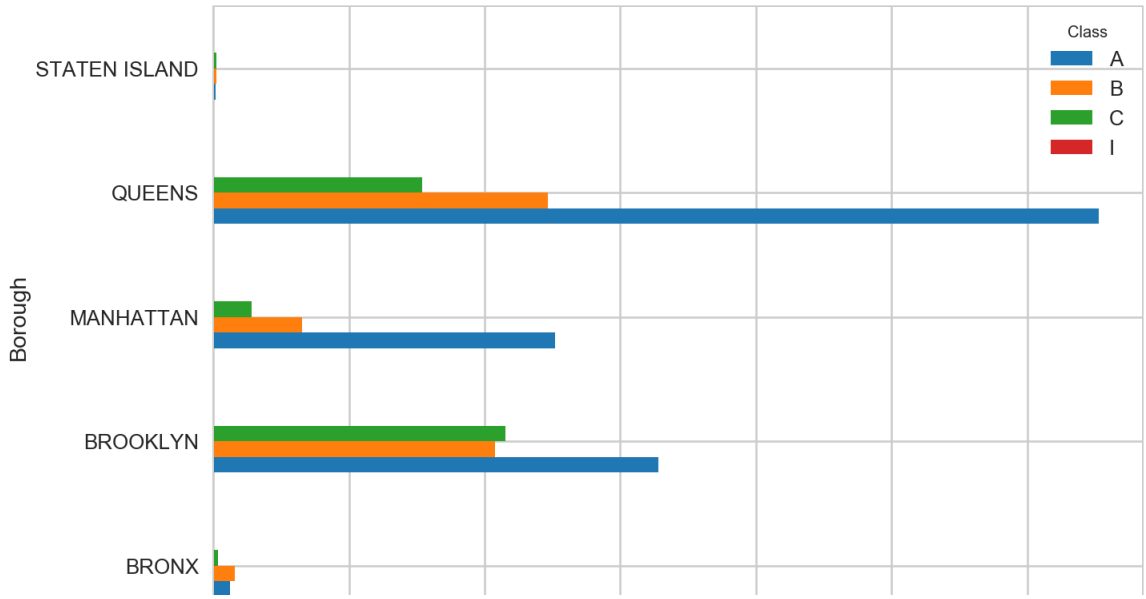
In the graph repsented below, *A* class violations (least severe) take the most time to get approved in Queens, followed by Brooklyn, then Manhattan. The *C* class violation takes the longest to get approved in Brooklyn, then Queens, then Manhattan.

In Manhattan, the approval rate is generally much faster than the other boroughs. Queens seems to have the slowest approval rate in all cases except class *C*, with Brooklyn being the slowest in that case.

To obtain this bar graph, we take "Time_Until_Approval" and graph it against "Class".

In [28]:

```
approval = housing[['Time_Until_Approval', 'Class']].set_index("Class")
approval_time= pd.pivot_table(
    data = housing.sample(3000),
    index='Borough',
    columns='Class',
    values='Time_Until_Approval'
).plot.barh()
#What are the units for Time Until Approval?
```



There's not a lot of I class (hazardous) violations compared to the other 3 classes, so let's look into that. Clearly from the KDE map and the value_counts, Brooklyn has the greatest number of I class violations. The top 9 streets with at least 1000 hazardous violations are found in Brooklyn's street of

1. **GREENE AVENUE 1642**
2. **BEDFORD AVENUE 1259**
3. **PUTNAM AVENUE 1223**
4. **JEFFERSON AVENUE 1206**
5. **MADISON STREET 1191**
6. **HANCOCK STREET 1180**
7. **3 AVENUE 1173**
8. **5 AVENUE 1140**
9. **BROADWAY 1041**

To obtain this data, we first create a new dataframe which we named "classI", which contains the 'HouseNumber', 'StreetName', 'Borough', 'Latitude', and 'Longitude' columns for only the class I violations.

In [29]:

```
classI = housing.loc[housing['Class']=="I", ['HouseNumber', 'StreetName', 'Borough', 'Latitude', 'Longitude']]
classI
```

Out[29]:

	HouseNumber	StreetName	Borough	Latitude	Longitude
35	115	WEST 118 STREET	MANHATTAN	40.803646	-73.949460
48	1137	CROES AVENUE	BRONX	40.828037	-73.870326
53	2507	CLARENDON ROAD	BROOKLYN	40.642967	-73.953376
82	12-07	31 AVENUE	QUEENS	40.768739	-73.933223
116	4	DOWNING STREET	BROOKLYN	40.685580	-73.961286
135	6903	15 AVENUE	BROOKLYN	40.620947	-73.999579
196	545	WEST 152 STREET	MANHATTAN	40.830194	-73.945653
198	106-35	156 STREET	QUEENS	40.696896	-73.798120
248	31-33	105 STREET	QUEENS	40.760551	-73.864216
256	147-24	230 STREET	QUEENS	40.657466	-73.752358
262	464	EMPIRE BOULEVARD	BROOKLYN	40.664146	-73.946837
279	811	CROWN STREET	BROOKLYN	40.665186	-73.930428
285	791	VAN NEST AVENUE	BRONX	40.844441	-73.862390
307	416	EAST 84 STREET	MANHATTAN	40.775304	-73.949745
309	80	SEGUINE PLACE	STATEN ISLAND	40.540608	-74.172801
310	108	HANCOCK STREET	BROOKLYN	40.682354	-73.951535
379	3512	BROADWAY	MANHATTAN	40.825517	-73.951105
389	196	CLERMONT AVENUE	BROOKLYN	40.692014	-73.970643
394	1935	OCEAN PARKWAY	BROOKLYN	40.601531	-73.966175
412	575	CENTRAL AVENUE	BROOKLYN	40.689408	-73.909889
432	2473	ADAM C POWELL BOULEVARD	MANHATTAN	40.820700	-73.939746
482	2739A	KINGSBRIDGE TERRACE	BRONX	40.871936	-73.903448
540	1824	55 STREET	BROOKLYN	40.624451	-73.983681
542	1624	PARKVIEW AVENUE	BRONX	40.845569	-73.827054

	HouseNumber	StreetName	Borough	Latitude	Longitude
549	1460	LELAND AVENUE	BRONX	40.836789	-73.864335
565	653	6 AVENUE	BROOKLYN	40.661808	-73.989338
584	52-02	72 PLACE	QUEENS	40.733355	-73.891247
608	2771	ATLANTIC AVENUE	BROOKLYN	40.676330	-73.891917
622	71-29	SUTTON PLACE	QUEENS	40.729549	-73.808149
625	35	EAST 95 STREET	BROOKLYN	40.664341	-73.926569
...	...	...	...	...	...
126800	53-68	METROPOLITAN AVENUE	QUEENS	40.713234	-73.911603
126803	225	WEST 78 STREET	MANHATTAN	40.782862	-73.979735
126829	326	JEFFERSON AVENUE	BROOKLYN	40.683858	-73.944954
126835	115-08	95 AVENUE	QUEENS	40.691473	-73.830352
126842	638A	HALSEY STREET	BROOKLYN	40.684269	-73.928357
126861	1982	ATLANTIC AVENUE	BROOKLYN	40.676961	-73.922420
126866	867	RUTLAND ROAD	BROOKLYN	40.660635	-73.930793
126868	1339	73 STREET	BROOKLYN	40.620911	-74.005284
126875	1050	EVERGREEN AVENUE	BRONX	40.824730	-73.880171
126906	8721	17 AVENUE	BROOKLYN	40.606073	-74.007466
126974	208	BAY 26 STREET	BROOKLYN	40.599598	-74.001091
127020	31-44	STEINWAY STREET	QUEENS	40.760237	-73.917987
127044	41-08	69 STREET	QUEENS	40.744253	-73.895939
127058	206-12	EMILY ROAD	QUEENS	40.788366	-73.787767
127060	7106	15 AVENUE	BROOKLYN	40.619822	-74.000774
127072	712	EAST 220 STREET	BRONX	40.884058	-73.861744
127114	312	CLERMONT AVENUE	BROOKLYN	40.688566	-73.969956
127201	29-35	ERICSSON STREET	QUEENS	40.762698	-73.864587
127233	136-45	35 AVENUE	QUEENS	40.764788	-73.830400
127235	828	SOUTH OAK DRIVE	BRONX	40.871929	-73.861975
127250	160	GLENMORE AVENUE	BROOKLYN	40.671851	-73.904949
127284	261	BAY 10 STREET	BROOKLYN	40.605545	-74.014391
127346	965	EAST 27 STREET	BROOKLYN	40.627403	-73.949892
127353	67-12	50 AVENUE	QUEENS	40.736953	-73.897545
127387	1103	WINTHROP STREET	BROOKLYN	40.661915	-73.923237
127409	96-22	35 AVENUE	QUEENS	40.753393	-73.871453
127416	332	RODNEY STREET	BROOKLYN	40.709227	-73.955941
127436	330	64 STREET	BROOKLYN	40.639393	-74.022113
127488	32-59	43 STREET	QUEENS	40.757414	-73.916944
127507	184	22 STREET	BROOKLYN	40.661767	-73.995668

7433 rows × 5 columns

Then we found which Borough had the most class I violations, which turned out to be Brooklyn.

In [30]:

```
classI['Borough'].value_counts()
```

Out[30]:

```
BROOKLYN        3654
```

```
QUEENS            1543
BRONX             1083
MANHATTAN         1049
STATEN ISLAND      104
Name: Borough, dtype: int64
```

Finally, we obtained a list containing the top ten streets for class I violations.

In [31]:

```
classI["StreetName"].value_counts().head(10)
```

Out[31]:

```
GREENE AVENUE      46
BROADWAY           36
60 STREET          34
HANCOCK STREET     34
FULTON STREET      33
3 AVENUE           29
MADISON STREET     29
DEAN STREET        29
PUTNAM AVENUE      28
41 STREET          28
Name: StreetName, dtype: int64
```

This is a map of all the I class violations found in NYC

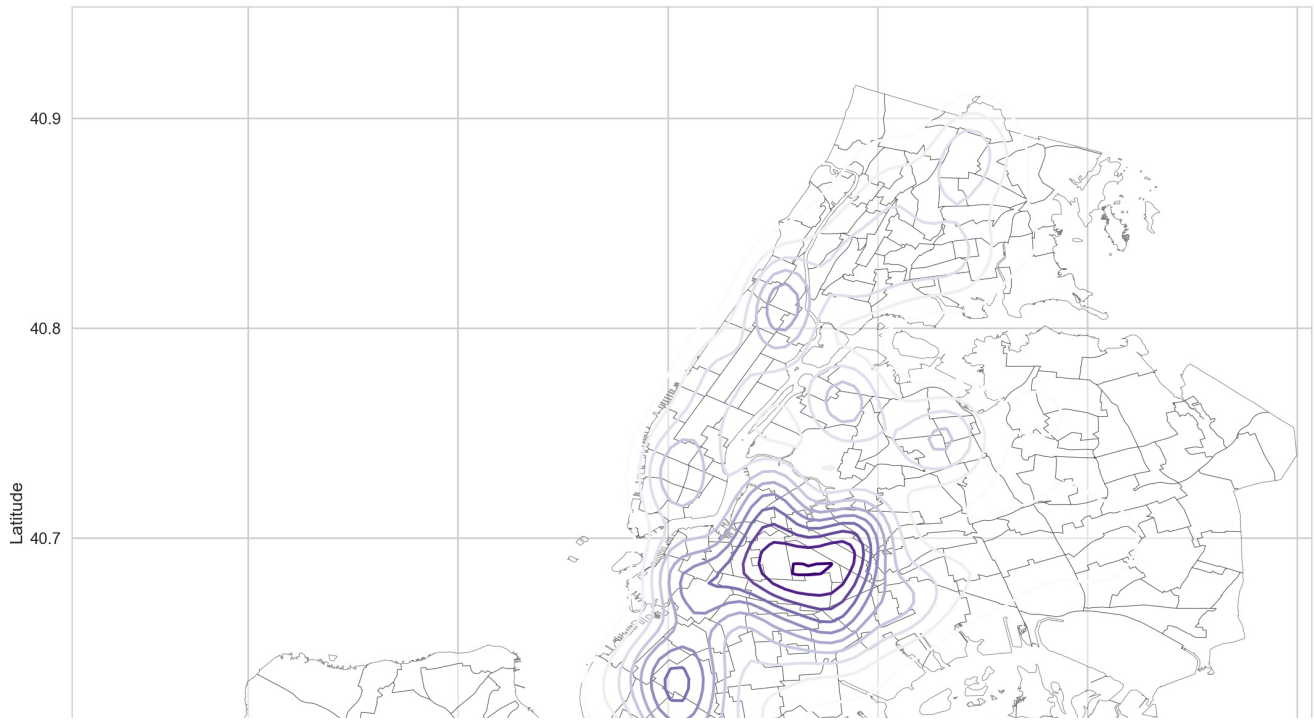
In [32]:

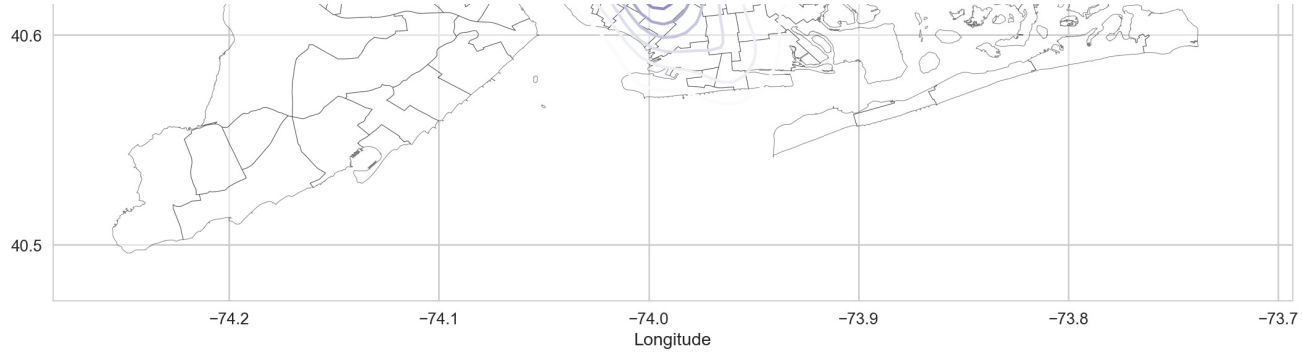
```
ny_map = df_nyc.plot(linewidth=0.5, color='White', edgecolor = 'Black', figsize = (20,15), alpha=0.5)

sns.kdeplot(
    classI.Longitude, classI.Latitude,
    gridsize=100,
    cmap=plt.cm.Purples,
    shade=False,
    shade_lowest=False,
    n_levels=10,
    ax=ny_map
)
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a248f5860>
```





Housing Complaints

In [3]:

```
complaints = pd.read_csv("https://data.cityofnewyork.us/api/views/uwyv-629c/rows.csv?accessType=DOWNLOAD")
```

In [34]:

```
len(complaints)
```

Out[34]:

```
1559398
```

Take a 10% sample of the data to make running code easier.

In [4]:

```
complaints_sample = complaints.sample(frac=0.1)
```

In [36]:

```
len(complaints_sample)
```

Out[36]:

```
155940
```

In [37]:

```
complaints_sample.columns
```

Out[37]:

```
Index(['ComplaintID', 'BuildingID', 'BoroughID', 'Borough', 'HouseNumber',
       'StreetName', 'Zip', 'Block', 'Lot', 'Apartment', 'CommunityBoard',
       'ReceivedDate', 'StatusID', 'Status', 'StatusDate'],
      dtype='object')
```

In [38]:

```
complaints_sample
```

Out[38]:

	ComplaintID	BuildingID	BoroughID	Borough	HouseNumber	StreetName	Zip	Block	Lot	Apartmen
1330839	8964213	886430	3	BROOKLYN	102	BELMONT AVENUE	11212.0	3743	15	1A
167458	7194961	382917	3	BROOKLYN	306	UNION AVENUE	11211.0	2791	1	6B

1555045	9350502	5053	1	MANHATTAN	4055	10 AVENUE	10034.0	2213	1	BLDG
ComplaintID	BuildingID	BoroughID	Borough	HouseNumber	StreetName	Zip	Block	Lot	Apartmen	
986319	8506973	64078	2	BRONX	737	EAST 156 STREET	10455.0	2646	36	BLDG
701738	8012012	52231	2	BRONX	1847	BRONXDALE AVENUE	10462.0	4056	76	1FL
189670	7231513	326267	3	BROOKLYN	1234	LINCOLN PLACE	11213.0	1389	20	2A
829864	8319150	5580	1	MANHATTAN	1496	AMSTERDAM AVENUE	10031.0	1987	36	3B
1135529	8716901	357002	3	BROOKLYN	214	PROSPECT PLACE	11238.0	1158	37	3A
1306014	8934418	103018	2	BRONX	1850	PHELAN PLACE	10453.0	2879	240	BLDG
340097	7411011	469256	4	QUEENS	35-27	72 STREET	11372.0	1272	63	1B
354040	7450000	40618	1	MANHATTAN	625	WEST 133 STREET	10027.0	2001	5	D5K
1311530	8961372	73800	2	BRONX	323	EAST MOSHOLU PARKWAY NORTH	10467.0	3333	1	BLDG
1444883	9133292	101049	2	BRONX	1812	PALISADE PLACE	10453.0	2877	551	2
345213	7385118	633731	4	QUEENS	3-44	BEACH 88 STREET	11693.0	16119	56	B
225718	7320622	69567	2	BRONX	723	EAST 222 STREET	10467.0	4836	35	1
163715	7149706	220873	3	BROOKLYN	3201	CLARENDON ROAD	11226.0	4932	39	3R
363503	7446203	218219	3	BROOKLYN	951	CARROLL STREET	11225.0	1280	1	5D
1516978	9350722	102591	2	BRONX	655	PELHAM PARKWAY NORTH	10467.0	4338	6	3J
328844	7426926	117	1	MANHATTAN	1286	1 AVENUE	10021.0	1464	1	3
1018332	8528237	117793	2	BRONX	3102	VILLA AVENUE	10468.0	3310	26	1C
1422146	9102224	63895	2	BRONX	408	EAST 152 STREET	10455.0	2374	38	2D
1469486	9243977	73782	2	BRONX	151	EAST MOSHOLU PARKWAY NORTH	10467.0	3335	110	3J
969288	8433359	213148	3	BROOKLYN	2954	BRIGHTON 12 STREET	11235.0	8711	36	2G
882250	8381463	95699	2	BRONX	1340	MERRIAM AVENUE	10452.0	2531	13	A23
1243625	8825393	328801	3	BROOKLYN	417	LORIMER STREET	11206.0	3022	25	14A
197883	7227854	109960	2	BRONX	1054	SOUTHERN BOULEVARD	10459.0	2743	19	410
257854	7309643	17854	1	MANHATTAN	107	EAST 88 STREET	10128.0	1517	7	4E
1533484	9351696	373064	3	BROOKLYN	1	SPENCER PLACE	11216.0	2000	14	BA

787775	8189334	811407	4	QUEENS	89-21	AVENUE	11373.0	1511	1	BLDG
758376	8129588	42824	1	MANHATTAN	532	WEST 159 STREET	10032.0	2117	20	ENTRY
...	...	...	...	...	...	...	...	...	...	...
1047872	8572036	822620	3	BROOKLYN	140	JOHNSON AVENUE	11206.0	3070	10	1
149135	7177816	187890	3	BROOKLYN	3220	AVENUE H	11210.0	7578	62	1C
432925	7648279	67249	2	BRONX	357	EAST 201 STREET	10458.0	3281	40	BLDG
1414072	9107469	347221	3	BROOKLYN	1439	OCEAN AVENUE	11230.0	7584	19	BLDG
1391993	9066345	225107	3	BROOKLYN	3120	CONEY ISLAND AVENUE	11235.0	8678	56	3C
1258125	8866783	352305	3	BROOKLYN	102	PARKVILLE AVENUE	11230.0	5427	8	1
72573	7110896	348665	3	BROOKLYN	902	OCEAN PARKWAY	11230.0	6518	5	BLDG
454048	7661817	687226	4	QUEENS	34-15	PARSONS BOULEVARD	11354.0	4995	1	2H
666532	7985484	21845	1	MANHATTAN	615	FT WASHINGTON AVENUE	10040.0	2179	354	BLDG
790999	8233163	477952	4	QUEENS	95-17	77 STREET	11416.0	9004	39	1
247998	7302435	55226	2	BRONX	4305	CARPENTER AVENUE	10466.0	5034	28	BLDG
1132252	8693414	87351	2	BRONX	2476	HUGHES AVENUE	10458.0	3076	39	1D
1006838	8486729	27990	1	MANHATTAN	1447	ST NICHOLAS AVENUE	10033.0	2165	38	61
1403299	9099581	240718	3	BROOKLYN	499	EAST 8 STREET	11218.0	5392	71	4K
1026054	8521139	116793	2	BRONX	1425	DR M L KING JR BOULEVARD	10452.0	2537	47	BLDG
1113845	8695622	807723	3	BROOKLYN	8831	BAY PARKWAY	11214.0	6471	90	16
323319	7432894	217470	3	BROOKLYN	1324	CARROLL STREET	11213.0	1292	7	2C
1040935	8531408	78296	2	BRONX	1245	FINDLAY AVENUE	10456.0	2436	33	6D
187373	7185389	69316	2	BRONX	907	EAST 221 STREET	10469.0	4692	35	2A
148487	7133033	323393	3	BROOKLYN	150	LEFFERTS AVENUE	11225.0	1328	16	5B
590819	7866632	305597	3	BROOKLYN	1235	HALSEY STREET	11237.0	3406	55	2R
417645	7543726	27412	1	MANHATTAN	106	RIVINGTON STREET	10002.0	411	72	3FLR
722068	8058377	570398	4	QUEENS	117-17	165 STREET	11434.0	12356	17	1ST
1485594	9184397	80791	2	BRONX	1553	ZEREGA AVENUE	10462.0	3972	61	1

416450	7347938	665250	4	QUEENS	142-10	HOOVER AVENUE	11435.0	9716	126	108
932059	8376043	122958	2	BRONX	1231	WHITE PLAINS ROAD	10472.0	3767	77	221
713155	8060265	40712	1	MANHATTAN	181	WEST 135 STREET	10030.0	1920	7	3D
1403146	9084160	50420	2	BRONX	1384	BOSTON ROAD	10456.0	2962	15	5
732696	8102590	538601	4	QUEENS	115-12	125 STREET	11420.0	11668	12	1FL
704537	8041224	355424	3	BROOKLYN	1581	PRESIDENT STREET	11213.0	1401	66	21

155940 rows × 15 columns

Here we are looking for the proportion of open and closed complaints.

In [39]:

```
complaints_status = complaints_sample.groupby('Status').count()
complaints_status['StatusID'].iloc[1]/complaints_status['StatusID'].iloc[0]
```

Out[39]:

```
0.015333528664908683
```

In [24]:

```
complaints_sample['ReceivedDate'] = pd.to_datetime(complaints_sample['ReceivedDate'],
format="%m/%d/%Y", errors = 'coerce')
complaints_sample['StatusDate'] = pd.to_datetime(complaints_sample['StatusDate'], format="%m/%d/%Y
", errors = 'coerce')
```

In [25]:

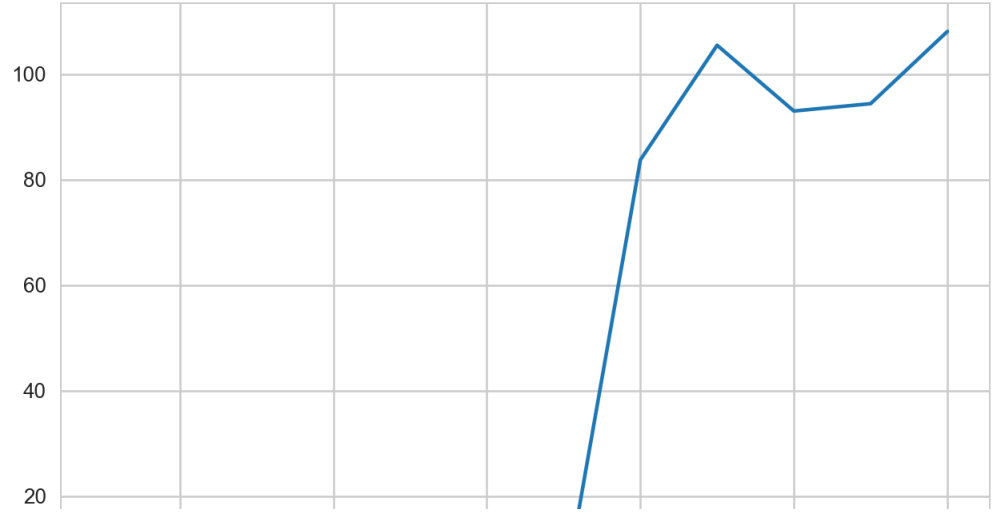
```
complaints_sample['Time_Until_Approval'] = (abs(complaints_sample['StatusDate'] -
complaints_sample['ReceivedDate']))
complaints_sample['Time_Until_Approval'] = (complaints_sample['Time_Until_Approval'] /
np.timedelta64(1, 'D')).fillna(0).astype(int)
```

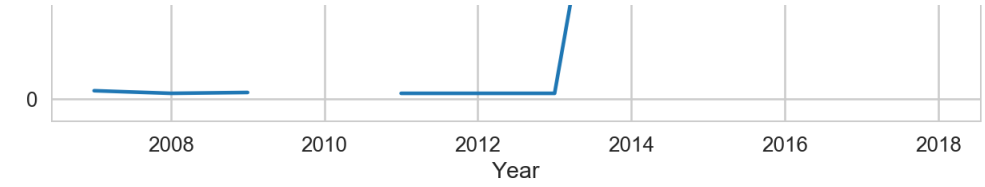
In [27]:

```
matplotlib.pyplot.xlabel("Year")
complaints_sample.StatusDate.value_counts().sort_index().resample('AS').mean().plot()
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1543bc88>
```





There was not enough data in the sample to form a continuous plot, so this is a plot of the number of complaints over time. Keep in mind that the data does not reach back as far as that of violations.

In [28]:

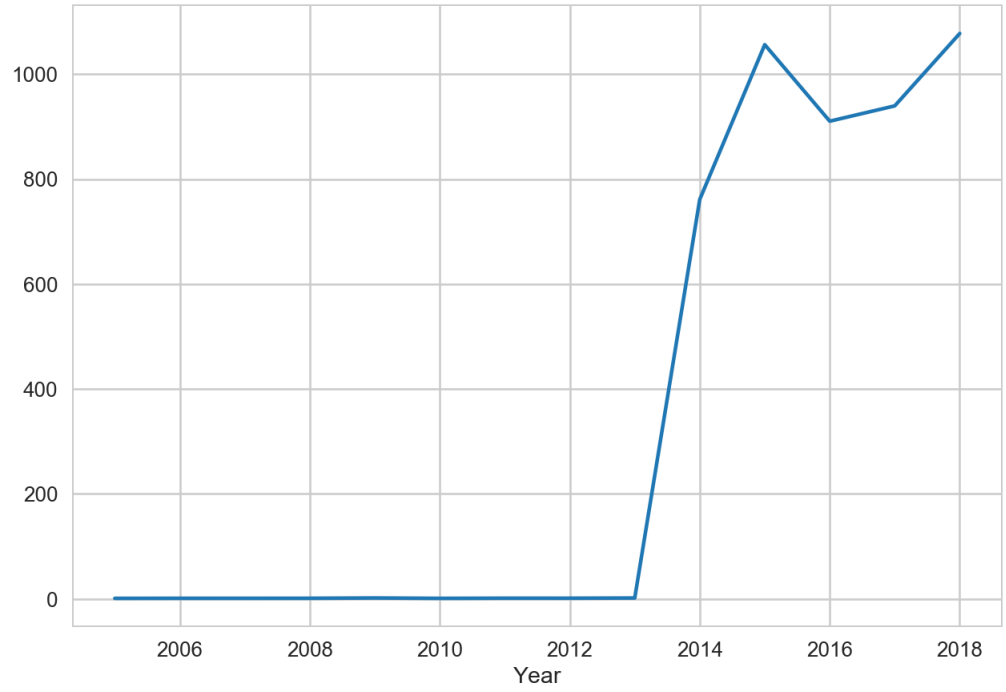
```
complaints['ReceivedDate'] = pd.to_datetime(complaints['ReceivedDate'], format="%m/%d/%Y", errors =
'coerce')
complaints['StatusDate'] = pd.to_datetime(complaints['StatusDate'], format="%m/%d/%Y", errors = 'co
erce')
```

In [29]:

```
#complaints_trend = complaints_sample[(complaints_sample.StatusDate > '2015-01-01')]
matplotlib.pyplot.xlabel("Year")
complaints.StatusDate.value_counts().sort_index().resample('AS').mean().plot()
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1710ca20>
```



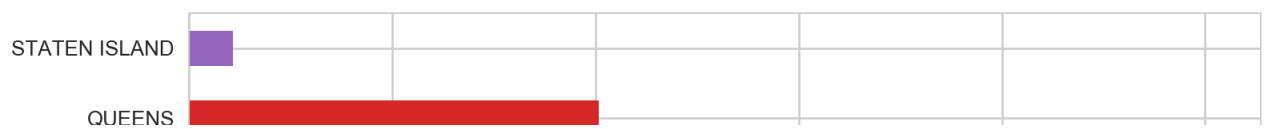
Let's take a look at the number of complaints in the five boroughs. The distribution mimics that of the violations above, where Brooklyn has the most number of complaints.

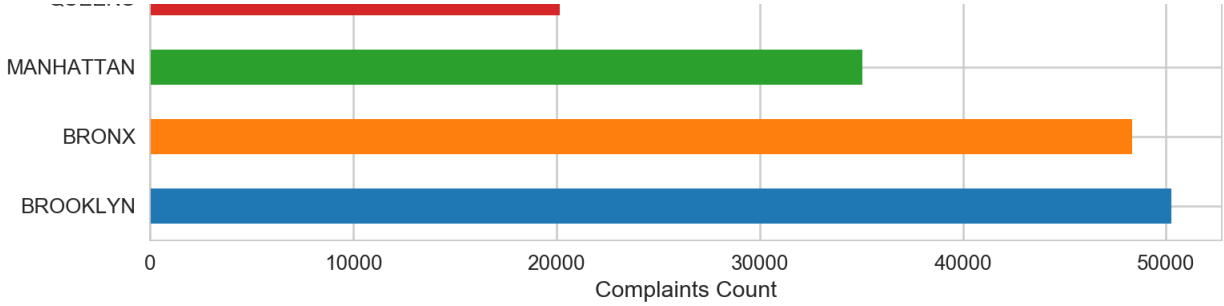
In [15]:

```
matplotlib.pyplot.xlabel("Complaints Count")
complaints_sample.Borough.value_counts().plot(kind='barh', figsize=(12,4))
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a148572b0>
```




MANHATTAN
BRONX
BROOKLYN
0
10000
20000
30000
40000
50000
Complaints Count