Project 1 : Customer Service Requests Analysis

DESCRIPTION

Background of Problem Statement :

NYC 311's mission is to provide the public with quick and easy access to all New York City government services and information while offering the best customer service. Each day, NYC311 receives thousands of requests related to several hundred types of non-emergency services, including noise complaints, plumbing issues, and illegally parked cars. These requests are received by NYC311 and forwarded to the relevant agencies such as the police, buildings, or transportation. The agency responds to the request, addresses it, and then closes it.

Problem Objective :

Perform a service request data analysis of New York City 311 calls. You will focus on the data wrangling techniques to understand the pattern in the data and also visualize the major complaint types. Domain: Customer Service

Tasks to be performed: 1.Import a 311 NYC service request. 2.Read or convert the columns ‘Created Date’ and Closed Date’ to datetime datatype and create a new column ‘Request_Closing_Time’ as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime) 3.Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining. 4.Order the complaint types based on the average ‘Request_Closing_Time’, grouping them for different locations. 5.Perform a statistical test for the following: a)Whether the average response time across complaint types is similar or not (overall) b) Are the type of complaint or service requested and location related?

1.Import a 311 NYC service request.

In [3]:
```
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [4]:
```
df = pd.read_csv('C:\\Users\\lenovo\\Desktop\\311_Service_Requests_from_2010_to_Present.csv')
```

```
C:\Users\lenovo\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (48,4
9) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

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

Out[5]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Bridge Highway Name	Bridge Highway Direction
0	32310363	12/31/2015 11:59:45 PM	01-01-16 0:55	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...	NaN	NaN
1	32309934	12/31/2015 11:59:44 PM	01-01-16 1:26	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...	NaN	NaN
2	32309159	12/31/2015 11:59:29 PM	01-01-16 4:51	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...	NaN	NaN
3	32305098	12/31/2015 11:57:46 PM	01-01-16 7:43	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	NaN	NaN
4	32306529	12/31/2015 11:56:58 PM	01-01-16 3:24	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	NaN	NaN

5 rows × 53 columns

In [6]:
```
df.shape
```

Out[6]:
```
(300698, 53)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency Name',
       'Complaint Type', 'Descriptor', 'Location Type', 'Incident Zip',
       'Incident Address', 'Street Name', 'Cross Street 1', 'Cross Street 2',
       'Intersection Street 1', 'Intersection Street 2', 'Address Type',
       'City', 'Landmark', 'Facility Type', 'Status', 'Due Date',
       'Resolution Description', 'Resolution Action Updated Date',
       'Community Board', 'Borough', 'X Coordinate (State Plane)',
       'Y Coordinate (State Plane)', 'Park Facility Name', 'Park Borough',
       'School Name', 'School Number', 'School Region', 'School Code',
       'School Phone Number', 'School Address', 'School City', 'School State',
       'School Zip', 'School Not Found', 'School or Citywide Complaint',
       'Vehicle Type', 'Taxi Company Borough', 'Taxi Pick Up Location',
       'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp',
       'Bridge Highway Segment', 'Garage Lot Name', 'Ferry Direction',
       'Ferry Terminal Name', 'Latitude', 'Longitude', 'Location'],
      dtype='object')
```

In [8]:
```
df['Complaint Type'].unique()
```

Out[8]:
```
array(['Noise - Street/Sidewalk', 'Blocked Driveway', 'Illegal Parking',
       'Derelict Vehicle', 'Noise - Commercial',
       'Noise - House of Worship', 'Posting Advertisement',
       'Noise - Vehicle', 'Animal Abuse', 'Vending', 'Traffic',
       'Drinking', 'Bike/Roller/Skate Chronic', 'Panhandling',
       'Noise - Park', 'Homeless Encampment', 'Urinating in Public',
       'Graffiti', 'Disorderly Youth', 'Illegal Fireworks',
       'Ferry Complaint', 'Agency Issues', 'Squeegee', 'Animal in a Park'],
      dtype=object)
```

In [9]:
```
df['Descriptor'].unique()
```

Out[9]:
```
array(['Loud Music/Party', 'No Access', 'Commercial Overnight Parking',
       'Blocked Sidewalk', 'Posted Parking Sign Violation',
       'Blocked Hydrant', 'With License Plate', 'Partial Access',
       'Unauthorized Bus Layover', 'Double Parked Blocking Vehicle',
       'Double Parked Blocking Traffic', 'Vehicle', 'Loud Talking',
       'Banging/Pounding', 'Car/Truck Music', 'Tortured',
       'In Prohibited Area', 'Congestion/Gridlock', 'Neglected',
       'Car/Truck Horn', 'In Public', 'Other (complaint details)', nan,
       'No Shelter', 'Truck Route Violation', 'Unlicensed',
       'Overnight Commercial Storage', 'Engine Idling',
       'After Hours - Licensed Est', 'Detached Trailer',
       'Underage - Licensed Est', 'Chronic Stoplight Violation',
       'Loud Television', 'Chained', 'Building', 'In Car',
       'Police Report Requested', 'Chronic Speeding',
       'Playing in Unsuitable Place', 'Drag Racing',
       'Police Report Not Requested', 'Nuisance/Truant', 'Homeless Issue',
       'Language Access Complaint', 'Disruptive Passenger',
       'Animal Waste'], dtype=object)
```

In [10]:
```
#get nan values in entire dataset
df.isnull().sum()
```

Out[10]:
```
Unique Key                             0
Created Date                           0
Closed Date                         2164
Agency                                 0
Agency Name                            0
Complaint Type                         0
Descriptor                          5914
Location Type                        131
Incident Zip                        2615
Incident Address                   44410
Street Name                        44410
Cross Street 1                     49279
Cross Street 2                     49779
Intersection Street 1             256840
Intersection Street 2             257336
Address Type                        2815
City                                2614
Landmark                          300349
Facility Type                       2171
Status                                 0
Due Date                               3
Resolution Description                 0
Resolution Action Updated Date      2187
Community Board                        0
Borough                                0
X Coordinate (State Plane)          3540
Y Coordinate (State Plane)          3540
Park Facility Name                     0
Park Borough                           0
School Name                            0
School Number                          0
School Region                          1
School Code                            1
School Phone Number                    0
School Address                         0
School City                            0
School State                           0
School Zip                             1
School Not Found                       0
School or Citywide Complaint      300698
Vehicle Type                      300698
Taxi Company Borough              300698
Taxi Pick Up Location             300698
Bridge Highway Name               300455
Bridge Highway Direction          300455
Road Ramp                         300485
Bridge Highway Segment            300485
Garage Lot Name                   300698
Ferry Direction                   300697
Ferry Terminal Name               300696
Latitude                            3540
Longitude                           3540
Location                            3540
dtype: int64
```

In [11]:
```
df.drop(columns=[], axis=1)
```

Out[11]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...
0	32310363	12/31/2015 11:59:45 PM	01-01-16 0:55	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...
1	32309934	12/31/2015 11:59:44 PM	01-01-16 1:26	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...
2	32309159	12/31/2015 11:59:29 PM	01-01-16 4:51	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...
3	32305098	12/31/2015 11:57:46 PM	01-01-16 7:43	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...
4	32306529	12/31/2015 11:56:58 PM	01-01-16 3:24	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...
...	...	...	...	...	...	...	...	...	...	...	...
300693	30281872	03/29/2015 12:33:41 AM	NaN	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	NaN	CRESCENT AVENUE	...
300694	30281230	03/29/2015 12:33:28 AM	03/29/2015 02:33:59 AM	NYPD	New York City Police Department	Blocked Driveway	Partial Access	Street/Sidewalk	11418.0	100-17 87 AVENUE	...
300695	30283424	03/29/2015 12:33:03 AM	03/29/2015 03:40:20 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	11206.0	162 THROOP AVENUE	...
300696	30280004	03/29/2015 12:33:02 AM	03/29/2015 04:38:35 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	10461.0	3151 EAST TREMONT AVENUE	...
300697	30281825	03/29/2015 12:33:01 AM	03/29/2015 04:41:50 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Store/Commercial	10036.0	251 WEST 48 STREET	...

300698 rows × 53 columns

In [12]:
```
#fix blank values in City column
df['City'].dropna(inplace=True)
```

In [13]:
```
#Shape after dropping nan values
df['City'].shape
```

Out[13]:
```
(300698,)
```

In [14]:
```
groupedby_complainttype=df.groupby('Complaint Type')
```

In [15]:
```
grp_df=groupedby_complainttype.get_group('Blocked Driveway')
grp_df.shape
```

Out[15]:
```
(77044, 53)
```

In [16]:
```
#count of null values in group city column data
grp_df['City'].isnull().sum()
```

Out[16]:
```
283
```

In [17]:
```
# fix those nan with unknown city value instead
grp_df['City'].fillna('Unknown City', inplace=True)
```

```
C:\Users\lenovo\Anaconda3\lib\site-packages\pandas\core\series.py:4463: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  return super().fillna(
```

2) Read or convert the columns ‘Created Date’ and Closed Date’ to datetime datatype and create a new column ‘Request_Closing_Time’ as the time elapsed between request creation and request closing.

In [18]:
```
import datetime
```

In [19]:
```
df['Created Date']= pd.to_datetime(df['Created Date'], infer_datetime_format= True)
```

In [20]:
```
df['Created Date']
```

Out[20]:
```
0        2015-12-31 23:59:45
1        2015-12-31 23:59:44
2        2015-12-31 23:59:29
3        2015-12-31 23:57:46
4        2015-12-31 23:56:58
                 ...
300693   2015-03-29 00:33:41
300694   2015-03-29 00:33:28
300695   2015-03-29 00:33:03
300696   2015-03-29 00:33:02
300697   2015-03-29 00:33:01
Name: Created Date, Length: 300698, dtype: datetime64[ns]
```

In [21]:
```
df['Closed Date']= pd.to_datetime(df['Closed Date'], infer_datetime_format=True)
```

In [22]:
```
df['Closed Date']
```

Out[22]:
```
0        2016-01-01 00:55:00
1        2016-01-01 01:26:00
2        2016-01-01 04:51:00
3        2016-01-01 07:43:00
4        2016-01-01 03:24:00
                 ...
300693                   NaT
300694   2015-03-29 02:33:59
300695   2015-03-29 03:40:20
300696   2015-03-29 04:38:35
300697   2015-03-29 04:41:50
Name: Closed Date, Length: 300698, dtype: datetime64[ns]
```

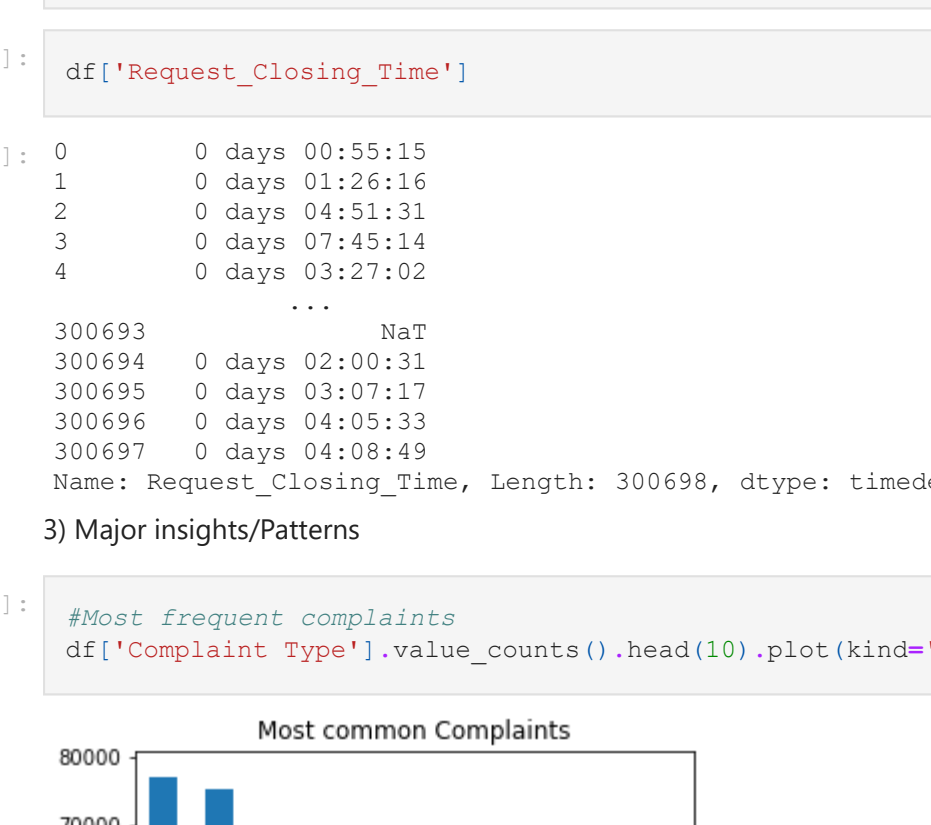
In [23]:
```
df['Request_Closing_Time']= df['Closed Date']-df['Created Date']
```

In [24]:
```
df['Request_Closing_Time']
```

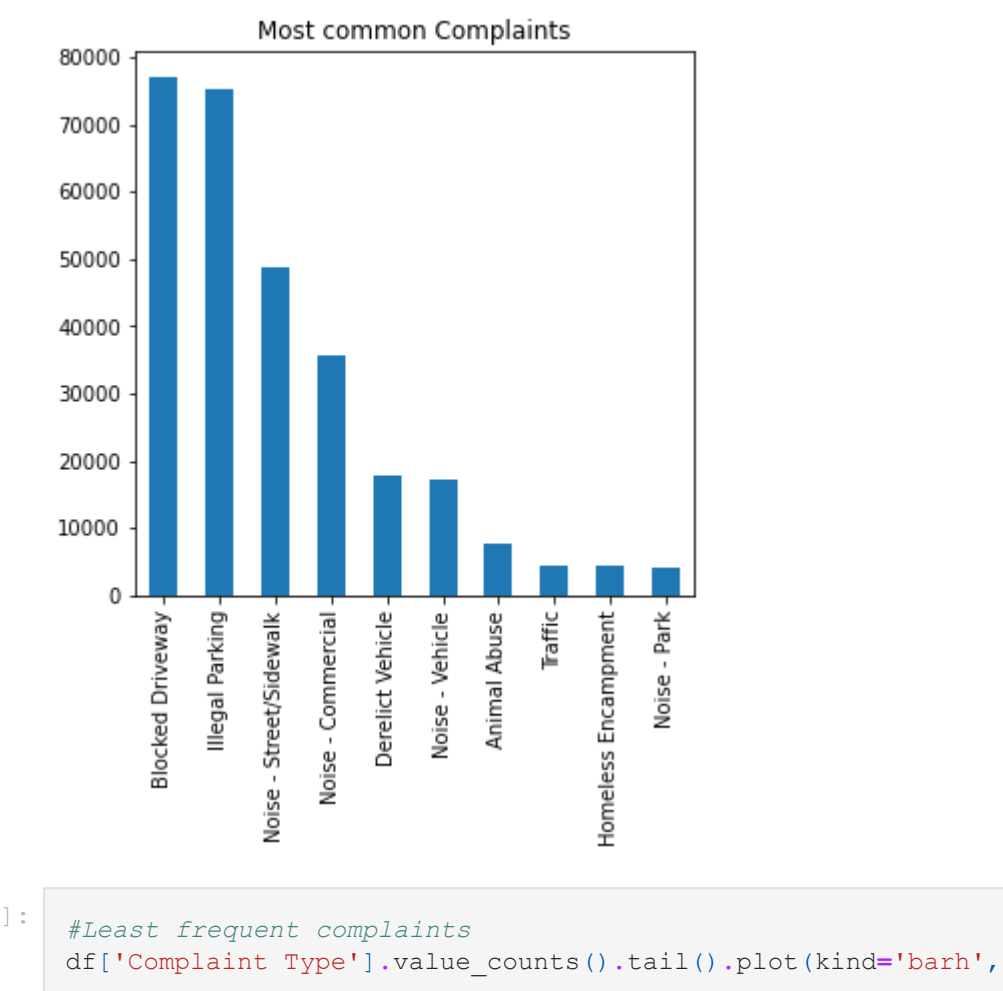
Out[24]:
```
0        0 days 00:55:15
1        0 days 01:26:16
2        0 days 04:51:31
3        0 days 07:45:14
4        0 days 03:27:02
               ...
300693               NaT
300694   0 days 02:00:31
300695   0 days 03:07:17
300696   0 days 04:05:33
300697   0 days 04:08:49
Name: Request_Closing_Time, Length: 300698, dtype: timedelta64[ns]
```

3) Major insights/Patterns

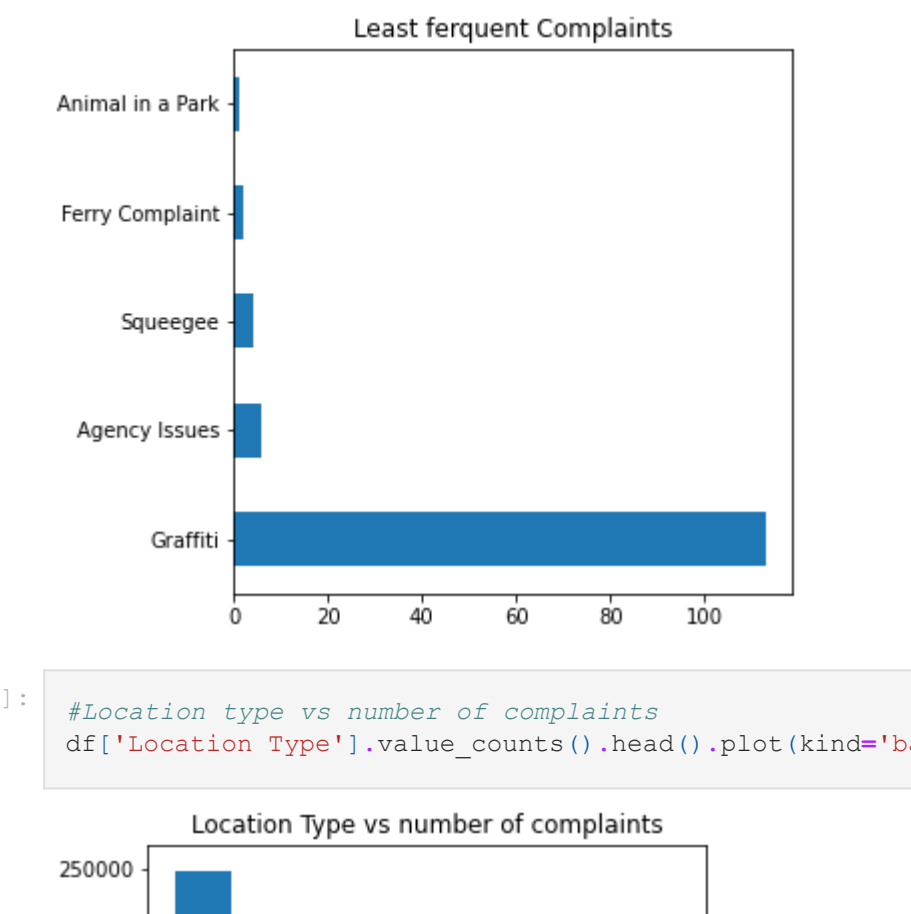
In [25]:
```
#Most frequent complaints
df['Complaint Type'].value_counts().head(10).plot(kind='bar',figsize=(5,5), title='Most common Complaints');
```



In [26]:
```
#Least frequent complaints
df['Complaint Type'].value_counts().tail().plot(kind='barh', figsize=(5,5), title='Least ferquent Complaints');
```



In [27]:
```
#Location type vs number of complaints
df['Location Type'].value_counts().head().plot(kind='bar', figsize=(5,5), title='Location Type vs number of com
```



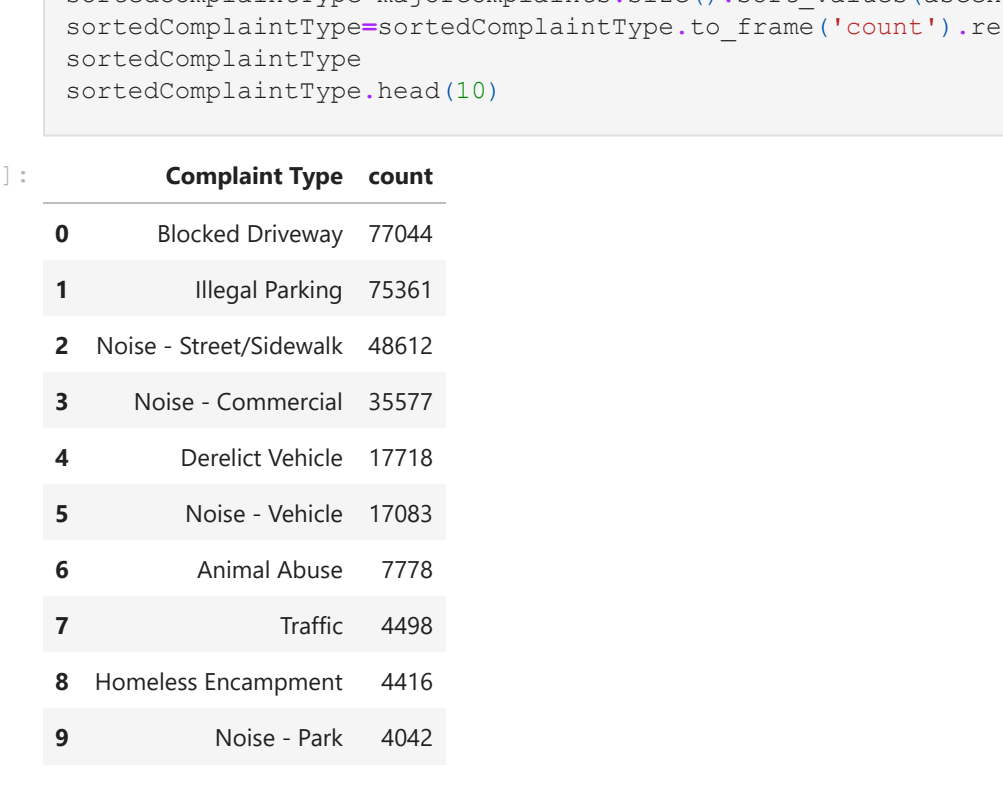
In [28]:
```
#Sorted complaint type
majorcomplaints=df.dropna(subset=['Complaint Type'])
majorcomplaints=df.groupby('Complaint Type')

sortedComplaintType=majorcomplaints.size().sort_values(ascending=False)
sortedComplaintType=sortedComplaintType.to_frame('count').reset_index()
sortedComplaintType
sortedComplaintType.head(10)
```

Out[28]:

	Complaint Type	count
0	Blocked Driveway	77044
1	Illegal Parking	75361
2	Noise - Street/Sidewalk	48612
3	Noise - Commercial	35577
4	Derelict Vehicle	17718
5	Noise - Vehicle	17083
6	Animal Abuse	7778
7	Traffic	4498
8	Homeless Encampment	4416
9	Noise - Park	4042

In [29]:
```
sortedComplaintType=sortedComplaintType.head()
plt.figure(figsize=(5,5))
plt.pie(sortedComplaintType['count'], labels=sortedComplaintType['Complaint Type'], autopct='%1.1f%%')
plt.show()
```



4) Order the complaint types based on the average ‘Request_Closing_Time’, grouping them for different locations.

In [30]:
```
df['Closing_time_in_secs']=df['Request_Closing_Time'].apply(lambda x:x.seconds)
df['Closing_time_in_secs']
```

Out[30]:
```
0          3315.0
1          5176.0
2         17491.0
3         27914.0
4         12422.0
           ...
300693        NaN
300694     7231.0
300695    11237.0
300696    14733.0
300697    14929.0
Name: Closing_time_in_secs, Length: 300698, dtype: float64
```

In [31]:
```
df.groupby(['Complaint Type','Location Type','City'])['Closing_time_in_secs'].mean()
```

Out[31]:
```
Complaint Type  Location Type    City
Animal Abuse    Commercial       BRONX              24713.900000
                                 BROOKLYN           30209.000000
                                 ELMHURST            7843.666667
                                 FAR ROCKAWAY        4550.000000
                                 JAMAICA             6180.000000
                                                        ...
Vending         Street/Sidewalk  STATEN ISLAND      15634.368421
                                 SUNNYSIDE          31725.600000
                                 WHITESTONE          8400.000000
                                 WOODHAVEN           7085.000000
                                 WOODSIDE           21244.166667
Name: Closing_time_in_secs, Length: 1394, dtype: float64
```

In [32]:
```
df.groupby(['Complaint Type','Location Type'])['Closing_time_in_secs'].agg(['min','max','mean','std'])
```

Out[32]:

Complaint Type	Location Type	min	max	mean	std
Animal Abuse	Commercial	1020.0	68340.0	16446.870968	16121.783030
	House and Store	577.0	68559.0	16189.677419	13931.121688
	Park/Playground	495.0	68820.0	11912.581967	11744.421817
	Parking Lot	540.0	79743.0	16018.163636	15429.423493
	Residential Building	286.0	79860.0	15822.929515	15481.245118
...	...	...	...	...	...
Urinating in Public	Subway Station	540.0	16728.0	4147.666667	5201.311536
Vending	Park/Playground	360.0	61130.0	12481.647619	11169.169389
	Residential Building/House	266.0	83540.0	14679.651741	15050.881356
	Store/Commercial	235.0	80382.0	13905.817130	13594.542076
	Street/Sidewalk	189.0	85800.0	13647.646809	13189.184260

71 rows × 4 columns

5) Perform a statistical test for the following: a) Whether the average response time across complaint types is similar or not (overall) b) Are the type of complaint or service requested and location related?

In [33]:
```
#Average response time across complaint types in seconds
df.groupby(['Complaint Type'])['Closing_time_in_secs'].mean().sort_values(ascending=True)
```

Out[33]:
```
Complaint Type
Animal in a Park              3005.000000
Posting Advertisement         6979.558642
Illegal Fireworks             9425.815476
Noise - House of Worship     10658.844995
Noise - Commercial           10768.414588
Traffic                      11512.049155
Noise - Street/Sidewalk      11597.681463
Noise - Park                 11720.108901
Noise - Vehicle              12154.427817
Urinating in Public          12326.261824
Disorderly Youth             12508.804196
Bike/Roller/Skate Chronic    12744.169811
Drinking                     13021.657255
Panhandling                  13475.734426
Vending                      13698.802899
Squeegee                     14564.250000
Homeless Encampment          14757.356884
Illegal Parking              14915.077671
Blocked Driveway             15784.925609
Animal Abuse                 16654.392508
Agency Issues                18937.166667
Derelict Vehicle             20119.817887
Graffiti                     20392.292035
Ferry Complaint                       NaN
Name: Closing_time_in_secs, dtype: float64
```

From the above data null hypothesis can be rejected. Since the average response time across complaint type are not equal. Null Hypothesis:Average response time across complaint type are equal. Alternate Hypothesis:Average response time across complaint type are not equal.

Following complaints have closing time in seconds which are very close. Disorderly Youth 12508.804196 Noise-Vehicle 12154.427817. One group can be formed for these compliants and one way Anova for these complaints can be performed.

In [34]:
```
df_clean=df[df['Closing_time_in_secs'].notnull()]
df_perfect=df_clean[df_clean['Closed Date']>=df_clean['Created Date']]
df_perfect['Day of Week']=df_perfect['Created Date'].dt.dayofweek
```

In [35]:
```
df_dis_youth=df_perfect[df_perfect['Complaint Type']=='Disorderly Youth']
df_dis_youth=df_dis_youth.loc[:,['Closing_time_in_secs']]
df_dis_youth.head()
```

Out[35]:

	Closing_time_in_secs
4670	713.0
9034	4605.0
12027	2345.0
12176	19415.0
17181	6849.0

In [36]:
```
df_noise_veh=df_perfect[df_perfect['Complaint Type']=='Noise - Vehicle']
df_noise_veh=df_noise_veh.loc[:,['Closing_time_in_secs']]
df_noise_veh.head()
```

Out[36]:

	Closing_time_in_secs
87	22949.0
156	7254.0
172	11319.0
221	10937.0
319	2615.0

In [37]:
```
df_type_res=df_perfect.loc[:,['Complaint Type', 'Closing_time_in_secs']]
df_type_res.head()
df_type_res.columns
```

Out[37]:
```
Index(['Complaint Type', 'Closing_time_in_secs'], dtype='object')
```

In [38]:
```
#stats f_oneway functions takes the groups as input and returns F and P-value
fvalue, pvalue= stats.f_oneway(df_dis_youth, df_noise_veh)
pvalue
```

Out[38]:
```
array([0.62538991])
```

Null hypothesis to be accepted for Disorderly Youth and Noise-Vehicle. p-value is 0.6

One Way Anova for Posting Advertisements And Derelict Vehicle

In [39]:
```
df_post_ad=df_perfect[df_perfect['Complaint Type']=='Posting Advertisement']
df_post_ad=df_post_ad.loc[:,['Closing_time_in_secs']]
df_post_ad.head()
```

Out[39]:

	Closing_time_in_secs
39	7596.0
42	7745.0
46	7834.0
49	8042.0
51	8137.0

In [40]:
```
df_der_veh=df_perfect[df_perfect['Complaint Type']=='Derelict Vehicle']
df_der_veh=df_der_veh.loc[:,['Closing_time_in_secs']]
df_der_veh.head()
```

Out[40]:

	Closing_time_in_secs
14	37763.0
151	14221.0
255	4913.0
256	14879.0
295	2712.0

In [41]:
```
#stats f_oneway functions takes the groups as input and returns F and P-value
fvalue, pvalue= stats.f_oneway(df_post_ad, df_der_veh)
pvalue
```

Out[41]:
```
array([1.93608425e-70])
```

Null Hypothesis for Posting Advertisement and Derelict Vehicle to be rejected p-value<0.05

Anova table for complaint type and closing time in seconds

In [42]:
```
df_perfect['Complaint_Type']=df_perfect['Complaint Type']
df_type_res=df_perfect.loc[:,['Complaint_Type','Closing_time_in_secs']]
#ordinary least square test model
model= ols('Closing_time_in_secs~Complaint_Type', data=df_type_res).fit()
anova_table= sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[42]:

	sum_sq	df	F	PR(>F)
Complaint_Type	1.824365e+12	22.0	443.372575	0.0
Residual	5.583167e+13	298511.0	NaN	NaN

Null Hypothesis to be rejected since p-value<0.05

Cross tab and Chi Square test for Location and Complaint type

In [43]:
```
df_city_type=pd.crosstab(df_perfect.City,df_perfect.Complaint_Type)
```

In [44]:
```
from scipy.stats import chi2_contingency
from scipy.stats import chi2
table=df_city_type
#print(table)
stat, p, dof, expected= chi2_contingency(table)
print('dof=%d'%dof)
print(expected)
#interpret test statistic
prob=0.95
critical= chi2.ppf(prob, dof)
print('probability=%.3f,critical=%.3f, stat=%.3f' %(prob, critical, stat))
if abs(stat)>=critical:
    print('Dependent(reject H0)')
else:
    print('Independent (fail to reject H0)')
#interpret p-value
alpha=1.0-prob
print('significant=%.3f, p=%.3f' %(alpha,p))
if p<=alpha:
    print('Dependent (reject H0)')
else:
    print('Independent(fail to reject H0)')
```

```
dof=1092
[[5.73348813e+00 7.38185674e-04 3.11514354e-01 ... 3.31740642e+00
  4.37005919e-01 2.80067645e+00]
 [1.64968090e+02 2.12396151e-02 8.96311756e+00 ... 9.54508301e+01
  1.25738521e+01 8.05830996e+01]
 [1.86598977e+01 2.40245883e-03 1.01383763e+00 ... 1.07966500e+01
  1.42225563e+00 9.11492880e+00]
 ...
 [6.41890057e+01 8.26432416e-03 3.48754479e+00 ... 3.71398728e+01
  4.89247990e+00 3.13548459e+01]
 [9.23612815e+01 1.18915001e-02 5.01821305e+00 ... 5.34404016e+01
  7.03976808e+00 4.51163515e+01]
 [3.12735716e+00 4.02646731e-04 1.69916921e-01 ... 1.80949441e+00
  2.38366865e-01 1.52764170e+00]]
probability=0.950,critical=1169.990, stat=119737.348
Dependent(reject H0)
significant=0.050, p=0.000
Dependent (reject H0)
```

In [ ]: