In [3]:

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

In [5]:

```
#Importing 311 nyc data
nyc=pd.read_csv("/Users/inmobi/Downloads/311_Service_Requests_from_2010_to_Present.csv")
nyc.head()
nyc
```

```
/opt/anaconda3/lib/python3.9/site-packages/IPython/core/interactiveshell.py:3444: DtypeWa
rning: Columns (48,49) have mixed types.Specify dtype option on import or set low_memory=
False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[5]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

300698 rows × 53 columns

In [7]:

```
#Checking the column features
nyc.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 [9]:

```
#Data information to see whether have any null
nyc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Unique Key                      300698 non-null  int64  
 1   Created Date                    300698 non-null  object 
 2   Closed Date                     298534 non-null  object 
 3   Agency                          300698 non-null  object 
 4   Agency Name                     300698 non-null  object 
 5   Complaint Type                  300698 non-null  object 
 6   Descriptor                      294784 non-null  object 
 7   Location Type                   300567 non-null  object 
 8   Incident Zip                    298083 non-null  float64
 9   Incident Address                256288 non-null  object 
 10  Street Name                     256288 non-null  object 
 11  Cross Street 1                  251419 non-null  object 
 12  Cross Street 2                  250919 non-null  object 
 13  Intersection Street 1           43858 non-null   object 
 14  Intersection Street 2           43362 non-null   object 
 15  Address Type                    297883 non-null  object 
 16  City                            298084 non-null  object 
 17  Landmark                        349 non-null     object 
 18  Facility Type                   298527 non-null  object 
 19  Status                          300698 non-null  object 
 20  Due Date                        300695 non-null  object 
 21  Resolution Description          300698 non-null  object 
 22  Resolution Action Updated Date  298511 non-null  object 
 23  Community Board                 300698 non-null  object 
 24  Borough                         300698 non-null  object 
 25  X Coordinate (State Plane)      297158 non-null  float64
 26  Y Coordinate (State Plane)      297158 non-null  float64
 27  Park Facility Name              300698 non-null  object 
 28  Park Borough                    300698 non-null  object 
 29  School Name                     300698 non-null  object 
 30  School Number                   300698 non-null  object 
 31  School Region                   300697 non-null  object 
 32  School Code                     300697 non-null  object 
 33  School Phone Number             300698 non-null  object 
 34  School Address                  300698 non-null  object 
 35  School City                     300698 non-null  object 
 36  School State                    300698 non-null  object 
 37  School Zip                      300697 non-null  object 
 38  School Not Found                300698 non-null  object 
 39  School or Citywide Complaint    0 non-null       float64
 40  Vehicle Type                    0 non-null       float64
 40  Vehicle Type                    0 non-null       float64
 41  Taxi Company Borough            0 non-null       float64
 42  Taxi Pick Up Location           0 non-null       float64
 43  Bridge Highway Name             243 non-null     object
 44  Bridge Highway Direction        243 non-null     object
 45  Road Ramp                       213 non-null     object
 46  Bridge Highway Segment          213 non-null     object
 47  Garage Lot Name                 0 non-null       float64
 48  Ferry Direction                 1 non-null       object
 49  Ferry Terminal Name             2 non-null       object
 50  Latitude                        297158 non-null  float64
 51  Longitude                       297158 non-null  float64
 52  Location                        297158 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 121.6+ MB
```

In [12]:

```
#To view the random sample data with all columns
pd.set_option("display.max_columns",None)
nyc.sample(5)
```

Out[12]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incide Z
175817	31127220	07/21/2015 06:24:57 PM	07/21/2015 07:13:30 PM	NYPD	New York City Police Department	Traffic	Congestion/Gridlock	Street/Sidewalk	11101
73549	31803413	10/20/2015 04:34:46 PM	10/20/2015 05:53:28 PM	NYPD	New York City Police Department	Illegal Parking	Double Parked Blocking Traffic	Street/Sidewalk	11366
79026	31755123	10/15/2015 10:36:50 AM	10/15/2015 05:18:55 PM	NYPD	New York City Police Department	Derelict Vehicle	With License Plate	Street/Sidewalk	10024
243746	30690167	05/24/2015 06:39:02 PM	05/24/2015 07:36:12 PM	NYPD	New York City Police Department	Noise - Vehicle	Car/Truck Music	Street/Sidewalk	10453
259502	30587117	05-10-15 16:15	05-10-15 16:58	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10456

In [14]:

```
#Drop the column, save it in another DataFrame and check the columns in new DataFrame
nyc_modf=nyc.drop(columns=["Unique Key"],axis=1)
nyc_modf
nyc_modf.columns
```

Out[14]:

```
Index(['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',
       '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 [19]:

```
#Evaluation of different outcome of different feature and count the value
pd.unique(nyc["Agency"])
nyc["Agency"].value_counts()
```

Out[19]:

```
NYPD    300698
Name: Agency, dtype: int64
```

In [20]:

```
nyc_modf=nyc_modf.drop(columns=["Agency"],axis=1)
```

In [22]:

```
pd.unique(nyc["Agency Name"])
nyc["Agency Name"].value_counts()
```

Out[22]:

```
New York City Police Department    300690
Internal Affairs Bureau                 6
NYPD                                    2
Name: Agency Name, dtype: int64
```

In [24]:

```
nyc["Complaint Type"].value_counts().head(5)
```

Out[24]:

```
Blocked Driveway           77044
Illegal Parking            75361
Noise - Street/Sidewalk    48612
Noise - Commercial         35577
Derelict Vehicle           17718
Name: Complaint Type, dtype: int64
```

In [25]:

```
nyc.Descriptor.value_counts().head(5)
```

Out[25]:

```
Loud Music/Party                 61430
No Access                        56976
Posted Parking Sign Violation    22440
Loud Talking                     21584
Partial Access                   20068
Name: Descriptor, dtype: int64
```

In [26]:

```
nyc["Location Type"].value_counts().head(5)
```

Out[26]:

```
Street/Sidewalk               249299
Store/Commercial               20381
```

```
Club/Bar/Restaurant            17360
Residential Building/House      6960
Park/Playground                 4773
Name: Location Type, dtype: int64
```

In [28]:

```
nyc["Incident Zip"].value_counts().head(5)
```

Out[28]:

```
11385.0    5167
11368.0    4298
11211.0    4225
11234.0    4150
11206.0    3781
Name: Incident Zip, dtype: int64
```

In [29]:

```
nyc["Incident Address"].value_counts().head(5)
```

Out[29]:

```
1207 BEACH AVENUE           904
78-15 PARSONS BOULEVARD     505
89 MOORE STREET             480
177 LAREDO AVENUE           311
2117 3 AVENUE               295
Name: Incident Address, dtype: int64
```

In [30]:

```
nyc["Street Name"].value_counts().head(5)
```

Out[30]:

```
BROADWAY          3237
3 AVENUE          1241
SHERMAN AVENUE    1156
BEACH AVENUE      1109
BEDFORD AVENUE     979
Name: Street Name, dtype: int64
```

In [31]:

```
nyc["Cross Street 1"].value_counts().head(5)
```

Out[31]:

```
BROADWAY            4338
BEND                4129
3 AVENUE            3112
5 AVENUE            3035
AMSTERDAM AVENUE    2651
Name: Cross Street 1, dtype: int64
```

In [32]:

```
nyc["Cross Street 2"].value_counts().head(5)
```

Out[32]:

```
BEND        4391
BROADWAY    3784
8 AVENUE    2766
DEAD END    2144
7 AVENUE    2140
Name: Cross Street 2, dtype: int64
```

In [33]:

```
nyc["Intersection Street 1"].value_counts().head(5)
```

Out[33]:

```
BROADWAY      672
170 STREET    441
44 STREET     355
6 AVENUE      348
85 STREET     237
Name: Intersection Street 1, dtype: int64
```

In [34]:

```
nyc["Intersection Street 2"].value_counts().head(5)
```

Out[34]:

```
BROADWAY    1358
6 AVENUE     715
2 AVENUE     617
5 AVENUE     551
3 AVENUE     487
Name: Intersection Street 2, dtype: int64
```

In [35]:

```
nyc["Address Type"].value_counts().head(5)
```

Out[35]:

```
ADDRESS         238644
INTERSECTION     43366
BLOCKFACE        12014
LATLONG           3509
PLACENAME          350
Name: Address Type, dtype: int64
```

In [38]:

```
nyc.City.value_counts().head(5)
```

Out[38]:

```
BROOKLYN         98307
NEW YORK         65994
BRONX            40702
STATEN ISLAND    12343
JAMAICA           7296
Name: City, dtype: int64
```

In [40]:

```
nyc.Landmark.value_counts().head(5)
```

Out[40]:

```
CENTRAL PARK              67
PROSPECT PARK             22
WASHINGTON SQUARE PARK    16
SUNSET PARK               13
UNION SQUARE PARK         13
Name: Landmark, dtype: int64
```

In [42]:

```
nyc["Facility Type"].value_counts().head()
```

Out[42]:

```
Precinct    298527
Name: Facility Type, dtype: int64
```

In [43]:

```
nyc.Status.value_counts().head()
```

Out[43]:

```
Closed      298471
Open          1439
Assigned       786
Draft            2
Name: Status, dtype: int64
```

In [46]:

```
nyc["Due Date"].value_counts().head(5)
```

Out[46]:

```
11-07-15 7:34    9
06-07-15 6:23    9
07-12-15 7:04    9
11-02-15 6:12    8
05-03-15 9:32    8
Name: Due Date, dtype: int64
```

In [47]:

```
nyc["Resolution Description"].value_counts().head(5)
```

Out[47]:

```
The Police Department responded to the complaint and with the information available obser
ved no evidence of the violation at that time.    90490
The Police Department responded to the complaint and took action to fix the condition.
61624
The Police Department responded and upon arrival those responsible for the condition were
gone.                                            58031
The Police Department responded to the complaint and determined that police action was no
t necessary.                                      38211
The Police Department issued a summons in response to the complaint.
28246
Name: Resolution Description, dtype: int64
```

In [48]:

```
nyc["School Name"].value_counts().head()
```

Out[48]:

```
Unspecified                        300697
Alley Pond Park - Nature Center         1
Name: School Name, dtype: int64
```

In [49]:

```
nyc["School Number"].value_counts().head()
```

Out[49]:

```
Unspecified    300697
Q001                1
Name: School Number, dtype: int64
```

In [50]:

```
nyc["School Region"].value_counts().head()
```

Out[50]:

```
Unspecified    300697
Name: School Region, dtype: int64
```

In [51]:

```
nyc["School Not Found"].value_counts().head()
```

Out[51]:

```
N    300698
Name: School Not Found, dtype: int64
```

In [52]:

```
nyc["School Code"].value_counts().head()
```

Out[52]:

```
Unspecified    300697
Name: School Code, dtype: int64
```

In [54]:

```
nyc["School Phone Number"].value_counts().head()
```

Out[54]:

```
Unspecified    300697
7182176034          1
Name: School Phone Number, dtype: int64
```

In [55]:

```
nyc["School Address"].value_counts().head()
```

Out[55]:

```
Unspecified                                     300697
Grand Central Parkway, near the soccer field         1
Name: School Address, dtype: int64
```

In [56]:

```
nyc["School City"].value_counts().head()
```

Out[56]:

```
Unspecified    300697
QUEENS              1
Name: School City, dtype: int64
```

In [57]:

```
nyc["School State"].value_counts().head()
```

Out[57]:

```
Unspecified    300697
NY                  1
Name: School State, dtype: int64
```

In [58]:

```
nyc["School Zip"].value_counts().head()
```

Out[58]:

```
Unspecified    300697
Name: School Zip, dtype: int64
```

In [59]:

```
nyc["School Not Found"].value_counts().head()
```

Out[59]:

```
N    300698
Name: School Not Found, dtype: int64
```

In [60]:

```
nyc["School or Citywide Complaint"].value_counts().head()
```

Out[60]:

Out[60]:

```
Series([], Name: School or Citywide Complaint, dtype: int64)
```

In [61]:

```
nyc.columns
```

Out[61]:

```
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 [63]:

```
nyc_modf=nyc_modf.drop(columns=['School Name', 'School Number', 'School Region', 'School
Code',
                                'School Phone Number', 'School Address', 'School City', 'S
chool State',
                                   'School Zip', 'School Not Found', 'School or Citywide C
omplaint'],axis=1)
```

In [65]:

```
nyc_modf.columns
```

Out[65]:

```
Index(['Created Date', 'Closed Date', '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',
       '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 [66]:

```
nyc['Vehicle Type'].value_counts()
```

Out[66]:

```
Series([], Name: Vehicle Type, dtype: int64)
```

In [67]:

```
nyc["Taxi Company Borough"].value_counts()
```

Out[67]:

```
Series([], Name: Taxi Company Borough, dtype: int64)
```

In [68]:

```
nyc["Taxi Pick Up Location"].value_counts()
```

Out[68]:

```
Series([], Name: Taxi Pick Up Location, dtype: int64)
```

In [1]:

```
nyc_modf = nyc_modf.drop(columns=['Vehicle Type','Taxi Company Borough','Taxi Pick Up Loc
ation'],axis=1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/3744689500.py in <module
>
----> 1 nyc_modf = nyc_modf.drop(columns=['Vehicle Type','Taxi Company Borough','Taxi Pic
k Up Location'],axis=1)

NameError: name 'nyc_modf' is not defined
```

In [2]:

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

In [3]:

```
nyc=pd.read_csv("/Users/inmobi/Downloads/311_Service_Requests_from_2010_to_Present.csv")
nyc.head()
nyc
```

```
/opt/anaconda3/lib/python3.9/site-packages/IPython/core/interactiveshell.py:3444: DtypeWa
rning: Columns (48,49) have mixed types.Specify dtype option on import or set low_memory=
False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[3]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...
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
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
		03/29/2015	03/29/2015		New York				

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip
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
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
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

300698 rows × 53 columns

In [7]:

```
nyc_modf=nyc.drop(columns=['Vehicle Type','Taxi Company Borough','Taxi Pick Up Location',
'Unique Key'],axis=1)
```

In [8]:

```
nyc_modf.columns
```

Out[8]:

```
Index(['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',
       '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 [12]:

```
nyc['Bridge Highway Name'].value_counts().head()
```

Out[12]:

```
FDR Dr                 33
Belt Pkwy              30
BQE/Gowanus Expwy      27
Staten Island Expwy    21
Cross Bronx Expwy      19
Name: Bridge Highway Name, dtype: int64
```

In [13]:

```
nyc['Bridge Highway Direction'].value_counts().head()
```

Out[13]:

```
East/Queens Bound            21
Northbound/Uptown            20
North/Bronx Bound            20
West/Staten Island Bound     18
North/Westbound (To GW Br)   17
Name: Bridge Highway Direction, dtype: int64
```

In [14]:

```
nyc['Road Ramp'].value_counts().head()
```

Out[14]:

```
Roadway     162
Roadway     102
Ramp         51
Name: Road Ramp, dtype: int64
```

In [15]:

```
nyc['Garage Lot Name'].value_counts().head()
```

Out[15]:

```
Series([], Name: Garage Lot Name, dtype: int64)
```

In [16]:

```
nyc['Ferry Direction'].value_counts().head()
```

Out[16]:

```
Manhattan Bound    1
Name: Ferry Direction, dtype: int64
```

In [17]:

```
nyc['Ferry Terminal Name'].value_counts().head()
```

Out[17]:

```
St. George Terminal (Staten Island)    1
Barberi                                1
Name: Ferry Terminal Name, dtype: int64
```

In [18]:

```
nyc_modf = nyc_modf.drop(columns=['Garage Lot Name','Ferry Direction','Ferry Terminal Nam
e'],axis=1)
```

In [19]:

```
nyc_modf.columns
```

Out[19]:

```
Index(['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',
       'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp',
       'Bridge Highway Segment', 'Latitude', 'Longitude', 'Location'],
      dtype='object')
```

In [20]:

```
nyc['Latitude'].value_counts().head()
```

Out[20]:

```
40.830362    902
40.721959    505
40.703819    480
40.647132    362
40.708726    341
Name: Latitude, dtype: int64
```

In [21]:

```
nyc['Longitude'].value_counts().head()
```

Out[21]:

```
-73.866022    902
-73.809697    505
-73.942073    480
-73.790654    341
-74.004623    340
Name: Longitude, dtype: int64
```

In [22]:

```
nyc['Location'].value_counts().head()
```

Out[22]:

```
(40.83036235589997, -73.86602154214397)     902
(40.72195913199264, -73.80969682426189)     505
(40.703818970933284, -73.94207345177706)    476
(40.708726489323325, -73.7906539235748)     341
(40.64713190020787, -74.00462341153786)     340
Name: Location, dtype: int64
```

In [23]:

```
nyc_modf.sample(10)
```

Out[23]:

	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	S
225213	06-09-15 5:49	06-09-15 8:32	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	10305.0	119 LAMPORT BOULEVARD	E
70264	10/23/2015 12:39:59 PM	10/23/2015 10:38:53 PM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11106.0	33-05 13 STREET	
88369	10-06-15 20:48	10-06-15 23:23	NYPD	New York City Police Department	Noise - Vehicle	Car/Truck Music	Street/Sidewalk	10458.0	NaN	
138984	08/23/2015 06:17:09 PM	08/23/2015 07:19:23 PM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10462.0	1265 OLMSTEAD AVENUE	
231855	06-03-15 20:03	06-04-15 6:01	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	11426.0	242-19 BRADDOCK AVENUE	
161885	08-02-15 23:05	08-02-15 23:37	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	10035.0	410 EAST 117 STREET	
60787	11-01-15 11:10	11-03-15 0:04	NYPD	New York City Police Department	Illegal Parking	Double Parked Blocking Vehicle	Street/Sidewalk	10461.0	2855 SAINT THERESA AVENUE	
115161	09/13/2015 10:29:47 AM	09/13/2015 03:34:02 PM	NYPD	New York City Police Department	Homeless Encampment	NaN	Street/Sidewalk	10018.0	WEST 35 STREET	
37513	11/23/2015 10:51:13 AM	11/23/2015 10:14:05 PM	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11102.0	12-05 30 DRIVE	
141738	08/21/2015 06:36:57 PM	08/21/2015 07:04:50 PM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	11209.0	169 72 STREET	

10 rows × 46 columns

In [24]:

```
nyc_modf.columns
```

Out[24]:

```
Index(['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',
       'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp',
       'Bridge Highway Segment', 'Latitude', 'Longitude', 'Location'],
      dtype='object')
```

In [25]:

```
#Data after cleaning
nyc_modf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 46 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Created Date                    300698 non-null  object 
 1   Closed Date                     298534 non-null  object 
 2   Agency                          300698 non-null  object 
 3   Agency Name                     300698 non-null  object 
 4   Complaint Type                  300698 non-null  object 
 5   Descriptor                      294784 non-null  object 
 6   Location Type                   300567 non-null  object 
 7   Incident Zip                    298083 non-null  float64
 8   Incident Address                256288 non-null  object 
 9   Street Name                     256288 non-null  object 
 10  Cross Street 1                  251419 non-null  object 
 11  Cross Street 2                  250919 non-null  object 
 12  Intersection Street 1           43858 non-null   object 
 13  Intersection Street 2           43362 non-null   object 
 14  Address Type                    297883 non-null  object 
 15  City                            298084 non-null  object 
 16  Landmark                        349 non-null     object 
 17  Facility Type                   298527 non-null  object 
 18  Status                          300698 non-null  object 
 19  Due Date                        300695 non-null  object 
 20  Resolution Description          300698 non-null  object 
 21  Resolution Action Updated Date  298511 non-null  object 
 22  Community Board                 300698 non-null  object 
 23  Borough                         300698 non-null  object 
 24  X Coordinate (State Plane)      297158 non-null  float64
 25  Y Coordinate (State Plane)      297158 non-null  float64
 26  Park Facility Name              300698 non-null  object 
 27  Park Borough                    300698 non-null  object 
 28  School Name                     300698 non-null  object 
 29  School Number                   300698 non-null  object 
 30  School Region                   300697 non-null  object 
 31  School Code                     300697 non-null  object 
 32  School Phone Number             300698 non-null  object 
 33  School Address                  300698 non-null  object 
 34  School City                     300698 non-null  object 
 35  School State                    300698 non-null  object 
 36  School Zip                      300697 non-null  object 
 37  School Not Found                300698 non-null  object 
 38  School or Citywide Complaint    0 non-null       float64
 39  Bridge Highway Name             243 non-null     object 
 40  Bridge Highway Direction        243 non-null     object 
 41  Road Ramp                       213 non-null     object 
 42  Bridge Highway Segment          213 non-null     object 
 42   Bridge Highway Segment            213 non-null     object 
 43   Latitude                          297158 non-null  float64
 44   Longitude                         297158 non-null  float64
 45   Location                          297158 non-null  object 
dtypes: float64(6), object(40)
memory usage: 105.5+ MB
```

1. **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**

Now, converting 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 [28]:

```
nyc_modf["Closed Date"]=pd.to_datetime(nyc_modf["Closed Date"])
nyc_modf["Created Date"]=pd.to_datetime(nyc_modf["Created Date"])

nyc_modf["Request_Closing_Time"]=nyc_modf["Closed Date"]-nyc_modf["Created Date"]


#nyc_modf = nyc_modf[(nyc_modf.Request_Closing_Time)>=0]
```

In [29]:

```
nyc_modf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 47 columns):
 #    Column                            Non-Null Count   Dtype         
---   ------                            --------------   -----         
 0    Created Date                      300698 non-null  datetime64[ns]
 1    Closed Date                       298534 non-null  datetime64[ns]
 2    Agency                            300698 non-null  object        
 3    Agency Name                       300698 non-null  object        
 4    Complaint Type                    300698 non-null  object        
 5    Descriptor                        294784 non-null  object        
 6    Location Type                     300567 non-null  object        
 7    Incident Zip                      298083 non-null  float64       
 8    Incident Address                  256288 non-null  object        
 9    Street Name                       256288 non-null  object        
 10   Cross Street 1                    251419 non-null  object        
 11   Cross Street 2                    250919 non-null  object        
 12   Intersection Street 1             43858 non-null   object        
 13   Intersection Street 2             43362 non-null   object        
 14   Address Type                      297883 non-null  object        
 15   City                              298084 non-null  object        
 16   Landmark                          349 non-null     object        
 17   Facility Type                     298527 non-null  object        
 18   Status                            300698 non-null  object        
 19   Due Date                          300695 non-null  object        
 20   Resolution Description            300698 non-null  object        
 21   Resolution Action Updated Date    298511 non-null  object        
 22   Community Board                   300698 non-null  object        
 23   Borough                           300698 non-null  object        
 24   X Coordinate (State Plane)        297158 non-null  float64       
 25   Y Coordinate (State Plane)        297158 non-null  float64       
 26   Park Facility Name                300698 non-null  object        
 27   Park Borough                      300698 non-null  object        
 28   School Name                       300698 non-null  object        
 29   School Number                     300698 non-null  object        
 30   School Region                     300697 non-null  object        
 31   School Code                       300697 non-null  object        
 32   School Phone Number               300698 non-null  object        
 33   School Address                    300698 non-null  object        
 34   School City                       300698 non-null  object        
 35   School State                      300698 non-null  object        
 36   School Zip                        300697 non-null  object        
 37   School Not Found                  300698 non-null  object        
 38   School or Citywide Complaint      0 non-null       float64       
 39   Bridge Highway Name               243 non-null     object        
 39   Bridge Highway Name             243 non-null     object
 40   Bridge Highway Direction        243 non-null     object
 41   Road Ramp                       213 non-null     object
 42   Bridge Highway Segment          213 non-null     object
 43   Latitude                        297158 non-null  float64
 44   Longitude                       297158 non-null  float64
 45   Location                        297158 non-null  object
 46   Request_Closing_Time            298534 non-null  timedelta64[ns]
dtypes: datetime64[ns](2), float64(6), object(38), timedelta64[ns](1)
memory usage: 107.8+ MB
```

In [30]:

```
nyc_modf=nyc_modf.drop(columns=['School Name','School Number','School Region','School Code','School Phone Number','School Address','School City','School State','School Zip','School Not Found'])
```

In [31]:

```
nyc_modf.columns
```

Out[31]:

```
Index(['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 or Citywide Complaint', 'Bridge Highway Name',
       'Bridge Highway Direction', 'Road Ramp', 'Bridge Highway Segment',
       'Latitude', 'Longitude', 'Location', 'Request_Closing_Time'],
      dtype='object')
```

In [32]:

```
nyc_modf["Closed Date"]=pd.to_datetime(nyc_modf["Closed Date"])
nyc_modf["Created Date"]=pd.to_datetime(nyc_modf["Created Date"])

nyc_modf["Request_Closing_Time"]=nyc_modf["Closed Date"]-nyc_modf["Created Date"]


#nyc_modf = nyc_modf[(nyc_modf.Request_Closing_Time)>=0]
```

In [33]:

```
nyc_modf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 37 columns):
 #    Column                          Non-Null Count   Dtype
---   ------                          --------------   -----
 0    Created Date                    300698 non-null  datetime64[ns]
 1    Closed Date                     298534 non-null  datetime64[ns]
 2    Agency                          300698 non-null  object
 3    Agency Name                     300698 non-null  object
 4    Complaint Type                  300698 non-null  object
 5    Descriptor                      294784 non-null  object
 6    Location Type                   300567 non-null  object
 7    Incident Zip                    298083 non-null  float64
 8    Incident Address                256288 non-null  object
 9    Street Name                     256288 non-null  object
 10   Cross Street 1                  251419 non-null  object
 11   Cross Street 2                  250919 non-null  object
 12   Intersection Street 1           43858 non-null   object
 13   Intersection Street 2           43362 non-null   object
 14   Address Type                    297883 non-null  object
 15   City                            298084 non-null  object
 16   Landmark                        349 non-null     object
 17   Facility Type                   298527 non-null  object
 17  Facility Type                   298527 non-null  object
 18  Status                          300698 non-null  object
 19  Due Date                        300695 non-null  object
 20  Resolution Description          300698 non-null  object
 21  Resolution Action Updated Date  298511 non-null  object
 22  Community Board                 300698 non-null  object
 23  Borough                         300698 non-null  object
 24  X Coordinate (State Plane)      297158 non-null  float64
 25  Y Coordinate (State Plane)      297158 non-null  float64
 26  Park Facility Name              300698 non-null  object
 27  Park Borough                    300698 non-null  object
 28  School or Citywide Complaint    0 non-null       float64
 29  Bridge Highway Name             243 non-null     object
 30  Bridge Highway Direction        243 non-null     object
 31  Road Ramp                       213 non-null     object
 32  Bridge Highway Segment          213 non-null     object
 33  Latitude                        297158 non-null  float64
 34  Longitude                       297158 non-null  float64
 35  Location                        297158 non-null  object
 36  Request_Closing_Time            298534 non-null  timedelta64[ns]
dtypes: datetime64[ns](2), float64(6), object(28), timedelta64[ns](1)
memory usage: 84.9+ MB
```

In [34]:

```
nyc_modf.sample(5)
```

Out[34]:

	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	Street
147560	2015-08-15 22:35:58	2015-08-15 22:53:47	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	11226.0	1037 ROGERS AVENUE	RO AV
117330	2015-09-11 18:14:00	2015-09-11 19:15:00	NYPD	New York City Police Department	Illegal Parking	Blocked Hydrant	Street/Sidewalk	11385.0	70-24 65 PLACE	65 P
156200	2015-08-08 09:07:00	2015-08-08 10:32:00	NYPD	New York City Police Department	Blocked Driveway	Partial Access	Street/Sidewalk	11419.0	107-28 116 STREET	116 ST
212518	2015-06-20 01:46:22	2015-06-20 05:29:52	NYPD	New York City Police Department	Noise - Vehicle	Car/Truck Music	Street/Sidewalk	11221.0	242 KOSCIUSKO STREET	KOSCI ST
44057	2015-11-17 09:13:10	2015-11-17 11:42:47	NYPD	New York City Police Department	Illegal Parking	Double Parked Blocking Traffic	Street/Sidewalk	11201.0	239 BALTIC STREET	B ST

5 rows × 37 columns

1. **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**

Let's visualize the feature "Complaint Type" first, then visualize the others.

In [48]:

```
#Measure the frequency of Complaint Type
nyc_complaint = nyc['Complaint Type'].value_counts()
nyc_complaint = nyc_complaint.to_frame()
nyc_complaint = nyc_complaint.rename(columns={'Complaint Type':'Counts'})
nyc_complaint
```

Out[48]:

	Counts
Blocked Driveway	77044
Illegal Parking	75361
Noise - Street/Sidewalk	48612
Noise - Commercial	35577
Derelict Vehicle	17718
Noise - Vehicle	17083
Animal Abuse	7778
Traffic	4498
Homeless Encampment	4416
Noise - Park	4042
Vending	3802
Drinking	1280
Noise - House of Worship	931
Posting Advertisement	650
Urinating in Public	592
Bike/Roller/Skate Chronic	427
Panhandling	307
Disorderly Youth	286
Illegal Fireworks	168
Graffiti	113
Agency Issues	6
Squeegee	4
Ferry Complaint	2
Animal in a Park	1

In [53]:

```
#Calculating the percentage of the complaints
nyc_complaint['Percentage'] = np.around((nyc_complaint.Counts/nyc_complaint.Counts.sum())
*100,decimals=2)
nyc_complaint
```

Out[53]:

	Counts	Percentage
Blocked Driveway	77044	25.62
Illegal Parking	75361	25.06
Noise - Street/Sidewalk	48612	16.17
Noise - Commercial	35577	11.83
Derelict Vehicle	17718	5.89
Noise - Vehicle	17083	5.68
Animal Abuse	7778	2.59
Traffic	4498	1.50
Homeless Encampment	4416	1.47
Noise - Park	4042	1.34
Vending	3802	1.26
Drinking	1280	0.43

	Counts	Percentage
Drinking	1280	0.43
Noise - House of Worship	931	0.31
Posting Advertisement	650	0.22
Urinating in Public	592	0.20
Bike/Roller/Skate Chronic	427	0.14
Panhandling	307	0.10
Disorderly Youth	286	0.10
Illegal Fireworks	168	0.06
Graffiti	113	0.04
Agency Issues	6	0.00
Squeegee	4	0.00
Ferry Complaint	2	0.00
Animal in a Park	1	0.00

In [54]:

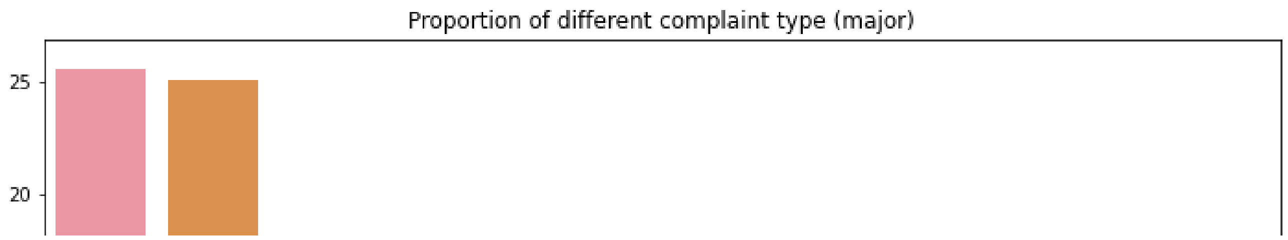
```
# Keeping the major complaint types

nyc_complaint = nyc_complaint[nyc_complaint.Percentage>1.0]
nyc_complaint = nyc_complaint.reset_index()
nyc_complaint = nyc_complaint.rename(columns={'index':'Complaint Type'})
nyc_complaint
```

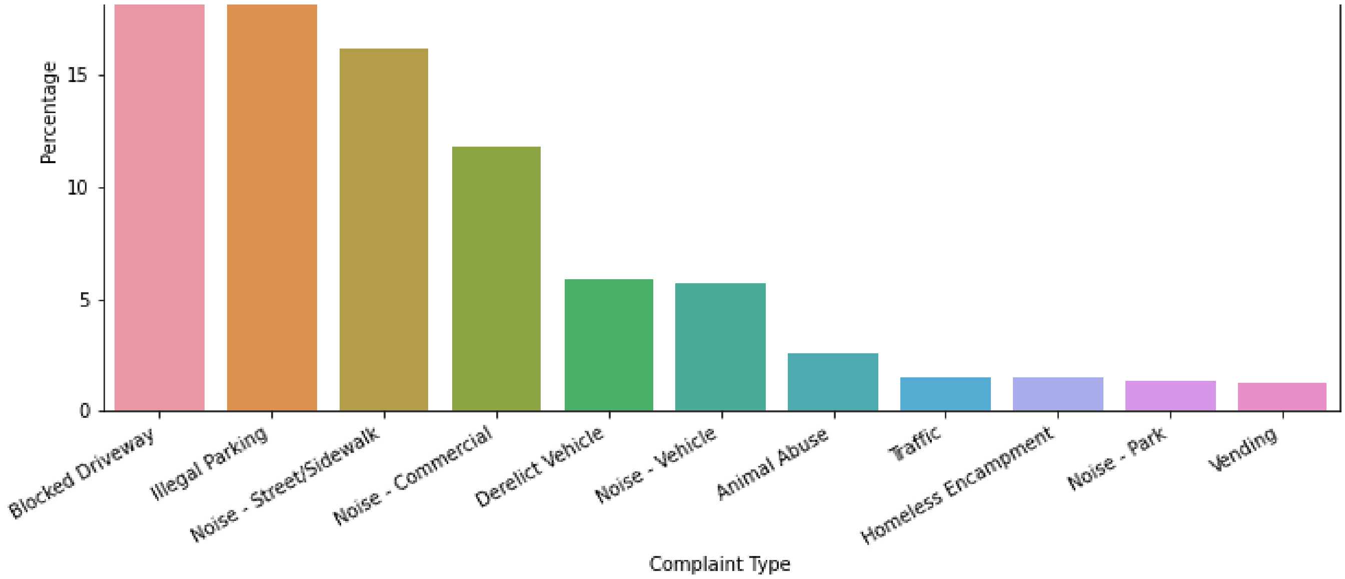
Out[54]:

	Complaint Type	Counts	Percentage
0	Blocked Driveway	77044	25.62
1	Illegal Parking	75361	25.06
2	Noise - Street/Sidewalk	48612	16.17
3	Noise - Commercial	35577	11.83
4	Derelict Vehicle	17718	5.89
5	Noise - Vehicle	17083	5.68
6	Animal Abuse	7778	2.59
7	Traffic	4498	1.50
8	Homeless Encampment	4416	1.47
9	Noise - Park	4042	1.34
10	Vending	3802	1.26

In [57]:

```
# Visualization of the above evaluated dataset

plt.figure(figsize=(12,6))
com_type = sns.barplot(x=nyc_complaint['Complaint Type'],y=nyc_complaint.Percentage,data=
nyc_complaint)
com_type.set_xticklabels(com_type.get_xticklabels(), rotation=30, ha="right")
plt.title('Proportion of different complaint type (major)')
plt.show()
plt.tight_layout()
```





```
<Figure size 432x288 with 0 Axes>
```

In [59]:

```
# Applying the above procedure for Descriptor

data_descriptor = np.around(((nyc_modf['Descriptor'].value_counts()*100) / nyc_modf['Descriptor'].value_counts().sum()),
                                decimals=2)
data_descriptor = data_descriptor.to_frame()
data_descriptor = data_descriptor.rename(columns={'Descriptor':'Percentage'})
data_descriptor['Descriptor'] = data_descriptor.index
cols = data_descriptor.columns.tolist()
cols = cols[-1:]+cols[:-1]
data_descriptor = data_descriptor[cols]
data_descriptor = data_descriptor[(data_descriptor.Percentage) >= 2.0]
data_descriptor = data_descriptor.reset_index()
data_descriptor = data_descriptor.drop(columns=['index'],axis=1)
data_descriptor
```

Out[59]:

	Descriptor	Percentage
0	Loud Music/Party	20.84
1	No Access	19.33
2	Posted Parking Sign Violation	7.61
3	Loud Talking	7.32
4	Partial Access	6.81
5	With License Plate	6.01
6	Blocked Hydrant	5.46
7	Commercial Overnight Parking	4.13
8	Car/Truck Music	3.82
9	Blocked Sidewalk	3.77

In [60]:

```
# Applying the above procedure for Location Type-------------------

data_location_type = np.around(((nyc_modf['Location Type'].value_counts()*100) / nyc_modf['Location Type'].value_counts().sum()),
                                decimals=2)
data_location_type = data_location_type.to_frame()
data_location_type = data_location_type.rename(columns={'Location Type':'Percentage'})
data_location_type['Location Type'] = data_location_type.index
cols = data_location_type.columns.tolist()
cols = cols[-1:]+cols[:-1]
data_location_type = data_location_type[cols]
data_location_type = data_location_type[(data_location_type.Percentage) >= 0.1]
data_location_type = data_location_type.reset_index()
data_location_type = data_location_type.drop(columns=['index'],axis=1)
data_location_type
```

Out[60]:

	Location Type	Percentage
0	Street/Sidewalk	82.94
1	Store/Commercial	6.78
2	Club/Bar/Restaurant	5.78
3	Residential Building/House	2.32
4	Park/Playground	1.59
5	House of Worship	0.31

In [61]:

```
# Applying the above procedure for City

data_city = np.around(((nyc_modf['City'].value_counts()*100) / nyc_modf['City'].value_co
unts().sum()),
                                decimals=2)
data_city = data_city.to_frame()
data_city = data_city.rename(columns={'City':'Percentage'})
data_city['City'] = data_city.index
cols = data_city.columns.tolist()
cols = cols[-1:]+cols[:-1]
data_city = data_city[cols]
data_city = data_city[(data_city.Percentage) >= 1.0]
data_city = data_city.reset_index()
data_city = data_city.drop(columns=['index'],axis=1)
data_city
```

Out[61]:

	City	Percentage
0	BROOKLYN	32.98
1	NEW YORK	22.14
2	BRONX	13.65
3	STATEN ISLAND	4.14
4	JAMAICA	2.45
5	ASTORIA	2.12
6	FLUSHING	2.00
7	RIDGEWOOD	1.73
8	CORONA	1.44
9	WOODSIDE	1.19

In [62]:

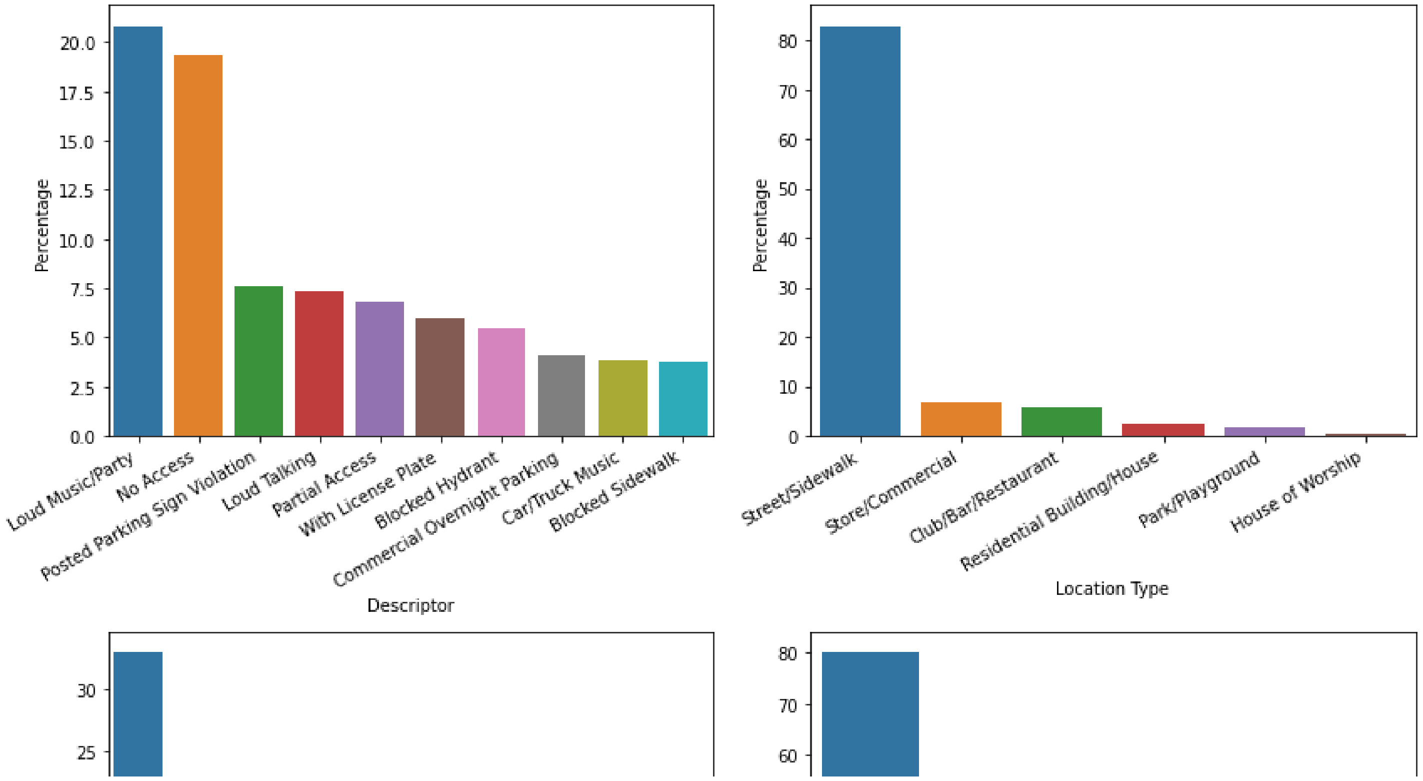
```
# Applying the above procedure for Address Type

data_address_type = np.around(((nyc_modf['Address Type'].value_counts()*100) / nyc_modf[
'Address Type'].value_counts().sum()),
                                decimals=2)
data_address_type = data_address_type.to_frame()
data_address_type = data_address_type.rename(columns={'Address Type':'Percentage'})
data_address_type['Address Type'] = data_address_type.index
cols = data_address_type.columns.tolist()
cols = cols[-1:]+cols[:-1]
data_address_type = data_address_type[cols]
#data_address_type = data_address_type[(data_address_type.Percentage) >= 1.0]
data_address_type = data_address_type.reset_index()
data_address_type = data_address_type.drop(columns=['index'],axis=1)
data_address_type
```

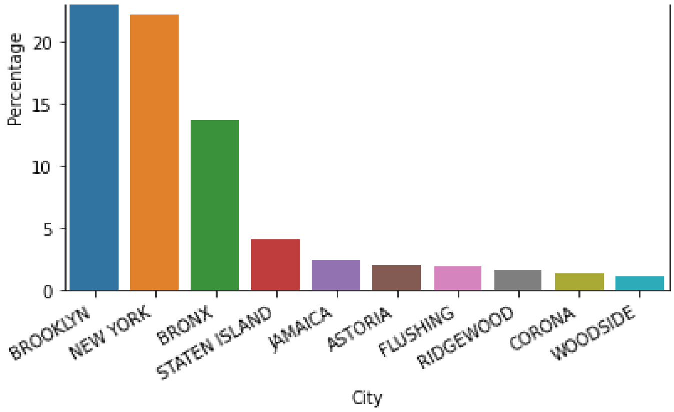
Out[62]:

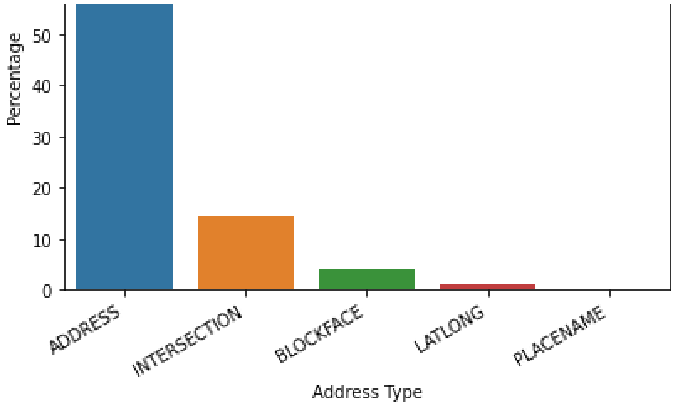
	Address Type	Percentage
0	ADDRESS	80.11
1	INTERSECTION	14.56
2	BLOCKFACE	4.03
3	LATLONG	1.18
4	PLACENAME	0.12

In [63]:

```
fig, ax = plt.subplots(2, 2, figsize=(12, 10))

#sns.set_theme(style="whitegrid")
#plt.suptitle("Proportion of different outcomes for few interesting features.")

descriptor = sns.barplot(ax=ax[0,0],x=data_descriptor.Descriptor,y=data_descriptor.Perce
ntage,)
descriptor.set_xticklabels(descriptor.get_xticklabels(), rotation=30, ha="right")

location_type = sns.barplot(ax=ax[0,1],x=data_location_type['Location Type'],y=data_loca
tion_type.Percentage,)
location_type.set_xticklabels(location_type.get_xticklabels(), rotation=30, ha="right")

city = sns.barplot(ax=ax[1,0],x=data_city['City'],y=data_city.Percentage,)
city.set_xticklabels(city.get_xticklabels(), rotation=30, ha="right")

address = sns.barplot(ax=ax[1,1],x=data_address_type['Address Type'],y=data_address_type
.Percentage,)
address.set_xticklabels(address.get_xticklabels(), rotation=30, ha="right")


#plt.subplots_adjust(left=None, bottom=None, right=None, top=0.0, wspace=None, hspace=Non
e)
plt.tight_layout()
```







So it is obvious that the Loud Music/party causes the biggest problem for the citizens. And it seems most complaints occur at Street/Sidewalk. And 'Brooklyn' faces the largest problems among all other cities. However, we have mostly solid information. The place where the problem occurs is pinpointed (Proper Address).

These observations are very preliminary. One can expect or guess the outcomes from these visualizations, regarding the corresponding features. However, it needs to be realized that we can not infer/predict from here without any proper statistical explanation.

Now, let's convert the time data ('timedelta64') into integer and store them (converting into hours) in a new column. Besides that let us cut the ambiguous data.

In [67]:

```
data_place_CType_RCTime = nyc_modf[['City','Complaint Type','Request_Closing_Time']]
data_place_CType_RCTime.dropna(subset = ['City','Complaint Type','Request_Closing_Time']
, inplace = True)
data_place_CType_RCTime['DeltaT(in_hr.)'] = np.around( (data_place_CType_RCTime['Request_
Closing_Time'].astype(np.int64)/
                                                                 (pow(10,9)*3600) ), decimals=2)
neg_time = data_place_CType_RCTime[data_place_CType_RCTime['DeltaT(in_hr.)'] < 0].sum()
print('The no negative time difference (Created Time > Clossing Time, which is not possib
le) = \n',neg_time)
#data_place_CType_RCTime['DeltaT(in sec)/Avg.'] = np.around((data_place_CType_RCTime['Del
taT(in sec)']/Avarage_time),decimals=1)
data_place_CType_RCTime.head(6)
```

```
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/1810123881.py:3: FutureW
arning: casting timedelta64[ns] values to int64 with .astype(...) is deprecated and will
raise in a future version. Use .view(...) instead.
  data_place_CType_RCTime['DeltaT(in_hr.)'] = np.around( (data_place_CType_RCTime['Reques
t_Closing_Time'].astype(np.int64)/
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/1810123881.py:3: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  data_place_CType_RCTime['DeltaT(in_hr.)'] = np.around( (data_place_CType_RCTime['Reques
t_Closing_Time'].astype(np.int64)/
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/1810123881.py in <module
>
      3 data_place_CType_RCTime['DeltaT(in_hr.)'] = np.around( (data_place_CType_RCTime['
Request_Closing_Time'].astype(np.int64)/
      4                                                                  (pow(10,9)*3600) ), decim
als=2)
----> 5 neg_time = data_place_CType_RCTime[data_place_CType_RCTime['DeltaT(in_hr.)'] < 0]
.sum()
      6 print('The no negative time difference (Created Time > Clossing Time, which is no
t possible) = \n',neg_time)
      7 #data_place_CType_RCTime['DeltaT(in sec)/Avg.'] = np.around((data_place_CType_RCT
ime['DeltaT(in sec)']/Avarage_time),decimals=1)
```

```
/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in sum(self, axis, skip
na, level, numeric_only, min_count, **kwargs)
  10706                 **kwargs,
  10707             ):
> 10708                 return NDFrame.sum(
  10709                     self, axis, skipna, level, numeric_only, min_count, **kwargs
  10710                 )

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in sum(self, axis, skip
na, level, numeric_only, min_count, **kwargs)
  10444             **kwargs,
  10445         ):
> 10446             return self._min_count_stat_function(
  10447                 "sum", nanops.nansum, axis, skipna, level, numeric_only, min_count,
**kwargs
  10448             )

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in _min_count_stat_func
tion(self, name, func, axis, skipna, level, numeric_only, min_count, **kwargs)
  10426                     numeric_only=numeric_only,
  10427                 )
> 10428             return self._reduce(
  10429                 func,
  10430                 name=name,

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/frame.py in _reduce(self, op, name
, axis, skipna, numeric_only, filter_type, **kwds)
   9856                     # Even if we are object dtype, follow numpy and return
   9857                     #  float64, see test_apply_funcs_over_empty
-> 9858                     out = out.astype(np.float64)
   9859
   9860                 if numeric_only is None and out.shape[0] != df.shape[1]:

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in astype(self, dtype,
copy, errors)
   5813             else:
   5814                 # else, only a single dtype is given
-> 5815                 new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5816                 return self._constructor(new_data).__finalize__(self, method="astype
")
   5817

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/internals/managers.py in astype(se
lf, dtype, copy, errors)
    416
    417     def astype(self: T, dtype, copy: bool = False, errors: str = "raise") -> T:
--> 418         return self.apply("astype", dtype=dtype, copy=copy, errors=errors)
    419
    420     def convert(

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/internals/managers.py in apply(sel
f, f, align_keys, ignore_failures, **kwargs)
    325                     applied = b.apply(f, **kwargs)
    326                 else:
--> 327                     applied = getattr(b, f)(**kwargs)
    328             except (TypeError, NotImplementedError):
    329                 if not ignore_failures:

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/internals/blocks.py in astype(self
, dtype, copy, errors)
    589         values = self.values
    590
--> 591         new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    592
    593         new_values = maybe_coerce_values(new_values)

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/dtypes/cast.py in astype_array_saf
e(values, dtype, copy, errors)
   1307
   1308     try:
-> 1309         new_values = astype_array(values, dtype, copy=copy)
   1310     except (ValueError, TypeError):
```

```
   1311         # e.g. astype_nansafe can fail on object-dtype of strings

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/dtypes/cast.py in astype_array(val
ues, dtype, copy)
   1255
   1256     else:
-> 1257         values = astype_nansafe(values, dtype, copy=copy)
   1258
   1259     # in pandas we don't store numpy str dtypes, so convert to object

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/dtypes/cast.py in astype_nansafe(a
rr, dtype, copy, skipna)
   1199     if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
   1200         # Explicit copy, or required since NumPy can't view from / to object.
-> 1201         return arr.astype(dtype, copy=True)
   1202
   1203     return arr.astype(dtype, copy=copy)

TypeError: float() argument must be a string or a number, not 'Timedelta'
```

Let us calculate some statistical parameters, in order to draw a conclusion on the solution time taken so that we can group them into different categories depending on the time interval.

In [68]:

```
Avarage_time = np.around((data_place_CType_RCTime['DeltaT(in_hr.)'].mean()),decimals=2)
print('Avarage time gap between logging the complaint and problem solved = ',Avarage_time
, 'hour')
Central_val = np.around((data_place_CType_RCTime['DeltaT(in_hr.)'].median()),decimals=2)
print('Central value of the distribution = ',Central_val, 'hour')
Most_occoor = np.around((data_place_CType_RCTime['DeltaT(in_hr.)'].mode()),decimals=2)
print('Most occered value = ',Most_occoor, 'hour')
stand_dev = np.around((data_place_CType_RCTime['DeltaT(in_hr.)'].std()),decimals=2)
print('Deviation is = ',stand_dev)
```

```
Avarage time gap between logging the complaint and problem solved =  4.31 hour
Central value of the distribution =  2.71 hour
Most occered value =  0    0.88
dtype: float64 hour
Deviation is =  6.08
```

So, one can take the central value as the normal time taken to solve the problem/issue. However, as it is clear from the deviation that it spreads around 6 hr.(more than the central value) from the distribution, so it is more practical to choose average time as the normal time to solve the problem. And categorize time interval as per the codes written below.

In [69]:

```
conditions = [data_place_CType_RCTime['DeltaT(in_hr.)'] <= 0.5,
              (0.50 < data_place_CType_RCTime['DeltaT(in_hr.)']) & (data_place_CType_RCTi
me['DeltaT(in_hr.)'] <= 1.00),
              (1.00 < data_place_CType_RCTime['DeltaT(in_hr.)']) & (data_place_CType_RCTi
me['DeltaT(in_hr.)'] <= 2.00),
              (2.00 < data_place_CType_RCTime['DeltaT(in_hr.)']) & (data_place_CType_RCTi
me['DeltaT(in_hr.)'] <= 6.00),
              (6.00 < data_place_CType_RCTime['DeltaT(in_hr.)']) & (data_place_CType_RCTi
me['DeltaT(in_hr.)'] <= 10.00),
              (10.00 < data_place_CType_RCTime['DeltaT(in_hr.)'])]

choices = ['Super fast','Very fast','Fast','Normal','Slow','Super Slow']

data_place_CType_RCTime['Solution Status'] = np.select(conditions,choices)
```

```
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/3789049267.py:10: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
```

```
data_place_CType_RCTime['Solution Status'] = np.select(conditions,choices)
```

In [70]:

```
data_place_CType_RCTime.head(6)
```

Out[70]:

	City	Complaint Type	Request_Closing_Time	DeltaT(in_hr.)	Solution Status
0	NEW YORK	Noise - Street/Sidewalk	0 days 00:55:15	0.92	Very fast
1	ASTORIA	Blocked Driveway	0 days 01:26:16	1.44	Fast
2	BRONX	Blocked Driveway	0 days 04:51:31	4.86	Normal
3	BRONX	Illegal Parking	0 days 07:45:14	7.75	Slow
4	ELMHURST	Illegal Parking	0 days 03:27:02	3.45	Normal
5	BROOKLYN	Illegal Parking	0 days 01:53:30	1.89	Fast

In [71]:

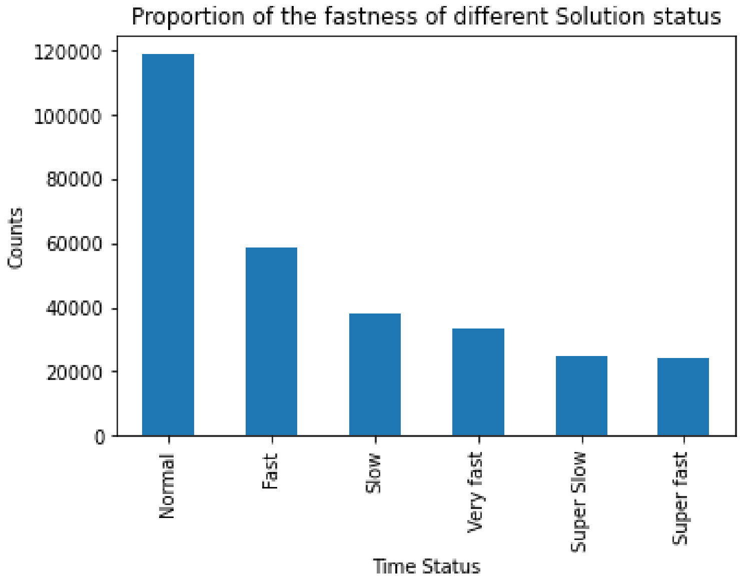
```
data_place_CType_RCTime['Solution Status'].value_counts()
```

Out[71]:

```
Normal         118955
Fast            58549
Slow            38068
Very fast       33459
Super Slow      24871
Super fast      24126
Name: Solution Status, dtype: int64
```

In [72]:

```
data_place_CType_RCTime['Solution Status'].value_counts().plot(kind='bar')
plt.xlabel('Time Status')
plt.ylabel('Counts')
plt.title('Proportion of the fastness of different Solution status')
plt.show()
plt.tight_layout()
```



```
<Figure size 432x288 with 0 Axes>
```

Based on the above-discussed approximation, the proportion of the time interval (expressed in different groups/status) to solve the problem, is depicted here. And it is obvious that the 'Normal' status will dominant since the range is chosen around the average value.

Now, let's see, is there any pattern for lodging a complaint?

Does it depend on a particular day or is there any month where too much or fewer problems are recorded?

In [73]:

```
nyc_modf['Created Date'].head(5)
```

Out[73]:

```
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
Name: Created Date, dtype: datetime64[ns]
```

In [75]:

```
#DataFrame Contain Days and Months of Complaint date

Year_Month_Day = pd.to_datetime(nyc_modf['Created Date'].dt.date)
Month_Day = pd.DataFrame()
Month_Day['Date'] = pd.to_datetime(Year_Month_Day.dt.date)
Month_Day['Month'] = Year_Month_Day.dt.month
Month_Day['Day'] = Year_Month_Day.dt.day
Month_Day['Month Name'] = Month_Day['Month'].apply(lambda x: calendar.month_abbr[x])
Month_Day['Day No'] = Month_Day['Date'].dt.weekday
Month_Day['Day Name'] = Month_Day['Day No'].map({0:'Monday',1:'Tuesday',2:'Wednesday',3:
'Thursday',4:'Friday',
                                                 5:'Saturday',6:'Sunday'})
Month_Day.sample(20)
```

Out[75]:

	Date	Month	Day	Month Name	Day No	Day Name
258226	2015-05-11	5	11	May	0	Monday
16508	2015-12-14	12	14	Dec	0	Monday
265323	2015-05-05	5	5	May	1	Tuesday
99307	2015-09-26	9	26	Sep	5	Saturday
183753	2015-07-14	7	14	Jul	1	Tuesday
137458	2015-08-25	8	25	Aug	1	Tuesday
176109	2015-07-21	7	21	Jul	1	Tuesday
293592	2015-04-06	4	6	Apr	0	Monday
62360	2015-10-31	10	31	Oct	5	Saturday
64285	2015-10-29	10	29	Oct	3	Thursday
197564	2015-07-02	7	2	Jul	3	Thursday
194600	2015-07-04	7	4	Jul	5	Saturday
246036	2015-05-22	5	22	May	4	Friday
164039	2015-08-01	8	1	Aug	5	Saturday
134435	2015-08-27	8	27	Aug	3	Thursday
77349	2015-10-16	10	16	Oct	4	Friday
260060	2015-05-10	5	10	May	6	Sunday
233878	2015-06-01	6	1	Jun	0	Monday
53272	2015-11-08	11	8	Nov	6	Sunday
101657	2015-09-24	9	24	Sep	3	Thursday

In [76]:

```
Month_plot = Month_Day['Month Name'].value_counts()
Month_plot = Month_plot.to_frame()
Month_plot = Month_plot.rename(columns={'Month Name':'Counts'})
Month_plot
```

Out[76]:

	Counts
May	36437
Sep	35427
Jun	35315
Aug	34956
Jul	34888
Oct	32605
Nov	30773
Dec	30521
Apr	27305
Mar	2471

In [77]:

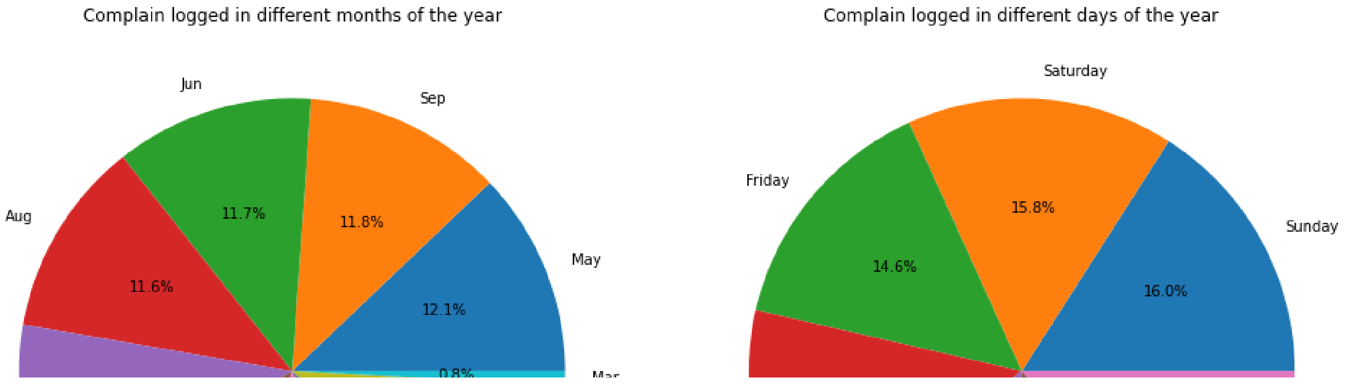
```
Day_plot = Month_Day['Day Name'].value_counts()
Day_plot = Day_plot.to_frame()
Day_plot = Day_plot.rename(columns={'Day Name':'Counts'})
Day_plot
```

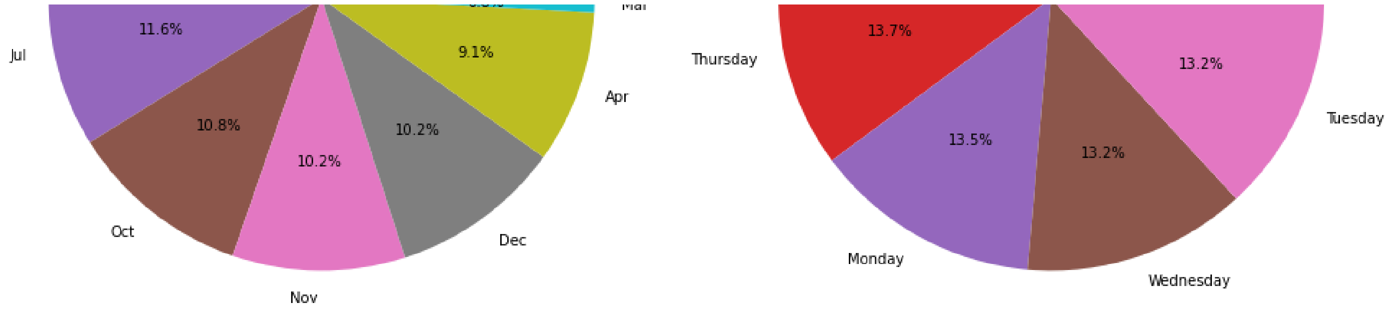
Out[77]:

	Counts
Sunday	47969
Saturday	47564
Friday	43995
Thursday	41342
Monday	40489
Wednesday	39788
Tuesday	39551

In [78]:

```
fig, axes = plt.subplots(1,2, figsize=(14,8))

axes[0].pie(Month_plot['Counts'], labels = Month_plot.index,autopct='%1.1f%%')
axes[0].set_title('Complain logged in different months of the year')

axes[1].pie(Day_plot['Counts'], labels = Day_plot.index,autopct='%1.1f%%')
axes[1].set_title('Complain logged in different days of the year')

plt.tight_layout()
```





So there is nothing abrupt for the months of lodging complaint. However, a very small amount of complaints recorded in the month of March.

The same observation can be made for the days. But if we look carefully, there is a small increment on the weekends compared to the weekly days.

However, looking at the days of a year might hide some extra information. It is better to check the days of each month of the year.

In [79]:

```
Month_Day_grouped = Month_Day.groupby(['Month Name','Day Name'],as_index=False)['Day No'
].count()
Month_Day_grouped_final = Month_Day_grouped.rename(columns={'Day No':'Counts'})
Month_Day_grouped_final.head(15)
```

Out[79]:

	Month Name	Day Name	Counts
0	Apr	Friday	3565
1	Apr	Monday	3222
2	Apr	Saturday	4227
3	Apr	Sunday	4069
4	Apr	Thursday	4323
5	Apr	Tuesday	3586
6	Apr	Wednesday	4313
7	Aug	Friday	4684
8	Aug	Monday	5042
9	Aug	Saturday	6913
10	Aug	Sunday	6293
11	Aug	Thursday	4198
12	Aug	Tuesday	3893
13	Aug	Wednesday	3933
14	Dec	Friday	4000

In [80]:

```
Month_Day[( (Month_Day['Month Name'] == 'Apr') & (Month_Day['Day Name'] == 'Monday') )].
count()
```

Out[80]:

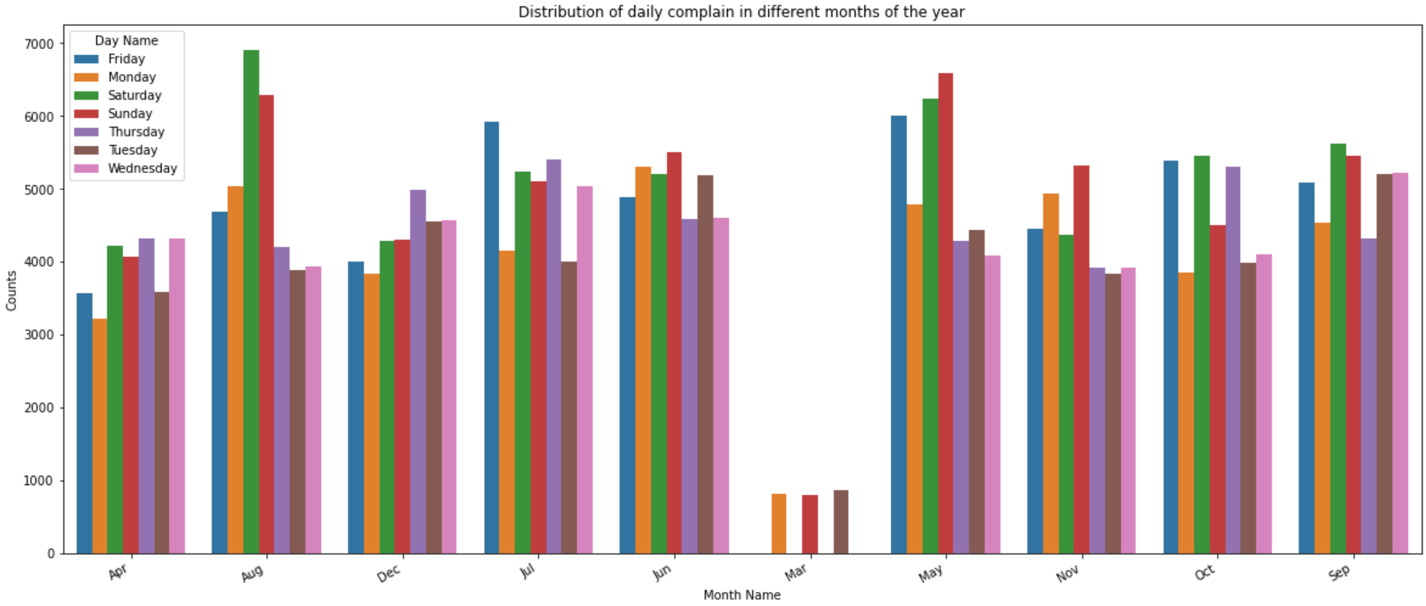
```
Date          3222
Month         3222
Day           3222
Month Name    3222
Day No        3222
Day Name      3222
dtype: int64
```

This is just to check whether the grouping operation is done correctly or not.

As you can see below, complaints created in each month for all seven days of the week are plotted. As we already counter that in March there is an abrupt decrement of complaint lodging compared to the other months. And Only three days of a week contributed here. It may contain seven days of the week, but with a very lesser amount. So let's check that to as well from the numbers.

In [81]:

```
plt.figure(figsize=(20,8))

month_day_plot = sns.barplot(x=Month_Day_grouped_final['Month Name'], y=Month_Day_groupe
d_final['Counts'],
                             hue=Month_Day_grouped_final['Day Name'], data=Month_Day_gro
uped_final)
month_day_plot.set_xticklabels(month_day_plot.get_xticklabels(), rotation=30, ha="right")
plt.title('Distribution of daily complain in different months of the year')
plt.show()
plt.tight_layout()
```



```
<Figure size 432x288 with 0 Axes>
```

In [82]:

```
Month_Day_grouped[Month_Day_grouped['Month Name'] == 'Mar']
```

Out[82]:

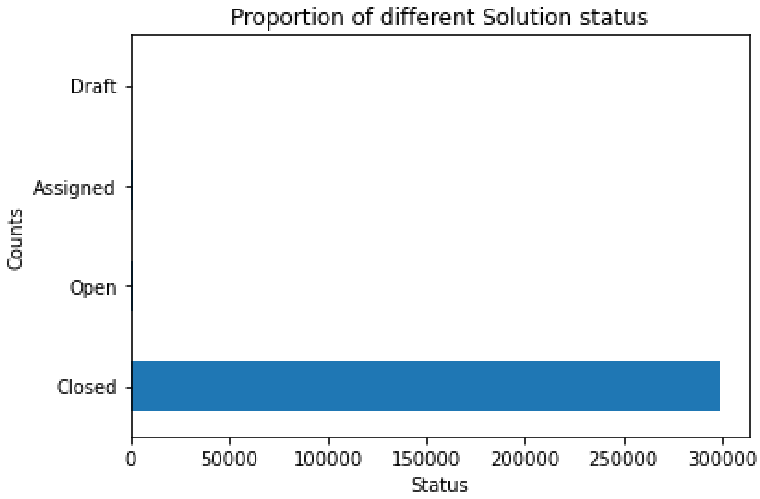
	Month Name	Day Name	Day No
35	Mar	Monday	807
36	Mar	Sunday	802
37	Mar	Tuesday	862

So complaints are recorded only in these three days of March.

And let's have a look quickly at the status of the complaints.

In [83]:

```
nyc_modf['Status'].value_counts().plot(kind='barh')
plt.xlabel('Status')
plt.ylabel('Counts')
plt.title('Proportion of different Solution status')
plt.show()
plt.tight_layout()
```



```
<Figure size 432x288 with 0 Axes>
```

1. **Order the complaint types based on the average ‘Request_Closing_Time’, grouping them for different locations Ordering the complaint types based on the average ‘Request_Closing_Time’ (converted into integer and kept in column 'DeltaT(in_hr.)') and grouping them for different locations (such as 'City').**

In [84]:

```
Complaint_City_AvgTime_grouped = data_place_CType_RCTime.groupby(['City','Complaint Type'
]).agg({'DeltaT(in_hr.)':'mean'})
Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.rename(
    columns={'DeltaT(in_hr.)':'Avg. Time(Given City, Complaint Type)'})
Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.transform('Avg. Time(Give
n City, Complaint Type)')
Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.to_frame()
Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.sort_values(
    ['City','Avg. Time(Given City, Complaint Type)'])

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
Complaint_City_AvgTime_grouped
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/67548920.py in <module>
      2 Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.rename(
      3     columns={'DeltaT(in_hr.)':'Avg. Time(Given City, Complaint Type)'})
----> 4 Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.transform('Avg. T
ime(Given City, Complaint Type)')
      5 Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.to_frame()
      6 Complaint_City_AvgTime_grouped = Complaint_City_AvgTime_grouped.sort_values(

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/frame.py in transform(self, func,
axis, *args, **kwargs)
   8578         op = frame_apply(self, func=func, axis=axis, args=args, kwargs=kwargs)
   8579         result = op.transform()
-> 8580         assert isinstance(result, DataFrame)
   8581         return result
   8582 

AssertionError: 
```

1. **Perform a statistical test for the following: (For the below statements you need to state the Null and Alternate and then provide a statistical test to accept or reject the Null Hypothesis along with the corresponding ‘p-value’.)**

Whether the average response time across complaint types is similar or not (overall) Are the type of complaint or service requested and location related?

In [85]:

```
import scipy.stats as stat
import scipy.stats as stat
```

Whether the average response time across complaint types is similar or not (overall)

In [86]:

```
# Average response time across complaint types

Complaint_AvgTime = data_place_CType_RCTime.groupby(['Complaint Type']).agg({'DeltaT(in_
hr.)':'mean'})
Complaint_AvgTime = pd.DataFrame(Complaint_AvgTime)
Complaint_AvgTime = Complaint_AvgTime.sort_values(['DeltaT(in_hr.)']).reset_index()
Complaint_AvgTime
```

Out[86]:

	Complaint Type	DeltaT(in_hr.)
0	Posting Advertisement	1.975926
1	Illegal Fireworks	2.761190
2	Noise - Commercial	3.136907
3	Noise - House of Worship	3.193240
4	Noise - Park	3.401706
5	Noise - Street/Sidewalk	3.438573
6	Traffic	3.446291
7	Disorderly Youth	3.558916
8	Noise - Vehicle	3.588570
9	Urinating in Public	3.626486
10	Bike/Roller/Skate Chronic	3.756611
11	Drinking	3.855354
12	Vending	4.013619
13	Squeegee	4.047500
14	Homeless Encampment	4.366029
15	Panhandling	4.372852
16	Illegal Parking	4.486005
17	Blocked Driveway	4.738187
18	Animal Abuse	5.213471
19	Graffiti	7.151062
20	Derelict Vehicle	7.346105
21	Animal in a Park	336.830000

1. **T-test (a) 1-sample T-test It is noteworthy that the value of the Avg. time due to complaint type 'Animal in a Park' quite out of the range. Let's find out the average with or without this particular complaint type.**

In [87]:

```
Tmean_without = float(Complaint_AvgTime[Complaint_AvgTime['Complaint Type']!='Animal in
a Park'].mean())
print("Without complaint type 'Animal in a Park' ----- ",Tmean_without)
Tmean_with = float(Complaint_AvgTime['DeltaT(in_hr.)'].mean())
print("With complaint type 'Animal in a Park' ----- ",Tmean_with)
```

```
Without complaint type 'Animal in a Park' -----  4.070219157949681
With complaint type 'Animal in a Park' -----  19.19566374167924
```

```
/var/folders/jb/zsxhv0ks3dxbqpkl2h4064xr0000gn/T/ipykernel_91036/2842791953.py:1: FutureW
arning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') i
s deprecated; in a future version this will raise TypeError.  Select only valid columns b
```

```
s deprecated, in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  Tmean_without = float(Complaint_AvgTime[Complaint_AvgTime['Complaint Type']!='Animal in
a Park'].mean())
```

With complaint type 'Animal in a Park'

In [88]:

```
ttest_with, pval_with = stat.ttest_1samp(Complaint_AvgTime['DeltaT(in_hr.)'], Tmean_with
)
print('T-statistic is =',ttest_with)
print('p value is =',np.around(pval_with))
```

```
T-statistic is = 0.0
p value is = 1.0
```

In [89]:

```
if (pval_with<0.05):
    print('Null hypothesis is rejected since p value ({}) is less than 0.05'.format(np.a
round(pval_with,decimals=2)))
else:
    print('Null hypothesis is accepted since p value ({}) is greater than 0.05'.format(n
p.around(pval_with,decimals=2)))
```

```
Null hypothesis is accepted since p value (1.0) is greater than 0.05
```

Without complaint type 'Animal in a Park'

In [90]:

```
Complaint_AvgTime_without = Complaint_AvgTime.drop([len(Complaint_AvgTime)-1],axis=0)
Complaint_AvgTime_without
```

Out[90]:

	Complaint Type	DeltaT(in_hr.)
0	Posting Advertisement	1.975926
1	Illegal Fireworks	2.761190
2	Noise - Commercial	3.136907
3	Noise - House of Worship	3.193240
4	Noise - Park	3.401706
5	Noise - Street/Sidewalk	3.438573
6	Traffic	3.446291
7	Disorderly Youth	3.558916
8	Noise - Vehicle	3.588570
9	Urinating in Public	3.626486
10	Bike/Roller/Skate Chronic	3.756611
11	Drinking	3.855354
12	Vending	4.013619
13	Squeegee	4.047500
14	Homeless Encampment	4.366029
15	Panhandling	4.372852
16	Illegal Parking	4.486005
17	Blocked Driveway	4.738187
18	Animal Abuse	5.213471

	Complaint Type	DeltaT(in_hr.)
19	Graffiti	7.151062
20	Derelict Vehicle	7.346105

In [91]:

```
ttest_without, pval_without = stat.ttest_1samp(Complaint_AvgTime_without['DeltaT(in_hr.)'
], Tmean_without)
print('T-statistic is =',ttest_without)
print('p value is =',np.around(pval_without,decimals=8))
```

```
T-statistic is = 0.0
p value is = 1.0
```

In [92]:

```
if (pval_without<0.05):
    print('Null hypothesis is rejected since p value ({}) is less than 0.05'.format(np.a
round(pval_without,decimals=2)))
else:
    print('Null hypothesis is accepted since p value ({}) is greater than 0.05'.format(n
p.around(pval_without,decimals=2)))
```

```
Null hypothesis is accepted since p value (1.0) is greater than 0.05
```

With or without the Hypothesis remain the same.

(b) 2-sample T-test

In [93]:

```
sample1 = Complaint_AvgTime.sample(frac=.5)
sample1
```

Out[93]:

	Complaint Type	DeltaT(in_hr.)
12	Vending	4.013619
18	Animal Abuse	5.213471
20	Derelict Vehicle	7.346105
9	Urinating in Public	3.626486
13	Squeegee	4.047500
21	Animal in a Park	336.830000
15	Panhandling	4.372852
0	Posting Advertisement	1.975926
16	Illegal Parking	4.486005
8	Noise - Vehicle	3.588570
1	Illegal Fireworks	2.761190

In [94]:

```
sample2 = Complaint_AvgTime.drop(sample1.index)
sample2
```

Out[94]:

	Complaint Type	DeltaT(in_hr.)
2	Noise - Commercial	3.136907
3	Noise - House of Worship	3.193240
4	Noise - Park	3.401706

	Complaint Type	DeltaT(in_hr.)
5	Noise - Street/Sidewalk	3.438573
6	Traffic	3.446291
7	Disorderly Youth	3.558916
10	Bike/Roller/Skate Chronic	3.756611
11	Drinking	3.855354
14	Homeless Encampment	4.366029
17	Blocked Driveway	4.738187
19	Graffiti	7.151062

In [95]:

```
print('Mean of 1st sample =',np.around(float(sample1['DeltaT(in_hr.)'].mean()),decimals=2
))
print('Standard dev. of 1st sample =',np.around(float(sample1['DeltaT(in_hr.)'].std()),de
cimals=2))
print('Mean of 2nd sample =',np.around(float(sample2['DeltaT(in_hr.)'].mean()),decimals=2
))
print('Standard dev. of 2nd sample =',np.around(float(sample2['DeltaT(in_hr.)'].std()),de
cimals=2))
```

```
Mean of 1st sample = 34.39
Standard dev. of 1st sample = 100.32
Mean of 2nd sample = 4.0
Standard dev. of 2nd sample = 1.15
```

In [96]:

```
ttest_2sp, p_val = stat.ttest_ind(sample1['DeltaT(in_hr.)'],sample2['DeltaT(in_hr.)'])
print('T-statistic is =',ttest_2sp)
print('p value is =',np.around(p_val,decimals=2))
```

```
T-statistic is = 1.0044450853811713
p value is = 0.33
```

In [97]:

```
if (p_val<0.05):
    print('Null hypothesis is rejected since p value ({}) is less than 0.05'.format(np.a
round(p_val,decimals=2)))
else:
    print('Null hypothesis is accepted since p value ({}) is greater than 0.05'.format(n
p.around(p_val,decimals=2)))
```

```
Null hypothesis is accepted since p value (0.33) is greater than 0.05
```

2. One way F-test (ANOVA)

In [98]:

```
sample1_anova = Complaint_AvgTime.sample(frac=1/3)
sample1_anova
```

Out[98]:

	Complaint Type	DeltaT(in_hr.)
14	Homeless Encampment	4.366029
20	Derelict Vehicle	7.346105
10	Bike/Roller/Skate Chronic	3.756611
7	Disorderly Youth	3.558916
0	Posting Advertisement	1.975926
4	Noise - Park	3.401706

	Complaint Type	DeltaT(in_hr.)
18	Animal Abuse	5.213471

In [99]:

```
rest_data = Complaint_AvgTime.drop(sample1_anova.index)
rest_data
```

Out[99]:

	Complaint Type	DeltaT(in_hr.)
1	Illegal Fireworks	2.761190
2	Noise - Commercial	3.136907
3	Noise - House of Worship	3.193240
5	Noise - Street/Sidewalk	3.438573
6	Traffic	3.446291
8	Noise - Vehicle	3.588570
9	Urinating in Public	3.626486
11	Drinking	3.855354
12	Vending	4.013619
13	Squeegee	4.047500
15	Panhandling	4.372852
16	Illegal Parking	4.486005
17	Blocked Driveway	4.738187
19	Graffiti	7.151062
21	Animal in a Park	336.830000

In [100]:

```
sample2_anova = rest_data.sample(frac=1/2)
sample2_anova
```

Out[100]:

	Complaint Type	DeltaT(in_hr.)
6	Traffic	3.446291
12	Vending	4.013619
9	Urinating in Public	3.626486
1	Illegal Fireworks	2.761190
13	Squeegee	4.047500
21	Animal in a Park	336.830000
8	Noise - Vehicle	3.588570
11	Drinking	3.855354

In [101]:

```
sample3_anova = rest_data.drop(sample2_anova.index)
sample3_anova
```

Out[101]:

	Complaint Type	DeltaT(in_hr.)
2	Noise - Commercial	3.136907
3	Noise - House of Worship	3.193240

	Complaint Type	DeltaT(in hr.)
5	Noise - Street/Sidewalk	3.438573
15	Panhandling	4.372852
16	Illegal Parking	4.486005
17	Blocked Driveway	4.738187
19	Graffiti	7.151062

In [102]:

```
print('Mean of 1st sample =',np.around(float(sample1_anova['DeltaT(in_hr.)'].mean()),decimals=2))
print('Standard dev. of 1st sample =',np.around(float(sample1_anova['DeltaT(in_hr.)'].std()),decimals=2))
print('Mean of 2nd sample =',np.around(float(sample2_anova['DeltaT(in_hr.)'].mean()),decimals=2))
print('Standard dev. of 2nd sample =',np.around(float(sample2_anova['DeltaT(in_hr.)'].std()),decimals=2))
print('Mean of 3rd sample =',np.around(float(sample3_anova['DeltaT(in_hr.)'].mean()),decimals=2))
print('Standard dev. of 3rd sample =',np.around(float(sample3_anova['DeltaT(in_hr.)'].std()),decimals=2))
```

```
Mean of 1st sample = 4.23
Standard dev. of 1st sample = 1.69
Mean of 2nd sample = 45.27
Standard dev. of 2nd sample = 117.81
Mean of 3rd sample = 4.36
Standard dev. of 3rd sample = 1.39
```

(a) Shapiro-Wilk normality test for each data group

In [103]:

```
f_val,p_val = stat.shapiro(sample1_anova['DeltaT(in_hr.)'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
F-statistic is = 0.9378947615623474
p value is = 0.62
```

In [104]:

```
f_val,p_val = stat.shapiro(sample2_anova['DeltaT(in_hr.)'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
F-statistic is = 0.4217081069946289
p value is = 0.0
```

In [105]:

```
f_val,p_val = stat.shapiro(sample3_anova['DeltaT(in_hr.)'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
F-statistic is = 0.8336941003799438
p value is = 0.09
```

All p values are greater than 0.05 Fail to reject the null hypothesis Samples come from populations that follow normal distribution

(b) Levene variance test

In [106]:

```
f_val,p_val = stat.levene(sample1_anova['DeltaT(in_hr.)'],sample2_anova['DeltaT(in_hr.)']
,sample3_anova['DeltaT(in_hr.)'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
F-statistic is = 0.8337429975380234
p value is = 0.45
```

All p values are greater than 0.05 Fail to reject the null hypothesis Samples have same varience

(c) One way ANOVA

In [107]:

```
f_val,p_val = stat.f_oneway(sample1_anova['DeltaT(in_hr.)'],sample2_anova['DeltaT(in_hr.)
'],sample3_anova['DeltaT(in_hr.)'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
F-statistic is = 0.8355969483362269
p value is = 0.45
```

In [108]:

```
if (p_val<0.05):
    print('Null hypothesis is rejected since p value ({}) is less than 0.05'.format(np.a
round(p_val,decimals=2)))
else:
    print('Null hypothesis is accepted since p value ({}) is greater than 0.05'.format(n
p.around(p_val,decimals=2)))
```

```
Null hypothesis is accepted since p value (0.45) is greater than 0.05
```

(d) Again independent 2-sample T-test

We already checked the independent T-test for 2 samples. Let's do the same for 3 samples and check the consistency.

In [109]:

```
t_val,p_val = stat.ttest_ind(sample1_anova['DeltaT(in_hr.)'],sample2_anova['DeltaT(in_hr.
)'])
print('T-statistic for sample 1 and 2 is =',t_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
T-statistic for sample 1 and 2 is = -0.917197673967634
p value is = 0.38
```

In [110]:

```
t_val,p_val = stat.ttest_ind(sample1_anova['DeltaT(in_hr.)'],sample3_anova['DeltaT(in_hr.
)'])
print('T-statistic for sample 1 and 3 is =',t_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
T-statistic for sample 1 and 3 is = -0.15495022830245944
p value is = 0.88
```

In [111]:

```
t_val,p_val = stat.ttest_ind(sample2_anova['DeltaT(in_hr.)'],sample3_anova['DeltaT(in_hr.
)'])
print('T-statistic for sample 2 and 3 is =',t_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
T-statistic for sample 2 and 3 is = 0.9143562202678671
p value is = 0.38
```

All the cases p-value is greater than 0.05 Fail to reject the null hypothesis. All the tests (T-test, F-test) provide a common conclusion. That is we fail to reject the 'Null hypothesis'.

Null Hypothesis states - there is no significant relationship among the average response time across complaint types

Alternate Hypothesis states - there is a significant relationship among the average response time across complaint types

Thus we may conclude that there is no significant relationship among the average response time across complaint types or they are not similar types.

Are the type of complaint or service requested and location related?

In [113]:

```
print('Null data in Complaint Type =',nyc_modf['Complaint Type'].isnull().sum())
print('Null data in City =',nyc_modf['City'].isnull().sum())
```

```
Null data in Complaint Type = 0
Null data in City = 2614
```

In [ ]:

```
df_cc = nyc_modf[['Complaint Type','City']]
df_cc = df_cc.dropna()
#df_cc.isnull().sum()
#df_cc
```

In [114]:

```
City_Complaint = pd.crosstab(nyc_modf['Complaint Type'],nyc_modf['City'],margins=True, m
argins_name='Total')
#City_Complaint = pd.crosstab(df_cc['Complaint Type'],df_cc['City'])
City_Complaint.head(6)
```

Out[114]:

City	ARVERNE	ASTORIA	Astoria	BAYSIDE	BELLEROSE	BREEZY POINT	BRONX	BROOKLYN	CAMBRIA HEIGHTS	CENTRAL PARK
Complaint Type										
Animal Abuse	38	125	0	37	7	2	1415	2394	11	0
Animal in a Park	0	0	0	0	0	0	0	0	0	0
Bike/Roller/Skate Chronic	0	15	0	0	1	0	20	111	0	0
Blocked Driveway	35	2618	116	377	95	3	12755	28148	147	0
Derelict Vehicle	27	351	12	198	89	3	1953	5181	115	0
Disorderly Youth	2	3	0	1	2	0	63	72	0	0

6 rows × 54 columns

Applying the ANOVA for a few combinations and let's see how does it go?

In [115]:

```
print("For 'ARVERNE' and 'ASTORIA' pair -------")
f_val,p_val = stat.f_oneway(City_Complaint['ARVERNE'],City_Complaint['ASTORIA'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
For 'ARVERNE' and 'ASTORIA' pair -------
F-statistic is = 3.3097701947747975
```

```
F-statistic is = 3.3097761947747975
p value is = 0.08
```

In [116]:

```
print("For 'ARVERNE' and 'BROOKLYN' pair -------")
f_val,p_val = stat.f_oneway(City_Complaint['ARVERNE'],City_Complaint['BROOKLYN'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
For 'ARVERNE' and 'BROOKLYN' pair -------
F-statistic is = 3.716772993046823
p value is = 0.06
```

In [117]:

```
print("For 'HOLLIS' and 'JAMAICA' pair -------")
f_val,p_val = stat.f_oneway(City_Complaint['HOLLIS'],City_Complaint['JAMAICA'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
For 'HOLLIS' and 'JAMAICA' pair -------
F-statistic is = 2.666621070410633
p value is = 0.11
```

In [118]:

```
print("For 'MASPETH' and 'QUEENS' pair -------")
f_val,p_val = stat.f_oneway(City_Complaint['MASPETH'],City_Complaint['QUEENS'])
print('F-statistic is =',f_val)
print('p value is =',np.around(p_val,decimals=2))
```

```
For 'MASPETH' and 'QUEENS' pair -------
F-statistic is = 3.368313812374042
p value is = 0.07
```

We have seen a few of the pairs. And it seems p-value is around 0.05. This is a very insufficient number of pair checking. So, though it looks like 'neglecting Null Hypothesis', but we can not certain unless checking all pairs (53 C 2 combinations for 53 cities). Even for 21 complaint types, it is still 21 C 2 combinations.

It is more proper to use the chai square contingency test for such data structure. It gives us the correlation between different features (here different cities for a given complaint type).

Null Hypothesis states - there is no dependence or relation among the features Alternate Hypothesis states - there is a relation among the features

Chai square Contigency test

In [ ]:

```
chi2, p_val, df, exp_frq = stat.chi2_contingency(City_Complaint)
```

In [123]:

```
print('Chi square value =','chi2')
print('p-value is =',p_val)
```

```
Chi square value = chi2
p-value is = 0.07322672892915565
```

In [124]:

```
if (p_val<0.05):
    print('Null hypothesis is rejected since p value ({}) is less than 0.05'.format(np.a
round(p_val,decimals=2)))
else:
    print('Null hypothesis is accepted since p value ({}) is greater than 0.05'.format(n
p.around(p_val,decimals=2)))
```

```
Null hypothesis is accepted since p value (0.07) is greater than 0.05
```

Thus we may conclude that there is a relationship between the type of complaint or service requested and location.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: