Project 3

Python - Real Time Project

New York City 311 calls

Jayateertha M Tatti

joy.tat@gmail.com

22-Dec-2016

Analyze data set from New York City 311 calls

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.

Project goals:

- Import a 311 NYC service request
- Basic data exploratory analysis
  - Explore data
  - Find patterns
  - Display the complaint type and city together
- Find major complaint types
  - Find the top 10 complaint types
  - Plot a bar graph of count vs. complaint types
- Visualize the complaint types
  - Display the major complaint types and their count

Dataset file is 311_Service_Requests_for_2009.csv. File size is 876 MB.

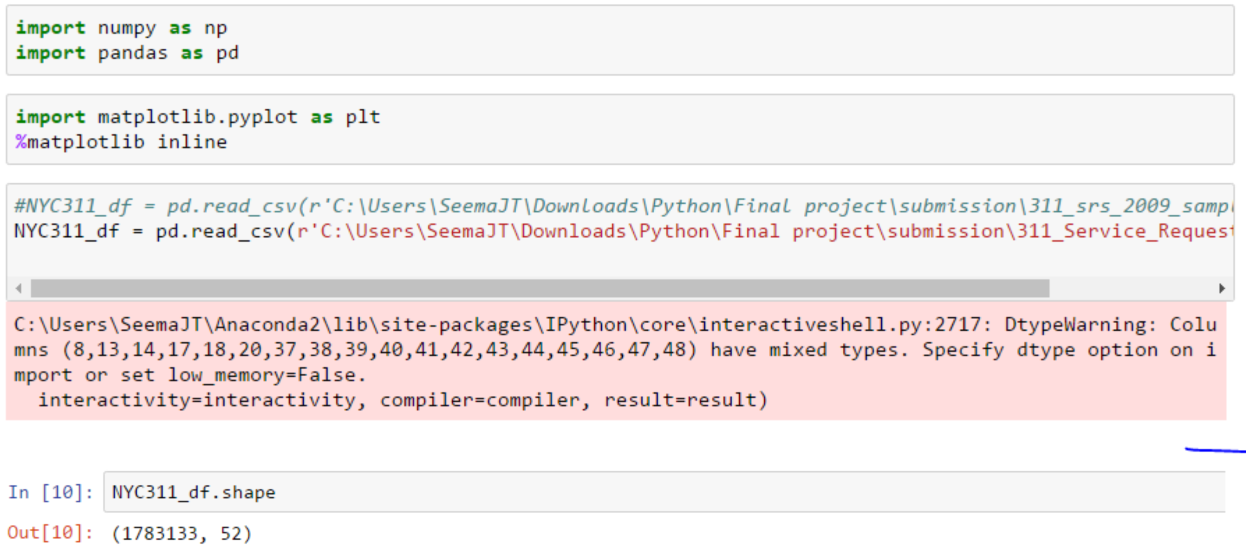
The data set contains the following attributes:

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

Dataset has observations **1783133** and variables **52**.
There are few observations for which close date is empty. Means those tickets are closed status. I will consider only closed tickets for analysis. After removing "open / pending" observations from dataset. Dataset has observations **1723802** and variables **52** to analyze. **59331** observations did not have the closed date value.

I have considered Complaint Type & City variables for analysis. There are 152 complaint types are present in the data set.

Source code details:



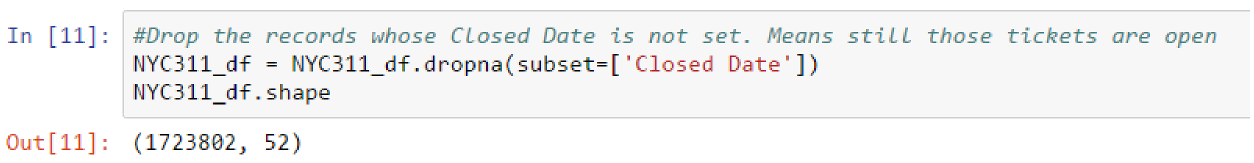
```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

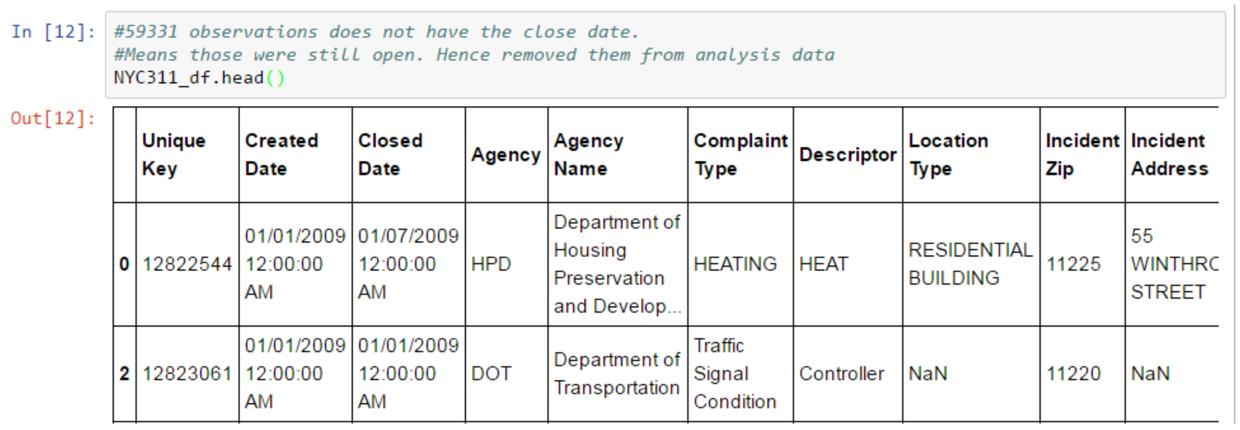
```
#NYC311_df = pd.read_csv(r'C:\Users\SeemaJT\Downloads\Python\Final project\submission\311_srs_2009_samp
NYC311_df = pd.read_csv(r'C:\Users\SeemaJT\Downloads\Python\Final project\submission\311_Service_Request
```

```
C:\Users\SeemaJT\Anaconda2\lib\site-packages\IPython\core\interactiveshell.py:2717: DtypeWarning: Colu
mns (8,13,14,17,18,20,37,38,39,40,41,42,43,44,45,46,47,48) have mixed types. Specify dtype option on i
mport or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
In [10]: NYC311_df.shape
Out[10]: (1783133, 52)
```

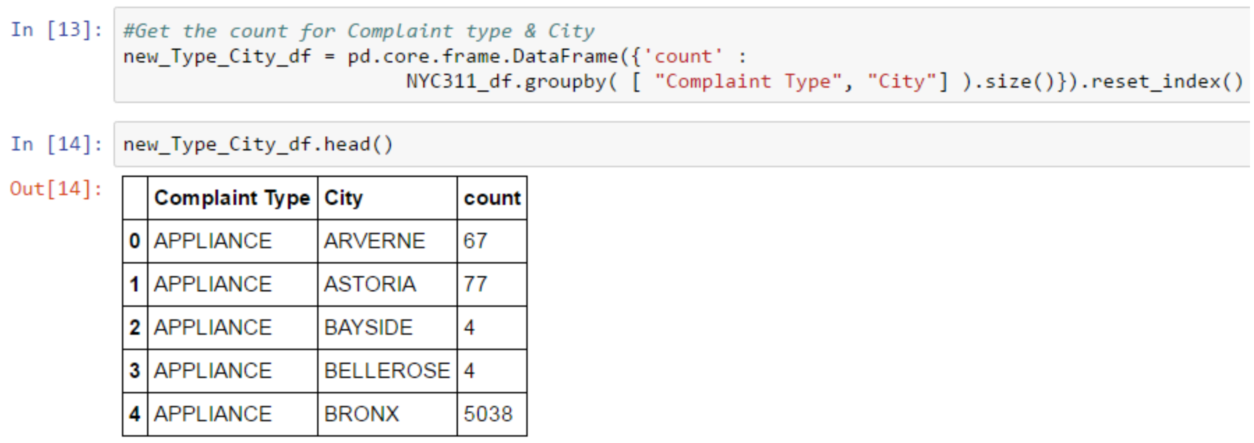


```
In [11]: #Drop the records whose Closed Date is not set. Means still those tickets are open
         NYC311_df = NYC311_df.dropna(subset=['Closed Date'])
         NYC311_df.shape

Out[11]: (1723802, 52)
```



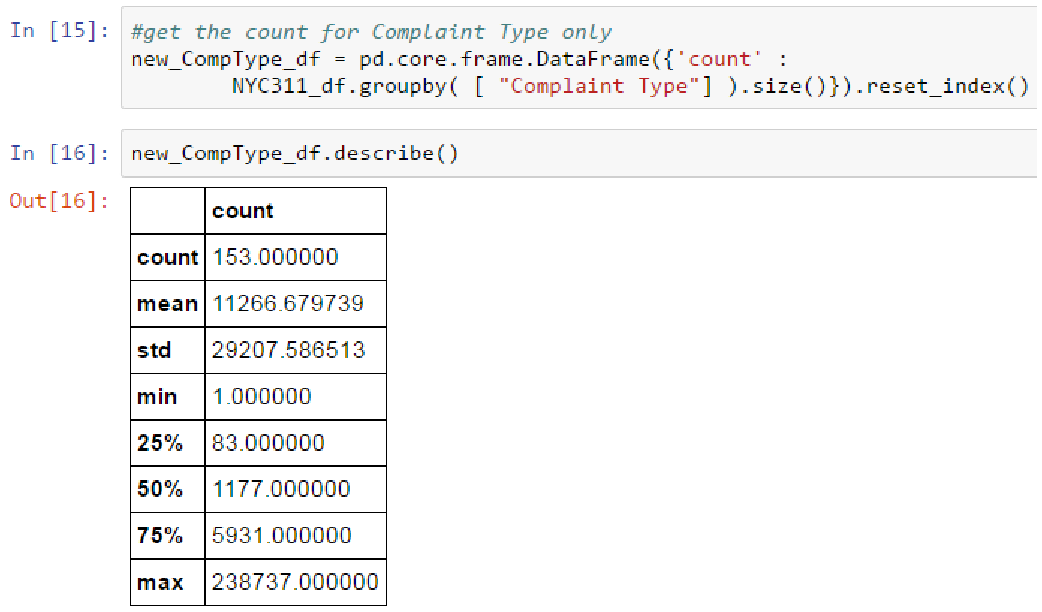
```
In [12]: #59331 observations does not have the close date.
         #Means those were still open. Hence removed them from analysis data
         NYC311_df.head()

Out[12]:
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address
0	12822544	01/01/2009 12:00:00 AM	01/07/2009 12:00:00 AM	HPD	Department of Housing Preservation and Develop...	HEATING	HEAT	RESIDENTIAL BUILDING	11225	55 WINTHRC STREET
2	12823061	01/01/2009 12:00:00 AM	01/01/2009 12:00:00 AM	DOT	Department of Transportation	Traffic Signal Condition	Controller	NaN	11220	NaN



```
In [13]: #Get the count for Complaint type & City
         new_Type_City_df = pd.core.frame.DataFrame({'count' :
                                     NYC311_df.groupby( [ "Complaint Type", "City"] ).size()}).reset_index()

In [14]: new_Type_City_df.head()

Out[14]:
```

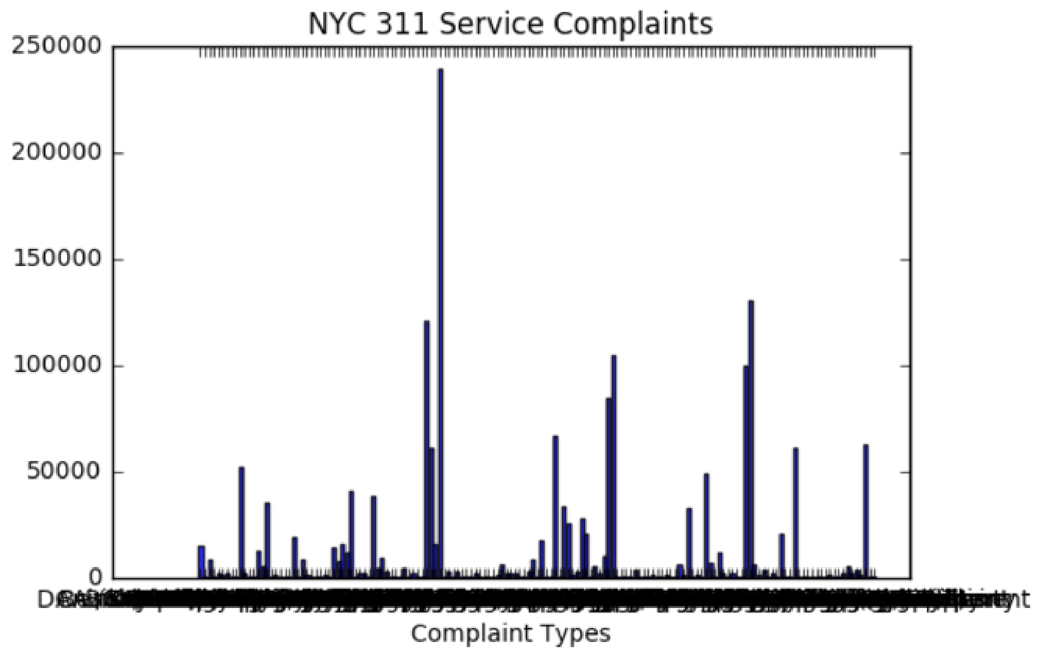
	Complaint Type	City	count
0	APPLIANCE	ARVERNE	67
1	APPLIANCE	ASTORIA	77
2	APPLIANCE	BAYSIDE	4
3	APPLIANCE	BELLEROSE	4
4	APPLIANCE	BRONX	5038



```
In [15]: #get the count for Complaint Type only
         new_CompType_df = pd.core.frame.DataFrame({'count' :
                   NYC311_df.groupby( [ "Complaint Type"] ).size()}).reset_index()

In [16]: new_CompType_df.describe()

Out[16]:
```

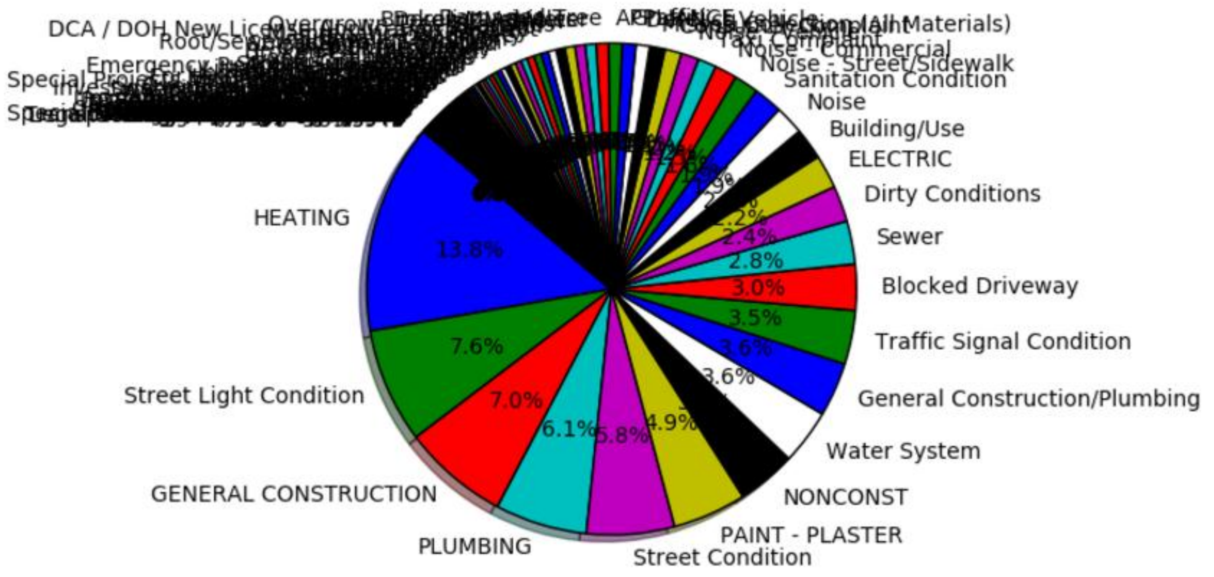
	count
count	153.000000
mean	11266.679739
std	29207.586513
min	1.000000
25%	83.000000
50%	1177.000000
75%	5931.000000
max	238737.000000

```
In [20]: y_pos = np.arange(len(new_CompType_df['Complaint Type']))
         plt.bar(y_pos, new_CompType_df['count'], align='center', alpha=0.8)
         plt.xticks(y_pos,  new_CompType_df['Complaint Type'])
         plt.xlabel('Complaint Types')
         plt.title('NYC 311 Service Complaints')

         plt.show()
```



Sort the data frame so that highest number of incidents are on top

```
In [21]: new_CompType_sorted_df = new_CompType_df.sort(['count'], ascending=[False])

         C:\Users\SeemaJT\Anaconda2\lib\site-packages\ipykernel\__main__.py:1: FutureWarning: sort(columns
         =....) is deprecated, use sort_values(by=.....)
           if __name__ == '__main__':

In [22]: new_CompType_sorted_df.shape

Out[22]: (153, 2)

In [24]: plt.pie(new_CompType_sorted_df['count'], labels=new_CompType_sorted_df['Complaint Type'], autopct='%1.1
                 shadow=True, startangle=140)
         plt.axis('equal')
         plt.show()
```

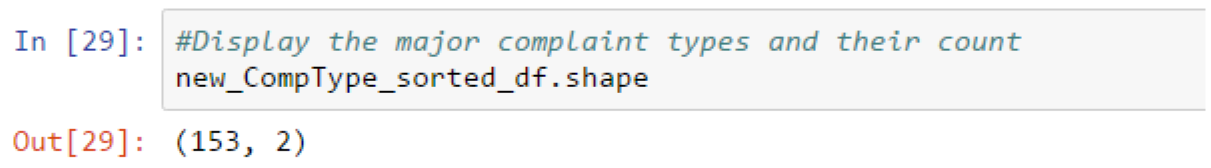


Sort the data frame so that highest number of incidents are on top

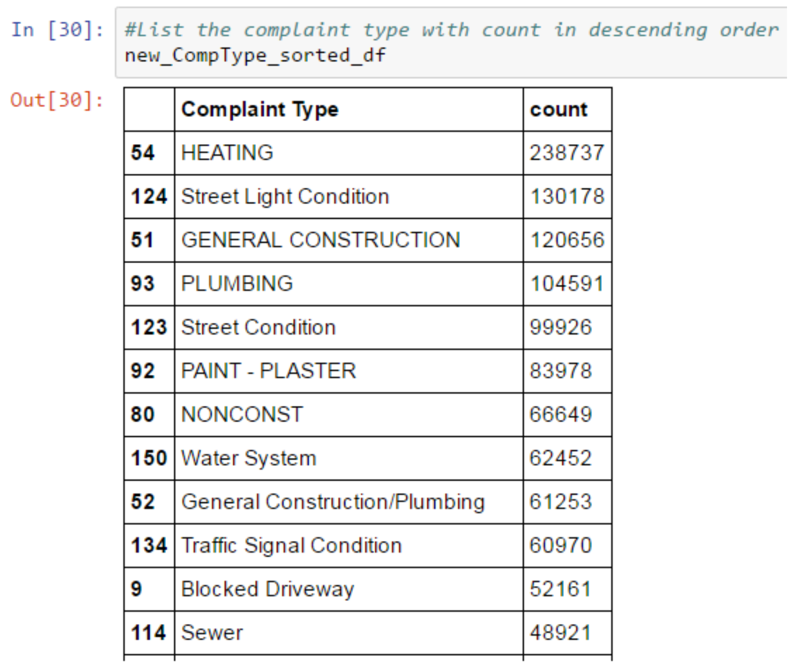
```
In [25]: #Display the complaint type and city together
         new_Type_City_sorted_df = new_Type_City_df.sort(['count'], ascending=[False])
         new_Type_City_sorted_df.shape

C:\Users\SeemaJT\Anaconda2\lib\site-packages\ipykernel\__main__.py:2: FutureWarning: sort(columns
=....) is deprecated, use sort_values(by=.....)
  from ipykernel import kernelapp as app

Out[25]: (5302, 3)
```

```
In [28]: #List the Complaint Type & City with count in descending order
         new_Type_City_sorted_df.head(10)

Out[28]:
```

	Complaint Type	City	count
2250	HEATING	BROOKLYN	79749
2249	HEATING	BRONX	76520
2272	HEATING	NEW YORK	48332
2114	GENERAL CONSTRUCTION	BROOKLYN	42524
2113	GENERAL CONSTRUCTION	BRONX	39114
3479	PLUMBING	BROOKLYN	36658
3478	PLUMBING	BRONX	33263
3433	PAINT - PLASTER	BRONX	28769
3434	PAINT - PLASTER	BROOKLYN	28437
4278	Street Condition	BROOKLYN	28392

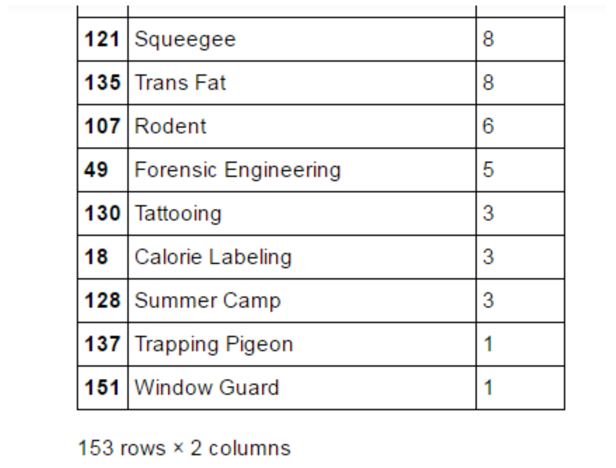


```
In [29]: #Display the major complaint types and their count
         new_CompType_sorted_df.shape

Out[29]: (153, 2)
```



```
In [30]: #List the complaint type with count in descending order
         new_CompType_sorted_df

Out[30]:
```

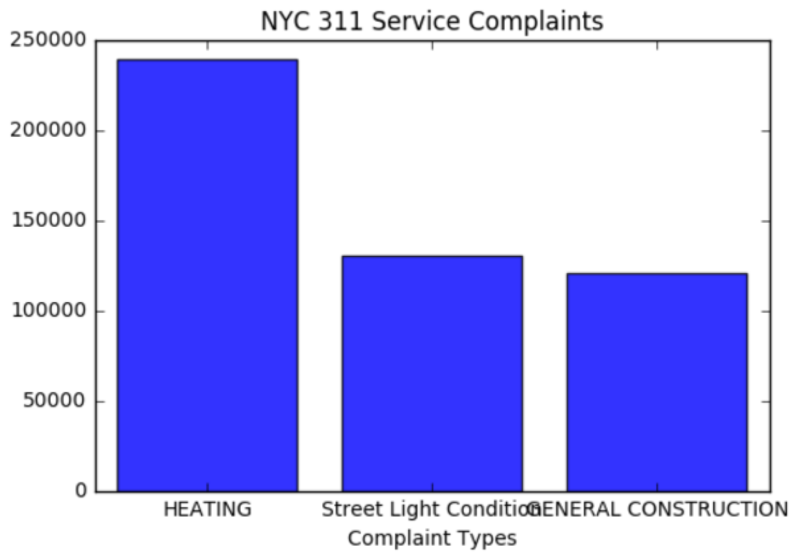
	Complaint Type	count
54	HEATING	238737
124	Street Light Condition	130178
51	GENERAL CONSTRUCTION	120656
93	PLUMBING	104591
123	Street Condition	99926
92	PAINT - PLASTER	83978
80	NONCONST	66649
150	Water System	62452
52	General Construction/Plumbing	61253
134	Traffic Signal Condition	60970
9	Blocked Driveway	52161
114	Sewer	48921



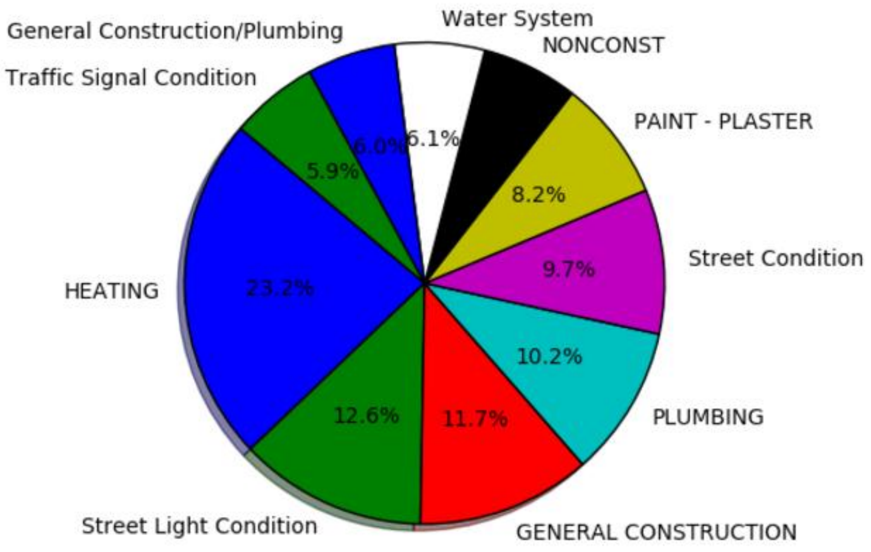
121	Squeegee	8
135	Trans Fat	8
107	Rodent	6
49	Forensic Engineering	5
130	Tattooing	3
18	Calorie Labeling	3
128	Summer Camp	3
137	Trapping Pigeon	1
151	Window Guard	1

153 rows × 2 columns

Top 3 complaint types



Top 10 complaint types


Water System
NONCONST
PAINT - PLASTER
Street Condition
PLUMBING
GENERAL CONSTRUCTION
Street Light Condition
HEATING
Traffic Signal Condition
General Construction/Plumbing