|  |  |
| --- | --- |
|  | Customer Service Requests Analysis |
|  |  |
|  | Margil Shah  DATA SCIENCE WITH PYTHON  11/2/20 |

Source Code – Full Project

Contents

[DESCRIPTION 2](#_Toc55823145)

[Background of Problem Statement 2](#_Toc55823146)

[Problem Objective 2](#_Toc55823147)

[Analysis Tasks to be performed 2](#_Toc55823148)

[Dataset Description 2](#_Toc55823149)

[Screen Shots: 5](#_Toc55823150)

[Analysis 1 (LOAD DATA) 5](#_Toc55823151)

[Analysis 2 5](#_Toc55823152)

[Analysis 3 5](#_Toc55823153)

[a. Insight 1 5](#_Toc55823154)

[b. Insight 2 6](#_Toc55823155)

[c. Insight 3 6](#_Toc55823156)

[d. Insight 4 7](#_Toc55823157)

[Analysis 4 7](#_Toc55823158)

[Analysis 5 8](#_Toc55823159)

[Source Code 9](#_Toc55823160)

# DESCRIPTION

Background of Problem Statement:

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

Problem Objective**:**

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

Analysis Tasks to be performed**:**

(Perform a service request data analysis of New York City 311 calls)

1. Import a 311 NYC service request.
2. Read or convert the columns ‘Created Date’ and Closed Date’ to datetime datatype and create a new column ‘Request\_Closing\_Time’ as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime)
3. Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining.
4. Order the complaint types based on the average ‘Request\_Closing\_Time’, grouping them for different locations.
5. Perform a statistical test for the following:

Please note: 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?

Dataset Description:

|  |  |
| --- | --- |
| **Field** | **Description** |
| Unique Key | (Plain text) - Unique identifier for the complaints |
| Created Date | (Date and Time) - The date and time on which the complaint is raised |
| Closed Date | (Date and Time)  - The date and time on which the complaint is closed |
| Agency | (Plain text) - Agency code |
| Agency Name | (Plain text) - Name of the agency |
| Complaint Type | (Plain text) - Type of the complaint |
| Descriptor | (Plain text) - Complaint type label (Heating - Heat, Traffic Signal Condition - Controller) |
| Location Type | (Plain text) - Type of the location (Residential, Restaurant, Bakery, etc) |
| Incident Zip | (Plain text) - Zip code for the location |
| Incident Address | (Plain text) - Address of the location |
| Street Name | (Plain text) - Name of the street |
| Cross Street 1 | (Plain text) - Detail of cross street |
| Cross Street 2 | (Plain text) - Detail of another cross street |
| Intersection Street 1 | (Plain text) - Detail of intersection street if any |
| Intersection Street 2 | (Plain text) - Detail of another intersection street if any |
| Address Type | (Plain text) - Categorical (Address or Intersection) |
| City | (Plain text) - City for the location |
| Landmark | (Plain text) - Empty field |
| Facility Type | (Plain text) - N/A |
| Status | (Plain text) - Categorical (Closed or Pending) |
| Due Date | (Date and Time) - Date and time for the pending complaints |
| Resolution Action Updated Date | (Date and Time) - Date and time when the resolution was provided |
| Community Board | (Plain text) - Categorical field (specifies the community board with its code) |
| Borough | (Plain text) - Categorical field (specifies the community board) |
| X Coordinate | (State Plane) (Number) |
| Y Coordinate | (State Plane) (Number) |
| Park Facility Name | (Plain text) - Unspecified |
| Park Borough | (Plain text) - Categorical (Unspecified, Queens, Brooklyn etc) |
| School Name | (Plain text) - Unspecified |
| School Number | (Plain text)  - Unspecified |
| School Region | (Plain text)  - Unspecified |
| School Code | (Plain text)  - Unspecified |
| School Phone Number | (Plain text)  - Unspecified |
| School Address | (Plain text)  - Unspecified |
| School City | (Plain text)  - Unspecified |
| School State | (Plain text)  - Unspecified |
| School Zip | (Plain text)  - Unspecified |
| School Not Found | (Plain text)  - Empty Field |
| School or Citywide Complaint | (Plain text)  - Empty Field |
| Vehicle Type | (Plain text)  - Empty Field |
| Taxi Company Borough | (Plain text)  - Empty Field |
| Taxi Pick Up Location | (Plain text)  - Empty Field |
| Bridge Highway Name | (Plain text)  - Empty Field |
| Bridge Highway Direction | (Plain text)  - Empty Field |
| Road Ramp | (Plain text)  - Empty Field |
| Bridge Highway Segment | (Plain text)  - Empty Field |
| Garage Lot Name | (Plain text)  - Empty Field |
| Ferry Direction | (Plain text)  - Empty Field |
| Ferry Terminal Name | (Plain text)  - Empty Field |
| Latitude | (Number) - Latitude of the location |
| Longitude | (Number) - Longitude of the location |
| Location | (Location) - Coordinates (Latitude, Longitude) |

# Screen Shots:

## Analysis 1 (LOAD DATA)

1. Import a 311 NYC service request.



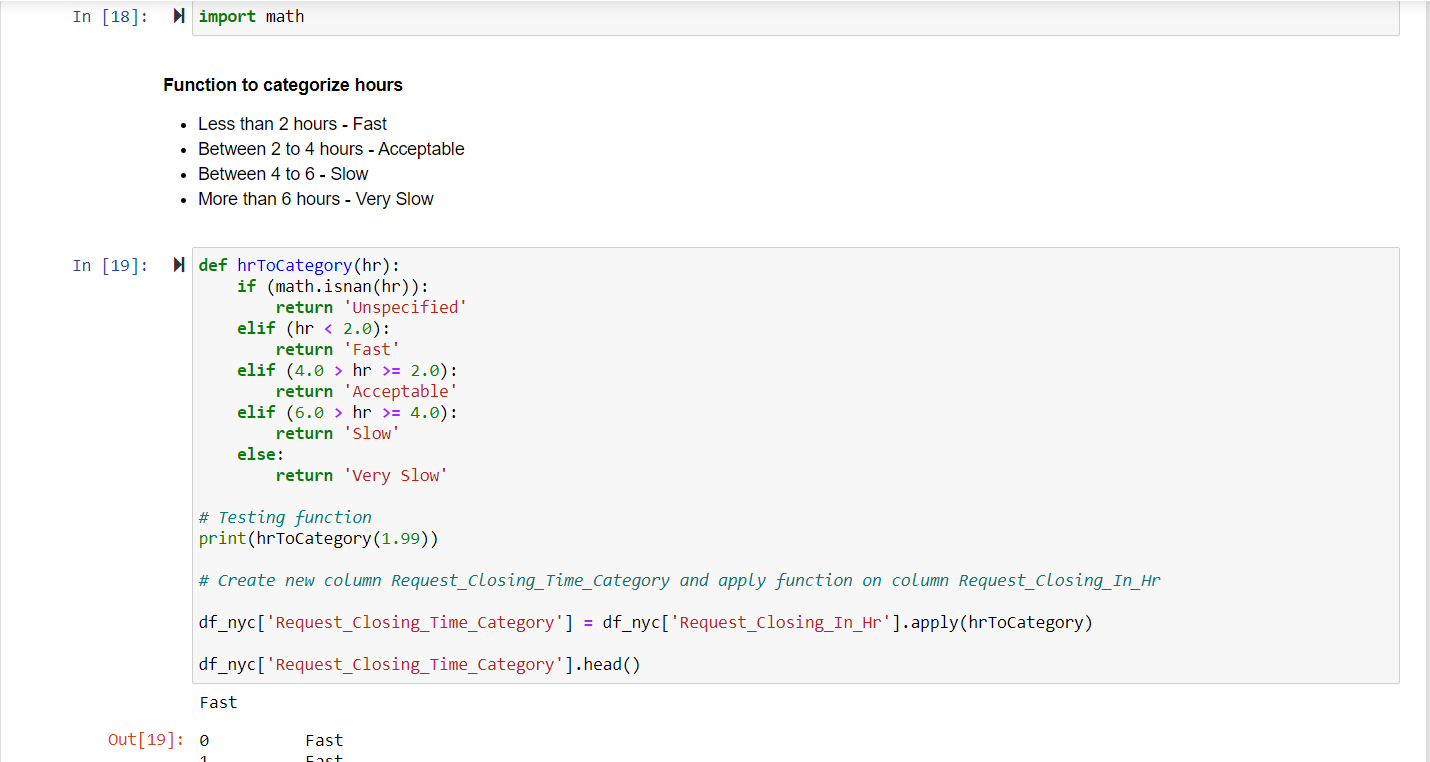
## Analysis 2

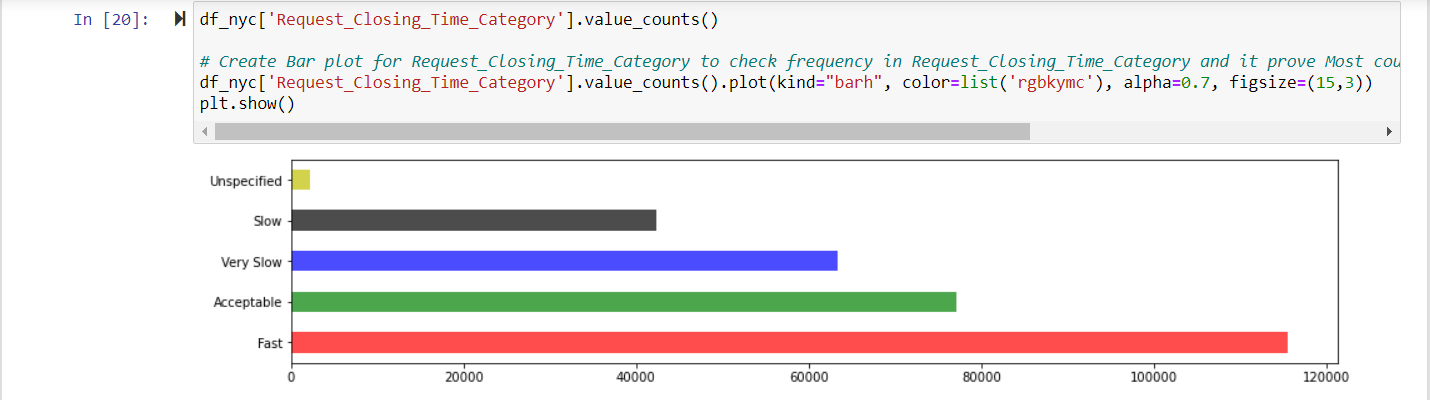
1. Read or convert the columns ‘Created Date’ and Closed Date’ to datetime datatype and create a new column ‘Request\_Closing\_Time’ as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime).



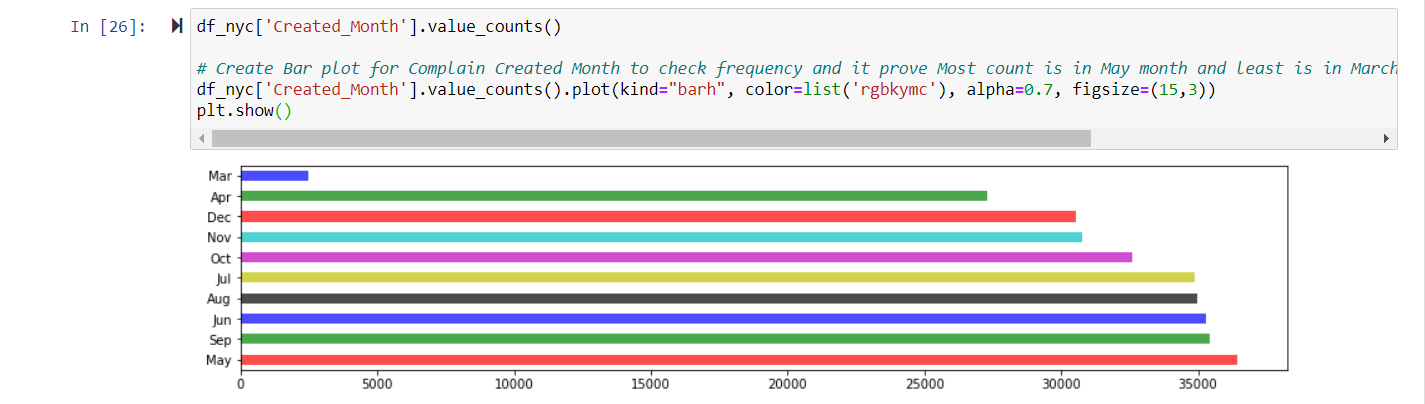
Analysis 3

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.
   1. Insight 1 - Categorize Request\_Closing\_Time as follows –
      1. Below 2 hours - Fast, Between 2 to 4 hours - Acceptable, Between 4 to 6 - Slow, More than 6 hours - Very Slow
      2. For this, first will create new column Request\_Closing\_In\_Hr and then create new column - Request\_Closing\_Time\_Category

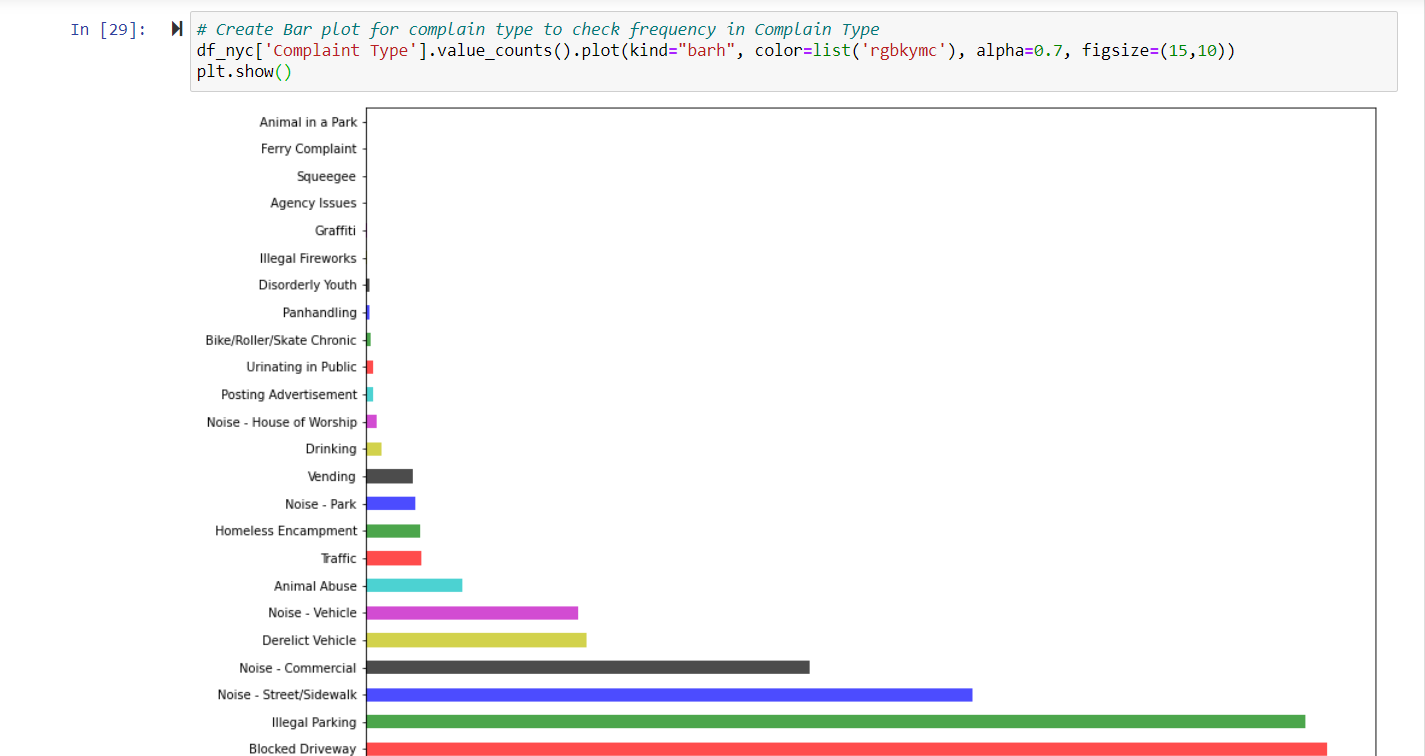




* 1. Insight 2 - To check with Month have Complain creation most and least



* 1. Insight 3: Check count in each complain type - sorted decreasing order.

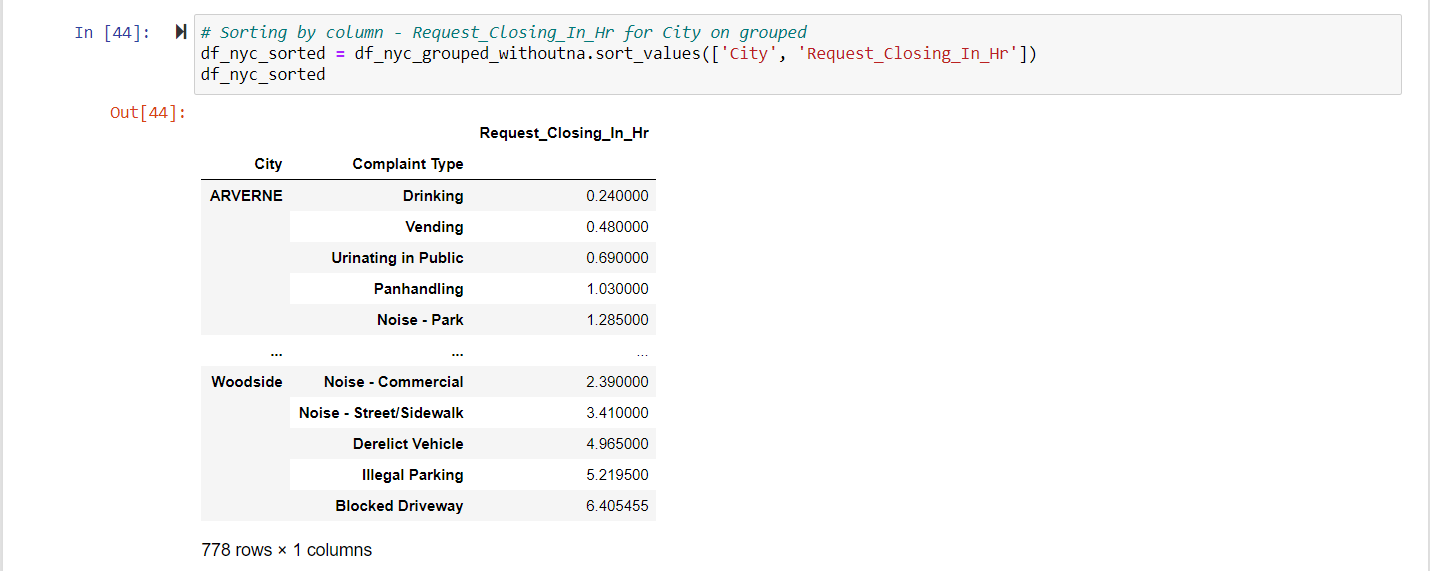


* 1. Insight 4: Let's check count for status type



## Analysis 4

1. Order the complaint types based on the average ‘Request\_Closing\_Time’, grouping them for different locations.

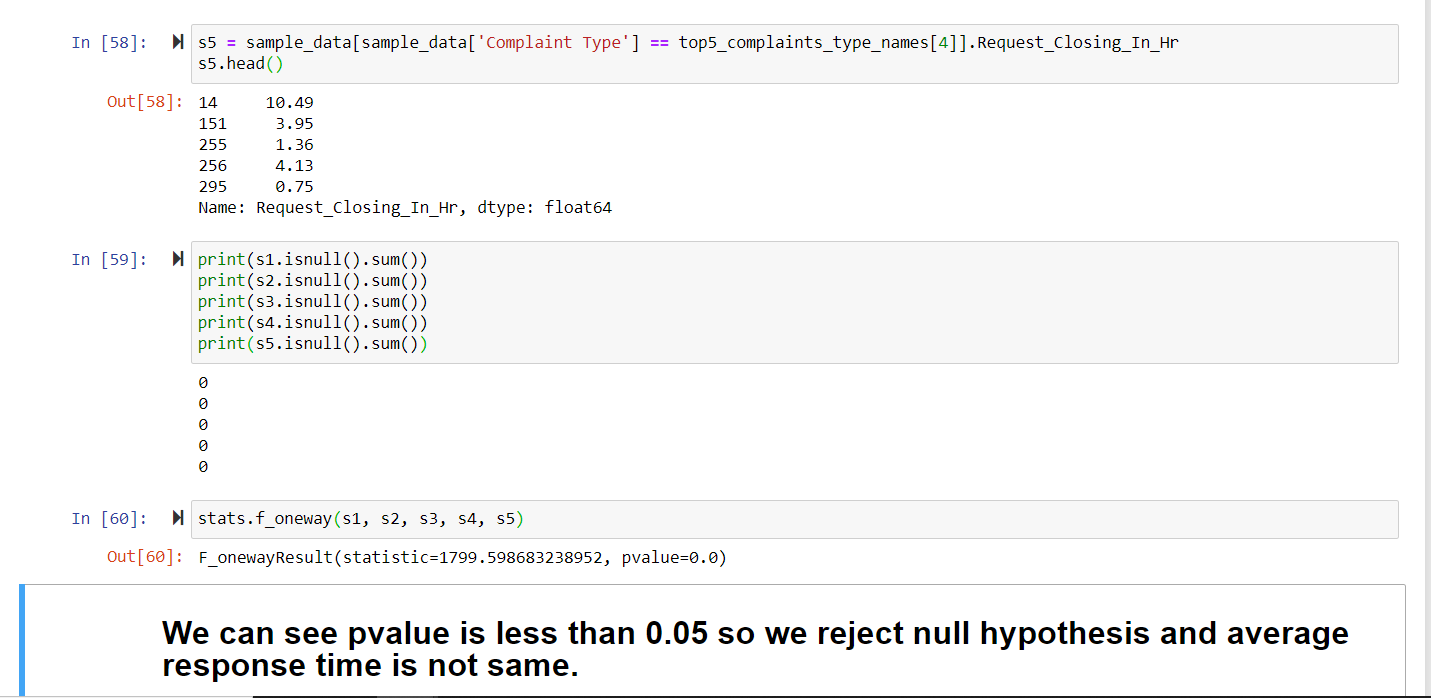


## Analysis 5

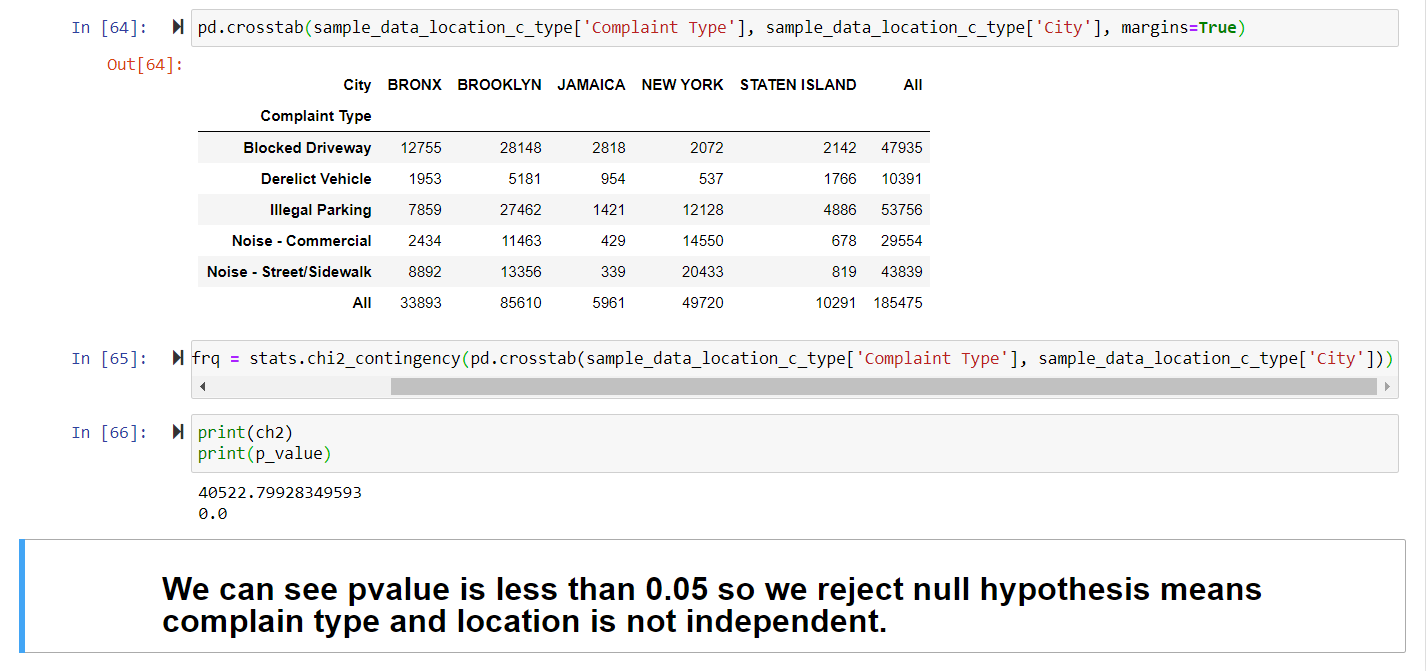
1. Perform a statistical test for the following:

Please note: 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?



# Source Code

DESCRIPTION

Background of Problem Statement:

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

Problem Objective:

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

Domain: Customer Service

Analysis Tasks to be performed: (Perform a service request data analysis of New York City 311 calls)

Import Required libraries

In [1]:

**import** **numpy** **as** **np** *# linear algebra*

**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*

**import** **os**

In [2]:

*#In Jupyter directory, created folder named - Projects and can see content of this folder as follows*

print(os.listdir("."))

['.ipynb\_checkpoints', '311\_Service\_Requests\_from\_2010\_to\_Present.csv', 'Data Science with Python Two.zip', 'Project 1 Customer Service Requests Analysis - ScreenShots.docx', 'Project 1 Customer Service Requests Analysis - Source Code.docx', 'Project 1 Customer Service Requests Analysis - Writeup.docx', 'Project 1 Customer Service Requests Analysis.ipynb', '~$oject 1 Customer Service Requests Analysis - Source Code.docx']

In [3]:

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

%**matplotlib** inline

1. Import a 311 NYC service request.

In [4]:

*# Solution 1*

*# Read csv*

df\_nyc = pd.read\_csv("311\_Service\_Requests\_from\_2010\_to\_Present.csv")

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (48,49) have mixed types.Specify dtype option on import or set low\_memory=False.

has\_raised = await self.run\_ast\_nodes(code\_ast.body, cell\_name,

In [5]:

*#Original Data*

df\_orig = pd.read\_csv("311\_Service\_Requests\_from\_2010\_to\_Present.csv")

2. Read or convert the columns ‘Created Date’ and Closed Date’ to datetime datatype and create a new column ‘Request\_Closing\_Time’ as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime)

In [6]:

*# Check shape of DataFrame*

df\_nyc.shape

Out[6]:

(300698, 53)

In [7]:

*# See columns*

df\_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 [8]:

*# First we should check which column has how many missing values*

df\_nyc.isnull().sum()

Out[8]:

Unique Key 0

Created Date 0

Closed Date 2164

Agency 0

Agency Name 0

Complaint Type 0

Descriptor 5914

Location Type 131

Incident Zip 2615

Incident Address 44410

Street Name 44410

Cross Street 1 49279

Cross Street 2 49779

Intersection Street 1 256840

Intersection Street 2 257336

Address Type 2815

City 2614

Landmark 300349

Facility Type 2171

Status 0

Due Date 3

Resolution Description 0

Resolution Action Updated Date 2187

Community Board 0

Borough 0

X Coordinate (State Plane) 3540

Y Coordinate (State Plane) 3540

Park Facility Name 0

Park Borough 0

School Name 0

School Number 0

School Region 1

School Code 1

School Phone Number 0

School Address 0

School City 0

School State 0

School Zip 1

School Not Found 0

School or Citywide Complaint 300698

Vehicle Type 300698

Taxi Company Borough 300698

Taxi Pick Up Location 300698

Bridge Highway Name 300455

Bridge Highway Direction 300455

Road Ramp 300485

Bridge Highway Segment 300485

Garage Lot Name 300698

Ferry Direction 300697

Ferry Terminal Name 300696

Latitude 3540

Longitude 3540

Location 3540

dtype: int64

In [9]:

*# As we seen Closed Date is important column and have many missing values*

df\_nyc[df\_nyc['Closed Date'].isnull()]

Out[9]:

|  | **Unique Key** | **Created Date** | **Closed Date** | **Agency** | **Agency Name** | **Complaint Type** | **Descriptor** | **Location Type** | **Incident Zip** | **Incident Address** | **...** | **Bridge Highway Name** | **Bridge Highway Direction** | **Road Ramp** | **Bridge Highway Segment** | **Garage Lot Name** | **Ferry Direction** | **Ferry Terminal Name** | **Latitude** | **Longitude** | **Location** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **416** | 32305700 | 12/31/2015 02:16:04 PM | NaN | NYPD | New York City Police Department | Illegal Parking | Posted Parking Sign Violation | Street/Sidewalk | NaN | 5426-5526 90TH ST | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **611** | 32309308 | 12/31/2015 09:58:06 AM | NaN | NYPD | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | NaN | 30 STREET | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1648** | 32303348 | 12/30/2015 05:13:42 AM | NaN | NYPD | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | NaN | 21600-2169 91ST AVE | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1816** | 32294519 | 12/29/2015 10:44:50 PM | NaN | NYPD | New York City Police Department | Derelict Vehicle | With License Plate | Street/Sidewalk | NaN | 127 STREET | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1965** | 32296487 | 12/29/2015 07:09:13 PM | NaN | NYPD | New York City Police Department | Derelict Vehicle | With License Plate | Street/Sidewalk | NaN | 5201-5299 68TH ST | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **300273** | 30287350 | 03/29/2015 02:40:19 PM | NaN | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | NaN | 3801-3999 23RD AVE | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **300492** | 30284963 | 03/29/2015 08:50:15 AM | NaN | NYPD | New York City Police Department | Vending | Unlicensed | Street/Sidewalk | NaN | COOPER AVE | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **300496** | 30285492 | 03/29/2015 08:44:13 AM | NaN | NYPD | New York City Police Department | Vending | Unlicensed | Street/Sidewalk | NaN | 80 STREET | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **300620** | 30282717 | 03/29/2015 01:55:35 AM | NaN | NYPD | New York City Police Department | Noise - Commercial | Loud Music/Party | Club/Bar/Restaurant | NaN | CRESCENT AVENUE | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **300693** | 30281872 | 03/29/2015 12:33:41 AM | NaN | NYPD | New York City Police Department | Noise - Commercial | Loud Music/Party | Club/Bar/Restaurant | NaN | CRESCENT AVENUE | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

2164 rows × 53 columns

In [10]:

*# We check data type of each column*

df\_nyc.dtypes

Out[10]:

Unique Key int64

Created Date object

Closed Date object

Agency object

Agency Name object

Complaint Type object

Descriptor object

Location Type object

Incident Zip float64

Incident Address object

Street Name object

Cross Street 1 object

Cross Street 2 object

Intersection Street 1 object

Intersection Street 2 object

Address Type object

City object

Landmark object

Facility Type object

Status object

Due Date object

Resolution Description object

Resolution Action Updated Date object

Community Board object

Borough object

X Coordinate (State Plane) float64

Y Coordinate (State Plane) float64

Park Facility Name object

Park Borough object

School Name object

School Number object

School Region object

School Code object

School Phone Number object

School Address object

School City object

School State object

School Zip object

School Not Found object

School or Citywide Complaint float64

Vehicle Type float64

Taxi Company Borough float64

Taxi Pick Up Location float64

Bridge Highway Name object

Bridge Highway Direction object

Road Ramp object

Bridge Highway Segment object

Garage Lot Name float64

Ferry Direction object

Ferry Terminal Name object

Latitude float64

Longitude float64

Location object

dtype: object

Solution 2

In [11]:

**import** **datetime** **as** **dt**

**import** **time**, **datetime**

In [12]:

*# Convert "Closed Date" to datetime dtype*

df\_nyc['Closed Date'] = pd.to\_datetime(df\_nyc['Closed Date'])

df\_nyc['Closed Date'].dtype

Out[12]:

dtype('<M8[ns]')

In [13]:

*# Convert "Created Date" to datetime dtype*

df\_nyc['Created Date'] = pd.to\_datetime(df\_nyc['Created Date'])

df\_nyc['Created Date'].dtype

Out[13]:

dtype('<M8[ns]')

In [14]:

*# Create new column Request\_Closing\_Time with time taken to close complain*

df\_nyc['Request\_Closing\_Time'] = df\_nyc['Closed Date'] - df\_nyc['Created Date']

df\_nyc['Request\_Closing\_Time'].head()

Out[14]:

0 00:55:15

1 01:26:16

2 04:51:31

3 07:45:14

4 03:27:02

Name: Request\_Closing\_Time, dtype: timedelta64[ns]

3. Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining.

Solution 3

* From here starting Insight
* Insight - 1 - Categorize Request\_Closing\_Time as follows -
* Below 2 hours - Fast, Between 2 to 4 hours - Acceptable, Between 4 to 6 - Slow, More than 6 hours - Very Slow
* For this, first will create new column Request\_Closing\_In\_Hr and then create new column - Request\_Closing\_Time\_Category

In [15]:

*# Function to convert TimeDelta in Hour*

**def** toHr(timeDel):

days = timeDel.days

hours = round(timeDel.seconds/3600, 2)

result = (days \* 24) + hours

**return** result

In [16]:

*# Testing of function with days*

test\_days = df\_nyc[df\_nyc['Unique Key'] == 32122264]['Request\_Closing\_Time']

print(toHr(test\_days[27704]))

print(test\_days[27704])

print(test\_days.dtype)

145.08

6 days 01:05:00

timedelta64[ns]

In [17]:

*# Apply this function to every row of column Request\_Closing\_Time*

df\_nyc['Request\_Closing\_In\_Hr'] = df\_nyc['Request\_Closing\_Time'].apply(toHr)

df\_nyc['Request\_Closing\_In\_Hr'].head()

Out[17]:

0 0.92

1 1.44

2 4.86

3 7.75

4 3.45

Name: Request\_Closing\_In\_Hr, dtype: float64

In [18]:

**import** **math**

#### Function to categorize hours

* Less than 2 hours - Fast
* Between 2 to 4 hours - Acceptable
* Between 4 to 6 - Slow
* More than 6 hours - Very Slow

In [19]:

**def** hrToCategory(hr):

**if** (math.isnan(hr)):

**return** 'Unspecified'

**elif** (hr < 2.0):

**return** 'Fast'

**elif** (4.0 > hr >= 2.0):

**return** 'Acceptable'

**elif** (6.0 > hr >= 4.0):

**return** 'Slow'

**else**:

**return** 'Very Slow'

*# Testing function*

print(hrToCategory(1.99))

*# Create new column Request\_Closing\_Time\_Category and apply function on column Request\_Closing\_In\_Hr*

df\_nyc['Request\_Closing\_Time\_Category'] = df\_nyc['Request\_Closing\_In\_Hr'].apply(hrToCategory)

df\_nyc['Request\_Closing\_Time\_Category'].head()

Fast

Out[19]:

0 Fast

1 Fast

2 Slow

3 Very Slow

4 Acceptable

Name: Request\_Closing\_Time\_Category, dtype: object

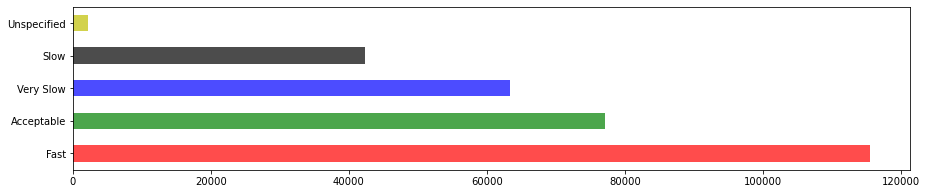
In [20]:

df\_nyc['Request\_Closing\_Time\_Category'].value\_counts()

*# Create Bar plot for Request\_Closing\_Time\_Category to check frequency in Request\_Closing\_Time\_Category and it prove Most count is in Fast category means closed less than 2 hours*

df\_nyc['Request\_Closing\_Time\_Category'].value\_counts().plot(kind="barh", color=list('rgbkymc'), alpha=0.7, figsize=(15,3))

plt.show()



In [21]:

df\_nyc.head()

Out[21]:

|  | **Unique Key** | **Created Date** | **Closed Date** | **Agency** | **Agency Name** | **Complaint Type** | **Descriptor** | **Location Type** | **Incident Zip** | **Incident Address** | **...** | **Bridge Highway Segment** | **Garage Lot Name** | **Ferry Direction** | **Ferry Terminal Name** | **Latitude** | **Longitude** | **Location** | **Request\_Closing\_Time** | **Request\_Closing\_In\_Hr** | **Request\_Closing\_Time\_Category** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 32310363 | 2015-12-31 23:59:45 | 2016-01-01 00:55:00 | NYPD | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | 10034.0 | 71 VERMILYEA AVENUE | ... | NaN | NaN | NaN | NaN | 40.865682 | -73.923501 | (40.86568153633767, -73.92350095571744) | 00:55:15 | 0.92 | Fast |
| **1** | 32309934 | 2015-12-31 23:59:44 | 2016-01-01 01:26:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 11105.0 | 27-07 23 AVENUE | ... | NaN | NaN | NaN | NaN | 40.775945 | -73.915094 | (40.775945312321085, -73.91509393898605) | 01:26:16 | 1.44 | Fast |
| **2** | 32309159 | 2015-12-31 23:59:29 | 2016-01-01 04:51:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 10458.0 | 2897 VALENTINE AVENUE | ... | NaN | NaN | NaN | NaN | 40.870325 | -73.888525 | (40.870324522111424, -73.88852464418646) | 04:51:31 | 4.86 | Slow |
| **3** | 32305098 | 2015-12-31 23:57:46 | 2016-01-01 07:43:00 | NYPD | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | 10461.0 | 2940 BAISLEY AVENUE | ... | NaN | NaN | NaN | NaN | 40.835994 | -73.828379 | (40.83599404683083, -73.82837939584206) | 07:45:14 | 7.75 | Very Slow |
| **4** | 32306529 | 2015-12-31 23:56:58 | 2016-01-01 03:24:00 | NYPD | New York City Police Department | Illegal Parking | Blocked Sidewalk | Street/Sidewalk | 11373.0 | 87-14 57 ROAD | ... | NaN | NaN | NaN | NaN | 40.733060 | -73.874170 | (40.733059618956815, -73.87416975810375) | 03:27:02 | 3.45 | Acceptable |

5 rows × 56 columns

In [22]:

*# Insight 2 - To check with Month have Complain creation most and least*

*# We will create one column with Create\_Month name*

*# Created Series for months in text format*

monthSeries = pd.Series({1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'})

print(monthSeries)

print(monthSeries[12])

1 Jan

2 Feb

3 Mar

4 Apr

5 May

6 Jun

7 Jul

8 Aug

9 Sep

10 Oct

11 Nov

12 Dec

dtype: object

Dec

In [23]:

df\_nyc['Created Date'].dtype

*# Function to fetch month from Created Date column*

**def** getMonth(cDate):

a = str(cDate)

datee = datetime.datetime.strptime(a, "%Y-%m-**%d** %H:%M:%S")

**return** monthSeries[datee.month]

*# Test function getMonth*

print(df\_nyc['Created Date'][0])

print(getMonth(df\_nyc['Created Date'][0]))

2015-12-31 23:59:45

Dec

In [24]:

*# Created new column Created\_Month and kept all text format months in that column*

df\_nyc['Created\_Month'] = df\_nyc['Created Date'].apply(getMonth)

df\_nyc['Created\_Month']

Out[24]:

0 Dec

1 Dec

2 Dec

3 Dec

4 Dec

...

300693 Mar

300694 Mar

300695 Mar

300696 Mar

300697 Mar

Name: Created\_Month, Length: 300698, dtype: object

In [25]:

df\_nyc.head()

Out[25]:

|  | **Unique Key** | **Created Date** | **Closed Date** | **Agency** | **Agency Name** | **Complaint Type** | **Descriptor** | **Location Type** | **Incident Zip** | **Incident Address** | **...** | **Garage Lot Name** | **Ferry Direction** | **Ferry Terminal Name** | **Latitude** | **Longitude** | **Location** | **Request\_Closing\_Time** | **Request\_Closing\_In\_Hr** | **Request\_Closing\_Time\_Category** | **Created\_Month** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 32310363 | 2015-12-31 23:59:45 | 2016-01-01 00:55:00 | NYPD | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | 10034.0 | 71 VERMILYEA AVENUE | ... | NaN | NaN | NaN | 40.865682 | -73.923501 | (40.86568153633767, -73.92350095571744) | 00:55:15 | 0.92 | Fast | Dec |
| **1** | 32309934 | 2015-12-31 23:59:44 | 2016-01-01 01:26:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 11105.0 | 27-07 23 AVENUE | ... | NaN | NaN | NaN | 40.775945 | -73.915094 | (40.775945312321085, -73.91509393898605) | 01:26:16 | 1.44 | Fast | Dec |
| **2** | 32309159 | 2015-12-31 23:59:29 | 2016-01-01 04:51:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 10458.0 | 2897 VALENTINE AVENUE | ... | NaN | NaN | NaN | 40.870325 | -73.888525 | (40.870324522111424, -73.88852464418646) | 04:51:31 | 4.86 | Slow | Dec |
| **3** | 32305098 | 2015-12-31 23:57:46 | 2016-01-01 07:43:00 | NYPD | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | 10461.0 | 2940 BAISLEY AVENUE | ... | NaN | NaN | NaN | 40.835994 | -73.828379 | (40.83599404683083, -73.82837939584206) | 07:45:14 | 7.75 | Very Slow | Dec |
| **4** | 32306529 | 2015-12-31 23:56:58 | 2016-01-01 03:24:00 | NYPD | New York City Police Department | Illegal Parking | Blocked Sidewalk | Street/Sidewalk | 11373.0 | 87-14 57 ROAD | ... | NaN | NaN | NaN | 40.733060 | -73.874170 | (40.733059618956815, -73.87416975810375) | 03:27:02 | 3.45 | Acceptable | Dec |

5 rows × 57 columns

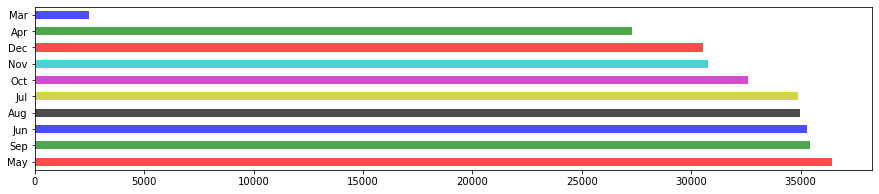
In [26]:

df\_nyc['Created\_Month'].value\_counts()

*# Create Bar plot for Complain Created Month to check frequency and it prove Most count is in May month and least is in March and in January there is no any complain*

df\_nyc['Created\_Month'].value\_counts().plot(kind="barh", color=list('rgbkymc'), alpha=0.7, figsize=(15,3))

plt.show()



In [27]:

*# To confirm doubt of January doesn't have any value, we used original dataframe and check if any entry for Jan month*

df\_orig[df\_orig['Created Date'].str.startswith('01/')]

Out[27]:

|  | **Unique Key** | **Created Date** | **Closed Date** | **Agency** | **Agency Name** | **Complaint Type** | **Descriptor** | **Location Type** | **Incident Zip** | **Incident Address** | **...** | **Bridge Highway Name** | **Bridge Highway Direction** | **Road Ramp** | **Bridge Highway Segment** | **Garage Lot Name** | **Ferry Direction** | **Ferry Terminal Name** | **Latitude** | **Longitude** | **Location** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

0 rows × 53 columns

In [28]:

*# Insight - 3*

*# Check count in each complain type - sorted decreasing order*

df\_nyc['Complaint Type'].value\_counts()

Out[28]:

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

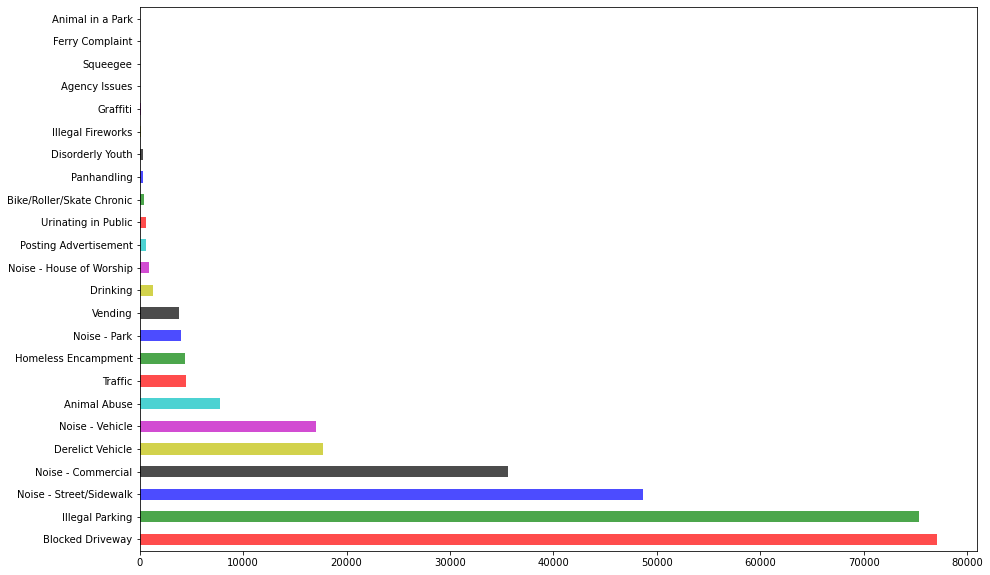
Name: Complaint Type, dtype: int64

In [29]:

*# Create Bar plot for complain type to check frequency in Complain Type*

df\_nyc['Complaint Type'].value\_counts().plot(kind="barh", color=list('rgbkymc'), alpha=0.7, figsize=(15,10))

plt.show()



In [30]:

*# Insight 4*

*# Let's check count for status type*

df\_nyc['Status'].value\_counts()

Out[30]:

Closed 298471

Open 1439

Assigned 786

Draft 2

Name: Status, dtype: int64

In [31]:

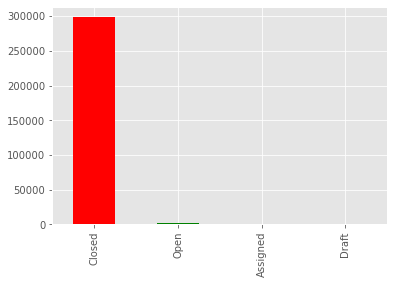
*# Draw Bar lot for Status*

**from** **matplotlib** **import** style

style.use('ggplot')

df\_nyc['Status'].value\_counts().plot(kind='bar', color=list('rgbkymc'))

plt.show()



Question 4.: Order the complaint types based on the average ‘Request\_Closing\_Time’, grouping them for different locations.

Solution 4:

In [32]:

*# For location we can choose here City, so first check if there is missing values there*

df\_nyc['City'].isnull().sum()

Out[32]:

2614

In [33]:

*# Fill all missing value with some default value here i used - Not Available*

df\_nyc['City'].fillna('Not Available', inplace=**True**)

In [34]:

df\_nyc['City'].head()

Out[34]:

0 NEW YORK

1 ASTORIA

2 BRONX

3 BRONX

4 ELMHURST

Name: City, dtype: object

In [35]:

df\_nyc['City']

Out[35]:

0 NEW YORK

1 ASTORIA

2 BRONX

3 BRONX

4 ELMHURST

...

300693 Not Available

300694 RICHMOND HILL

300695 BROOKLYN

300696 BRONX

300697 NEW YORK

Name: City, Length: 300698, dtype: object

In [36]:

*# Group them for City (location) first and Complain Type in that*

df\_nyc\_grouped = df\_nyc.groupby(['City', 'Complaint Type'])

In [37]:

*# get average of this grouped dataframe, and get Request\_Closing\_Time column from there*

df\_nyc\_mean = df\_nyc\_grouped.mean()['Request\_Closing\_In\_Hr']

df\_nyc\_mean.isnull().sum()

Out[37]:

4

In [38]:

*# Group by City(location) first and then Complain Type and showing average of Request Closing in Hour*

df\_nyc\_grouped = df\_nyc.groupby(['City','Complaint Type']).agg({'Request\_Closing\_In\_Hr': 'mean'})

df\_nyc\_grouped

Out[38]:

|  |  | **Request\_Closing\_In\_Hr** |
| --- | --- | --- |
| **City** | **Complaint Type** |  |
| **ARVERNE** | **Animal Abuse** | 2.153158 |
| **Blocked Driveway** | 2.526000 |
| **Derelict Vehicle** | 2.968889 |
| **Disorderly Youth** | 3.595000 |
| **Drinking** | 0.240000 |
| **...** | **...** | ... |
| **Woodside** | **Blocked Driveway** | 6.405455 |
| **Derelict Vehicle** | 4.965000 |
| **Illegal Parking** | 5.219500 |
| **Noise - Commercial** | 2.390000 |
| **Noise - Street/Sidewalk** | 3.410000 |

782 rows × 1 columns

In [39]:

*# Check if any value is NaN*

df\_nyc\_grouped[df\_nyc\_grouped['Request\_Closing\_In\_Hr'].isnull()]

Out[39]:

|  |  | **Request\_Closing\_In\_Hr** |
| --- | --- | --- |
| **City** | **Complaint Type** |  |
| **Not Available** | **Ferry Complaint** | NaN |
| **Noise - House of Worship** | NaN |
| **Panhandling** | NaN |
| **Posting Advertisement** | NaN |

In [40]:

*# Check total rows*

print(df\_nyc\_grouped)

Request\_Closing\_In\_Hr

City Complaint Type

ARVERNE Animal Abuse 2.153158

Blocked Driveway 2.526000

Derelict Vehicle 2.968889

Disorderly Youth 3.595000

Drinking 0.240000

... ...

Woodside Blocked Driveway 6.405455

Derelict Vehicle 4.965000

Illegal Parking 5.219500

Noise - Commercial 2.390000

Noise - Street/Sidewalk 3.410000

[782 rows x 1 columns]

In [41]:

*# drop null values from this group*

df\_nyc\_grouped\_withoutna = df\_nyc\_grouped.dropna()

In [42]:

*# verify if new group has null values*

df\_nyc\_grouped\_withoutna.isnull().sum()

Out[42]:

Request\_Closing\_In\_Hr 0

dtype: int64

In [43]:

*# verify number of rows after dropping null values*

print(df\_nyc\_grouped\_withoutna)

Request\_Closing\_In\_Hr

City Complaint Type

ARVERNE Animal Abuse 2.153158

Blocked Driveway 2.526000

Derelict Vehicle 2.968889

Disorderly Youth 3.595000

Drinking 0.240000

... ...

Woodside Blocked Driveway 6.405455

Derelict Vehicle 4.965000

Illegal Parking 5.219500

Noise - Commercial 2.390000

Noise - Street/Sidewalk 3.410000

[778 rows x 1 columns]

In [44]:

*# Sorting by column - Request\_Closing\_In\_Hr for City on grouped*

df\_nyc\_sorted = df\_nyc\_grouped\_withoutna.sort\_values(['City', 'Request\_Closing\_In\_Hr'])

df\_nyc\_sorted

Out[44]:

|  |  | **Request\_Closing\_In\_Hr** |
| --- | --- | --- |
| **City** | **Complaint Type** |  |
| **ARVERNE** | **Drinking** | 0.240000 |
| **Vending** | 0.480000 |
| **Urinating in Public** | 0.690000 |
| **Panhandling** | 1.030000 |
| **Noise - Park** | 1.285000 |
| **...** | **...** | ... |
| **Woodside** | **Noise - Commercial** | 2.390000 |
| **Noise - Street/Sidewalk** | 3.410000 |
| **Derelict Vehicle** | 4.965000 |
| **Illegal Parking** | 5.219500 |
| **Blocked Driveway** | 6.405455 |

778 rows × 1 columns

Question 5: Perform a statistical test for the following:

Please note: 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 [45]:

**import** **scipy.stats** **as** **stats**

**from** **math** **import** sqrt

In [46]:

*##### Try ANOVA for first one*

*# H0 : All Complain Types average response time mean is similar*

*# H1 : Not similar*

df\_nyc['Complaint Type'].value\_counts()

Out[46]:

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

Name: Complaint Type, dtype: int64

In [47]:

top5\_complaints\_type = df\_nyc['Complaint Type'].value\_counts()[:5]

top5\_complaints\_type

Out[47]:

Blocked Driveway 77044

Illegal Parking 75361

Noise - Street/Sidewalk 48612

Noise - Commercial 35577

Derelict Vehicle 17718

Name: Complaint Type, dtype: int64

In [48]:

top5\_complaints\_type\_names = top5\_complaints\_type.index

top5\_complaints\_type\_names

Out[48]:

Index(['Blocked Driveway', 'Illegal Parking', 'Noise - Street/Sidewalk',

'Noise - Commercial', 'Derelict Vehicle'],

dtype='object')

In [49]:

sample\_data = df\_nyc.loc[df\_nyc['Complaint Type'].isin(top5\_complaints\_type\_names), ['Complaint Type', 'Request\_Closing\_In\_Hr']]

sample\_data.head()

Out[49]:

|  | **Complaint Type** | **Request\_Closing\_In\_Hr** |
| --- | --- | --- |
| **0** | Noise - Street/Sidewalk | 0.92 |
| **1** | Blocked Driveway | 1.44 |
| **2** | Blocked Driveway | 4.86 |
| **3** | Illegal Parking | 7.75 |
| **4** | Illegal Parking | 3.45 |

In [50]:

sample\_data.shape

Out[50]:

(254312, 2)

In [51]:

sample\_data.isnull().sum()

Out[51]:

Complaint Type 0

Request\_Closing\_In\_Hr 2059

dtype: int64

In [52]:

*#sample\_data[~sample\_data.isin(['NaN', 'NaT']).any(axis=1)]*

*#sample\_data[sample\_data.isnull()]*

sample\_data.dropna(how='any', inplace=**True**)

sample\_data.isnull().sum()

*# sample\_data\_without\_null[sample\_data\_without\_null.isnull()]*

Out[52]:

Complaint Type 0

Request\_Closing\_In\_Hr 0

dtype: int64

In [53]:

sample\_data.shape

Out[53]:

(252253, 2)

In [54]:

s1 = sample\_data[sample\_data['Complaint Type'] == top5\_complaints\_type\_names[0]].Request\_Closing\_In\_Hr

s1.head()

Out[54]:

1 1.44

2 4.86

7 1.80

9 1.38

10 7.80

Name: Request\_Closing\_In\_Hr, dtype: float64

In [55]:

s2 = sample\_data[sample\_data['Complaint Type'] == top5\_complaints\_type\_names[1]].Request\_Closing\_In\_Hr

s2.head()

Out[55]:

3 7.75

4 3.45

5 1.89

6 1.96

8 8.55

Name: Request\_Closing\_In\_Hr, dtype: float64

In [56]:

s3 = sample\_data[sample\_data['Complaint Type'] == top5\_complaints\_type\_names[2]].Request\_Closing\_In\_Hr

s3.head()

Out[56]:

0 0.92

12 2.48

19 0.78

38 0.49

54 1.50

Name: Request\_Closing\_In\_Hr, dtype: float64

In [57]:

s4 = sample\_data[sample\_data['Complaint Type'] == top5\_complaints\_type\_names[3]].Request\_Closing\_In\_Hr

s4.head()

Out[57]:

17 0.85

18 2.93

22 1.26

29 2.50

30 1.99

Name: Request\_Closing\_In\_Hr, dtype: float64

In [58]:

s5 = sample\_data[sample\_data['Complaint Type'] == top5\_complaints\_type\_names[4]].Request\_Closing\_In\_Hr

s5.head()

Out[58]:

14 10.49

151 3.95

255 1.36

256 4.13

295 0.75

Name: Request\_Closing\_In\_Hr, dtype: float64

In [59]:

print(s1.isnull().sum())

print(s2.isnull().sum())

print(s3.isnull().sum())

print(s4.isnull().sum())

print(s5.isnull().sum())

0

0

0

0

0

In [60]:

stats.f\_oneway(s1, s2, s3, s4, s5)

Out[60]:

F\_onewayResult(statistic=1799.598683238952, pvalue=0.0)

**We can see pvalue is less than 0.05 so we reject null hypothesis and average response time is not same.**

#### Try ChiSquare Test for second one - # Are the type of complaint or service requested and location related?

* H0 : 2 categories - Complain Type and Location is independent means not related
* Ha : 2 categories - Complain Type and Location is dependent means related

In [61]:

top5\_location = df\_nyc['City'].value\_counts()[:5]

top5\_location

Out[61]:

BROOKLYN 98307

NEW YORK 65994

BRONX 40702

STATEN ISLAND 12343

JAMAICA 7296

Name: City, dtype: int64

In [62]:

top5\_location\_names = top5\_location.index

top5\_location\_names

Out[62]:

Index(['BROOKLYN', 'NEW YORK', 'BRONX', 'STATEN ISLAND', 'JAMAICA'], dtype='object')

In [63]:

sample\_data\_location\_c\_type = df\_nyc.loc[(df\_nyc['Complaint Type'].isin(top5\_complaints\_type\_names)) & (df\_nyc['City'].isin(top5\_location\_names)), ['Complaint Type', 'City']]

sample\_data\_location\_c\_type.head()

Out[63]:

|  | **Complaint Type** | **City** |
| --- | --- | --- |
| **0** | Noise - Street/Sidewalk | NEW YORK |
| **2** | Blocked Driveway | BRONX |
| **3** | Illegal Parking | BRONX |
| **5** | Illegal Parking | BROOKLYN |
| **6** | Illegal Parking | NEW YORK |

In [64]:

pd.crosstab(sample\_data\_location\_c\_type['Complaint Type'], sample\_data\_location\_c\_type['City'], margins=**True**)

Out[64]:

| **City** | **BRONX** | **BROOKLYN** | **JAMAICA** | **NEW YORK** | **STATEN ISLAND** | **All** |
| --- | --- | --- | --- | --- | --- | --- |
| **Complaint Type** |  |  |  |  |  |  |
| **Blocked Driveway** | 12755 | 28148 | 2818 | 2072 | 2142 | 47935 |
| **Derelict Vehicle** | 1953 | 5181 | 954 | 537 | 1766 | 10391 |
| **Illegal Parking** | 7859 | 27462 | 1421 | 12128 | 4886 | 53756 |
| **Noise - Commercial** | 2434 | 11463 | 429 | 14550 | 678 | 29554 |
| **Noise - Street/Sidewalk** | 8892 | 13356 | 339 | 20433 | 819 | 43839 |
| **All** | 33893 | 85610 | 5961 | 49720 | 10291 | 185475 |

In [65]:

ch2, p\_value, df, exp\_frq = stats.chi2\_contingency(pd.crosstab(sample\_data\_location\_c\_type['Complaint Type'], sample\_data\_location\_c\_type['City']))

In [66]:

print(ch2)

print(p\_value)

40522.79928349593

0.0

**We can see pvalue is less than 0.05 so we reject null hypothesis means complain type and location is not independent.**