In [1]:

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
nypd_orig = pd.read_csv('./Dataset/311_Service_Requests_from_2010_to_Present.csv')
nypd_copy = pd.read_csv('./Dataset/311_Service_Requests_from_2010_to_Present.csv')
```

```
c:\users\aditya\appdata\local\programs\python\python37\lib\site-packages\IPy
thon\core\interactiveshell.py:3063: DtypeWarning: Columns (48,49) have mixed
types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:

```
nypd_copy.shape
```

Out[3]:

```
(300698, 53)
```

In [5]:

```
nypd_copy.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
 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 [10]:

```
nypd_copy.drop(nypd_copy.columns[10:-3],axis=1,inplace=True)
```

In [11]:

```
nypd_copy.shape
```

Out[11]:

```
(300698, 13)
```

In [13]:

```
nypd_copy.dropna(inplace=True)
```

In [14]:

```
nypd_copy.shape
```

Out[14]:

```
(248848, 13)
```

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 [47]:

```
nypd_copy['Created Date']=pd.to_datetime(nypd_copy['Created Date'])
nypd_copy['Closed Date']=pd.to_datetime(nypd_copy['Closed Date'])
nypd_copy['Request_Closing_Time']=(nypd_copy['Closed Date']-nypd_copy['Created Date']).asty
nypd_copy['Month'] = nypd_copy['Created Date'].dt.month
```

In [33]:

```
nypd_copy
```

Out[33]:

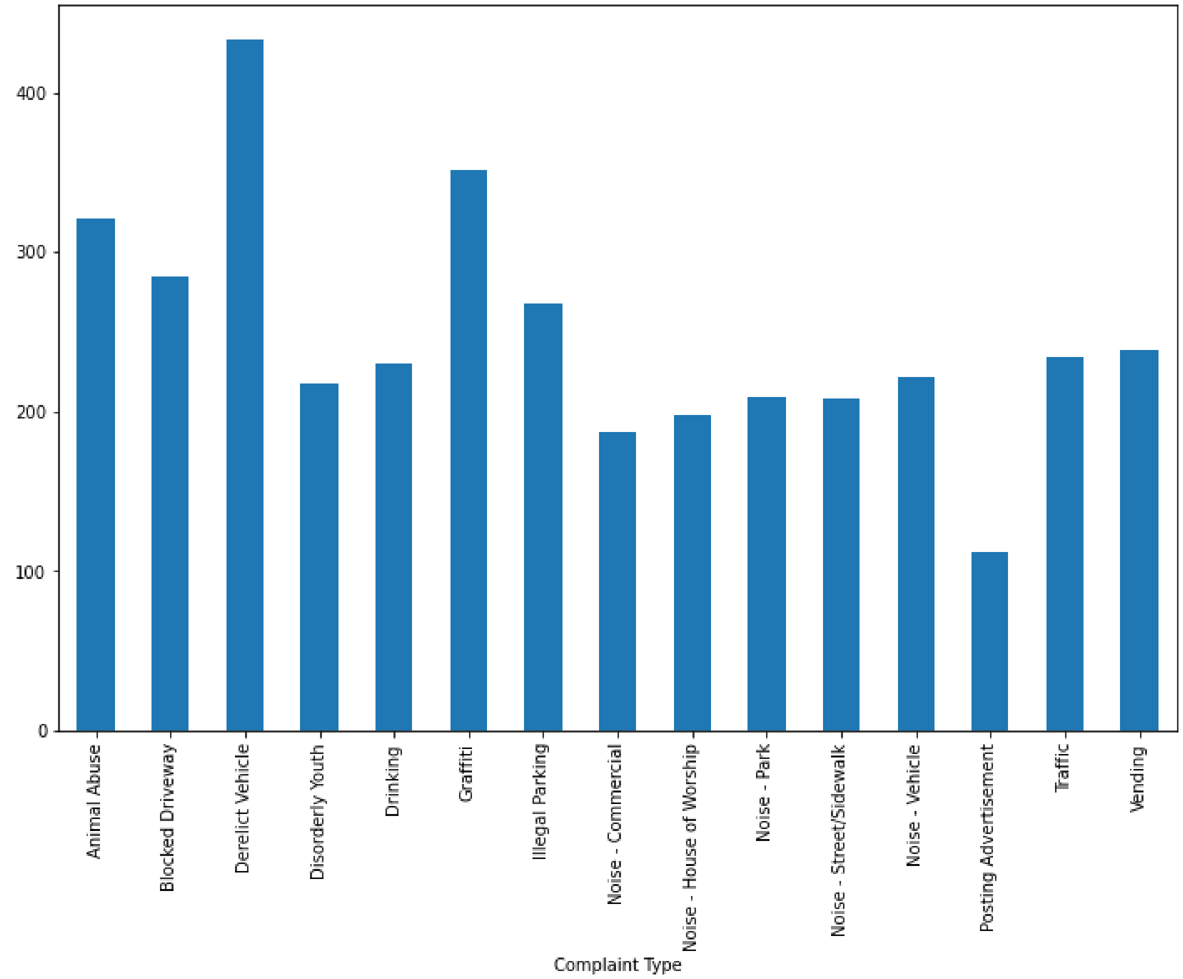
	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Loc
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:00	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Stree
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:00	NYPD	New York City Police Department	Blocked Driveway	No Access	Stree
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:00	NYPD	New York City Police Department	Blocked Driveway	No Access	Stree
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	Stree
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:00	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Stree
...	...	...	...	...	...	...	...	
300692	30281370	2015-03-29 00:34:32	2015-03-29 01:13:01	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Store/C
300694	30281230	2015-03-29 00:33:28	2015-03-29 02:33:59	NYPD	New York City Police Department	Blocked Driveway	Partial Access	Stree
300695	30283424	2015-03-29 00:33:03	2015-03-29 03:40:20	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/
300696	30280004	2015-03-29 00:33:02	2015-03-29 04:38:35	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/
300697	30281825	2015-03-29 00:33:01	2015-03-29 04:41:50	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Store/C

248848 rows × 15 columns

Solution2 Observation1

In [21]:

```
ax = plt.subplot()
nypd_copy.groupby('Complaint Type')['Request_Closing_Time'].mean().plot(kind='bar',
                                                        figsize=(12,8),
                                                        ax=ax)
```



In [49]:

```
nypd_copy.groupby('Complaint Type')['Request_Closing_Time'].mean()
```

Out[49]:

```
Complaint Type
Animal Abuse                 321.657499
Blocked Driveway             284.096237
Derelict Vehicle             433.390360
Disorderly Youth             217.075269
Drinking                     230.163690
Graffiti                     351.810526
Illegal Parking              268.250286
Noise - Commercial           187.472970
Noise - House of Worship     197.902299
Noise - Park                 209.650394
Noise - Street/Sidewalk      208.555170
Noise - Vehicle              221.685964
Posting Advertisement        111.748792
Traffic                      234.766050
Vending                      238.206600
Name: Request_Closing_Time, dtype: float64
```

Highest average response time is for Complaint type Derelict vehicles: 433

Solution2 Observation2

In [42]:

```
ax = plt.subplot()
nypd_copy.groupby('Location Type')['Request_Closing_Time'].mean().plot(kind='bar',
                                                        figsize=(12,8),
                                                        ax=ax)
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e803feb248>
```

In [29]:

```
nypd_copy.groupby('Location Type')['Request_Closing_Time'].mean().max()
```

Out[29]:

```
984.75
```

Highest average response time is for location type Roadway Tunnel: 984

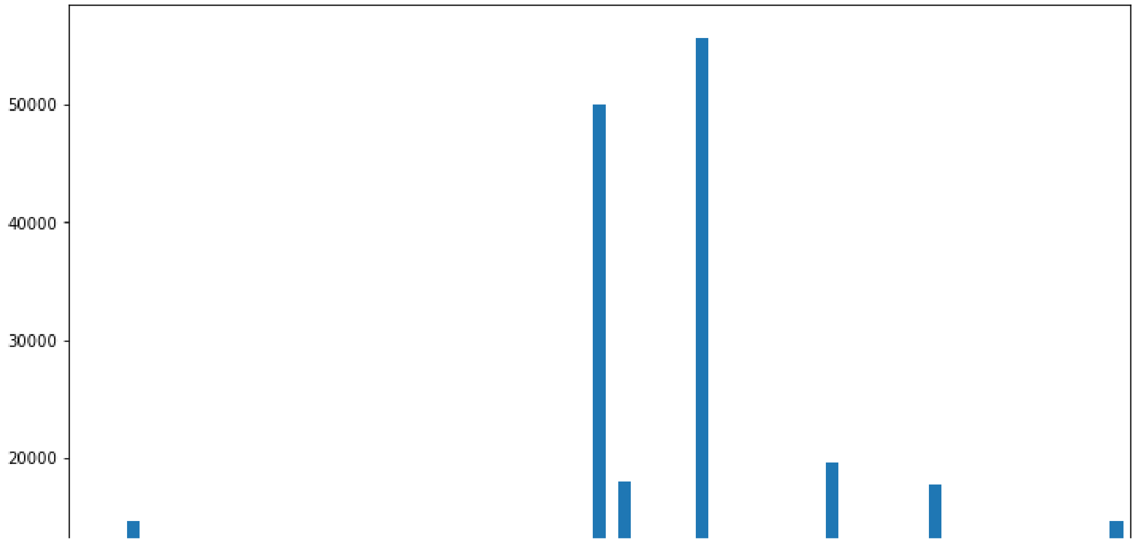
Solution2 Observation3

In [40]:

```
ax = plt.subplot()
nypd_copy.groupby('Descriptor')['Request_Closing_Time'].count().plot(kind='bar',
                                                        figsize=(12,8),
                                                        ax=ax)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e803faa348>
```



In [39]:

```
nypd_copy.groupby('Descriptor')['Request_Closing_Time'].count().max()
```

Out[39]:

```
55708
```

Highest count of complaints is for 'No Access': 55708

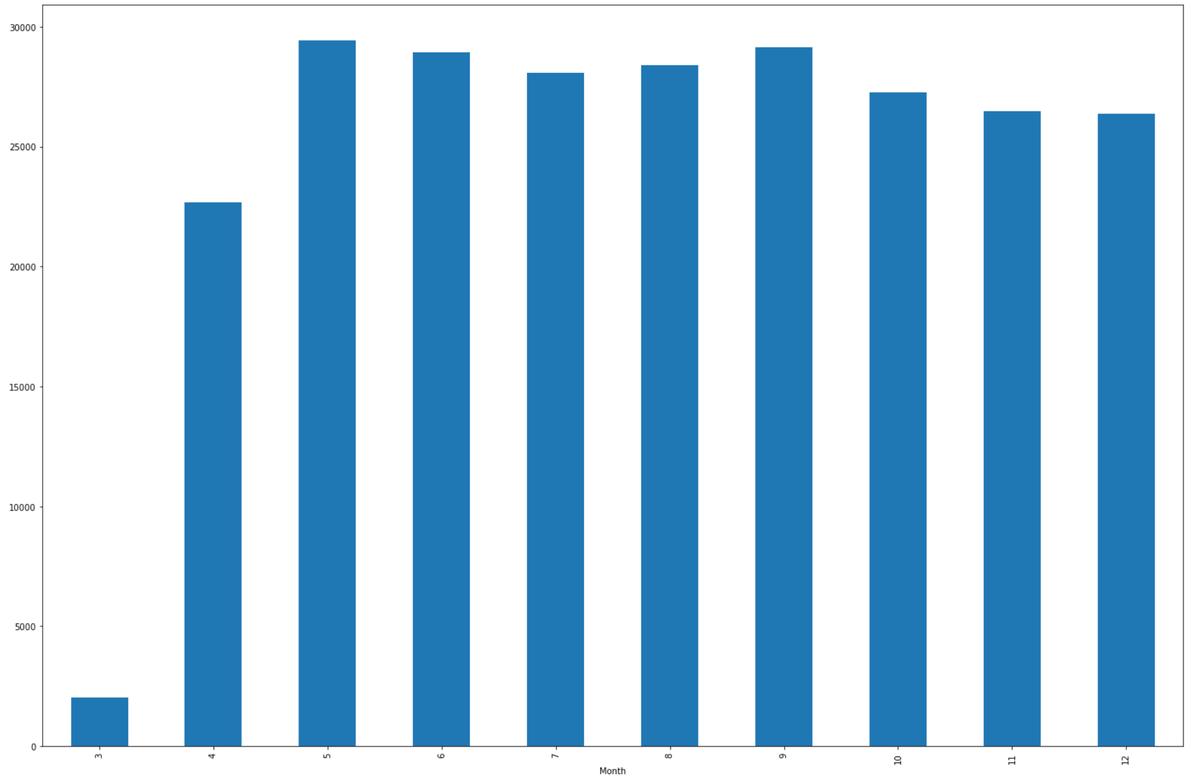
Solution2 Observation4

In [36]:

```
ax = plt.subplot()
nypd_copy.groupby('Month')['Request_Closing_Time'].count().plot(kind='bar',
                                                        figsize=(24,16),
                                                        ax=ax)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e803c97cc8>
```



In [38]:

```
nypd_copy.groupby('Month')['Request_Closing_Time'].count().max()
```

Out[38]:

```
29437
```

Highest count of complaints are in the month of May: 29437

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

In [52]:

```
nypd_loacation_complaintType= nypd_copy.groupby(['Location','Complaint Type']).agg({'Reques
```

In [56]:

```
nypd_loacation_complaintType.sort_values(['Request_Closing_Time','Location'])
```

Out[56]:

		Request_Closing_Time
Location	**Complaint Type**	
(40.678429539269835, -73.98361397723242)	**Noise - Commercial**	2.000000
(40.69371028050496, -73.95499211670034)	**Illegal Parking**	2.000000
(40.71598512070559, -73.9509008064274)	**Illegal Parking**	2.000000
(40.76848580086362, -73.91235250532725)	**Noise - Vehicle**	2.000000
(40.510211690922475, -74.24398548733994)	**Illegal Parking**	3.000000
...	**...**	...
(40.68349308751147, -73.73091308242111)	**Derelict Vehicle**	10146.000000
(40.6449662497121, -73.99499837340035)	**Animal Abuse**	10485.333333
(40.65367609466097, -73.95792751148433)	**Derelict Vehicle**	11556.000000
(40.64466438582295, -73.95635848114169)	**Derelict Vehicle**	13401.000000
(40.59814521498835, -73.98935198928409)	**Illegal Parking**	34641.000000

124103 rows × 1 columns

In [57]:

```
import scipy.stats as stats
from math import sqrt
```

The hypothesis being tested

- **Null hypothesis (H0): the average response time across complaint types is similar**
- **Alternative hypothesis (H1): the average response time across complaint types is NOT similar**

In [59]:

```
nypd_copy['Complaint Type'].value_counts()
```

Out[59]:

```
Blocked Driveway             75366
Illegal Parking              61294
Noise - Street/Sidewalk      39451
Noise - Commercial           32334
Derelict Vehicle             14689
Noise - Vehicle              12126
Animal Abuse                  6581
Vending                       2212
Noise - Park                  1270
Drinking                      1008
Traffic                        919
Noise - House of Worship       696
Posting Advertisement          621
Disorderly Youth               186
Graffiti                        95
Name: Complaint Type, dtype: int64
```

In [61]:

```
sampledata = nypd_copy[['Complaint Type','Request_Closing_Time']]
sampledata.head()
```

Out[61]:

	Complaint Type	Request_Closing_Time
0	Noise - Street/Sidewalk	55.0
1	Blocked Driveway	86.0
2	Blocked Driveway	291.0
3	Illegal Parking	465.0
4	Illegal Parking	207.0

In [64]:

```
sampledata.isna().sum()
```

Out[64]:

```
Complaint Type          0
Request_Closing_Time    0
dtype: int64
```

In [67]:

```
sampledata.groupby('Complaint Type')['Request_Closing_Time'].describe().T
```

Out[67]:

Complaint Type	Animal Abuse	Blocked Driveway	Derelict Vehicle	Disorderly Youth	Drinking	Graffiti	
count	6581.000000	75366.000000	14689.000000	186.000000	1008.000000	95.000000	6
mean	321.657499	284.096237	433.390360	217.075269	230.163690	351.810526	
std	542.620315	334.974111	639.775973	224.736301	317.060966	330.061061	
min	3.000000	2.000000	3.000000	6.000000	4.000000	9.000000	
25%	101.000000	96.000000	101.000000	86.250000	70.750000	129.500000	
50%	203.000000	190.000000	241.000000	157.000000	152.500000	260.000000	
75%	395.000000	357.000000	507.000000	281.750000	294.250000	436.500000	
max	31156.000000	8897.000000	13401.000000	1683.000000	5686.000000	1594.000000	3

In [68]:

```
sampledata['Complaint Type'] = pd.Categorical(sampledata['Complaint Type'])
```

```
c:\users\aditya\appdata\local\programs\python\python37\lib\site-packages\ipy
kernel_launcher.py:1: SettingWithCopyWarning:
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_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  """Entry point for launching an IPython kernel.
```

In [69]:

```
sampledata['code'] = sampledata['Complaint Type'].cat.codes
```

```
c:\users\aditya\appdata\local\programs\python\python37\lib\site-packages\ipy
kernel_launcher.py:1: SettingWithCopyWarning:
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_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  """Entry point for launching an IPython kernel.
```

In [73]:

```
sampledata.code.unique()
```

Out[73]:

```
array([10,  1,  6,  2,  7,  8, 12,  0, 14, 11,  4, 13,  9,  5,  3],
      dtype=int8)
```

In [74]:

```
sampledata.groupby('code')['Request_Closing_Time'].describe().T
```

Out[74]:

code	0	1	2	3	4	5	
count	6581.000000	75366.000000	14689.000000	186.000000	1008.000000	95.000000	61294
mean	321.657499	284.096237	433.390360	217.075269	230.163690	351.810526	268
std	542.620315	334.974111	639.775973	224.736301	317.060966	330.061061	357
min	3.000000	2.000000	3.000000	6.000000	4.000000	9.000000	2
25%	101.000000	96.000000	101.000000	86.250000	70.750000	129.500000	84
50%	203.000000	190.000000	241.000000	157.000000	152.500000	260.000000	176
75%	395.000000	357.000000	507.000000	281.750000	294.250000	436.500000	342
max	31156.000000	8897.000000	13401.000000	1683.000000	5686.000000	1594.000000	34641

In [76]:

```
from scipy import stats
stats.f_oneway(sampledata['Request_Closing_Time'][sampledata['code'] == 0],
               sampledata['Request_Closing_Time'][sampledata['code'] == 1],
               sampledata['Request_Closing_Time'][sampledata['code'] == 2],
              sampledata['Request_Closing_Time'][sampledata['code'] == 3],
               sampledata['Request_Closing_Time'][sampledata['code'] == 4],
               sampledata['Request_Closing_Time'][sampledata['code'] == 5],
              sampledata['Request_Closing_Time'][sampledata['code'] == 6],
               sampledata['Request_Closing_Time'][sampledata['code'] == 7],
               sampledata['Request_Closing_Time'][sampledata['code'] == 8],
              sampledata['Request_Closing_Time'][sampledata['code'] == 9],
               sampledata['Request_Closing_Time'][sampledata['code'] == 10],
               sampledata['Request_Closing_Time'][sampledata['code'] == 11],
              sampledata['Request_Closing_Time'][sampledata['code'] == 12],
               sampledata['Request_Closing_Time'][sampledata['code'] == 13],
               sampledata['Request_Closing_Time'][sampledata['code'] == 14],)
```

Out[76]:

```
F_onewayResult(statistic=477.2510130458414, pvalue=0.0)
```

In [77]:

```
from statsmodels.formula.api import ols
result = ols('Request_Closing_Time ~ C(code)', data = sampledata).fit()
```

In [78]:

```
print(result.summary())
```

```
                            OLS Regression Results                            
============================================================================
====
Dep. Variable:     Request_Closing_Time   R-squared:                        
0.026
Model:                              OLS   Adj. R-squared:                   
0.026
Method:                   Least Squares   F-statistic:                     4
77.3
Date:                  Fri, 12 Jun 2020   Prob (F-statistic):               
0.00
Time:                          01:06:35   Log-Likelihood:            -1.8126
e+06
No. Observations:                248848   AIC:                         3.625
e+06
Df Residuals:                    248833   BIC:                         3.625
e+06
Df Model:                            14                                         
Covariance Type:              nonrobust                                         
============================================================================
=====
                    coef    std err          t      P>|t|      [0.025      
0.975]
----------------------------------------------------------------------------
-----
Intercept       321.6575      4.345     74.035      0.000     313.142     33
0.173
C(code)[T.1]    -37.5613      4.530     -8.291      0.000     -46.441     -2
8.682
C(code)[T.2]    111.7329      5.228     21.371      0.000     101.486     12
1.980
C(code)[T.3]   -104.5822     26.206     -3.991      0.000    -155.945     -5
3.219
C(code)[T.4]    -91.4938     11.921     -7.675      0.000    -114.859     -6
8.129
C(code)[T.5]     30.1530     36.421      0.828      0.408     -41.232     10
1.538
C(code)[T.6]    -53.4072      4.572    -11.681      0.000     -62.368     -4
4.446
C(code)[T.7]   -134.1845      4.766    -28.152      0.000    -143.526    -12
4.843
C(code)[T.8]   -123.7552     14.049     -8.809      0.000    -151.290     -9
6.220
C(code)[T.9]   -112.0071     10.802    -10.369      0.000    -133.179     -9
0.835
C(code)[T.10]  -113.1023      4.693    -24.100      0.000    -122.301    -10
3.904
C(code)[T.11]   -99.9715      5.396    -18.526      0.000    -110.548     -8
9.395
C(code)[T.12]  -209.9087     14.796    -14.187      0.000    -238.908    -18
0.909
C(code)[T.13]   -86.8914     12.412     -7.001      0.000    -111.218     -6
2.565
C(code)[T.14]   -83.4509      8.662     -9.634      0.000    -100.429     -6
6.473
============================================================================
==
Omnibus:                   449443.107   Durbin-Watson:                   1.9
34
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       4835122344.6
56
Skew:                          12.396   Prob(JB):                         0.
00
Kurtosis:                     685.427   Cond. No.                          5
6.8
============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

As the pvalue is less than 0.05 so we reject null hypothesis and can conclude that average response time is not same.

In [ ]:

```

```

The hypothesis being tested

- **Null hypothesis (H0): the type of complaint or service requested and location are independent**
- **Alternative hypothesis (H1):the type of complaint or service requested and location are not independent**

In [79]:

```python
top5_location = nypd_copy['Location'].value_counts()[:5]
top5_location
```

Out[79]:

```
(40.83036235589997, -73.86602154214397)     901
(40.72195913199264, -73.80969682426189)     505
(40.703818970933284, -73.94207345177706)    476
(40.549093797686275, -74.17363282481907)    311
(40.79770758865914, -73.9401822682408)      295
Name: Location, dtype: int64
```

In [80]:

```
top5_location_names = top5_location.index
top5_location_names
```

Out[80]:

```
Index(['(40.83036235589997, -73.86602154214397)',
       '(40.72195913199264, -73.80969682426189)',
       '(40.703818970933284, -73.94207345177706)',
       '(40.549093797686275, -74.17363282481907)',
       '(40.79770758865914, -73.9401822682408)'],
      dtype='object')
```

In [82]:

```
top5_complaints_type = nypd_copy['Complaint Type'].value_counts()[:5]
top5_complaints_type
```

Out[82]:

```
Blocked Driveway           75366
Illegal Parking            61294
Noise - Street/Sidewalk    39451
Noise - Commercial         32334
Derelict Vehicle           14689
Name: Complaint Type, dtype: int64
```

In [83]:

```
top5_complaints_type_names = top5_complaints_type.index
top5_complaints_type_names
```

Out[83]:

```
Index(['Blocked Driveway', 'Illegal Parking', 'Noise - Street/Sidewalk',
       'Noise - Commercial', 'Derelict Vehicle'],
      dtype='object')
```

In [85]:

```
sample_data = nypd_copy.loc[(nypd_copy['Complaint Type'].isin(top5_complaints_type_names))
sample_data.head()
```

Out[85]:

	Complaint Type	Location
385	Illegal Parking	(40.72195913199264, -73.80969682426189)
441	Noise - Street/Sidewalk	(40.83036235589997, -73.86602154214397)
478	Noise - Commercial	(40.83036235589997, -73.86602154214397)
862	Noise - Commercial	(40.83036235589997, -73.86602154214397)
1010	Noise - Commercial	(40.83036235589997, -73.86602154214397)

In [87]:

```
pd.crosstab(sample_data['Complaint Type'], sample_data['Location'], margins=True)
```

Out[87]:

Location	(40.549093797686275, -74.17363282481907)	(40.703818970933284, -73.94207345177706)	(40.72195913199264, -73.80969682426189)	(40.7977075886 -73.940182268
Complaint Type				
Blocked Driveway	1	0	2	
Illegal Parking	54	0	503	
Noise - Commercial	0	458	0	
Noise - Street/Sidewalk	0	18	0	
All	55	476	505	

In [90]:

```
ch2, p_value, df, exp_frq = stats.chi2_contingency(pd.crosstab(sample_data['Complaint Type'
```

In [95]:

```
print(f"ch^2:{ch2}")
print(f"p-value:{p_value}")
```

```
ch^2:2637.1351752488144
p-value:0.0
```

As the pvalue is pvalue is less than 0.05 so we reject null hypothesis and conclude that complain type and location are not independent.

In [ ]:

```
```