Customer_Service_Request-2

November 21, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# 1. Import a 311 NYC service request.
df = pd.read_csv('311_Service_Requests_2.csv', low_memory=False)
```

[3]:
```
df.head()
```

[3]:
```
   Unique Key            Created Date    Closed Date Agency  \
0    32310363  12/31/2015 11:59:45 PM  01-01-16 0:55   NYPD
1    32309934  12/31/2015 11:59:44 PM  01-01-16 1:26   NYPD
2    32309159  12/31/2015 11:59:29 PM  01-01-16 4:51   NYPD
3    32305098  12/31/2015 11:57:46 PM  01-01-16 7:43   NYPD
4    32306529  12/31/2015 11:56:58 PM  01-01-16 3:24   NYPD

                       Agency Name           Complaint Type  \
0  New York City Police Department  Noise - Street/Sidewalk
1  New York City Police Department         Blocked Driveway
2  New York City Police Department         Blocked Driveway
3  New York City Police Department           Illegal Parking
4  New York City Police Department           Illegal Parking

                    Descriptor    Location Type  Incident Zip  \
0            Loud Music/Party  Street/Sidewalk       10034.0
1                   No Access  Street/Sidewalk       11105.0
2                   No Access  Street/Sidewalk       10458.0
3  Commercial Overnight Parking  Street/Sidewalk       10461.0
4            Blocked Sidewalk  Street/Sidewalk       11373.0

       Incident Address  … Bridge Highway Name Bridge Highway Direction  \
0    71 VERMILYEA AVENUE  …                 NaN                      NaN
1        27-07 23 AVENUE  …                 NaN                      NaN
2  2897 VALENTINE AVENUE  …                 NaN                      NaN
3    2940 BAISLEY AVENUE  …                 NaN                      NaN
4          87-14 57 ROAD  …                 NaN                      NaN
```

```
  Road Ramp Bridge Highway Segment Garage Lot Name Ferry Direction  \
0       NaN                    NaN             NaN             NaN
1       NaN                    NaN             NaN             NaN
2       NaN                    NaN             NaN             NaN
3       NaN                    NaN             NaN             NaN
4       NaN                    NaN             NaN             NaN

  Ferry Terminal Name   Latitude  Longitude  \
0                 NaN  40.865682 -73.923501
1                 NaN  40.775945 -73.915094
2                 NaN  40.870325 -73.888525
3                 NaN  40.835994 -73.828379
4                 NaN  40.733060 -73.874170

                                   Location
0   (40.86568153633767, -73.92350095571744)
1  (40.775945312321085, -73.91509393898605)
2  (40.870324522111424, -73.88852464418646)
3   (40.83599404683083, -73.82837939584206)
4  (40.733059618956815, -73.87416975810375)

[5 rows x 53 columns]
```

[4]:
```
df.columns
```

[4]:
```
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')
```

[5]:
```
df.shape
```

[5]:
```
(255717, 53)
```

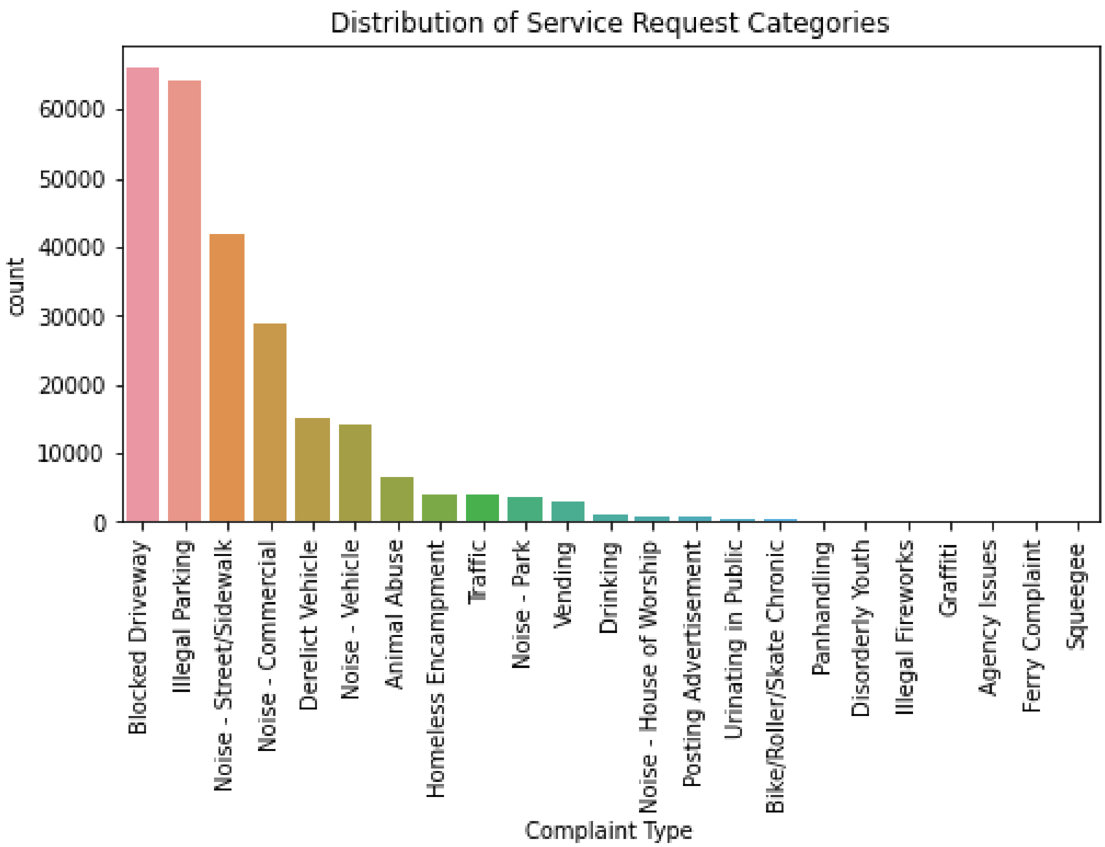
[6]:
```python
2.#Read or convert the columns ‘Created Date’ and Closed Date’ to datetime␣
↪datatype and
 #create a new column ‘Request_Closing_Time’ as the time elapsed between␣
↪request creation and request closing.
#(Hint: Explore the package/module datetime)
# Convert 'Created Date' and 'Closed Date' to datetime
df['Created Date'] = pd.to_datetime(df['Created Date'])
df['Closed Date'] = pd.to_datetime(df['Closed Date'])

# Calculate the time difference using the datetime module
df['Request_Closing_Time'] = df['Closed Date'] - df['Created Date']

# Display the updated DataFrame
print(df.head())
```

```
   Unique Key        Created Date         Closed Date Agency  \
0    32310363 2015-12-31 23:59:45 2016-01-01 00:55:00   NYPD
1    32309934 2015-12-31 23:59:44 2016-01-01 01:26:00   NYPD
2    32309159 2015-12-31 23:59:29 2016-01-01 04:51:00   NYPD
3    32305098 2015-12-31 23:57:46 2016-01-01 07:43:00   NYPD
4    32306529 2015-12-31 23:56:58 2016-01-01 03:24:00   NYPD

                       Agency Name           Complaint Type  \
0  New York City Police Department  Noise - Street/Sidewalk
1  New York City Police Department         Blocked Driveway
2  New York City Police Department         Blocked Driveway
3  New York City Police Department          Illegal Parking
4  New York City Police Department          Illegal Parking

                    Descriptor    Location Type  Incident Zip  \
0             Loud Music/Party  Street/Sidewalk       10034.0
1                    No Access  Street/Sidewalk       11105.0
2                    No Access  Street/Sidewalk       10458.0
3  Commercial Overnight Parking  Street/Sidewalk       10461.0
4             Blocked Sidewalk  Street/Sidewalk       11373.0

        Incident Address  … Bridge Highway Direction Road Ramp  \
0    71 VERMILYEA AVENUE  …                      NaN       NaN
1       27-07 23 AVENUE  …                      NaN       NaN
2  2897 VALENTINE AVENUE  …                      NaN       NaN
3    2940 BAISLEY AVENUE  …                      NaN       NaN
4          87-14 57 ROAD  …                      NaN       NaN

  Bridge Highway Segment Garage Lot Name Ferry Direction Ferry Terminal Name  \
0                    NaN             NaN             NaN                 NaN
1                    NaN             NaN             NaN                 NaN
2                    NaN             NaN             NaN                 NaN
```

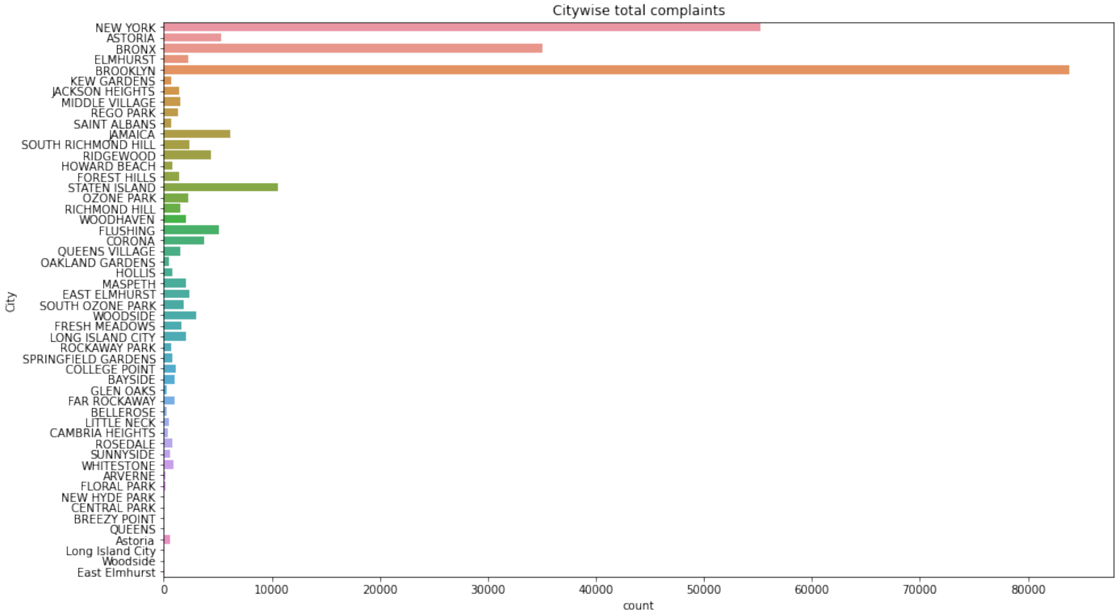
```
3                    NaN              NaN              NaN                   NaN
4                    NaN              NaN              NaN                   NaN

    Latitude  Longitude                                  Location  \
0  40.865682 -73.923501   (40.86568153633767, -73.92350095571744)
1  40.775945 -73.915094  (40.775945312321085, -73.91509393898605)
2  40.870325 -73.888525  (40.870324522111424, -73.88852464418646)
3  40.835994 -73.828379   (40.83599404683083, -73.82837939584206)
4  40.733060 -73.874170  (40.733059618956815, -73.87416975810375)

  Request_Closing_Time
0      0 days 00:55:15
1      0 days 01:26:16
2      0 days 04:51:31
3      0 days 07:45:14
4      0 days 03:27:02

[5 rows x 54 columns]
```

[7]:

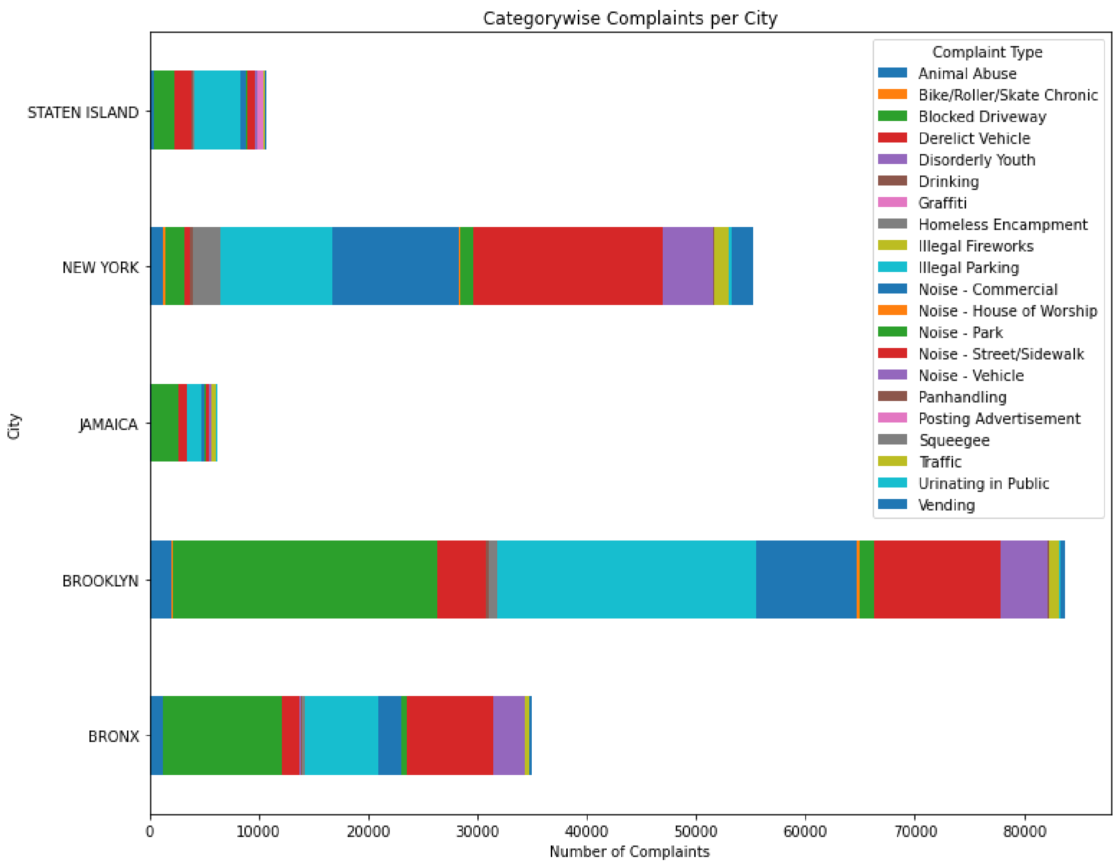
```python
# 3.Provide major insights/patterns that you can offer in a visual format␣
↪(graphs or tables);
# at least 4 major conclusions that you can come up with after generic data␣
↪mining.
# Plotting the distribution of service request categories
plt.figure(figsize=(8, 4))
sns.countplot(x='Complaint Type', data=df, order=df['Complaint Type'].
↪value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Service Request Categories')
plt.show()
```



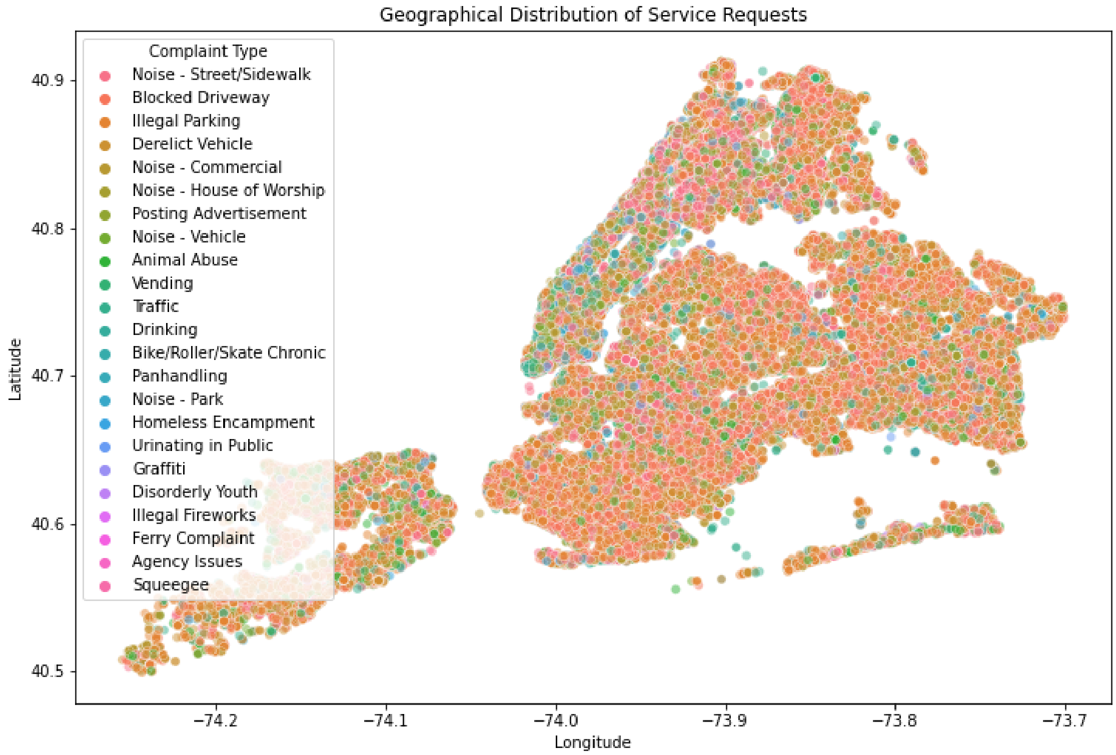
[8]:
```
#citywise complaint counts(total)
plt.figure(figsize=(15,9))
plt.title('Citywise total complaints')
sns.countplot(y='City',data=df)
plt.show()
```



[14]:
```
top5cities = df['City'].value_counts().head(5).index.to_list()
dstop5 = df[df['City'].isin(top5cities)]

# Citywise complaint counts (typewise)
df1 = pd.crosstab(dstop5['City'], dstop5['Complaint Type'])

# Plotting citywise complaint counts (typewise)
df1.plot(kind='barh', stacked=True, figsize=(12, 10))
plt.title('Categorywise Complaints per City')
plt.xlabel('Number of Complaints')
plt.ylabel('City')
plt.show()
```

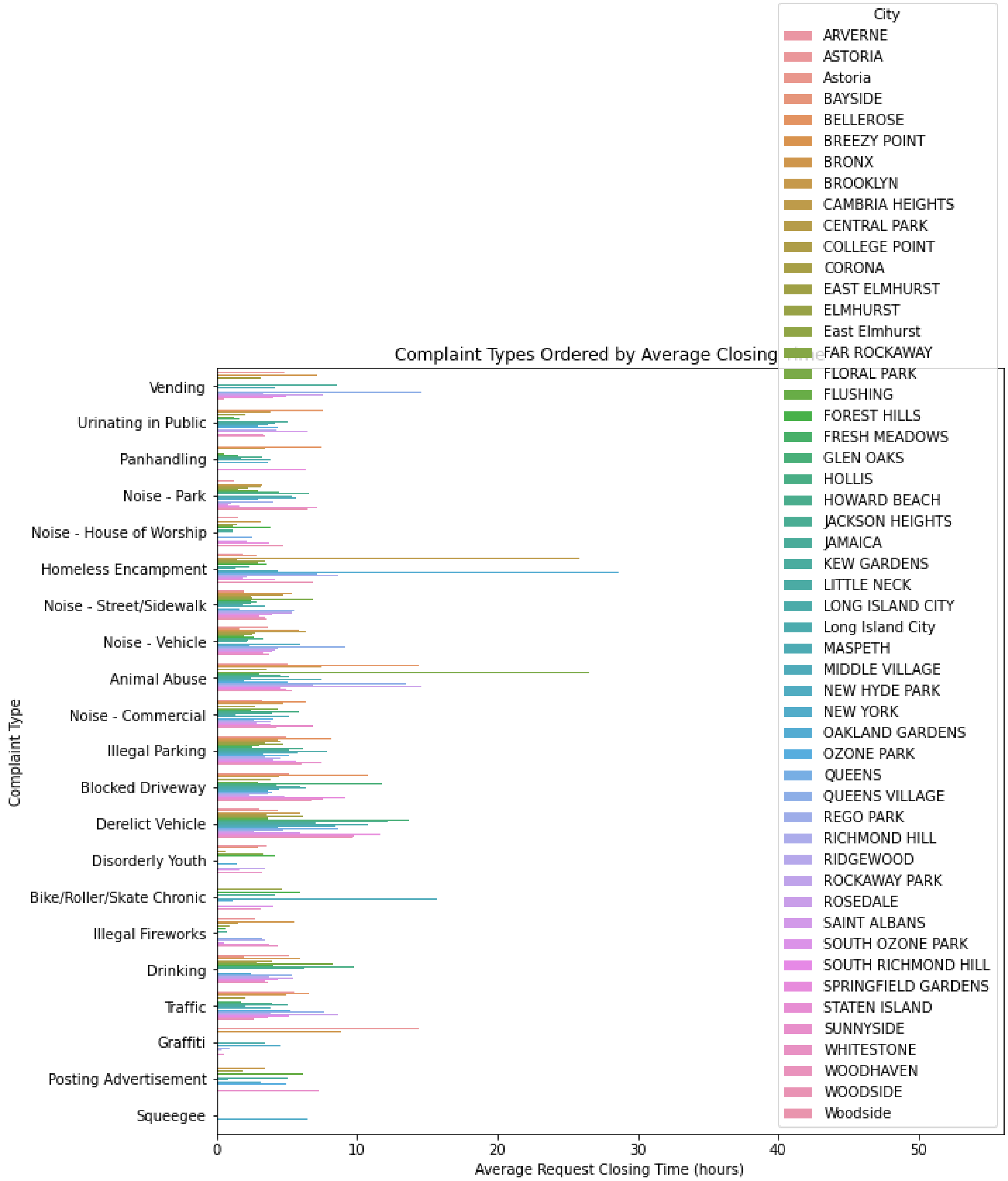


[15]:
```python
# Assuming 'Latitude' and 'Longitude' are columns in the dataset
plt.figure(figsize=(12, 8))
sns.scatterplot(x='Longitude', y='Latitude', hue='Complaint Type', data=df,␣
↪alpha=0.5)
plt.title('Geographical Distribution of Service Requests')
plt.show()
```



[19]:
```
# 4. Order the complaint types based on the average ‘Request_Closing_Time’,
# grouping them for different locations.

# Assuming 'Created Date' and 'Closed Date' are columns in the dataset
df['Created Date'] = pd.to_datetime(df['Created Date'])
df['Closed Date'] = pd.to_datetime(df['Closed Date'])

# Calculate the 'Request_Closing_Time' for each row
df['Request_Closing_Time'] = (df['Closed Date'] - df['Created Date']).dt.
 ↪total_seconds() / 3600

# Group by 'City' and 'Complaint Type', calculate the average␣
 ↪'Request_Closing_Time'
grouped_df = df.groupby(['City', 'Complaint Type'])['Request_Closing_Time'].
 ↪mean().reset_index()

# Order the complaint types based on average closing time within each city
ordered_complaints = grouped_df.groupby('City').apply(lambda x: x.
 ↪sort_values('Request_Closing_Time')).reset_index(drop=True)

# Plotting the ordered complaint types based on average closing time
plt.figure(figsize=(10, 10))
```

```
sns.barplot(x='Request_Closing_Time', y='Complaint Type', hue='City',
    data=ordered_complaints)
plt.title('Complaint Types Ordered by Average Closing Time')
plt.xlabel('Average Request Closing Time (hours)')
plt.ylabel('Complaint Type')
plt.show()
```



[ ]:
```
# 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?
```

[21]:
```
import scipy.stats as stats

# Assuming 'Request_Closing_Time' is the column representing response time
complaint_types = df['Complaint Type'].unique()

# Create a list to store complaint types with non-missing and non-infinite␣
 ↪response times
valid_data = []

# Filter and clean the data
for complaint_type in complaint_types:
    subset = df[df['Complaint Type'] == complaint_type]['Request_Closing_Time']
    subset = subset.replace([np.inf, -np.inf], np.nan).dropna()

    if not subset.empty:
        valid_data.append(subset)

# Perform one-way ANOVA
statistic, p_value = stats.f_oneway(*valid_data)

# Check the p-value
alpha = 0.05
if p_value < alpha:
    print(f'Reject the Null Hypothesis. Average response time across complaint␣
 ↪types is not similar (p-value: {p_value})')
else:
    print(f'Fail to reject the Null Hypothesis. Average response time across␣
 ↪complaint types is similar (p-value: {p_value})')
```

```
Reject the Null Hypothesis. Average response time across complaint types is not
similar (p-value: 0.0)
```

[22]:
```
# Assuming 'Complaint Type' and 'Location' are the relevant columns
contingency_table = pd.crosstab(df['Complaint Type'], df['Location'])

# Perform Chi-Square Test
chi2_stat, p_value, dof, expected = stats.chi2_contingency(contingency_table)
```

```python
# Check the p-value
alpha = 0.05
if p_value < alpha:
    print(f'Reject the Null Hypothesis. There is a relationship between the␣
 ↪type of complaint/service and location (p-value: {p_value})')
else:
    print(f'Fail to reject the Null Hypothesis. There is no relationship␣
 ↪between the type of complaint/service and location (p-value: {p_value})')
```

```
Reject the Null Hypothesis. There is a relationship between the type of
complaint/service and location (p-value: 0.0)
```

[ ]:
```python
In practical terms, this means that the distribution of complaint types is not␣
 ↪independent of the location,
and there is an association between the type of complaint/service and the␣
 ↪location where the complaint/
service is requested.
```