Introduction

Phase 1 Project

This is an "end-of-phase" project that I am conducting within the Flatiron School curriculum. For this project, I will be looking at NYC's 311 database which contains records of every 311 Service Request since 2010. With this I will be asking at least five descriptive questions and presenting my findings.

Descriptive Questions

These questions are questions that the data found in this project will answer

- Which Agency handled the most amount of service requests in 2023?
- On Average, How Long Did Each Agency Take to Respond to a Service Request in 2023?
- In 2023, What was the Relationship Between Agency's Response Time and Amount of Service Requests?
- What were the Most Common Types of Service Requests in 2023?
- Which Agency Handled the Most Common Requests in 2023?
- How Many Service Requests did Each Borough Have During 2023?
- What was the Relationship Between Borough Population and Amount of Service Requests?

Data

The data I will be using is found from the NYC Open Data database and contains over 36 million 311 service requests starting from 2010 until July 7, 2024 when I downloaded the data. For this project I will be looking at the data recorded in 2023.

Imports

To start, I am going to import the 4 major python packages I will be using

- Pandas
- NumPy
- Matplotlib
- Seaborn

```
# imports
import pandas as pd
import numpy as np

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

```
# turning my dataset into a pandas dataframe
nyc = pd.read_csv('311_Service_Requests.csv')
```

```
<ipython-input-3-299a66677f63>:2: DtypeWarning: Columns (8) have mixed types. Specify dtype option on import or set low_memory=False.
  nyc = pd.read_csv('311_Service_Requests.csv')
```

Data Cleaning

I was able to do a lot of my data cleaning during the data acquisition at the website containing the data allowing me to filter out columns and entries that I didn't want to use.

```
# I like using tail instead of head because it will show me the highest index I have in my dataframe
nyc.tail()
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	
4929457	56418136	01/01/2023 12:00:46 AM	01/01/2023 01:01:43 AM	NYPD	New York City Police Department	Noise - Residential	Loud Music/Party	Re
4929458	56418795	01/01/2023 12:00:45 AM	01/01/2023 01:24:10 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	
4929459	56416252	01/01/2023 12:00:42 AM	01/01/2023 05:34:15 PM	NYPD	New York City Police Department	Noise - Residential	Loud Music/Party	Re
4929460	56417527	01/01/2023 12:00:09 AM	01/01/2023 12:36:06 AM	NYPD	New York City Police Department	Illegal Fireworks	NaN	

Next thing for me to do was to prepare my data to answer my questions. My goal is to have all the data in formats and types that I can later call with ease when answering my questions and creating visualizations.

First, I want to get the desired timeframe, which for this project will be the 2023 calender year, turn the dates into a datetime format, and creating a new category that describes the amount of time taken to complete each service request

```
# Turning the dates into a datetime format
date_format = '%m/%d/%Y %I:%M:%S %p'
nyc['Closed Date'] = pd.to_datetime(nyc['Closed Date'], format=date_format)
nyc['Created Date'] = pd.to_datetime(nyc['Created Date'], format=date_format)


# filtering to specifically 2023
nyc_2023 = nyc[nyc['Created Date'].dt.year == 2023]


# finding the elapsed time between the open and close dates
nyc_2023['time_to_complete'] = (nyc_2023['Closed Date'] - nyc_2023['Created Date']).dt.total_seconds()
```

```
<ipython-input-7-cc87f9bcf167>:2: 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-c
  nyc_2023['time_to_complete'] = (nyc_2023['Closed Date'] - nyc_2023['Created Date']).dt.total_seconds()
```

I also want to create my color map, which is a gradient of colors that i can apply to my graphs, at this point. I want to use the colors of the NYC flag which is blue, white and orange stripes.

Also I will import all the logos for each individual agency which will be used in the creation of the graphs

```
# importing the correct package to create the color map
from matplotlib.colors import LinearSegmentedColormap
# Define the colors: blue to white to orange
colors = ["blue", "white", "orange"]
# Create the colormap
cmap = LinearSegmentedColormap.from_list("blue_white_orange", colors)
```

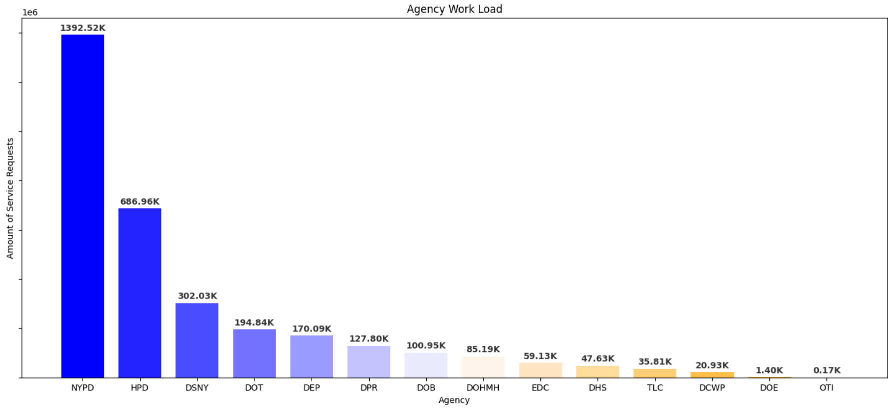
```
# from PIL import Image  # Python Imaging Library

# # Load Images
# DCWP_image = Image.open('DCWP.jpg')
# DEP_image = Image.open('DEP.jpg')
# DHS_image = Image.open('DHS.png')
# DOB_image = Image.open('DOB.jpg')
# DOE_image = Image.open('DOE.png')
# DOHMH_image = Image.open('DOHMH.png')
# DOT_image = Image.open('DOT.png')
# DPR_image = Image.open('DPR.png')
# DSNY_image = Image.open('DSNY.jpg')
# HPD_image = Image.open('HPD.png')
# EDC_image = Image.open('EDC.png')
# NYPD_image = Image.open('NYPD.png')
# OTI_image = Image.open('OTI.jpg')
# TLC_image = Image.open('TLC.png')
```

Which Agency handled the most amount of service requests in 2023?

For this question I am going to count each instance where the Agency appears and graph it within a barplot

```
# getting the value counts for each agency instance
nyc_2023_agency = nyc_2023['Agency'].value_counts()
# Convert each value to a string that signifies it being measured in thousands
nyc_2023_agency_in_thousands = nyc_2023_agency.apply(lambda x: f"{x / 1_000:.2f}K")


fig, ax = plt.subplots(figsize=(18, 10))
fig.subplots_adjust(bottom=0.3)

# Plot bars with colormap
bars = ax.bar(nyc_2023_agency.index, nyc_2023_agency.values, color=plt.cm.get_cmap(cmap, len(nyc_2023_agency))(np.linspace(0, 1, len(nyc_20

# Annotate bars with values
for bar, value in zip(bars, nyc_2023_agency_in_thousands.values):
    ax.annotate(f"{value}",
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),
                textcoords="offset points",
                ha='center', va='bottom', fontweight='bold', color='#383838')

# Set plot title and labels
plt.title('Agency Work Load')
plt.xlabel('Agency')
plt.ylabel('Amount of Service Requests')
plt.yticks(visible=False)
plt.savefig('Agency Work Load.png')
plt.show()
```

```
<ipython-input-11-29e5dfe63c94>:5: MatplotlibDeprecationWarning: The get_cmap function
  bars = ax.bar(nyc_2023_agency.index, nyc_2023_agency.values, color=plt.cm.get_cmap(cm
```

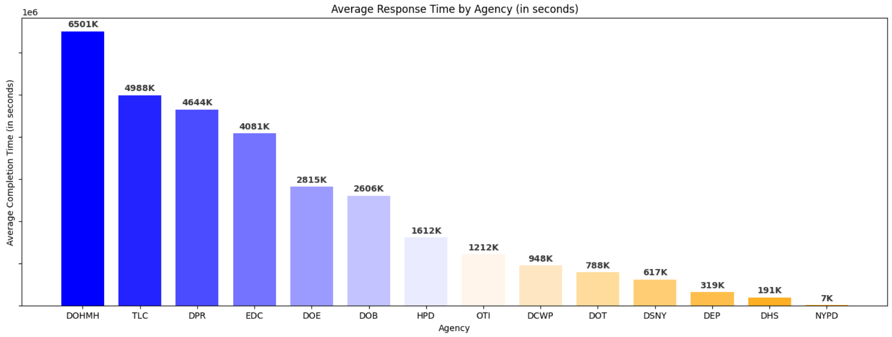


On Average, How Long Did Each Agency Take to Respond to a Service Request in 2023?

For this question, I am going to group the data by on the agency and then calculate the mean elasped time

```
# Values for following graph
nyc_2023_average_time = nyc_2023.groupby('Agency')['time_to_complete'].mean()
nyc_2023_average_time = nyc_2023_average_time.sort_values(ascending=False)
# Convert each value to a string that signifies it being measured in thousands
nyc_2023_agency_response_in_thousands = nyc_2023_average_time.apply(lambda x: f"{x / 1_000:.0f}K")


# plot size
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# data
bars = ax.bar(nyc_2023_average_time.index, nyc_2023_average_time.values, color=plt.cm.get_cmap(cmap, len(nyc_2023_agency))(np.linspace(0, 1

# Annotate bars with values
for bar, value in zip(bars, nyc_2023_agency_response_in_thousands.values):
    ax.annotate(f"{value}",
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom', fontweight='bold', color='#383838')
#labels
plt.title('Average Response Time by Agency (in seconds)')
plt.xlabel('Agency')
plt.ylabel('Average Completion Time (in seconds)')
plt.yticks(visible=False)
# plt.savefig('Average Response Time by Agency.png')
plt.show()
```

```
<ipython-input-13-fb0667cd9e9f>:6: MatplotlibDeprecationWarning: The get_cmap function
  bars = ax.bar(nyc_2023_average_time.index, nyc_2023_average_time.values, color=plt.cm
```



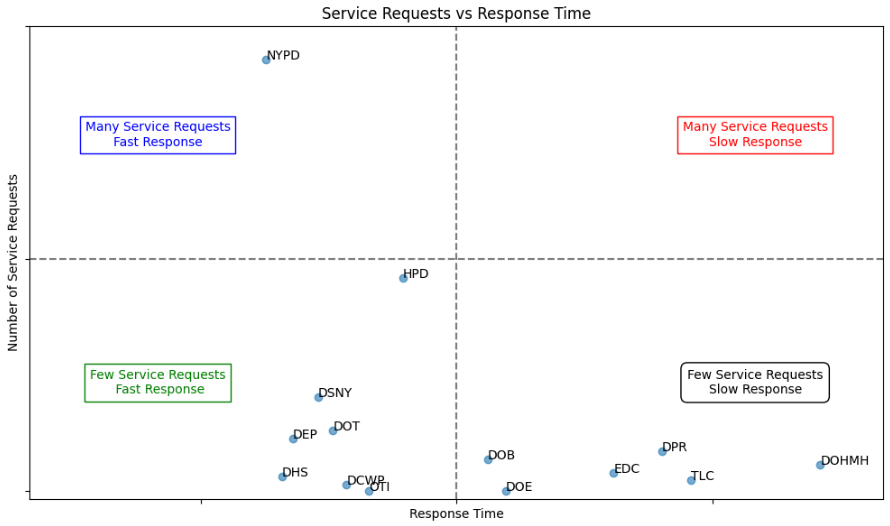
In 2023, What was the Relationship Between Agency's Response Time and Amount of Service Requests?

My immediate thought was "Are the Agencies that are getting a lot of service requests taking a long time to complete them?" In other words, is a specific agency being overworked?

To answer this, I decided to plot the data from the two previous charts against each other in a scatter plot.

```
# Getting the values from Question 1
agency_counts = nyc_2023['Agency'].value_counts()

# Getting the values from Question 2
mean_response_time = nyc_2023.groupby('Agency')['time_to_complete'].mean()

# Combine the values into a single DataFrame
combined_counts = pd.DataFrame({
    'Complaint Counts': agency_counts,
    'Mean Response Time': mean_response_time
})


#standardizing my counts to make easier to visualize
counts_time_standardized = (combined_counts['Mean Response Time'] - combined_counts['Mean Response Time'].mean())/1000000
counts_agency = combined_counts['Complaint Counts']/1000000
```

```
#setting cross lines
complaints_threshold = 0
response_time_threshold = 0.75

fig, ax = plt.subplots(figsize=(10, 6))

# Scatter plot
ax.scatter(y=counts_agency, x=counts_time_standardized, alpha=0.6)

# Add vertical and horizontal lines to create quadrants
ax.axvline(complaints_threshold, color='gray', linestyle='--')
ax.axhline(response_time_threshold, color='gray', linestyle='--')

# Adding the Agency tag to each point
for i, txt in enumerate(combined_counts.index):
    ax.annotate(txt, (counts_time_standardized[i], counts_agency[i]))

# Adding labels for each quadrant
ax.text(-3.5, 1.15, "Many Service Requests\nFast Response", color="blue", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="blue"))
ax.text(3.5, 1.15, "Many Service Requests\nSlow Response", color="red", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="red"))
ax.text(-3.5, 0.35, "Few Service Requests\n Fast Response", color="green", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="green"))
ax.text(3.5, 0.35, "Few Service Requests\nSlow Response", color="black", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="black", boxstyle="round,pad=0.5"))

# Set labels, title, and y-axis limits and ticks
ax.set_ylabel('Number of Service Requests')
ax.set_xlabel('Response Time')
plt.title('Service Requests vs Response Time')
ax.set_ylim(-.025, 1.5)
ax.set_yticks([0, 0.75, 1.5])
ax.set_xticks([-3, 0, 3])
ax.set_xlim (-5,5)
plt.xticks(visible=False)
plt.yticks(visible=False)
plt.tight_layout()
# plt.savefig('Service Requests vs Response Time.png')
plt.show()
```



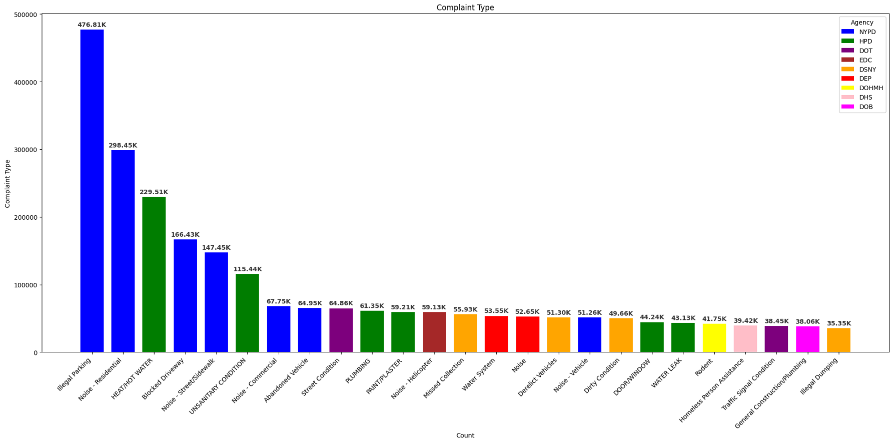
What were the Most Common Types of Service Requests in 2023?

Which Agency Handles the Most Common Requests in 2023?

For this question I wanted to simply graph the value counts for the top 25 most common service requests and apply color coding to the bars with each color representing a different agency

```
# Getting the top 25 most common service requests
nyc_2023_complaint_type = nyc_2023['Complaint Type'].value_counts().head(25)


# creating a dictionary to store data
nyc_complaint_agency = {}

# Loop through each unique complaint type
for i, complaint_type in enumerate(nyc_2023['Complaint Type'].value_counts().head(25).index):
    mask = nyc_2023['Complaint Type'] == complaint_type
    nyc_2023_complaint_type = nyc_2023[mask]
    nyc_complaint_agency[complaint_type] = nyc_2023_complaint_type['Agency'].value_counts()


data = []
for complaint_type, agency_series in nyc_complaint_agency.items():
    for agency, count in agency_series.items():
        data.append({'Complaint Type': complaint_type, 'Agency': agency, 'Count': count})

# Convert the list of dictionaries to a DataFrame
nyc_complaints_agency_df = pd.DataFrame(data)


# creating the labels that go onto of the bars
nyc_complaints_agency_df['Count_Label'] = nyc_complaints_agency_df['Count'].apply(lambda x: f"{x / 1_000:.2f}K")


# setting the colors for each agency
agency_colors = {
    'NYPD': 'blue',
    'HPD': 'green',
    'DSNY': 'orange',
    'DOT': 'purple',
    'DEP': 'red',
    'DPR': 'cyan',
    'DOB': 'magenta',
    'DOHMH': 'yellow',
    'EDC': 'brown',
    'DHS': 'pink',
    'TLC': 'gray',
    'DCWP': 'black',
    'DOE': 'white',
    'OTI': 'lime'
}

# Apply colors based on 'Agency' column
colors = nyc_complaints_agency_df['Agency'].map(agency_colors)
```

```
fig, ax = plt.subplots(figsize=(20, 10))
bars = ax.bar(nyc_complaints_agency_df['Complaint Type'], nyc_complaints_agency_df['Count'], width= 0.75, align='center', color=colors)
for bar, value in zip(bars, nyc_complaints_agency_df['Count_Label']):
    ax.annotate(f"{value}",
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom', fontweight='bold', color='#383838')
# Create legend using unique agencies and their corresponding bar handles
handles = []
labels = []
for bar, agency in zip(bars, nyc_complaints_agency_df['Agency']):
    if agency not in labels:
        handles.append(bar)
        labels.append(agency)

ax.legend(handles=handles, labels=labels, title="Agency")
plt.title('Complaint Type')
plt.xticks(rotation=45, ha='right')
plt.xlabel('Count')
plt.ylabel('Complaint Type')
plt.tight_layout()
plt.savefig('Complaint Type.png')
plt.show()
```

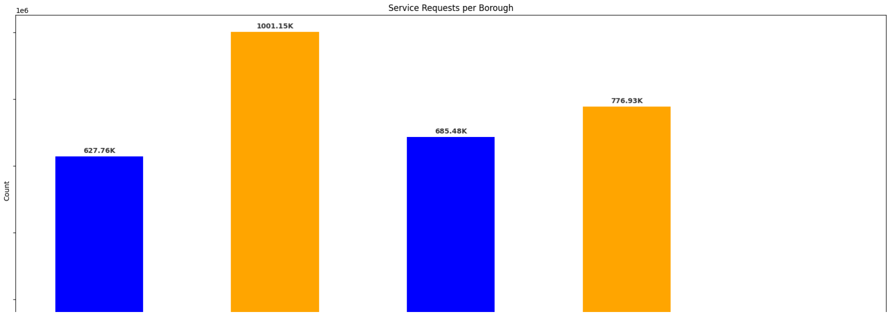


How Many Service Requests did Each Borough Have During 2023?

For this, I am going to use the 'Borough' column and count every instance

```
# creating a dictionary with borough population, abreviation and count
boroughs = {'Bronx':{
            'Pop' : 1356476,
            'ABV' : 'BRX',
            'C_Count': 627761},
            'Brooklyn':{
            'Pop' : 2561225,
            'ABV' : 'BKN',
            'C_Count': 1001154},
            'Manhattan':{
            'Pop' : 1597451,
            'ABV' : 'MHTN',
            'C_Count': 685483},
            'Queens':{
            'Pop' : 2252196,
            'ABV' : 'QUEN',
            'C_Count': 776930},
            'Staten Island':{
            'Pop' : 490687,
            'ABV' : 'STN ISLD',
            'C_Count': 128079}}

# turning the dictionary into a dataframe
boroughs_df = pd.DataFrame(boroughs).T
boroughs_df.reset_index(inplace=True)
boroughs_df.rename(columns={'index': 'Borough'}, inplace=True)


# creating the labels that go onto of the bars
boroughs_df['C_Count_Label'] = boroughs_df['C_Count'].apply(lambda x: f"{x / 1_000:.2f}K")


fig, ax = plt.subplots(figsize=(18, 8))
bars = ax.bar(boroughs_df['Borough'], boroughs_df['C_Count'], width= 0.5, align='center',color=plt.cm.get_cmap(cmap, 2)(np.linspace(0, 1, 2
for bar, value in zip(bars, boroughs_df['C_Count_Label']):
    ax.annotate(f"{value}",
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),
                textcoords="offset points",
                ha='center', va='bottom', fontweight='bold', color='#383838')
plt.title('Service Requests per Borough')
plt.ylabel('Count')
plt.xlabel('Borough')
plt.yticks(visible=False)
plt.tight_layout()
plt.savefig('Service Requests per Borough.png')
plt.show()
```

```
<ipython-input-25-733e59cd5dfe>:2: MatplotlibDeprecationWarning: The get_cmap function
  bars = ax.bar(boroughs_df['Borough'], boroughs_df['C_Count'], width= 0.5, align='cent
```



What was the Relationship Between Borough Population and Amount of Service Requests?

For this, I took the values from the previous chart and placed them in a scatter plot against the population count.

```
#standardizing my counts to make easier to visualize
boroughs_pop_standardized =  (boroughs_df['Pop'] - boroughs_df['Pop'].mean())/1000000
boroughs_count_standardized = (boroughs_df['C_Count'] - boroughs_df['C_Count'].mean())/1000000


#setting cross lines
complaints_threshold = 0
response_time_threshold = 0

fig, ax = plt.subplots(figsize=(10, 6))

# Scatter plot
ax.scatter(x=boroughs_pop_standardized, y=boroughs_count_standardized, alpha=0)

# Add vertical and horizontal lines to create quadrants
ax.axvline(complaints_threshold, color='gray', linestyle='--')
ax.axhline(response_time_threshold, color='gray', linestyle='--')

# Adding the tag to each point
for i, txt in enumerate(boroughs_df["Borough"]):
    ax.annotate(txt, (boroughs_pop_standardized[i], boroughs_count_standardized[i]), fontsize=25, va='top')

# Adding labels for each quadrant
ax.text(-1, 0.4, "Many Service Requests\nSmall Population", color="red", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="red"))
ax.text(1, 0.4, "Many Service Requests\nLarge Population", color="blue", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="blue"))
ax.text(-1, -0.4, "Few Service Requests\nSmall Population", color="black", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="black"))
ax.text(1, -0.4, "Few Service Requests\nLarge Population", color="green", va="center", ha="center",
        bbox=dict(facecolor="none", edgecolor="green", boxstyle="round,pad=0.5"))

# Set labels, title, and y-axis limits and ticks
ax.set_xlabel('Borough Population')
ax.set_ylabel('Borough Service Requests')
plt.title('Service Requests vs Population')
ax.set_xlim(-1.5, 1.5)
```