Project 1 - Analyzing Millions of NYC Fire Incident Dispatch Data

Alessandro Sciorilli

1. AWS Setup

I start an EC2 instance using the Amazon Linux 2023 AMI. Since my objective is to load more than 5 million records into Elasticsearch, I select the t2.2xlarge instance type, which offers 8 vCPUs and 32 GiB of memory. As a second step, I create a domain called "project1" in the Amazon OpenSearch Service, with the following characteristics: an instance type of r6g.large.search, distribution across 3 availability zones, and 3 nodes, each with an EBS storage size of 1000 GiB. Next, I access the EC2 instance via a web browser. Within it, I create a directory named "project01", which contains the Dockerfile, a text file with the required dependencies, and a sub-directory named "src" where the Main.py code is stored.

requirements.txt

The file contains the necessary dependencies for this project: the Sodapy and Requests libraries. Sodapy enables interaction with the Socrata Open Data API (SODA), allowing to retrieve the data from the Fire Incident Dispatch Dataset. The Requests library is also essential to connect with Elasticsearch.

Dockerfile

Since my python code (Main.py) is hosted on the virtual computer (EC2), I need to create a copy of Main.py inside the Docker container, in a directory that I call “app”**.**

FROM python:3.9	*New Docker image is built on top of Python image version 3.9*
WORKDIR /app	*I set the working directory to “/app”*
COPY . /app	*I copy the Main.py and all project files inside the app directory*
RUN pip install -r requirements.txt	*I install the dependances listed in the requirements.txt file (sodapy and requests)*
ENTRYPOINT ["python", "src/main.py"]	*The entrypoint specifies the default command that should be executed when the Docker container starts. In this case, using the “python” interpreter, the Docker runs the main.py code stored in the directory called “src”.*

Main.py

The code is designed to extract data from Socrata dataset, transform the data to fit the structure of Elasticsearch index and load the transformed data into Elasticsearch for querying and analysis. Argument parsing is employed to eliminate the need for hardcoding environment variables and to define the number and size of pages as per specific requirements when executing the Dockerfile.

In particular, I highlight the implementation of a “for” loop to iterate the loading of the rows over the specified number of pages (args.num_pages). I use the “offset” argument of the get function to specify the number of records to skip before starting to retrieve data. In this case, “offset” must be set equal to args.page_size*page, to ensure that any records already loaded in previous pages are skipped, thus

preventing duplication. Moreover, I incorporate a nested “for” loop to filter out rows with null values for fields “starfire_incident_id” and “incident_datetime”.

```
for page in range(args.num_pages):
    rows = client.get(DATASET_ID, limit=args.page_size, offset=args.page_size*page )
    es_rows=[]

    for row in rows:
        if row.get("starfire_incident_id") is None or row.get("incident_datetime") is None:
            print("Skipping row due to missing required fields")
            continue
```

Docker build – Docker run

Inside the terminal, with docker build function I create a docker image called bigdataproject1:1.0 .

```
docker build -t bigdataproject1:1.0 .
```

docker build	*Creates container images*
-t	*Indicates the tag*
bigdataproject1	*Name of my image*
1.0	*Version of the created image*
.	*Shorthand for the current working directory*

In the Docker run function, I configure the environment variables and access credentials necessary for the data loading process. To achieve the upload of more than 5 million data, I set the page size to 1,000 records per page for a total of 5,500 pages. I deliberately choose a relatively small page size (1,000 records per page) to minimize the risk of data loss in case of a code crash during the loading operation. By using a smaller page size, I ensure that even if the code crashes before completing the loading of a page, I won't lose the entire batch of data. For instance, if I had chosen a larger page size, such as 1 million records per page, and a crash occurred before the first page completed, all the data intended for that page would be lost.

```
docker run -e INDEX_NAME="fire" -e DATASET_ID="8m42-w767" -e APP_TOKEN="my_token" -e
ES_HOST="https://search-project01-i7ozes2ad3xlacia3m4wzakofm.us-east-2.es.amazonaws.com" -e
ES_USERNAME="asciorilli" -e ES_PASSWORD="my_password" bigdataproject1:1.0 --pagesize=1000 -
num_pages=5500
```

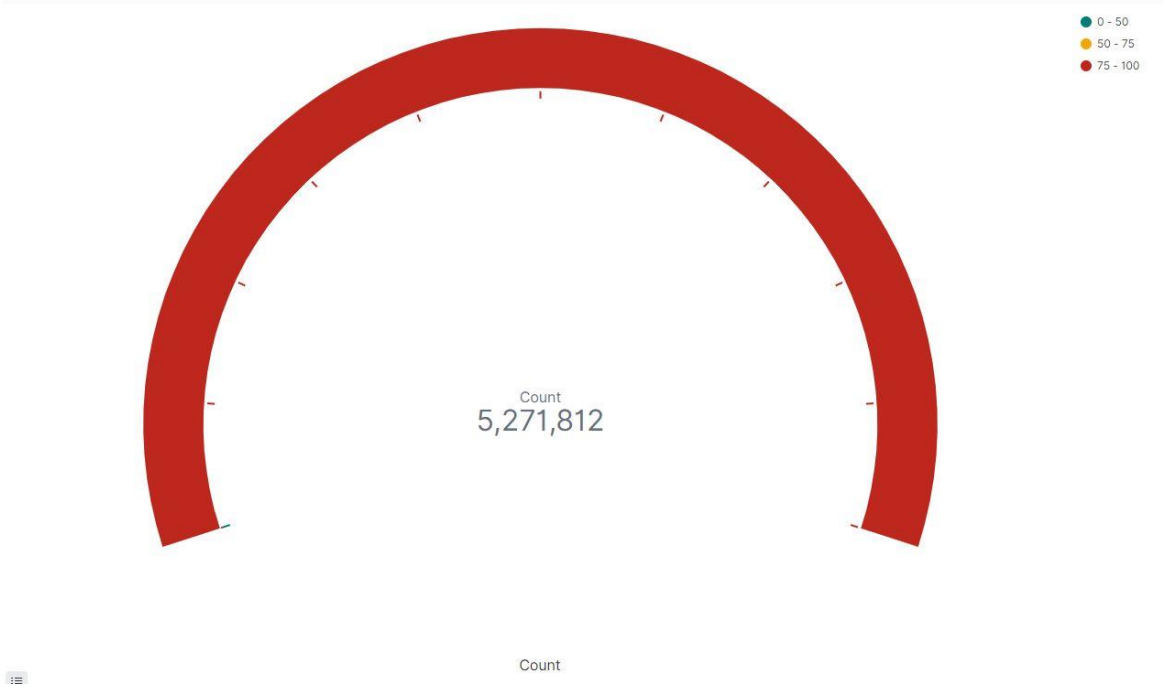
2. NYC Fire Incident Dispatch Data Analysis

For this analysis, I select a set of 12 specific fields, which I believe are the most important to analyze to get a clear understanding of the fire incident scenario in New York City. Specifically, I focus on data pertaining to emergency response times, the nature of emergencies, and the resources deployed. I analyze these metrics based on their geographic distribution, aiming to accentuate the performance and management disparities across the 5 boroughs of New York City. Adopting a geographical perspective proves valuable in identifying both vulnerable areas in need of improved performance standards. Below is reported

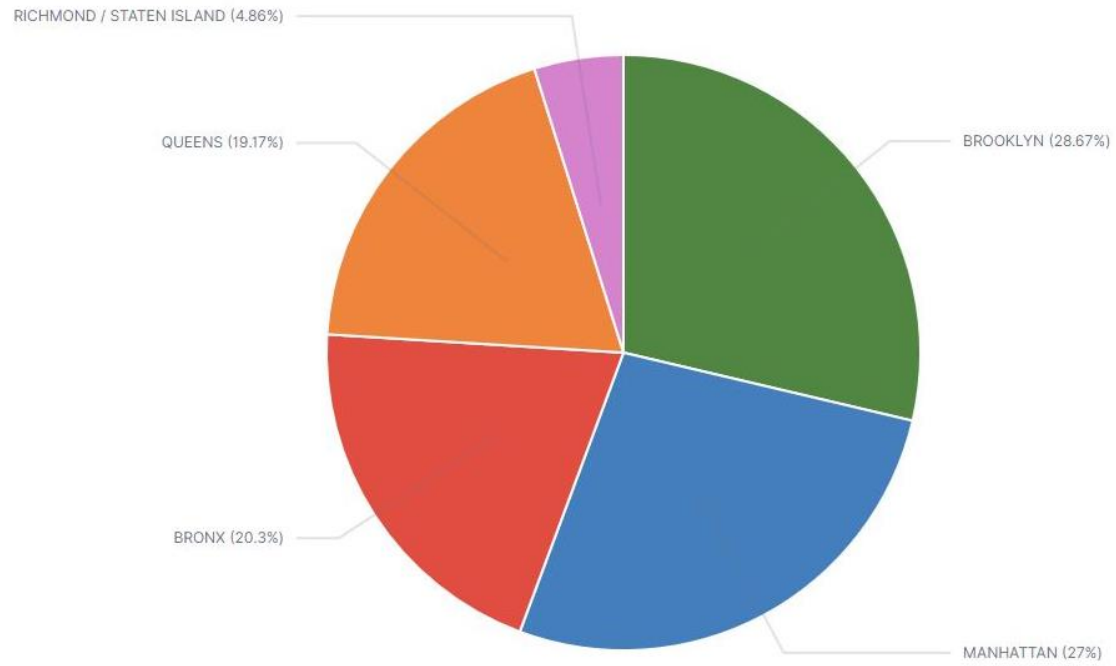
- The list of the selected fields, accompanied by a brief description
- The Gauge Chart, displaying the total count of records the analysis is based on (5,271,812 records)

Field	Description
starfire_incident_id	Unique identifier for each fire incident in the dataset
incident_datetime	Date and time when the fire incident occurred
incident_borough	Borough of New York City where the incident occurred
alarm_source_description_tx	Source of the alarm or report that led to the dispatch of fire units
incident_classification_group	Categorizes incidents into groups based on their nature or type
dispatch_response_seconds_qy	Number of seconds it took for the dispatched units to respond to the incident after receiving the call
incident_response_seconds_qy	Number of seconds it took for the responding units to arrive at the incident location from the time of dispatch
incident_travel_tm_seconds_qy	Number of seconds it took for units to travel to the incident location from their respective stations
engines_assigned_quantity	Quantity of fire engine units that were assigned to respond to the incident
ladders_assigned_quantity	Quantity of ladder units assigned to the incident
other_units_assigned_quanti ty	Number of additional firefighting or emergency response units assigned to the incident
zipcode	ZIP code of the incident location

Plot 1 – Gauge Chart

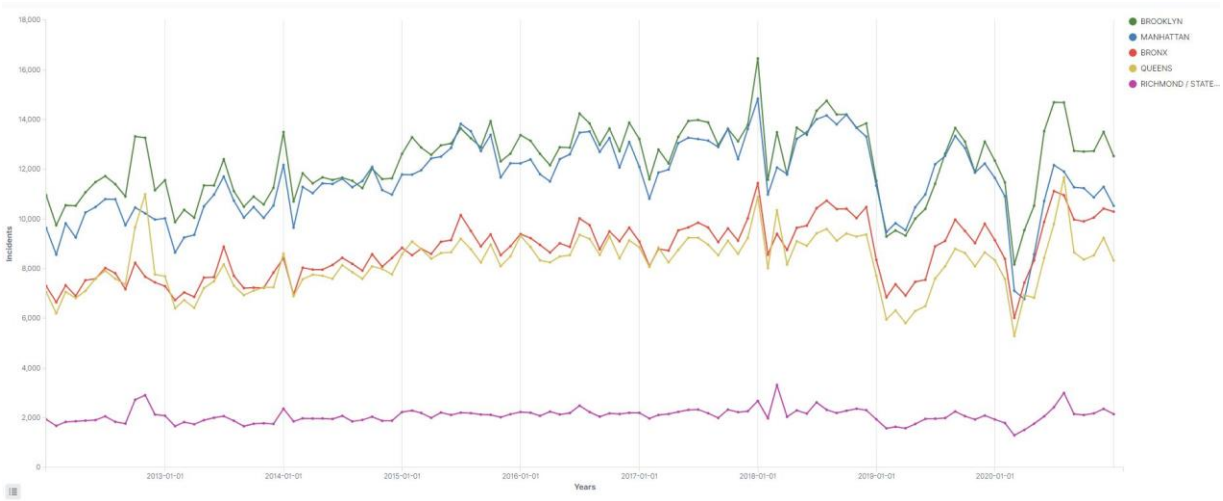


Plot 2 - Distribution of Incidents Across New York City Boroughs



To begin with, I create a simple pie chart to visualize the incident rates in the different boroughs of New York City. We can observe that Brooklyn holds the highest incident rate, accounting for 28.67% of the total, closely followed by Manhattan at 27%. Moving further away from the city center, the Bronx represents 20.3% of incidents and Queens stands at 19.17%. Interestingly, Richmond/Staten Island, being more residential and less densely populated, shows the smallest share of incidents at 4.86%.

Plot 3 - Fire Incidents in New York City Boroughs: 2012-2021

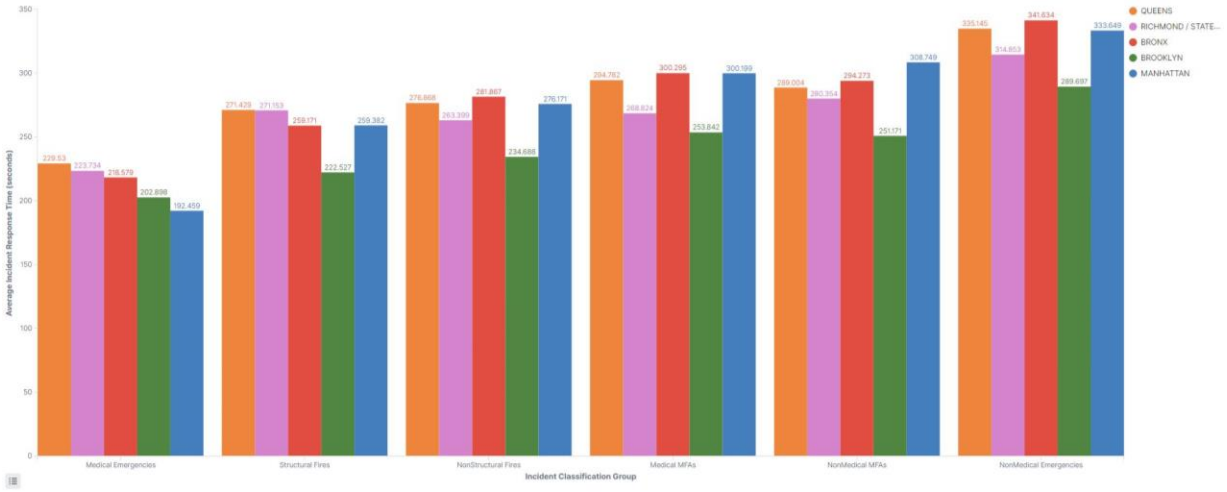


As second step, I create a line chart to visualize the trend of fire incidents over a span of nine years. From January 2013 to January 2018, every borough—except for Richmond/Staten Island—experiences a mild and steady increase in incidents. This consistency suggests that there are no significant external factors impacting the frequency of fire incidents in these boroughs during this period. However, a noticeable change in trend begins in the last months of 2017, with the graph displaying more pronounced fluctuations across all boroughs. In particular, a sharp peak in reported fire incidents is observed in January 2018. This surge is followed by a significant drop at the start of 2019, only to see another rebound from the second quarter of 2019 onwards.

An intriguing trend is evident in the second quarter of 2020, when the fire incidents reach their lowest point in the entire time series: approximately 5,000 in Queens and 8,000 in Brooklyn. Yet, the latter half of 2020 marks another significant increase in incidents. This shows that the trend of incidents in recent years becomes increasingly unstable, making the management of emergencies and rescue personnel more challenging.

Notably, Richmond/Staten Island consistently displays a lower trend over the years, with no significant peaks or troughs.

Plot 4 - Average Incident Response Times by Emergency Type Across NYC Boroughs

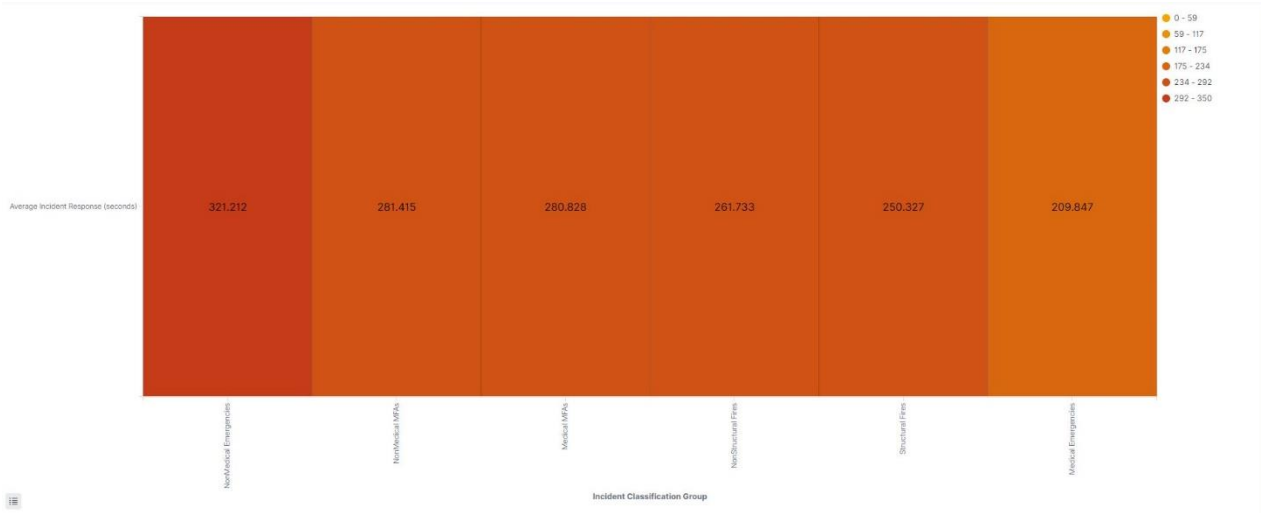


Analyzing the average response time for different types of emergencies across New York City's boroughs offers crucial insights into the efficiency of emergency services. At a glance, the most rapid response times across all boroughs are recorded for medical emergencies. Manhattan stands out as the top performer in this category, with an impressive average response time of just 192.459 seconds. In contrast, Queens lags slightly with an average time of 229.53 seconds.

Non-Medical Emergencies generally require a longer response time. Here, the Bronx struggles with an average time of 341.634 seconds, making it the slowest among the boroughs. On the other hand, Brooklyn emerges as the quickest responder for this category, averaging 289.697 seconds.

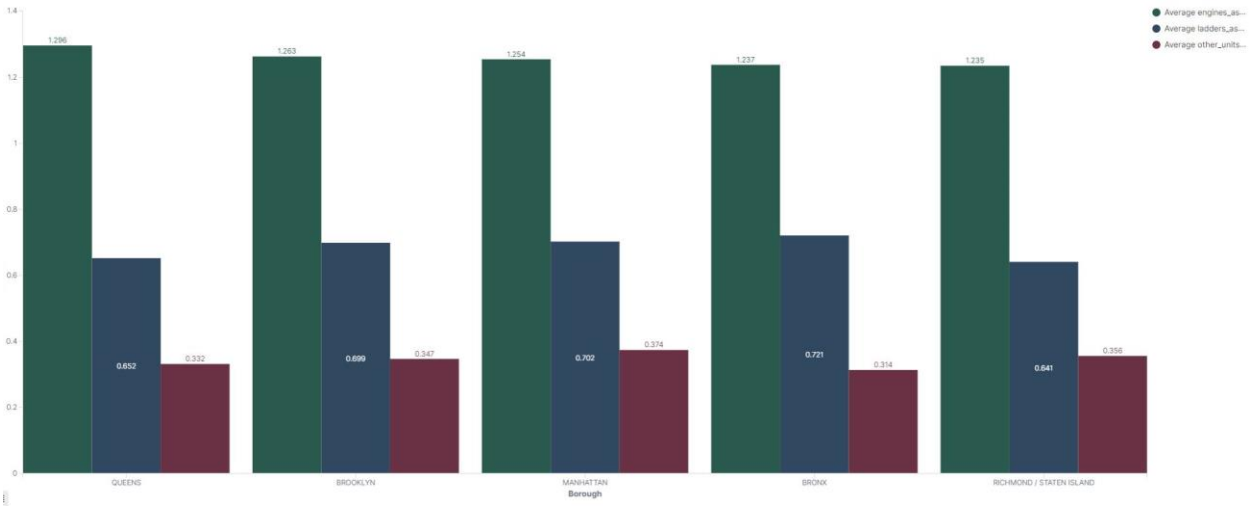
Excluding medical emergencies, Brooklyn consistently proves to be the most efficient borough in terms of response speed for all emergency categories. Conversely, the Bronx and Queens often have longer response times, indicating potential areas for improvement.

Plot 5 – Heatmap of average Incident Response Times by Emergency Type Across NYC Boroughs



The proposed heatmap provides further insights into the average incident response times for each incident classification group. Medical emergencies have the quickest average response time at 209.847 seconds, while Non-Medical Emergencies have the slowest, averaging 321.212 seconds.

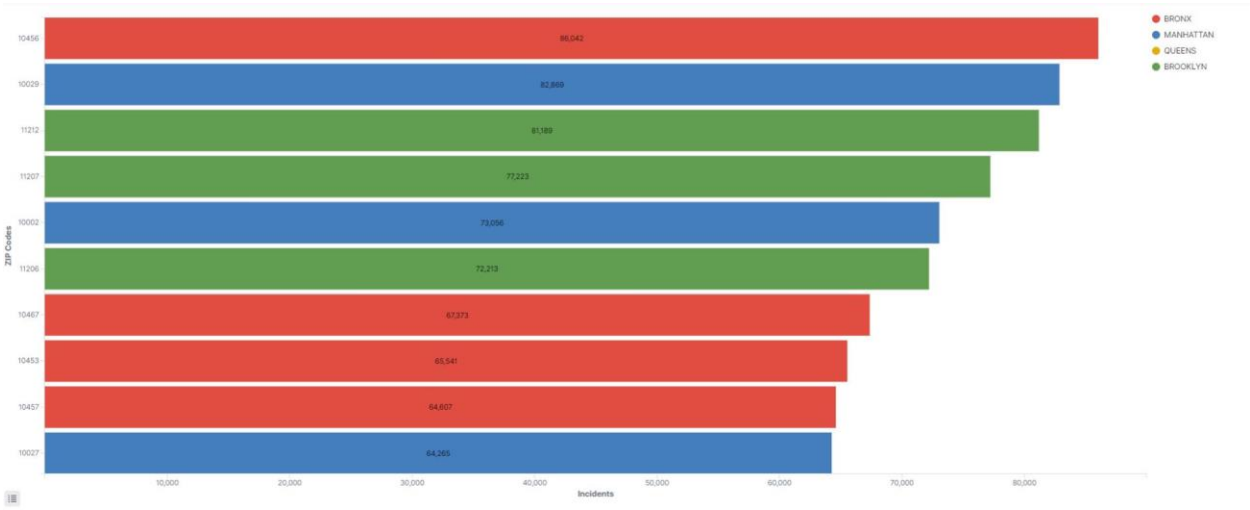
Plot 6 - Average Engines, Ladders and Other Units Assigned Per Incident Across NYC Boroughs



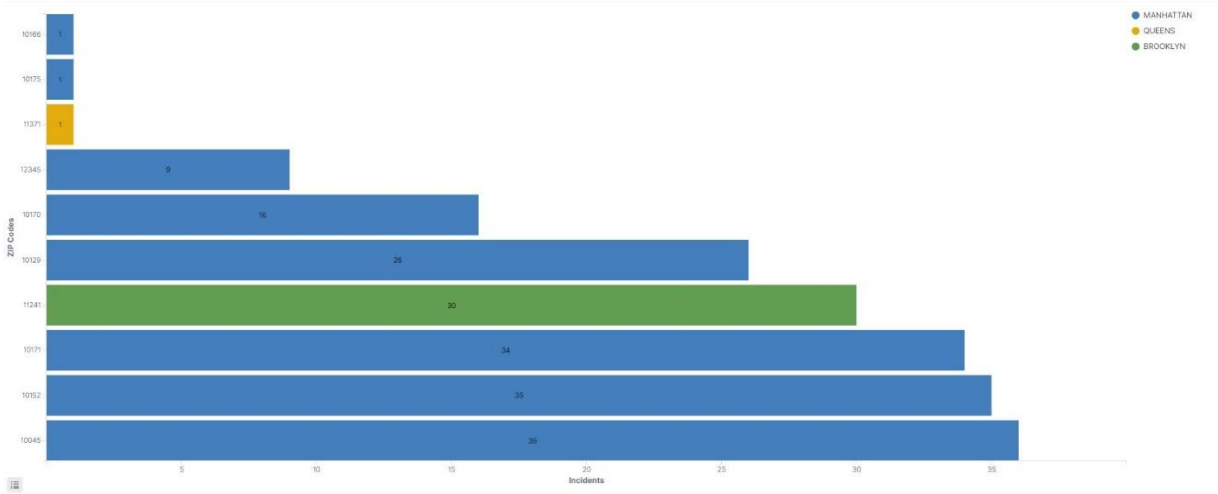
This histogram chart illustrates the average resources (in terms of engines, ladders and other units) each borough allocates to respond to incidents. Even though Brooklyn has the highest number of emergency situations, it is Queens that deploys the highest average number of engines per incident, at 1.296. This could indicate one of two things: either the incidents in Queens are, on average, more severe, requiring the frequent dispatch of more than two units per call, or Queens has more resources at its disposal, allowing it to send more units to support regardless of the severity of the incident. If the latter is true, it might be advisable to implement a resource management efficiency policy, especially given Queens'

underperformance in response time, as highlighted in Plot 4. It is also worth noting that Manhattan has the highest average of "other/non-fire" units assigned to each incident. This number correlates with its densely populated demographic.

Plot 7 - Zip Codes with the Highest Rate of Fire Accidents



Plot 8 - Zip Codes with the Lowest Rate of Fire Accidents



Lastly, I would like to analyze in more detail both the neighborhoods with the highest number of incidents and those with the least. Plot 7 visualizes the ZIP codes with the highest rates of fire accidents across four NYC boroughs. ZIP code 10456, located in the Bronx, stands out with the highest number of incidents, surpassing 86,000 over a 10 years' time frame. This is closely followed by ZIP code 10029 in Manhattan, with approximately 83,000 incidents. Generally, we can observe that the Bronx has a notable presence among the top zip codes for fire incidents, suggesting that this borough may have particular challenges or vulnerabilities related to fire safety.

On the other hand, Plot 8 offers a visualization of the ZIP codes with the lowest rates of fire accidents. We can discern that certain areas in these boroughs exhibit a notably low frequency of fire-related incidents. Specifically, ZIP codes 10166, 10175, and 11371 each record a mere single fire incident, indicating an admirable safety record in these regions. It's clear that the borough of Manhattan (in blue) dominates this chart with the most zip codes reflecting low fire accident rates.

Even though it's crucial to emphasize that the absolute number of fire incidents reported in each ZIP code can be due to various reasons, primarily the size of the urban area covered by that ZIP code and the population density, observing these charts is vital. It helps identify and address high-risk areas and efficiently allocate resources. Moreover, examining zones with the fewest incident reports could provide insights into best practices or conditions that could be replicated in higher-risk areas to mitigate fire incidents.