A Statistical Analysis of NYC Drinking Water Tank Inspections and Audits Compliance

By

Edward Del Rosario

A Research Project Submitted in
Partial Fulfillment of the
Requirements for the Degree of Master of Science in Statistics
Specialization in Financial Statistics and Risk Management (or Data Science for MSDS)
At
Rutgers, the State University of New Jersey

December 2025

1 Abstract

Upon closer inspection, the quality of NYC drinking water receives more violations than official reports and public officials' statements would suggest. My partner Ananya Chepuri and I conducted an analysis on the publicly held data provided by the official NYC Health Department to further investigate the proportion and nature of health violations found for drinking water tanks in NYC. Of the 16,541 observations found in the dataset, a large proportion of buildings inspected throughout NYC were given at least one violation. Most of these violations documented are due to a failure to submit an inspection report by the annual due date. Even more concerning were the number of buildings that had not only more repeat inspections, but also more repeated violations. An overwhelming number of these buildings were found to be in Manhattan, as other boroughs showed a much lower number of violations found, even when taking account the population of each borough.

2 Table of Contents

3 List of Tables

4 List of Graphs

5 Body

5.1.1 Background

According to the 2023 NYC official Drinking Water Supply Quality report, the reservoirs, feeder streams, and wells which are the sources of NYC's drinking water have passed all standards. These standards include conventional physical and chemical parameters such as Alkalinity, Aluminum, Calcium, Chloride, and Lead. No major microbial samples taken from the tap water are a violation, and the small traces found are naturally present in the environment. Small amounts of Cryptosporidium and Giardia were detected in NYC reservoirs, but these are found pre-treatment. There are no sources that suggest that the amounts of these substances are substantial post-treatment. Between chemicals, organic compounds, microbials, and Lead and Copper samples, the tap water in NYC passed on all counts.[1] From these metrics, it may seem as if the tap water in NYC is free of contaminants.

There is more to the drinking water than just the quality and purity at the source, however. According to NYC.gov, buildings with more than six floors have pumps which push water to rooftop tanks. The water stays in these tanks until needed. Water tank owners must conduct annual inspections and perform water quality sampling. Inspectors must be a licenced professional, whether that's a plumber, engineer, or supervised by a plumber or engineer.[2] While this may sound like a robust policy to ensure the water's quality in these tanks, the data suggests otherwise.

5.1.2 Dataset

To further investigate the quality of the water from these drinking water tanks, we took a public dataset from the NYC.gov website. This dataset, titled: "NYC Drinking Water Tank Inspections and Audits Compliance Results", will be the main data source used for this investigation. It includes the reported inspections and violations found for different buildings throughout NYC.[3] With over 16.5 thousand observations, the raw dataset has the following 23 columns:

- BIN (Building Identification Number)
- House Number
- Street Name
- Violation Text
- Compliance Year
- Date of Occurrence
- Summons Number
- Borough, Block, Lot
- Council District of the building with the drinking water tank
- Census Tract of the location of the

1 (NYC Environmental Protection [NYC DEP], 2023)
2 ("Drinking Water Tank Inspection Reporting," 2025)
3 (Department of Health and Mental Hygiene [DOHMH], 2025)

- Zip Code
- Borough
- Status (of a building's drinking water tank(s))
- Number of Drinking Water Tanks
- Activity Type
- Activity Year
- Violation Code
- Law Section
- Number (BBL in the dataset) of the building with the drinking water tank
- Longitude
- Latitude
- Community Board of the building with the drinking water tank
- building with the drinking water tank
- NTA (Neighborhood Tabulation Areas) Code of the location of the building with the drinking water tank

Each of the observations in this dataset are their own individual drinking water tank oversight, meaning that some of these will be an investigation alone, while others will also have a violation. The observations fall between 2021 and 2024, but there is little coverage for 2021 and 2024. Most observations fall between 2022 and 2023. The below table shows the number of observations from different years as a table:

Table 1: Number of Observations by Year

Year	Number of Observations
2021	5623
2022	5509
2023	5391
2024	30

While this is accessible to the public and is easy to pull from with their API and is in structured XML, there are still some obstacles to overcome in order to draw conclusions from these data. The dataset is also periodically updated, so the most recent entries in the dataset which show results from 2024 are sparse and will largely be ignored for this investigation. As there are observations where an inspection is held but no violation is found, our investigation must determine which rows showed a violation and which did not, as well as whether there were multiple inspections and/or violations for a given building.

5.1.3 Objectives

Many conclusions can be drawn from this information, but our investigation will focus on the following objectives. Our objectives can be summed up with the following questions:

- Are there seasonal differences in violations?
- Which boroughs have more violations?
- Are there specific buildings with more inspections or violations?

- How many buildings are captured in this dataset?
- Does the number of tanks affect pass/fail rate?
- Is there a trend for specific violations?

5.2 Methodology

Of all the columns which were included in the raw dataset, the following covariates were considered for our analysis:

- Zip code
- Borough
- Building Identification Number (BIN)
- Number of drinking water tanks
- Oversight activity year
- Latitude
- Longitude
- Violation code

To parse the data into a more malleable format, we parsed the XML using Python's xml.etree.ElementTree library and transformed it into a pandas DataFrame. Many columns had to be coerced to a tabular form with types such as dates, numerics, and categories.

5.2.1 Cleaning

Rows with NaN or NA values in almost any of the columns were dropped immediately. One of the few exceptions were the violation-related fields such as violation code and law section, which were filled with string "No Violation" for cleaner plots. Numerical columns were explicitly coerced to the Pandas numeric type, and the date column to the datetime type. Zip codes and text columns such as borough and street name were explicitly coerced to strings and were stripped of leading and trailing whitespace. Positive longitudes were set to negative values to reflect accurate NYC coordinates. These were likely erroneously recorded as positive. Columns like inspection_outcome and violations_per_million were both created for easier visualization and processing.

5.2.2 Data Enrichment

Borough-wise population estimates for 2023 were taken from NYC.gov and added to our dataframes used for analysis.[4] This was to add another dimension to the results and explore population and borough location as possible variables which affect the number of violations by boroughs and buildings. Thus, per-capita violation analysis will be discussed later in this paper.

Table 2: Borough-Wise Population

Borough	Population
Manhattan	1,597,451
Brooklyn	2,561,225
Bronx	1,356,476

4 (NYC Department of City Planning, 2024)

Queens	2,252,196
Staten Island	490,687

Building data was taken from a separate NYC.gov public dataset called "BUILDING", which has information for every building footprint in NYC.[5] Most important to our analysis of drinking water compliance is the dataset's shared BINs (which both datasets share), the borough (from the BIN), the counts of buildings between boroughs, and the building heights (from column Height Roof). With this data, we can determine how likely it is to see the water tank compliance dataset's distribution of inspection data amongst buildings.

5.2.3 Sanity Checks

Due to the data showing evidence of incomplete or even incorrect information for numerous observations, Python assertions were utilized to ensure the integrity of the data in question. The assertions are as follows:

1. Borough should not be null for any observation
2. Only 5 unique boroughs should be found
3. Population must not have any null values
4. The 5 unique boroughs in the dataset should be one of the 5 values: Manhattan, Brooklyn, Bronx, Queens, and Staten Island
5. There should be a reasonable number of rows remaining. The number of observations after cleaning should be at least 8,000 observations. This is ~50% of the total observations found when scraping from the dataset successfully. This is a conservative assertion as this is to cover catastrophic scraping or cleaning errors
6. There should be no duplicated observations
7. Longitude should be between -74.3 and -73.7
8. Latitude should be between 40.4 and 40.95

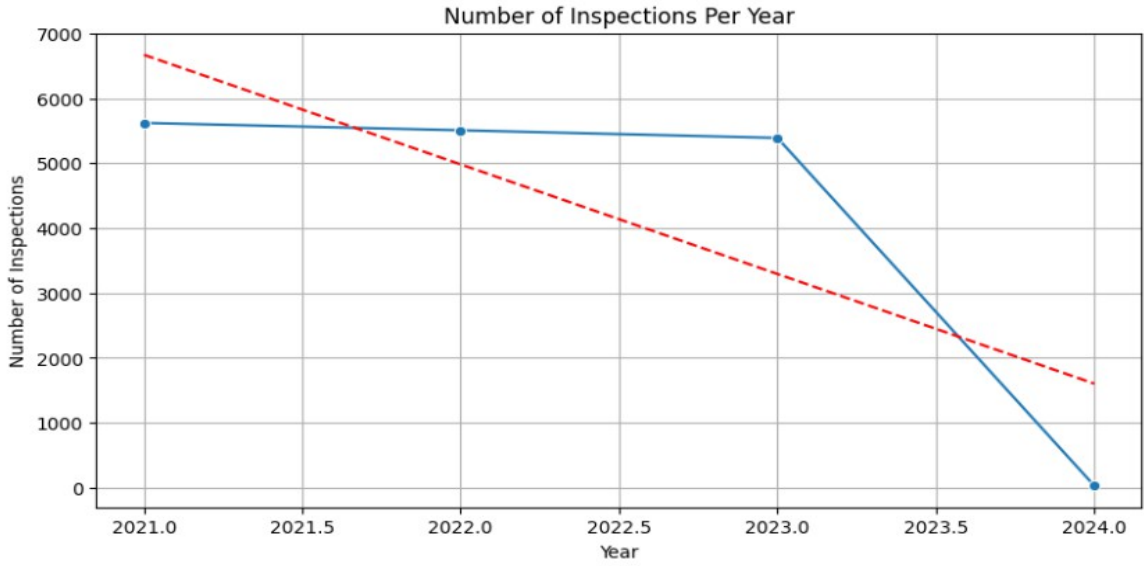
5.3 Results

Are there seasonal differences in violations?

To understand the true behavior of the inspections and violations data, we firstly investigated its behavior over time. Figure 1 below displays the number of inspections reported on a yearly basis. The drastic dip in 2024 is more due to the lack of observations for that year, as this dataset lags behind in terms of the actual recorded inspections and violations. Thus, the trend line in the figure below should not be given much credence.

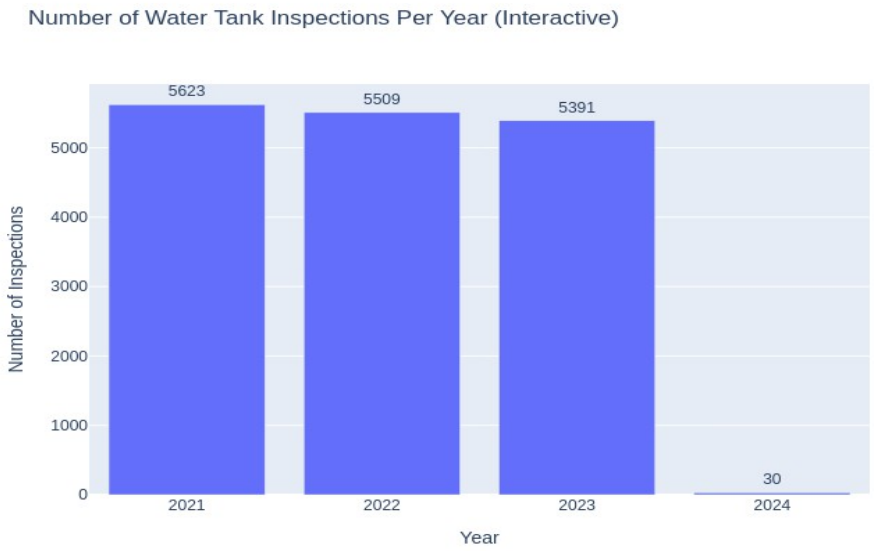
5 (Department of Health and Mental Hygiene, NYC, 2025)

Figure 1



When looking at the number of inspections by year, we observed a slight downward trend between years 2021 and 2023. Between these years, there are about 116 less inspections done on average. The number of inspections found per year in this dataset are individually displayed in Figure 2.

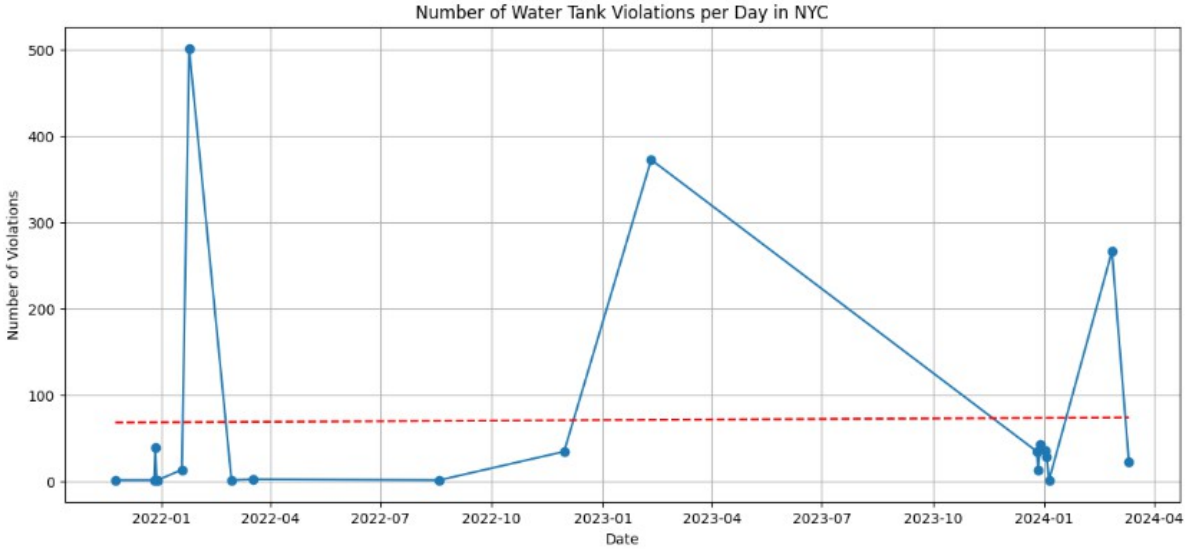
Figure 2



On a daily basis, we can see that throughout each year there are not many violations reported on most days of the year. For most days, there are near 0 violations reported in certain months of the year. It should be noted, however, that there seems to be strong seasonality even when looking at the daily data. Large outliers tend to be between January and February in every year recorded in this dataset. There is no strong trend in the average number

of daily violations. The below figure shows a time series of the number of water tank violations on a daily basis.

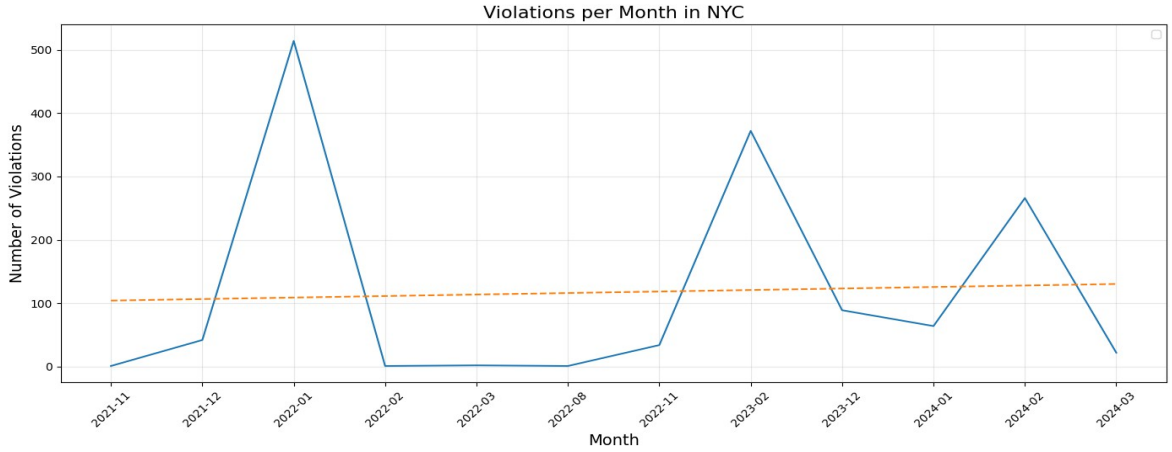
Figure 3



The seasonality of the violations reported is better illustrated when observing the number of violations on a monthly basis instead of daily. With less noise from the more frequent daily frequency of observations, instead lumping violations reported within their respective months, it becomes more clear that the number of violations increases drastically between the months of January and February. The average number of monthly violations seems to be increasing based on this metric, but it should be acknowledged that the strong seasonality of the data may make the naive average misleading.

Upon monthly STL decomposition of this time series (shown in Figure 5), a more accurate depiction of the violation data can be observed. This decomposition breaks the time series into three main parts: trend, seasonality, and residuals. The near-zero trend in the monthly water tank violations demonstrates that there is no clear upward or downward trend in the number of monthly violations for the entire dataset. In seasonality, the previous observations are corroborated by this figure, confirming that the number of monthly violations increase substantially between the months of January and February. In 2022, there were near 0 violations reported between the months of February and November.

Figure 4



While the seasonality plot in Figure 5 makes it seem like there were more violations found throughout the year of 2022 when compared with 2023, it should be noted that there is a large gap in observations of violations in 2023. This gap in the data is better demonstrated in Table 3. In this figure, it is better seen that the only months recorded in 2023 were February and December. There is likely substantial violation and inspection data in 2023 which were not captured by the NYC Health Department. This is consistent with the discussion about the integrity and accuracy of the dataset in Section 5.4.

Table 3: Month to Number of Violations Reported

Year-Month	Number of Violations Reported
2021-11	1
2021-12	42
2022-01	514
2022-02	1
2022-03	2
2022-08	1
2022-11	34
2023-02	372
2023-12	89
2024-01	64
2024-02	266
2024-03	22

Observing the residual plot of the STL decomposition in Figure 5, we could see how much of the data is still unexplained after the trend and seasonal parts of the data. The decomposition can be thought of in an additive sense like this:

$$y_2 = T_t + S_t + R_t$$

Where y_t is the series, T_t is the trend component, S_t is the seasonal component, and R_t is the Residual component. The 1e-14 above the plot show the scale of the residuals displayed,

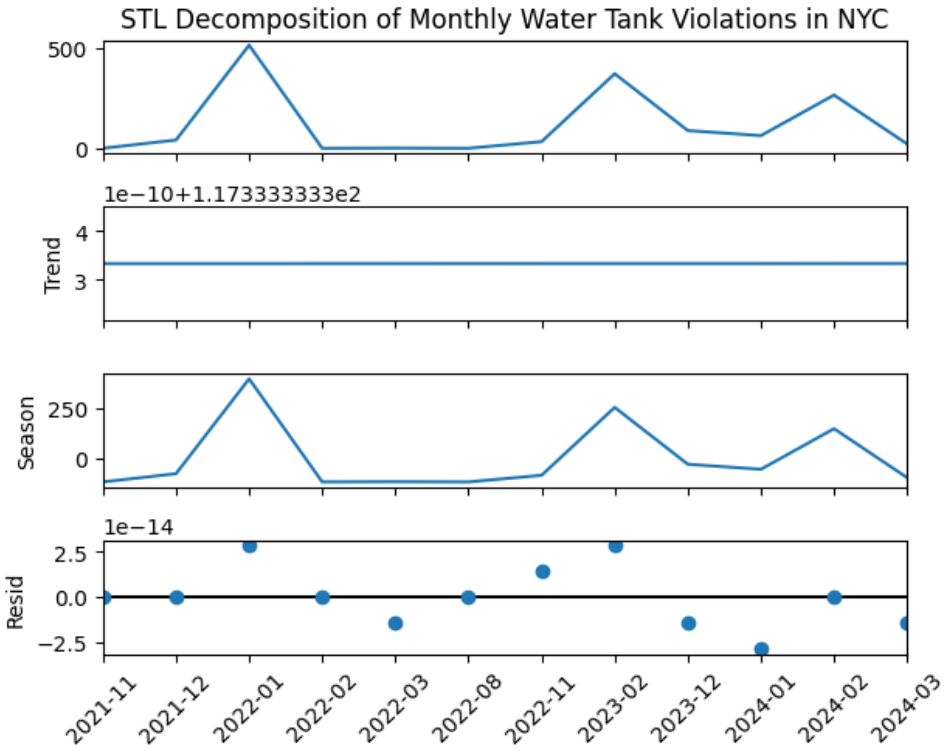
meaning that all of the residuals in the time series scale close to 0. The algorithm extracts the seasonal component by removing the trend estimate. This is done by computing the detrended series:

$$d_t = y_t - T_t$$

d_t is then grouped by seasonal position (all Februarys, for example), then LOESS is applied within each of these groups (which are their own subseries). LOESS effectively smooths out the seasonal patterns over the years to determine an estimate of seasonality by fitting a local polynomial weighted by a neighborhood on a yearly basis. This gives us the predicted S_t. Smoothing this S_t gives us the trend estimate T_t. Then, the residual component is calculated like so:

$$R_t = y_t - T_t - S_t$$

Figure 5



Does the number of tanks affect pass/fail rate?

Based on whether a violation code was recorded in an inspection found in the dataset, we indicated a “PASS” or “FAIL” in a new column we created which holds the inspection outcome called inspection_outcome. After giving every observation a categorical outcome to indicate a pass or fail in inspection, it was found that the majority of inspections across NYC result in no violation.

When calculating the percentages for passes and fails, it was found that for each year in the dataset were these percentages:

Table 4: Pass and Fail Rate from 2021-2024

Year	Pass Rate	Fail Rate
2021	90.04%	9.96%
2022	92.59%	7.41%
2023	91.84%	8.16%
2024	100.00%	0.00%

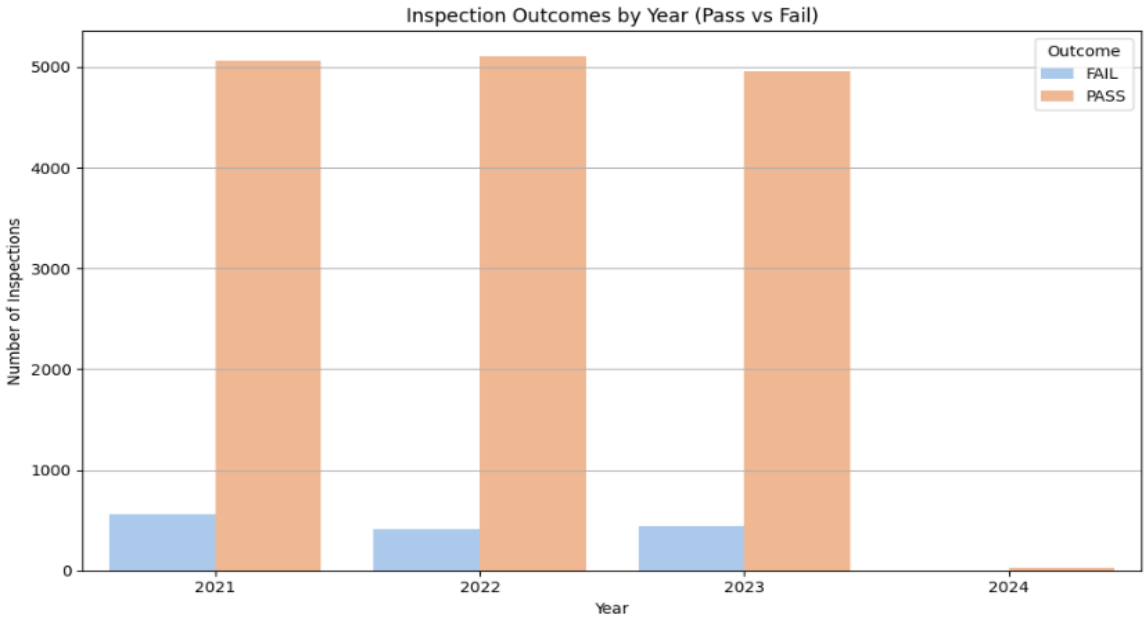
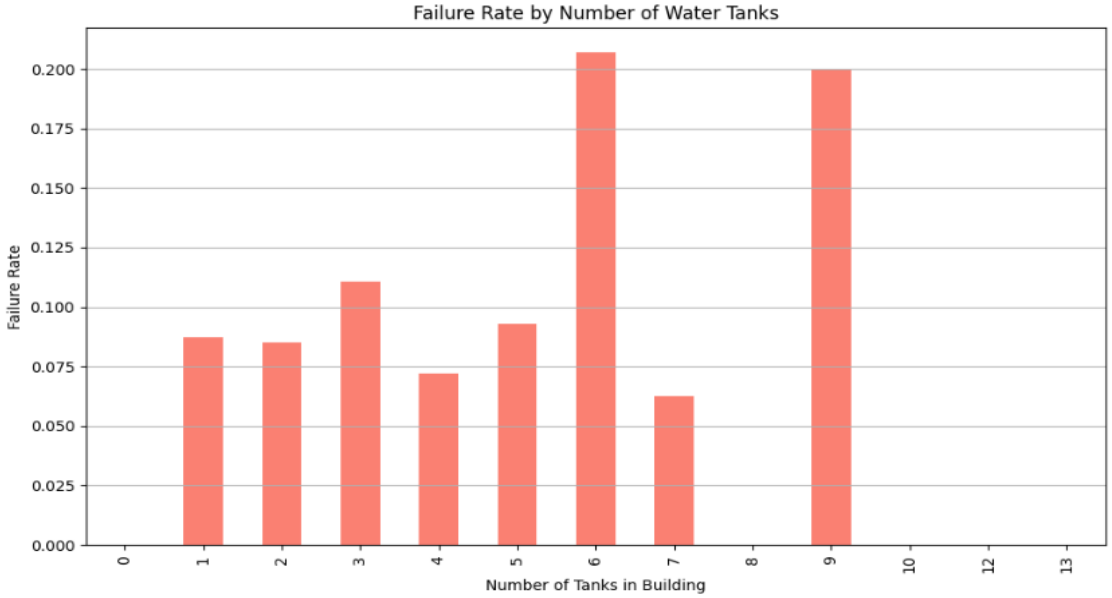


Figure 6

It should be noted that due to the lack of observations in 2024 within this dataset, that the true ratio of passes to fails are not as the above figures suggest. It is likely that it has a similar distribution as the former three years given that they all had a similar distribution.

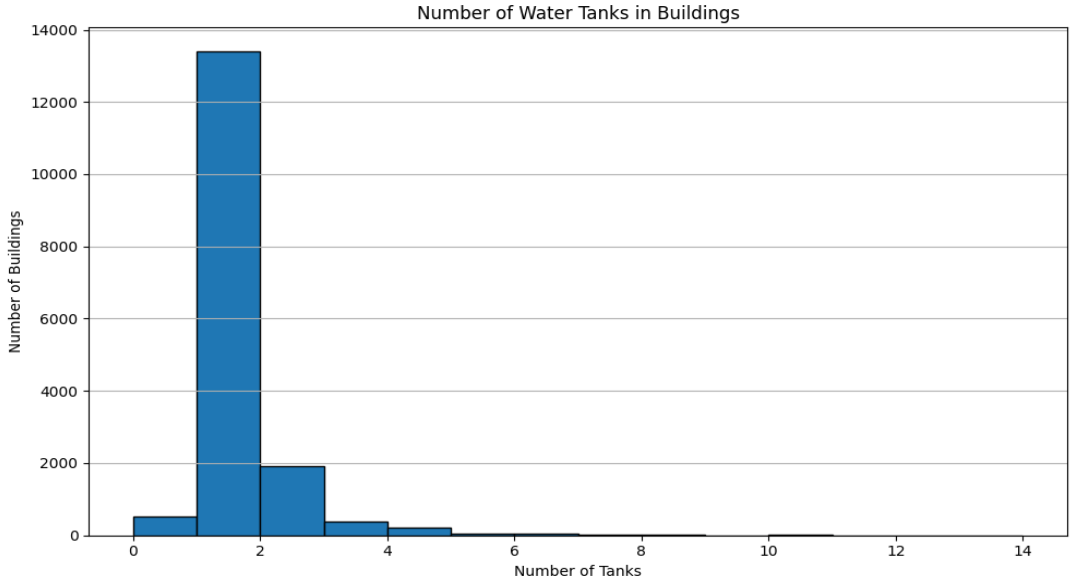
When grouping observations by the number of drinking water tanks and the inspection outcomes, we found that when buildings have 6 or 8 tanks, there is a substantially higher failure rate. While this makes it seem as though there is a correlation between a higher number of drinking water tanks and the failure rate, Figure 11 shows that the distribution of buildings with more than three violations across the four-year period have a heavy skew towards having only one water tank.

Figure 7



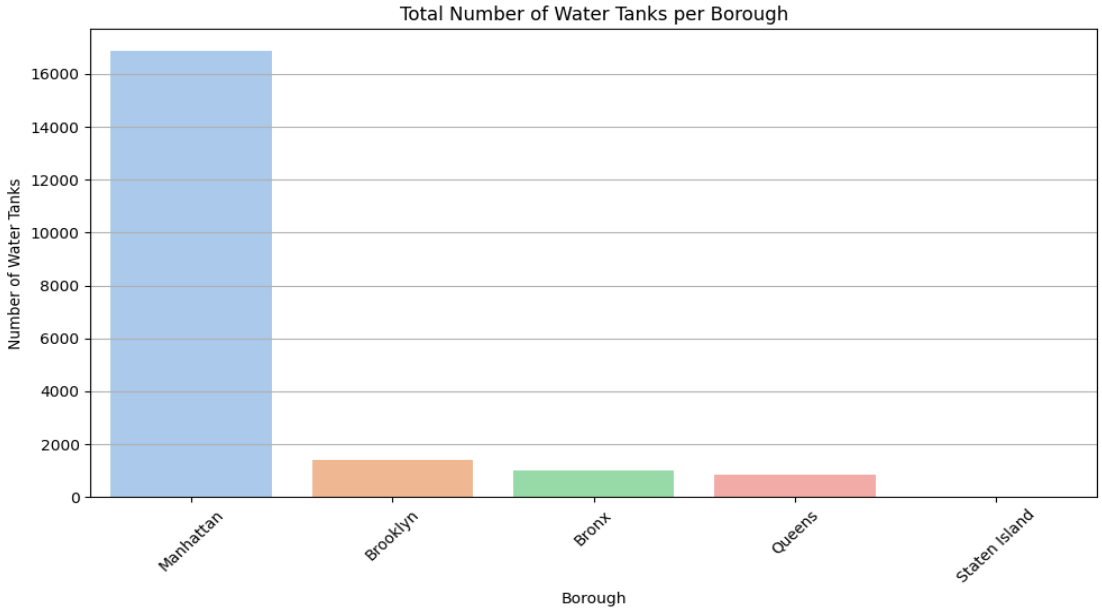
The small failure rates for 7 and 8 tanks should be noted, but there is also a small number of observations of buildings with more than 1-2 drinking water tanks. The figure below demonstrates the distribution for the number of tanks for given buildings in this dataset:

Figure 8



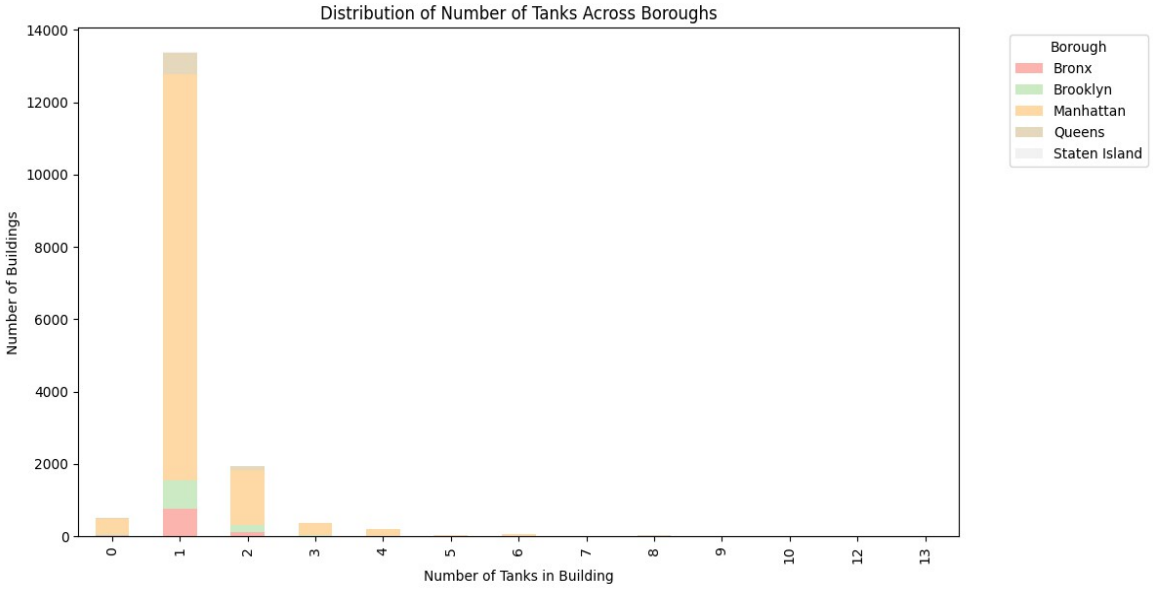
It should not be ignored that in terms of the aggregated number of drinking water tanks by borough, Manhattan leads by far. Manhattan has over 16,000 drinking water tanks in total, while the other boroughs have less than 2,000.

Figure 9



When looking at the distribution of the number of drinking water tanks per building between boroughs, we observed that an overwhelming majority of the single-tank buildings were in Manhattan.

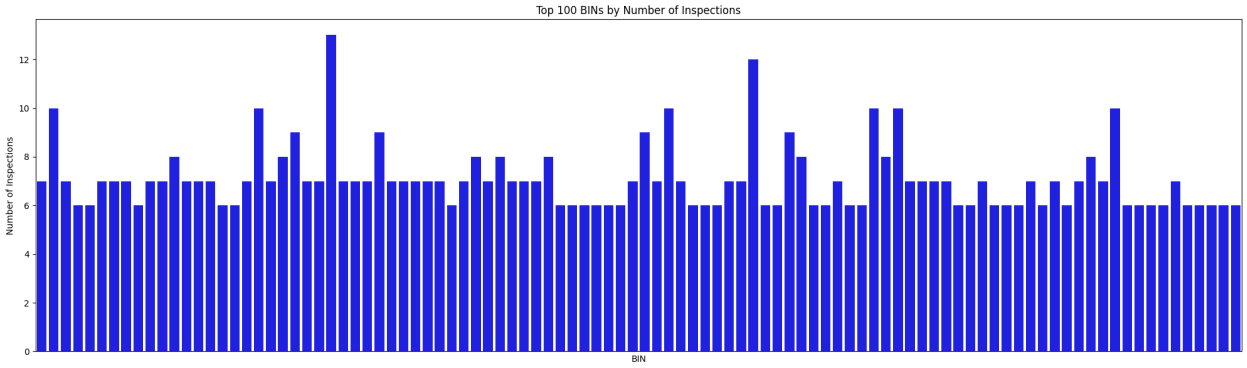
Figure 10



Are there specific buildings with more inspections or violations?

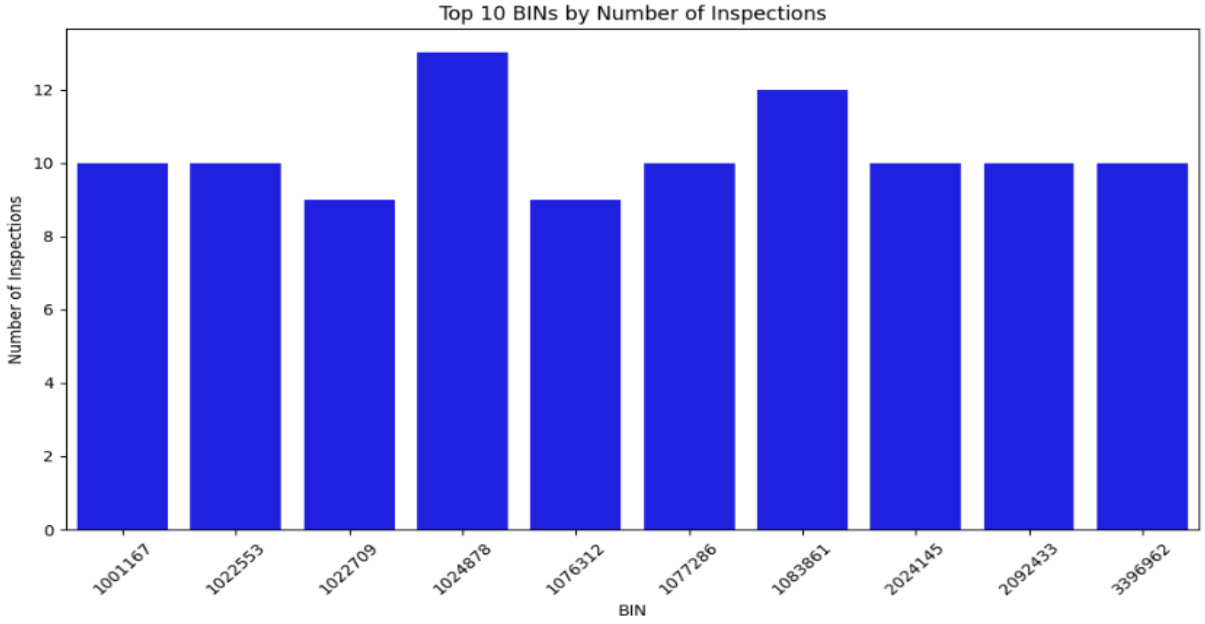
Turning our focus to the relationship between specific buildings and repeated inspections, we found that most buildings in this dataset had around 3 inspections. Given that the majority of observations fall within the 2021-2023 range, this makes sense given that these inspections are done annually. However, in the dataset we found 259 buildings with over 3 inspections.

Figure 11



When looking at the top buildings by number of inspections, some have as many as ten inspections over the four-year period.

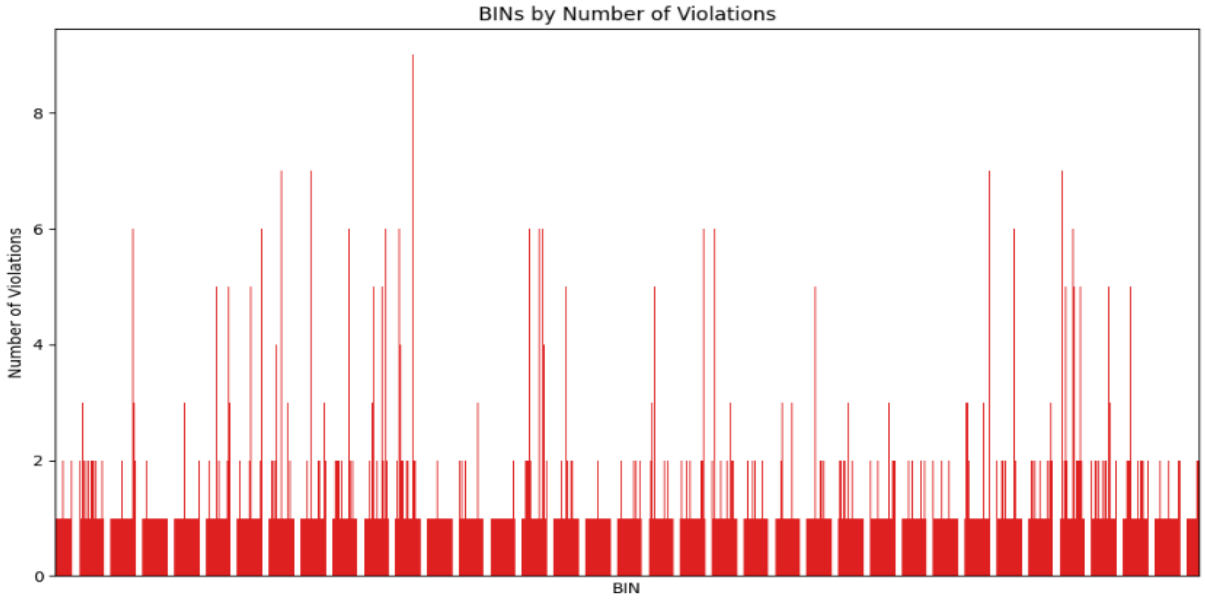
Figure 12



There is more variance in the number of violations by building, as is seen in the figure below. While the majority have only one violation over this four-year period, there are a substantial number of buildings with two violations. Less so is the number of buildings with

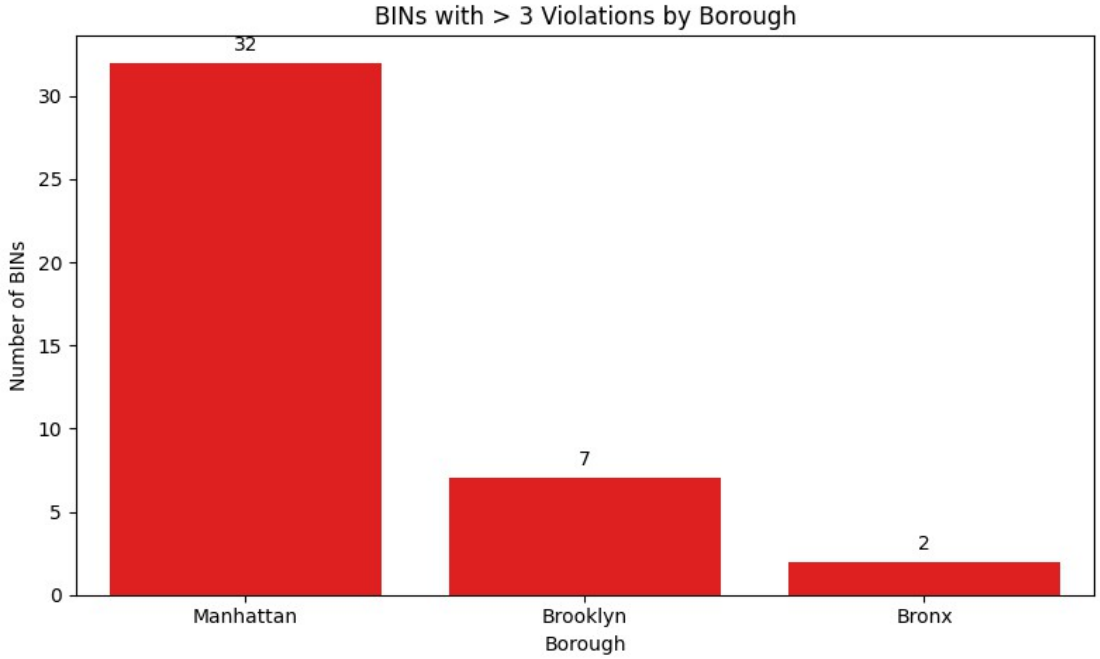
greater than three violations, but this is not to be ignored.

Figure 13



After separating these buildings with more than three violations by their respective borough, it was found that the overwhelming majority of these were buildings in Manhattan. Surprisingly, not a single building in Queens or Staten Island was found to have more than three violations.

Figure 14



How many buildings are captured in this dataset?

After aggregating the number of unique building identification numbers (BINs) from both the drinking water tank compliance dataset as well as NYC's official building dataset as well, we observed the following distribution of coverage:

Table 5: Building Coverage Comparison

Borough	Unique Buildings in Water Tank Dataset	Total Unique Buildings	Proportion Reported
Staten Island	6	142301	0.00%
Queens	232	461033	0.05%
Brooklyn	344	330198	0.10%
Bronx	275	104263	0.26%
Manhattan	4483	45200	9.92%

While Manhattan had the best coverage by far, this still shows that a small minority of buildings even have drinking water compliance documentation. The conclusions from this will be further explored in section 5.4 of this paper.

An important aside analysis of the buildings dataset gives us the number and proportion of buildings with greater than or equal to six stories. This will be important for our later discussion in section 5.4, since buildings with more than 6 floors which require drinking water need a tank. There are proportionately more buildings with greater than six floors in Manhattan than there are for any other borough.

Table 6: Proportion of Buildings with > 6 Floors

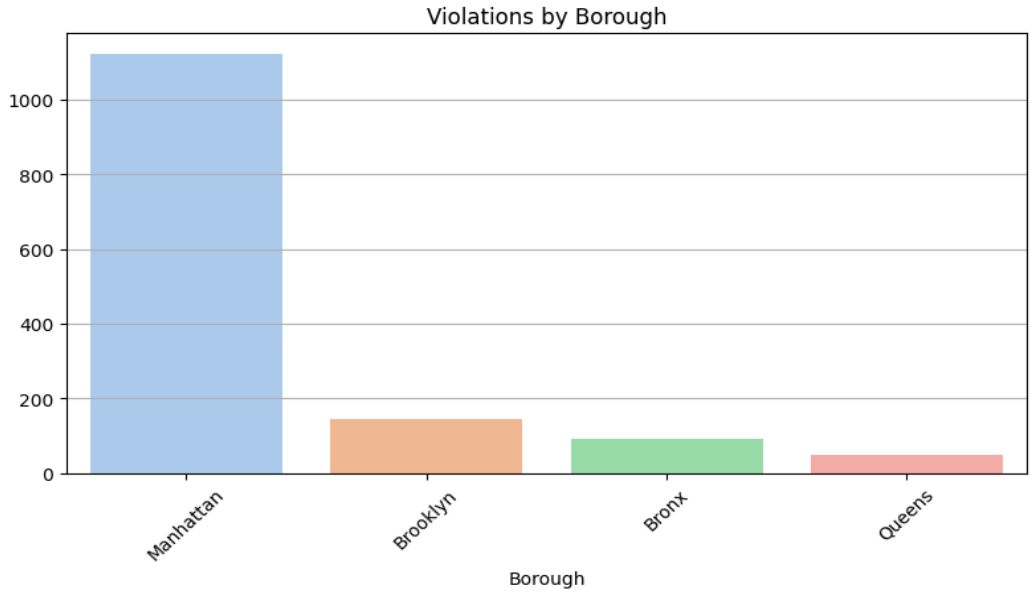
borough	Buildings > 6 Floors	Total Buildings	Proportion > 6 Floors to Total Buildings
Bronx	5517	104263	5.29%
Brooklyn	7095	330198	2.15%
Manhattan	19945	45200	44.13%
Queens	3898	461033	0.85%
Staten Island	265	142301	0.19%
Total	36720	1082995	3.39%

Which boroughs have more violations?

In terms of violations found overall, a similar trend is observed. While Brooklyn, Queens, and the Bronx all had more counts of violations, Manhattan had substantially more. Over a thousand were found over this four-year period. No violations were found in Staten Island, and thus Staten Island is not included in many of the borough-wise figures.

When grouping observations by whether the building had greater than 3 violations and grouping by borough, it was found that the majority of buildings in every borough have only one tank. Manhattan has the most variance in number of tanks found, where the distribution of buildings with two, three, and six tanks were similar. This is much like the previous figure, except it specifically shows the distribution for buildings with greater than three violations. Amongst the group of buildings with greater than 3 violations, there were none besides ones with one, two, three, and six tanks.

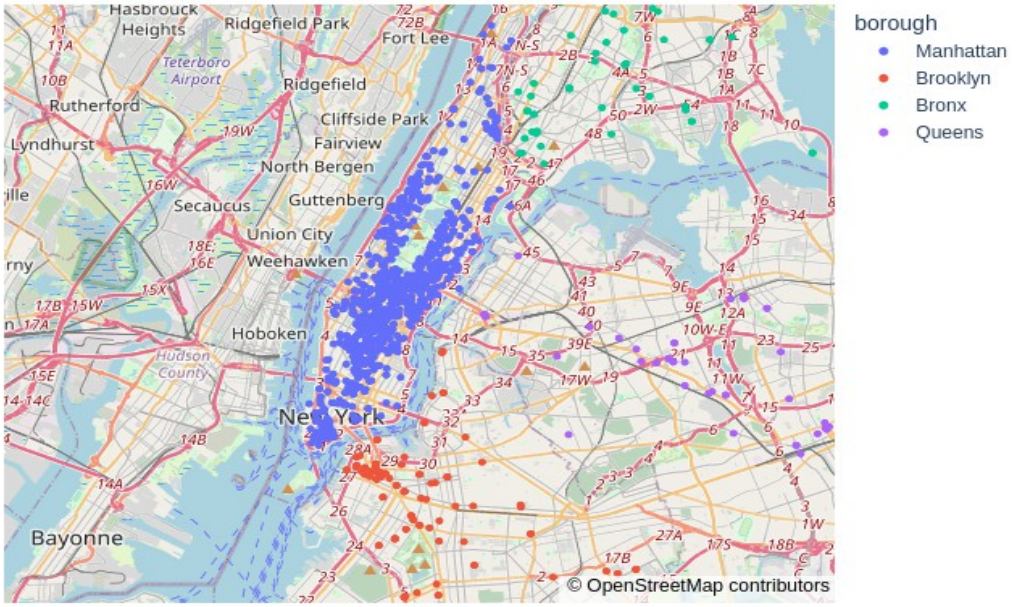
Figure 15



Visualizing the number of violations onto a map, it becomes abundantly clear that these violations are found throughout Manhattan. Violations are not localized to a specific area; It is a widespread issue observed throughout the borough.

Figure 16

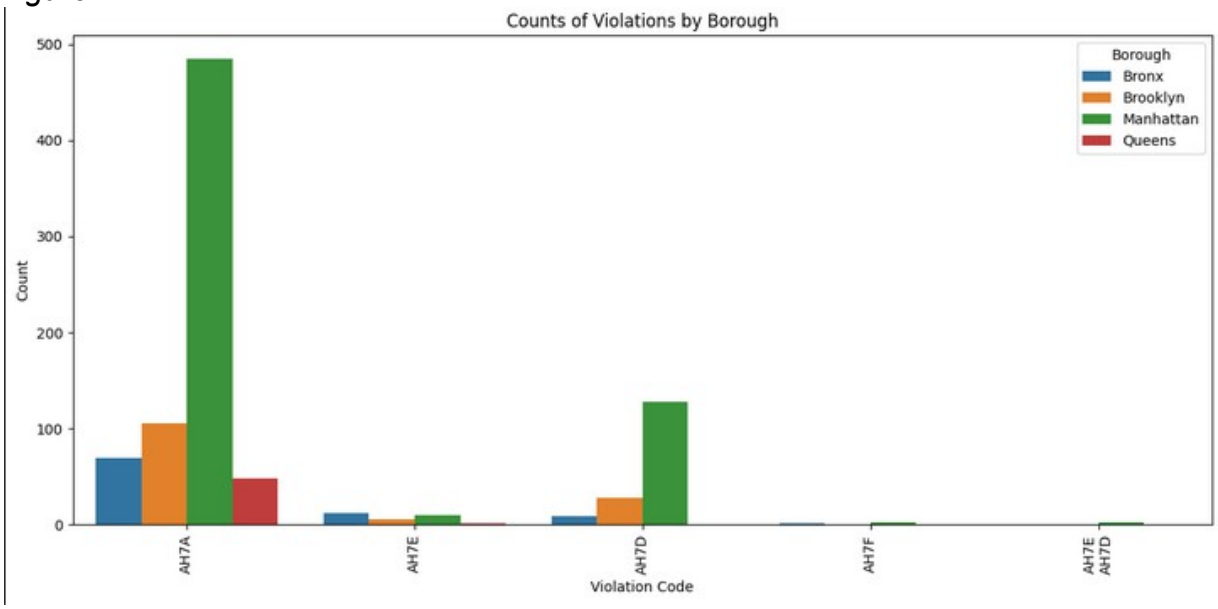
NYC Water Tank Violations by Location



Is there a trend for specific violations?

Observing the specific violations in question, however, paints a clearer picture of the nature of these violations. When grouping these violations by their specific code, it was found that the overwhelming majority of violations were due to a failure to even submit a report of the previous year's inspection results.

Figure 17



In addition, most of this specific violation type are from Manhattan. There were 969 violations of this type within the four-year period of this dataset, with the majority within only the first three years of this dataset.

Table 7: Distribution of Violation Codes

borough	violation_code	violation_text	count
Bronx	AH7A	Failure to submit report of previous year's inspection results by January 15 of the following year	69
Bronx	AH7E	Failure to provide results of drinking water tank inspection within five business days after receipt of Department's request	12
Bronx	AH7D	Failure to inspect drinking water tank, including testing water for bacteriological content	9
Bronx	AH7F	Failure to correct unsanitary condition of drinking water tank	2
Brooklyn	AH7A	Failure to submit report of previous year's inspection results by January 15 of the following year	106

Brooklyn	AH7D	Failure to inspect drinking water tank, including testing water for bacteriological content	28
Brooklyn	AH7E	Failure to provide results of drinking water tank inspection within five business days after receipt of Department's request	12
Manhattan	AH7A	Failure to submit report of previous year's inspection results by January 15 of the following year	969
Manhattan	AH7D	Failure to inspect drinking water tank, including testing water for bacteriological content	128
Manhattan	AH7E	Failure to provide results of drinking water tank inspection within five business days after receipt of Department's request	20
Manhattan	AH7F	Failure to correct unsanitary condition of drinking water tank	2
Queens	AH7A	Failure to submit report of previous year's inspection results by January 15 of the following year	48
Queens	AH7E	Failure to provide results of drinking water tank inspection within five business days after receipt of Department's request	1

5.4 Discussion

Analyzing the inspections and violations over time, there was a consistent trend in the number of inspections and violations over the four-year period. An initial glance at the time series may give the impression of a downward trend, but upon STL decomposition, there was no evidence of either a downward or upward trend. From both the time series itself and its STL decomposition, it is clear that there is strong seasonality in the data. It seems like violations tend to be found around January and February. This makes sense, given that the deadline for drinking water tank inspections is January 15th of the following year. We know that most violations reported in this dataset are due to the negligence of the owners to submit a report, so the most likely cause of the seasonality of violations over time is due to the yearly wave of unsubmitted reports. The failure rate over the time period covered in this dataset does not show evidence of an upward or downward trend, either.

Most inspections lead to no violation at all, according to our dataset, as around ~90% of inspections pass with no violation code reported over this four-year period. There are external reports which cause suspicion of the validity of this ratio, however. As we know from the NYC DOHMH, drinking water tank owners are required to bring a qualified inspector to conduct these inspections as well as submit the results to the Health Department.[6] This is conceptually an effective way to minimize situations where owners dishonestly report their own results, but in practice, it is ineffective. In 2018, freelance NYC investigative reporter Frank G. Runyeon conducted an investigation of the condition, inspection, and regulation of drinking water tanks in NYC. When scrutinizing the process in which these owners conduct their inspections and violation reporting, it was found that there are owners that work around these regulations for bacteriological test results: "...in almost every case the [bacteriological] tests are conducted only after the tanks have been disinfected…". Reports of E. coli and coliform bacteria in water samples taken by the NY Times were found in 2014, yet NYC health officials disputed them. Layers of bacterial slime and dirt were found at the bottom of several drinking water tanks that were investigated.[7] From this evidence, it is evident that the bacteriological inspections of these tanks are many times not as reputable as one might expect. Many tanks may be passing only due to the timing of the inspection rather than due to the reliably good condition of the tank throughout the year. The tanks may be in poor condition for the vast majority of the year and cleaned right before inspection in order to pass.

It's not just bacteria that were found in some of these tanks either. There are numerous anecdotes of tank cleaning personnel finding dead pigeons and squirrels sitting at the bottom of tanks, sometimes clogging the pipes. They suggest that this is not an entirely uncommon occurrence, one of them stating that 1 in 50 tanks is a dead pigeon found in a water tank. They have also found evidence of animals getting into the drinking water tanks.[8] Times reporters Ray Rivera, Frank G. Runyeon, and Russ Buettner conducted an investigation in 2013 which found "muddy sediment and e-coli bacteria accumulated at the bottom of some water tanks" in Manhattan, Queens, and Brooklyn. They suggest that these water tanks may possibly have been neglected for cleaning or inspection for years.[9] This is consistent with our analysis, which found that most violations overall are due to a lack of an inspection report being submitted at all. With this information in mind, while around 90% of inspections in our dataset led to no violations given, statistical and in-person evidence from NYC reporters suggests that this does not guarantee that the water tank(s) inspected truly meet health standards like they are supposed to.

Another point of discussion is the validity and integrity of the water tank count data. Runyeon states: "City health and buildings officials cannot even say for sure how many water tanks are in use. Estimates range from 12,000 to 17,000, based on the inventory of buildings that stand seven stories or taller".[10] Runyeon has continued to report this lack of more substantial tracking of water tanks as recent as 2019.[11] This evidence shows that even the

6 (DOHMH, 2025)
7 (Runyeon, 2018)
8 (Runyeon, 2018)
9 (Goldstein, 2025)
10 (Rivera, Runyeon, and Buettner 2014)
11 (Runyeon, 2019)

officials do not know how many water tanks there are, as they can only give a vague estimate rather than a certain count. This also implies that there is likely no robust water tank tracking measure in effect. As Runyeon concludes, this would mean that there is likely a number of tanks that remain to be tracked at all.[12] The lax nature of the tracking and reporting of water tanks gives reason to believe that this number is large though unknown. While the NYC Health Department's dataset shows the majority of buildings have only one tank, the true distribution could be completely different. The underlying truth that may not be captured in our dataset could be that most non-compliant tanks could be in buildings where there are many tanks, or perhaps when there are fewer tanks. Given how unreliable the reporting on the number of tanks for buildings is, there is reasonable doubt as to what exactly the effect the number of drinking water tanks in a building has on the number of violations found. From the data analyzed here, however, it seems as though a higher number of drinking water tanks does not necessarily affect the number of violations found. Most buildings with greater than three violations were overwhelmingly buildings with only one tank, whereas the buildings with two, three, and six tanks had a similar, sparse distribution.

The water tank compliance dataset's coverage is also put into question when looking further into the buildings dataset. After aggregating the observations in both the water tank compliance and buildings datasets, we observed that a small proportion of buildings were covered in the water tank dataset. For boroughs other than Manhattan, the number of buildings which were in the water tank compliance dataset was < 1%. For Manhattan, 9.92% of its buildings were reported in the water tank compliance dataset. This may seem reasonable at first before considering that there are only 4,483 Manhattans buildings in the water tank compliance dataset, whereas there are an estimated 19,945 Manhattan buildings which are greater than six stories tall. If we assume the water tank compliance dataset is true, then we would have to assume that only about 22.48% of Manhattan buildings at least six stories have at least one drinking water tank (since they have submitted a filing which was reflected in the water tank compliance dataset). This is a highly improbable proportion, as Manhattan is a densely populated area with many high-rise commercial and residential buildings. This further supports the argument that the water tank compliance data collection and reporting have considerable gaps which cannot be ignored.

There are some considerable limitations to this analysis as well. It is still unclear whether the population is truly a substantial predictor for the number of violations given. While Manhattan does have the most violations per million residents, it also naturally has the most violations as it also has the most drinking water tanks. How much these variables confound is unclear from this analysis. Moreover, while our data comes directly from a public NYC Health dataset, the department officials' behavior and the evidence of poor data integrity are present. Drinking water tank counts per building and inspection data are inherently fallible due to how vague the inspection guidelines are and how weakly these inspections are enforced. NYC Health officials and the Health Department's statistics show questionable reputability because of the previously discussed evidence of passing tanks which have bacterial contamination and dead animal remains inside them. This is made worse by health officials' repeated efforts to inflate inspection statistics or downplay poor tank maintenance. In 2017, only 34% of buildings provided proof of tank inspections, while health officials testified a higher rate of at least 60%

12 (Rivera, Runyeon, and Buettner 2014)

or even 73%, but they have not given substantial evidence or explained their reasoning to back up their claims.[13] The questionable integrity and accuracy of the data also put into question just how accurate conclusions based on such data would be.

Runyeon's testimony is not to be taken with absolute certainty, however. Runyeon had performed investigations on only 12 buildings which were from Manhattan, Queens, and Brooklyn. While most had some form of bacteriological contamination, how much this reflects on the population of water tanks is still uncertain. Runyeon's conclusion that there may be a substantial number of water tanks which are untracked and likely unkempt is largely inference based on the city's lack of a certain count of water tanks and the lax nature of the NYC Health Department's enforcement of their water tank compliance policies. He supports his conclusions by citing a few plumbing and tank cleaning companies (3 confirmed in "Lax oversight, dubious testing in water tanks pose health risks", although in the other articles cited here it is uncertain how many companies he approached), but this is anecdotal evidence which may or may not reflect the underlying truth.[14] While almost all the tanks inspected by Runyeon were found to be in poor condition and in violation of health guidelines, this could be by pure chance, given his observations are only from a 12-sample set. Further investigation needs to be done on a larger scale to get a more realistic estimate of how many buildings may be in violation during the year before inspections.

There is at least some reasonable suspicion from this external data, however, that it is likely that the number of tanks which do not actually meet health guidelines is higher, and not lower than the official dataset suggests. It is likely that the failure rate is much greater than 10%, and the nature of the passes and violations are likely different than they may seem. Given the reports of passing tanks which were in poor condition, 90% of buildings which passed could and likely would fail if they were instead inspected at a random time by an individual not picked out by the owner themselves. Besides the likely many false inspection passes, the fact that most violations were due to a failure to submit a report point to the likely possibility that there are more tanks which likely do not meet the health guidelines.

5.5 Conclusion

Chepuri and I have demonstrated with this analysis that there is substantial reason for suspicion towards the health compliance and quality of water in NYC's drinking water tanks. Perhaps the water reservoirs truly are free of contaminants and pass the bacteriological, chemical, and mineral tests for human consumption, but the data and external evidence on the drinking water tanks raise considerable concerns. Tanks that have passed inspections have been found to have bacteriological contaminants and animal carcass remains, most of the violations reported are from a lack of inspection reporting, and even the number of inspection passes could be substantially inflated given the evidence of water tank owners setting up their tests to have a positive bias. It is entirely possible that there is a widespread health compliance issue with these drinking water tanks, especially in the Manhattan area, which remain to be properly documented. There are a substantial number of buildings in the Manhattan borough which have more than three violations over the four-year period covered in the dataset, with

13 (Runyeon, 2018)
14 (Runyeon, 2018)

some even reaching six or eight violations. Even worse, are the many ways health officials have made statements which suggest that there is more certainty in the condition and documentation of these water tanks than there is in reality and unsubstantiated conflations of statistics which paint the department in a more positive light.

Given evidence of such a systematic and likely widespread issue within at least the Manhattan borough, further analysis is needed to ensure the compliance of water tanks in NYC. More specifically, there needs to be a deeper analysis of the effect of population on the violation rate, as well as better data collection done ideally by parties independent from the NYC Health Department. The dataset analyzed here understandably does not hold this data, but it may be helpful for future investigation to know who owns these buildings. Perhaps we may see the same select people who are responsible for the buildings with the most offenses over the four-year period. At the very least, it would be helpful to know the owners of the buildings with the most violations.

6 Appendices

6.1 Appendix A

```
import requests
import xml.etree.ElementTree as ET
import pandas as pd

# Step 1: Download the XML data
url = "https://data.cityofnewyork.us/api/views/rytv-g5ui/rows.xml?accessType=DOWNLOAD"
response = requests.get(url)

if response.status_code == 200:
    print("XML file downloaded successfully!")
else:
    raise Exception(f"Failed to download XML. Status code: {response.status_code}")

# Step 2: Parse XML content
root = ET.fromstring(response.content)

# Step 3: Extract each <row> into a dictionary
records = []

for row in root.findall(".//row"):
    record = {}
    for field in row:
        record[field.tag] = field.text
    records.append(record)

# Step 4: Create DataFrame
df = pd.DataFrame(records)

# Step 5: Save the parsed DataFrame
df.to_csv('nyc_water_tank_compliance_final_from_xml.csv', index=False)

print(f"Parsed {df.shape[0]} rows and {df.shape[1]} columns successfully!")
df.head()


len(df.columns)

df.columns
```

```
# Count rows with at least one NaN
num_rows_with_na = df.isna().any(axis=1).sum()
print(f"Number of rows with at least one NA: {num_rows_with_na}")

#cleaning

# 1. Drop the extra empty first row
df = df.dropna(how='all')  # Drop rows where all values are NaN

# 2. Drop the unnecessary 'row' column if it exists
if 'row' in df.columns:
    df = df.drop(columns=['row'])

# 3. Reset index after cleaning
df = df.reset_index(drop=True)

# 4. Remove duplicate rows
df = df[df.duplicated() == False]

# Save the cleaned raw data
df.to_csv('nyc_water_tank_compliance_cleaned_from_xml.csv', index=False)

# Quick check
print(df.shape)
df.head()


df.info()
df.describe(include='all')


# See missing value summary
print(df.isnull().sum())

# For critical missing fields: drop rows if needed
df = df.dropna(subset=['borough', 'street_name'])

# For violation columns (violation_code, law_section, summons_number, violation_text),
we can keep missing as NaN (means no violation).


# 1. Drop rows missing critical fields
df = df.dropna(subset=[
    'bin', 'house', 'street_name', 'borough',
    'status', 'number_of_dwt', 'activity_type',
```

```
    'activity_year', 'bbl', 'latitude', 'community_board',
    'council_district', 'census_tract', 'nta_code'
])
df = df[~df.eq('ERROR: #N/A').any(axis=1)]


# 2. Convert numerical columns
numerical_fields = ['number_of_dwt', 'activity_year', 'compliance_year', 'longitude',
'latitude']
for field in numerical_fields:
  df[field] = pd.to_numeric(df[field], errors='coerce')

# 3. Convert date column
df['date_of_occurrence'] = pd.to_datetime(df['date_of_occurrence'], errors='coerce')

# 4. Keep zip_code as string
df['zip_code'] = df['zip_code'].astype(str)

# 5. Clean text columns (optional but good)
text_columns = ['borough', 'street_name', 'status', 'activity_type']
for col in text_columns:
    df[col] = df[col].str.strip().str.title()

# 6. Save clean dataset
df.to_csv('nyc_water_tank_compliance_ready_for_analysis.csv', index=False)

print(f"Final shape after cleaning: {df.shape}")


df

# Fill missing violation-related fields for cleaner plots
violation_cols = ['violation_code', 'law_section', 'violation_text', 'summons_number']
df[violation_cols] = df[violation_cols].fillna('No Violation')

# Check latitude/longitude ranges
print(df['latitude'].min(), df['latitude'].max())
print(df['longitude'].min(), df['longitude'].max())


#correcting longtitudes sign
# Flip positive longitude values to negative
df.loc[df['longitude'] > 0, 'longitude'] = -df['longitude']


df.to_csv('sign correction.csv', index=False)
```

```
print(df.isnull().sum())

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Load final dataset
df = pd.read_csv('sign correction.csv')

# Fill violation fields with 'No Violation' if NaN
violation_cols = ['violation_code', 'law_section', 'violation_text', 'summons_number']
df[violation_cols] = df[violation_cols].fillna('No Violation')

# Define inspection outcome
df['inspection_outcome'] = df['violation_code'].apply(lambda x: 'PASS' if x == 'No
Violation' else 'FAIL')


import numpy as np
# Inspections per year
inspections_by_year = df['activity_year'].value_counts().sort_index()

years = inspections_by_year.index.to_numpy()
inspections = inspections_by_year.values
trend_line = np.poly1d(np.polyfit(years, inspections, 1))

# Line plot
plt.figure(figsize=(10,5))
sns.lineplot(x=inspections_by_year.index, y=inspections_by_year.values, marker='o')
plt.plot(years, trend_line(years), color='red', linestyle='--', label='Trend')
plt.title('Number of Inspections Per Year')
plt.xlabel('Year')
plt.ylabel('Number of Inspections')
plt.grid(True)
plt.show()

# Analysis
print(inspections_by_year)
filtered = inspections_by_year.loc[2021:2023]
years = filtered.index.to_numpy()
inspections = filtered.values
slope = (inspections[-1] - inspections[0]) / (years[-1] - years[0])
print(f'slope (2021–2023): {slope}')
```

```
# Make sure date_of_occurrence is a datetime type
df['date_of_occurrence'] = pd.to_datetime(df['date_of_occurrence'], errors='coerce')

# Filter only rows with valid dates
violations_with_dates = df.dropna(subset=['date_of_occurrence'])

pass_count = (df['inspection_outcome'] == 'PASS').sum()
fail_count = (df['inspection_outcome'] == 'FAIL').sum()
print(f'Number of passes: {pass_count}')
print(f'Number of violatons: {fail_count}')
print(f"Number of violations with dates: {len(violations_with_dates)}")
violations_with_dates.head()

plt.figure(figsize=(12,6))

# Scatter plot: each violation occurrence as a point
plt.scatter(violations_with_dates['date_of_occurrence'],
[1]*len(violations_with_dates),
        alpha=0.5,
        marker='|',
        s=400)

plt.title('Timeline of Water Tank Violations in NYC')
plt.xlabel('Date of Occurrence')
plt.yticks([])
plt.grid(True)
plt.show()


import numpy as np
from scipy.signal import find_peaks
violations_per_day = violations_with_dates.groupby('date_of_occurrence').size()
# For trend line
x = violations_per_day.index.map(lambda date: date.toordinal())
y = violations_per_day.values
trend_line = np.poly1d(np.polyfit(x, y, deg=1))

# Plot violations per day.
plt.figure(figsize=(14,6))
plt.plot(violations_per_day.index, violations_per_day.values, marker='o', linestyle='-')
plt.plot(violations_per_day.index, trend_line(x), color='red', linestyle='--', label='Trend')
plt.title('Number of Water Tank Violations per Day in NYC')
plt.xlabel('Date')
plt.ylabel('Number of Violations')
```

```
plt.grid(True)
plt.show()

# Find the peaks to understand at what dates were the highest number of violations.
peaks, _ = find_peaks(violations_per_day.values)
peak_dates = violations_per_day.index[peaks]
peak_counts = violations_per_day.values[peaks]
for date, count in zip(peak_dates, peak_counts):
    print(f"Peak on {date.date()} with {count} violations")

coef = np.polyfit(x, y, 1)
slope = coef[0]
intercept = coef[1]

trend_min = trend_line(x).min()
trend_max = trend_line(x).max()

print(f'trend_min: {trend_min}')
print(f'trend_max: {trend_max}')
print(f'intercept: {intercept}')
print(f'slope: {slope}')

# Create a "year_month" column
# Use .copy() to avoid SettingWithCopyWarning
violations_with_dates = violations_with_dates.copy()
violations_with_dates['year_month'] =
violations_with_dates['date_of_occurrence'].dt.to_period('M')

# Count violations per month
violations_by_month = violations_with_dates['year_month'].value_counts().sort_index()

# Convert PeriodIndex to string (optional for better x-axis)
violations_by_month.index = violations_by_month.index.astype(str)

# Preview
print(violations_by_month)

violations_with_dates['year_month'].unique()

x = np.arange(len(violations_by_month))
y = violations_by_month.values

coef = np.polyfit(x, y, 1)
slope = coef[0]
intercept = coef[1]
```

```
trend = np.polyval(coef, x)
trend_min = trend.min()
trend_max = trend.max()

plt.figure(figsize=(14, 6))
plt.plot(violations_by_month.index, y)
plt.plot(violations_by_month.index, trend, linestyle="--")

plt.title("Violations per Month in NYC", fontsize=16)
plt.xlabel("Month", fontsize=14)
plt.ylabel("Number of Violations", fontsize=14)

plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()

print(f'trend_min: {trend_min}')
print(f'trend_max: {trend_max}')
print(f'intercept: {intercept}')
print(f'slope: {slope}')

from statsmodels.tsa.seasonal import STL

series = violations_by_month
series.name = "STL Decomposition of Monthly Water Tank Violations in NYC"

stl = STL(series, period=12).fit()

trend_component = stl.trend
seasonal_component = stl.seasonal
residual_component = stl.resid

fig = stl.plot()

plt.xticks(rotation=45)
plt.show()

x = np.arange(len(trend_component))
y = trend_component.values
slope = np.polyfit(x, y, 1)[0]
print("Slope of trend:", slope)

#violations over time
plt.figure(figsize=(12,6))
```

```
# Create a barplot
sns.barplot(x=violations_by_month.index, y=violations_by_month.values,
color='royalblue')

plt.title('Monthly Violations of Water Tank Inspections in NYC')
plt.xlabel('Month')
plt.ylabel('Number of Violations')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y')
plt.tight_layout()
plt.show()


import plotly.express as px

# We'll use violations_with_dates DataFrame (violations only)

# Make sure date_of_occurrence is datetime
violations_with_dates['date_of_occurrence'] =
pd.to_datetime(violations_with_dates['date_of_occurrence'], errors='coerce')


# Creation of dummy column outside px.scatter() to avoid conflict errors.
violations_with_dates['y'] = 1
fig = px.scatter(
    violations_with_dates,
    x='date_of_occurrence',
    y='y',  # place all points at y=1
    hover_data={
        'borough': True,
        'street_name': True,
        'zip_code': True,
        'date_of_occurrence': True,
        'y': False  # hide the y dummy column
    },
    title="Timeline of Water Tank Violations in NYC (Interactive)",
    labels={"date_of_occurrence": "Date", "y": ""}
)

fig.update_traces(marker=dict(size=8, color='blue'))
fig.update_layout(
    showlegend=False,
    yaxis=dict(visible=False),
    xaxis_title="Date of Occurrence",
    margin=dict(l=20, r=20, t=40, b=20)
```

```
)

fig.show()


#inspections over time
import seaborn as sns
import matplotlib.pyplot as plt

# Count number of inspections per year
inspections_per_year = df['activity_year'].value_counts().sort_index()

# Quick look
print(inspections_per_year)

# Bar plot
plt.figure(figsize=(10,6))
sns.barplot(x=inspections_per_year.index.astype(int), y=inspections_per_year.values,
palette='pastel')

plt.title('Number of Water Tank Inspections Per Year in NYC')
plt.xlabel('Year')
plt.ylabel('Number of Inspections')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()


import plotly.express as px

fig = px.bar(
    x=inspections_per_year.index.astype(int),
    y=inspections_per_year.values,
    labels={'x': 'Year', 'y': 'Number of Inspections'},
    text=inspections_per_year.values,
    title="Number of Water Tank Inspections Per Year (Interactive)"
)

fig.update_traces(textposition='outside')
fig.update_layout(
    yaxis_title="Number of Inspections",
    xaxis_title="Year",
    xaxis=dict(tickmode='linear')
)
fig.show()
```

```
#Step 3: What % of inspections pass vs fail?
# Overall outcome counts
outcome_counts = df['inspection_outcome'].value_counts()

# Bar plot
plt.figure(figsize=(6,4))
sns.barplot(x=outcome_counts.index, y=outcome_counts.values, palette='pastel')
plt.title('Inspection Outcomes (Pass vs Fail)')
plt.xlabel('Outcome')
plt.ylabel('Number of Inspections')
plt.show()

# Print percentage
total = len(df)
print(f"Pass rate: {outcome_counts['PASS'] / total:.2%}")
print(f"Fail rate: {outcome_counts['FAIL'] / total:.2%}")



# Group by activity year and inspection outcome
inspection_outcomes_by_year = (
    df.groupby(['activity_year', 'inspection_outcome'])
      .size()
      .reset_index(name='count')
)

# Compute percentages within each year
inspection_outcomes_by_year['%'] = (
    inspection_outcomes_by_year
    .groupby('activity_year')['count']
    .apply(lambda x: x / x.sum())
    .reset_index(drop=True)
)

print(inspection_outcomes_by_year)


plt.figure(figsize=(10,6))
sns.barplot(
    data=inspection_outcomes_by_year,
    x='activity_year',
    y='count',
    hue='inspection_outcome',
```

```
    palette='pastel'
)

plt.title('Inspection Outcomes by Year (Pass vs Fail)')
plt.xlabel('Year')
plt.ylabel('Number of Inspections')
plt.grid(axis='y')
plt.legend(title='Outcome')
plt.tight_layout()
plt.show()


# Pivot for easier calculations
pivot = inspection_outcomes_by_year.pivot(index='activity_year',
columns='inspection_outcome', values='count')

pivot['pass_rate'] = (pivot['PASS'] / (pivot['PASS'] + pivot['FAIL'])) * 100
pivot['fail_rate'] = (pivot['FAIL'] / (pivot['PASS'] + pivot['FAIL'])) * 100

pivot['pass_rate'] = pivot['pass_rate'].round(2)
pivot['fail_rate'] = pivot['fail_rate'].round(2)

print(pivot[['pass_rate', 'fail_rate']])


#Which boroughs have more violations?

# Filter only FAIL inspections
failures = df[df['inspection_outcome'] == 'FAIL']

# Count failures per borough
violations_by_borough = failures['borough'].value_counts()

# Plot
plt.figure(figsize=(8,5))
sns.barplot(x=violations_by_borough.index, y=violations_by_borough.values,
palette='pastel')
plt.title('Violations by Borough')
plt.xlabel('Borough')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Show numbers
print(violations_by_borough)
```

```
# Filter all inspections in Staten Island
staten_island_inspections = df[df['borough'] == 'Staten Island']

# Quick check
print(staten_island_inspections.shape)
inspection_counts = staten_island_inspections['inspection_outcome'].value_counts()
print(inspection_counts)

#Where are violations geographically located?
# Simple scatter plot
plt.figure(figsize=(10,8))
plt.scatter(failures['longitude'], failures['latitude'], alpha=0.5, s=10, color='red')

plt.title('Geographical Locations of Violations (Water Tank Inspections)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()


import plotly.express as px

fig = px.scatter_mapbox(
    failures,
    lat='latitude',
    lon='longitude',
    color='borough',
    hover_data=['street_name', 'zip_code', 'date_of_occurrence'],
    zoom=10,
    height=600,
    mapbox_style="open-street-map",
    title="NYC Water Tank Violations by Location"
)

fig.show()


#Does number of tanks affect pass/fail rate?

# Group by number_of_dwt and outcome
tank_group = df.groupby(['number_of_dwt',
'inspection_outcome']).size().unstack(fill_value=0)

# Calculate failure rate
```

```
tank_group['failure_rate'] = tank_group['FAIL'] / (tank_group['FAIL'] +
tank_group['PASS'])

# Plot failure rate vs number of tanks
plt.figure(figsize=(10,6))
tank_group['failure_rate'].plot(kind='bar', color='salmon')

plt.title('Failure Rate by Number of Water Tanks')
plt.xlabel('Number of Tanks in Building')
plt.ylabel('Failure Rate')
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Show table
print(tank_group[['failure_rate']])

tank_distribution = df['number_of_dwt'].value_counts().sort_index()

print(tank_distribution)

plt.figure(figsize=(10,6))
plt.hist(df['number_of_dwt'], bins=range(df['number_of_dwt'].min(),
                                         df['number_of_dwt'].max() + 2),
         edgecolor='black')

plt.title('Number of Water Tanks in Buildings')
plt.xlabel('Number of Tanks')
plt.ylabel('Number of Buildings')
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Aggregate total number of tanks per borough
tanks_by_borough = df.groupby('borough')['number_of_dwt'].sum().reset_index()

# Sort descending for nicer visualization
tanks_by_borough = tanks_by_borough.sort_values(by='number_of_dwt',
ascending=False)

print(tanks_by_borough)

plt.figure(figsize=(10,6))
sns.barplot(
    data=tanks_by_borough,
    x='borough',
```

```
    y='number_of_dwt',
    palette='pastel'
)
plt.title('Total Number of Water Tanks per Borough')
plt.xlabel('Borough')
plt.ylabel('Number of Water Tanks')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Create a cross-tab of number_of_dwt vs borough
tank_counts = pd.crosstab(df['number_of_dwt'], df['borough'])

# Plot
tank_counts.plot(kind='bar', stacked=True, figsize=(12,6), colormap='Pastel1')
plt.title('Distribution of Number of Tanks Across Boroughs')
plt.xlabel('Number of Tanks in Building')
plt.ylabel('Number of Buildings')
plt.legend(title='Borough', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

from scipy.signal import find_peaks

# Do certain buildings have more inspections or violations? Are there buildings that are
repeat violators?
print(f'number of unique bins {df.bin.nunique()}')
max_bin = df['bin'].value_counts().idxmax()
max_bin_count = df['bin'].value_counts().max()

print(f'max_bin: {max_bin}')
print(f'max_bin_count: {max_bin_count}')

inspections_by_bin = df['bin'].value_counts().reset_index()
inspections_by_bin.columns = ['bin', 'num_inspections']

violations_by_bin = df[df['inspection_outcome'] == 'FAIL']
['bin'].value_counts().reset_index()
violations_by_bin.columns = ['bin', 'num_violations']

borough_by_bin = df.groupby('bin')['borough'].agg(lambda x:
x.mode().iloc[0]).reset_index()

inspections_by_bin = inspections_by_bin.merge(borough_by_bin, on='bin', how='left')
violations_by_bin = violations_by_bin.merge(borough_by_bin, on='bin', how='left')
```

```
summary = inspections_by_bin.merge(violations_by_bin,
                                   on = 'bin',
                                   how='left')
summary['num_violations'] = summary['num_violations'].fillna(0).astype(int)

# Analysis of bins by number of inspections:
peak_inspection_bins = inspections_by_bin[inspections_by_bin.num_inspections > 3]
print(f'Number of BINs with over 3 inspections: {len(peak_inspection_bins)}')
print(f'Boroughs of BINs with over 3 inspections:
{peak_inspection_bins.borough.unique()}')

bin_counts_by_borough = peak_inspection_bins['borough'].value_counts()
print(f'bin_counts_by_borough: {bin_counts_by_borough}')

plt.figure(figsize=(8, 5))
sns.barplot(x=bin_counts_by_borough.index, y=bin_counts_by_borough.values)
for i, value in enumerate(bin_counts_by_borough.values):
    plt.text(i, value + 0.5, str(value), ha='center', va='bottom', fontsize=10)
plt.title('BINs with > 3 Inspections by Borough')
plt.xlabel('Borough')
plt.ylabel('Number of BINs')
plt.tight_layout()
plt.show()

# Analysis of bins by number of violations:
peak_violation_bins = violations_by_bin[violations_by_bin.num_violations > 3]
print(f'Number of BINs with over 3 violations: {len(peak_inspection_bins)}')
print(f'Boroughs of BINs with over 3 violations:
{peak_inspection_bins.borough.unique()}')

bin_counts_by_borough = peak_violation_bins['borough'].value_counts()
print(f'bin_counts_by_borough: {bin_counts_by_borough}')

plt.figure(figsize=(8, 5))
sns.barplot(x=bin_counts_by_borough.index, y=bin_counts_by_borough.values, color =
'red')
for i, value in enumerate(bin_counts_by_borough.values):
    plt.text(i, value + 0.5, str(value), ha='center', va='bottom', fontsize=10)
plt.title('BINs with > 3 Violations by Borough')
plt.xlabel('Borough')
plt.ylabel('Number of BINs')
plt.tight_layout()
plt.show()

bin_inspections_desc = inspections_by_bin.sort_values('num_inspections',
```

```
ascending=False)
bin_inspections_desc


# All bins' inspections:
plt.figure(figsize=(20, 6))
sns.barplot(x = 'bin', y = 'num_inspections', data = bin_inspections_desc.head(100),
color = 'blue')
plt.xticks([], [])
plt.title('Top 100 BINs by Number of Inspections')
plt.xlabel('BIN')
plt.ylabel('Number of Inspections')
plt.tight_layout()
plt.show()

# Top 10 bins' inspections:
bin_inspections_desc.head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x = 'bin', y = 'num_inspections', data = bin_inspections_desc.head(10), color
= 'blue')
plt.xticks(rotation = 45)
plt.title('Top 10 BINs by Number of Inspections')
plt.xlabel('BIN')
plt.ylabel('Number of Inspections')
plt.tight_layout()
plt.show()

# Analysis of bins by number of violations:
# All bins' violations:
plt.figure(figsize=(10, 6))
sns.barplot(x = 'bin', y = 'num_violations', data = violations_by_bin, color = 'red')
plt.xticks([], [])
plt.title('BINs by Number of Violations')
plt.xlabel('BIN')
plt.ylabel('Number of Violations')
plt.tight_layout()
plt.show()

# Top 10 bins' violations:
top_bin_violations = summary.sort_values('num_violations', ascending =
False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x = 'bin', y = 'num_violations', data = top_bin_violations, color = 'red')
plt.xticks(rotation = 45)
```

```
plt.title('Top 10 BINs by Number of Violations')
plt.xlabel('BIN')
plt.ylabel('Number of Violations')
plt.tight_layout()
plt.show()

tanks_per_bin = df.groupby('bin')['number_of_dwt'].max().reset_index()
tanks_per_bin.columns = ['bin', 'num_tanks']

# Merge tanks into the top 10 violations table
top_violators_with_tanks = top_bin_violations.merge(tanks_per_bin, on='bin', how='left')

print(top_violators_with_tanks[['bin', 'num_violations', 'num_tanks']])

# BINs with more than 3 violations
bins_over_3_violations = violations_by_bin[violations_by_bin.num_violations > 3]['bin']
df_over_3_violations = df[df['bin'].isin(bins_over_3_violations)]

# Get tanks per BIN (one row per BIN)
tanks_per_bin = df_over_3_violations.groupby('bin')
['number_of_dwt'].max().reset_index()

# Add borough information (each BIN has a single borough)
borough_per_bin = df_over_3_violations.groupby('bin')['borough'].agg(lambda x:
x.mode().iloc[0]).reset_index()

# Merge tanks + borough
tanks_with_borough = tanks_per_bin.merge(borough_per_bin, on='bin', how='left')

# Overall tank distribution
tank_distribution = tanks_with_borough['number_of_dwt'].value_counts().sort_index()
print("Overall tank distribution:")
print(tank_distribution)

# Tank distribution *by borough*
tank_dist_by_borough = (
    tanks_with_borough
    .groupby(['borough', 'number_of_dwt'])
    .size()
    .reset_index(name='count')
)
print("\nTank distribution by borough:")
print(tank_dist_by_borough)


# Create a cross-tab for buildings with >3 violations
```

```
tank_counts_filtered = pd.crosstab(
    tanks_with_borough['number_of_dwt'],
    tanks_with_borough['borough']
)

# Plot in the same style
tank_counts_filtered.plot(
    kind='bar',
    stacked=True,
    figsize=(12,6),
    colormap='Pastel1'
)

plt.title('Distribution of Number of Tanks Across Boroughs (BINs with > 3 Violations)')
plt.xlabel('Number of Tanks in Building')
plt.ylabel('Number of Buildings')
plt.legend(title='Borough', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()


# Enter borough population estimate in 2023
# https://www.nyc.gov/assets/planning/download/pdf/planning-level/nyc-population/population-estimates/current-population-estimates-2023-June2024-release.pdf?r=1
borough_pop_2023 = {
    "Bronx": 1356476,
    "Brooklyn": 2561225,
    "Manhattan": 1597451,
    "Queens": 2252196,
    "Staten Island": 490687
}

df['borough'] = df['borough'].str.strip().str.title()

df['population'] = df['borough'].map(borough_pop_2023)

df.to_csv('with_live_population.csv', index=False)



print(df[['borough', 'population']].drop_duplicates())

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

```
print(f'df.head(): {df.head()}')
borough_and_pop = df[['borough', 'population']]
print(f'df[[\'borough\', \'population\']]: {borough_and_pop}')
# Step 1: Filter only FAIL inspections
failures = df[df['inspection_outcome'] == 'FAIL']

# Step 2: Count number of violations per borough
violations_by_borough = failures['borough'].value_counts().reset_index()
violations_by_borough.columns = ['borough', 'violations']

# Step 3: Merge borough population
violations_by_borough = violations_by_borough.merge(
    df[['borough', 'population']].drop_duplicates(),
    on='borough',
    how='left'
)

# Step 4: Calculate violations per million residents
violations_by_borough['violations_per_million'] = (violations_by_borough['violations'] /
violations_by_borough['population']) * 1_000_000

# Step 5: Sort for nicer plot
violations_by_borough = violations_by_borough.sort_values('violations_per_million',
ascending=False)

# Step 6: Bar Plot
plt.figure(figsize=(10,6))
sns.barplot(x='borough', y='violations_per_million', data=violations_by_borough,
palette='pastel')

plt.title('Violations per Million Residents by Borough')
plt.xlabel('Borough')
plt.ylabel('Violations per Million Residents')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Step 7: (Optional) See numbers
print(violations_by_borough)


import plotly.express as px

# Step 1: Prepare the data
# Make sure "Error: #N/A" borough is removed
```

```
violations_by_borough_clean =
violations_by_borough[violations_by_borough['borough'] != 'Error: #N/A']

# Step 2: Create the interactive Plotly bar plot
fig = px.bar(
    violations_by_borough_clean,
    x='borough',
    y='violations_per_million',
    text='violations_per_million',
    title='Violations per Million Residents by Borough (Interactive)',
    labels={'borough': 'Borough', 'violations_per_million': 'Violations per Million
Residents'},
    color='borough',
    color_discrete_sequence=px.colors.qualitative.Pastel
)

# Step 3: Make it look even better
fig.update_traces(
    texttemplate='%{text:.2f}',
    textposition='outside'
)
fig.update_layout(
    uniformtext_minsize=8,
    uniformtext_mode='hide',
    xaxis_title='Borough',
    yaxis_title='Violations per Million Residents',
    yaxis=dict(showgrid=True),
    xaxis=dict(tickangle=-45),
    showlegend=False,
    height=600,
    width=900
)

# Step 4: Show plot
fig.show()


import plotly.express as px
import requests
import json

# Step 1: Load NYC Borough GeoJSON
geojson_url = 'https://raw.githubusercontent.com/dwillis/nyc-
maps/master/boroughs.geojson'
geojson_data = requests.get(geojson_url).json()
```

```
# Step 2: Prepare cleaned violations_by_borough
violations_by_borough_clean =
violations_by_borough[violations_by_borough['borough'] != 'Error: #N/A']

# Make sure the borough names match exactly with geojson "BoroName"
borough_name_corrections = {
    'Bronx': 'The Bronx',  # Fix for geojson name
    'Brooklyn': 'Brooklyn',
    'Manhattan': 'Manhattan',
    'Queens': 'Queens',
    'Staten Island': 'Staten Island'
}
violations_by_borough_clean['geo_borough'] =
violations_by_borough_clean['borough'].map(borough_name_corrections)

# Step 3: Build Choropleth Map
fig = px.choropleth_mapbox(
    violations_by_borough_clean,
    geojson=geojson_data,
    locations='geo_borough',
    featureidkey='properties.BoroName',  # Important: match to geojson "BoroName"
    color='violations_per_million',
    color_continuous_scale="Reds",
    range_color=(0, violations_by_borough_clean['violations_per_million'].max()),
    mapbox_style="carto-positron",
    zoom=9,
    center={"lat": 40.7128, "lon": -74.0060},
    opacity=0.7,
    labels={'violations_per_million':'Violations/Million'},
    title="NYC Water Tank Violations Per Million Residents (Choropleth)"
)

# Step 4: Show map
fig.update_layout(margin={"r":0,"t":50,"l":0,"b":0})
fig.show()


# Violation Type Analysis
df = pd.read_csv('with_live_population.csv')
print(df.columns)

print(f"Before Cleaning: {df['violation_code'].unique()}")

# Clean "No Violation" and invalid violation code rows.
violations_only = df[(df['violation_code'] != 'No Violation') &
(~df['violation_code'].str.contains('\n'))]
```

```
print(f"After Cleaning: {violations_only['violation_code'].unique()}")

print(violations_only['violation_text'].unique())
violations_only['violation_text'] = violations_only['violation_text'].str.rstrip('.')
print(violations_only['violation_text'].unique())


# Group by borough, violation_code, and violation_text
violation_counts_borough = violations_only.groupby(
    ['borough', 'violation_code', 'violation_text']
).size().reset_index(name='count')

# Sort by borough and count descending
violation_counts_borough = violation_counts_borough.sort_values(
    by=['borough', 'count'], ascending=[True, False]
)

print(violation_counts_borough)

violation_counts_borough.to_csv('violation_counts_borough.csv', index=False)

plt.figure(figsize=(12, 6))

# Barplot: x = violation_text, y = count, hue = borough
sns.barplot(
    data=violation_counts_borough,
    x='violation_code',
    y='count',
    hue='borough',
    errorbar=None
)

plt.xticks(rotation=90)
plt.xlabel('Violation Code')
plt.ylabel('Count')
plt.title('Counts of Violations by Borough')
plt.legend(title='Borough')
plt.tight_layout()
plt.show()

import pandas as pd

def run_data_quality_checks(file_path):
    """
    Runs data quality tests on the cleaned dataset.
    """
```

```
    df = pd.read_csv(file_path)

    # Remove trailing whitespace
    df['borough'] = df['borough'].str.strip()

    print("Running data quality checks...")

    # Test 1: Borough should not be null
    assert df['borough'].isnull().sum() == 0, "Missing boroughs found."

    # Test 2: Only 5 unique boroughs expected
    assert df['borough'].nunique() == 5, "Unexpected number of boroughs after cleaning."

    # Test 3: Population must not have missing values
    assert df['population'].isnull().sum() == 0, "Missing population after enrichment."

    # Test 4: Verify that the 5 unique boroughs found in the dataset are actual NYC
boroughs.
    assert df['borough'].isin(
        ['Manhattan', 'Brooklyn', 'Bronx', 'Queens', 'Staten Island']
    ).all(), "DataFrame contains invalid borough names!"

    # Test 5: Reasonable number of total rows
    assert df.shape[0] > 8000, "Unexpectedly few inspection records."

    # Test 6: No duplicate rows
    assert df.duplicated().sum() == 0, "Duplicate rows found."

    # Test 7: Longitude range sanity check
    assert ((df['longitude'].astype(float) >= -74.3) & (df['longitude'].astype(float) <=
-73.7)).all(), "Longitude values out of NYC expected range."

    # Test 8: Latitude range sanity check
    assert ((df['latitude'].astype(float) >= 40.4) & (df['latitude'].astype(float) <=
40.95)).all(), "Latitude values out of NYC bounds."

    print("All data quality checks passed successfully!")

if __name__ == "__main__":
    run_data_quality_checks('with_live_population.csv')


import pandas as pd
import requests
from io import StringIO
```

```
APP_TOKEN = "removed"
API_KEY = "removed"

url = "https://data.cityofnewyork.us/api/v3/views/5zhs-2jue/query.csv"

headers = {
    "X-App-Token": APP_TOKEN
}

response = requests.get(url, headers=headers)

if response.status_code == 200:
    buildings_df = pd.read_csv(StringIO(response.text))
    print(buildings_df)
else:
    print(f"Request failed with status code {response.status_code}")


display(buildings_df.columns)

display(buildings_df.shape)

print(buildings_df["base_bbl"].unique())

# https://a836-pts-access.nyc.gov/care/search/commonsearch.aspx?mode=persprop
# Shows the borough numbers in BBL
borough_mapping = {
  1: "Manhattan",
  2: "Bronx",
  3: "Brooklyn",
  4: "Queens",
  5: "Staten Island"
}

buildings_df['borough_code'] = buildings_df['base_bbl'].astype(str).str[0].astype(int)
buildings_df

buildings_df.shape

water_tank_df = pd.read_csv("nyc_water_tank_compliance_ready_for_analysis.csv")
water_tank_df

results = []

for code, borough_name in borough_mapping.items():
    # Filter buildings for this borough
```

```
    borough_buildings = buildings_df[buildings_df['borough_code'] == code]

    # Get unique BINs
    building_bins = set(borough_buildings['bin'].unique())
    tank_bins = set(water_tank_df[water_tank_df['borough'] == borough_name]
['bin'].unique())

    # Find overlap
    bins_in_both = building_bins & tank_bins

    # Compute proportion
    proportion_reported = len(bins_in_both) / len(building_bins) if len(building_bins) > 0
else 0

    # Add to results
    results.append({
        'Borough': borough_name,
        'Unique Buildings in Water Tank Dataset': len(bins_in_both),
        'Total Unique Buildings': len(building_bins),
        'Proportion Reported': proportion_reported
    })

# Convert results to DataFrame
borough_summary_df = pd.DataFrame(results)

# Format proportion as percentage
borough_summary_df['Proportion Reported'] = borough_summary_df['Proportion
Reported'].apply(lambda x: f"{x:.2%}")

borough_summary_df = borough_summary_df.reset_index(drop=True)
borough_summary_df

avg_floor_height = 10
buildings_df['estimated_floors'] = (buildings_df['height_roof'] /
avg_floor_height).apply(np.ceil)
buildings_df

# Estimate number of floors
avg_floor_height = 10
buildings_df['estimated_floors'] = (buildings_df['height_roof'] /
avg_floor_height).apply(np.ceil)

buildings_df['borough_code'] = buildings_df['base_bbl'].astype(str).str[0].astype(int)
buildings_df['borough'] = buildings_df['borough_code'].map(borough_mapping)

tall_buildings_df = buildings_df[buildings_df['estimated_floors'] > 6]
```

```
# Tall buildings summary by borough
tall_summary = tall_buildings_df.groupby('borough')['bin'].nunique().reset_index()
tall_summary.rename(columns={'bin': 'Buildings > 6 Floors'}, inplace=True)

# Total buildings by borough
total_summary = buildings_df.groupby('borough')['bin'].nunique().reset_index()
total_summary.rename(columns={'bin': 'Total Buildings'}, inplace=True)

borough_summary_df = pd.merge(tall_summary, total_summary, on='borough')

borough_summary_df['Proportion > 6 Floors to Total Buildings'] =
borough_summary_df['Buildings > 6 Floors'] / borough_summary_df['Total Buildings']

borough_summary_df['Proportion > 6 Floors to Total Buildings'] =
borough_summary_df['Proportion > 6 Floors to Total Buildings'].apply(lambda x:
f"{x:.2%}")

total_row = pd.DataFrame([['Total',
                           borough_summary_df['Buildings > 6 Floors'].sum(),
                           borough_summary_df['Total Buildings'].sum(),
                           f"{borough_summary_df['Buildings > 6 Floors'].sum() /
borough_summary_df['Total Buildings'].sum():.2%}"]],
                         columns=borough_summary_df.columns)
borough_summary_df = pd.concat([borough_summary_df, total_row],
ignore_index=True)

borough_summary_df asdf
```

7 References

Department of Health and Mental Hygiene, NYC (2025). *NYC Drinking Water Tank Inspections and Audits Compliance Results* [Data set]. NYC Open Data. https://data.cityofnewyork.us/ (Accessed November 29, 2025)

Department of Health and Mental Hygiene, NYC (2025). *BUILDING* [Data set]. NYC Open Data. https://data.cityofnewyork.us/ (Accessed December 6, 2025)

"Drinking Water Tank Inspection Reporting." *Drinking Water Tank Inspection Reporting - NYC Health*, www.nyc.gov/site/doh/business/permits-and-licenses/drinking-water-tank-inspection-reporting.page. Accessed 13 Nov. 2025.

"Inside City's Water Tanks, Layers of Neglect (Published 2014)." The New York Times, 27 Jan. 2014, www.nytimes.com/2014/01/27/nyregion/inside-citys-water-tanks-layers-of-neglect.html.

Is Poor Maintenance of Rooftop Water Tanks Endangering New York City's High Quality Drinking Water?, www.nrdc.org/bio/eric-goldstein/poor-maintenance-rooftop-water-tanks-endangering-new-york-citys-high-quality. Accessed 13 Nov. 2025.

NYC Department of City Planning. "Population Estimates for New York City and Boroughs, Vintage 2023." NYC.Gov, 14 Mar. 2024, www.nyc.gov/assets/planning/download/pdf/planning-level/nyc-population/population-estimates/current-population-estimates-2023-June2024-release.pdf?r=1.

NYC Environmental Protection. "NEW YORK CITY DRINKING WATER SUPPLY AND QUALITY

REPORT 2023." *NYC.Gov*, 2023, www.nyc.gov/assets/dep/downloads/pdf/water/drinking-water/drinking-water-supply-quality-report/2023-drinking-water-supply-quality-report.pdf.

Runyeon, Frank G. "Lax Oversight, Dubious Testing in Water Tanks Pose Health Risks." *City & State NY*, City & State New York, 28 May 2018, www.cityandstateny.com/policy/2018/05/lax-oversight-dubious-testing-in-water-tanks-pose-health-risks/178440/.

Runyeon, Frank G. "New York City to Pass Water Tank Reforms." City & State NY, City & State New York, 6 July 2021, www.cityandstateny.com/policy/2019/04/new-york-city-to-pass-water-tank-reforms/177496/.