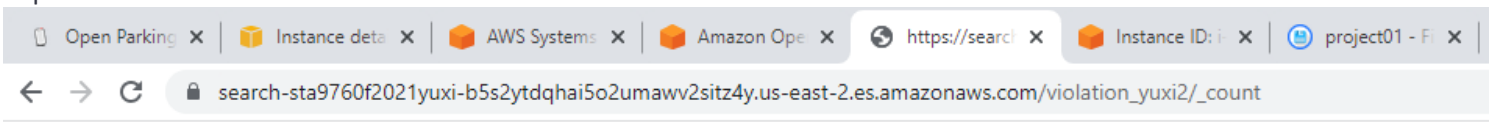
Yuxi Song

I uploaded total 1m+ records into Elasticsearch:




```
{"count":1023820,"_shards":{"total":1,"successful":1,"skipped":0,"failed":0}}
```


Steps to Build:

- Command prompt:
  a. cd project01/
  b. docker build -t project01:1.0 .
  c. docker run project01:1.0 python src/main.py
  d. docker run -v $PWD:/app project01:1.0 python src/main.py
  e. docker run \
     -e DATASET_ID="nc67-uf89" \
     -e APP_TOKEN="Tcnb7vEIccElonCJWpcHrQeNr" \
     -e ES_HOST="https://search-sta9760f2021yuxi-b5s2ytdqhai5o2umawv2sitz4y.us-east-2.es.amazonaws.com" \
     -e INDEX_NAME="violation_yuxi2" \
     -e ES_USERNAME="yuxi" \
     -e ES_PASSWORD='Syxjzl1031##' \
     project01:1.0 --num_pages=X --page_size=Y
  - DATASET_ID: the dataset identifier for Open Parking and Camera Violations
  - APP_TOKEN: the unique key for accessing the API
  - ES_HOST: the Elasticsearch host address
  - ES_USERNAME: the Elasticsearch username
  - ES_PASSWORD: the Elasticsearch password
  - page_size: an optional argument to specify how many records to request from the API per call. If not provided, it defaults to 1,000 records per request
  - num_pages: an optional argument to specify how many calls to make to the API. If not provided, it defaults to requesting all data in the dataset

Command Line Arguments:

- I used argparse library for handling of the command line argument, page_size and num_pages.

```
parser = argparse.ArgumentParser(description="Process data from violation.")
parser.add_argument("--page_size", type=int, help="How many rows to fetch per page", required=True)
parser.add_argument("--num_pages", type=int, help="How many pages to fetch")
args = parser.parse_args(sys.argv[1:])
print(args)
```

- I used sodapy to import Socrata Open data API(To get dataset from Open Parking and Camera Violations).

Elasticsearch Index and Mapping:

I wrote a function that sets the mapping for uploaded documents.

```
"settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
},
"mappings": {
    "properties": {
        "plate": {"type": "text"},
        "state": {"type": "text"},
        "license_type": {"type": "text"},
        "summons_number": {"type": "text"},
        "issue_date": {"type": "date", "format": "MM/dd/yyyy"},
        "violation": {"type": "keyword"},
        "fine_amount": {"type": "float"},
        "penalty_amount": {"type": "float"},
        "interest_amount": {"type": "float"},
        "reduction_amount": {"type": "float"},
        "payment_amount": {"type": "float"},
        "amount_due": {"type": "float"},
        "precinct": {"type": "keyword"},
        "county": {"type": "keyword"},
    }
}
```

Usage of Elasticsearch:

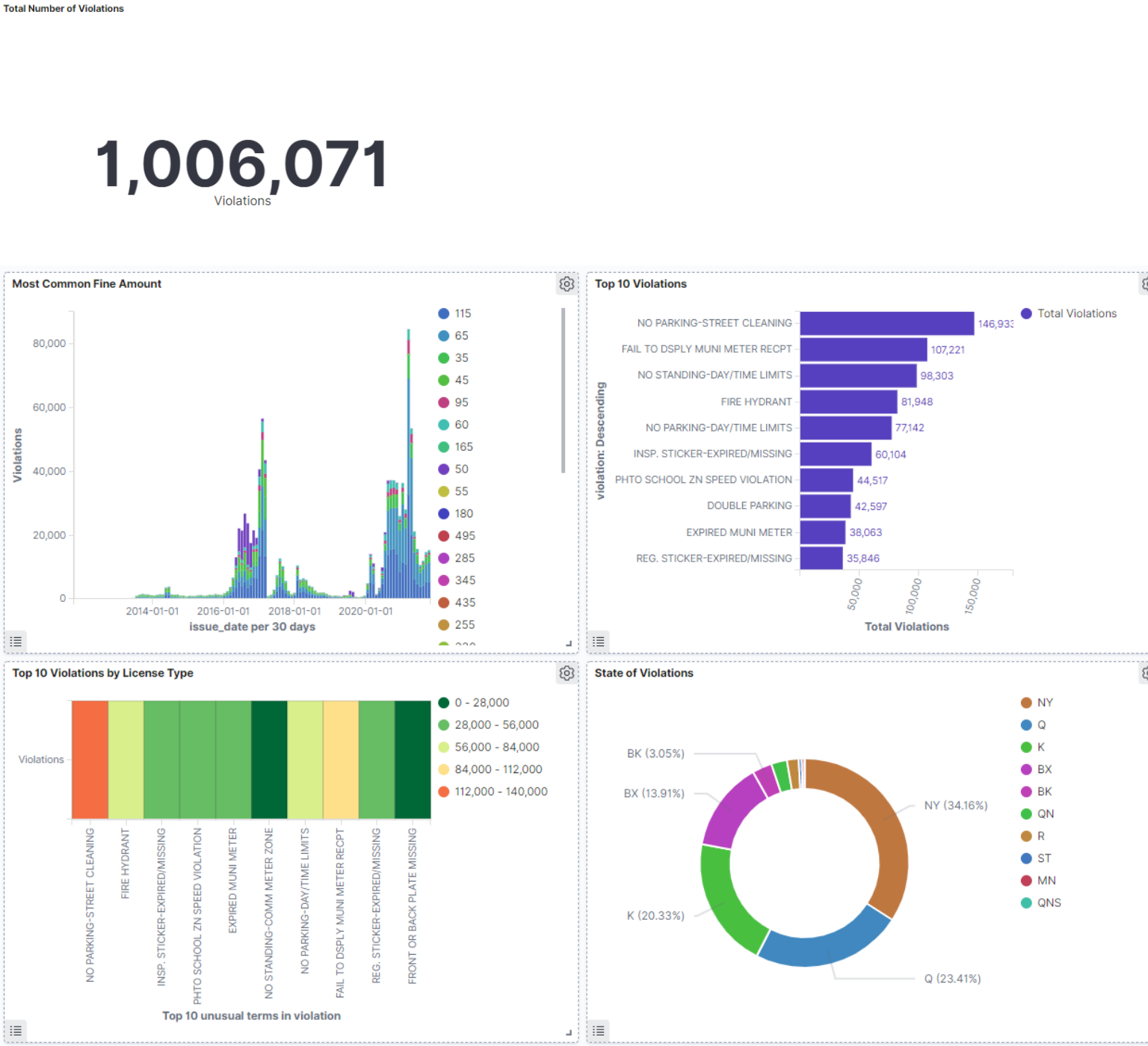
I chose to use the Bulk API to upload a huge volume of data to Elasticsearch. I could upload to Elasticsearch the full page_size of documents for each call to the API.

```
bulk_upload_data = ""
for i, line in enumerate(es_rows):
    print(f"Handling row {line['summons_number']} {i}")
    action = '{"index": {"_index": "'+INDEX_NAME+'", "_type" : "_doc"}}'
    data = json.dumps(line)
    bulk_upload_data += f"{action}\n"
    bulk_upload_data += f"{data}\n"

print(bulk_upload_data)
```

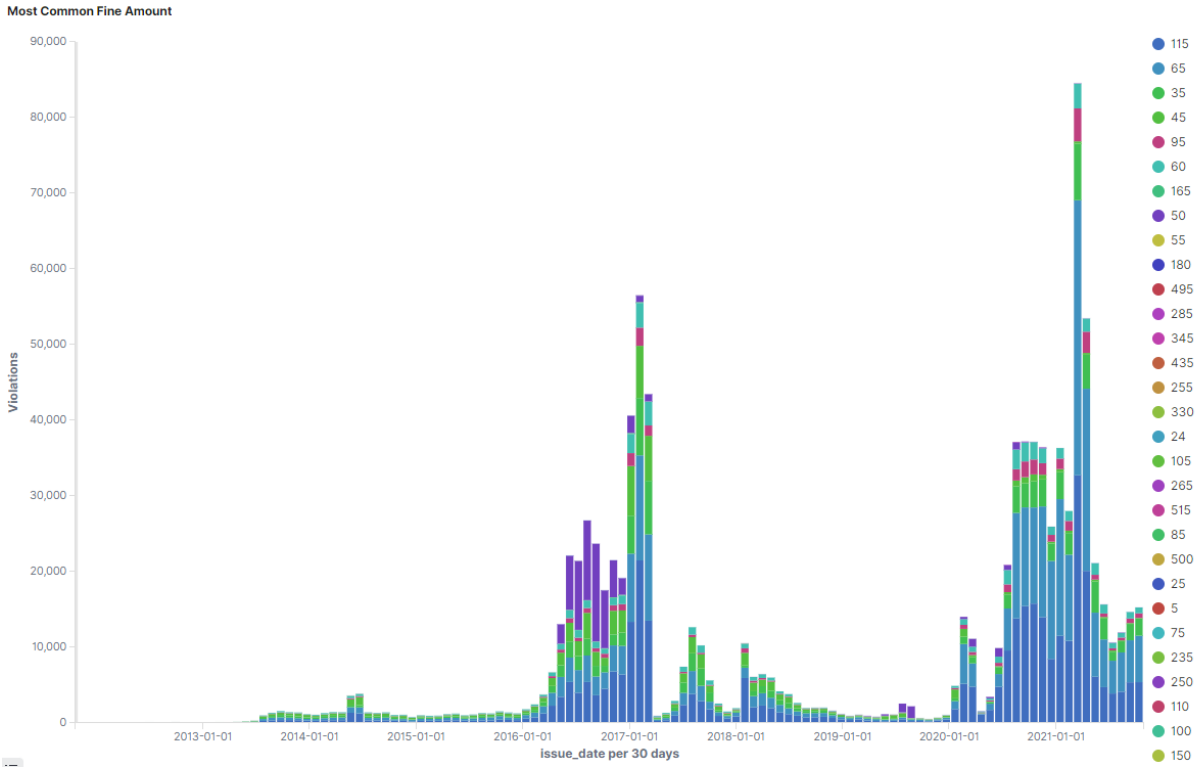
Blog Post: Kibana Visualizations:

I built a dashboard. Timeframe: Last 10 years.

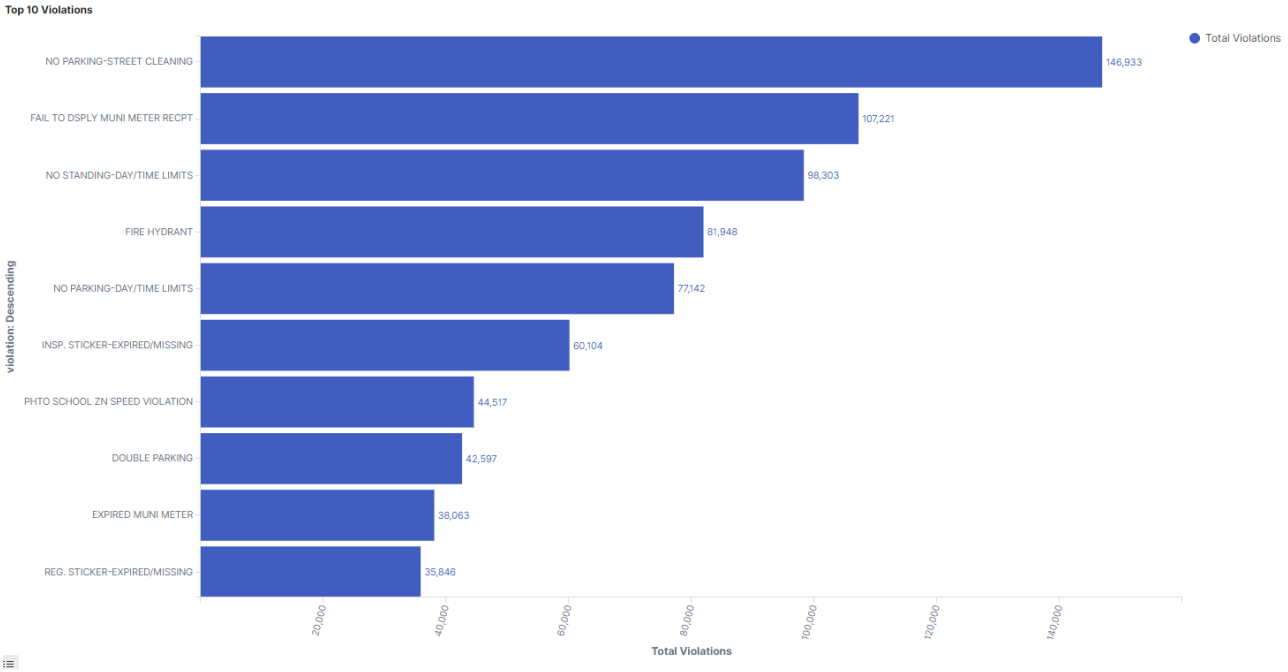


Visualizations:

I had uploaded around 1 million records from the Open Parking and Camera Violation dataset to Elasticsearch. My first visualization displays fine_amount within the last 10 years. According to the chart, the most common fine amount is $115. It means most drivers have to pay more than $100 dollars once they violate the rules. In other words, the fine amount is almost the same as some drivers' daily earnings. If you want to save money, you would follow rules.



My second visualization shows the top 10 violations in the last 10 years. The most frequent violation is "No Parking-Street Cleaning" From this visualization, drivers need to check the date of street cleaning when they are parking, or they can move cars before street cleaning. If we are paying more attention, this type of violation could avoid.



More specifically, my third visualization shows the top 10 violations by license type “PAS”. License type “PAS” has the most violations due to No Parking – Street Cleaning. According to my second and third visualization, license type “PAS” violated 89%(131,207/146,933=89%) of Top 1 violation(“No Parking – Street Cleaning”) in the last 10 years.

My fourth visualization states that NY has the most violations. Everyone is in a hurry in NY, and a lot of people live and work in NY, so drivers can’t completely avoid parking violations. Therefore, we should be careful when we drive in NY.