CSDA1020 - Big Data Analytics Tools
Project 4
ELK Data Analytics
& Visualization

by:

Sam Vuong

Raymond Huang

Carmon Ho

Table of Contents

Introduction

In this project we will be using the **NYC OpenData** "**311 Service Requests** from 2010 to Present" dataset from the website:

https://nycopendata.socrata.com/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

The data is provided by 311 Department of Information Technology. Dataset owner is NYC OpenData. The database was created on October 10th 2011 and is updated daily. The Metadata was last updated on April 22nd 2020.

We downloaded a copy of the dataset on Sunday May 10th 2020. The dataset has approximately 22.8 million rows. Each row is a 311 Service Request.

The dataset is in CSV format and has 41 columns. The data definition of the columns is in the Appendix.

For the project, we will:

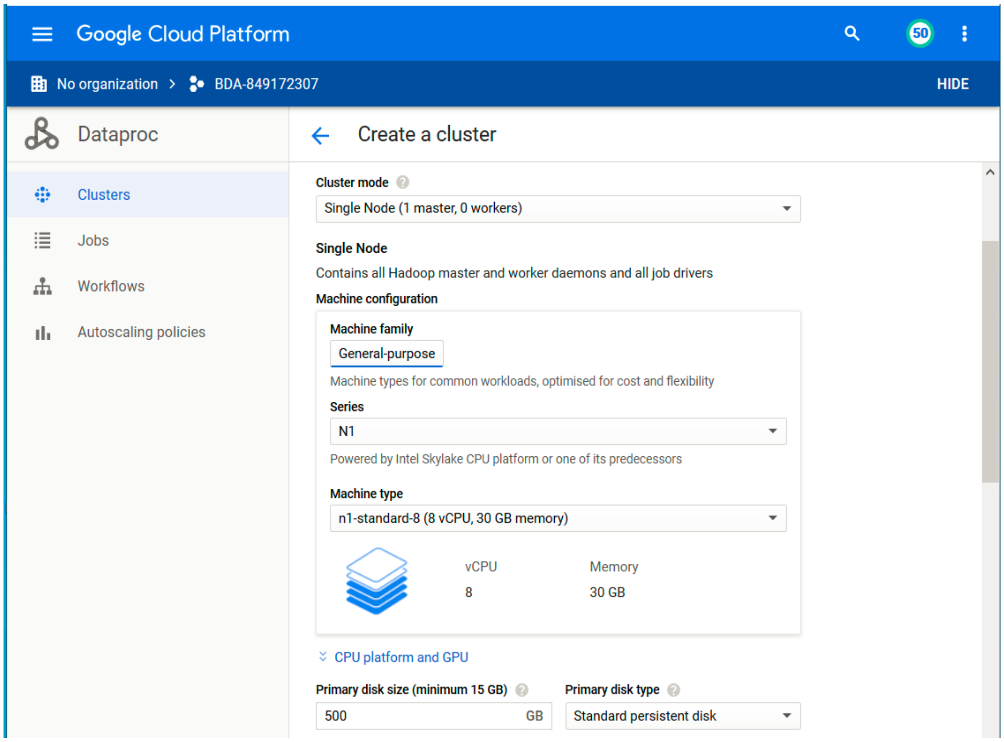
- Set-up a **Hadoop Cluster** on Google Cloud.
- Install and configure **ELK stack.**
- Prepare the **data mappings** and config file.
- Run **Logstash to load data** in the dataset into **Elasticsearch**.
- Use **Kibana visualizations and dashboard** features for our data analytics.

Setup ELK Stack

1. Setup Hadoop Cluster on Google Cloud

We create a Hadoop Cluster on Google Cloud with the following properties:

- Use default values for all properties, except the followings:
- Cluster mode: **Single Note (1 master, 0 workers)**
- Machine type: **n1-standard-8 (8vCPU, 30 GB memory)**
- Primary disk size: **500 GB**



2. Install and Configure ELK Stack Software

1. Install and Configure Elasticsearch v7.5.1:

We run the following 2 Linux commands to **download** and **install** Elasticsearch v7.5.1 on node machine.

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.5.1-linux-x86_64.tar.gz
tar -xzf elasticsearch-7.5.1-linux-x86_64.tar.gz
```

```
vi elasticsearch-7.5.1/config/elasticsearch.yml
```

We use Linux "vi" command above to **edit** Elasticsearch config file and set the following property values:

- **network.host:** 0.0.0.0
- **discovery.seed_hosts**: ["10.128.0.8:9300"] ### with the internal IP address of the cluster.
- **cluster.initial_master_nodes**: ["cluster-elk-stack-m"] ### with the name of cluster.

We run the below Linux commands to **start** Elasticsearch server (in background mode).

Note that we need to set **vm.max_map_count**=262144, otherwise Elasticsearch will not startup.

```
sudo sysctl vm.max_map_count=262144
cd elasticsearch-7.5.1/
bin/elasticsearch -d
```

2. Install and Configure Kibana v7.5.1

We run the following 2 Linux commands to **download** and **install** Kibana v7.5.1 on the node machine.

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-7.5.1-linux-x86_64.tar.gz
tar -xzf kibana-7.5.1-linux-x86_64.tar.gz
```

```
vi kibana-7.5.1-linux-x86_64/config/kibana.yml
```

We use Linux "vi" command above to **edit** Kibana config file and set the following property values:

- **server.port**: 5601
- **server.host**: "0.0.0.0"

We run the below Linux commands to **start** Kibana server.

```
cd kibana-7.5.2-linux-x86_64/
bin/kibana
```

3. Install and Configure Logstash v7.5.1

We run the following 2 Linux commands to download and install Logstash v7.5.1 on the node machine.

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-7.5.1.tar.gz
tar -xzf logstash-7.5.1.tar.gz
```

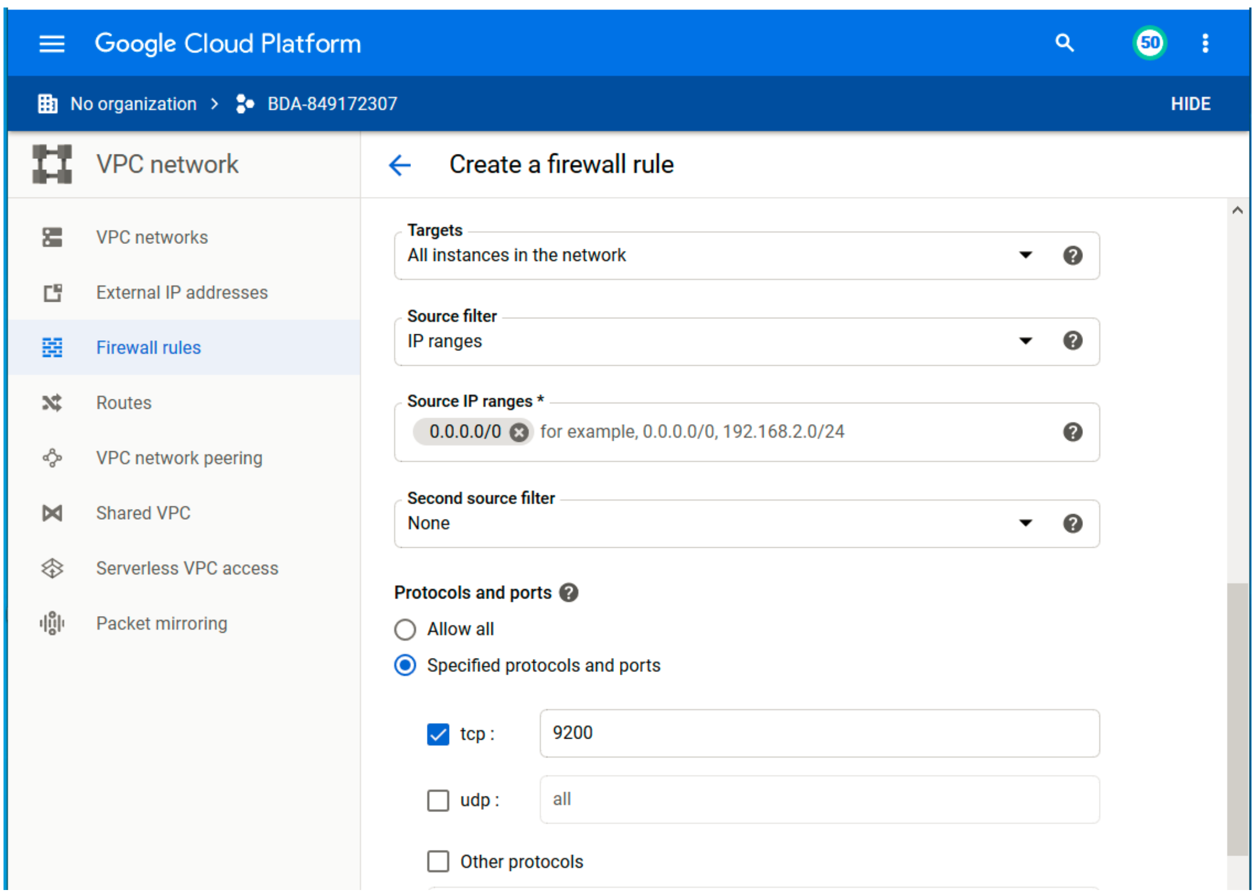
We use all Logstash config files "as is" -- there is no need to make any custom configuration changes.

3. Setup Firewall Rules for ELK Stack

1. Firewall Rule for Elasticsearch

We create a firewall rule for Elasticsearch with the following property values:

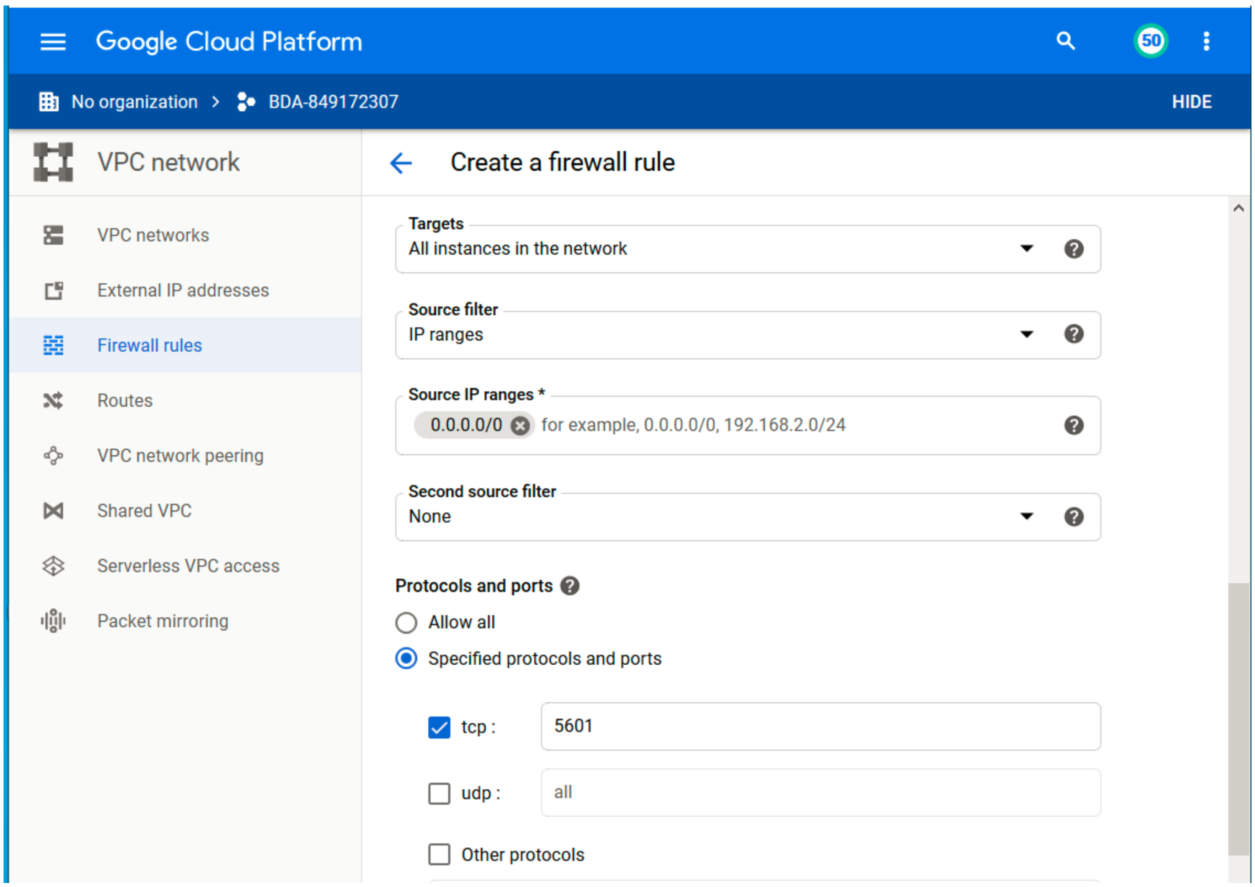
- Use default values for all properties, except the followings:
- Targets: **All instance in the network**
- Source filters: **IP ranges**
- Source IP ranges: **0.0.0.0/0**
- Specified protocols and ports: **tcp: 9200**



2. Firewall Rule for Kibana

We create a firewall rule for Kibana with the following property values:

- Use default values for all properties, except the followings:
- Targets: **All instance in the network**
- Source filters: **IP ranges**
- Source IP ranges: **0.0.0.0/0**
- Specified protocols and ports: **tcp: 5601**



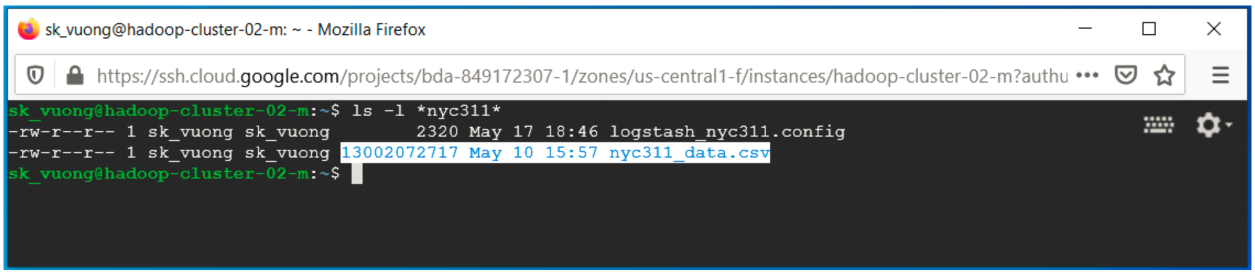
Loading Data

1. Download Input Dataset

We run the following Linux commands to **download** the dataset from NYC311 website and rename it to "**nyc311_data.csv**".

```
wget https://nycopendata.socrata.com/api/views/erm2-nwe9/rows.csv?accessType=DOWNLOAD
mv rows.csv?accessType=DOWNLOAD nyc311_data.csv
```

The dataset is approximately **13GB** in size and has approximately **22.8 million rows**. The dataset is in CSV format and has **41 columns**. The data definition of the columns is in the Appendix.



2. Prepare Geo_Point Mapping for Location

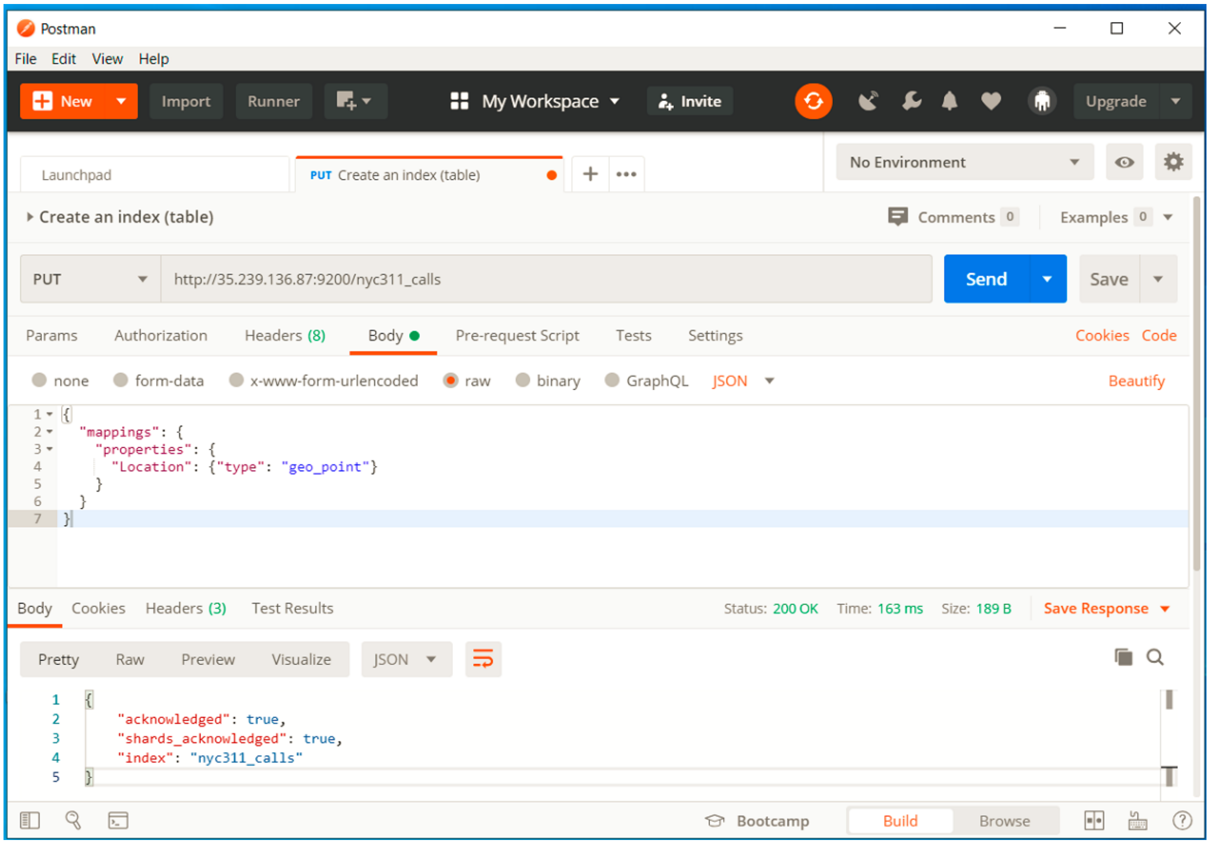
In order to perform "Coordinate Map" visualization with Kibana, we need to set-up "geo_point" type mapping for the Location field.

We can execute the mapping code below either directly on a Linux shell window or using the Postman tool as shown in the screenshot below.

<u>Mapping code used</u>:

```
curl -H "Content-Type: application/json" -XPUT 127.0.0.1:9200/nyc311_calls -d '
{
  "mappings": {
          "properties": {
          "Location": {"type": "geo_point"}
          }
  }
}'
```

Postman Screenshot:



3. Prepare Logstash Config File

Logstash config file has 3 sections: input, filter and output.

1. Input section:

In the input section, we specify the **path** to the input data file and the **starting** position.

Below is the code for the input section in our Logstash config file.

```
input {
   file {
          path=>"/home/sk_vuong/nyc311_data.csv"
          start_position => "beginning"
          sincedb_path => "/dev/null"
   }
}
```

2. Filter section:

In the filter section, we specify details about our input file and data fields and all the data mappings and conversions.

Below is the code for the “csv” section within the filter section. In this “csv” section we use 3 parameters for our input data file:

1. **separator** ### This is used to specify the separator between data values in the input file.
2. **skip header** ### This is used to specify that the input file has a header row.
3. **columns** ### This is used to specify all column names.

```
filter {
	csv {
		separator => ","
		skip_header => true
		columns => ["Unique Key","Created Date","Closed Date","Agency","Agency
		Name","Complaint Type","Descriptor","Location Type","Incident Zip","Incident
		Address","Street Name","Cross Street 1","Cross Street 2","Intersection Street
		1","Intersection Street 2","Address Type","City","Landmark","Facility
		Type","Status","Due Date","Resolution Description","Resolution Action Updated
		Date","Community Board","BBL","Borough","X Coordinate (State Plane)","Y
		Coordinate (State Plane)","Open Data Channel Type","Park Facility Name","Park
		Borough","Vehicle Type","Taxi Company Borough","Taxi Pick Up Location","Bridge
		Highway Name","Bridge Highway Direction","Road Ramp","Bridge Highway
		Segment","Latitude","Longitude","Location"]
	}
	### The code for mappings / conversions for data fields are placed after this ###
}
```

After the “csv” section, we will have code to perform mappings / conversions for data fields that are required for our data analytics and visualizations.

The code below performs the following data mapping mappings / conversions:

1. Convert **City** name to upper -- to avoid multiple values for the same city name.
2. Convert **Latitude** and **Longitude** type “float” data type -- default data type is “string”.
3. Set value for **Location** based on Latitude and Longitude values.
4. Set **Location** to [0,0] for events with **null value** for Latitude and Longitude.
5. Set Logstash **@timestamp** field to "**Created Date**" -- default is timestamp as events are loaded.
6. Convert "**Created Date**" and "**Closed Date**" to “date” data type -- default data type is “string”.
7. Calculate “'**Duration**” = "Closed Date" - "Created Date" for events with Status = "Closed".

```
filter {
	### The code for the “csv” section is placed before this ###
	mutate {uppercase => ["City"] }
	mutate {convert => ["Latitude", "float"]}
```

```
        mutate {convert => ["Longitude", "float"]}
        mutate {replace => {"Location" => "%{Latitude},%{Longitude}"}}

        # Set Location to [0.0,0.0] for records with null value for Latitude/Longitude
        if [Latitude] == "" or [Longitude] == "" or [Location] == "" or [Location] ==
        "%{Latitude},%{Longitude}" {
                mutate {replace => {"Location" => "0.0,0.0"}}
        }
        date {
                timezone => "US/Eastern"
                match => [ "Created Date", "MM/dd/yyyy hh:mm:ss aa" ]
                target => "@timestamp"
        }
        date {
                timezone => "US/Eastern"
                match => [ "Created Date", "MM/dd/yyyy hh:mm:ss aa" ]
                target => "Created Date"
        }
        date {
                timezone => "US/Eastern"
                match => [ "Closed Date", "MM/dd/yyyy hh:mm:ss aa" ]
                target => "Closed Date"
        }
        if [Status] == "Closed" {
                ruby {
                        init => "require 'time'"
                        code => "duration = ( (event.get('Closed Date') - event.get('Created Date')) /
                        (3600 * 24) ) rescue nil; event.set('Duration (in Days)', duration); "
                }
        }}
```

3. Output section:

In the output section, we specify the **hosts** ("localhost") and the **index name** ("nyc311_calls").

Below is the code for the output section in our Logstash config file.

```
output {
   elasticsearch{
        hosts => "localhost"
        index => "nyc311_calls"
        # document type => "_doc"
   }
   # stdout {codec => dots}
}
```

<u>**Notes**</u>:

- For **document type** parameter:

  We **commented out** the document type parameter setting to avoid the warning from Logstash. Document type is a deprecated config setting, which will be removed in Logstash 7.0.

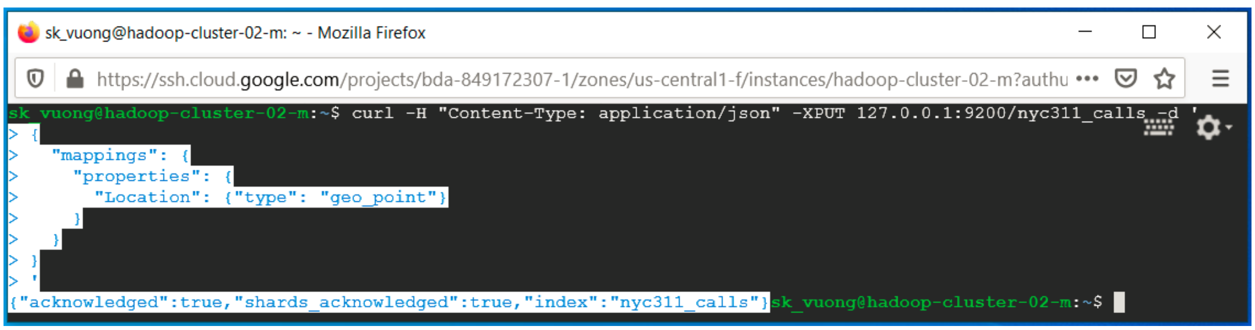
  Note also that after the **"geo point" type mapping** code for the Location field being executed, if the parameter "document type" has any value setting other than "**_doc**", Logstash loading will **fail with the error message**: "Rejecting mapping update to [index name] as the final mapping would have more than 1 type.

- For **stdout** parameter:

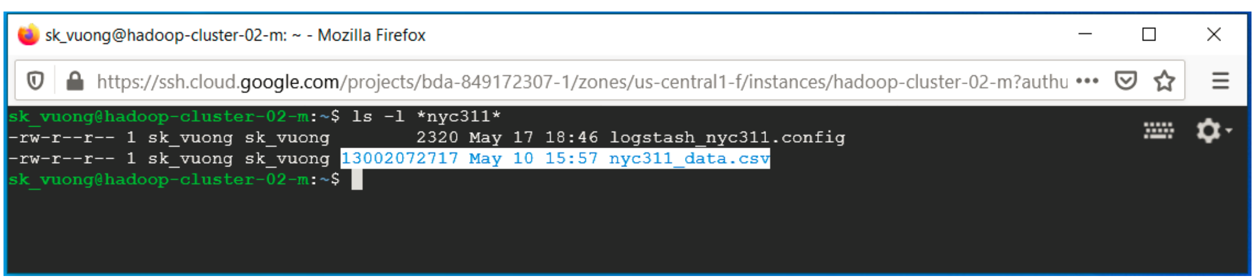
  We **commented out** the stdout parameter to avoid printing out info messages for each event being loaded. This helps speed up the loading time a bit, especially when our input dataset has 22.8 million records.

4. Run Logstash to Load data into Elasticsearch

1. We execute the "geo_point" **mapping code** for the Location field directly on a Linux shell window:



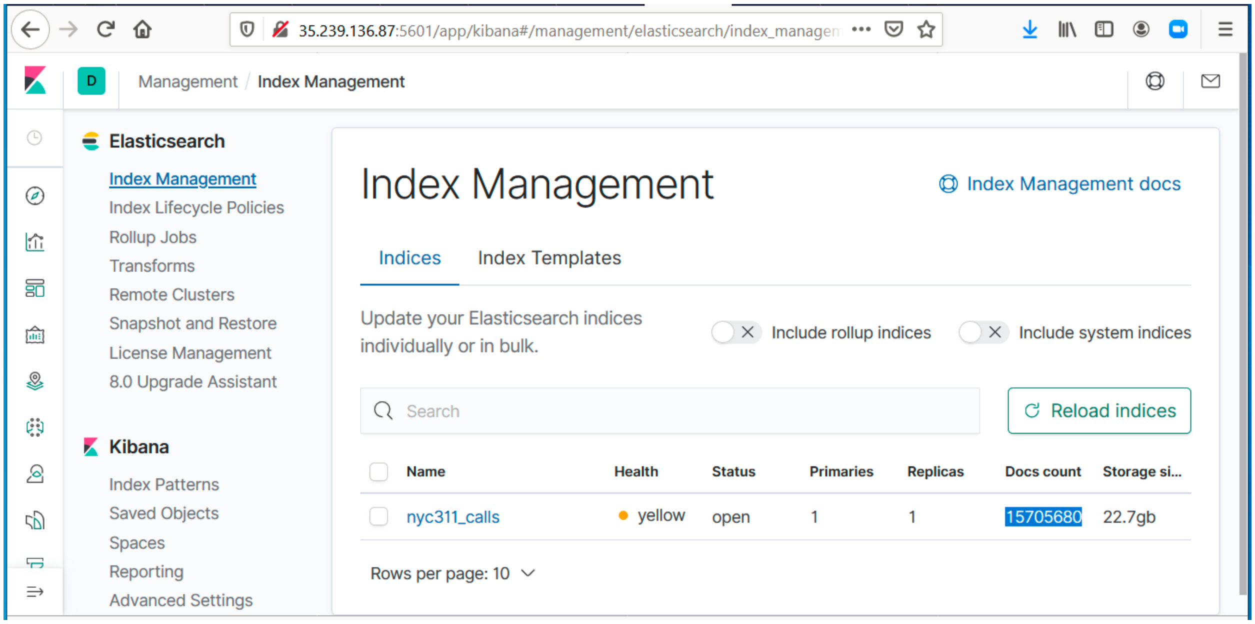
2. We place both the Logstash **config file** and **input dataset** in the HOME directory as shown in screenshot:



3. We execute the following Linux commands to **run Logstash** to **load data** into Elasticsearch:

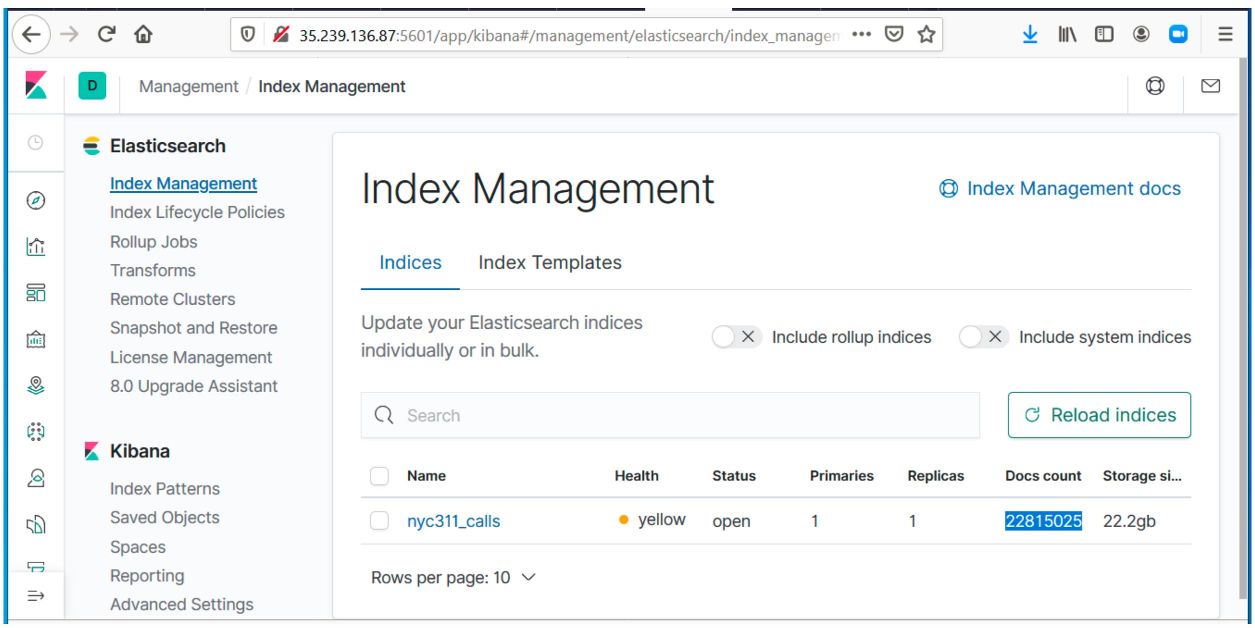
```
cd logstash-7.5.2/
bin/logstash -f ~/logstash_nyc311.config
```

4. We use the **Kibana Index Management** page to monitor the data loading (check the count) of our index as shown in the screenshot below. Logstash reaches the end of the dataset when the count stops increasing.

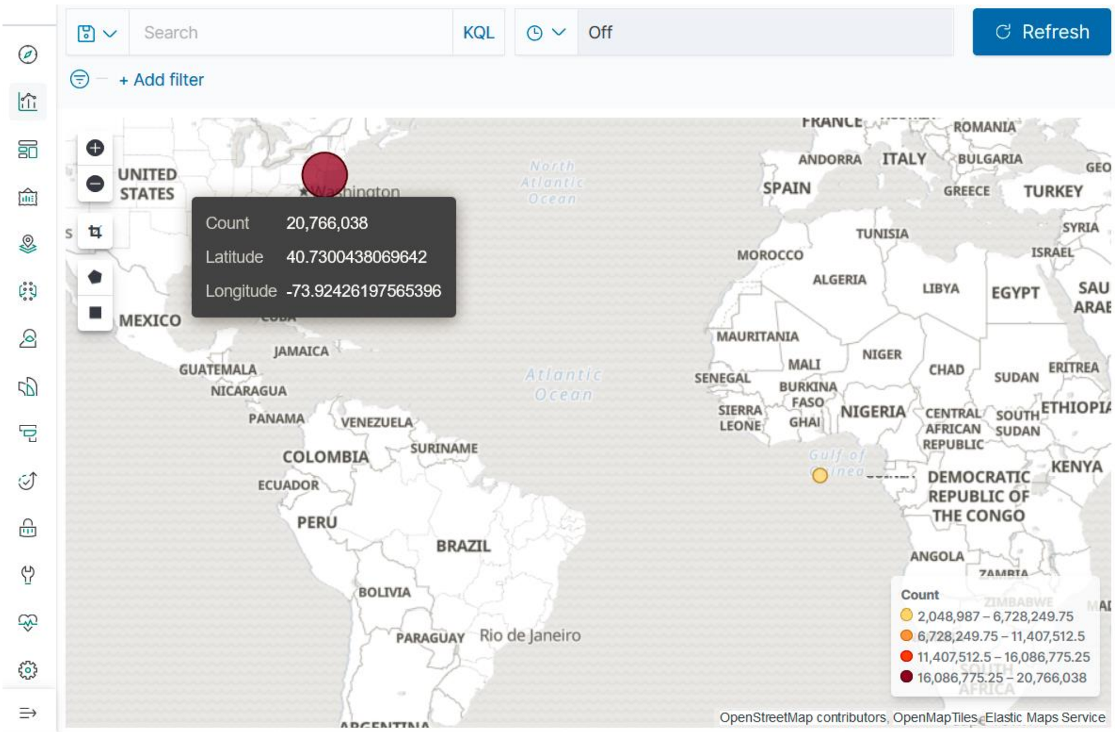


4. We check Logstash loading results:

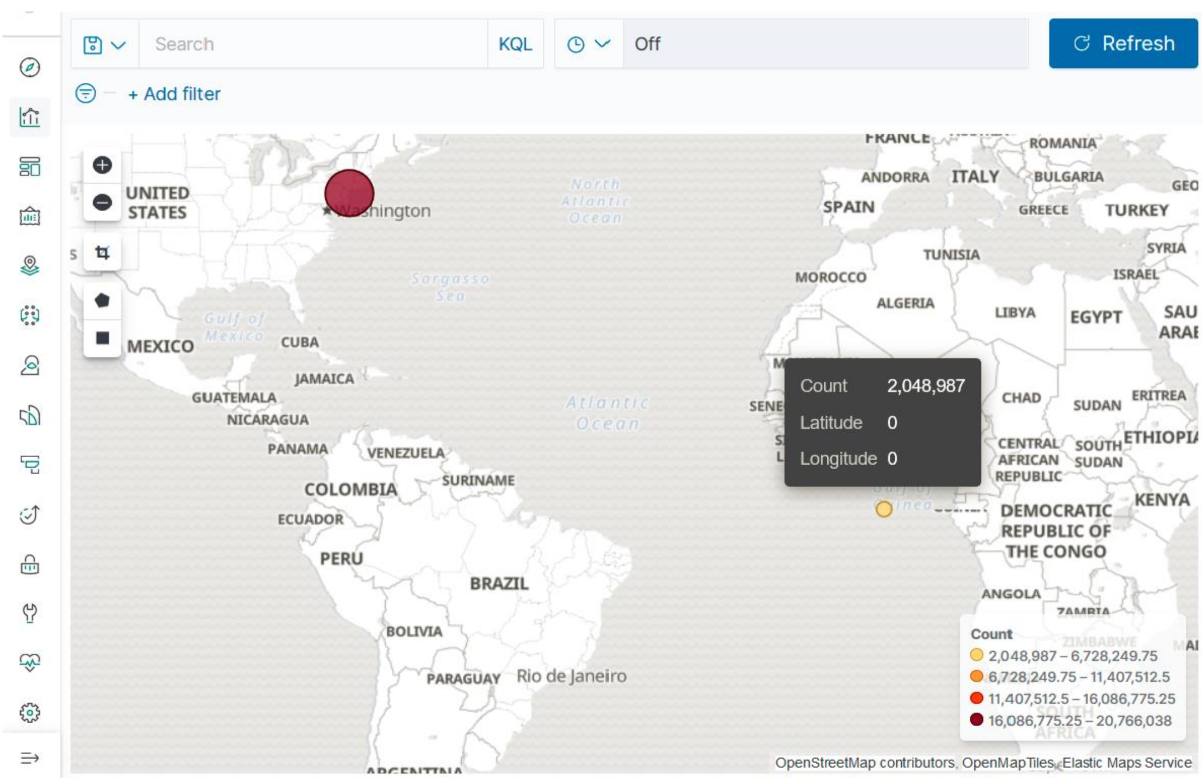
a) It took Logstash approximately **70 minutes** to finish and loaded **22,815,025 records** (documents).



b) There are **20,766,038 events** with values for Location within New York city area:



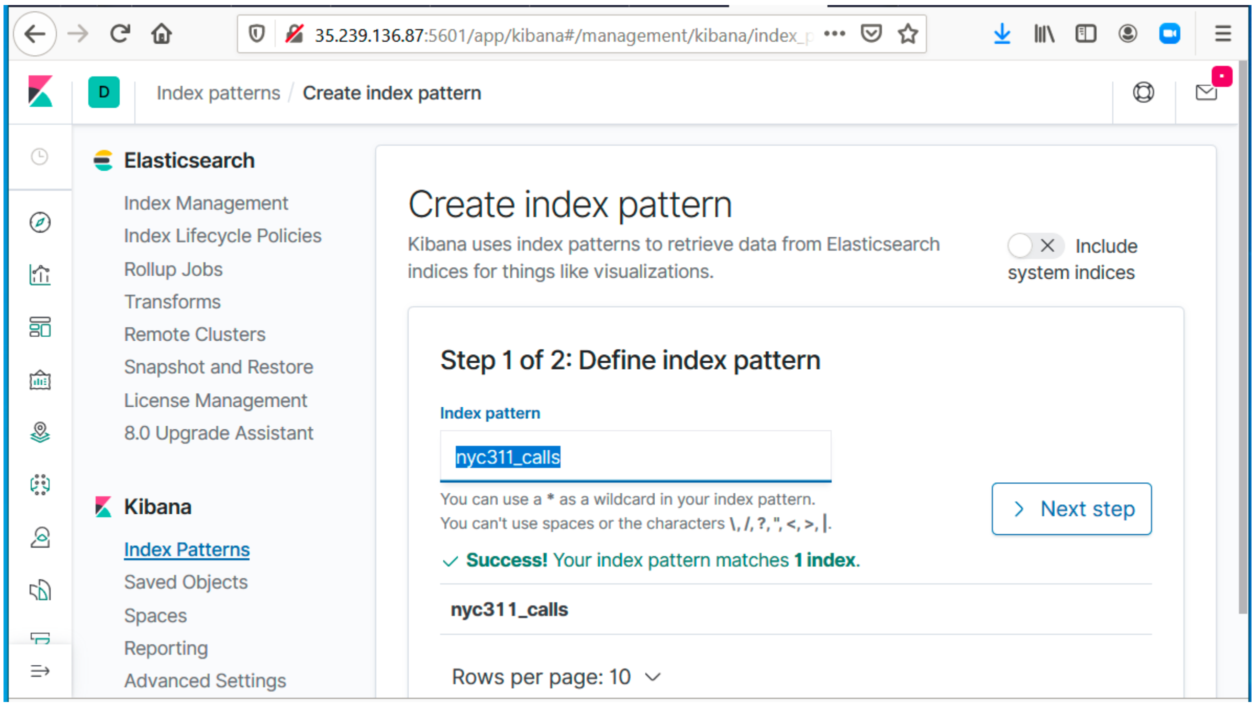
c) There are **2,048,987 events** (approximately 9.0%) without Location - set to [0,0] by code in config file:



5. Setup Index Pattern for Kibana Visualization

To set-up Index Pattern for our data analysis and visualization, we perform the following steps:

- Go to our **Kibana main** page at port 5601
- Select **Kibana Management** (the wheel icon)
- Select **Index Pattern** and create a new index pattern name "**nyc311_calls**", as shown in the screenshot below.



Data Analysis & Visualization

1. Data Table - Top 10 Cities alongside Top 10 Call Descriptors

Question 1: Create a table showing the top 10 cities with the highest calls alongside the count of top 10 complaint calls (by Descriptor) in each city.

Visualization 1 - Top 10 Cities with Highest # of calls:
In New Visualization tab, choose "**Data Table**" visualization and "**nyc311_calls**" as index, then:

- Selecting **Count** as **Metrics**
- Customer Label: **# of calls**
- Select **Buckets** Add, then **Split Rows**
- Select **Terms** as aggregation and **City.keyword** as field
- Customer Label: **Top 10 Cities with Highest # of calls**
- Order: **Descending** and Size: **10**

The Data Table of the top-10 cities with highest # of calls is shown below. **Brooklyn, New York and Bronx** are leading the top 3 of the top-10, with more than 4 million calls each and 3 million calls more than the other 7 top-10 cities.

Top 10 Cities with Highest # of calls	# of calls
BROOKLYN	6,823,188
NEW YORK	4,326,795
BRONX	4,045,532
STATEN ISLAND	1,130,043
JAMAICA	546,426
FLUSHING	407,317
ASTORIA	353,668
RIDGEWOOD	278,791
CORONA	177,638
WOODSIDE	171,903

Visualization 2 - Top 10 Cities with Highest # of calls along with Top 10 Call Descriptors
In New Visualization tab, choose "**Data Table**" visualization and "**nyc311_calls**" as index, then:

- Selecting **Count** as **Metrics**
- Customer Label: **# of calls**
- Select **Buckets** Add, then **Split Table**, then **Rows**
- Select **Terms** as aggregation and **City.keyword** as field
- Customer Label: **Top 10 Cities with Highest calls**
- Order: **Descending** and Size: **10**
- Select **Sub-Buckets** Add, then **Split Rows**
- Select **Terms** as aggregation and **Descriptor.keyword** as field
- Customer Label: **Top 10 Descriptors**
- Order: **Descending** and Size: **10**

1. Top 10 Descriptors for Brooklyn city:

BROOKLYN: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
Loud Music/Party	563,091
HEAT	275,255
No Access	266,321
ENTIRE BUILDING	244,179
Banging/Pounding	158,814
APARTMENT ONLY	152,062
Pothole	146,548
CEILING	123,967
Street Light Out	120,898
Request Large Bulky Item Collection	118,902

2. Top 10 Descriptors for New York city:

NEW YORK: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
Loud Music/Party	519,560
ENTIRE BUILDING	209,228
HEAT	190,614
Banging/Pounding	139,172
Noise: Construction Before/After Hours (NM1)	117,496
Driver Complaint	113,817
Loud Talking	108,120
N/A	101,608
APARTMENT ONLY	85,883
Pothole	85,277

3. Top 10 Descriptors for Bronx city:

BRONX: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
Loud Music/Party	406,490
ENTIRE BUILDING	287,847
HEAT	274,487
Banging/Pounding	141,740
APARTMENT ONLY	140,259
No Access	123,124
CEILING	112,843
MOLD	81,295
FLOOR	75,688
Pothole	72,540

4. Top 10 Descriptors for Staten Island city:

STATEN ISLAND: Top 10 Cities with Highest calls	
Top 10 Descriptors	**# of calls**
Pothole	80,449
Street Light Out	51,778
Loud Music/Party	47,481
With License Plate	25,572
Request Large Bulky Item Collection	25,535
1 Missed Collection	22,069
Recycling Electronics	22,016
No Access	18,619
Dirty Water (WE)	17,298
Sewer Backup (Use Comments) (SA)	17,121

5. Top 10 Descriptors for Jamaica city:

JAMAICA: Top 10 Cities with Highest calls	
Top 10 Descriptors	**# of calls**
Loud Music/Party	36,203
No Access	27,171
Street Light Out	16,417
Pothole	16,228
Sewer Backup (Use Comments) (SA)	14,462
14 Derelict Vehicles	13,483
HEAT	13,251
With License Plate	12,826
Partial Access	10,805
Banging/Pounding	10,678

6. Top 10 Descriptors for Flushing city:

FLUSHING: Top 10 Cities with Highest calls	
Top 10 Descriptors	**# of calls**
No Access	23,307
Pothole	16,473
Loud Music/Party	14,236
Street Light Out	13,028
Partial Access	11,414
ENTIRE BUILDING	11,095
Banging/Pounding	10,016
HEAT	9,662
Request Large Bulky Item Collection	9,362
Illegal Conversion Of Residential Building/Space	8,289

7. Top 10 Descriptors for Astoria city:

ASTORIA: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
Loud Music/Party	33,768
No Access	23,450
ENTIRE BUILDING	11,159
HEAT	10,021
Banging/Pounding	9,412
Pothole	8,340
Street Light Out	8,184
Request Large Bulky Item Collection	7,998
Partial Access	7,983
Loud Talking	6,703

8. Top 10 Descriptors for Ridgewood city:

RIDGEWOOD: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
Loud Music/Party	22,865
No Access	16,996
Request Large Bulky Item Collection	12,843
Blocked Hydrant	10,463
Street Light Out	6,616
HEAT	6,487
Pothole	6,332
ENTIRE BUILDING	6,322
With License Plate	5,517
Partial Access	5,373

9. Top 10 Descriptors for Corona city:

CORONA: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
No Access	25,050
Loud Music/Party	19,676
Partial Access	5,557
ENTIRE BUILDING	4,550
HEAT	4,366
Street Light Out	4,162
APARTMENT ONLY	3,902
Pothole	3,675
Banging/Pounding	3,411
14 Derelict Vehicles	2,889

10. Top 10 Descriptors for Woodside city:

WOODSIDE: Top 10 Cities with Highest calls

Top 10 Descriptors	# of calls
No Access	12,828
Loud Music/Party	12,153
ENTIRE BUILDING	6,852
Pothole	5,825
HEAT	5,042
Street Light Out	4,889
Partial Access	4,499
Banging/Pounding	3,533
Loud Talking	3,527
With License Plate	3,427

2. Pie Chart - Top 5 Cities alongside Top 5 Call Descriptors

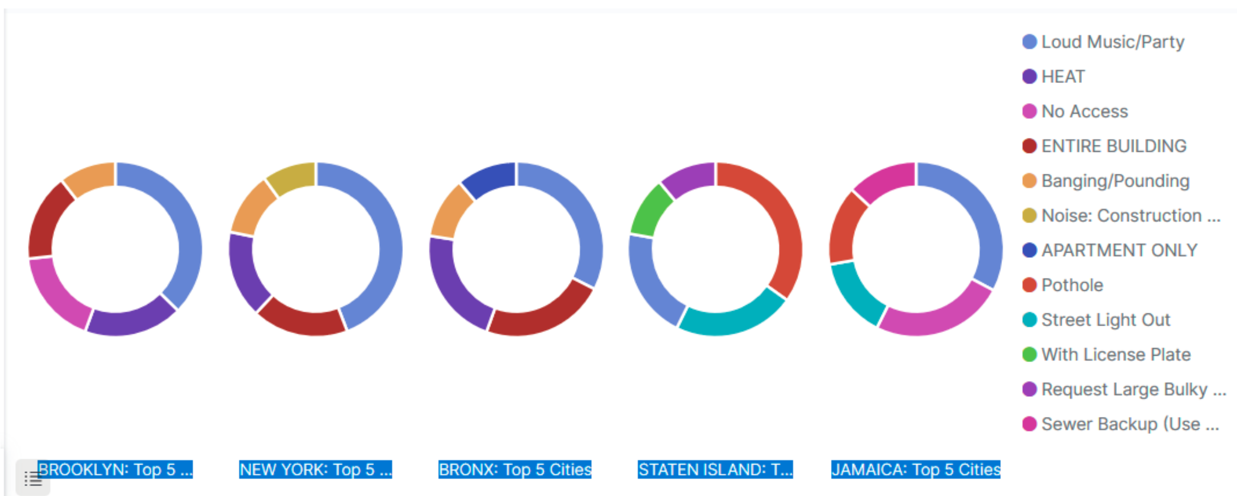
Question 2. Create a pie chart showing the top 5 cities with the highest calls alongside the top five calls (Descriptor) in each city.

Visualization:
In New Visualization tab, choose "**Pie**" visualization and "**nyc311_calls**" as index, then:

- Selecting **Slice size** as **Metrics**
- Select **Count** as aggregation
- Customer Label: **# of calls**
- Select **Buckets** Add, then **Split Chart**, then **Columns**
- Select **Terms** as aggregation and **City.keyword** as field
- Customer Label: **Top 5 Cities**
- Order: **Descending** and Size: **5**
- Select **Sub-Buckets** Add, then **Split Slices**, then **Rows**
- Select **Terms** as aggregation and **Descriptor.keyword** as field
- Customer Label: **Top 5 Descriptors**
- Order: **Descending** and Size: **5**

From the Pie Chart below, the Top 5 cities are **Brooklyn, New York, Bronx, Staten Island** and **Jamaica**.

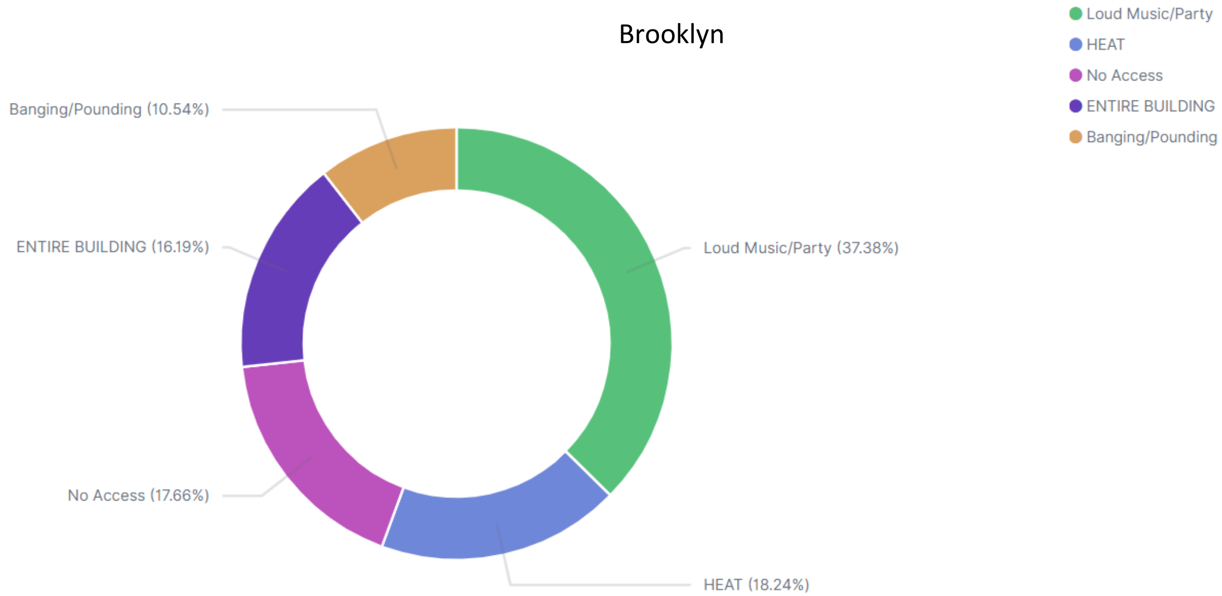
Brooklyn was the top one in the pie chart with **New York** and **Bronx** following behind.



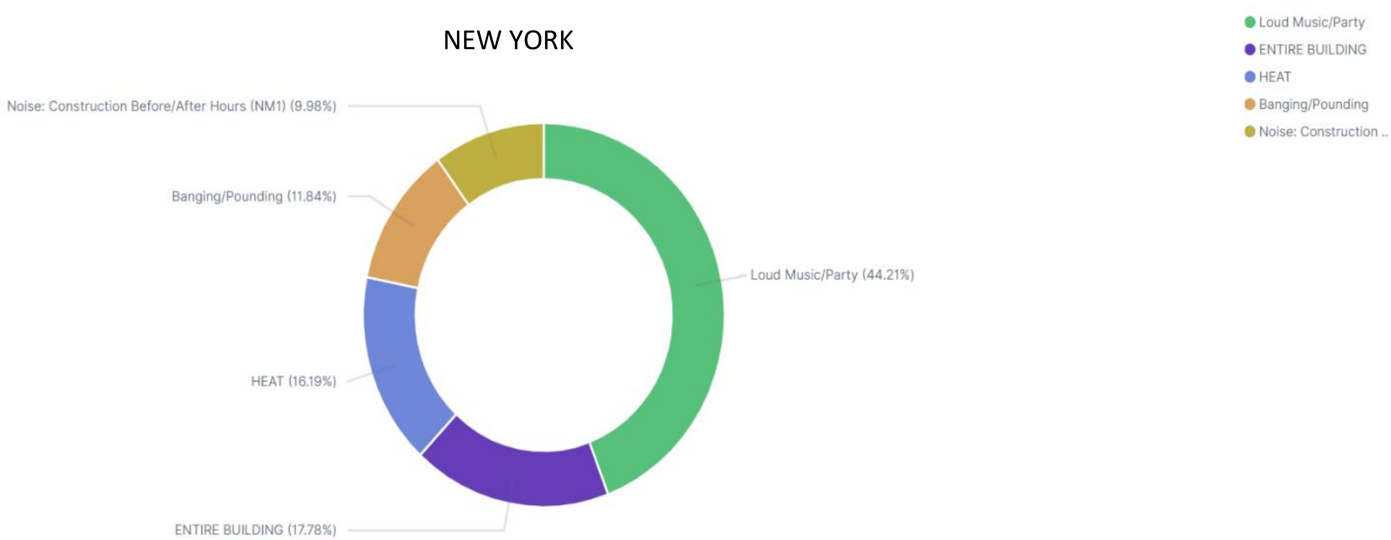
Among all top 5 cities, **Loud Music/Party** was the top cause of calling. **Heat, No Access, Entire Building, Banding/Pouding, Constructions Noise, Pothole** and **Street Light Out** were also major top call descriptors.

The **Top-5 Descriptors for each city** are shown in the detailed charts on the following pages.

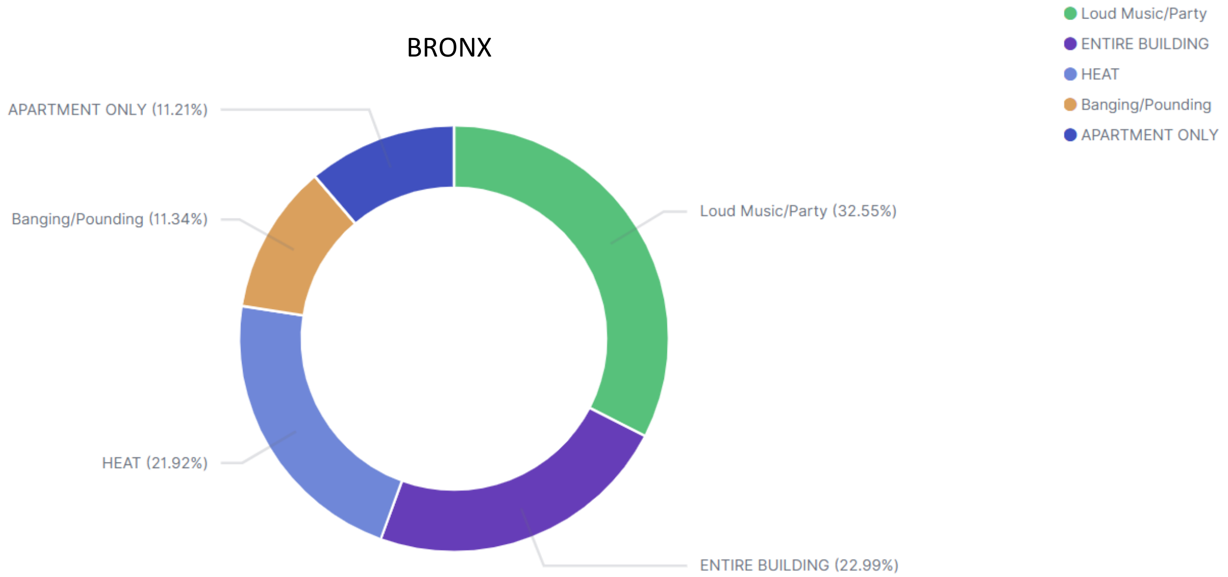
1. Top 5 Descriptors for Brooklyn city:



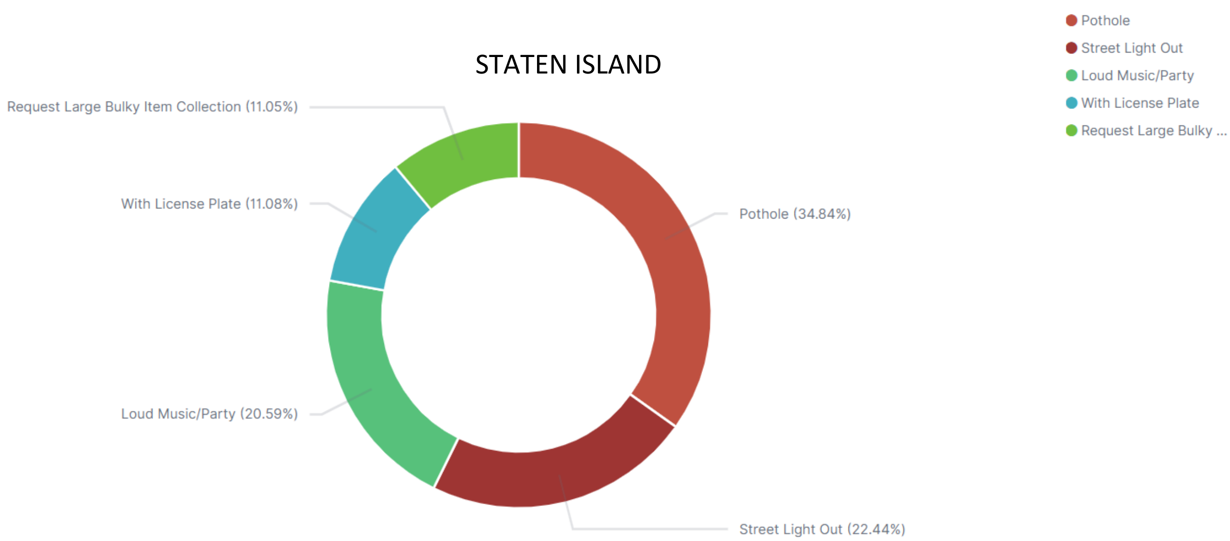
2. Top 5 Descriptors for New York city:



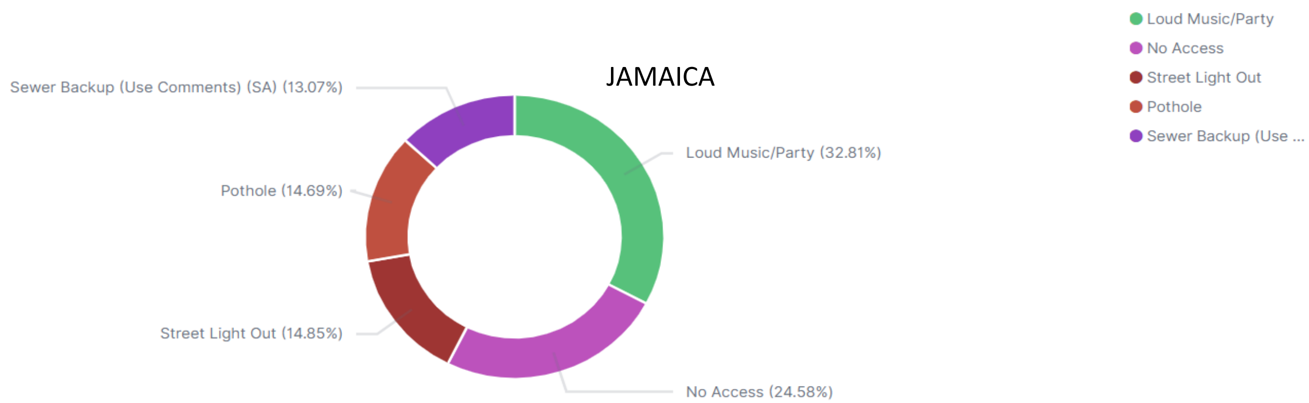
3. Top 5 Descriptors for Bronx city:



4. Top 5 Descriptors for Staten Island city:



5. Top 5 Descriptors for Staten Island city:



Observations:

From the Pie Charts above, we observe that **Loud Music/Party**, **No Access, Constructions Noise, Pothole** and **Street Light Out** were major calls and common among the Top 5 cities.

These cities can use this information to improve life quality and provide a better life for citizens and reduce the complaints.

3. Tag Cloud - Top 20 Call Descriptors

Question 3: Create a tag cloud representing the top 20 call descriptors.

For this question, we create 2 Tag Cloud charts:

- One Tag Cloud for top 20 **Call Descriptors**.
- One Tag Cloud for top 20 **Complaint Types.**

1. Tag Cloud for Top 20 Call Descriptors:

In New Visualization tab, choose "**Tag Cloud**" visualization and "**nyc311_calls**" as index, then:

- Selecting **Tag size** as **Metrics**
- Select **Count** as aggregation
- Customer Label: **# of calls**
- Select **Buckets** Add, then **Tags**
- Select **Terms** as aggregation and **Descriptor.keyword** as field
- Customer Label: **Top 20 Descriptors**
- Order: **Descending** and Size: **20**

Observations:

From the **Tag Cloud** chart above, **Loud Music/Party** was the top issue why people called 311. That is consistent with what the **Data Table** and the **Pie** charts above.

Entire Building, Heat, **No Access**, **Pothole, Street Light Out** and **Banging/Pounding** were also major calls.

2. Tag Cloud for Top 20 Complaint Types:

In New Visualization tab, choose "**Tag Cloud**" visualization and "**nyc311_calls**" as index, then:

- Selecting **Tag size** as **Metrics**
- Select **Count** as aggregation
- Customer Label: **# of calls**
- Select **Buckets** Add, then **Tags**
- Select **Terms** as aggregation and **Complaint Types.keyword** as field
- Customer Label: **Top 20 Complaint Types Tag Cloud**
- Order: **Descending** and Size: **20**

Observations:

From the above Top 20 Complaint Types Tag Cloud, **Resident Noise** was the top call, followed by **Heat/Hot Water**, **Illegal Parking, Blocked Driveway**, **Plumbing** and **Street Conditions**.

From both Tag Cloud charts, **Resident Noise** and **Loud Music** were the top issues why people called 311. This information could be useful for the New York city department to train their agents to deal with those issues and/or to speed up the process of dealing with those issues.

4. Coordinated Map - Major Call Descriptors in each city

Question 4: Create a coordinated map of all the major call descriptors in each city.

For this question, we create 2 Coordinated Maps:

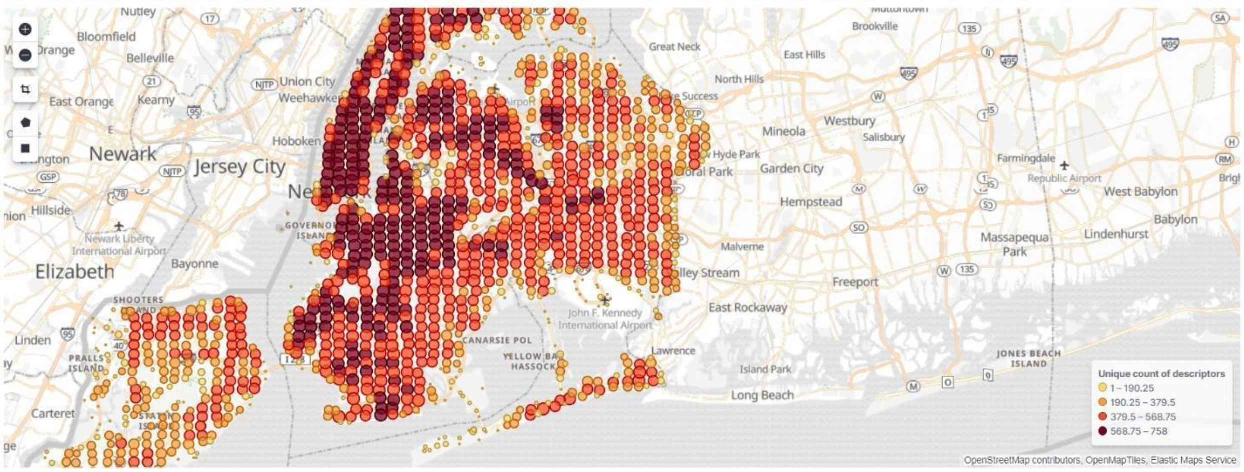
- One Coordinated Maps using **Unique count of call descriptors**.
- One Coordinated Maps using **Count (e.g. Numbers of calls).**

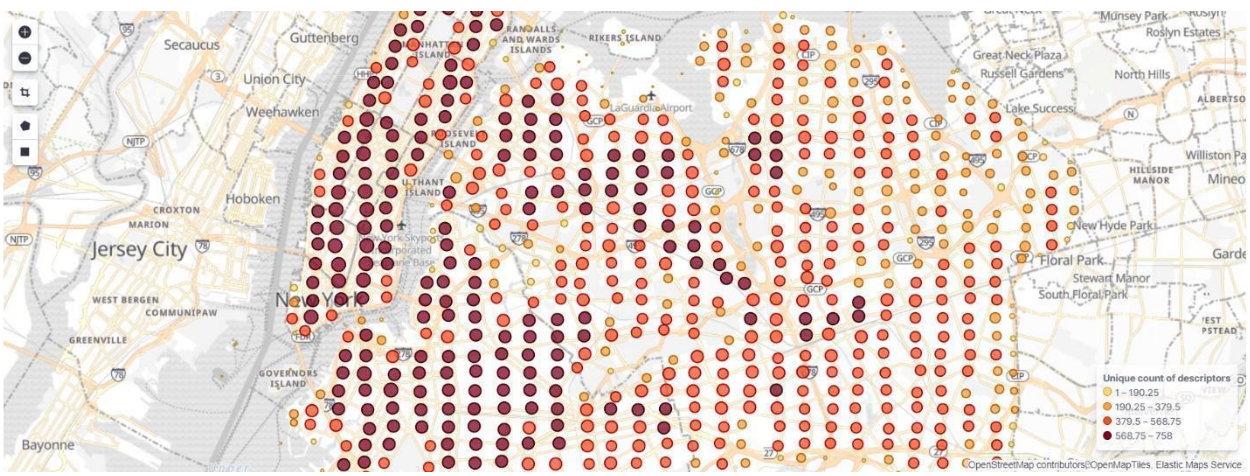
1. Coordinated Map using Unique Count of Call Descriptors:

In New Visualization tab, choose "**Coordinated Map**" visualization and "**nyc311_calls**" as index, then:

- Select **Value**, then **Unique Count** as aggregation, then **Descriptor.keyword** as field
- Customer Label: **Unique Count of Descriptors**
- Select **Buckets** Add, then **Geo Coordinates**
- Select **Geohash** as aggregation, then **Location** as Field
- Click **Apply Changes**, then **Enlarge New York area**

Using a unique count of descriptors and geohash, the coordinated map was presented below. New York city was picked for expansion to find out information as it was the second city with highest calls. New York city has denser color in round circles compared to other cities which implies that New York city has more descriptors types than other cities.



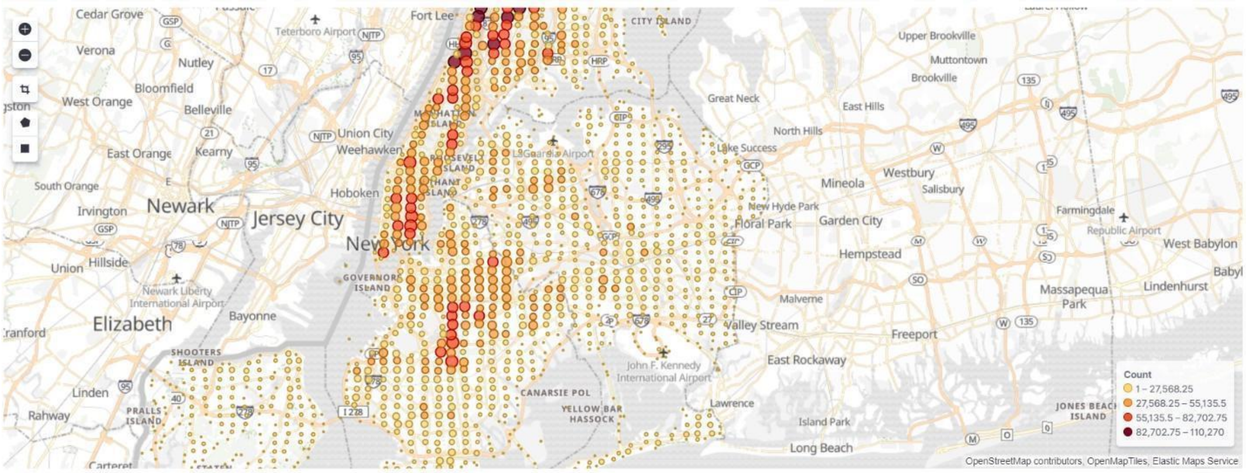


2. Coordinated Map using Count (e.g. Numbers of calls):

In New Visualization tab, choose “**Coordinated Map**” visualization and “**nyc311_calls**” as index, then:

- Select Value **Count** as **Metrics**
- Customer Label: **Unique Count of Major Descriptors**
- Select **Buckets** Add, then **Geo Coordinates**
- Select **Geohash** as aggregation, then **Location** as Field
- Click **Apply Changes**, then **Enlarge New York area**

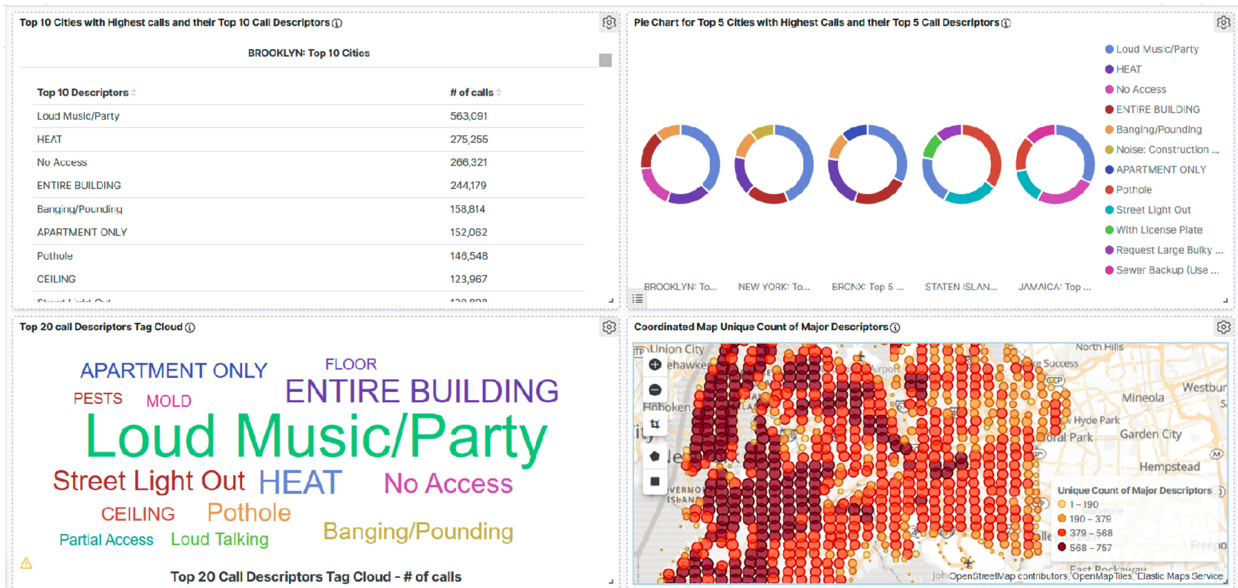
Below is the Coordinated Map based on numbers of call count, with expansion in the New York city area. We observe that most of the calls were concentrated more on the North of New York.



5. Create Dashboard from Kibana for visualizations from 1 to 4

Question 5: Create a dashboard for all visualizations of 1 to 4 above.

The Dashboard below includes the 4 visualizations for questions 1 to 4 above.

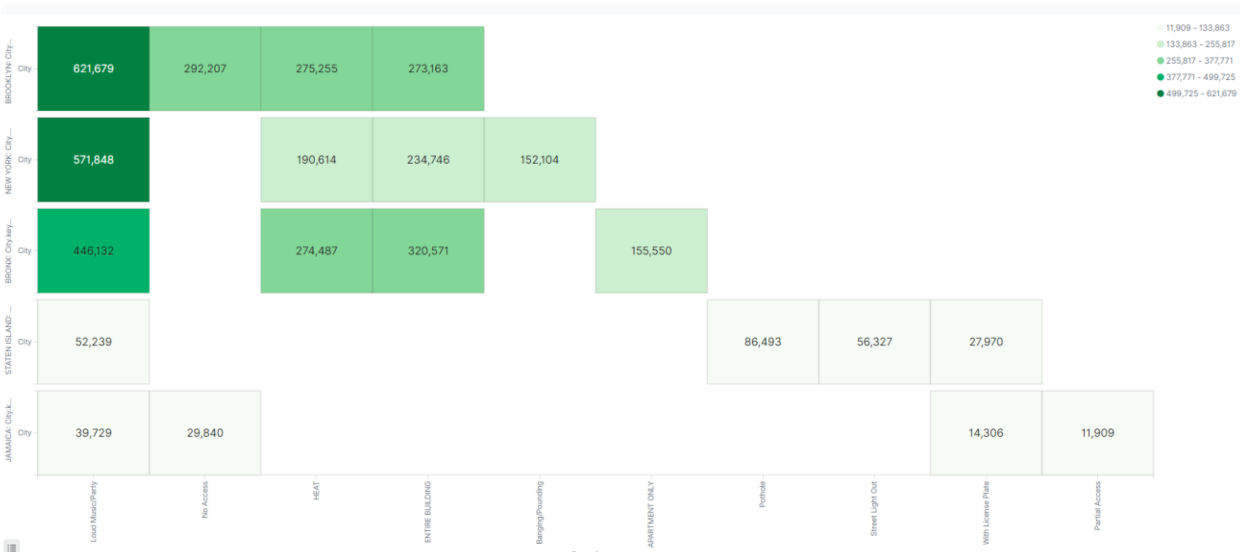


6. Heat Map - Top 5 Cities alongside Top 10 Descriptors

Visualization using Heat Map:

- Select "**Heat Map**" in New Visualization tab
- For **X-axis**, choose **Descriptor,** select **Top 10** in **Descending Order**
- For **Y-axis**, choose **City,** select **Top 5** in **Descending Order**

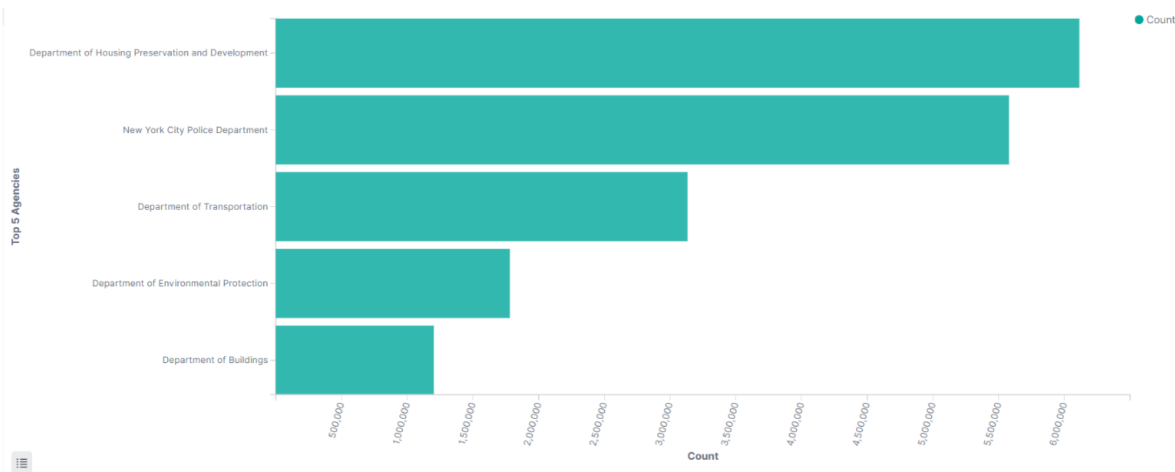
Below is the Heat Map of the Top 5 cities with the Top 10 descriptors. It shows the cities had many similar calls. All 5 cities had occurrences of **Loud Music/Party**. It also shows that **Brooklyn**, **New York** and **Bronx** are cities with the highest 311 calls. This is consistent with the results shown in visualizations for questions 1 and 2 above.



7. Lens, Gauge and Metric - Agency Performance Analysis

<u>Visualization using Lens</u>:

- Select “**Lens**” in New Visualization tab
- For **X-axis**, choose **Count**
- For **Y-axis**, choose **Agency Name**
- Select **Top 5** in **Descending Order**
- Select **Horizontal Bars**



From the chart above, the **top 5 agencies with highest # of calls** were:

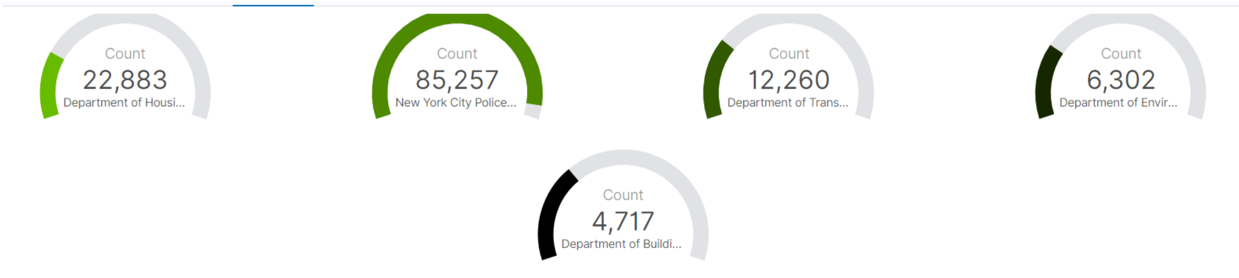
- Department of Housing Preservation and Development
- New York City Police Department
- Department of Transportation
- Department of Environment Protection, and
- Department of Building

The Department of Housing Preservation and Development and New York City Police Department were lead first 2 places with highest calls and with 3 millions more than other departments.

<u>Visualization using TSVB Gauge</u>:

- Select "**TSVB Gauge**" in New Visualization tab
- Choose **Count** as Metrics
- Select **Agency Name**
- Select **Top 5** in **Descending Order**
- Select **Gauge**

The TSVB Gauge chart below shows the **top 5 agencies together with # of calls**.



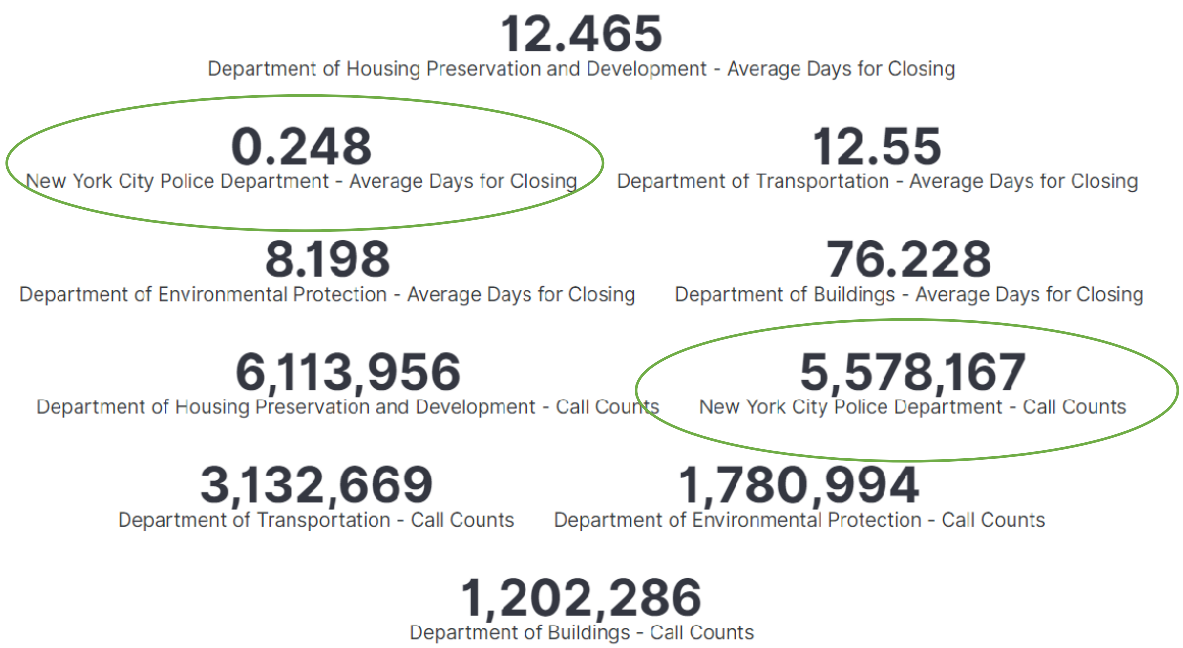
<u>Visualization using Metric - for overall average days of Closing</u>:

- Select "**Metric**" in New Visualization tab
- Choose **Average** in aggregation
- Choose **Duration** in field

The chart below shows the overall **average number of days** for call closing is 15 days.

Visualization using Metric - for average days of Closing for Top 5 Agencies:

- Select "**Metric**" in New Visualization tab
- Choose **Count** in aggregation
- Select **Agency Name** in field
- Select **Top 5** in **Descending Order**
- Add **Metric**
- Select **Average** in aggregation
- Select **Duration** in field
- Select **Order by Count**



Above are **top 5 agencies** with highest **call counts** and their **average days for closing**.

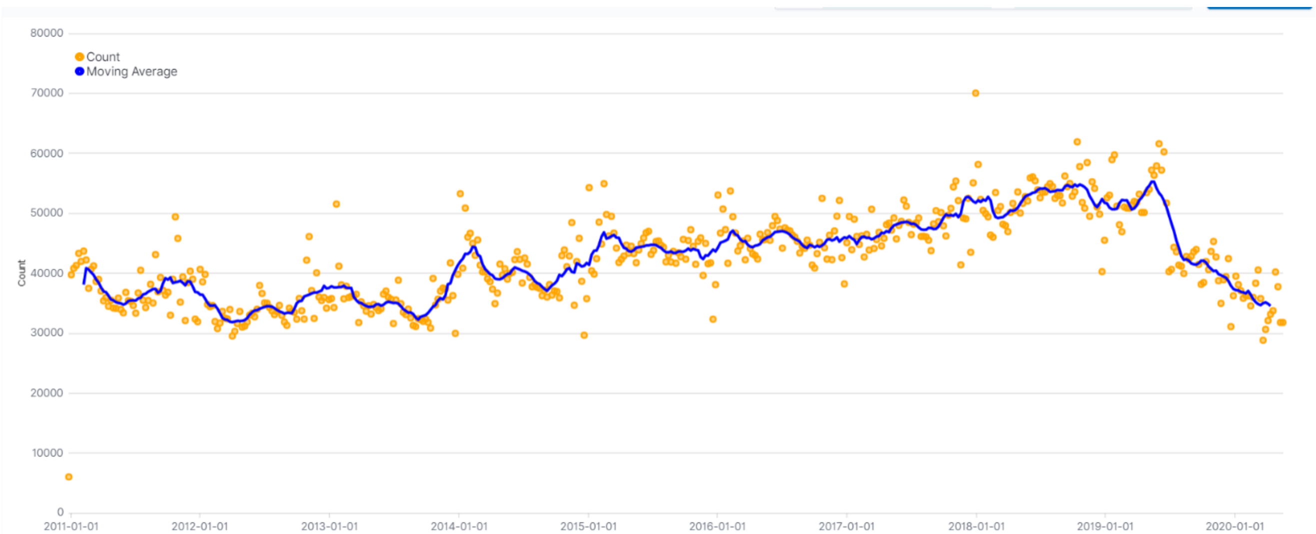
Observations:

- The **New York City Police Department** was the **most efficient** with lowest average days (**less than 1 day**) for closing and highest numbers of calls to deal with.
- The **Department of Environmental Protection** was the **2nd most efficient** with an average of **8 days** for closing.
- The **Department of Housing Preservation and Development** was with the highest calls, but their duration of closing the calls took **12 days** on average.
- The **Department of Transportation** was third on calls count, and also had the same **12 days** on average to close the accounts.
- The **Department of Building** took **76 days** for closing calls on average, which was far more than others in the top 5 cities with highest call counts.

8. Timelion - Time Series Analysis - Top 3 Cities and Calls Count vs. Year

Visualization using Timelion:

- Select "**Timelion**" in New Visualization tab
- Type in the **code below** for **Timelion Expression**
  .es(index=nyc311_calls, timefield="Created Date",
  metric=count).points(fill=5).color(orange).label('Count').yaxis(label="Count"),
  .es(index=nyc311_calls, timefield="Created Date",
  metric=count).movingaverage(12).lines().color(blue).label('Moving Average')
- **Filter** at the top for **Range between Year 2011 and 2020** (Berhane, 2020).



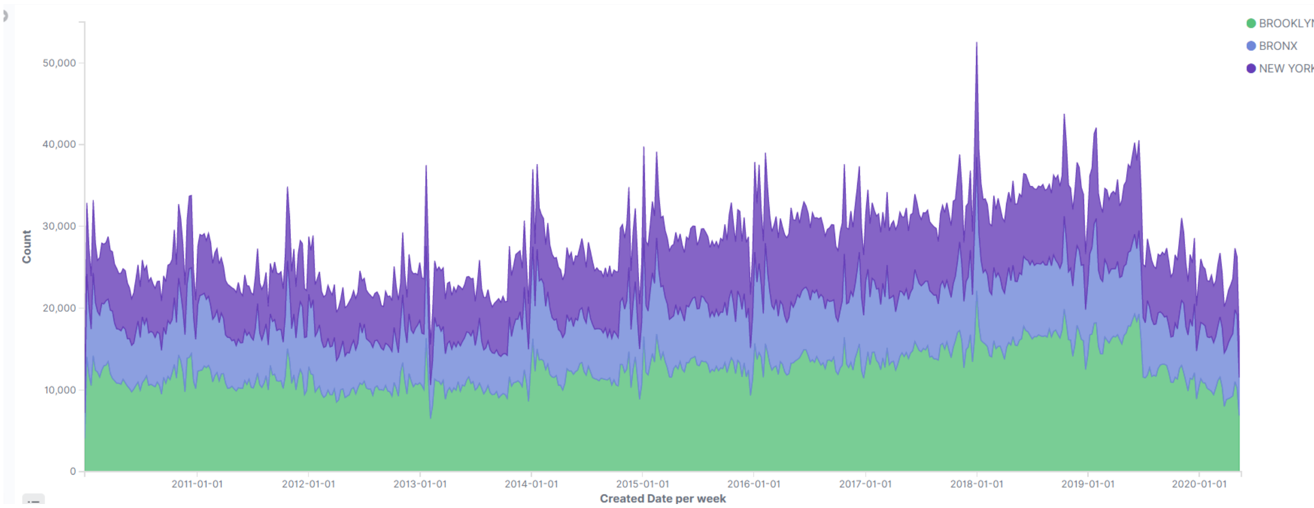
Above is the **Timelion chart** for the **call counts for all cities** during 2011 to 2020.

From the Timelion chart above, we observe that the **call counts** for all cities:

- **increased** between 2011 and 2019,
- **peaked in 2019**, then
- **dropped in 2020** - (this is because we do not have a full year of data for 2020)

<u>Visualization using Area Chart:</u>

- Select "**Area**" in New Visualization tab
- For **X-axis**, select **Created Date as Data Histogram** in aggregation
- Select **Split Series**, then **Term** as aggregation, then **City** as field
- Select **Top 3** in **Descending Order**



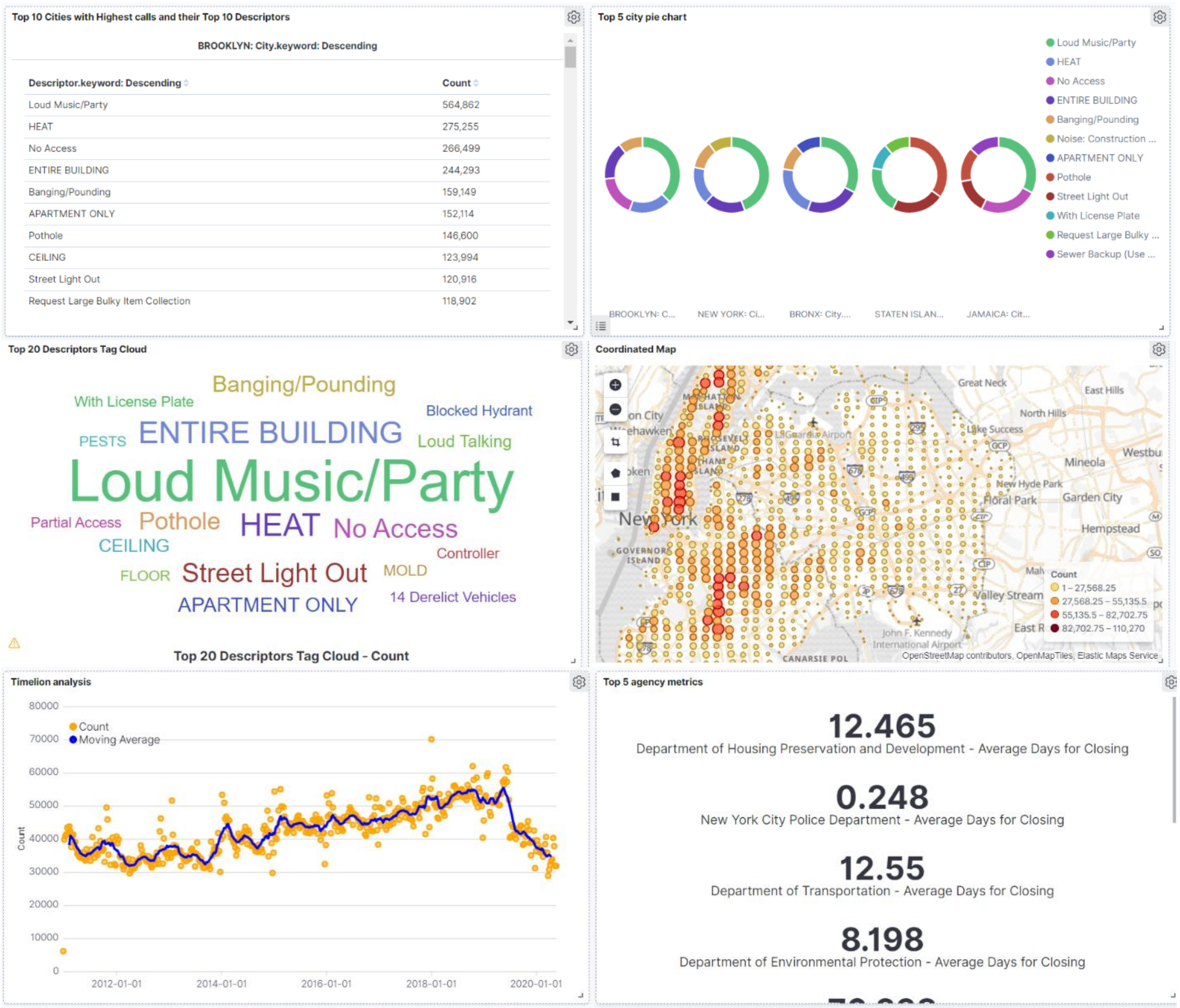
Above is the **Area chart** for the **call counts for the top 3 cities** during 2011 to 2020.

From the Area chart, we observe that:

- The **peak number of calls** usually **occurs in January** where the new year comes.
- The calls count slightly **increased with years** between 2011 and 2019, but **dropped in 2020**. This is consistent with the Timelion analysis above. (Note that the calls count drops in 2020 is because we do not have a full year data for 2020.)

9. Create Dashboard from Kibana with Additional Visualizations

The Dashboard below includes the 6 visualizations -- 4 from visualizations for questions 1 to 4 above, plus 2 additional visualizations from section 7 and 8 above.



Summary

1. From the **Data Table**, Brooklyn, New York and Bronx were the top 3 out of the top 10 and were at least 3 million more calls compared to the rest.

2. Based on **Pie charts**, the top 5 cities are Brooklyn, New York, Bronx, Staten Island and Jamaica. Among all top 5 cities, loud music and party was the major cause of calling. Heat, no access to enter the building, banding, noise and pothole were major top 5 call descriptors.

3. According to the **Tag Cloud**, loud music/party was the major issue why people were calling on 311 if descriptor was picked as term while noise from residents became the first one when the complaint type was selected as term.

4. From the **Coordinated Map**, most of the calls were concentrated more on the North of New York.

5. In **Agency Performance Analysis**, the top 5 agencies were department of building, department of environment protection, New York city police department, department of transportation and department of housing preservation and development. Among top 5, the New York city department was efficient with lowest days for closing and highest numbers of calls they dealt with.

6. From the **Timelion Analysis**, the peak for all call counts for all cities was reached at the beginning of each year and the calls slightly increased between 2011 and 2019.

7. **ELK stack** is a full text searching engine and searches for similar expressions. When checking for top 10 cities, Jamaica had two expressions, namely Jamaica and JAMAICA. ELK produced some inaccurate information if City Names were not capitalized before grouping.

8. **ELK stack** is great for unstructured data analysis especially JSON type. It was used heavily for blog or some online application logs analysis.

9. **Kibana visualization and dashboard features** are powerful and easy to use.

Appendix

Data Definition for NYC311 Dataset

Column Name	Description	Type
Unique Key	Unique identifier of a Service Request (SR) in the open data set	Plain Text
Created Date	Date SR was created	Date & Time
Closed Date	Date SR was closed by responding agency	Date & Time
Agency	Acronym of responding City Government Agency	Plain Text
Agency Name	Full Agency name of responding City Government Agency	Plain Text
Complaint Type	This is the first level of a hierarchy identifying the topic of the incident or condition. Complaint Type may have a corresponding Descriptor (below) or may stand alone.	Plain Text
Descriptor	This is associated to the Complaint Type, and provides further detail on the incident or condition. Descriptor values are dependent on the Complaint Type, and are not always required in SR.	Plain Text

Location Type	Describes the type of location used in the address information	Plain Text
Incident Zip	Incident location zip code, provided by geo validation.	Plain Text
Incident Address	House number of incident address provided by submitter.	Plain Text
Street Name	Street name of incident address provided by the submitter	Plain Text
Cross Street 1	First Cross street based on the geo validated incident location	Plain Text
Cross Street 2	Second Cross Street based on the geo validated incident location	Plain Text
Intersection Street 1	First intersecting street based on geo validated incident location	Plain Text
Intersection Street 2	Second intersecting street based on geo validated incident location	Plain Text
Address Type	Type of incident location information available.	Plain Text

City	City of the incident location provided by geovalidation.	Plain Text
Landmark	If the incident location is identified as a Landmark the name of the landmark will display here	Plain Text
Facility Type	If available, this field describes the type of city facility associated to the SR	Plain Text
Status	Status of SR submitted	Plain Text
Due Date	Date when the responding agency is expected to update the SR. This is based on the Complaint Type and internal Service Level Agreements (SLAs).	Date & Time
Resolution Description	Describes the last action taken on the SR by the responding agency. May describe next or future steps.	Plain Text
Resolution Action Updated Date	Date when the responding agency last updated the SR.	Date & Time
Community Board	Provided by geovalidation.	Plain Text
BBL	Borough Block and Lot, provided by geovalidation. Parcel number to identify the	Plain Text

	location of location of buildings and properties in NYC.	
Borough	Provided by the submitter and confirmed by geovalidation.	Plain Text
X Coordinate (State Plane)	Geo validated, X coordinate of the incident location.	Number
Y Coordinate (State Plane)	Geo validated, Y coordinate of the incident location.	Number
Open Data Channel Type	Indicates how the SR was submitted to 311. i.e. By Phone, Online, Mobile, Other or Unknown.	Plain Text
Park Facility Name	If the incident location is a Parks Dept facility, the Name of the facility will appear here	Plain Text
Park Borough	The borough of incident if it is a Parks Dept facility	Plain Text
Vehicle Type	If the incident is a taxi, this field describes the type of TLC vehicle.	Plain Text
Taxi Company Borough	If the incident is identified as a taxi, this field will display the borough of the taxi company.	Plain Text

Taxi Pick Up Location	If the incident is identified as a taxi, this field displays the taxi pick up location	Plain Text
Bridge Highway Name	If the incident is identified as a Bridge/Highway, the name will be displayed here.	Plain Text
Bridge Highway Direction	If the incident is identified as a Bridge/Highway, the direction where the issue took place would be displayed here.	Plain Text
Road Ramp	If the incident location was Bridge/Highway this column differentiates if the issue was on the Road or the Ramp.	Plain Text
Bridge Highway Segment	Additional information on the section of the Bridge/Highway where the incident took place.	Plain Text
Latitude	Geo based Lat of the incident location	Number
Longitude	Geo based Long of the incident location	Number
Location	Combination of the geo based lat & long of the incident location	Location