Integrating and Warehousing NYC Rodent Data to Improve Public Safety

CIS 9440 - Data Warehousing for Analytics
Section 17161
Professor Isaac Vaghefi, PhD

Group 5
Angela Lee - angela.lee4@baruchmail.cuny.edu
Derek Strang - derek.strang@baruchmail.cuny.edu
Komsit Rattana - komsit.rattana@baruchmail.cuny.edu
Mariya Mithaiwala - mariya.mithaiwala@baruchmail.cuny.edu

Table of Contents

Overview....1
Problem statement....1
Key Performance Indicators....1
Data Sources....2
Primary Data Source....2
Secondary Data Sources....2
Data Ingestion....4
Dimensional Model....4
NYC 311 Service Requests....6
DOHMH New York City Restaurant Inspection Results....8
2020 Census Tracts/Blocks Mapped....9
BigQuery Geography Functions....10
NYC Open Restaurant....10
ETL Process Development....11
The Big Picture....11
Product Selection....11
Build Pipelines using Google DataFlow....12
Populate Dimension and Fact Tables....22
Final Dimensional Schema....24
Final Dimensional Schema....25
KPI Visualizations....29
Visualization Tool....29
Conclusion....35
Software and database tools....35
Group Experience....35
References:....37
Appendix A - Mapping of Complaint Types....38

Overview

The rodent problem in New York City has persisted and gotten worse in recent years. Not only did it grab the attention of the citizens of the city, but also the mayor. Starting in 2023, the city agencies have invested $3.5 million to mitigate this problem with the pilot area in Harlem. In addition, Major Adams appointed the city's first Director of Rodent Mitigation to lead this initiative citywide.

In this regard, this proposal presents the project of creating an end-to-end data warehousing platform that analyzes the integrated data from NYC 311 rodent complaints data and complementary data such as restaurant inspection data, demographics, the boundaries of the NYC neighborhood, and zip code.

Through these integrations, the goal is to gain insights into the impact of these infestations on restaurant establishments across the city and to learn if there is a relationship between population density and the prevalence of infestations in particular areas.

Problem statement

New York City has faced persistent challenges in minimizing the severity of the rodent infestation, which has had a sizable impact on the public health of the city and its businesses.

By integrating rodent complaints data with either population density data or restaurant inspection data, we intend to address the following questions:

1. How does the frequency of rodent infestations correlate with population density in different neighborhoods across NYC?
2. What impact have rodent infestations had on businesses, as illustrated by complaint data and inspection data?
3. Can we identify areas where there needs to be more rodent control resources allocated to improve the public safety of the city and its businesses, particularly restaurants?
4. Has the Covid-19 pandemic contributed to the rise in rodent infestation?

Key Performance Indicators

1. Growth rate of complaints and violations related to rodent by time period (daily, weekly) and location (neighborhood, zip code)

2. Comparison of rodent complaints between residential and commercial buildings by location (neighborhood, zip code) and time period (e.g. lunch, dinner, late night)
3. Correlation between rodent complaints and other types e.g. trash, homeless encampment, etc.
4. Number of rodent complaints per location (neighborhood, zip code) over time
5. Average time to resolve rodent complaints by location over time
6. Total number of inspection violations by location over time (build widget to allow to drill down on borough, or neighborhoods)
7. Top 20 neighborhoods with most inspection violations
8. Build a heatmap that shows the neighborhoods that are most infected

Data Sources

Primary Data Source

NYC 311 Service Requests from 2010 to Present

Daily-updated service requests were made on NYC 311 for different complaint types. This project will focus on the complaint types of Rodent, Sanitation Enforcement, and Litter.

https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

Secondary Data Sources

DOHMH New York City Restaurant Inspection Results

The restaurant inspection results provide a daily-updated dataset of the violation codes, grades, and scores of the inspected restaurants as well as their addresses and geolocations. The violation codes can be resolved using the penalty schedule document from NYC Health.

- https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j
- https://www.nyc.gov/assets/doh/downloads/pdf/rii/ri-violation-penalty.pdf

NYC Open Restaurant

According to the increase of 71 percent in the rodent complaint since 2020, New York City released the Temporary Open Restaurants Program for restaurants to provide seating on sidewalks and roadway spaces to mitigate the impact from COVID-19.

This dataset contains the records of the program applications and their status. The data may be used to analyze and find relationships with rodent complaints.

https://data.cityofnewyork.us/Business/HISTORICAL-Sidewalk-Caf-Licenses-and-Applications/qcdj-rwhu

Geographical Boundaries of New York City

The geographical boundaries of New York City in the different levels of granularity can facilitate the analysis by transforming the geolocations of the complaints and the restaurant inspection results into categorical data and its hierarchy. The following datasets provide the boundaries from the largest to the lowest grains respectively.

NYC Borough Boundaries

This dataset contains the boundaries of five boroughs of New York City.

https://data.cityofnewyork.us/City-Government/Borough-Boundaries/tqmj-j8zm

NYC NTA map

The boundaries of New York City's neighborhoods are stored in this dataset.

https://data.cityofnewyork.us/City-Government/NTA-map/d3qk-pfyz

2020 Census Tracts - Mapped

Tracts are defined by the U.S. Census Bureau to identify the boundaries of small areas in the city or country.

- https://data.cityofnewyork.us/City-Government/2020-Census-Tracts-Mapped/weqx-t5xr
- https://data.cityofnewyork.us/City-Government/2020-Census-Tracts-Tabular/63ge-mke6

2020 Census Blocks - Mapped

Similar to Census Tracts, Census Blocks are the smallest geographical boundaries that the U.S. Census records its statistical data.

https://data.cityofnewyork.us/City-Government/2020-Census-Blocks-Mapped/sbvv-hpwv

Decennial Census data in 2020 geographies

The dataset consists of demographics data of the U.S. Census geographic units.
https://www.nyc.gov/site/planning/planning-level/nyc-population/2020-census.page

Data Ingestion

Dimensional Model

The dimensional model has been designed from the complaint data as the primary data source. The secondary data sources are transformed to create different dimensions and to find any possible relationship with the rodent issue.

Draft version of the model

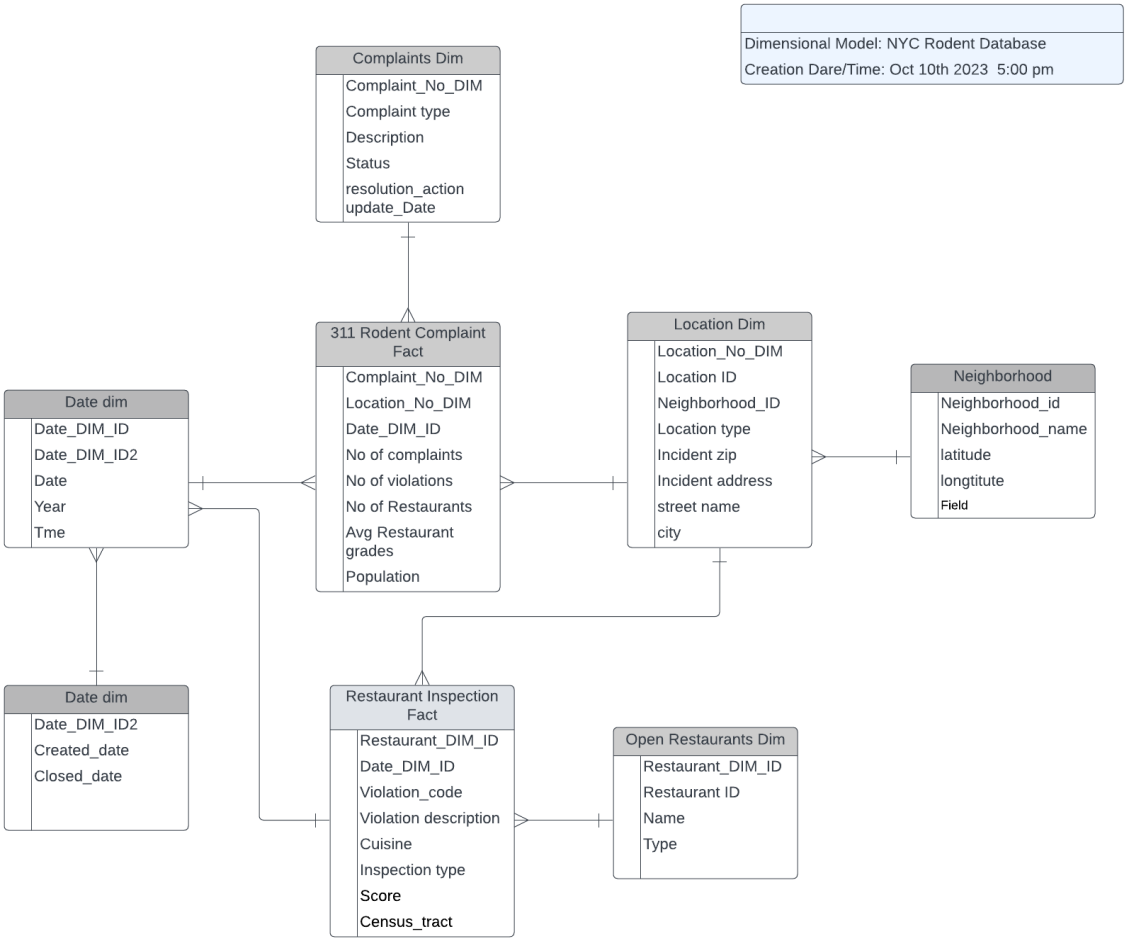


Figure 1 - The draft dimensional model

Revised edition of the model

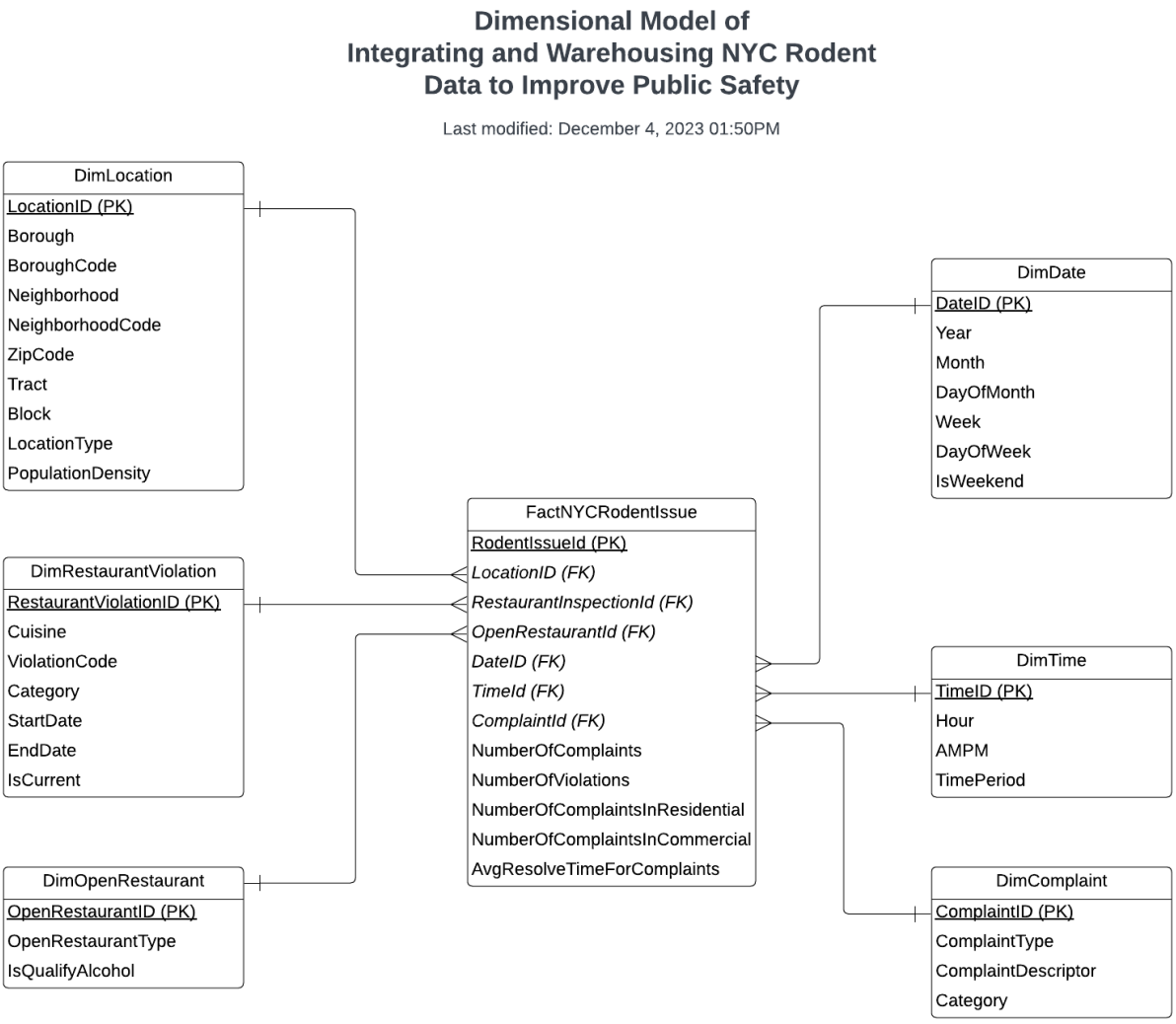


Figure 2 - The revised dimensional model

The dimensional model uses the grain of day that aligns with the daily incoming service requests and inspections. The model consists of one fact table and six dimension tables of location, restaurant violation, open restaurant seating, date, time, and complaint type.

There is an implementation of Slowly Changing Dimension (SCD) Type 2 in the restaurant violation table due to the possibility that categories of the violation codes may be changed over time based on the direction of the city administration.

NYC 311 Service Requests

There are records in 34,835,004 total of this dataset including all complaint types and the records submitted since 2010. Since the goal of this project is to analyze the rodent issue specifically, the irrelevant complaint types can be filtered out to improve the overall performance.

Data Profiling with the relevant complaint types

By filtering the service requests with the relevant complaint types and scope down the service requests created from 2016 - 3 years before the COVID crisis and the same amount of the duration to the present date, the dataset can be reduced to approximately 3 million records.

In addition, there are a number of complaint types that can be categorized into relevant categories including: Pests, Sewage, Animal Waste, Homelessness, and Dirty Conditions. This grouping can help reduce data points in the analysis. See Appendix A for the full mapping of complaint type, description, and category.

Complaint Type	**Total Records**
UNSANITARY CONDITION	658910
Rodent	274507
Sewer	247114
Homeless Person Assistance	240708
Missed Collection (All Materials)	230800
Sanitation Condition	208523
Dirty Conditions	201665
Missed Collection	139568
Dirty Condition	99422
Encampment	98844
Maintenance or Facility	89528
Food Establishment	78243
Illegal Dumping	66391
Obstruction	29124
Residential Disposal Complaint	22132

Complaint Type	Total Records
Homeless Encampment	21611
Dead Animal	21215
Animal in a Park	18302
Unsanitary Animal Pvt Property	16081
Street Sweeping Complaint	16041
Homeless Street Condition	15449
Mobile Food Vendor	14311
Highway Condition	9018
Overflowing Litter Baskets	8184
Indoor Sewage	7448
Litter Basket Request	7197
Unsanitary Condition	6750
Sanitation Worker or Vehicle Complaint	6551
Unsanitary Pigeon Condition	6125
Sweeping/Missed	5986
Commercial Disposal Complaint	5369
Lot Condition	4280
Dumpster Complaint	3846
Litter Basket Complaint	3310
Sewer Maintenance	2777
DPR Internal	1880
Sweeping/Inadequate	1577
Unsanitary Animal Facility	655
Municipal Parking Facility	308
Recycling Basket Complaint	158

Complaint Type	Total Records
Water Drainage	136
Overflowing Recycling Baskets	96
Transfer Station Complaint	25

Table 1 - Distribution of the focus complaint types in NYC 311 Service Requests

Considerations

- Whether there is a possibility to find relationships between the rodent problem and other complaint types
- Rat sightings increased 71 percent[1] from 2020 to 2023. What could be the possible trigger of the rodent problem in New York City?
    - Covid lockdown
    - Open restaurant program
    - Homeless
    - Pigeon (feeding)
    - Dumping
- There is a subset of the data focusing on rat sightings; however, using this dataset may stop the analysis from linking this problem with the others. https://data.cityofnewyork.us/Social-Services/Rat-Sightings/3q43-55fe
- The issues related to homelessness can be sensitive to be analyzed. Considerations are required during the analysis and conclusion.

Complaint Type *complaint_type*	count_complaint_type *count_complaint_type*
Homeless Encampment	38,208
Homeless Person Assistance	243,724
Homeless Street Condition	15,583

Figure 3 - The number of service requests related to homelessness

DOHMH New York City Restaurant Inspection Results

In the restaurant inspection data set, there is a column of restaurant violation codes. It requires a mapping to the description of the codes from the document from the NYC Health Department called Chapter 23 - Appendix 23-C Food service establishment and non-retail food services

[1] https://en.wikipedia.org/wiki/Rats_in_New_York_City

establishment penalty schedule[2]. The following relevant violation codes incorporate approximately 60,000 records.

Similar to the complaint type in 311 Service Requests, the restaurant violation code can be categorized into the same set of categories which helps link different datasets together.

Violation code	Description	Category
04F	Sewage and liquid waste not properly disposed of	Sewage
04K	Evidence of rats	Pests
04L	Evidence of mice	Pests
04M	Evidence of roaches	Pests
04N	Filth flies	Pests
04O	Live animal other than fish in tank or service animal	Animal Waste
05A	Sewage disposal system improper or unapproved	Sewage
08A	• Prevention and control measures not used for pest management • Door openings into the establishment from the outside not properly equipped • Pest monitors incorrectly used • Contract with pest exterminator or record of pest extermination activities not kept on premises	Pests
08B	• Garbage not properly removed or stored • Garbage receptacles and covers not cleaned after emptying and prior to reuse	Dirty Conditions
08C	Pesticides not properly labeled, not authorized for use, or improperly used	Pests
10B	• Potable water not protected from backflow, back siphonage or cross-connection • Improper disposal of sewage or liquid waste • Condensation pipes not properly installed or maintained	Sewage

Table 2 - Violation codes of interest for this project

2020 Census Tracts/Blocks Mapped

To view the rodent problem and solve it efficiently, analyzing the data down to the tract and block levels of New York City which provide better granularity than the zip codes, the city officials prepare and resolve the issue faster and directly.

[2] https://www.nyc.gov/assets/doh/downloads/pdf/rii/ri-violation-penalty.pdf

Incorporate Population Statistics

The population of each NYC Block is provided in the 2020 Census Data-census blocks on the NYC Planning website. The data consists of demographics of each block including total population, population by gender, ethnicity, age, and the area size of the block. For this analysis, only the total population and the area size are used to calculate the density.

The population data can be mapped to each Block using its BCTCB2020 code.

BigQuery Geography Functions

Google BigQuery supports geography functions[3] that can parse the geographical boundaries directly from the dataset. This function facilitates the analysis and reduces effort from developing additional transformations.

NYC Open Restaurant

New York City developed the Open Restaurant program to help the restaurant business cope with the COVID-19 restrictions on indoor dining. This program happened at the same time as the rising issue of the rodent problem. This project includes this dataset to analyze whether the rodent issue is relevant to this program.

The limitation of this dataset is that it is the historical data of the restaurants signing up for this program up to 2020. However, it is clearly seen that there are still remaining outdoor patios and sidewalk dining in the city to date. The project puts the assumption that this data is still relevant.

[3] https://cloud.google.com/bigquery/docs/reference/standard-sql/geography_functions

ETL Process Development

The Big Picture

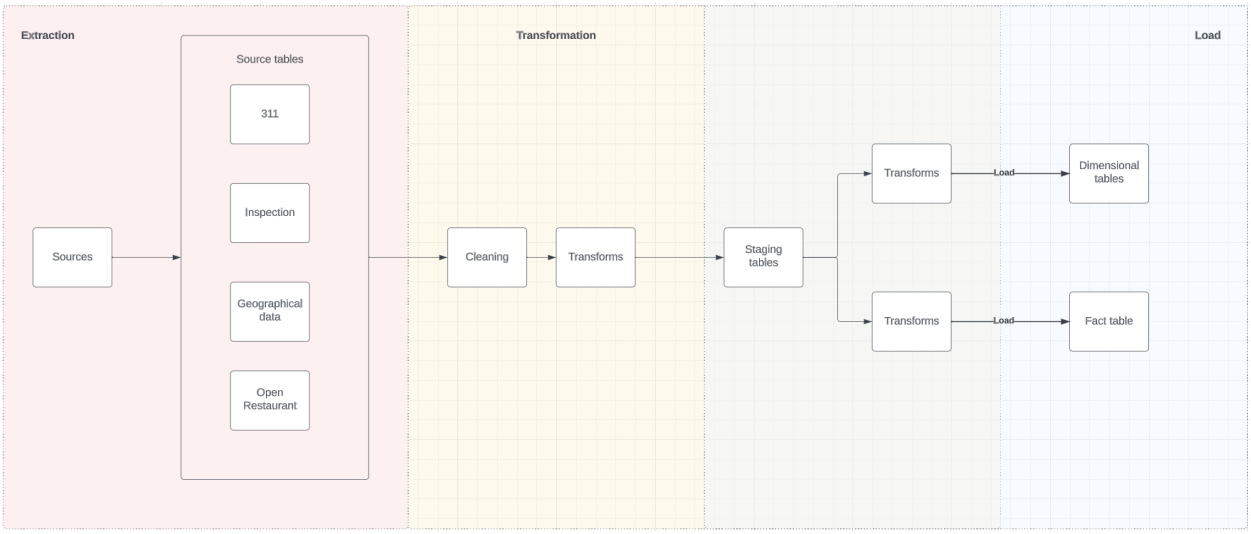


Figure 4 - The high-level view of the ETL process

The ETL process is divided into three stages:

- **Extraction**
  Extract the data from the sources into the intermediary tables in Google BigQuery for further analysis
- **Transformation**
  Transform the data into the desired format and prepare to load the data into the dimension and fact tables
- **Load**
  This stage overlaps the transformation stage. Once the transformation is complete, the final data is pushed and persistently stored in the project's dimensional model.

Product Selection

Data Repositories

REST API Connections

The ETL tool can connect to the REST APIs with JSON data for NYC 311 Service Requests, Restaurant Inspection Results, and NYC Open Restaurant (historical). It allows incremental data extraction and reduces processing time.

Google BigQuery

The data will be loaded into Google BigQuery for profiling, analysis, and storage for visualization.

CSV Files

Geographical boundaries of NYC are provided in CSV files and rarely changed. The ETL tool can process the CSV files directly.

ETL Tools

Google Dataflow

Google Dataflow provides seamless integration with Google Cloud services, and the team can benefit from collaboration on the cloud platform. The main challenge of this tool is that it has a steep learning curve for its underlying technology, Apache Beam.

Google BigQuery Scheduled Queries

Google BigQuery has functionalities to schedule queries, which is similar to the Google Dataflow pipeline. It can be used as an alternative automation solution for simple data tasks rather than setting up the complex pipeline using Google Dataflow.

Python Script

The only purpose of using Python scripts in the ETL process for this project is that some data sources e.g. NYC Block Boundaries, and Open Restaurant Application, contain malformed or invalid data format after exporting the data in the CSV format. These data sources have one thing in common that is the data is rarely changed over time, for example, the U.S. Census data is only changed once every 10 years.

Hence, the data is required to be exported in the JSON format or needs fixing before loading into the database. The Python scripts provide this flexibility to solve this data issue and are manually executed to populate the data into the database.

Build Pipelines using Google DataFlow

This project uses three Google Dataflow pipelines to process the long-running or complex data tasks as follows:

1. Load snapshot or incremental data of NYC 311 Service Requests
2. Load snapshot or incremental data of NYC Restaurant Inspection
3. Transform the location of NYC 311 Service Requests, Restaurant Inspection, and Open Restaurant to populate the location dimension table

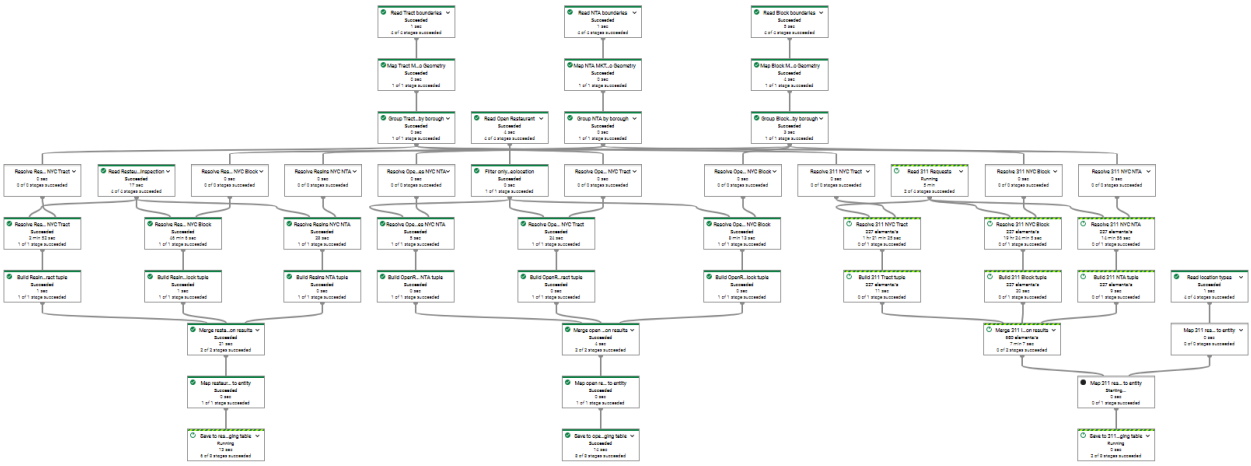


Figure 5 - The steps in the location staging data pipeline

The steps to develop the ETL process using Google DataFlow Pipelines are described as follows:

1. Follow the instructions on Create a Dataflow pipeline using Python to prepare the local machine to connect to Google Cloud and install dependencies for Apache Beam.
2. Develop pipelines using Python on Visual Studio Code

```
        with beam.Pipeline(options=beam_options) as p:
            (p
            | 'Load focus complaint types' >> beam.io.ReadFromBigQuery(
                table=f'{self.GCLOUD_BIGQUERY_PROJECT_ID}:{self.GCLOUD_BIGQUERY_DATASET}.focus_complaint_type',
                method=beam.io.ReadFromBigQuery.Method.DIRECT_READ)
            | 'Extract complaint type value' >> beam.Map(lambda lookup: lookup['complaint_type'])
            | 'Generate NYC Open Data API pagination URLs' >> beam.ParDo(GenerateNYCOpenDataAPIUrls(
                {'X-App-Token': self.NYC_OPENDATA_API_KEY, 'Accept': 'application/json'},
                self.dataStartDate,
                self.dataEndDate,
                1000))
            | 'Flat-map API URL' >> beam.FlatMap(lambda urls: urls)
            | 'Request NYC Open Data API' >> beam.ParDo(CallNYCOpenDataAPI(
                {'X-App-Token': self.NYC_OPENDATA_API_KEY, 'Accept': 'application/json'}))
            | 'Flat-map API response' >> beam.FlatMap(lambda records: records)
            | 'Map API response to entity' >> beam.ParDo(MapAPIResponseToBigQueryRecord(DATE_TIME_FORMAT))
            | 'Save to BigQuery' >> beam.io.WriteToBigQuery(
                dest_table_spec,
                schema=TARGET_TABLE_SCHEMA,
                write_disposition=self.destWriteDisposition,
                create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED)
            )

if __name__ == "__main__":
    logging.getLogger().setLevel(logging.INFO)
    program = Program()
    program.main()
```

Figure 6 - The 311 Service Request pipeline's main function

3. A transformation is required to map the API response from NYC 311 data API to the schema in Google BigQuery.
    a. get_string_value()
    Retrieves a string value from the record, handling cases where the value may be None.
    b. get_date_value()
    Parses date values using datetime.strptime.
    c. get_float_value()
    Attempts to parse float values, handling cases where the value may not be a valid float.

```
def process(self, record):
    out = {
        'unique_key': self.get_string_value(record, 'unique_key'),
        'created_date': self.get_date_value(record, 'created_date'),
        'closed_date': self.get_date_value(record, 'closed_date'),
        'complaint_type': self.get_string_value(record, 'complaint_type'),
        'descriptor': self.get_string_value(record, 'descriptor'),
        'location_type': self.get_string_value(record, 'location_type'),
        'incident_zip': self.get_string_value(record, 'incident_zip'),
        'incident_address': self.get_string_value(record, 'incident_address'),
        'street_name': self.get_string_value(record, 'street_name'),
        'cross_street_1': self.get_string_value(record, 'cross_street_1'),
        'cross_street_2': self.get_string_value(record, 'cross_street_2'),
        'intersection_street_1': self.get_string_value(record, 'intersection_street_1'),
        'intersection_street_2': self.get_string_value(record, 'intersection_street_2'),
        'address_type': self.get_string_value(record, 'address_type'),
        'city': self.get_string_value(record, 'city'),
        'landmark': self.get_string_value(record, 'landmark'),
        'facility_type': self.get_string_value(record, 'facility_type'),
        'status': self.get_string_value(record, 'status'),
        'due_date': self.get_date_value(record, 'due_date'),
        'resolution_description': self.get_string_value(record, 'resolution_description'),
        'resolution_action_updated_date': self.get_date_value(record, 'resolution_action_updated_date'),
        'community_board': self.get_string_value(record, 'community_board'),
        'bbl': self.get_string_value(record, 'bbl'),
        'borough': self.get_string_value(record, 'borough'),
        'x_coordinate_state_plane': self.get_float_value(record, 'x_coordinate_state_plane'),
        'y_coordinate_state_plane': self.get_float_value(record, 'y_coordinate_state_plane'),
        'open_data_channel_type': self.get_string_value(record, 'open_data_channel_type'),
        'park_facility_name': self.get_string_value(record, 'park_facility_name'),
        'park_borough': self.get_string_value(record, 'park_borough'),
        'latitude': self.get_float_value(record, 'latitude'),
        'longitude': self.get_float_value(record, 'longitude'),
        'compute_zip': self.get_string_value(record, ':@computed_region_efsh_h5xi'),
        'compute_borough_boundaries': self.get_string_value(record, ':@computed_region_yeji_bk3q')
    }

    return [out]
```

Figure 7 - The data transformation step from API to BigQuery record

4. Another transformation with similar methods is also required to load the restaurant inspection data into the Google BigQuery table.

```
def process(self, record):
    out = {
        'camis': self.get_string_value(record, 'camis'),
        'dba': self.get_string_value(record, 'dba'),
        'boro': self.get_string_value(record, 'boro'),
        'building': self.get_string_value(record, 'building'),
        'street': self.get_string_value(record, 'street'),
        'zipcode': self.get_string_value(record, 'zipcode'),
        'phone': self.get_string_value(record, 'phone'),
        'cuisine_description': self.get_string_value(record, 'cuisine_description'),
        'inspection_date': self.get_date_value(record, 'inspection_date'),
        'action': self.get_string_value(record, 'action'),
        'violation_code': self.get_string_value(record, 'violation_code'),
        'violation_description': self.get_string_value(record, 'violation_description'),
        'critical_flag': self.get_string_value(record, 'critical_flag'),
        'score': self.get_float_value(record, 'score'),
        'grade': self.get_string_value(record, 'grade'),
        'grade_date': self.get_date_value(record, 'grade_date'),
        'record_date': self.get_date_value(record, 'record_date'),
        'inspection_type': self.get_string_value(record, 'inspection_type'),
        'latitude': self.get_float_value(record, 'latitude'),
        'longitude': self.get_float_value(record, 'longitude'),
        'community_board': self.get_string_value(record, 'community_board'),
        'council_district': self.get_string_value(record, 'council_district'),
        'census_tract': self.get_string_value(record, 'census_tract'),
        'bin': self.get_string_value(record, 'bin'),
        'bbl': self.get_string_value(record, 'bbl'),
        'nta': self.get_string_value(record, 'nta')
    }

    return [out]
```

Figure 8 - The data transformation step of NYC Restaurant Inspection data

5. Test the pipelines on the local machine using Python Shell command
6. Create Google Cloud resources for hosting Google DataFlow pipelines
    a. Cloud Storage for storing template files and keeping log and staging data during the pipeline executions

Name	Created	Location type	Location	Default storage class	Last modified
cis_9440_geographical_data	Nov 24, 2023, 1:56:54 PM	Multi-region	us	Standard	Nov 24, 2023, 1:56:54 PM
dataflow-staging-us-east4-10613008834...	Nov 24, 2023, 1:30:28 AM	Region	us-east4	Standard	Nov 24, 2023, 1:30:28 AM
avid-garage-399623_cloudbuild	Nov 24, 2023, 1:12:26 AM	Multi-region	us	Standard	Nov 24, 2023, 1:12:26 AM
cis9440-kr-dataflow	Nov 21, 2023, 2:20:40 AM	Multi-region	us	Standard	Nov 21, 2023, 2:20:40 AM

Figure 9 - Google Cloud Storage for the pipelines

b. Artifact Repository for storing docker image contained the pipeline code and its dependencies

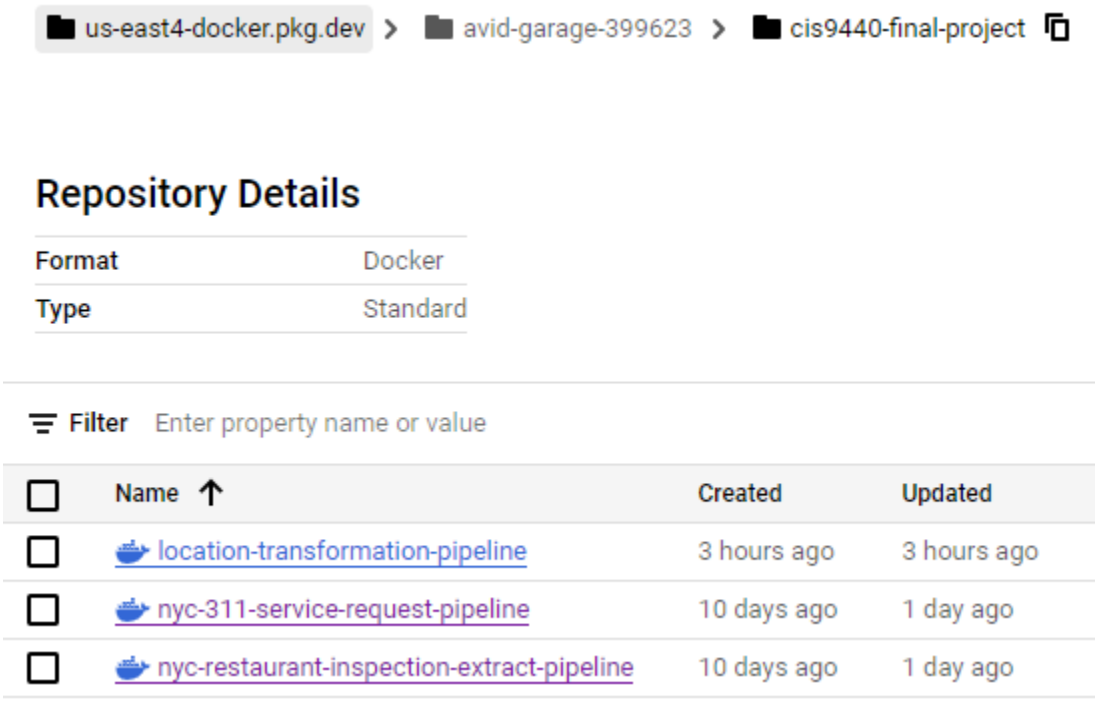


Figure 10 - Pipeline docker image on Artifact Registry

7. Build Google DataFlow Template using this command:

```
gcloud dataflow flex-template build
gs://{STORAGE_NAME}/dataflow/templates/{TEMPLATE_NAME}.json
--image-gcr-path
"us-east4-docker.pkg.dev/{PROJECT_ID}/{ARTIFACT_REGISTRY_NAME}/
{DOCKER_IMAGE_NAME}:latest" --sdk-language "PYTHON"
--flex-template-base-image
"gcr.io/dataflow-templates-base/python39-template-launcher-base
" --metadata-file "metadata.json" --py-path "." --env
"FLEX_TEMPLATE_PYTHON_PY_FILE={PIPELINE_PYTON_FILENAME}.py"
```

```
--env "FLEX_TEMPLATE_PYTHON_REQUIREMENTS_FILE=requirements.txt"
```

8. Create Google DataFlow Pipeline
    a. Create a new pipeline from the custom template built in the step earlier
    b. Set the schedule
        i. Develop a snapshot pipeline to initialize the table.Google DataFlow Pipeline lacks support for manual execution, thereby removing the option for automatic scheduling.
        ii. Implement an incremental pipeline designed to append data daily at 12:30 AM (New York local time), loading information from the preceding day. This pipeline is configured to execute on a daily basis, automatically incorporating new data into the existing dataset.

Name	Status	Type	Region	Total Runs	Schedule
CIS 9440 Prepare Location Staging Data Daily	Active	Batch	us-east4	0	daily at 12:30 AM America/New_York
CIS 9440 Prepare Location Staging Data Snapshot	Active	Batch	us-east4	1	in Nov on day 24 at 1:35 AM America/New_York
CIS 9440 NYC Restaurant Inspection Daily	Active	Batch	us-east4	10	daily at 12:30 AM America/New_York
CIS 9440 NYC Restaurant Inspection Snapshot	Active	Batch	us-east4	4	in Nov on day 24 at 1:35 AM America/New_York
CIS 9440 NYC 311 Service Request Daily	Active	Batch	us-east4	10	daily at 12:30 AM America/New_York
CIS 9440 NYC 311 Service Request Snapshot	Active	Batch	us-east4	9	in Nov on day 24 at 1:35 AM America/New_York

Figure 11 - Snapshot and incremental pipelines

9. Run the snapshot pipelines to initialize the data in the repository

Name	Type	End time	Elapsed time	Start time	Status
cis-9440-prepare-location-s-mp--1701710960-6457532215576832526	Batch		3 hr 30 min	Dec 4, 2023, 12:29:23 PM	Running
cis-9440-nyc-311-service-re-mp--1701667801-6905922735693078085	Batch	Dec 4, 2023, 12:36:58 AM	6 min 57 sec	Dec 4, 2023, 12:30:01 AM	Succeeded
cis-9440-nyc-restaurant-ins-mp--1701667800-6827526071654145993	Batch	Dec 4, 2023, 12:36:49 AM	6 min 49 sec	Dec 4, 2023, 12:30:00 AM	Succeeded
cis-9440-nyc-311-service-re-mp--1701614917-1635185139181190747	Batch	Dec 3, 2023, 10:29:13 AM	40 min 35 sec	Dec 3, 2023, 9:48:38 AM	Succeeded
cis-9440-nyc-restaurant-ins-mp--1701585338-581554585465770149	Batch	Dec 3, 2023, 1:42:51 AM	7 min 12 sec	Dec 3, 2023, 1:35:39 AM	Succeeded
cis-9440-nyc-311-service-re-mp--1701581400-4989902846123325889	Batch	Dec 3, 2023, 12:36:54 AM	6 min 54 sec	Dec 3, 2023, 12:30:00 AM	Succeeded
cis-9440-nyc-restaurant-ins-mp--1701581400-13277421283220621694	Batch	Dec 3, 2023, 12:36:33 AM	6 min 32 sec	Dec 3, 2023, 12:30:01 AM	Succeeded
cis-9440-nyc-311-service-re-mp--1701495000-14652244791655035645	Batch	Dec 2, 2023, 12:37:01 AM	7 min	Dec 2, 2023, 12:30:01 AM	Succeeded
cis-9440-nyc-restaurant-ins-mp--1701495000-1855089578742435368	Batch	Dec 2, 2023, 12:36:32 AM	6 min 31 sec	Dec 2, 2023, 12:30:01 AM	Succeeded
cis-9440-nyc-311-service-re-mp--1701408601-67621126990328374	Batch	Dec 1, 2023, 12:36:21 AM	6 min 20 sec	Dec 1, 2023, 12:30:01 AM	Succeeded
cis-9440-nyc-restaurant-ins-mp--1701408600-7459030164136971798	Batch	Dec 1, 2023, 12:36:05 AM	6 min 5 sec	Dec 1, 2023, 12:30:00 AM	Succeeded

Figure 12 - Pipeline scheduled job status report

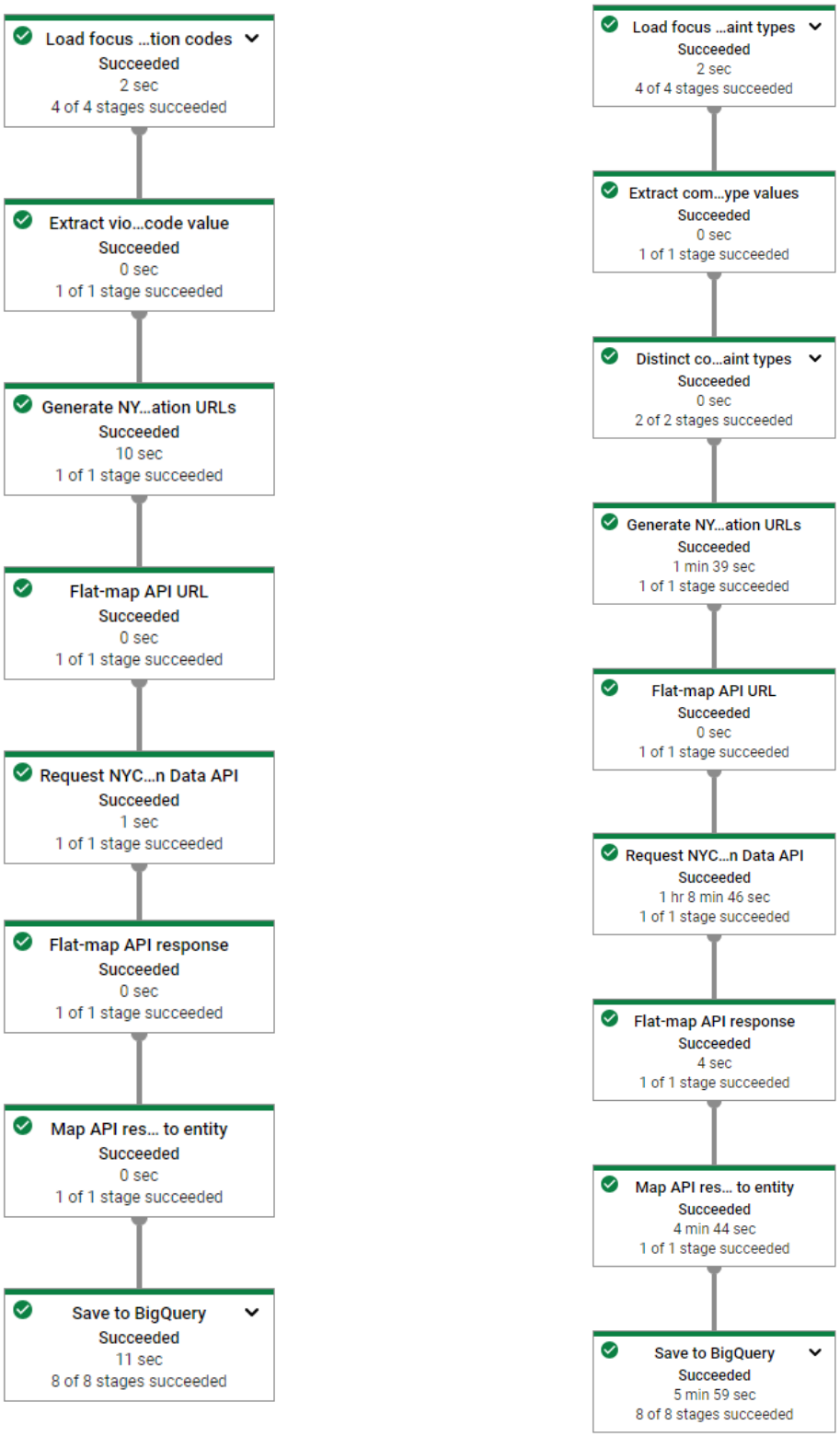


Figure 13 - Pipeline execution status report

10. Validate the results in the BigQuery tables

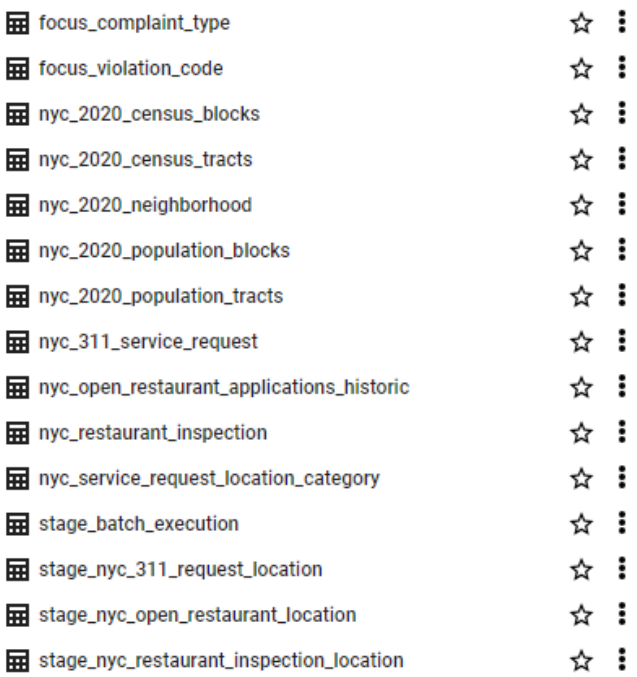


Figure 14 - Data sources and intermediary tables in Google BigQuery

The tables focus_violation_code and focus_complaint_type are two different datasets created to facilitate the filtering of relevant information, considering the substantial size of the primary dataset.

Geographical data including the tables with their names starts with nyc_2020 and the Open Restaurant data consists of historical data that do not need to be updated on a daily basis and was uploaded directly into BigQuery using CSV or Python scripts.

There are also staging tables that store the data for further processing for populating and aggregating data into the dimension and fact tables. It also tracks the execution of batch jobs and the latest record date to perform incremental loads.

stage_batch_execution QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
job_name	STRING	REQUIRED					
latest_record_date	DATETIME	REQUIRED					
execution_date	DATETIME	REQUIRED					

Figure 15 - Data sources and intermediary tables in Google BigQuery

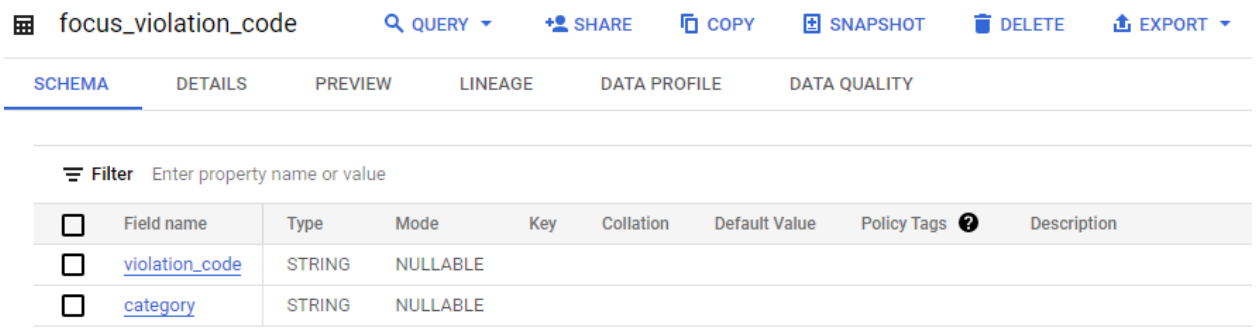
nyc_restaurant_inspection QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
camis	STRING	REQUIRED					
dba	STRING	NULLABLE					
boro	STRING	NULLABLE					
building	STRING	NULLABLE					
street	STRING	NULLABLE					
zipcode	STRING	NULLABLE					
phone	STRING	NULLABLE					
cuisine_description	STRING	NULLABLE					
inspection_date	DATETIME	REQUIRED					
action	STRING	NULLABLE					
violation_code	STRING	REQUIRED					
violation_description	STRING	NULLABLE					
critical_flag	STRING	NULLABLE					
score	NUMERIC	NULLABLE					

Figure 16 - Schema of NYC Restaurant Inspection table



focus_violation_code

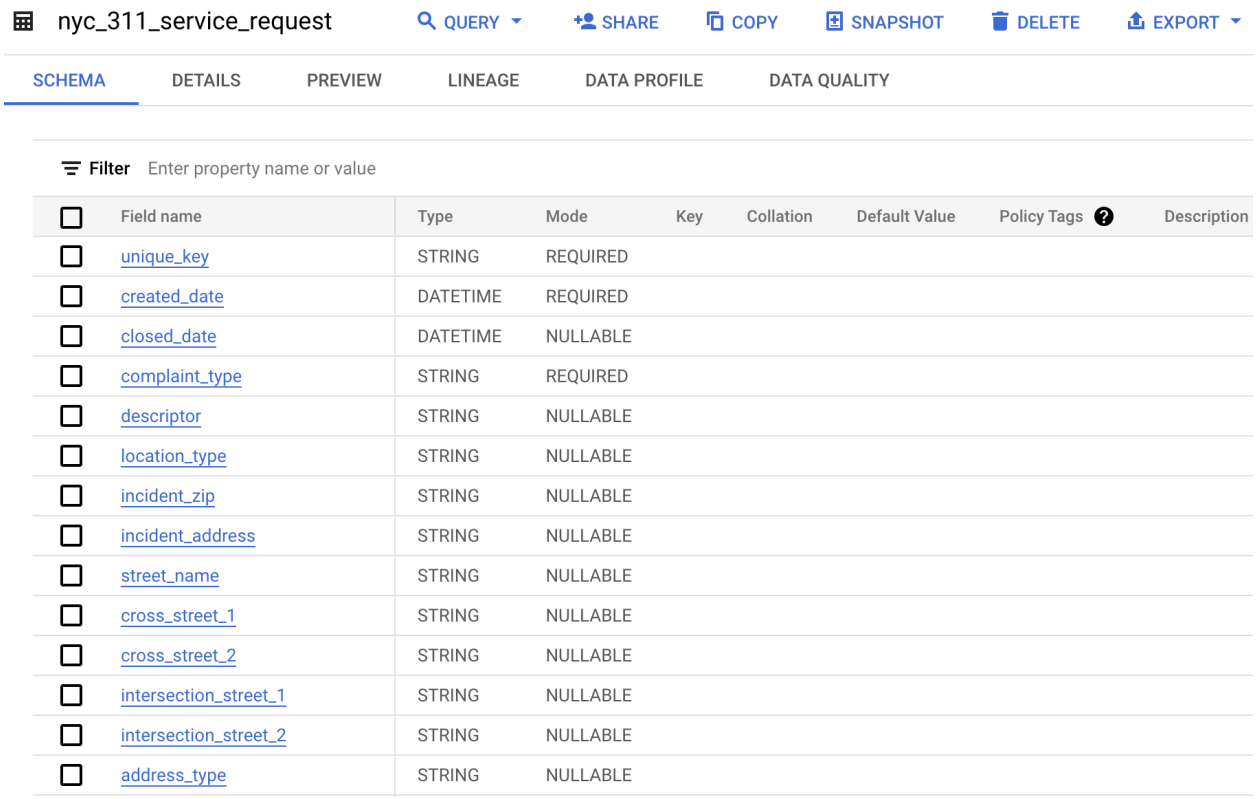
QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
violation_code	STRING	NULLABLE					
category	STRING	NULLABLE					

Figure 17 - Schema of the violation lookup table



nyc_311_service_request

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
unique_key	STRING	REQUIRED					
created_date	DATETIME	REQUIRED					
closed_date	DATETIME	NULLABLE					
complaint_type	STRING	REQUIRED					
descriptor	STRING	NULLABLE					
location_type	STRING	NULLABLE					
incident_zip	STRING	NULLABLE					
incident_address	STRING	NULLABLE					
street_name	STRING	NULLABLE					
cross_street_1	STRING	NULLABLE					
cross_street_2	STRING	NULLABLE					
intersection_street_1	STRING	NULLABLE					
intersection_street_2	STRING	NULLABLE					
address_type	STRING	NULLABLE					

Figure 18 - Schema of NYC 311 Service Requests table

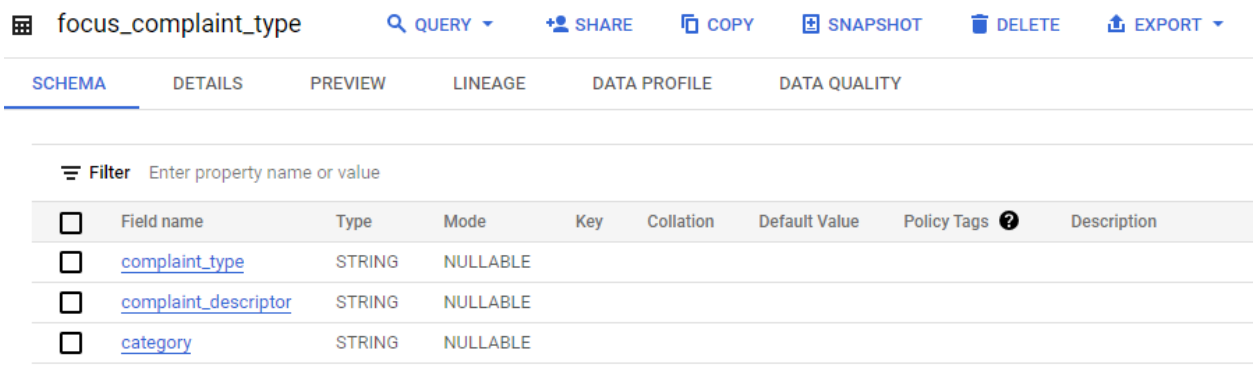


Figure 19 - Schema of the complaint type lookup table

Populate Dimension and Fact Tables

In addition to the data cleaning steps performed within the pipeline code, further data transformation and load processes are implemented by leveraging Google BigQuery Scheduled Queries. These scheduled SQL queries will aggregate data in the staging tables and generate relevant records for the dimension tables.

The SQL queries are scheduled to be executed at 2:00AM (New York local time) to wait until all the pipeline executions are complete first.

	Display name	Source	Schedule (UTC)	Region
✓	Incremental load DimDate	Scheduled Query	every day 07:00	us-east4
✓	Incremental load DimLocation	Scheduled Query	every day 07:00	us-east4
✓	Incremental load DimRestaurantViolation	Scheduled Query	every day 07:00	us-east4
✓	Incremental load Fact	Scheduled Query	every day 08:00	us-east4
●	Refrest DimOpenRestaurant	Scheduled Query	every 24 hours	us-east4

Figure 20 - Scheduled queries for data loading

Incremental load DimRestaur...ion RUN SCHEDULE MORE

```
DECLARE `latestProcessedInspectionDate` DEFAULT (
  SELECT IFNULL(MAX(`latest_record_date`), DATE(1900, 1, 1))
  FROM `avid-garage-399623.cis9440_project.stage_batch_execution`
  WHERE `job_name` = 'Load_DimRestaurantInspection'
  LIMIT 1
);

DECLARE `latestInspectionDate` DEFAULT (
  SELECT IFNULL(MAX(`inspection_date`), DATE(1900, 1, 1))
  FROM `avid-garage-399623.cis9440_project.nyc_restaurant_inspection`
  LIMIT 1
);

DECLARE `executionTime` DEFAULT CURRENT_DATETIME('America/New_York');

-- Update old records if category is changed
UPDATE `avid-garage-399623.cis9440_project.DimRestaurantViolation`
SET
  `EndDate` = `executionTime`,
  `IsCurrent` = 0
WHERE `RestaurantViolationID` IN (
  SELECT `dim`.`RestaurantViolationID`
  FROM `avid-garage-399623.cis9440_project.nyc_restaurant_inspection` AS `src`
  INNER JOIN `avid-garage-399623.cis9440_project.focus_violation_code` AS `fvc` ON `fvc`.`violation_code` = `src`.`violation_code`
  LEFT JOIN `avid-garage-399623.cis9440_project.DimRestaurantViolation` AS `dim` ON (
    `dim`.`Cuisine` = `src`.`cuisine_description`
    AND `dim`.`ViolationCode` = `src`.`violation_code`
    AND `dim`.`Category` <> `fvc`.`category`
  )
  WHERE `src`.`inspection_date` > `latestProcessedInspectionDate`
    AND `dim`.`RestaurantViolationID` IS NOT NULL
    AND `dim`.`IsCurrent` = 1
);

-- Insert new records
INSERT INTO `avid-garage-399623.cis9440_project.DimRestaurantViolation` (
  `RestaurantViolationID`, `Cuisine`, `ViolationCode`, `Category`, `StartDate`, `EndDate`, `IsCurrent`
)
SELECT
  GENERATE_UUID() AS `RestaurantViolationID`,
  `out`.*
FROM (
  SELECT DISTINCT
    `src`.`cuisine_description` AS `Cuisine`,
    `src`.`violation_code` AS `ViolationCode`,
    `fvc`.`category` AS `Category`,
    `executionTime` AS `StartDate`,
```

Figure 21 - SQL script for incremental/snapshot data load

Final Dimensional Schema

DimComplaint QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE

Filter Enter property name or value

Field name	Type	Mode	Key	Collation
ComplaintID	STRING	NULLABLE	PK	
ComplaintType	STRING	NULLABLE		
ComplaintDescription	STRING	NULLABLE		
Category	STRING	NULLABLE		

DimComplaint QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Row	ComplaintID	ComplaintType	ComplaintDescription	Category
1	eef12592-5397-4c54-a56d-99f...	Not Interested	N/A	None
2	f695cb69-aa5a-44b4-ac7d-c08...	Rodent	Rat Sighting	Pests
3	3512d5c4-8572-4457-85ce-a4b...	Rodent	Condition Attracting Rodents	Pests
4	e1680513-ac2b-43ff-acad-e18...	Rodent	Mouse Sighting	Pests
5	1876e215-5600-4698-998e-53...	Rodent	Signs of Rodents	Pests
6	e5eb8640-b5c8-46e5-9ae1-0fa...	Rodent	Rodent Bite - PCS Only	Pests
7	84c31413-5f38-4da5-997c-e9c...	Dead Animal	Rat or Mouse	Pests
8	7855b3c1-5e47-47c7-b55f-ce2...	Food Establishment	Rodents/Insects/Garbage	Pests
9	62643ca1-515b-472a-8f5e-12e...	Food Establishment	Pesticide	Pests
10	49d0b661-6609-417e-9007-95...	Mobile Food Vendor	Insects / Pests	Pests
11	af95fc47-71be-4ffc-8cde-ac67...	UNSANITARY CONDITION	PESTS	Pests
12	4623e84d-b21f-43b4-b71e-dba...	Unsanitary Condition	Pests	Pests
13	94e2332f-8c36-4843-90ac-9ba...	Unsanitary Condition	Rodents/Mice	Pests
14	fe47c7ef-d262-4be4-a62e-cd1f...	Unsanitary Condition	Insects / Pests	Pests
15	4ee3edfa-891b-496a-bb71-313...	Maintenance or Facility	Rodent Sighting	Pests
16	8328fde5-3010-4bfc-bc01-4c0...	Transfer Station Complaint	Insects or Rodents	Pests
17	2fae45df-8b43-45f2-b3ac-4eae...	Sewer	Sewer Backup (Use Comments...	Sewage
18	0f712ca5-4ff1-444d-af69-22c6...	Sewer	Catch Basin Clogged/Flooding ...	Sewage
19	015540f4-b369-4ec7-8475-538...	Sewer	Sewer Odor (SA2)	Sewage
20	e772582d-1443-4ae5-b8e7-b26...	Indoor Sewage	Sewage Odor	Sewage

Results per page: 50 1 – 50 of 124

DimDate QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE DA

Filter Enter property name or value

Field name	Type	Mode	Key	Col
DateID	STRING	NULLABLE	PK	
Year	INTEGER	NULLABLE		
Month	STRING	NULLABLE		
DayOfMonth	INTEGER	NULLABLE		
Week	INTEGER	NULLABLE		
DayOfWeek	STRING	NULLABLE		
IsWeekend	INTEGER	NULLABLE		

DimDate QUERY SHARE COPY SNAPSHOT DELETE EXPORT REFRESH

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Row	DateID	Year	Month	DayOfMonth	Week	DayOfWeek	IsWeekend
1	2016-01-01	2016	January	1	0	Friday	0
2	2021-01-01	2021	January	1	0	Friday	0
3	2020-01-03	2020	January	3	0	Friday	0
4	2018-01-05	2018	January	5	0	Friday	0
5	2019-01-04	2019	January	4	0	Friday	0
6	2023-01-06	2023	January	6	1	Friday	0
7	2020-01-10	2020	January	10	1	Friday	0
8	2016-01-08	2016	January	8	1	Friday	0
9	2017-01-06	2017	January	6	1	Friday	0
10	2019-01-11	2019	January	11	1	Friday	0
11	2021-01-08	2021	January	8	1	Friday	0
12	2022-01-07	2022	January	7	1	Friday	0
13	2018-01-12	2018	January	12	1	Friday	0
14	2021-01-15	2021	January	15	2	Friday	0
15	2019-01-18	2019	January	18	2	Friday	0
16	2023-01-13	2023	January	13	2	Friday	0
17	2017-01-13	2017	January	13	2	Friday	0
18	2022-01-14	2022	January	14	2	Friday	0
19	2020-01-17	2020	January	17	2	Friday	0
20	2018-01-19	2018	January	19	2	Friday	0

Results per page: 50 1 – 50 of 2894

DimOpenRestaurant QUERY SHARE

SCHEMA DETAILS PREVIEW LINEAGE DATA

Filter Enter property name or value

Field name	Type	Mode	Key
OpenRestaurantID	STRING	NULLABLE	PK
OpenRestaurantType	STRING	NULLABLE	
IsQualifyAlcohol	INTEGER	NULLABLE	

DimOpenRestaurant QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE

Row	OpenRestaurantID	OpenRestaurantType	IsQualifyAlcohol
1	b6af35a1-a8f9-4945-8cfa-d8e5...	both	0
2	8a4074f8-969b-406d-bbe7-ff8d...	both	1
3	1bb39e02-4e6e-4ec6-af95-14b...	none	0
4	77bd2b14-f395-4135-a43c-80b...	roadway	0
5	a00c29c6-69e8-4d8f-aa71-30b...	roadway	1
6	181c7176-2804-4880-8dbd-b1...	sidewalk	0
7	41b143c3-f9de-4d69-b8b9-b40...	sidewalk	1

DimRestaurantViolation QUERY SHARE C

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE

Filter Enter property name or value

Field name	Type	Mode	Key	Collation
RestaurantViolationID	STRING	NULLABLE	PK	
Cuisine	STRING	NULLABLE		
ViolationCode	STRING	NULLABLE		
Category	STRING	NULLABLE		
StartDate	DATETIME	NULLABLE		
EndDate	DATETIME	NULLABLE		
IsCurrent	INTEGER	NULLABLE		

DimRestaurantViolation QUERY SHARE COPY SNAPSHOT DELETE EXPORT REFRESH

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Row	RestaurantViolationID	Cuisine	ViolationCode	Category	StartDate	EndDate	IsCurrent
1	c6bf9bdc-2394-4bfb-a5d8-463...	Tex-Mex	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
2	1cfbc778-cfc7-4075-bed3-0fad...	Polish	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
3	320d69e9-2c12-46ac-afdc-7dd...	Irish	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
4	bb736bba-727d-4af3-bfda-12e...	Salads	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
5	ce7315bd-a215-4abd-b6d7-bae...	Italian	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
6	43944b39-1c48-4564-aa09-b03...	Australian	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
7	0d79b386-59e9-4e91-825b-e4b...	Indian	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
8	f55b1ba7-a308-4597-922f-46e...	Other	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
9	44a8a1f3-4944-4901-b791-c04...	Creole	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
10	120ed4e0-c53c-4587-ac31-8fc...	Bangladeshi	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
11	d484228a-0aac-4178-ae66-f5d...	Caribbean	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
12	4dc820d9-6a09-4fba-87ed-cf3...	Frozen Desserts	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
13	d390c4bc-b68b-46a7-8a20-afe...	Hotdogs	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
14	fe47ee6e-b610-437a-b69e-955...	Vegetarian	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
15	14f85d9c-22c2-4e91-b4de-0f7...	Mediterranean	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
16	586e6ab9-5bf9-4c8b-a79c-fb5...	New American	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
17	6a0a3c5e-4792-4351-b0ea-237...	Spanish	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
18	5a6a4d59-3bf4-433c-8f8e-422...	Californian	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
19	0db5aa00-e7b1-4dcd-8e54-3f6...	Pizza	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
20	eb18854e-2e41-4ecb-86b3-2b8...	Filipino	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
21	abf831e5-d50b-4196-b3a5-7c1...	Middle Eastern	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
22	fec1bdd3-f44a-411d-9991-e7c...	Jewish/Kosher	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
23	58eeaf97-25e1-407c-9fd4-8ce4...	Vegan	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
24	063995d1-1a63-4381-99ab-d1...	Brazilian	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
25	a8f39cda-be5a-4c8a-aa71-e89...	Latin American	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
26	2b1f2881-f89b-4244-bba3-21e...	Pakistani	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
27	a9f709ba-55c5-48d0-a0b4-f86...	Hamburgers	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1
28	f86d1812-c908-44b7-be49-f77...	Turkish	04K	Pests	2023-12-04T04:22:49.303666	9999-12-31T00:00:00	1

Results per page: 50 1 – 50 of 715

DimTime QUERY SHARE

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode	Key
TimeID	STRING	NULLABLE	PK
Hour	INTEGER	NULLABLE	
AMPM	STRING	NULLABLE	
TimePeriod	STRING	NULLABLE	

DimTime QUERY SHARE COPY SNAPSHOT DELETE EXPORT

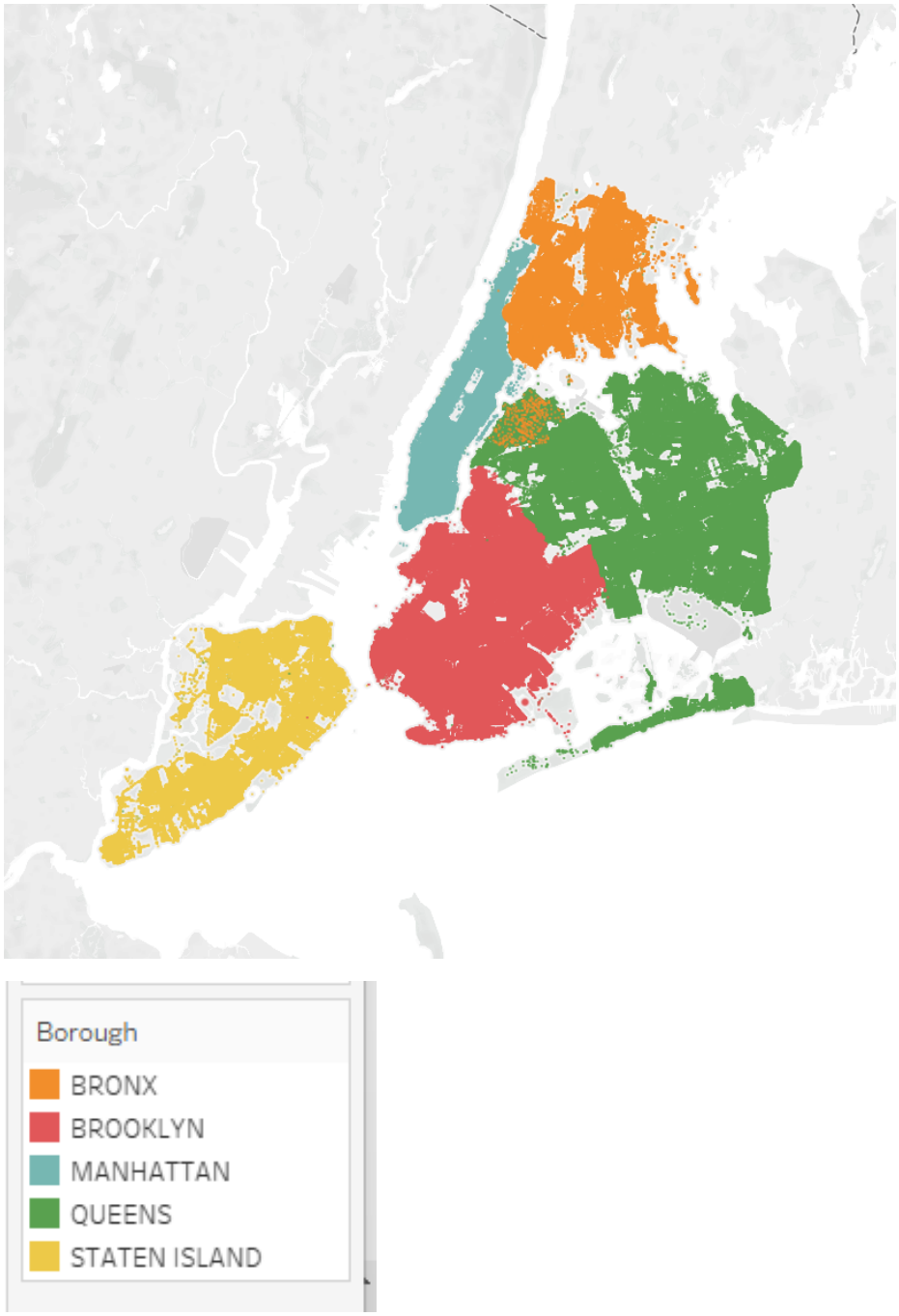
SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

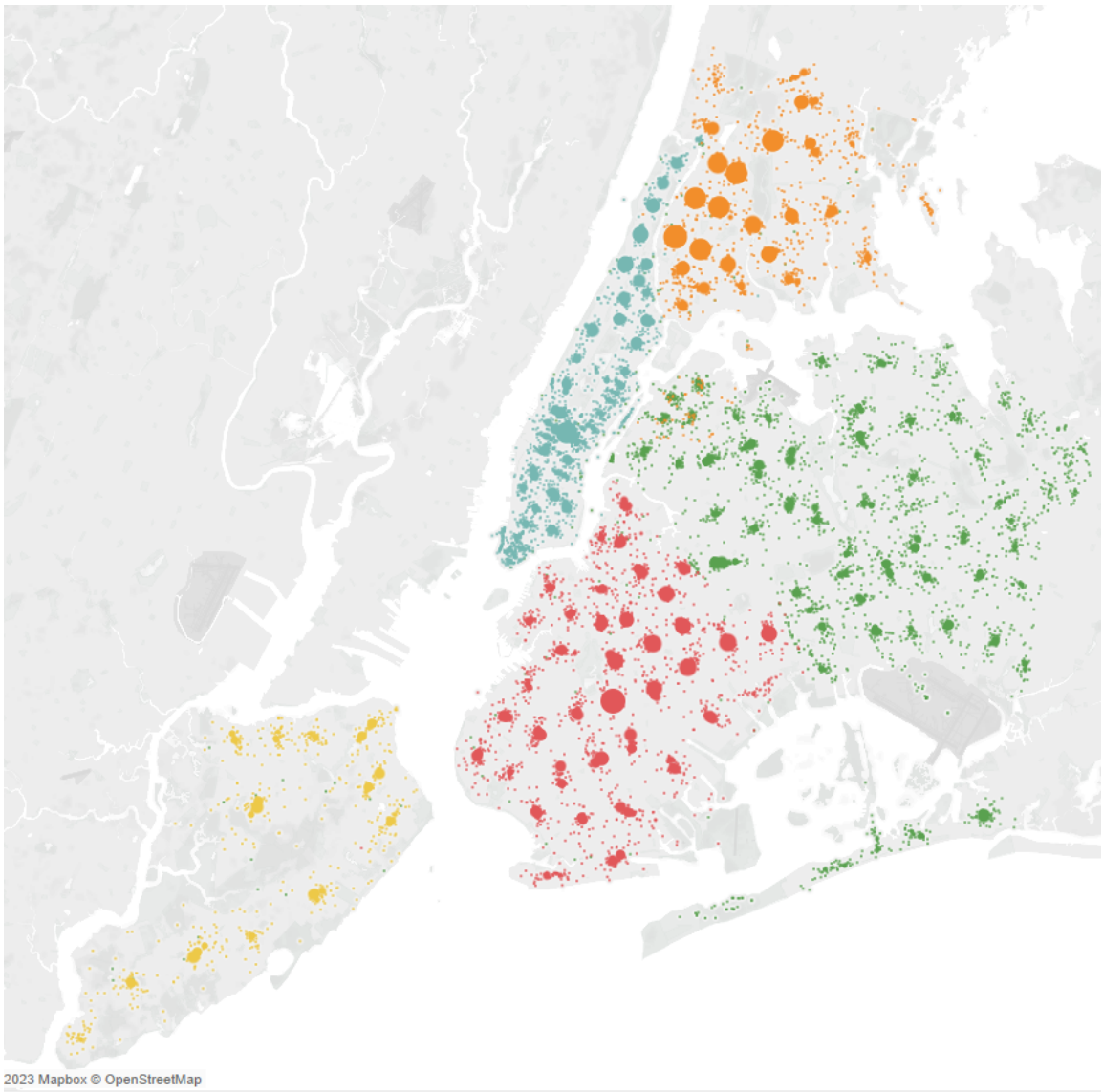
Row	TimeID	Hour	AMPM	TimePeriod
1	3f430bf8-673f-4e20-b32a-d0ae...	0	AM	Night
2	8295f3e0-39f1-45dd-a279-d65...	1	AM	Night
3	14467eff-351a-400d-80db-4ff2...	2	AM	Night
4	19846467-f287-4607-9bfe-6b6...	3	AM	Night
5	0a2b144d-98e9-4962-808d-f53...	4	AM	Night
6	6864c369-6103-43e9-adeb-3c8...	5	AM	Early Morning
7	c7fd9d91-c06f-4f10-95f0-3453...	6	AM	Early Morning
8	bb58f110-574a-4f60-844e-02b...	7	AM	Breakfast
9	289e049e-ab3e-4e17-8da2-ea0...	8	AM	Breakfast
10	19f11524-0b7a-4195-b029-a9f...	9	AM	Breakfast
11	cca5f45f-c590-4982-b0c2-272...	10	AM	Morning
12	df0fec38-bf10-43ff-9a35-e0817...	11	AM	Morning
13	f7e7b84d-ecae-46ec-8494-345...	12	PM	Lunch
14	4f319087-8076-414f-9c46-959...	13	PM	Lunch
15	fd224ded-e68c-4fff-8388-7986...	14	PM	Lunch
16	e674e608-b31d-4d33-a41e-e79...	15	PM	Afternoon
17	c0b6bed5-96d7-4cf0-82cf-e1f3...	16	PM	Afternoon
18	e0df3fd3-c4a7-4f66-b5cc-d25a...	17	PM	Afternoon
19	eeb2b163-540d-4743-97d6-56...	18	PM	Dinner
20	72ce2bc3-16f3-427c-b57f-4f75...	19	PM	Dinner
21	cf8623cd-5612-45f9-a8bf-9213...	20	PM	Dinner
22	8a90f234-ab20-413b-bab0-766...	21	PM	Dinner
23	34245197-4456-4a51-a91e-65e...	22	PM	Night
24	dcb58ef5-603c-4dc4-a751-1e6...	23	PM	Night

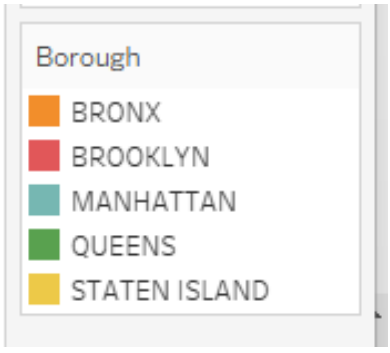
Results per page: 50 1 – 24 of 24

KPI Visualizations:

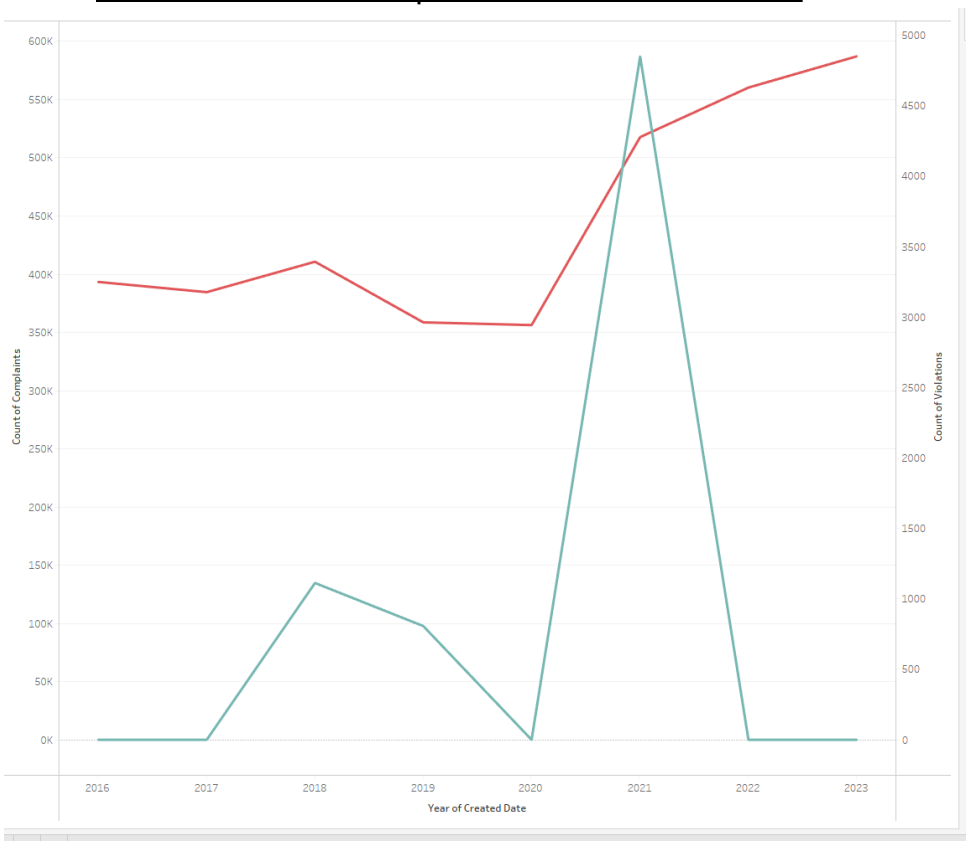
1) Number of Rodent Complaints overtime:

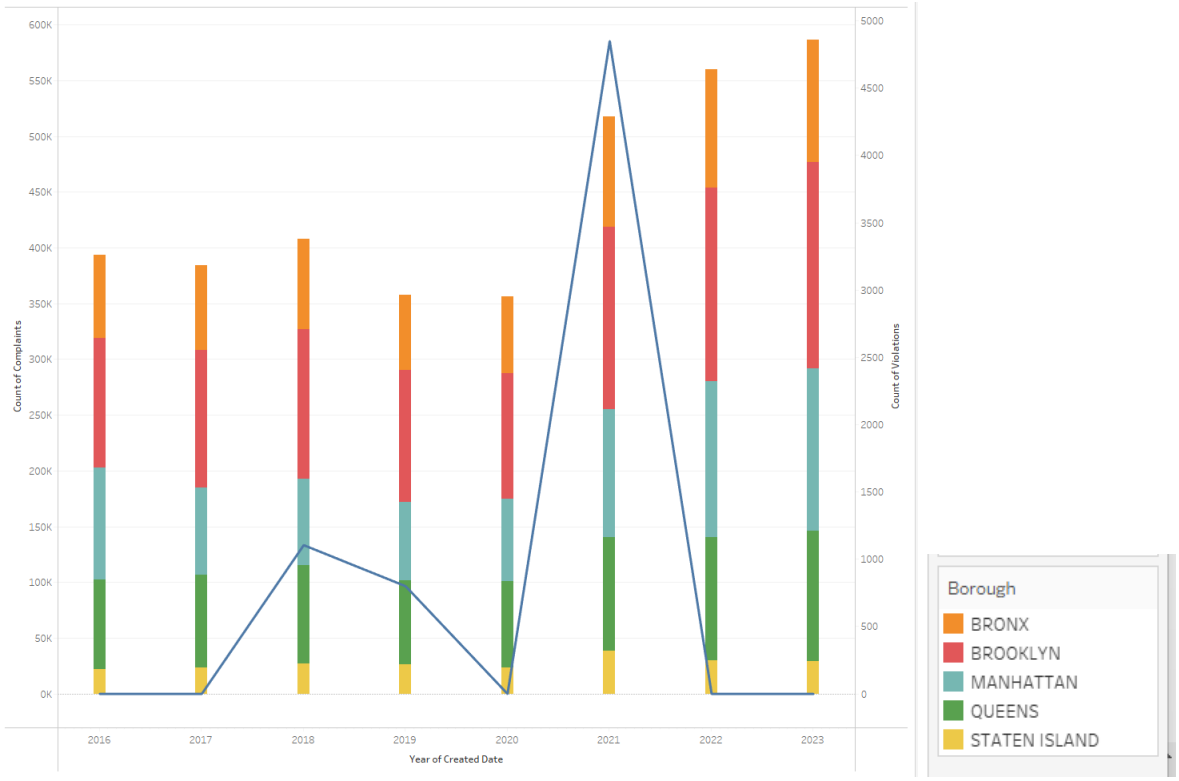


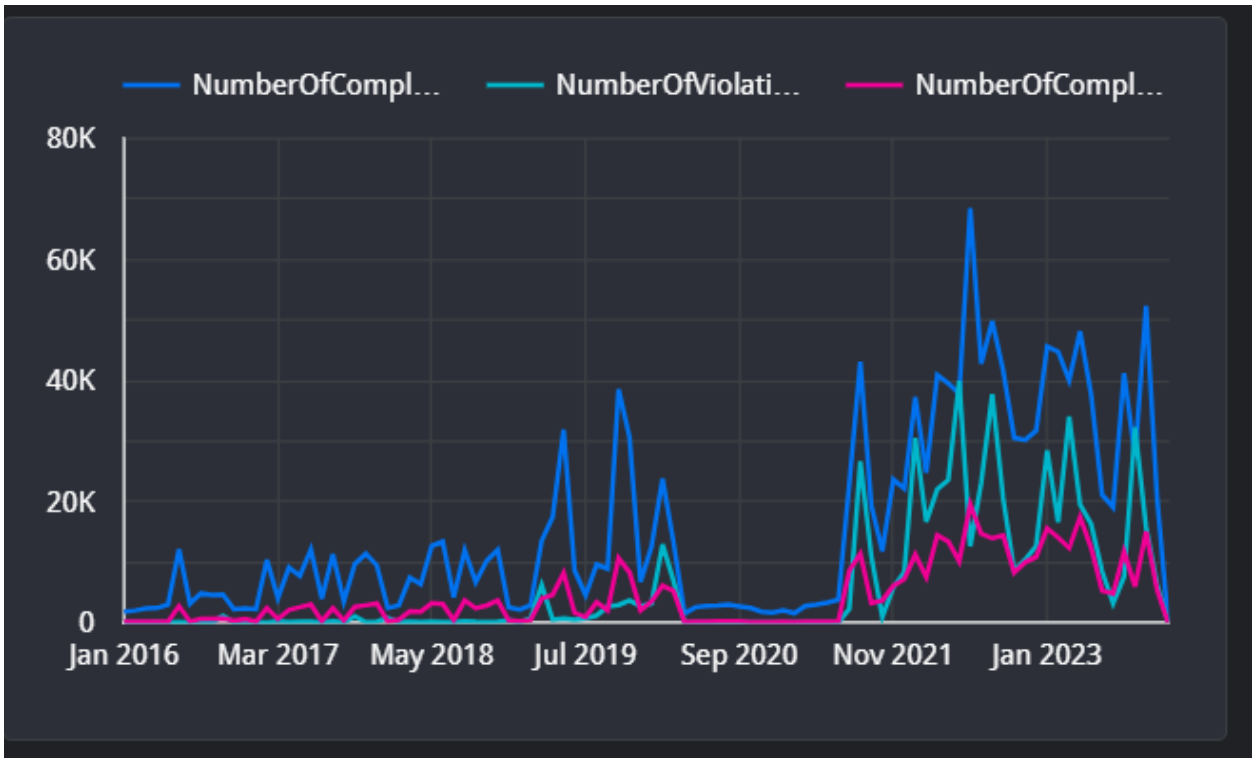

2023 Mapbox © OpenStreetMap

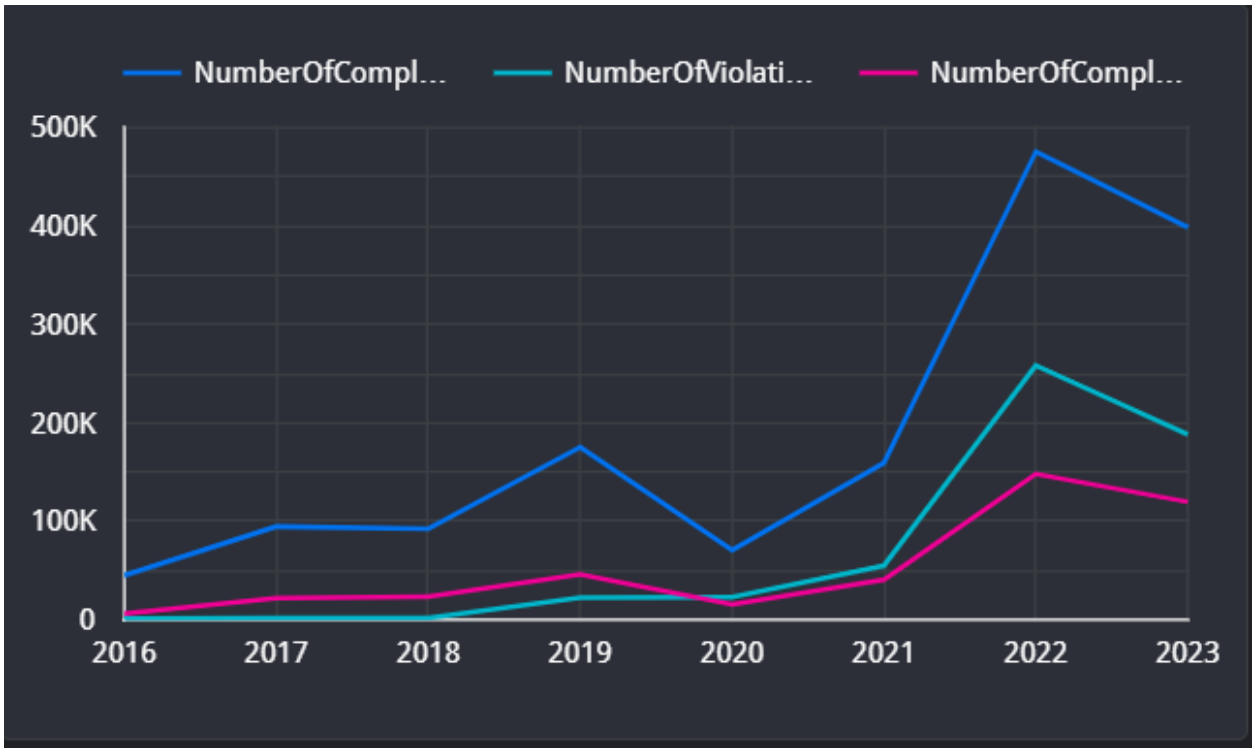

Borough
BRONX
BROOKLYN
MANHATTAN
QUEENS
STATEN ISLAND

2) Number of Complaints vs Violations

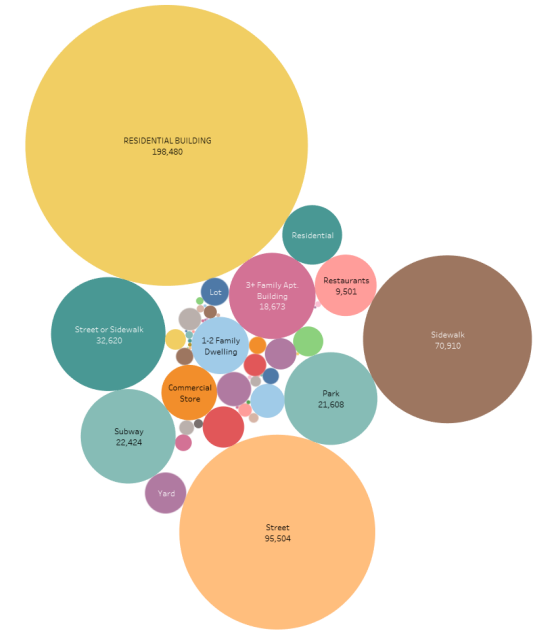


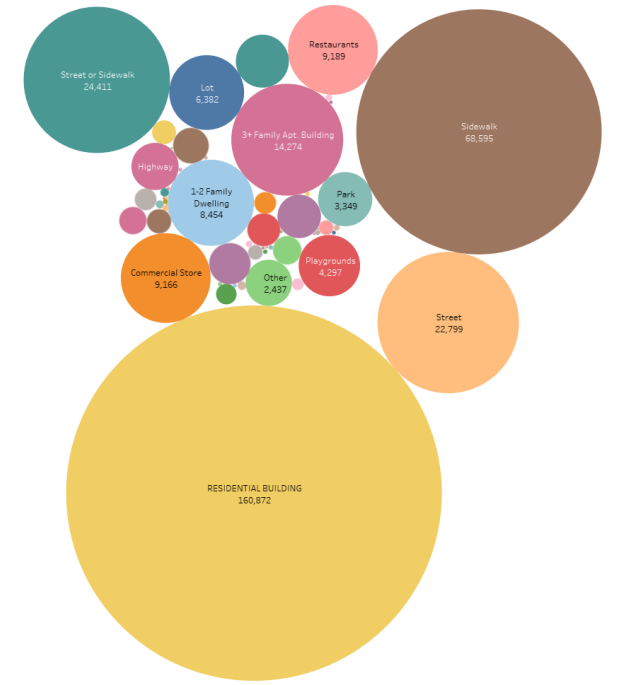



NumberOfCompl...
NumberOfViolati...
NumberOfCompl...
80K
60K
40K
20K
0
Jan 2016
Mar 2017
May 2018
Jul 2019
Sep 2020
Nov 2021
Jan 2023

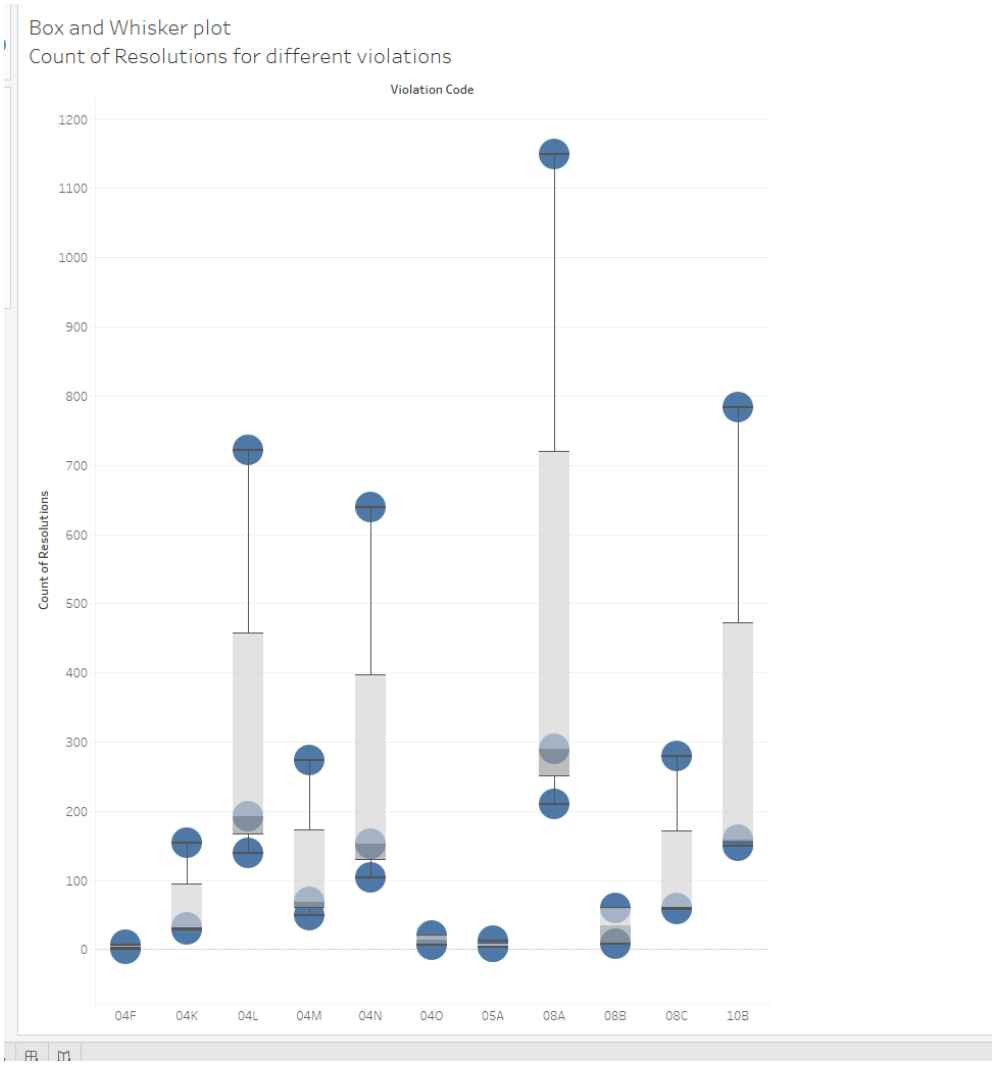

NumberOfCompl...
NumberOfViolati...
NumberOfCompl...
500K
400K
300K
200K
100K
0
2016
2017
2018
2019
2020
2021
2022
2023

3) Number of Complaints per location Type





4) Total Resolutions for every violation type



violation_code	category
04F	Sewage
04K	Pests
04L	Pests
04M	Pests
04N	Pests
04O	Animal Waste
05A	Sewage
08A	Pests
08B	Dirty Conditions
08C	Pests
10B	Sewage

Conclusion

Software and database tools

Throughout this project, the following software and database tools have been utilized to carry out the project:

- **Python**: developed the custom scripts to process and load the data
- **Apache Beam**: used for developing data processing pipelines
- **Google Dataflow**: used to orchestrate the ETL pipeline execution
- **Google BigQuery**: used as the data warehouse solution and to analyze and query large datasets
- **Google BigQuery Scheduled Queries**: used to load the aggregated data to the dimensional model
- **Google Docs**: used for collaborative documentation and sharing project-related information
- **Google Slides**: used to create and communicate presentation information
- **Google Cloud**: used to host and manage various project-related resources such as datasets and storage
- **Looker studio**: used to create data visualizations and build dashboards
- **Tableau**: a secondary visual tool used to create more detailed data visualizations
- **Zoom**: platform to meet with group members to discuss each deliverable
- **Visual Studio Code**: used as a development platform for writing and testing code

Group Experience

Engaging in the project was a natural application of the knowledge acquired during the course. As a team, we swiftly reached a consensus on the project's focal point. Moreover, seamlessly integrating Google's suite of tools proved to be a smooth experience. However, it took some effort to familiarize ourselves with all the tools that were needed. Notably, employing LookerStudio for our dashboard was a step into unfamiliar territory for most team members. Thankfully, the instructional foundation laid in class proved immensely beneficial.

The main challenge of this project is the large amount of data that needs to be processed. If the traditional SQL script was the only tool used in the project, the execution time would be endless and the analysis could not be delivered on time.

Initially, Pentaho seemed like the ideal fit, but as we delved deeper, it became evident that

collaborating on the project with this tool would be challenging. Exploring alternative options led us to the realization that constructing our code was the optimal path forward. Google Dataflow was selected to be the main ETL tool as it is capable of horizontally scaling to meet the processing requirement; however, Google Dataflow requires a substantial amount of learning curve to understand the parallel pipeline processing concept and how to use Apache Beam which is the underlying technology of Google Dataflow. Moreover, Google Cloud poses a relatively low processing quota for trial and new accounts. This issue made the pipeline execution time take a significant amount to complete and hindered the development and testing process. Considering a classic data-formatting issue, this project also faced this problem. Some exported data contained invalid values and formats that prevented it from directly loading the CSV data using Google's built-in tool. An extra development effort, then, was required to create Python scripts to read, process, and send the data back to the cloud. Consequently, we recognized the need for a more extended timeframe to complete the ETL process. This phase emerged as the most time-intensive aspect of the entire project.

Proposed benefits can be realized by the new system

Because the new BI intelligence application is able to effectively integrate data from multiple sources it allows the end user to explore the data in a variety of perspectives and transformations. These data sources allow the end users to slice and dice data from multiple dimensional views that allow them to drill down on dimensions such as time, location, and date in order to aggregate the data for more useful analytics. By analyzing the relationship between rodent infestations, population density, inspection violations, and geographical locations, for example, our system can identify high-risk areas as well as diagnose potential causes. In addition, by leveraging the date and time dimensions, further insights into weather seasons can generate significant analytics. These data-driven insights will offer city officials and policymakers suggestions on how to allocate their resources to combat the rodent infestation issue in order to enhance public safety and protect businesses from infestation.

Final comments and conclusions

The project required the complex integration of data from various sources into a dimensional data warehouse so that the data could be consumed for analytics and business intelligence applications that support insight-driven decision-making. Throughout the course of this project, the team was successfully able to navigate the complexities of building an end-to-end data warehouse. Due to the intensity of the project and the limited timeframe to complete it, we would encourage further data cleansing to ensure data integrity that provides more insightful analytics. However, due to the structure of the project, we were able to become more knowledgeable with new data warehousing concepts and tools.

References:

- “Mayor Adams Anoints Kathleen Corradi as NYC’s First-Ever “Rat Czar.”” The Official Website of the City of New York, 2023, www.nyc.gov/office-of-the-mayor/news/249-23/mayor-adams-anoints-kathleen-corradi-nyc-s-first-ever-rat-czar-#/0. Accessed 11 Sept. 2023.
- “NYC Mayor Introduces City’s First Rat Czar Kathleen Corradi: “She HATES Rats” - CBS News.” Www.cbsnews.com, 12 Apr. 2023, www.cbsnews.com/news/nyc-rat-czar-kathleen-corradi/. Accessed 7 Oct. 2023.
- Warerkar, Tanay. “A Timeline of COVID-19’S Impact on NYC’s Restaurant Industry.” Eater NY, 30 Dec. 2020, ny.eater.com/2020/12/30/22203053/nyc-coronavirus-timeline-restaurants-bars-2020.
- "Rats in New York City." Wikipedia. https://en.wikipedia.org/wiki/Rats_in_New_York_City. Accessed September 11, 2023.
- New York City Department of Health - Violation Penalty PDF: "Violation Penalty Schedule." New York City Department of Health. https://www.nyc.gov/assets/doh/downloads/pdf/rii/ri-violation-penalty.pdf. Accessed September 11, 2023.
- Google Cloud BigQuery Documentation - Geography Functions: "Geography Functions - BigQuery Standard SQL." Google Cloud. https://cloud.google.com/bigquery/docs/reference/standard-sql/geography_functions. Accessed September 11, 2023.
- "Apache Beam Python Introduction Guide." Apache Beam Tours. https://tour.beam.apache.org/tour/python/introduction/guide. Accessed November 21, 2023.
- Deploying a Pipeline." Google Cloud Dataflow Documentation. https://cloud.google.com/dataflow/docs/guides/deploying-a-pipeline. Accessed November 21, 2023.

Appendix A - Mapping of Complaint Types

Complaint Type	Descriptor	Category	Records
UNSANITARY CONDITION	PESTS	Pests	409465
Sanitation Condition	15 Street Cond/Dump-Out/Drop-Off	Dirty Conditions	257587
Rodent	Rat Sighting	Pests	232204
Sewer	Sewer Backup (Use Comments) (SA)	Sewage	190600
Missed Collection (All Materials)	1 Missed Collection	Dirty Conditions	146170
Homeless Person Assistance		Homelessness	133373
Dirty Conditions	E3 Dirty Sidewalk	Dirty Conditions	127967
Sewer	Catch Basin Clogged/Flooding (Use Comments) (SC)	Sewage	111661
Homeless Person Assistance	N/A	Homelessness	103752
Sanitation Condition	12 Dead Animals	Animal Waste	101872
Encampment	N/A	Homelessness	100726
Rodent	Condition Attracting Rodents	Pests	94742
Dirty Condition	Trash	Dirty Conditions	88928
Dirty Conditions	E3A Dirty Area/Alleyway	Dirty Conditions	69278
Missed Collection	Trash	Dirty Conditions	61519
Missed Collection (All Materials)	2 Bulk-Missed Collection	Dirty Conditions	56251

Complaint Type	Descriptor	Category	Records
Illegal Dumping	Removal Request	Dirty Conditions	55664
Rodent	Mouse Sighting	Pests	54626
Missed Collection (All Materials)	1R Missed Recycling-All Materials	Dirty Conditions	52966
Dirty Conditions	E1 Improper Disposal	Dirty Conditions	40939
Maintenance or Facility	Garbage or Litter	Dirty Conditions	39294
Homeless Encampment	N/A	Homelessness	38208
Food Establishment	Rodents/Insects/Garbage	Pests	37286
Rodent	Signs of Rodents	Pests	36188
Dirty Conditions	E12 Illegal Dumping Surveillance	Dirty Conditions	31681
Missed Collection (All Materials)	1RG Missed Recycling Paper	Dirty Conditions	29527
Dirty Conditions	E8 Canine Violation	Animal Waste	28872
Missed Collection (All Materials)	1RB Missed Recycling - M/G/Pl	Dirty Conditions	23902
Missed Collection (All Materials)	2R Bulk-Missed Recy Collection	Dirty Conditions	23439
UNSANITARY CONDITION	SEWAGE	Sewage	22047
Sewer	Sewer Odor (SA2)	Sewage	19092
Missed Collection	Recycling - Paper/Metal/Glass/Rigid Plastic	Dirty Conditions	17725
Missed Collection	Bulky Trash	Dirty Conditions	17365

Complaint Type	Descriptor	Category	Records
Dirty Conditions	E5 Loose Rubbish	Dirty Conditions	17298
Overflowing Litter Baskets	6 Overflowing Litter Baskets	Dirty Conditions	16629
Homeless Street Condition	N/A	Homelessness	15583
Dirty Conditions	E2 Receptacle Violation	Dirty Conditions	14230
Missed Collection	Recycling - Paper	Dirty Conditions	13314
Unsanitary Animal Pvt Property	Dog	Animal Waste	12008
Dirty Conditions	E11 Litter Surveillance	Dirty Conditions	11867
Illegal Dumping	Chronic Dumping	Dirty Conditions	11658
Maintenance or Facility	Rodent Sighting	Pests	11539
Street Sweeping Complaint	Inadequate Sweeping	Dirty Conditions	9706
Sweeping/Missed	3A Sweeping/Missed	Dirty Conditions	9531
Indoor Sewage	Sewage Odor	Sewage	9219
Unsanitary Pigeon Condition	Pigeon Waste	Animal Waste	9121
Missed Collection	Recycling - Metal/Glass/Rigid Plastic	Dirty Conditions	9107
Obstruction	Trash or Recycling	Dirty Conditions	8875
Dead Animal	Cat	Dirty Conditions	8770
Homeless Person Assistance	No Status Call	Homelessness	8730

Complaint Type	Descriptor	Category	Records
Dirty Conditions	E4 18" Law	Dirty Conditions	8657
Highway Condition	Litter	Dirty Conditions	8624
Missed Collection (All Materials)	1RO Missed Recycling Organics	Dirty Conditions	8568
Missed Collection	Bulky Recycling	Dirty Conditions	8480
Residential Disposal Complaint	Trash or Recycling Not Secure	Dirty Conditions	8362
Street Sweeping Complaint	Street Not Swept	Dirty Conditions	7709
Missed Collection	Compost	Dirty Conditions	7643
Sweeping/Missed-Inadequate	3 Sweeping/Missed-Inadequate	Dirty Conditions	5913
Residential Disposal Complaint	Waste Set Out Too Early or Too Late	Dirty Conditions	5829
Unsanitary Animal Pvt Property	Animal Waste	Animal Waste	5660
Unsanitary Animal Pvt Property	Cat	Animal Waste	5388
Animal in a Park	Dead Animal	Animal Waste	4828
Dirty Condition	Dog Waste	Animal Waste	4610
Litter Basket Request	New Basket	Dirty Conditions	4408
Unsanitary Animal Pvt Property	Animal Odor	Animal Waste	4130
Missed Collection	Trash and Bulky Trash	Dirty Conditions	4072
Residential Disposal Complaint	Waste Left in Front of Other Residence	Dirty Conditions	4030

Complaint Type	Descriptor	Category	Records
Homeless Person Assistance	Status Call	Homelessness	3963
Dirty Conditions	E1A Litter Basket / Improper Use	Dirty Conditions	3827
Lot Condition	Trash	Dirty Conditions	3779
Indoor Sewage	Sewage Leak	Sewage	3463
Highway Condition	Dead Animal	Animal Waste	3287
Dead Animal	Raccoon	Animal Waste	3063
Missed Collection (All Materials)	1C Uncollected Xmas Trees	Dirty Conditions	3000
Litter Basket Complaint	Overflowing	Dirty Conditions	2872
Litter Basket Request	Replacement Basket	Dirty Conditions	2821
Dead Animal	Rat or Mouse	Pests	2675
Sweeping/Inadequate	3B Sweeping/Inadequate	Dirty Conditions	2627
Sanitation Worker or Vehicle Complaint	Spilled Garbage	Dirty Conditions	2568
Dirty Conditions	E13 Throw-Out	Dirty Conditions	2469
Animal in a Park	Animal Waste	Animal Waste	2014
Dead Animal	Opossum	Animal Waste	1931
Dumpster Complaint	Overflowing	Dirty Conditions	1917
Dumpster Complaint	Blocking Sidewalk or Street	Dirty Conditions	1633

Complaint Type	Descriptor	Category	Records
Sewer Maintenance	Backup	Sewage	1550
Homeless Person Assistance	Non-Chronic	Homelessness	1469
Dead Animal	Other	Animal Waste	1448
Dead Animal	Squirrel	Animal Waste	1433
Dead Animal	Bird	Animal Waste	1339
Commercial Disposal Complaint	Waste Left in Front of Residence	Dirty Conditions	1235
Dead Animal	Dog	Animal Waste	940
Unsanitary Pigeon Condition	Pigeon Odor	Animal Waste	794
Homeless Person Assistance	Chronic	Homelessness	683
Food Establishment	Sewage	Sewage	672
Unsanitary Animal Facility	Animal Odor	Animal Waste	625
Missed Collection (All Materials)	1L Missed Recycling Leaves	Dirty Conditions	600
Residential Disposal Complaint	Storage Area Not Provided	Dirty Conditions	589
Unsanitary Animal Facility	Animal Waste	Animal Waste	582
Sewer Maintenance	Catch Basin Clogged	Sewage	550
Mobile Food Vendor	Insects / Pests	Pests	534
Litter Basket Complaint	Misuse	Dirty Conditions	454

Complaint Type	Descriptor	Category	Records
Dead Animal	Deer	Animal Waste	451
Mobile Food Vendor	Garbage	Dirty Conditions	436
Residential Disposal Complaint	Containers Left Out After Collection	Dirty Conditions	417
Food Establishment	Pesticide	Pests	398
Dirty Conditions	E7 Private Carter Spillage	Dirty Conditions	360
Dumpster Complaint	Uncovered	Dirty Conditions	310
Commercial Disposal Complaint	Waste Disposal	Dirty Conditions	196
Overflowing Recycling Baskets	6R Overflowing Recycling Baskets	Dirty Conditions	196
DPR Internal	Illegal Dumping	Dirty Conditions	150
Water Drainage	Drain/Pipe Clogged	Sewage	136
Recycling Basket Complaint	Overflowing	Dirty Conditions	125
Municipal Parking Facility	Dirt, Litter, Debris - Lot	Dirty Conditions	112
Rodent	Rodent Bite - PCS Only	Pests	62
Residential Disposal Complaint	Containers Set Out Too Early	Dirty Conditions	51
Municipal Parking Facility	Dirt, Litter, Debris - Garage	Dirty Conditions	41
Unsanitary Condition	Pests	Pests	4
Unsanitary Condition	Sewage	Sewage	2

Complaint Type	Descriptor	Category	Records
Encampment		Homelessness	1
Transfer Station Complaint	Insects or Rodents	Pests	1
Transfer Station Complaint	Water or Sewer Runoff	Sewage	1
Unsanitary Condition	Rodents/Mice	Pests	1
Unsanitary Condition	Insects / Pests	Pests	1