Uncovering the Link: Collisions and Noise Complaints in NYC

Group members:

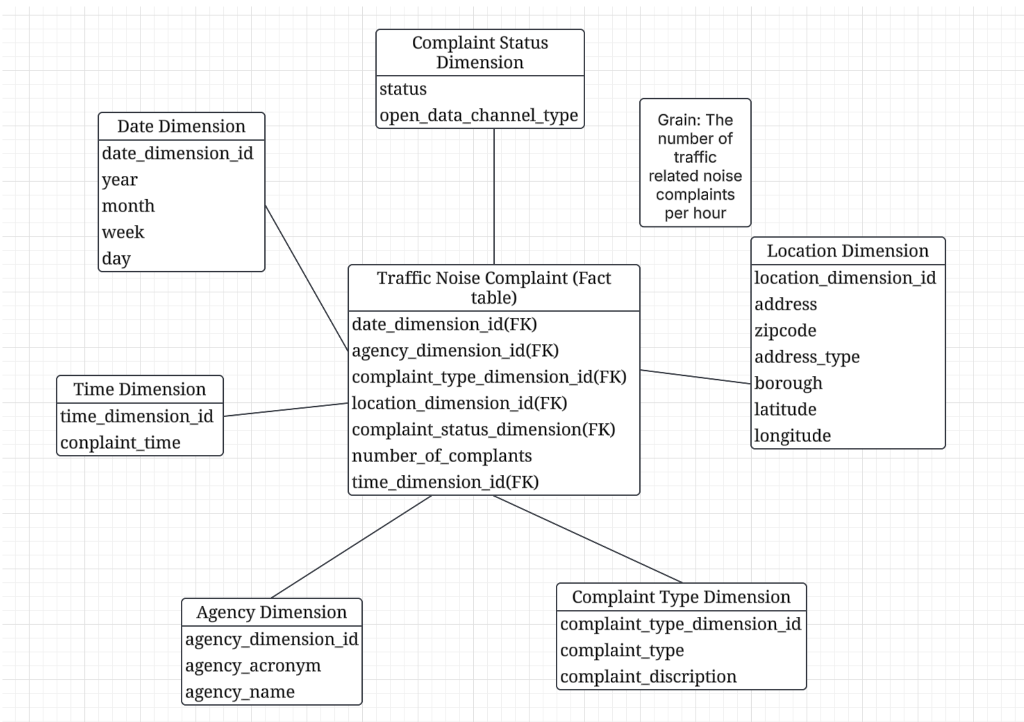
- Yu-na Choi (yuna.choi@baruchmail.cuny.edu)
- Ismail Bakere (ismail.bakere@baruchmail.cuny.edu)
- Wesley Lau (wesley.lau@baruchmail.cuny.edu)
- Mohammad Imdadul Alam (mohammad.alam3@baruchmail.cuny.edu)
- Katherine Alvarado (Katherine.alvarado@baruchmail.cuny.edu)

Introduction:

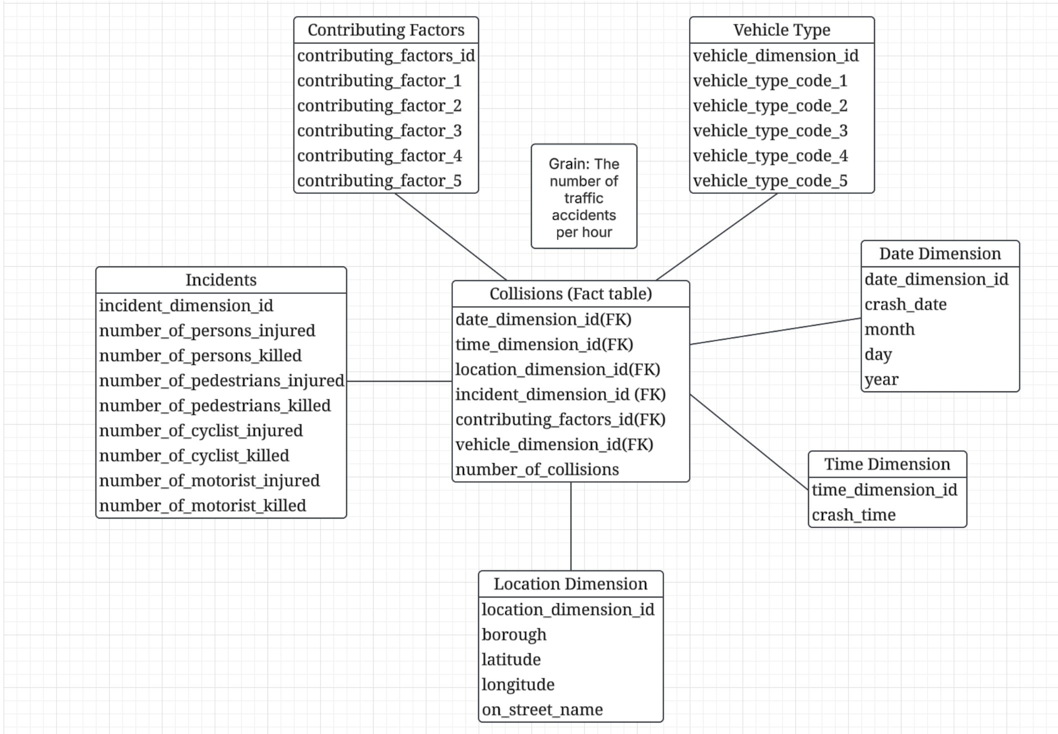
This project aims to integrate and analyze two key datasets to explore potential correlations between vehicle-related noise complaints and traffic accident patterns across New York City. The analysis draws on the 311 Service Requests dataset–focused specifically on traffic and vehicle noise complaints–and the Motor Vehicle Collisions dataset, which includes detailed records of crashes, such as location, time, and contributing factors. Traffic accidents and noise complaints are often interconnected, with collisions leading to congestion, prolonged idling, honking, and emergency responses–all of which contribute to elevated noise levels and resident frustration, particularly in densely populated areas. Through spatial and temporal analysis, this project seeks to identify areas of overlap and uncover patterns that may inform data-driven approaches to traffic management and noise mitigation.

Proposed Dimensional Model

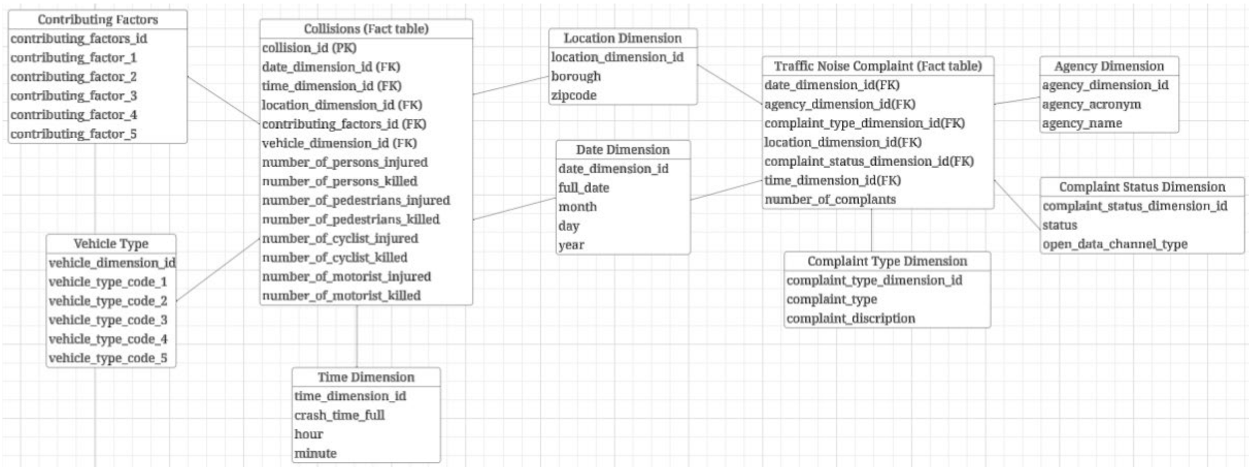
- Dimensional model for the 311 Complaints Data Mart: It's broken down into 5 separate dimensions and the Fact table. We chose to set the granularity at the transaction level to enable a more detailed analysis. Our initial plan was to record the number of complaints per day, but we realized that this level of aggregation might omit important patterns. As the project evolved, we decided to keep the dataset at the transaction level. This shift allows us to explore trends such as fluctuations between daytime and nighttime, or peak versus off-peak periods.



- Dimensional model for crashes Data Mart: This data mart is based on the NYPD's record of all reported motor vehicle crashes within New York City. It is structured into six dimension tables, with the fact table capturing records at the transaction level. Similar to our 311 dataset, we've maintained a transaction-level granularity to ensure consistency across both data marts. One of the challenges we encountered was how to model the contributing factors and vehicle type dimensions. The confusion was from the fact that it was broken into several columns depending on how many factors were involved. A suggestion for the future would be to derive boolean columns from contributing factors and vehicle type dimensions. For example, is Truck involved? -> 0 or 1.



- Integrated Data Warehouse model: This is our final integrated data model, which brings together both datasets through shared dimensions. The two fact tables can be joined using either the date dimension or the location dimension, depending on the type of analysis being performed. The date dimension has been broken down into day, month, and year, allowing for flexible time-based aggregation and trend analysis. For the location dimension, we kept things simple by including only the borough and ZIP code. This strikes a balance between geographic detail and simplicity, making it easier to identify patterns and correlations without overcomplicating the schema.



ETL Process:

For our ETL process, the objective was to create a clean, scalable, and analytics-ready data warehouse for NYC 311 Service Requests and Motor Vehicle Collisions data using GCS, BigQuery, and dbt.

Extraction:

Data was sourced from NYC Open Data Portal in CSV format. The extraction process involved:
- Downloading data to a local drive then uploading it to Google Cloud Storage (GCS), using that as Temporary storage for our large files
- Profiling the data with Python (Pandas, PySpark) and LinuxVM.
- Cleaning data: removing irrelevant columns, filtering for specific complaint types, splitting date fields.

Steps:

- Cleaned the data(dropped unnecessary rows and columns, split the date columns into year, month, and day separately)

  # Filtered the dataset to keep only the rows where the "Complaint Type" is either "Noise - Residential" or "Noise - Vehicle."

```
from pyspark.sql.functions import col

noise_df = sample_df.filter(
    (col("Complaint Type") == "Noise - Residential") | (col("Complaint Type") == "Noise - Vehicle")
)
```

  # Selected only the necessary columns

```
columns_to_keep = [
    "Unique Key",
    "Created Date",
    "Closed Date",
    "Agency",
    "Agency Name",
    "Complaint Type",
    "Descriptor",
    "Incident Zip",
    "City",
    "Borough",
    "Status",
    "Open Data Channel Type"
]
```

```
filtered_df = noise_df.select(columns_to_keep)
```


```
filtered_df = noise_df.select(columns_to_keep)
```


```
filtered_df.printSchema()
```

```
root
 |-- Unique Key: integer (nullable = true)
 |-- Created Date: string (nullable = true)
 |-- Closed Date: string (nullable = true)
 |-- Agency: string (nullable = true)
 |-- Agency Name: string (nullable = true)
 |-- Complaint Type: string (nullable = true)
 |-- Descriptor: string (nullable = true)
 |-- Incident Zip: string (nullable = true)
 |-- City: string (nullable = true)
 |-- Borough: string (nullable = true)
 |-- Status: string (nullable = true)
 |-- Open Data Channel Type: string (nullable = true)
```

Split the date into separate year, month, and day columns

```python
from pyspark.sql.functions import to_timestamp, year, month, dayofmonth

filtered_df = filtered_df.withColumn(
    "Created Date",
    to_timestamp("Created Date", "MM/dd/yyyy hh:mm:ss a")
)
```

```python
filtered_df = filtered_df.withColumn("year", year("Created Date")) \
                         .withColumn("month", month("Created Date")) \
                         .withColumn("day", dayofmonth("Created Date"))
```

```python
filtered_df.select("Created Date", "year", "month", "day").show(5)
```

```
+-------------------+----+-----+---+
|       Created Date|year|month|day|
+-------------------+----+-----+---+
|2025-03-04 01:31:18|2025|    3|  4|
|2025-03-03 16:53:02|2025|    3|  3|
|2025-03-03 21:30:32|2025|    3|  3|
|2025-03-03 22:05:13|2025|    3|  3|
|2025-03-03 02:22:11|2025|    3|  3|
+-------------------+----+-----+---+
only showing top 5 rows
```

- Saved cleaned data into the bucket
- Uploaded cleaned CSVs to GCS (my-school-etl-data-project bucket).
- Loaded into BigQuery tables:
  - raw_311_data
  - crash_data
- Validated schemas (date types, integers, strings)

After this, we moved to the transformation stage

Transformation

Transformation was executed using dbt to:
- Create staging models, to further clean and filter data.
- Build dimension tables for time, location, agency, complaint types, status, contributing factors, and vehicle types.
- Construct fact tables for noise complaints and collision events linked to dimensions.
- Create views for large fact tables to optimize performance and facilitate scalability.

Transformation steps

dbt Setup:

- **Created a dbt project: my_school_etl_project**
- **Connected using a service account.**
- **Ran dbt debug to validate connection.**

Staging Models:

- **stg_311_data: Cleaned and filtered Noise Complaints.**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`
      OPTIONS()
    as (
      WITH source AS (
  SELECT * FROM `my-school-etl-project`.`crash_collision_dataset`.`raw_311_data`
)

SELECT
  unique_key,
  created_date,
  closed_date,
  agency,
  agency_name,
  complaint_type,
  descriptor,
  location_type,
  incident_zip,
  incident_address,
  street_name,
  city,
  borough,
  status,
  open_data_channel_type,
  latitude,
  longitude
FROM source
WHERE complaint_type IS NOT NULL);
```

☐	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
☐	unique_key	INTEGER	NULLABLE	-	-	-	-
☐	created_date	TIMESTAMP	NULLABLE	-	-	-	-
☐	closed_date	TIMESTAMP	NULLABLE	-	-	-	-
☐	agency	STRING	NULLABLE	-	-	-	-
☐	agency_name	STRING	NULLABLE	-	-	-	-
☐	complaint_type	STRING	NULLABLE	-	-	-	-
☐	descriptor	STRING	NULLABLE	-	-	-	-
☐	location_type	STRING	NULLABLE	-	-	-	-
☐	incident_zip	INTEGER	NULLABLE	-	-	-	-
☐	incident_address	STRING	NULLABLE	-	-	-	-
☐	street_name	STRING	NULLABLE	-	-	-	-
☐	city	STRING	NULLABLE	-	-	-	-

- **stg_collision_data: Cleaned collision data, formatted crash_time properly.**

```
create or replace view `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`
OPTIONS() as
SELECT
  collision_id,
  crash_date,
  CASE
    WHEN LENGTH(crash_time) = 4 THEN FORMAT('%02d:%02d:00', SAFE_CAST(SUBSTR(crash_time, 1, 1) AS INT64), SAFE_CAST(SUBSTR(crash_time, 3, 2) AS INT64))
    WHEN LENGTH(crash_time) = 5 THEN FORMAT('%02d:%02d:00', SAFE_CAST(SUBSTR(crash_time, 1, 2) AS INT64), SAFE_CAST(SUBSTR(crash_time, 4, 2) AS INT64))
    WHEN LENGTH(crash_time) = 8 THEN crash_time
    ELSE NULL
  END AS crash_time, borough, zip_code,
  -- Factors and Vehicles
  contributing_factor_vehicle_1,
  contributing_factor_vehicle_2,
  contributing_factor_vehicle_3,
  contributing_factor_vehicle_4,
  contributing_factor_vehicle_5,
  vehicle_type_code1,
  vehicle_type_code2,
  vehicle_type_code_3,
  vehicle_type_code_4,
  vehicle_type_code_5,
  -- Injury and Death counts
  number_of_persons_injured,
  number_of_persons_killed,
  number_of_pedestrians_injured,
  number_of_pedestrians_killed,
  number_of_cyclist_injured,
  number_of_cyclist_killed,
  number_of_motorist_injured,
  number_of_motorist_killed

FROM `my-school-etl-project`.`crash_collision_dataset`.`crash_data`
WHERE collision_id IS NOT NULL;
```

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
collision_id	INTEGER	NULLABLE	-	-	-	-	-
crash_date	TIMESTAMP	NULLABLE	-	-	-	-	-
crash_time	STRING	NULLABLE	-	-	-	-	-
borough	STRING	NULLABLE	-	-	-	-	-
zip_code	INTEGER	NULLABLE	-	-	-	-	-
contributing_factor_vehicle_1	STRING	NULLABLE	-	-	-	-	-
contributing_factor_vehicle_2	STRING	NULLABLE	-	-	-	-	-
contributing_factor_vehicle_3	STRING	NULLABLE	-	-	-	-	-
contributing_factor_vehicle_4	STRING	NULLABLE	-	-	-	-	-
contributing_factor_vehicle_5	STRING	NULLABLE	-	-	-	-	-
vehicle_type_code1	STRING	NULLABLE	-	-	-	-	-
vehicle_type_code2	STRING	NULLABLE	-	-	-	-	-
vehicle_type_code_3	STRING	NULLABLE	-	-	-	-	-
vehicle_type_code_4	STRING	NULLABLE	-	-	-	-	-

Dimension Tables:

- **dim_date**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_date`
OPTIONS()
    as (
WITH dates AS (
  SELECT
    DISTINCT DATE(created_date) AS full_date
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`

  UNION DISTINCT

  SELECT
    DISTINCT DATE(crash_date) AS full_date
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`
)

SELECT
  FORMAT_TIMESTAMP('%Y%m%d', TIMESTAMP(full_date)) AS date_dimension_id,
  full_date,
  EXTRACT(YEAR FROM full_date) AS year,
  FORMAT_DATE('%b', full_date) AS month_name
FROM dates
WHERE full_date IS NOT NULL
ORDER BY full_date
    );
```

Filter Enter property name or value

	Field name	Type	Mode
☐	date_dimension_id	STRING	NULLABLE
☐	full_date	DATE	NULLABLE
☐	year	INTEGER	NULLABLE
☐	month_name	STRING	NULLABLE

- **dim_time**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_time`
OPTIONS()
    as (
 WITH times AS (
SELECT DISTINCT crash_time
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`
  WHERE crash_time IS NOT NULL

  UNION DISTINCT

  SELECT DISTINCT FORMAT_TIMESTAMP('%H:%M:%S', created_date) AS crash_time
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`
  WHERE created_date IS NOT NULL
)

SELECT
  CONCAT(
    LPAD(SPLIT(crash_time, ':')[SAFE_OFFSET(0)], 2, '0'),
    LPAD(SPLIT(crash_time, ':')[SAFE_OFFSET(1)], 2, '0')
  ) AS time_dimension_id,
  crash_time AS full_time,
  SPLIT(crash_time, ':')[SAFE_OFFSET(0)] AS hour,
  CASE
    WHEN SAFE_CAST(SPLIT(crash_time, ':')[SAFE_OFFSET(0)] AS INT64) BETWEEN 5 AND 11 THEN 'Morning'
    WHEN SAFE_CAST(SPLIT(crash_time, ':')[SAFE_OFFSET(0)] AS INT64) BETWEEN 12 AND 16 THEN 'Afternoon'
    WHEN SAFE_CAST(SPLIT(crash_time, ':')[SAFE_OFFSET(0)] AS INT64) BETWEEN 17 AND 20 THEN 'Evening'
    ELSE 'Night'
  END AS time_of_day
FROM times
ORDER BY full_time
    );
```

Field name
time_dimension_id
full_time
hour
time_of_day

- **dim_location**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_location`
    OPTIONS()
    as (
 WITH locations AS (
  SELECT DISTINCT
    CONCAT(CAST(incident_zip AS STRING), '_', LOWER(borough)) AS location_dimension_id,
    borough,
    incident_zip AS zipcode
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`
  WHERE borough IS NOT NULL AND incident_zip IS NOT NULL

  UNION DISTINCT

  -- From Collision Data
  SELECT DISTINCT
    CONCAT(CAST(zip_code AS STRING), '_', LOWER(borough)) AS location_dimension_id,
    borough,
    zip_code AS zipcode
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`
  WHERE borough IS NOT NULL AND zip_code IS NOT NULL
)

SELECT
  location_dimension_id, borough,
  zipcode
FROM locations
ORDER BY borough, zipcode
    );
```

Field name
location_dimension_id
borough
zipcode

- **dim_agency**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_agency`
    OPTIONS()
    as (
WITH agencies AS (
  SELECT DISTINCT
    agency,
    agency_name
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`
  WHERE agency IS NOT NULL
)
SELECT
  ROW_NUMBER() OVER (ORDER BY agency) AS agency_dimension_id,  -- Unique numeric ID
  agency AS agency_acronym,
  agency_name
FROM agencies
ORDER BY agency_dimension_id
    );
```

Field name
agency_dimension_id
agency_acronym
agency_name

- **dim_complaint_type**

```sql
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_complaint_type`
    OPTIONS()
    as (
WITH complaint_types AS (
  SELECT DISTINCT
    complaint_type,
    descriptor
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`
  WHERE complaint_type IS NOT NULL
)
SELECT
  ROW_NUMBER() OVER (ORDER BY complaint_type) AS complaint_type_dimension_id, complaint_type,
  descriptor AS complaint_description
FROM complaint_types
ORDER BY complaint_type_dimension_id
    );
```

	Field name
☐	complaint_type_dimension_id
☐	complaint_type
☐	complaint_description

- **dim_complaint_status**

```sql
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_complaint_status`
    OPTIONS()
    as (
WITH statuses AS (

  SELECT DISTINCT
    status,
    open_data_channel_type
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`
  WHERE status IS NOT NULL

)

SELECT
  ROW_NUMBER() OVER (ORDER BY status) AS complaint_status_dimension_id, status,
  open_data_channel_type
FROM statuses
ORDER BY complaint_status_dimension_id
    );
```

Filter Enter property name or value

	Field name
☐	complaint_status_dimension_id
☐	status
☐	open_data_channel_type

- **dim_contributing_factors**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_contributing_factors`
    OPTIONS()
    as (
WITH factors AS (
  SELECT DISTINCT
    contributing_factor_vehicle_1,
    contributing_factor_vehicle_2,
    contributing_factor_vehicle_3,
    contributing_factor_vehicle_4,
    contributing_factor_vehicle_5
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`
)
SELECT
  ROW_NUMBER() OVER () AS contributing_factors_id,    -- Auto ID
  contributing_factor_vehicle_1,
  contributing_factor_vehicle_2,
  contributing_factor_vehicle_3,
  contributing_factor_vehicle_4,
  contributing_factor_vehicle_5
FROM factors
    );
```

☐	contributing_factors_id
☐	contributing_factor_vehicle_1
☐	contributing_factor_vehicle_2
☐	contributing_factor_vehicle_3
☐	contributing_factor_vehicle_4
☐	contributing_factor_vehicle_5

- **dim_vehicle_type**

```
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`dim_vehicle_type`
    OPTIONS()
    as (


WITH vehicles AS (

  SELECT DISTINCT
    vehicle_type_code1,
    vehicle_type_code2,
    vehicle_type_code_3,
    vehicle_type_code_4,
    vehicle_type_code_5
  FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`
)

SELECT
  ROW_NUMBER() OVER () AS vehicle_dimension_id,
  vehicle_type_code1,
  vehicle_type_code2,
  vehicle_type_code_3,
  vehicle_type_code_4,
  vehicle_type_code_5
FROM vehicles
    );
```

☐	vehicle_dimension_id
☐	vehicle_type_code1
☐	vehicle_type_code2
☐	vehicle_type_code_3
☐	vehicle_type_code_4
☐	vehicle_type_code_5

Fact Tables:

- **fact_noise_complaints: Linked to dimensions, 1 record = 1 complaint.**

```sql
create or replace table `my-school-etl-project`.`crash_collision_dataset`.`fact_noise_complaints`
   OPTIONS()
    as (
WITH source AS (
    SELECT *
    FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_311_data`)
SELECT
    src.unique_key,
    date_dim.date_dimension_id,
    time_dim.time_dimension_id,
    agency_dim.agency_dimension_id,
    complaint_type_dim.complaint_type_dimension_id,
    location_dim.location_dimension_id,
    status_dim.complaint_status_dimension_id,
    1 AS number_of_complaints,  -- Each record is one complaint
    DATE(src.created_date) AS created_date,
    DATE(src.closed_date) AS closed_date,
    src.complaint_type,
    src.borough,
    src.city
FROM source AS src
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_date` AS date_dim
  ON DATE(src.created_date) = date_dim.full_date
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_time` AS time_dim
  ON FORMAT_TIME('%H:%M:%S', TIME(src.created_date)) = time_dim.full_time
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_agency` AS agency_dim
  ON src.agency = agency_dim.agency_acronym -- JOIN to Agency Dimension
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_complaint_type` AS complaint_type_dim
  ON src.complaint_type = complaint_type_dim.complaint_type
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_location` AS location_dim
  ON src.borough = location_dim.borough
  AND src.incident_zip = location_dim.zipcode
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_complaint_status` AS status_dim
  ON src.status = status_dim.status);
```

Field name	Type	Mode
unique_key	INTEGER	NULLABLE
date_dimension_id	STRING	NULLABLE
time_dimension_id	STRING	NULLABLE
agency_dimension_id	INTEGER	NULLABLE
complaint_type_dimension_id	INTEGER	NULLABLE
location_dimension_id	STRING	NULLABLE
complaint_status_dimension_id	INTEGER	NULLABLE
number_of_complaints	INTEGER	NULLABLE
created_date	DATE	NULLABLE
closed_date	DATE	NULLABLE
complaint_type	STRING	NULLABLE

- **fact_collision_events: Linked to time, date, location, vehicle, factors.**

```sql
create or replace view `my-school-etl-project`.`crash_collision_dataset`.`fact_collision_events`
  OPTIONS()
  as
WITH source AS (
    SELECT *
    FROM `my-school-etl-project`.`crash_collision_dataset`.`stg_collision_data`)
SELECT src.collision_id, date_dim.date_dimension_id,
    time_dim.time_dimension_id,
    location_dim.location_dimension_id,
    factor_dim.contributing_factors_id,
    vehicle_dim.vehicle_dimension_id,
    src.number_of_persons_injured,
    src.number_of_persons_killed,
    src.number_of_pedestrians_injured,
    src.number_of_pedestrians_killed,
    src.number_of_cyclist_injured,
    src.number_of_cyclist_killed,
    src.number_of_motorist_injured,
    src.number_of_motorist_killed,
    DATE(src.crash_date) AS crash_date,
    src.borough,
    src.zip_code
FROM source AS src
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_date` AS date_dim
  ON DATE(src.crash_date) = date_dim.full_date
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_time` AS time_dim
  ON src.crash_time = time_dim.full_time
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_location` AS location_dim
  ON src.borough = location_dim.borough
  AND src.zip_code = location_dim.zipcode
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_contributing_factors` AS factor_dim
  ON src.contributing_factor_vehicle_1 = factor_dim.contributing_factor_vehicle_1
LEFT JOIN `my-school-etl-project`.`crash_collision_dataset`.`dim_vehicle_type` AS vehicle_dim
  ON src.vehicle_type_code1 = vehicle_dim.vehicle_type_code1;
```

	Field name	Type	Mode
☐	collision_id	INTEGER	NULLABLE
☐	date_dimension_id	STRING	NULLABLE
☐	time_dimension_id	STRING	NULLABLE
☐	location_dimension_id	STRING	NULLABLE
☐	contributing_factors_id	INTEGER	NULLABLE
☐	vehicle_dimension_id	INTEGER	NULLABLE
☐	number_of_persons_injured	INTEGER	NULLABLE
☐	number_of_persons_killed	INTEGER	NULLABLE
☐	number_of_pedestrians_injured	INTEGER	NULLABLE
☐	number_of_pedestrians_killed	INTEGER	NULLABLE
☐	number_of_cyclist_injured	INTEGER	NULLABLE

Both fact tables initially materialized as views to improve performance.

Once the transformation stage was complete, we now moved to deploying our models to big Query. However, prior to that, we made sure our models were functional and accurate.

Testing, Validation & Documentation

Quality assurance was ensured using dbt tests, including not_null and unique constraints. Issues with duplicates and nulls were resolved before deployment. We then used dbt docs to generate documentation.

Loading

At the loading stage, final tables were deployed into BigQuery under 'crash_collision_dataset'. Fact tables were materialized as views due to data volume. Tables were structured into Source, Staging, Dimension, Fact, and Aggregation layers for clarity.

- Fact and dimension tables successfully deployed into BigQuery.
- Fact tables set to view for collision events (due to size).

- Tables organized under dataset: crash_collision_dataset

Once the loading phase was complete, we had a data warehouse that was scalable, modular, and analytics-ready. With proper documentation and rigorous testing, it supports further integration with BI tools like Tableau and Looker Studio

Final Warehouse Structure

The data warehouse structure includes:

- Source: raw_311_data, crash_data

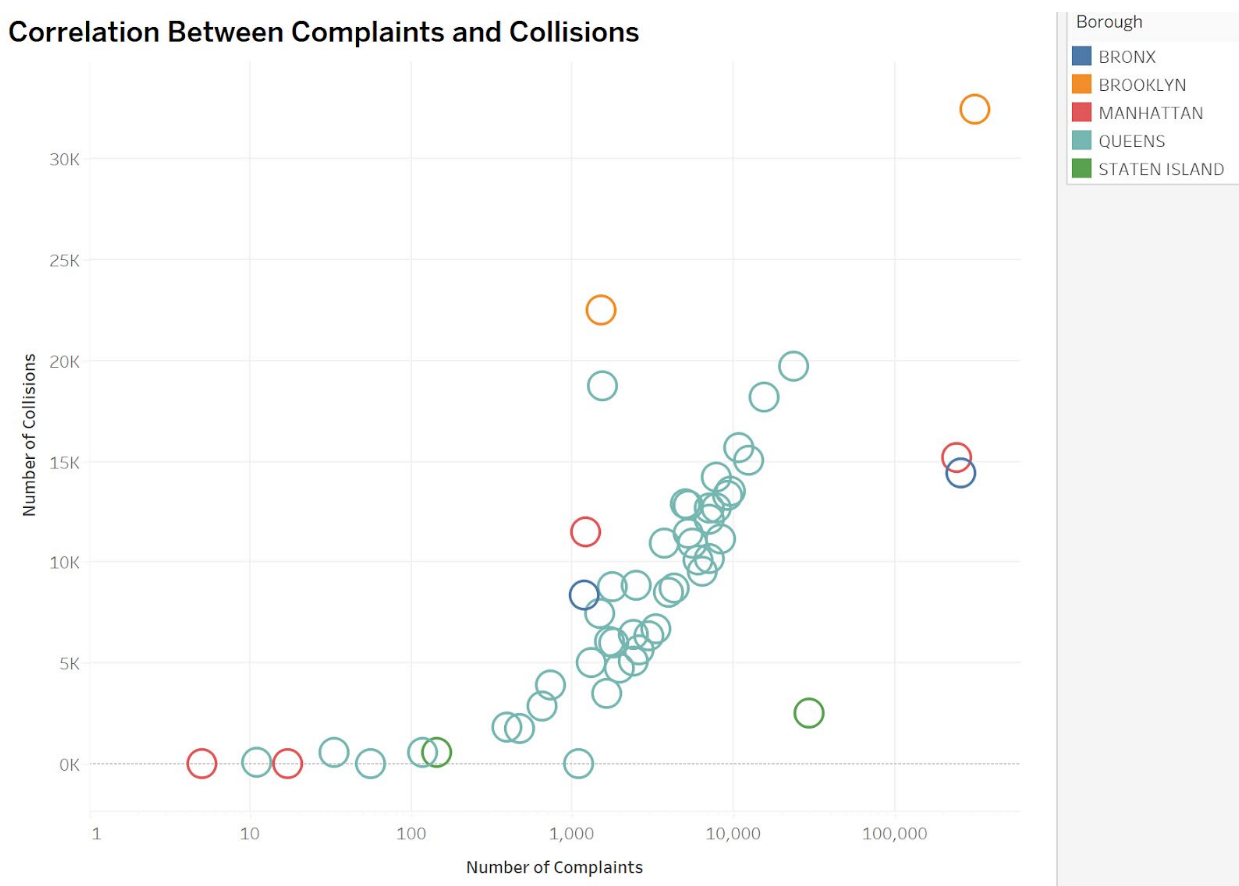
- Staging: stg_311_data, stg_collision_data

- Dimension: dim_date, dim_time, dim_location, dim_agency, dim_complaint_type, dim_complaint_status, dim_contributing_factors, dim_vehicle_type

- Fact: fact_noise_complaints, fact_collision_events

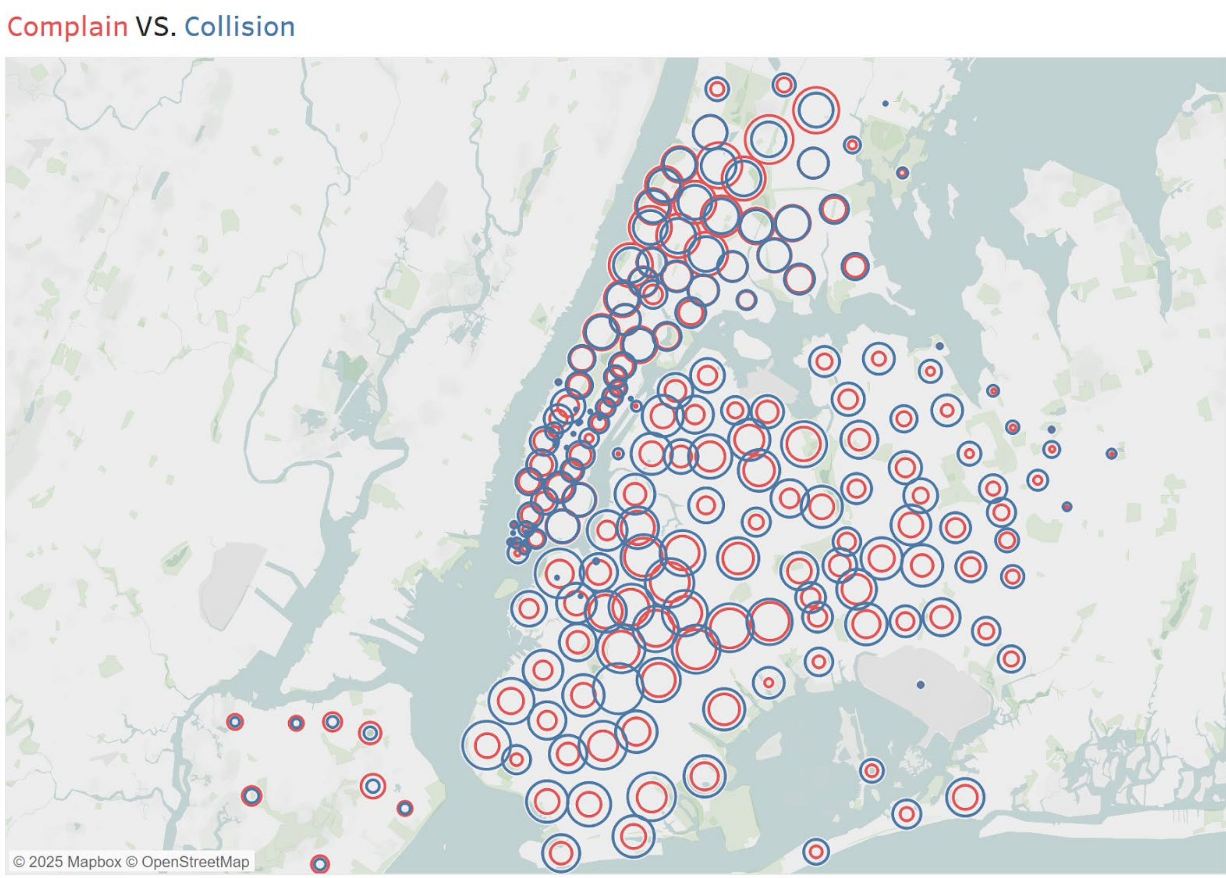
KPI visualizations:

1. Correlation Between Noise Complaints and Collision Incidents



This scatter plot illustrates the correlation between vehicle collision incidents and noise complaints across different cities, with each dot representing a city and colored by borough. The x-axis indicates the number of complaints, while the y-axis shows the number of collisions. Overall, cities with a higher number of collisions tend to report more noise complaints, suggesting that collisions may be a contributing factor to increased noise-related complaints. While there is a general pattern, the relationship isn't perfectly consistent across all cities.

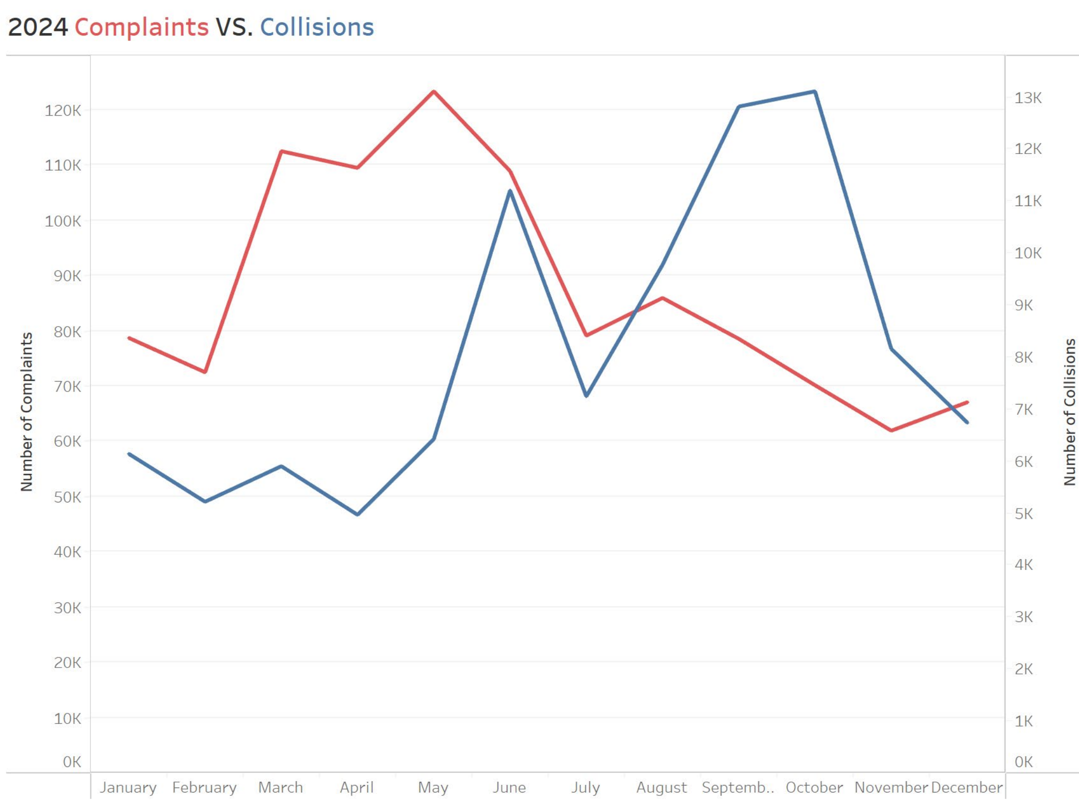
2. Spatial Overlap of Complaints and Collisions

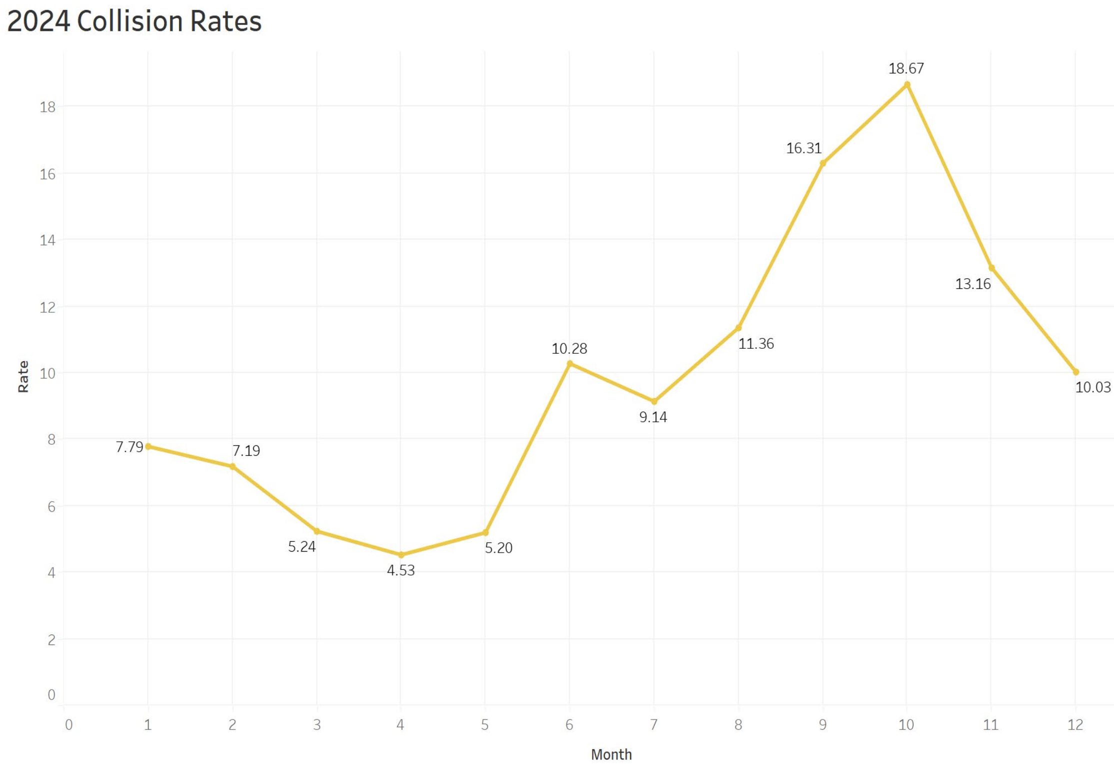


This map visualizes the spatial overlap between vehicle noise complaints (in red) and collision incidents (in blue) across New York

City. Each location is represented by overlapping circles, allowing us to easily spot areas where both complaints and collisions are concentrated. A high density of red and blue circles appearing together suggests that certain neighborhoods experience both frequent collisions and related noise complaints. This spatial clustering supports the idea that traffic collisions may contribute directly to noise complaints in those areas.

3. Impact of Traffic Complaints on Collision Rates





We focused on data from 2024 instead of the full range from 2010 to 2025 to provide a more current and relevant analysis. The first chart compares monthly traffic complaints (in red) with the number of collisions (in blue). To measure the impact of complaints on collision trends, we calculated the collision rate using the formula:

(Number of Collisions ÷ Number of Complaints) × 100

The second chart shows how this rate changed month by month. While complaints decreased after June, the collision rate increased in the later months, especially in October, showing that a lower number of complaints did not necessarily align with fewer collisions.

Conclusion

A) The tools and software we used to coordinate and manage the project as well as carry out the programming tasks were as follows:

a) Zoom: To host meetings and discuss how to split up the project's responsibilities.
b) Google Drive: We created a folder holding various documents such as Google Docs, Google Collabs, datasets, etc.
c) Dbt (Data BuildTool): was used to clean, structure, test, and document your data transformations in a modular and repeatable way—turning raw NYC data into reliable, query-ready models in BigQuery
d) BigQuery: was used as cloud data warehouse to store, transform, optimize, and serve your project's data—acting as the foundation for all ETL steps beyond extraction
e) Google Collabs: Originally used in the extract process, but sharing the document was complicated.
f) Python: Used locally on members' computers for the Extract stage. Python was used to clean the data as well.
g) Tableau: is widely used in data visualization and data analytics to uncover patterns, track KPIs, and support data-driven decision-making.
h) PySpark: is used for data cleaning, transformation, and exploratory analysis

B) Since this was the first time anyone in our group has ever done data warehousing, this project was challenging for all of us. Thankfully, we didn't have any difficulties working with each other and having our usual zoom meetings on Saturdays at 3PM reinforced our cohesion. The most challenging part was building the ETL/ELT pipeline because this process required a learning curve. The easiest part was defining the requirements and choosing what technologies to use because we pretty much agreed on everything. If we had to do it again, learning how to implement DAG into our project would've probably made this project easier.

C) Since we've established a strong correlation between noise complaints and vehicle crashes, we can show policymakers where they can allocate resources to launch a two-pronged campaign against noise pollution and vehicle crashes. Noise complaints can predict vehicle crashes before they occur, so we suggest streaming live 311 noise complaint data into a real-time dashboard. This will allow policymakers to see potential hotspots of vehicle crashes and make actionable insights on how to manage traffic in certain neighborhoods. Our research can also be used where the policymakers can raise public safety awareness campaigns.

D) Although we've established a strong correlation between noise complaints and vehicle crashes, this is only the beginning in the fight for vehicle safety. This project was never intended to be a comprehensive be-all, end-all solution; it was more like the growing pain stage in developing our data warehousing skills. This project was a challenging, but also rewarding experience where everyone learned and will use these skills in their future endeavors.

References

1. **311 Service Requests Dataset** – Focusing on noise complaints related to vehicles and traffic
2. **Motor Vehicle Collisions (Crashes) Dataset** – Containing detailed records of reported traffic accidents, including locations, time, and contributing factors.