The Correlation Between Traffic Violations and Motor Vehicle Collisions in NYC

CIS 4400 - CMWA

Cristopher Zuleta - [cristopher.zuleta@baruchmail.cuny.edu](mailto:christopher.zuleta@baruchmail.cuny.edu)

Steven Cai - [steven.cai@baruchmail.cuny.edu](mailto:steven.cai@baruchmail.cuny.edu)

Joshua Chirinos - [joshua.chirinos@baruchmail.cuny.edu](mailto:joshua.chirinos@baruchmail.cuny.edu)

Muhammad Megahed - [muhammad.megahed@baruchmail.cuny.edu](mailto:muhammad.megahed@baruchmail.cuny.edu)

Nazmus Sakib - [nazmus.sakib@baruchmail.cuny.edu](mailto:nazmus.sakib@baruchmail.cuny.edu)

Baruch College

Fall 2022

**Project Description:**

Our group decided to find a correlation between the status of the city streets and the occurrence of vehicle collisions. For this we will utilize fields from the 311 database such as: Street Condition, Street Light Condition, Traffic Signal Condition.

For this we will integrate the daily database of Motor Vehicles Collisions - Crashes to establish a correlation and a trend in behaviors/complaints. For example, we will try to establish correlations between inebriated drivers and the severity of any traffic violations. The goal of this project is to find possible solutions, based on the nature of the complaint and collision.

**Datasets:**

Motor Vehicle Collisions - Crashes

<https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95>

311 Service Requests from 2010 to Present

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

**KPIs:**

* Number of accidents per Zip Code.
* Number of accidents per Street.
* Number of complaints per Borough.
* Number of fatal injuries
* Number of non-fatal injuries
* Number of fatal collisions per factor
* Number of non-fatal collisions per factor
* Number of accidents by contributing factor
* Number of accidents by Latitude and Longitude
* Number of accidents by hour of the day
* Number of people injured by accident
* Number of people killed by accident

**Milestone 2:**

Diagram, schematic

Description automatically generated

**Diagram, schematic

Description automatically generated**

**Milestone #3:**

Updated Models:

* 311 Model:

Diagram

Description automatically generated

* Crash Model:

Diagram, schematic

Description automatically generated

* Combined Model:

**Diagram

Description automatically generated**

**Milestone #4:**

ETL Tool and Target DBMS

To implement our project, we will be using Google BigQuery and for our ETL tool we will be using DBT. We came to this decision after a group discussion and agree that all members feel comfortable and capable of contributing to the project while working on this platform.

**Milestone #5:** ETL Programming

Data Profiling using Python for the 311\_data and collisions\_data between 2019 and 2022:

Graphical user interface

Description automatically generated

* Had an error with the 311 data but upon checking the report file, everything seemed to look good:

**311\_data: **

Graphical user interface, application

Description automatically generated

* Alerts varied between High Cardinality, Constant, High correlation, Missing, Unsupported, and one alert of Unique

Graphical user interface, text, application

Description automatically generated

**Collisions\_data: **

Graphical user interface, text

Description automatically generated

* Alerts varied between High Cardinality, High correlation, Missing, Uniform, Zeros and one alert of Unique

Graphical user interface, text, application

Description automatically generated

**In DBT:**

* Created 2 folders within the models’ folder:
  + core: for the data and the fact tables.
  + Staging: for the dimensions
* Created 3 folders within the staging folder:
  + 311: for the 311 dimensions
  + collisions: for the collision dimensions
  + shared\_models: for the shared dimensions (date, time, location)

**core:**

* 311\_data.sql query and fixing:

with all\_data as (

  select

    unique\_key,

    PARSE\_DATETIME("%m/%d/%Y %I:%M:%S %p", TRIM(created\_date)) AS created\_date,

    PARSE\_DATETIME("%m/%d/%Y %I:%M:%S %p", TRIM(closed\_date)) AS closed\_date,

    agency,

    agency\_name,

    complaint\_type,

    descriptor,

    location\_type,

    incident\_zip,

    incident\_address,

    street\_name,

    cross\_street\_1,

    cross\_street\_2,

    intersection\_street\_1,

    intersection\_street\_2,

    address\_type,

    city,

    landmark,

    facility\_type,

    status,

    PARSE\_DATETIME("%m/%d/%Y %I:%M:%S %p", TRIM(due\_date)) AS due\_date,

    resolution\_description,

    PARSE\_DATETIME("%m/%d/%Y %I:%M:%S %p", TRIM(resolution\_action\_updated\_date)) AS resolution\_action\_updated\_date,

    community\_board,

    bbl,

    borough,

    x\_coordinate,

    y\_coordinate,

    open\_data\_channel\_type,

    park\_facility\_name,

    park\_borough,

    vehicle\_type,

    taxi\_company\_borough,

    taxi\_pick\_up\_location,

    bridge\_highway\_name,

    bridge\_highway\_direction,

    road\_ramp,

    bridge\_highway\_segment,

    latitude,

    longitude,

    location

  from

      {{ source ('project4400team10', '311model')}}

)

select

  unique\_key,

  cast(FORMAT\_DATE("%Y-%m-%d", created\_date) as date) as created\_date,

  format\_timestamp("%H:%M:%S", created\_date) as created\_time,

  cast(FORMAT\_DATE("%Y-%m-%d", closed\_date) as date) as closed\_date,

  format\_timestamp("%H:%M:%S", closed\_date) as closed\_time,

  coalesce(agency, "Not Available") as agency,

  coalesce(agency\_name, "Not Available")  as agency\_name,

  CAST(complaint\_type AS string) as complaint\_type,

  coalesce(descriptor, "Unknown") as descriptor,

  coalesce(location\_type, "Not Available") as location\_type,

  coalesce(incident\_zip, "Not Available") as incident\_zip,

  coalesce(incident\_address, "Not Available") as incident\_address,

  street\_name,

  cross\_street\_1,

  cross\_street\_2,

  concat(coalesce(street\_name, "Unknown"), " and ", coalesce(cross\_street\_1, "Unknown")) as intersecting\_streets,

  intersection\_street\_1,

  intersection\_street\_2,

  address\_type,

  coalesce(city, "Not Available") as city,

  landmark,

  facility\_type,

  status,

  cast(FORMAT\_DATE("%Y-%m-%d", due\_date) as date) as due\_date,

  format\_timestamp("%H:%M:%S", due\_date) as due\_time,

  resolution\_description,

  cast(FORMAT\_DATE("%Y-%m-%d", resolution\_action\_updated\_date) as date) as resolution\_action\_updated\_date,

  format\_timestamp("%H:%M:%S", resolution\_action\_updated\_date) as resolution\_action\_updated\_time,

  community\_board,

  bbl,

  coalesce(borough, "Not Available") as borough,

  x\_coordinate,

  y\_coordinate,

  open\_data\_channel\_type,

  park\_facility\_name,

  park\_borough,

  vehicle\_type,

  taxi\_company\_borough,

  taxi\_pick\_up\_location,

  bridge\_highway\_name,

  bridge\_highway\_direction,

  road\_ramp,

  bridge\_highway\_segment,

  coalesce(round(latitude, 6), 0.0) as latitude,

  coalesce(round(longitude, 6), 0.0) as longitude,

  location

from all\_data

WHERE created\_date BETWEEN '2019-07-01' AND '2022-11-28'

AND complaint\_type = "Street Condition"

AND latitude IS NOT NULL

AND longitude IS NOT NULL

* Collisions\_data.sql query:

with corrected as (

  select

    crash\_date,

    case

      when crash\_time like '\_:\_\_' then concat('0', crash\_time)

      else crash\_time

      end as crash\_time,

    borough,

    zip\_code,

    latitude,

    longitude,

    location,

    on\_street\_name,

    cross\_street\_name,

    concat(coalesce(on\_street\_name, "Unknown"), " and ", coalesce(cross\_street\_name, "Unknown")) as intersecting\_streets,

    off\_street\_name,

    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,

    contributing\_factor\_vehicle\_1,

    contributing\_factor\_vehicle\_2,

    contributing\_factor\_vehicle\_3,

    contributing\_factor\_vehicle\_4,

    contributing\_factor\_vehicle\_5,

    collision\_id,

    vehicle\_type\_code\_1,

    vehicle\_type\_code\_2,

    vehicle\_type\_code\_3,

    vehicle\_type\_code\_4,

    vehicle\_type\_code\_5

  from

    {{ source ('project4400team10', 'collisions')}}

  WHERE

    latitude != 0

    AND latitude IS NOT NULL

    AND longitude != 0

    AND longitude IS NOT NULL

    AND borough IS NOT NULL

    AND zip\_code IS NOT NULL

)

select

  crash\_date,

  CAST(PARSE\_TIME("%H:%M", crash\_time) AS STRING) as crash\_time,

  borough,

  cast(zip\_code as STRING) as zip\_code,

  round(latitude, 6) as latitude,

  round(longitude, 6) as longitude,

  location,

  on\_street\_name,

  cross\_street\_name,

  intersecting\_streets,

  off\_street\_name,

  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,

  coalesce(CONTRIBUTING\_FACTOR\_VEHICLE\_1, "Unspecified") as CONTRIBUTING\_FACTOR\_VEHICLE\_1,

  coalesce(CONTRIBUTING\_FACTOR\_VEHICLE\_2, "Unspecified") as CONTRIBUTING\_FACTOR\_VEHICLE\_2,

  contributing\_factor\_vehicle\_3,

  contributing\_factor\_vehicle\_4,

  contributing\_factor\_vehicle\_5,

  collision\_id,

  coalesce(vehicle\_type\_code\_1, "Unknown") as vehicle\_type\_code\_1,

  coalesce(vehicle\_type\_code\_2, "Unknown") as vehicle\_type\_code\_2,

  vehicle\_type\_code\_3,

  vehicle\_type\_code\_4,

  vehicle\_type\_code\_5

from

  corrected

WHERE crash\_date BETWEEN '2019-07-01' AND '2022-11-28'

**staging:**

* **311:**
  + 311\_location.sql

with location\_dim as (

  select distinct

    location\_type,

    incident\_address,

    incident\_zip,

    intersecting\_streets,

    city,

    borough,

    latitude,

    longitude

  from

    {{ ref ('311\_data')}}

)

select

  row\_number() over () as location\_dim\_id,

  location\_type,

  incident\_address,

  incident\_zip,

  intersecting\_streets,

  city,

  borough,

  latitude,

  longitude

from

  location\_dim

* + agency\_type.sql

with agency\_dim as (

  select distinct

    agency,

    agency\_name

  from

    {{ ref ('311\_data')}}

)

select

  row\_number() over () as agency\_type\_dim\_id,

  agency,

  agency\_name

from

  agency\_dim

* + channel\_type.sql

with channel\_dim as (

  select distinct

    open\_data\_channel\_type as open\_data\_channel\_type

  from

    {{ ref ('311\_data')}}

)

select

  row\_number() over () as channel\_type\_dim\_id,

  open\_data\_channel\_type,

from

  channel\_dim

* + complaint\_type.sql

with complaint as (

    select distinct

        complaint\_type,

        descriptor

    from {{ ref ('311\_data')}}

)

select

row\_number() over () as complaint\_type\_dim\_id,

complaint\_type,

descriptor

from complaint

* + status.sql

with status\_dimension as (

  select distinct

    status as status

  from

    {{ ref ('311\_data')}}

)

select

  row\_number() over () as status\_dim\_id,

  status,

from

  status\_dimension

* **collisions:**
  + collision.sql

with collision\_dim as (

  select distinct

    vehicle\_type\_code\_1,

    vehicle\_type\_code\_2

  from

    {{ ref ('collisions\_data')}}

)

select

  row\_number() over () as collision\_dim\_id,

  \*

from

  collision\_dim

* + collisions\_location.sql

with location\_dim as (

  select distinct

    intersecting\_streets,

    borough,

    zip\_code,

    latitude,

    longitude

  from

    {{ ref ('collisions\_data')}}

)

select

  row\_number() over () as location\_dim\_id,

  \*

from

  location\_dim

* + factors.sql

with factors\_dim as (

  select distinct

    CONTRIBUTING\_FACTOR\_VEHICLE\_1,

    CONTRIBUTING\_FACTOR\_VEHICLE\_2

  from

    {{ ref ('collisions\_data')}}

)

select

  row\_number() over () as factors\_dim\_id,

  \*

from

  factors\_dim

* + fatalities.sql

with fatal\_dim as (

  select distinct

    number\_of\_persons\_killed,

    number\_of\_pedestrians\_killed,

    number\_of\_cyclist\_killed,

    number\_of\_motorist\_killed

  from

    {{ ref ('collisions\_data')}}

)

select

  row\_number() over () as fatalities\_dim\_id,

  \*

from

  fatal\_dim

* + injuries.sql

with injur\_dim as (

  select distinct

    number\_of\_persons\_injured,

    number\_of\_pedestrians\_injured,

    number\_of\_cyclist\_injured,

    number\_of\_motorist\_injured

  from

    {{ ref ('collisions\_data')}}

)

select

  row\_number() over () as injuries\_dim\_id,

  \*

from

  injur\_dim

* **shared\_models**
  + date\_dimension.sql (day as the lowest grain using “generate\_date\_array”)

SELECT

  row\_number() over () as date\_dim\_id,

  d as full\_date,

  EXTRACT(YEAR FROM d) AS year,

  EXTRACT(MONTH FROM d) AS month,

  EXTRACT(DAY FROM d) AS day,

FROM (

  SELECT

    \*

  FROM

    UNNEST(GENERATE\_DATE\_ARRAY('2016-10-01', '2022-11-28', INTERVAL 1 DAY)) AS d )

* + time\_dimension.sql (second as the lowest grain using “generate\_timestamp\_array”)

SELECT

  row\_number() over () as time\_dim\_id,

  format\_timestamp("%H:%M:%S", t) as full\_time,

  EXTRACT(HOUR FROM t) AS hour,

  EXTRACT(MINUTE FROM t) AS minute,

  EXTRACT(SECOND FROM t) AS second

FROM (

  SELECT

    \*

  FROM

    UNNEST(GENERATE\_TIMESTAMP\_ARRAY('2010-01-01', '2010-01-02', INTERVAL 1 SECOND)) AS t )

* + location\_dimension.sql (generated from the two staging locations made earlier)

with complaint\_location as (

    select \*

    from {{ref ("311\_location")}}

),

collision\_location as (

    select \*

    from {{ref ("collisions\_location")}}

),

location\_dimension as (

    select

        location\_type,

        coalesce(complaint\_location.intersecting\_streets, collision\_location.intersecting\_streets, "Not Applicable") as intersecting\_streets,

        coalesce(complaint\_location.incident\_zip, collision\_location.zip\_code) as zip\_code,

        coalesce(complaint\_location.borough, collision\_location.borough) as borough,

        coalesce(complaint\_location.latitude, collision\_location.latitude) as latitude,

        coalesce(complaint\_location.longitude, collision\_location.longitude) as longitude

    from

        complaint\_location

        full join collision\_location

        on complaint\_location.latitude = collision\_location.latitude

        and complaint\_location.longitude = collision\_location.longitude

)

select

    row\_number() over () as location\_dim\_id,

    \*

from

    location\_dimension

* **core (fact tables)**
  + fct\_311.sql with screenshot of preview

{{ config(

    materialized='table'

)}}

with location as (

    select \*

    from {{ ref ("location\_dimension")}}

),

agency as (

    select \*

    from {{ ref ("agency\_type")}}

),

channel as (

    select \*

    from {{ ref ("channel\_type")}}

),

complaint as (

    select \*

    from {{ ref ("311\_data")}}

),

complaint\_types as (

    select \*

    from {{ ref ("complaint\_type")}}

),

status as (

    select \*

    from {{ ref ("status")}}

),

dates as (

    select \*

    from {{ ref ("date\_dimension")}}

),

times as (

    select \*

    from {{ ref("time\_dimension")}}

),

fact\_table as (

    select

        complaint.unique\_key,

        complaint\_types.complaint\_type\_dim\_id,

        agency.agency\_type\_dim\_id,

        dates.date\_dim\_id,

        times.time\_dim\_id,

        location.location\_dim\_id,

        channel.channel\_type\_dim\_id,

        status.status\_dim\_id

    from

        complaint

    left join agency on complaint.agency = agency.agency

    and complaint.agency\_name = agency.agency\_name

    left join complaint\_types on complaint.complaint\_type = complaint\_types.complaint\_type

    and complaint.descriptor = complaint\_types.descriptor

    left join dates on complaint.created\_date = dates.full\_date

    left join times on complaint.created\_time = times.full\_time

    left join location on complaint.intersecting\_streets = location.intersecting\_streets

    and complaint.incident\_zip = location.zip\_code

    and complaint.latitude = location.latitude

    and complaint.longitude = location.longitude

    and complaint.location\_type = location.location\_type

    left join channel on complaint.open\_data\_channel\_type = channel.open\_data\_channel\_type

    left join status on complaint.status = status.status

)

select \*

from fact\_table

Graphical user interface, application

Description automatically generated

* + fct\_collisions.sql with screenshot of preview

{{ config(

    materialized='table'

)}}

with collisions as (

    select \*

    from {{ ref ("collisions\_data")}}

),

collision as (

    select \*

    from {{ ref ("collision")}}

),

location as (

    select \*

    from {{ ref ("location\_dimension")}}

),

factors as (

    select \*

    from {{ ref ("factors")}}

),

fatalities as (

    select \*

    from {{ ref ("fatalities")}}

),

injuries as (

    select \*

    from {{ ref ("injuries")}}

),

dates as (

    select \*

    from {{ ref ("date\_dimension")}}

),

times as (

    select \*

    from {{ ref ("time\_dimension")}}

),

fact\_table as (

    select

        collisions.collision\_id,

        collision.collision\_dim\_id,

        location.location\_dim\_id,

        factors.factors\_dim\_id,

        fatalities.fatalities\_dim\_id,

        injuries.injuries\_dim\_id,

        dates.date\_dim\_id,

        times.time\_dim\_id

    from

        collisions

    left join collision on collisions.vehicle\_type\_code\_1 = collision.vehicle\_type\_code\_1

    and collisions.vehicle\_type\_code\_2 = collision.vehicle\_type\_code\_2

    left join location on collisions.intersecting\_streets = location.intersecting\_streets

    and collisions.zip\_code = location.zip\_code

    and collisions.latitude = location.latitude

    and collisions.longitude = location.longitude

    left join factors on collisions.CONTRIBUTING\_FACTOR\_VEHICLE\_1 = factors.CONTRIBUTING\_FACTOR\_VEHICLE\_1

    and collisions.CONTRIBUTING\_FACTOR\_VEHICLE\_2 = factors.CONTRIBUTING\_FACTOR\_VEHICLE\_2

    left join fatalities on collisions.number\_of\_persons\_killed = fatalities.number\_of\_persons\_killed

    and collisions.number\_of\_pedestrians\_killed = fatalities.number\_of\_pedestrians\_killed

    and collisions.number\_of\_cyclist\_killed = fatalities.number\_of\_cyclist\_killed

    and collisions.number\_of\_motorist\_killed = fatalities.number\_of\_motorist\_killed

    left join injuries on collisions.number\_of\_persons\_injured = injuries.number\_of\_persons\_injured

    and collisions.number\_of\_pedestrians\_injured = injuries.number\_of\_pedestrians\_injured

    and collisions.number\_of\_cyclist\_injured = injuries.number\_of\_cyclist\_injured

    and collisions.number\_of\_motorist\_injured = injuries.number\_of\_motorist\_injured

    left join dates on collisions.crash\_date = dates.full\_date

    left join times on collisions.crash\_time = times.full\_time

)

select \*

from fact\_table

Graphical user interface, application, table

Description automatically generated

* **YML and MD files and their content**
  + sources.yml (destination and documentation for sources)

Text, timeline

Description automatically generated

* + core.yml (documentation for 311\_data, collisions\_data, and fact tables)

Timeline

Description automatically generated

* + 311.yml (documentation for 311 dimensions)

Timeline

Description automatically generated

* + 311.md (documentation for status column made using a table)

Graphical user interface, text, application, email

Description automatically generated

* + collisions.yml (documentation for collisions dimestions)

Timeline

Description automatically generated with low confidence

* + dbt\_project.yml

Text

Description automatically generated

* Directory overview:

A picture containing table

Description automatically generated

* DBT commands:

Graphical user interface, application

Description automatically generated

* Final DAG

Graphical user interface

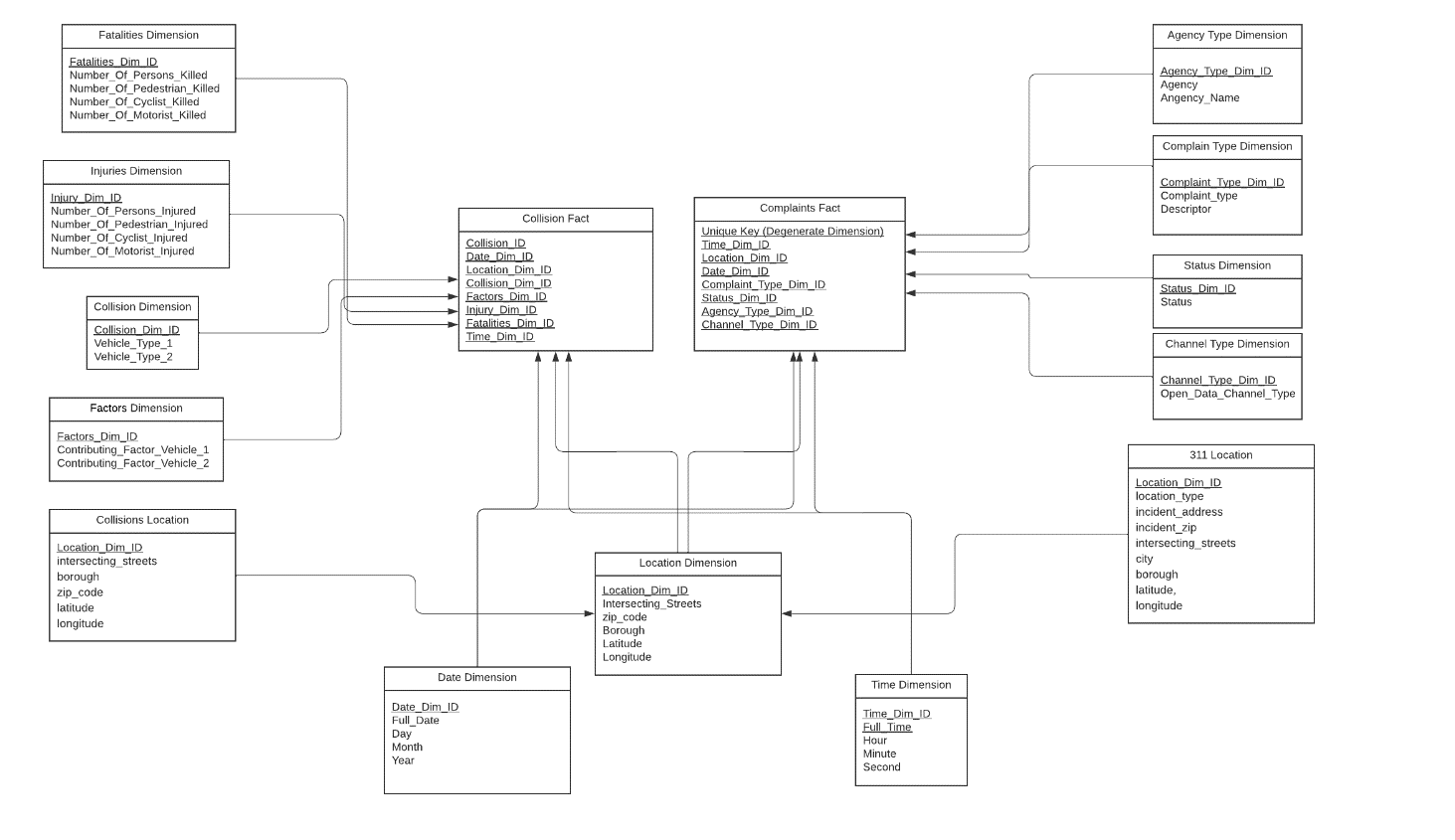
Description automatically generated

* BigQuery overview:

Table

Description automatically generated

* **Final view of the dimensional model:**

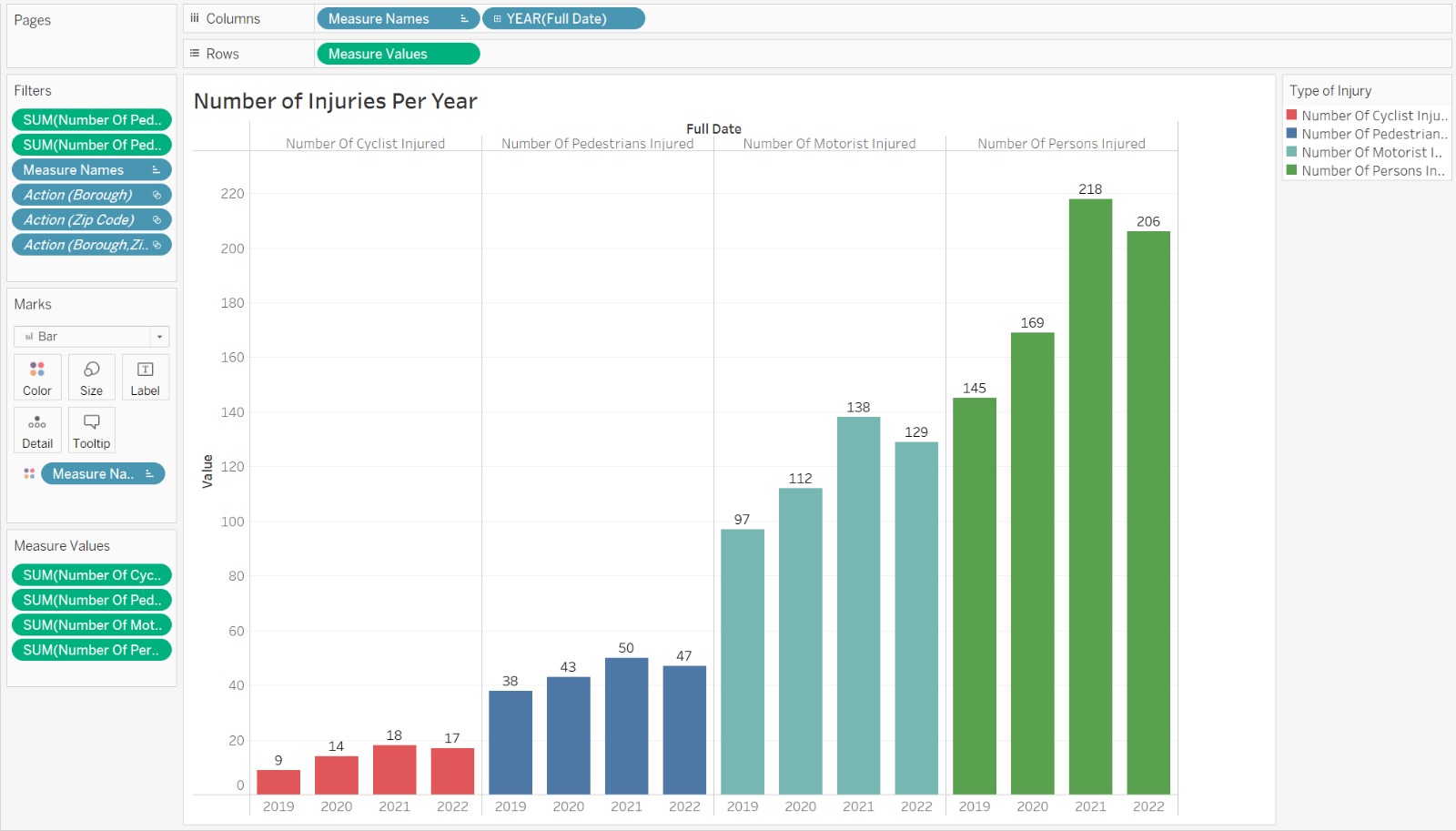


* + The 311 data had five staging models: status, complaint\_type, agency\_type, channel\_type, and the 311 location dimension.
  + The collisions data had five staging models: factors, collisions, fatalities, injuries, and the collisions’ location dimension.
  + The 311 location dimension and the collisions’ location dimension were used to create the location dimension for the fact tables.
  + The date and time dimensions were created generically using SQL functions and linked to the fact table. The format for time and data were also matched with the ones in 311 data and the collisions data.
* **Bi Application**

The BI and Data Visualization tool was used to create a dashboard application consisting of four sheets to help users navigate through the important aspects of the data easily and be informed on some of the important KPIs:

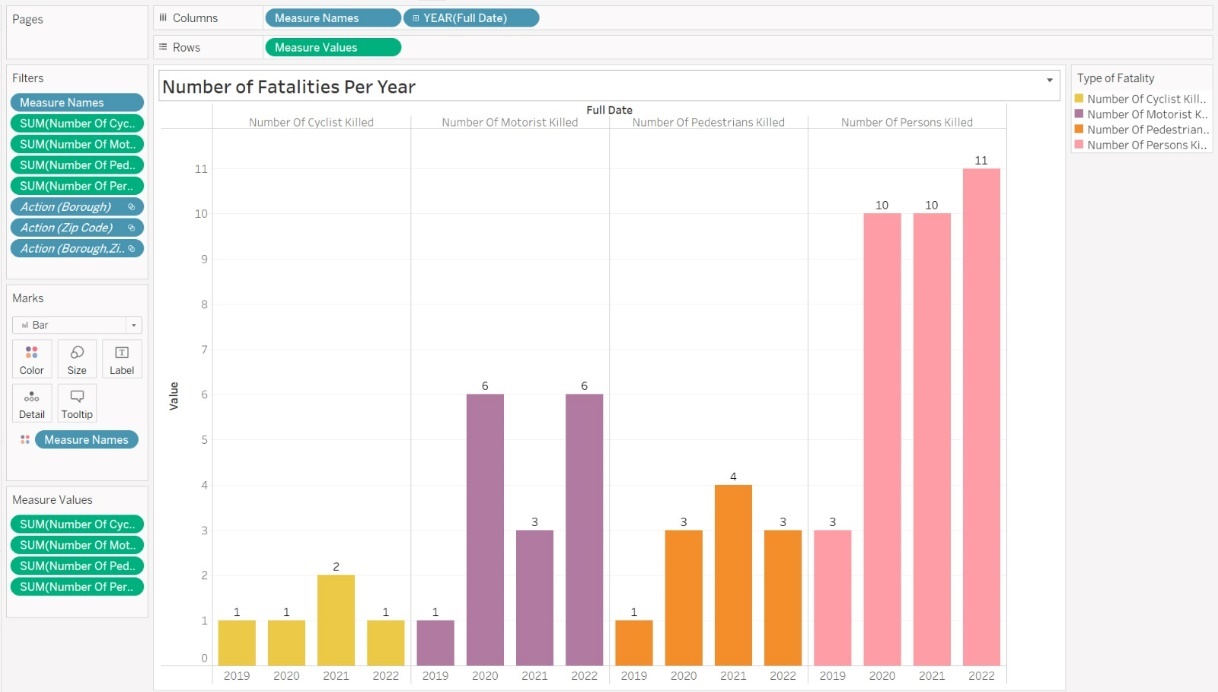
* + **Number of Injuries:**

This visualization was created using columns from the injuries dimension and the “full date” column in the date dimension. The type of injured personnel was used as a color filter.



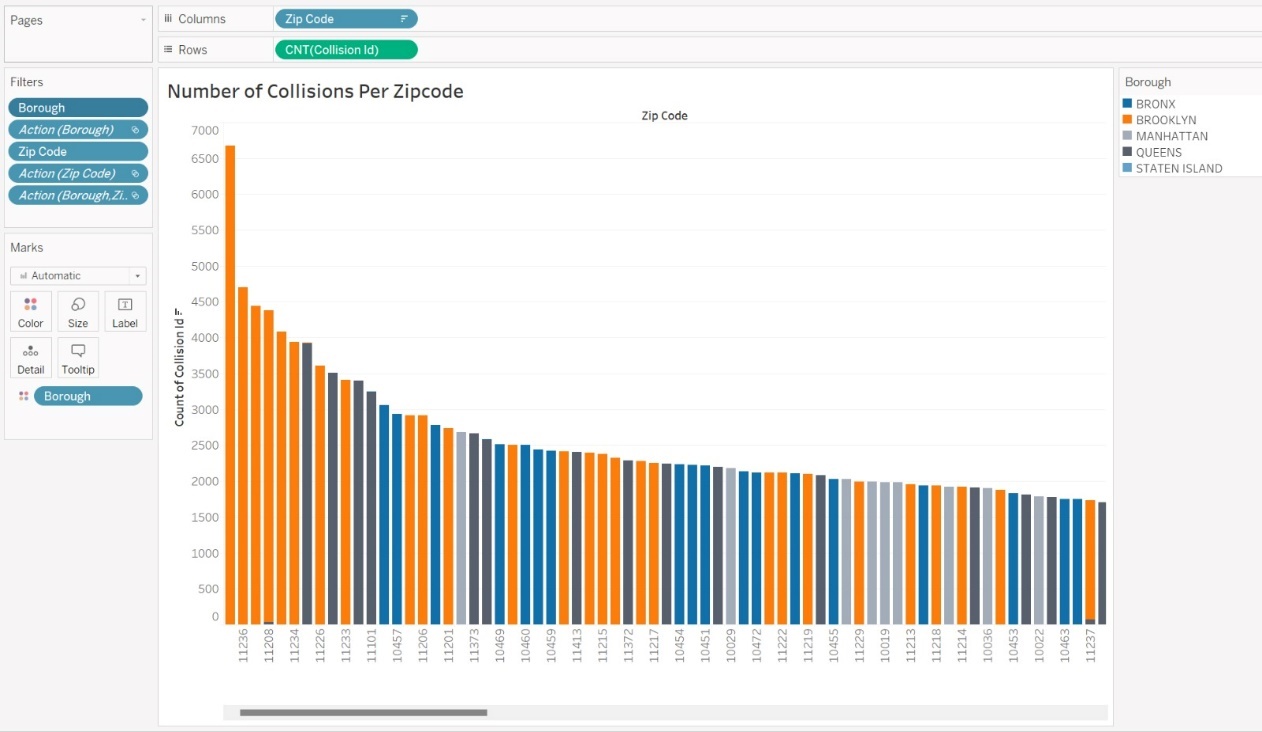
* + **Number of Fatalities**

This visualization was created using columns from the fatalities dimension and the “full date” column in the date dimension. The type of personnel involved was used as a color filter.



* + **Number of Collisions Per Zip code**

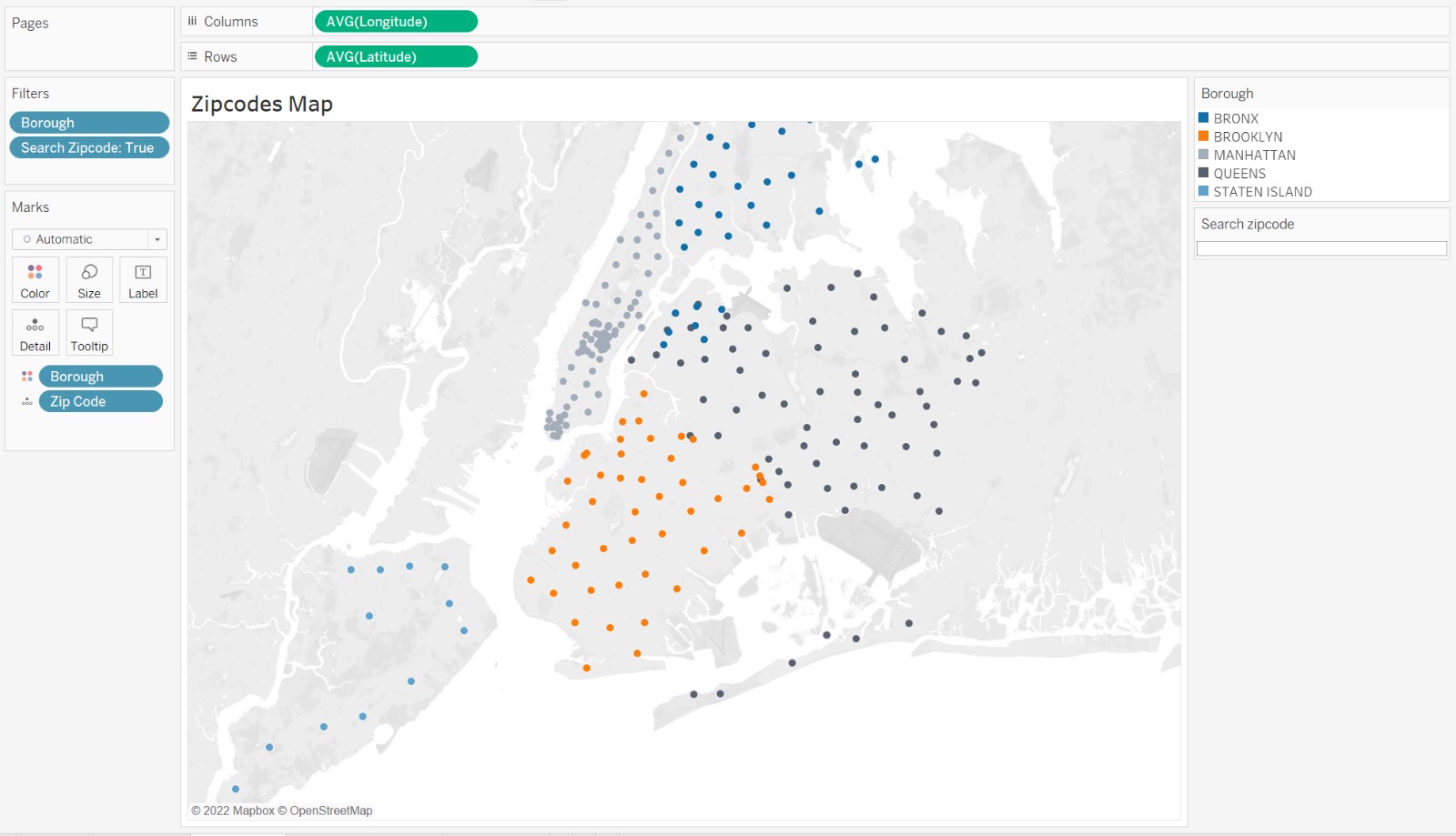
This was a direct visualization that we created using the zip code and the collision id column from the fact table. The collision Id had to be changed to a measure and used “count” the aggregate. Also, Borough was used as a color filter



* + **Zip Code Map:**

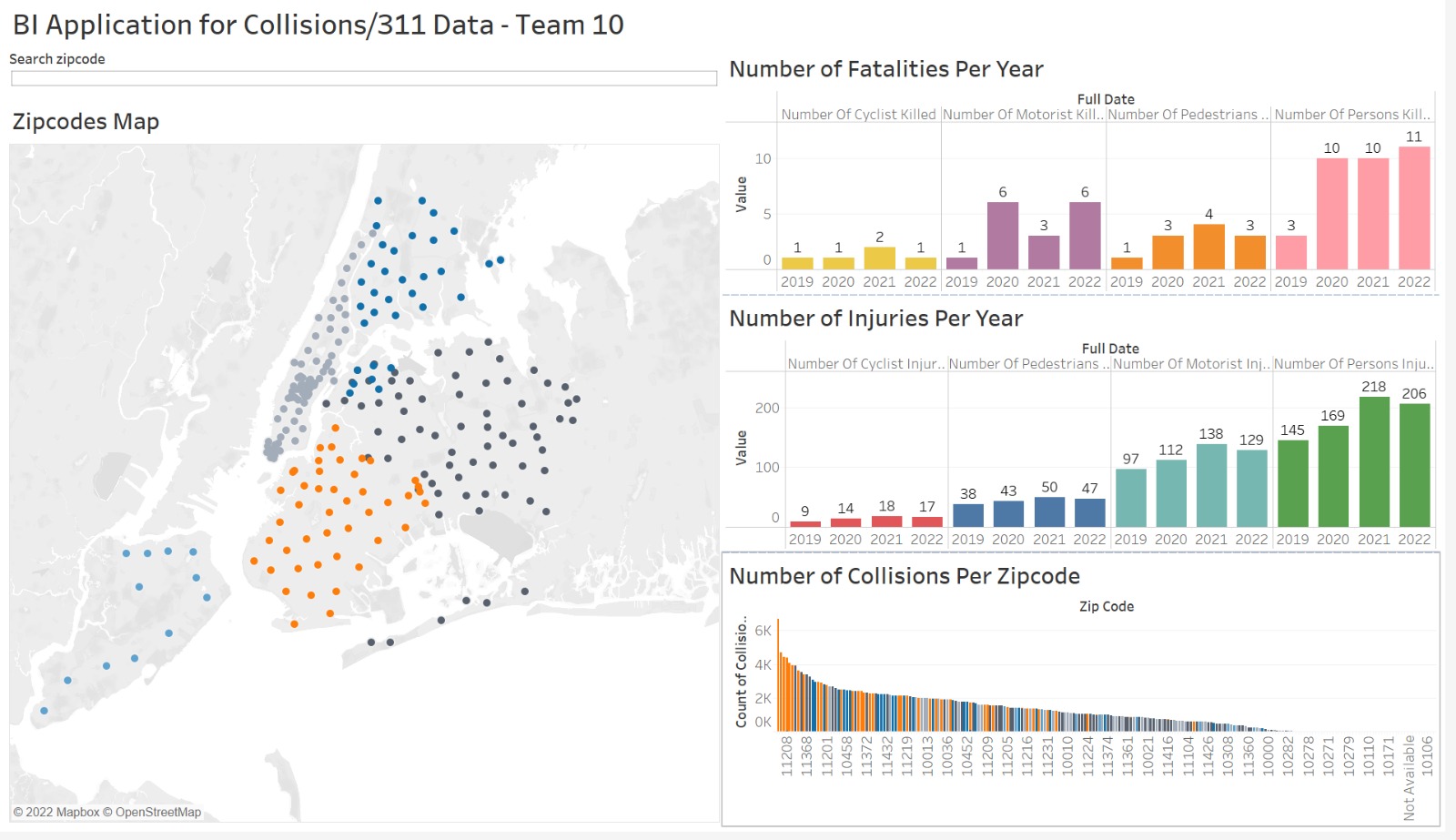
This visualization was created using the longitude and latitude as measures, zip code as the details, and borough as a color filter.

We also created a search parameter for this sheet that takes any zip code as a value, and the parameter was also added to the dashboard application.



* + **Dashboard**

The dashboard consists of the four sheets above and uses the “zip code map” sheet as a filter. The “search zip code” parameter was also added to the dashboard.



* + Link:

The dashboard was uploaded to Tableau Public and can be accessed and used by clicking this link:

[BI Application for Collisions/311 Data - Team 10](https://public.tableau.com/app/profile/muhammad.megahed/viz/BIApplicationforCollisions311Data-Team10/Dashboard1?publish=yes)

* **Tools and Applications Used**

To complete our project, we used several tools. Some of the tools we utilized are database tools, visualization tool, ETL (extract, transform, load) tools, and programming languages.

Database are used to store, organize, and manage data in a structured way. Therefore, we used Google BigQuery as our target DBMS (Database Management Systems) to find a correlation between the status of the city streets and vehicle collisions. To find the correlations, we utilized the data collected from the 311 and Motor Vehicle Collisions databases.

Visualizations are a graphical representation of data, such as charts, graphs, and maps. We utilized Lucid Chart to create our three dimensional model diagrams: 311 Model, Crash Model, and Combined Model. In addition, DBT generates a DAG (Directed Acyclic Graph) for our project. Tableau was used to create a BI application consisting of visualizations that help the user navigate through the data and visualize important KPIs.

ETL tools are used to extract data from multiple sources, transform the data into a standardized format, and load it into a destination, such as a database. The ETL tool we used for this project was DBT. DBT made it easier to work with large and complex datasets.

Programming languages are used to execute code that can solve a wide range of questions or problems. We utilized SQL and Python in our project. SQL was used in Google BigQuery to test queries and in DBT to create dimensions. Python was used for our data profiling on 311\_data and collisions\_data.

* **Conclusion**

For the group project, our team used a variety of software and database tools to successfully complete and coordinate our milestones. The mentioned tools are listed below:

* + WhatsApp - to plan out the project, store our notes/important information, and assign tasks to the whole team.
  + Zoom - to facilitate our meetings.
  + LucidChart - to create our data models like our three dimensional data models.
  + DBT - to handle communication with BigQuery so that we can process requests
  + BigQuery - to store our large dataset and run SQL commands on
  + Tableau - used to create visualizations for our data like the KPIs
  + Python & Jupyterlab - to run scripts to complete the database requirements for the project.

Overall, we had a very pleasant experience with the project. It was genuinely a great experience being able to work with people who were willing to put in the work needed to complete the project. WhatsApp and Zoom were our main methods of communication and coordination, and they were a great help in us working on and finishing the milestones before the deadlines.

The changes that we proposed for a future attempt at the project are at their core very simple foundational steps that would have made it easier for us to build the project on top of. In essence we basically needed better communication and preparation during, before and after meetings. Something of that nature is very feasible for a group to accomplish and would always guarantee a better experience/final product because of the higher quality of the input used to create the end result.

I think maybe for next year’s class on Data Warehousing there should be a bigger focus on NoSQL databases and unstructured data. While we spent a lot of time on SQL and SQL databases in class which led to it being what we used for our project I would have loved to approach this project with a NoSQL database like DynamoDB or FireStore, but unfortunately since we did not spend enough time on it in class the entire team did not feel comfortable with it, therefore, we decided to use SQL. It would be great for students next year to have more time being spent on NoSQL databases which are just as important as SQL databases.

* **Meeting Logs**

| **Date** | **Time Started - Time Ended** | **Members Present** | **What We Discussed** | **Platform** |
| --- | --- | --- | --- | --- |
| 9/21/2022 | 12:00PM - 1:30PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Accomplishing Milestone 1.  Trying to workout our individual schedules.  Finding data models to use.  Establishing our KPIs. | Zoom |
| 10/07/2022 | 8:30PM -10:30PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Accomplishing Milestone 2.  New KPIs based on feedback.  Dimensional Modeling. | Zoom |
| 10/20/2022 | 7:00PM-8:00PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Accomplishing Milestone 3  Finalizing Dimensional Model | Zoom |
| 11/13/2022 | 7:00PM-8:00PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Accomplishing Milestone 4. | Zoom |
| 11/28/2022 | 9:00PM-10:00PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Discussion regarding Milestone 5. | Zoom |
| 12/06/2022 | 7:00PM-8:00PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Follow-up on Milestone 5. | Zoom |
| 12/09/2022 | 9:00PM-10:00PM | Nazmus,  Steven,  Muhammad,  Christopher,  Joshua | Finalizing  Milestone 5. | Zoom |

* **Referneces:**
* <https://lucid.app/> - LucidChart for creating the Models
* <https://console.cloud.google.com/bigquery> – BigQuery from Google Cloud for the target DMBS.
* <https://cloud.google.com/bigquery/docs/reference/> - BigQuery Documentation for functions and syntax.
* <https://docs.getdbt.com/reference/dbt_project.yml> – DBT documentation for edits to default files and navigation through the Develop and Deploy hub.
* <https://courses.getdbt.com/courses/fundamentals> – DBT Fundementals course from homework 3 was a great reference in building the dimensions and creating documentation.
* <https://courses.getdbt.com/courses/jinja-macros-packages> – Jinja Course from homework 3 was a great reference to Jinja syntax.
* <https://zoom.us/> - Zoom for hosting meetings to finish milestones.
* <https://www.tableau.com/products/desktop/download> – For creating the BI application for the project.
* <https://stackoverflow.com/> - Answered a lot of questions that we encountered throughout the project.
* <https://www.w3schools.com/sql/> - A quick refresher to some of the needed SQL functions like coalesce( ).
* <https://docs.python.org/3/> - Used to check error messages and to review for the data profiling process.