Car Accidents & 311 Street Light Complaints  
short lineshort lineshort lineshort lineshort lineshort lineshort lineshort lineshort lineshort lineshort lineshort lineCIS 4400 LEC CTRA 17193

Professor Ramah Al Balawi

Group 2: Members

Sharia Hoque

Dorwin Liang

Joann Mei

Kelvin Nguyen

Gabriela Sanchez Mateo

# 

# 

# Table of Contents

[**Table of Contents 0**](#_yhitpvzbsy8s)

[**Milestone 1: Proposal 1**](#_xkl86wummqi1)

[Initial KPIs: 2](#_eizq1a6yz0t1)

[**Milestone 2: Final KPIs 3**](#_pvvjs9h32hda)

[Final KPIs: 3](#_s253rpcsdgnj)

[Data Sources: 4](#_up0qt9ahemke)

[**Milestone 3: Model Finalized 4**](#_irf56dkuqiw)

[311 Service Requests: 4](#_jkthzft4ke8n)

[Motor Vehicle Collisions - Crashes: 5](#_s39wc7dbp9mv)

[MERGED Models: 7](#_tptefgk8jk5f)

[**Milestone 4: ETL Tools and Target DBMS Selected 7**](#_ti6s1wtsx76y)

[**Milestone 5: ETL completed 8**](#_vbg5mqqrzjeq)

[Data Profiling: 8](#_d6jiro19mxj4)

[ETL completed 11](#_4n63hi9c01sc)

[311 complaint dimensions and Fact Table: 11](#_fkog1kx1aucn)

[Collisions dimensions and Fact Table: 12](#_ji4x219t2y10)

[Merged location: 14](#_6l3aditb5fmr)

[Merged Dates 14](#_2bte13b4vdln)

[**Milestone 6: BI Application Design and Development 15**](#_fa3lzkgibyk4)

[Top 10 Zip Codes With Highest Collision Counts: 15](#_vj17714st3cv)

[Traffic Complaints Number by Month by Year: 16](#_tkpvwglol7si)

[Collisions Heat Map: 16](#_hcniorbe2i4l)

[Count of People Injured by Collisions: 16](#_k1zh9nm66gtb)

[Collision Time 16](#_9m6k0of174ke)

[**Narrative: 16**](#_tm3fyxyeyib3)

[**Appendix: 17**](#_mlpt36rh94vp)

[Milestone 4: ETL Tools and Target DBMS Selected: Extract and Load step 17](#_c5jd2k7tm9sg)

[Milestone 5: ETL Completed Code 20](#_l6tz91pym0eo)

[ETL (EXTRACT) Completed Code for 311 dataset (updated) 20](#_l6tz91pym0eo)

[Data Profiling Updated 21](#_l6tz91pym0eo)

[ETL ( EXTRACT) Completed Code for Collisions Dataset (updated) 21](#_l6tz91pym0eo)

[Transform Step 311 complaint dataset 22](#_l6tz91pym0eo)

[Transform Step Collision dimensions and Facts 28](#_m5wrcd4ysvcv)

[Merged Location 32](#_1gpt2wx8awut)

[MERGED Date 34](#_g4qx3787xh0x)

[**KPI Visualizations 35**](#_qlaikqdkcke)

[**Reference List 36**](#_h0di2gkb30r5)

[**Attendance: 37**](#_3w4148ccgqd)

# 

# Milestone 1: Proposal

We are addressing the concerns around traffic light complaints in search of a correlation with vehicle accidents around New York City for **2018-2023.** From the 311 complaints, we are using the complaint type: ***Traffic Signal Condition****,* alongside with dataset, ***Motor Vehicle Collisions*** in New York City. This project will assess if traffic light complaints correlate to the number of motor accidents and if more serious actions must be taken to address these complaints.

Within the 311 complaints dataset, each record includes information about the locations (coordinates, zip code), description of the complaint, complaints’ status, dates, and others. In the 311 complaint dataset, for the complaint type: Traffic Signal Condition, there are 572593 records for 2010 to 2023 and for the last *five years,* ***2018-2023, there are 186343*** *records* as of Oct 10, 2023.

The dataset Motor Vehicle Collisions- Crashes contains details on the crash event with different fields of the number of people injured or killed; such as; the number of persons injured and killed, *coordinates, zip codes, dates, and others* for each record. For this dataset, there are **665359 records from 2018 - 2023** as of Oct 10, 2023.

Both 311 and Collisions datasets have the same location fields such as zip code, borough, longitude, and latitude, and the same date range of 2018-2023. By integrating those datasets through the records’ longitude and latitude, and same time period using crash\_date (Collision dataset) and complaint\_created\_date (311 Traffic Signal Condition dataset), we can determine an accurate location or proximity of where and when the complaints were made in order to observe a correlation. We will create two data marts for the two datasets. Frequently when complaints are filed regarding traffic light conditions, a specific location must be provided. Thus, it is within reason that grouping these two within a specific coordinate would be effective.

## 

## Initial KPIs:

For the key performance indicators, we must assess the following:

* How many accidents have a correlating traffic light report?
* How many accidents do not have a correlating traffic light report?
* Identify the area with the most abundant traffic light reports from the Traffic Light Complaints table.
* Identify the area with the most accidents from the Vehicle Collisions table.
* Clustering traffic light complaints with vehicle collisions.
* The average distance of an accident from the nearest cluster of complaints.

Those KPIs will allow us to visualize whether there are clustered groups of accidents and the percentage of recorded accidents that can be attributed to traffic light complaints. Naturally, there are assumptions to be made. We must introduce a bias, assuming that if a vehicle accident occurred near or next to a traffic light complaint, it was due to the traffic light. In addition, we must establish the required parameters to determine how far the boundaries of an accident were influenced by the proximity of a traffic light. For instance, two blocks or a certain amount of meters before it is out of bounds and the traffic light complaint is considered unrelated.

# 

# Milestone 2: Final KPIs

## Final KPIs:

1. Accidents per Zip Code Heatmap *[COLLISION]*
2. Number of pedestrians injured per Zip Code per Date
3. Number of pedestrians killed per Zip Code per Date
4. Number of cyclists injured per Zip Code per Date
5. Number of cyclists killed per Zip Code per Date
6. Number of accidents per Zip Code *[Collision Table]*
7. Number of accidents per Month
8. Number of accidents per Week
9. Number of complaints per Month *[from 311 table]*
10. Average Time Passed Between CREATED DATE and CLOSED DATE
11. Average Distance Between Location of Complaint and Place of Collision
12. Complaints per Zip Code per Date
13. Complaints per Zip Code
14. Complaints per Month
15. Complaints per Week
16. Injuries per Zip Code
17. Deaths per Zip Code
18. Injuries per Month
19. Injuries per Week
20. Injuries per Day

## Data Sources:

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

* Year Range: 2010-2023
* Records for Complaint Type: **Traffic Signal Condition**;
* Records as of Oct 10, 2023: 572,593; Attributes: 41

**Complaint Type: Traffic Signal Condition:**

* **Motor Vehicle Collisions - Crashes** - [Motor Vehicle Collisions - Crashes | NYC Open Data](https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95) [Link]
  + **Citation**: New York City. (2023). Motor Vehicle Collisions (Crashes) [Data set]. City of New York Open Data. Retrieved from<https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95>
    - Year Range: 2012-2023
    - Records: 2,032413; Attributes: 29
    - (Note: contains details on the crash event with the number of people injured or killed)

# 

# Milestone 3: Model Finalized

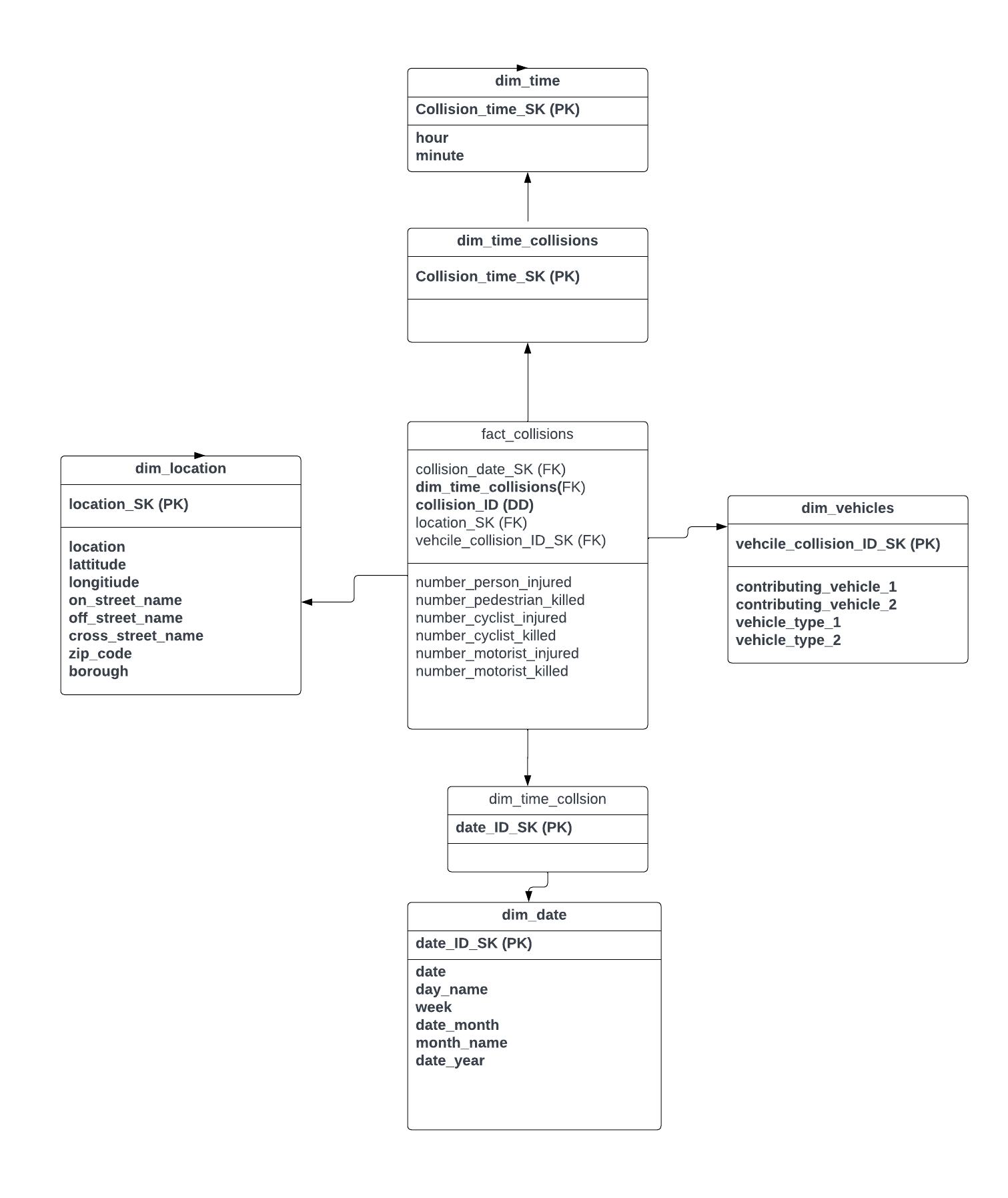
## 311 Service Requests:

* Model of 311 services transactional grain:

****

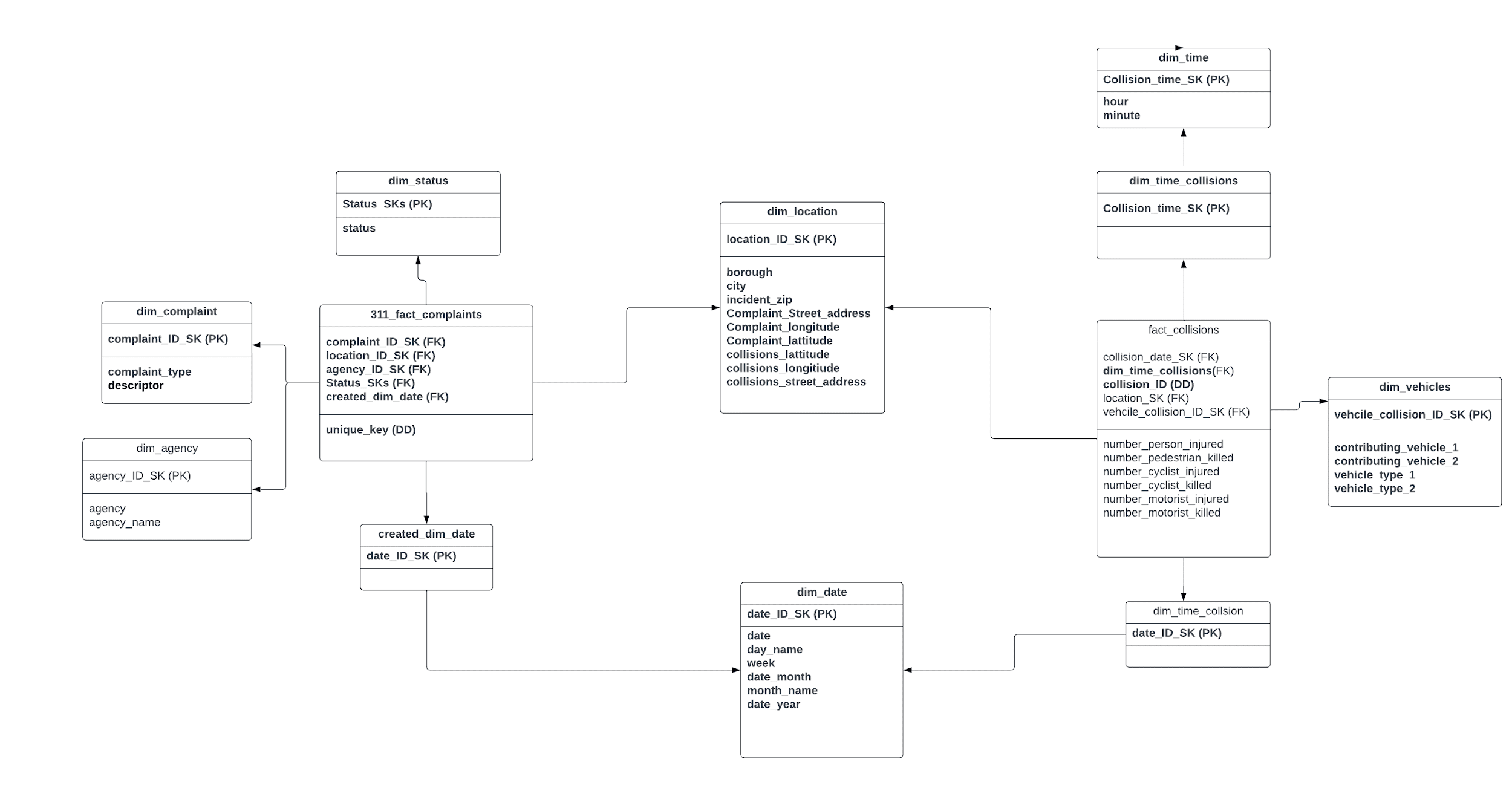
## Motor Vehicle Collisions - Crashes:

* Model of Motor Vehicle Collisions transaction grain:

****

# 

## MERGED Models:



# Milestone 4: ETL Tools and Target DBMS Selected

We will build the ETL process using **dbt Cloud** and host the Data Warehouse in **Google BigQuery**.

**Extract**: In this phase, we used API, to retrieve from the datasets 311 Service Requests and Motor Vehicle Collisions - Crashes from NYC Open data. We used Python to store the data in Google BigQuery. The code for this step is in this [appendix](#_mlpt36rh94vp) section.

**Transform**: We are using the ETL tool DBT for the transformation step. In this step, we are creating dimensions (for dates, time, locations, status, compliant\_types, number of collisions, and others) by handling duplicates, missing, inconsistencies, and outlier values and correcting each field's data types. In this step, we will also integrate the datasets in this phase.

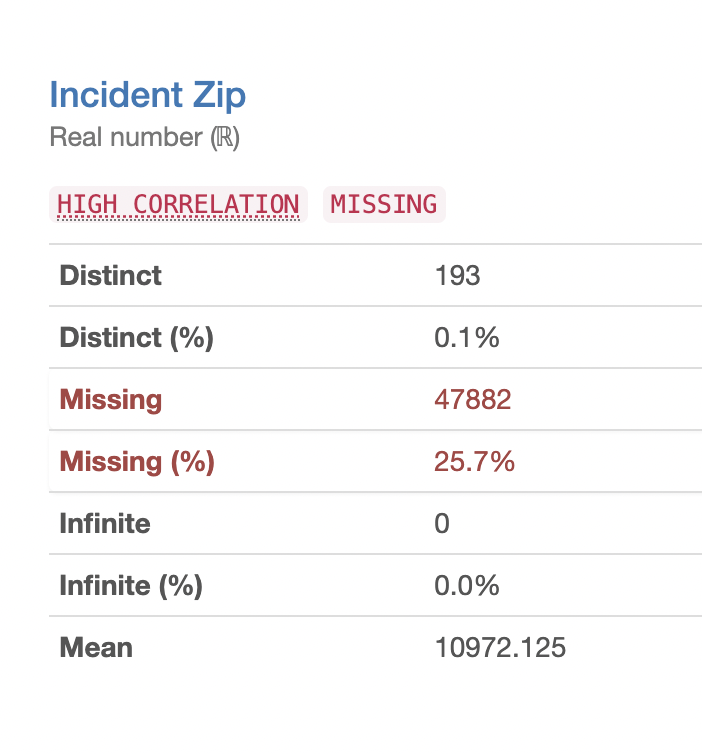
**Load**: In the loading phase, we will take the transformed and cleaned data, or dimensions, and load them into fact tables in Google BigQuery using dbt.

# 

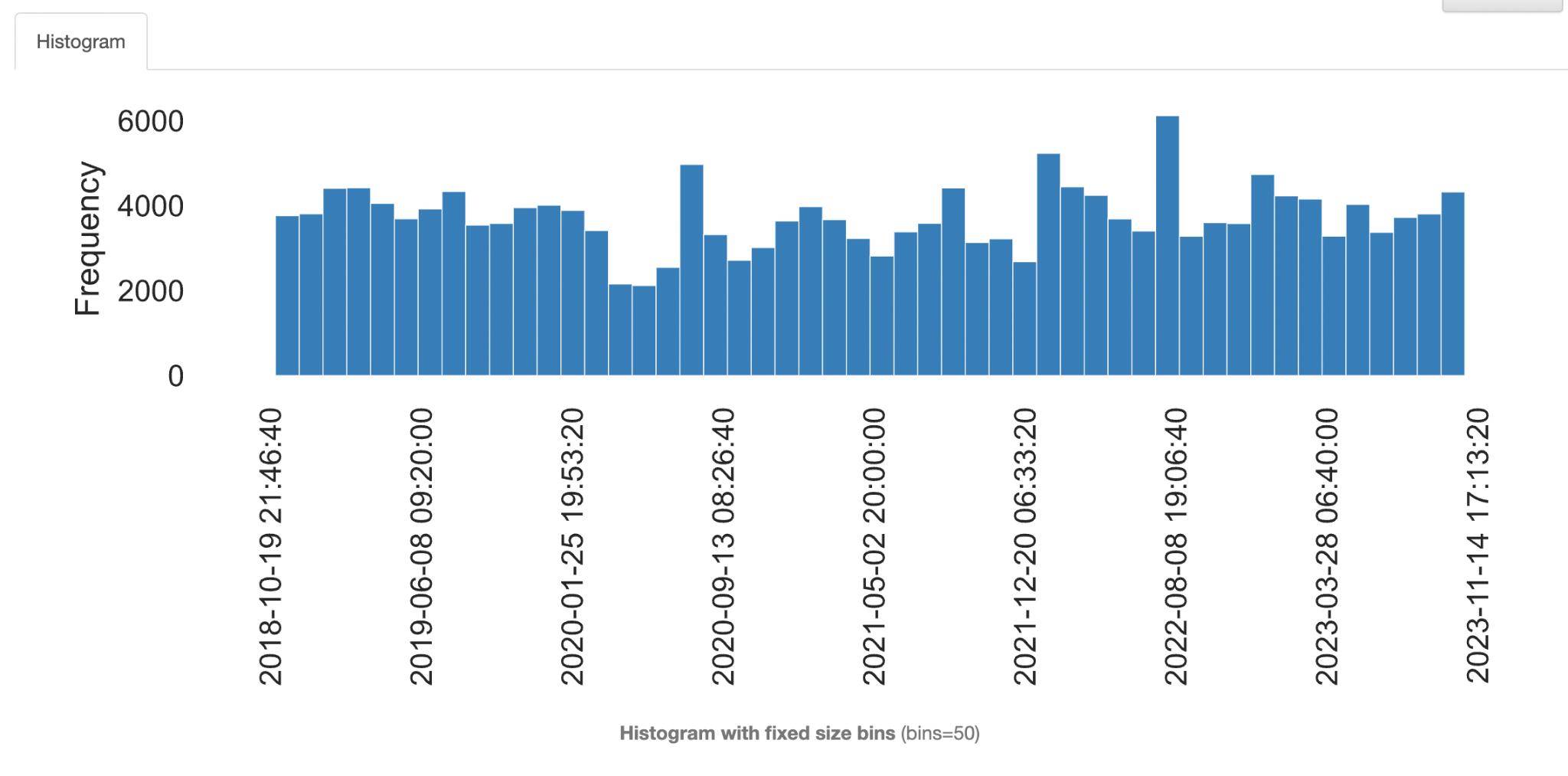
# Milestone 5: ETL completed

## Data Profiling:

**311 data:**

****

**25.7% zip code missing** in 311 from 2018-2023 for 311 Traffic Signal Condition.



[Created date frequency chart for 311 data when the complaint type is Traffic Signal Condition]



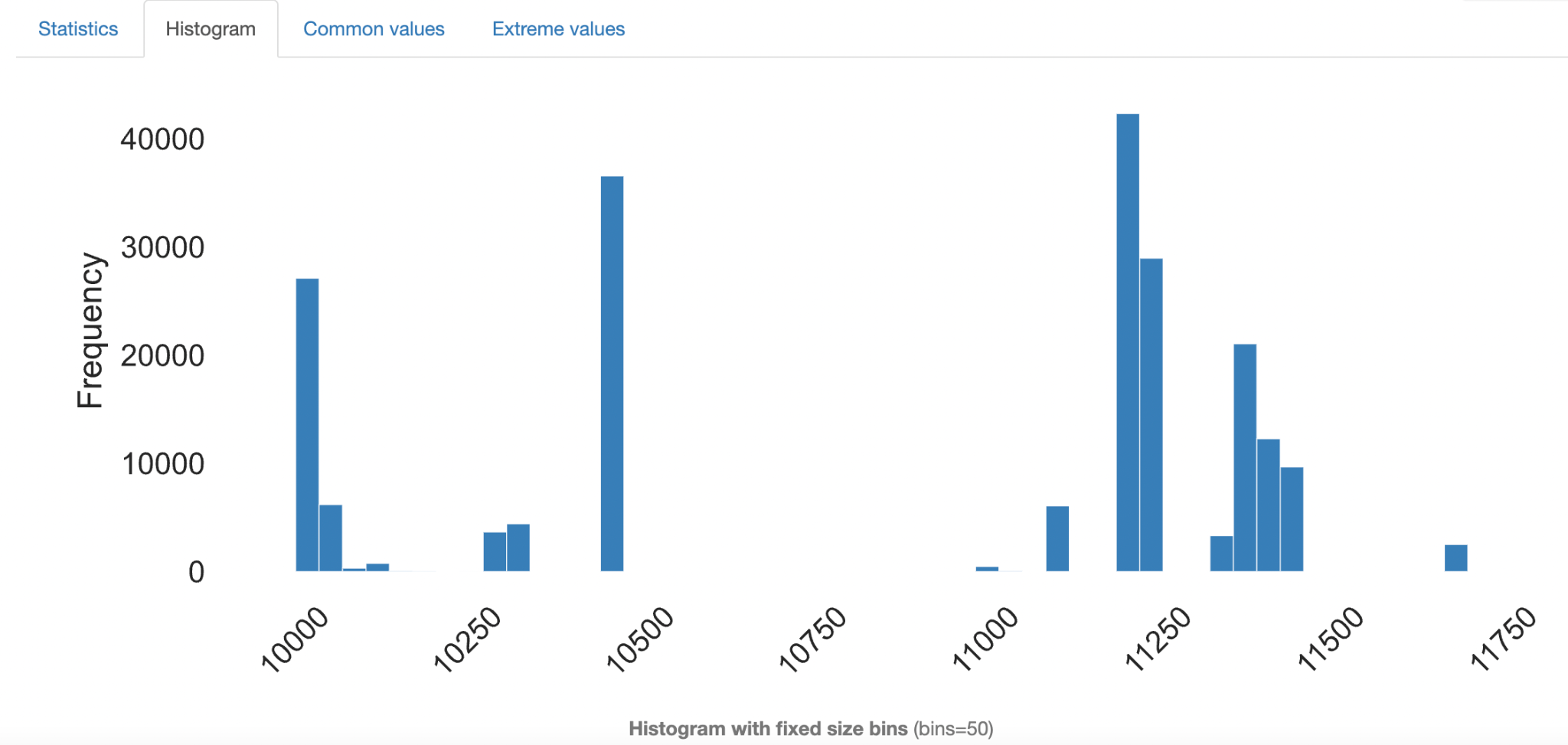
The top 10 zip code for Traffic Signal Condition

**Collision Data:**

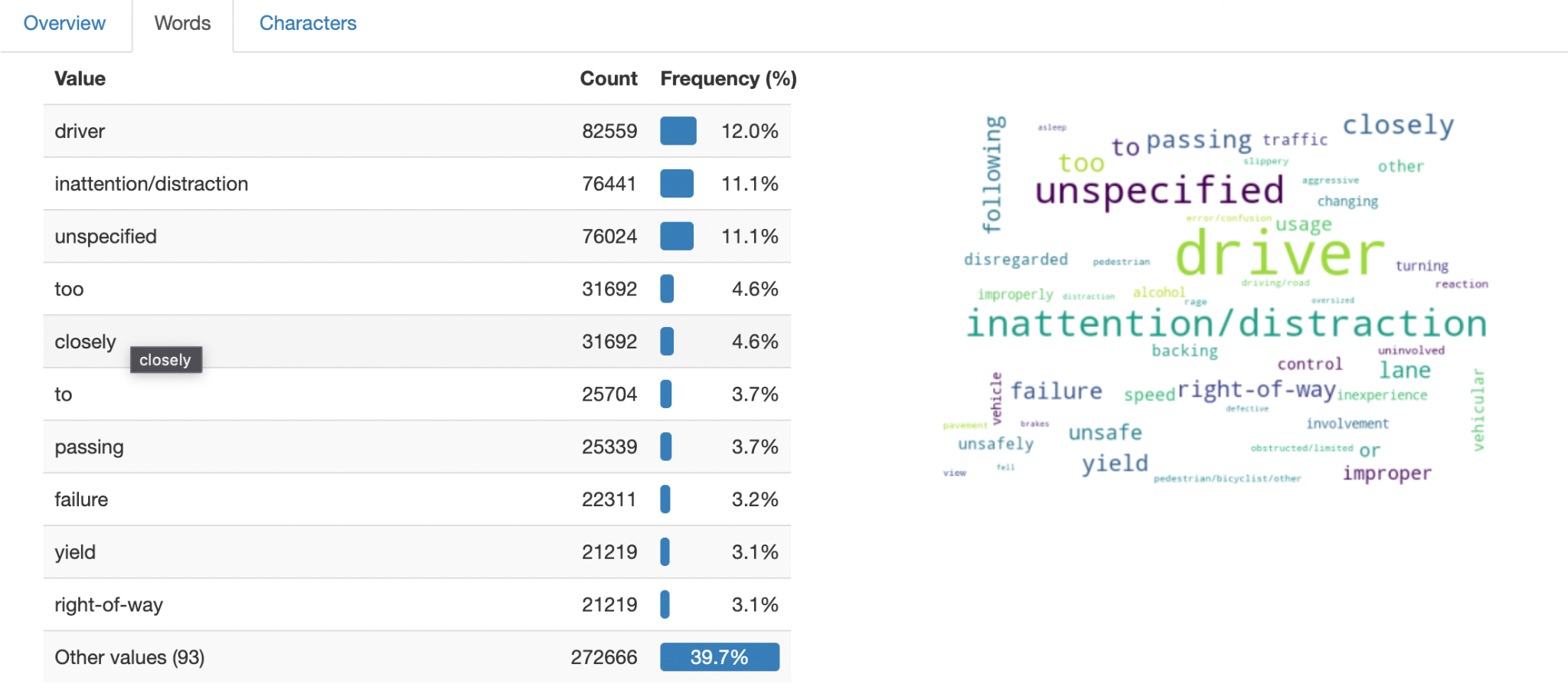
****

**[Missing values]**

When the fields are Contributing Factore Vehicle 3 or larger and the Vehicle type Code is 3 and larger, more than 90% of data are missing for those values. This is why we decided to use Only Contributing Factore Vehicles 1 &2 and the Vehicle type Code is 1 & 2 instead of using all fields.

****

**[Frequency of ZIp codes for collision data]**



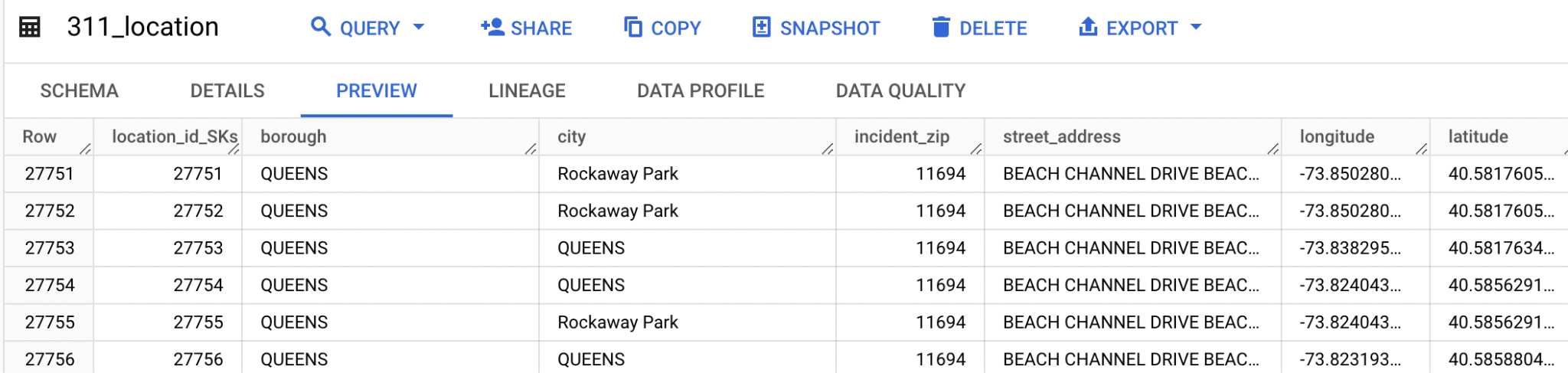
The most common words for reasons of collisions are unspecific, distractions, and inattention.

## ETL completed

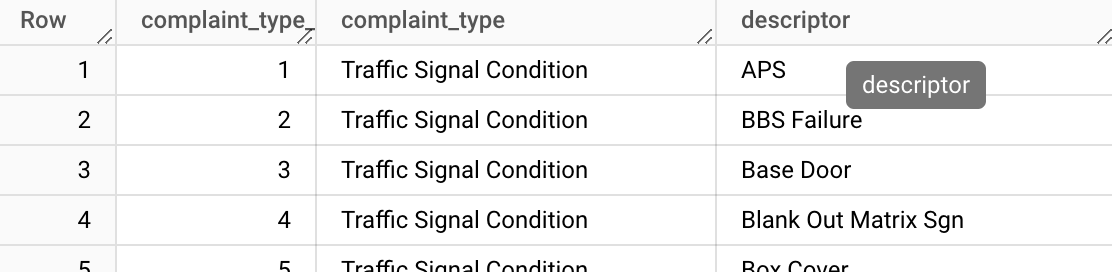
**Summary:** In this phase, we extracted and loaded the data using Python. We added the API in the Python code shown below ([311](#_l6tz91pym0eo), [Collisions](#_l6tz91pym0eo)) in which that data was extracted into a CSV file and uploaded into GCS. From there the data is imported into a big query where we could move on towards the Transform process of this milestone. Additionally, the code for transformation was used through DBT to create dimensions for both 311 and collisions. which is also attached below in the appendix section for [311](#_l6tz91pym0eo) and [Collisions](#_l6tz91pym0eo).

## [311 complaint dimensions and Fact Table:](#_l6tz91pym0eo)

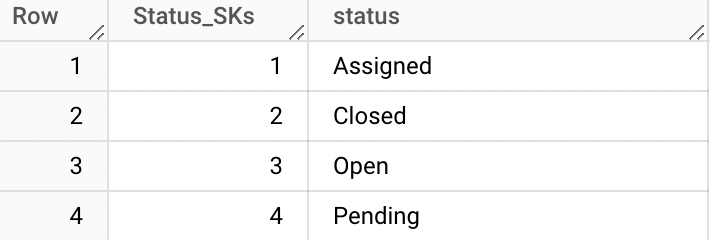
Locations dimensions:



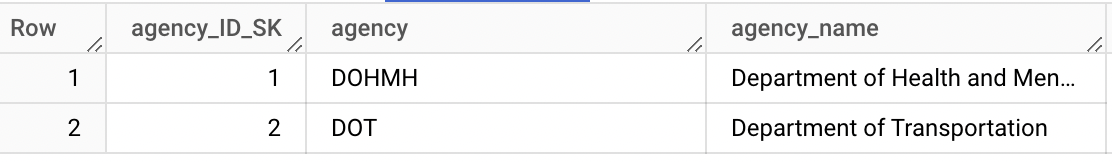
Complaint type dimensions:



Status dimensions:



Agency dimensions:

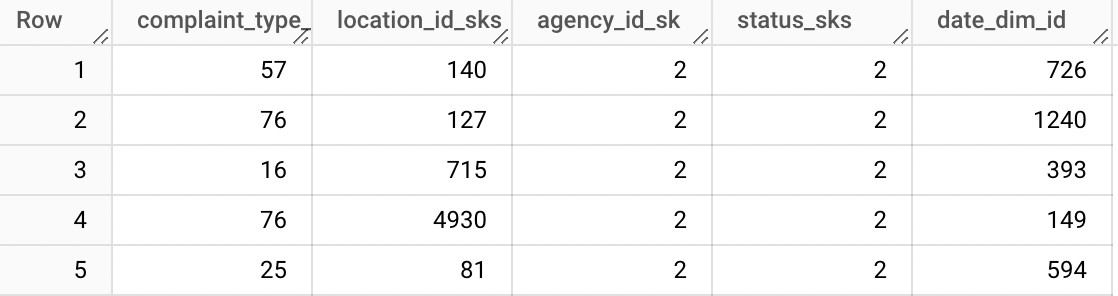


Complaint Date dimensions:

*Note: This dimension is created from Date view*

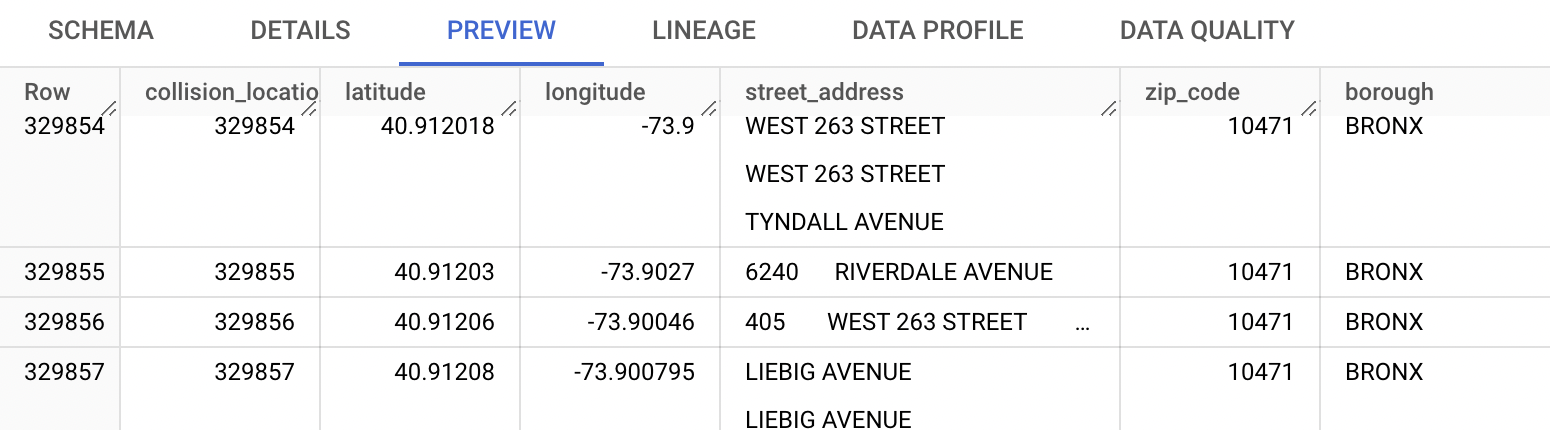
**

311 complaint FACT table:



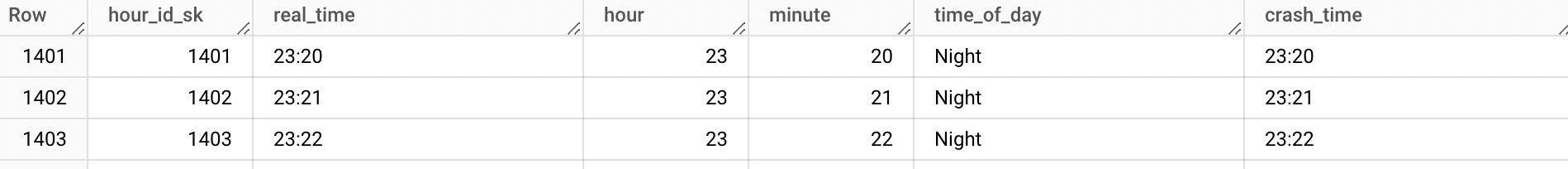
## [Collisions dimensions and Fact Table:](#_m5wrcd4ysvcv)

Collisions Location dimensions:



Collisions time dimensions:

*Note: this dimension is created from time view*

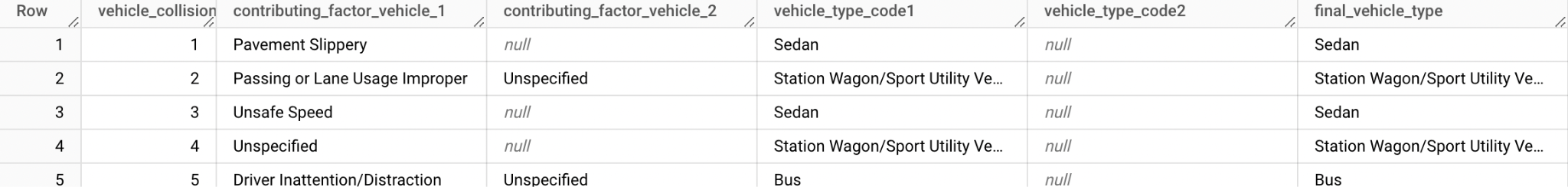


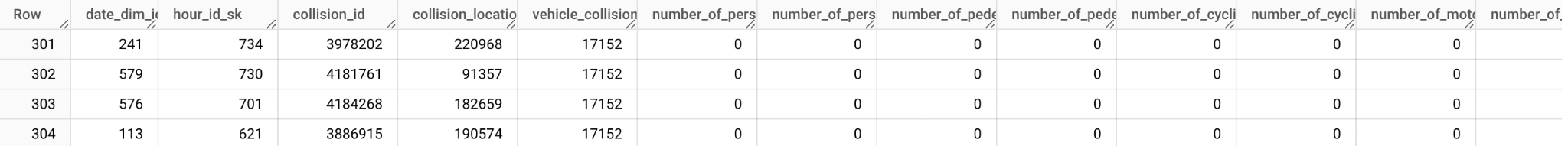
Collisions Date dimensions:

*Note: This dimension is created from Date view*

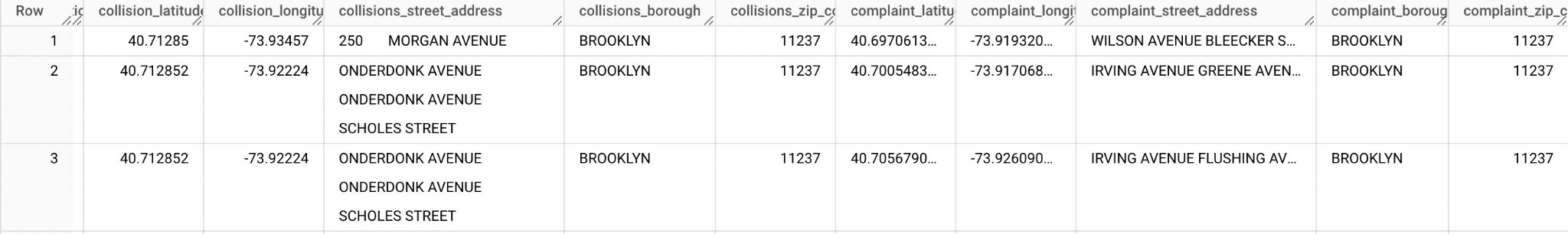


Vehicles dimensions:



Collision Fact table: 

## Merged location:

****

**Additional Notes:** For both datasets, there are no common latitude and longitude, in this case, we joined both datasets with zip codes

## Merged Dates



*Notes: Both datasets are merged by the full date*

# Milestone 6: BI Application Design and Development

## 

## Top 10 Zip Codes With Highest Collision Counts:

The top 10 displayed shows that 11385 is the Zip Code with the highest collision accidents which hold true to the heatmap as well and 11385 is notorious for this problem. This finding is consistent with our other visualizations.

## Traffic Complaints Number by Month by Year:

This chart depicts a time series graph of complaints made per month by year. This is consistent with COVID-19, where no accidents could occur due to the pandemic. Interestingly, there have been fewer complaints after most COVID-19 restrictions were lifted.

## Collisions Heat Map:

The heat map shows a temperature model of collision incidents, where light green is the lowest occurrence and red is the highest.

## Count of People Injured by Collisions:

The following line shows that in the middle of each year, the line tends to peak which means people are affected the most when it is around the middle of the year.

The graph also shows the forecast of the total injured for 2024-25 which will increase than previous years.

## Collision Time

This line shows that when it is **16:00 or 4 p.m.** the collision happens the most (based on 2018-2023).

# Narrative:

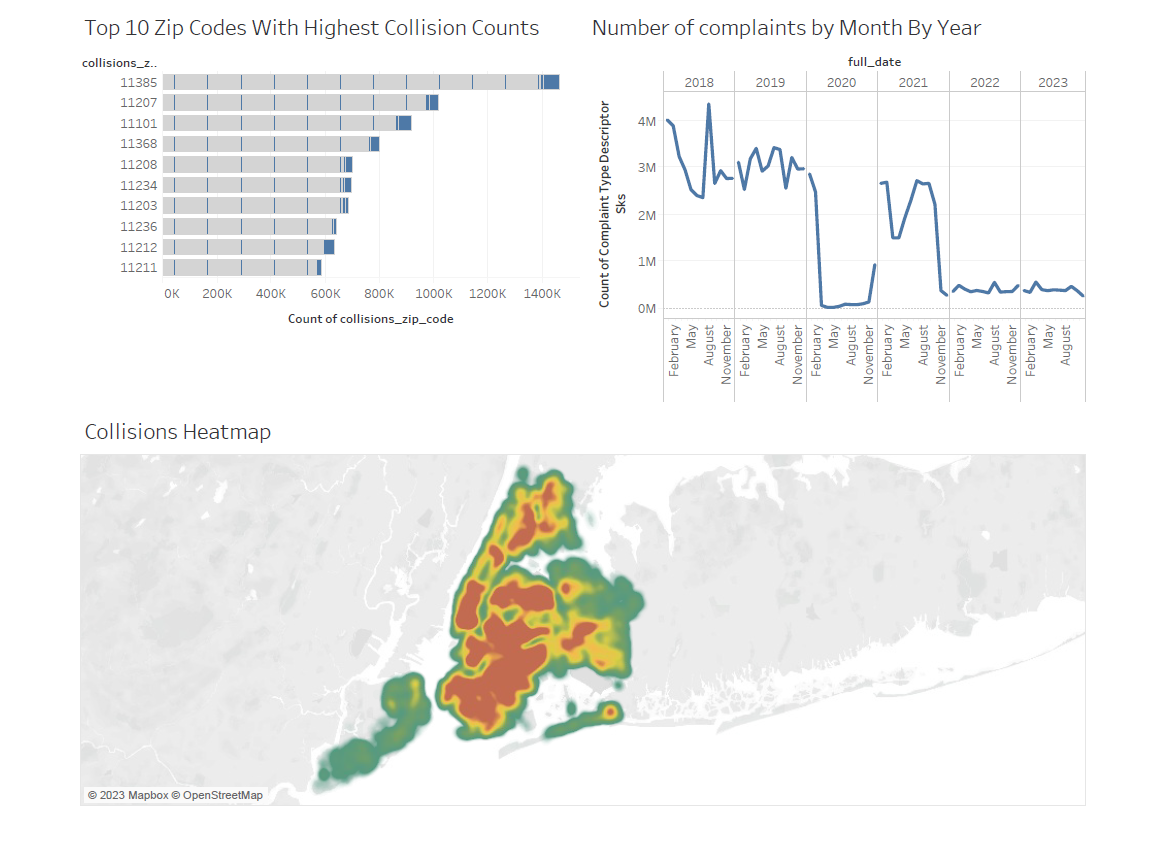
1. Software Used:
   1. **DBT**: Transformation portion of ETL
   2. **Google BigQuery**: Database storage
   3. **Tableau**: Dashboard and Business Intelligence
   4. **Google Collab**: Extraction and Loading
   5. **Google Cloud Storage**: Storing data loaded by google collab to load into BQ
   6. **WhatsApp**: fast communication between team members
   7. **Discord** : Real-time Live screen sharing for productivity and collaboration.
2. the group’s experience with the project (which steps were the most difficult? Which were the easiest? What did you learn that you did not imagine you would have? If you had to do it all over again, what would you have done differently?)
   1. The most difficult step was making Google BigQuery accessible to all team members to collaborate because we did not have previous experience with this and it was a challenge to learn and figure out and use DBT to connect it to the Google BigQuery.
   2. The easiest step was creating the Tableau data visualizations and the dashboard because we had previous experience working with Tableau.
   3. One thing that we learned that we did not imagine we would have learned is the panda's data profiling tool. We did not know this tool existed until now, and it is very useful.
   4. If we had to do this all over again, something we would have done differently is take time to slowly develop our database schema so that we wouldn’t run into any problems later on when we try to transform the data into tables and join them together.
   5. The new proposed benefits can be realized by the new system by going back and readjusting the DBT according to the new model. For instance, we had to convert latitude and longitude for collisions and complaints into dimensions to use in Tableau, however, this process was easy as Tableau allowed that change easily.
   6. Overall, this project greatly expanded our knowledge of databases, data warehousing, data engineering, and business intelligence tools like Tableau. From creating a dimensional model diagram, creating and executing an ETL process, and connecting the Google BigQuery database to Tableau for data visualization, we were able to understand the process of developing a data warehouse from start to finish.

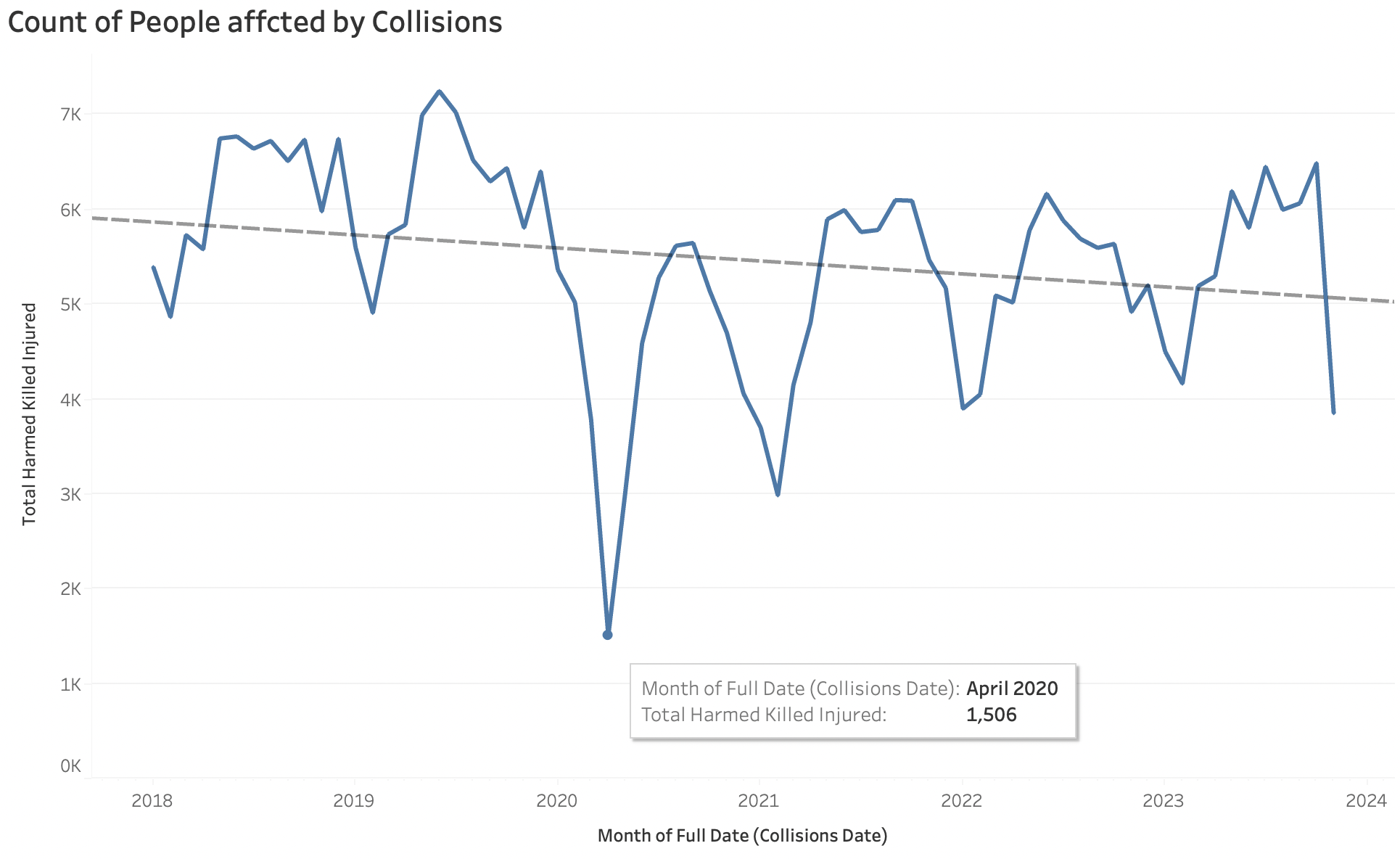
# Appendix:

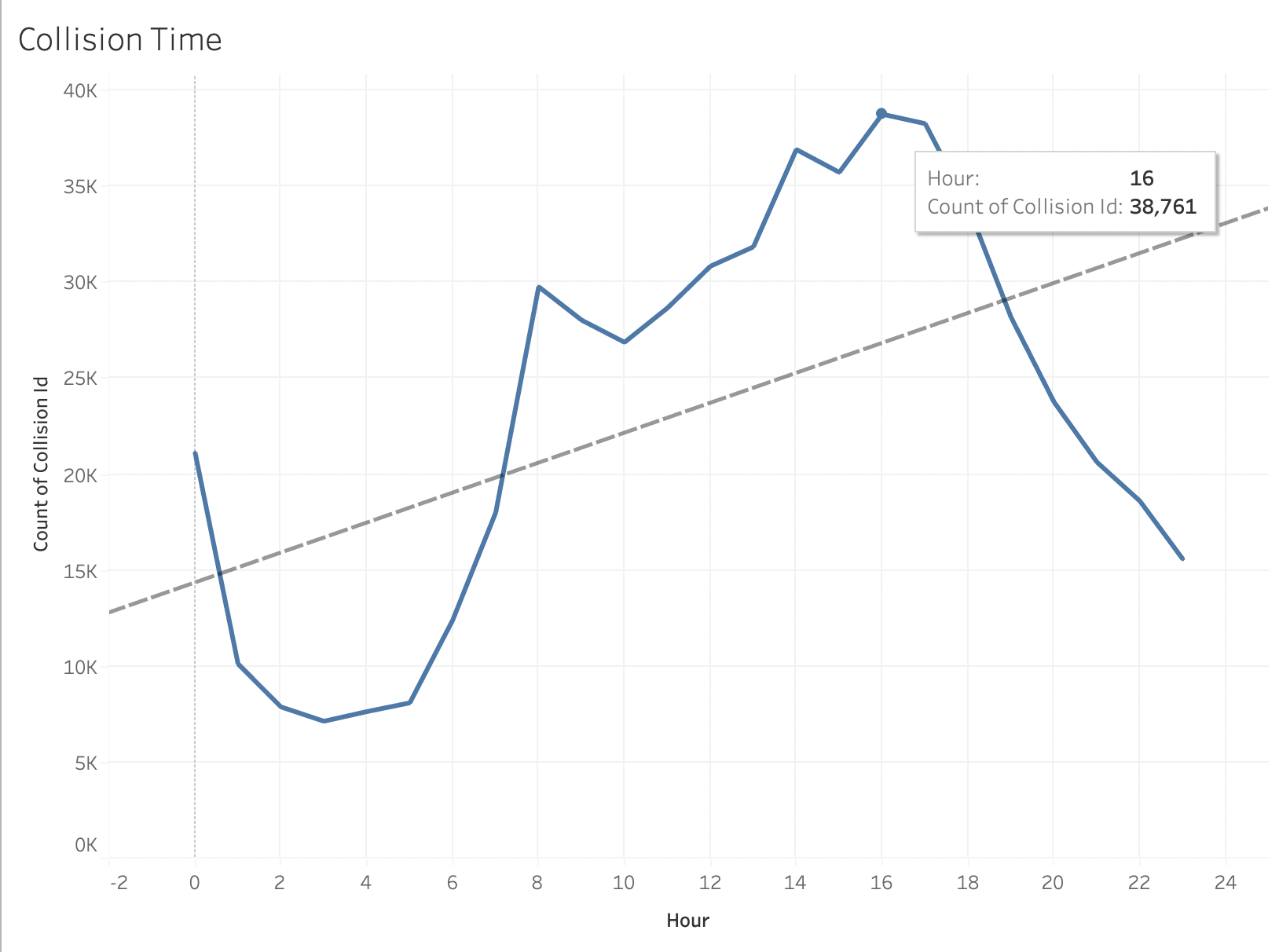
|  |
| --- |
| Milestone 4: ETL Tools and Target DBMS Selected: Extract and Load step |
| **Direction:**  Reference: <https://holowczak.com/getting-started-with-nyc-opendata-and-the-socrata-api/2/>   1. pip install packages (in the local machine):    * pip install sodapy,    * pip install pandas,    * pip install os,    * pip install --upgrade google-auth google-auth-oauthlib google-auth-httplib2 2. Create API and App token in the developer account NYC open data 3. Create a service account in the google cloud (IAM & Admin) with key in JSON.    * it will download a json file 4. Import the packages 5. Follow the codes below (directions are there) |
| import os  import pandas as pd  from google.auth import exceptions  from sodapy import Socrata  from google.oauth2 import service\_account |
| 311 Service Requests |
| data\_url = 'data.cityofnewyork.us'  data\_set = 'erm2-nwe9'  app\_token = '51aSrlHeY2Ww9GYGwDAWItAUx'  # Authenticate with your Google Cloud credentials  credentials = service\_account.Credentials.from\_service\_account\_file(  '/Users/shariahoque/Desktop/Fall\_2023/CIS 4400/Project /json\_from\_Big\_Query.json', # Replace with the path to your JSON key file  scopes=["https://www.googleapis.com/auth/cloud-platform"],  )  # Create the client to point to the API endpoint  client = Socrata(data\_url, app\_token)  # Set the timeout to 60 seconds  client.timeout = 60  #Several clauses  where\_clause= "Complaint\_Type = 'Traffic Signal Condition' AND date\_extract\_y(created\_date) between 2018 AND 2023"  select\_clause = "Unique\_Key, Created\_Date, Closed\_Date, Complaint\_Type, Descriptor,Agency,Agency\_Name,Status,City,Borough,Incident\_Zip,Street\_Name,Cross\_Street\_1,Cross\_Street\_2,Intersection\_Street\_1,Intersection\_Street\_2,Latitude,Longitude"  orderby\_clause = "date\_extract\_y(Created\_Date) DESC"  # Retrieve the data as a Pandas DataFrame & change the limit  results = client.get(data\_set, where=where\_clause, select=select\_clause,order = orderby\_clause, limit =3000000)  df = pd.DataFrame.from\_records(results)  # Define the project\_id and destination BigQuery table  project\_id = 'cis4400project-403800'  destination\_table = 'cis4400project-403800.projectDatasets.311\_traffic\_lights'    # # Upload the Pandas DataFrame to BigQuery  # df.to\_gbq(destination\_table, project\_id=project\_id, if\_exists='replace')  df.to\_gbq(destination\_table, project\_id=project\_id, if\_exists='replace') |
| Motor Vehicle Collisions - Crashes |
| data\_url = 'data.cityofnewyork.us'  data\_set = 'h9gi-nx95'  app\_token = '51aSrlHeY2Ww9GYGwDAWItAUx'  # Authenticate with your Google Cloud credentials  credentials = service\_account.Credentials.from\_service\_account\_file(  '/Users/shariahoque/Desktop/Fall\_2023/CIS 4400/Project /json\_from\_Big\_Query.json', # Replace with the path to your JSON key file  scopes=["https://www.googleapis.com/auth/cloud-platform"],  )  # Create the client to point to the API endpoint  client = Socrata(data\_url, app\_token)  # Set the timeout to 60 seconds  client.timeout = 60  # #Several clauses  where\_clause = "CRASH\_DATE BETWEEN '2018-01-01' AND '2023-12-31'"  select\_clause = "CRASH\_DATE,CRASH\_TIME,BOROUGH,ZIP\_CODE,LATITUDE,LONGITUDE,ON\_STREET\_NAME,CROSS\_STREET\_NAME,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\_CODE1,VEHICLE\_TYPE\_CODE2,VEHICLE\_TYPE\_CODE\_3,VEHICLE\_TYPE\_CODE\_4,VEHICLE\_TYPE\_CODE\_5"  orderby\_clause = "CRASH\_DATE DESC"  # Retrieve the data as a Pandas DataFrame & change the limit  results = client.get(data\_set,where = where\_clause, select = select\_clause, order = orderby\_clause, limit =1000000)  df = pd.DataFrame.from\_records(results)  project\_id = 'cis4400project-403800'  destination\_table = 'cis4400project-403800.projectDatasets.Motor\_Vehicle \_Collisions'  df.to\_gbq(destination\_table, project\_id=project\_id, if\_exists='replace') |

|  |
| --- |
| Milestone 5: ETL Completed Code |
| ETL (EXTRACT) Completed Code for 311 dataset (updated) |
| import pandas as pd  from sodapy import Socrata  data\_url='data.cityofnewyork.us' # The Host Name for the API endpoint (the https:// part will be added automatically)  data\_set='erm2-nwe9' # The data set at the API endpoint (311 data in this case)  app\_token='fRtgN6KKZ5bEIkbEB12tbzK7T' # The app token created in the prior steps  client = Socrata(data\_url,app\_token) # Create the client to point to the API endpoint  # Set the timeout to 60 seconds  client.timeout = 60  metadata = client.get\_metadata(data\_set)  [x['name'] for x in metadata['columns']]  ['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency Name', 'Complaint Type', 'Descriptor',  'Location Type', 'Incident Zip', 'Incident Address', 'Street Name', 'Cross Street 1', 'Cross Street 2',  'Intersection Street 1', 'Intersection Street 2', 'Address Type', 'City', 'Landmark', 'Facility Type',  'Status', 'Due Date', 'Resolution Description', 'Resolution Action Updated Date', 'Community Board', 'BBL',  'Borough', 'X Coordinate (State Plane)', 'Y Coordinate (State Plane)', 'Open Data Channel Type',  'Park Facility Name', 'Park Borough', 'Vehicle Type', 'Taxi Company Borough', 'Taxi Pick Up Location',  'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp', 'Bridge Highway Segment', 'Latitude',  'Longitude', 'Location', 'Zip Codes', 'Community Districts', 'Borough Boundaries', 'City Council Districts',  'Police Precincts']  # Retrieve the first 2000 results returned as JSON object from the API where the range falls within the years  where\_clause = "complaint\_type = 'Traffic Signal Condition' AND created\_date BETWEEN '2018-01-01' AND '2023-12-31'"  results = client.get(data\_set, where=where\_clause,limit=100000000)  # Convert the list of dictionaries to a Pandas data frame  df = pd.DataFrame.from\_records(results)  # Save the data frame to a CSV file  df.to\_csv("my\_311\_data.csv")  row = 1  df.at[row, 'incident\_zip'] = 11234  # Assuming df is your DataFrame  df['unique\_key'] = pd.to\_numeric(df['unique\_key'], errors='coerce')  # Replace NaN values with a default integer value or any other appropriate handling  default\_value = 0 # Replace with your desired default value  df['unique\_key'].fillna(default\_value, inplace=True)  # Convert the column to integers  df['unique\_key'] = df['unique\_key'].astype(int)  df['incident\_zip'] = pd.to\_numeric(df['incident\_zip'], errors='coerce')  df['incident\_zip'].fillna(default\_value, inplace=True)  df['incident\_zip'] = df['incident\_zip'].astype(int)  df['x\_coordinate\_state\_plane'] = pd.to\_numeric(df['x\_coordinate\_state\_plane'], errors='coerce')  df['x\_coordinate\_state\_plane'].fillna(default\_value, inplace=True)  df['x\_coordinate\_state\_plane'] = df['x\_coordinate\_state\_plane'].astype(int)  df['y\_coordinate\_state\_plane'] = pd.to\_numeric(df['y\_coordinate\_state\_plane'], errors='coerce')  df['y\_coordinate\_state\_plane'].fillna(default\_value, inplace=True)  df['y\_coordinate\_state\_plane'] = df['y\_coordinate\_state\_plane'].astype(int)  df['bbl'] = pd.to\_numeric(df['bbl'], errors='coerce')  df['bbl'].fillna(default\_value, inplace=True)  df['bbl'] = df['bbl'].astype(int) |
| Data Profiling Updated |
| from ydata\_profiling import ProfileReport  import pandas as pd  #311 data  df = pd.read\_csv('/Users/shariahoque/Desktop/Fall\_2023/CIS 4400/Project /Raw\_Data\_311\_Service\_Requests\_from\_2010\_to\_Present.csv')  data\_report=ProfileReport(df)  data\_report.to\_file('Raw\_Data\_311\_Service\_Requests.html')  # Collision data  df2 = pd.read\_csv('/Users/shariahoque/Desktop/Fall\_2023/CIS 4400/Project /collision\_raw\_data.csv')  data\_report=ProfileReport(df2)  data\_report.to\_file('collision\_raw\_data.html') |
| ETL ( EXTRACT) Completed Code for Collisions Dataset (updated) |
| data\_url2='data.cityofnewyork.us' # The Host Name for the API endpoint (the https:// part will be added automatically)  data\_set2='h9gi-nx95' # The data set at the API endpoint (311 data in this case)  app\_token2='fRtgN6KKZ5bEIkbEB12tbzK7T' # The app token created in the prior steps  client = Socrata(data\_url2,app\_token2) # Create the client to point to the API endpoint  # Set the timeout to 60 seconds  client.timeout = 60  # Retrieve the first 2000 results returned as JSON object from the API  # The SoDaPy library converts this JSON object to a Python list of dictionaries  where\_clause2 = "date\_extract\_y(crash\_date) BETWEEN 2018 AND 2023"  results2 = client.get(data\_set2, where=where\_clause2, limit=100000000)  # Convert the list of dictionaries to a Pandas data frame  df2 = pd.DataFrame.from\_records(results2)  # Save the data frame to a CSV file  df2.to\_csv("collisions.csv") |
| Transform Step 311 complaint dataset |
| 311 complaint **location dimension** |
| WITH  location AS (  SELECT  DISTINCT  ifnull(borough, City) as borough,  ifnull(City, borough) as city,  incident\_zip,  -- ifnull(intersection\_street\_1,cross\_street\_1) as intersection\_street\_1,  -- ifnull(intersection\_street\_2,cross\_street\_2) as intersection\_street\_2,  TRIM(COALESCE(intersection\_street\_1, cross\_street\_1) ||COALESCE(' ') || COALESCE(intersection\_street\_2, cross\_street\_2)) AS street\_address,  longitude,  latitude    FROM  cis4400project-403800.projectDatasets.311\_service\_requests)  SELECT  ROW\_NUMBER() OVER () AS location\_id\_SKs,  \*  -- ,case when cross\_street\_1 = intersection\_street\_1 then 1 else 0 end as borough\_equal\_city  FROM  location  order by location\_id\_SKs |
| 311 complaint **Complaint Type dimension** |
| {{ config(materialized="table") }}  with  complaint\_type as (  select distinct (complaint\_type), descriptor  from `cis4400project-403800.projectDatasets.311\_service\_requests`  )  select row\_number() over () as complaint\_type\_descriptor\_SKs, \*  from  complaint\_type  order by complaint\_type\_descriptor\_SKs |
| 311 complaint **STATUS dimension** |
| {{ config(materialized="table") }}  WITH  dim\_status AS (  SELECT  status,  DENSE\_RANK() OVER () AS Status\_SKs  FROM  cis4400project-403800.projectDatasets.311\_service\_requests )  SELECT  DISTINCT Status\_SKs,  status  FROM  dim\_status  order by Status\_SKs |
| 311 complaint **Agency dimension** |
| {{ config(materialized="table") }}  with agencies as (  select distinct  agency,  agency\_name  FROM cis4400project-403800.projectDatasets.311\_service\_requests  )  select  row\_number() over () as agency\_ID\_SK,\*  from agencies |
| Conformed date dimension: |
| {{ config(materialized="table") }}  WITH date\_data AS (  SELECT  d,  EXTRACT(YEAR FROM d) AS year,  EXTRACT(WEEK FROM d) AS year\_week,  EXTRACT(DAY FROM d) AS year\_day,  EXTRACT(YEAR FROM d) AS fiscal\_year,  FORMAT\_DATE('%Q', d) AS fiscal\_qtr,  EXTRACT(MONTH FROM d) AS month,  FORMAT\_DATE('%B', d) AS month\_name,  FORMAT\_DATE('%w', d) AS week\_day,  FORMAT\_DATE('%A', d) AS day\_name,  CASE WHEN FORMAT\_DATE('%A', d) IN ('Sunday', 'Saturday') THEN 0 ELSE 1 END AS day\_is\_weekday  FROM  UNNEST(GENERATE\_DATE\_ARRAY('2018-01-01', '2024-01-01', INTERVAL 1 DAY)) AS d  )  SELECT  ROW\_NUMBER() OVER() AS date\_dim\_id,  FORMAT\_DATE("%Y%m%d", d) AS date\_integer,  d AS full\_date,  year,  year\_week,  year\_day,  month,  month\_name,  week\_day,  day\_name  FROM  date\_data  order by date\_dim\_id |
| ~~311 complaint~~ **~~Date dimension~~** |
| ~~SELECT~~  ~~distinct date\_dim.\*~~  ~~FROM~~  ~~cis4400project-403800.projectDatasets.vehicle\_collisions~~  ~~left join {{ ref('date\_dim') }} as date\_dim on DATE(crash\_date) = full\_date~~  ~~order by date\_dim\_id~~ |
|  |
| 311 complaint **FACT Table** |
| {{ config(materialized="table") }}  -- list of 311 dimentions:  -- 311 dimentions:  -- Location  -- Complaint type  -- status  -- agency  -- date  -- -- non-null location  with  all\_complaints\_data as (select \* from {{ ref("All\_complaint\_data") }}),  complaint\_location as (select \* from {{ ref("311\_location") }}),  complaint\_type as (select \* from {{ ref("complaint\_type") }}),  status as (select \* from {{ ref("Status") }}),  agecny as (select \* from {{ ref("agency") }}),  dates as (select \* from {{ ref("date\_dim") }}),  all\_ids as (select  complaint\_type\_descriptor\_sks,  location\_id\_sks,  agecny.agency\_id\_sk,  status\_sks,  date\_dim\_id  from all\_complaints\_data  left join  complaint\_location  on all\_complaints\_data.latitude = complaint\_location.latitude  or (all\_complaints\_data.latitude is null and complaint\_location.latitude is null)  and all\_complaints\_data.longitude = complaint\_location.longitude  or (all\_complaints\_data.longitude is null and complaint\_location.longitude is null)  and all\_complaints\_data.borough = complaint\_location.borough  or (all\_complaints\_data.borough is null and complaint\_location.borough is null)  and all\_complaints\_data.city = complaint\_location.city  or (all\_complaints\_data.city is null and complaint\_location.city is null)  and all\_complaints\_data.incident\_zip = complaint\_location.incident\_zip  or (  all\_complaints\_data.incident\_zip is null  and complaint\_location.incident\_zip is null  )  and all\_complaints\_data.street\_address = complaint\_location.street\_address  or (  all\_complaints\_data.street\_address is null  and complaint\_location.street\_address is null  )  left join  complaint\_type  on all\_complaints\_data.complaint\_type = complaint\_type.complaint\_type  and all\_complaints\_data.descriptor = complaint\_type.descriptor  left join status on all\_complaints\_data.status = status.status  left join dates on all\_complaints\_data.created\_date = dates.full\_date  left join agecny on all\_complaints\_data.agency = agecny.agency)  select row\_number()over() as Main\_Ids\_Sks, \*  from all\_ids  order by Main\_Ids\_Sks |
| 311 entire data (It is created to build fact table) |
| {{ config(materialized="table") }}  -- non-null location  with  complaint\_data as (  select  unique\_key,  cast(created\_date as date) created\_date,  cast(closed\_date as date) closed\_date,  agency,  agency\_name,  complaint\_type,  descriptor,  status,  incident\_zip,  ifnull(borough, city) as borough,  ifnull(city, borough) as city,  TRIM(COALESCE(intersection\_street\_1, cross\_street\_1) ||COALESCE(' ') || COALESCE(intersection\_street\_2, cross\_street\_2)) AS street\_address,  latitude,  longitude  from cis4400project-403800.projectDatasets.311\_service\_requests  )  select  ROW\_NUMBER() over() as complaint\_data\_SKs,\*  from complaint\_data  where incident\_zip is not null |
| Transform Step Collision dimensions and Facts |
| Location Dimension |
| {{ config(materialized="table") }}  WITH  location AS (  SELECT  DISTINCT latitude,  longitude,  -- on\_street\_name,  -- off\_street\_name,  -- ifnull(cross\_street\_name, on\_street\_name) as cross\_street\_name,  COALESCE(cross\_street\_name, on\_street\_name) ||' ' || COALESCE(on\_street\_name, ' ') || ' '||COALESCE(off\_street\_name, ' ') AS street\_address,  zip\_code,  borough  FROM  cis4400project-403800.projectDatasets.vehicle\_collisions )  SELECT  ROW\_NUMBER() OVER ( ) AS collision\_location\_SK,  \*  FROM  location  order by collision\_location\_SK |
| Time VIEW |
| {{ config(materialized="table") }}  WITH dim\_time AS (  SELECT  t,  EXTRACT(HOUR FROM t) AS hour,  EXTRACT(MINUTE FROM t) AS minute  FROM  UNNEST(GENERATE\_TIMESTAMP\_ARRAY('2023-01-01T00:00:00', '2023-01-01T23:59:59', INTERVAL 1 MINUTE)) AS t  )  SELECT  ROW\_NUMBER() OVER () AS hour\_id\_sk,  FORMAT\_TIMESTAMP('%H:%M', t) AS real\_time,  hour,  minute,  CASE  WHEN hour >= 6 AND hour < 12 THEN 'Morning'  WHEN hour >= 12 AND hour < 18 THEN 'Afternoon'  ELSE 'Night'  END AS time\_of\_day  FROM  dim\_time  order by hour\_id\_sk |
| Collision Time dimensions |
| {{ config(materialized="table") }}  SELECT DISTINCT time\_dim.\*, crash\_time    FROM cis4400project-403800.projectDatasets.vehicle\_collisions  left join {{ ref('time\_dim') }} as time\_dim on crash\_time = real\_time |
|  |
| ~~Date Dimension (created by using date view)~~ |
| ~~SELECT~~  ~~distinct date\_dim.\*~~    ~~FROM~~  ~~cis4400project-403800.projectDatasets.311\_service\_requests~~  ~~left join {{ ref('date\_dim') }} as date\_dim on DATE(created\_date) = full\_date~~  ~~order by date\_dim\_id~~ |
| Vehicles Dimension |
| {{ config(materialized="table") }}  WITH  vehicles AS (  SELECT  DISTINCT contributing\_factor\_vehicle\_1,  contributing\_factor\_vehicle\_2,  -- COALESCE(contributing\_factor\_vehicle\_1,' ') || ' ' || COALESCE(contributing\_factor\_vehicle\_2, ' ') AS final\_contributing\_factor\_vehicle,  vehicle\_type\_code1,  vehicle\_type\_code2,  trim (COALESCE(vehicle\_type\_code1,' ') || ' ' || COALESCE(vehicle\_type\_code2, ' ') || ' ' || COALESCE(vehicle\_type\_code2, ' ') || ' ' || COALESCE(vehicle\_type\_code\_3, ' ') || ' ' || COALESCE(vehicle\_type\_code\_4, ' ')) AS final\_vehicle\_type  FROM  cis4400project-403800.projectDatasets.vehicle\_collisions )  SELECT  ROW\_NUMBER() OVER () AS vehicle\_collision\_ID\_SK,  \*  FROM  vehicles  order by vehicle\_collision\_ID\_SK |
| Collisions Fact |
| {{ config(materialized="table") }}  -- list of collisions dimentions:  -- all data  -- Location  -- time  -- vehicles  -- date  -- -- non-null location  WITH  all\_collision\_data AS (SELECT \* FROM {{ ref("all\_Collisions\_data") }}),  collisions\_location AS (SELECT \* FROM {{ ref("collisions\_location") }}),  types\_of\_vehicles AS (SELECT \* FROM {{ ref("vehicles") }}),  time\_dim AS (SELECT \* FROM {{ ref("time\_dim") }}),  dates AS (SELECT \* FROM {{ ref("date\_dim") }}),  all\_ids as( SELECT  date\_dim\_id,  hour\_id\_sk,  collision\_id,  collision\_location\_sk,  vehicle\_collision\_id\_sk,  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,  total\_killed,  total\_harmed\_killed\_injured  FROM  all\_collision\_data  LEFT JOIN  collisions\_location ON  (  all\_collision\_data.latitude = collisions\_location.latitude OR  (all\_collision\_data.latitude IS NULL AND collisions\_location.latitude IS NULL)  )  AND (  all\_collision\_data.longitude = collisions\_location.longitude OR  (all\_collision\_data.longitude IS NULL AND collisions\_location.longitude IS NULL)  )  AND (  all\_collision\_data.borough = collisions\_location.borough OR  (all\_collision\_data.borough IS NULL AND collisions\_location.borough IS NULL)  )  AND (  all\_collision\_data.zip\_code = collisions\_location.zip\_code OR  (all\_collision\_data.zip\_code IS NULL AND collisions\_location.zip\_code IS NULL)  )  AND (  all\_collision\_data.street\_address = collisions\_location.street\_address OR  (all\_collision\_data.street\_address IS NULL AND collisions\_location.street\_address IS NULL)  )  LEFT JOIN  types\_of\_vehicles ON  (  all\_collision\_data.final\_vehicle\_type = types\_of\_vehicles.final\_vehicle\_type OR  (all\_collision\_data.final\_vehicle\_type IS NULL AND types\_of\_vehicles.final\_vehicle\_type IS NULL)  )  AND (  all\_collision\_data.contributing\_factor\_vehicle\_1 = types\_of\_vehicles.contributing\_factor\_vehicle\_1 OR  (all\_collision\_data.contributing\_factor\_vehicle\_1 IS NULL AND types\_of\_vehicles.contributing\_factor\_vehicle\_1 IS NULL)  )  AND (  all\_collision\_data.contributing\_factor\_vehicle\_2 = types\_of\_vehicles.contributing\_factor\_vehicle\_2 OR  (all\_collision\_data.contributing\_factor\_vehicle\_2 IS NULL AND types\_of\_vehicles.contributing\_factor\_vehicle\_2 IS NULL)  )  LEFT JOIN  time\_dim ON all\_collision\_data.crash\_time = time\_dim.real\_time  LEFT JOIN  dates ON all\_collision\_data.crash\_date = dates.full\_date)  select row\_number() over() as main\_ids, \*  from all\_ids |
| Merged Location |
| {{ config(materialized="table") }}  WITH  collisions\_location AS (SELECT \* FROM {{ ref("collisions\_location") }}),  raw\_311\_location AS (SELECT \* FROM {{ ref("311\_location") }}),  all\_location AS (  SELECT  -- COALESCE(raw\_311\_location.street\_address, collisions\_location.street\_address) AS street\_address,  -- COALESCE(raw\_311\_location.borough, collisions\_location.borough) AS borough,  -- COALESCE(raw\_311\_location.incident\_zip, collisions\_location.zip\_code) AS zip\_code,  -- COALESCsE(raw\_311\_location.latitude, collisions\_location.latitude) AS latitude,  -- COALESCE(raw\_311\_location.longitude, collisions\_location.longitude) AS longitude,  collisions\_location.latitude AS collision\_latitude,  collisions\_location.longitude AS collision\_longitude,  TRIM(collisions\_location.street\_address) AS collisions\_street\_address,  collisions\_location.borough AS collisions\_borough,  collisions\_location.zip\_code AS collisions\_zip\_code,  raw\_311\_location.latitude AS complaint\_latitude,  raw\_311\_location.longitude AS complaint\_longitude,  TRIM(raw\_311\_location.street\_address) AS complaint\_street\_address,  raw\_311\_location.borough AS complaint\_borough,  raw\_311\_location.incident\_zip AS complaint\_zip\_code  FROM  collisions\_location  FULL JOIN  raw\_311\_location  ON (  raw\_311\_location.incident\_zip = collisions\_location.zip\_code  OR (  raw\_311\_location.incident\_zip IS NULL  AND collisions\_location.zip\_code IS NULL  )  )  AND COALESCE(raw\_311\_location.borough, '') = COALESCE(collisions\_location.borough, '')  where incident\_zip is not null and zip\_code is not null  )  SELECT  ROW\_NUMBER() OVER () AS locations\_sks,  -- COUNT(CASE WHEN complaint\_zip\_code IS  -- NULL THEN 1 END)  \*  FROM  all\_location  ORDER BY  locations\_sks |
| MERGED Date |
| {{ config(materialized="table") }}  WITH  collisions\_date AS (SELECT \* FROM {{ ref('collisions\_date') }}),  complaint\_date AS (SELECT \* FROM {{ ref('complaint\_date') }})  SELECT DISTINCT  collisions\_date.date\_dim\_id,  collisions\_date.full\_date,  collisions\_date.year,  collisions\_date.year\_week,  collisions\_date.year\_day,  collisions\_date.fiscal\_year,  collisions\_date.fiscal\_qtr,  collisions\_date.month,  collisions\_date.month\_name,  collisions\_date.week\_day,  collisions\_date.day\_name,  collisions\_date.day\_is\_weekday  FROM  collisions\_date  FULL JOIN  complaint\_date ON collisions\_date.full\_date = complaint\_date.full\_date  order by date\_dim\_id |

# KPI Visualizations







# Reference List

[www.fortnite.com](http://www.fortnite.com)

<https://help.tableau.com/current/pro/desktop/en-us/examples_googlebigquery.htm>

<https://help.tableau.com/current/pro/desktop/en-us/examples_googlebigquery.htm>

<https://colab.research.google.com/notebooks/snippets/gcs.ipynb>

<https://cloud.google.com/bigquery/docs/cloud-storage-transfer>

<https://holowczak.com/getting-started-with-nyc-opendata-and-the-socrata-api/>

# Attendance:

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Time | Meeting About | Attendance |
| 9/17/2023 | 3:00 pm - 4:00 pm | * KPIs * Purpose of the project * Plan for future meetings | Kelvin Nguyen  Gabriela Sanchez  Sharia Hoque  Dorwin Liang  Joann Mei |
| 10/8/2023 | 10:00 AM - 11:30 AM | * KPIs * Dimensional Modeling * Star Schema * Listing out the KPIs | Kelvin Nguyen  Gabriela Sanchez  Sharia Hoque  Joann Mei |
| 10/8/2023 | 3:00 pm - 4:30 pm | * Continuation of the 10AM meeting * Fact Table and Dimension Tables | Kelvin Nguyen  Sharia Hoque  Dorwin Liang  Joann Mei |
| 10/22/2023 | 3:00 pm - 5:00 pm | * Finalized Fact Table and Dimension Tables | Kelvin Nguyen  Sharia Hoque  Dorwin Liang  Joann Mei  Gabriela Sanchez |
| 10/24/2023 | 8:30 pm | * Continuation of the previous meeting and finalizing | Kelvin Nguyen  Sharia Hoque  Dorwin Liang  Joann Mei  Gabriela Sanchez |
| 10/31/2023 | 8:00pm | * Selecting BigQuery for data warehouse * Selecting dbt for ETL * Sharing datasets with group | Kelvin Nguyen  Sharia Hoque  Dorwin Liang  Joann Mei  Gabriela Sanchez |
| 11/26/2023 | 3:00pm | * Built E and L in python, extracted data into a google cloud storage bucket and loaded it into BQ * Worked on loading the tables into BigQuery * Created dimensions and SK’s | Kelvin Nguyen  Sharia Hoque  Dorwin Liang  Joann Mei  Gabriela Sanchez |
| 12/10/2023 | 3:00pm | * Finalized Dashboard on Tableau * Made Three visualzations to answer KPI’s * The final meeting | Kelvin Nguyen  Sharia Hoque  Dorwin Liang  Joann Mei  Gabriela Sanchez |
|  |  |  |  |
|  |  |  |  |