Taxi and Limousine Complaints - Final Report

12.08.2021

Atabay Kadiroglu - ATABAY.KADIROGLU@baruchmail.cuny.edu
Carla Pena Jimenez - CARLA.PENAJIMENEZ@baruchmail.cuny.edu
Fengping Zhao - FENGPING.ZHAO@baruchmail.cuny.edu
Jason Lau - JASON.LAU1@baruchmail.cuny.edu
Kwadwo Ennin - KWADWO.ENNIN@baruchmail.cuny.edu

CIS9440 - UTA 28242
Data Warehousing and Analytics

Description of Problem

New York City is well-known for many things around the world, none less so than its yellow taxis and no-nonsense, short-fuse attitudes. In line with these, our project examines yellow cab drivers, and their rides for passengers within New York, before and during the pandemic and how taxi ridership has evolved within that period. Particularly, we want to examine this from the perspective of passenger complaints to the NYC 311 Call center. Some of the questions we are looking to investigate include:

(1) Are yellow cab drivers overcharging New Yorkers for trips?
(2) Which boroughs have the most taxi complaints in NYC?

Our data sources for this project are TLC trip records provided by the City of New York website data page. Additionally, our secondary data source is the NYC 311 service requests/ complaints call data. We are looking to merge these two (2) sources of data within Google BigQuery for the analysis involved in this project.

Dataset Links

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

https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

KPIs

1. **Comparison of passenger complaint descriptions by longitude and latitude over time**
2. **Number of passenger complaints and percentage of increase/decrease over time in years, and months.**
3. **Comparison of average trip distance by borough - date filter (**pre-pandemic vs. post pandemic**)**
4. **Comparison of average fare amount by borough - date filter**
5. **Comparison of average tip amount by borough - date filter**
6. **Which day of the week has the highest number of trips and complaints**
7. **Passenger group size by day of the week**

DIMENSIONS

- **LOCATION**

    -LOCATION_ID
    -STREET
    -BOROUGH
    -ZIPCODE
    -PICKUP_ID
    -DROPOFF_ID

-**DATE**

    -DATE_ID
    -DAYOFWEEK_ID
    -DAY_ID
    -WEEK_ID
    -MONTH_ID
    -YEAR_ID

-COMPLAINT_TYPE

    -COMPLAINT_ID

-COMPLAINT_TYPE

-COMPLAINT_DESC

-COMPLAINT_TYPE
-COMPLAING_DESC
-INCIDENT_ZIP
-INCIDENT_ADDRESS
-STREET_NAME
-CITY
-BOROUGH

-AGENCY TYPE

-AGENCY NAME

-PAYMENTS

- PAYMENT_ID
- PAYMENT_TYPE
- PAYMENT_AMOUNT
- TIP_AMOUNT
- TAX_AMOUNT
- TOLLS_AMOUNT
- MTA_TAX
- EXTRA
- IMPORVEMENT_SURCHARGE
- TOTAL_AMOUNT

TRIP

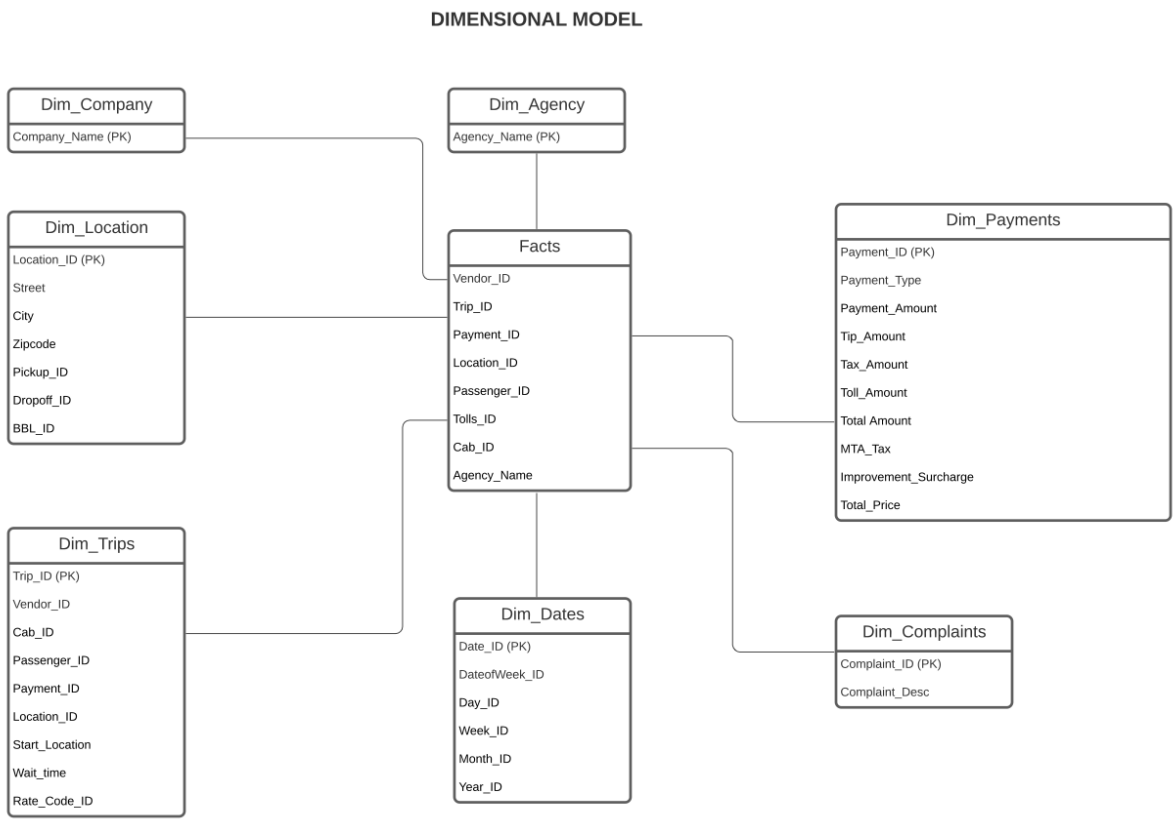
- TRIP_ID
- VENDOR_ID
- PASSENGER_ID
- PAYMENT_ID
- WAIT_TIME
- TRIP START TIMESTAMP
- TRIP END TIMESTAMP
- START LOCATION
- LOCATION_ID
- RATE_CODE_ID
- PASSENGER_COUNT
- TRIP_DISTANCE

FACTS

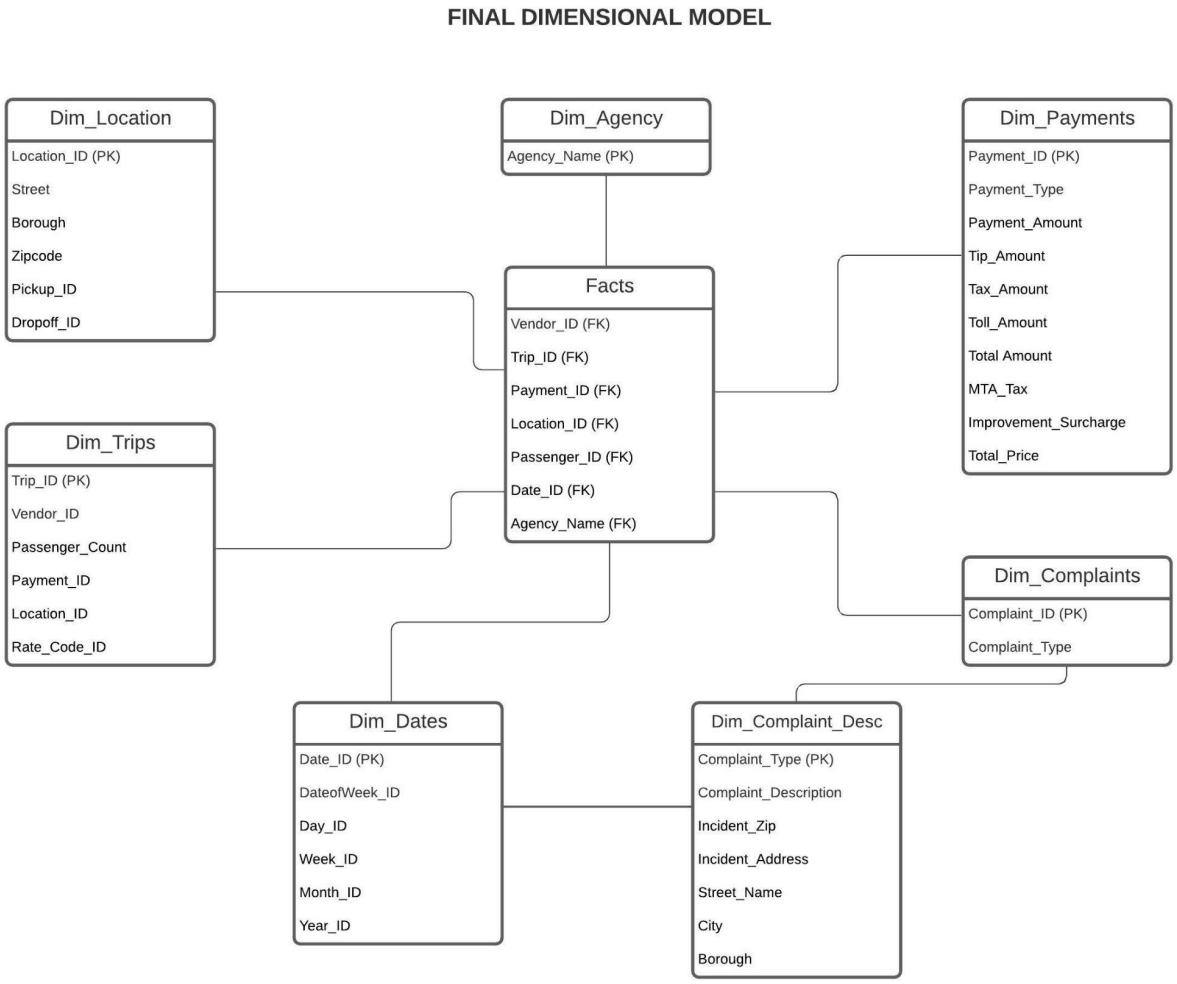
- VENDOR_ID

- TRIP_ID
- PAYMENT_ID
- LOCATION_ID
- DATE_ID
- PASSENGER_ID
- AGENCY_NAME

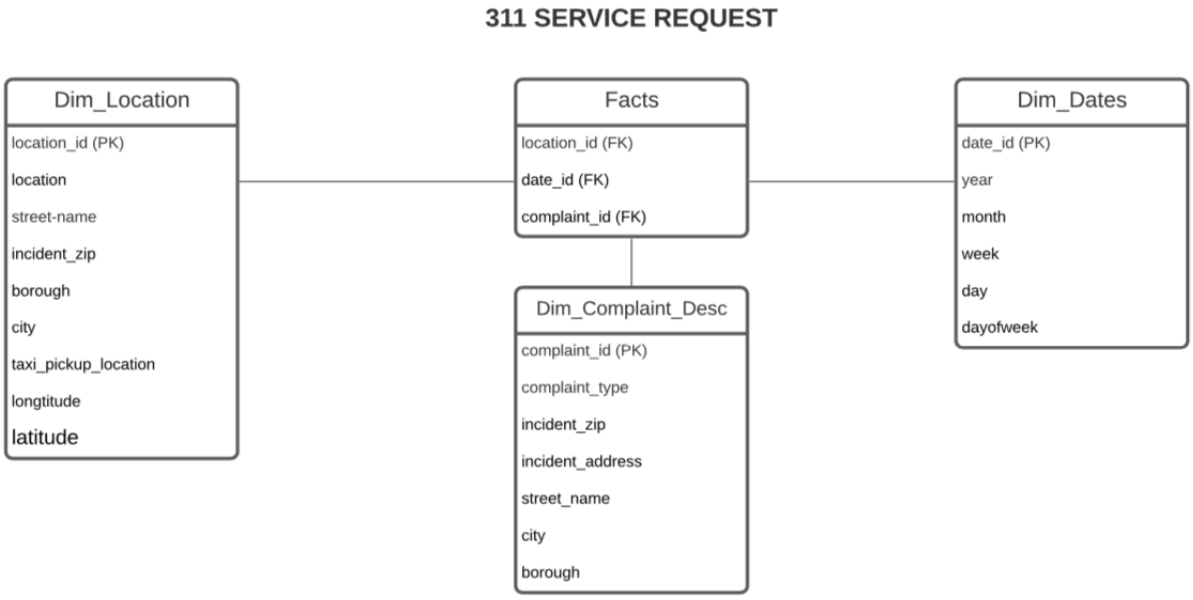
DIMENSIONAL MODEL DRAFT 1

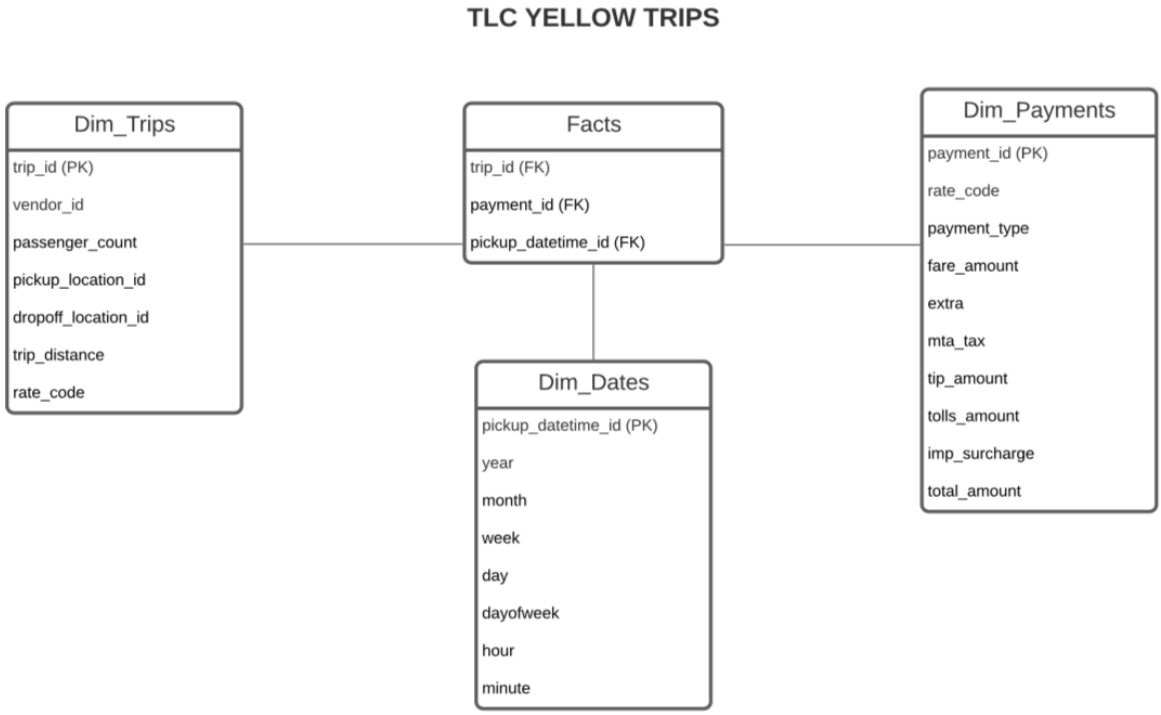


DIMENSIONAL MODEL DRAFT 2



DIMENSIONAL MODEL FINAL

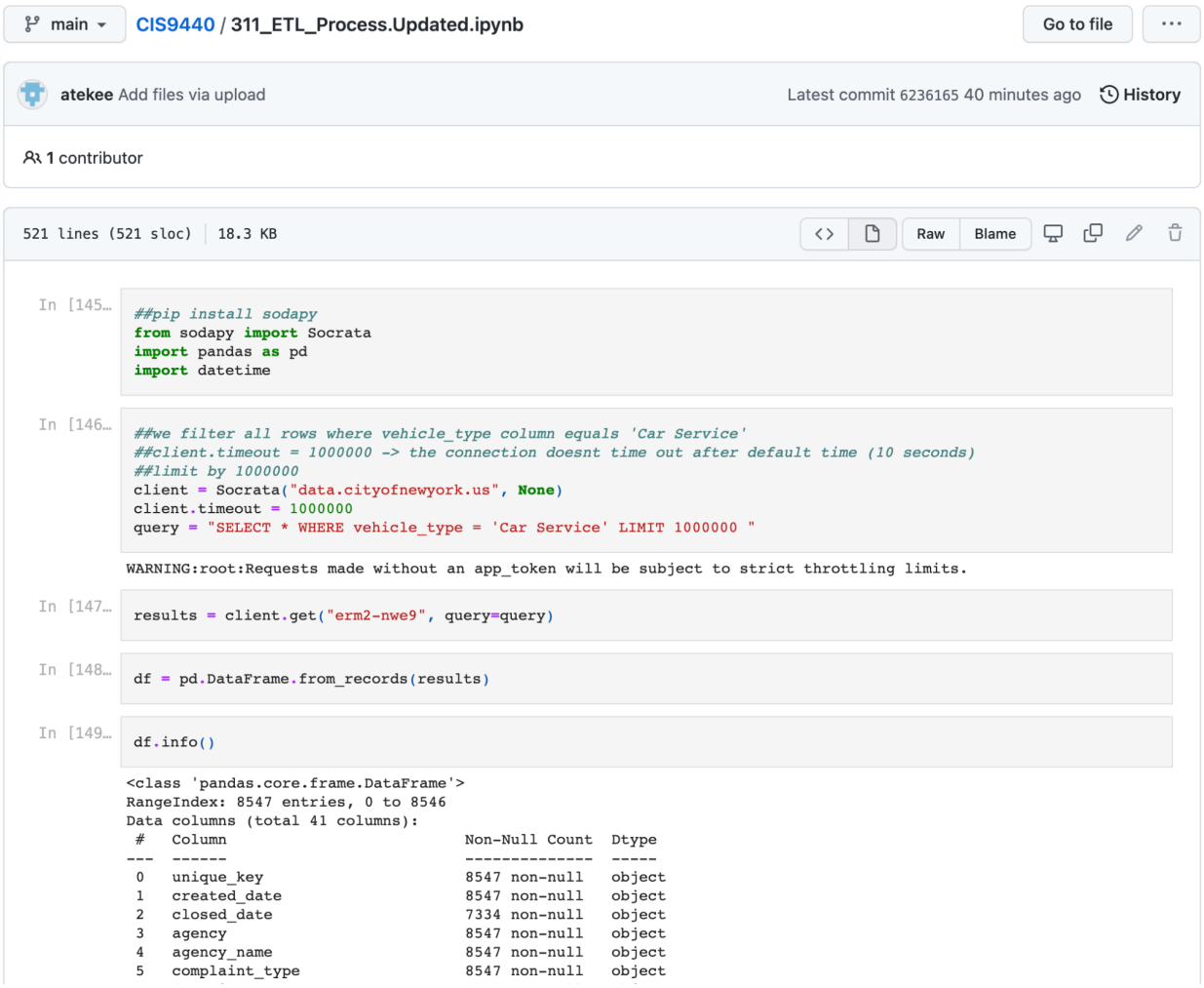




ETL Process and Tools

I. We decided to use Google Big Query and Google Cloud Storage because it is a collaborative platform which enables the group to work simultaneously on the tables and queries. Google also provides Data Studio for data visualizations and dashboards that link with bigquery. We think it is a very powerful and convenient tool as it aids in communicating our project's results.

II. We used Python to create and clean up the 311 Complaints dataset via API. After getting the 311 dataset into Python, we chose the relevant columns, and filtered out the data using dropna function to parse through each column looking for important data that is missing and restricting the data to the years of 2017 to 2021, also changed the type of data.



```
 36  address_type                    6285 non-null   object
 37  bridge_highway_name             328 non-null    object
 38  bridge_highway_direction        314 non-null    object
 39  road_ramp                       296 non-null    object
 40  bridge_highway_segment          234 non-null    object
dtypes: object(41)
memory usage: 2.7+ MB
```

In [150…

```
##choose relevant columns
df = df[['unique_key', 'created_date', 'agency', 'complaint_type', 'descriptor', 'incident_zip',
'incident_address', 'street_name', 'city', 'borough', 'taxi_pick_up_location',
'latitude', 'longitude', 'location']]
```

In [151…

```
##dropping null values from these columns as I'll change data type to number
df.dropna(subset=['latitude','longitude'], inplace=True)

##incident_zip has "N/A" value it needs to be classified as NaN first, then it can be dropped.
df['incident_zip'] = pd.to_numeric(df['incident_zip'], errors='coerce')
df.dropna(subset=['incident_zip'], inplace=True)
```

In [152…

```
##change data types
df['unique_key'] = df['unique_key'].astype('int')
df['latitude'] = df['latitude'].astype('float')
df['longitude'] = df['longitude'].astype('float')
df['incident_zip'] = df['incident_zip'].astype('int')

#convert date column to datetime
df['created_date'] = pd.to_datetime(df['created_date'],
                                    format='%Y-%m-%dT%H:%M:%S.%f')
df.dtypes
```

Out[152…

```
unique_key                     int64
created_date          datetime64[ns]
agency                        object
complaint_type                object
descriptor                    object
incident_zip                   int64
incident_address              object
street_name                   object
city                          object
borough                       object
taxi_pick_up_location         object
latitude                     float64
longitude                    float64
location                      object
dtype: object
```

In [153…

```
##creating year, mont, dayofweek columns
df['year'] = df['created_date'].dt.year
df['month'] = df['created_date'].dt.month
df['day'] = df['created_date'].dt.dayofweek
```

In [154…

```
##filtering from 2017 to 2021, and creating a new df called "df311"
startdate=2017
enddate=2021
df311 = df[(df['year']>= startdate) & (df['year']< enddate)]
df311.reset_index(drop=True, inplace=True)
```

In [155…

```
##confirm the data includes values from 2017 to 2020
print(min(df311['created_date']))
print(max(df311['created_date']))
```

```
2017-01-04 15:27:21
2020-12-31 09:28:13
```

In [156…

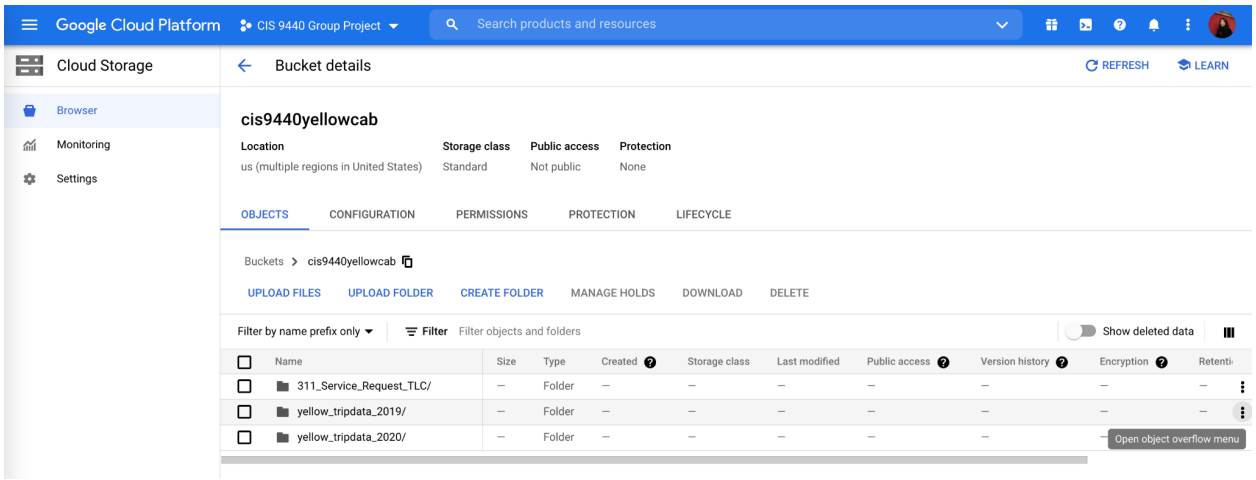
```
df311.head()
```

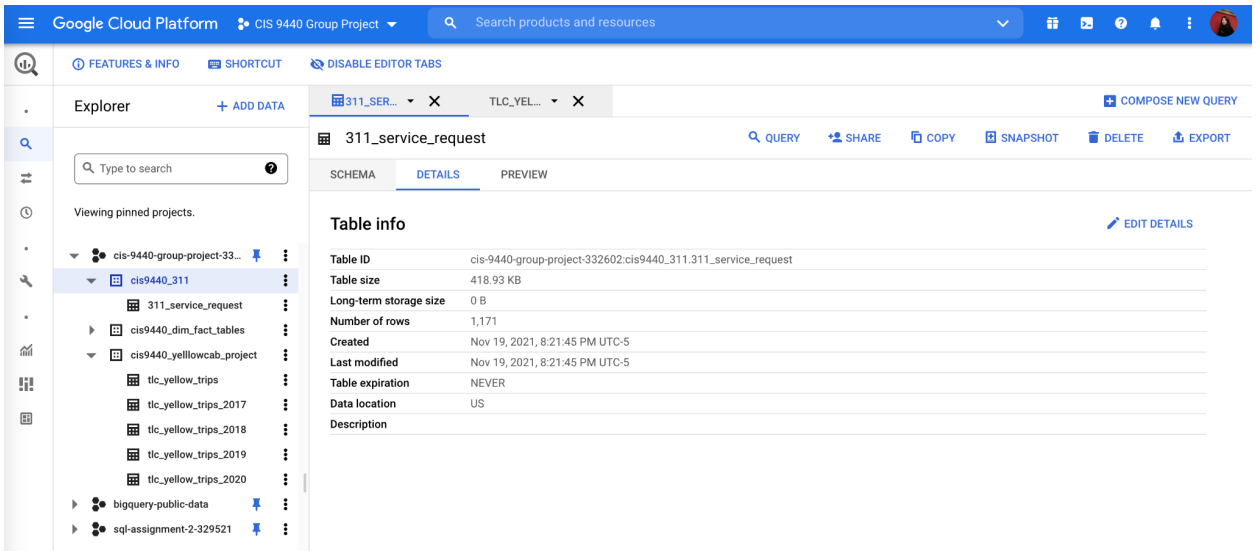
Out[156…

	unique_key	created_date	agency	complaint_type	descriptor	incident_zip	incident_address	street_name	city	borough	taxi_pick_up_location	latitu
0	45906206	2020-03-27 11:33:26	TLC	For Hire Vehicle Complaint	Car Service Company Complaint	11203	462 EAST 52 STREET	EAST 52 STREET	BROOKLYN	BROOKLYN	462 EAST 52 STREET, BROOKLYN, NY, 11203	40.6493
1	45937007	2020-04-02 16:06:02	TLC	For Hire Vehicle Complaint	Car Service Company Complaint	10467	3211 PARKSIDE PLACE	PARKSIDE PLACE	BRONX	BRONX	3211 PARKSIDE PLACE, BRONX, NY, 10467	40.8742
2	45975433	2020-04-09 12:21:01	TLC	For Hire Vehicle Complaint	Car Service Company Complaint	10005	27 WILLIAM STREET	WILLIAM STREET	NEW YORK	MANHATTAN	27 WILLIAM STREET, MANHATTAN (NEW YORK), NY, 1...	40.7055
3	45995158	2020-04-13 23:02:59	TLC	For Hire Vehicle Complaint	Car Service Company Complaint	11203	KINGS COUNTY HOSPITAL	KINGS COUNTY HOSPITAL	BROOKLYN	BROOKLYN	KINGS COUNTY HOSPITAL CENTER, BROOKLYN ,NY, 11203	40.6564
4	46018036	2020-01-30 02:15:07	TLC	For Hire Vehicle Report	Driver Report - Passenger	10456	1070 WASHINGTON AVENUE	WASHINGTON AVENUE	BRONX	BRONX	1070 WASHINGTON AVENUE, BRONX, NY, 10456	40.8278

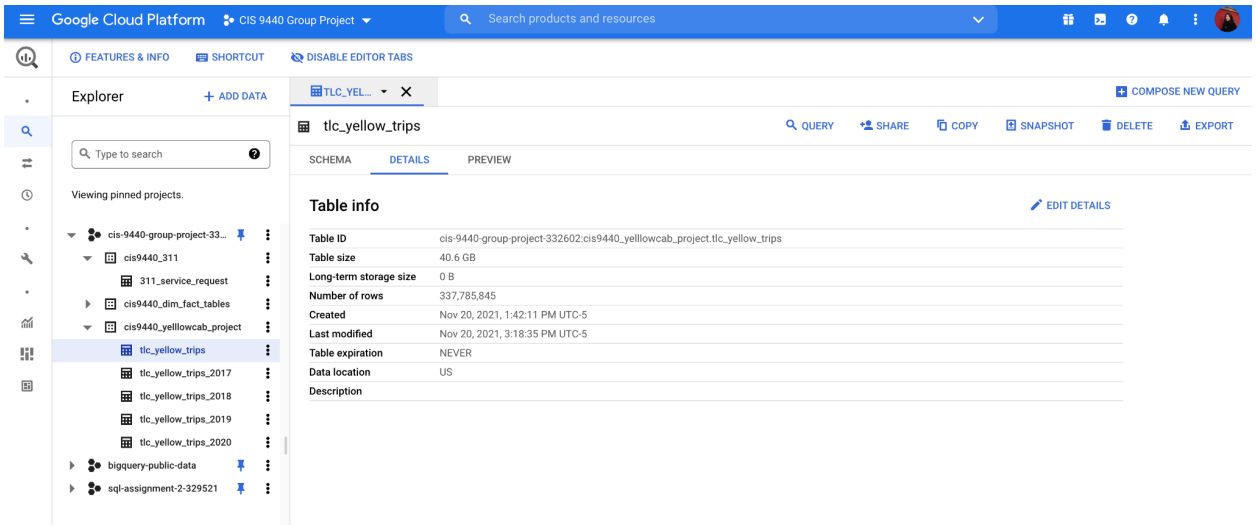
```
In [157…  ##see null values in our final 311 data
          df311.isnull().sum()

Out[157…  unique_key                0
          created_date              0
          agency                    0
          complaint_type            0
          descriptor                0
          incident_zip              0
          incident_address         34
          street_name              34
          city                    138
          borough                   0
          taxi_pick_up_location    54
          latitude                  0
          longitude                 0
          location                  0
          year                      0
          month                     0
          day                       0
          dtype: int64

In [158…  ##export the data into a csv file
          df311.to_csv ('311_Service_Request_TLC.csv', index = False, header=True)

In [ ]:
```

III. We extracted the 2019 and 2020 TLCs and the updated 311 Complaints dataset to Google Cloud Storage's bucket in our project. Then we created tables in the Google Bigquery from the GCS which is the source of our dataset.





Also, we combined the 2019 and 2020 TLC yellow cabs dataset with the 2017 and 2018 one from the bigquery public data into one table named “tlc_yellow_trips” in our project which makes it easier and more convenient for us to load the data.



For this process, we renamed below columns from 2019-2020 TLC Trips Data as below to keep them consistent with already loaded 2017-2018 TLC Data.

```
VendorID AS vendor_id,
tpep_pickup_datetime AS pickup_datetime,
tpep_dropoff_datetime AS dropoff_datetime,
PULocationID AS pickup_location_id,
```

```
RatecodeID AS rate_code,
DOLocationID AS dropoff_location_id,
improvement_surcharge AS imp_surcharge,
```

We also dropped below columns from our TLC Datasets as they didn't exist in all versions of 2017, 2018, 2019, 2020

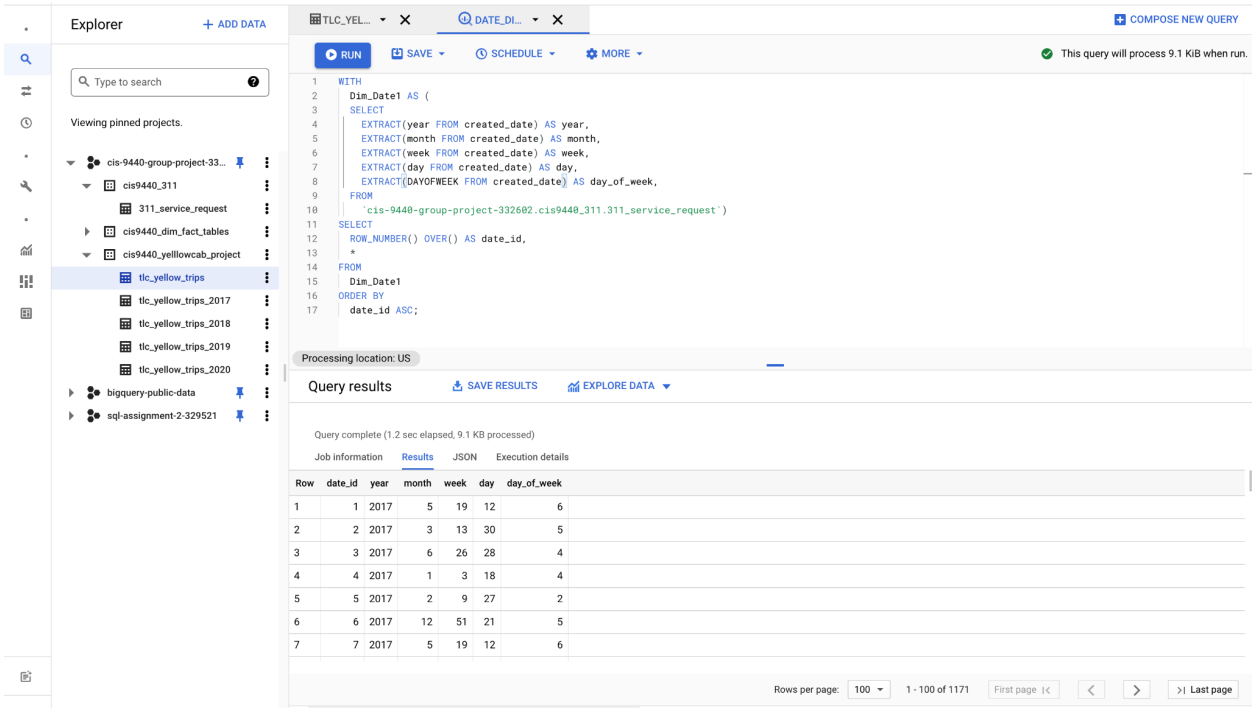
```
Pickup_longitude;
Pickup_latitude;
Dropoff_longitude;
Dropoff_latitude;
Congestion_surcharge;
store_and_fwd_flag;
```

And then, we updated the column types in 2020-2019 TLC Datasets, to be able to combine them with 2017-2018 TLC Datasets. Below code works for changing names and updating the column type.

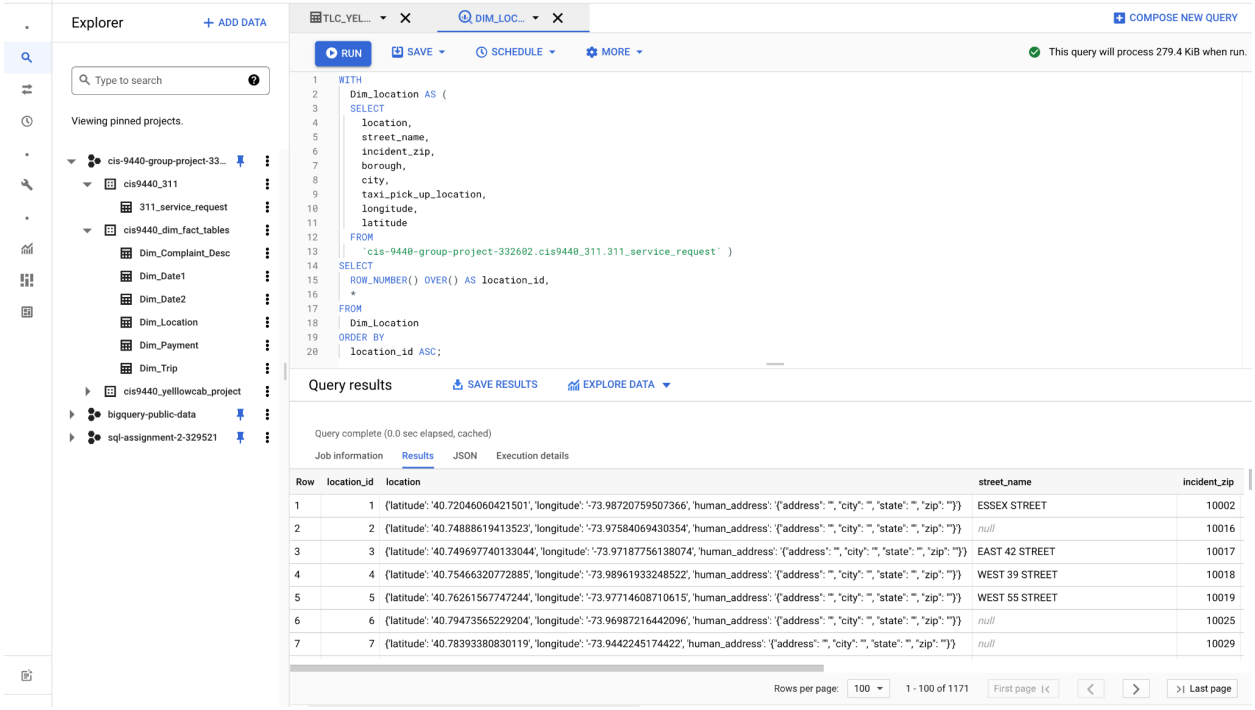
```
CAST(VendorID AS STRING ) AS vendor_id,
CAST(tpep_pickup_datetime AS DATETIME) AS pickup_datetime,
CAST(tpep_dropoff_datetime AS DATETIME) AS dropoff_datetime,
passenger_count,
CAST(trip_distance AS NUMERIC ) AS trip_distance,
CAST(RatecodeID AS STRING ) AS rate_code,
CAST(payment_type AS STRING ) AS payment_type,
CAST(fare_amount AS NUMERIC ) AS fare_amount,
CAST(extra AS NUMERIC ) AS extra,
CAST(mta_tax AS NUMERIC ) AS mta_tax,
CAST(tip_amount AS NUMERIC ) AS tip_amount,
CAST(tolls_amount AS NUMERIC ) AS tolls_amount,
CAST(improvement_surcharge AS NUMERIC ) AS imp_surcharge,
CAST(total_amount AS NUMERIC ) AS total_amount,
CAST(PULocationID AS STRING ) AS pickup_location_id,
CAST(DOLocationID AS STRING ) AS dropoff_location_id
```

IV. To create each of the six dimensional models we created six unique data frames that would take the data from their respective dataframe. Also, we created the identifiers for each of the dimensions with a suffix "_id". The screenshots below show our data dimensional model for the 311 dataset and TLC dataset.

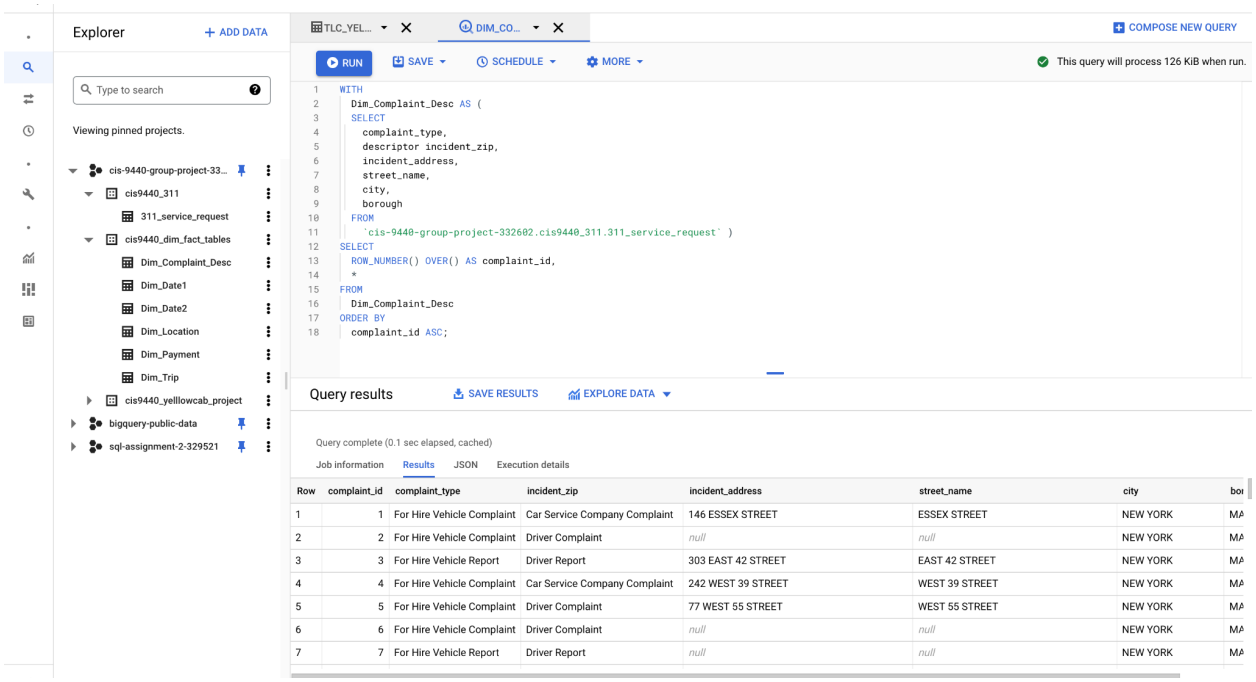
- This screenshot showcases the date dimensional model for the 311 dataset with its own Dim_Date identifier.



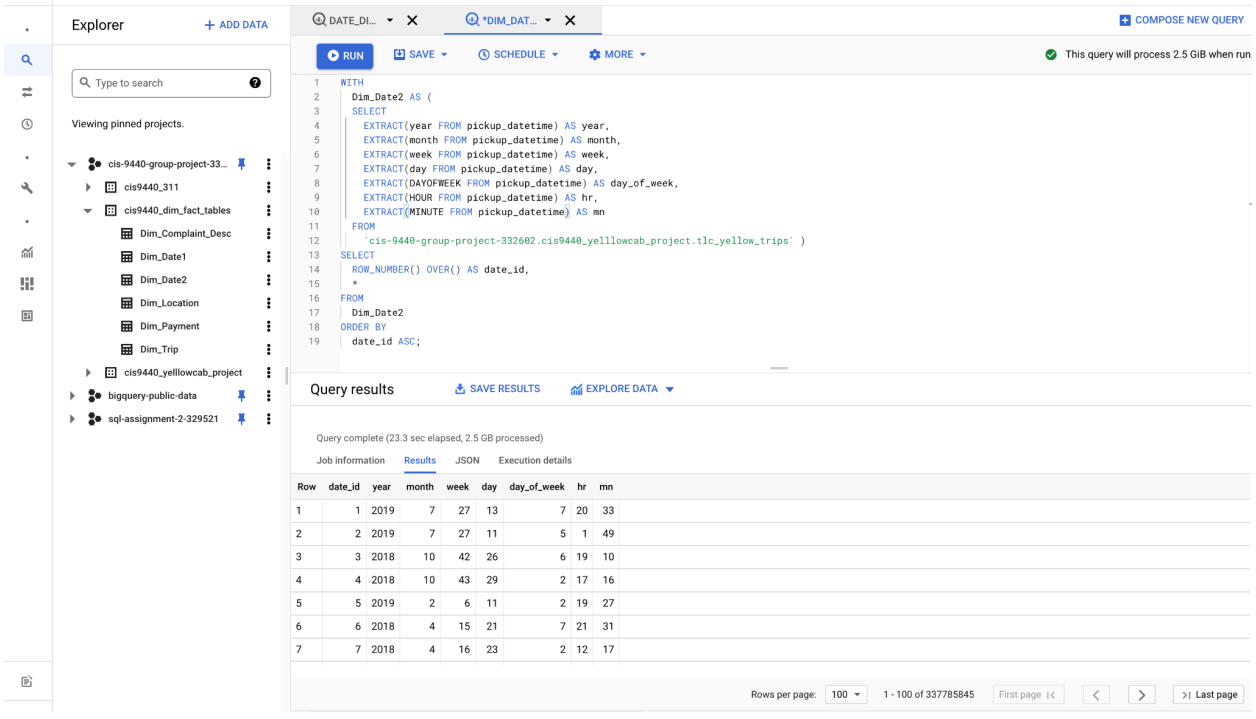
- This screenshot shows the location dimensional model for the 311 dataset with its own Dim_location identifier.



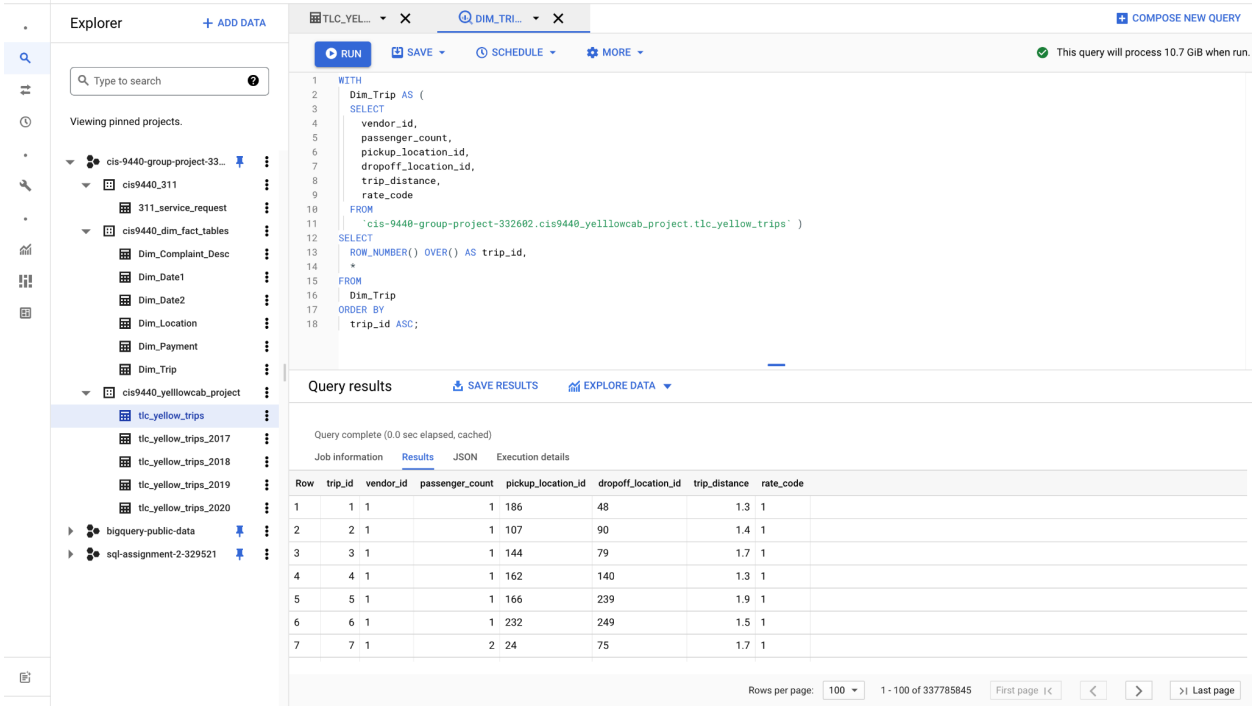
- Our third dataframe which encompasses the complaint dimensional model for the 311 dataset, along with the Dim_Complaint_Desc identifier.



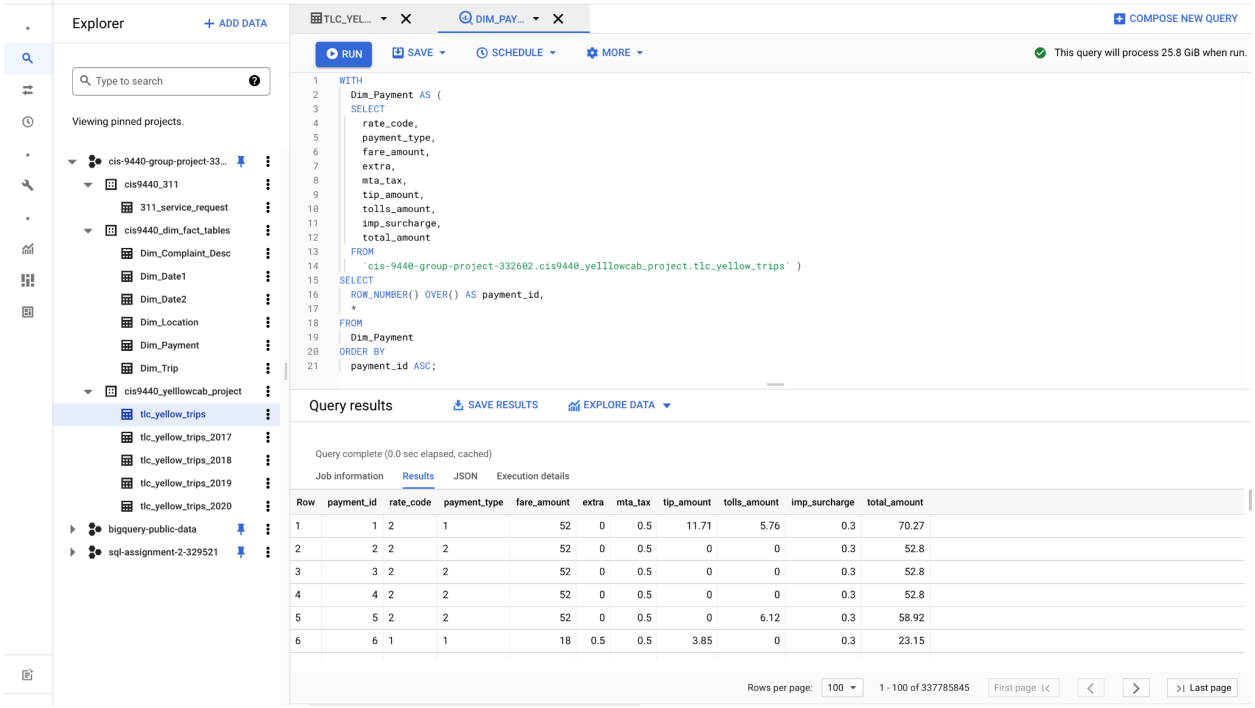
- This screenshot showcases the date dimensional model for the TLC dataset with its own Dim_Date identifier.



- This screenshot shows the trip dimensional model for the TLC dataset with its own Dim_Trip identifier.



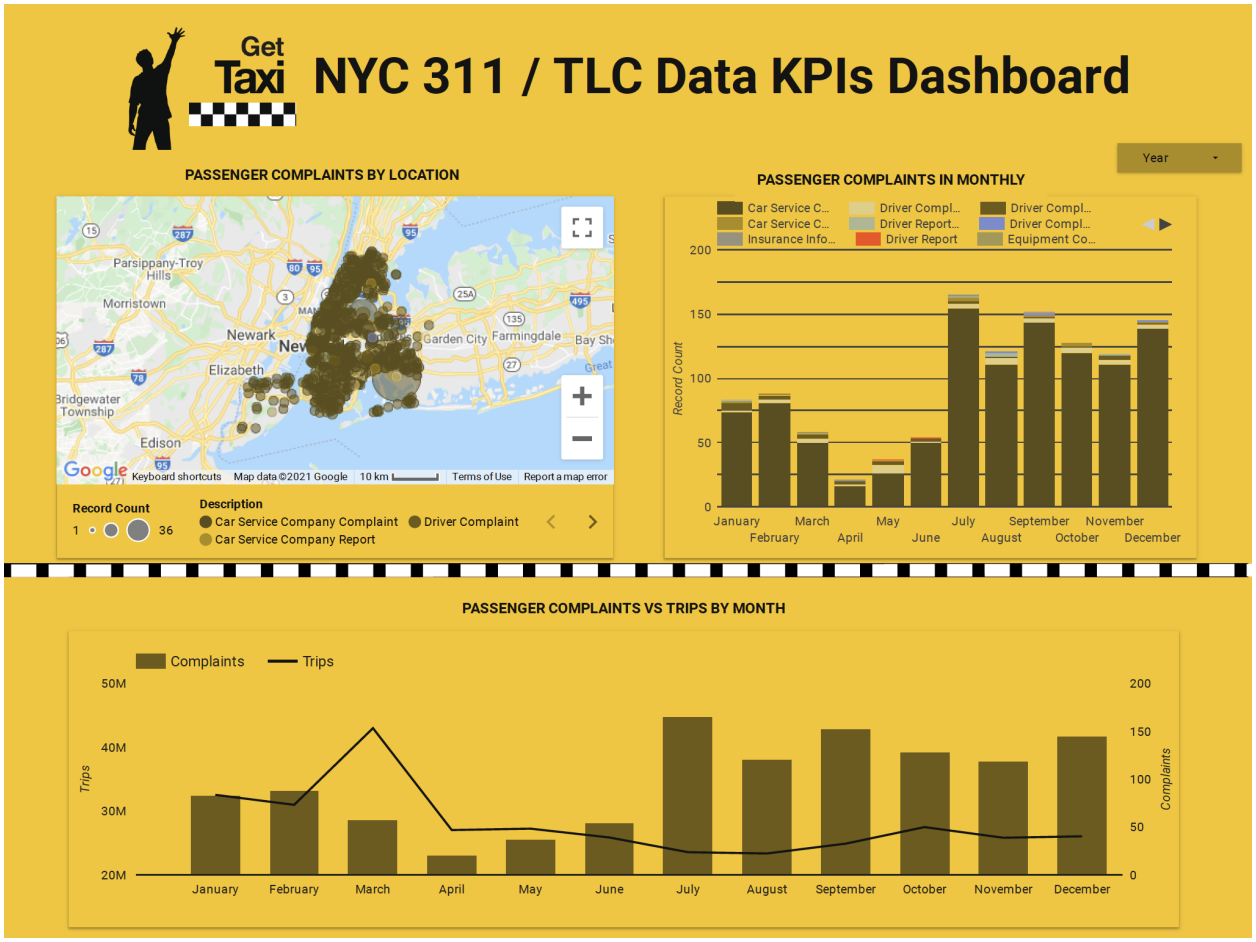
- This is our last dimensional model which showcases the payment dimensional model for the TLC dataset with its own Dim_Payment identifier.

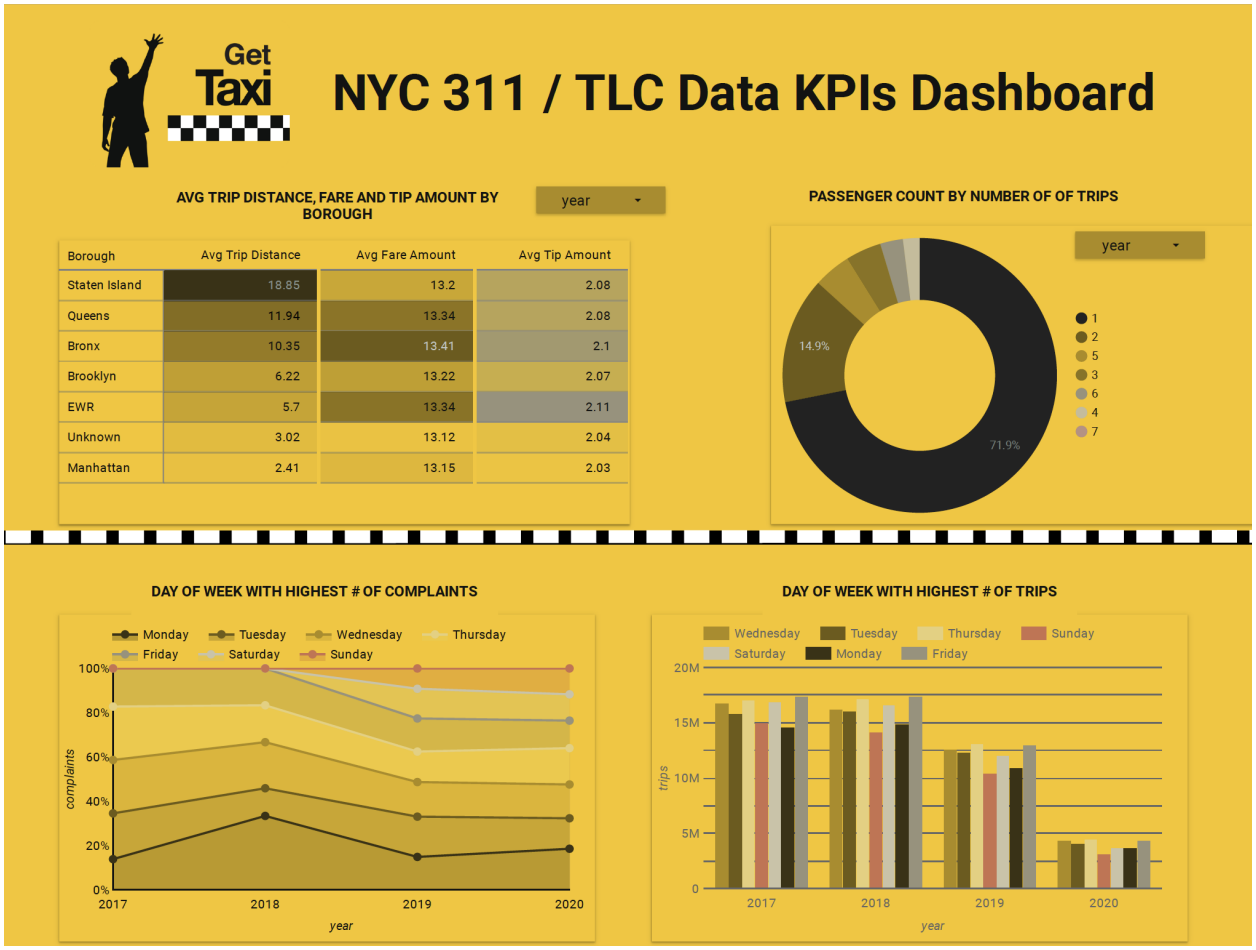


Dashboards

Please see the link below for the full detailed dashboards.

https://datastudio.google.com/s/oX20XMwvkKk





Meeting Logs

MEETING LOG			
Date	**Time**	**Attendees**	**Topics Discussed**
Thursday, September 23rd	9:00PM-10:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Selected topic • Selected which datasets to combine • Problem definition • Decided on KPI measurements
Sunday, September 26th	11:00AM-11:45AM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Expanded on Introduction • Revised KPI's and description of problem • Formatted deliverable
Thursday,	8:30PM -10:00PM	All members	• Revised first deliverabl

October 7th		(Atabay, Carla, Fengping, Jason, Kwadwo)	• Draft of dimensional model
Friday, November 5th	9:00PM - 10:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Revised dimensional model • Incorporated into final draft of proposal
Thursday, November 18th	9:30PM - 11:30PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Uploaded TLC dataset in Google Cloud Storage Platform • Created TLC Tables in Big Query via GCS
Friday, November 19th	8:00PM - 10:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• used Python to create and clean up 311 service dataset via APIs • Cleaned up in dataset • Combined TLC dataset
Saturday, November 20th	2:30PM - 5:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Cleaned up in dataset • Created Tables for Dimension Models • Incorporated ETL into reports
Thursday, December 2nd	8:30PM - 10:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Created Dashboards for each KPIs
Friday, December 3rd	8:30PM - 10:30PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Edited Dashboards in theme, positioning, colors, size, graphs, etc. • Created presentation slides for the report
Saturday, December 4th	8:30PM - 9:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	• Completed the presentation slides • Prepared for presentation individually