



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/ejm2-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
- COMPLAINING_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

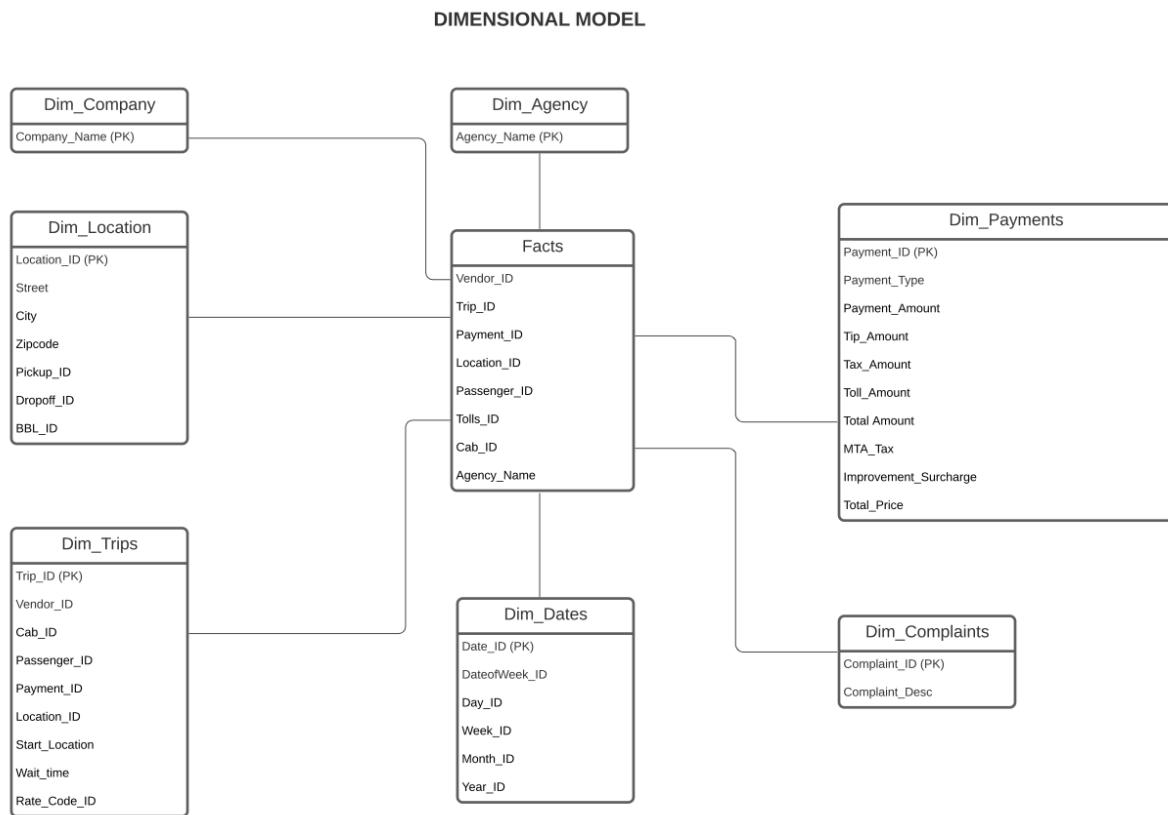
- TRIP_ID
- VENDOR_ID
- PASSENGER_ID
- PAYMENT_ID
- WAIT_TIME
- TRIP START TIMESTAMP
- TRIP END TIMESTAMP
- START LOCATION
- LOCATION_ID
- RATE_CODE_ID
- PASSENGER_COUNT
- TRIP_DISTANCE

FACTS

- VENDOR_ID

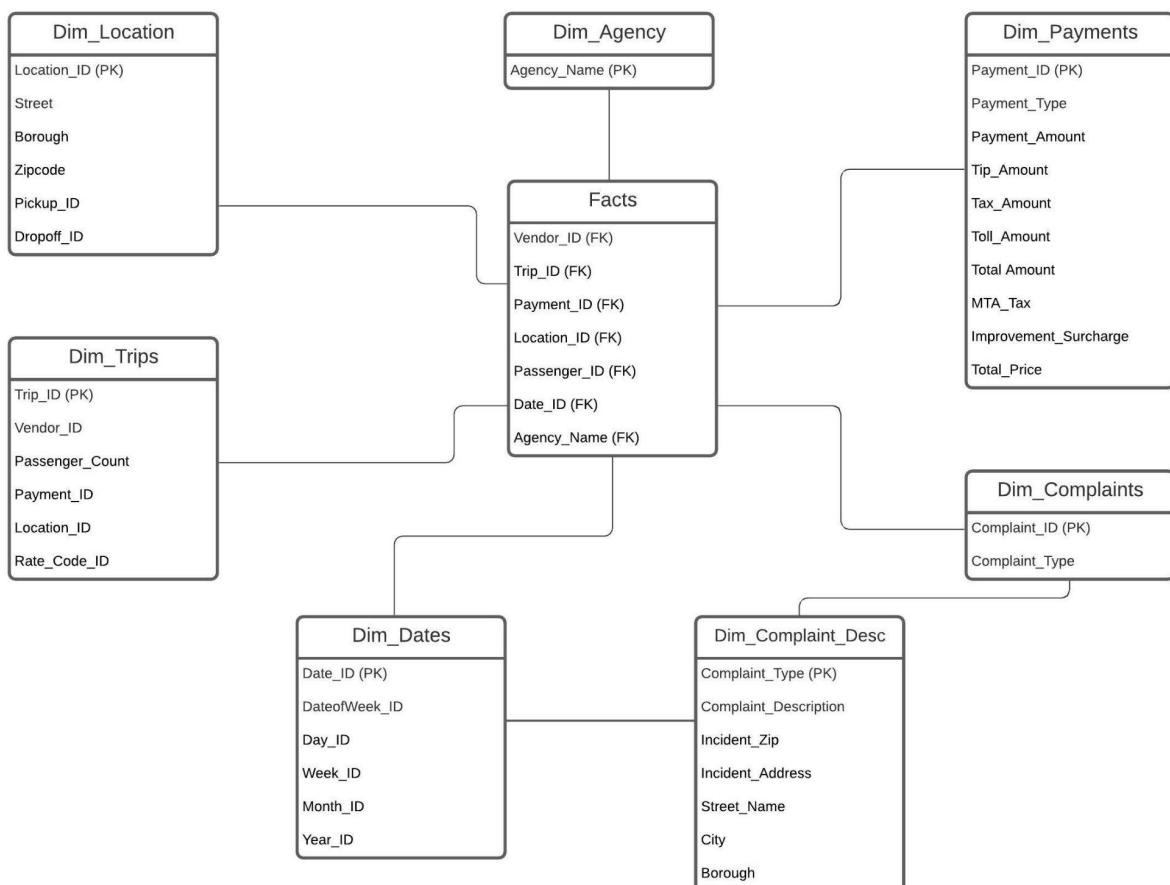
- TRIP_ID
- PAYMENT_ID
- LOCATION_ID
- DATE_ID
- PASSENGER_ID
- AGENCY_NAME

DIMENSIONAL MODEL DRAFT 1



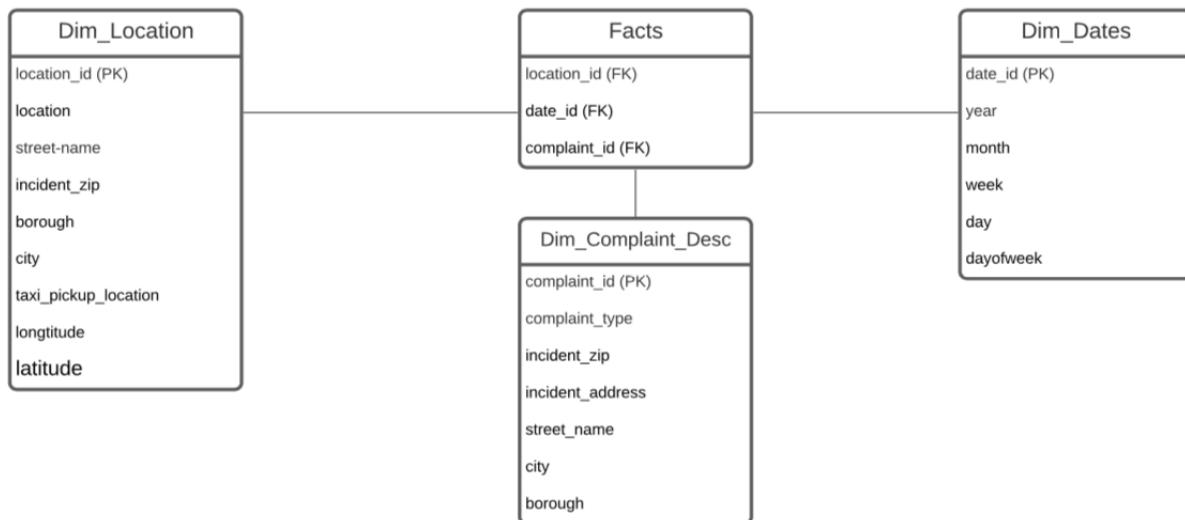
DIMENSIONAL MODEL DRAFT 2

FINAL DIMENSIONAL MODEL

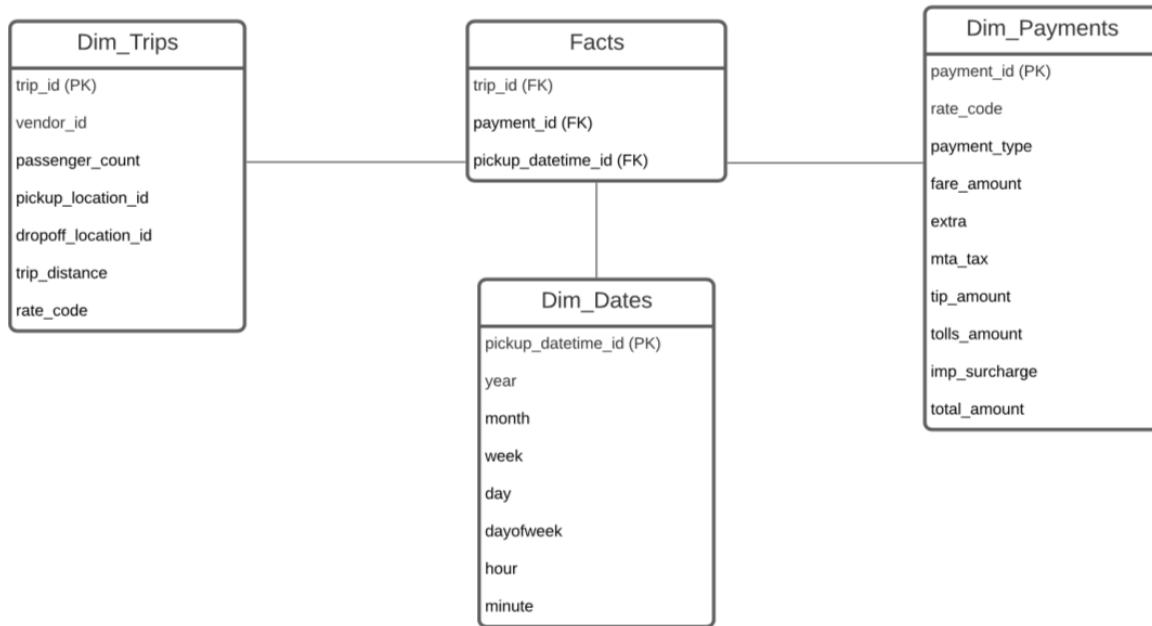


DIMENSIONAL MODEL FINAL

311 SERVICE REQUEST

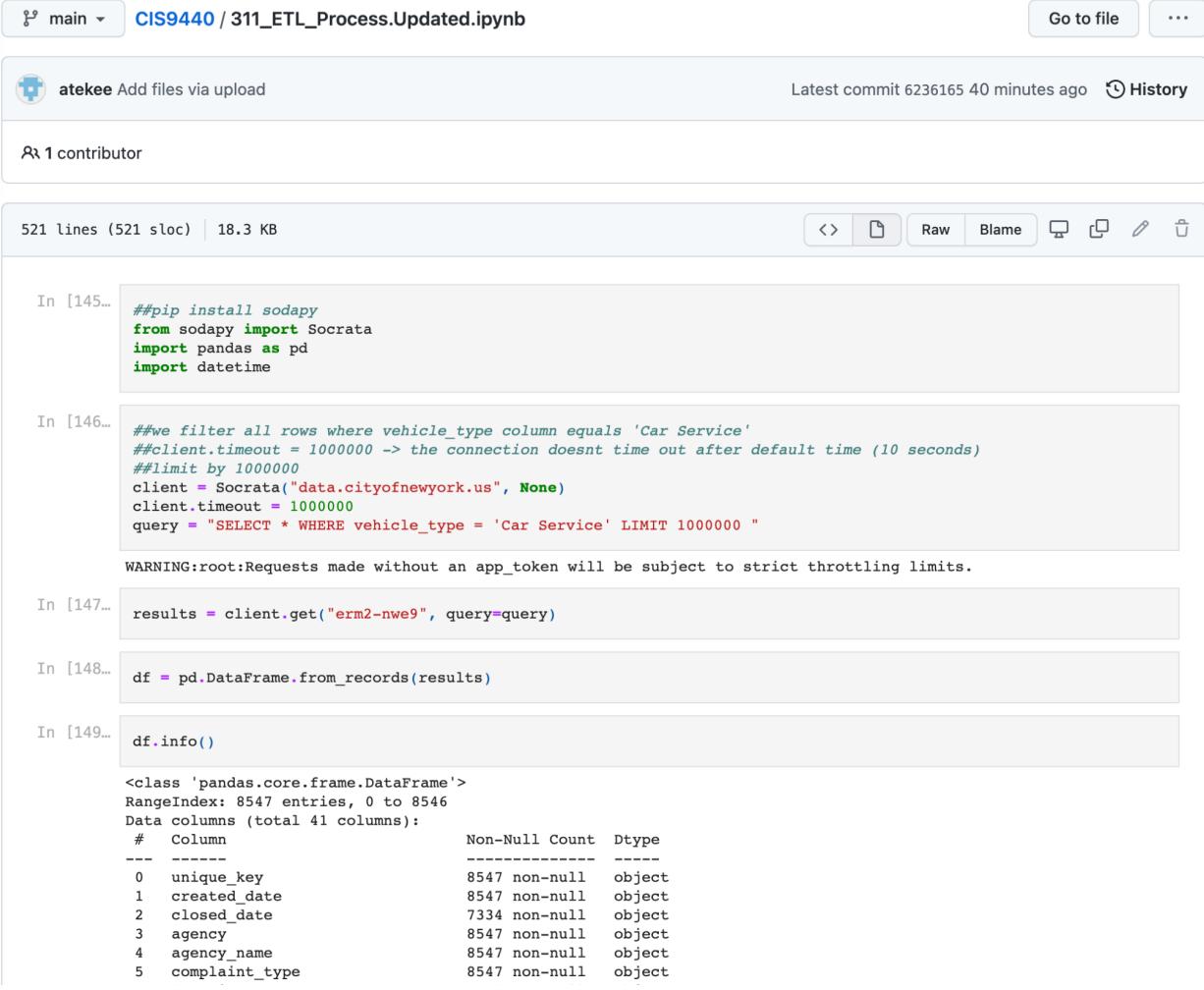


TLC YELLOW TRIPS



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 dropdown 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.

A screenshot of a Jupyter Notebook interface. The title bar shows 'main' and 'CIS9440 / 311_ETL_Process.Updated.ipynb'. The notebook contains several code cells:

- In [145]: Python code to install sodapy and import Socrata, pandas, and datetime.
- In [146]: Python code to filter rows where vehicle_type equals 'Car Service', set client.timeout to 1000000, and execute a query to limit results to 1000000.
- In [147]: Python code to get results from the client using the specified query.
- In [148]: Python code to convert results into a pandas DataFrame.
- In [149]: Python code to display the DataFrame's information, including its structure and non-null counts.

The notebook also shows a commit history with one commit by 'atekee' and a note about 1 contributor.

```

In [145...]:
##pip install sodapy
from sodapy import Socrata
import pandas as pd
import datetime

In [146...]:
##we filter all rows where vehicle_type column equals 'Car Service'
##client.timeout = 1000000 -> the connection doesn't time out after default time (10 seconds)
###limit by 1000000
client = Socrata("data.cityofnewyork.us", None)
client.timeout = 1000000
query = "SELECT * WHERE vehicle_type = 'Car Service' LIMIT 1000000"

WARNING:root:Requests made without an app_token will be subject to strict throttling limits.

In [147...]:
results = client.get("erm2-nwe9", query=query)

In [148...]:
df = pd.DataFrame.from_records(results)

In [149...]:
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8547 entries, 0 to 8546
Data columns (total 41 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   unique_key       8547 non-null    object 
 1   created_date    8547 non-null    object 
 2   closed_date     7334 non-null    object 
 3   agency          8547 non-null    object 
 4   agency_name     8547 non-null    object 
 5   complaint_type  8547 non-null    object 

```

```

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, 10005 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.

Name	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention
311_Service_Request_TLC/	Folder	—	—	—	—	—	—	—
yellow_tripdata_2019/	Folder	—	—	—	—	—	—	—
yellow_tripdata_2020/	Folder	—	—	—	—	—	—	—

The screenshot shows the Google Cloud Platform BigQuery interface. The top navigation bar includes 'Google Cloud Platform', 'CIS 9440 Group Project', a search bar, and various navigation icons. Below the navigation is a sidebar titled 'Explorer' with a 'Type to search' input field. The main content area displays the '311_service_request' table details. The 'DETAILS' tab is selected, showing the following information:

Table ID	cis-9440-group-project-332602:cis9440_311.311_service_request
Table size	418.93 KB
Long-term storage size	0 B
Number of rows	1,171
Created	Nov 19, 2021, 8:21:45 PM UTC-5
Last modified	Nov 19, 2021, 8:21:45 PM UTC-5
Table expiration	NEVER
Data location	US
Description	(empty)

Below the table details, there are buttons for 'COMPOSE NEW QUERY', 'SHARE', 'COPY', 'SNAPSHOT', 'DELETE', and 'EXPORT'.

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.

The screenshot shows the Google Cloud Platform BigQuery interface. The top navigation bar includes 'Google Cloud Platform', 'CIS 9440 Group Project', a search bar, and various navigation icons. Below the navigation is a sidebar titled 'Explorer' with a 'Type to search' input field. The main content area displays the 'tlc_yellow_trips' table details. The 'DETAILS' tab is selected, showing the following information:

Table ID	cis-9440-group-project-332602:cis9440_yellowcab_project.tlc_yellow_trips
Table size	40.6 GB
Long-term storage size	0 B
Number of rows	337,785,845
Created	Nov 20, 2021, 1:42:11 PM UTC-5
Last modified	Nov 20, 2021, 3:18:35 PM UTC-5
Table expiration	NEVER
Data location	US
Description	(empty)

Below the table details, there are buttons for 'COMPOSE NEW QUERY', 'SHARE', 'COPY', 'SNAPSHOT', 'DELETE', and 'EXPORT'.

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

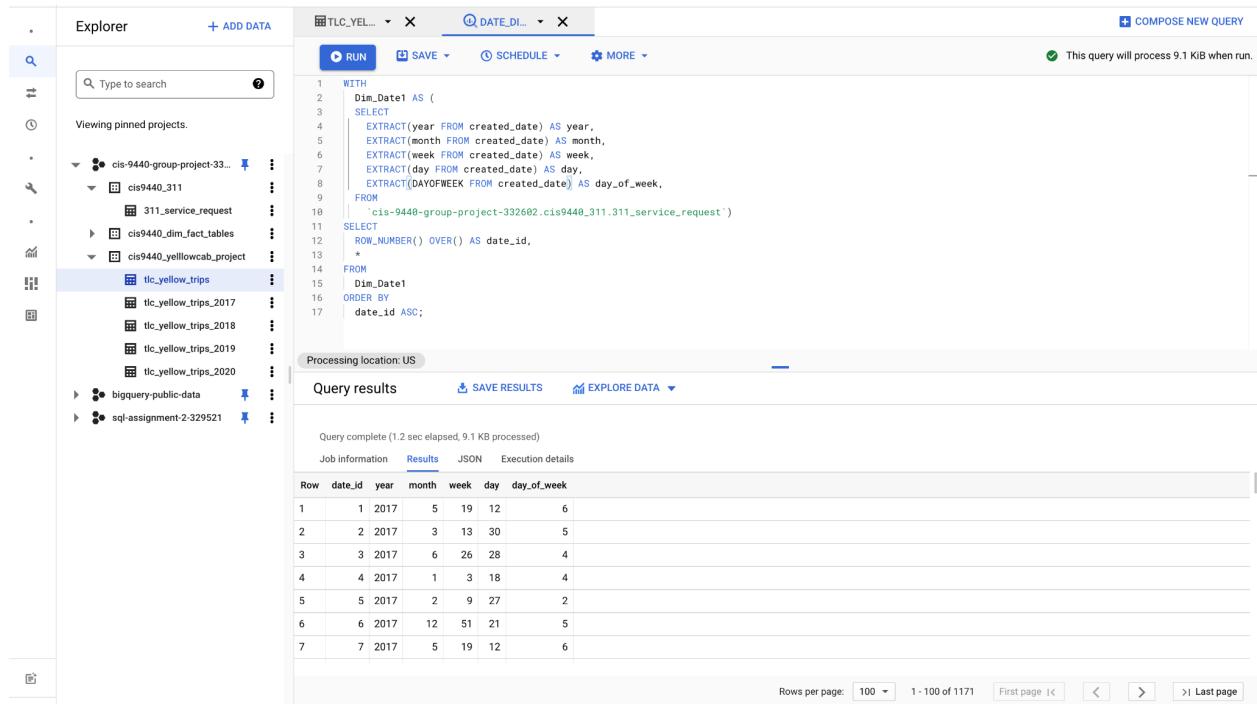
```
Pickup_longitude;
Pickup_latitude;
Dropoff_longitude;
Dropoff_latitude;
Congestion_surcharge;
store_and_fwd_flag;
```

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.



The screenshot shows the BigQuery interface with the following details:

- Explorer:** Shows pinned projects including cis-9440-group-project-332602, cis9440_311, and cis9440_yellowcab_project.
- Query Editor:** A query titled "TLC_YEL..." is running. The code uses a WITH clause to define a Dim_Date1 alias, then performs a SELECT on the 311_service_request table, ordering by a generated date_id column. The code is as follows:

```

1 WITH
2   Dim_Date1 AS (
3     SELECT
4       EXTRACT(year FROM created_date) AS year,
5       EXTRACT(month FROM created_date) AS month,
6       EXTRACT(week FROM created_date) AS week,
7       EXTRACT(day FROM created_date) AS day,
8       EXTRACT(DAYOFWEEK FROM created_date) AS day_of_week,
9     FROM
10    `cis-9440-group-project-332602.cis9440_311.311_service_request` )
11   SELECT
12     ROW_NUMBER() OVER() AS date_id,
13     *
14   FROM
15   Dim_Date1
16   ORDER BY
17     date_id ASC;

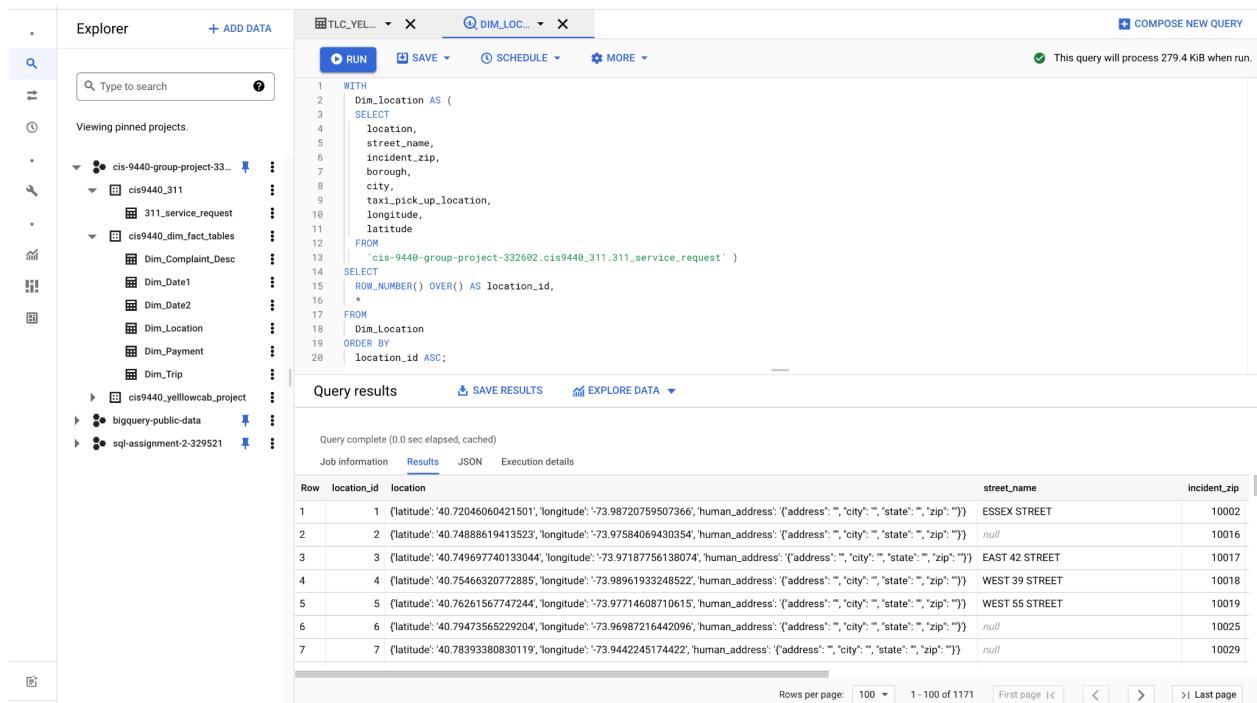
```

- Processing location:** US
- Query results:** The results show a table with columns: Row, date_id, year, month, week, day, and day_of_week. The data for the first 7 rows is as follows:

Row	date_id	year	month	week	day	day_of_week
1	1	2017	5	19	12	6
2	2	2017	3	13	30	5
3	3	2017	6	26	28	4
4	4	2017	1	3	18	4
5	5	2017	2	9	27	2
6	6	2017	12	51	21	5
7	7	2017	5	19	12	6

- Job information:** Query complete (1.2 sec elapsed, 9.1 KB processed).
- Results tab:** Active tab.
- Execution details:** Not visible in the screenshot.
- Page controls:** Rows per page: 100, 1 - 100 of 1171, First page, Last page.

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



The screenshot shows the BigQuery interface with the following details:

- Explorer:** Shows pinned projects including cis-9440-group-project-332602, cis9440_311, and cis9440_yellowcab_project.
- Query Editor:** A query titled "TLC_YEL..." is running. The code uses a WITH clause to define a Dim_Location alias, then performs a SELECT on the 311_service_request table, ordering by a generated location_id column. The code is as follows:

```

1 WITH
2   Dim_Location AS (
3     SELECT
4       location,
5       street_name,
6       incident_zip,
7       borough,
8       city,
9       taxi_pick_up_location,
10      longitude,
11      latitude
12    FROM
13    `cis-9440-group-project-332602.cis9440_311.311_service_request` )
14   SELECT
15     ROW_NUMBER() OVER() AS location_id,
16     *
17   FROM
18   Dim_Location
19   ORDER BY
20     location_id ASC;

```

- Processing location:** US
- Query results:** The results show a table with columns: Row, location_id, location, street_name, and incident_zip. The data for the first 7 rows is as follows:

Row	location_id	location	street_name	incident_zip
1	1	{'latitude': '40.72046060421501', 'longitude': '-73.98720759507366', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	ESSEX STREET	10002
2	2	{'latitude': '40.7488619413523', 'longitude': '-73.97584069430354', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	null	10016
3	3	{'latitude': '40.749697740133044', 'longitude': '-73.97187756138074', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	EAST 42 STREET	10017
4	4	{'latitude': '40.75466320772885', 'longitude': '-73.98961933248522', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	WEST 39 STREET	10018
5	5	{'latitude': '40.76261567747244', 'longitude': '-73.97714608710615', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	WEST 55 STREET	10019
6	6	{'latitude': '40.79473565229204', 'longitude': '-73.96987216442096', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	null	10025
7	7	{'latitude': '40.7839338030119', 'longitude': '-73.9442245174422', 'human_address': {'address': '', 'city': '', 'state': '', 'zip': ''}}	null	10029

- Job information:** Query complete (0.0 sec elapsed, cached).
- Results tab:** Active tab.
- Execution details:** Not visible in the screenshot.
- Page controls:** Rows per page: 100, 1 - 100 of 1171, First page, Last page.

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

The screenshot shows the BigQuery interface with two tabs open: 'TLC_YEL...' and 'DIM_CO...'. The code in the editor is:

```

1 WITH
2   Dim_Complaint_Desc AS (
3     SELECT
4       complaint_type,
5       incident_incident_zip,
6       incident_address,
7       street_name,
8       city,
9       borough
10      FROM
11        `cis-9440-group-project-332602.cis9440_311.311_service_request` )
12
13   SELECT
14     ROW_NUMBER() OVER() AS complaint_id,
15     *
16   FROM
17     Dim_Complaint_Desc
18   ORDER BY
19     complaint_id ASC;

```

The results table shows 7 rows of data:

Row	complaint_id	complaint_type	incident_zip	incident_address	street_name	city	bor
1	1	For Hire Vehicle Complaint	Car Service Company Complaint	146 ESSEX STREET	ESSEX STREET	NEW YORK	MA
2	2	For Hire Vehicle Complaint	Driver Complaint	null	null	NEW YORK	MA
3	3	For Hire Vehicle Report	Driver Report	303 EAST 42 STREET	EAST 42 STREET	NEW YORK	MA
4	4	For Hire Vehicle Complaint	Car Service Company Complaint	242 WEST 39 STREET	WEST 39 STREET	NEW YORK	MA
5	5	For Hire Vehicle Complaint	Driver Complaint	77 WEST 55 STREET	WEST 55 STREET	NEW YORK	MA
6	6	For Hire Vehicle Complaint	Driver Complaint	null	null	NEW YORK	MA
7	7	For Hire Vehicle Report	Driver Report	null	null	NEW YORK	MA

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

The screenshot shows the BigQuery interface with two tabs open: 'DATE_DL...' and 'DIM_DAT...'. The code in the editor is:

```

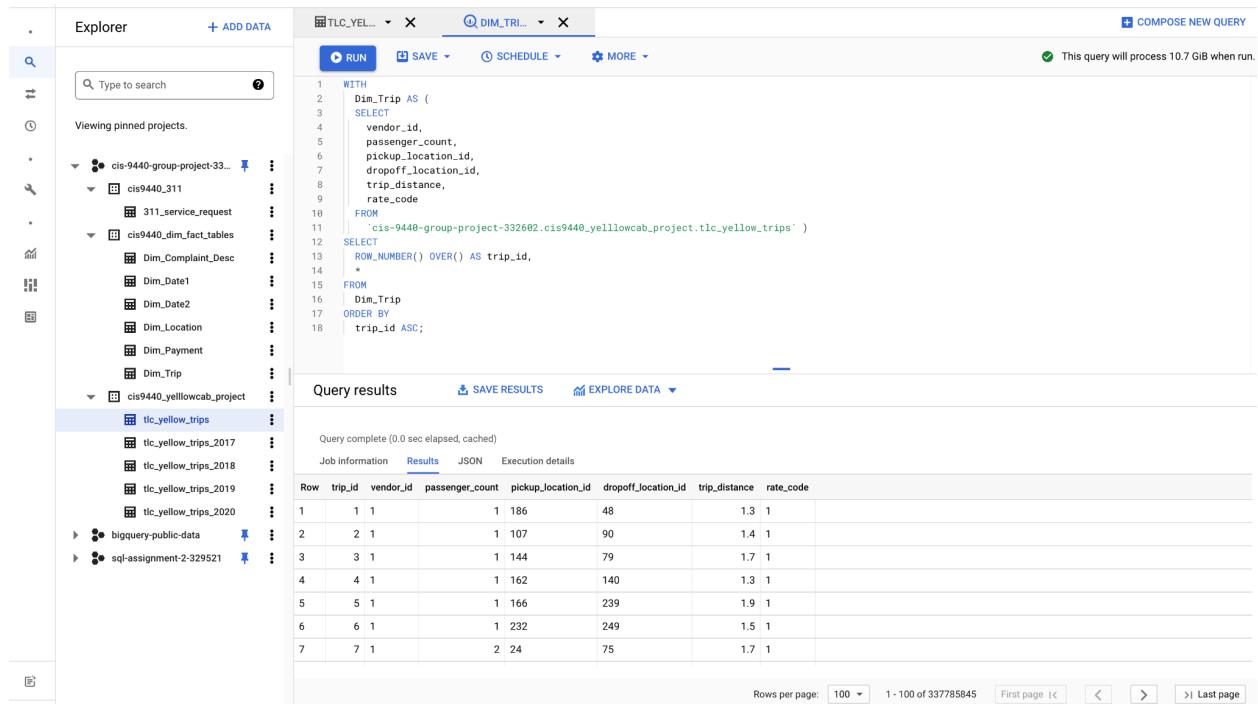
1 WITH
2   Dim_Date2 AS (
3     SELECT
4       EXTRACT(year FROM pickup_datetime) AS year,
5       EXTRACT(month FROM pickup_datetime) AS month,
6       EXTRACT(week FROM pickup_datetime) AS week,
7       EXTRACT(day FROM pickup_datetime) AS day,
8       EXTRACT(DAYOFWEEK FROM pickup_datetime) AS day_of_week,
9       EXTRACT(HOUR FROM pickup_datetime) AS hr,
10      EXTRACT(MINUTE FROM pickup_datetime) AS mn
11    FROM
12      `cis-9440-group-project-332602.cis9440_yellowcab_project.tlc_yellow_trips` )
13
14   SELECT
15     ROW_NUMBER() OVER() AS date_id,
16     +
17   FROM
18     Dim_Date2
19   ORDER BY
20     date_id ASC;

```

The results table shows 7 rows of data:

Row	date_id	year	month	week	day	day_of_week	hr	mn	
1	1	2019	7	27	13		7	20	33
2	2	2019	7	27	11		5	1	49
3	3	2018	10	42	26		6	19	10
4	4	2018	10	43	29		2	17	16
5	5	2019	2	6	11		2	19	27
6	6	2018	4	15	21		7	21	31
7	7	2018	4	16	23		2	12	17

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



The screenshot displays the BigQuery interface for a query named 'Dim_Trip'. The code is as follows:

```

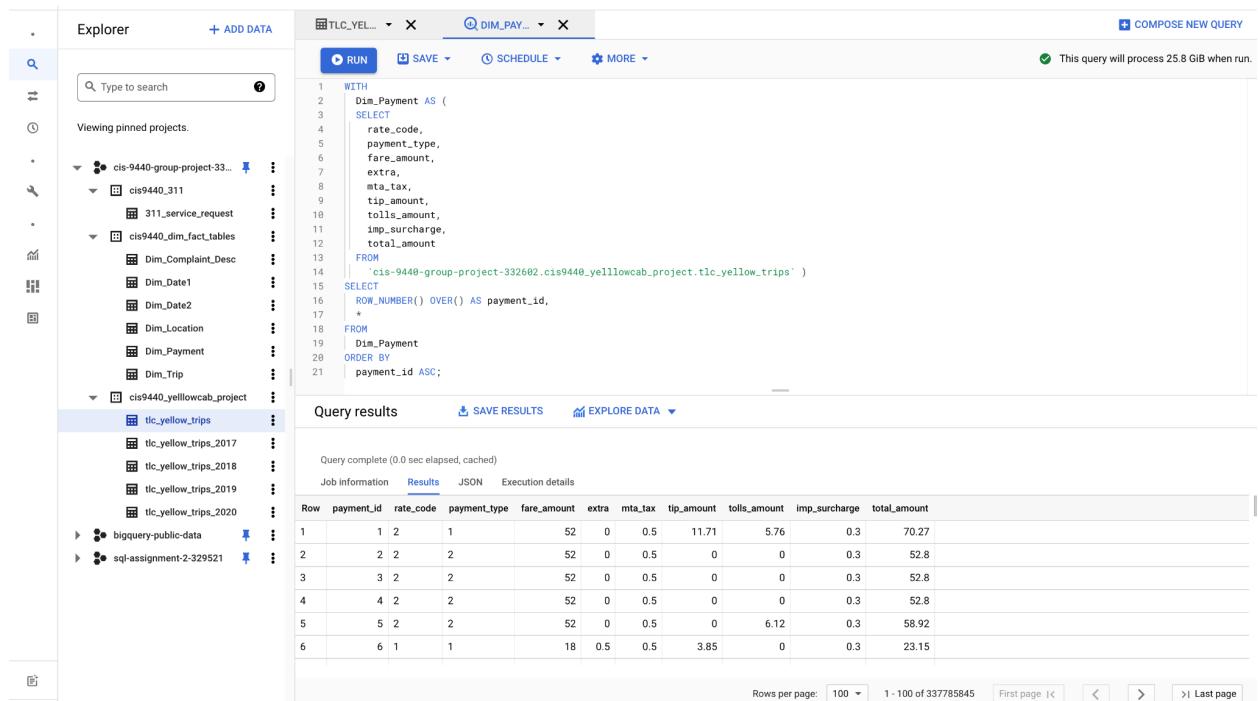
1 WITH
2   Dim_Trip AS (
3     SELECT
4       vendor_id,
5       passenger_count,
6       pickup_location_id,
7       dropoff_location_id,
8       trip_distance,
9       rate_code
10    FROM
11      `cis-9440-group-project-332602.cis9440_yellowcab_project.tlc_yellow_trips`
12
13      ROW_NUMBER() OVER() AS trip_id,
14
15    *
16
17   FROM
18     Dim_Trip
19   ORDER BY
20     trip_id ASC;

```

The 'Query results' section shows the first 100 rows of data:

Row	trip_id	vendor_id	passenger_count	pickup_location_id	dropoff_location_id	trip_distance	rate_code
1	1	1	1	186	48	1.3	1
2	2	1	1	107	90	1.4	1
3	3	1	1	144	79	1.7	1
4	4	1	1	162	140	1.3	1
5	5	1	1	166	239	1.9	1
6	6	1	1	232	249	1.5	1
7	7	1	2	24	75	1.7	1

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



The screenshot displays the BigQuery interface for a query named 'Dim_Payment'. The code is as follows:

```

1 WITH
2   Dim_Payment AS (
3     SELECT
4       rate_code,
5       payment_type,
6       fare_amount,
7       extra,
8       mta_tax,
9       tip_amount,
10      tolls_amount,
11      imp_surcharge,
12      total_amount
13    FROM
14      `cis-9440-group-project-332602.cis9440_yellowcab_project.tlc_yellow_trips`
15
16      ROW_NUMBER() OVER() AS payment_id,
17
18    *
19
20   FROM
21     Dim_Payment
22   ORDER BY
23     payment_id ASC;

```

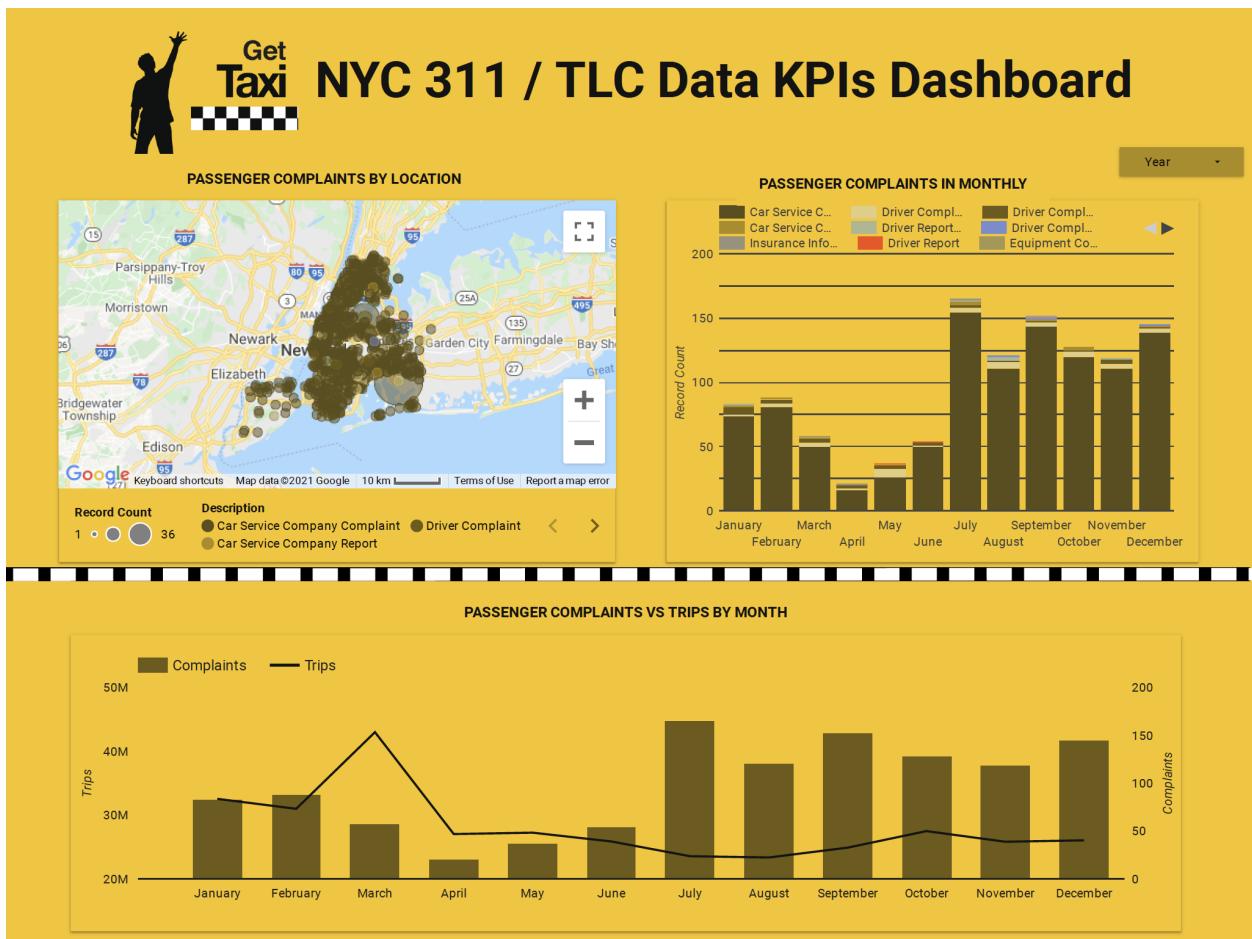
The 'Query results' section shows the first 100 rows of data:

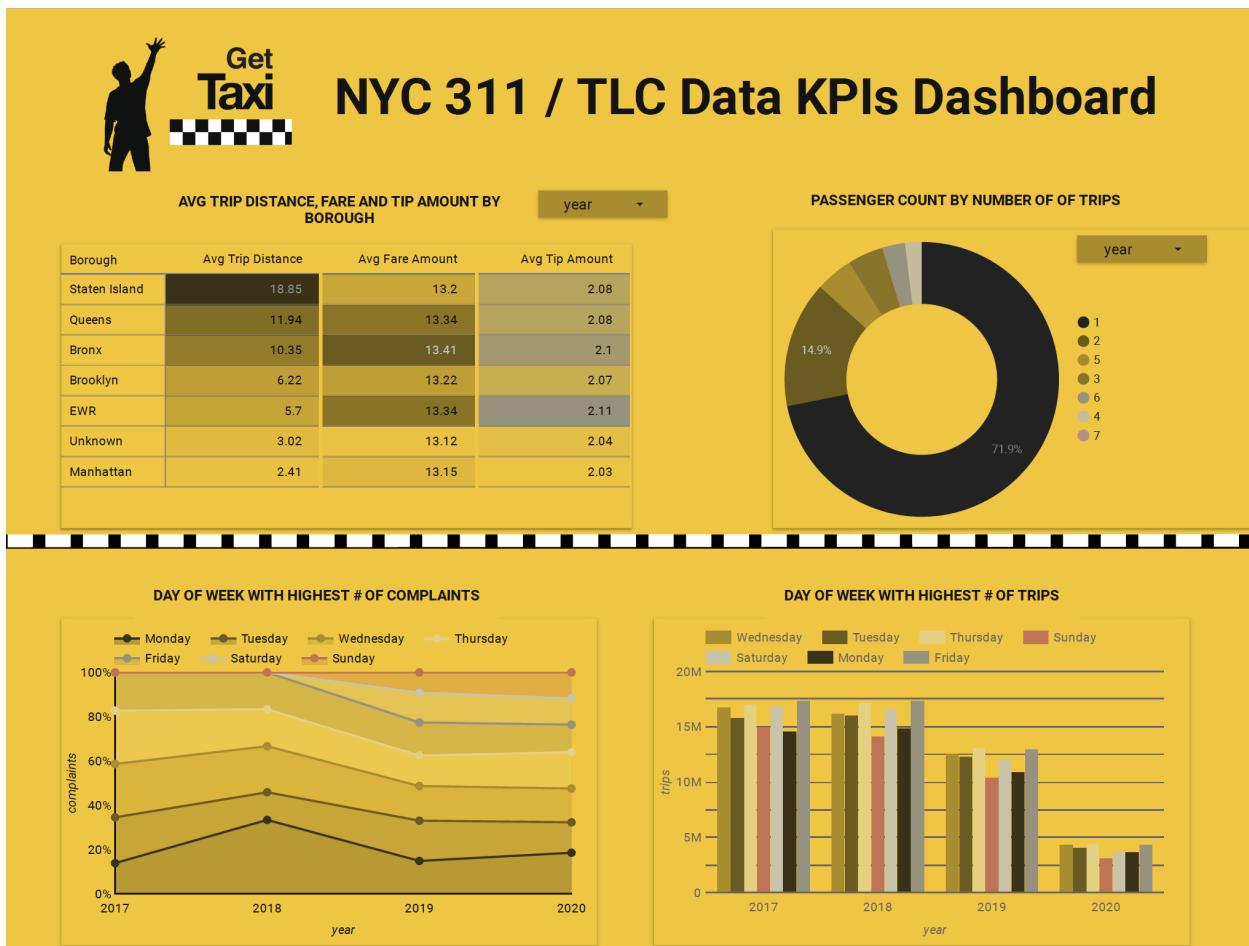
Row	payment_id	rate_code	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	imp_surcharge	total_amount
1	1	2	1	52	0	0.5	11.71	5.76	0.3	70.27
2	2	2	2	52	0	0.5	0	0	0.3	52.8
3	3	2	2	52	0	0.5	0	0	0.3	52.8
4	4	2	2	52	0	0.5	0	0	0.3	52.8
5	5	2	2	52	0	0.5	0	6.12	0.3	58.92
6	6	1	1	18	0.5	0.5	3.85	0	0.3	23.15

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)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> Expanded on Introduction Revised KPI's and description of problem Formatted deliverable
Thursday,	8:30PM -10:00PM	All members	<ul style="list-style-type: none"> Revised first deliverabl



October 7th		(Atabay, Carla, Fengping, Jason, Kwadwo)	<ul style="list-style-type: none"> Draft of dimensional model
Friday, November 5th	9:00PM - 10:00PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	<ul style="list-style-type: none"> Revised dimensional model Incorporated into final draft of proposal
Thursday, November 18th	9:30PM - 11:30PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> Created Dashboards for each KPIs
Friday, December 3rd	8:30PM - 10:30PM	All members (Atabay, Carla, Fengping, Jason, Kwadwo)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> Completed the presentation slides Prepared for presentation individually