New York City Street Conditions
Data Warehouse/Analytics

I. Business Scenario

A narrative description of the issue or problem the group hopes to address. Include in this description your group's choice of the type of 311 complaints (can be a combination of types) and a second data set to integrate with the 311 data.

New York City, one of the greatest and richest cities in the world, faces one huge problem; its crippling infrastructure. Millions of people take the subway, drive, bike, and walk on the streets of New York City each day. There are constant subway delays, potholes, trash, and flooding. It is unacceptable for New Yorkers to have to deal with such bad conditions and a poor quality of life when they pay one of the highest tax rates in the country!

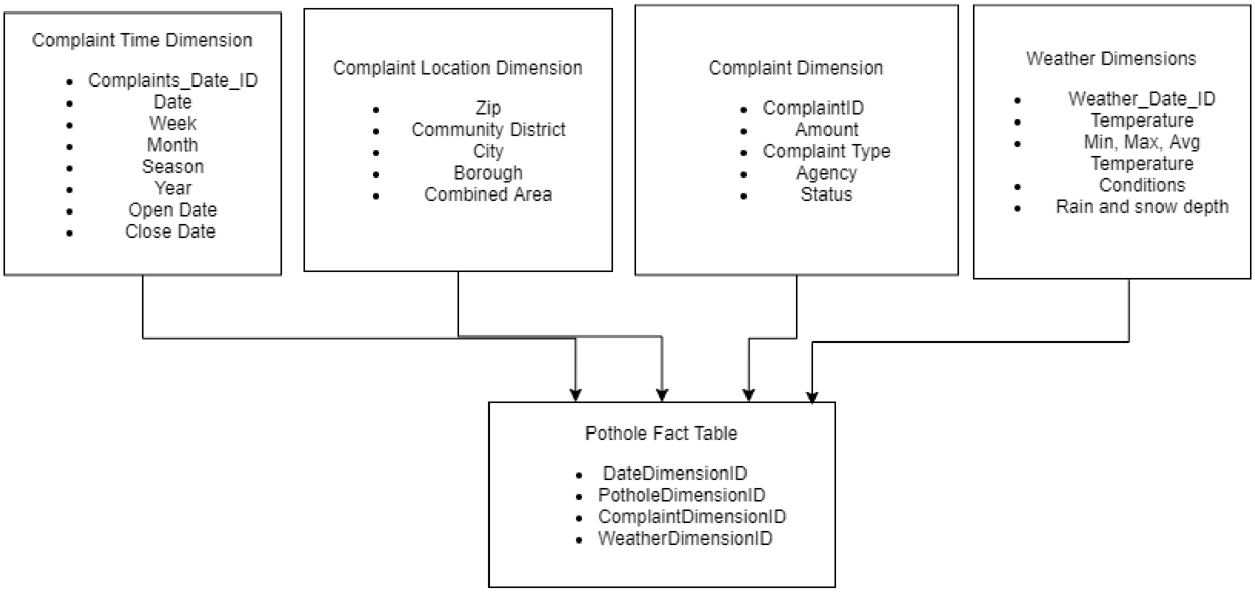
Taking a look at 311 complaints, we can see that a majority of the complaints are about street conditions, unsanitary conditions, and plumbing issues- all which could be considered an infrastructure issue. One particular type of problem that NYC faces that we hope to address is its **street conditions**, racking up a total of 93,270 complaints to 311 in just the year 2017 alone. As citizens who ride on the streets of New York daily and are frustrated with the conditions, we as data analysts were determined to figure out why the street or road conditions were so bad. We plan to use Street Pothole Work Orders and/or a weather dataset in combination with the street conditions complaints from 311 to dig deeper into the issue.

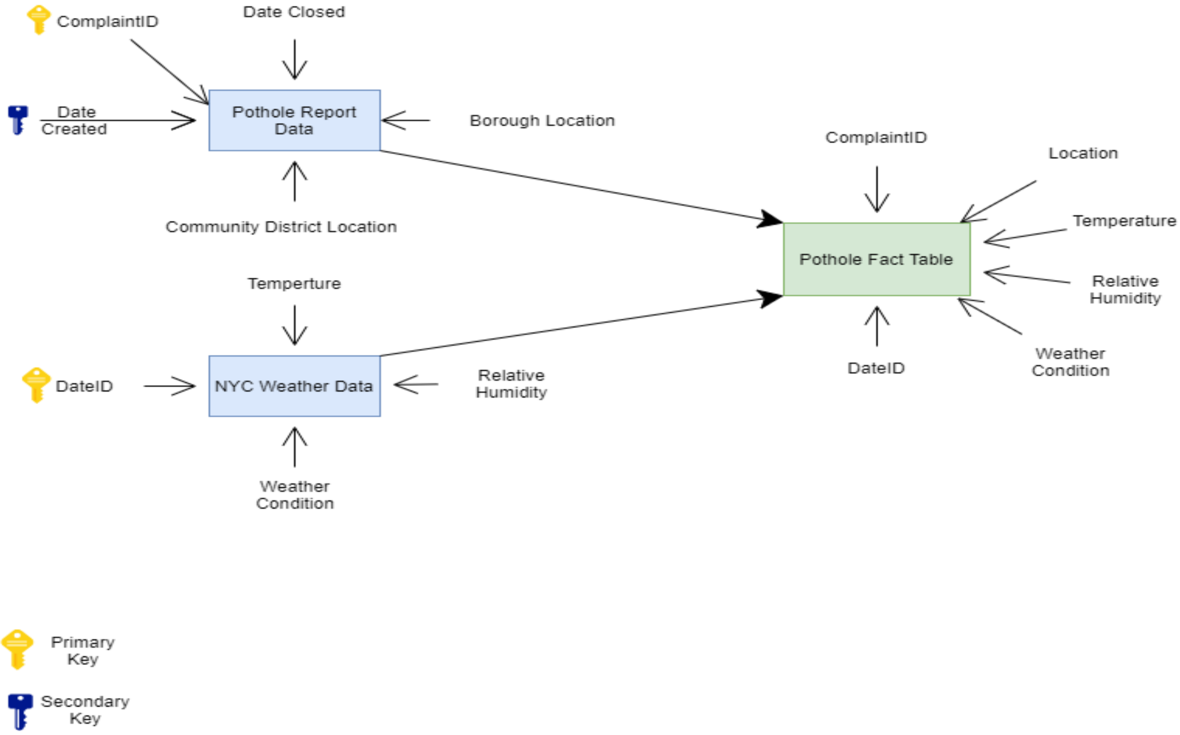
Key Performance Indicators:

Number of Complaints after Inclement Weather
Number of Potholes per Complaint
Number of Complaints per Road
Number of Complaints per Month
Graffiti to Street Condition Complaints
City Budget to Street Condition Complaints
Congestion/Traffic to Street Condition Complaints
Road work to Street Condition Complaints
Additional Time Traveled to Street Condition Complaints
Street Conditions to Car Accidents
Number of Complaints per Zip Code

II. Draft of Dimensional Model

Groups should confirm their data sources are available by downloading sample data. Then formalize the KPIs. Draft dimensional model; design the data warehouse schema by defining dimensions and fact tables.





GRAINS: DATE, TEMPERATURE, COMPLAINT

We chose potholes that have been fixed, as it was the most accurate way to know if the city observed the existence of the pothole and that it was not a duplicate report.

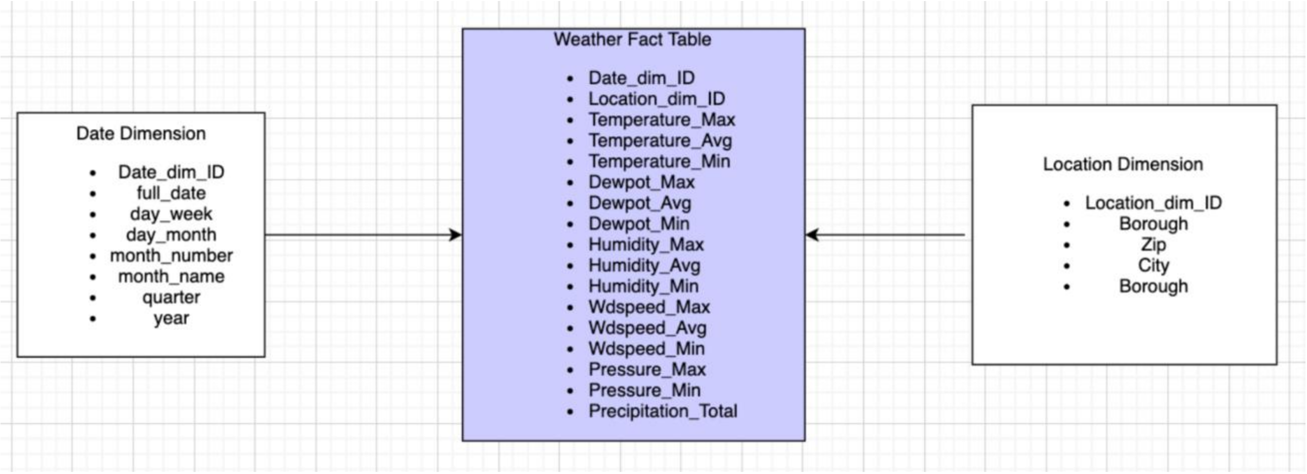
We have decided that we are using the transaction grain; it is the most applicable based on the broad context of observing longer periods of time such as weekends/weekdays and seasons such as fall/spring.

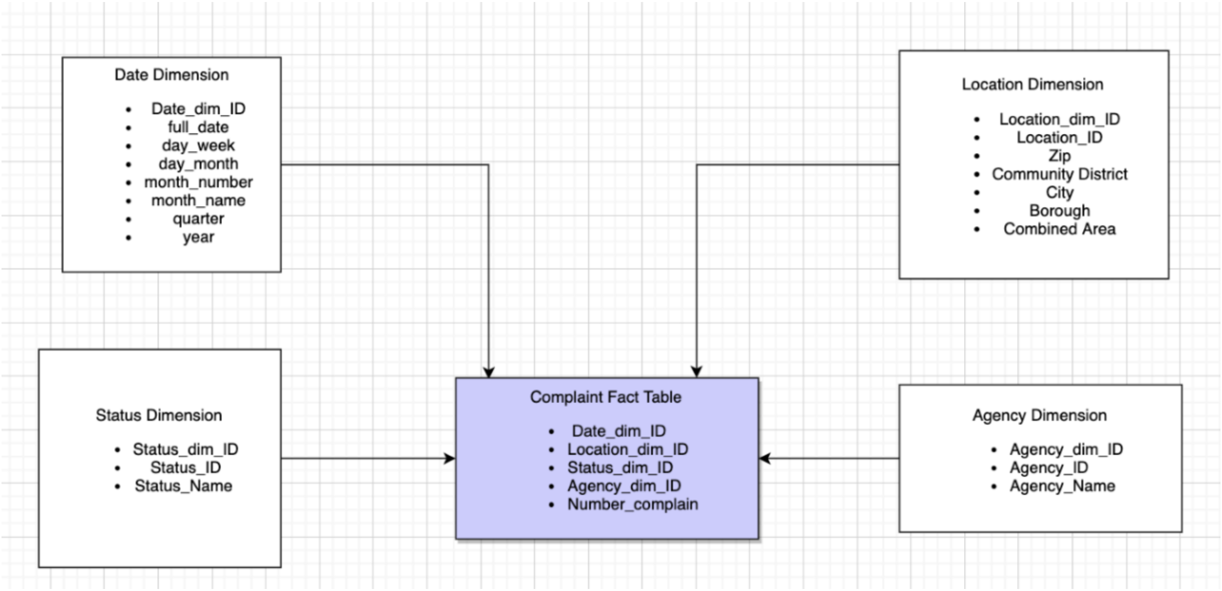
FINALIZED LIST OF KPIs

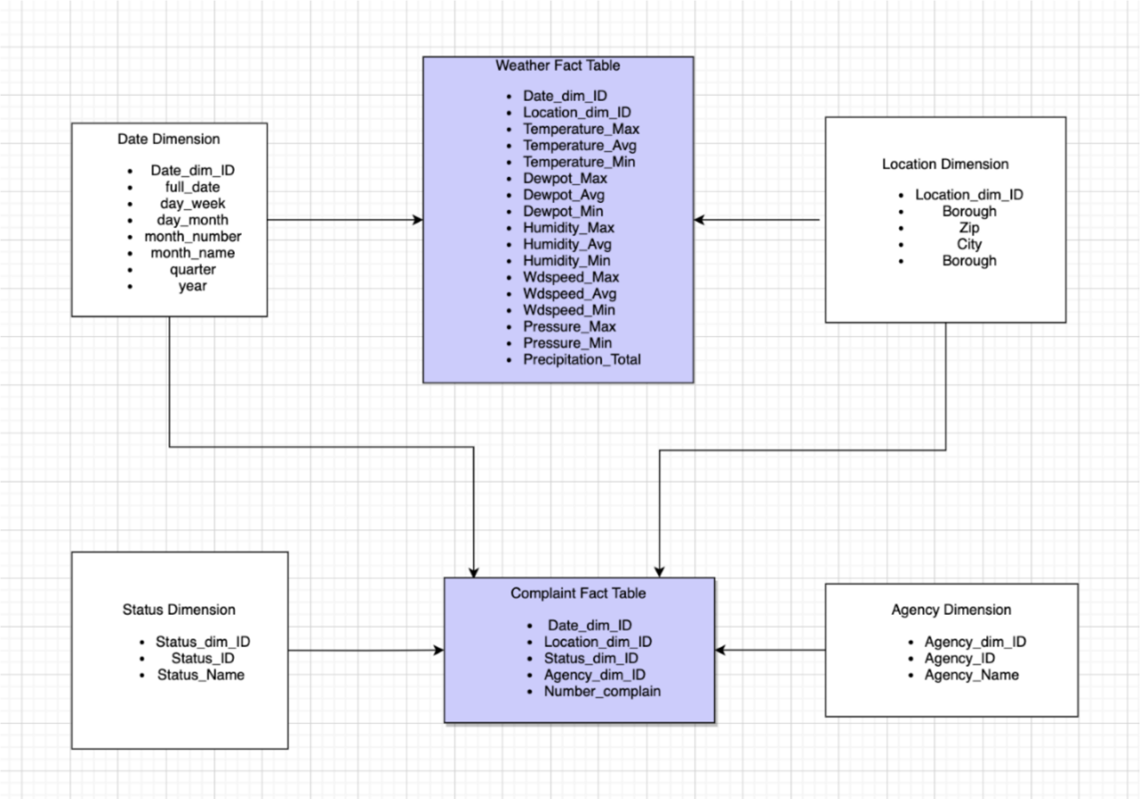
- Pothole occurences per district/borough after inclement weather 10/20/30 days after
- Humidity effects on pothole complaints
- Seasonal complaints per District
- Pothole occurence rate per district
- Number of Complaints per Month
- Time until street condition request closed

III. Dimensional Model Finalized

(October 20th)







Weather Fact Table

- Date_dim_ID
- Location_dim_ID
- Temperature_Max
- Temperature_Avg
- Temperature_Min
- Dewpot_Max
- Dewpot_Avg
- Dewpot_Min
- Humidity_Max
- Humidity_Avg
- Humidity_Min
- Wdspeed_Max
- Wdspeed_Avg
- Wdspeed_Min
- Pressure_Max
- Pressure_Min
- Precipitation_Total

Date Dimension

- Date_dim_ID
- full_date
- day_week
- day_month
- month_number
- month_name
- quarter
- year

Location Dimension

- Location_dim_ID
- Borough
- Zip
- City
- Borough

Status Dimension

- Status_dim_ID
- Status_ID
- Status_Name

Complaint Fact Table

- Date_dim_ID
- Location_dim_ID
- Status_dim_ID
- Agency_dim_ID
- Number_complain

Agency Dimension

- Agency_dim_ID
- Agency_ID
- Agency_Name

IV. ETL Tools and Target DBMS

For this project our group will utilize dbt for our ETL tool. We want to attempt to use Informatica once dbt is working, and if Informatica works we will utilize that for the final portion. A benefit of this is that there are plenty of resources and guides on Youtube to utilize. Having a redundancy for each part of this project is the smart choice as to avoid roadblocks that may occur by switching. For hosting, Google Cloud seems to be the way to go due to us already having credit there. For the DBMS we will utilize Google big query as we are already familiar with the service.

AWS is a runner up and if time permits, we would like to dip our toes into it and attempt to use that to host our data. Amazon Redshift and perhaps Amazon RDS seems the way to go, as AWS Database services are commonly used, understanding them is a lucrative skill in the job market. These choices will only be considered as an extra to our project if time permits, so we can increase our own personal knowledge.

ETL Tool: dbt

Secondary: Informatica PowerCenter

Hosting Environment: Google Cloud

Secondary: AWS

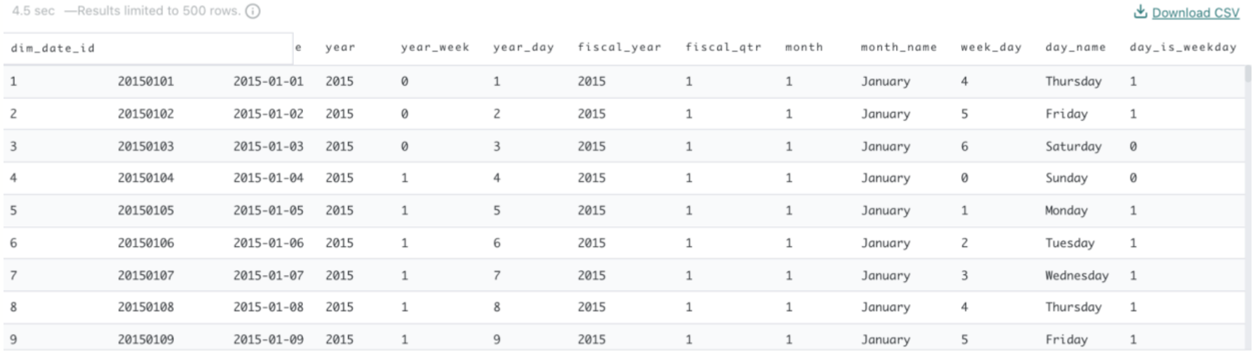
DBMS: Google BigQuery

Secondary: Amazon Redshift

V. ETL Programming

dim_date.sql:

```
with dim_date as (
SELECT
    ROW_NUMBER() OVER() as dim_date_id,
    FORMAT_DATE("%Y%m%d",d) as date_integer,
    d AS full_date,
    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 (SELECT * FROM UNNEST (GENERATE_DATE_ARRAY('2015-01-01', '2023-01-01', INTERVAL 1
DAY) ) AS d ))

select * from dim_date
```



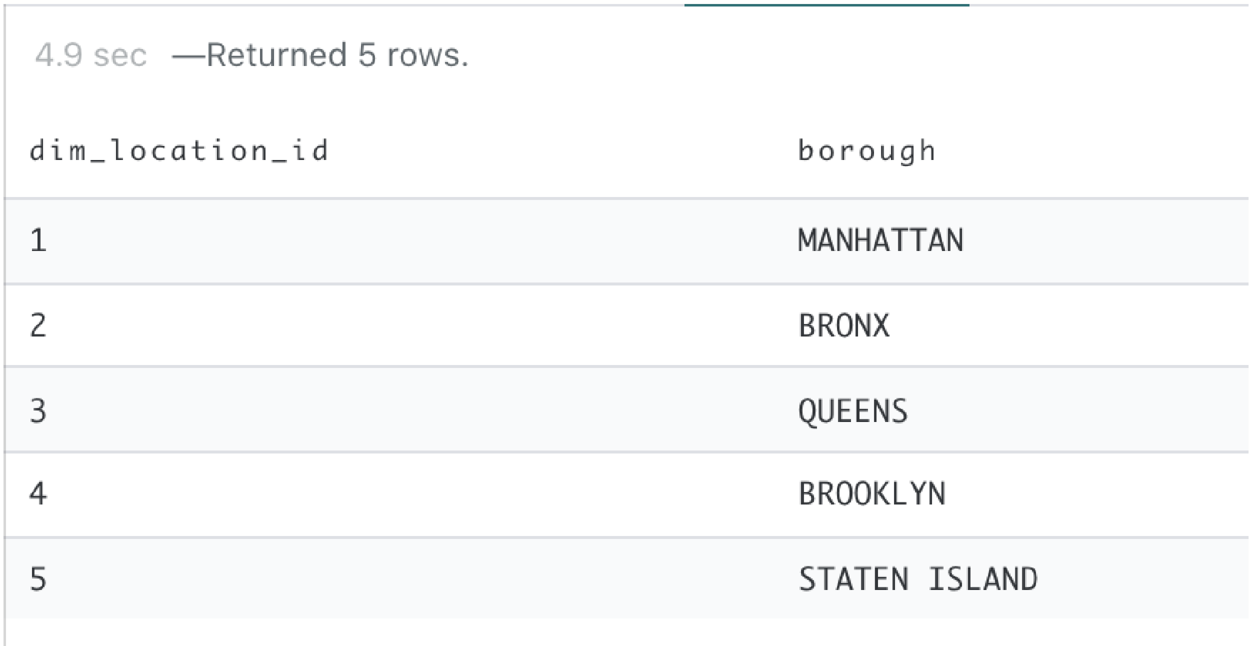
4.5 sec —Results limited to 500 rows.

Download CSV

dim_date_id		e	year	year_week	year_day	fiscal_year	fiscal_qtr	month	month_name	week_day	day_name	day_is_weekday
1	20150101	2015-01-01	2015	0	1	2015	1	1	January	4	Thursday	1
2	20150102	2015-01-02	2015	0	2	2015	1	1	January	5	Friday	1
3	20150103	2015-01-03	2015	0	3	2015	1	1	January	6	Saturday	0
4	20150104	2015-01-04	2015	1	4	2015	1	1	January	0	Sunday	0
5	20150105	2015-01-05	2015	1	5	2015	1	1	January	1	Monday	1
6	20150106	2015-01-06	2015	1	6	2015	1	1	January	2	Tuesday	1
7	20150107	2015-01-07	2015	1	7	2015	1	1	January	3	Wednesday	1
8	20150108	2015-01-08	2015	1	8	2015	1	1	January	4	Thursday	1
9	20150109	2015-01-09	2015	1	9	2015	1	1	January	5	Friday	1

dim_location:

```
{{
  config(
    materialized = "table",
  )
}}

with dim_location as
(SELECT DISTINCT
    upper(Borough) as borough
 FROM `complaint-project-331901.311_Compaints.Weather_Raw`
)
select ROW_NUMBER () OVER() AS dim_location_id, * from dim_location
```

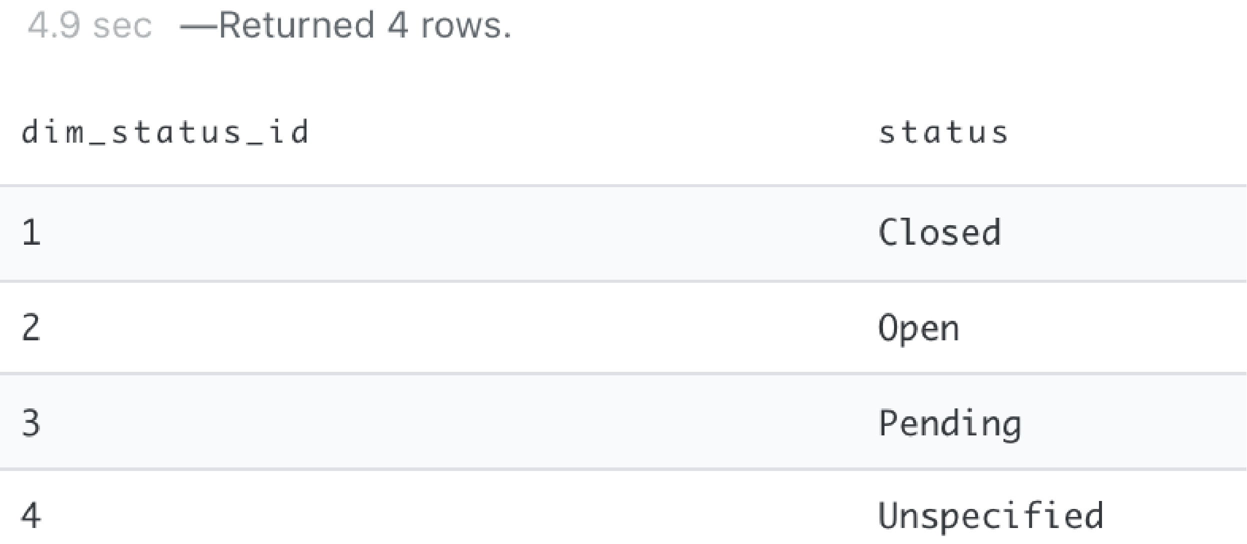


4.9 sec —Returned 5 rows.

dim_location_id	borough
1	MANHATTAN
2	BRONX
3	QUEENS
4	BROOKLYN
5	STATEN ISLAND

dim_Status:

```
{{
  config(
    materialized = "table",
  )
}}
with dim_status as (SELECT
    ROW_NUMBER () OVER() AS dim_status_id,
    Status as status
FROM (select distinct Status from `complaint-project-
331901.311_Compaints.Pothole_Complaints_Raw`) as s)

select * from dim_status
```

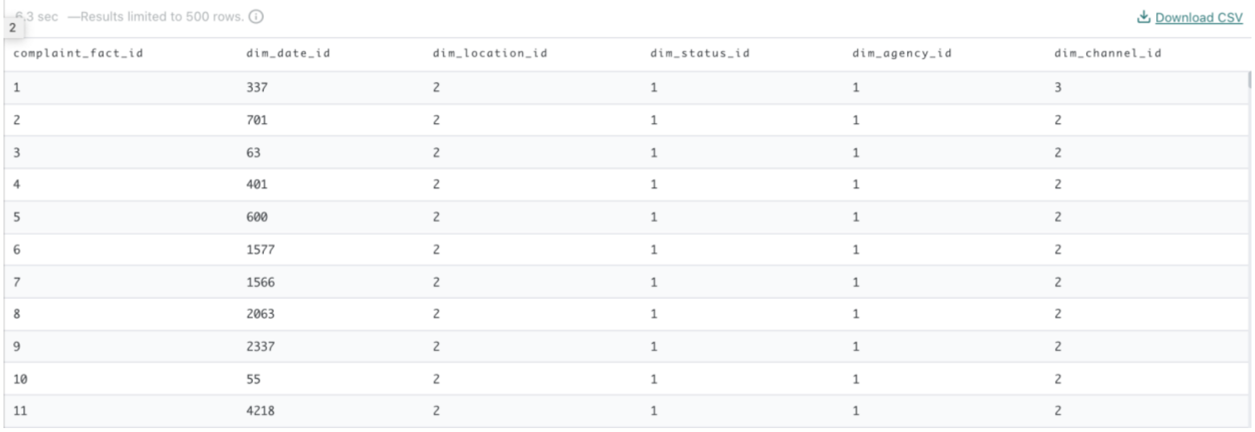


4.9 sec —Returned 4 rows.

dim_status_id	status
1	Closed
2	Open
3	Pending
4	Unspecified

fc_compaints:

```
{{
  config(
    materialized = "table",
  )
}}

select
    ROW_NUMBER () OVER() AS complaint_fact_id,
    dim_date_id,
    dim_location_id,
    dim_status_id,
    dim_agency_id,
    dim_channel_id,
from {{ref ('stg_complaint')}} as complaint
inner join {{ref ('dim_date')}} as complaint_date on complaint.create_date =
complaint_date.full_date
inner join {{ref ('dim_location')}} as complaint_loc on complaint.borough =
complaint_loc.borough
inner join {{ref ('dim_status')}} as complaint_status on complaint.status =
complaint_status.status
inner join {{ref ('dim_agency')}} as complaint_agency on complaint.agency =
complaint_agency.agency
inner join {{ref ('dim_channel')}} as complaint_channel on complaint.channel =
complaint_channel.channel
```

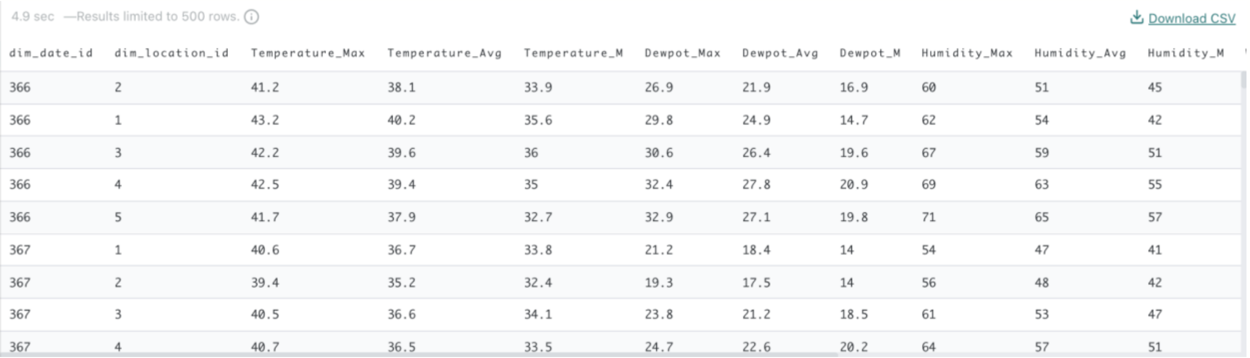


2 6.3 sec —Results limited to 500 rows. ⓘ

complaint_fact_id	dim_date_id	dim_location_id	dim_status_id	dim_agency_id	dim_channel_id
1	337	2	1	1	3
2	701	2	1	1	2
3	63	2	1	1	2
4	401	2	1	1	2
5	600	2	1	1	2
6	1577	2	1	1	2
7	1566	2	1	1	2
8	2063	2	1	1	2
9	2337	2	1	1	2
10	55	2	1	1	2
11	4218	2	1	1	2

fc_weather:

```
{{
  config(
    materialized = "table",
  )
}}

select
    dim_date_id,
    dim_location_id,
    Temperature_Max,
    Temperature_Avg,
    Temperature_M,
    Dewpot_Max,
    Dewpot_Avg,
    Dewpot_M,
    Humidity_Max,
    Humidity_Avg,
    Humidity_M,
    Wdspeed_Max,
    Wdspeed_Avg,
    Wdspeed_M,
    Pressure_Max,
    Pressure_M,
    Precipitation_Total
from `complaint-project-331901.311_Compaints.Weather_Raw` as weather

INNER JOIN {{ref ('dim_date')}} AS date_dimension on weather.WDate =
date_dimension.full_date

INNER JOIN {{ref ('dim_location')}} AS location_dimension on upper(weather.Borough) =
location_dimension.borough
```



4.9 sec —Results limited to 500 rows.

dim_date_id	dim_location_id	Temperature_Max	Temperature_Avg	Temperature_M	Dewpot_Max	Dewpot_Avg	Dewpot_M	Humidity_Max	Humidity_Avg	Humidity_M
366	2	41.2	38.1	33.9	26.9	21.9	16.9	60	51	45
366	1	43.2	40.2	35.6	29.8	24.9	14.7	62	54	42
366	3	42.2	39.6	36	30.6	26.4	19.6	67	59	51
366	4	42.5	39.4	35	32.4	27.8	20.9	69	63	55
366	5	41.7	37.9	32.7	32.9	27.1	19.8	71	65	57
367	1	40.6	36.7	33.8	21.2	18.4	14	54	47	41
367	2	39.4	35.2	32.4	19.3	17.5	14	56	48	42
367	3	40.5	36.6	34.1	23.8	21.2	18.5	61	53	47
367	4	40.7	36.5	33.5	24.7	22.6	20.2	64	57	51

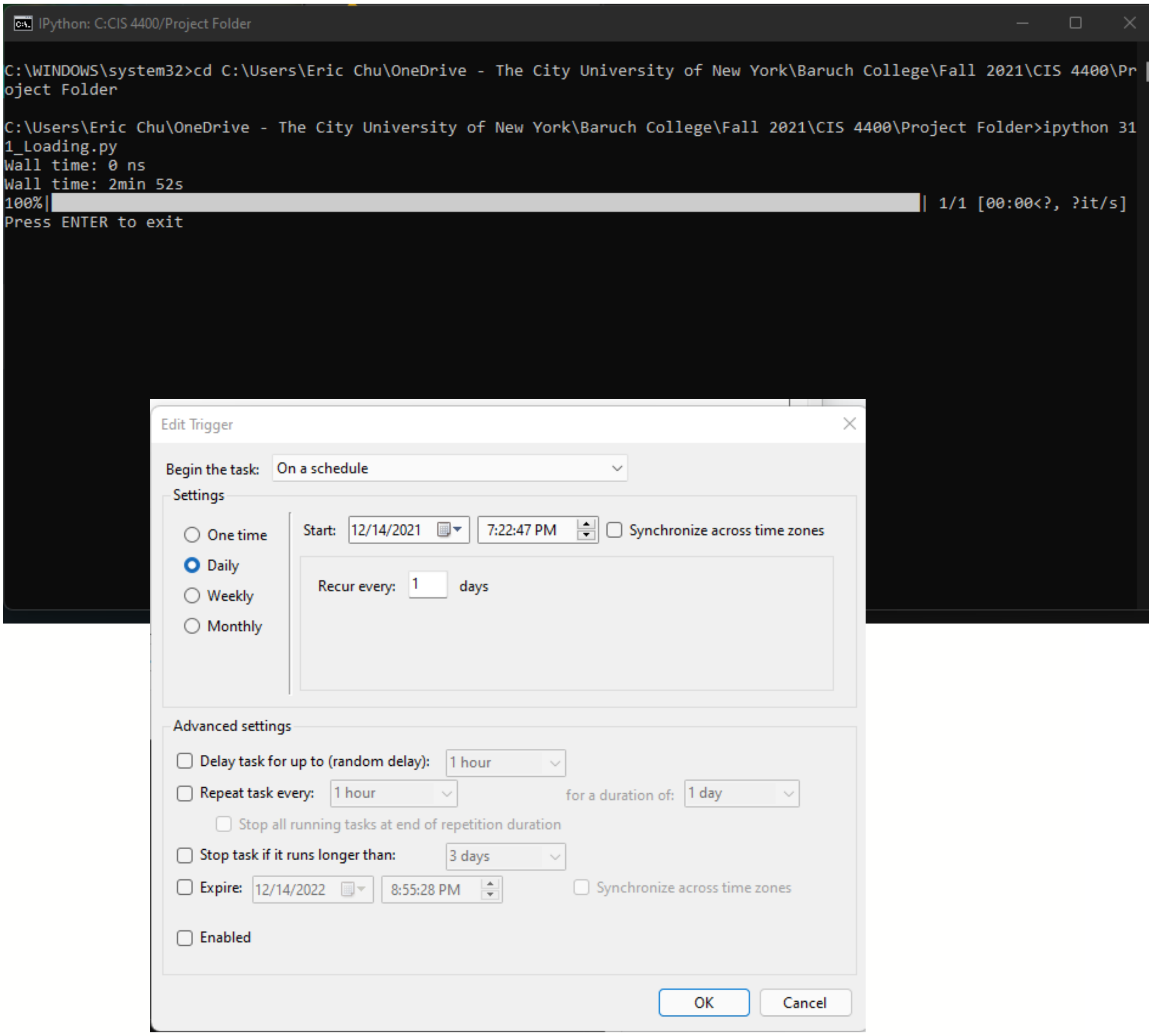
VI. Automating Pulling Data from the API:

We were able to write a script in Python that pulled 311 data from Socrata API every 24 hours using Windows Task Scheduler.

The following script was first written using Jupyter Notebook and then converted into a Python file.

```
import pandas as pd
from datetime import datetime
from sodapy import Socrata
import timeit
from google.cloud import bigquery
from pandas.io import gbq
data_url='data.cityofnewyork.us'
data_set='erm2-nwe9'
app_token='ci7wMrkkG4xvIdindGlDe8QGy'
client = Socrata(data_url,app_token)

record_count = client.get(data_set, where="Descriptor = 'Pothole' ", select="COUNT(*)")
print(record_count)


start = 0
chunk_size = 2000
results =[]
where_clause="Descriptor = 'Pothole' AND date_extract_y(created_date)=2021"

select_col ="created_date, agency, agency_name,\
            complaint_type, open_data_channel_type,\
            status,borough, incident_zip, city"


while True:

    results.extend( client.get(data_set, where=where_clause, offset=start, select=select_col, limit=chunk_size))

    start = start + chunk_size

    if (start > int(record_count[0]['COUNT']) ):
       break

df = pd.DataFrame.from_records(results)

df.to_gbq(destination_table='311_Compaints.Pothole_Complaints',project_id='complaint-project-
331901',if_exists='append')

print('everything is good.')
```

There was a problem with directly running the script from Windows Task Scheduler because the script needed iPython to run properly but my Python installation was messed up. I ended up having to create a workaround which was creating a new text document, adding commands “cd {file directory}” & “iPython {file_name}” , saving the file as a “.bat” file, and finally adding the “.bat” file that I made into Windows Task Scheduler. From then on, the file is being run properly and I added a “Trigger” that runs the file every 24 hours. Everytime the file was run, it would pull the 311 data from the Socrata API and then update our Google BigQuery Database with the latest data.



IPython: C:CIS 4400/Project Folder

```
C:\WINDOWS\system32>cd C:\Users\Eric Chu\OneDrive - The City University of New York\Baruch College\Fall 2021\CIS 4400\Pr
oject Folder

C:\Users\Eric Chu\OneDrive - The City University of New York\Baruch College\Fall 2021\CIS 4400\Project Folder>ipython 31
1_Loading.py
Wall time: 0 ns
Wall time: 2min 52s
100%|██████████████████████████████████████████████████████████████████████████| 1/1 [00:00<?, ?it/s]
Press ENTER to exit
```

Edit Trigger

Begin the task: On a schedule

Settings

○ One time
● Daily
○ Weekly
○ Monthly

Start: 12/14/2021 7:22:47 PM □ Synchronize across time zones

Recur every: 1 days

Advanced settings

□ Delay task for up to (random delay): 1 hour

□ Repeat task every: 1 hour for a duration of: 1 day

□ Stop all running tasks at end of repetition duration

□ Stop task if it runs longer than: 3 days

□ Expire: 12/14/2022 8:55:28 PM □ Synchronize across time zones

□ Enabled

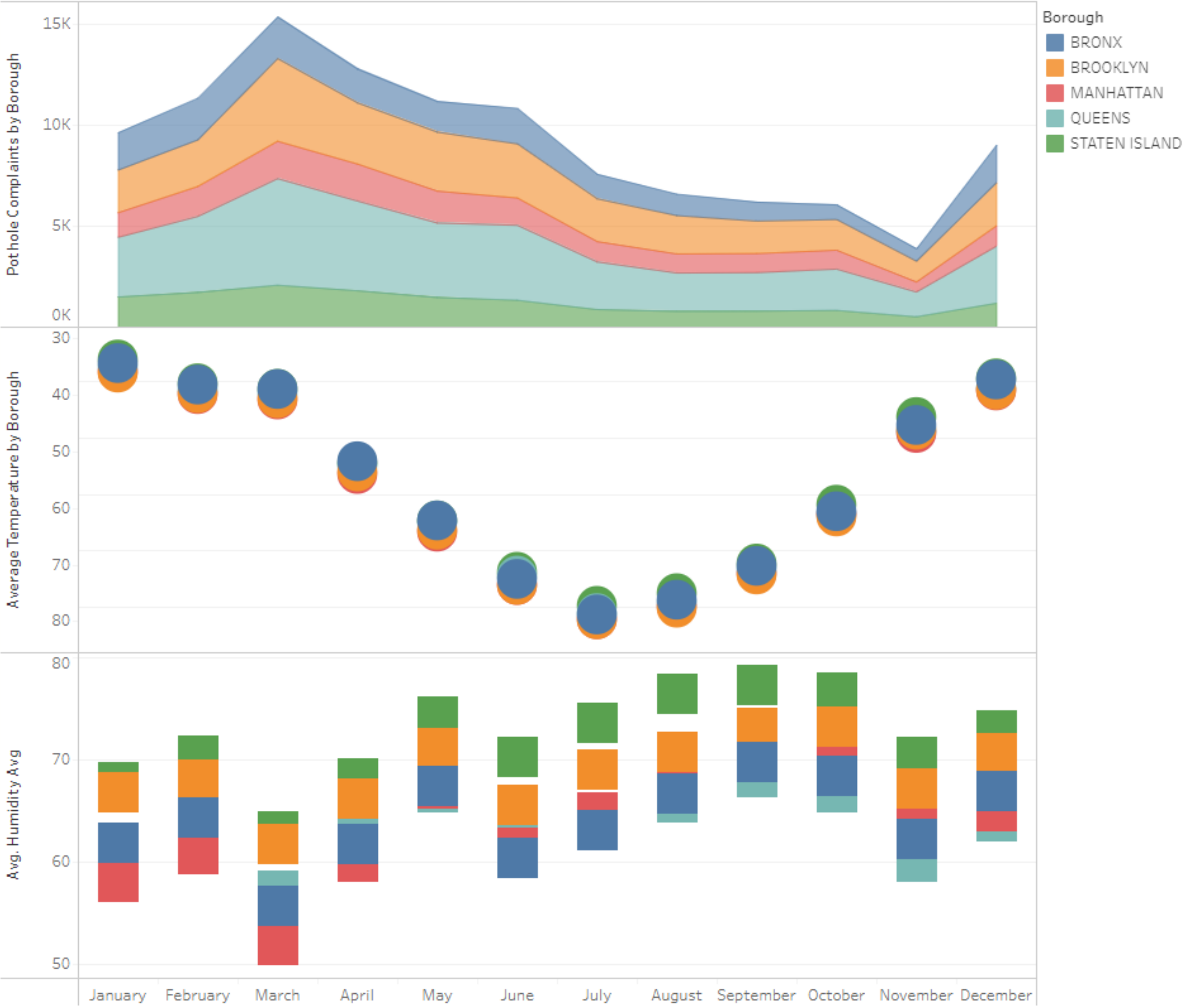
The command prompt shows a successful automated 311 data pull from the Socrata API.

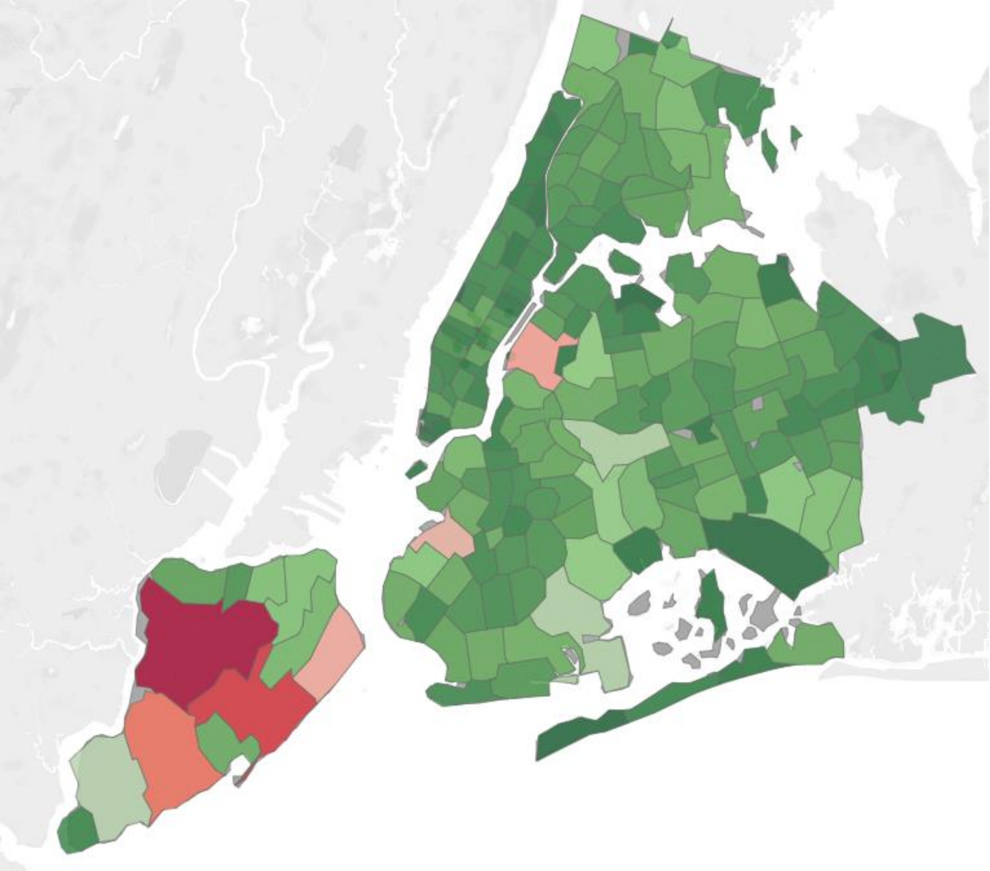
VII. Visualization KPIs

Link to our interactive Tableau Storyboard

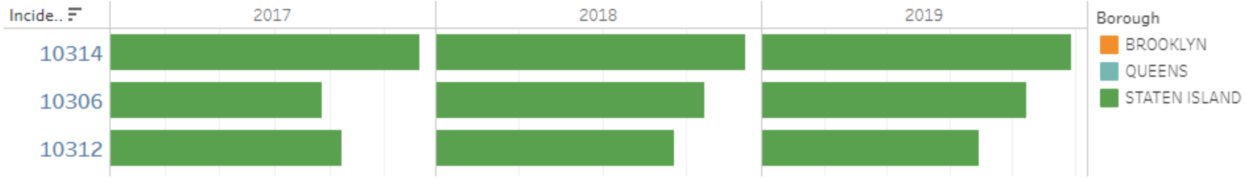
With our visualizations we strove to find visible correlations in the data. We did glean new information also from the data.

The following figure showed the largest correlation on temperature influencing an increase of pothole complaints. All the complaints were also filtered so there were no duplicates and these complaints were observed by the Department of Transportation so they were confirmed potholes. The following graphs which can also be seen in more detail within the link provide a view of the colder months correlating to an increase of potholes. There is a time lag between the coldest weather and potholes occurrence. The weather data below is reversed to show a similar curve. Also there is humidity data, and it seems to show that large shifts in humidity also contribute to potholes, making February, March and April prime months for potholes to occur.





The above heat map was also possible through tableau to show the occurrences of potholes throughout NYC zip codes. The dark green having less and dark red having more pothole complaints. After seeing this a portion of Staten Island stood out right away which led us to look deeper into there



The data showed that these 3 zip codes in Staten Island had the most pothole complaints over the three years observed. When doing further digging into those zip codes it doesn’t seem that weather shifts were more of a contributing factor than they were with other zip codes. It seems another variable is at play here, perhaps that area being the closest to the Ramapo fault line then anywhere else in New York City.

VIII. Conclusion

Software and Database Tools

Question: *The software and database tools the group used to coordinate and manage the project as well as carry out the programming tasks (list of bullet points with software or service and one sentence of what it was used for)*

1. **The Socrata Open Data API:** In order to extract the data
2. **Google Bigquery** : In order to save data
3. **DBT Cloud:** In order to transform and upload to the data warehouse
4. **Tableau:** In order to visualize the KPI and do further analysis

Question: *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?) c) if the proposed benefits can be realized by the new system. d) any final comments and conclusions*

As a group, we felt that the project was difficult but very rewarding in the end. We thought that the most difficult part of the project was the ETL process. This part was the most time consuming and the way Google BigQuery and DBT Cloud worked made it hard to coordinate with our group members. We had to share screens on Zoom and walk through the process together on one person's screen which made the process more complex. It would have been better if we could meet in person over someone's computer, but it was already hard for us to meet online since the majority of us are in senior year and we had many other obligations. It was also difficult to come up with ETL code and write the Python script to pull 311 data from the API daily because we had to know and review both SQL and Python syntax when we honestly haven't used it in a while. However, thanks to the efforts of the team and many hours of trial and error, we were able to create the right code that helped us answer our proposed idea for this project.

The easiest part of this project was using Tableau to create a basic "dashboard" application that visualized our KPI's. We say that it's the easiest because we already had all the data that we needed to create these visualizations in Tableau. It was honestly rewarding and very fun to see all of our hard work from the previous months be put together into something that we could visualize for everybody to see.

While working as a group on this project together, we learned how important teamwork and communication skills are, gained real world hands-on experience creating a data warehouse, and learned how to visualize the data from the data warehouse. It was great to apply the knowledge that we learned in class into a real data warehouse.

If we were given the opportunity to do it all over again, we would definitely manage time better and change the way that our meetings were structured. We would have made it so that we were consistent and had weekly meetings instead of scheduling meetings last minute. If we had scheduled our meetings at least a week in advance, it would ensure that everyone could make time for the meeting, everyone would be on top of things, we wouldn't have to rush the project to meet the deadlines, and we wouldn’t have to work on the project during odd hours.

The system can tell us which month, borough has more pothole complaints. By knowing the weather ahead, we can have a general idea of which borough is going to have more complaints, so relevant departments can take some corresponding actions.

After finishing the whole project, we are happy that we are able to leverage all the knowledge we learned in class. We successfully built a dimensional data warehouse, ELT pipeline, Tableau Dashboard and determined the relationship between weather and pothole complaints. The next step we might take is to dig into the dashboard and try to find more useful information.

IX. References

List that provides the web sites and other sources for data, techniques, methods, software, etc. used to complete the project.

- NYC OpenData Street Pothole Work Orders - Closed (Dataset)
    - https://data.cityofnewyork.us/Transportation/Street-Pothole-Work-Orders-Closed-Dataset-/x9wy-ing4
- NOAA Daily Summaries Station Climate Data
    - https://www.ncdc.noaa.gov/cdo-web/datasets/GHCND/stations/GHCND:USW00094728/detail
- Google bigquery
- DBT Cloud
- Python
    - Automation
- Tableau
    - Visualization