NYC Treecovery

CIS 4400

Data Warehousing for Analytics
Section CMWA

Group 7

Group Members	Email Addresses
Xianglin Chen	XIANGLIN.CHEN@baruchmail.cuny.edu
Xingxing Gao	XINGXING.GAO@baruchmail.cuny.edu
Caroline A Miller	CAROLINE.MILLER@baruchmail.cuny.edu
Venkata R Singavarapu	VENKATA.SINGAVARAPU@baruchmail.cuny.edu
Shu Ying Zou	SHUYING.ZOU@baruchmail.cuny.edu

Introduction

Our group addressed the nature of the City of New York's response to ailing trees reported to 311. We incorporated three types of complaints: Damaged Tree, Dead/Dying Tree, and Illegal Tree Damage. In particular, we examined the city's response times, using the Created Date and Closed Date values to calculate the duration of the city's response to each complaint. Because the 311 complaint tickets include location data, we were also able to analyze whether location (at the borough level) is correlated with response time. Though the COVID-19 pandemic certainly affected the 311 complaint process, we ultimately chose to incorporate data from 2017 through 2020.

We also incorporated daily weather information from Weather Underground for our second dataset. This source offered several data points regarding daily weather (from temperature to windspeed), recorded at several stations around the city, dating back several years. We were able to use this dataset to examine the effect of inclement weather upon the city's response to tree complaints.

KPIs

1. Response time per Damaged Tree, Dead/Dying Tree, or Illegal Tree Damage complaint
2. Response time per borough
3. Number of cases opened per month
4. Number of cases opened per heavy rainy day
5. Number of cases opened per heavy snowy day

Data Sources

- DoITT NYC Gov Lab & Studio. "311 Service Requests from 2010 to Present." *NYC Open Data*, City of New York, 10 Oct. 2011, https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9.

- Daily Weather Dataset, Weather Underground (Static copy provided by Professor Holowczak; exact citation unavailable.

Grain

We selected periodic grain for this dimension, because we intend to analyze the number of complaints per day, and because the available weather data is also limited to a daily scope, so we would not have accurate weather data for any length of time greater than one day. As a result, we will be using periodic grains for both data lakes.

Dimensional Model Details

Date Dimension

- Date_Dim_ID
- Full_Date
- Month_Name
- Month_Number
- Year

Location Dimension

- Location_Dim_ID
- Latitude
- Longitude
- City
- State
- ZipCode
- Borough

Agency Dimension

- Agency_Dim_ID
- Agency_Name

Alert_Levels_Dimension

- Alert_Levels__Dim_ID
- Weather_Type
- Alert_Levels

Complaint Type

- Complaint_Type_ID
- Complaint_Type
- Cumulative_Response_Time
- Status

Complaint Fact

- Complaint_Type_Dim_ID(fk)
- Location_Dim_ID(fk)
- Created_Date_Dim_Id(fk)
- Closed_Date_Dim_Id(fk)
- Agency_Dim_Id(fk)

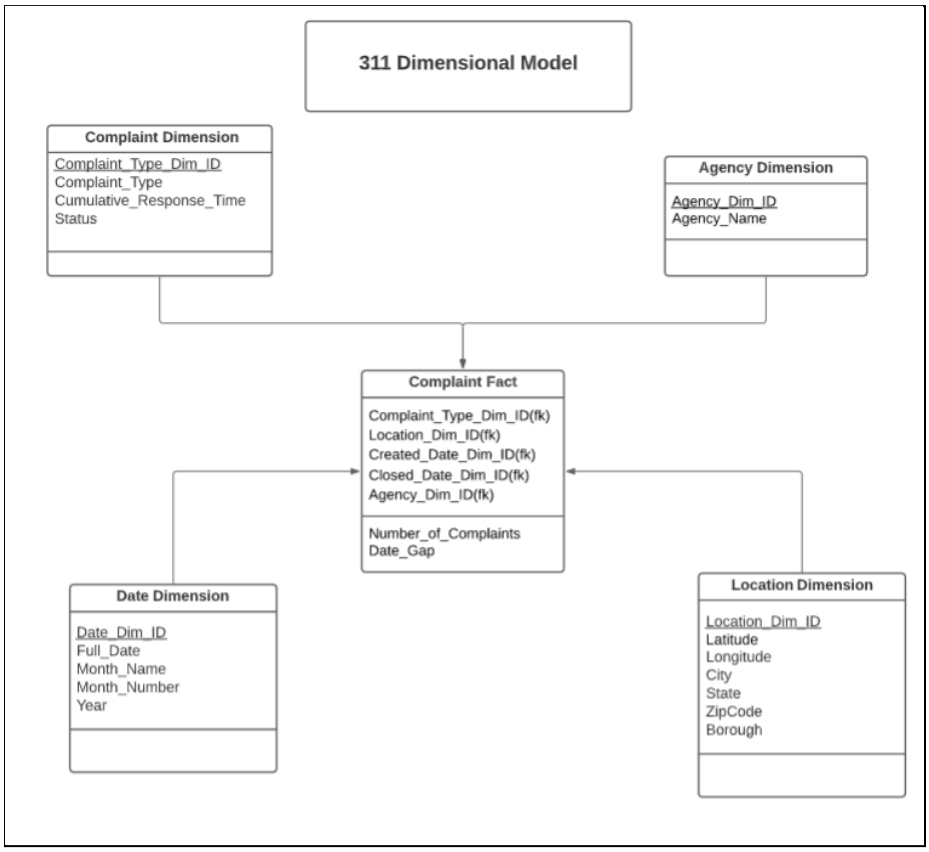
Number_of_Complaints
Date_Gap

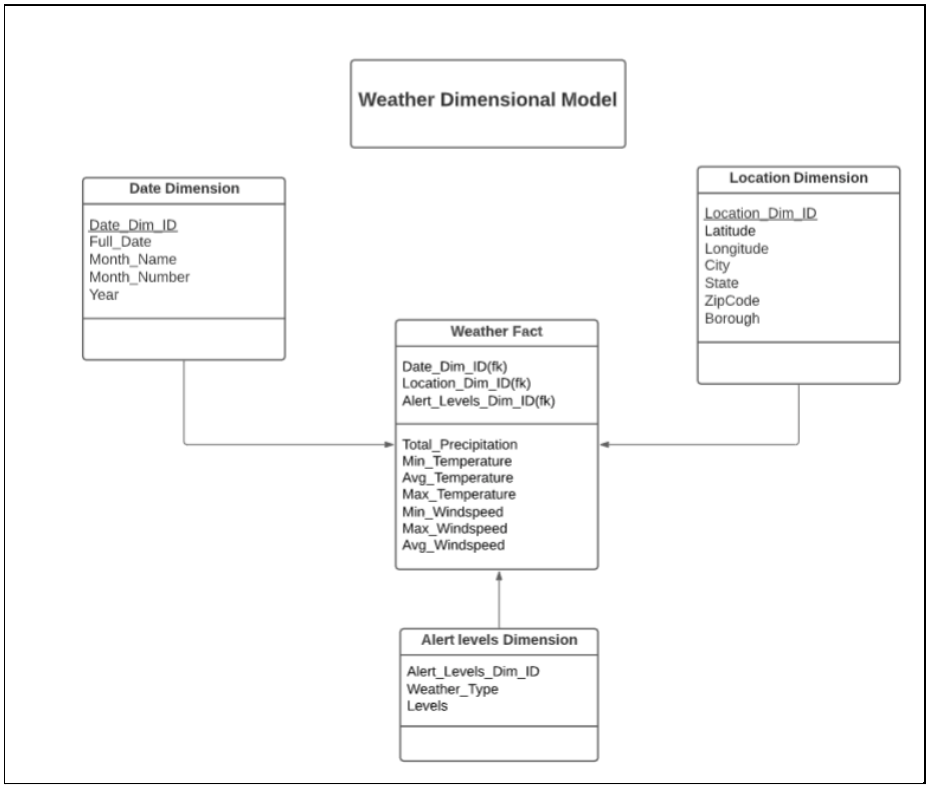
Weather Fact

- Date_Dim_ID(fk)
- Location_Dim_ID(fk)
- Alert_Levels_Dim_ID(fk)

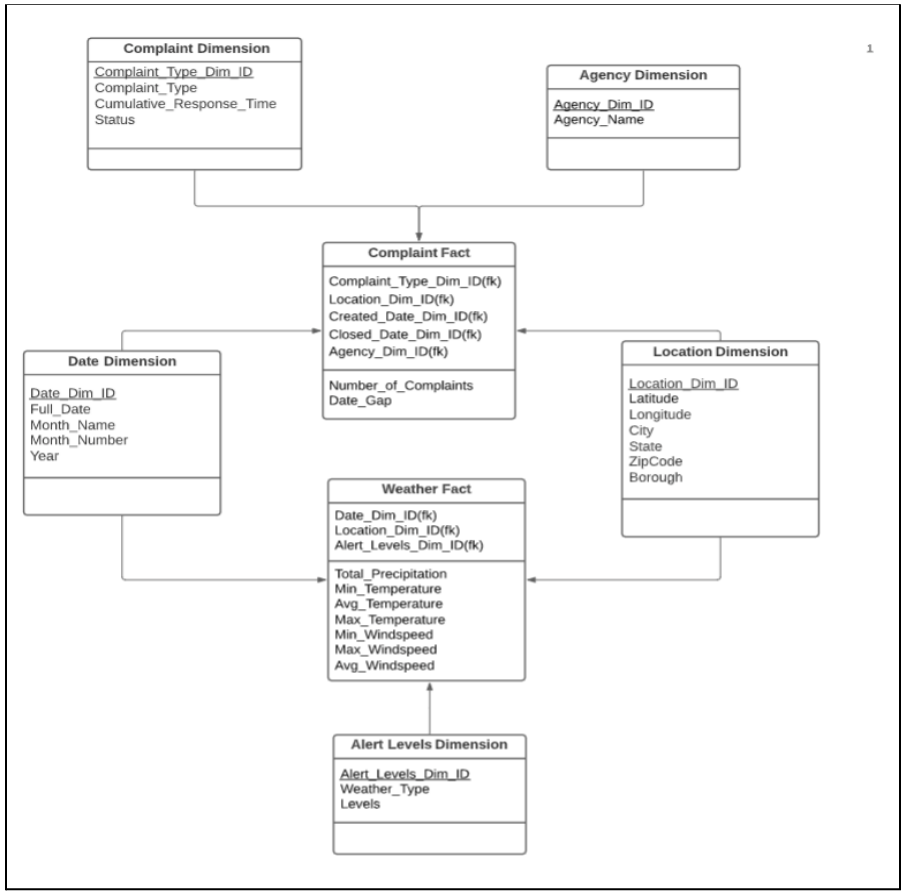
Total_Precipitation
Min_Temperature
Avg_Temperature
Max_Temperature
Min_Windspeed
Max_Windspeed
Avg_Windspeed

Original Dimensional Models





Final Dimensional Model on Lucidchart



Data Sources: Weather Data

We retrieved daily weather data for the corresponding years from the website WeatherUnderground, by extracting the data to a csv file and storing it in Excel.

Using Python, we created k-means clusters to group the data into logical tiers for alert levels and weather type. After the clustering, a level of 4 indicates the most serious inclement weather. Weather type has been clustered into the following categories: Sunny, Windy, Light Snow, Heavy Snow, Snow Storm, Light Rain, Heavy Rain, Storm. The alert levels and weather type will both be appended into the alert levels dimension. The k-means cluster for weather was based on the methods found in "Weather Data Clustering using K-means" by Prakhar Rathi on Kaggle, which can be viewed here: kaggle.com/prakharrathi25/weather-data-clustering-using-k-means.

We also reformatted the date values, which were originally stored in one field, such that we now have three separate fields containing the month, day, and year values. This was necessary for future date-based analysis, so that we can more easily refer to each of those values in our code.

In sum, via the above steps, we improved the accessibility of our data values and created a dataset that can be analyzed much more easily and efficiently – without any loss of data integrity or correctness. We exported this new and improved dataset as CSV files. All work is shown below on the following pages, along with sample datasets.

```
def pd_centers(featuresUsed, centers):
    colNames = list(featuresUsed)
    colNames.append('prediction')

    # Zip with a column called 'prediction' (index)
    Z = [np.append(A, index) for index, A in enumerate(centers)]

    # Convert to pandas data frame for plotting
    P = pd.DataFrame(Z, columns=colNames)
    P['prediction'] = P['prediction'].astype(int)
    return P
P = pd_centers(features, centers)
P
```

	Temperature_M	Temperature_Avg	Temperature_Max	Wdspeed_M	Wdspeed_Avg	Wdspeed_Max	Precipitation_Total	prediction
0	72.279878	79.920732	88.776829	0.173171	3.921341	16.629878	0.118293	0
1	22.097895	28.358947	34.472632	0.288421	5.069474	18.285263	0.028316	1
2	55.515976	63.566272	72.674556	0.146746	3.018343	13.358580	0.119822	2
3	41.511321	48.529245	55.975472	0.277358	7.666038	28.218868	0.219811	3
4	33.447208	39.439594	45.442132	0.161929	2.943147	12.106091	0.125178	4
5	45.573973	52.286986	59.826027	0.060959	1.986301	10.111644	0.153219	5
6	66.392045	73.238068	81.115341	0.005114	1.381818	8.236364	0.129034	6

```
Type = []
for row in range(len(Test)):
    if Test['Temperature_Avg'][row] <= 48.529245 or Test['Temperature_M'][row] <= 41.511321 and Test['Temperature_Max'][row] <= 55.975472:
        if Test['Precipitation_Total'][row] <= 0.219811 and Test['Precipitation_Total'][row] > 0 :
            Type.append('Light Snow')


        elif Test['Precipitation_Total'][row] >  0.219811:
                if Test['Wdspeed_Max'][row] > 28.218868:
                    Type.append('SnowStorm')
                else:
                    Type.append('Heavy_Snow')

        else :
            if Test['Wdspeed_Avg'][row] >= 5.300000 and Test['Wdspeed_Max'][row]>= 19.900000 :
                Type.append('Windy')
            else:
                Type.append('Sunny')

    else:
        if Test['Precipitation_Total'][row] < 0.219811 and Test['Precipitation_Total'][row] > 0:
            Type.append('Light_Rain')

        elif Test['Precipitation_Total'][row] > 0.219811:
            if Test['Wdspeed_Max'][row] > 28.218868:
                Type.append('Storm')
            else:
                Type.append('Heavy_Rain')

        else:
            if Test['Wdspeed_Avg'][row] >= 5.300000 and Test['Wdspeed_Max'][row]>= 19.900000 :
                Type.append('Windy')

            else:
                Type.append('Sunny')

Test['Weather Type'] = Type
```

```
import pandas as pd
Month_change = ['January','February','March','April','May','June','July','August','September','October','November','December']
Final = pd.read_csv('TreeRecovery_WeatherFinal.csv')
Month_list = []
Month_number_list=[]
Day_list =[]
Year_list=[]
ChangeDate = []

def changeMonthDay(i):
    if len(i)>1:
        return i
    else:
        return('0'+ i)

for row in range(len(Final)):
    Date = Final['WDate'][row].split('/')
    Year = '20'+ Date[2]
    ChangeDate.append(Year + '-' + changeMonthDay(Date[0])+ '-' + changeMonthDay(Date[1]))
    Day = Date[1]
    Month = Month_change[int(Date[0])-1]
    Month_Number = Date[0]
    Month_list.append(Month)
    Month_number_list.append(Month_Number)
    Day_list.append(Day)
    Year_list.append(Year)

Final['Year'] = Year_list
Final['Month'] = Month_list
Final['Month_Number'] = Month_number_list
Final['Day'] = Day_list
Final['Date'] = ChangeDate

del Final['Unnamed: 0']
del Final['WeatherID']
del Final['WDate']
Final.to_csv("CIS4400_WeatherFinal.csv")
```

Sample Weather Datasets

	Station	latitude	longitude	Borough	City	State	ZipCode	Temperature_Max	Temperature_Avg	Temperature_M	Dewpot_Max	Dewpot_Avg	Dewpot_M	Humidity_Max	Humidity_Avg	Humidity_M	Wdspeed_Max
0	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	41.2	38.1	33.9	26.9	21.9	16.9	60	51	45	28.4
1	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	39.4	35.2	32.4	19.3	17.5	14	56	48	42	25.3
2	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	44.7	38.6	34.5	23.5	21.1	19.5	60	49	36	28.4
3	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	35.6	25.9	12.1	23.8	9	-0.8	62	48	31	36.5
4	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	28.8	18.5	10	6.5	-0.2	-6.9	64	45	21	27.5
5	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	42.1	31.8	21.3	10.9	4	-2.6	53	32	15	17.7
6	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	46.7	37.1	28.1	16.1	10.1	3.6	49	33	17	11.6
7	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	43.1	37.7	30.4	34	24.5	8.7	72	59	36	23
8	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	44.7	42.2	38.2	43	37.3	31	94	82	71	25.7
9	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	57.1	49.4	40.3	55.8	43.8	23.5	97	83	46	52.3
10	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	40.2	31.4	25.1	23	7.7	0.6	50	36	24	40.5
11	KNYBRONX14	40.8616	-73.8809	Bronx	Botanical_Garden	NY	10458	43	32	22.8	29.8	14.3	1.3	78	48	33	32.7

Wdspeed_Avg	Wdspeed_M	Pressure_Max	Pressure_M	Precipitation_Total	Level	Weather_Type	Year	Month	Month_Number	Day	Date
11.3	1.3	30.1	29.96	0	Level_2	Windy	2016	January	1	1	2016-01-01
10.1	0	30.11	29.95	0	Level_1	Windy	2016	January	1	2	2016-01-02
10.8	1.6	29.97	29.78	0	Level_4	Windy	2016	January	1	3	2016-01-03
10.2	1.1	30.39	29.86	0	Level_1	Windy	2016	January	1	4	2016-01-04
6.2	0	30.64	30.37	0	Level_1	Windy	2016	January	1	5	2016-01-05
4.7	0	30.6	30.41	0	Level_1	Sunny	2016	January	1	6	2016-01-06
2.5	0	30.42	30.18	0	Level_2	Sunny	2016	January	1	7	2016-01-07
5.4	0	30.26	30.16	0	Level_2	Windy	2016	January	1	8	2016-01-08
6.5	0	30.23	30.05	0	Level_4	Windy	2016	January	1	9	2016-01-09

Data Sources: 311 Complaints

We downloaded the 311 Complaints dataset from the NYC Open Data website by using the Socrata API in Python (Jupyter Notebook). Then, we cleaned and reformatted the data, using the same methodology in Python that we had applied to the weather dataset. In this case, we also transformed single-field date values into three fields each containing the day, month, and year values, but this transformation was applied twice, to both the Created Date and Closed Date values for each complaint ticket. For the physical address associated with each complaint, we capitalized the first letter of each word, and we added a State value into the dataset – in both cases so that the address values would precisely match the weather dataset. All work is shown below, along with sample datasets on the following page. More detail can be found on GitHub: https://github.com/Xianglin17/CIS4400-Data-Cleaning-/blob/main/ETL%20process.py

```
#Function to capitalize the first letter of each word
def changecap(words):
    final_word = ''
    if len(words)>1:
        words_list = words.split(' ')
        for word in words_list:
            word = word.capitalize()
            final_word += word
            final_word += '_'
        return final_word[:-1]
    else:
        return word


newcity = []
newborough = []
state = []
for row in range(len(df311)):
    newcity_name = changecap(df311['city'][row])
    newborough_name = changecap(df311['borough'][row])
    state.append('NY')
    newcity.append(newcity_name)
    newborough.append(newborough_name)

df311['city'] = newcity
df311['borough'] = newborough
df311['state'] = state
```

Sample 311 Complaints Datasets

CID	unique_key	agency	agency_name	complaint_type	status	city	borough	incident_zip	latitude	longitude	Created_Year	Created_Month	Created_Month_Number
0	45244842	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Ridgewood	Queens	11385	40.70475321	-73.88497841	2019	December	12
1	17573251	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Brooklyn	Brooklyn	11208	40.67258229	-73.87754275	2010	July	7
2	43202583	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Bayside	Queens	11360	40.77435649	-73.78037073	2019	July	7
3	17574216	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Brooklyn	Brooklyn	11223	40.6064722	-73.98326069	2010	July	7
4	45245555	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Bellerose	Queens	11426	40.73369602	-73.72207931	2019	December	12
5	45245562	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Bronx	Bronx	10466	40.89152155	-73.85388673	2019	December	12
6	39498033	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Brooklyn	Brooklyn	11208	40.68560339	-73.87742509	2018	June	6
7	17574396	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Bayside	Queens	11360	40.7836576	-73.77708293	2010	July	7
8	17574540	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	Flushing	Queens	11358	40.76991622	-73.79110059	2010	July	7
9	44067986	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	New_York	Manhattan	10025	40.80162003	-73.96254039	2019	October	10
10	39572677	DPR	Department_of_Parks_and_Recreation	Damaged_Tree	Closed	South_Ozone_Park	Queens	11420	40.68097166	-73.82540728	2018	June	6

Created_Day	Created_Date	Closed_Year	Closed_Month	Closed_Month_Number	Closed_Day	Closed_Date	state
25	2019-12-25	2020	January	1	3	2020-01-03	NY
3	2010-07-03	2010	July	7	6	2010-07-06	NY
5	2019-07-05	2019	July	7	9	2019-07-09	NY
3	2010-07-03	2010	July	7	22	2010-07-22	NY
26	2019-12-26	2020	March	3	31	2020-03-31	NY
26	2019-12-26	2020	September	9	25	2020-09-25	NY
19	2018-06-19	2019	October	10	20	2019-10-20	NY
3	2010-07-03	2010	September	9	10	2010-09-10	NY
3	2010-07-03	2010	July	7	6	2010-07-06	NY
16	2019-10-16	2019	October	10	17	2019-10-17	NY

Data Profiling

The following is an analysis and profiling of our data sources for both the 311 Complaint data and the Weather Underground data. The data was profiled using the pandas library in Python.

311 Complaint Data Profiling

Overview

Overview | Alerts 49 | Reproduction

Dataset statistics

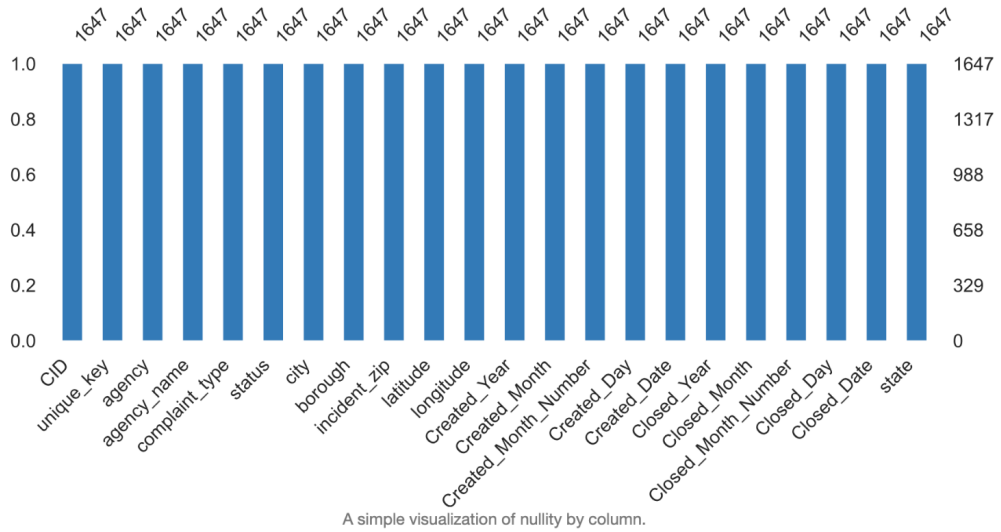
Number of variables	22
Number of observations	1647
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	283.2 KiB
Average record size in memory	176.1 B

Variable types

Numeric	11
Categorical	11

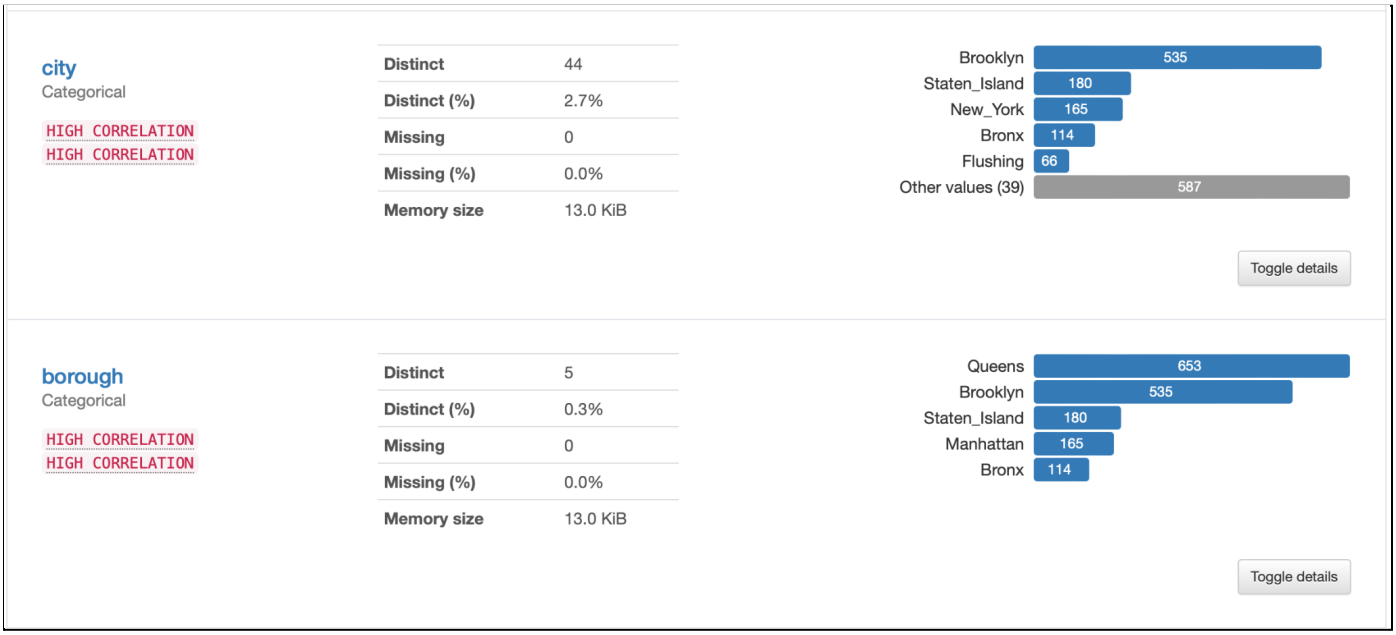
Missing values

Count | Matrix



A simple visualization of nullity by column.

311 Complaint Data: Sample Variables



city
Categorical

HIGH CORRELATION
HIGH CORRELATION

Distinct	44
Distinct (%)	2.7%
Missing	0
Missing (%)	0.0%
Memory size	13.0 KiB

borough
Categorical

HIGH CORRELATION
HIGH CORRELATION

Distinct	5
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	13.0 KiB

Weather Data Profiling

Overview

Overview | Alerts 98 | Reproduction

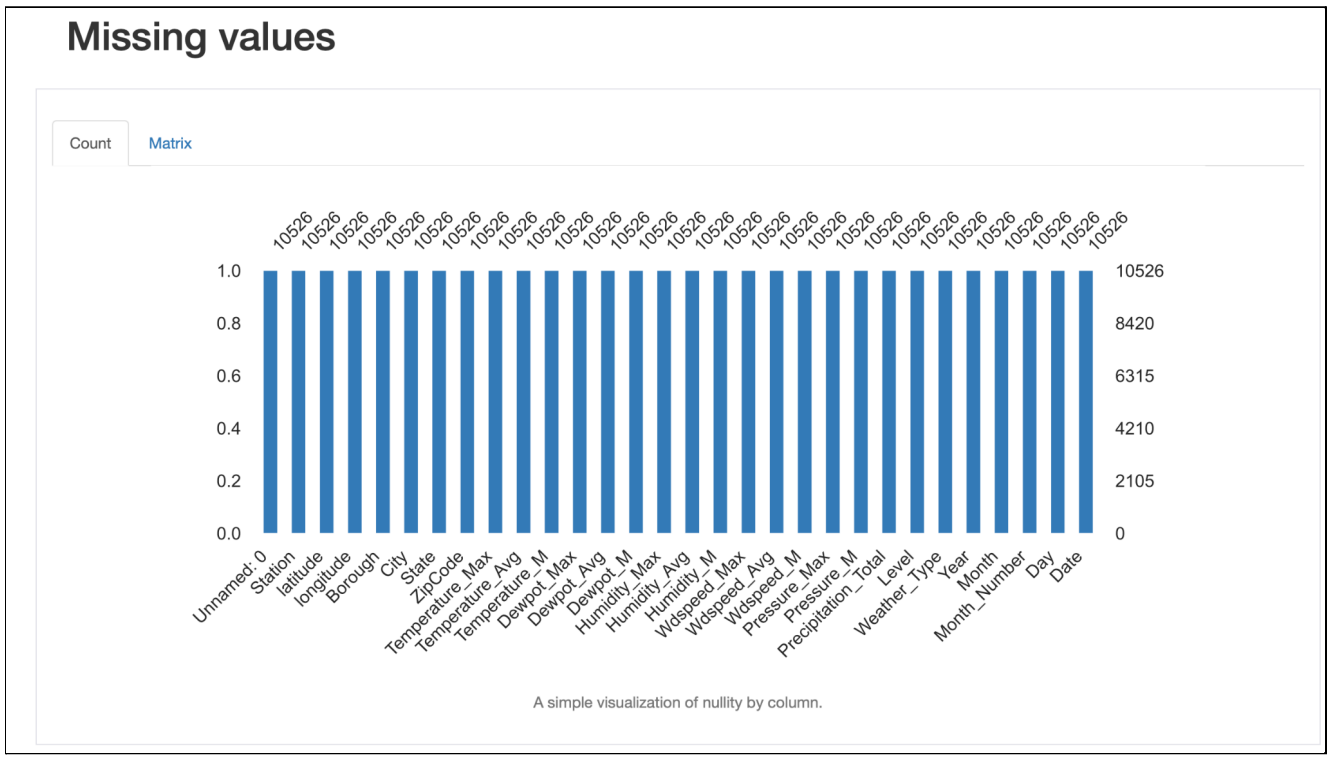
Dataset statistics

Number of variables	30
Number of observations	10526
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	2.4 MiB
Average record size in memory	240.0 B

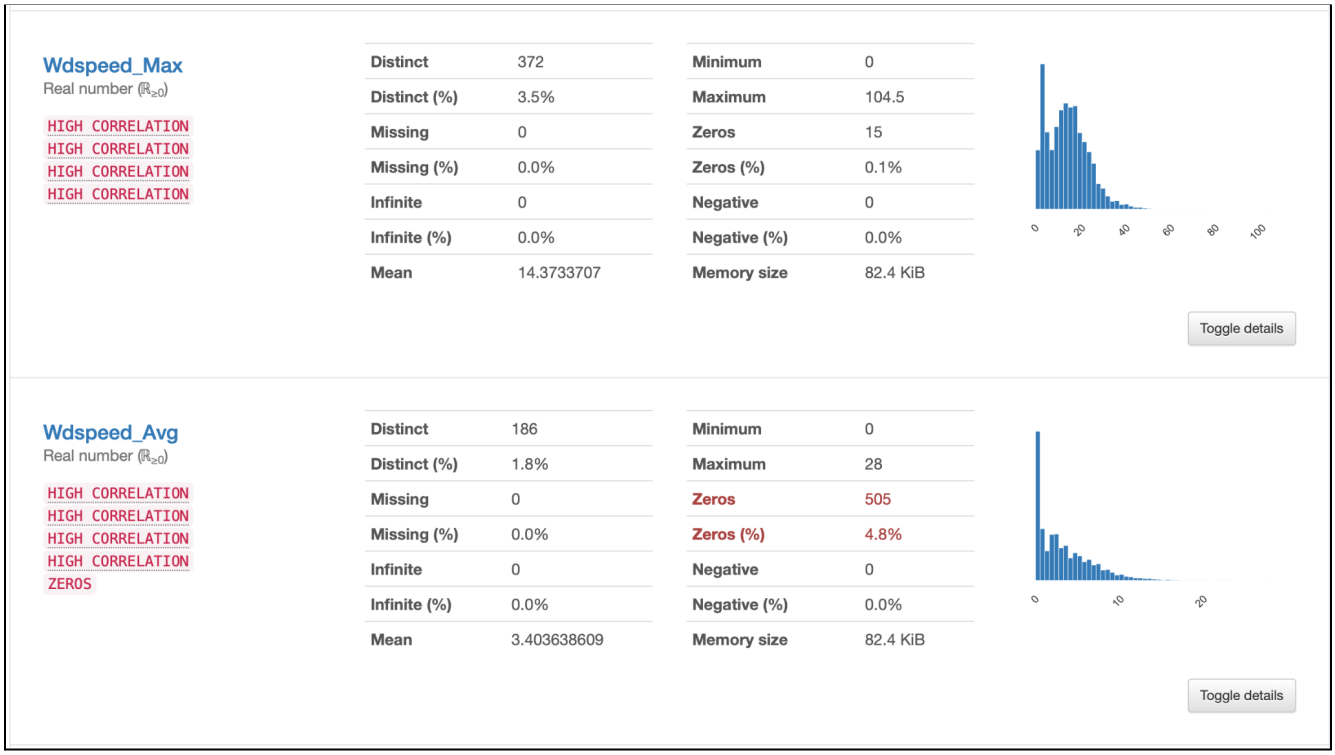
Variable types

Numeric	20
Categorical	10

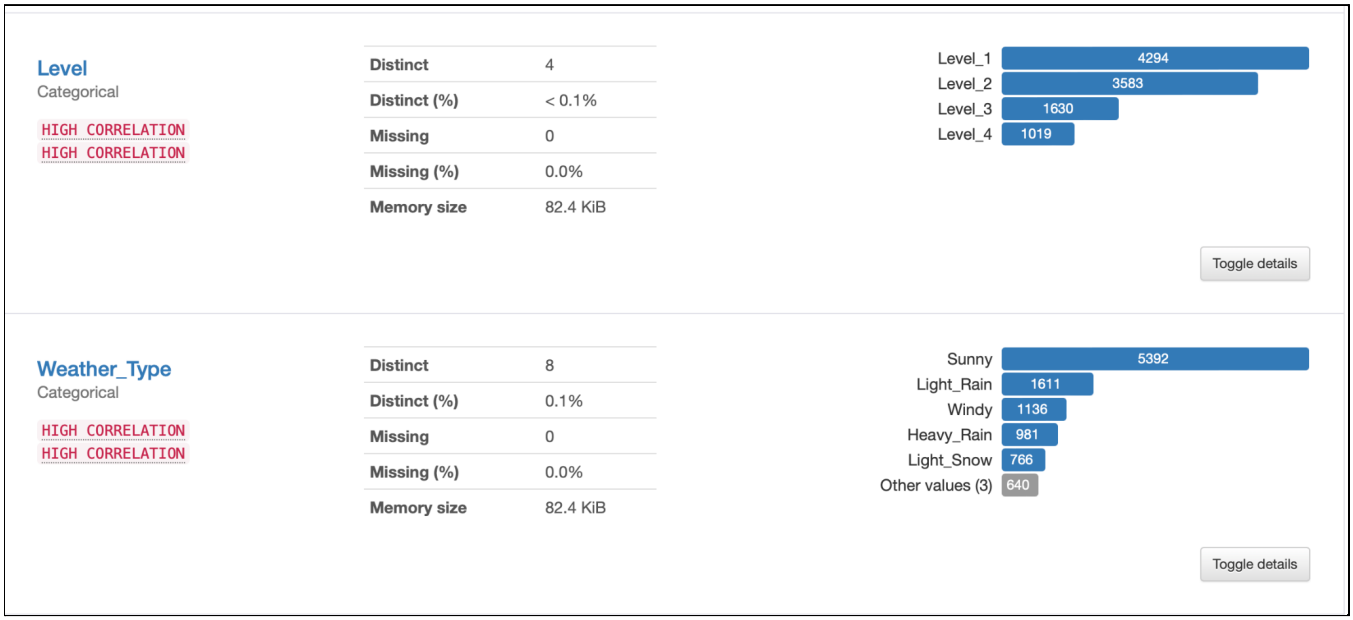
Weather Data Profiling, cont.



Weather Underground Data: Sample Variables



Weather Underground Data: Sample Variables, cont.



Level
Categorical

HIGH CORRELATION
HIGH CORRELATION

Distinct	4
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	82.4 KiB

Toggle details

Weather_Type
Categorical

HIGH CORRELATION
HIGH CORRELATION

Distinct	8
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	82.4 KiB

Toggle details

Data Profiling: HTML Reports

At the below link are two HTML reports, one each for the 311 Complaint data and the Weather Underground data, which were profiled for the full report. The reports must be downloaded, unzipped, and opened with a web browser in order to view the contents. The reports are located on GitHub here: https://github.com/galaxys10pluse/cs

ETL with dbt

More details about this ETL process and the below steps can be viewed here on GitHub: https://github.com/Xianglin17/NYC-Treecovery

Seed

After cleaning the data in Python as shown in the above steps, we copied the two datasets into dbt as new sources, using the *dbt seed* command.

311_complaints.csv

save

```
CID,unique_key,agency,agency_name,complaint_type,status,city,borough,incident_zip,latitude,longitude,Created_Year,Created_Month,Created_Month_Number,Cr
0,45244842,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Ridgewood,Queens,11385,40.70475321,-73.88497841,2019,December,12,25,2019-12-25,20
1,17573251,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Brooklyn,Brooklyn,11208,40.67258229,-73.87754275,2010,July,7,3,2010-07-03,2010,Ju
2,43202583,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Bayside,Queens,11360,40.77435649,-73.78037073,2019,July,7,5,2019-07-05,2019,July,
3,17574216,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Brooklyn,Brooklyn,11223,40.6064722,-73.98326069,2010,July,7,3,2010-07-03,2010,Jul
4,45245555,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Bellerose,Queens,11426,40.73369602,-73.72207931,2019,December,12,26,2019-12-26,20
5,45245562,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Bronx,Bronx,10466,40.89152155,-73.85388673,2019,December,12,26,2019-12-26,2020,Se
6,39498033,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Brooklyn,Brooklyn,11208,40.68560339,-73.87742509,2018,June,6,19,2018-06-19,2019,O
7,17574396,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Bayside,Queens,11360,40.7836576,-73.77708293,2010,July,7,3,2010-07-03,2010,Septem
8,17574540,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Flushing,Queens,11358,40.76991622,-73.79110059,2010,July,7,3,2010-07-03,2010,July
9,44067986,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,New_York,Manhattan,10025,40.80162003,-73.96254039,2019,October,10,16,2019-10-16,2
10,39572677,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,South_Ozone_Park,Queens,11420,40.68097166,-73.82540728,2018,June,6,26,2018-06-26
11,36117580,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Springfield_Gardens,Queens,11413,40.67312465,-73.7522966,2017,May,5,5,2017-05-05
12,36117581,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Fresh_Meadows,Queens,11366,40.73297499,-73.77388599,2017,May,5,5,2017-05-05,2017
13,44215660,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Bronx,Bronx,10465,40.82527525,-73.82249938,2019,November,11,2,2019-11-02,2020,Ja
14,39994471,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Bronx,Bronx,10463,40.88159402,-73.9216311,2018,August,8,13,2018-08-13,2020,April
15,45248003,DPR,Department_of_Parks_and_Recreation,Damaged_Tree,Closed,Brooklyn,Brooklyn,11217,40.67838529,-73.98141116,2019,December,12,26,2019-12-26,
```

311_complaints.csv | weather.csv

save

```
WeatherID,Station,latitude,longitude,Borough,City,State,ZipCode,Temperature_Max,Temperature_Avg,Temperature_M,Dewpot_Max,Dewpot_Avg,Dewpot_M,Humidity_M
0,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,41.2,38.1,33.9,26.9,21.9,16.9,60,51,45,28.4,11.3,1.3,30.1,29.96,0,Level_2,Windy,2016,Janu
1,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,39.4,35.2,32.4,19.3,17.5,14,56,48,42,25.3,10.1,0,30.11,29.95,0,Level_1,Windy,2016,January
2,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,44.7,38.6,34.5,23.5,21.1,19.5,60,49,36,28.4,10.8,1.6,29.97,29.78,0,Level_4,Windy,2016,Jan
3,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,35.6,25.9,12.1,23.8,9,-0.8,62,48,31,36.5,10.2,1.1,30.39,29.86,0,Level_1,Windy,2016,Januar
4,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,28.8,18.5,10,6.5,-0.2,-6.9,64,45,21,27.5,6.2,0,30.64,30.37,0,Level_1,Windy,2016,January,1
5,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,42.1,31.8,21.3,10.9,4,-2.6,53,32,15,17.7,4.7,0,30.6,30.41,0,Level_1,Sunny,2016,January,1,
6,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,46.7,37.1,28.1,16.1,10.1,3.6,49,33,17,11.6,2.5,0,30.42,30.18,0,Level_2,Sunny,2016,January
7,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,43.1,37.7,30.4,34,24.5,8.7,72,59,36,23,5.4,0,30.26,30.16,0,Level_2,Windy,2016,January,1,8
8,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,44.7,42.2,38.2,43,37.3,31,94,82,71,25.7,6.5,0,30.23,30.05,0,Level_4,Windy,2016,January,1,
9,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,57.1,49.4,40.3,55.8,43.8,23.5,97,83,46,52.3,14.2,0,30.06,29.29,2,Level_4,Storm,2016,Janua
10,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,40.2,31.4,25.1,23,7.7,0.6,50,36,24,40.5,16.3,2,30.14,29.6,0,Level_1,Windy,2016,January,1
11,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,43,32,22.8,29.8,14.3,1.3,78,48,33,32.7,8.1,0,30.12,29.64,0,Level_1,Windy,2016,January,1,
12,KNYBRONX14,40.8616,-73.8809,Bronx,Botanical_Garden,NY,10458,28.7,24.7,21.6,6.5,3.8,0.6,49,40,30,45,16.3,2,30.04,29.73,0,Level_1,Windy,2016,January,1
```

dbt seed

weather_dimensions

Logs

Passed	2	0	0	0	0	21:41:15	10 seconds
RUN STATUS	PASS	WARN	FAIL	SKIPPED	QUEUED	START	DURATION

SYSTEM LOGS

view logs

DETAILS

311_complaints 5s

Summary Details

```
21:41:19 | 1 of 2 START seed file dbt_xchen.311_complaints..... [RUN]
21:41:24 | 1 of 2 OK loaded seed file dbt_xchen.311_complaints..... [INSERT 1647 in 5.01s]
```

weather 6s

Summary Details

```
21:41:19 | 2 of 2 START seed file dbt_xchen.weather..... [RUN]
21:41:25 | 2 of 2 OK loaded seed file dbt_xchen.weather..... [INSERT 10526 in 5.91s]
```

Staging

We created a Staging folder in dbt, containing the initial look of the dimensional model. The date and location for 311 data and weather data are in separate dimensions. For each dimension, each row is distinct. With the exception of the location and date dimension, all other dimensions in this folder have been assigned a unique dim_ID.

311 Complaints Data

stg_311_location

stg_311_location.sql | stg_agency.sql

```sql
SELECT DISTINCT
latitude as Latitude,
longitude as Longtitude,
city as City,
incident_zip as ZipCode,
borough as Borough,
state as STATE
from {{ref('311_complaints')}}
```

stg_agency

stg_311_location.sql | stg_agency.sql | stg_complaint.sql

```sql
SELECT ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS Agency_Dim_ID, agency, agency_name
FROM (SELECT DISTINCT agency, agency_name
FROM {{ref('311_complaints')}})

```

stg_compaint

stg_311_location.sql | stg_agency.sql | stg_complaint.sql

```sql
SELECT
ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS Complaint_Type_Dim_ID,
complaint_type,
status
from (SELECT DISTINCT complaint_type,status
FROM {{ref('311_complaints')}})

```

stg_complaint_date

stg_311_location.sql | stg_agency.sql | stg_complaint.sql | **stg_complaints_date.sql**

```sql
select
Created_Year as Year,
Created_Month as Month,
Created_Day as Day,
Created_Date as Date_ID
from{{ref('311_complaints')}}
Union all
select
Closed_Year as Year,
Closed_Month as Month,
Closed_Day as Day,
Closed_Date as Date_ID
from{{ref('311_complaints')}}
```

Weather Data

stg_alert_levels

Agency_Dimension.sql | Alert_Levels_Dimension.... | Complaint_Dimension.sql | Date_Dimensior

```sql
with levels as (select DISTINCT
level,
Weather_Type
from {{ref('weather')}})

select *, ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS Alert_Levels_Dim_ID
from levels

```

stg_weather_date

stg_alert_levels.sql | **stg_weather_date.sql**

```sql
select
Year,
Month,
Day,
Date as Date_ID
from{{ref('weather')}}
```

stg_location

stg_alert_levels.sql | stg_weather_date.sql | **stg_weather_location.sql**

```sql
SELECT DISTINCT
latitude as Latitude,
longitude as Longtitude,
city as City,
ZipCode,
Borough,
state as STATE
from {{ref('weather')}}
```

all_date (includes all the date from 2010 to 2021 and assigned a unique dim_ID)

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

WITH DATE_SPINE as (
{{ dbt_utils.date_spine(
    datepart="day",
    start_date="cast('2010-01-01' as date)",
    end_date="cast('2022-01-01' as date)"
   )
}} )

SELECT
ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) as Date_Dim_ID,
(CAST(DATE_DAY AS DATE)) AS DATE_ID
FROM DATE_SPINE
```

Dimensions

In the Dimensions folder, we converted the initial look of the model to the final look by combining the date and location dimensions, assigning dim_ID where necessary, and materializing each dimension to tables.

Agency_Dimension

Agency_Dimension.sql | Alert_Levels_Di

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

select *
FROM {{ref('stg_agency')}}
```

Preview | Compile | Query Results | Compiled SQL | Lineage

3.7 sec —Returned 1 row.

Agency_Dim_ID	agency	agency_name
1	DPR	Department_of_Parks_and_Recreation

Altert_Levels_Dimension

Agency_Dimension.sql | Alert_Levels_Dimension.sql

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

select *
from {{ref('stg_alert_levels')}}
```

4.3 sec —Returned 28 rows.

level	Weather_Type	Alert_Levels_Dim_ID
Level_1	Sunny	1
Level_1	Windy	2
Level_1	SnowStorm	3
Level_1	Heavy_Rain	4
Level_1	Heavy_Snow	5
Level_1	Light_Rain	6
Level_1	Light_Snow	7

Complaint_Dimension

Agency_Dimension.sql | Alert_Levels_Dimension.sql | **Complaint_Dimension.sql**

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

SELECT *
from {{ref('stg_complaint')}}

```

Complaint_Type_Dim_ID	complaint_type	status
1	Damaged_Tree	Closed
2	Dead/Dying_Tree	Closed
3	Illegal_Tree_Damage	Closed
4	Illegal_Tree_Damage	In Progress
5	Damaged_Tree	In Progress
6	Dead/Dying_Tree	In Progress

Date_Dimension

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

with union_date as (select * from {{ref('stg_complaints_date')}}
Union ALL
select * from {{ref('stg_weather_date')}}
)

select *
FROM(
select DISTINCT *
from union_date)
Left Join {{ref('all_date')}} USING (DATE_ID)
```

DATE_ID	Year	Month	Day	Date_Dim_ID
2016-01-06	2016	January	6	2197
2016-01-20	2016	January	20	2211
2016-01-22	2016	January	22	2213
2016-02-01	2016	February	1	2223
2016-02-09	2016	February	9	2231
2016-03-05	2016	March	5	2256
2016-03-08	2016	March	8	2259
2016-03-13	2016	March	13	2264

Location_Dimension

Agency_Dimension.sql | Alert_Levels_Dimension.sql | Complaint_Dimension.sql | Date_Dimension.sql | **Location_Dimension.sql** | +

save

```
1   {{ config (
2       materialized="table"
3   )}}
4   with total as(
5   select *
6   from {{ref('stg_311_location')}}
7   Union All
8   select *
9   from {{ref('stg_weather_location')}}
10  )
11
12  SELECT DISTINCT *,ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS location_dim_ID
13  from total
```

Preview | Compile | **Query Results** | Compiled SQL | Lineage

3.8 sec —Results limited to 500 rows. Download CSV

Latitude	Longtitude	City	ZipCode	Borough	STATE	location_dim_ID
40.73297499	-73.77388599	Fresh_Meadows	11366	Queens	NY	1
40.69967439	-73.7664831	Saint_Albans	11412	Queens	NY	2
40.6888609	-73.83143725	South_Richmond_Hi…	11419	Queens	NY	3
40.70808239	-74.0034013	New_York	10038	Manhattan	NY	4
40.61184813	-73.77135007	Far_Rockaway	11691	Queens	NY	5
40.65738686	-73.97659466	Brooklyn	11215	Brooklyn	NY	6
40.79766761	-73.93726401	New_York	10035	Manhattan	NY	7
40.88831514	-73.91310592	Bronx	10463	Bronx	NY	8

Fact Tables

In the Fact Tables folder, we inserted the dim_id into the fact table, as well as the necessary weather metric values and complaint details, such as Total_Precipitation, Min_Temperature , Avg_Temperature, Max_Temperature , Min_Windspeed, Max_Windspeed, Avg_Windspeed, date gap, and number of complaints.

Complaint_Fact

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

with complaint_table as (
    select * from {{ref('Complaint_Dimension')}}
),
agency_table as (
    select * from {{ref('Agency_Dimension')}}
),
date_table as (
    select * from {{ref('Date_Dimension')}}
),
location_table as (
    select *
    from {{ref('Location_Dimension')}}
),
closed_date_table as (
    select  Date_Dim_ID as Closed_Date_Dim_ID,
            DATE_ID as Closed_Date

          FROM (select DISTINCT
                Closed_Year as Year,
                Closed_Month as Month,
                Closed_Day as Day,
                Closed_Date as DATE_ID,
                from {{ref('311_complaints')}}
          )
    INNER JOIN {{ref('all_date')}} USING (DATE_ID)

),
total_table as (
    select Created_Year as Year,
            Created_Month as Month,
            Created_Day as Day,
            Created_Date as Date_ID,
            Closed_Date,
            latitude as Latitude,
            longitude as Longtitude,
            city as City,
            incident_zip as ZipCode,
            borough as Borough,
            state as STATE,
            agency,
            agency_name,
            complaint_type,
            status
            from {{ref('311_complaints')}}
            where Created_Year > 2016
)
SELECT Complaint_Type_Dim_ID,
        Agency_Dim_ID,
        Created_Date_Dim_ID,
        Closed_Date_Dim_ID,
        Location_Dim_ID,
        count(Final_ID) as Number_of_Complaint,
        ABS(date_diff(DATE(Closed_Date),DATE(Date_ID),day)) as Date_Gap

FROM (
SELECT Complaint_Type_Dim_ID,
        Agency_Dim_ID,
        Date_Dim_ID as Created_Date_Dim_ID,
        Closed_Date_Dim_ID,
```

Complaint_Fact (cont.)

```
64          Location_Dim_ID,
65          ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) as Final_ID,
66          Closed_Date,
67          Date_ID
68
69
70      from total_table
71      INNER JOIN complaint_table USING (complaint_type,status)
72      INNER JOIN agency_table USING (agency,agency_name)
73      INNER JOIN date_table USING (Year, Month, Day, Date_ID)
74      INNER JOIN location_table USING (Latitude,Longtitude,City,ZipCode,Borough,STATE)
75      INNER JOIN closed_date_table USING (Closed_Date)
76      )
77
78      Group by Complaint_Type_Dim_ID, Agency_Dim_ID, Created_Date_Dim_ID, Location_Dim_ID, Closed_Date_Dim_ID, Closed_Date, Date_ID
```

Complaint_Type_Dim_ID	Agency_Dim_ID	Created_Date_Dim_ID	Closed_Date_Dim_ID	Location_Dim_ID	Number_of_Complaint	Date_Gap
1	1	2682	2683	1	1	1
1	1	2683	2686	2	1	3
1	1	2691	2696	3	1	5
1	1	2692	2692	4	1	0
1	1	2699	2708	5	1	9
1	1	2700	2703	6	1	3
1	1	2701	2703	7	1	2
1	1	2702	2703	8	1	1
2	1	3774	3794	9	1	20

N.B. In the *closed_date* table, each closed date transaction has been assigned a dim key by joining with the *all_date* table.

Weather_Fact

Complaint_Fact.sql | **Weather_Fact.sql**

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

with alert_levels_table as (
    select * from {{ref('Alert_Levels_Dimension')}}
),
datetable_table as (
    select *
    from {{ref('Date_Dimension')}}
),
location_table as (
    select *
    from {{ref('Location_Dimension')}}
),
total_table as (
    select  Year,
            Month,
            Day,
            Date as Date_ID,
            latitude as Latitude,
            longitude as Longtitude,
            City,
            ZipCode,
            Borough,
            STATE,
            level,
            Weather_Type,
            Precipitation_Total,
            Temperature_Max,
            Temperature_Avg,
            Temperature_M,
```

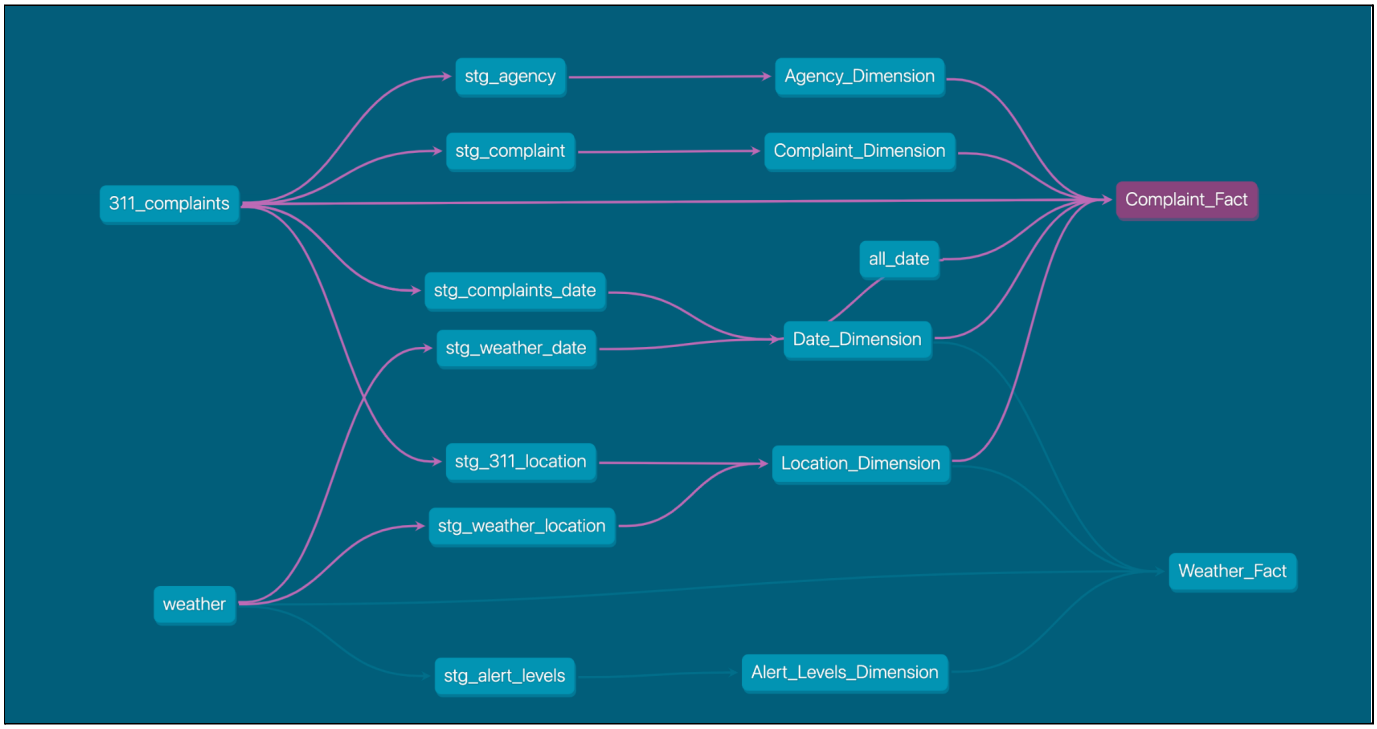
Weather_Fact (cont.)

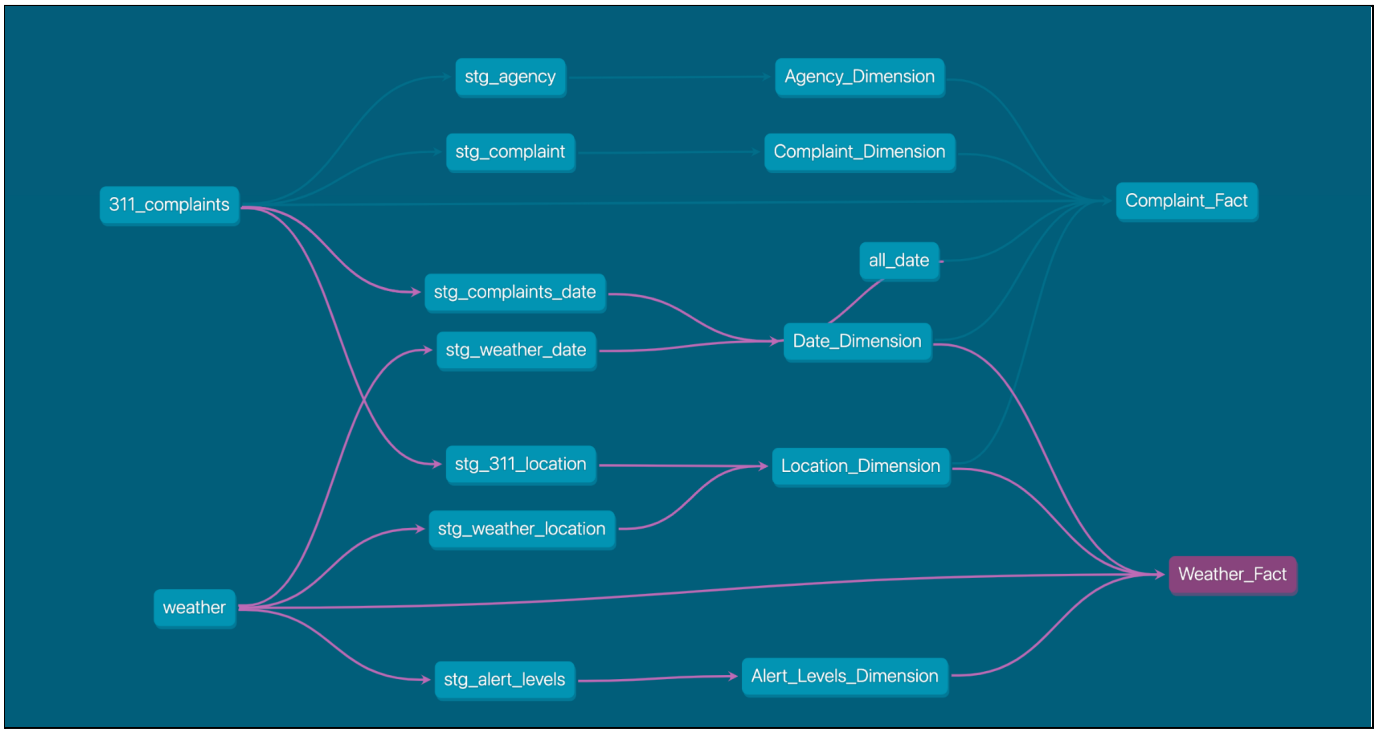
```
            Wdspeed_Avg,
            Wdspeed_Max,
            Wdspeed_M,
            from {{ref('weather')}}
)

select Alert_Levels_Dim_ID ,
       Date_Dim_ID,
       Location_Dim_ID,
       Precipitation_Total,
       Temperature_Max,
       Temperature_Avg,
       Temperature_M as Temperature_Min,
       Wdspeed_Avg as Average_Wind_Speed,
       Wdspeed_Max as Max_Wind_Speed,
       Wdspeed_M as Min_Wind_Speed
from total_table
LEFT JOIN alert_levels_table USING (level, Weather_Type)
LEFT JOIN datetable_table USING (Year, Month, Day, Date_ID)
LEFT JOIN location_table USING (Latitude,Longtitude,City,ZipCode,Borough,STATE)
Order by Date_Dim_ID
```

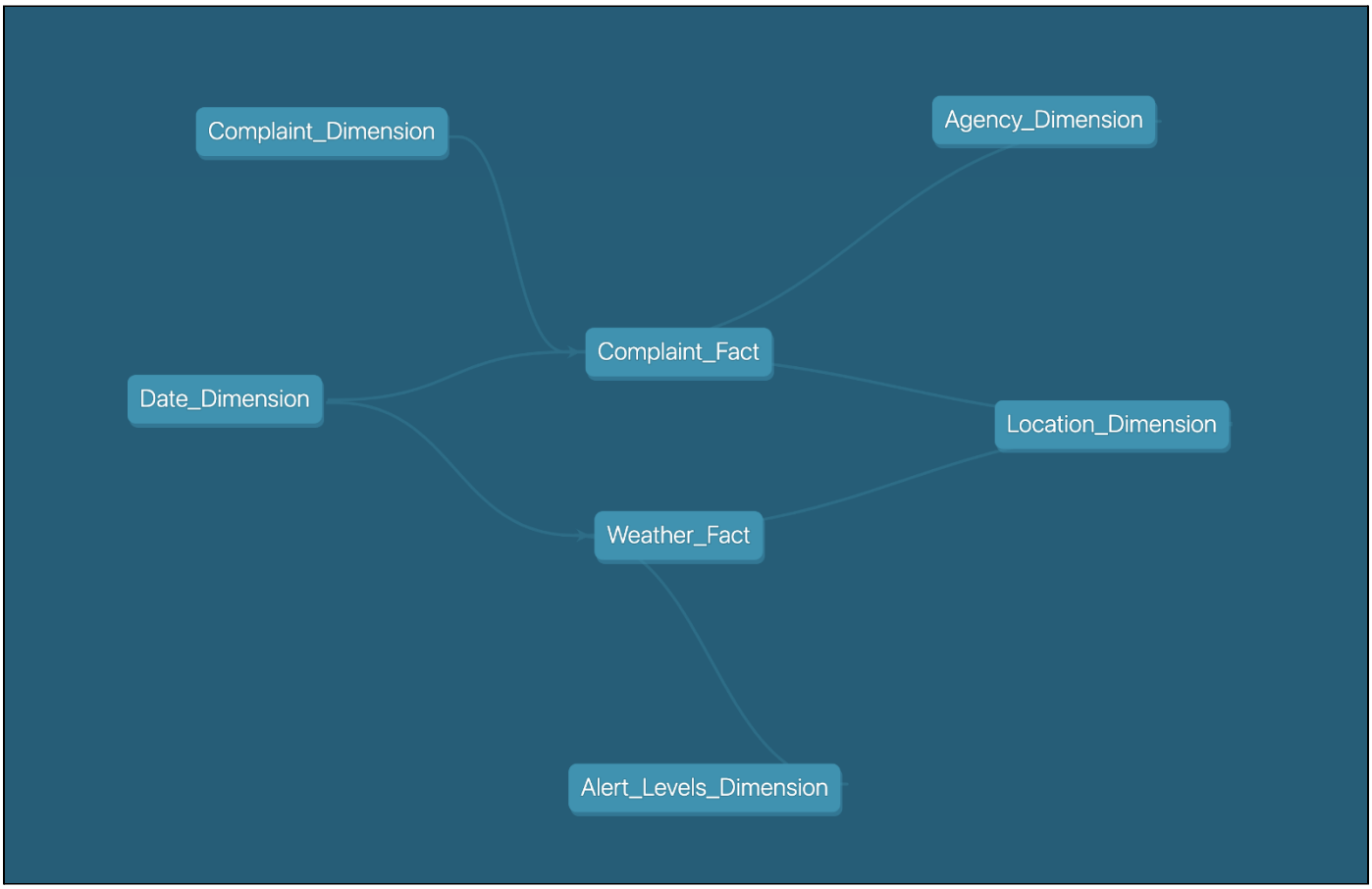
Alert_Levels_Dim_ID	Date_Dim_ID	Location_Dim_ID	Precipitation_Total	Temperature_Max	Temperature_Avg	Temperature_Min	Average_Wind_Speed	Max_Wind_Speed	Min_Wind_
9	116	3	0	43.2	40.2	35.6	0.2	3.4	0
10	116	4	0	42.2	39.6	36	8.7	21	0.9
10	116	1	0	41.2	38.1	33.9	11.3	28.4	1.3
9	116	5	0	41.7	37.9	32.7	4.2	15	0
10	116	2	0	42.5	39.4	35	7.1	20	1
9	117	2	0	40.7	36.5	33.5	6.1	15	0
9	117	3	0	40.6	36.7	33.8	0.1	1.3	0

Final Looks


311_complaints
stg_agency
stg_complaint
stg_complaints_date
stg_weather_date
stg_311_location
stg_weather_location
weather
stg_alert_levels
Agency_Dimension
Complaint_Dimension
all_date
Date_Dimension
Location_Dimension
Alert_Levels_Dimension
Complaint_Fact
Weather_Fact


311_complaints
stg_agency
stg_complaint
stg_complaints_date
stg_weather_date
stg_311_location
stg_weather_location
weather
stg_alert_levels
Agency_Dimension
Complaint_Dimension
all_date
Date_Dimension
Location_Dimension
Alert_Levels_Dimension
Complaint_Fact
Weather_Fact

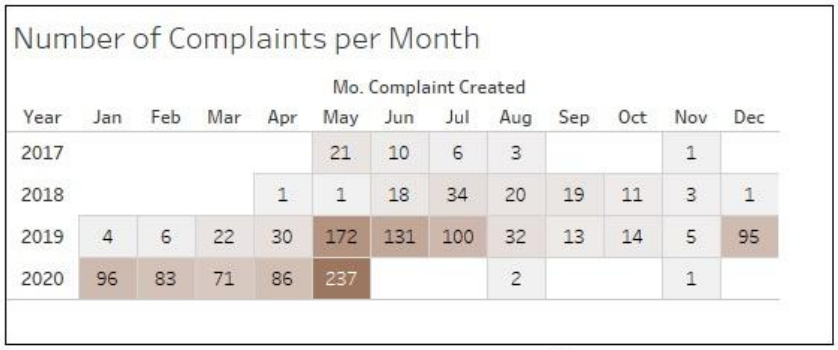
Final Schema

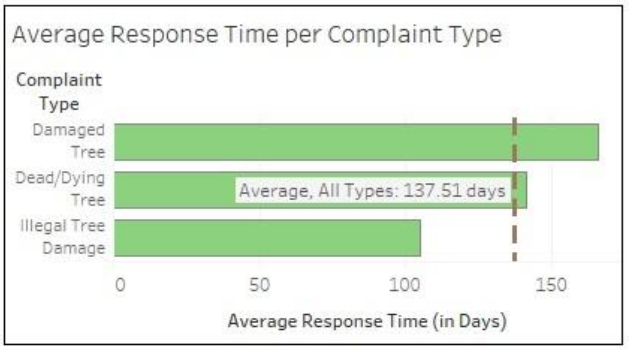


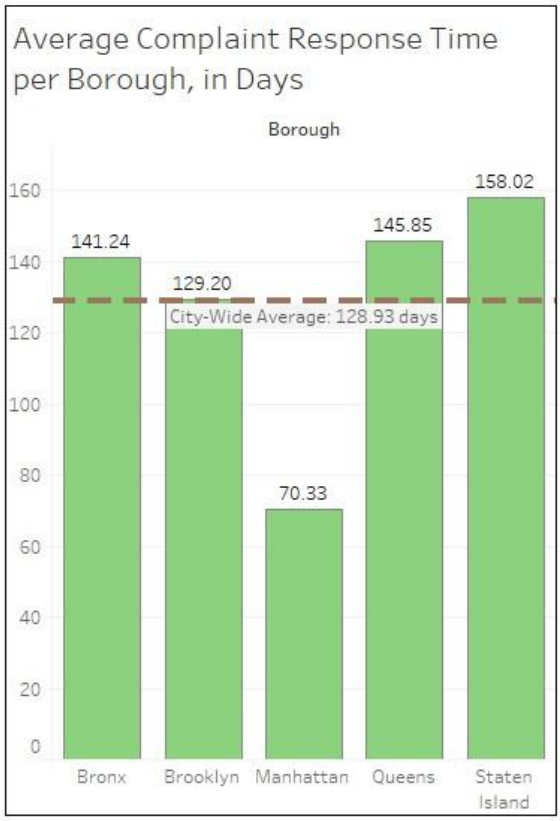
The final schema was built based on the dimensional models designed in Lucidchart. It includes five dimensions and two fact tables by using the distinct dim_ID. Each transaction in each dimension is unique and has its own dim_ID. The fact tables provided additional information about necessary weather metric values and complaint details. This allowed the dimensions and fact tables to be joined easily when creating the KPI visualizations.

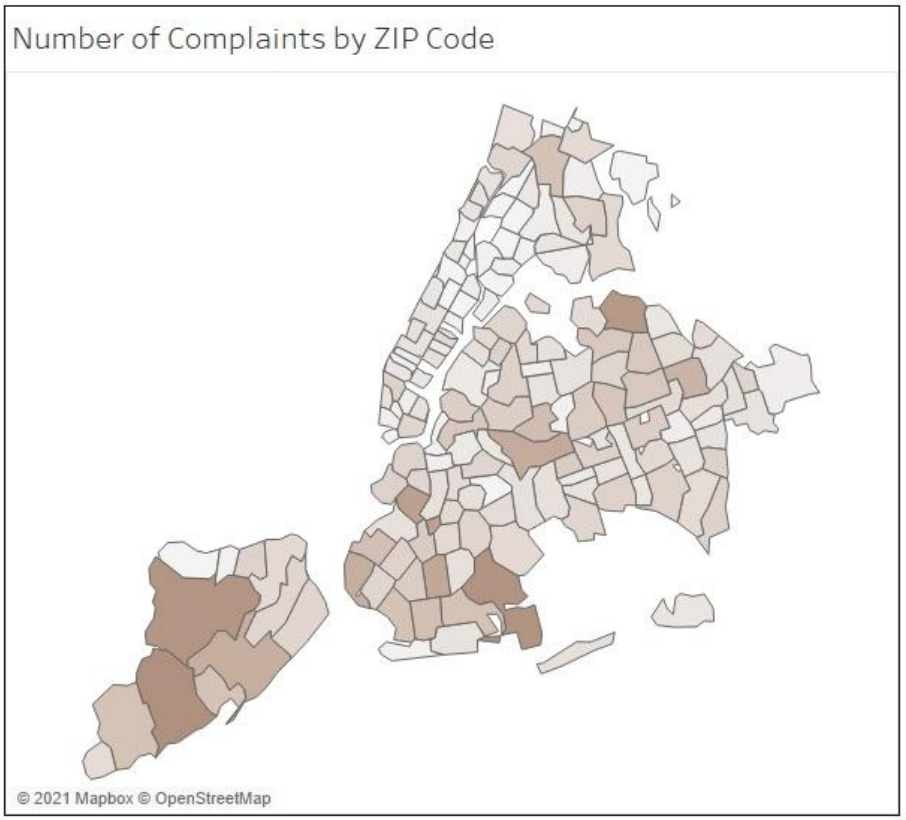
Dashboard & KPI Visualizations

Below is the dashboard created in Tableau to represent our first three KPIs. Descriptions and explanations of each individual visualization can be found on the following pages. The dashboard color scheme is brown and green (the colors of trees) to match its content thematically, and allow users who manage several dashboards to immediately identify this one at a glance. A reddish brown was chosen for maximum contrast against the green bars.



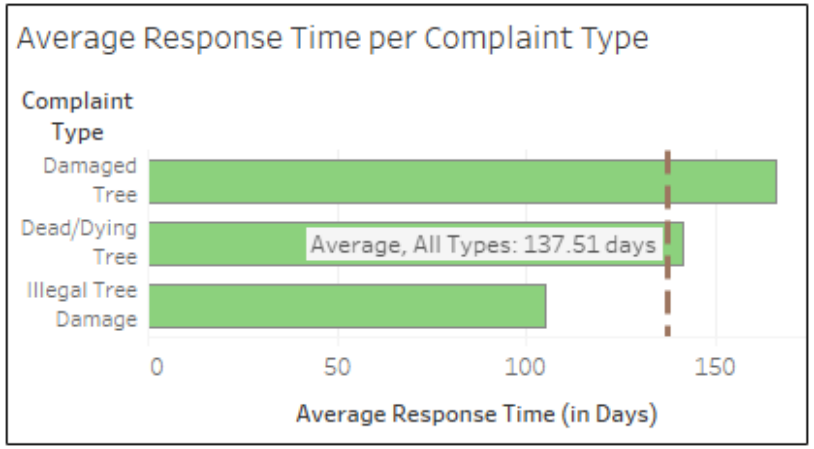






KPI 1

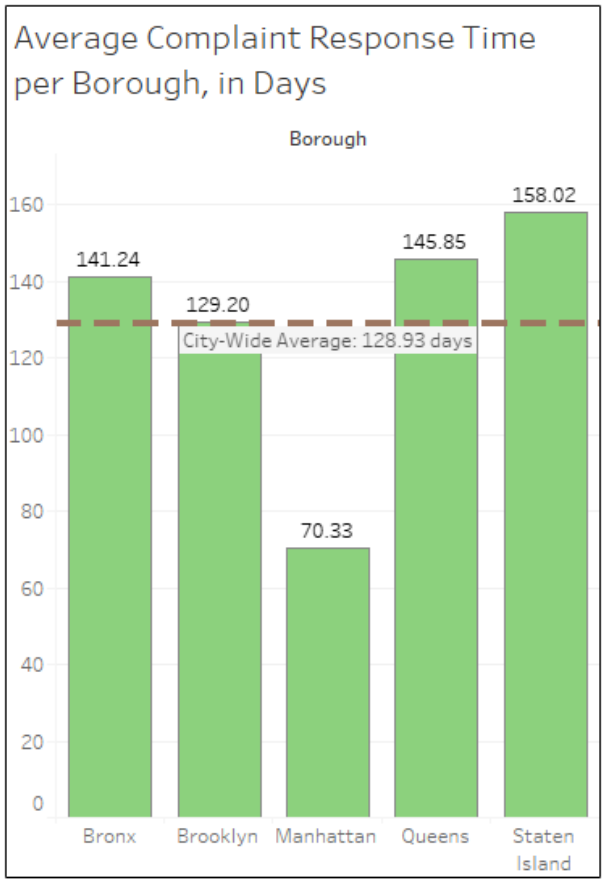
Response time per Damaged Tree, Dead/Dying Tree, or Illegal Tree Damage complaint



This chart tabulates the city's response time to each individual 311 complaint (calculated by subtracting the Created Date value from the Closed Date value), then sums them conditionally by the Complaint_Type value. This allows the city to see whether there are significant differences in their response time based on the type of tree damage reported, and if/when they attempt to remedy that disparity, they will be able to see in real time whether their response times are approaching parity or not.

KPI 2

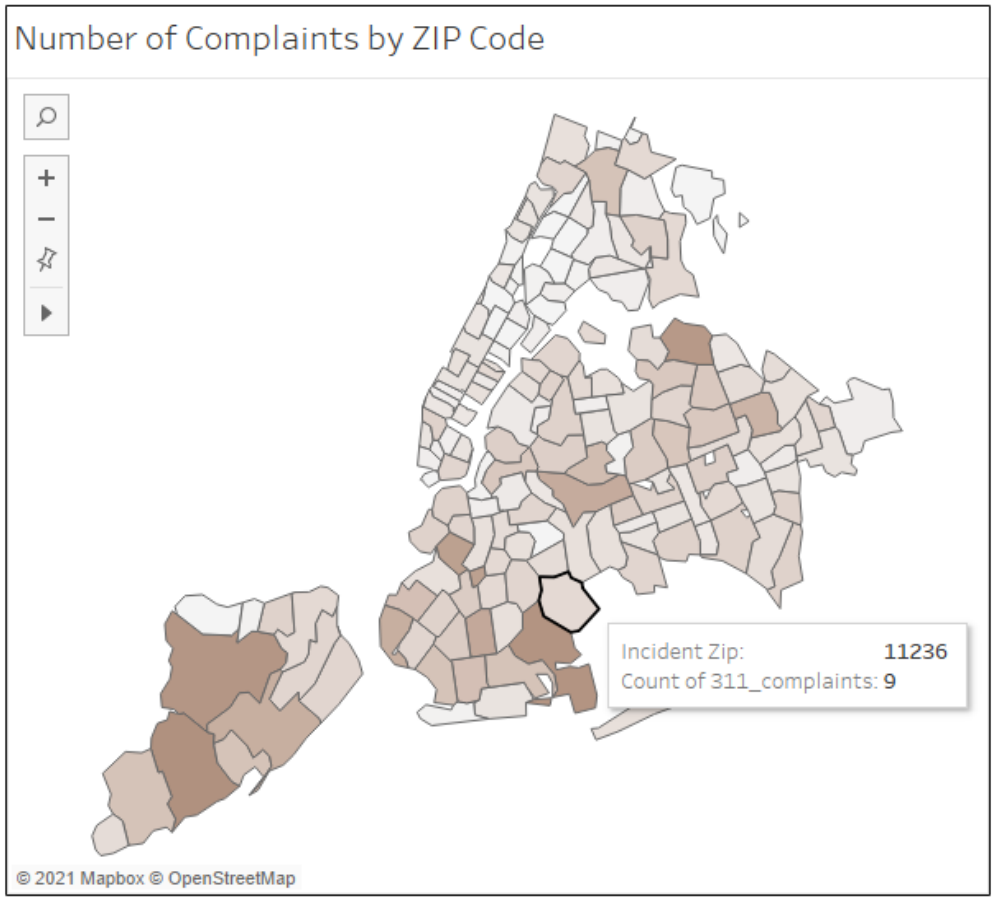
Response time per borough



This chart tabulates the city's response time to each individual 311 complaint, then averages them conditionally by the Borough value. This visualization allows the city to see whether there is inequity in their response times according to the borough where the complaint was recorded. The chart also includes a reference line with the city-wide average as a benchmark for assessing the borough averages.

KPI 2

Response time per borough



This is a "bonus" visualization intended to provide additional context for KPI 2. It is a heatmap that displays the relative frequency of complaints by ZIP code within the city. It is interactive, as demonstrated in the screenshot; users can hover over an individual ZIP code to see the number of complaints recorded. This is an important addition, because it allows for complaint volume to factor into assessment of the response time per borough.

KPI 3

Number of cases opened per month

Number of Complaints per Month

	Mo. Complaint Created											
Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2017					21	10	6	3			1	
2018				1	1	18	34	20	19	11	3	1
2019	4	6	22	30	172	131	100	32	13	14	5	95
2020	96	83	71	86	237			2			1	

This chart tabulates the number of distinct records per month per year, using the Month_Name and Year values. It allows the user to easily observe trends in complaint volume, particularly if any seasonality exists in the data. This is an important metric for planning and budgeting.

Tools

We choose to use **data build tool (dbt)** to perform the extract, transform, and load (ETL) process detailed above, to facilitate analysis of the data stored in our warehouse. Our target database is **Google BigQuery**. ETL coding was performed in Python, and visualizations were created (including the underlying calculations) in Tableau.

Conclusion

Tools used for this project

- Lucidchart: Used to design and visualize the schema of a data warehouse.
- Python: Used to scrape data using API and clean the data by reformatting, removing the missing values, and data profiling. Also used to perform machine learning to cluster the weather types and alert levels.
- Excel: Used to store the weather transactions from 2017 - 2020 for 5 boroughs.
- dbt Cloud: Used to process ETL programming to build dimensions and fact tables.
- Google BigQuery: Used as a data warehouse, storing data from both 311 complaints and weather dataframes, with dimensions and fact tables.
- Tableau: Used for KPI calculations and visualizations.

Group's experience with the project

The most difficult step was cleaning and transforming to maintain consistency between the 311 complaints and weather dataframes. Some values, such as date and address, needed to be split and reformatted into distinct fields (i.e. day, month, year) to avoid inaccurate information. Also, to satisfy the KPI, it was necessary to cluster the weather data to identify "heavy rain" and "heavy snow." Therefore, we chose to reference the method we found on Kaggle to make samples, but we still needed to improve the accuracy with more machine learning methods.

Another step that posed a challenge for us was obtaining the data, and ensuring that our data profiling portrayed appropriate results, so the data sources could be analyzed for trends according to the real-world problems in our KPIs. The 311 data did have some issues in terms of its profile having duplicate rows and missing data, which caused us to download and filter them again. The weather data was also particularly difficult to work with because some of the values were missing, or had spaces in cells, which impaired our queries in BigQuery.

But after extracting the data and cleaning the data, the process got smoother. The easiest step was combining the attributes from each dimension into two fact tables and calculating the values we needed, because the values had been reformatted and were in good shape by that point, so that they could easily be loaded into the fact tables.

We self-learned how to clean the dataset from the real world using Python. It was complex and we never imagined we would use Python rather than SQL to do that. We also learned how to use dbt, which is one of the ETL tools to build a warehouse located in BigQuery.

If we had to do it again and had enough time, we would try to make a warehouse with automated updates from both weather and 311 complaints. This would require us to find open-source weather data instead of using a static copy in Excel files. We would also simplify the location dimension and make it easier to track. The number of transactions stored in our warehouse could be expanded in order to generate better visualizations with more statistically sound conclusions for our KPIs.

Benefits realized by the new system

The new system, i.e. the data warehouse located in BigQuery, can inform various different KPIs based on weather, location, date, complaint types and more. All of the possible metrics can provide numerical context and solutions to improve future planning to address specific problems. For example, as shown on our dashboard, the months of May, June, and July tend to have the highest amount of complaints within the year. We could therefore suggest that the city hire more seasonal employees to address tree complaints during these months. Also, by tracking the weather type, which is clustered by temperature, windspeed, and precipitation, we can see which types of weather the city needs to prepare for better.

Final comments and conclusions

We learned a lot over the course of this project, in completing the various steps needed to get a data warehouse up and running. We learned how to design a star schema and create dimensional models, learned how to profile data sources and clean them, do transformations for our ETL process, and load the data into our data warehouse, while analyzing it using a visualization software. These are all skills that will prove useful in our future careers.

References

1. “BigQuery: Cloud Data Warehouse.” *Google Cloud*, Google, cloud.google.com/bigquery.

2. “BigQuery Configurations.” *dbt Docs*, dbt Labs, docs.getdbt.com/reference/resource-configs/bigquery-configs.

3. Holowczak, Richard. “Data Warehouse Dimensional Modeling.” *Holowczak.com Tutorials*, 3 Oct. 2013, holowczak.com/data-warehouse-dimensional-modeling/.

4. Holowczak, Richard. “Getting Started with NYC OpenData and the Socrata API.” *Holowczak.com Tutorials*, 26 Nov. 2019, holowczak.com/getting-started-with-nyc-opendata-and-the-socrata-api/.

5. Pandas Development Team. “Input/Output.” *Pandas 1.3.5 Documentation*, 12 Dec. 2021, pandas.pydata.org/docs/reference/io.html#json.

6. Rathi, Prakhar. “Weather Data Clustering Using K-Means.” *Kaggle*, 20 May 2019, www.kaggle.com/prakharrathi25/weather-data-clustering-using-k-means.

7. “What Is dbt?” *dbt Docs*, dbt Labs, docs.getdbt.com/docs/introduction.