5310 Final Report

A Proposal Report to NYC 311 Service Requests

Group 7

Xinyang Li, Xinxin Zhang, Nuo Chen

Yuelin Guo, Xuefan Han, Jiaxin Shen

Scenario

New York 311 gives access to non-emergency city services and info about City government programs. It is not only a platform for accepting citizens' complaints, but also a link for citizens to cooperate closely with the government. The platform allows residents to solve their problems efficiently through the platform and improve their attention and enthusiasm for urban construction and government policies. On the other hand, the platform also allows the government to guide the direction of the future government from the macro perspective of the complaints collected.

Moreover, the platform currently has many customer service representatives who have been trained to send requests to the appropriate departments. They are the majority who handle the complaint calls. As the data presented, during Covid-19, 311 received more than 180000 calls per day at peak volume.

Because of the situation that 311 is currently faced with, they would need to have their data extracted and analyzed in order to help the government prioritize policies and help decision makers to detect small changes in the city and respond in time. Extracting and analyzing data can help the government to adjust their current policies and solve the complaints more easily. Moreover, we can also build an automated system that can auto deal with the data and supply the answers directly to the customers. It may help the platform to become more efficient and it can also improve the reaction time of customer service representatives. On the other hand, since the manpower is expansive, we would like to maximally reduce the high manpower. Therefore, a relational database that can help to sort and find information would be a good solution. And it can improve the efficiency of info retrieval and help to automate some of the service inquiries. Analyzing 311 service requests can help the government departments realize their main problem across the city. It can allow the government to decide which part they have to improve first. And it can help the government to reallocate labor forces to solve the complaints and problems. With analyzing the data, the government may reduce the crime/violet rates, improve NYC residents' living happiness, and build up a more positive city image.

Team Contract

Based on the multiple tasks we have; our team has separated the responsibility of different tasks to different group members. The table (Table 1) below is the Team Contract which infers the duty each group member has for different Tasks following with the timeline and completion circumstances.

Sprint	Task	Responsible	Due Date	Complete Y/N
1	Brainstorm/Decide project topic	All	3/26	Y
	Submit Checkpoint 1	Xinxin Zhang	3/26	Y
2	Team Contract	Lead: Jiaxin Shen Support: rest of the team	4/2	Y
	Preliminary draft design		4/2	Y
	Finalize Project Dataset		4/2	Y
	Submit Checkpoint 2	Jiaxin Shen	4/2	Y
3	Database Schema for Review	Lead: Xuefan Han Support: rest of the team	4/9	Y
	Develop SQL codes		4/9	Y
	Submit Checkpoint 3	Xuefan Han	4/9	Y
4	Data Plan for Review	Lead: Yuelin Guo Support: rest of the team	4/16	Y
	Develop R/Python Script		4/16	Y
	Submit Checkpoint 4	Yuelin Guo	4/16	Y
5	Customer Interaction Plan - Analyst	Lead: Nuo Chen	4/23	Y

	Customer Interaction Plan - C-Suite	Support: rest of the team	4/23	Y
	Customer Interaction Plan - tools used		4/23	Y
	Customer Interaction Plan - redundancy/performance		4/23	Y
	Submit Checkpoint 5	Nuo Chen	4/23	Y
6	**Project Report**	**All**	**4/29**	Y
	-Problem statement	Nuo Chen	4/29	Y
	-Proposal	Jiaxin Shen	4/29	Y
	-Team structure and timeline	Xinxin Zhang	4/29	Y
	-Database schema	Xuefan Han	4/29	Y
	-Analytics applications	Yuelin Guo	4/29	Y
	Submit Report	Xinyang Li	4/29	Y
7	Presentation	All	4/30	Y

Data Selection

The Dataset (See Figure 1) we choose is data about New York City 311 hotlines. The dataset contains all 311 Service Requests from 2010 to present. And it is published on NYC Open Data (Here is the link for the dataset: https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9). In the original dataset, it contains data of 311 Service Requests from 2010 to present. It contains 28.1 million rows and 41 columns. The data set contains 32 text columns, 4 numeric columns, 4 date columns and 1 location column. The data set also has a unique key to identify the service request.

Uniq...	Cre...	Close...	Agen...	Agen...	Com...	Desc...	Loca...	Incid...	Incid...	Stree...
54018714	04/27/20...		NYPD	New York...	Noise - St...	Loud Mus...	Street/Sid...	10065	21 EAST 6...	EAST 66 S...
54016610	04/27/20...		NYPD	New York...	Non-Eme...	Trespassi...	Residenti...	11235	3052 BRI...	BRIGHTO...
54017629	04/27/20...		NYPD	New York...	Noise - R...	Loud Mus...	Residenti...	11226	390 PARK...	PARKSIDE...
54015553	04/27/20...		NYPD	New York...	Noise - V...	Car/Truck...	Street/Sid...	11105	42-01 19 ...	19 AVENUE
54012351	04/27/20...		NYPD	New York...	Noise - R...	Loud Mus...	Residenti...	10474	233 COST...	COSTER S...
54014479	04/27/20...		NYPD	New York...	Illegal Par...	Blocked S...	Street/Sid...	10452	100 WEST...	WEST 163...
54011324	04/27/20...		NYPD	New York...	Noise - V...	Car/Truck...	Street/Sid...	10457	1853 ANT...	ANTHON...
54011320	04/27/20...		NYPD	New York...	Noise - St...	Loud Talk...	Street/Sid...	11218	2836 FOR...	FORT HA...
54012355	04/27/20...		NYPD	New York...	Noise - St...	Loud Mus...	Street/Sid...	11105	40-19 20 ...	20 AVENUE
54011300	04/27/20...		NYPD	New York...	Illegal Par...	Double P...	Street/Sid...	10458	2702 BAI...	BAINBRID...
54017636	04/27/20...		NYPD	New York...	Noise - St...	Loud Mus...	Street/Sid...	10473	319 TAYL...	TAYLOR A...
54013405	04/27/20...		NYPD	New York...	Noise - R...	Loud Mus...	Residenti...	11358	46-37 188...	188 STREET
54015526	04/27/20...		NYPD	New York...	Illegal Par...	Double P...	Street/Sid...	10452	96 WEST ...	WEST 163...

(Figure 1)

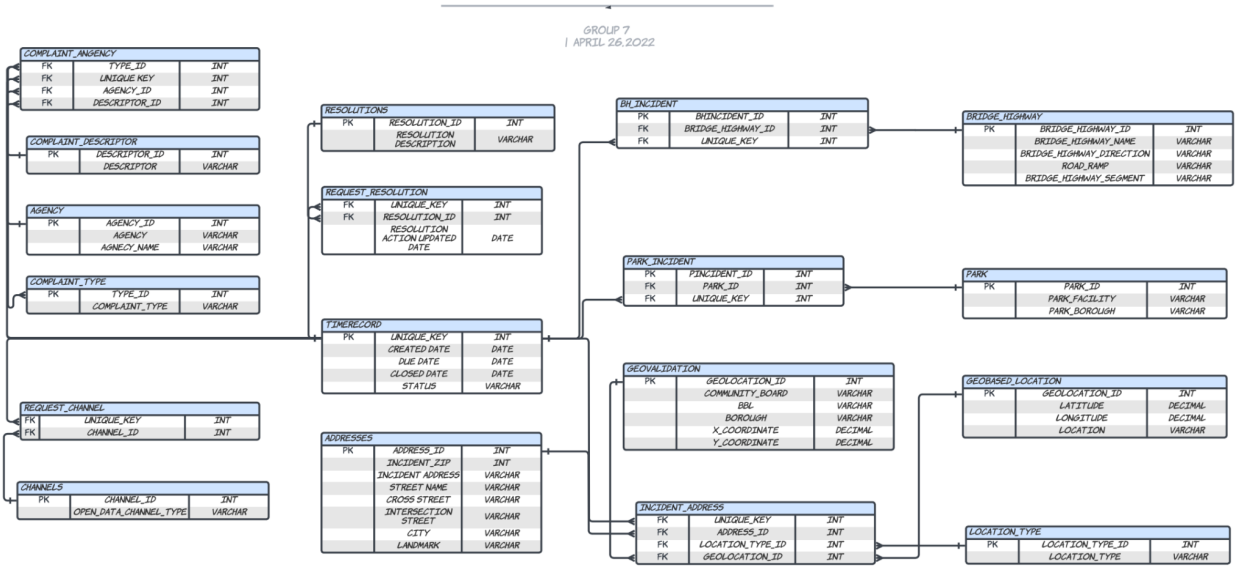
The database contains all the necessary information of every 311-complaint phone call. The information includes a Unique key, agency, complaint type, location, create date, etc. The dataset can be used to analyze the problem of current complaints that exist in New York City. And the information can be used to take a glance of the Areas that have the highest volume of complaints. And analyzing the dataset can be used to help the government to improve the current policy and take action to improve citizens' life qualities.

For the project purposes, we plan to filter the dataset to extract the latest information and cut down the number of rows. We decide to split the current data into several tables that each table contains specific information. For instance, we would use different tables to store the caller information, Customer service representative information, address information, vehicle information, parking information, and etc. We will also add ids into different tables to specialize the different information. And we will clean the dataset first and restructure it for a better read. And with the new unique id in the different tables, it would be easier for us to connect different tables and can make it more convenient to sort and find information from the tables.

Normalization Plan

After reviewing the whole dataset, we are planning to normalize the data and create tables for it. It requires the team to add some unique ids to identify different variables that appear in the dataset. For instance, we have to add a Channel_ID variable to identify the channel that the

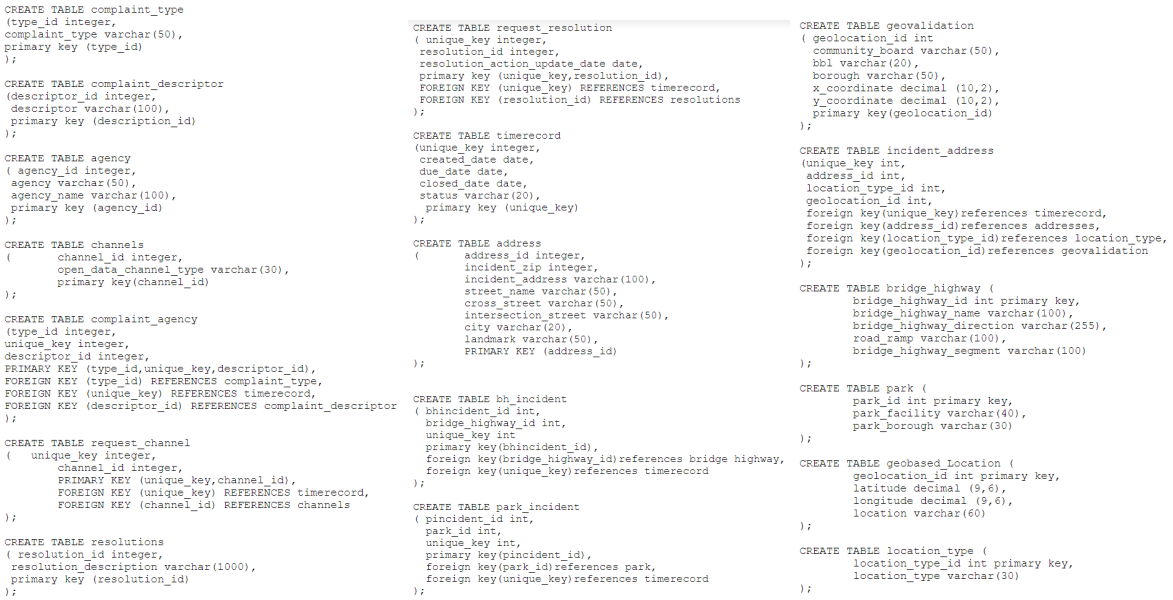
phone call is about. Also, in order to better collect all the tables and the variables within them, the team decided to add bridge tables as the bridge to connect different tables. With a whole consideration of the whole 41 columns, the team conducted a normalization process that split the dataset into 18 different tables (See Figure 2, the ERD). And the team has added 13 unique ids that can be used to identify the type, agency, descriptor, channel, resolution, address, bridge and highway incident, bridge and highway, location type, etc. After creating new unique ids for the different variables, the team started to implement tables into the database.



(Figure 2 ERD)

https://lucid.app/lucidchart/58b520fd-55df-468f-9c7a-a8b137ebee48/edit?invitationId=inv_7f9b4cba-4cdd-451f-a34e-33ee0ca3106f

Because the team needed each table to contain only values that can be identified and eliminate data duplication and avoid data anomalies, we have split the table in the following orders. First we created tables that contained a single type of variables. We have created complaint_type, agency, complaint_descriptor, resolutions, request, timerecord, channels, addresses, Bridge highway, park, location type, and geovalidation tables. We are using SQL to create and normalize these tables (See Figure 3). These tables contain data with no duplication and no anomalies. And then we created the bridge table that used to connect the previous tables we created. The bridge tables are Complaint_angency table, request_channel table, request_resuolution table, bridge and highway incident table , incident_address table.



```
CREATE TABLE complaint_type
(type_id integer,
complaint_type varchar(50),
primary key (type_id)
);

CREATE TABLE complaint_descriptor
(descriptor_id integer,
 descriptor varchar(100),
 primary key (description_id)
);

CREATE TABLE agency
( agency_id integer,
 agency varchar(50),
 agency_name varchar(100),
 primary key (agency_id)
);

CREATE TABLE channels
(       channel_id integer,
        open_data_channel_type varchar(30),
        primary key(channel_id)
);

CREATE TABLE complaint_agency
(type_id integer,
unique_key integer,
descriptor_id integer,
PRIMARY KEY (type_id,unique_key,descriptor_id),
FOREIGN KEY (type_id) REFERENCES complaint_type,
FOREIGN KEY (unique_key) REFERENCES timerecord,
FOREIGN KEY (descriptor_id) REFERENCES complaint_descriptor
);

CREATE TABLE request_channel
(   unique_key integer,
        channel_id integer,
        PRIMARY KEY (unique_key,channel_id),
        FOREIGN KEY (unique_key) REFERENCES timerecord,
        FOREIGN KEY (channel_id) REFERENCES channels
);

CREATE TABLE resolutions
( resolution_id integer,
 resolution_description varchar(1000),
 primary key (resolution_id)
);
```

```
CREATE TABLE request_resolution
( unique_key integer,
 resolution_id integer,
 resolution_action_update_date date,
 primary key (unique_key,resolution_id),
 FOREIGN KEY (unique_key) REFERENCES timerecord,
 FOREIGN KEY (resolution_id) REFERENCES resolutions
);

CREATE TABLE timerecord
(unique_key integer,
 created_date date,
 due_date date,
 closed_date date,
 status varchar(20),
  primary key (unique_key)
);

CREATE TABLE address
(       address_id integer,
        incident_zip integer,
        incident_address varchar(100),
        street_name varchar(50),
        cross_street varchar(50),
        intersection_street varchar(50),
        city varchar(20),
        landmark varchar(50),
        PRIMARY KEY (address_id)
);


CREATE TABLE bh_incident
( bhincident_id int,
  bridge_highway_id int,
  unique_key int
  primary key(bhincident_id),
  foreign key(bridge_highway_id)references bridge highway,
  foreign key(unique_key)references timerecord
);

CREATE TABLE park_incident
( pincident_id int,
  park_id int,
  unique_key int,
  primary key(pincident_id),
  foreign key(park_id)references park,
  foreign key(unique_key)references timerecord
);
```

```
CREATE TABLE geovalidation
( geolocation_id int
  community_board varchar(50),
  bbl varchar(20),
  borough varchar(50),
  x_coordinate decimal (10,2),
  y_coordinate decimal (10,2),
  primary key(geolocation_id)
);

CREATE TABLE incident_address
(unique_key int,
 address_id int,
 location_type_id int,
 geolocation_id int,
 foreign key(unique_key)references timerecord,
 foreign key(address_id)references addresses,
 foreign key(location_type_id)references location_type,
 foreign key(geolocation_id)references geovalidation
);

CREATE TABLE bridge_highway (
        bridge_highway_id int primary key,
        bridge_highway_name varchar(100),
        bridge_highway_direction varchar(255),
        road_ramp varchar(100),
        bridge_highway_segment varchar(100)
);

CREATE TABLE park (
        park_id int primary key,
        park_facility varchar(40),
        park_borough varchar(30)
);

CREATE TABLE geobased_Location (
        geolocation_id int primary key,
        latitude decimal (9,6),
        longitude decimal (9,6),
        location varchar(60)
);

CREATE TABLE location_type (
        location_type_id int primary key,
        location_type varchar(30)
);
```

(Figure 3)

ETL Process

The team is using Google Colab as the place to store and test the code. Here is the link for the Google Colab.
https://drive.google.com/file/d/1P-EdQbtCbC2vWIUhoBdHx3tjv6hhxyiw/view?usp=sharing
For the project, our team first built an environment of PostgreSQL with Google Colab and connected to the PostgreSQL system by using the sqlalchemy package in python. After the connection with the PostgreSQL database and our file, we used SQL to create all the tables that we have created for normalization and store the table into the database. In order to access the data from 311 data source in NYC Open Data, we downloaded the csv file from NYC Open Data and then stored the whole csv file into a Pandas data frame. Using Pandas Data frame can help us to store the data before we insert it into the tables we created in the PostgreSQL database.

For the regular tables, which contains resolutions table, time record table, addresses table, agency table, channels table, bridge highway table, park table, Geobased_location table, location type table, complaint type table and complaint descriptor table, we have firstly dropped all the duplicate values within the Pandas Data Frame columns. And then, we have added unique IDs into the tables that we created within the Pandas Data Frame. The id has been added based on the length of the data we have filtered and chosen from the original database. After creating the tables, we used the ‘append’ method to append all the data we have used in different tables into the PostgreSQL Database (See Figure 4).

```
1.Resolutions table

[ ]  #drop the duplicate value in the dataset and then create a table for detailed resolution
     Resolutions_data = pd.DataFrame(data['Resolution Description'].unique(), columns=['resolution_description'])
     Resolutions_data.insert(0, 'resolution_id', range(1, 1 + len(Resolutions_data)))

[ ]  Resolutions_data.to_sql(name='resolutions', con=engine, if_exists='append', index=False)
```

(Figure 4)

After creating all the individual tables and inserting the data into the corresponding table in the PostgreSQL database, we started to Merge values into the bridge table(See Figure 5). We used the merge function in the Pandas package and combined the data in the previous data frame to create a new table that only contains the unique ids that can be used to identify the variables. What needs to be noticed that is we have also added a unique key column that the original database has into most of the new bridge tables for better identification. And then we insert the data into the PostgreSQL database bridge tables.

```
#merge four tables
bridge2 = pd.merge(data,agency_data,
                        how='left',
                        left_on=['Agency'],
                        right_on = ['agency'])

bridge3 = pd.merge(bridge2,complaints_data,
                   how='left',
                   left_on=['Complaint Type'],
                   right_on=['complaint_type'])


bridge4 = pd.merge(bridge3,complaint_detail,
                   how='left',
                   left_on=['Descriptor'],
                   right_on=['descriptor'])

#create subset
data_complaint_angency = bridge4[['Unique Key','agency_id','type_id','descriptor_id']]
data_complaint_angency.columns=['unique_key','agency_id','type_id','descriptor_id']
```

(Figure 5)

Therefore, we have created a complete environment and tables that contain all the data we gathered from the original dataset.

10 analytical procedures

Based on the data we have gathered and stored in the PostgreSQL database, Our team has come up with 10 analytical problems and 4 strategic questions. And for the 10 analytical questions, we have separated the questions into 4 different sections: Number of complaints, Agency, Complaint Types and Resolutions.

The Analytical Questions are listed below:

Number of complaints

1. What are the top 3 areas that have the most complaints in New York?
2. Which bridge has the most complaints?
3. Which boroughs have the most call incidents?

Agency

4. What is the daily average number of 311 calls received by each agency?
5. Which agency has the most incidents?

Complaint types

6. Which complaint type has the most calls in each region?
7. What are the most complaints reported in Manhattan?
8. What is the accumulated sum for each type of complaint by time series?

Resolutions

9. What's the average resolution speed for each complaint type?
10. Which agency solves the problem fastest?

To answer all these questions properly, our team has used SQL to conduct a search in the database and analyze the results based on the information we get by SQL methods. To have a clearer view on different questions, the team would discuss each question in this part.

1. What are the top 3 areas that have the most complaints in New York?

(Result and SQL implement See Figure 6)

For this question, we want to find the areas that have the most complaints. We need to consider the relation between complaints and the areas. And the result has shown that Brooklyn, New York and the Bronx are the areas that have the most complaints in New York.

```
select count(tt.unique_key) as total, city
from timerecord tt
left join incident_address ia
on tt.unique_key = ia.unique_key
left join address add
on ia.address_id = add.address_id
where city is not null
group by city
order by count(tt.unique_key) desc
limit(3);
```

	total bigint	city character varying (20)
1	10383	BROOKLYN
2	5995	NEW YORK
3	5536	BRONX

(Figure 6)

2. Which bridge has the most complaints? (Result and SQL implement See Figure 7)

In this question, the team aims to find the bridge location that has the greatest number of complaints. It may infer that the bridge has the most serious problem.

```
select count(tr.unique_key), bridge_highway_name
from timerecord tr
left join bh_incident bi
on tr.unique_key = bi.unique_key
left join bridge_highway bh
on bi.bridge_highway_id = bh.bridge_highway_id
where bridge_highway_name is not null
group by bridge_highway_name
order by count(tr.unique_key) desc
limit(5);
```

	count bigint	bridge_highway_name character varying (100)
1	20	F
2	17	Q
3	16	Long Island Expwy
4	15	R
5	14	E

(Figure 7)

3. Which boroughs have the most call incidents?

(Result and SQL implement See Figure 8)

In this question, the team aims to find the boroughs that have the highest number of incident input phone calls in the 311 hotlines.

```
select borough,count(tr.unique_key) as total
from timerecord tr
left join incident_address ia
on tr.unique_key = ia.unique_key
left join geovalidation geo
on ia.geolocation_id = geo.geolocation_id
where borough is not null
group by borough
order by count(tr.unique_key) desc
limit(3);
```

	borough character varying (50)	total bigint
1	BROOKLYN	10298
2	QUEENS	8303
3	MANHATTAN	6442

(Figure 8)

4. What is the daily number of 311 calls received by each agency?

(Result and SQL implement See Figure 9)

In the question, the team aims to find the everyday volume of phone calls that each agency is taking care of.

```
with table_2 as (select tt.unique_key, agency, agency_name, complaint_type, created_date
from timerecord tt
left join
complaint_agency ca
on tt.unique_key = ca.unique_key
left join agency aa
on ca.agency_id = aa.agency_id
left join complaint_type ct
on ca.type_id = ct.type_id
left join incident_address ia
on tt.unique_key = ia.unique_key)
select agency, agency_name, complaint_type,
count(unique_key)/nullif((max(created_date)-min(created_date)),0)
from table_2
group by agency, agency_name,complaint_type
order by count(unique_key) desc ;
```

agency_name character varying (100)	complaint_type character varying (50)	?column? bigint
New York City Police Department	Illegal Parking	1346
New York City Police Department	Noise - Residential	995
Department of Housing Preservation and Development	HEAT/HOT WATER	665
New York City Police Department	Blocked Driveway	502
Department of Transportation	Street Condition	366
New York City Police Department	Noise - Street/Sidewalk	305
Department of Housing Preservation and Development	UNSANITARY CONDITION	284
New York City Police Department	Abandoned Vehicle	213

(Figure 9)

5. Which agency has the most incidents? (Result and SQL implement See Figure 10)
In the Question, the team aims to identify the agency that needs to deal with the highest number of incidents.

```
select distinct aa.agency, aa.agency_name, count(created_date) as count
from agency aa
join complaint_agency ca on ca.agency_id = aa.agency_id
join timerecord tt on ca.unique_key = tt.unique_key
group by agency,agency_name
order by count desc;
```

	agency character varying (50)	agency_name character varying (100)	count bigint
1	NYPD	New York City Police Department	24270
2	HPD	Department of Housing Preservation and Development	11134
3	DOT	Department of Transportation	5625
4	DSNY	Department of Sanitation	4415
5	DEP	Department of Environmental Protection	3067
6	DPR	Department of Parks and Recreation	1852

(Figure 10)

6. Which complaint type has the most calls in each region?
(Result and SQL implement See Figure 11)

In this question, the team aims to find out the complaint type that most citizens are caring about.

```
create view region_complaints as
select city,complaint_type, count(*) over(partition by complaint_type) as total
from timerecord tt
left join complaint_agency ca
on tt.unique_key = ca.unique_key
left join complaint_type ct
on ca.type_id = ct.type_id
left join incident_address ia
on tt.unique_key = ia.unique_key
left join address add on
ia.address_id = add.address_id
where city is not null
group by city, complaint_type
order by city, total desc;

select city, complaint_type, total
from (
  select * ,rank() over(partition by city order by total desc) as rk
  from region_complaints
) x
where rk = 1
```

	city character varying (20)	complaint_type character varying (50)	total bigint
1	AMITYVILLE	Lead	28
2	ARVERNE	Illegal Parking	46
3	ARVERNE	Blocked Driveway	46
4	ARVERNE	HEAT/HOT WATER	46
5	ASTORIA	Blocked Driveway	46
6	ASTORIA	HEAT/HOT WATER	46
7	ASTORIA	Illegal Parking	46
8	BAYSIDE	HEAT/HOT WATER	46

(Figure 11)

7. What are the most complaints reported in Manhattan?
(Result and SQL implement See Figure 12)
In this question, the team aims to find the specific complaint type that has been mostly reported in a specific area.

```
select complaint_type, count(*) as total from timerecord tt
left join incident_address ia
on tt.unique_key = ia.unique_key
left join complaint_agency ca
on tt.unique_key = ca.unique_key
left join complaint_type ct
on ca.type_id = ct.type_id
left join address add
on ia.address_id = add.address_id
where city = 'MANHATTAN'
group by complaint_type
order by total desc;
```

	complaint_type character varying (50)	total bigint
1	Illegal Parking	57
2	Blocked Driveway	30
3	Noise - Residential	26
4	HEAT/HOT WATER	20
5	Street Condition	15
6	Abandoned Vehicle	13
7	Graffiti	11
8	UNSANITARY CONDITION	11

(Figure 12)

8. What is the accumulated sum for each type of complaint by time series?
(Result and SQL implement See Figure 13)
In this question, the team aims to find the total amount of complaints within a period.

```
select complaint_type,created_date, count(tr.unique_key)
over(partition by complaint_type order by tr.created_date desc
rows between current row and unbounded following) as
accumulate_count
from timerecord tr
left join complaint_agency ca
on tr.unique_key = ca.unique_key
left join complaint_type ct
on ca.type_id = ct.type_id
group by complaint_type,created_date,tr.unique_key;
```

	complaint_type character varying (50)	created_date date	accumulate_count bigint
1	Abandoned Bike	2022-04-10	31
2	Abandoned Bike	2022-04-10	30
3	Abandoned Bike	2022-04-10	29
4	Abandoned Bike	2022-04-10	28
5	Abandoned Bike	2022-04-10	27
6	Abandoned Bike	2022-04-10	26
7	Abandoned Bike	2022-04-10	25
8	Abandoned Bike	2022-04-09	24
9	Abandoned Bike	2022-04-09	23

(Figure 13)

9. What's the average resolution speed for each complaint type?

(Result and SQL implement See Figure 14)

In this question, the team aims to find the speed of question solving for each complaint type.

```
select ct.complaint_type, avg(t.closed_date-t.created_date) as
resolved_in_days
from timerecord t
left join complaint_agency ca
on t.unique_key=ca.unique_key
left join complaint_type ct
on ca.type_id=ct.type_id
WHERE t.closed_date is not NULL
group by ct.complaint_type
order by resolved_in_days desc;
```

	complaint_type character varying (50)	resolved_in_days numeric
1	Highway Sign - Damaged	7.0000000000000000
2	Ferry Complaint	5.0000000000000000
3	Illegal Posting	4.4666666666666667
4	LinkNYC	4.2500000000000000
5	Recycling Basket Complaint	4.0000000000000000
6	Sidewalk Condition	3.8901098901098901
7	Litter Basket Complaint	3.7142857142857143
8	APPLIANCE	3.3698630136986301

(Figure 14)

10. Which agency solves the problem fastest?

(Result and SQL implement See Figure 15)

In this question, the team aims to find the agency with the fastest speed of problem solving.

```
select a.agency,avg(t.closed_date-t.created_date) as
resolved_in_days
from timerecord t
left join complaint_agency ca
on t.unique_key=ca.unique_key
left join agency a
on ca.agency_id=a.agency_id
WHERE t.closed_date is not NULL
group by a.agency
order by resolved_in_days;
```

	agency character varying (50)	resolved_in_days numeric
1	NYPD	0.11031896480672545949
2	DHS	0.61742983751846381093
3	DOB	0.66448152562574493445
4	DOHMH	0.83809523809523809524
5	DEP	0.89653489507076622743
6	DOT	0.96223888591322978040
7	TLC	1.0718954248366013
8	DPR	1.1116751269035533

(Figure 15)

Analyzing Plan

As the analytical problems listed above, we are using direct query to get the answer of the previous questions. And the analysts can directly access the queries to perform the results.
For customers of our project (C-level), we created dashboards in Metabse for them to review the results from the analytical questions. On top of the questions illustrated above, we include insightful graphs that answer additional questions.

1. What are the most frequent complaint types?
2. What are the most time-consuming requests?
3. What is the most appropriate agency to handle each request type?

The dashboards help managers to fine tune business decisions based on multiple data points. For example, from the complaints-related queries, C-level will know the areas where need to reallocate more staff and attention and thus have a better performance on the controlling process of the NY city image. Additionally, from the agency-related queries, C-level can also hire more staff for the agency with most incidents and lay off some agency staff whose workload is not that heavy.

Tools Used

In the project, we use Python and SQL as programming languages, PostgreSQL and PgAdmin as database technologies, Lucidchart and Metabase as visualization tools.
Specifically, we first pull the original dataset and divide it into 3NF tables. We use Lucidchart to draw the ER diagram to present clear relationships between the 3NF tables. Then we use Python to wrangle and clean the original dataset, then divide and merge them into subsets according to our tables. Later we use SQL to write queries used to create tables and constraints. After that, we write Python codes to interface PostgreSQL, perform the SQL queries, and impute data subsets into the tables we created. The resulting database and relations can be directly accessed via PgAdmin. We then write analytical SQL queries which can generate business insights. Finally, we use Metabase to create dashboards and visualizations to deliver meaningful information to C-level executives and other non-technical audiences. Dashboards are easy to comprehend and allow the management to consolidate information across various areas.

Redundancy and Performance

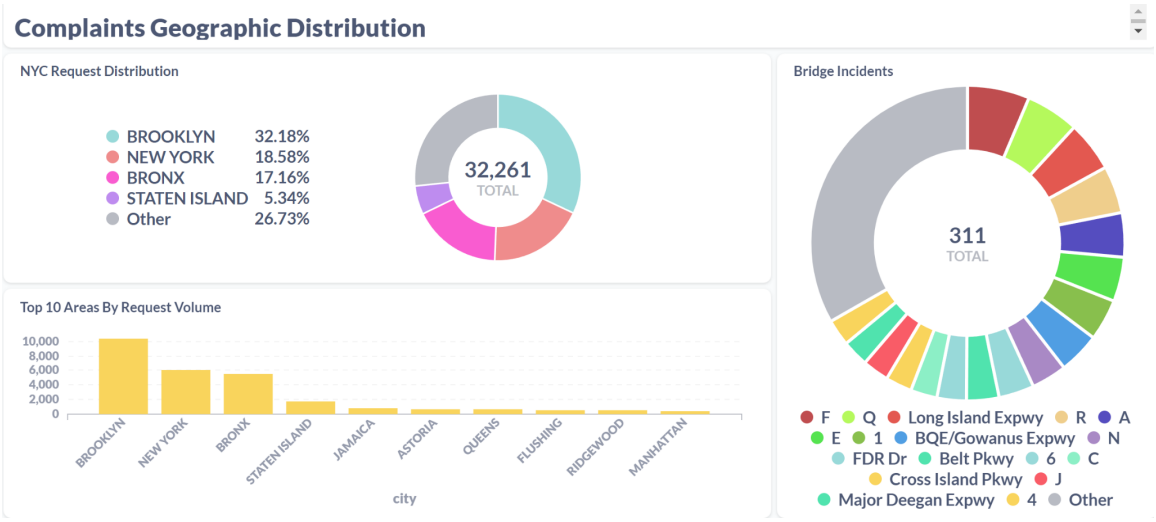
Redundancy problems arise when data is not well-normalized, resulting in unnecessary recurrences of the same piece of data. Specifically, in our original dataset, columns such as Agency, Agency Name, Open Data Channel Type, and Park Borough all have a relatively little number of categories.

However, each category appears multiple times with each unique row. To solve this problem, we create new tables and unique identifiers to store this data. On top of that, we set up primary keys and foreign keys to establish table linkage in order to connect each piece of information. Furthermore, there are some columns that are quite sparse or contain the same piece of information for every row. Since they cannot yield any meaningful insights, we also tease them. Columns that simply merge other columns are deleted as well, for example, the Location column stores spatial data that combines data in Latitude and Longitude columns.

As for performance considerations, we first improve performance by minimizing redundancy. In the analytical procedures, we will consider using stored procedures, CTE, view etc. to create intermediate reusable results. We also might want to invest in cloud platforms for data storage.

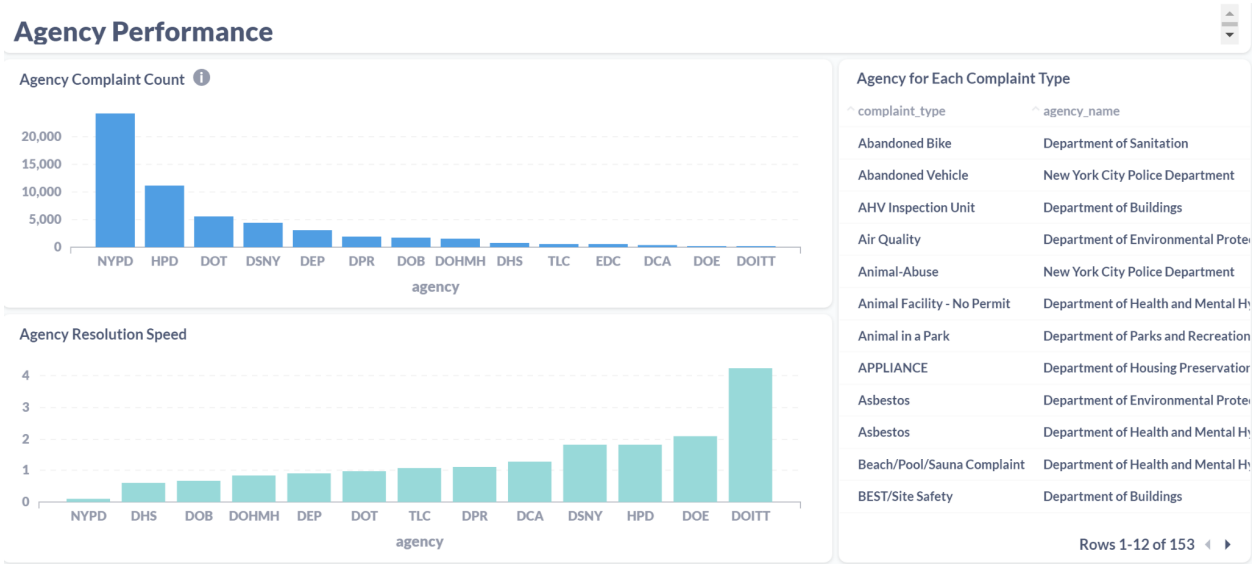
Dashboards

The first section of the dashboard is the geographic distribution of complaints. Management can quickly gain insights into the most important aspects of the data and then integrate the insight into the decision-making process. For instance, the areas with the most frequent calls require more attention and resources.



(Figure 16)

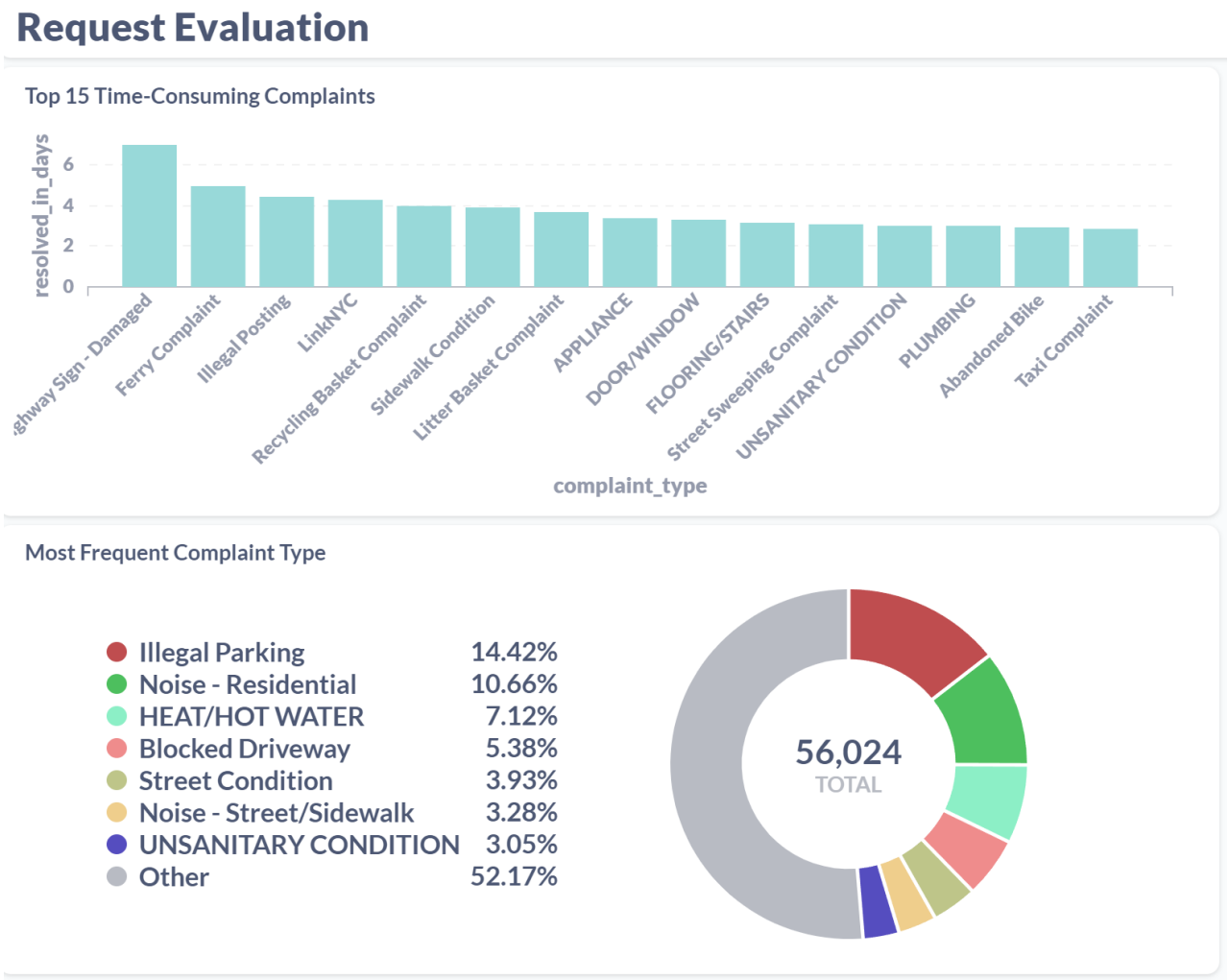
The second segment is agency assessment. The below charts evaluate the efficiency of major agencies. The table can help 311 agents to direct the complaint to the most appropriate agency. Currently, most of the 311 calls are handled by human representatives. With the help of data analytics and machine learning, this call redirecting process can be automated by the system. This will not only increase the accuracy of the call routing but also free more human resources to focus on other tasks.



(Figure 17)

The last section is request evaluation. Decision-makers should investigate the potential cause of the delays for the most time-consuming complaints. Moreover, the proportion of request types

serve as a baseline on which department might need more capacity in order to provide fast and quality service.



(Figure 18)

Conclusion

To conclude, the team has collected the data from the 311 hotline daily reports. It allows our team to conduct analysis on the current situation of the 311 hotline and have an eye on the potential risks that New York City is facing. The Analysis result of the project gives the government clear and vivid information on the complaints that New York Citizens have about the city and the problems the citizens are facing. Therefore, it may lead the government to adjust their current policy to benefit the whole state better. Moreover, the analysis may help the government department to have a vivid picture of the volume of complaints regarding the Agency departments. It may help the city to properly assign labor force and to better assist New York. It allows the departments to pre-assign employees to face a high volume of complaints in the future circumstances.

Using RDMS, it allows the team to create an efficient database for government agencies to store the 311 hotline information. And it allows the clients to interact with the data and find the result they need. One of the most significant benefits of using the RDMS is the security. As the 311 hotline data mostly contains privacy information of each caller. They may not want the information to be easily accessed by others. Therefore, the RDMS can secure the data and make it as easy as possible for the client to do their own query search in the database. On the other hand, the RDMS allows clients to edit and import new data into the database anytime.
Through our analysis, we are able to present the government with the current situation and performance of the Agency department. It can help the government to adjust their policy and regulate the agency. Moreover, it may raise their awareness of the complaints and current problems existing in New York.

Based on our analysis, the team comes with the following insights:

1. Brooklyn Area has the highest volume of complaints, it may need the government to focus on the regulation of this Area.
2. Noise complaints and illegal parking complaints are the two most recorded complaints.
3. Decision-makers should investigate the potential cause of the delays for the most time-consuming complaints.

The PostgreSQL database and final analysis our team built allows the government to have a new method to gather a more vivid view of the current situation of the 311 hotline data. And can have a better view on the potential problems existing in New York. Overall, the insights can help the government to make proper adjustments at the proper time. And it can be used to improve the living quality of the New York citizens.