PROJECT REPORT ON

**“DOB Job Application Filings Database”**

Submitted by

Kexin Lian

Dongquan Qiu

Wenqian Liu

Junyao Xie

Chunye Xie

Prepared for

Apan 5310

SQL & Relational Databases

Instructor Nickolaos Machairas

Master’s in Applied Analytics

Department of School of professional Studies

Columbia University

August 10, 2020

**Problem Statement**

For this project, we assume that we’re a data consulting team hired by the NYC Department of Buildings (DOB).

Every year, there are numerous building construction going on in New York City, the number of residential and business buildings raised DOB receives a large amount of construction job applications, but only a small amount of them can be approved and put into action. The data comes from different sources and platforms, and is manually input into the system, which results in many null values, errors and typos in the information stored. We are hired by the client to build a database to make the data storage more accurate and efficient, and to generate some insights of building construction progress in NYC.

Our data resources include:

1. DOB job application filings: <https://data.cityofnewyork.us/Housing-Development/DOB-Job-Application-Filings/ic3t-wcy2?fr%20om=groupmessage&isappinstalled=0>
2. DOB Now: build-job application filings: <https://data.cityofnewyork.us/Housing-Development/DOB-NOW-Build-Job-Application-Filings/w9ak-ipjd>
3. DOB permit insurance: <https://data.cityofnewyork.us/Housing-Development/DOB-Permit-Issuance/ipu4-2q9a>
4. Labor Statistics for the New York City region: <https://www.labor.ny.gov/stats/nyc/>

**Proposal**

Our client has a lot of historical building construction job application data on hand, as well as new data that would come in which need to be stored on a timely basis. In order to help the client better manage the data and conduct analysis, we will develop a relational schema and load old data for our client. To fit the client’s situation, we will develop a comprehensive schema that can allow them to store the data, extract the data, and update the database on an ad-hoc basis when new data comes in. The database will contain validation and constraints to ensure correct data types and data values are input into the system. At the end, we will build an interactive dashboard for visualization purposes. After we finish our analysis, they can also see the trends and gain valuable insights in the future.

After building the database and visualization dashboard, we will show the client how to use the database to generate insights that will benefit the department as well as the city. The building construction application database can provide valuable insights of city development and building safety. First, we can understand the types, locations, dates, scales and facilities of the buildings that were on the planning. By investigating the data, we can see the trend of how New York City evolved over years and depict a picture of how the city will look like in the near future. For the department, it’s necessary to monitor and manage buildings in the city to develop better plans for city construction. This information can also be used as a tool to attract investors, residents, employers and promote the city economy.

**Team structure and Timeline**

**Team structure**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **roles** |
| Junyao Xie | jx2409@columbia.edu | General member |
| Dongquan Qiu | dq2148@columbia.edu | General member |
| Chunye Xie | cx2245@columbia.edu | General member |
| Wenqian Liu | wl2749@columbia.edu | General member |
| Kexin Lian | kl3191@columbia.edu | General member |

**Team contract**

PROJECT VISION

Our group’s grade expectation on this project is A and our sprint goal is A+.

ELEMENTS OF EFFECTIVE TEAMWORK

**Communication**

Due to the geographic diversity of our group members and for effective teamwork to occur, an environment of speaking free, open and expressing appropriate ideas and feelings is needed. Each member is supposed to actively listen to other members, respond to other members in time, and provide effective non-judgmental feedback. We will use WeChat and Slack as tools for communication.

**Participation**

The success of our team relies on team members’ participation and contributions. Our group will try the best to assign equal assignments and tasks to each member, respect and help each other when anyone needs. Team members need to contribute fully to the best of their ability. Members need to take initiative in participating in the group tasks, especially in areas where they may have strengths. Those with greater ability may also need to help those who may be struggling by guiding, coaching or critiquing. To make teams work well, without informing the group in advance, all members need to attend our weekly meetings, express their ideas and thoughts, actively communicate with others on platforms such as Slack, complete their assigned tasks, and revise their work after receiving feedback.

**Progress And Assessment**

Members of an effective team will contribute to an attitude of action and momentum. Often, progress is a good indicator of how well the team is working together. Regular assessment is necessary for a team to ensure it is continuing to work well together. An effective team is not afraid to make changes in how it is organized or in it’s procedures so that improvement in achieving the goal/objective occurs.

CONTRACT AGREEMENT

This is an official contract. Once you have signed it you are accountable.

Name: Junyao Xie Signature: Junyao Xie Date: July 7, 2020

Name: Chunye Xie Signature: Chunye Xie Date: July 7, 2020

Name: Kexin Lian Signature: Kexin Lian Date: July 7, 2020

Name: Dongquan Qiu Signature: Dongquan Qiu Date: July 7, 2020

Name: Wenqian Liu Signature: Wenqian Liu Date: July 7, 2020

**Project Timeline**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Progress Stage** | **Start Date** | **Tasks/Requirements** | **Who is responsible for** | **Due Date** |
| 1 | Jul 6, 2020 | Finalize the project scenario. Detail the reasoning behind our choice, our motivation, the research we have performed and our initial plan of action. | All members | Jul 13, 2020 |
| 2 | Jul 13, 2020 | **Find the dataset** | All members | Jul 20, 2020 |
| Create the database schema. | Junyao Xie, Chunye Xie |
| Draw the ER diagram. | Dongquan Qiu, Wenqian Liu, Kexin Lian |
| 3 | Jul 20, 2020 | Revise the database schema and ER diagram according to feedback. | Junyao Xie | Jul 27, 2020 |
| Clean and combine Datasets into one. | Dongquan Qiu |
| Transform and enter the data to the database system using Python. | Chunye Xie, Wenqian Liu, kexin Lian |
| 4 | Jul 27, 2020 | Revise Python code for transforming and entering the data to the database system. | Kexin Lian | Aug 3, 2020 |
| Plan for how the customers will interact with the database system. | Chunye Xie, Junyao Xie |
| plan for redundancy and performance. | Dongquan Qiu, Wenqian Liu |
| 5 | Aug 3, 2020 | Metabase Dashboard | All members | Aug 10, 2020 |

**The Most Challenging task**

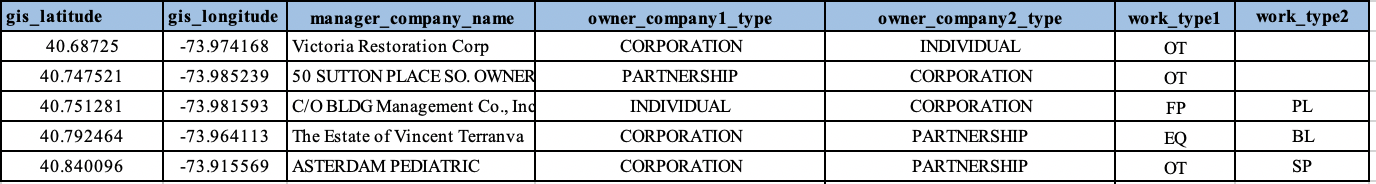
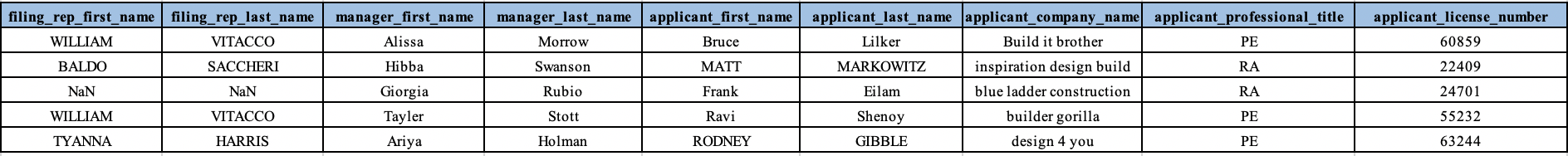
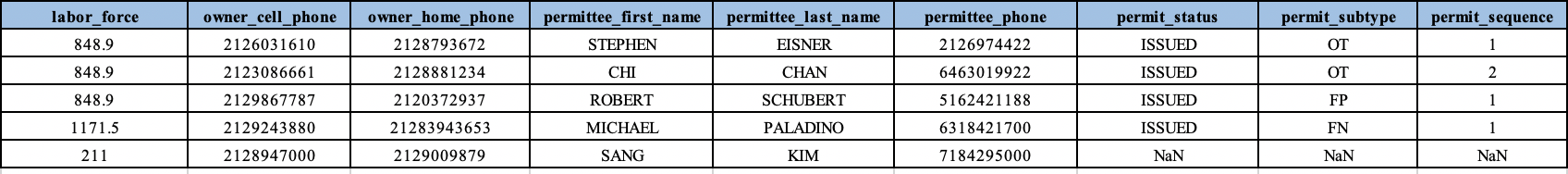
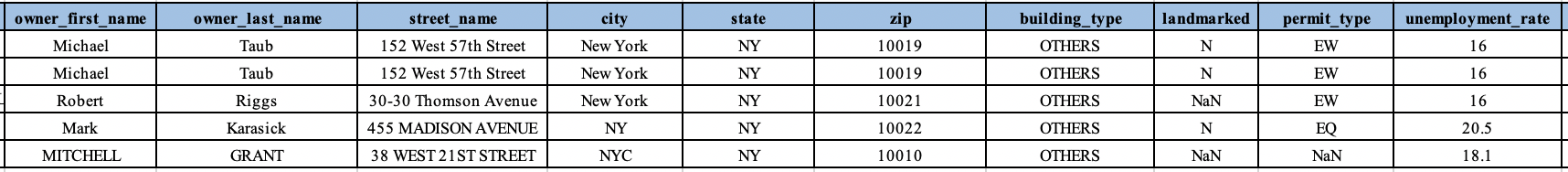
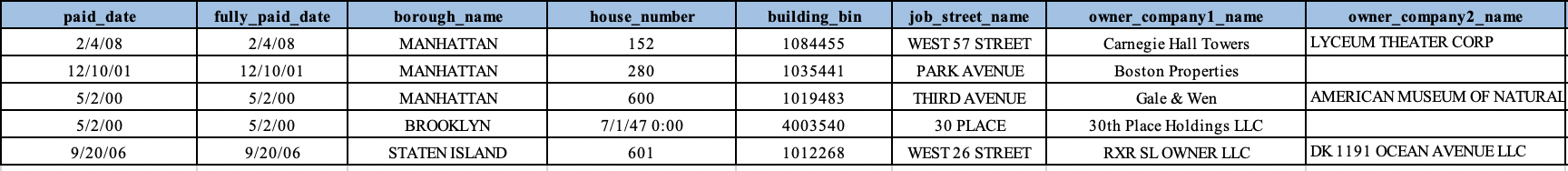
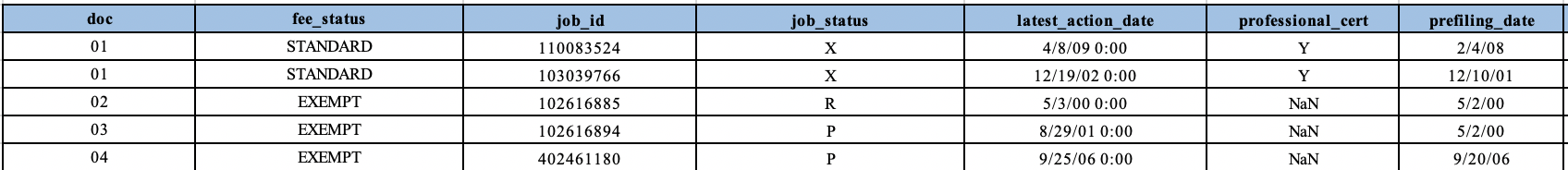
As highlighted on the table above, the most challenging task we met in this project is finding the dataset. The dataset in this project is unusual since it requires us to normalize it into at least 15 tables. Such a dataset might be common in any data-related organization, but it was difficult for students to find in an open resource. Everyone in our team had spent several days exploring datasets on kaggle, kdnuggets, Quora, etc., but all of them are not appropriate for our project. Some of them are pre-defined, and others can not result in 15 tables at 3NF. Finally, we found our dataset on a government web site.

**Database Schema**

**Normalization plan and execution**

There are 50 variables in our dataset combined and organized by four related datasets. We normalized the relational database in accordance with a series of so-called normal form[s](https://en.wikipedia.org/wiki/Database_normalization#Normal_forms) in order to reduce data redundancy and improve data integrity. Our SQL code for creating tables is provided in Appendix.

**Initial data**

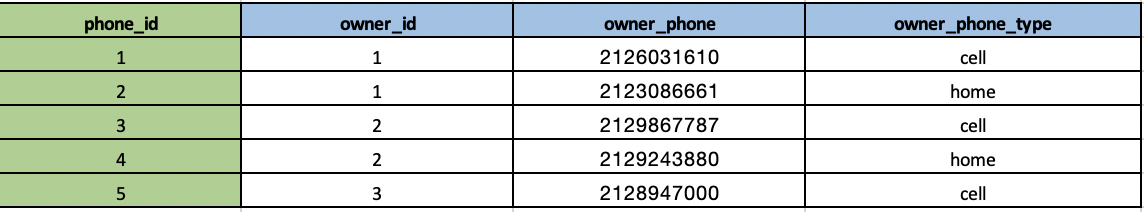
****

**Satisfying 1NF**

To satisfy 1NF, the values in each column of a table must be atomic. Entries in a column are the same type. All our initial tables satisfy these requirements except for the onwer\_phone1, onwer\_phone2(cell phone and home phone), work\_type1, work\_type2, owner\_company1, and owner\_company2. Therefore, we separated information out of the initial table and made a table called phone\_number, another table called work\_type, and a third table called owner\_copmany as following:

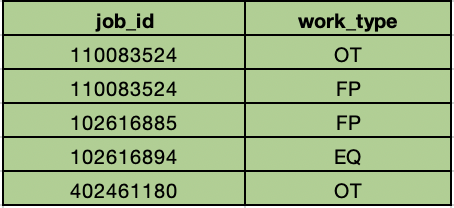
\*Green indicates that this column is the primary key.

Phone\_number



\*We assign each owner an id. The owner information table is shown in 2NF.

Work\_type



Owner\_company

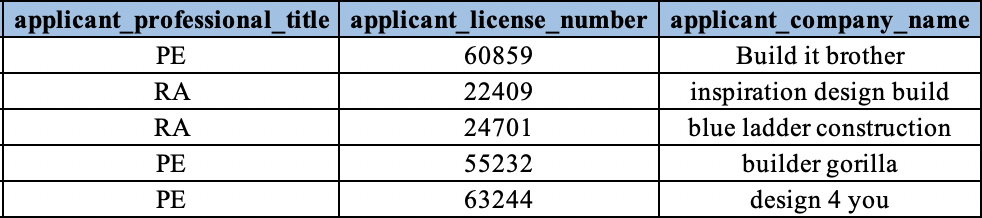
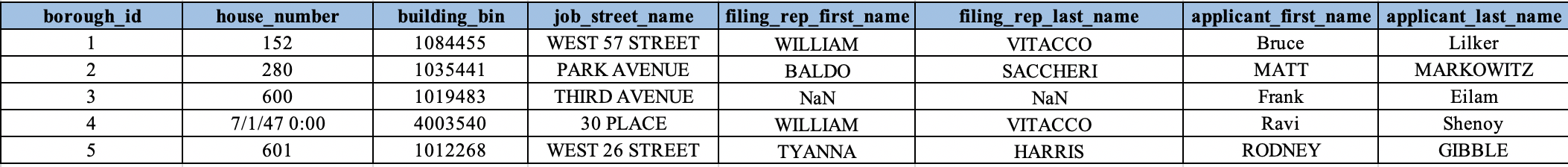
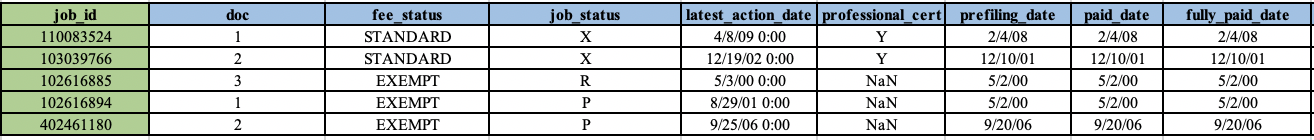


\*We assign each owner’s company an id. The owner’s company information table is shown in 2NF.

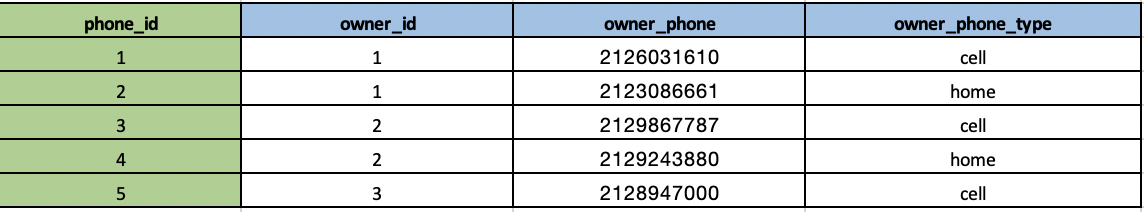
**Satisfying 2NF**

To conform to 2NF and remove duplicities, every non key attribute must depend on the key attribute. Obviously, if we assume job\_id is the primary key for the initial table, not all attributes depend on it. For example, unemployment\_rate and labor\_force depend on boroughs(borough\_name) rather than on doc\_id. Here, we divided the initial tables into 8 tables.

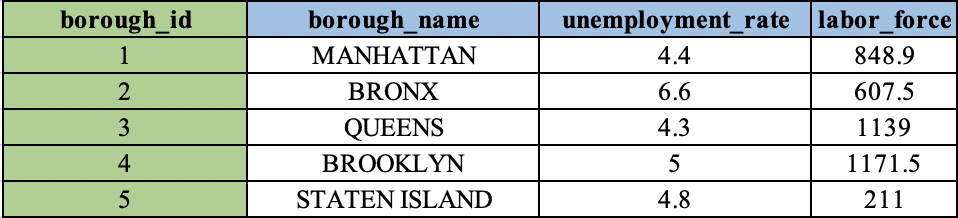
Job\_info



Phone number



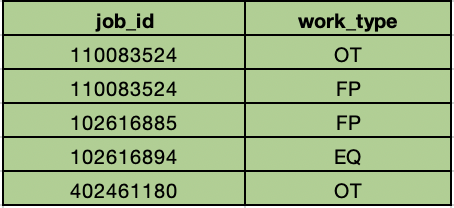
Borough info



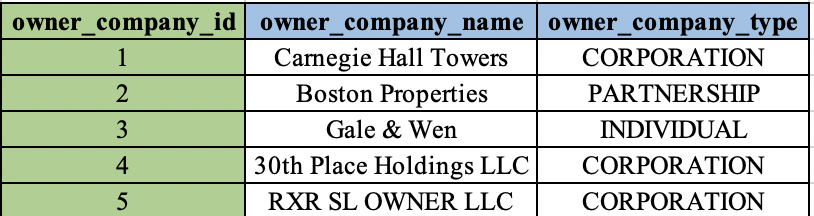
Owner\_company



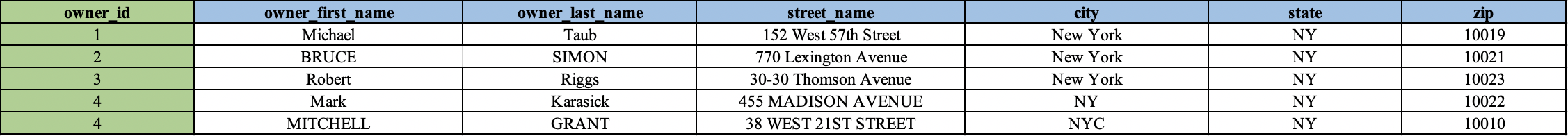
Work\_type



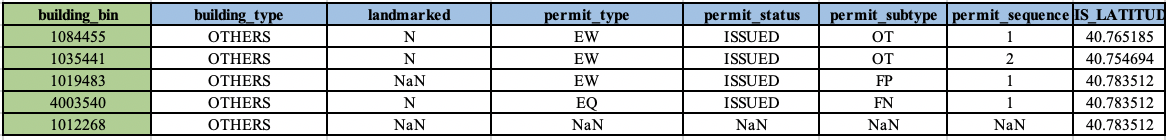
owner\_company\_info

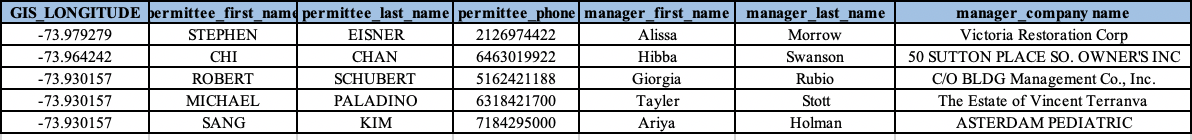


Owner



Building



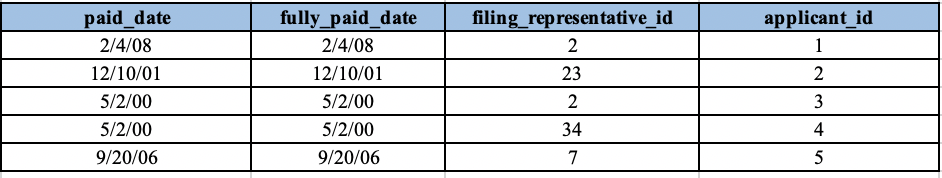
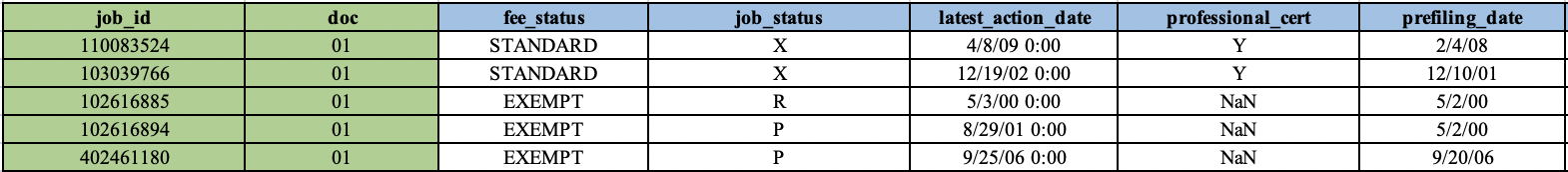


**Satisfying 3NF**

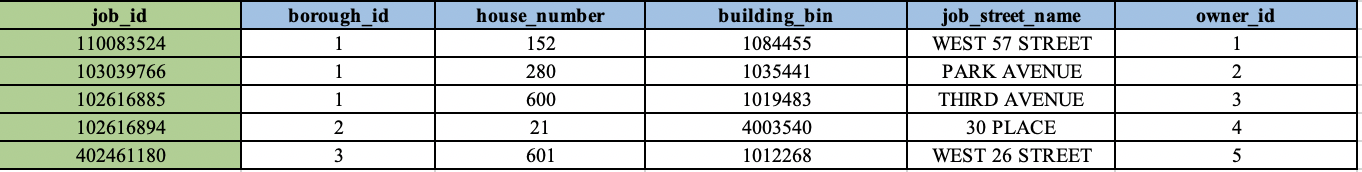
A table in third normal form is a table in 2NF that has no transitive dependencies. This means that all fields can be determined and only by the key in the table and no other column.

In our 2NF tables, some attributes do not only depend on the primary key, but also depend on the non-key attribute. For example, in the job table, fee\_status, job\_status, latest-action\_data, professional\_cert, prefilling\_date, paid\_date, fully\_paid\_date, and filing\_representative not only depend on job\_id, but also on doc. However, borough\_id, house\_number, building\_bin, and job-street\_name only depend on the job\_id, not on doc. Therefore, we separated them into two tables. Our all 3NF tables are shown as following:

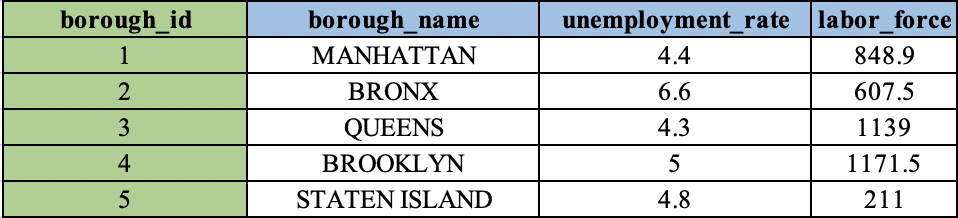
Job\_info



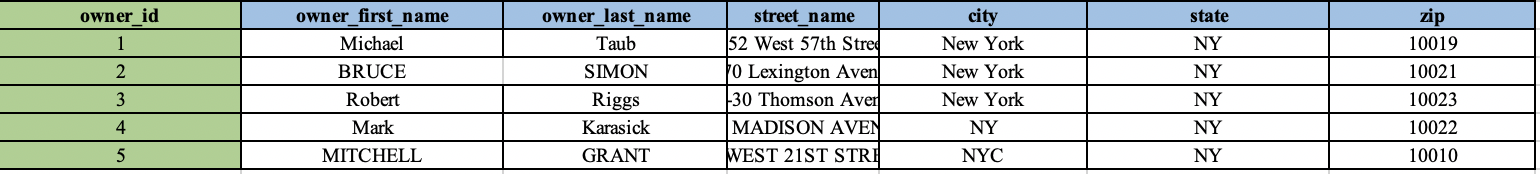
Job-location



Borough\_info



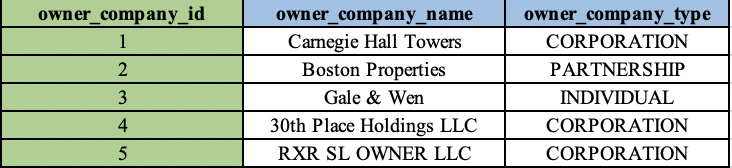
Owner



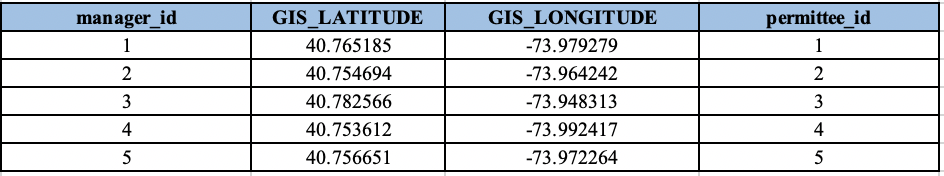
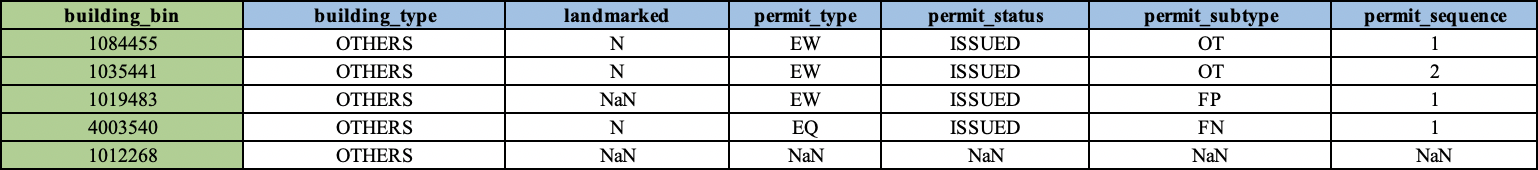
Owner\_company



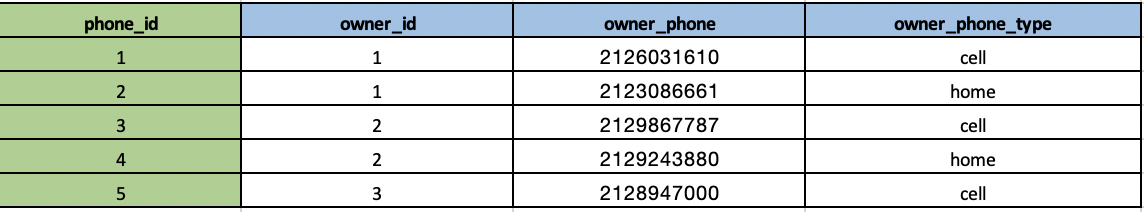
Owner\_company\_type



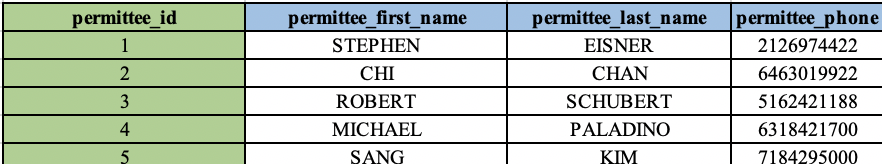
Building



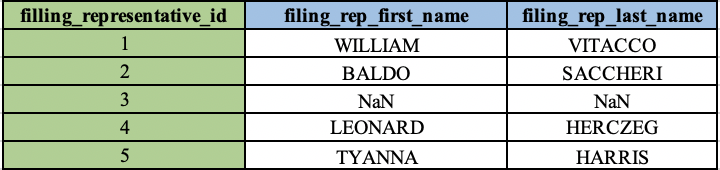
Phone\_number



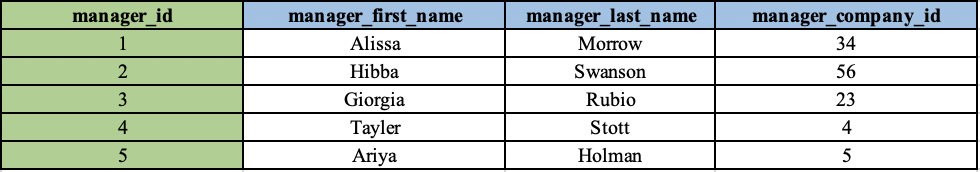
Permittee



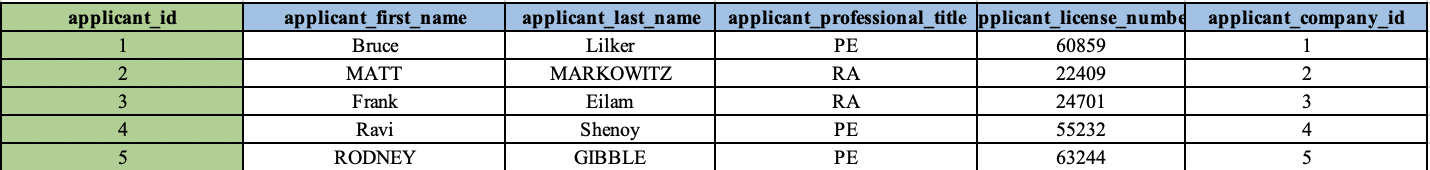
Filing\_representatives



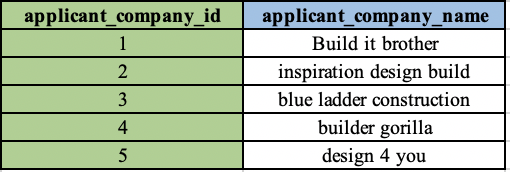
Manager



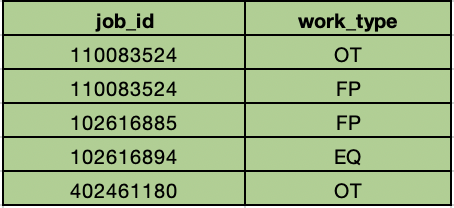
Applicant



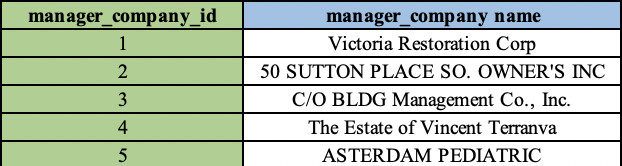
Applicant\_company\_info



Work\_type



Manager\_company\_info



**ER diagrams**

We created an ER diagram on LucidChart. See Appendix “ER diagram”. Here is the link to LucidChart: <https://app.lucidchart.com/documents/edit/6b149624-2460-4009-9d59-d029f81fbc21/0_0>

**ETL plan and execution**

We use Python to perform the ETL process and we used the Jupyter Notebook as our coding tool because it shows the results line by line and provides great visualization after each process. It can also be used to show the process to others.

The first step is to load necessary packages that will allow us to perform the ETL process. We mainly used ‘Pandas’ for dataframe manipulations and ‘Numpy’ for mathematical computations. We also used sqlalchemy to build connections between Python and PostgreSql. We wrote the SQL code based on our normalization plan in advance and tested the code for each table in PostgreSql to ensure correctness. We then use Python to connect to the database and pass the statements to create the tables in our database.

With the database and all tables created, we started to extract, transform and load the dataset into the database. We loaded the dataset from the local file path. Our dataset is saved as the Excel format, therefore we use the read\_Excel function from Pandas to load the data. Before making the modifications to the dataset, we performed some dataset exploration process to ensure that our normalization plan properly corresponds to the information stored in the dataset. Next, we started our ETL process and created small data frames for each table we have in the database. We extracted values based on the attributes for each table and added unique identifiers as the primary keys. We first load the data into those tables which can exist independently, and then load the data into the tales with foreign key constraints.

In the process of ETL, we discovered some inherent errors in the original dataset. For example, there is a “zip code” column in the dataset. We designed the relevant table and assigned integer data type to this attribute since we assume that all zip code should be stored in a 5 digit integer format such as “12345” after initial examination of the original dataset. However, there are values such as ‘12345X’ or ‘11001\’, which violates our data type constraint. Therefore, for such values, we made some manual adjustments to the wrong values in order for the dataset to be loaded into the database. For similar errors in other tables, we use different ways to make small adjustments. We are constrained by the original dataset design and therefore we can’t add ‘not null’ constraint to most of our tables, or we won’t be able to load the complete data into the database. We took our most efforts to modify the data based on our table design. However, due to the inherent limitation and the scale of this project, we don’t have the ability to make massive changes to the original dataset. Therefore, there might still be some wrong values in the database.

After making proper adjustments and creating dataframes, we repeated the ETL process for 15 tables and successfully loaded all the data into our database. We performed several Sql queries in PostgreSql to make sure that data is stored properly and fits our table design. Due to the fact that the Python code is too long, we won’t be attaching it to the appendix. You can find the full code and results of our ETL process by clicking the following link:

<https://github.com/cinnabar723/5310-project/blob/master/Project_ETL_v3.ipynb>

**Interaction**

1. **How your customers will interact with the database system you designed.**

NYC Department of buildings(DOB) requires an application submitted for review before any building construction project begins in New York City. Applicants need to download the form from DOB website and mail it to DOB. The database our group designs will comply with DOB’s process and help DOB perform tasks. A complete process of this interaction includes collecting data, cleaning data, organizing data, and transferring data to a cloud data warehouse for future analysis.

Considering the volume and complexity of DOB’s data, we decide not to use the Hadoop system. Although at the present, our customers are using the paper to collect data, they are moving towards an electronic filing system where applicants and companies can submit forms online. Based on DOB’s current situation, firstly, we will build a html website for them to collect the data. Second, DOB uses a relational database management system such as MySQL and PostgreSQL to create and store their source data. For this job application filling data, we created 16 tables, so DOB’s employees need to create 16 tables in a relational database system as we created in the first place. Then, application information will be automatically updated into MySQL every one hour by Fervvo’s database connector. Frevvo’s RESTful Database Connector uses Extensible Markup Language (XML) and JavaScript Object Notation (JSON) to connect the HTML forms and our database via a secure HTTPS connection. Third, since job application filling is just a small segment of DOB’s service, DOB needs an ETL tool such as Talend, Informatica, and Kettle to extract, transform, and load data. For example, ETL tools can save time and be functional when DOB needs to combine job application data with other data like building data, or when DOB wants to migrate data from their database to other departments’ databases for comprehensive decision-making. At last, structured data will be transferred and stored in a cloud database.

1. **What will you implement for analysts (direct querying) and for "C" level officers (reports)? What tools are you using?**

The analysts can use some programming tools like Jupyter Notebook(Python), R studios(R programming), etc. to connect to our database, directly query the database, extract the data they need and analyze the data. Besides, they can also use some non-technical tools like looker and tableau to extract and analyze the data.

As for “C” level officers, we will create a real time dashboard to visualize the data and show the results. It is time-consuming and can’t update timely when we manually make and update a dashboard. Therefore, we will create a SQL dashboard by a tool called Klipfolio that can pull in data from our database, build custom data visualizations and dashboard, and update the dashboard automatically and timely when there are some updates in the database. It has data visualization and dashboard updates in near-real time(refreshing) up to every 60 seconds.

1. **Did you plan for redundancy and performance?**

In addition, we planned for the redundancy and performance. First, the data replication can help prevent data redundancy by storing the same data in multiple locations. We can also have a central and master field to update all of the places where the data is redundant through one central access point. It can ensure the consistency and receive the information needed. Second, certain cloud platforms and databases, including AWS and Talend Data Fabric will help improve the performance and data quality. It will both ensure the data quality and visualization report for different internal staffs as well as obtaining real-time data updates.

**Analytics Applications**

According to the analytical procedures, we are able to apply SQL and Python codes to generate valuable insights through the number and visualization as well. Certain questions are listed below as examples.

1. How many jobs are professionally certified? (#51046)
2. How many jobs available are there in each borough in NYC?
3. How many jobs are available in the last 5 years?
4. What is the most common work type?
5. How many residential buildings (family) are in job applications?
6. What are the most popular streets for building construction?
7. Top 5 manager companies that engaged the most construction?
8. How does the top 1 company distribute its construction on Map?
9. Top 5 owner companies that own the most constructions here?
10. What is the trending of job markets and which area can be the most potential one?

At the beginning of the analytics, we decided to explore the insights in the application and job datasets, in order to find out the characteristics in the job market over the past few years. First, there are 51046 jobs that are professionally certified, which require applicants with certain professional knowledge for the work. The most 5 work types are OT, PL, EQ, MH, and SP. Moreover, the number of available jobs is gradually increasing, from 779 in 2015 to 20036 in 2019, with ~225% compounded growth rate according to the past 5 years. Meanwhile, among boroughs in the New York region, Queens offers the most available jobs in the market and Manhattan offers the second most jobs. In addition, Manhattan obtains the lowest unemployment rate in 2019, indicating that the opportunities and demands are increasing in this specific borough. We believe that Queens and Manhattan maintain the most market shares of jobs with our research and analytics.

On the other hand, we explored the data from the building’s perspective, in order to understand what companies most engaged in the work and what type are they. First, 31009 out of 100009 buildings are categorized as 1-2-3 FAMILY, while the rest are categorized as OTHERS. 1-2-3 FAMILY type building counts as ~44.9% of all. Among the building constructions, the most popular borough is in Queens while the most popular street name is BROADWAY street. It matches our conclusion that Queens maintain the most available jobs above as well. Second, the top 3 companies that own the most construction buildings are NY School Construction Authority, NYC HPD, and NYCHA. These are the local institutions in the area. In addition, BEECH ASSO is the top 1 manager company that engaged the most construction buildings in this case. The majority of its distribution are located in Manhattan and Queens. Overall, these two boroughs are the most popular areas with construction buildings work and maintain huge opportunities to offer work among all boroughs.

**Metabase**

We put four insights into our Metabase dashboard. The first one was automatically generated by Metabase, the other three were generated by Sql query executions in Metabase and further visualization.

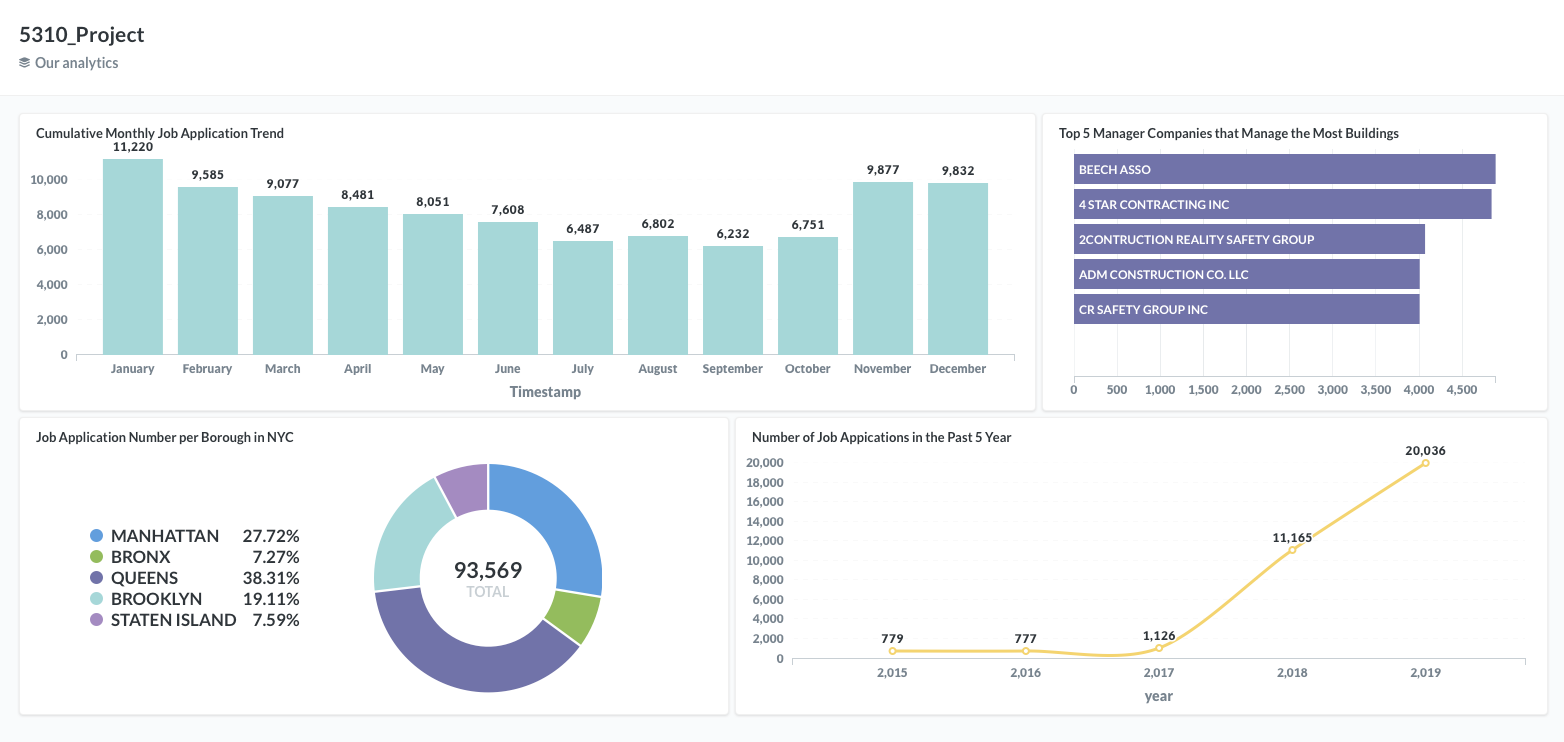
First, we showed the annual and monthly trend of the number of job applications in each month for all years. We can see that there is a clear declining trend from January to October, then the number of job applications increased from November to December, which means people plan more construction work during winters, and plan less during summer.

Second, we listed the top 5 companies that manage the most buildings. Those companies are normally large companies and monitor most of the city’s constructions. It is always a good idea to maintain good relationships with these companies and keep an eye on them.

Third, we listed the number of job applications for each borough in NYC. From the data shown, we can see that most building construction planning took place in Queens, and least in Bronx.

Lastly, we draw a curve and show the job construction planning growth situation for the past 5 years. We can see a dramatic increase in the number of job applications since 2017, which could be an indicator of growth in favorable policies, number of residents and boost of the economy.

The dashboard provides a basic visualization and demonstration of the situation of the building construction progress in NYC. Audiences will have a understanding of the amount, location, time and trend of the constructions based on the information provided on the dashboard.

****

The link for the interactive dashboard can be found below:

http://localhost:13195/public/dashboard/3a35b13d-7a1f-4e0b-bfdc-817823ea9362

**Conclusion**

We aimed to build an accurate and efficient database for our client. Firstly, we executed the normalization and drew an ER diagram to figure out the database schema. Then we prepared the sql code for each table. After that, we ETL the dataset into our database through Python. Besides, we made a dashboard for the C-level officers to timely check the change.

**Appendix**

**Schema SQL Code**

create table applicant\_company\_info(

applicant\_company\_id integer,

applicant\_company\_name varchar(200),

PRIMARY KEY (applicant\_company\_id)

);

create table applicant(

applicant\_id integer,

applicant\_first\_name varchar(50),

applicant\_last\_name varchar(50),

applicant\_professional\_title varchar(50),

applicant\_license\_number varchar(10),

applicant\_company\_id integer,

primary key (applicant\_id),

FOREIGN KEY (applicant\_company\_id) REFERENCES applicant\_company\_info (applicant\_company\_id)

);

create table permittee(

permittee\_id integer,

permittee\_first\_name varchar(50),

permittee\_last\_name varchar(50),

permittee\_phone varchar(20),

primary key (permittee\_id)

);

create table filing\_representative(

filing\_representative\_id integer,

filing\_rep\_first\_name varchar(50),

filing\_rep\_last\_name varchar(50),

primary key (filing\_representative\_id)

);

create table borough\_info(

borough\_id integer,

borough\_name varchar(50),

unemployment\_rate varchar(10),

labor\_force integer,

primary key (borough\_id),

check (borough\_name in('BRONX','BROOKLYN','QUEENS','STATEN ISLAND','MANHATTAN'))

);

create table manager\_company\_info (

manager\_company\_id integer,

manager\_company\_name varchar(200),

PRIMARY KEY (manager\_company\_id)

);

create table manager(

manager\_id integer,

manager\_first\_name varchar(50),

manager\_last\_name varchar(50),

manager\_company\_id integer,

primary key (manager\_id),

FOREIGN KEY (manager\_company\_id) REFERENCES manager\_company\_info(manager\_company\_id)

);

create table owner(

owner\_id integer,

owner\_first\_name varchar,

owner\_last\_name varchar,

street\_name varchar(200),

city varchar(50),

state char(2),

zip integer,

primary key (owner\_id)

);

create table owner\_company\_type(

owner\_company\_id integer,

owner\_company\_name varchar(100),

owner\_company\_type varchar(100),

PRIMARY KEY (owner\_company\_id)

);

create table owner\_company(

owner\_company\_id integer,

owner\_id integer,

primary key (owner\_company\_id, owner\_id),

FOREIGN KEY (owner\_company\_id) REFERENCES owner\_company\_type (owner\_company\_id),

FOREIGN KEY (owner\_id) REFERENCES owner (owner\_id)

);

create table phone(

phone\_id integer,

owner\_id integer,

owner\_phone varchar(20),

owner\_phone\_type varchar(10),

primary key(phone\_id),

foreign key(owner\_id) references owner(owner\_id)

);

create table building(

building\_bin integer,

building\_type varchar(50),

landmarked char(1),

permit\_type varchar(2),

permit\_status varchar(15),

permit\_subtype varchar(2),

permit\_sequence varchar(5),

manager\_id integer,

permittee\_id integer,

gis\_latitude numeric(10,6),

gis\_longitude numeric(10,6),

primary key(building\_bin),

Foreign key (manager\_id) references manager (manager\_id),

Foreign key (permittee\_id) references permittee (permittee\_id),

check (landmarked in('Y','N','L','C')),

check (building\_type in('1-2-3 FAMILY','OTHERS')),

check (permit\_status in ('IN PROCESS','ISSUED','REVOKE','RE-ISSUED')),

check (permit\_type in ('AL','DM','EQ','EW','FO','NB','PL','SG'))

);

create table job\_location(

job\_id integer,

borough\_id integer,

house\_number varchar(20),

building\_bin integer,

job\_street\_name varchar(100),

owner\_id integer,

primary key(job\_id),

Foreign key (borough\_id) references borough\_info(borough\_id),

Foreign key (building\_bin) references building(building\_bin),

FOREIGN key (owner\_id) REFERENCES owner (owner\_id)

);

create table job(

job\_id integer,

doc\_id integer,

fee\_status varchar(20),

job\_status varchar(1),

latest\_action\_date date,

professional\_cert varchar(1),

prefiling\_date date,

paid\_date date,

fully\_paid\_date date,

filing\_representative\_id integer,

applicant\_id integer,

primary key(job\_id, doc\_id),

FOREIGN KEY (job\_id) REFERENCES job\_location (job\_id),

FOREIGN KEY (filing\_representative\_id) REFERENCES filing\_representative (filing\_representative\_id),

FOREIGN KEY (applicant\_id) REFERENCES applicant (applicant\_id),

check (professional\_cert in('J','N','Y')),

check (job\_status in('3','A','B','C','D','E','F','H','J','K','P','Q','R','U','X'))

);

create table job\_type(

job\_id integer,

work\_type varchar(2),

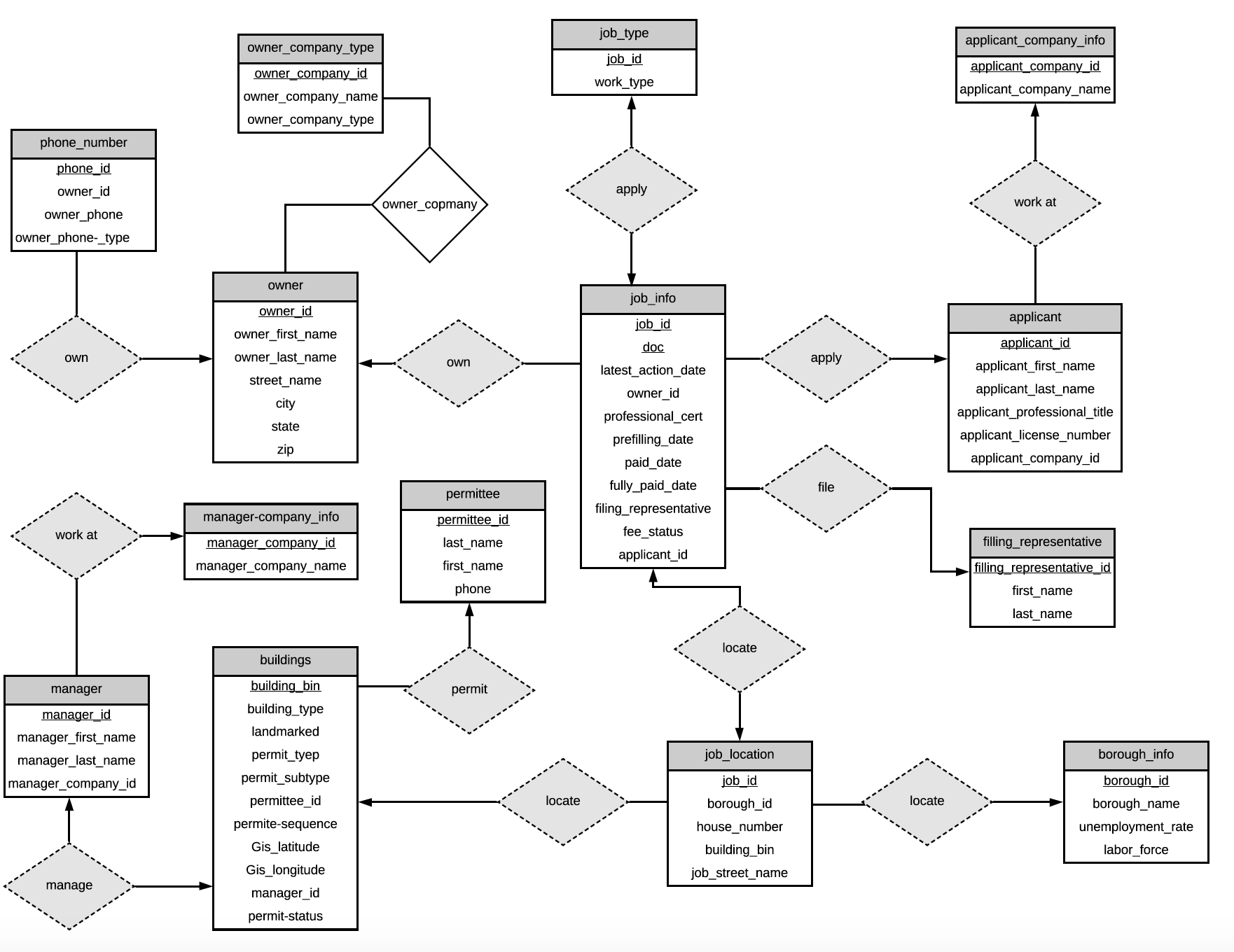
primary key (job\_id, work\_type),

FOREIGN KEY (job\_id) REFERENCES job\_location (job\_id),

check(work\_type in ('BL','CC','EQ','FA','FB','FP','FS','MH','OT','SC','SD','SF','SH','SP','PL'))

);

**ER Diagram**

****