BROOKLYN PROPERTY SALES

A proposal report to the property management company in NYC

Prepared by:
Ergeta Muca, Kristina Liapchin, Iris Zhao, Keerthana (Keetu) Palani, Lang (Lexi) Lin
December 8th, 2019

Table of Contents

Problem Statement

As one of the oldest and largest industries, the real estate industry has been known to be rather slow in adapting to the recent rise of data and technology. Therefore, most real estate companies still fail to make use of available data to inform their business decisions, which can help them gain competitive advantage. Data-driven decision making is becoming a key to success for companies across industries, which is why our team was excited to work on a project that aims to bridge that gap for companies operating in the real estate space.

In order to kick off our project, our team decided to come in open-minded and looked at many potential datasets from different sources. After having searched through Kaggle, data.gov, kdnuggets, and individual companies' public data and API offerings, we have found a New York City Department of Finance dataset on Brooklyn home sales from 2003 - 2017, which we decided to use for our final project. To focus our efforts and narrow down the large dataset for the purpose of this project, we have decided to proceed with solely the data for year 2017.

For this project, our team is working with a property management company in New York City. Our client has acquired data on home sales in Brooklyn in 2017 and is now looking to make sense of that data in order to inform their upcoming property investments and business development and marketing strategies. The client is also planning to have new data come in regarding property sales in New York, which will need to be automatically stored and available for efficient retrieval when needed.

Proposal

By undertaking this project, our team aims to help the client make use of available property data, enabling them to process complex data and translate findings into actionable insights to inform and support the decision-making of internal stakeholders.

Based on the scenario outlined above, our team is going to build a PostgreSQL relational database for the client based on the Brooklyn home sales data for 2017. We will also be accounting for the fact that the client is planning to acquire additional data on home sales across boroughs in NYC to further inform their business and investments. This database will enable the client to easily and efficiently feed and store all property sales data in one place, while also making it available for retrieval whenever needed for further analysis.

In addition to the database, our team will also provide the client with two Metabase dashboards and a number of suggested analyses that can help drive business value and can be conducted using SQL. These procedures and the first dashboard will enable analysts to answer key

business questions, helping inform company-wide decision making as relating to upcoming investments and strategies for business development and marketing. As part of this, we will also build another fully functioning Metabase dashboard with visualizations of key metrics and analyses, which will help translate complex data into valuable insights for key stakeholders, especially the non-technical stakeholders, such as managers and executives. These dashboards will also be automatically updated when new data is added, making it simple to maintain them.

Overall, these tools and processes will help the client generate valuable business insights and enable them to improve their data-driven decision making when it comes to scouting and planning for new investments and developing well-informed strategies for business development, property acquisition, and marketing teams.

Team structure

Our team in this project is comprised of 5 individuals: Ergeta Muca, Kristina Liapchin, Iris Zhao, Lang (Lexi) Lin and Keetu Palani. The team is mobilized in a Scrum project fashion by focusing on 6 main Sprints revolving around completing various key components of the project, specifically: defining the database schema, the ETL process, the analytical procedures (Python + SQL), dashboard visualization for key stakeholders and overall project report/presentation write-ups. The team members are assigned official roles to keep them accountable for specific components, but we organically plan to evolve our roles to help each other out and improve deliverable quality. For each task, designated team members are assigned to complete them, and for quality control, a reviewer is assigned to check the task completed prior to submission. The following table synthesizes our task breakdown throughout the project timeline:

Team Roles

Project Manager: Ergeta Muca
Communications Leader: Kristina Liapchin
Note taker: Iris Zhao
Technical QA Leader: Lang (Lexi) Lin
Business QA Leader: Keetu Palani

Task Breakdown and Project Timeline

Sprint	Task	Responsible	Reviewer	Due	Complete?
1	Brainstorm/Decide project topic	All	N/A	11/1	Yes
1	**Submit Project Checkpoint 1**	**Ergeta**	**Kristina**	**11/3**	**Yes**
2	Develop Team Contract	Ergeta	All	11/8	Yes

2	Develop Project Plan	Ergeta	All	11/8	Yes
2	Individually Draft Schema Design of the Dataset	All	All	11/9	Yes
2	Regroup and Finalize Draft Schema Design	All	All	11/10	Yes
2	**Submit Project Checkpoint 2**	**Ergeta**	**Kristina**	**11/11**	**Yes**
3	Finalize Schema Design	All	All	11/15	Yes
3	Develop ER Diagram on Lucid Chart	Kristina, Lexi	Keetu	11/16	Yes
3	Develop SQL Code	Ergeta, Keetu	Iris	11/7	Yes
3	**Submit Project Checkpoint 3**	**Ergeta**	**Kristina**	**11/18**	**Yes**
4	Develop Python Scripts	Ergeta, Kristina	Iris	11/23	Yes
4	Write up plan	Keetu, Iris	Lexi	11/24	Yes
4	**Submit Project Checkpoint 4**	**Ergeta**	**Kristina**	**11/25**	**Yes**
5	Write up customer interaction plan - analysts	Kristina	Lexi	12/1	Yes
5	Write up customer interaction plan - C Level	Keetu	Ergeta	12/1	Yes
5	Write up customer interaction plan - tools used	Iris	Keetu	12/1	Yes
5	Write up customer interaction plan - redundancy/performance	Ergeta	Kristina	12/1	Yes
5	Set up Metabase dashboard	Kristina	Ergeta	12/1	Yes
5	**Submit Project Checkpoint 5**	**Ergeta**	**Kristina**	**12/2**	**Yes**
6	Report Write Up - Scenario Description	Kristina	Iris	12/6	Yes
6	Report Write Up - Team Structure & Timeline	Ergeta	Keetu	12/6	Yes
6	Report Write Up - Sample of Data + Dataset	Kristina	Ergeta	12/6	Yes
6	Report Write Up - Normalization + Schema/ER Diagram	Iris	Lexi	12/6	Yes
6	Report Write Up - Work on Normalization & ETL Process	Ergeta	Kristina	12/6	Yes
6	Report Write Up - Analytical Procedures (SQL) Explanation	Lexi	Keetu	12/6	Yes
6	Report Write Up- Interaction on Analyst + C level Suite	Keetu	Iris	12/6	Yes
6	Report Write Up- Redundancy & Performance/ Cloud On Prem	Keetu	Kristina	12/6	Yes
6	Report Write Up- Metabase Dashboard Explanation	Keetu	Lexi	12/6	Yes
6	Report Write Up- Conclusion	Kristina	Ergeta	12/6	

6	Finalizing ETL + Schema	Ergeta	Iris	12/5	Yes
6	Finalizing SQL Queries	Lexi	Kristina	12/6	Yes
6	Metabase Dashboard Creation	Keetu, Ergeta, Lexi	Kristina	12/6	Yes
6	Prepare Presentation Slides	Iris	Lexi	12/6	Yes
6	Finalize Presentation Slides	Ergeta	Iris	12/6	Yes
6	Rehearse the presentation	All	All	**12/7**	Yes
6	**Submit Report and Final Project**	**Ergeta**	**Kristina**	**11/9**	Yes

Some of the more challenging tasks throughout the project include the various revisions done to the dataset and the ETL process. These are due to new information that are added to the original dataset and additional database expansion considerations we are taking into account. The team held weekly team meetings to touch base on checkpoint requirements as well as met to rehearse for the final presentation.

Database Schema Design

Before getting started with the project, we have narrowed down our initial Brooklyn home sales dataset to 23,955 rows, focusing solely on data from 2017, and 26 columns. Below is a snapshot of the dataset:

address	total_units	land_sqft	residential_units	commercial_units	block	lot	sale_price	sale_date	year_of_sale	apartment_number	gross_sqft	zip_code
21 CLARK STREET	1	20267	0	1	230	1	202500000	10/31/2017	2017		356000	11201
16 COURT STREET	102	12500	0	102	250	44	171000000	10/10/2017	2017		290440	11241
20 NORTH 12 STREET	0	60400	0	0	2287	16	160000000	4/19/2017	2017		0	11249
55 PROSPECT STREET	2	20704	0	2	63	1	138106368	3/31/2017	2017		253096	11201
90 SANDS STREET	1	21175	0	1	87	9	135000000	8/29/2017	2017		363100	11201

building_class_category	building_class	OwnerType	OwnerName	tax_class	XCoord	YCoord	neighborhood	FireComp	PolicePrct	CD	Council	SchoolDist
26 OTHER HOTELS	H8	X	WATCHTOWER C/O REAL P	4	985622	193713	BROOKLYN HEIGHTS	E205	84	302	33	13
21 OFFICE BUILDINGS	O4	P	16 COURT STREET OWNER	4	986784	191977	BROOKLYN HEIGHTS	E205	84	302	33	13
41 TAX CLASS 4 - OTHER	Z9	P	10TH STREET LLC	4	995151	202986	WILLIAMSBURG-NORTH	L106	94	301	33	14
21 OFFICE BUILDINGS	O6		55 PROSPECT OWNER LLC	4	987524	194630	DOWNTOWN-METROTECH	L118	84	302	33	13
26 OTHER HOTELS	H8	P	WATCHTOWER C/O REAL P	4	987811	194154	DOWNTOWN-METROTECH	L118	84	302	33	13

Our team then combined this dataset with additional external data on the individual community and school districts, police precincts, and fire companies. This information will be particularly valuable to the client when analyzing the different neighborhoods and potential properties for investment, as it can help paint a fuller picture of the location and surroundings. The individual data sources used to create our final dataset can be found in the references section of the report. Once the final dataset is complete, we proceed with schema design and normalization.

Normalization and Schema Design

For this project, 15 tables were created that capture essential relationships between our entities of interest. The details of these tables are as follows:

** Please see the full normalization plan at:*
https://drive.google.com/file/d/1zw9ZPf8peE_5KcCW3bu450jfX3P1xvZI/view?usp=sharing

Owners:

owners		
owner_id int (PK)	owner_name varchar(100) NOT NULL	owner_type char(1)

This table includes the owner of the property and the type of the owner (C, O, X and P).

Addresses:

addresses				
address_id int (PK)	address varchar(100) NOT NULL	zipcode int	lon numeric(10,6),	lat numeric(10,6),

This table includes the address of the properties and the zipcode. Range of Zip Code is limited/ defined, so we do not need to separate zip code to a new table. Longitude and latitude are pure numeric values, so we did not split it into a new table as well.

Neighborhoods:

neighborhoods		
neighborhood_id int (PK)	neighborhood varchar(100) NOT NULL	Borough varchar(70)

This table is the property's neighborhood. Department of Finance assessors determine the neighborhood name in the course of valuing properties. Eg. BROOKLYN HEIGHTS, WILLIAMSBURG-NORTH. The borough information was added to account for additional information that might be added in the future (i.e database expanding beyond just Brooklyn).

Building_class_categories:

building_class_categories	
building_class_category_id int (PK)	building_class_category varchar(100) NOT NULL

This is a field that we are including so that users of the Rolling Sales Files can easily identify similar properties by broad usage (e.g. One Family Homes) without looking up

individual Building Classes.Sample content: 26 OTHER HOTELS, 30 WAREHOUSES.

Building_classes:

building_classes	
building_class_id int (PK)	building_class char(2) NOT NULL

The Building Classification is used to describe a property's constructive use. The first position of the Building Class is a letter that is used to describe a general class of properties (for example "A" signifies one-family homes, "O" signifies office buildings. "R" signifies condominiums). The second position, a number, adds more specific information about the property's use or construction style (using our previous examples "A0" is a Cape Cod style one family home, "O4" is a tower type office building and "R5" is a commercial condominium unit). The term Building Class used by the Department of Finance is interchangeable with the term Building Code used by the Department of Buildings.

Building_class_category_combinations:

building_class_categories_combinations	
building_class_id int (PK)	building_class_category_id int

This table combines the classes with their categories. The former two tables serve as look up tables of this combination table.

Tax_classes:

tax_classes	
tax_class_id int (PK)	tax_class varchar(10) NOT NULL

Every property in the city is assigned to one of four tax classes (Classes 1, 2, 3, and 4), based on the use of the property.

Community_districts:

community_districts				
community_district_id int (PK)	community_district int NOT NULL	population int	area_sq_mi numeric(3,1)	male_perc numeric(3,1)
female_perc numeric(3,1)	foreign_born_pop numeric(3,1)	majority_race_pop varchar(10)	access_to_parks int	commute_to_work_in_min numeric(3,1)

education_bachelor_level numeric(3,1)				

This table describes the community features where the property stays in. It serves as a look-up table for table Properties.

Council_districts:

council_districts	
city_council_district_id int (PK)	city_council_district int NOT NULL

This table describes the council district where the property belongs. It serves as a look-up table for table Properties.

School_districts:

school_districts				
school_district_id int (PK)	school_district int NOT NULL	students_enrolled int	nr_schools int	school_dist_address_id int

This table describes the school districts where the property belongs to. It serves as a look-up table for table Properties.

Fire_Companies:

fire_companies		
fire_company_id int (PK)	fire_company char(4) NOT NULL	fire_comp_address_id int

This table describes the fire companies near the property. It serves as a lookup table for table Properties.

Police_precincts:

police_precincts			
police_precinct_id int (PK)	police_precinct int NOT NULL	police_prct_address_id int	phone_number varchar(12)

This table describes the police stations/precincts near the property. It serves as a look-up table for table Properties.

Properties:

properties				
property_id int (PK)	address_id int	neighborhood_id int NOT NULL	building_class_category_id int	building_class_id int

police_precinct_id int	land_sqft numeric(10,1) NOT NULL	total_units int NOT NULL	res_units int NOT NULL	comm_units int NOT NULL
tax_class_id int	community_district_id int	city_council_district_id int	school_district_id int	fire_company_id int
block int NOT NULL	lot int NOT NULL			

This table is the describes all the features of each property.

Property_sales:

property_sales				
sale_id int (PK)	property_id int NOT NULL	sale_price numeric (20,1) NOT NULL	sale_date date NOT NULL	year_of_sale numeric(4,0) NOT NULL
tax_class_id int	apartment_number varchar(20)	gross_sqft numeric(20,1) NOT NULL		

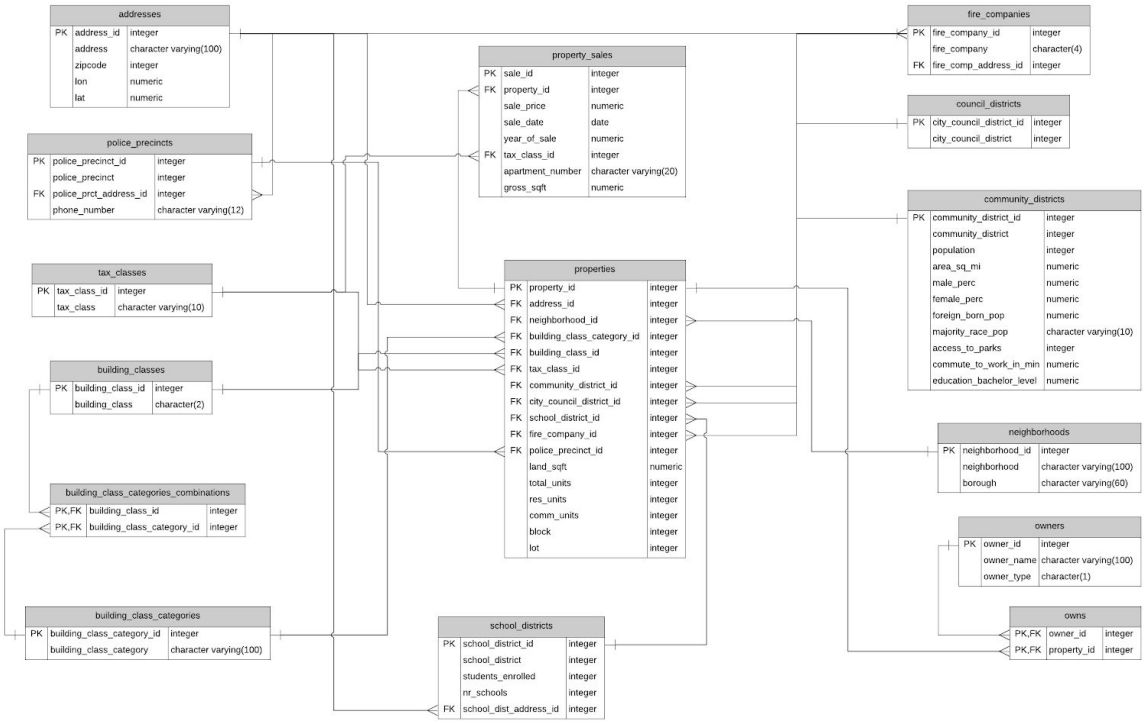
This table describes every property trade/sale. Note that one property can be traded multiple times and one owner can own multiple properties.

Owns:

owns	
owner_id int (PK)	**property_id int (PK)**

This table describes the ownership relation between owners and properties.

ER Diagram



ETL Process

The following steps are taken in our ETL process:

Connect to the PostgreSQL server and database

The team has its own server to operate on for this project. A database called ***brooklyn_home*** is created on pgAdmin to store our data.

Load and inspect the original data

The original property sales dataset from Kaggle is loaded into a Python jupyter notebook. Additional tables containing extra information about community districts, school districts, fire companies and police precincts are loaded and merged together with the original property sales dataset. The extra information on community districts, school districts, fire companies and police precincts are extracted from official NY data website, which are cited in the references section. This information was extracted via web scraping and API functionalities.

Transform the original data into necessary tables

The data is processed and transformed to create the 15 tables in our final database. Various techniques are used to transform the data, including dropping duplicates, removing NAs, renaming columns to fit the final database schema details, merging new tables with reference tables previously created.

Ingest the transformed data into the PostgreSQL database and the respective tables

Once the table transformations are complete, the tables are loaded to our PostgreSQL database and loading was verified on pgAdmin.

**Details of our ETL process can be found on our Github repository:*
https://github.com/ergetamuca/sql_project_team4/blob/master/ETL%20Process%20-%20Brooklyn%20Home%20Sales.ipynb

Analytics Applications and Procedures

In this project, we came up with 10 analytical procedures which aim to provide insights on the properties and the property sales made in Brooklyn in 2017. The first three analytical procedures are some statistics and facts for the property sales made in 2017 and the rest of them are some insightful ideas for management, business development and sales/marketing team. The following are 10 analytical procedures and the explanations for them:

What date of every month in 2017 has closed the most frequent property sales?

This analytical metric is important for marketing teams to help them better plan their marketing campaign start times. It also helps internal team to better allocate sales team resources and mobilize them for client interactions based on specific times of the month.

```
WITH daily_sales AS (
SELECT
COUNT(sale_id) as num_sales,
sale_date
FROM property_sales
GROUP BY sale_date
ORDER BY num_sales DESC
),
sales_ranked AS(
SELECT
MAX(num_sales) as max_sales,
RANK() OVER (PARTITION BY EXTRACT(MONTH FROM sale_date) ORDER BY num_sales DESC) as
max_sales_rank,
sale_date
FROM daily_sales
GROUP BY num_sales, sale_date
)
SELECT
max_sales as num_sales,
sale_date
FROM sales_ranked
WHERE max_sales_rank = 1;
```

What is the average square footage of properties sold in the top 10 selling neighborhoods?

This analytical metric is important to understand where the highest volume of sales is happening in Brooklyn and also get some insight on the typology of properties sold. Square footage is an important measure in understanding what consumers' preferences are in property purchases.

```
SELECT neighborhood, rank, avg_sqft
FROM
(SELECT
 p.neighborhood_id,
 RANK() OVER (ORDER BY COUNT(*) DESC) AS rank,
 ROUND(AVG(s.gross_sqft),2) as avg_sqft
 FROM property_sales s
 JOIN properties p
 ON s.property_id = p.property_id
 GROUP BY p.neighborhood_id) a
JOIN
neighborhoods n
ON a.neighborhood_id = n.neighborhood_id
WHERE a.rank<=10
ORDER BY a.rank;
```

What's the trend of the changes in sales from month-to-month?

This analytical procedure calculates the changes in number of property sales between the current and the previous month. This can help build a solid predictive model for the property management team in the future and can help the team focus their sales efforts into specific times of the year.

```
WITH cnt_views AS (
 SELECT
 DATE_PART('month', sale_date) AS sale_month,
 COUNT(sale_id) AS month_views
 FROM
 property_sales
 GROUP BY
 sale_month
)
SELECT *,
(month_views - previous_month_views) as change
FROM
(SELECT
 sale_month,
 month_views,
 LAG(month_views,1) OVER (
 ORDER BY sale_month
 ) previous_month_views
FROM
```

```
 cnt_views) a
;
```

How many properties are sold by neighborhood and month?

This metric will offer an idea of the "hottest" locations for the properties sold. This insight is helpful for executives to determine their property management strategies in the future.

```
SELECT
N.neighborhood,
EXTRACT('month' FROM sale_date) as sales_month,
COUNT(ps.sale_id) as nr_property_sales
FROM property_sales AS ps
JOIN properties p ON ps.property_id=p.property_id
JOIN neighborhoods n ON n.neighborhood_id=p.neighborhood_id
GROUP BY n.neighborhood_id, sales_month
ORDER BY sales_month;
```

For owners that made more than one purchase, what are the overall trends in purchasing?

This insight will be helpful for them in setting their sales/marketing team up for success and prioritizing their marketing strategies and resources according to the purchasing trends of the clients made frequent purchases.

```
SELECT distinct ow.owner_id, o.owner_name, COUNT(*) as nr_property_purchases,
MAX(ps.sale_price) as most_expensive_property_bought,
n.neighborhood as neighborhood_of_most_expensive_property_bought,
sale_date as date_most_expensive_property_was_bought
FROM owns as ow
JOIN owners as o on ow.owner_id=o.owner_id
JOIN property_sales as ps on ow.property_id=ps.property_id
JOIN properties as p on ps.property_id=p.property_id
JOIN neighborhoods as n on p.neighborhood_id=n.neighborhood_id
GROUP BY ow.owner_id, o.owner_name, n.neighborhood, sale_date
HAVING COUNT(*) > 1
ORDER BY n.neighborhood;
```

What are the top tax classes of the sold properties within the top 10 selling neighborhoods?

This insight provides an overview of the majority tax class of properties within the top selling neighborhoods. It might be helpful for the sales team to communicate with clients regarding the property assessment and it also helps focus the investment opportunity efforts for the management team.

```
SELECT a.neighborhood, a.nr_sales,a.nr_prop_sold_for_majority_tax_class,a.majority_tax_class
      FROM
      (SELECT n.neighborhood, COUNT(ps.sale_id) AS nr_sales, RANK() OVER (ORDER BY COUNT(ps.sale_id)
DESC),
```

```
	(SELECT MAX(tax_class)
	FROM (VALUES
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1'))),
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1B'))),
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1C'))),
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2'))),
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2A'))),
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2B'))),
	((COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='4')))) AS value(tax_class))
	AS nr_prop_sold_for_majority_tax_class,
	(SELECT CASE GREATEST(
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1')),
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1B')),
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1C')),
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2')),
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2A')),
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2B')),
	(COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='4')))
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1')) THEN 'tax_class_1'
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1B')) THEN 'tax_class_1B'
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='1C')) THEN 'tax_class_1C'
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2')) THEN 'tax_class_2'
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2A')) THEN 'tax_class_2A'
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='2B')) THEN 'tax_class_2B'
	WHEN (COUNT(ps.tax_class_id) FILTER (WHERE t.tax_class='4')) THEN 'tax_class_4'
	END AS majority_tax_class)
	FROM property_sales As ps
	JOIN properties AS p ON ps.property_id=p.property_id
	JOIN neighborhoods AS n ON p.neighborhood_id=n.neighborhood_id
	JOIN tax_classes AS t ON t.tax_class_id=ps.tax_class_id
	JOIN building_classes AS bc ON bc.building_class_id=p.building_class_id
	JOIN building_class_categories AS bcc ON
bcc.building_class_category_id=p.building_class_category_id
	GROUP BY n.neighborhood
	ORDER BY nr_sales DESC) a
	WHERE a.rank<=10;
```

What are the details of the outlier properties sold (i.e properties sold at a very high price)?

There might be houses that have sold for a high amount of money. It would be helpful for the management to see what characteristics these houses have that is making them sell for so high.

```
SELECT n.neighborhood,ROUND(ps.sale_price) property_price, ps.sale_date,  ps.gross_sqft
FROM property_sales AS ps
JOIN properties p ON ps.property_id=p.property_id
JOIN neighborhoods n ON n.neighborhood_id=p.neighborhood_id
WHERE ps.sale_price>=100000000;
```

Who are the owners of the top 5 sold properties (the ones with the highest price point)?

This insight will be helpful for prospective client management and business development.

```
SELECT a.owner_name, a.sale_price
FROM
(SELECT ow.owner_name, ps.sale_price, RANK() OVER (ORDER BY ps.sale_price DESC) as rank
FROM property_sales AS ps
JOIN owns o ON ps.property_id=o.property_id
JOIN owners AS ow ON ow.owner_id=o.owner_id
ORDER BY ps.sale_price DESC) a
WHERE rank<=5;
```

What is the typology of the top 5 neighborhoods with the highest number of property sales? I.e how many school districts, police districts, fire companies do they include?

This insight will give the management an indication of how individuals are selecting properties based on their location. Paired with additional research, this could help analyze investment opportunities in areas that might be surrounded by more schools, police districts etc.

```
SELECT a.neighborhood, a.nr_property_sales, a.comm_dist, a.council_dist, a.fire_comp, a.police_prec
FROM
(SELECT n.neighborhood,
COUNT(ps.sale_id) as nr_property_sales,
COUNT(DISTINCT c.community_district_id) comm_dist,
COUNT(DISTINCT cd.city_council_district_id) council_dist,
COUNT(DISTINCT f.fire_company_id) fire_comp,
COUNT(DISTINCT pp.police_precinct_id) police_prec,
RANK() OVER (ORDER BY COUNT(ps.sale_id) DESC)
FROM property_sales AS ps
JOIN properties p ON ps.property_id=p.property_id
JOIN neighborhoods n ON n.neighborhood_id=p.neighborhood_id
JOIN community_districts c ON p.community_district_id=c.community_district_id
JOIN council_districts cd ON p.city_council_district_id=cd.city_council_district_id
JOIN fire_companies f ON p.fire_company_id=f.fire_company_id
JOIN police_precincts pp ON p.police_precinct_id=pp.police_precinct_id
GROUP BY n.neighborhood_id
ORDER BY nr_property_sales DESC) a
WHERE a.rank<=5;
```

What are the distances (euclidean) between property and fire companies, police districts and school districts?

This will offer an insight on the relationship between the sale price and the distance between the property and public services or infrastructure. This will be helpful for the company to improve their future investment strategies.

```
CREATE OR REPLACE FUNCTION calculate_distance(lat1 float, lon1 float, lat2 float, lon2 float, units varchar)
RETURNS float AS $dist$
 DECLARE
```

```
 dist float = 0;
    radlat1 float;
    radlat2 float;
    theta float;
    radtheta float;
  BEGIN
    IF lat1 = lat2 OR lon1 = lon2
      THEN RETURN dist;
    ELSE
      radlat1 = pi() * lat1 / 180;
      radlat2 = pi() * lat2 / 180;
      theta = lon1 - lon2;
      radtheta = pi() * theta / 180;
      dist = sin(radlat1) * sin(radlat2) + cos(radlat1) * cos(radlat2) * cos(radtheta);

      IF dist > 1 THEN dist = 1; END IF;

      dist = acos(dist);
      dist = dist * 180 / pi();
      dist = dist * 60 * 1.1515;

      IF units = 'K' THEN dist = dist * 1.609344; END IF;
      IF units = 'N' THEN dist = dist * 0.8684; END IF;

      RETURN dist;
    END IF;
  END;
$dist$ LANGUAGE plpgsql;
```

- ***The euclidean distances between property and fire companies coordinates***

```
SELECT b.property_id, b.fire_company_id, calculate_distance(lat1,lon1,lat2,lon2,'M') AS eud_distance
FROM
(SELECT p.property_id,f.fire_company_id,f.fire_comp_address_id, a1.lat as lat1, a1.lon as lon1,a.lat as lat2, a.lon as
lon2
FROM fire_companies f JOIN properties p
ON f.fire_company_id= p.fire_company_id
JOIN addresses a on a.address_id=f.fire_comp_address_id
JOIN addresses a1 on a1.address_id=p.address_id) b
ORDER BY eud_distance;
```

- ***The euclidean distances between property and fire companies coordinates***

```
SELECT b.property_id, b.school_district_id, calculate_distance(lat1,lon1,lat2,lon2,'M') AS eud_distance
FROM
(SELECT p.property_id,s.school_district_id,s.school_dist_address_id, a1.lat as lat1, a1.lon as lon1,a.lat as lat2, a.lon
as lon2
FROM school_districts s JOIN properties p
ON s.school_district_id= p.school_district_id
```

```
JOIN addresses a on a.address_id=s.school_dist_address_id
JOIN addresses a1 on a1.address_id=p.address_id) b
ORDER BY eud_distance;
```

- ***The euclidean distances between property and police district coordinates***

```
SELECT b.property_id, b.police_precinct_id, calculate_distance(lat1,lon1,lat2,lon2,'M') AS eud_distance
FROM
(SELECT p.property_id,pp.police_precinct_id,pp.police_prct_address_id, a1.lat as lat1, a1.lon as lon1,a.lat as lat2,
a.lon as lon2
FROM police_precincts pp JOIN properties p
ON pp.police_precinct_id= p.police_precinct_id
JOIN addresses a on a.address_id=pp.police_prct_address_id
JOIN addresses a1 on a1.address_id=p.address_id) b
ORDER BY eud_distance;
```

Tools Used

The tools used in this project include Python, PostgreSQL, PgAdmin, and Metabase. Initially, we used Python to clean up the data and organize it into different tables for our database. We then connected to our PostgreSQL server through Python in order to create the tables in the database and ingest the data accordingly. Once the database was all set up and the data ingested, we used PostgreSQL to query the data and conduct multiple analyses that would generate valuable business insights. Finally, we used Metabase for visualization and created a dashboard that will be especially valuable to C-level executives and non-technical teams. So, let's break this down into the interaction on the data analyst level vs. the C level suite.

As for the data analyst level, there are many analysts in the real estate business that would benefit from the relational database our team has built throughout the course of this project. Let's take, for instance, a Business Development Analyst who is responsible for bringing in new business for a Property management company. Business development analysts care about the prospects they can bring in. Therefore, for them, an analysis with direct queries would be the most useful. As they are more technical and want to know specific details, the way that they would interact with our database system would be directly through PostgreSQL and PgAdmin. For self-serve reporting purposes, we devised a dashboard for the analysts to interact with on Metabase. Now let's now look into how this database will be useful for property management executives. The report that will be sent to the C-suite is a quarterly and annual one. Executives typically care about high level, aggregated metrics presented in a clear, comprehensive manner. Based on this, we suggest using more visuals when reporting to executives and, therefore, we also built a dashboard using Metabase to illustrate how such a report could look like. Details on these dashboards will be explained further in the paper.

Redundancy and Performance, Cloud vs On-prem

When it comes to the redundancy and performance considerations, we have taken multiple factors into account. Specifically, when it comes to data redundancy, it means that there are multiple copies of the same data in the database. The redundancy problem arises when the database is not normalized. Some problems caused due to redundancy are: insertion anomaly, deletion anomaly, and updation anomaly. Specifically with our database, it is in relational DBMS form and also normalized so that we could avoid many data redundancy issues. For example,we created a table called “owns” that connects the owners table to the property sold. By doing this, we would be getting rid of any redundancies like if an owner owned more than one property. Similarly, the building_class_category_combinations table is supposed to be another way of our database normalizing the redundancies. This table gives us a way to link the building classes to the building class categories without having them repeat within the same table.

As for the performance considerations, the team might want to invest in something like Amazon Redshift, Microsoft Azure or Google Cloud Platform for data storage. Big data storage like these enables us to store and sort our data in such a way that it can easily be accessed, used and processed by applications and services that we are working on. In addition, cloud storage will enable us to be able to flexibly scale as required.

Metabase Dashboards

For this project, we devised two dashboards: one self-serve reporting one for analysts, and a general C-level suite dashboard. The purpose of the analyst self-serve reporting dashboard is to allow the analyst to directly interact with our database and enable them to run complex queries to find the information they require. This dashboard provides a template for them to work with that answers their most important questions, and allows for additional edits to be made and customized as the scope of the database increases in the future. Our analyst dashboard includes components such as trend of monthly sales and month-to-month changes, sold properties by building class categories, average sales price by date etc. The analyst dashboard can be found here:
http://f19server.apan5310.com:3204/public/dashboard/b3851a7c-337f-4d25-86f2-9851467c6cc5

The second dashboard we devised was one for the C-level executives. This dashboard will enable the executives of the property management company to interact with the data and receive important information on investment opportunities within the Brooklyn area. The narrative for the C-level dashboard includes providing a high level overview of important metrics from the database such as the number of properties, average square footage of properties sold etc, followed by a general overview of the best performing neighborhoods with associated metrics.

Then, to meet our goal of providing possible investment opportunities insights, we focus on two specific neighborhoods that are investment-promising: one that is well-known as being top performing, and one that is up and coming. The C-level dashboard can be found here: http://f19server.apan5310.com:3204/public/dashboard/d655cd67-6747-472d-906c-d18f3ad6e56d

Conclusion

To conclude, the outcomes of this project enabled our client, a property management company in New York City, to make data-driven, better-informed decisions regarding potential property investments and business development and marketing strategies.

Using RDMS not only helped us create an efficient and seamless way for the client to store and interact with the data in order to generate business insights, but it was also set up in a scalable way, enabling the client to ingest newly acquired data at any time into the database. Through our analysis, we were able to show the client how the new system can be used to generate actionable insights that can help inform the decision-making of both analysts and C-level executives through direct querying capabilities and interactive dashboards. Overall, our analysis proved to be valuable in a few key areas:

- better understanding the typology of popular properties as relating to internal and external factors
- identifying trends in property sales throughout different seasons and months of the year
- determining potential property investment opportunities based on location, price, surrounding amenities (schools, police precincts), etc.

Based on this, the new PostgreSQL database, processes, and analyses we have set up will enable our client to easily and faster generate actionable insights from property sales data in New York. These insights will help inform and support the decision-making of internal stakeholders when it comes to upcoming property investments and acquisition, business development, and marketing strategies.

References

Brooklyn Home Sales Data 2003 - 2017:
https://www.kaggle.com/tianhwu/brooklynhomes2003to2017

Police Precinct Data: https://www1.nyc.gov/site/nypd/bureaus/patrol/precincts-landing.page

Fire Companies Data:
https://data.cityofnewyork.us/Public-Safety/Brooklyn-Fire-Stations/7bi8-bdu5

Brooklyn School Data:
https://www.neighborhoodscout.com/ny/brooklyn/schools

Brooklyn Community Districts Data:
https://communityprofiles.planning.nyc.gov/

Address Geocoding:
https://geocoding.geo.census.gov/geocoder/locations/addressbatch?form

Team 4 Github Repo:
https://github.com/ergetamuca/sql_project_team4