New York Housing Market

Deepak Nallapaneni (50524851)
Engineering Science Data Science
University at Buffalo
Buffalo, New York
dnallapa@buffalo.edu

Akhila Rachuri (50540669)
Engineering Science Data Science
University at Buffalo
Buffalo, New York
rachuri@buffalo.edu

Gowtham Kalluri (50537244)
Engineering Science Data Science
University at Buffalo
Buffalo, New York
kalluri2@buffalo.edu

***Abstract*—This project analyzes the dataset of real estate listings from New York, using SQL databases and queries. The focus is on obtaining useful information by employing SQL queries to examine different types of properties, their prices, sizes, and locations. The goal is to provide essential insights to real estate agents, prospective buyers, and market researchers, aiming to clarify New York's real estate market and facilitate informed investment decisions. This work demonstrates the utility of SQL in extracting valuable information from extensive real estate data, aiding individuals in making informed decisions regarding city properties.**

Index Terms: Real Estate Listings, SQL Databases

I. Introduction

This project presents an in-depth analysis of a New York real estate database, constructed and queried using SQL. It aims to uncover valuable insights from property listings, capturing the complexity of the city's real estate dynamics. The database design adheres to rigorous normalization standards to ensure data integrity and efficient query performance. Our research provides stakeholders with actionable intelligence, enhancing decision-making in the property market. The integration of this database with a web interface showcases the practical application of our findings in a user-friendly manner.

II. Dataset

The ”New York Housing Market” dataset encapsulates an array of properties in the New York City area for the year 2019. This dataset features a wealth of attributes for each listing, including broker information, property type, price, number of bedrooms and bathrooms, property size, and precise location details. It serves as a vital resource for our analysis and modeling efforts, offering a window into the property market of New York City. Leveraging this dataset, we aim to discern prevalent trends across various neighborhoods, comprehend pricing patterns, and identify the property features that attract buyers and investors.

III. Tasks Performed

A. Milestone 1 and 2

In the initial phase of our project, Milestone 1, we concentrated on designing a robust database schema to house data from the New York housing market. This involved creating six well-structured tables—Broker, PropertyType, Area, Location, Address, and Properties. Each table was meticulously crafted with a primary key to ensure data integrity and streamline data operations. Attributes were chosen to accurately represent and uniquely identify details such as broker titles, property types, area descriptions, and specific location data. This setup provided a solid foundation for efficient data indexing and relationship management between tables.

Advancing into Milestone 2, our efforts shifted towards refining the database's functionality and ensuring its adherence to normalization standards. We conducted a thorough analysis to confirm that all tables were in compliance with Boyce-Codd Normal Form (BCNF), analyzing functional dependencies and optimizing the schema to prevent data anomalies and redundancy. Concurrently, we finalized the Entity-Relationship (ER) diagram to visually depict the relationships and constraints within our database, which was crucial for understanding and managing the complex interactions among the data.

Furthermore, Milestone 2 was important in demonstrating the practical application of the database. We developed and executed a variety of SQL queries, including four complex SELECT queries utilizing different SQL operations such as GROUP BY, JOIN, and sub-queries. Additionally, we executed queries for inserting, deleting, and updating operations, which allowed us to manage the dataset dynamically. During this stage, we also identified and addressed performance issues in three specific queries. By utilizing PostgreSQL's EXPLAIN tool, we analyzed these queries, identifying inefficiencies and devising strategies to optimize their execution.

To enhance user interaction and demonstrate the database's capabilities, we built a running website that linked directly to our database. This website served as a platform to visualize and interact with the data through various queries, providing a user-friendly interface to display and analyze the query results.

B. BCNF Status of New York Housing Market Database Tables and Functional Dependencies

For the New York Housing Market dataset, our database design strategy involved crafting a robust and efficient structure through the decomposition of data into six specifically defined tables. Each table was designed to manage distinct aspects of the dataset, with careful attention to the principles of normalization, particularly Boyce-Codd Normal Form (BCNF).

Each table within our database was designed to include exactly one functional dependency, where the primary key uniquely determines all other attributes. This ensures that each table strictly adheres to BCNF, eliminating any need for further decomposition. Such a design enhances data integrity and simplifies database operations by minimizing redundancy and dependency complications.

The Broker Table uses brokerid as the primary key, which uniquely determines the brokertitle. This clear dependency ensures the table is in BCNF, optimizing data retrieval and update processes. Similarly, the PropertyType Table has typeid as its primary key, singularly defining the typename. This setup not only maintains data clarity but also prevents data duplication, ensuring efficient management of property types.

1) Broker Table:

- **brokerid** → brokertitle

2) PropertyType Table:

- **typeid** → typename

3) Area Table:

- **areaid** → administrative_area_level_2, sublocality, formattedaddress

4) Location Table:

- **locationid** → latitude, longitude, locality, areaid

5) Address Table:

- **addressid** → streetname, mainaddress, locationid

6) Properties Table:

- **propertyid** → brokerid, typeid, price, beds, baths, propertysqft, addressid

C. Functional Dependencies

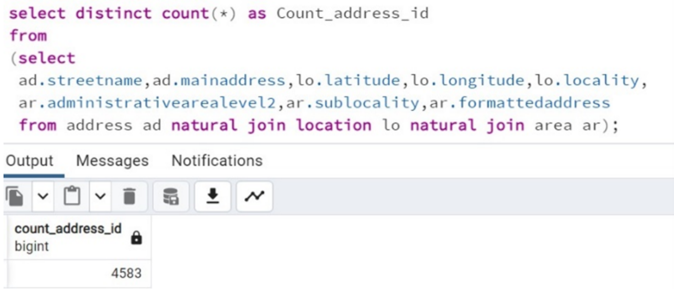
These dependencies ensure that each table is in BCNF because the left-hand side of each dependency is a superkey of the table. This transformation allowed us to eliminate transitive dependencies, reduce redundancy, and minimize the potential for update anomalies.

By ensuring that each table contains only one functional dependency and that each adheres to BCNF, we have optimized our database for high performance and scalability. This approach not only streamlines database management but also significantly enhances the capability of our database system to handle complex queries and extensive data operations effectively in the real estate sector.

D. Decomposition into Tables

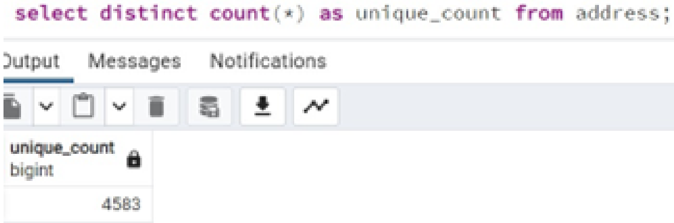
Initially, our comprehensive New York Housing Market database encompassed a singular, aggregated table that attempted to consolidate information pertinent to Addresses, Locations, and Areas. However, this monolithic structure did not adhere to the stringent standards set by Boyce-Codd Normal Form (BCNF). The presence of multiple functional dependencies within this singular table compromised the normalization process. Specifically, the functional dependencies in the combined table took a form where multiple non-key attributes were dependent on non-prime attributes, indicating a clear deviation from BCNF.

Upon decomposing the features into the Address, Location, and Area tables, we have achieved a design where each table has a single primary key that uniquely determines the other attributes within that table. The consistency of the addressid count before and after the decomposition, as indicated by the query results showing the same count, confirms that the integrity of the data was maintained during the process. The natural join of the Address, Location, and Area tables did not result in any loss of information or introduce any anomalies, further validating the success of the decomposition.



```
select distinct count(*) as Count_address_id
from
(select
 ad.streetname,ad.mainaddress,lo.latitude,lo.longitude,lo.locality,
 ar.administrativearealevel2,ar.sublocality,ar.formattedaddress
 from address ad natural join location lo natural join area ar);
```

```
count_address_id
bigint
            4583
```



```
select distinct count(*) as unique_count from address;
```

```
unique_count
bigint
        4583
```

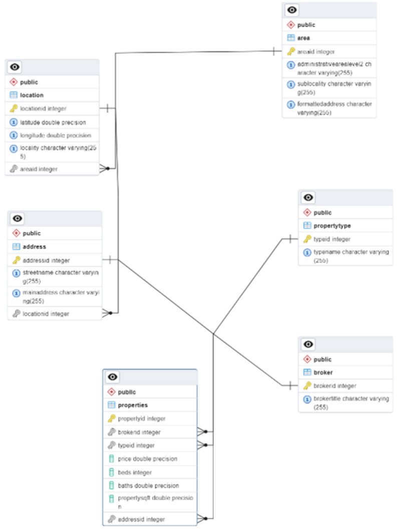
IV. E/R Diagram

A. Entities

- **Broker**: Holds information about brokers. Attributes include brokerid (primary key), brokertitle (name or title of the broker).
- **PropertyType**: Describes various types of properties available. Attributes include typeid (primary key), typename (description of the property type).
- **Area**: Represents geographical areas. Attributes include areaid (primary key), administrative_area_level_2, sublocality, formattedaddress.
- **Location**: Stores specific geographic coordinates and locality information. Attributes include locationid (primary key), latitude, longitude, locality, areaid (foreign key).
- **Address**: Contains detailed information about property addresses. Attributes include addressid (primary key), streetname, mainaddress, locationid (foreign key).
- **Properties**: Central entity detailing individual property listings. Attributes include propertyid (primary key), brokerid (foreign key), typeid (foreign key), price, beds, baths, propertysqft, addressid (foreign key).

B. Relationships

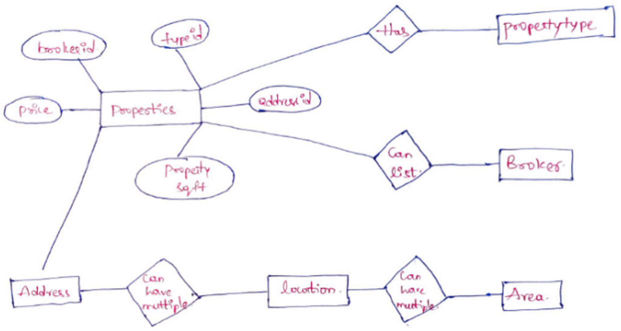
- **Broker-Properties**: A one-to-many relationship indicating that a broker can list multiple properties, but each property is listed by one broker.
- **PropertyType-Properties**: A one-to-many relationship showing that a property type can classify multiple properties, but each property has one specific property type.


public
area
areaid integer
administrativearealevel2 character varying(255)
sublocality character varying(255)
formattedaddress character varying(255)
public
location
locationid integer
latitude double precision
longitude double precision
locality character varying(255)
areaid integer
public
propertytype
typeid integer
typename character varying (255)
public
address
addressid integer
streetname character varying(255)
mainaddress character varying(255)
locationid integer
public
broker
brokerid integer
brokertitle character varying (255)
public
properties
propertyid integer
brokerid integer
typeid integer
price double precision
beds integer
baths double precision
propertysqft double precision
addressid integer

- **Area-Location**: A one-to-many relationship where an area can include multiple locations, but each location is situated in one area.
- **Location-Address**: A one-to-many relationship where each location can have multiple addresses associated with it, but each address corresponds to one location.
- **Address-Properties**: A one-to-many relationship suggesting that an address can have multiple properties associated with it, especially in the case of apartment buildings or multi-property lots, but each property has one unique address.

C. High-level Design: ER-Diagram

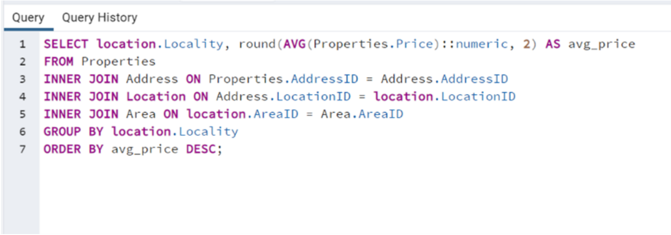
Using diamonds for relationships, rectangles for tables, and ovals for attributes in a high-level ER diagram aids in visually representing the database schema. It enhances clarity, facilitates communication among stakeholders, and serves as documentation for future reference. Additionally, it assists in analyzing the database design for optimization and improvement. The ER outlines a real estate database with tables for Properties, Location, Area, Broker, and PropertyType. Properties are central, linked to Brokers who manage them and PropertyTypes defining their category. Areas are connected to Locations, which in turn are associated with Properties, indicating where each property is situated. Diamonds represent relationships, indicating that Properties can have multiple associated records in Location, Area, and other tables.



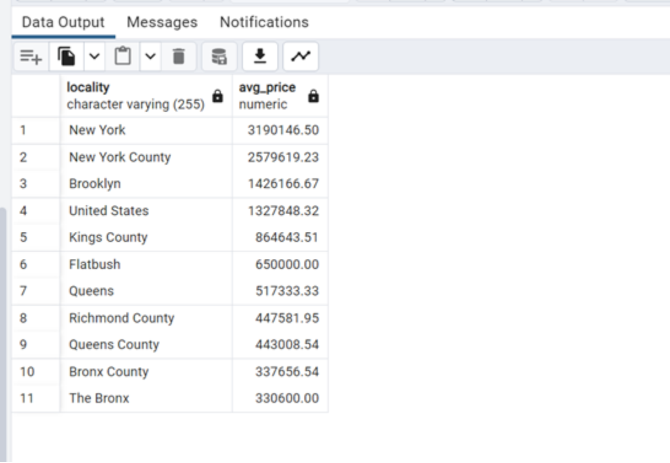
V. Queries

A. Select Queries

1) Average Property Price for Each Locality: This SQL query calculates the average property prices for each locality by joining the Properties table with the Address, Location, and Area tables. It then groups the results by locality and rounds the average prices to two decimal places. The output is a summary of localities and their corresponding average property prices.

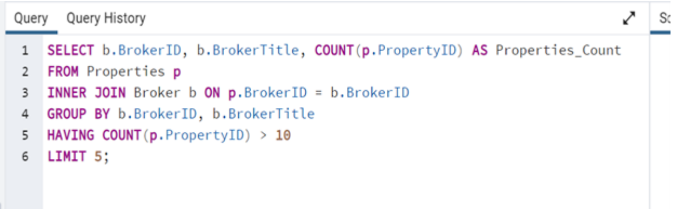


```sql
SELECT location.Locality, round(AVG(Properties.Price)::numeric, 2) AS avg_price
FROM Properties
INNER JOIN Address ON Properties.AddressID = Address.AddressID
INNER JOIN Location ON Address.LocationID = location.LocationID
INNER JOIN Area ON location.AreaID = Area.AreaID
GROUP BY location.Locality
ORDER BY avg_price DESC;
```

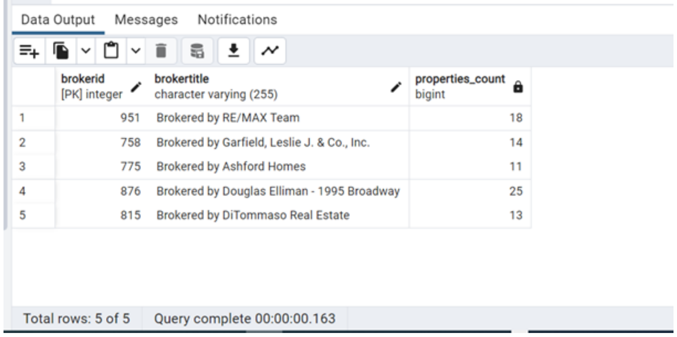


	locality character varying (255)	avg_price numeric
1	New York	3190146.50
2	New York County	2579619.23
3	Brooklyn	1426166.67
4	United States	1327848.32
5	Kings County	864643.51
6	Flatbush	650000.00
7	Queens	517333.33
8	Richmond County	447581.95
9	Queens County	443008.54
10	Bronx County	337656.54
11	The Bronx	330600.00

2) Brokers with More Than 10 Properties: This SQL query retrieves the brokerid, brokertitle, and the count of properties listed by each broker. It joins the Properties table with the Broker table on the brokerid, groups the results by brokerid and brokertitle, and filters the results to include only brokers with more than 10 listed properties.



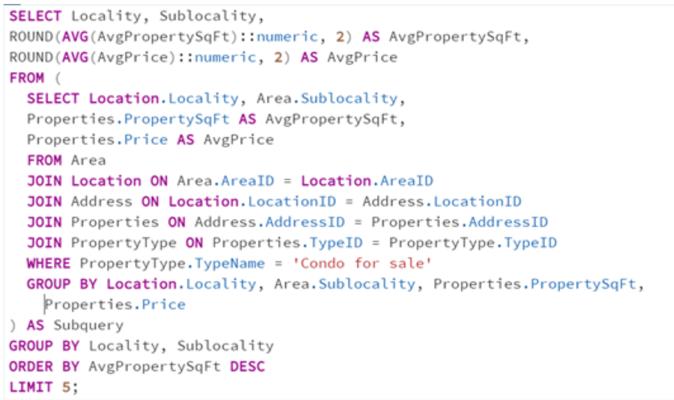
```sql
SELECT b.BrokerID, b.BrokerTitle, COUNT(p.PropertyID) AS Properties_Count
FROM Properties p
INNER JOIN Broker b ON p.BrokerID = b.BrokerID
GROUP BY b.BrokerID, b.BrokerTitle
HAVING COUNT(p.PropertyID) > 10
LIMIT 5;
```



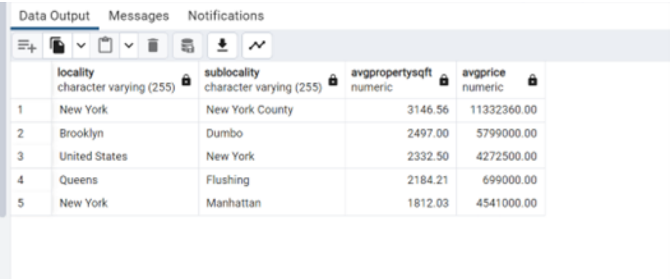
	brokerid [PK] integer	brokertitle character varying (255)	properties_count bigint
1	951	Brokered by RE/MAX Team	18
2	758	Brokered by Garfield, Leslie J. & Co., Inc.	14
3	775	Brokered by Ashford Homes	11
4	876	Brokered by Douglas Elliman - 1995 Broadway	25
5	815	Brokered by DiTommaso Real Estate	13

Total rows: 5 of 5   Query complete 00:00:00.163

3) Top 5 Localities and Sublocalities with Highest Average Property Square Footage and Average Price for Condos for Sale: This SQL query retrieves the locality, sublocality, average property square footage (avgpropertysqft), and average price (avgprice) for condos listed for sale. It first selects relevant data from the database by joining the Area, Location, Address, Properties, and PropertyType tables. Then, it calculates the average property square footage and average price for each locality and sublocality. The results are grouped by locality and sublocality, ordered by average property square footage in descending order, and limited to the top 5 results.

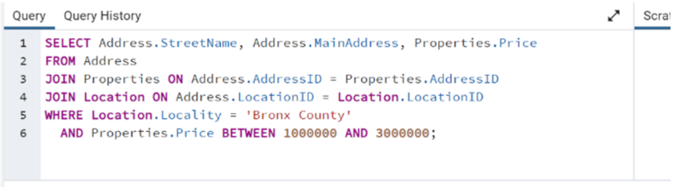


```
SELECT Locality, Sublocality,
ROUND(AVG(AvgPropertySqFt)::numeric, 2) AS AvgPropertySqFt,
ROUND(AVG(AvgPrice)::numeric, 2) AS AvgPrice
FROM (
  SELECT Location.Locality, Area.Sublocality,
  Properties.PropertySqFt AS AvgPropertySqFt,
  Properties.Price AS AvgPrice
  FROM Area
  JOIN Location ON Area.AreaID = Location.AreaID
  JOIN Address ON Location.LocationID = Address.LocationID
  JOIN Properties ON Address.AddressID = Properties.AddressID
  JOIN PropertyType ON Properties.TypeID = PropertyType.TypeID
  WHERE PropertyType.TypeName = 'Condo for sale'
  GROUP BY Location.Locality, Area.Sublocality, Properties.PropertySqFt,
    Properties.Price
) AS Subquery
GROUP BY Locality, Sublocality
ORDER BY AvgPropertySqFt DESC
LIMIT 5;
```

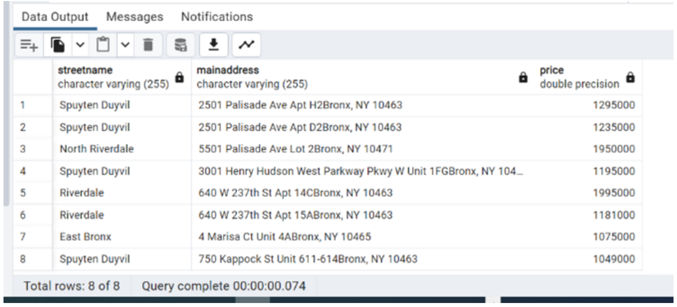


	locality character varying (255)	sublocality character varying (255)	avgpropertysqft numeric	avgprice numeric
1	New York	New York County	3146.56	11332360.00
2	Brooklyn	Dumbo	2497.00	5799000.00
3	United States	New York	2332.50	4272500.00
4	Queens	Flushing	2184.21	699000.00
5	New York	Manhattan	1812.03	4541000.00

4) Properties Available in Desired Locality within Price Range: This SQL query retrieves the street name, main address, and price of properties located in Bronx County with prices between $1,000,000and$3,000,000. It first joins the Address table with the Properties table on the addressid, then joins the Location table on the locationid.



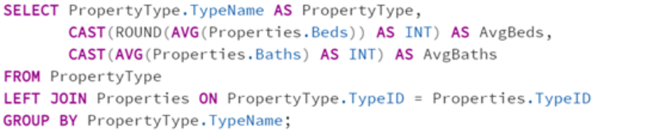
```
SELECT Address.StreetName, Address.MainAddress, Properties.Price
FROM Address
JOIN Properties ON Address.AddressID = Properties.AddressID
JOIN Location ON Address.LocationID = Location.LocationID
WHERE Location.Locality = 'Bronx County'
  AND Properties.Price BETWEEN 1000000 AND 3000000;
```



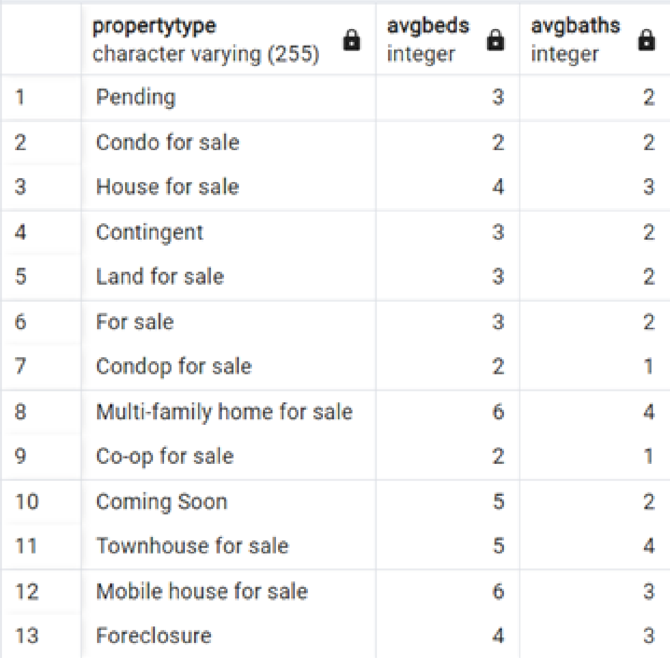
	streetname character varying (255)	mainaddress character varying (255)	price double precision
1	Spuyten Duyvil	2501 Palisade Ave Apt H2Bronx, NY 10463	1295000
2	Spuyten Duyvil	2501 Palisade Ave Apt D2Bronx, NY 10463	1235000
3	North Riverdale	5501 Palisade Ave Lot 2Bronx, NY 10471	1950000
4	Spuyten Duyvil	3001 Henry Hudson West Parkway Pkwy W Unit 1FGBronx, NY 104...	1195000
5	Riverdale	640 W 237th St Apt 14CBronx, NY 10463	1995000
6	Riverdale	640 W 237th St Apt 15ABronx, NY 10463	1181000
7	East Bronx	4 Marisa Ct Unit 4ABronx, NY 10465	1075000
8	Spuyten Duyvil	750 Kappock St Unit 611-614Bronx, NY 10463	1049000

Total rows: 8 of 8   Query complete 00:00:00.074

5) Average Number of Bedrooms and Bathrooms for Each Property Type: This SQL query retrieves the average number of bedrooms (avgbeds) and baths (avgbaths) for each property type (PropertyType). It rounds the average number of bedrooms to the nearest whole number and casts it to an integer. The results are grouped by property type, and the query includes all property types even if there are no corresponding properties.



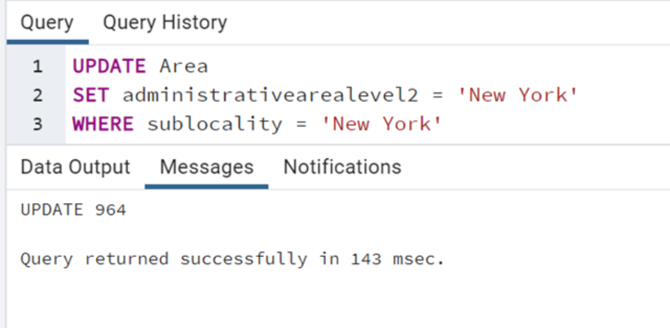
```
SELECT PropertyType.TypeName AS PropertyType,
       CAST(ROUND(AVG(Properties.Beds)) AS INT) AS AvgBeds,
       CAST(AVG(Properties.Baths) AS INT) AS AvgBaths
FROM PropertyType
LEFT JOIN Properties ON PropertyType.TypeID = Properties.TypeID
GROUP BY PropertyType.TypeName;
```



	propertytype character varying (255)	avgbeds integer	avgbaths integer
1	Pending	3	2
2	Condo for sale	2	2
3	House for sale	4	3
4	Contingent	3	2
5	Land for sale	3	2
6	For sale	3	2
7	Condop for sale	2	1
8	Multi-family home for sale	6	4
9	Co-op for sale	2	1
10	Coming Soon	5	2
11	Townhouse for sale	5	4
12	Mobile house for sale	6	3
13	Foreclosure	4	3

B. Update Queries

1) Updating Administrative Area Level 2 Entries: Upon reviewing the database, it was found that the administrativearealevel2 field was storing integers for entries where the sublocality was ’New York’. To correct this and ensure uniformity across the dataset, an update query was executed, changing these values to the string ’New York’. This action was essential for maintaining data integrity, ensuring the accuracy of query results, and preserving the reliability of the system for data analytics and operational processes.

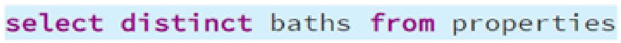


```
UPDATE Area
SET administrativearealevel2 = 'New York'
WHERE sublocality = 'New York'
```

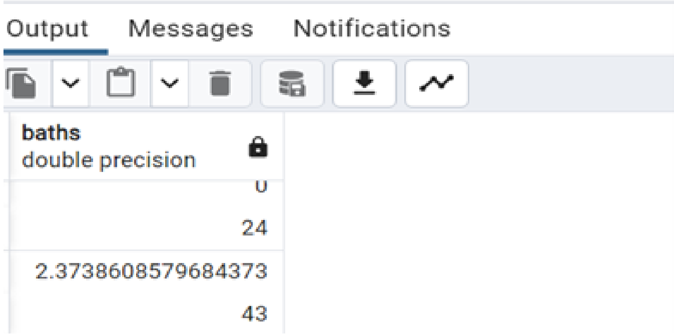
```
UPDATE 964

Query returned successfully in 143 msec.
```

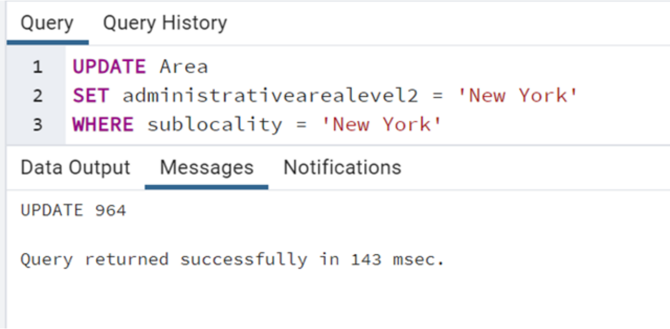
2) Updating Bathroom Counts: Upon observing an unrealistic bathroom count of ‘2.37‘ in our properties database, an update query was executed to revise this figure to a whole number, specifically changing it to ‘2‘. This correction ensures the database accurately mirrors real-life property attributes and maintains data integrity for reliable analysis and reporting.



```
select distinct baths from properties
```



```
baths
double precision
                 0
                24
2.3738608579684373
                43
```



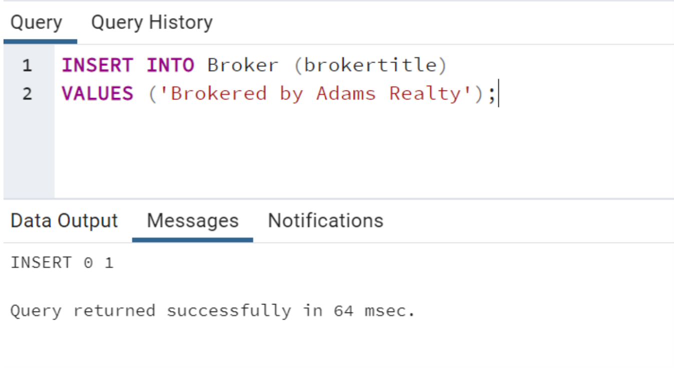
```
UPDATE Area
SET administrativearealevel2 = 'New York'
WHERE sublocality = 'New York'
```

```
UPDATE 964

Query returned successfully in 143 msec.
```

C. Insert Queries

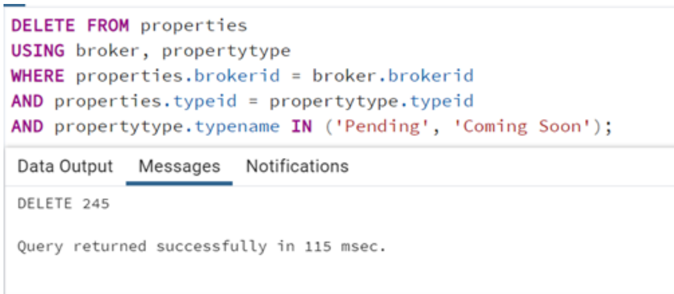
The INSERT statement adds a new record into the ”Broker” table.



```
INSERT INTO Broker (brokertitle)
VALUES ('Brokered by Adams Realty');
```

```
INSERT 0 1

Query returned successfully in 64 msec.
```



```
DELETE FROM properties
USING broker, propertytype
WHERE properties.brokerid = broker.brokerid
AND properties.typeid = propertytype.typeid
AND propertytype.typename IN ('Pending', 'Coming Soon');
```

```
DELETE 245

Query returned successfully in 115 msec.
```

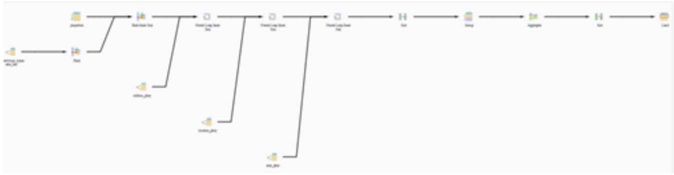
D. Delete Queries

1) Deleting Properties Associated with Brokers with ’Pending’ or ’Coming Soon’ Status: Properties labeled as ’Pending’ or ’Coming Soon’ may indicate outdated or unavailable listings. Deleting these records helps maintain database accuracy and improves the user experience by removing clutter and ensuring users find relevant listings. This action streamlines the platform, focusing attention on available properties and enhancing usability.

VI. Query Execution Analysis with Explain Tool

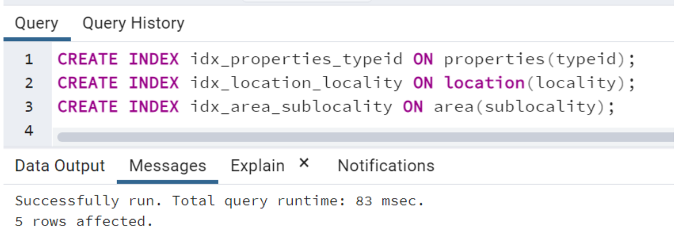
Among the queries analyzed, three were considered problematic for their high execution cost and time consumption.

A. Top 5 Localities and Sublocalities with Highest Average Property Square Footage and Average Price for Condos for Sale



It reduces the execution times. These changes are anticipated to minimize full table scans and The analysis of the execution plan indicates that targeted indexing is the primary opportunity for performance enhancement. By introducing indexes on the properties.typeid for join efficiency, and on location.locality and area.sublocality for faster grouping, the database engine can significantly streamline group by operations. Prior to production deployment, it is crucial to test these optimizations in a staging environment to confirm their effectiveness and to ensure that they do not introduce any negative impacts on the system's performance.

1) Indexing: The addition of indexes to the properties(typeid), location(locality), and area(sublocality) columns has resulted in a measurable performance gain for the query execution time. Prior to indexing, the query execution time was recorded at 99 milliseconds. After the implementation of the indexes, this time was reduced to 73 milliseconds. This reduction by 26 milliseconds indicates a performance improvement of approximately 26.26 percent. The indexes have facilitated more efficient database operations by allowing the query optimizer to rapidly locate and retrieve the relevant rows, thus substantiating the benefit of proper indexing in database performance tuning.

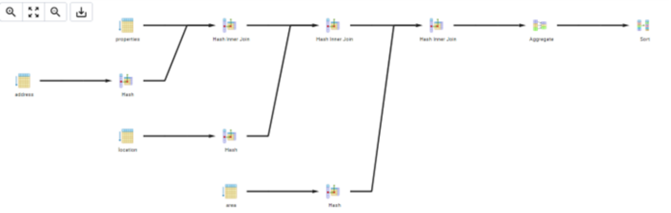


```
CREATE INDEX idx_properties_typeid ON properties(typeid);
CREATE INDEX idx_location_locality ON location(locality);
CREATE INDEX idx_area_sublocality ON area(sublocality);
```

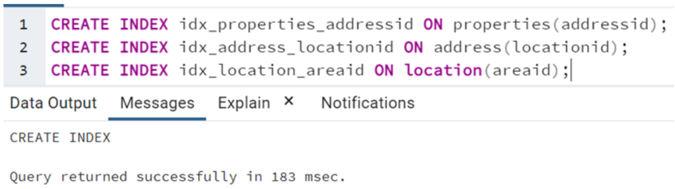
```
Successfully run. Total query runtime: 83 msec.
5 rows affected.
```

B. *Average Property Price for Each Locality*

The query plan indicates that adding indexes on the Properties.AddressID, Address.LocationID, and Location.AreaID columns could improve performance, as the current plan shows sequential scans on these tables. By introducing these indexes, the database can perform more efficient hash joins, likely reducing the execution time of 12.208 milliseconds.



1) Indexing: Post-implementation of the above indexes, the execution time for the query improved from 12.208 milliseconds to 7.340 milliseconds, a reduction of nearly 40 percent.This demonstrates that the indexes have effectively optimized the query's performance by facilitating faster join operations and data retrieval, confirming the significance of proper indexing in database optimization.



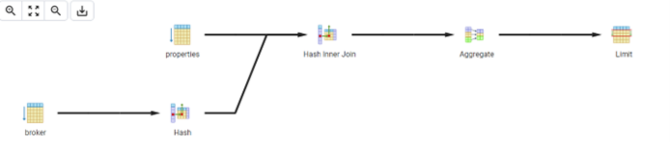
```
CREATE INDEX idx_properties_addressid ON properties(addressid);
CREATE INDEX idx_address_locationid ON address(locationid);
CREATE INDEX idx_location_areaid ON location(areaid);
```

```
CREATE INDEX

Query returned successfully in 183 msec.
```

C. *Brokers with More Than 10 Properties*

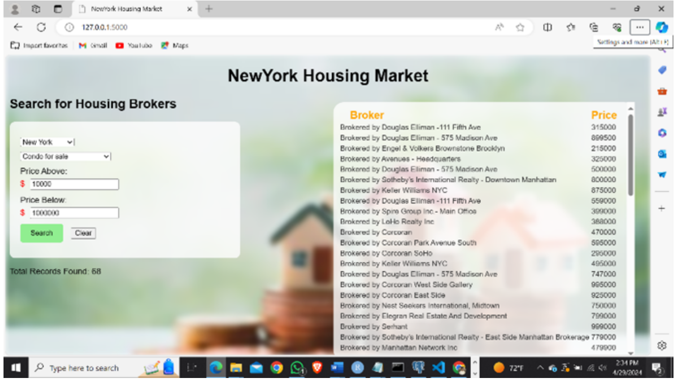
The execution plan reveals that the query performs well, with an execution time of 2.128 milliseconds. However, indexing p.BrokerID could further improve join efficiency. Additionally, the Broker table's brokerid being indexed would enhance the hash join.



1) Indexing: Post-indexing, the query's execution time improved dramatically from 2.128 milliseconds to 0.417 milliseconds. This substantial enhancement underscores the efficacy of indexing on key columns for join operations, demonstrating a nearly fivefold increase in query speed and showcasing the vital role of indexes in database performance optimization.

VII. User-Interface Generation

This project implements a Flask backend to process user inputs of state, property type, and price range. This data is then utilized to dynamically populate a webpage with broker titles and their associated prices, creating a responsive user interface for property search.



NewYork Housing Market

Search for Housing Brokers

New York

Condo for sale

Price Above: $ 10000

Price Below: $ 1000000

Total Records Found: 68

Broker	Price
Brokered by Douglas Elliman -111 Fifth Ave	315000
Brokered by Douglas Elliman - 575 Madison Ave	899500
Brokered by Engel & Volkers Brownstone Brooklyn	215000
Brokered by Avenues - Headquarters	325000
Brokered by Douglas Elliman - 575 Madison Ave	500000
Brokered by Sotheby's International Realty - Downtown Manhattan	800000
Brokered by Keller Williams NYC	875000
Brokered by Douglas Elliman -111 Fifth Ave	559000
Brokered by Spire Group Inc - Main Office	399000
Brokered by LoHo Realty Inc	388000
Brokered by Corcoran	470000
Brokered by Corcoran Park Avenue South	595000
Brokered by Corcoran SoHo	295000
Brokered by Keller Williams NYC	495000
Brokered by Douglas Elliman - 575 Madison Ave	747000
Brokered by Corcoran West Side Gallery	995000
Brokered by Corcoran East Side	925000
Brokered by Nest Seekers International, Midtown	750000
Brokered by Elegran Real Estate And Development	799000
Brokered by Serhant	999000
Brokered by Sotheby's International Realty - East Side Manhattan Brokerage	779000
Brokered by Manhattan Network Inc	479900

The user interface of a ”New York Housing Market” website that allows users to search for housing brokers. This tool is useful for users looking to buy, sell, or rent properties in New York, as it aggregates listings from various brokers, presenting them along with pertinent details like location and price. The interface includes filters to search properties within specific price ranges, potentially making the property search experience more efficient and tailored to user needs.

URL Link: http://127.0.0.1:5000

VIII. REFERENCES

- https://www.w3schools.com/mysql/mysql_select.asp
- Class Slides
- https://www.sqlshack.com/learn-sql-sql-triggers/
- https://www.tutorialspoint.com/sql/sql-indexes.htm
- https://www.tutorialspoint.com/sql/sql-sub-queries.htm
- Data Source: https://www.kaggle.com/datasets/nelgiriyewithana/new-york-housing-market

IX. CONTRIBUTION

Team Member	Contribution
Deepak Nallapaneni	33.3%
Akhila Rachuri	33.3%
Gowtham Kalluri	33.3%