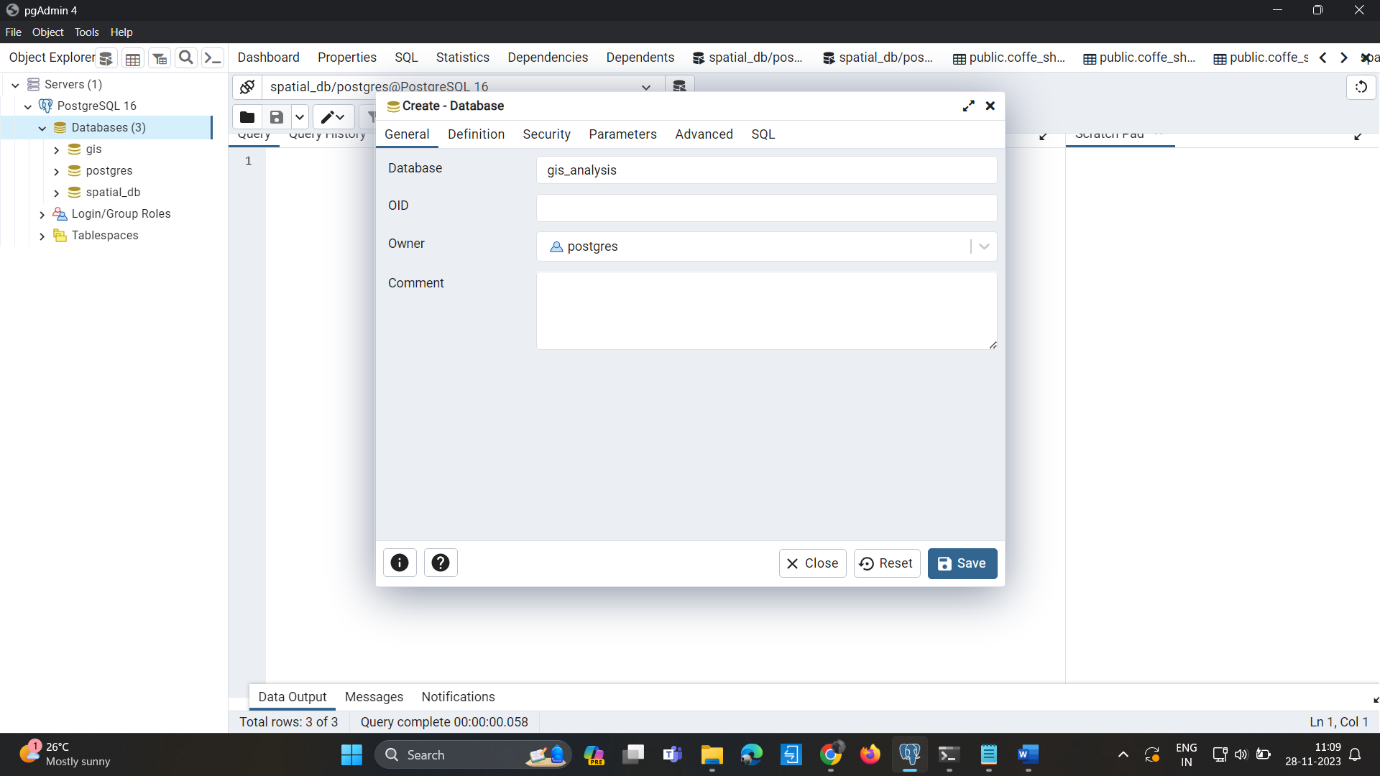
**Geographic Information System (GIS) Analysis of coffee shop**

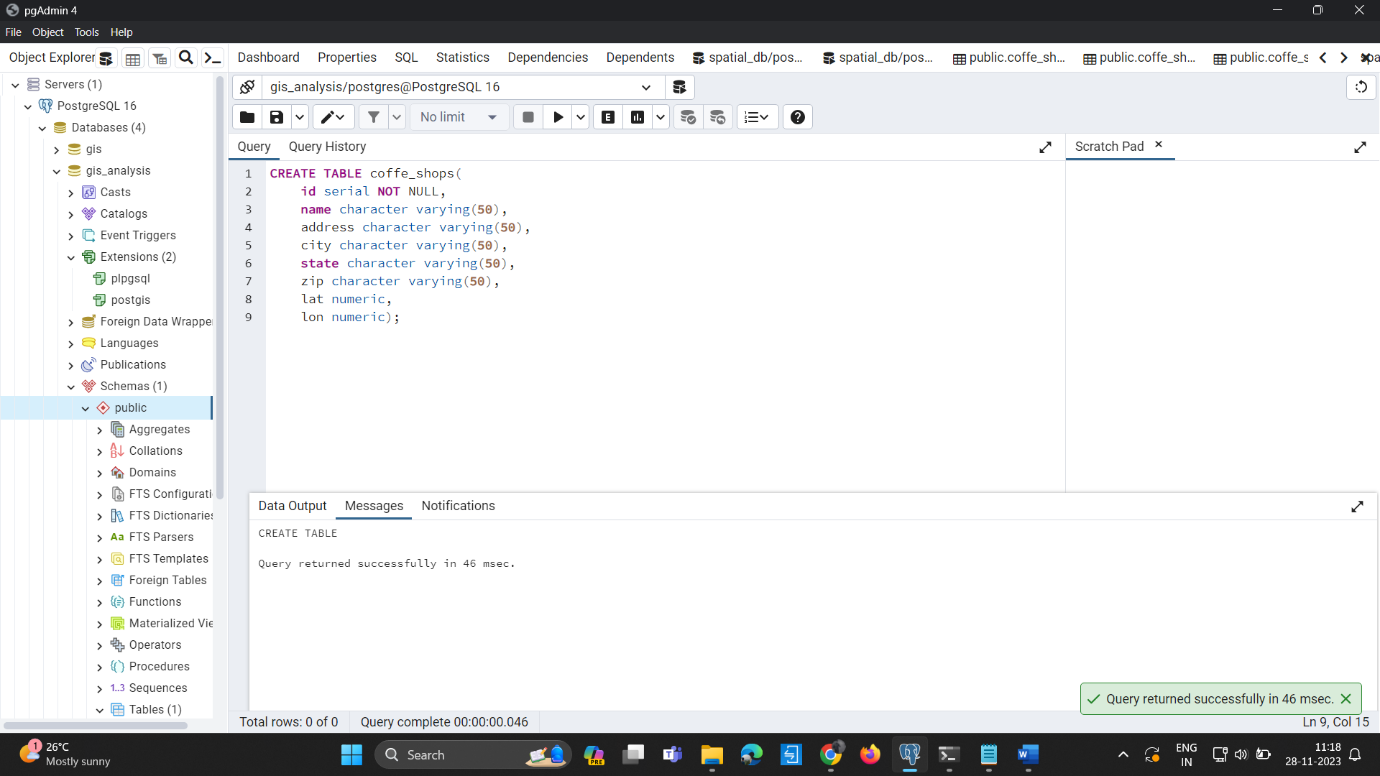
**Team Members:**

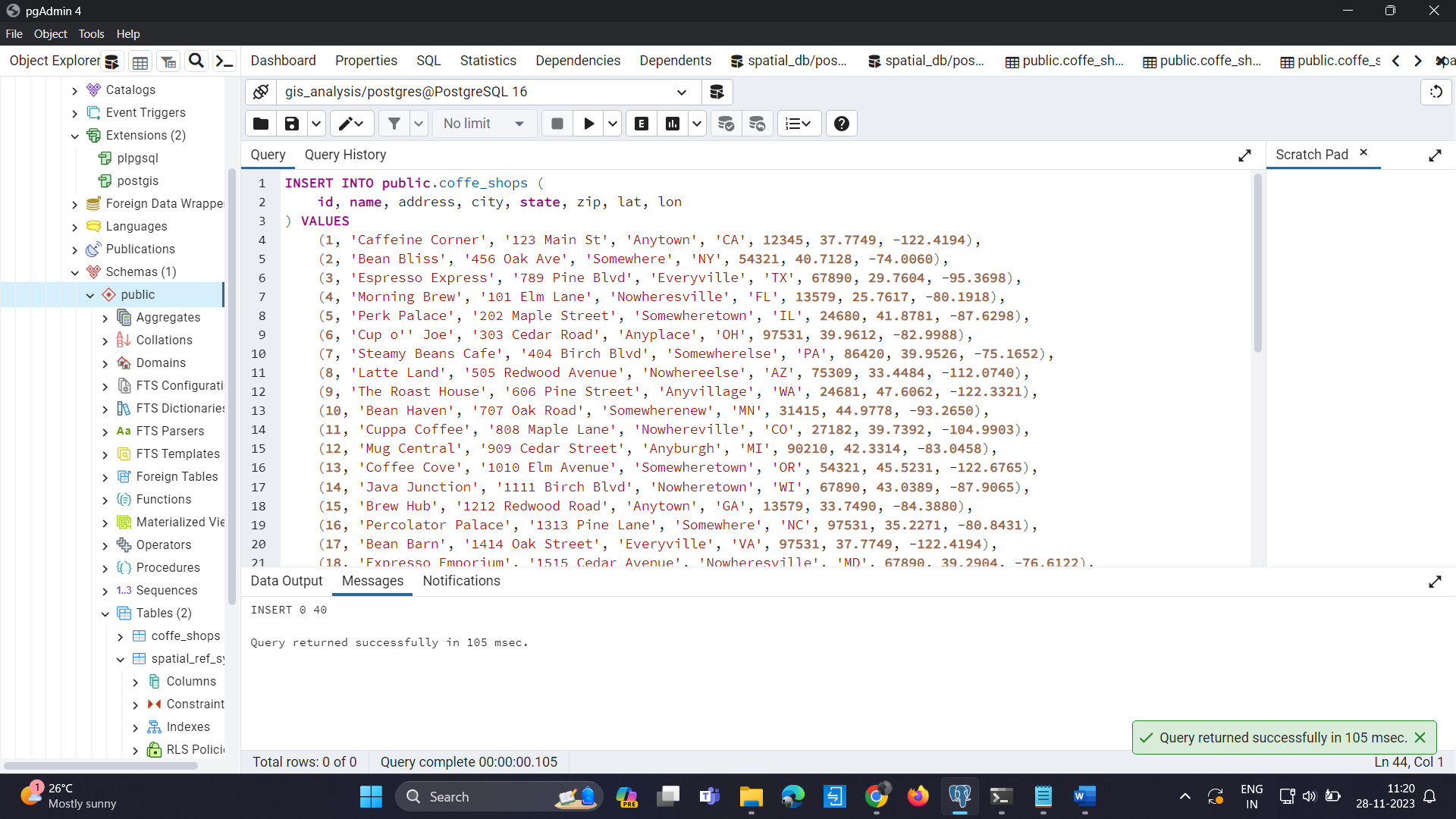
1. **Sandeep Malyala** sm86925n@pace.edu
2. **Pravallika Challa** pc94436n@pace.edu
3. **Chetan Kumar Reddy Vaddi** CV33511N@pace.edu
4. **Ramadugu Bhavitha** br31457n@pace.edu
5. **Sai Kiran Nunna** sn01516n@pace.edu

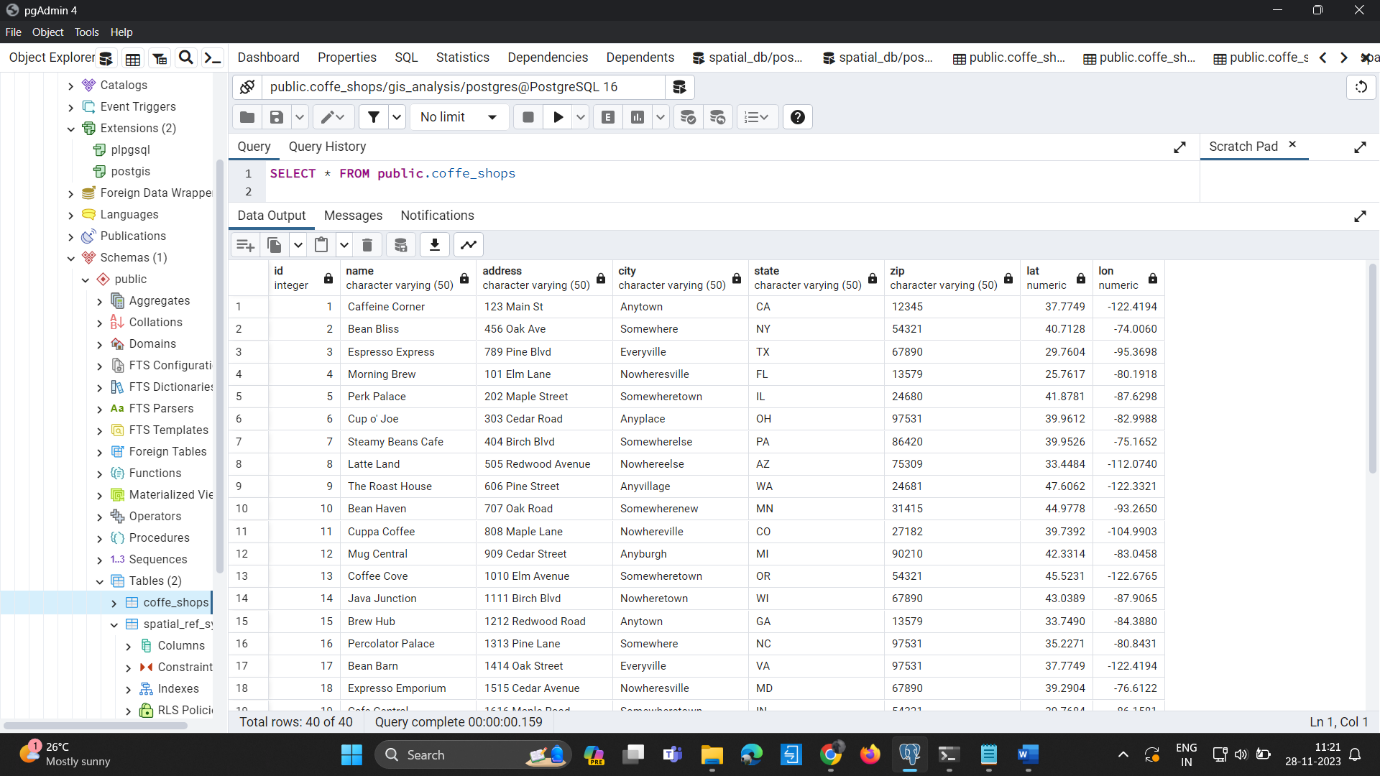
**Step By Step process:**

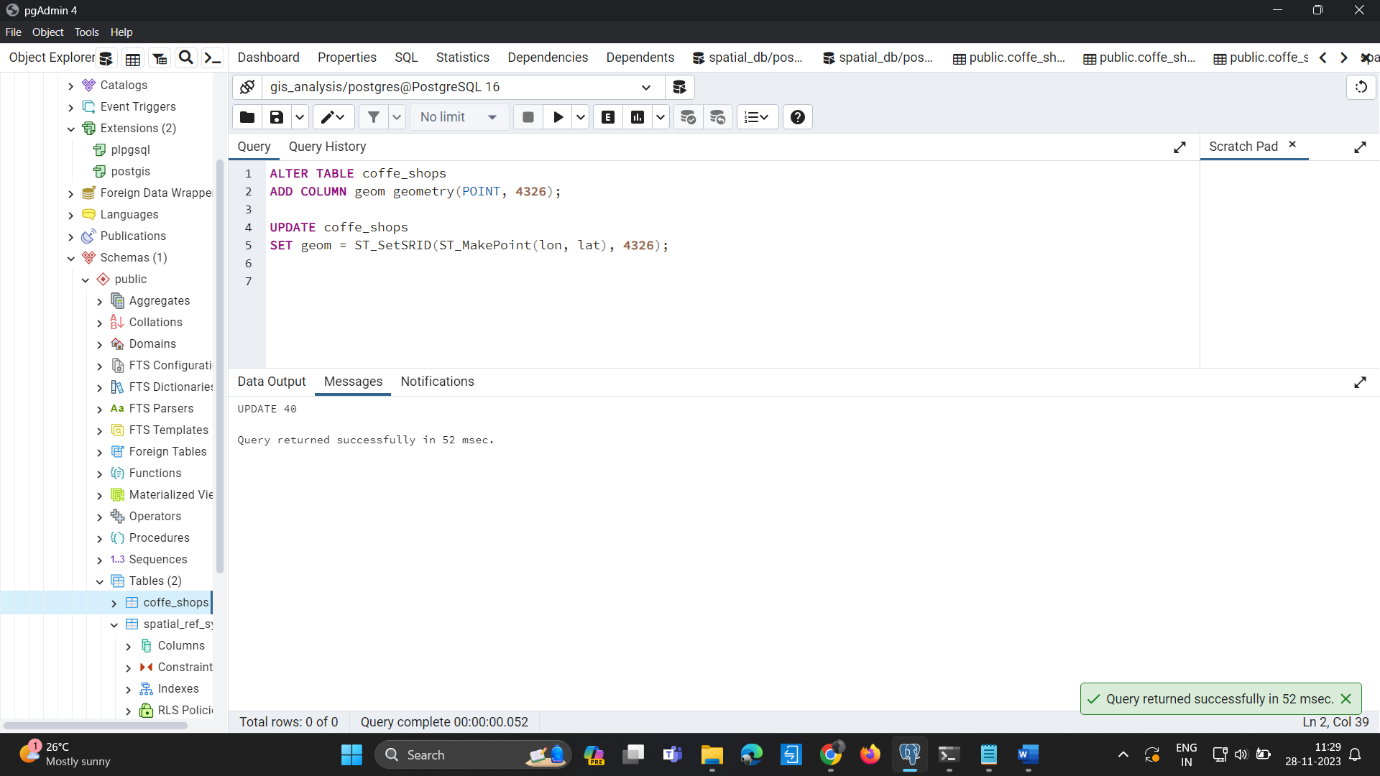
1. **Create Database:**



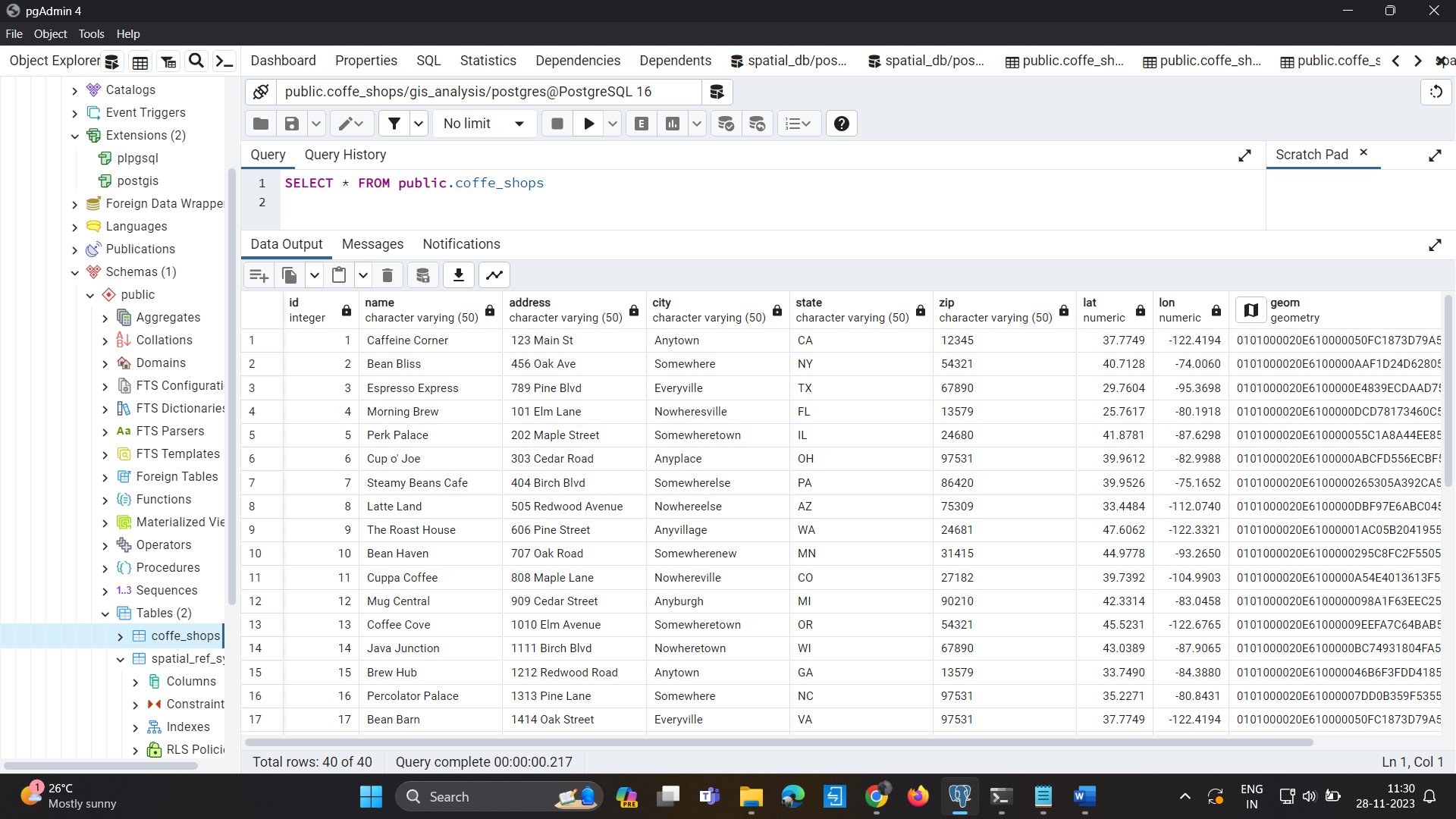
1. **Create Table:**
2. **Insert data into coffe\_shop table:**



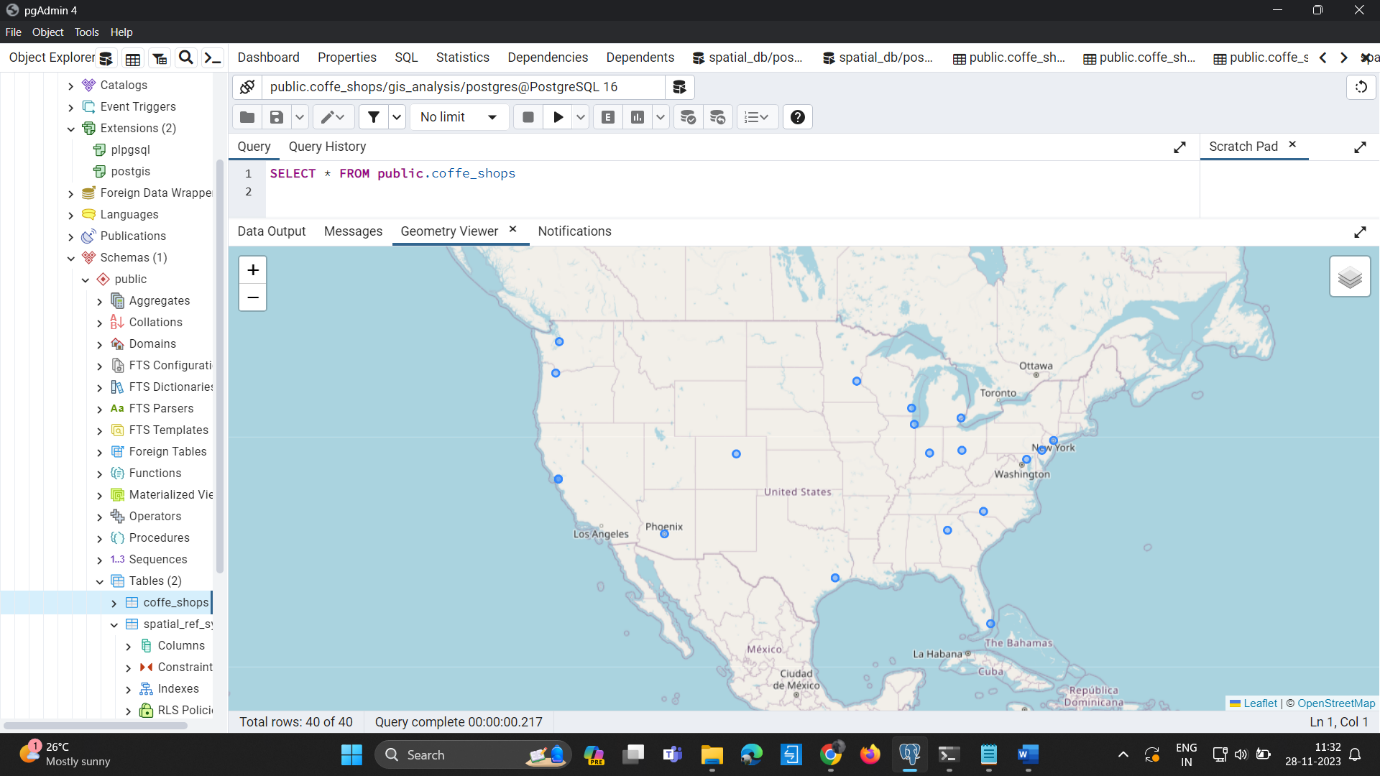
1. **View data:**
2. **Insert geom column query:**



1. **Geom Data:**



1. **Map\_view**



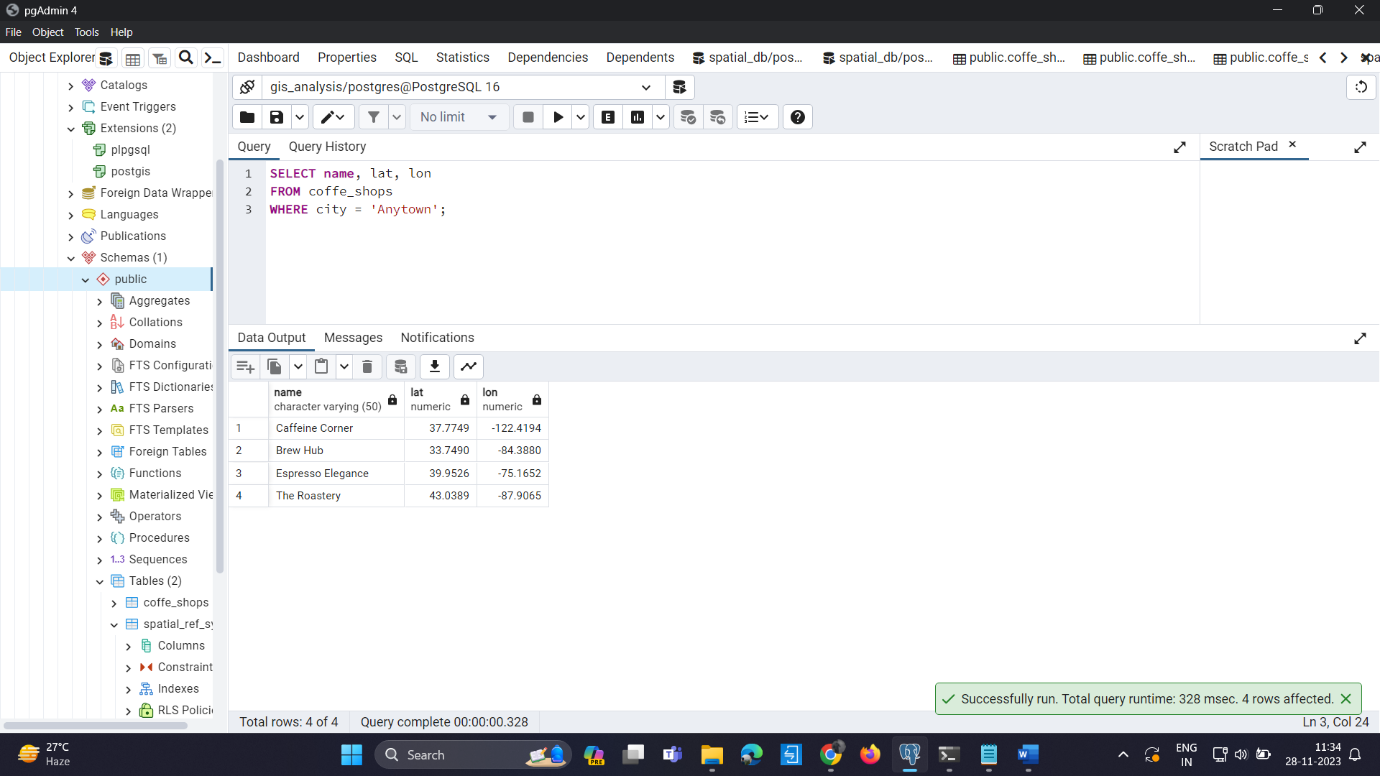
**CS 223 Practical Answers:**

* **Retrieve Locations of specific features:**

**Query:**

SELECT name, lat, lon

FROM coffe\_shops

WHERE city = 'Anytown';

* **Calculate Distance between points:**

**Query:**

SELECT

id1,

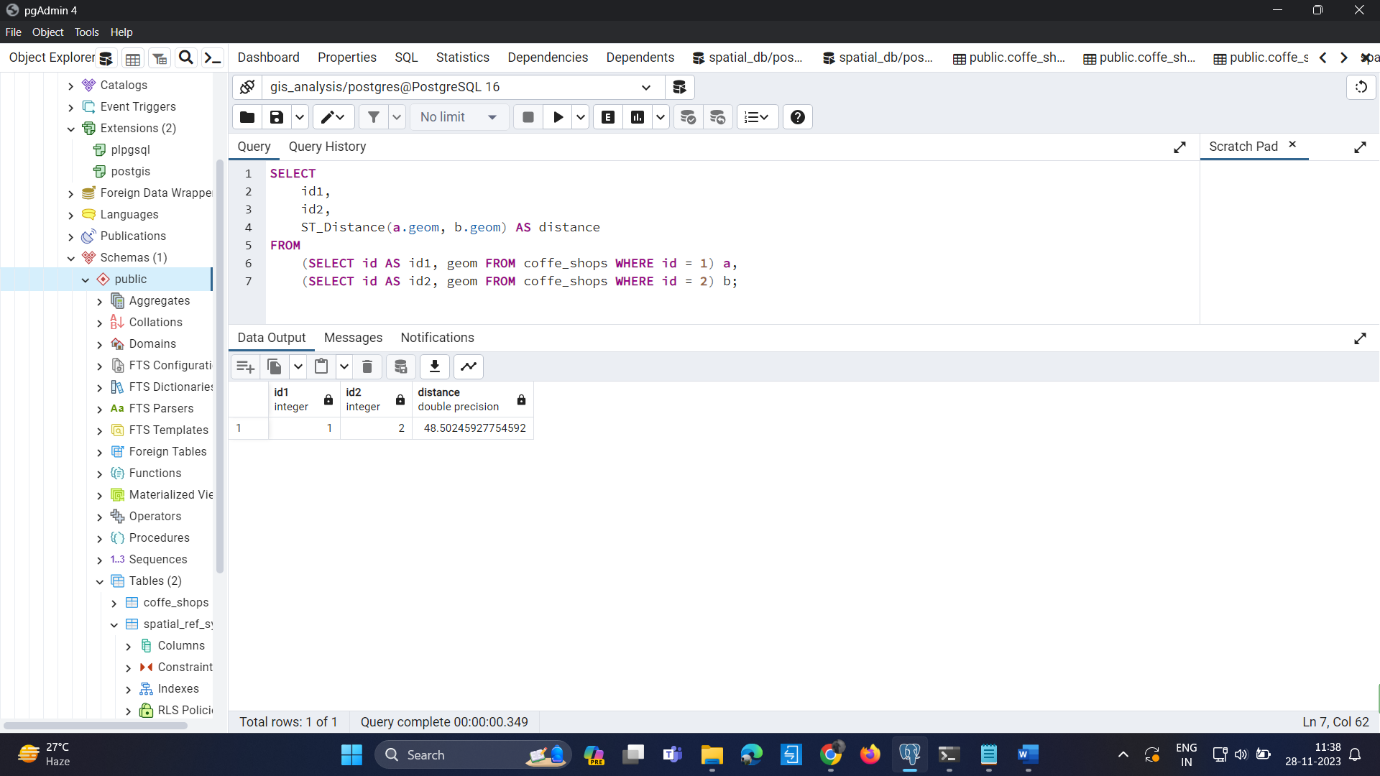
id2,

ST\_Distance(a.geom, b.geom) AS distance

FROM

(SELECT id AS id1, geom FROM coffe\_shops WHERE id = 1) a,

(SELECT id AS id2, geom FROM coffe\_shops WHERE id = 2) b;



* **Calculate Areas of Interest (specific to each group):**

**Query**:

1. **Add Group Column:**

**Query1)**

ALTER TABLE coffe\_shops

ADD COLUMN group\_id INTEGER;

**Query2)**

UPDATE coffe\_shops

SET group\_id = 1

WHERE id IN (1, 2, 3);

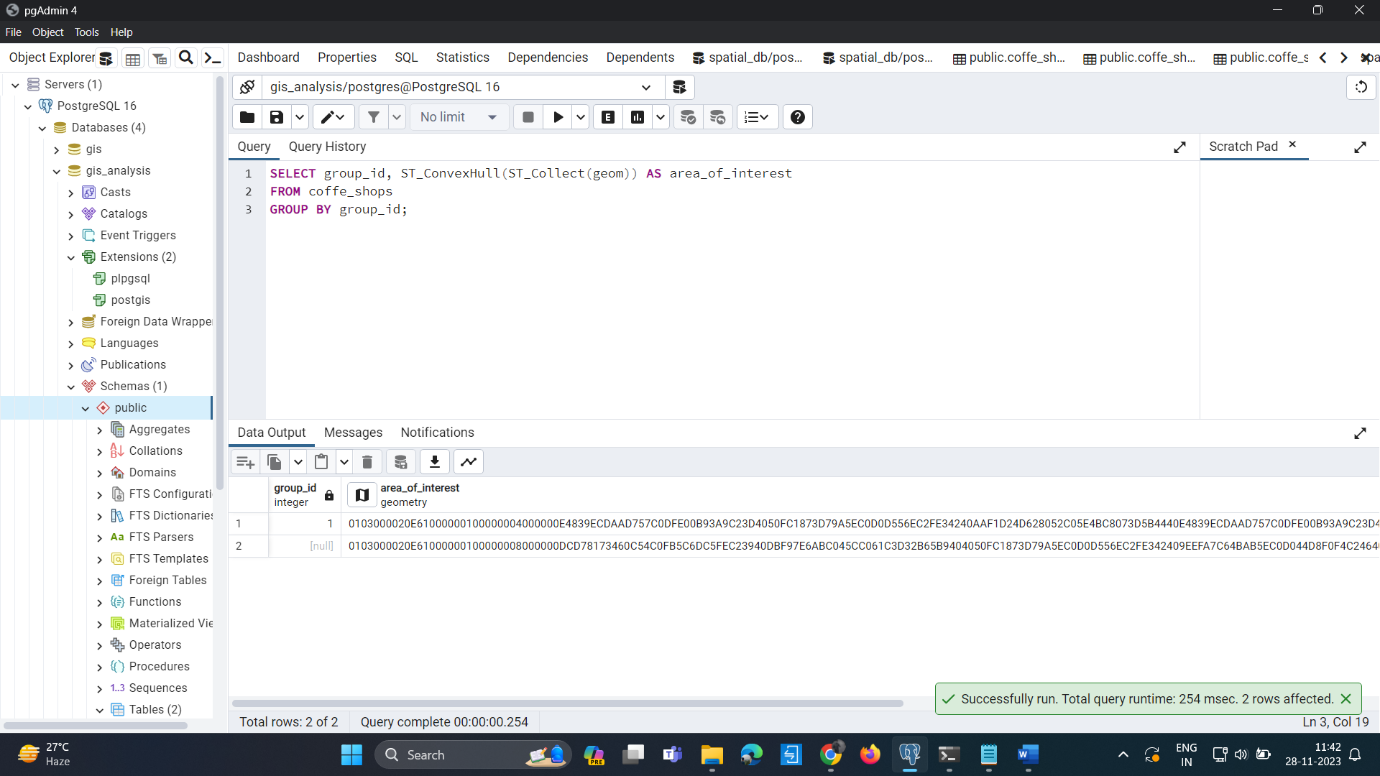
1. **Calculate Areas of Interest:**

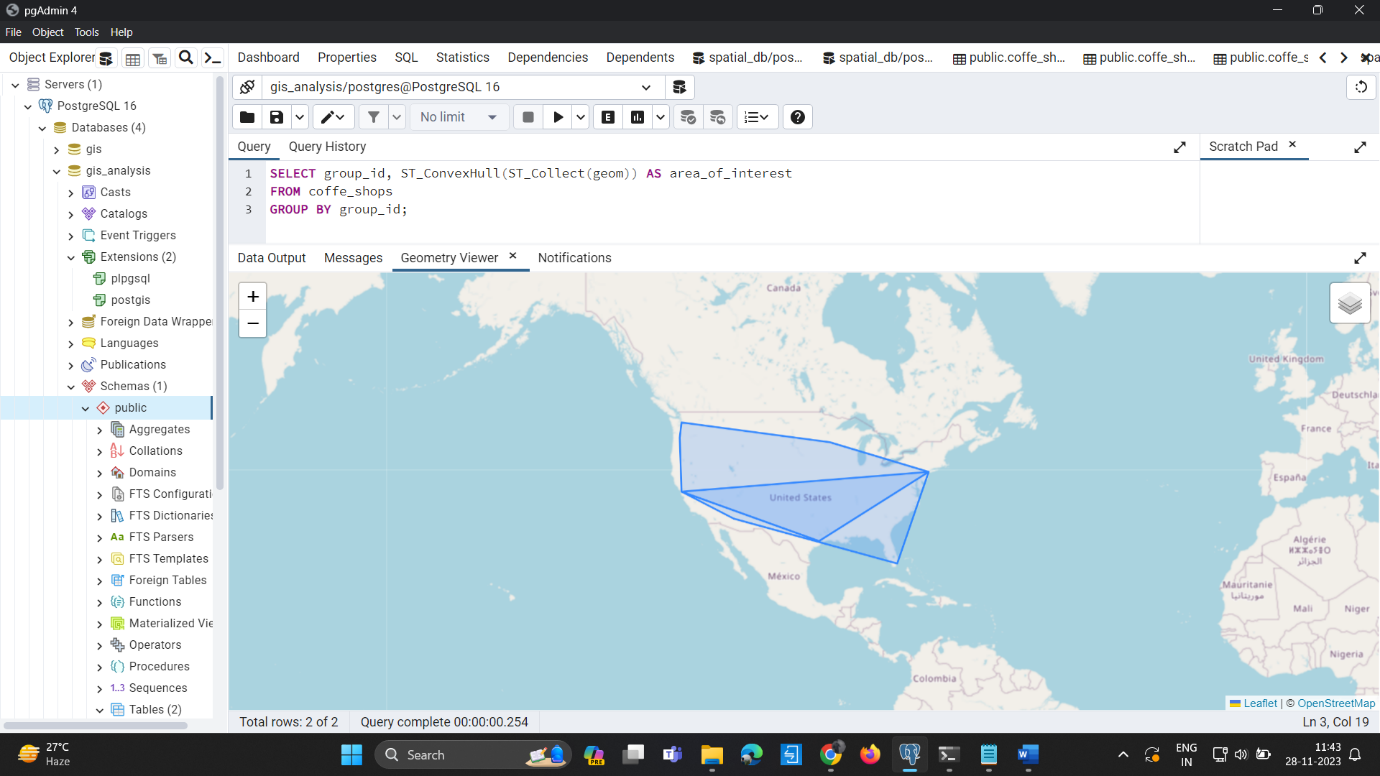
SELECT group\_id, ST\_ConvexHull(ST\_Collect(geom)) AS area\_of\_interest

FROM coffe\_shops

GROUP BY group\_id;

**Data Output:**



**Otuput in Geometric Map:**

* **Analyze the queries:**

**1) Create Table Query:**

**Query:**

CREATE TABLE coffe\_shops (

id SERIAL PRIMARY KEY,

name VARCHAR(255),

address VARCHAR(255),

city VARCHAR(255),

state VARCHAR(2),

zip INTEGER,

lat DOUBLE PRECISION,

lon DOUBLE PRECISION

);

**Explaination:**

This query creates a table named coffe\_shops with the following columns: id, name, address, city, state, zip, lat, and lon.

The id column is an auto-incrementing primary key.

The state column is defined as VARCHAR(2), assuming it's a two-letter state abbreviation.

The zip column is an INTEGER, assuming it represents a ZIP code.

The lat and lon columns are DOUBLE PRECISION, presumably representing latitude and longitude.

**2)Insert Data Query:**

**Query:**

INSERT INTO public.coffe\_shops (

id, name, address, city, state, zip, lat, lon

) VALUES

(1, 'Caffeine Corner', '123 Main St', 'Anytown', 'CA', 12345, 37.7749, -122.4194),

-- ... (similar lines for other entries)

(40, 'Buzz Brews', '3737 Cedar Street', 'Nowherenew', 'CA', 97531, 37.7749, -122.4194);

**Explaination:**

This query inserts data into the coffe\_shops table.

It inserts 40 rows of data, each representing a coffee shop with its attributes.

The data includes information such as name, address, city, state, ZIP code, latitude, and longitude.

**3)Distance Calculation Query:**

SELECT

id,

name,

ST\_Distance(

ST\_SetSRID(ST\_MakePoint(lon, lat), 4326),

ST\_SetSRID(ST\_MakePoint(-122.4194, 37.7749), 4326)

) AS distance\_from\_reference

FROM

coffe\_shops;

**Explaination:**

This query calculates the distance of each coffee shop from a reference point with longitude -122.4194 and latitude 37.7749.

It uses the ST\_Distance function with the reference point and the coffee shop coordinates.

**4)Insert Geom Column Query:**

ALTER TABLE coffe\_shops

ADD COLUMN geom geometry(POINT, 4326);

**Explaination:**

This query adds a new column named geom to the coffe\_shops table, representing a point geometry in the SRID 4326 (WGS 84).

**5)Update Geom Column Query:**

UPDATE coffe\_shops

SET geom = ST\_SetSRID(ST\_MakePoint(lon, lat), 4326);

**Explaination:**

This query updates the geom column for each row in the coffe\_shops table with the corresponding point geometry based on the lat and lon values.

**6)Distance Calculation (Geom) Query:**

SELECT

id,

name,

ST\_Distance(

geom,

ST\_SetSRID(ST\_MakePoint(-122.4194, 37.7749), 4326)

) AS distance\_from\_reference

FROM

coffe\_shops;

**Explaination:**

This query calculates the distance of each coffee shop from a reference point using the geom column, which contains point geometries.

7**)Calculate Areas of Interest Query (Assuming Grouping):**

SELECT

group\_id,

ST\_Area(ST\_Union(geom)) AS total\_area\_of\_interest

FROM

coffe\_shops

GROUP BY

group\_id;

**Explaination:**

This query calculates the total area of interest for each group based on the coffee shop locations.

It assumes the existence of a group\_id column in the coffe\_shops table.

* **Sorting and Limit Executions:**

**Query:**

SELECT

id,name,city,state

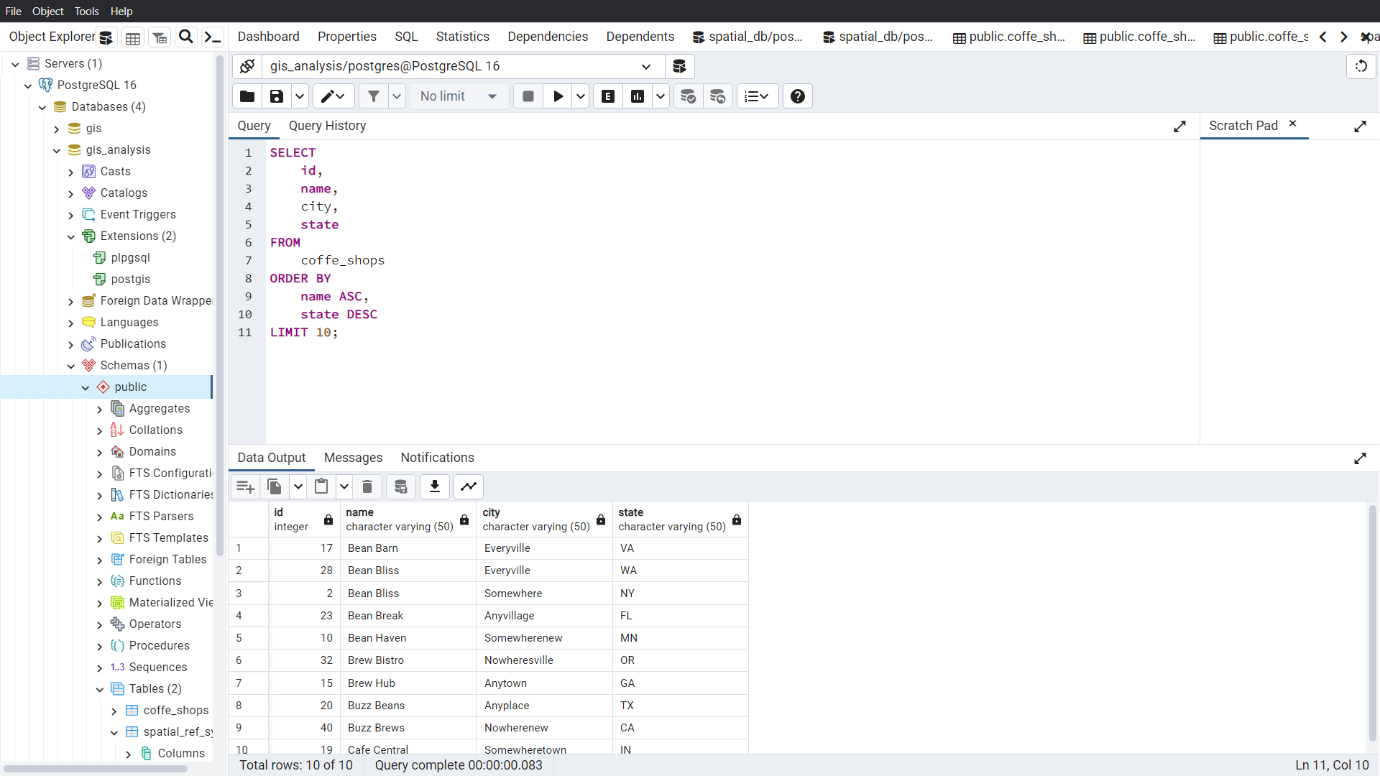
FROM

coffe\_shops

ORDER BY

name ASC,state DESC

LIMIT 10;

**Output:**

* **Optimize the queries to speed up execution time:**

**Query:**

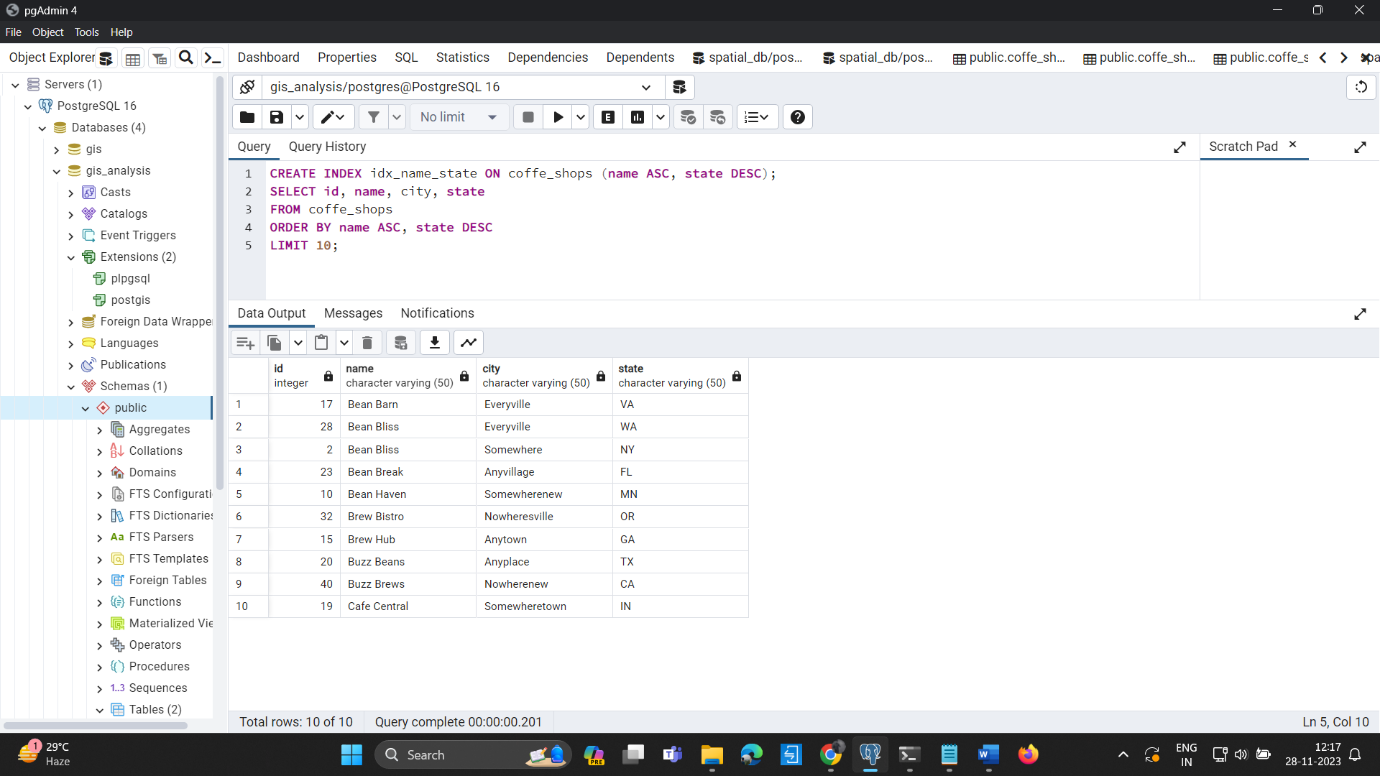
CREATE INDEX idx\_name\_state ON coffe\_shops (name ASC, state DESC);

SELECT id, name, city, state

FROM coffe\_shops

ORDER BY name ASC, state DESC

LIMIT 10;

**Output:**

**Optimized Sort and Limit (Different Columns) Query:**

CREATE INDEX idx\_lon\_lat ON coffe\_shops (lon DESC, lat ASC);

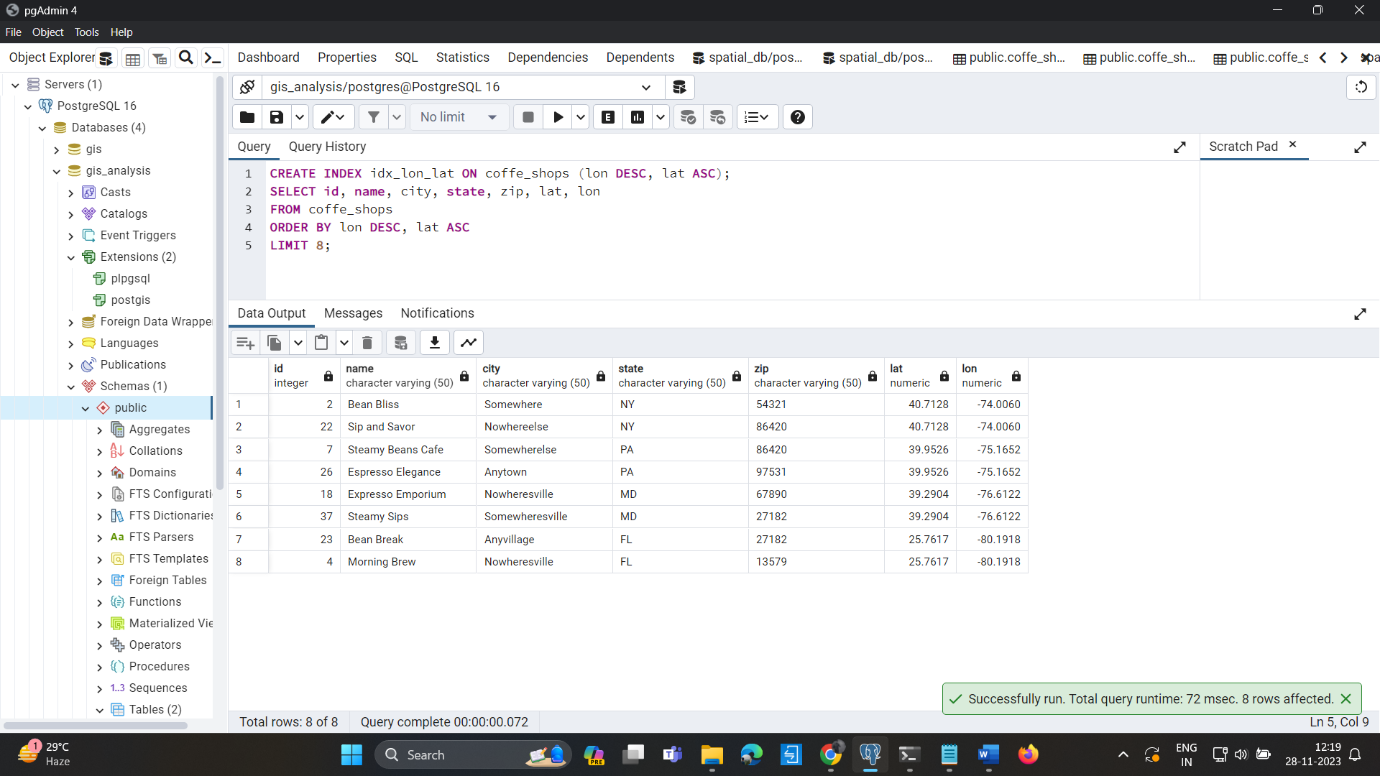
SELECT id, name, city, state, zip, lat, lon

FROM coffe\_shops

ORDER BY lon DESC, lat ASC

LIMIT 8;

**Output:**



* **N-Optimization:**

Optimizing queries for performance involves various strategies, and the specific optimizations depend on the structure of the database, the nature of queries, and the underlying hardware. Here are some general tips for optimizing queries:

**Indexing:** Ensure that appropriate indexes are created on columns used in WHERE clauses and JOIN conditions. For example:

**Query:**

CREATE INDEX idx\_name ON coffe\_shops (name);

CREATE INDEX idx\_group\_id ON coffe\_shops (group\_id);

**\*\*Avoid SELECT \*\*\*:** Only select the columns you need. This reduces the amount of data that needs to be processed.

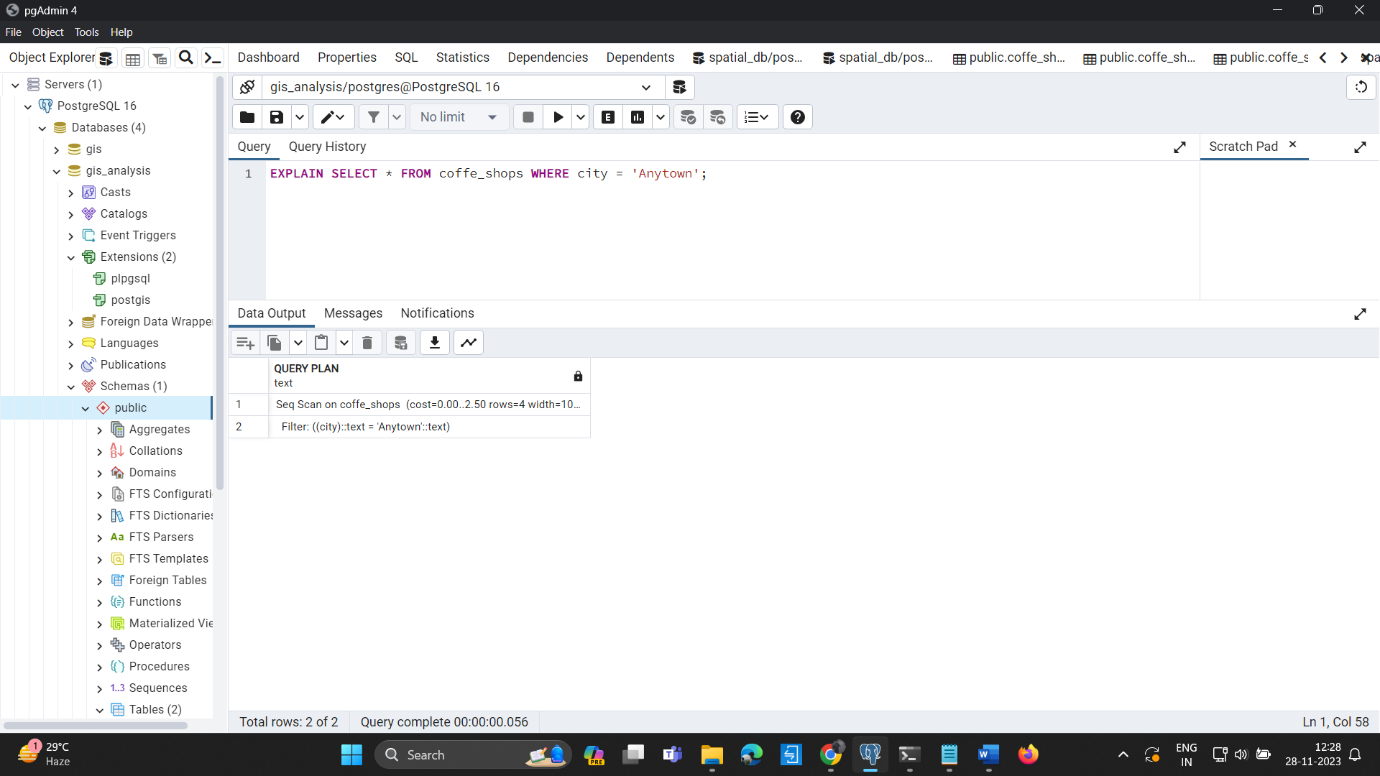
**Limit the Result Set:** Use LIMIT and OFFSET to restrict the number of rows returned, especially for paginated results.

**Query Optimization:** Make sure your queries are well-optimized. Use the EXPLAIN command to analyze query plans.

**Query:**

EXPLAIN SELECT \* FROM coffe\_shops WHERE city = 'Anytown';

**Output:**



**Normalization:** Ensure that your database is normalized to an appropriate degree. This can reduce redundancy and improve query performance.

**Caching:** Consider caching frequently used queries or results.

**Partitioning:** If your table is very large, consider partitioning it based on a key. This can improve query performance.

**Hardware and Server Optimization:** Ensure that your database server has enough resources (RAM, CPU) to handle the workload.

**Team Members:**

1. **Sandeep Malyala** sm86925n@pace.edu
2. **Pravallika Challa** pc94436n@pace.edu
3. **Chetan Kumar Reddy Vaddi** CV33511N@pace.edu
4. **Ramadugu Bhavitha** br31457n@pace.edu
5. **Sai Kiran Nunna** sn01516n@pace.edu