CS610 Project

GIS

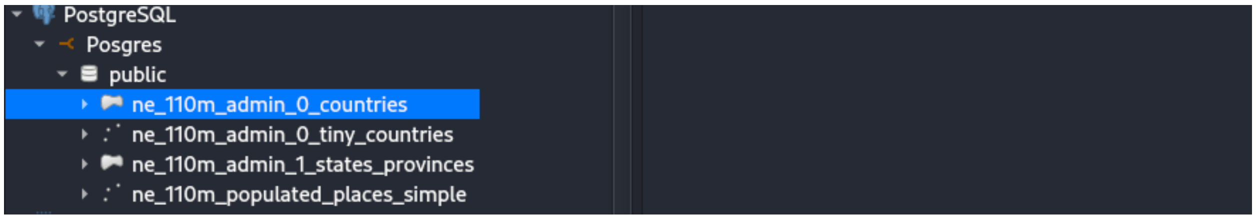
->The main thing we have two be doing to work on spatial datas is to install postgis in postgresql database.

->We use the query -
'Create extension postgis'. In the terminal.

Now let us get into the project.

->This project has severals subparts lets discuss them individually-

->At first instead of creating several table we are directly uploading the dataset that we need to fulfill this project's requirements.


```
PostgreSQL
  Posgres
    public
      ne_110m_admin_0_countries
      ne_110m_admin_0_tiny_countries
      ne_110m_admin_1_states_provinces
      ne_110m_populated_places_simple
```

1)Retrieve Locations of Specific Features.

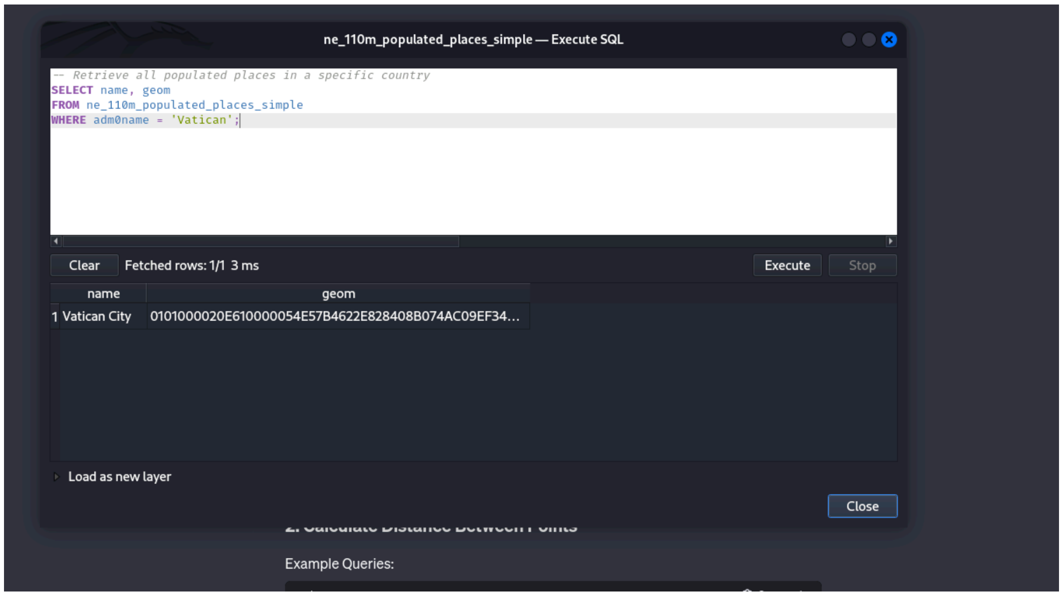
<u>Code:</u>

```
-- Retrieve all populated places in a specific country
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE adm0name = 'Vatican';

-- Retrieve cities with a population greater than a certain
    threshold
```

```
SELECT name, geom
FROM ne_110m_populated_places_simple
```

Output:

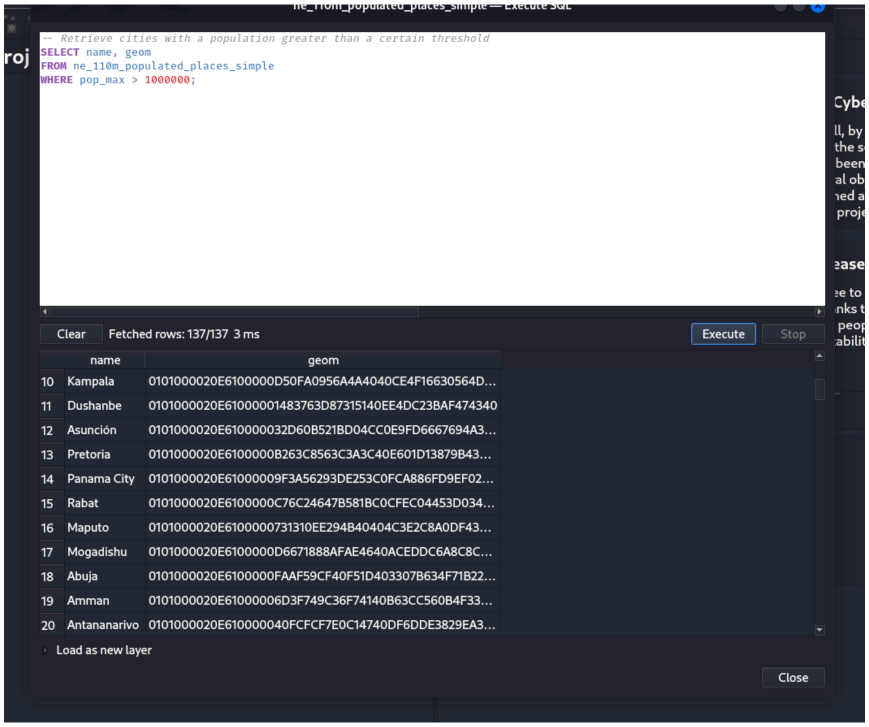


ne_110m_populated_places_simple — Execute SQL

```
-- Retrieve all populated places in a specific country
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE adm0name = 'Vatican';
```

Fetched rows: 1/1 3 ms

	name	geom
1	Vatican City	0101000020E610000054E57B4622E828408B074AC09EF34…



```
-- Retrieve cities with a population greater than a certain threshold
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE pop_max > 1000000;
```

Fetched rows: 137/137 3 ms

	name	geom
10	Kampala	0101000020E6100000D50FA0956A4A4040CE4F16630564D…
11	Dushanbe	0101000020E61000001483763D87315140EE4DC23BAF474340
12	Asunción	0101000020E610000032D60B521BD04CC0E9FD6667694A3…
13	Pretoria	0101000020E6100000B263C8563C3A3C40E601D13879B43…
14	Panama City	0101000020E61000009F3A56293DE253C0FCA886FD9EF02…
15	Rabat	0101000020E6100000C76C24647B581BC0CFEC04453D034…
16	Maputo	0101000020E6100000731310EE294B40404C3E2C8A0DF43…
17	Mogadishu	0101000020E6100000D6671888AFAE4640ACEDDC6A8C8C…
18	Abuja	0101000020E6100000FAAF59CF40F51D403307B634F71B22…
19	Amman	0101000020E61000006D3F749C36F74140B63CC560B4F33…
20	Antananarivo	0101000020E610000040FCFCF7E0C14740DF6DDE3829EA3…

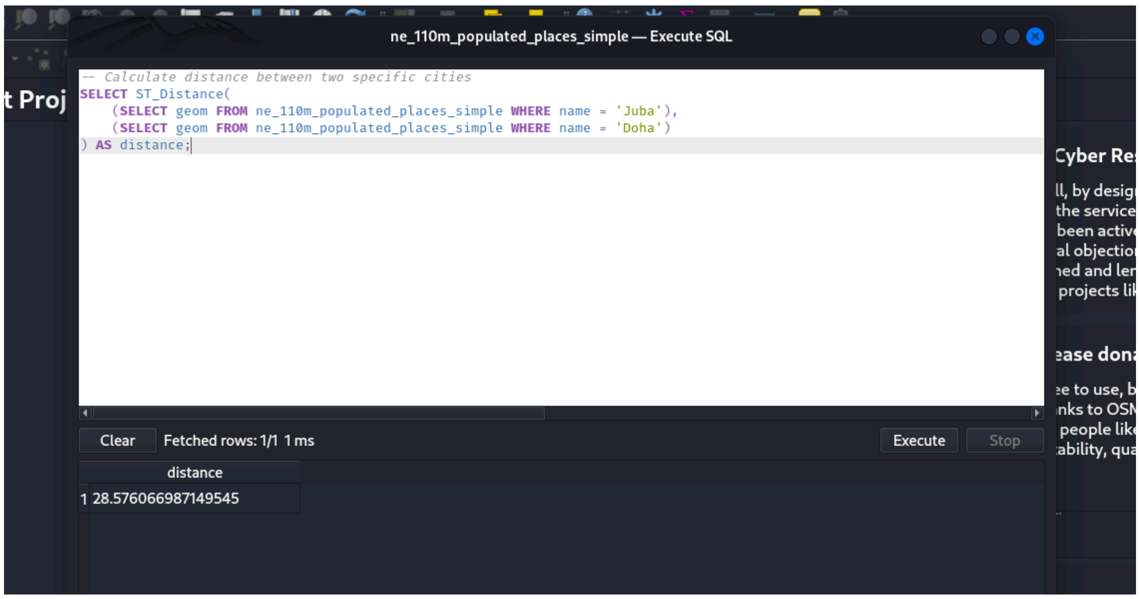
2)Distance Between Points.

<u>Code:</u>

```
        -- CALCULATE DISTANCE BETWEEN TWO SPECIFIC CITIES
    SELECT ST_Distance(
        (SELECT geom FROM ne_110m_populated_places_simple
        WHERE name = 'Juba'),
        (SELECT geom FROM ne_110m_populated_places_simple
        WHERE name = 'Doha')
    ) AS distance;
    = 28.57
```

```
        -- CALCULATE DISTANCES FROM A SPECIFIC CITY TO ALL
OTHER CITIES
    SELECT name, ST_Distance(
        (SELECT geom FROM ne_110m_populated_places_simple
        WHERE name = 'Doha'),
        geom
    ) AS distance
    FROM ne_110m_populated_places_simple
    WHERE name != 'Doha' ORDER BY distance DESC LIMIT 10;
```

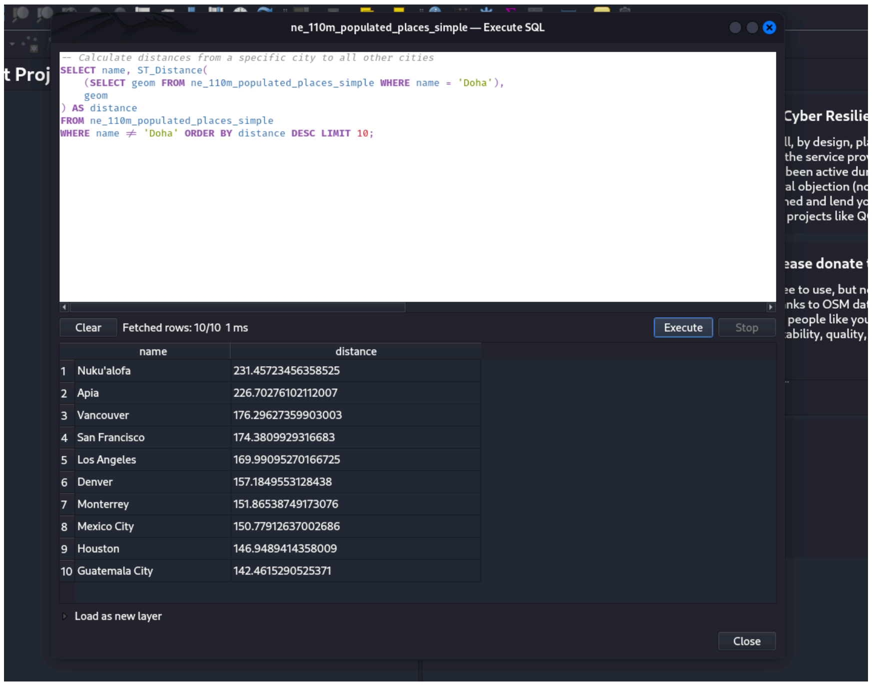
<u>Output:</u>



```
-- Calculate distance between two specific cities
SELECT ST_Distance(
    (SELECT geom FROM ne_110m_populated_places_simple WHERE name = 'Juba'),
    (SELECT geom FROM ne_110m_populated_places_simple WHERE name = 'Doha')
) AS distance;
```

```
Fetched rows: 1/1 1 ms

  distance
1 28.576066987149545
```



ne_110m_populated_places_simple — Execute SQL

```sql
-- Calculate distances from a specific city to all other cities
SELECT name, ST_Distance(
    (SELECT geom FROM ne_110m_populated_places_simple WHERE name = 'Doha'),
    geom
) AS distance
FROM ne_110m_populated_places_simple
WHERE name ≠ 'Doha' ORDER BY distance DESC LIMIT 10;
```

Fetched rows: 10/10 1 ms

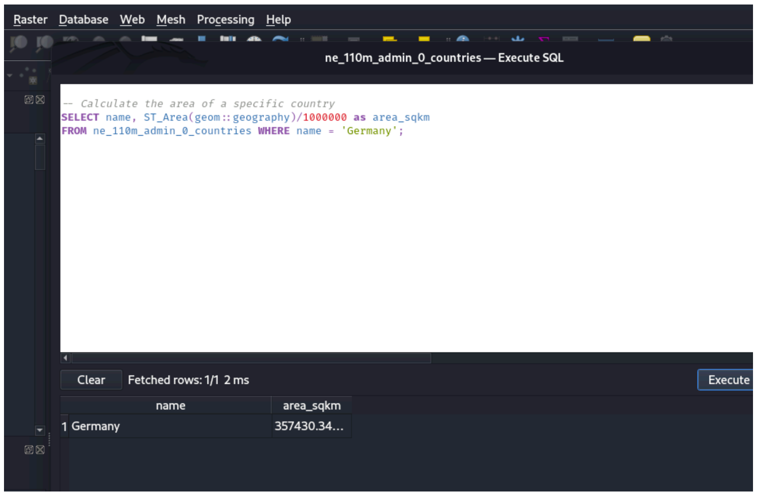
	name	distance
1	Nuku'alofa	231.45723456358525
2	Apia	226.70276102112007
3	Vancouver	176.29627359903003
4	San Francisco	174.3809929316683
5	Los Angeles	169.99095270166725
6	Denver	157.1849553128438
7	Monterrey	151.86538749173076
8	Mexico City	150.77912637002686
9	Houston	146.9489414358009
10	Guatemala City	142.4615290525371

3)Calculate Areas of Interest.

<u>Code</u>:

-- CALCULATE THE AREA OF A SPECIFIC COUNTRY
SELECT name, ST_Area(geom::geography)/1000000 as area_sqkm
FROM ne_110m_admin_0_countries WHERE name = 'Germany';

<u>Output:</u>



ne_110m_admin_0_countries — Execute SQL

```sql
-- Calculate the area of a specific country
SELECT name, ST_Area(geom::geography)/1000000 as area_sqkm
FROM ne_110m_admin_0_countries WHERE name = 'Germany';
```

Fetched rows: 1/1 2 ms

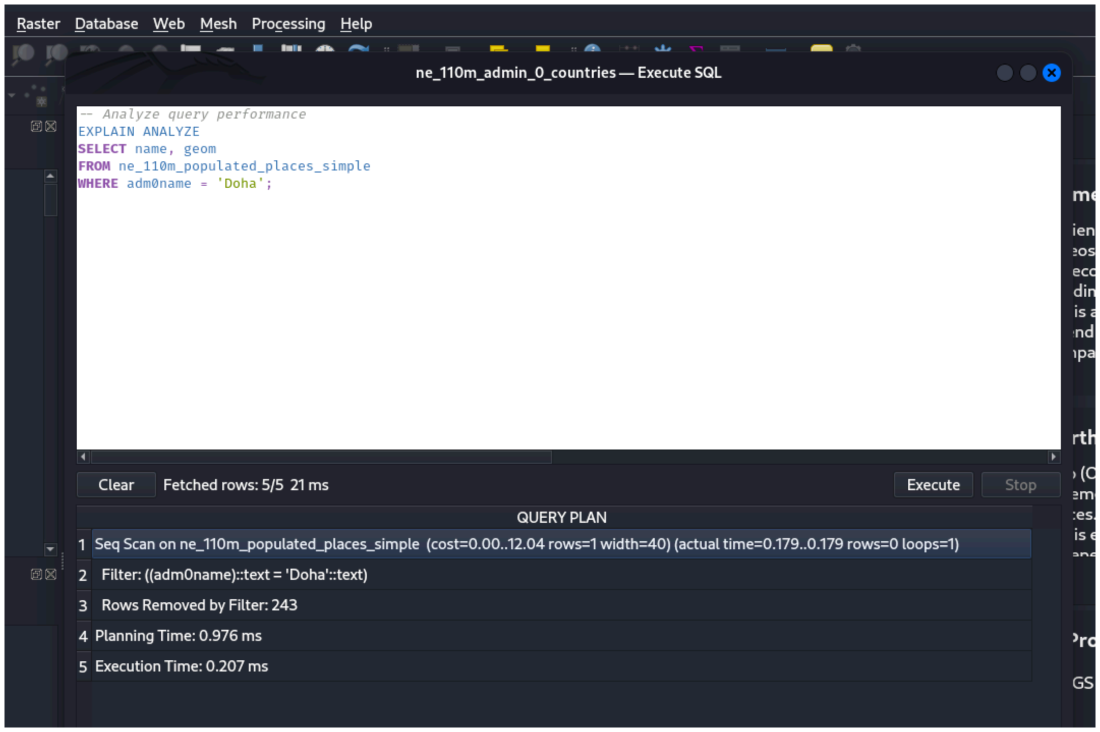
	name	area_sqkm
1	Germany	357430.34...

4) Analysing Queries.

Code:

```
-- ANALYZE QUERY PERFORMANCE
EXPLAIN ANALYZE
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE adm0name = ‘Doha';
```

Output:


Raster Database Web Mesh Processing Help
ne_110m_admin_0_countries — Execute SQL
Clear
Execute
Stop
QUERY PLAN

```
-- Analyze query performance
EXPLAIN ANALYZE
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE adm0name = 'Doha';
```

Fetched rows: 5/5 21 ms

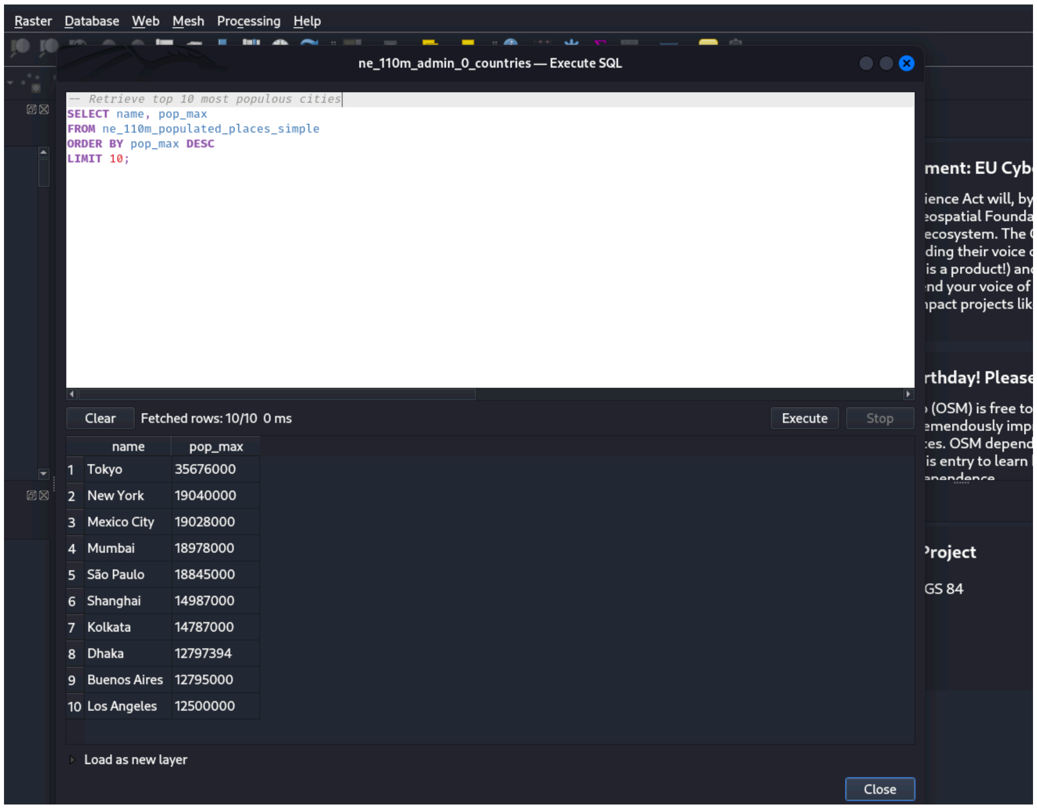
```
1 Seq Scan on ne_110m_populated_places_simple  (cost=0.00..12.04 rows=1 width=40) (actual time=0.179..0.179 rows=0 loops=1)
2   Filter: ((adm0name)::text = 'Doha'::text)
3   Rows Removed by Filter: 243
4 Planning Time: 0.976 ms
5 Execution Time: 0.207 ms
```

5)Sorting and Limit Executions.

Code:

```
-- RETRIEVE TOP 10 MOST POPULOUS CITIES
SELECT name, pop_max
FROM ne_110m_populated_places_simple
ORDER BY pop_max DESC
LIMIT 10;
```

Output:



```
-- Retrieve top 10 most populous cities
SELECT name, pop_max
FROM ne_110m_populated_places_simple
ORDER BY pop_max DESC
LIMIT 10;
```

	name	pop_max
1	Tokyo	35676000
2	New York	19040000
3	Mexico City	19028000
4	Mumbai	18978000
5	São Paulo	18845000
6	Shanghai	14987000
7	Kolkata	14787000
8	Dhaka	12797394
9	Buenos Aires	12795000
10	Los Angeles	12500000

6)Optimize the Queries to Speed Up Execution Time.

Code:

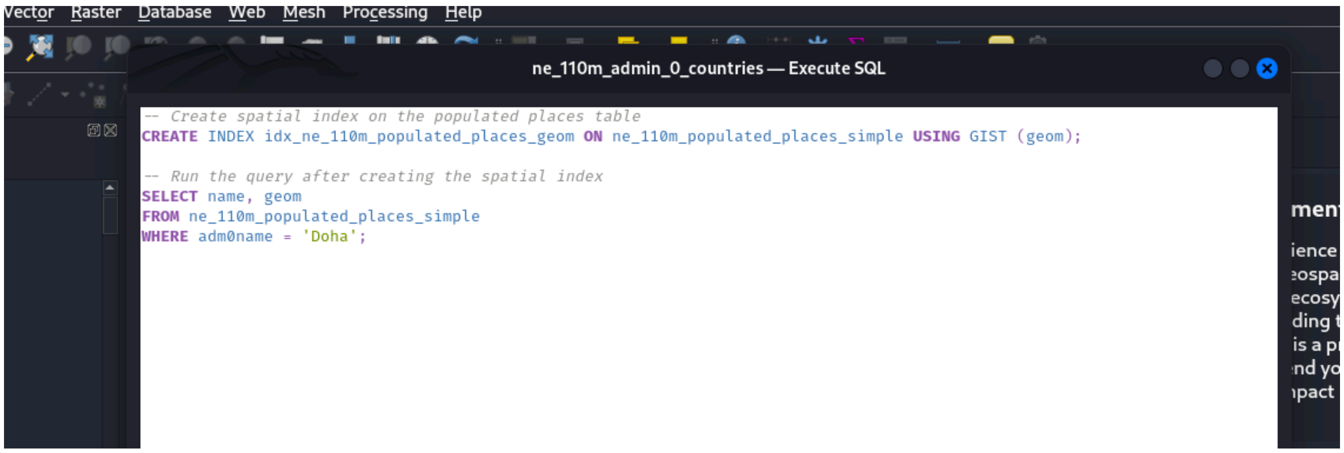
-- CREATE SPATIAL INDEX ON THE POPULATED PLACES TABLE

CREATE INDEX idx_ne_110m_populated_places_geom ON ne_110m_populated_places_simple USING GIST (geom);

-- RUN THE QUERY AFTER CREATING THE SPATIAL INDEX

SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE adm0name = ‘Doha';

Output:


Vector Raster Database Web Mesh Processing Help
ne_110m_admin_0_countries — Execute SQL

```
-- Create spatial index on the populated places table
CREATE INDEX idx_ne_110m_populated_places_geom ON ne_110m_populated_places_simple USING GIST (geom);

-- Run the query after creating the spatial index
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE adm0name = 'Doha';
```

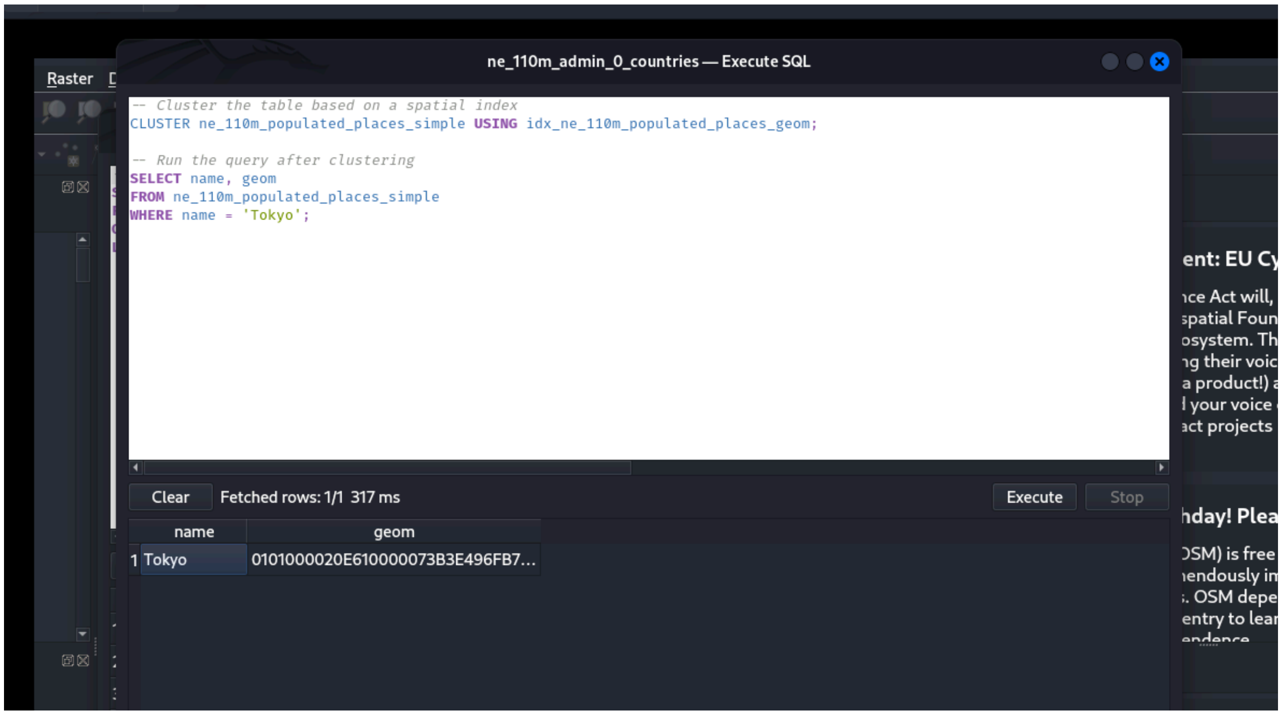
7)N-Optimization of Queries.

Code:

```
-- CLUSTER THE TABLE BASED ON A SPATIAL INDEX
CLUSTER ne_110m_populated_places_simple USING
idx_ne_110m_populated_places_geom;

-- RUN THE QUERY AFTER CLUSTERING
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE name = ‘Tokyo';
```

Output:



```
-- Cluster the table based on a spatial index
CLUSTER ne_110m_populated_places_simple USING idx_ne_110m_populated_places_geom;

-- Run the query after clustering
SELECT name, geom
FROM ne_110m_populated_places_simple
WHERE name = 'Tokyo';
```

Fetched rows: 1/1 317 ms

	name	geom
1	Tokyo	0101000020E610000073B3E496FB7...