DATABASE MANAGEMENT SYSTEMS

FALL 2023

CS – 623 PROJECT

Professor: Mr. Buchi Okoli Okoli

Project Members

Vishnu Vardhan Reddy Bijjam, Manikanta Reddy Yerolla

GOAL
Our goal is to provide the Geographic Information System (GIS) analysis, using database that supports spatial data types which is PostGIS.

DATASOURCE: https://download.geofabrik.de/north-america/us.html

In the above git hub link, the Geographic Information of US spatial data information is available. It includes types of buildings, amount of land used by different sectors, details about transportation and terrain details.

Data importation will be conducted utilizing the PostGIS shapefile import/export manager. This utility facilitates the transfer of shapefiles from a designated directory into the database.

Once the files are uploaded through the user interface, modify the Spatial Reference ID (SRID) to 4326. This SRID corresponds to the WGS 84 coordinate system, which uses longitude and latitude to represent spatial data on the Earth's surface. After adjusting the SRID, proceed with the data importation.

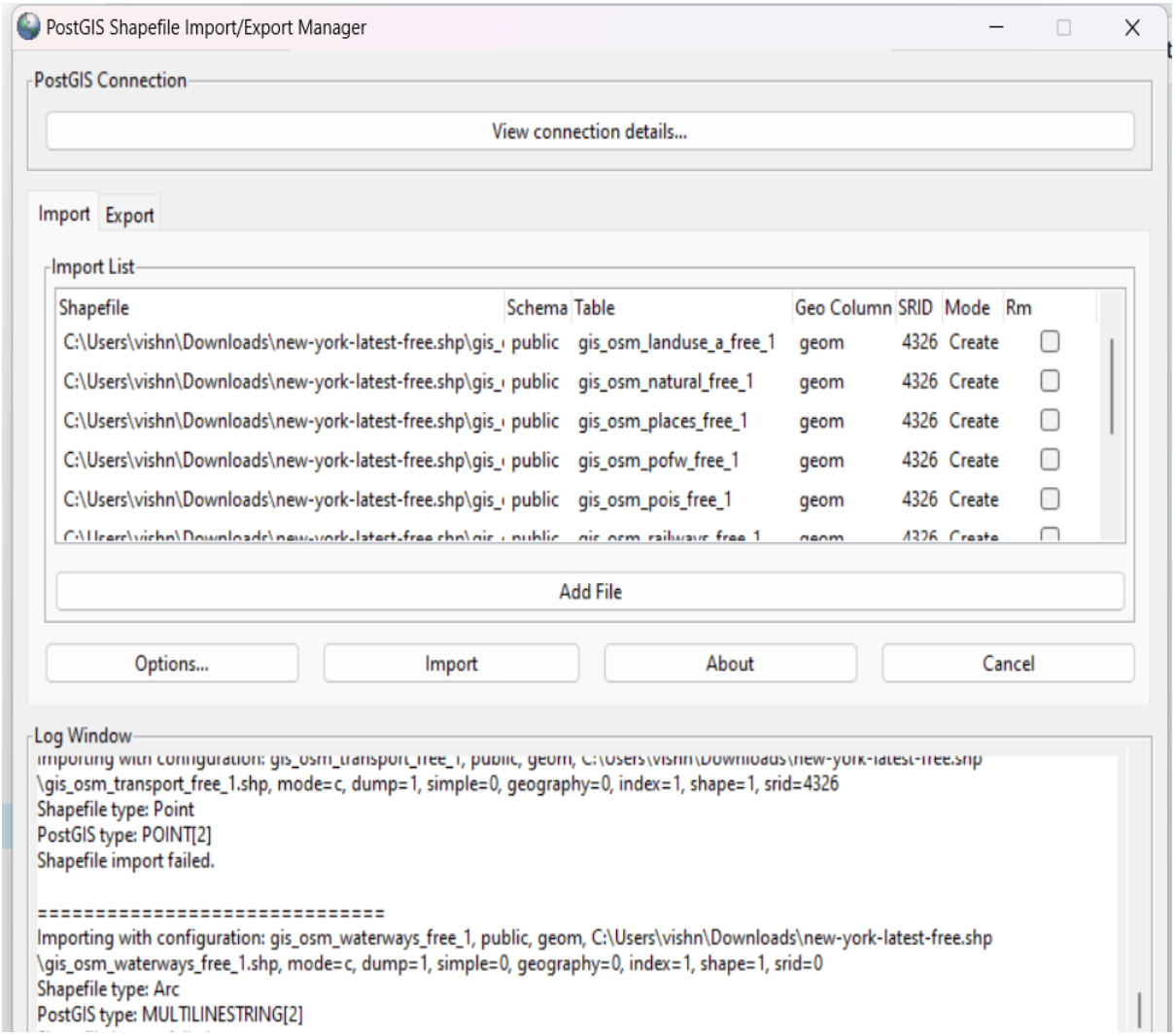


Table schemas:

Query | Query History

```
SELECT
     column_name,
     data_type,
     is_nullable
FROM
     information_schema.columns
WHERE
     table_name = 'gis_osm_buildings_a_free_1'
```

Data Output | Messages | Notifications

	column_name name	data_type character varying	is_nullable character varying (3)
1	gid	integer	NO
2	osm_id	character varying	YES
3	code	smallint	YES
4	fclass	character varying	YES
5	name	character varying	YES
6	type	character varying	YES
7	geom	USER-DEFINED	YES

Query | Query History

```
SELECT
     column_name,
     data_type,
     is_nullable
FROM
     information_schema.columns
WHERE
     table_name = 'gis_osm_landuse_a_free_1'
```

Data Output | Messages | Notifications

	column_name name	data_type character varying	is_nullable character varying (3)
1	gid	integer	NO
2	osm_id	character varying	YES
3	code	smallint	YES
4	fclass	character varying	YES
5	name	character varying	YES
6	geom	USER-DEFINED	YES

Query | Query History

```sql
SELECT
     column_name,
     data_type,
     is_nullable
FROM
     information_schema.columns
WHERE
     table_name = 'gis_osm_roads_free_1'
```

Data Output | Messages | Notifications

	column_name name	data_type character varying	is_nullable character varying (3)
1	gid	integer	NO
2	osm_id	character varying	YES
3	code	smallint	YES
4	fclass	character varying	YES
5	name	character varying	YES
6	ref	character varying	YES
7	oneway	character varying	YES
8	maxspeed	smallint	YES
9	layer	double precision	YES
10	bridge	character varying	YES
11	tunnel	character varying	YES

Total rows: 12 of 12 | Query complete 00:00:00.185

Query | Query History

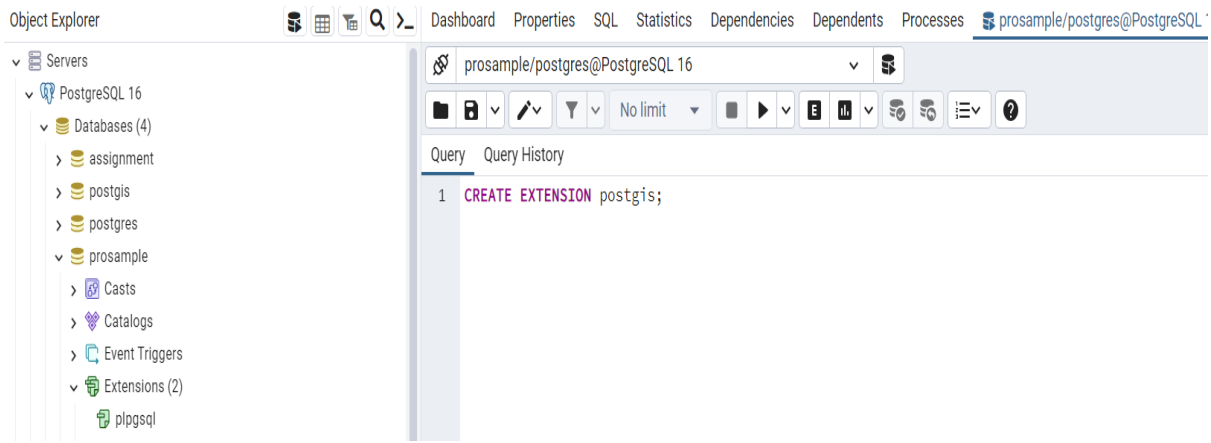
```sql
SELECT
    column_name,
    data_type,
    is_nullable
FROM
    information_schema.columns
WHERE
    table_name = 'gis_osm_natural_free_1'
```

Data Output | Messages | Notifications

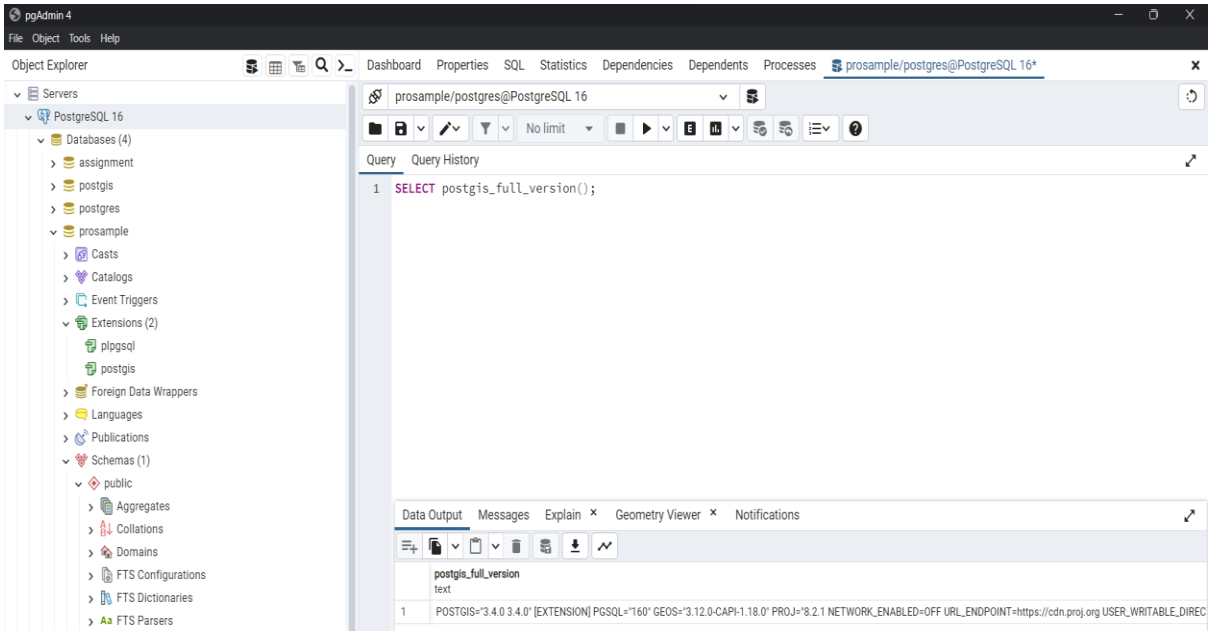
	column_name name	data_type character varying	is_nullable character varying (3)
1	gid	integer	NO
2	osm_id	character varying	YES
3	code	smallint	YES
4	fclass	character varying	YES
5	name	character varying	YES
6	geom	USER-DEFINED	YES

Creating postgis extension

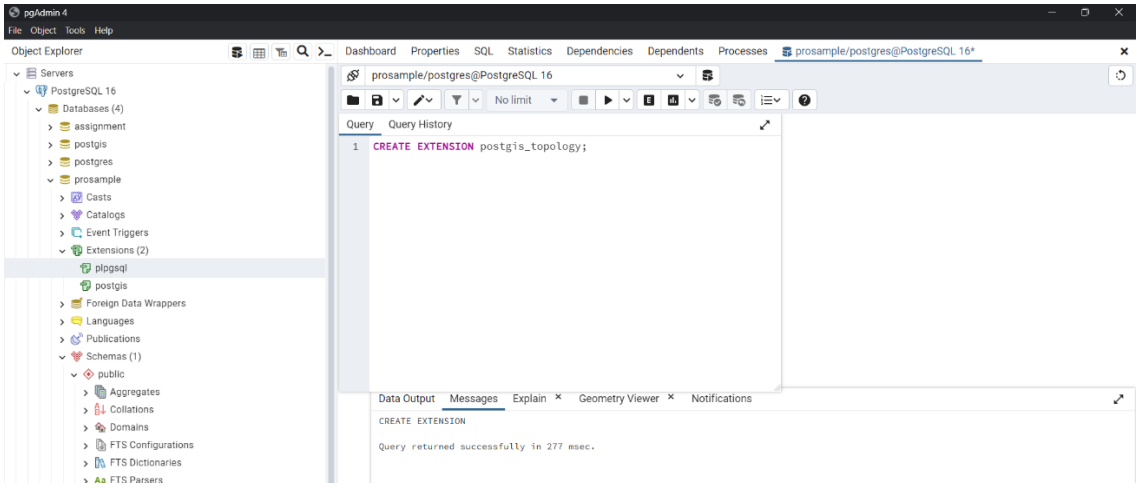
CREATE EXTENSION postgis;



Verifying postgis version



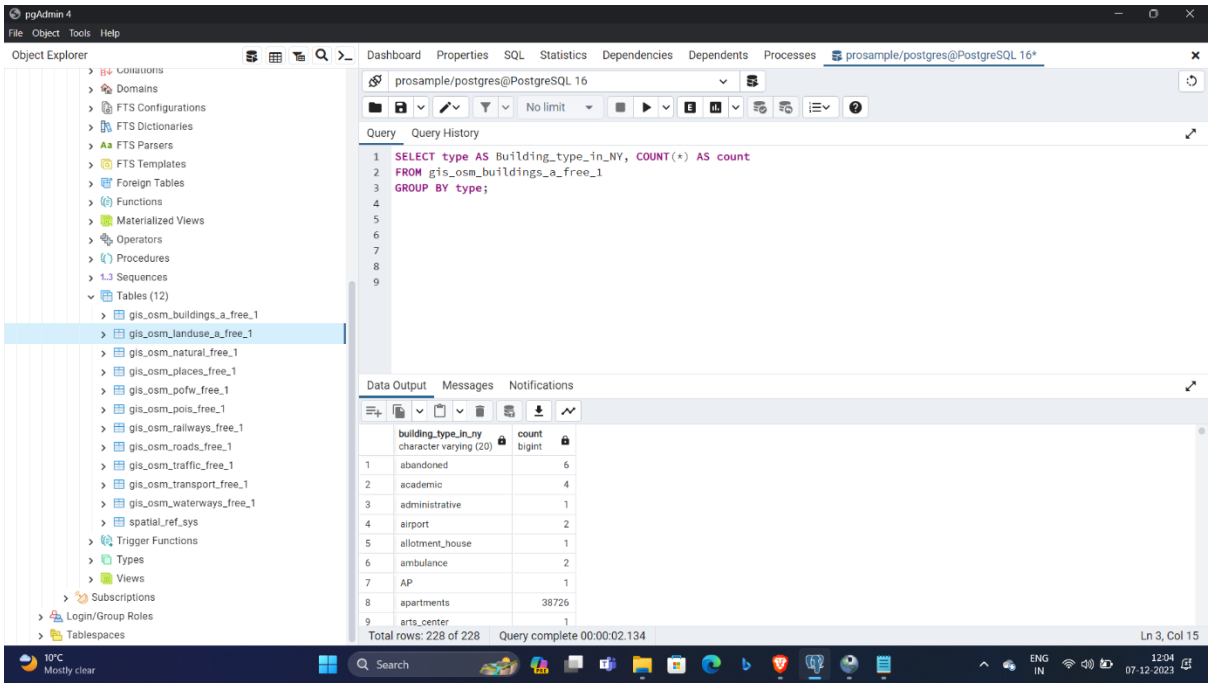
Creating postgis topology



1.Retrieve Locations of specific features

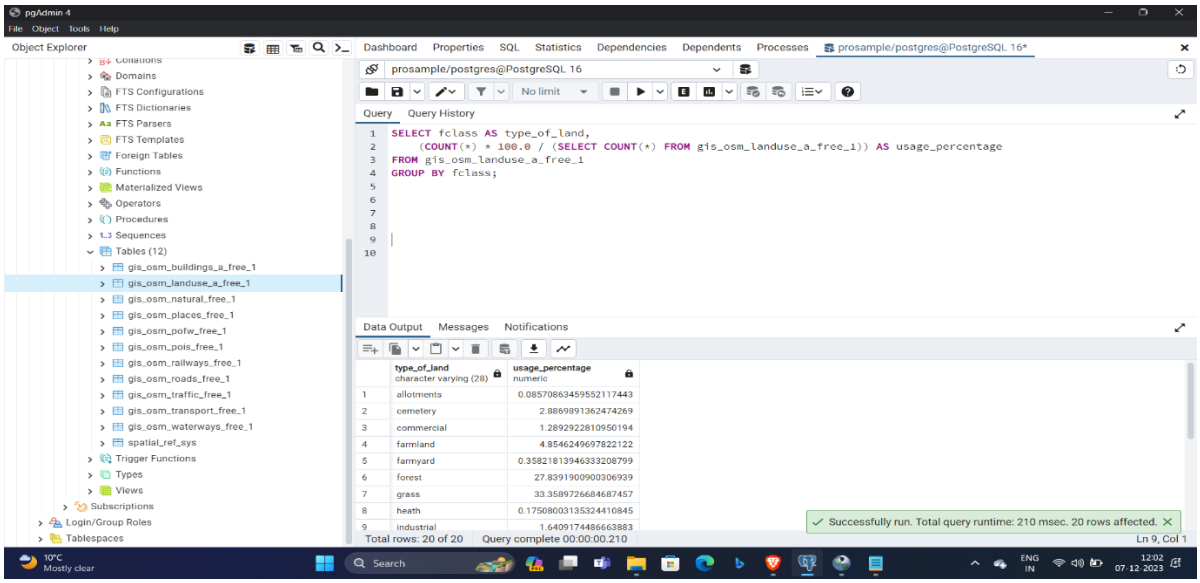
1.1 Retrieving number of buildings of each type in New York.

```
SELECT type AS Building_type_in_NY, COUNT (*) AS count
FROM gis_osm_buildings_a_free_1
GROUP BY type;
```



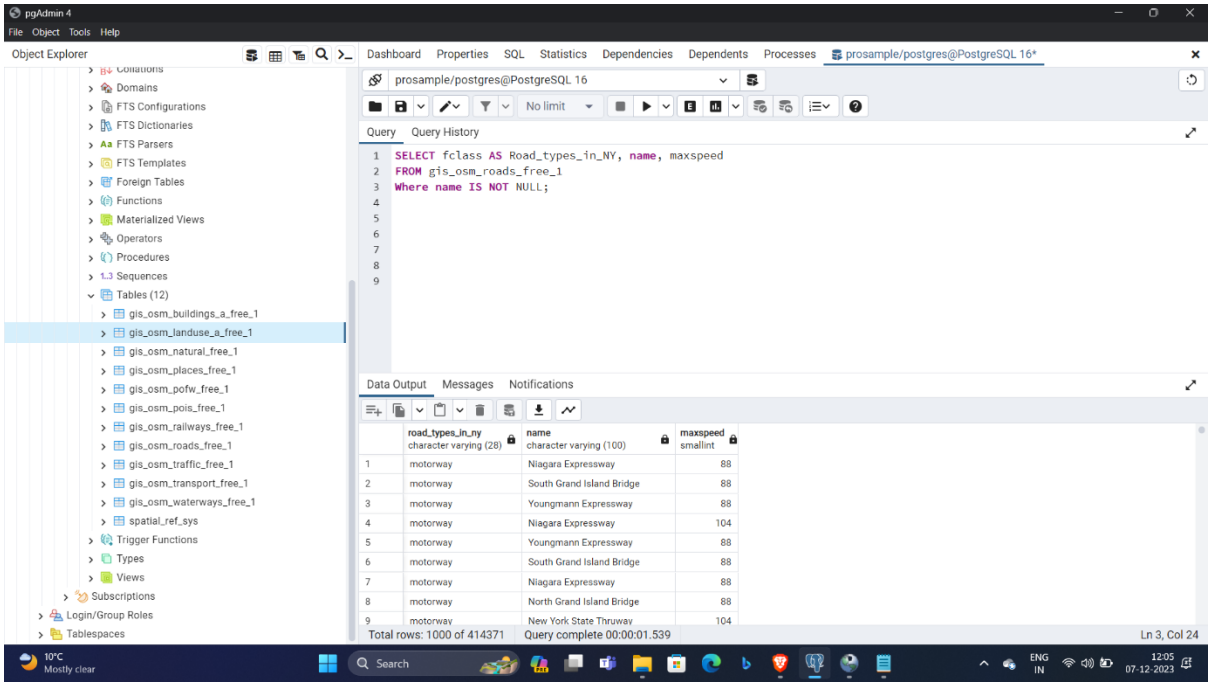
1.2 Retrieving percentage of usage of land for different purpose in NY.

```
SELECT fclass AS type_of_land,
   (COUNT (*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS
usage_percentage
FROM gis_osm_landuse_a_free_1
GROUP BY fclass;
```



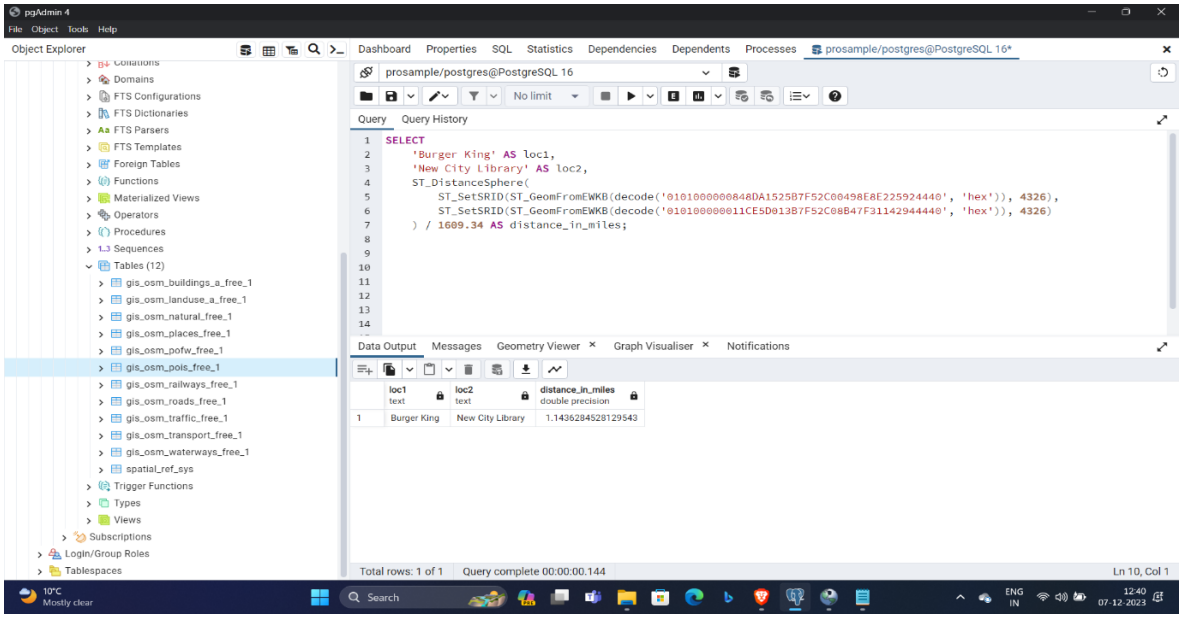
1.3 Retrieving type of roads and their max speed.

```
SELECT fclass AS Road_types_in_NY, name, maxspeed
FROM gis_osm_roads_free_1
Where name IS NOT NULL;
```



2. Calculate Distance between two points

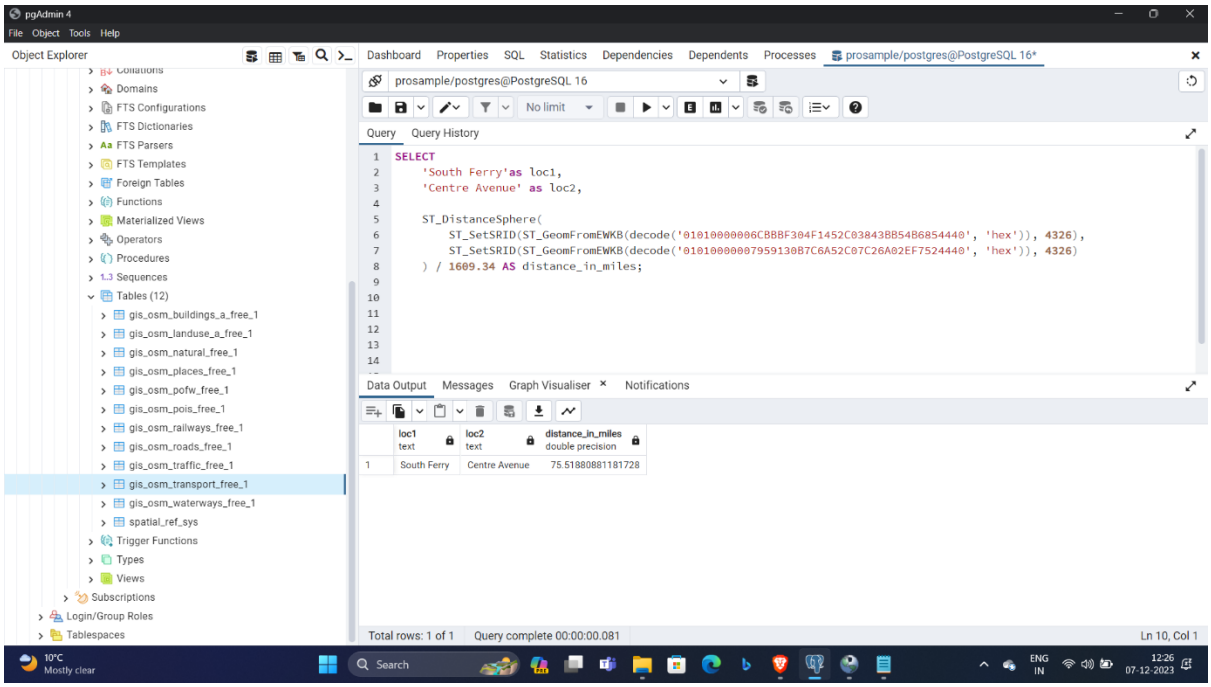
```
SELECT
    'Burger King' AS loc1,
    'New City Library' AS loc2,
    ST_DistanceSphere(

ST_SetSRID(ST_GeomFromEWKB(decode('0101000000848DA1525B7F52C00498E8E225924440',
'hex')), 4326),

ST_SetSRID(ST_GeomFromEWKB(decode('010100000011CE5D013B7F52C08B47F31142944440',
'hex')), 4326)
    ) / 1609.34 AS distance_in_miles;
```



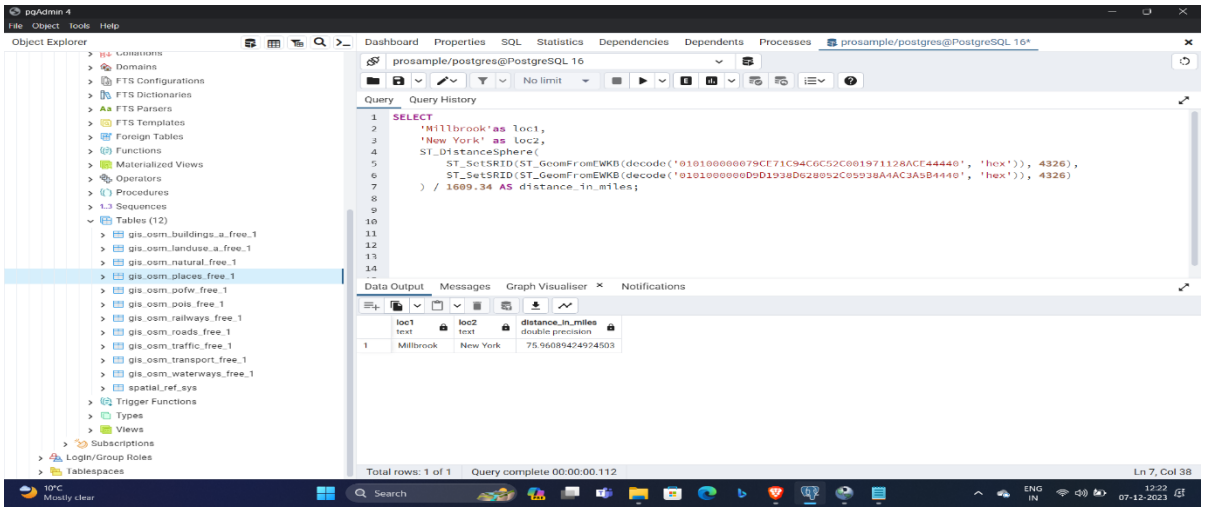
SELECT
'South Ferry' as loc1,
'Centre Avenue' as loc2,
ST_DistanceSphere(

ST_SetSRID(ST_GeomFromEWKB(decode('0101000000C6BBBF304F1452C03843BB568B54440', 'hex')), 4326),

ST_SetSRID(ST_GeomFromEWKB(decode('01010000007959130B7C6A52C07C6A620EF7524440', 'hex')), 4326)
) / 1609.34 AS distance_in_miles;



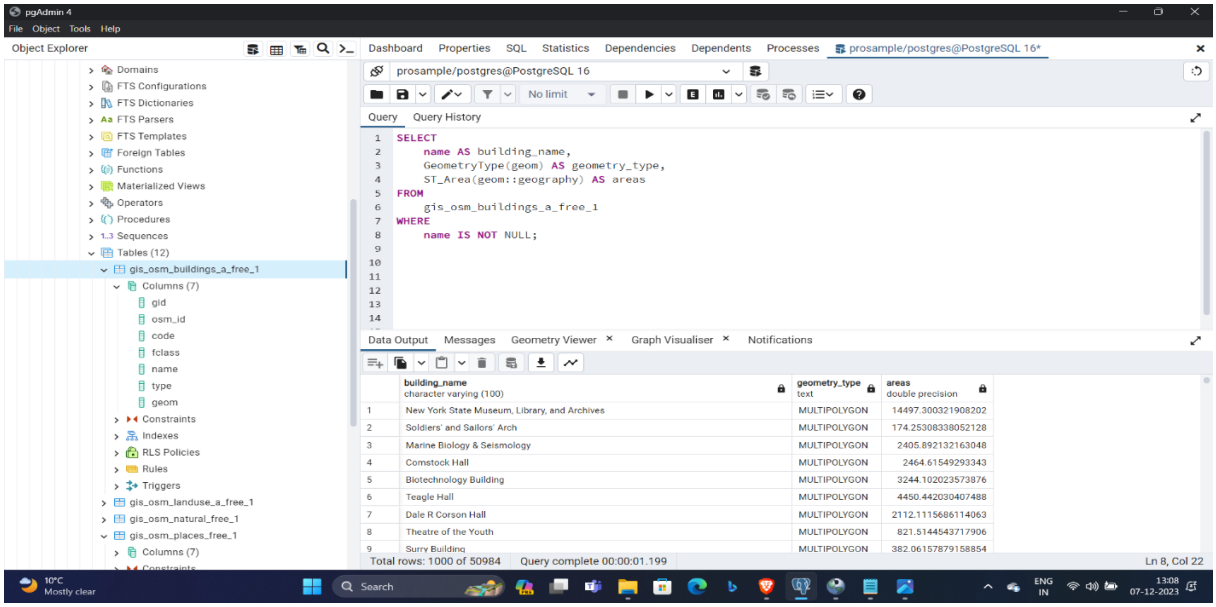
```
SELECT
        'Millbrook'as loc1,
        'New York' as loc2,
   ST_DistanceSphere(

ST_SetSRID(ST_GeomFromEWKB(decode('010100000079CE71C94C6C52C001971128ACE44440
', 'hex')), 4326),

ST_SetSRID(ST_GeomFromEWKB(decode('0101000000D9D1938D628052C05938A4AC3A5B444
0', 'hex')), 4326)
   ) / 1609.34 AS distance_in_miles;
```



3. Calculate Areas of Interest

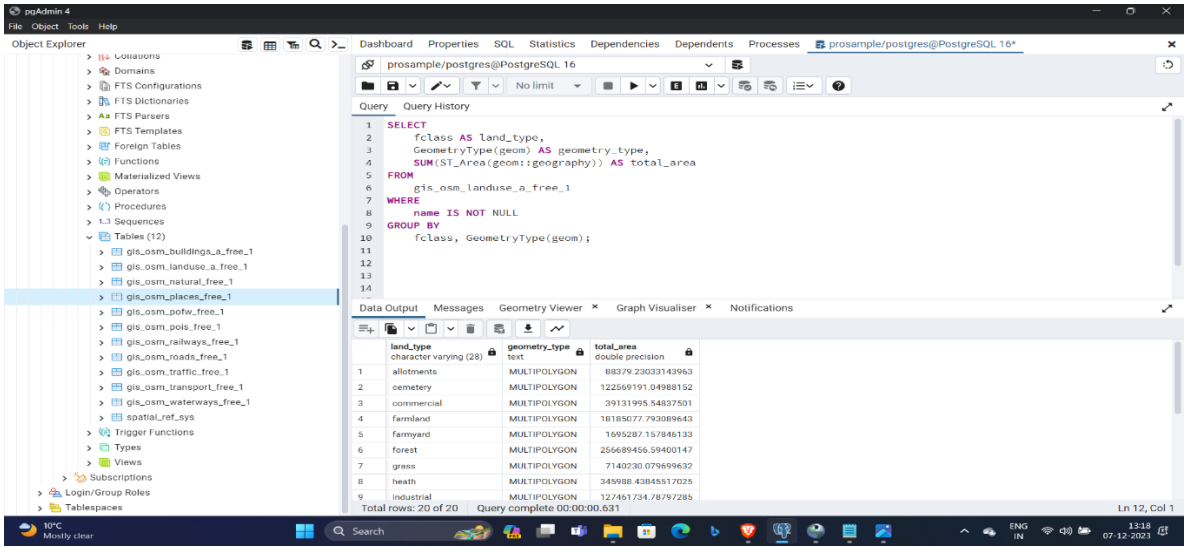
- It selects the name of each building, referring to it as building_name in the output.
- It uses the GeometryType function to get the type of geometry of the building's shape (e.g., POLYGON, MULTIPOLYGON) and labels this as geometry_type in the output.
- It calculates the area of each building's geometry, treating the geometry as geographical coordinates (using the geography data type) to get a more accurate area measurement on the Earth's surface. This value is referred to as areas in the output.
- The query only includes rows where the name column is not null, meaning it only considers buildings that have a name assigned to them.

```
SELECT
   name AS building_name,
   GeometryType(geom) AS geometry_type,
   ST_Area(geom::geography) AS areas
FROM
   gis_osm_buildings_a_free_1
WHERE
   name IS NOT NULL;
```



The query below will evaluate the land utilization across various sectors and geometrical shapes, summarizing their respective total areas.

```
SELECT
    fclass AS land_type,
    GeometryType(geom) AS geometry_type,
    SUM(ST_Area(geom::geography)) AS total_area
FROM
    gis_osm_landuse_a_free_1
WHERE
    name IS NOT NULL
GROUP BY
    fclass, GeometryType(geom);
```

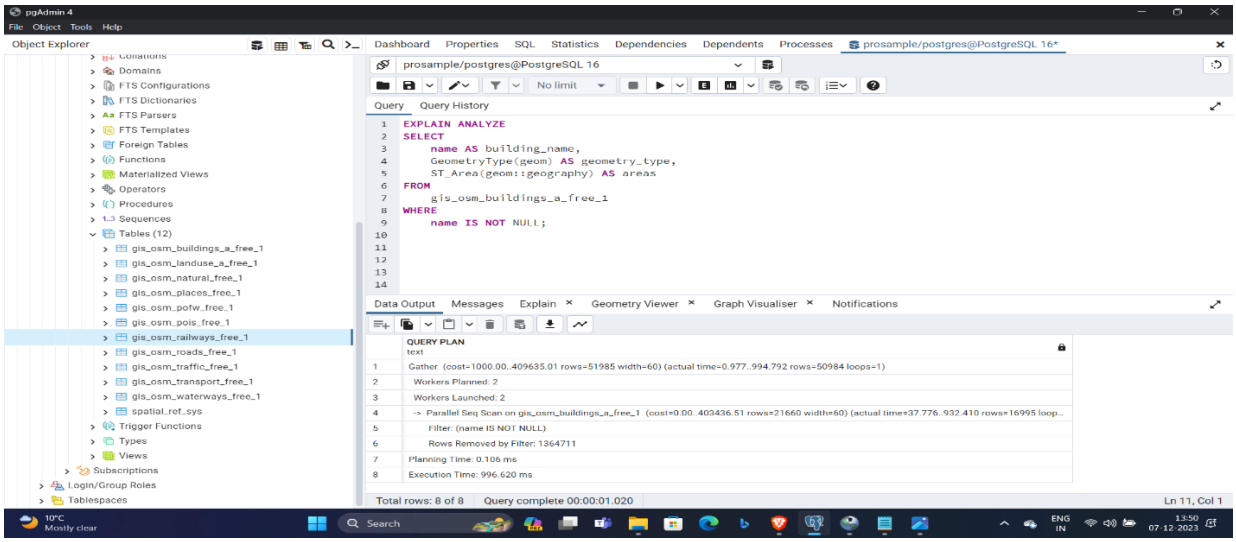


4. Analyze the queries

The query selects the name, geometry type, and area of buildings from the gis_osm_buildings_a_free_1 table, excluding entries without a building name.

The query utilizes the GeometryType function to identify the spatial geometry type of each entry .It also calculates the area in square meters for each geometry using the ST_Area function on the geom column cast to geography type, assuming the data is stored using a geographic coordinate system.

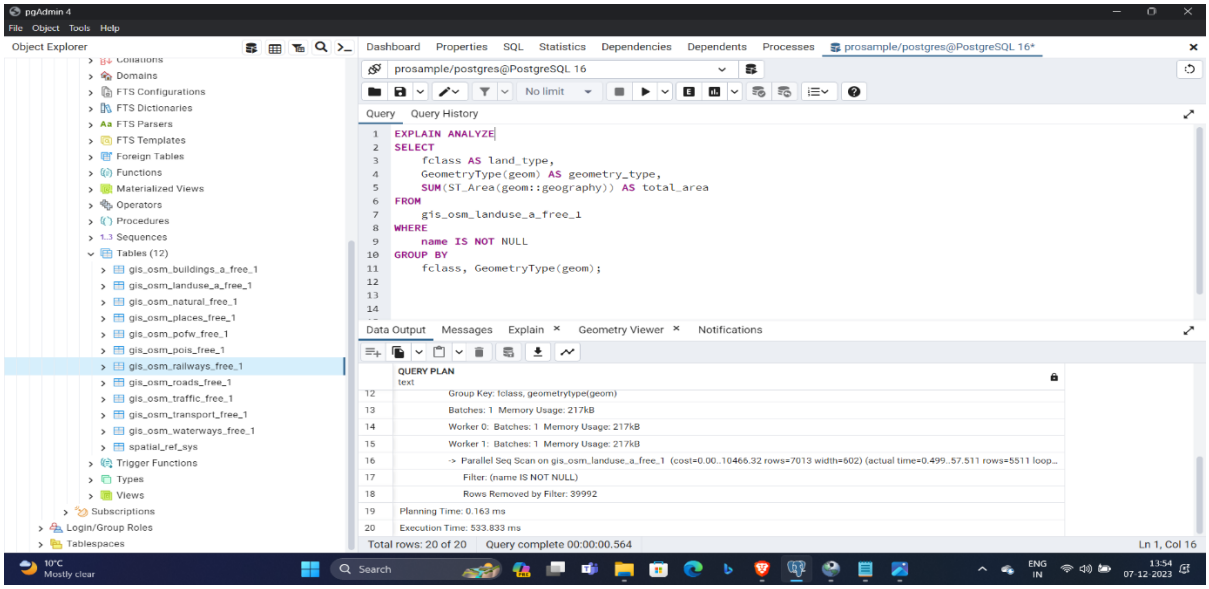
```
EXPLAIN ANALYZE
SELECT
    name AS building_name,
    GeometryType(geom) AS geometry_type,
    ST_Area(geom::geography) AS areas
FROM
    gis_osm_buildings_a_free_1
WHERE
    name IS NOT NULL;
```



The EXPLAIN ANALYZE output indicates that the query execution plan involves a parallel sequential scan (Parallel Seq Scan) of the gis_osm_buildings_a_free_1 table. Two worker processes are planned and launched to execute the query in parallel, aiming to improve performance. The filter name IS NOT NULL is applied to exclude records with null names. A significant number of rows (1,364,711) are removed by this filter, which suggests that many entries in the table do not have a building name.The cost estimate for the query starts at 1000.00 and goes up to 409635.01, with the actual time taken to execute the query ranging from **0.678 to 1056.140 milliseconds**. The estimated number of rows to be returned is 51,985, and the average width of the rows (in bytes) is expected to be 60.

2.

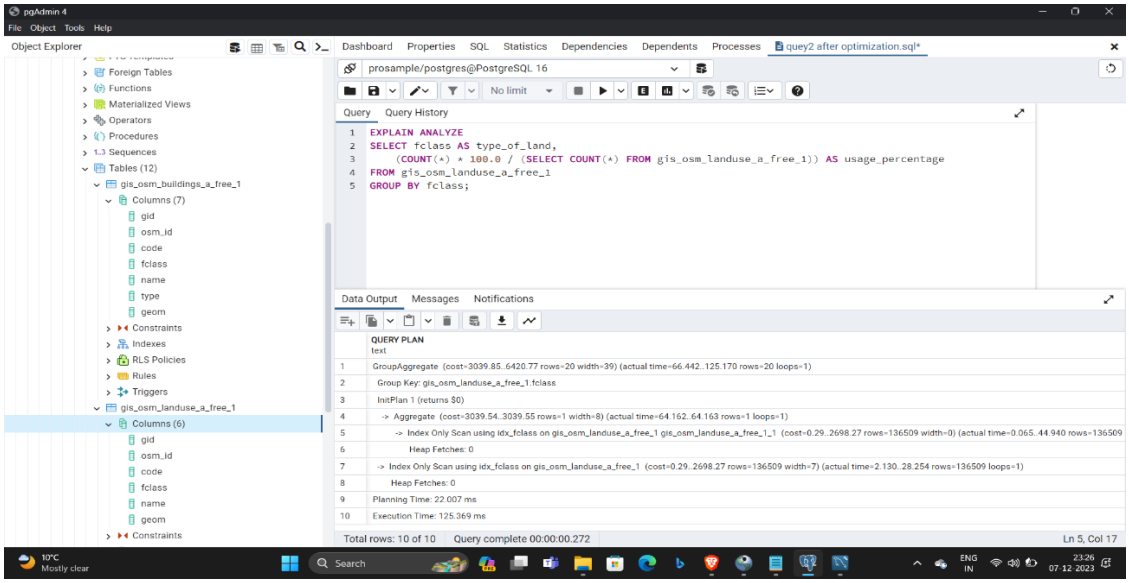
```
EXPLAIN ANALYZE
SELECT
   fclass AS land_type,
   GeometryType(geom) AS geometry_type,
   SUM(ST_Area(geom::geography)) AS total_area
FROM
   gis_osm_landuse_a_free_1
WHERE
   name IS NOT NULL
GROUP BY
   fclass, GeometryType(geom);
```



The query efficiently utilizes parallel processing to compute the total area for each land use category, grouping by fclass and geometry type. The majority of the time is spent in the actual execution of the parallel sequential scan and the aggregation process. The plan indicates that the data is well-suited for parallel processing, and the presence of a significant number of rows removed by the filter suggests that indexing the name column could further optimize performance for similar queries. The quicksort sorting method and the memory usage indicate that the dataset is managed efficiently in memory without needing to write to disk.
The total time taken to execute the query was about 759 ms, which is reasonable given the operations performed.

3.

```
EXPLAIN ANALYZE
SELECT fclass AS type_of_land,
   (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS
usage_percentage
FROM gis_osm_landuse_a_free_1
GROUP BY fclass;
```



The query effectively uses parallel processing to calculate the percentage usage of each land type. It benefits from an index-only scan for counting total rows, which improves efficiency. The actual sorting and aggregation steps are also done in memory without needing to access the disk, further contributing to the query's performance. The usage of parallel workers suggests that the table is large enough to warrant dividing the work, which PostgreSQL handles well.

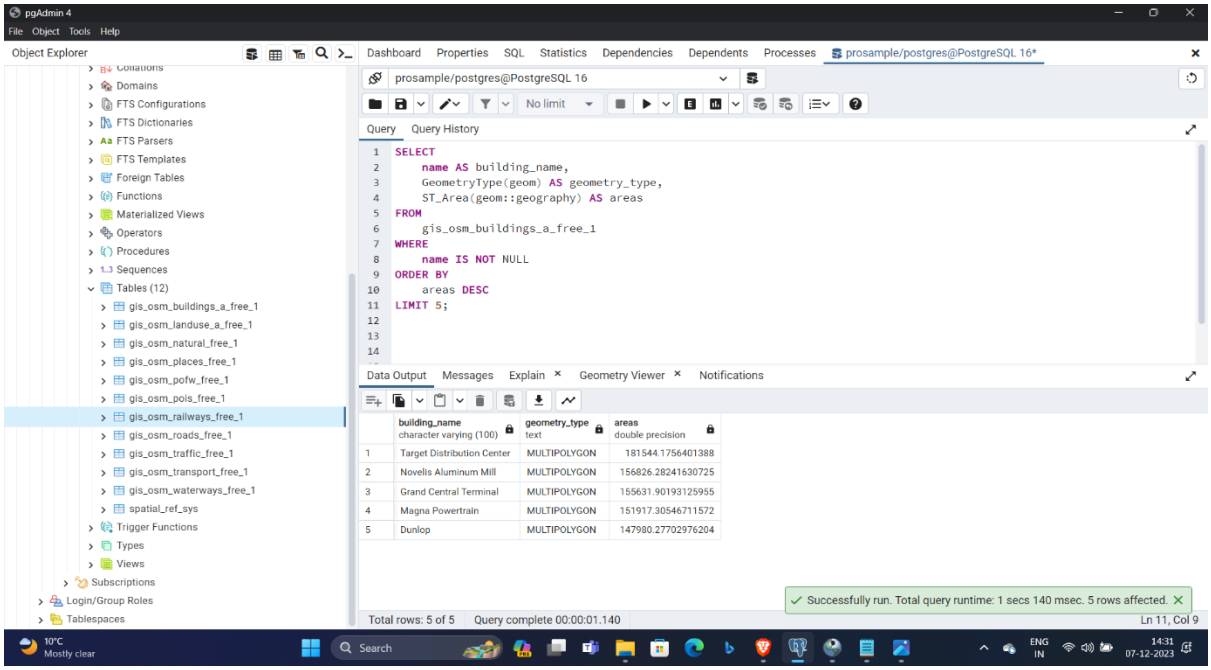
The resulting plan demonstrates that the query is well-optimized, with the most resource-intensive part being the sequential scans done in parallel. The quick execution suggests that the server is well-configured to handle this type of query. If this kind of query is common and the dataset grows, continued monitoring of execution plans will help maintain performance.

5. Sorting and Limit Executions

5.1

```
SELECT
   name AS building_name,
   GeometryType(geom) AS geometry_type,
   ST_Area(geom::geography) AS areas
FROM
   gis_osm_buildings_a_free_1
WHERE
   name IS NOT NULL
ORDER BY
   areas DESC
LIMIT 5;
```

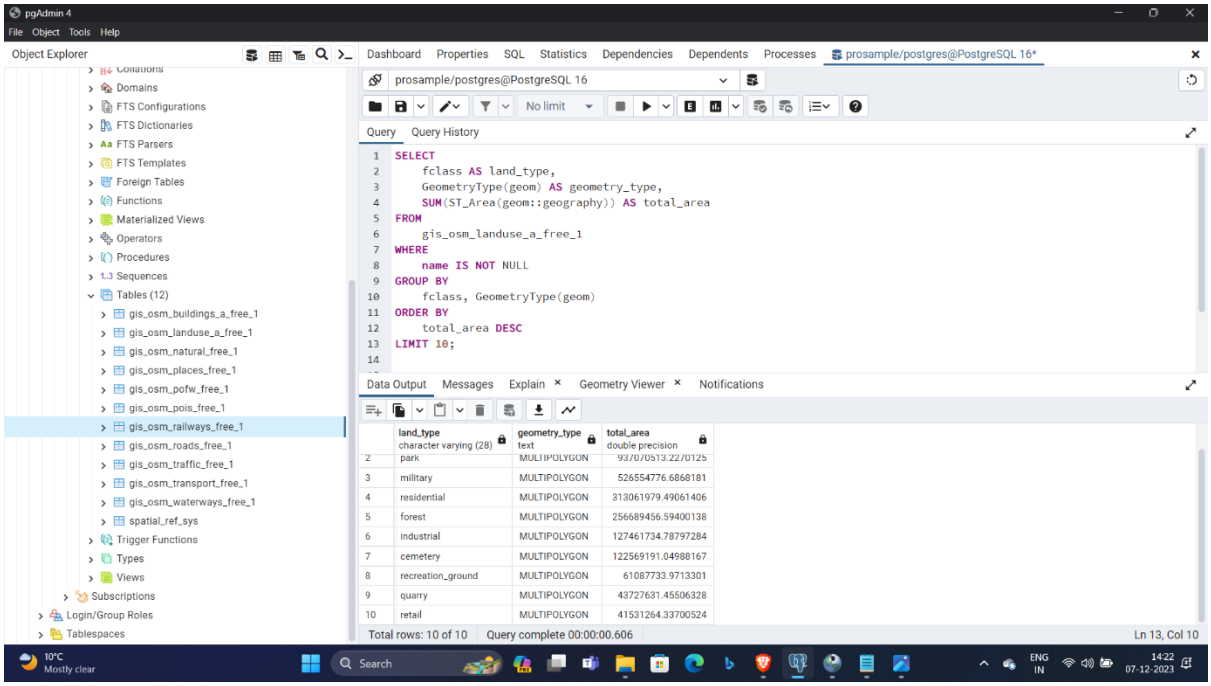
This query is designed to retrieve and list the top five largest named buildings from the given dataset, providing information about their names, geometric shapes, and areas.



5.2

```
SELECT
   fclass AS land_type,
   GeometryType(geom) AS geometry_type,
   SUM(ST_Area(geom::geography)) AS total_area
FROM
   gis_osm_landuse_a_free_1
WHERE
   name IS NOT NULL
GROUP BY
   fclass, GeometryType(geom)
ORDER BY
   total_area DESC
LIMIT
   10;
```

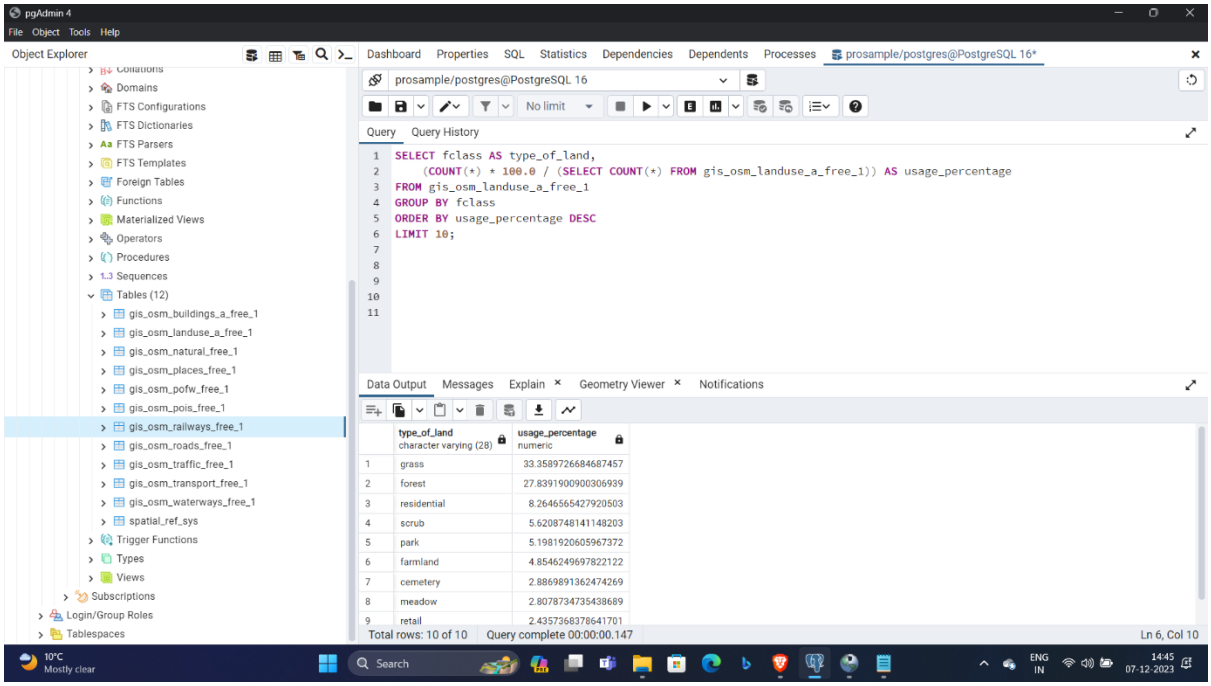
This query aims to identify and list the top ten land use types and their geometric shapes in the dataset, based on the total area, focusing specifically on named land features. It provides insights into which combinations of land use categories and geometric shapes occupy the most significant areas.



5.3

```
SELECT fclass AS type_of_land,
   (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS
usage_percentage
FROM gis_osm_landuse_a_free_1
GROUP BY fclass
ORDER BY usage_percentage DESC
LIMIT 10;
```

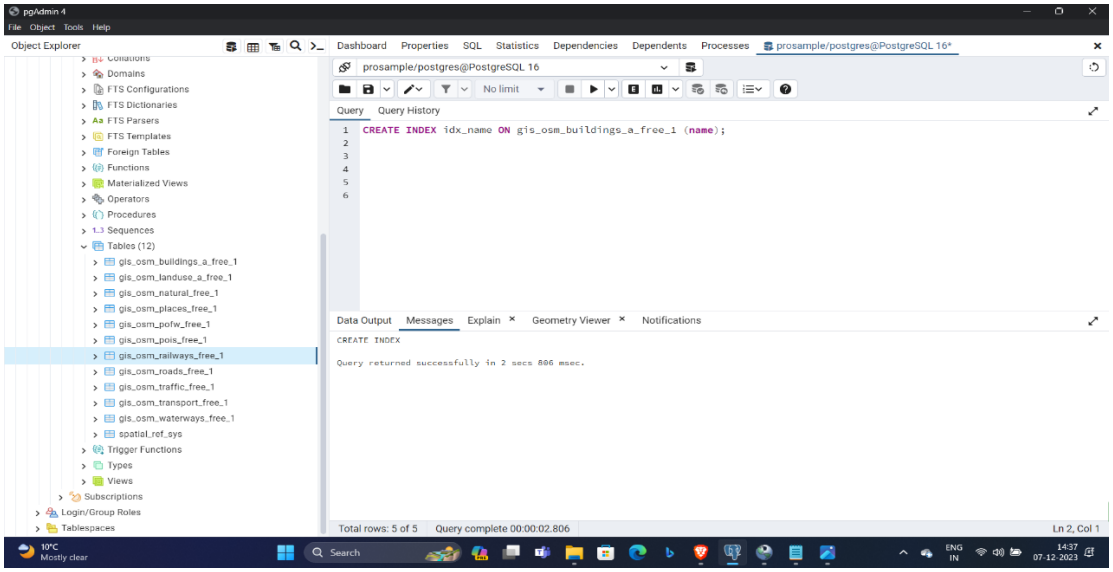
This SQL query analyzes land use categories within the gis_osm_landuse_a_free_1 table. It renames fclass to type_of_land and calculates each category's usage percentage by comparing its occurrence frequency against the total record count in the table. The data is grouped by land use type, and the usage percentages are computed to reflect how prevalent each type is within the overall dataset. The results are then ordered in descending order to showcase the most common land use types.

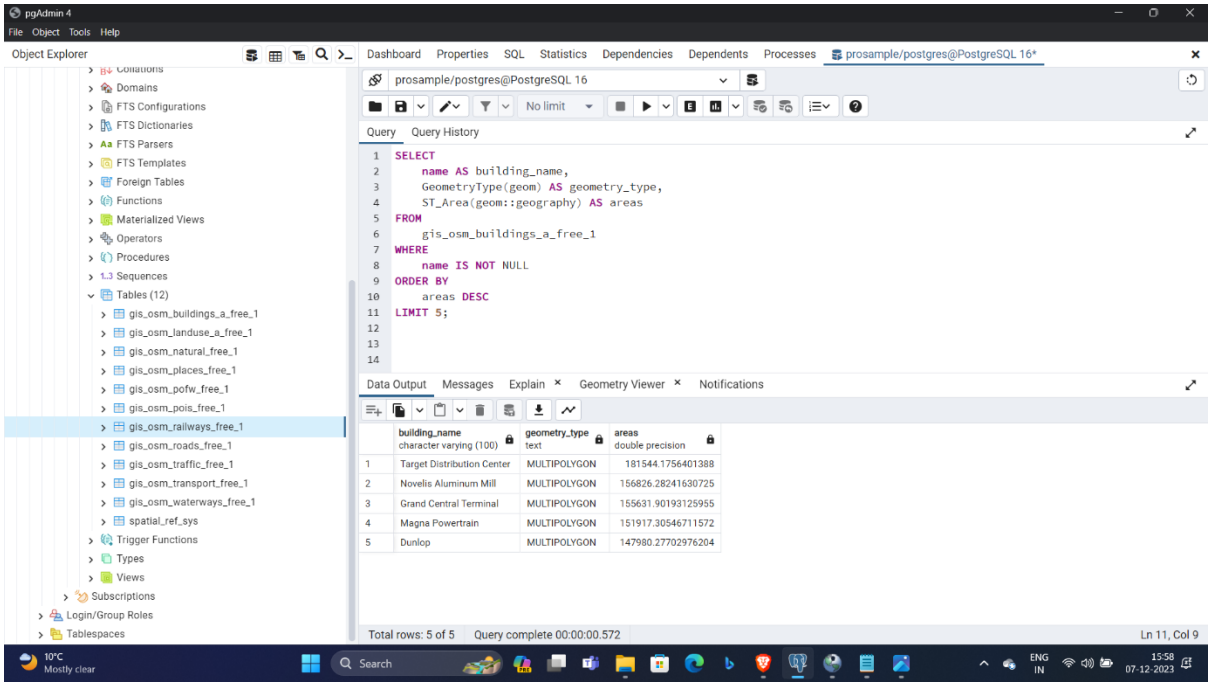


6. Optimize the queries to speed up execution time

6.1

```
CREATE INDEX idx_name ON gis_osm_buildings_a_free_1 (name);

SELECT
    name AS building_name,
    GeometryType(geom) AS geometry_type,
    ST_Area(geom::geography) AS areas
FROM
    gis_osm_buildings_a_free_1
WHERE
    name IS NOT NULL
ORDER BY
    areas DESC
LIMIT 5;
```

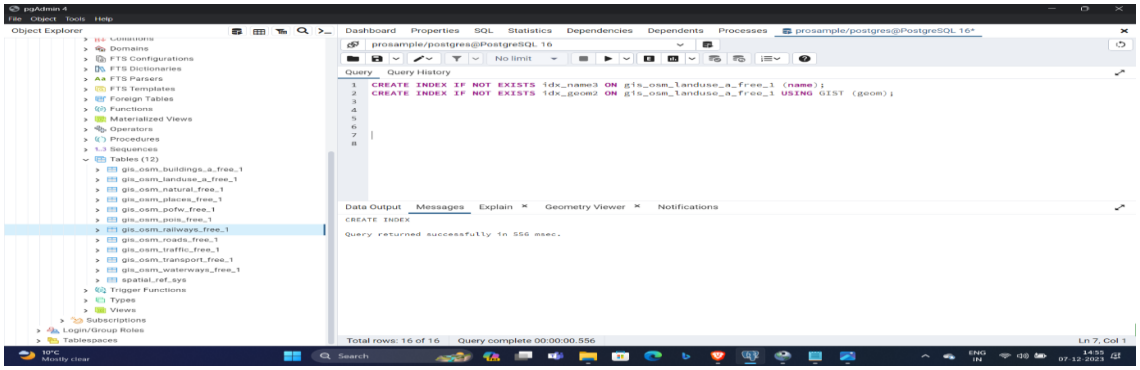


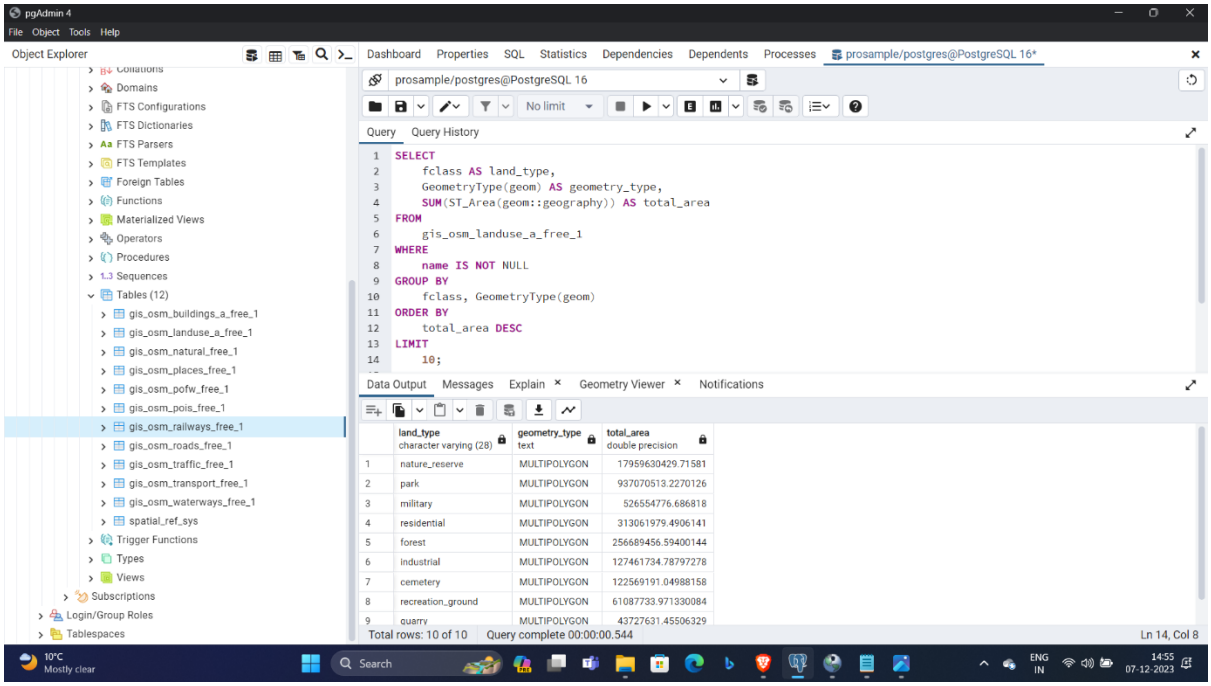


After creating the index the execution time will be saved by 564 milliseconds.

6.2

```
CREATE INDEX IF NOT EXISTS idx_name3 ON gis_osm_landuse_a_free_1 (name);
CREATE INDEX IF NOT EXISTS idx_geom2 ON gis_osm_landuse_a_free_1 USING GIST
(geom);

SELECT
   fclass AS land_type,
   GeometryType(geom) AS geometry_type,
   SUM(ST_Area(geom::geography)) AS total_area
FROM
   gis_osm_landuse_a_free_1
WHERE
   name IS NOT NULL
GROUP BY
   fclass, GeometryType(geom)
ORDER BY
   total_area DESC
LIMIT
   10;
```

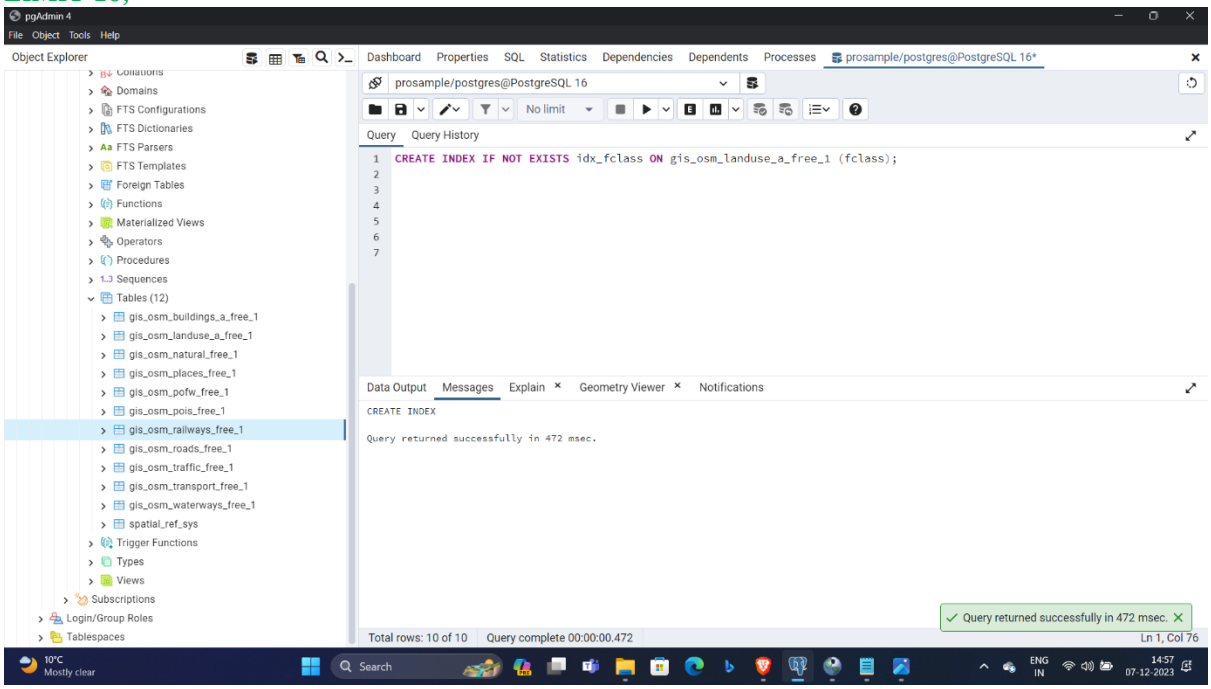


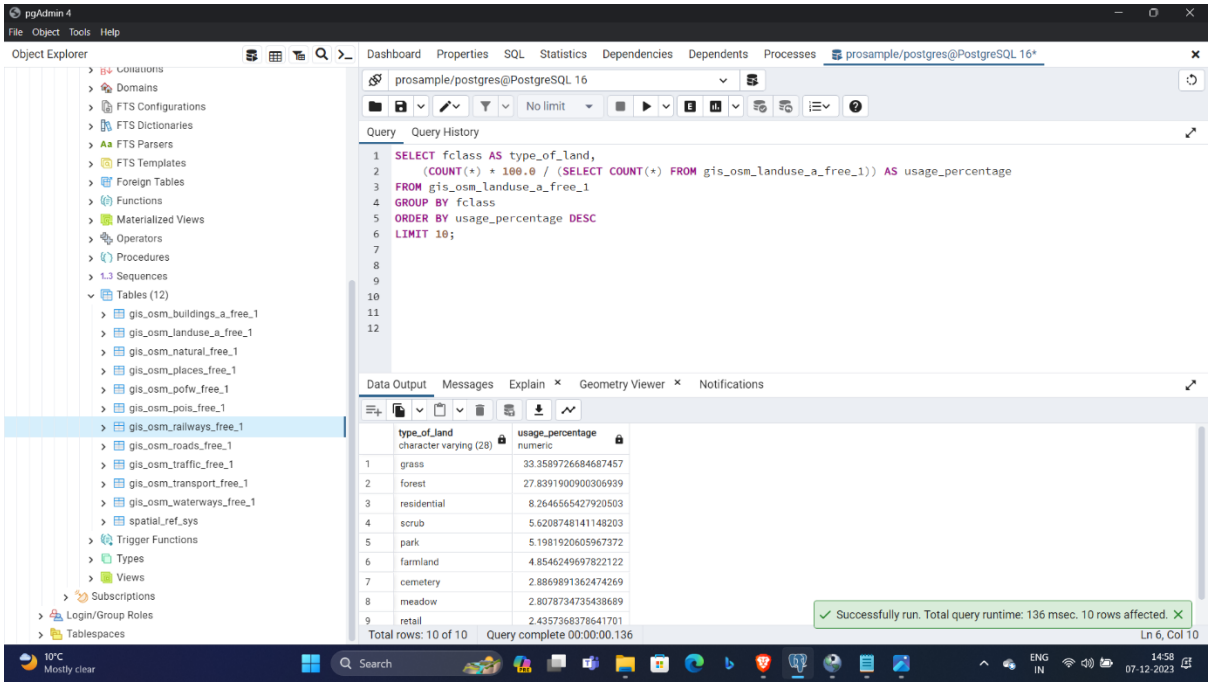


A reduction of 60 milliseconds in execution time represents a substantial enhancement in processing speed.

5.3

```
CREATE INDEX IF NOT EXISTS idx_fclass ON gis_osm_landuse_a_free_1 (fclass);

SELECT fclass AS type_of_land,
    (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS
usage_percentage
FROM gis_osm_landuse_a_free_1
GROUP BY fclass
ORDER BY usage_percentage DESC
LIMIT 10;
```



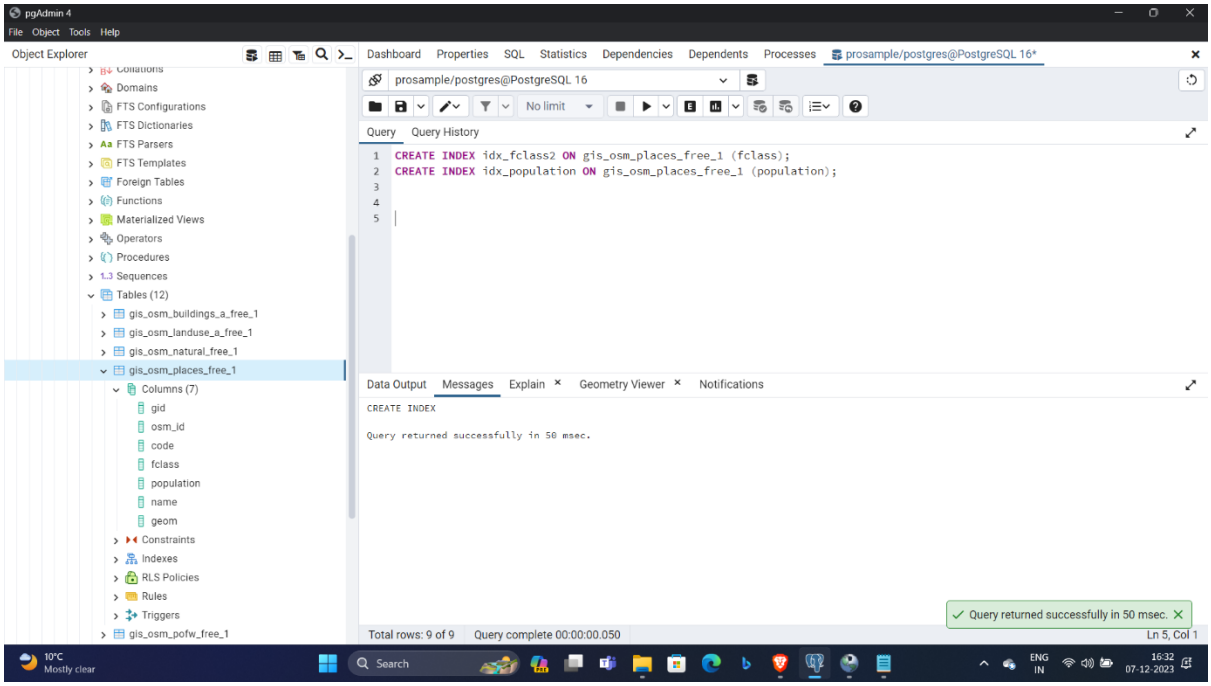


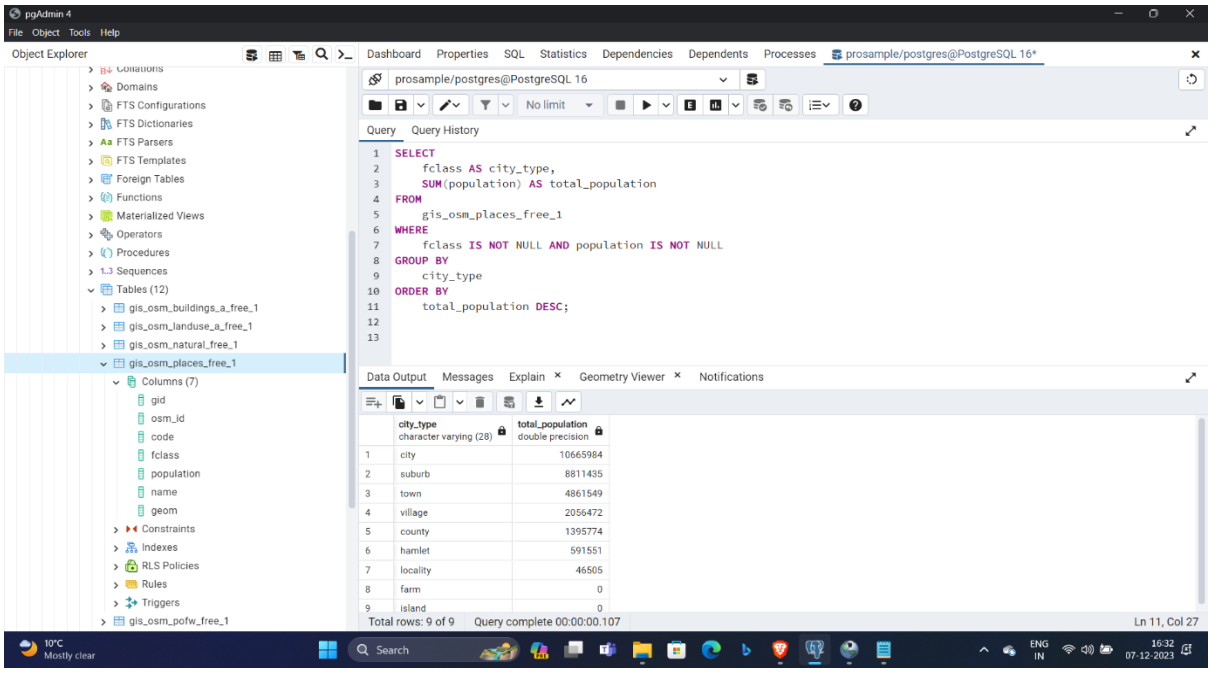
A reduction of 20 milliseconds in execution time represents a substantial enhancement in processing speed.

7. N-Optimization of queries

7.1

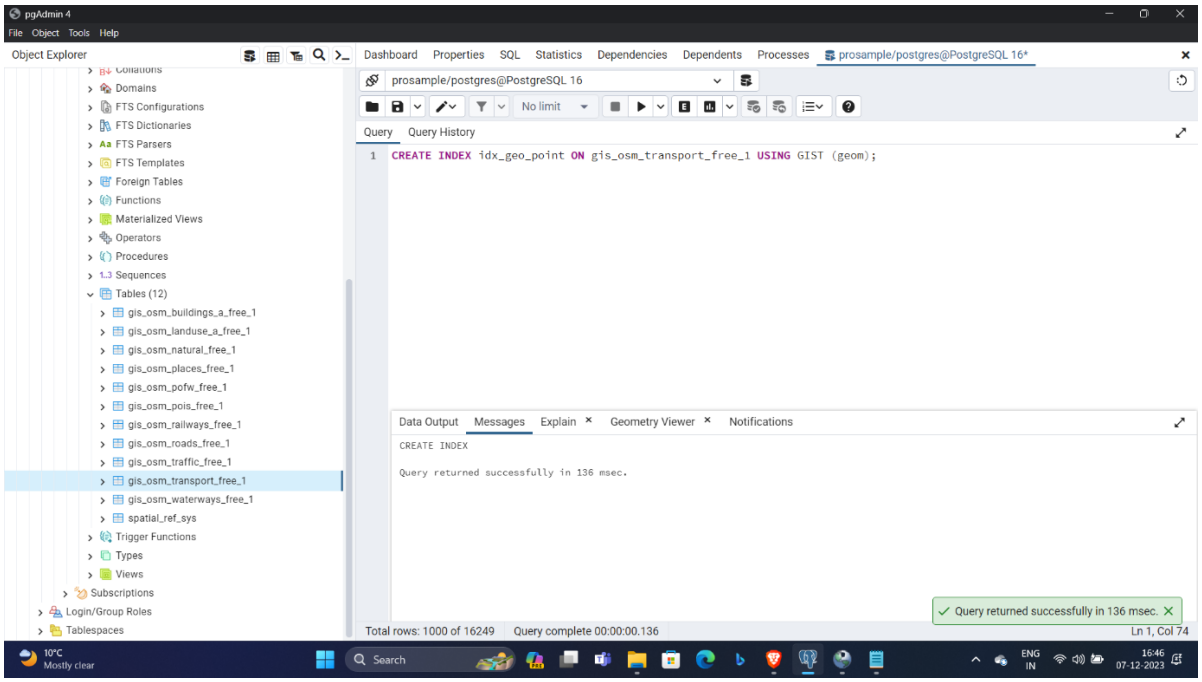
```
-- Create an index on the fclass column
CREATE INDEX idx_fclass2 ON gis_osm_places_free_1 (fclass);

-- Create an index on the population column
CREATE INDEX idx_population ON gis_osm_places_free_1 (population);

SELECT
    fclass AS city_type,
    SUM(population) AS total_population
FROM
    gis_osm_places_free_1
WHERE
    fclass IS NOT NULL AND population IS NOT NULL
GROUP BY
    city_type
ORDER BY
    total_population DESC;
```

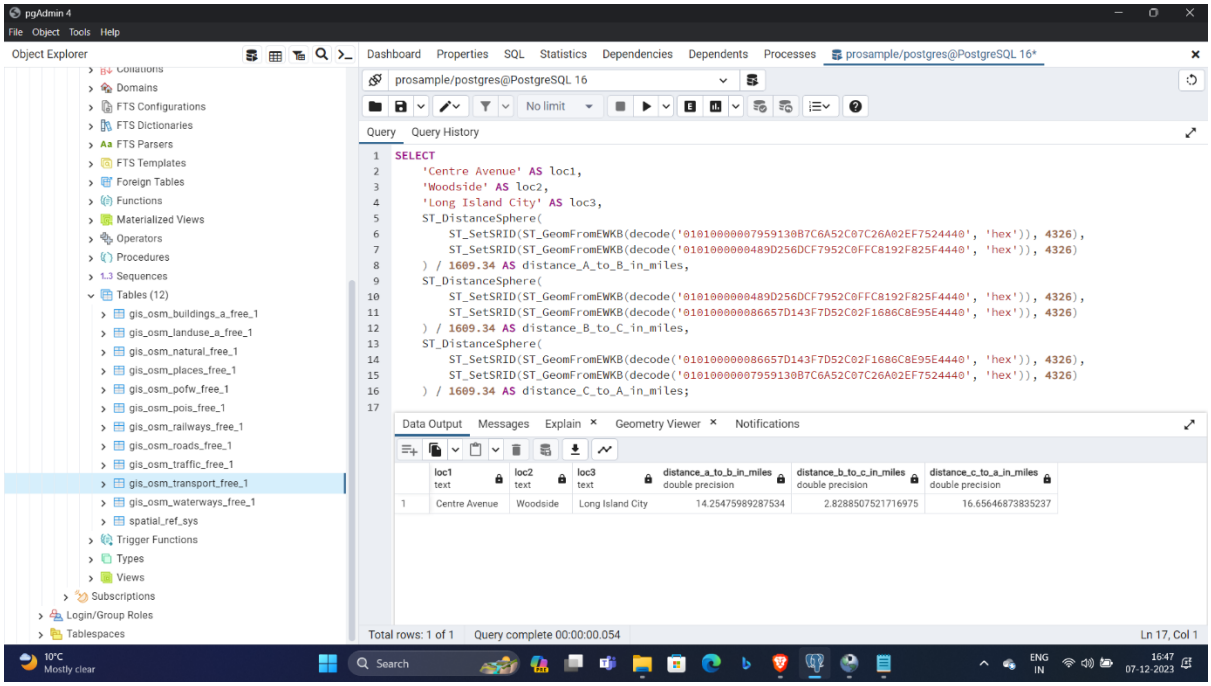

pgAdmin 4
File Object Tools Help
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*
prosample/postgres@PostgreSQL 16
No limit
Query Query History
CREATE INDEX idx_fclass2 ON gis_osm_places_free_1 (fclass);
CREATE INDEX idx_population ON gis_osm_places_free_1 (population);
Data Output Messages Explain Geometry Viewer Notifications
CREATE INDEX
Query returned successfully in 50 msec.
Query returned successfully in 50 msec.
Total rows: 9 of 9
Query complete 00:00:00.050
Ln 5, Col 1
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (12)
gis_osm_buildings_a_free_1
gis_osm_landuse_a_free_1
gis_osm_natural_free_1
gis_osm_places_free_1
Columns (7)
gid
osm_id
code
fclass
population
name
geom
Constraints
Indexes
RLS Policies
Rules
Triggers
gis_osm_pofw_free_1
10°C
Mostly clear
Search
ENG
IN
16:32
07-12-2023


pgAdmin 4
File Object Tools Help
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*
prosample/postgres@PostgreSQL 16
No limit
Query Query History
SELECT
fclass AS city_type,
SUM(population) AS total_population
FROM
gis_osm_places_free_1
WHERE
fclass IS NOT NULL AND population IS NOT NULL
GROUP BY
city_type
ORDER BY
total_population DESC;
Data Output Messages Explain Geometry Viewer Notifications
city_type character varying (28)
total_population double precision
1 city 10665984
2 suburb 8811435
3 town 4861549
4 village 2056472
5 county 1395774
6 hamlet 591551
7 locality 46505
8 farm 0
9 island 0
Total rows: 9 of 9
Query complete 00:00:00.107
Ln 11, Col 27
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (12)
gis_osm_buildings_a_free_1
gis_osm_landuse_a_free_1
gis_osm_natural_free_1
gis_osm_places_free_1
Columns (7)
gid
osm_id
code
fclass
population
name
geom
Constraints
Indexes
RLS Policies
Rules
Triggers
gis_osm_pofw_free_1
10°C
Mostly clear
Search
ENG
IN
16:32
07-12-2023

7.2

```
CREATE INDEX idx_geo_point ON gis_osm_transport_free_1 USING GIST (geom);

SELECT
    'Centre Avenue' AS loc1,
    'Woodside' AS loc2,
    'Long Island City' AS loc3,
    ST_DistanceSphere(

ST_SetSRID(ST_GeomFromEWKB(decode('01010000007959130B7C6A52C07C26A02EF7524440'
, 'hex')), 4326),

ST_SetSRID(ST_GeomFromEWKB(decode('0101000000489D256DCF7952C0FFC8192F825F4440'
, 'hex')), 4326)
    ) / 1609.34 AS distance_A_to_B_in_miles,
    ST_DistanceSphere(

ST_SetSRID(ST_GeomFromEWKB(decode('0101000000489D256DCF7952C0FFC8192F825F4440'
, 'hex')), 4326),

ST_SetSRID(ST_GeomFromEWKB(decode('010100000086657D143F7D52C02F1686C8E95E4440',
'hex')), 4326)
    ) / 1609.34 AS distance_B_to_C_in_miles,
    ST_DistanceSphere(

ST_SetSRID(ST_GeomFromEWKB(decode('010100000086657D143F7D52C02F1686C8E95E4440',
'hex')), 4326),

ST_SetSRID(ST_GeomFromEWKB(decode('01010000007959130B7C6A52C07C26A02EF7524440'
, 'hex')), 4326)
    ) / 1609.34 AS distance_C_to_A_in_miles;
```




pgAdmin 4
File Object Tools Help
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*
prosample/postgres@PostgreSQL 16
No limit
Query Query History
SELECT
'Centre Avenue' AS loc1,
'Woodside' AS loc2,
'Long Island City' AS loc3,
ST_DistanceSphere(
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000795913OB7C6A52C07C26A02EF7524440', 'hex')), 4326),
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000489D256DCF7952C0FFC8192F825F4440', 'hex')), 4326)
) / 1609.34 AS distance_A_to_B_in_miles,
ST_DistanceSphere(
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000489D256DCF7952C0FFC8192F825F4440', 'hex')), 4326),
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000866S7D143F7D52C02F1686C8E95E4440', 'hex')), 4326)
) / 1609.34 AS distance_B_to_C_in_miles,
ST_DistanceSphere(
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000866S7D143F7D52C02F1686C8E95E4440', 'hex')), 4326),
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000795913OB7C6A52C07C26A02EF7524440', 'hex')), 4326)
) / 1609.34 AS distance_C_to_A_in_miles;
Data Output Messages Explain Geometry Viewer Notifications
loc1 text
loc2 text
loc3 text
distance_a_to_b_in_miles double precision
distance_b_to_c_in_miles double precision
distance_c_to_a_in_miles double precision
1 Centre Avenue Woodside Long Island City 14.25475989287534 2.8288507521716975 16.65646873835237
Total rows: 1 of 1
Query complete 00:00:00.054
Ln 17, Col 1
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (12)
gis_osm_buildings_a_free_1
gis_osm_landuse_a_free_1
gis_osm_natural_free_1
gis_osm_places_free_1
gis_osm_pofw_free_1
gis_osm_pois_free_1
gis_osm_railways_free_1
gis_osm_roads_free_1
gis_osm_traffic_free_1
gis_osm_transport_free_1
gis_osm_waterways_free_1
spatial_ref_sys
Trigger Functions
Types
Views
Subscriptions
Login/Group Roles
Tablespaces
10°C
Mostly clear
Search
ENG
IN
16:47
07-12-2023