



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.

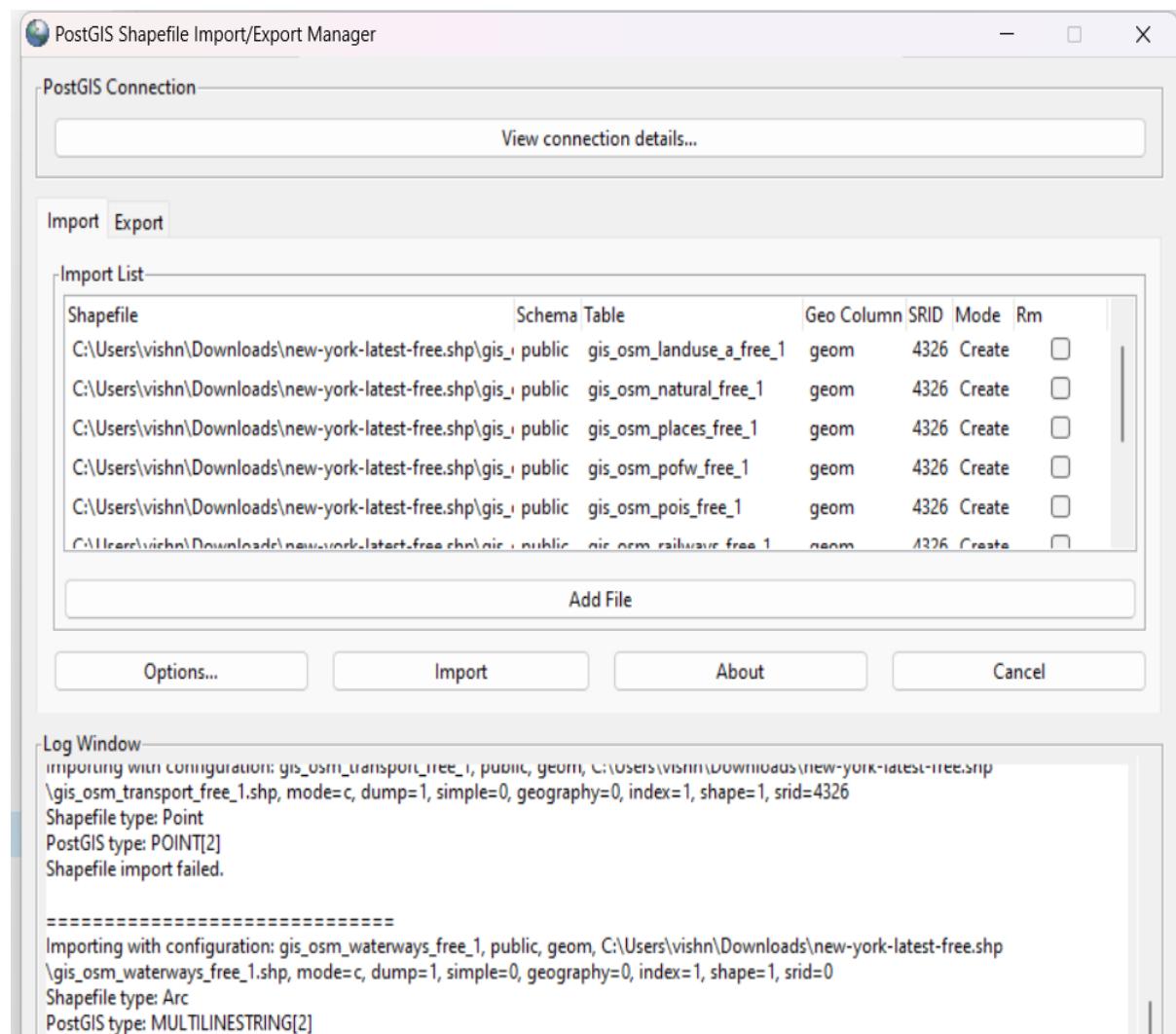


Table schemas:

Query Query History

```
1  SELECT
2      column_name,
3      data_type,
4      is_nullable
5  FROM
6      information_schema.columns
7 WHERE
8      table_name = 'gis_osm_buildings_a_free_1' |
9
10
11
```

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

```
1  SELECT
2      column_name,
3      data_type,
4      is_nullable
5  FROM
6      information_schema.columns
7 WHERE
8      table_name = 'gis_osm_landuse_a_free_1'
9
10
11
```

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

```

1  SELECT
2      column_name,
3      data_type,
4      is_nullable
5  FROM
6      information_schema.columns
7  WHERE
8      table_name = 'gis_osm_roads_free_1'
9
10
11

```

Data Output Messages Notifications

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

Total rows: 12 of 12

Query complete 00:00:00.185

Query Query History

```

1  SELECT
2      column_name,
3      data_type,
4      is_nullable
5  FROM
6      information_schema.columns
7  WHERE
8      table_name = 'gis_osm_natural_free_1'
9
10
11

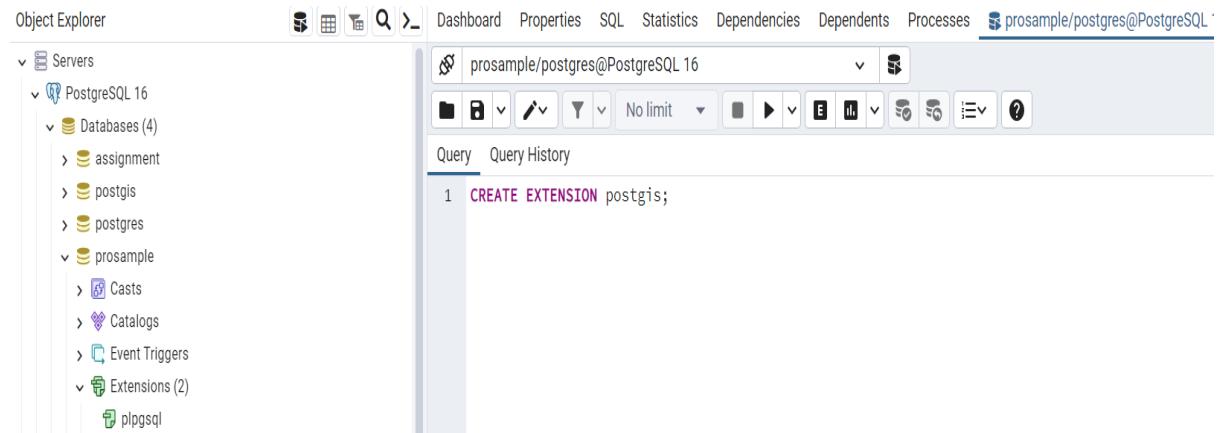
```

Data Output Messages Notifications

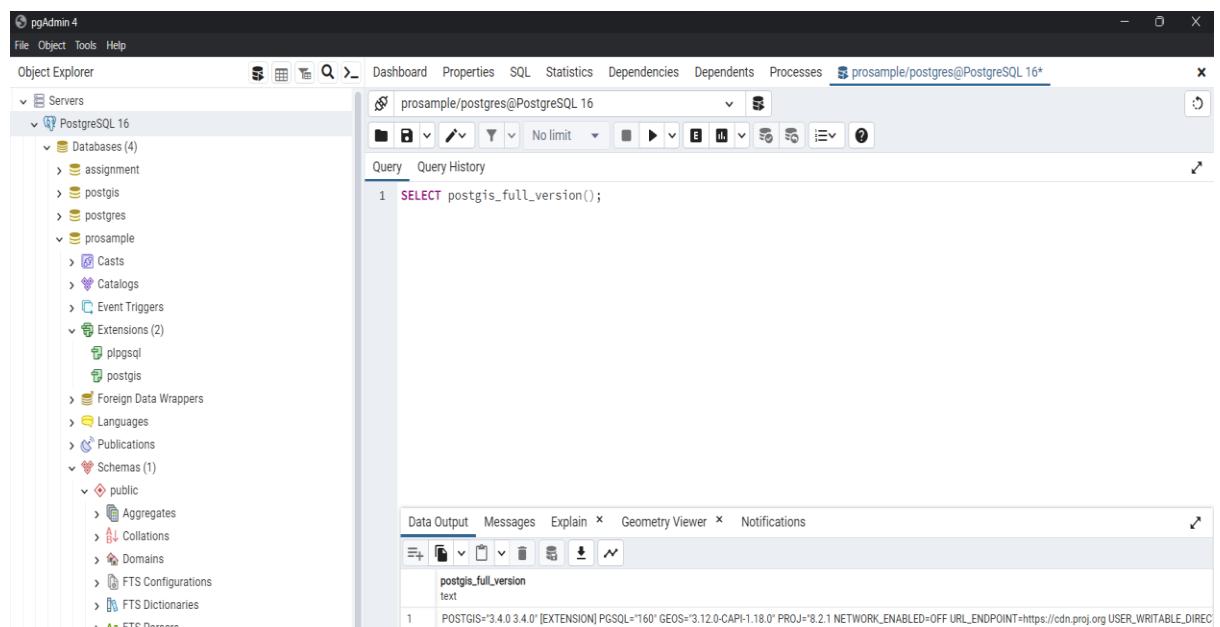
	column_name	data_type	is_nullable
1	name	character varying	character varying (3)
2	gid	integer	NO
3	osm_id	character varying	YES
4	code	smallint	YES
5	fclass	character varying	YES
6	name	character varying	YES
7	geom	USER-DEFINED	YES

Creating postgis extension

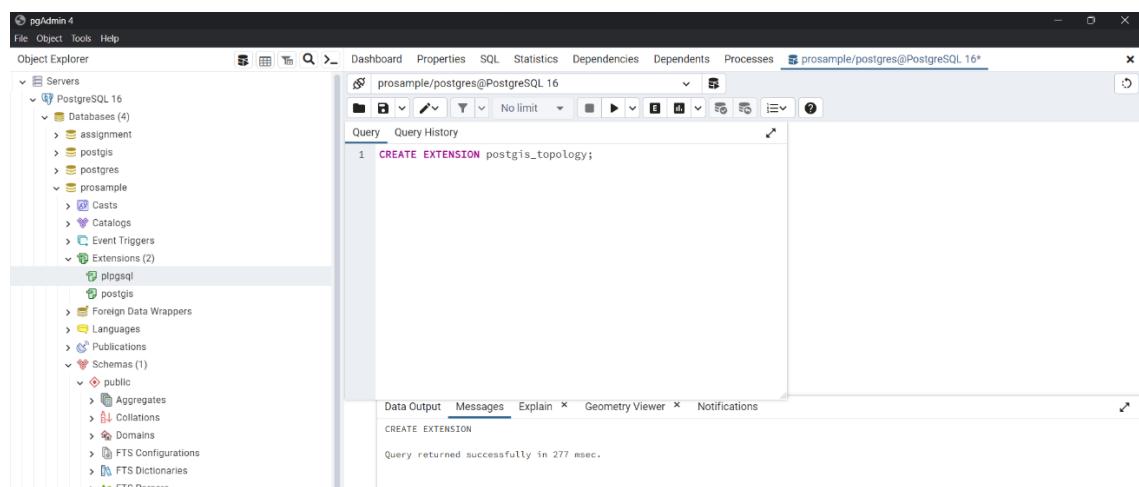
`CREATE EXTENSION postgis;`



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;
```

```
pgAdmin 4
File Object Tools Help
Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16+
Query Query History
1 SELECT type AS Building_type_in_NY, COUNT (*) AS count
2 FROM gis_osm_buildings_a_free_1
3 GROUP BY type;
4
5
6
7
8
9
```

building_type_in_ny	count
abandoned	6
academic	4
administrative	1
airport	2
allotment_house	1
ambulance	2
AP	1
apartments	38726
arts_center	1

Total rows: 228 of 228 Query complete 00:00:02.134 Ln 3, Col 15

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;
```

```
pgAdmin 4
File Object Tools Help
Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16+
Query Query History
1 SELECT fclass AS type_of_land,
2 (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS usage_percentage
3 FROM gis_osm_landuse_a_free_1
4 GROUP BY fclass;
5
6
7
8
9
10
```

type_of_land	usage_percentage
allotments	0.08570863459552117443
cemetery	2.8869981362474269
commercial	1.289292810950194
farmland	4.8546249697822122
farmyard	0.3582181394633208799
forest	27.8391909903069939
grass	33.35897266464687457
heath	0.17508003135324410845
industrial	1.440174466663883

Total rows: 20 of 20 Query complete 00:00:00.210 Successfully run. Total query runtime: 210 msec. 20 rows affected. Ln 9, Col 1

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;
```

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree structure of database objects. The right pane is the Query Editor, showing the executed SQL query and its results. The results table shows data from the 'gis_osm_roads_free_1' table, specifically for rows where 'name' is not null. The results are as follows:

road_types_in_ny	name	maxspeed
motorway	Niagara Expressway	88
motorway	South Grand Island Bridge	88
motorway	Youngmann Expressway	88
motorway	Niagara Expressway	104
motorway	Youngmann Expressway	88
motorway	South Grand Island Bridge	88
motorway	Niagara Expressway	88
motorway	North Grand Island Bridge	88
motorway	New York State Thruway	104

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;
```

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*

Query Query History

```

1 SELECT
2   'Burger King' AS loc1,
3   'New City Library' AS loc2,
4   ST_DistanceSphere(
5     ST_SetSRID(ST_GeomFromEWKB(decode('0101000000848DA152B7F52C00498E8E225924440', 'hex')), 4326),
6     ST_SetSRID(ST_GeomFromEWKB(decode('010100000011CED013B7F52C08B47F3114294440', 'hex')), 4326)
7   ) / 1609.34 AS distance_in_miles;
8
9
10
11
12
13
14

```

Data Output Messages Geometry Viewer Graph Visualiser Notifications

	loc1	loc2	distance_in_miles
1	Burger King	New City Library	1.1436284528129543

Total rows: 1 of 1 Query complete 00:00:00.144 Ln 10, Col 1

10°C Mostly clear

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;
```

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*

Query Query History

```

1 SELECT
2   'South Ferry'as loc1,
3   'Centre Avenue' as loc2,
4
5   ST_DistanceSphere(
6     ST_SetSRID(ST_GeomFromEWKB(decode('01010000006CBBBF304F1452C03843BB54B6854440', 'hex')), 4326),
7     ST_SetSRID(ST_GeomFromEWKB(decode('01010000007959130B7C6A52C07C6A02EF7524440', 'hex')), 4326)
8   ) / 1609.34 AS distance_in_miles;
9
10
11
12
13
14

```

Data Output Messages Graph Visualiser Notifications

	loc1	loc2	distance_in_miles
1	South Ferry	Centre Avenue	75.51880881181728

Total rows: 1 of 1 Query complete 00:00:00.081 Ln 10, Col 1

10°C Mostly clear

```

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;

```

	loc1	loc2	distance_in_miles
1	Millbrook	New York	75.96089424924503

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;

```

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*

Query Query History

```

1 SELECT
2     name AS building_name,
3     GeometryType(geom) AS geometry_type,
4     ST_Area(geom::geography) AS areas
5 FROM
6     gis_osm_buildings_a_free_1
7 WHERE
8     name IS NOT NULL;
9
10
11
12
13
14

```

Data Output Messages Geometry Viewer Graph Visualiser Notifications

building_name	geometry_type	areas
New York State Museum, Library, and Archives	MULTIPOLYGON	14497.300321908202
Soldiers' and Sailors' Arch	MULTIPOLYGON	174.25308330052128
Marine Biology & Seismology	MULTIPOLYGON	2405.892132163048
Comstock Hall	MULTIPOLYGON	2464.6154929343
Biotechnology Building	MULTIPOLYGON	3244.102023573976
Teagle Hall	MULTIPOLYGON	4450.442030407488
Dale R Corson Hall	MULTIPOLYGON	2112.1115686114063
Theatre of the Youth	MULTIPOLYGON	821.5144543717906
Surry Building	MULTIPOLYGON	382.06157879158854

Total rows: 1000 of 10984 Query complete 00:00:01.199 Ln 8, Col 22

10°C Mostly clear 13:08 07-12-2023

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);

```

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*

Query Query History

```

1 SELECT
2     fclass AS land_type,
3     GeometryType(geom) AS geometry_type,
4     SUM(ST_Area(geom::geography)) AS total_area
5 FROM
6     gis_osm_landuse_a_free_1
7 WHERE
8     name IS NOT NULL
9 GROUP BY
10    fclass, GeometryType(geom);
11
12
13
14

```

Data Output Messages Geometry Viewer Graph Visualiser Notifications

land_type	geometry_type	total_area
allotments	MULTIPOLYGON	883.7923033143963
cemetery	MULTIPOLYGON	122569191.04988152
commercial	MULTIPOLYGON	39131995.54837501
farmland	MULTIPOLYGON	16185077.79308964
farmyard	MULTIPOLYGON	1695287.157846133
forest	MULTIPOLYGON	256689456.59400147
grass	MULTIPOLYGON	7140230.079699632
heath	MULTIPOLYGON	345988.43845517025
industrial	MULTIPOLYGON	127461734.78797285

Total rows: 20 of 20 Query complete 00:00:00.631 Ln 12, Col 1

10°C Mostly clear 13:15 07-12-2023

4. Analyze the queries

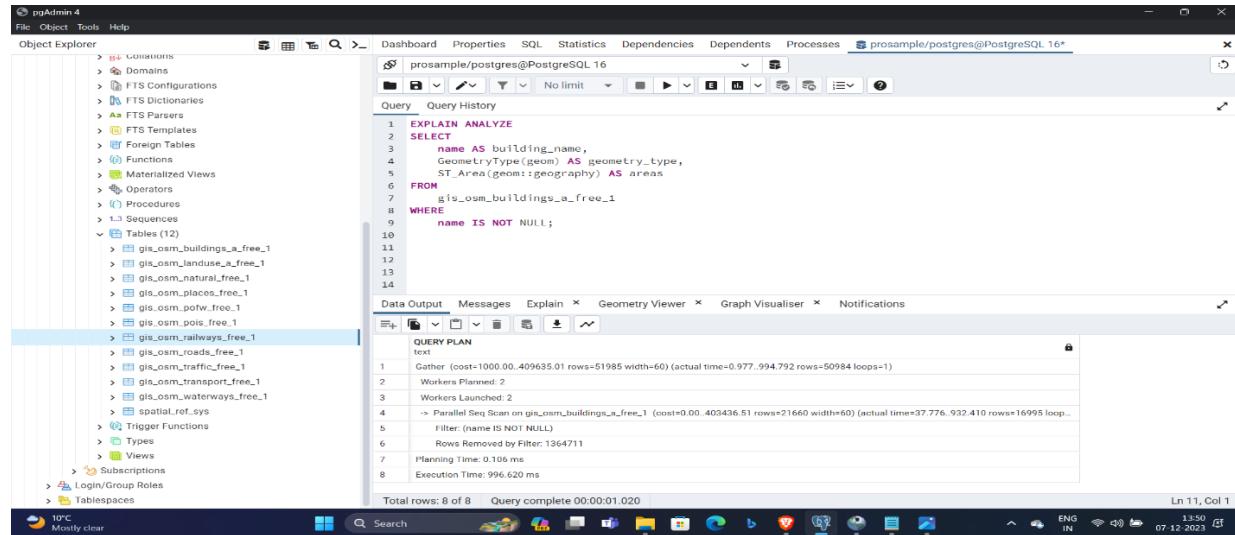
The query selects the name, geometry type, and area of buildings from the gis_osm_buildings_a_free_1 table, excluding entries without a building name.

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;
```



```
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;
```

QUERY PLAN

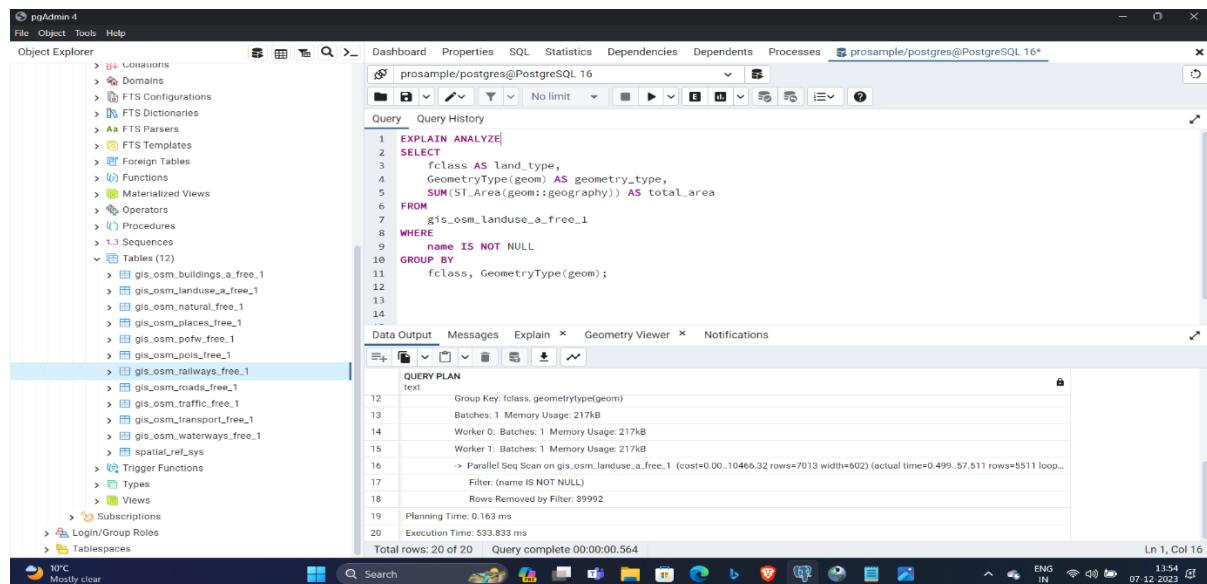
```
text
1 Gather (cost=1000.00..409635.01 rows=51985 width=60) (actual time=0.977..994.792 rows=50984 loops=1)
  2 Workers Planned: 2
  3 Workers Launched: 2
  4 -> Parallel Seq Scan on gis_osm_buildings_a_free_1 (cost=0.00..403436.51 rows=21660 width=60) (actual time=37.776..932.410 rows=16995 loops=1)
      Filter: (name IS NOT NULL)
      Rows Removed by Filter: 1364711
    Planning Time: 0.106 ms
    Execution Time: 996.620 ms
```

Total rows: 8 of 8 Query complete 00:00:01.020

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);
```

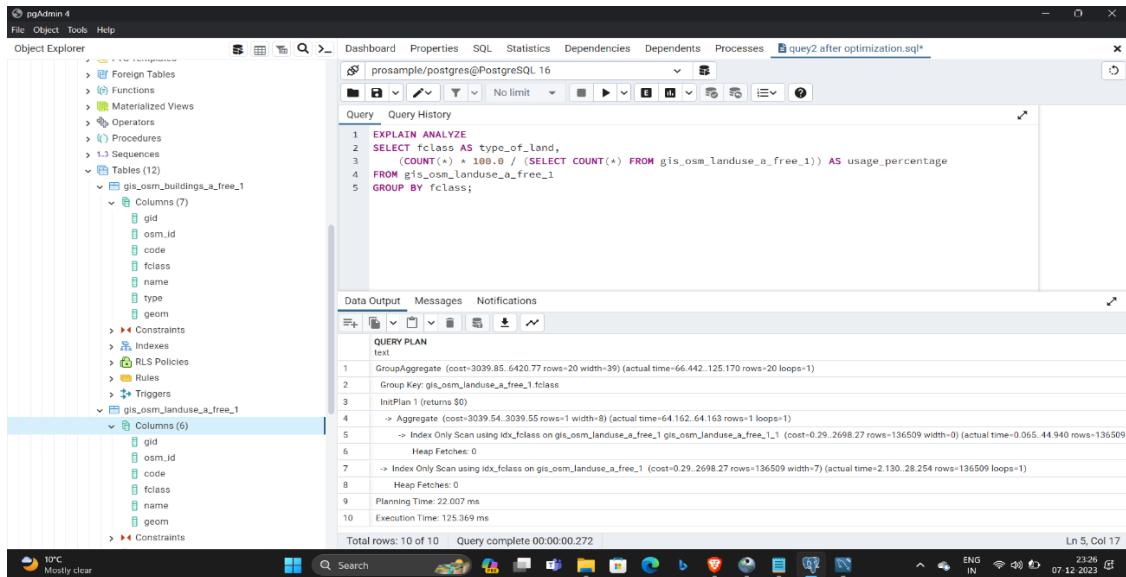


The query efficiently utilizes parallel processing to compute the total area for each land use category, grouping by fclass and geometry type. The majority of the time is spent in the actual execution of the parallel sequential scan and the aggregation process. The plan indicates that the data is well-suited 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

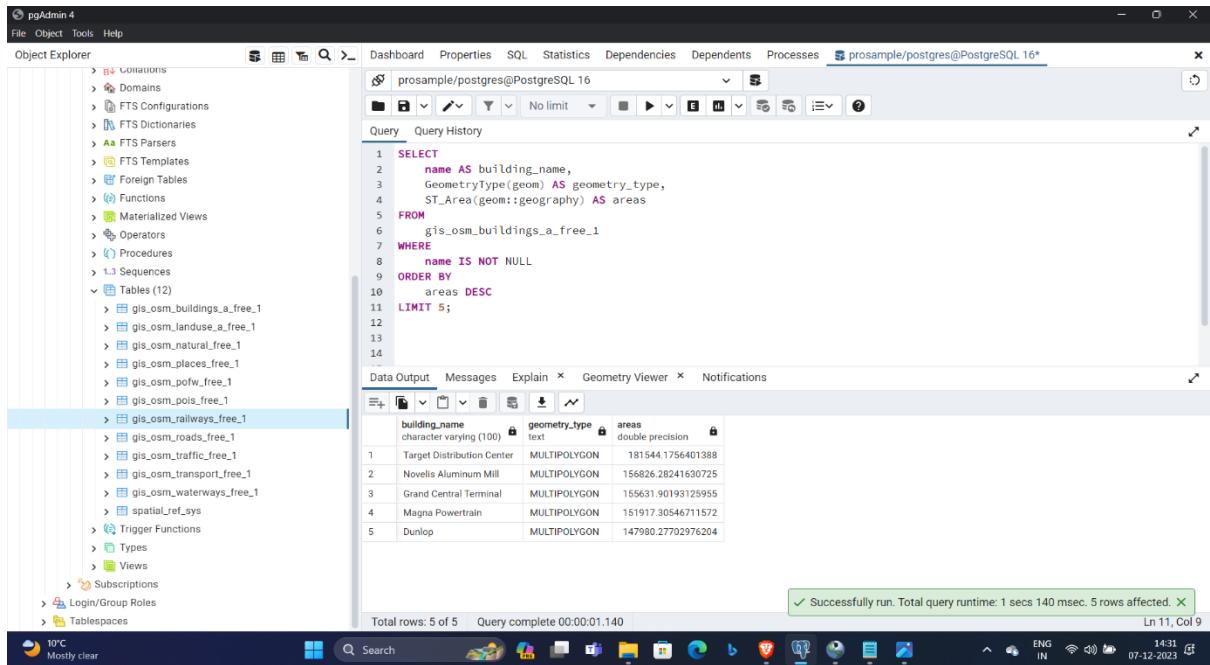
5.1

```

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.

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree view of database objects. The right pane contains a Query Editor window with the following SQL query:

```

1 SELECT
2     fclass AS land_type,
3     GeometryType(geom) AS geometry_type,
4     SUM(ST_Area(geom::geography)) AS total_area
5 FROM
6     gis_osm_landuse_a_free_1
7 WHERE
8     name IS NOT NULL
9 GROUP BY
10    fclass, GeometryType(geom)
11 ORDER BY
12    total_area DESC
13 LIMIT 10;

```

The Data Output tab shows the results of the query:

land_type	geometry_type	total_area
park	MULTIPOLYGON	93/0/0513.22/0125
military	MULTIPOLYGON	526554776.6668181
residential	MULTIPOLYGON	313061979.49061406
forest	MULTIPOLYGON	256689456.59400138
industrial	MULTIPOLYGON	127461734.78797284
cemetery	MULTIPOLYGON	122569191.04988167
recreation_ground	MULTIPOLYGON	6108773.971301
quarry	MULTIPOLYGON	43727631.45506328
retail	MULTIPOLYGON	41531264.33700524

Total rows: 10 of 10 | Query complete 00:00:00.606

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.

```

pgAdmin 4
File Object Tools Help
Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*
Query Query History
1 SELECT fclass AS type_of_land,
2   (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS usage_percentage
3 FROM gis_osm_landuse_a_free_1
4 GROUP BY fclass
5 ORDER BY usage_percentage DESC
6 LIMIT 10;
7
8
9
10
11
Data Output Messages Explain × Geometry Viewer × Notifications


|   | type_of_land | usage_percentage   |
|---|--------------|--------------------|
| 1 | grass        | 33.359972668467457 |
| 2 | forest       | 27.839190090306939 |
| 3 | residential  | 8.264656542792503  |
| 4 | scrub        | 5.620874814148203  |
| 5 | park         | 5.1981920605967372 |
| 6 | farmland     | 4.854624969732212  |
| 7 | cemetery     | 2.8869891362474269 |
| 8 | meadow       | 2.807873473543869  |
| 9 | retail       | 2.4357368378641701 |


Total rows: 10 of 10 | Query complete 00:00:00.147
Ln 6, Col 10

```

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;

```

```

pgAdmin 4
File Object Tools Help
Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*
Query Query History
1 CREATE INDEX idx_name ON gis_osm_buildings_a_free_1 (name);
2
3
4
5
6
Data Output Messages Explain × Geometry Viewer × Notifications
CREATE INDEX
Query returned successfully in 2 secs 806 msec.
Ln 2, Col 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;

```

Total rows: 5 of 5 Query complete 00:00:00.572 Ln 11, Col 9

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;

```

```

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);

```

CREATE INDEX
Query returned successfully in 556 msec.
Ln 7, Col 1

```

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;

```

land_type	geometry_type	total_area
nature_reserve	MULTIPOLYGON	17959630429.71581
park	MULTIPOLYGON	937070513.2270126
military	MULTIPOLYGON	52654776.686818
residential	MULTIPOLYGON	313061979.4060141
forest	MULTIPOLYGON	256689456.59400144
industrial	MULTIPOLYGON	127461734.78797278
cemetery	MULTIPOLYGON	122569191.04988158
recreation_ground	MULTIPOLYGON	61087733.971330084
quarry	MULTIPOLYGON	43727631.45506329

Total rows: 10 of 10 Query complete 00:00:00.544 Ln 14, Col 8

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;

```

```

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

```

Query returned successfully in 472 msec.

✓ Query returned successfully in 472 msec. X

Total rows: 10 of 10 Query complete 00:00:00.472 Ln 1, Col 76

```

1 SELECT fclass AS type_of_land,
2     (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM gis_osm_landuse_a_free_1)) AS usage_percentage
3 FROM gis_osm_landuse_a_free_1
4 GROUP BY fclass
5 ORDER BY usage_percentage DESC
6 LIMIT 10;

```

	type_of_land	usage_percentage
1	grass	33.3599726684684757
2	forest	27.839190090306939
3	residential	8.264656542792503
4	scrub	5.6208748141148203
5	park	5.1981920605967372
6	farmland	4.854624969732212
7	cemetery	2.8869891362474269
8	meadow	2.8078734735438689
9	retail	2.4357368378641701

Total rows: 10 of 10 Query complete 00:00:00.136

✓ Successfully run. Total query runtime: 136 msec. 10 rows affected.

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*

Query Query History

```
1 CREATE INDEX idx_fclass2 ON gis_osm_places_free_1 (fclass);
2 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.

Total rows: 9 of 9 Query complete 00:00:00.050

✓ Query returned successfully in 50 msec. X

Ln 5, Col 1

10°C Mostly clear

Search

16:32 ENG IN 07-12-2023

This screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with the 'Tables' node expanded, showing 'gis_osm_places_free_1'. The main pane contains a query editor with two CREATE INDEX statements. The first statement creates an index on the 'fclass' column, and the second creates an index on the 'population' column. Below the query editor, the Data Output tab shows the successful execution of the queries. The status bar at the bottom right indicates the query completed in 50 msec.

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes prosample/postgres@PostgreSQL 16*

Query Query History

```
1 SELECT
2     fclass AS city_type,
3     SUM(population) AS total_population
4 FROM
5     gis_osm_places_free_1
6 WHERE
7     fclass IS NOT NULL AND population IS NOT NULL
8     GROUP BY
9         city_type
10    ORDER BY
11        total_population DESC;
```

Data Output Messages Explain × Geometry Viewer × Notifications

city_type	total_population
city	10665984
suburb	8811435
town	4861549
village	2056472
county	1395774
hamlet	591551
locality	46505
farm	0
island	0

Total rows: 9 of 9 Query complete 00:00:00.107

Ln 11, Col 27

10°C Mostly clear

Search

16:32 ENG IN 07-12-2023

This screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with the 'Tables' node expanded, showing 'gis_osm_places_free_1'. The main pane contains a query editor with a SELECT statement. The query groups the data by 'fclass' (which is mapped to 'city_type') and calculates the sum of 'population' for each group, ordering the results by the total population in descending order. The Data Output tab shows the resulting table, which lists various city types and their corresponding total populations. The status bar at the bottom right indicates the query completed in 00:00:00.107.

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),
```

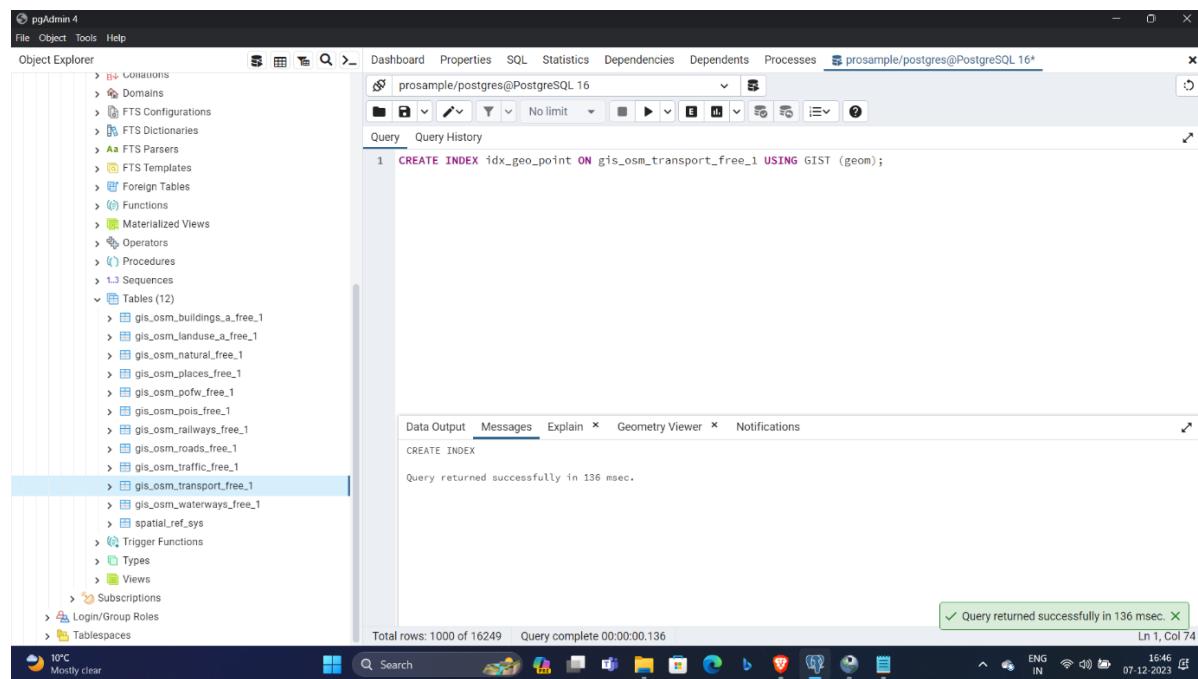
```
ST_SetSRID(ST_GeomFromEWKB(decode('0101000000489D256DCF7952C0FFC8192F825F4440'  
, 'hex')), 4326)  
) / 1609.34 AS distance_A_to_B_in_miles,  
ST_DistanceSphere(
```

```
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*

Query Query History

```

1 SELECT
2   'Centre Avenue' AS loc1,
3   'Woodside' AS loc2,
4   'Long Island City' AS loc3,
5   ST_DistanceSphere(
6     ST_SetSRID(ST_GeomFromEWKB(decode('0101000000795913087C6A52C07C26A02EF7524440', 'hex')), 4326),
7     ST_SetSRID(ST_GeomFromEWKB(decode('0101000000489D0256DCF7952C0FFC8192F825F4440', 'hex')), 4326)
8   ) / 1669.34 AS distance_A_to_B_in_miles,
9   ST_DistanceSphere(
10    ST_SetSRID(ST_GeomFromEWKB(decode('010100000086657D143F7D52C02F1686C8E95E4440', 'hex')), 4326),
11    ST_SetSRID(ST_GeomFromEWKB(decode('010100000086657D143F7D52C02F1686C8E95E4440', 'hex')), 4326)
12  ) / 1669.34 AS distance_B_to_C_in_miles,
13  ST_DistanceSphere(
14    ST_SetSRID(ST_GeomFromEWKB(decode('010100000086657D143F7D52C02F1686C8E95E4440', 'hex')), 4326),
15    ST_SetSRID(ST_GeomFromEWKB(decode('0101000000795913087C6A52C07C26A02EF7524440', 'hex')), 4326)
16  ) / 1669.34 AS distance_C_to_A_in_miles;

```

Data Output Messages Explain × Geometry Viewer × Notifications

loc1	loc2	loc3	distance_a_to_b_in_miles	distance_b_to_c_in_miles	distance_c_to_a_in_miles
text	text	text	double precision	double precision	double precision
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

10°C Mostly clear

Search

System tray icons: File Explorer, Task View, Start, Taskbar, Network, Power, Volume, Battery, Signal strength, Date/Time.