



Find value in open data and geospatial



✉ hi@bnhr.xyz

🌐 bnhr.xyz

🐙 [@benhur07b](https://github.com/benhur07b)

📘 [@bnhr.xyz](https://www.facebook.com/bnhr.xyz)

📧 fosstodon.org/@bnhrdotxyz

📺 [@bnhrdotxyz](https://www.youtube.com/@bnhrdotxyz)

Introduction to PostGIS and Spatial Databases

Table of Contents

Table of Contents.....	2
Credits and License.....	4
What is a spatial database?.....	5
What is PostGIS?.....	5
Exercise 01: PostGIS data management.....	6
Creating a database template with the postgis extension.....	6
Create a database named geo_phl with postgis_template as template.....	7
Create a table named volcanoes inside the geo_phl database.....	9
Adding a geometry column.....	9
About geometries.....	10
Insert rows manually using SQL.....	11
Adding data directly from a shapefile.....	12
Add phl_provinces data to the geo_phl database.....	15
Viewing the Data in QGIS.....	16
The QGIS DB Manager.....	18
Spatial queries.....	19
Exercise 02: PostGIS spatial queries.....	19
Reproject Butuan_Barangay to 32651 projection and create a new table.....	19
What is the total land area of Butuan?.....	20
Geometry vs Geography.....	20
How long is the Butuan bridge?.....	24
How far is the “University:Caraga State University” from “Restaurant:Margie's Restaurant”?.....	25
What Barangay contains the “Bank:Enterprise Bank”?.....	27
Spatial relationships and predicates.....	28
What barangays intersects the bounding box UL:125.49271, 8.95625; LR:125.63853, 8.87170?.....	32
Union all barangays starting with the letter “A”. What percentage of the land area of the municipality are barangays starting with the letter “A”?.....	33
List all barangays that are within 1 km of a “primary road”? Visualize the results.....	34
Tag all the POIs per Barangay.....	38
Snap all the POIs to the Butuan Roads.....	40
Spatial joins.....	42
Nearest neighbor.....	43
Nearest neighbor join.....	43



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [yt @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

Trigger functions and history tracking.....	44
---	----



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
📍 [@bnhr.xyz](#) | 📺 [@bnhrdotxyz](#) | 🐙 [mastodon.social/@benhur07b](#)

Credits and License

Introduction to PostGIS and Spatial Databases

BNHR

version 2023



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

Other works (software, source code, etc.) referenced in this work are under their own respective licenses.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

What is a spatial database?

A spatial database extends the functionalities of a typical database but includes support for:

1. Data types including **Spatial Types**
 1. numbers, date, strings, **geometry, geography, raster**
2. Indexes including **Spatial Indexes**
 1. B-tree, Hash, **R-tree, Quadtree**
3. Functions including **Spatial Functions**
 1. SELECT, SUM, GROUP BY, **ST_Area, ST_Within**

What is PostGIS?

Other resources

- PostGIS workshop: <https://www.postgis.net/workshops/postgis-intro/>
- What people call “PostGIS” is actually just a combination PostgreSQL and the PostGIS spatial “extension”.
- Inherits all the enterprise features of PostgreSQL.
- Provides the same feature set as Oracle or Microsoft SQL Server.
- Runs on platforms large and small. Windows, MacOS, Linux. Intel, ARM, Sparc, Power, MIPS. Cloud VMs and IBM mainframes.
- Supported by a large, multi-vendor community. Not beholden to just one company, not going to be closed, not subject to whims of venture capitalists, and has been around for decades as an open source community project with no sign of stopping.

Note:

For the exercises in this workbook, make sure that you have the postgis extension installed together with your PostgreSQL installation.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

Exercise 01: PostGIS data management

Creating a database template with the postgis extension

USING PSQL ON THE TERMINAL

```
psql -d postgis_template -c CREATE EXTENSION postgis;
```

USING PGADMIN4

1. Create a new database named **postgis_template**. Go to the Definition tab and check Template?
2. **Right click on the postgis_template database > Create > Extension.** This will open the Create – Extension dialog. Under Name, look for **postgis**.

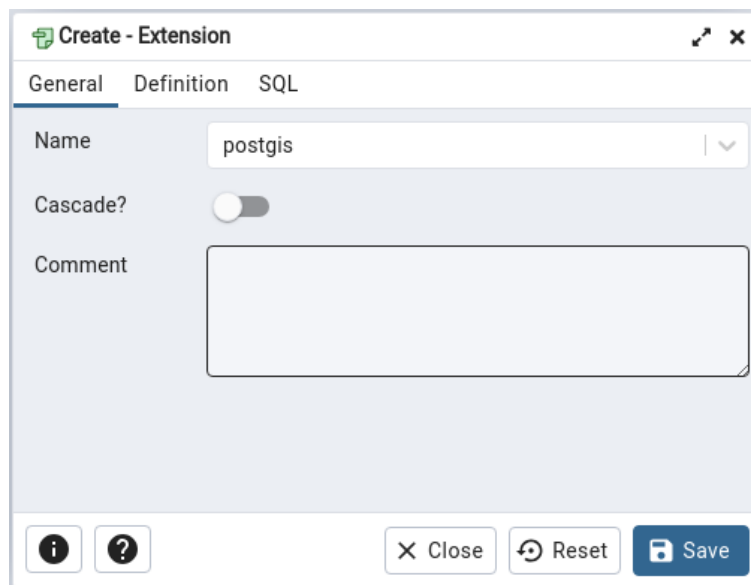


Figure 1: Adding the postgis extension to the postgis_template database

3. Click **Save**.

PRO TIP:

- Next time, if you need to create a database with the postgis extension enabled, you can simply use the template instead of adding the extension as another step.
- This is also useful if you want to add multiple extensions at the time of database creation.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

Create a database named *geo_phl* with *postgis_template* as template

IMPORTANT:

- Disconnect the *postgis_template* database before you use it as a template
- Right click on the database > Disconnect from database

USING PSQL ON THE TERMINAL

```
CREATE DATABASE geo_phl TEMPLATE=postgis_template;
```

USING PGADMIN4

1. Create a database.
2. Name it **geo_phl**.

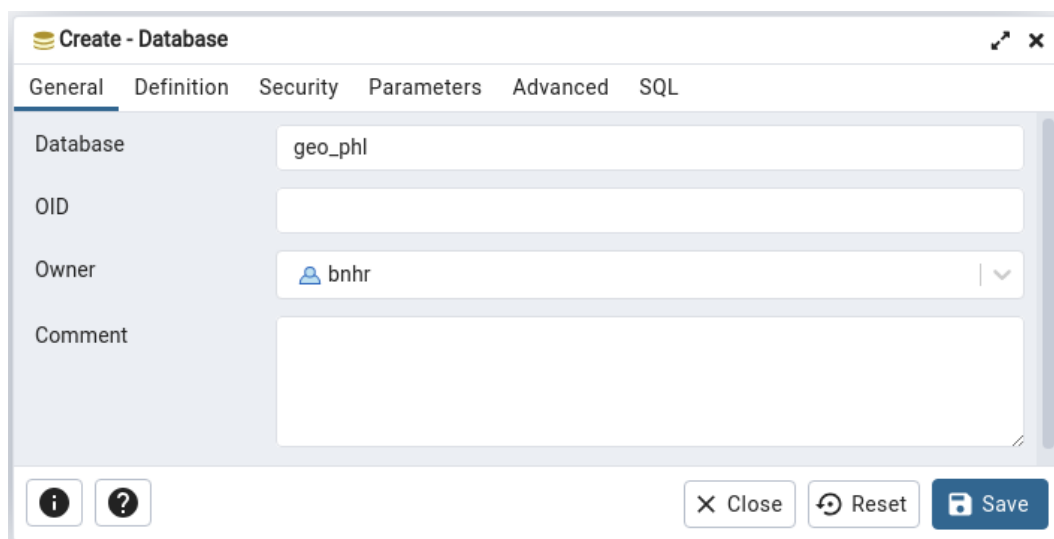


Figure 2: Creating the *geo_phl* table

3. Under Definition, select **postgis_template** for the Template.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

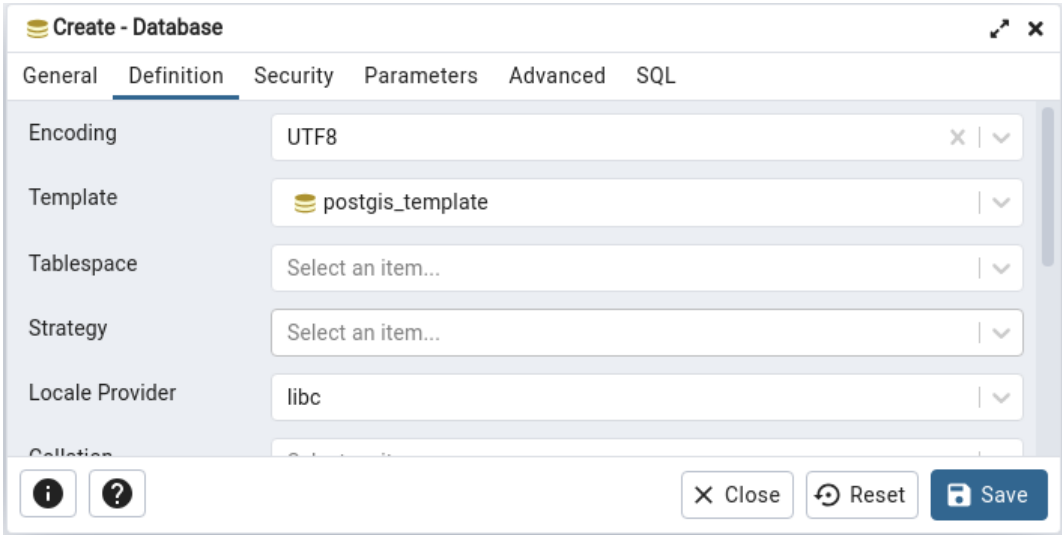


Figure 3: Using the `postgis_template` as a template

View the database using pgAdmin and notice the new items in the db.

One of the items in the db should be **the spatial_ref_sys table**. In this table a listing of most of the projections used is shown.

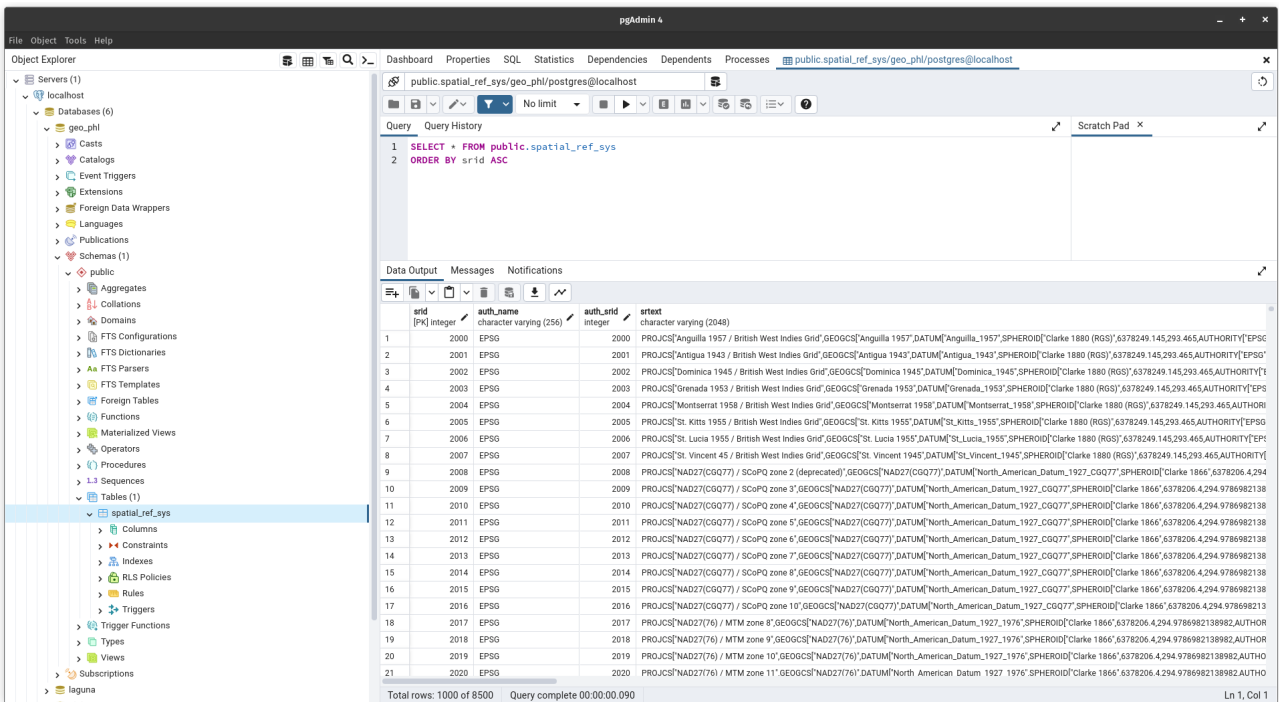


Figure 4: The `spatial_ref_sys` table of the `geo_phl` database



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

 [@bnhr.xyz](https://www.facebook.com/bnhr.xyz) | [@bnhrdotxyz](https://www.youtube.com/@bnhrdotxyz) | mastodon.social/@benhur07b

Create a table named *volcanoes* inside the *geo_phl* database

```
CREATE TABLE volcanoes (
  name character varying,
  elev integer,
  prov character varying,
  erup integer
);
```

Adding a geometry column

Let's add a column to hold the point location of our volcanoes.

```
SELECT AddGeometryColumn('public', 'volcanoes', 'geom', 4326, 'POINT',
2);
```

1. **SELECT:** This is the standard SQL keyword used to retrieve data from a database, but in this case, it's used as part of a specific function call, and it's not meant to return actual data. Instead, it's used to execute a function.
2. **AddGeometryColumn('public', 'volcanoes', 'geom', 4326, 'POINT', 2):** This is a function call. It's calling the **AddGeometryColumn** function with several arguments:
 - **'public':** This is the schema name where the table "volcanoes" is located. Schemas are used to organize database objects.
 - **'volcanoes':** This is the name of the table to which you want to add the new geometry column.
 - **'geom':** This is the name of the new geometry column you want to add to the "volcanoes" table.
 - **4326:** This is the Spatial Reference Identifier (SRID) number, which defines the coordinate system used for the geometry data. In this case, it's using SRID 4326, which corresponds to the WGS 84 coordinate system commonly used for latitude and longitude coordinates.
 - **'POINT':** This specifies the type of geometry you're adding to the column. In this case, it's a "POINT," which is a single set of X and Y coordinates representing a specific location.
 - **2:** This indicates the dimensionality of the geometry. A value of 2 means you're working with 2D (two-dimensional) coordinates, typically for latitude and longitude.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

About geometries

- Geometries are stored on disk in a format only used by PostGIS.
- External programs to insert and retrieve useful geometries, they need to be converted into a format that other applications can understand.
- PostGIS supports emitting and consuming geometries in a large number of formats.
- Geometry Constructors:
 - **WKT (Well-known text)**: Can refer either to the text representation of geometries, with strings starting "POINT", "LINESTRING", "POLYGON", etc. Or can refer to the text representation of a CRS, with strings starting "PROJCS", "GEOGCS", etc. Well-known text representations are OGC standards, but do not have their own specification documents. The first descriptions of WKT (for geometries and for CRS) appeared in the SFSQL 1.0 specification.
 - **WKB (Well-known binary)**: Refers to the binary representation of geometries described in the Simple Features for SQL specification (SFSQL).
 - **GML (Geography Markup Language)**: GML is the OGC standard XML grammar for expressing geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. As with most XML based grammars, there are two parts to the grammar – the schema that describes the document and the instance document that contains the actual data.
 - **JSON (Javascript Object Notation)**: a text format that is very fast to parse in Javascript virtual machines. In spatial, the extended specification for GeoJSON is commonly used.
 - **GeoJSON**: A format for encoding a variety of geographic data structures. A GeoJSON object may represent a geometry, a feature, or a collection of features. GeoJSON supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. Features in GeoJSON contain a geometry object and additional properties, and a feature collection represents a list of features.
 - **KML (Keyhole Markup Language)**: the spatial XML format used by Google Earth. Google Earth was originally written by a company named "Keyhole", hence the (now obscure) reference in the name.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
 📄 [@bnhr.xyz](#) | 📺 [@bnhrdotxyz](#) | 🐙 [mastodon.social/@benhur07b](#)

Insert rows manually using SQL

Look at the volcanoes.csv file. How can we add the values without using COPY?

```
INSERT INTO volcanoes (name, elev, prov, erup, geom)
VALUES ('Banahaw',2158,'QUEZON',0,ST_GeomFromText('POINT(14.07
121.48)',4326));
```

You can also insert multiple rows at the same time.

```
INSERT INTO volcanoes (name, elev, prov, erup, geom)
VALUES ('Bulusan',1565,'SORSOGON',17,ST_GeomFromText('POINT(12.77
124.05)',4326)),
('Cagua',1133,'CAGAYAN',1,ST_GeomFromText('POINT(18.222 122.123)',4326)),
('Camiguin de Babuyan',1133,'CAGAYAN',1,ST_GeomFromText('POINT(18.83
121.86)',4326));
```

1. **INSERT INTO volcanoes:** This part of the statement specifies the target table where you want to insert the data, which is "volcanoes."
2. **(name, elev, prov, erup, geom):** These are the names of the columns in the "volcanoes" table to which you are inserting data.
3. **VALUES:** This is used to specify the values you want to insert into the respective columns.
4. **('Banahaw', 2158, 'QUEZON', 0, ST_GeomFromText('POINT(14.07 121.48)', 4326)):** These are the values you're inserting into the corresponding columns:
 - **'Banahaw':** This is the value for the "name" column, indicating the name of the volcano as "Banahaw."
 - **2158:** This is the value for the "elev" column, representing the elevation of the volcano (2158 meters).
 - **'QUEZON':** This is the value for the "prov" column, specifying the province where the volcano is located as "QUEZON."
 - **0:** This is the value for the "erup" column, indicating information about the eruption (in this case, "0" might represent no recent eruptions).
 - **ST_GeomFromText('POINT(14.07 121.48)', 4326):** This is the value for the "geom" column, representing the location of the volcano as a point geometry. The ST_GeomFromText function constructs a point with coordinates (latitude 14.07, longitude 121.48) using the SRID 4326, which specifies the WGS 84 coordinate system.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

SOMETHING TO THINK ABOUT:

After creating a spatial table, take a peek at the Geometry view. What is this table for? What about the other columns?

Adding data directly from a shapefile

PostGIS comes with a utility known as **shp2pgsql**. There is a command-line tool and a GUI **shp2pgsql-gui**.

In windows, these commands can usually be found at: C:\Program Files\PostgreSQL\<version>\bin.

IMPORTANT:

- When loading data from a shapefile, we need to specify the SRID of the data.

Try to load the Butuan_POIs shapefile into the geo_phl database.

USING THE SHP2PGSQL COMMAND LINE TOOL

```
shp2pgsql -s 4326 "PATH\T0\Butuan_POIS.shp" public.butuan_pois >
"PATH\T0\butuan_pois.sql"
* -s is used to specify the SRID
```

1. **-s 4326**: This flag specifies the Spatial Reference Identifier (SRID) to be used. In this case, it's set to 4326, which corresponds to the WGS 84 coordinate system.
2. **"PATH\T0\Butuan_POIS.shp"**: This is the path to the source Shapefile you want to convert. Make sure to provide the correct file path to your Shapefile.
3. **public.butuan_pois**: This part specifies the target table where the data from the Shapefile will be inserted. In this case, it's using the **public** schema and naming the table **"butuan_pois."**
4. **>**: Output redirection. It directs the output of **shp2pgsql** command to a file.
5. **"PATH\T0\butuan_pois.sql"**: This is the path and file name where the SQL statements generated by **shp2pgsql** will be saved. The SQL file will contain the INSERT statements necessary to populate the **"butuan_pois"** table in your PostgreSQL database.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [yt @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

USING SHP2PGSQL-GUI

1. Open the shp2pgsql-gui application.

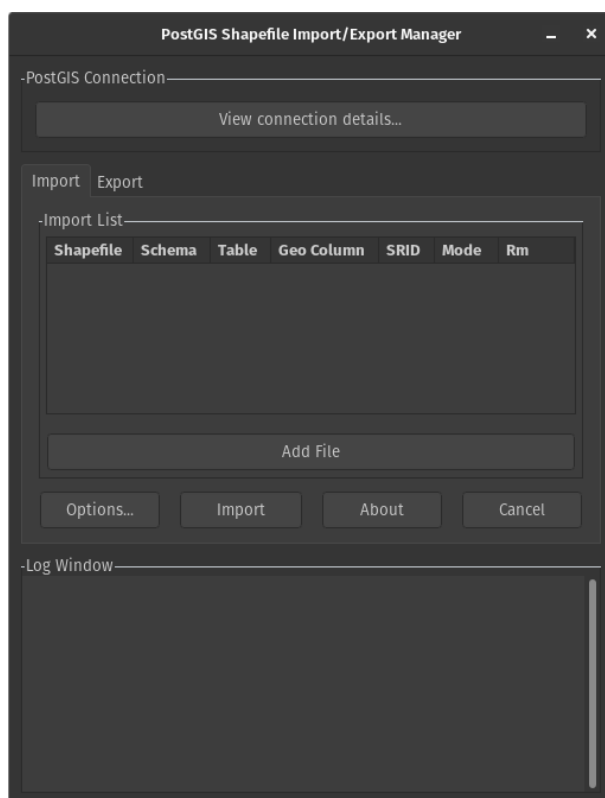


Figure 5: The shp2pgsql-gui interface

2. Connect to your database.

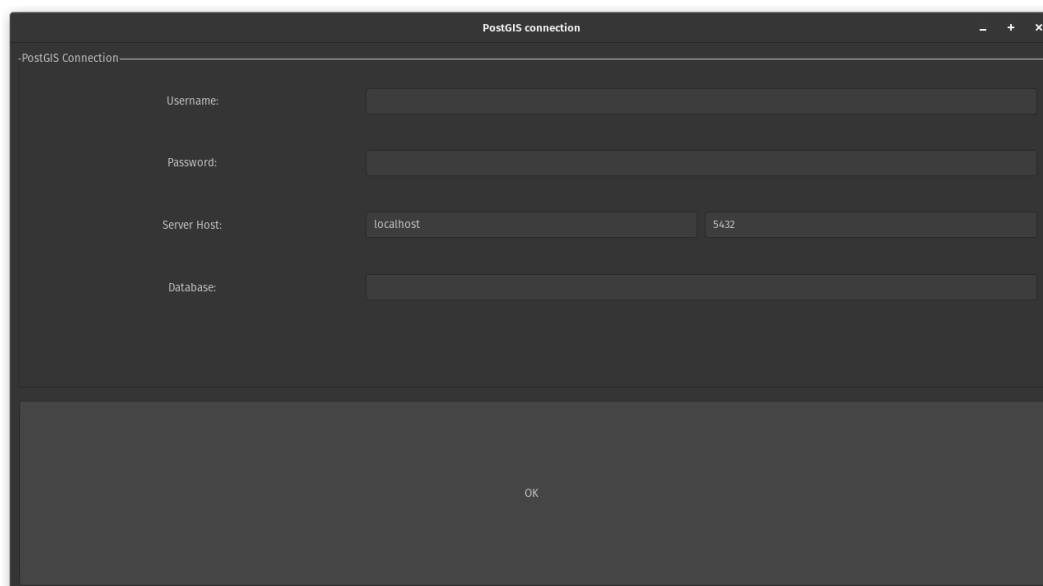
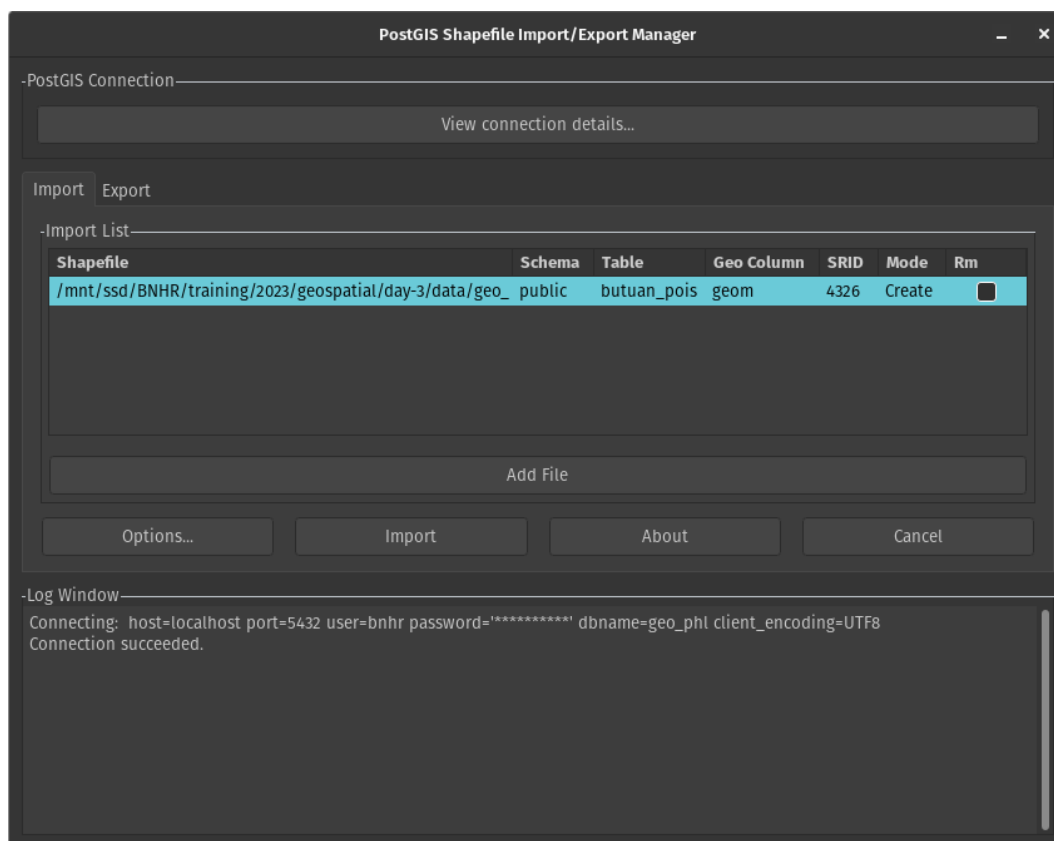


Figure 6: Add database connection details



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [yt @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

3. Add the shapefiles by clicking **Add file**.4. Import the shapefile into the database by clicking **Import**.**TRY FOR YOURSELF:**

Check if the data was actually uploaded to the database.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [yt @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

AUTOMATING USING SCRIPTS

If you're more comfortable with using scripts (such as in BASH), the following script can also be used to load multiple shapefiles in one directory with the same SRID into the database.

```
#!/bin/bash

for f in *.shp
do
    shp2pgsql -I -s 4326 $f `basename $f .shp` > `basename $f .shp`.sql
done

for f in *.sql
do
    psql -d geo_phl -f $f
done
```

Add phl_provinces data to the geo_phl database

1. Create a non-spatial table named phl_provinces_table using the phl_provinces_table.csv
2. Create a spatial table by loading the phl_provinces shapefile using shp2pgsql.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
📄 [@bnhr.xyz](#) | 📺 [@bnhrdotxyz](#) | 🐙 [mastodon.social/@benhur07b](#)

Viewing the Data in QGIS

1. On the QGIS Browser Panel, **right click on PostGIS > New connection.**

Figure 7: Connecting QGIS to a PostGIS database

Figure 8: Add a user/role that has access to the database



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

2. You should now have a GEO_PHL (or whatever name you used) connection under PostgreSQL in the QGIS Browser.

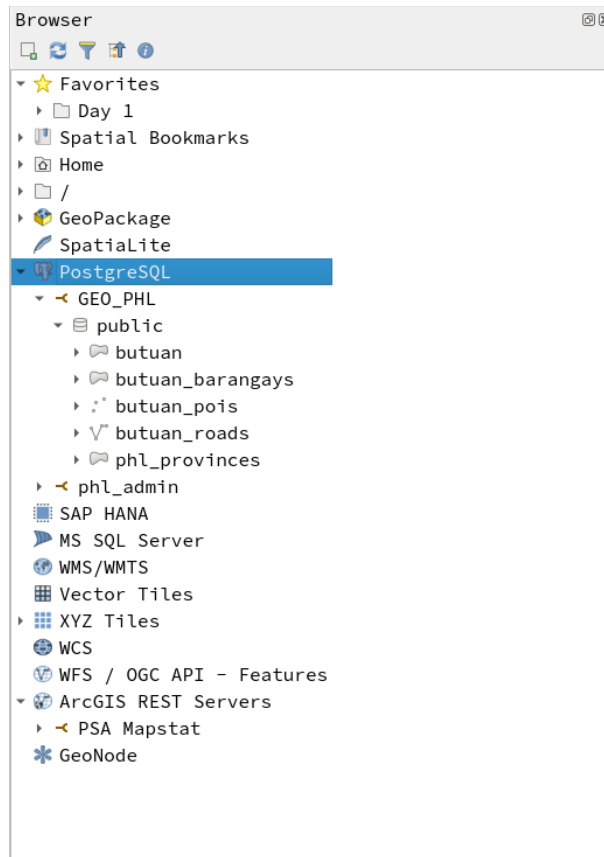


Figure 9: GEO_PHL PostgreSQL database connected



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

3. Load some of the layers.

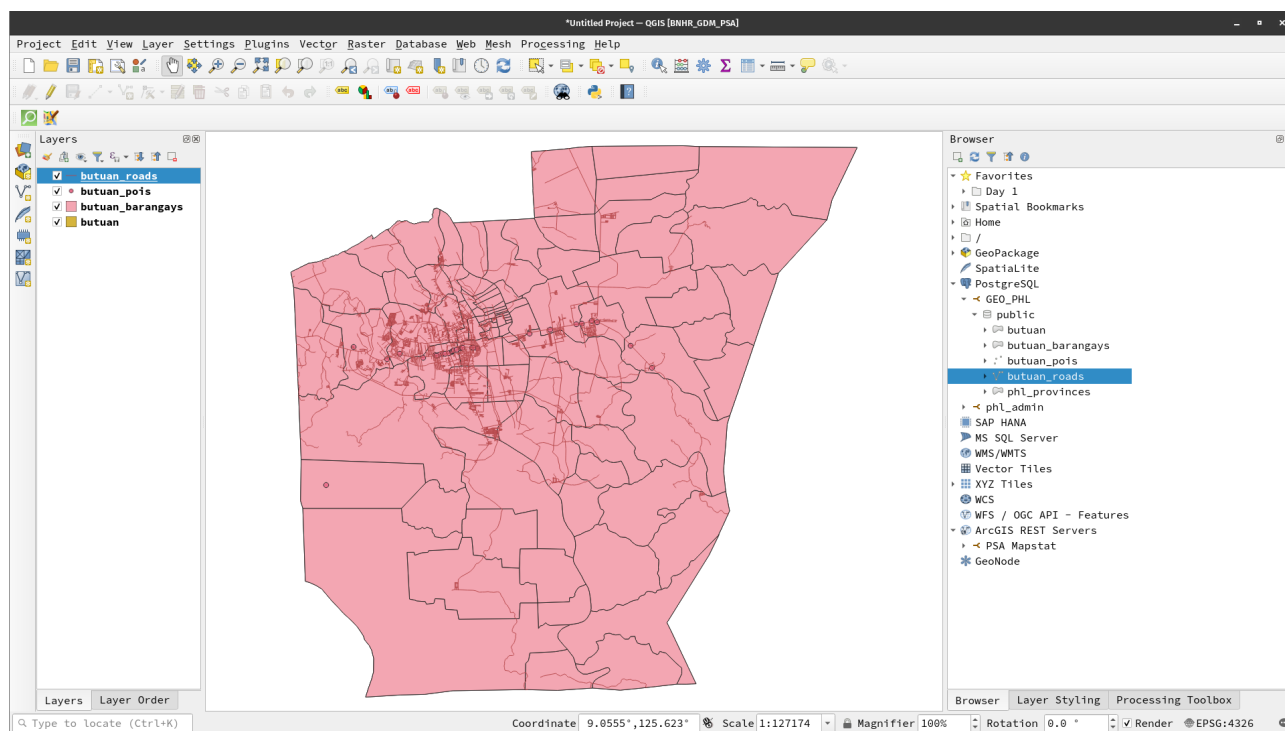


Figure 10: Butuan data loaded from a database

The QGIS DB Manager

You can use QGIS to do queries directly to connected databases via the **DB Manager**. To open the DB Manager, go to **Database > DB Manager**

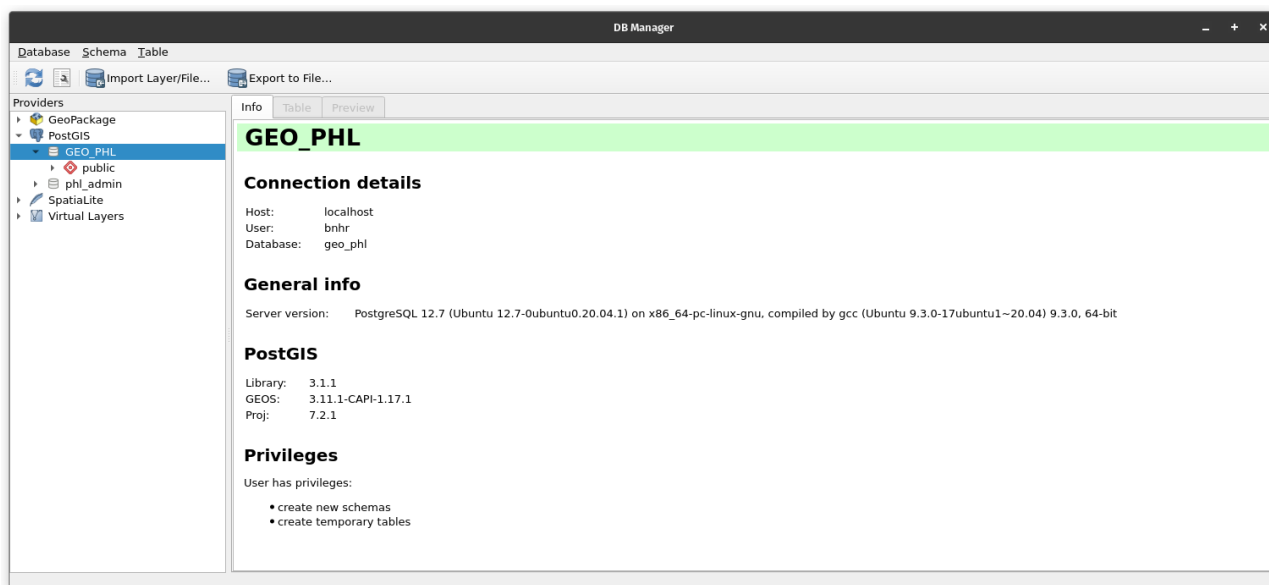


Figure 11: The QGIS DB Manager



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[@bnhr.xyz](#) | [@bnhrdotxyz](#) | [mastodon.social/@benhur07b](#)

Spatial queries

Additional resource:

- PostGIS reference: <https://postgis.net/docs/reference.html>

Note:

For our next exercises, we will use the QGIS DB Manager to directly interact with our database.

Exercise 02: PostGIS spatial queries

Reproject Butuan_Barangay to 32651 projection and create a new table.

```
CREATE TABLE butuan_barangays_32651 AS
  SELECT * FROM butuan_barangays;

ALTER TABLE butuan_barangays_32651
  ALTER COLUMN geom TYPE geometry(MultiPolygon, 32651)
  USING ST_Transform(geom, 32651);
```

- The **ST_Transform** function is used to transform a geometry from one spatial reference system (SRID) to another. In this specific context, it is being used to change the SRID of a geometry stored in the "geom" column of the table.
- **geom**: This is the name of the column containing the geometry that you want to transform.
- **32651**: This is the target SRID to which you want to transform the geometry. In this case, the SRID 32651 represents a specific spatial reference system, often used for specific cartographic applications.

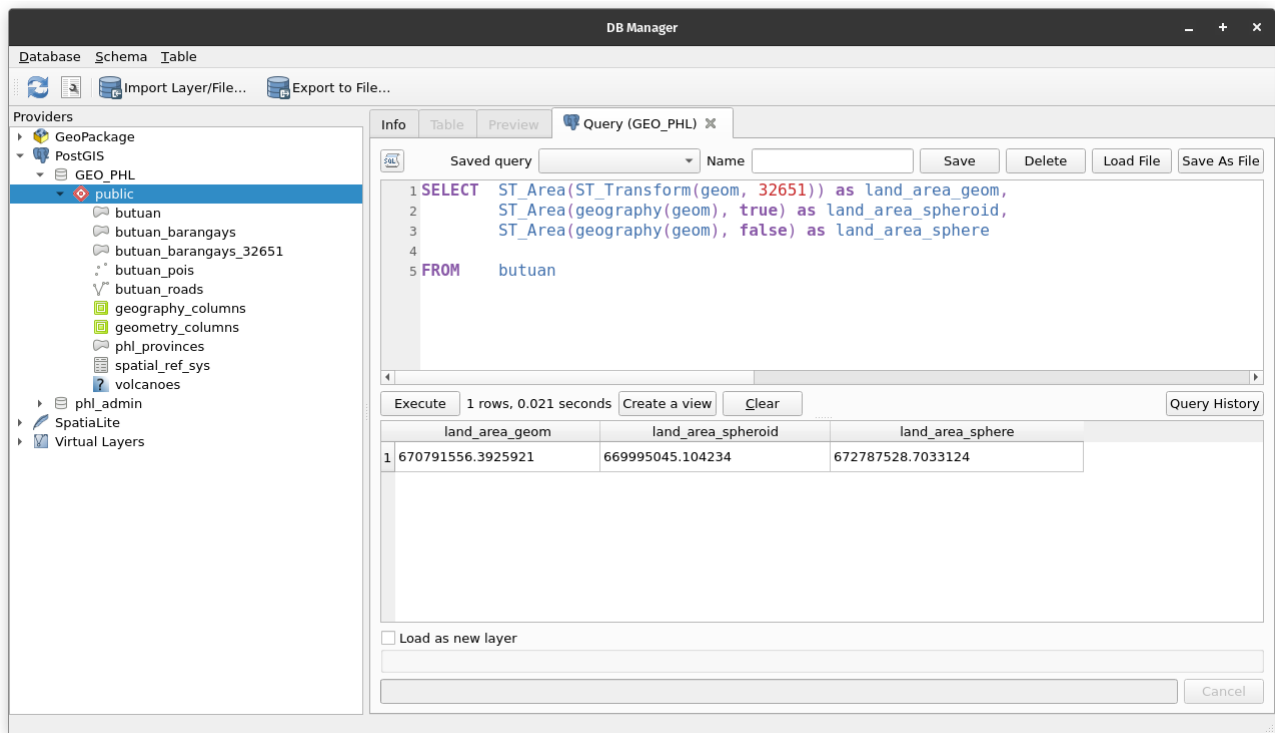


This work and its contents by Ben Hur Pintor is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
 @bnhr.xyz | @bnhrdotxyz | mastodon.social/@benhur07b

What is the total land area of Butuan?

```
SELECT      ST_Area(ST_Transform(geom, 32651)) as land_area_geom,
            ST_Area(geography(geom), true) as land_area_spheroid,
            ST_Area(geography(geom), false) as land_area_sphere

FROM butuan
```



Geometry vs Geography

- **geometry** (geom type) uses **planar coordinates** and is usually much faster while **geography** uses **ellipsoidal computations**.
- **geometry** is used for **most applications**. **geography** is applicable for **large areas** where the curvature of the earth will play a role.
- **degrees** are not units of distance or area.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

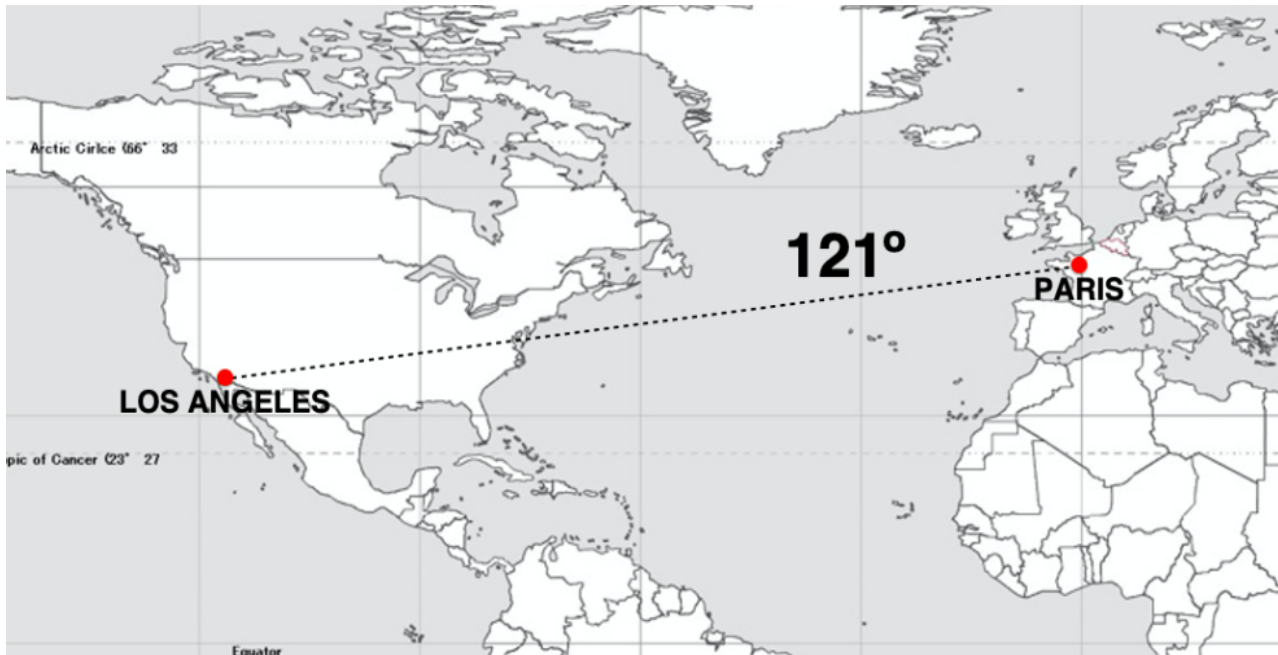


Figure 12: Geometry distance



Figure 13: Geography distance



This work and its contents by Ben Hur Pintor is licensed under a
 Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
 @bnhr.xyz | @bnhrdotxyz | mastodon.social/@benhur07b



Figure 14: Checking intersection with geometry



Figure 15: Checking intersection with the same layers using geography



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

- Geography native functions
 - ST_Distance(G1, G2)
 - ST_DWithin(G1, G2, R)
 - ST_Area(geog)
 - ST_Length(geography)
 - ST_Covers(G1, G2)
 - ST_CoveredBy(G1, G2)
 - ST_Intersects(G1, G2)
 - ST_AsText(G1)
 - ST_AsBinary(G1)
 - ST_AsSVG(G1)
 - ST_AsGML(G1)
 - ST_AsKML(G1)
 - ST_AsGeoJson(G1)
 - ST_Buffer(G1, R)
 - ST_Intersection(G1, G2)
- **Why don't we just always use geography?**
 - Not all functions in geography have native on-the-sphere implementations yet.
 - The computational cost of geography compared to geometry is quite high.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
📱 [@bnhr.xyz](#) | 📺 [@bnhrdotxyz](#) | 🐙 [mastodon.social/@benhur07b](#)

How long is the Butuan bridge?

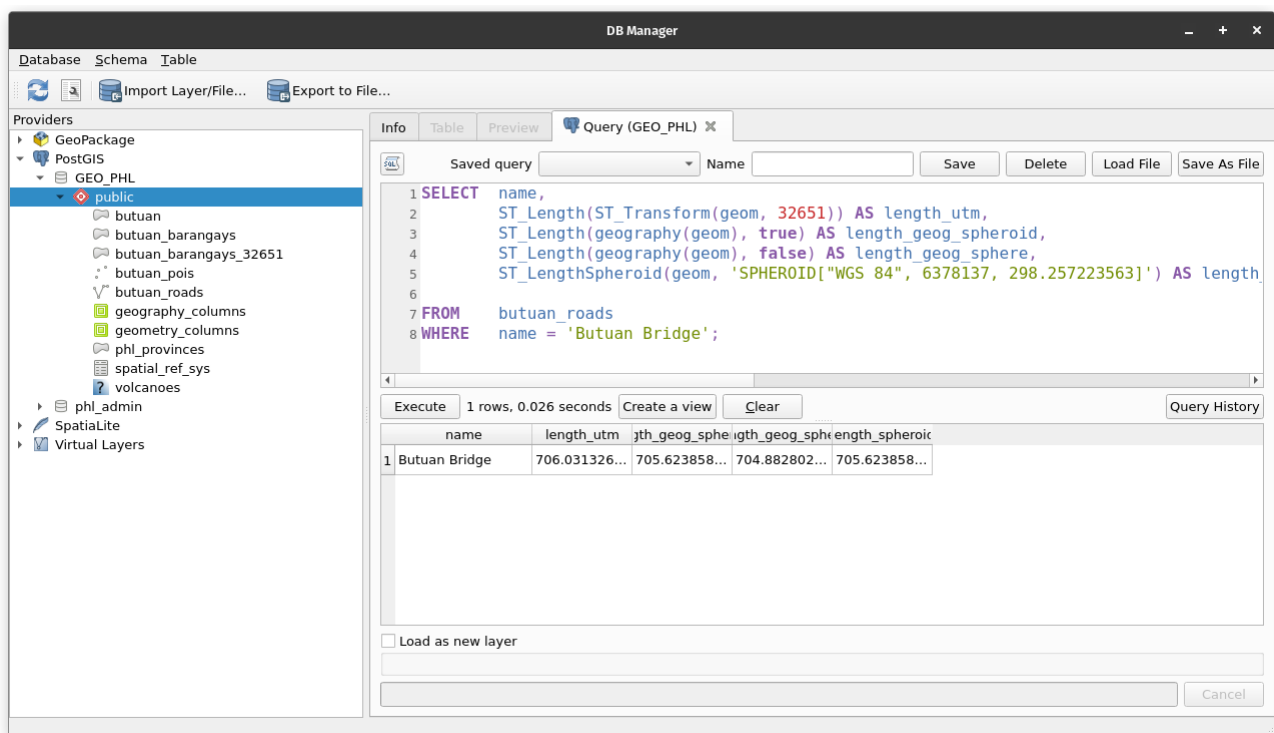
```

SELECT      name,
            ST_Length(ST_Transform(geom, 32651)) AS length_utm,
            ST_Length(geography(geom), true) AS length_geog_spheroid,
            ST_Length(geography(geom), false) AS length_geog_sphere,
            ST_LengthSpheroid(geom, 'SPHEROID["WGS 84", 6378137,
298.257223563]') AS length_spheroid

FROM        butuan_roads

WHERE       name = 'Butuan Bridge';

```



- **ST_Length(ST_Transform(geom, 32651)) AS length_utm:** This calculates the length of the road after transforming the geometry to SRID 32651 and aliases the result as "length_utm."
- **ST_Length(geography(geom), true) AS length_geog_spheroid:** This calculates the length of the road using the geography type, considering the spheroid shape of the Earth, and aliases the result as "length_geog_spheroid."



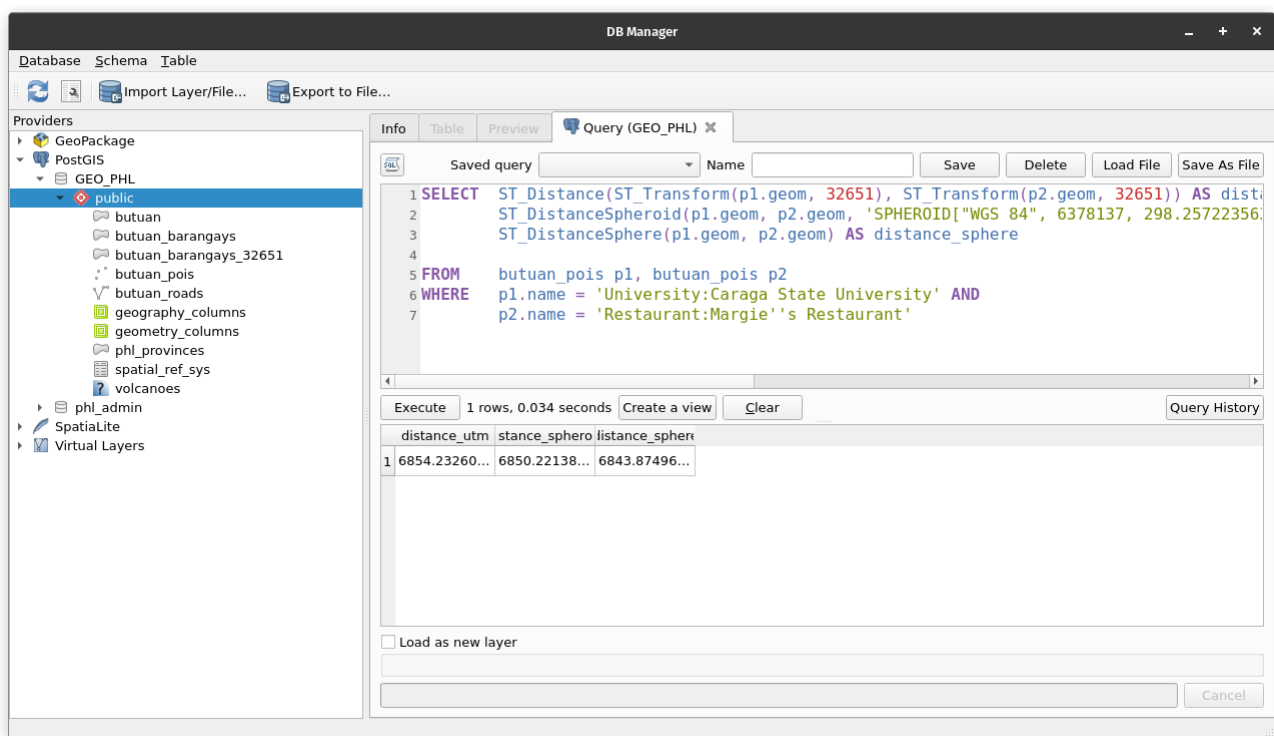
This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[@bnhr.xyz](#) | [@bnhrdotxyz](#) | [mastodon.social/@benhur07b](#)

- **ST_Length(geography(geom), false) AS length_geog_sphere:** This calculates the length of the road using the geography type, assuming a spherical Earth, and aliases the result as "length_geog_sphere."
- **ST_LengthSpheroid(geom, 'SPHEROID["WGS 84", 6378137, 298.257223563]') AS length_spheroid:** This calculates the length of the road based on a specific spheroid definition ('WGS 84') and aliases the result as "length_spheroid."

How far is the "University:Caraga State University" from "Restaurant:Margie's Restaurant"?

```
SELECT      ST_Distance(ST_Transform(p1.geom, 32651),
ST_Transform(p2.geom, 32651)) AS distance_utm,
           ST_DistanceSpheroid(p1.geom, p2.geom, 'SPHEROID["WGS 84",
6378137, 298.257223563]') AS distance_spheroid,
           ST_DistanceSphere(p1.geom, p2.geom) AS distance_sphere

FROM        butuan_pois p1, butuan_pois p2
WHERE       p1.name = 'University:Caraga State University' AND
           p2.name = 'Restaurant:Margie''s Restaurant'
```



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[@bnhr.xyz](#) | [@bnhrdotxyz](#) | [mastodon.social/@benhur07b](#)

Functions:

ST_Distance — Returns the distance between two geometry or geography values

ST_DistanceSpheroid — Returns the minimum distance between two lon/lat geometries using a spheroidal earth model.

ST_DistanceSphere — Returns minimum distance in meters between two lon/lat geometries using a spherical earth model.

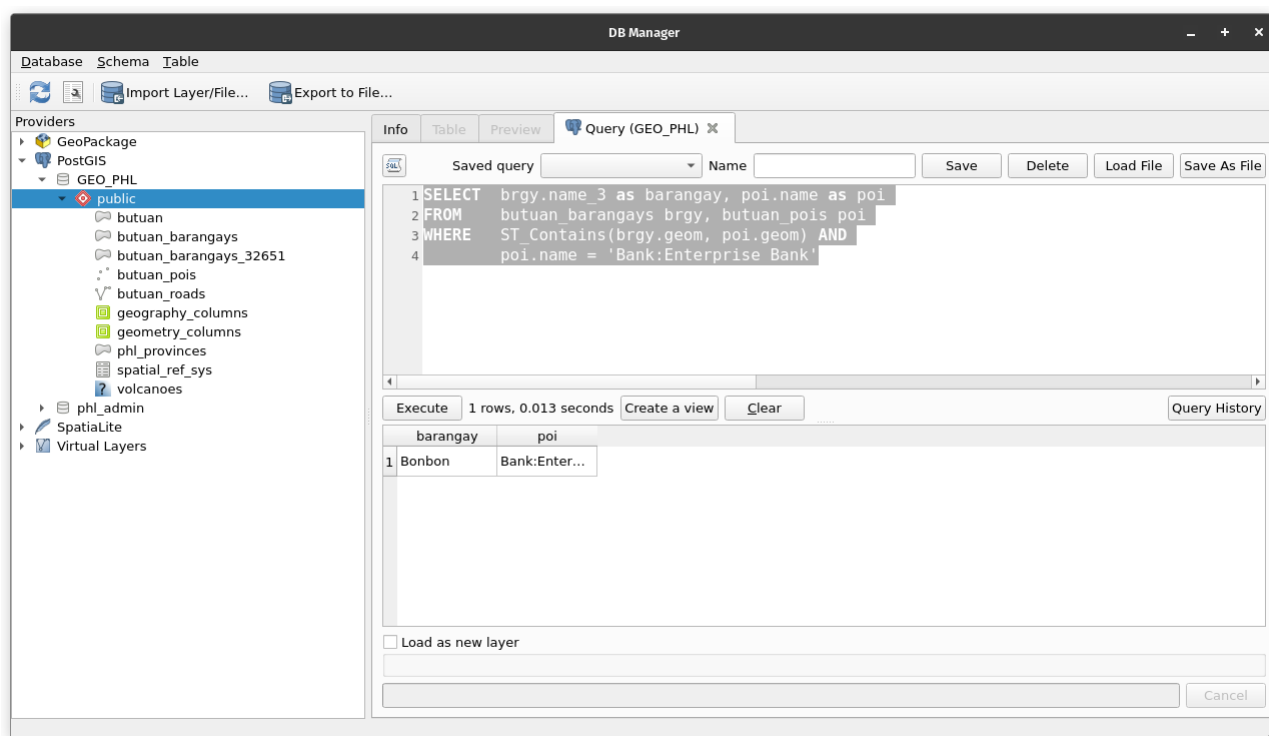
- **ST_Distance(ST_Transform(p1.geom, 32651), ST_Transform(p2.geom, 32651))**
AS distance_utm:
 - ST_Transform(p1.geom, 32651) and ST_Transform(p2.geom, 32651): These functions transform the geometries represented by p1.geom and p2.geom to a common spatial reference system (SRID 32651) before calculating the distance. This transformation ensures that both geometries are in the same coordinate system for accurate distance calculation.
 - AS distance_utm: This aliases the result as "distance_utm."
 - ST_DistanceSpheroid(p1.geom, p2.geom, 'SPHEROID["WGS 84", 6378137, 298.257223563]') AS distance_spheroid:
- **ST_DistanceSpheroid:**
 - p1.geom and p2.geom: These are the geometries representing the two points for which you want to calculate the distance.
 - 'SPHEROID["WGS 84", 6378137, 298.257223563]': This is a specific spheroid definition for the WGS 84 ellipsoid. It defines the shape of the Earth for the distance calculation.
 - AS distance_spheroid: This aliases the result as "distance_spheroid."
 - ST_DistanceSphere(p1.geom, p2.geom) AS distance_sphere:
- **ST_DistanceSphere:**
 - p1.geom and p2.geom: These are the geometries representing the two points you want to calculate the distance between.
 - AS distance_sphere: This aliases the result as "distance_sphere."



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [yt @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

What Barangay contains the "Bank:Enterprise Bank"?

```
SELECT brgy.name_3 as barangay, poi.name as poi
FROM butuan_barangays brgy, butuan_pois poi
WHERE ST_Contains(brgy.geom, poi.geom) AND
poi.name = 'Bank:Enterprise Bank'
```



Functions:

ST_Contains — Tests if every point of B lies in A, and their interiors have a point in common

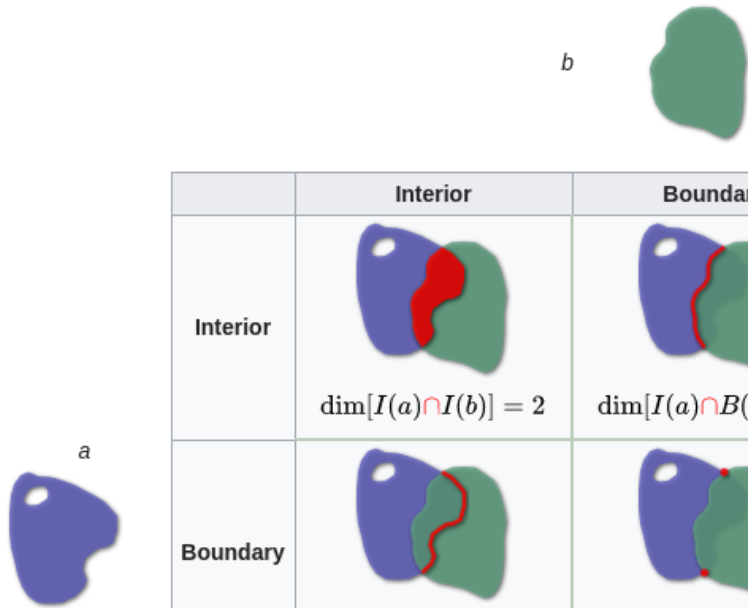


This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

Spatial relationships and predicates

Spatial relationships and predicates are fundamental concepts in GIS. They describe the relationships between geometries in space, such as whether one geometry contains, intersects, or touches another geometry.

These spatial relationships or spatial predicates are based on Dimensionally Extended Nine-Intersection Model (DE-9IM) which is a standard used to describe the spatial relations of two regions or geometries. It is a topological model that describes the spatial relationships between geometries using a matrix of nine values. The matrix describes the intersection of the interior, boundary, and exterior of two geometries in terms of their dimensionality.



	Interior	Boundary	Exterior
Interior	 $\dim[I(a) \cap I(b)] = 2$	 $\dim[I(a) \cap B(b)] = 1$	 $\dim[I(a) \cap E(b)] = 2$
Boundary	 $\dim[B(a) \cap I(b)] = 1$	 $\dim[B(a) \cap B(b)] = 0$	 $\dim[B(a) \cap E(b)] = 1$
Exterior	 $\dim[E(a) \cap I(b)] = 2$	 $\dim[E(a) \cap B(b)] = 1$	 $\dim[E(a) \cap E(b)] = 2$

Figure 16: The DE-9IM matrix of two overlapping geometries
(<https://en.wikipedia.org/wiki/DE-9IM>)



This work and its contents by Ben Hur Pintor is licensed under a
Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
[f @bnhr.xyz](https://www.facebook.com/bnhr.xyz) | [y @bnhrdotxyz](https://www.youtube.com/channel/UCbnhrdotxyz) | [m mastodon.social/@benhur07b](https://mastodon.social/@benhur07b)

In words:

Two geometries are topologically **EQUAL** if their boundaries intersect and no part of the interior or boundary of one geometry intersects the exterior of the other.

They are **DISJOINT** if they have no point in common.

They **TOUCH** if they have at least one point in common, but their interiors do not intersect.

A **CONTAINS** B iff no points of B lie in the exterior of A, and at least one point of the interior of B lies in the interior of A.

They **INTERSECT** if they have at least one point in common.

The **spatial predicates** that you will encounter the most include:

- Disjoint
- Intersects
- Equals
- Touches
- Crosses
- Overlaps
- Contains
- Within



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

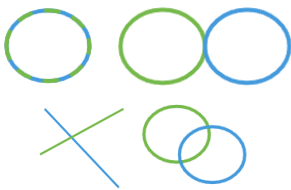
You can better understand what these spatial predicates mean by imagining two geometries (one **G in green** and another **B in blue**) and seeing their interaction with one another.

Disjoint



- The Boundaries and Interiors do not intersect
- Applicable to all geometry combinations
- G is disjoint from B and vice versa

Intersects



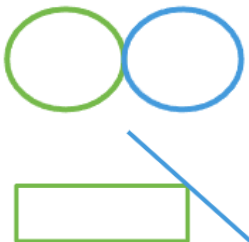
- Not DISJOINT
- Applicable to all geometry combinations

Equals



- Both geometries have the same boundary and the same interior
- Applicable only to combinations of the same kind of geometry (e.g. Point/Point, Line/Line, Polygon/Polygon)
- $G = B$ and vice versa

Touches



- Boundaries may intersect or one boundary may intersect the other interior.
- The interiors do not touch.
- G touches B and vice versa
- Point/Point – undefined

Crosses



- Interiors intersect and the base's interior intersects the candidate's exterior.
- Line/Line - the intersection of the interiors forms a point.
- G crosses B and vice versa
- Point/Point, Area/Area – undefined



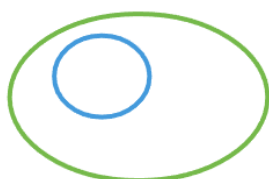
This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

Overlaps



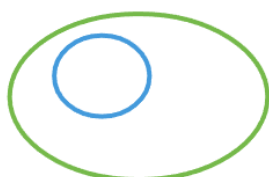
- Interiors intersect but neither feature is contained by the other nor are the features equal.
- G overlaps B and vice versa
- Point/Line, Point/Area, Line/Area - undefined

Contains



- Interiors intersect and no part of the candidate's interior or boundary intersects the base's exterior.
- Possible for the boundaries to intersect.
- G contains B
- Inverse of WITHIN.

Within



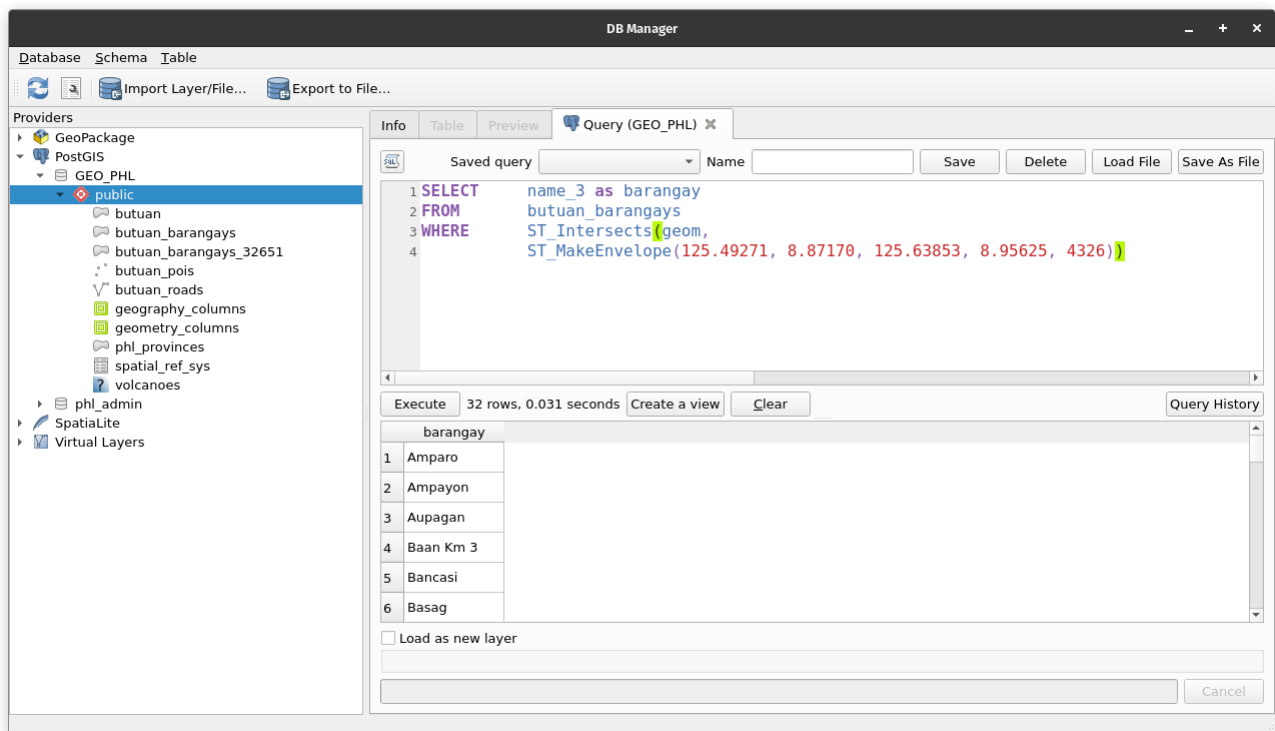
- Interiors intersect and no part of the base's interior or boundary intersects the candidate's exterior.
- Possible for the boundaries to intersect.
- B is with G
- Inverse of CONTAINS.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

What barangays intersects the bounding box UL:125.49271, 8.95625; LR:125.63853, 8.87170?

```
SELECT    name_3 as barangay
FROM      butuan_barangays
WHERE     ST_Intersects(geom,
                        ST_MakeEnvelope(125.49271, 8.87170, 125.63853, 8.95625, 4326))
```



Functions:

ST_Intersects — Tests if two geometries intersect (they have at least one point in common)

ST_MakeEnvelope — Creates a rectangular Polygon from minimum and maximum coordinates.



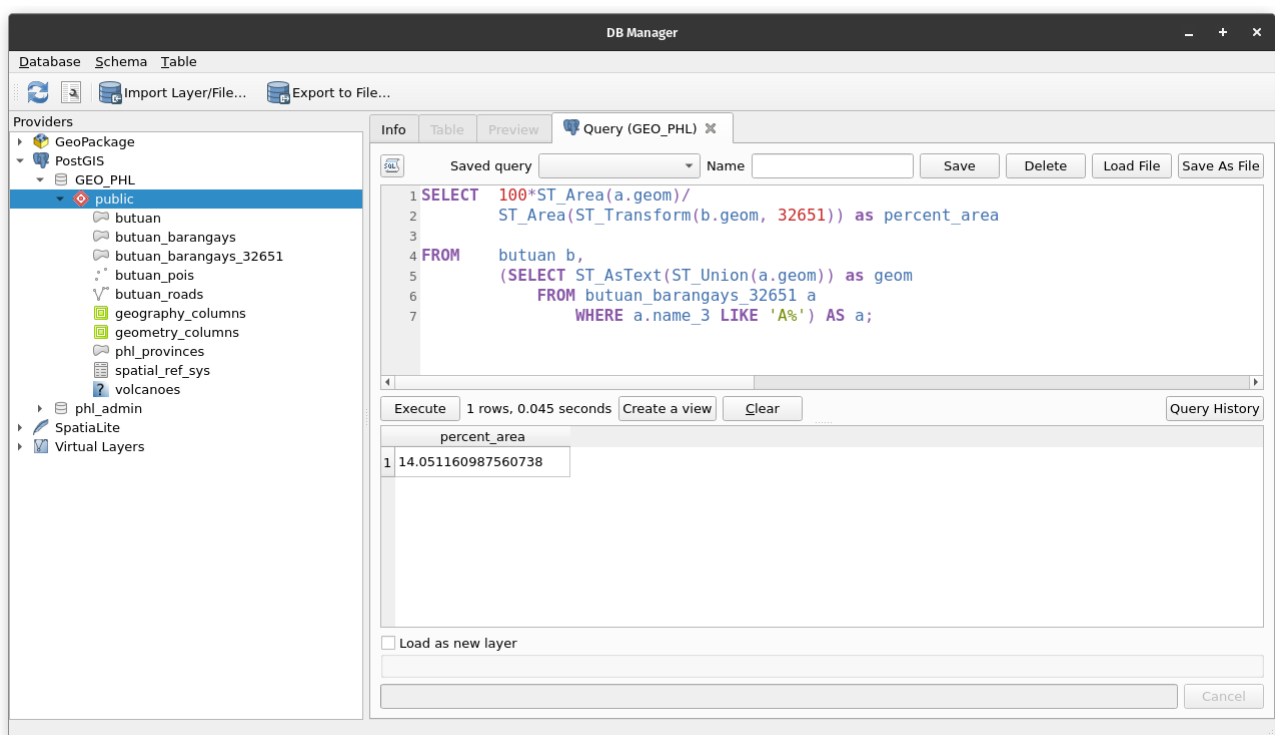
This work and its contents by Ben Hur Pintor is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

[f @bnhr.xyz](#) |
 [y @bnhrdotxyz](#) |
 [m mastodon.social/@benhur07b](#)

Union all barangays starting with the letter “A”. What percentage of the land area of the municipality are barangays starting with the letter “A”?

```
SELECT      100*ST_Area(a.geom)/
            ST_Area(ST_Transform(b.geom, 32651)) as percent_area

FROM        butuan b,
            (SELECT ST_AsText(ST_Union(a.geom)) as geom
             FROM butuan_barangays_32651 a
             WHERE a.name_3 LIKE 'A%') AS a;
```



Functions:

ST_Area — Returns the area of a polygonal geometry.

ST_Transform — Return a new geometry with coordinates transformed to a different spatial reference system.

ST_AsText — Return the Well-Known Text (WKT) representation of the geometry/geography without SRID metadata.

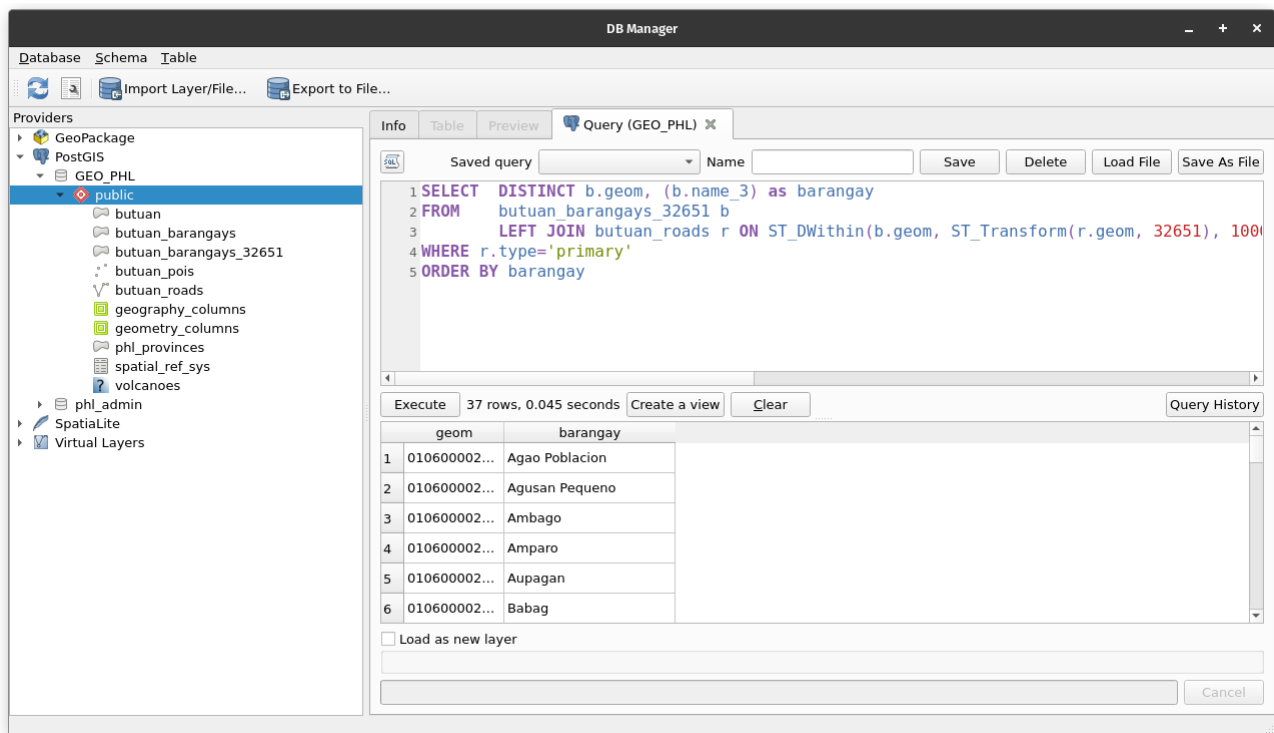
ST_Union — Computes a geometry representing the point-set union of the input geometries.



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

List all barangays that are within 1 km of a “primary road”? Visualize the results.

```
SELECT DISTINCT b.geom, (b.name_3) as barangay
FROM butuan_barangays_32651 b
LEFT JOIN butuan_roads r ON ST_DWithin(b.geom,
ST_Transform(r.geom, 32651), 1000)
WHERE r.type='primary'
ORDER BY barangay
```



Functions

ST_DWithin — Tests if two geometries are within a given distance

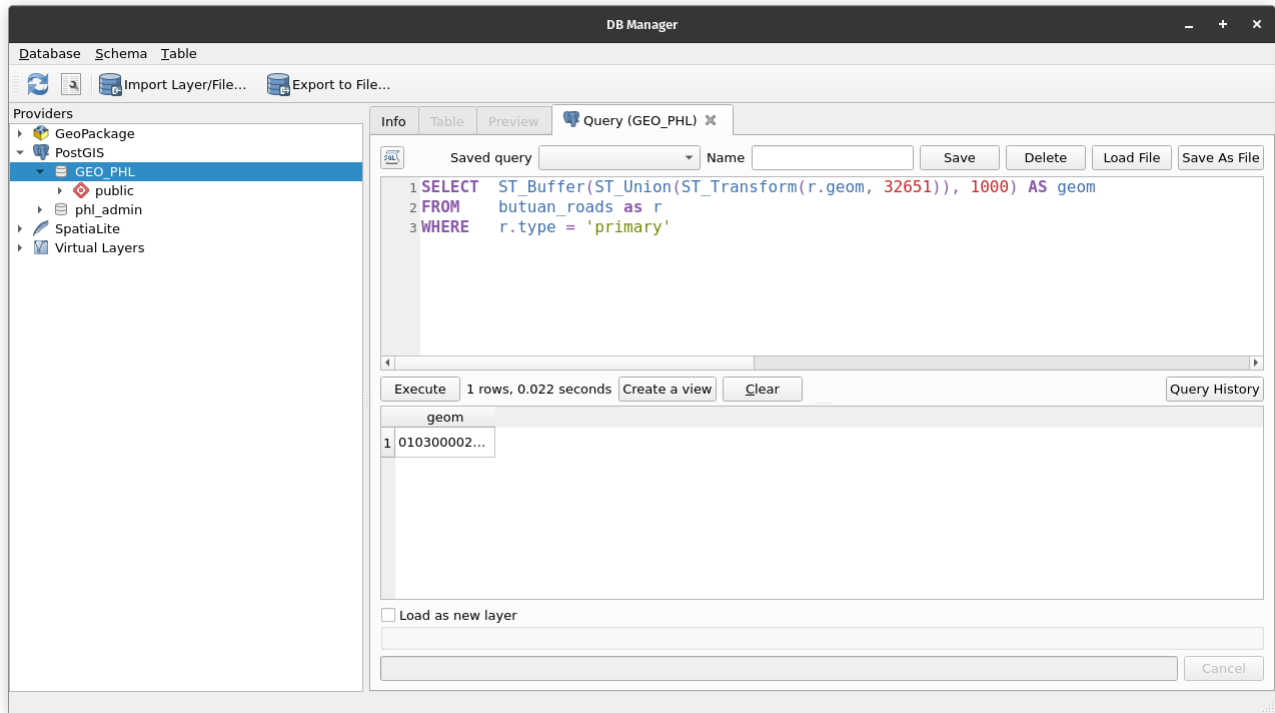
ST_Transform — Return a new geometry with coordinates transformed to a different spatial reference system.



This work and its contents by Ben Hur Pintor is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[@bnhr.xyz](#) | [@bnhrdotxyz](#) | [mastodon.social/@benhur07b](#)

BUFFER

```
SELECT ST_Buffer(ST_Union(ST_Transform(r.geom, 32651)), 1000) AS geom
FROM butuan_roads as r
WHERE r.type = 'primary'
```



Functions

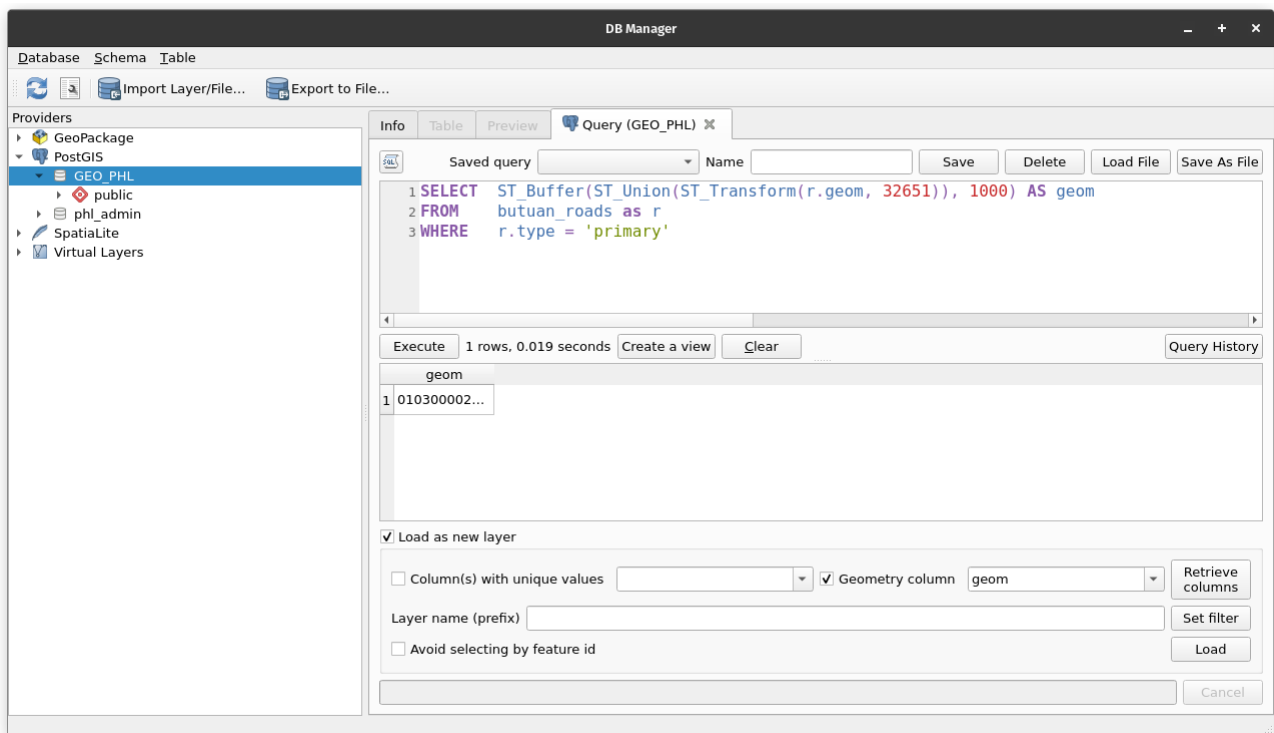
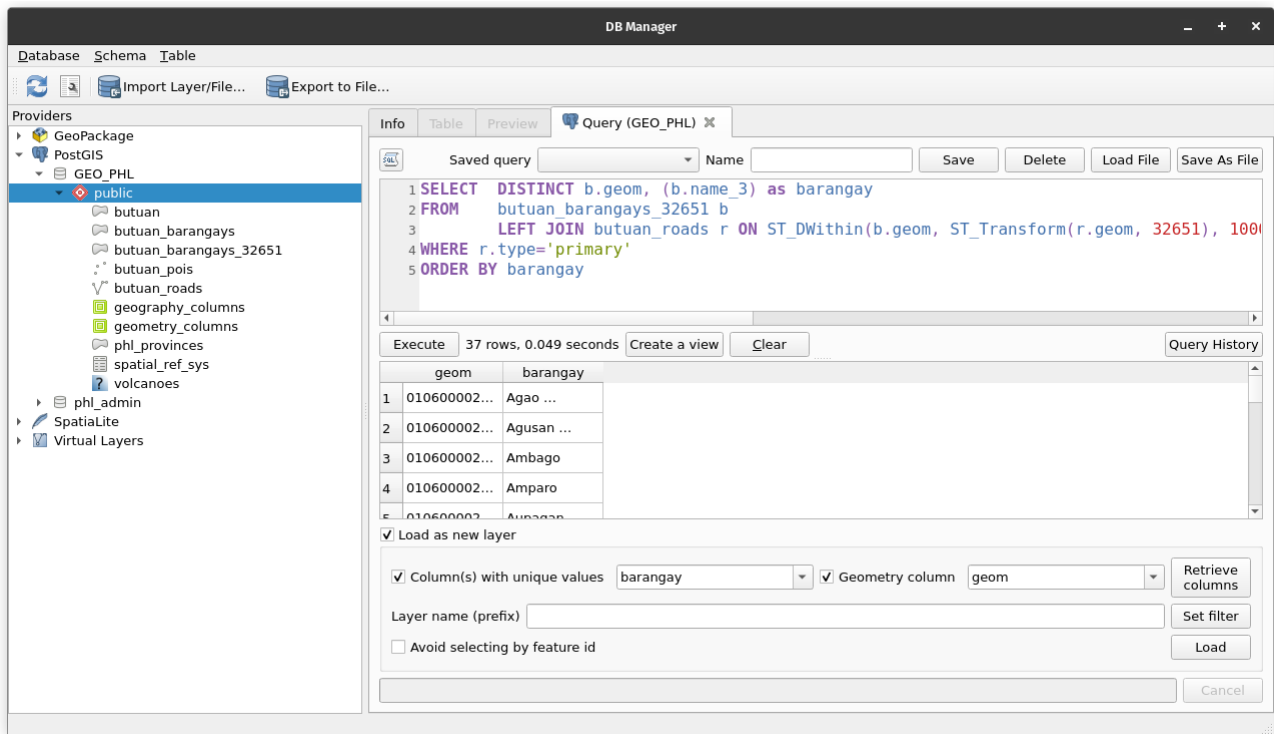
ST_Buffer — Computes a geometry covering all points within a given distance from a geometry.

ST_Union — Computes a geometry representing the point-set union of the input geometries.

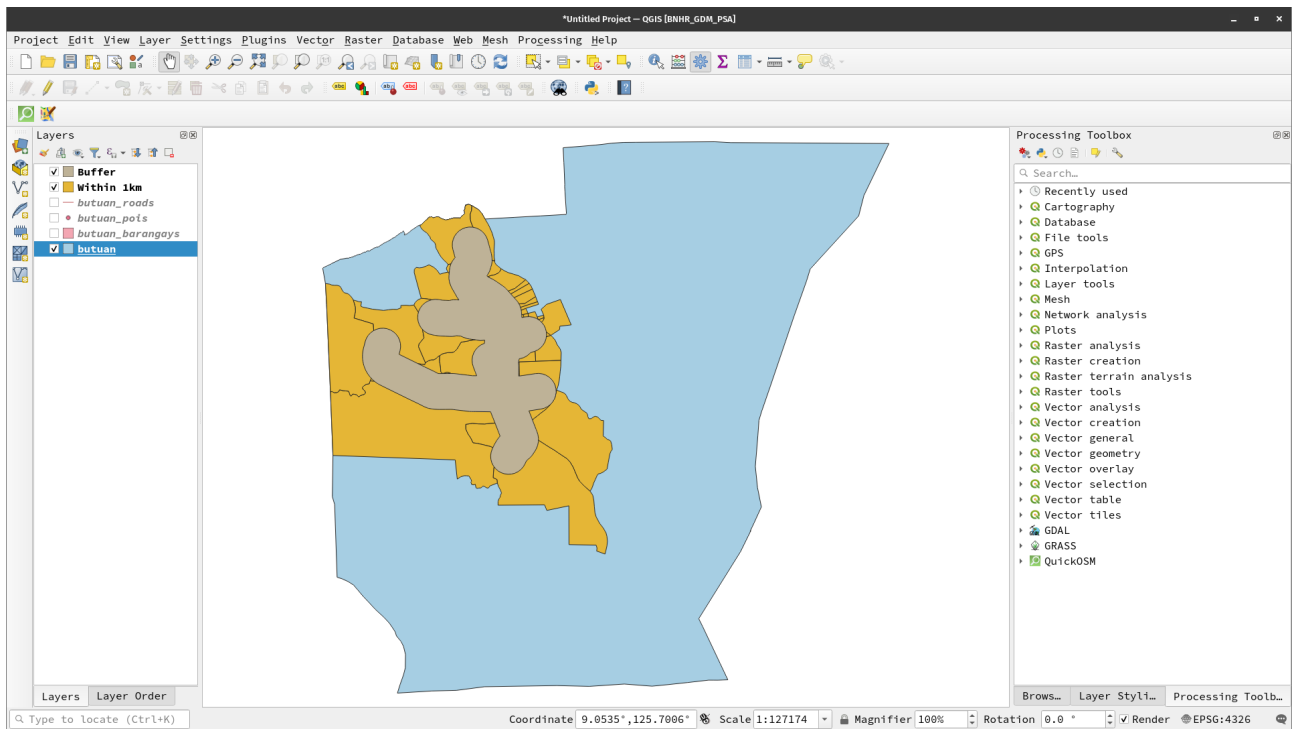
ST_Transform — Return a new geometry with coordinates transformed to a different spatial reference system.



This work and its contents by Ben Hur Pintor is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)



This work and its contents by Ben Hur Pintor is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).
 @bnhr.xyz | @bnhrdotxyz | mastodon.social/@benhur07b

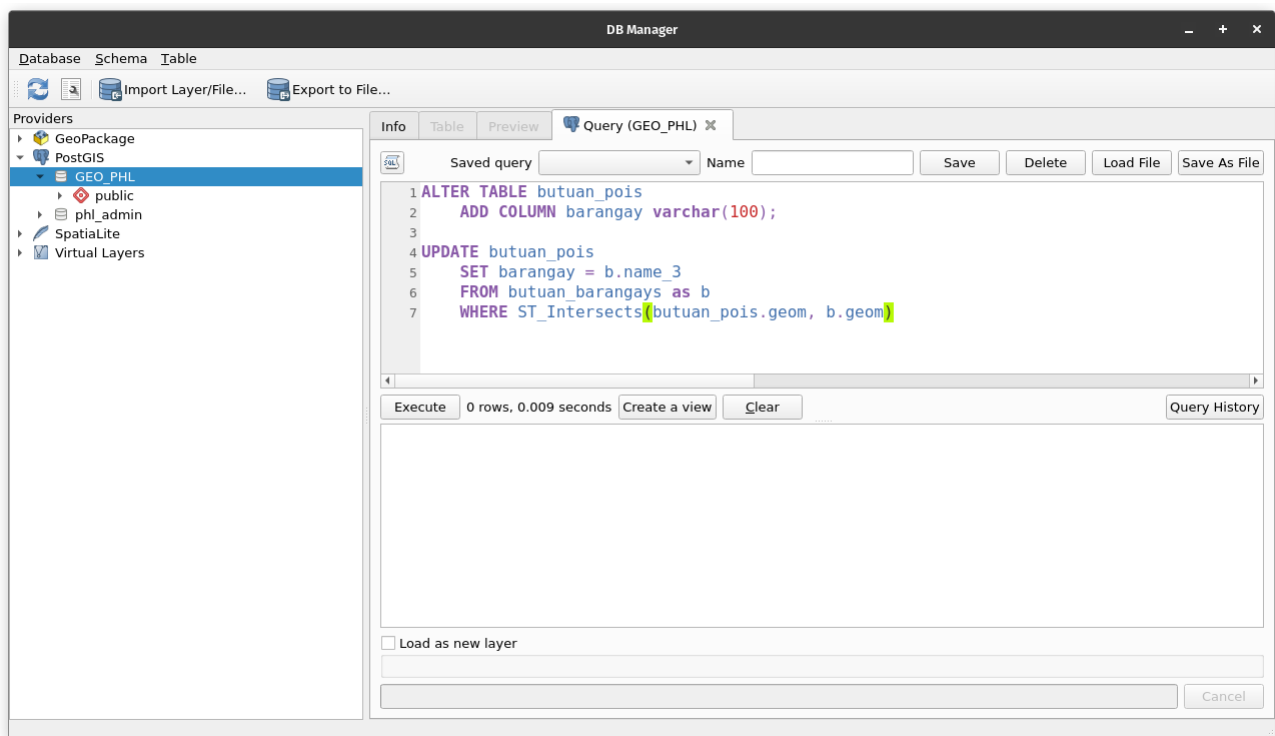


This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[@bnhr.xyz](#) | [@bnhrdotxyz](#) | [mastodon.social/@benhur07b](#)

Tag all the POIs per Barangay.

```
ALTER TABLE butuan_pois
  ADD COLUMN barangay varchar(100);

UPDATE butuan_pois
  SET barangay = b.name_3
  FROM butuan_barangays as b
  WHERE ST_Intersects(butuan_pois.geom, b.geom)
```



Functions

ST_Intersects — Tests if two geometries intersect (they have at least one point in common)



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).
[f @bnhr.xyz](#) | [y @bnhrdotxyz](#) | [m mastodon.social/@benhur07b](#)

DB Manager

Database Schema Table

Import Layer/File... Export to File...

Providers

- GeoPackage
- PostGIS
 - GEO_PHL
 - public
 - phl_admin
 - Spatialite
 - Virtual Layers

Info Table Preview Query (GEO_PHL) x

Saved query Name Save Delete Load File Save As File

```
1 SELECT * FROM butuan_pois;
```

Execute 31 rows, 0.002 seconds Create a view Clear Query History

	gid	category	name	geom	barangay
1	1	Automotive	Fuel:Phoenix	010100002...	Bancasi
2	2	Automotive	Fuel:Shell	010100002...	Bonbon
3	3	Automotive	Fuel:Petron	010100002...	Bonbon
4	4	Automotive	Fuel:Shell	010100002...	Bonbon
5	5	Automotive	Fuel:Caltex	010100002...	Bonbon
6	6	Automotive	Fuel:Petron	010100002...	Mahay

☐ Load as new layer

Cancel



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

[f @bnhr.xyz](#) |
 [y @bnhrdotxyz](#) |
 [m mastodon.social/@benhur07b](#)

Snap all the POIs to the Butuan Roads.

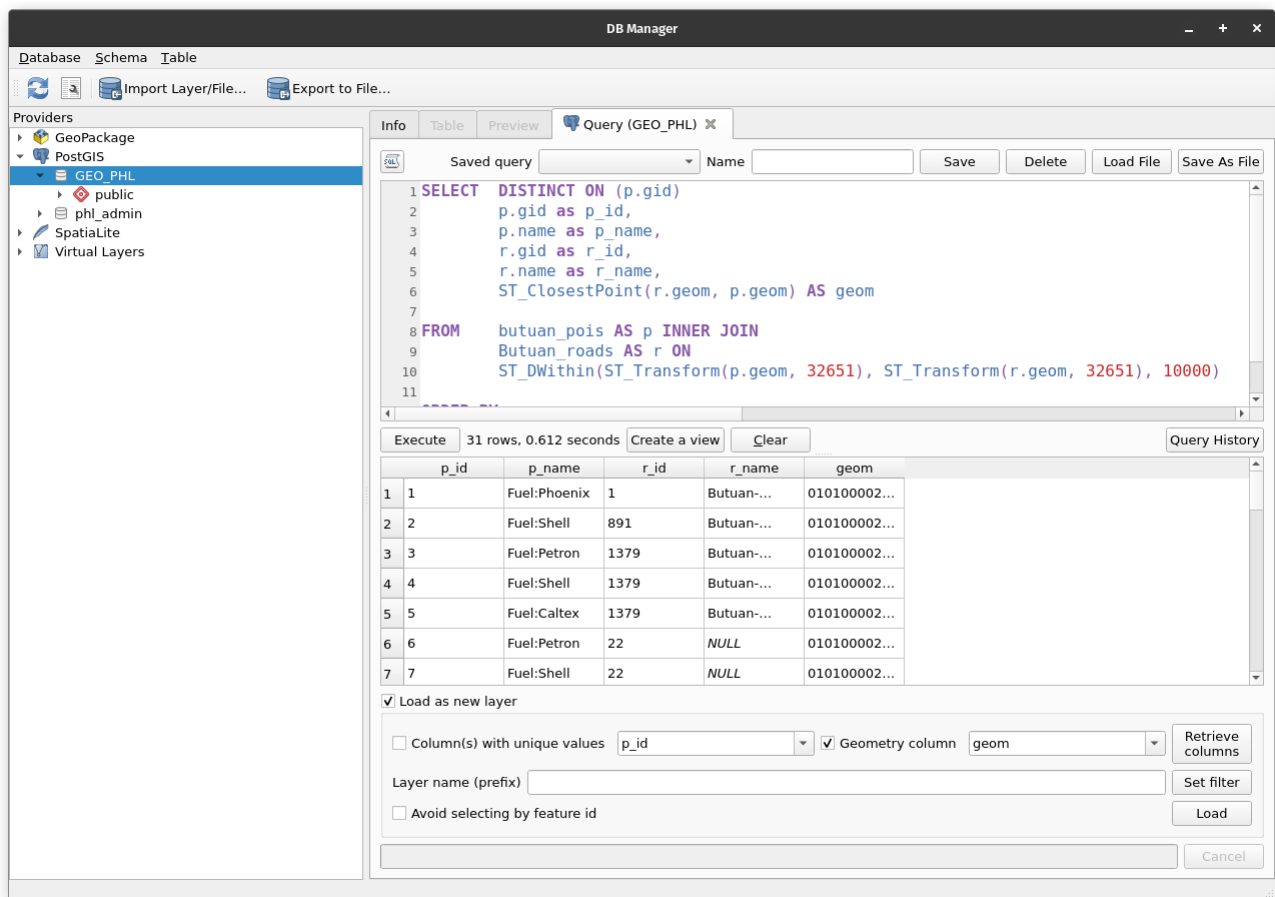
```

SELECT      DISTINCT ON (p.gid)
                p.gid as p_id,
                p.name as p_name,
                r.gid as r_id,
                r.name as r_name,
                ST_ClosestPoint(r.geom, p.geom) AS geom

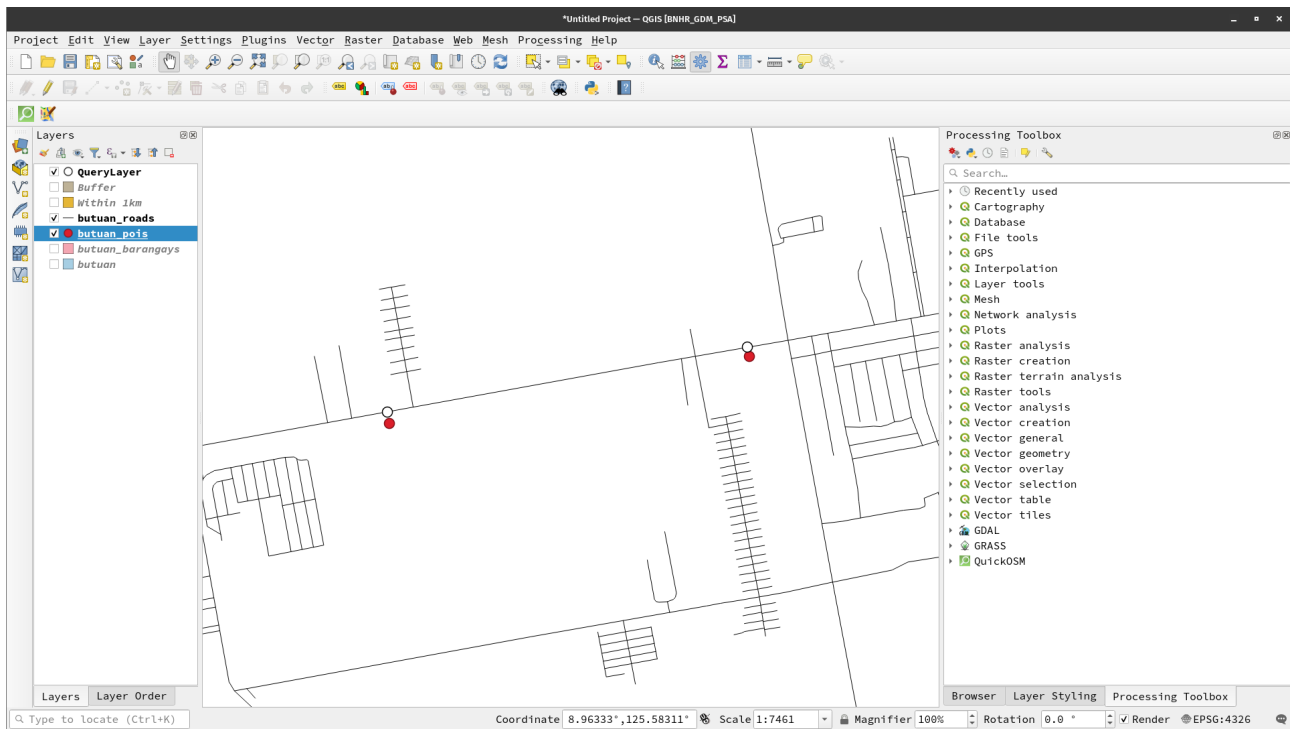
FROM         butuan_pois AS p INNER JOIN
                Butuan_roads AS r ON
                ST_DWithin(ST_Transform(p.geom, 32651), ST_Transform(r.geom,
32651), 10000)

ORDER BY
                p.gid, ST_Distance(r.geom, p.geom)

```



This work and its contents by Ben Hur Pintor is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).
[f @bnhr.xyz](https://www.facebook.com/bnhr.xyz) | [y @bnhrdotxyz](https://www.youtube.com/channel/UCbnhrdotxyz) | [m mastodon.social/@benhur07b](https://mastodon.social/@benhur07b)



Functions

ST_ClosestPoint — Returns the 2D point on g1 that is closest to g2. This is the first point of the shortest line from one geometry to the other.

ST_DWithin — Tests if two geometries are within a given distance

ST_Transform — Return a new geometry with coordinates transformed to a different spatial reference system.

ST_Distance — Returns the distance between two geometry or geography values



This work and its contents by [Ben Hur Pintor](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

[@bnhr.xyz](#) | [@bnhrdotxyz](#) | [mastodon.social/@benhur07b](#)

Spatial joins

Spatial joins are a fundamental function to any spatial application—spatial databases included. It is a technique that allows you to combine information from different tables using the spatial relationships/predicates that were previously discussed as the join key. A lot of the common “GIS analysis” questions that you will encounter can be answered using spatial joins.

For example, in the earlier exercises where we identified barangays within a specified bounding box or tagged/sampled Points of Interest (POIs) with their respective barangays, these tasks were accomplished through the application of spatial joins. Particularly, we used the `ST_Intersects` function to check whether a spatial relationship (i.e. Intersects) exists between two geometries (or tables) to determine whether to join them or not.

In general, any function that provides a true or false relationship between two tables can be used to drive a spatial join but the most common ones you will use are: `ST_Intersects`, `ST_Contains`, and `ST_DWithin`.

Let’s try to do more spatial joins, this time using the tutorial¹ from postgis.net itself.

IMPORTANT:

- Get data here: <https://s3.amazonaws.com/s3.cleverelephant.ca/postgis-workshop-2020.zip>
- See exercise here: https://postgis.net/workshops/postgis-intro/joins_advanced.html

¹ This material was originally created by Boundless. Subsequent revisions by Paul Ramsey and others.



Nearest neighbor

Another common spatial question that we want to answer involves finding the nearest feature from/to a query feature. For example, the nearest Jollibee from your home. Unlike a distance search, which specifies a fixed distance range, the "nearest neighbor" search is not limited by a specific distance; it simply aims to identify the closest feature, regardless of how far away it is.

PostgreSQL has a cool features for answering nearest neighbor problems: the **"order by distance"** operator ($\leftrightarrow$). This operator enduces the database to **optimize the retrieval of results** by leveraging an index to return a sorted set of features. By using the "order by distance" operator, a **nearest neighbor query can easily return the "N nearest features" by adding sorting and limiting the result set to N entries.**

The "order by distance" operator is applicable to both geometry and geography data types. The primary distinction between the two types is the nature of the distance value returned:

- **geometry data:** $\leftrightarrow$ returns the same result as ST_Distance, which is dependent on the units of the spatial reference system in use.
- **geography data:** the distance value returned represents sphere distance, as opposed to the more precise spheroidal distance provided by ST_Distance(geography, geography).

Nearest neighbor join

The index-assisted "order by" operator does come with a significant limitation: it only works for a single geometry literal on one side of the operator. This is okay for finding the nearest objects to a single query object (or location) but is not useful when the objective is to identify the nearest neighbor for each item in a set of candidate objects. For example: if you want to find the nearest Jollibee for each of your friends' houses.

Fortunately, there's a powerful SQL language feature that enables us to execute a query iteratively within a loop: the **LATERAL** join. This feature allows you to run your nearest neighbor query repeatedly in a loop.

Let's try some nearest neighbor analysis using the tutorial¹ from postgis.net itself.

IMPORTANT:

- Get data here: <https://s3.amazonaws.com/s3.cleverelephant.ca/postgis-workshop-2020.zip>
- See exercise here: <https://postgis.net/workshops/postgis-intro/knn.html>

¹ This material was originally created by Boundless. Subsequent revisions by Paul Ramsey and others.



Trigger functions and history tracking

History tracking is a common requirement for enterprise or organization-wide databases as it allows you to have an audit trail of the changes to the database over time. This helps you answer questions about what data was modified, when it was modified, who made the changes, and the previous states of the data.

Using trigger functions, you can create your own history/change tracking for any table directly on your database. It works by creating a separate history/audit table that records—for example—for every:

- **creation/addition:** who created it and when
- **deletion:** who deleted it and when
- **update:** adding a deletion record (for the previous state) and a creation record (for the new state)

Let's try some nearest neighbor analysis using the tutorial¹ from postgis.net itself.

IMPORTANT:

- Get data here: <https://s3.amazonaws.com/s3.cleverelephant.ca/postgis-workshop-2020.zip>
- See exercise here: https://postgis.net/workshops/postgis-intro/history_tracking.html

¹ This material was originally created by Boundless. Subsequent revisions by Paul Ramsey and others.

