Lab 02. Spatial Data

**Due date:** Thursday, Feb 6 submitted as Word document to Canvas ***Lab02*** link. This lab counts 9 % toward your total grade.

**Objectives:**

Please execute the SQL script provided below to create multiple geometric features required for completing Lab02. After running the SQL script, use the Geometry Viewer to verify the generated geometries. The resulting geometries should resemble those depicted in Figure 1. Ensure to replace ch03 with your designated schema name before executing the script.

**Format of answer:** Submit your answers as a **Word document or pdf** with graphs under **Output** section, properly labeled in the task sequence.

**Notice:** All SQL commands are in blue color

**In-Class Exercise (4pts)**

PostGIS maintains several system tables and views (we can call it “metatables”) that store metadata about spatial data and coordinate systems. Let’s focus on two of them: one is the **geometry\_columns** and the other one is **spatial\_ref\_sys**.

A close-up of a table

Description automatically generated

Spatial\_ref\_sys stores all known coordinate systems (SRIDs) supported by PostGIS.

Geometry\_columns is the table that tracks all geometry columns in the database. Let’s type the following command in the query tool:

SELECT \* FROM geometry\_columns;

A screenshot of a computer

Description automatically generated

Here are the information of each columns:

**f\_table\_schema**: schema name (e.g., ch02, ch03, public)

**f\_table\_name**: table name (e.g., clarku, restaurants, streets, boston\_geometry)

**f\_geometry\_column**: name of the geometry column (e.g., point, linestring, geom)

**coord\_dimension**: coordinate dimensions (2 (x, y), 3 (x, y, z), or 4 (x, y, z, m))

**srid**: spatial reference ID (e.g., 4326 for WGS84; 102003 for Albers Equal Area)

**type**: Geometry type (e.g., POINT, POINTM, MULTIPOLYGON, LINESTRING, etc.)

In-Class Practice: The Wizard’s Guide to PostGIS Magis

Hi Young Wizards, today we will use PostGIS to map the **Platform 9¾ (point), Diagon Alley (LineString), Hogwarts Castle (Polygon), and the Forbidden Forest (Polygon with a Hole)**.

Let’s do it step by step:

1. Create a `Hogwarts` schema to organize your magical GIS data.

CREATE SCHEMA IF NOT EXISTS hogwarts;

You will see the Hogwarts schema under your current database

A screenshot of a computer

Description automatically generated

2. Create the `magical\_locations` table to store geometries.

CREATE TABLE hogwarts.magical\_locations (

id SERIAL PRIMARY KEY, -- Unique identifier (like a magical seal)

name VARCHAR(100), -- Location name (e.g., "Platform 9¾")

geom GEOMETRY(Geometry, 4326) -- Geometry column (stores shapes in WGS84)

);

A close up of words

Description automatically generated

3. Add Platform 9¾ (King’s Cross Station, London) as a POINT.

INSERT INTO hogwarts.magical\_locations (name, geom)

VALUES (

'Platform 9¾',

ST\_SetSRID(ST\_GeomFromText('POINT(-0.1239 51.5325)'), 4326)

);

4. Add Diagon Alley as a STRINGLINE along Charing Cross Road, London

INSERT INTO hogwarts.magical\_locations (name, geom)

VALUES (

'Diagon Alley',

ST\_SetSRID(ST\_GeomFromText(

'LINESTRING(-0.1278 51.5074, -0.1303 51.5155)' -- From Trafalgar Square to St. Giles

), 4326)

);

5. Add Hogwarts Castle (Alnwick Castle, Alnwick, UK) as a POLYGON.

INSERT INTO hogwarts.magical\_locations (name, geom)

VALUES (

'Hogwarts Castle',

ST\_SetSRID(ST\_GeomFromText(

'POLYGON((

-1.7063 55.4133,

-1.7050 55.4125,

-1.7040 55.4130,

-1.7055 55.4140,

-1.7063 55.4133

))'

), 4326)

);

6. Add the Forbidden Forest (Forest of Dean) as a Polygon With Hole (Coleford GL16 7EG, UK).

INSERT INTO hogwarts.magical\_locations (name, geom)

VALUES (

'Forbidden Forest',

ST\_SetSRID(ST\_GeomFromText(

'POLYGON(

(-2.5500 51.8000, -2.5400 51.8000, -2.5400 51.7900, -2.5500 51.7900, -2.5500 51.8000),

(-2.5450 51.7950, -2.5450 51.7930, -2.5430 51.7950, -2.5450 51.7950)

)'

), 4326)

);

7. View Your Map

SELECT \* FROM hogwarts.magical\_locations;

A screenshot of a computer

Description automatically generated

8. Use **SELECT…FROM…WHERE** to uncover hidden magical locations!

8.1 List out geometry\_columns again (replace \_\_\_\_\_ with your answer, keep in red; and paste the screenshot)

SELECT \* FROM geometry\_columns;

8.2 Filter by Name: Find Diagon Alley (replace \_\_\_\_\_ with your answer, keep in red; and paste the screenshot)

SELECT name, ST\_AsText(geom)

FROM hogwarts.magical\_locations

WHERE name = ‘Diagon Alley’;

Tips: ST\_AsText(geom) is a function in PostGIS converts a geometry into Well-Known Text (WKT), which is a human-readable format

8.3 Filter by Type: Find all Polygons (replace \_\_\_\_\_ with your answer, keep in red; and paste the screenshot)

SELECT name, ST\_GeometryType(geom)

FROM hogwarts.magical\_locations

WHERE ST\_GeometryType(geom) = 'ST\_Polygon';

Tips: ST\_GeometryType() is a function to check the geometry types for rows.

8.4 List Geometry Types for All Rows by using **SELECT… FROM…** (replace \_\_\_\_\_ with your answer, keep in red; and paste the screenshot)

SELECT name, ST\_GeometryType(geom)

FROM hogwarts.magical\_locations

8.5 SELECT all locations EXCEPT the Forbidden Forest to avoid danger! (replace \_\_\_\_\_ with your answer, keep in red; and paste the screenshot)

SELECT \*

FROM hogwarts.magical\_locations

WHERE <> 'Forbidden Forest';

Tips: Not equal to: **<>** / **!=**

**On Your Own (5pts)**

After executing the SQL queries using the specified functions for each Task, provide the following outputs:

* A screenshot of the Geometry Viewer displaying the results.
* SQL statements used to retrieve the geometry properties.

Create the table for the tasks **in schema ch03**:

CREATE TABLE ch03.boston\_geometry (name varchar(100), geom geometry (Geometry, 4326));

INSERT INTO ch03.boston\_geometry VALUES

('Point', ST\_GeomFromText('POINT( -71.1052 42.3508)', 4326)),

('Linestring', ST\_GeomFromText('LINESTRING(

-71.1173 42.3513 ,

-71.1033 42.3496 ,

-71.0971 42.3490 ,

-71.0887 42.3491

)', 4326)),

('Polygon', ST\_GeomFromText('POLYGON((

-71.0724 42.3554,

-71.0632 42.3577,

-71.0622 42.3566,

-71.0646 42.3525,

-71.0706 42.3519,

-71.0724 42.3554

))', 4326)),

('PolygonWithHole', ST\_GeomFromText('POLYGON((

-71.1145 42.3473,

-71.1108 42.3468,

-71.1105 42.3454,

-71.1136 42.3446,

-71.1145 42.3473

), (

-71.1120 42.3466,

-71.1113 42.3461,

-71.1117 42.3459,

-71.1121 42.3465,

-71.1120 42.3466

))', 4326));

A map of a city

Description automatically generated

**Figure 1 Geometry of boston\_geometry**

Task 1. Utilize the following PostGIS functions to retrieve key properties of the geometries stored in the database:

* ST\_GeometryType() – Determines the type of geometry (e.g., Point, LineString, Polygon).
* ST\_NDims() – Returns the number of dimensions of the geometry (2D or 3D).
* ST\_SRID() – Retrieves the Spatial Reference Identifier (SRID) of the geometry.

Task 1 Output:

A screenshot of a computer

Description automatically generated

SELECT name, ST\_GeometryType(geom), ST\_NDims(geom), ST\_SRID(geom)

FROM ch03.boston\_geometry;

Task 2. Write an SQL query to select all geometries from the database and convert them to Well-Known Text (WKT) format. (Hint: Use the function ST\_AsText() to convert geometries to WKT format.)

Task 2 Output:

A screenshot of a computer

Description automatically generated

SELECT name, ST\_AsText(geom) FROM ch03.boston\_geometry;

Task 3. Write an SQL query using the structure SELECT … FROM … WHERE … to retrieve the coordinates of the geometry from st\_astext in the boston\_geometry table where the name is 'Point'.

Task 3 Output:

A screenshot of a computer

Description automatically generated

SELECT ST\_AsText(geom)

FROM ch03.boston\_geometry

WHERE name = 'Point';

Task 4. Write an SQL query using the structure SELECT … FROM … WHERE … to retrieve the length of the Linestring geometry from the boston\_geometry table using the ST\_Length() function.

Task 4 Output:

A screenshot of a computer

Description automatically generated

SELECT ST\_Length(geom)

FROM ch03.boston\_geometry

WHERE name = 'Linestring';

Task 5. Write an SQL query using the structure SELECT … FROM … WHERE … to calculate the area of all polygon geometries in the boston\_geometry table where the name includes the character pattern ‘Polygon%’ (hint: WHERE name LIKE ‘Polygon%’).

Task 5 Output:

A screenshot of a phone

Description automatically generated

SELECT name, ST\_Area(geom)

FROM ch03.boston\_geometry

WHERE name LIKE 'Polygon%';