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

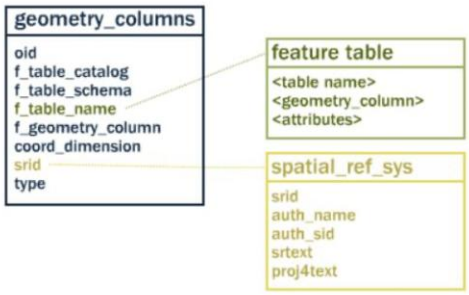
In-Class Exercise (4pts)

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**.

METADATA TABLES

- PostGIS provides two tables to track and report on the geometry types available in a given database.
- The first table, *spatial_ref_sys*, defines all the spatial reference systems known to the database and will be described in greater detail later.
- The second table (actually, a view), *geometry_columns*, provides a listing of all "features" (defined as an object with geometric attributes), and the basic details of those features.

Table Relationships



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;

GEOMETRY_COLUMNS TABLE IN DATABASE

```
SELECT * FROM
geometry_columns;
```

Data Output | Messages | Notifications

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	spatialanalysis	public	us_tract_2020	geometry	2	102003	MULTIPOLYGON
2	spatialanalysis	ch02	clarku	p	2	0	POINT
3	spatialanalysis	ch02	clarku	pz	3	0	POINT
4	spatialanalysis	ch02	clarku	pm	3	0	POINTM
5	spatialanalysis	ch02	clarku	pzm	4	0	POINT
6	spatialanalysis	ch02	clarku	p_srid	2	4326	POINT
7	spatialanalysis	public	restaurants	geom	2	4326	MULTIPOINT
8	spatialanalysis	ch02	restaurants	geom	2	4326	MULTIPOINT
9	spatialanalysis	ch02	multi_street	geom	2	4326	MULTILINESTRING
10	spatialanalysis	ch02	streets	line_str	2	0	LINESTRING
11	spatialanalysis	ch02	streets	line_srid	2	4326	LINESTRING
12	spatialanalysis	ch02	campus	geom	2	4326	GEOMETRYCOLLECTION
13	spatialanalysis	ch02	pts_geom	geom_pts	2	4326	POINT

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

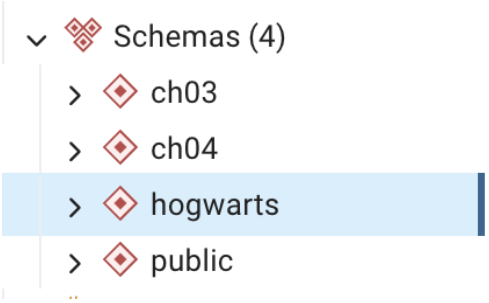
Hi Young Wizards, today we will use PostGIS to map the **Platform 9¾ (point), Diagon Alley (LineString), Hogwarts Castle (Polygon), and the Forbidden Forest (Polygon with a Hole)**.

Let's do it step by step:

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

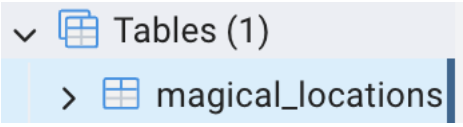
```
CREATE SCHEMA IF NOT EXISTS hogwarts;
```

You will see the Hogwarts schema under your current database



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)
);
```



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;
```

	id [PK] integer	name character varying (100)	geom geometry
1	1	Platform 9¾	0101000020E61000004772F90FE9B7BFBF5C8FC2F528C44940
2	2	Diagon Alley	0102000020E6100000020000000EBE2361AC05BC0BFC5FEB27BF2C049403D9B559FABADC0BF448B6CE7FBC14940
3	3	Hogwarts Castle	0103000020E6100000010000000500000005A3923A014DFBBFE9B7AF03E7B44B4048E17A14AE47FBBFCDCCCCCCCCB44B40DD2406819543FBBFBE9
4	4	Forbidden Forest	0103000020E6100000020000000500000066666666666604C06666666666E6494052B81E85EB5104C06666666666E6494052B81E85EB5104C085EB51

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)

____

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 = '____';
```

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 ____
FROM ____
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)

____

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 ____
WHERE ____;
```

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));
```

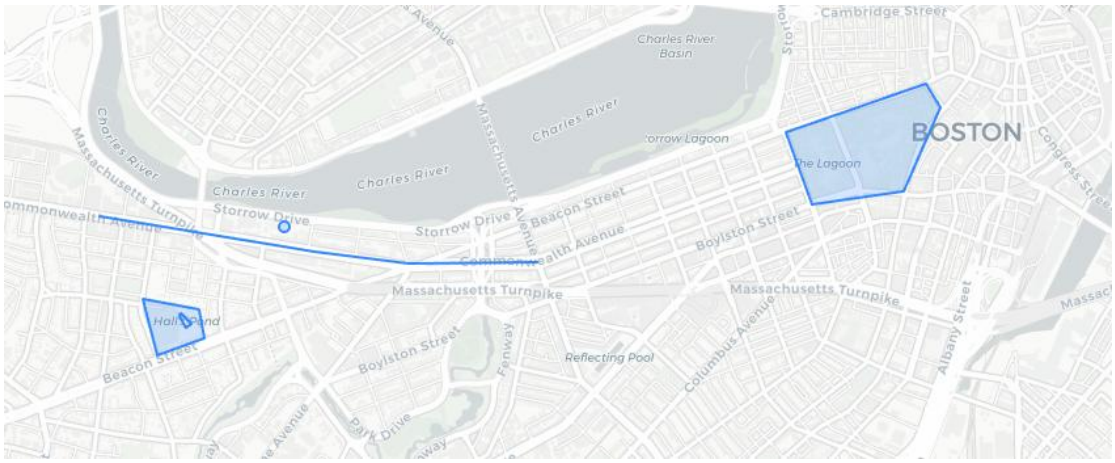


Figure 1 Geometry of boston_geometry

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:

Data Output | Messages | Geometry Viewer | Notifications

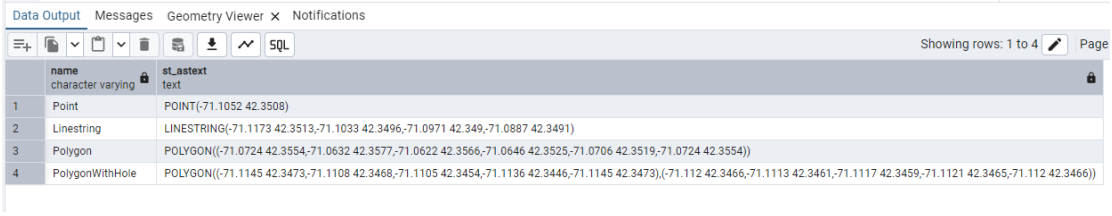
SQL

	name character varying	st_geometrytype text	st_ndims smallint	st_srid integer
1	Point	ST_Point	2	4326
2	Linestring	ST_LineString	2	4326
3	Polygon	ST_Polygon	2	4326
4	PolygonWithHole	ST_Polygon	2	4326

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:



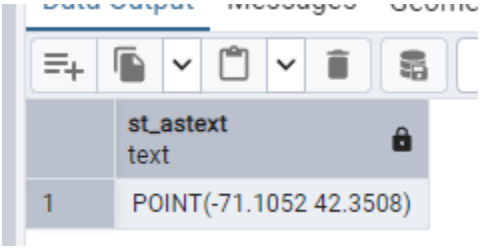
Data Output | Messages | Geometry Viewer | Notifications

Showing rows: 1 to 4 | Page

	name character varying	st_astext text
1	Point	POINT(-71.1052 42.3508)
2	Linestring	LINESTRING(-71.1173 42.3513,-71.1033 42.3496,-71.0971 42.349,-71.0887 42.3491)
3	Polygon	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))
4	PolygonWithHole	POLYGON((-71.1145 42.3473,-71.1108 42.3468,-71.1105 42.3454,-71.1136 42.3446,-71.1145 42.3473),(-71.112 42.3466,-71.1113 42.3461,-71.1117 42.3459,-71.1121 42.3465,-71.112 42.3466))

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'.

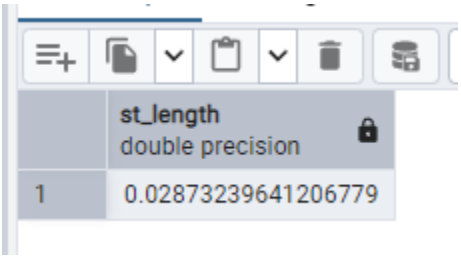
Task 3 Output:



	st_astext text
1	POINT(-71.1052 42.3508)

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.

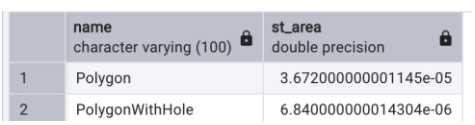
Task 4 Output:



	st_length double precision
1	0.02873239641206779

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:



	name character varying (100)	st_area double precision
1	Polygon	3.672000000001145e-05
2	PolygonWithHole	6.840000000014304e-06