INFS4205/7205 Advanced Techniques for High Dimensional Data

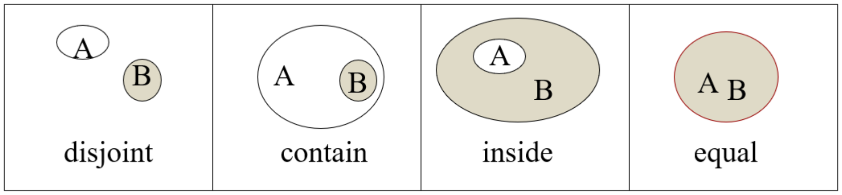
Tutorial 1: Spatial Database Introduction

Semester 1, 2021

Question 1: The 9-Intersectin Matrix is a topological model and a standard used to describe the spatial relations of two polygons (although it can be extended to line, we only discuss polygon here). Each polygon has three parts: A boundary, an interior, and an exterior. Therefore, for any two polygons, their topological relationships can be characterized using nine possible intersections between their interiors/boundaries/exteriors.

$$\begin{pmatrix} A_b \cap B_b & A_b \cap B_i & A_b \cap B_e \\ A_i \cap B_b & A_i \cap B_i & A_i \cap B_e \\ A_e \cap B_b & A_e \cap B_i & A_e \cap B_e \end{pmatrix}$$

Depending on the *True/False* of each intersection, eight and only eight topological relations can be identified. Please discuss and understand them.



$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

disjoint

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

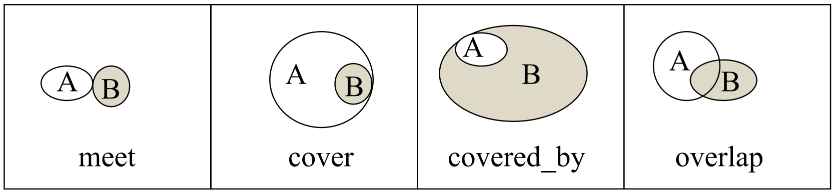
contain

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

inside

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

equal



$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

meet

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

cover

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

covered_by

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

overlap

Question 2: A demonstration of spatial database using PostGIS

PostGIS is a spatial extension that adds support for geographic objects to the PostgreSQL (Just like Oracle Spatial to Oracle). This tutorial works as a demonstration of some basic spatial operations. Some of the useful tools and link:

- PostGIS, you can download and try it on your computer : https://postgis.net
- The complete tutorial can here: https://postgis.net/workshops/postgis-intro/
- To view the spatial data, you can use the free QGIS: http://qgis.org
- The New York dataset used in this tutorial: http://s3.cleverelephant.ca/postgis-workshop-2018.zip

Spatial Database Creation

After the installation, open the pgsql command line to interact with PostgreSQL

- Create a database

  ```
  CREATE DATABASE sdbms;
  ```
- Connect to the new database

  ```
  \c sdbms
  ```
- Load the PostGIS spatial extension

  ```
  Create EXTENSION postgis;
  ```
- Confirm that PostGIS is installed

  ```
  SELECT postgis_full_version();
  ```

Data Import

The data used in this tutorial is the format of “shapefile”, which commonly refers to a collection of files with .shp, .shx, .dbf that have the same prefix name. A shapefile is a simple, nontopological format for storing the geometric location and attribute information of geographic features.

Geographic features in a shapefile can be represented by points, lines, or polygons (areas). The workspace containing shapefiles may also contain dBASE tables, which can store additional attributes that can be joined to a shapefile's features.

For example, the three mandatory files of the nyc_street shapefile are:

- **nyc_street.shp**: The main shape file, store the feature geometry like points, lines, and polygons (areas). The area features are represented as closed loop, double-digitized polygons. All the geometries are stored as a set of vector coordinates.
- **nyc_street.shx**: The shape index file, stores the offset of the corresponding main file record from the beginning of the main file.
- **nyc_street.dbf**: The attribute table file, stores the columnar attributes for each shape in dBASE format. It can be joined to a shapefile's features.

More details of the shapefile can be found here:
http://downloads.esri.com/support/whitepapers/mo_/shapefile.pdf

The earth is not flat, and there is no simple way of putting it down on a flat paper map, so people have come up with all sorts of ingenious solutions. Some projections preserve area, so all objects have a relative size to each other; other projections preserve angles; some projections try to find a good intermediate mix with only little distortion on several parameters. Which projection to choose depends on how you will be using the data.

To import a shapefile into the database, we first use shp2pgsql command line tool that shipped with PostGIS to convert it into SQL script. For example:

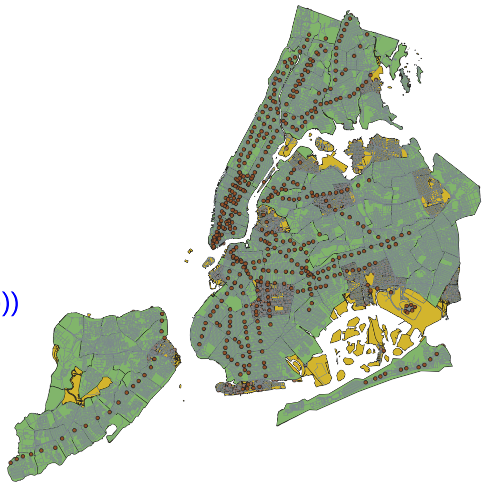
- shp2pgsql -s 26918 nyc_streets.shp nyc_streets > nyc_streets.sql

The -s parameter set the *Spatial Reference Identifier* (SRID), which defines all the parameters of the data's geographic coordinate system and projection. All the other shapefiles can be converted similarly.

After that, in our sdmbs, we import these SQL scripts using command \i

- \i path\to\nyc_streets.sql

Now you can use QGIS to connect to your database and see these geometries.



POINT(0 0)
MULTIPOINT((0 0),(1 2))

LINESTRING(0 0,1 1,1 2)
MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4)))

POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))

GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))

Data Description

Before we query into the data, let's first describe their attributes:

nyc_census_blocks

A census block is the smallest geography for which census data is reported. All higher level census geographies (block groups, tracts, metro areas, counties, etc) can be built from unions of census blocks.

blkid	A 15-digit code that uniquely identifies every census **block**. Eg: 360050001009000
popn_total	Total number of people in the census block
popn_white	Number of people self-identifying as "White" in the block
popn_black	Number of people self-identifying as "Black" in the block
popn_nativ	Number of people self-identifying as "Native American" in the block
popn_asian	Number of people self-identifying as "Asian" in the block
popn_other	Number of people self-identifying with other categories in the block
boroname	Name of the New York borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
geom	Polygon boundary of the block

nyc_neighborhoods

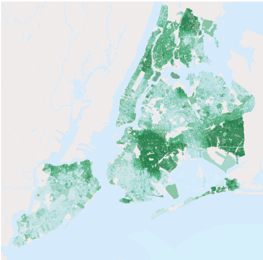
name	Name of the neighborhood
boroname	Name of the New York borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
geom	Polygon boundary of the neighborhood

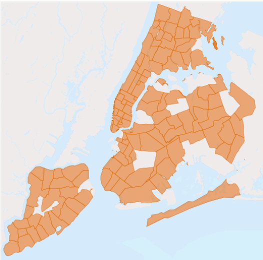
nyc_streets

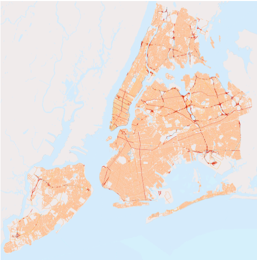
name	Name of the street
oneway	Is the street one-way? “yes” = yes, “” = no
type	Road type (primary, secondary, residential, motorway)
geom	Linear centerline of the street

nyc_subway_stations

name	Name of the station
borough	Name of the New York borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
routes	Subway lines that run through this station
transfers	Lines you can transfer to via this station
express	Stations where express trains stop, “express” = yes, “” = no
geom	Point location of the station


nyc_census_blocks


nyc_neighborhoods


nyc_streets

nyc_subway_station

Simple Spatial SQL

1. What is the population of the City of New York?

```
SELECT Sum(popn_total) AS population
FROM     nyc_census_blocks;
```

try with:

```
SELECT popn_total, blkid
FROM nyc_census_blocks;
```

2. What is the population of the Bronx?

```
SELECT Sum(popn_total) AS population
FROM     nyc_census_blocks
WHERE  boroname = 'The Bronx';
```

3. For each borough, what percentage of the population is white?

```
SELECT
  boroname,  100 * Sum(popn_white)/Sum(popn_total) AS white_pct
FROM         nyc_census_blocks
GROUP BY boroname;
```

Geometry Query

1. What is the area of the ‘West Village’ neighborhood?

   ```
   SELECT ST_Area(geom)
   FROM   nyc_neighborhoods
   WHERE  name = 'West Village';
   ```

   ST_Area(): Returns the area of a polygonal geometry.

2. What is the area of Manhattan in acres?

   ```
   SELECT  Sum(ST_Area(geom)) / 4047
   FROM    nyc_neighborhoods
   WHERE   boroname = 'Manhattan';
   ```

3. How many census blocks in New York City have a hole in them?

   ```
   SELECT  Count(*)
   FROM    nyc_census_blocks
   WHERE   ST_NumInteriorRings(ST_GeometryN(geom,1)) > 0;
   ```

   ST_GeometryN(geom,n): Returns the nth geometry within a geometry object.

4. What is the total length of streets (in kilometers) in New York City?

   ```
   SELECT  Sum(ST_Length(geom)) / 1000
   FROM    nyc_streets;
   ```

   ST_NumInteriorRings(geom): Return the number of interior rings of the first polygon in the geometry. This will work with both POLYGON and MULTIPOLYGON types but only looks at the first polygon. Return NULL if there is no polygon in the geometry.

5. How long is ‘Columbus Cir’ (Columbus Circle)?

   ```
   SELECT  ST_Length(geom)
   FROM    nyc_streets
   WHERE   name = 'Columbus Cir';
   ```

6. How many polygons are in the ‘West Village’ multipolygon?

   ```
   SELECT  ST_NumGeometries(geom)
   FROM    nyc_neighborhoods
   WHERE   name = 'West Village';
   ```

7. What is the length of streets in New York City, summarized by type?

   ```
   SELECT      type, Sum(ST_Length(geom)) AS length
   FROM        nyc_streets
   GROUP BY    type
   ORDER BY    length DESC;
   ```

   sum up the length by type

Spatial Relationship Function

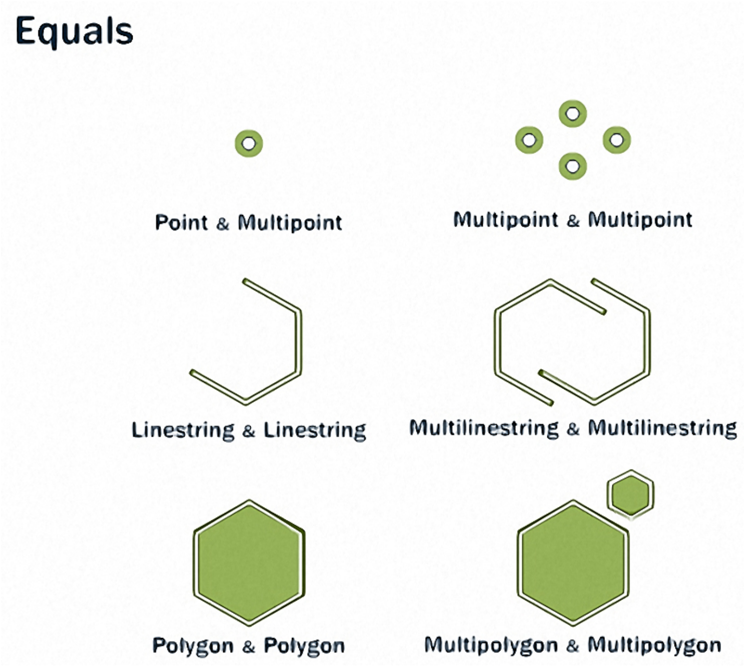
ST_Equals(geometry A, geometry B) tests the spatial equality of two geometries. It returns TRUE if two geometries of the same type have identical x,y coordinate values, i.e. if the second shape is equal (identical) to the first shape.

the ordering of points can be different but represent the same geometry structure

Example:

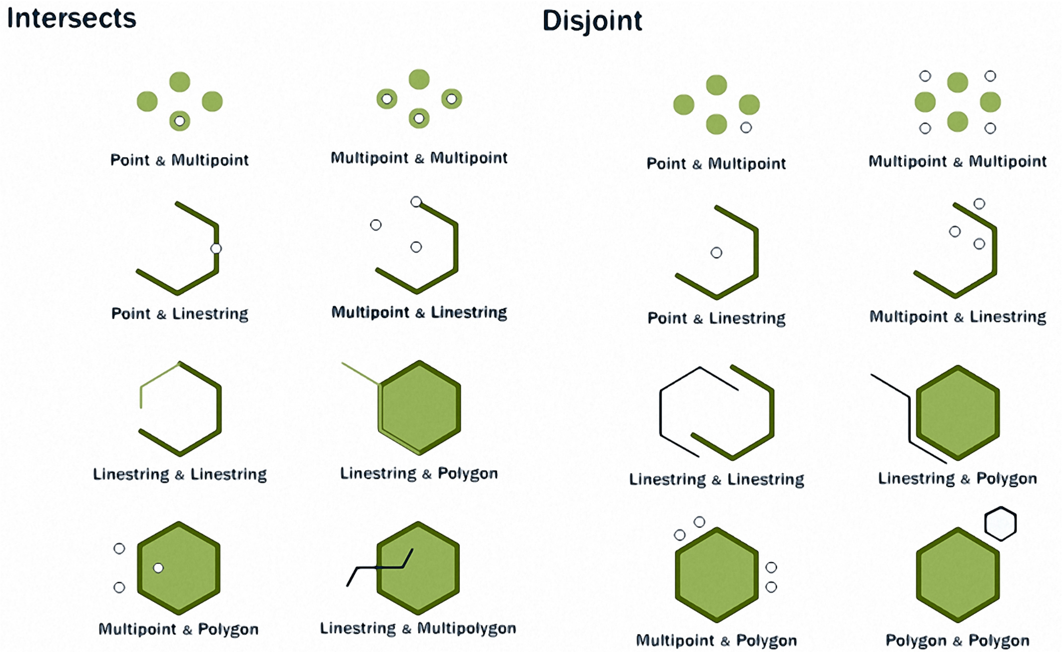
```
SELECT name, geom, ST_AsText(geom)
FROM   nyc_subway_stations
WHERE  name = 'Broad St';

SELECT  name
FROM    nyc_subway_stations
WHERE   ST_Equals(geom,
'0101000020266900000EEBD4CF27CF2141BC17D69516315141');
```

the code is the representation of a point



ST_Intersects(geometry A, geometry B) returns t (TRUE) if the two shapes have any space in common, i.e., if their boundaries or interiors intersect.

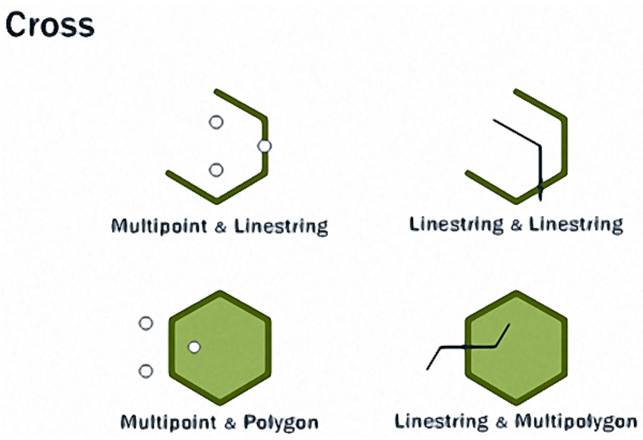
ST_Disjoint(geometry A , geometry B) is the Opposite of ST_INTERSECTS. If two geometries are disjoint, they do not intersect, and vice-versa. In fact, it is often more efficient to test "not intersects" than to test "disjoint" because the intersects tests can be spatially indexed, while the disjoint test cannot.

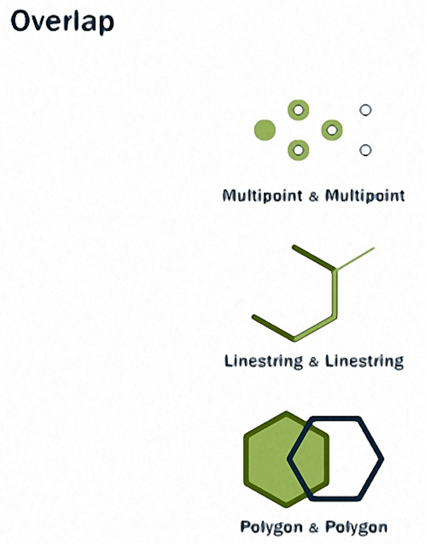


ST_Crosses takes two geometry objects and returns TRUE if their intersection "spatially cross", that is, the geometries have some, but not all interior points in common. The intersection of the interiors of the geometries must not be the empty set and must have a dimensionality less than the maximum dimension of the two input geometries.

ST_Crosses(geometry A, geometry B) returns t (TRUE) if the intersection results in a geometry whose dimension is one less than the maximum dimension of the two source geometries and the intersection set is interior to both source geometries.

ST_Overlaps(geometry A, geometry B) compares two geometries of the same dimension and returns TRUE if their intersection set results in a geometry different from both but of the same dimension.





Example:

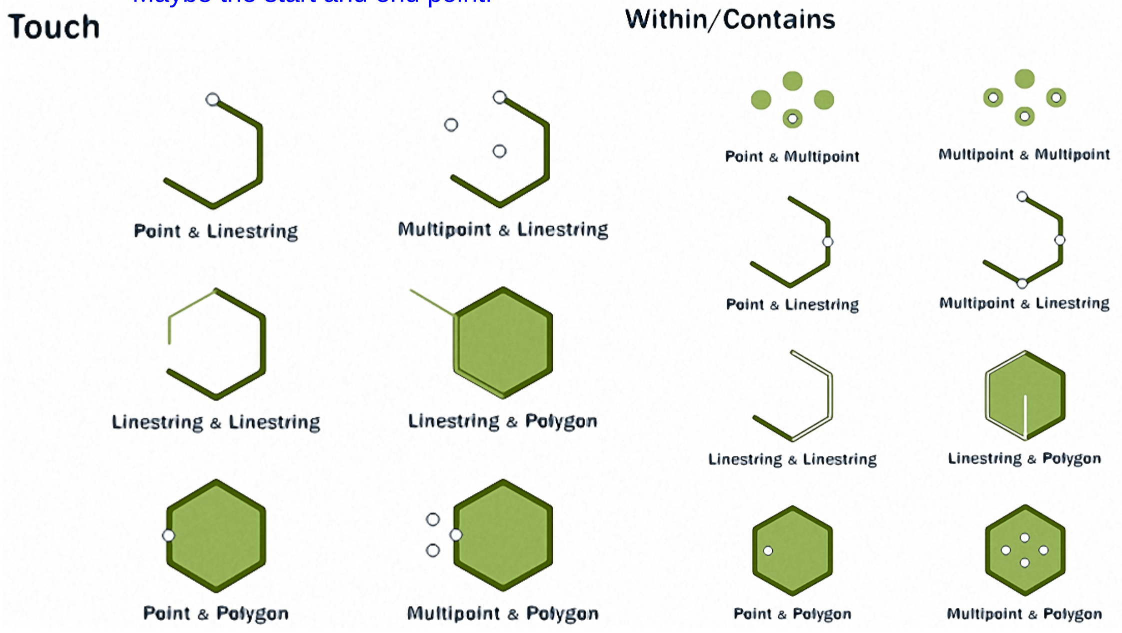
```
        SELECT  name, ST_AsText(geom)
        FROM    nyc_subway_stations
        WHERE   name = 'Broad St';
        SELECT name, boroname
        FROM    nyc_neighborhoods
        WHERE ST_Intersects(geom, ST_GeomFromText('POINT(583571
4506714)',26918));
```

ST_Touches tests whether two geometries touch at their boundaries, but do not intersect in their interiors. **ST_Touches(geometry A, geometry B)** returns TRUE if either of the geometries' boundaries intersect or if only one of the geometry's interiors intersects the other's boundary.

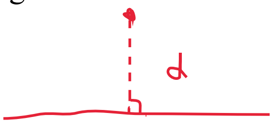
ST_Within(geometry A , geometry B) returns TRUE if the first geometry is completely within the second geometry. ST_Within tests for the exact opposite result of ST_Contains.

ST_Contains(geometry A, geometry B) returns TRUE if the second geometry is completely contained by the first geometry.

How to understand the boundary of a line?
Maybe the start and end point!



The **ST_Distance(geometry A, geometry B)** calculates the *shortest* distance between two geometries and returns it as a float. This is useful for actually reporting back the distance between objects.



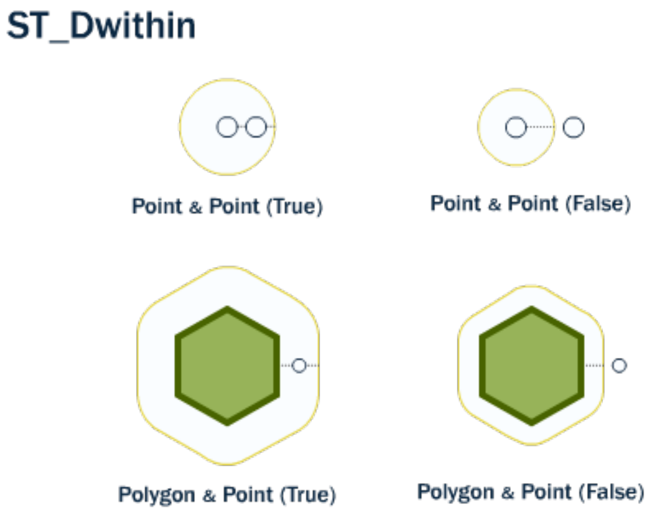
Example:

```
    SELECT ST_Distance(
            ST_GeometryFromText('POINT(0 5)'),
            ST_GeometryFromText('LINESTRING(-2 2, 2 2)'));
```

For testing whether two objects are within a distance of one another, the **ST_DWithin** function provides an index-accelerated true/false test. This is useful for questions like “how many trees are within a 500 meter buffer of the road?”. You don’t have to calculate an actual buffer, you just have to test the distance relationship.

Example:

```
    SELECT  name
    FROM    nyc_streets
    WHERE  ST_DWithin(
            geom,
            ST_GeomFromText('POINT(583571 4506714)',26918),
            10);
```



Spatial Relationship Query

1. “What is the geometry value for the street named ‘Atlantic Commons’?”

```
SELECT ST_AsText(geom)
FROM     nyc_streets
WHERE  name = 'Atlantic Commons';
```

2. “What neighborhood and borough is Atlantic Commons in?”

```
SELECT name, boroname
FROM    nyc_neighborhoods
WHERE ST_Intersects(            try with Cross?
        geom,
        ST_GeomFromText('LINESTRING(586782 4504202,586864
        4504216)', 26918));
```

3. “What streets does Atlantic Commons join with?”

```
SELECT  name
FROM     nyc_streets
WHERE  ST_DWithin(
        geom,
        ST_GeomFromText('LINESTRING(586782 4504202,586864
4504216)', 26918),
        0.1);
```

4. “Approximately how many people live on (within 50 meters of) Atlantic Commons?”

```
SELECT Sum(popn_total)
FROM nyc_census_blocks
WHERE ST_DWithin(
        geom,
        ST_GeomFromText('LINESTRING(586782 4504202,586864
4504216)', 26918),
        50);
```

Spatial Join Query

In the previous section, we explored spatial relationships using a two-step process: first we extracted a subway station point for ‘Broad St’; then, we used that point to ask further questions such as “what neighborhood is the ‘Broad St’ station in?”

The JOIN clause combines two FROM items. By default, we are using an INNER JOIN, but there are four other types of joins. For further information see the join_type definition in the PostgreSQL documentation.

Using a spatial join, we can answer the question in one step, retrieving information about the subway station and the neighborhood that contains it:

```
SELECT      subways.name AS subway_name,
            neighborhoods.name AS neighborhood_name,
            neighborhoods.boroname AS borough
FROM        nyc_neighborhoods AS neighborhoods
JOIN        nyc_subway_stations AS subways
ON          ST_Contains(neighborhoods.geom, subways.geom)
WHERE       subways.name = 'Broad St';
```

1. “What is the population and racial make-up of the neighborhoods of Manhattan?”

1. Locate NBs of Manhattan
2. Find same areas from census
3. sum up the population by NBs

```
        SELECT
        neighborhoods.name AS neighborhood_name,
        Sum(census.popn_total) AS population,
        100.0 * Sum(census.popn_white) / Sum(census.popn_total) AS
white_pct,
        100.0 * Sum(census.popn_black) / Sum(census.popn_total) AS black_pct
        FROM        nyc_neighborhoods AS neighborhoods
        JOIN        nyc_census_blocks AS census
        ON          ST_Intersects(neighborhoods.geom, census.geom)
        WHERE       neighborhoods.boroname = 'Manhattan'
        GROUP BY    neighborhoods.name
        ORDER BY    white_pct DESC;
```

2. “What subway station is in ‘Little Italy’? What subway route is it on?”

1. find the neighbourhood
2. find the station points contained by the neighborhood

```
SELECT      s.name, s.routes
FROM        nyc_subway_stations AS s
JOIN        nyc_neighborhoods AS n
      ON    ST_Contains(n.geom, s.geom)
WHERE       n.name = 'Little Italy';
```

3. “What are all the neighborhoods served by the 6-train?”

```
SELECT      DISTINCT n.name, n.boroname
FROM        nyc_subway_stations AS s
JOIN        nyc_neighborhoods AS n
      ON    ST_Contains(n.geom, s.geom)
WHERE       strpos(s.routes,'6') > 0;
```

strpos(str, substr): find the position, from where the substring is being matched within the string.

4. “After 9/11, the ‘Battery Park’ neighborhood was off limits for several days. How many people had to be evacuated?”

```
SELECT      Sum(popn_total)
FROM        nyc_neighborhoods AS n
JOIN        nyc_census_blocks AS c
       ON   ST_Intersects(n.geom, c.geom)
WHERE       n.name = 'Battery Park';
```

5. “What are the population density (people / km^2) of the ‘Upper West Side’ and ‘Upper East Side’?”

```
SELECT
  n.name,
  Sum(c.popn_total) / (ST_Area(n.geom) / 1000000.0) AS popn_per_sqkm
FROM        nyc_census_blocks AS c
JOIN        nyc_neighborhoods AS n
       ON   ST_Intersects(c.geom, n.geom)
WHERE       n.name = 'Upper West Side'
       OR   n.name = 'Upper East Side'
GROUP BY    n.name, n.geom;
```