

# Introduction to PostGIS

**<http://postgis.net/workshops/postgis-intro>**

# **Section 1 - Welcome**

# Questions, questions, so many ...

- We **really** want to hear questions.
- We **don't mind** stopping to answer them.
- Ask them **when they occur** to you.
- Really, just **blurt them out!**

## Download PgAdmin

- <https://www.pgadmin.org/download/>

## Download data bundle

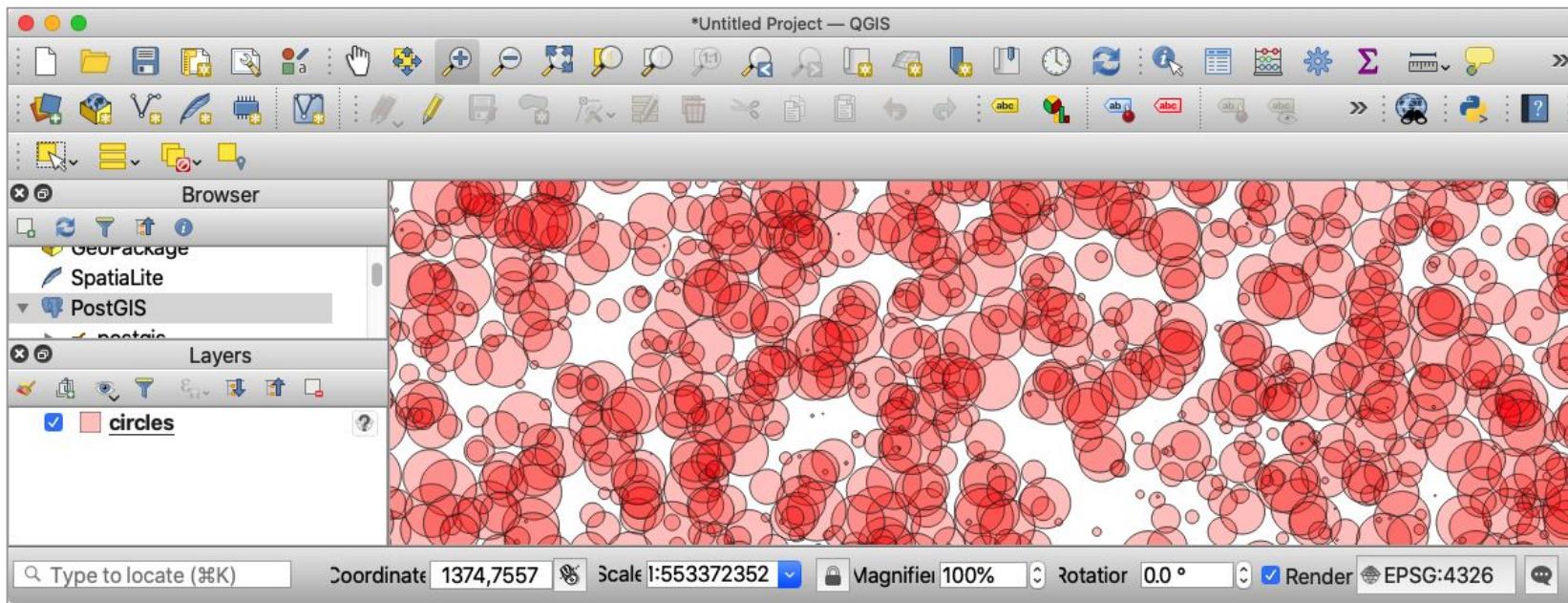
- <http://s3.cleverelephant.ca/postgis-workshop-2020.zip>

## Open workshop materials online

- <http://postgis.net/workshops/postgis-intro>

# Download QGIS?

- <https://qgis.org/>



# Ready?

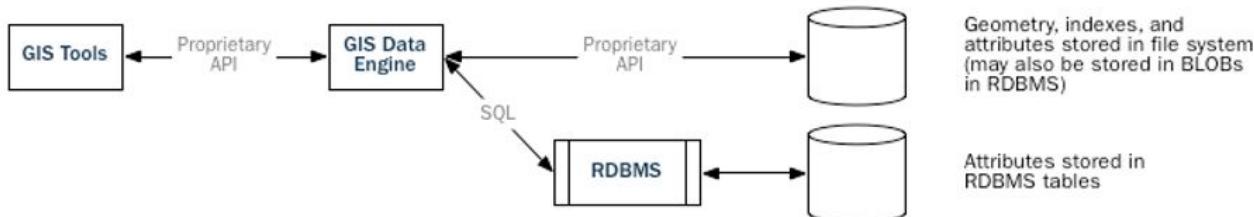
# Section 2 - Introduction

## Evolution of GIS Architectures

### First-Generation GIS:



### Second-Generation GIS:



### Third-Generation GIS:



# What is a database?

System for storage and random access of relationally (tables of rows and columns) structured data, providing the following capabilities for that data.

- **Data Types**
  - number, date, and string
- **Indexes**
  - b-tree, hash
- **Functions**
  - `strlen(string)`, `pow(float, float)`, `now()`

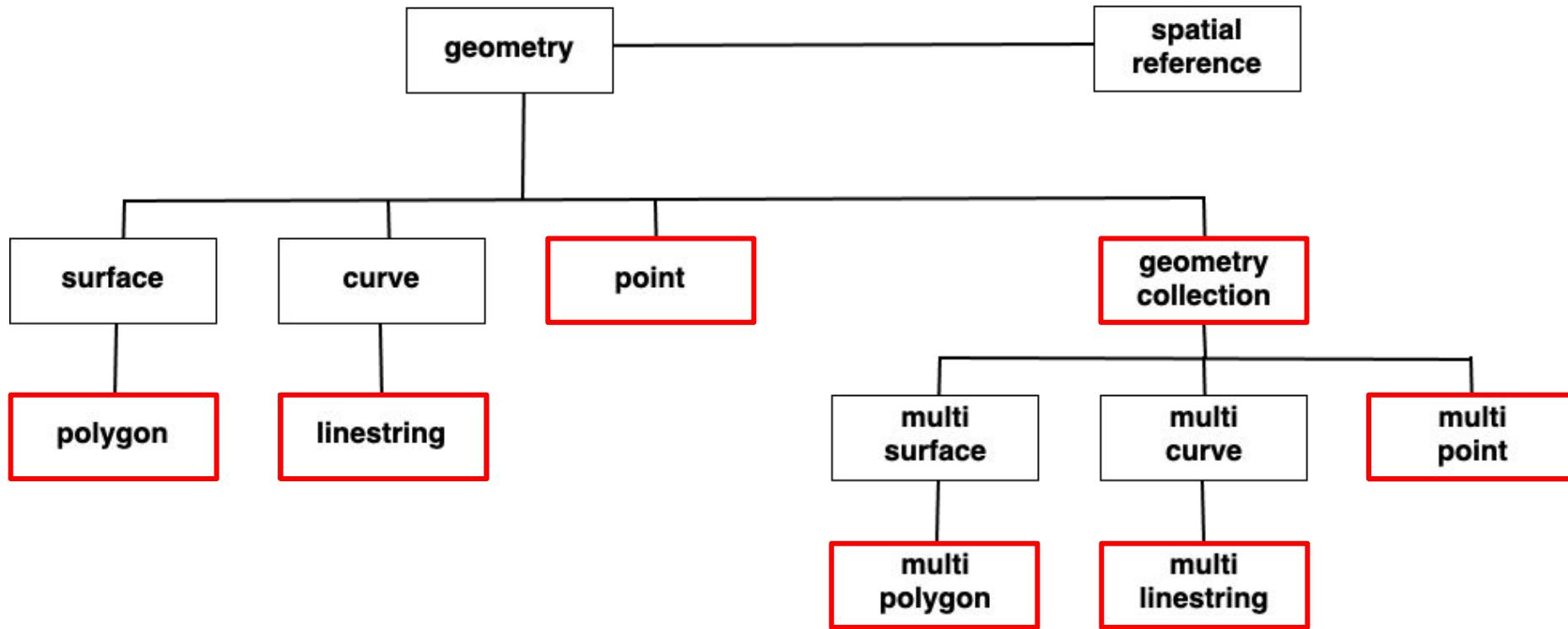
# What is a **spatial database**?

System for storage and random access of relationally (tables of rows and columns) structured data, providing the following capabilities for that data.

- **Data Types** including **Spatial Types**
  - number, date, string, **geometry**, **geography** and **raster**
- **Indexes** including **Spatial Indexes**
  - b-tree, hash, **rtree**, quadtree
- **Functions** including **Spatial Functions**
  - `strlen(string)`, `pow(float, float)`, `now()`, **ST\_Area()**, **ST\_Distance()**

**Spatial databases  
store and manipulate  
spatial objects  
like *any other object*  
in the database.**

# Spatial Types



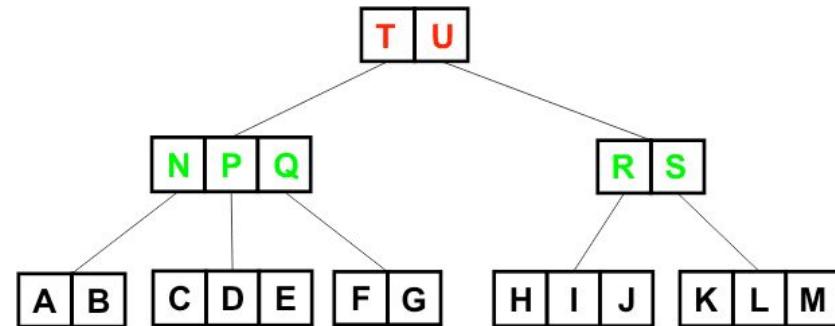
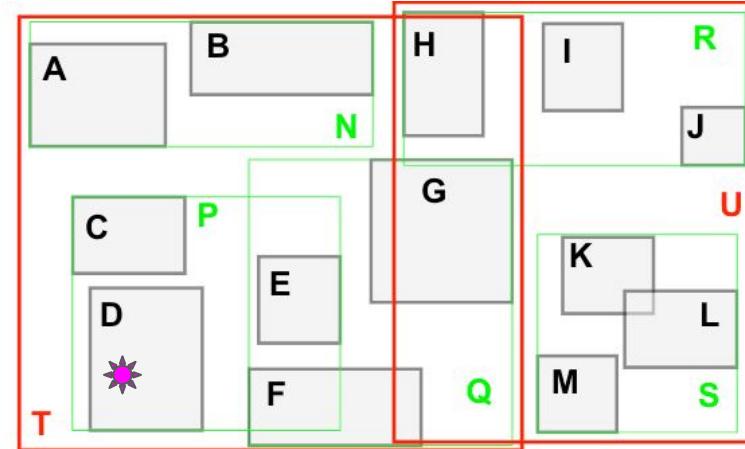
## Spatial Indexes

This R-Tree organizes the spatial objects so that a spatial search is a quick walk through the tree.

To find what object contains  ?

- The system first checks if it is in **T** or **U** (**T**)
- Then it checks if it is in **N**, **P** or **Q** (**P**)
- Then it checks if it is in **C**, **D** or **E** (**D**)

Only 8 boxes have to be tested. A full table scan would require *all 13 boxes* to be tested. The larger the table, the *more powerful* the index is.

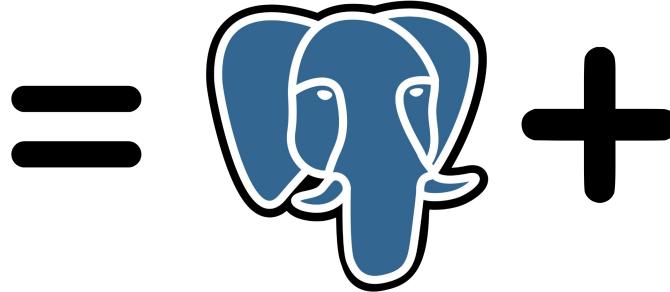


# Spatial Functions

For example:

- `ST_GeometryType(geometry) → text`
- `ST_Area(geometry) → float`
- `ST_Distance(geometry, geometry) → float`
- `ST_Buffer(geometry, radius) → geometry`
- `ST_Intersection(geometry, geometry) → geometry`
- `ST_Union([geometry]) → geometry`

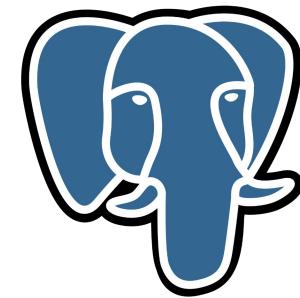
# What is PostGIS?



**CREATE  
EXTENSION  
postgis;**

# What is PostgreSQL?

- Enterprise RDBMS
- Functionally equivalent to Oracle / MSSQL
- Multi-vendor open source community
- Multi-platform support, and available on all clouds
- Highly extensible by design
  - Types, functions, indexes, replication slots, foreign data, core hooks
  - What makes PostGIS possible



# Why not files?

- Shape, FGDB, GeoPackage?
- No lingua franca for file access, every format has its own library
  - Database has SQL
  - Databases have JDBC, ODBC, others
- Multi-user access to files results in either
  - Global file locking and performance issues (best case)
  - File corruption and data loss (worst case)
- All database-style queries have to be implemented on client
  - Joins, aggregations, set-based logic



# PostGIS history



2000

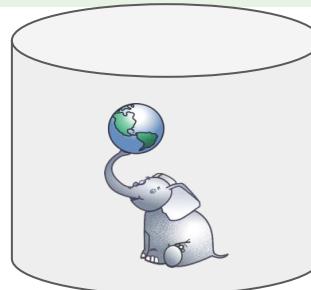
**BC Watershed  
Atlas v1**



“we need a spatial  
type!”

2001

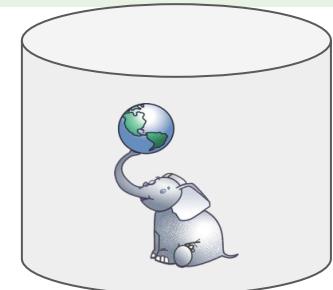
**BC Digital  
Road Atlas**



distance()  
indexes (v0.1)

2004

**BC Corporate  
Watersheds**



“lightweight”  
geometry (v1.0)



## PostGIS reference users - government



Ressources naturelles  
Canada



Ordnance Survey



## PostGIS reference users - private sector



Ball Aerospace  
& Technologies Corp.



# PostGIS 3rd party integration

## Desktop



AutoCAD  
Map 3D

*and more...*

## Middle



mapnik



*and more...*

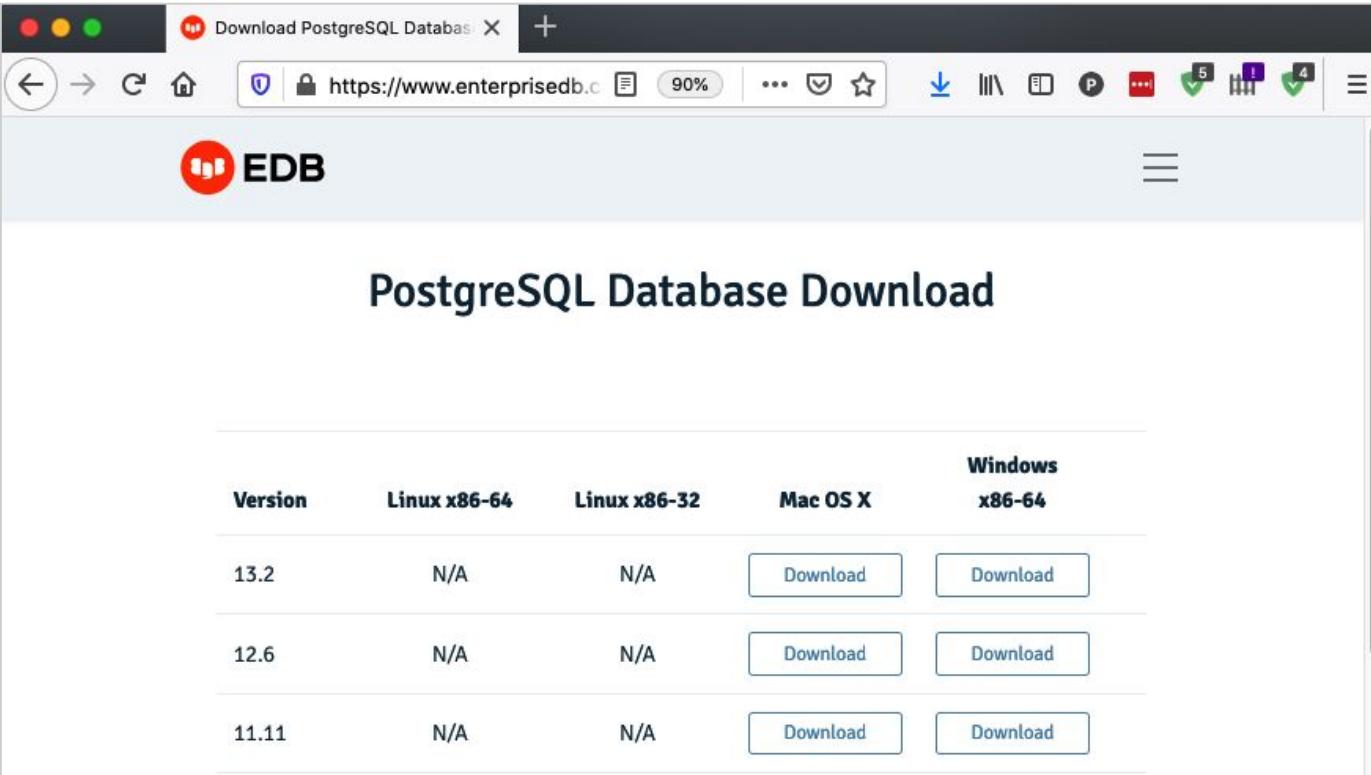
## Language



*and more...*

# Section 3 - Installation

## Microsoft Windows - [postgresql.org/download/windows/](https://postgresql.org/download/windows/)

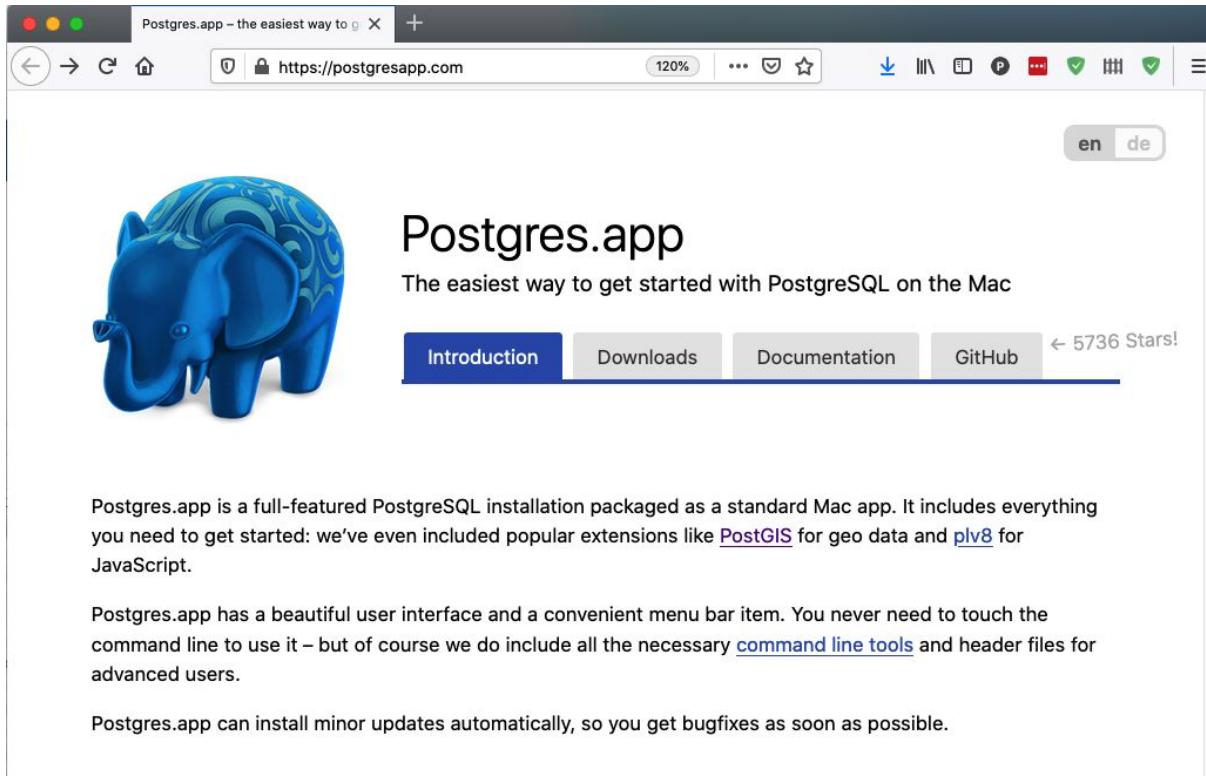


The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** "Download PostgreSQL Database" X +
- Address Bar:** https://www.enterprisedb.com
- Content Area:**
  - Logo:** EDB
  - Section Header:** PostgreSQL Database Download
  - Table:** A grid showing download links for PostgreSQL versions 13.2, 12.6, and 11.11 across platforms.

Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64
13.2	N/A	N/A	<a href="#">Download</a>	<a href="#">Download</a>
12.6	N/A	N/A	<a href="#">Download</a>	<a href="#">Download</a>
11.11	N/A	N/A	<a href="#">Download</a>	<a href="#">Download</a>

# Apple MacOS - postgresapp.com



The screenshot shows a Mac OS X browser window displaying the Postgres.app website at <https://postgresapp.com>. The page features a large blue elephant icon on the left, the text "Postgres.app" and "The easiest way to get started with PostgreSQL on the Mac" in the center, and a navigation bar with tabs for "Introduction" (which is highlighted in blue), "Downloads", "Documentation", and "GitHub". A "5736 Stars!" badge is also visible. Below the main content, there are three paragraphs of text describing the app's features and user interface.

Postgres.app is a full-featured PostgreSQL installation packaged as a standard Mac app. It includes everything you need to get started: we've even included popular extensions like [PostGIS](#) for geo data and [plv8](#) for JavaScript.

Postgres.app has a beautiful user interface and a convenient menu bar item. You never need to touch the command line to use it – but of course we do include all the necessary [command line tools](#) and header files for advanced users.

Postgres.app can install minor updates automatically, so you get bugfixes as soon as possible.

## Linux - [postgresql.org/download/](http://postgresql.org/download/)

The postgresql.org site will walk you through the many options for selecting a Linux binary package, based on the version you want and the distribution.

Make sure you install the appropriate “postgis” package!

**yum search postgis  
apt-cache search postgis**

### Packages and Installers

Select your operating system family:



Select your Linux distribution:



## Cloud - DBaaS includes PostGIS by default

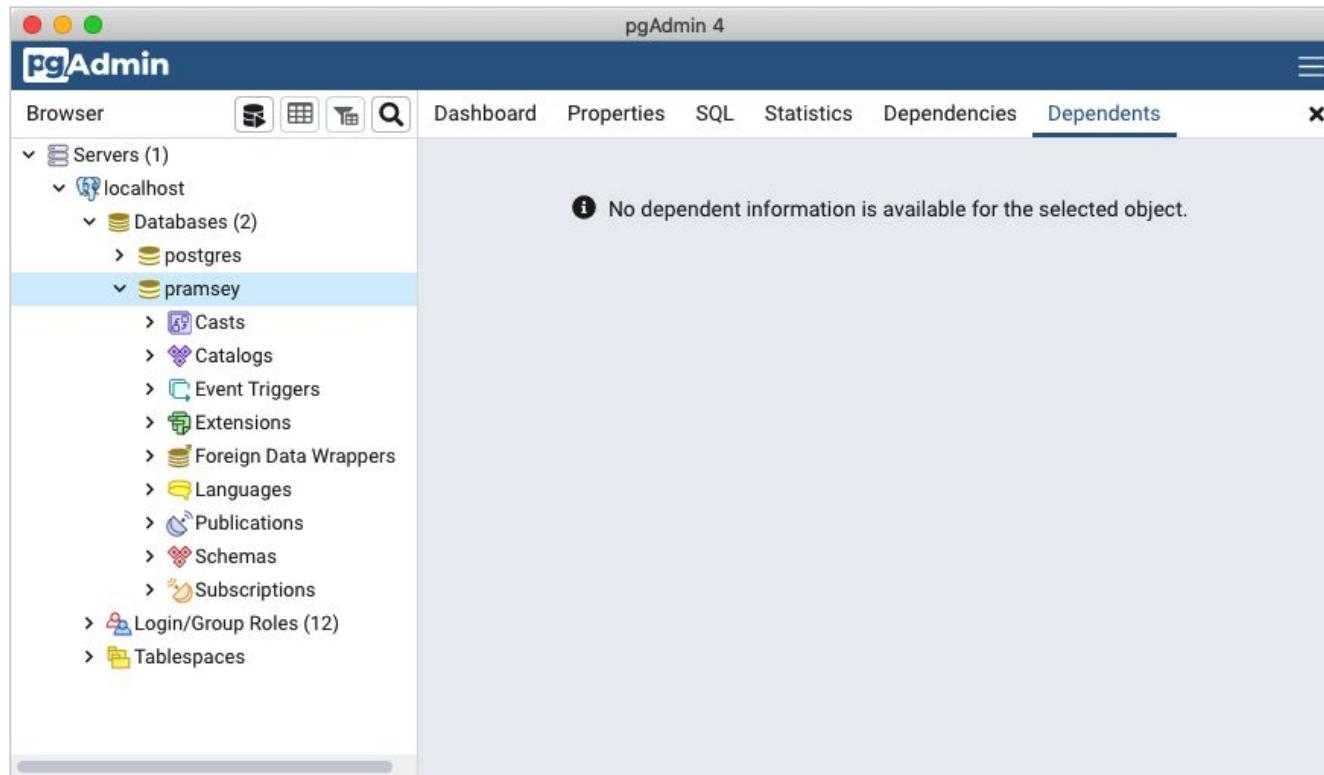


Google Cloud Platform



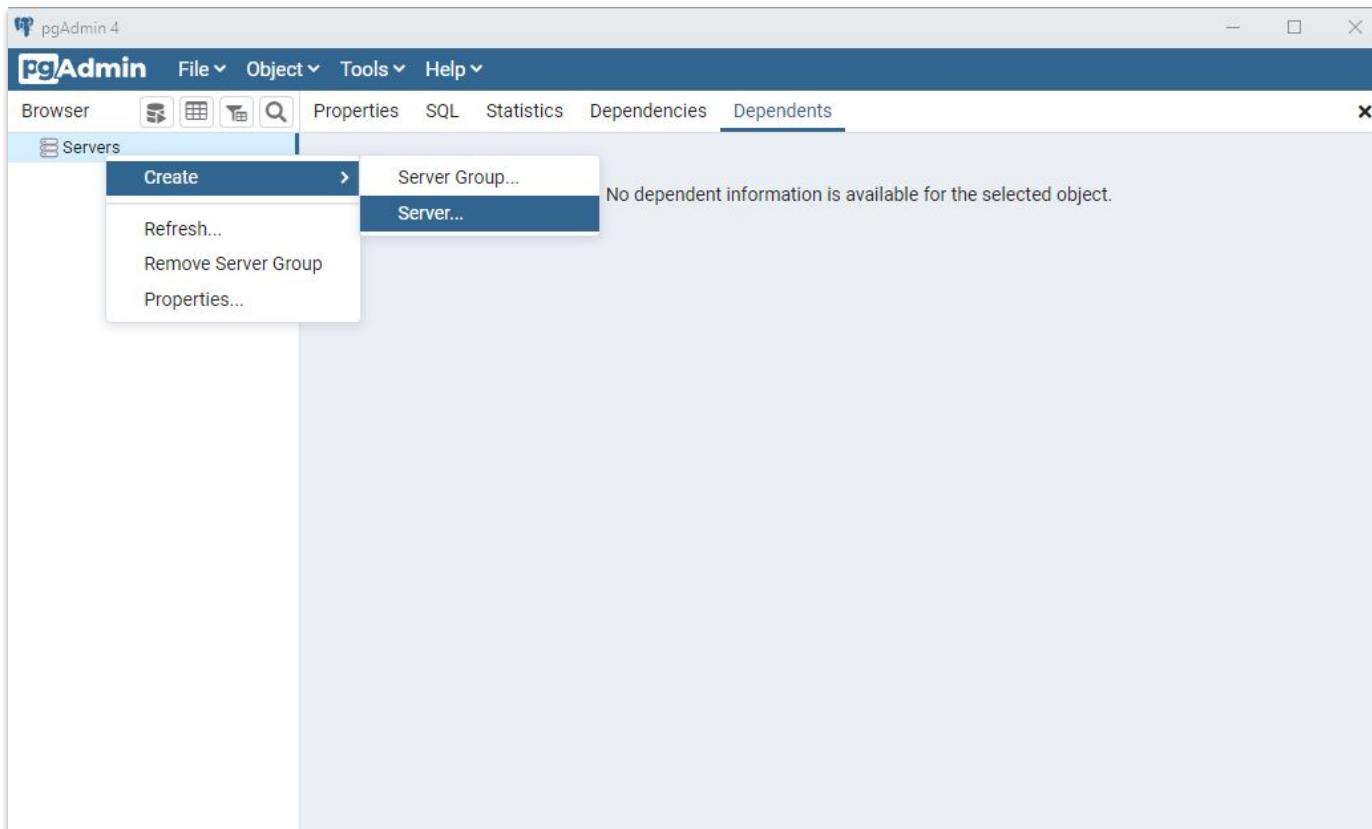
**crunchy** data

# PgAdmin - pgadmin.org



# **Section 4 - Creating a Spatial Database**

# Login



# Connection parameters

- Host name/address
- Port = **5432**
- Username = **postgres**
- Password

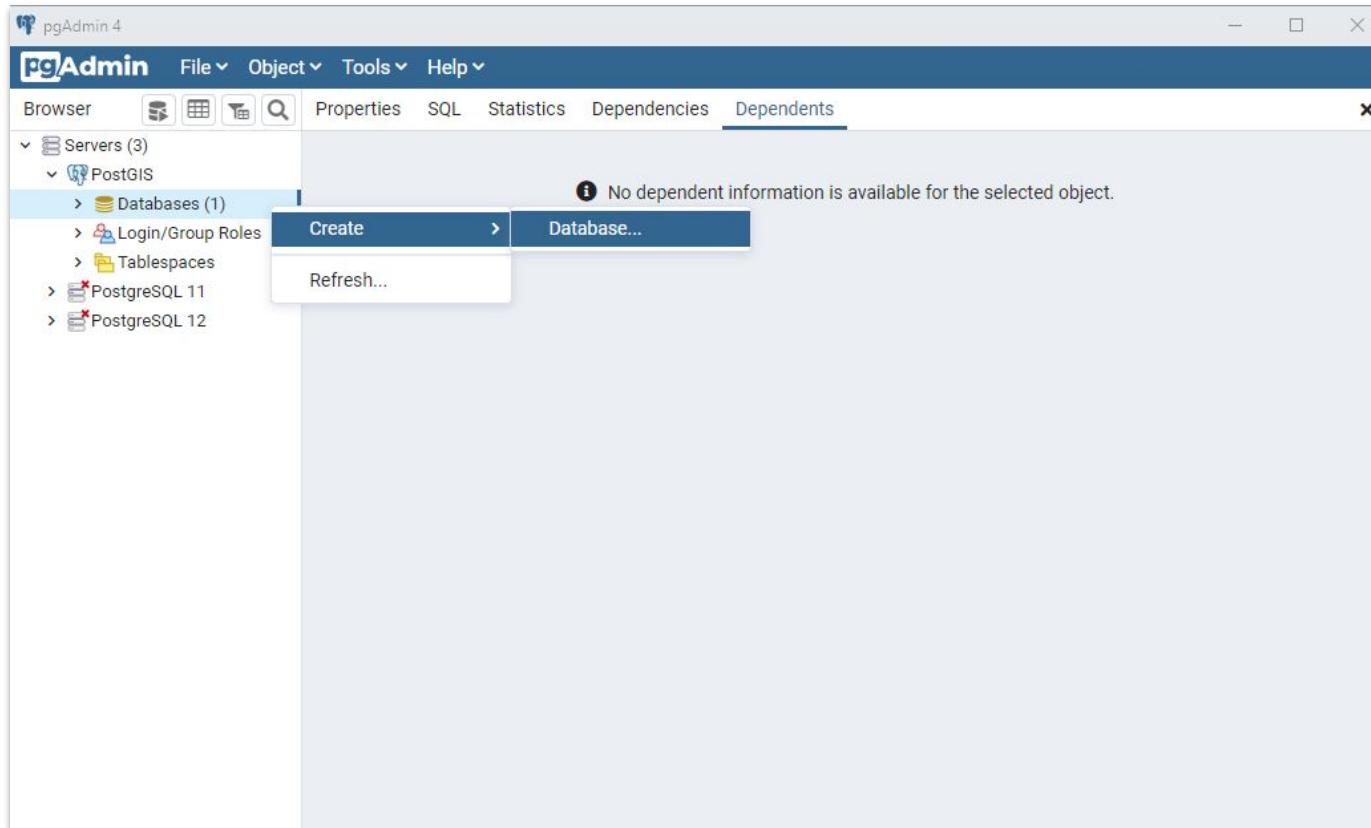
Create - Server

General Connection SSL SSH Tunnel Advanced

Host name/address	p.2gdmzr2pcbadzcstrkuolvtpq.db.postgresbridge.cor
Port	5432
Maintenance database	postgres
Username	postgres
Password	.....
Save password?	<input type="checkbox"/>
Role	
Service	

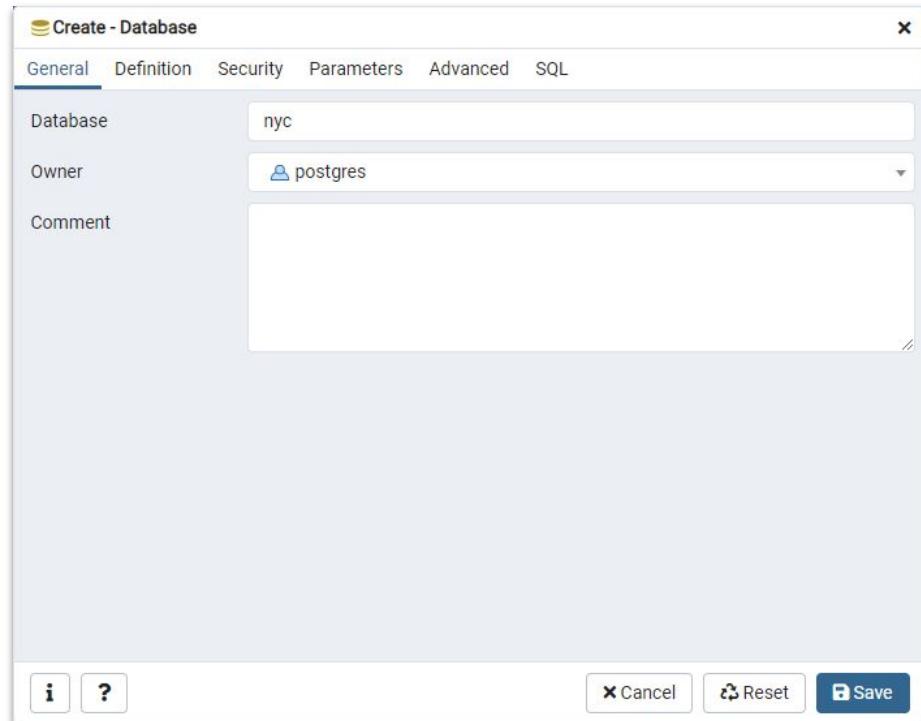
**i** **?** **Cancel** **Reset** **Save**

# New Database

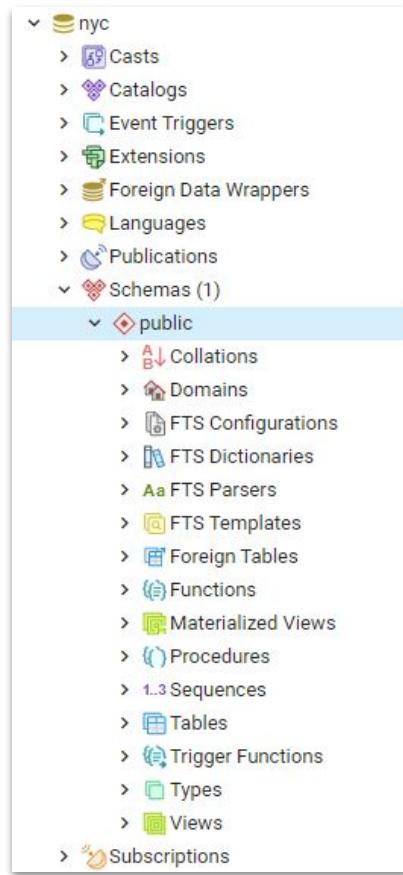


# New Database

- Database = **nyc**
- Owner = **postgres**



# Open "nyc" database



The screenshot shows a tree view of a PostgreSQL database structure. The root node is 'nyc'. Under 'nyc', there are several items: 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', and 'Schemas (1)'. The 'Schemas (1)' node is expanded, showing a single child node 'public'. The 'public' node is also expanded, listing various database objects: 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', 'Tables', 'Trigger Functions', 'Types', 'Views', and 'Subscriptions'. The 'public' schema node is highlighted with a blue selection bar.

- ▼ nyc
  - > Casts
  - > Catalogs
  - > Event Triggers
  - > Extensions
  - > Foreign Data Wrappers
  - > Languages
  - > Publications
  - > Schemas (1)
    - ▼ public
      - > Collations
      - > Domains
      - > FTS Configurations
      - > FTS Dictionaries
      - > FTS Parsers
      - > FTS Templates
      - > Foreign Tables
      - > Functions
      - > Materialized Views
      - > Procedures
      - > Sequences
      - > Tables
      - > Trigger Functions
      - > Types
      - > Views
    - > Subscriptions

## Run SQL Query

Tools > Query Tool



```
CREATE EXTENSION postgis;
```



## Run SQL Query

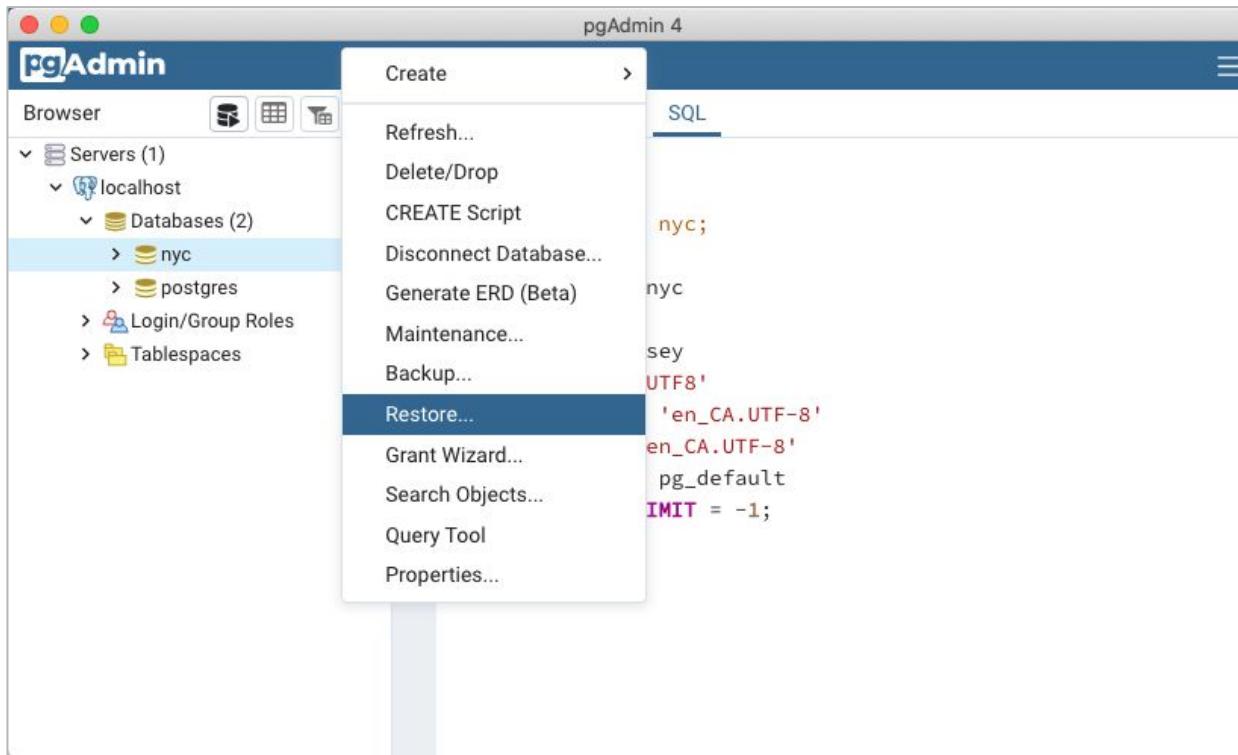


```
SELECT postgis_full_version();
```



# **Section 5 - Loading Spatial Data**

# postgis-workshop/data/nyc\_data.backup





pgAdmin 4

Select file

/tmp/postgis-workshop/data/nyc\_data.backup

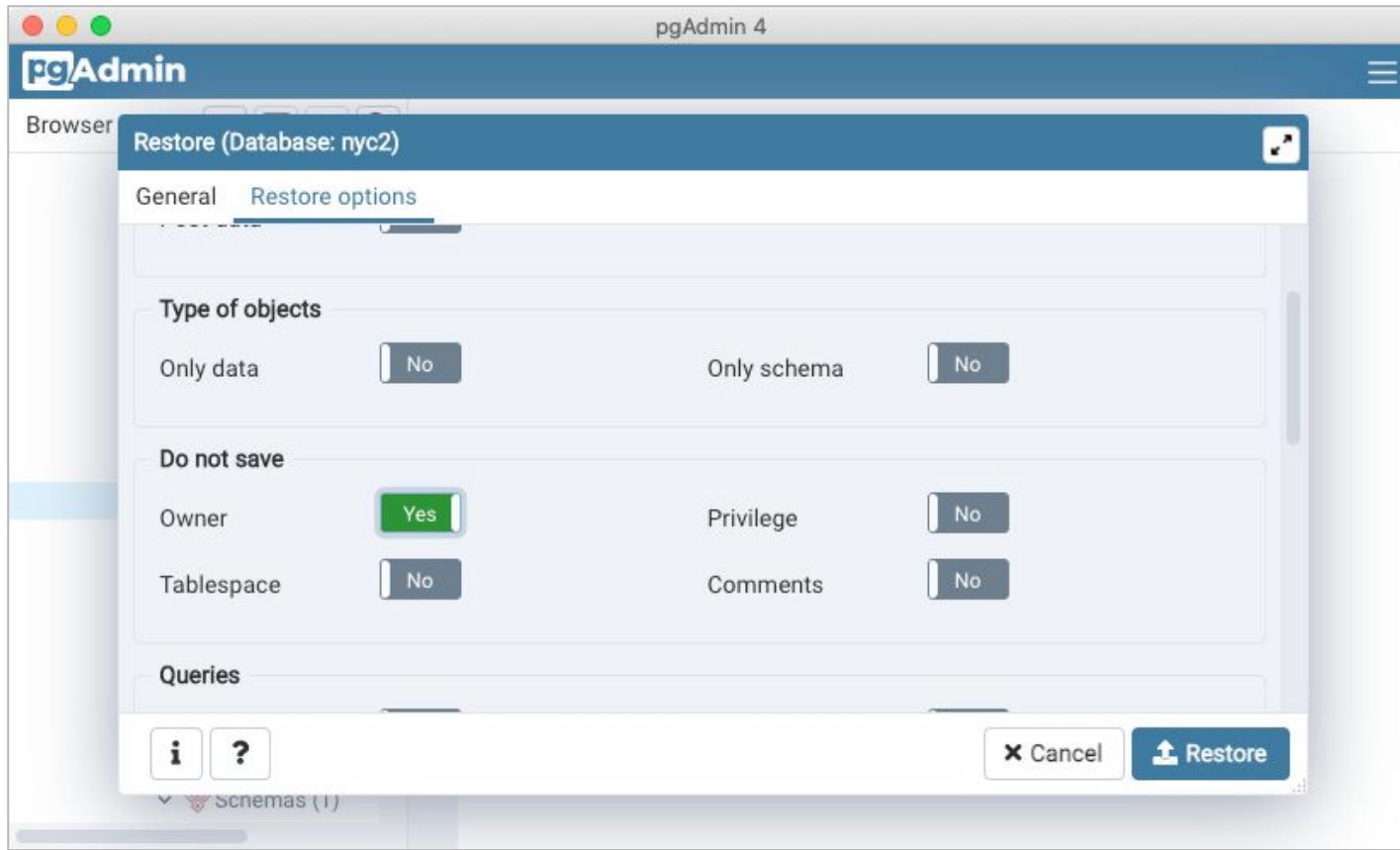
Name	Size	Modified
2000	224.0 B	Tue Sep 8 06:46:03 2015
nyc_data.backup	9.1 MB	Thu Jun 10 10:28:13 2021

Show hidden files and folders?

Format

> Extensions

The screenshot shows the pgAdmin 4 interface with a file selection dialog open. The dialog title is "Select file" and the path is "/tmp/postgis-workshop/data/nyc\_data.backup". Inside the dialog, there is a table with three columns: Name, Size, and Modified. The table contains two rows: "2000" (224.0 B, modified Tue Sep 8 06:46:03 2015) and "nyc\_data.backup" (9.1 MB, modified Thu Jun 10 10:28:13 2021). The "nyc\_data.backup" file is highlighted. Below the table, there is a checkbox labeled "Show hidden files and folders?" and a "Format" dropdown set to "backup". At the bottom of the dialog are "Cancel" and "Select" buttons. The pgAdmin 4 interface has a dark blue header and sidebar, with a light gray main area. The sidebar on the left shows a tree view with "Extensions" selected. The overall theme is professional and user-friendly.



pgAdmin 4

pgAdmin

Browser       

Servers (1)  
localhost  
Databases (2)  
nyc  
postgres  
Login/Group Roles  
Tablespaces

SQL

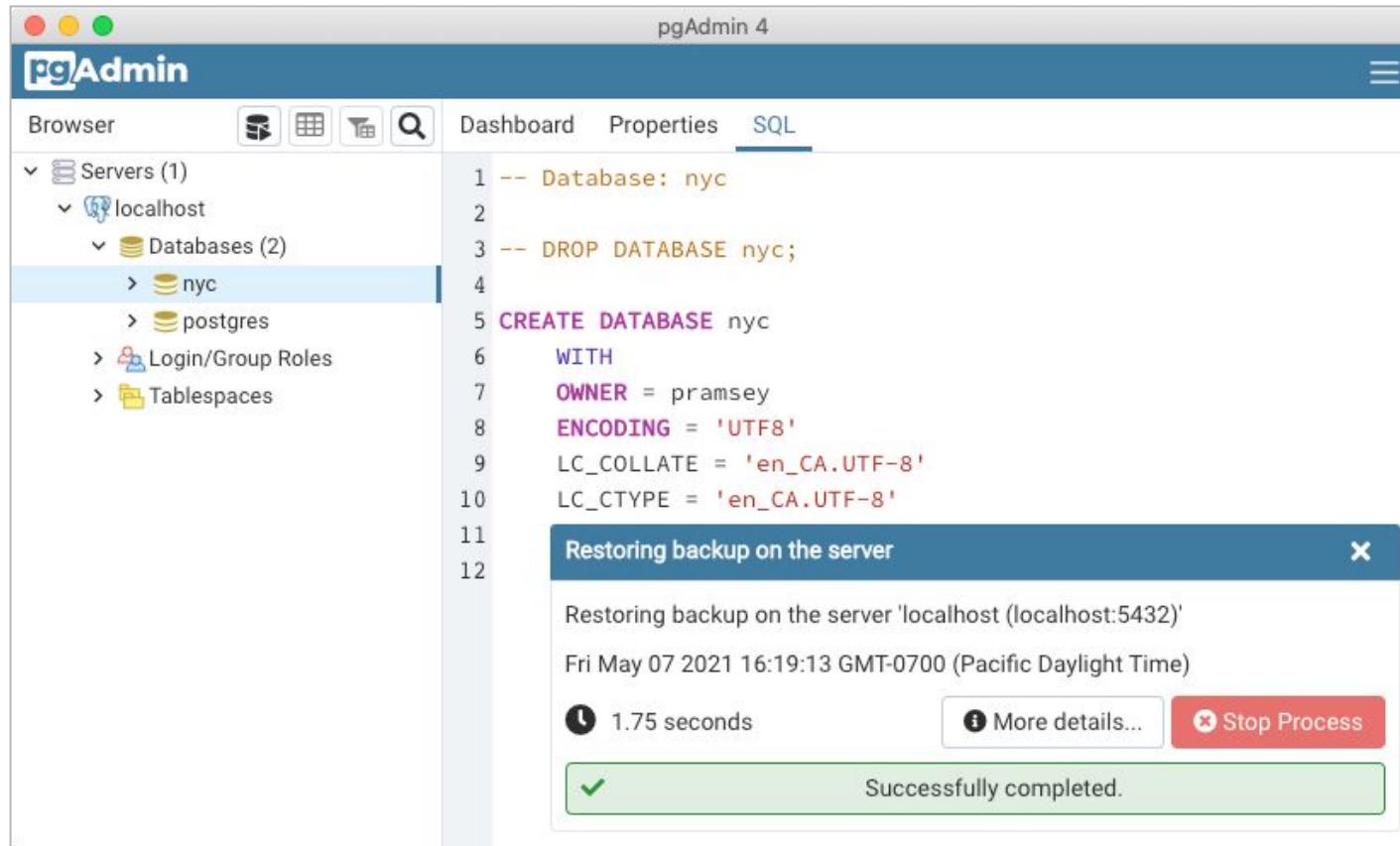
```
1 -- Database: nyc
2
3 -- DROP DATABASE nyc;
4
5 CREATE DATABASE nyc
6   WITH
7     OWNER = pramsey
8     ENCODING = 'UTF8'
9     LC_COLLATE = 'en_CA.UTF-8'
10    LC_CTYPE = 'en_CA.UTF-8'
```

Restoring backup on the server X

Restoring backup on the server 'localhost (localhost:5432)'  
Fri May 07 2021 16:19:13 GMT-0700 (Pacific Daylight Time)

1.75 seconds More details... Stop Process

✓ Successfully completed.



# Loading “GIS data files”

## ogr2ogr

- supports many formats
  - shp, fgdb, geojson, etc
- supports many databases
  - sqlserver, oracle, mysql
- many processing options
- separate install
- complex command line
- connects directly to database

## shp2pgsql

- supports only shape file
- supports only pgsql
- minimal processing options
- comes with postgis
- simpler command line
- generates SQL file to pass into the database

## ogr2ogr

```
ogr2ogr \
-nln nyc_census_blocks_2000 \
-nlt PROMOTE_TO_MULTI \
-lco GEOMETRY_NAME=geom \
-lco FID=gid \
-lco PRECISION=NO \
Pg:'dbname=nyc host=localhost user=postgres port=5432' \
nyc_census_blocks_2000.shp
```

new layer name

new layer type

geometry column  
name

pk name

full precision?

source data

destination data

# What is a “shape file”?

- **nyc\_streets.shp**
  - The actual “shapes”, in a binary format
- **nyc\_streets.dbf**
  - The “attributes” of the streets, a spreadsheet of one row per shape with the non-spatial information (name, street number, city, etc)
- **nyc\_streets.shx**
  - An offset index. (Shape 5 begins at byte 1023 in the shp file, etc)
- **nyc\_streets.prj**
  - The spatial reference system of the data in WKT format.

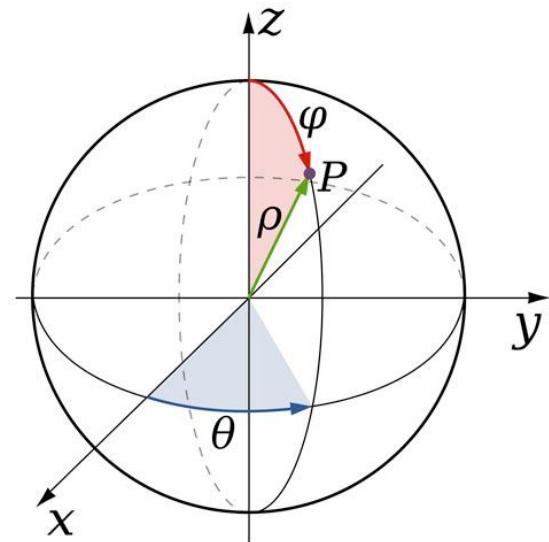
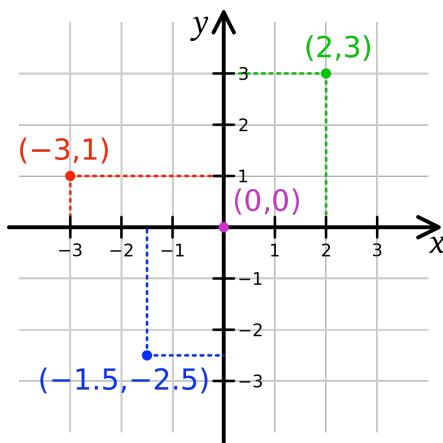
## nyc\_streets.prj (EPSG:26918)

```
PROJCS[ "NAD_1983_UTM_Zone_18N",
  GEOGCS[ "GCS_North_American_1983",
    DATUM[ "D_North_American_1983",
      SPHEROID[ "GRS_1980", 6378137, 298.257222101]],
    PRIMEM[ "Greenwich", 0],
    UNIT[ "Degree", 0.017453292519943295]],
  PROJECTION[ "Transverse_Mercator"],
  PARAMETER[ "latitude_of_origin", 0],
  PARAMETER[ "central_meridian", -75],
  PARAMETER[ "scale_factor", 0.9996],
  PARAMETER[ "false_easting", 500000],
  PARAMETER[ "false_northing", 0],
  UNIT[ "Meter", 1]]
```

# nyc\_streets.prj (EPSG:26918)

**Universal Transverse Mercator  
EPSG:26918**

POINT(586020 4513147)



**WGS 1984  
EPSG:4326**

POINT(-73.9808 40.7648)

# shp2pgsql

```
shp2pgsql \
-D \
-I \
-s 26918 \
nyc_census_blocks_2000.shp \
nyc_census_blocks_2000 \
| psql dbname=nyc user=postgres host=localhost
```

dump format

build spatial  
index

spatial reference  
id of data

source file

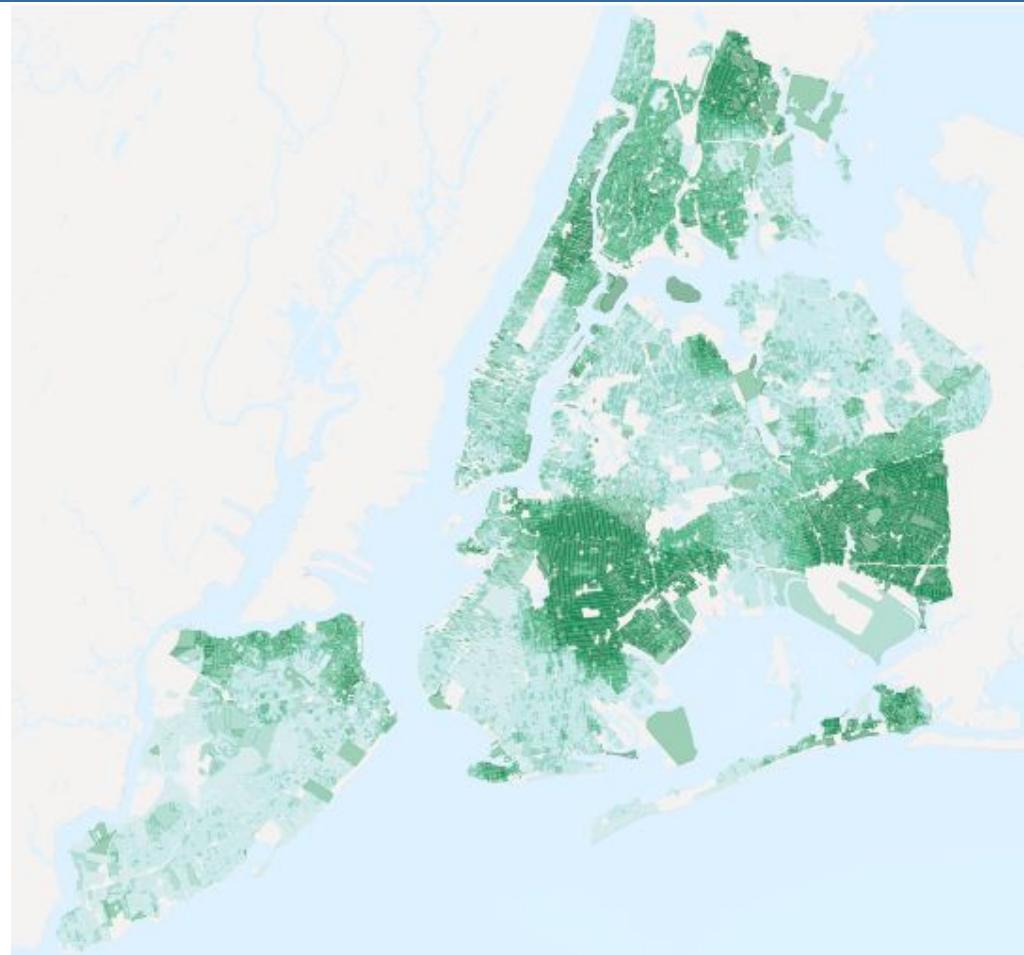
destination table  
name

destination database  
connection

# **Section 6 - About our data**

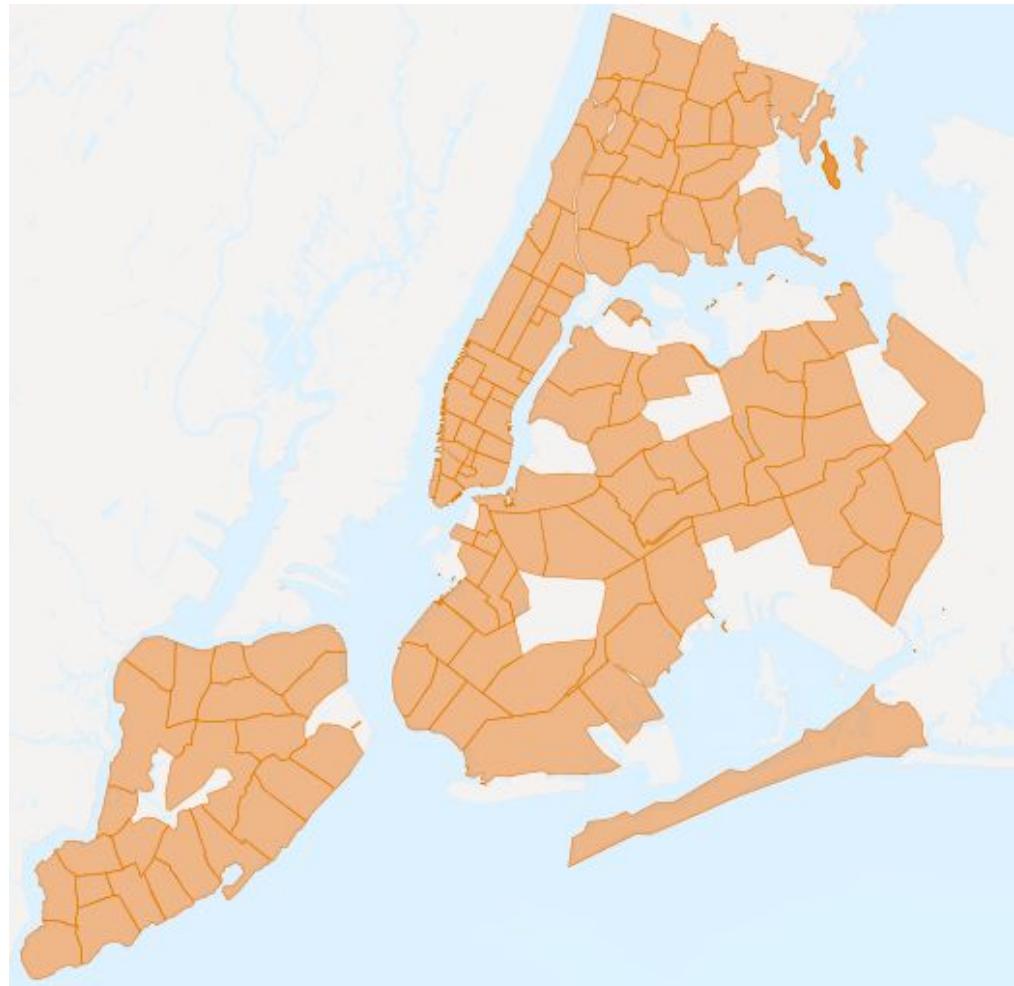
### **nyc\_census\_blocks**

```
blkid  
popn_total  
popn_white  
popn_black  
popn_nativ  
popn_asian  
popn_other  
boroname  
geom
```



## nyc\_neighborhoods

**name**  
**boroname**  
**geom**



# nyc\_streets

**name**  
**oneway**  
**type**  
**geom**



# nyc\_subway\_stations

```
name  
borough  
routes  
transfers  
express  
geom
```



## nyc\_census\_sociodata

tractid	transit_total	transit_public
transit_private	transit_other	transit_time_mins
family_county	family_income_median	family_income_aggregate
edu_total	edu_no_highschool_dipl	edu_highscool_dipl
edu_college_dipl	edu_graduate_dipl	

# **Section 7 - Simple SQL**



```
SELECT name  
FROM nyc_neighborhoods;
```



## PgAdmin and Maps

PgAdmin includes a magic “map” button which is fun to use to visualize geometric outputs. If the data are in “geographics” (EPSG:4326) then it will place a base map underneath the data. This can be pretty.

To see a map with your output, wrap it in **ST\_Transform(geom, 4326)**. This will make more sense as we work through the materials.

```
SELECT name,  
       ST_Transform(geom, 4326)  
FROM nyc_neighborhoods;
```

	name character varying (64)	st_transform geometry	
1	Bensonhurst	0106000020E6100000...	
2	East Village	0106000020E6100000...	
3	West Village	0106000020E6100000...	
4	Throggs Neck	0106000020E6100000...	
5	Wakefield-Williamsbridge	0106000020E6100000...	
6	Auburndale	0106000020E6100000...	
7	Battery Park	0106000020E6100000...	

pgAdmin 4

Browser        Dashboard Properties SQL Statistics Dependencies Dependents [nyc/pramsey@localhost](#) < > x

Servers (1)  
localhost  
Databases (4)  
nyc  
Casts  
Catalogs  
Event Triggers  
Extensions  
Foreign Data Wrappers  
Languages  
Publications  
Schemas  
Subscriptions  
postgis32  
postgres  
pramsey  
Login/Group Roles  
Tablespaces

Query Editor    Query History    Scratch Pad x

No limit x

Query Editor:

```
1 SELECT name,
2      ST_Transform(geom, 4326)
3 FROM nyc_neighborhoods;
```

Data Output   Explain   Messages   Notifications   Geometry Viewer x

Geometry Viewer: A map of New York City showing neighborhood boundaries. The map includes labels for various towns and cities like Hopatcong, Montville, Paterson, Yonkers, Bayville, Huntington, Smithtown, Islip, and many others. A large blue polygon highlights a specific area in Manhattan, likely representing the geographic scope of the query results.

 OpenStreetMap

## Four Verbs of SQL

- **SELECT**
  - returns rows in response to a query
- **INSERT**
  - adds new rows to a table
- **UPDATE**
  - alters existing rows in a table
- **DELETE**
  - removes rows from a table

**What are ...  
the names of all the neighborhoods in Brooklyn?**

```
SELECT name  
FROM nyc_neighborhoods  
WHERE boroname = 'Brooklyn';
```

**What is the number of letters in the names of all the neighborhoods in Brooklyn?**

```
SELECT char_length(name)
FROM nyc_neighborhoods
WHERE boroname = 'Brooklyn';
```

**What is the average number of letters and standard deviation of number of letters in the names of all the neighborhoods in Brooklyn?**

```
SELECT avg(char_length(name)),  
       stddev(char_length(name))  
  FROM nyc_neighborhoods  
 WHERE boroname = 'Brooklyn';
```

**What is the average number of letters in the names of all the neighborhoods in New York City, reported by borough?**

```
SELECT boroname,  
       avg(char_length(name))  
FROM nyc_neighborhoods  
GROUP BY boroname;
```

# Section 8 - Simple SQL Exercises

GEOMETRY FUNCTIONS	
ST_Area(geometry)	ST_SRID(geometry)
ST_AsBinary(geometry)	ST_StartPoint(linestring)
ST_AsGML(geometry)	ST_SymDifference(geometry,geometry)
ST_AsGeoJSON(geometry)	ST_Touches(geometry, geometry)
ST_AsKML(geometry)	ST_Union(geometry)
ST_AsSVG(geometry)	ST_Within(geometry,geometry)
ST_AsText(geometry)	ST_X(point)
ST_Buffer(geometry,distance)	ST_Y(point)
ST_Centroid(geometry)	ST_ZipCode()
ST_ConvexHull(geometry)	ST_M(point)
ST_Crossed(geometry,geometry)	
ST_Difference(geometry,geometry)	
ST_Disjoint(geometry,geometry)	
ST_Distanced(geometry,geometry)	
ST_EndPoint(geometry)	
ST_Equal(geometry,geometry)	
ST_ExteriorRing(geometry)	
ST_GeomFromGML(text)	
ST_GeomFromKML(text)	
ST_GeomFromText(text)	
ST_GeomFromWKB(bytea)	
ST_GeometryN(collection,n)	
ST_GeometryType(geometry)	
ST_InteriorRingN(polygon)	
ST_Intersection(geometry,geometry)	
ST_Intersection(geometry,geometry)	
ST_IsValid(geometry)	
ST_IsValidReason(geometry)	
ST_Length(geometries)	
ST_MakePoint(double,double)	
ST_NPoints(geometry)	
ST_NumGeometries(collection)	
ST_OrderingEqual(geometry,geometry)	
ST_Overlaps(geometry,geometry)	
ST_Perimeter(geometry)	
ST_PointsOnSurface(geometry)	
ST_Relate(geometry,geometry)	
ST_Reverse(geometry)	
ST_Simplify(geometry)	

GEOGRAPHY FUNCTIONS	
ST_Area(geography)	ST_AsBinary(geography)
ST_AsGML(geography)	ST_AsGeoJSON(geography)
ST_AsSVG(geography)	ST_AsText(geography)
ST_Buffer(geography,distance)	ST_CoveredBy(geography,geography)
ST_CoveredBy(geography,geography)	ST_Covers(geography,geography)
ST_DWithin(geography,geography,float8)	ST_DWithin(geography,geography,float8)
ST_Distance(geography,geography)	ST_Disjoint(geography,geography)
ST_GeogFromWKID(bytea)	ST_Equals(geography,geography)
ST_GeographyFromText(text)	ST_Intersects(geography,geography)
ST_Intersects(geography,geography)	ST_IsValid(geography)
ST_Length(geography)	ST_IsValidReason(geography)
ST_MakePoint(double,double)	
ST_NPoints(geography)	
ST_NumGeometries(collection)	
ST_OrderingEqual(geometry,geometry)	
ST_Overlaps(geography,geography)	
ST_Perimeter(geometry)	
ST_PointsOnSurface(geometry)	
ST_Relate(geometry,geometry)	
ST_Reverse(geometry)	
ST_Simplify(geometry)	

nyc_census_blocks (36592 records)	
blkid	A 15-digit code that uniquely identifies every census block. Eg: 360050001090900
popn_total	Total population 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_native	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
house_total	Number of housing units in the block
house_own	Number of owner-occupied housing units in the block
house_rent	Number of renter-occupied housing units in the block
boroughname	Name of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the block

nyc_neighborhoods (129 records)	
name	Name of the neighborhood
boroughname	Name of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the neighborhood

nyc_streets (19091 records)	
name	Name of the street
oneway	Is the street one-way? "yes" = yes, "no" = no
type	Road type: e.g. primary, secondary, residential, motorway
the_geom	Linear centerline of the street

nyc_subway_stations (491 records)	
name	Name of the station
borough	Name of 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" = no
the_geom	Point location of the station

nyc_census_sociodata	
tractid	An 11-digit code for every census tract. Eg: 36005000100
transit_total	Number of workers in the tract
transit_public	Number of workers in the tract who take public transit
transit_private	Number of workers in the tract who use private automobiles / motorcycles
transit_other	Number of workers in the tract who use other forms like walking / biking
transit_time_mean	Total minutes spent commuting by all workers in the tract (minutes)
family_count	Number of families in the tract
family_income_mean	Median family income in the tract (dollar)
family_income_aggregate	Total income of all families in the tract (dollar)
edu_total	Number of people with educational history
edu_no_highschool_dipl	Number of people with no highschool diploma
edu_highschool_dipl	Number of people with highschool diplomas and no further education
edu_college_dipl	Number of people with college diplomas and no further education
edu_graduate_dipl	Number of people with graduate school diplomas

**How many records are in the nyc\_streets table?**

```
SELECT Count(*)  
FROM nyc_streets;
```

## How many streets in NYC start with 'B'?

```
SELECT Count(*)  
FROM nyc_streets  
WHERE name LIKE 'B%';
```

## What is the population of NYC?

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

## What is the population of ‘The Bronx’?

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

## How many neighborhoods are in each borough?

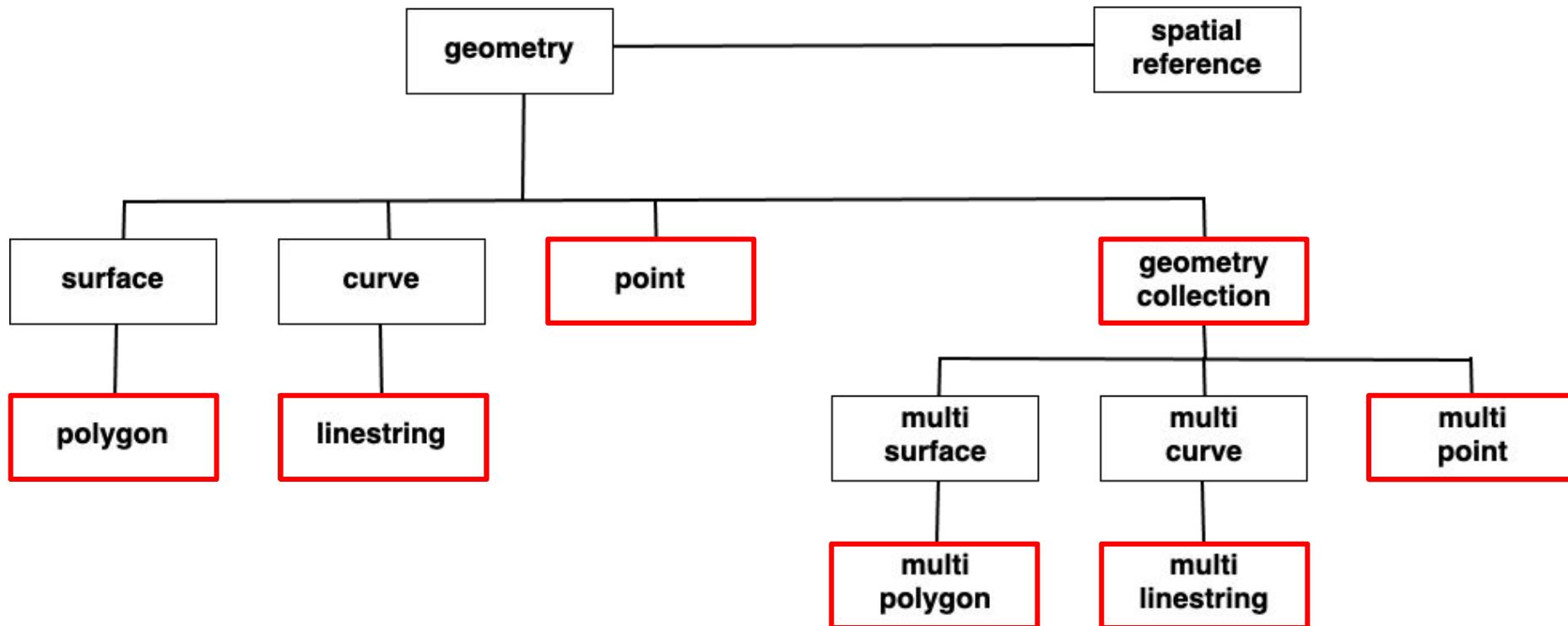
```
SELECT boroname, Count(*)  
FROM nyc_neighborhoods  
GROUP BY boroname;
```

**For each borough in NYC, what is percentage of the population is “white”?**

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

# Section 9 - Geometries

# Spatial Types



## Creating a table with geometry

```
CREATE TABLE geometries  
(  
    name varchar,  
    geom geometry  
);
```

## Creating a table with geometry

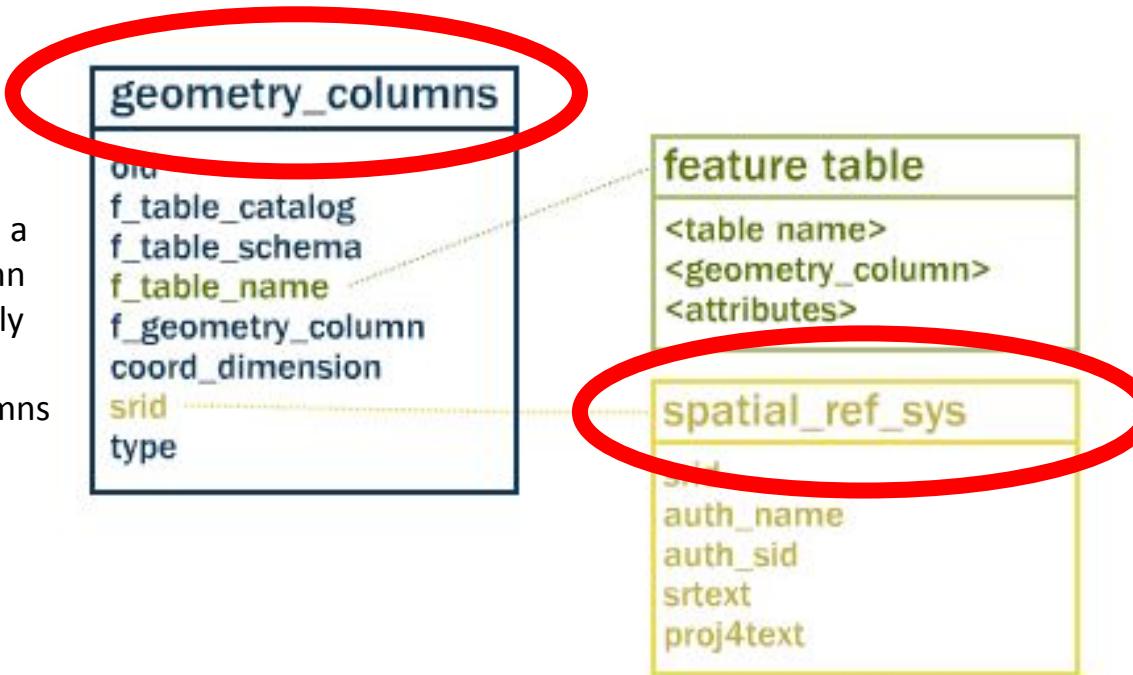
```
INSERT INTO geometries (name, geom) VALUES
('Point', 'POINT(0 0)'),
('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),
('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
('PolygonWithHole', 'POLYGON((...))'),
('Collection', 'GEOMETRYCOLLECTION(...)');
```

## Creating a table with geometry

```
SELECT name, ST_AsText(geom)  
FROM geometries;
```

# Table Relationships

Every table with a geometry column will automatically appear in the geometry\_columns view.



The SRID numbers on geometries and columns are converted into coordinate transforms using data from the spatial\_ref\_sys table.

## geometry\_columns

```
SELECT *
FROM geometry_columns
```

# geometry\_columns

nyc/postgres@localhost ▾

Query Editor **Query History** Scratch Pad

```
1 SELECT * FROM geometry_columns;
```

Data Output Explain 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	nyc	public	nyc_census_blocks	geom	2	26918	MULTIPOLYGON
2	nyc	public	nyc_homicides	geom	2	26918	POINT
3	nyc	public	nyc_neighborhoods	geom	2	26918	MULTIPOLYGON
4	nyc	public	nyc_streets	geom	2	26918	MULTILINESTRING
5	nyc	public	nyc_subway_stati...	geom	2	26918	POINT

## Metadata functions

**SELECT**

```
name,  
ST_GeometryType(geom),  
ST_NDims(geom),  
ST_SRID(geom)
```

**FROM** geometries;

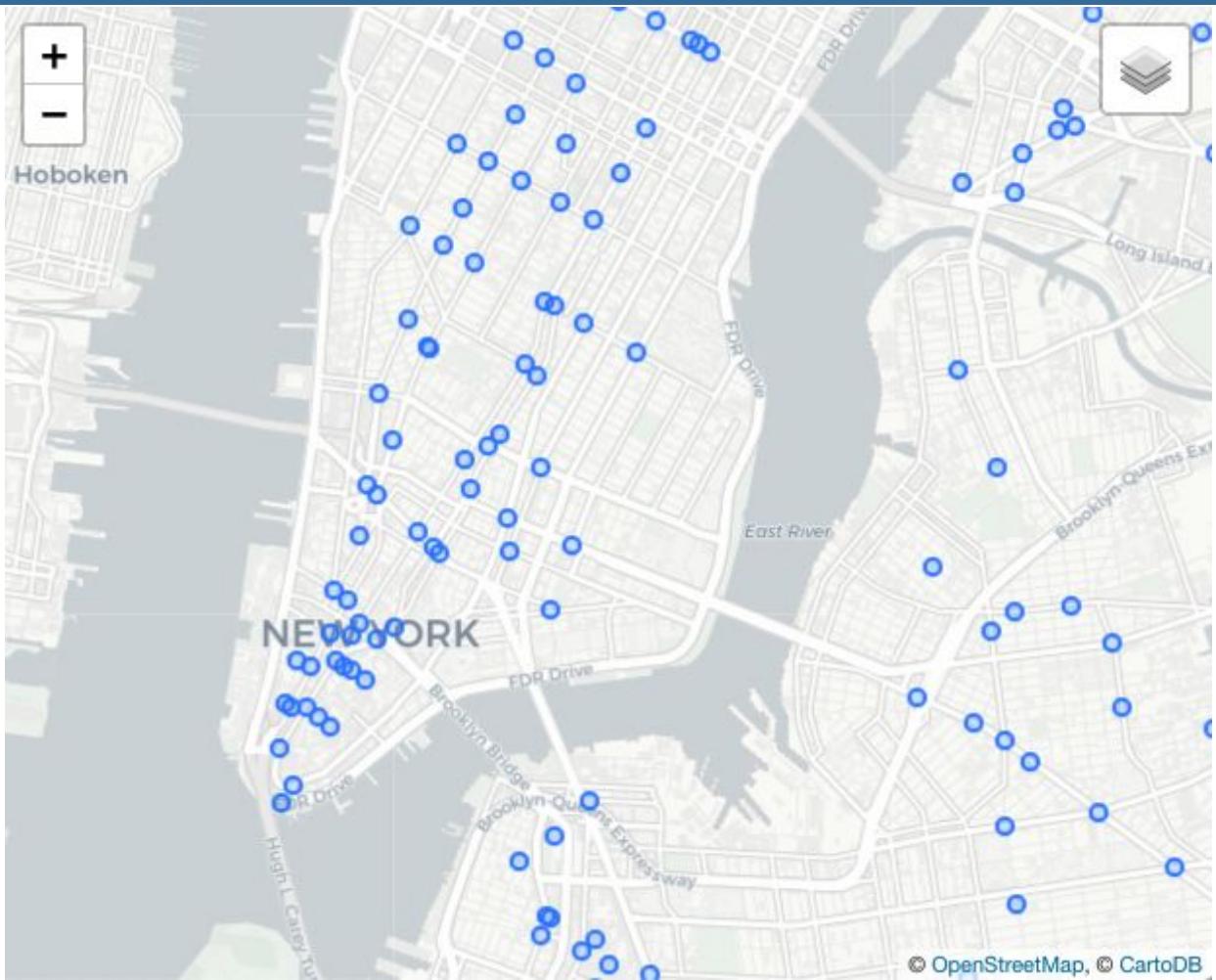
## Metadata functions

<code>name</code>	<code>st_geometrype</code>	<code>st_ndims</code>	<code>st_srid</code>
<code>Point</code>	<code>ST_Point</code>	2	0
<code>Linestring</code>	<code>ST_LineString</code>	2	0
<code>Polygon</code>	<code>ST_Polygon</code>	2	0
<code>PolygonWithHole</code>	<code>ST_Polygon</code>	2	0
<code>Collection</code>	<code>ST_GeometryCollection</code>	2	0

# Points

“Point” or “MultiPoint”, representing one or more 0-dimensional locations.

New York city subway stations, stop signs, man holes, address points, current locations of vehicles, might all use a “Point” geometry type.



## Points

```
SELECT ST_AsText(geom)  
FROM geometries  
WHERE name = 'Point';
```

---

POINT(0 0)

## Points

```
SELECT
    ST_X(geom),
    ST_Y(geom)
FROM geometries
WHERE name = 'Point'
```

---

θ θ

## Points

```
SELECT
    name,
    ST_AsText(geom)
FROM nyc_subway_stations
LIMIT 1;
```

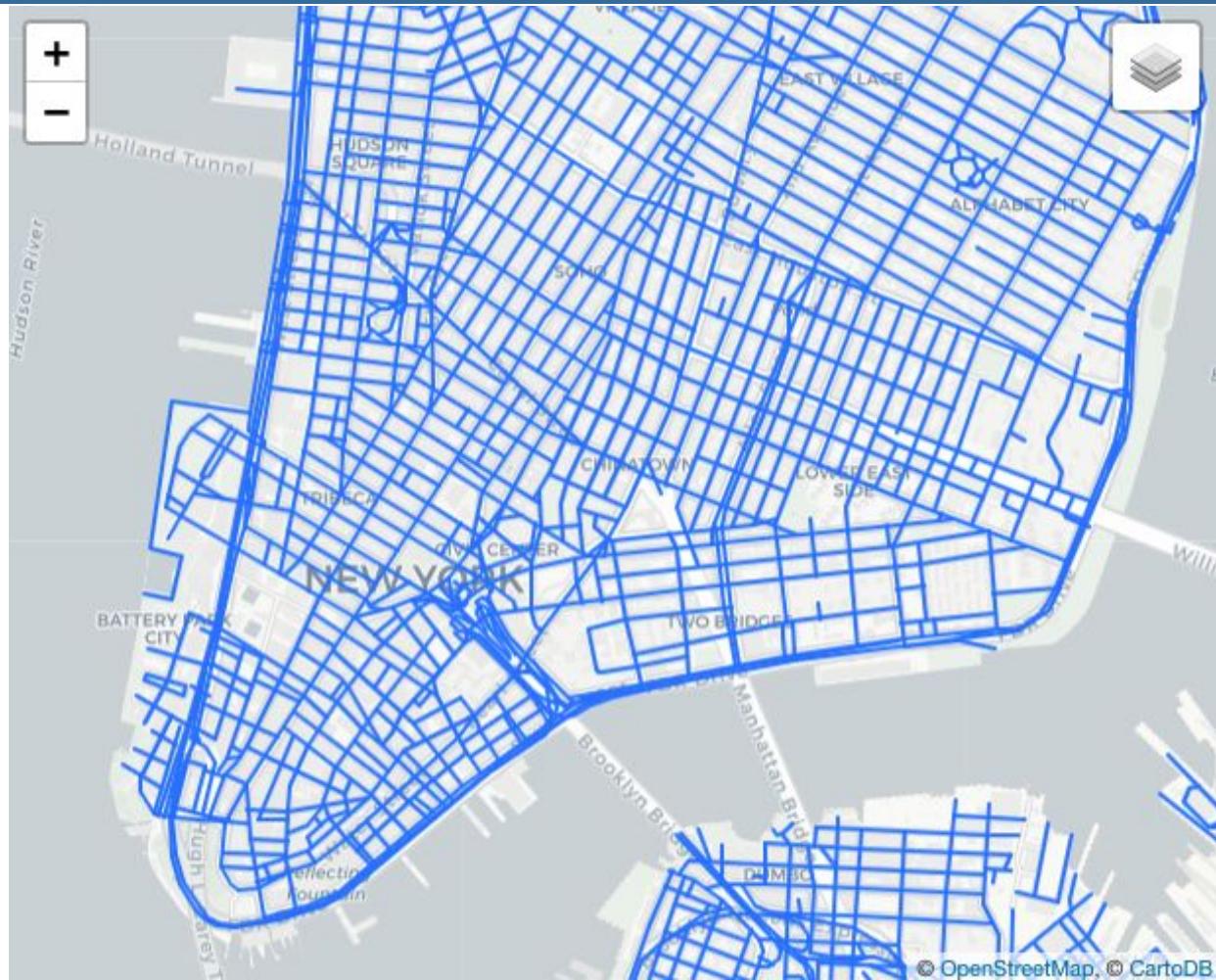
---

Cortlandt St | POINT(583521 4507077)

# LineStrings

“LineString” or  
“MultiLineString”,  
representing one or more  
1-dimensional objects.

Streets, streams, bus routes,  
power lines, driven routes,  
highways, might all use a  
“LineString” geometry type.



## LineStrings

```
SELECT ST_AsText(geom)  
FROM geometries  
WHERE name = 'Linestring';
```

---

**LINESTRING(0 0,1 1,2 1,2 2)**

## LineStrings

```
SELECT ST_Length(geom)  
FROM geometries  
WHERE name = 'Linestring';
```

---

3.41421356237309

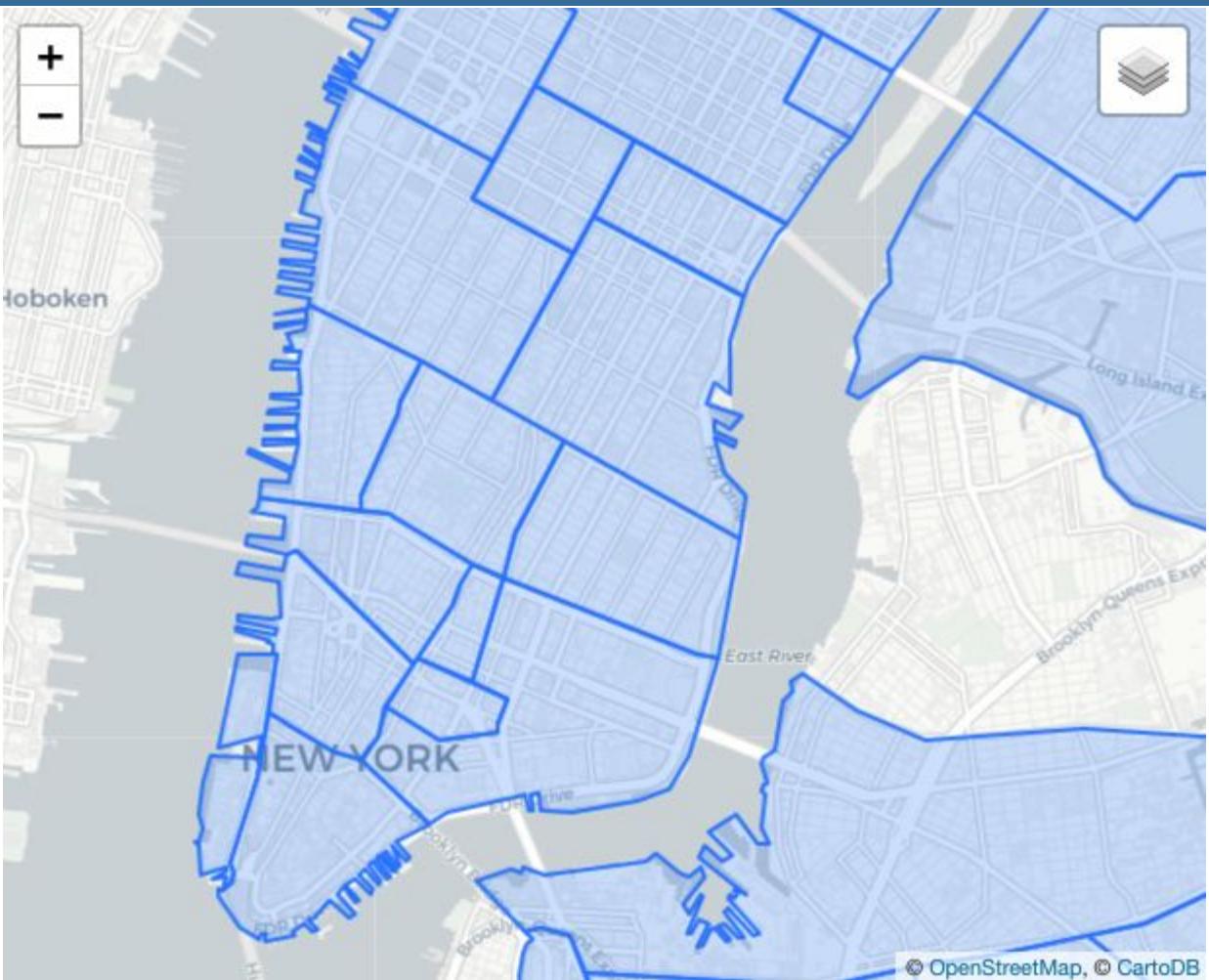
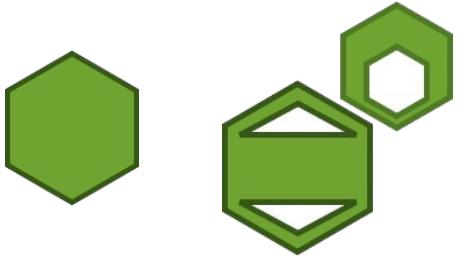
# LineStrings

- `ST_Length(linestring)`
- `ST_StartPoint(linestring)`
- `ST_EndPoint(linestring)`
- `ST_NumPoints(linestring)`

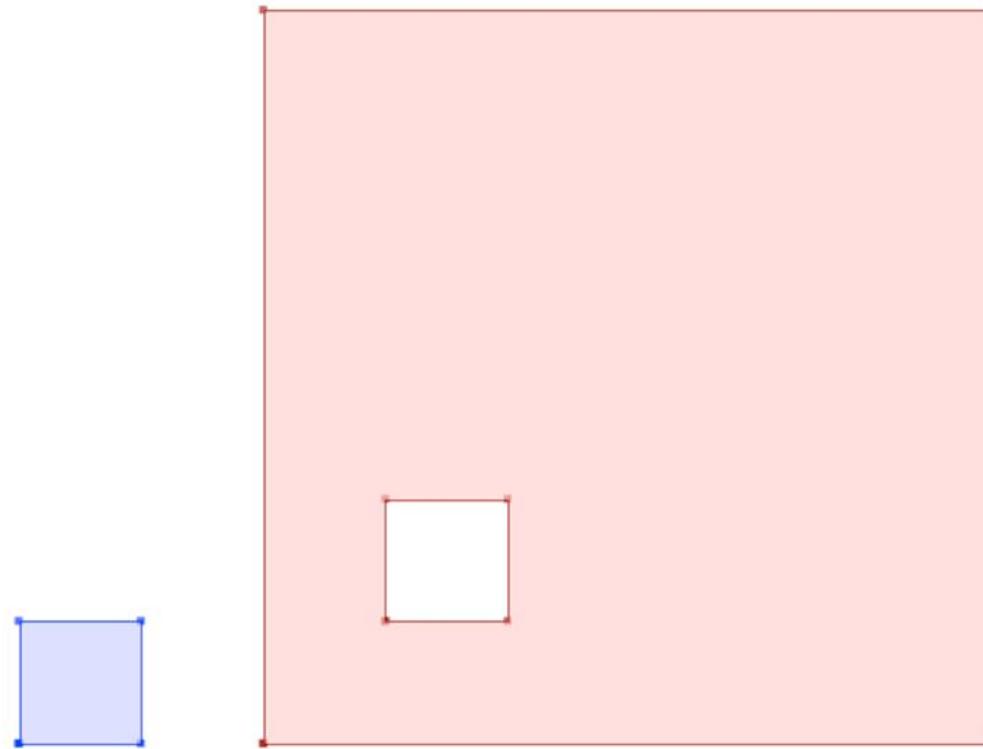
# Polygons

“Polygon” or  
“MultiPolygon”,  
representing one or more  
2-dimensional objects.

Census areas, parcels,  
counties, countries,  
neighborhoods, zoning  
areas, watersheds, and  
more.



# Polygons



## Polygons

```
SELECT ST_AsText(geom)
  FROM geometries
 WHERE name LIKE 'Polygon%';
```

---

```
POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))
POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),
         (1 1, 1 2, 2 2, 2 1, 1 1))
```

# Polygons

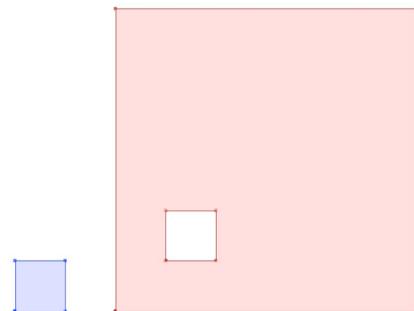
- `ST_Area(polygon)`
- `ST_NumInteriorRings(polygon)`
- `ST_ExteriorRing(polygon)`
- `ST_InteriorRing(polygon,n)`
- `ST_Perimeter(polygon)`

## Polygons

```
SELECT name, ST_Area(geom)  
FROM geometries  
WHERE name LIKE 'Polygon%';
```

---

Polygon		1
PolygonWithHole		99



## Geometry Formats

### **ST\_As...**

Text, EWKT, GML, KML, SVG, GeoJSON,  
Binary, EWKB

### **ST\_GeomFrom...**

Text, EWKT, GML, KML, GeoJSON,  
Binary, EWKB

## Geometry Formats

```
SELECT ST_AsText(  
    ST_GeometryFromText(  
        'LINESTRING(0 0 0,1 0 0,1 1 2)'  
    )  
);
```

---

LINESTRING Z (0 0 0,1 0 0,1 1 2)

# Geometry Formats

```
SELECT ST_AsGeoJSON(  
    ST_GeomFromGML(  
        '<gml:Point>  
            <gml:coordinates>  
                1,1  
            </gml:coordinates>  
        </gml:Point>'  
    ));
```

---

```
{"type": "Point", "coordinates": [1,1]}
```

## All roads lead to Rome ... (Geometry construction)

```
SELECT ST_AsEWKT(  
    ST_GeomFromText('POINT(1 1)', 4326)  
);
```

---

**SRID=4326;POINT(1 1)**

## All roads lead to Rome ... (Geometry construction)

```
SELECT ST_AsEWKT(  
    ST_SetSRID(  
        ST_GeomFromText('POINT(1 1)'),  
        4326  
    )  
);
```

---

SRID=4326;POINT(1 1)

## All roads lead to Rome ... (Geometry construction)

```
SELECT ST_AsEWKT(  
    ST_SetSRID(  
        ST_MakePoint(1, 1),  
        4326  
    )  
);
```

---

SRID=4326;POINT(1 1)

## All roads lead to Rome ... (Geometry construction)

```
SELECT ST_AsEWKT(  
    ST_SetSRID(  
        'POINT(1 1)'::geometry,  
        4326  
    )  
);
```

---

SRID=4326;POINT(1 1)

## All roads lead to Rome ... (Geometry construction)

```
SELECT ST_AsEWKT(  
    'SRID=4326;POINT(1 1)' ::geometry  
) ;
```

---

```
SRID=4326;POINT(1 1)
```

# Section 10 - Geometry Exercises

GEOMETRY FUNCTIONS		GEOGRAPHY FUNCTIONS		nyc_census_blocks (36592 records)		nyc_neighborhoods (129 records)		nyc_streets (19091 records)		nyc_subway_stations (491 records)		nyc_census_sociodata	
ST_Area(geometry)	ST_SRID(geometry)	ST_StartPoint(linestring)	ST_SymDifference(geometry,geometry)	blkid	A 15-digit code that uniquely identifies every census block. Eg: 360050001090900	name	Name of the neighborhood	name	Name of the street	name	An 11-digit code for every census tract. Eg: 36005000100	tractid	
ST_AsBinary(geometry)	ST_EndPoint(linestring)	ST_SymDifference(geometry,geometry)	ST_Touches(geometry, geometry)	popn_total	Total population in the census block	borough	Name of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	oneway	Is the street one-way? "yes" = yes, "no" = no	routes	Subway lines that run through this station	transit_total	
ST_AsGML(geometry)	ST_Union(geometry)	ST_Touches(geometry, geometry)	ST_Union(geometry)	popn_white	Number of people self-identifying as "White" in the block	transit_public	Number of workers in the tract who take public transit	type	Road type: e.g. primary, secondary, residential, motorway	transfers	Lines you can transfer to via this station	transit_private	
ST_AsGeoJSON(geometry)	ST_Within(geometry,geometry)	ST_Union(geometry)	ST_X(point)	popn_black	Number of people self-identifying as "Black" in the block	transit_other	Number of workers in the tract who use private automobiles / motorcycles	the_geom	Linear centerline of the street	express	Stations where express trains stop, "express" = yes, "no" = no	family_count	
ST_AsKML(geometry)	ST_XM(point)	ST_Within(geometry,geometry)	ST_Y(point)	popn_native	Number of people self-identifying as "Native American" in the block	family_income	Number of families in the tract	the_geom	Polygon boundary of the neighborhood	edu_total	Median family income in the tract (dollar)	income_aggregate	
ST_AsSVG(geometry)	ST_ZipPoint()	ST_XM(point)	ST_ZipPoint()	popn_asian	Number of people self-identifying as "Asian" in the block	edu_no_highschool_dipl	Total income of all families in the tract (dollar)	edu_no_highschool_dipl	Polygon boundary of the block	edu_highschool_dipl	Number of people with no highschool diploma	edu_college_dipl	
ST_AsText(geometry)	ST_M(point)	ST_ZipPoint()	ST_M(point)	popn_other	Number of people self-identifying with other categories in the block	edu_highschool_dipl	Number of people with highschool diplomas and no further education	edu_college_dipl	Number of people with college diplomas and no further education	edu_graduate_dipl	Number of people with graduate school diplomas	edu_graduate_dipl	
ST_Buffer(geometry,distance)	ST_AsText(geometry)	ST_M(point)	ST_AsText(geometry)	hours_total	Number of housing units in the block	hours_own	Number of owner-occupied housing units in the block	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Centroid(geometry)	ST_ConvexHull(geometry)	ST_X(point)	ST_X(point)	hours_own	Number of owner-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Contains(geometry,geometry)	ST_Crossed(geometry,geometry)	ST_Y(point)	ST_Y(point)	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_ConvexHull(geometry)	ST_Difference(geometry,geometry)	ST_ZipPoint()	ST_ZipPoint()	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Crossed(geometry,geometry)	ST_Disjoint(geometry,geometry)	ST_M(point)	ST_M(point)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Difference(geometry,geometry)	ST_Disjoint(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Disjoint(geometry,geometry)	ST_DWithin(geometry,geometry,radius)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_DistancedBuffer(geometry,geometry)	ST_EndPoint(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_EndPoint(geometry)	ST_Equal(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_ExteriorRing(geometry)	ST_ExteriorRing(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_GeomFromGML(text)	ST_GeomFromKML(text)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_GeomFromKML(text)	ST_GeomFromText(text)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_GeomFromText(text)	ST_GeomFromWKB(bytea)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_GeomFromWKB(bytea)	ST_GeometryN(collection,n)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_GeometryN(geometry,n)	ST_GeometryType(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_GeometryType(geometry)	ST_InteriorRingN(polygon,n)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_InteriorRingN(polygon,n)	ST_Intersects(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Intersects(geometry,geometry)	ST_Intersection(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Intersection(geometry,geometry)	ST_IsValid(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_IsValid(geometry)	ST_IsValidReason(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_IsValidReason(geometry)	ST_Length(geography)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Length(geography)	ST_MakePoint(double,double)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_MakePoint(double,double)	ST_NPoints(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_NPoints(geometry)	ST_NRings(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_NRings(geometry)	ST_OrderingEqual(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_OrderingEqual(geometry,geometry)	ST_Overlaps(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Overlaps(geometry,geometry)	ST_Perimeter(polyline)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Perimeter(polyline)	ST_PointsOnSurface(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_PointsOnSurface(geometry)	ST_Relate(geometry,geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Relate(geometry,geometry)	ST_Reverse(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	
ST_Reverse(geometry)	ST_Simplify(geometry)	ST_AsText(geometry)	ST_AsText(geometry)	the_geom	Polygon boundary of the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	Number of renter-occupied housing units in the block	boroughname	None of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	hours_rent	

**What is the **area** of the 'West Village' neighborhood?**

**SELECT**

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

---

**1044614.5296486**

**What is the geometry type of 'Pelham St'? The length?**

**SELECT**

**ST\_GeometryType(geom),**

**ST\_Length(geom)**

**FROM nyc\_streets**

**WHERE name = 'Pelham St';**

---

**ST\_MultiLineString | 50.323**

**What is the **GeoJSON** representation of ‘Broad St’ subway station?**

**SELECT**

```
ST_AsGeoJSON(geom, 0)
FROM nyc_subway_stations
WHERE name = 'Broad St';
```

---

```
{"type": "Point",
"crs": {"type": "name",
          "properties": {"name": "EPSG:26918"}},
"coordinates": [583571, 4506714]}
```

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

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

---

**10418.904717199996**

**What is the area of Manhattan in acres?**

**SELECT**

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

---

**14601.3987215548**

## What is the **most westerly** subway station?

**SELECT**

```
ST_X(geom), name  
FROM nyc_subway_stations  
ORDER BY ST_X(geom)  
LIMIT 1;
```

---

563292.1 | Tottenville

# **Section 11 - Spatial Relationships**

# Spatial Relationship Functions

- `ST_Intersects(A, B)`
- `ST_DWithin(A, B, d)`
- `ST_Distance(A, B)`
- `ST_Within, ST_Contains(A, B)`
- `ST_Equals(A, B)`
- `ST_Touches(A, B)`
- `ST_Disjoint, ST_Crosses, ST_Overlaps(A, B)`



## ST\_Equals(A, B)

## ST\_OrderingEquals(A, B)

Equals tests that A and B cover the same space, regardless of representation differences (extra vertices, order of vertices).

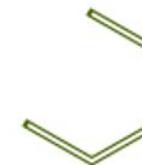
OrderingEquals insists on structural identity.



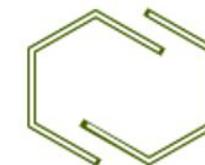
Point & Multipoint



Multipoint & Multipoint



Linestring & Linestring



Multilinestring & Multilinestring



Polygon & Polygon



Multipolygon & Multipolygon

## What is geometry of Broad Street subway station?

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

---

0101000020266900000EEBD4CF27CF2141BC17D69516315141

## What subway station record matches that geometry?

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

---

Broad St

# ST\_Intersects(A, B)

# ST\_Disjoint(A, B)

Intersects and disjoint are opposites.  
Any kind of interactions between two shapes is an intersection, and implies the pair are not disjoint, and vice versa.

A intersects B  $\Rightarrow$  A not disjoint B

A disjoint B  $\Rightarrow$  A not intersects B



## What is the well-known text (WKT) of Broad Street subway station?

```
SELECT name, ST_AsText(geom, 0)
  FROM nyc_subway_stations
 WHERE name = 'Broad St';
```

---

POINT(583571 4506714)

## What neighborhood intersects that subway station?

```
SELECT name, boroname
FROM nyc_neighborhoods
WHERE ST_Intersects(
    geom,
    ST_GeomFromText(
        'POINT(583571 4506714)' ,
        26918));
```

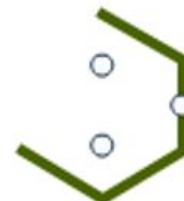
---

Financial District | Manhattan

## ST\_Crosses(A, B)

Mostly used to test linestrings, which can be said to cross when their interiors have interactions.

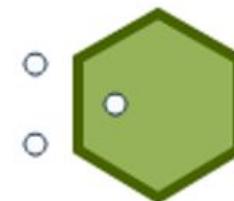
When linestrings cross polygon boundaries, the crosses condition is also true.



Multipoint & Linestring



Linestring & Linestring



Multipoint & Polygon



Linestring & Multipolygon

## ST\_Overlaps(A, B)

Shapes overlap when their interiors interact with each other and also with the exterior of the shape.

So objects that are contained or within do not overlap, overlaps is what normal people might call “partial overlap”.



Multipoint & Multipoint



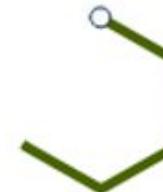
Linestring & Linestring



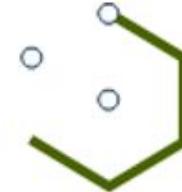
Polygon & Polygon

## ST\_Touches(A, B)

Shapes touch when their boundaries interact but their interiors do not. End points for lines, exterior rings for polygons. Usually used for testing that polygons have ring-touching only.



Point &amp; Linestring



Multipoint &amp; Linestring



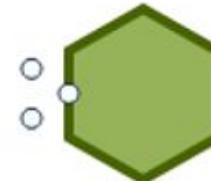
Linestring &amp; Linestring



Linestring &amp; Polygon



Point &amp; Polygon

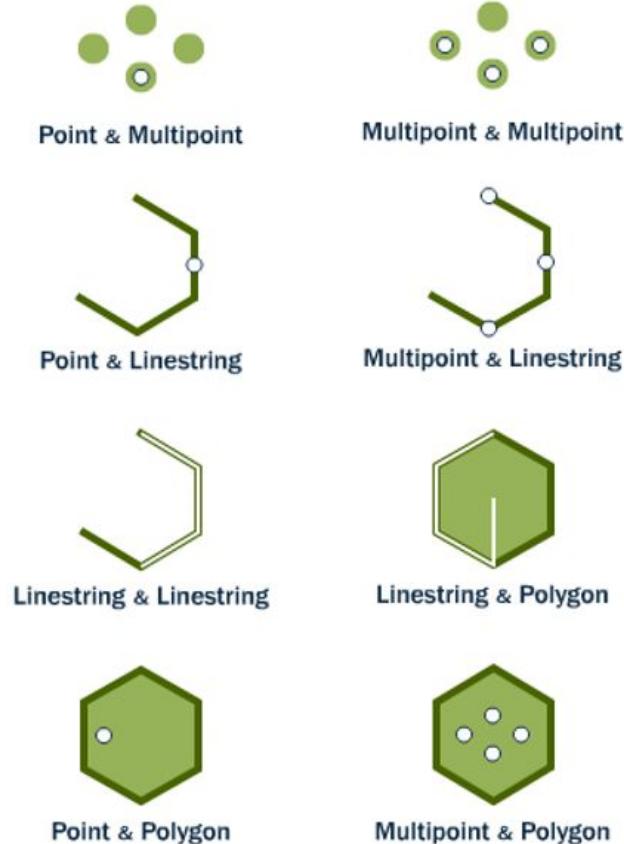


Multipoint &amp; Polygon

# ST\_Within(A, B)

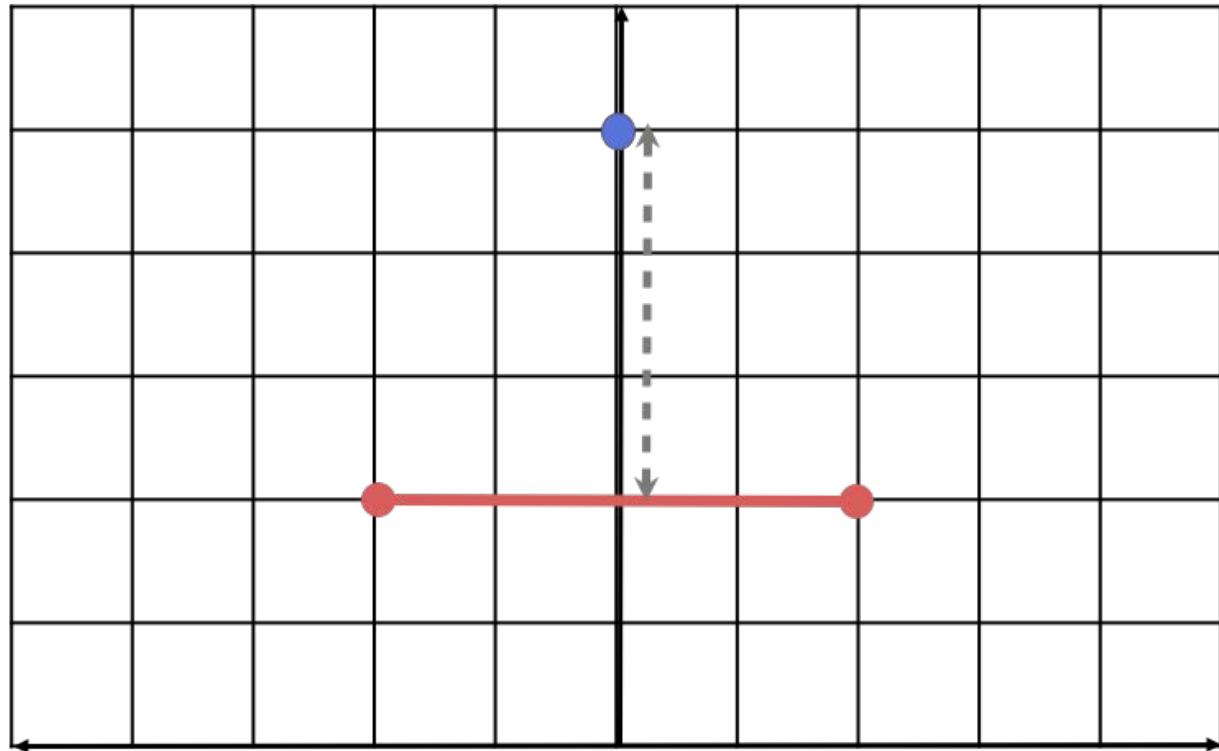
# ST\_Contains(B, A)

Within and contains are about objects being fully inside. One important caveat, for both functions an object on the **boundary** is not considered within. So a point on the outer ring of a polygon is not within the polygon.



## ST\_Distance(A, B)

Returns the **shortest** distance between the two geometries, in this case the distance from the point to the line mid-point.



## ST\_Distance(A, B)

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

---

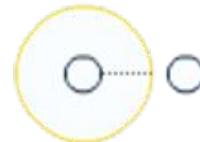
## ST\_DWithin(A, B, R)

Index-enabled radius search function. True when the distance from geometry A to geometry B is less than radius R. False otherwise.

Use instead of  
**ST\_Distance(A, B) < R**, in  
order to get benefit of spatial  
index.



Point & Point (True)



Point & Point (False)



Polygon & Point (True)



Polygon & Point (False)

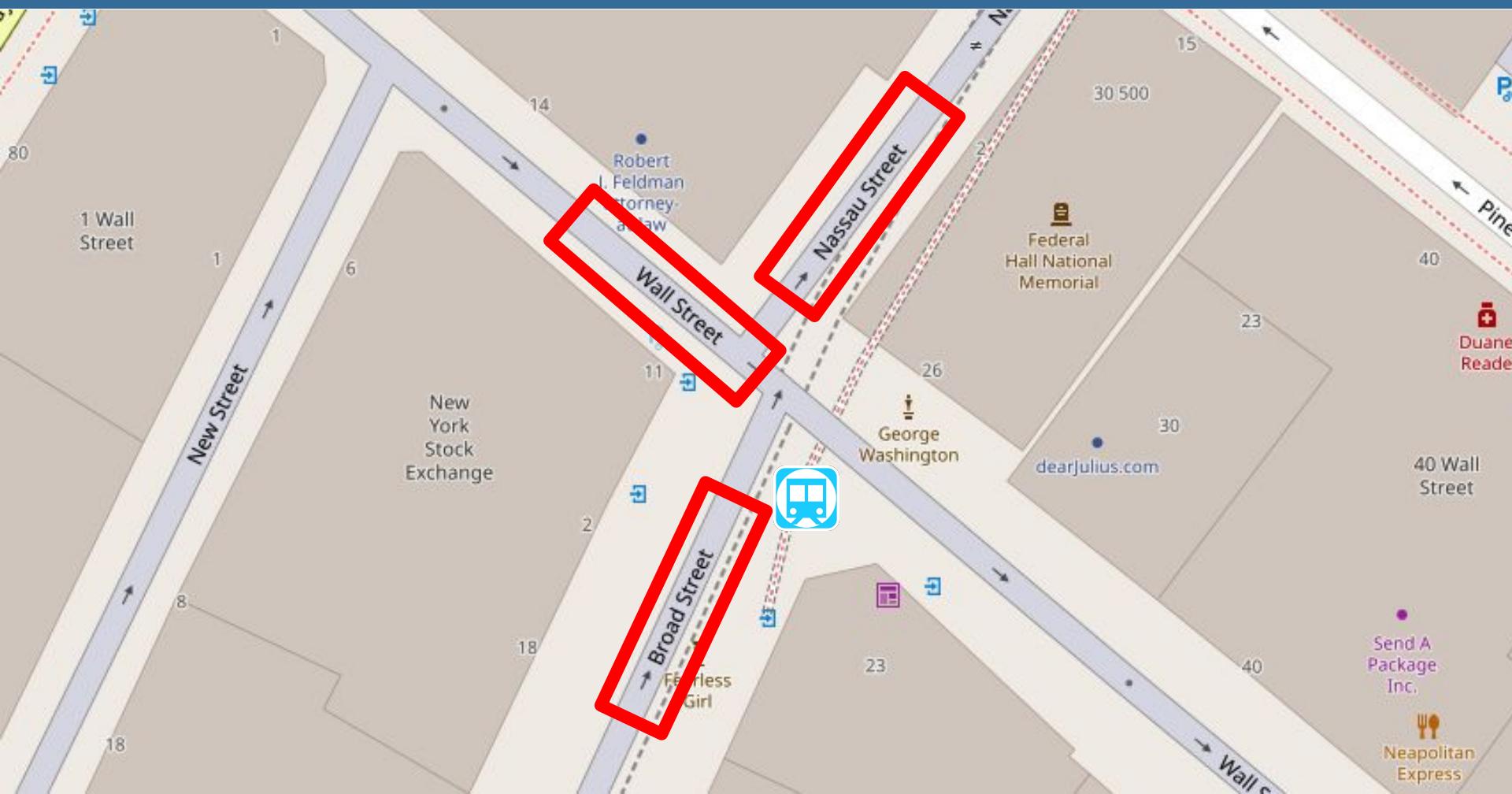
## What streets are within 10 meters of Broad Street subway station?

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

## Section 11 - Spatial Relationships



PostGIS



# Section 12 - Spatial Relationship Exercises

GEOMETRY FUNCTIONS		GEOGRAPHY FUNCTIONS		nyc_census_blocks (36592 records)		nyc_neighborhoods (129 records)		nyc_streets (19091 records)		nyc_subway_stations (491 records)		nyc_census_sociodata	
ST_Area(geometry)	ST_SRID(geometry)	ST_StartPoint(linestring)	ST_SymDifference(geometry,geometry)	blkid	A 15-digit code that uniquely identifies every census block. Eg: 360050001090900	name	Name of the neighborhood	name	Name of the street	name	Name of the tract	tractid	An 11-digit code for every census tract. Eg: 36005000100
ST_AsBinary(geometry)	ST_EndPoint(linestring)	ST_SymDifference(geometry,geometry)	ST_Touches(geometry, geometry)	popn_total	Total population in the census block	borough	Name of borough: Manhattan, The Bronx, Brooklyn, Staten Island, Queens	oneway	Is the street one-way? "yes" = yes, "no" = no	routes	Subway lines that run through this station	transit_total	Number of workers in the tract
ST_AsGML(geometry)	ST_Union(geometry, geometry)	ST_Touches(geometry, geometry)	ST_Union(geometry, geometry)	popn_white	Number of people self-identifying as "White" in the block	transit_public	Number of workers in the tract who take public transit	type	Road type: e.g. primary, secondary, residential, motorway	transit_private	Number of workers in the tract who use private automobiles / motorcycles	transit_other	Number of workers in the tract who use other forms like walking / biking
ST_AsGeoJSON(geometry)	ST_Within(geometry,geometry)	ST_X(point)	ST_ZipCode()	popn_black	Number of people self-identifying as "Black" in the block	transit_time_mins	Total minutes spent commuting by all workers in the tract (minutes)	the_geom	Polygon boundary of the neighborhood	express	Stations where express trains stop, "express" = yes, "no" = no	family_count	Number of families in the tract
ST_AsKML(geometry)	ST_XM(point)	ST_Y(point)	ST_M(point)	popn_native	Number of people self-identifying as "Native American" in the block	family_income	Median family income in the tract (dollar)	edu_total	Number of people with educational history	edu_highschool_dipl	Number of people with no highschool diploma	family_income_aggregate	Total income of all families in the tract (dollar)
ST_AsSVG(geometry)	ST_ZipCode()	ST_M(point)		popn_asian	Number of people self-identifying as "Asian" in the block	edu_highschool_dipl	Number of people with highschool diplomas and no further education	edu_college_dipl	Number of people with college diplomas and no further education	edu_graduate_dipl	Number of people with graduate school diplomas	popn_other	Number of people self-identifying with other categories in the block
ST_AsText(geometry)				popn_amer	Number of people self-identifying as "American" in the block	popn_brown	Number of renter-occupied housing units in the block	hours_own	Number of owner-occupied housing units in the block	hours_rent	Number of renter-occupied housing units in the block	hours_brown	Number of houses in the block
ST_Buffer(geometry,distance)				hours_total	Number of housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Centroid(geometry)				hours_own	Number of owner-occupied housing units in the block	the_geom	Polygon boundary of the neighborhood	hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Containing(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_ConvexHull(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Crossed(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Difference(geometry,geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_DWithin(geometry,geometry,radius)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Disjoint(geometry,geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Distanced(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_EndPoint(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Equal(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_ExteriorRing(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_GeomFromGML(text)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_GeomFromKML(text)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_GeomFromText(text)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_GeomFromWKB(bytea)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_GeometryN(collection,n)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_GeometryType(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_InteriorRingN(polygon,n)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Intersection(geometry,geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Intersection(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_IsValid(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_IsValidReason(geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Length(geometries)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_MakePoint(xdouble,ydouble)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_NPoints(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_NRings(polygon)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_NumGeometries(collection)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_OrderingEqual(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Overlaps(geometry,geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Perimeter(polygons)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_PointsOnSurface(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Relate(geometry,geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Reverse(geometry)				hours_own	Number of owner-occupied housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block
ST_Simplify(geometry)				hours_total	Number of housing units in the block			hours_rent	Number of renter-occupied housing units in the block	boroughname	Polygon boundary of the neighborhood	hours_brown	Number of houses in the block

**What is the well-known text for the street  
'Atlantic Commons'?**

```
SELECT ST_AsText(geom, 0)  
FROM nyc_streets  
WHERE name = 'Atlantic Commons';
```

---

```
MULTILINESTRING((  
  586782 4504202,  
  586864 4504216))
```

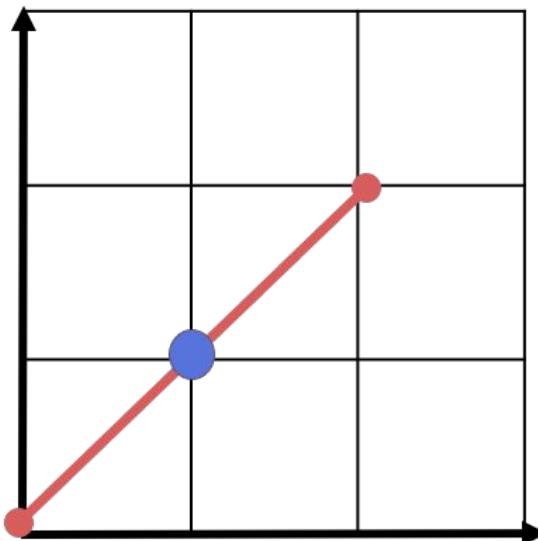
**What neighborhood and borough is  
'POINT(586782 4504202)' in?**

```
SELECT name, boroname
FROM nyc_neighborhoods
WHERE ST_Intersects(
    geom,
    ST_GeomFromText(
        'POINT(586782 4504202)', 26918)
);
```

**How many people live within 50 meters of  
'POINT(586782 4504202)'?**

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

For '**LINESTRING(0 0, 2 2)**' and '**POINT(1 1)**' which of these relationships are true?



**A**

Intersects,  
Touches,  
Contains,  
Disjoint,  
Overlaps,  
Crosses,  
Within

**B**

# Point and Line Relationships

```
SELECT
```

```
ST_Intersects(l,p), ST_Touches(l,p),  
ST_Contains(l,p), ST_Disjoint(l,p),  
ST_Overlaps(l,p), ST_Crosses(l,p),  
ST_Within(l,p)
```

```
FROM (
```

```
SELECT
```

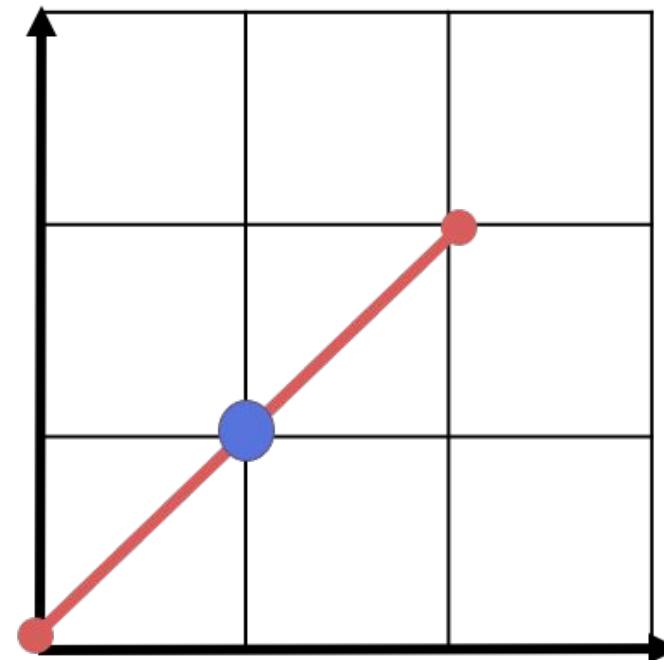
```
'LINESTRING(0 0, 2 2))::geometry AS l,
```

```
'POINT(1 1)::geometry AS p
```

```
) AS subquery;
```

# Point and Line Relationships

st_intersects	t
st_touches	f
st_contains	t
st_disjoint	f
st_overlaps	f
st_crosses	f
st_within	f



## How far apart are 'Columbus Cir' and 'Fulton Ave'?

```
SELECT ST_Distance(a.geom, b.geom)
FROM nyc_streets a,
     nyc_streets b
WHERE a.name = 'Fulton Ave'
  AND b.name = 'Columbus Cir';
```

# Section 13 - Spatial Joins

**What neighborhood is the 'Broad St' station in?**

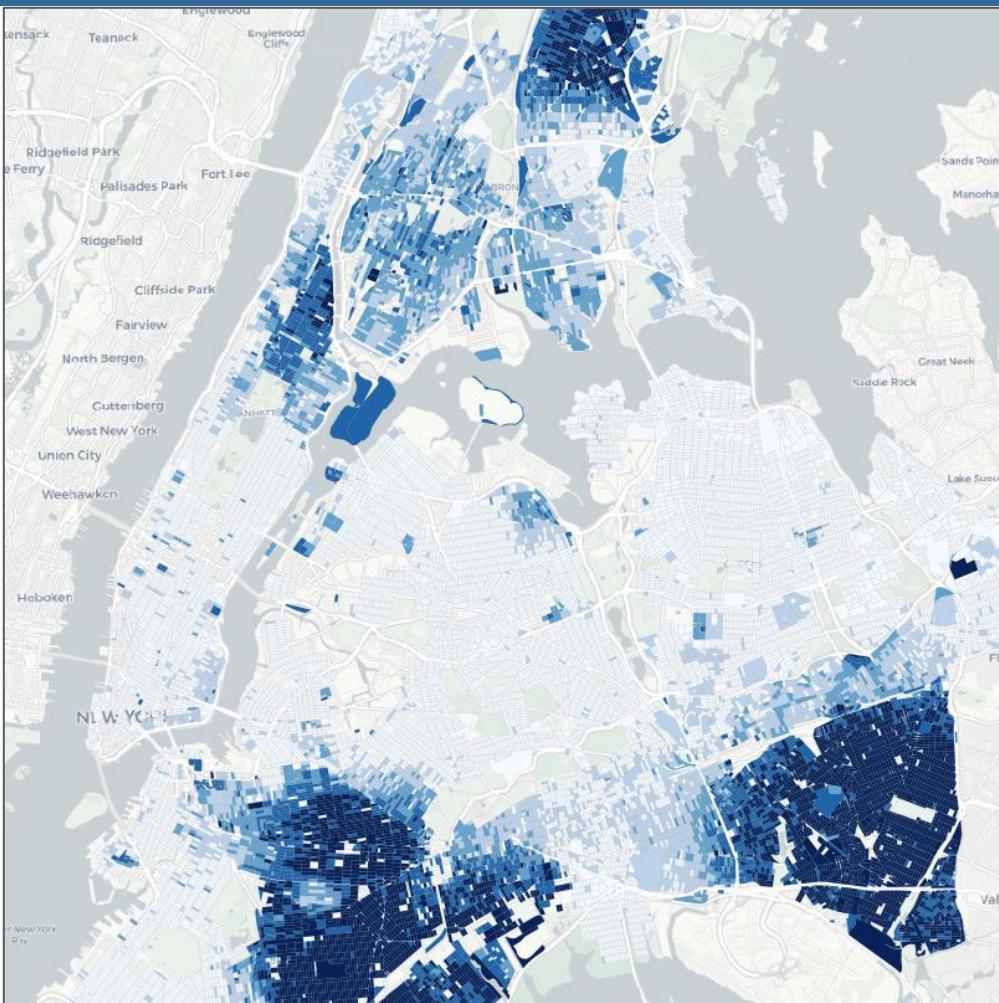
## Remember...

```
SELECT name, boroname
FROM nyc_neighborhoods
WHERE
    ST_Intersects(
        geom,
        ST_GeomFromText(
            'POINT(583571 4506714)',
            26918));
```

**Do it in one step, with a spatial join!**

```
SELECT s.name, n.name, n.boroname
FROM nyc_neighborhoods AS n
JOIN nyc_subway_stations AS s
ON ST_Contains(
    n.geom,
    s.geom
)
WHERE s.name = 'Broad St';
```

**What is the  
population and racial  
make-up of the  
neighborhoods of  
Manhattan?**



## SELECT

```
n.name AS neighborhood_name,  
SUM(c.popn_total) AS population,  
100*SUM(c.popn_white)/SUM(c.popn_total) AS white_pct,  
100*SUM(c.popn_black)/SUM(c.popn_total) AS black_pct
```

```
FROM nyc_neighborhoods AS n  
JOIN nyc_census_blocks AS c  
ON ST_Intersects(  
    n.geom,  
    c.geom  
)
```

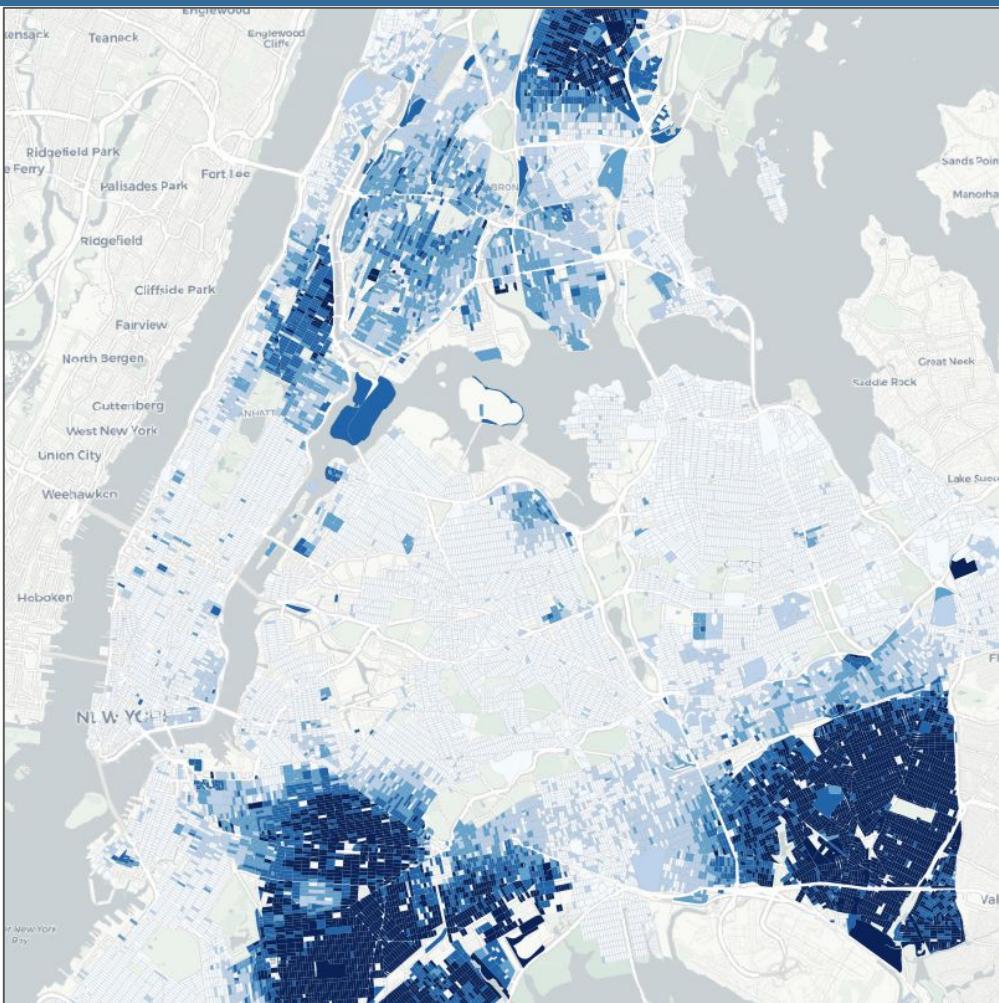
```
WHERE n.boroname = 'Manhattan'  
GROUP BY n.name  
ORDER BY white_pct DESC;
```

## Section 13 - Spatial Joins

neighborhood_name	popn	white %	black %
Carnegie Hill	18763	90.1	1.4
West Village	26718	87.6	2.2
North Sutton Area	22460	87.6	1.6
Upper East Side	203741	85.0	2.7
Soho	15436	84.6	2.2
Greenwich Village	57224	82.0	2.4
Central Park	46600	79.5	8.0
Tribeca	20908	79.1	3.5
Gramercy	104876	75.5	4.7
Murray Hill	29655	75.0	2.5
Chelsea	61340	74.8	6.4
Upper West Side	214761	74.6	9.2
Midtown	76840	72.6	5.2
Battery Park	17153	71.8	3.4

neighborhood_name	popn	white %	black %
Financial District	34807	69.9	3.8
Clinton	32201	65.3	7.9
East Village	82266	63.3	8.8
Garment District	10539	55.2	7.1
Morningside Heights	42844	52.7	19.4
Little Italy	12568	49.0	1.8
Yorkville	58450	35.6	29.7
Inwood	50047	35.2	16.8
Washington Heights	169013	34.9	16.8
Lower East Side	96156	33.5	9.1
East Harlem	60576	26.4	40.4
Hamilton Heights	67432	23.9	35.8
Chinatown	16209	15.2	3.8
Harlem	134955	15.1	67.1

**Let's explore the  
racial geography  
of New York City...**



# Overall Racial Make-up of NYC

**SELECT**

```
100.0*SUM(popn_white)/SUM(popn_total) AS white_pct,  
100.0*SUM(popn_black)/SUM(popn_total) AS black_pct,  
SUM(popn_total) AS popn_total  
FROM nyc_census_blocks;
```

---

white_pct	black_pct	popn_total
44.00395007628105	25.546578900241613	8175032



You, must take  
the A train.  
To, go to Sugar  
Hill way up in  
Harlem.

**What is the racial make-up of  
the areas served by the A train?**

**First, we must determine where the A train stops.**

# Our routes are comma-separated strings!

```
SELECT DISTINCT routes  
FROM nyc_subway_stations;
```

---

4,5

N,Q,R,W

J

B,M,Q,R

D,F,N,Q

J,M

E,F

...

## Postgres string function: **strpos()**

**strpos(routes, 'A')** returns a non-zero number if 'A' is in the routes field

Check out <https://www.postgresql.org/docs/current/functions-string.html>

## Find all routes with an “A”

```
SELECT DISTINCT routes  
FROM nyc_subway_stations AS subways  
WHERE strpos(subways.routes, 'A') > 0;
```

---

A,C

A,B,C,D

A,C,E,L

A,C,F

A,B,C

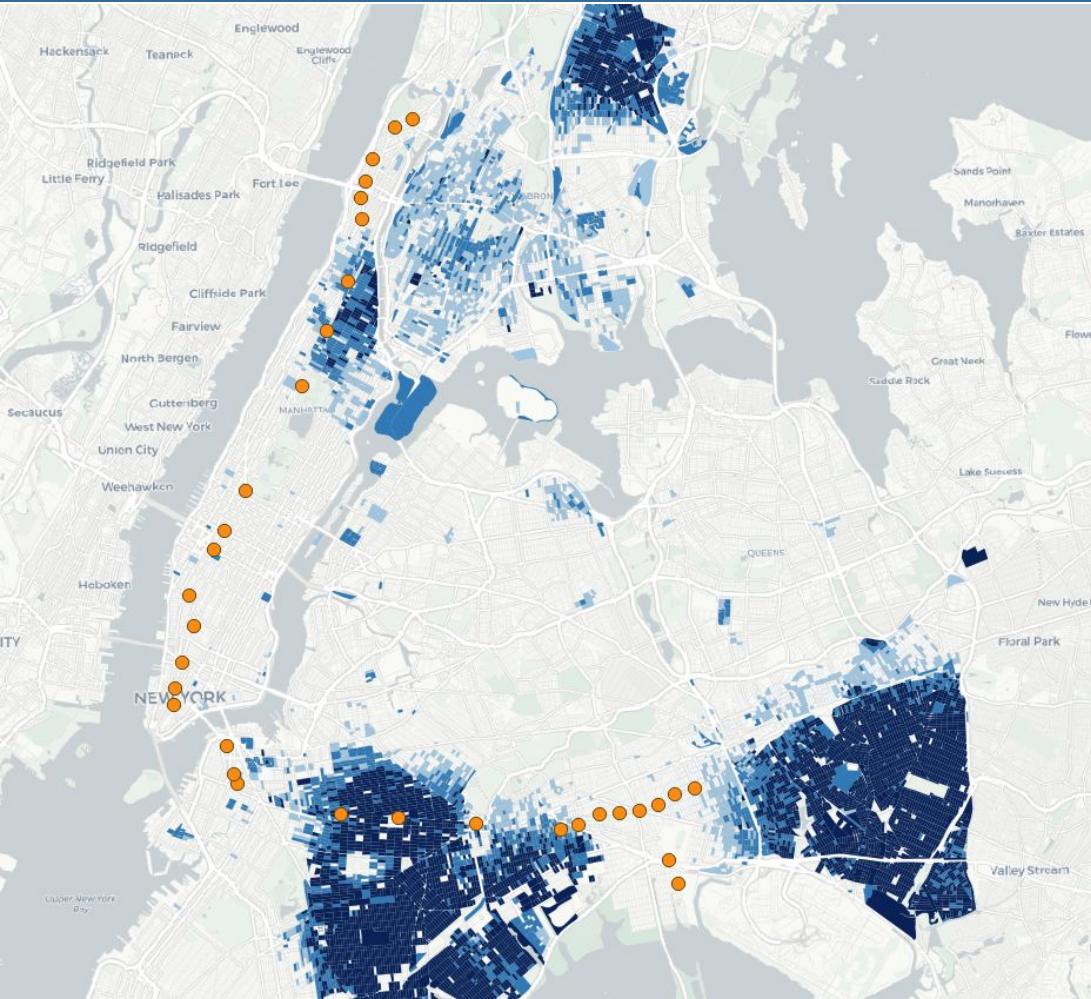
A,S

A,C,E

...

### The route of the A train.

What is the racial makeup within 200 meters of each stop? Who is served by the A train?



## Summarize population 200m from A train stops

SELECT

```
100*SUM(c.popn_white)/SUM(c.popn_total) AS white_pct,  
100*SUM(c.popn_black)/SUM(c.popn_total) AS black_pct,  
SUM(popn_total) AS popn_total
```

FROM nyc\_census\_blocks AS c

JOIN nyc\_subway\_stations AS s

ON ST\_DWithin(

c.geom,

s.geom,

200

)

WHERE strpos(s.routes, 'A') > 0;

# New York

44.00% white

25.55% black

# A Train

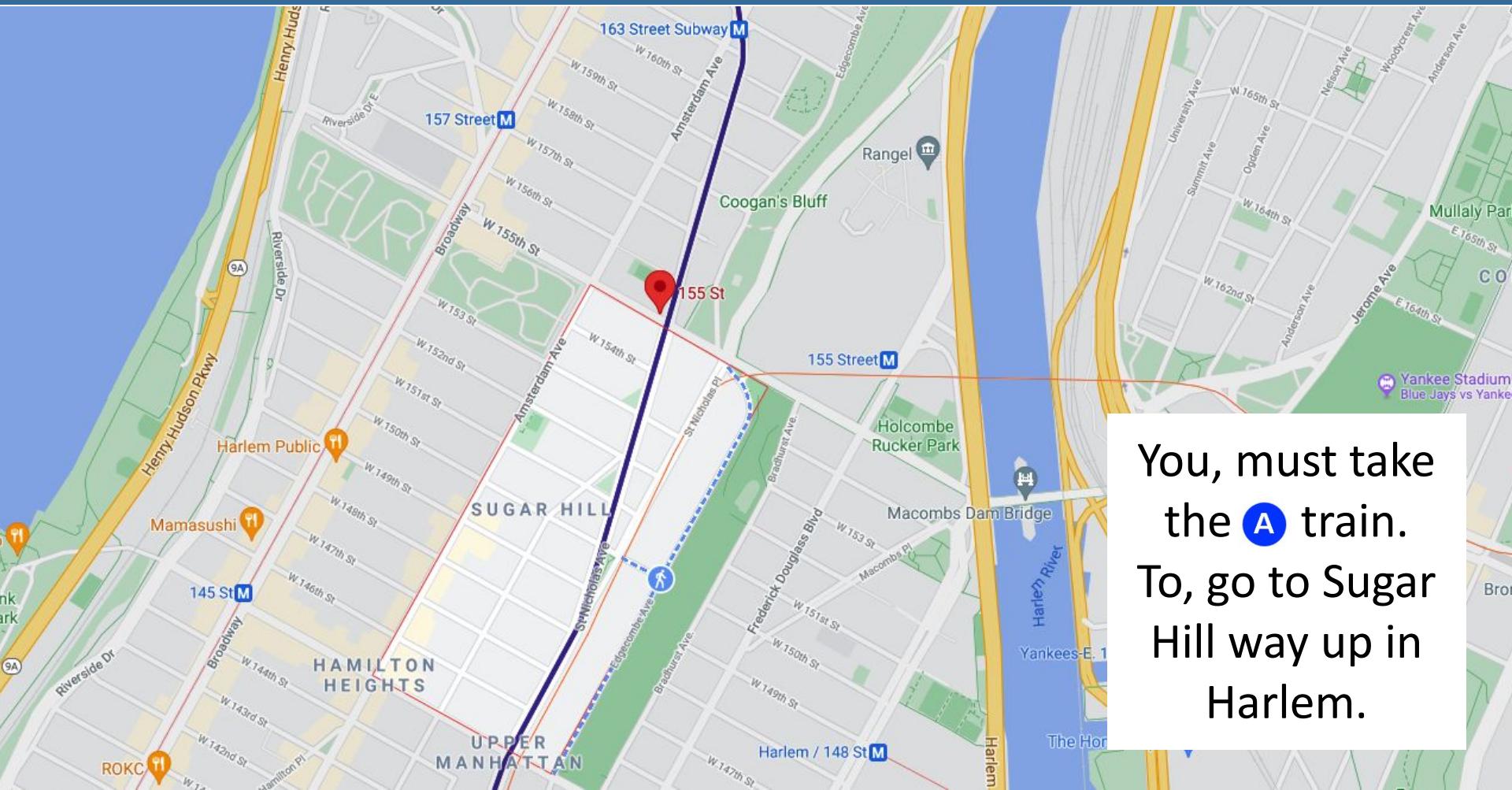
45.59% white

22.09% black

## Section 13 - Spatial Joins



PostGIS



You, must take  
the **A** train.  
To, go to Sugar  
Hill way up in  
Harlem.

# Section 14 - Spatial Joins Exercises

GEOMETRY FUNCTIONS		GEOGRAPHY FUNCTIONS		nyc_census_blocks (36592 records)		nyc_neighborhoods (129 records)		nyc_streets (19091 records)		nyc_subway_stations (491 records)		nyc_census_socidata	
ST_Area(geometry)	ST_SRID(geometry)	ST_StartPoint(linestring)	ST_SymDifference(geometry,geometry)	blkid	A 15-digit code that uniquely identifies every census block. Eg: 360050001009000	name	Name of the neighborhood	name	Name of the street	name	Name of the station	tractid	An 11-digit code for every census tract. Eg: 36005000100
ST_AsBinary(geometry)	ST_EndPoint(geom)	ST_Touches(geometry, geometry)	ST_Union(geometry, geometry)	popn_total	The number of people in the census block	boroname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens	oneway	Is the street one-way? "yes" = yes, "no" = no	borough	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens	transit_total	Number of workers in the tract
ST_AsGML(geometry)	ST_AsGeoJSON(geometry)	ST_Intersection(geometry, geometry)	ST_DWithin(geometry,geometry,radius)	popn_white	Number of people self-identifying as "White" in the block	the_geom	Polygon boundary of the neighborhood	type	Road type. Eg. primary, secondary, residential, motorway	routes	Subway lines that run through this station	transit_public	Number of workers in the tract who take public transit
ST_AsGeoJSON(geometry)	ST_AsKML(geometry)	ST_Disjoint(geometry,geometry)	ST_Buffer(geometry,distance)	popn_black	Number of people self-identifying as "Black" in the block	transit_private	Number of workers in the tract who use private automobiles / motorcycles	express	Stations where express trains stop, "express" = yes, "no" = no	transfers	Lines you can transfer to via this station	transit_other	Number of workers in the tract who use other forms like walking / biking
ST_AsKML(geometry)	ST_AsSVG(geometry)	ST_Crosses(geometry,geometry)	ST_Centroid(geometry)	popn_native	Number of people self-identifying as "Native American" in the block	family_count	Total number of families in the tract	the_geom	Point location of the station	family_mins	Total minutes spent in transit by all workers in the tract (minutes)	family_income_mean	Median family income in the tract (dollars)
ST_AsText(geometry)	ST_GeomFromWKB(bytes)	ST_Difference(geometry,geometry)	ST_Contains(geometry,geometry)	popn_asian	Number of people self-identifying as "Asian" in the block	family_income_aggregate	Total income for all families in the tract (dollars)	edu_total	Number of people with education history	edu_no_highschool_dipl	Number of people with no highschool diploma	edu_highschool_dipl	Number of people with highschool diploma and no further education
ST_Buffer(geometry,distance)	ST_GeomFromText(text)	ST_Disjoint(geometry,geometry)	ST_ConvexHull(geometry)	popn_other	Number of people self-identifying with other categories in the block	edu_college_dipl	Number of people with college diploma and no further education	edu_college_dipl	Number of people with graduate school diploma	edu_graduate_dipl	Number of people with postgraduate diplomas	edu_postgrad_dipl	Number of people with postgraduate diplomas
ST_Centroid(geometry)	ST_GeomFromWKB(bytes)	ST_Disjoint(geometry,geometry)	ST_ConvexHull(geometry)	house_total	Number of housing units in the block	the_geom	Polygon boundary of the block	the_geom	Polygon boundary of the block	the_geom	Polygon boundary of the block	tractid	An 11-digit code for every census tract. Eg: 36005000100
ST_Contains(geometry,geometry)	ST_GeomFromWKB(bytes)	ST_Disjoint(geometry,geometry)	ST_Crosses(geometry,geometry)	house_own	Number of owner-occupied housing units in the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_ConvexHull(geometry)	ST_GeomFromText(text)	ST_Disjoint(geometry,geometry)	ST_Difference(geometry,geometry)	house_rent	Number of renter-occupied housing units in the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Crosses(geometry,geometry)	ST_GeomFromWKB(bytes)	ST_Disjoint(geometry,geometry)	ST_DWithin(geometry,geometry,radius)	boroname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_Difference(geometry,geometry)	ST_GeomFromText(text)	ST_Disjoint(geometry,geometry)	ST_EndPoint(geom)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_DWithin(geometry,geometry,radius)	ST_GeomFromText(text)	ST_Disjoint(geometry,geometry)	ST_M(point)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_Disjoint(geometry,geometry)	ST_GeomFromWKB(bytes)	ST_Disjoint(geometry,geometry)	ST_M(point)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Distance(geometry,geometry)	ST_GeomFromWKB(bytes)	ST_Disjoint(geometry,geometry)	ST_Point(x,y)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_EndPoint(geom)	ST_GeomFromText(text)	ST_Disjoint(geometry,geometry)	ST_Point(x,y)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Equals(geometry,geometry)	ST_GeomFromWKB(bytes)	ST_Disjoint(geometry,geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_ExteriorRing(polygon)	ST_GeomFromText(text)	ST_Disjoint(geometry,geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_GeomFromKML(text)	ST_GeomFromText(text)	ST_Intersects(geometry,geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_GeomFromWKB(bytes)	ST_GeomFromText(text)	ST_Intersects(geometry,geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_GeometryN(collection,n)	ST_GeometryType(geometry)	ST_IsValid(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_GeometryN(collection,n)	ST_GeometryType(geometry)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_InteriorRingN(polygon,n)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_Intersects(geometry,geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Intersection(geometry,geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_IsValid(geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Length(linestring)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_MakePoint(double,double)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_MakePoint(double,double)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_NPoints(geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_NRings(polygon)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_NumGeometries(collection)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_OrderingEquals(geometry,geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_Overlaps(geometry,geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Perimeter(polygon)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_PointOnSurface(geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Relate(geometry,geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education
ST_Reverse(geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Number of people with college diploma and no further education	tractid	Number of people with graduate school diploma	tractid	Number of people with postgraduate diplomas	tractid	Number of people with postgraduate diplomas
ST_Simplify(geometry)	ST_GeomFromText(text)	ST_IsValidReason(geometry)	ST_Polygonize(geometry)	the_geom	Polygon boundary of the block	tractid	Median household income in the tract (dollars)	tractid	Number of people with education history	tractid	Number of people with no highschool diploma	tractid	Number of people with highschool diploma and no further education

**What subway station is in ‘Little Italy’? What subway route is it on?**

```
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';
```

---

Spring St | 6

## 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;
```

---

Midtown

Hunts Point

...

**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';
```

---

17153

## What neighborhood has the highest population density (persons/km<sup>2</sup>)?

```
SELECT n.name,  
       1000000 * Sum(c.popn_total) /  
                  ST_Area(n.geom) AS popn_per_sqkm  
FROM nyc_census_blocks AS c  
JOIN nyc_neighborhoods AS n  
ON ST_Intersects(c.geom, n.geom)  
GROUP BY n.name, n.geom  
ORDER BY popn_per_sqkm DESC;
```

---

North Sutton Area | 68435.13283772678

# **Section 15 - Spatial Indexing**

# A spatial database has...

- **Spatial Data Types**
  - geometry, geography
- **Spatial Indexes**
  - r-tree, quad-tree, kd-tree
- **Spatial Functions**
  - ST\_Length(geometry), ST\_X(geometry)

# A spatial index speeds spatial query ...

- Join two tables of 10,000 records each

## Without Index

$10,000 * 10,000 = \mathbf{100,000,000}$   
comparisons

## With Index

$10,000 + 10,000 = \mathbf{20,000}$   
comparisons

**To prove it... remove the index.**

```
DROP INDEX nyc_census_blocks_geom_idx;
```

**Run a spatial join.**

```
SELECT b.blkid
FROM nyc_census_blocks b
JOIN nyc_subway_stations s
  ON ST_Contains(b.geom, s.geom)
WHERE s.name LIKE 'B%';
```

~300 ms

## Create the index again.

```
CREATE INDEX nyc_census_blocks_geom_idx  
ON nyc_census_blocks USING GIST (geom);
```

## Run the join again.

```
SELECT blocks.blkid  
FROM nyc_census_blocks b  
JOIN nyc_subway_stations s  
  ON ST_Contains(b.geom, s.geom)  
WHERE s.name LIKE 'B%';
```

~90 ms

## Spatial Index Cliff Notes

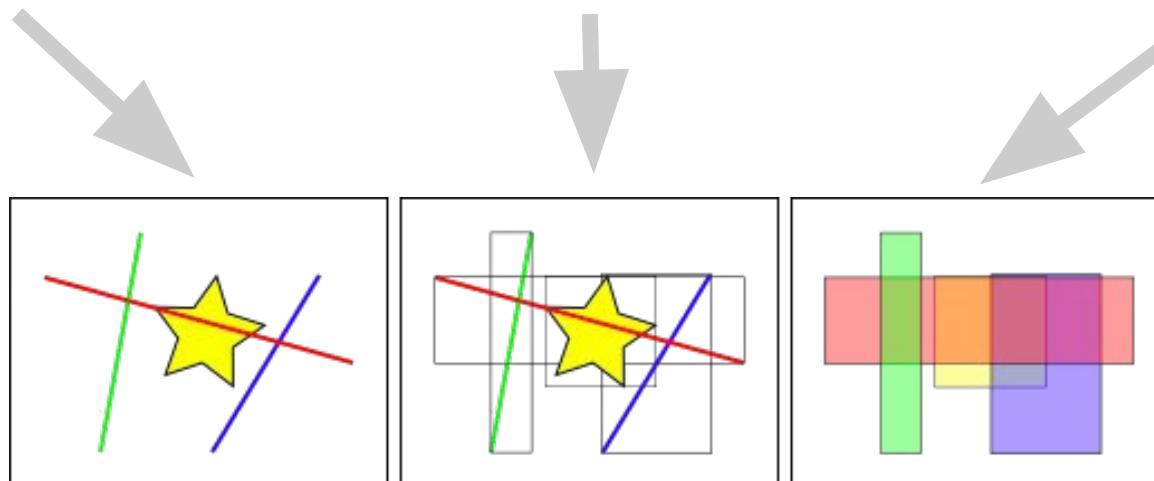
- **CREATE INDEX index\_name ON table\_name USING GIST (geom)**
- Use a “spatially indexed function” in **JOIN** or **WHERE** clause
  - **ST\_Intersects(A, B), ST\_Contains(A, B), ST\_Within(A, B)**
  - **ST\_DWithin(A, B, R)**

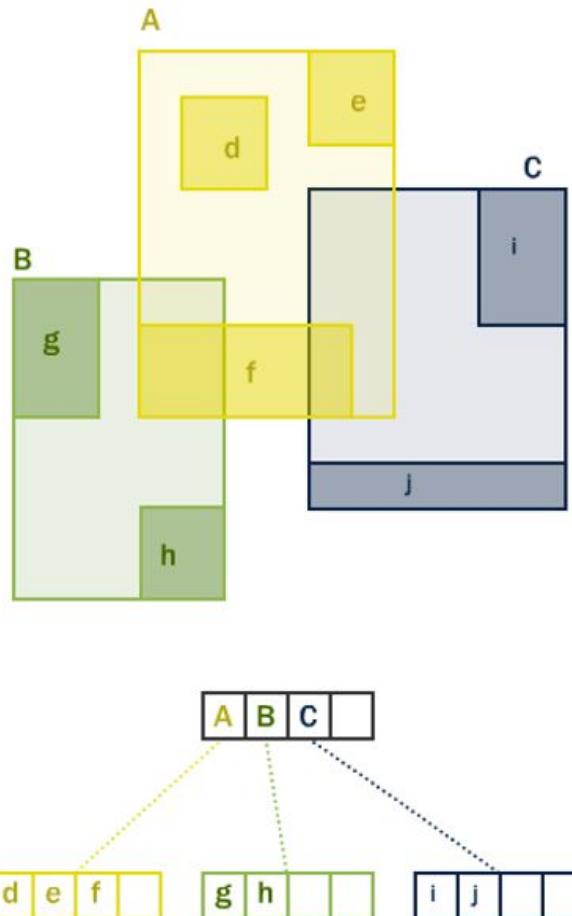
# Spatial Index Internals

Some spatial objects (like the star) are quite large and complex. Comparing complex objects is expensive!

Instead of indexing objects directly, spatial indexes work on the bounding boxes of the objects.

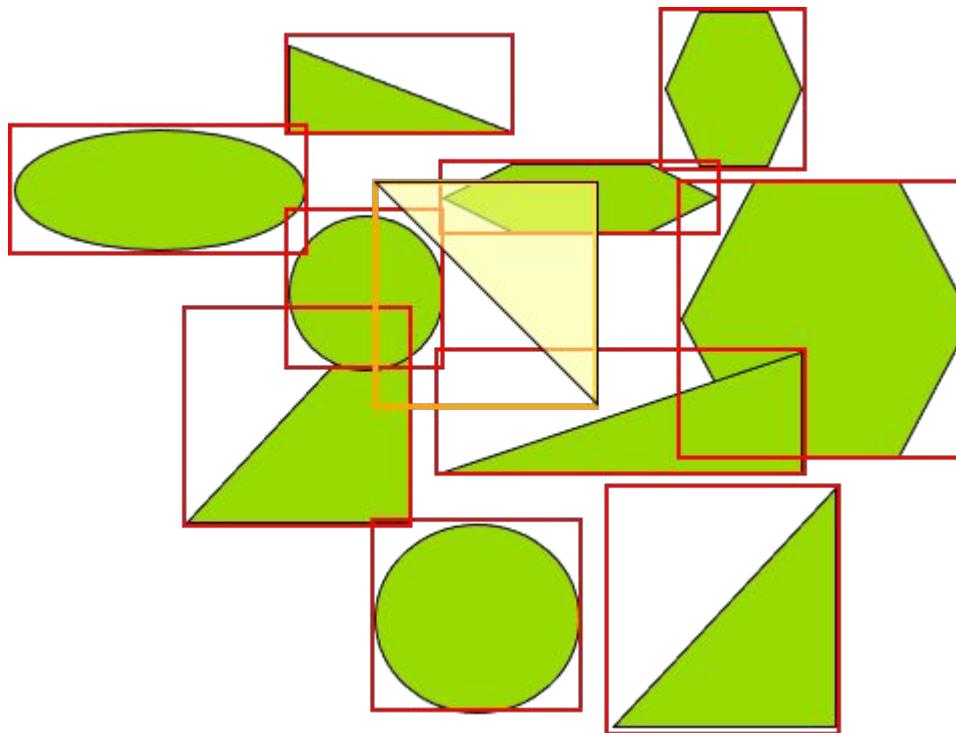
The boxes are of uniform size, and can be compared to determine spatial relationships very quickly.



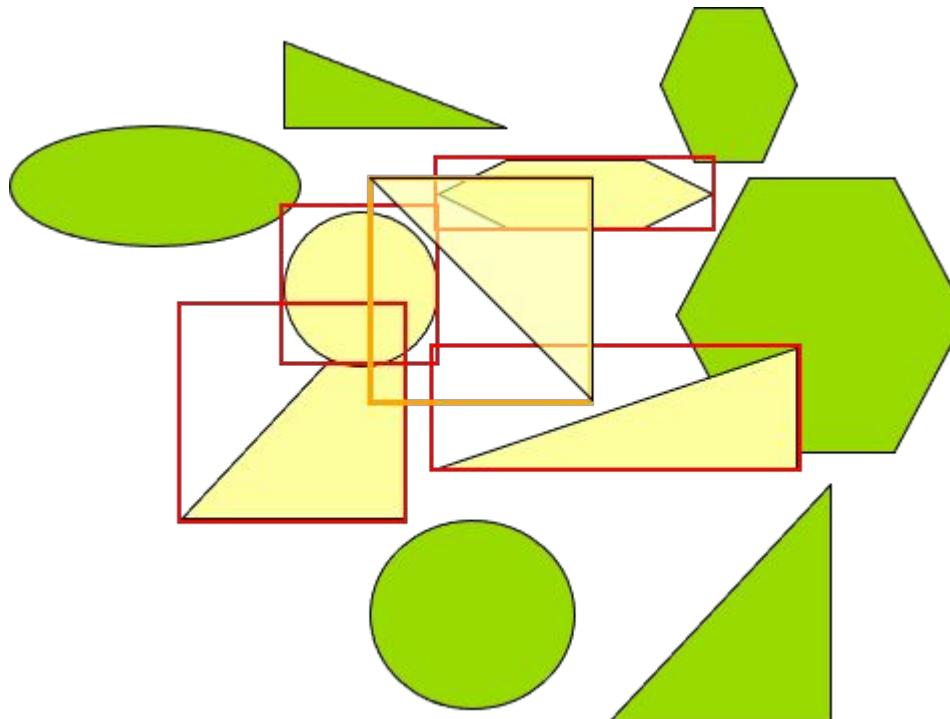


The boxes can be arranged in a hierarchy, so that a query can quickly discard portions of the search space that will not interact with a query box. Depending on the algorithm, different hierarchies can be built. PostGIS uses an “R\*tree” algorithm.

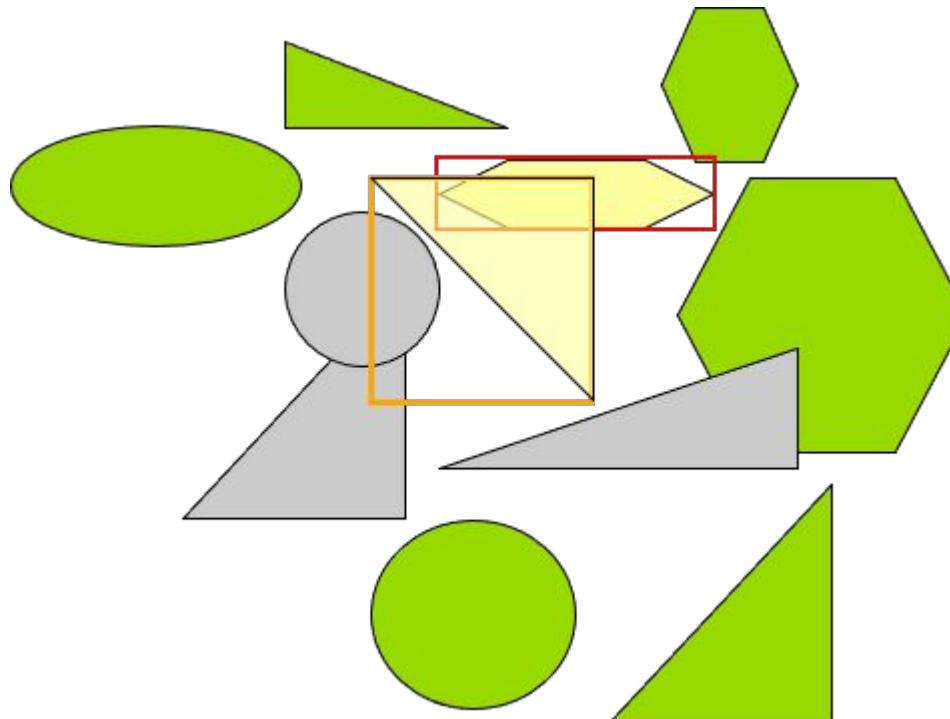
What green objects intersect the yellow query shape?



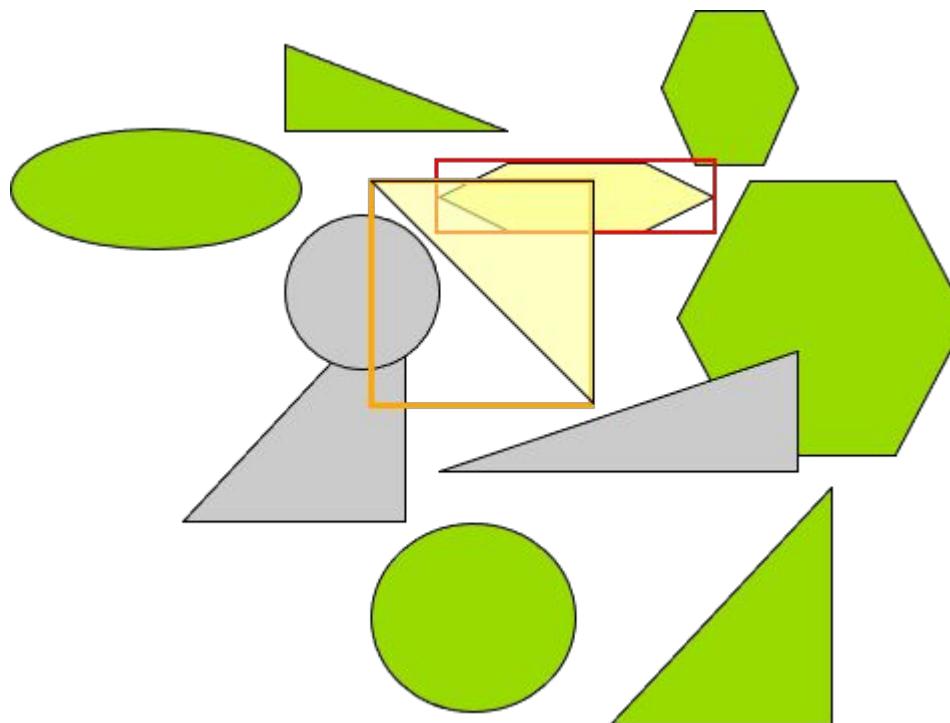
Use **index** to quickly finds the objects with **bounding box intersection**.



Exactly compute relationships in index result to find true intersection.



### Index-only queries



# Index-enabled Spatial Functions

- **ST\_Intersects()**
- **ST\_Contains()**
- **ST\_Within()**
- **ST\_DWithin()**
- ST\_ContainsProperly()
- ST\_CoveredBy()
- ST\_Covers()
- ST\_Overlaps()
- ST\_Crosses()
- ST\_DFullyWithin()
- ST\_3DIntersects()
- ST\_3DDWithin()
- ST\_3DDFullyWithin()
- ST\_LineCrossingDirection()
- ST\_OrderingEquals()
- ST\_Equals()

## Index-only queries

**geom\_a && geom\_b**

The “&&” operator is the “bounding boxes overlap” operator.

It returns “true” when the bounds of the left and right arguments overlap.

Operators like “=” or “>” are symbols that express relationships between the left- and right-hand side arguments. “&&” is just another operator like any other.

## What is the population of the West Village?

```
SELECT Sum(blk.opn_total)
FROM nyc_neighborhoods nh
JOIN nyc_census_blocks blk
ON nh.geom && blk.geom
WHERE nh.name = 'West Village';
```

---

49821

## What is the population of the West Village?

```
SELECT Sum(blk.opn_total)
FROM nyc_neighborhoods nh
JOIN nyc_census_blocks blk
ON ST_Intersects(nh.geom, blk.geom)
WHERE nh.name = 'West Village';
```

---

26718

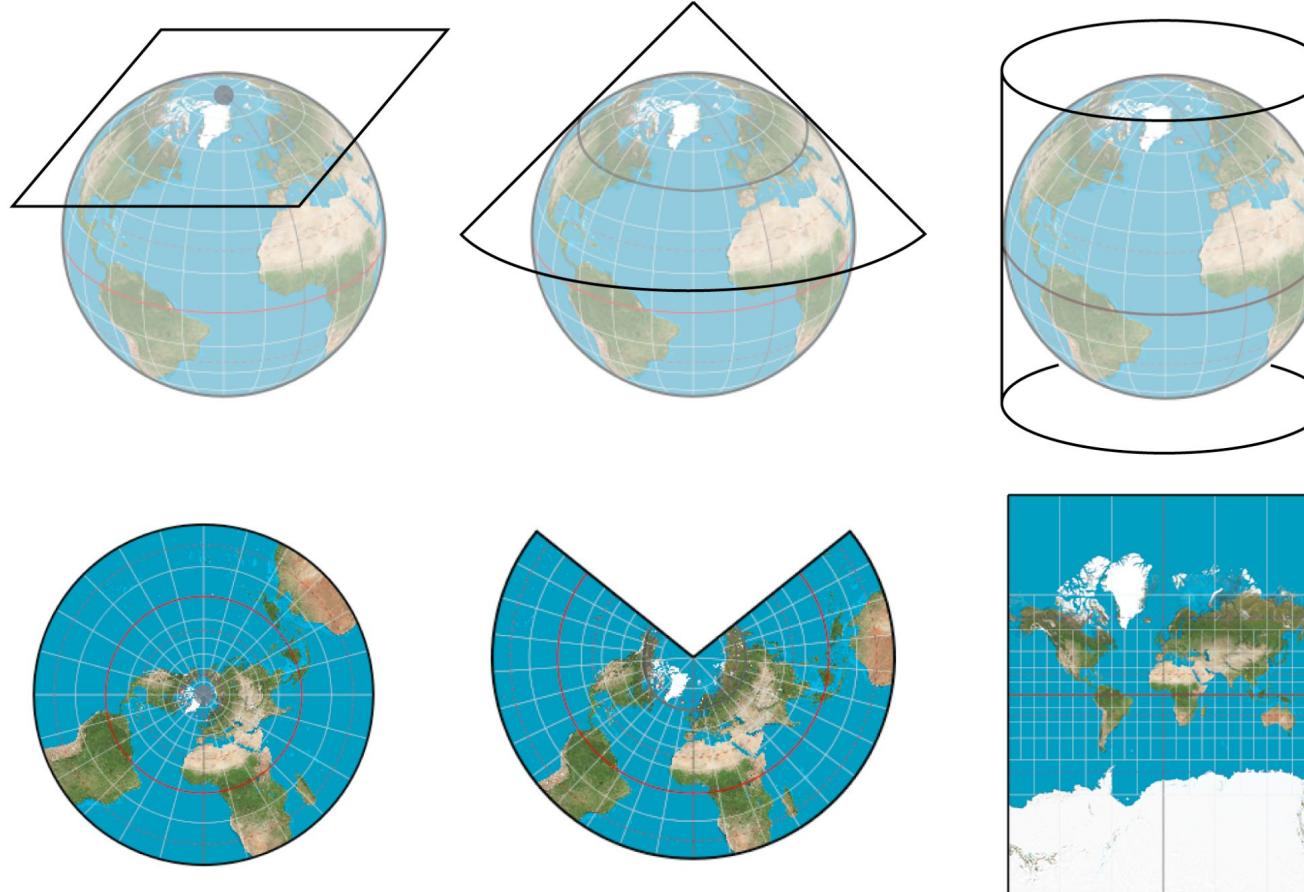
## Spatial Indexes

```
VACUUM ANALYZE nyc_census_blocks;
```

The database planner can only know when to use indexes if the tables have been “analyzed”.

The database executor will run more efficiently if dead tuple bloat has been removed with “vacuum”. Most important on tables with bulk data changes (insert/update/delete).

# **Section 16 - Projecting Data**



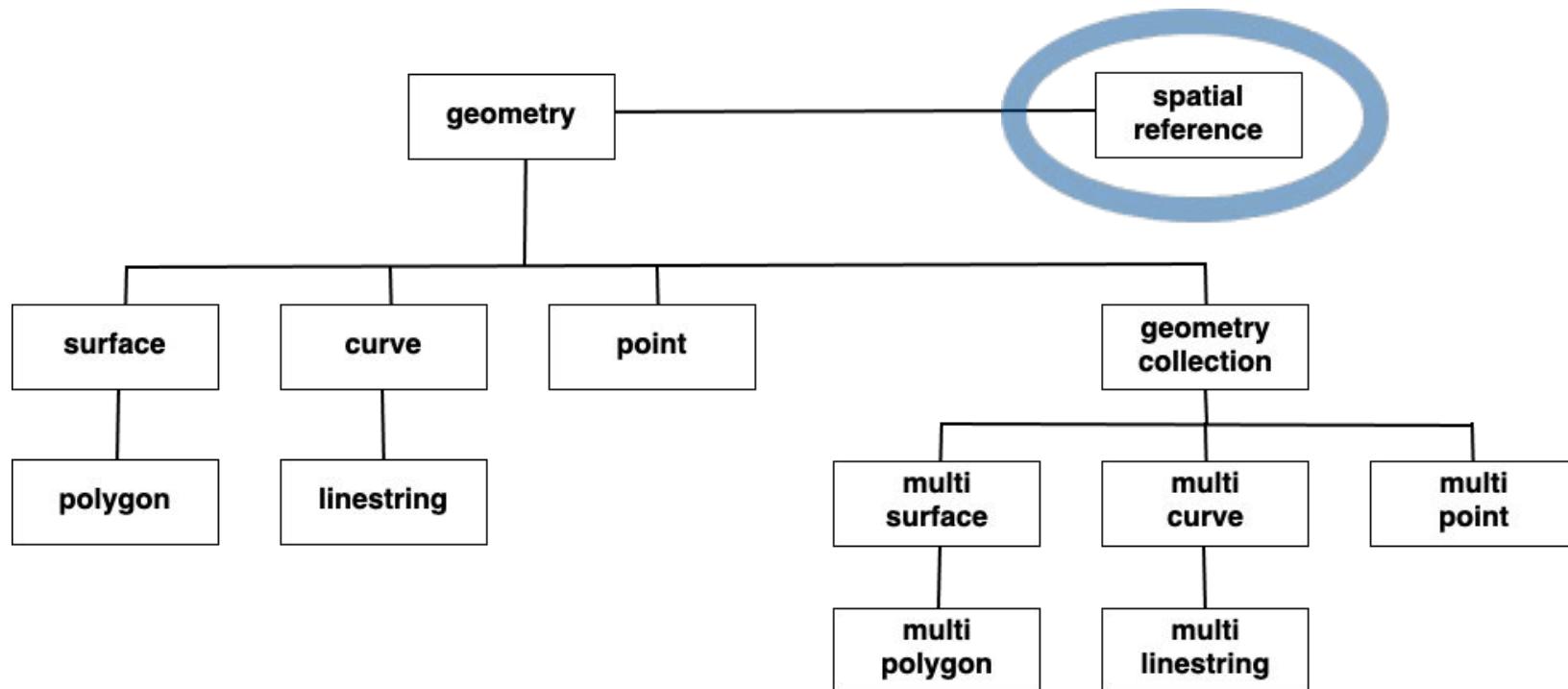
The earth is not flat, and there is no simple way of putting it down on a flat paper map (or computer screen), so people have come up with all sorts of ingenious solutions, each with pros and cons.

$$f(\theta, \Phi) \rightarrow (x, y)$$

Forward projection converts spherical coordinates (longitude, latitude) to cartesian coordinates (x and y)

$$f^{-1}(x, y) \rightarrow (\theta, \Phi)$$

Inverse projection converts cartesian coordinates (x, y) to spherical coordinates (longitude, latitude)



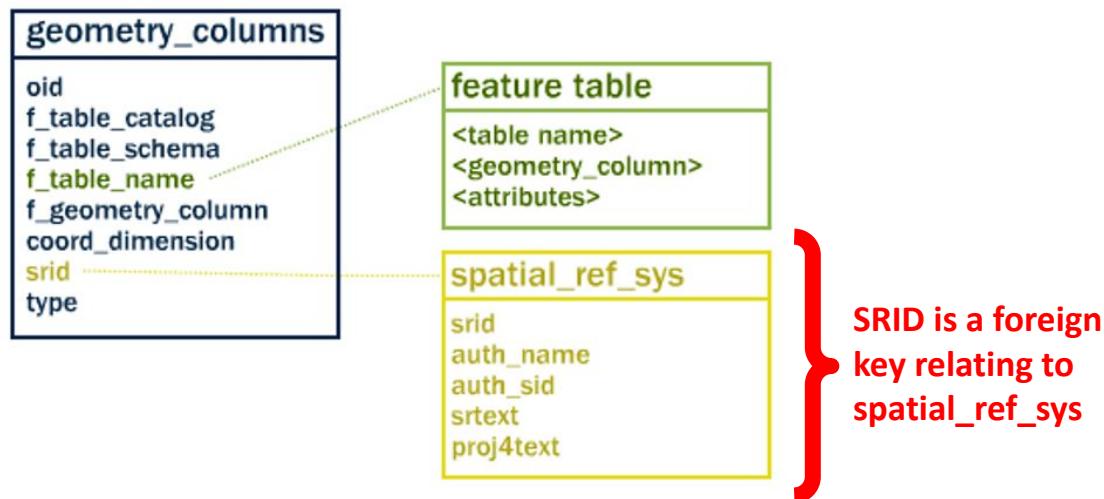
## What is the SRID of our subways?

```
SELECT ST_SRID(geom)  
FROM nyc_subway_stations  
LIMIT 1;
```

---

26918

# What does SRID 26918 mean though?



## What does SRID 26918 mean though?

```
SELECT srtext  
FROM spatial_ref_sys  
WHERE srid = 26918;
```

Also, see: <https://epsg.io/26918>

# What does SRID 26918 mean though?

```
PROJCS["NAD83 / UTM zone 18N",
GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
        SPHEROID["GRS 1980",6378137,298.257222101],
        TOWGS84[0,0,0,0,0,0,0],
        AUTHORITY["EPSG","6269"]]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4269"]],
PROJECTION["Transverse_Mercator"],
PARAMETER["latitude_of_origin",0],
PARAMETER["central_meridian",-75],
PARAMETER["scale_factor",0.9996],
PARAMETER["false_easting",500000],
PARAMETER["false_northing",0],
UNIT["metre",1],
AXIS["Easting",EAST],
AXIS["Northing",NORTH]]
```

**What are coordinates of the “Broad St” subway station in geographic?**

```
SELECT  
    ST_AsText(ST_Transform(geom,4326))  
FROM nyc_subway_stations  
WHERE name = 'Broad St';
```

---

POINT(-74.0106714 40.7071048)

# Section 17 - Projection Exercises

GEOMETRY FUNCTIONS	
ST_Area(geometry)	ST_SRID(geometry)
ST_AsBinary(geometry)	ST_StartPoint(geomtry)
ST_AsGML(geometry)	ST_SymDifference(geometry,geometry)
ST_AsGeoJSON(geometry)	ST_Touches(geometry, geometry)
ST_AsKML(geometry)	ST_Union(geometry, geometry)
ST_AsSVG(geometry)	ST_Union(geometry, geometry))
ST_AsText(geometry)	ST_Within(geometry,geometry)
ST_Buffer(geometry,distance)	ST_Xpoint()
ST_Centroid(geometry)	ST_Ypoint()
ST_Contains(geometry,geometry)	ST_Zipoint()
ST_CoveredBy(geometry,geometry)	ST_M(point)
ST_Crosses(geometry,geometry)	
ST_Difference(geometry,geometry)	
ST_DWithin(geometry,geometry,radius)	
ST_Disjoint(geometry,geometry)	
ST_Distances(geometry,geometry)	
ST_EndPoint(geometry)	
ST_Equals(geometry,geometry)	
ST_ExteriorRing(polygon)	
ST_GeomFromGML(text)	
ST_GeomFromKML(text)	
ST_GeomFromText(text)	
ST_GeomFromWKB(bytse)	
ST_GeometryN(collection,n)	
ST_GeometryType(geometry)	
ST_Intersecting(polygons)	
ST_Intersection(geometry,geometry)	
ST_Intersection(geometry,geometry)	
ST_IsValid(geometry)	
ST_Length(linestrings)	
ST_MakePoint(double,double)	
ST_NPoints(geometry)	
ST_NRings(polygons)	
ST_NumberGeometries(collection)	
ST_OrderedEqual(geometry,geometry)	
ST_Overlaps(geometry,geometry)	
ST_Perimeter(polygons)	
ST_PlanarSurface(geometry)	
ST_Relate(geometry,geometry)	
ST_Reverse(geometry)	
ST_Simplify(geometry)	

GEOGRAPHY FUNCTIONS	
ST_Area(geography)	ST_AsBinary(geography)
ST_AsGML(geography)	ST_AsGeoJSON(geography)
ST_AsGeoBBox(geography)	ST_AsKML(geography)
ST_AsSVG(geography)	ST_AsText(geography)
ST_AsTextFromWKT(geography)	ST_CoveredBy(geography,geography)
ST_CoveredBy(geography,geography)	ST_Covers(geography,geography)
ST_DWithin(geography,geography, float)	ST_DWithin(geography,geography, float)
ST_Distance(geography,geography)	ST_Intersects(geography,geography)
ST_GeomFromWKB(bytse)	ST_Intersects(geography,geography)
ST_GeographyFromText(text)	ST_Intersects(geography,geography)
ST_Interfering(polygons)	ST_Length(geography)
ST_NumberGeometries(collection)	
ST_OrderedEqual(geometry,geometry)	
ST_Overlaps(geometry,geometry)	
ST_Perimeter(polygons)	

nyc_census_blocks (36592 records)	
blkid	A 15-digit code that uniquely identifies every census block. Eg: 3600500010900
popu_total	Total number of people in the census block
popu_white	Number of people self-identifying as "White" in the block
popu_black	Number of people self-identifying as "Black" in the block
popu_native	Number of people self-identifying as "Native American" in the block
popu_asian	Number of people self-identifying as "Asian" in the block
popu_other	Number of people self-identifying with other categories in the block
house_total	Number of housing units in the block
house_own	Number of owner-occupied housing units in the block
house_rent	Number of renter-occupied housing units in the block
boroughname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the block

nyc_neighborhoods (129 records)	
name	Name of the neighborhood
boroughname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the neighborhood

nyc_streets (19991 records)	
name	Name of the street
oneway	Is the street one-way? "yes" = yes, "no" = no
type	Road type. Eg: primary, secondary, residential, motorway
the_geom	Linear centerline of the street

nyc_subway_stations (491 records)	
name	Name of the station
borough	Name of 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" = no
the_geom	Point location of the station

nyc_census_socidata	
tractid	An 11-digit code for every census tract. Eg: 3600500010100
tracti_total	Number of workers in the tract
tracti_public	Number of workers in the tract who take public transit
tracti_private	Number of workers in the tract who use private automobiles/ motorcycles
tracti_other	Number of workers in the tract who use other forms like walking/ biking
transit_time_mins	Total minutes spent in transit by all workers in the tract (minutes)
family_size	Number of families in the tract
family_income_mean	Mean family income in the tract (dollars)
family_income_aggregate	Total income for all families in the tract (dollars)
edu_total	Number of people with educational history
edu_no_highschool_dipl	Number of people with no highschool diploma
edu_highschool_dipl	Number of people with highschool diplomas and no further education
edu_college_dipl	Number of people with college diplomas and no further education
edu_graduate_dipl	Number of people with graduate school diplomas

**What is the length of all streets in New York, as measured in UTM 18?**

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

---

**10418904.7172**

## What is the WKT definition of SRID 2831?

```
SELECT srtext  
FROM spatial_ref_sys  
WHERE SRID = 2831;
```

---

PROJCS["NAD83(HARN) / New York Long Island",

...

**What is the length of all streets in New York, as measured in SRID 2831 (Stateplane Long Island)?**

```
SELECT Sum(ST_Length(  
    ST_Transform(geom, 2831)  
)  
FROM nyc_streets;
```

---

10421993.706374

## How many streets cross the 74th meridian?

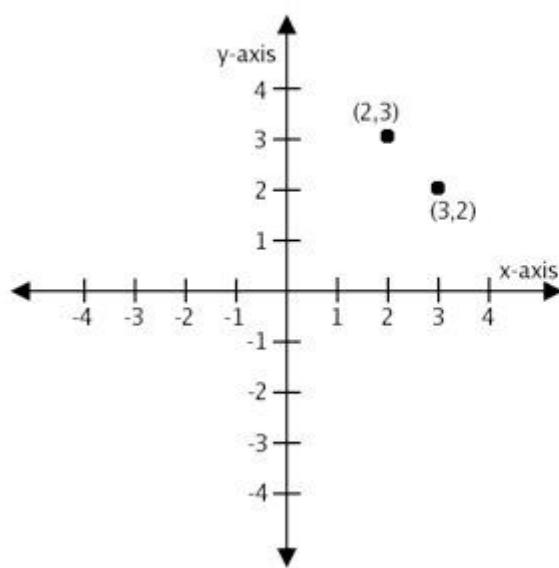
```
SELECT Count(*)
FROM nyc_streets
WHERE ST_Intersects(
    ST_Transform(geom, 4326),
    'SRID=4326;LINESTRING(-74 20, -74 60)'
);
```

---

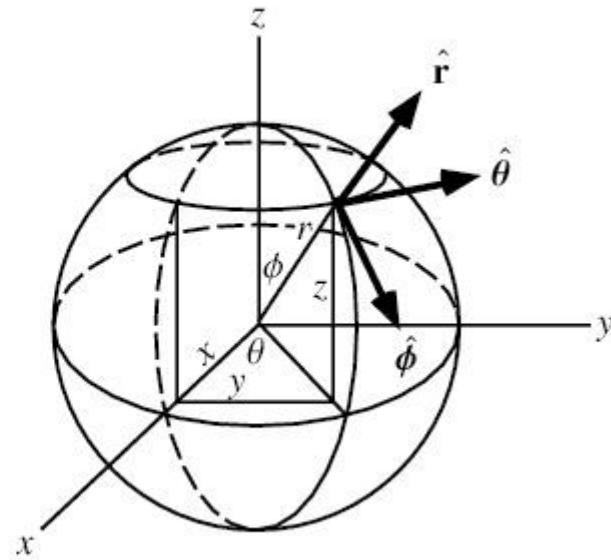
# **Section 18 - Geography**

# Geographic Coordinate Systems

Cartesian



Spherical



## What is the distance between Los Angeles and Paris using **ST\_Distance(geometry, geometry)**?

```
SELECT ST_Distance(  
    -- Los Angeles (LAX)  
    'SRID=4326;POINT(-118.4079 33.9434)' ::geometry,  
    -- Paris (CDG)  
    'SRID=4326;POINT(2.5559 49.0083)' ::geometry  
) ;
```

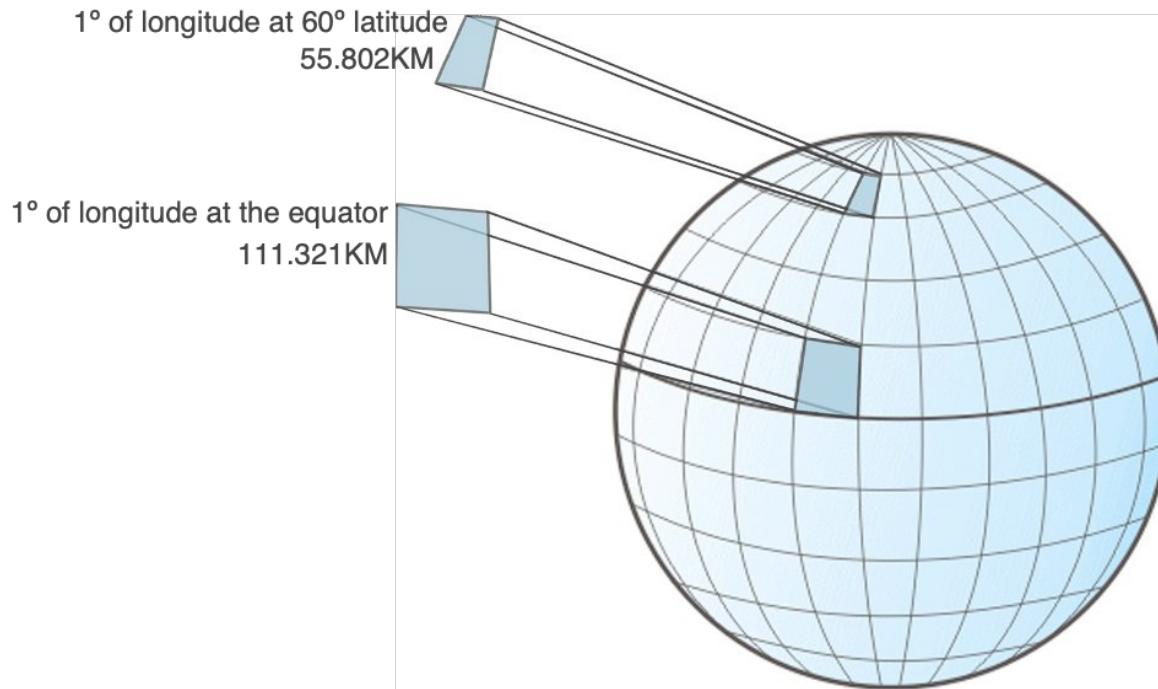
---

121.898285970107

## Section 18 - Geography



**Degrees are not units of distance**  
**Degrees are not units of area**



## What is the distance between Los Angeles and Paris using **ST\_Distance(geography, geography)**?

```
SELECT ST_Distance(  
    -- Los Angeles (LAX)  
    'SRID=4326;POINT(-118.4079 33.9434)' ::geography,  
    -- Paris (CDG)  
    'SRID=4326;POINT(2.5559 49.0083)' ::geography  
) ;
```

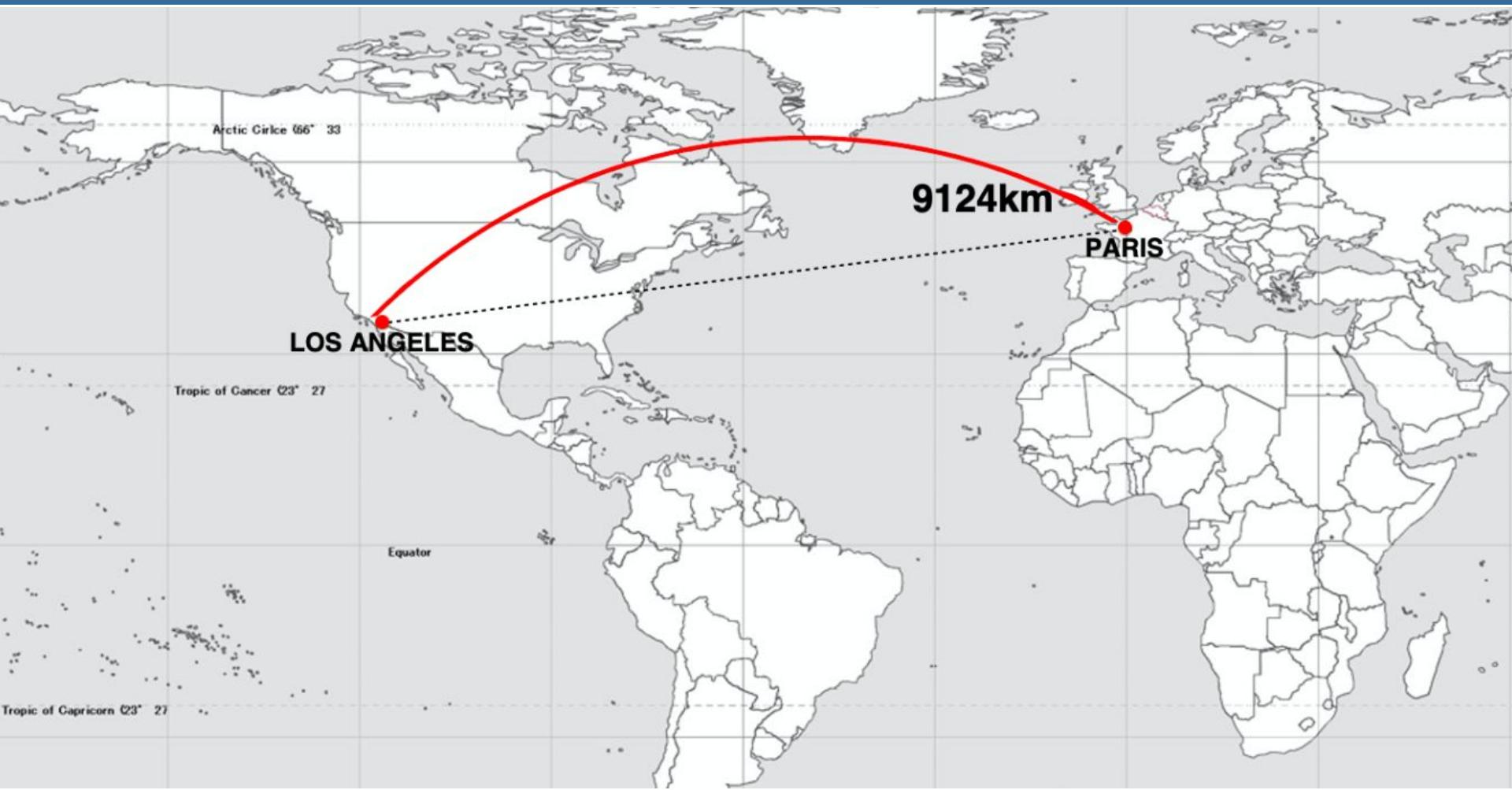
---

9124665.27317673

## Section 18 - Geography



PostGIS



# How close will a flight from Los Angeles to Paris come to Iceland?

```
SELECT ST_Distance(
    -- LAX-CDG
    'SRID=4326;LINESTRING(
        -118.4079 33.9434,
        2.5559 49.0083)' ::geography,
    -- Iceland
    'SRID=4326;POINT(-21.8628 64.1286)' ::geography
);
```

---

531773.75711106

## Section 18 - Geography

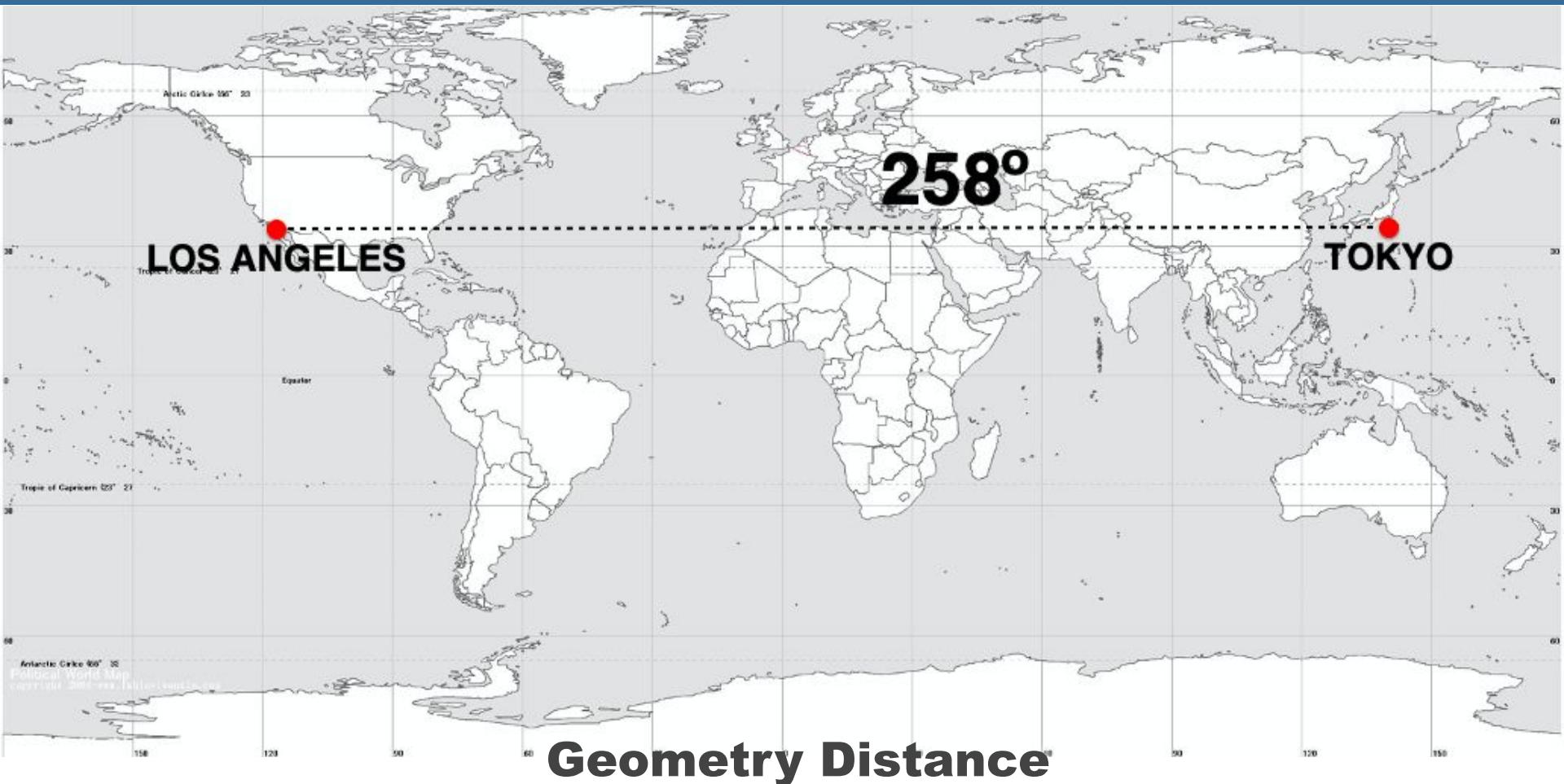


# What is the shortest great-circle route from Los Angeles to Tokyo?

```
SELECT ST_Distance(  
    'SRID=4326;POINT( -118.408 33.943 )' ::geometry, -- LAX  
    'SRID=4326;POINT( 139.733 35.567 )' ::geometry) -- NRT  
        AS geometry_distance,  
ST_Distance(  
    'POINT(-118.408 33.943)' ::geography, -- LAX  
    'POINT( 139.733 35.567 )' ::geography) -- NRT  
        AS geography_distance;
```

---

geometry\_distance: 258.14610835  
geography\_distance: 8833973.30246194





## Using Geography - Casting

```
CREATE TABLE nyc_subway_stations_geog AS
SELECT
    ST_Transform(geom, 4326)::geography AS geog,
    name,
    routes
FROM nyc_subway_stations;
```

## Using Geography - Indexing

```
CREATE INDEX nyc_subway_stations_geog_gix
ON nyc_subway_stations_geog
USING GIST (geog);
```

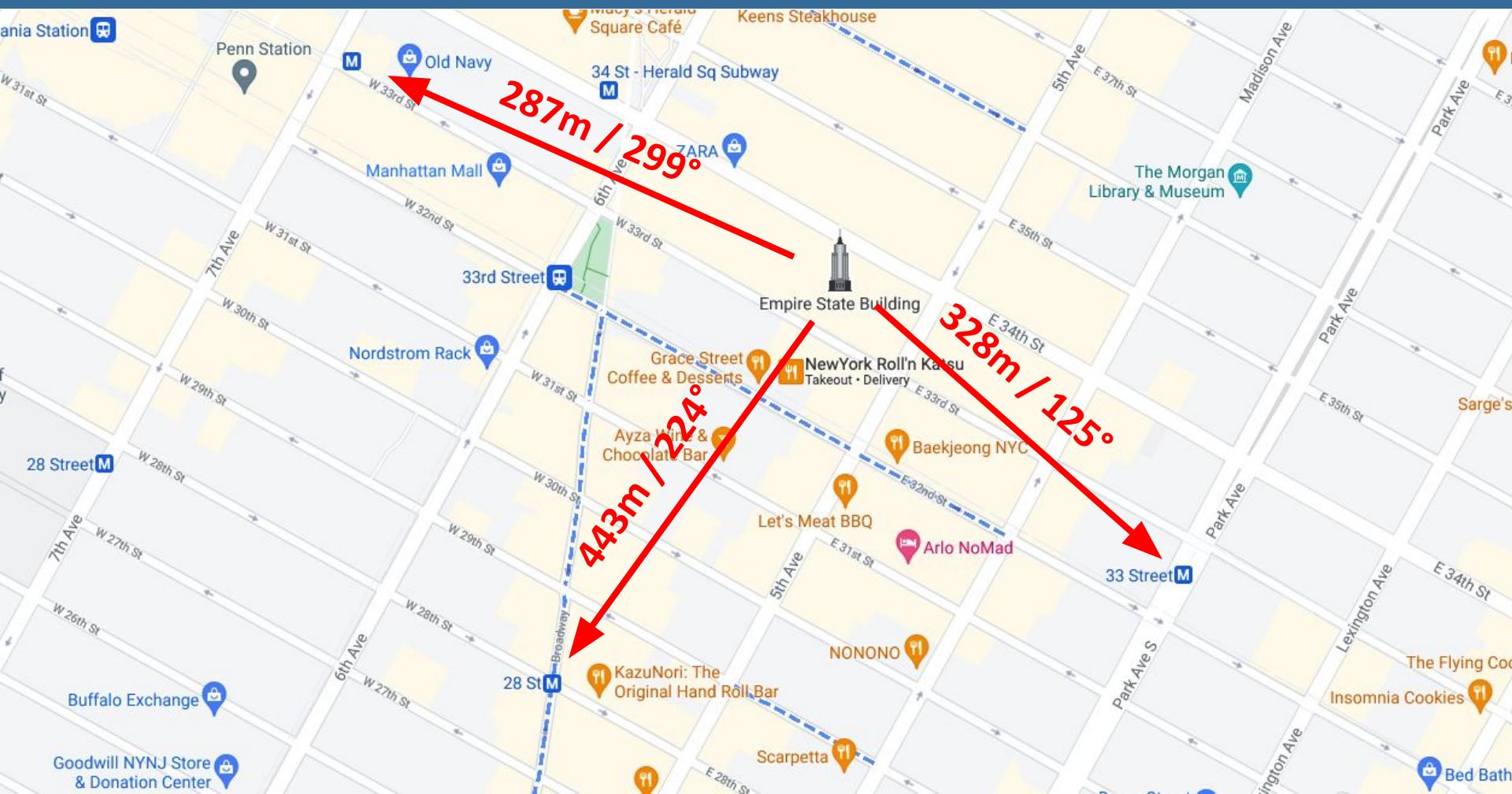
# Using Geography - Querying

```
WITH empire_state_building AS (
    SELECT 'POINT(-73.98501 40.74812)'::geography AS geom
)
SELECT name,
    ST_Distance(esb.geom, ss.geom) AS distance,
    degrees(ST_Azimuth(esb.geom, ss.geom)) AS direction
FROM nyc_subway_stations_geog ss,
    empire_state_building esb
WHERE ST_DWithin(ss.geom, esb.geom, 500);
```

## Section 18 - Geography



PostGIS



# Using Geography - From Scratch

```
CREATE TABLE airports (
    code VARCHAR(3),
    geog GEOGRAPHY(Point)
);
```

```
INSERT INTO airports
VALUES ('LAX', 'POINT(-118.4079 33.9434)');
INSERT INTO airports
VALUES ('CDG', 'POINT(2.5559 49.0083)');
INSERT INTO airports
VALUES ('KEF', 'POINT(-22.6056 63.9850));
```

# Using Geography - From Scratch

```
SELECT * FROM geography_columns;
```

f_table_name	f_geography_column	srid	type
nyc_subway_stations_geog	geog	0	Geometry
airports	geog	4326	Point

## Casting to Geometry

```
SELECT  
    code,  
    ST_X(geog::geometry) AS longitude  
FROM airports;
```



The “::” syntax tells PostgreSQL to attempt to coerce the data into the new data type, if there is an available path.

# 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)`

# Geography is the Magic Solution?



The complexity of dealing with planar projections (choosing one, getting used to it) drives some users to fixate on the **geography** type as a simple cure-all.

However:

- Not all functions in geography have native on-the-sphere implementations yet.
- The computational cost of geography compared to geometry is quite high.

# geography distance

```
double R = 6371000; /* meters */
double d_lat = lat2-lat1; /* radians */
double d_lon = lon2-lon1; /* radians */
double sin_lat = sin(d_lat/2);
double sin_lon = sin(d_lon/2);
double a = sin_lat * sin_lat +
           cos(lat1) * cos(lat2) *
           sin_lon * sin_lon;
double c = 2 * atan2(sqrt(a),
                     sqrt(1-a));
double d = R * c;
```

# geometry distance

```
double dx = x2 - x1;
double dy = y2 - y1;
double d2 = dx * dx +
            dy * dy;
double d = sqrt(d2);
```

# Section 19 - Geography Exercises

GEOMETRY FUNCTIONS	
ST_Area(geometry)	ST_SRID(geometry)
ST_AsBinary(geometry)	ST_StartPoint(geometries)
ST_AsGML(geometry)	ST_SymDifference(geometry,geometry)
ST_AsGeoJSON(geometry)	ST_Touches(geometry, geometry)
ST_AsKML(geometry)	ST_Union(geometry, geometry)
ST_AsSVG(geometry)	ST_Union(geometry, geometry))
ST_AsText(geometry)	ST_Within(geometry,geometry)
ST_Buffer(geometry,distance)	ST_Xpoint()
ST_Centroid(geometry)	ST_Ypoint()
ST_Contains(geometry,geometry)	ST_ZipPoint()
ST_ConvexHull(geometry)	ST_M(point)
ST_Crosses(geometry,geometry)	
ST_Difference(geometry,geometry)	
ST_DWithin(geometry,geometry,radius)	
ST_Disjoint(geometry,geometry)	
ST_Distances(geometry,geometry)	
ST_EndPoint(geometry)	
ST_Equals(geometry,geometry)	
ST_ExteriorRing(polygons)	
ST_GeomFromGML(text)	
ST_GeomFromKML(text)	
ST_GeomFromText(text)	
ST_GeomFromWKB(bytess)	
ST_GeometryN(collection,n)	
ST_GeometryType(geometry)	
ST_IntersectingN(polygons)	
ST_Intersects(geometry,geometry)	
ST_Intersection(geometry,geometry)	
ST_IsValid(geometry)	
ST_IsValidReason(geometry)	
ST_Length(linestrings)	
ST_MakePoint(double,double)	
ST_NPoints(geometry)	
ST_NRings(polygons)	
ST_NumberGeometries(collection)	
ST_OrderedEqual(geometry,geometry)	
ST_Overlaps(geometry,geometry)	
ST_Perimeter(polygons)	
ST_Planarize(geometry)	
ST_Polygonize(geometry)	
ST_Relate(geometry,geometry)	
ST_Reverse(geometry)	
ST_Simplify(geometry)	

GEOGRAPHY FUNCTIONS	
ST_Area(geography)	ST_AsBinary(geography)
ST_AsGML(geography)	ST_AsKML(geography)
ST_AsGeoJSON(geography)	ST_AsMVT(geography)
ST_AsSVG(geography)	ST_AsText(geography)
ST_Boundary(geography)	ST_CoveredBy(geography,geography)
ST_CoveredBy(geography,geography)	ST_Covers(geography,geography)
ST_DWithin(geography,geography,foot)	ST_DWithin(geography,geography,foot)
ST_Distance(geography,geography)	ST_Distance(geography,geography)
ST_GeomFromWKB(bytess)	ST_GeomFromText(text)
ST_InterpolateCoordinate(geography)	ST_Intersects(geography,geography)
ST_Intersects(geography,geography)	ST_Intersects(geography,geography)
ST_Length(geography)	ST_Length(geography)

nyc_census_blocks (36592 records)	
blkid	A 15-digit code that uniquely identifies every census block. Eg: 36005000109000
popu_total	Total number of people in the census block
popu_white	Number of people self-identifying as "White" in the block
popu_black	Number of people self-identifying as "Black" in the block
popu_native	Number of people self-identifying as "Native American" in the block
popu_asian	Number of people self-identifying as "Asian" in the block
popu_other	Number of people self-identifying with other categories in the block
house_total	Number of housing units in the block
house_own	Number of owner-occupied housing units in the block
house_rent	Number of renter-occupied housing units in the block
boroughname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the block

nyc_neighborhoods (129 records)	
name	Name of the neighborhood
boroughname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the neighborhood

nyc_streets (19991 records)	
name	Name of the street
oneway	Is the street one-way? "yes" = yes, "no" = no
type	Road type. Eg: primary, secondary, residential, motorway
the_geom	Linear centerline of the street

nyc_subway_stations (491 records)	
name	Name of the station
borough	Name of 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" = no
the_geom	Point location of the station

nyc_census_socidata	
tractid	An 11-digit code for every census tract. Eg: 3600500010100
tracti_total	Number of workers in the tract
transit_public	Number of workers in the tract who take public transit
transit_private	Number of workers in the tract who use private automobiles/ motorcycles
transit_other	Number of workers in the tract who use other forms like walking/ biking
transit_time_mins	Total minutes spent in transit by all workers in the tract (minutes)
family_size	Number of families in the tract
family_income_median	Median family income in the tract (dollars)
family_income_aggregate	Total income for all families in the tract (dollars)
edu_total	Number of people with educational history
edu_no_highschool_dipl	Number of people with no highschool diploma
edu_highschool_dipl	Number of people with highschool diplomas and no further education
edu_college_dipl	Number of people with college diplomas and no further education
edu_graduate_dipl	Number of people with graduate school diplomas

**How far is New York from Seattle? What are the units of the answer?**

```
SELECT ST_Distance(  
    'POINT(-74.0064 40.7142)'::geography,  
    'POINT(-122.3331 47.6097)'::geography  
)
```

---

3875538.57141352

**What is the total length of all streets in New York,  
calculated on the spheroid?**

```
SELECT Sum(  
    ST_Length(ST_Transform(geom, 4326)::geography)  
)  
FROM nyc_streets;
```

---

**10421999.666**

**What is the total length of all streets in New York,  
calculated on the plane?**

```
SELECT Sum(  
    geom  
)  
FROM nyc_streets;
```

---

**10418904.717**

# How good are projections?

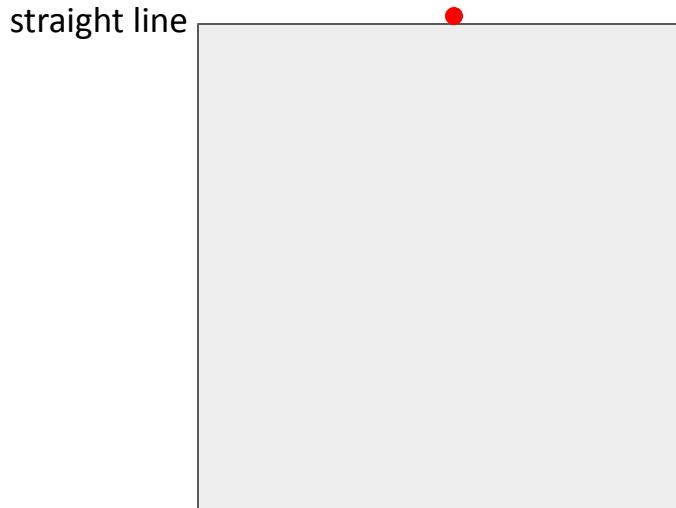
Projection	Total Length	Difference
Geography	10421999.666032	0 / 0%
Long Island State Plane	10421993.706374	-5.9m / 0.00006%
UTM Zone 18	10418904.717199	-3095m / 0.03%

**Does 'POINT(1 2.0001)' intersect with  
'POLYGON((0 0, 0 2, 2 2, 2 0, 0 0))' in geography?  
In geometry? Why the difference?**

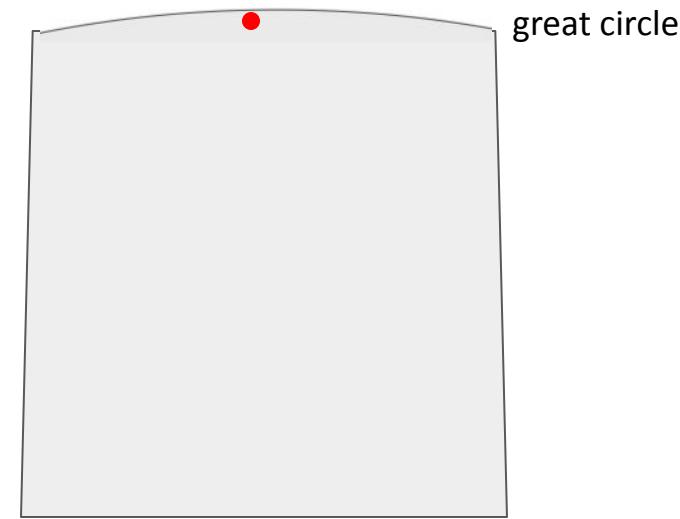
```
SELECT ST_Intersects(  
    'POINT(1 2.0001)'::geography,  
    'POLYGON((0 0,0 2,2 2,2 0,0 0))'::geography  
)
```

```
SELECT ST_Intersects(  
    'POINT(1 2.0001)'::geometry,  
    'POLYGON((0 0,0 2,2 2,2 0,0 0))'::geometry  
)
```

**Does 'POINT(1 2.0001)' intersect with  
'POLYGON((0 0, 0 2, 2 2, 2 0, 0 0))' in geography?  
In geometry? Why the difference?**



geometry



geography

# **Section 20 - Geometry Constructing Functions**

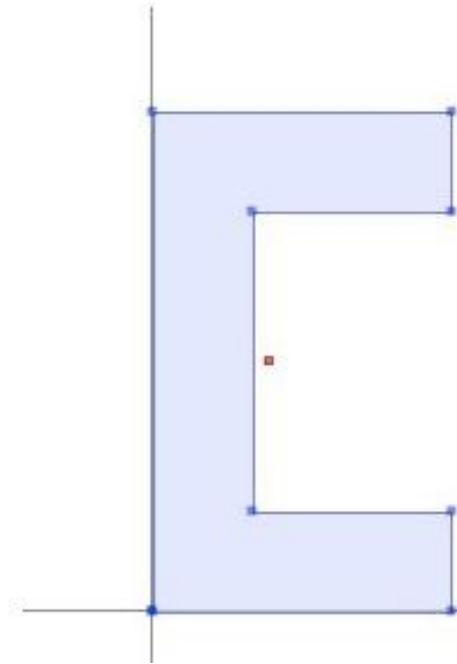
# Functions so far...

- Analysis
  - `ST_Length(geometry)` → float
  - `ST_Area(geometry)` → float
- Conversion
  - `ST_AsText(geometry)` → text
  - `ST_AsGML(geometry)` → text
- Retrieval
  - `ST_RingN(geometry,n)` → geometry
- Comparison
  - `ST_Contains(geometry,geometry)` → boolean

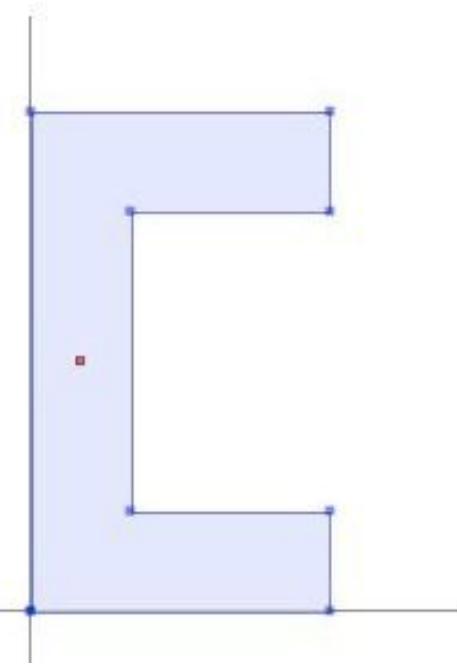
# Geometry constructing functions!

- **ST\_Buffer(geometry)** → geometry
- **ST\_Centroid(geometry)** → geometry
- **ST\_Intersection(geometry, geometry)** → geometry
- **ST\_Union(geometry[])** → geometry
- **ST\_Collect(geometry[])** → geometry

### **ST\_Centroid**



### **ST\_PointOnSurface**



# ST\_Buffer



Buffering a point



Buffering a multipoint

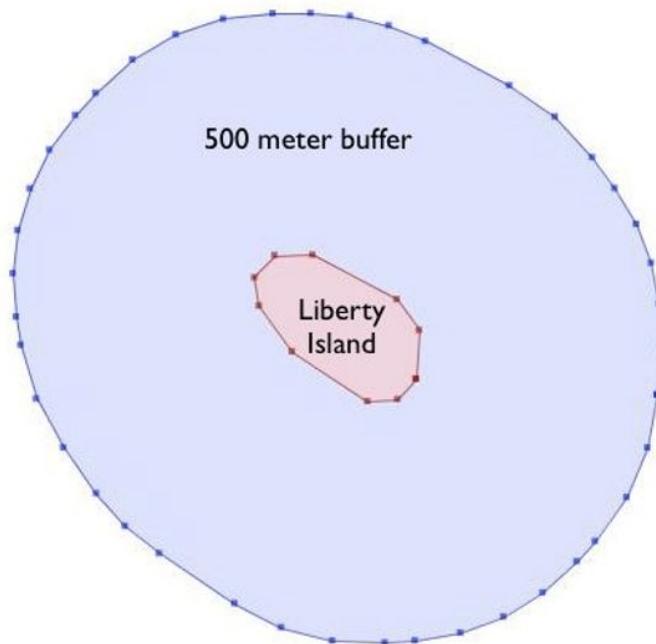


Buffering a linestring



Buffering a polygon  
with one interior ring

**“What would a **500 meter** marine traffic zone around **Liberty Island** look like?”**

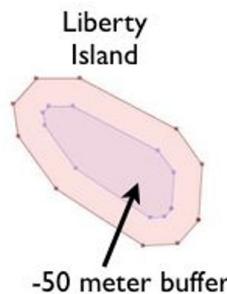


**“What would a **500 meter** marine traffic zone around **Liberty Island** look like?”**

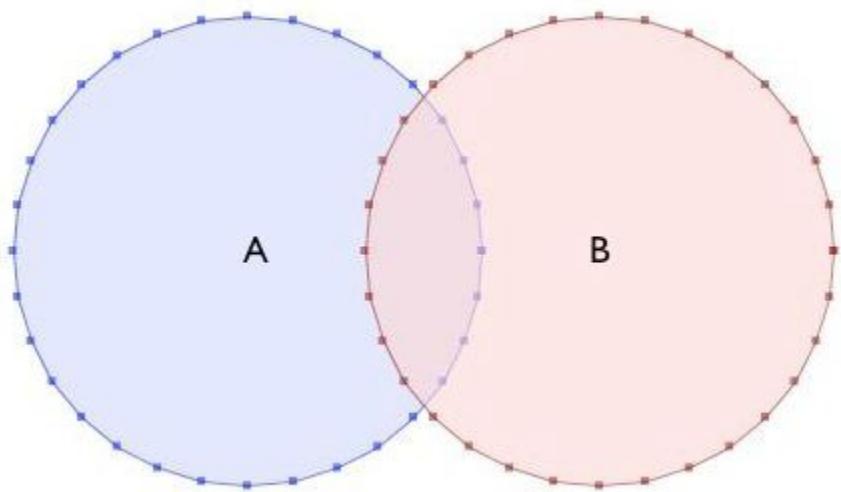
```
-- New table with a Liberty Island  
-- 500m buffer zone
```

```
CREATE TABLE liberty_island_zone AS  
SELECT  
    ST_Buffer(geom, 500)::Geometry(Polygon,26918)  
AS geom  
FROM nyc_census_blocks  
WHERE blkid = '360610001001001';
```

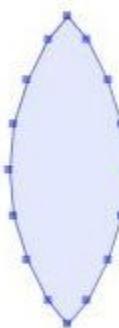
**“What would a **negative 50 meter** marine traffic zone around **Liberty Island** look like?”**



# **ST\_Intersection(A, B)**

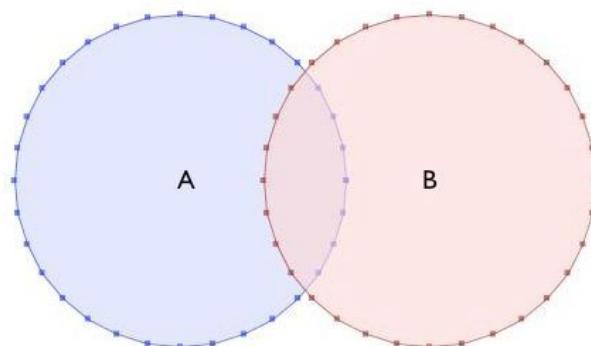


**ST\_Intersection(A,B)**



**“What is the area these two circles have in common?”**

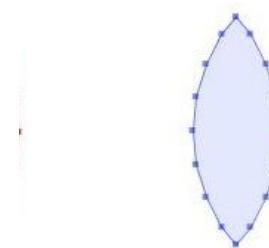
```
SELECT ST_AsText(ST_Intersection(  
    ST_Buffer('POINT(0 0)', 2),  
    ST_Buffer('POINT(3 0)', 2)  
));
```



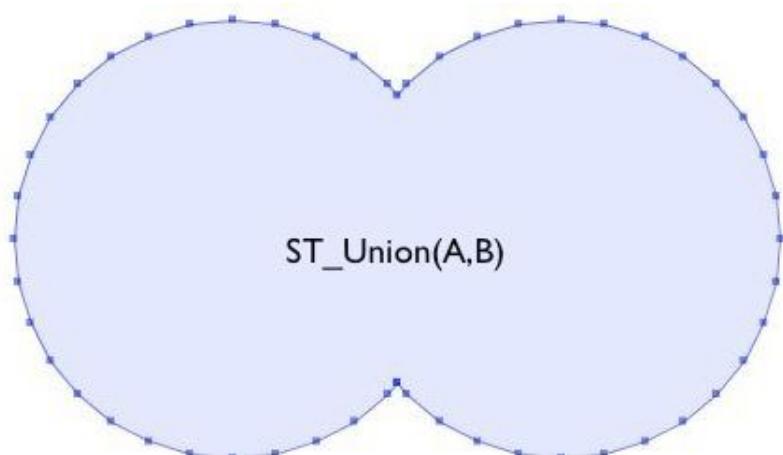
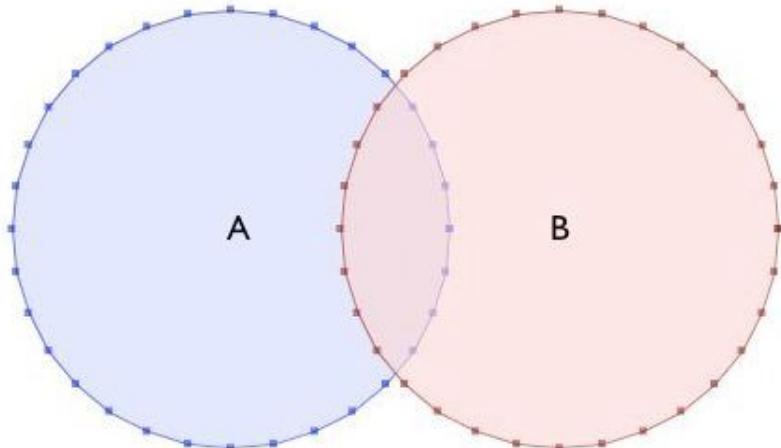
# “What is the area these two circles have in common?”

```
POLYGON((  
    2 0,  
    1.96157056080646 -0.390180644032256,  
    1.84775906502257 -0.765366864730179,  
    1.66293922460509 -1.1111404660392,  
    1.5 -1.30968248567708,  
    1.33706077539491 -1.11114046603921,  
    1.15224093497743 -0.765366864730185,  
    1.03842943919354 -0.390180644032262,  
    1 -6.46217829773035e-15,  
    1.03842943919354 0.39018064403225,  
    1.15224093497742 0.765366864730173,  
    1.33706077539491 1.1111404660392,  
    1.5 1.30968248567708,  
    1.66293922460509 1.11114046603921,  
    1.84775906502257 0.765366864730184,  
    1.96157056080646 0.390180644032261,  
    2 0))
```

ST\_Intersection(A,B)



### **ST\_Union(A, B)**



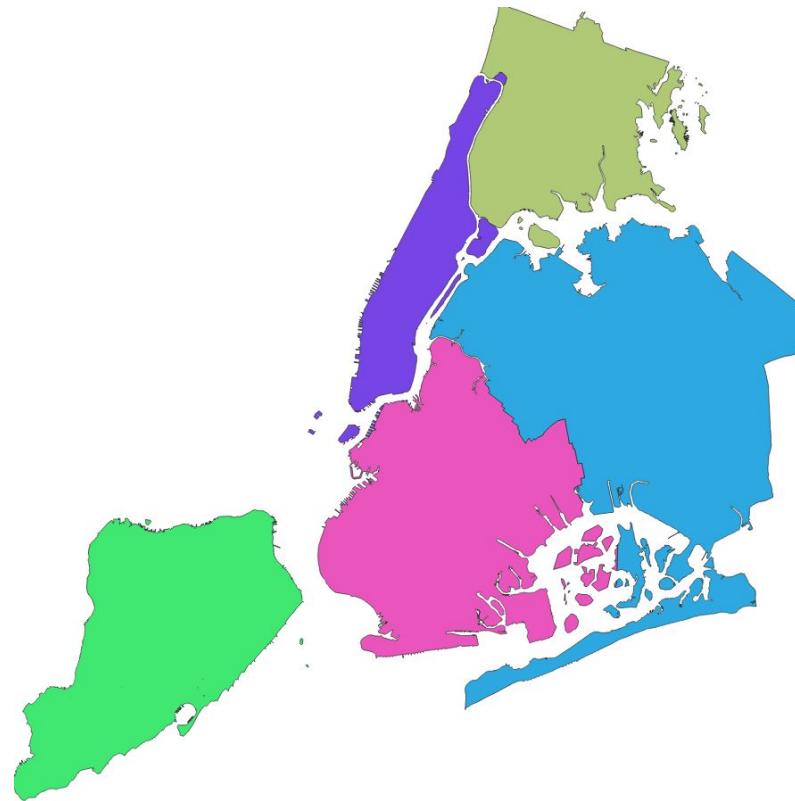
# Terminology

- Esri “dissolve” == PostGIS “union”
  - Melt together small things into larger things.
- Esri “union” == PostGIS “overlay”
  - Cookie cut larger things into smaller things.

# Forms

- `ST_Union(geom1, geom2)`
  - Melt together two geometries.
- `ST_Union(geometry[])`
  - Melt together a set of geometries. “Aggregate” function like `Sum()` or `Average()`. Use with `GROUP BY`.

**“How would you make a county map from census blocks?”**



## Census Block ID

US Census Block IDs encode the *geographic hierarchy* used by the census.

**360610001001001 = 36 061 000100 1 001**

**36** = State of New York

**061** = New York County (Manhattan)

**000100** = Census Tract

**1** = Census Block Group

**001** = Census Block

**“How would you make a county map from census blocks?”**

```
-- An nyc_census_counties table
-- by merging census blocks
CREATE TABLE nyc_census_counties AS
SELECT
    ST_Union(geom) AS geom,
    SubStr(blkid,1,5) AS countyid
FROM nyc_census_blocks
GROUP BY countyid;
```

# Section 21 - Geometry Constructing Exercises

GEOMETRY FUNCTIONS	
ST_Area(geometry)	ST_SRID(geometry)
ST_AsBinary(geometry)	ST_StartPoint(geometries)
ST_AsGML(geometry)	ST_SymDifference(geometry,geometry)
ST_AsGeoJSON(geometry)	ST_Touches(geometry, geometry)
ST_AsKML(geometry)	ST_Union(geometry, geometry)
ST_AsSVG(geometry)	ST_Union(geometry)
ST_AsText(geometry)	ST_Validate(geometry,geometry)
ST_Buffer(geometry,distance)	ST_Xpoint
ST_Centroid(geometry)	ST_Ypoint
ST_Contains(geometry,geometry)	ST_ZipPoint
ST_CoveredBy(geometry,geometry)	ST_M(point)
ST_Crosses(geometry,geometry)	
ST_Difference(geometry,geometry)	
ST_DWithin(geometry,geometry,radius)	
ST_Disjoint(geometry,geometry)	
ST_Distances(geometry,geometry)	
ST_EndPoint(geometry)	
ST_Equals(geometry,geometry)	
ST_Intersection(geometry,polygon)	
ST_GeomFromGML(text)	
ST_GeomFromKML(text)	
ST_GeomFromText(text)	
ST_GeomFromWKB(byts)	
ST_GeometryN(collection,n)	
ST_GeometryType(geometry)	
ST_IntersectingN(polygons)	
ST_Intersects(geometry,geometry)	
ST_Intersection(geometry,geometry)	
ST_IsValid(geometry)	
ST_IsValidReason(geometry)	
ST_Length(linestrings)	
ST_MakePoint(double,double)	
ST_NPoints(geometry)	
ST_NRings(polygons)	
ST_NumberGeometries(collection)	
ST_OrderedEqual(geometry,geometry)	
ST_Overlaps(geometry,geometry)	
ST_Perimeter(polygons)	
ST_PlanarSurface(geometry)	
ST_Relate(geometry,geometry)	
ST_Reverse(geometry)	
ST_Simplify(geometry)	

GEOGRAPHY FUNCTIONS	
ST_Area(geography)	ST_AsBinary(geography)
ST_AsGML(geography)	ST_AsGeoJSON(geography)
ST_AsKML(geography)	ST_AsMVT(geography)
ST_AsSVG(geography)	ST_AsText(geography)
ST_AsWKT(geography)	ST_Boundary(geography,distance)
ST_CoveredBy(geography,geography)	ST_Covers(geography,geography)
ST_Covers(geography,geography)	ST_DWithin(geometry,geometry,threshold)
ST_Distance(geometry,geometry)	ST_Disjoint(geometry,geometry)
ST_GeomFromWKB(byts)	ST_GeomFromText(text)
ST_GeographyFromText(text)	ST_Intersects(geometry,geography)
ST_Intersects(geometry,geography)	ST_Intersects(geometry,geography)
ST_Intersects(geometry,geography)	ST_Length(geometry)

nyc_census_blocks (36592 records)	
blkid	A 15-digit code that uniquely identifies every census block. Eg: 36005000109000
popu_total	Total number of people in the census block
popu_white	Number of people self-identifying as "White" in the block
popu_black	Number of people self-identifying as "Black" in the block
popu_native	Number of people self-identifying as "Native American" in the block
popu_asian	Number of people self-identifying as "Asian" in the block
popu_other	Number of people self-identifying with other categories in the block
house_total	Number of housing units in the block
house_own	Number of owner-occupied housing units in the block
house_rent	Number of renter-occupied housing units in the block
boroughname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the block

nyc_neighborhoods (129 records)	
name	Name of the neighborhood
boroughname	Name of borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
the_geom	Polygon boundary of the neighborhood

nyc_streets (19991 records)	
name	Name of the street
oneway	Is the street one-way? "yes" = yes, "no" = no
type	Road type. Eg: primary, secondary, residential, motorway
the_geom	Linear centerline of the street

nyc_subway_stations (491 records)	
name	Name of the station
borough	Name of 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" = no
the_geom	Point location of the station

nyc_census_socidata	
tractid	An 11-digit code for every census tract. Eg: 36005000100
tracti_total	Number of workers in the tract
tracti_public	Number of workers in the tract who take public transit
tracti_private	Number of workers in the tract who use private automobiles/ motorcycles
tracti_other	Number of workers in the tract who use other forms like walking/ biking
tracti_time_mins	Total minutes spent in transit by all workers in the tract (minutes)
family_size	Number of families in the tract
family_income_median	Median family income in the tract (dollars)
family_income_aggregate	Total income for all families in the tract (dollars)
edu_total	Number of people with educational history
edu_no_highschool_dipl	Number of people with no highschool diploma
edu_highschool_dipl	Number of people with highschool diplomas and no further education
edu_college_dipl	Number of people with college diplomas and no further education
edu_graduate_dipl	Number of people with graduate school diplomas

**How many census blocks don't contain their own centroid?**

```
SELECT Count(*)
FROM nyc_census_blocks
WHERE NOT
      ST_Contains(
          geom,
          ST_Centroid(geom)
      );
```

**Union all the census blocks into a single output.  
What kind of geometry is it? How many parts does it have?**

```
CREATE TABLE nyc_census_blocks_merge AS  
SELECT ST_Union(geom) AS geom  
FROM nyc_census_blocks;
```

---

```
SELECT ST_GeometryType(geom),  
       ST_NumGeometries(geom)  
FROM nyc_census_blocks_merge;
```

**“What is the area of a one unit buffer around the origin?  
How different is it from what you would expect? Why?”**

```
SELECT ST_Area(ST_Buffer('POINT(0 0)', 1));
```

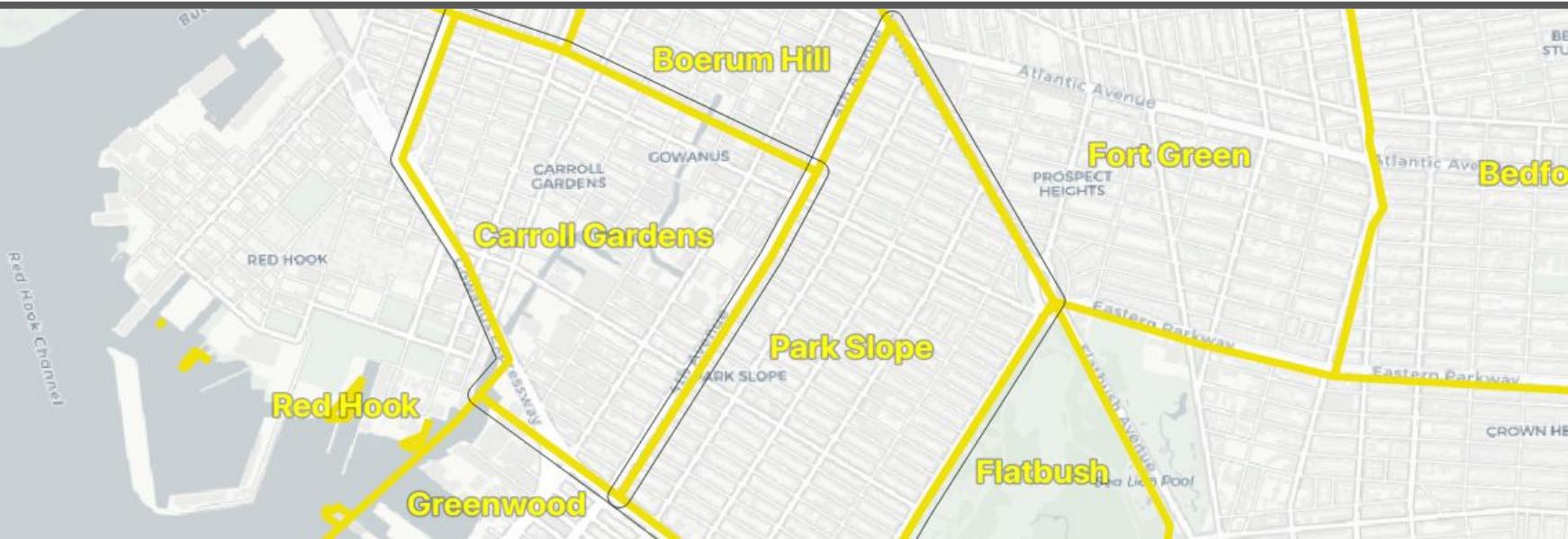
3.121445152258052

---

```
SELECT pi();
```

3.141592653589793

**The Brooklyn neighborhoods of Park Slope and Carroll Gardens are going to war! Construct a polygon delineating a 100 meter wide DMZ on the border between the neighborhoods. What is the area of the DMZ?**



```
CREATE TABLE brooklyn_dmz AS
SELECT
    ST_Intersection(
        ST_Buffer(ps.geom, 50),
        ST_Buffer(CG.geom, 50))
    } AS geom
FROM
    nyc_neighborhoods ps,
    nyc_neighborhoods CG
WHERE ps.name = 'Park Slope'
AND CG.name = 'Carroll Gardens';
```

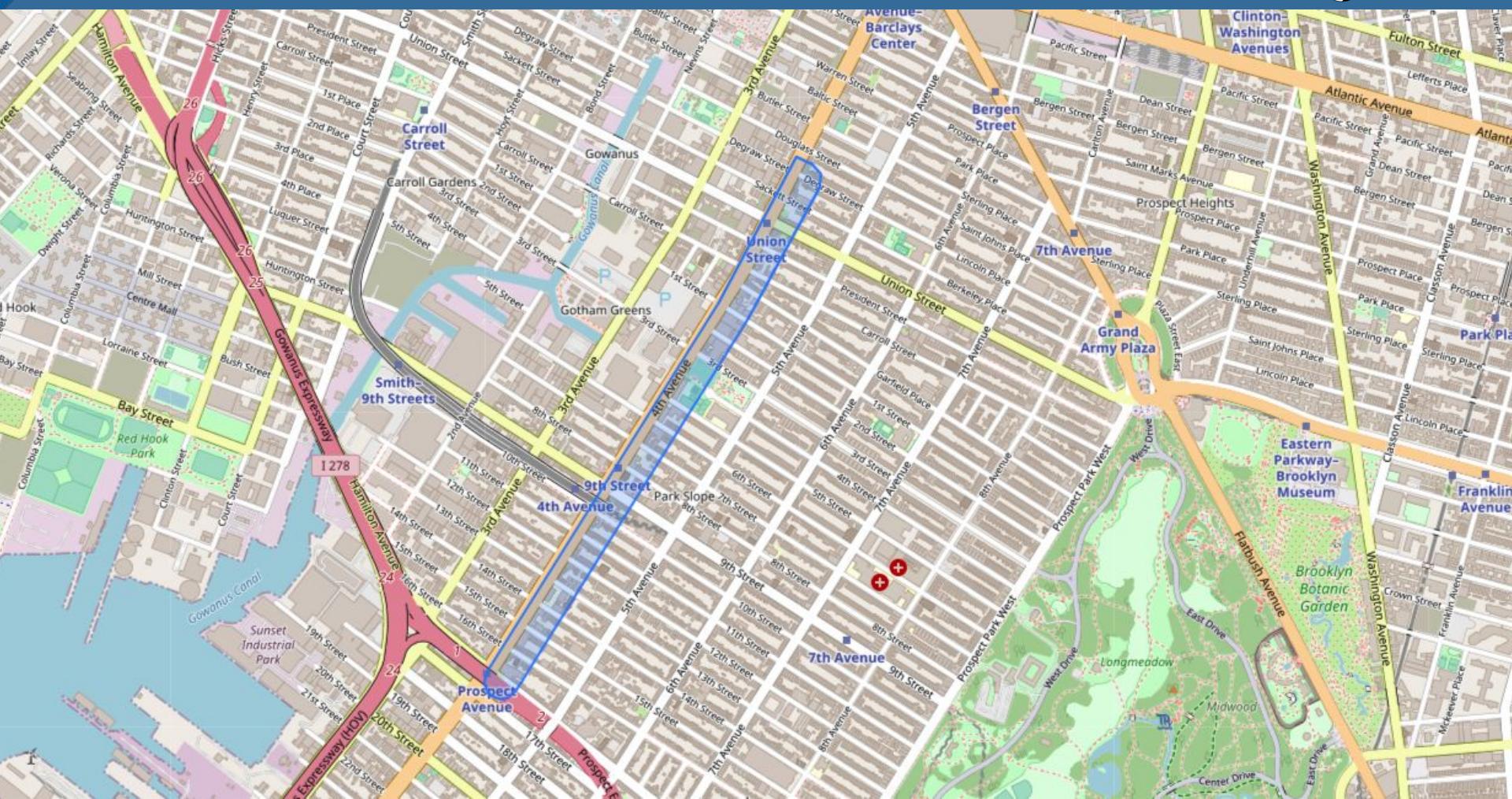
Wrap in ST\_Area(...) for area

Wrap in ST\_Transform(..., 4326) to  
see base map in PgAdmin map

## Section 21 - Geometry Constructing Exercises



PostGIS



# **Section 22 - More Spatial Joins!**

## Load the nyc\_census\_sociodata.sql table

1. Open the **Query Tool**  in pgAdmin
2. Select **Open File** 
3. Browse to the nyc\_census\_sociodata.sql file
4. Run query 

---

```
-- 2167
SELECT
    Count(*)
FROM nyc_census_sociodata;
```

**How would you make  
a census tract map  
from census blocks?**

## Recall...

Liberty Island blkid

**360610001001001 = 36 061 000100 1 001**

**36** = State of New York

**061** = New York County (Manhattan)

**000100** = Census Tract

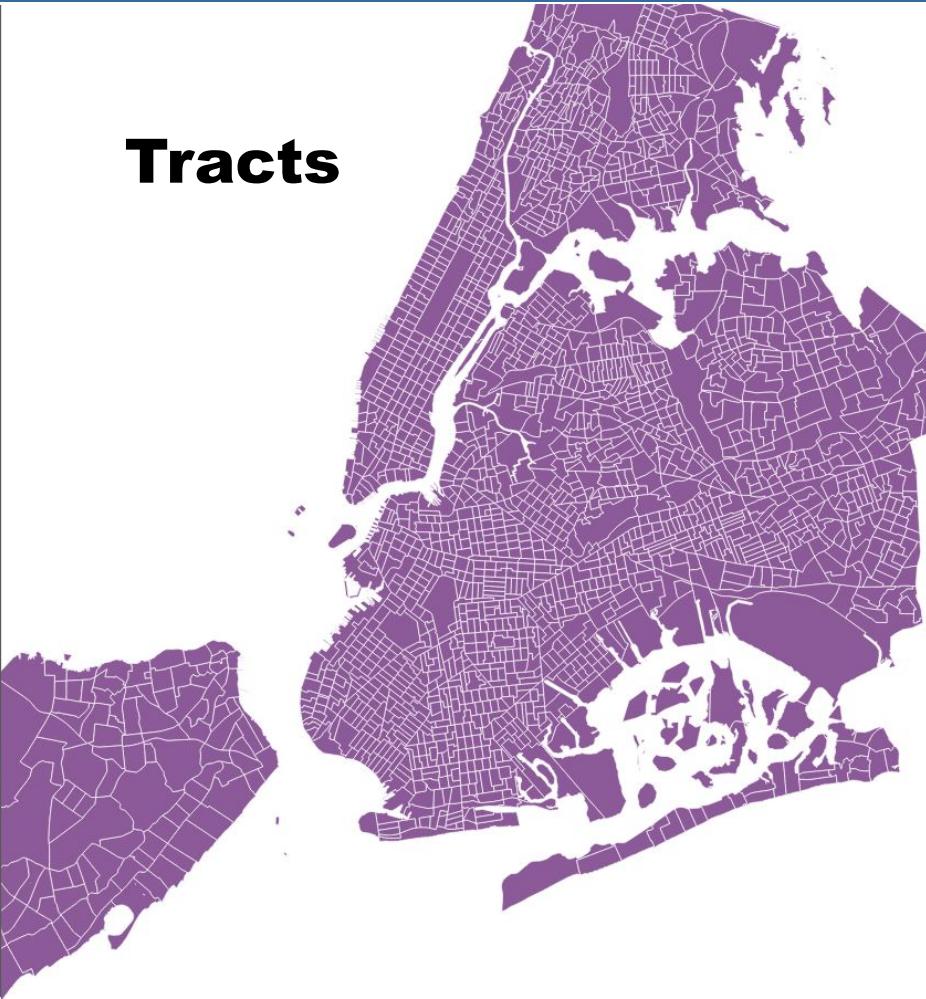
**1** = Census Block Group

**001** = Census Block

**Blocks**



**Tracts**



## ST\_Union() Blocks into Tracts

-- Make the tracts table

```
CREATE TABLE nyc_census_tract_geoms AS
SELECT
    ST_Union(geom) AS geom,
    substr(blkid,1,11) AS tractid
FROM nyc_census_blocks
GROUP BY tractid;
```

-- Index the tractid

```
CREATE INDEX nyc_census_tract_geoms_tractid_idx
ON nyc_census_tract_geoms (tractid);
```

**How can you associate census data with your  
census tract map?**

## ST\_Union() Blocks into Tracts

-- Make the tracts table

```
CREATE TABLE nyc_census_tracts AS
SELECT g.geom, a.*
FROM nyc_census_tract_geoms g
JOIN nyc_census_sociodata a
ON g.tractid = a.tractid;
```

-- Index the geometries

```
CREATE INDEX nyc_census_tract_gidx
ON nyc_census_tracts USING GIST (geom);
```

**“List top 10 New York neighborhoods ordered by  
the proportion of people who have graduate  
degrees?”**

# Graduate Degree Population %

SELECT

```
100.0 * Sum(t.edu_graduate_dipl) /  
      Sum(t.edu_total) AS graduate_pct,  
      n.name, n.boroname
```

FROM nyc\_neighborhoods n

JOIN nyc\_census\_tracts t

ON ST\_Intersects(n.geom, t.geom)

WHERE t.edu\_total > 0

GROUP BY n.name, n.boroname

ORDER BY graduate\_pct DESC

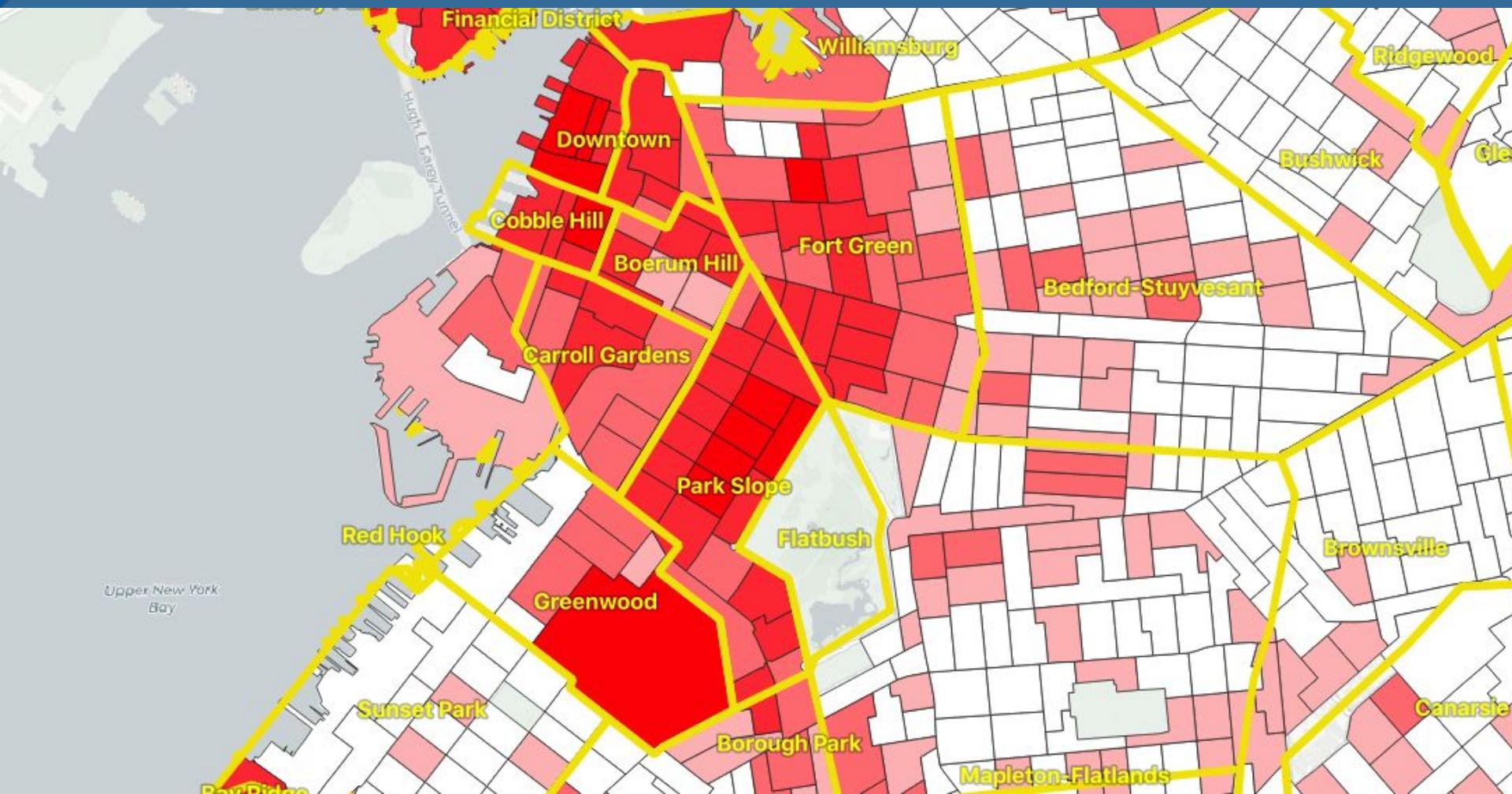
LIMIT 10;

# Graduate Degree Population %

graduate_pct	name	boroname
47.6469321851453175	Carnegie Hill	Manhattan
42.1632365492235696	Upper West Side	Manhattan
41.0656645950763598	Battery Park	Manhattan
39.5611557679774060	Flatbush	Brooklyn
39.3409549428379287	Tribeca	Manhattan
39.2188240872451399	North Sutton Area	Manhattan
38.6922550118291620	Greenwich Village	Manhattan
38.6054942073575506	Upper East Side	Manhattan
37.8834795573140662	Murray Hill	Manhattan
37.3714416181491744	Central Park	Manhattan



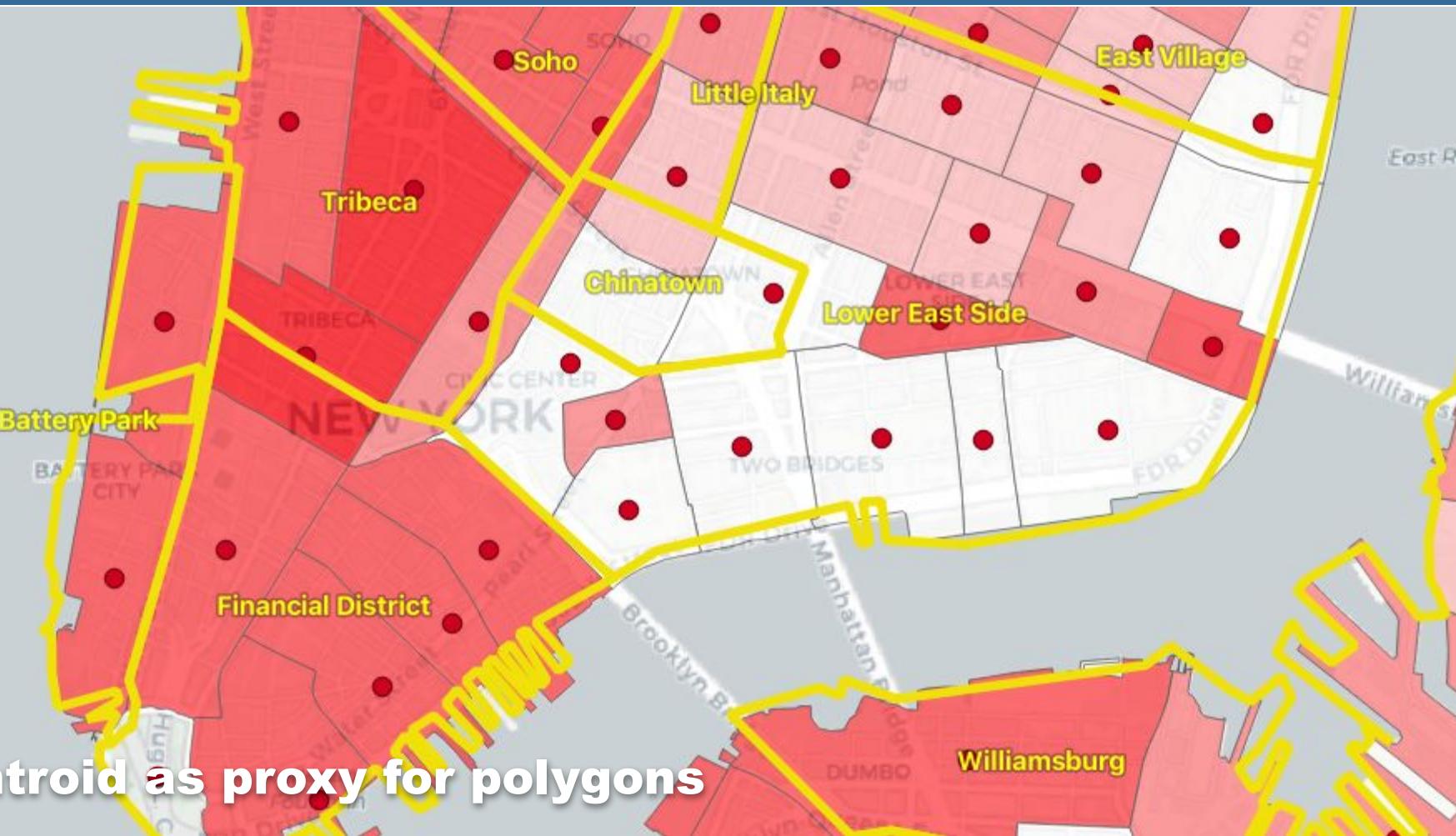
PostGIS





The otherwise-empty “Flatbush” neighborhood polygon (which mostly covers Prospect Park) just grazes one high-education tract polygon, resulting in a spurious high measurement for the neighborhood.

**What if a tract falls on the border between two neighborhoods?**



Centroid as proxy for polygons

# Join on ST\_Centroid

**SELECT**

```
100.0 * Sum(t.edu_graduate_dipl) /  
        Sum(t.edu_total) AS graduate_pct,
```

```
n.name,
```

```
n.boroname
```

```
FROM nyc_neighborhoods n
```

```
JOIN nyc_census_tracts t
```

```
ON ST_Contains(n.geom, ST_Centroid(t.geom))
```

```
WHERE t.edu_total > 0
```

```
GROUP BY n.name, n.boroname
```

```
ORDER BY graduate_pct DESC
```

```
LIMIT 10;
```

# Join on intersects vs centroid

ST\_Intersects()

graduate_pct	name	boroname
47.6	Carnegie Hill	Manhattan
42.2	Upper West Side	Manhattan
41.1	Battery Park	Manhattan
39.6	Flatbush	Brooklyn
39.3	Tribeca	Manhattan
39.2	North Sutton Area	Manhattan
38.7	Greenwich Village	Manhattan
38.6	Upper East Side	Manhattan
37.9	Murray Hill	Manhattan
37.4	Central Park	Manhattan

ST\_Centroid()

graduate_pct	name	boroname
48.0	Carnegie Hill	Manhattan
44.2	Morningside Heights	Manhattan
42.1	Greenwich Village	Manhattan
42.0	Upper West Side	Manhattan
41.4	Tribeca	Manhattan
40.7	Battery Park	Manhattan
39.5	Upper East Side	Manhattan
39.3	North Sutton Area	Manhattan
37.4	Cobble Hill	Brooklyn
37.4	Murray Hill	Manhattan

**How many people live within 500m of a subway station?**

# ST\_Centroid

How many people in New York?

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

## ST\_Centroid

How many people 500m from a subway station?

`SELECT`

```
  Sum(popn_total)
FROM nyc_census_blocks census
JOIN nyc_subway_stations subway
ON ST_DWithin(
    census.geom,
    subway.geom,
    500
);
```

## **ST\_Centroid**

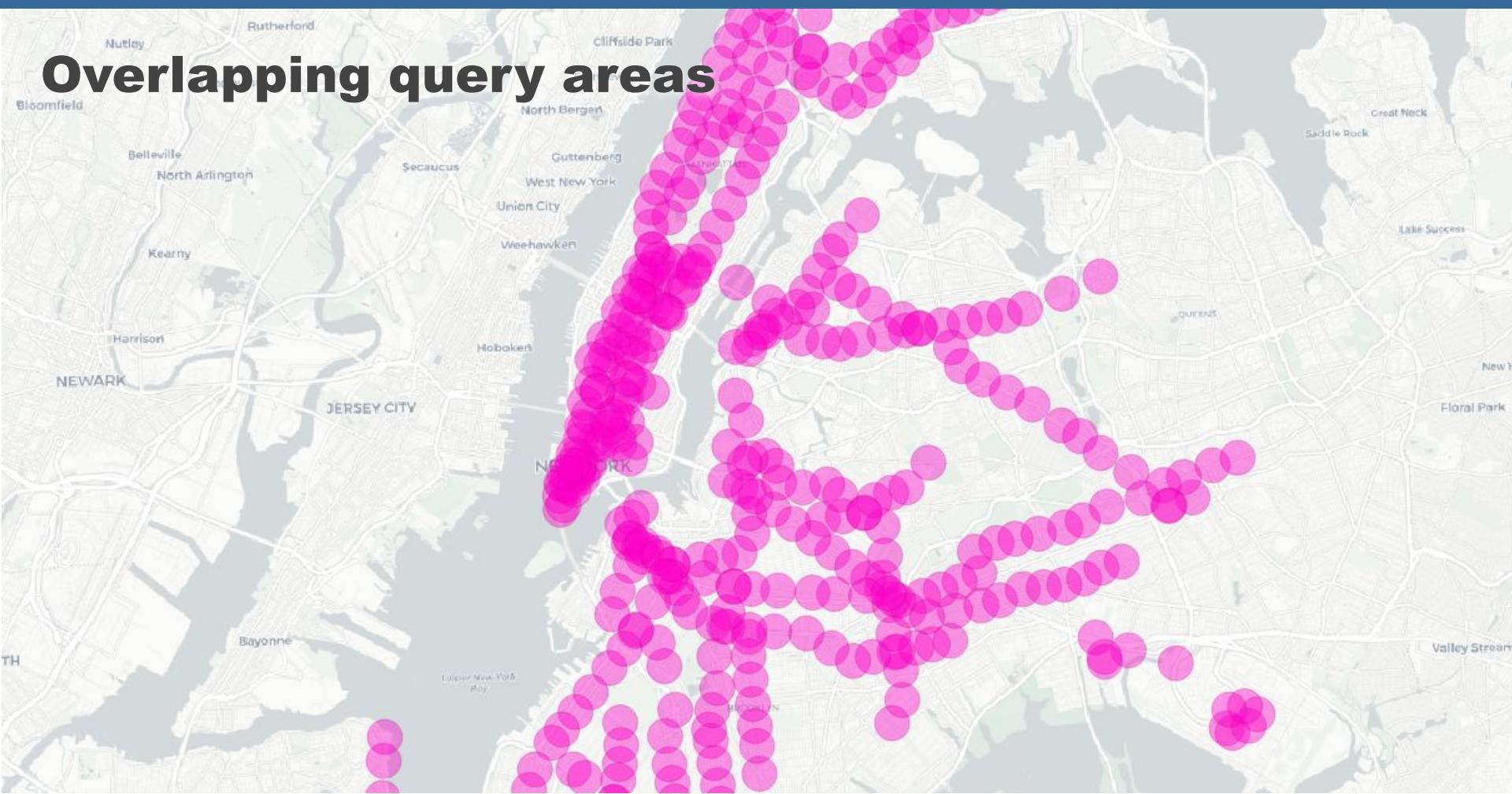
**How many people in New York?**

**8,175,032**

**How many people 500m from a subway station?**

**10,855,873 ?!?!?**

# Overlapping query areas



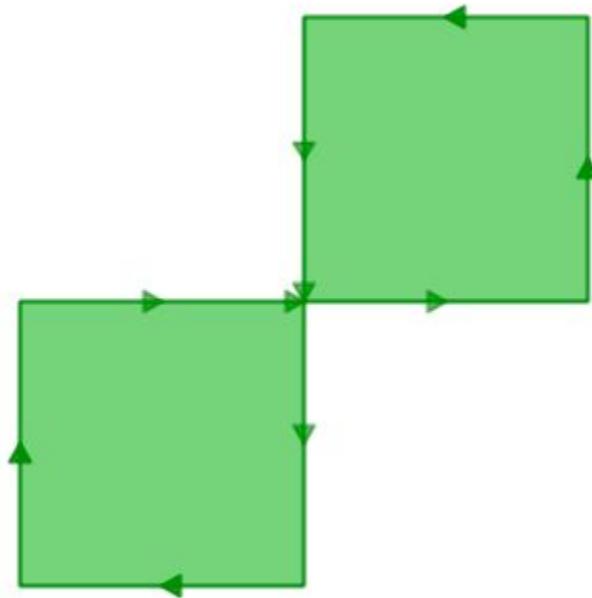
# Overlapping query areas

How many people 500m from a subway station?

```
WITH distinct_blocks AS (
  SELECT DISTINCT ON (blkid) popn_total
  FROM nyc_census_blocks census
  JOIN nyc_subway_stations subway
  ON ST_DWithin(
    census.geom,
    subway.geom,
    500)
)
SELECT Sum(popn_total)
FROM distinct_blocks;
```

# **Section 23 - Validity**

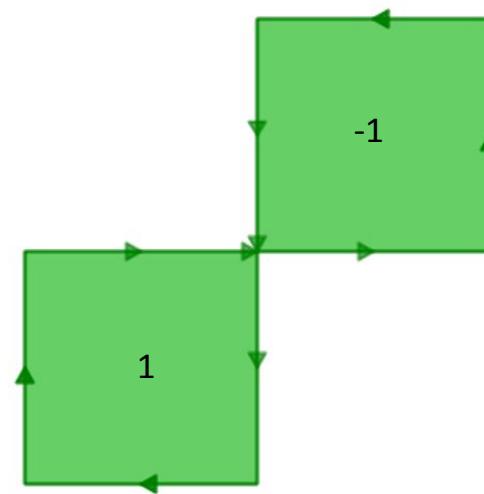
**POLYGON((0 0, 0 1, 2 1, 2 2, 1 2, 1 0, 0 0))**



# Why does validity matter?

Geometry algorithms rely on properties enforced by validity: ring orientation, self-crossing, self-touching. All can confuse different algorithms.

```
SELECT ST_Area(ST_GeomFromText(  
    'POLYGON((0 0, 0 1, 1 1,  
              2 1, 2 2, 1 2,  
              1 1, 1 0, 0 0))'  
));
```



0

# Can we test validity?

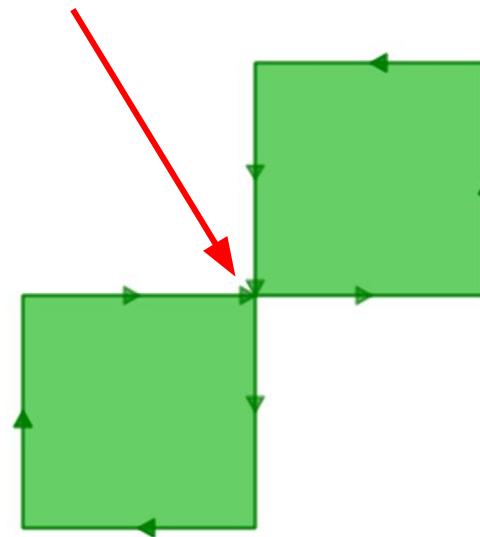
`ST_IsValid()` returns a boolean for validity, and `ST_IsValidReason()` returns a coordinate of where the error is, and a textual reason.

```
SELECT ST_IsValid(ST_GeomFromText(  
    'POLYGON((0 0, 0 1, 1 1,  
             2 1, 2 2, 1 2,  
             1 1, 1 0, 0 0))'  
));
```

---

false

Self-intersection[1 1]



## Can we test validity in bulk?

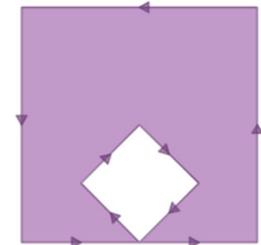
```
SELECT name, boroname,  
       ST_IsValidReason(geom)  
FROM nyc_neighborhoods  
WHERE NOT ST_IsValid(geom);
```

---

Howard Beach	Queens	Self-intersection[596394 4500899]
Corona	Queens	Self-intersection[595483 4513817]
Red Hook	Brooklyn	Self-intersection[582655 4500908]
Steinway	Queens	Self-intersection[593198 4515125]

## Can we fix validity in bulk?

The “banana polygon” is a polygon with a hole, formed by a ring that touches itself at a single point.



```
SELECT ST_AsText(ST_MakeValid(  
    'POLYGON((0 0, 2 0, 1 1, 2 2, 3 1, 2 0, 4 0, 4 4, 0 4, 0 0))'))
```

ST\_MakeValid() juggles the components of an invalid polygon to form a “best guess” valid interpretation of the rings. The “banana polygon” gets turns into a traditional exterior/interior ring polygon.

```
POLYGON((0 0,0 4,4 4,4 0,2 0,0 0),(3 1,2 2,1 1,2 0,3 1))
```

# Can we fix validity in bulk?

```
UPDATE nyc_neighborhoods  
SET geom = ST_MakeValid(geom)  
WHERE NOT ST_IsValid(geom);
```

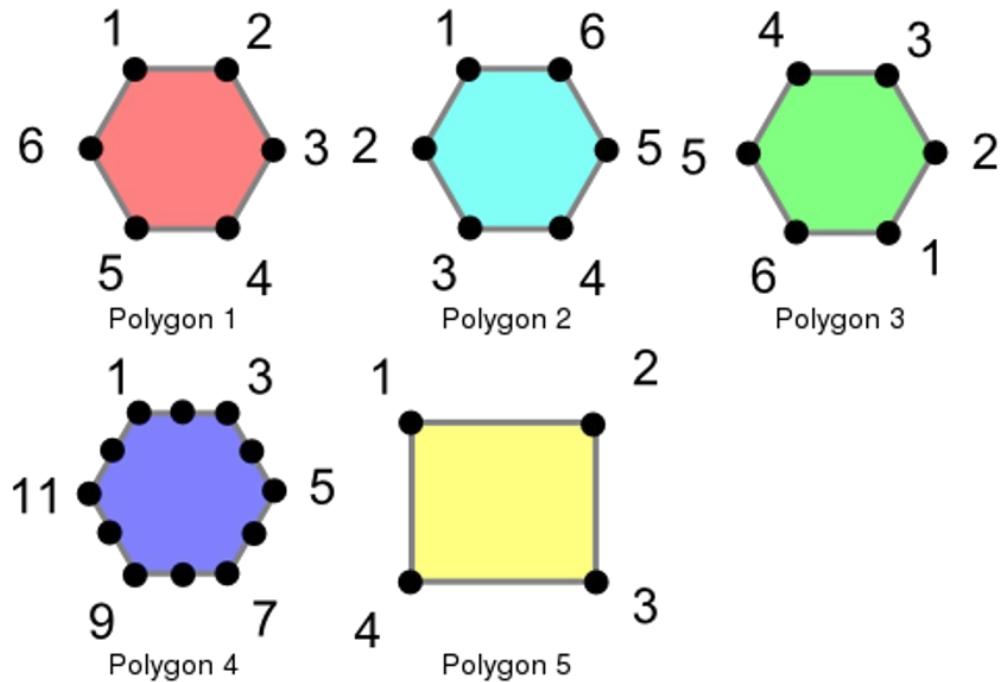
---

## **ST\_MakeValid(geom, options)**

PostGIS 3.2+ includes text options to change the repair algorithm.

```
'method=linework'  
'method=structure keepcollapsed=false'
```

# **Section 24 - Equality**



# Create the test polygons

```
CREATE TABLE polygons (id integer, name varchar, poly geometry);
```

```
INSERT INTO polygons VALUES
```

```
(1, 'Polygon 1', 'POLYGON((-1 1.732,1 1.732,2 0,1 -1.732,
    -1 -1.732,-2 0,-1 1.732))'),
(2, 'Polygon 2', 'POLYGON((-1 1.732,-2 0,-1 -1.732,1 -1.732,
    2 0,1 1.732,-1 1.732))'),
(3, 'Polygon 3', 'POLYGON((1 -1.732,2 0,1 1.732,-1 1.732,
    -2 0,-1 -1.732,1 -1.732))'),
(4, 'Polygon 4', 'POLYGON((-1 1.732,0 1.732, 1 1.732,1.5 0.866,
    2 0,1.5 -0.866,1 -1.732,0 -1.732,-1 -1.732,-1.5 -0.866,
    -2 0,-1.5 0.866,-1 1.732))'),
(5, 'Polygon 5', 'POLYGON((-2 -1.732,2 -1.732,2 1.732,
    -2 1.732,-2 -1.732)));
```

# Ways of testing equality!

`ST_OrderingEquals(A, B)`

`ST_Equals(A, B)`

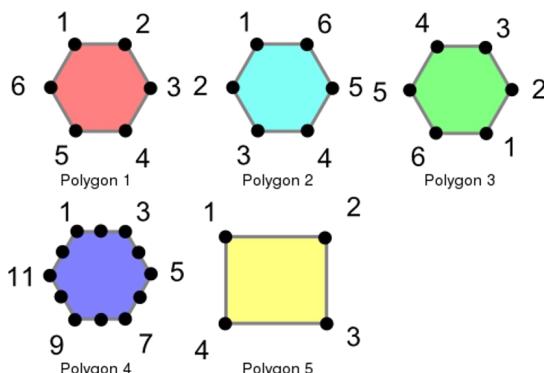
$A = B$

$A \approx B$

```

SELECT a.name, b.name,
CASE WHEN
    ST_OrderingEquals(a.poly, b.poly)
    THEN 'Exactly Equal'
    ELSE 'Not Exactly Equal' END
FROM polygons AS a,
polygons AS b;

```

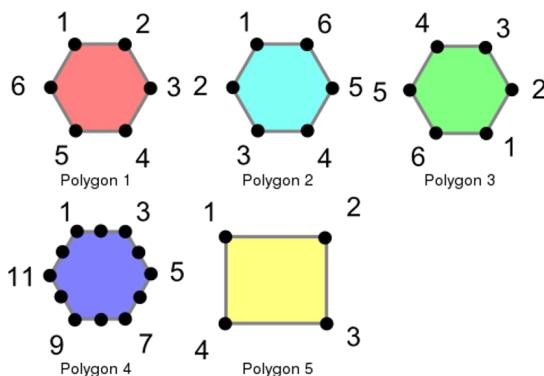


Polygon 1	Polygon 1	Exactly Equal
Polygon 1	Polygon 2	Not Exactly Equal
Polygon 1	Polygon 3	Not Exactly Equal
Polygon 1	Polygon 4	Not Exactly Equal
Polygon 1	Polygon 5	Not Exactly Equal
Polygon 2	Polygon 1	Not Exactly Equal
Polygon 2	Polygon 2	Exactly Equal
Polygon 2	Polygon 3	Not Exactly Equal
Polygon 2	Polygon 4	Not Exactly Equal
Polygon 2	Polygon 5	Not Exactly Equal
Polygon 3	Polygon 1	Not Exactly Equal
Polygon 3	Polygon 2	Not Exactly Equal
Polygon 3	Polygon 3	Exactly Equal
Polygon 3	Polygon 4	Not Exactly Equal
Polygon 3	Polygon 5	Not Exactly Equal
Polygon 4	Polygon 1	Not Exactly Equal
Polygon 4	Polygon 2	Not Exactly Equal
Polygon 4	Polygon 3	Not Exactly Equal
Polygon 4	Polygon 4	Exactly Equal
Polygon 4	Polygon 5	Not Exactly Equal
Polygon 5	Polygon 1	Not Exactly Equal
Polygon 5	Polygon 2	Not Exactly Equal
Polygon 5	Polygon 3	Not Exactly Equal
Polygon 5	Polygon 4	Not Exactly Equal
Polygon 5	Polygon 5	Exactly Equal

```

SELECT a.name, b.name,
CASE WHEN ST_Equals(a.poly, b.poly)
      THEN 'Spatially Equal'
      ELSE 'Not Equal' END
FROM polygons AS a,
polygons AS b;

```

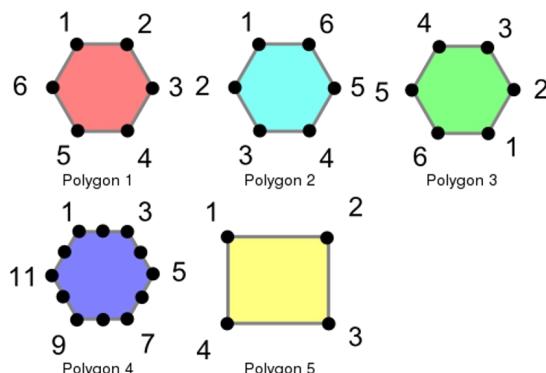


Polygon 1	Polygon 1	Spatially Equal
Polygon 1	Polygon 2	Spatially Equal
Polygon 1	Polygon 3	Spatially Equal
Polygon 1	Polygon 4	Spatially Equal
Polygon 1	Polygon 5	Not Equal
Polygon 2	Polygon 1	Spatially Equal
Polygon 2	Polygon 2	Spatially Equal
Polygon 2	Polygon 3	Spatially Equal
Polygon 2	Polygon 4	Spatially Equal
Polygon 2	Polygon 5	Not Equal
Polygon 3	Polygon 1	Spatially Equal
Polygon 3	Polygon 2	Spatially Equal
Polygon 3	Polygon 3	Spatially Equal
Polygon 3	Polygon 4	Spatially Equal
Polygon 3	Polygon 5	Not Equal
Polygon 4	Polygon 1	Spatially Equal
Polygon 4	Polygon 2	Spatially Equal
Polygon 4	Polygon 3	Spatially Equal
Polygon 4	Polygon 4	Spatially Equal
Polygon 4	Polygon 5	Not Equal
Polygon 5	Polygon 1	Not Equal
Polygon 5	Polygon 2	Not Equal
Polygon 5	Polygon 3	Not Equal
Polygon 5	Polygon 4	Not Equal
Polygon 5	Polygon 5	Spatially Equal

```

SELECT a.name, b.name,
CASE WHEN a.poly = b.poly
    THEN 'Spatially ='
    ELSE 'Not =' END
FROM polygons AS a,
polygons AS b;

```

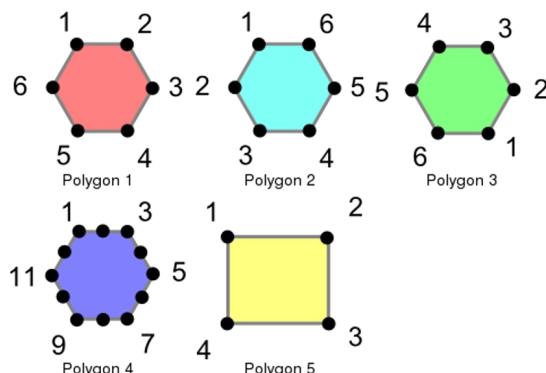


Polygon 1	Polygon 1	Spatially =
Polygon 1	Polygon 2	Not =
Polygon 1	Polygon 3	Not =
Polygon 1	Polygon 4	Not =
Polygon 1	Polygon 5	Not =
Polygon 2	Polygon 1	Not =
Polygon 2	Polygon 2	Spatially =
Polygon 2	Polygon 3	Not =
Polygon 2	Polygon 4	Not =
Polygon 2	Polygon 5	Not =
Polygon 3	Polygon 1	Not =
Polygon 3	Polygon 2	Not =
Polygon 3	Polygon 3	Spatially =
Polygon 3	Polygon 4	Not =
Polygon 3	Polygon 5	Not =
Polygon 4	Polygon 1	Not =
Polygon 4	Polygon 2	Not =
Polygon 4	Polygon 3	Not =
Polygon 4	Polygon 4	Spatially =
Polygon 4	Polygon 5	Not =
Polygon 5	Polygon 1	Not =
Polygon 5	Polygon 2	Not =
Polygon 5	Polygon 3	Not =
Polygon 5	Polygon 4	Not =
Polygon 5	Polygon 5	Spatially =

```

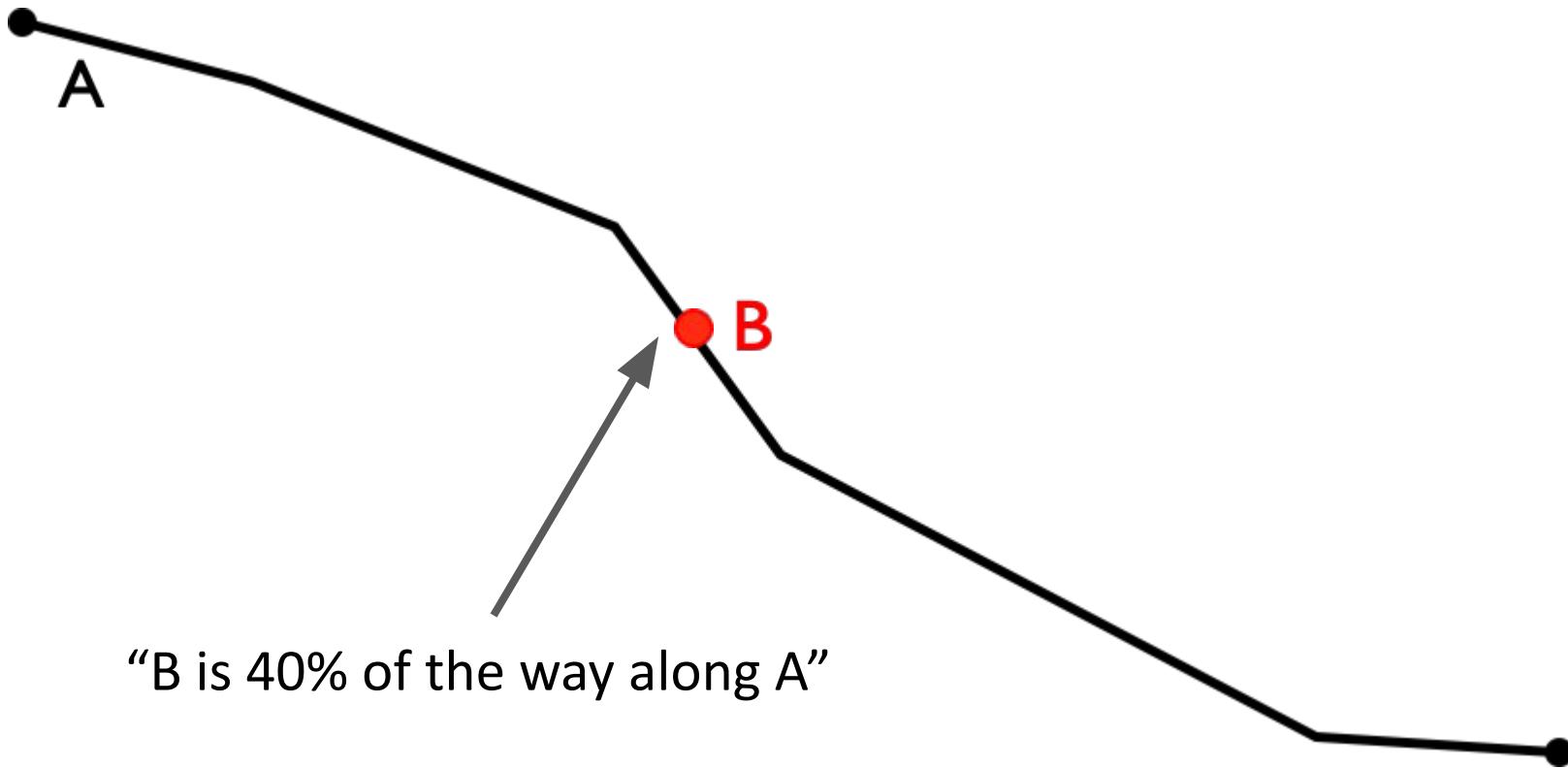
SELECT a.name, b.name,
CASE WHEN a.poly ~= b.poly
    THEN 'Bounds Equal'
    ELSE 'Bounds Not Equal' END
FROM polygons AS a,
polygons AS b;

```

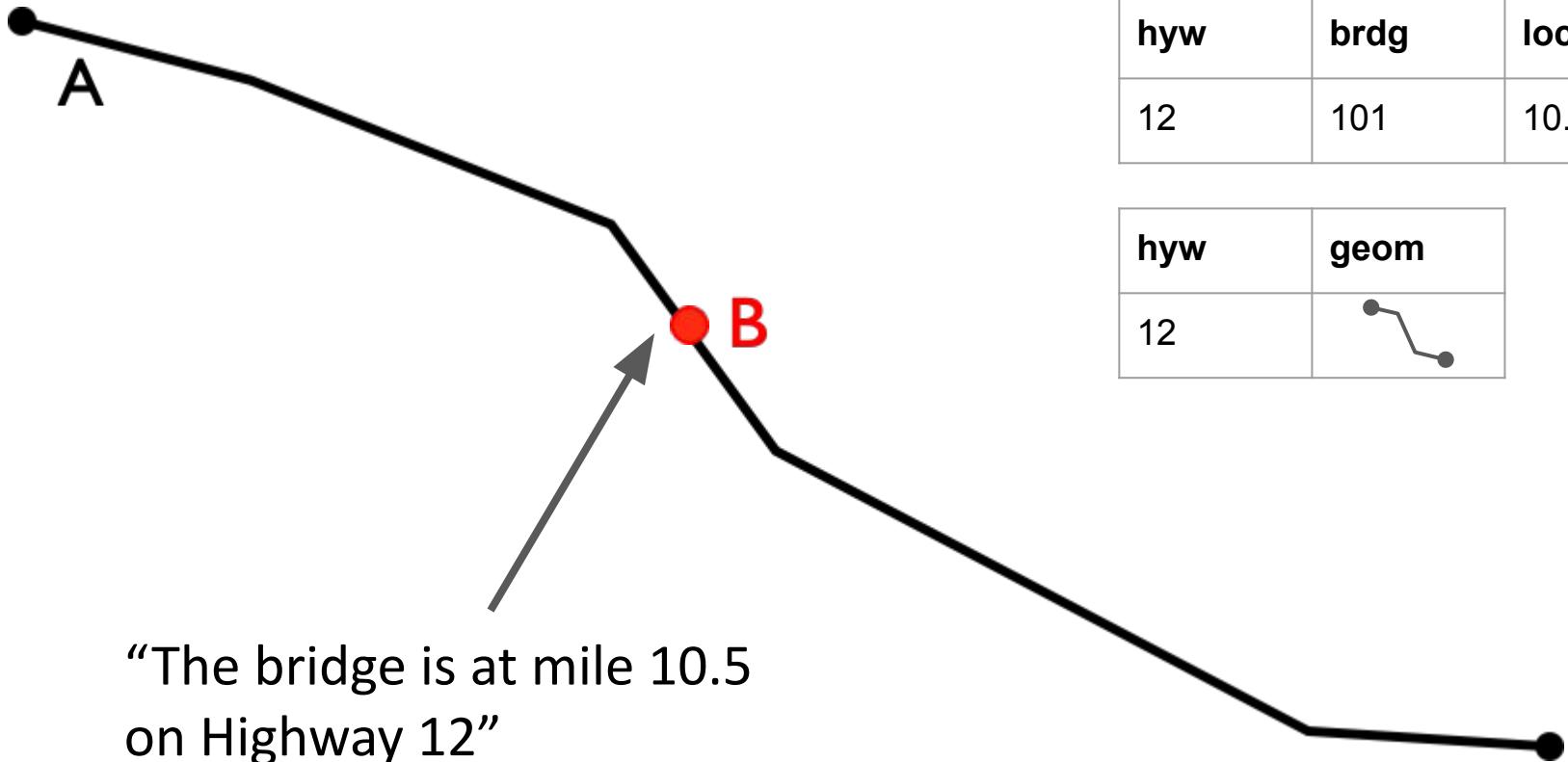


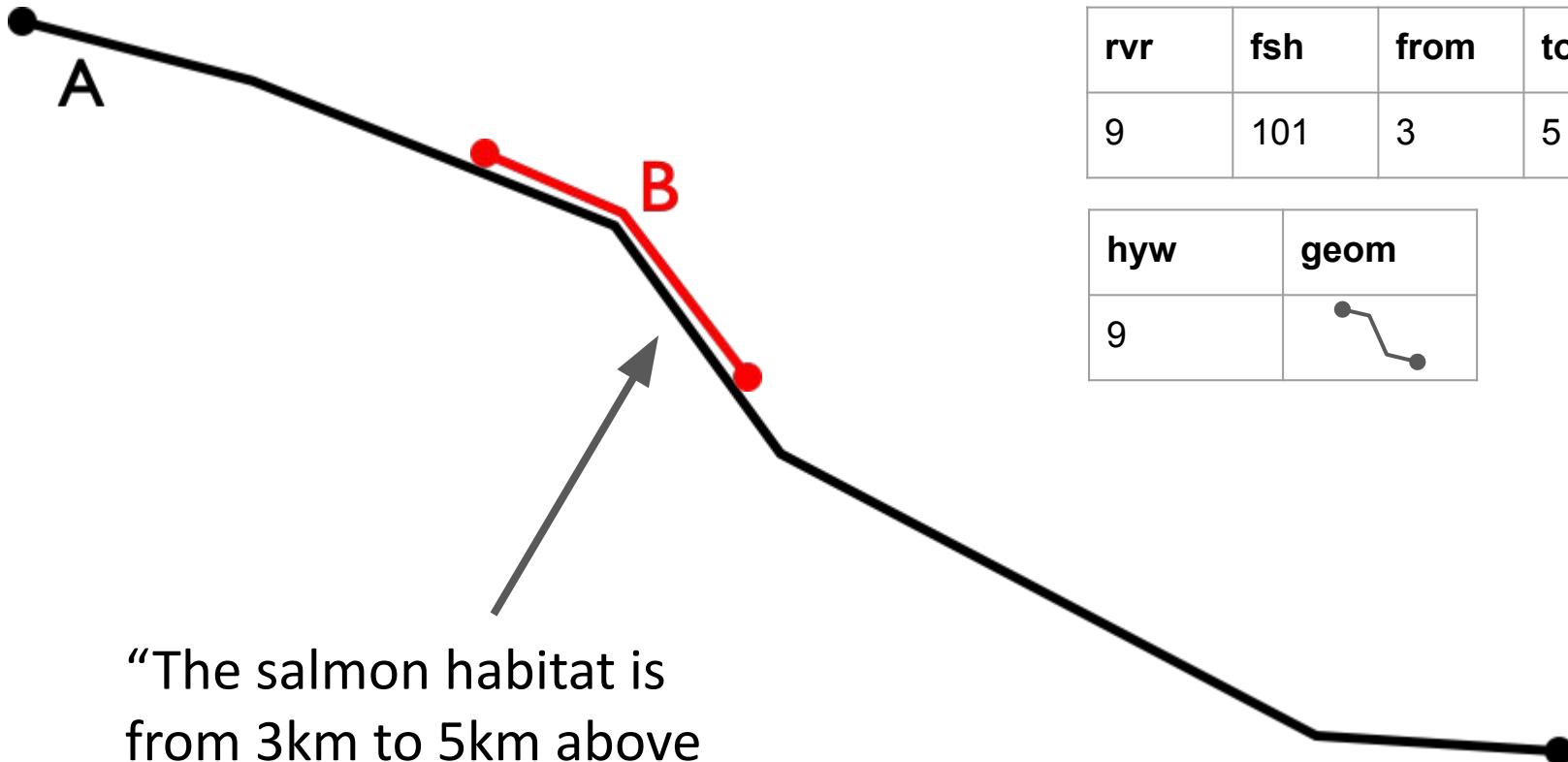
Polygon 1	Polygon 1	Bounds Equal
Polygon 1	Polygon 2	Bounds Equal
Polygon 1	Polygon 3	Bounds Equal
Polygon 1	Polygon 4	Bounds Equal
Polygon 1	Polygon 5	Bounds Equal
Polygon 2	Polygon 1	Bounds Equal
Polygon 2	Polygon 2	Bounds Equal
Polygon 2	Polygon 3	Bounds Equal
Polygon 2	Polygon 4	Bounds Equal
Polygon 2	Polygon 5	Bounds Equal
Polygon 3	Polygon 1	Bounds Equal
Polygon 3	Polygon 2	Bounds Equal
Polygon 3	Polygon 3	Bounds Equal
Polygon 3	Polygon 4	Bounds Equal
Polygon 3	Polygon 5	Bounds Equal
Polygon 4	Polygon 1	Bounds Equal
Polygon 4	Polygon 2	Bounds Equal
Polygon 4	Polygon 3	Bounds Equal
Polygon 4	Polygon 4	Bounds Equal
Polygon 4	Polygon 5	Bounds Equal
Polygon 5	Polygon 1	Bounds Equal
Polygon 5	Polygon 2	Bounds Equal
Polygon 5	Polygon 3	Bounds Equal
Polygon 5	Polygon 4	Bounds Equal
Polygon 5	Polygon 5	Bounds Equal

# **Section 25 - Linear Referencing**



"B is 40% of the way along A"





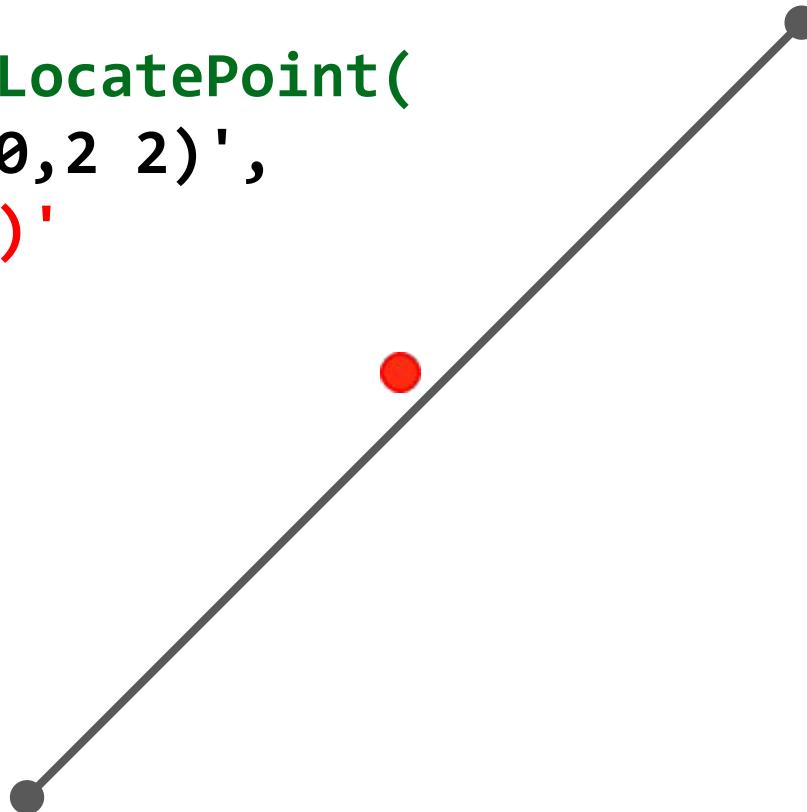
“The salmon habitat is  
from 3km to 5km above  
the confluence”

rvr	fsh	from	to
9	101	3	5

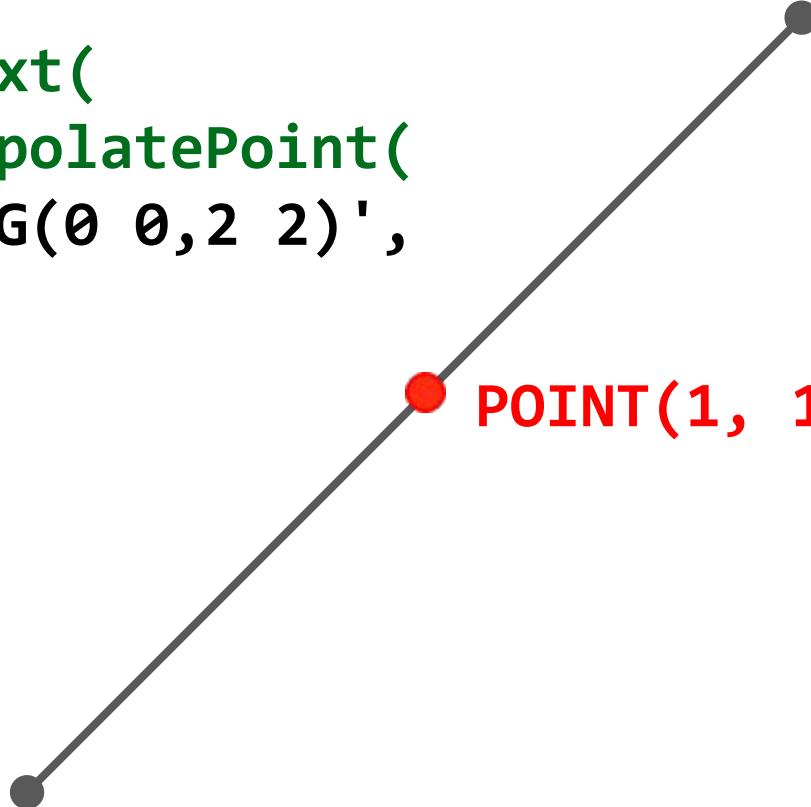
hyw	geom
9	

```
SELECT ST_LineLocatePoint(  
    'LINESTRING(0 0,2 2)',  
    'POINT(0.9 1.1)'  
)
```

0.5



```
SELECT ST_AsText(  
    ST_LineInterpolatePoint(  
        'LINESTRING(0 0,2 2)',  
        0.5  
    ));
```

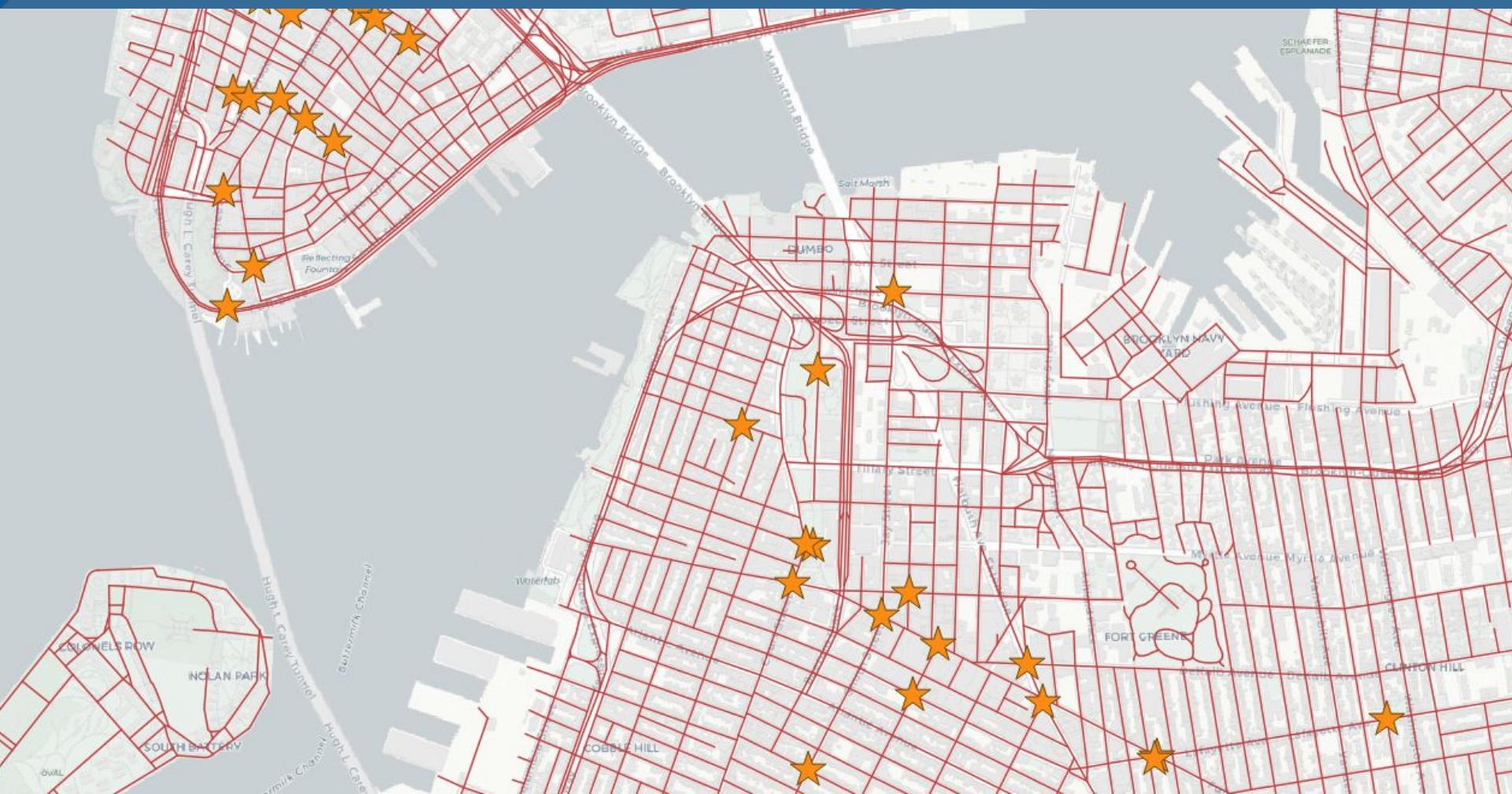


POINT(1, 1)

## Section 25 - Linear Referencing



PostGIS



# Find nearest street to each subway station

```
WITH ordered_nearest AS (
  SELECT
    ST_GeometryN(str.geom,1) AS streets_geom,
    str.gid AS str_gid,
    sub.geom AS subways_geom,
    sub.gid AS subways_gid,
    ST_Distance(str.geom, sub.geom) AS distance
  FROM nyc_streets str
  JOIN nyc_subway_stations sub
    ON ST_DWithin(str.geom, sub.geom, 200)
  ORDER BY subways_gid, distance ASC
)
```

# Find measure of station on nearest street

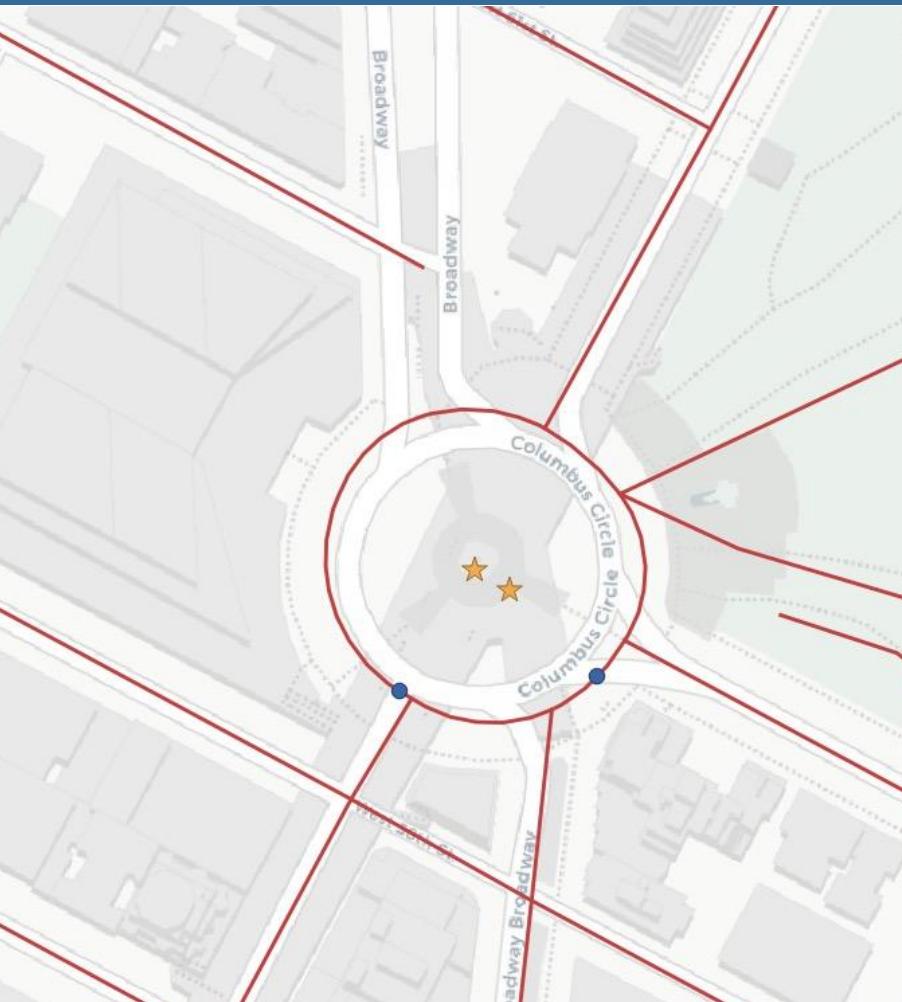
```
SELECT
  DISTINCT ON (subways_gid)
    subways_gid,
    streets_gid,
    distance,
    ST_LineLocatePoint(
      streets_geom,
      subways_geom) AS measure
  FROM ordered_nearest;
```

## Find measure of station on nearest street

subways_gid	streets_gid	measure
1	17404	0.0023154983819572554
2	17318	0.6354078182846773
3	19086	0.24946227178552738
4	1924	0.11187222763997673
5	2067	0.9261874246426975
6	1934	0.33457647816803476
7	2024	0.5549461001845787
8	2469	0.2296616075093935
9	2024	0.9069811058590412
10	2067	0.6202998183141508

## How to visualize events? Turn them back into points.

```
-- New view that turns events back
-- into spatial objects
CREATE OR REPLACE
VIEW nyc_subway_stations_lrs AS
SELECT
    events.subways_gid,
    ST_LineInterpolatePoint(
        ST_GeometryN(streets.geom, 1),
        events.measure) AS geom,
    events.streets_gid
FROM nyc_subway_station_events events
JOIN nyc_streets streets
ON (streets.gid = events.streets_gid);
```

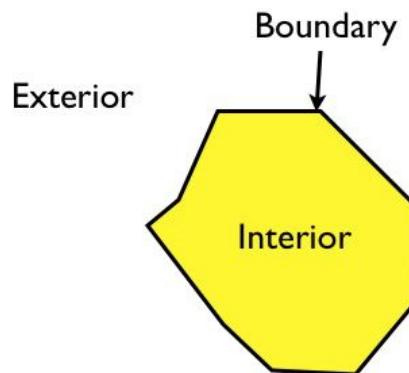


Original subway stations (orange stars) on Columbus Circle have been snapped over to the nearby roadways in the LRS view (blue circles)

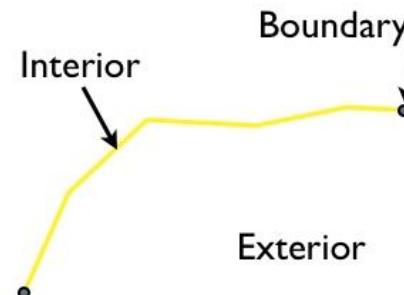
Shows how LRS functions can be used to snap points to a network, as well as to manage actual LRS data.

# **Section 26 - Dimensionally Extended 9 Intersection Model (DE9IM)**

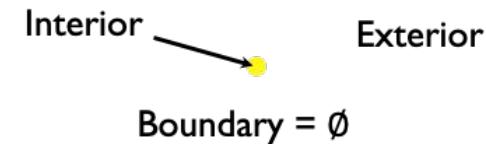
# Interior, Boundary and Exterior



**Polygons**



**Linestrings**



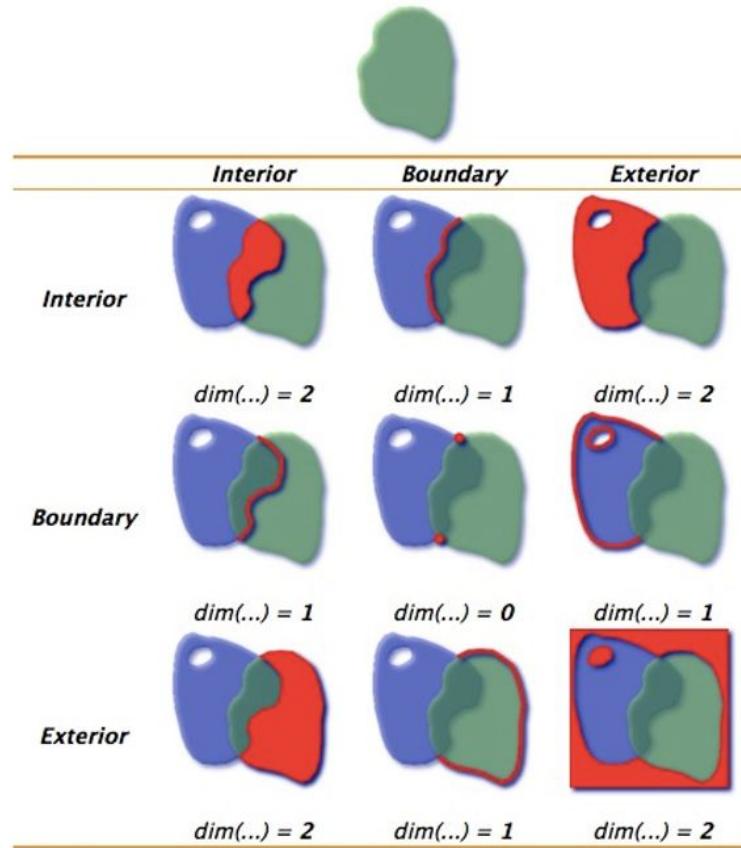
**Points**

# DE9IM

The DE9IM relationship between two geometries is calculated by examining the dimensionality of each intersection in the grid.



The full set of 9 intersections precisely defines the nature of the spatial relationship between the geometries.



## DE9IM



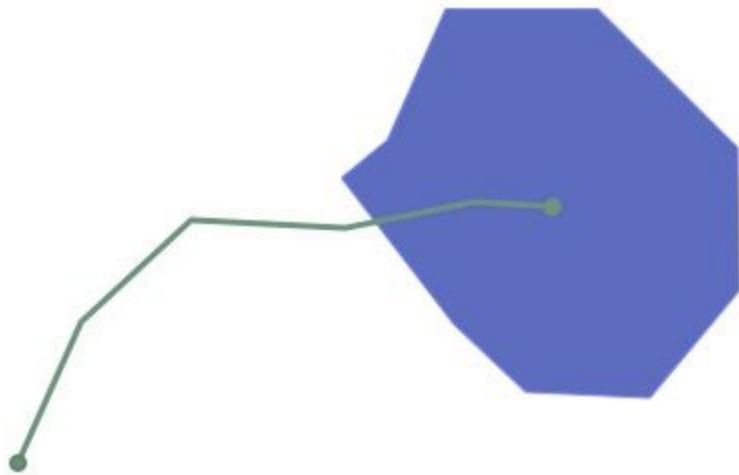
$$\dim(I(a) \cap I(b)) = 2$$



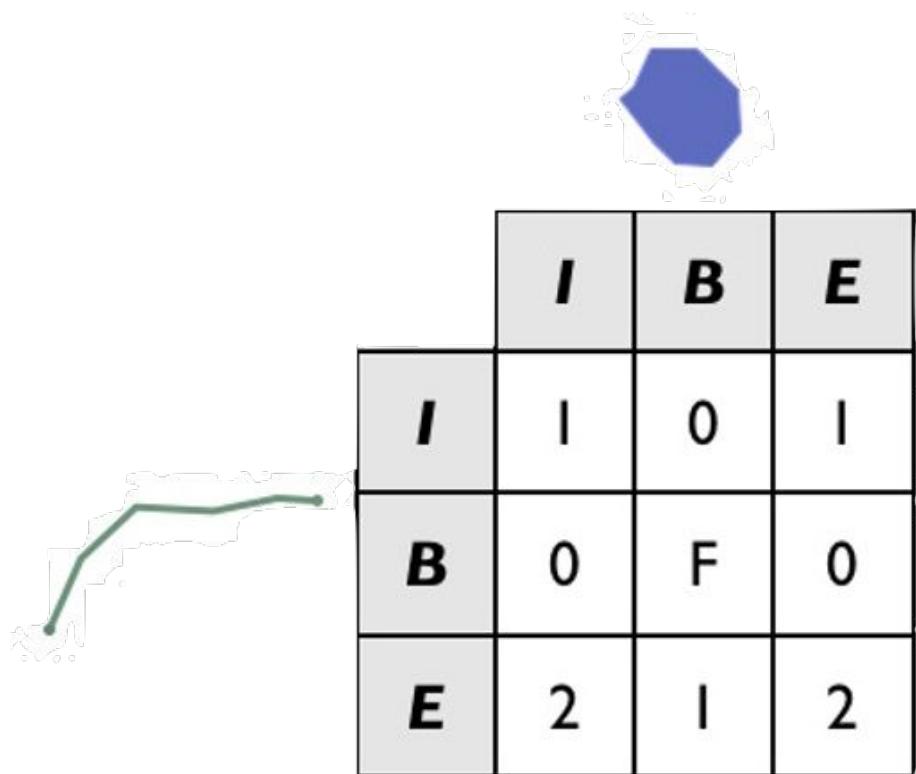
$$\dim(B(a) \cap B(b)) = 0$$

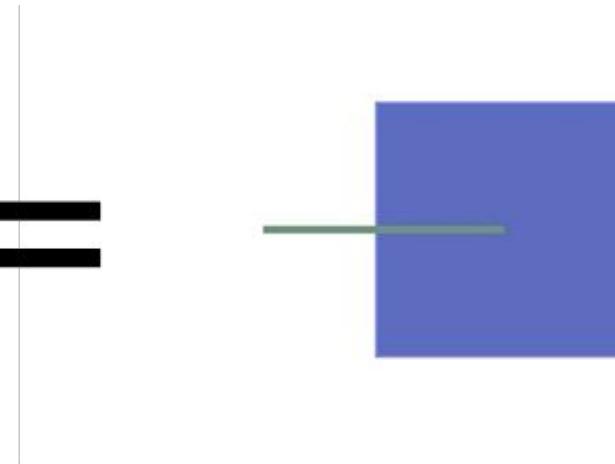
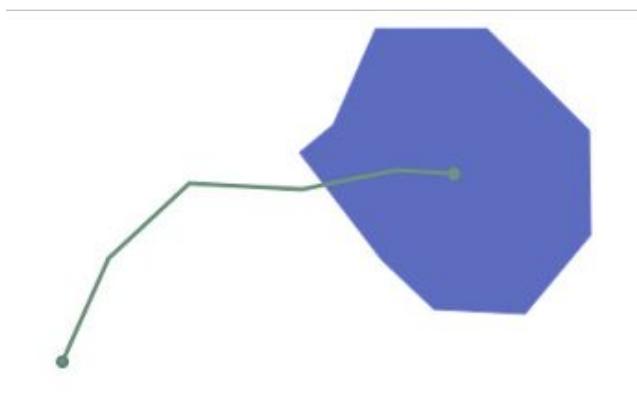
## DI9EM matrix codes

- **0** = 0-d intersection (puntal)
- **1** = 1-d intersection (linear)
- **2** = 2-d intersection (areal)
- **T** = some intersection of any dimension
- **F** = no intersection
- **\*** = no preference



Fill in the **DE9IM** matrix for these two geometries, examining each interaction, and then filling in the cell.





In terms of relationships, the more complex picture on the left is the same as the simple one on the right, that we will use for SQL examples.

## Calculate the DE9IM matrix for two geometries

```
SELECT ST_Relate(  
    'LINESTRING(0 0, 2 0)',  
    'POLYGON((1 -1, 1 1, 3 1, 3 -1, 1 -1))'  
) ;
```

**1010F0212**

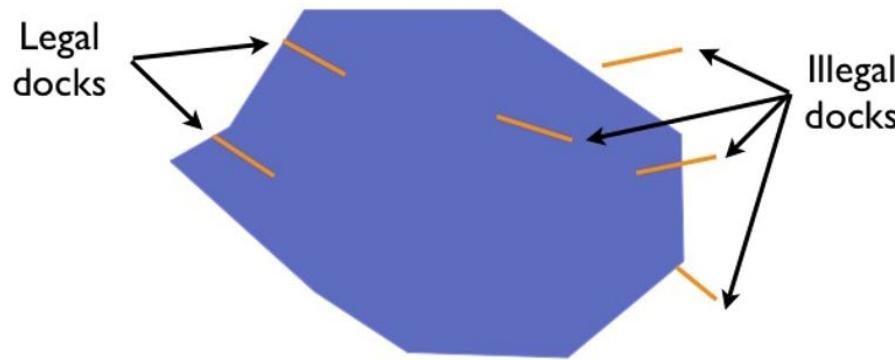


**101  
0F0  
212**



	I	B	E
I	1	0	1
B	0	F	0
E	2	1	2

# What is this for?



Data quality control. Some features have very specific allowed relationships to other features. Find the ones that break the rules in order to flag and fix them.

"Legal" docks have a specific pattern:  
IFF0OF212

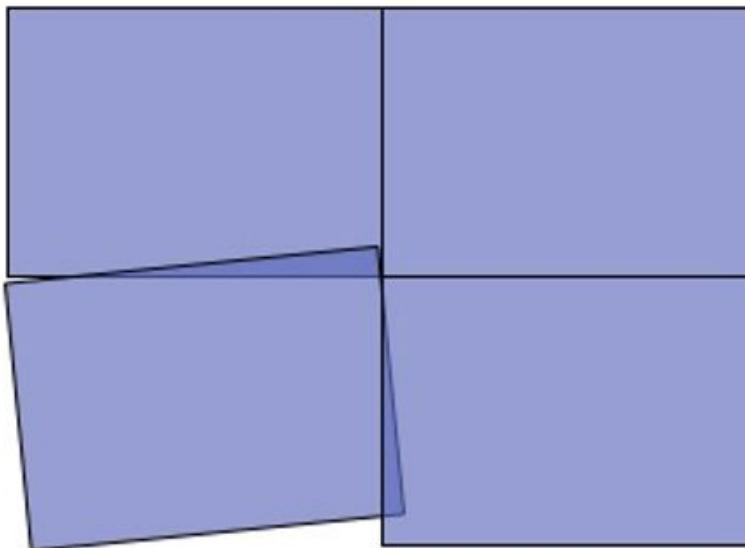


	I	B	E
I	I	F	F
B	0	0	F
E	2	I	2

**What is this for? Find any illegal docks.**

```
SELECT docks.*  
FROM docks  
JOIN lakes  
ON ST_Intersects(docks.geom, lakes.geom)  
WHERE NOT  
ST_Relate(docks.geom,  
          lakes.geom,  
          '1FF00F212');
```

# What is this for? Find overlapping census blocks!



Tracts with an overlap?

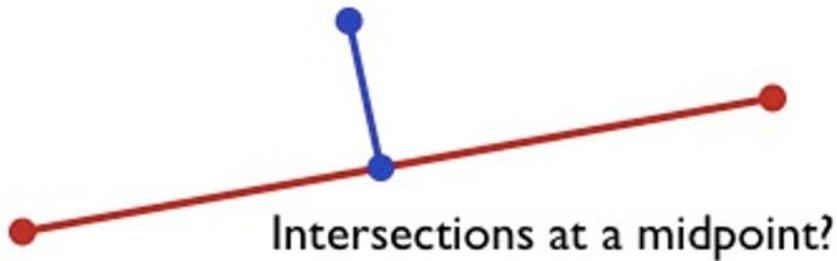
	I	B	E
I	2	*	*
B	*	*	*
E	*	*	*

# What is this for? Find overlapping census blocks!

```
SELECT a.gid, b.gid
FROM nyc_census_blocks a,
     nyc_census_blocks b
WHERE ST_Intersects(a.geom, b.geom)
      AND ST_Relate(a.geom,
                     b.geom,
                     '2*****')
      AND a.gid != b.gid
LIMIT 10;
```

	I	B	E
I	2	*	*
B	*	*	*
E	*	*	*

# What is this for? Find non-noded streets!



Non-noded intersections could be either 0-dimensional (simple crossing) or 1-dimensional (shared interior while). **Any** intersection of interiors is a non-noded case, so we test for the “T” condition: any intersection at all.

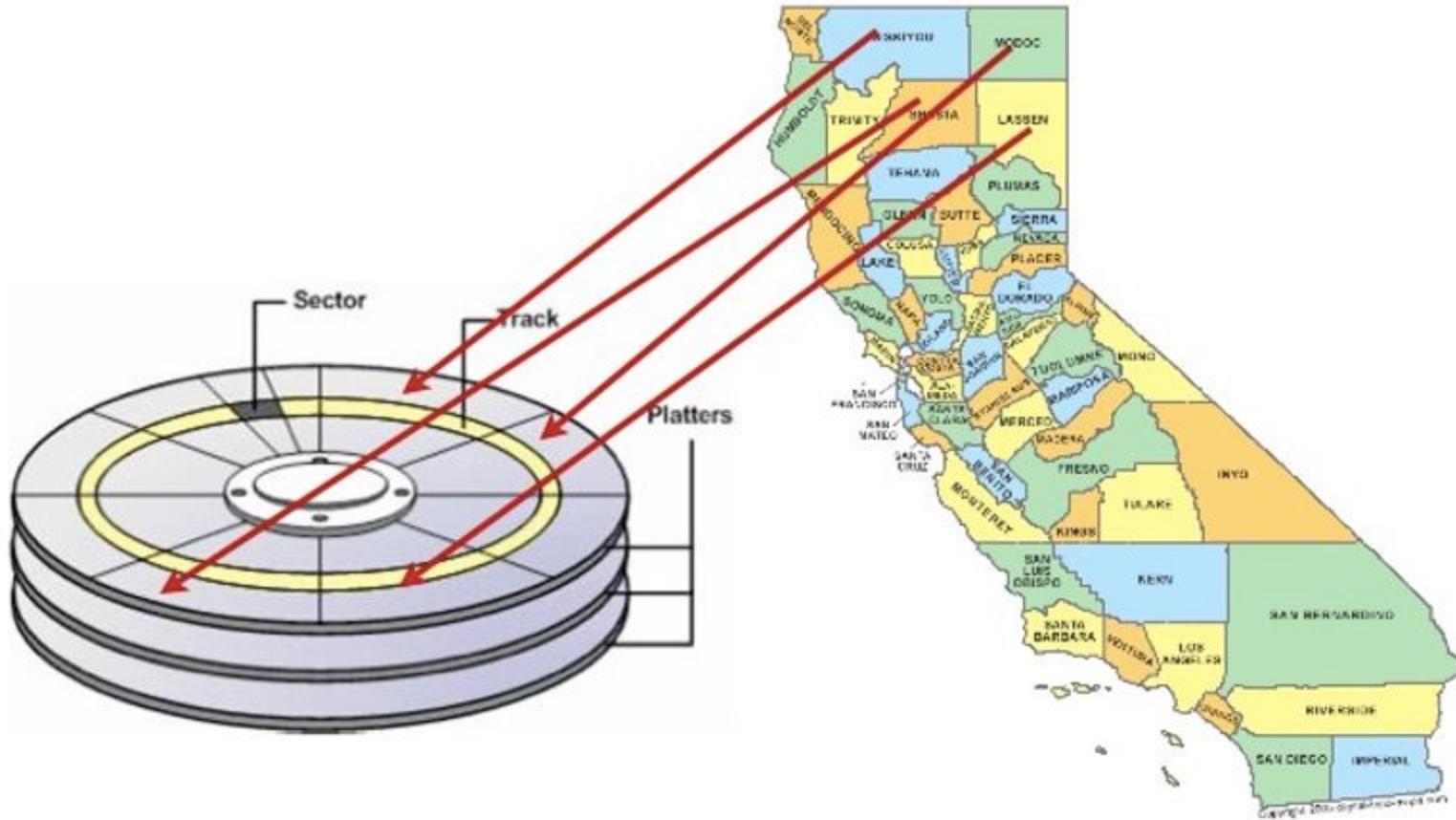
	I	B	E
I	*	*	*
B	*	T	*
E	*	*	*

## What is this for? Find non-noded streets!

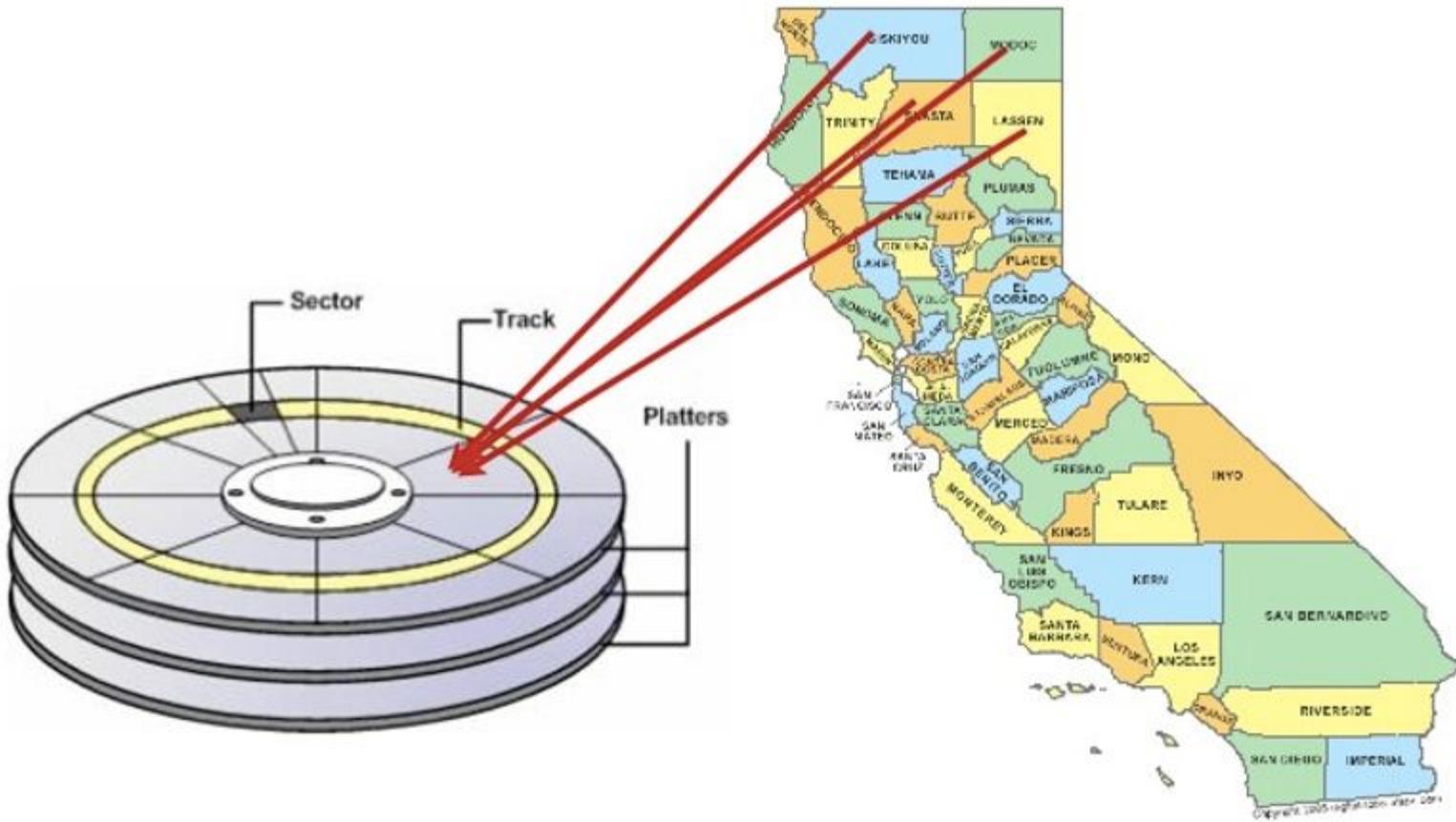
```
SELECT a.gid, b.gid
FROM nyc_streets a,
     nyc_streets b
WHERE ST_Intersects(a.geom,
                    b.geom)
  AND NOT ST_Relate(a.geom,
                     b.geom,
                     '*****T****')
  AND a.gid != b.gid
LIMIT 10;
```

	I	B	E
I	*	*	*
B	*	T	*
E	*	*	*

# **Section 27 - Clustering on Indexes**



## Section 27 - Clustering on Indexes

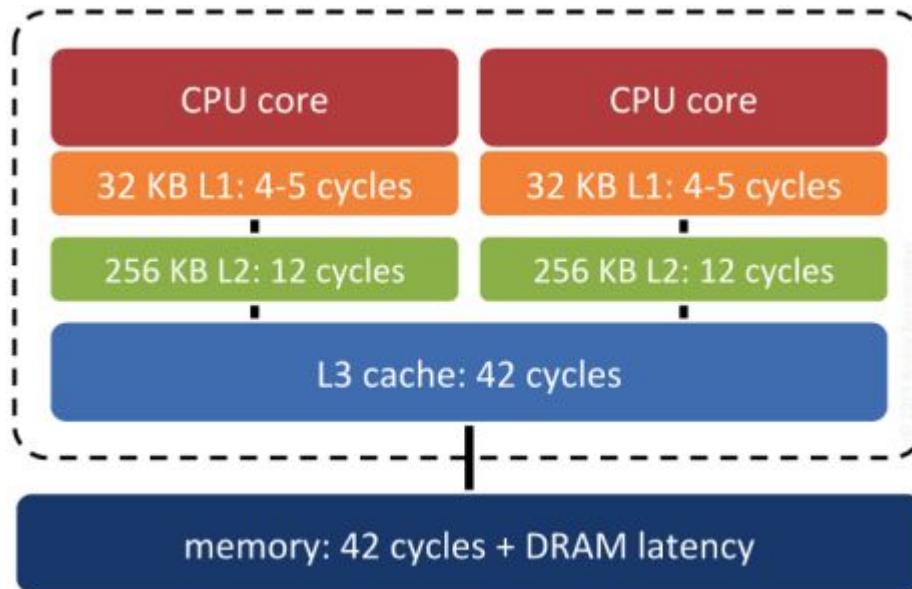


## Cluster using Spatial Index

```
-- Cluster the blocks table based  
-- on their spatial index
```

```
CLUSTER nyc_census_blocks  
USING nyc_census_blocks_geom_gist;
```

# Disk Versus Memory/SSD



# **Section 28 - 3D**

## Dimensions

- **X** - distance from origin along X (east/west) axis
- **Y** - distance from origin along Y (north/south) axis
- **Z** - elevation
- **M** - “measure” of the coordinate, sometimes used to hold “time”, “distance from start”

# Higher Dimension WKT

**POINT (0 0)**

**LINESTRING (0 0, 1 1)**

**POINT Z (0 0 0)**

**LINESTRING Z (0 0 0, 1 1 1)**

**POINT M (0 0 0)**

**LINESTRING M (0 0 0, 1 1 1)**

**POINT ZM (0 0 0 0)**

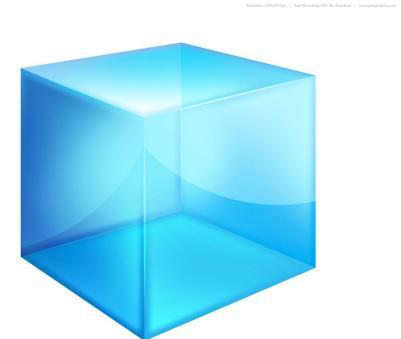
**LINESTRING ZM (0 0 0 0, 1 1 1 1)**

Etc, etc, etc....

# Higher Dimension Types

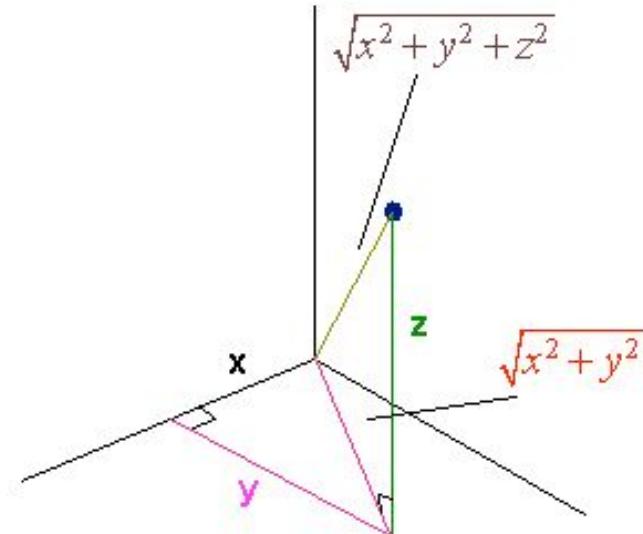
POLYHEDRALSURFACE Z (

```
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),  
((0 0 0, 0 1 0, 0 1 1, 0 0 1, 0 0 0)),  
((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),  
((1 1 1, 1 0 1, 0 0 1, 0 1 1, 1 1 1)),  
((1 1 1, 1 0 1, 1 0 0, 1 1 0, 1 1 1)),  
((1 1 1, 1 1 0, 0 1 0, 0 1 1, 1 1 1)))  
)
```



## 3D Functions

- ST\_3DClosestPoint(g1, g2)
- ST\_3DDistance(g1, g2)
- ST\_3DDWithin(g1, g2, r)
- ST\_3DDFullyWithin(g1, g2, r)
- ST\_3DIntersects(g1, g2)
- ST\_3DLongestLine(g1, g2)
- ST\_3DMaxDistance(g1, g2)
- ST\_3DShortestLine(g1, g2)



## ST\_3DDistance

```
-- sqrt(x^2 + y^2 + z^2) = sqrt(3)
```

```
SELECT ST_3DDistance(  
    'POINT Z (1 1 1)'::geometry,  
    'POINT Z (2 2 2)'::geometry  
) ;
```

1.73205080756888

## N-D Indexes

```
-- Default index build uses 2d  
-- operator class gist_geometry_ops_2d  
CREATE INDEX nyc_streets_gix_2d ON nyc_streets  
USING GIST (geom);
```

```
-- Specify the N-D operator class  
CREATE INDEX nyc_streets_gix_nd ON nyc_streets  
USING GIST (geom gist_geometry_ops_nd);
```



“operator class”  
*gist\_geometry\_ops\_nd*

# Index-enabled Spatial Functions

- ST\_Intersects()
- ST\_Contains()
- ST\_Within()
- ST\_DWithin()
- ST\_ContainsProperly()
- ST\_CoveredBy()
- ST\_Covers()
- ST\_Overlaps()
- ST\_Crosses()
- **ST\_DFullyWithin()**
- **ST\_3DIntersects()**
- **ST\_3DDWithin()**
- **ST\_3DDFullyWithin()**
- ST\_LineCrossingDirection()
- ST\_OrderingEquals()
- ST\_Equals()

## &&& N-D Index Operator

To search using the N-D index explicitly, use the “N-D bounding boxes intersect” operator.

```
-- Returns true (both 3-D on the zero plane)
```

```
SELECT 'POINT Z (1 1 0)'::geometry &&&
       'POLYGON ((0 0 0, 0 2 0, 2 2 0, 2 0 0, 0 0 0))'::geometry;
```

```
-- Returns false (one 2-D one 3-D)
```

```
SELECT 'POINT Z (1 1 1)'::geometry &&&
       'POLYGON ((0 0, 0 2, 2 2, 2 0, 0 0))'::geometry;
```

## &&& N-D Index Operator

To search using the N-D index explicitly, use the “N-D bounding boxes intersect” operator.

```
-- Returns true (the volume around the linestring interacts  
-- with the point)
```

```
SELECT 'LINESTRING Z(0 0 0, 1 1 1)::geometry &&&  
'POINT(0 1 1)::geometry;
```

## &&& N-D Index Operator

N-D index search can be used on 2-D data, at a slight performance penalty.

```
-- N-D index operator
SELECT gid, name
FROM nyc_streets
WHERE geom &&&
      ST_SetSRID('LINESTRING(586785 4492901,587561 4493037)',26918);
```



```
-- 2-D index operator
SELECT gid, name
FROM nyc_streets
WHERE geom &&
      ST_SetSRID('LINESTRING(586785 4492901,587561 4493037)',26918);
```



# **Section 29 - Nearest Neighbor Searching**

# Nearest Neighbor Search

“What is the nearest fire station to this address?”

“What are the 10 nearest gas stations to the current locations?”

# Nearest Neighbor Join

“Add the nearest fire station to every parcel in the table.”

# Nearest Neighbor Search

```
-- The location of Broad St station  
-- SRID=26918;POINT(583571.9 4506714.3)  
SELECT streets.gid, streets.name,  
    ST_Distance(streets.geom,  
        'SRID=26918;POINT(583571.9 4506714.3)' ) AS dist  
FROM nyc_streets streets  
ORDER BY  
    streets.geom <->  
    'SRID=26918;POINT(583571.9 4506714.3)' ::geometry  
LIMIT 3;
```

no WHERE clause

ORDER BY distance

LIMIT clause

## Section 29 - Nearest Neighbor Searching



# Nearest Neighbor Join

```
SELECT subways.gid AS subway_gid,  
       subways.name AS subway,  
       streets.name AS street,  
       streets.gid AS street_gid  
  FROM nyc_subway_stations subways  
CROSS JOIN LATERAL (  
    SELECT streets.name, streets.geom, streets.gid  
      FROM nyc_streets streets  
     ORDER BY streets.geom <-> subways.geom  
    LIMIT 1  
) streets;
```

LATERAL join

outer parameter

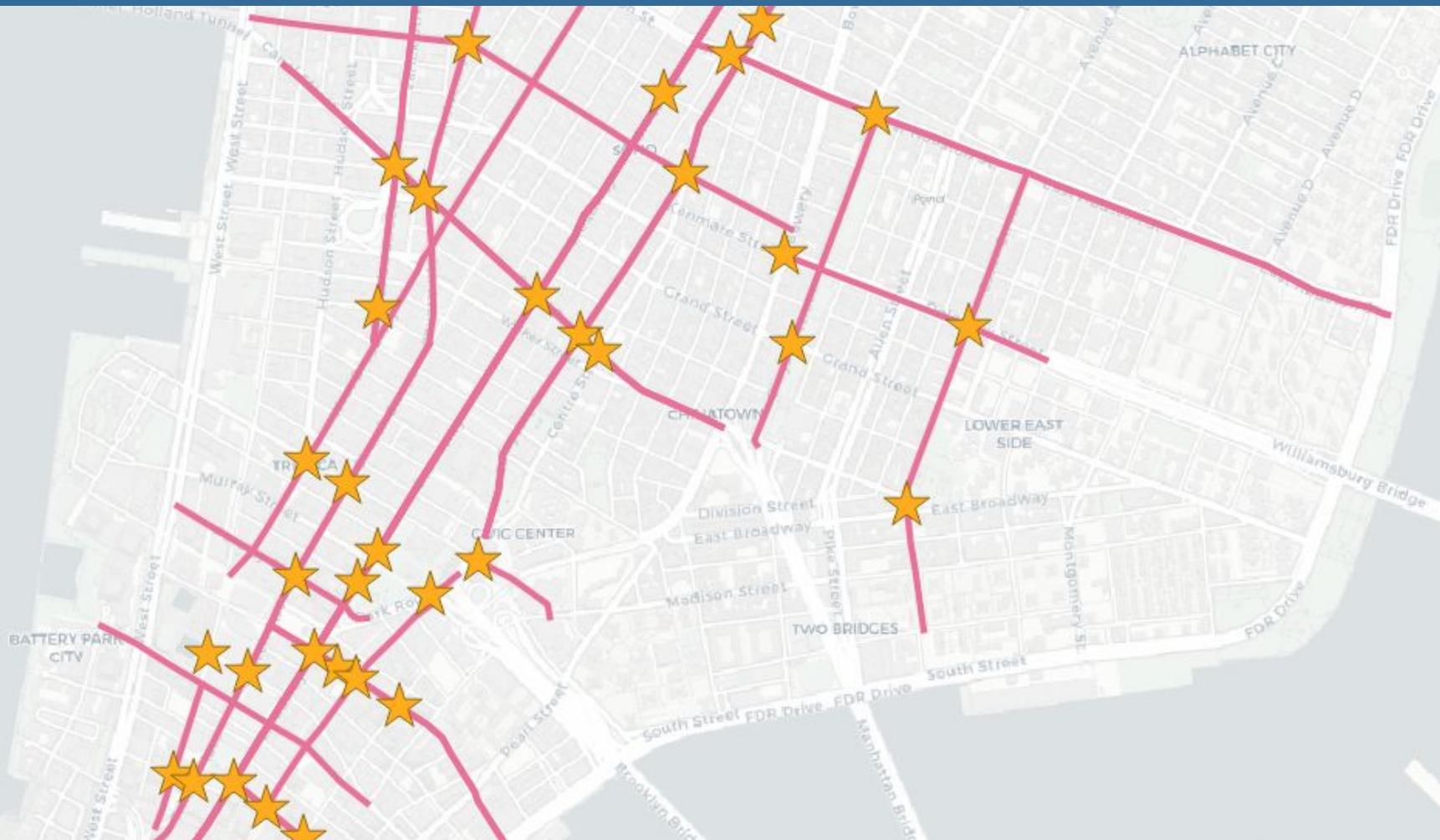
LIMIT clause

ORDER BY distance

## Section 29 - Nearest Neighbor Searching



PostGIS

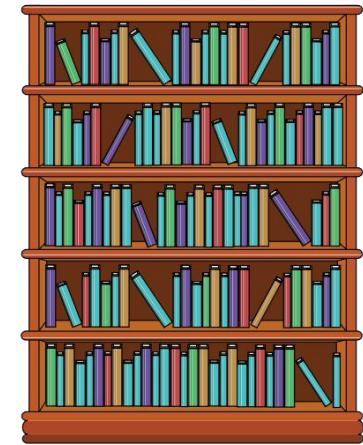


# **Section 30 - Tracking History**

# History Tracking

GIS database models too frequently only model the **current state** of the data. Fortunately, it is easy to **automatically** capture history information **right in the database** without any special changes to client software.

- **Who** created this feature?
- **When** was this feature created?
- What was the **state of the map** on  
YYYY/MM/DD?
- Who **made the most edits** in this area?



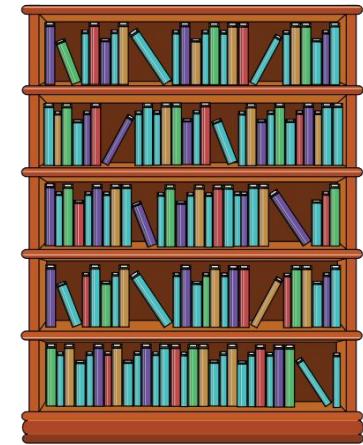
# History Tracking

For every tracked table, add a “history table”:

`<tablename>_history`

History has the columns of the original, plus these extras:

- `created_by` (varchar)
- `deleted_by` (varchar)
- `valid_range` (tstzrange)



# History Tracking

Add triggers to the working table that will keep the state of the history table in sync with the state of the working table.

ON INSERT to working table	ON DELETE from working table	ON UPDATE of working table
<p>Add record to history table:</p> <ul style="list-style-type: none"><li>• Primary key of working table to link history to working record</li><li>• “Created by” record with current user.</li><li>• Valid range from now to infinity.</li></ul>	<p>Update live record in working table:</p> <ul style="list-style-type: none"><li>• “Deleted by” record with current user.</li><li>• Set end of valid range to now.</li></ul>	<p>Run the delete routine on the current live history record. Sets “deleted by” and closes the valid range.</p> <p>Run the insert routine with the new state of the working record. Sets “created by” and starts new open valid range.</p>

```
CREATE TABLE nyc_streets_history (
    hid           SERIAL PRIMARY KEY,
    gid           INTEGER,
    id            FLOAT8,
    name          VARCHAR(200),
    oneway        VARCHAR(10),
    type          VARCHAR(50),
    geom          GEOMETRY(MultiLinestring, 26918),
    valid_range   TSTZRANGE,
    created_by    VARCHAR(32),
    deleted_by    VARCHAR(32)
);
```

## Add current state to history

```
INSERT INTO nyc_streets_history  
(gid, id, name, oneway, type,  
geom, created, created_by)  
SELECT  
    gid, id, name, oneway,  
    type, geom,  
    tstzrange(now(), NULL),  
    current_user  
FROM nyc_streets;
```

```
CREATE OR REPLACE FUNCTION nyc_streets_update() RETURNS trigger AS
$$
BEGIN

    UPDATE nyc_streets_history
        SET deleted = current_timestamp, deleted_by = current_user
    WHERE valid_range @> current_timestamp AND gid = OLD.gid;

    INSERT INTO nyc_streets_history
        (gid, id, name, oneway, type, geom, created, created_by)
    VALUES
        (NEW.gid, NEW.id, NEW.name, NEW.oneway, NEW.type, NEW.geom,
         tstzrange(current_timestamp, NULL), current_user);

    RETURN NEW;

END;
$$
LANGUAGE plpgsql;
```

# Demonstrate History Tracking

- Edit some features and see what happens
  - Layer > Toggle Editing 
- Look at nyc\_streets\_history and see records added / altered as features are added / altered / deleted.
- Create the view showing the state of the table 10 minutes ago.
- Watch the view in QGIS.



# **Section 31 - Basic PostgreSQL Tuning**

# Dividing up RAM

- **shared\_buffers** - allocated **once** and shared by all connections
- **work\_mem** - allocated for **each connection**
- **maintenance\_work\_mem** - allocated for each VACUUM process



These settings actually change how much memory the database uses when running, so be judicious. The `work_mem` in particular is easy to over-allocate if you under-estimate the number of concurrent connections your database will be handling.

# Tuning for the Query Planner

These settings won't have any impact on the amount of resources the database uses, they just alter the kinds of query plans that are used.

<b>random_page_cost</b>	<b>seq_page_cost</b>	<b>effective_cache_size</b>
Relative to the sequential page cost and other fixed costs. Random access on spinning media is costly. Random access in RAM is fast. If your disks are fast, or SSD, this value should be lower.	For most media a sequential access is cheap, because the system will “prefetch” data ahead of the current request, so it tends to have the next needed datum in memory already. Leave this value at the default.	Operating systems cache heavily used portions of disk (like, say, indexes) into memory. This value should reflect how much “free” space you expect the OS to have, after all database and other system use is accounted for.

## Tuning on the Fly

Some settings can be changed at run-time, which is useful if you are testing changes, and also for some workloads.

Building a new index? You can speed up index builds by increasing **work\_mem** before running the build.

```
SET work_mem TO '1GB';
CREATE INDEX ...;
SET work_mem TO '32MB';
```

# **Section 32 - PostgreSQL Security**

# Read-Only Role

```
-- A user account for the web app
CREATE USER app1;
-- Web app needs access to specific
-- data tables
GRANT SELECT ON nyc_streets TO app1;

-- A generic role for access to PostGIS
-- functionality
CREATE ROLE postgis_reader INHERIT;
-- Give that role to the web app
GRANT postgis_reader TO app1;
```

# Read-Only Role

-- This works!

```
SELECT * FROM nyc_streets LIMIT 1;
```

-- This doesn't work!

```
SELECT ST_AsText(ST_Transform(geom, 4326))  
FROM nyc_streets LIMIT 1;
```

## Why??!

# Read-Only Role

```
GRANT SELECT  
    ON geometry_columns TO postgis_reader;  
GRANT SELECT  
    ON geography_columns TO postgis_reader;  
GRANT SELECT  
    ON spatial_ref_sys TO postgis_reader;
```

-- This works now!

```
SELECT  
    ST_AsText(ST_Transform(geom, 4326))  
FROM nyc_streets LIMIT 1;
```

# Read-Write Role

```
-- Make a postgis writer role
CREATE ROLE postgis_writer;

-- Start by giving it the
-- postgis_reader powers
GRANT postgis_reader TO postgis_writer;

-- OPTIONAL insert/update/delete powers for
-- the spatial_ref_sys table
GRANT INSERT, UPDATE, DELETE
ON spatial_ref_sys TO postgis_writer;
```

## Read-Write Role

```
-- Add insert/update/delete abilities to  
-- our web application
```

```
GRANT INSERT, UPDATE, DELETE  
ON nyc_streets TO postgis_writer;
```

```
-- Make app1 a PostGIS writer to see  
-- if it works!
```

```
GRANT postgis_writer TO app1;
```

# Encryption in PostgreSQL

- SSL connection encryption (postgresql.conf)
  - certificate-based authentication
- data encryption (pgcrypto)
  - symmetric
  - public key
  - hashes

# Enabling SSL Connections

```
# Create a new certificate, filling out  
# the certification info as prompted  
openssl req -new -text -out server.req  
  
# Strip the passphrase from the certificate  
openssl rsa -in privkey.pem -out server.key  
  
# Convert the certificate into a self-signed cert  
openssl req -x509 -in server.req -text \  
    -key server.key -out server.crt  
  
# Set the permission of the key  
# to private read/write  
chmod og-rwx server.key
```

# Enabling SSL Connections

- Copy the server.crt and server.key PostgreSQL data directory.
- Enable SSL support in the postgresql.conf file by turning the “ssl” parameter to “on”.
- Re-start PostgreSQL

# Enabling Data Encryption

```
CREATE EXTENSION pgcrypto;
-- encrypt a string using blowfish (bf)
SELECT encrypt('this is a test phrase',      -- string
               'mykey',                  -- key
               'bf');                   -- cipher

-- round-trip a string using blowfish (bf)
SELECT decrypt(encrypt('this is a test phrase',
                       'mykey',
                       'bf'),
               'mykey',
               'bf');
```

# Authentication (pg\_hba.conf)

- password (MD5)
- kerberos
  - SSPI (MicroSoft AD)
  - GSSAPI
- LDAP
- SSL certificate
- PAM (Solaris, Linux)

# Authentication (pg\_hba.conf)

```
# TYPE  DATABASE        USER        CIDR-ADDRESS      METHOD  
  
# "local" is for Unix domain socket connections only  
local   all            all          trust  
  
# IPv4 local connections:  
host    all            all          127.0.0.1/32      trust  
  
# IPv4 local network connections:  
host    all            all          192.168.1.0/24    md5  
  
# IPv6 local connections:  
host    all            all          ::1/128          trust  
  
# remote connections for nyc database only  
host    nyc           all          10.10.1.0/16     ldap
```

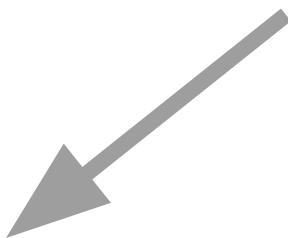
# **Section 33 - Schemas**

# Why Schemas?

- data management
  - backups / upgrades
- user/application isolation
  - application upgrades

# Why Schemas?

- ›  nyc
  - ›  Casts
  - ›  Catalogs
  - ›  Event Triggers
  - ›  Extensions
  - ›  Foreign Data Wrappers
  - ›  Languages
  - ›  Publications
  - ›  Schemas (2)
    - ›  public
    - ›  test
  - ›  Subscriptions



Without schemas, tables default to being put into the “public” catch-all schema. This is fine if your database only has one user, but if you have multiple users or applications, segmenting them into schemas will make access control and general cleanliness much easier to achieve.

# Common Schema Tasks

```
-- create a schema
```

```
CREATE SCHEMA census;
```

```
-- move a table into a schema
```

```
ALTER TABLE nyc_census_blocks SET SCHEMA census;
```

```
-- reference a table with schema
```

```
SELECT * FROM census.nyc_census_blocks LIMIT 1;
```

```
-- add schema to search path
```

```
SET search_path = census, public;
```

```
-- permanently add schema to search path
```

```
ALTER USER postgres  
  SET search_path = census, public;
```

# Setting a User Schema

```
-- create a user  
CREATE USER myuser WITH ROLE postgis_writer;  
-- create a user schema  
CREATE SCHEMA myuser AUTHORIZATION myuser;
```

- connect as “myuser”
- see where tables are created?
  - **in the “myuser” schema!**
- see what the default visibility is?
- what is the “search\_path” set to?
  - **search\_path = \$user, public**



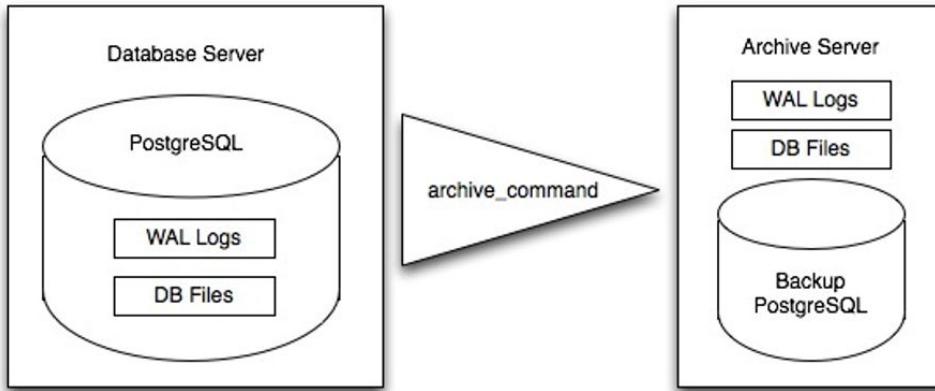
If there is a schema with the same name as a user, the default search\_path results in all new tables going into that schema and default searches **finding** those tables.

# **Section 34 - Backup and Restore**

# Backup / Restore Options

- **pg\_dump / pg\_restore**
  - portable, simple, classic
- **pg\_backrest**
  - scalable, incremental, durable
  - true backup via snapshot and WAL archive
- **online replication**
  - not really a backup
  - scalable, incremental, real-time

# Generic Backup Theory



- periodically take a snapshot of all the database files pg\_dump
- continually copy the WAL files into a safe storage location
- restore consists of taking snapshot and playing WAL files back until point-in-time-recovery is reached
- utility like pg\_backrest automates many common desires and avoids common pitfalls
  - store snapshots / WAL on object store
  - avoid failed archive commands

# Two Kinds of Upgrade

- PostgreSQL upgrades
  - dump/restore
  - pg\_upgrade
- PostGIS upgrades
  - soft upgrade
  - hard upgrade

# PostgreSQL Upgrades

- **Minor upgrades**
  - 12.2 → 12.3
  - install new software, restart
- **Major upgrades**
  - 12 → 13
  - dump/restore or pg\_upgrade

# PostgreSQL Dump/Restore

- pg\_dumpall
- install new PostgreSQL
- install identical PostGIS
  - why? PostGIS function signatures
- pg\_restore

# PostgreSQL pg\_upgrade

- install new PostgreSQL
- install identical PostGIS
- create new data area
- pg\_upgrade

```
pg_upgrade
--old-datadir "/var/lib/postgres/12/data"
--new-datadir "/var/lib/postgres/13/data"
--old-bindir "/usr/pgsql/12/bin"
--new-bindir "/usr/pgsql/13/bin"
```

# PostGIS Upgrades

- Minor upgrades (soft upgrade)
  - 3.1.1 → 3.1.2 or 3.0.1 → 3.1.0
- install new software
  - **ALTER EXTENSION postgis UPDATE TO '3.1.0';**