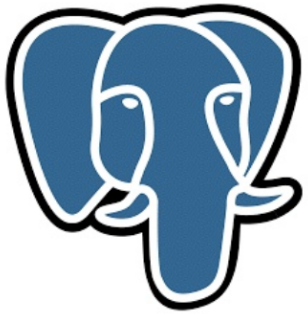
Lab - PostGIS

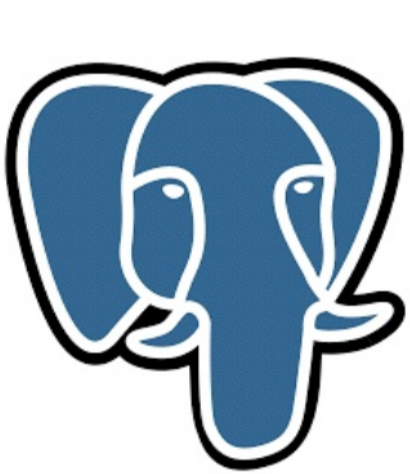
PostgreSQL

- ORDBMS
- Open Source
- Easy to add custom functions
- Support many third-party library



PostgreSQL

What is PostGIS



PostgreSQL

+

GIS

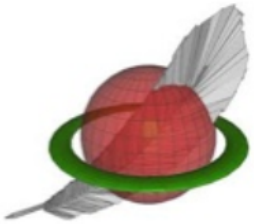
=

PostGIS

(spatial types, indexes, and functions)

Other Spatial DB

- Spatialite
- Oracle spatial
- MySQL/MariaDB spatial
- Microsoft SQLServer spatial database



ORACLE SPATIAL

MySQL MariaDB

Microsoft SQL Server

Installation Guide

- For Windows -
(Mac OS is later introduced)

PostgreSQL Installation

- Find the corresponding OS version From https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
  - ☆ Install version 11.x

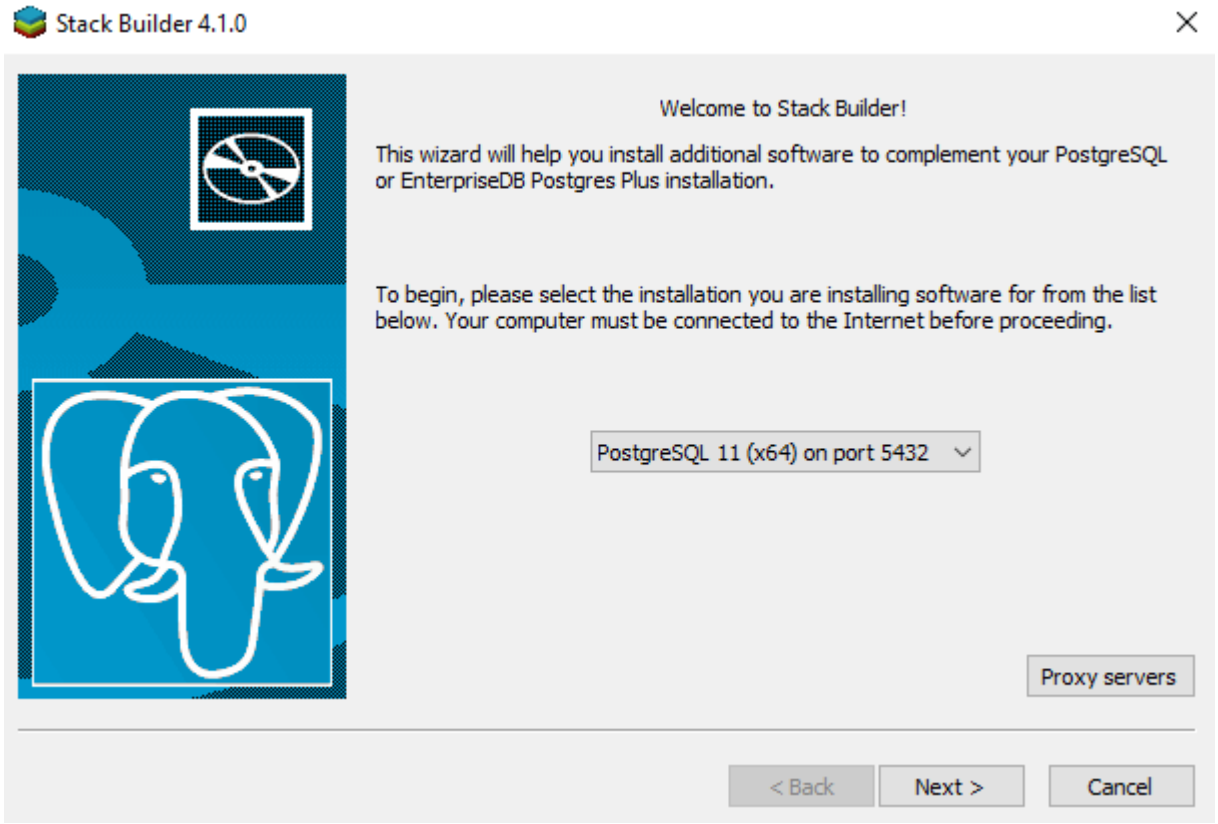
PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
11.0	N/A	N/A	Download	Download	N/A
10.5	Download	Download	Download	Download	Download
9.6.10	Download	Download	Download	Download	Download
9.5.14	Download	Download	Download	Download	Download
9.4.19	Download	Download	Download	Download	Download
9.3.24	Download	Download	Download	Download	Download

PostgreSQL Installation

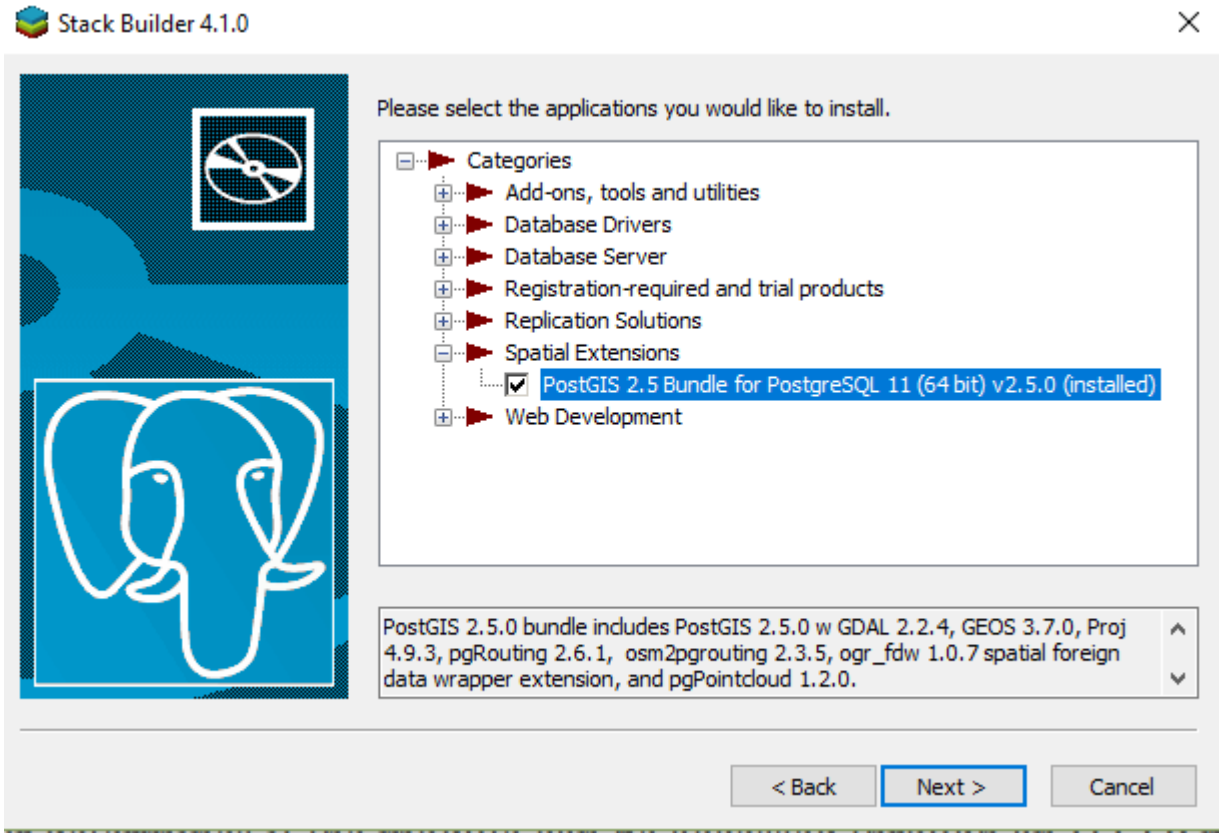
- Follow the installation guide

https://www.enterprisedb.com/docs/en/11.0/PG_Inst_Guide_v11/PostgreSQL_Installation_Guide.1.08.html

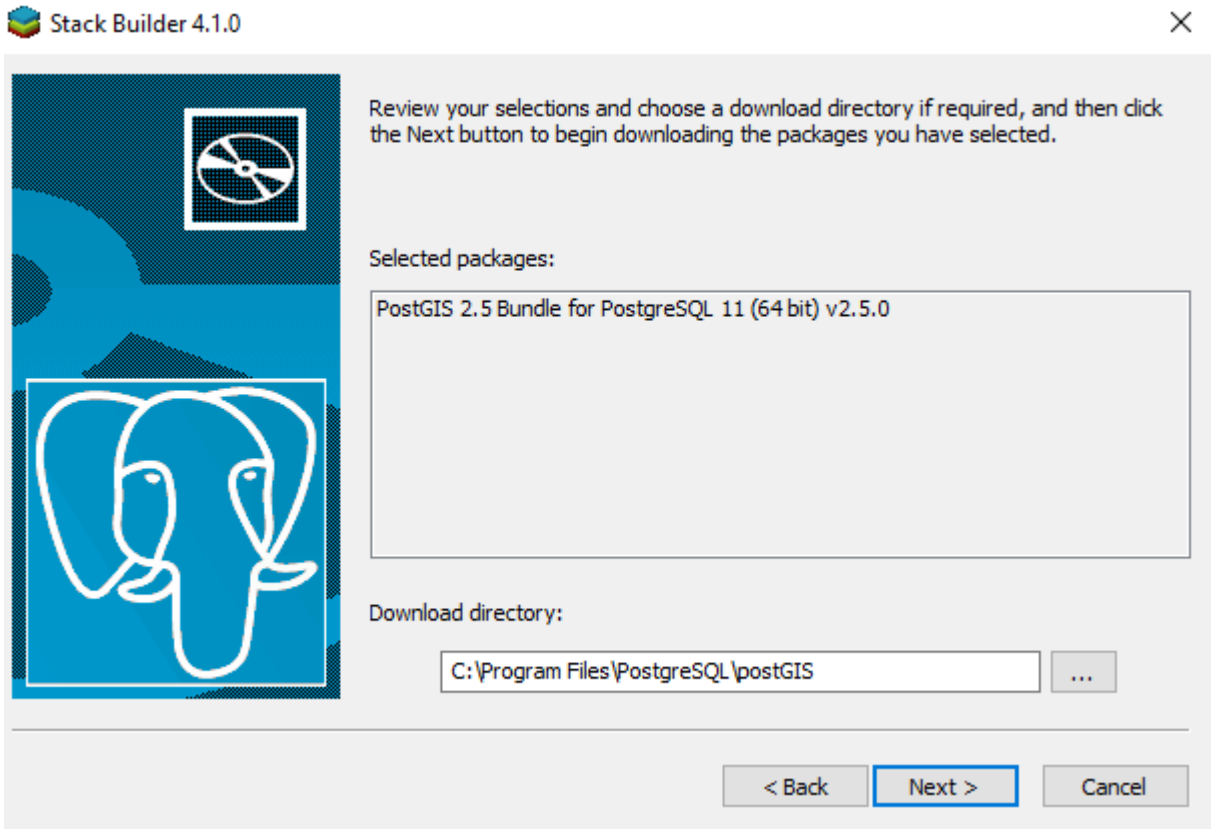
Stack Builder



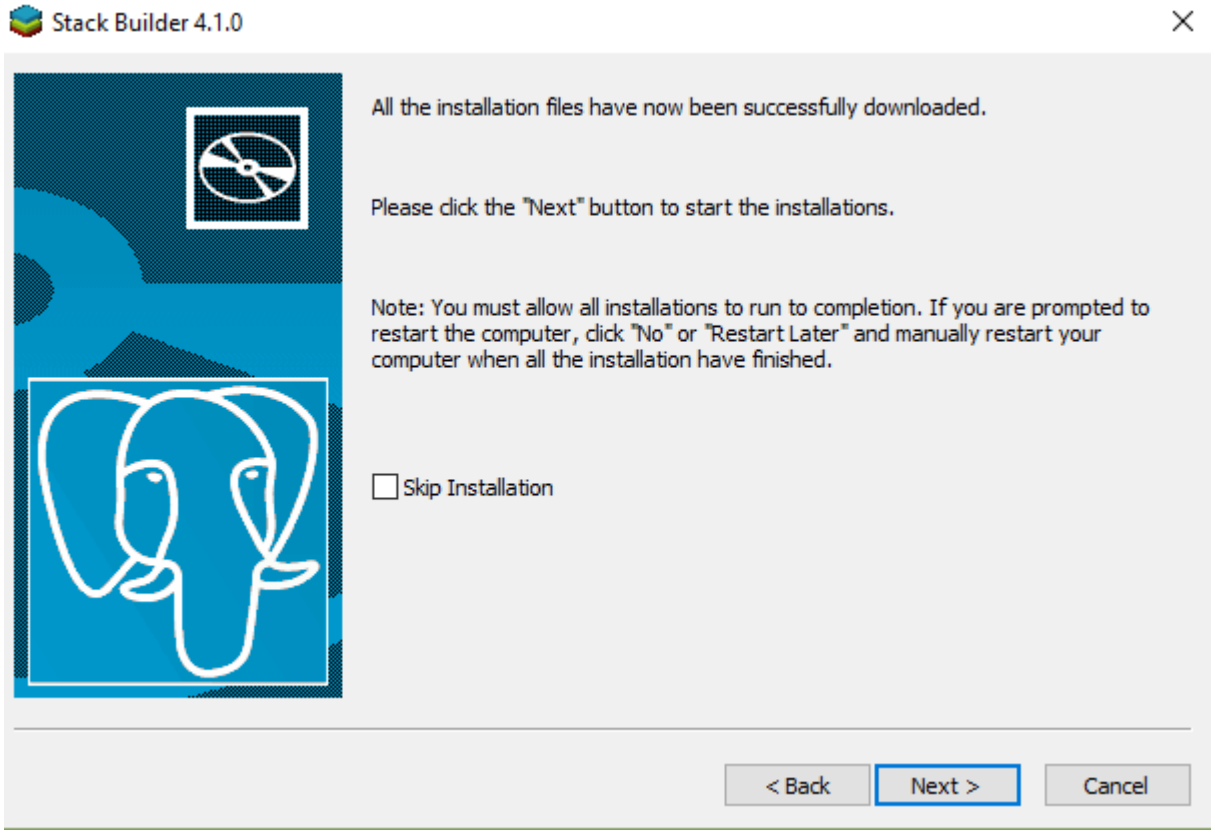
Stack Builder



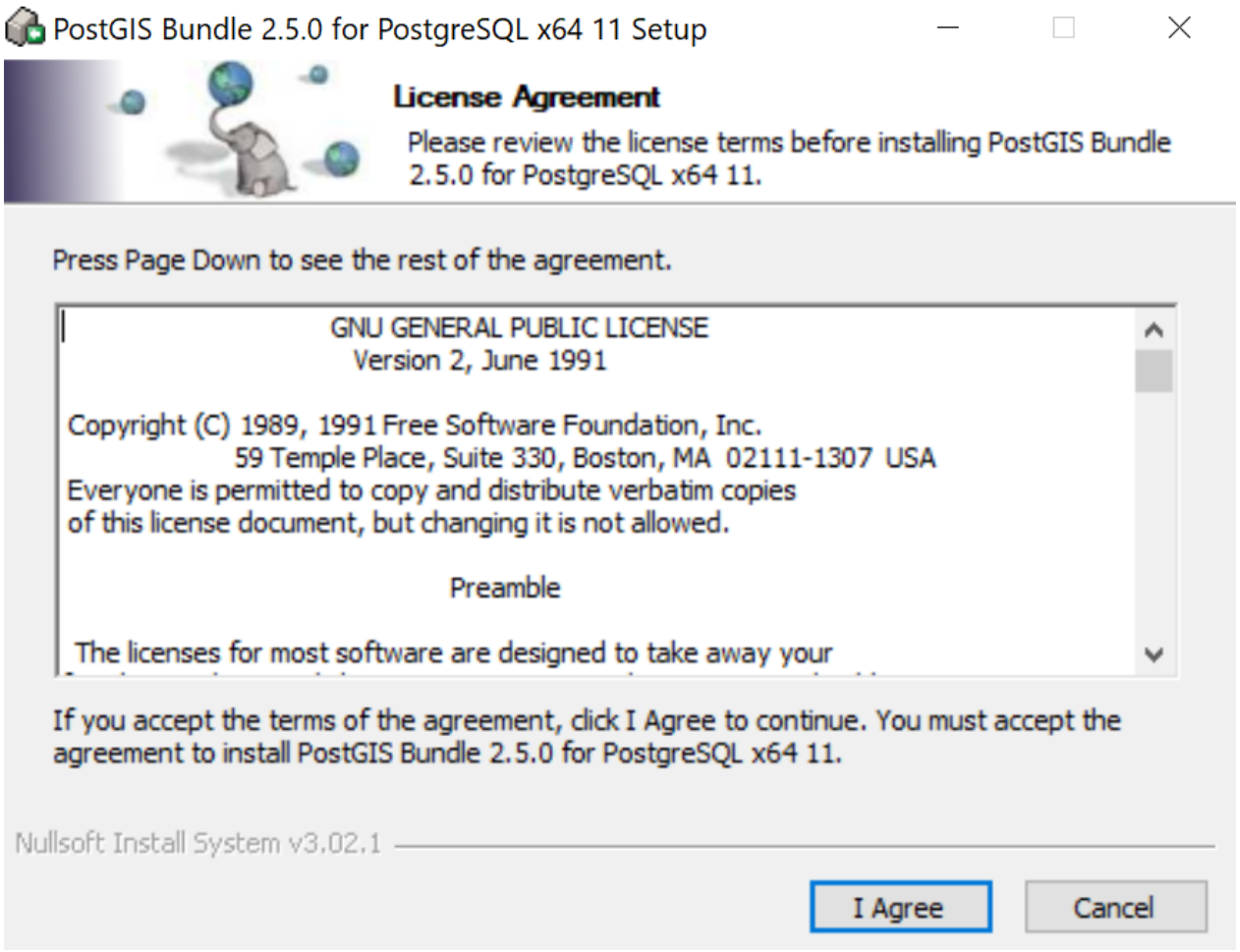
Stack Builder



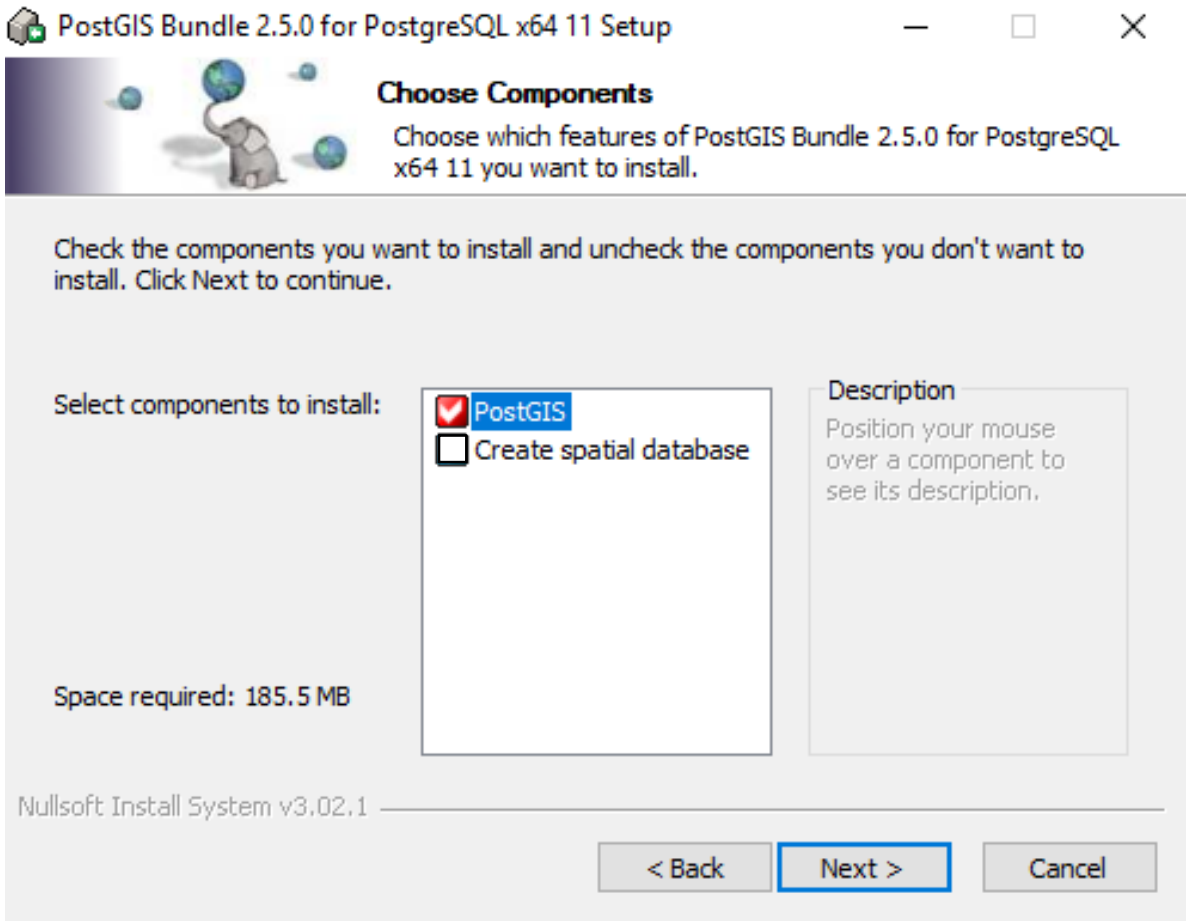
Stack Builder



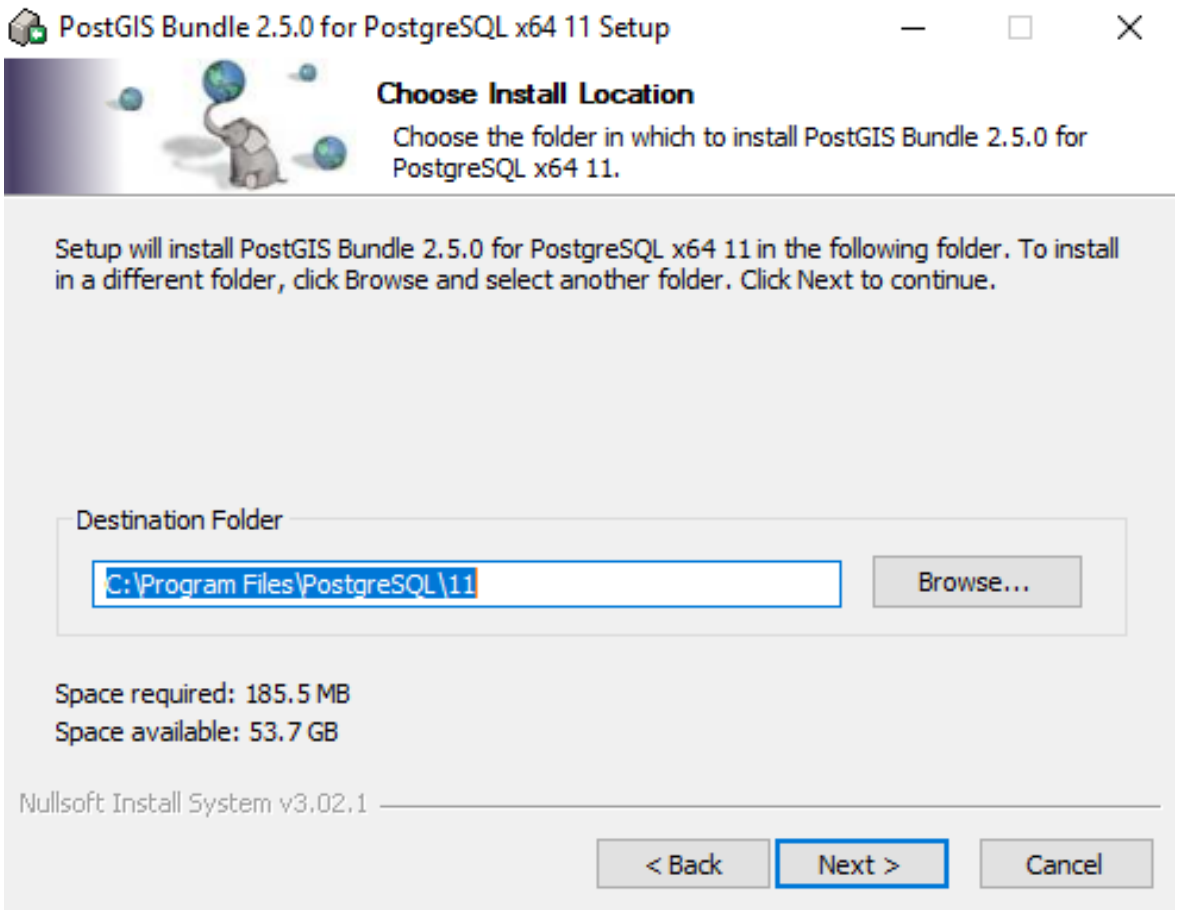
PostGIS Installer



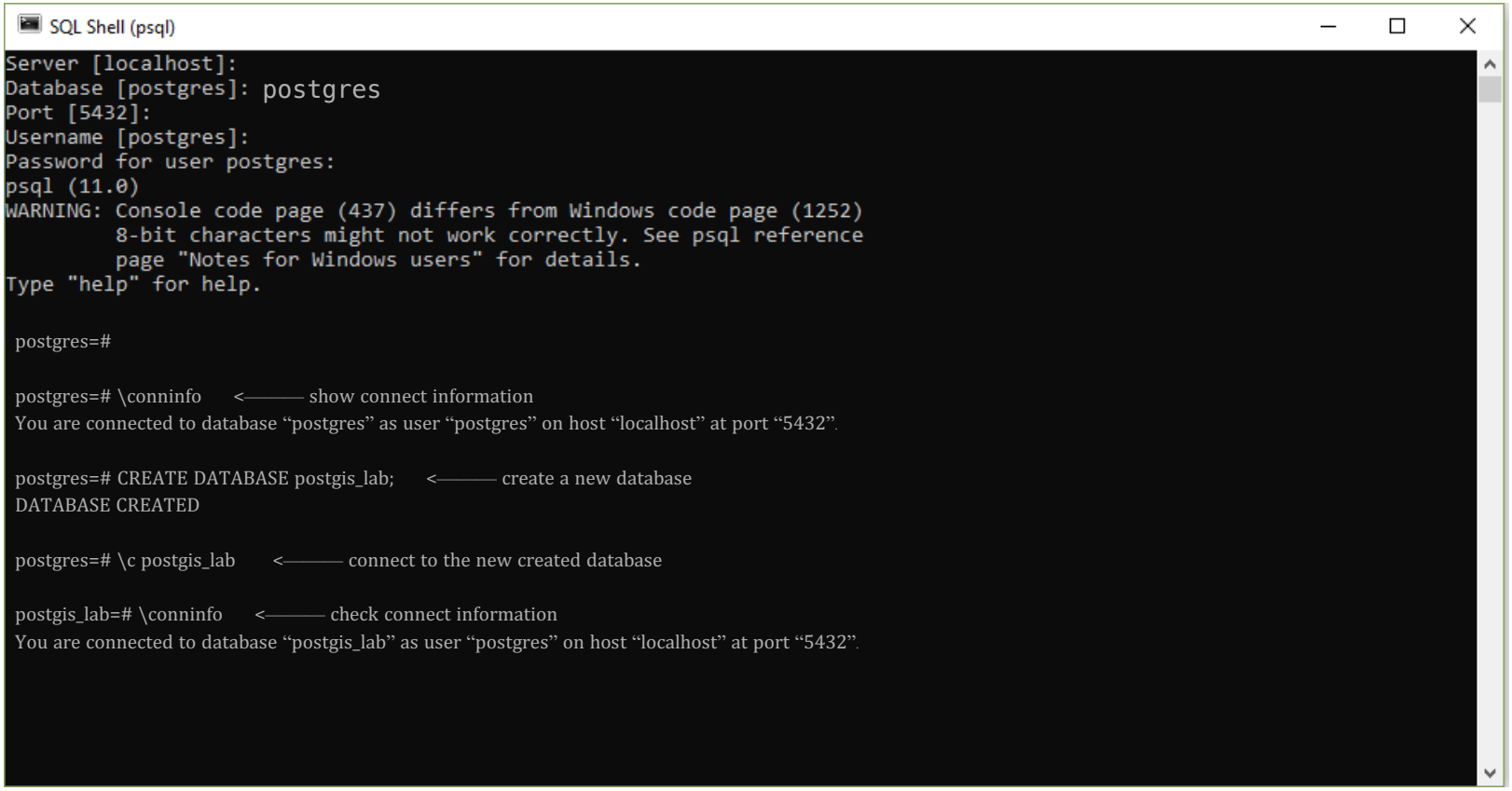
PostGIS Installer



PostGIS Installer



After installing, open psql



Installation Guide

- For Mac OS -

Homebrew Installation

- Website: https://brew.sh/

Postgres & PostGIS Installation

- Remove old versions (Optional)

```
$ brew uninstall --force postgis postgresql
$ rm -rf /usr/local/var/postgres
```

- Install new ones

```
$ brew install postgres postgis
```

- Start server

```
$ pg_ctl -D /usr/local/var/postgres start
```

- Shutdown server

```
$ pg_ctl -D /usr/local/var/postgres -l logfile stop
```


Reference: https://medium.com/@Umesh_Kafle/postgresql-and-postgis-installation-in-mac-os-87fa98a6814d

Post installation

- Create database storage

  `$ initdb /usr/local/var/postgres`

  - If terminal shows an error

```
initdb: directory "/usr/local/var/postgres" exists but is not empty
If you want to create a new database system, either remove or empty
the directory "/usr/local/var/postgres" or run initdb
with an argument other than "/usr/local/var/postgres".
```

  - Remove old database files `$ rm -r /usr/local/var/postgres`
  - Re-run the initdb command `$ initdb /usr/local/var/postgres`

- Create a new database

  `$ createdb postgis_lab`

- Operate the created database with psql

  `$ psql postgis_lab`

  → `postgis_lab=#`

Reference: https://medium.com/@Umesh_Kafle/postgresql-and-postgis-installation-in-mac-os-87fa98a6814d

PSQL

SQL shell for PostgreSQL

PSQL

- Useful commands in psql (`postgis_lab=#`)
  - Check for all command `\?`
  - List table, view `\d`
  - Connect to another database `\c {DBNAME}`
  - Change the current working directory `\cd [DIR]`
  - Quit `\q` or `CTRL+D`

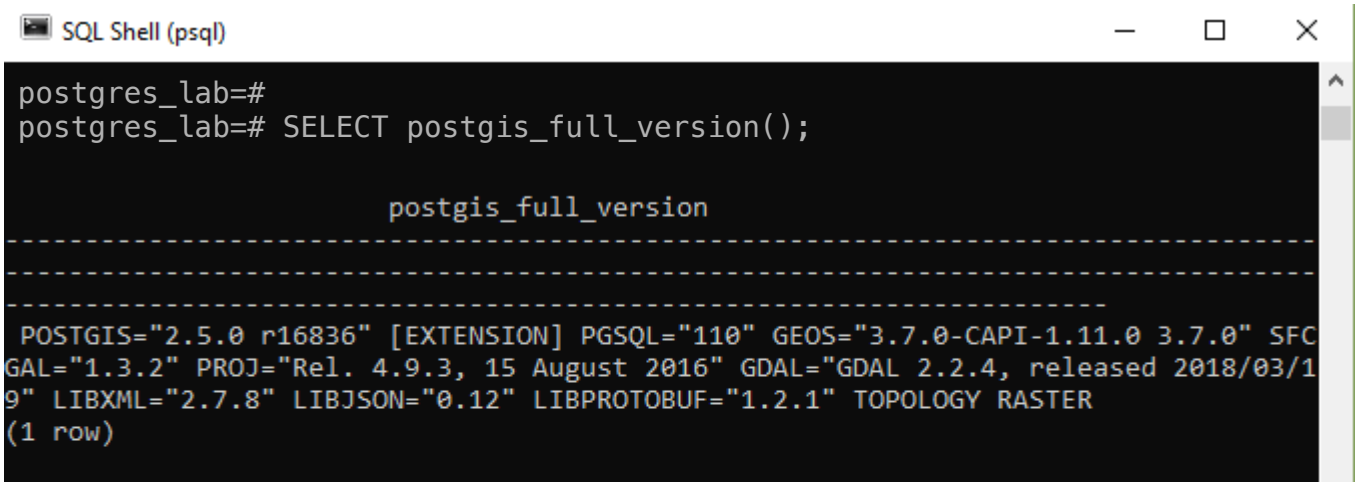
Enable GIS Function in Postgres

- Load PostGIS spatial extension

```
# CREATE EXTENSION postgis;
```

- Confirm whether PostGIS installed successfully

```
# SELECT postgis_full_version();
```



Loading Spatial Data

Spatial Data format

- **Shapefile**: a popular vector data for GIS format
- **GeoJSON:** JSON format for the spatial extended specification
- **GML:** XML format for representing spatial feature information.
- **KML:** the spatial XML format used by Google Earth

Shapefiles

- **.shp :** shape format; the feature geometry itself
- **.shx :** shape index format; a positional index of the feature geometry
- **.dbf :** attribute format; columnar attributes for each shape
- **(.prj)** : projection format; the coordinate system and projection information

SRID(Spatial Reference Identifier)

- It packs all the information about a map projection (.prj) into a single number.

```
PROJCS["NAD83 / UTM zone 18N",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],
      AUTHORITY["EPSG","6269"]],
    PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4269"]],
  UNIT["metre",1,AUTHORITY["EPSG","9001"]],
  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],
  AUTHORITY["EPSG","26918"],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH]]
```

SRID (Spatial Reference Identifier)

- Plug the contents of the .prj file into http://prj2epsg.org.
- This will give you the number that most closely match your projection definition.

```
$ python3 esriprj2standards.py [prj file]
```

Loading data into database (1)

- Go to cmd/terminal
- Use shp2pgsql tool to convert shapefile into database.

```
$ shp2pgsql -s SRID FILE.shp | psql -d DBNAME -U USERNAME
```

- (Find Username)

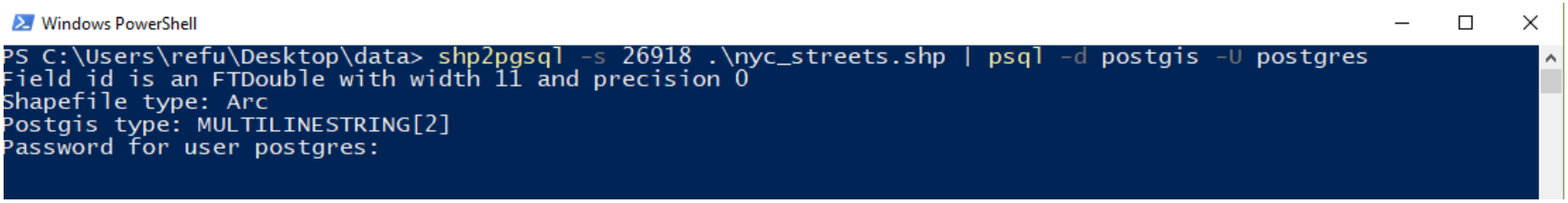
```
> SELECT usename from pg_user;
```

```
postgis_lab=# SELECT usename from pg_user;
 usename
---------
 yenhao
(1 row)
```

```
$ shp2pgsql -s 26918 nyc_streets.shp | psql -d postgis_lab -U postgres
```

For Windows

```
INSERT INTO "nyc_streets" ("id","name","oneway","type",geom) VALUES ('1020','FDR
Dr',NULL,'residential','0105000020266900000100000001020000000200000C9D2C4E351F52
141E91561A27037514170DAEFAB3FF52141C419EE3D72375141');
INSERT INTO "nyc_streets" ("id","name","oneway","type",geom) VALUES ('1152','FDR
Dr',NULL,'residential','0105000020266900000100000001020000000200000004F6C8F179AF72
141E23CDE3EEB375141B95D381AFAF721417829E25FFE375141');
```



Loading data into database (2)

- After data loaded

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
COMMIT
ANALYZE
```

- Check data inserted by

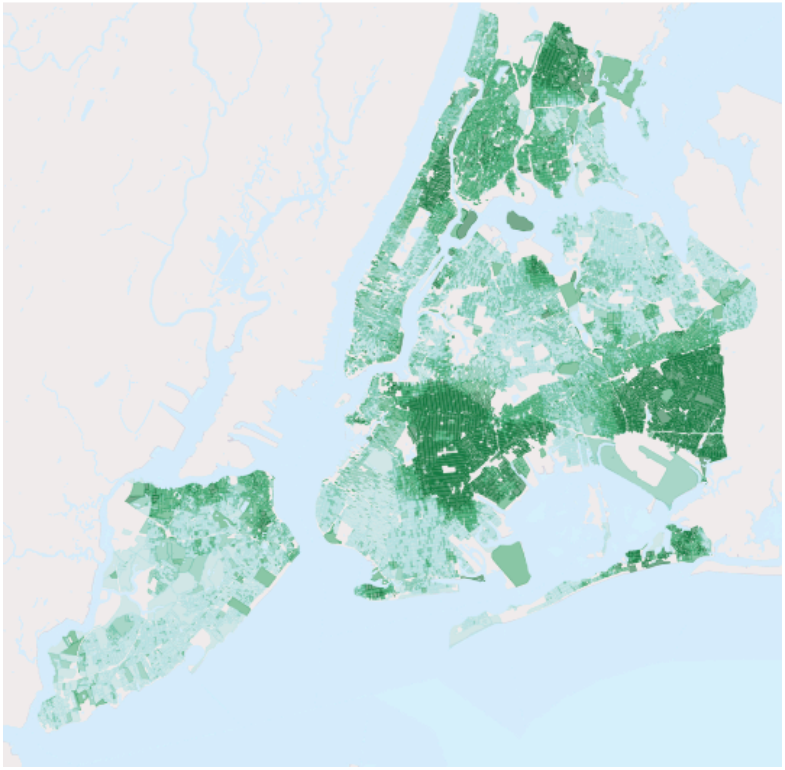
```
> \d
```

```
postgis_lab=# \d
                List of relations
 Schema |        Name          |   Type   | Owner
--------+----------------------+----------+--------
 public | geography_columns    | view     | yenhao
 public | geometry_columns     | view     | yenhao
 public | nyc_streets          | table    | yenhao
 public | nyc_streets_gid_seq  | sequence | yenhao
 public | spatial_ref_sys      | table    | yenhao
(5 rows)
```

- Repeat for the rest of data (nyc_subway_stations/ nyc_neighborhoods/ nyc_homicides/ nyc_census_blocks.shx)

nyc_census_blocks

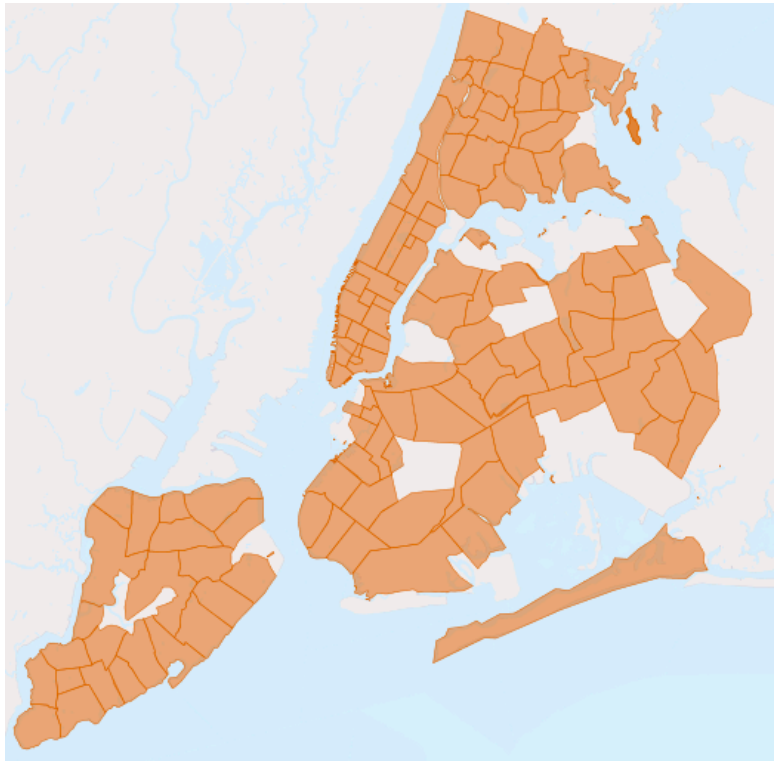
- A census block is the smallest geography for which census data is reported



blkid
popn_total
popn_white
popn_black
popn_nativ
popn_asian
popn_other
boroname
geom

nyc_neighborhoods

- Neighborhoods are social constructs that do not follow lines laid down by the government.



name
boroname
geom

nyc_streets

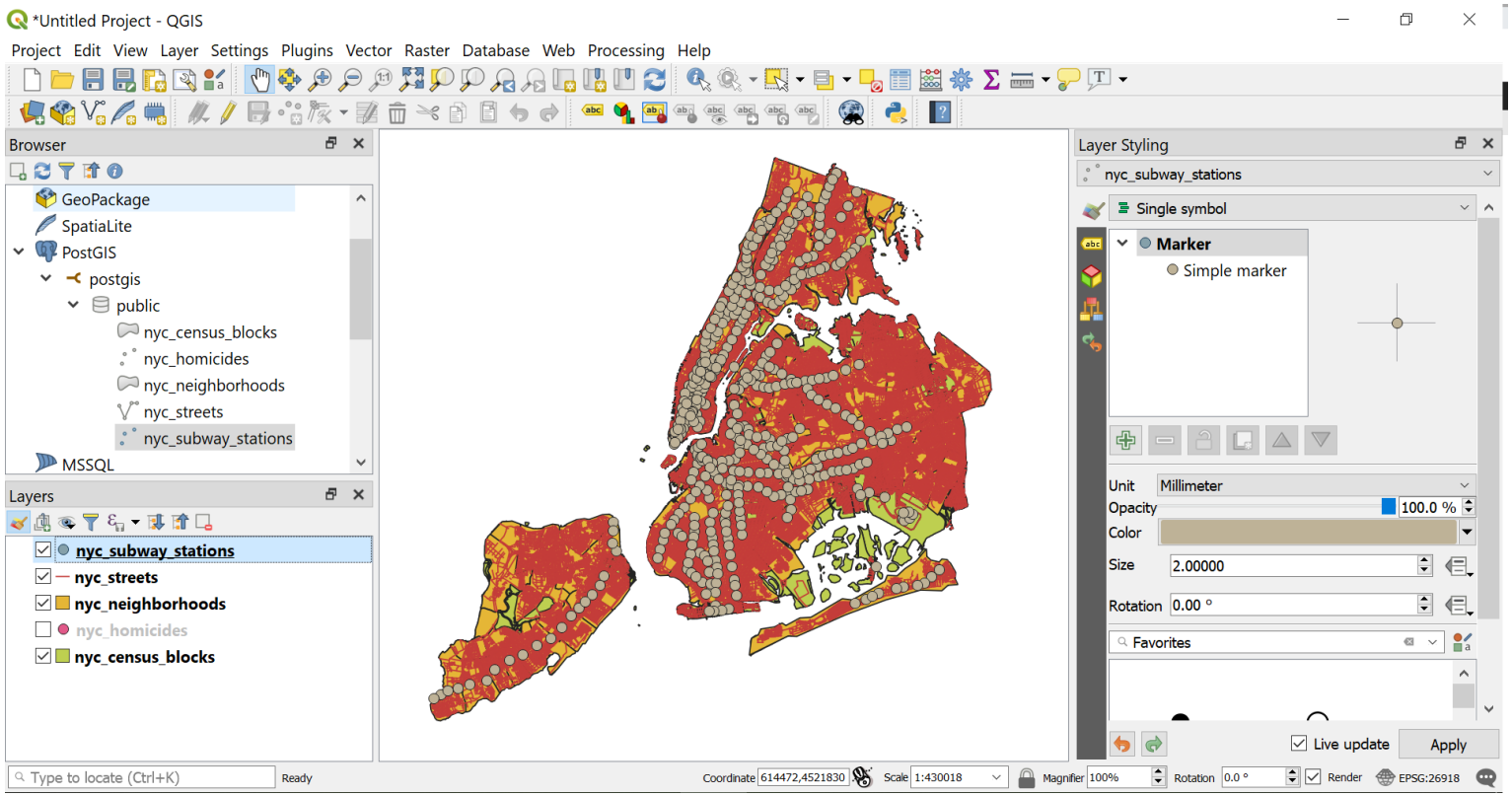
name
oneway
type
geom

nyc_subway_stations

name
borough
routes
transfers
express
geom

QGIS

- You can use QGIS to visualize data.



Spatial SQL

Basic PostgreSQL

- **Find column_names of table**

```
postgis=# SELECT
postgis-#    COLUMN_NAME
postgis-# FROM
postgis-#    information_schema.COLUMNS
postgis-# WHERE
postgis-#    TABLE_NAME = 'nyc_neighborhoods';
 column_name
-------------
 gid
 boroname
 name
 geom
(4 rows)
```

- **Or**

```
> \d table_name
```

Basic PostgreSQL

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

```
postgis=# SELECT name
postgis-#        FROM nyc_neighborhoods
postgis-#        WHERE boroname = 'Brooklyn';
           name
-------------------------
 Bensonhurst
 Bay Ridge
 Boerum Hill
 Cobble Hill
 Downtown
 Sunset Park
 Borough Park
 East Brooklyn
 Flatbush
 Park Slope
 Williamsburg
 Canarsie
 Greenwood
 Gravesend-Sheepshead Bay
 Dyker Heights
 Brownsville
 Bushwick
 Fort Green
 Mapleton-Flatlands
 Bedford-Stuyvesant
 Carroll Gardens
 Coney Island
 Red Hook
(23 rows)
```

Basic PostgreSQL - exercise

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

- Table : **nyc_census_blocks**

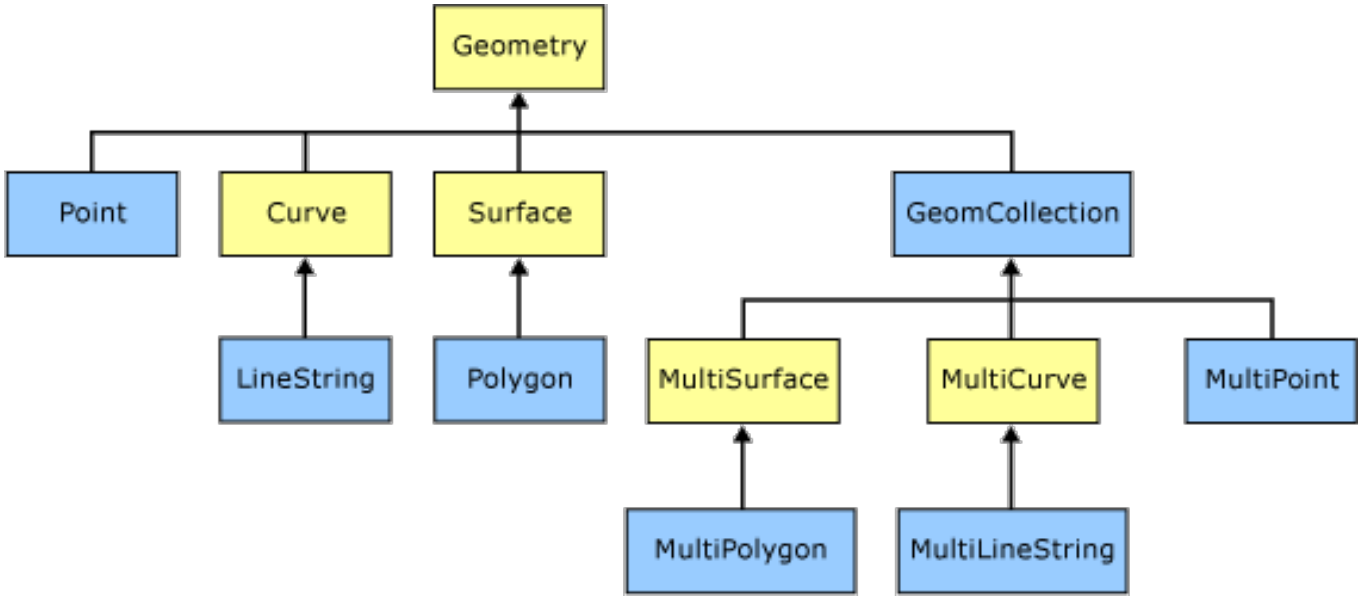
A census block is the smallest geography for which census data is reported. All higher level census geographies (block groups, tracts, metro areas, counties, etc) can be built from unions of census blocks. We have attached some demographic data to our collection of blocks.

Number of records: 36592

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

Spatial Data Types - Geometry

- **Point, LineString, Polygon, GeoCollection**



Geometry - create

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

> INSERT INTO geometries 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((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2,
2 2, 2 1, 1 1))'),
  ('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1,
0 1, 0 0)))');
```

Geometry - type

- Geometry type stores in **WKB(Well-known Binary)** format

```
postgis_lab=# SELECT * from geometries WHERE name='Point';
 name  |                    geom
-------+--------------------------------------------
 Point | 0101000000000000000000000000000000000000
(1 row)
```

- **ST_AsText(geom)** can show in **WKT(Well-known Text)** format

```
postgis_lab=# SELECT name, ST_AsText(geom) from geometries WHERE name='Point';
 name  | st_astext
-------+------------
 Point | POINT(0 0)
(1 row)

postgis_lab=# SELECT name, ST_AsText(geom) from geometries;
      name       |                            st_astext
-----------------+-------------------------------------------------------------------
 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((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 2,2 1,1 1))
 Collection      | GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0,1 0,1 1,0 1,0 0)))
(5 rows)
```

Geometry - Point

- A spatial **point** represents a single location. Points are used to represent objects when details are not important at the target scale

- Useful spatial function
  - **ST_X(geom)** returns the X ordinate
  - **ST_Y(geom)** returns the Y ordinate
  - **ST_NDims(geom)** returns the number of dimensions
  - **Exercise: print out any station from "nyc_subway_stations"**

```
     name     |                st_astext
--------------+-------------------------------------------
 Cortlandt St | POINT(583521.854408956 4507077.86259909)
(1 row)
```

Geometry - Linestrings

- A **linestring** is a path between locations. It takes the form of an ordered series of two or more points.

Simple non-closed linestring

Simple multilinestring defined by 4 endpoints of 2 elements

- Useful spatial function
  - **ST_Length(geom)** returns the length of the linestring
  - **ST_StartPoint(geom)** returns the first coordinate as a point
  - **ST_EndPoint(geom)** returns the last coordinate as a point
  - **ST_NPoints(geom)** returns the number of coordinates in the linestring

```
postgis_lab=# SELECT name, ST_Length(geom),
postgis_lab-# ST_AsText(ST_StartPoint(geom)),
postgis_lab-# ST_AsText(ST_EndPoint(geom)), ST_NPoints(geom)
postgis_lab-# FROM geometries WHERE name = 'Linestring';
    name    |     st_length     | st_astext  | st_astext  | st_npoints
------------+-------------------+------------+------------+------------
 Linestring | 3.414213562373095 | POINT(0 0) | POINT(2 2) |          4
(1 row)
```

Geometry - Polygons

- A **polygon** is a representation of an area. It is composed by rings.

- Useful spatial function
  - **ST_Area(geom)** returns the area of the polygons
  - **ST_NRings(geom)** returns the number of rings
  - **ST_ExteriorRing(geom)** returns the outer ring as a linestring
  - **ST_InteriorRingN(geom,n)** returns a specified interior ring as a linestring

```
postgis_lab=# SELECT name, ST_Area(geom), ST_NRings(geom),
ST_AsText(ST_ExteriorRing(geom))
FROM geometries WHERE name LIKE 'Polygon%';

      name       | st_area | st_nrings |              st_astext
-----------------+---------+-----------+-------------------------------------
 Polygon         |       1 |         1 | LINESTRING(0 0,1 0,1 1,0 1,0 0)
 PolygonWithHole |      99 |         2 | LINESTRING(0 0,10 0,10 10,0 10,0 0)
(2 rows)
```

  - **Exercise: Please print the interior ring of “PolygonWithHole”**

Geometry - Collections

- MultiPoint, MultiLineString, MultiPolygon, GeometryCollection

- Useful spatial function
  - ST_NumGeometries(geometry) returns the number of parts in the collection
  - ST_GeometryN(geometry,n) returns the specified part

```
postgis_lab=# SELECT name, ST_AsText(geom), ST_NumGeometries(geom)
FROM geometries
WHERE name = 'Collection';
    name    |                          st_astext                          | st_numgeometries
------------+-------------------------------------------------------------+------------------
 Collection | GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0,1 0,1 1,0 1,0 0))) |                2
(1 row)
```

Geometry - Exercise

- What is the name and length of street which contains most points ?

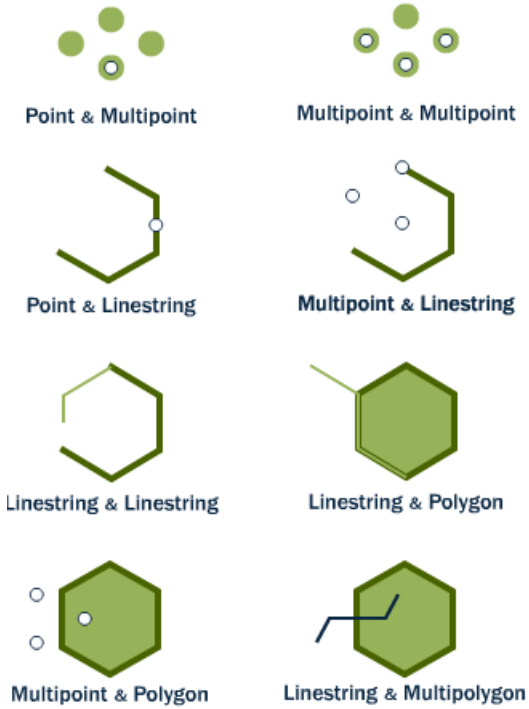
(Table: nyc_streets)

- What is the area of each borough?

(Table: nyc_census_blocks)

Spatial Relationships

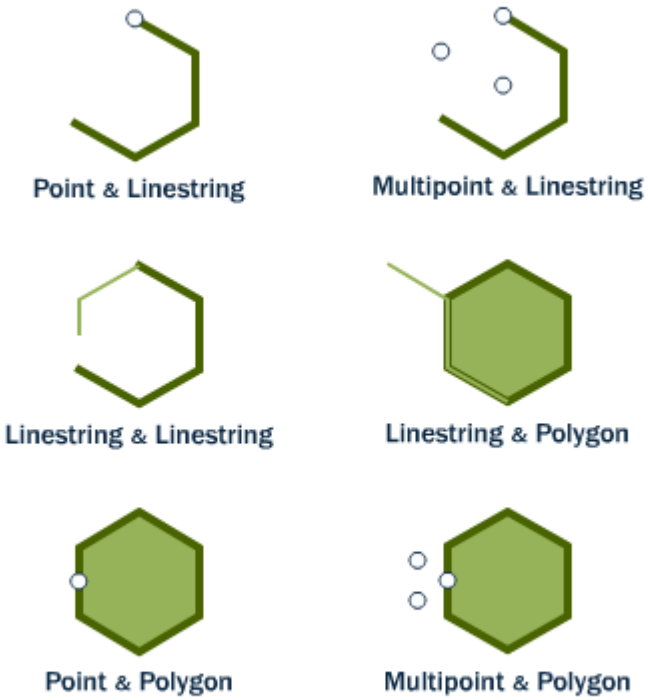
- **ST_Intersects(geometry A, geometry B):** returns TRUE if the two shapes have any space in common




```
postgis_lab=# SELECT name, boroname
postgis_lab-# FROM nyc_neighborhoods
postgis_lab-# WHERE ST_Intersects( geom,
postgis_lab(# (SELECT geom FROM nyc_subway_stations WHERE name = 'Broad St'));
        name        | boroname
--------------------+-----------
 Financial District | Manhattan
(1 row)
```

Spatial Relationships

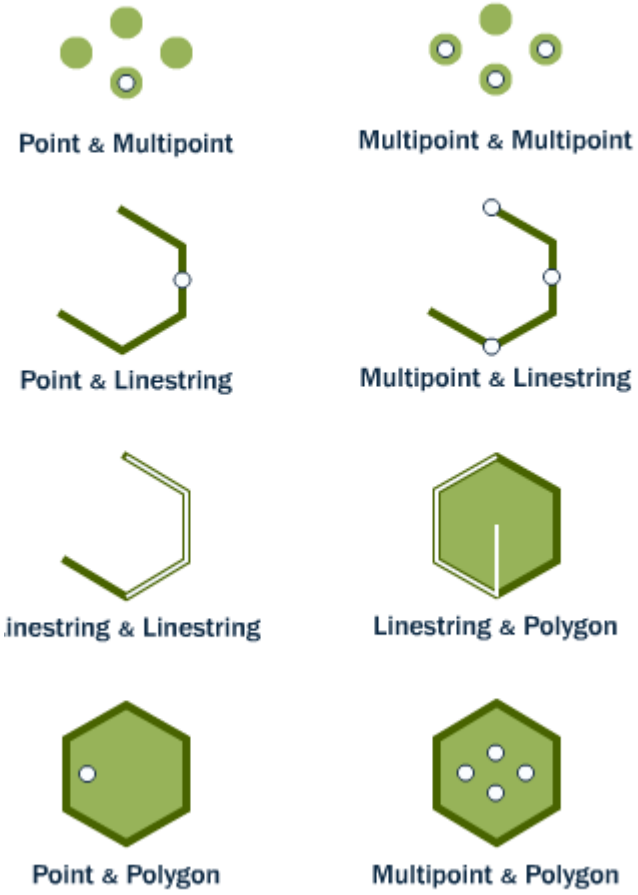
- **ST_Touches(geometry A, geometry B)**: tests whether two geometries touch at their boundaries, but do not intersect in their interiors



Spatial Relationships

- **ST_Within(geometry A, geometry B):** returns TRUE if the first geometry is completely within the second geometry

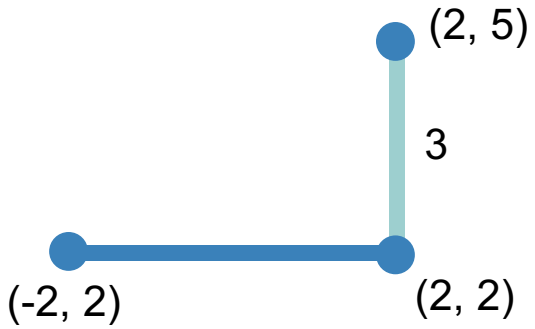
- **ST_Contains(geometry A, geometry B):** returns TRUE if the second geometry is completely contained by the first geometry.




```
postgis=# SELECT name
postgis-# FROM nyc_neighborhoods
postgis-# WHERE ST_Contains(geom, ST_GeomFromText('POINT(583571 4506714)',26918));
        name
--------------------
 Financial District
```

Spatial Relationships

- **ST_Distance(geometry A, geometry B):** calculates the shortest distance between two geometries.

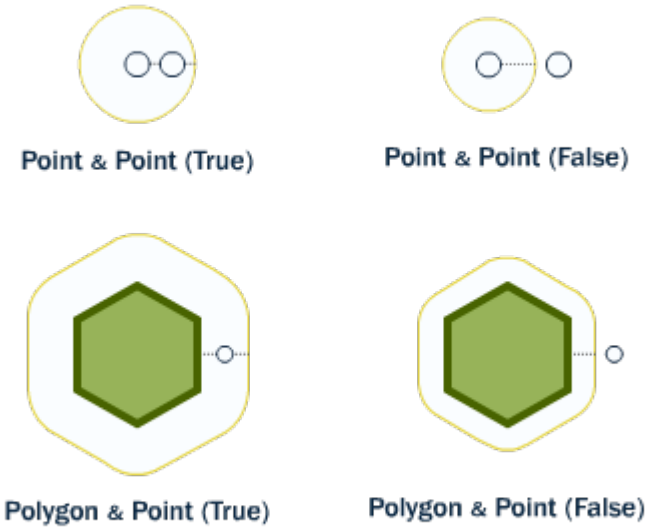



```
postgis_lab=# SELECT ST_Distance(
postgis_lab(# ST_GeometryFromText('POINT(2 5)'),
postgis_lab(# ST_GeometryFromText('LINESTRING(-2 2, 2 2)'));

 st_distance
-------------
           3
(1 row)
```

Spatial Relationships

- **ST_DWithin(geometry A, geometry B, radius)**: Returns true if the geometries are within the specified distance (radius) of one another.




```
postgis_lab=# SELECT name FROM nyc_streets
postgis_lab-# WHERE ST_DWithin(geom,
postgis_lab(# (SELECT geom FROM nyc_streets WHERE name = 'Atlantic Commons'),
postgis_lab(# 10);

       name
------------------
 S Oxford St
 Atlantic Commons
 Cumberland St
(3 rows)
```

Spatial Relationships - Exercise

- Street ‘Atlantic Commons’
- What neighborhood is Atlantic Commons in? (Table:nyc_neighborhoods)
- Approximately how many people live on (within 50 meters of) Atlantic Commons? (Table: nyc_census_blocks)

Spatial Relationships – Problem

- Find the neighborhood which contains Broad Station.

- ```
  SELECT name, boroname
  FROM nyc_neighborhoods
  WHERE ST_Contains(geom,
        (SELECT geom FROM nyc_subway_stations
        WHERE name = 'Broad St'));
  ```

- How about that if we want to select data from both tables?
  ```

Spatial Join

- Spatial Join allow you to combine information from different tables by using spatial relationships as the join key.
- Retrive the station, neighborhood which contains Broad Station.

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

Spatial Join

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

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

Spatial Join - Exercise

- Each subway station may have many routes passing.

e.g. J-train, M-train, Z-train pass though Broad Station.

- Find out the racial make-up of within 200 meters of the A-train line.

(Table: nyc_subway_stations, nyc_census_blocks)

Reference

1. http://www.postgresqltutorial.com/