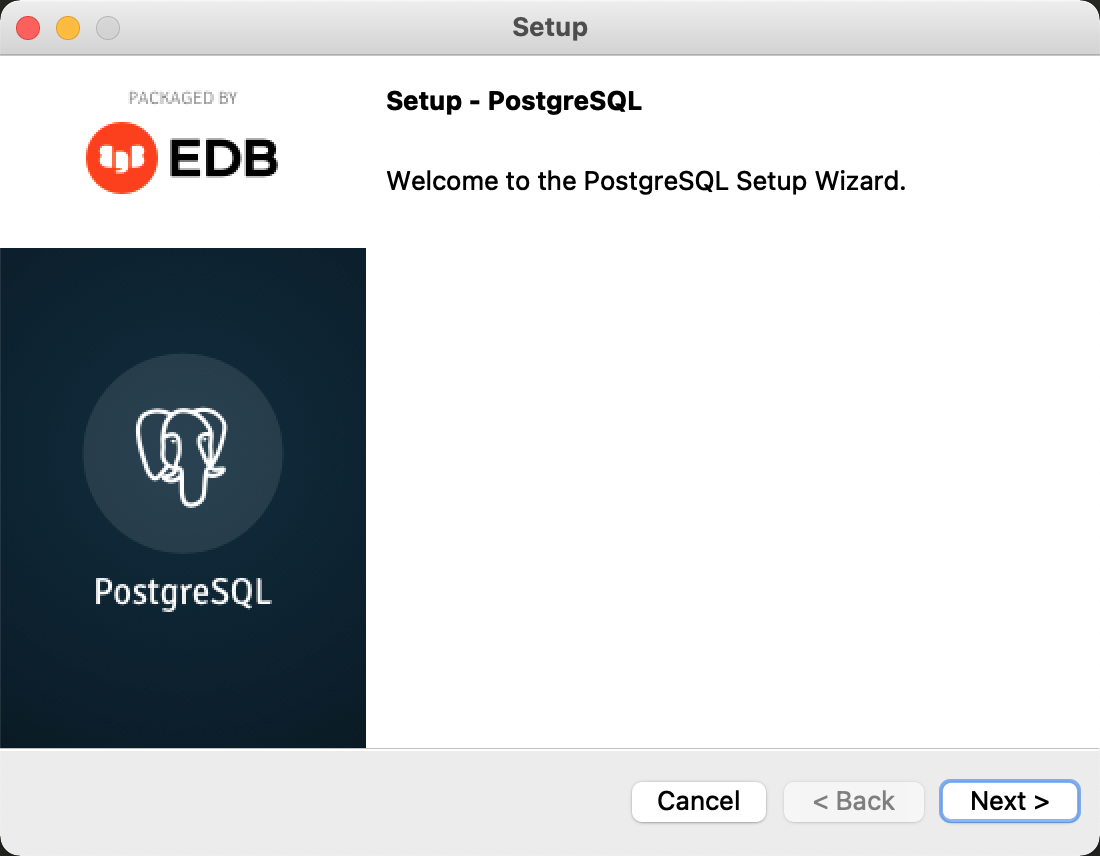
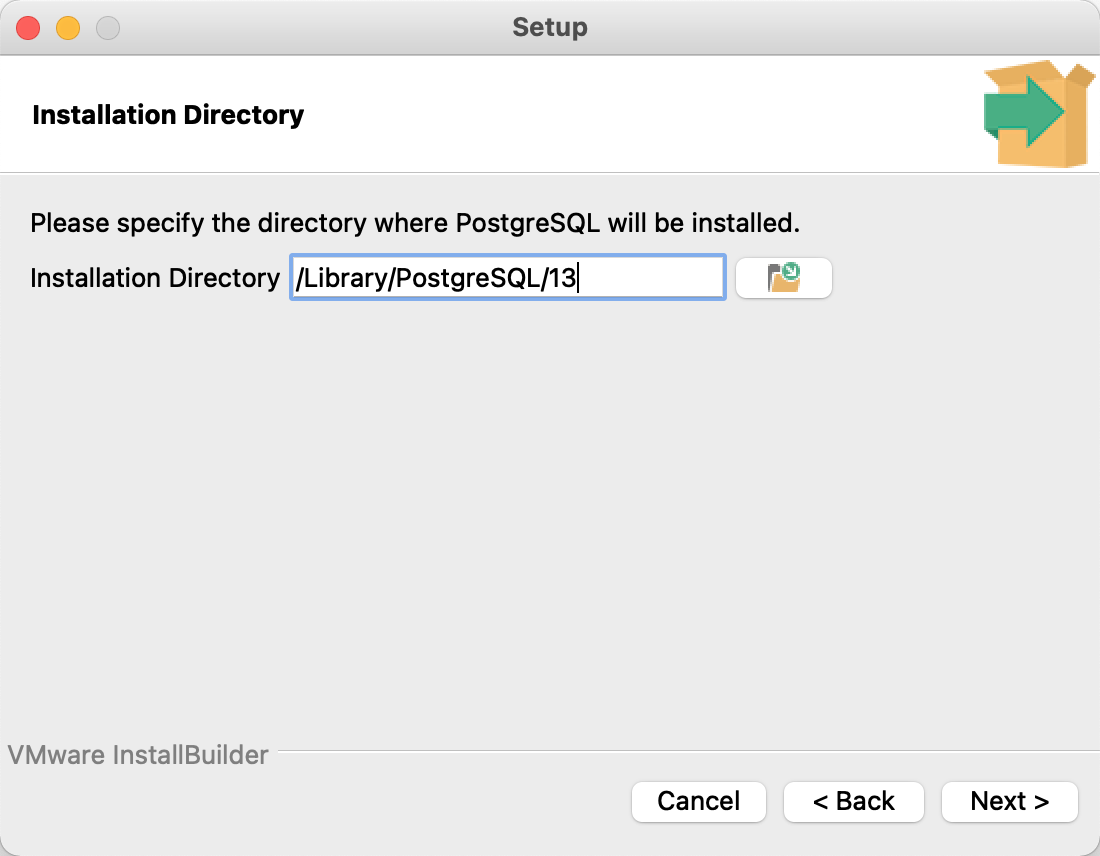
**Spatial Database operations**

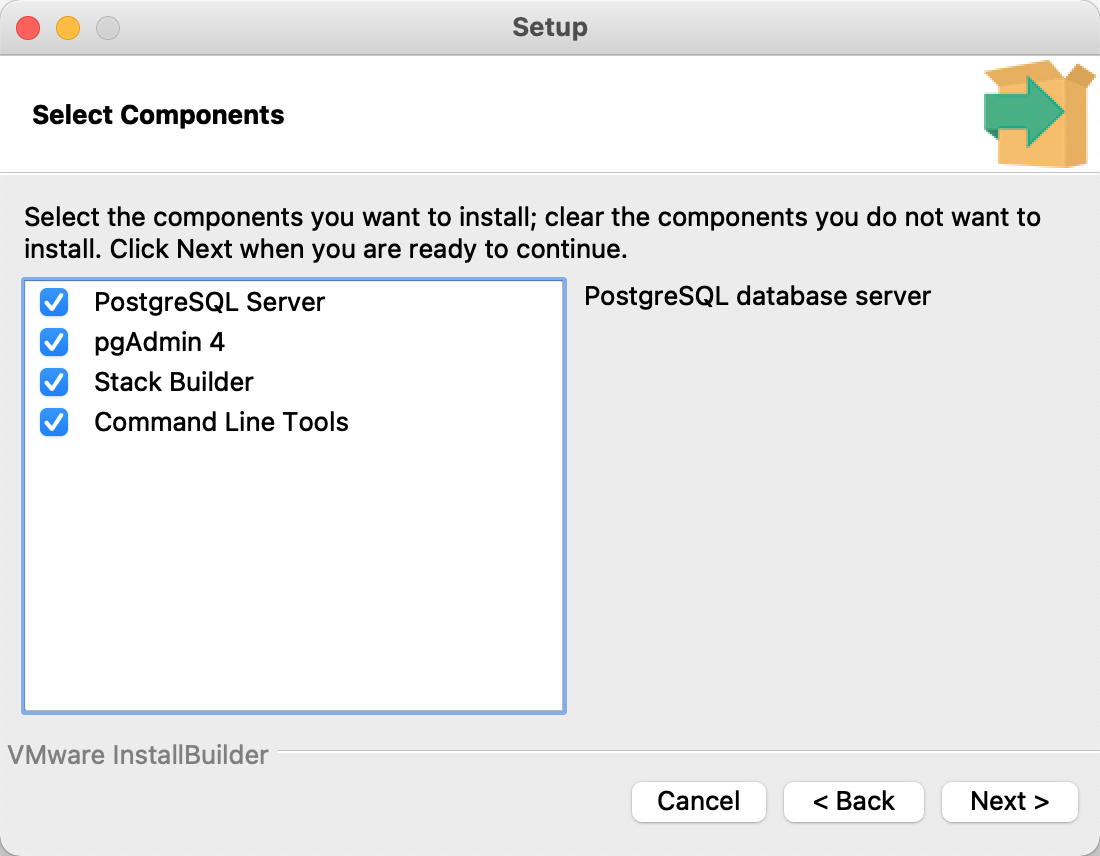
Database system make the spatial operation much simpler, and it is more efficient to handle large number of records. For example, the Excel has limit of the number of records, however, there is no such limit in database system. In addition, you probably cannot handle really large shapefile with millions of records. Using the spatial database system in PostGIS, you can make the spatial operations much more efficient. There are many available database systems and spatial database system. This class, we are going to focus on one of the most popular open-source database system Postgres/PostGIS. We are going to spend two weeks on the database system. The first class is about configure the Postgres/PostGIS system. The second class is about using Python to query and manipulate the database system. You will learn how to write SQL to do query in the database.

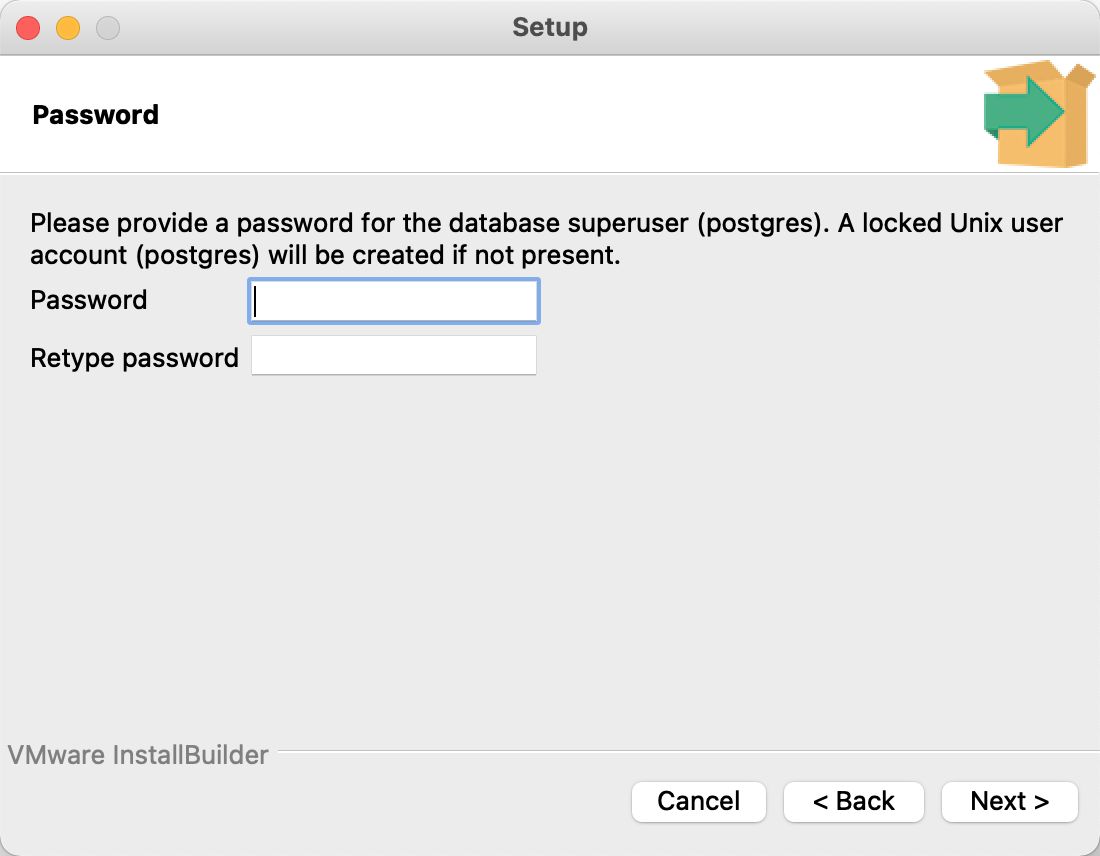
**1. Configure the Postgres/PostGIS**

Download the Postgres/PostGIS, <https://www.postgresql.org/download/>

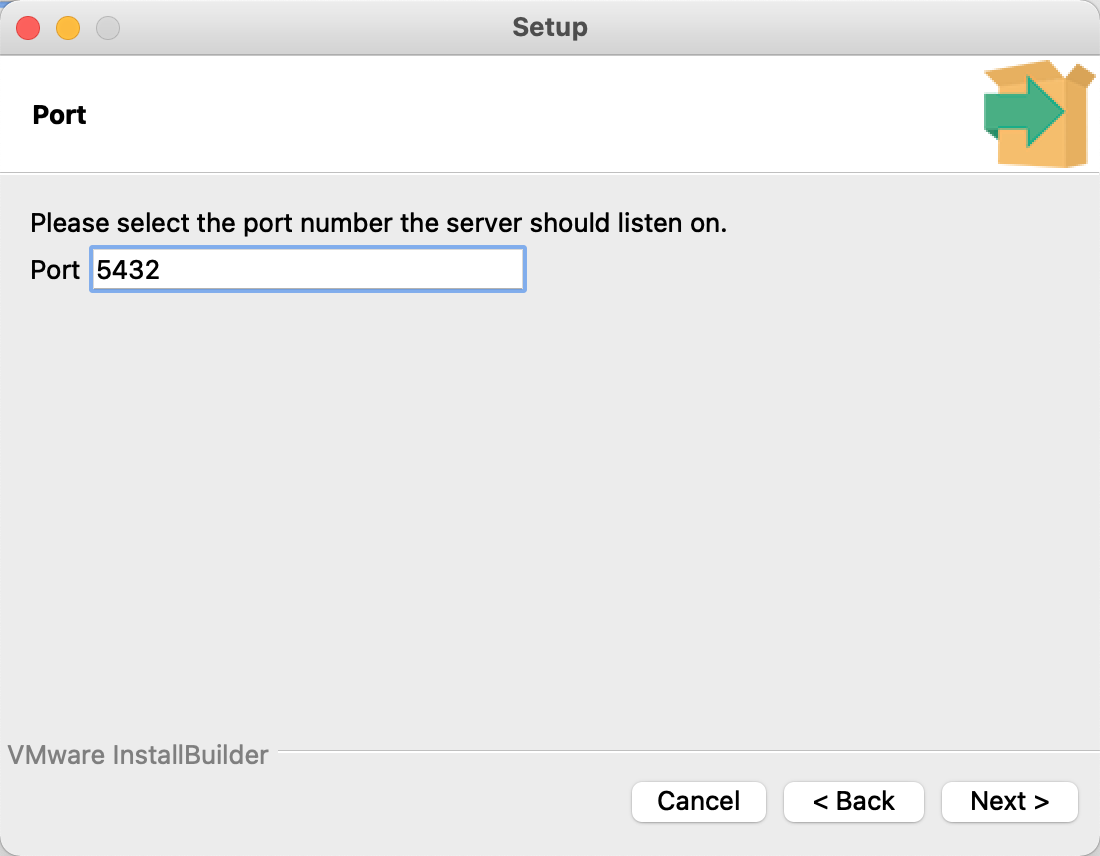


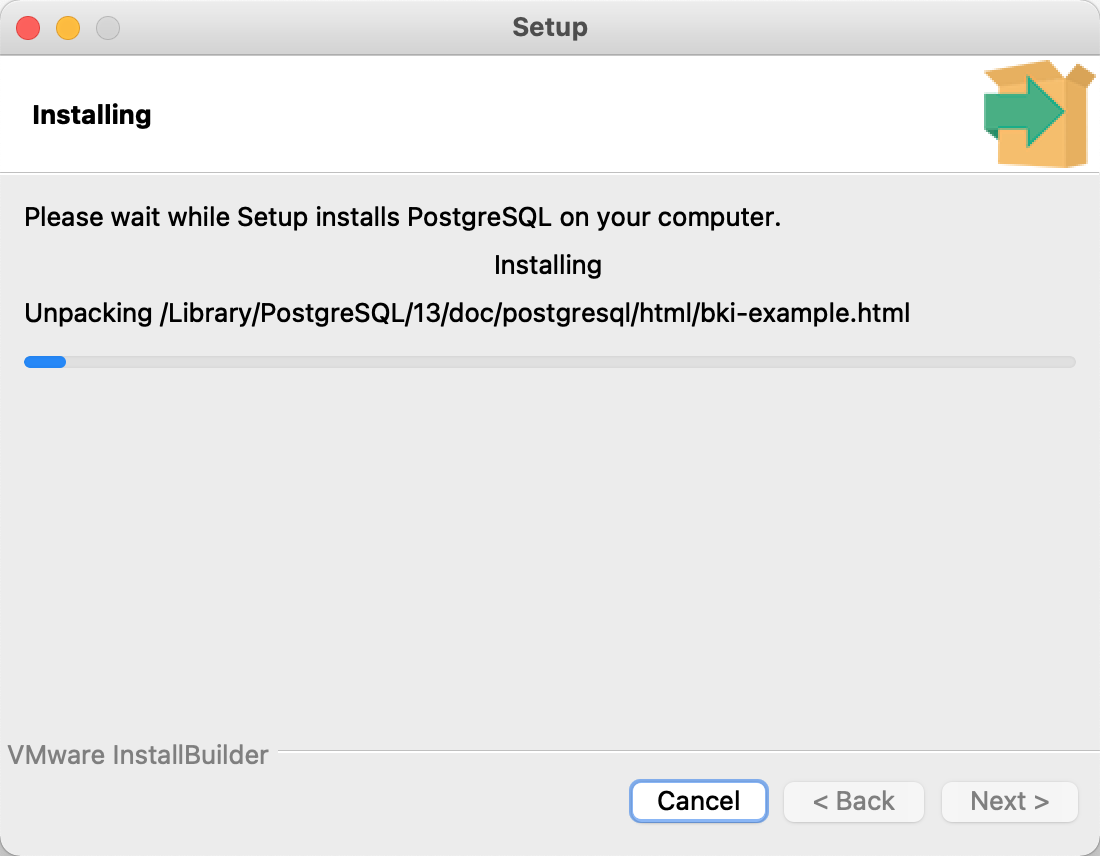


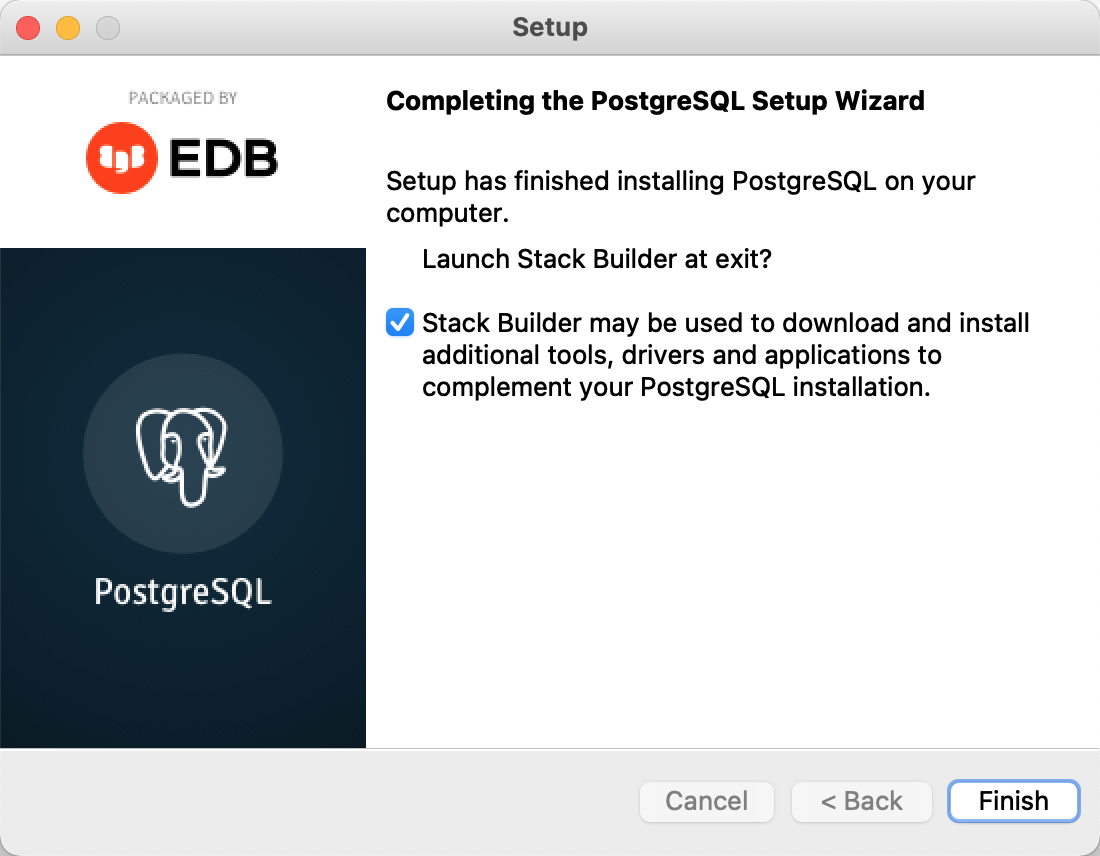


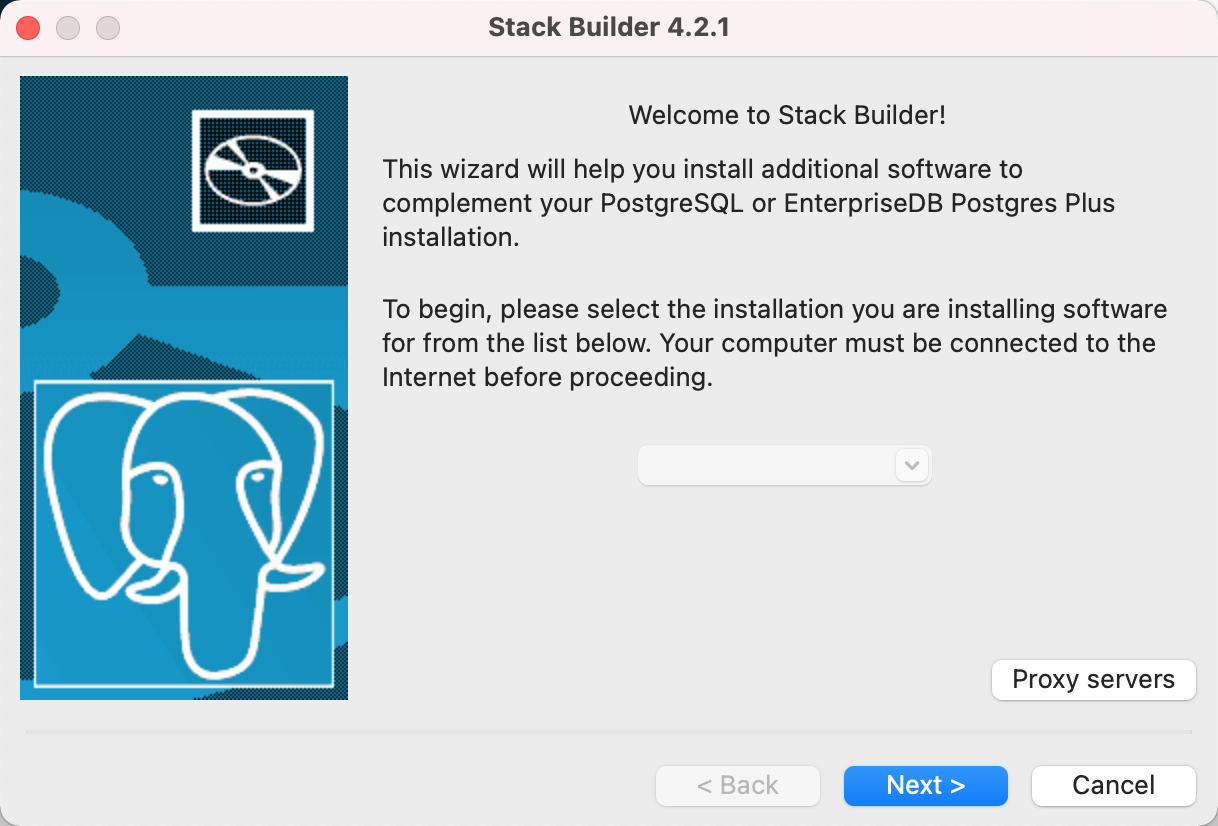


Password: 1234

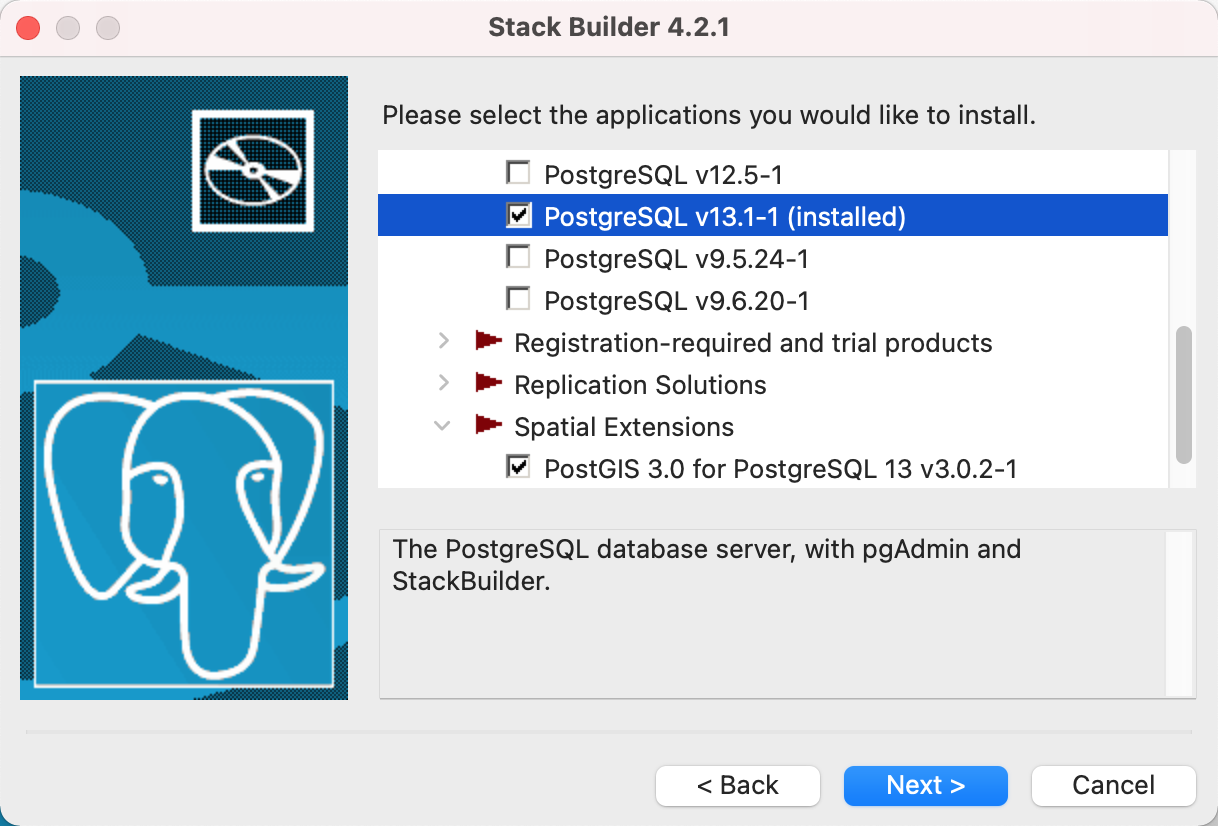






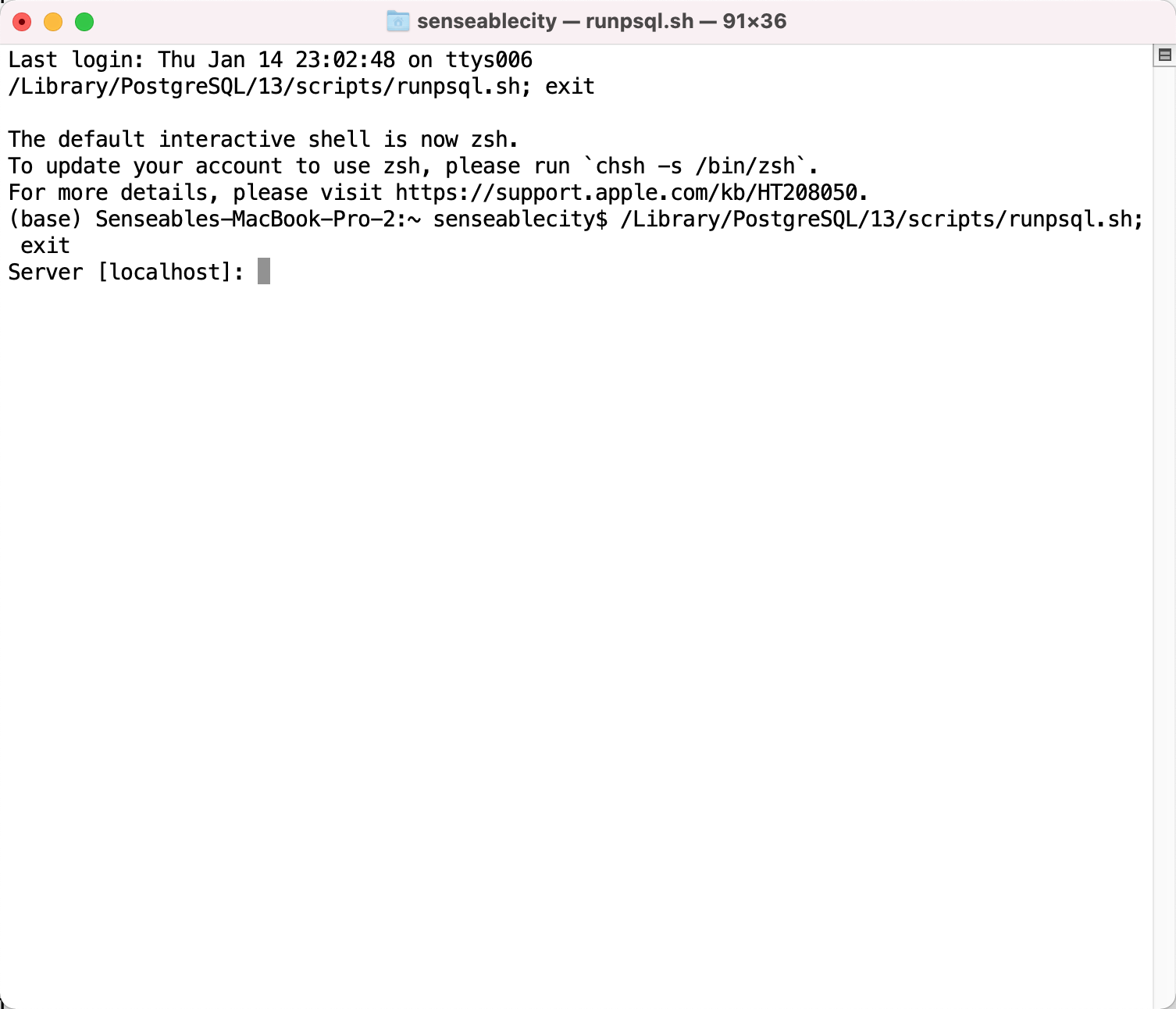




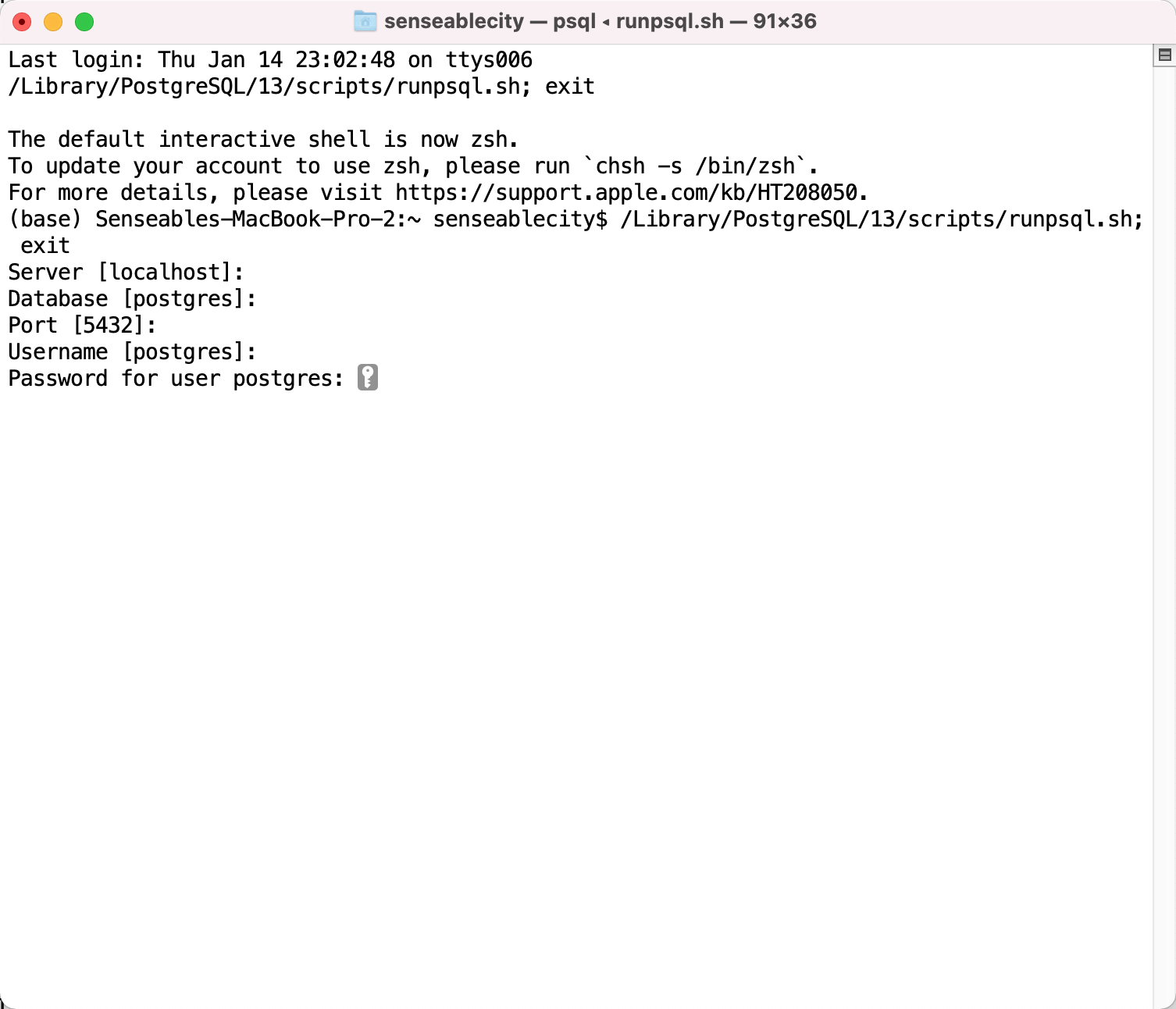


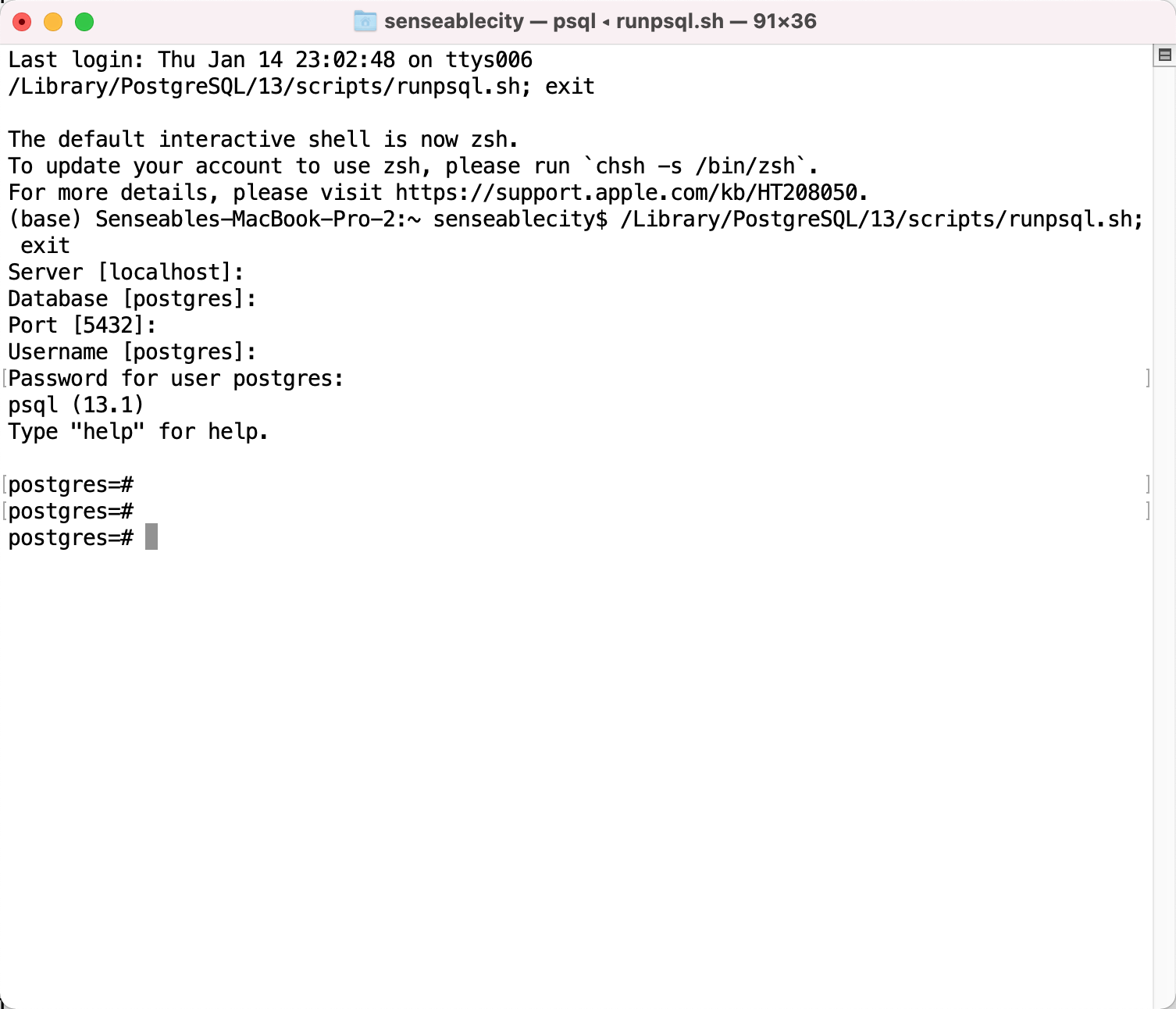
**2. Be familiar with the database**

Open PSQL terminal,



Typein your password,





2.3.1 Create tables and db opearations

Commands in psql terminal,

List all databases,

\l

Connect to a certain database, \c databasename

\c dbname

List all tables,

\c dbname

\dt

Create a PostGIS table for the Runkeeper metadata,

CREATE EXTENSION postgis;

CREATE TABLE IF NOT EXISTS boston

(tripid VARCHAR(40) PRIMARY KEY NOT NULL,

userid VARCHAR(40),

rtype VARCHAR(15),

age VARCHAR(5),

gender VARCHAR(1),

distance REAL,

startT VARCHAR(20),

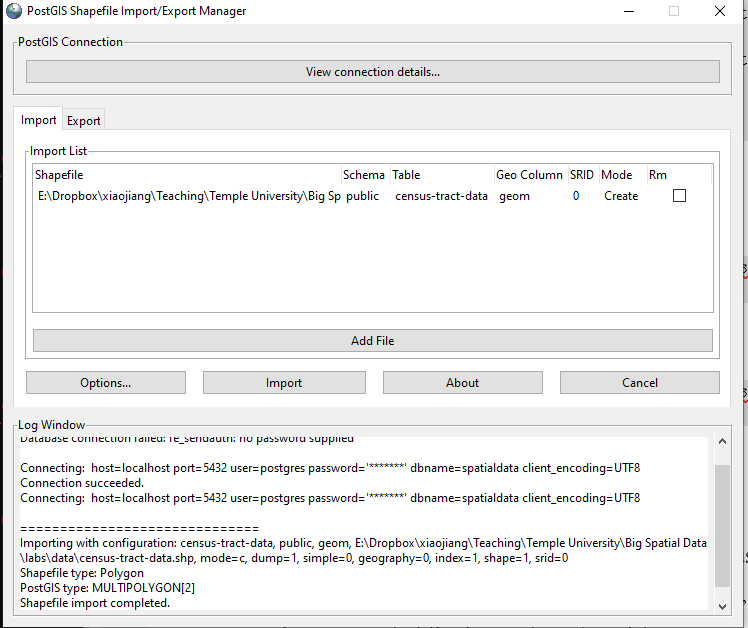
endT VARCHAR(20),

geom geometry(LineString, 3857));

**3. Convert the shapefile into database,**

Download the tool for convert the shapefile to spatial data base table,

<http://download.osgeo.org/postgis/windows/>



**4. Write SQL to query the table**

Enable PostGIS.

Create extension of PostGIS

CREATE EXTENSION IF NOT EXISTS PostGIS CASCADE;

Then you can check the version of the installed PostGIS,

SELECT PostGIS\_version();

Create a database,

create DATABASE runners;

CREATE EXTENSION IF NOT EXISTS PostGIS CASCADE;

SELECT PostGIS\_version();

CREATE TABLE IF NOT EXISTS TEST

(id VARCHAR(30) PRIMARY KEY NOT NULL,

year VARCHAR(10),

month VARCHAR(2),

city VARCHAR(20));

INSERT INTO TEST (id, year, month, city) VALUES('1', '2015', '07', 'Philadelphia');

INSERT INTO TEST (id, year, month, city) VALUES('2', '2016', '07', 'Boston');

INSERT INTO TEST (id, year, month, city) VALUES('3', '2017', '07', 'New York City');

INSERT INTO TEST (id, year, month, city) VALUES('4', '2018', '07', 'Houston');

SELECT \* FROM TEST;

Add the geom field to the table,

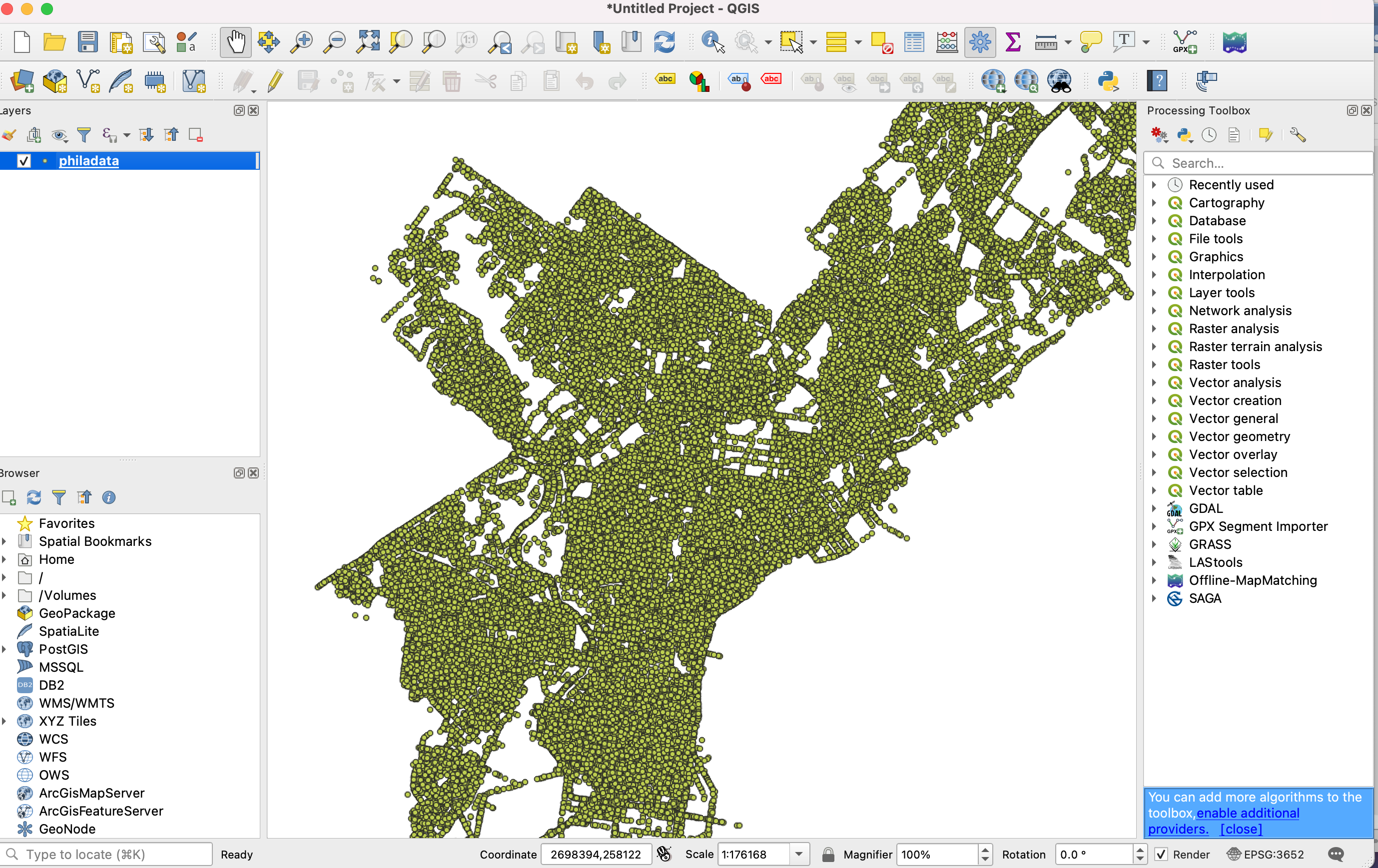
ALTER TABLE gvi\_phily\_yale ADD COLUMN geom geometry(Point, 4326);

UPDATE gvi\_phily\_yale SET geom = ST\_SetSRID(ST\_MakePoint(lon, lat), 4326);

Save the PostGIS database into a shapefile,

pgsql2shp -u postgres -h localhost -P 5424796 -f philadata geospatial "SELECT \* FROM gvi\_phily\_yale;"

Here is your results look like,



You can save the shapefile as a spatial data base using the above tool,

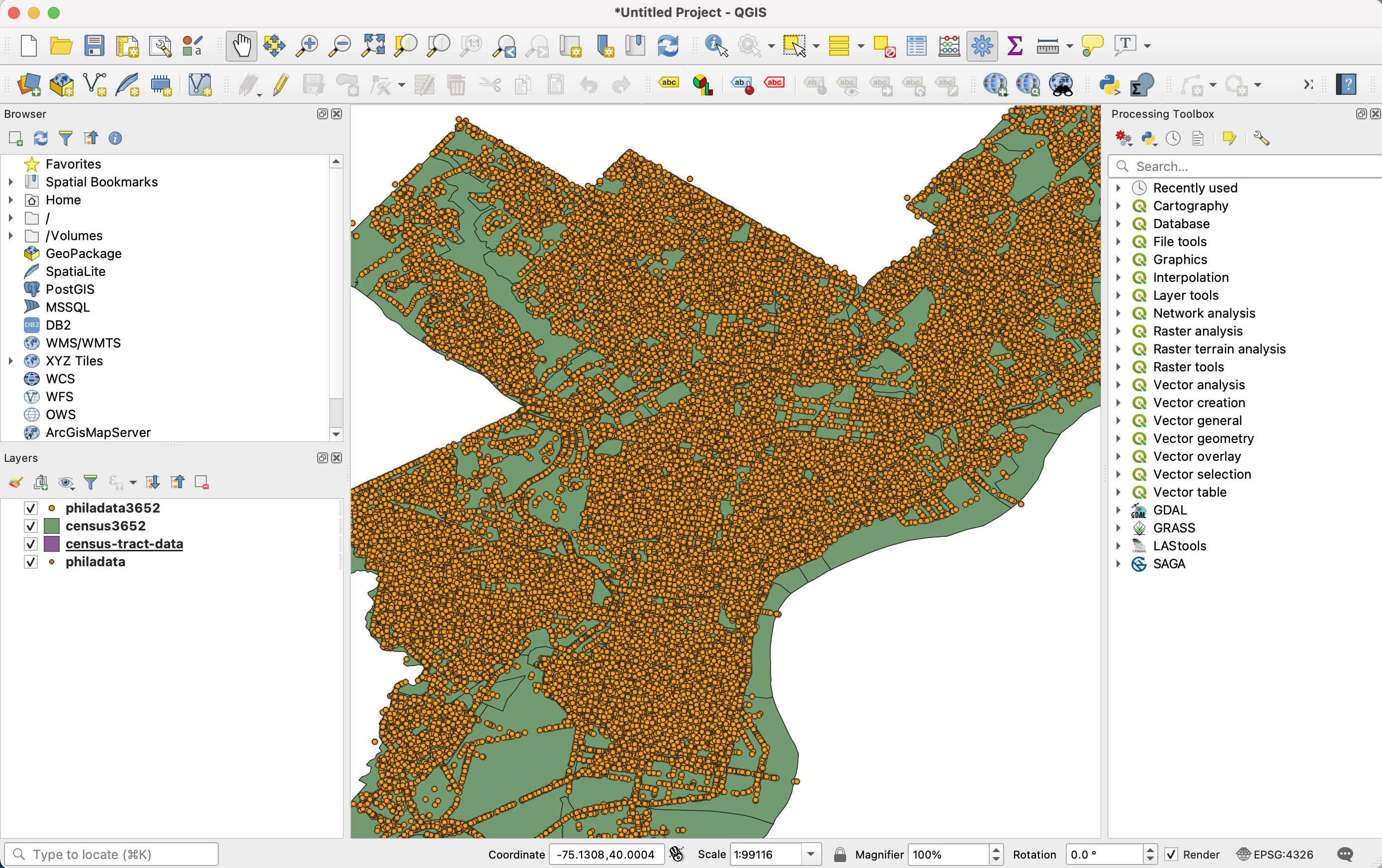
It is usually way more efficient and flexible to use database to manage the trajectory data. Therefore, different from the previous version code, here we can use the database to manage the huge amount of Runkeeper trajectory data.

**5. A spatial query using PostGIS**

This part, we are going to the database to do spatial interact operation. Let’s assume, we have Count the number of points in each census tract using PostGIS.

5.1 Prepare the database

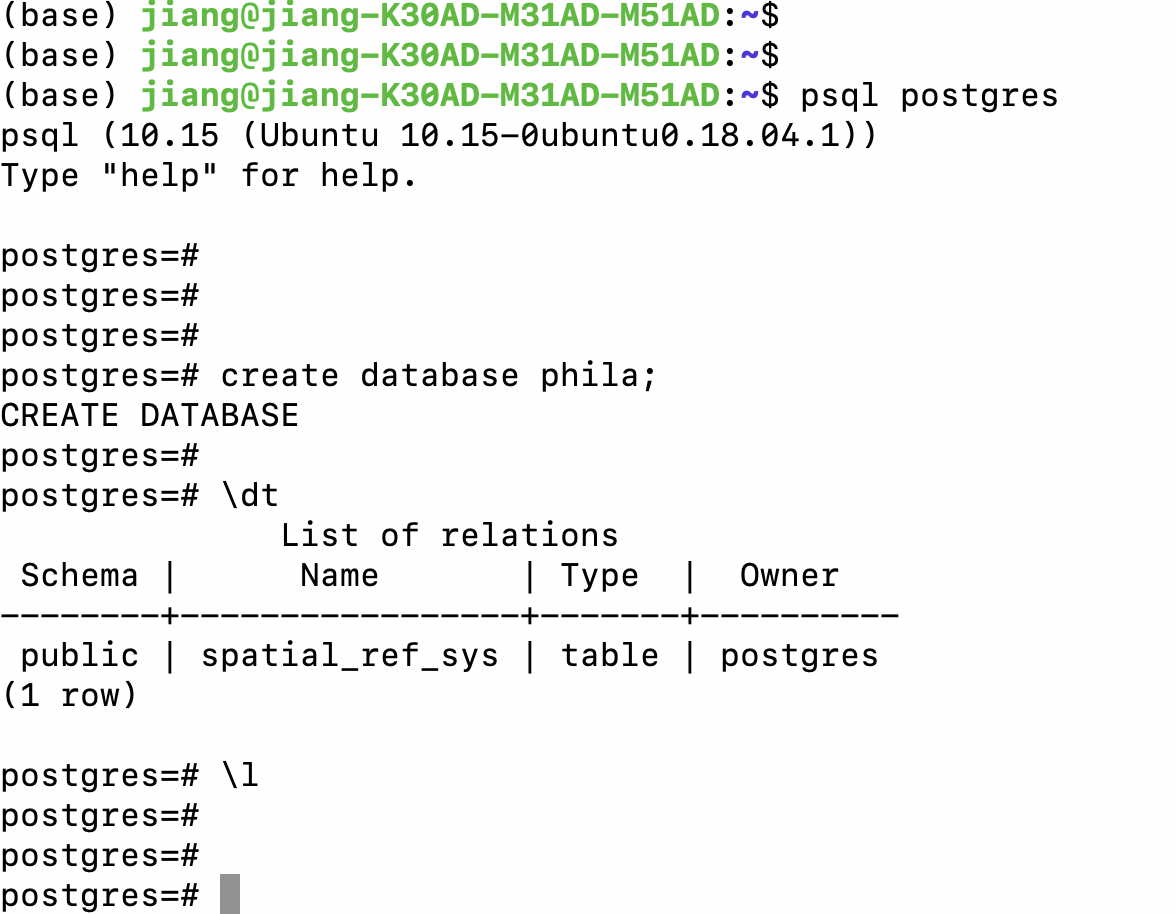
Prepare the shapefiles of the point of GSV points and the census tract polygons shapefile for Philadelphia. Make sure the shapefiles are in the same projection. Let’s use the local projection for Pennsylvania South feet first (EPSG: 3652). You can use QGIS or ArcGIS to do the projection transform easily.



Let then convert these two shapefiles into two tables in a database of PostGIS. You can use command line to do the conversion or the software to do the conversion.

Create a database to store the tables

create database phila;



Convert the shapefile of GSV points in Philadelphia into a PostGIS table.

\c phila;

shp2pgsql -s 3652 philadata> philadata.sql

This is using PostGIS database, therefore, we need to add one more statement to enable the PostGIS. Open the “philadata.sql”, and add the statement on top,

CREATE EXTENSION postgis;

Now your “philadata.sql” statement looks like,

Run the .sql and convert the shapefile int Postgres database table,

psql -h localhost -U postgres -d phila -p 5432 -f philadata.sql

-h: the host of localhost

-U: the username

-d: the database name

-p: portal, default of 5432

-f: the .sql file

Convert the shapefile of the census tract of Philadelphia into the table

shp2pgsql -s 3652 census3652> census3652.sql

Then you will find there is a census3652.sql file. The next step is to run the commands and insert the data to your database, make sure you start the PostGIS service first in your psql terminal,

CREATE EXTENSION postgis;

Then you can run the .sql in your Mac/Linux terminal,

psql -h localhost -U postgres -d phila -p 5432 -f census3652.sql

5.2 Spatial interaction

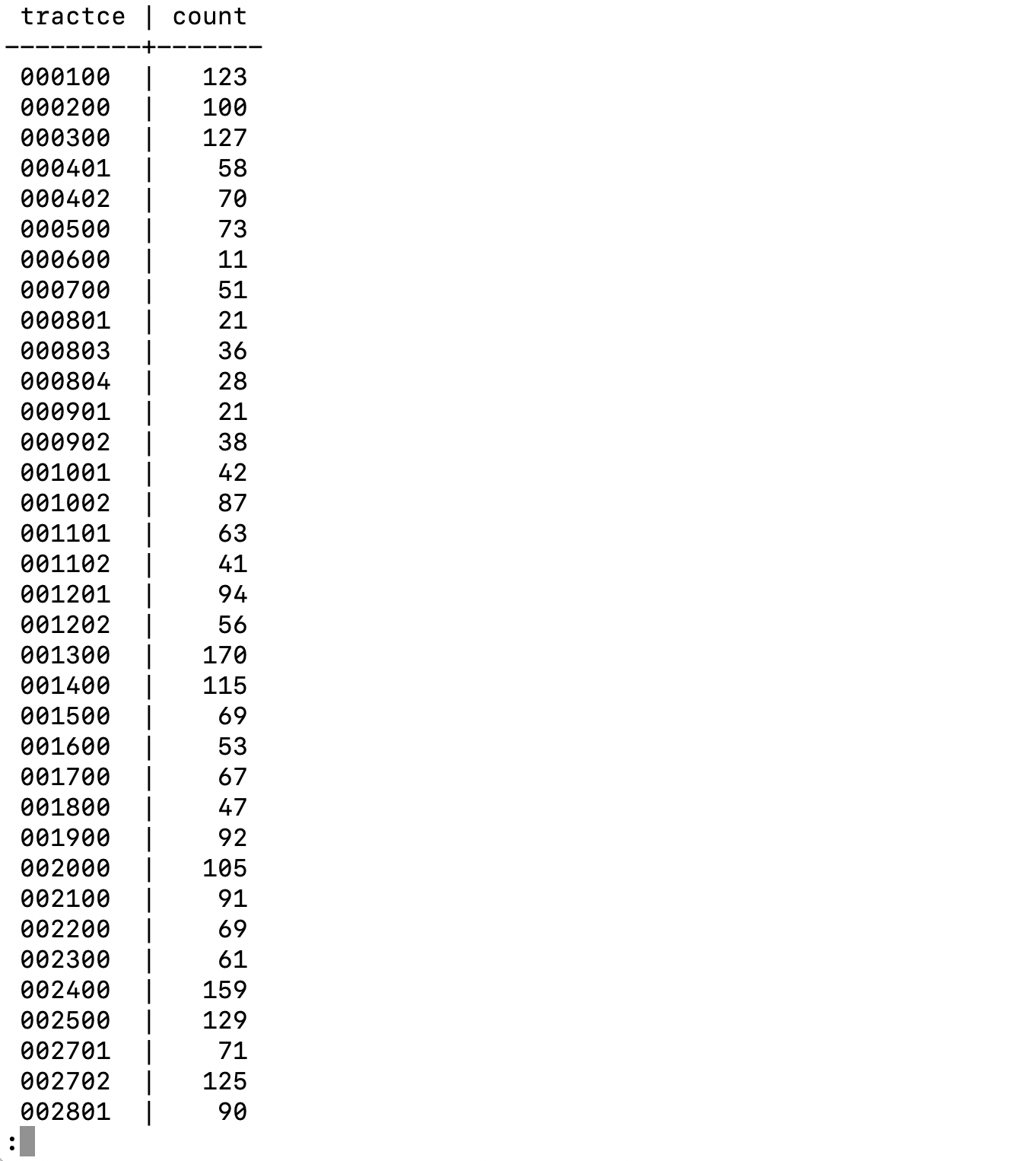
Now we are ready for the spatial database tables. Then we can use the database tables to do the spatial join of the point and the census tract polygon file.

SELECT census3652.tractce, count(philadata3652.panoid)

FROM philadata3652

LEFT JOIN census3652 ON ST\_Intersects(census3652.geom, philadata3652.geom)

GROUP BY census3652.tractce;



Let’s save the result of the number of point in each polygon as a shapefile, we need to first query the census tract information and the corresponding number of GSV sites in each census tract,

SELECT c.tractce, c.countyfp, c.statefp, c.geom, t.num FROM (SELECT census3652.tractce, count(philadata3652.panoid) as num

FROM philadata3652

LEFT JOIN census3652 ON ST\_Intersects(census3652.geom, philadata3652.geom)

GROUP BY census3652.tractce) t JOIN census3652 c ON c. tractce = t.tractce;

Since we want to save the result as a new shapefile, we need to create a new table of the queried result,

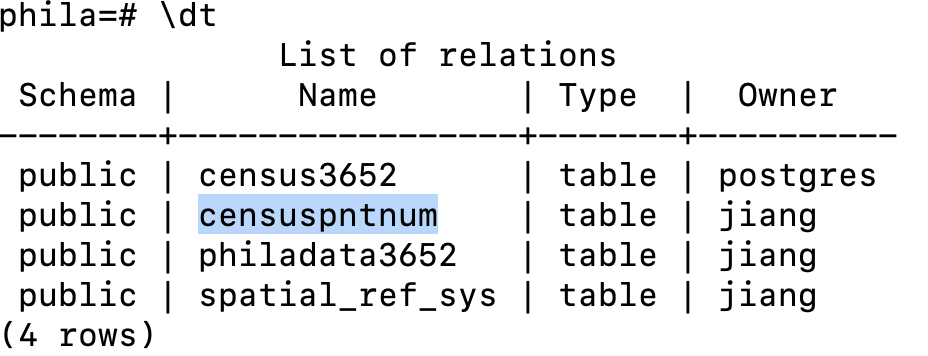
CREATE TABLE CensusPntNum AS SELECT c.tractce, c.countyfp, c.statefp, c.geom, t.num FROM (SELECT census3652.tractce, count(philadata3652.panoid) as num

FROM philadata3652

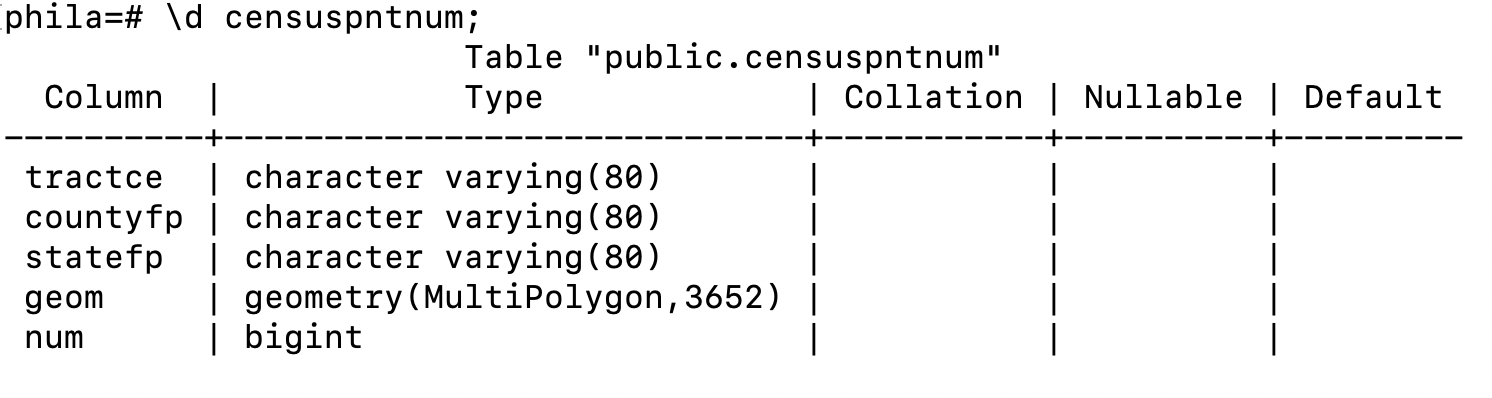
LEFT JOIN census3652 ON ST\_Intersects(census3652.geom, philadata3652.geom)

GROUP BY census3652.tractce) t JOIN census3652 c ON c. tractce = t.tractce;

Then you will find you have a new table “censuspntnum” created,



You can check the structure of the newly created table by,



Let’s save the table as a shapefile, using the following command in terminal (not your psql command),

pgsql2shp -u postgres -h localhost -P 5424796 -f pntshp phila "SELECT \* FROM censuspntnum;"

-u: the username of the database

-h: the host, here is the localhost

-P: is your password

-f: the file you going to save

phila: the database name

**6. Access the database table using Python**

6.1 Query from the database

You can use Python to access the database and manipulate the table,

You need to install psycopg2 module first by using,

pip install psycopg2

Then you can open your Jupyter Notebook and access the database table,

import psycopg2

conn = psycopg2.connect(host="localhost",database="phila", user="postgres", password="5424796")

cur = conn.cursor()

cur.execute("SELECT version();")

create\_table\_query = '''SELECT tractce, countyfp,statefp,num FROM censuspntnum;'''

cur.execute(create\_table\_query)

rows = cur.fetchall()

for row in rows:

print(" -------- ", row)

conn.commit()

6.2 Do more complicated queries through Python

Let’s replicate our previous queries through Python