

Spatial data loading and Visualization using PostgreSQL

June 29, 2021

9:30 am IST | 12:00 pm SGT

Mansur Shaikh | Sr. SE | EDB

June 29, 2021



Doing More with Postgres...



Open source alternative to commercial RDBMS

- Reduce cost
- Leverage in-house talent
- Flexible license model

RDBMS platform for new developments

- Proven RDBMS
- SQL compliant
- Extremely stable

Innovative DBMS Platform

- Not only SQL (SQL + JSON/KVP)
- Foreign Data Wrappers
- PostGIS

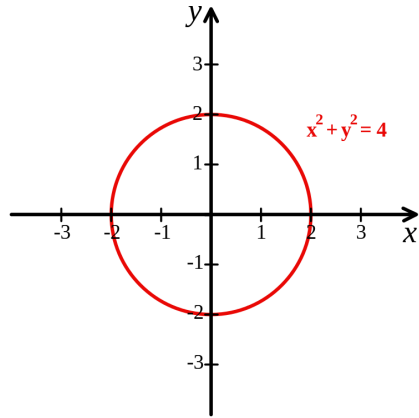
Spatial Database Capabilities in PostgreSQL

Why use spatial databases (Cont)

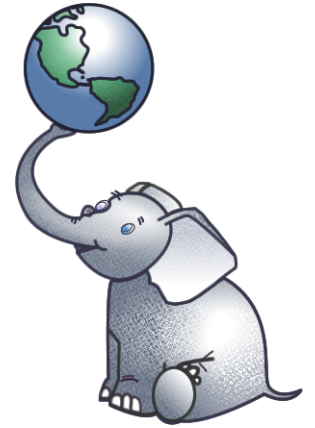
- Accessible from multiple clients
 - Desktop GIS
 - Non-GIS software
 - Custom Application
 - Internet
- SQL Queries
- Scalability
 - Replication
- Common data store
 - Integrate non-spatial data



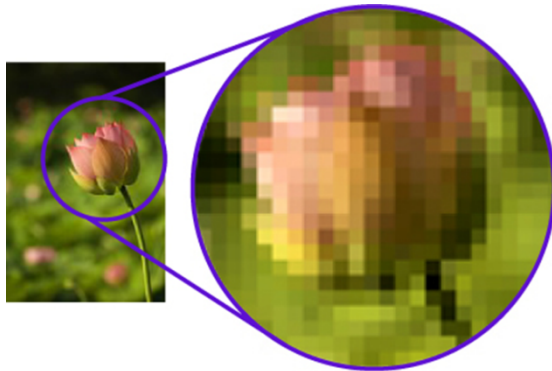
Geometry



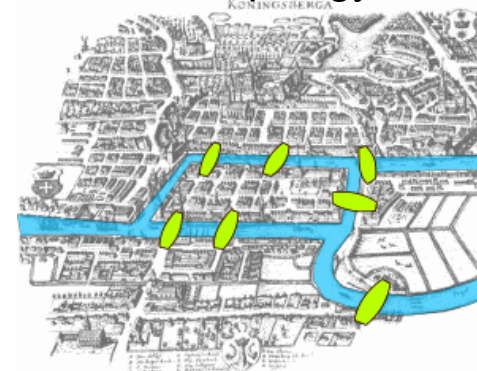
Geography



Raster



Topology

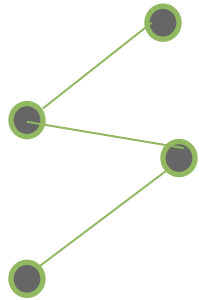


PostGIS Data subtypes

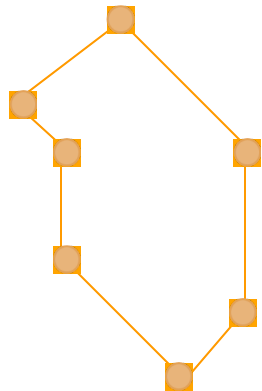
Point



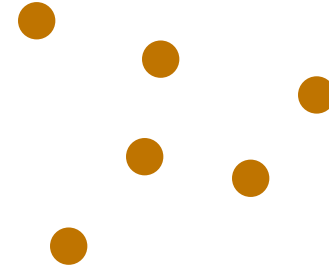
Linestring



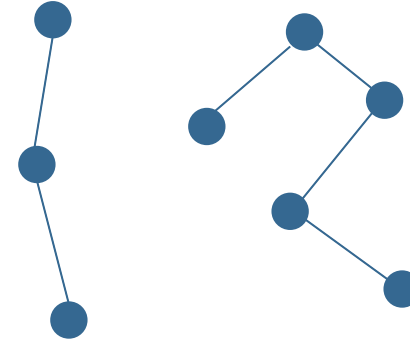
Polygon



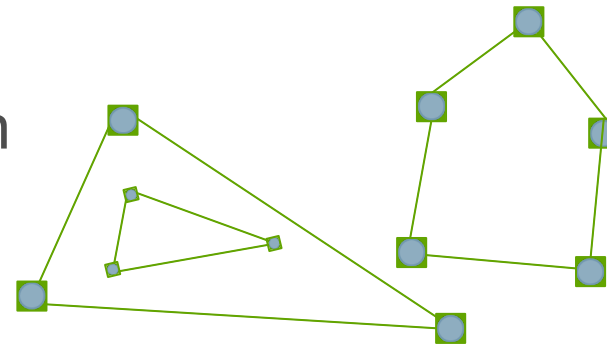
Multipoint



Multilinestring



Multipolygon



PostgreSQL , PostGIS and QGIS installation



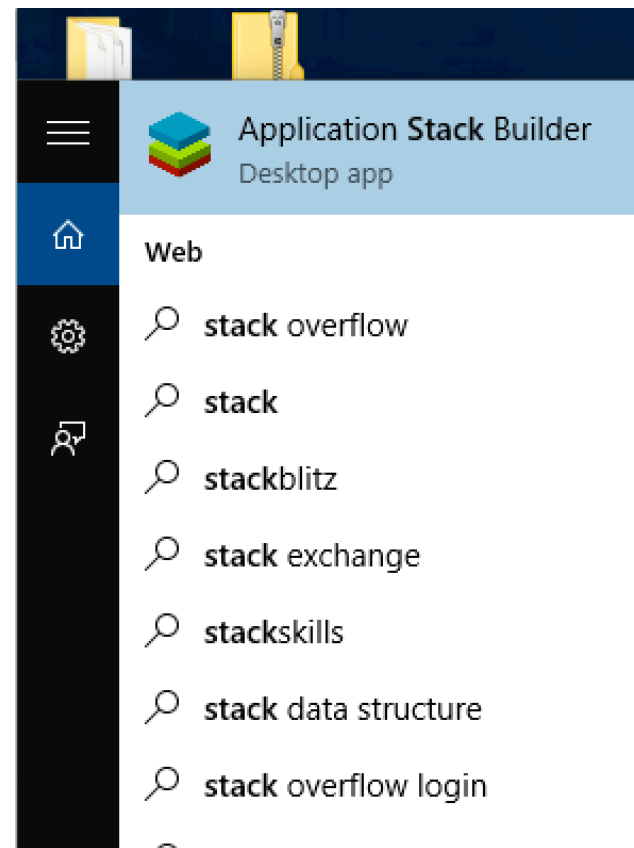
Installing PostgreSQL on your Local Computer

- Download Postgresql
 - <https://www.enterprisedb.com/downloads/postgresql>
- Interactive Installer
 - postgresql-11.12-2-windows-x64.exe



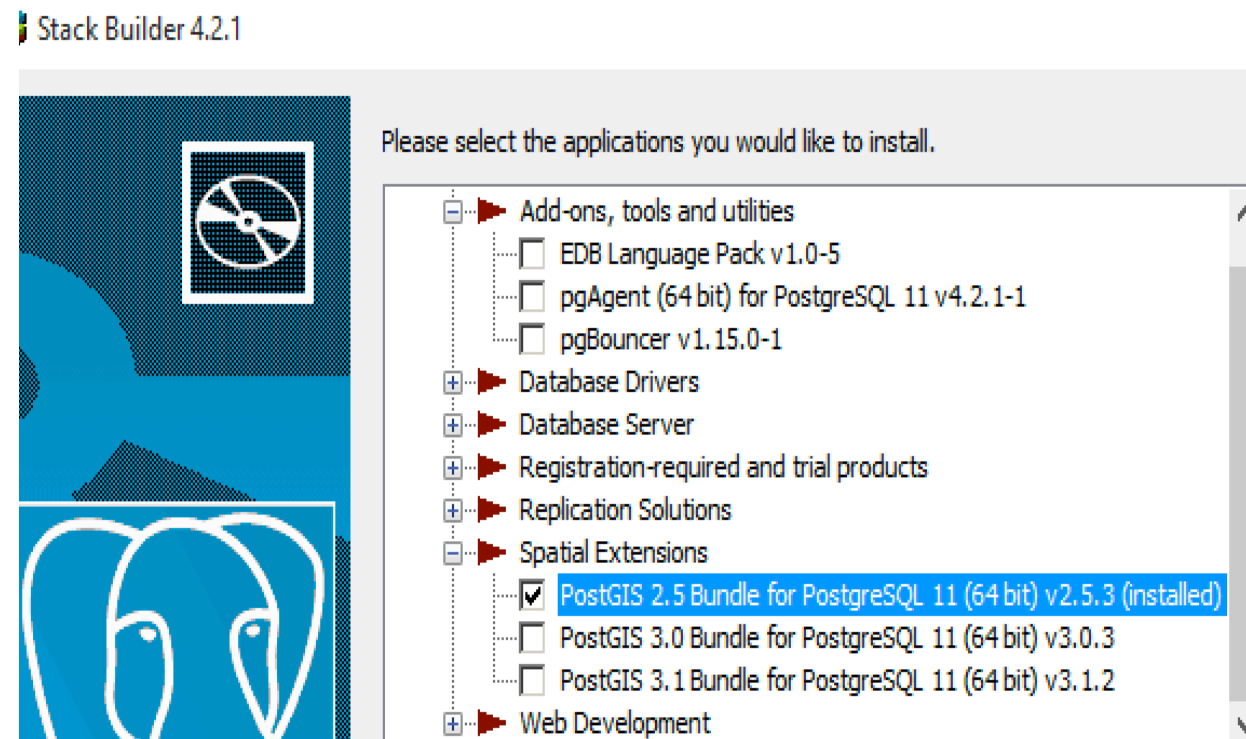
PostGIS Installation

- <https://www.enterprisedb.com/downloads/postgis>
- <https://www.enterprisedb.com/downloads/postgis>



PostGIS Installation

- <https://www.enterprisedb.com/downloads/postgis>
- <https://www.enterprisedb.com/downloads/postgis>



PgAdmin4

Open PgAdmin4

Select on server and create server

Right click on database and create database -> sdb_course

Select sdb_course-> verify extensions and schemas and functions

Add PostGIS extension

Select sdb_course

Open query tool-> create extension postgis;

Verify functions, extensions and public schema

Verify

spatial_ref_sys

Installing QGIS

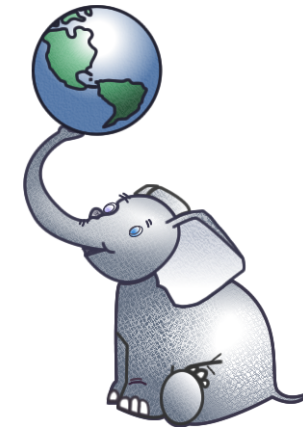
- Download link

<https://qgis.org/en/site/forusers/download.html>

- Windows Installer
QGIS-3.10.2 "a coruna setup"

Loading Spatial data into PostGIS

- Resource file :- sdb_data.zip
- Loading shapefile
 - Using PostGIS shapefile and dbf exporter
- Command line :- c:\programefiles\postgresql\10\bin
 - raster2pgsql , shp2pgsql
- Open Postgres shapefile import/export manager
- View connection detail-> import->add file
- Goto S DB_DATA folder -> load baea-nests.shp and bowl_habitat.shp
- Give special reference id is 4326



Loading Spatial data into PostGIS using QGIS

- Load into QGIS and then put into PostGIS
 - Any type of vector data you can load and view – ESRI Geo, or Jeojson, GPS file
 - Select SDB_DATA folder from browser panel and drag and paste layers panel
 - Select three file GBH_Rookeries.shp, Linear_Projects.shp, Raptor_Nests-shp
 - **First make connection**
 - Select database-> Dbmanager ->
 - Make a connection with postgis from Browser panel -> provide connection information
 - Click on basic for authentication and click on store -> Test connection
 - List tables with no geometry
 - Click on ok
 - Click on import
 - Give input file GBH_Rookeries, Linear_projects, Raptor_nests one by one
 - Go to PGAdmin4 and verify sdb_course with these five tables

Loading non spatial data

- Get data from excel sheet to QGIS and QGIS to PostGIS
 - Data source :- SDB_survey file :- baea_survey, buowl_survey, raptor_survey
 - Open QGIS -> Open data source manager -> click on vector
 - Browse to file location and select three files-> open
 - Layers panel right click on file and open attribute table

Now

- Database->Dbmanager
- Select PostGIS->sdb_course-> public schema
- Provide table name baea_surveys

Accessing PostGIS from the commandline, the pgAdmin, QGIS

- C:\programefile\postgresql\10\bin\shp2pgsql
- C:\programefile\postgresql\10\bin\raster2pgsql

- pgAdmin4
- Open query tools

- Select * from baue_nests

Some other tables with geometry column

Spatial Reference ID

- Unique identifier for a coordinate reference system
 - Coordinate System :- based in angles based on earth
 - Projection
 - Zone
 - Datum
- In PostGIS you indicate the Spatial Reference by a number
- That number corresponds to well known Text
- Individual Geometries can have a spatial reference
- Geometry columns have a spatial reference

The Geometry Field

- The geometry field is what makes a table spatial
- Its just a column in the database that can store a feature geometry
- It includes the SRID, the geometry type, and the actual coordinates in binary form
- We can't interpret the binary data but PostGIS provides a number of functions that can output the geometry in human readable format
- binary data could be picture, shape any things
- These are functions that can output geometry as human redable text
- ST_AsText(), ST_AsEWKT(), ST_AsGeoJSON, etc.
- The combination of a geometry and the attribute that describes that geometry is known as features

Geometry vs Geography data types

- Two different ways to store a features spatial information
 - Geometry (data type). Based on plain surface
 - Geography. (data type). Based on circular earth , math behind calculation is complicated
- Geometry
 - Based on planner surface
 - Can be in a variety of coordinates reference systems defined by OGC
 - Are mathematically simple
 - Rich set of functions available
 - OGC standard , GEOS libraries, etc.
 - Accuracy declines as spatial extent increases due to cuveture

Getting started (loading data...)

Events Table

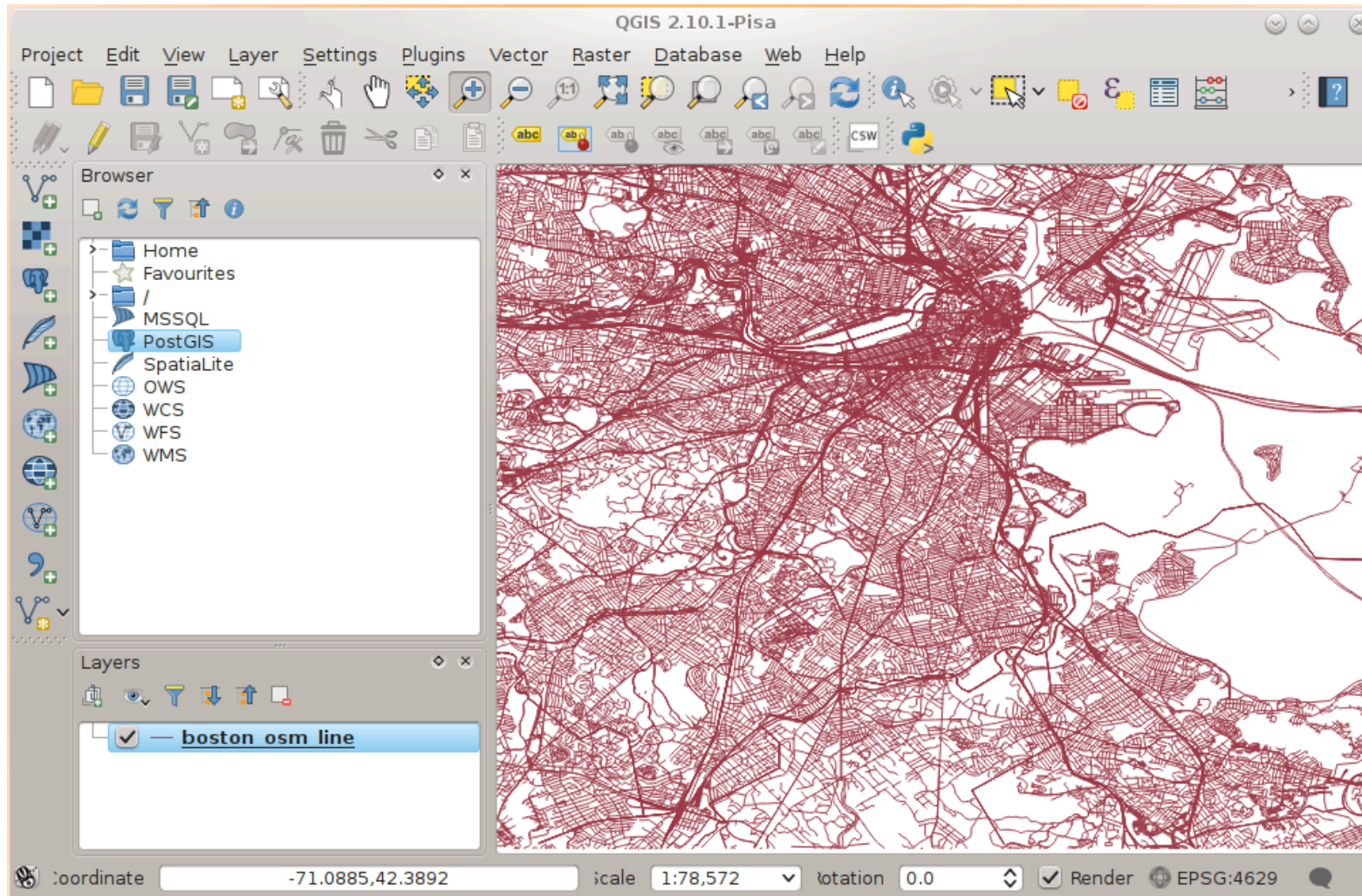
```
eventname character varying  
lat double precision  
lon double precision  
Date timestamp with time zone
```

events_geo Table

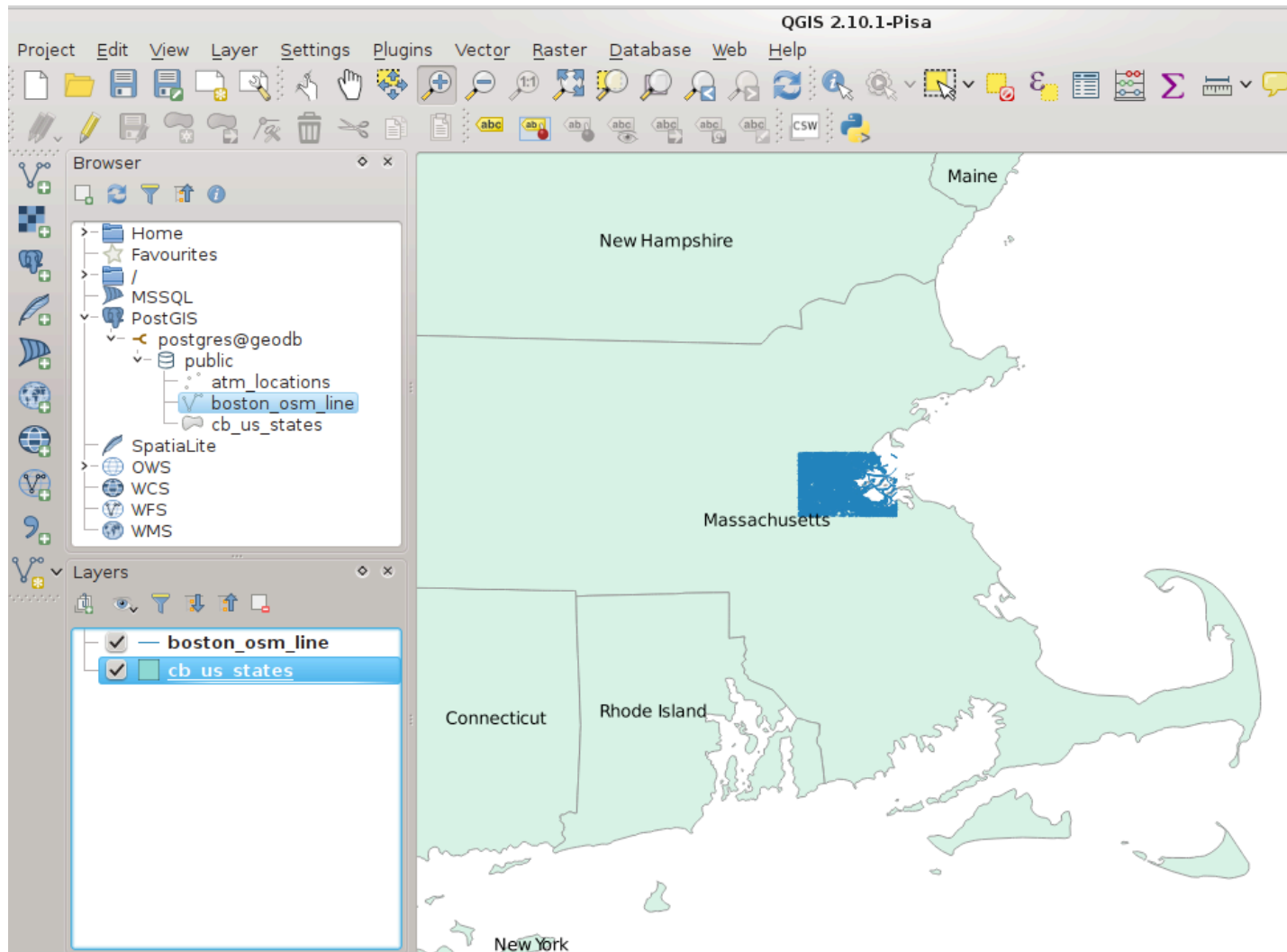
```
eventname character varying  
lat double precision  
Lon double precision  
Date timestamp with time zone
```

```
INSERT into events_geo(eventname,geom,date)  
SELECT eventname,  
st_setsrid(st_makepoint(lon,lat),4326) as geom,date  
FROM events;
```

Getting started (visual inspection)



Getting started (visual inspection)



Getting started (Geometry Input and Output)

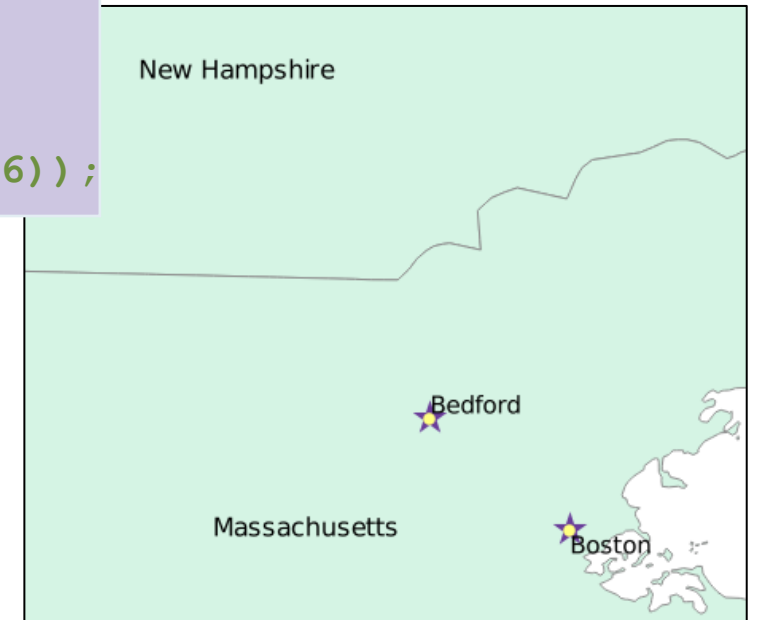
```
geodb=# select st_astext(geom),st_asgeojson(geom),st_askml(geom),st_asgml(geom)
from atm_locations limit 1;
-[ RECORD 1 ]+-----
st_astext  | POINT(-81.7842060002066 30.2915309995561)
st_asgeojson | {"type":"Point","coordinates":[-81.7842060002066,30.2915309995561]}
st_askml   | <Point><coordinates>-81.782688523520804,30.292466063489726</coordinates></Point>
st_asgml   | <gml:Point srsName="EPSG:4629"><gml:coordinates>-81.784206000206609,30.29153099955613</gml:coordinates></gml:Point>
```

Example Input function

```
Insert into cities (name,state,geom)
```

```
Values ('Bedford','MA',
```

```
ST_GeomFromText('POINT(-71.248063 42.510547)',4326));
```



Getting started (Power of GIS & SQL)

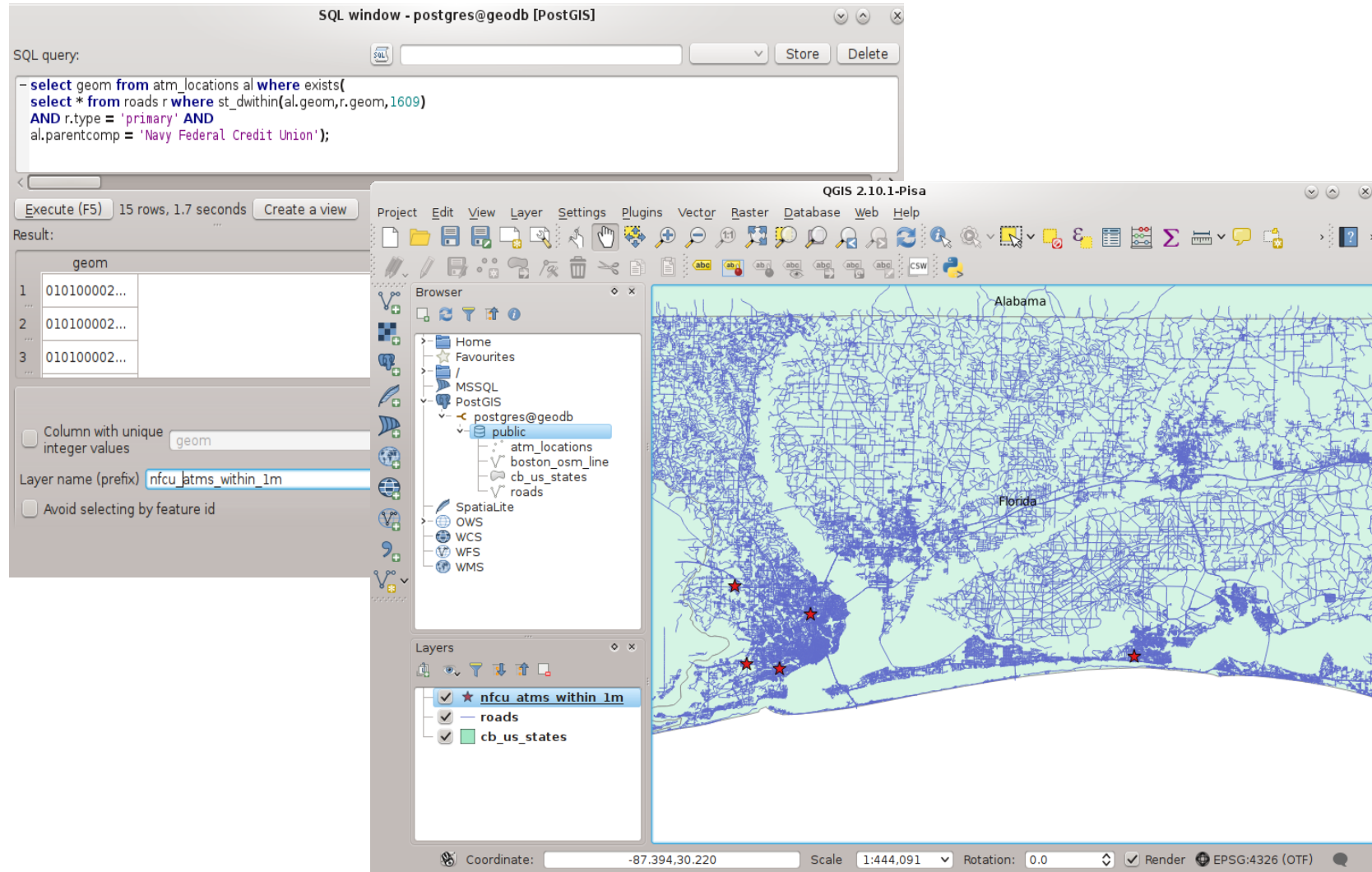
Write a function to tell whether a given lat/lon pair is within a Point-Radius ring

```
IF ST_DWithin(check_pt,point_pt,outerRadius) AND NOT
ST_DWithin(check_pt,point_pt,innerRadius)
THEN return 1;
ELSE
return 0;
END IF;
```

Create a route from a collection of waypoints captured in sequence

```
geodb=# create table paths as select routeid,st_makeline(geom) as geom
from (select routeid,seq,geom from waypoints order by seq) a
group by routeid;
```


Getting started (Power of GIS & SQL)



The screenshot displays the QGIS 2.10.1-Pisa interface. A SQL window is open, showing a query that filters ATM locations based on their proximity to primary roads. The query is as follows:

```
select geom from atm_locations al where exists(  
select * from roads r where st_dwithin(al.geom,r.geom,1609)  
AND r.type = 'primary' AND  
al.parentcomp = 'Navy Federal Credit Union');
```

The SQL window shows the execution result: 15 rows in 1.7 seconds. The result table has a single column named 'geom' with three rows of hexagonal geometry IDs.

The main QGIS window shows a map of Florida with a network of roads overlaid. Several red stars are placed on the map, indicating the locations of ATM branches. The 'Layers' panel on the left shows the following layers:

- ★ nfcu_atms_within_1m
- roads
- cb_us_states

The 'Browser' panel on the left shows the database structure, including the 'public' schema with tables for 'atm_locations', 'boston_osm_line', 'cb_us_states', and 'roads'.

Thank you!

edbpostgres.com

