**Introduction/Overview**

This document is designed to help you understand how you can use an Azure Database for PostgreSQL instance to enable the business scenario of site selection. We will rely on the robust geospatial capabilities within PostgreSQL for this.

Site selection is an important function in a variety of industries including retail, telecommunications, and transportation/logistics. In this example, we will execute the process of site selection from the point of view of a scooter company. The scooter company must decide where to put their first scooter charging facility in a city to ensure that scooters are charged and ready to go each morning.

Once the scooter company deploys their scooters, they can use location data from the scooters to make decisions about subsequent charging site locations. But for the initial site selection, they will be relying on information about the following:

* Population (higher is better)
* Distance from subway stations (higher is better)
* Average commute times (higher is better)
* Crime rate (lower is better)

In short, the scooter company wants to service higher numbers of potential customers with longer commutes farther from subway stations in areas with lower crime. They would like to situate their charging stations at corners/intersections not associated with highways so that they can take in scooters from two streets instead of one.

We will use New York City (NYC) for this example because we have easy access to all the data needed to illustrate this scenario for NYC. Please note that this walkthrough is heavily inspired by and adapted from materials found at the [PostGIS Introductory Workshop](https://postgis.net/workshops/postgis-intro/) that have been published under a [Attribution-ShareAlike 3.0 United States (CC BY-SA 3.0 US) license](https://creativecommons.org/licenses/by-sa/3.0/us/).

**Tools and Software Required**

To run this walkthrough, you will need the following:

* An Azure subscription (free option [here](https://azure.microsoft.com/free/free-account-faq/)) and [an Azure Database for PostgreSQL instance](https://ms.portal.azure.com/#create/hub) (please note that you can use a single server and basic service tier with 1 or 2 vCores and 50GB of storage – this setup will consume roughly $2 USD or less per day from either your paid or free trial subscription if you leave it running 24 hours a day).
* [PostGIS](http://postgis.net/install/) – the client tools include the PostGIS 2.0 Shapefile and DBF Loader Exporter which we will use to import geospatial data related to NYC to our PostgreSQL instance
* [QGIS](https://qgis.org/en/site/forusers/download.html) – an excellent toolset for visualizing geospatial data from PostgreSQL and other databases
* [pgAdmin](https://www.pgadmin.org/download/) – a robust administration tool for PostgreSQL

**Creating and Populating Your NYC Database**

Once your Azure Database for PostgreSQL instance is created, you will need to create a PostgreSQL database in that instance. Use pgAdmin to connect to your Azure Database for PostgreSQL instance using the connection string from Azure and the username/password that you specified during setup of your instance. Once pgAdmin is connected to your Azure Database for PostgreSQL instance, right-click on your instance name, select Create > Database… and create a database named nyc using the defaults.

Once your nyc database is created, you will next use pgAdmin to ensure that your nyc database has the postgis extension installed. Right-click on your newly created nyc database, select Create > Extension… and select postgis from the Name field in the “Create – Extension” popup window.

Now that full geospatial capabilities are enabled, you will use the PostGIS 2.0 Shapefile and DBF Loader Exporter application to load the four geospatial data elements for this example into the nyc database: streets, subway stations, census tracts, and homicides. Homicide data is our proxy for crime rate and we will use census tracts as a relatively small geographic area as our high-level granularity for site selection. Please note that you will need to unzip this geospatial data before you can import it into PostgreSQL. PostGIS has an excellent explanation of how to use the PostGIS 2.0 Shapefile and DBF Loader Exporter [here](https://postgis.net/workshops/postgis-intro/loading_data.html).

As you saw in the previous link, you will need to provide your login credentials for your Azure Database for PostgreSQL instance to the PostGIS 2.0 Shapefile and DBF Loader Exporter. Your username and password will be what you specified when you created your Azure Database for PostgreSQL instance. Your server host will be the public endpoint associated with your Azure Database for PostgreSQL instance – for example, your\_instance\_name.postgres.database.azure.com. The port will be 5432 and your database name will be nyc.

Once you have specified your login credentials, you will need to add the .shp files from our geospatial dataset which you extracted from the zip file containing this data. Click on the “Add File” button, add the four .shp files and configure the importer to use a value of 26918 for the SRID field of each of the geospatial entities.

You will know you have done this correctly when the window in your PostGIS 2.0 Shapefile and DBF Loader Exporter looks like this:

A screenshot of a cell phone

Description automatically generated

When you have configured this correctly, click the “Import” button to import these four geospatial entities into your nyc database.

Each of these four entities/tables has geospatial elements in the geom column and “traditional” relational elements in other columns. The geospatial characteristics of each entity might be a point (in the case of subway stations and homicides), one/multiple lines (in the case of streets), or one/multiple polygons (in the case of census tracts).

Details on these entities/tables and some of the useful columns therein are described in the data dictionary below.

## Geospatial Data Dictionary

Useful columns in the “nyc\_streets” table:

|  |  |
| --- | --- |
| Name | Common name of the street |
| Type | What kind of street is this? |
| Geom | MultiLinestring geometry of the street |

Useful columns in the “nyc\_subway\_stations” table:

|  |  |
| --- | --- |
| Name | Common name of the station |
| geom | Point geometry of the street |

Useful columns in the “nyc\_homicides” table:

|  |  |
| --- | --- |
| weapon | Weapon used in the homicide |
| geom | Point geometry of the homicide |

Useful columns in the “nyc\_census\_tracts” table:

|  |  |
| --- | --- |
| tractid | An 11-digit code that uniquely identifies every census tract. Eg: 36005000100 |
| geom | MultiPolygon boundary of the tract |

**Enriching Our Geospatial Dataset with Socioeconomic Data**

Our geospatial data has some relational elements already, but we need to combine this data with additional relational data to achieve our business objective.

For this additional data, we will use the file named nyc\_census\_sociodata.sql to import data into our nyc database. Within pgAdmin, right-click on the nyc database and select query tool. Paste the contents of our nyc\_census\_sociodata.sql file into this new query window and then execute the query by either pressing F5 on your keyboard or by clicking the small lightning bolt button on the toolbar.

This script creates a new relational table with socioeconomic attributes for each census tract and includes a column named tractid that acts as a foreign key to the nyc\_census\_tracts table. In addition, this script creates three views which together form an algorithm which scores each census tract by most of the characteristics that we care about: population, average transit time, and crime rate. We will set these characteristics and scores aside for a while to deal with the fourth characteristic (distance from subway stations) first.

**Visualizing our Geospatial Data**

To begin visualizing our geospatial data, open the QGIS application that you downloaded and installed previously. Within QGIS, start a new project, right-click on the PostGIS node in the browser area, and select New Connection…

A screenshot of a social media post

Description automatically generated

Input into the resulting window (highlighted fields below) the appropriate credentials and information for your Azure Database for PostgreSQL instance. You will likely want to use a friendly name, as shown below. Please note that SSL is required to connect to your Azure Database for PostgreSQL instance.

A screenshot of a computer

Description automatically generated

Once you have authenticated, you will see our four geospatial entities/tables in the nyc database: census tracts, homicides, streets, and subway stations. Select all four, right-click on any of them, and select “Add Selected Layers to Project” as shown below.

A screenshot of a social media post

Description automatically generated

Once the geospatial data is loaded into the QGIS project, we can turn on/off layers. From the default of seeing all layers in the resulting visualization, uncheck the box next to homicides so that we are only viewing census tracts, streets, and subway stations as shown below.

A screenshot of a cell phone

Description automatically generated

**Manipulating Geospatial Data and Re-Visualizing the Output**

Now that our initial setup in QGIS is complete, we will perform several operations back in pgAdmin. You may recall that our business scenario involves selecting a location at an intersection of streets so that our site has access to more than one street at a time. We can create the intersection table by running a script in the pgAdmin query tool. In pgAdmin, right-click on your nyc database and select Query Tool… to have a place to paste/run the following script:

CREATE TABLE public.nyc\_intersections

(

gid SERIAL PRIMARY KEY,

street1 character varying(200) COLLATE pg\_catalog."default",

street2 character varying(200) COLLATE pg\_catalog."default",

geom geometry NOT NULL

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

/\* calculate all the intersections in NYC based on the geometries of the streets \*/

INSERT INTO nyc\_intersections (street1, street2, geom)

SELECT s1.name, s2.name, ST\_Intersection(s1.geom, s2.geom) AS intersection\_geom

FROM nyc\_streets AS s1

JOIN nyc\_streets AS s2

ON ST\_Intersects(s1.geom, s2.geom)

WHERE

s1.name IS NOT NULL AND

s2.name IS NOT NULL AND

s1.name <> s2.name

ORDER BY s1.name, s2.name;

/\* Remove duplicate intersections \*/

DELETE FROM nyc\_intersections

WHERE gid NOT IN (

SELECT gid FROM (

SELECT MIN(gid) as gid, geom

FROM nyc\_intersections

GROUP BY geom

) AS uniqueintersections);

This script creates point geographies from the intersections of the lines that are streets. Because we are also performing a Cartesian join between two instances of the streets table, it also eliminates duplicate intersections – if we have the intersection of 1st Avenue and Allen Street, we do not also need the intersection of Allen Street and 1st Avenue.

Once you have created your intersections, return to QGIS. To visualize the newly-created intersections in QGIS, right-click on the PostGIS connection you originally created and select “Refresh”. You should see a new geospatial entity named nyc\_intersections alongside the original four entities that we saw before. Right-click on nyc\_intersections and select “Add Layer to Project”. With nearly 43,000 non-duplicate intersections, the QGIS visualization should fill up with new points that almost completely cover the other geospatial entities.

Because we want to only consider safe intersections not including a highway that are 500 meters or more from a subway station, we can replace the previous pgAdmin script with the following script.

/\* Remove from consideration intersections that are within 500 meters of a subway station \*/

DELETE FROM nyc\_intersections

WHERE gid IN(

SELECT nyc\_intersections.gid

FROM nyc\_intersections

JOIN nyc\_subway\_stations

ON ST\_DWithin(nyc\_intersections.geom, nyc\_subway\_stations.geom, 500)

);

/\* Delete those intersections that include one or more highways or highway links) \*/

DELETE FROM nyc\_intersections

WHERE gid IN (

SELECT nyc\_intersections.gid

FROM nyc\_intersections

JOIN (SELECT \* FROM nyc\_streets WHERE type IN ('motorway', 'motorway\_link')) nyc\_streets

ON ST\_DWithin(nyc\_intersections.geom, nyc\_streets.geom, 10)

);

Running this script in pgAdmin will remove over 16,000 intersections from consideration. We can see the impact of this manipulation by returning to QGIS and refreshing our view of the intersections. To do this, right-click on the nyc\_intersections listing in the layers window of QGIS, and select “Update SQL Layer…” as shown below.

A screenshot of a map

Description automatically generated

Click on the Update button in the lower-right corner of the resulting DB Manager window to refresh the intersection data, as shown below:

A screenshot of a social media post

Description automatically generated

When QGIS finishes refreshing the intersections layer, you will see that there is now a buffer zone around subway stops. In case the subway stations and the intersections have similar color dots associated with them, this buffer zone becomes easily visible if you toggle the visibility of the subway stations.

**Final Manipulation of Data**

Now that we have removed from consideration the intersections that involve highways or are too near to subway stations, we can return to narrowing down our choices based on population, average commute times, and crime rate. You may recall that one of the views we created previously scores census tracts by these variables – we can see the top 10 census tract IDs by running the following query in pgAdmin:

SELECT tractid

FROM vw\_nyc\_site\_selection\_data

LIMIT 10

The results of this query enable us to build a query that excludes all intersections that are not in or near this Top 10 list of census tracts:

/\* delete those intersections that are NOT WITHIN 500 METERS of the top 10 census tracts \*/

DELETE FROM nyc\_intersections

WHERE gid NOT IN (

SELECT nyc\_intersections.gid

FROM nyc\_intersections

JOIN nyc\_census\_tracts

ON ST\_DWithin(nyc\_intersections.geom, nyc\_census\_tracts.geom, 500)

WHERE tractid IN ('36005046201', '36061013600', '36061015400', '36061013800', '36061006600', '36061013400', '36061024500', '36061015700', '36061013200', '36081045500')

)

After you run this last query in pgAdmin, you can refresh the nyc\_intersections layer again in QGIS to see that we have narrowed our considerations significantly, down to just over 350 intersections that meet our criteria. It may not be possible to automatically refine further than this because we would need to also evaluate property availability, rental/purchase prices, etc. However, we have very quickly eliminated over 99% of the intersections in New York City through a combination of traditional and geospatial queries as shown below.

A close up of a map

Description automatically generated

For even more details on geospatial data and queries, please do refer to the [Introduction to PostGIS workshop](https://postgis.net/workshops/postgis-intro/) which inspired and informed this walkthrough.