**PostgreSQL Database Backup and Queries**

After I got the original database dump from the previous group, I created a similar database based on the original schema that contains the same attributes. Then I imported the updated data from the AFDC website to the *PostgreSQL* database. It also uses the *PostGIS* extension to create a reduced station network and generating the OD matrix via the *Esri REST API* for *Python*. Here is a summary of the three steps I have taken:

* The new data of 2020 for the hydrogen stations and compressed natural gas stations are in csv format, and it is imported into the database named “afv” on my computer. The *Postgres* database are created and managed using both command-line tools and interfaces such as *pgAdmin* and *Postgres App*.
* The reduced network is constricted by the *ST\_distance* and *ST\_Dwithin* functions in the *PostGIS* extension. The former function returns the straight-line geodetic distance (not the actual driving distance), and the latter selects all pairs of nodes within 400 miles of the geodetic distance.
* The *Esri API* has many ways to access in *Python*, and I have to use *SAML* login to be able to use the ASU students’ *Esri* credits. Although there are other methods such as anonymous login and *Esri* build-in user login, they only provide limited credits which are not sufficient for large-size data.

The csv files from AFDC website have many fields that are unnecessary for my project, such as the *telephone numbers* and the *open date*. I cleaned the data by deleting the excessive fields in the *Excel* before importing the csv files to the database. The *id, zip, station name, lat*, and *lon* fields are preserved. The data type of the *id* is *serial*, which are automatic increment integers. The *lat* and *lon* fields should be set to *double precision,* while the rest ones should be *varchar* (character varying). I used to set the *zip* filed to *int*, but then I realized some zones in North America has letters in their zip codes. Thus, the *varchar* data type is more suitable for the *zip* field.

Then I created a table that contains *geometry* objects that are transferred from the *lat* and *lon*. However, the distances that are returned from the *ST\_distance* function are extremely small, such as 0.000001, because the function returns the value in the unit of the degree of the *lat* and *lon* if the input arguments are *geometries*. By using *geography* objects instead, it will return distance in meters. One can either transfer the objects to *geography* objects using the command:

st\_distance(h2stations.geom::geography)

Or create the *geography* objects when importing the csv data:

ALTER TABLE h2stations ADD COLUMN geog geography(POINT,4326);

UPDATE h2stations SET geog = ST\_GeogFromText('SRID=4326;POINT(' || lon || ' ' || lat || ')');

To make all pairs of stations within 400 miles in the database, one can select the *id* from one table and left join it with the copy of the table, then exclude the pairs that have the same *id* and the reversed-id pair. For example, we do not need the pair when id1 = id2 because it means they are the same stations, and we only need one of the pairs: (id1 = 1, id2 = 2), or (id1 = 2, id2 = 1). This could be achieved by one of the two commands:

CREATE TABLE h2pairsDistinct AS

SELECT DISTINCT ON (h2pairs.st\_distance) id1, id2 FROM h2pairs WHERE id1 != id2;

Or alternatively:

SELECT h2stations.id id1, h2stationsCopy.id id2, st\_distance(h2stations.geog, h2stationsCopy.geog) FROM h2stations, h2stationsCopy WHERE st\_dwithin(h2stations.geog, h2stationsCopy.geog, 643738) and h2stations.id < h2stationscopy.id;

Then, one can left join the *lat* and *lon* to the table of the station pairs (<400 miles) to create a table that can be used for *Esri API*. The command is here:

CREATE TABLE leftjoin1 AS SELECT h2pairsdistinct.id1, h2pairsdistinct.id2, h2stations.lat, h2stations.lon FROM h2pairsdistinct LEFT JOIN h2stations ON h2stations.id = h2pairsdistinct.id1;

The data from AFDC already has an id field that has been used for the previous project, so I deleted the id field I created and uses the primary id to generate the OD matrix.