# Considerations for using Quest to implement the 2024 Geographic Access Specifications

This is meant to be a helpful document in how to apply Quest Analytics Suite (QAS) and SQL code to modeling the proposed 2024 geographic access specifications. This is for analysts or plan staff who are already using QAS and SQL, and do not have the means to utilize Python or cannot license a new service tool for calculating geographic access. These notes assume some familiarity with SQL and QAS.

## Member Geographic Designations

* **SQL approach**
  + Geocode member dataset using QAS
  + Load geocoded data into a database table.
  + Load population center attributes into a separate database table.
    - Provided by OHA and includes centroid coordinates, and designation type
  + Transform QAS lat/long outputs
    - QAS uses specially formatted latitude and longitude values. In order to calculate distance, we need to convert these into a standard format.
    - Add decimals after first 2 digits for latitude and after the first 3 digits for longitude
    - Make longitude values negative
  + Using SQL and the two database tables created previously, cross join member dataset to centroid list (proliferates record count, can pre-filter on county)
  + Calculate proximity of each member to population centroids using Spherical Law of Cosines formula in SQL.
    - Helpful reference for Spherical Law of Cosines: <https://www.movable-type.co.uk/scripts/latlong.html>
    - Example SQL code is included on last page. This code may need to be adapted to your specific SQL environment.
    - Partition results by member and filter results for each member’s lowest distance only. Then filter for only those members less than 10 mi from a pop center. You have now identified your Large Urban and Urban members.
  + Identify geographic designation (Rural or Frontier) for remainder of membership based on geocoded County value
* **Excel approach**
  + Similar process as SQL, but using Power Query to build tables, join tables, calculate distances, and filter results
* **Python approach**
  + Use geopandas library and spatial join geocoded member points against OHA shapefile
  + Alternatively use geocoded member data and calculate Spherical Law of Cosines in python

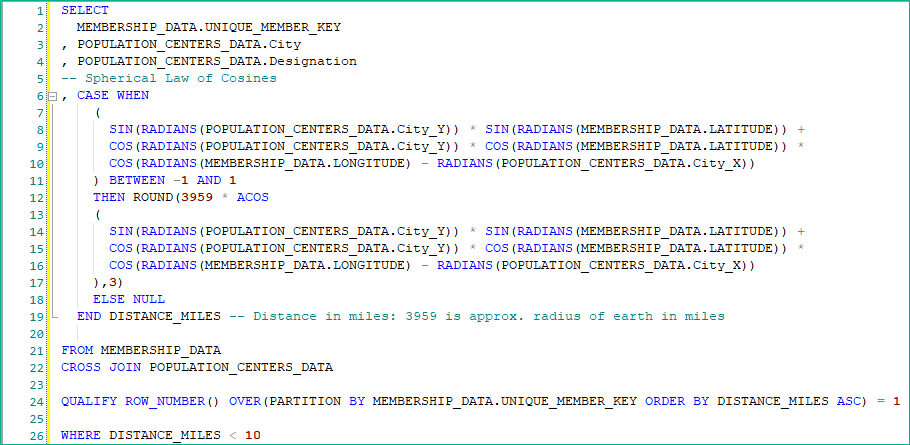
## Provider Specialties Transformation

* DSN provider specialties must be represented in a single Specialty column for QAS. Starting with the DSN:
* Join your completed DSN to the specialty matrix on taxonomy
* Then transform the matrix indicator fields into a single specialty column. One approach:
  + Possible in SQL, python’s Pandas library, or Excel Power Query – melt your dataset into a new/temp table that includes a unique record key from the original table.
  + Then join that back to the original DSN table, keeping only records with an affirmative indicator for each specialty. Then remove all original specialty matrix columns.
  + Typical function names might include: melt, unpivot, pivot long
    - For Python, this can be done in Pandas library with melt function
* Another approach (see python code example):
  + Create list of specialties using Provider Specialty Matrix header rows (columns 9-95; should be 87 specialties)
  + Loop through list of specialties
  + For each specialty:
    - Query merged table for Specialty=1
    - Add column for Specialty name
    - Drop other Provider Specialty Matrix columns
  + Combine/concatenate specialty tables

## Geographic Access Model in QAS

* Final member and provider datasets can be loaded to format of your choice to be used as a data source for QAS
* Define member groups based on geographic designations
* Build access criteria for member groups and specialty tiers

SQL code example for Spherical Law of Cosines:



Python code example to transform Specialty into a single column:

