**Problem 1—Programming Skills and Databases Management Write-up**

The code I used to complete this task (with some description of the process) can be found in the submitted Python Jupyter Notebook. For working in Python as the client-side language, I made use of the pyscopg2 library.

First, I familiarized myself with the data with a Pandas dataframe, determining how many unique node types there were, and the general structure of the data.

Then, reasoning that JSON-like collections of lists of dictionaries make for ready importation into databases, I created such data structures for both the nodes and flows data. The description of the *node\_name* field as “a string recording the type of supply chain node” was confusing to me at first, but I reasoned that it must actually mean the name of a particular node, e.g., the name of the port, and not just “port”, even though port would be the type strictly speaking.

My numbering scheme for the node\_id was a bit simple—I numbered them in numerical order of first appearance within the dataframe, along the row-axis.

An issue I encountered with the **flows** *path* field was in trying to create a data type INTEGER[] referencing the *node\_id* as part of the table’s definition. Researching the problem led me to believe that this is not allowed in SQL (e.g. a one-to-many relationship conditioned on the primary-foreign key relationship). I suspect there was a possible work-around of some sort involving constraints or an intermediate table, but the description of the problem requiring 2 tables led me to believe that the latter would be an incorrect method. I decided that, if the actual values in a *path* array accurately matched the *node\_id* elements, and that their arrangement in a *path*, as reflecting a common ordering of *node\_type* elements, was consistent across the entire table, then the relationship would be well-represented and could be used to faithfully reconstruct the path of *node\_name* elements programmatically.

I computed aggregated volumes (sums) for particular *paths* prior to creation of the **flows**table, since trying to do this first in psql led to an error in which the *path* arrays would lose their ordering.

One final note I would like to add is that the code I’ve written does not make use of many functions. This is because of the format of the Jupyter Notebook, which while a convenient tool, leads me to write code as easily re-executable cells, somewhat obviating the benefits of creating reusable functions. If I had made use of scripts instead, I would have made heavier use of functional programming.