Creation of a procurement database with Requisition and Purchasing Order information

Background (real world request with live data):

The purchasing and procurements department of a local company had been running their procurements process through a series of un-linked excel files, that were maintained and updated individually by two different employees. In some cases, data was updated or duplicated in one file, but not updated with the same information in other files. The process had become unwieldy, and the company suggested trying to use the original files to set up a database, which would facilitate addition of new entries in the future, while ensuring updates to all tables with any new information.

There were some suggestions made for the proposed database scheme that could not be implemented at the time. For example, there was a suggestion that the approval data table (tracking purchase order requests), and the order header table (tracking full, approved purchase orders) be linked by using the requisition number(also referred to as document number) as a foreign key in the order header table referencing the order requests in the approval data table. However, the company had a practice of reusing requisition numbers from orders that were not approved, and as such, the requisition number/document number was not a unique value that could be used as a primary key for approval data (and thus could not serve as a foreign key in the order header table).

Therefore, this suggestion cannot be implemented unless the company decides to change their internal practice. Until then, the information for requisition number must continue to be update manually in the database.

Extract:

The 5 original Excel files, which comprised the prior procurement tracking system and came directly from the company's eProcurement tool, were converted to csv format:

- **Approval data** – tracking purchase order requests
- **Exchange rates** – tracking currency exchange rates, needed to convert prices/cost for orders to and from different countries
- **Order header** – tracking full, approved purchase orders as individual line items
- **Order Line** – tracking individual items within approved purchase orders, with each item as an individual entry
- **STAR data** – tracking codes for different characteristics attributed to purchase request items

Data was the loaded into python pandas library for data transformation and data cleaning.

Transform:

Each of the files needed to be cleaned up: renaming column headers, eliminating unnecessary columns, and rearranging column order to prioritize key data first, and to provide a better sense of flow during data entry and revision. Two tables were particularly difficult to clean:

- **Approval data**: The main challenge here was converting the date-time columns to a date-24h format and then to have Postgres accept the values of this conversion. In the end, it was decided to perform a second conversion of the date-24h back to string values in order to help Postgres to accept the information.

- **Order Lines**: In theory, every purchase order number in the order header table should have appeared in the order line table. However, given that the two tables had been updated manually, and at times by separate employees, there were some purchase orders in the order line data that was missing from order header data. Since part of the project required linking the two tables using the purchase order number as a primary/foreign key, the main challenge was to identify and delete problematic values. This was accomplished by performing an INNER JOIN in PostGres, with the resulting data saved as a new order line table (preserving the old, unaltered data in the old order line table), and exported back into pandas.

Load:

Loading was done through an engine linked directly to PostGres. As mentioned above, an additional table called old_order_lines was created in order to preserve records that did not pass the quality control needed at the data transformation phase. The company could then later investigate further the reason for inaccuracies in the data entries and take steps to update the data for inclusion in the new database.

Test Queries:

Once the new database scheme and populated with the cleaned data, the data was reloaded into pandas. A couple test queries were run at the end of the jupyter notebook used for ETL, to confirm that the data had been loaded correctly and could be queried effectively.