Generating Transaction_Items Table

Preliminary Plan:

- 7500 customer transactions will be created.

- Each customer transaction will initially be stored as a list of length 9 where the first item in the list will be the unique transaction ID and the following 8 values are lists of length 2. Each nested list will contain the randomly selected item (as its unique item_id) and its price separated by the * character.

- The items will be selected using the following process:
  - A for-loop will iterate once for each unique transaction and first generate the transaction_id, which assigns the string 'trans' + the nth iteration of the for-loop as the unique transaction_id.

- Next, the for-loop will iterate over each category (smoothie and juice here are joined as 1 category making 4 total categories) and skip, or select 1 or 2 items.

- The list of lists will be appended to a pandas DataFrame with the columns [transaction_id, item_price1, item_price2, … item_price8].

- Then the DataFrame will be 'melted' so that there are just 2 columns: [transaction_id, item_price]. In this melted DataFrame, transaction_id will no longer be a unique column, as it will be duplicated for the number of items that the customer ordered.

- The item_price column will be split so that item_id and price are in their own unique columns. A new column 'transaction_item_id' will be created that is simply an ID for each row where each ID is the nth row of the DataFrame.

- Next, a 5th column will be created 'quantity'. This will search by transaction_id number for duplicate rows and if a duplicate is found, the 'quantity' column will be assigned the number 2. Otherwise, if there are no duplicates, a 1 will be assigned to the quantity column.

The data for the transaction_items table is complete.

The Next step is to create the transactions table from the data in the transaction_items table. This process will be done through PostgreSQL.

Why that didn't work:

- everything worked fluidly up until the melting step. I found it impossible to melt the data and retain the unique transaction IDs. When trying to melt the DataFrame, the best result I had was each item was assigned a unique transaction ID instead of the item retaining its original transaction ID.
- I ran into another issue when I began to create a column for item quantity. The issue was that I didn't have a detailed plan in place to accomplish this task. Without brainstorming first

and coming up with a plan, I again fumbled until I committed myself to planning and then executing. I also wrote out some of my thoughts and different options as I conceived them.

What worked:

First, I cried and tried 1 million different ways to melt or pivot the DataFrame. Then I returned to the for-loop that created the data. I fumbled for almost an hour before I decided to focus my thoughts and write out a quick plan. Creating an outline for a plan enabled me to avoid overwhelming myself with different ideas and the different facets of the problem all at once

Separation of columns:

- I returned to the generation phase and created 1 list per transaction where each element in the list was a string containing the transaction id, the item id, and the price of that item all separated by an '_' character.
- The transaction list was added to a master list and then reset in the next iteration of the for-loop. In the end, the master list was transformed into a DataFrame and the split method was used to separate the transaction id, item id, and item price into 3 separate columns.

Creating quantity column:

- I used the .duplicated() method (specifying to label ALL duplicates as opposed to only the first, which is the default setting) to create a Boolean column called duplicate labeling duplicate rows as True.
- Then I created a column for the quantity and all rows quantity = 1.
- Next, I used a for-loop to iterate row by row and if the value for duplicate = True, then the quantity value would be changed to 2. This still leaves all duplicates in the dataframe.
- Next, I used the drop_duplicates() method (specifying the keep parameter = first or last). This dropped all duplicates.
- Finally, I dropped the duplicate column.