Notes and Justifications

**Looking At Data Preperation**

- Converted the dtypes of Transaction Date and Act Ship date to datetime object, both of these are strings (‘objects’ in the dataframe) and need to be datetime since they are dates. The purpose of this is to be able to filter transaction date for order only made in 2021 (since if you do this in Traverse, it will still include orders that were made in different years)

- Changed the dtype of Transaciton ID to string and formatted to look like the transaction id’s in Traverse. The code, *data[‘Transaction ID’].apply(lambda x: x.zfill(8))* , This code applies the zfill() function to each element in the column Transaction ID of the dataframe data. Zfill just places a user specified amount of zeros in front of a string.

- There are customer id’s included in the data that are not customers, some of them are used for trade shows others are used for the print shop, and others are used for getting items collected for taking photos. These customer id’s were removed.

In the code *data = data[~data['Customer ID'].isin(customer\_id\_filter)]* the squiggly line means ‘to take the opposite of’ . so this code is keeping the data that is not in the customer id filter, which is a list containing the customer id’s that we want to remove.

- Saved a text file of unique items to quickly skim over to make sure that all the items made sense and should be here. (example, there are some clothing items that are still in the system, but aren’t being sold)

- So there are items without Price Codes. Price Code is the only way to get higher level groupings of items inside SO Detail History. The problem with Price Code is there are a lot of items that don’t have them and items that do have them have jumbled price codes that don’t make much sense. There is a betteralternative that can be used called Product Lines. Product Lines for the items can be found in Traverse in the IN Item View. Made an excel file of all the items with their price codes.

- Load the IN Item View into a dataframe called items, then filter it to keep the columns that are useful.

- This list needs to be a unique list of items, since the list will be used in a join. The items dataframe was checked for duplicates, which existed, then the duplicates were dropped.

- the items dataframe was joined with the full data in a left join (just in case there were items in the full data that did not show up in the items datafame, but this wasn’t the case)

- Loaded a particular transaction id to be sure the join was successful

- removed Price Code column and reordered the dataframe. Renamed the columns ‘Units’ since it has a whitespace in it

- checked to see if there was any unique items that did not have a product line ( the reason I did unique was so that each instance of one of these items with no product line would show once, instead of for each order these appeared in. For example, purr-minor kit is shown once, but if I didn’t make this unique, then it would have shown for each order that was made of it). There were items with nan product lines

- For some reason, Traverse failed to include product lines for all of these items (except crucible pots). Traverse is a strange application, it does strange things for unknown reasons. Looked up each of these nan product line items in Traverse for their product lines and added them to the data dataframe in python. There was an item called ‘sun’ that could be removed.

- looked at the Line Items with nan product lines, all of these were NAN across the rows and had Amount as 0, so these could all be removed

- looked at Misc and there was one entry that needed to stay

- Adding a Drop ship column to the data.

- Looked at the NAN data in the dataset. For the Act Ship Date, entries that have NaT were not shipped. Item ID, Product Line, and Units that are NaN can be changed to No Item ID, No Product Line and No Units

- Looked at the NaN values for Ship Via, these are shipping amounts of unknown value. The values would have to be manually checked and inserted and also ABR Imagery switched shipping carriers in 2021, so a lot of the shipping values would be lost.

- this handles all of the NAN data. Note that the Act Ship Date NaT should stay and so should the ones for Location ID since these were orders placed in 2021 but did not ship (most of these are back ordered items, ie items that ABR Imagery did not have when the order was placed so the order was shipped with the instock items and the rest were backordered or replaced)

- restructured the order of columns and re-indexed the dataframe to keep things clean

- Adding Geographic information to orders. It is impossible to get geographic info on customer from SO Detail History in Traverse. Made a file from Traverse called ‘AR Transaction History View All’. This file contains geographic information on the customers for the orders. Will join this with the main dataset.

- loaded the file

- converted Transaction ID to Traverse format and select data from the AR transaction History View All that I needed

- performed the join, left

- reorganized the columns

- Went into some detail on how Unit Cost Sell is a Disaster and is data that is not usable. This is because there are items that have the Unit Cost Sell greater than the Unit Price Sell. There are items where the Unit Cost Sell is equal to the Unit Price Sell. Obviously in practice this is not the case, but in Traverse, which is the only resource that is available for this data, it is. So I will omit Unit cost sell. An estimation could be made, but for the project, the data must be true and validated, thus estimations of the unit cost sell will not be made.

- addressing address data that is NaN

- There were NaNs in Ship to Region, only need a ship to region for US orders. The search was confined to US orders that had a ship to region of NaN and for the column of Ship To City

- next, orders that had USA as the shipping count but NaN region were checked, all of those orders had sales tax and Location ID of B01, so this must mean these ‘shipped’ from Bloomington, IN (really this were pick up orders)

- Checked the remaining Ship To Region NaN counts in each country and confirmed that the remaining NaNs were in foreign countries, so nothing more need sto be done with them

* Checked the NaN cities and only needed to change 1 Transaction ID city

- removed History Type of Payment, doesn’t give any valuable information

- Corrected Foreign Countries. The Ship To Country column can have the same country, but with different lettering. Each country was examined and a unique spelling was given for the countries that needed it. There were 5 countries that needed this altered

- The Ship To Region for US orders also has the same state in multiple spelling formats. Each state was given a unique identifier and the ship to region was changed to this identifier when necessary, overall, there were 40 states that needed this changed. Tableau was also used to help with this, the Ship To Region would be placed in a new sheet and looked over to see if each state had 1 single identifier (can also see the count there of 50 states in the end, which was used to confirm that indeed each state only had 1 unique identifier)

- Due to Traverse and the web being strange, there are orders made in the US that have the Ship To Region being ‘A’, there were manually looked over by comparing to Traverse and adjustments were made

- Changed some Item ID’s of ‘No Item ID’ over to Freight and Sales Tax to make more sense with my visualizations.

- Did add Customer Names by creating an excel file in Traverse and joining this file to the main dataset, but this part is omitted to the public

- note that all the item prices have been smartly randomized. That is to say, the prices of the products are random, but maintain their magnitude in terms of how high their actual prices are