Master folder: R:\FDA\2732 - FDA Tribal Tobacco Retailers\Technical\Tribal\_Master

Project Goals:

1. Confirm whether the retail establishment is in Indian Country

2. Identify or update the associated federally-recognized tribe

3. Confirm the name and address of each establishment

4. Identify retailers that are no longer in operation

5. Identify the legal owner for each establishment

6. (Manufacturers’ only) Identify the nature of the manufacturer’s activities, if needed

Tribal\_Master py files 0-3 complete goals 1-4. Final output of all this is step\_3\_work\output\full\_retailer\_list.csv.

Master to do list:

1. Complete Goal 5 – Ownership Identification
2. Complete Manufacturers list
3. Complete compiling into final data spreadsheets
4. Complete QA of all data

1. Complete Goal 5 – Ownership Identification

*1a. Complete independent scraping of ownership documents.*

List of the status of each state is recorded in input/State business websites final v2.xlsx. There are 35 states that have at least one record. I completed automatic scraping for the top 10 states with the largest number of retailers. These are found in /state\_scrapers. They each do the following for their state:

* Read in full\_retailer\_list.csv and filter to records for that state
* Scrap possible matches from state SoS website search engine
* Select the best match from state SoS
* Output a copy of full\_retailer\_list.csv with ownership added to their state’s records

Their output csv’s are stored in step\_4\_work, and the supporting documentation (website screenshots, articles of incorporation, etc.) is saved in either output/Existing Retailers or output/New Retailers.

The other 25 states need to be scraped by hand. I finished the 9 smallest states already, and record my manual determinations for these records in 4a\_manual\_owner\_doc.py, which reads in ‘step\_3\_work/output/full\_retailer\_list.csv ‘ makes the recorded determinations , and outputs ‘step\_4\_work/output/full\_retailer\_list\_w\_owner.csv’. There are 777 records in the remaining 16 states, to save time let’s default to using Lexis data for these states. I think all of these are small enough that it’s more efficient to do by hand rather than building a scraper for.

*1b. Merge in alternate sources of ownership*

Primarily, I am anticipating this is the Lexis data. Once we get this data, we need to:

* Go through and manually check for false positives in their assigned matches
* Merge in the ownership data with our own results.
* Deal with conflicts between independent matches and Lexis matches (am thinking would just give Lexis matches precedence.)

Mason has also noted Hoovers may have some limited usefulness in verifying ownership.

*1c. Retrieve documents from SoS for Lexis matches.*

For those retailers matched by Lexis, we need to retrieve the documents from the SoS website, as Lexis can’t do this for us. We can adapt the state scrapers’ document retrieval code (each scraper has a \_get\_docs.py or a similar filename) for the states we have built scrapers for. Not all websites offer documents for free, which in that case we just take screenshots of the website. Documents should be stored in either output/Existing Retailers or output/New Retailers. Note the standardized filenames I’ve used in these folders.

For those states a scraper hasn’t been built for, we’ll need to either manually retrieve the document or build separate document retrieval code, whichever is more efficient. We should also record the latest year documents are available for each retailer, if possible.

One other note, I built a scraper for NY that identifies matches, but captcha prevented me from building a document retrieval tool. The matches identified in "\step\_4\_work\NY\NY\_ownership\_results.csv" will have to have their documents manually scraped as well.

*1d. Compile ownership results into a single file*

I update all results from the automatic scraping in 4b\_compile\_scrap\_results.py. This file reads in ‘step\_4\_work/output/full\_retailer\_list\_w\_owner.csv’, updates the ownership data with the matches found from the state scrapers, and outputs the final output file of this step is step\_4\_work/output/full\_retailer\_list\_w\_owner\_final.csv.

Once completed, the remaining ownership match results from Lexis should be added to the 4b .py file.

2. Complete Manufacturers list

Mason has started on this. Should end up with spreadsheets in similar format to the retailers.

3. Complete compiling into final data spreadsheets

5\_create\_documentation\_sheet.py creates 7 spreadsheets for submission in the format that we have agreed with FDA on. This file is complete for all the data I have scraped so far. These are the 7 files in /output that start with ‘#\_’:

1\_Original\_Data.csv: The original data FDA sent us (existing retailers).

2\_Revised\_Data\_Change\_Log.csv: The revised data for existing retailers, with columns that show tracked changes with where we made changes to their existing data

3\_Revised\_Data\_Clean.csv: The existing retailers revised data with no track change columns

4\_Existing\_Sources\_Documentation.xlsx: Hyperlinks to the documents for each existing retailer.

5\_New\_Retailers.csv: Data on new retailers identified in Nielsen data in the same format as 3\_Revised\_Data\_Clean.

6\_Nielsen\_Supporting\_Data.csv: Raw data from Nielsen for the new retailers identified.

7\_New\_Sources\_Documentation.xlsx: Hyperlinks to the documents for each new retailer.

Once everything else is finished, we should make sure these files are still properly creating these files.

4. Complete QA of all data

Things I can think of:

* Double check Peyton’s goal 1-4 QA file (3i\_Peyton Tribal Retailers QA v2.py)
* Double check the matching results of the 10 scrapers (similar to Peyton’s QA above; I think there are some false positives with the chain stores).
* Overwrite ownership determinations for the retailers Peyton overwrote in 3i\_Peyton Tribal Retailers QA v2.py; scrapers would have used the wrong data for these.
* Double check the hyperlinks in the final source documentation spreadsheets work.