Data Pipeline & API

**Overview**

Create a data pipeline to efficiently get, validate, transform, and save data from the WHO Global Health Observatory to a dedicated AWS server. Data will get extracted from the WHO GHO database via the Athena API in python, and initially stored locally. The data will undergo validation: which will indicate that the data has all required fields, and transformation: which will transform the data into a format which will be specified. After the data has gone through these steps it will be stored via cloud storage in AWS.

**Technical Steps**

**API** – The first step of the process, we will be using the Athena API from the WHO Global Health Observatory. The API will retrieve the data via requests using the following http format <https://apps.who.int/gho/athena/api/GHO/> under the template of [*http:****//****HOST****[:****PORT****]/****PATH****/****athena****/****INSTANCE****/[****DIMENSION****[/****CODE****[,****CODE2****[,****CODEn****]][.****EXTENSION****][?****QUERY\_PARAMETERS*](http://HOST[:PORT]/PATH/athena/INSTANCE/%5bDIMENSION%5b/CODE%5b,CODE2%5b,CODEn%5d%5d%5b.EXTENSION%5d%5b?QUERY_PARAMETERS)***]]].*** No API key is not required as it is public data and accessible to anyone. The data want is that of the cigarette and tobacco use estimates, and more specifically only from the countries of: US, Canada, China, and Japan. The Dimension for our data is *“M\_Est\_tob\_curr\_std”* and the filter would be *filter=COUNTRY:USA; COUNTRY:CAN; COUNTRY:CHN; COUNTRY:JPN*, so our API link would be *https://apps.who.int/gho/athena/api/GHO/M\_Est\_tob\_curr\_std.json?profile=simple&filter=COUNTRY:USA;COUNTRY:CAN;COUNTRY:CHN;COUNTRY:JPN .*

**Staging** – After getting the filtered JSON data we want, the data will be stored and timestamped in a staging folder. This allows us to have a copy of the data for reference before it enters any stage of the actual pipeline, so that we may determine a lineage and be able to debug efficiently.

**Validation** – At this stage of the pipeline the data will be validated to make sure that entry has all the required fields. Any entry with missing fields will be supplied with a default value. Data will be checked for dupes sequentially as there is no unique identifier for each entry. The idea for this is that Data will first be parsed by country, then making sure that each year of each country is unique. Example: We have a dictionary with each key being a country that we want, and each country key’s value is a dictionary with three more keys, each being a sex: Male, Female, and Both. And each of those keys have a list of all the years, with an additional element at the end of each being a counter. We validate by loading the JSON file into the program as a dictionary, we iterate through each entry in the JSON dictionary and check to initially see first if the country of the entry matches a key of our own dictionary, then we check to see if the sex of the JSON dictionary matches a key of our currently selected country key, and then we see if the year of the entry is in the list of currently selected sex key, and if it is we remove it from the list, and add 1 to the counter at the end of the list. If at any step above it does the entry element does not match whatever it is we are looking for from our dictionary, it fails to validate. If the total counter at the end of every list in the dictionary is not 9 (all the years which are elements in the list) the program fails to validate. (Very rough code below. This validates only for USA. Needs additional work to check other fields)

1. **import** json
3. f **=** open('data.json')
4. data **=** json.load(f)
5. f.close()
7. is\_validated **=** True
8. validate **=** {"United States of America": {"Male":[2000, 2005, 2010, 2015, 2018, 2019, 2020, 2023, 2025, 0],
9. "Female":[2000, 2005, 2010, 2015, 2018, 2019, 2020, 2023, 2025, 0], "Both sexes":[2000, 2005, 2010, 2015, 2018, 2019, 2020, 2023, 2025, 0]}}
11. **for** entry **in** data["fact"]:
12. **if** entry["dim"]["COUNTRY"] **in** validate.keys():
13. **if** entry["dim"]["SEX"] **in** validate[entry["dim"]["COUNTRY"]].keys():
14. **if** int(entry["dim"]["YEAR"]) **in** validate[entry["dim"]["COUNTRY"]][entry["dim"]["SEX"]]:
15. validate[entry["dim"]["COUNTRY"]][entry["dim"]["SEX"]].remove(int(entry["dim"]["YEAR"]))
16. validate[entry["dim"]["COUNTRY"]][entry["dim"]["SEX"]][**-**1] **=** validate[entry["dim"]["COUNTRY"]][entry["dim"]["SEX"]][**-**1] **+**1
17. **else**: is\_validated **=** False
18. **break**
19. **else**: is\_validated **=** False
20. **break**
21. **else**: is\_validated **=** False, break

**Transformation** **& ETC**– After validation, the data will be transformed into a format fitting for export. The data will be put into data frame via pandas where all necessary fields are filled. Once we get the data formatted, we will put the data into processed folder where it will await transfer to storage. Then, if the transfer to storage is successful, it will additionally be added to the archived folder. If the transfer fails, it will add to the error folder where it will await debugging, and potentially be put back into processed to be retried.

General Pipeline Diagram

A picture containing diagram

Description automatically generated