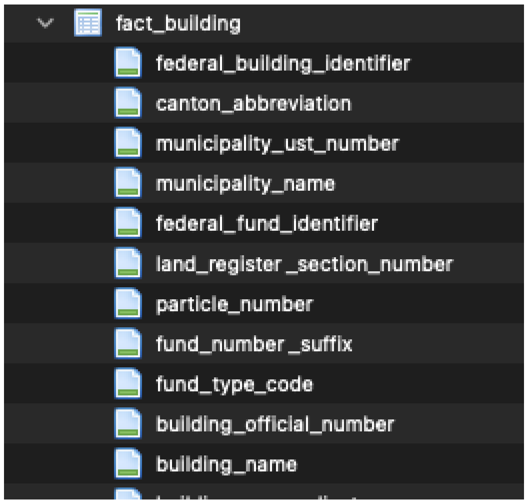
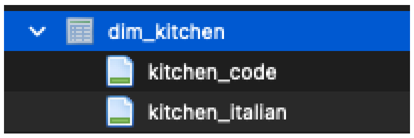
Steps undertaken:

1) First I understood the dataset and translated all the field names into English, for this I extracted from the specifications pdfs a table (function can be seen in utils.py) and used google translate on each of the fields (results can be seen in https://docs.google.com/spreadsheets/d/19OBC0l4yGghavFbIS2u4bOK7GN8hBgR1MxK5MOkhCG0/edit?usp=sharing) . The result of this exercise was a mapping of the codes that can be seen in config.py.
2) Then I created the scripts to download the file and unzip it.
3) I decided to stay in sqlite as a database for simplicity, but any other database of choice could be selected.
4) Then I defined the desired data model, I opted for a star schema with three fact tables: fact_building, fact_dwelling and fact_entrance and 27 dimensional tables for example dim_builiding_category and dim_building_class, the idea here is to easily be able to get the attribute values for the fact tables joining with the dimensions using the attributes codes. This is a preview of the fact_building table and dim_kitchen fields.

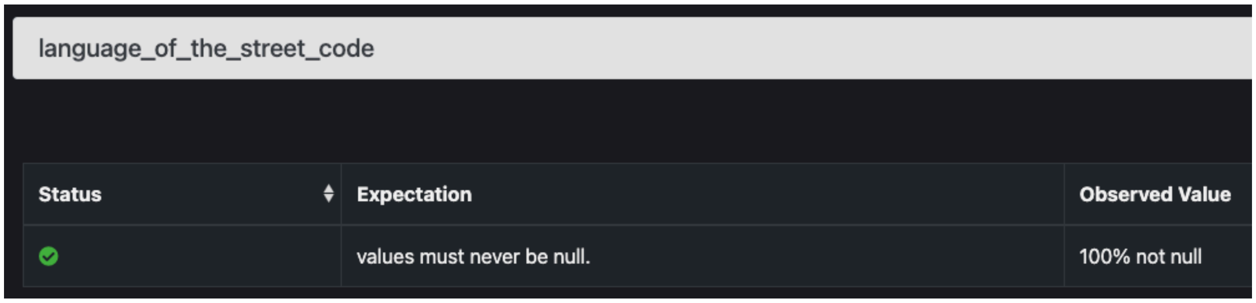


fact_building
- federal_building_identifier
- canton_abbreviation
- municipality_ust_number
- municipality_name
- federal_fund_identifier
- land_register _section_number
- particle_number
- fund_number _suffix
- fund_type_code
- building_official_number
- building_name



dim_kitchen
- kitchen_code
- kitchen_italian

5) For building the fact tables, I created separate sql files which can be seen in the sql folder to map the name of the fields to their english counterpart, also to cast the adequate data types and verify for NULL values. For the dimensional tables I automated the process iterating over the different dimensions obtaining the attribute codes and names from the codes table.
6) For running the tests, I used the Great expectations library to test for referential integrity issues, where I tested that the codes present in the fact tables have a counterpart in the dimension tables, the pipeline will fail if any of the tests fails. This is an example of one of the tests ran, we can see from Great expectations UI that the the language_of_the_street_code has a counterpart in the dimensional table in a 100% of the cases.


language_of_the_street_code

Status	Expectation	Observed Value
✓	values must never be null.	100% not null

7) Regarding error handling, it is intentional to not catch any unknown errors, when there is an error the pipeline should fail and notify me, error handling will be done in that case as needed.
8) Finally I collected all the different functions into a pipeline function that can be ran in a schedule to update the data every day.
9) To be able to run the pipeline a Python virtual environment must be created, then dependencies on requirements.txt installed and then the file main.py ran.