**Workflow Summary**

**Setup and Configuration**

**1. Import Libraries**

The script begins by importing necessary libraries:

1. **os**, **re**, **json**: File handling, regular expressions, and JSON operations.
2. **requests**, **BytesIO**: For API requests and handling in-memory file-like objects.
3. **datetime**, **pytz**: For handling timestamps with time zones.
4. **logging**: To log information and errors.
5. **pandas**: For data manipulation.
6. **ThreadPoolExecutor**: For parallel processing of datasets.

### ****2. Configure Logging****

The logging module is configured to output messages in the format:

<timestamp> - <log level> - <message>

**3. Constants Setup**

* **LOCAL\_PROCESSED\_DIR**: Directory where processed CSV files are saved (./processed\_data/).

**4. Create a Persistent HTTP Session**

* A requests.Session object is created to optimize API calls by reusing TCP connections.

### ****5. Define Helper Functions****

#### **5.1** to\_snake\_case()

* Converts column names into snake\_case by:
  + Lowercasing the text.
  + Removing non-alphanumeric characters.
  + Replacing spaces with underscores.

#### **5.2** load\_metadata()

* Loads the metadata from metadata.json to track the datasets processed during the last run.
* If the file doesn't exist, it returns an empty dictionary.

#### **5.3** save\_metadata()

* Saves updated metadata (including the timestamp of the current run) back into metadata.json.

#### **5.4** process\_dataset()

* Processes a single dataset:
  1. Fetches the CSV file from the dataset's downloadURL.
  2. Reads the content into a pandas DataFrame.
  3. Converts column names to snake\_case.
  4. Saves the cleaned data to the LOCAL\_PROCESSED\_DIR directory.
  5. Handles errors and logs them if the process fails.

### ****6. Main Function Execution****

#### **6.1 Create Processed Data Directory**

* Ensures the ./processed\_data/ directory exists. If not, it creates one.

#### **6.2 Fetch Dataset Metadata**

* Sends an API request to retrieve the list of datasets from the CMS API.
* Filters datasets to include only those related to "Hospitals".

#### **6.3 Load Metadata**

* Reads metadata.json to determine the last\_run\_time—the timestamp of the last successful execution.

#### **6.4 Filter New or Updated Datasets**

* Filters datasets that have been modified after the last\_run\_time.

#### **6.5 Parallel Dataset Processing**

* Uses ThreadPoolExecutor to process datasets in parallel (up to 4 threads).
* Each dataset is passed to the process\_dataset() function.

#### **6.6 Save Metadata**

* Updates metadata.json with the current timestamp to mark this execution as complete.

### ****7. Execution****

* The script is run under the if \_\_name\_\_ == "\_\_main\_\_": block, which ensures the main function only executes when the script is directly run (not imported).

### ****Execution Flow****

1. The script fetches all hospital datasets from the CMS API.
2. It compares each dataset's modified timestamp with the last run timestamp from metadata.json.
3. Only new or updated datasets are processed:
   * Download the dataset from its URL.
   * Clean and format the data (snake\_case column names).
   * Save the processed data to the ./processed\_data/ directory.
4. Logs progress and errors at each step.
5. Updates metadata.json to reflect the successful run.

**Result**

* Processed hospital datasets are saved in the ./processed\_data/ directory as CSV files.
* The script ensures efficient execution using parallel processing and retry logic.
* Metadata is updated to avoid reprocessing unchanged datasets in future runs.