Overview

This project aims to streamline table construction in Databricks by providing ready-made components and example notebooks. The idea was to build a lot of pipelines quickly.

The functions folder implements the pipeline pieces. For example, functions.utility defines the job-type presets used by many settings files. The utilities folder has scripts that do useful things like rescue a table or inspect a checkpoint folder. There are other things like a sanity checker, a bad records checker, etc.

Settings files under layer\_01\_bronze and layer\_02\_silver (and layer\_03\_gold if it exists) define how each table is built. The pipeline merges individual table settings with sensible defaults for each job type, including streaming checkpoint locations and schema locations. Read the Simple Settings Reference for more about simple settings and job types.

.  
├─ layer\_01\_bronze/ # raw ingestion configs  
├─ layer\_02\_silver/ # cleaned & deduplicated configs  
├─ layer\_03\_gold/ # optional gold layer  
├─ functions/ # PySpark helper modules  
│ ├─ read.py, transform.py, write.py, history.py  
├─ utilities/ # notebooks and shell scripts  
├─ dashboards/ # data exploration notebooks  
├─ sandbox/ # ad‑hoc notebooks  
└─ \*.ipynb # workflow notebooks (00\_job\_settings …)

In databricks-utilities (a separate repo) there is a little utility called get-schema.py which determines some information by inspecting a datafile. This can be copy/pasted directly into a table’s job settings.

In the Documentation repo, there are procedures for doing the most common tasks like building a table pipeline, resetting the checkpoint folder, or rescuing a silver table. There are also reference documents and an overview (this file).

Workflow

The basic workflow is a few steps. When you are in the Workflow/Jobs UI, you should see one box for each step, i.e. 4 boxes.

* **Downloader**
  + The download and scraper scripts work together to get files. Marker files are placed into folders representing whether files have already been downloaded for that date.
* **Job Settings**
  + Reads all settings for all layers and distributes them as task values, then runs a sanity check. The settings are retrieved by later tasks.
* **Bronze**
  + Each task performs its ingest and then **after writing occurs** it checks for bad records and builds history tables. If bad records are found they are turned into a table and the pipeline is stopped before silver can run. Once either the data files or the code are corrected, there are two choices.
  + The first choice is: the table will need to be restored to the prior version from before the load took place, the checkpoint folder will need to be erased, and the json readStreamOptions will need to have the startingVersion setting applied. Then the pipeline can be safely restarted.
  + The second choice is that the bad records can be manually fixed in place and manually added to the table.
* **Silver**
  + Each task performs a data quality check in between the transfer and write stages of execution of the code. If data quality failures are found they are turned into a table the pipeline is stopped **before writing occurs**.
  + Once the dqx settings are corrected or the problem is found, the pipeline can be safely restarted since nothing was written to silver.

This is how the workflow looks in Databricks.

