Delta Pipeline Framework Overview

This document describes a generalized Delta Lake pipeline framework suitable for ingesting, transforming, and writing structured data. The design separates the pipeline into bronze, silver, and gold layers, with each table's logic defined in JSON configuration files. Python functions handle reusable logic, and Databricks notebooks orchestrate the workflow.

.  
├─ layer\_01\_bronze/ # raw ingestion configs  
├─ layer\_02\_silver/ # cleaned & deduplicated configs  
├─ layer\_03\_gold/ # final tables  
├─ functions/ # PySpark helper modules  
│ ├─ read.py, transform.py, write.py, history.py  
│ ├─ internal/ # sanity checks & rescue helpers  
│ └─ catch\_up/ # experimental catch‑up code  
├─ utilities/ # notebooks and shell scripts  
├─ dashboards/ # data exploration notebooks  
├─ sandbox/ # ad‑hoc notebooks  
└─ \*.ipynb # workflow notebooks (00\_job\_settings …)

Configuration Files

Each dataset's settings live in JSON files under each layer directory. These configurations map directly to the pipeline's Python function modules. An example configuration illustrates how to define read, transform, and write operations for any given table:

{  
 "read\_function": "functions.stream\_read\_table",  
 "transform\_function": "functions.silver\_scd2\_transform",  
 "write\_function": "functions.stream\_upsert\_table",  
 "upsert\_function": "functions.microbatch\_scd2\_upsert",  
 "src\_table\_name": "source\_layer.table\_name",  
 "dst\_table\_name": "target\_layer.table\_name",  
 "business\_key": ["id", "key", "systemId"],  
 "surrogate\_key": ["attribute", …],  
 "use\_row\_hash": true,  
 "row\_hash\_col": "row\_hash",  
 "writeStreamOptions": {  
 "mergeSchema": "false",  
 "checkpointLocation": "/Volumes/framework/silver/checkpoints/table\_name",  
 "delta.columnMapping.mode": "name"  
 },  
 "ingest\_time\_column": "derived\_ingest\_time"  
}

Core Functions

All reusable ETL functions are registered in functions/\_\_init\_\_.py, enabling notebooks to dynamically reference them by string path.

Key operations include:

Layered Transformation – cleans and standardizes columns, attaches metadata, and generates timestamps.

Microbatch SCD2 Upsert – enables dimension management and historical tracking in the silver layer.

History Reconstruction – reads Delta logs and replays file-level operations for traceability.

Downloader Utility – fetches source files and arranges them into staging directories.

The internal module also includes validation tools to rebuild \_\_init\_\_.py and check JSON configurations.

Notebook Workflow

The notebook suite implements the full orchestration of the framework:

00\_job\_settings – gathers table configurations.

01\_rebuild\_init\_files – rebuilds function entry points.

02\_sanity\_check – performs config validation.

03\_ingest – runs ingestion jobs.

06\_bad\_records / 06\_history – runs diagnostics and history checks.

Additional notebooks manage checkpoints, table recovery, and exploratory data validation.

This modular framework allows scalable ingestion and transformation pipelines by externalizing logic into configurations and enabling reuse of tested ETL functions.