Mini ETL Framework - One-Page App Summary

What it is

- A lightweight Python ETL app that ingests JSON/CSV files, maps source schemas to one canonical model, validates records, and writes JSONL outputs.
- The pipeline is modular and CLI-driven so local runs are reproducible and observable.

Who it’s for

- Primary persona: Not found in repo.
- Likely audience from README context: data engineers or learners practicing modular ETL design.

What it does

- Auto-detects input format (‘--format auto‘) or enforces JSON/CSV mode.
- Loads schema definitions from ‘schemas/*.json‘ and resolves schema by filename prefix or fixed schema.
- Maps source fields to canonical keys using schema ‘field_map‘ definitions.
- Normalizes signup dates and computes ‘total_order_value‘ plus ‘order_count‘.
- Validates canonical records and quarantines failures with detailed errors.
- Streams clean and bad outputs to separate JSONL files.
- Optionally writes a run report with timing, config, and file/record stats.

How it works (repo-evidence architecture)

- ‘main.py‘ orchestrates args, file discovery, schema selection, transform, validation, and output/report writes.
- ‘reader.py‘ reads JSON arrays and CSV rows into dict records.
- ‘schema_registry.py‘ loads ‘SchemaDef‘ metadata and supports nested-path lookups.
- ‘transformer_schema.py‘ applies field mapping and aggregate calculations into canonical output.
- ‘validator.py‘ defines reusable validators; ‘main.py‘ runs canonical checks in the live flow.
- Data flow: ‘data/raw/*‘ -> reader -> schema map/transform -> canonical validation -> ‘data/out/cleaned.jsonl‘ and ‘data/bad/bad_records.jsonl‘.

How to run (minimal getting started)

- Prerequisite: Python 3.8+ (README states no external runtime dependencies).
- From repo root: ‘python main.py‘.
- Optional: ‘python main.py --format auto --schema-mode filename --report data/out/run_report.json‘.
- Defaults: input=‘data/raw‘, clean output=‘data/out/cleaned.jsonl‘, bad output=‘data/bad/bad_records.jsonl‘.