DATA ENGINEERING · MARKET-DATA QUALITY & PIPELINE

Market Data Health Monitor: *flag a bad feed* before a model uses it.

A data-quality monitor and the production data-engineering pipeline behind it. A PySpark ETL validates a market feed, models it into a star-schema lakehouse, and publishes per-feed data-quality SLAs with column-level lineage. An Airflow DAG orchestrates the run, with a Databricks and Azure variant for the cloud.

Author S. Ize-Iyamu **Audience** Data platform & quant data teams **Length** 3 pages **Status** Technical brief

Stack PySpark · Airflow · Databricks

The Problem

Trading and research run on clean data, so the first job is catching a feed that's wrong before it reaches a model. The expensive failures are usually hard to spot at first: a calendar gap, a stale value held flat for days, a duplicated timestamp, or a price that should never have printed. By the time a downstream model misbehaves, the bad input is hard to trace back. The discipline is to test every feed at ingestion and tell genuine faults apart from normal market moves, so the verdict it publishes is one a consumer can trust.

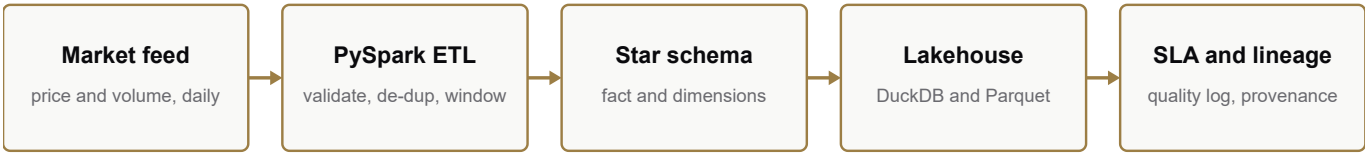
This project covers both halves of that job. For a researcher, an interactive **monitor** answers whether a feed is healthy right now. The batch **pipeline** behind it goes further, validating, modeling, warehousing, and logging the feed so a downstream team knows what's safe to publish. That second layer is what a data platform actually runs.

ENGINE	WAREHOUSE	ORCHESTRATION	CLOUD VARIANT	LINEAGE
PySpark	**Star schema**	**Airflow**	**Databricks**	**Column-level**

Architecture

The pipeline ingests a daily price-and-volume feed, either synthetic with injected faults or a real CSV. It validates and de-duplicates the rows, computes returns and rolling anomaly scores with ordered window functions, then models the result into a dimensional star schema and loads a DuckDB and Parquet lakehouse. Each run also writes a per-feed data-quality and SLA log. An Airflow DAG schedules and gates the run, and a Databricks variant runs the same logic to Delta on Azure Data Lake.

FIGURE 1 · PIPELINE ARCHITECTURE



orchestrated by an Apache Airflow DAG · cloud variant: Databricks and Delta on Azure Data Lake

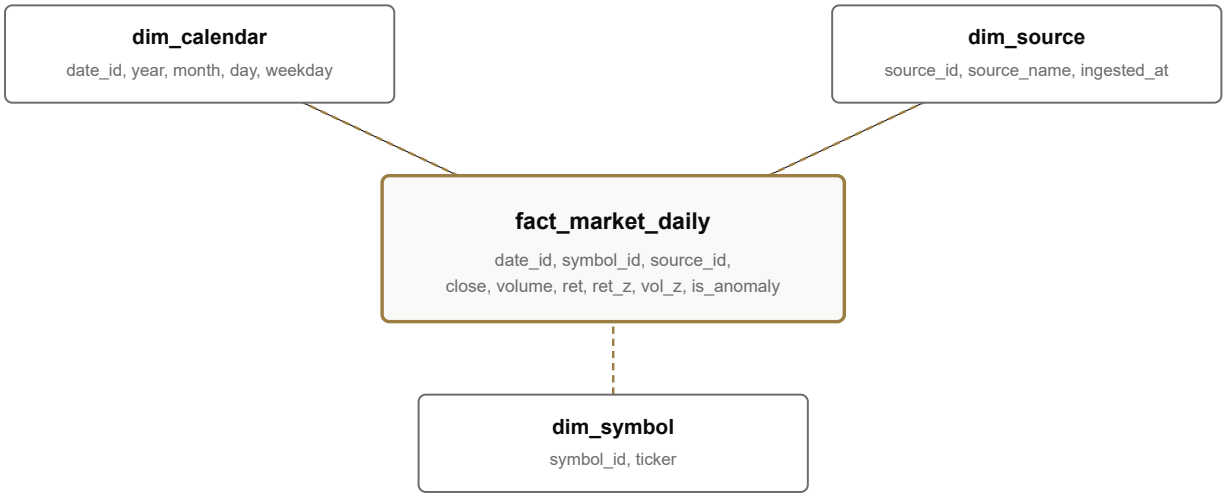
From feed to a trusted, modeled, and documented dataset; orchestrated by Airflow, with a Databricks/ADLS cloud variant.

> THE POINT
> The monitor shows health at a glance. The pipeline does the heavier job, making the data safe to publish and carrying the SLA verdict and lineage a platform owes its consumers. Keeping those two layers separate is the design.

The Dimensional Model

fact_market_daily is one row per symbol per trading day at a documented grain, with surrogate keys to three dimensions. A Kimball star keeps the fact narrow and additive, so it answers analytical questions with simple joins, like a monthly average close or an SLA scorecard, instead of ad-hoc queries against raw data. Returns, rolling means and standard deviations, and z-scores are computed in PySpark with ordered window functions before the model is built.

FIGURE 2 · STAR SCHEMA



A central fact keyed by surrogate keys to dim_symbol, dim_calendar, and dim_source; the standard dimensional trade-off (Kimball & Ross).

Data-Quality Checks & SLAs

Each feed is tested on six checks that map to the DAMA data-quality dimensions, then assigned an SLA verdict (freshness within bound and zero invalid prices). Anomaly flags use rolling z-score control limits on returns and volume, a Shewhart control chart at the 3-sigma limit. The result is one row per feed per run in **data_quality_log**, the availability signal a platform publishes to consumers.

CHECK	DAMA DIMENSION	WHAT IT CATCHES
Calendar coverage	Completeness	missing business days, longest gap
Duplicate timestamps	Uniqueness	rows duplicated on (ticker, date)
Valid prices	Validity	non-positive or null close
Stale values	Validity	repeated close (zero return)
Freshness	Timeliness	age of last point vs SLA
Anomaly (z-score)	Accuracy	return / volume beyond 3 sigma

A sample scorecard (illustrative) shows the SLA gate separating publishable feeds from those to quarantine:

FEED	ROWS	INVALID	DUPES	ANOMALIES	FRESHNESS	SLA
NOVA	178	0	0	1	0d	PASS
ORBIT	178	1	1	3	0d	FAIL
VERTEX	175	0	0	1	1d	PASS

Lineage & Metadata

The pipeline publishes **source-to-target column lineage** (a *data_lineage* table and a *lineage.json*) tracing every warehouse column back to its source column and transform. A per-run metadata record captures the run id, engine, source, sink, table row counts, and SLA pass count. This is the provenance that lets a consumer trace and audit any field, the metadata-management layer an enterprise data platform is expected to expose.

TARGET COLUMN	SOURCE	TRANSFORM
`fact_market_daily.ret`	feed.close	close / lag(close) - 1 [window: ticker, date]
`fact_market_daily.ret_z`	ret	(ret - avg(ret,20)) / stddev(ret,20)
`fact_market_daily.is_anomaly`	ret_z, vol_z	abs(z) > 3 [Shewhart 3-sigma]
`dim_symbol.symbol_id`	feed.ticker	row_number() surrogate key

Orchestration & Cloud

An **Apache Airflow** DAG runs the batch as *run_etl → data_quality_gate → publish_run_report*. It schedules the job on business days and enforces the SLA as a gate that can fail or quarantine a feed, then publishes the report and lineage. That turns a one-off script into a scheduled pipeline a team can observe and gate. A **Databricks** variant runs the same validation and star-schema logic on a cluster, writing **Delta** tables to **Azure Data Lake (ADLS Gen2)** through Unity Catalog. The local job's sink is configurable to a DBFS or ADLS mount, so the same code writes its lakehouse copy to the cloud.

> **SO WHAT**
> End to end, this is what a production data feed looks like: ingest, validate, model, warehouse, document lineage, gate on quality, schedule, and deploy to the cloud. It's built from first principles and runs locally for free.

Method & Sources

The data-correctness checks map to the **DAMA** data-quality dimensions (completeness, uniqueness, timeliness, validity, accuracy, consistency). Anomaly detection uses a rolling **Shewhart** control chart at the 3-sigma limit, and the warehouse follows a **Kimball** dimensional star. The engine runs PySpark in local mode with DuckDB and Parquet; the Databricks, ADLS, and Airflow layers are the cloud and orchestration forms of the same logic. Live app at data-health-monitor.streamlit.app.

References: DAMA Netherlands, *Dimensions of Data Quality (DDQ)* (2020); Shewhart, W. A., *Economic Control of Quality of Manufactured Product* (1931), three-sigma control limits; Kimball & Ross, *The Data Warehouse Toolkit* (3rd ed., 2013), dimensional modeling; Apache Airflow and Apache Spark project documentation (2025).