Banana Prices Pipeline — High-Level System Design

Summary

This project creates a streamlined data pipeline that captures banana price data from a UK government website. The system preserves a raw copy of the data including its lineage—tracking exactly where and when it was collected. It then cleans and standardizes the information before aggregating it by year and country of origin to support reporting and dashboards.

Following the **Medallion architecture**, the data flows through three distinct stages:

- **Bronze**: The raw, original data.
- **Silver**: The cleaned and standardized data.
- **Gold**: The final aggregated data.

The pipeline is versatile and can be executed on **Databricks** in the cloud or locally on a laptop. This document provides a clear overview of the design to help readers quickly understand the project's structure and purpose.

Key Terms

Term	Meaning
DEFRA	The UK government department that publishes the Banana Prices dataset on gov.uk.
Medallion Architecture	A data organization pattern with three layers: Bronze (raw), Silver (cleaned), and Gold (aggregated).
Bronze	The raw data layer where information is stored exactly as received, along with lineage metadata.
Silver	The cleaned data layer where column names and types are standardized, dates are parsed, and duplicates are removed.
Gold	The business layer containing aggregated data used specifically for reporting and dashboards.
Lineage	Metadata that tracks where the data came from (URL), the specific Run ID, and the time it was collected.
Delta Lake	A storage format used on Databricks that supports data updates and "time travel" (version history).
Parquet	A file format used for efficient data storage when running the pipeline locally.
ADLS Gen2	Azure Data Lake Storage Gen2, which is the primary storage used with Databricks.
DBFS	Databricks File System, used to reference file locations within the workspace.
Orchestrator	The component (like a Databricks Job or Azure Data Factory) that automatically triggers the pipeline on a schedule.

Overview

This pipeline automates the ingestion of UK government banana price data from DEFRA (gov.uk) and organises it using a Medallion Architecture (Bronze, Silver, and Gold). It is designed for Azure Databricks with Delta Lake and ADLS Gen2, while also supporting local execution using Parquet files.

Purpose

To ingest the DEFRA Banana Prices CSV, capture full lineage (where the data came from and when it was collected), clean and standardise the dataset, and generate year- and origin-level summaries for business reporting.

Scope

The workflow covers the full data lifecycle: gov.uk CSV → Bronze (raw data + lineage) → Silver (cleaned and structured) → Gold (aggregated outputs).

Users

The pipeline supports data analysts, BI tools, and automated dashboards, and can also feed downstream APIs that need reliable, curated data.

Why Bronze, Silver, Gold?

- **Bronze:** Stores the source data exactly as received to support auditing and reprocessing.
- **Silver:** Applies one consistent cleaning and formatting step to create a standard, dependable dataset.
- **Gold:** Produces business-ready aggregates so reporting is faster, simpler, and more accurate.

High-Level Architecture

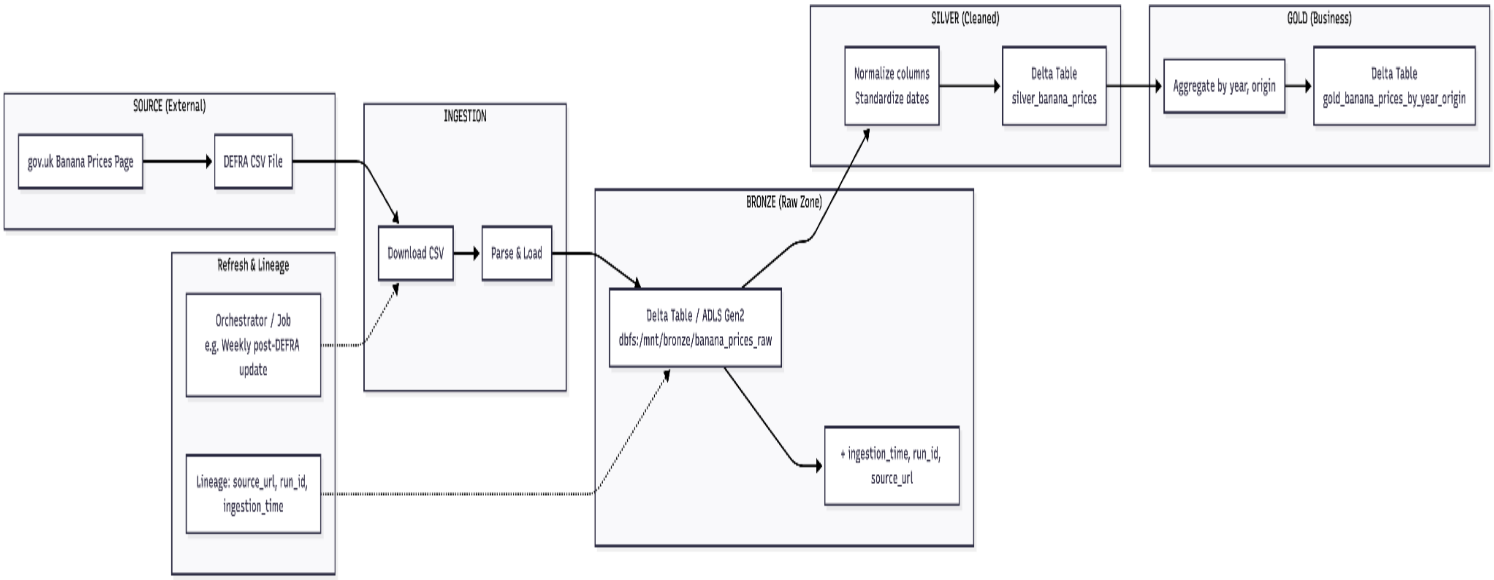
SOURCE (External)
gov.uk Banana Prices Page
DEFRA CSV File
INGESTION
Download CSV
Parse & Load
Refresh & Lineage
Orchestrator / Job e.g. Weekly post-DEFRA update
Lineage: source_url, run_id, ingestion_time
BRONZE (Raw Zone)
Delta Table / ADLS Gen2 dbfs:/mnt/bronze/banana_prices_raw
+ ingestion_time, run_id, source_url
SILVER (Cleaned)
Normalize columns Standardize dates
Delta Table silver_banana_prices
GOLD (Business)
Aggregate by year, origin
Delta Table gold_banana_prices_by_year_origin

Figure: End-to-end Banana Prices data pipeline using the Medallion architecture (Source → Bronze → Silver → Gold), with orchestration and lineage.

External Source

The dataset comes from DEFRA’s banana prices page on gov.uk. Because there isn’t an official API, the pipeline finds the correct CSV by parsing the page HTML and identifying the most recent download link.

Pipeline Execution

The workflow runs on a scheduler such as Databricks Jobs, Azure Data Factory, or a cron job. When triggered, it performs the following steps:

- **URL Discovery:** Reads the landing page and selects the latest CSV link.
- **Data Retrieval:** Downloads the raw file from the source.
- **Validation:** Loads the file into a DataFrame to confirm the schema and check row counts.
- **Bronze Ingestion:** Adds lineage metadata and stores the raw data in the Bronze layer.
- **Silver Refinement:** Cleans and standardises formats, removes duplicates, and writes the refined dataset to Silver.
- **Gold Aggregation:** Aggregates the Silver data by year and country of origin, then publishes the final dataset to Gold.

Storage Strategy

The pipeline stores each layer in separate paths within ADLS Gen2 or DBFS. Delta Lake is used for Databricks-based runs, while Parquet is used for local execution to keep the solution portable.

Data Consumption

Most users consume the curated **Gold** layer through BI tools, dashboards, ad-hoc SQL queries, or downstream APIs.

Logical Flow Summary

gov.uk (CSV) → Orchestrator → Ingestion + Processing → Bronze → Silver → Gold → Analytics / API

Logical Pipeline Diagram (Image Using Mermaid)

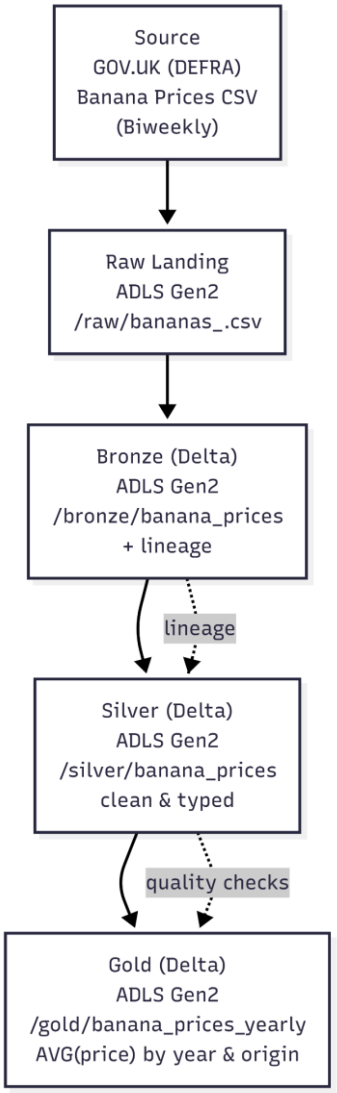

Figure 2 shows the logical flow of the data pipeline created using Mermaid diagrams. This diagram focuses on clearly representing the movement of data across the Source, Raw Landing, Bronze, Silver, and Gold layers, along with data lineage and quality checks. Mermaid was chosen to keep the architecture precise, readable, and easy to reproduce.

Visual Pipeline Representation (AI-generated)

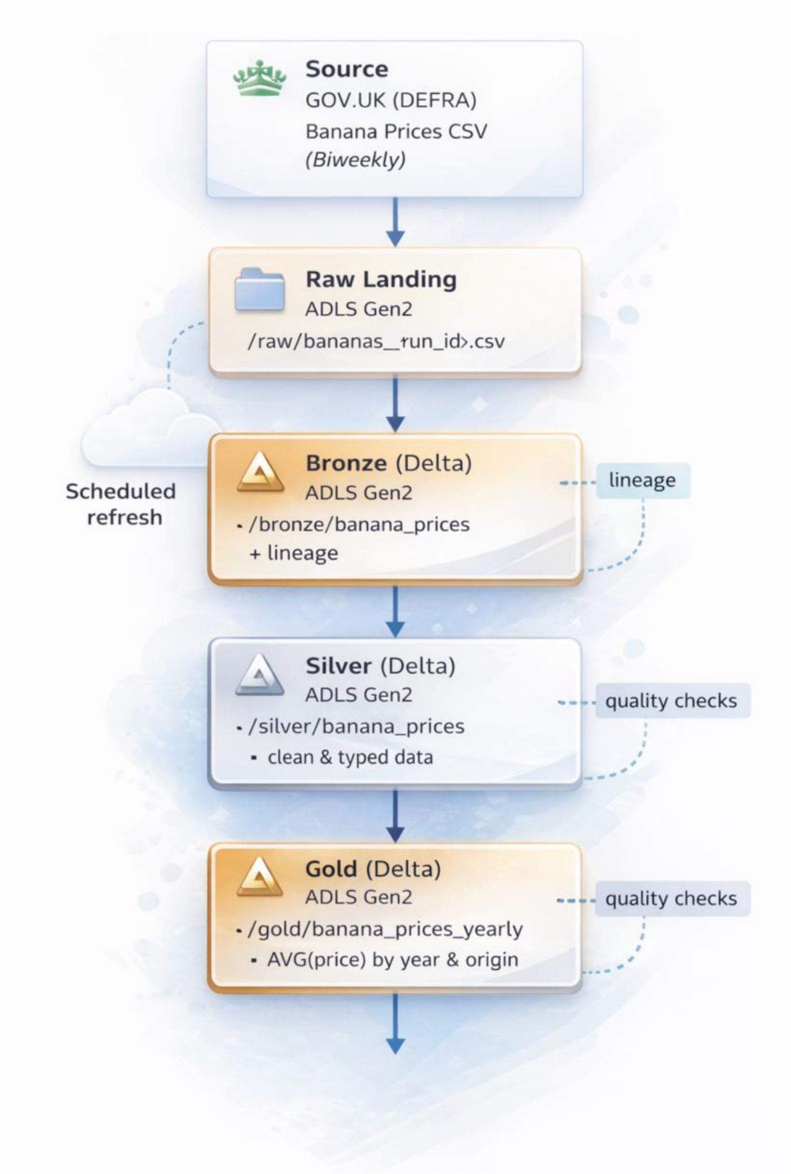

Figure 3 provides a more visually rich representation of the same pipeline. This image was generated using AI to enhance visual appeal and make the pipeline easier to understand for non-technical stakeholders. The underlying architecture remains identical to the logical design shown in Figure 2.

Components

Component	Role
Orchestrator	Manages the automated execution of the pipeline based on a set schedule (e.g., weekly or bi-weekly).
Ingestion (Source → Bronze)	Dynamically locates the CSV URL, downloads the file, verifies the schema and row counts, attaches lineage metadata, and commits it to the Bronze layer.
Bronze Layer	Serves as the immutable raw storage, housing the original dataset alongside essential lineage columns like ingestion_time, run_id, and source_url.
Silver Layer	Acts as the refined data zone where information is standardized: column names are normalized, dates are parsed, prices are converted to numeric types, and duplicate records are removed.
Gold Layer	The high-value business layer containing pre-calculated metrics, such as average prices and row counts, grouped by year and country of origin.
Storage	The physical data repository (ADLS Gen2 or DBFS), with connection paths dynamically managed through environment variables or configuration files.

Data Model

Layer	Key Columns	Description
Source (CSV)	Origin, Date, Price, Units	The original, unaltered data published by DEFRA.
Bronze	Origin, Date, Price, Units, ingestion_time, run_id, source_url	Raw data enriched with metadata for full auditability and lineage.
Silver	origin, price_date, price, units	Standardized data with normalized column names, correct data types, and duplicates removed.
Gold	year, origin, avg_price, row_count	High-level aggregated metrics, providing a

Layer	Key Columns	Description
		single summary row for each year and origin.

Technology Stack

Area	Selection
Orchestration	Databricks Jobs, Azure Data Factory (ADF), or standard cron for scheduling.
Compute	Azure Databricks Spark clusters or a local Python environment.
Language	Python 3.9+
Libraries	pandas, requests (ingestion), tenacity (retries), pydantic-settings (config), pyarrow.
Spark / Delta	Utilized for high-performance processing on Databricks (optional for local runs).
Storage	ADLS Gen2 or DBFS; formats include Delta Lake (cloud) or Parquet (local).
Configuration	Managed via environment variables (BANANA_*) or Databricks Secrets for security.

Data Flow (Operational Sequence)

The pipeline follows a structured, end-to-end execution sequence:

1. **Trigger**: The Orchestrator initiates a scheduled run.
2. **Discovery**: The system scrapes the gov.uk landing page for the most recent data.
3. **Extraction**: The specific CSV link is identified and the file is downloaded.
4. **Ingestion**: Data is loaded into a Spark or Pandas DataFrame.
5. **Validation**: The system verifies the schema and ensures the row count is within expected limits.
6. **Bronze Entry**: Lineage metadata is attached, and the records are committed to the Bronze layer.
7. **Refinement**: Data is cleaned and deduplicated before being moved to the Silver layer.
8. **Aggregation**: Final metrics are summarized and written to the Gold layer.
9. **Completion**: The run finishes, and performance metrics are logged for monitoring.

Failure Handling & Resilience

Potential Failure	**Mitigation Strategy**
gov.uk unavailable	Implements HTTP retries with exponential backoff; triggers an alert upon final failure.
CSV link not found	Terminates the process immediately with a descriptive error log.
Schema or parse error	Triggers a validation failure to prevent corrupt data from entering the pipeline.
Unexpectedly low row count	Flags a data quality exception to ensure the source file is complete.
Storage write failure	Logs the specific I/O error and raises an exception to prevent partial data states.

Deployment Environments

The pipeline is designed to be environment-agnostic:

- **Databricks**: Executed via notebooks or scripts, with configuration managed through Databricks Secrets or environment variables.
- **Local**: Can be installed as a package (pip install -e .) and executed via the command line using python -m banana_pipeline --layers full.

Constraints and Scalability

- **Source**: Optimized for single CSV source retrieval in the absence of a formal API.
- **Volume**: Efficiently handles the current scale of ~13K rows using batch processing.
- **Traceability**: Maintains persistent lineage in the Bronze layer for audit purposes.

Project Deliverables

- **Full Medallion Pipeline**: A complete, multi-stage architecture (Source → Bronze → Silver → Gold).
- **Data Integrity**: Built-in validation and clear lineage tracking for every record.
- **Versatility**: A modular design that supports both **Delta Lake** on Databricks and **Parquet** for local development.