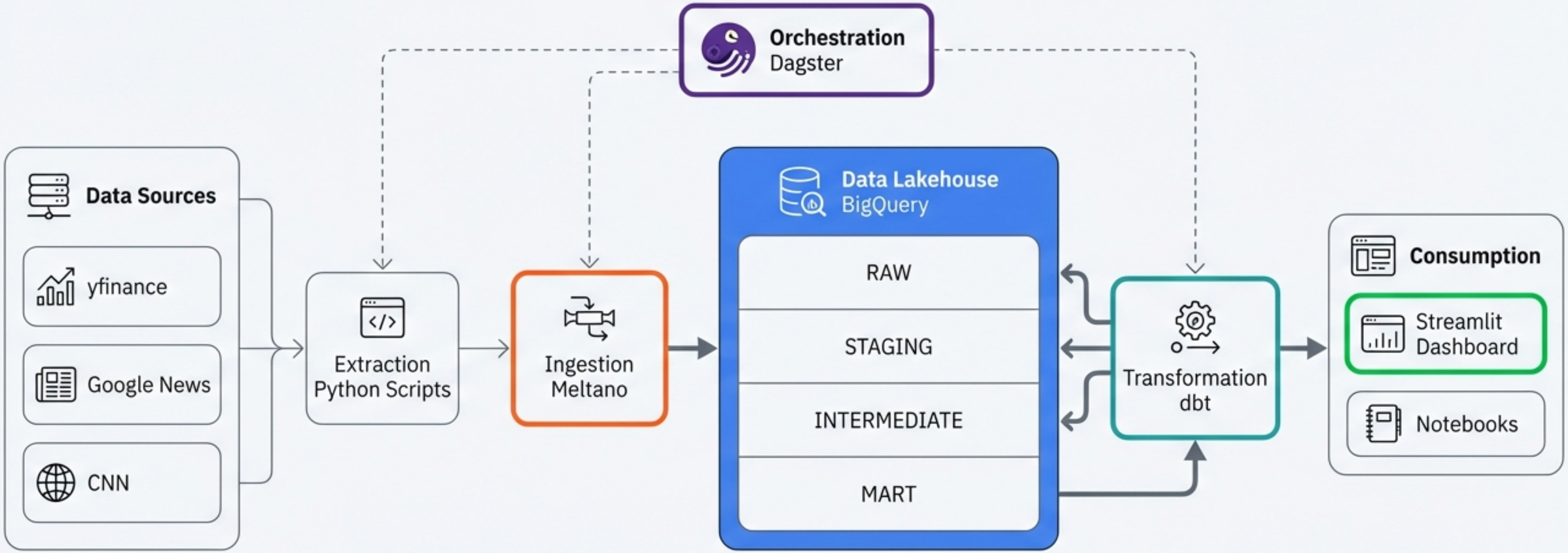
Market (Mag7) & Sentiment Intelligence Lakehouse

An end-to-end data platform for transforming raw market data into refined, actionable intelligence.

The Blueprint: From Raw Data to Refined Intelligence



This project implements a complete ELT pipeline to automate the daily extraction, ingestion, and modelling of stock and news data, culminating in an analytics-ready data warehouse and an interactive dashboard.

The Raw Materials: Sourcing Market and Sentiment Data

Market Prices & Volatility

Source: `yfinance` API

Data: Historical and daily stock prices for the Magnificent Seven, key market indexes (e.g., S&P 500), and the VIX volatility index.

News & Public Sentiment

Source: Google News RSS, GDELT Project (via BigQuery Public Data)

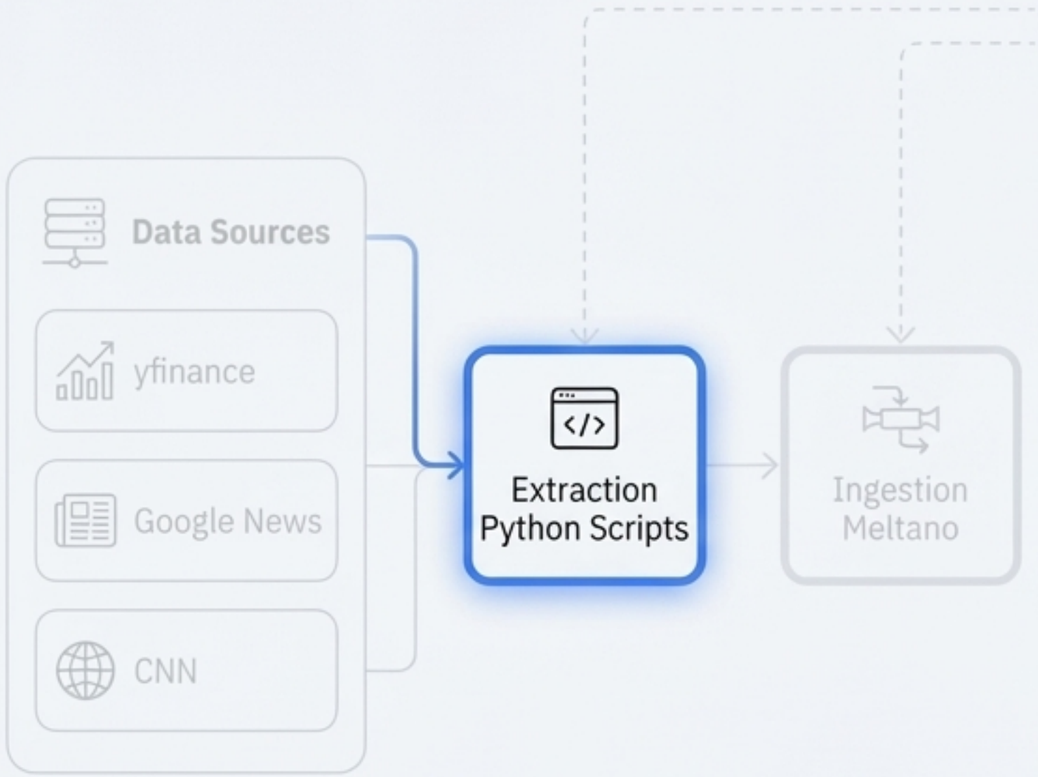
Data: Real-time news headlines related to specific tickers. Includes sentiment analysis performed by a FinBERT model during extraction.

Market Fear & Greed

Source: CNN Fear & Greed Index

Data: A daily metric for overall market sentiment.

Step 1: Intelligent Extraction with Custom Python Scripts



Our **Python extractors** are designed for both historical data loading and daily incremental updates.

stocks_extractor.py

Pulls data from `yfinance`.

```
# Historical Backfill
python src/extractors/stock_extractor.py --mode backfill --universe mag7_with_indexes

# Daily Incremental Load
python src/extractors/stocks_extractor.py --mode incremental --universe mag7_with_indexes
```

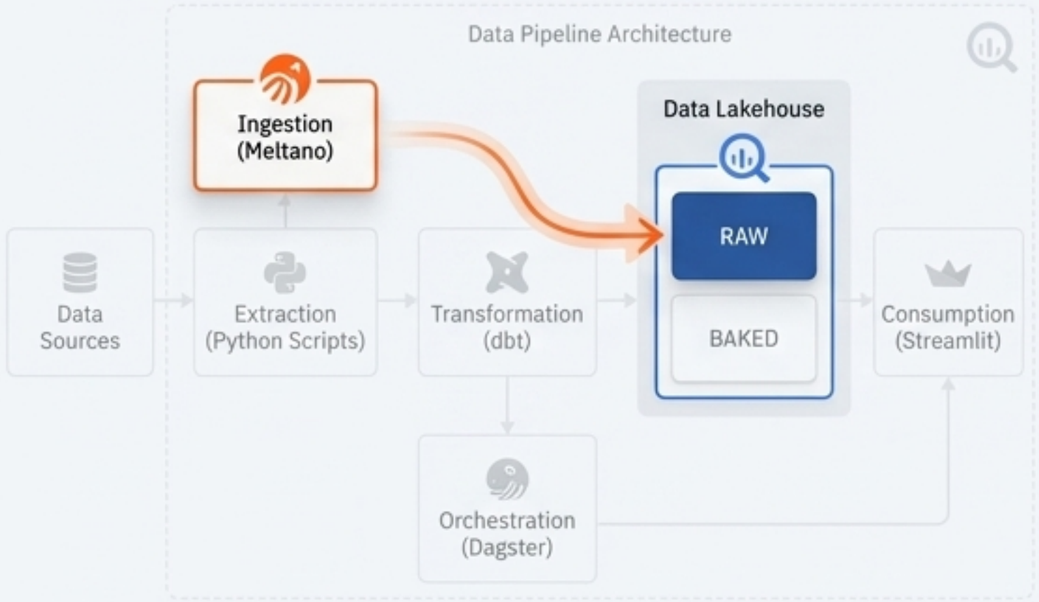
news_extractor.py

Fetches news headlines and applies FinBERT sentiment analysis.

```
# Recent News (last 24h)
python src/extractors/news_extractor.py --window 1d

# Targeted Search
python src/extractors/news_extractor.py --tickers AAPL MSFT --window 7d
```

Step 2: Reliable Ingestion using Meltano



Core Task

Meltano manages the EL (Extract-Load) part of the pipeline, using pre-built 'taps' and 'targets' to efficiently load the CSV files generated by our extractors into a raw BigQuery dataset.

Configuration as Code

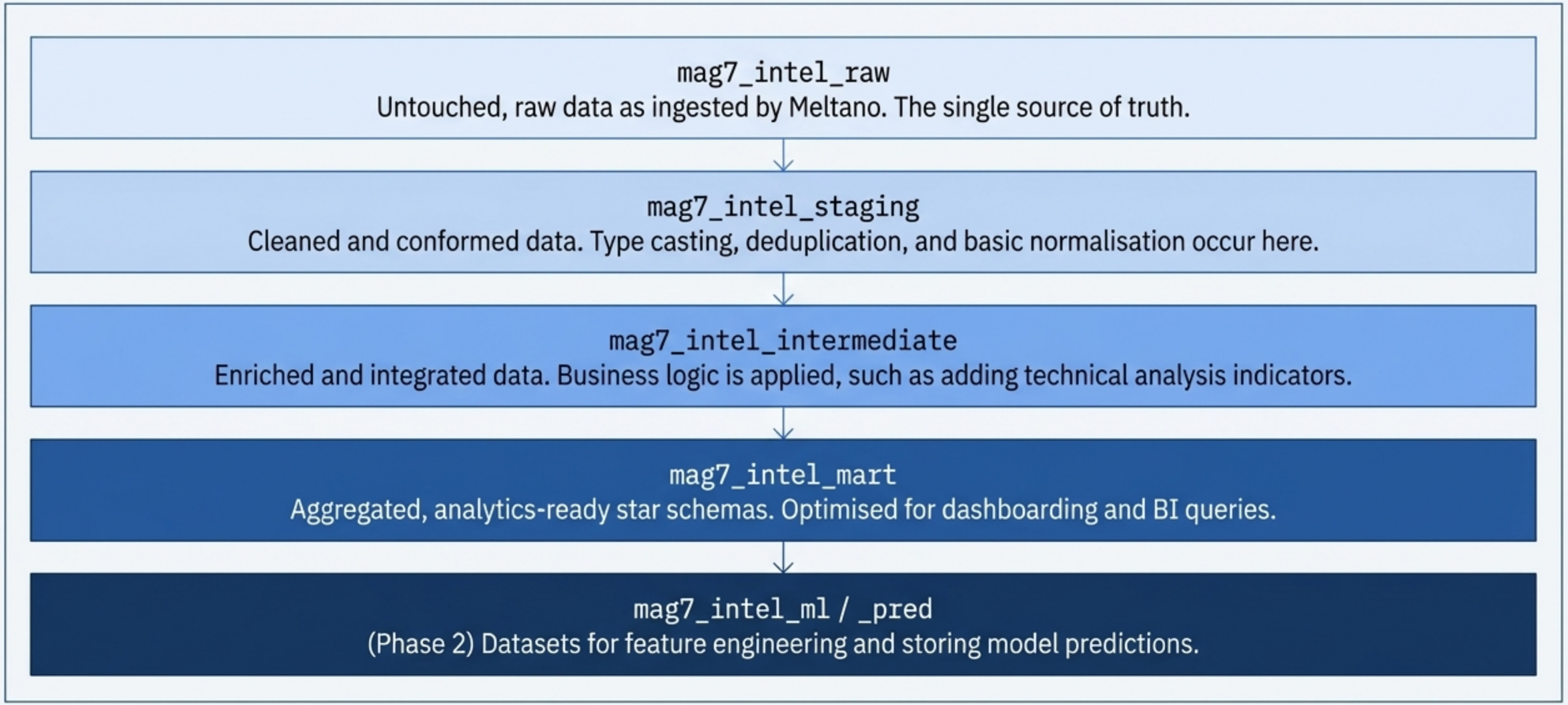
We define ingestion jobs declaratively in `meltano.yml` for reproducibility and version control.

meltano.yml - IBM Plex Sans Medium

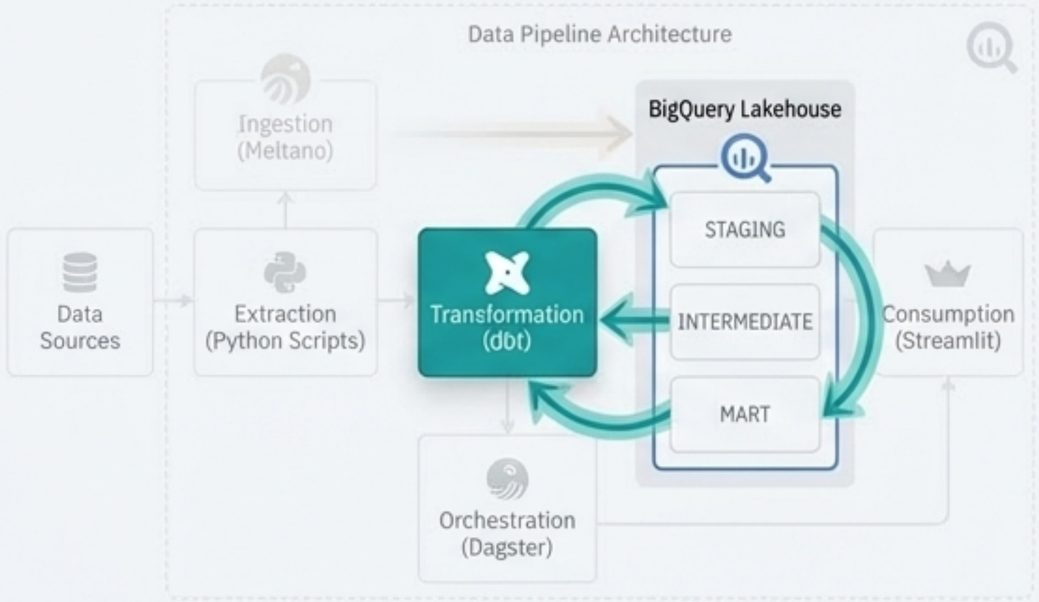
```yaml
jobs:
  - name: load_csvs
    tasks:
    - tap-csv
    - target-bigquery

# Loader config uses environment variables for security and flexibility
loaders:
  - name: target-bigquery
    variant: z3z1ma
    pip_url: git+https://github.com/z3z1ma/target-bigquery.git
    config:
      credentials_path: ${GOOGLE_APPLICATION_CREDENTIALS}
      dataset: ${BQ_DATASET_RAW}
      project: ${GCP_PROJECT_ID}
```

The Foundation: A Multi-Layered BigQuery Lakehouse



Step 3: Refining Data with dbt Transformations



Core Role

dbt orchestrates the ‘T’ in ELT directly within BigQuery. It transforms raw, ingested data into clean, tested, and analytics-ready datasets using SQL-based models.

Key dbt Activities

Modeling: Defining the structure of staging, intermediate, and mart tables.

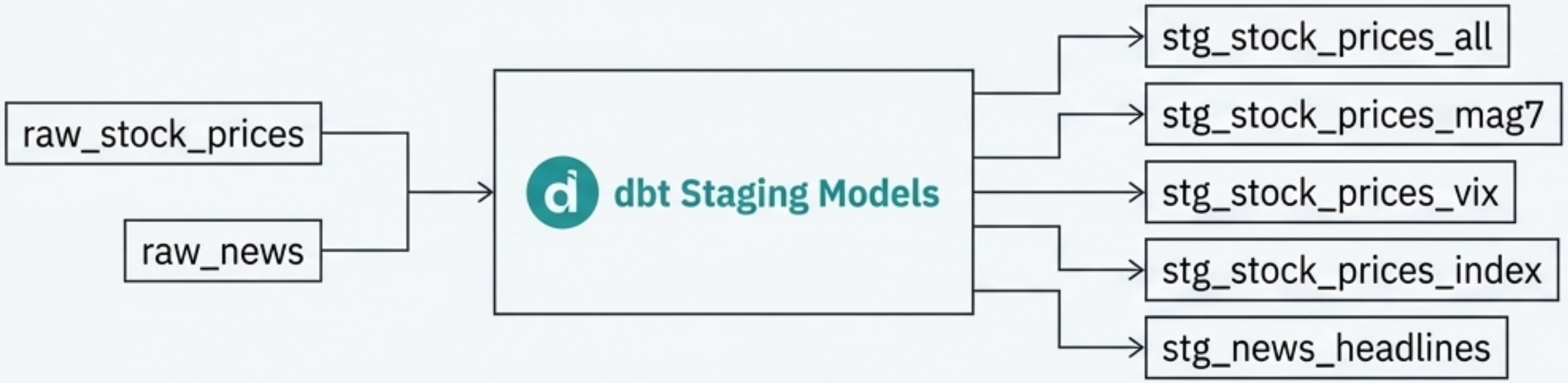
Testing: Implementing data quality checks to ensure accuracy and integrity.

Documentation: Generating a full data lineage graph and documentation.

“dbt allows us to build a robust, scalable, and well-documented transformation pipeline entirely in SQL.”

dbt in Action: The Staging Layer

Cleaning, conforming, and structuring the raw inputs.



Specific Staging Models & Tasks

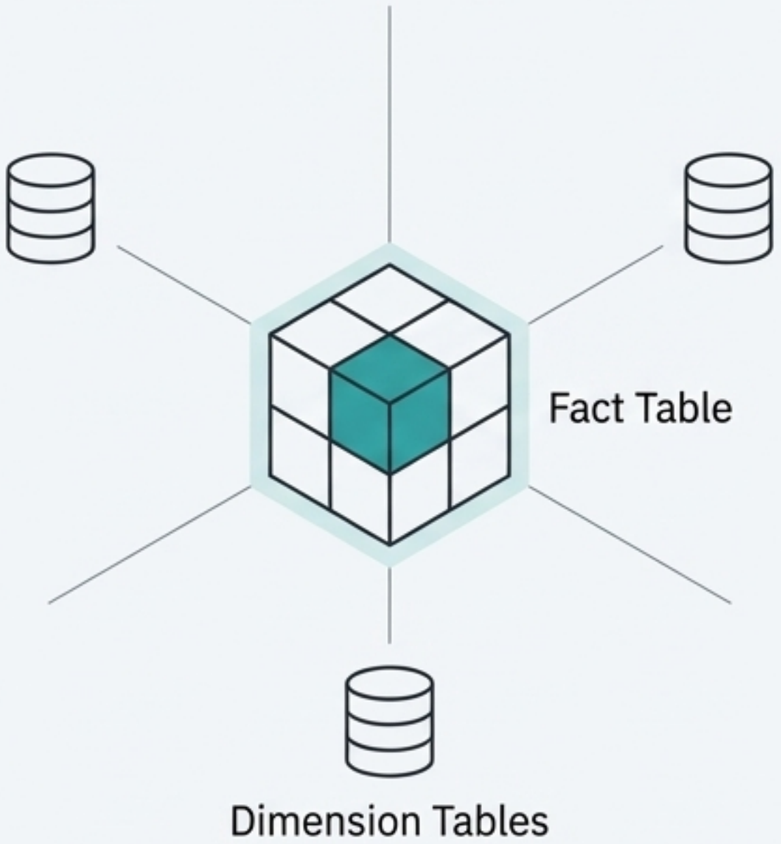
- `stg_stock_prices_all.sql`: Performs type casting, column renaming, and deduplication on all incoming price data.
- `stg_stock_prices_mag7.sql`: Filters for Magnificent Seven tickers.
- `stg_stock_prices_vix.sql`: Isolates VIX data for volatility analysis.
- `stg_stock_prices_index.sql`: Isolates market index data for benchmarking.
- `stg_news_headlines.sql`: Cleans news text and standardises ticker symbols.

Execution Command

```
dbt run --select staging
```

dbt in Action: The Mart Layer

Creating aggregated, analytics-ready datasets for consumption.



Key Analytics Tables

`mart_price_analytics`

- **Content:** Pre-calculated metrics like daily/weekly returns, historical volatility, and performance drawdowns.
- **Purpose:** Powers performance analysis and charting.

`mart_factor_signals`

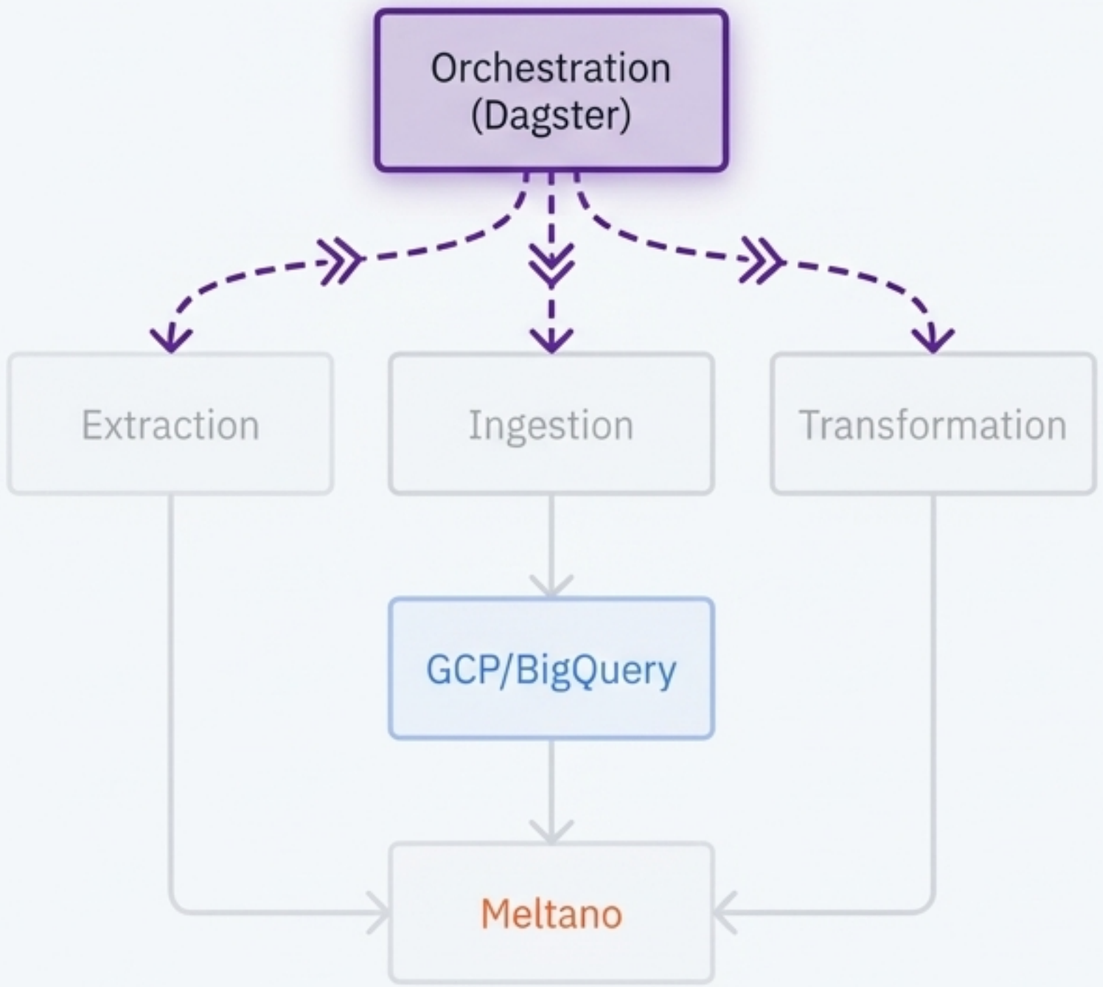
- **Content:** Simple quantitative signals and factor ranks (e.g., momentum).
- **Purpose:** Initial layer for quantitative research.

`mart_sentiment_daily`

- **Content:** Aggregates daily news and GDELT sentiment scores by ticker.
- **Purpose:** Provides a view of market sentiment over time.

The Conductor: Automating the Pipeline with Dagster

Automating and orchestrating the end-to-end data pipeline.



Core Function

Dagster serves as the master orchestrator, defining the entire ELT pipeline as a graph of data assets with clear dependencies. It triggers and monitors each step of the process.

Key Dagster Assets Defined in `assets.py`

1. **Extraction Assets**
   Python functions that trigger `stocks_extractor.py` and `news_extractor.py`, materialising CSV files.
2. **Meltano Ingestion Asset**
   A software-defined asset that executes the `meltano run load_csvs` job, dependent on the successful creation of the CSVs.
3. **dbt Transformation Assets**
   An asset that runs the `dbt build` command, dependent on the successful loading of raw data by Meltano.

Engineering for Collaboration: Project Setup & Workflow

Ensuring a consistent and secure development environment

Key Components

Isolated Environment: Uses `Conda` with Python 3.11 to manage dependencies listed in `requirements.txt`.

Containerisation: A `Dockerfile` and `docker-compose.yml` are provided to containerise the Dagster deployment for production-like environments.

Automated Environment Variables: A Conda `activate.d` hook is used to automatically load and unset secrets and configuration variables, preventing them from being hard-coded or leaked.

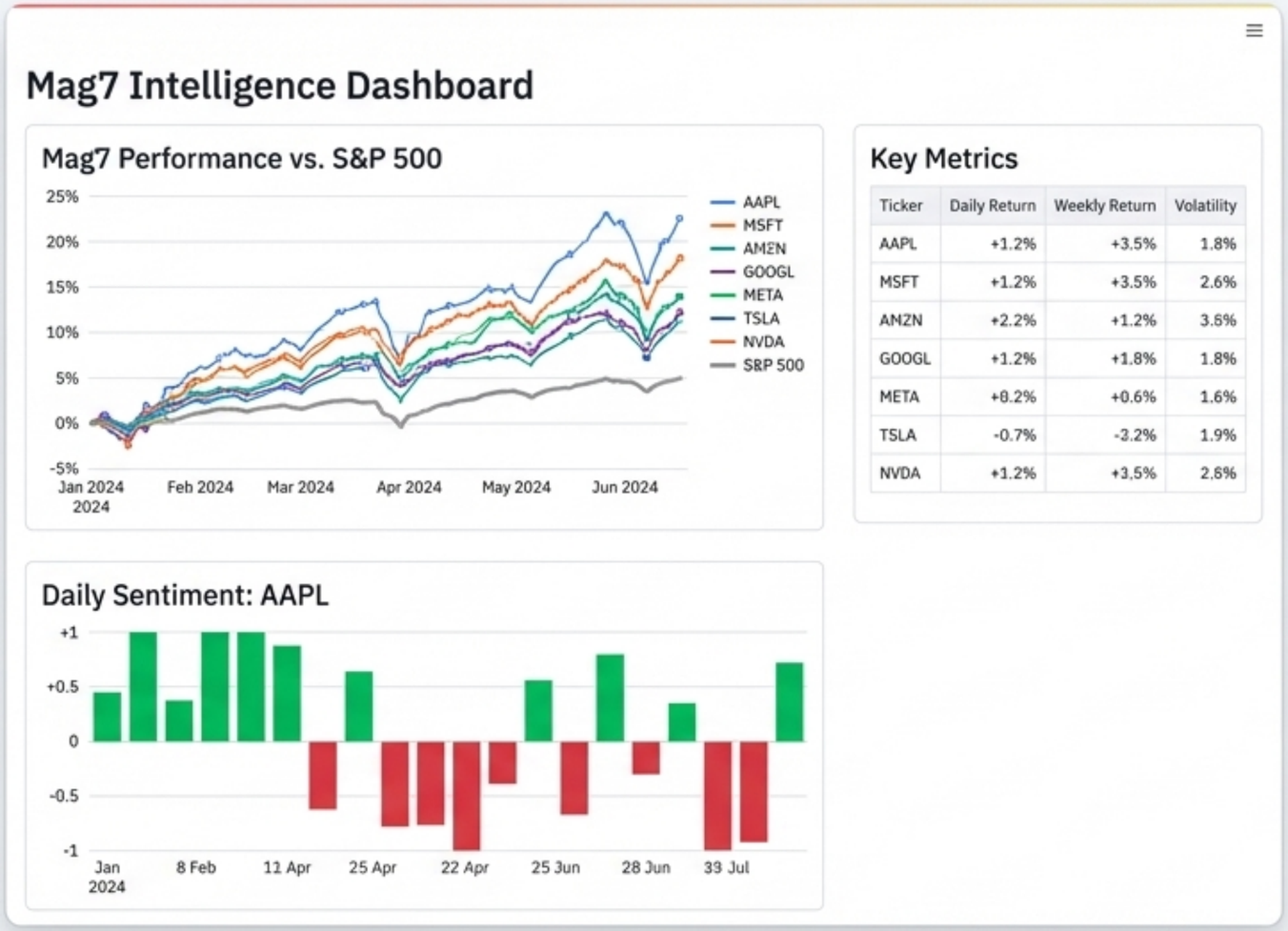
activate.d/env_vars.sh

```
#!/bin/bash
# This script automatically loads environment variables upon
# activating the conda environment.

export GOOGLE_APPLICATION_CREDENTIALS="</path/to/service-
account.json>"
export GCP_PROJECT_ID="<your-gcp-project-id>"
export BQ_DATASET_RAW="mag7_intel_raw"
# ... and other variables
```

The Final Product: Insights Delivered via Streamlit

Delivering interactive insights from the intelligence lakehouse.



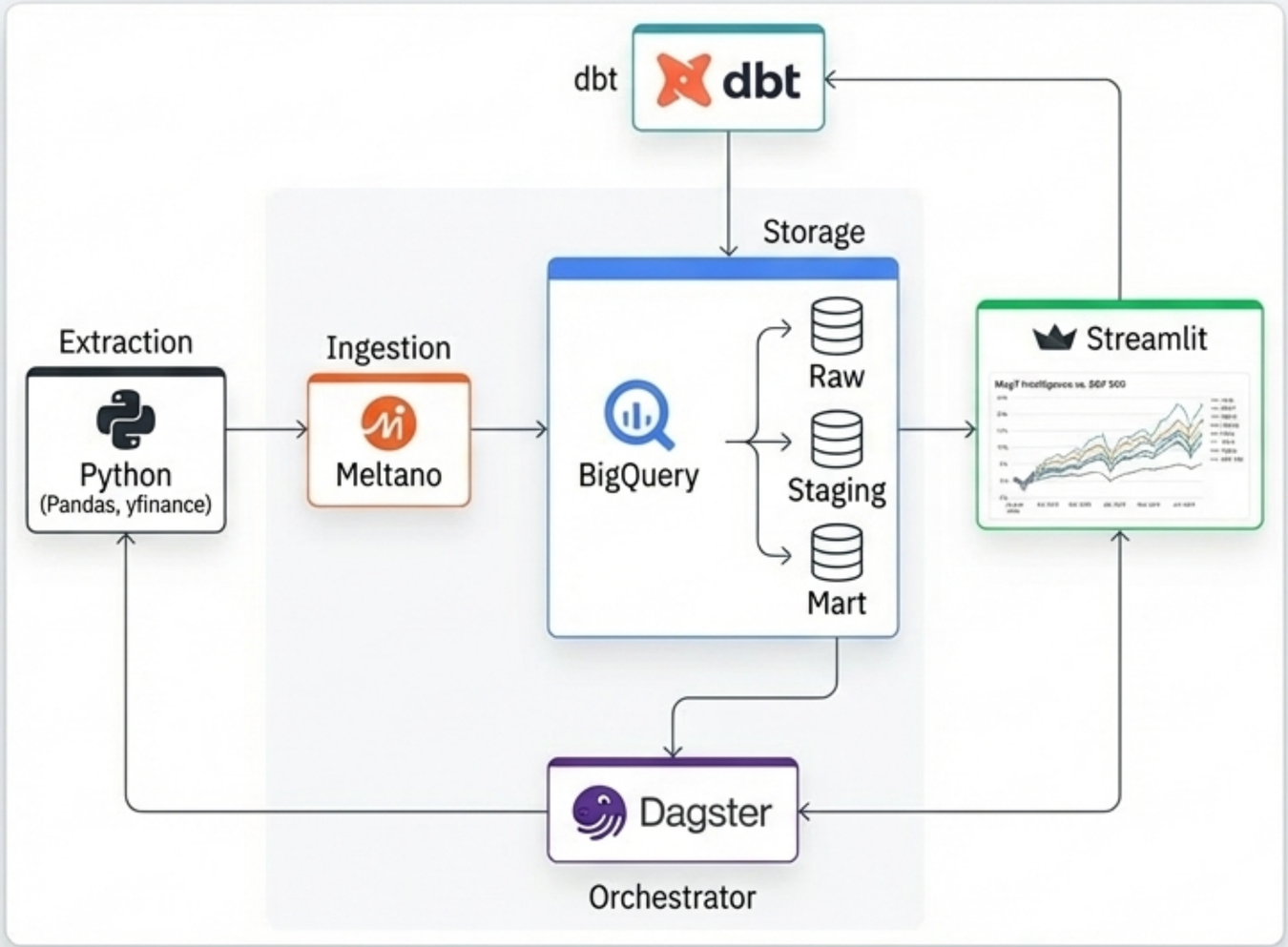
Purpose & Functionality

The Streamlit application provides the front-end for our intelligence lakehouse. It directly queries the `mart` tables in BigQuery to deliver:

- An at-a-glance overview of market performance and sentiment.
- Interactive charts for exploring trends.
- A user-friendly interface for non-technical stakeholders to consume the data.

The Complete Mag7 Intelligence Lakehouse

Full blueprint, final system architecture



Technology Stack Summary

Cloud & Warehousing
Google Cloud Platform, BigQuery

Ingestion
Meltano

Transformation
dbt

Orchestration
Dagster

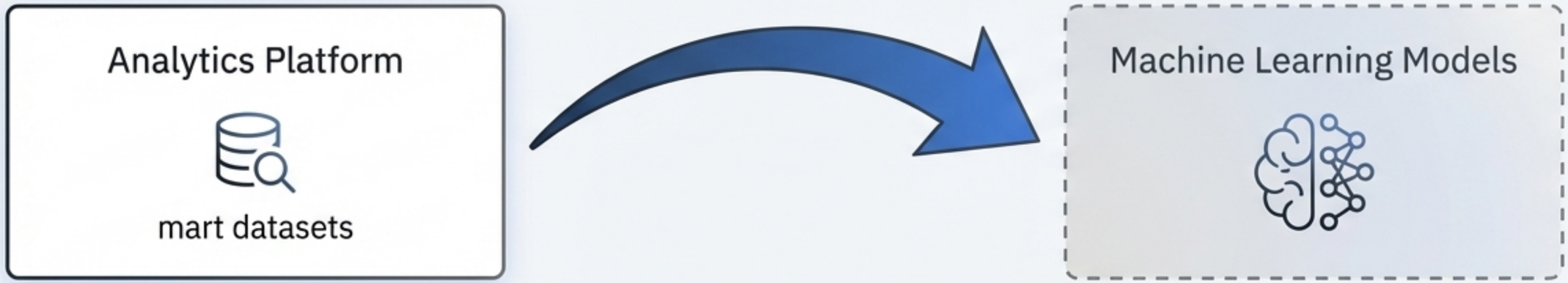
Extraction & Analysis
Python (Pandas, yfinance)

Visualisation
Streamlit

Environment
Conda, Docker

The Next Frontier: From Analytics to Prediction

Roadmap: Phase 2: The current lakehouse provides the foundational data asset for the next phase of development.



Future Initiatives

- **ML Feature Store**: Build a dedicated feature store within BigQuery (`mag7_intel_ml`) containing engineered features like rolling averages (ATR), volatility metrics, and lagged returns.
- **Predictive Models**: Develop and deploy models to generate predictive signals, with outputs stored in the `mag7_intel_pred` dataset.
- **Enhanced Dashboard**: Integrate model predictions and signals directly into the Streamlit dashboard.