AGRITRADE DATA ENGINE

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

PROJECT: WEATHER-TO-PRICE ETL PIPELINE

Yann Demuyt

SOMMAIRE

01

EXECUTIVE SUMMARY

1.1 Project Objective

To develop an automated, cloud-ready ETL (Extract, Transform, Load) pipeline that correlates live Brazilian weather data with agricultural commodity futures prices. The system provides a clean, validated dataset for machine learning models to forecast 7-day price trends.

1.2 Core Value Proposition

- **Data Observability:** Proactive monitoring of data freshness, volume, and schema integrity.
- **Code Quality:** Strict enforcement of Python type hints, linting, and 90%+ test coverage.
- **Resilience:** Automated failure detection and "circuit breakers" to prevent bad data from polluting the database.

02

FUNCTIONAL REQUIREMENTS

2.1 Pipeline Stages

Feature	Description	Priority
Extraction	Automated retrieval from yfinance (Soybeans/Coffee) and OpenWeatherMap.	High
Validation	Schema enforcement using **Pandera** to verify data types and ranges.	High
Transformation	Merging datasets on date index and feature engineering for ML.	High
Loading	Upserting cleaned data into a containerized **PostgreSQL** instance.	High
Forecasting	7-day price movement prediction using **XGBoost/Prophet**.	Medium

2.2 Data Quality Rules

- **Freshness:** The pipeline must complete by 07:00 BRT daily; alerts fire if data is >24 hours old.
- **Completeness:** Fail the build if precipitation or price columns contain NULL values.
- **Accuracy:** Sanity checks for outliers (e.g., rainfall < 0 or price drops > 50% in one day).

03

TECHNICAL ARCHITECTURE & STACK DECISIONS

3.1 Layered Data Design

1. **Ingestion Layer:** Python scripts using yfinance and requests.
2. **Processing Layer:** Pandas/Pandera for transformation and validation.
3. **Storage Layer:** PostgreSQL (structured relational schema).
4. **Orchestration Layer:** GitHub Actions (or Cron) for scheduling.

3.2 Technology Stack Selection (ADR)

- **Language:** Python 3.12+ (emphasizing Type Hints/Pydantic).
- **Database:** PostgreSQL (Dockerized).
- **DevOps:** GitHub Actions, Docker Compose.
- **Observability:** Loguru for logging; GitHub Action status for alerting.

04

QUALITY ASSURANCE STRATEGY

This project will adopt a **Multi-Layer Quality Assurance** strategy to ensure high reliability and regression prevention.

4.1 Automated Testing Pyramid

- **Unit Tests (Pytest):** Verify individual transformation logic and API mock responses.
- **Integration Tests:** Verify successful INSERT operations into the PostgreSQL container.
- **Data Validation Tests (Pandera):** Run during the pipeline to "Stop the Line" if schema shifts occur.

4.2 Code Quality Tools

- **Ruff:** Fast linting/formatting to ensure LDC-standard code style.
- **Pre-commit Hooks:** Enforce linting and static analysis before any git push.

05

ROADMAP

5.1 Project Roadmap

Milestone	Deliverable	Technical Focus
M1: Core Pipeline	Functional ETL script and database schema.	Python yfinance/requests setup, Repository Pattern, PostgreSQL schema design. +2
M2: Data Validation	Validated dataset with automated error logging.	**Pandera** schema enforcement, strict type hinting, Pydantic model integration.
M3: Containerization	Dockerized environment for App and DB.	docker-compose configuration, volume persistence, environment variable management.
M4: CI/CD & QA	Automated test suite and deployment pipeline.	**Pytest** (Unit/Integration), **Ruff** linting, GitHub Actions workflow setup. +2
M5: Analytics & ML	7-day price forecasting model and dashboard.	Feature engineering (Weather/Price), model training, deployment of prediction endpoint. +1