1. Environment & Configuration

- **Python Environment**: Created dedicated Conda environment (Python 3.12) with required packages including pandas, numpy, yfinance, matplotlib, seaborn, and arch for GARCH modeling.
- **Configuration Management**: All runtime settings stored in config/.env file and loaded via src/utils/config.py, ensuring no hard-coded values.
- **Project Structure**: Modular ETL pipeline with separate extract, transform, and load components for maintainability and scalability.

2. Utility Infrastructure

- **Logging System (src/utils/logger.py)**: Unified logging that outputs to both console and rotating log files (logs/pipeline.log) with consistent formatting across all modules
- **Database Integration:** PostgreSQL staging table (**marketpulse.stg_stocks**) implemented for data persistence and potential multi-user access. Current pipeline processes data via CSV workflow for simplicity, with database capability available for future scaling requirements.
- **Configuration Management**: Centralized settings management for file paths, database credentials, and pipeline parameters

3. Data Extraction & Sourcing

- **Data Source:** Yahoo Finance API via yfinance library for comprehensive OHLCV market data
- **Scope:** One year of daily trading data (July 2024 - July 2025) for major technology stocks (AAPL, MSFT, GOOGL, TSLA)
- **Extract Module (src/main/extract/fetch_data.py):**
  - Automated downloads with error handling and retry logic
  - Atomic file operations using temporary files to prevent corrupted writes
  - Data validation and logging of extraction success/failure rates

4. Data Cleaning & Preprocessing

- **Cleaning Module (src/main/transform/clean_data.py):** Production-grade data cleaning pipeline with comprehensive validation
- **Data Quality Issues Addressed:**
  - Removal of corrupted header rows and invalid data entries
  - Data type conversion for price columns with error handling
  - Date standardization and chronological sorting
  - Validation of expected OHLCV column structure

    - Missing value analysis with business logic
- **Output:** Combined ‘clean_stocks.csv’ with standardized format across all tickers
- **Quality Metrics:** Automated logging of data loss percentages, validation results, and quality warnings

5. Financial Feature Engineering

- **Transform Module (src/main/transform/calculate_returns_volatility.py):** Converts raw price data into analysis-ready financial metrics
- **Engineered Features:**
- **Daily Log Returns:** ln(P_t/P_t-1), preferred for financial modeling due to additive properties and normal distribution assumptions
- **Rolling Volatility:** 14-day rolling standard deviation of returns, annualized by √252 trading days factor
- **Simple Returns:** Percentage returns for business interpretability
- **Cumulative Returns:** Total return performance from period start
- **Technical Implementation:**
    - Group-by-ticker processing to maintain time series integrity
    - Proper handling of expected missing values at series boundaries
    - Validation of calculated metrics against expected ranges

6. Advanced Statistical Modeling

- **GARCH Implementation (src/main/ml/forecast_volatility.py):**
    - GARCH(1,1) models fitted for each ticker using arch library
    - Model parameter estimation with maximum likelihood
    - Comprehensive model diagnostics and validation
- **Statistical Analysis:**
    - **Model Selection:** AIC-based comparison across tickers
    - **Persistence Analysis:** Calculation of $\alpha + \beta$ coefficients for volatility clustering assessment
    - **Forecasting:** 14-day ahead volatility predictions with confidence intervals
    - **Model Validation:** Residual analysis and convergence checks
- **Key Findings:**
    - MSFT: Best GARCH fit (AIC: 919.53) indicating predictable volatility patterns
    - TSLA: Highest volatility but complex dynamics (AIC: 1464.23)
    - All models exhibit high persistence (0.93-0.98) confirming volatility clustering

7. Production Pipeline Architecture

- **ETL Orchestration (`src/main/main.py`):** Central pipeline controller managing end-to-end data flow
- **Pipeline Stages:**
  - Raw data extraction from Yahoo Finance API
  - Data cleaning and standardization with quality checks
  - Financial feature engineering and validation
  - GARCH modeling and volatility forecasting
  - Output generation for dashboard consumption
- **Error Handling:** Comprehensive exception handling with detailed logging at each stage, ensuring pipeline resilience
- **Data Lineage:** Clear data flow from raw CSVs → cleaned data → processed metrics → forecasts → dashboard-ready outputs

8. Output & Deliverables

- **Primary Dataset:** 'processed_stocks.csv' containing all OHLCV data plus engineered financial features
- **Forecast Data:** 'all_tickers_garch_forecasts.csv' with 14-day volatility predictions and confidence intervals
- **Model Results:** 'garch_model_comparison.csv' with performance metrics and parameter estimates
- **Data Quality Validation:**
  - Expected missing values documented (4 initial returns, 56 initial volatility values due to rolling windows)
  - Statistical validation of all calculated metrics
  - Automated quality checks and warnings
- **Dashboard Integration:** Optimized datasets for Power BI consumption with proper relationships and measures

9. Statistical Validation

- **Data Quality Checks:**
  - Automated validation of price ranges and statistical properties
  - Missing value analysis with business logic validation
  - Outlier detection and extreme value flagging
- **Model Validation:**
  - GARCH parameter significance testing
  - Residual analysis and diagnostic checks
  - Forecast accuracy assessment and model comparison
- **Performance Metrics:**
  - AIC-based model ranking revealing varying volatility dynamics across tickers
  - Persistence analysis indicating long-lasting volatility shocks across all stocks
  - Model convergence and stability validation

10. Reproducibility & Deployment

- **Automation:** Fully automated pipeline from raw data extraction to analysis-ready outputs
- **Configuration Management:** Environment-specific settings enable deployment across development, testing, and production environments
- **Documentation:** Comprehensive logging and documentation ensuring reproducible results and easy maintenance
- **Scalability:** Modular architecture supports easy addition of new tickers, different time periods, or alternative modeling approaches