S&P 500 Sector-Sensitivity Monitor

A system that reveals the hidden drivers of S&P 500 by measuring real sector influence

Team Members: Disha, Haritha, Rahul, Ruchika

The Problem: "Illusion of Safety"

Flaw:

Diversification is an **'Illusion of Safety'**—even with 500 stocks, Technology's **High Impact Weight** effectively overrides the rest of the index

Gap:

Traditional dashboards track **weight** (Size) but miss **Sensitivity** (behavior), leaving investors blind to hidden correlations

Solution:

We measured each sector's **sensitivity** to SPY using **rolling betas = Covariance (sector, benchmark) / Variance (benchmark)**, enabling **apples-to-apples** comparison of sector risk

"Simply put, Beta tells us:

If the S&P 500 moves 1%, how much does this sector move?"

- **If Beta = 1.0:** It moves **1%** (In-sync)
- **If Beta = 1.5:** It moves **1.5%** (High Risk)
- **If Beta = 0.5:** It moves **0.5%** (Defensive)

Data Foundation (Historical & Daily)

Source:

Yahoo Finance API (Daily OHLCV)

Scope:

6 Years of History (Backfilled) + Daily Incremental Ingestion

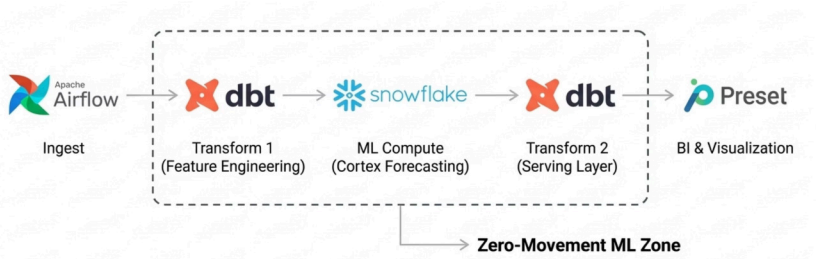
Entities:

11 S&P 500 Sectors (XLK, XLE, XLF, etc.) + SPY Benchmark

Storage Strategy:

Raw data ingested into Snowflake

High-Level Architecture



Ingest:

Airflow triggers Python extraction to snowflake

Transform 1 (Feature Eng):

dbt calculates **rolling beta** & **log-returns**

ML Compute:

Snowflake Cortex trains on warehouse data and does the forecasting

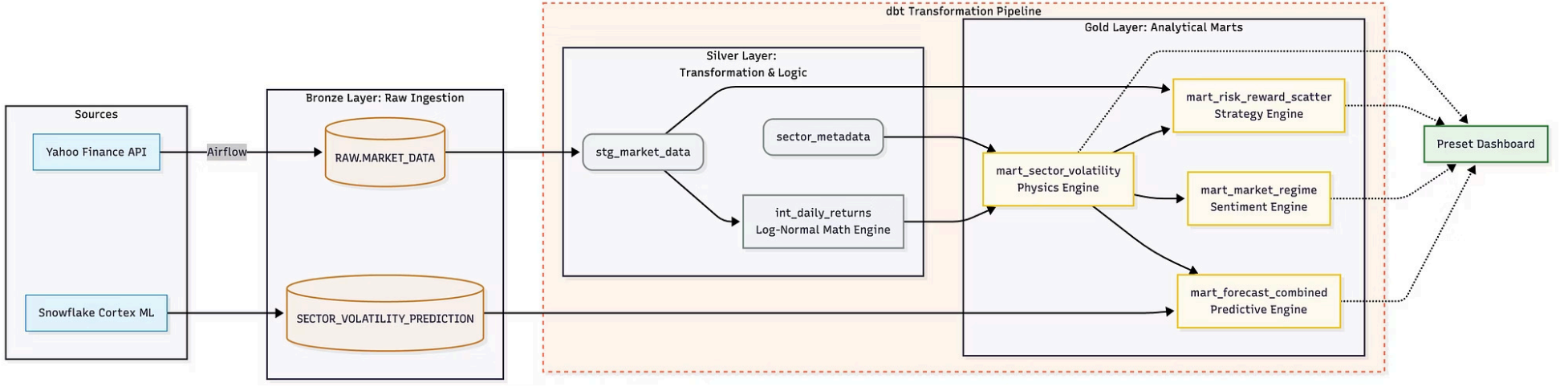
Transform 2 (Serving):

dbt stitches forecasts with history

Visualize:

Preset consumes the final marts

Database Schema



Raw Ingestion:

Lands daily OHLCV data from Yahoo Finance

Staging Layer:

Standardizes types and enforces **idempotency**

Compute layer 1:

Calculates **log-returns** to ensure statistical stationarity

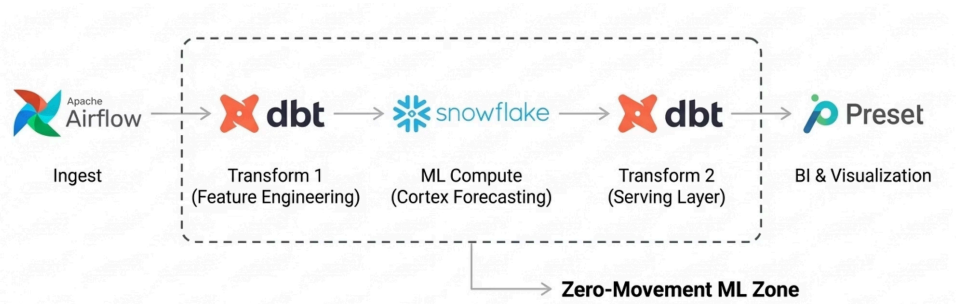
Compute layer 2:

Computes **90-day rolling beta** and **weighted risk contribution**

Strategy & ML:

Merges Snowflake Cortex predictions with historical actuals

ETL & Orchestration



Orchestration:

4 Decoupled Airflow DAGs using ExternalTaskSensor to prevent cascade failures

Idempotency:

Implemented **truncate-insert** patterns to ensure safe re-runs without data duplication

Transformation Logic:

window functions: Used **ROWS BETWEEN 90 PRECEDING** for O(N) scalable covariance calculation

Data Quality:

Automated dbt tests (**not_null**, **relationships**) run before every dashboard update.

Visualization (Live Demo Setup)

Preset Dashboard

Conclusion & Future Roadmap

Summary:

Successfully operationalized an end-to-end ELT pipeline (Airflow, Snowflake, dbt, Cortex) that quantifies S&P 500 diversification using **rolling beta** instead of just Price

Wins:

- Implemented **truncate-insert** patterns to guarantee idempotency during ingestion.
- Utilized **window functions** for covariance calculations, replacing expensive self-joins

Lessons/Future Work:

- **Sector share splits** event revealed that static models break during structural changes, creating a critical need for historical preservation
- Implement **dbt snapshots** (SCD Type 2) to track regime changes without overwriting history
- Move sector weights to versioned **seed files** to automatically adjust beta logic during future splits or rebalancing events
- Implement multi-factor models

Disclaimer 😇 : Our model suggests the S&P 500 as stable, however, this is not a financial advice, just data engg.