

Stock Market Trend Analysis Using ELT

An End-to-End Data Pipeline with Airflow, dbt, Snowflake, and Superset

By Veda Kulkarni and Rishi Visweswar Boppana

Department of Applied Data Science

San Jose State University

Date: April 24, 2025

Abstract

This project presents an end-to-end data analytics pipeline designed to analyze historical stock market trends. The system focuses on extracting, storing, and transforming stock data to generate structured outputs that can be easily explored through interactive dashboards. These dashboards enable users to gain insight into market behavior by visualizing time-based changes and comparing stock activity across different companies. The pipeline is designed to be modular and automated, supporting efficient data flow and integration across components. It emphasizes scalability and adaptability, making it suitable for analytical use cases such as financial reporting, performance tracking, and trend-based analysis. By combining automation with a structured ELT process, the system enables users to draw meaningful insights from complex financial datasets.

1. Introduction

Stock market data analysis is critical for businesses and investors aiming to navigate the complexities of financial markets. By examining historical stock data, analysts can identify patterns and trends that inform future market movements. Accurate predictions enable stakeholders to make informed decisions, optimize investment strategies, and manage risks effectively. This project delves into the methodologies and techniques used to analyze stock market data, providing insights into the factors that influence stock prices and offering a framework for analyzing market trends. Through comprehensive data analysis, the project aims to enhance understanding of market dynamics and support strategic financial planning.

2. Problem Statement

Analyzing stock market trends is challenging due to high volatility and external influencing factors. Traditional models often fall short in handling the nonlinear behavior of financial time series data. This project addresses the problem by implementing a robust ELT pipeline that extracts stock data using the yfinance API, transforms it using financial indicators, and presents insights through an interactive BI dashboard for analysts.

3. Solution Requirements

3.1 System Requirements

To support the implementation and execution of the stock market analytics system, the following technologies and platforms are required:

- **Programming Language:** Python, for writing ETL pipelines and transformation scripts.
- **Data Source:** `yfinance` API, used to fetch historical stock data for six selected companies.
- **Data Warehouse:** Snowflake, used to store both raw and transformed data.
- **Workflow Orchestration Tool:** Apache Airflow, responsible for scheduling and running ETL/ELT pipelines.
- **Transformation Framework:** dbt (Data Build Tool), used to create abstracted and analytical data models.
- **Visualization Tool:** Apache Superset, used for building interactive dashboards to visualize trends and insights.
- **Version Control System:** Git and GitHub, used for code collaboration and versioning.
- **Optional Tools:** Docker or a virtual environment to manage dependencies and isolate the project environment.

3.2 System Capabilities

The system is designed to automate the end-to-end process of stock data extraction, transformation, and visualization. Its core capabilities include:

- **Automated Data Extraction:** Pulls daily historical stock data for selected companies using the `yfinance` API.
- **ETL with Airflow:** Loads the raw data into Snowflake and schedules regular updates via Airflow DAGs.
- **Data Transformation with dbt:** Performs advanced data modeling and analytics, including:
 - Moving averages to detect price trends
 - RSI (Relative Strength Index) for momentum analysis
 - Rolling volatility calculations to measure market risk
 - Gap analysis between opening and closing prices
- **Visual Insights with Superset:** Presents key metrics using:
 - Line charts for price movement and trend analysis
 - Box plots for volatility distribution
 - Bar charts for comparative stock performance
 - Date-range filters for temporal exploration
- **Pipeline Automation:** Ensures all processes run on schedule with minimal manual intervention.

3.3 System Limitation

While the system is efficient for batch-based stock analysis, it has some limitations:

- **Data Source Dependency:** Reliance on the `yfinance` API means data availability and accuracy are subject to the API's performance and limitations (e.g., rate limiting).
- **No Real-Time Processing:** The system does not support real-time stock predictions or high-frequency trading.
- **Prediction Accuracy Constraints:** Stock price movements are influenced by unpredictable market forces; thus, predictions based solely on historical data may have limited precision.
- **Scalability Considerations:** The current pipeline is optimized for six companies. Scaling to larger datasets or broader analysis will require performance tuning and resource allocation.

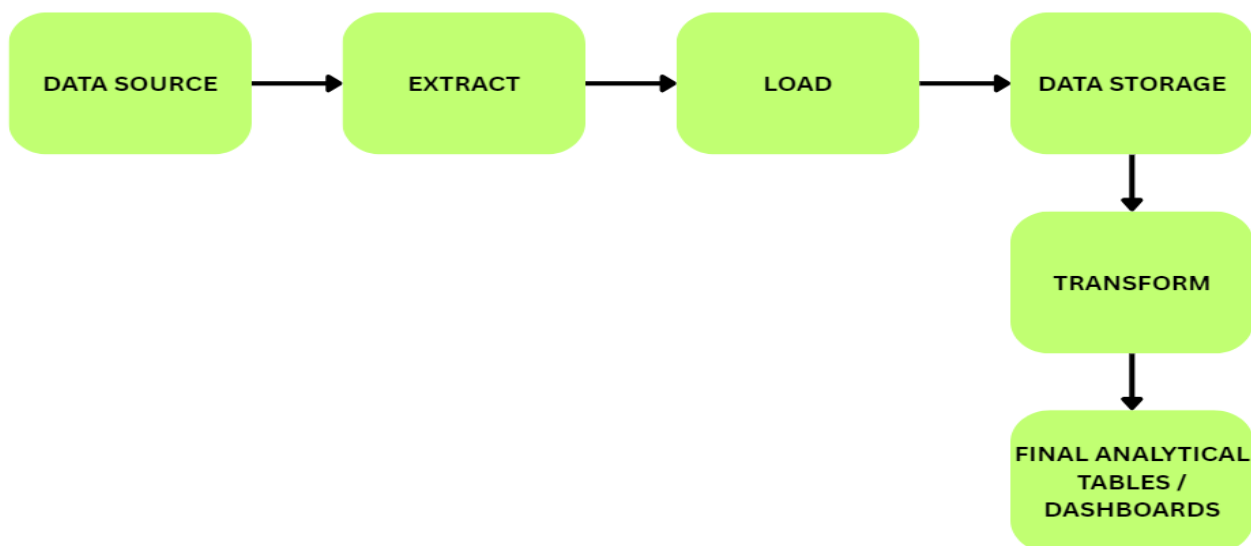
3.4 User Interaction

Different users will interact with the system in the following ways:

- Data Engineers will monitor and maintain the Airflow pipeline, ensuring the seamless flow of data from ingestion to visualization.
- Data Analysts and Business Users will explore insights using Superset dashboards, enabling them to:
 - Track price trends and momentum
 - Compare stocks across metrics like volatility and RSI
 - Apply filters and generate custom reports
- Developers can extend the system by integrating additional data sources or implementing new dbt models for deeper analysis.

4. System Architecture

The architecture of this project is designed as an end-to-end data analytics pipeline for stock market analysis and prediction. It is composed of four major stages: data ingestion, transformation, storage, and visualization, each powered by specialized tools to ensure scalability, automation, and analytical depth.

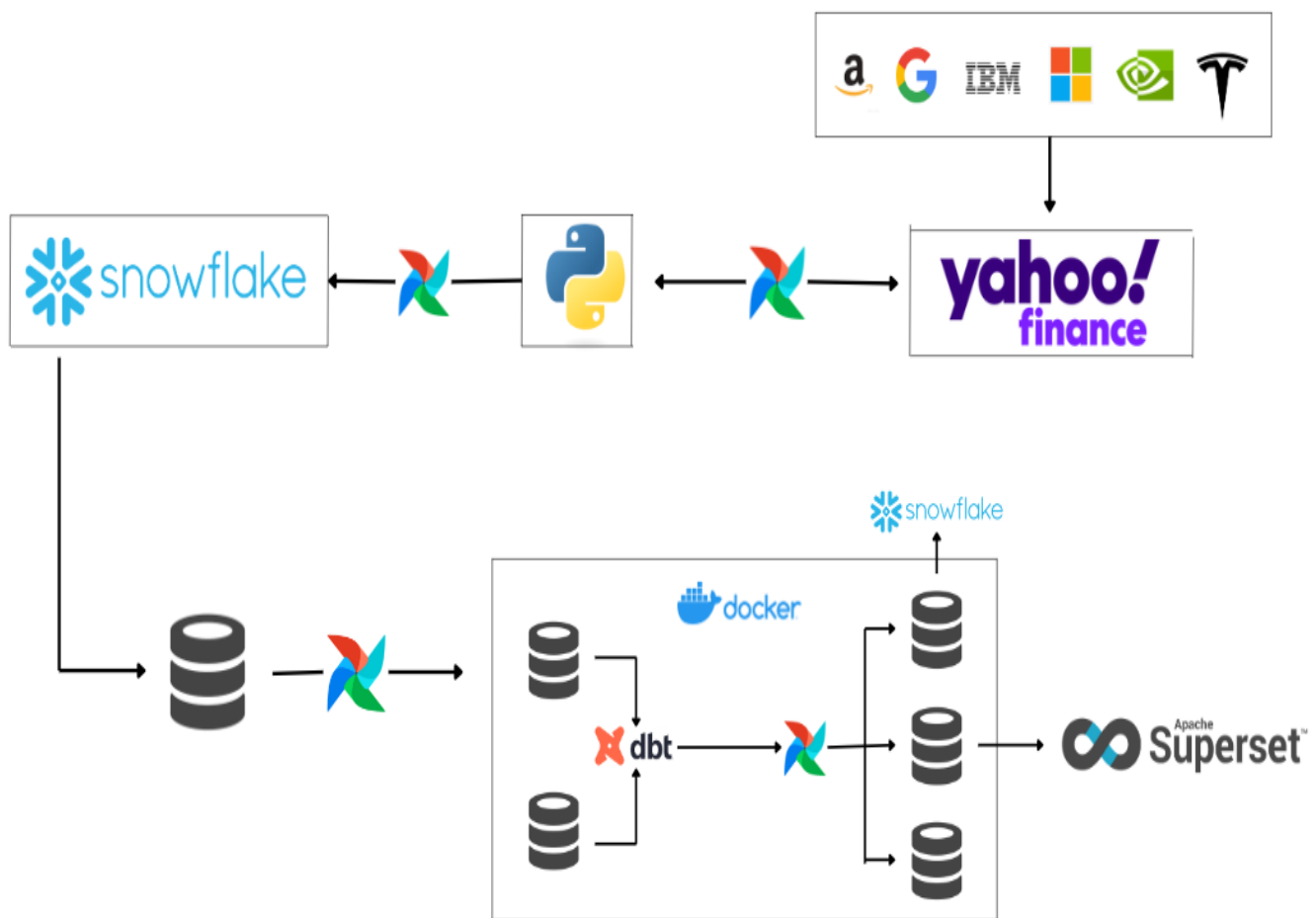


- **Data Source:** The pipeline begins with historical financial data collected through the yfinance API, which provides stock price information including opening and closing values, highs, lows, and volumes.
- **Data Processing (Airflow + dbt):**
 - Apache Airflow serves as the workflow orchestrator, automating the scheduling and running of data pipelines.
 - dbt (Data Build Tool) is used to transform raw data into structured models including gap analysis, moving averages, rolling volatility, RSI, and volatility.
- **Data Storage (Snowflake):** All raw and processed data is stored in Snowflake, enabling scalable and efficient querying and analysis.

- **Visualization (Apache Superset):** The final stage presents processed data through dashboards in Superset. This layer allows users to explore data visually via line charts, box plots, and bar charts with support for filtering and date-based exploration.

This modular and automated architecture supports efficient stock market analysis by streamlining data flow, maintaining quality, and providing accessible insights to users.

5. ELT Workflow Diagram Explanation



1. Data Source (yfinance):

The pipeline begins with historical stock data pulled from Yahoo Finance using the yfinance API, focused on selected companies such as Amazon, Google, and Tesla.

2. Extract (Python + Airflow):

Airflow orchestrates Python scripts to extract stock data daily and ensure scheduling, retries, and automation.

3. Load (Airflow → Snowflake):

Extracted data is directly loaded into Snowflake's raw tables, which represent the unprocessed data layer.

4. Transform (dbt within Docker):

dbt models, executed via Airflow inside Docker containers, transform raw data into analytical models. These include gap analysis, moving averages, RSI, rolling volatility, and intraday volatility. Each transformation is modular and version-controlled.

5. Storage (Snowflake):

Both raw and transformed data are persisted in Snowflake. Transformed models are structured for efficient querying and used by downstream BI tools.

6. Dashboard (Superset):

Final analytical tables are used to build interactive dashboards in Apache Superset. Users can explore trends, apply filters, and generate insights visually.

This diagram complements the architecture section and highlights the modular, automated nature of the system, where each tool is used purposefully within the ELT paradigm.

6. Analytical Features for Stock Analysis

6.1 Gap Analysis

Gap analysis measures the difference between a stock's opening price and the previous closing price. It reflects sudden shifts in market sentiment, often triggered by overnight news or economic events. A significant gap can indicate momentum or a reversal, making it a valuable early signal for traders.

Purpose:

In stock analysis, detecting gap movements helps identify potential breakout days or anomalies worth closer inspection.

```
select
  date,
  symbol,
  open - lag(close) over(partition by symbol order by date) as Gap,
from {{ source('raw', 'market_price') }}
```

This dbt model calculates the price gap per symbol by subtracting the previous day's closing price from the current day's opening price using a `lag()` window function.

6.2 Moving Averages

Moving averages smooth short-term fluctuations and highlight longer-term trends. A 10-day moving average helps detect momentum, indicating when to buy or sell based on trend confirmation.

Purpose:

They are fundamental for identifying bullish or bearish signals and help reduce noise in price data.

```
select
```

```
date,  
symbol,  
avg(close) over(partition by symbol order by date rows between 9 preceding and  
current row) as ma_10  
from {{source('raw','market_price')}}}
```

This model calculates a 10-day rolling average of the closing price, offering a trend-based view of stock performance.

6.3 Rolling Volatility

Rolling volatility measures the variation in returns over a fixed window. It's used to assess market risk — higher volatility means higher uncertainty, which could lead to larger gains or losses.

Purpose: This feature helps determine whether a stock is entering a period of stability or turbulence.

```
with deviation as (  
  select  
    date,  
    symbol,  
    (close/lag(close) over(partition by symbol order by date) - 1) as deviation  
  from {{source('raw','market_price')}}  
)  
select  
  date,  
  symbol,  
  stddev(deviation) over(partition by symbol order by date rows between 13  
preceding and current row) as rolling_volatility  
from deviation
```

The inner query calculates daily returns (deviation), and the outer query computes the rolling standard deviation over 14 days.

6.4 Relative Strength Index (RSI)

RSI is a momentum indicator that compares recent gains to recent losses. It ranges from 0 to 100, helping detect when a stock is overbought (>70) or oversold (<30), which could signal reversals.

Purpose:

RSI helps anticipate trend changes and refine entry/exit points for trades.

```
with delta_close as(
select
date,
symbol,
close - lag(close) over(partition by symbol order by date) as delta
from {{source('raw','market_price')}}
),
gain_and_loss as (
select
date,
symbol,
case when delta > 0 then delta else 0 end as gain,
case when delta < 0 then abs(delta) else 0 end as lose
from delta_close
),
rs as (
select
date,
symbol,
avg(gain) over(partition by symbol order by date rows between 13 preceding and
current row) as avg_gain,
avg(lose) over(partition by symbol order by date rows between 13 preceding and
current row) as avg_loss
from gain_and_loss
)
select
date,
symbol,
round(100 - 100/(1 + (avg_gain/nullif(avg_loss,0)))) as rsi
from rs
```

This model calculates the RSI using a rolling average of gains and losses over a 14-day window. nullif protects against division by zero.

6.5 Volatility (High-Low Range)

This is a simpler form of volatility that measures the intra-day price range. While not as robust as rolling volatility, it quickly shows how much a stock fluctuates during a trading day.

Purpose:

This metric is helpful to understand daily market pressure and helps compare stocks in terms of daily price movement intensity.


```
select
date,
symbol,
high - low as volatility
from {{source('raw', 'market_price')}}
```

The difference between daily high and low prices is used as a basic volatility indicator. It can be plotted to visualize stock stability on a daily basis.

7. Table Structure

7.1 Table: Market_price

Column Name	Data Type	Constraints	Description
CLOSE	Float	Not NULL	Stocks closing price for that day
DATE	Date	Not NULL	Date of that particular day
HIGH	Float	Not NULL	Highest point to where that stock went
LOW	Float	Not NULL	Lowest point to where that stock went
OPEN	Float	Not NULL	Price of the stock at the beginning of the day
SYMBOL	Varchar	Not NULL	Symbol/Stock logo of a company
VOLUME	Float	Not NULL	Size of the stock in the market

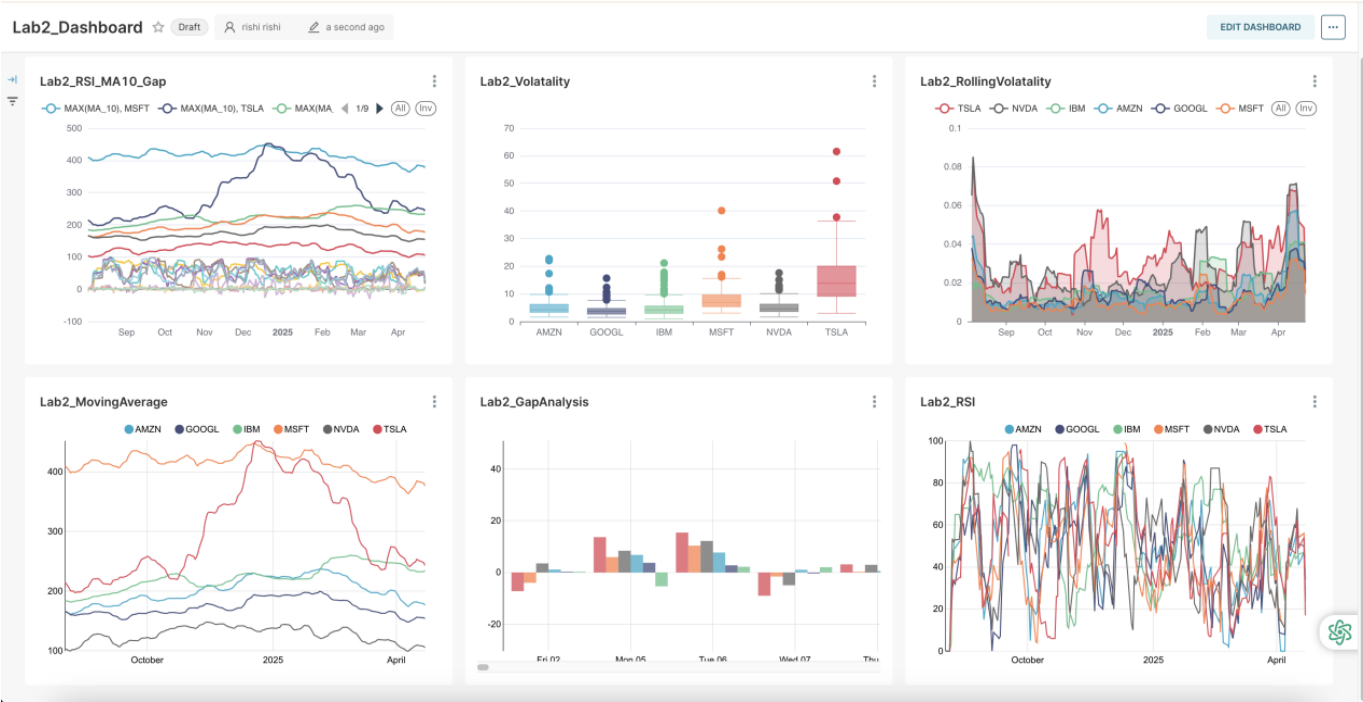
7.2 Table: Session_Summary

Column Name	Data Type	Constraints	Description
DATE	DATE	NOT NULL	The trading date of the stock data
SYMBOL	VARCHAR(16777216)	NOT NULL	The stock ticker symbol (e.g., AAPL, GOOGL)
MA_10	FLOAT	NULLABLE	10-day moving average of the closing price
GAP	FLOAT	NULLABLE	Difference between current day's open and previous day's close

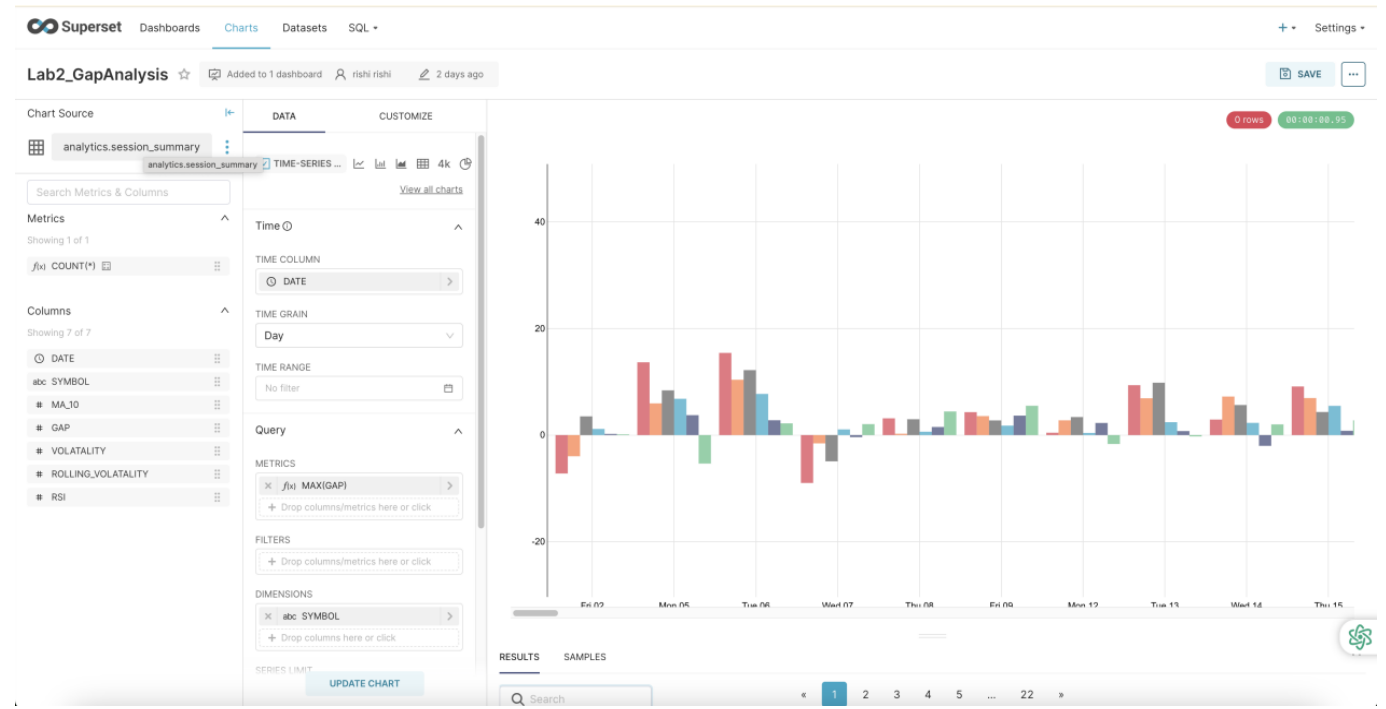
Column Name	Data Type	Constraints	Description
VOLATILITY	FLOAT	NULLABLE	Intra-day volatility (high - low)
ROLLING_VOLATILITY	FLOAT	NULLABLE	Rolling standard deviation of price changes over a 14-day window
RSI	FLOAT	NULLABLE	Relative Strength Index, measuring stock momentum over time

8. Visualization

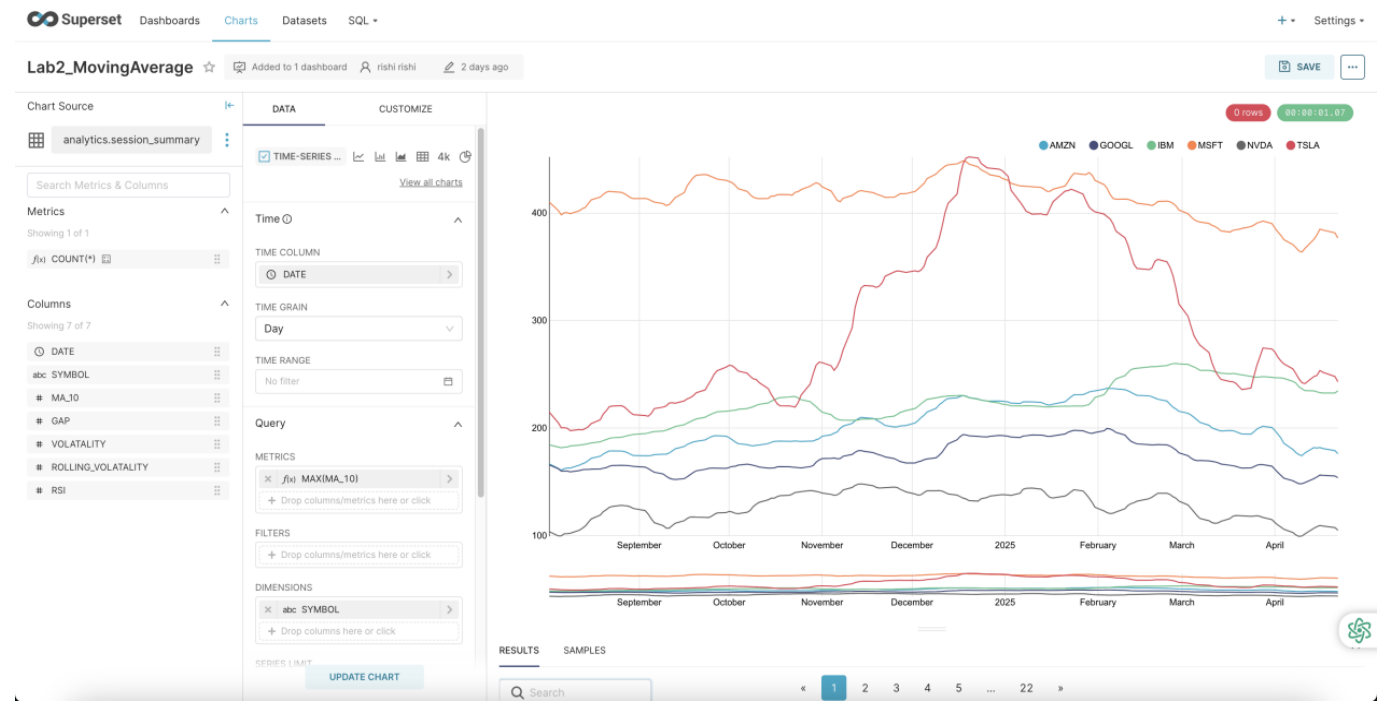
8.1 Dashboard



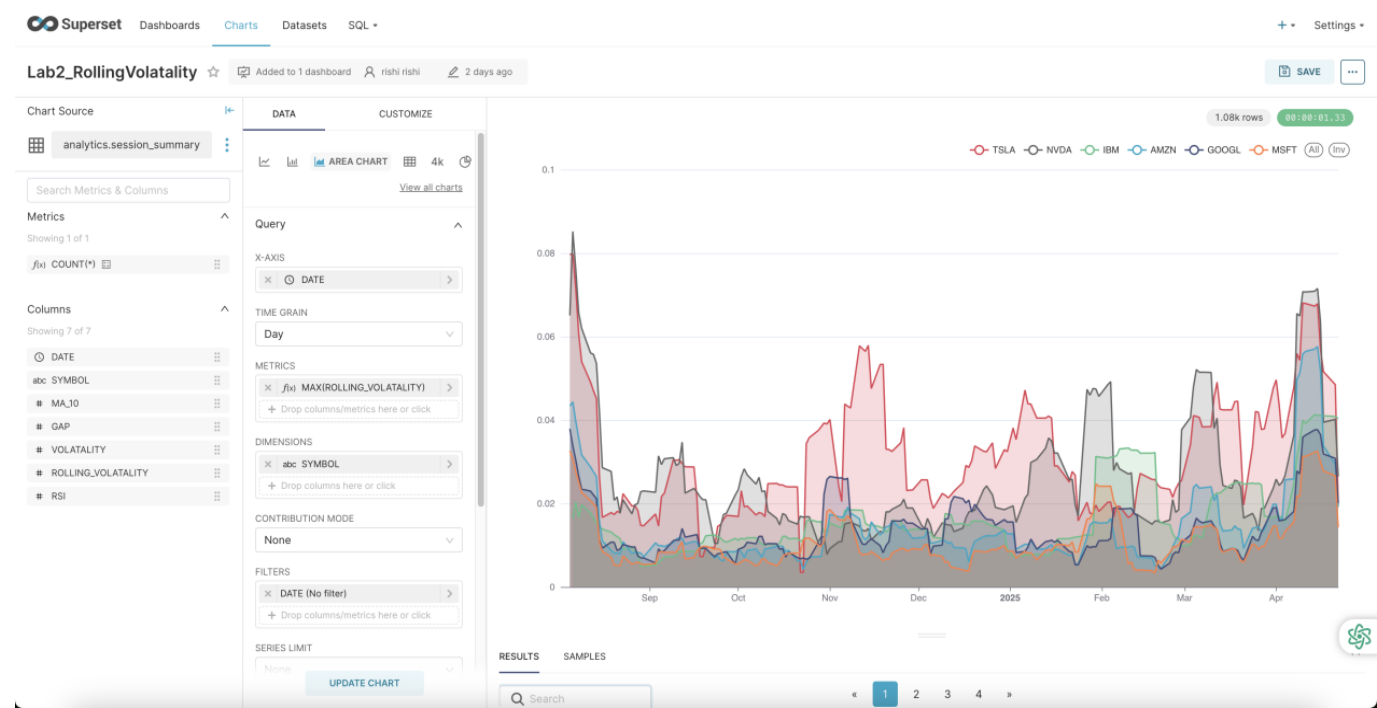
8.1.1 Gap Analysis



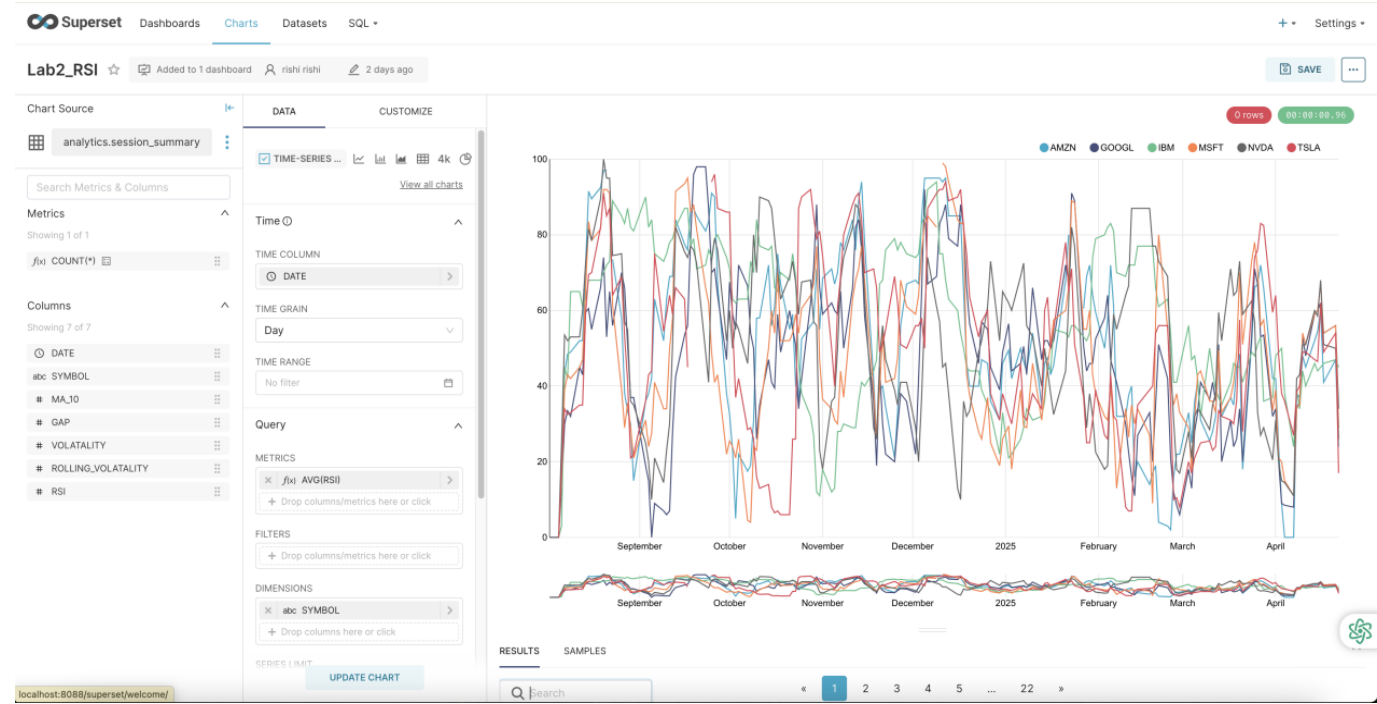
8.1.2 Moving Average



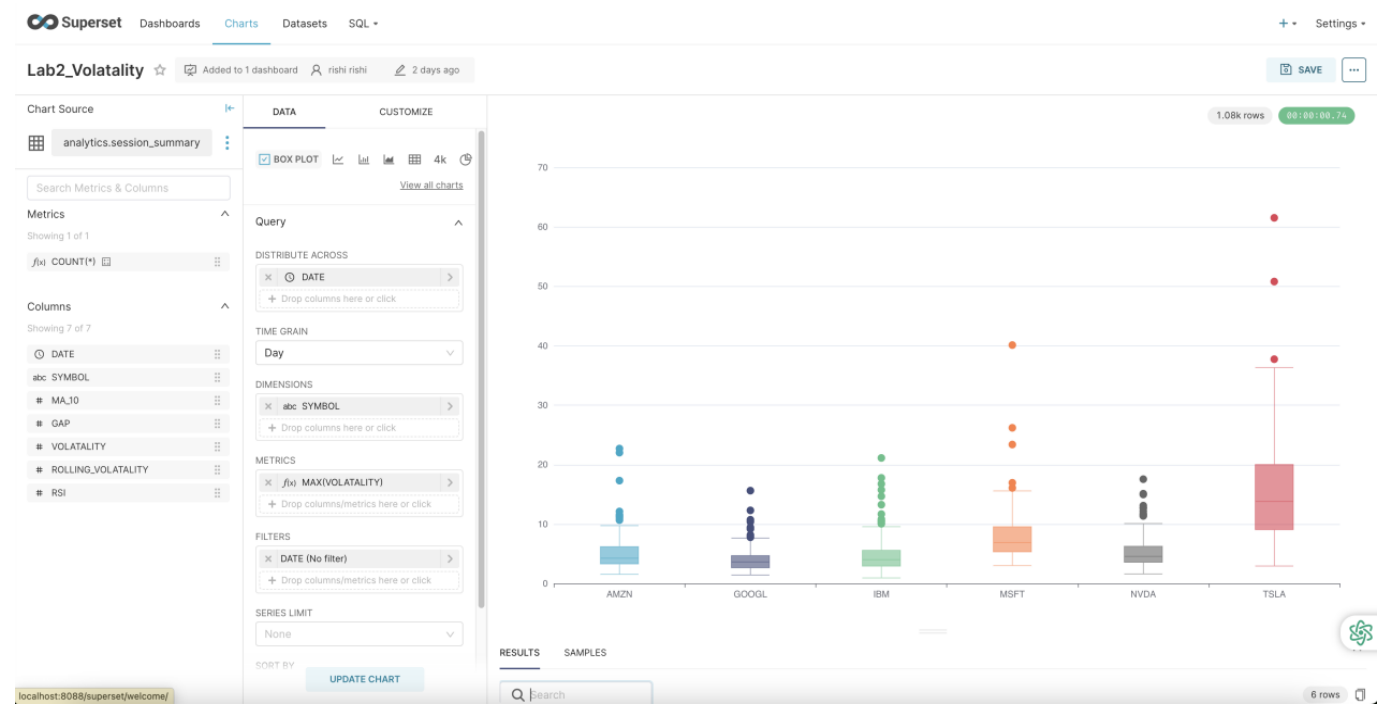
8.1.3 Rolling Volatility



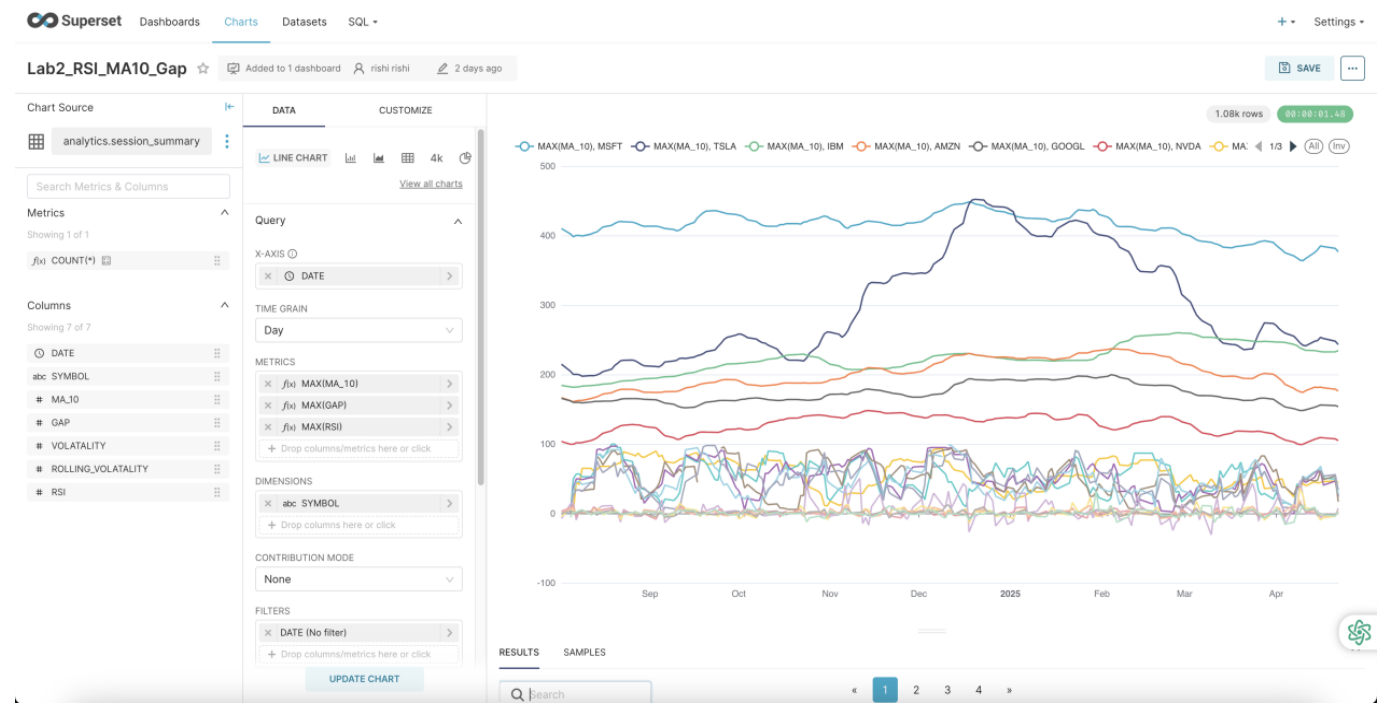
8.1.4 Relative Strength Index (RSI)



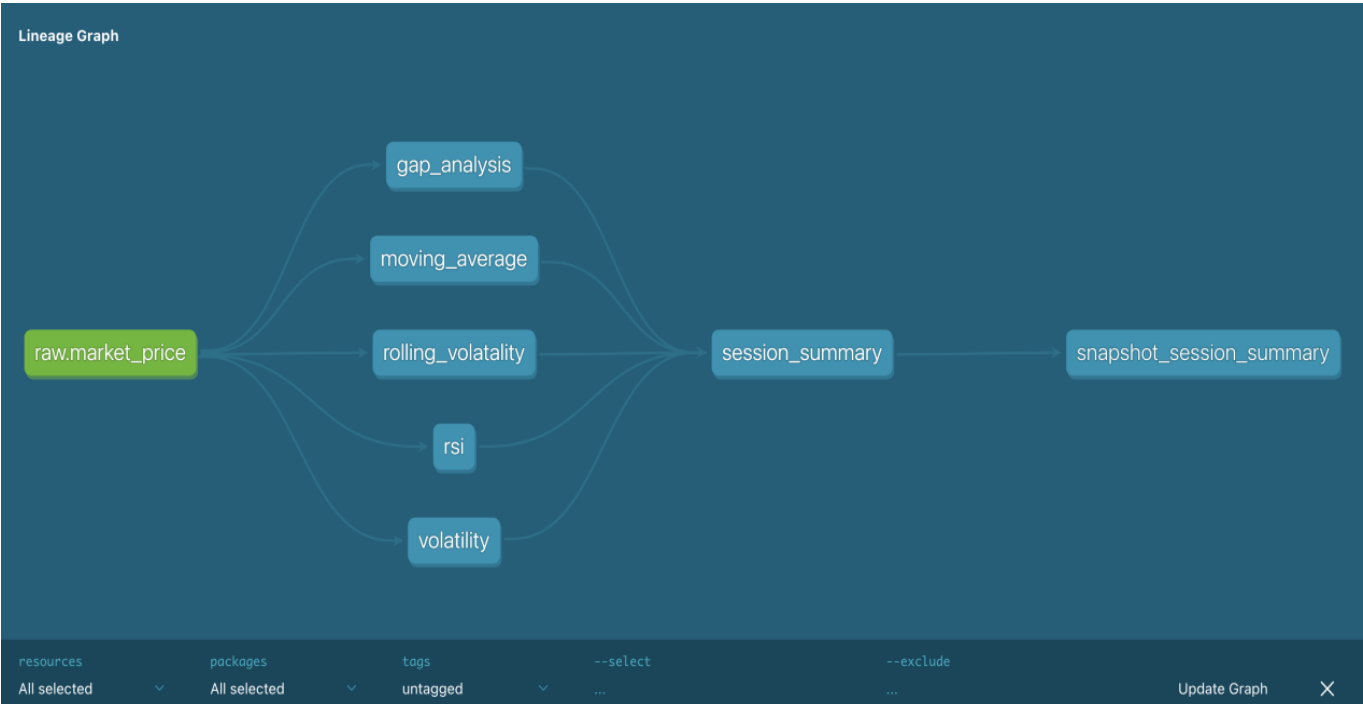
8.1.5 Volatility



8.1.6 Relative Strength Index (RSI) and Gap Analysis




8.1.7 Linage Graph



8.2 Airflow

8.2.1 Airflow Connections

 Airflow

DAGs

Cluster Activity

Datasets

Security

Browse

Admin

Docs

Edit Variable

Key *

tickers

Val

IBM,NVDA,MSFT,GOOGL,AMZN,TSLA,APPL

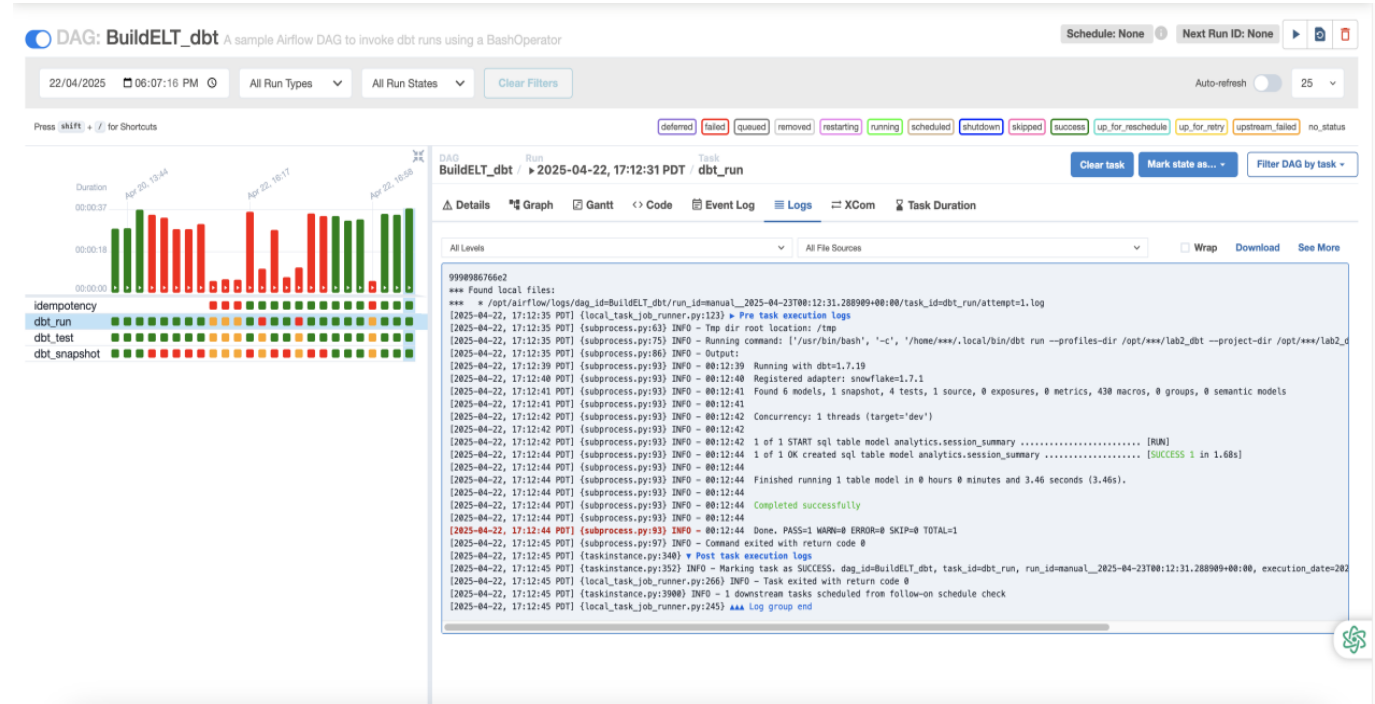
Description

Description

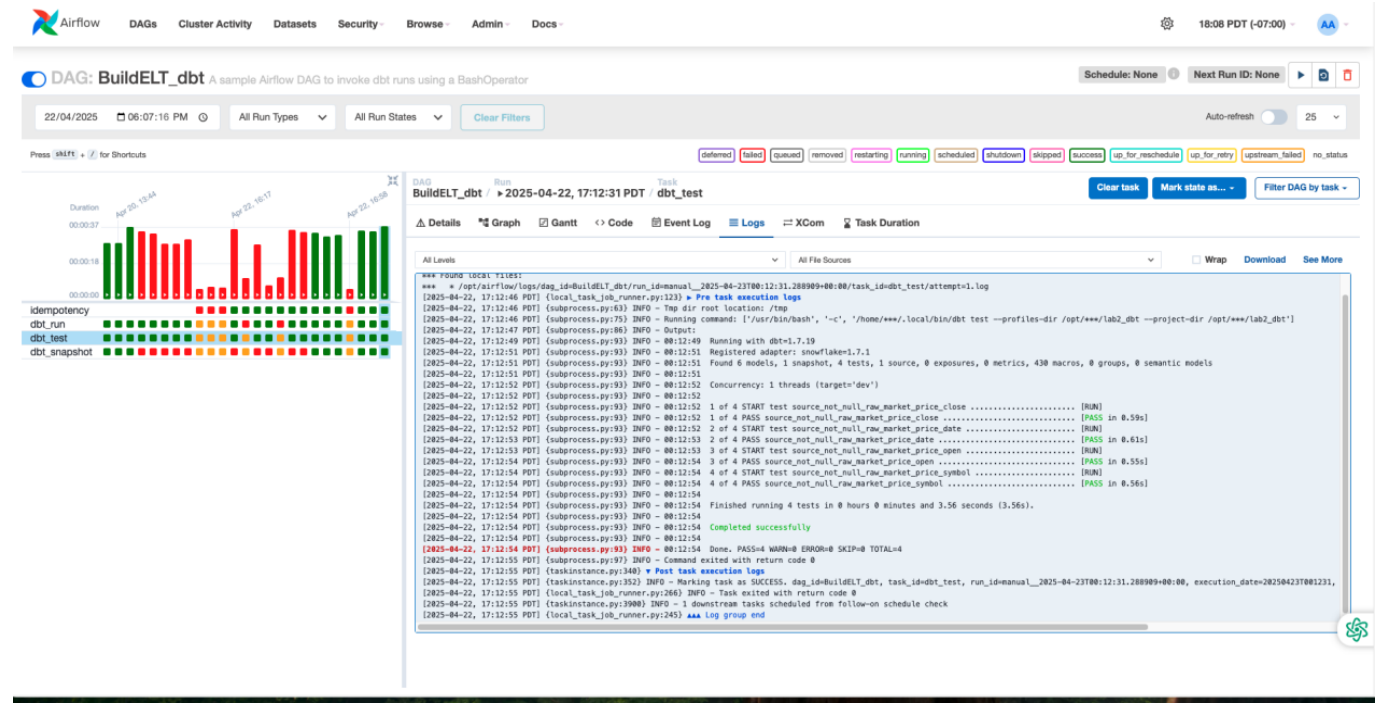
Save

←

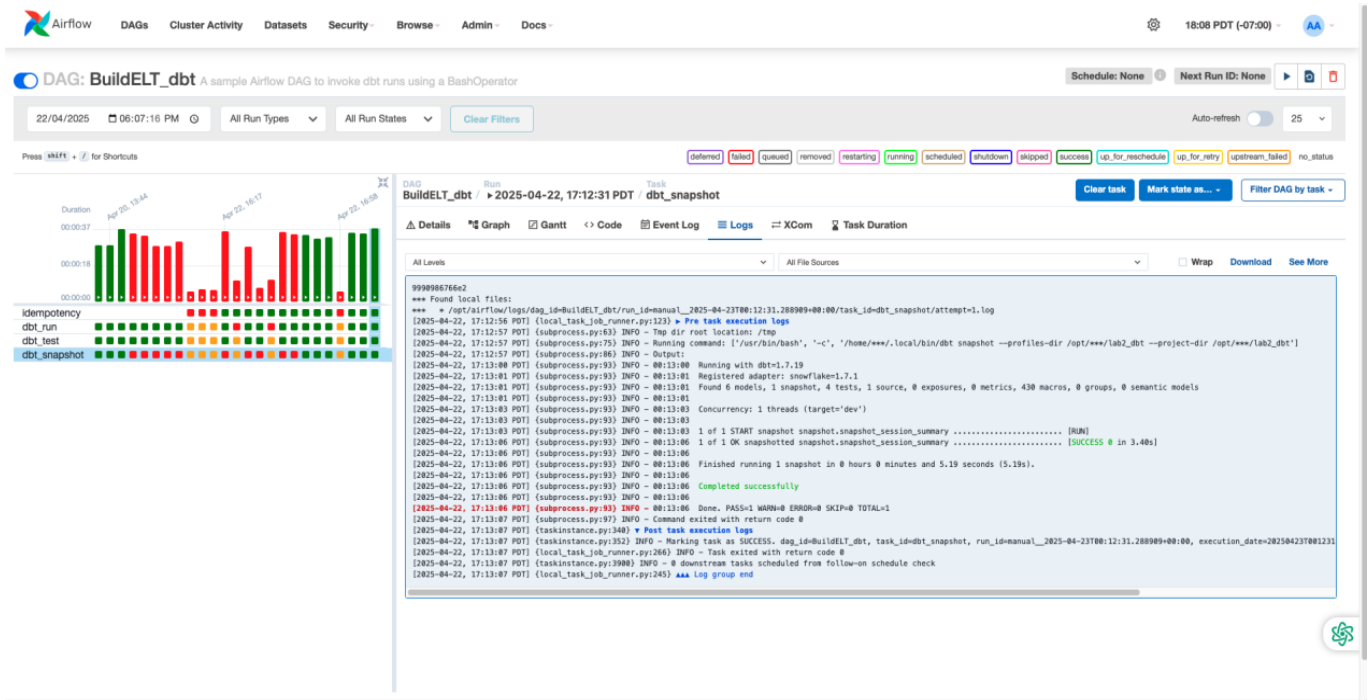
8.2.1 dbt_run



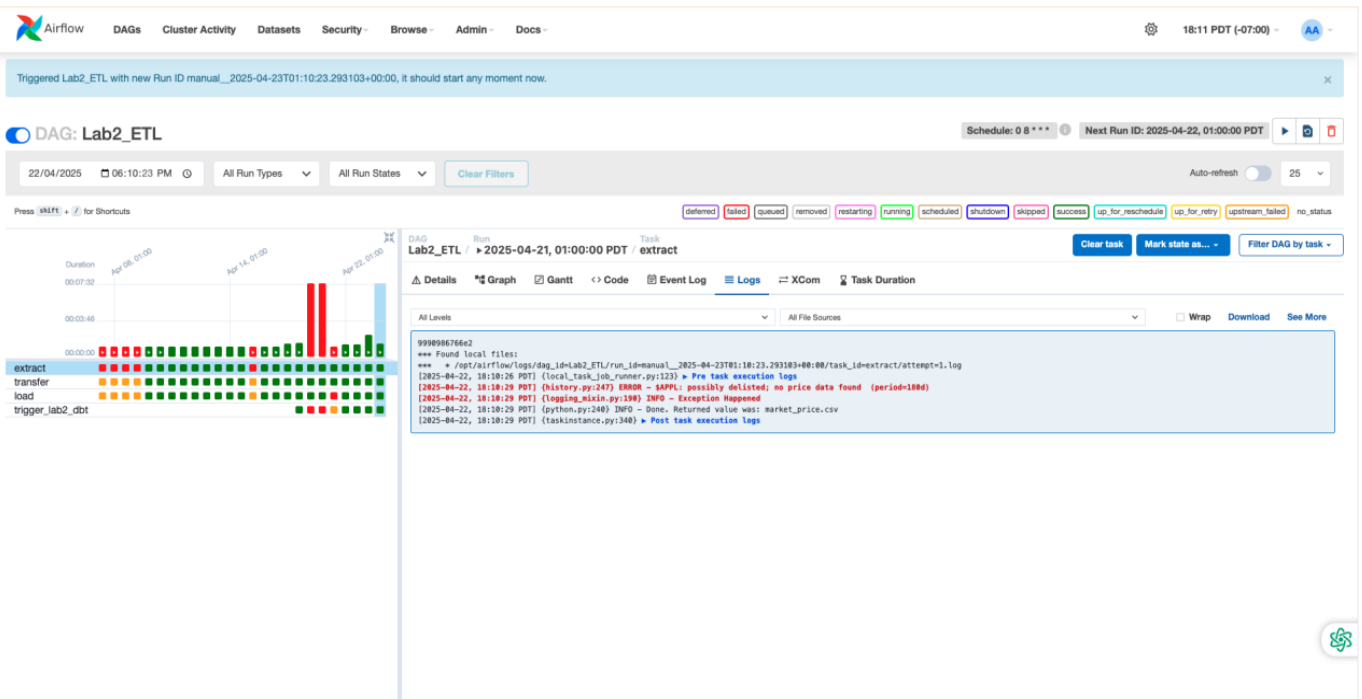
8.2.2 BuildELT_dbt



8.2.3 BuildELT_dbt

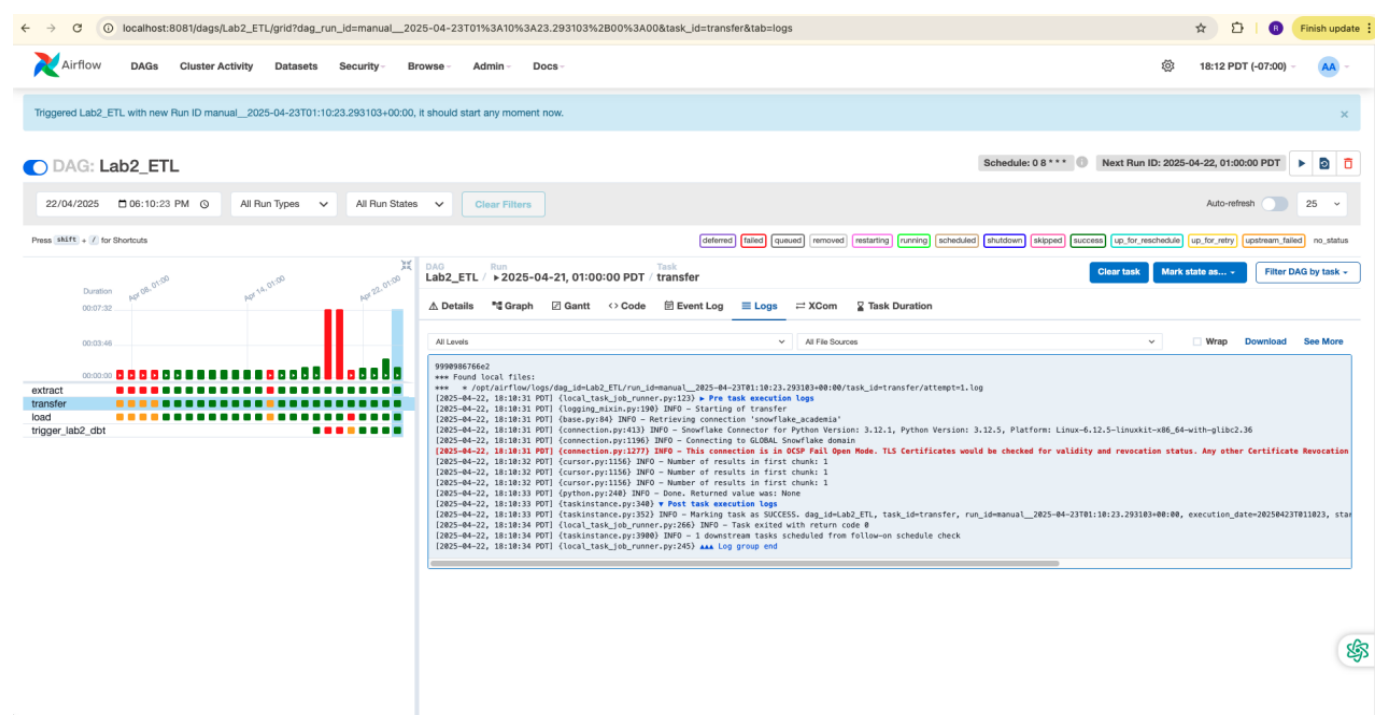


8.2.4 ETL

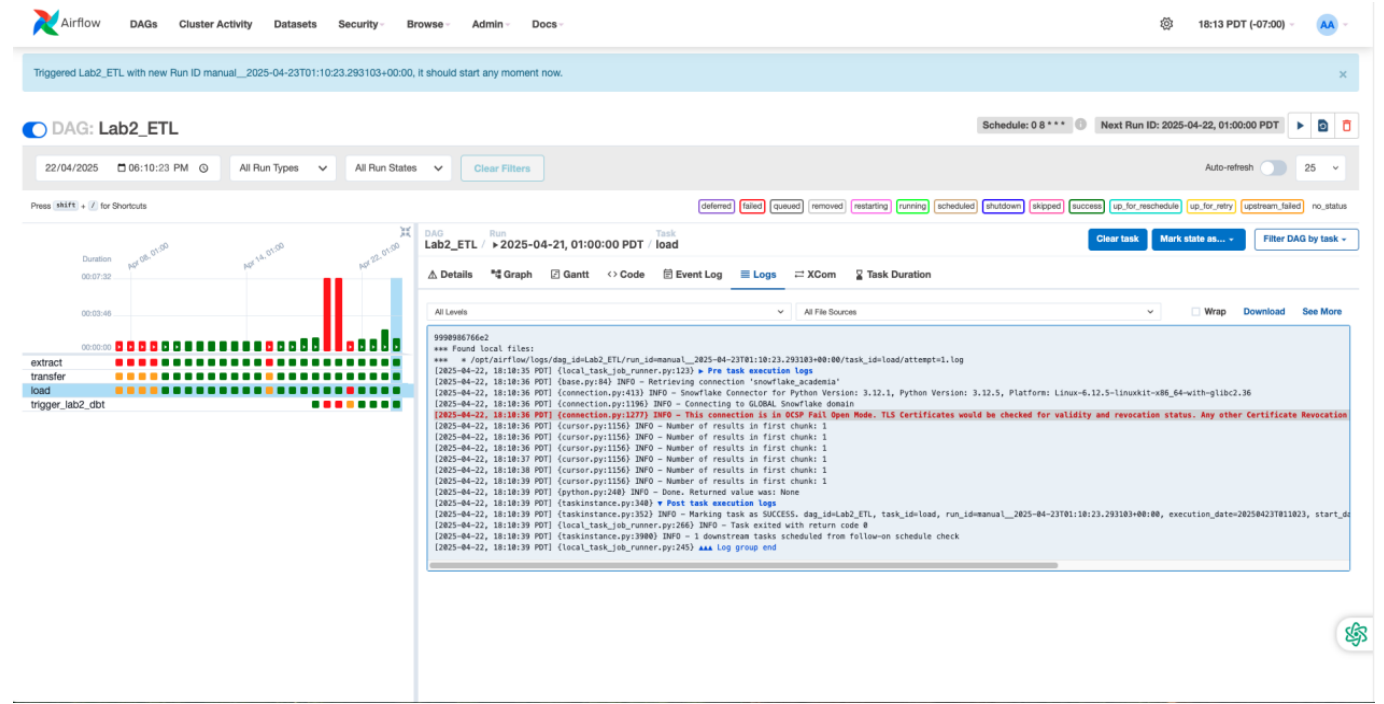


The error encountered occurred because Apple Inc. has been delisted from the yfinance API, making its stock data no longer accessible through the platform.

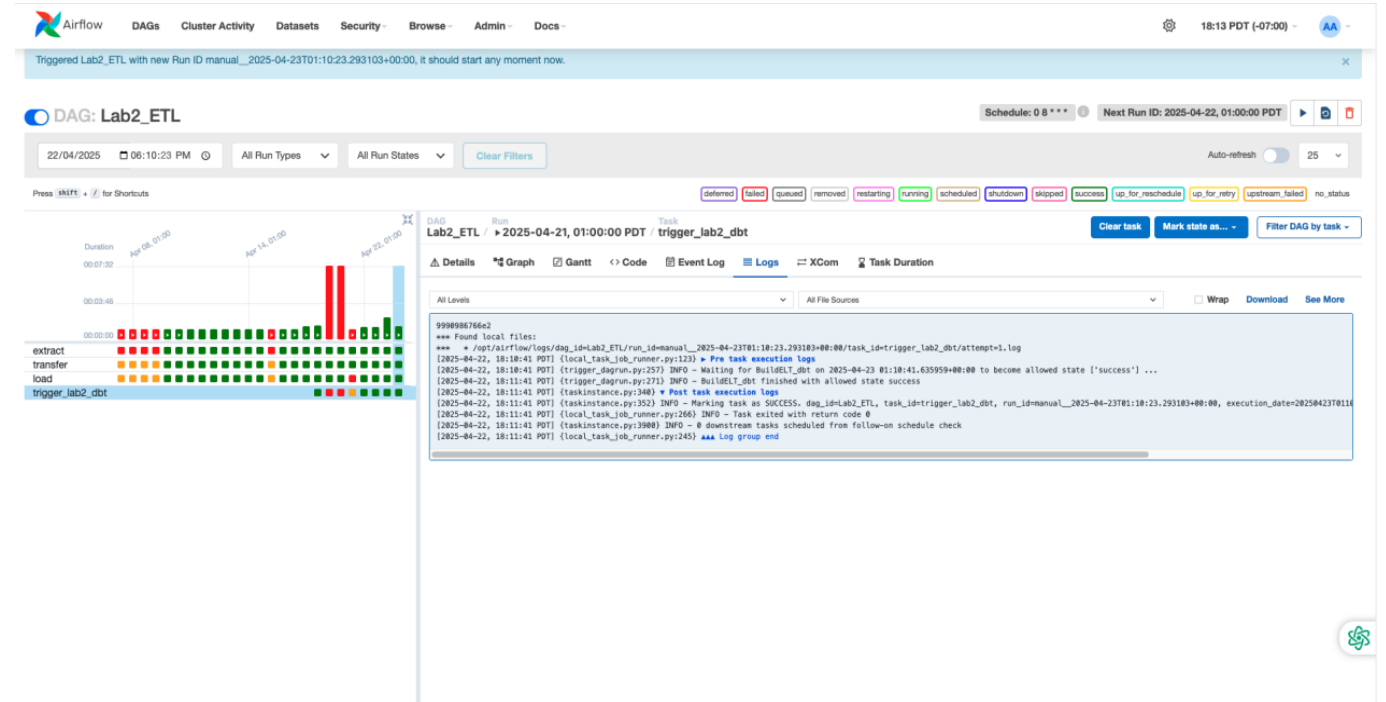
8.2.5 ETL



8.2.6 ETL



8.2.7 ETL



The screenshots above represent the ETL DAG. Upon successful completion, this DAG automatically triggers the ELT DAG, which carries out data transformation and model generation tasks.

9. Conclusion:

This project successfully demonstrates the design and implementation of a complete ELT-based data pipeline for stock market trend analysis. By automating the extraction of historical stock data, transforming it into meaningful analytical models, and visualizing the results through interactive dashboards, the system provides a reliable and scalable solution for financial data exploration. Each component in the architecture — from data ingestion to visualization — is modular and contributes to the overall efficiency of the pipeline. The solution enables data analysts and business users to identify trends, monitor stock behavior, and gain actionable insights in a clear and accessible format. This foundation also allows for future enhancements, such as expanding to additional datasets or integrating forecasting models if needed.

10. Code:

```
create or replace transient table user_db_lobster1.analytics.session_summary as
(
  with _dbtcte_moving_average as (
    select
      date,
      symbol,
      avg(close) over(partition by symbol order by date rows between 9 preceding
and current row) as ma_10
    from user_db_lobster1.raw.market_price
  ),
  _dbtcte_gap_analysis as (
    select
```

```

        date,
        symbol,
        open - lag(close) over(partition by symbol order by date) as gap
    from user_db_lobster1.raw.market_price
),
_dbtcte_volatility as (
    select
        date,
        symbol,
        high - low as volatility
    from user_db_lobster1.raw.market_price
),
_dbtcte_rolling_volatility as (
    with deviation as (
        select
            date,
            symbol,
            (close / lag(close) over(partition by symbol order by date) - 1) as
deviation
            from user_db_lobster1.raw.market_price
        )
    select
        date,
        symbol,
        stddev(deviation) over(partition by symbol order by date rows between 13
preceding and current row) as rolling_volatility
    from deviation
),
_dbtcte_rsi as (
    with delta_close as (
        select
            date,
            symbol,
            close - lag(close) over(partition by symbol order by date) as delta
        from user_db_lobster1.raw.market_price
    ),
    gain_and_loss as (
        select
            date,
            symbol,
            case when delta > 0 then delta else 0 end as gain,
            case when delta < 0 then abs(delta) else 0 end as lose
        from delta_close
    ),
    rs as (
        select
            date,
            symbol,
            avg(gain) over(partition by symbol order by date rows between 13 preceding
and current row) as avg_gain,
            avg(lose) over(partition by symbol order by date rows between 13 preceding
and current row) as avg_loss
        from gain_and_loss
    )

```

```
select
    date,
    symbol,
    round(100 - 100 / (1 + (avg_gain / nullif(avg_loss, 0)))) as rsi
from rs
)
select
    ma.date,
    ma.symbol,
    ma.ma_10,
    gap.gap,
    v.volatility,
    rv.rolling_volatility,
    rsi.rsi
from _dbtcte_moving_average ma
join _dbtcte_gap_analysis gap on ma.date = gap.date and ma.symbol = gap.symbol
join _dbtcte_volatility v on v.date = ma.date and v.symbol = ma.symbol
join _dbtcte_rolling_volatility rv on rv.date = ma.date and rv.symbol =
ma.symbol
join _dbtcte_rsi rsi on rsi.date = ma.date and rsi.symbol = ma.symbol
);
```

References:

<https://pypi.org/project/yfinance>

<https://docs.google.com/document/d/1NDo9FoG5JH-ssRB3G9xp6LDjEXxG9D2Nw4-T9oJdTAE/edit?usp=sharing>

<https://www.clearpointstrategy.com/blog/gap-analysis-template>

[https://www.sigmacomputing.com/blog/calculate-a-moving-average#:~:text=The%20simple%20moving%20average%20\(SMA\)%20is%20calculated%20by%20taking%20the,and%20dividing%20by%20that%20number](https://www.sigmacomputing.com/blog/calculate-a-moving-average#:~:text=The%20simple%20moving%20average%20(SMA)%20is%20calculated%20by%20taking%20the,and%20dividing%20by%20that%20number)