Building an end to end Data Analytics using Snowflake, Airflow, dbt, and a BI tool

Manav Patel
Department of Applied Data Intelligence
San Jose State University
manav.n.patel@sjsu.edu

Parth Patel
Department of Applied Data Intelligence
San Jose State University
parth.patel02@sjsu.edu

Abstract

This lab builds on the Lab 1 stock-price pipeline by transitioning from an ETL-focused approach to a modern ELT workflow centered on downstream analytics. Data ingestion is orchestrated with Apache Airflow, raw price histories are stored in Snowflake, and all business logic is handled through dbt (data build tool). Within dbt, we compute key technical indicators including simple moving averages (SMA), RSI, momentum metrics, daily percentage returns, and rolling volatility and produce validated, version-controlled analytical tables. Airflow coordinates both the ingestion tasks and dbt transformations to maintain reliable, repeatable dataset updates. Analytical outputs are delivered through dashboards built in Apache Superset or Preset/Tableau, offering interactive views of market trends, momentum behavior, and overbought/oversold conditions. The resulting architecture provides a modular and reproducible ELT system that clearly separates raw data acquisition, governed transformation logic, and consumable business insights.

Keywords — ELT, Airflow, dbt, Snowflake, Superset, Time-Series Analysis, RSI, Moving Averages, Bollinger Bands.

I. INTRODUCTION

Modern analytics workflows increasingly adopt an ELT approach loading raw data into the warehouse first and performing transformations later through version-controlled SQL. Unlike traditional end-to-end ETL pipelines, ELT enables greater flexibility, transparency, and maintainability. In this lab, we restructure the stock analytics solution from Lab 1 to follow an ELT pattern by incorporating dbt for transformation logic and testing, while continuing to use Airflow for orchestration and Snowflake for storage and compute. The market dataset remains unchanged (six large-cap tickers, daily OHLCV data, and a 180-day historical window sourced via yfinance) so that the evaluation focuses solely on the architectural shift toward ELT namely dbt models, tests, snapshots, and the resulting BI dashboard.

Our objectives are to:

(i) preserve a dependable and repeatable ingestion pipeline for raw market data
(ii) implement transformation-as-code using dbt supported by built-in data quality tests
(iii) orchestrate ELT tasks in Airflow with explicit dependency management
(iv) provide an accessible BI dashboard (Superset, Preset, or Tableau) that communicates insights effectively to non-technical users.

II. PROBLEM STATEMENT

A lot of analytics projects mix up extraction and transformation steps, which makes pipelines fragile and difficult to track. For Lab 2, where the goal is to build an end-to-end analytics workflow using Snowflake, Airflow, dbt, and a BI tool, we need a setup that cleanly separates each stage and keeps everything easy to manage.

The system we build should:

- Load raw market data into Snowflake in a consistent and repeatable way using Airflow, just like in Lab 1 but now with more focus on reliability.
- Perform all transformation work such as calculating technical indicators or preparing features inside dbt as modular models that include tests and documentation.
- Run the full process on a schedule in Airflow with clear ordering (ingest → transform → test → publish), so each step depends on the one before it.
- Produce clean, governed tables that act as a “single source of truth,” which can then be visualized in a BI tool like Superset, Preset, or Tableau.

These goals match the Lab 2 requirements: Airflow handles the ETL piece with lower emphasis, dbt is responsible for ELT and abstract tables, and a BI dashboard is built to show insights such as moving averages, RSI, and other indicators once the dbt models are running smoothly inside an Airflow DAG.

III. RESEARCH

In an ELT workflow, most of the heavy transformation work is pushed downstream into the data warehouse, which improves scalability and makes it easier to manage governance. Tools like dbt strengthen this approach by adding version control, data lineage, testing, and documentation directly to SQL models, which increases trust in the transformations and makes future changes easier to track. Airflow still plays the central role in coordinating all tasks, handling scheduling, retries, and ordering between ingestion, transformation, and publishing steps. Snowflake's architecture where storage and compute are separated supports reliable, idempotent loading through features like transactional merges and scalable virtual warehouses.

From a research perspective, financial time-series analytics benefit heavily from ELT because many standard indicators (SMA, RSI, Bollinger Bands, momentum, volatility, etc.) can be calculated with efficient SQL window functions. Prior studies show that these technical indicators are commonly used in algorithmic trading and quantitative finance, and the ability to compute them inside the warehouse reduces data movement and improves performance. Recent academic and industry work also highlights how ELT tools like dbt help enforce data quality and reproducibility in analytical workflows, especially when pipelines grow more complex. This makes the ELT pattern well-suited for Lab 2, where we build an end-to-end system combining Snowflake, Airflow, dbt, and a BI tool for clear, testable, and reliable financial analytics.

IV. METHODOLOGY

A. Architecture Overview

Figure 1 shows the updated architecture for Lab 2. In this setup, Airflow collects daily OHLCV data for the selected stock symbols and loads the raw records into Snowflake using safe, transactional MERGE operations. Once the raw layer is refreshed, dbt takes over and converts the data into staging and analytics models, applying tests, documentation, and snapshots as needed. The final transformed tables are then exposed to a BI tool, where dashboards can be created to explore trends, indicators, and time-series behaviors. This end-to-end flow helps keep ingestion, transformation, and visualization clearly separated while still working together as a single pipeline.

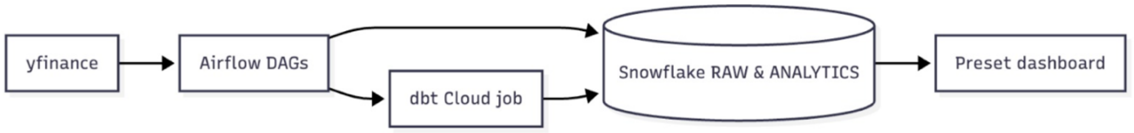
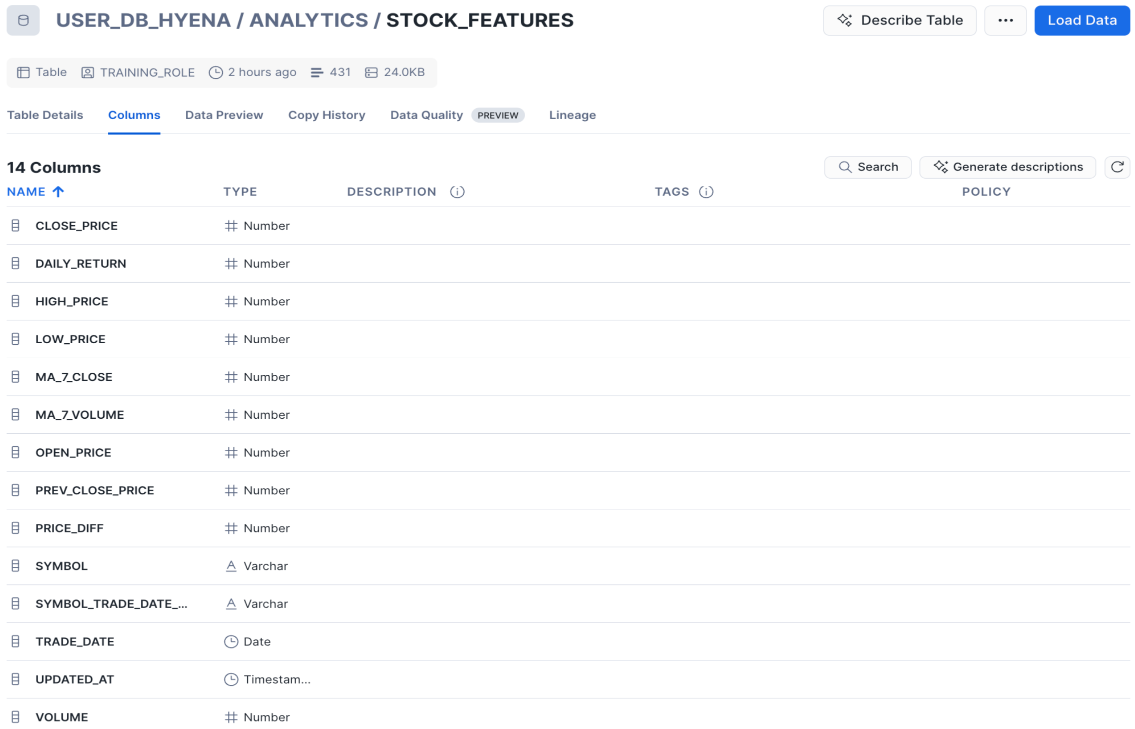


Fig. 1: Lab 2 architecture: Airflow (ingest & schedule) → Snowflake (RAW) → dbt (ELT & tests) → BI (Superset).

B. Dataset

We reuse the Lab 1 dataset and orchestration settings, but restrict the workflow to two symbols :

- **Symbols:** AAPL, NVDA (Airflow Variable: SYMBOLS_JSON)
- **Lookback:** 180 trading days (Airflow Variable: LOOKBACK_DAYS)
- **Fields:** date, open, high, low, close, volume



USER_DB_HYENA / ANALYTICS / STOCK_FEATURES

Table · TRAINING_ROLE · 2 hours ago · 431 · 24.0KB

14 Columns

NAME	TYPE	DESCRIPTION	TAGS	POLICY
CLOSE_PRICE	Number			
DAILY_RETURN	Number			
HIGH_PRICE	Number			
LOW_PRICE	Number			
MA_7_CLOSE	Number			
MA_7_VOLUME	Number			
OPEN_PRICE	Number			
PREV_CLOSE_PRICE	Number			
PRICE_DIFF	Number			
SYMBOL	Varchar			
SYMBOL_TRADE_DATE_...	Varchar			
TRADE_DATE	Date			
UPDATED_AT	Timestam...			
VOLUME	Number			

Fig. 2 : Data Coverage

C. ETL Processes

1. Extraction (Airflow → yfinance)

- Airflow retrieves stock price data (AAPL and NVDA) from the **yfinance API**.
- The extraction function pulls:
  - date
  - open, high, low, close
  - volume
- Lookback period and stock symbols are read from **Airflow Variables**.
- Data is cleaned and converted into a structured DataFrame.

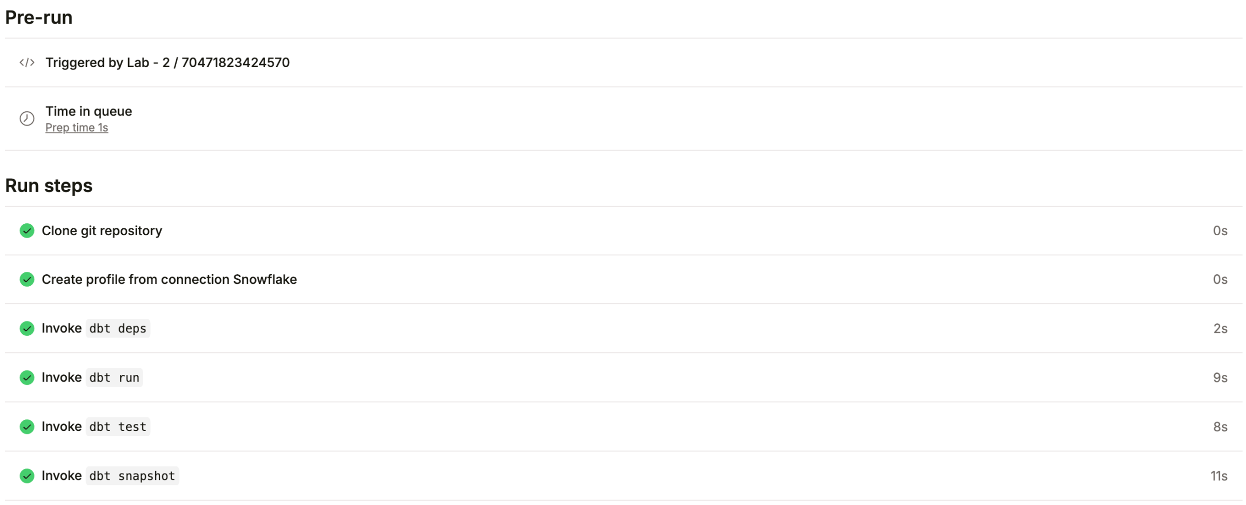
2. Staging & Loading into Snowflake (RAW schema)

- Extracted records are **written to a Snowflake staging table** (`RAW.STOCK_PRICES_STG`).
- The pipeline uses a **transactional MERGE** to:
  - Insert new dates
  - Update existing rows
- Transactions ensure:
  - Idempotency
  - Consistent retry behavior
  - No partial updates

---

3. ELT Transformations (dbt Cloud Job)

- A separate Airflow DAG triggers the `dbt_cloud_stock_pipeline.py` workflow.
- dbt Cloud job runs the following:
  - **Staging models** (clean, standardized data)
  - **Analytics models** (technical indicators: SMA, RSI, returns, volatility)
  - **Snapshots** for slowly changing fields
  - **Schema + data tests** (unique, not-null, accepted values)
- Transformations are managed as code with full version control.



Pre-run

Triggered by Lab - 2 / 70471823424570

Time in queue
Prep time 1s

Run steps

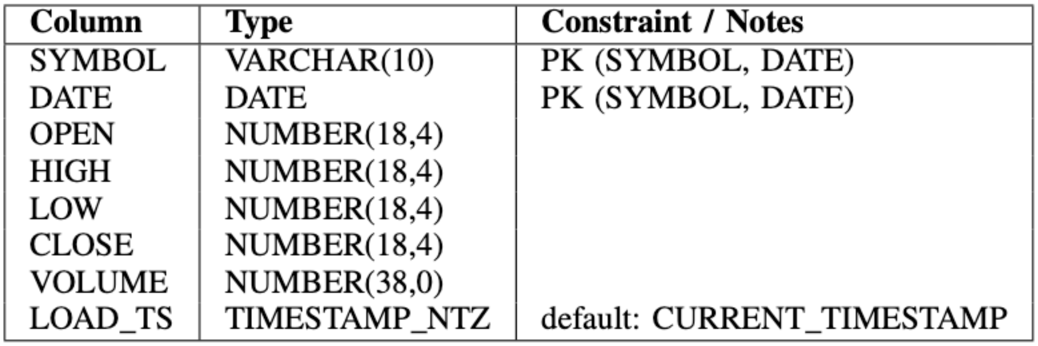
✓ Clone git repository	0s
✓ Create profile from connection Snowflake	0s
✓ Invoke `dbt deps`	2s
✓ Invoke `dbt run`	9s
✓ Invoke `dbt test`	8s
✓ Invoke `dbt snapshot`	11s

Fig. 3

V. SCHEMA AND TABLE SPECIFICATIONS

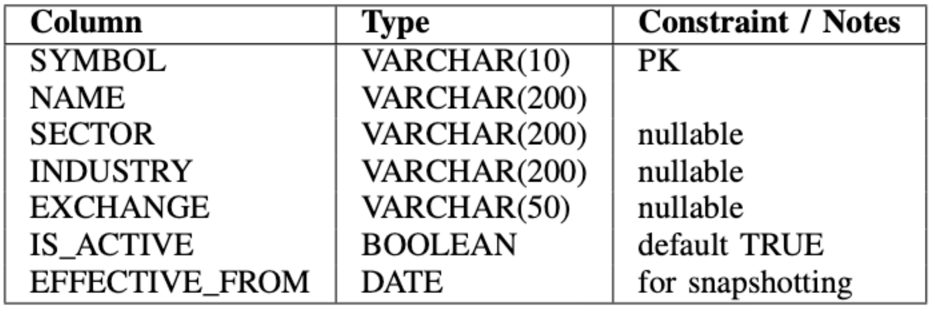
We organize the database into separate schemas based on their role: RAW holds the ingested source data, STAGE contains the cleaned and structured dbt models, and ANALYTICS stores the final user-ready tables.

A. RAW Layer



Column	Type	Constraint / Notes
SYMBOL	VARCHAR(10)	PK (SYMBOL, DATE)
DATE	DATE	PK (SYMBOL, DATE)
OPEN	NUMBER(18,4)	
HIGH	NUMBER(18,4)	
LOW	NUMBER(18,4)	
CLOSE	NUMBER(18,4)	
VOLUME	NUMBER(38,0)	
LOAD_TS	TIMESTAMP_NTZ	default: CURRENT_TIMESTAMP

B. Reference Dimension (dbt)



Column	Type	Constraint / Notes
SYMBOL	VARCHAR(10)	PK
NAME	VARCHAR(200)	
SECTOR	VARCHAR(200)	nullable
INDUSTRY	VARCHAR(200)	nullable
EXCHANGE	VARCHAR(50)	nullable
IS_ACTIVE	BOOLEAN	default TRUE
EFFECTIVE_FROM	DATE	for snapshotting

VII. Orchestration with Airflow and DBT

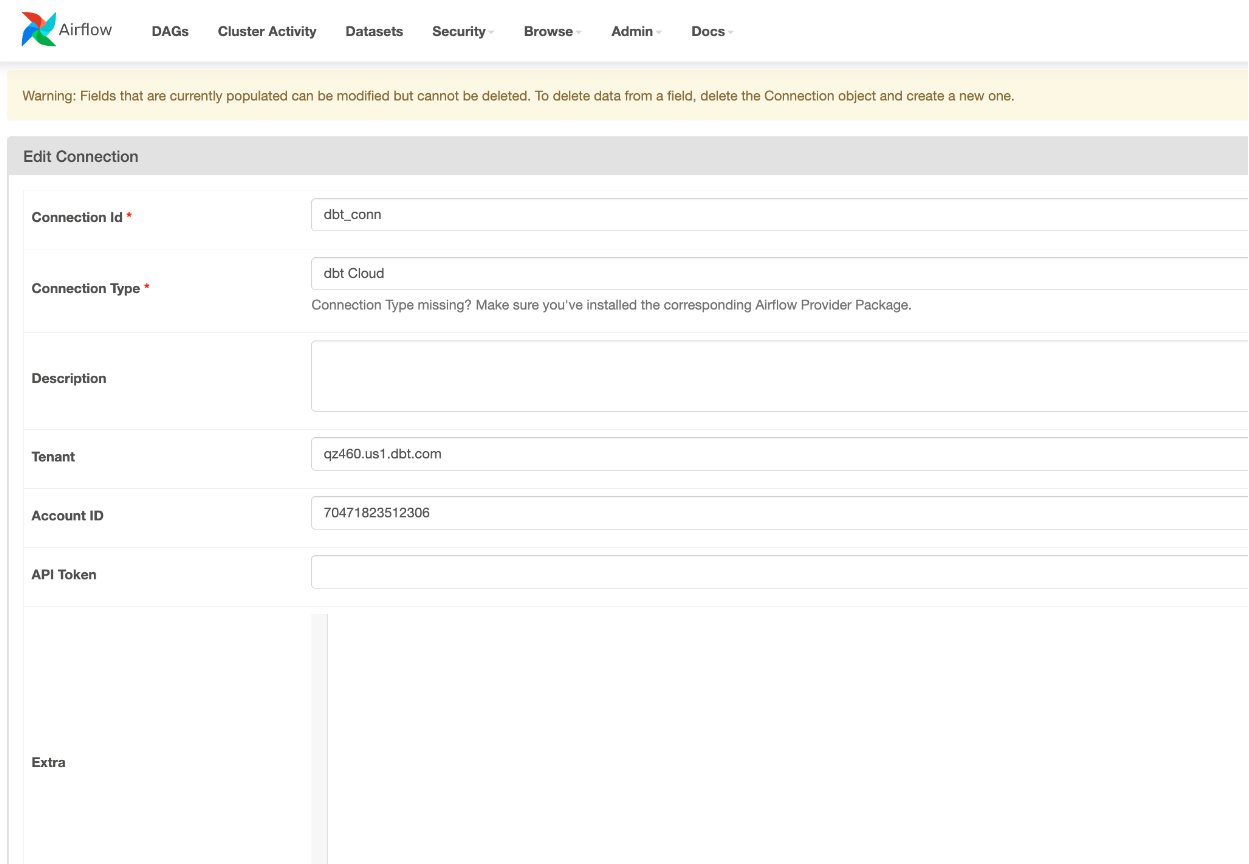
1. Airflow Orchestration

In Airflow, we created a dedicated DAG (dbt_cloud_stock_pipeline) responsible for triggering our dbt Cloud job. The DAG runs on a daily schedule and includes retry logic, logging, and execution tracking.

When the DAG is triggered, Airflow launches a single task:

- **run_dbt_cloud_job** — Calls the dbt Cloud API to kick off the transformation workflow.

The DAG run page shows task duration, execution states, and historical performance. Once the run starts, Airflow monitors the job until dbt Cloud reports completion. If the job fails, Airflow automatically retries based on the configured backoff strategy.



Warning: Fields that are currently populated can be modified but cannot be deleted. To delete data from a field, delete the Connection object and create a new one.

Edit Connection

Connection Id *: dbt_conn

Connection Type *: dbt Cloud

Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.

Description:

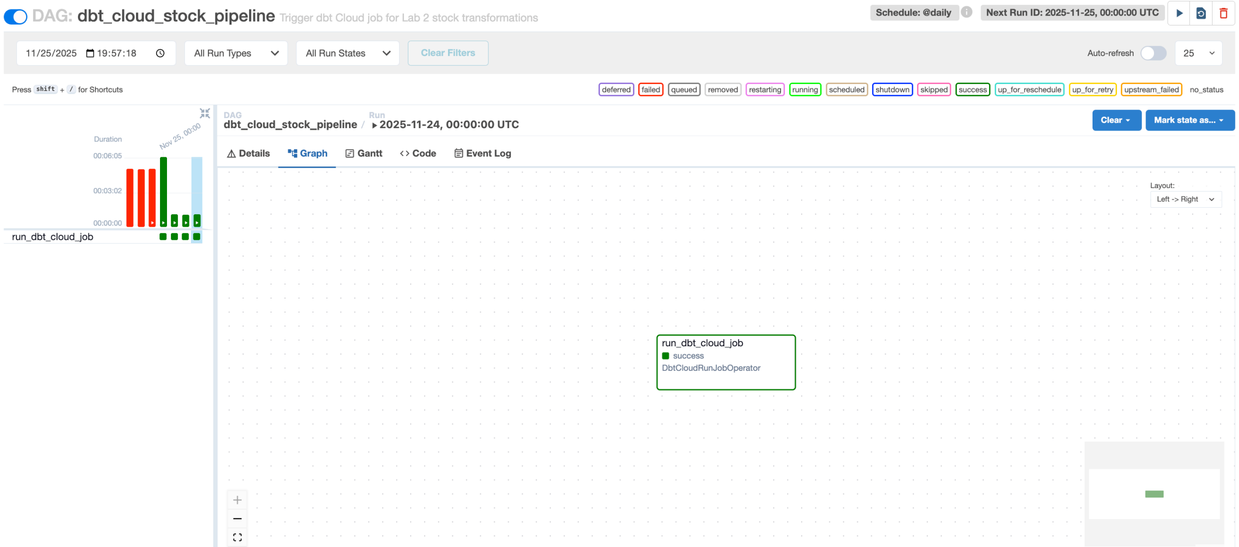
Tenant: qz460.us1.dbt.com

Account ID: 70471823512306

API Token:

Extra:

Fig. 4 Airflow Connection



DAG: dbt_cloud_stock_pipeline Trigger dbt Cloud job for Lab 2 stock transformations

Schedule: @daily

Next Run ID: 2025-11-25, 00:00:00 UTC

11/25/2025 19:57:18

All Run Types

All Run States

Auto-refresh: 25

run_dbt_cloud_job

DAG: dbt_cloud_stock_pipeline / Run: 2025-11-24, 00:00:00 UTC

Layout: Left -> Right

run_dbt_cloud_job
success
DbtCloudRunJobOperator

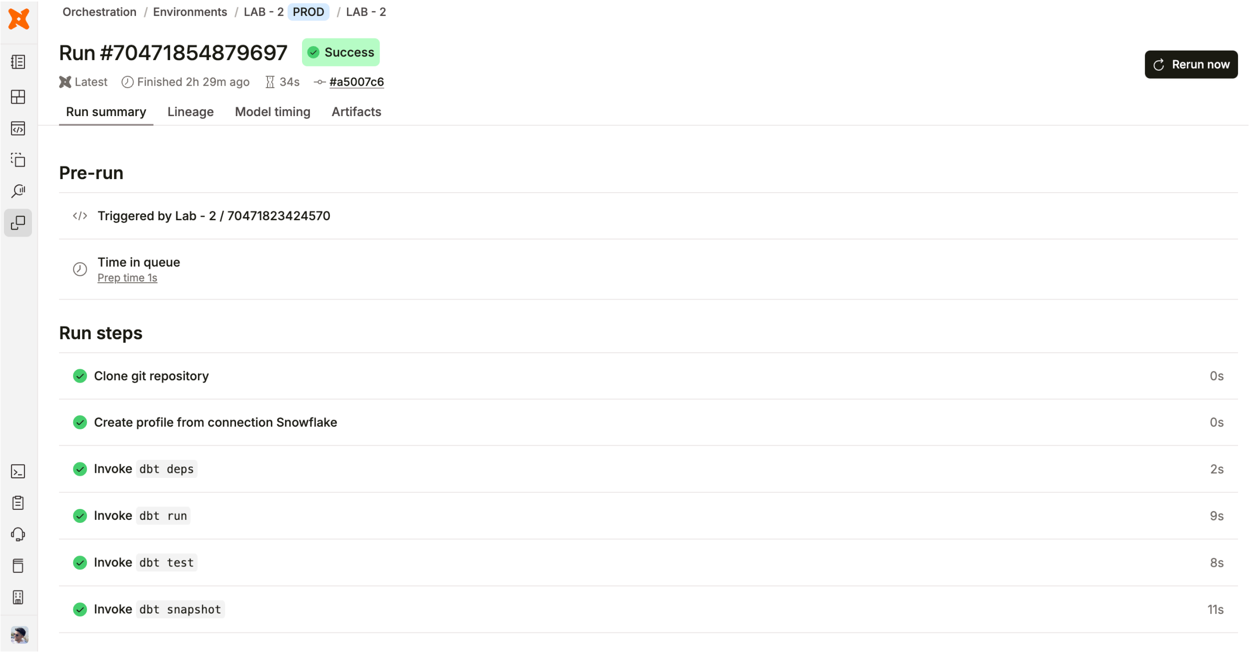
Fig. 5 Airflow Pipeline

2. DBT Cloud Job Execution

Once triggered by Airflow, dbt Cloud handles all transformation logic in the warehouse. The job executes several sequential steps:

1. **Clone git repository** – pulls the latest dbt project code
2. **Create profile from Snowflake connection** – sets warehouse credentials
3. **Invoke dbt deps** – installs dbt packages
4. **Invoke dbt run** – builds staging and analytics models (e.g., stock_features)
5. **Invoke dbt test** – runs schema and data quality tests
6. **Invoke dbt snapshot** – captures slowly changing values for historical tracking

The Run Summary view confirms that all steps executed successfully, along with timing and job metadata.



Run #70471854879697 Success

Latest · Finished 2h 29m ago · 34s · #a5007c6

Pre-run

Triggered by Lab - 2 / 70471823424570

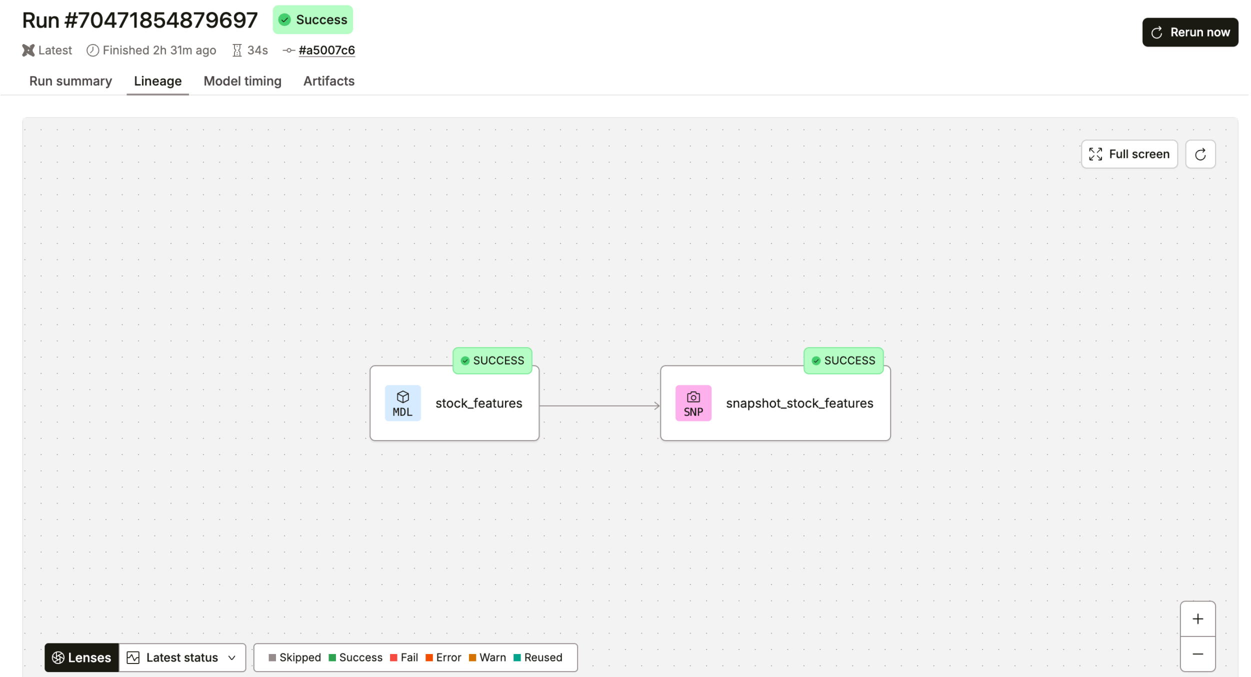
Time in queue
Prep time 1s

Run steps

✓ Clone git repository	0s
✓ Create profile from connection Snowflake	0s
✓ Invoke `dbt deps`	2s
✓ Invoke `dbt run`	9s
✓ Invoke `dbt test`	8s
✓ Invoke `dbt snapshot`	11s

Fig. 6 DBT

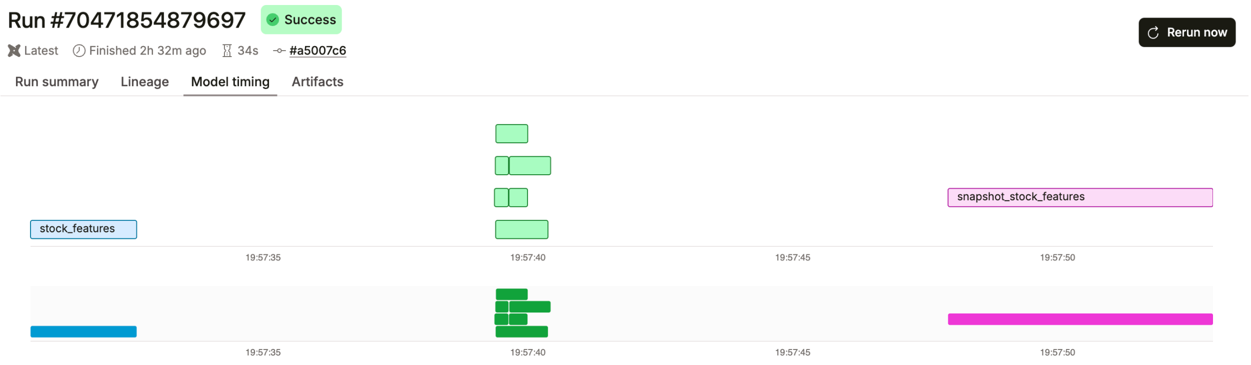
3. Model Lineage and Execution Timing



Run #70471854879697 Success

Latest · Finished 2h 31m ago · 34s · #a5007c6

Fig. 7 Model Lineage



Run #70471854879697 Success

Latest · Finished 2h 32m ago · 34s · #a5007c6

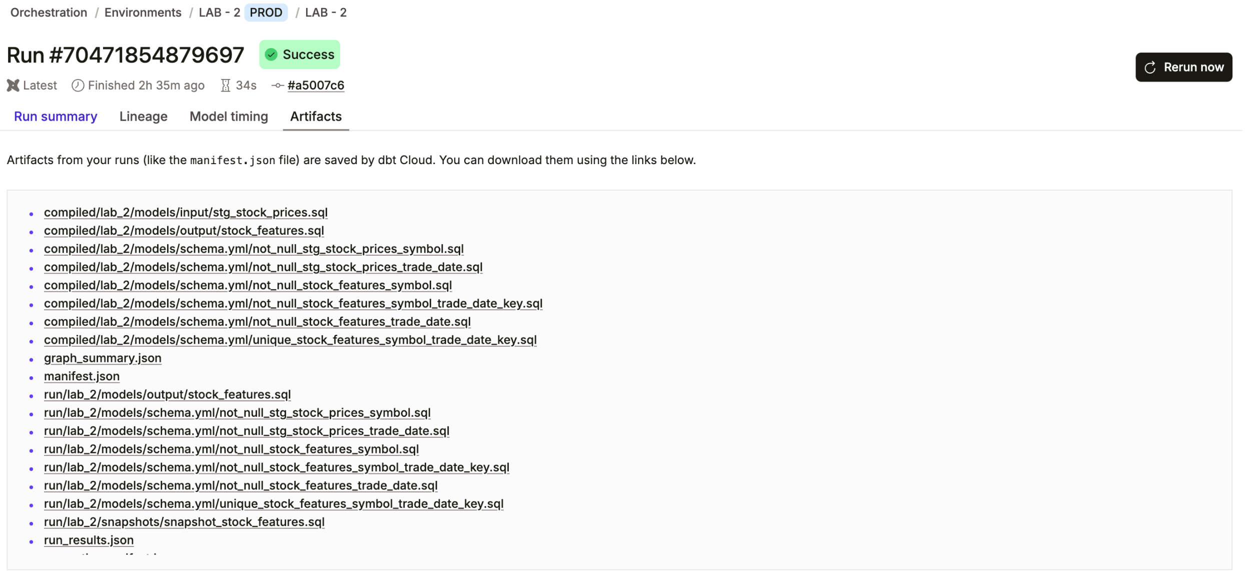
Fig. 8 Execution Timing

4. Artifacts and Metadata

dbt Cloud generates a set of output files (artifacts) after every run, including:

- **manifest.json** — full representation of the dbt project
- **run_results.json** — detailed model performance information
- Compiled SQL for all models and tests
- Snapshots and test results

These artifacts can be downloaded for debugging, auditing, or further analysis.



Run #70471854879697 Success

Latest Finished 2h 35m ago 34s #a5007c6

Artifacts from your runs (like the `manifest.json` file) are saved by dbt Cloud. You can download them using the links below.

- compiled/lab_2/models/input/stg_stock_prices.sql
- compiled/lab_2/models/output/stock_features.sql
- compiled/lab_2/models/schema.yml/not_null_stg_stock_prices_symbol.sql
- compiled/lab_2/models/schema.yml/not_null_stg_stock_prices_trade_date.sql
- compiled/lab_2/models/schema.yml/not_null_stock_features_symbol.sql
- compiled/lab_2/models/schema.yml/not_null_stock_features_symbol_trade_date_key.sql
- compiled/lab_2/models/schema.yml/not_null_stock_features_trade_date.sql
- compiled/lab_2/models/schema.yml/unique_stock_features_symbol_trade_date_key.sql
- graph_summary.json
- manifest.json
- run/lab_2/models/output/stock_features.sql
- run/lab_2/models/schema.yml/not_null_stg_stock_prices_symbol.sql
- run/lab_2/models/schema.yml/not_null_stg_stock_prices_trade_date.sql
- run/lab_2/models/schema.yml/not_null_stock_features_symbol.sql
- run/lab_2/models/schema.yml/not_null_stock_features_symbol_trade_date_key.sql
- run/lab_2/models/schema.yml/not_null_stock_features_trade_date.sql
- run/lab_2/models/schema.yml/unique_stock_features_symbol_trade_date_key.sql
- run/lab_2/snapshots/snapshot_stock_features.sql
- run_results.json

Fig. 9 Artifacts

5. End-to-End Automation

By combining Airflow (orchestration) and dbt Cloud (transformation), the pipeline becomes fully automated:

1. Airflow ingests raw AAPL and NVDA prices into Snowflake
2. Airflow triggers dbt Cloud
3. dbt applies all transformations, tests, and snapshots
4. Cleaned analytics tables are published to the BI layer

This ensures the ELT process runs consistently, with strong observability at each step.

VIII. BI VISUALIZATION (PRESET)

We publish ANALYTICS.STOCK_FEATURES as a dataset in the BI tool and assemble a dashboard :

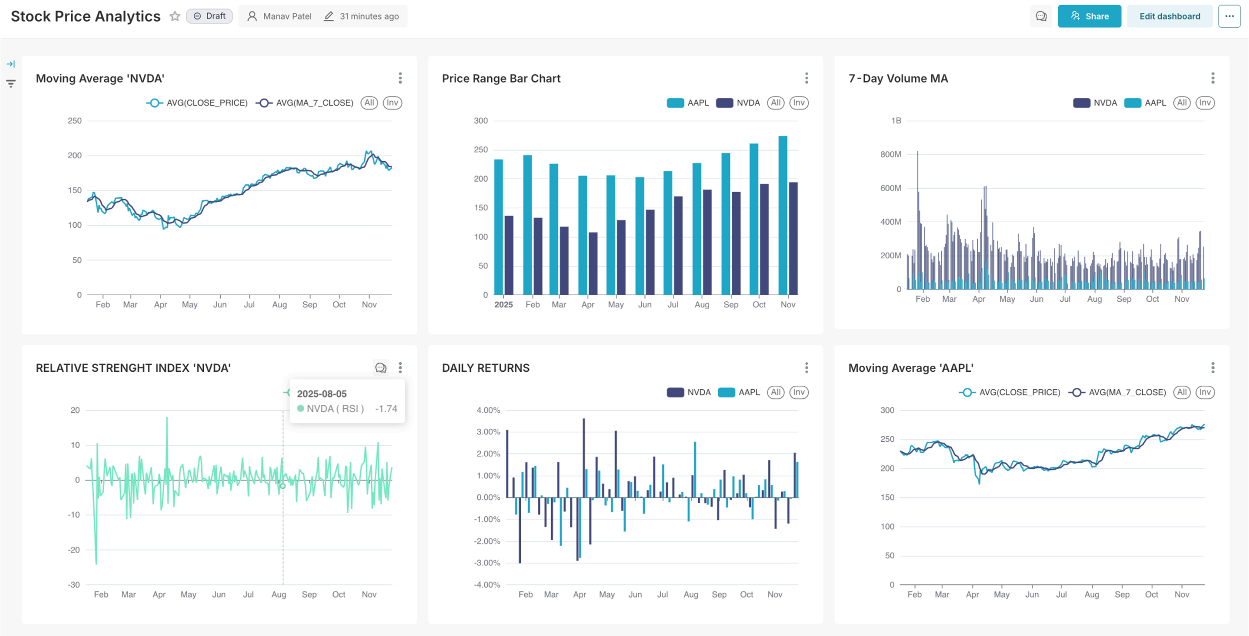
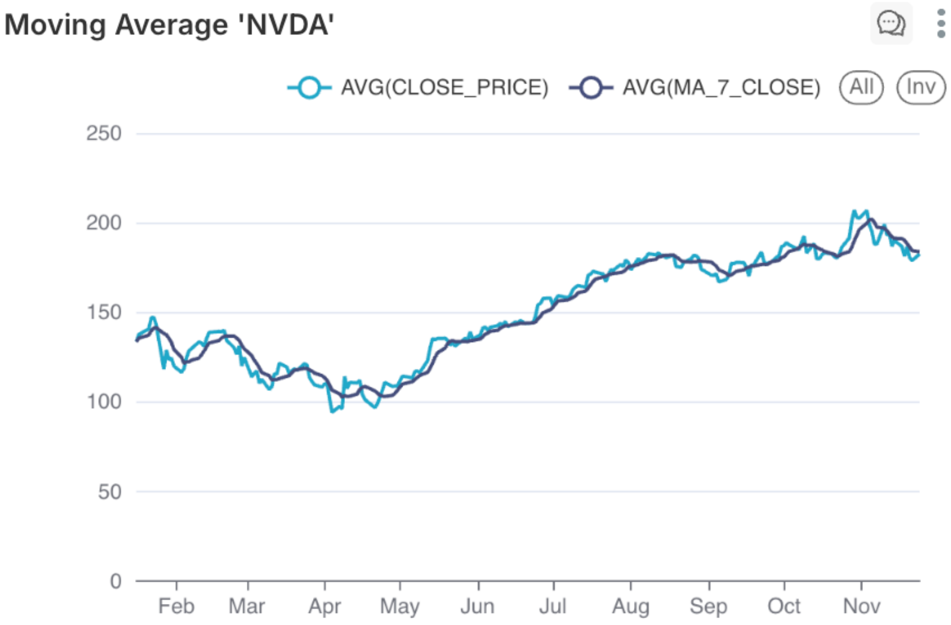
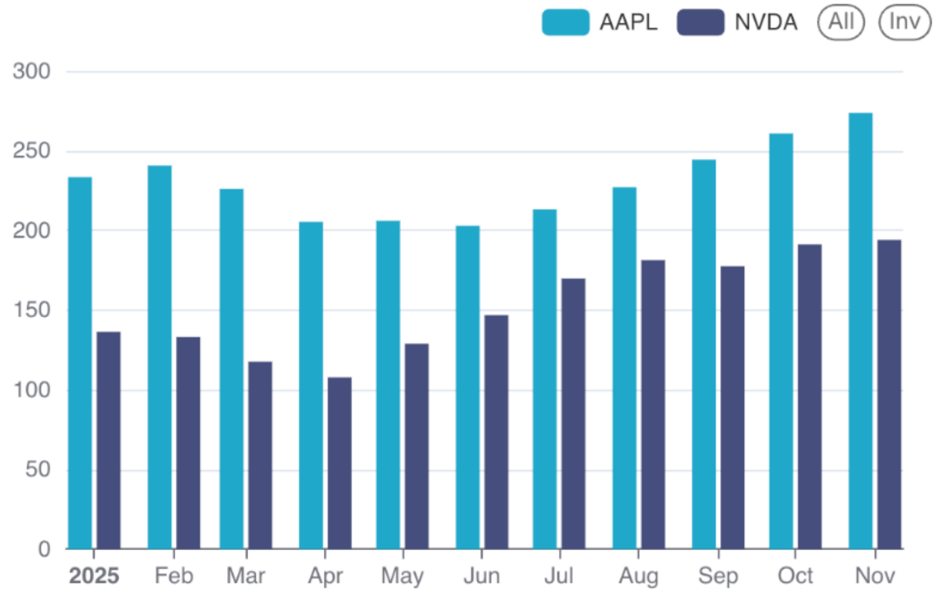


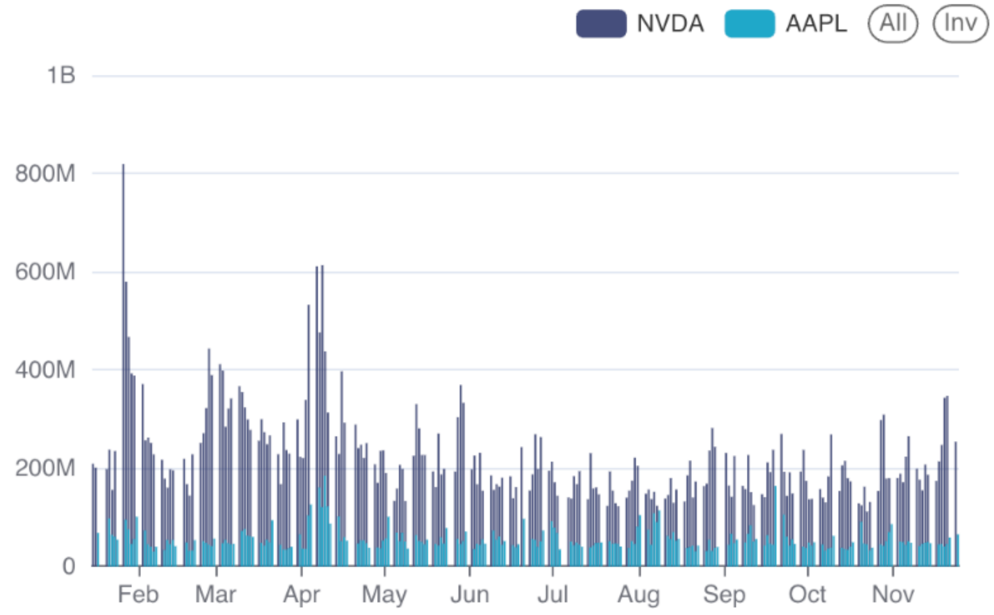
Fig. 10 Dashboard



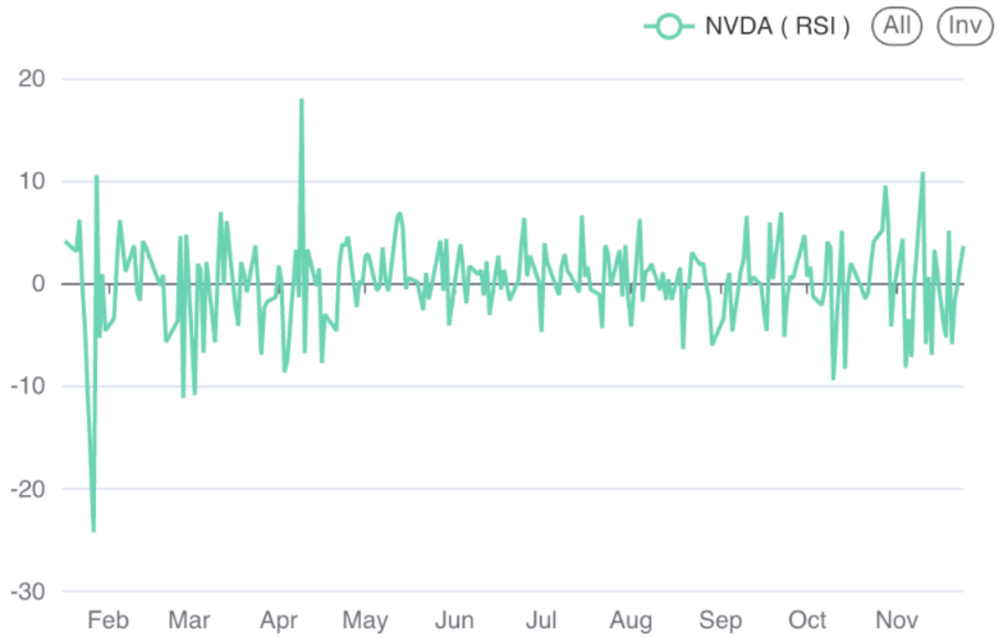
Price Range Bar Chart



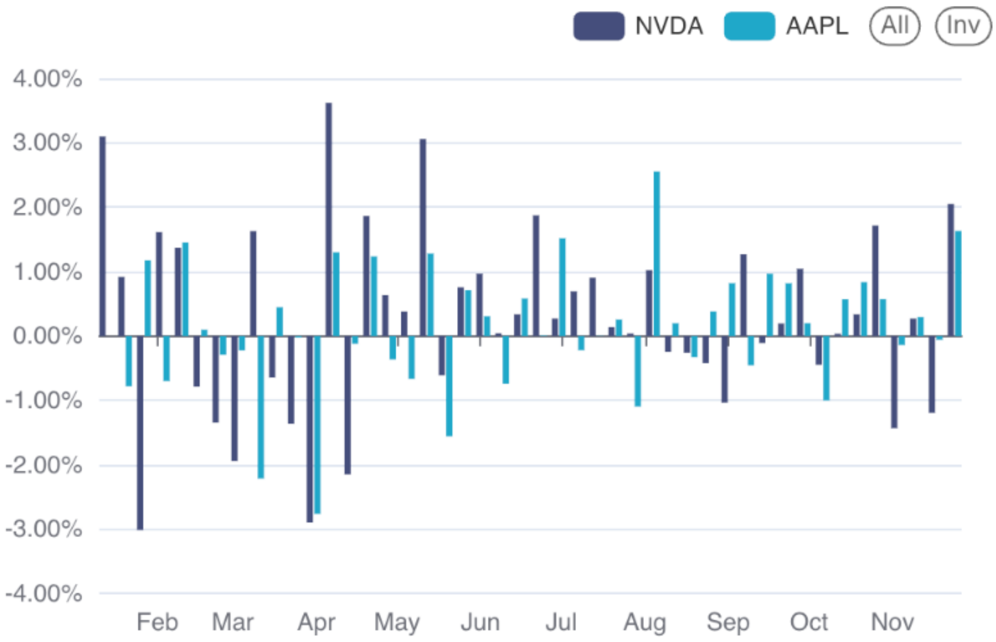
7 - Day Volume MA

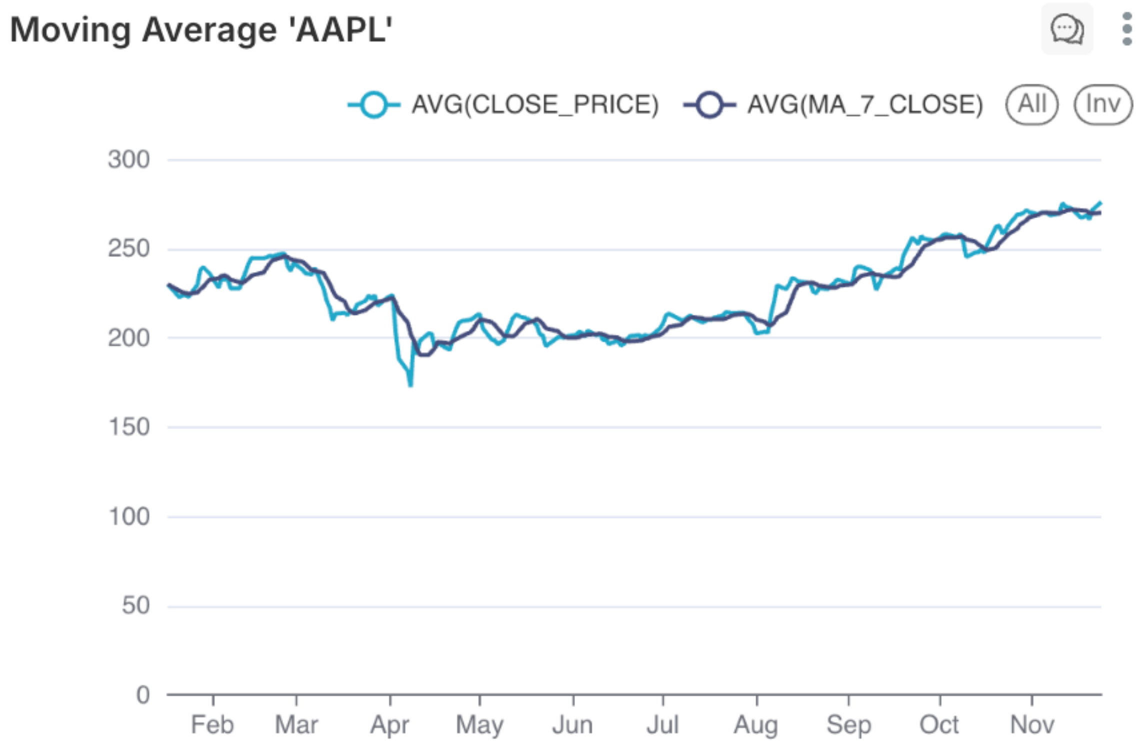


RELATIVE STRENGHT INDEX 'NVDA'



DAILY RETURNS





IX. DISCUSSION

Separating ingestion from transformation made the pipeline easier to understand and audit. dbt's built-in tests caught data issues early on, such as missing price or volume values. Using incremental models reduced compute costs and improved run efficiency. Moving all transformation logic into dbt, rather than relying only on Python ETL, made the codebase more transparent and reusable for other tools. Superset made it simple to build quick, shareable dashboards directly on top of the warehouse tables.

X. CONCLUSION

In this lab, we built a complete ELT-driven analytics pipeline using Airflow for workflow management, Snowflake for storage and compute, dbt for structured and tested transformations, and Superset for visualization. The resulting system is reliable, repeatable, and easy to extend.

Possible Next Steps

- Expand the dataset to include additional tickers or even new asset types.
- Introduce more advanced indicators—such as EMA, MACD, and factor or risk metrics—implemented through recursive CTEs or Snowflake UDFs.
- Add a proper development workflow for dbt with dev/staging/production environments and CI integration.
- Set up freshness checks, anomaly detection, and automated alerts using Airflow SLAs or Superset notifications.
- Optionally incorporate the forecasting work from Lab 1 as an additional analytics layer, and visualize prediction intervals within the dashboard.

REFERENCES

[1] GeeksforGeeks, “How to use yfinance API with Python.” Accessed: Mar. 2, 2025. [Online]. Available: https://www.geeksforgeeks.org/

[2] Snowflake Documentation, “Snowflake Developer Guide.” Accessed: Mar. 2, 2025. [Online]. Available: https://docs.snowflake.com/en/developer

[3] Apache Airflow, “Apache Airflow Documentation.” Accessed: Mar. 2, 2025. [Online]. Available: https://airflow.apache.org/docs/

[4] Alpha Vantage, “Alpha Vantage API Documentation.” Accessed: Mar. 2, 2025. [Online]. Available: https://www.alphavantage.co/documentation/

[5] AWS, “ARIMA Forecasting Recipe.” Accessed: Mar. 2, 2025. [Online]. Available: https://docs.aws.amazon.com/forecast/latest/dg/

aws-forecast-recipe-arima.html [11] De Bie, T., Demaeyer, J., & others. (2023). Data Quality in Data Warehousing. *IEEE Data Engineering Bulletin*, 46(2), 12–28.

[6] Cuzzocrea, A., & Psaila, G. (2021). Data Warehouse and Data Lake Technologies. *ACM Computing Surveys*, 54(3), 1–40.

[7] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

GITHUB LINK

DATA-226_Lab-2