An End-to-End Data Analytics Pipeline - Stock Performance Visualization

By Anshika Goel, Sai Chaitanya Munagala, San Jose State University
Applied Data Science Department
22/04/2025

Abstract

Building scalable and efficient data pipelines is essential for analytics and business intelligence in the era of data-driven industries. This project explicitly aims and demonstrates an end-to-end ELT [Extract, Load, Transform] pipeline architecture using various tech stacks like dbt, Apache Airflow, Looker, and Snowflake. The ETL pipeline extracts historical stock price data from Apple, Google, and Microsoft using yFinance, loads it into Snowflake, transforms the data using dbt models, and presents insights through Looker visualizations. This report provides an in-depth technical breakdown of each component and how they interact to form a cohesive and modular analytical system. It showcases the importance of automation in the data world and generates quick, easy efficient insights to the business needs that transform the business decisions.

1. Introduction

A. Background

The vast exponential growth from various sources has created an urge for robust and efficient scalable engineering workflows. Traditionally, ETL systems face challenges when dealing with the enormous amount of data, popularly called big data, and complex schemas. The modern ELT approach is the solution, it leverages cloud data warehouses for scalability and performance. In this project, we aim to design and deploy an automated ELT approach that not only ingests data efficiently but can also perform some transformation close to the storage layer. This minimizes I/O bottlenecks and supports efficient downstream analytics. This shift also aligns with the growing demand for near-real-time insights and reduced data latency in decision-making. By leveraging tools like dbt and Airflow, ELT workflows become more modular, testable, and maintainable. Such a setup always lays good groundwork and foundation for advanced analytical use cases.

B. Problem Statement

In the current data-driven world, almost every organization deals with disparate datasets that need to be ingested, cleaned, enriched, and visualized in near real-time. The absence of automated orchestration, scalable layers, manual transformations, and integrated reporting tools not only leads to manual errors but also results in slow reporting cycles. The lack of visibility is also an issue that needs to be addressed. This project aims to challenge the automation of stock market data analytics using open-source orchestration tools and scalable cloud data platforms. In the end, it also visualises the performance obtained from the transformation to see the insights.

C. **Objectives**

1. Extract stock data for Apple, Google, and Microsoft for the last 180 days using the *Yahoo Finance API*.
2. Build a modular and maintainable *ELT* pipeline using *Apache Airflow*.
3. Use *dbt* to define, test, and version control transformation logic.
4. Visualize meaningful insights via *Looker*.
5. Timely demonstrates *reproducibility*, *modularity,* and *extensibility* in the pipeline.

D. Requirements and Specifications

This project requires leveraging Apache Airflow, dbt, Snowflake, and Looker. The system is required to extract, load, and transform stock price data and visualize key metrics using a BI tool. The technical requirements and specifications for these are classified as below:

1. Airflow: Two DAGS, one for ETL and the other for ELT, to trigger dbt transformations.
2. dbt: Models created for metric calculations. Profiles.yml file for Snowflake connectivity.
3. Snowflake: A Warehouse and 2 Schemas for populating tables.
4. Looker: For visualizing data and drawing patterns as per the business requirements.

---

2. System Architecture

A. Technology Stack

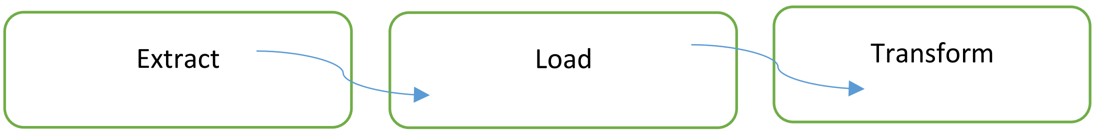
1. *Apache Airflow*: A Workflow orchestration tool to manage and automate data pipelines.
2. *Snowflake*: Cloud-based data warehouse for storing and processing raw and curated data.
3. *dbt* [Data Build Tool]: Transformation tool that manages completely through *SQL*-based transformations with lineage, testing, and documentation.
4. *Looker*: A Data visualization platform to create dashboards and charts from Snowflake tables.
5. *yFinance*: Open-source Python library to fetch historical stock prices and indicators.
6. *Docker*: Containerization technology used to encapsulate the entire stack.
7. *Python:* Scripting language used for *DAG* and API integration.

Apache Airflow   dbt   snowflake   Looker

B. Architecture Overview

The architecture follows an ELT pattern:

1. *Extract:* yFinance fetches historical data.
2. *Load:* Data is loaded into Snowflake's schema using Snowflake Connector.
3. *Transform:* dbt runs transformation models on top of tables to create analytics-ready tables in the analytics schema.
4. Visualize: *Looker* connects to Snowflake and visualizes curated tables.



This architecture keeps the pipeline modular, scalable and easy to maintain. Each of the data extraction, transformation and visualization can be monitored and improved independently. Tools like dbt and Looker make it easier to collaborate and deliver insights faster. This setup ensures the analytics layer always reflects the latest, most reliable data

---

3. Implementation Phase and Overall System diagram

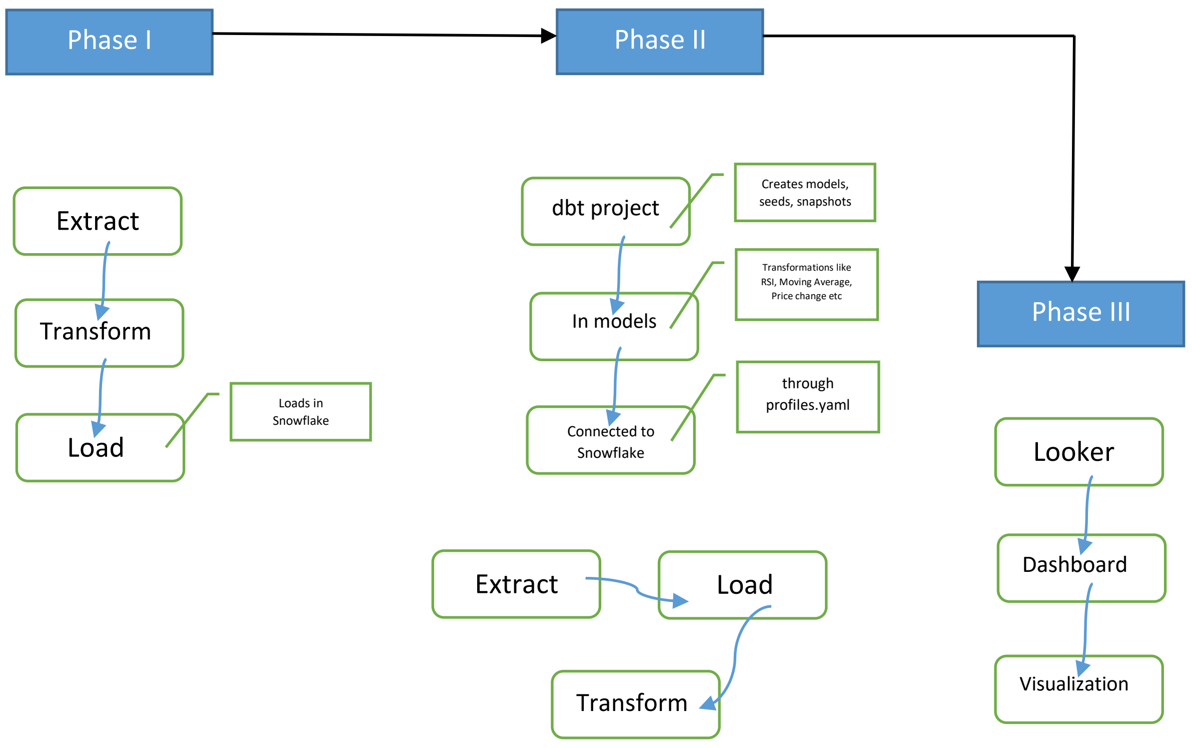
A. Implementation Phase

The implementation of the project was executed in three core stages: They are:

1. *ETL (Extract, Transform, Load)* using *Apache Airflow*,
2. *ELT modelling* using *dbt (data build tool)*, and the automation through *ELT* DAG.
3. *Data visualization* using *Looker*.

Each of these above stages is strategically planned and well implemented to ensure smooth, efficient data flow and to maintain transformation accuracy, and also for analytical insight generation. Below is a concise narrative of how the implementation phase unfolded at each of its respective stages.

B. Overall System Diagram:



First, to begin with, the environment has to be set up; the entire infrastructure for running the workflow was initially set up using Docker. A pre-configured Airflow Docker setup was utilized, where services like the Airflow webserver, scheduler, worker, and triggerer were defined within the *docker-compose.yml* file. The file was also updated to ensure all the necessary Python packages, such as snowflake-connector-python, dbt-snowflake, and apache-airflow-providers-snowflake, and at a later stage, Looker, a powerful dashboard for Business Intelligence, is required for installation. This Looker setup will later help us connect with Snowflake at a later stage to run visuals and draw insights. Once this was in place, the Airflow stack was launched using the *docker compose up* command.

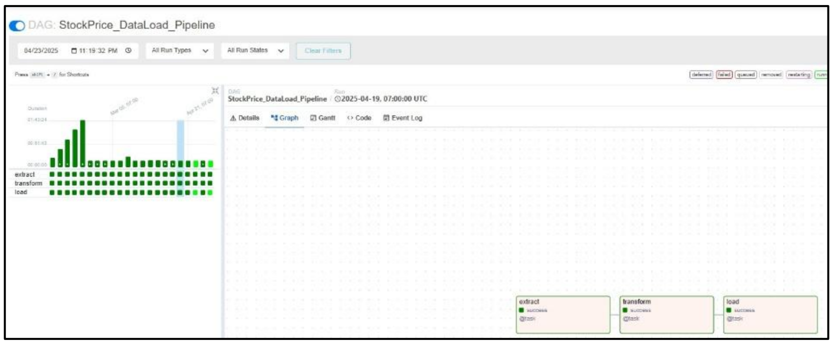
This setup provides a solid foundation to orchestrate our pipeline in a clean and efficient way. The above system diagram at every necessary layer ensures SQL transactions with a try/except method, validates tests at every dbt transformation stage, and checks by maintaining idempotency at every required step.

4. Methodology

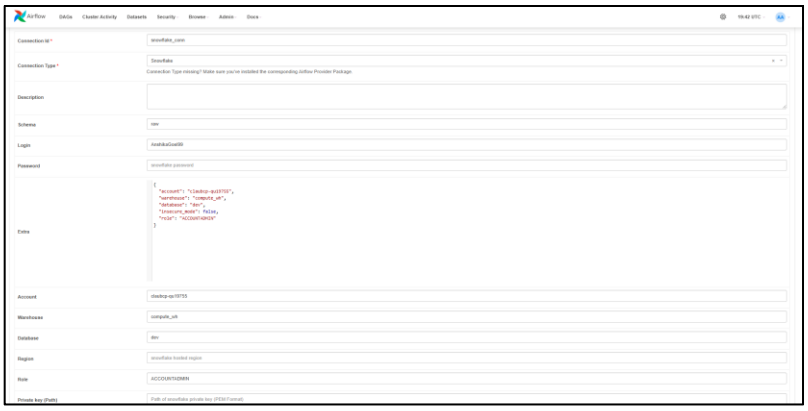
As discussed in the Implementation phase, the methodology will be categorised into three crucial phases, namely, Phase I, Phase II, and Phase III

A. Phase I

The first phase of data processing focused on ETL. A custom DAG *(StockPrice_DataLoad_Pipeline.py)* was created to automate the extraction of stock price data by using the *yfinance*. The DAG was configured to fetch the last 180 days of data for selected stock symbols [i.e., Apple, Google, Microsoft]. This data was fetched using a PythonOperator through three tasks, namely, extract, transform, and load within the DAG, and transformed into a structured format suitable for Snowflake.



[i] Screenshot Airflow ETL - DAG with its respective tasks



Connection Id *	snowflake_conn
Connection Type *	Snowflake Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Schema	raw
Login	AnshikaGoel99
Password	snowflake password
Extra	{ "account": "claubcp-qu19755", "warehouse": "compute_wh", "database": "dev", "insecure_mode": false, "role": "ACCOUNTADMIN" }
Account	claubcp-qu19755
Warehouse	compute_wh
Database	dev
Region	snowflake hosted region
Role	ACCOUNTADMIN
Private key (Path)	Path of snowflake private key (PEM Format)

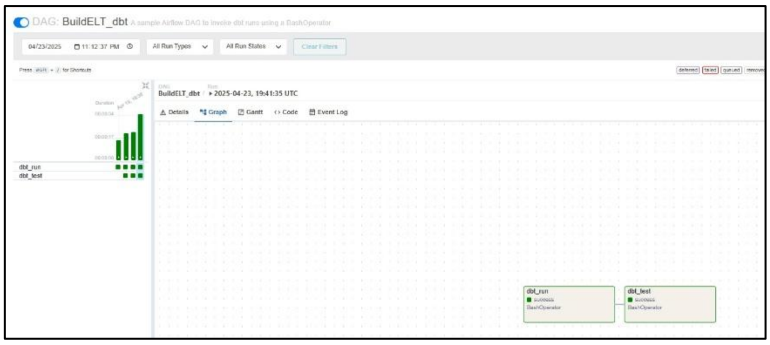
[i] Screenshot Airflow Connections

Subsequently, this data was inserted into Snowflake. Before insertion, a Snowflake connection was established within the *Airflow UI*, supplying account credentials, *database, schema, warehouse,* and *role*. This connection enabled the DAG to push data securely and seamlessly from Airflow into the Snowflake data warehouse. This is now officially called a *data pipeline* focused on *ETL*, insured best with SQL transactions and idempotency checks at every necessary stage.

B. Phase II

The Second phase, an important phase where the project will go through its transition, called the ELT phase, using dbt. For this, a fresh DBT project was initialized locally, which generates the necessary folder structure and configuration files. The connection to Snowflake was defined in *profiles.yml*, where authentication parameters like database names and other metadata were provided. With the dbt project configured, model files were written under the */models directory*. These SQL-based models transform the raw stock data into analytical insights. For instance, *stock_summary.sql* was designed to calculate technical indicators like the moving averages, the Relative Strength Index, the Simple moving averages, gain/loss ratios, and more. These transformations help to derive cleaner and more insightful data for analysis.

To execute the dbt transformations within the pipeline, a second Airflow DAG *(BuiltELT_dbt.py)* was created. This DAG used a BashOperator to run dbt run directly from the Airflow scheduler container. An alternate manual option to trigger these transformations is to prompt *dbt run*, *dbt test* through command prompts, but since this project focuses completely on automation without manual intervention, we choose to create a Dag *(BuiltELT_dbt.py)* as mentioned earlier, which primarily focuses on triggering the transformations. When triggered, this DAG compiles and runs all *SQL* models defined in the dbt project, thereby materializing new analytical tables into the *analytics* schema of Snowflake. This represents the complete automation of the ELT process from transformation design to execution. This, now from here, will be called a complete *end-to-end ELT Automated Pipeline.*



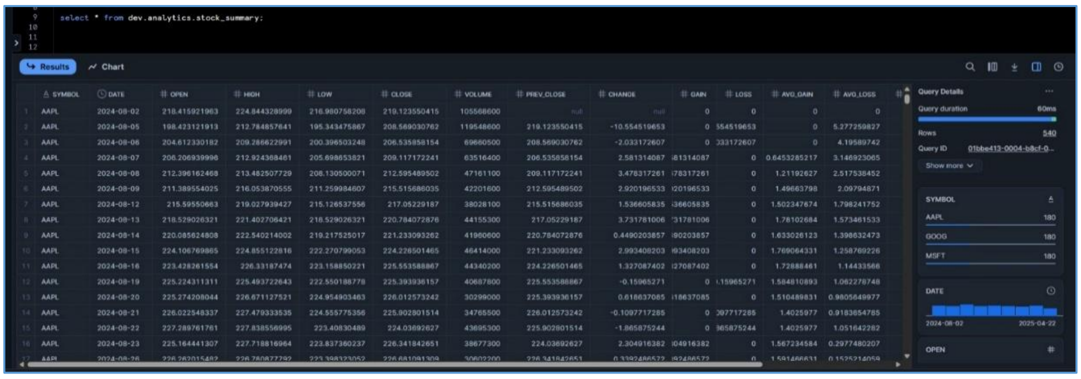
[i] Screenshot Airflow ELT DAG and its respective tasks

[i] Detailed Table Structure:

This is the output table generated via dbt transformations. It is built on top of raw stock data and includes analytical metrics such as RSI and Moving Averages, etc.

Field Name	Data Type	Description	Constraints
Date	Date	Date of Stock Metrics	Not Null, Primary Key along with Symbol
Symbol	String	Ticker symbol of the stock	Not Null, Primary Key, along with Date
Open	Float	Price of the stock at the beginning	Nullable
Close	Float	Price of the stock on the closing day	Nullable
High	Float	Highest Price of the day	Nullable
Low	Float	Lowest Price of the day	Nullable
Volume	Float	Size of the stock	Nullable
Previous close	Float	Previous day closing Price	Nullable
Change	Float	[Closing Price – Previous price]	Nullable
Gain	Float	Positive change in price, else 0	Nullable
Loss	Float	Negative change in price, else 0	Nullable
Avg_Gain	Float	15-day rolling average of gains	Nullable
Avg_Loss	Float	15-day rolling average of losses	Nullable
RSI	Float	15-day RSI with avg_gain and avg_loss	Nullable
Moving_avg	Float	20-day rolling average Closing price	Nullable

[i] Table in Snowflake



```
select * from dev.analytics.stock_summary;
```

Results | Chart

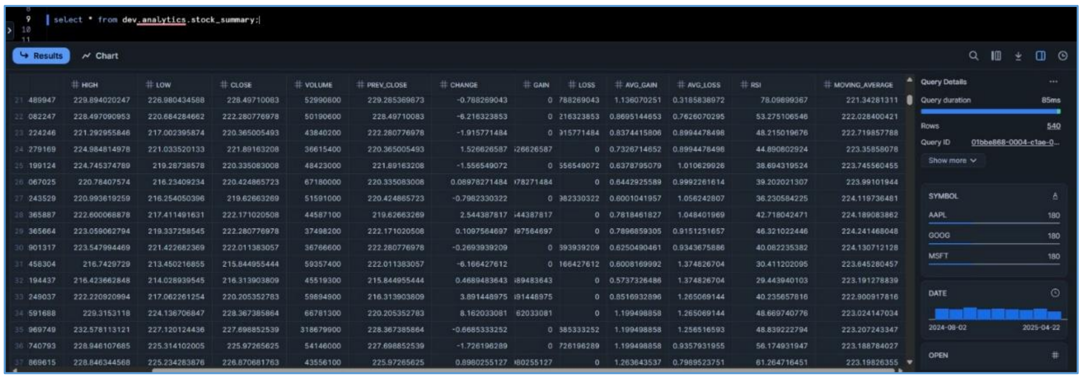
	SYMBOL	DATE	OPEN	HIGH	LOW	CLOSE	VOLUME	PREV_CLOSE	CHANGE	GAIN	LOSS	AVG_GAIN	AVG_LOSS
1	AAPL	2024-08-02	218.415921963	224.844328999	216.980758208	219.123550415	105568600	null	null	0	0	0	0
2	AAPL	2024-08-05	198.423121913	212.784857641	195.343475867	208.569030762	119548600	219.123550415	-10.554519653	0	554519653	0	5.277259827
3	AAPL	2024-08-06	204.612330182	209.286622991	200.396503248	206.535858154	69660500	208.569030762	-2.033172607	0	033172607	0	4.19589742
4	AAPL	2024-08-07	206.206939996	212.924368461	205.698653821	209.117172241	63516400	206.535858154	2.581314087	581314087	0	0.6453285217	3.146923065
5	AAPL	2024-08-08	212.396162468	213.482507729	208.130500071	212.595489502	47161100	209.117172241	3.478317261	478317261	0	1.21192627	2.517538452
6	AAPL	2024-08-09	211.389554025	216.053870555	211.259984607	215.515686035	42201600	212.595489502	2.920196533	920196533	0	1.49663798	2.09794871
7	AAPL	2024-08-12	215.59550663	219.027939427	215.126537556	217.05229187	38028100	215.515686035	1.536605835	536605835	0	1.502347674	1.798241752
8	AAPL	2024-08-13	218.529026321	221.402706421	218.529026321	220.784072876	44155300	217.05229187	3.731781006	731781006	0	1.78102684	1.573461533
9	AAPL	2024-08-14	220.085624808	222.540214002	219.217525017	221.233093262	41960600	220.784072876	0.4490203857	490203857	0	1.633026123	1.398632473
10	AAPL	2024-08-15	224.106769865	224.855122816	222.270799053	224.226501465	46414000	221.233093262	2.993408203	993408203	0	1.769064331	1.258769226
11	AAPL	2024-08-16	223.428261554	226.33187474	223.158850221	225.553588867	44340200	224.226501465	1.327087402	327087402	0	1.72888461	1.14433566
12	AAPL	2024-08-19	225.224311311	225.493722643	222.550188778	225.393936157	40687800	225.553588867	-0.15965271	0	.15965271	1.584810893	1.062278748
13	AAPL	2024-08-20	225.274208044	226.671127521	224.954903463	226.012573242	30299000	225.393936157	0.618637085	618637085	0	1.510489831	0.9805649977
14	AAPL	2024-08-21	226.022548337	227.479333535	224.555775356	225.902801514	34765500	226.012573242	-0.1097717285	0	097717285	1.4025977	0.9183654785
15	AAPL	2024-08-22	227.289761761	227.838556995	223.40830489	224.03692627	43695300	225.902801514	-1.865875244	0	865875244	1.4025977	1.051642282
16	AAPL	2024-08-23	225.164441307	227.718816964	223.837360237	226.341842651	38677300	224.03692627	2.304916382	304916382	0	1.567234584	0.2977480207
17	AAPL	2024-08-26	226.262015482	226.780877792	223.398323052	226.681091309	30602200	226.341842651	0.3392486572	392486572	0	1.591466631	0.1525214059

Query Details
Query duration 60ms
Rows 540
Query ID 01bbe413-0004-b8cf-0...
Show more

SYMBOL	
AAPL	180
GOOG	180
MSFT	180

DATE
2024-08-02 2025-04-22

OPEN



```
select * from dev.analytics.stock_summary;
```

Results | Chart

		HIGH	LOW	CLOSE	VOLUME	PREV_CLOSE	CHANGE	GAIN	LOSS	AVG_GAIN	AVG_LOSS	RSI	MOVING_AVERAGE
21	489947	229.894020247	226.980434588	228.49710083	52990800	229.285369873	-0.788269043	0	788269043	1.136070251	0.3185838972	78.09899367	221.34281311
22	082247	228.497090953	220.684284662	222.280776978	50190600	228.49710083	-6.216323853	0	216323853	0.8695144653	0.7626070295	53.275106546	222.028400421
23	224246	221.292955846	217.002395874	220.365005493	43840200	222.280776978	-1.915771484	0	915771484	0.8374415806	0.8994478498	48.215019676	222.719857788
24	279169	224.984814978	221.033520133	221.89163208	36615400	220.365005493	1.526626587	26626587	0	0.7326714652	0.8994478498	44.890802924	223.35858078
25	199124	224.745374789	219.28738578	220.335083008	48423000	221.89163208	-1.556549072	0	556549072	0.6378795079	1.010629926	38.694319524	223.745560455
26	067025	220.78407574	216.23409234	220.424865723	67180000	220.335083008	0.08978271484	78271484	0	0.6442925589	0.9992261614	39.202021307	223.99101944
27	243529	220.993619259	216.254050396	219.62663269	51591000	220.424865723	-0.7982330322	0	982330322	0.6001041957	1.056242807	36.230584225	224.119736481
28	365887	222.600068878	217.411491631	222.171020508	44587100	219.62663269	2.544387817	44387817	0	0.7818461827	1.048401969	42.718042471	224.189083862
29	365664	223.059062794	219.337258545	222.280776978	37498200	222.171020508	0.1097564697	97564697	0	0.7896859305	0.9151251657	46.321022446	224.241468048
30	901317	223.547994469	221.422682369	222.011383057	36766600	222.280776978	-0.2693939209	0	693939209	0.6250490461	0.9343675886	40.082235382	224.130712128
31	458304	216.7429729	213.450216855	215.844955444	59357400	222.011383057	-6.166427612	0	166427612	0.6008169992	1.374826704	30.411202095	223.645280457
32	194437	216.423662848	214.028939545	216.313903809	45519300	215.844955444	0.4689483643	89483643	0	0.5737326486	1.374826704	29.443940103	223.191278839
33	249037	222.220920994	217.062261254	220.205352783	59894900	216.313903809	3.891448975	91448975	0	0.8516932896	1.265069144	40.235657816	222.900917816
34	591688	229.3153118	224.136706847	228.367385864	66781300	220.205352783	8.162033081	62033081	0	1.199498858	1.265069144	48.669740776	223.024147034
35	969749	232.578113121	227.120124436	227.698852539	318679900	228.367385864	-0.6685333252	0	685333252	1.199498858	1.256516593	48.839222794	223.207243347
36	740793	228.946107685	225.314102005	225.97265625	54146000	227.698852539	-1.726196289	0	726196289	1.199498858	0.9357931955	56.174931947	223.188784027
37	869615	228.846344568	225.234283876	226.870681763	43556100	225.97265625	0.8980255127	80255127	0	1.263643537	0.7989523751	61.264716451	223.19826355

Query Details
Query duration 85ms
Rows 540
Query ID 01bbe868-0004-c1ae-0...
Show more

SYMBOL	
AAPL	180
GOOG	180
MSFT	180

DATE
2024-08-02 2025-04-22

OPEN

E. Phase III

After building and loading all analytical models into *Snowflake*, the final phase involves connecting the data warehouse to a BI visualization tool, which is popularly called *Looker*.

Looker, a modern open-source business intelligence tool that enables users to explore and visualize data through interactive dashboards. It is used by analysts to interactively explore metrics across selected fields and across periods. It not only supports datasets to load but also supports SQL-based querying, chart creation, and filter integration, along with some metric options. The drag-and-drop option is easy to use, making it ideal for analysing large datasets quickly and securely using a web-based interface. Within the Looker web interface, a new database connection to Snowflake was created using the same credentials previously used in Airflow and dbt. This brings us one step away from drawing trends, predicting patterns, and visualising insights from the charts.

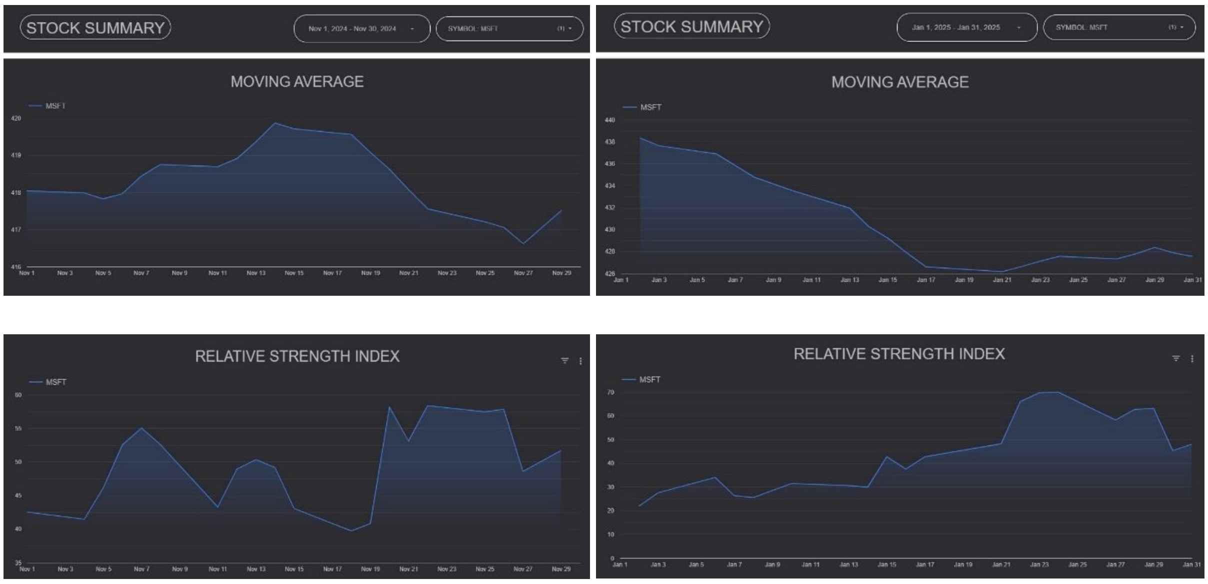
Once the connection in Looker is verified, the table *analytics.stock_summary* was added as a dataset for visualization. Using Looker's drag-and-drop chart builder, multiple insightful visualizations were created.

---

5. Visualization

The ETL, ELT with dbt, along with Snowflake and Looker, has brought us to the final destination where all businesses wanted to be. This is where we start drawing patterns, predicting trends, and analysing future potential. As mentioned earlier, the data pipeline focused its automation on three companies, namely, Microsoft, Apple, and Google. Let us now unfold the hidden insights for efficient business purposes.

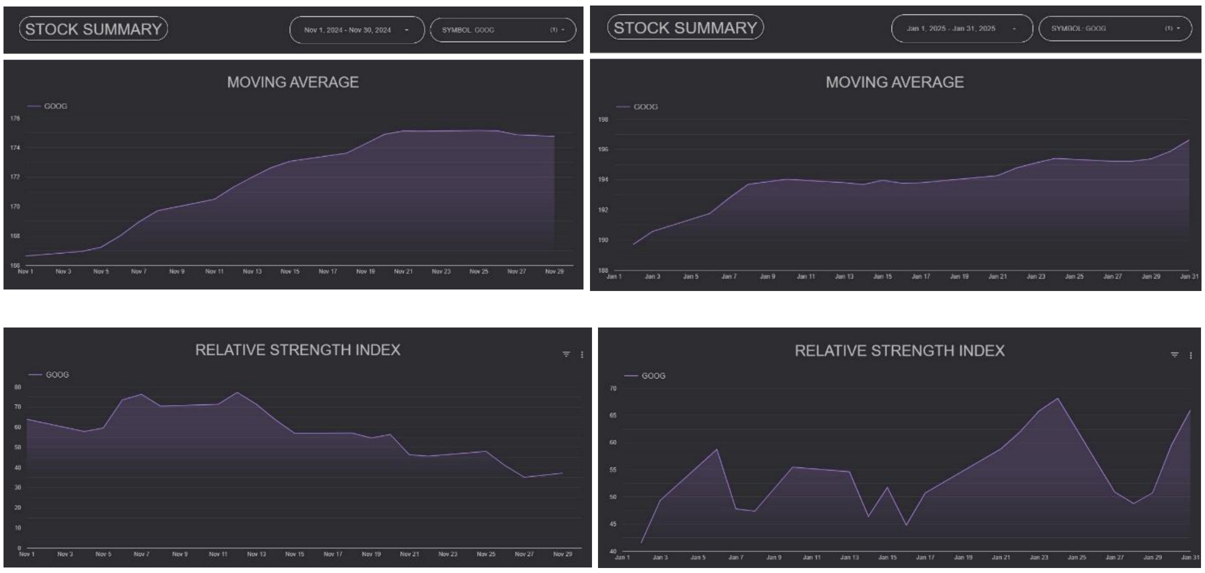
A. Microsoft



These charts from Looker showcase Microsoft’s stock performance across two time periods: **November 2024** and **January 2025**. The **Moving Average** plots indicate smoothed trends in stock prices, while November shows a gradual rise and fall, and January reflects a consistent downward trend. The **Relative Strength Index [RSI]** charts below highlight market momentum; RSI values in November fluctuated between 40-60, suggesting neutral momentum. However, in January, RSI began low and showed a gradual increase, peaking around mid-month before stabilizing. This might be due to oversold conditions early on, with recovery signs later.

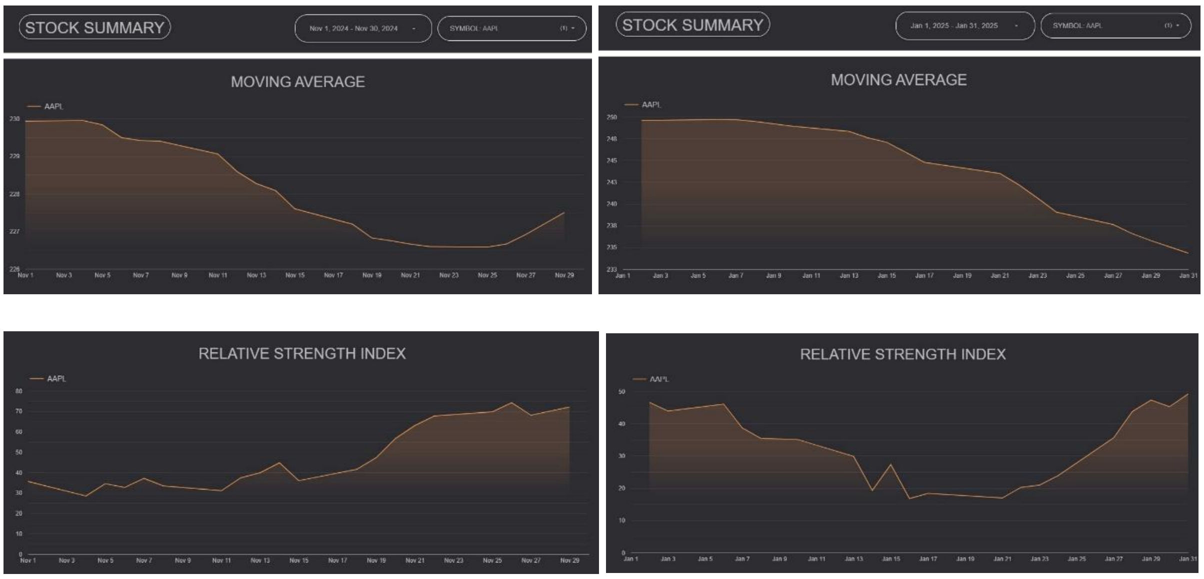
These insights are especially valuable for identifying entry and exit points based on market momentum. By leveraging Looker’s interactivity, users can filter by stock symbols and custom time ranges to explore deeper trends. This allows analysts to monitor changes in technical indicators like RSI in real time. Overall, the dashboard offers a streamlined way to track and interpret Microsoft’s stock health visually.

B. Google



The visualizations present Google’s stock trends over two periods: **November 2024** and **January 2025**. In November, the **Moving Average** steadily increased, showing consistent upward momentum throughout the month. January's chart continues this trend, with a more gradual rise suggesting some market stabilization. The **Relative Strength Index (RSI)** during November started strong, hovering above 60 and even touching 75, indicating potentially overbought conditions. However, it began to decline mid-month, suggesting a slowdown in momentum. In contrast, January's RSI was more volatile, fluctuating sharply but ending on a higher note, which draws a good signal to the business industry for renewed buying interest. These dashboards make it easy to track momentum shifts and price movement patterns at a glance.

C. Apple



These visualizations provide a clear view of Apple's stock behaviour over **November 2024** and **January 2025**. In both months, the **Moving Average** shows a downward trend, suggesting a steady decline in stock value over time. November's chart hints at a slight recovery at the end of the month, but the overall trend remained negative. January's movement was more consistent—gradually dipping throughout the month with no major upward corrections. On the **RSI [Relative Strength Index]** side, Apple started November with low momentum, but the RSI climbed steadily, even breaching 60 by the end of the month—indicating increasing buying pressure. In January, the RSI was more volatile early on, but eventually trended upward in the last 10 days, possibly hinting at a future price rebound or renewed investor interest.

---

6. Results

The implementation of this end-to-end data pipeline demonstrates both functional accuracy and architectural efficiency. This seamless automation helps us to draw business insights without any manual intervention at a precision rate. As the project unfolded to its final stages, the following outcomes were achieved

1. Fully automated pipeline: Extract – Transforms – Loads and Extract – Load – Transforms into Snowflake via dbt.
2. Modular DAGs: ETL and ELT split into independent Airflow DAGs.
3. Error handling: Airflow logs failures and retries as needed.
4. Clean transformations with dbt: Clear SQL logic and version control.
5. Visual insights: Traders and analysts can quickly identify stock trends, volatility, and RSI thresholds.

---

7. Discussion

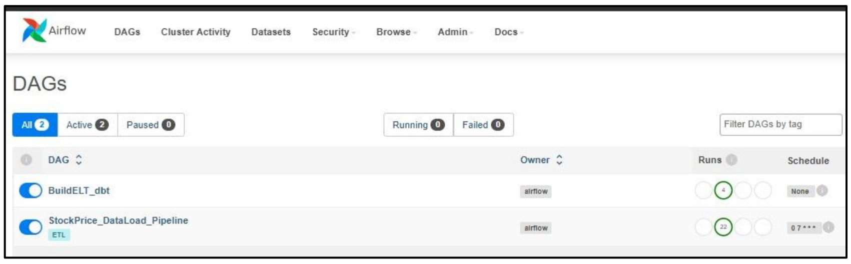
The project’s discussion highlights the strengths across the data pipeline components. Each tool contributes uniquely to the pipeline’s functionality, reliability, and user experience. The following points summarize key observations

1. DBT tests are useful in verifying data quality and primary keys.
2. Airflow DAG scheduling allowed full automation without human intervention.
3. Snowflake’s scalability supported fast queries even on growing datasets.
4. Looker enabled interactive exploration, but filter setup required trial and error for appropriate filters.

---

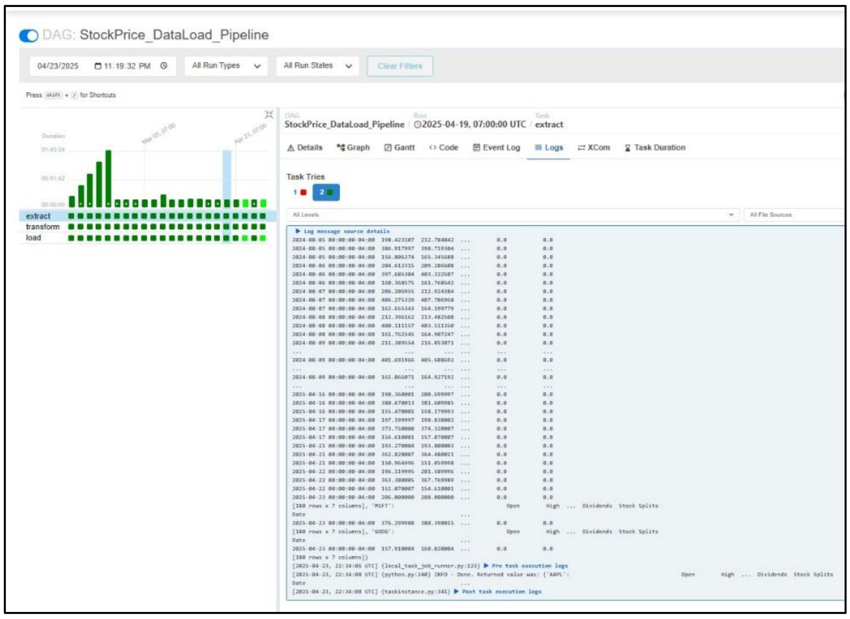
8. Execution Screenshots

[i] Screenshot – ETL and ELT Active status



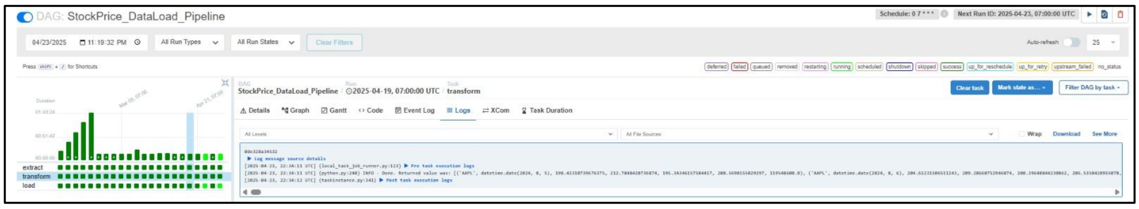
DAG	Owner	Runs	Schedule
BuildELT_dbt	airflow	4	None
StockPrice_DataLoad_Pipeline ETL	airflow	22	0 7 * * *

[i] Screenshots – Extract Transform Load Airflow log screens



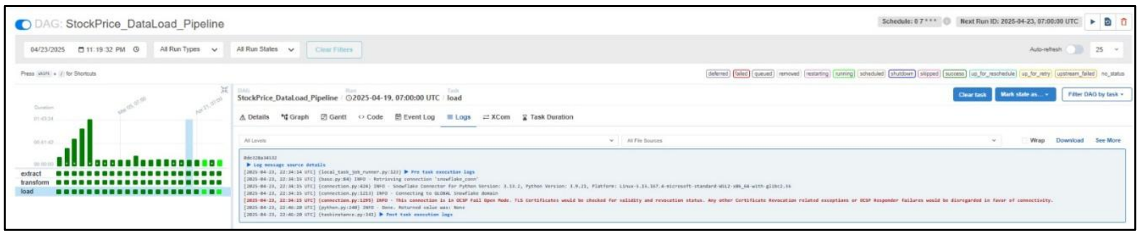
StockPrice_DataLoad_Pipeline / 2025-04-19, 07:00:00 UTC / extract

```
▶ Log message source details
2024-08-05 00:00:00-04:00  198.423107  212.784842  ...       0.0          0.0
2024-08-05 00:00:00-04:00  386.917997  398.719304  ...       0.0          0.0
2024-08-05 00:00:00-04:00  156.806374  165.345688  ...       0.0          0.0
2024-08-06 00:00:00-04:00  204.612315  209.286608  ...       0.0          0.0
2024-08-06 00:00:00-04:00  397.685304  403.322507  ...       0.0          0.0
2024-08-06 00:00:00-04:00  160.368575  161.768542  ...       0.0          0.0
2024-08-07 00:00:00-04:00  206.206955  212.924384  ...       0.0          0.0
2024-08-07 00:00:00-04:00  406.275129  407.706968  ...       0.0          0.0
2024-08-07 00:00:00-04:00  162.655343  164.199779  ...       0.0          0.0
2024-08-08 00:00:00-04:00  212.396162  213.482508  ...       0.0          0.0
2024-08-08 00:00:00-04:00  400.111157  403.511350  ...       0.0          0.0
2024-08-08 00:00:00-04:00  161.762545  164.987247  ...       0.0          0.0
2024-08-09 00:00:00-04:00  211.389554  216.053871  ...       0.0          0.0
...                               ...         ...  ...       ...          ...
2024-08-09 00:00:00-04:00  401.691966  405.688692  ...       0.0          0.0
...                               ...         ...  ...       ...          ...
2024-08-09 00:00:00-04:00  161.066071  164.927192  ...       0.0          0.0
...                               ...         ...  ...       ...          ...
2025-04-16 00:00:00-04:00  198.360001  200.699997  ...       0.0          0.0
2025-04-16 00:00:00-04:00  380.670013  381.609985  ...       0.0          0.0
2025-04-16 00:00:00-04:00  155.470001  158.179993  ...       0.0          0.0
2025-04-17 00:00:00-04:00  197.199997  198.830002  ...       0.0          0.0
2025-04-17 00:00:00-04:00  373.750000  374.320007  ...       0.0          0.0
2025-04-17 00:00:00-04:00  156.610001  157.070007  ...       0.0          0.0
2025-04-21 00:00:00-04:00  193.270004  193.800003  ...       0.0          0.0
2025-04-21 00:00:00-04:00  362.820007  364.480011  ...       0.0          0.0
2025-04-21 00:00:00-04:00  150.964996  151.059998  ...       0.0          0.0
2025-04-22 00:00:00-04:00  196.119995  201.589996  ...       0.0          0.0
2025-04-22 00:00:00-04:00  363.380005  367.769989  ...       0.0          0.0
2025-04-22 00:00:00-04:00  151.070007  154.610001  ...       0.0          0.0
2025-04-23 00:00:00-04:00  206.000000  208.000000  ...       0.0          0.0
[180 rows x 7 columns], 'MSFT':                               Open        High  ...  Dividends  Stock Splits
Date                                                ...
2025-04-23 00:00:00-04:00  376.299988  380.390015  ...       0.0          0.0
[180 rows x 7 columns], 'GOOG':                               Open        High  ...  Dividends  Stock Splits
Date                                                ...
2025-04-23 00:00:00-04:00  157.910004  160.020004  ...       0.0          0.0
[180 rows x 7 columns]}
[2025-04-23, 22:34:06 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-04-23, 22:34:08 UTC] {python.py:240} INFO - Done. Returned value was: {'AAPL':                               Open        High  ...  Dividends  Stock Splits
Date                                                ...
[2025-04-23, 22:34:08 UTC] {taskinstance.py:341} ▶ Post task execution logs
```



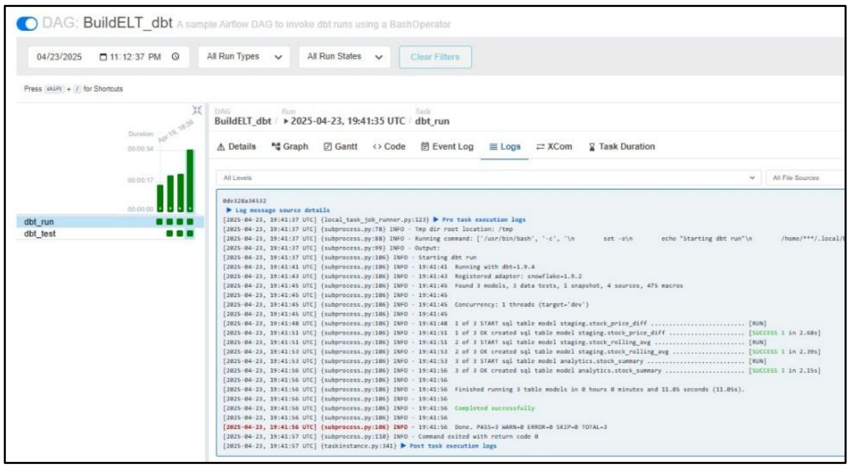
StockPrice_DataLoad_Pipeline / 2025-04-19, 07:00:00 UTC / transform

```
0de328a34532
▶ Log message source details
[2025-04-23, 22:34:11 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-04-23, 22:34:11 UTC] {python.py:240} INFO - Done. Returned value was: [('AAPL', datetime.date(2024, 8, 5), 198.42310729676375, 212.7848420736874, 195.34346157584417, 208.5690155029297, 119548600.0), ('AAPL', datetime.date(2024, 8, 6), 204.61231506511243, 209.28660752946874, 200.19648844230862, 206.5358428955078,
[2025-04-23, 22:34:12 UTC] {taskinstance.py:341} ▶ Post task execution logs
```

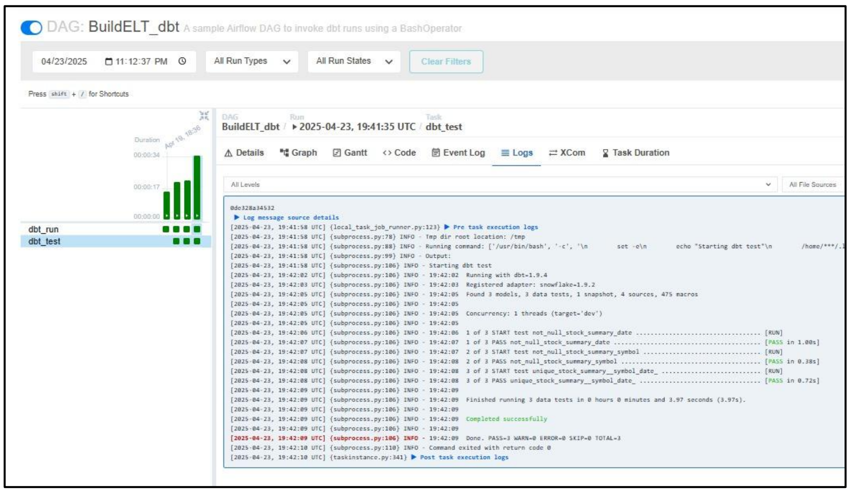


```
0dc328a34532
▶ Log message source details
[2025-04-23, 22:34:14 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-04-23, 22:34:15 UTC] {base.py:84} INFO - Retrieving connection 'snowflake_conn'
[2025-04-23, 22:34:15 UTC] {connection.py:424} INFO - Snowflake Connector for Python Version: 3.13.2, Python Version: 3.9.21, Platform: Linux-5.15.167.4-microsoft-standard-WSL2-x86_64-with-glibc2.36
[2025-04-23, 22:34:15 UTC] {connection.py:1213} INFO - Connecting to GLOBAL Snowflake domain
[2025-04-23, 22:34:15 UTC] {connection.py:1295} INFO - This connection is in OCSP Fail Open Mode. TLS Certificates would be checked for validity and revocation status. Any other Certificate Revocation related exceptions or OCSP Responder failures would be disregarded in favor of connectivity.
[2025-04-23, 22:40:20 UTC] {python.py:240} INFO - Done. Returned value was: None
[2025-04-23, 22:40:20 UTC] {taskinstance.py:341} ▶ Post task execution logs
```

[i] Screenshots – ELT Airflow log screens



```
0dc328a34532
▶ Log message source details
[2025-04-23, 19:41:37 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-04-23, 19:41:37 UTC] {subprocess.py:78} INFO - Tmp dir root location: /tmp
[2025-04-23, 19:41:37 UTC] {subprocess.py:88} INFO - Running command: ['/usr/bin/bash', '-c', '\n        set -e\n        echo "Starting dbt run"\n        /home/***/.local/
[2025-04-23, 19:41:37 UTC] {subprocess.py:99} INFO - Output:
[2025-04-23, 19:41:37 UTC] {subprocess.py:106} INFO - Starting dbt run
[2025-04-23, 19:41:41 UTC] {subprocess.py:106} INFO - 19:41:41  Running with dbt=1.9.4
[2025-04-23, 19:41:43 UTC] {subprocess.py:106} INFO - 19:41:43  Registered adapter: snowflake=1.9.2
[2025-04-23, 19:41:45 UTC] {subprocess.py:106} INFO - 19:41:45  Found 3 models, 3 data tests, 1 snapshot, 4 sources, 475 macros
[2025-04-23, 19:41:45 UTC] {subprocess.py:106} INFO - 19:41:45
[2025-04-23, 19:41:45 UTC] {subprocess.py:106} INFO - 19:41:45  Concurrency: 1 threads (target='dev')
[2025-04-23, 19:41:45 UTC] {subprocess.py:106} INFO - 19:41:45
[2025-04-23, 19:41:48 UTC] {subprocess.py:106} INFO - 19:41:48  1 of 3 START sql table model staging.stock_price_diff .......................... [RUN]
[2025-04-23, 19:41:51 UTC] {subprocess.py:106} INFO - 19:41:51  1 of 3 OK created sql table model staging.stock_price_diff ..................... [SUCCESS 1 in 2.68s]
[2025-04-23, 19:41:51 UTC] {subprocess.py:106} INFO - 19:41:51  2 of 3 START sql table model staging.stock_rolling_avg ......................... [RUN]
[2025-04-23, 19:41:53 UTC] {subprocess.py:106} INFO - 19:41:53  2 of 3 OK created sql table model staging.stock_rolling_avg .................... [SUCCESS 1 in 2.39s]
[2025-04-23, 19:41:53 UTC] {subprocess.py:106} INFO - 19:41:53  3 of 3 START sql table model analytics.stock_summary ........................... [RUN]
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56  3 of 3 OK created sql table model analytics.stock_summary ...................... [SUCCESS 1 in 2.15s]
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56  Finished running 3 table models in 0 hours 0 minutes and 11.05 seconds (11.05s).
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56  Completed successfully
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56
[2025-04-23, 19:41:56 UTC] {subprocess.py:106} INFO - 19:41:56  Done. PASS=3 WARN=0 ERROR=0 SKIP=0 TOTAL=3
[2025-04-23, 19:41:57 UTC] {subprocess.py:110} INFO - Command exited with return code 0
[2025-04-23, 19:41:57 UTC] {taskinstance.py:341} ▶ Post task execution logs
```



```
0de328a34532
▶ Log message source details
[2025-04-23, 19:41:58 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-04-23, 19:41:58 UTC] {subprocess.py:78} INFO - Tmp dir root location: /tmp
[2025-04-23, 19:41:58 UTC] {subprocess.py:88} INFO - Running command: ['/usr/bin/bash', '-c', '\n        set -e\n        echo "Starting dbt test"\n        /home/***/.l
[2025-04-23, 19:41:58 UTC] {subprocess.py:99} INFO - Output:
[2025-04-23, 19:41:58 UTC] {subprocess.py:106} INFO - Starting dbt test
[2025-04-23, 19:42:02 UTC] {subprocess.py:106} INFO - 19:42:02  Running with dbt=1.9.4
[2025-04-23, 19:42:03 UTC] {subprocess.py:106} INFO - 19:42:03  Registered adapter: snowflake=1.9.2
[2025-04-23, 19:42:05 UTC] {subprocess.py:106} INFO - 19:42:05  Found 3 models, 3 data tests, 1 snapshot, 4 sources, 475 macros
[2025-04-23, 19:42:05 UTC] {subprocess.py:106} INFO - 19:42:05
[2025-04-23, 19:42:05 UTC] {subprocess.py:106} INFO - 19:42:05  Concurrency: 1 threads (target='dev')
[2025-04-23, 19:42:05 UTC] {subprocess.py:106} INFO - 19:42:05
[2025-04-23, 19:42:06 UTC] {subprocess.py:106} INFO - 19:42:06  1 of 3 START test not_null_stock_summary_date ................................. [RUN]
[2025-04-23, 19:42:07 UTC] {subprocess.py:106} INFO - 19:42:07  1 of 3 PASS not_null_stock_summary_date ....................................... [PASS in 1.00s]
[2025-04-23, 19:42:07 UTC] {subprocess.py:106} INFO - 19:42:07  2 of 3 START test not_null_stock_summary_symbol ............................... [RUN]
[2025-04-23, 19:42:08 UTC] {subprocess.py:106} INFO - 19:42:08  2 of 3 PASS not_null_stock_summary_symbol ..................................... [PASS in 0.38s]
[2025-04-23, 19:42:08 UTC] {subprocess.py:106} INFO - 19:42:08  3 of 3 START test unique_stock_summary__symbol_date_ .......................... [RUN]
[2025-04-23, 19:42:08 UTC] {subprocess.py:106} INFO - 19:42:08  3 of 3 PASS unique_stock_summary__symbol_date_ ................................ [PASS in 0.72s]
[2025-04-23, 19:42:09 UTC] {subprocess.py:106} INFO - 19:42:09
[2025-04-23, 19:42:09 UTC] {subprocess.py:106} INFO - 19:42:09  Finished running 3 data tests in 0 hours 0 minutes and 3.97 seconds (3.97s).
[2025-04-23, 19:42:09 UTC] {subprocess.py:106} INFO - 19:42:09
[2025-04-23, 19:42:09 UTC] {subprocess.py:106} INFO - 19:42:09  Completed successfully
[2025-04-23, 19:42:09 UTC] {subprocess.py:106} INFO - 19:42:09
[2025-04-23, 19:42:09 UTC] {subprocess.py:106} INFO - 19:42:09  Done. PASS=3 WARN=0 ERROR=0 SKIP=0 TOTAL=3
[2025-04-23, 19:42:10 UTC] {subprocess.py:110} INFO - Command exited with return code 0
[2025-04-23, 19:42:10 UTC] {taskinstance.py:341} ▶ Post task execution logs
```

9. Conclusion

This project demonstrates how to build and deploy an end-to-end ELT pipeline using modern data engineering tools. By combining Airflow, Snowflake, dbt, and Looker, we established a scalable, automated, and insightful pipeline capable of real-time stock market analysis. Looker played a key role in transforming data into actionable insights through interactive dashboards and its unique metric feature. Its flexibility in visualizing and filtering enables analysts to explore stock trends dynamically across multiple dimensions. This modular design supports future extensions and deeper analytics.

10. References

[1] Apache Airflow Documentation: https://airflow.apache.org/
[2] dbt Documentation: https://docs.getdbt.com/
[3] Snowflake Documentation: https://docs.snowflake.com/
[4] Looker Documentation: https://cloud.google.com/looker/docs
[5] yFinance: https://github.com/ranaroussi/yfinance

GitHub Link: https://github.com/GoelAnshika99/Lab2-Group16/tree/main