LAB 1: Project Report for Stock Price Prediction Analytics using Airflow & Snowflake

1.Team Introduction

Lab group: 9
Elsa Rose
Kruthika Virupakshappa

Our team consists of two passionate data science students specializing in applied analytics, data engineering, and financial modeling. With strong backgrounds in Python programming, cloud-based data pipelines, and statistical analysis, we collaborated to design, develop, and implement a robust stock price prediction system for this lab. Each team member contributed expertise in ETL pipeline development, machine learning modeling, and data visualization using industry-standard tools such as Apache Airflow, Docker, Python Colab, and Snowflake. Our complementary skills enabled us to tackle challenges in automation, data orchestration, and advanced time series forecasting, resulting in a comprehensive and scalable financial analytics solution for the Lab.

2. Introduction

This project focuses on building a Finance Data Analytics system utilizing the yfinance API for extracting comprehensive historical stock market data including open, high, low, close prices, and trading volumes. Leveraging this rich dataset enables conducting various financial analyses such as stock price prediction, trend and volatility analysis, technical indicator evaluation, and portfolio optimization. By integrating automated data collection through yfinance with cloud-based storage and processing in Snowflake, the system provides an efficient pipeline for storing, and analyzing stock market data.

3. Problem Statement

The objective of this project is to develop a stock price prediction and analysis system focused on forecasting the stock prices of key companies such as NVIDIA (NVDA) and TESLA (TSLA) over the next seven or more days by leveraging historical stock price data. Accurate forecasting of stock prices is invaluable to investors for informed decision-making, effective risk management, and understanding market trends.

To support this goal, the system utilizes a Snowflake database to store large volumes of daily stock price data across multiple companies. Data pipelines, orchestrated by Airflow, automate the extraction, transformation, and loading (ETL) of this stock data. These pipelines ensure the data remains fresh, consistent, and ready for downstream forecasting processes.

4. Solution Requirements

4.1 Functional Requirements:

- Extract last 180 days of stock prices for selected companies using yfinance API.
- Load the data into Snowflake tables with proper schema.
- Automate data collection with Airflow DAGs running daily.
- Implement ML forecasting pipelines also as Airflow DAGs.
- Schedule forecasting after the loading pipeline completes.

4.2 Non-functional Requirements:

- Pipeline idempotency ensures data loads can be re-run safely.
- Secure management of credentials (Snowflake, yfinance API).
- Scalability to add more stock symbols.

5. Functional Analysis

The proposed system comprises several interconnected components, each serving a critical role in achieving accurate stock price forecasting through automation and scalable processing.

5.1 Component Overview

Data Extraction:
Python scripts leverage the yfinance API to fetch historical stock data, including open, high, low, close, and volume metrics for multiple companies such as NVDA and TSLA.

Data Loading:
The Snowflake Python connector inserts the extracted data into structured database tables. This process ensures data integrity through SQL transaction management, with error handling to prevent inconsistencies.

Data Pipeline Automation:
Apache Airflow orchestrates daily workflows to automate the ETL (Extract, Transform, Load) processes. Connections and credentials are securely managed via Airflow's connection settings, ensuring seamless and reliable execution of data pipelines.

ML Forecasting Module:
A dedicated Airflow DAG triggers machine learning models which utilize the stored historical data. The models, trained on the data, generate forecasts of the upcoming 7+ days, storing predictions in a separate Snowflake table.

5.2 Union and Presentation:
A final SQL step unions actual historical data with forecasted prices to generate a comprehensive, unified view. This enables efficient analysis and decision-making based on combined historical and predictive insights.

5.3 Database and Data Pipeline Interaction

- Extracted data is transformed into a predefined schema with fields such as symbol, date, open, close, high, low, and volume.
- The data pipeline sequences ensure that the forecasting models only run after successful data loading, maintaining data consistency.
- SQL transactions with try/except error handling prevent data corruption and ensure rollback on failures, maintaining system robustness.

This integrated architecture supports scalable, automated, and accurate stock market analysis, providing a solid foundation for predictive analytics and strategic investment insights.

6. Dataset

The dataset for this project is sourced from the yfinance API, which provides comprehensive historical stock market data directly from Yahoo Finance. The standard dataset includes the following key fields for each stock ticker and date:

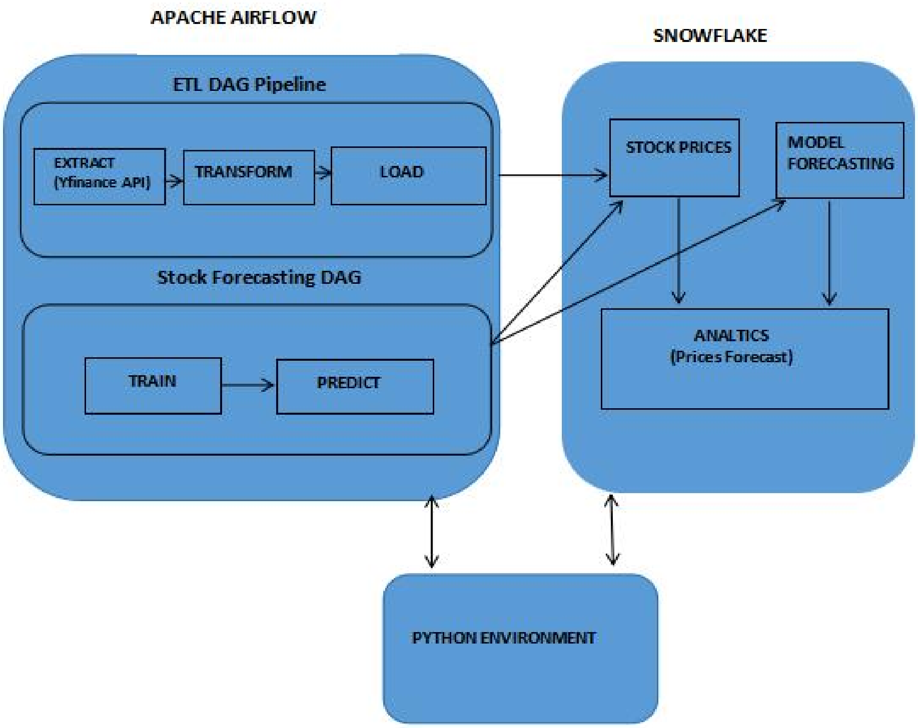
- Date: The specific trading day for the market data.
- Open: The price at which the stock started trading on the day.
- High: The highest price reached during the trading session.
- Low: The lowest price reached during the trading session.
- Close: The final price at which the stock traded when the market closed.
- Volume: The total number of shares traded during the day.
- Symbol (Ticker): The stock symbol, such as "TSLA" for Tesla or "NVDA" for NVIDIA.

yfinance lets users specify the date range (last 180 days) and download data at daily or other time intervals. The resulting dataset is structured as a table where each row captures the above attributes for a company on a given date. This structure is ideal for time series analysis, trend detection, and building forecasting models for financial analytics.

7. Github Link

Kruthika V
Elsa Rose

8. Architectural Diagram



This architectural diagram illustrates the overall design of your stock price prediction and analytics system, highlighting how the components work together according to the project requirements.

7.1 ETL Data Flow with Airflow

ETL DAG Pipeline:

- Extract: Airflow runs scheduled DAGs to extract historical stock data (open, high, low, close, volume) from the yfinance API.
- Transform: The raw data is transformed/cleaned to ensure it is consistent and ready for loading.
- Load: The cleansed data is loaded into the "STOCK PRICES" table in Snowflake, ensuring records for each trading day and company are captured.

7.2 Machine Learning Forecasting Orchestration

Stock Forecasting DAG:

- Train: The Airflow ML pipeline triggers model training, potentially using data pulled from Snowflake for supervised learning or time series models.
- Predict: Forecasted stock prices for the next 7 days are generated and inserted into Snowflake.

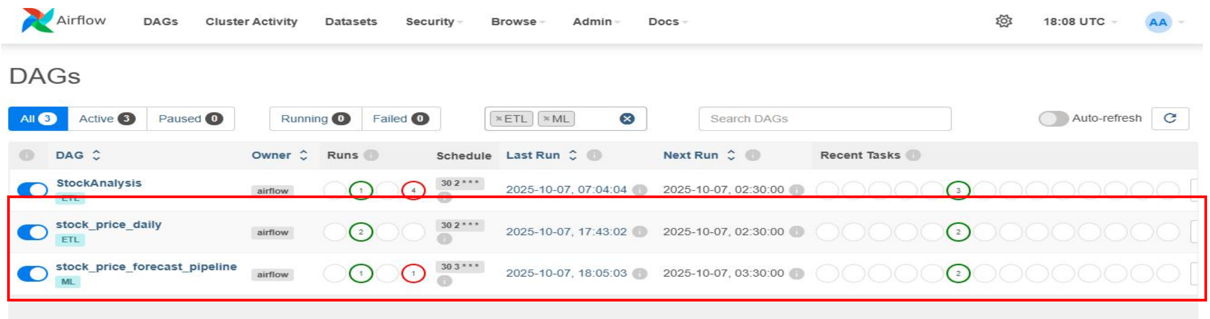


Figure: Aiflow Dag Screen

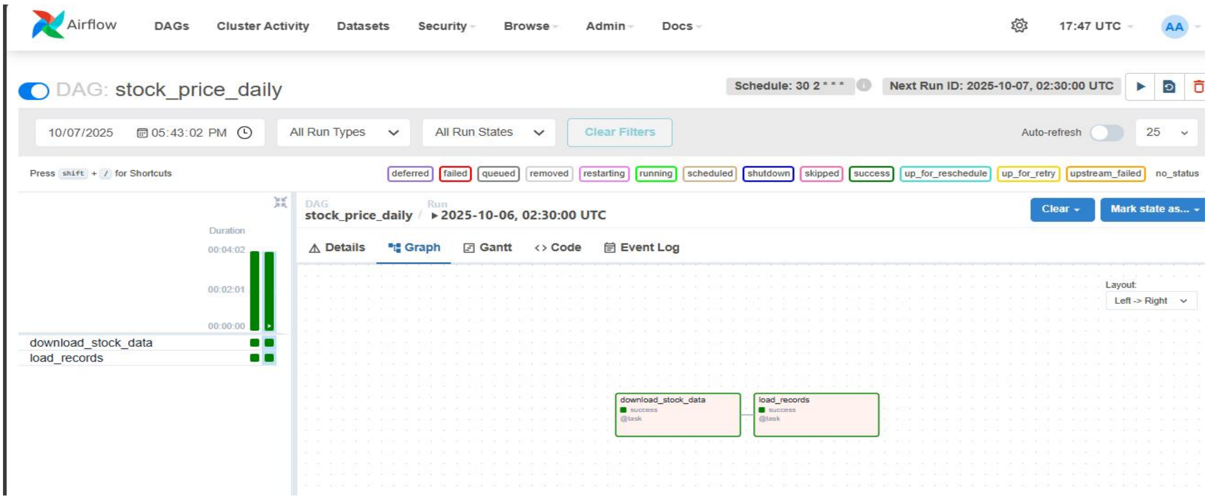


Figure: Airflow Stock_daily

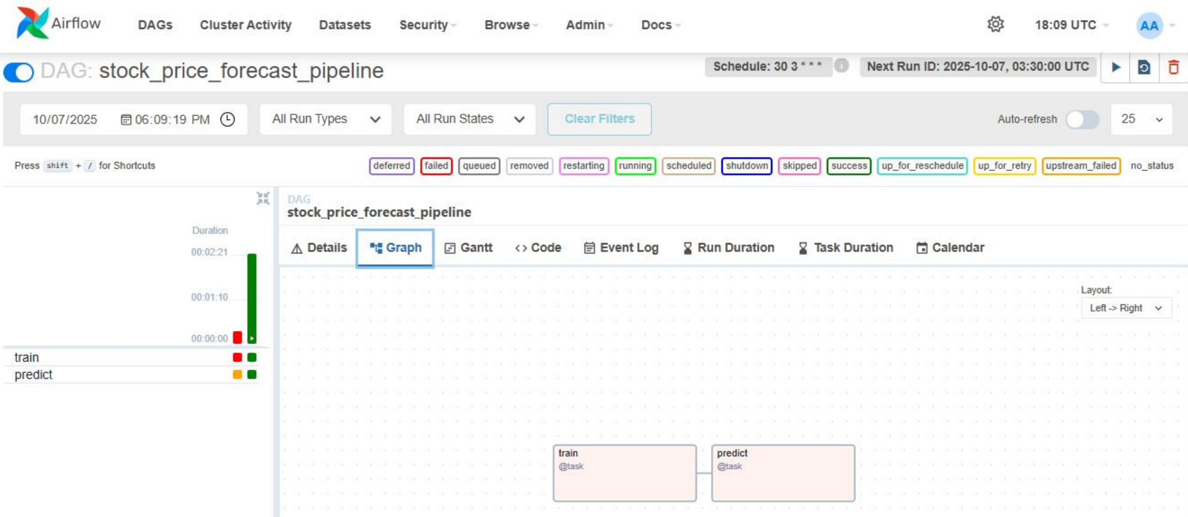


Figure: Stock Prediction Analysis graph view

7.3 Central Data Warehouse: Snowflake

- Stock Prices Table: This stores all historical stock market data populated daily by the ETL pipeline.
- Model Forecasting Table: Holds the machine learning model outputs, i.e., the predicted prices for coming days.
- Analytics Table: Combines actual and predicted prices for reporting and downstream analytics.

7.4 Python Environment & Integration

Python Environment (Colab):

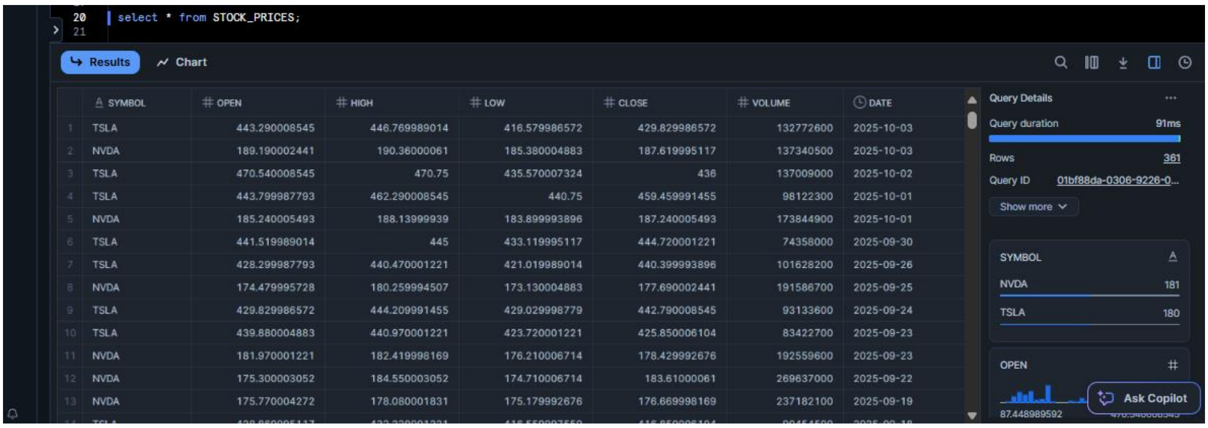
- Acts as the interactive coding/development space for rapid prototyping, data analysis, and testing.
- Connected with both Airflow and Snowflake, allowing direct Python queries, result visualization, or ad-hoc analysis outside of scheduled pipelines.

7.5 Component Interactions:

- Airflow is containerized (assumed via Docker): Provides reliable orchestration and scheduling for both ETL and ML pipelines.
- Snowflake: Centralizes both raw and forecast data for scalable analytics.
- Python Environment: Supports user experimentation, diagnostics, and model refinement.

9. Tables

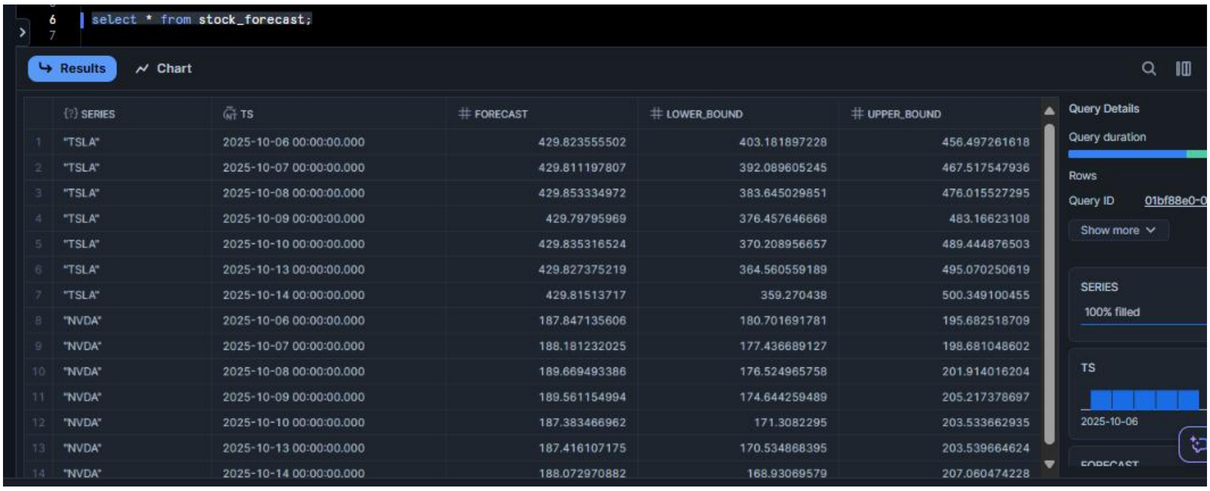
8.1 Sample output STOCK_PRICES for 180 days



```
select * from STOCK_PRICES;
```

	SYMBOL	OPEN	HIGH	LOW	CLOSE	VOLUME	DATE
1	TSLA	443.290008545	446.769989014	416.579986572	429.829986572	132772600	2025-10-03
2	NVDA	189.190002441	190.36000061	185.380004883	187.619995117	137340500	2025-10-03
3	TSLA	470.540008545	470.75	435.570007324	436	137009000	2025-10-02
4	TSLA	443.799987793	462.290008545	440.75	459.459991455	98122300	2025-10-01
5	NVDA	185.240005493	188.13999939	183.899993896	187.240005493	173844900	2025-10-01
6	TSLA	441.519989014	445	433.119995117	444.720001221	74358000	2025-09-30
7	TSLA	428.299987793	440.470001221	421.019989014	440.399993896	101628200	2025-09-26
8	NVDA	174.479995728	180.259994507	173.130004883	177.690002441	191586700	2025-09-25
9	TSLA	429.829986572	444.209991455	429.029998779	442.790008545	93133600	2025-09-24
10	TSLA	439.880004883	440.970001221	423.720001221	425.850006104	83422700	2025-09-23
11	NVDA	181.970001221	182.419998169	176.210006714	178.429992676	192559600	2025-09-23
12	NVDA	175.300003052	184.550003052	174.710006714	183.61000061	269637000	2025-09-22
13	NVDA	175.770004272	178.080001831	175.179992676	176.669998169	237182100	2025-09-19

8.2 Sample output STOCK_FORECAST for 7 days



```
select * from stock_forecast;
```

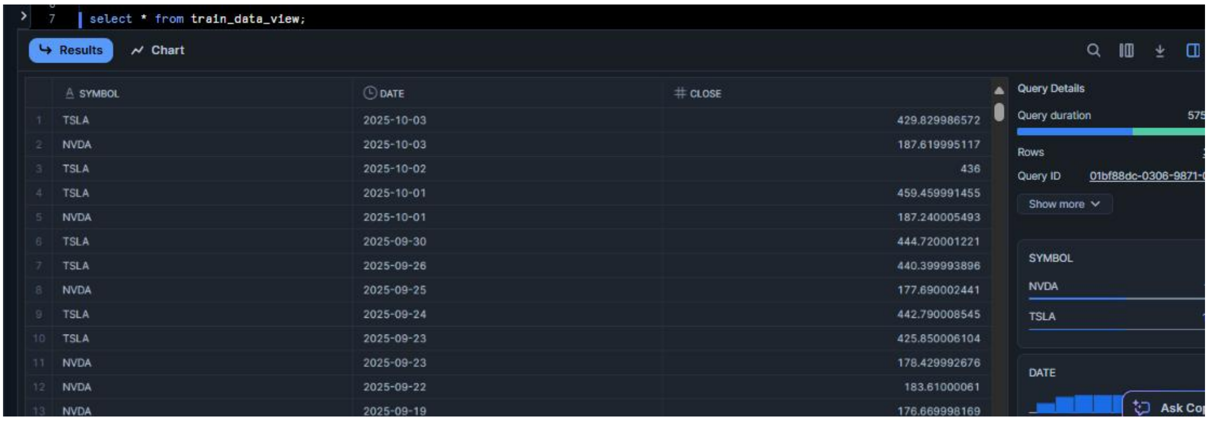
	SERIES	TS	FORECAST	LOWER_BOUND	UPPER_BOUND
1	"TSLA"	2025-10-06 00:00:00.000	429.823555502	403.181897228	456.497261618
2	"TSLA"	2025-10-07 00:00:00.000	429.811197807	392.089605245	467.517547936
3	"TSLA"	2025-10-08 00:00:00.000	429.853334972	383.645029851	476.015527295
4	"TSLA"	2025-10-09 00:00:00.000	429.79795969	376.457646668	483.16623108
5	"TSLA"	2025-10-10 00:00:00.000	429.835316524	370.208956657	489.444876503
6	"TSLA"	2025-10-13 00:00:00.000	429.827375219	364.560559189	495.070250619
7	"TSLA"	2025-10-14 00:00:00.000	429.81513717	359.270438	500.349100455
8	"NVDA"	2025-10-06 00:00:00.000	187.847135606	180.701691781	195.682518709
9	"NVDA"	2025-10-07 00:00:00.000	188.181232025	177.436689127	198.681048602
10	"NVDA"	2025-10-08 00:00:00.000	189.669493386	176.524965758	201.914016204
11	"NVDA"	2025-10-09 00:00:00.000	189.561154994	174.644259489	205.217378697
12	"NVDA"	2025-10-10 00:00:00.000	187.383466962	171.3082295	203.533662935
13	"NVDA"	2025-10-13 00:00:00.000	187.416107175	170.534868395	203.539664624
14	"NVDA"	2025-10-14 00:00:00.000	188.072970882	168.93069579	207.060474228

8.3 Schema Summary ANALYTICS

```
SELECT table_name, column_name, data_type, is_nullable, column_default
FROM USER_DB_PIRANHA.INFORMATION_SCHEMA.COLUMNS
WHERE table_schema = 'ANALYTICS'
ORDER BY table_name;
```

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	IS_NULLABLE	COLUMN_DEFAULT
1	FINAL_RESULT	SYMBOL	TEXT	YES	null
2	FINAL_RESULT	DATE	TIMESTAMP_NTZ	YES	null
3	FINAL_RESULT	FORECAST	FLOAT	YES	null
4	FINAL_RESULT	UPPER_BOUND	FLOAT	YES	null
5	FINAL_RESULT	ACTUAL	FLOAT	YES	null
6	FINAL_RESULT	LOWER_BOUND	FLOAT	YES	null

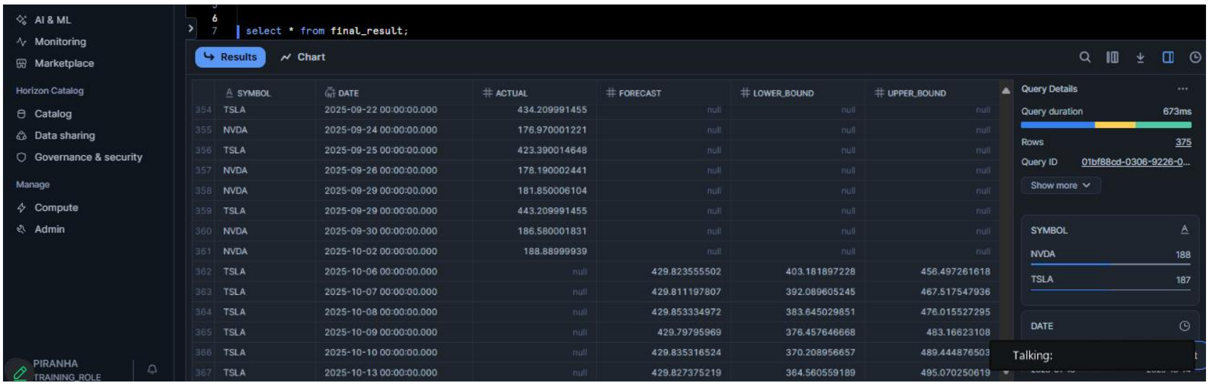
8.4 View TRAIN_DATA_VIEW



```
select * from train_data_view;
```

	SYMBOL	DATE	CLOSE
1	TSLA	2025-10-03	429.829986572
2	NVDA	2025-10-03	187.619995117
3	TSLA	2025-10-02	436
4	TSLA	2025-10-01	459.459991455
5	NVDA	2025-10-01	187.240005493
6	TSLA	2025-09-30	444.720001221
7	TSLA	2025-09-26	440.399993896
8	NVDA	2025-09-25	177.690002441
9	TSLA	2025-09-24	442.790008545
10	TSLA	2025-09-23	425.850006104
11	NVDA	2025-09-23	178.429992676
12	NVDA	2025-09-22	183.61000061
13	NVDA	2025-09-19	176.669998169

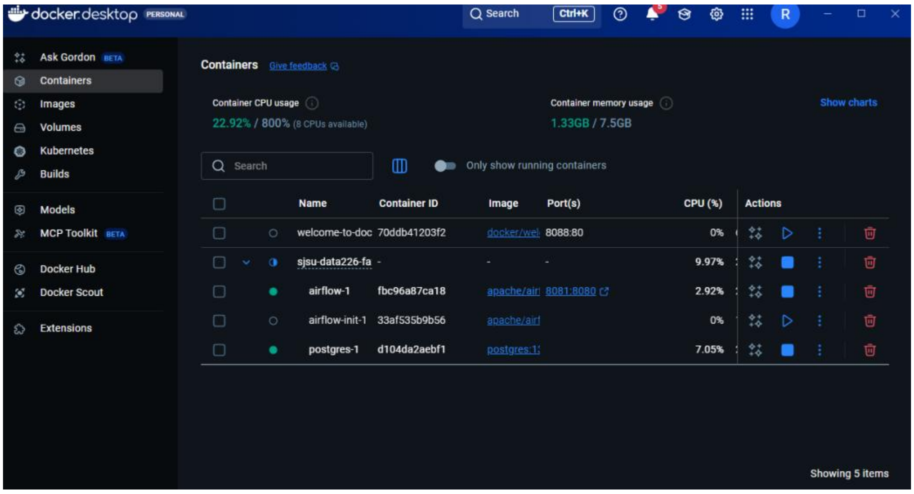
8.5 Sample output ANALYTICS.FINAL_RESULT union of STOCK ACTUAL and STOCK FORECAST



10. Implementation

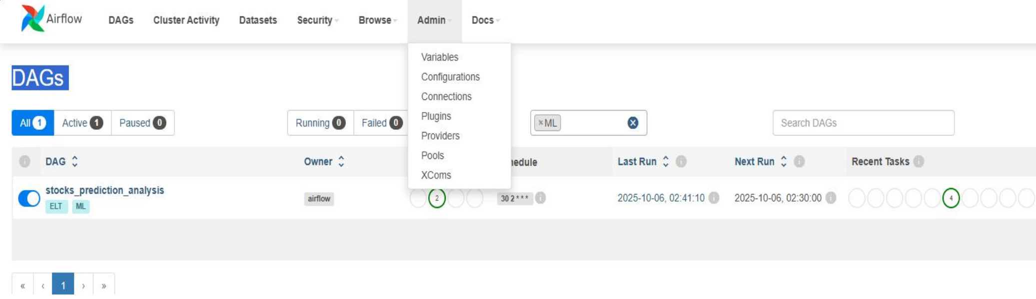
9.1 Airflow Connections

Airflow running in Docker can be accessed from the host machine at http://localhost:8080 when the webserver container's port 8080 is published to the host.

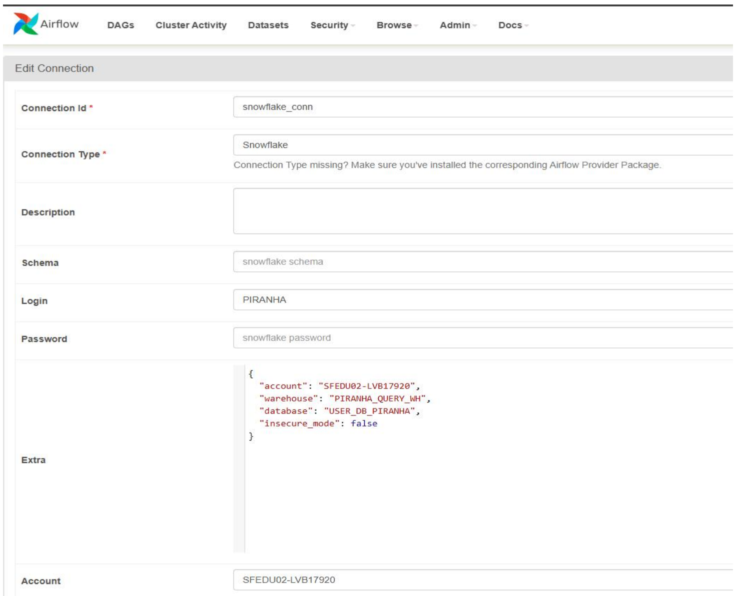


Browser requests the URL http://localhost:8080, which reaches the Docker host port bound by docker-compose or docker run. Docker routes the request into the airflow-webserver container's port 8080 where the Airflow webserver process handles it and serves the login UI

Connections are used to store credentials and configuration to connect Airflow to snowflake connections can be managed in the Airflow UI at Admin > Connections



To add a connection, click + Add Connection, then set a connection ID snowflake_conn



11. Lessons:

10.1 Building Automated, Scalable Data Pipelines

Implementing Airflow DAGs for ETL and stock forecasting automated daily data ingestion, transformation, and prediction. This automation improved reliability, reduced manual effort, and ensured that the analytics system maintained up-to-date data, critical for timely financial insights.

10.2 Importance of Modular Architecture and Containerization

Separating ETL and ML forecasting pipelines facilitated independent development, debugging, and scaling of each component. Containerizing Airflow with Docker standardized deployment environments, enhancing reproducibility and simplifying pipeline management across different platforms.

10.3 Effective Integration of Tools for Financial Analytics

The combination of yfinance for data extraction, Snowflake for cloud storage and analytics, Airflow for orchestration, and Python Colab for interactive development proved powerful.

This toolchain enabled rapid prototyping, smooth data flow, and reliable forecasting workflows.

12. Future Work

Integration of Multi-Source Data for Improved Forecasting
Future work could focus on incorporating alternative and more diverse data sources, such as social media sentiment, news articles, macroeconomic indicators, and real-time market feeds. Combining these with existing stock price data is expected to improve model accuracy by adding context and capturing market sentiment, thus enabling more robust and reliable predictions.

13. Conclusion

This project developed an automated stock price prediction system using Snowflake and Airflow to streamline financial data analytics. By leveraging the yfinance API, historical stock data for selected companies was collected daily and stored in Snowflake through an Airflow ETL pipeline. A separate forecasting pipeline predicted stock prices for the next seven days, ensuring continuous data updates and insights. The integration of data ingestion, transformation, and forecasting in a single automated workflow highlights the efficiency of combining Snowflake's data capabilities with Airflow's orchestration power. Overall, this system provides a scalable, reliable framework for real-time financial forecasting and analysis.
