Stock Price Prediction using Airflow

Lab Group 17:

Group Members:

- Omkar Rajale
- Yashashree Shinde

Abstract

This document serves as a comprehensive project proposal for the development of the **Stock Price Prediction System**. This application aims to provide daily forecasts of stock prices for selected companies (e.g., NVDA, APPL). The proposal details the requirement for a robust data analytics pipeline, outlining the extraction of 180 days of historical stock data via the yfinance API, orchestration of daily data ingestion and cleaning using **Apache Airflow**, and the execution of Machine Learning (ML) forecasting tasks within **Snowflake**. The goal is to deliver a reliable, automated, and scalable system capable of predicting stock prices for the next 7+ days.

1. Introduction

1.1 Project Overview

This section introduces the proposed application system, **Stock Price Prediction Data Analytics System**, and briefly summarizes its primary goal: to deliver automated, daily, data-driven forecasts of future stock prices. The system is designed around modern data engineering practices, utilizing a cloud data warehouse (Snowflake) and an orchestration engine (Airflow).

2. Problem Statement

This section clearly defines the application system your team intends to build and articulates the necessity of the project.

- **2.1 The Problem:** Stock price volatility and the difficulty of accurate short-term forecasting pose a significant challenge for individual investors and automated trading systems. The core problem is the lack of a standardized, automated, and repeatable process to ingest daily historical data, prepare it, and apply machine learning models to generate timely, reliable price predictions.

- **2.2 Proposed Application System:** The **Stock Price Prediction Data Analytics System** is a serverless data pipeline designed to automate the end-to-end process of stock price forecasting. It focuses on repeatability and accuracy by ensuring a daily feed of fresh, 180-day historical data is available for ML tasks targeting the next 7+ days of price movement.
- **2.3 Rationale for Database and Data Pipelines:** A persistent database (Snowflake) is essential for:
    1. **Data Integrity:** Storing raw and refined daily time-series data in a structured, queryable format.
    2. **ML Execution:** Providing a high-performance environment (Snowflake ML) where complex time-series forecasting models can be executed directly on the data with powerful SQL. The data pipeline (Airflow DAG) is mandatory for:
    3. **Automation:** Ensuring the data ingestion from the yfinance API runs reliably **daily** at a scheduled time.
    4. **Repeatability:** Managing the complex multi-step ETL process (Extraction, Loading, ML Model Training, Prediction, and Result Aggregation) as a single, monitorable workflow.

3. Solution Requirements

This section details the necessary criteria, actions, and limitations of the final system, based on the requirements analysis.

- **3.1 Functional Requirements (What the system will do):** List the core capabilities and actions the application must perform.
    - FR1: The system **shall** successfully connect to the yfinance API and retrieve the last 180 days of historical stock data (Open, Close, Min, Max, Volume) for at least two chosen companies (e.g., NVDA, APPL).
    - FR2: The system **shall** use an Airflow DAG to automate the data ingestion process and ensure it runs successfully every day.
    - FR3: The system **shall** load the ingested data into a designated table in Snowflake with the defined schema.
    - FR4: The system **shall** execute SQL-based ML forecasting tasks within Snowflake to predict stock prices for the next 7 days.
    - FR5: The system **shall** aggregate the historical data and the forecasted data into a final output table (`STOCK_PRICES_FINAL`) via a UNION operation.
- **3.2 Non-Functional Requirements (Limitations and Qualities):** Define the criteria for judging the system's operation.
    - **3.2.1 Performance:** The daily Airflow DAG execution (ingestion, processing, and forecasting) must complete within 30 minutes.
    - **3.2.2 Scalability:** The pipeline must be designed to easily add new stock symbols with minimal configuration changes.

    - **3.2.3 Data Integrity:** Data loaded into Snowflake must pass a basic null check on the `Close` and `Date` columns.
    - **3.2.4 Observability:** The Airflow interface must provide clear logging and status monitoring for all daily pipeline runs.
- **3.3 System Users and Usage:** The primary users of this system are **Data Analysts/Scientists** who rely on the final `STOCK_PRICES_FINAL` table for analysis and decision-making.
    - **Use Case 1 (Daily Run):** The Airflow DAG automatically triggers at midnight, ingests fresh data, runs the ML model, and updates the `STOCK_PRICES_FINAL` table.
    - **Use Case 2 (Analyst Query):** A user executes a simple SQL query against the `STOCK_PRICES_FINAL` table to see the predicted prices for NVDA and AAPL for the upcoming week.

4. Functional Analysis and Conceptual Design

This section discusses the application's components and presents the conceptual architecture, illustrating how the parts interact to solve the problem.

- **4.1 Conceptual Architecture:** Provide a high-level overview of the system components.

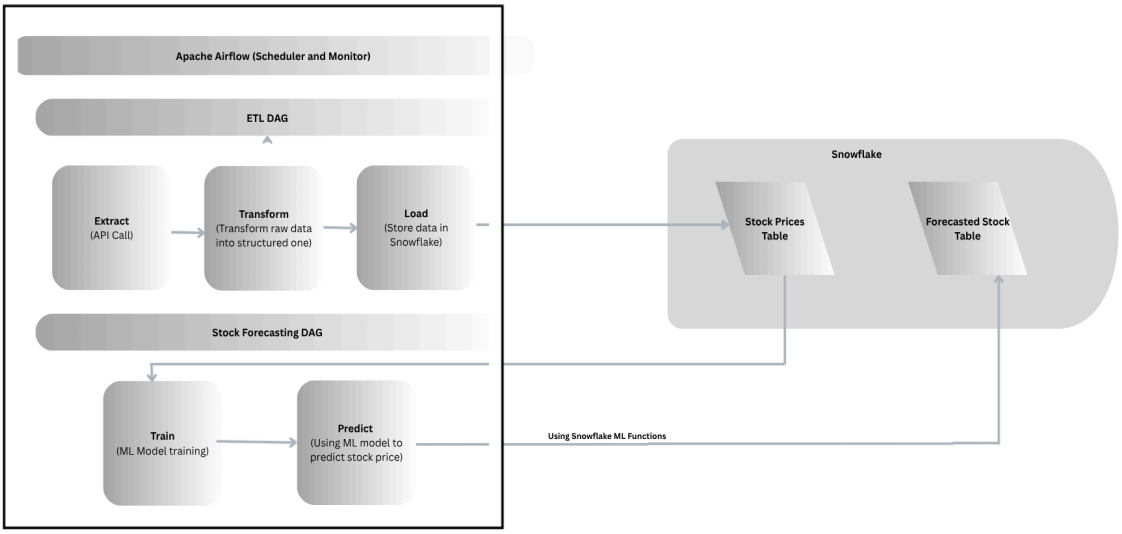


Figure 1. Conceptual Data Pipeline Architecture. This diagram illustrates the sequential flow from the external yfinance API through the Airflow orchestration layer to the Snowflake Data Warehouse for storage and ML forecasting

- **4.2 Functional Components Breakdown:**
  - **4.2.1 Data Ingestion Module (Python/yfinance):**
    - **Role:** Extracts 180 days of stock data from the `yfinance` API.
    - **Database Interactions:** Loads the raw/cleaned data into the Snowflake staging table (`STOCK_PRICES_180D`).
    - **Data Pipeline Interactions:** This module is executed as a task within the Airflow DAG.
  - **4.2.2 Orchestration Layer (Airflow DAG):**
    - **Role:** Schedules the daily execution of all pipeline tasks, manages dependencies, and handles failures.
    - **Database Interactions:** Triggers the Python ingestion module (writing to Snowflake) and executes the SQL forecasting module (reading/writing within Snowflake) using the Airflow Snowflake hook.
    - **Data Pipeline Interactions:** Defines the sequence: Ingest -> Forecast -> Union.
  - **4.2.3 Data Warehouse & ML Engine (Snowflake):**
    - **Role:** Persistent storage for all historical data and the execution environment for ML models via SQL.
    - **Database Interactions:** Reads data from `STOCK_PRICES_180D`; executes `CREATE OR REPLACE MODEL ...` and `SELECT FORECAST(...)` SQL queries; writes predictions to `STOCK_PRICE_FORECAST_7D`.
    - **Data Pipeline Interactions:** Executes the core transformation and prediction logic defined in the DAG.

5. Implementation Details and Appendices

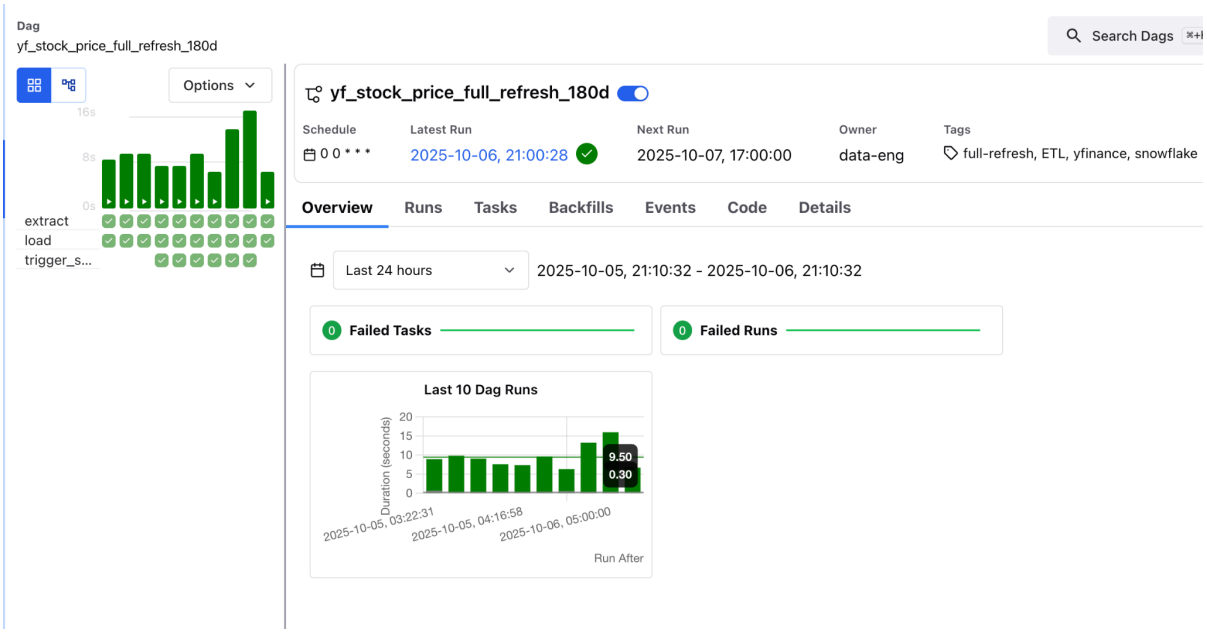
This section provides a summary of the technical output and links to the supporting artifacts.

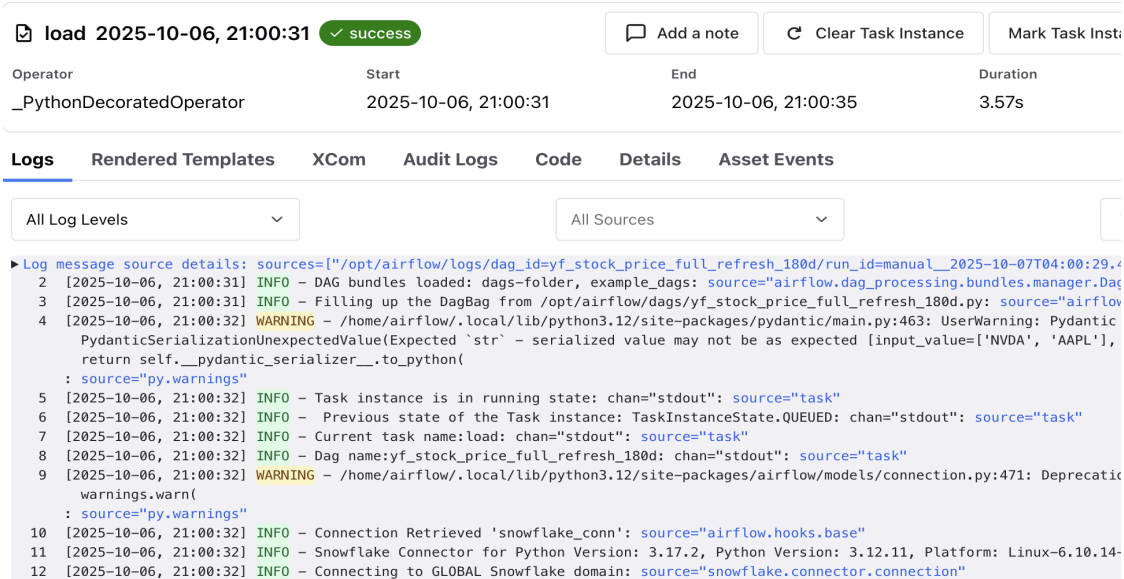
- **5.1 Tables Structure:**

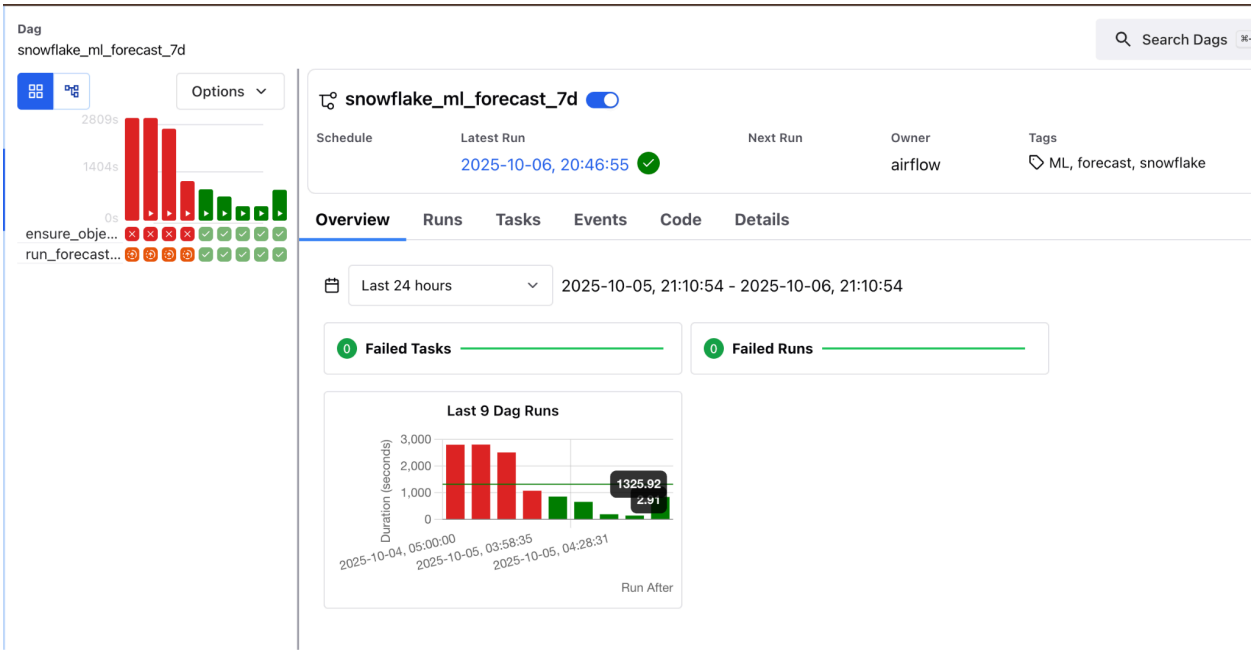
Table Name	Key Fields	Other Columns	Description
`STOCK_PRICES_180D`	`date` (PK), `symbol` (PK)	`open, close, min, max, volume`	Stores the daily 180-day history pulled from yfinance

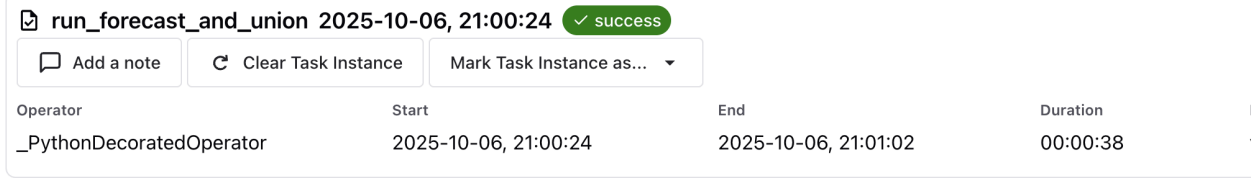
STOCK_PRICE_FORECAST_7D	date (PK), symbol (PK)	open, close, min, max, volume	Stores the 7+ day forecast generated by Snowflake ML.
STOCK_PRICES_FINAL	date (PK), symbol (PK)	open, close, min, max, volume	The final aggregated table (historical UNION prediction).

- **5.2 Screenshots :** Included Airflow Dags, logs, airflow variables, airflow connection, and final union table screenshots.





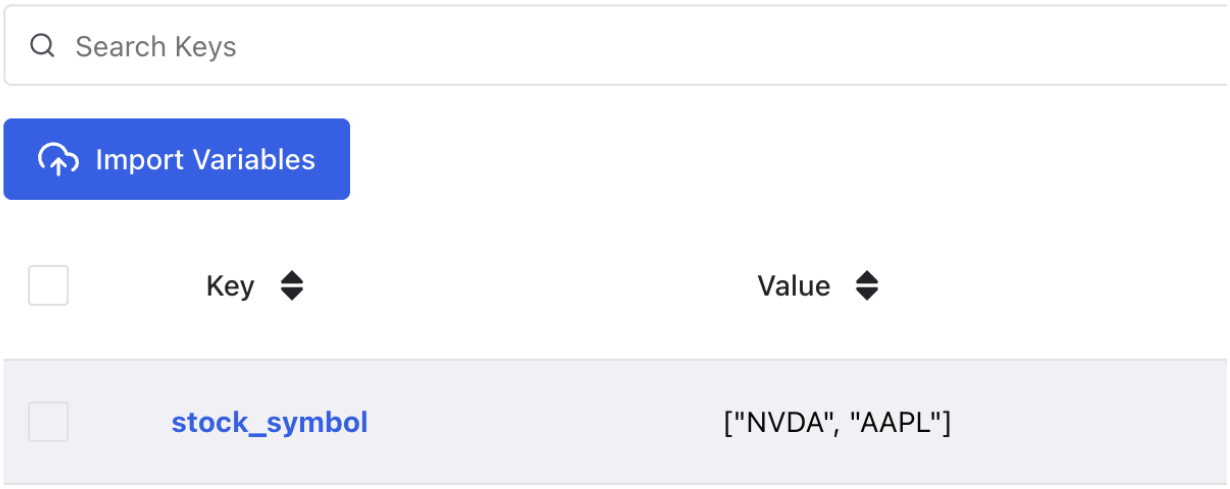


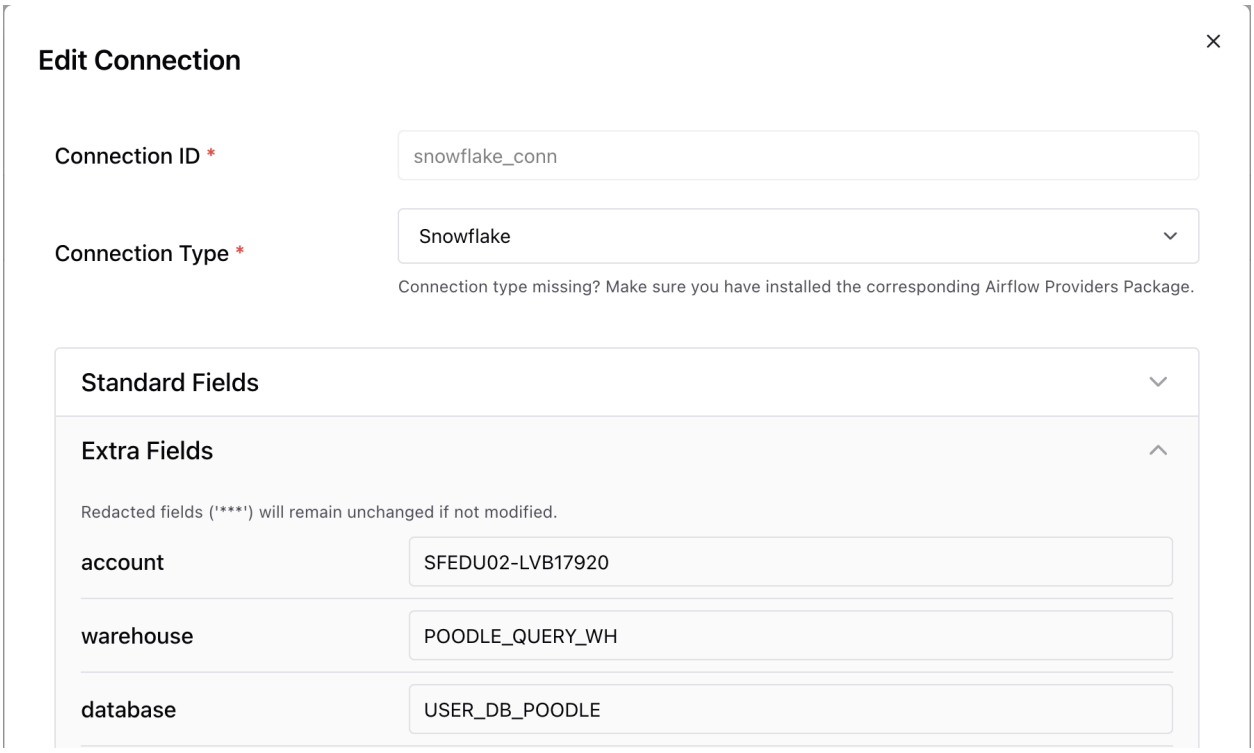


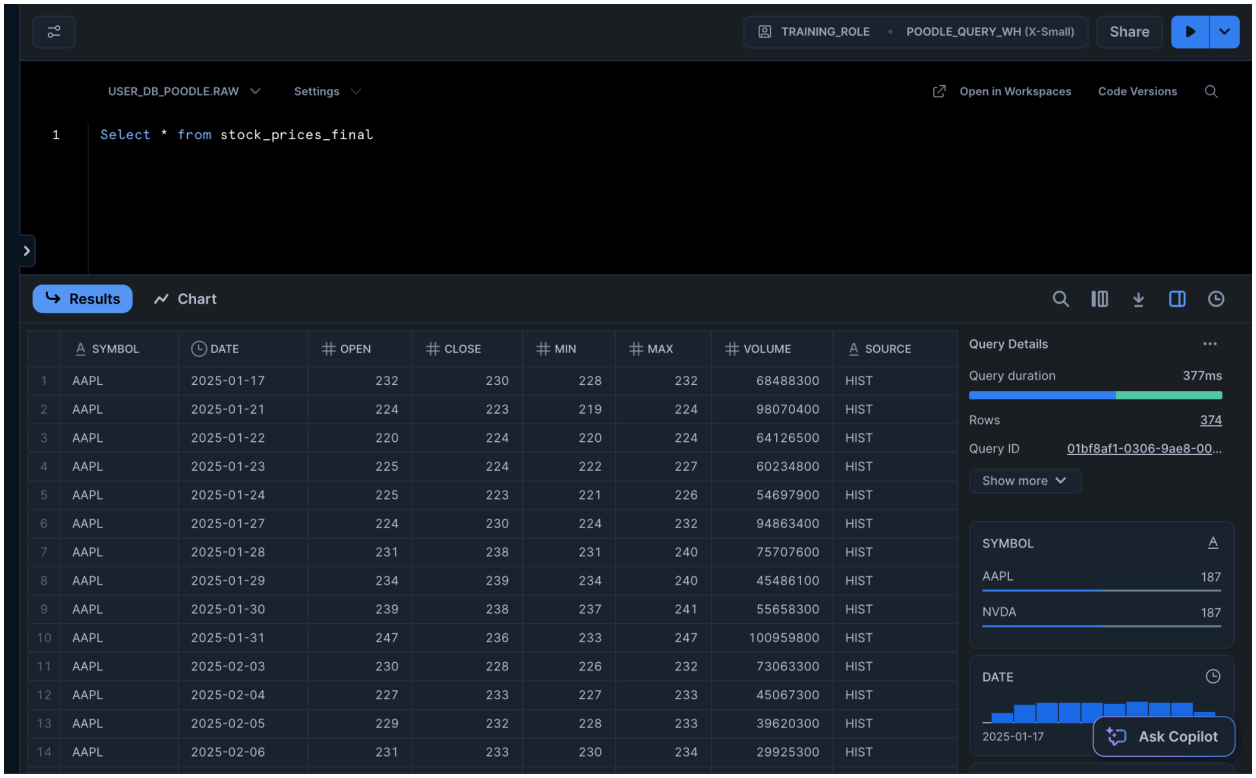
Logs | Rendered Templates | XCom | Audit Logs | Code | Details | Asset Events

All Log Levels | All Sources | Wrap

```
▶ Log message source details: sources=["/opt/airflow/logs/dag_id=snowflake_ml_forecast_7d/run_id=manual__2025-10-07T03:46:55.227050+00:00/
  2  [2025-10-06, 21:00:24] INFO - DAG bundles loaded: dags-folder, example_dags: source="airflow.dag_processing.bundles.manager.DagBund
  3  [2025-10-06, 21:00:24] INFO - Filling up the DagBag from /opt/airflow/dags/dag_snowflake_ml_forecast_7d.py: source="airflow.models.
  4  [2025-10-06, 21:00:25] INFO - Task instance is in running state: chan="stdout": source="task"
  5  [2025-10-06, 21:00:25] INFO -  Previous state of the Task instance: TaskInstanceState.QUEUED: chan="stdout": source="task"
  6  [2025-10-06, 21:00:25] INFO - Current task name:run_forecast_and_union: chan="stdout": source="task"
  7  [2025-10-06, 21:00:25] INFO - Dag name:snowflake_ml_forecast_7d: chan="stdout": source="task"
  8  [2025-10-06, 21:00:25] WARNING - /home/airflow/.local/lib/python3.12/site-packages/airflow/models/connection.py:471: DeprecationWar
       warnings.warn(
     : source="py.warnings"
  9  [2025-10-06, 21:00:25] INFO - Connection Retrieved 'snowflake_conn': source="airflow.hooks.base"
 10  [2025-10-06, 21:00:25] INFO - Snowflake Connector for Python Version: 3.17.2, Python Version: 3.12.11, Platform: Linux-6.10.14-linu
 11  [2025-10-06, 21:00:25] INFO - Connecting to GLOBAL Snowflake domain: source="snowflake.connector.connection"
 12  [2025-10-06, 21:00:58] INFO - Forecast data committed successfully: chan="stdout": source="task"
 13  [2025-10-06, 21:01:02] INFO - Final table populated. Counts by source: [(14, 'FORECAST'), (360, 'HIST')]: chan="stdout": source="ta
 14  [2025-10-06, 21:01:02] INFO - Done. Returned value was: None: source="airflow.task.operators.airflow.providers.standard.decorators.
 15  [2025-10-06, 21:01:02] INFO - Task instance in success state: chan="stdout": source="task"
 16  [2025-10-06, 21:01:02] INFO -  Previous state of the Task instance: TaskInstanceState.RUNNING: chan="stdout": source="task"
 17  [2025-10-06, 21:01:02] INFO - Task operator:<Task(_PythonDecoratedOperator): run_forecast_and_union>: chan="stdout": source="task"
```


Search Keys
Import Variables
Key
Value
stock_symbol
["NVDA", "AAPL"]


Edit Connection
Connection ID *
snowflake_conn
Connection Type *
Snowflake
Connection type missing? Make sure you have installed the corresponding Airflow Providers Package.
Standard Fields
Extra Fields
Redacted fields ('***') will remain unchanged if not modified.
account
SFEDU02-LVB17920
warehouse
POODLE_QUERY_WH
database
USER_DB_POODLE



	SYMBOL	DATE	OPEN	CLOSE	MIN	MAX	VOLUME	SOURCE
1	AAPL	2025-01-17	232	230	228	232	68488300	HIST
2	AAPL	2025-01-21	224	223	219	224	98070400	HIST
3	AAPL	2025-01-22	220	224	220	224	64126500	HIST
4	AAPL	2025-01-23	225	224	222	227	60234800	HIST
5	AAPL	2025-01-24	225	223	221	226	54697900	HIST
6	AAPL	2025-01-27	224	230	224	232	94863400	HIST
7	AAPL	2025-01-28	231	238	231	240	75707600	HIST
8	AAPL	2025-01-29	234	239	234	240	45486100	HIST
9	AAPL	2025-01-30	239	238	237	241	55658300	HIST
10	AAPL	2025-01-31	247	236	233	247	100959800	HIST
11	AAPL	2025-02-03	230	228	226	232	73063300	HIST
12	AAPL	2025-02-04	227	233	227	233	45067300	HIST
13	AAPL	2025-02-05	229	232	228	233	39620300	HIST
14	AAPL	2025-02-06	231	233	230	234	29925300	HIST

- **5.3 Code Repository Link:**

  GITHUB REPO LINK :
  https://github.com/omkarrajale1499/Stock_Price_Prediction_Airflow.git

6. Conclusion

This proposal outlines a clear plan for the development of the **Stock Price Prediction System using Airflow**. The requirements analysis confirms the critical need for an automated daily forecasting solution, and the conceptual design provides a robust framework utilizing industry-standard tools (Airflow, Snowflake, Python). We are confident that this approach will yield a successful and impactful application that meets all specified requirements for daily, 7+ day stock price prediction.