

Stock Price Prediction Using Airflow and Snowflake: A Data Pipeline Approach

AKHILESH
018206812
Data 226 - LAB 1

Abstract

This project's main objective is to create a system that can use historical stock data from the last 180 days to forecast the stock values of specific firms over the course of the following seven days. The yfinance API is used to extract data, which is then processed and stored in Snowflake. Machine learning techniques are then used to estimate future stock values. Airflow DAGs automate every step of the process, from data extraction to forecasting, guaranteeing frequent updates and simple scalability for later usage.

1. Problem Statement

In the financial industry, stock price prediction is a crucial undertaking where precise forecasting may help investors make well-informed judgments. However, using past data to predict future stock values is a complicated process that calls for meticulous data analysis and model execution. By examining the historical stock prices retrieved from the yfinance API, this project seeks to forecast the future stock values of a chosen group of companies.

In order to do this, the system must:

- Collect and preserve information: Get the last 180 days' worth of stock price information for firms (like NVDA and AAPL).
- Project future prices: Using this information, forecast the stock prices for the upcoming seven days.
- Make the pipeline automated: For precise forecasts and constant updates, the entire procedure must be automated to run every day.
- Make use of a database: To facilitate simple querying and analysis, Snowflake must be used to store both the historical stock data and the predicted outcomes.
- Use Airflow to automate tasks: Airflow DAGs should be used to automate the data pipeline so that the system can operate on a periodic basis without human intervention.

Large dataset management, process automation, and system scalability to accommodate future changes in stocks or data sources all depend on a database and data pipeline.

2. Solution Requirements

The problem's solution will concentrate on combining multiple technologies, such as Airflow, Snowflake, and Yfinance, and it will adhere to following specifications:

1. Data Collection:

At least two businesses' stock data (NVDA and AAPL) will be gathered using the yfinance API. The last 180 days should be covered by the data.

The following information will be required: Date, Open Price, Close Price, Low Price, High Price, Volume, and Stock Symbol.

2. **Data Storage:**

A structured format will be used to store the retrieved stock data in Snowflake, a cloud-based data warehouse. For storing and querying stock data, Snowflake's scalability and efficiency are crucial for managing massive data volumes.

3. **Data Transformation and Automation:**

To make sure the data is clean and formatted correctly before being entered into Snowflake, data transformation is essential.

Airflow, a program for building and overseeing data workflows, will be used to automate this process. Every day, the system need to be able to retrieve data automatically and import it into Snowflake without the need for human involvement.

4. **Forecasting:**

The stock prices for the upcoming seven days will be predicted using forecasting models (such as LSTM and ARIMA) based on the data gathered. These forecasts will be computed by Snowflake's SQL queries.

5. **Integration of Data and Forecasting:**

In order to create a single table for further analysis, the system will automatically merge the historical stock data with the predicted outcomes in Snowflake.

6. **Error Handling:**

SQL transactions and try/except blocks will be used to handle any mistakes that occur during data collection, transformation, or forecasting. This will guarantee that failures are controlled and do not interfere with the process as a whole.

3. Functional Analysis

The system's functional elements are separated into modules, each of which is essential to the seamless transition of data from extraction to forecasting:

1. **Module for Data Collection:**

For stock data, a query will be made to the Yfinance API. The last 180 days' worth of stock data for the chosen companies must be downloaded by this module. Daily open, close, low, high, and volume data points will all be included in the data.

2. **Module for Data Transformation:**

The raw data from YFinance will be ready for Snowflake's use thanks to this module. Data will be prepared, cleaned, and arranged in a systematic manner (such as a Pandas DataFrame or a collection of tuples). This stage guarantees that the information satisfies Snowflake's storage needs.

3. **Pipeline for Airflow Data:**

The automated process execution will be overseen by Airflow. Two primary pipelines will be present:

- ETL Pipeline: Every day, the Snowflake database will get the stock data that has been extracted and formatted appropriately.
- Forecasting Pipeline: This pipeline will execute the forecasting models to make predictions about future stock prices once the data has been imported into Snowflake. Any further transformations needed for the finished product will likewise be handled by it.

4. Forecasting with Machine Learning:

Forecasts based on past data will be produced using SQL queries in Snowflake. To forecast stock values for the upcoming seven days, the forecasting will make use of simple time-series models. This part operates directly inside the database by integrating with the Snowflake design.

5. Data merging and integration:

The predicted values will be combined with the data gathered from the ETL pipeline. A combined table with past stock values and anticipated future prices will be the end result.

6. Handling Errors:

SQL transactions and Airflow's error handling features will be used to appropriately handle data collection or transformation errors. For instance, the system will attempt the action again or log the error for human review if a data fetch fails.

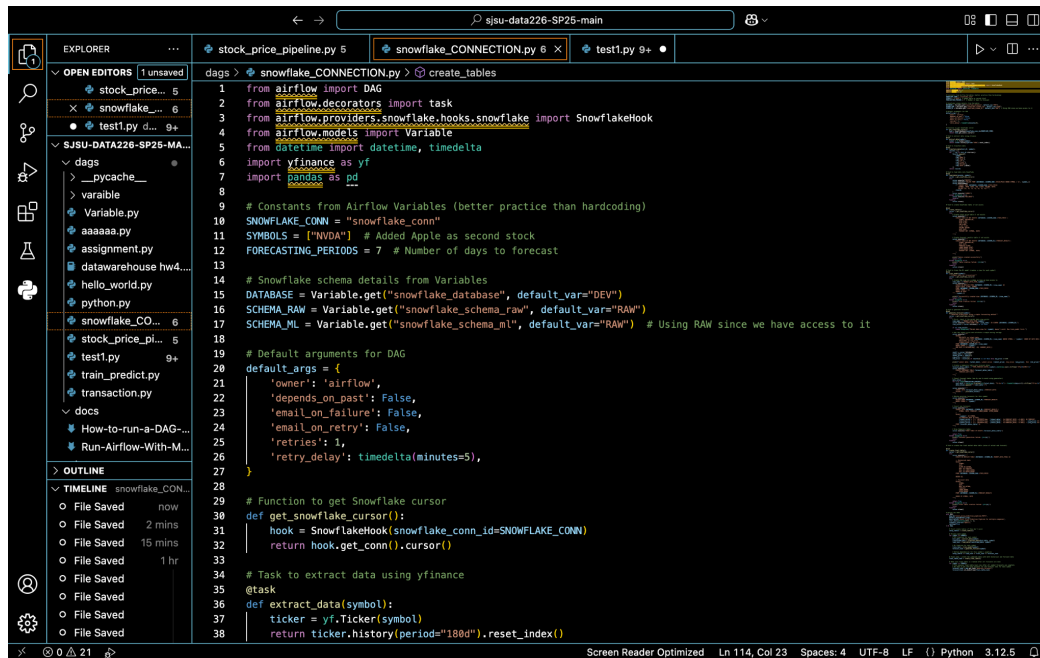
4. Tables Structure, Screenshots, Python Codes, and SQL Queries

The Snowflake table storing stock data will have the following structure:

Column Name	Data Type	Description
SYMBOL	VARCHAR	Stock ticker symbol (e.g., NVDA, AAPL)
DATE	DATE	Date of the stock price record
OPEN	FLOAT	Opening stock price for the day
CLOSE	FLOAT	Closing stock price for the day
LOW	FLOAT	Lowest stock price for the day
HIGH	FLOAT	Highest stock price for the day
VOLUME	INT	Trading volume on that day

Python Code: Airflow DAG for yfinance ETL Pipeline

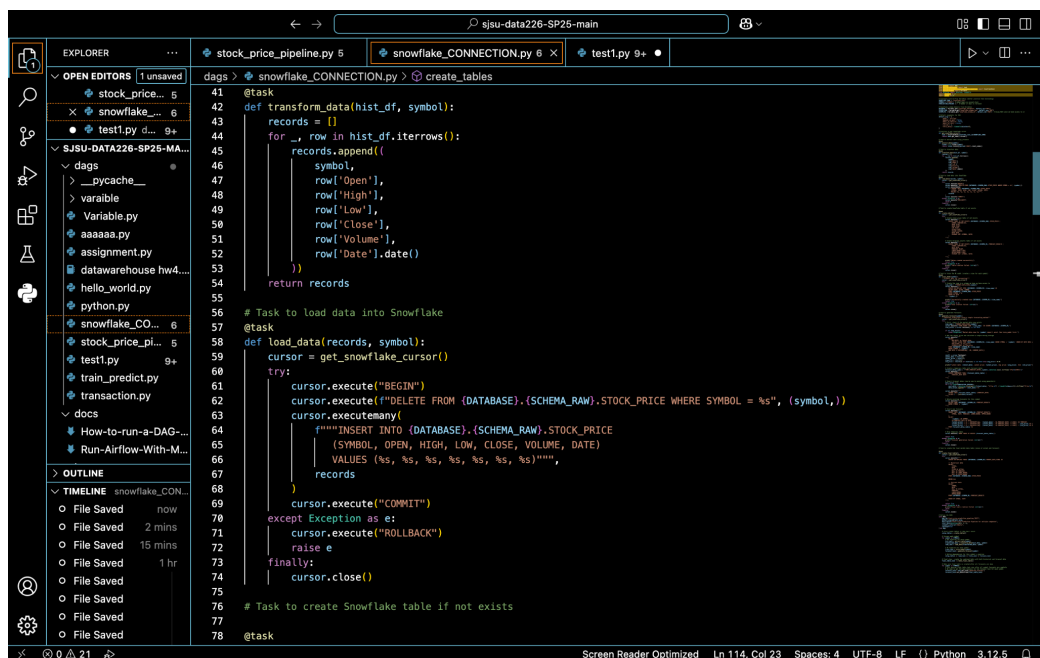
1. Data Extraction



The screenshot shows the VS Code editor with the file explorer on the left. The active file is `snowflake_CONNECTION.py`. The code defines a function `get_snowflake_cursor` and a task `extract_data` using the `@task` decorator. The task uses the `yfinance` library to fetch stock data for a list of symbols.

```
1 from airflow import DAG
2 from airflow.decorators import task
3 from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook
4 from airflow.models import Variable
5 from datetime import datetime, timedelta
6 import yfinance as yf
7 import pandas as pd
8
9 # Constants from Airflow Variables (better practice than hardcoding)
10 SNOWFLAKE_CONN = "snowflake_conn"
11 SYMBOLS = ["NVDA"] # Added Apple as second stock
12 FORECASTING_PERIODS = 7 # Number of days to forecast
13
14 # Snowflake schema details from Variables
15 DATABASE = Variable.get("snowflake_database", default_var="DEV")
16 SCHEMA_RAW = Variable.get("snowflake_schema_raw", default_var="RAW")
17 SCHEMA_ML = Variable.get("snowflake_schema_ml", default_var="RAW") # Using RAW since we have access to it
18
19 # Default arguments for DAG
20 default_args = {
21     'owner': 'airflow',
22     'depends_on_past': False,
23     'email_on_failure': False,
24     'email_on_retry': False,
25     'retries': 1,
26     'retry_delay': timedelta(minutes=5),
27 }
28
29 # Function to get Snowflake cursor
30 def get_snowflake_cursor():
31     hook = SnowflakeHook(snowflake_conn_id=SNOWFLAKE_CONN)
32     return hook.get_conn().cursor()
33
34 # Task to extract data using yfinance
35 @task
36 def extract_data(symbol):
37     ticker = yf.Ticker(symbol)
38     return ticker.history(period="180d").reset_index()
```

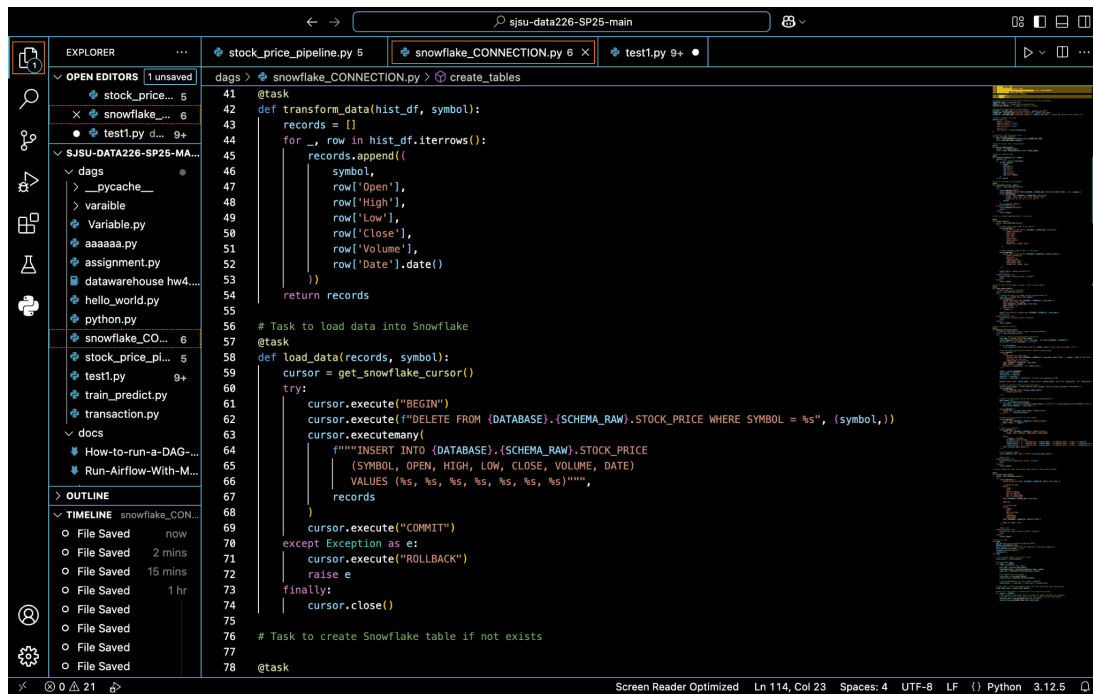
2. Data Transform



The screenshot shows the VS Code editor with the file explorer on the left. The active file is `snowflake_CONNECTION.py`. The code defines a function `transform_data` and a task `load_data` using the `@task` decorator. The task uses the `transform_data` function to transform the stock data into a list of records and then loads them into the Snowflake database.

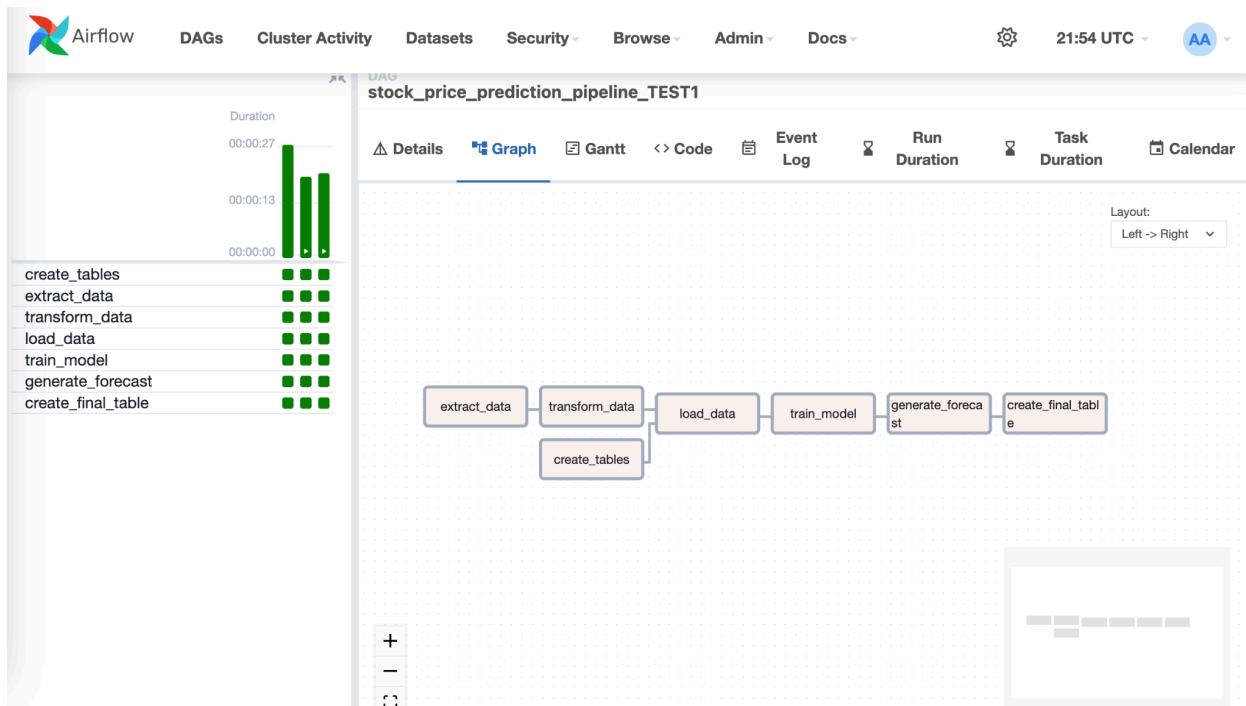
```
41 @task
42 def transform_data(hist_df, symbol):
43     records = []
44     for _, row in hist_df.iterrows():
45         records.append([
46             symbol,
47             row['Open'],
48             row['High'],
49             row['Low'],
50             row['Close'],
51             row['Volume'],
52             row['Date'].date()
53         ])
54     return records
55
56 # Task to load data into Snowflake
57 @task
58 def load_data(records, symbol):
59     cursor = get_snowflake_cursor()
60     try:
61         cursor.execute("BEGIN")
62         cursor.execute(f"DELETE FROM {DATABASE}.{SCHEMA_RAW}.STOCK_PRICE WHERE SYMBOL = %s", (symbol,))
63         cursor.executemany(
64             f"INSERT INTO {DATABASE}.{SCHEMA_RAW}.STOCK_PRICE "
65             f"({SYMBOL}, OPEN, HIGH, LOW, CLOSE, VOLUME, DATE) "
66             f"VALUES (%s, %s, %s, %s, %s, %s, %s)",
67             records
68         )
69         cursor.execute("COMMIT")
70     except Exception as e:
71         cursor.execute("ROLLBACK")
72         raise e
73     finally:
74         cursor.close()
75
76 # Task to create Snowflake table if not exists
77 @task
78 def create_table():
79     cursor = get_snowflake_cursor()
80     cursor.execute(f"CREATE TABLE IF NOT EXISTS {DATABASE}.{SCHEMA_RAW}.STOCK_PRICE (
81         SYMBOL VARCHAR(10),
82         OPEN FLOAT,
83         HIGH FLOAT,
84         LOW FLOAT,
85         CLOSE FLOAT,
86         VOLUME INT,
87         DATE DATE
88     );")
89     cursor.close()
```

3. Data Loading



4. Airflow UI Screenshot:

Screenshot of the Airflow Web UI showing both yfinance ETL pipeline and forecasting pipeline



5. SQL Query:

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with the following hierarchy:

- DEV
 - ANALYTICS
 - INFORMATION_SCHEMA
 - PUBLIC
 - RAW
 - Tables
 - FORECAST_RESULTS
 - MARKET_DATA_FINAL
 - STOCK_PRICE
 - Views
 - MARKET_DATA_VIEW
 - MARKET_DATA_VIEW_NVDA
 - SNOWFLAKE
 - SNOWFLAKE_SAMPLE_DATA

The main editor area contains the following SQL code:

```
5 CREATE TABLE DEV.RAW.STOCK_PRICE (  
6     SYMBOL STRING,  
7     OPEN FLOAT,  
8     HIGH FLOAT,  
9     LOW FLOAT,  
10    CLOSE FLOAT,  
11    VOLUME BIGINT,  
12    DATE DATE  
13 );  
14  
15  
16 CREATE TABLE IF NOT EXISTS DEV.ANALYTICS.MARKET_DATA_FINAL (  
17     SYMBOL STRING,  
18     DATE DATE,  
19     ACTUAL FLOAT,  
20     FORECAST FLOAT,  
21     LOWER_BOUND FLOAT,  
22     UPPER_BOUND FLOAT,  
23     PRIMARY KEY (SYMBOL, DATE)  
24 );  
25  
26  
27 CREATE TABLE IF NOT EXISTS DEV.ANALYTICS.PREDICT_STOCK_PRICE (  
28     SYMBOL STRING,  
29     DATE DATE,  
30     FORECAST FLOAT,  
31     LOWER_BOUND FLOAT,  
32     UPPER_BOUND FLOAT  
33 );
```

6. Snowflake table data:

The screenshot shows the Snowflake SQL Editor interface with the following SQL code:

```
27     SYMBOL STRING,  
28     DATE DATE,  
29     FORECAST FLOAT,  
30     LOWER_BOUND FLOAT,  
31     UPPER_BOUND FLOAT  
32 );  
33  
34 SELECT * FROM DEV.ANALYTICS.MARKET_DATA_FINAL;  
35  
36  
37 SELECT * FROM DEV.ANALYTICS.MARKET_DATA_FINAL;
```

The query results are displayed in a table with the following columns: SYMBOL, OPEN, HIGH, LOW, CLOSE. The results show data for NVDA stock across 7 rows.

	SYMBOL	OPEN	HIGH	LOW	CLOSE
2	NVDA	131.119749917	136.308950964	130.669822453	135.55906677
3	NVDA	139.778407023	140.73825018	129.499996277	130.75979614
4	NVDA	127.1003713	130.609831378	124.280807104	126.55045318
5	NVDA	123.220963662	124.440776456	118.021769845	118.09175872
6	NVDA	121.18127449	126.48045882	119.301567651	126.07051849
7	NVDA	126.110522951	128.100213561	122.581069198	126.38048553

Query Details:

- Query duration: 41ms
- Rows: 180
- Query ID: 01bad5b0-0004-b4f6-0...

7. Github Link :

<https://github.com/AkhileshTandur/Stock-Price-Prediction-Analytics>

5. Conclusion

This research successfully automates the stock price prediction process by combining a number of potent tools and technology. The solution can manage big datasets and generate precise forecasts by combining yfinance for data collecting, Airflow for automation, and Snowflake for data storage and analysis. While Snowflake provides the scale required to enable complicated queries and forecasting models, Airflow guarantees that the entire process operates automatically without human involvement. In order to assist stakeholders in making well-informed investment decisions, the final output offers practical insights on stock price movements.

References

1. yfinance: <https://pypi.org/project/yfinance/>
2. Airflow Documentation: <https://airflow.apache.org/>
3. Snowflake Documentation: <https://docs.snowflake.com/>

