Data 226 Lab 2

Yilin Sun, Yongxin Li

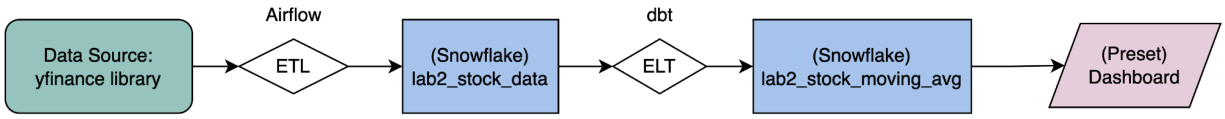
Problem Statement

The objective of this lab is to design and implement an automated and scalable data pipeline for stock market data analysis. The system extracts historical stock data for the past 180 days using a public API, loads it into Snowflake for structured storage, transforms the data to calculate 7-day and 30-day moving averages using dbt, and visualizes the analysis in a Preset dashboard.

Requirements and Specifications

- **Data Source**: Use yfinance library to fetch 180 days of stock data
- **Storage**: Transforms raw data into tabular structures in Snowflake
- **ELT**: Use dbt to transform raw data by computing 7-day and 30-day moving averages
- **Pipeline Orchestration**: Use Airflow for scheduling and monitoring
- **Visualization**: Build interactive dashboards in Preset
- **Idempotency**: Use SQL transactions with error handling
- **Reusability**: Use Airflow connections and variables for flexibility

Overall System Diagram



The system consists of:

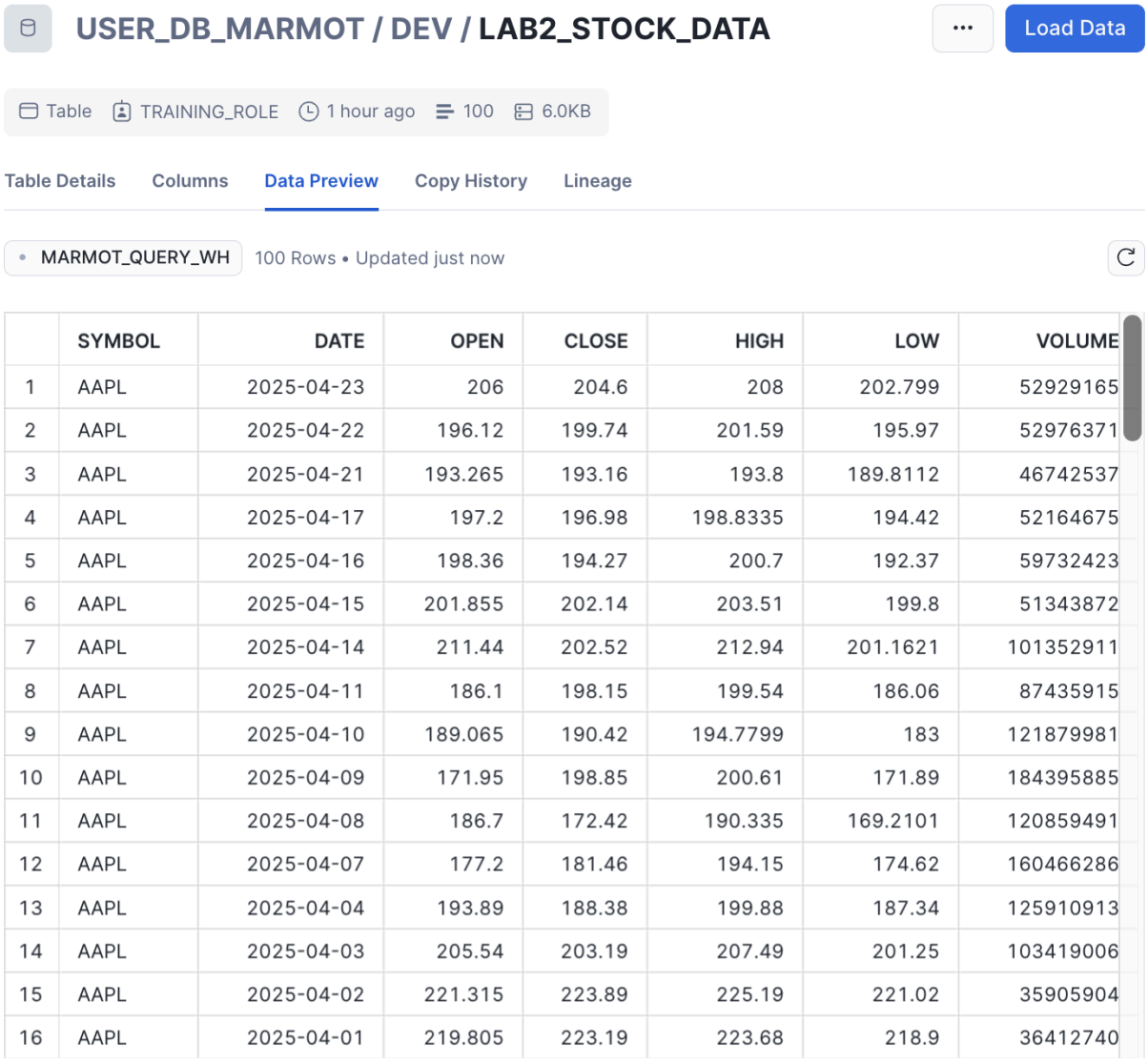
- **Snowflake tables**: lab2_stock_data (under ‘dev’) and lab2_stock_moving_avg (under ‘dev’)
- **Airflow DAG**: handles orchestration of ETL tasks
- **dbt models**: perform ELT for moving averages
- **Preset dashboards**: enable data visualization and analysis

Snowflake Tables

lab2_stock_data

- Purpose: stores raw daily stock data fetched from yfinance
- Schema: contains fields for stock symbols (AAPL selected), dates, OHLC (open, high, low, close) prices along with trading volume

```
CREATE OR REPLACE TABLE TABLE USER_DB_MARMOT.DEV.LAB2_STOCK_DATA (
        SYMBOL VARCHAR(16777216),
        DATE DATE,
        OPEN FLOAT,
        CLOSE FLOAT,
        HIGH FLOAT,
        LOW FLOAT,
        VOLUME NUMBER(38,0)
);
```

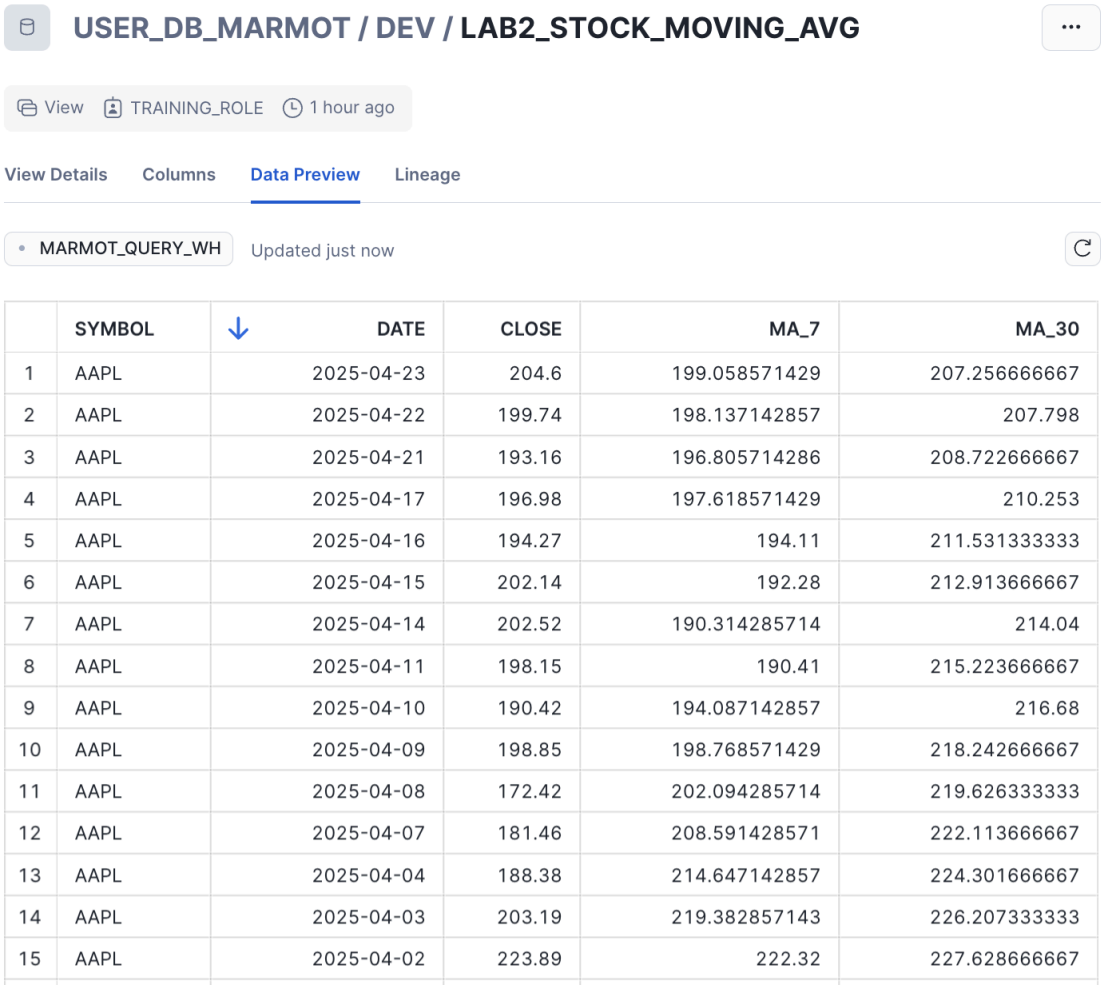


	SYMBOL	DATE	OPEN	CLOSE	HIGH	LOW	VOLUME
1	AAPL	2025-04-23	206	204.6	208	202.799	52929165
2	AAPL	2025-04-22	196.12	199.74	201.59	195.97	52976371
3	AAPL	2025-04-21	193.265	193.16	193.8	189.8112	46742537
4	AAPL	2025-04-17	197.2	196.98	198.8335	194.42	52164675
5	AAPL	2025-04-16	198.36	194.27	200.7	192.37	59732423
6	AAPL	2025-04-15	201.855	202.14	203.51	199.8	51343872
7	AAPL	2025-04-14	211.44	202.52	212.94	201.1621	101352911
8	AAPL	2025-04-11	186.1	198.15	199.54	186.06	87435915
9	AAPL	2025-04-10	189.065	190.42	194.7799	183	121879981
10	AAPL	2025-04-09	171.95	198.85	200.61	171.89	184395885
11	AAPL	2025-04-08	186.7	172.42	190.335	169.2101	120859491
12	AAPL	2025-04-07	177.2	181.46	194.15	174.62	160466286
13	AAPL	2025-04-04	193.89	188.38	199.88	187.34	125910913
14	AAPL	2025-04-03	205.54	203.19	207.49	201.25	103419006
15	AAPL	2025-04-02	221.315	223.89	225.19	221.02	35905904
16	AAPL	2025-04-01	219.805	223.19	223.68	218.9	36412740

Lab2_stock_moving_avg

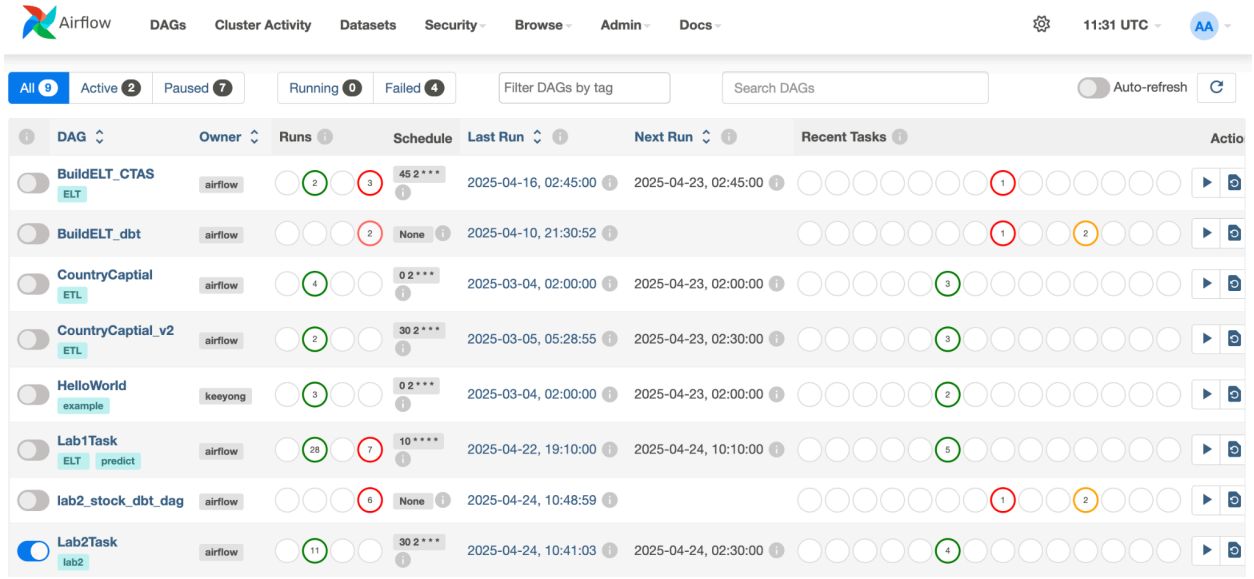
- Purpose: calculates and stores moving averages 7 days and 30 days for stock closing prices

```
CREATE OR REPLACE TABLE USER_DB_MARMOT.DEV.LAB2_STOCK_MOVING_AVG(
       SYMBOL,
       DATE,
       CLOSE,
       MA_7,
       MA_30
) as (
    SELECT
    symbol,
    date,
    close,
    AVG(close) OVER (PARTITION BY symbol ORDER BY date ROWS BETWEEN 6 PRECEDING AND CURRENT
ROW) AS ma_7,
    AVG(close) OVER (PARTITION BY symbol ORDER BY date ROWS BETWEEN 29 PRECEDING AND CURRENT
ROW) AS ma_30
FROM USER_DB_MARMOT.dev.lab2_stock_data
ORDER BY symbol, date
  );
```



	SYMBOL	DATE	CLOSE	MA_7	MA_30
1	AAPL	2025-04-23	204.6	199.058571429	207.256666667
2	AAPL	2025-04-22	199.74	198.137142857	207.798
3	AAPL	2025-04-21	193.16	196.805714286	208.722666667
4	AAPL	2025-04-17	196.98	197.618571429	210.253
5	AAPL	2025-04-16	194.27	194.11	211.531333333
6	AAPL	2025-04-15	202.14	192.28	212.913666667
7	AAPL	2025-04-14	202.52	190.314285714	214.04
8	AAPL	2025-04-11	198.15	190.41	215.223666667
9	AAPL	2025-04-10	190.42	194.087142857	216.68
10	AAPL	2025-04-09	198.85	198.768571429	218.242666667
11	AAPL	2025-04-08	172.42	202.094285714	219.626333333
12	AAPL	2025-04-07	181.46	208.591428571	222.113666667
13	AAPL	2025-04-04	188.38	214.647142857	224.301666667
14	AAPL	2025-04-03	203.19	219.382857143	226.207333333
15	AAPL	2025-04-02	223.89	222.32	227.628666667

Airflow DAG



lab2.py (simplified) – airflow dag for Snowflake data

- Functionality: uses airflow's 'SnowflakeHook' to connect to the Snowflake database, ensures secure and reusable connections
- ELT task: calculates moving averages and inserts results into lab2_stock_moving_avg
- DAG(): defines pipeline's tasks and execution order, uses task decorators for modularity and schedules dag to run daily.

```
# Importing
from ... import ...

# Defining Snowflake connection
def return_Snowflake_conn():
    # Initialize the SnowflakeHook
    hook = SnowflakeHook(Snowflake_conn_id='Snowflake_conn')
    # Execute the query and fetch results
    conn = hook.get_conn()
    return conn.cursor()

@task
def extract(apikey, num_of_days, stock_symbol):
    ...
@task
def transform(input_data, num_of_days, stock_symbol):
    ...
@task
def load(cursor, target_table, stock_data_input):
    try:
        ...
    except Exception as e:
```

```
#########
## ELT ##
#########
@task
def calculate_moving_averages(cursor, source_table, target_table):
   # This ELT task calculates the moving average of past 7 and 30 days and insert them into
a new table
   try:
       cursor.execute("BEGIN;")

       cursor.execute(f"""
           CREATE OR REPLACE TABLE user_db_marmot.dev.{target_table} (
               symbol STRING,
               date DATE,
               close FLOAT,
               ma_7 FLOAT,
               ma_30 FLOAT,
               PRIMARY KEY (symbol, date)
           );
       """)

       # Insert with moving averages (Snowflake SQL with window functions)
       cursor.execute(f"""
           INSERT INTO user_db_marmot.dev.{target_table}
           SELECT
               symbol,
               date,
               close,
               AVG(close) OVER (PARTITION BY symbol ORDER BY date ROWS BETWEEN 6 PRECEDING
AND CURRENT ROW) AS ma_7,
               AVG(close) OVER (PARTITION BY symbol ORDER BY date ROWS BETWEEN 29 PRECEDING
AND CURRENT ROW) AS ma_30
           FROM user_db_marmot.dev.{source_table}
           ORDER BY symbol, date;
       """)

       cursor.execute("COMMIT;")

   except Exception as e:
       cursor.execute("ROLLBACK;")
       print("Error in ELT:", e)
       raise(e)

with DAG(
   dag_id = 'Lab2Task',
   start_date = datetime(2025,4,20),
   catchup=False,
   tags=['lab2'],
   schedule = '30 2 * * *'
) as dag:

   data = extract(api_key, num_of_days, stock_symbol)
```

```
transformed_data = transform(data, num_of_days, stock_symbol)
load_task = load(cursor, target_table, transformed_data)
elt_task = calculate_moving_averages(cursor, target_table, "lab2_stock_moving_avg")

# Dependency
data >> transformed_data >> load_task >> elt_task
```

dbt – lab2_stock_data.sql, lab2_stock_moving_avg.sql, lab2_stock_data_snapshot.sql, schema.yml, sources.yml, dbt_project.yml

- lab2_stock_data.sql – retrieves raw stock data from Snowflake for dbt transformations.

```
SELECT symbol, date, open, close, high, low, volume
FROM {{ source('dev', 'lab2_stock_data') }}
```

lab2_stock_moving_avg.sql – computes moving averages for stock prices using Snowflake SQL

```
SELECT symbol, date, close,
   AVG(close) OVER (PARTITION BY symbol ORDER BY date ROWS BETWEEN 6 PRECEDING AND CURRENT
ROW) AS ma_7,
   AVG(close) OVER (PARTITION BY symbol ORDER BY date ROWS BETWEEN 29 PRECEDING AND CURRENT
ROW) AS ma_30
FROM {{ ref('lab2_stock_data') }}
ORDER BY symbol, date
```

- lab2_stock_data_snapshot.sql – track changes in lab2_stock_data over time

```
{% snapshot lab2_stock_moving_avg_snapshot %}
```

```
{{
 config(
   target_schema='snapshot',
   unique_key=['symbol', 'date'],
   strategy='check',
   check_cols=['close']
 )
}}

SELECT
   *,
   ROW_NUMBER() OVER (PARTITION BY symbol, date ORDER BY date) AS row_id
FROM {{ source('dev', 'lab2_stock_data') }}

{% endsnapshot %}
```

schema.yml

```
version: 2
models:
 - name: lab2_stock_data
   description: "Stock data model"
   columns:
     - name: symbol
       description: "Stock symbol"
     - name: date
       description: "Date of stock data"
     - name: close
       description: "Closing price"
```

sources.yml

```
version: 2

sources:
 - name: dev
   database: USER_DB_MARMOT
   schema: dev
   tables:
     - name: lab2_stock_data
       description: "Raw daily stock prices including open, close, high, low, and volume
data."
       columns:
         - name: symbol
           description: "Stock ticker symbol (e.g., AAPL)."
         - name: date
           description: "Date of the stock data."
         - name: close
           description: "Closing price of the stock."
```

```
      - name: open
        description: "Opening price of the stock."
      - name: high
        description: "Highest price of the stock for the day."
      - name: low
        description: "Lowest price of the stock for the day."
      - name: volume
        description: "Number of shares traded for the day."
```

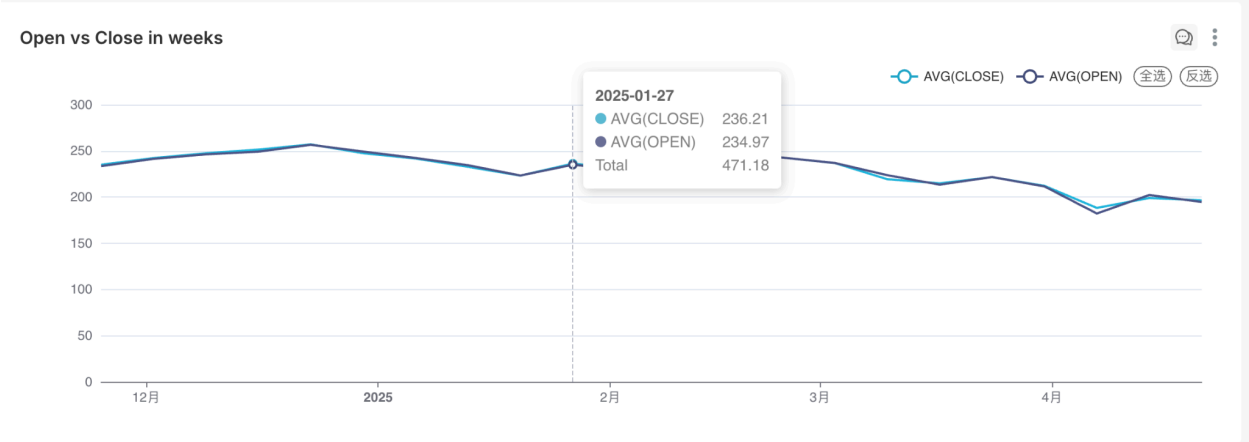
dbt_project.yml

```
name: 'dbt_lab2_1'
version: '1.0.0'
profile: 'dbt_lab2_1'
model-paths: ["models"]
analysis-paths: ["analyses"]
test-paths: ["tests"]
seed-paths: ["seeds"]
macro-paths: ["macros"]
snapshot-paths: ["snapshots"]
clean-targets:
 - "target"
 - "dbt_packages"
models:
 dbt_lab2_1:
   +materialized: view
   output:
     +materialized: table
   input:
     +materialized: ephemeral
```

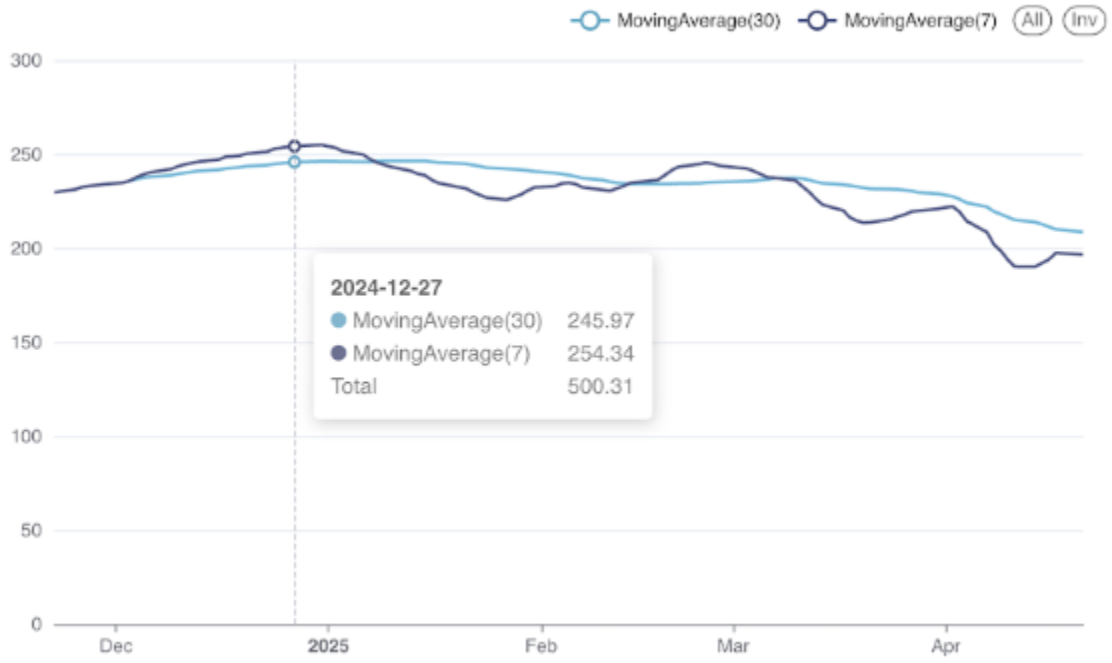
```
.venv(base) leali@lixixis-body-2 dbt_lab2_1 % dbt test
10:57:17  Running with dbt=1.9.3
10:57:18  Registered adapter: snowflake=1.9.2
10:57:18  [WARNING]: Configuration paths exist in your dbt_project.yml file which do not apply to any resources.
There are 2 unused configuration paths:
- models.dbt_lab2_1.input
- models.dbt_lab2_1.output
10:57:18  Found 2 models, 1 snapshot, 1 source, 474 macros
10:57:18  Nothing to do. Try checking your model configs and model specification args
.venv(base) leali@lixixis-body-2 dbt_lab2_1 % dbt run
10:57:21  Running with dbt=1.9.3
10:57:21  Registered adapter: snowflake=1.9.2
10:57:22  [WARNING]: Configuration paths exist in your dbt_project.yml file which do not apply to any resources.
There are 2 unused configuration paths:
- models.dbt_lab2_1.input
- models.dbt_lab2_1.output
10:57:22  Found 2 models, 1 snapshot, 1 source, 474 macros
10:57:22
10:57:22  Concurrency: 1 threads (target='dev')
10:57:22
10:57:23  1 of 2 START sql table model dev.lab2_stock_data ............................... [RUN]
10:57:24  1 of 2 OK created sql table model dev.lab2_stock_data .......................... [SUCCESS 1 in 0.82s]
10:57:24  2 of 2 START sql view model dev.lab2_stock_moving_avg .......................... [RUN]
10:57:24  2 of 2 OK created sql view model dev.lab2_stock_moving_avg ..................... [SUCCESS 1 in 0.37s]
10:57:24
10:57:24  Finished running 1 table model, 1 view model in 0 hours 0 minutes and 2.31 seconds (2.31s).
10:57:24
10:57:24  Completed successfully
10:57:24
10:57:24  Done. PASS=2 WARN=0 ERROR=0 SKIP=0 TOTAL=2
.venv(base) leali@lixixis-body-2 dbt_lab2_1 % dbt snapshot
10:57:30  Running with dbt=1.9.3
10:57:30  Registered adapter: snowflake=1.9.2
10:57:30  [WARNING]: Configuration paths exist in your dbt_project.yml file which do not apply to any resources.
There are 2 unused configuration paths:
- models.dbt_lab2_1.input
- models.dbt_lab2_1.output
10:57:30  Found 2 models, 1 snapshot, 1 source, 474 macros
10:57:30
10:57:30  Concurrency: 1 threads (target='dev')
10:57:30
10:57:31  1 of 1 START snapshot snapshot.lab2_stock_moving_avg_snapshot .................. [RUN]
10:57:35  1 of 1 OK snapshotted snapshot.lab2_stock_moving_avg_snapshot .................. [SUCCESS 0 in 3.36s]
10:57:35
10:57:35  Finished running 1 snapshot in 0 hours 0 minutes and 4.40 seconds (4.40s).
10:57:35
10:57:35  Completed successfully
10:57:35
10:57:35  Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
```

Data Visualization

- Open vs Close: Use the lab2_stock_data table stored in Snowflake to compare the weekly average closing prices with the weekly average opening prices.



- Moving average: Compare the 30-day moving average and the 7-day moving average for the same day to analyze trends and variations in the data.



Summary

The project combines **Airflow** for pipeline orchestration, **Snowflake** for data storage, **dbt** for data transformation, and **Preset** for visualization. Each component serves a distinct purpose in ensuring a reliable and efficient data workflow.

Github link: https://github.com/lea2105/DATA226LAB2_Sun_Li