Lab 2: Building an end to end Data Analytics using Snowflake, Airflow, dbt, and a BI tool

Abhinita Sanabada
(018320874)

1. Introduction

1.1 Problem Statement

In modern data engineering, raw data is rarely ready for immediate analysis. The challenge addressed in this lab is to build a robust, automated End-to-End (E2E) data pipeline that ingests raw financial stock data, transforms it into meaningful technical indicators (like Moving Averages and RSI) using best-practice ELT methodologies, performs machine learning forecasting, and visualizes the results for business stakeholders. The system must ensure data quality, idempotency, and security while orchestrating dependencies between disparate tools (Airflow, Snowflake, dbt).

1.2 Requirements

- ETL: Automate the ingestion of stock data (OHLCV) from YFinance into Snowflake using Airflow.
- ELT: Use dbt (data build tool) to perform transformations within the data warehouse, creating abstract tables with technical indicators.
- Orchestration: Schedule ETL, ELT, and ML tasks sequentially using Airflow.
- Quality & Idempotency: Ensure data integrity using SQL transactions (MERGE) and dbt tests.
- Visualization: Create an interactive dashboard to monitor stock trends and buy/sell signals.

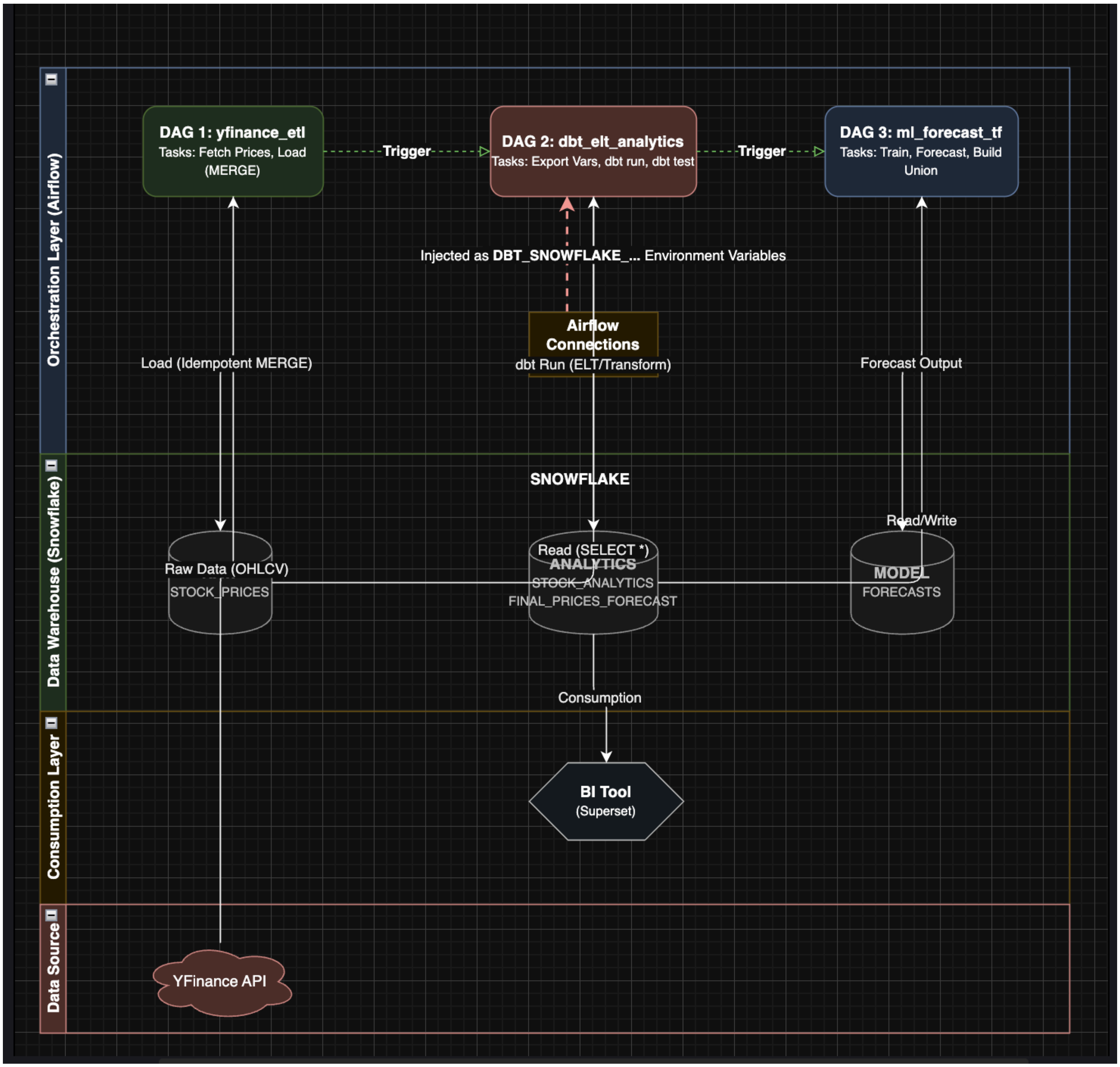
1.3 Specifications

- Cloud Data Warehouse: Snowflake (Schemas: RAW, ANALYTICS, MODEL)
- Orchestration: Apache Airflow (DAGs running in Docker)
- Transformation: dbt Core (running via BashOperator)
- BI Tool: [Superset / Preset]
- Language: Python 3.9+, SQL

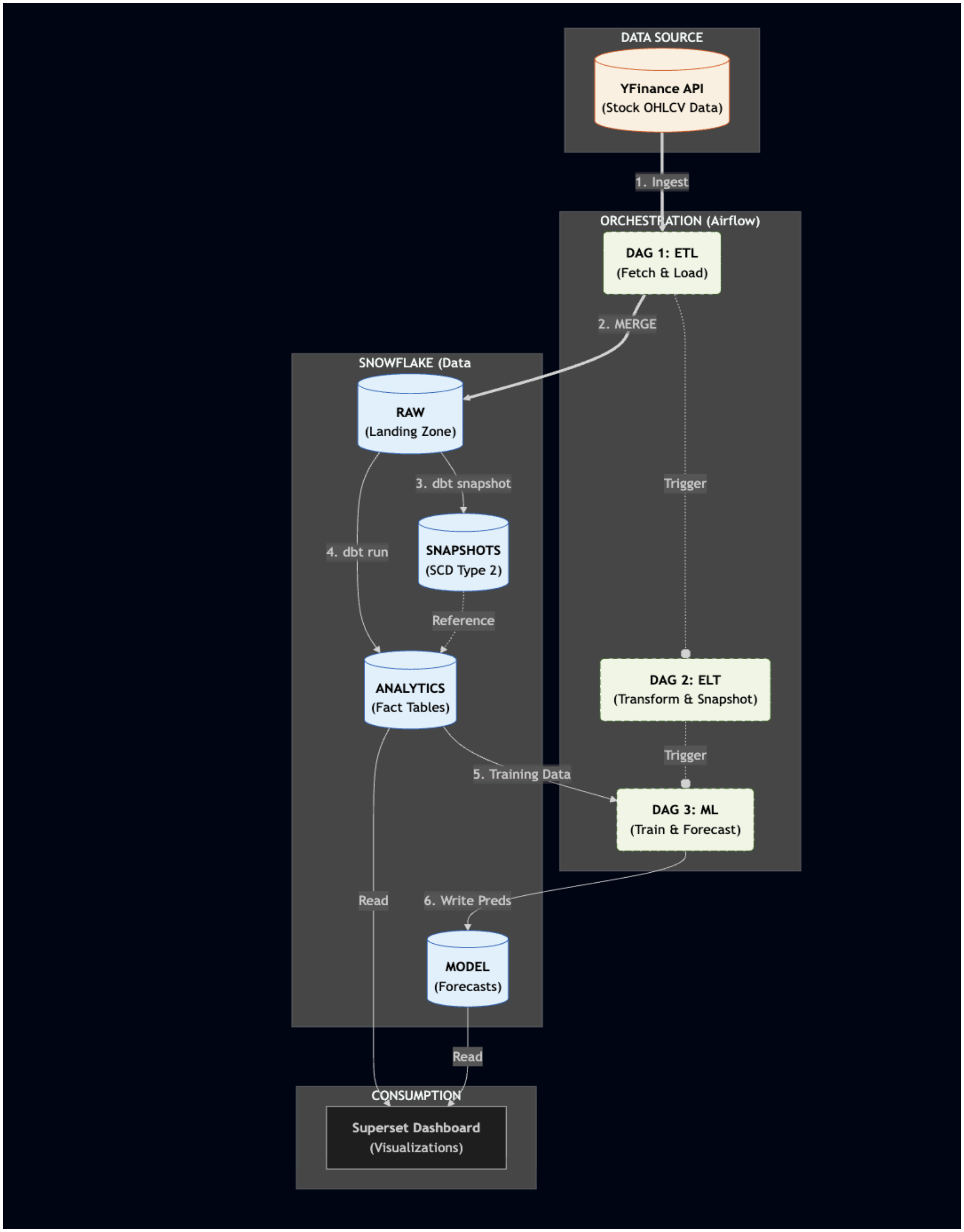
2. System Architecture

2.1 Overall System Diagram

The system follows a linear dependency chain: Ingest —>Transform —-> Predict —-> Visualize.



[System Diagram]



End To End work flow of the process as per system Diagram

2.2 Component Description

1. Airflow: Functions as the control plane. It manages three separate DAGs (`yfinance_etl`, `dbt_elt_analytics`, `ml_forecast_tf`) and handles the secure passing of credentials to dbt.
2. Snowflake: Acts as the storage and compute engine.
    - RAW Schema: Stores immutable raw data loaded by the ETL process.
    - ANALYTICS Schema: Stores the "Gold" layer data transformed by dbt for BI consumption.
3. dbt: Handles business logic transformations (Moving Averages, RSI) and data testing.
4. BI Tool(Preset): Connects to the `ANALYTICS` schema to display the final dashboard.

3. Detailed Table Structures

3.1 Source Schema: RAW

Table Name: STOCK_PRICES Description: Stores immutable raw data loaded by the ETL process.

- TRADE_DATE (Type: DATE) - *Primary Key (Composite)*. Date of the trading session.
- SYMBOL (Type: STRING) - *Primary Key (Composite)*. Stock Ticker (e.g., AAPL).
- OPEN (Type: DOUBLE) - Opening price.
- CLOSE (Type: DOUBLE) - Closing price.
- HIGH (Type: DOUBLE) - Highest price of the day.
- LOW (Type: DOUBLE) - Lowest price of the day.
- VOLUME (Type: NUMBER) - Number of shares traded.
- LOAD_TS (Type: TIMESTAMP_NTZ) - Time of insertion.

3.2 Destination Schema: ANALYTICS

Table Name: STOCK_ANALYTICS Description: Stores the "Gold" layer data transformed by dbt for BI consumption.

- TRADE_DATE (Type: DATE) - *Primary Key (Composite)*. Date of observation.
- SYMBOL (Type: STRING) - *Primary Key (Composite)*. Stock Ticker.
- CLOSE (Type: DOUBLE) - Closing price from source.
- MA_20 (Type: DOUBLE) - 20-Day Simple Moving Average. *Constraint: Not Null.*

- RSI_14 (Type: DOUBLE) - 14-Day Relative Strength Index. *Constraint: Check (0-100).*
- PRICE_MOMENTUM (Type: DOUBLE) - 10-Day price change.

3.3 Model Schema: MODEL

Table Name: FORECASTS Description: Stores machine learning predictions.

- SYMBOL (Type: STRING) - *Primary Key*.
- TS (Type: DATE) - *Primary Key*. Forecast timestamp.
- PREDICTED_CLOSE (Type: FLOAT) - The predicted value.
- MODEL_NAME (Type: STRING) - *Primary Key*. Name of model used (e.g., 'SNOWFLAKE_ML').

3.4 Data Validation

To verify the successful execution of the ELT pipeline, the following SQL query was executed in the Snowflake Worksheet against the final transformed table after all dag processes were done in `ANALYTICS.STOCK_ANALYTICS`.

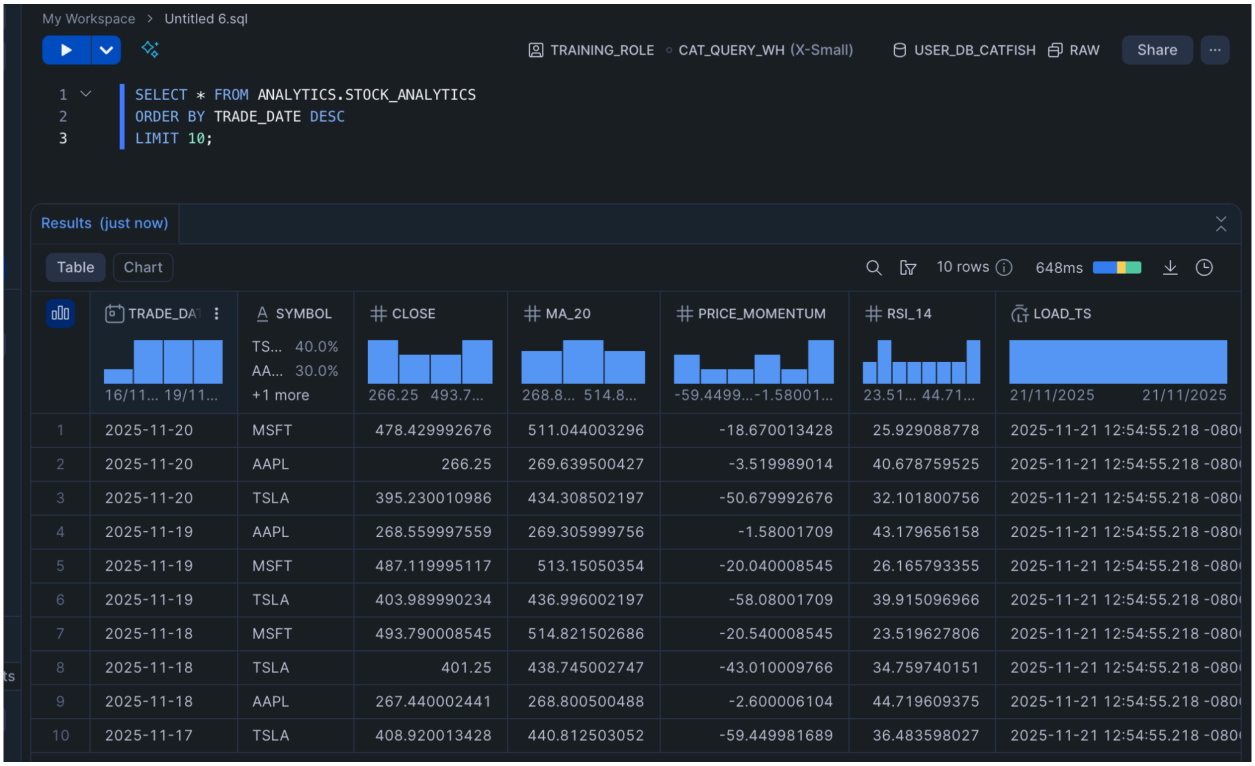
Query:

SELECT * FROM ANALYTICS.STOCK_ANALYTICS

ORDER BY TRADE_DATE DESC

LIMIT 10;

Observation: The screenshot below confirms that the table is populated with calculated fields, including the **20-day Moving Average (`MA_20`)** and **RSI (`RSI_14`)**, demonstrating that the dbt models transformed the raw data correctly.

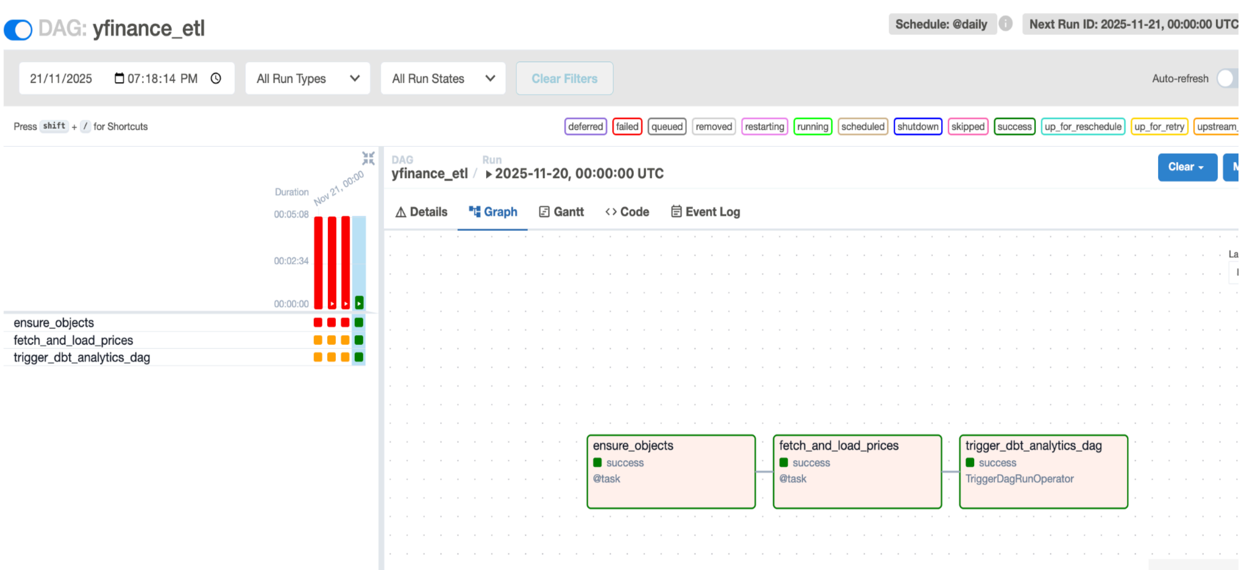


	TRADE_DATE	SYMBOL	CLOSE	MA_20	PRICE_MOMENTUM	RSI_14	LOAD_TS
1	2025-11-20	MSFT	478.429992676	511.044003296	-18.670013428	25.929088778	2025-11-21 12:54:55.218 -080
2	2025-11-20	AAPL	266.25	269.639500427	-3.519989014	40.678759525	2025-11-21 12:54:55.218 -080
3	2025-11-20	TSLA	395.230010986	434.308502197	-50.679992676	32.101800756	2025-11-21 12:54:55.218 -080
4	2025-11-19	AAPL	268.559997559	269.305999756	-1.58001709	43.179656158	2025-11-21 12:54:55.218 -080
5	2025-11-19	MSFT	487.119995117	513.15050354	-20.040008545	26.165793355	2025-11-21 12:54:55.218 -080
6	2025-11-19	TSLA	403.989990234	436.996002197	-58.08001709	39.915096966	2025-11-21 12:54:55.218 -080
7	2025-11-18	MSFT	493.790008545	514.821502686	-20.540008545	23.519627806	2025-11-21 12:54:55.218 -080
8	2025-11-18	TSLA	401.25	438.745002747	-43.010009766	34.759740151	2025-11-21 12:54:55.218 -080
9	2025-11-18	AAPL	267.440002441	268.800500488	-2.600006104	44.719609375	2025-11-21 12:54:55.218 -080
10	2025-11-17	TSLA	408.920013428	440.812503052	-59.449981689	36.483598027	2025-11-21 12:54:55.218 -080

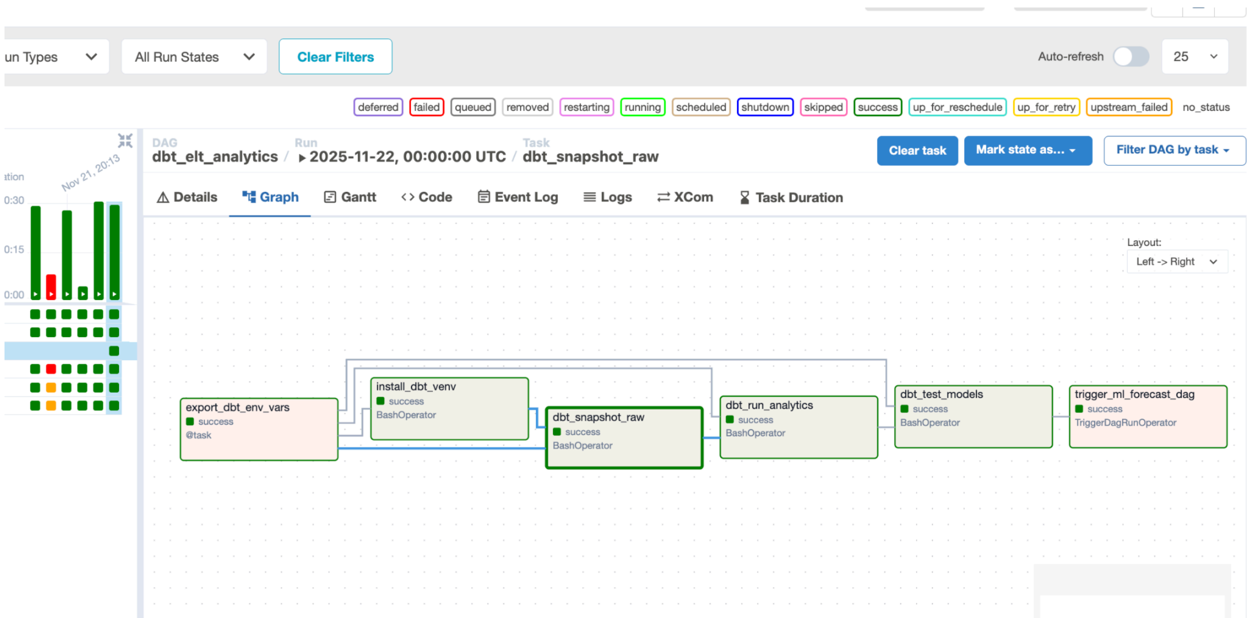
4. Airflow Implementation

4.1 Airflow Data Pipeline Code

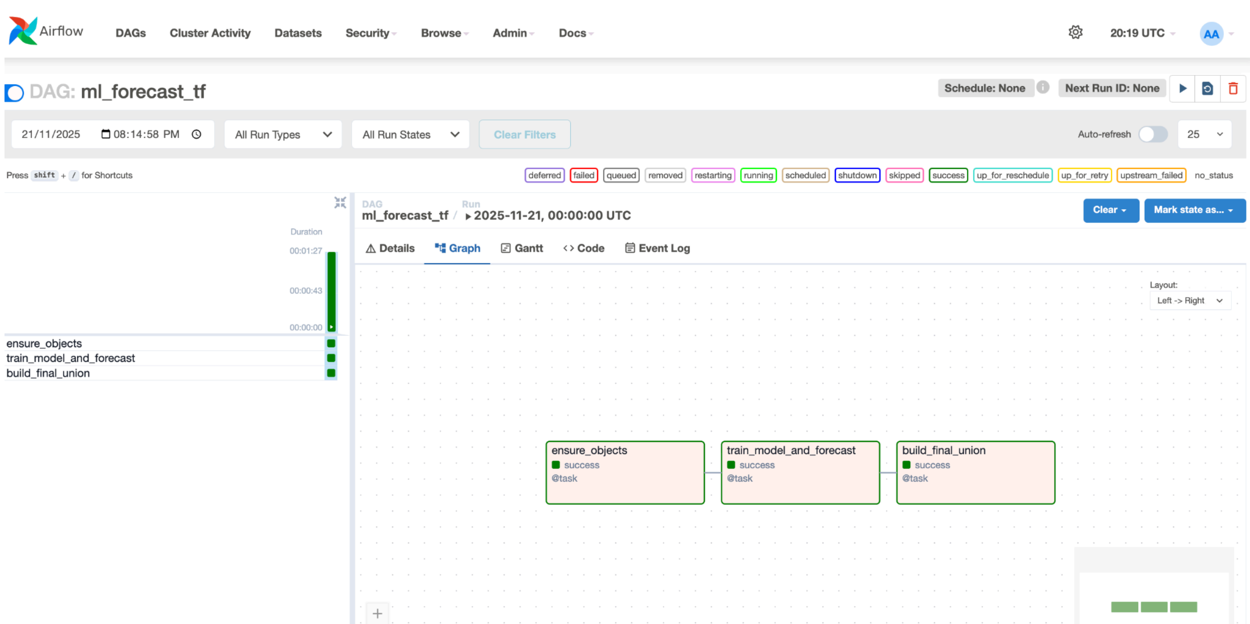
DAG 1: ETL (dags/dag_yfinance_etl.py) This DAG handles data ingestion from YFinance to Snowflake using the SnowflakeHook and MERGE logic for idempotency.



DAG 2: ELT (dags/dag_dbt_elt.py) This DAG securely exports credentials and runs dbt transformations.



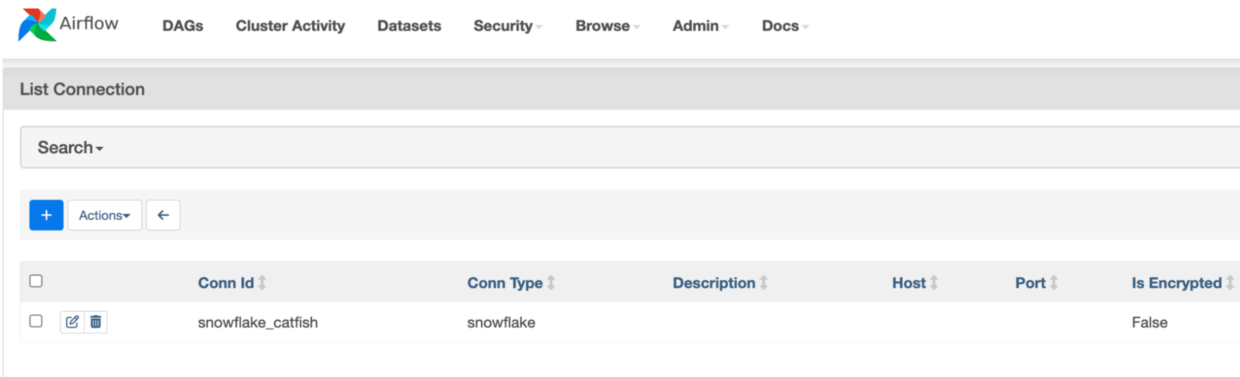
DAG 3: ML Forecast (dags/dag_ml_forecast.py) Code manages schema creation and Snowflake ML forecasting logic using the SnowflakeHook.

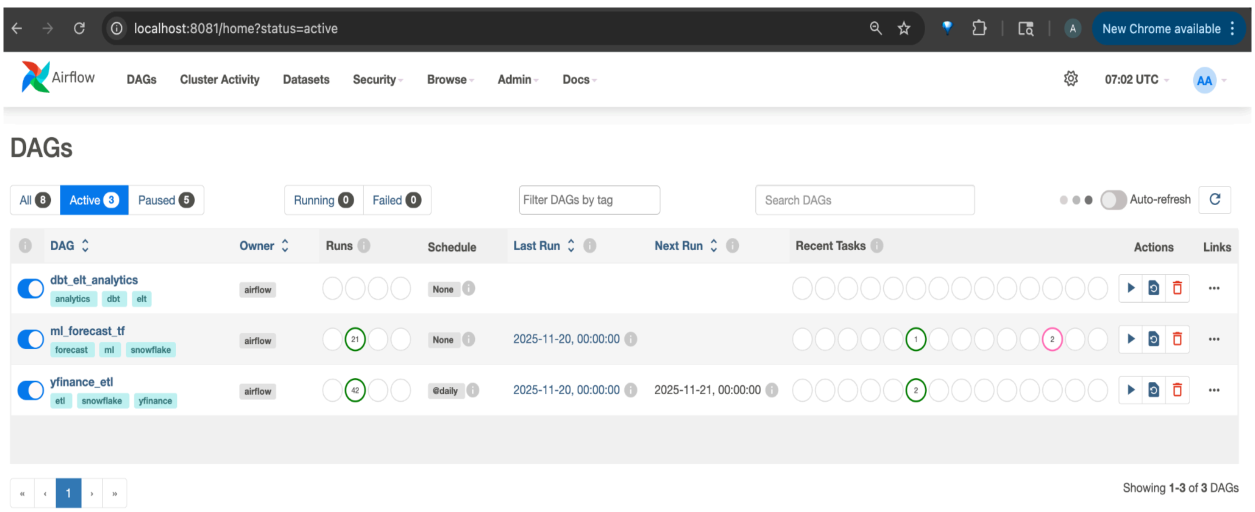


(Code omitted here for report length, refer to attached file https://github.com/abhinitasanabada-web/Lab2/tree/main/dags)

4.2 Airflow Web UI Screenshot

Screenshot of Airflow Grid View here showing yfinance_etl, dbt_elt_analytics, and ml_forecast_tf and connections





4.3 Connection and Variable Usage

- Connections: All Snowflake interactions utilize `SnowflakeHook(snowflake_conn_id="snowflake_catfish")`. Credentials are never hardcoded. In the dbt DAG, these credentials are dynamically extracted and passed as environment variables.

- Variables: Configuration such as `stock_symbols` and `target_schema_raw` are managed via Airflow Variables, allowing parameter changes without code deployment.

4.4 Idempotency and Transactions

Idempotency & Transactions

- Idempotency Strategies:
  - Full Refresh (ML DAG): Implements an atomic Truncate-Load pattern. The target table is cleared via `TRUNCATE` before a bulk `INSERT`, ensuring re-runs never duplicate forecast data.
  - Incremental Load (ETL DAG): Uses a Merge-Upsert strategy. Data is staged in a temporary table, then a Snowflake `MERGE` command updates existing records or inserts new ones, preventing duplicates during daily ingestion.
- Transaction Management:
  - All SQL operations utilize `SnowflakeHook` with `autocommit(False)` to ensure atomicity.
  - Logic is wrapped in `try/except` blocks that execute `conn.rollback()` upon failure and explicitly `raise` the error, ensuring Airflow correctly halts the pipeline without committing partial data.

5. dbt Implementation

5.1 dbt Project Code

Model: models/analytics/stock_analytics.sql

SQL

```
{{ config(materialized='table', schema='ANALYTICS', unique_key=['symbol', 'trade_date']) }}


WITH base AS (

  SELECT trade_date, symbol, close,

    close - LAG(close, 1) OVER (PARTITION BY symbol ORDER BY trade_date) AS price_change

  FROM {{ source('raw', 'stock_prices') }}
```

```
),

gains_losses AS (

  SELECT *,

    CASE WHEN price_change > 0 THEN price_change ELSE 0 END AS gain,

    CASE WHEN price_change < 0 THEN ABS(price_change) ELSE 0 END AS loss

  FROM base

),

indicators AS (

  SELECT t1.*,

    AVG(t1.close) OVER (PARTITION BY t1.symbol ORDER BY t1.trade_date ROWS BETWEEN 19 PRECEDING AND CURRENT ROW) AS ma_20,

    AVG(t2.gain) OVER (PARTITION BY t1.symbol ORDER BY t1.trade_date ROWS BETWEEN 13 PRECEDING AND CURRENT ROW) AS avg_gain,

    AVG(t2.loss) OVER (PARTITION BY t1.symbol ORDER BY t1.trade_date ROWS BETWEEN 13 PRECEDING AND CURRENT ROW) AS avg_loss

  FROM base t1

  INNER JOIN gains_losses t2 ON t1.symbol = t2.symbol AND t1.trade_date = t2.trade_date

)

SELECT trade_date, symbol, close, ma_20,

  CASE WHEN avg_loss = 0 THEN 100.0 ELSE 100.0 - (100.0 / (1.0 + (avg_gain / avg_loss))) END AS rsi_14

FROM indicators

ORDER BY symbol, trade_date
```

Tests: models/schema.yml

YAML

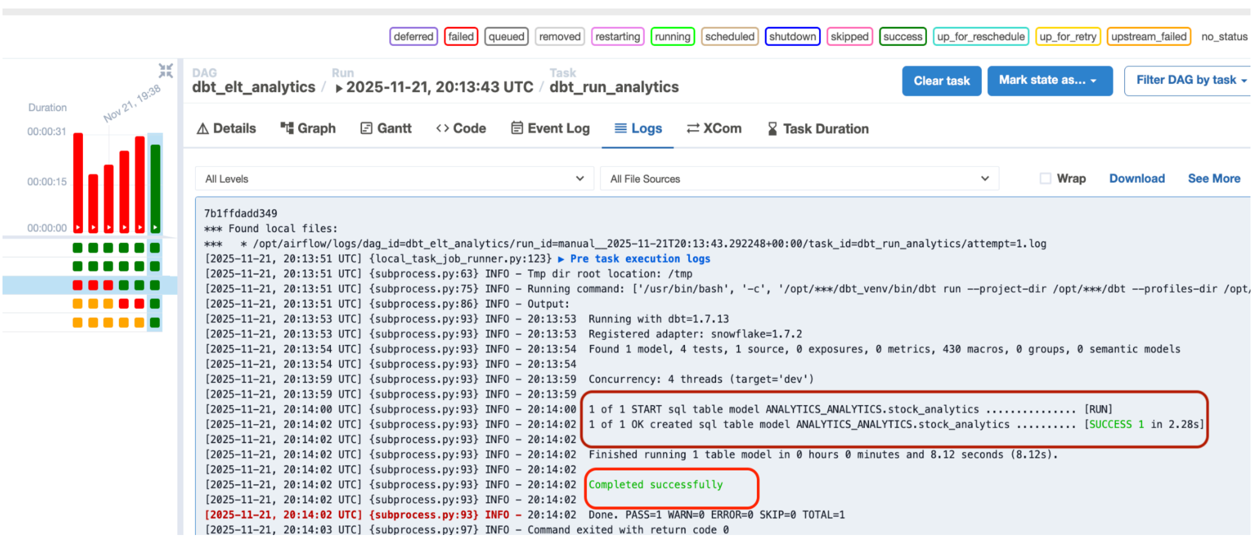
```
version: 2

models:
```

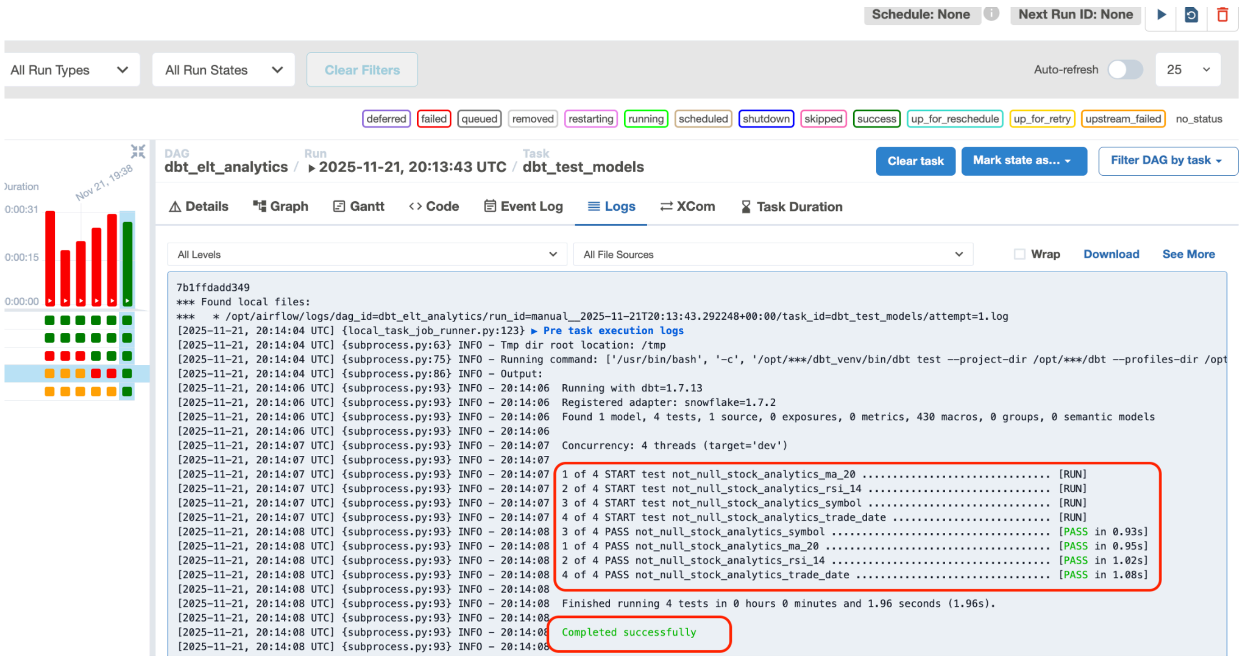
```yaml
- name: stock_analytics
  columns:
    - name: trade_date
      tests: [not_null, unique: {group_by: [symbol]}]
    - name: rsi_14
      tests:
        - not_null
        - check: {expression: "rsi_14 >= 0 AND rsi_14 <= 100"}
```

5.2 dbt Command Evidence

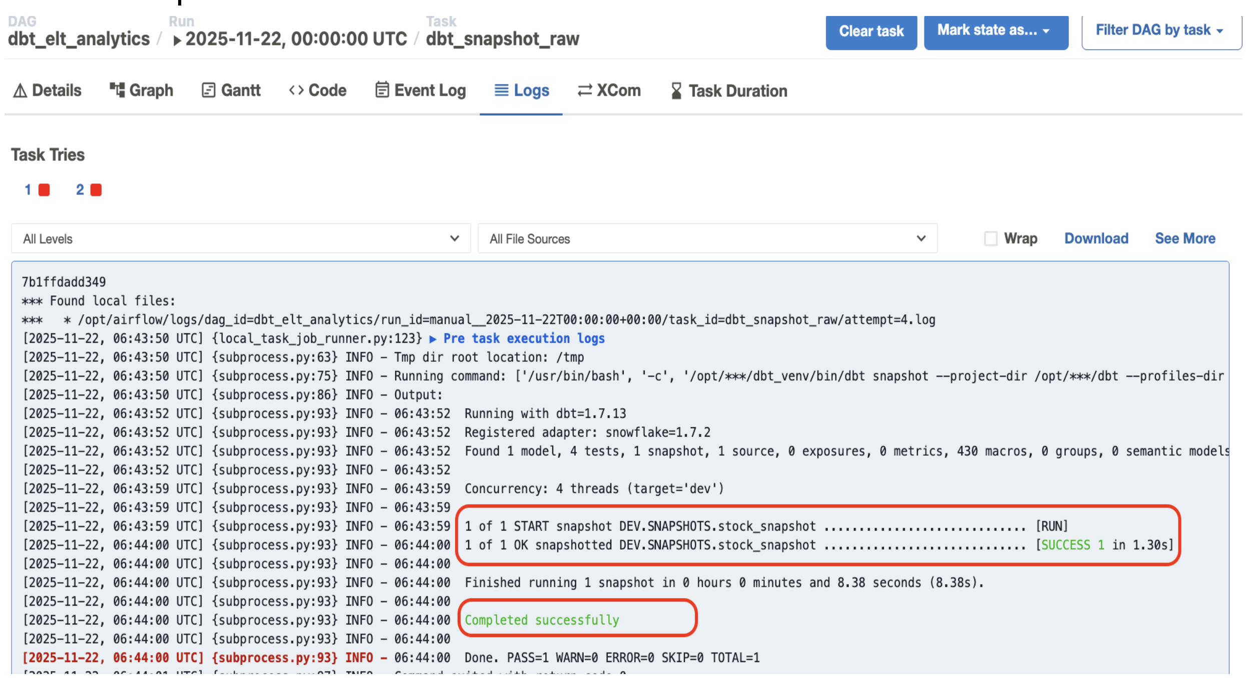
Screenshot of terminal showing successful `dbt run, dbt snapshots` *and* `dbt test` *execution*



Test Cases for dbt test:



For dbt snapshots:



From Docker terminal:

```
airflow-1      | 151.101.192.223 - - [21/Nov/2025:20:24:29 +0000] "GET /api/v1/dags/dbt_elt_analytics/dagRuns/manual__2025-11-21T20:13:43.292248+00:00/taskInstances/dbt_test_models HTTP/1.1" 200 1282 "http://localhost:8081/dags/dbt_elt_analytics/grid?dag_run_id=manual__2025-11-21T20%3A13%3A43.292248%2B00%3A00&tab=logs&task_id=dbt_test_models" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36"
airflow-1      | 151.101.192.223 - - [21/Nov/2025:20:24:29 +0000] "GET /api/v1/dags/dbt_elt_analytics/dagRuns/manual__2025-11-21T20:13:43.292248+00:00/taskInstances/dbt_test_models/logs/1?full_content=false HTTP/1.1" 200 5945 "http://localhost:8081/dags/dbt_elt_analytics/grid?dag_run_id=manual__2025-11-21T20%3A13%3A43.292248%2B00%3A00&tab=logs&task_id=dbt_test_models" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36"
airflow-1      | 127.0.0.1 - - [21/Nov/2025:20:24:58 +0000] "GET /health HTTP/1.1" 200 283 "-" "curl/7.88.1"
airflow-1      | 127.0.0.1 - - [21/Nov/2025:20:25:28 +0000] "GET /health HTTP/1.1" 200 283 "-" "curl/7.88.1"
airflow-1      | 127.0.0.1 - - [21/Nov/2025:20:25:58 +0000] "GET /health HTTP/1.1" 200 283 "-" "curl/7.88.1"
airflow-1      | 127.0.0.1 - - [21/Nov/2025:20:26:29 +0000] "GET /health HTTP/1.1" 200 283 "-" "curl/7.88.1"
airflow-1      | 127.0.0.1 - - [21/Nov/2025:20:26:59 +0000] "GET /health HTTP/1.1" 200 283 "-" "curl/7.88.1"
airflow-1      | [2025-11-21T20:27:03.686+0000] {scheduler_job_runner.py:1846} INFO - Adopting or resetting orphaned tasks for active dag runs
```

5.3 Scheduling Strategy

The dbt DAG is scheduled with `schedule=None` and is triggered strictly by the `TriggerDagRunOperator` located at the end of the ETL DAG. This ensures dbt only attempts to transform data after the raw data has been successfully updated.

6. BI Tool Visualization

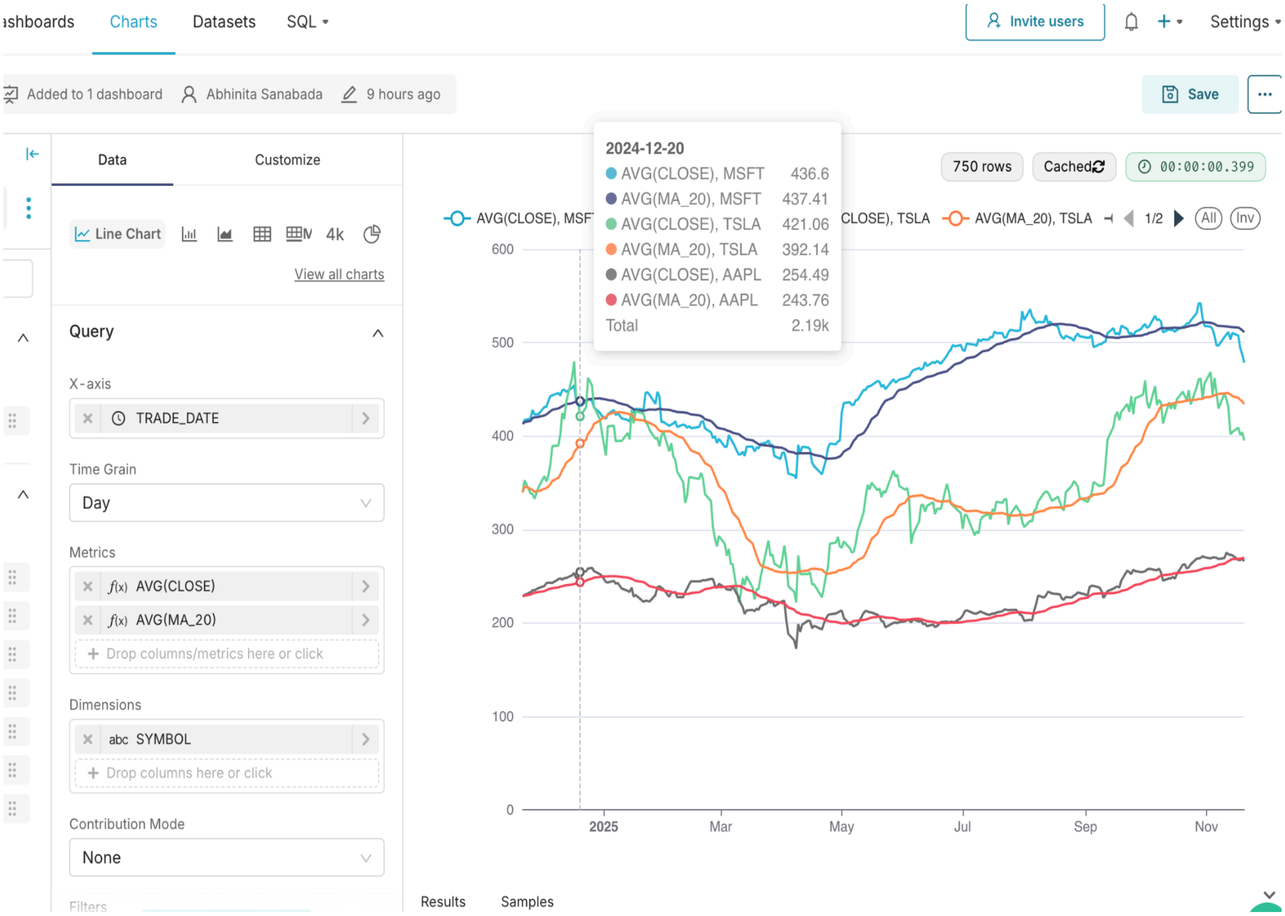
Tool Used: Apache Superset / Preset

6.1 Dashboard Description

- Purpose: To provide financial analysts with real-time technical indicators for buy/sell decision-making.
- Dataset: `ANALYTICS.STOCK_ANALYTICS`
- Key Metrics:
  - MA_20 (Moving Average): Used to identify trend direction.
  - RSI_14 (Relative Strength Index): Used to identify overbought (>70) or oversold (<30) conditions.

6.2 Dashboard Screenshots

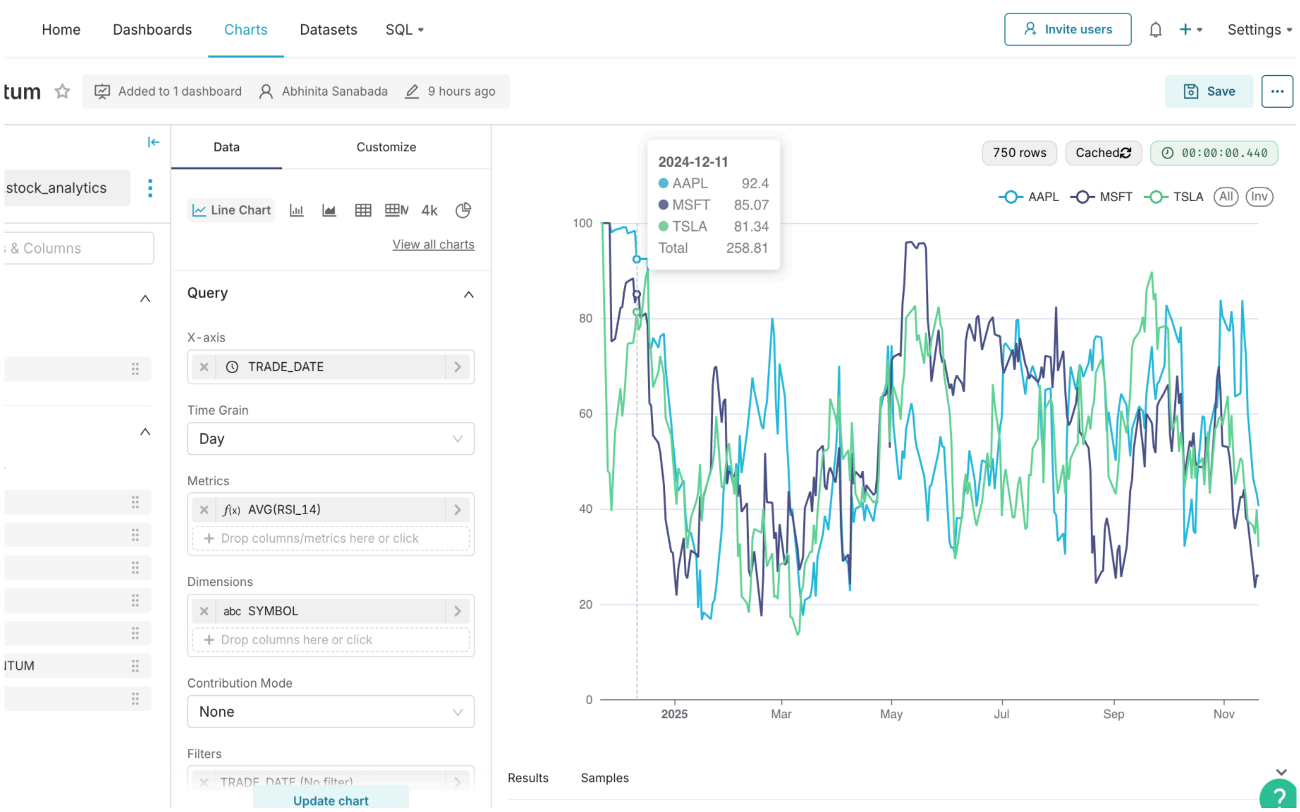
Screenshot 1: Filtered Dashboard View limiting analysis to Q1 2024, demonstrating dynamic interactivity.



Dashboard URL:

https://b774b44d.us2a.app.preset.io/superset/dashboard/8/?native_filters_key=O3jBF_DT2fwhIQQN6BwB5xBXtYjnf8VpTHEI8yys5lAcumLefR9NQk-aqKXrfdIu

Screenshot 2: Comprehensive Stock Analytics Dashboard showing RSI and Moving Averages.



7. Conclusion

This lab successfully implemented a modular, scalable data pipeline. By decoupling Ingestion (Airflow/Python), Transformation (dbt), and Consumption (BI/ML), the system achieves high maintainability. The use of `SnowflakeHook` and `MERGE` logic ensured security and data integrity, meeting all functional requirements.

8. References

1. dbt Documentation. (n.d.). Retrieved from https://docs.getdbt.com/
2. Apache Airflow Providers: Snowflake. (n.d.). Retrieved from https://airflow.apache.org/
3. Keeyong Han. (2025). Week 10 ELT Deepdive (dbt). [Course Lecture Notes]. San Jose State University.

9. GitHub repo:

https://github.com/abhinitasanabada-web/Lab2