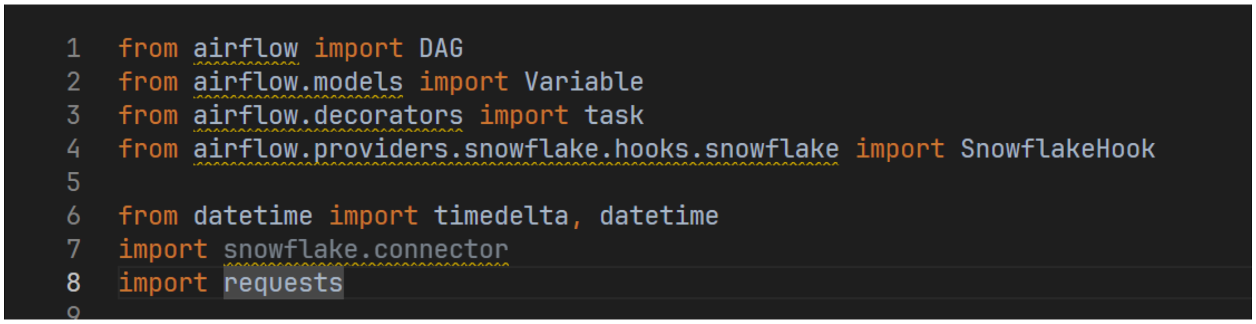
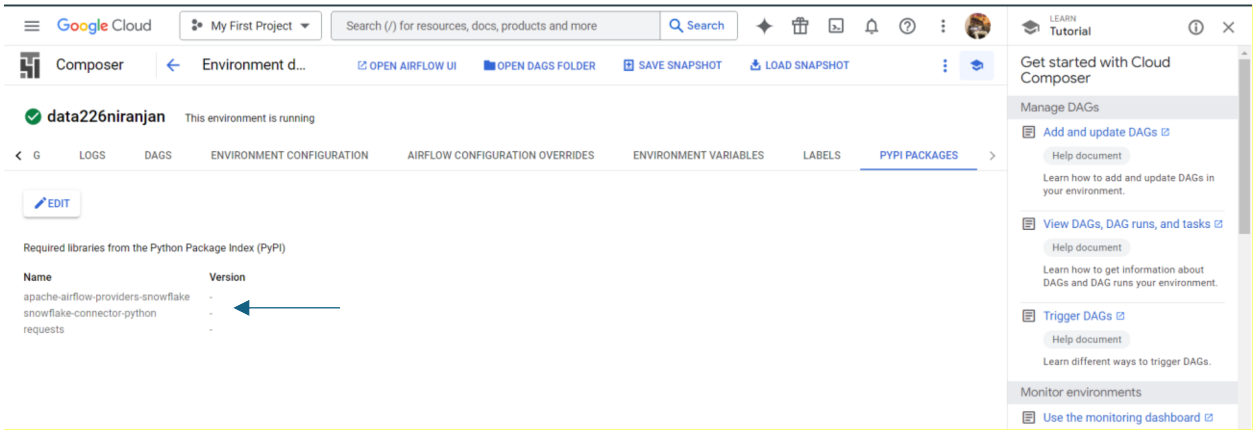
Homework 5

- (+1) Import all the required python modules



```python
from airflow import DAG
from airflow.models import Variable
from airflow.decorators import task
from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook

from datetime import timedelta, datetime
import snowflake.connector
import requests
```

- (+1) Ensure that any missing package(s) are added to the PYPI packages
    - Capture the screenshot (an example will be provided ①)



data226niranjan This environment is running

Required libraries from the Python Package Index (PyPI)

Name	Version
apache-airflow-providers-snowflake	-
snowflake-connector-python	-
requests	-

Get started with Cloud Composer

Manage DAGs

Add and update DAGs

Learn how to add and update DAGs in your environment.

View DAGs, DAG runs, and tasks

Learn how to get information about DAGs and DAG runs your environment.

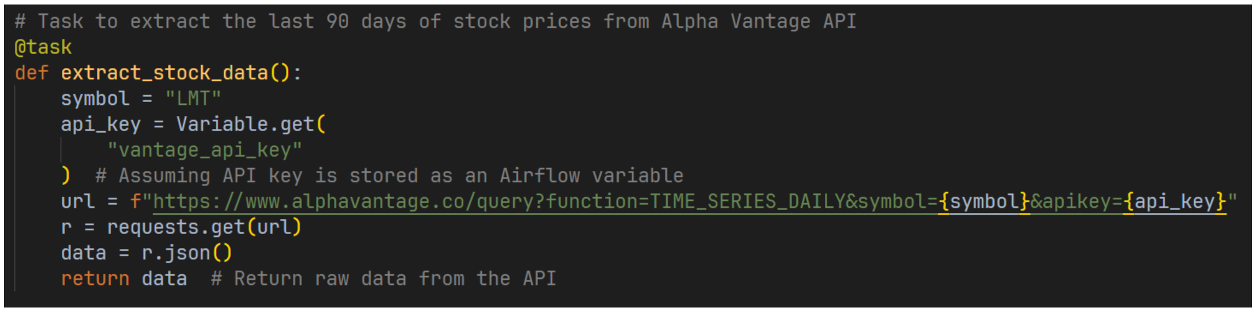
Trigger DAGs

Learn different ways to trigger DAGs.

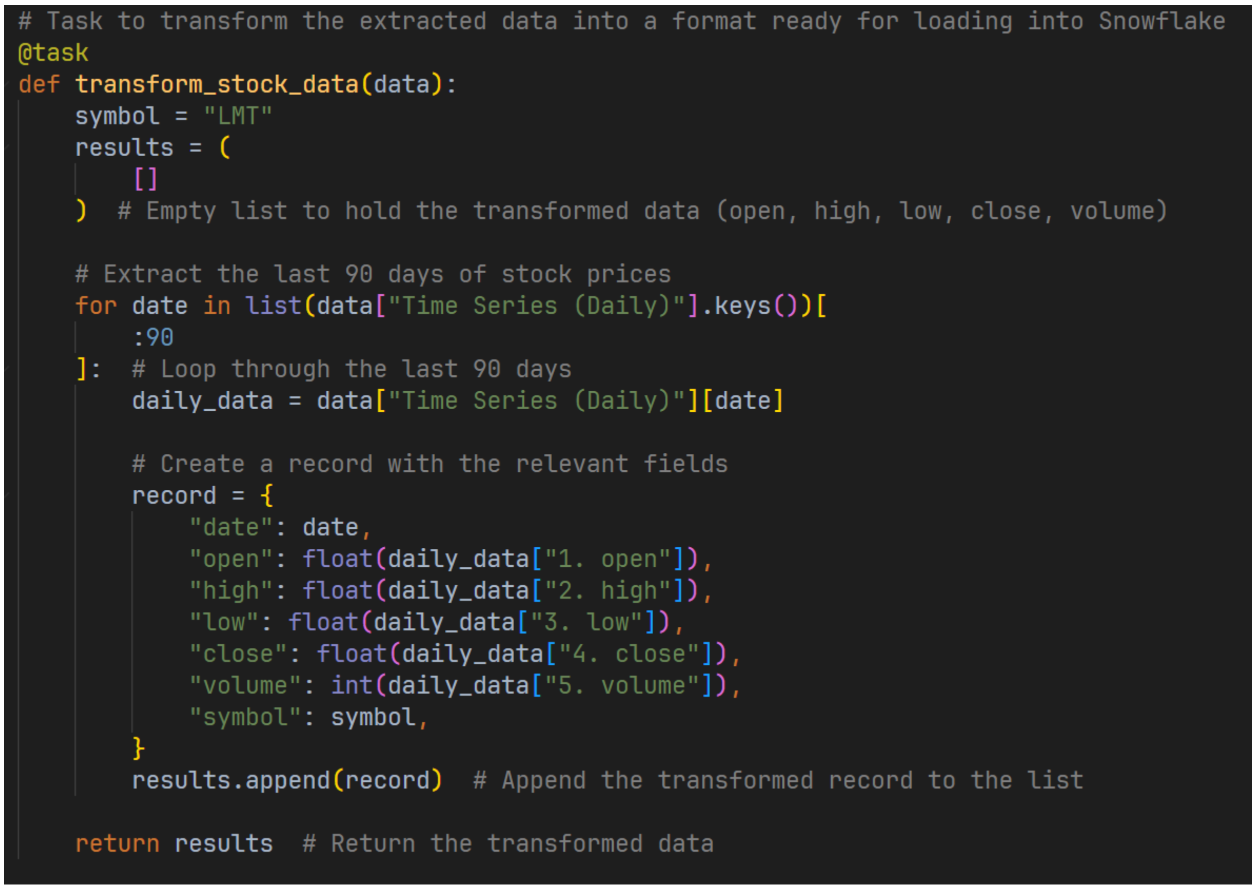
Monitor environments

Use the monitoring dashboard

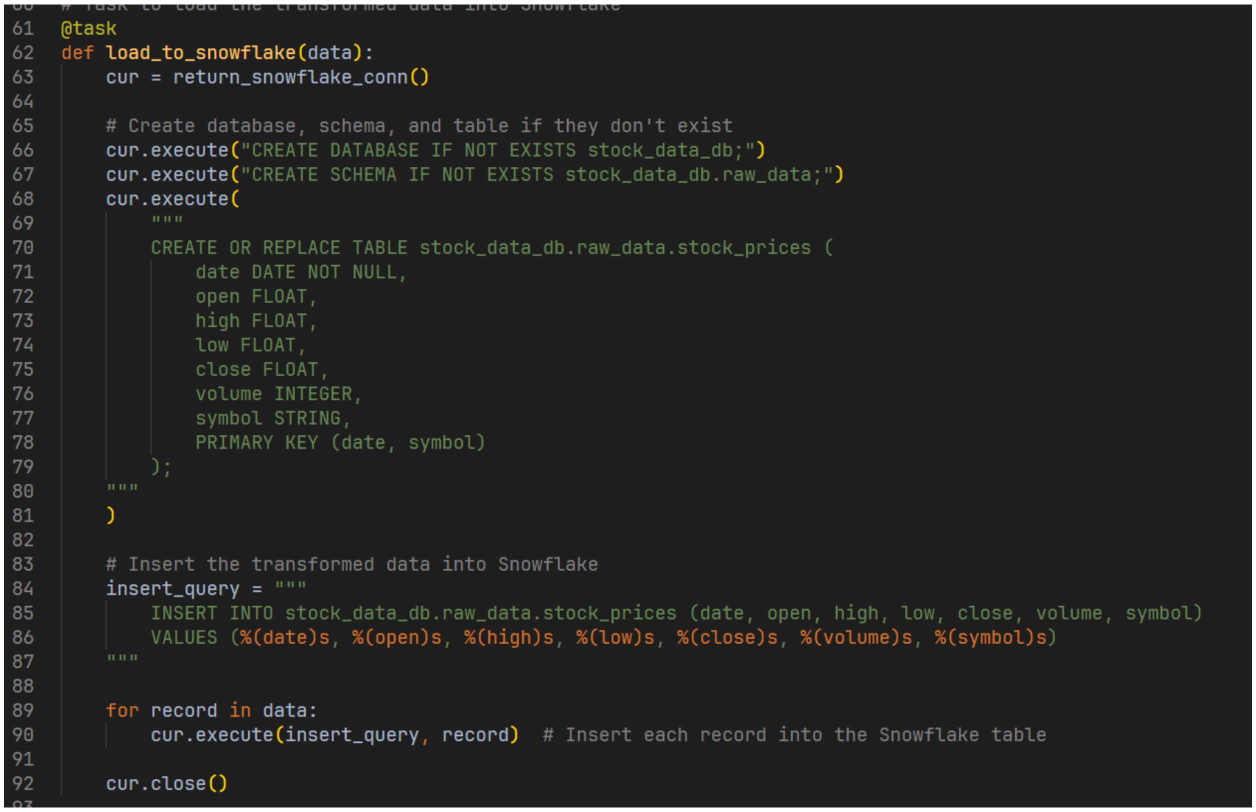
- (+3) Create tasks using @task decorator



```python
# Task to extract the last 90 days of stock prices from Alpha Vantage API
@task
def extract_stock_data():
    symbol = "LMT"
    api_key = Variable.get(
        "vantage_api_key"
    )  # Assuming API key is stored as an Airflow variable
    url = f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={api_key}"
    r = requests.get(url)
    data = r.json()
    return data  # Return raw data from the API
```

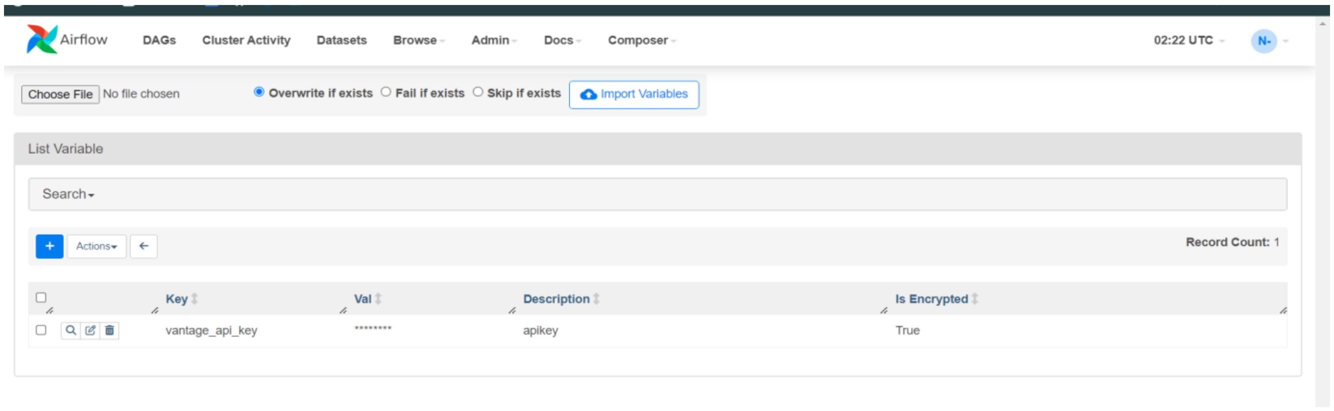


```python
# Task to transform the extracted data into a format ready for loading into Snowflake
@task
def transform_stock_data(data):
    symbol = "LMT"
    results = (
        []
    )  # Empty list to hold the transformed data (open, high, low, close, volume)

    # Extract the last 90 days of stock prices
    for date in list(data["Time Series (Daily)"].keys())[
        :90
    ]:  # Loop through the last 90 days
        daily_data = data["Time Series (Daily)"][date]

        # Create a record with the relevant fields
        record = {
            "date": date,
            "open": float(daily_data["1. open"]),
            "high": float(daily_data["2. high"]),
            "low": float(daily_data["3. low"]),
            "close": float(daily_data["4. close"]),
            "volume": int(daily_data["5. volume"]),
            "symbol": symbol,
        }
        results.append(record)  # Append the transformed record to the list

    return results  # Return the transformed data
```



```python
# Task to load the transformed data into Snowflake
@task
def load_to_snowflake(data):
    cur = return_snowflake_conn()

    # Create database, schema, and table if they don't exist
    cur.execute("CREATE DATABASE IF NOT EXISTS stock_data_db;")
    cur.execute("CREATE SCHEMA IF NOT EXISTS stock_data_db.raw_data;")
    cur.execute(
        """
        CREATE OR REPLACE TABLE stock_data_db.raw_data.stock_prices (
            date DATE NOT NULL,
            open FLOAT,
            high FLOAT,
            low FLOAT,
            close FLOAT,
            volume INTEGER,
            symbol STRING,
            PRIMARY KEY (date, symbol)
        );
    """
    )

    # Insert the transformed data into Snowflake
    insert_query = """
        INSERT INTO stock_data_db.raw_data.stock_prices (date, open, high, low, close, volume, symbol)
        VALUES (%(date)s, %(open)s, %(high)s, %(low)s, %(close)s, %(volume)s, %(symbol)s)
    """

    for record in data:
        cur.execute(insert_query, record)  # Insert each record into the Snowflake table

    cur.close()
```

- (+1) Set up a variable for Alpha Vantage API key
    - Use the variable in your code (Variable.get)
    - Capture the Admin -> Variables screenshot (an example will be provided ②)



No file chosen

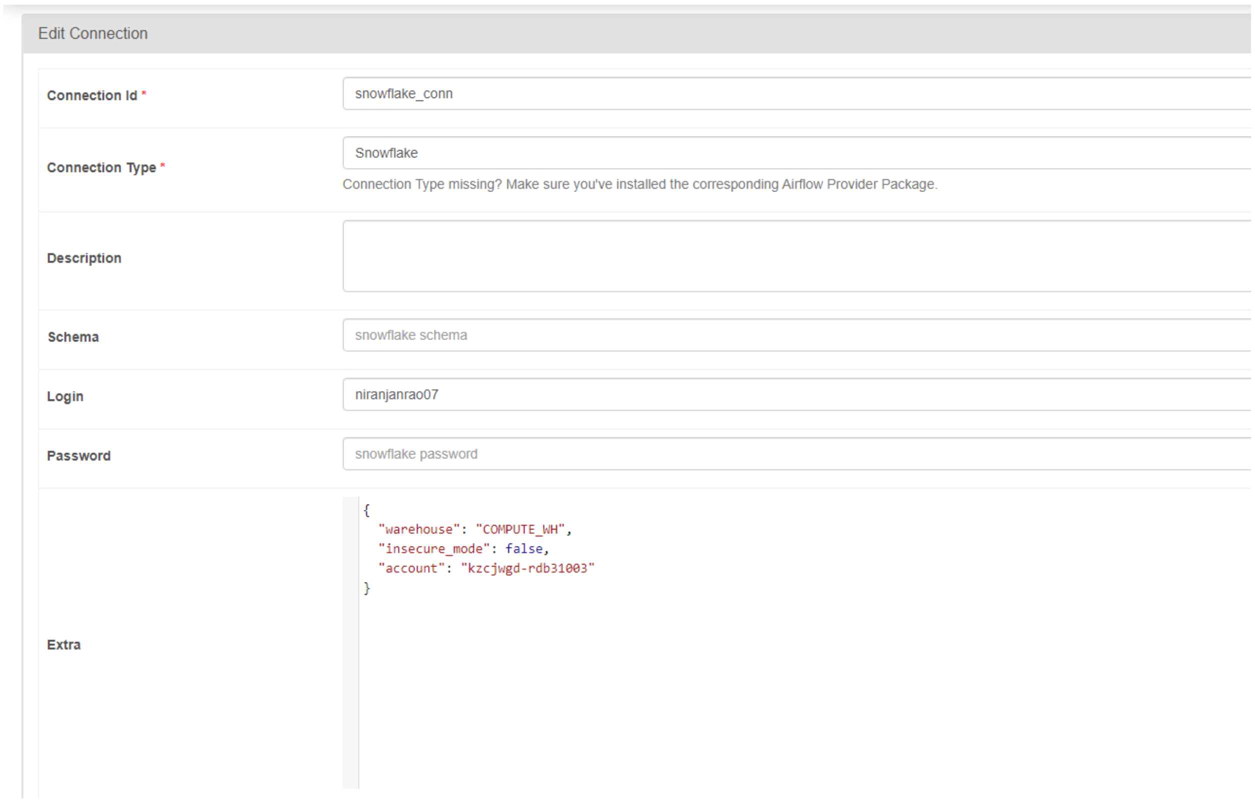
Overwrite if exists [✓] Fail if exists Skip if exists

List Variable

Record Count: 1

Key	Val	Description	Is Encrypted
vantage_api_key	********	apikey	True

- (+2) Set up Snowflake Connection (refer to GitHub linkLinks to an external site.)
    - Use the connection in your code
    - Capture the Connection detail page screenshot (an example will be provided ③)

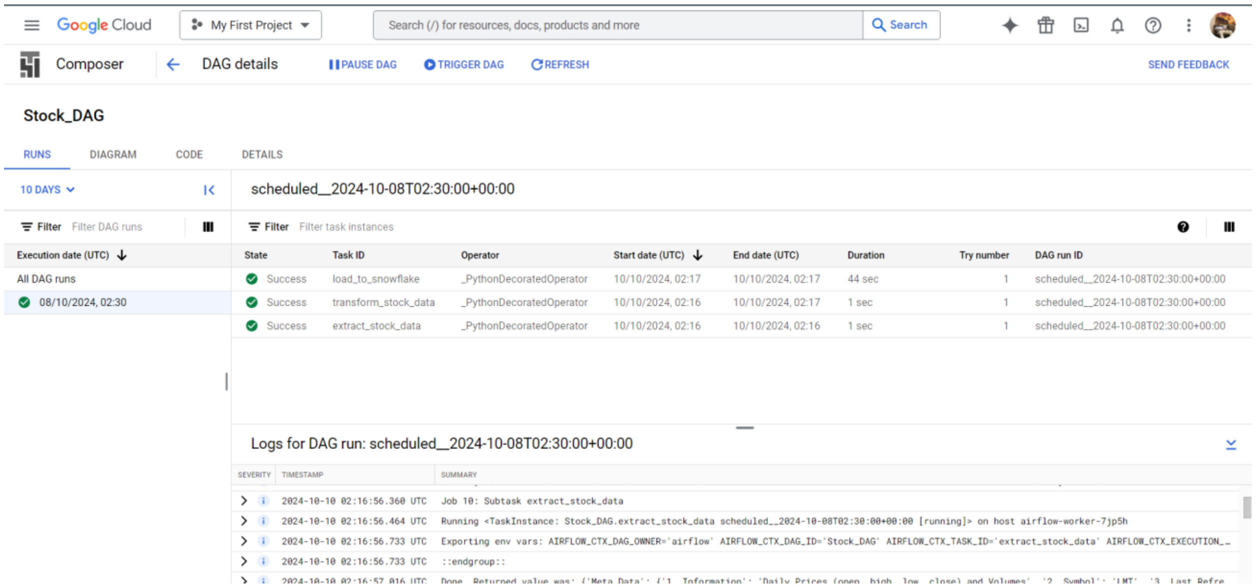


Edit Connection

Connection Id *	snowflake_conn
Connection Type *	Snowflake Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Schema	snowflake schema
Login	niranjanrao07
Password	snowflake password
Extra	

```
{
  "warehouse": "COMPUTE_WH",
  "insecure_mode": false,
  "account": "kzcjwgd-rdb31003"
}
```

- (+4) Ensure the overall DAG runs successfully
    - A github link with the entire code needs to be submitted



Stock_DAG

Execution date (UTC)
All DAG runs
08/10/2024, 02:30

scheduled__2024-10-08T02:30:00+00:00

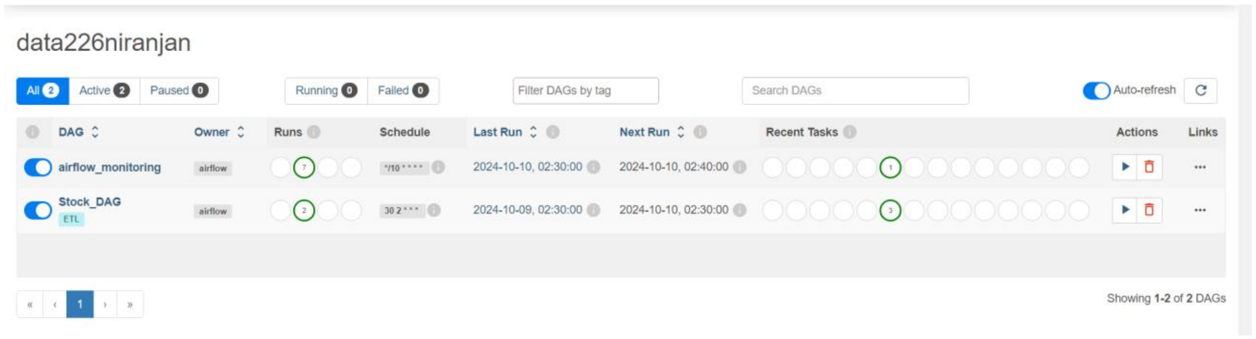
State	Task ID	Operator	Start date (UTC)	End date (UTC)	Duration	Try number	DAG run ID
Success	load_to_snowflake	_PythonDecoratedOperator	10/10/2024, 02:17	10/10/2024, 02:17	44 sec	1	scheduled__2024-10-08T02:30:00+00:00
Success	transform_stock_data	_PythonDecoratedOperator	10/10/2024, 02:16	10/10/2024, 02:17	1 sec	1	scheduled__2024-10-08T02:30:00+00:00
Success	extract_stock_data	_PythonDecoratedOperator	10/10/2024, 02:16	10/10/2024, 02:16	1 sec	1	scheduled__2024-10-08T02:30:00+00:00

Logs for DAG run: scheduled__2024-10-08T02:30:00+00:00

```
SEVERITY  TIMESTAMP                    SUMMARY
          2024-10-10 02:16:56.360 UTC  Job 10: Subtask extract_stock_data
          2024-10-10 02:16:56.464 UTC  Running <TaskInstance: Stock_DAG.extract_stock_data scheduled__2024-10-08T02:30:00+00:00 [running]> on host airflow-worker-7jp5h
          2024-10-10 02:16:56.733 UTC  Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='Stock_DAG' AIRFLOW_CTX_TASK_ID='extract_stock_data' AIRFLOW_CTX_EXECUTION_…
          2024-10-10 02:16:56.733 UTC  ::endgroup::
          2024-10-10 02:16:57.016 UTC  Done. Returned value was: {'Meta Data': {'1. Information': 'Daily Prices (open, high, low, close) and Volumes', '2. Symbol': 'LMT', '3. Last Refre
```

https://github.com/NiranjanRao07/airflow-DAG

- (+2) Capture two screenshot of your Airflow Web UI (examples to follow)
    - One with the Airlow homepage showing the DAG (④)

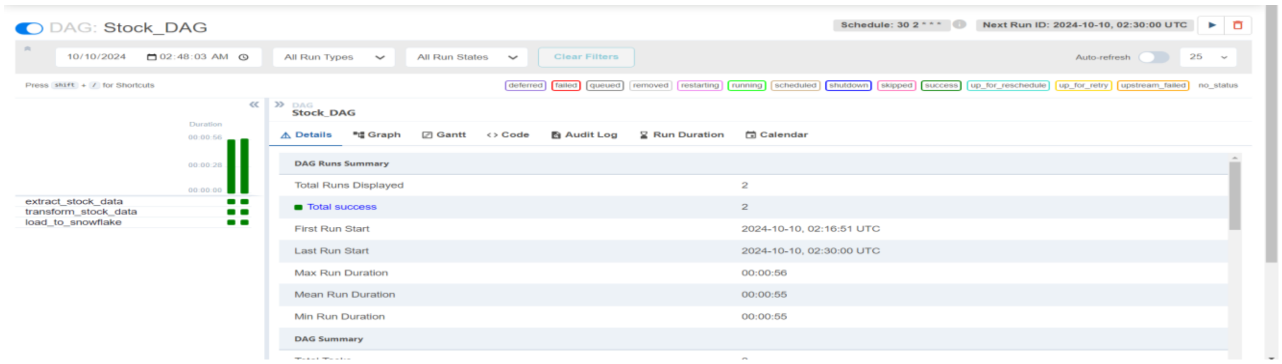


data226niranjan

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
airflow_monitoring	airflow	7	*/10 * * * *	2024-10-10, 02:30:00	2024-10-10, 02:40:00	1		
Stock_DAG ETL	airflow	2	30 2 * * *	2024-10-09, 02:30:00	2024-10-10, 02:30:00	3		

Showing **1-2** of **2** DAGs

- The other with the log screen of the DAG (⑤)



DAG: Stock_DAG

Schedule: 30 2 * * *

Next Run ID: 2024-10-10, 02:30:00 UTC

10/10/2024 02:48:03 AM

DAG Stock_DAG

DAG Runs Summary	
Total Runs Displayed	2
Total success	2
First Run Start	2024-10-10, 02:16:51 UTC
Last Run Start	2024-10-10, 02:30:00 UTC
Max Run Duration	00:00:56
Mean Run Duration	00:00:55
Min Run Duration	00:00:55
DAG Summary	