

## Assignment 5

USERNAME: PIRANHA

Porting homework #4 to Airflow (13 pts)

- (+2) Create tasks using @task decorator (refer to [GitHub linkLinks to an external site.](#))
  - You can use as many tasks as you want

```
import requests
from airflow import DAG
from airflow.models import Variable
from airflow.decorators import task
from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook
from airflow.hooks.base import BaseHook
from datetime import timedelta
from datetime import datetime
import snowflake.connector
```

```
@task
def extract(symbol):
    api_key=Variable.get("vantage_api_key")
    symbol="AAPL"
    url =
f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={
symbol}&apikey={api_key}'
    r = requests.get(url)
    data = r.json()
    results = []
    for d in data["Time Series (Daily)"]:
        stock_details = data["Time Series (Daily)"][d].copy()
        stock_details["date"] = d
        results.append(stock_details)
    return results

@task
def transform(price_list):
    price_list = sorted(price_list, key=lambda d:
d['date'],reverse=True)[:90]
    records = []
    for p in price_list:
        record = {
            "date": p["date"],
            "open": float(p["1. open"]),
```

```

        "high": float(p["2. high"]),
        "low": float(p["3. low"]),
        "close": float(p["4. close"]),
        "volume": float(p["5. volume"])}
    records.append(record)
return records

```

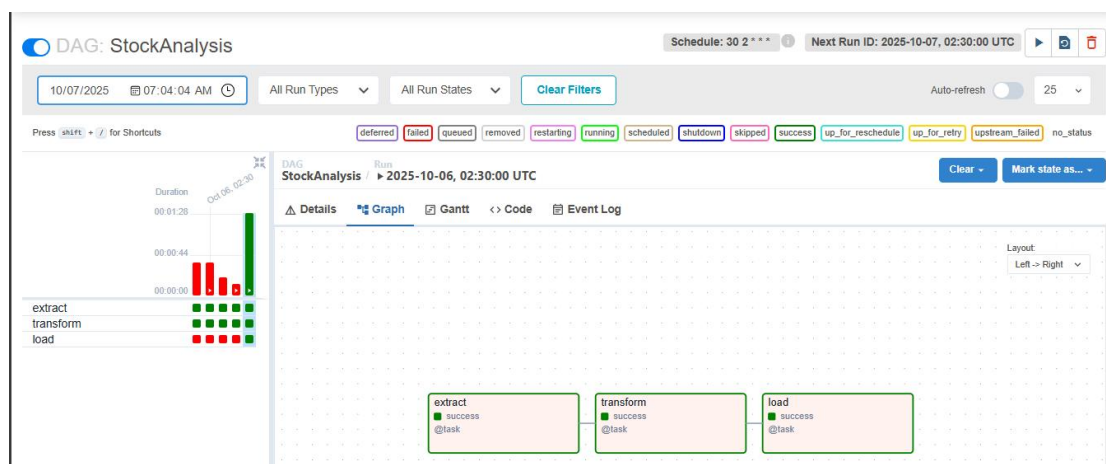
- Schedule the tasks properly (task dependency)

```

with DAG(
    dag_id="StockAnalysis",
    start_date=datetime(2025,9,29),
    catchup=False,
    tags=["ETL"],
    schedule='30 2 * * *'
)as dag:
    #pass
    #symbol = "AAPL"
    target_table="raw.stock_datas"
    symbol = "AAPL"
    vantage_api_key=Variable.get("vantage_api_key")
    price_list = extract("AAPL")
    records = transform(price_list)
    load(records=records, symbol= "AAPL")

    extract_task >> transform_task >> load_task

```



- (+1) Set up a variable for Alpha Vantage API key
  - Use the variable in your code (Variable.get)

```

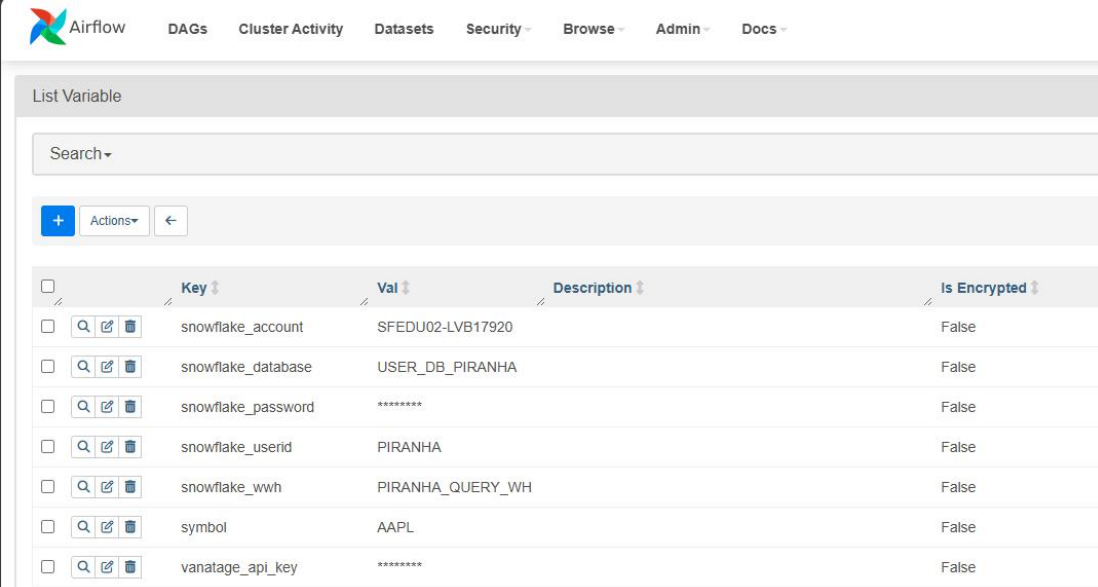
Variable.get = staticmethod(lambda key: "your_secret_key" if key ==
    "vantage_api_key" else None)

```

```
vantage_api_key = Variable.get("vantage_api_key")
```

```
@task
def extract(symbol):
    api_key=Variable.get("vantage_api_key")
    symbol="AAPL"
    url =
f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={
symbol}&apikey={api_key}'
    r = requests.get(url)
    data = r.json()
    results = []
    for d in data["Time Series (Daily)"]:
        stock_details = data["Time Series (Daily)"][d].copy()
        stock_details["date"] = d
        results.append(stock_details)
    return results
```

Capture the Admin -> Variables screenshot (an example will be provided ①)



The screenshot shows the Airflow Admin interface. The top navigation bar includes links for DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The main content area is titled 'List Variable' and features a search bar and a table of variables. The table has columns for Key, Val, Description, and Is Encrypted. The variables listed are:

| Key                | Val              | Description | Is Encrypted |
|--------------------|------------------|-------------|--------------|
| snowflake_account  | SFEDU02-LVB17920 |             | False        |
| snowflake_database | USER_DB_PIRANHA  |             | False        |
| snowflake_password | *****            |             | False        |
| snowflake_userid   | PIRANHA          |             | False        |
| snowflake_wwh      | PIRANHA_QUERY_WH |             | False        |
| symbol             | AAPL             |             | False        |
| vantage_api_key    | *****            |             | False        |

(+2) Set up Snowflake Connection (refer to [GitHub link](#)Links to an external site.)

- Use the connection in your code

```
def return_snowflake_conn():
    hook= SnowflakeHook(snowflake_conn_id="snowflake_conn")
    conn= hook.get_conn()
    return conn
```

- Capture the Connection detail page screenshot (an example will be provided ②)

The screenshot shows the Airflow web interface. At the top, there's a navigation bar with links to Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. Below this, the 'List Connection' page is displayed. It features a search bar, a table with connection details, and a 'Record Count: 1' indicator. The table has columns for Conn Id, Conn Type, Description, Host, Port, Is Encrypted, and Is Extra Encrypted. The single entry is 'snowflake\_conn' with type 'snowflake'. Above the table, a form shows the details for this connection, including a description, schema, login, password, and a JSON configuration for the Snowflake hook.

```
{
  "account": "SFEDU02-LVB17920",
  "warehouse": "PIRANHA_QUERY_MH",
  "database": "USER_DB_PIRANHA",
  "insecure_mode": false
}
```

At the bottom of the screenshot, a Windows taskbar is visible with the date 10/6/20 and time 11:47 PM.

- (+5) Ensure the overall DAG is implemented properly and runs successfully
  - A github link with the entire code needs to be submitted (2 pts)

https://github.com/elsasjsu/Stock-Prediction-Analyzer/blob/main/StockAnalysis\_Assignment\_5.py

elsasjsu / Stock-Prediction-Analyzer

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main

Go to file

DAGS

README.md

STOCK\_PRICE\_DAILY.py

StockAnalysis\_Assignment\_5.py

docker-compose.yaml

lab1\_STOCK\_ETL.py

Stock-Prediction-Analyzer / StockAnalysis\_Assignment\_5.py

elsasjsu Add files via upload b7f8b1d · 16 minutes ago

Code Blame 123 lines (113 loc) · 3.56 KB

```
1 import requests
2 from airflow import DAG
3 from airflow.models import Variable
4 from airflow.decorators import task
5 from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook
6 from airflow.hooks.base import BaseHook
7 from datetime import timedelta
8 from datetime import datetime
9 import snowflake.connector
10
11 Variable.get = staticmethod(lambda key: "your_secret_key" if key == "vantage_api_key"
12
13 vantage_api_key = Variable.get("vantage_api_key")
14
15 def return_snowflake_conn():
16     hook= SnowflakeHook(snowflake_conn_id="snowflake_conn")
17     conn= hook.get_conn()
```

Symbols

Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.

Filter symbols

const vantage\_api\_key

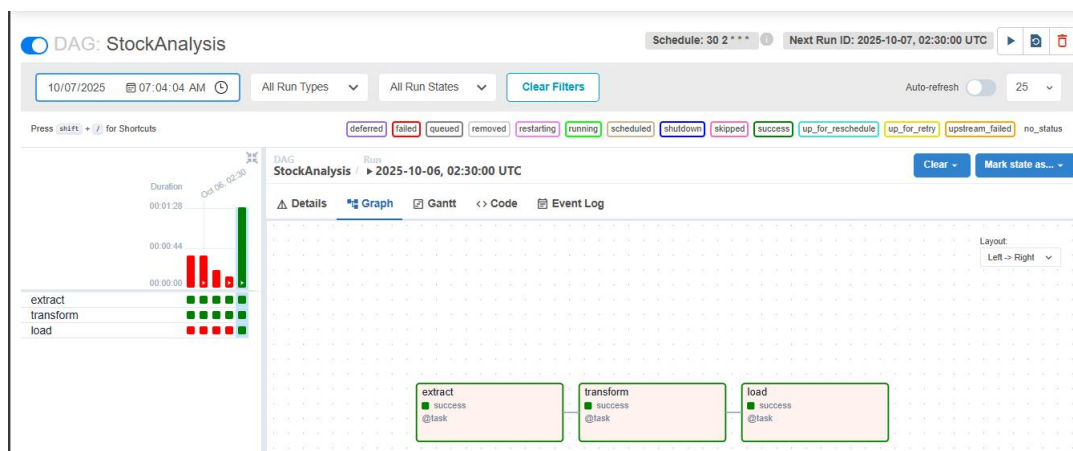
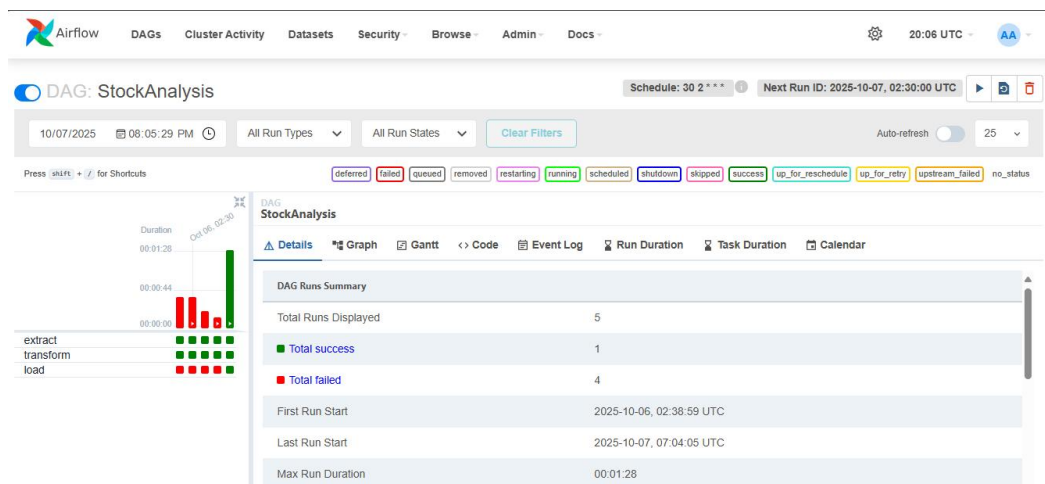
func return\_snowflake\_conn

func extract

func transform

func load

Git Hub Link: [Stock-Prediction-Analyzer/StockAnalysis\\_Assignment\\_5.py at main · elsasjsu/Stock-Prediction-Analyzer](https://github.com/elsasjsu/Stock-Prediction-Analyzer/blob/main/StockAnalysis_Assignment_5.py)



- Implement the same full refresh using SQL transaction (3 pts)

```
@task
def load(records, symbol):
    target = "RAW.STOCK_DATAS"
    conn = return_snowflake_conn()
    cur = conn.cursor()
    try:
        cur.execute("BEGIN")
        extras =
(BaseHook.get_connection("snowflake_conn").extra_dejson or {})
        wh = extras.get("warehouse")
        db = extras.get("database")
        if wh:
            cur.execute(f"USE WAREHOUSE {wh}")
        db = extras.get("database")
        if db:
            cur.execute(f"USE DATABASE {db}")
        cur.execute(f"CREATE SCHEMA IF NOT EXISTS RAW")
        cur.execute("USE SCHEMA RAW")

        cur.execute(f"""
            CREATE TABLE IF NOT EXISTS {target} (
                symbol VARCHAR NOT NULL,
                date DATE NOT NULL,
                open FLOAT,
                high FLOAT,
                low FLOAT,
                volume NUMBER,
                PRIMARY KEY (symbol, date)
            );
        """)
        sql = f"""
            INSERT INTO {target} (symbol, date, open, high, low, volume)
            VALUES (%s, %s, %s, %s, %s, %s)
        """
        cur.execute(f"DELETE FROM {target};")
        for row in records:
            cur.execute(
                sql,
                (
                    symbol,
                    str(row["date"]),
                    float(row["open"]),
                    float(row["high"]),
                    float(row["low"]),
                    float(row["volume"]),
                ),
            )
    except:
```

```

        cur.execute("COMMIT;")
    except Exception as e:
        cur.execute("ROLLBACK;")
        print(e)
        raise
    finally:
        cur.close()
        conn.close()

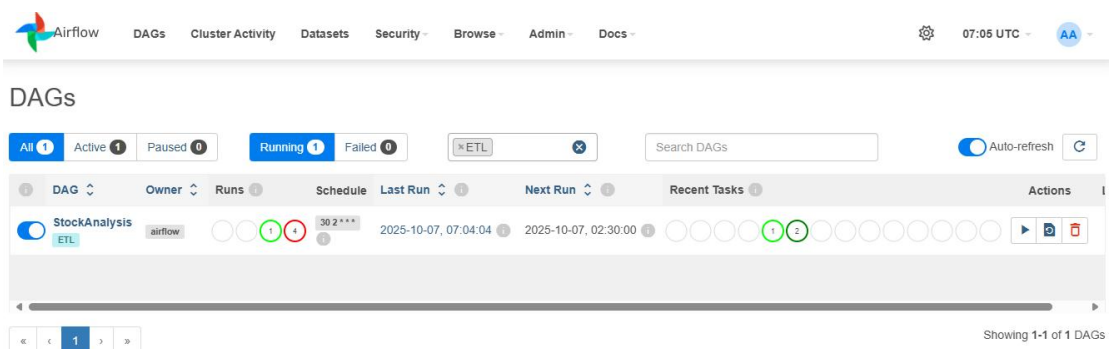
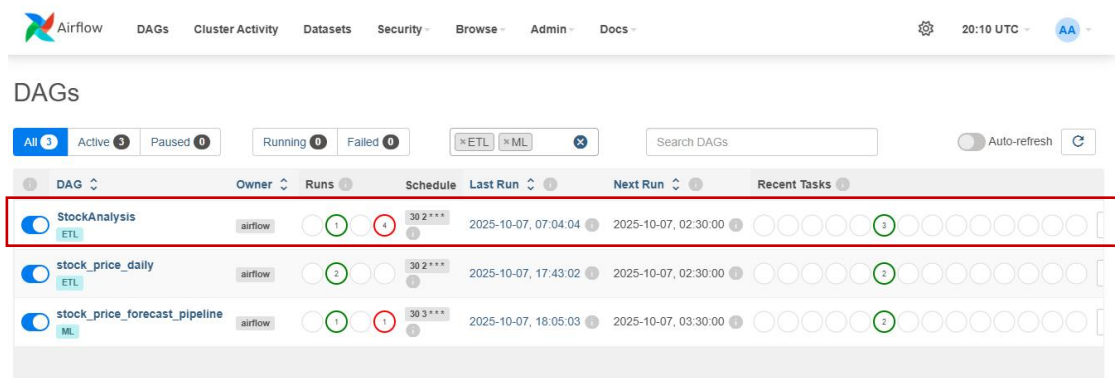
```

```

# Usage example outside the function:
#cur = return_snowflake_conn()
#load_v(cur, database=database, df=df, symbol="AAPL")

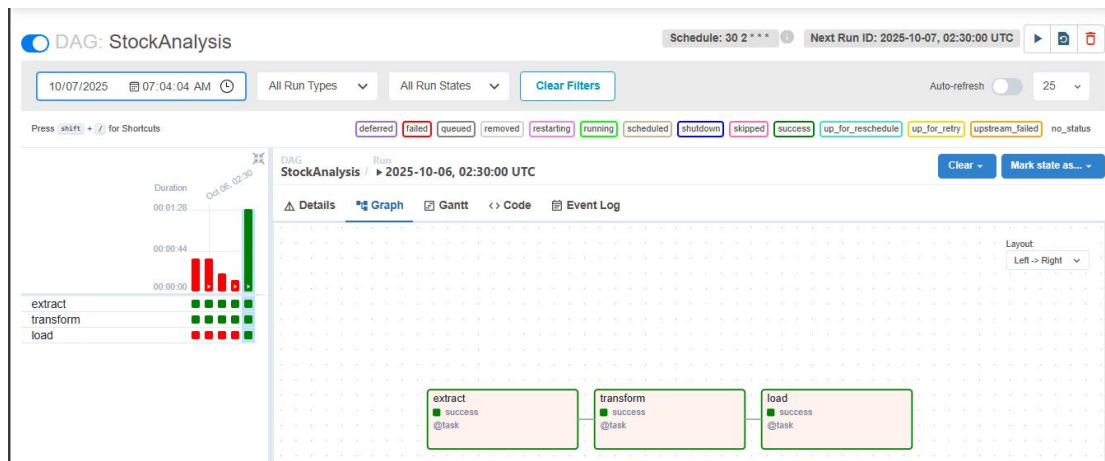
```

- (+2) Capture two screenshot of your Airflow Web UI (examples to follow)
  - One with the Airflow homepage showing the DAG (③)



- The other with the log screen of the DAG (④)

## GRAPH VIEW



## LOG SCREEN

DAG: StockAnalysis / Run: 2025-10-06, 02:30:00 UTC

Clear Mark state as...

Details Graph Gantt Code Event Log

Show Logs After: 10/07/2025 07:06:41 AM

Show Logs Before: 10/07/2025 07:06:41 AM

Events to: Include Exclude

View full cluster Audit Log

| WHEN                     | TASK ID   | MAP INDEX | TRY NUMBER | EVENT   | USER    | DETAILS |
|--------------------------|-----------|-----------|------------|---------|---------|---------|
| 2025-10-07, 07:05:33 UTC | load      |           | 1          | success | airflow |         |
| 2025-10-07, 07:04:14 UTC | load      |           | 1          | running | airflow |         |
| 2025-10-07, 07:04:12 UTC | transform |           | 1          | success | airflow |         |
| 2025-10-07, 07:04:11 UTC | transform |           | 1          | running | airflow |         |
| 2025-10-07, 07:04:09 UTC | extract   |           | 1          | success | airflow |         |
| 2025-10-07, 07:04:08 UTC | extract   |           | 1          | running | airflow |         |

1-6

- (+1) Overall formatting