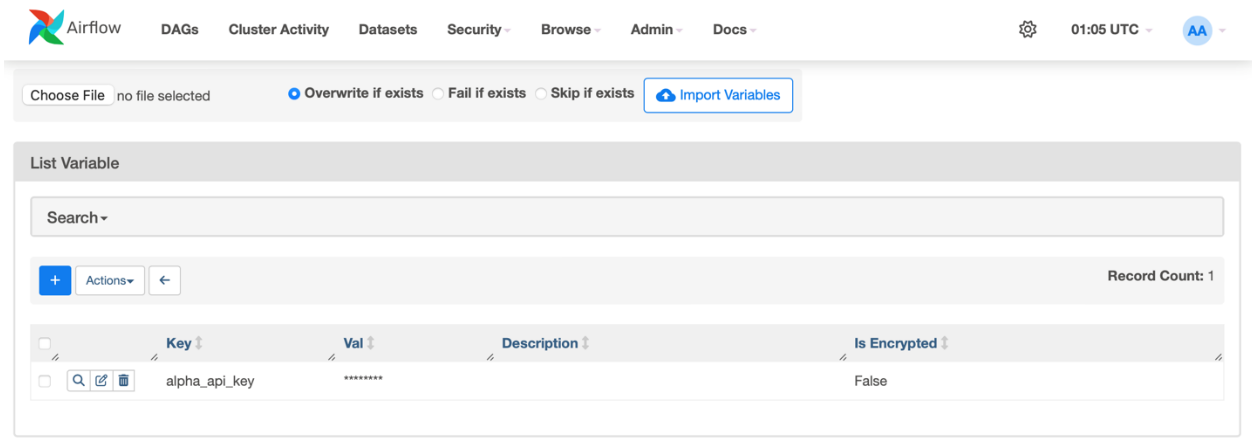
1. (+2) Create tasks using @task decorator.
    a. You can use as many tasks as you want
    b. Schedule the tasks properly (task dependency)

```
@task
def fetch_last_90d(symbol):
# Get the API key securely from Airflow Variables
api_key = Variable.get('alpha_api_key')

# Alpha Vantage endpoint for daily time series
url =
f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&
apikey={api_key}"

# Make GET request to the API
r = requests.get(url)
data = r.json()

# Calculate the cutoff date (90 days ago from today)
cutoff = datetime.today().date() - timedelta(days=90)
results = []

# Parse the API JSON response
for d in data["Time Series (Daily)"]:
trade_date = datetime.strptime(d, "%Y-%m-%d").date()
if trade_date >= cutoff:
stock_info = data["Time Series (Daily)"][d]
stock_info["date"] = d  # add date key for reference
results.append(stock_info)

return results


@task
def load_to_snowflake(cur, records, symbol):
    target_table = "RAW.STOCK_API"  # destination table in Snowflake

    try:
        # Begin SQL transaction
        cur.execute("BEGIN;")

        # Create table if it does not already exist
        cur.execute(f"""
            CREATE TABLE IF NOT EXISTS {target_table} (
                symbol VARCHAR NOT NULL,
                trade_date DATE NOT NULL,
                open  NUMBER(18,4),
                close NUMBER(18,4),
                high  NUMBER(18,4),
                low   NUMBER(18,4),
                volume NUMBER(38,0),
                CONSTRAINT pk_symbol_date PRIMARY KEY (symbol, trade_date) NOT
ENFORCED
            );
        """)

        # Delete existing records to refresh dataset (optional)
        cur.execute(f"DELETE FROM {target_table}")

        # Insert each record individually
        for r in records:
            trade_date = r["date"]
            open_ = r["1. open"]
            high_ = r["2. high"]
            low_ = r["3. low"]
            close_ = r["4. close"]
            volume_ = r["5. volume"]

            # Build INSERT SQL command
            insert_sql = f"""
                INSERT INTO {target_table}
                (symbol, trade_date, open, close, high, low, volume)
                VALUES (
                    '{symbol}',
                    TO_DATE('{trade_date}','YYYY-MM-DD'),
                    {open_}, {close_}, {high_}, {low_}, {volume_}
                );
            """
            cur.execute(insert_sql)

        # Commit the transaction if all inserts succeed
        cur.execute("COMMIT;")
        print(f"[SUCCESS] Loaded {len(records)} records for {symbol} into
{target_table}")

    except Exception as e:
        # Roll back in case of any failure
        cur.execute("ROLLBACK;")
        print(f"[ERROR] Failed to load data for {symbol}: {e}")
        raise


with DAG(
    dag_id='AlphaVantage_to_Snowflake',
    start_date=datetime(2025, 9, 29),
    catchup=False,
    tags=['ETL', 'StockAPI'],
    schedule='30 2 * * *'  # cron format (02:30 UTC)
) as dag:

    # Stock symbol to load
    symbol = "ELV"

    # Get a Snowflake connection cursor
    cur = return_snowflake_conn()

    # Define the Airflow task flow
    fetch_task = fetch_last_90d(symbol)
    load_task = load_to_snowflake(cur, fetch_task, symbol)

    # Task dependency chain
    fetch_task >> load_task
```

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

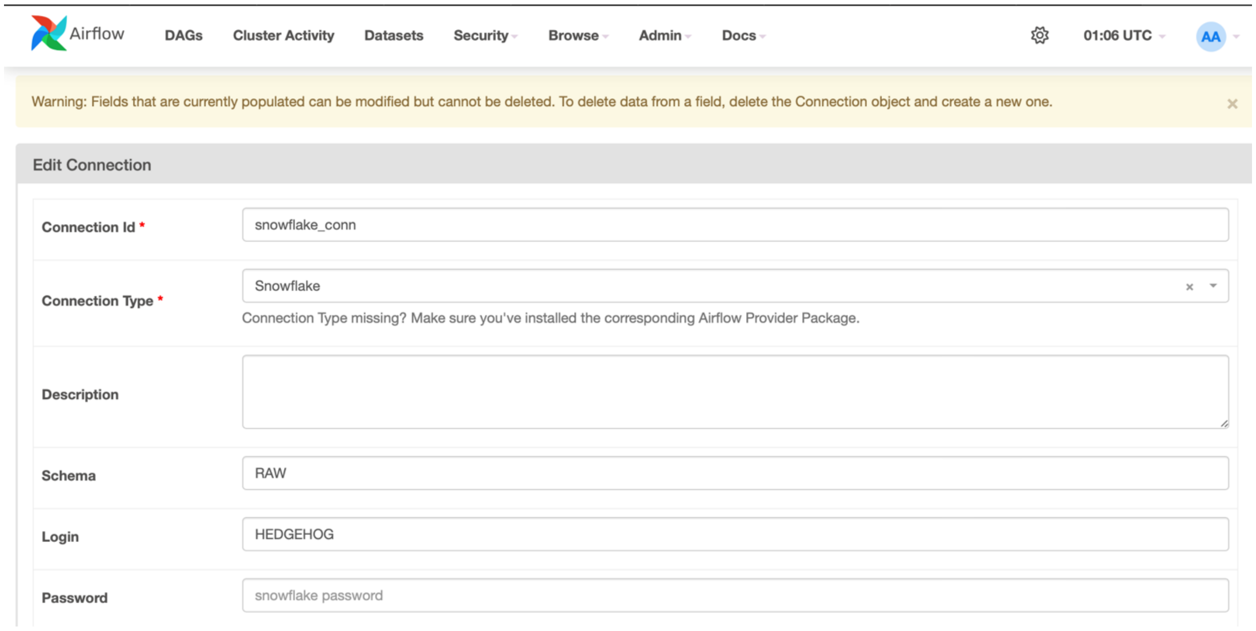


List Variable

Record Count: 1

Key	Val	Description	Is Encrypted
alpha_api_key	********		False

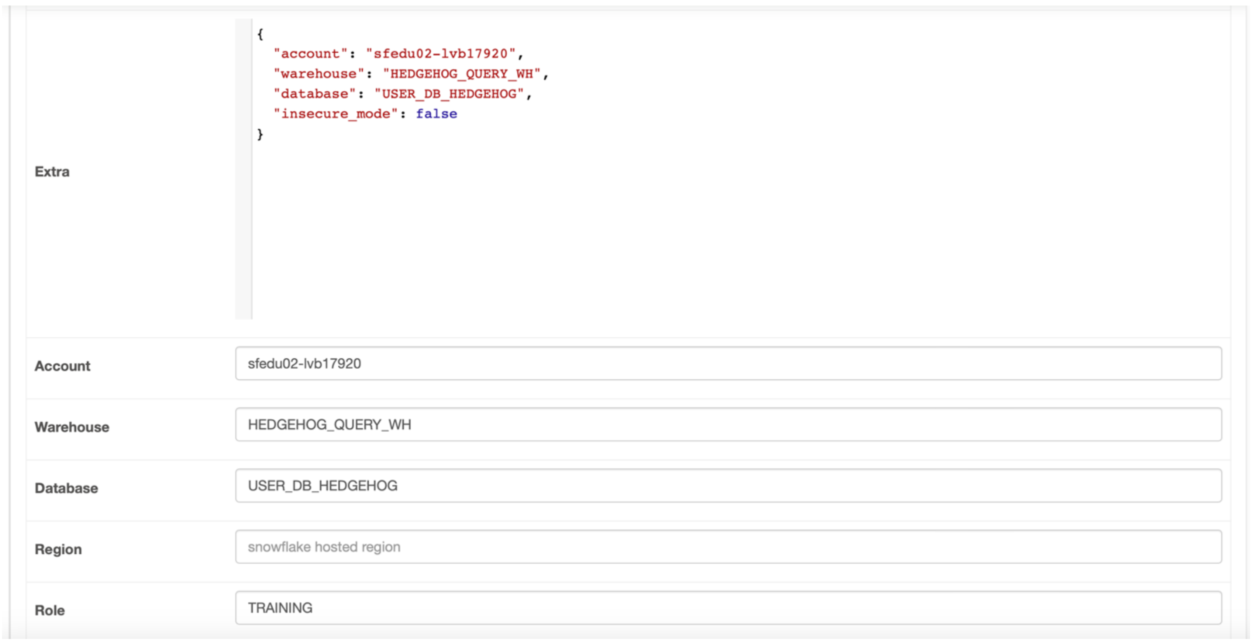
3. (+2) Set up Snowflake Connection (refer to GitHub link Links to an external site.)
    a. Use the connection in your code
    b. Capture the Connection detail page screenshot (an example will be provided ②)



Warning: Fields that are currently populated can be modified but cannot be deleted. To delete data from a field, delete the Connection object and create a new one.

Edit Connection

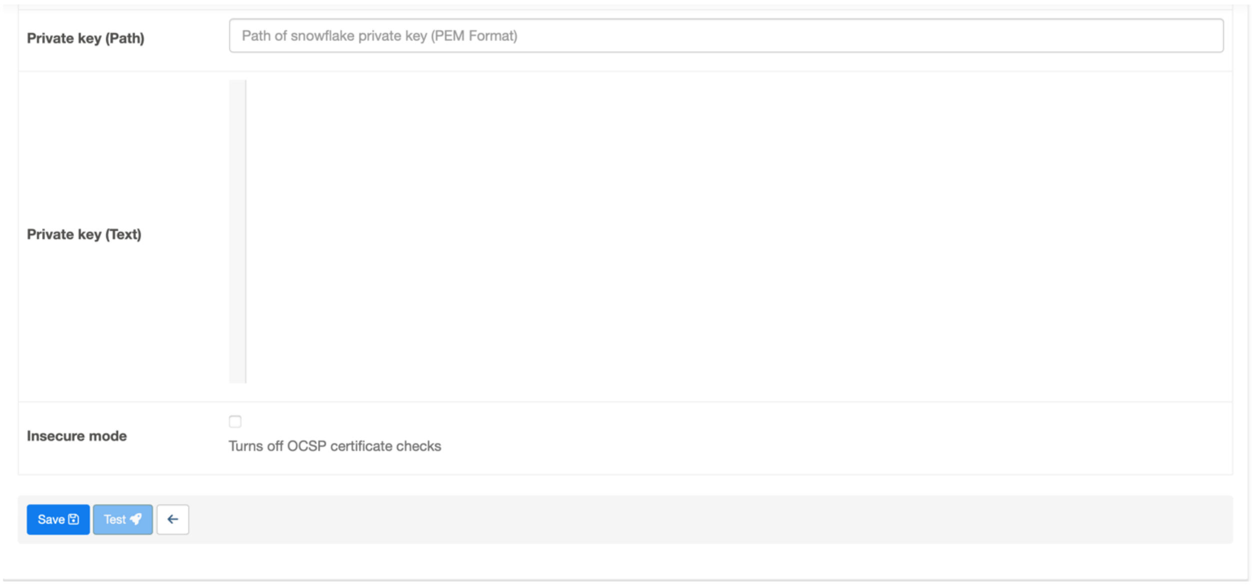
Connection Id *	snowflake_conn
Connection Type *	Snowflake Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Schema	RAW
Login	HEDGEHOG
Password	snowflake password



Extra

```
{
  "account": "sfedu02-lvb17920",
  "warehouse": "HEDGEHOG_QUERY_WH",
  "database": "USER_DB_HEDGEHOG",
  "insecure_mode": false
}
```

Account	sfedu02-lvb17920
Warehouse	HEDGEHOG_QUERY_WH
Database	USER_DB_HEDGEHOG
Region	snowflake hosted region
Role	TRAINING



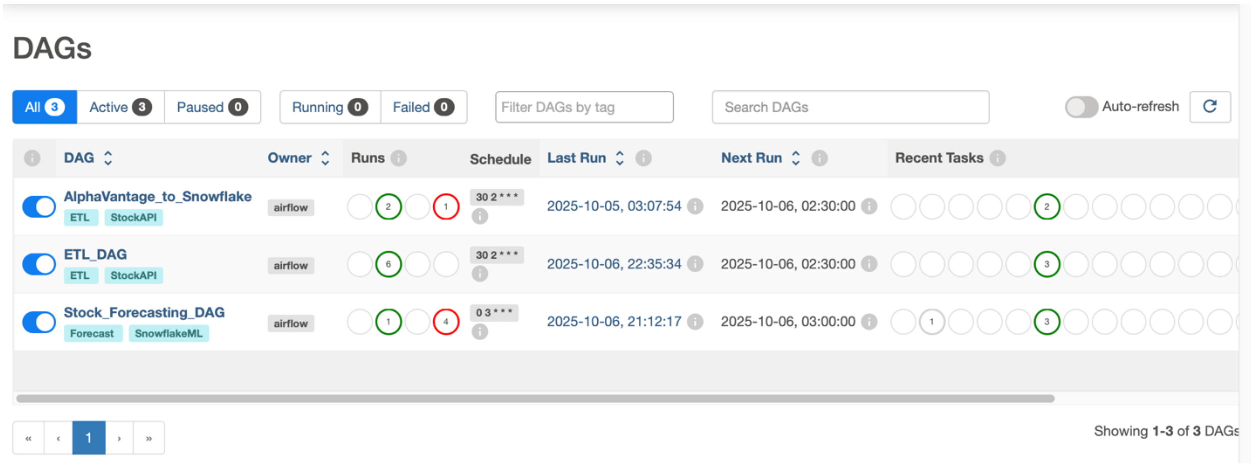
Private key (Path)	Path of snowflake private key (PEM Format)
Private key (Text)	
Insecure mode	☐ Turns off OCSP certificate checks

4. (+5) Ensure the overall DAG is implemented properly and runs successfully
   a. A github link with the entire code needs to be submitted (2 pts)
   b. Implement the same full refresh using SQL transaction (3 pts)

Code Link: https://github.com/danwaseem/SJSU-DATA226/blob/main/HW5/hw5_final.py

Airflow Link: https://github.com/danwaseem/SJSU-DATA226/tree/main/Airflow

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

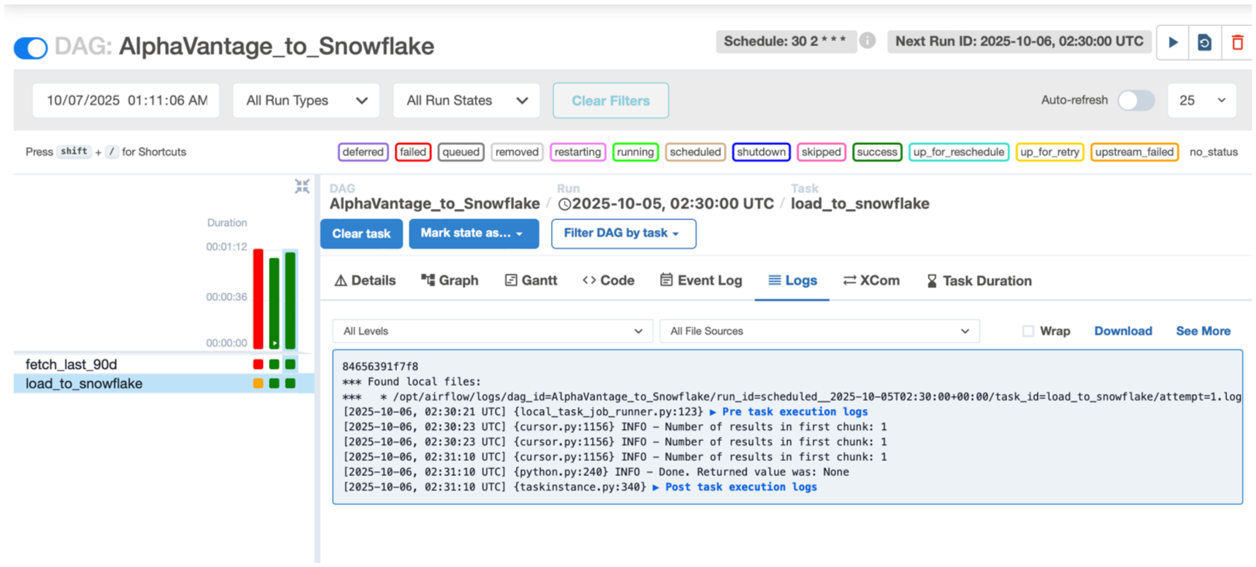


DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
AlphaVantage_to_Snowflake ETL StockAPI	airflow	2 1	30 2 * * *	2025-10-05, 03:07:54	2025-10-06, 02:30:00	2
ETL_DAG ETL StockAPI	airflow	6	30 2 * * *	2025-10-06, 22:35:34	2025-10-06, 02:30:00	3
Stock_Forecasting_DAG Forecast SnowflakeML	airflow	1 4	0 3 * * *	2025-10-06, 21:12:17	2025-10-06, 03:00:00	1 3

Showing **1-3** of **3** DAGs

   b. The other with the log screen of the DAG (④)


DAG: AlphaVantage_to_Snowflake
Schedule: 30 2 * * *
Next Run ID: 2025-10-06, 02:30:00 UTC
10/07/2025 01:11:06 AM
All Run Types
All Run States
Clear Filters
Auto-refresh
25
Press shift + / for Shortcuts
deferred failed queued removed restarting running scheduled shutdown skipped success up_for_reschedule up_for_retry upstream_failed no_status
Duration
00:01:12
00:00:36
00:00:00
fetch_last_90d
load_to_snowflake
DAG AlphaVantage_to_Snowflake / Run 2025-10-05, 02:30:00 UTC / Task load_to_snowflake
Clear task
Mark state as...
Filter DAG by task
Details Graph Gantt Code Event Log Logs XCom Task Duration
All Levels
All File Sources
Wrap
Download
See More

```
84656391f7f8
*** Found local files:
***   * /opt/airflow/logs/dag_id=AlphaVantage_to_Snowflake/run_id=scheduled__2025-10-05T02:30:00+00:00/task_id=load_to_snowflake/attempt=1.log
[2025-10-06, 02:30:21 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-10-06, 02:30:23 UTC] {cursor.py:1156} INFO - Number of results in first chunk: 1
[2025-10-06, 02:30:23 UTC] {cursor.py:1156} INFO - Number of results in first chunk: 1
[2025-10-06, 02:31:10 UTC] {cursor.py:1156} INFO - Number of results in first chunk: 1
[2025-10-06, 02:31:10 UTC] {python.py:240} INFO - Done. Returned value was: None
[2025-10-06, 02:31:10 UTC] {taskinstance.py:340} ▶ Post task execution logs
```

6. (+1) Overall formatting