DATA 226- DATAWAREHOUSE

Homework 5

Name : Vimalanandhan Sivanandham

SJSU ID: 017596436

Porting homework #4 to Airflow (13 pts)

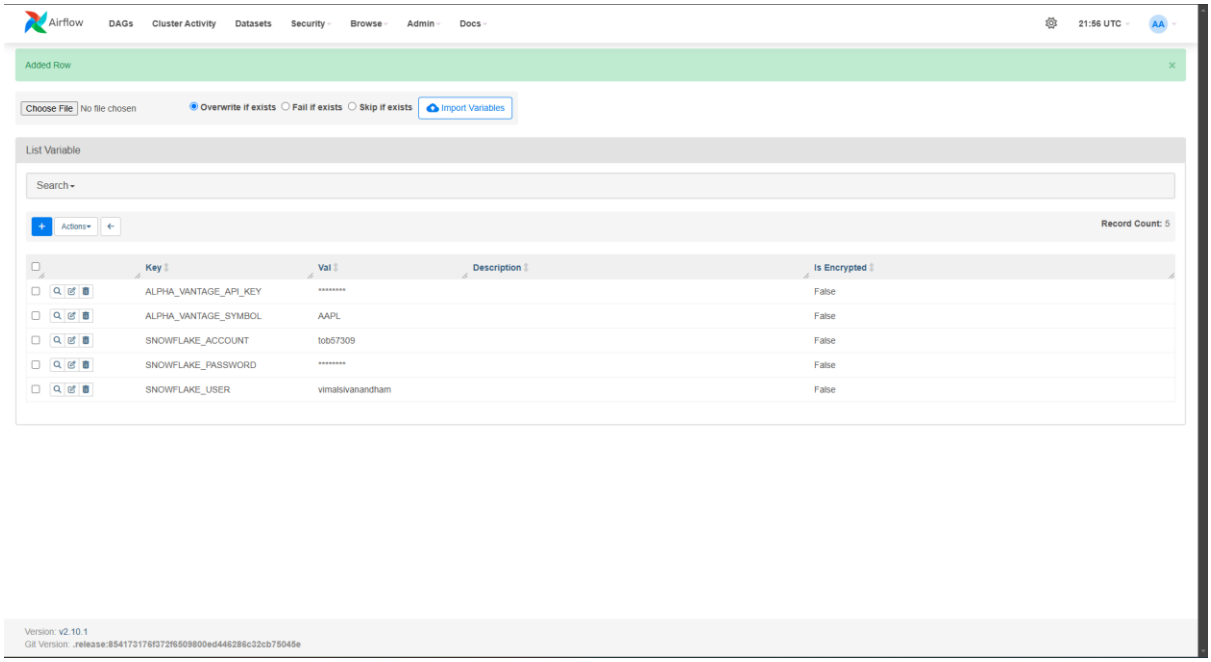
- (+2) Create tasks using @task decorator (refer to GitHub linkLinks to an external site.)
    - You can use as many tasks as you want
    - Schedule the tasks properly (task dependency)

```python
from airflow import DAG
from airflow.models import Variable
from airflow.decorators import task
import snowflake.connector
from datetime import datetime
import requests
import logging

def return_snowflake_conn():
    """Initialize Snowflake connection manually using `snowflake.connector`."""

    # Retrieve credentials from Google Colab
    snowflake_user = Variable.get("SNOWFLAKE_USER")
    snowflake_password = Variable.get("SNOWFLAKE_PASSWORD")
    snowflake_account = Variable.get("SNOWFLAKE_ACCOUNT")

    # Connect to Snowflake
    conn = snowflake.connector.connect(
        user=snowflake_user,
        password=snowflake_password,
        account=snowflake_account
    )
    return conn.cursor()

@task
def extract():
    """Extract AAPL stock data from Alpha Vantage API"""
    api_key = Variable.get("ALPHA_VANTAGE_API_KEY")  # Get API key from Airflow Variables
    symbol = "AAPL"
    url = f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={api_key}&outputsize=compact"

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json().get("Time Series (Daily)", {})
        logging.info(f"Extracted {len(data)} records")
        return data  # XCom push
    except Exception as e:
        logging.error(f"Error in extract: {str(e)}")
        raise
```

```python
@task
def extract():
    """Extract AAPL stock data from Alpha Vantage API"""
    api_key = Variable.get("ALPHA_VANTAGE_API_KEY")  # Get API key from Airflow Variables
    symbol = "AAPL"
    url = f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={api_key}&outputsize=compact"

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json().get("Time Series (Daily)", {})
        logging.info(f"Extracted {len(data)} records")
        return data  # XCom push
    except Exception as e:
        logging.error(f"Error in extract: {str(e)}")
        raise

@task
def transform(data):
    """Transform extracted stock data into structured format"""
    records = []
    for date, values in data.items():
        records.append([
            date, float(values["1. open"]), float(values["2. high"]),
            float(values["3. low"]), float(values["4. close"]),
            int(values["5. volume"])
        ])
    logging.info(f"Transformed {len(records)} records")
    return records  # XCom push

@task
def load(records):
    """Load transformed data into Snowflake"""
    cur = return_snowflake_conn()
    try:
        cur.execute("BEGIN;")
        cur.execute("""
            CREATE TABLE IF NOT EXISTS stock_data.raw.stock_data (
                date DATE PRIMARY KEY,
                open FLOAT,
                high FLOAT,
                low FLOAT,
                close FLOAT,
                volume INT
            );
        """)
        cur.execute("DELETE FROM stock_data.raw.stock_data;")  # Full refresh
```

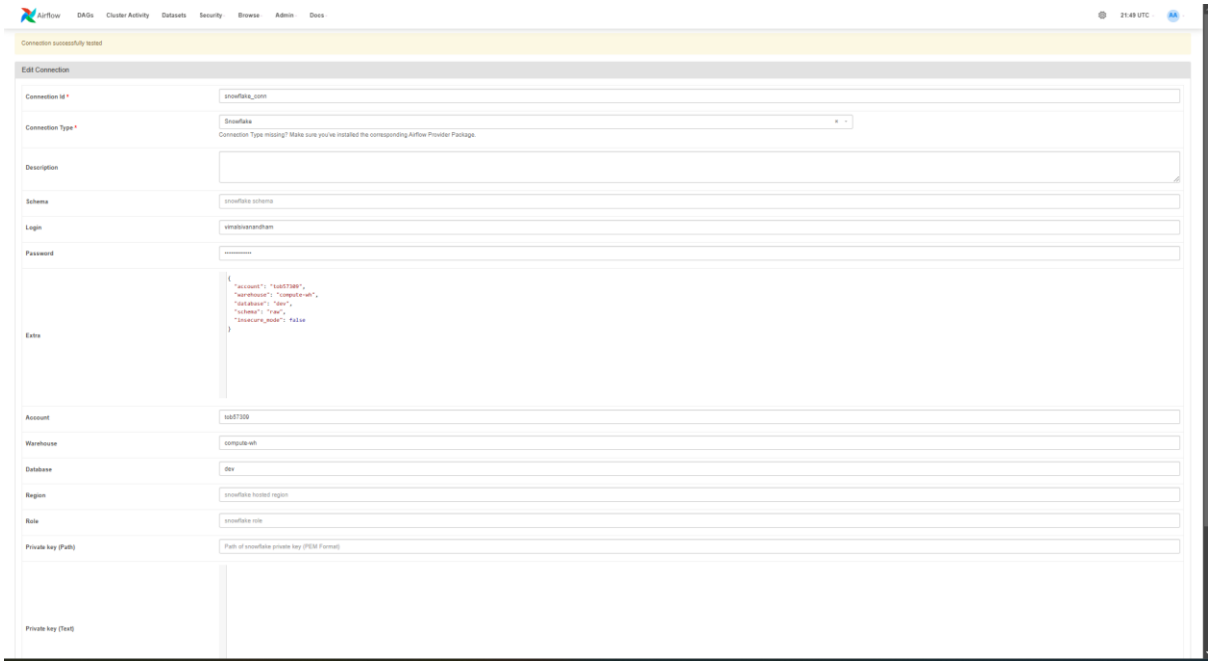
- (+1) Set up a variable for Alpha Vantage API key
    - Use the variable in your code (Variable.get)
    - Capture the Admin -> Variables screenshot (an example will be provided ①)

```python
from airflow import DAG
from airflow.models import Variable
from airflow.decorators import task
import snowflake.connector
from datetime import datetime
import requests
import logging

def return_snowflake_conn():
    """Initialize Snowflake connection manually using `snowflake.connector`."""

    # Retrieve credentials from Google Colab
    snowflake_user = Variable.get("SNOWFLAKE_USER")
    snowflake_password = Variable.get("SNOWFLAKE_PASSWORD")
    snowflake_account = Variable.get("SNOWFLAKE_ACCOUNT")

    # Connect to Snowflake
    conn = snowflake.connector.connect(
        user=snowflake_user,
        password=snowflake_password,
        account=snowflake_account
    )
    return conn.cursor()

@task
def extract():
    """Extract AAPL stock data from Alpha Vantage API"""
    api_key = Variable.get("ALPHA_VANTAGE_API_KEY")  # Get API key from Airflow Variables
    symbol = "AAPL"
    url = f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={api_key}&outputsize=compact"

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json().get("Time Series (Daily)", {})
        logging.info(f"Extracted {len(data)} records")
        return data  # XCom push
    except Exception as e:
        logging.error(f"Error in extract: {str(e)}")
        raise
```



- (+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 ②)

```python
from airflow import DAG
from airflow.models import Variable
from airflow.decorators import task
import snowflake.connector
from datetime import datetime
import requests
import logging

def return_snowflake_conn():
    """Initialize Snowflake connection manually using `snowflake.connector`."""

    # Retrieve credentials from Google Colab
    snowflake_user = Variable.get("SNOWFLAKE_USER")
    snowflake_password = Variable.get("SNOWFLAKE_PASSWORD")
    snowflake_account = Variable.get("SNOWFLAKE_ACCOUNT")

    # Connect to Snowflake
    conn = snowflake.connector.connect(
        user=snowflake_user,
        password=snowflake_password,
        account=snowflake_account
    )
    return conn.cursor()

@task
def extract():
    """Extract AAPL stock data from Alpha Vantage API"""
    api_key = Variable.get("ALPHA_VANTAGE_API_KEY")  # Get API key from Airflow Variables
    symbol = "AAPL"
    url = f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={api_key}&outputsize=compact"

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json().get("Time Series (Daily)", {})
        logging.info(f"Extracted {len(data)} records")
        return data  # XCom push
    except Exception as e:
        logging.error(f"Error in extract: {str(e)}")
        raise
```



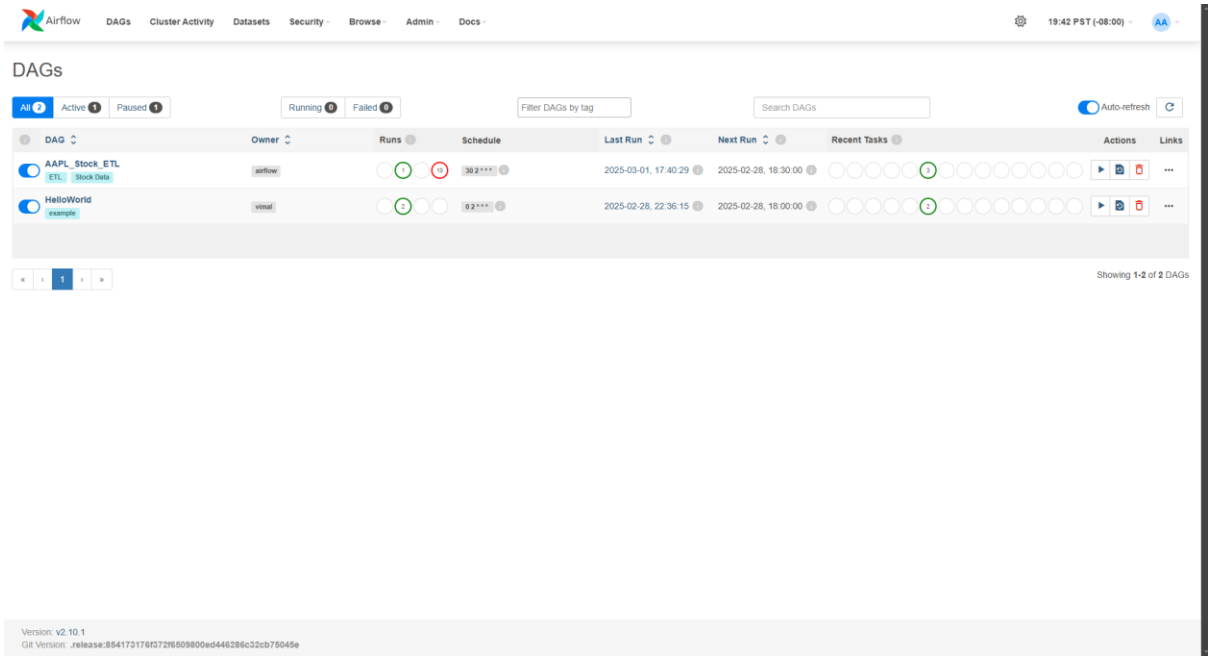
- (+5) Ensure the overall DAG is implemented properly and runs successfully
    - A github link with the entire code needs to be submitted (2 pts)
    - Implement the same full refresh using SQL transaction (3 pts)

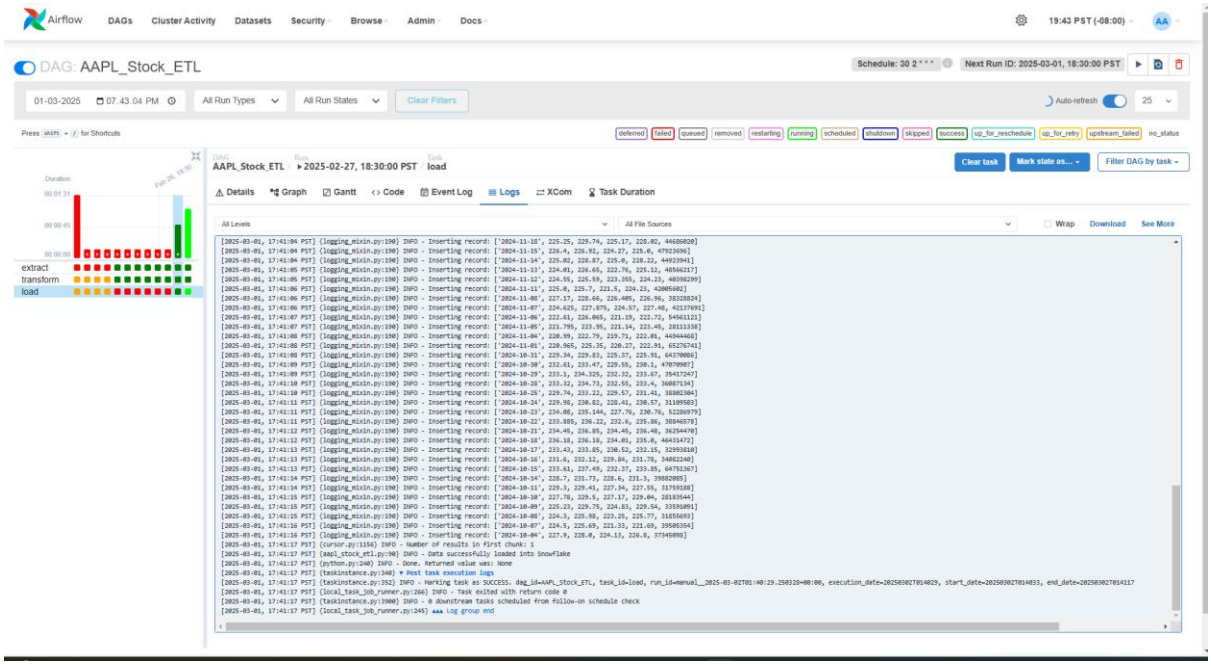
https://github.com/Vimalanandhan/DATA-226---DATAWAREHOUSE/tree/main/Homework/Homework5

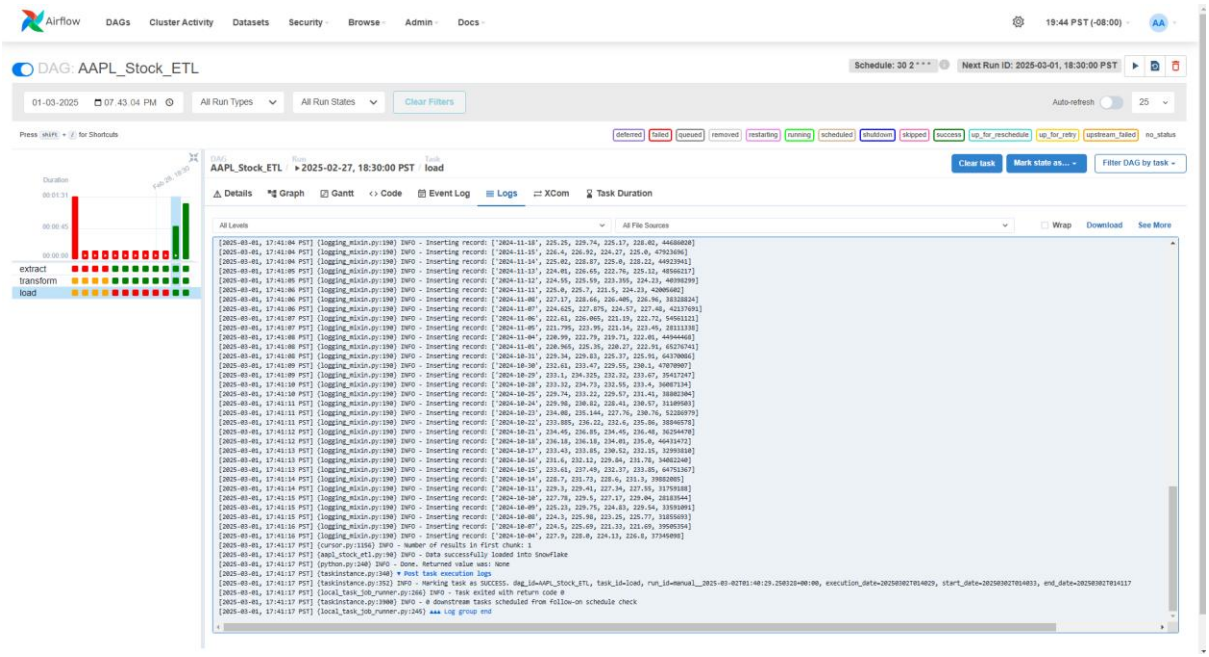
```python
@task
def extract():
    """Extract AAPL stock data from Alpha Vantage API"""
    api_key = Variable.get("ALPHA_VANTAGE_API_KEY")  # Get API key from Airflow Variables
    symbol = "AAPL"
    url = f"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={api_key}&outputsize=compact"

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json().get("Time Series (Daily)", {})
        logging.info(f"Extracted {len(data)} records")
        return data  # XCom push
    except Exception as e:
        logging.error(f"Error in extract: {str(e)}")
        raise

@task
def transform(data):
    """Transform extracted stock data into structured format"""
    records = []
    for date, values in data.items():
        records.append([
            date, float(values["1. open"]), float(values["2. high"]),
            float(values["3. low"]), float(values["4. close"]),
            int(values["5. volume"])
        ])
    logging.info(f"Transformed {len(records)} records")
    return records  # XCom push

@task
def load(records):
    """Load transformed data into Snowflake"""
    cur = return_snowflake_conn()
    try:
        cur.execute("BEGIN;")
        cur.execute("""
            CREATE TABLE IF NOT EXISTS stock_data.raw.stock_data (
                date DATE PRIMARY KEY,
                open FLOAT,
                high FLOAT,
                low FLOAT,
                close FLOAT,
                volume INT
            );
        """)
        cur.execute("DELETE FROM stock_data.raw.stock_data;")  # Full refresh
```

```python
def load(records):
        cur.execute("BEGIN;")
        cur.execute("""
            CREATE TABLE IF NOT EXISTS stock_data.raw.stock_data (
                date DATE PRIMARY KEY,
                open FLOAT,
                high FLOAT,
                low FLOAT,
                close FLOAT,
                volume INT
            );
        """)
        cur.execute("DELETE FROM stock_data.raw.stock_data;")  # Full refresh

        for record in records:
            # Debug: Print each record before inserting
            print(f"Inserting record: {record}")

            # Check for NULL values and replace them
            cleaned_record = [
                value if value is not None else 0  # Replace NULLs with 0
                for value in record
            ]

            sql = f"""
            INSERT INTO stock_data.raw.stock_data (date, open, high, low, close, volume)
            VALUES ('{cleaned_record[0]}', {cleaned_record[1]}, {cleaned_record[2]}, {cleaned_record[3]}, {cleaned_record[4]}, {cleaned_record[5]});
            """
            cur.execute(sql)

        cur.execute("COMMIT;")
        logging.info("Data successfully loaded into Snowflake")
    except Exception as e:
        cur.execute("ROLLBACK;")
        logging.error(f"Error in load: {str(e)}")
        raise

# Define the Airflow DAG
with DAG(
    dag_id='AAPL_Stock_ETL',
    start_date=datetime(2024, 9, 21),
    catchup=False,
    tags=['ETL', 'Stock Data'],
    schedule_interval='30 2 * * *'  # Run daily at 2:30 AM UTC
) as dag:
    data = extract()
    transformed_data = transform(data)
    load(transformed_data)
```

- (+2) Capture two screenshot of your Airflow Web UI (examples to follow)
    - One with the Airlow homepage showing the DAG (③)
    - The other with the log screen of the DAG (④)

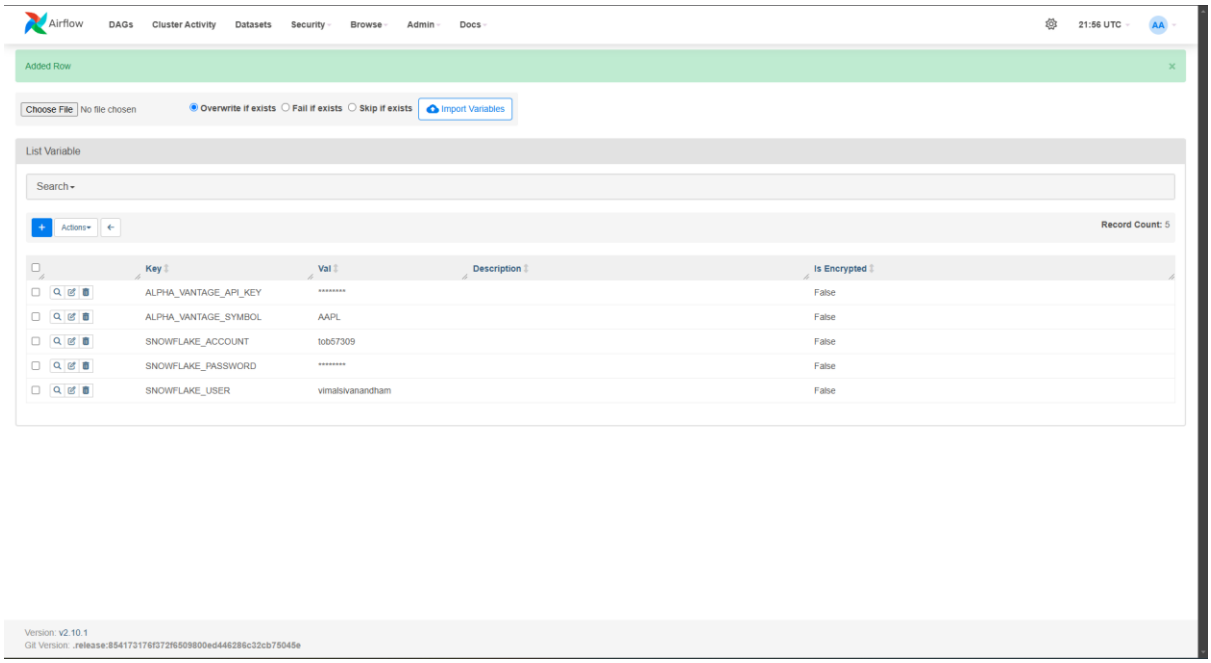

Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
19:42 PST (-08:00)
DAGs
All 2
Active 1
Paused 1
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
Links
AAPL_Stock_ETL
ETL
Stock Data
airflow
30 2 * * *
2025-03-01, 17:40:29
2025-02-28, 18:30:00
HelloWorld
example
vimal
0 2 * * *
2025-02-28, 22:36:15
2025-02-28, 18:00:00
Showing 1-2 of 2 DAGs
Version: v2.10.1
Git Version: .release:854173176f372f6509800ed446286c32cb75045e

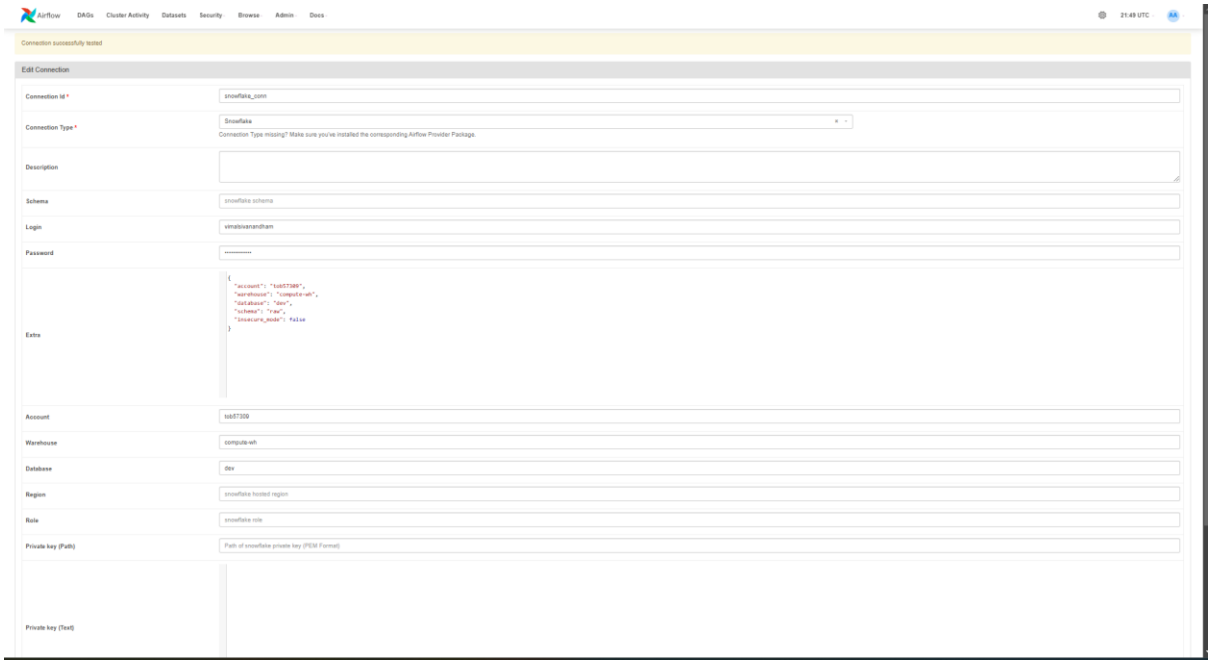

Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
19:43 PST (-08:00)
DAG: AAPL_Stock_ETL
Schedule: 30 2 * * *
Next Run ID: 2025-03-01, 18:30:00 PST
01-03-2025
07.43.04 PM
All Run Types
All Run States
Clear Filters
Auto-refresh
25
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
DAG
AAPL_Stock_ETL
Run
2025-02-27, 18:30:00 PST
Task
load
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
extract
transform
load

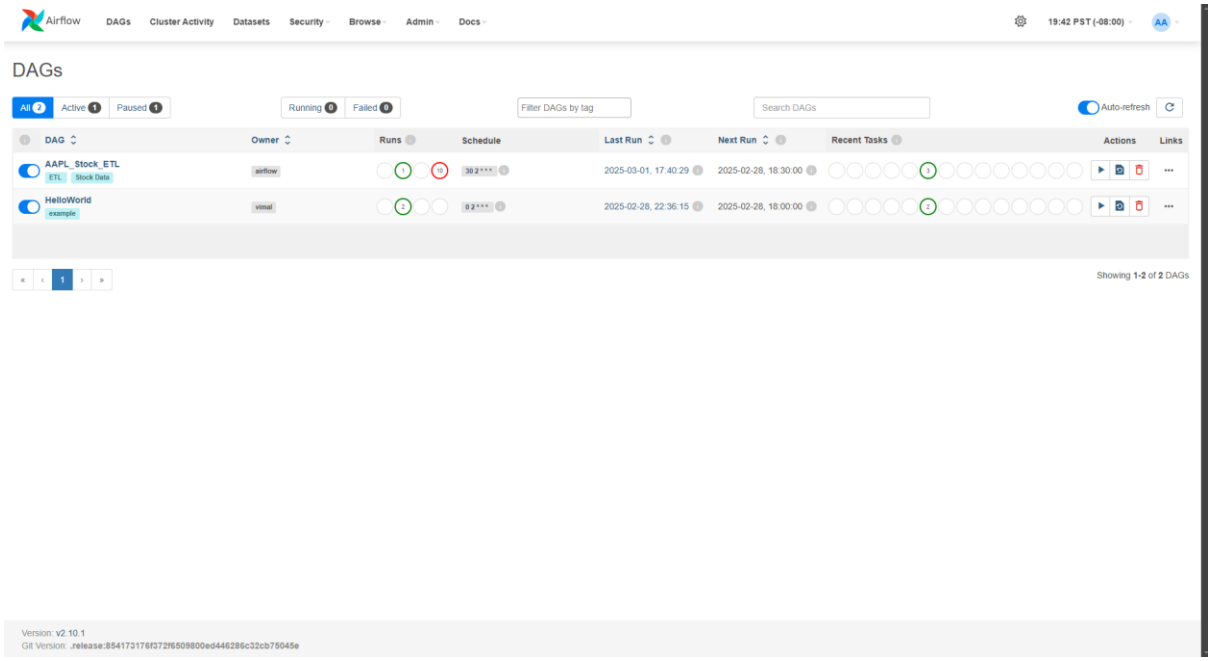


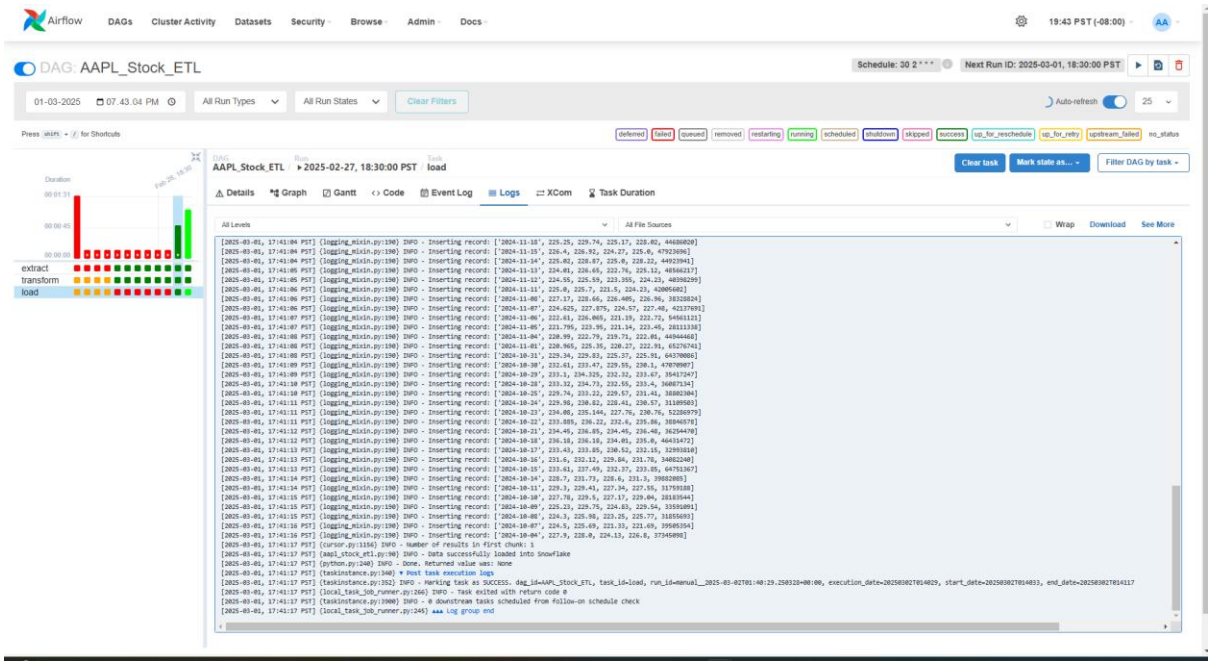
- (+1) Overall formatting

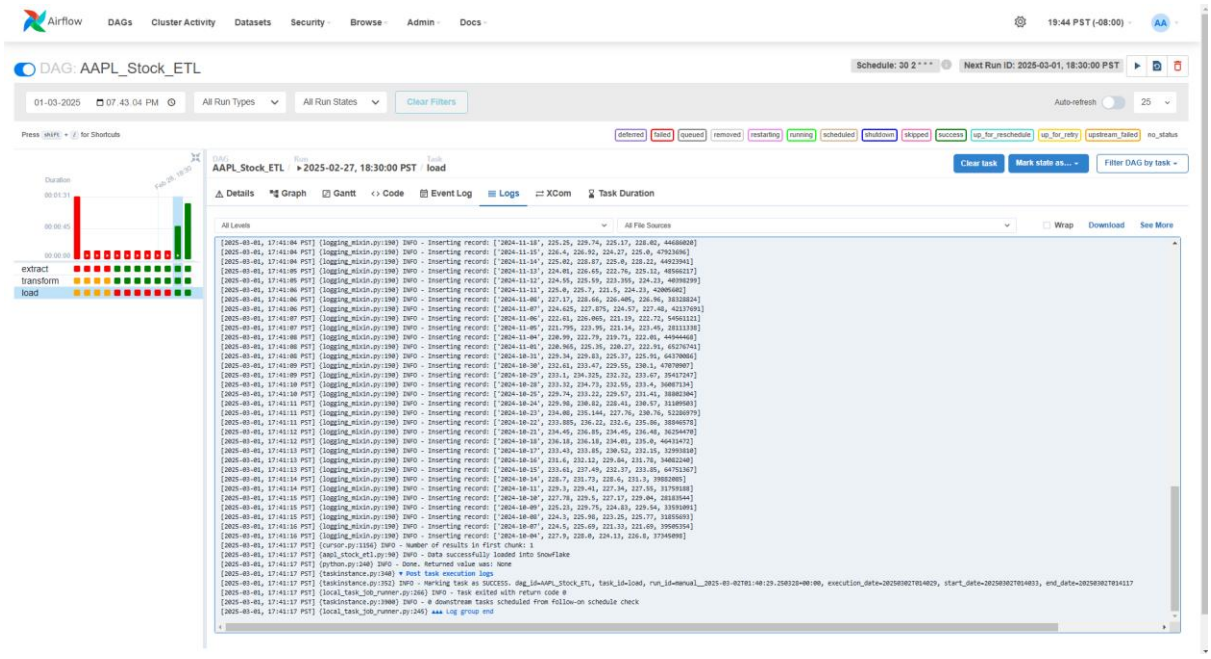
4 screenshot examples are in the lecture notes (from slides 62 to 64)




Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
Connection sucessfully tested
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
vimalsivanandham
Password
Extra
{
"account": "tob57309",
"warehouse": "compute-wh",
"database": "dev",
"schema": "raw",
"insecure_mode": false
}
Account
tob57309
Warehouse
compute-wh
Database
dev
Region
snowflake hosted region
Role
snowflake role
Private key (Path)
Path of snowflake private key (PEM Format)
Private key (Text)


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
19:42 PST (-08:00)
AA
DAGs
All 2
Active 1
Paused 1
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
Links
AAPL_Stock_ETL
ETL
Stock Data
airflow
30 2 * * *
2025-03-01, 17:40:29
2025-02-28, 18:30:00
HelloWorld
example
vimal
0 2 * * *
2025-02-28, 22:36:15
2025-02-28, 18:00:00
Showing 1-2 of 2 DAGs
Version: v2.10.1
Git Version: .release:854173176f372f6509800ed446286c32cb75045e


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
19:43 PST (-08:00)
DAG: AAPL_Stock_ETL
Schedule: 30 2 * * *
Next Run ID: 2025-03-01, 18:30:00 PST
01-03-2025
07.43.04 PM
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
DAG: AAPL_Stock_ETL
Run: 2025-02-27, 18:30:00 PST
Task: load
Clear task
Mark state as...
Filter DAG by task
Details
Graph
Gantt
Code
Event Log
Logs
XCom
Task Duration
Duration
extract
transform
load
All Levels
All File Sources
Wrap
Download
See More


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
19:44 PST (-08:00)
DAG: AAPL_Stock_ETL
Schedule: 30 2 * * *
Next Run ID: 2025-03-01, 18:30:00 PST
01-03-2025
07.43.04 PM
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
DAG: AAPL_Stock_ETL
Run: 2025-02-27, 18:30:00 PST
Task: load
Clear task
Mark state as...
Filter DAG by task
Details
Graph
Gantt
Code
Event Log
Logs
XCom
Task Duration
Duration
extract
transform
load
All Levels
All File Sources
Wrap
Download
See More