

Stock Data Load

Stock Data Load in Airflow

Shweta Ajay Shinde

Masters in Data Analytics, San Jose State University

Data 226: Data warehousing

Instructor: Keeyong Han

10th Oct 2024

Stock Data Load

Load Stocks Data in Snowflake Table using Airflow

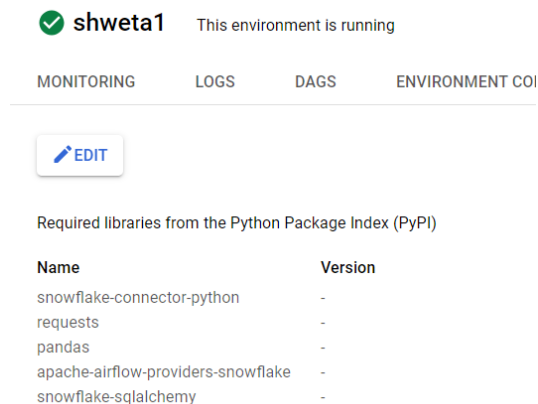
- (+1) Import all the required python modules.

We need to import all required python modules for smooth running of the code.

```
#Step 1 Import all required modules
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
from datetime import datetime
import snowflake.connector
import requests
import pandas as pd
```

- (+1) Ensure that any missing package(s) are added to the PYPI packages
Step 1: After login into Cloud Composer click on PYPI packages.
Step 2: Add all the require packages and save it



✓ shweta1 This environment is running

MONITORING LOGS DAGS ENVIRONMENT CONFIGURATION

[EDIT](#)

Required libraries from the Python Package Index (PyPI)

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

- (+3) Create tasks using @task decorator (refer to [GitHub link to an external site.](#)).
The @task decorator tells the system that a function is a special task that can be run separately. It allows the function to work in the background or at the same time as other tasks. This is useful when you want to do multiple things at once, like processing data or sending emails. By using @task, you make it easier to organize and run different tasks in a program. For task we need to import: **from airflow.decorators import task**

```
@task
def return_last_90d_price(symbol):
    """
    - return the last 90 days of the stock prices of symbol as a list of json strings
    """
    vantage_api_key = Variable.get('apikey')
    url = f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={vantage_api_key}'
    r = requests.get(url)
    data = r.json()
    symbol_value = data["Meta Data"]["2. Symbol"]
```

Stock Data Load

```
@task
def create_load_incremental(records):
    staging_table = "dev.stock.merck_stock_stage"
    target_table = "dev.stock.merck_stock_price_incremental"
    conn = return_snowflake_conn()
    try:
        conn.execute(f"""
            CREATE TABLE IF NOT EXISTS {target_table} (
                date DATE PRIMARY KEY NOT NULL,
                open DECIMAL(10, 2) NOT NULL,
                high DECIMAL(10, 2) NOT NULL,
                low DECIMAL(10, 2) NOT NULL,
                close DECIMAL(10, 2) NOT NULL,
            )
        """)
```

```
@task
def create_load_full(records):
    target_table = "dev.stock.merck_stock_price_full"
    conn = return_snowflake_conn()
    try:
        conn.execute(f"""
            CREATE OR REPLACE TABLE {target_table} (
                date DATE PRIMARY KEY NOT NULL,
                open DECIMAL(10, 2) NOT NULL,
                high DECIMAL(10, 2) NOT NULL,
                low DECIMAL(10, 2) NOT NULL,
                close DECIMAL(10, 2) NOT NULL,
            )
        """)
```

- (+1) Set up a variable for Alpha Vantage API key

Under Admin we find option Variable where we can store all our confidential information in key value pair.

	Key ↕	Val ↕	Description ↕
<input type="checkbox"/>	apikey	*****	Api key to access source Data
<input type="checkbox"/>	SNOWFLAKE_ACCOUNT	mihzyek-aeb64943	Snowflake account details
<input type="checkbox"/>	SNOWFLAKE_DATABASE	dev	
<input type="checkbox"/>	SNOWFLAKE_PASSWORD	*****	Snowflake Password
<input type="checkbox"/>	SNOWFLAKE_SCHEMA	stock	
<input type="checkbox"/>	SNOWFLAKE_USER	shweta12shinde	Snowflake user
<input type="checkbox"/>	SNOWFLAKE_WAREHOUSE	compute_wh	

In code we need **from airflow.models import Variable**

```
def return_snowflake_conn():
    user_id = Variable.get('SNOWFLAKE_USER')
    password = Variable.get('SNOWFLAKE_PASSWORD')
    account = Variable.get('SNOWFLAKE_ACCOUNT')

    conn = snowflake.connector.connect(
        user=user_id,
        password=password,
        account=account,
        warehouse=Variable.get('SNOWFLAKE_WAREHOUSE'),
        database=Variable.get('SNOWFLAKE_DATABASE'),
        schema=Variable.get('SNOWFLAKE_SCHEMA')
    )
    return conn.cursor()
```

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

We create this connection to connect with Snowflake

Stock Data Load

Connection Id *	snowflake_conn
Connection Type *	Snowflake Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package
Description	
Schema	stock
Login	shweta12shinde
Password	*****
	<pre>{ "account": "mihzyek-aeb64943", "warehouse": "compute_wh", "database": "dev", "insecure_mode": true }</pre>

Code Screenshot:-


































```
def return_snowflake_conn():
    # Initialize the SnowflakeHook
    hook = SnowflakeHook(snowflake_conn_id='snowflake_conn')
    # Execute the query and fetch results
    conn = hook.get_conn()
    return conn.cursor()
```

- (+4) Ensure the overall DAG runs successfully

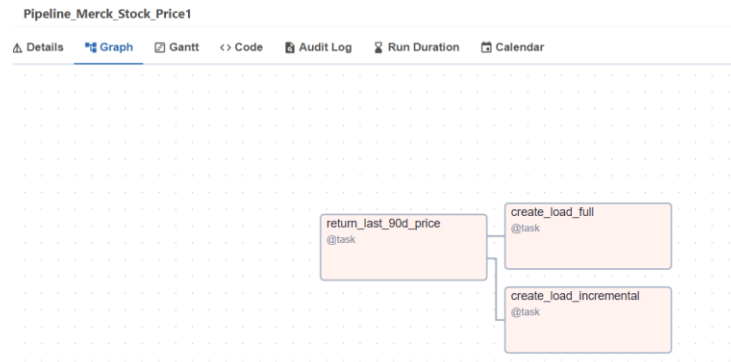
GitHub link to the code:-

https://github.com/ShindeShwetaK/DW_Project/blob/main/Merck_stock_pipeline_new.py

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

All 3	Active 3	Paused 0	Running 0	Failed 0	Filter DAGs by tag	Search DAGs	
1	DAG ⌵	Owner ⌵	Runs 1	Schedule	Last Run ⌵ 1	Next Run ⌵ 1	Recent Tasks 1
	airflow_monitoring	airflow	 2095	 */10 * * * *	2024-10-11, 04:30:00 	2024-10-11, 04:40:00 	      1
	Pipeline_Merck_Stock_Price1 ETL	airflow	 2	 @daily	2024-10-10, 23:57:33 	2024-10-11, 00:00:00 	      3
	Pipeline_Merck_Stock_Price_HookConnect ETL	airflow	 1	 @daily	2024-10-10, 00:00:00 	2024-10-11, 00:00:00 	      3

Stock Data Load



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

DAG: Pipeline_Merck_Stock_Price_HookConnect Schedule: @daily Next Run ID: 2024-10-11, 00:00:00 UTC

10/11/2024 07:55:21 PM All Run Types All Run States Clear Filters Auto-refresh 25

Press **SHIFT** + **/** for Shortcuts

Task: Pipeline_Merck_Stock_Price_HookConnect / 2024-10-11, 00:00:00 UTC / create_load_full

Clear task Mark state as... Filter DAG by task

Duration: 00:00:59, 00:00:25, 00:00:00

return_last_90d_price create_load_incremental create_load_full

Logs

default-hostname

```

*** Reading remote logs from Cloud Logging.
[2024-10-11, 04:43:49 UTC] [local_task_job_runner.py:120] ▶ Pre task execution logs
[2024-10-11, 04:43:53 UTC] [base.py:84] INFO - Using connection ID 'snowflake_conn' for task execution.
[2024-10-11, 04:43:53 UTC] [connection.py:413] INFO - Snowflake Connector for Python Version: 3.12.2, Python Version: 3.11.8, Platform: Linux-6.1.100+-x86_64-with-glibc2.31
[2024-10-11, 04:43:53 UTC] [connection.py:1196] INFO - Connecting to GLOBAL Snowflake domain
[2024-10-11, 04:43:53 UTC] [connection.py:1277] INFO - This connection is in OSCP Full Open Mode. TLS certificates would be checked for validity and revocation status. Any other Certificate Revocation
[2024-10-11, 04:43:53 UTC] [connection.py:1287] INFO - THIS CONNECTION IS IN INSECURE MODE. IT MEANS THE CERTIFICATE WILL BE VALIDATED BUT THE CERTIFICATE REVOCATION STATUS WILL NOT BE CHECKED.
[2024-10-11, 04:43:53 UTC] [ssl_wrap_socket.py:181] INFO - THIS CONNECTION IS IN INSECURE MODE. IT MEANS THE CERTIFICATE WILL BE VALIDATED BUT THE CERTIFICATE REVOCATION STATUS WILL NOT BE CHECKED.
[2024-10-11, 04:43:54 UTC] [cursor.py:1156] INFO - Number of results in first chunk: 1
[2024-10-11, 04:44:20 UTC] [logging_util.py:188] INFO - Target table create 'dev.stock_merck_stock_price_full', Data loaded successfully in both the tables using full load
[2024-10-11, 04:44:20 UTC] [python.py:237] INFO - Done. Returned value was: None
[2024-10-11, 04:44:20 UTC] [taskinstance.py:441] ▶ Post task execution logs
  
```

DAG: Pipeline_Merck_Stock_Price1 Schedule: @daily Next Run ID: 2024-10-11, 00:00:00 UTC

10/11/2024 07:57:12 PM All Run Types All Run States Clear Filters Auto-refresh 25

Press **SHIFT** + **/** for Shortcuts

Task: Pipeline_Merck_Stock_Price1 / 2024-10-11, 00:00:00 UTC / return_last_90d_price

Clear task Mark state as... Filter DAG by task

Duration: 00:00:54, 00:00:27, 00:00:00

return_last_90d_price create_load_incremental create_load_full

Logs

default-hostname

```

*** Reading remote logs from Cloud Logging.
[2024-10-11, 00:00:08 UTC] [local_task_job_runner.py:120] ▶ Pre task execution logs
[2024-10-11, 00:00:10 UTC] [python.py:237] INFO - Done. Returned value was: [{"1. open": "110.7000", "2. high": "110.8300", "3. low": "109.1700", "4. close": "109.4000", "5. volume": "5906291", "date": "2024-09-05", "symbol": "MRK"}], [{"1. open": "116.6600", "2. high": "116.7300", "3. low": "114.4700", "4. close": "115.8000", "5. volume": "35509393", "date": "2024-07-30", "symbol": "MRK"}], [{"1. open": "125.7750", "2. high": "127.9900", "3. low": "124.9200", "4. close": "127.7800", "5. volume": "12009664", "date": "2024-09-05", "symbol": "MRK"}]]
[2024-10-11, 00:00:10 UTC] [taskinstance.py:441] ▶ Post task execution logs
  
```

Stock Data Load

Snowflake Screenshot:-

Full Load

The screenshot shows the Snowflake interface with a query editor and results table. The query is: `select * from DEV.STOCK.MERCK_STOCK_PRICE_FULL;`

The results table displays 11 rows of stock price data for MRK. The columns are: DATE, OPEN, HIGH, LOW, CLOSE, VOLUME, and SYMBOL.

	DATE	OPEN	HIGH	LOW	CLOSE	VOLUME	SYMBOL
1	2024-10-10	110.70	110.83	109.17	109.40	5986251	MRK
2	2024-10-09	108.41	110.30	108.20	110.27	8627785	MRK
3	2024-10-08	107.97	108.75	107.81	108.52	8832415	MRK
4	2024-10-07	109.70	110.98	108.18	108.59	8654772	MRK
5	2024-10-04	109.49	110.30	109.39	109.77	9721325	MRK
6	2024-10-03	111.82	112.08	109.83	110.18	10280553	MRK
7	2024-10-02	114.55	114.60	111.98	112.08	9041754	MRK
8	2024-10-01	113.96	114.79	113.11	114.74	7715526	MRK
9	2024-09-30	113.61	114.01	112.70	113.56	10949982	MRK
10	2024-09-27	113.17	114.34	113.00	113.69	11920112	MRK
11	2024-09-26	113.88	114.00	112.93	113.09	11132094	MRK

Incremental Load

The screenshot shows the Snowflake interface with a query editor and results table. The query is: `select * from DEV.STOCK.MERCK_STOCK_PRICE_INCREMENTAL;`

The results table displays 11 rows of stock price data for MRK. The columns are: DATE, OPEN, HIGH, LOW, CLOSE, VOLUME, and SYMBOL.

	DATE	OPEN	HIGH	LOW	CLOSE	VOLUME	SYMBOL
1	2024-10-10	110.70	110.83	109.17	109.40	5986251	MRK
2	2024-10-09	108.41	110.30	108.20	110.27	8627785	MRK
3	2024-10-08	107.97	108.75	107.81	108.52	8832415	MRK
4	2024-10-07	109.70	110.98	108.18	108.59	8654772	MRK
5	2024-10-04	109.49	110.30	109.39	109.77	9721325	MRK
6	2024-10-03	111.82	112.08	109.83	110.18	10280553	MRK
7	2024-10-02	114.55	114.60	111.98	112.08	9041754	MRK
8	2024-10-01	113.96	114.79	113.11	114.74	7715526	MRK
9	2024-09-30	113.61	114.01	112.70	113.56	10949982	MRK
10	2024-09-27	113.17	114.34	113.00	113.69	11920112	MRK
11	2024-09-26	113.88	114.00	112.93	113.09	11132094	MRK