

Airflow DAG for Full refresh on stock price using Alpha Vantage API

- Set up required variables:

Q Search Keys

Import Variables

<input type="checkbox"/>	Key	Value
<input type="checkbox"/>	snowflake_userid	POODLE
<input type="checkbox"/>	snowflake_password	***
<input type="checkbox"/>	snowflake_account	SFEDU
<input type="checkbox"/>	alpha_vantage_api	7JUC

- Set up Snowflake connection:

Add Connection

Connection ID *

snowflake_conn

Connection Type *

Snowflake

Connection type missing? Make sure you have installed the corresponding Airflow Providers Package.

Standard Fields

Description

login

POODLE

password

.....

schema

snowflake schema

Extra Fields

Extra Fields JSON

Save

Extra Fields JSON

```
1 {
2   "account": "SFEDU02-LVB17920",
3   "warehouse": "POODLE_QUERY_WH",
4   "database": "USER_DB_POODLE",
5   "region": null,
6   "role": null,
7   "private_key_file": null,
8   "private_key_content": null,
9   "insecure_mode": false
10 }
```

Standard Fields

Extra Fields

Account

SFEDU02-LVB17920

Warehouse

POODLE_QUERY_WH

Database

USER_DB_POODLE

Region

Role

Private key (Path)

Private key (Text)

Insecure mode

☐ Turns off OCSP certificate checks

- Airflow Homepage showing my DAG “stock_price_full_refresh_v2”:

Sample DAG with Display Name example

ScheduleLatest RunNext Run

stock_price_full_refresh_v2 snowflake, alphavantage, full-refresh, ETL

ScheduleLatest RunNext Run

30 2 ***2025-10-04, 19:30:00

tutorial example

ScheduleLatest RunNext Run

1 day, 0:00:002025-10-04, 13:18:40

tutorial_dag example

ScheduleLatest RunNext Run

- Log screen of my DAG (Load Operation):

Dag stock_price_full_refresh_v2 Dag Run 2025-10-04, 13:14:29 Task load

Options

extracttransformload

load 2025-10-04, 13:14:38 success

Add a noteClear Task InstanceMark Task Instance as...

OperatorPythonDecoratedOperatorStart2025-10-04, 13:14:38End2025-10-04, 13:14:44Duration6.05sDAG Versionv1

LogsRendered TemplatesXComAudit LogsCodeDetailsAsset Events

All Log LevelsAll SourcesWrap

Log message source details: sources=[/opt/airflow/logs/dag_id=stock_price_full_refresh_v2/run_id=manual_2025-10-04T20:14:32.326892+00:00/task_id=load]

2 [2025-10-04, 13:14:38] INFO - DAG bundles loaded: dags-folder, example_dags: source="airflow.dag_processing.bundles.manager.DagBundlesManager"

3 [2025-10-04, 13:14:38] INFO - Filling up the DagBag from /opt/airflow/dags/stock_price_dag.py: source="airflow.models.dagbag.DagBag"

4 [2025-10-04, 13:14:38] INFO - Task instance is in running state: chan="stdout": source="task"

5 [2025-10-04, 13:14:38] INFO - Previous state of the Task Instance: TaskInstanceState.QUEUED: chan="stdout": source="task"

6 [2025-10-04, 13:14:38] INFO - Current task name:load: chan="stdout": source="task"

7 [2025-10-04, 13:14:38] INFO - Dag name:stock_price_full_refresh_v2: chan="stdout": source="task"

8 [2025-10-04, 13:14:38] WARNING - /home/airflow/.local/lib/python3.12/site-packages/airflow/models/connection.py:471: DeprecationWarning: Using warnings.warn() source="py.warnings"

9 [2025-10-04, 13:14:38] INFO - Connection Retrieved 'snowflake_conn': source="airflow.hooks.base"

10 [2025-10-04, 13:14:38] INFO - Snowflake Connector for Python Version: 3.17.2, Python Version: 3.12.11, Platform: Linux-6.10.14-linuxkit-aarch64: source="snowflake.connector.connection"

11 [2025-10-04, 13:14:38] INFO - Connecting to GLOBAL Snowflake domain: source="snowflake.connector.connection"

12 [2025-10-04, 13:14:44] INFO - Full refresh committed. Row count in target: 90: chan="stdout": source="task"

13 [2025-10-04, 13:14:44] INFO - Done. Returned value was: None: source="airflow.task.decorators.python_decorator"

14 [2025-10-04, 13:14:44] INFO - Task instance in success state: chan="stdout": source="task"

15 [2025-10-04, 13:14:44] INFO - Previous state of the Task Instance: TaskInstanceState.RUNNING: chan="stdout": source="task"

16 [2025-10-04, 13:14:44] INFO - Task operator:Task(PythonDecoratedOperator): load: chan="stdout": source="task"

- Code Snippets:

```
12 SYMBOL = "AAPL"
13 TARGET_TABLE = "USER_DB_POODLE.RAW.STOCK_PRICE_DAG"
14
15 VANTAGE_API_KEY = Variable.get("alpha_vantage_api")
16
17 def return_snowflake_conn():
18     hook = SnowflakeHook(snowflake_conn_id = "snowflake_conn")
19     conn = hook.get_conn()
20     return conn.cursor()
21
```

Establishing snowflake connection and accessing vantage api key

```
23 @task
24 def extract(symbol: str) -> str:
25     url = (
26         "https://www.alphavantage.co/query"
27         f"?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={VANTAGE_API_KEY}"
28     )
29     resp = requests.get(url, timeout = 60)
30     resp.raise_for_status()
31     return resp.text
32
```

Extract task

```
33 @task
34 def transform(text: str):
35     """
36     Parse JSON and produce list of tuples:
37     (OPEN, HIGH, LOW, CLOSE, TRADE_VOLUME, TRADE_DATE, SYMBOL)
38     Limited to the last 90 calendar days, newest first. Casts numeric types.
39     """
40     payload = json.loads(text)
41     if "Time Series (Daily)" not in payload:
42         raise RuntimeError(f"Alpha Vantage response missing daily series: {payload}")
43
44     series = payload["Time Series (Daily)"]
45     sorted_dates = sorted(series.keys(), reverse=True)[:90]
46
47     rows = []
48     for d in sorted_dates:
49         v = series[d]
50         rows.append([
51             float(v["1. open"]),
52             float(v["2. high"]),
53             float(v["3. low"]),
54             float(v["4. close"]),
55             int(v["5. volume"]),
56             d,
57             SYMBOL,
58         ])
59     return rows
```

Transform Task

```

62 @task
63 def load(records):
64     """
65     Full refresh using a SQL transaction:
66     1) Ensure table exists
67     2) BEGIN
68     3) TRUNCATE TABLE
69     4) Bulk INSERT
70     5) COMMIT (or ROLLBACK on error)
71     """
72     if not records:
73         raise ValueError("No rows to load; aborting to avoid truncating to empty table.")
74
75     cur = return_snowflake_conn()
76     conn = cur.connection
77     try:
78         cur.execute(f"""
79             CREATE TABLE IF NOT EXISTS {TARGET_TABLE} (
80                 OPEN NUMBER,
81                 HIGH NUMBER,
82                 LOW NUMBER,
83                 CLOSE NUMBER,
84                 TRADE_VOLUME NUMBER,
85                 TRADE_DATE DATE,
86                 SYMBOL VARCHAR,
87                 PRIMARY KEY (TRADE_DATE, SYMBOL)
88             )
89         """)

```

```

94
95     # TRUNCATE for a true full refresh
96     cur.execute(f"TRUNCATE TABLE {TARGET_TABLE}")
97
98     # Bulk insert
99     insert_sql = f"""
100         INSERT INTO {TARGET_TABLE}
101         (OPEN, HIGH, LOW, CLOSE, TRADE_VOLUME, TRADE_DATE, SYMBOL)
102         VALUES (%s, %s, %s, %s, %s, %s, %s)
103     """
104     cur.executemany(insert_sql, records)
105
106     cur.execute("COMMIT")
107
108     cur.execute(f"SELECT COUNT(*) FROM {TARGET_TABLE}")
109     count = cur.fetchone()[0]
110     print(f"Full refresh committed. Row count in target: {count}")
111
112 except Exception as e:
113     try:
114         cur.execute("ROLLBACK")
115     except Exception:
116         pass
117     print("Error during full refresh:", e)
118     raise
119 finally:
120     try:
121         cur.close()

```

Load Task (Full Refresh using SQL transactions)

```

126 with DAG(
127     dag_id='stock_price_full_refresh_v2',
128     start_date=datetime(2025, 10, 4),
129     catchup=False,
130     tags=['ETL', 'alphavantage', 'snowflake', 'full-refresh'],
131     schedule='30 2 * * *',
132     default_args={
133         "owner": "data-eng",
134         "retries": 2,
135         "retry_delay": timedelta(minutes=5),
136     },
137     description="Fetch last 90d AAPL prices and full-refresh load into Snowflake using a SQL transaction",
138 ) as dag:
139
140     data = extract(SYMBOL)
141     rows = transform(data)
142     load(rows)
143

```

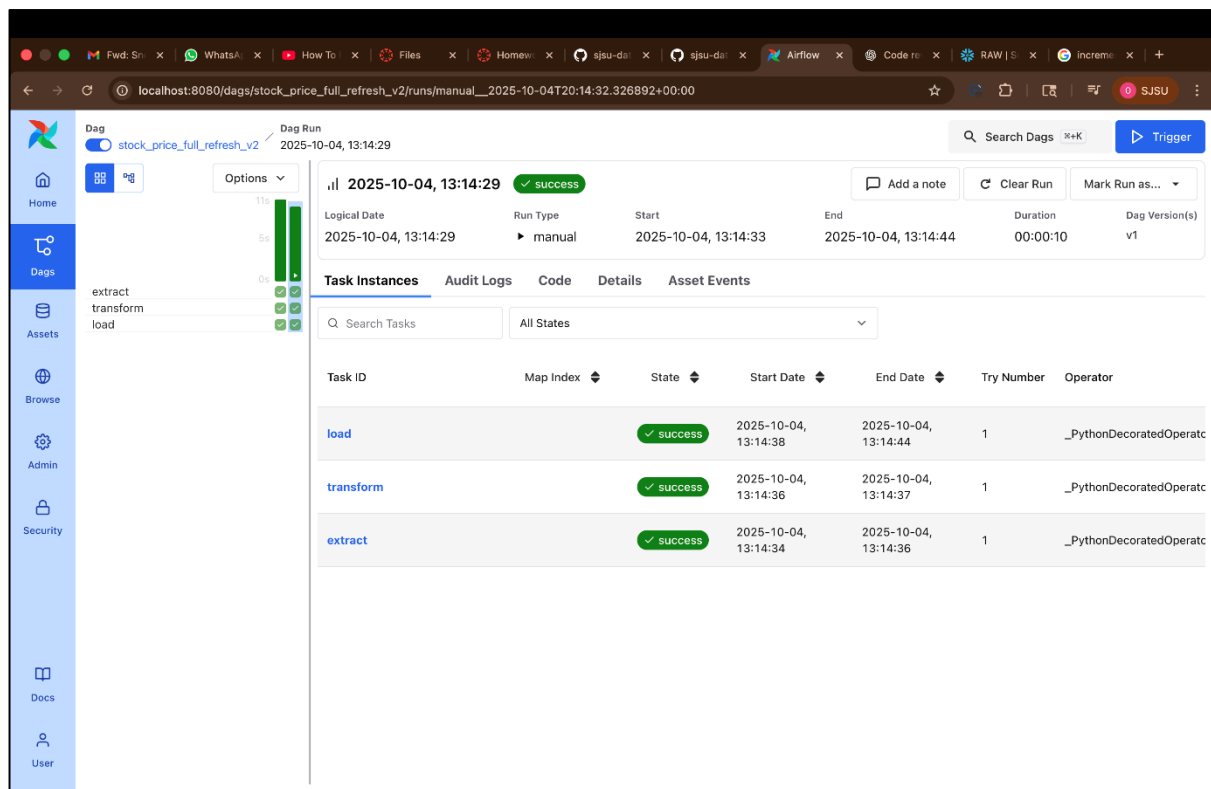
Creating “stock_price_full_refresh_v2” DAG

- Extra Screenshots:

The screenshot shows the Snowflake Database Explorer interface. The left sidebar contains navigation options like 'Work with data', 'Projects', 'Ingestion', 'Transformation', 'AI & ML', 'Monitoring', 'Marketplace', 'Horizon Catalog', 'Catalog', 'Data sharing', 'Governance & security', 'Manage', 'Compute', and 'Admin'. The main panel displays the 'Database Explorer' for 'USER_DB_POODLE / RAW / STOCK_PRICE_DAG'. The 'Data Preview' tab is active, showing a table with 180 rows. The table columns are: OPEN, HIGH, LOW, CLOSE, TRADE_VOLUME, TRADE_DATE, and SYMBOL. The data represents AAPL stock prices from 2025-09-09 to 2025-10-03.

	OPEN	HIGH	LOW	CLOSE	TRADE_VOLUME	TRADE_DATE	SYMBOL
1	255	259	254	258	49155614	2025-10-03	AAPL
2	257	258	254	257	42630239	2025-10-02	AAPL
3	255	259	255	255	48713940	2025-10-01	AAPL
4	255	256	253	255	37704259	2025-09-30	AAPL
5	255	255	253	254	40127687	2025-09-29	AAPL
6	254	258	254	255	46076258	2025-09-26	AAPL
7	253	257	252	257	55202075	2025-09-25	AAPL
8	255	256	251	252	42303710	2025-09-24	AAPL
9	256	257	254	254	60275187	2025-09-23	AAPL
10	248	257	248	256	105517416	2025-09-22	AAPL
11	241	246	240	246	163741314	2025-09-19	AAPL
12	240	241	237	238	44249576	2025-09-18	AAPL
13	239	240	238	239	46508017	2025-09-17	AAPL
14	237	241	236	238	63421099	2025-09-16	AAPL
15	237	238	235	237	42699524	2025-09-15	AAPL
16	229	235	229	234	55824216	2025-09-12	AAPL
17	227	230	227	230	50208578	2025-09-11	AAPL
18	232	232	226	227	83440810	2025-09-10	AAPL
19	237	239	233	234	66313918	2025-09-09	AAPL

Snowflake UI showing the output table “STOCK_PRICE_DAG” after successful execution of airflow DAG



Airflow UI showing successful execution of my "stock_price_full_refresh_v2" DAG

- Omkar Rajale