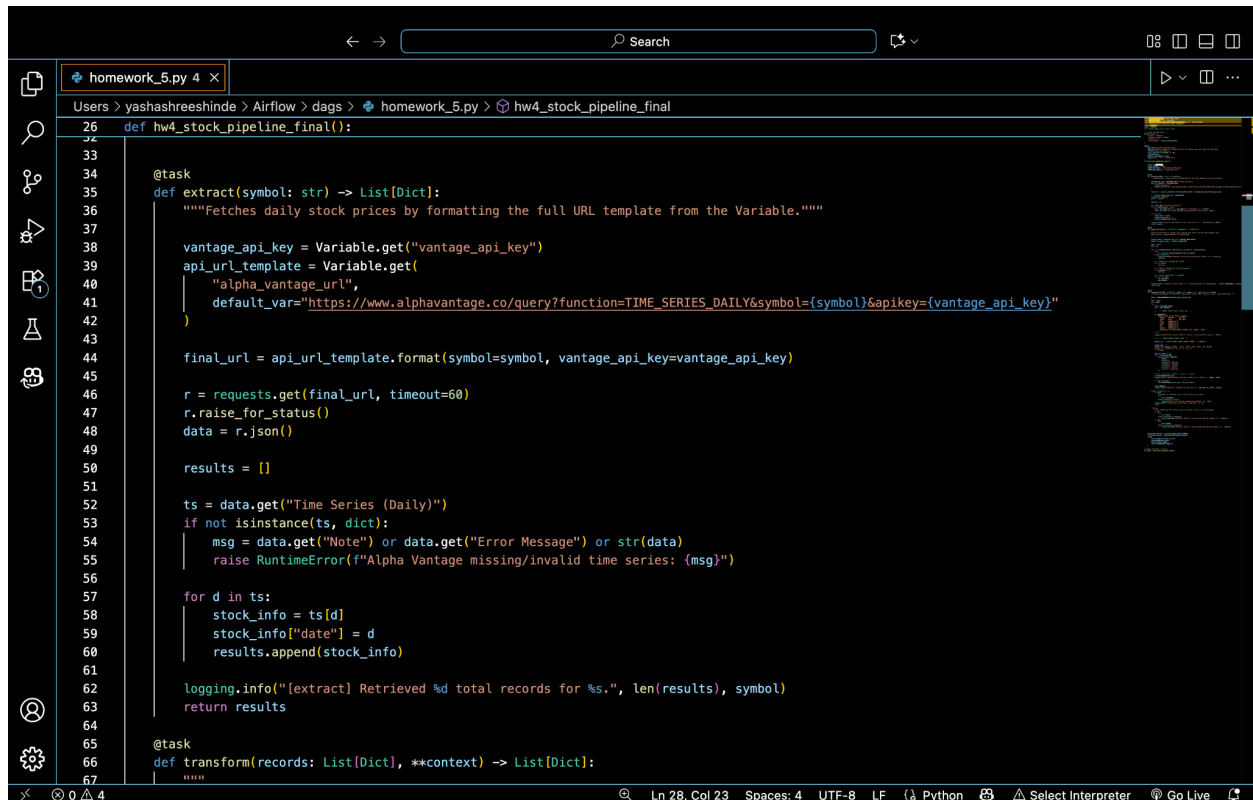


Homework 5

Name - Yashashree Shinde

Porting homework #4 to Airflow

1) Create tasks using @task decorator TASK-1



The screenshot shows a code editor with a file named 'homework_5.py'. The code defines a task decorator and a task function. The task decorator, @task, is applied to the function 'extract'. The 'extract' function fetches daily stock prices from Alpha Vantage. It uses 'Variable.get' to retrieve 'vantage_api_key' and 'alpha_vantage_url'. It constructs a 'final_url' using 'api_url_template.format'. It then uses 'requests.get' to fetch the data, raises an exception for non-200 status codes, and parses the JSON response. The data is then processed into a list of dictionaries, 'results', and a log message is generated. Finally, the 'results' are returned. The task decorator also defines a 'transform' function that takes a list of records and returns them as is.

```
26 def hw4_stock_pipeline_final():
27
28
29
30
31
32
33
34 @task
35 def extract(symbol: str) -> List[Dict]:
36     """Fetches daily stock prices by formatting the full URL template from the Variable."""
37
38     vantage_api_key = Variable.get("vantage_api_key")
39     api_url_template = Variable.get(
40         "alpha_vantage_url",
41         default_var="https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={vantage_api_key}"
42     )
43
44     final_url = api_url_template.format(symbol=symbol, vantage_api_key=vantage_api_key)
45
46     r = requests.get(final_url, timeout=60)
47     r.raise_for_status()
48     data = r.json()
49
50     results = []
51
52     ts = data.get("Time Series (Daily)")
53     if not isinstance(ts, dict):
54         msg = data.get("Note") or data.get("Error Message") or str(data)
55         raise RuntimeError(f"Alpha Vantage missing/invalid time series: {msg}")
56
57     for d in ts:
58         stock_info = ts[d]
59         stock_info["date"] = d
60         results.append(stock_info)
61
62     logging.info("[extract] Retrieved %d total records for %s.", len(results), symbol)
63     return results
64
65 @task
66 def transform(records: List[Dict], **context) -> List[Dict]:
67     """
```

CODE

```
def extract(symbol: str) -> List[Dict]:
    """Fetches daily stock prices by formatting the full URL template from the Variable."""

    vantage_api_key = Variable.get("vantage_api_key")
    api_url_template = Variable.get(
        "alpha_vantage_url",
        default_var="https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={vantage_api_key}"
    )

    final_url = api_url_template.format(symbol=symbol, vantage_api_key=vantage_api_key)
```

```

r = requests.get(final_url, timeout=60)
r.raise_for_status()
data = r.json()

results = []
ts = data.get("Time Series (Daily)")
if not isinstance(ts, dict):
    msg = data.get("Note") or data.get("Error Message") or str(data)
    raise RuntimeError(f"Alpha Vantage missing/invalid time series: {msg}")

for d in ts:
    stock_info = ts[d]
    stock_info["date"] = d
    results.append(stock_info)
logging.info("[extract] Retrieved %d total records for %s.", len(results), symbol)
return results

```

TASK-2

```

26 def hw4_stock_pipeline_final():
63     return results
64
65 @task
66 def transform(records: List[Dict], **context) -> List[Dict]:
67
68     logical_date = context['dag_run'].logical_date.date()
69     cutoff = logical_date - timedelta(days=90)
70
71     seen = set()
72     out = []
73
74     for r in sorted(records, key=lambda x: x["date"], reverse=True):
75         try:
76             d = datetime.fromisoformat(r["date"]).date()
77         except ValueError:
78             logging.warning("Skipping record with invalid date format: %s", r["date"])
79             continue
80
81         # 1. ENFORCE 90 CALENDAR DAY CUTOFF
82         if d < cutoff:
83             continue
84
85         # 2. ENFORCE TRADING DAY FILTER (Weekends)
86         if d.weekday() >= 5:
87             continue
88
89         # 3. Prevent duplicates (if needed)
90         if d not in seen:
91             out.append(r)
92             seen.add(d)
93
94     logging.info("[transform] Cutoff Date: %s. Filtered and kept %d trading days.", cutoff.isoformat(), len(out))
95     return out
96
97

```

CODE

```

@task
def transform(records: List[Dict], **context) -> List[Dict]:
    logical_date = context['dag_run'].logical_date.date()

```

```

cutoff = logical_date - timedelta(days=90)

seen = set()
out = []

for r in sorted(records, key=lambda x: x["date"], reverse=True):
    try:
        d = datetime.fromisoformat(r["date"]).date()
    except ValueError:
        logging.warning("Skipping record with invalid date format: %s", r["date"])
        continue

    # 1. ENFORCE 90 CALENDAR DAY CUTOFF
    if d < cutoff:
        continue

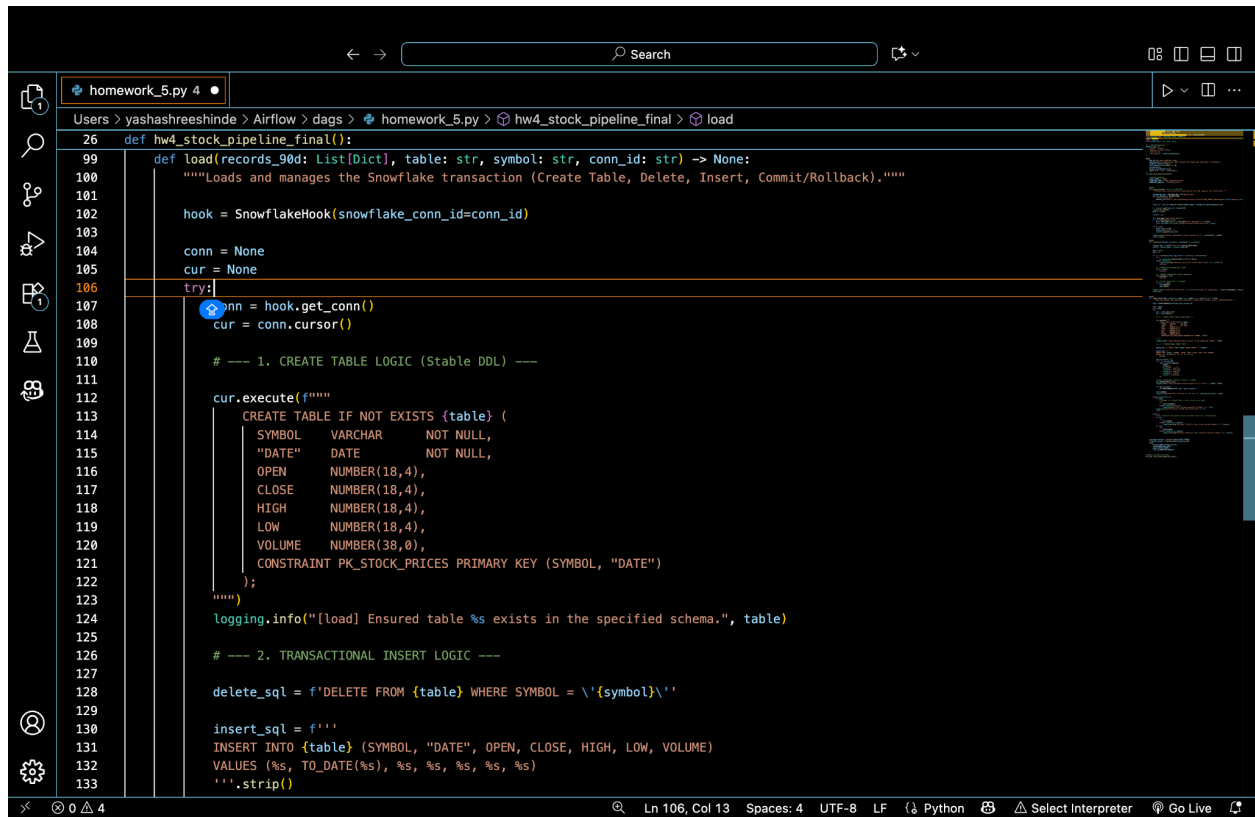
    # 2. ENFORCE TRADING DAY FILTER (Weekends)
    if d.weekday() >= 5:
        continue

    # 3. Prevent duplicates (if needed)
    if d not in seen:
        out.append(r)
        seen.add(d)

logging.info("[transform] Cutoff Date: %s. Filtered and kept %d trading days.",
cutoff.isoformat(), len(out))
return out

```

TASK-3



```
26 def hw4_stock_pipeline_final():
99     def load(records_90d: List[Dict], table: str, symbol: str, conn_id: str) -> None:
100         """Loads and manages the Snowflake transaction (Create Table, Delete, Insert, Commit/Rollback)."""
101
102         hook = SnowflakeHook(snowflake_conn_id=conn_id)
103
104         conn = None
105         cur = None
106         try:
107             conn = hook.get_conn()
108             cur = conn.cursor()
109
110             # --- 1. CREATE TABLE LOGIC (Stable DDL) ---
111
112             cur.execute(f"""
113                 CREATE TABLE IF NOT EXISTS {table} (
114                     SYMBOL    VARCHAR    NOT NULL,
115                     "DATE"    DATE       NOT NULL,
116                     OPEN      NUMBER(18,4),
117                     CLOSE     NUMBER(18,4),
118                     HIGH      NUMBER(18,4),
119                     LOW        NUMBER(18,4),
120                     VOLUME    NUMBER(38,0),
121                     CONSTRAINT PK_STOCK_PRICES PRIMARY KEY (SYMBOL, "DATE")
122                 );
123             """)
124             logging.info("[load] Ensured table %s exists in the specified schema.", table)
125
126             # --- 2. TRANSACTIONAL INSERT LOGIC ---
127
128             delete_sql = f'DELETE FROM {table} WHERE SYMBOL = \'{symbol}\''
129
130             insert_sql = f'''
131             INSERT INTO {table} (SYMBOL, "DATE", OPEN, CLOSE, HIGH, LOW, VOLUME)
132             VALUES (%s, TO_DATE(%s), %s, %s, %s, %s, %s)
133             '''
134             cur.execute(delete_sql)
135             cur.execute(insert_sql)
136             conn.commit()
137         except Exception as e:
138             logging.error(f"Error in load function: {e}")
139             if conn:
140                 conn.rollback()
141         finally:
142             if conn:
143                 conn.close()
144             cur.close()
145
146     load(records_90d, table, symbol, conn_id)
```

CODE

@task

```
def load(records_90d: List[Dict], table: str, symbol: str, conn_id: str) -> None:
    """Loads and manages the Snowflake transaction (Create Table, Delete, Insert,
    Commit/Rollback)."""
```

```
    hook = SnowflakeHook(snowflake_conn_id=conn_id)
```

```
    conn = None
```

```
    cur = None
```

```
    try:
```

```
        conn = hook.get_conn()
```

```
        cur = conn.cursor()
```

```
    # --- 1. CREATE TABLE LOGIC (Stable DDL) ---
```

```
    cur.execute(f"""
```

```
        CREATE TABLE IF NOT EXISTS {table} (
```

```
            SYMBOL    VARCHAR    NOT NULL,
```

```
            "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_STOCK_PRICES PRIMARY KEY (SYMBOL, "DATE")
    );
    """
logging.info("[load] Ensured table %s exists in the specified schema.", table)

# --- 2. TRANSACTIONAL INSERT LOGIC ---

delete_sql = f'DELETE FROM {table} WHERE SYMBOL = \"{symbol}\"'

insert_sql = f'''
INSERT INTO {table} (SYMBOL, "DATE", OPEN, CLOSE, HIGH, LOW, VOLUME)
VALUES (%s, TO_DATE(%s), %s, %s, %s, %s, %s)
'''
insert_sql.strip()
rows_to_insert = []
for r in records_90d:
    rows_to_insert.append((
        symbol,
        r["date"],
        float(r["1. open"]),
        float(r["4. close"]),
        float(r["2. high"]),
        float(r["3. low"]),
        int(r["5. volume"]),
    ))

# Start transaction: Delete -> Insert -> Commit
cur.execute(delete_sql)
logging.info("[load] Deleted existing records for %s from %s.", symbol, table)
if rows_to_insert:
    cur.executemany(insert_sql, rows_to_insert)

conn.commit()
logging.info("Committed. Inserted %d rows into %s", len(rows_to_insert), table)

except Exception as e:
    if conn:
        # Attempt to rollback (this is the correct error path)
        try:
            conn.rollback()
        except Exception as rb_e:
            logging.error("Error during connection rollback: %s", rb_e)
        logging.error("Transaction failed and rolled back: %s", e)
        raise

finally:
    if cur:
        try:

```

```

        cur.close()
    except Exception as close_e:
        logging.warning("Warning: Failed to close cursor during cleanup: %s", close_e)
if conn:
    try:
        conn.close()
    except Exception as close_e:
        logging.warning("Warning: Failed to close connection during cleanup: %s", close_e)

```

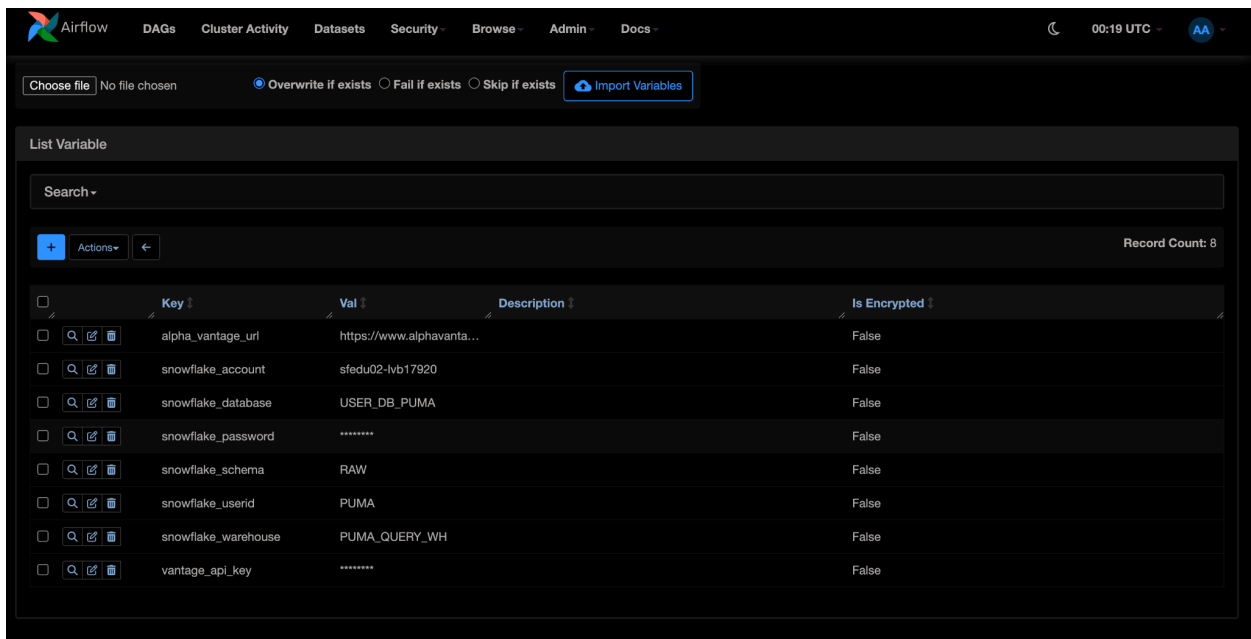
```

extracted_records = extract(symbol=STOCK_SYMBOL)
filtered_records = transform(extracted_records)
load(
    records_90d=filtered_records,
    table=SNOWFLAKE_TABLE,
    symbol=STOCK_SYMBOL,
    conn_id=SNOWFLAKE_CONN_ID
)

```

2) Set up a variable for Alpha Vantage API key

- Use the variable in your code (Variable.get)



The screenshot shows the Airflow web interface. At the top, there's a navigation bar with links: Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. Below the navigation bar, there's a section for 'List Variable'. It includes a search bar and a table of variables. The table has columns: Key, Val, Description, and Is Encrypted. There are 8 records listed.

Key	Val	Description	Is Encrypted
alpha_vantage_url	https://www.alphavanta...		False
snowflake_account	sfedu02-lvb17920		False
snowflake_database	USER_DB_PUMA		False
snowflake_password	*****		False
snowflake_schema	RAW		False
snowflake_userid	PUMA		False
snowflake_warehouse	PUMA_QUERY_WH		False
vantage_api_key	*****		False

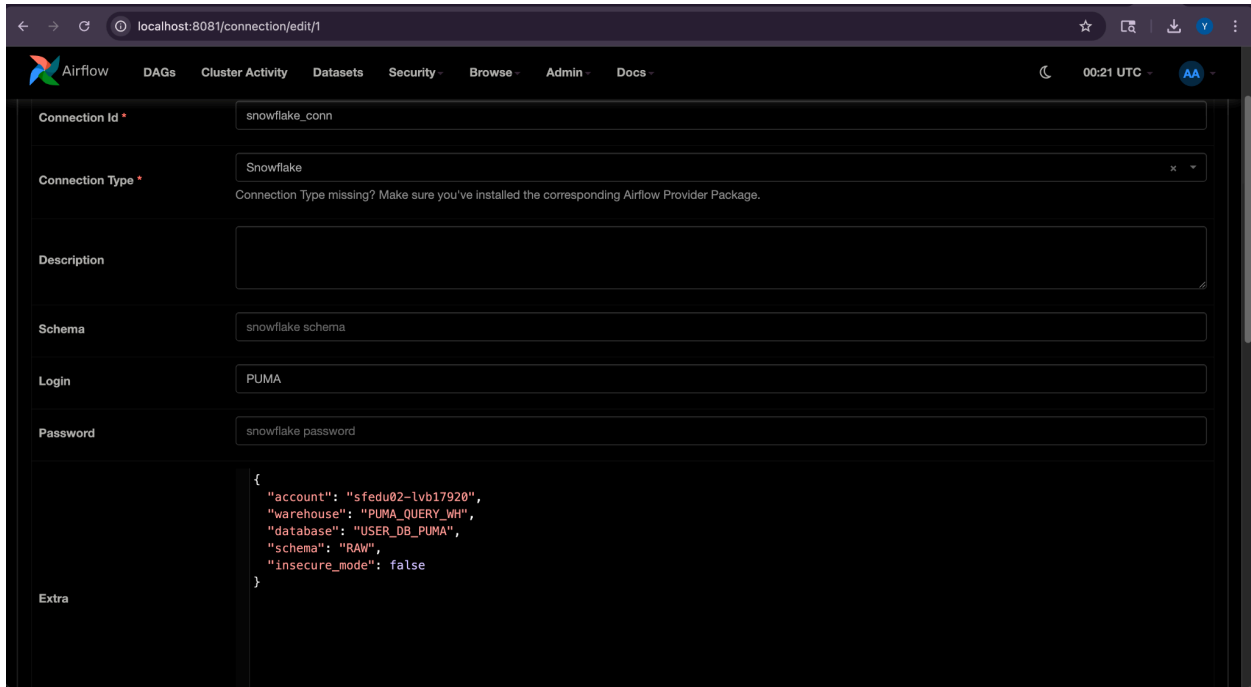
```

vantage_api_key = Variable.get("vantage_api_key")
api_url_template = Variable.get(
    "alpha_vantage_url",
    default_var="https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={symbol}&apikey={vantage_api_key}"
)

```

3) Set up Snowflake Connection

- Use the connection in your code
- Capture the Connection detail page screenshot



The screenshot shows the Airflow web interface at localhost:8081/connection/edit/1. The connection is named 'snowflake_conn' and is of type 'Snowflake'. The description field is empty. The schema is 'snowflake.schema', the login is 'PUMA', and the password is 'snowflake password'. The 'Extra' field contains a JSON configuration for the Snowflake connection.

```

{
  "account": "sfedu02-lvb17920",
  "warehouse": "PUMA_QUERY_WH",
  "database": "USER_DB_PUMA",
  "schema": "RAW",
  "insecure_mode": false
}

```

```

@dag(
    dag_id="hw4_stock_pipeline_final",
    description="ETL pipeline for HW4: Extracts 90 trading days and loads to Snowflake.",
    schedule_interval="@daily",
    start_date=datetime(2025, 5, 30),
    catchup=False,
    default_args=default_args,
    tags=["hw4", "etl", "snowflake"],
)
def hw4_stock_pipeline_final():

    # Define constants
    STOCK_SYMBOL = "AMZN"
    SNOWFLAKE_TABLE = "RAW.STOCK_PRICES_DAG"
    SNOWFLAKE_CONN_ID = "snowflake_conn"

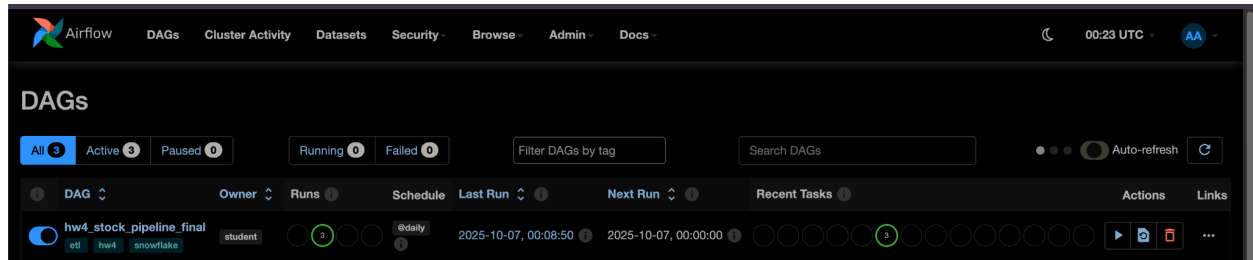
```

4) Ensure the overall DAG is implemented properly and runs successfully

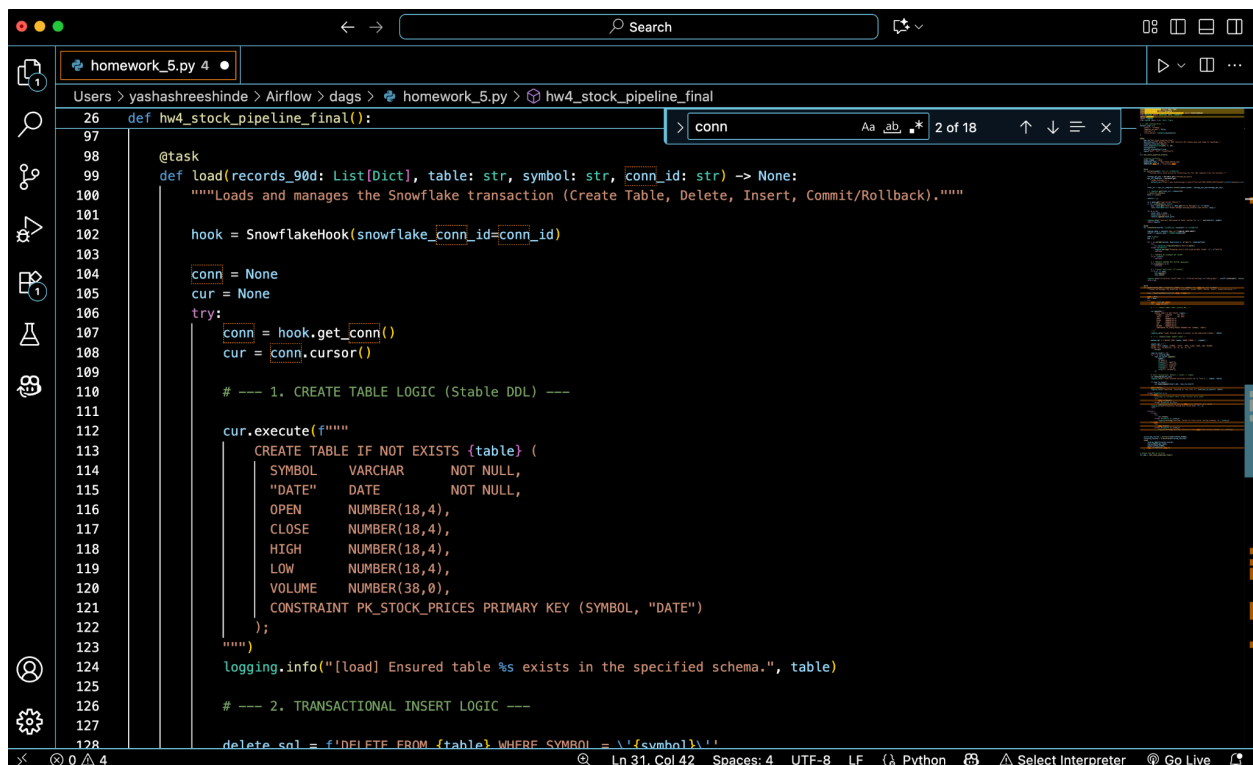
- A github link with the entire code needs to be submitted
- Implement the same full refresh using SQL transaction

GITHUB LINK -

<https://github.com/yashashree5/DATA-226-Data-Warehouse/blob/main/Homeworks/Homework5/>



Sql transactions




```
homework_5.py 4
Users > yashashreeshinde > Airflow > dags > homework_5.py > hw4_stock_pipeline_final
26 def hw4_stock_pipeline_final():
99 def load(records_90d: List[Dict], table: str, symbol: str, conn:
126 # --- 2. TRANSACTIONAL INSERT LOGIC ---
127
128 delete_sql = f'DELETE FROM {table} WHERE SYMBOL = \"{symbol}\"'
129
130 insert_sql = f'''
131 INSERT INTO {table} (SYMBOL, "DATE", OPEN, CLOSE, HIGH, LOW, VOLUME)
132 VALUES (%s, TO_DATE(%s), %s, %s, %s, %s, %s)
133 '''
134
135 rows_to_insert = []
136 for r in records_90d:
137     rows_to_insert.append((
138         symbol,
139         r["date"],
140         float(r["1. open"]),
141         float(r["4. close"]),
142         float(r["2. high"]),
143         float(r["3. low"]),
144         int(r["5. volume"]),
145     ))
146
147 # Start transaction: Delete -> Insert -> Commit
148 cur.execute(delete_sql)
149 logging.info("[load] Deleted existing records for %s from %s.", symbol, table)
150
151 if rows_to_insert:
152     cur.executemany(insert_sql, rows_to_insert)
153
154 conn.commit()
155 logging.info("Committed. Inserted %d rows into %s", len(rows_to_insert), table)
156
Ln 31, Col 42 Spaces: 4 UTF-8 LF Python Select Interpreter Go Live
```

```
homework_5.py 4
Users > yashashreeshinde > Airflow > dags > homework_5.py > hw4_stock_pipeline_final > load
26 def hw4_stock_pipeline_final():
99 def load(records_90d: List[Dict], table: str, symbol: str, conn_id: str) -> conn
152     cur.executemany(insert_sql, rows_to_insert)
153
154     conn.commit()
155     logging.info("Committed. Inserted %d rows into %s", len(rows_to_insert), table)
156
157 except Exception as e:
158     if conn:
159         # Attempt to rollback (this is the correct error path)
160         try:
161             conn.rollback()
162         except Exception as rb_e:
163             logging.error("Error during connection rollback: %s", rb_e)
164             logging.error("Transaction failed and rolled back: %s", e)
165             raise
166
167 finally:
168     if cur:
169         try:
170             cur.close()
171         except Exception as close_e:
172             logging.warning("Warning: Failed to close cursor during cleanup: %s", close_e)
173     if conn:
174         try:
175             conn.close()
176         except Exception as close_e:
177             logging.warning("Warning: Failed to close connection during cleanup: %s", close_e)
178
179
180 extracted_records = extract(symbol=STOCK_SYMBOL)
181 filtered_records = transform(extracted_records)
182 load(
183     records_90d=filtered_records,
184     table=SNOWFLAKE_TABLE,
185
Ln 144, Col 41 Spaces: 4 UTF-8 LF Python Select Interpreter Go Live
```

CODE

```
def load(records_90d: List[Dict], table: str, symbol: str, conn_id: str) -> None:
    """Loads and manages the Snowflake transaction (Create Table, Delete, Insert,
    Commit/Rollback)."""
```

```
    hook = SnowflakeHook(snowflake_conn_id=conn_id)
```

```
    conn = None
```

```
    cur = None
```

```
    try:
```

```
        conn = hook.get_conn()
```

```
        cur = conn.cursor()
```

```
    # --- 1. CREATE TABLE LOGIC (Stable DDL) ---
```

```
    cur.execute(f"""
```

```
        CREATE TABLE IF NOT EXISTS {table} (
```

```
            SYMBOL VARCHAR NOT NULL,
```

```
            "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_STOCK_PRICES PRIMARY KEY (SYMBOL, "DATE")
```

```
        );
```

```
    """)
```

```
    logging.info("[load] Ensured table %s exists in the specified schema.", table)
```

```
    # --- 2. TRANSACTIONAL INSERT LOGIC ---
```

```
    delete_sql = f'DELETE FROM {table} WHERE SYMBOL = \"{symbol}\"'
```

```
    insert_sql = f'''
```

```
    INSERT INTO {table} (SYMBOL, "DATE", OPEN, CLOSE, HIGH, LOW, VOLUME)
```

```
    VALUES (%s, TO_DATE(%s), %s, %s, %s, %s, %s)
```

```
    """.strip()
```

```
    rows_to_insert = []
```

```
    for r in records_90d:
```

```
        rows_to_insert.append((
```

```
            symbol,
```

```
            r["date"],
```

```
            float(r["1. open"]),
```

```
            float(r["4. close"]),
```

```
            float(r["2. high"]),
```

```
            float(r["3. low"]),
```

```
            int(r["5. volume"]),
```

```
        ))
```

```
    # Start transaction: Delete -> Insert -> Commit
```

```

cur.execute(delete_sql)
logging.info("[load] Deleted existing records for %s from %s.", symbol, table)
if rows_to_insert:
    cur.executemany(insert_sql, rows_to_insert)

conn.commit()
logging.info("Committed. Inserted %d rows into %s", len(rows_to_insert), table)

except Exception as e:
    if conn:
        # Attempt to rollback (this is the correct error path)
        try:
            conn.rollback()
        except Exception as rb_e:
            logging.error("Error during connection rollback: %s", rb_e)
        logging.error("Transaction failed and rolled back: %s", e)
        raise

finally:
    if cur:
        try:
            cur.close()
        except Exception as close_e:
            logging.warning("Warning: Failed to close cursor during cleanup: %s", close_e)
    if conn:
        try:
            conn.close()
        except Exception as close_e:
            logging.warning("Warning: Failed to close connection during cleanup: %s", close_e)

```

5) Capture two screenshot of your Airflow Web UI (examples to follow)

- One with the Airflow homepage showing the DAG
- The other with the log screen of the DAG

Airflow DAGs overview page. The interface shows a list of DAGs with columns for DAG name, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, and Actions. The DAGs listed are:

- hw4_stock_pipeline_final** (Owner: student, Schedule: @daily, Last Run: 2025-10-07, 00:08:50, Next Run: 2025-10-07, 00:00:00, Recent Tasks: 3)
- yfinance_ETL** (Owner: airflow, Schedule: 0 2 * * *, Last Run: 2025-10-05, 02:00:00, Next Run: 2025-10-06, 02:00:00, Recent Tasks: 3)
- yfinance_train** (Owner: airflow, Schedule: 30 2 * * *, Last Run: 2025-10-06, 00:18:48, Next Run: 2025-10-06, 02:30:00, Recent Tasks: 2)

Showing 1-3 of 3 DAGs

Version: v2.10.1
Git Version: .release:854173176f372f6509800ed446286c32cb75045e

Airflow DAG: hw4_stock_pipeline_final ETL pipeline for HW4: Extracts 90 trading days and loads to Snowflake. Schedule: @daily, Next Run ID: 2025-10-07, 00:00:00 UTC.

07/10/2025 12:28:21 AM All Run Types All Run States Clear Filters Auto-refresh 25

Press **shift** + **/** for Shortcuts

DAG: hw4_stock_pipeline_final

Details Graph Gantt Code Event Log Run Duration Task Duration Calendar

Layout: Left -> Right

extract @task transform @task load @task

localhost:8081/dags/hw4_stock_pipeline_final/grid?tab=logs&dag_run_id>manual__2025-10-07T00%3A08%3A50.925186%2B00%3A00&task_id=extract

Airflow DAGs Cluster Activity Datasets Security Browse Admin Docs

DAG: hw4_stock_pipeline_final ETL pipeline for HW4: Extracts 90 trading days and loads to Snowflake. Schedule: @daily Next Run ID: 2025-10-07, 00:00:00 UTC

07/10/2025 12:28:21 AM All Run Types All Run States Clear Filters Auto-refresh 25

Press shift + / for Shortcuts

Task: hw4_stock_pipeline_final / 2025-10-06, 00:00:00 UTC / extract

Details Graph Gantt Code Event Log Logs XCom Task Duration

All Levels All File Sources Wrap Download See More

```
35c868a34991
*** Found local files:
***  * /opt/airflow/logs/dag_id=hw4_stock_pipeline_final/run_id>manual__2025-10-07T00:08:50.925186+00:00/task_id=extract/attempt=1.log
***  * [2025-10-07, 00:08:52 UTC] {local_task_job_runner.py:123} * Pre task execution logs
[2025-10-07, 00:08:52 UTC] {homework_5.py:62} INFO - [extract] Retrieved 100 total records for AMZN.
[2025-10-07, 00:08:52 UTC] {python.py:240} INFO - Done. Returned value was: [{'1. open': '221.0000', '2. high': '221.7300', '3. low': '216.0300', '4. close': '220.9000', '5
[2025-10-07, 00:08:52 UTC] {taskinstance.py:340} * Post task execution logs
```

SNOWFLAKE VALIDATIONS

app.snowflake.com/sfedu02/1vb17920/#/data/databases/USER_DB_PUMA/schemas/RAW/table/STOCK_PRICES_DAG

snowflake Database Explorer HORIZON CATALOG

Databases

Work with data Projects Ingestion Transformation AI & ML Monitoring Marketplace Horizon Catalog Catalog Data sharing Governance & security Manage Compute Admin

Search

Tables

MARKET_DATA MY_FORECASTS_2025_09_20 PROD_NEW_DATA PROD_OLD_DATA STOCK_PRICES STOCK_PRICES_DAG STOCK_PRICES_LAB

Views Stages File Formats

USER_DB_PYTHON USER_DB_QUAIL USER_DB_RACCOON

USER_DB_PUMA / RAW / STOCK_PRICES_DAG Describe Table Load Data

Table TRAINING_ROLE 28 minutes ago 63 3.5KB

Table Details Columns Data Preview Copy History Data Quality PREVIEW Lineage

Table definition

```
1 create or replace TABLE USER_DB_PUMA.RAW.STOCK_PRICES_DAG (
2     SYMBOL VARCHAR(16777216) NOT NULL,
3     DATE DATE NOT NULL,
4     OPEN NUMBER(18, 4),
5     CLOSE NUMBER(18, 4),
6 )
```

Show more

Privileges

Group by Role + Privilege

TRAINING_ROLE (Current Role) OWNERSHIP

app.snowflake.com/sfedu02/1vb17920/wKugPVvtclK?query

2025-10-05 8:56pm

TRAINING_ROLE PUMA_QUERY_WH (X-Small) Share

USER_DB_PUMA.RAW Settings

```
SELECT * FROM stock_prices_DAG
```

Results Chart

	SYMBOL	DATE	# OPEN	# CLOSE	# HIGH	# LOW	# VOLUME
51	AMZN	2025-07-25	232.2200	231.4400	232.4800	231.1800	28712095
52	AMZN	2025-07-24	229.1700	232.2300	236.0000	228.6400	42902266
53	AMZN	2025-07-23	228.4700	228.2900	228.7900	227.0900	28294852
54	AMZN	2025-07-22	229.6800	227.4700	230.0000	226.3500	37483702
55	AMZN	2025-07-21	225.8350	229.3000	229.6900	225.6500	40297556
56	AMZN	2025-07-18	225.1400	226.1300	226.4000	222.9800	37833807
57	AMZN	2025-07-17	223.3200	223.8800	224.5000	222.5100	31855831
58	AMZN	2025-07-16	225.8750	223.1900	226.1000	222.1800	39535926
59	AMZN	2025-07-15	226.2000	226.3500	227.2700	225.4550	34907294
60	AMZN	2025-07-14	225.0700	225.6900	226.6600	224.2400	35702597
61	AMZN	2025-07-11	223.5800	225.0200	226.6799	222.3700	50518307
62	AMZN	2025-07-10	221.5500	222.2600	222.7900	219.7000	30370591
63	AMZN	2025-07-09	221.0700	222.5400	224.2900	220.4700	38155121

Query Details

Query duration 363ms

Rows 63

Query ID 01bf8a09-0306-9879-0...

Show more

SYMBOL AMZN 63

DATE 2025-07-09

Ask Copilot