DATA 226: DATA WAREHOUSE AND PIPELINE

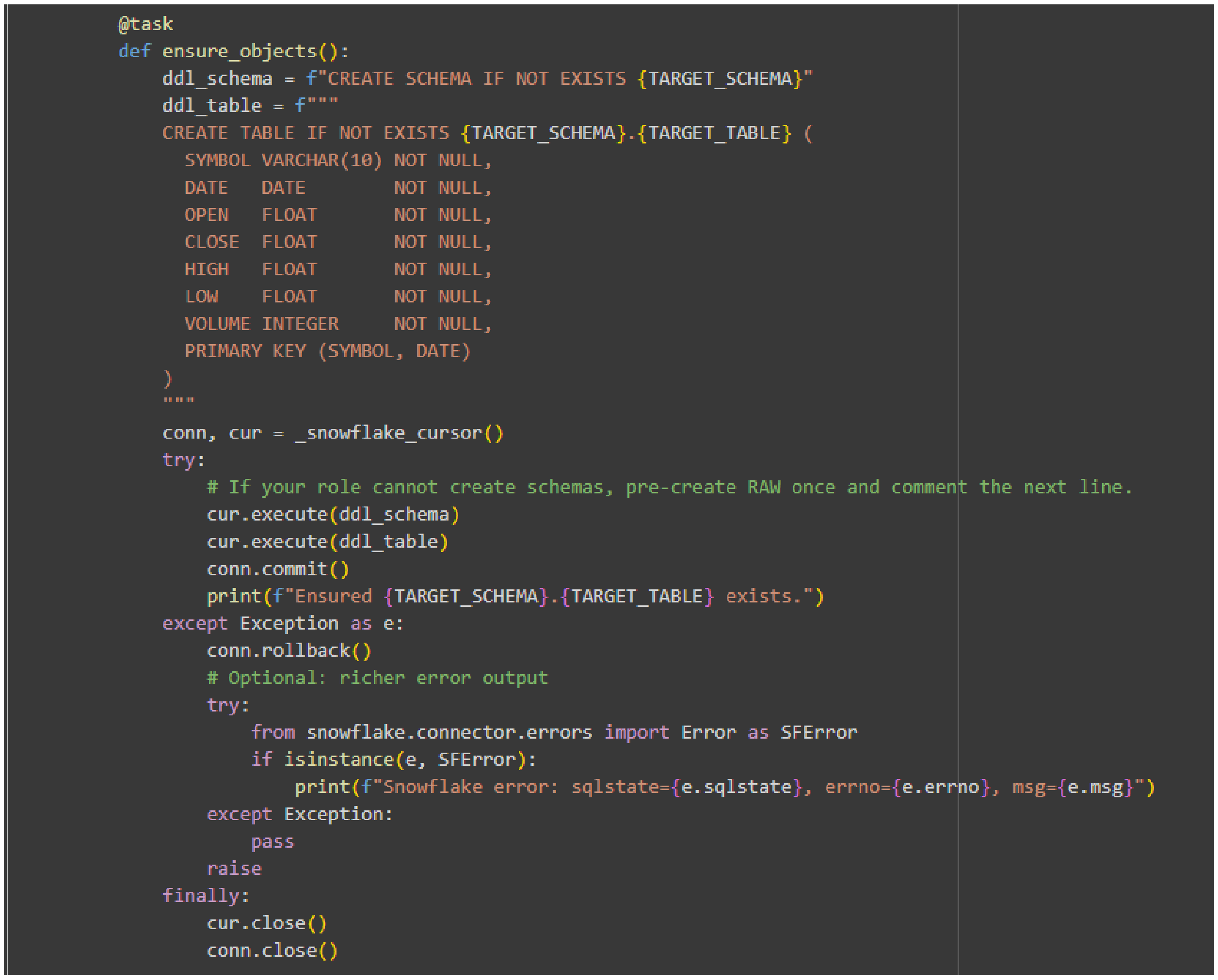
INSTRUCTOR: PROF KEEYONG HAN

ASSIGNMENT 5

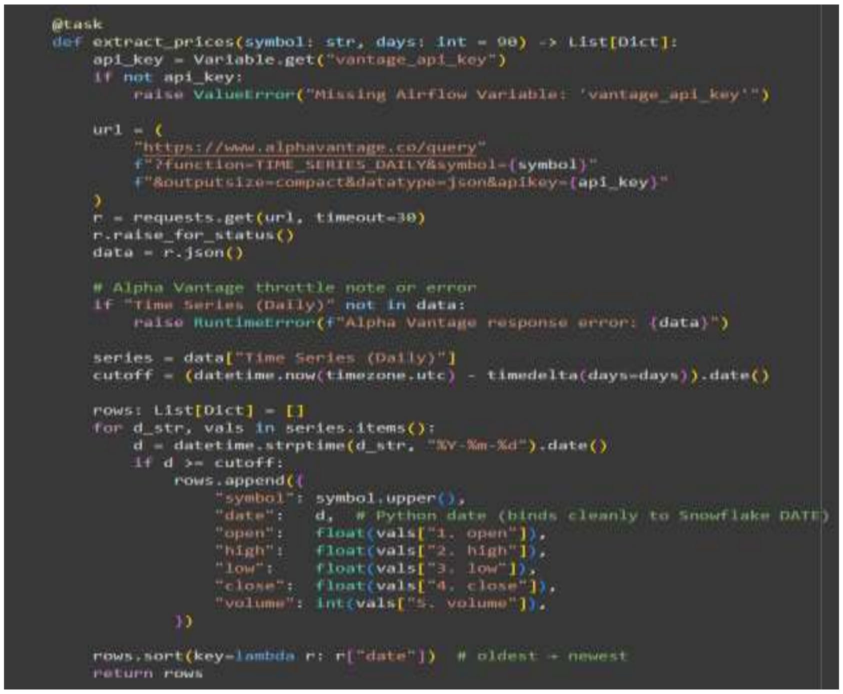
STEP 1:

Create tasks using @task decorator

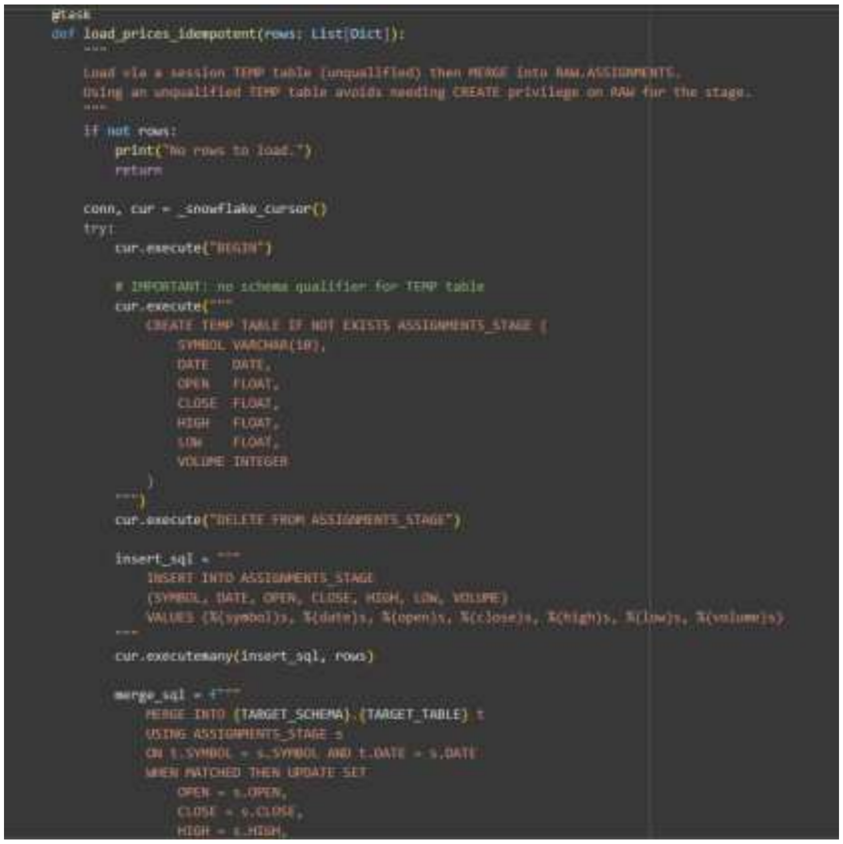
- You can use as many tasks as you want
- Schedule the tasks properly (task dependency)



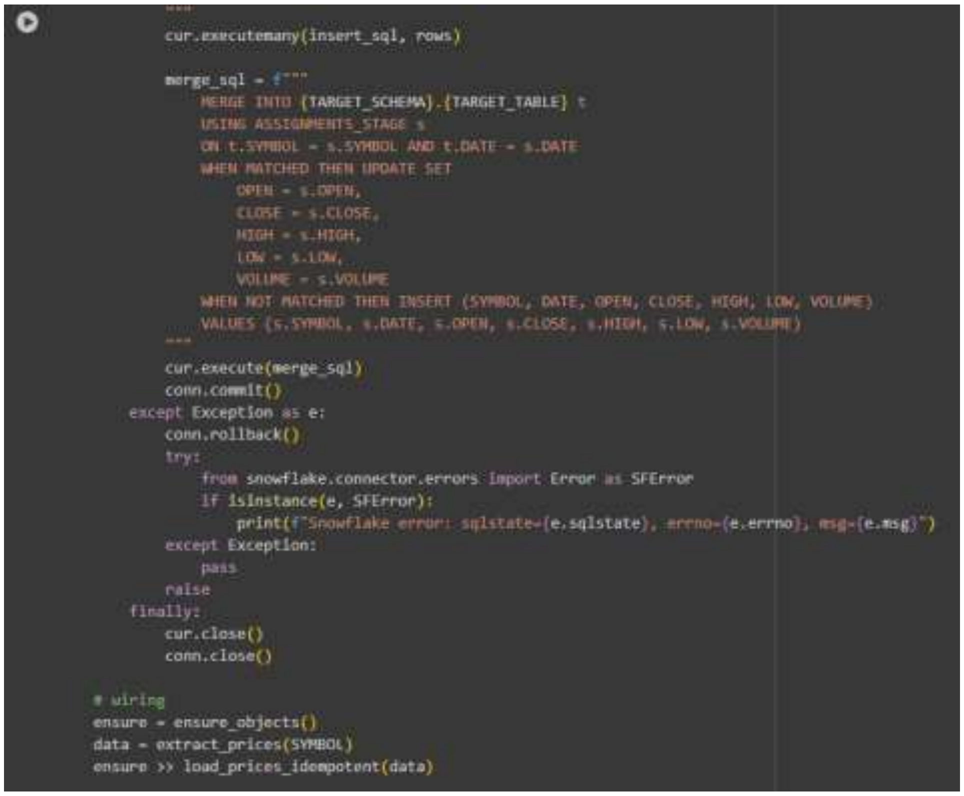
```
@task
def ensure_objects():
    ddl_schema = f"CREATE SCHEMA IF NOT EXISTS {TARGET_SCHEMA}"
    ddl_table = f"""
    CREATE TABLE IF NOT EXISTS {TARGET_SCHEMA}.{TARGET_TABLE} (
      SYMBOL VARCHAR(10) NOT NULL,
      DATE   DATE        NOT NULL,
      OPEN   FLOAT       NOT NULL,
      CLOSE  FLOAT       NOT NULL,
      HIGH   FLOAT       NOT NULL,
      LOW    FLOAT       NOT NULL,
      VOLUME INTEGER     NOT NULL,
      PRIMARY KEY (SYMBOL, DATE)
    )
    """
    conn, cur = _snowflake_cursor()
    try:
        # If your role cannot create schemas, pre-create RAW once and comment the next line.
        cur.execute(ddl_schema)
        cur.execute(ddl_table)
        conn.commit()
        print(f"Ensured {TARGET_SCHEMA}.{TARGET_TABLE} exists.")
    except Exception as e:
        conn.rollback()
        # Optional: richer error output
        try:
            from snowflake.connector.errors import Error as SFError
            if isinstance(e, SFError):
                print(f"Snowflake error: sqlstate={e.sqlstate}, errno={e.errno}, msg={e.msg}")
        except Exception:
            pass
        raise
    finally:
        cur.close()
        conn.close()
```



```python
@task
def extract_prices(symbol: str, days: int = 90) -> List[Dict]:
    api_key = Variable.get("vantage_api_key")
    if not api_key:
        raise ValueError("Missing Airflow Variable: 'vantage_api_key'")

    url = (
        "https://www.alphavantage.co/query"
        f"?function=TIME_SERIES_DAILY&symbol={symbol}"
        f"&outputsize=compact&datatype=json&apikey={api_key}"
    )
    r = requests.get(url, timeout=30)
    r.raise_for_status()
    data = r.json()

    # Alpha Vantage throttle note or error
    if "Time Series (Daily)" not in data:
        raise RuntimeError(f"Alpha Vantage response error: {data}")

    series = data["Time Series (Daily)"]
    cutoff = (datetime.now(timezone.utc) - timedelta(days=days)).date()

    rows: List[Dict] = []
    for d_str, vals in series.items():
        d = datetime.strptime(d_str, "%Y-%m-%d").date()
        if d >= cutoff:
            rows.append({
                "symbol": symbol.upper(),
                "date":   d,  # Python date (binds cleanly to Snowflake DATE)
                "open":   float(vals["1. open"]),
                "high":   float(vals["2. high"]),
                "low":    float(vals["3. low"]),
                "close":  float(vals["4. close"]),
                "volume": int(vals["5. volume"]),
            })

    rows.sort(key=lambda r: r["date"])  # oldest → newest
    return rows
```

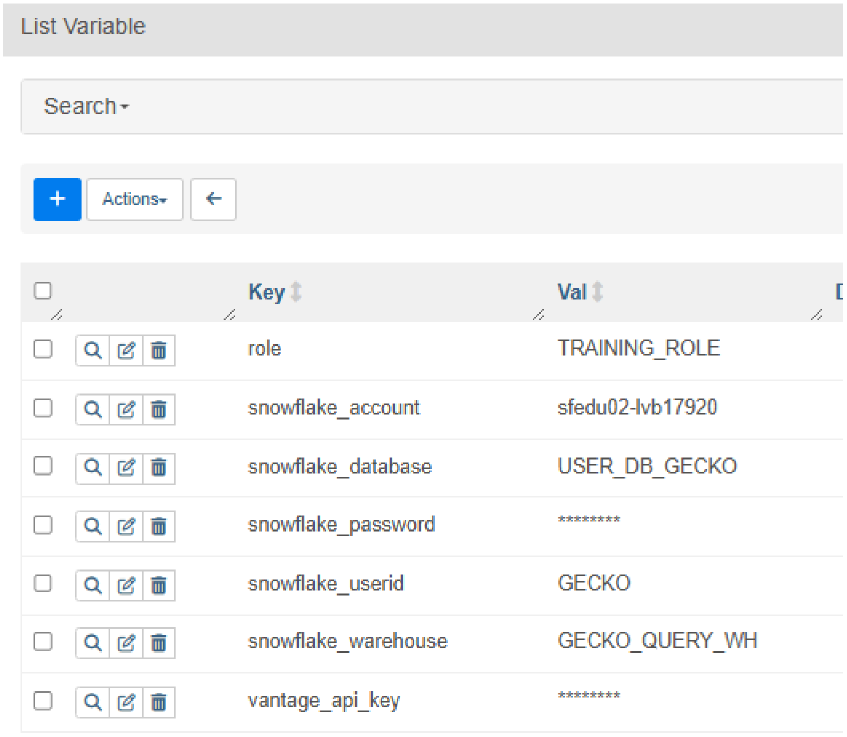


```python
@task
def load_prices_idempotent(rows: List[Dict]):
    """
    Load via a session TEMP table (unqualified) then MERGE into RAW.ASSIGNMENTS.
    Using an unqualified TEMP table avoids needing CREATE privilege on RAW for the stage.
    """
    if not rows:
        print("No rows to load.")
        return

    conn, cur = _snowflake_cursor()
    try:
        cur.execute("BEGIN")

        # IMPORTANT: no schema qualifier for TEMP table
        cur.execute("""
            CREATE TEMP TABLE IF NOT EXISTS ASSIGNMENTS_STAGE (
                SYMBOL VARCHAR(10),
                DATE   DATE,
                OPEN   FLOAT,
                CLOSE  FLOAT,
                HIGH   FLOAT,
                LOW    FLOAT,
                VOLUME INTEGER
            )
        """)
        cur.execute("DELETE FROM ASSIGNMENTS_STAGE")

        insert_sql = """
            INSERT INTO ASSIGNMENTS_STAGE
            (SYMBOL, DATE, OPEN, CLOSE, HIGH, LOW, VOLUME)
            VALUES (%(symbol)s, %(date)s, %(open)s, %(close)s, %(high)s, %(low)s, %(volume)s)
        """
        cur.executemany(insert_sql, rows)

        merge_sql = f"""
            MERGE INTO {TARGET_SCHEMA}.{TARGET_TABLE} t
            USING ASSIGNMENTS_STAGE s
            ON t.SYMBOL = s.SYMBOL AND t.DATE = s.DATE
            WHEN MATCHED THEN UPDATE SET
                OPEN = s.OPEN,
                CLOSE = s.CLOSE,
                HIGH = s.HIGH,
```



```
        """
        cur.executemany(insert_sql, rows)

        merge_sql = f"""
            MERGE INTO {TARGET_SCHEMA}.{TARGET_TABLE} t
            USING ASSIGNMENTS_STAGE s
            ON t.SYMBOL = s.SYMBOL AND t.DATE = s.DATE
            WHEN MATCHED THEN UPDATE SET
                OPEN = s.OPEN,
                CLOSE = s.CLOSE,
                HIGH = s.HIGH,
                LOW = s.LOW,
                VOLUME = s.VOLUME
            WHEN NOT MATCHED THEN INSERT (SYMBOL, DATE, OPEN, CLOSE, HIGH, LOW, VOLUME)
            VALUES (s.SYMBOL, s.DATE, s.OPEN, s.CLOSE, s.HIGH, s.LOW, s.VOLUME)
        """
        cur.execute(merge_sql)
        conn.commit()
    except Exception as e:
        conn.rollback()
        try:
            from snowflake.connector.errors import Error as SFError
            if isinstance(e, SFError):
                print(f"Snowflake error: sqlstate={e.sqlstate}, errno={e.errno}, msg={e.msg}")
        except Exception:
            pass
        raise
    finally:
        cur.close()
        conn.close()

# wiring
ensure = ensure_objects()
data = extract_prices(SYMBOL)
ensure >> load_prices_idempotent(data)
```

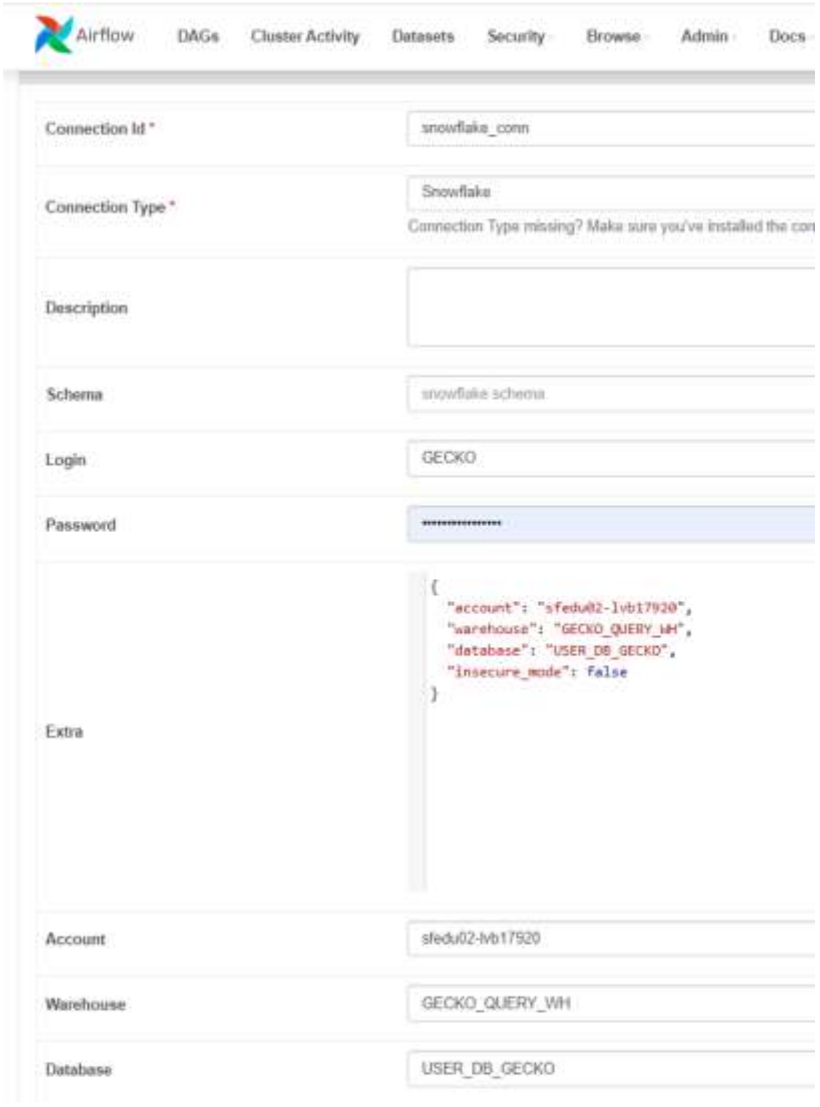
Step 2:
Set up a variable for Alpha Vantage API key



List Variable

Key	Val
role	TRAINING_ROLE
snowflake_account	sfedu02-lvb17920
snowflake_database	USER_DB_GECKO
snowflake_password	********
snowflake_userid	GECKO
snowflake_warehouse	GECKO_QUERY_WH
vantage_api_key	********

Step 3:
Set up Snowflake Connection



Connection Id * snowflake_conn

Connection Type * Snowflake

Connection Type missing? Make sure you've installed the cor

Description

Schema snowflake schema

Login GECKO

Password ••••••••••••••••

Extra

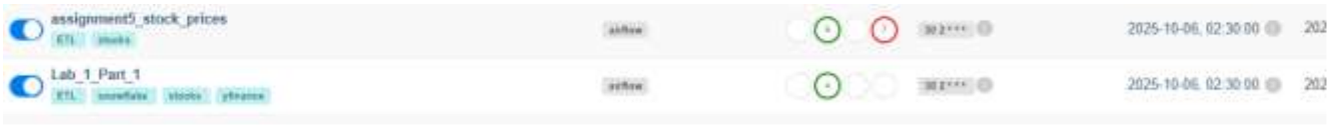
```
{
  "account": "sfedu02-lvb17920",
  "warehouse": "GECKO_QUERY_WH",
  "database": "USER_DB_GECKO",
  "insecure_mode": false
}
```

Account sfedu02-lvb17920

Warehouse GECKO_QUERY_WH

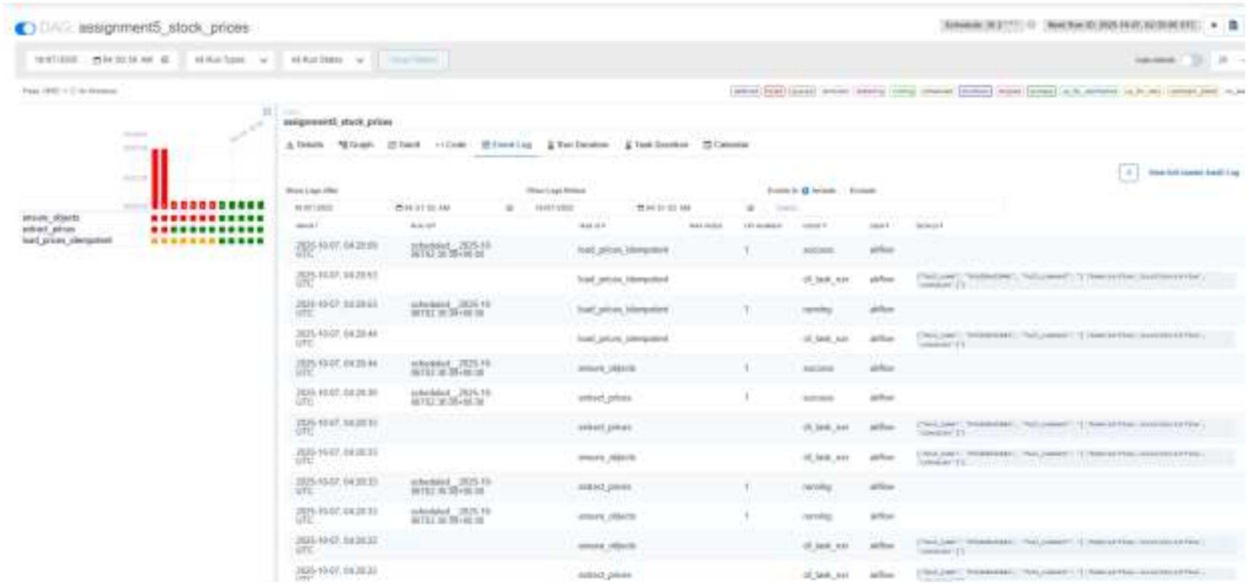
Database USER_DB_GECKO

Step 4: Ensure the overall DAG is implemented properly and runs successfully



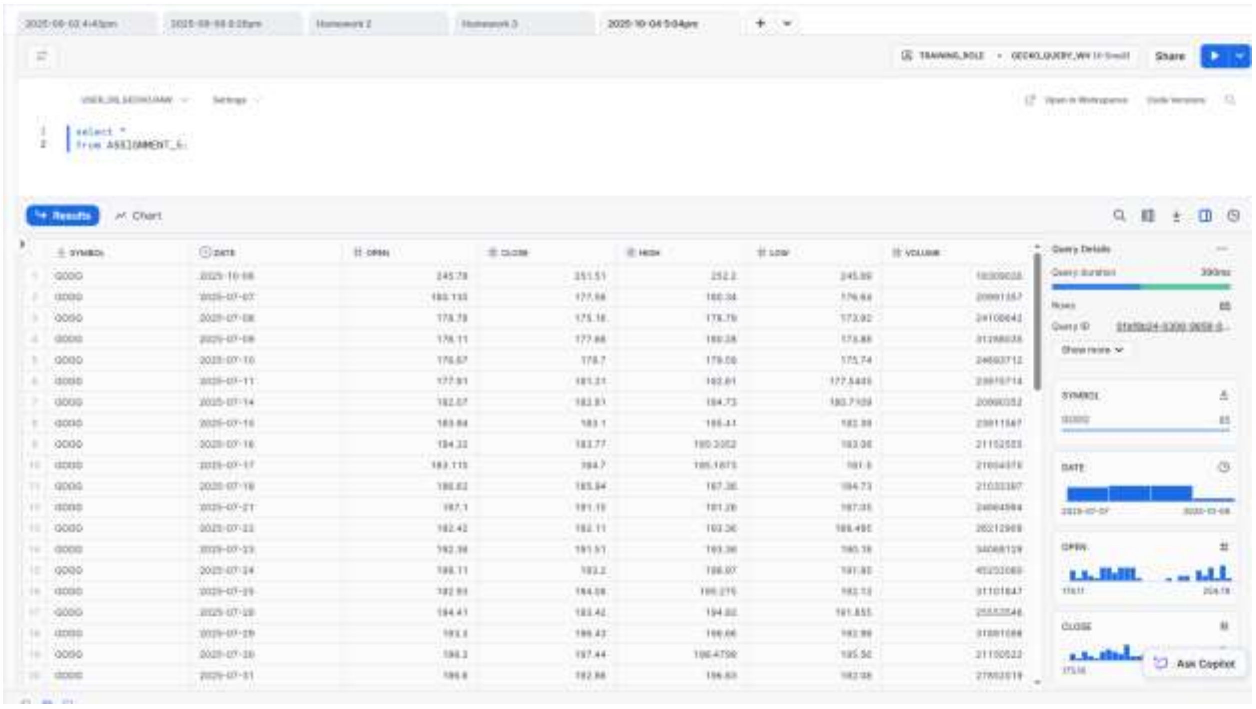
assignment5_stock_prices ETL stocks	airflow	30 2 * * *	2025-10-06, 02:30:00	202
Lab_1_Part_1 ETL snowflake stocks yfinance	airflow	30 2 * * *	2025-10-06, 02:30:00	202

Step 5: Capture two screenshot of your Airflow Web UI



[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2025-10-07, 04:20:05 UTC	scheduled__2025-10-06T02:30:00+00:00	load_prices_idempotent		1	success	airflow	
2025-10-07, 04:20:53 UTC		load_prices_idempotent			cli_task_run	airflow	[illegible]
2025-10-07, 04:20:03 UTC	scheduled__2025-10-06T02:30:00+00:00	load_prices_idempotent		1	running	airflow	
2025-10-07, 04:20:44 UTC		load_prices_idempotent			cli_task_run	airflow	[illegible]
2025-10-07, 04:20:44 UTC	scheduled__2025-10-06T02:30:00+00:00	ensure_objects		1	success	airflow	
2025-10-07, 04:20:30 UTC	scheduled__2025-10-06T02:30:00+00:00	extract_prices		1	success	airflow	
2025-10-07, 04:20:31 UTC		extract_prices			cli_task_run	airflow	[illegible]
2025-10-07, 04:20:33 UTC		ensure_objects			cli_task_run	airflow	[illegible]
2025-10-07, 04:20:33 UTC	scheduled__2025-10-06T02:30:00+00:00	extract_prices		1	running	airflow	
2025-10-07, 04:20:33 UTC	scheduled__2025-10-06T02:30:00+00:00	ensure_objects		1	running	airflow	
2025-10-07, 04:20:22 UTC		ensure_objects			cli_task_run	airflow	[illegible]
2025-10-07, 04:20:22 UTC		extract_prices			cli_task_run	airflow	[illegible]

Snowflake Answer:



```
select *
from ASSIGNMENT_5;
```

	SYMBOL	DATE	OPEN	CLOSE	HIGH	LOW	VOLUME
1	GOOG	2025-10-06	245.78	251.51	252.2	245.59	18309028
2	GOOG	2025-07-07	180.135	177.56	180.34	176.64	20991357
3	GOOG	2025-07-08	178.78	175.16	178.79	173.92	24100642
4	GOOG	2025-07-09	176.11	177.66	180.28	173.86	31298035
5	GOOG	2025-07-10	176.67	178.7	179.08	175.74	24660712
6	GOOG	2025-07-11	177.91	181.21	182.81	177.5405	23615714
7	GOOG	2025-07-14	182.07	182.81	184.73	180.7109	20660352
8	GOOG	2025-07-15	183.84	183.1	186.41	182.59	23611567
9	GOOG	2025-07-16	184.32	183.77	185.3352	183.06	21152553
10	GOOG	2025-07-17	183.115	184.7	185.1873	181.5	21604376
11	GOOG	2025-07-18	186.62	185.94	187.36	184.73	21033387
12	GOOG	2025-07-21	187.1	191.15	191.26	187.05	24664584
13	GOOG	2025-07-22	182.42	192.11	193.36	188.495	26212969
14	GOOG	2025-07-23	192.38	191.51	193.36	190.18	34068129
15	GOOG	2025-07-24	198.11	193.2	198.07	191.85	45253060
16	GOOG	2025-07-25	192.93	194.08	195.275	192.13	31101847
17	GOOG	2025-07-28	194.41	193.42	194.82	191.855	25552546
18	GOOG	2025-07-29	193.3	196.43	196.66	192.99	31881068
19	GOOG	2025-07-30	196.2	197.44	198.4799	195.50	21150522
20	GOOG	2025-07-31	196.8	192.66	196.63	192.08	27052019

The Github Link:
https://github.com/NAMAN-CHHEDA/DATA-226-ASSIGNEMNT-5-