

Assignment - Airflow Connection & DAGs Running

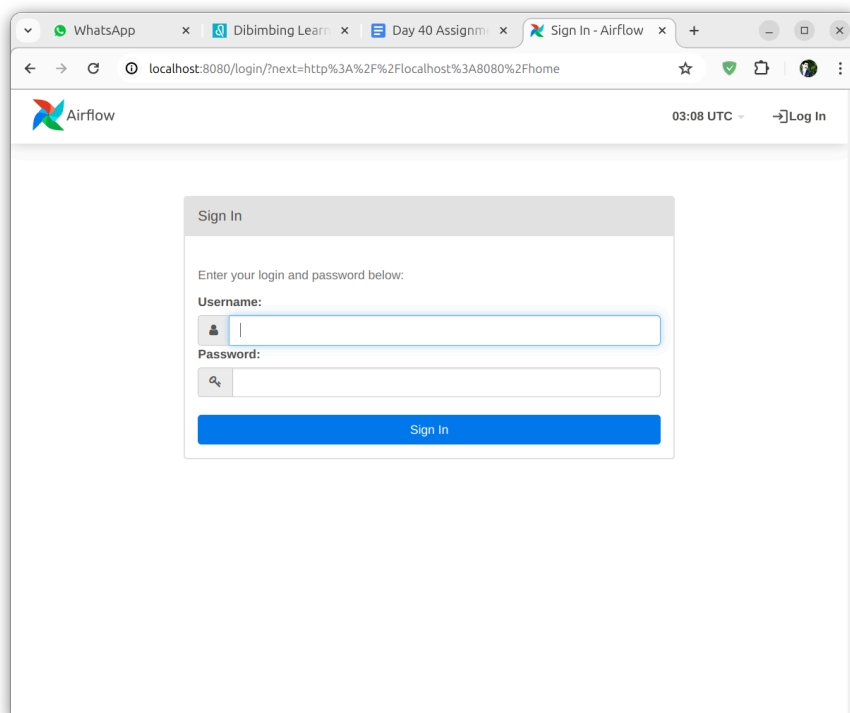
1. (10 point) Jalankan docker apache airflow kemudian lampirkan bukti:

a. Screenshot terminal

```
(venv) ~/Documents/dibimbing/airflow/MSIB-DW-Batch_6-Airflow/airflow git:[main]
docker compose up -d
WARN[0000] /home/agung/Documents/dibimbing/airflow/MSIB-DW-Batch_6-Airflow/airfl
[+] Running 4/4
  ✓ Container postgres          Healthy
  ✓ Container airflow-init      Exited
  ✓ Container airflow-scheduler Started
  ✓ Container airflow-webserver Started
```

```
(venv) ~/Documents/dibimbing/airflow/MSIB-DW-Batch_6-Airflow/airflow git:[main]
docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Status}}"
CONTAINER ID   IMAGE                                NAMES                STATUS
158890d0d438   apache/airflow:2.5.1-python3.10    airflow-webserver    Up 6 minutes (healthy)
6b6d5157a74a   apache/airflow:2.5.1-python3.10    airflow-scheduler    Up 6 minutes
fd29d75a41eb   postgres:13                         postgres             Up 6 minutes (healthy)
```

b. Screenshot localhost:8080



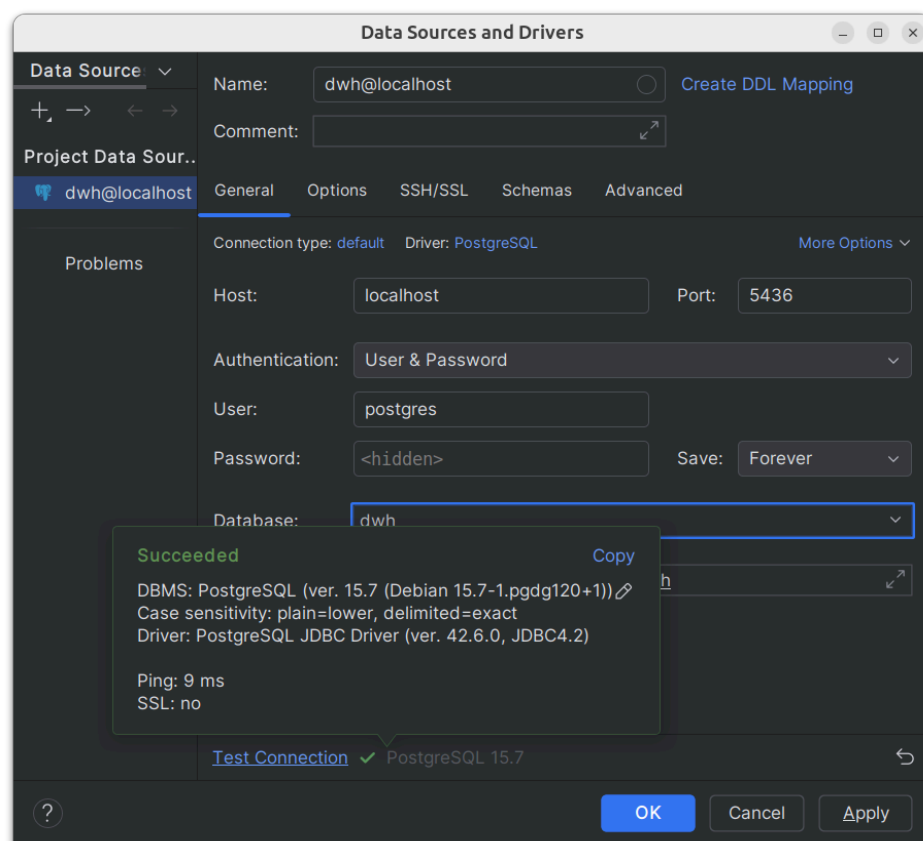
2. (10 point) Jalankan docker database mysql dan postgresql kemudian lampirkan bukti:

a. Screenshoot terminal

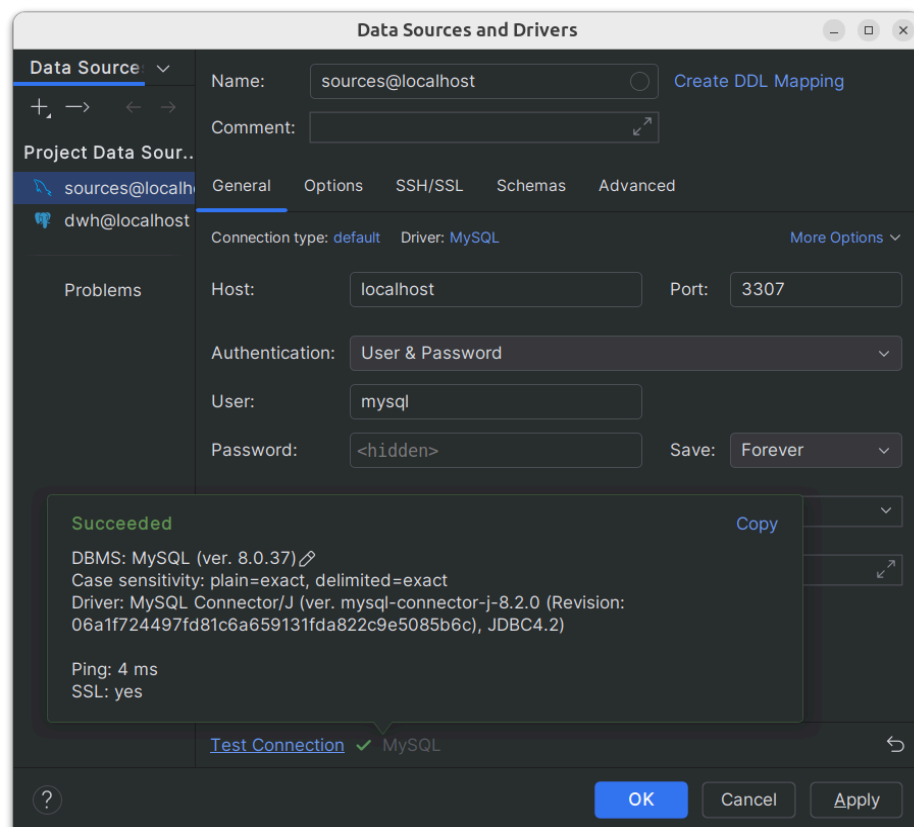
```
(venv) ~/Documents/dibimbing/airflow/MSIB-DW-Batch_6-
docker compose up -d
WARN[0000] /home/agung/Documents/dibimbing/airflow/MS
[+] Running 2/2
 ✓ Container db-mysql      Started
 ✓ Container db-postgres   Started
```

b. Screenshoot Dbeaver / MySQL Connection Extension yang telah terhubung dengan postgresql dan mysql

Postgres



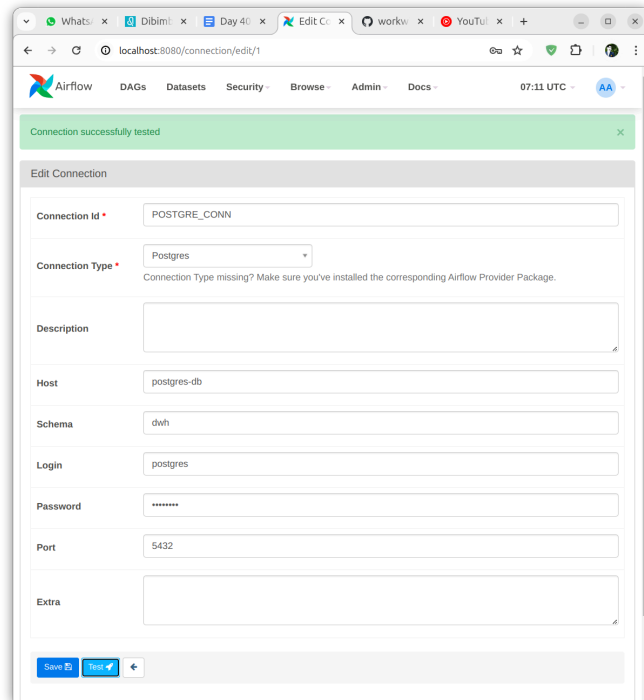
MySQL



3. (10 point) Buat connection MySQL dan PostgreSQL pada Apache Airflow kemudian

a. Screenshoot UI hasil testing yang menunjukkan **connection successful**

Postgre Connection

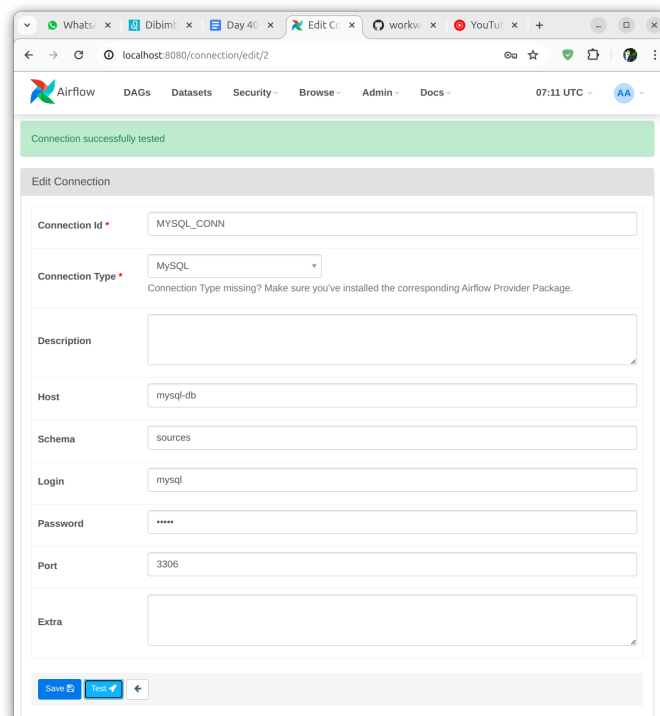


The screenshot shows the Apache Airflow web interface at localhost:8080/connection/edit/1. A green banner at the top indicates "Connection successfully tested". The "Edit Connection" form is visible with the following details:

- Connection Id: POSTGRE_CONN
- Connection Type: Postgres
- Description: (empty)
- Host: postgres-db
- Schema: dwh
- Login: postgres
- Password: (masked with asterisks)
- Port: 5432
- Extra: (empty)

At the bottom of the form, there are buttons for "Save", "Test", and a back arrow.

MySQL Connection



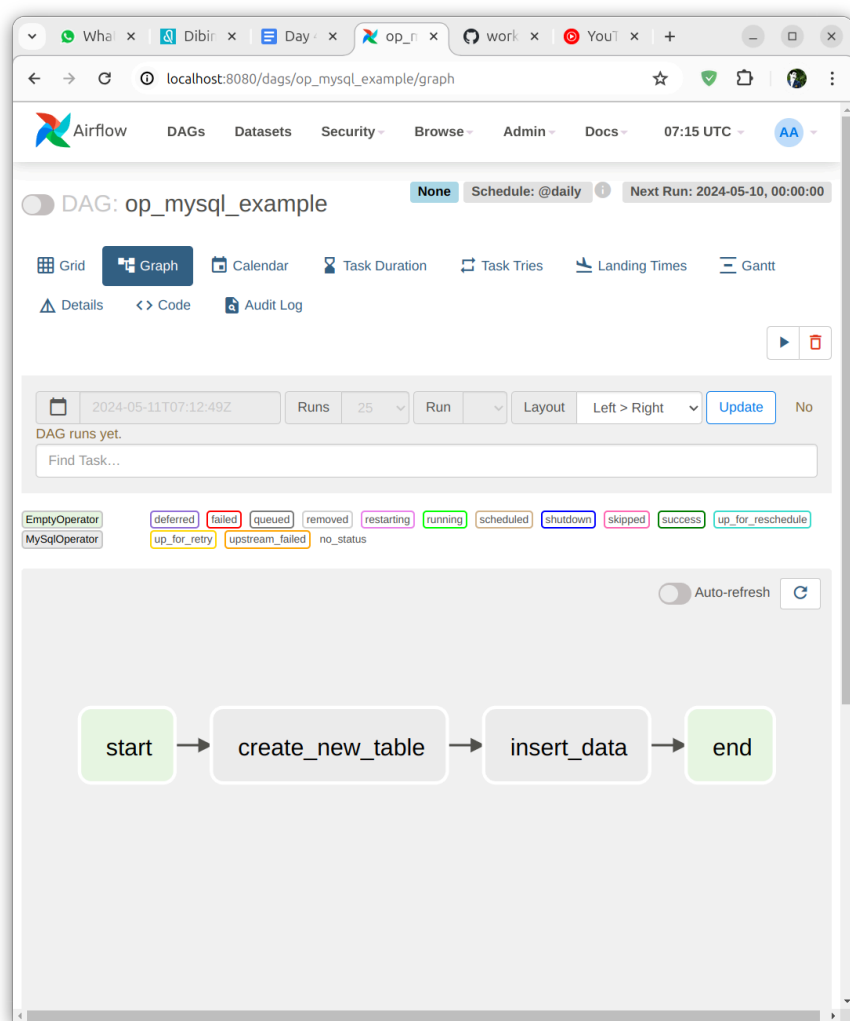
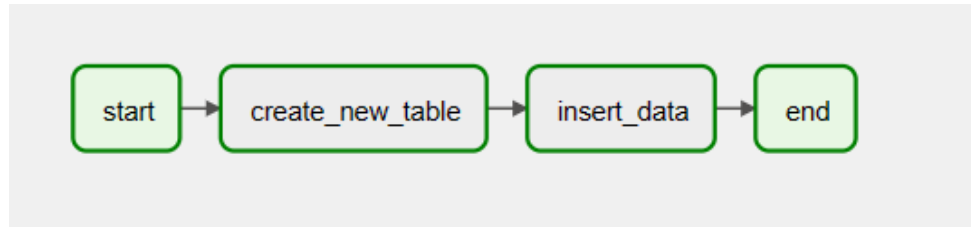
The screenshot shows the Apache Airflow web interface at localhost:8080/connection/edit/2. A green banner at the top indicates "Connection successfully tested". The "Edit Connection" form is visible with the following details:

- Connection Id: MYSQL_CONN
- Connection Type: MySQL
- Description: (empty)
- Host: mysql-db
- Schema: sources
- Login: mysql
- Password: (masked with asterisks)
- Port: 3306
- Extra: (empty)

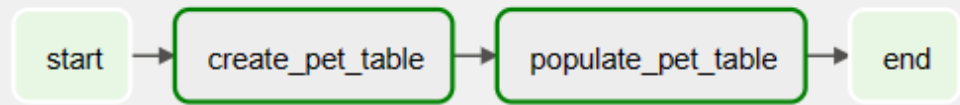
At the bottom of the form, there are buttons for "Save", "Test", and a back arrow.

4. (10 point) Modifikasi DAGs `op_mysql` dan `op_postgresql` yang sudah ada dengan menambahkan `EmptyOperator` sehingga tampilan kedua DAG tersebut menjadi sebagai berikut:

a. `op_mysql` :



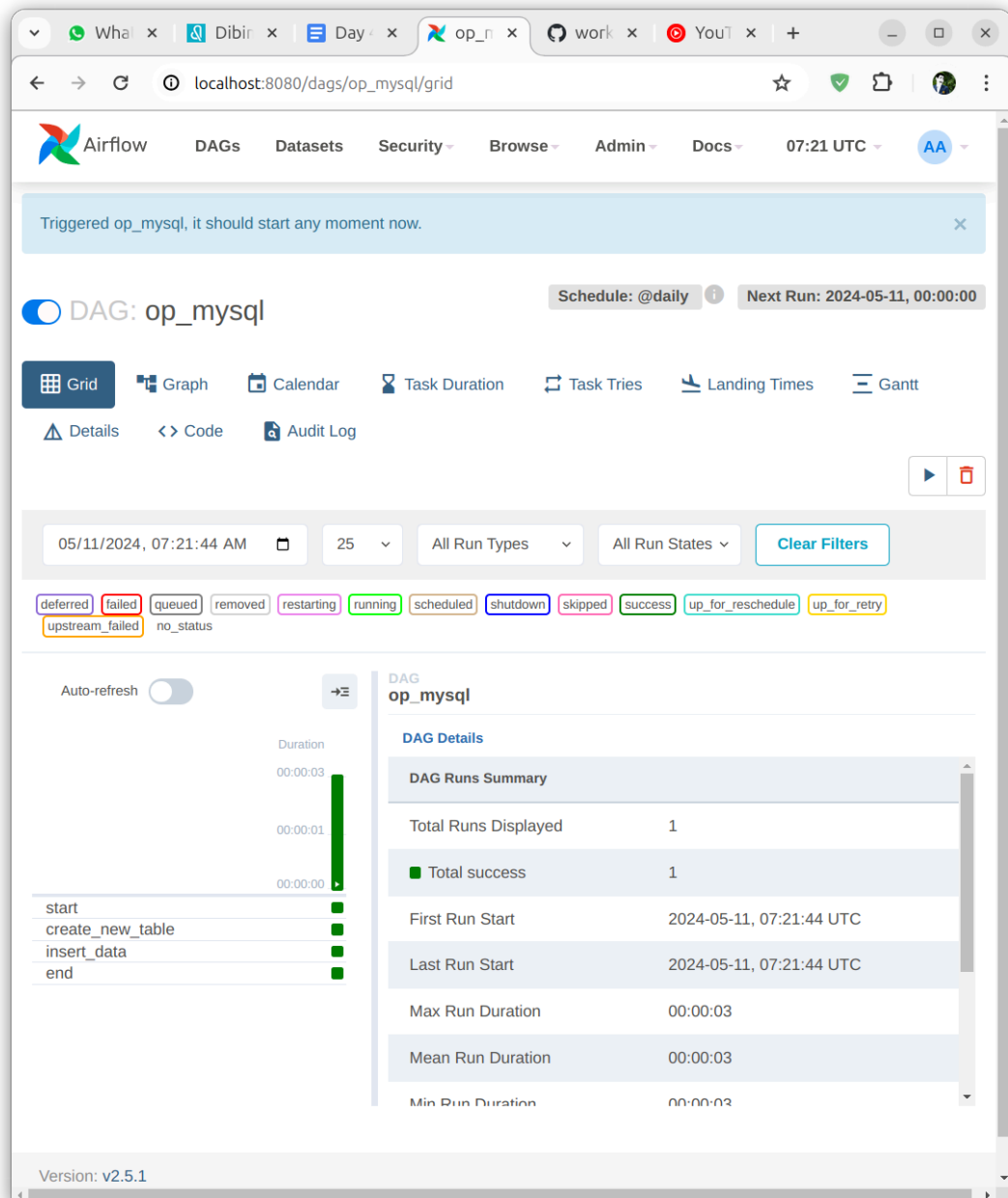
b. op_postgresql



The screenshot displays the Airflow web interface for the DAG 'op_postgres_example'. The interface includes a top navigation bar with links for DAGs, Datasets, Security, Browse, Admin, and Docs. The DAG is currently in a 'None' state with a '@weekly' schedule and a next run time of '2024-05-05, 00:00:00'. Below the DAG name, there are tabs for Grid, Graph (selected), Calendar, Task Duration, Task Tries, Landing Times, and Gantt. A 'Details' link and a 'Code' button are also visible. The main area shows the DAG's graph, which is a linear flow from 'start' to 'create_pet_table' to 'populate_pet_table' to 'end'. The graph is displayed in a 'Left > Right' layout. A legend at the bottom lists various task states: EmptyOperator, PostgresOperator, deferred, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up_for_retry, up_for_reschedule, upstream_failed, and no_status. The 'Auto-refresh' toggle is turned off.

5. (10 point) Jalankan DAGs `op_mysql` dan `op_postgresql` hingga `success` kemudian lampirkan hasil screenshotnya. Usahakan sampai pipeline berjalan `success` semua

DAG `op_mysql`



op_postgresql

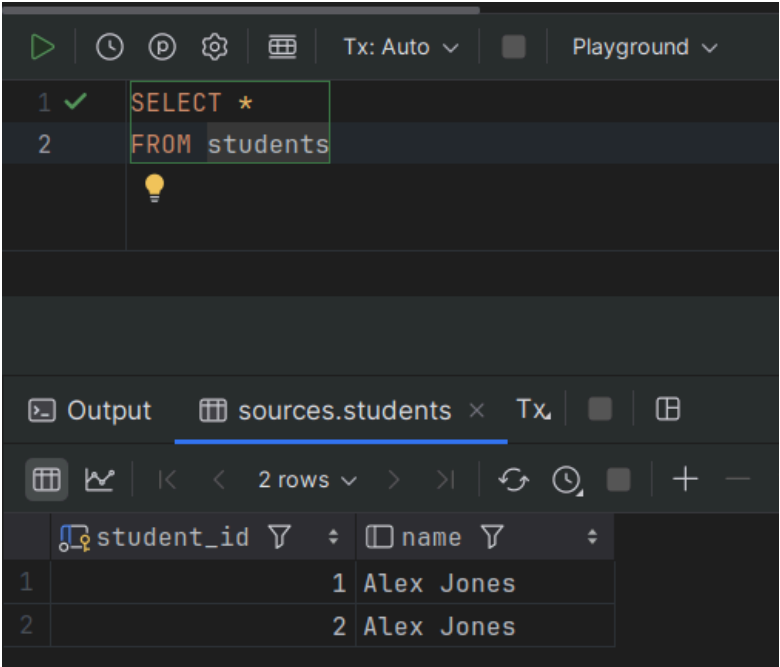
The screenshot displays the Airflow web interface for the DAG 'op_postgres'. The interface includes a top navigation bar with links to DAGs, Datasets, Security, Browse, Admin, and Docs. A notification at the top states: 'Triggered op_postgres, it should start any moment now.' The DAG is scheduled '@weekly' with the next run on '2024-05-12, 00:00:00'. The interface shows various views: Grid (selected), Graph, Calendar, Task Duration, Task Tries, Landing Times, and Gantt. Below these are links for Details, Code, and Audit Log. A filter bar shows the date '05/11/2024, 07:22:23 AM' and a dropdown for '25' items. A legend at the bottom lists various run states: deferred, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up_for_reschedule, up_for_retry, upstream_failed, and no_status. The 'op_postgres' DAG details are shown on the right, including a summary table with the following data:

DAG Runs Summary	
Total Runs Displayed	1
Total success	1
First Run Start	2024-05-11, 07:22:24 UTC
Last Run Start	2024-05-11, 07:22:24 UTC
Max Run Duration	00:00:03
Mean Run Duration	00:00:03
Min Run Duration	00:00:03

The bottom of the interface shows the version 'v2.5.1'.

6. (10 point) Tampilkan hasil table yang dibuat oleh `op_mysql` dan `op_postgresql`

Table student (op_mysql)



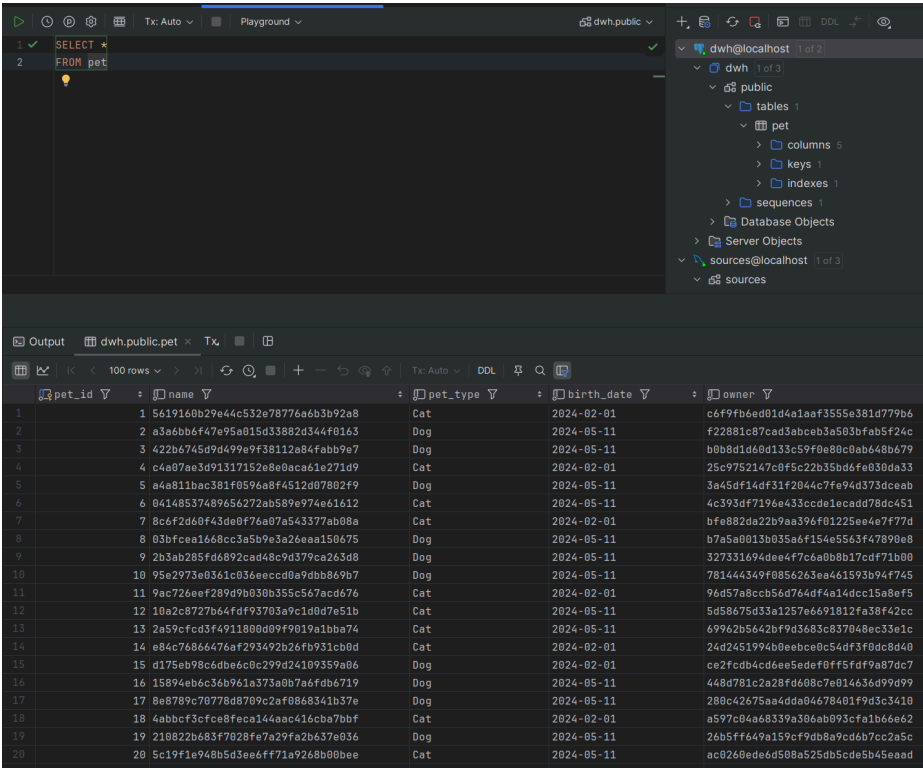
The screenshot shows a SQL playground interface. The query editor contains the following SQL code:

```
1 SELECT *
2 FROM students
```

The results pane shows the output of the query, displaying two rows of data from the `students` table:

student_id	name
1	Alex Jones
2	Alex Jones

Table



The screenshot shows a SQL playground interface. The query editor contains the following SQL code:

```
1 SELECT *
2 FROM pet
```

The results pane shows the output of the query, displaying 20 rows of data from the `pet` table:

pet_id	name	pet_type	birth_date	owner
1	5619160b29e44c532e78776a6b3b92a8	Cat	2024-02-01	c6f9fb6ed01d4a1aef3555e381d779b6
2	a3a6bb6f47e95a015d33882d344f0163	Dog	2024-05-11	f22881c87cad3abceb3a503bfbaf5f24c
3	422b6745d9d499e9f38112a84fab9e7	Dog	2024-05-11	b0b8d1d60d133c59f0e80c0ab648b679
4	c4a07ae3d91317152e8e0aca61e271d9	Cat	2024-02-01	25c9752147c0f5c22b35bd6fe030da33
5	a4a811bac381f0596a8f4512d07802f9	Dog	2024-05-11	3a45df14df31f2044c7fe94d373dceab
6	04148537489656272ab589e974e61a12	Cat	2024-05-11	4c393df719eae433cccd1ecadd78dc451
7	8c4cf2d60f43de0f76a07a543377ab08a	Cat	2024-02-01	bfe882da22b9aa39f0122ee4e7477d
8	03bfce1a68cc3a5b9e3a2eaa150675	Dog	2024-05-11	b7a5a0013b035a6f154e5563f47890e8
9	2b3ab285fd6892cad48c9d379ca263d8	Dog	2024-05-11	327331694dee4f7c6a0b8b17cdf71b00
10	95e2973e0361c036eccc0a9dbb869b7	Dog	2024-05-11	781444349f0856263ea461593b94f745
11	9ac726eef289d9b03b355c567ac6d76	Cat	2024-02-01	96d57a8ccb56d764df4a14dcc15a8ef5
12	10a2c8727b64fd9f93703a9c1d0d7e51b	Cat	2024-05-11	5d58675d33a1257e6691812fa30f42cc
13	2a59cfc3f4911800d09f9019a1bba74	Cat	2024-05-11	69962b5642bf9d3683c837048ec33e1c
14	e84c76866476af293492b26fb931cb0d	Cat	2024-02-01	24d2451994b0eebce0e54df3f0dc8d40
15	d175eb98c6db6c6c0c299d24109359a86	Dog	2024-02-01	ce2fdb4cd4ee5ede0ff5f5df9a87dc7
16	15894eb6c36b961a373a0b7a6fdb6719	Dog	2024-05-11	448d781c2a28fd608c7e014636d99d99
17	8e8789c70778d8709c2af0868341b37e	Dog	2024-05-11	280c42675aa4dda04678401f9d3c3410
18	4abbcf3cfe8feca144aac416cba7bbf	Cat	2024-02-01	a597c04a68339a306ab093cfa1b66e62
19	210822b683f7028fe7a29fa2b637e036	Dog	2024-05-11	26b5ff649a159cf9db8a9cd6b7cc2a5c
20	5c19f1e948b5d3ee6ff71a9268b0bbee	Cat	2024-05-11	ac026ede6d508a525db5cde5b45eadd

7. (30 point) Jelaskan apa yang dikerjakan oleh `op_mysql` dan `op_postgresql`? Apa perbedaan sebelum dan sesudah dijalankannya DAG? dan apa yang dapat di improve dari kedua DAG tersebut?

a. `op_mysql`

Proses yang dikerjakan oleh DAG ini adalah:

- Mengeksekusi SQL query dari parameter function untuk membuat table students jika tabel tersebut belum ada dengan model (student_id, name).
- Mengeksekusi SQL query dari parameter fungsi untuk menambahkan 1 data (insert) ke kolom nama dengan nilai 'Alex Jones'.
- Proses ini dijadwalkan berjalan setiap hari.

Perbedaan yang terjadi setelah dijalankan adalah munculnya 1 tabel baru yaitu tabel students jika sebelumnya tabel tersebut belum ada. Selain tabel, juga akan muncul 1 record baru dengan student_id yang otomatis terisi dan nama student 'Alex Jones'.

Pada dasarnya DAG ini hanya melakukan proses dummy sederhana, namun tetap ada beberapa hal yang bisa diimprove seperti dengan membuat random student name generator sehingga setiap DAG dijalankan, nama student yang di-insert akan berbeda.

b. `op_postgresql`

Proses yang dikerjakan oleh DAG ini adalah:

- Mengeksekusi SQL query dari parameter fungsi untuk membuat table pet dengan model (pet_id, name, pet_type, birth_date, owner).
- Mengeksekusi SQL query dari file sql dari direktori `sql/insert_pet.sql` untuk menambahkan data hewan peliharaan ke tabel pet yang dibuat pada langkah sebelumnya.
- Script `insert_pet.sql` sendiri akan membuat 100 data hewan peliharaan dummy. Untuk column id pet dan id pemilik, script akan membuat data hash MD5 dummy dari random text. Untuk

jenis hewan, script akan membuat pilihan random antara 'Dog' dan 'Cat' berdasarkan nilai random yang dikembalikan oleh fungsi random(). Terakhir, untuk tanggal lahir, script akan membuat data tanggal lahir dummy dengan mengurangi tanggal hari ini dengan jumlah hari yang dikalikan nilai random dalam rentang waktu 100 hari. Selanjutnya data akan diinsert ke tabel pet.

Setelah DAG dijalankan, akan terdapat perbedaan pada database yaitu terdapat 1 tabel baru (tabel pet). Tabel tersebut memiliki 100 record baru (jika DAG dijalankan pertama kali).

Terdapat beberapa hal yang bisa improve antara lain menambahkan index pada tabel owner dan jenis hewan peliharaan untuk meningkatkan performa pencarian.

8. (10 point) Buat Kesimpulan mengenai Apache Airflow.

Apache Airflow merupakan platform open source yang digunakan untuk menjalankan job/pekerjaan secara otomatis dan terjadwal. Operasi-operasi dalam job tersebut dituangkan dalam DAG (Directed Acyclic Graph) dimana setiap operasi memiliki arah tertentu dan memiliki awal dan akhir yang jelas. Operasi dalam task-task airflow didefinisikan dalam bahasa Python sehingga pengguna lebih leluasa untuk melakukan pengolahan karena tidak terbatas hanya di SQL saja. Pemantauan juga menjadi lebih mudah karena air flow sudah menyediakan menu monitoring yang cukup lengkap. Kita bisa melihat proses yang sukses gagal dan sedang berjalan baik dalam bentuk grid dan kalender. Selain itu kita juga bisa melihat efektivitas dari operator yang kita definisikan karena kita bisa melihat waktu dari masing-masing operator yang dijalankan.