Case Study Basic SQL Queries

Oleh: Iffat A. Ahnaf

Proyek ini mendokumentasikan serangkaian query SQL dasar yang dibuat untuk praktik manipulasi dan pengambilan data dari dua skema database: dibimbing (pembuatan skema baru) dan dvdrental (kueri pada data yang sudah ada).

Case study ini mencakup pembuatan tabel (DDL), penyisipan data (DML), fungsi agregasi, logika kondisional (CASE), penggabungan tabel (JOIN), dan operasi set (UNION).

Tools yang Digunakan

- DBeaver: Sebagai database management tool untuk menjalankan query SQL dan mengelola skema database.
- PostgreSQL (dvdrental): Sebagai database utama untuk case study.

---

1. Membuat Database dan Tabel

1.1. Buatlah database bernama `dibimbing`.

- Query:

```
-- Membuat database
CREATE DATABASE dibimbing;
```

1.2. Di dalamnya, buat tabel `students` dengan skema berikut:

`id` (int, primary key)
`nama` (varchar)
`institute` (varchar)
`berat_badan` (float)
`tinggi_badan` (float)

- Query:

```
-- Membuat table
CREATE TABLE students (
      id INT PRIMARY KEY,
      nama VARCHAR(100),
      institute VARCHAR(100),
      berat_badan FLOAT,
      tinggi_badan FLOAT);
```

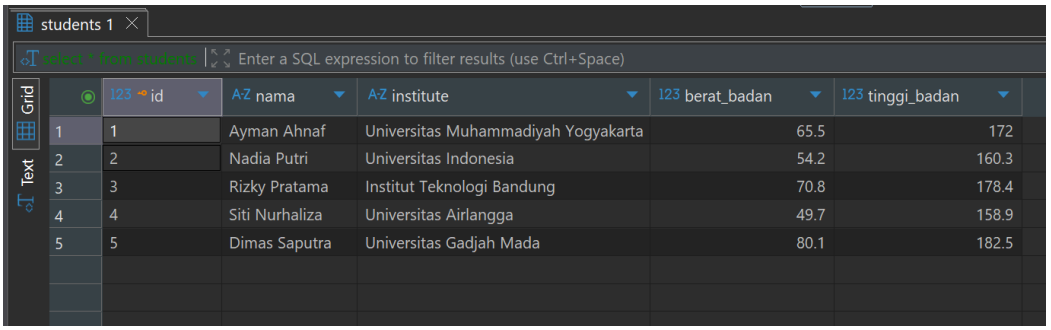
1.3. Masukkan minimal 5 data dengan nilai yang berbeda.

- Query:

```
-- Insert value
INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan)
VALUES
(1, 'Ayman Ahnaf', 'Universitas Muhammadiyah Yogyakarta', 65.5, 172.0),
```

```
(2, 'Nadia Putri', 'Universitas Indonesia', 54.2, 160.3),
(3, 'Rizky Pratama', 'Institut Teknologi Bandung', 70.8, 178.4),
(4, 'Siti Nurhaliza', 'Universitas Airlangga', 49.7, 158.9),
(5, 'Dimas Saputra', 'Universitas Gadjah Mada', 80.1, 182.5);
```

- Table:



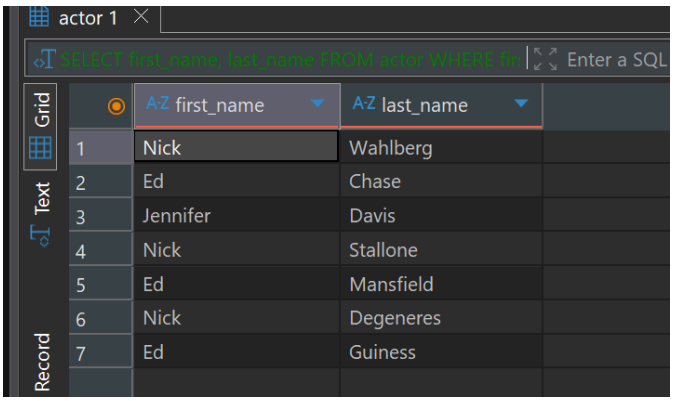
2. Query Data pada Skema `dvdrental`

2.1. Tampilkan `first_name` dan `last_name` dari aktor yang memiliki `first_name` "Jennifer", "Nick", atau "Ed".

- Query:

```
SELECT first_name, last_name
FROM actor
WHERE first_name IN ('Jennifer', 'Nick', 'Ed');
```

- Table:



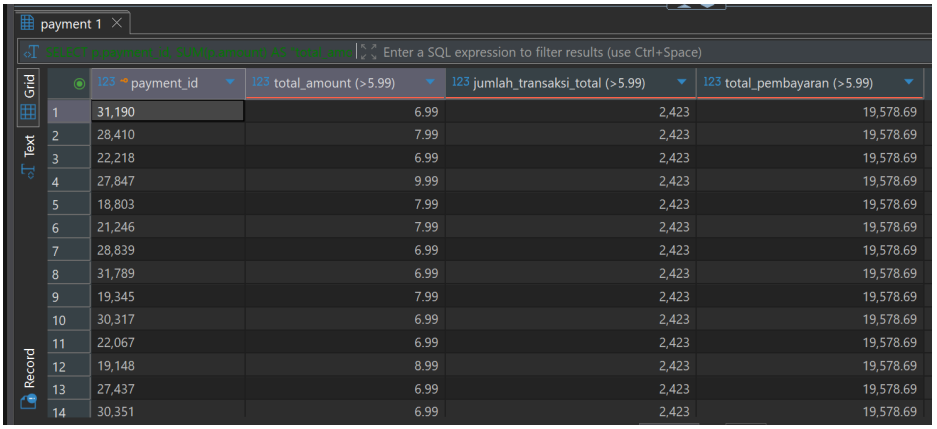
2.2. Hitung total pembayaran (`amount`) untuk setiap `payment_id` yang lebih besar dari 5.99.

- Query:

```
SELECT
    p.payment_id,
    SUM(p.amount) AS "total_amount (>5.99)",
    -- subquery untuk total keseluruhan
          (SELECT  COUNT(*)  FROM  payment  WHERE  amount  >  5.99)  AS
"jumlah_transaksi_total (>5.99)",
         (SELECT  SUM(amount)  FROM  payment  WHERE  amount  >  5.99)  AS
"total_pembayaran (>5.99)"
FROM payment p
```

```
WHERE p.amount > 5.99
GROUP BY p.payment_id;
```

- Table:



payment 1

Enter a SQL expression to filter results (use Ctrl+Space)

	payment_id	total_amount (>5.99)	jumlah_transaksi_total (>5.99)	total_pembayaran (>5.99)
1	31,190	6.99	2,423	19,578.69
2	28,410	7.99	2,423	19,578.69
3	22,218	6.99	2,423	19,578.69
4	27,847	9.99	2,423	19,578.69
5	18,803	7.99	2,423	19,578.69
6	21,246	7.99	2,423	19,578.69
7	28,839	6.99	2,423	19,578.69
8	31,789	6.99	2,423	19,578.69
9	19,345	7.99	2,423	19,578.69
10	30,317	6.99	2,423	19,578.69
11	22,067	6.99	2,423	19,578.69
12	19,148	8.99	2,423	19,578.69
13	27,437	6.99	2,423	19,578.69
14	30,351	6.99	2,423	19,578.69

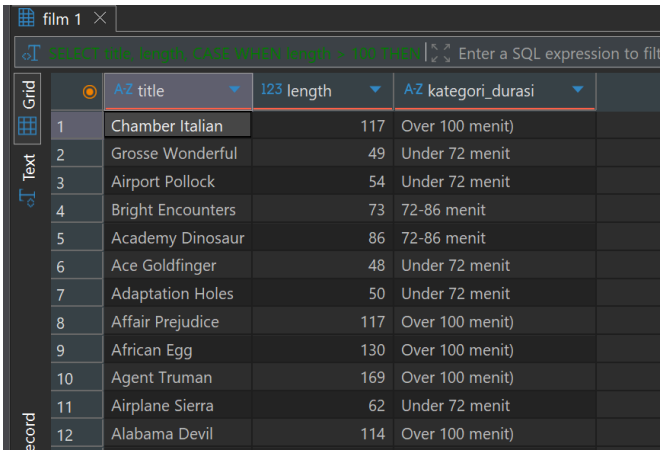
2.3. Kelompokkan film berdasarkan durasi menjadi 4 kategori:

Over 100 menit
87–100 menit
72–86 menit
Under 72 menit

- Query:

```
SELECT
    title,
    length,
    CASE
        WHEN length > 100 THEN 'Over 100 menit)'
        WHEN length BETWEEN 87 AND 100 THEN '(87-100 menit)'
        WHEN length BETWEEN 72 AND 86 THEN '72-86 menit'
        WHEN length < 72 THEN 'Under 72 menit'
        ELSE 'Unknown'
    END AS kategori_durasi
FROM film;
```

- Table:



film 1

Enter a SQL expression to fil

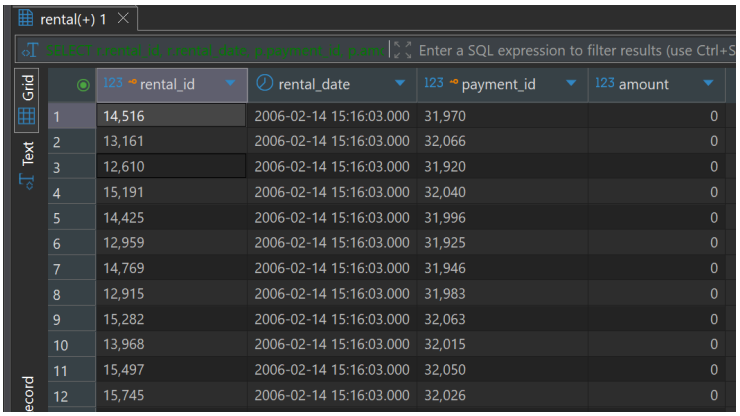
	title	length	kategori_durasi
1	Chamber Italian	117	Over 100 menit)
2	Grosse Wonderful	49	Under 72 menit
3	Airport Pollock	54	Under 72 menit
4	Bright Encounters	73	72-86 menit
5	Academy Dinosaur	86	72-86 menit
6	Ace Goldfinger	48	Under 72 menit
7	Adaptation Holes	50	Under 72 menit
8	Affair Prejudice	117	Over 100 menit)
9	African Egg	130	Over 100 menit)
10	Agent Truman	169	Over 100 menit)
11	Airplane Sierra	62	Under 72 menit
12	Alabama Devil	114	Over 100 menit)

2.4. Gabungkan data dari tabel `rental` dan `payment` untuk menampilkan `rental_id`, `rental_date`, `payment_id`, dan `amount`, urutkan berdasarkan `amount` secara ascending.

- Query:

```
SELECT
    r.rental_id,
    r.rental_date,
    p.payment_id,
    p.amount
FROM rental r
JOIN payment p ON r.rental_id = p.rental_id
ORDER BY p.amount ASC;
```

- Table:



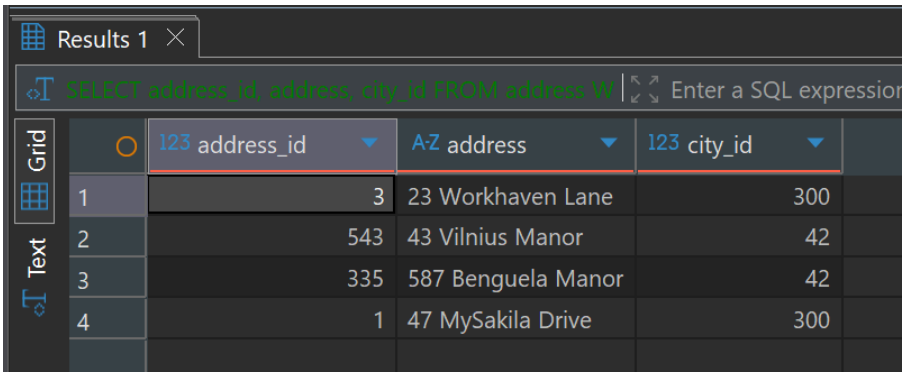
	rental_id	rental_date	payment_id	amount
1	14,516	2006-02-14 15:16:03.000	31,970	0
2	13,161	2006-02-14 15:16:03.000	32,066	0
3	12,610	2006-02-14 15:16:03.000	31,920	0
4	15,191	2006-02-14 15:16:03.000	32,040	0
5	14,425	2006-02-14 15:16:03.000	31,996	0
6	12,959	2006-02-14 15:16:03.000	31,925	0
7	14,769	2006-02-14 15:16:03.000	31,946	0
8	12,915	2006-02-14 15:16:03.000	31,983	0
9	15,282	2006-02-14 15:16:03.000	32,063	0
10	13,968	2006-02-14 15:16:03.000	32,015	0
11	15,497	2006-02-14 15:16:03.000	32,050	0
12	15,745	2006-02-14 15:16:03.000	32,026	0

2.5. Gunakan `UNION` untuk menggabungkan alamat (`address`) yang memiliki city_id = 42 dengan city_id = 300.

- Query:

```
SELECT
    address_id,
    address,
    city_id
FROM address
WHERE city_id = 42
UNION
SELECT
    address_id,
    address,
    city_id
FROM address
WHERE city_id = 300;
```

- Table:


Results 1
SELECT address_id, address, city_id FROM address W
Enter a SQL expressior
Grid
Text
123 address_id
A-Z address
123 city_id
1 3 23 Workhaven Lane 300
2 543 43 Vilnius Manor 42
3 335 587 Benguela Manor 42
4 1 47 MySakila Drive 300