

- Buatlah sebuah database dengan nama **dibimbng**.
- Di dalam database tersebut, buat sebuah tabel bernama **students** dalam skema **public**, dengan struktur kolom sebagai berikut:
 - id** (integer)
 - nama** (varchar)
 - institute** (varchar)
 - berat_badan** (float)
 - tinggi_badan** (float)

Isi tabel tersebut dengan minimal **5 baris data**, dan pastikan setiap baris memiliki nilai yang **berbeda-beda**. Nilai data dapat ditentukan secara bebas.

```
--create Table students
CREATE TABLE students(
    id INTEGER PRIMARY KEY,
    nama VARCHAR,
    institute VARCHAR,
    berat_badan FLOAT,
    tinggi_badan FLOAT
);

--data students
INSERT INTO public.students (id, nama, institute, berat_badan, tinggi_badan)
VALUES
(1, 'Joko kendil', 'Universitas Indonesia', 65.5, 172.0),
(2, 'Budi krempeng', 'Institut Teknologi Bandung', 72.3, 178.5),
(3, 'Otong surotong', 'Universitas Gadjah Mada', 54.8, 160.2),
(4, 'Eko ganteng', 'Universitas Diponegoro', 59.0, 165.4),
(5, 'Ucup surucup', 'Universitas Airlangga', 68.7, 170.0);
```

The screenshot shows the DBeaver interface with the following details:

- Left Panel (Object Navigator):**
 - Connections: DBeaver Sample Database (SQLite)
 - Databases: dibimbng (selected), postgres
 - Tables: students (under the selected database)
 - Foreign Tables: Foreign Tables
 - Views: Views
 - Materialized Views: Materialized Views
 - Indexes: Indexes
 - Functions: Functions
 - Sequences: Sequences
 - Data types: Data types
 - Aggregate functions: Aggregate functions
 - Event Triggers: Event Triggers
 - Extensions: Extensions
 - Storage: Storage
 - System Info: System Info
 - Roles: Roles
- Center Panel (SQL Editor):**

```
--create Table students
CREATE TABLE students(
    id INTEGER PRIMARY KEY,
    nama VARCHAR,
    institute VARCHAR,
    berat_badan FLOAT,
    tinggi_badan FLOAT
);

--data students
INSERT INTO public.students (id, nama, institute, berat_badan, tinggi_badan)
VALUES
(1, 'Joko kendil', 'Universitas Indonesia', 65.5, 172.0),
(2, 'Budi krempeng', 'Institut Teknologi Bandung', 72.3, 178.5),
(3, 'Otong surotong', 'Universitas Gadjah Mada', 54.8, 160.2),
(4, 'Eko ganteng', 'Universitas Diponegoro', 59.0, 165.4),
(5, 'Ucup surucup', 'Universitas Airlangga', 68.7, 170.0);
```
- Right Panel (Results Grid):**

	1	A-Z nama	A-Z institute	123 berat_badan	123 tinggi_badan
Grid	1	Joko kendil	Universitas Indonesia	65.5	172
2	2	Budi krempeng	Institut Teknologi Bandung	72.3	178.5
3	3	Otong surotong	Universitas Gadjah Mada	54.8	160.2
4	4	Eko ganteng	Universitas Diponegoro	59	165.4
5	5	Ucup surucup	Universitas Airlangga	68.7	170

Details at the bottom of the results panel:
 - Refresh, Save, Cancel, Export data, 200 rows.
 - 5 row(s) fetched - 0.001s, on 2025-08-04 at 20:42:27
 - WIB | en |

Part 2: Query on Existing Schema `dvdrental`

Gunakan skema `dvdrental` untuk menjawab pertanyaan berikut:

Tampilkan kolom `first_name` dan `last_name` dari tabel `actor` untuk aktor yang memiliki `first_name` **Jennifer, **Nick**, dan **Ed**.**

--Tampilkan kolom `first_name` dan `last_name` dari tabel `actor` untuk aktor
--yang memiliki `first_name` Jennifer, Nick, dan Ed.

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

The screenshot shows the MySQL Workbench interface. On the left is a tree view of the schema structure under the `dvdrental` database. The `Tables` section is expanded, showing tables like `actor`, `address`, `category`, etc. The `actor` table is selected. In the center, the SQL editor contains the query: `SELECT first_name, last_name FROM actor WHERE first_name IN ('Jennifer', 'Nick', 'Ed');`. To the right of the editor is a preview pane titled "actor 1" showing the results of the query:

	A-Z first_name	A-Z last_name
1	Nick	Wahlberg
2	Ed	Chase
3	Jennifer	Davis
4	Nick	Stallone
5	Ed	Mansfield
6	Nick	Degeneres
7	Ed	Guiness

Hitung total `amount` untuk setiap `payment_id` dari tabel `payment`, dan tampilkan hanya baris yang memiliki total `amount` lebih dari **5.99.**

Petunjuk: Gunakan klausula `HAVING`.

--Hitung total amount untuk setiap `payment_id` dari tabel `payment`,
--dan tampilkan hanya baris yang memiliki total amount lebih dari 5.99.

```
SELECT payment_id, SUM(amount) AS total_amount
FROM payment
GROUP BY payment_id
HAVING SUM(amount) > 5.99;
```

```
-- Hitung total amount untuk setiap payment_id dari tabel payment,
-- dan tampilkan hanya baris yang memiliki total amount lebih dari 5.99.

SELECT payment_id, SUM(amount) AS total_amount
FROM payment
GROUP BY payment_id
HAVING SUM(amount) > 5.99;
```

	payment_id	total_amount
1	18.803	7.99
2	31.789	6.99
3	25.886	10.99
4	28.031	9.99
5	23.154	7.99
6	18.474	7.99
7	26.032	9.99
8	25.571	7.00

Dari tabel `film`, tampilkan `film_id`, `title`, `description`, dan `length`.

Kelompokkan `length` ke dalam **4 kategori** sebagai berikut:

- Over 100
- Between 87 and 100
- Between 72 and 86
- Under 72

Penamaan kategori dapat ditentukan sendiri. Gunakan klausula `CASE WHEN` atau `BETWEEN`.

SELECT

```

    film_id,
    title,
    description,
    length,
CASE
    when length > 100 then 'Over 100'
    when length > 86 and length <= 100 then 'Between 87 and 100'
    when length > 71 and length <= 87 then 'Between 72 and 86'
    when length < 72 then 'Under 72'
end as length_category
from film;
```

```

-- KELUARPOKKEN TENGUN KE GULAM & KATEGORI SEORGAI DENTIKU:
SELECT
    film_id,
    title,
    description,
    length,
    CASE
        when length > 100 then 'Over 100'
        when length > 86 and length <= 100 then 'Between 87 and 100'
        when length > 71 and length <= 87 then 'Between 72 and 86'
        when length < 72 then 'Under 72'
    end as length_category
from film;

```

film_id	title	description	length	length_category
51	Balloon Homeward	A Insightful Panorama of a Forensic Psychologist And a Ma	75	Between 72 and 86
52	Ballroom Mockingbi	A Thrilling Documentary of a Composer And a Monkey whc	173	Over 100
53	Bang Kwai	A Epic Drama of a Madman And a Cat who must Face a A S	87	Between 87 and 100
54	Banger Pinocchio	A Awe-Inspiring Drama of a Car And a Pastry Chef who mu:	113	Over 100
55	Barbarella Streetcar	A Awe-Inspiring Story of a Feminist And a Cat who must Cc	65	Under 72
56	Barefoot Manchuria	A Intrepid Story of a Cat And a Student who must Vanquish	129	Over 100
57	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	99	Between 72 and 86
58	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
59	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
60	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
61	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
62	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
63	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
64	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
65	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
66	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86
67	Batman Returns	A Crime-Fighting Fantasy Of A Cat And A Student Who Must Vanquish	100	Between 72 and 86

Gabungkan tabel `rental` dan `payment` untuk menampilkan **10 baris pertama** dengan kolom:

- o `rental_id`
- o `rental_date`
- o `payment_id`
- o `amount`

Urutkan hasil berdasarkan `amount` secara **ascending**.

--Gabungkan tabel `rental` dan `payment` untuk menampilkan 10 baris pertama dengan kolom:
--`rental_id`, `rental_date`, `payment_id`, `amount`

```

SELECT
    r.rental_id,
    r.rental_date,
    p.payment_id,
    p.amount
FROM rental r
INNER JOIN payment p ON r.rental_id = p.rental_id
ORDER BY p.amount asc
LIMIT 10;

```

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view is expanded to show the 'public' schema, which contains various tables like actor, address, category, city, country, customer, departments, employees, film, film_actor, film_category, inventory, language, payment, rental, staff, and store. The 'Tables' section shows the size of each table in kilobytes. In the center, a SQL editor window displays a query to select rental_id, rental_date, payment_id, and amount from rental and payment tables, ordered by payment amount asc, with a limit of 10 rows. Below the editor is a results grid titled 'rental(+) 1' showing the first 10 rows of the query results.

Dari tabel `address`, gabungkan seluruh kolom dari alamat yang memiliki `city_id = 42` dan `city_id = 300` menggunakan `UNION`.

--Dari tabel address, gabungkan seluruh kolom dari alamat yang memiliki
--`city_id = 42` dan `city_id = 300` menggunakan UNION.

```
SELECT
    city_id
FROM
    address
WHERE
    city_id = 42
UNION
SELECT
    city_id
FROM
    address
WHERE
    city_id = 300
ORDER BY
    city_id;
```

The screenshot shows a database management interface with the following elements:

- Schemas:** A tree view showing the **public** schema containing tables, foreign tables, views, materialized views, indexes, functions, and procedures.
- Tables:** A list of tables with their sizes:
 - actor: 72K
 - address: 152K
 - category: 24K
 - city: 112K
 - country: 24K
 - customer: 208K
 - departments: 32K
 - employees: 32K
 - film: 936K
 - film_actor: 488K
 - film_category: 112K
 - inventory: 440K
 - language: 24K
 - payment: 1.8M
 - rental: 2.3M
 - staff: 32K
 - store: 40K
- Query Editor:** A code editor window displaying a multi-select query:

```
    city_id
FROM
    address
WHERE
    city_id = 42
UNION
SELECT
    city_id
FROM
    address
WHERE
    city_id = 300
ORDER BY
    city_id;
```
- Results Window:** A results grid titled "Results 1" showing the output of the query:

city_id
42
300