

1. Buatlah database bernama 'dibimbing'. Buat table dengan nama table 'students' di schema 'public' berisi kolom 'id'(int), 'nama' (varchar), 'institute'(varchar), 'berat_badan' (float), 'tinggi_badan' (float). Isi table tersebut minimal 5 data dengan value yang berbeda-beda. Value dibeaskan isinya

JAWABAN NO.1

-- Membuat tabel 'students' di schema 'public'

CREATE TABLE students (

id INT PRIMARY KEY,

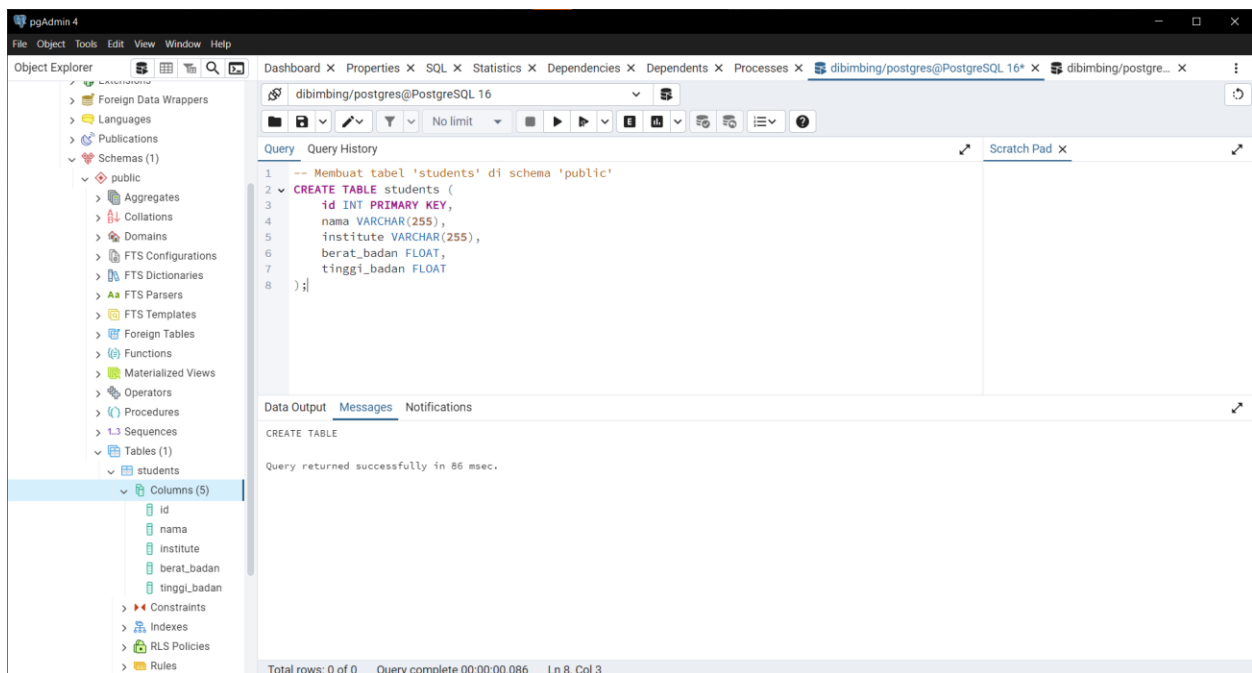
nama VARCHAR(255),

institute VARCHAR(255),

berat_badan FLOAT,

tinggi_badan FLOAT

);



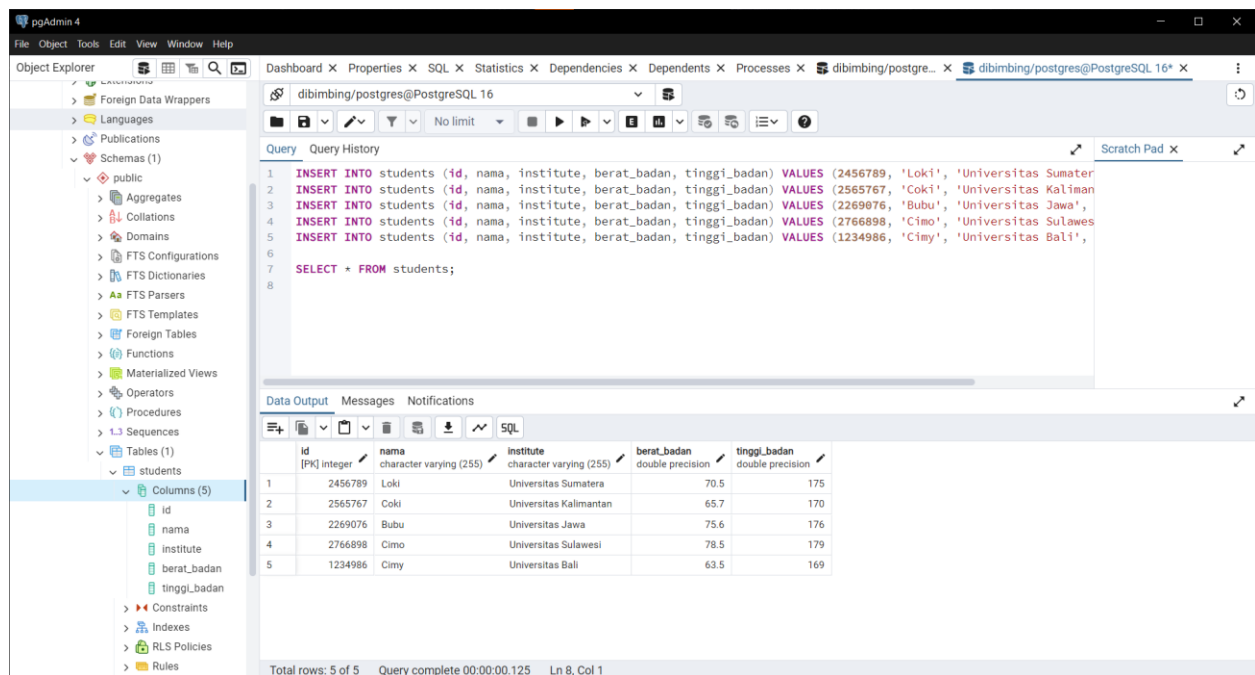
INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2456789, 'Loki', 'Universitas Sumatera', 70.5, 175.0);

INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2565767, 'Coki', 'Universitas Kalimantan', 65.7, 170.0);

INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2269076, 'Bubu', 'Universitas Jawa', 75.6, 176.0);

INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2766898, 'Cimo', 'Universitas Sulawesi', 78.5, 179.0);

INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (1234986, 'Cimy', 'Universitas Bali', 63.5, 169.0);



The screenshot displays the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure, including the 'students' table. The main pane shows the SQL query editor with the following queries:

```
1 INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2456789, 'Loki', 'Universitas Sumatera', 70.5, 175.0);
2 INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2565767, 'Coki', 'Universitas Kalimantan', 65.7, 170.0);
3 INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2269076, 'Bubu', 'Universitas Jawa', 75.6, 176.0);
4 INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (2766898, 'Cimo', 'Universitas Sulawesi', 78.5, 179.0);
5 INSERT INTO students (id, nama, institute, berat_badan, tinggi_badan) VALUES (1234986, 'Cimy', 'Universitas Bali', 63.5, 169.0);
6
7 SELECT * FROM students;
8
```

The Data Output pane shows the result of the SELECT query, displaying 5 rows of student data:

	id [PK] Integer	nama character varying (255)	institute character varying (255)	berat_badan double precision	tinggi_badan double precision
1	2456789	Loki	Universitas Sumatera	70.5	175
2	2565767	Coki	Universitas Kalimantan	65.7	170
3	2269076	Bubu	Universitas Jawa	75.6	176
4	2766898	Cimo	Universitas Sulawesi	78.5	179
5	1234986	Cimy	Universitas Bali	63.5	169

Total rows: 5 of 5 Query complete 00:00:00.125 Ln 8, Col 1

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Object Explorer' with the 'public' schema expanded, showing a table named 'students'. The main pane shows the 'Query' editor with a SQL statement that inserts 5 rows into the 'students' table. The 'Data Output' pane displays the result of the query, showing 5 rows of data.

id	nama	institute	berat_badan	tinggi_badan
2456789	Loki	Universitas Sumatera	70.5	175
2565767	Coki	Universitas Kalimantan	65.7	170
2269076	Bubu	Universitas Jawa	75.6	176
2766898	Cimo	Universitas Sulawesi	78.5	179
1234986	Cimy	Universitas Bali	63.5	169

2. Tunjukkan first_name dan last_name actor yang memiliki first_name Jennifer, Nick, dan Ed

JAWABAN NO. 2

SELECT first_name, last_name

FROM actor

WHERE first_name IN ('Jennifer', 'Nick', 'Ed');

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Object Explorer' with the 'public' schema expanded, showing a table named 'actor'. The main pane shows the 'Query' editor with a SQL statement that selects the first_name and last_name of actors whose first_name is Jennifer, Nick, or Ed. The 'Data Output' pane displays the result of the query, showing 7 rows of data.

first_name	last_name
Nick	Wahlberg
Ed	Chase
Jennifer	Davis
Nick	Stallone
Ed	Mansfield
Nick	Degeneres
Ed	Guinness

3. Hitung Total Amount untuk setiap payment_id yang Total Amount-nya lebih dari 5.99 (hint: menggunakan HAVING)

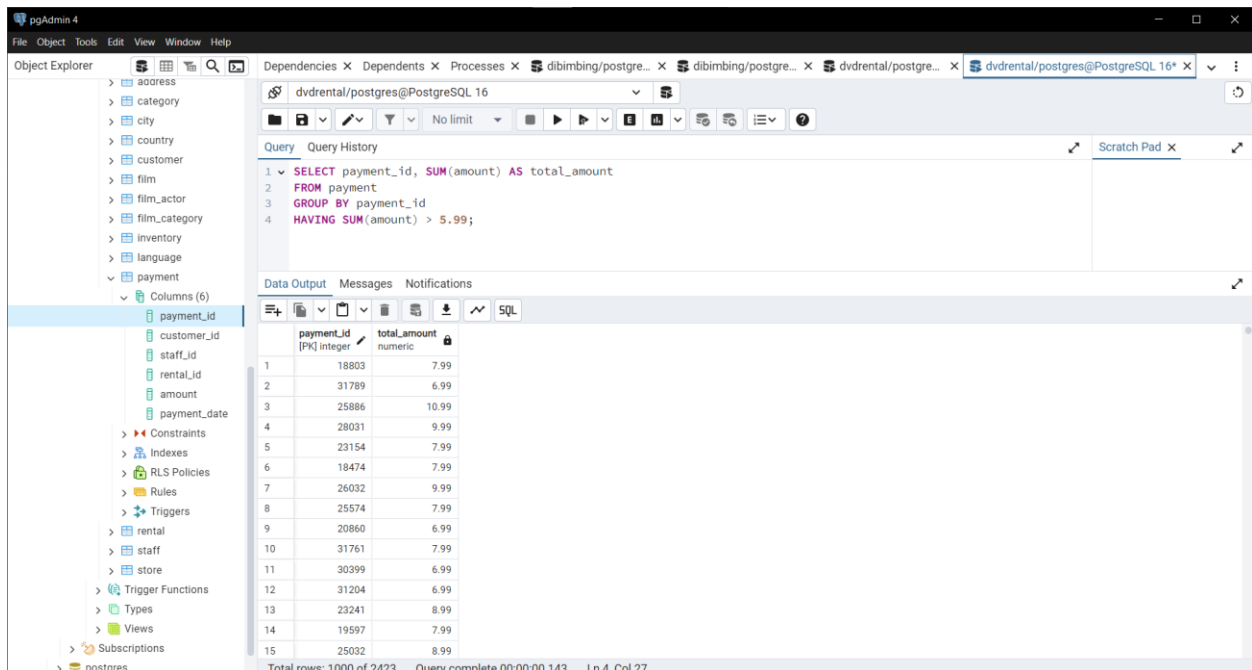
JAWABAN NO.3

```
SELECT payment_id, SUM(amount) AS total_amount
```

```
FROM payment
```

```
GROUP BY payment_id
```

```
HAVING SUM(amount) > 5.99;
```



The screenshot shows the pgAdmin 4 interface. On the left, the 'Object Explorer' pane displays the database structure, with the 'payment' table selected under the 'Columns (6)' section. The main pane shows a SQL query in the 'Query' tab:

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

Below the query, the 'Data Output' pane displays the results of the query in a table format:

payment_id [PK] integer	total_amount numeric
1	18803
2	31789
3	25886
4	28031
5	23154
6	18474
7	26032
8	25574
9	20860
10	31761
11	30399
12	31204
13	23241
14	19597
15	25032

The status bar at the bottom indicates 'Total rows: 1000 of 2423' and 'Query complete 00:00:00.143 Ln 4, Col 27'.

4. Tampilkan film.id, film.title, film.description and film_length. Kelompokkan film.length ke dalam 4 categories(over 100, 87-100, 72-86 and under 72). Penamaan kelompok dibebaskan

JAWABAN NO. 4

```
SELECT
```

```
film.film_id,
```

```
film.title,
```

```
film.description,
```

```
film.length,
```

```
CASE
```

WHEN film.length > 100 THEN 'Over 100 Minutes'

WHEN film.length BETWEEN 87 AND 100 THEN '87-100 Minutes'

WHEN film.length BETWEEN 72 AND 86 THEN '72-86 Minutes'

ELSE 'Under 72 Minutes'

END AS length_category

FROM film;

The screenshot shows the pgAdmin 4 interface. The left pane displays the 'film' table under the 'dvdrental/postgres@PostgreSQL 16' database. The main pane shows a SQL query with a CASE statement that categorizes film lengths. The 'Data Output' tab displays the results of the query.

film_id	title	description	length	length_category
1	133 Chamber Italian	A Fateful Reflection of a Moose And a Husband who must Overcome a Monkey in Nigeria	117	Over 100
2	384 Grosse Wonderful	A Epic Drama of a Cat And a Explorer who must Redeem a Moose in Australia	49	Under 72
3	8 Airport Pollock	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India	54	Under 72
4	98 Bright Encounters	A Fateful Yam of a Lumberjack And a Feminist who must Conquer a Student in A Jet Boat	73	72-86 Mi
5	1 Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies	86	72-86 Mi
6	2 Ace Goldfinger	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China	48	Under 72
7	3 Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factory	50	Under 72
8	4 Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank	117	Over 100
9	5 African Egg	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in The Gulf of Mexico	130	Over 100

Total rows: 1000 of 1000 Query complete 00:00:00.174 Ln 12, Col 11

The screenshot shows the pgAdmin 4 interface. The left pane displays the 'film' table under the 'dvdrental/postgres@PostgreSQL 16' database. The main pane shows a SQL query with a CASE statement that categorizes film lengths. The 'Data Output' tab displays the results of the query.

film_id	title	description	length	length_category
1	133 Chamber Italian	A Fateful Reflection of a Moose And a Husband who must Overcome a Monkey in Nigeria	117	Over 100 Minutes
2	384 Grosse Wonderful	A Epic Drama of a Cat And a Explorer who must Redeem a Moose in Australia	49	Under 72 Minutes
3	8 Airport Pollock	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India	54	Under 72 Minutes
4	98 Bright Encounters	A Fateful Yam of a Lumberjack And a Feminist who must Conquer a Student in A Jet Boat	73	72-86 Minutes
5	1 Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies	86	72-86 Minutes
6	2 Ace Goldfinger	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China	48	Under 72 Minutes
7	3 Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factory	50	Under 72 Minutes
8	4 Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank	117	Over 100 Minutes
9	5 African Egg	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in The Gulf of Mexico	130	Over 100 Minutes
10	6 Agent Truman	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient China	169	Over 100 Minutes
11	7 Airplane Sierra	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat	62	Under 72 Minutes
12	9 Alabama Devil	A Thoughtful Panorama of a Database Administrator And a Mad Scientist who must Outgun a Mad Scientist in A Jet Boat	114	Over 100 Minutes
13	10 Aladdin Calendar	A Action-Packed Tale of a Man And a Lumberjack who must Reach a Feminist in Ancient China	63	Under 72 Minutes
14	11 Alamo Videotape	A Boring Epistle of a Butler And a Cat who must Fight a Pastry Chef in A MySQL Convention	126	Over 100 Minutes
15	12 Alaska Phantom	A Fanciful Saga of a Hunter And a Pastry Chef who must Vanquish a Boy in Australia	136	Over 100 Minutes
16	213 Date Speed	A Touching Saga of a Composer And a Moose who must Discover a Dentist in A MySQL Convention	104	Over 100 Minutes
17	13 Ali Forever	A Action-Packed Drama of a Dentist And a Crocodile who must Battle a Feminist in The Canadian Rockies	150	Over 100 Minutes
18	14 Alice Fantasia	A Emotional Drama of a A Shark And a Database Administrator who must Vanquish a Pioneer in Soviet Georgia	94	87-100 Minutes
19	15 Alien Center	A Brilliant Drama of a Cat And a Mad Scientist who must Battle a Feminist in A MySQL Convention	46	Under 72 Minutes
20	16 Alley Evolution	A Fast-Paced Drama of a Robot And a Composer who must Battle a Astronaut in New Orleans	180	Over 100 Minutes
21	17 Alone Trip	A Fast-Paced Character Study of a Composer And a Dog who must Outgun a Boat in An Abandoned Fun House	82	72-86 Minutes

Total rows: 1000 of 1000 Query complete 00:00:00.174 Ln 12, Col 11

5. Dari tabel rental dan payment, tunjukkan 10 baris rental_id, rental_date, payment_id, dan amount. Ordered by amount in ascending order.

JAWABAN NO.5

SELECT

rental.rental_id,

rental.rental_date,

payment.payment_id,

payment.amount

FROM rental

JOIN payment ON rental.rental_id = payment.rental_id

ORDER BY payment.amount ASC

LIMIT 10;

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, with the 'rental' table selected under the 'Columns (?)' section. The main query editor displays the following SQL query:

```
1 SELECT
2     rental.rental_id,
3     rental.rental_date,
4     payment.payment_id,
5     payment.amount
6 FROM rental
7 JOIN payment ON rental.rental_id = payment.rental_id
8 ORDER BY payment.amount ASC
9 LIMIT 10;
```

The Data Output pane at the bottom shows the results of the query, displaying 10 rows of data. The columns are rentalId (integer), rentalDate (timestamp without time zone), paymentId (integer), and amount (numeric (5,2)).

	rentalId	rentalDate	paymentId	amount
1	13577	2006-02-14 15:16:03	31966	0.00
2	14425	2006-02-14 15:16:03	31996	0.00
3	12959	2006-02-14 15:16:03	31925	0.00
4	14769	2006-02-14 15:16:03	31946	0.00
5	14516	2006-02-14 15:16:03	31970	0.00
6	12915	2006-02-14 15:16:03	31983	0.00
7	13713	2006-02-14 15:16:03	31918	0.00
8	12610	2006-02-14 15:16:03	31920	0.00
9	11782	2006-02-14 15:16:03	31942	0.00
10	13464	2006-02-14 15:16:03	32001	0.00

Total rows: 10 of 10 Query complete 00:00:00.105 Ln 9, Col 10

6. Gabungkan address (seluruh kolom) yang memiliki city_id = 42 dengan city_id=300. Gunakan UNION

JAWABAN NO.6

SELECT *

FROM address

WHERE city_id = 42

UNION

SELECT *

FROM address

WHERE city_id = 300;

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, with the 'city' table selected under the 'address' schema. The main query editor displays the following SQL query:

```
1 SELECT *
2 FROM address
3 WHERE city_id = 42
4
5 UNION
6
7 SELECT *
8 FROM address
9 WHERE city_id = 300;
```

Below the query editor, the Data Output tab shows the results of the query. The results are displayed in a table with the following columns: address_id, address, address2, district, city_id, postal_code, phone, and last_update. The table contains 4 rows of data.

address_id	address	address2	district	city_id	postal_code	phone	last_update
1	47 MySakila Drive	[null]	Alberta	300			2006-02-15 09:45:30
2	335 587 Benguela Manor		Illinois	42	91590	165450987037	2006-02-15 09:45:30
3	543 43 Vilnius Manor		Colorado	42	79814	484500282381	2006-02-15 09:45:30
4	23 Workhaven Lane	[null]	Alberta	300		14033335568	2006-02-15 09:45:30

Total rows: 4 of 4 Query complete 00:00:00.105 Ln 9, Col 21