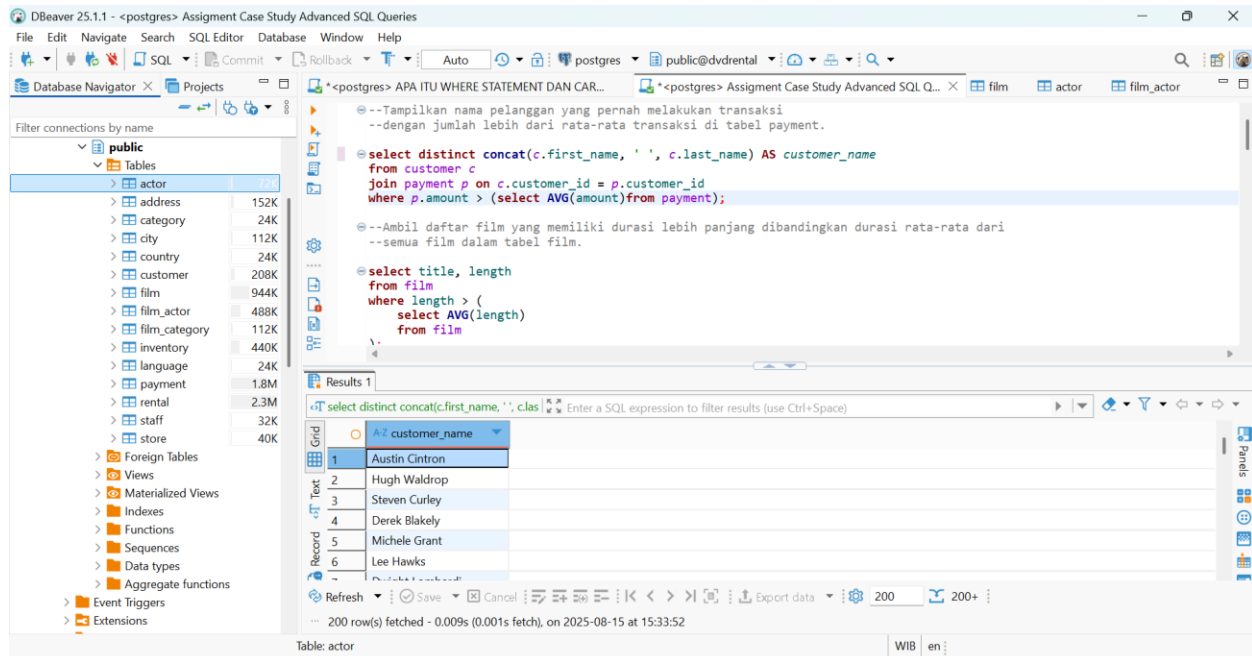


```
--Tampilkan nama pelanggan yang pernah melakukan transaksi
--dengan jumlah lebih dari rata-rata transaksi di tabel payment.
select distinct concat(c.first_name, ' ', c.last_name) AS customer_name
from customer c
join payment p on c.customer_id = p.customer_id
where p.amount > (select AVG(amount) from payment);
```



```
--Ambil daftar film yang memiliki durasi lebih panjang dibandingkan durasi rata-rata
dari
--semua film dalam tabel film.
```

```
select title, length
from film
where length > (
    select AVG(length)
    from film
);
```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator: public Tables (actor, address, category, city, country, customer, film, film\_actor, film\_category, inventory, language, payment, rental, staff, store)

SQL Editor:

```

where p.amount > (select AVG(amount) from payment);

--Ambil daftar film yang memiliki durasi lebih panjang dibandingkan durasi rata-rata dari
--semua film dalam tabel film.

select title, length
from film
where length > (
    select AVG(length)
    from film
);

--Buat query untuk menampilkan aktor yang hanya membintangi satu film dalam database.
select distinct concat(a.first_name, ' ', a.last_name) as actor_name
from actor a
join film_actor fa on a.actor_id = fa.actor_id

```

Results 1 film 2

	title	length
1	Chamber Italian	117
2	Affair Prejudice	117
3	African Egg	130
4	Agent Truman	169
5	Alamo Videotape	126
6	Alaska Phantom	136
7	Ali Forever	150

--Buat query untuk menampilkan aktor yang hanya membintangi satu film dalam database.

```

select distinct concat(a.first_name, ' ', a.last_name) as actor_name
from actor a
join film_actor fa on a.actor_id = fa.actor_id
group by a.actor_id, a.first_name, a.last_name
having count(distinct fa.film_id) = 1;

```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator: public Tables (actor, address, category, city, country, customer, film, film\_actor, film\_category, inventory, language, payment, rental, staff, store)

SQL Editor:

```

from film
where length > (
    select AVG(length)
    from film
);

--Buat query untuk menampilkan aktor yang hanya membintangi satu film dalam database.
select distinct concat(a.first_name, ' ', a.last_name) as actor_name
from actor a
join film_actor fa on a.actor_id = fa.actor_id
group by a.actor_id, a.first_name, a.last_name
having count(distinct fa.film_id) = 1;

--Gunakan RANK() untuk menentukan peringkat film berdasarkan rental_rate.
select title,
       rental_rate,
       RANK() over (order by rental_rate desc) as film_rank

```

Results 1 film 2 Results 3 Results 4

Results 3: select distinct concat(a.first\_name, ' ', a.last\_name) as actor\_name

	actor_name
--	------------

--Gunakan RANK() untuk menentukan peringkat film berdasarkan rental\_rate.

```

select title,
       rental_rate,
       RANK() over (order by rental_rate desc) as film_rank
from film;

```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator

Filter connections by name

- public
  - Tables
    - actor 72K
    - address 152K
    - category 24K
    - city 112K
    - country 24K
    - customer 208K
    - film 944K
    - film\_actor 488K
    - film\_category 112K
    - inventory 440K
    - language 24K
    - payment 1.8M
    - rental 2.3M
    - staff 32K
    - store 40K
  - Foreign Tables
  - Views
  - Materialized Views
  - Indexes
  - Functions
  - Sequences
  - Data types
  - Aggregate functions
  - Event Triggers
  - Extensions

SQL Editor

```

join film_actor fa on a.actor_id = fa.actor_id
group by a.actor_id, a.first_name, a.last_name
having count(distinct fa.film_id) = 1;

--Gunakan RANK() untuk menentukan peringkat film berdasarkan rental_rate.
select title,
       rental_rate,
       RANK() over (order by rental_rate desc) as film_rank
from film;

--Gunakan DENSE_RANK() untuk menentukan peringkat pelanggan
--berdasarkan total transaksi yang mereka lakukan.
select concat(c.first_name, ' ', c.last_name) as customer_name,
       SUM(p.amount) as total_transaksi,
       DENSE_RANK() over (order by SUM(p.amount) desc) as customer_rank
from customer c
join payment p on c.customer_id = p.customer_id

```

Results 1 film 2 Results 3 Results 4 film 5

select title, rental\_rate, RANK() over (orde

	title	rental_rate	film_rank
1	French Holiday	4.99	1
2	Bucket Brotherhood	4.99	1
3	Frisco Forrest	4.99	1
4	Prejudice Oleander	4.99	1
5	Frontier Cabin	4.99	1
6	Poseidon Forever	4.99	1
7	Fugitive Maguire	4.99	1
8	Whispering Storm	4.00	1

WIB en Writable Smart Insert 30:11:1035

```

--Gunakan DENSE_RANK() untuk menentukan peringkat pelanggan
--berdasarkan total transaksi yang mereka lakukan.
select concat (c.first_name, ' ', c.last_name) as customer_name,
       SUM(p.amount) as total_transaksi,
       DENSE_RANK() over (order by SUM(p.amount) desc) as customer_rank
from customer c
join payment p on c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name;

```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator

Filter connections by name

- public
  - Tables
    - actor 72K
    - address 152K
    - category 24K
    - city 112K
    - country 24K
    - customer 208K
    - film 944K
    - film\_actor 488K
    - film\_category 112K
    - inventory 440K
    - language 24K
    - payment 1.8M
    - rental 2.3M
    - staff 32K
    - store 40K
  - Foreign Tables
  - Views
  - Materialized Views
  - Indexes
  - Functions
  - Sequences
  - Data types
  - Aggregate functions
  - Event Triggers
  - Extensions

SQL Editor

```

from film;

--Gunakan DENSE_RANK() untuk menentukan peringkat pelanggan
--berdasarkan total transaksi yang mereka lakukan.
select concat(c.first_name, ' ', c.last_name) as customer_name,
       SUM(p.amount) as total_transaksi,
       DENSE_RANK() over (order by SUM(p.amount) desc) as customer_rank
from customer c
join payment p on c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name;

--Gunakan ROW_NUMBER() untuk memberikan nomor urut
--pada daftar film berdasarkan release_year.
select title,
       release_year,
       row_number() over (order by release_year desc) as nomor_urut
from film;

```

Results 1 film 2 Results 3 Results 4 film 5 Results 6

select concat (cfirst\_name, ' ', clast\_name)

	customer_name	total_transaksi	customer_rank
1	Eleanor Hunt	211.55	1
2	Karl Seal	208.58	2
3	Marion Snyder	194.61	3
4	Rhonda Kennedy	191.62	4
5	Clara Shaw	189.6	5
6	Tommy Collazo	183.63	6
7	Ana Bradley	167.67	7
8	Charlie Lebo	167.67	8

WIB en Writable Smart Insert 39:51:1450

```
--Gunakan ROW_NUMBER() untuk memberikan nomor urut
--pada daftar film berdasarkan release_year.
select title,
       release_year,
       row_number() over (order by release_year desc) as nomor_urut
from film;
```

The screenshot shows the DBeaver 25.1.1 interface. The left sidebar displays the database schema for 'public', including tables like actor, address, category, city, country, customer, film, film\_actor, film\_category, inventory, language, payment, rental, staff, and store. The main editor window contains two SQL queries. The first query uses ROW\_NUMBER() to rank films by release\_year. The second query uses a CTE named 'customer\_trans' to identify customers with more than 10 transactions. The bottom panel shows the results of the first query, displaying a table with columns 'title', 'release\_year', and 'nomor\_urut'.

	title	release_year	nomor_urut
1	Chamber Italian	2.006	1
2	Grosse Wonderful	2.006	2
3	Airport Pollock	2.006	3
4	Bright Encounters	2.006	4
5	Academy Dinosaur	2.006	5
6	Ace Goldfinger	2.006	6

--Gunakan CTE untuk membuat daftar pelanggan yang melakukan transaksi lebih dari 10 kali.

```
with customer_trans as (
    select customer_id,
           count(payment_id) as jumlah_transaksi
    from payment
    group by customer_id
)
select concat(c.first_name, ' ', c.last_name) as customer_name,
       ct.jumlah_transaksi
from customer_trans ct
join customer c on ct.customer_id = c.customer_id
where ct.jumlah_transaksi > 10
order by ct.jumlah_transaksi desc;
```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator

- public
  - actor (72K)
  - address (152K)
  - category (24K)
  - city (112K)
  - country (24K)
  - customer (208K)
  - film (944K)
  - film\_actor (488K)
  - film\_category (112K)
  - inventory (440K)
  - language (24K)
  - payment (1.8M)
  - rental (2.3M)
  - staff (32K)
  - store (40K)
- Foreign Tables
- Views
- Materialized Views
- Indexes
- Functions
- Sequences
- Data types
- Aggregate functions
- Event Triggers
- Extensions

SQL Editor

```

--Gunakan CTE untuk membuat daftar pelanggan yang melakukan transaksi lebih dari 10 kali.
with customer_trans as (
  select customer_id,
         count(payment_id) as jumlah_transaksi
  from payment
  group by customer_id
)
select concat(c.first_name, ' ', c.last_name) as customer_name,
       ct.jumlah_transaksi
from customer_trans ct
join customer c on ct.customer_id = c.customer_id
where ct.jumlah_transaksi > 10
order by ct.jumlah_transaksi desc;

--Gunakan CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak.

```

Results 1 film 2 Results 3 Results 4 film 5 Results 6 film 7 Results 8

with customer\_trans as (select customer\_id, count(payment\_id) as jumlah\_transaksi from payment group by customer\_id where customer\_trans.jumlah\_transaksi > 10 order by customer\_trans.jumlah\_transaksi desc)

	customer_name	jumlah_transaksi
1	Eleanor Hunt	45
2	Karl Seal	42
3	Clara Shaw	40
4	Tammy Sanders	39
5	Marcia Dean	39
6	Marion Snyder	39

Refresh Save Cancel Export data 200 597

200 row(s) fetched - 0.001s, on 2025-08-15 at 15:44:13

WIB en Writable Smart Insert 60:35:2146

--Gunakan CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak.

```

with film_rental_count as (
  select f.film_id,
         f.title,
         count(r.rental_id) as jumlah_rental
  from film f
  join inventory i on f.film_id = i.film_id
  join rental r on i.inventory_id = r.inventory_id
  group by f.film_id, f.title
)
select title,
       jumlah_rental
from film_rental_count
order by jumlah_rental desc;

```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator

- public
  - actor (72K)
  - address (152K)
  - category (24K)
  - city (112K)
  - country (24K)
  - customer (208K)
  - film (944K)
  - film\_actor (488K)
  - film\_category (112K)
  - inventory (440K)
  - language (24K)
  - payment (1.8M)
  - rental (2.3M)
  - staff (32K)
  - store (40K)
- Foreign Tables
- Views
- Materialized Views
- Indexes
- Functions
- Sequences
- Data types
- Aggregate functions
- Event Triggers
- Extensions

SQL Editor

```
--Gunakan CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak.
with film_rental_count as (
  select f.film_id,
         f.title,
         count(r.rental_id) as jumlah_rental
  from film f
  join inventory i on f.film_id = i.film_id
  join rental r on i.inventory_id = r.inventory_id
  group by f.film_id, f.title
)
select title,
       jumlah_rental
from film_rental_count
order by jumlah_rental desc;
```

Results 1 film 2 Results 3 Results 4 film 5 Results 6 film 7 Results 8 film 9

	Az title	jumlah_rental
1	Bucket Brotherhood	34
2	Rocketeer Mother	33
3	Grit Clockwork	32
4	Forward Temple	32
5	Ridgemont Submarine	32
6	Juggler Hardy	32
7	Scalawag Duck	32
7 rows		21

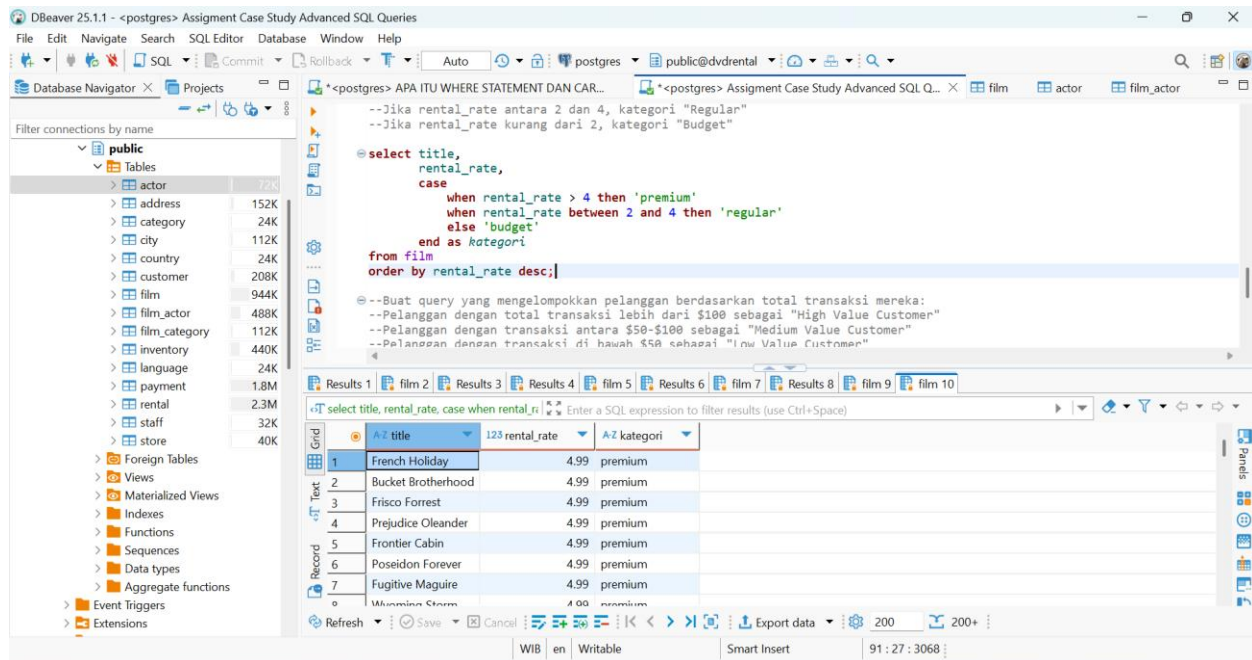
Refresh Save Cancel Export data 200 200+

WIB en Writable Smart Insert 75 : 29 : 2591

```
--Buat query yang mengelompokkan film berdasarkan rental_rate:
--Jika rental_rate lebih dari 4, kategori "Premium"
--Jika rental_rate antara 2 dan 4, kategori "Regular"
--Jika rental_rate kurang dari 2, kategori "Budget"
```

```
select title,
       rental_rate,
       case
         when rental_rate > 4 then 'premium'
         when rental_rate between 2 and 4 then 'regular'
         else 'budget'
       end as kategori
from film
order by rental_rate desc;
```





```
--Buat query yang mengelompokkan pelanggan berdasarkan total transaksi mereka:
--Pelanggan dengan total transaksi lebih dari $100 sebagai "High Value Customer"
--Pelanggan dengan transaksi antara $50-$100 sebagai "Medium Value Customer"
--Pelanggan dengan transaksi di bawah $50 sebagai "Low Value Customer"
```

```
select concat (c.first_name, ' ', c.last_name) as customer_name,
       sum(p.amount) as total_transaksi,
       case
         when sum(p.amount) > 100 then 'high value customer'
         when sum(p.amount) between 50 and 100 then 'medium value customer'
         else 'low value customer'
       end as kategori
from customer c
join payment p on c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name
order by total_transaksi desc;
```

DBeaver 25.1.1 - <postgres> Assignment Case Study Advanced SQL Queries

File Edit Navigate Search SQL Editor Database Window Help

Auto postgres public@dvdrental

Database Navigator Projects

Filter connections by name

- public
  - Tables
    - actor 72K
    - address 152K
    - category 24K
    - city 112K
    - country 24K
    - customer 208K
    - film 944K
    - film\_actor 488K
    - film\_category 112K
    - inventory 440K
    - language 24K
    - payment 1.8M
    - rental 2.3M
    - staff 32K
    - store 40K
  - Foreign Tables
  - Views
  - Materialized Views
  - Indexes
  - Functions
  - Sequences
  - Data types
  - Aggregate functions
  - Event Triggers
  - Extensions

```
--Pelanggan dengan transaksi di bawah $50 sebagai "Low Value Customer"
select concat(c.first_name, ' ', c.last_name) as customer_name,
       sum(p.amount) as total_transaksi,
       case
         when sum(p.amount) > 100 then 'high value customer'
         when sum(p.amount) between 50 and 100 then 'medium value customer'
         else 'low value customer'
       end as kategori
from customer c
join payment p on c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name
order by total_transaksi desc;
```

Results 1 film 2 Results 3 Results 4 film 5 Results 6 film 7 Results 8 film 9 film 10 Results 11

select concat(c.first\_name, ' ', c.last\_name) as customer\_name, sum(p.amount) as total\_transaksi, case when sum(p.amount) > 100 then 'high value customer' when sum(p.amount) between 50 and 100 then 'medium value customer' else 'low value customer' end as kategori

	Az customer_name	123 total_transaksi	Az kategori
1	Eleanor Hunt	211.55	high value customer
2	Karl Seal	208.58	high value customer
3	Marion Snyder	194.61	high value customer
4	Rhonda Kennedy	191.62	high value customer
5	Clara Shaw	189.6	high value customer
6	Tommy Collazo	183.63	high value customer

Refresh Save Cancel Export data 200 200+ ...

200 row(s) fetched - 0.005s, on 2025-08-15 at 15:47:08

WIB en Writable Smart Insert 107 : ... 3857 ...