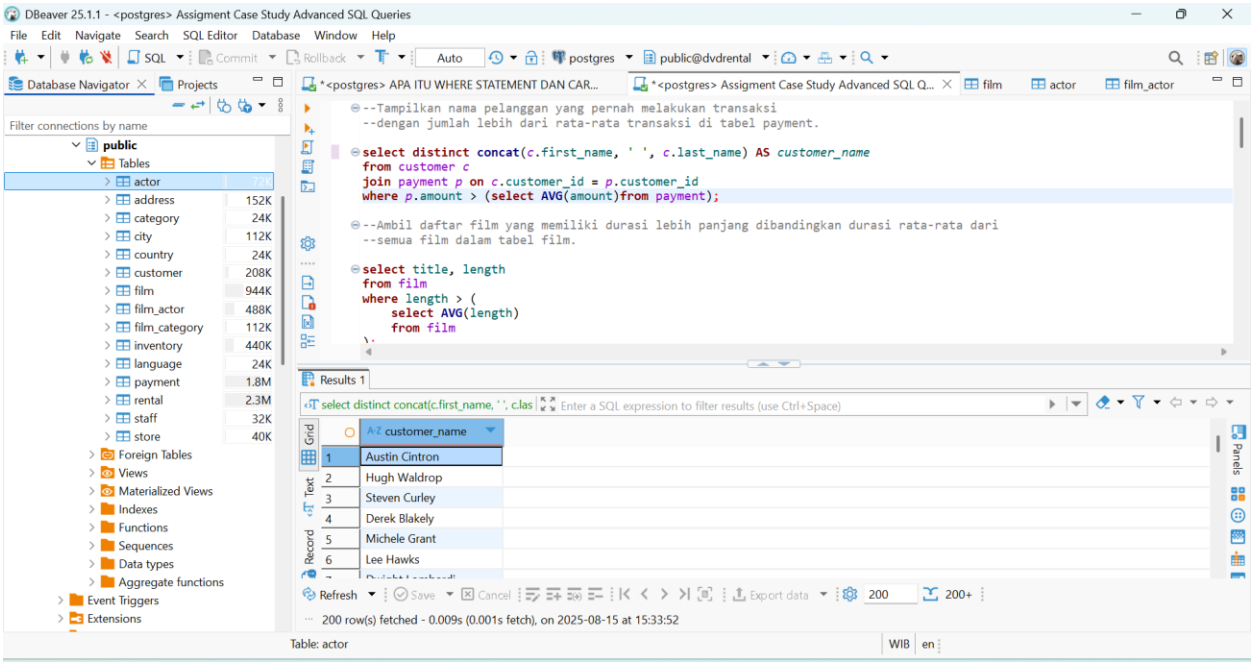
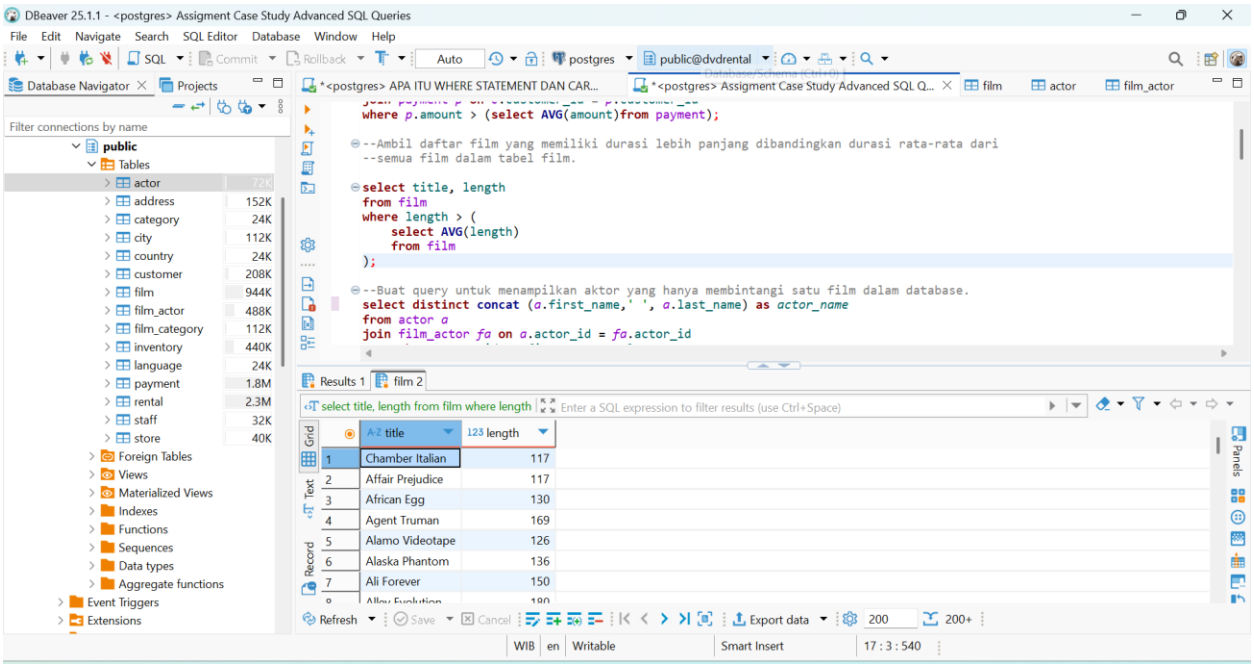
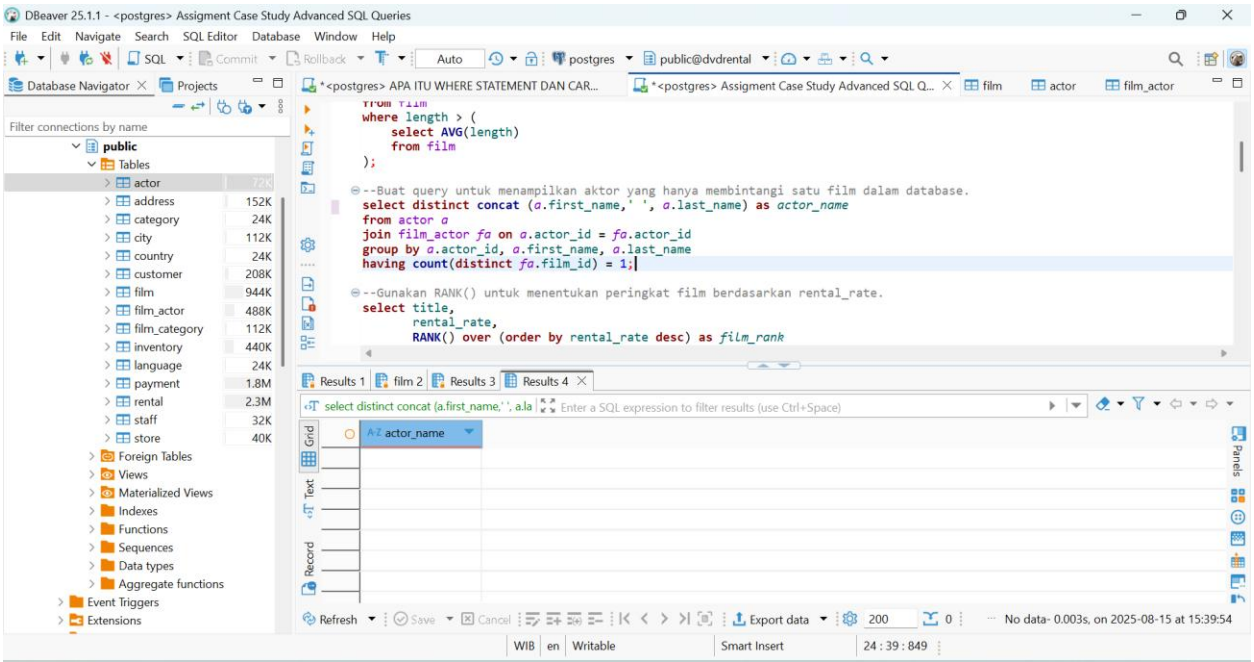
```sql
--Tampilkan nama pelanggan yang pernah melakukan transaksi
--dengan jumlah lebih dari rata-rata transaksi di tabel payment.
select distinct concat(c.first_name, ' ', c.last_name) AS customer_name
from customer c
join payment p on c.customer_id = p.customer_id
where p.amount > (select AVG(amount)from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



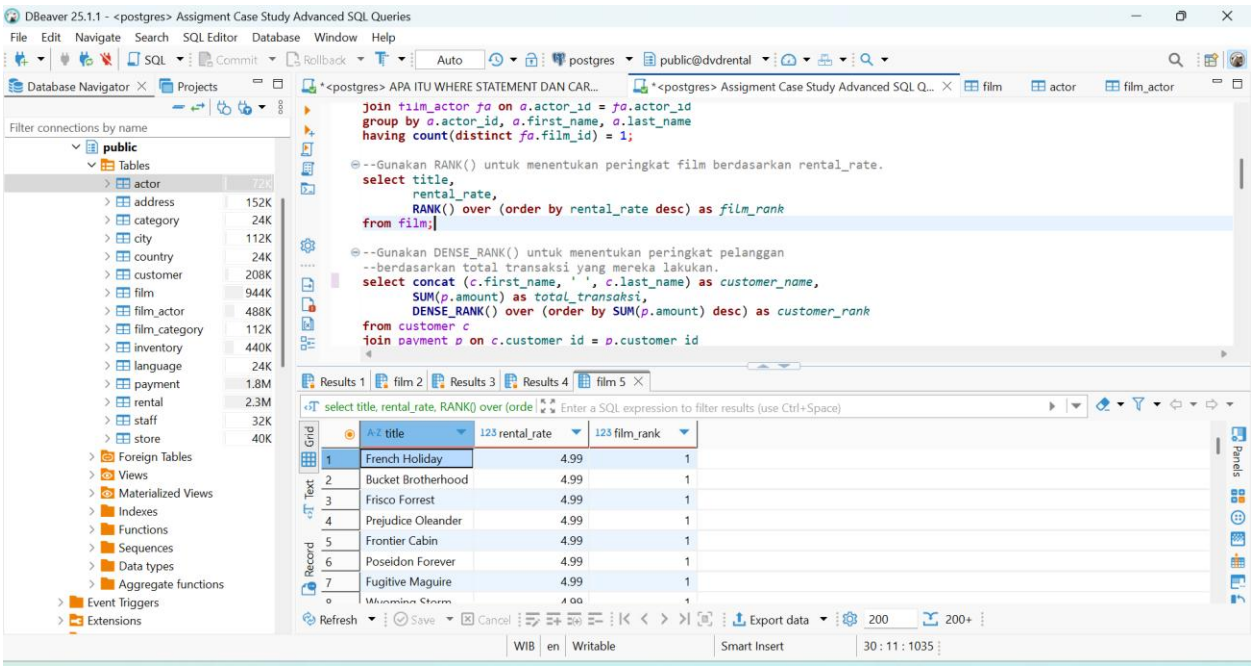

```sql
--Ambil daftar film yang memiliki durasi lebih panjang dibandingkan durasi rata-rata dari
--semua film dalam tabel film.

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



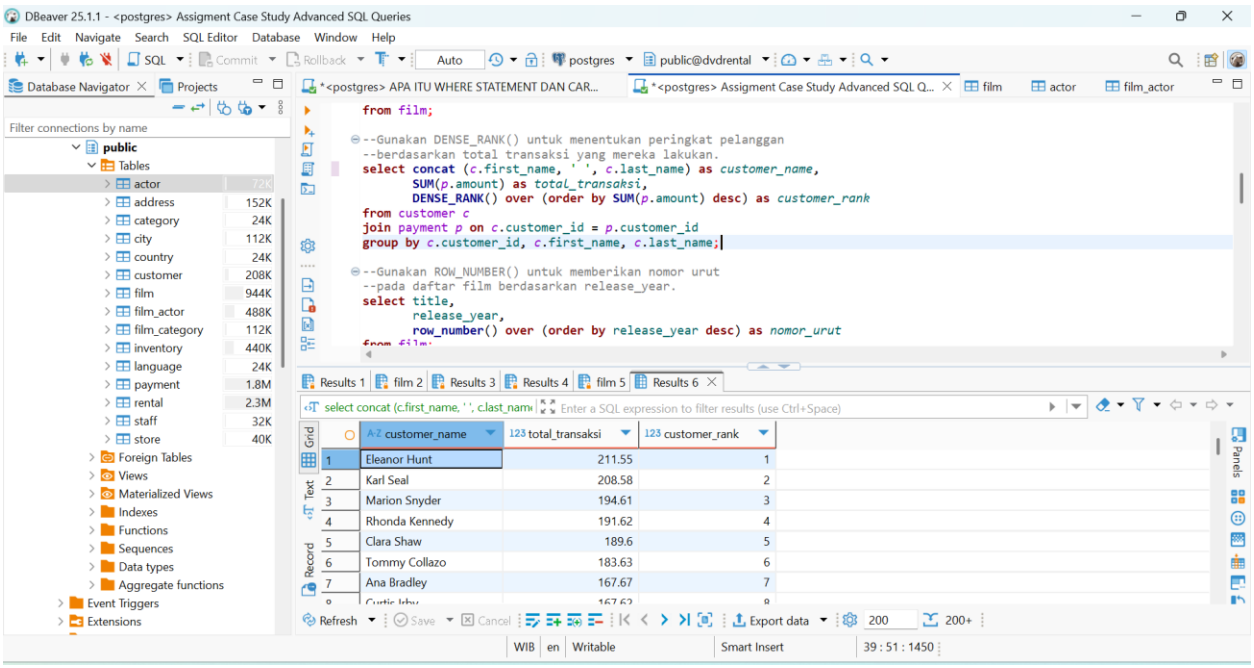

```
--Buat query untuk menampilkan aktor yang hanya membintangi satu film dalam database.
select distinct concat (a.first_name,' ', a.last_name) as actor_name
from actor a
join film_actor fa on a.actor_id = fa.actor_id
group by a.actor_id, a.first_name, a.last_name
having count(distinct fa.film_id) =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



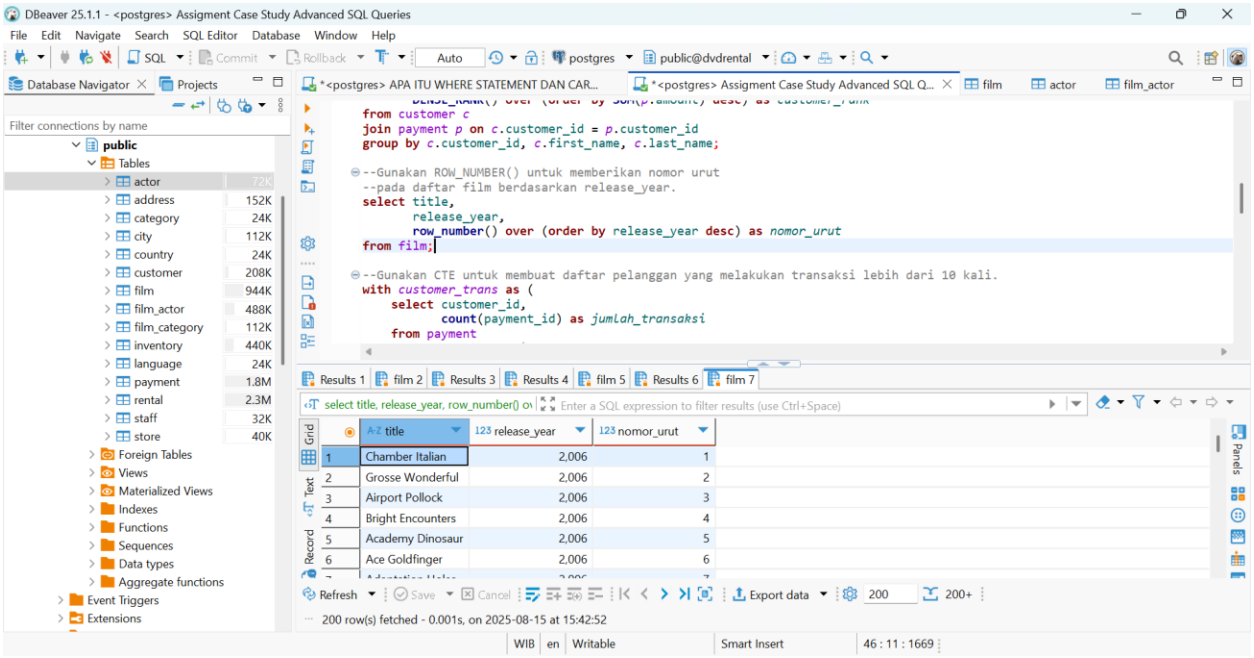

```
--Gunakan RANK() untuk menentukan peringkat film berdasarkan rental_rate.
select title,
       rental_rate,
       RANK() over (order by rental_rate desc) as film_rank
from film;
```



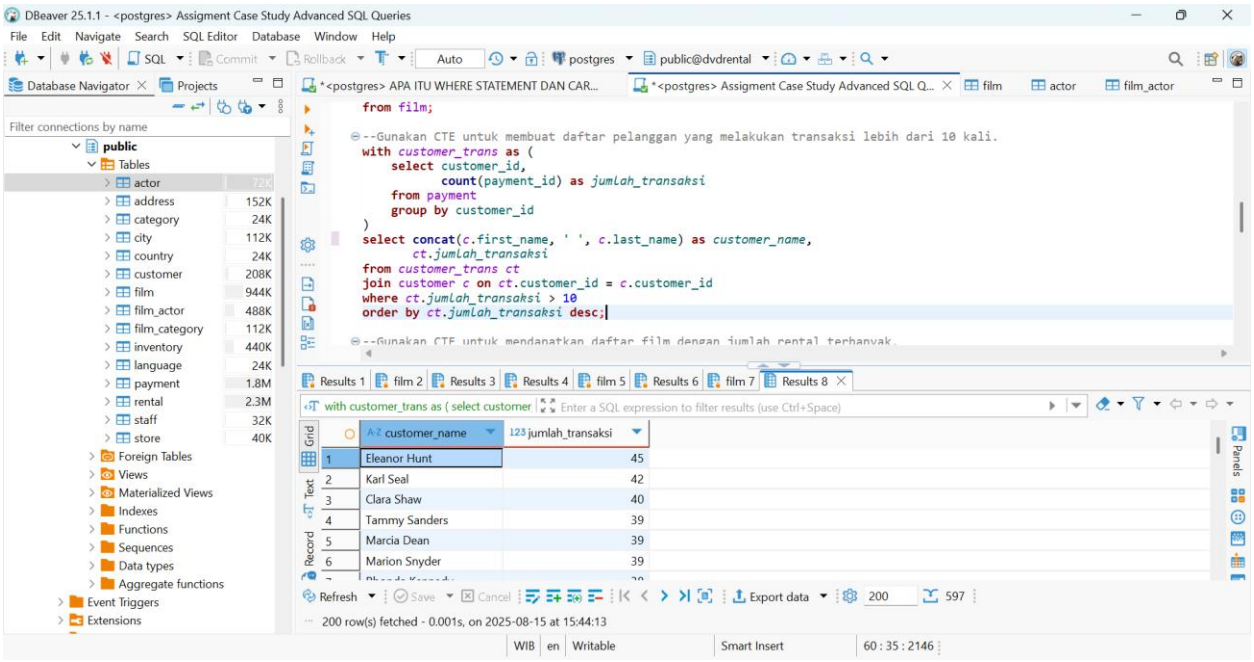

```
--Gunakan DENSE_RANK() untuk menentukan peringkat pelanggan
--berdasarkan total transaksi yang mereka lakukan.
select concat (c.first_name, ' ', c.last_name) as customer_name,
       SUM(p.amount) as total_transaksi,
       DENSE_RANK() over (order by SUM(p.amount) desc) as customer_rank
from customer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



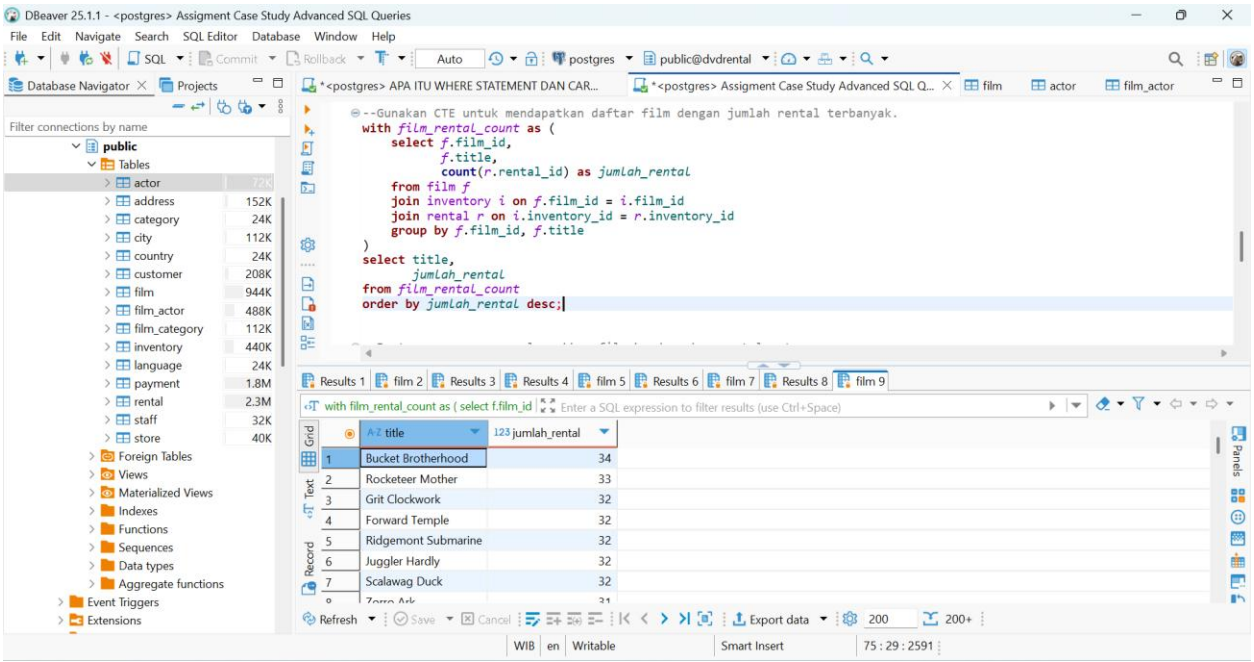
```sql
--Gunakan ROW_NUMBER() untuk memberikan nomor urut
--pada daftar film berdasarkan release_year.
select title,
       release_year,
       row_number() over (order by release_year desc) as nomor_urut
from film;
```



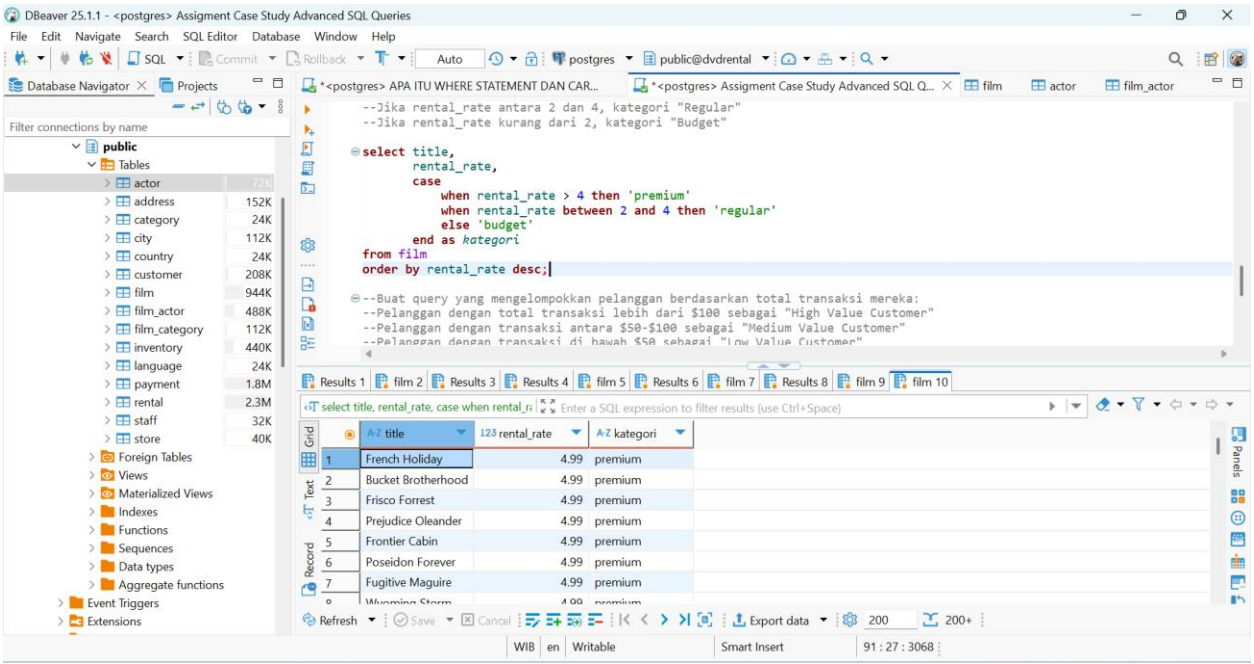

```sql
--Gunakan CTE untuk membuat daftar pelanggan yang melakukan transaksi lebih dari 10 kali.
with customer_trans as (
    select customer_id,
           count(payment_id) as jumlah_transaksi
    from payment
    group by customer_id
)
select concat(c.first_name, ' ', c.last_name) as customer_name,
       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;
```



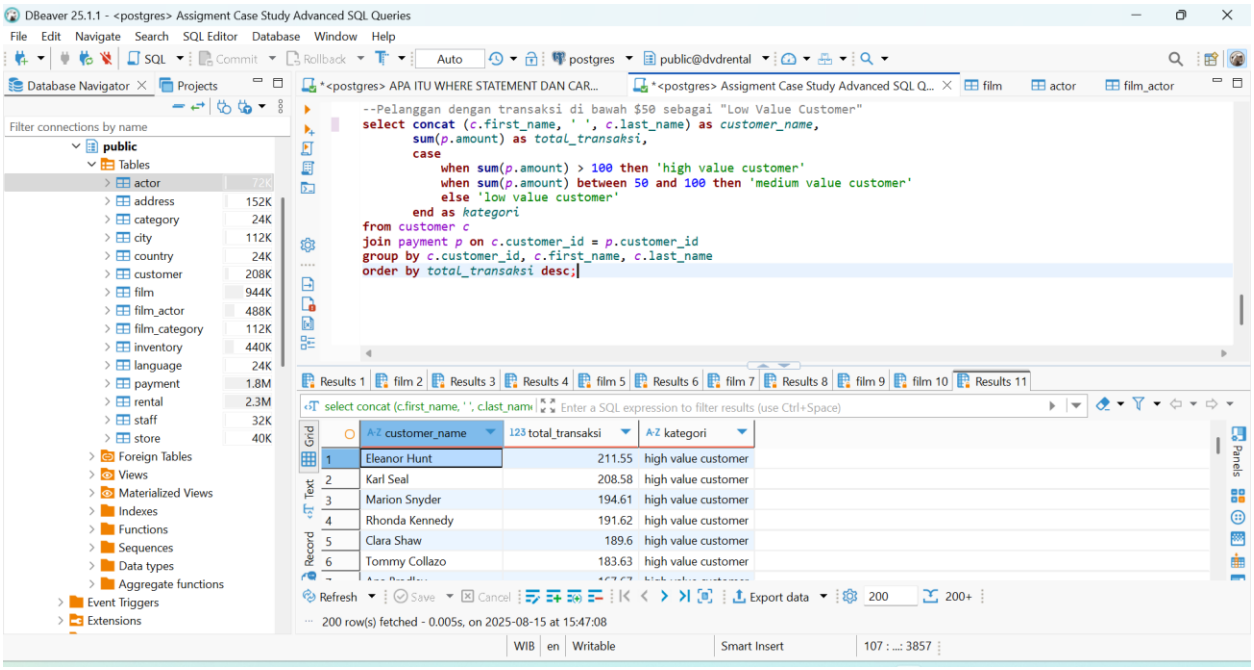

```sql
--Gunakan CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak.
with film_rental_count as (
    select f.film_id,
           f.title,
           count(r.rental_id) as jumlah_rental
    from film f
    join inventory i on f.film_id = i.film_id
    join rental r on i.inventory_id = r.inventory_id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




```sql
--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;
```




```sql
--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> Assigment Case Study Advanced SQL Queries
File Edit Navigate Search SQL Editor Database Window Help
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
*<postgres> APA ITU WHERE STATEMENT DAN CAR...
*<postgres> Assigment Case Study Advanced SQL Q...
film
actor
film_actor
--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_nam
Enter a SQL expression to filter results (use Ctrl+Space)
customer_name
total_transaksi
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