

Dibimbing

DS33B Day 20

Specialized SQL Techniques

By Rizky Febri Ibra Habibie

# Table of Contents

<i>Table of Figures.....</i>	<i>3</i>
<i>1. Menggunakan Subquery.....</i>	<i>4</i>
a. Tampilkan nama pelanggan yang pernah melakukan transaksi dengan jumlah lebih dari rata-rata transaksi di tabel payment. ....	4
b. Ambil daftar film yang memiliki durasi lebih panjang dibandingkan durasi rata-rata dari semua film dalam tabel film. ....	5
c. Buat query untuk menampilkan aktor yang hanya membintangi satu film dalam database. .	6
<i>2. Menggunakan Window Functions .....</i>	<i>9</i>
a. Gunakan RANK() untuk menentukan peringkat film berdasarkan rental_rate. ....	9
b. Gunakan DENSE_RANK() untuk menentukan peringkat pelanggan berdasarkan total transaksi yang mereka lakukan. ....	9
c. Gunakan ROW_NUMBER() untuk memberikan nomor urut pada daftar film berdasarkan release_year. ....	10
<i>3. Menggunakan Common Table Expressions (CTE).....</i>	<i>11</i>
a. Gunakan CTE untuk membuat daftar pelanggan yang melakukan transaksi lebih dari 10 kali.	11
b. Gunakan CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak. ....	12
<i>4. Menggunakan CASE WHEN untuk Klasifikasi Data.....</i>	<i>14</i>
a. Buat query yang mengelompokkan film berdasarkan rental_rate: ....	14
b. Buat query yang mengelompokkan pelanggan berdasarkan total transaksi mereka .....	15
<i>SQL File Link .....</i>	<i>16</i>

# Table of Figures

Figure 1. transaksi dengan jumlah lebih dari rata-rata transaksi di tabel payment .....	4
Figure 2. daftar film yang memiliki durasi lebih panjang dibandingkan durasi rata-rata dari semua film dalam tabel film .....	5
Figure 3. query untuk menampilkan aktor yang hanya membintangi satu film dalam database .....	7
Figure 4. Checking .....	7
Figure 5. RANK() untuk menentukan peringkat film berdasarkan rental_rate .....	9
Figure 6. DENSE_RANK() untuk menentukan peringkat pelanggan berdasarkan total transaksi yang mereka lakukan .....	10
Figure 7. ROW_NUMBER() untuk memberikan nomor urut pada daftar film berdasarkan release_year .....	11
Figure 8. CTE untuk membuat daftar pelanggan yang melakukan transaksi lebih dari 10 kali .....	12
Figure 9. CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak .....	13
Figure 10. query yang mengelompokkan film berdasarkan rental_rate .....	15
Figure 11. query yang mengelompokkan pelanggan berdasarkan total transaksi .....	16

## 1. Menggunakan Subquery

- Tampilkan nama pelanggan yang pernah melakukan transaksi dengan jumlah lebih dari rata-rata transaksi di tabel payment.

**select**

**payment\_id,**

**amount,**

**(select AVG(amount) from payment) avg\_amount,**

**case**

**when amount > (select AVG(amount) from payment) then 'Above Average'**

**else 'Below Average'**

**end as average\_flag**

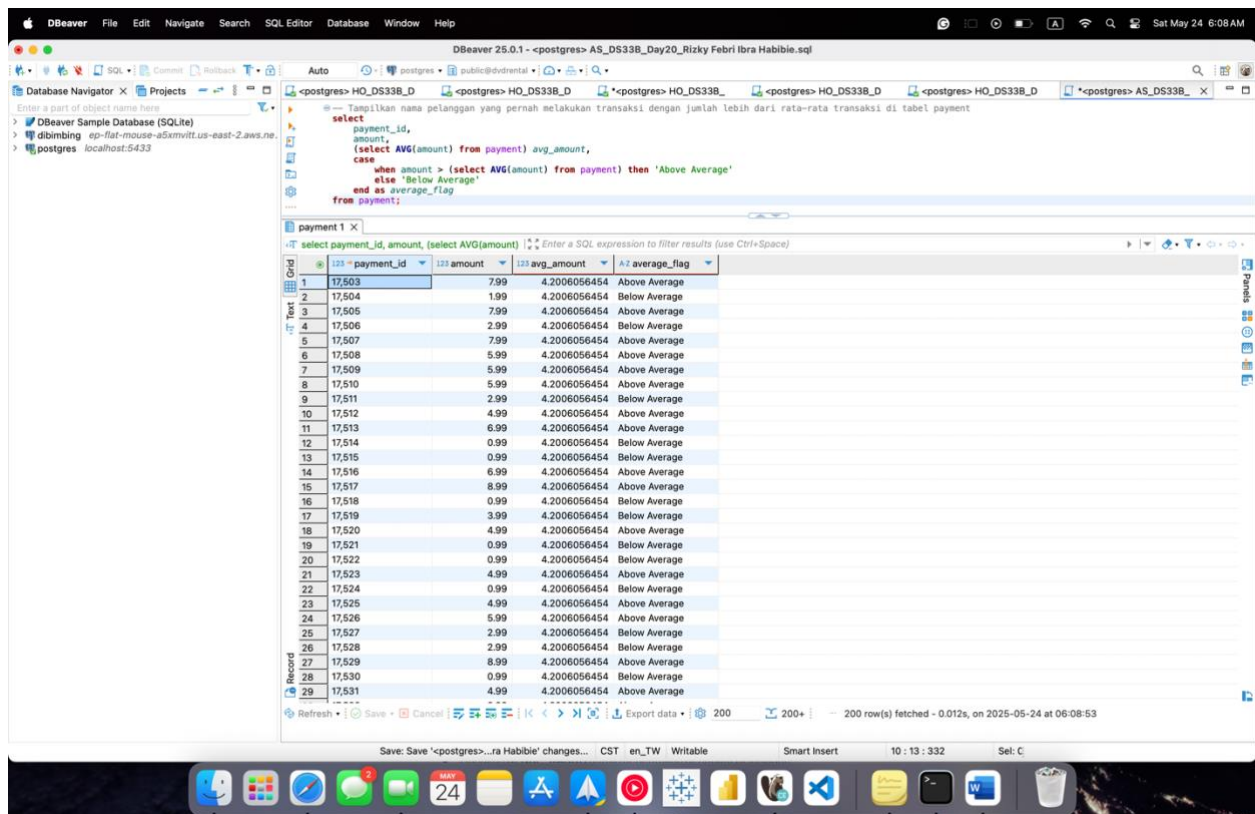


Figure 1. transaksi dengan jumlah lebih dari rata-rata transaksi di tabel payment

**from payment;**

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

**select**

**film\_id,**

**title,**

**length,**

**(select AVG(length) from film) avg\_length,**

**case**

**when length > (select AVG(length) from film) then 'Above Average'**

**else 'Below Average'**

**end as average\_flag**

**from film**

The screenshot shows the DBeaver SQL Editor interface. The SQL Editor window displays the following query:

```
select
  film_id,
  title,
  length,
  (select AVG(length) from film) avg_length,
  case
    when length > (select AVG(length) from film) then 'Above Average'
    else 'Below Average'
  end as average_flag
from film
order by 2 asc;
```

The query results are displayed in a grid view, showing 28 rows of data. The columns are: film\_id, title, length, avg\_length, and average\_flag. The results are ordered by title (film\_id).

film_id	title	length	avg_length	average_flag
1	Academy Dinosaur	86	115.272	Below Average
2	Ace Goldfinger	48	115.272	Below Average
3	Adaptation Holes	50	115.272	Below Average
4	Affair Prejudice	117	115.272	Above Average
5	African Egg	130	115.272	Above Average
6	Agent Truman	169	115.272	Above Average
7	Airplane Sierra	62	115.272	Below Average
8	Airport Pollock	54	115.272	Below Average
9	Alabama Devil	114	115.272	Below Average
10	Aladdin Calendar	63	115.272	Below Average
11	Alamo Videotape	126	115.272	Above Average
12	Alaska Phantom	136	115.272	Above Average
13	Ali Forever	150	115.272	Above Average
14	Alice Fantasia	94	115.272	Below Average
15	Alien Center	46	115.272	Below Average
16	Alley Evolution	180	115.272	Above Average
17	Alone Trip	82	115.272	Below Average
18	Alter Victory	57	115.272	Below Average
19	Amadeus Holy	113	115.272	Below Average
20	Amelle Hellfighters	79	115.272	Below Average
21	American Circus	129	115.272	Above Average
22	Amistad Midsommer	85	115.272	Below Average
23	Anaconda Confessions	92	115.272	Below Average
24	Analyze Hoosiers	181	115.272	Above Average
25	Angels Life	74	115.272	Below Average
26	Annie Identity	86	115.272	Below Average
27	Anonymous Human	179	115.272	Above Average
28	Anthem Luke	91	115.272	Below Average

Figure 2. daftar film yang memiliki durasi lebih panjang dibandingkan durasi rata-rata dari semua film dalam tabel film

**order by 2 asc;**

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

```
select
    actor_id,
    first_name,
    last_name
from actor
where actor_id in (
    select actor_id
    from film_actor
    group by actor_id
    having count (film_id) = 1);
```

-- Check

```
select
    a.actor_id,
    a.first_name,
    a.last_name,
    COUNT(fa.film_id) film_count
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
ORDER BY film_count asc;
```

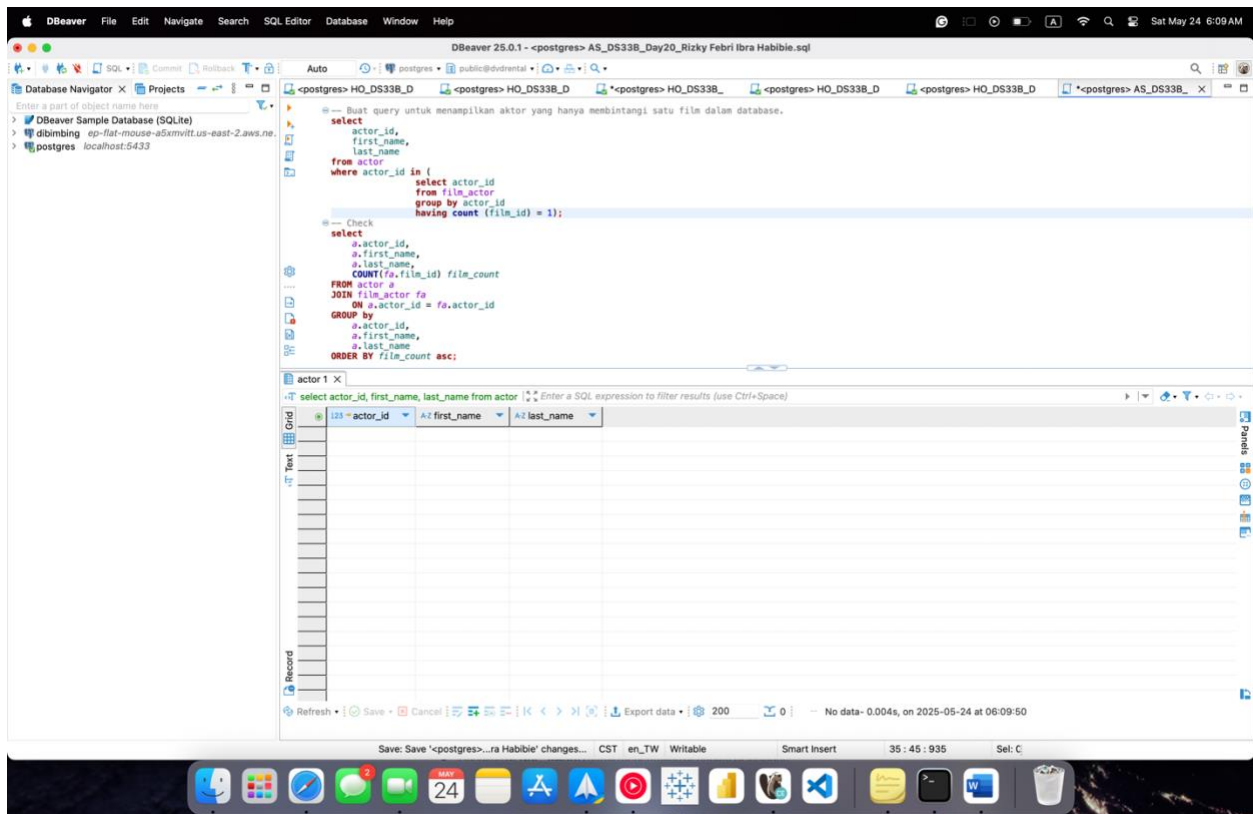


Figure 3. query untuk menampilkan aktor yang hanya membintangi satu film dalam database

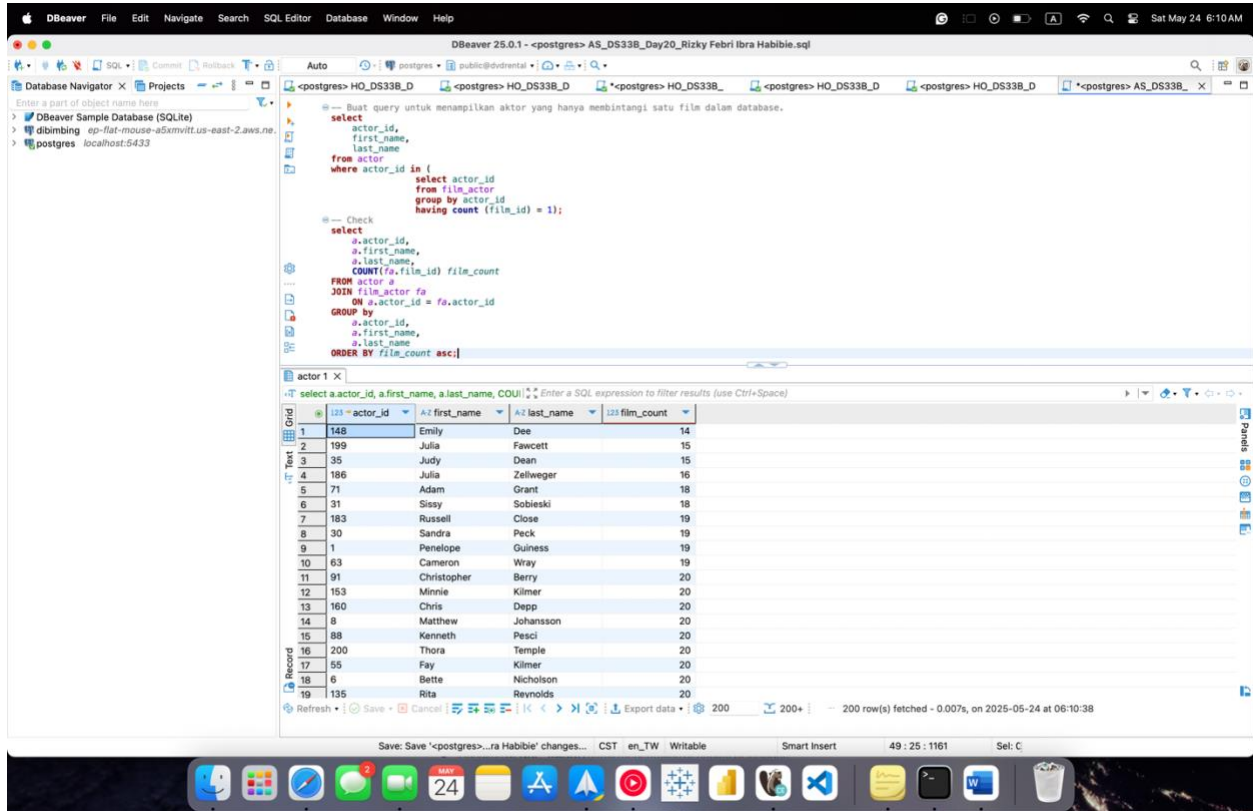


Figure 4. Checking





## 2. Menggunakan Window Functions

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

**select** \* ,

**RANK()** **OVER**(order by **rental\_rate** **DESC**) *ranking*

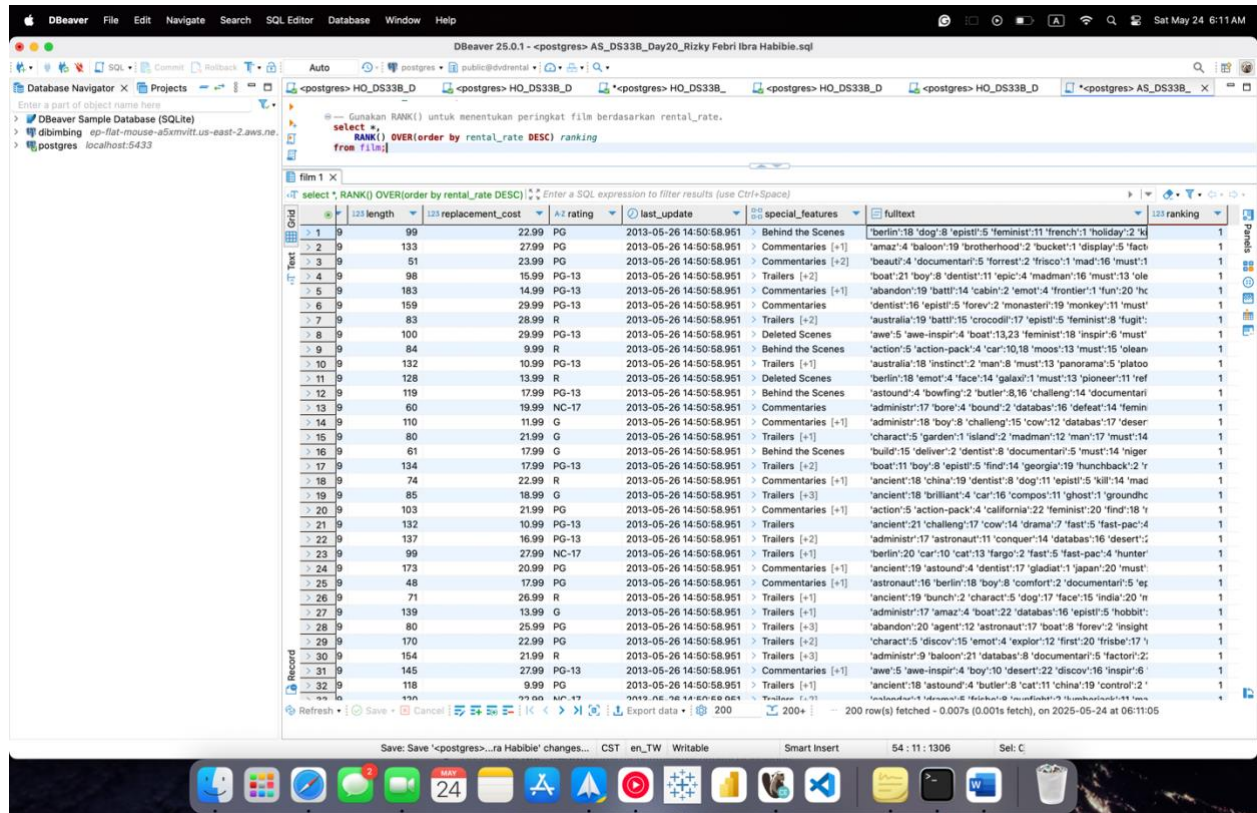


Figure 5. RANK() untuk menentukan peringkat film berdasarkan rental\_rate

**from** film;

- b. Gunakan **DENSE\_RANK()** untuk menentukan peringkat pelanggan berdasarkan total transaksi yang mereka lakukan.

**select**

**c.customer\_id**,

**c.first\_name**,

**c.last\_name**,

**COUNT**(**c.customer\_id**) **as** *total\_transaction*,

**DENSE\_RANK()** **OVER**(order by **COUNT**(c.customer\_id) **DESC**)

*transaction\_rank*

from customer c

join payment p on c.customer\_id = p.customer\_id

group by 1,2,3

order by *transaction\_rank*;

The screenshot shows the DBeaver SQL editor with a query that uses `DENSE_RANK()` to rank customers by their total transaction count. The query is as follows:

```
-- Gunakan DENSE_RANK() untuk menentukan peringkat pelanggan berdasarkan total transaksi yang mereka lakukan.
select
  c.customer_id,
  c.first_name,
  c.last_name,
  COUNT(c.customer_id) as total_transaction,
  DENSE_RANK() OVER (order by COUNT(c.customer_id) DESC) transaction_rank
from customer c
join payment p on c.customer_id = p.customer_id
group by 1,2,3
order by transaction_rank;
```

The results table displays the following data (first 28 rows shown):

	customer_id	first_name	last_name	total_transaction	transaction_rank
1	148	Eleanor	Hunt	45	1
2	526	Karl	Seal	42	2
3	144	Clara	Shaw	40	3
4	75	Tammy	Sanders	39	4
5	236	Marcia	Dean	39	4
6	178	Marion	Snyder	39	4
7	410	Curtis	Irby	38	5
8	137	Rhonda	Kennedy	38	5
9	459	Tommy	Collazo	37	6
10	366	Brandon	Huey	36	7
11	380	Russell	Brinson	35	8
12	29	Angela	Hernandez	35	8
13	469	Wesley	Bull	35	8
14	5	Elizabeth	Brown	35	8
15	468	Tim	Cary	34	9
16	348	Roger	Quintanilla	34	9
17	30	Melissa	King	34	9
18	91	Lois	Butler	34	9
19	373	Louis	Leone	34	9
20	38	Martha	Gonzalez	34	9
21	129	Carrie	Porter	34	9
22	257	Marsha	Douglas	34	9
23	439	Alexander	Fennell	33	10
24	181	Ana	Bradley	33	10
25	66	Janice	Ward	33	10
26	211	Stacey	Montgomery	33	10
27	267	Margie	Wade	33	10
28	354	Justin	Ngo	33	10

Figure 6. `DENSE_RANK()` untuk menentukan peringkat pelanggan berdasarkan total transaksi yang mereka lakukan

- c. Gunakan **ROW\_NUMBER()** untuk memberikan nomor urut pada daftar film berdasarkan **release\_year**.

**select** \*,

**ROW\_NUMBER()** **OVER**(order by **release\_year** **DESC**) *RN\_ranking*

**from** film;

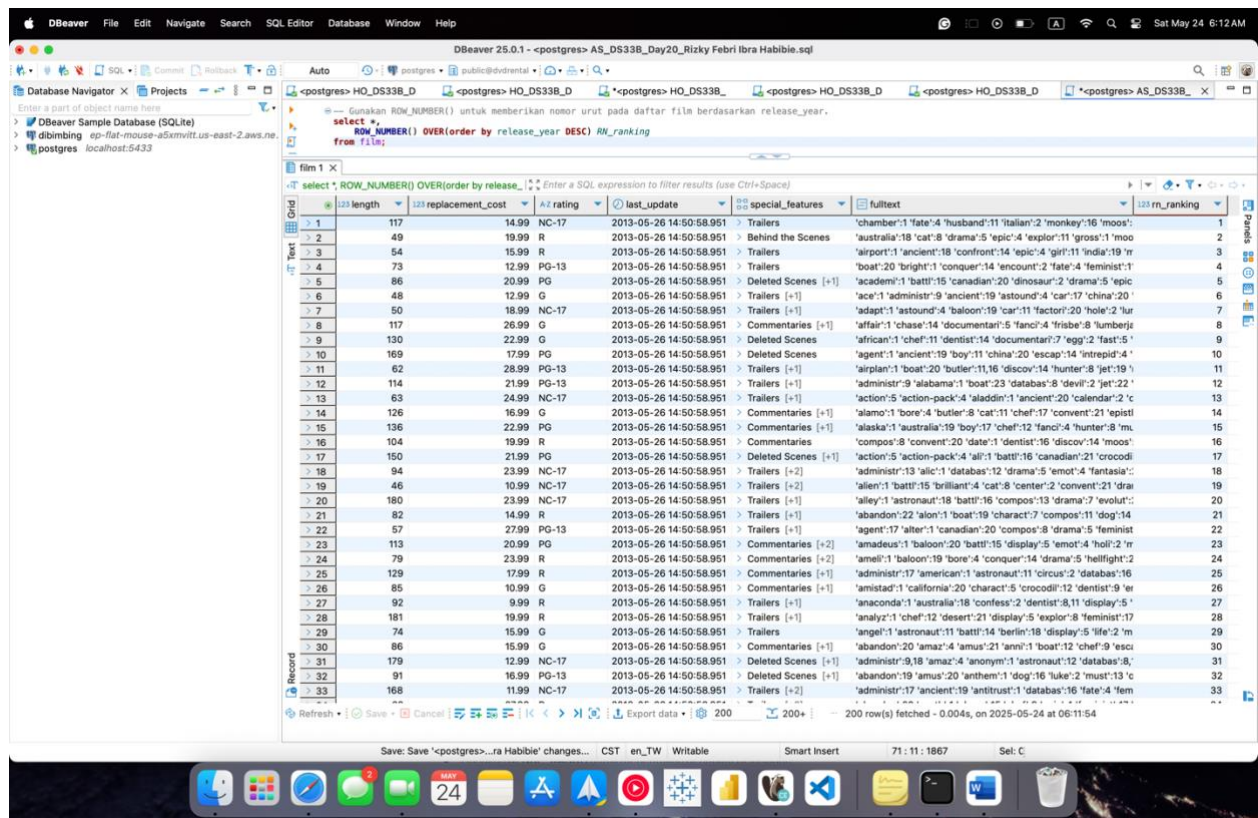


Figure 7. ROW\_NUMBER() untuk memberikan nomor urut pada daftar film berdasarkan release\_year

### 3. Menggunakan Common Table Expressions (CTE)

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

WITH *transaction\_count* AS (

SELECT

customer\_id,

COUNT(payment\_id) AS *total\_transactions*

FROM payment

GROUP BY customer\_id

)

SELECT

c.customer\_id,

*c.first\_name,*

*c.last\_name,*

*tc.total\_transactions*

FROM *transaction\_count* *tc*

JOIN customer *c* ON *c.customer\_id* = *tc.customer\_id*

WHERE *tc.total\_transactions* > 10

ORDER BY *tc.total\_transactions* DESC;

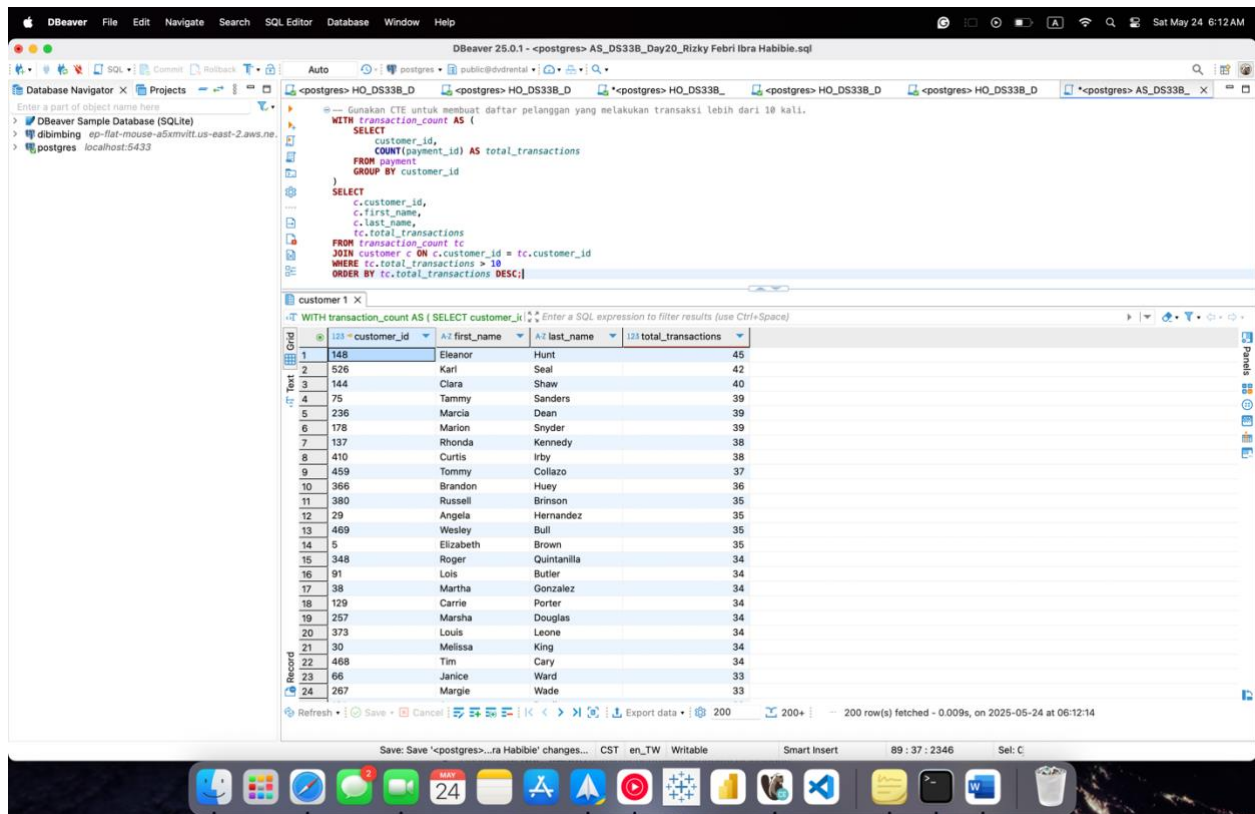


Figure 8. CTE untuk membuat daftar pelanggan yang melakukan transaksi lebih dari 10 kali

b. Gunakan **CTE** untuk mendapatkan daftar film dengan jumlah rental terbanyak.

WITH *film\_rental\_count* AS (

SELECT

*f.film\_id,*

COUNT(*r.rental\_id*) AS *rental\_count*



```

FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
GROUP BY i.film_id
)
SELECT
    f.film_id,
    f.title,
    frc.rental_count
FROM film_rental_count frc
JOIN film f ON f.film_id = frc.film_id
ORDER BY frc.rental_count DESC;

```

Gunakan CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak.

```

WITH film_rental_count AS (
    SELECT
        i.film_id,
        COUNT(r.rental_id) AS rental_count
    FROM rental r
    JOIN inventory i ON r.inventory_id = i.inventory_id
    GROUP BY i.film_id
)
SELECT
    f.film_id,
    f.title,
    frc.rental_count
FROM film_rental_count frc
JOIN film f ON f.film_id = frc.film_id
ORDER BY frc.rental_count DESC;

```

film_id	title	rental_count
103	Bucket Brotherhood	34
738	Rocketeer Mother	33
767	Scalawag Duck	32
382	Grit Clockwork	32
331	Forward Temple	32
489	Juggler Hardly	32
730	Ridgemont Submarine	32
753	Rush Goodfellas	31
369	Goodfellas Salute	31
735	Robbers Joon	31
621	Network Peak	31
418	Hobbit Allen	31
31	Apache Divine	31
973	Wife Turn	31
891	Timberland Sky	31
1,000	Zorro Ark	31
869	Suspects Quills	30
702	Pulp Beverly	30
609	Muscle Bright	30
979	Witches Panic	30
239	Dogma Family	30
341	Frost Head	30
563	Massacre Usual	30
374	Graffiti Love	30
127	Cat Coneheads	30

Figure 9. CTE untuk mendapatkan daftar film dengan jumlah rental terbanyak

#### 4. Menggunakan CASE WHEN untuk Klasifikasi Data

- a. Buat query yang mengelompokkan film berdasarkan **rental\_rate**:
  - i. Jika **rental\_rate** lebih dari 4, kategori "Premium"
  - ii. Jika **rental\_rate** antara 2 dan 4, kategori "Regular"
  - iii. Jika **rental\_rate** kurang dari 2, kategori "Budget"

**SELECT**

film\_id,

title,

rental\_rate,

**CASE**

**WHEN** rental\_rate > 4 **THEN** 'Premium'

**WHEN** rental\_rate **BETWEEN** 2 **AND** 4 **THEN** 'Regular'

**WHEN** rental\_rate < 2 **THEN** 'Budget'

**END AS** *rental\_category*

**FROM** film

**ORDER BY** rental\_rate **DESC**;

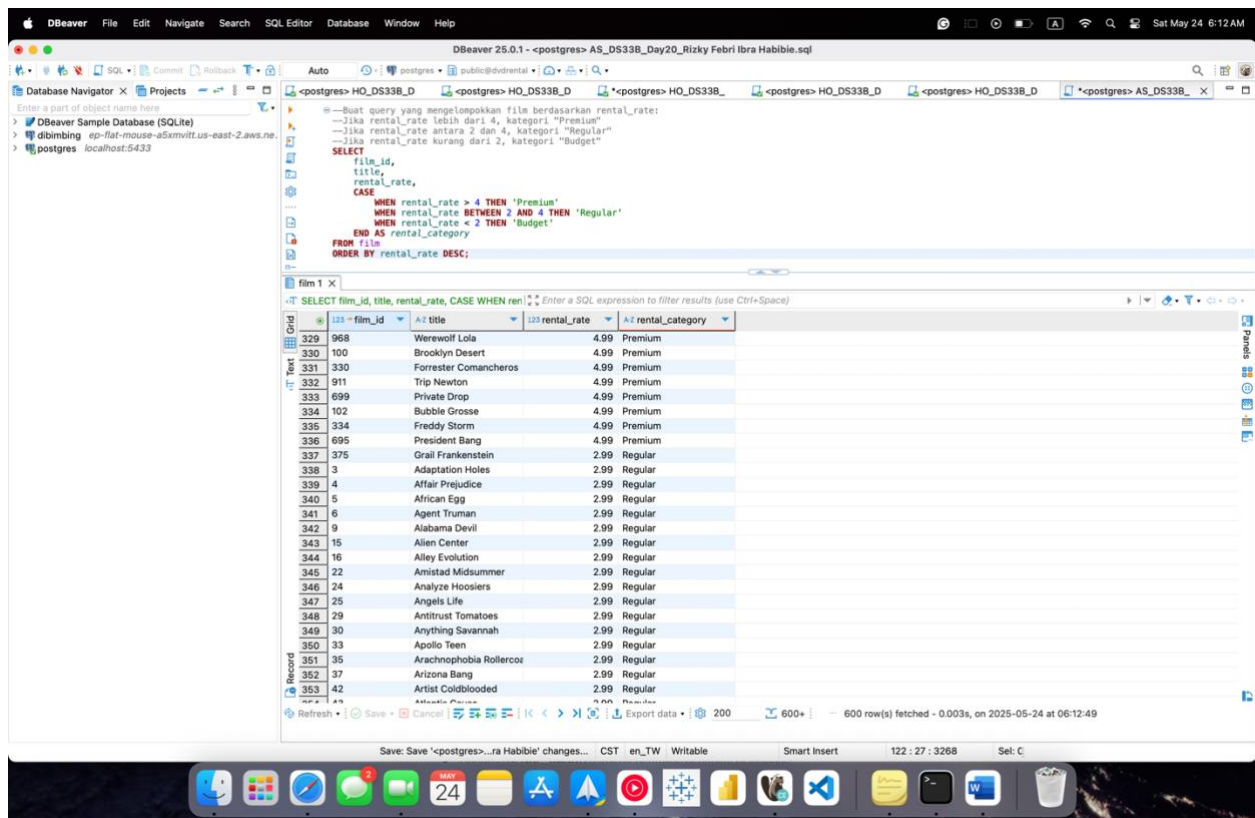


Figure 10. query yang mengelompokkan film berdasarkan rental\_rate

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

**SELECT**

**c.customer\_id,**

**c.first\_name,**

**c.last\_name,**

**SUM(p.amount) AS total\_transaction,**

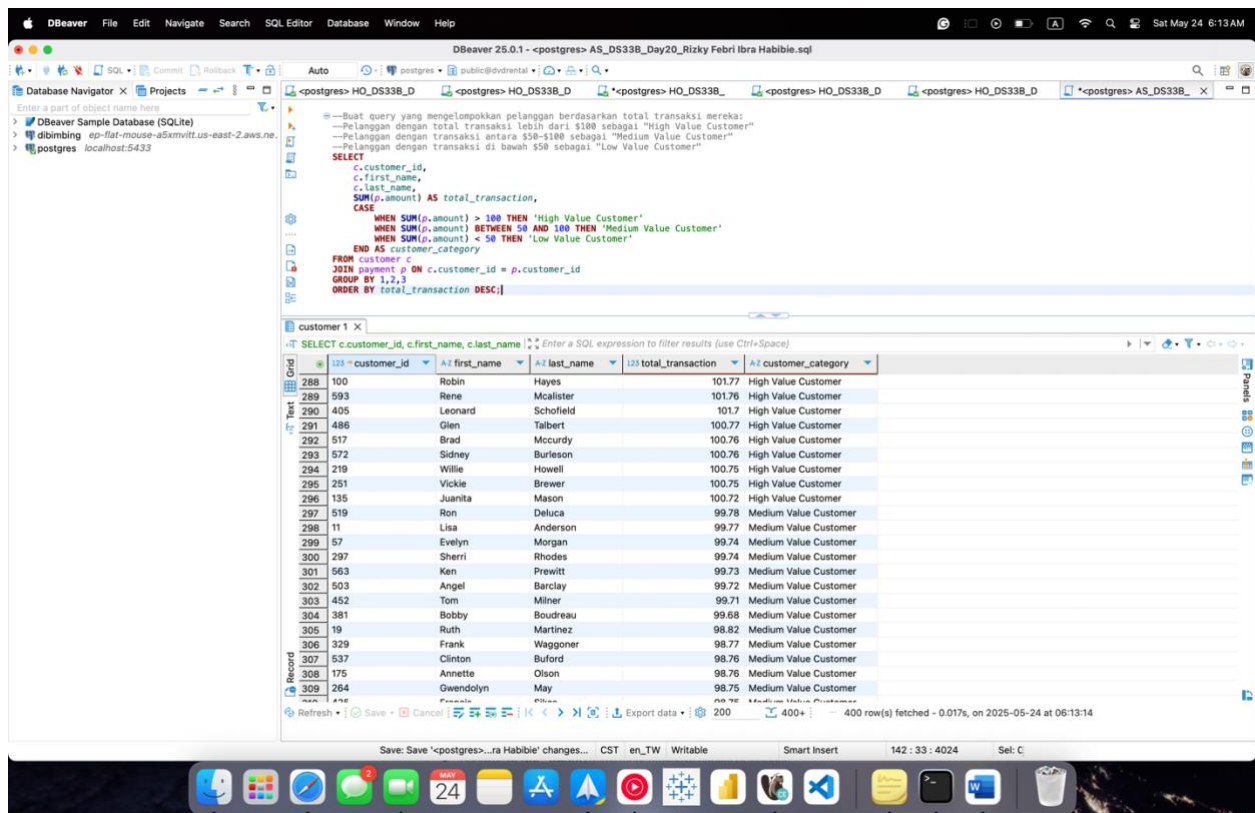
**CASE**

**WHEN SUM(p.amount) > 100 THEN 'High Value Customer'**

```

        WHEN SUM(p.amount) BETWEEN 50 AND 100 THEN 'Medium Value Customer'
        WHEN SUM(p.amount) < 50 THEN 'Low Value Customer'
    END AS customer_category
FROM customer c
JOIN payment p ON c.customer_id = p.customer_id
GROUP BY 1,2,3

```



The screenshot shows the DBeaver SQL Editor interface. The SQL Editor window displays a query that categorizes customers based on their total transaction amount. The query uses a CASE statement to assign categories: 'High Value Customer' for transactions over \$100, 'Medium Value Customer' for transactions between \$50 and \$100, and 'Low Value Customer' for transactions under \$50. The results are displayed in a table with columns for customer ID, first name, last name, total transaction amount, and customer category. The results are ordered by total transaction amount in descending order.

customer_id	first_name	last_name	total_transaction	customer_category
100	Robin	Hayes	101.77	High Value Customer
593	Rene	Mcalister	101.76	High Value Customer
405	Leonard	Schofield	101.7	High Value Customer
486	Glen	Talbert	100.77	High Value Customer
517	Brad	Mccurdy	100.76	High Value Customer
572	Sidney	Burleson	100.76	High Value Customer
219	Willie	Howell	100.75	High Value Customer
251	Vickie	Brewer	100.75	High Value Customer
135	Juanita	Mason	100.72	High Value Customer
519	Ron	Deluca	99.78	Medium Value Customer
11	Lisa	Anderson	99.77	Medium Value Customer
57	Evelyn	Morgan	99.74	Medium Value Customer
297	Sherri	Rhodes	99.74	Medium Value Customer
563	Ken	Prewitt	99.73	Medium Value Customer
503	Angel	Barclay	99.72	Medium Value Customer
452	Tom	Milner	99.71	Medium Value Customer
381	Bobby	Boudreau	99.68	Medium Value Customer
19	Ruth	Martinez	98.82	Medium Value Customer
329	Frank	Waggoner	98.77	Medium Value Customer
537	Clinton	Buford	98.76	Medium Value Customer
175	Annette	Olson	98.76	Medium Value Customer
264	Gwendolyn	May	98.75	Medium Value Customer

Figure 11. query yang mengelompokkan pelanggan berdasarkan total transaksi

```
ORDER BY total_transaction DESC;
```

## SQL File Link

[Click here](#)