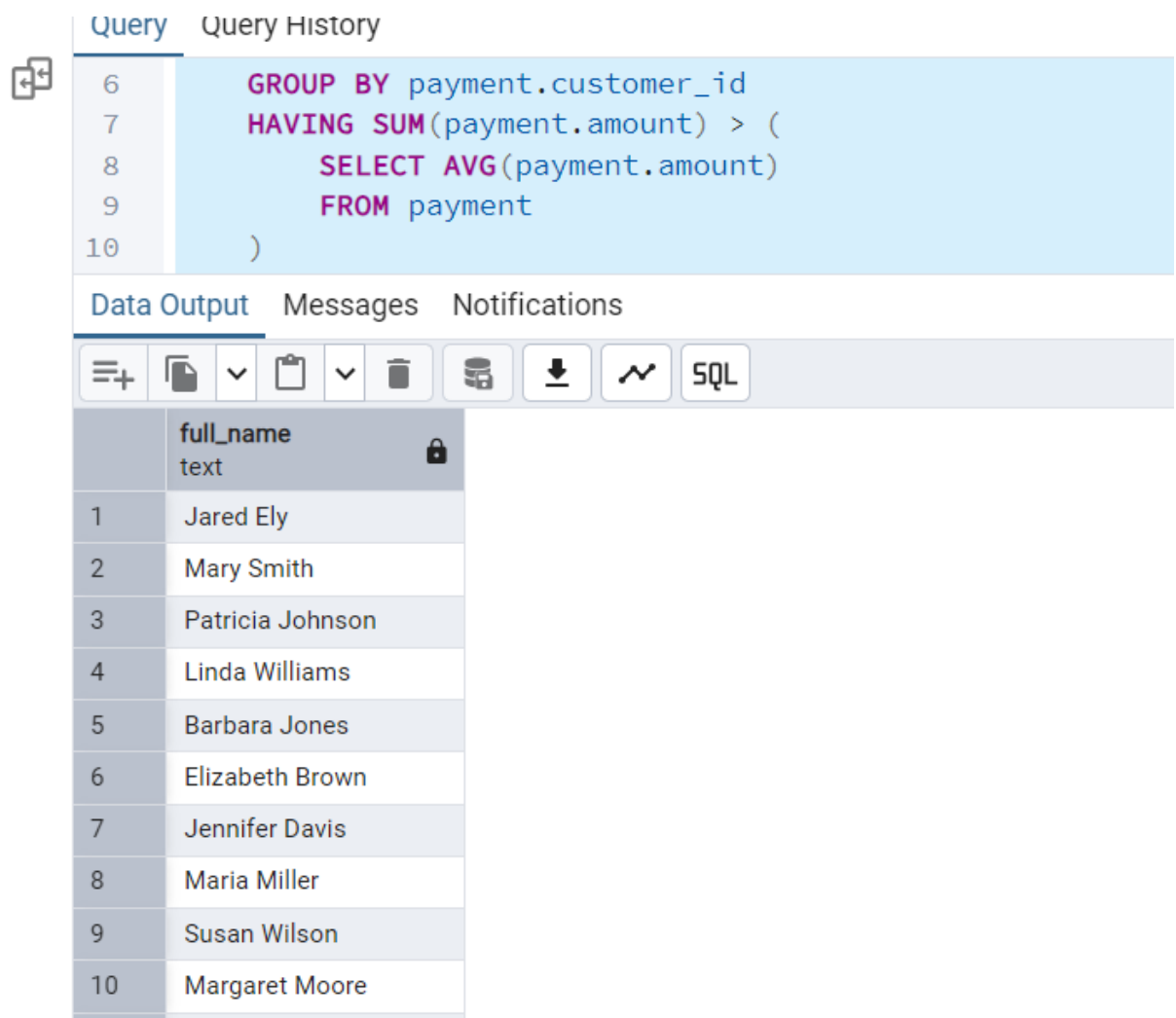


Lakukan langkah-langkah berikut untuk menyelesaikan tugas:

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

Jawab

```
SELECT CONCAT(customer.first_name, ' ', customer.last_name) AS full_name
FROM customer
WHERE customer.customer_id IN (
    SELECT payment.customer_id
    FROM payment
    GROUP BY payment.customer_id
    HAVING SUM(payment.amount) > (
        SELECT AVG(payment.amount)
        FROM payment
    )
);
```



The screenshot shows a SQL IDE interface. At the top, there are tabs for "Query" and "Query History". Below the tabs, a query is entered in the editor, spanning lines 6 to 10. The query is a subquery that filters customers based on their total payment amount compared to the average payment amount. Below the editor, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is active, showing a table with 10 rows of customer names. The table has a single column named "full\_name" with a text data type and a lock icon. The names listed are Jared Ely, Mary Smith, Patricia Johnson, Linda Williams, Barbara Jones, Elizabeth Brown, Jennifer Davis, Maria Miller, Susan Wilson, and Margaret Moore.

	full_name text
1	Jared Ely
2	Mary Smith
3	Patricia Johnson
4	Linda Williams
5	Barbara Jones
6	Elizabeth Brown
7	Jennifer Davis
8	Maria Miller
9	Susan Wilson
10	Margaret Moore

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

Jawab

```
SELECT title
FROM film
WHERE length > (
    SELECT AVG(length)
    FROM film
);
```

Query Query History

```
3 WHERE length > (
4     SELECT AVG(length)
5     FROM film
6 );
7
```

Data Output Messages Notifications

	title character varying (255)
1	Chamber Italian
2	Affair Prejudice
3	African Egg
4	Agent Truman
5	Alamo Videotape
6	Alaska Phantom
7	Ali Forever
8	Alley Evolution
9	American Circus
10	Analyze Hoosiers
11	Anonymous Human
12	Antitrust Tomatoes
13	Apocalypse Flamingos
14	Apollo Teen
15	Arachnophobia Rollercoaster

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

Jawab

```
SELECT concat(actor.first_name, ' ', actor.last_name) as actor_name
FROM actor
WHERE actor.actor_id IN (
    SELECT actor_id
    FROM film_actor
```

```
GROUP BY actor_id  
HAVING COUNT(film_id) = 1  
);
```

Query Query History

```
1 SELECT concat(actor.first_name, ' ', actor.last_name) as actor_name  
2 FROM actor  
3 WHERE actor.actor_id IN (  
4     SELECT actor_id  
5     FROM film_actor
```

Data Output Messages Notifications



actor\_name  
text

## 2. Menggunakan Window Functions

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

Jawab

```
SELECT title, rental_rate, RANK() OVER (ORDER BY rental_rate DESC) AS  
rental_rate_rank  
FROM film;
```

```

SELECT title, rental_rate, RANK() OVER (ORDER BY rental_rate)
FROM film;

```

ata Output Messages Notifications

	title character varying (255)	rental_rate numeric (4,2)	rental_rate_rank bigint
	French Holiday	4.99	1
	Bucket Brotherhood	4.99	1
	Frisco Forrest	4.99	1
	Prejudice Oleander	4.99	1
	Frontier Cabin	4.99	1
	Poseidon Forever	4.99	1
	Fugitive Maguire	4.99	1
	Wyoming Storm	4.99	1
	Pluto Oleander	4.99	1
0	Platoon Instinct	4.99	1
1	Galaxy Sweethearts	4.99	1
2	Games Bowfinger	4.99	1
3	Pity Bound	4.99	1
4	Trap Guys	4.99	1

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

Jawab

```

SELECT customer.first_name, customer.last_name, SUM(payment.amount) AS
total_spent,
       DENSE_RANK() OVER (ORDER BY SUM(payment.amount) DESC) AS customer_rank
FROM customer
JOIN payment ON customer.customer_id = payment.customer_id
GROUP BY customer.customer_id
ORDER BY customer_rank;

```

```

3 FROM customer
4 JOIN payment ON customer.customer_id = payment.customer_id
5 GROUP BY customer.customer_id
6 ORDER BY customer_rank;
7

```

Data Output Messages Notifications

	first_name character varying (45)	last_name character varying (45)	total_spent numeric	customer_rank bigint
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	189.60	5
6	Tommy	Collazo	183.63	6
7	Ana	Bradley	167.67	7
8	Curtis	Irby	167.62	8
9	Marcia	Dean	166.61	9
10	Mike	Way	162.67	10
11	Arnold	Havens	161.68	11
12	Wesley	Bull	158.65	12
13	Gordon	Allard	157.69	13
14	Louis	Leone	156.66	14
15	Lena	Jensen	154.70	15

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

Jawab

```

SELECT title, release_year, ROW_NUMBER() OVER (ORDER BY release_year DESC) AS
row_num
FROM film;

```

Query		Query History	
1	▼	<b>SELECT</b> title, release_year, <b>ROW_NUMBER()</b> <b>OVER</b> ( <b>ORDER B</b>	
2		<b>FROM</b> film;	
3			

Data Output		Messages	Notifications
-------------	--	----------	---------------

≡	📄	▼	📋	▼	🗑️	🗄️	⬇️	📈	SQL
---	---	---	---	---	----	----	----	---	-----

	title character varying (255) 🔒	release_year integer 🔒	row_num bigint 🔒
1	Chamber Italian	2006	1
2	Grosse Wonderful	2006	2
3	Airport Pollock	2006	3
4	Bright Encounters	2006	4
5	Academy Dinosaur	2006	5
6	Ace Goldfinger	2006	6
7	Adaptation Holes	2006	7
8	Affair Prejudice	2006	8
9	African Egg	2006	9
10	Agent Truman	2006	10
11	Airplane Sierra	2006	11
12	Alabama Devil	2006	12
13	Aladdin Calendar	2006	13

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

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

Jawab

```
WITH Frequent_Customers AS (
  SELECT customer_id, COUNT(*) AS transaction_count
  FROM payment
  GROUP BY customer_id
  HAVING COUNT(*) > 10
)
SELECT customer.first_name, customer.last_name,
Frequent_Customers.transaction_count
FROM Frequent_Customers
JOIN customer ON Frequent_Customers.customer_id = customer.customer_id;
```

```

7  SELECT customer.first_name, customer.last_name, Frequent_Cust
8  FROM Frequent_Customers
9  JOIN customer ON Frequent_Customers.customer_id = customer.cu:
10

```

Data Output Messages Notifications

	first_name character varying (45)	last_name character varying (45)	transaction_count bigint
1	Jared	Ely	17
2	Mary	Smith	30
3	Patricia	Johnson	26
4	Linda	Williams	24
5	Barbara	Jones	22
6	Elizabeth	Brown	35
7	Jennifer	Davis	25
8	Maria	Miller	28
9	Susan	Wilson	23
10	Margaret	Moore	20
11	Dorothy	Taylor	24
12	Lisa	Anderson	23
13	Nancy	Thomas	26
14	Karen	Jackson	27
15	Betty	White	23

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

Jawab

```

WITH Film_Rentals AS (
  SELECT film_id, COUNT(*) AS rental_count
  FROM inventory
  JOIN rental ON inventory.inventory_id = rental.inventory_id
  GROUP BY film_id
  ORDER BY rental_count DESC
)
SELECT film.title, Film_Rentals.rental_count
FROM Film_Rentals
JOIN film ON Film_Rentals.film_id = film.film_id
LIMIT 10;

```

Query Query History

```

8  SELECT film.title, Film_Rentals.rental_count
9  FROM Film_Rentals
10 JOIN film ON Film_Rentals.film_id = film.film_id
11 LIMIT 10;
12

```

Data Output Messages Notifications

	title character varying (255)	rental_count bigint
1	Bucket Brotherhood	34
2	Rocketeer Mother	33
3	Grit Clockwork	32
4	Forward Temple	32
5	Ridgemont Submarine	32
6	Juggler Hardly	32
7	Scalawag Duck	32
8	Zorro Ark	31
9	Network Peak	31
10	Goodfellas Salute	31

4. Menggunakan CASE WHEN untuk Klasifikasi Data

- 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"

Jawab

```

SELECT 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_rate_category

```



```
FROM film;
```

```
SELECT 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'
```

SQL Output Messages Notifications

title	rental_rate	rental_rate_category
character varying (255)	numeric (4,2)	text
Chamber Italian	4.99	Premium
Grosse Wonderful	4.99	Premium
Airport Pollock	4.99	Premium
Bright Encounters	4.99	Premium
Academy Dinosaur	0.99	Budget
Ace Goldfinger	4.99	Premium
Adaptation Holes	2.99	Regular
Affair Prejudice	2.99	Regular
African Egg	2.99	Regular
Agent Truman	2.99	Regular
Airplane Sierra	4.99	Premium
Alabama Devil	2.99	Regular
Aladdin Calendar	4.99	Premium
Alamo Videotape	0.99	Budget
Alaska Phantom	0.99	Budget

- 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"

Jawab

```
SELECT customer.first_name, customer.last_name, SUM(payment.amount) AS  
total_spent,  
       CASE  
         WHEN SUM(payment.amount) > 100 THEN 'High Value Customer'  
         WHEN SUM(payment.amount) BETWEEN 50 AND 100 THEN 'Medium Value  
Customer'  
         WHEN SUM(payment.amount) < 50 THEN 'Low Value Customer'  
       END AS customer_category  
FROM customer  
JOIN payment ON customer.customer_id = payment.customer_id  
GROUP BY customer.customer_id;
```

