



GROUP BY Statements Lecture

Aggregation Functions



ศึกษาเพิ่มเติม

9.21. Aggregate Functions

Aggregate functions compute a single result from a set of input values. The built-in general-purpose aggregate functions are listed in Table 9.58 while statistical

 <https://www.postgresql.org/docs/current/functions-aggregate.html>



- Most Common Aggregate Functions:
 - AVG() - returns average value
 - COUNT() - returns number of values
 - MAX() - returns maximum value
 - MIN() - returns minimum value
 - SUM() - returns the sum of all values



การเรียกใช้ Aggregation Functions จะเกิดขึ้นเฉพาะในส่วนคำสั่ง SELECT หรือส่วนคำสั่ง HAVING

```
SELECT MIN(replacement_cost) FROM film;
```

ใช้ ROUND ปรับทศนิยม

```
SELECT ROUND(AVG(replacement_cost), 2) FROM film;
```

GROUP BY

- `GROUP BY` อยู่หลัง `FROM` หรือหลัง `WHERE`
- Tip - คอลัมน์ไหนที่อยู่ `GROUP BY` ให้เราเขียนคอลัมน์นั้นใน `SELECT` ด้วย 😊
- `SELECT category_col , AGG(data_col)`
`FROM table`
`GROUP BY category_col`
- The GROUP BY clause must appear right after a FROM or WHERE statement.



Category	Data Value
A	10
A	5
B	2
B	4
C	12
C	6

A	10
A	5

B	2
B	4

C	12
C	6

Aggregate Function
SUM

Category	Result
A	15
B	6
C	18

PIERIAN  DATA

- `SELECT category_col , AGG(data_col)`
`FROM table`
`GROUP BY category_col`
- In the SELECT statement, columns must either have an aggregate function or be in the GROUP BY call.



เราใช้ **HAVING** สำหรับ filter กลุ่ม (เขียนหลังจาก **GROUP BY** clause)

ส่วน **WHERE** ใช้ filter ข้อมูลในตาราง (เขียนก่อน **GROUP BY** clause)

- `SELECT company, division, SUM(sales)`
`FROM finance_table`
`WHERE division IN ('marketing', 'transport')`
`GROUP BY company, division`
- WHERE statements should not refer to the aggregation result, later on we will learn to use HAVING to filter on those results.

```
SELECT customer_id, COUNT(amount) FROM payment
GROUP BY customer_id
ORDER BY COUNT(amount) DESC;
```

```
SELECT DATE(payment_date), SUM(amount) FROM payment
GROUP BY DATE(payment_date)
ORDER BY SUM(amount) DESC;
--> เป็นการนำฟังก์ชัน DATE เพื่อแยกวันที่เดือนปีออกมาจาก payment_date
--> แล้วเราก็ แยก SUM(amount) ตามข้อมูลที่เรา GROUP BY ไว้
```

Data Output		Messages	Notifications
	date date	sum numeric	
1	2007-04-30	5723.89	
2	2007-03-21	2868.27	
3	2007-03-01	2808.24	
4	2007-04-29	2717.60	
5	2007-03-18	2701.76	
6	2007-04-27	2673.57	
Total rows: 32 of 32		Query complete 00:00:00.039	

Challenge → GROUP BY

- เรามีพนักงานสองคน โดยมีรหัสพนักงาน 1 และ 2 เราต้องการมอบโบนัสให้กับพนักงานที่จัดการการชำระเงินมากที่สุด (ส่วนใหญ่อยู่ในเงื่อนไขของจำนวนการชำระเงินที่ประมวลผล ไม่ใช่จำนวนเงินทั้งหมด)
- พนักงานแต่ละคนจัดการการชำระเงินจำนวนเท่าใดและใครได้รับโบนัส

Solution

```
SELECT staff_id, COUNT(amount) FROM payment
GROUP BY staff_id;
--> 1 7292
--> 2 7304
```

- สำนักงานใหญ่ของบริษัทกำลังดำเนินการศึกษาความสัมพันธ์ระหว่างต้นทุนการเปลี่ยนสินค้า (replacement cost) และการจัดเรต MPAA ของภาพยนตร์ (เช่น G, PG, R ฯลฯ...)
- ต้นทุนการเปลี่ยนโดยเฉลี่ยต่อคะแนน MPAA คืออะไร
หมายเหตุ: คุณอาจต้องขยายคอลัมน์ AVG เพื่อดูผลลัพธ์ที่ต้องการ

Solution

```
SELECT rating, ROUND(AVG(replacement_cost),2) FROM film
GROUP BY rating;
```

Data Output			Messages	Notifications
	rating mpaa_rating	round numeric		
1	PG	18.96		
2	R	20.23		
3	NC-17	20.14		
4	PG-13	20.40		
5	G	20.12		
Total rows: 5 of 5			Query complete 00:00:00.041	

- เรากำลังจัดโปรโมชันเพื่อตอบแทนลูกค้า 5 อันดับแรกของเราด้วยคูปอง
- รหัสลูกค้าของลูกค้า 5 อันดับแรกตามการใช้จ่ายทั้งหมดคืออะไร

Solution

```
SELECT customer_id , SUM(amount)
FROM payment
GROUP BY customer_id
ORDER BY SUM(amount) DESC
LIMIT 5;
```

Data Output			Messages	Notifications
	customer_id smallint	sum numeric		
1	148	211.55		
2	526	208.58		
3	178	194.61		
4	137	191.62		
5	144	189.60		
Total rows: 5 of 5			Query complete 00:00:00.046	

HAVING



เราใช้ **HAVING** สำหรับ filter กลุ่ม (เขียนหลังจาก **GROUP BY** clause)

- SELECT company, SUM(sales)
FROM finance_table
WHERE company != 'Google'
GROUP BY company
- We can not use WHERE to filter based off of aggregate results, because those happen **after** a WHERE is executed.



เราไม่สามารถใช้ WHERE เพื่อกรองตามผลลัพธ์รวมได้ เนื่องจากสิ่งเหล่านั้นเกิดขึ้นหลังจาก WHERE ถูกดำเนินการไปแล้ว
แต่ HAVING ช่วยให้เราสามารถใช้ผลลัพธ์รวมเป็นตัวกรองพร้อมกับ GROUP BY ได้

- SELECT company, SUM(sales)
FROM finance_table
WHERE company != 'Google'
GROUP BY company
HAVING SUM(sales) > 1000
- HAVING allows us to use the aggregate result as a filter along with a GROUP BY

```
SELECT customer_id , SUM(amount)
FROM payment
WHERE customer_id NOT IN (184, 87, 477)
GROUP BY customer_id;
--> แบบนี้นั้นจะ filter ก่อน แล้วพอมานะ GROUP BY จะไม่ได้ผลลัพธ์แยกเพราะมัน SUM รวมกันหมดแล้ว
```

```
SELECT customer_id , SUM(amount)
FROM payment
GROUP BY customer_id
HAVING SUM(amount) > 100;
--> จะเป็นการ GROUP BY ก่อน แล้วค่อยไปทำเงื่อนไข SUM
```

Challenge → HAVING

- เรา กำลังเปิดตัวบริการระดับแพลตินัมสำหรับลูกค้าที่ภักดีที่สุดของเรา
- เราจะกำหนดสถานะแพลตินัมให้กับลูกค้าที่มีธุรกรรมการชำระเงิน 40 รายการขึ้นไป
- customer_ids ใดที่มีสิทธิ์ได้รับสถานะแพลตินัมนี้

Solution

```
SELECT customer_id, COUNT(*)
FROM payment
GROUP BY customer_id
HAVING COUNT(*) >= 40;
```

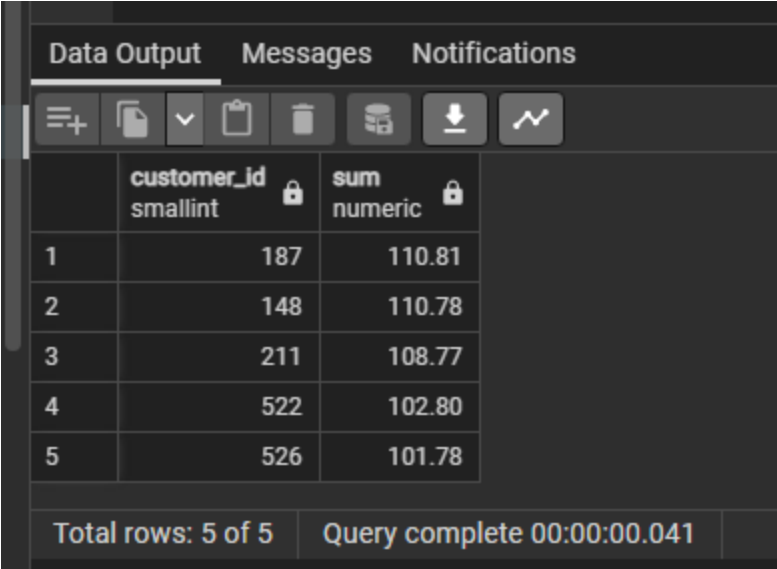
Data Output		Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>			
	customer_id smallint	count bigint	
1	144	40	
2	526	42	
3	148	45	
Total rows: 3 of 3		Query complete 00:00:00.040	

- รหัสลูกค้าของลูกค้าที่ใช้จ่ายมากกว่า \$100 ในธุรกรรมการชำระเงิน (payment transactions) กับ staff_id สมาชิกที่ 2 คืออะไร

```
SELECT customer_id, SUM(amount)
FROM payment
GROUP BY staff_id, customer_id
HAVING SUM(amount) > 100
AND staff_id = 2;
```

หรือ

```
SELECT customer_id, SUM(amount)
FROM payment
WHERE staff_id = 2
GROUP BY customer_id
HAVING SUM(amount) > 100;
```



The screenshot shows a database query result interface with tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with icons for adding, saving, deleting, and other actions. The main area displays a table with the following data:

	customer_id smallint	sum numeric
1	187	110.81
2	148	110.78
3	211	108.77
4	522	102.80
5	526	101.78

At the bottom of the interface, it shows 'Total rows: 5 of 5' and 'Query complete 00:00:00.041'.