GROUP BY collapses. PARTITION BY keeps every row.

D8A.ACADEMY/RESOURCES/GROUPBY-VS-PARTITION

01 Two different questions

Ask what shape of answer you need before you write a single line.

GROUP BY	PARTITION BY
· One row per department	· All 200 rows survive
· Detail rows are gone	· Each row carries its context
· "What is the average per dept?"	· "How does each person compare?"
· Aggregation, full stop	· A window, not a collapse

→ The word "per" in the question usually means GROUP BY. The words "vs its" or "compared to" mean a window.

02 The collapse

Three departments in, three rows out.

```sql
SELECT dept,
       AVG(salary) AS avg_salary
FROM employees
GROUP BY dept;

-- 200 employees -> 3 rows
-- who earns what? gone.
```

03 The window

Every employee, next to their own department average.

```sql
SELECT name, dept, salary,
       AVG(salary) OVER (
         PARTITION BY dept
       ) AS dept_avg,
       salary - AVG(salary) OVER (
         PARTITION BY dept
       ) AS vs_dept
FROM employees;
```

→ This is the interview classic: "show each employee against their department average". No self-join needed.

⚠ COMMON TRAPS

04 Trap: filtering on a window

WHERE runs before window functions exist. It cannot see them.

```sql
-- broken: WHERE can't see dept_avg
WHERE salary > dept_avg

-- works: wrap it, then filter
WITH ranked AS ( ...OVER()... )
SELECT * FROM ranked
WHERE salary > dept_avg;
```

→ Same story as WHERE vs HAVING in sheet #018: the execution order is the explanation, every time.