CTE or subquery? When to name a step.

D8A.ACADEMY/RESOURCES/CTE-VS-SUBQUERY

THE CTE

`WITH step AS (…)`	name a step
`SELECT … FROM step`	read top-down
`WITH a AS(), b AS()`	chain steps

THE SUBQUERY

`(SELECT …) AS t`	inline, one use
`IN (SELECT id …)`	filter by lookup
`EXISTS (SELECT 1)`	presence check

CHOOSE

used twice?	CTE
one-line lookup?	subquery
debugging?	CTE, run each step
3 levels deep?	CTE, always

INTERVIEW LINE

top-N per group	CTE + ROW_NUMBER
`WHERE rn = 1`	filter outside
vs self-join	CTE wins on reads

PERFORMANCE

CTE ≈ subquery	engines inline both
`AS MATERIALIZED`	force compute once
correlated sub	runs once per row

GOTCHAS

`NOT IN` + NULL	← returns nothing.
sub in SELECT	row-by-row trap
`WITH RECURSIVE`	exists, rare

WORTH MEMORIZING

01 Top-N per group, the CTE way

The interview classic: latest order per customer. Name the ranking step, then filter it.

```
WITH ranked AS (
  SELECT customer_id, order_date,
         ROW_NUMBER() OVER (
           PARTITION BY customer_id
           ORDER BY order_date DESC) AS rn
  FROM orders)
SELECT * FROM ranked WHERE rn = 1;
```

→ You cannot filter on ROW_NUMBER in the same SELECT. The CTE is what makes WHERE rn = 1 legal.

02 Trap: NOT IN meets NULL

One NULL in the subquery and the whole result is empty. No error, no warning.

YOU WROTE	WHAT HAPPENS	WRITE INSTEAD
`NOT IN (SELECT ref_id …)`	one NULL = zero rows back	`NOT EXISTS (SELECT 1 …)`
`(SELECT SUM(…)) per row`	reruns for every row	JOIN a grouped CTE
4 subqueries nested	unreadable in a week	one CTE per step

→ x NOT IN (1, 2, NULL) is never true in SQL logic. NOT EXISTS treats NULLs the way you meant.