

# One table. Two names. One join.

D8A.ACADEMY/RESOURCES/SALARY-ABOVE-MANAGER

## THE QUESTION

|           |                                   |
|-----------|-----------------------------------|
| employees | id, name, salary, manager_id      |
| find      | who earns more than their manager |

## THE ANSWER

|                           |                                |
|---------------------------|--------------------------------|
| FROM employees e          | the employee side              |
| JOIN employees m          | the same table, as the manager |
| ON e.manager_id = m.id    | ← the join is the whole answer |
| WHERE e.salary > m.salary | then compare                   |

## REMEMBER

|           |                                     |
|-----------|-------------------------------------|
| aliases   | two names, one table                |
| LEFT JOIN | if people with no manager must stay |

## WORTH MEMORIZING

### 01 The query

Same table twice, under two names. The ON clause says which side is the employee and which is the manager.

```
SELECT e.name AS employee,
       e.salary,
       m.name AS manager,
       m.salary AS manager_salary
FROM employees e
JOIN employees m
  ON e.manager_id = m.id
WHERE e.salary > m.salary;
```

→ Read the ON clause out loud: the employee row (e) carries a manager\_id, and it must match the id of the manager row (m). Reverse it and you list managers next to their reports, which is a different question.

### 02 Trap: the ON clause, backwards

Both joins run. Only one answers the question.

| ON CLAUSE           | E IS         | M IS                 |
|---------------------|--------------|----------------------|
| e.manager_id = m.id | the employee | their manager        |
| e.id = m.manager_id | the manager  | one of their reports |

→ Interview phrasing: I alias the table twice, join the employee's manager\_id to the manager's id, then filter on salary.