D8A CHEATSHEET · #015 · SQL

Why your query returns zero rows.

D8A.ACADEMY/RESOURCES/SQL-NULL-HANDLING

01 Why = NULL never matches

NULL is absence, and you cannot compare a value to absence.

```
email = NULL       -- unknown
email != NULL      -- unknown too
NULL = NULL        -- still unknown

-- WHERE keeps only TRUE rows.
-- unknown is not true. rows dropped.
```

→ SQL logic has three values: true, false, unknown. Any comparison with NULL is unknown, and unknown never survives a WHERE.

02 The 3-character fix

Ask about existence, not equality.

```
-- who has no email?
SELECT * FROM users
WHERE email IS NULL;

-- who has one?
WHERE email IS NOT NULL;
```

→ IS NULL is the only operator designed for absence: it always answers true or false, never unknown.

03 COALESCE: absence with a face

First non-null value wins.

```
SELECT COALESCE(email, 'no email')
FROM users;

-- chains left to right
COALESCE(nickname, name, 'anon')
```

→ Perfect for display defaults and reports. Do not use it to paper over data-quality problems you should fix upstream.

⚠ COMMON TRAPS

04 Trap: NOT IN meets NULL

One NULL in the list and every row disappears.

```
-- returns 0 rows if logs has a NULL
WHERE id NOT IN
  (SELECT ref_id FROM logs)

-- fix: exclude the NULLs
  (SELECT ref_id FROM logs
   WHERE ref_id IS NOT NULL)
```

→ id NOT IN (1, NULL) means id != 1 AND id != NULL, and that second test is unknown for every row. NOT EXISTS is immune.

05 Trap: aggregates skip NULLs

Same column, two different counts.

```
COUNT(*)       -- all rows
COUNT(email)   -- non-null only
AVG(score)     -- ignores NULLs

-- average over ALL rows instead:
SUM(score) / COUNT(*)
```

→ AVG divides by the non-null count. If NULL means zero in your data, say so explicitly with COALESCE(score, 0).