Your query is not slow. Your plan is.

D8A.ACADEMY/RESOURCES/SLOW-QUERY-DEBUG

FIRST MOVE

`EXPLAIN ANALYZE`	the real plan
`Seq Scan`	red flag on big tables
`rows: est vs real`	way off? stale stats

USUAL SUSPECTS

`SELECT *`	drags every column
`WHERE f(col) = x`	← kills the index.
`LIKE '%foo'`	no index for you

INDEXES

`index WHERE cols`	filters first
`composite order`	left prefix rules
`covering index`	skips the table

JOINS

`join on keys`	indexed both sides
`EXISTS over IN`	for big subqueries
`filter early`	shrink, then join

QUICK WINS

`ANALYZE table`	refresh stats
`LIMIT while debug`	iterate fast
`drop dead indexes`	writes pay for them

STILL SLOW?

`partition big facts`	prune whole chunks
`nightly rollup`	pre-aggregate
`cache the answer`	← fastest: no query.

WORTH MEMORIZING

01 The index killer

Wrap the column in a function and the database cannot use its index on it.

```
-- 41s: function on the column, index ignored
WHERE DATE(created_at) = '2026-09-10'

-- 0.2s: same rows, index range scan
WHERE created_at >= '2026-09-10'
  AND created_at <  '2026-09-11'
```

→ Same trap in every dialect: LOWER(email), CAST(id AS TEXT), date math. Rewrite around the column, not on it.

02 Trap: reading the plan wrong

The plan tells you exactly what is slow. Three lines cover most real cases.

IN THE PLAN	IT MEANS	THE MOVE
`Seq Scan on 40M rows`	no usable index for the filter	index the WHERE cols
`rows=100, actual=2M`	stats are stale, plan is blind	ANALYZE the table
`Nested Loop on huge sets`	join order went wrong	check keys + filters

→ Fix the biggest node first. A 41s Seq Scan makes every other optimization a rounding error.