When indexes help, when they hurt.

D8A.ACADEMY/RESOURCES/SQL-INDEXING

01 Why your index sat idle

The engine can only seek in what it has sorted.

```
-- the index stores created_at, sorted
-- YEAR(created_at) is a NEW value

WHERE YEAR(created_at) = 2026
-- computed row by row: full scan
```

→ An index is sorted on raw column values. Wrap the column in a function and that ordering no longer applies, so the engine walks every row instead.

02 The sargable rewrite

Ask for a range of raw values, not a computed one.

```
SELECT * FROM orders
WHERE created_at >= '2026-01-01'
  AND created_at <  '2027-01-01';

-- Index Scan · 0.2s
```

→ Same rows, same meaning, but the condition now compares the indexed column directly, so the engine jumps straight to the range. Works for dates, prefixes, any range.

03 What an index actually is

A sorted copy of one or a few columns, with pointers back to the rows.

```
CREATE INDEX idx_orders_created
ON orders (created_at);

-- lookup: a few B-tree hops,
-- not a 12M-row walk
```

→ Like a book index: find the entry in the sorted list, jump to the page. The table itself stays unsorted; the index is a side structure the engine maintains for you.

⚠ COMMON TRAPS

04 Trap: the leftmost prefix

A composite index only works from its first column.

```
CREATE INDEX idx ON orders
  (customer_id, created_at);

WHERE customer_id = 42    -- uses idx
WHERE created_at > '2026-01-01'
                          -- ignores idx
```

→ The index is sorted by customer_id first, created_at second. Like a phone book sorted by last name: useless for searching first names alone.

05 Trap: indexes are not free

Every write updates every index on the table.

`writes`	each INSERT / UPDATE maintains every index
`low cardinality`	an index on a 3-value column barely helps
`blind adds`	unused indexes cost writes and give nothing
`the check`	EXPLAIN: Seq Scan vs Index Scan, no guessing

→ Index the columns your WHERE and JOIN clauses actually use, verify with EXPLAIN, and drop the indexes no query ever touches.