Month 10 retention went up.

D8A.ACADEMY/RESOURCES/COHORT-RETENTION-SQL

THE TABLE

`cohort`	month of the first event, per user
`month_n`	months since that: 0, 1, 2 …
`month 0`	← 100%, or the join is wrong

THE QUERY

`MIN(ts) per user`	one row per user: the cohort
`join activity`	months between activity and cohort
`COUNT(DISTINCT)`	users per cell, not events

REMEMBER

`the triangle`	young cohorts have no month 6 yet
`curve going up`	← only old cohorts reach month 10

WORTH MEMORIZING

01 The cohort table, one query

One row per user for the cohort, one row per user and month for the activity, a join, a count.

```sql
WITH cohorts AS (
  SELECT user_id,
         MIN(DATE_TRUNC('month', ts)) AS cohort
  FROM events GROUP BY user_id
), activity AS (
  SELECT DISTINCT user_id,
         DATE_TRUNC('month', ts) AS month
  FROM events
)
SELECT c.cohort,
       EXTRACT(YEAR FROM AGE(a.month, c.cohort)) * 12
     + EXTRACT(MONTH FROM AGE(a.month, c.cohort))
       AS month_n,
       COUNT(DISTINCT a.user_id) AS users
FROM cohorts c JOIN activity a USING (user_id)
GROUP BY 1, 2 ORDER BY 1, 2;
```

→ Postgres. Percentages: divide each cell by the cohort's month 0, FIRST_VALUE(users) OVER (PARTITION BY cohort ORDER BY month_n).

02 Trap: three ways the table lies

Each one looks like a retention story. Each one is a query story.

YOU SEE	CAUSE	FIX
`month 0 = 83%`	cohort from signups, activity from events	`both from the same table`
`month 1 = 140%`	COUNT(*) on events, not users	`COUNT(DISTINCT user_id)`
`month 10 jumps up`	only the 2025 cohorts are 10 months old	`show n per cell, cut the tail`

→ The blended curve averages over whoever has reached that age. At month 10 that is the early adopters only. The curve did not improve, the mix did.