Interview Questions

# User Retention SQL Interview Question: Month-1 Retention

Retention is one of the most business-critical metrics a data team tracks, which is exactly why it shows up so often in SQL interviews for data analyst, data science, and growth roles.

## Why Interviewers Ask This

Anyone can write a basic query. Retention questions test whether a candidate can translate a fuzzy business definition, such as whether users are sticking around, into precise SQL logic, and whether they know where the classic traps are hiding.

## The Question

Given a table of user logins, calculate Month-1 retention: of the users who were active in January 2024, what percentage were also active in February 2024?

### Schema

```
CREATE TABLE logins (
  id INTEGER PRIMARY KEY,
  user_id INTEGER,
  login_date DATE
);
```

### Sample Data

| USER_ID | LOGIN_DATE |
|---|---|
| 1 | 2024-01-05 |
| 1 | 2024-02-10 |
| 2 | 2024-01-08 |

| USER_ID | LOGIN_DATE |
|---|---|
| 3 | 2024-01-20 |
| 3 | 2024-02-02 |

In this sample, users 1 and 3 return in February while user 2 does not, so retention comes out to 2 out of 3, roughly 66.67 percent.

## Step-by-Step Approach

1. Get the distinct set of users active in January. A user with ten January logins should still only count once.
2. Get the distinct set of users active in February.
3. Count how many January users also appear in the February set.
4. Divide retained users by total January users, and force floating point division, or the result silently truncates to an integer.

## SQLite Solution

```
WITH jan_users AS (
  SELECT DISTINCT user_id
  FROM logins
  WHERE STRFTIME('%Y-%m', login_date) = '2024-01'
),
feb_users AS (
  SELECT DISTINCT user_id
  FROM logins
  WHERE STRFTIME('%Y-%m', login_date) = '2024-02'
)
SELECT
  (SELECT COUNT(*) FROM jan_users) AS jan_active_users,
  (SELECT COUNT(*) FROM jan_users j
     JOIN feb_users f ON j.user_id = f.user_id) AS retained_users,
  ROUND(
    100.0 * (SELECT COUNT(*) FROM jan_users j JOIN feb_users f ON j.user_id = f.user_id)
    / (SELECT COUNT(*) FROM jan_users), 2
  ) AS retention_rate_pct;
```

## PostgreSQL Solution

```
WITH jan_users AS (
  SELECT DISTINCT user_id
  FROM logins
```

```
  WHERE TO_CHAR(login_date, 'YYYY-MM') = '2024-01'
),
feb_users AS (
  SELECT DISTINCT user_id
  FROM logins
  WHERE TO_CHAR(login_date, 'YYYY-MM') = '2024-02'
)
SELECT
  (SELECT COUNT(*) FROM jan_users) AS jan_active_users,
  (SELECT COUNT(*) FROM jan_users j
    JOIN feb_users f ON j.user_id = f.user_id) AS retained_users,
  ROUND(
    (100.0 * (SELECT COUNT(*) FROM jan_users j JOIN feb_users f ON j.user_id = f.user_id)
    / (SELECT COUNT(*) FROM jan_users))::NUMERIC, 2
  ) AS retention_rate_pct;
```

The month-matching logic swaps from SQLite STRFTIME to PostgreSQL TO_CHAR. Everything else is structurally identical.

## Common Mistakes

1. Integer division. Writing 100 times retained divided by total instead of 100.0 times retained divided by total truncates the result to a whole number in engines that perform integer division on two integers. This is the most common silent bug in retention queries.
2. Not deduplicating logins. Skipping DISTINCT means a power user with fifteen January logins gets overcounted relative to someone who logs in once.
3. Ambiguous month boundaries. Active in February could mean the calendar month or a rolling 30 day window from first login. Always clarify which definition is expected before writing a single line of SQL, since this is a strong signal of seniority.
4. Wrong direction on the join. Retention is retained users divided by starting cohort size, not the reverse. Double check the denominator.

## Common Variants to Prepare For

- Day-1 retention: the same logic, but swap the month grouping for an exact next-day login check, common for mobile apps.
- Full cohort retention table: extend this into a matrix showing retention for months zero through N since signup, not just month one.
- Churn rate: simply 100 minus the retention rate, though interviewers sometimes ask for it to be derived from scratch to check understanding of the relationship.

## Frequently Asked Questions

### What is the difference between retention and churn?

Retention is the percentage of users who stay active, while churn is the percentage who stop. They are complements, since churn equals 100 minus retention, but interviewers sometimes ask for each to be computed independently to check reasoning.

### How do you build a full retention cohort table in SQL?

Extend the same logic across every month offset since a user first logged in, typically using a self-join or a date difference calculation, then pivot the results into a cohort-by-month grid.

### Why do interviewers care about retention queries specifically?

Retention sits at the center of most growth and product analytics roles. A candidate who can reason precisely about cohorts and time windows is demonstrating exactly the skill the role requires, not just SQL syntax knowledge.

## Keep Practicing

Retention logic is easy to follow on paper and easy to fumble under pressure. Practice cohort and retention style SQL puzzles with our SQL practice questions until the self-join and division logic become second nature, or explore full scenarios in our case studies.