Great question 🔥 You picked a classic **"Nth day login / consecutive login"** type of SQL puzzle. Let's break this query step by step like I'm teaching you in a DSA way (inputs → operations → outputs).

The Query

sql Copy code

```sql
SELECT ROUND(
    COUNT(DISTINCT player_id)/(SELECT COUNT(DISTINCT player_id) FROM Activity), 2)
    AS fraction
FROM Activity
WHERE (player_id, DATE_SUB(event_date, INTERVAL 1 DAY)) IN(
    SELECT player_id, MIN(event_date) AS first_login
    FROM Activity
    GROUP BY player_id
);
```

1️⃣ Understand the problem

We have a table `Activity` with:

- `player_id` → user
- `event_date` → when they logged in

We want to find the **fraction of players who logged in again the day after their very first login**.

2️⃣ Break the query into parts

🔹 Part A: Inner Subquery

sql

```sql
SELECT player_id, MIN(event_date) AS first_login
FROM Activity
GROUP BY player_id;
```

- For each player, find the earliest (minimum) `event_date`.
- This gives **each player's first login date**.

👉 Example:

player_id	event_date
1	2020-01-01
1	2020-01-02
2	2020-01-05
2	2020-01-06

Subquery result:

player_id	first_login
1	2020-01-01
2	2020-01-05

🔹 Part B: WHERE condition

sql

```sql
WHERE (player_id, DATE_SUB(event_date, INTERVAL 1 DAY)) IN (...)
```

- For each record in `Activity`, check:
  - Take its `event_date`
  - Subtract **1 day**
  - See if `(player_id, that_date)` matches a `(player_id, first_login)` pair from subquery.

⚡ Meaning:
We're checking if the current login happened **the day after the player's first login**.

👉 Continuing example:

- For player 1 → second row `2020-01-02`, `DATE_SUB(2020-01-02, 1 DAY)` = `2020-01-01`. Matches `first_login` ✅
- For player 2 → second row `2020-01-06`, `DATE_SUB(2020-01-06, 1 DAY)` = `2020-01-05`. Matches `first_login` ✅

So this WHERE keeps those rows.

🔹 Part C: Outer SELECT fraction

sql

```sql
COUNT(DISTINCT player_id) / (SELECT COUNT(DISTINCT player_id) FROM Activity)
```

- **Numerator** → how many distinct players had a login on "day after first login".
- **Denominator** → total number of distinct players.
- `ROUND(..., 2)` → round to 2 decimal places.

3️⃣ Execution Flow

1. Subquery finds **first login per player**.
2. Main query scans `Activity` rows, and keeps only those where login = **first_login + 1 day**.
3. Count distinct players in that filtered set.
4. Divide by total players.
5. Round to 2 decimal places.

4️⃣ Example Final Walkthrough

Input `Activity`:

player_id	event_date
1	2020-01-01
1	2020-01-02
2	2020-01-05
2	2020-01-07
3	2020-01-10

- Player 1 → first login `2020-01-01`. They logged in on `2020-01-02` ✅
- Player 2 → first login `2020-01-05`. Next login was `2020-01-07` ❌ (not exactly 1 day after).
- Player 3 → only 1 login ❌

👉 Numerator = 1 (only player 1 qualifies)
👉 Denominator = 3
👉 Fraction = `1/3 = 0.33`

5️⃣ Why it works

- Subquery isolates the **first_login**.
- DATE_SUB trick checks if later logins are **first_login + 1 day**.
- Aggregation gives fraction.

🔥 That's how execution happens step by step.