DESCRIPTION

Table: Signups

```
+----------------+----------+
| Column Name    | Type     |
+----------------+----------+
| user_id        | int      |
| time_stamp     | datetime |
+----------------+----------+
```

user_id is the column of unique values for this table.

Each row contains information about the signup time for the user with ID user_id.

Table: Confirmations

```
+----------------+----------+
| Column Name    | Type     |
+----------------+----------+
| user_id        | int      |
| time_stamp     | datetime |
| action         | ENUM     |
+----------------+----------+
```

(user_id, time_stamp) is the primary key (combination of columns with unique values) for this table.

user_id is a foreign key (reference column) to the Signups table.

action is an ENUM (category) of the type ('confirmed', 'timeout')

Each row of this table indicates that the user with ID user_id requested a confirmation message at time_stamp and that confirmation message was either confirmed ('confirmed') or expired without confirming ('timeout').

The **confirmation rate** of a user is the number of 'confirmed' messages divided by the total number of requested confirmation messages. The confirmation rate of a user that did not request any confirmation messages is 0. Round the confirmation rate to **two decimal** places.

Write a solution to find the **confirmation rate** of each user.

Return the result table in **any order**.

The result format is in the following example.

Example 1:

Input:

Signups table:

```
+---------+---------------------+
| user_id | time_stamp          |
+---------+---------------------+
| 3       | 2020-03-21 10:16:13 |
| 7       | 2020-01-04 13:57:59 |
| 2       | 2020-07-29 23:09:44 |
| 6       | 2020-12-09 10:39:37 |
+---------+---------------------+
```

Confirmations table:

```
+---------+---------------------+-----------+
| user_id | time_stamp          | action    |
+---------+---------------------+-----------+
| 3       | 2021-01-06 03:30:46 | timeout   |
| 3       | 2021-07-14 14:00:00 | timeout   |
| 7       | 2021-06-12 11:57:29 | confirmed |
| 7       | 2021-06-13 12:58:28 | confirmed |
| 7       | 2021-06-14 13:59:27 | confirmed |
| 2       | 2021-01-22 00:00:00 | confirmed |
| 2       | 2021-02-28 23:59:59 | timeout   |
+---------+---------------------+-----------+
```

Output:

```
+---------+-------------------+
| user_id | confirmation_rate |
+---------+-------------------+
| 6       | 0.00              |
| 3       | 0.00              |
| 7       | 1.00              |
| 2       | 0.50              |
+---------+-------------------+
```

Explanation:

User 6 did not request any confirmation messages. The confirmation rate is 0.

User 3 made 2 requests and both timed out. The confirmation rate is 0.

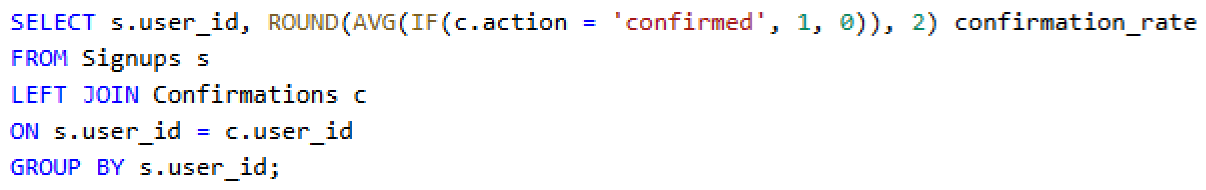
User 7 made 3 requests and all were confirmed. The confirmation rate is 1.

User 2 made 2 requests where one was confirmed and the other timed out. The confirmation rate is 1 / 2 = 0.5.

SOLUTION

MySQL:

- SELECT user_id and confirmation_rate if action equals ‘confirmed’, then 1 else 0 using IF()
- Find Average of ‘confirmed’ counts using AVG() and round the result to 2 decimals using ROUND()
- JOIN Signups and Confirmations using LEFT JOIN



```sql
SELECT s.user_id, ROUND(AVG(IF(c.action = 'confirmed', 1, 0)), 2) confirmation_rate
FROM Signups s
LEFT JOIN Confirmations c
ON s.user_id = c.user_id
GROUP BY s.user_id;
```

PostgreSQL:

Option 1:

- SELECT user_id and confirmation_rate when action equals ‘confirmed’, then 1 else 0 using CASE
- Find Average of ‘confirmed’ counts using AVG() and round the result to 2 decimals using ROUND()
- JOIN Signups and Confirmations using LEFT JOIN

```sql
SELECT s.user_id, ROUND(AVG(CASE WHEN c.action = 'confirmed' THEN 1 ELSE 0 END), 2)
confirmation_rate
FROM Signups s
LEFT JOIN Confirmations c
ON c.user_id = s.user_id
GROUP BY 1;
```

Option 2:

- Using CTE (WITH)

```sql
WITH t AS(
SELECT s.user_id, COUNT(c.user_id) request, CASE WHEN c.action = 'confirmed' THEN COUNT(c.
action) ELSE 0 END AS status
FROM Signups s
LEFT JOIN Confirmations c
ON c.user_id = s.user_id
GROUP BY 1, c.action)
SELECT user_id, ROUND(SUM(status)/SUM(request+0.0001), 2) confirmation_rate
FROM t
GROUP BY 1;
```