/\* \*\*\*\*597. Friend Requests I: Overall Acceptance Rate\*\*\*\*\*\*\*/

In social network like Facebook or Twitter, people send friend requests and accept others’ requests as well. Now given two tables as below:

Table: friend\_request

| sender\_id | send\_to\_id |request\_date|

|-----------|------------|------------|

| 1 | 2 | 2016\_06-01 |

| 1 | 3 | 2016\_06-01 |

| 1 | 4 | 2016\_06-01 |

| 2 | 3 | 2016\_06-02 |

| 3 | 4 | 2016-06-09 |

Table: request\_accepted

| requester\_id | accepter\_id |accept\_date |

|--------------|-------------|------------|

| 1 | 2 | 2016\_06-03 |

| 1 | 3 | 2016-06-08 |

| 2 | 3 | 2016-06-08 |

| 3 | 4 | 2016-06-09 |

| 3 | 4 | 2016-06-10 |

Write a query to find the overall acceptance rate of requests rounded to 2 decimals, which is the number of acceptance divide the number of requests.

For the sample data above, your query should return the following result.

|accept\_rate|

|-----------|

| 0.80|

Note:

The accepted requests are not necessarily from the table friend\_request. In this case, you just need to simply count the total accepted requests

(no matter whether they are in the original requests), and divide it by the number of requests to get the acceptance rate.

It is possible that a sender sends multiple requests to the same receiver, and a request could be accepted more than once.

In this case, the ‘duplicated’ requests or acceptances are only counted once.

If there is no requests at all, you should return 0.00 as the accept\_rate.

Explanation: There are 4 unique accepted requests, and there are 5 requests in total. So the rate is 0.80.

Follow-up:

Can you write a query to return the accept rate but for every month?

How about the cumulative accept rate for every day?

select coalesce(round(count(distinct requester\_id, accepter\_id)/count(distinct sender\_id, send\_to\_id),2),0) as accept\_rate

from friend\_request, request\_accepted;

/\*The COALESCE() function returns the first non-null value in a list.\*/

/\* https://www.w3schools.com/sql/func\_mysql\_coalesce.asp \*/

select

round(

ifnull(

(select count(\*) from (select distinct requester\_id, accepter\_id from request\_accepted) as A)

/

(select count(\*) from (select distinct sender\_id, send\_to\_id from friend\_request) as B),

0)

, 2) as accept\_rate;

/\* COUNT(\*) 函数返回表中的记录数：\*/

select

round(

ifnull(

(select count(distinct requester\_id, accepter\_id) from request\_accepted as A)

/

(select count(distinct sender\_id, send\_to\_id) from friend\_request as B),

0)

, 2) as accept\_rate;

SELECT IF(req.req = 0, 0.00, round(acp.acp / req.req, 2)) AS accept\_rate

FROM (SELECT COUNT(DISTINCT requester\_id, accepter\_id) AS acp FROM request\_accepted) acp,

(SELECT COUNT(DISTINCT sender\_id, send\_to\_id) AS req FROM friend\_request) req

/\*

# Follow-up 1: return the accept rate for every month\*/

SELECT IF(req.req = 0, 0.00, round(acp.acp / req.req, 2)) AS accept\_rate, acp.month

FROM (SELECT COUNT(DISTINCT requester\_id, accepter\_id) AS acp, Month(accept\_date) AS month FROM request\_accepted) acp,

(SELECT COUNT(DISTINCT sender\_id, send\_to\_id) AS req, Month(request\_date) AS month FROM friend\_request) req

WHERE acp.month = req.month

GROUP BY acp.month

/\* Result

{

"headers": ["accept\_rate", "month"],

"values": [[0.80, 6]]

}

\*/

SELECT IFNULL(round(acp.acp / req.req, 2)，0.00)

SELECT coalesce(round(acp.acp / req.req, 2),0.00)

/\*# Follow-up 2: return the cumulative accept rate for every day

# Without null check\*/

SELECT ROUND(

COUNT(DISTINCT requester\_id, accepter\_id) / COUNT(DISTINCT sender\_id, send\_to\_id), 2

) AS rate, date\_table.dates

FROM request\_accepted acp, friend\_request req,

(SELECT request\_date AS dates FROM friend\_request

UNION

SELECT accept\_date FROM request\_accepted

ORDER BY dates) as date\_table

WHERE acp.accept\_date <= date\_table.dates

AND req.request\_date <= date\_table.dates

GROUP BY date\_table.dates

/\* Result

{

"headers": ["rate", "dates"],

"values": [[0.25, "2016-06-03"], [0.75, "2016-06-08"], [0.80, "2016-06-09"], [0.80, "2016-06-10"]]

}

\*/