

Problem 1892: Page Recommendations II

Problem Information

Difficulty: Hard

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Table:

Friendship

+-----+-----+ | Column Name | Type | +-----+-----+ | user1_id | int || user2_id | int | +-----+-----+ (user1_id, user2_id) is the primary key (combination of columns with unique values) for this table. Each row of this table indicates that the users user1_id and user2_id are friends.

Table:

Likes

+-----+-----+ | Column Name | Type | +-----+-----+ | user_id | int | | page_id | int | +-----+-----+ (user_id, page_id) is the primary key (combination of columns with unique values) for this table. Each row of this table indicates that user_id likes page_id.

You are implementing a page recommendation system for a social media website. Your system will

recommend

a page to

user_id

if the page is

liked

by

at least one

friend of

user_id

and is

not liked

by

user_id

Write a solution to find all the possible

page recommendations

for every user. Each recommendation should appear as a row in the result table with these columns:

user_id

: The ID of the user that your system is making the recommendation to.

page_id

: The ID of the page that will be recommended to

user_id

friends_likes

: The number of the friends of

user_id

that like

page_id

Return the result table in

any order

The result format is in the following example.

Example 1:

Input:

Friendship table: +-----+-----+ | user1_id | user2_id | +-----+-----+ | 1 | 2 || 1 | 3 ||
1 | 4 || 2 | 3 || 2 | 4 || 2 | 5 || 6 | 1 | +-----+-----+ Likes table: +-----+-----+ | user_id
| page_id | +-----+-----+ | 1 | 88 || 2 | 23 || 3 | 24 || 4 | 56 || 5 | 11 || 6 | 33 || 2 | 77 || 3 |
77 || 6 | 88 | +-----+-----+

Output:

+-----+-----+-----+ | user_id | page_id | friends_likes |
+-----+-----+-----+ | 1 | 77 | 2 || 1 | 23 | 1 || 1 | 24 | 1 || 1 | 56 | 1 || 1 | 33 | 1 || 2 |
24 | 1 || 2 | 56 | 1 || 2 | 11 | 1 || 2 | 88 | 1 || 3 | 88 | 1 || 3 | 23 | 1 || 4 | 88 | 1 || 4 | 77 | 1 || 4 |
23 | 1 || 5 | 77 | 1 || 5 | 23 | 1 | +-----+-----+

Explanation:

Take user 1 as an example: - User 1 is friends with users 2, 3, 4, and 6. - Recommended pages are 23 (user 2 liked it), 24 (user 3 liked it), 56 (user 3 liked it), 33 (user 6 liked it), and 77 (user 2 and user 3 liked it). - Note that page 88 is not recommended because user 1 already liked it.

Another example is user 6: - User 6 is friends with user 1. - User 1 only liked page 88, but user 6 already liked it. Hence, user 6 has no recommendations.

You can recommend pages for users 2, 3, 4, and 5 using a similar process.

Code Snippets

MySQL:

```
# Write your MySQL query statement below
```

MS SQL Server:

```
/* Write your T-SQL query statement below */
```

PostgreSQL:

```
-- Write your PostgreSQL query statement below
```

Oracle:

```
/* Write your PL/SQL query statement below */
```

Pandas:

```
import pandas as pd

def recommend_page(friendship: pd.DataFrame, likes: pd.DataFrame) ->
    pd.DataFrame:
```

Solutions

MySQL Solution:

```
# Write your MySQL query statement below
```

MS SQL Server Solution:

```
/* Write your T-SQL query statement below */
```

PostgreSQL Solution:

```
-- Write your PostgreSQL query statement below
```

Oracle Solution:

```
/* Write your PL/SQL query statement below */
```

Pandas Solution:

```
import pandas as pd

def recommend_page(friendship: pd.DataFrame, likes: pd.DataFrame) ->
    pd.DataFrame:
```