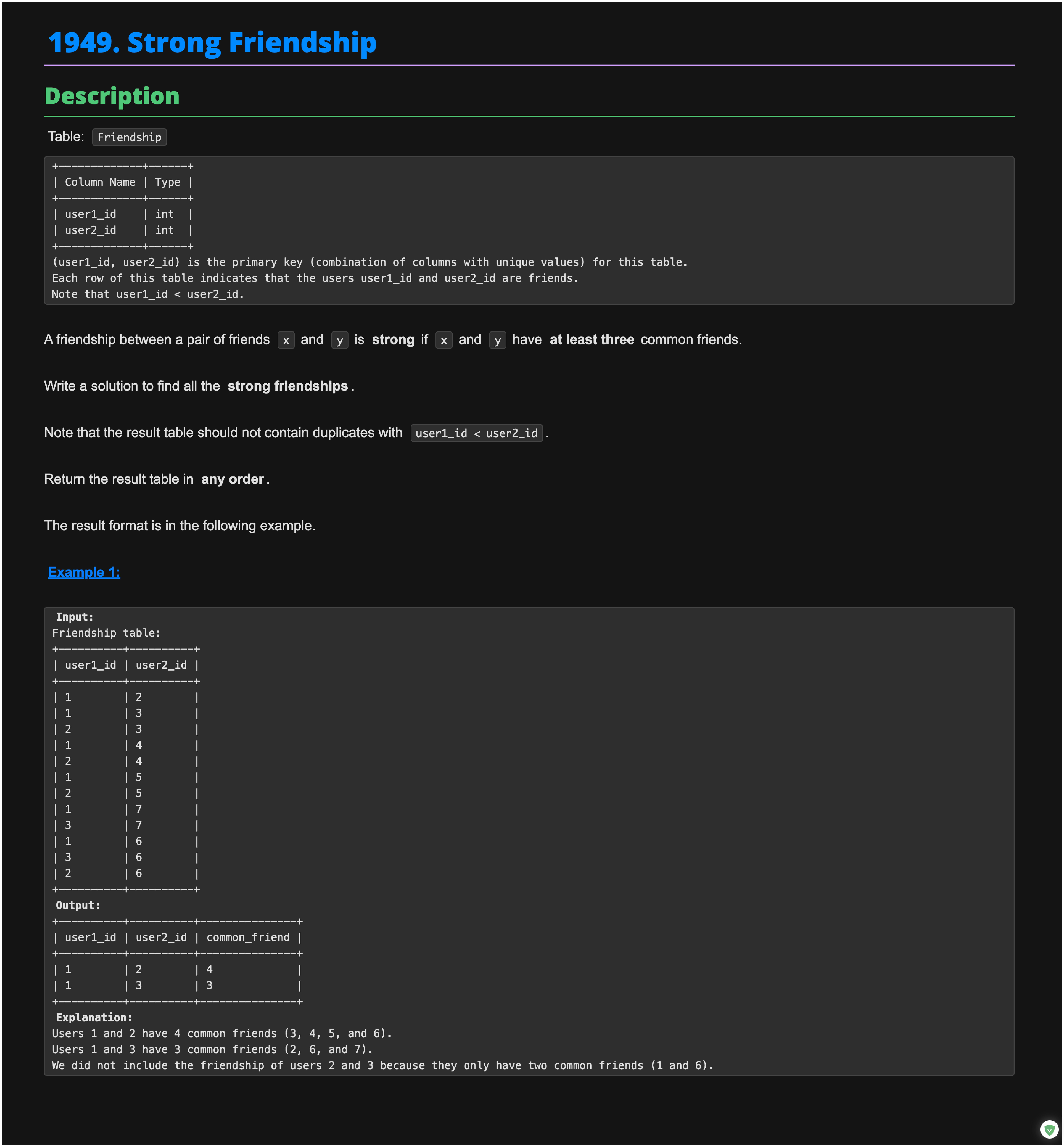

1949. Strong Friendship

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.
Note that user1_id < user2_id.
```

A friendship between a pair of friends `x` and `y` is **strong** if `x` and `y` have **at least three** common friends.

Write a solution to find all the **strong friendships**.

Note that the result table should not contain duplicates with `user1_id < user2_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        |
| 2        | 3        |
| 1        | 4        |
| 2        | 4        |
| 1        | 5        |
| 2        | 5        |
| 1        | 7        |
| 3        | 7        |
| 1        | 6        |
| 3        | 6        |
| 2        | 6        |
+----------+----------+
Output:
+----------+----------+---------------+
| user1_id | user2_id | common_friend |
+----------+----------+---------------+
| 1        | 2        | 4             |
| 1        | 3        | 3             |
+----------+----------+---------------+
Explanation:
Users 1 and 2 have 4 common friends (3, 4, 5, and 6).
Users 1 and 3 have 3 common friends (2, 6, and 7).
We did not include the friendship of users 2 and 3 because they only have two common friends (1 and 6).
```