Problem 3554: Find Category Recommendation Pairs

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Table:

ProductPurchases

+-------------+------+ | Column Name | Type | +-------------+------+ | user_id | int | | product_id | int | | quantity | int | +-------------+------+ (user_id, product_id) is the unique identifier for this table. Each row represents a purchase of a product by a user in a specific quantity.

Table:

ProductInfo

+-------------+---------+ | Column Name | Type | +-------------+---------+ | product_id | int | | category | varchar | | price | decimal | +-------------+---------+ product_id is the unique identifier for this table. Each row assigns a category and price to a product.

Amazon wants to understand shopping patterns across product categories. Write a solution to:

Find all

category pairs

(where

category1

<

category2

)

For

each category pair

, determine the number of

unique

customers

who purchased products from

both

categories

A category pair is considered

reportable

if at least

3

different customers have purchased products from both categories.

Return

the result table of reportable category pairs ordered by

customer_count

in

descending

order, and in case of a tie, by

category1

in

ascending

order lexicographically, and then by

category2

in

ascending

order.

The result format is in the following example.

Example:

Input:

ProductPurchases table:

+---------+------------+----------+ | user_id | product_id | quantity | +---------+------------+----------+ | 1 | 101 | 2 | | 1 | 102 | 1 | | 1 | 201 | 3 | | 1 | 301 | 1 | | 2 | 101 | 1 | | 2 | 102 | 2 | | 2 | 103 | 1 | | 2 | 201 | 5 | | 3 | 101 | 2 | | 3 | 103 | 1 | | 3 | 301 | 4 | | 3 | 401 | 2 | | 4 | 101 | 1 | | 4 | 201 | 3 | | 4 | 301 | 1 | | 4 | 401 | 2 | | 5 | 102 | 2 | | 5 | 103 | 1 | | 5 | 201 | 2 | | 5 | 202 | 3 | +---------+------------+----------+

ProductInfo table:

+------------+-------------+-------+ | product_id | category | price | +------------+-------------+-------+ | 101 | Electronics | 100 | | 102 | Books | 20 | | 103 | Books | 35 | | 201 | Clothing | 45 | | 202 | Clothing | 60 | | 301 | Sports | 75 | | 401 | Kitchen | 50 | +------------+-------------+-------+

Output:

+-------------+-------------+----------------+ | category1 | category2 | customer_count | +-------------+-------------+----------------+ | Books | Clothing | 3 | | Books | Electronics | 3 | | Clothing | Electronics | 3 | | Electronics | Sports | 3 | +-------------+-------------+----------------+

Explanation:

Books-Clothing

:

User 1 purchased products from Books (102) and Clothing (201)

User 2 purchased products from Books (102, 103) and Clothing (201)

User 5 purchased products from Books (102, 103) and Clothing (201, 202)

Total: 3 customers purchased from both categories

Books-Electronics

:

User 1 purchased products from Books (102) and Electronics (101)

User 2 purchased products from Books (102, 103) and Electronics (101)

User 3 purchased products from Books (103) and Electronics (101)

Total: 3 customers purchased from both categories

Clothing-Electronics

:

User 1 purchased products from Clothing (201) and Electronics (101)

User 2 purchased products from Clothing (201) and Electronics (101)

User 4 purchased products from Clothing (201) and Electronics (101)

Total: 3 customers purchased from both categories

Electronics-Sports

:

User 1 purchased products from Electronics (101) and Sports (301)

User 3 purchased products from Electronics (101) and Sports (301)

User 4 purchased products from Electronics (101) and Sports (301)

Total: 3 customers purchased from both categories

Other category pairs like Clothing-Sports (only 2 customers: Users 1 and 4) and Books-Kitchen (only 1 customer: User 3) have fewer than 3 shared customers and are not included in the result.

The result is ordered by customer_count in descending order. Since all pairs have the same customer_count of 3, they are ordered by category1 (then category2) in ascending order.

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 find_category_recommendation_pairs(product_purchases: pd.DataFrame,
product_info: 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 find_category_recommendation_pairs(product_purchases: pd.DataFrame,
product_info: pd.DataFrame) -> pd.DataFrame:
```