Problem 3230: Customer Purchasing Behavior Analysis

Problem Information

Difficulty: Medium
Acceptance Rate: 36.31%
Paid Only: Yes
Tags: Database

Problem Description

Table: `Transactions`

+------------------+---------+ | Column Name | Type | +------------------+---------+ | transaction_id | int | | customer_id | int | | product_id | int | | transaction_date | date | | amount | decimal | +------------------+---------+ transaction_id is the unique identifier for this table. Each row of this table contains information about a transaction, including the customer ID, product ID, date, and amount spent.

Table: `Products`

+-------------+---------+ | Column Name | Type | +-------------+---------+ | product_id | int | | category | varchar | | price | decimal | +-------------+---------+ product_id is the unique identifier for this table. Each row of this table contains information about a product, including its category and price.

Write a solution to analyze customer purchasing behavior. For **each customer** , calculate:

* The total amount spent. * The number of transactions. * The number of **unique** product categories purchased. * The average amount spent. * The **most frequently** purchased product category (if there is a tie, choose the one with the most recent transaction). * A **loyalty score** defined as: (Number of transactions * 10) + (Total amount spent / 100).

Round `total_amount`, `avg_transaction_amount`, and `loyalty_score` to `2` decimal places.

Return _the result table ordered by_ `loyalty_score` _in**descending** order_, _then by_`customer_id` _in**ascending** order_.

The query result format is in the following example.

Example:

Input:

`Transactions` table:

+----------------+-------------+------------+------------------+--------+ | transaction_id | customer_id | product_id | transaction_date | amount | +----------------+-------------+------------+------------------+--------+ | 1 | 101 | 1 | 2023-01-01 | 100.00 | | 2 | 101 | 2 | 2023-01-15 | 150.00 | | 3 | 102 | 1 | 2023-01-01 | 100.00 | | 4 | 102 | 3 | 2023-01-22 | 200.00 | | 5 | 101 | 3 | 2023-02-10 | 200.00 | +----------------+-------------+------------+------------------+--------+

`Products` table:

+------------+----------+--------+ | product_id | category | price | +------------+----------+--------+ | 1 | A | 100.00 | | 2 | B | 150.00 | | 3 | C | 200.00 | +------------+----------+--------+

Output:

+-------------+--------------+-------------------+-------------------+------------------------+--------------+---------------+ | customer_id | total_amount | transaction_count | unique_categories | avg_transaction_amount | top_category | loyalty_score | +-------------+--------------+-------------------+-------------------+------------------------+--------------+---------------+ | 101 | 450.00 | 3 | 3 | 150.00 | C | 34.50 | | 102 | 300.00 | 2 | 2 | 150.00 | C | 23.00 | +-------------+--------------+-------------------+-------------------+------------------------+--------------+---------------+

Explanation:

* For customer 101: * Total amount spent: 100.00 + 150.00 + 200.00 = 450.00 * Number of transactions: 3 * Unique categories: A, B, C (3 categories) * Average transaction amount: 450.00 / 3 = 150.00 * Top category: C (Customer 101 made 1 purchase each in categories A, B, and C. Since the count is the same for all categories, we choose the most recent transaction, which is category C on 2023-02-10) * Loyalty score: (3 * 10) + (450.00 / 100) = 34.50 * For customer 102: * Total amount spent: 100.00 + 200.00 = 300.00 * Number of

transactions: 2 * Unique categories: A, C (2 categories) * Average transaction amount: 300.00 / 2 = 150.00 * Top category: C (Customer 102 made 1 purchase each in categories A and C. Since the count is the same for both categories, we choose the most recent transaction, which is category C on 2023-01-22) * Loyalty score: (2 * 10) + (300.00 / 100) = 23.00

Note: The output is ordered by loyalty_score in descending order, then by customer_id 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
```