Seen Data Engineering - SQL Coding Exercise

Introduction

This exercise is designed to assess your understanding and practical proficiency in SQL, specifically focusing on your ability to manipulate and modify tables to solve business problems. The data used to complete this exercise can be found in a SQLite database, which can be downloaded here, and a schema description (Seen Warehouse Schema). Please return to us your responses, in the form of SQL scripts answering the questions below.

Questions

1. Write a query to determine the total balances each customer owes at the end of each day. The balance starts at 0 and changes with each transaction. Output columns should include:

   `customer_id` , `snapshot_date` , `customer_balance`

2. Write a query that lists unusual transactions. It's up to you to define what unusual means. You might want to use some statistical functions.

3. Using the data in “page_view_events”, what are the 5 most common paths that people follow in the app? Please keep in mind that we are looking for *complete* paths. (a → b → c).

4. Using the data in “page_view_events”, write a query that can be used to display the following funnel:
    a. /product-disclosures (first page)
    b. /input-email
    c. /input-password
    d. /verify-email
    e. /input-phone
    f. /input-sms
    g. /kyc-info
    h. /input-name
    i. /input-address
    j. /date-of-birth
    k. /input-ssn
    l. /input-income
    m. /confirm-details (last page)

    The order of the steps is outlined above. Please create a query that outputs the page name, the count of visitors that entered the step, and the percentage which dropped off before the next step.

5. Using the data within the “accounts_days_past_due” table , write a query that shows the delinquency periods for a customer, using the context in the above section as a guide. Your output should one row per delinquency period, with

the following columns:

`customer_id`, `account_id`, `delinquency_period`, `delinquency_start_date`, `delinquency_end_date`

a. What was the total count of rows in your output?

Additional Context:

Transactions

This table contains the full history of transactions for a customer. *All* transactions in this table effect the customer balance.

Delinquencies

- Payment cycles run from 28th to the 27th of the following month
- If you do not make a payment by the 27th, you are considered delinquent
- For every day after the 27th that you are delinquent we track that in variable called “days_past_due”
- Once you make a payment the “days_past_due” returns to 0
- Customers can find themselves in any/all of the situations described below.

For example:

Simple Case: The following customer did not pay by the 27th. On the 28th he is now considered 1 day past due. They make a payment on the 30th, and are now 0 days past due.

```
| account_id | customer_id | asof_date  | days_past_due |

.
.
.
| 1          | 1           | 2020-01-22 | 0             |
| 1          | 1           | 2020-01-23 | 0             |
| 1          | 1           | 2020-01-24 | 0             |
| 1          | 1           | 2020-01-25 | 0             |
| 1          | 1           | 2020-01-26 | 0             |
| 1          | 1           | 2020-01-27 | 0             |
| 1          | 1           | 2020-01-28 | 1             |
| 1          | 1           | 2020-01-29 | 2             |
| 1          | 1           | 2020-01-30 | 0             |
| 1          | 1           | 2020-01-31 | 0             |
| 1          | 1           | 2020-02-01 | 0             |
| 1          | 1           | 2020-02-02 | 0             |
| 1          | 1           | 2020-02-03 | 0             |
| 1          | 1           | 2020-02-04 | 0             |
| 1          | 1           | 2020-02-05 | 0             |
.
.
.

Delinquency Period Start Date: 2020-01-28
Delinquency Period End Date: 2020-01-29
```

Returned Payment Case: customers make a payment but that payment then “bounces” at a later date. As we cannot go back in time, the counter “jumps” to reflect that the previous days did not count. For example, the customer below made a payment on 27th. He was then not delinquent (i.e. days_past_due = 0). However, a few days later on the 30th, his payment bounced and we then set the counter to 3 (e.g. from the 27th → 30th). Hey repays again on the 1st and his payment does not bounce.

```
| account_id | customer_id | asof_date  | days_past_due |
| 1          | 1           | 2020-01-22 | 0             |
| 1          | 1           | 2020-01-23 | 0             |
| 1          | 1           | 2020-01-24 | 0             |
| 1          | 1           | 2020-01-25 | 0             |
| 1          | 1           | 2020-01-26 | 0             |
| 1          | 1           | 2020-01-27 | 0             |
| 1          | 1           | 2020-01-28 | 0             |
| 1          | 1           | 2020-01-29 | 0             |
| 1          | 1           | 2020-01-30 | 3             |
| 1          | 1           | 2020-01-31 | 4             |
| 1          | 1           | 2020-02-01 | 5             |
| 1          | 1           | 2020-02-02 | 0             |
| 1          | 1           | 2020-02-03 | 0             |
| 1          | 1           | 2020-02-04 | 0             |
| 1          | 1           | 2020-02-05 | 0             |
| 1          | 1           | 2020-02-06 | 0             |
| 1          | 1           | 2020-02-07 | 0             |
| 1          | 1           | 2020-02-08 | 0             |
| 1          | 1           | 2020-02-09 | 0             |
| 1          | 1           | 2020-02-10 | 0             |
.
.
.

Delinquency Period Start Date: 2020-01-30
Delinquency Period End Date: 2020-02-01
```

Partial Repayment Case: customers may make a partial repayment of their credit obligation, which *only partially* reduces their counter. Partial repayments will reduce the `days_past_due` counter by 30 day increments. Although a customer has partially repaid, they are still *within* a delinquency period in this case.

```
| account_id | customer_id | asof_date  | days_past_due |
.
.
.
| 1          | 1           | 2020-01-22 | 61            |
| 1          | 1           | 2020-01-23 | 62            |
| 1          | 1           | 2020-01-24 | 63            |
| 1          | 1           | 2020-01-25 | 64            |
| 1          | 1           | 2020-01-26 | 65            |
| 1          | 1           | 2020-01-27 | 66            |
| 1          | 1           | 2020-01-28 | 36            | <- rec
| 1          | 1           | 2020-01-29 | 37            |
| 1          | 1           | 2020-01-30 | 38            |
| 1          | 1           | 2020-01-31 | 39            |
| 1          | 1           | 2020-02-01 | 40            |
| 1          | 1           | 2020-02-02 | 41            |
| 1          | 1           | 2020-02-03 | 42            |
| 1          | 1           | 2020-02-04 | 43            |
| 1          | 1           | 2020-02-05 | 44            |
| 1          | 1           | 2020-02-06 | 45            |
| 1          | 1           | 2020-02-07 | 46            |
| 1          | 1           | 2020-02-08 | 47            |
| 1          | 1           | 2020-02-09 | 48            |
| 1          | 1           | 2020-02-10 | 49            |
| 1          | 1           | 2020-02-11 | 50            |
| 1          | 1           | 2020-02-12 | 51            |
| 1          | 1           | 2020-02-13 | 52            | (lates

Delinquency Period Start Date: 2019-11-23
Delinquency Period End Date: NULL (still delinquent)
```