Analytics Engineering Challenge

For the tasks, assume that we have a database with the following schema:

```
Unset
CREATE TABLE transactions (
      transaction_id UUID,
      date DATE,
      user_id UUID,
      is_blocked BOOL,
      transaction_amount INTEGER,
      transaction_category_id INTEGER
);

CREATE TABLE users (
      user_id UUID,
      is_active BOOLEAN
);
```

Example data for these tables is stored in the corresponding CSV files transactions.csv and users.csv, which can be generated from the generate_data.py script.

N26

Task 1

We want to compute a DWH table for transactions. For every user transaction, it should compute the number of transactions the user had within the previous seven days. The resulting table should look something like the following example:

transaction_id	user_id	date	no_txn_last_7days
ef05-4247	becf-457e	2020-01-01	0
c8d1-40ca	becf-457e	2020-01-05	1
fc2b-4b36	becf-457e	2020-01-07	2
3725-48c4	becf-457e	2020-01-15	0
5f2a-47c2	becf-457e	2020-01-16	1
7541-412c	5728-4f1c	2020-01-01	0
3deb-47d7	5728-4f1c	2020-01-12	0

Your task is now to write a SQL query, which computes this table based on the transactions table from the described schema. What would the query planner of the database need to consider in order to optimize the query?

Please note that it doesn't matter if the day of the current transaction is included or excluded for the calculation of the previous seven days, both solutions are fine. It also doesn't matter which SQL implementation the solution is using.

N26

Task 2

We want to compute the result of the following query:

```
Unset
SELECT
      t.transaction_category_id,
      sum(t.amount) AS sum_amount,
      count(DISTINCT t.user_id) AS num_users
FROM transactions t
JOIN users u USING (user_id)
WHERE t.is_blocked = False
      AND u.is_active = 1
GROUP BY t.transaction_category_id
ORDER BY sum_amount DESC;
```

But unfortunately the database query planner cannot optimize the query well enough in order for us to get the results.

Your task is now to write a Python program using only the **standard libraries** (i.e. any external package that would require installation should not be used), which reads data from the CSV files transactions.csv and users.csv, and computes the equivalent result of the SQL query. The result should be printed to stdout.

Please note that the scope of the task is **not to parse the SQL query** or to generalize the computation in any way, but only to write a program which computes the result of this one specific query.

Please provide the SQL query for task 1 and the Python script to solve task 2. If you want to include some additional explanation, please feel free to add it to a README file or a PDF document.

Please don't hesitate to contact us in case something needs to be clarified.