Завдання 1: SQL

Платіжна воронка

Напиши SQL запит на основі описаної вище таблиці, що відтворює платіжну воронку наших користувачів у межах їх підписки. Воронка повинна включати лише транзакції типу `SALE`.

Окрім визначення порядку платежів, потрібно пронумерувати повторні спроби списання в межах одного поновлення. Наприклад, користувач оформив підписку (здійснив успішну оплату першого місяця), наступного місяця здійснюється повторне списання, але з першої спроби нам не вдається здійснити оплату і ми пробуємо повторно через кілька проміжків часу — це формує retry-стратегію. Таким чином будь якому успішному ребілу може передувати серія declined транзакцій які є його ретраями. Для того щоб якісно аналізувати перформанс retry-стратегію потрібно визначити порядок ретраїв.
Результат запиту має містити наступні колонки:

- `user_id` — ідентифікатор користувача
- `transaction_id` — ідентифікатор транзакції
- `project` — проєкт
- `plan_id` — тип підписки, яку купує користувач
- `transaction_time` — дата та час транзакції
- `status` — статус транзакції
- `renewal_number` — номер поновлення підписки*, порядок ребілу.
- `retry_number` — в межах кожного поновлення підписки*, порядок ребілу.
- `net_revenue` — revenue з урахуванням повернення коштів по цій транзакції та усіх комісій

**Якщо користувач мав декілька підписок, кожна з них вважається окремим ланцюжком*

```sql
WITH
renewals_count AS (
  -- Визначаємо renewal_number
  SELECT
    user_id,
    transaction_id,
    project,
    plan_id,
    transaction_time,
    status,
    -- Рахуємо кількість попередніх успішних транзакцій для визначення номеру поновлення
    COUNT(CASE WHEN status = 'SUCCESS' THEN 1 END) OVER (
        PARTITION BY user_id, project, plan_id
      ORDER BY transaction_time
      ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING
    ) + 1 AS renewal_number
  FROM subscription_transactions
  WHERE transaction_type = 'SALE'
),
retries_count as (
        -- Визначаємо retry_number
        SELECT
          user_id,
          transaction_id,
          project,
          plan_id,
          transaction_time,
          status,
```

```
        renewal_number,
        -- Рахуємо спроби в межах одного юзера, плану та номера поновлення
        ROW_NUMBER() OVER (
          PARTITION BY user_id, project, plan_id, renewal_number
          ORDER BY transaction_time
        ) AS retry_number
      FROM renewals_count
),
revenue_calculation AS (
  -- Визначаємо суму всіх пов'язаних транзакцій (Refunds, Chargebacks, Captures) для кожного SALE
  SELECT
      t.user_id,
    t.transaction_id,
    SUM(COALESCE(related.amount_usd, 0)) + t.amount_usd AS total_amount,
    SUM(COALESCE(related.fee_usd, 0)) + t.fee_usd AS total_fees,
    count(t.project ) as rows_n
  FROM subscription_transactions as t
      LEFT JOIN subscription_transactions as related ON t.transaction_id = related.reference_transaction_id
  WHERE t.transaction_type = 'SALE'
  GROUP BY t.user_id, t.transaction_id, t.amount_usd, t.fee_usd
),
dataset as (
      -- Додаємо до датасету ретраїв датасет revenue
      SELECT
        c.user_id,
        c.transaction_id,
        c.project,
        c.plan_id,
        c.transaction_time,
        c.status,
        c.renewal_number,
        c.retry_number,
        (r.total_amount - r.total_fees) AS net_revenue
      FROM retries_count c
          JOIN revenue_calculation r ON c.transaction_id = r.transaction_id
)
select * from dataset
ORDER BY user_id, project, plan_id, transaction_time;
```

Завдання 2: EDA + BI

В цьому завданні ми хочемо оцінити твої навички аналізу та візуалізації даних.

1. Дослідження даних

Спершу потрібно провести дослідження датасету (EDA). Для дослідження даних бажано використовувати Python. Як результат можна надати colab-ноутбук, який обов’язково включатиме висновки EDA.

colab-ноутбук:
https://colab.research.google.com/drive/18WKIm8pnzlAPnTyWcE6ZdsxLjqkDWmRY?usp=sharing

2. High-level payment dashboard

Побудуй high-level дашборд на основі наданого датасету, що включатиме критично важливі метриками веб-платежів для щоденного моніторингу перформансу і стабільності платіжної інфраструктури та продуктів, що нею користуються.

Ти можеш використовувати **будь-який BI-інструмент**, але окрім самого BI файлу потрібно надати **текстовий опис** до нього.

Опис має містити:

- логіку або формулу розрахунку метрик;
- пояснення, чому цю метрику важливо трекати;
- перелік фільтрів/параметрів;
- скріншоти візуалізацій.

Додатково можеш надати рекомендації:

- яких даних не вистачає для більш якісного аналізу;
- які ще метрики важливо моніторити (можливо вони не є критично важливими, але ти вважаєш що їх варто додати до моніторингу).

☺

Ми цінуємо твій час і час команди, тому просимо сфокусуватися на ключових метриках та інсайтах. Надмірна деталізація не є необхідною — для нас важливі твій підхід і логіка прийняття рішень.

У файлі ‘Payment Dashboard.pdf’ (або у ‘Payment Dashboard.pbix’) показаний Дашборд та текстовий опис до нього.