Задание 3:
0. Создать таблицы для задания
Скрипт:

```
CREATE TABLE transactions (
    id SERIAL PRIMARY KEY,
    account_id INT NOT NULL,
    created_at TIMESTAMP NOT NULL,
    op_type TEXT NOT NULL,
    amount NUMERIC(12,2) NOT NULL,
    description TEXT
);

INSERT INTO transactions (account_id, created_at, op_type, amount, description) VALUES
(1,'2025-01-01 10:00','purchase', -50,'Аптека — товар A'),
(1,'2025-01-02 09:00','purchase', -30,'Аптека — товар B'),
(1,'2025-01-03 12:00','refund', +30,'Возврат товара B'),
(1,'2025-01-04 18:00','purchase', -200,'Абонемент'),
(1,'2025-01-05 19:00','purchase', -500,'Покупка техники'),
(2,'2025-01-01 11:00','purchase', -100,'Покупка книги'),
(2,'2025-01-02 09:00','refund', +50,'Возврат книги'),
(2,'2025-01-04 16:00','purchase', -200,'Одежда');
CREATE TABLE order_history (
    id SERIAL PRIMARY KEY,
    order_id INT NOT NULL,
    created_at TIMESTAMP NOT NULL,
    field_name TEXT NOT NULL,
    old_value TEXT,
    new_value TEXT
);

INSERT INTO order_history (order_id, created_at, field_name, old_value, new_value) VALUES

(1, '2025-01-01 09:00','status_id', NULL,'1'),
(1, '2025-01-02 10:00','status_id','1','2'),
(1, '2025-01-05 08:00','status_id','2','3'),
(1, '2025-01-07 12:00','status_id','3','5'), -- выполнен


(2, '2025-01-03 11:00','status_id',NULL,'1'),
(2, '2025-01-04 15:00','status_id','1','2'),
(2, '2025-01-06 18:00','status_id','2','0'), -- отменён


(3, '2025-01-04 12:00','status_id',NULL,'1'),
(3, '2025-01-06 10:00','status_id','1','5'); -- выполнен
CREATE TABLE customer_visit (
    id SERIAL PRIMARY KEY,
    customer_id INT NOT NULL,
    created_at TIMESTAMP NOT NULL,
```

```
    visit_length INT NOT NULL,
    landing_page TEXT,
    exit_page TEXT,
    utm_source TEXT
);

INSERT INTO customer_visit (customer_id, created_at, visit_length, landing_page, exit_page,
utm_source) VALUES
(1,'2025-02-01 12:00',300,'/home','/cart','google'),
(1,'2025-02-05 15:00',120,'/catalog','/catalog','google'),
(2,'2025-02-02 10:00',200,'/home','/product','facebook'),
(3,'2025-02-03 09:00',400,'/blog','/checkout','google'),
(3,'2025-02-10 17:00',150,'/home','/home','yandex');
CREATE TABLE customer_visit_page (
    id SERIAL PRIMARY KEY,
    visit_id INT NOT NULL,
    page TEXT NOT NULL,
    time_on_page INT NOT NULL
);

INSERT INTO customer_visit_page (visit_id, page, time_on_page) VALUES
(1,'/home',60),
(1,'/catalog',120),
(1,'/cart',120),

(2,'/catalog',120),

(3,'/home',50),
(3,'/product',150),

(4,'/blog',200),
(4,'/checkout',200),

(5,'/home',150);
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    created_at TIMESTAMP NOT NULL,
    first_name TEXT,
    last_name TEXT,
    phone TEXT,
    email TEXT
);

INSERT INTO customers (created_at, first_name, last_name, phone, email) VALUES
('2024-01-01','Иван','Петров','111-11-11','ivan@example.com'),
('2024-02-01','Мария','Сидорова','222-22-22','maria@example.com'),
('2024-03-01','Олег','Смирнов','333-33-33','oleg@example.com');
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
```

```
    created_at TIMESTAMP NOT NULL,
    customer_id INT NOT NULL,
    manager_id INT,
    status_id INT,
    is_paid BOOLEAN,
    total_sum NUMERIC(10,2),
    utm_source TEXT
);

INSERT INTO orders (created_at, customer_id, manager_id, status_id, is_paid, total_sum, utm_source)
VALUES
('2025-01-01',1,10,5,TRUE,2000,'google'),
('2025-01-05',1,11,2,FALSE,1500,'google'),
('2025-02-01',2,10,5,TRUE,3000,'facebook'),
('2025-02-10',3,12,0,FALSE,500,'google'),
('2025-02-11',3,12,5,TRUE,700,'yandex');
```

вывод скрипта:

```
CREATE TABLE
INSERT 0 8
CREATE TABLE
INSERT 0 9
CREATE TABLE
INSERT 0 5
CREATE TABLE
INSERT 0 9
CREATE TABLE
INSERT 0 3
CREATE TABLE
INSERT 0 5
```

1. Добавить нумерацию строк для выборки.

Скрипт:

```
SELECT
    t.*,
    ROW_NUMBER() OVER (ORDER BY created_at) AS row_num
```

FROM transactions t;
вывод скрипта:

```
FROM transactions t;
 id | account_id |     created_at      | op_type  | amount  |   description    | row_num
----+------------+---------------------+----------+---------+------------------+---------
  1 |          1 | 2025-01-01 10:00:00 | purchase |  -50.00 | Аптека – товар A |       1
  6 |          2 | 2025-01-01 11:00:00 | purchase | -100.00 | Покупка книги    |       2
  2 |          1 | 2025-01-02 09:00:00 | purchase |  -30.00 | Аптека – товар B |       3
  7 |          2 | 2025-01-02 09:00:00 | refund   |   50.00 | Возврат книги    |       4
  3 |          1 | 2025-01-03 12:00:00 | refund   |   30.00 | Возврат товара B |       5
  8 |          2 | 2025-01-04 16:00:00 | purchase | -200.00 | Одежда           |       6
  4 |          1 | 2025-01-04 18:00:00 | purchase | -200.00 | Абонемент        |       7
  5 |          1 | 2025-01-05 19:00:00 | purchase | -500.00 | Покупка техники  |       8
(8 rows)
```

2. Пронумеровать строки в каждой группе (например, отдельно спортсменов, принимающих участие в соревнованиях по шахматам, отдельно – по настольному теннису и т.д.).
Скрипт:

```
SELECT
    t.*,
    ROW_NUMBER() OVER (PARTITION BY op_type ORDER BY created_at) AS row_in_group
FROM transactions t;
```

вывод скрипта:

```
 id | account_id |     created_at      | op_type  | amount  |   description    | row_in_group
----+------------+---------------------+----------+---------+------------------+--------------
  1 |          1 | 2025-01-01 10:00:00 | purchase |  -50.00 | Аптека – товар A |            1
  6 |          2 | 2025-01-01 11:00:00 | purchase | -100.00 | Покупка книги    |            2
  2 |          1 | 2025-01-02 09:00:00 | purchase |  -30.00 | Аптека – товар B |            3
  8 |          2 | 2025-01-04 16:00:00 | purchase | -200.00 | Одежда           |            4
  4 |          1 | 2025-01-04 18:00:00 | purchase | -200.00 | Абонемент        |            5
  5 |          1 | 2025-01-05 19:00:00 | purchase | -500.00 | Покупка техники  |            6
  7 |          2 | 2025-01-02 09:00:00 | refund   |   50.00 | Возврат книги    |            1
  3 |          1 | 2025-01-03 12:00:00 | refund   |   30.00 | Возврат товара B |            2
(8 rows)
```

3. Составить таблицу транзакций с отражением номера операции, суммы, конечного баланса (за транзакцию рассматривать, например, покупку/возврат в аптеке, покупку/возврат абонемента, покупку/возврат билета).
Скрипт:

```
SELECT
    t.id,
    t.account_id,
    t.created_at,
    t.op_type,
    t.amount,
    ROW_NUMBER() OVER (PARTITION BY account_id ORDER BY created_at) AS op_num,
    SUM(t.amount) OVER (PARTITION BY account_id ORDER BY created_at) AS balance_after
FROM transactions t
ORDER BY account_id, created_at;
```

вывод скрипта:

```
 id | account_id |     created_at      | op_type  | amount  | op_num | balance_after
----+------------+---------------------+----------+---------+--------+---------------
  1 |          1 | 2025-01-01 10:00:00 | purchase |  -50.00 |      1 |        -50.00
  2 |          1 | 2025-01-02 09:00:00 | purchase |  -30.00 |      2 |        -80.00
  3 |          1 | 2025-01-03 12:00:00 | refund   |   30.00 |      3 |        -50.00
  4 |          1 | 2025-01-04 18:00:00 | purchase | -200.00 |      4 |       -250.00
  5 |          1 | 2025-01-05 19:00:00 | purchase | -500.00 |      5 |       -750.00
  6 |          2 | 2025-01-01 11:00:00 | purchase | -100.00 |      1 |       -100.00
  7 |          2 | 2025-01-02 09:00:00 | refund   |   50.00 |      2 |        -50.00
  8 |          2 | 2025-01-04 16:00:00 | purchase | -200.00 |      3 |       -250.00
(8 rows)
```

4. Дополнить таблицу с транзакциями дополнительными столбцами (например, процент от общей суммы и т.д.).
Скрипт:

```
SELECT
    t.*,
    SUM(amount) OVER () AS total_amount,
    ROUND(amount / SUM(amount) OVER () * 100, 2) AS percent_of_total
FROM transactions t;
```

вывод скрипта:

```
 id | account_id |     created_at      | op_type  | amount  |   description    | total_amount | percent_of_total
----+------------+---------------------+----------+---------+------------------+--------------+------------------
  1 |          1 | 2025-01-01 10:00:00 | purchase |  -50.00 | Аптека – товар A |     -1000.00 |             5.00
  2 |          1 | 2025-01-02 09:00:00 | purchase |  -30.00 | Аптека – товар B |     -1000.00 |             3.00
  3 |          1 | 2025-01-03 12:00:00 | refund   |   30.00 | Возврат товара B |     -1000.00 |            -3.00
  4 |          1 | 2025-01-04 18:00:00 | purchase | -200.00 | Абонемент        |     -1000.00 |            20.00
  5 |          1 | 2025-01-05 19:00:00 | purchase | -500.00 | Покупка техники  |     -1000.00 |            50.00
  6 |          2 | 2025-01-01 11:00:00 | purchase | -100.00 | Покупка книги    |     -1000.00 |            10.00
  7 |          2 | 2025-01-02 09:00:00 | refund   |   50.00 | Возврат книги    |     -1000.00 |            -5.00
  8 |          2 | 2025-01-04 16:00:00 | purchase | -200.00 | Одежда           |     -1000.00 |            20.00
(8 rows)
```

5. Модифицировать запрос из п.4 с использованием WINDOW для одинаковых выражений.
Скрипт:

```
SELECT
    x.*,
    x.total_sum,
    ROUND(x.amount / x.total_sum * 100, 2) AS percent_of_total
FROM (
    SELECT
        t.*,
        SUM(amount) OVER w AS total_sum
    FROM transactions t
    WINDOW w AS ()
) x;
```

вывод скрипта:

```
 id | account_id |     created_at      | op_type  | amount  |   description    | total_sum | total_sum | percent_of_total
----+------------+---------------------+----------+---------+------------------+-----------+-----------+------------------
  1 |          1 | 2025-01-01 10:00:00 | purchase |  -50.00 | Аптека – товар A |  -1000.00 |  -1000.00 |             5.00
  2 |          1 | 2025-01-02 09:00:00 | purchase |  -30.00 | Аптека – товар B |  -1000.00 |  -1000.00 |             3.00
  3 |          1 | 2025-01-03 12:00:00 | refund   |   30.00 | Возврат товара B |  -1000.00 |  -1000.00 |            -3.00
  4 |          1 | 2025-01-04 18:00:00 | purchase | -200.00 | Абонемент        |  -1000.00 |  -1000.00 |            20.00
  5 |          1 | 2025-01-05 19:00:00 | purchase | -500.00 | Покупка техники  |  -1000.00 |  -1000.00 |            50.00
  6 |          2 | 2025-01-01 11:00:00 | purchase | -100.00 | Покупка книги    |  -1000.00 |  -1000.00 |            10.00
  7 |          2 | 2025-01-02 09:00:00 | refund   |   50.00 | Возврат книги    |  -1000.00 |  -1000.00 |            -5.00
  8 |          2 | 2025-01-04 16:00:00 | purchase | -200.00 | Одежда           |  -1000.00 |  -1000.00 |            20.00
```

6. Отфильтровать результаты запроса из пункта 5 (используя подзапрос).
Скрипт:

```
SELECT *
FROM (
    SELECT
        t.*,
        SUM(amount) OVER (PARTITION BY account_id ORDER BY created_at) AS balance_after
    FROM transactions t
) x
```

WHERE amount < 0;
вывод скрипта:

```
 id | account_id |     created_at      | op_type  | amount  |   description    | balance_after
----+------------+---------------------+----------+---------+------------------+---------------
  1 |          1 | 2025-01-01 10:00:00 | purchase |  -50.00 | Аптека — товар A |        -50.00
  2 |          1 | 2025-01-02 09:00:00 | purchase |  -30.00 | Аптека — товар B |        -80.00
  4 |          1 | 2025-01-04 18:00:00 | purchase | -200.00 | Абонемент        |       -250.00
  5 |          1 | 2025-01-05 19:00:00 | purchase | -500.00 | Покупка техники  |       -750.00
  6 |          2 | 2025-01-01 11:00:00 | purchase | -100.00 | Покупка книги    |       -100.00
  8 |          2 | 2025-01-04 16:00:00 | purchase | -200.00 | Одежда           |       -250.00
(6 rows)
```

Задача 3.1. История изменения заказа
Составить отчет:
Статус заказа | Среднее время пребывания заказа в этом статусе
Скрипт:

```
WITH status_changes AS (
   SELECT
      order_id,
      new_value::INT AS status_id,
      created_at AS status_start,
      LEAD(created_at) OVER (PARTITION BY order_id ORDER BY created_at) AS next_change
   FROM order_history
   WHERE field_name = 'status_id'
)
SELECT
   status_id,
   AVG(EXTRACT(EPOCH FROM (COALESCE(next_change, NOW()) - status_start))) AS
avg_seconds
FROM status_changes
GROUP BY status_id
ORDER BY status_id;
```

вывод скрипта:

```
 status_id |          avg_seconds
-----------+-------------------------
         0 | 28949077.389124000000
         1 |   118800.000000000000
         2 |   217800.000000000000
         3 |   187200.000000000000
         5 | 28931077.389124000000
(5 rows)
```

Задача 3.2. Визиты клиентов
Составить отчеты:
ID клиента | Дата последнего визита
ID клиента | Среднее количество просмотров страниц за визит
ID клиента | Адреса страниц с визитами дольше среднего времени визита этого клиента
Скрипт1:

```
SELECT
   customer_id,
   MAX(created_at) AS last_visit
FROM customer_visit
GROUP BY customer_id;
```

вывод скрипта:

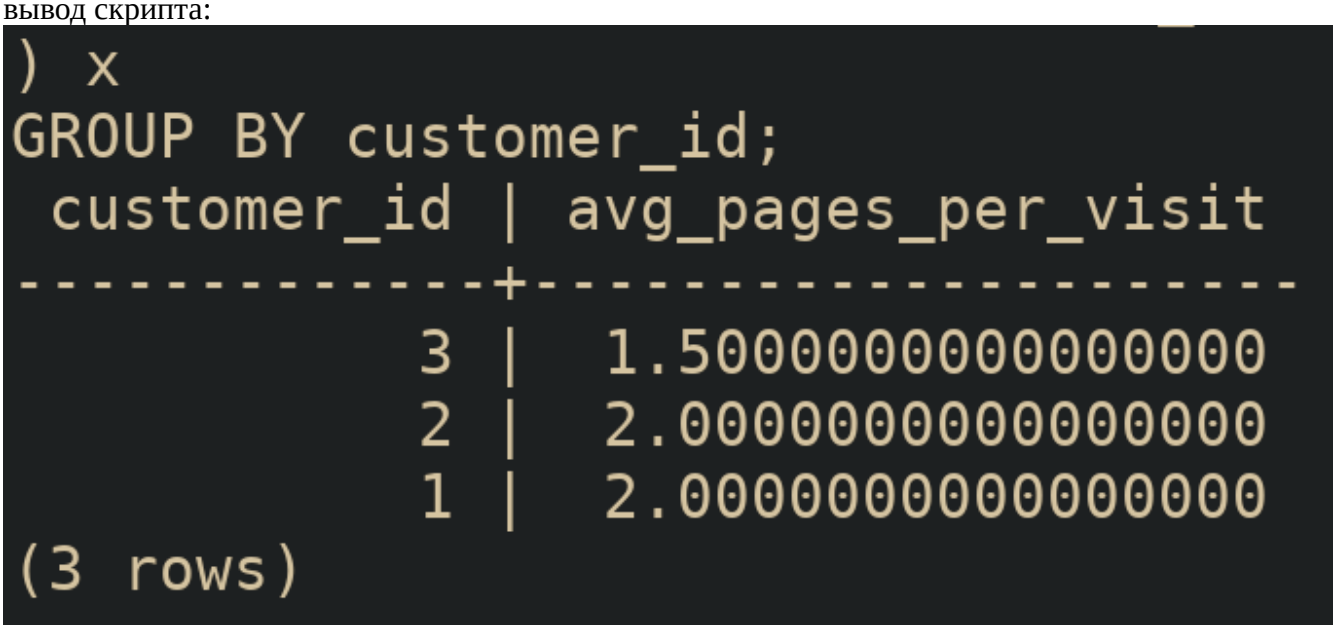
```
FROM customer_visit
GROUP BY customer_id;
 customer_id |     last_visit
-------------+---------------------
           3 | 2025-02-10 17:00:00
           2 | 2025-02-02 10:00:00
           1 | 2025-02-05 15:00:00
(3 rows)
```

Скрипт2:

```
SELECT
```

```
    x.customer_id,
    AVG(page_count) AS avg_pages_per_visit
FROM (
    SELECT
        cv.id AS visit_id,
        cv.customer_id,
        COUNT(p.id) AS page_count
    FROM customer_visit cv
    LEFT JOIN customer_visit_page p ON p.visit_id = cv.id
    GROUP BY cv.id, cv.customer_id
) x
GROUP BY customer_id;
```

вывод скрипта:



Скрипт3:

```
WITH avg_visit_time AS (
    SELECT
        customer_id,
        AVG(visit_length) AS avg_len
    FROM customer_visit
    GROUP BY customer_id
)
SELECT
    p.visit_id,
    cv.customer_id,
    p.page,
    p.time_on_page
FROM customer_visit_page p
JOIN customer_visit cv ON cv.id = p.visit_id
JOIN avg_visit_time a ON a.customer_id = cv.customer_id
WHERE p.time_on_page > a.avg_len;
```

вывод скрипта:

```
 visit_id | customer_id | page | time_on_page
----------+-------------+------+--------------
(0 rows)
```

Задача 3.3 Расчет конверсии
Составить отчеты
ID клиента | Среднее время между заказами
ID клиента | Количество визитов | Количество заказов
Источник трафика | Количество визитов с этим источником | Количество созданных заказов | Количество оплаченных заказов | Количество выполненных заказов
ID менеджера | Среднее время выполнения заказов | Доля отмененных заказов | Сумма выполненных заказов | Средняя стоимость заказа
ID менеджера | Рейтинг менеджера
Рейтинг считается как (Доля выполненных менеджером заказов - доля выполненных заказов в среднем) + (Среднее время выполнения заказов менеджером - Среднее время выполнения заказов итого) - (Процент отмененных менеджером заказов - Процент отмененных заказов всего)
Скрипт1:

```
SELECT
   utm_source,
   COUNT(*) FILTER (WHERE status_id = 5) AS completed_orders,
   COUNT(*) AS all_orders,
   COUNT(*) FILTER (WHERE status_id = 5)::DECIMAL / COUNT(*) AS conversion
FROM orders
GROUP BY utm_source;
```

вывод скрипта:

```
 utm_source | completed_orders | all_orders |       conversion
------------+------------------+------------+------------------------
 facebook   |                1 |          1 | 1.00000000000000000000
 google     |                1 |          3 | 0.33333333333333333333
 yandex     |                1 |          1 | 1.00000000000000000000
(3 rows)
```

Скрипт2:

```
SELECT
   customer_id,
   COUNT(*) FILTER (WHERE status_id = 5) AS completed_orders,
   COUNT(*) AS total_orders,
   COUNT(*) FILTER (WHERE status_id = 5)::DECIMAL / COUNT(*) AS conversion
FROM orders
GROUP BY customer_id;
```

вывод скрипта:

```
 customer_id | completed_orders | total_orders |       conversion
-------------+------------------+--------------+------------------------
           3 |                1 |            2 | 0.50000000000000000000
           2 |                1 |            1 | 1.00000000000000000000
           1 |                1 |            2 | 0.50000000000000000000
(3 rows)
```

Скрипт3:

```
SELECT
   manager_id,
   COUNT(*) FILTER (WHERE status_id = 5) AS success,
   COUNT(*) AS total,
   COUNT(*) FILTER (WHERE status_id = 5)::DECIMAL / COUNT(*) AS conversion
FROM orders
GROUP BY manager_id;
```

вывод скрипта:

```
 manager_id | success | total |       conversion
------------+---------+-------+------------------------
         11 |       0 |     1 | 0.00000000000000000000
         10 |       2 |     2 | 1.00000000000000000000
         12 |       1 |     2 | 0.50000000000000000000
(3 rows)
```

Скрипт4:

```
WITH finished_orders AS (
   SELECT *
   FROM orders
   WHERE status_id IN (0, 5)
),
manager_stats AS (
   SELECT
      manager_id,
      COUNT(*) AS total_orders,
      COUNT(CASE WHEN status_id = 5 THEN 1 END) AS completed_orders,
      COUNT(CASE WHEN status_id = 0 THEN 1 END) AS canceled_orders,
      AVG(EXTRACT(EPOCH FROM (NOW() - created_at))/86400) FILTER (WHERE status_id = 5)
AS avg_completion_time
   FROM finished_orders
   GROUP BY manager_id
),
overall_stats AS (
   SELECT
      COUNT(*) AS total_orders,
      COUNT(CASE WHEN status_id = 5 THEN 1 END) AS completed_orders,
      COUNT(CASE WHEN status_id = 0 THEN 1 END) AS canceled_orders,
```

```
      AVG(EXTRACT(EPOCH FROM (NOW() - created_at))/86400) FILTER (WHERE status_id = 5)
AS avg_completion_time
   FROM finished_orders
)
SELECT
   m.manager_id,
   ROUND(
      (m.completed_orders::decimal / m.total_orders - o.completed_orders::decimal / o.total_orders)
      + (m.avg_completion_time - o.avg_completion_time)
      - (m.canceled_orders::decimal / m.total_orders - o.canceled_orders::decimal / o.total_orders)
   , 2) AS manager_rating
FROM manager_stats m
CROSS JOIN overall_stats o
ORDER BY manager_rating DESC;
```

вывод скрипта: