Лабораторна робота № 3

Квартюк Олексій

МІТ-31

Перша частина:

Функції:

-- Функція для розрахунку загальної суми замовлення по айді

CREATE OR REPLACE FUNCTION calculate\_order(customer\_id INT)

RETURNS DECIMAL

LANGUAGE plpgsql AS

$$

DECLARE

    total DECIMAL(10, 2);

BEGIN

    SELECT SUM(total\_amount) INTO total FROM "Order" WHERE "Order".customer\_id = calculate\_order.customer\_id;

    RETURN total;

END;

$$;

SELECT calculate\_order (1);

SELECT \* FROM "Order"

-- Функція для отримання інформації про ім'я та пошту по айді

CREATE OR REPLACE FUNCTION get\_info(customer\_id INT)

RETURNS TABLE(customer\_name VARCHAR, customer\_email VARCHAR)

LANGUAGE plpgsql AS

$$

BEGIN

    RETURN QUERY

    SELECT name, email

    FROM "Customer"

    WHERE

        "Customer".customer\_id = get\_info.customer\_id;

END;

$$;

SELECT \* FROM get\_info (1);

-- Функція для отримання всій інформації по айді

CREATE OR REPLACE FUNCTION get\_details(order\_id\_param INT)

RETURNS TABLE(

    order\_id INT,

    customer\_id INT,

    date DATE,

    total\_amount DECIMAL(10, 2),

    product\_name VARCHAR(255),

    product\_price DECIMAL(10, 2),

    quantity INT)

LANGUAGE plpgsql AS

$$

BEGIN

    RETURN QUERY

    SELECT

        o.order\_id,

        o.customer\_id,

        o.date,

        o.total\_amount,

        p.name AS product\_name,

        p.price AS product\_price,

        op.quantity

    FROM

        "Order" o

        JOIN "Order\_Product" op ON o.order\_id = op.order\_id

        JOIN "Product" p ON op.product\_id = p.product\_id

    WHERE

        o.order\_id = order\_id\_param;

END;

$$;

SELECT \* FROM get\_details(2);

Індексація:

CREATE INDEX idx\_order\_customer\_id ON "Order" (customer\_id);

CREATE INDEX idx\_order\_product\_product\_id ON "Order\_Product" (product\_id);

EXPLAIN ANALYZE SELECT \* FROM "Order" WHERE customer\_id = 123;

Процедури:

-- Процедура для вставки нового замовлення в "Order"

CREATE OR REPLACE PROCEDURE insert\_order(

    customer\_id\_param INT,

    date\_param DATE,

    total\_amount\_param DECIMAL(10, 2)

    )

LANGUAGE plpgsql AS

$$

BEGIN

    INSERT INTO

        "Order" (

            customer\_id, date, total\_amount

        )

    VALUES (

            customer\_id\_param, date\_param, total\_amount\_param

        );

END;

$$;

-- Виклик першої процедури

CALL insert\_order (1, '2024-03-24', 299);

-- Процедура для вставки нового продукту в "Product"

CREATE OR REPLACE PROCEDURE insert\_product(

    name\_param VARCHAR(255),

    price\_param DECIMAL(10, 2)

    )

LANGUAGE plpgsql AS

$$

BEGIN

    INSERT INTO

        "Product" (name, price)

    VALUES (name\_param, price\_param);

END;

$$;

-- Виклик другої процедури

CALL insert\_product ('Cheeseburger\_2', 4.99);

-- Процедура для видалення замовлення по id

CREATE OR REPLACE PROCEDURE delete\_order(

    order\_id\_param INT)

LANGUAGE plpgsql AS

$$

BEGIN

    DELETE FROM "Order"

    WHERE

        order\_id = order\_id\_param;

END;

$$;

-- Виклик третьої процедури

CALL delete\_order (10);

SELECT \* FROM "Order";

SELECT \* FROM "Product";

Тригери:

-- Тригер для оновлення дати в таблиці "Order" при зміні значень в "Order\_Product"

CREATE OR REPLACE FUNCTION update\_order\_date()

RETURNS TRIGGER AS

$$

BEGIN

    UPDATE "Order"

    SET

        date = CURRENT\_DATE

    WHERE

        order\_id = NEW.order\_id;

    RETURN NEW;

END;

$$;

CREATE OR REPLACE TRIGGER update\_order\_date\_trigger

AFTER UPDATE ON "Order\_Product"

FOR EACH ROW EXECUTE FUNCTION update\_order\_date ();

SELECT \* FROM "Order";

SELECT \* FROM "Order\_Product";

UPDATE "Order\_Product"

SET quantity = 5000

WHERE order\_id = 5 AND product\_id = 5;

-- Тригер для виклику помилки при спробі ввести від'ємне значення ціни в таблиці "Product"

CREATE OR REPLACE FUNCTION check\_product\_price()

RETURNS TRIGGER AS

$$

BEGIN

    IF NEW.price < 0 THEN RAISE EXCEPTION 'Ціна продукту повинна бути не менше 0';

END

    IF;

    RETURN NEW;

END;

$$

LANGUAGE

plpgsql;

CREATE OR REPLACE TRIGGER tr\_check\_product\_price

BEFORE INSERT ON "Product" FOR EACH ROW

EXECUTE FUNCTION check\_product\_price ();

INSERT INTO "Product" (name, price) VALUES ('Test Product', -10.99);

-- Аналогічний тригер, але для таблиці "Order"

CREATE OR REPLACE FUNCTION check\_order\_total()

RETURNS TRIGGER AS

$$

BEGIN

    IF NEW.total\_amount < 0 THEN RAISE EXCEPTION 'Загальна сума замовлення повинна бути не менше 0';

END

    IF;

    RETURN NEW;

END;

$$

LANGUAGE

plpgsql;

CREATE OR REPLACE TRIGGER tr\_check\_order\_total

BEFORE INSERT ON "Order"

FOR EACH ROW EXECUTE FUNCTION check\_order\_total ();

INSERT INTO

    "Order" (

        customer\_id, date, total\_amount

    )

VALUES (1, CURRENT\_DATE, -10.99);

Транзакції:

-- Транзакція:

BEGIN;

WITH

    inserted\_customer AS (

        INSERT INTO

            "Customer" (name, email, phone)

        VALUES (

                'John Doe', 'john@example.com', '123-456-7890'

            ) RETURNING customer\_id

    )

INSERT INTO

    "Order" (

        customer\_id, date, total\_amount

    )

SELECT customer\_id, CURRENT\_DATE, 50.00

FROM inserted\_customer;

COMMIT;

SELECT \* FROM "Customer"

SELECT \* FROM "Order"

BEGIN;

UPDATE "Product" SET price = 15.99 WHERE product\_id = 1;

WITH

    inserted\_order AS (

        INSERT INTO

            "Order" (

                customer\_id, date, total\_amount

            )

        VALUES (2, CURRENT\_DATE, 0.00) RETURNING order\_id

    )

INSERT INTO

    "Order\_Product" (

        order\_id, product\_id, quantity

    )

VALUES (

        (

            SELECT order\_id

            FROM inserted\_order

        ), 1, 2

    );

COMMIT;

BEGIN;

DELETE FROM "Order\_Product"

WHERE

    order\_id IN (

        SELECT order\_id

        FROM "Order"

        WHERE

            customer\_id = 2

    );

DELETE FROM "Order" WHERE customer\_id = 2;

DELETE FROM "Customer" WHERE customer\_id = 2;

COMMIT;

Друга частина:

Index.html(Робив через фласк але при здачі показував постман):

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Orders</title>

    <style>

        table {

            width: 100%;

            border-collapse: collapse;

        }

        th,

        td {

            border: 1px solid #ddd;

            padding: 8px;

            text-align: left;

        }

        tr:nth-child(even) {

            background-color: #f2f2f2;

        }

        .container {

            margin-top: 20px;

        }

        .btn {

            padding: 10px 20px;

            background-color: #4CAF50;

            color: white;

            border: none;

            cursor: pointer;

            border-radius: 5px;

        }

        .btn:hover {

            background-color: #45a049;

        }

    </style>

</head>

<body>

    <h2>Orders</h2>

    <div class="container">

        <button onclick="showForm()">Add New Order</button>

        <div id="orderForm" style="display: none;">

            <h3>Add New Order</h3>

            <form id="addOrderForm">

                <label for="customer\_id">Customer ID:</label>

                <input type="text" id="customer\_id" name="customer\_id"><br><br>

                <label for="date">Date:</label>

                <input type="date" id="date" name="date"><br><br>

                <label for="total\_amount">Total Amount:</label>

                <input type="text" id="total\_amount" name="total\_amount"><br><br>

                <button type="submit" class="btn">Submit</button>

            </form>

        </div>

        <br><br>

        <table id="ordersTable">

            <tr>

                <th>Order ID</th>

                <th>Customer ID</th>

                <th>Date</th>

                <th>Total Amount</th>

                <th>Actions</th>

            </tr>

            <!-- Orders will be dynamically inserted here -->

        </table>

    </div>

    <script>

        // JavaScript для обработки формы и отображения списка заказов

        // Функция для отображения формы добавления заказа

        function showForm() {

            var form = document.getElementById('orderForm');

            if (form.style.display === 'none') {

                form.style.display = 'block';

            } else {

                form.style.display = 'none';

            }

        }

        // Функция для отправки данных формы добавления заказа

        document.getElementById('addOrderForm').addEventListener('submit', function (event) {

            event.preventDefault(); // Предотвращаем стандартное поведение формы

            var formData = new FormData(this);

            fetch('/orders', {

                method: 'POST',

                body: JSON.stringify({

                    customer\_id: formData.get('customer\_id'),

                    date: formData.get('date'),

                    total\_amount: formData.get('total\_amount')

                }),

                headers: {

                    'Content-Type': 'application/json'

                }

            })

                .then(response => response.json())

                .then(data => {

                    console.log(data);

                    // Перезагрузка страницы для отображения нового заказа

                    window.location.reload();

                })

                .catch(error => console.error('Error:', error));

        });

        // Функция для загрузки списка заказов при загрузке страницы

        window.onload = function () {

            fetch('/orders')

                .then(response => response.json())

                .then(data => {

                    console.log(data);

                    var ordersTable = document.getElementById('ordersTable');

                    data.forEach(order => {

                        var row = `<tr>

                               <td>${order.order\_id}</td>

                               <td>${order.customer\_id}</td>

                               <td>${order.date}</td>

                               <td>${order.total\_amount}</td>

                               <td><button onclick="deleteOrder(${order.order\_id})">Delete</button></td>

                           </tr>`;

                        ordersTable.innerHTML += row;

                    });

                })

                .catch(error => console.error('Error:', error));

        };

        // Функция для удаления заказа

        function deleteOrder(orderId) {

            fetch(`/orders/${orderId}`, {

                method: 'DELETE'

            })

                .then(response => response.json())

                .then(data => {

                    console.log(data);

                    // Перезагрузка страницы для обновления списка заказов

                    window.location.reload();

                })

                .catch(error => console.error('Error:', error));

        }

    </script>

</body>

</html>

CRUD:

from flask import Flask, request, jsonify, render\_template

import psycopg2

app = Flask(\_\_name\_\_)

conn = psycopg2.connect(

    dbname="mcdonalds",

    user="manager",

    password="manager",

    host="localhost",

    port="5432"

)

cur = conn.cursor()

@app.route('/')

def index():

    cur.execute("SELECT \* FROM \"Order\"")

    orders = cur.fetchall()

    return render\_template('index.html', orders=orders)

@app.route('/orders', methods=['POST'])

def create\_order():

    data = request.get\_json()

    customer\_id = data['customer\_id']

    date = data['date']

    total\_amount = data['total\_amount']

    cur.execute("INSERT INTO \"Order\" (customer\_id, date, total\_amount) VALUES (%s, %s, %s) RETURNING order\_id", (customer\_id, date, total\_amount))

    order\_id = cur.fetchone()[0]

    conn.commit()

    return jsonify({'order\_id': order\_id}), 201

@app.route('/orders', methods=['GET'])

def get\_orders():

    cur.execute("SELECT \* FROM \"Order\"")

    orders = cur.fetchall()

    orders\_list = [{'order\_id': order[0], 'customer\_id': order[1], 'date': order[2].isoformat(), 'total\_amount': float(order[3])} for order in orders]

    return jsonify(orders\_list), 200

@app.route('/orders/<int:order\_id>', methods=['GET'])

def get\_order(order\_id):

    cur.execute("SELECT \* FROM \"Order\" WHERE order\_id = %s", (order\_id,))

    order = cur.fetchone()

    if order:

        order\_dict = {'order\_id': order[0], 'customer\_id': order[1], 'date': order[2].isoformat(), 'total\_amount': float(order[3])}

        return jsonify(order\_dict), 200

    else:

        return jsonify({'message': 'Order not found'}), 404

@app.route('/orders/<int:order\_id>', methods=['PUT'])

def update\_order(order\_id):

    data = request.get\_json()

    customer\_id = data['customer\_id']

    date = data['date']

    total\_amount = data['total\_amount']

    cur.execute("UPDATE \"Order\" SET customer\_id = %s, date = %s, total\_amount = %s WHERE order\_id = %s", (customer\_id, date, total\_amount, order\_id))

    conn.commit()

    return jsonify({'message': 'Order updated successfully'}), 200

@app.route('/orders/<int:order\_id>', methods=['DELETE'])

def delete\_order(order\_id):

    cur.execute("DELETE FROM \"Order\" WHERE order\_id = %s", (order\_id,))

    conn.commit()

    return jsonify({'message': 'Order deleted successfully'}), 200

@app.route('/calculate\_order', methods=['GET'])

def calculate\_order\_route():

    customer\_id = request.args.get('customer\_id')

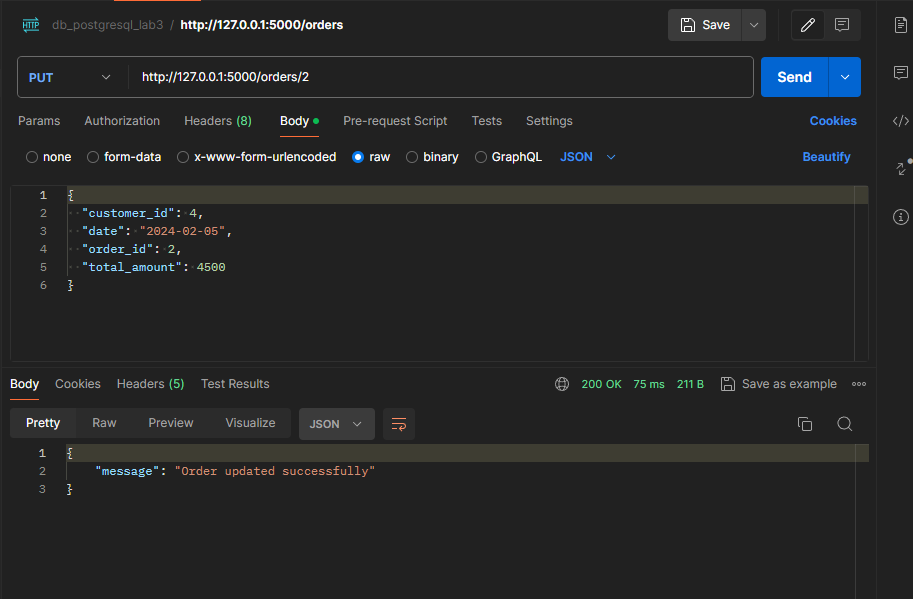
    cur.execute("SELECT calculate\_order(%s)", (customer\_id,))

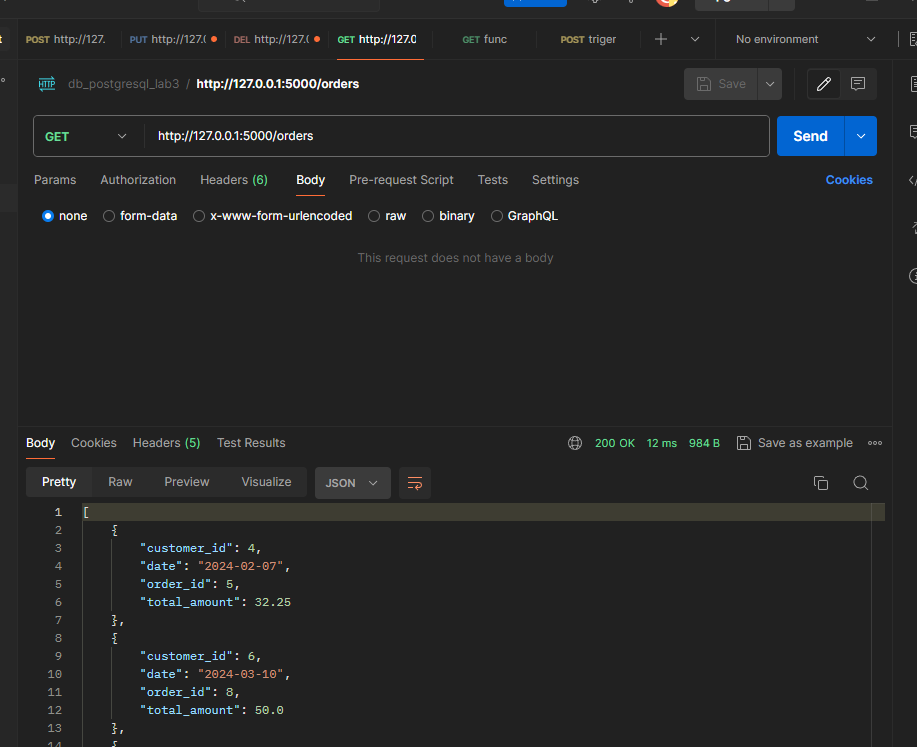
    result = cur.fetchone()[0]

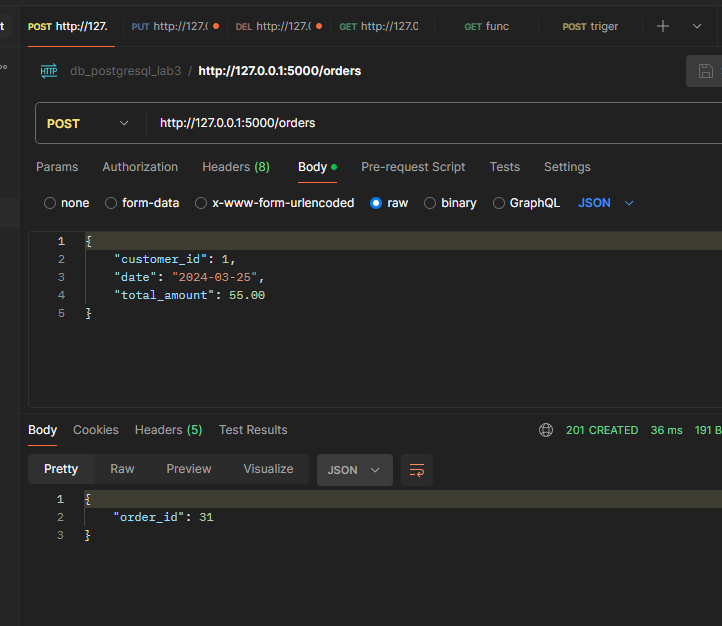
    return jsonify({'result': result}), 200

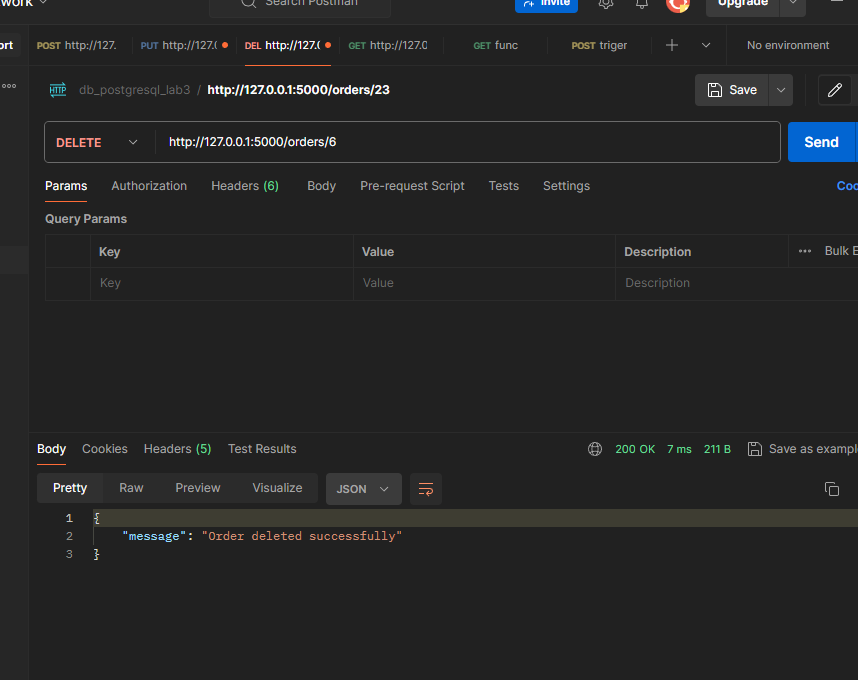
if \_\_name\_\_ == '\_\_main\_\_':

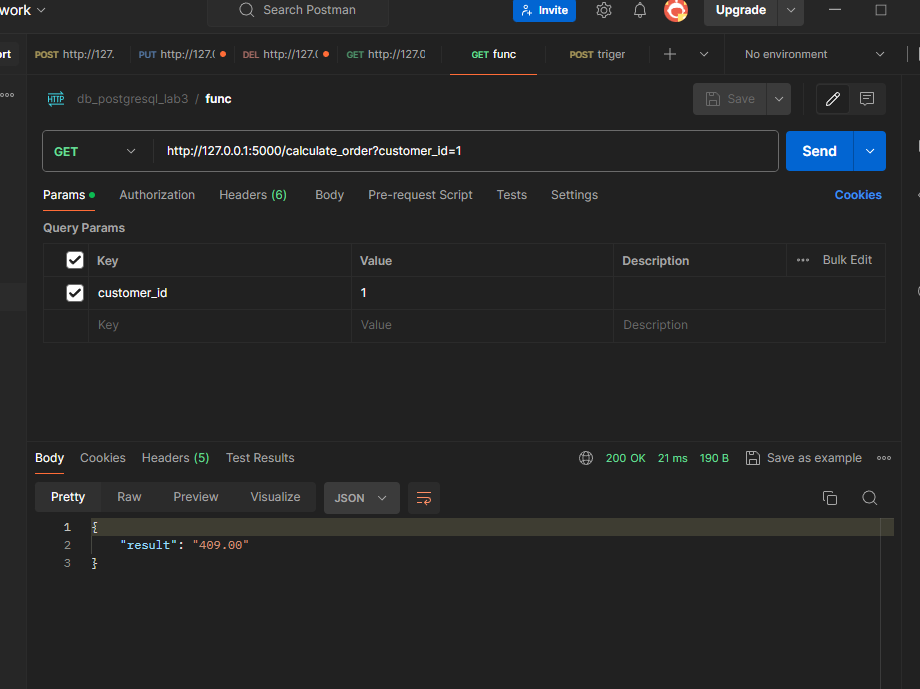
    app.run(debug=True)

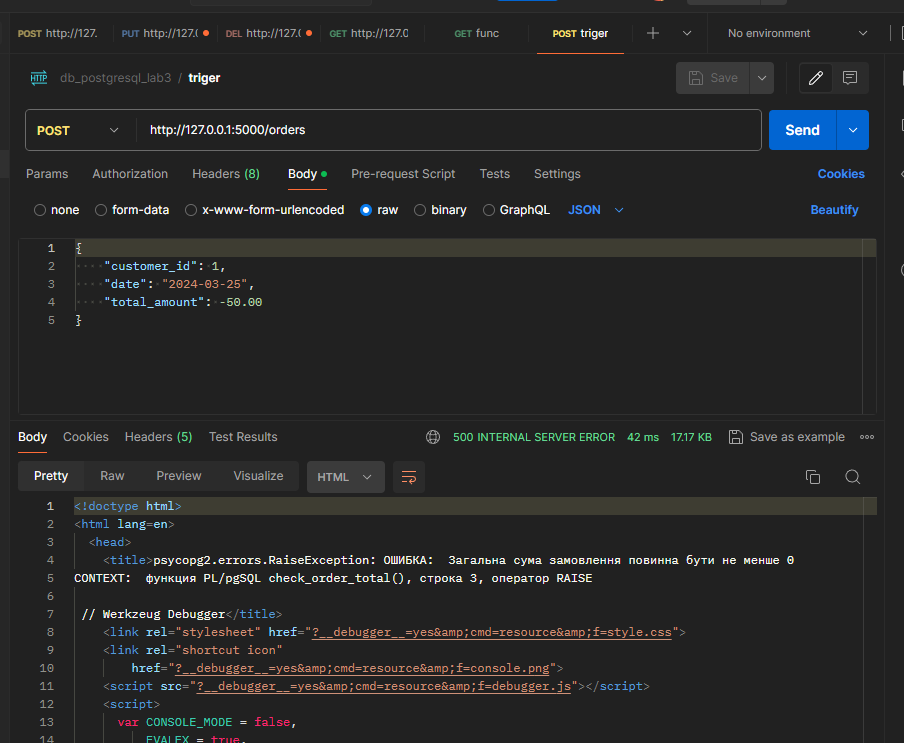












**Висновок:** Отже, ми зробили по 5 тригерів, функцій, транзакцій та індексацію в першій частина та протестували роботу crud в постман у другій частині.