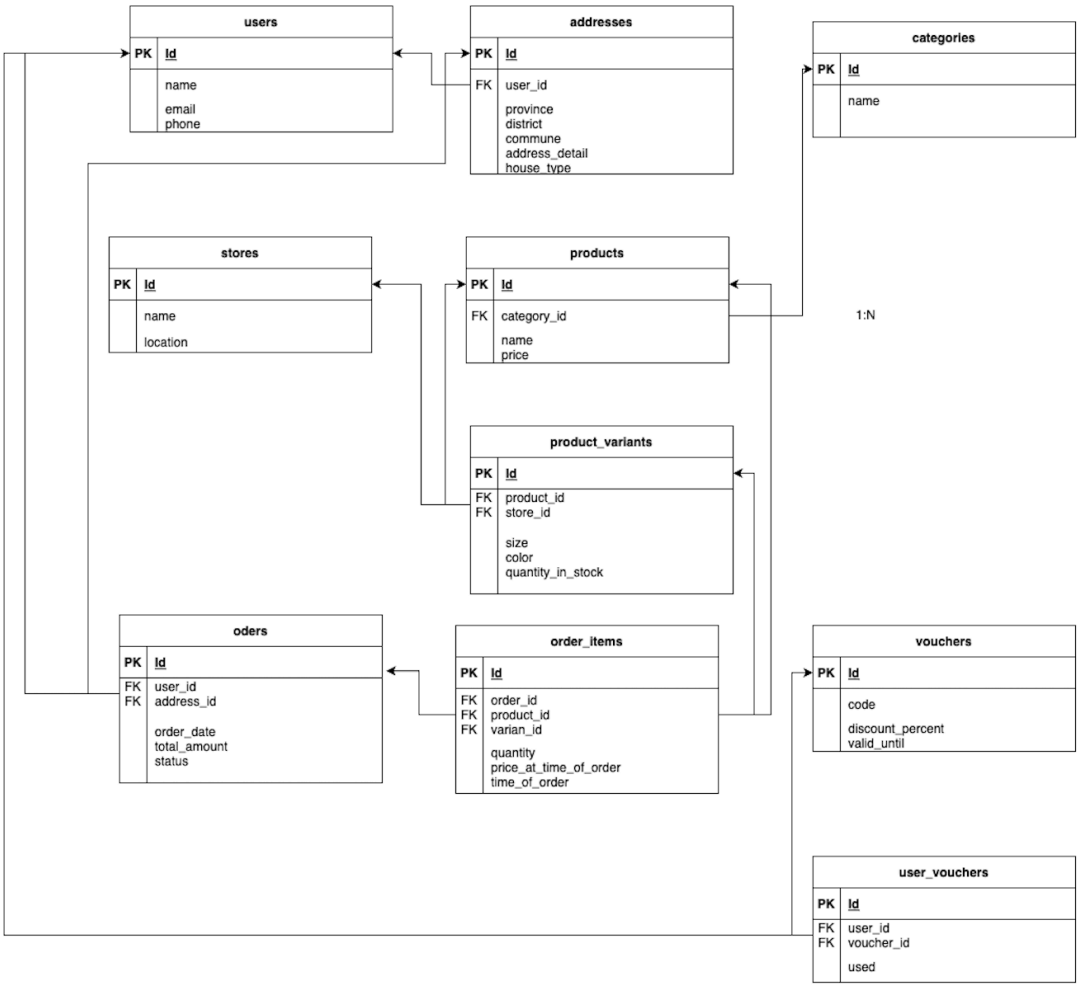
PRODUCT BACKEND TECHNICAL ASSESSMENT

Questions:

a) Design a relational database to store all the information contained in the above images such as products, addresses, stores, categories, orders, users, ... And describe the types of database normalization you used.



Chi tiết các bảng:

1. **Bảng users**:
    - **id**: Mã người dùng (khoá chính).
    - **name**: Tên người dùng.
    - **email**: Địa chỉ email của người dùng, phải là duy nhất.
    - **phone**: Số điện thoại của người dùng.
2. **Bảng addresses**:
    - **id**: Mã địa chỉ (khoá chính).
    - **user_id**: Mã người dùng (liên kết với bảng users).

    - **province**: Tỉnh thành.
    - **district**: Quận/huyện.
    - **commune**: Xã/phường.
    - **address_detail**: Chi tiết địa chỉ (đường phố, số nhà,...).
3. **Bảng categories**:
    - **id**: Mã danh mục (Khoá chính).
    - **name**: Tên danh mục (ví dụ: điện thoại, laptop, thời trang,...).
4. **Bảng stores**:
    - **id**: Mã cửa hàng (Khoá chính).
    - **name**: Tên cửa hàng.
    - **location**: Địa chỉ hoặc vị trí của cửa hàng.
5. **Bảng products**:
    - **id**: Mã sản phẩm (Khoá chính).
    - **name**: Tên sản phẩm.
    - **category_id**: Mã danh mục sản phẩm (liên kết với bảng categories).
    - **price**: Giá của sản phẩm.
6. **Bảng product_variants**:
    - **id**: Mã biến thể sản phẩm (Khoá chính).
    - **product_id**: Mã sản phẩm (liên kết với bảng products).
    - **size**: Kích cỡ của sản phẩm (nếu có).
    - **color**: Màu sắc của sản phẩm (nếu có).
    - **quantity_in_stock**: Số lượng sản phẩm còn trong kho.
    - **store_id**: Mã cửa hàng nơi bán sản phẩm này (liên kết với bảng stores).
7. **Bảng orders**:
    - **id**: Mã đơn hàng (Khoá chính).
    - **user_id**: Mã người dùng (liên kết với bảng users).
    - **address_id**: Mã địa chỉ (liên kết với bảng addresses).
    - **order_date**: Ngày giờ đặt hàng (mặc định là thời gian hiện tại).
    - **total_amount**: Tổng giá trị đơn hàng.
    - **status**: Trạng thái đơn hàng (mặc định là 'pending', có thể là 'completed', 'cancelled',...).
8. **Bảng order_items**:
    - **id**: Mã mục đơn hàng (Khoá chính).
    - **order_id**: Mã đơn hàng (liên kết với bảng orders).
    - **product_variant_id**: Mã biến thể sản phẩm (liên kết với bảng product_variants).
    - **quantity**: Số lượng sản phẩm trong đơn hàng.
    - **price_at_time_of_order**: Giá sản phẩm tại thời điểm đặt hàng.

9. **Bảng vouchers**:
    - **id**: Mã voucher (Khoá chính).
    - **code**: Mã voucher (dùng để nhập khi thanh toán).
    - **discount_percent**: Phần trăm giảm giá của voucher.
    - **valid_until**: Ngày hết hạn của voucher.
10. **Bảng user_vouchers**:

- **id**: Mã bản ghi (Khoá chính).
- **user_id**: Mã người dùng (liên kết với bảng users).
- **voucher_id**: Mã voucher (liên kết với bảng vouchers).
- **used**: Cờ đánh dấu xem người dùng đã sử dụng voucher hay chưa (mặc định là false).

b) User "assessment", with information as shown, purchased the product "KAPPA Women's Sneakers" in yellow, size 36, quantity 1. Please write a query to insert the order that this person purchased database.

Đây là các câu lệnh chèn:

```
-- Chèn thông tin người dùng
INSERT INTO users (name, email, phone)
VALUES ('assessment', 'truong.caonguyenvan@hcmut.edu.vn', '328355333')
ON CONFLICT (email) DO NOTHING;

-- Chèn địa chỉ người dùng
INSERT INTO addresses (user_id, province, district, commune, address_detail, house_type)
VALUES (
  (SELECT id FROM users WHERE email = 'truong.caonguyenvan@hcmut.edu.vn'),
  'Bắc Kạn', 'Ba Bể', 'Phúc Lộc', '73 tân hoà 2',
  'nhà riêng'
);

--Chèn thông tin cửa hàng
INSERT INTO stores (name, location)
VALUES ('Store 1', 'Hà Nội');
-- Chèn thông tin danh mục sản phẩm
-- Giả sử category_id là 1
INSERT INTO categories (name)
VALUES ('Sneakers');

-- Chèn thông tin sản phẩm
INSERT INTO products (name, price, category_id)
VALUES ('KAPPA Women''s Sneakers', 980000, 1);  -- category_id phải phù hợp với giá trị trong bảng categories


-- Chèn thông tin biến thể sản phẩm
INSERT INTO product_variants (product_id, size, color, quantity_in_stock, store_id)
VALUES (
  (SELECT id FROM products WHERE name = 'KAPPA Women''s Sneakers'),
  36, 'yellow', 5, 1  -- store_id giả sử là 1, có thể thay đổi theo cửa hàng cụ thể
);
-- Chèn đơn hàng
INSERT INTO orders (user_id, address_id, total_amount, status)
VALUES (
  (SELECT id FROM users WHERE email = 'truong.caonguyenvan@hcmut.edu.vn'),
  (SELECT id FROM addresses WHERE user_id = (SELECT id FROM users WHERE email = 'truong.caonguyenvan@hcmut.edu.vn')),
  980000, 'pending'
)
RETURNING id;
```

```
-- Chèn các mục đơn hàng
INSERT INTO order_items (order_id, product_variant_id, quantity, price_at_time_of_order)
VALUES (
  (SELECT id FROM orders WHERE user_id = (
      SELECT id
      FROM users
      WHERE email = 'truong.caonguyenvan@hcmut.edu.vn')
      ORDER BY order_date DESC LIMIT 1),
  (SELECT id FROM product_variants WHERE product_id = (
      SELECT id
      FROM products
      WHERE name = 'KAPPA Women''s Sneakers') AND size = 36 AND color = 'yellow'),
  1, 980000
);
```

Muốn truy vấn thông tin của người dùng ta sử dụng câu truy vấn:

```
SELECT
  u.name, u.email, u.phone, a.province, a.district, a.commune, a.address_detail,a.house_type
FROM
  users u
JOIN
  addresses a ON u.id = a.user_id
WHERE
  u.email = 'truong.caonguyenvan@hcmut.edu.vn';
```

ta được kết quả thông tin người dùng như sau:

	name text	email text	phone text	province text	district text	commune text	address_detail text	house_type text
1	assessment	truong.caonguyenvan@hcmut.edu.vn	328355333	Bắc Kạn	Ba Bể	Phúc Lộc	73 tân hoà 2	nhà riêng

Đây là câu lệnh truy vấn thông tin sản phẩm:

```
SELECT
  p.name AS product_name,
  p.price,
  pv.size,
  pv.quantity_in_stock AS quantity,
  pv.color
FROM
  products p
JOIN
  product_variants pv ON p.id = pv.product_id
WHERE
  p.name = 'KAPPA Women''s Sneakers' ;
```

Và ta được kết quả như sau:

	product_name text	price integer	size integer	quantity integer	color text
1	KAPPA Women's Sneakers	980000	36	5	yellow

c) Write a query to calculate the average order value (total price of items in an order) for each month in the current year.

```
SELECT
    EXTRACT(MONTH FROM o.order_date) AS month,
    AVG(oi.quantity * oi.price_at_time_of_order) AS average_order_value
FROM
    orders o
JOIN
    order_items oi ON o.id = oi.order_id
WHERE
    EXTRACT(YEAR FROM o.order_date) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY
    EXTRACT(MONTH FROM o.order_date)
ORDER BY
    month;
```

Giải thích:

- **EXTRACT(MONTH FROM o.order_date)**: Lấy tháng từ trường order_date trong bảng orders để nhóm các đơn hàng theo tháng.
- **AVG(oi.quantity * oi.price_at_time_of_order)**: Tính tổng giá trị các mặt hàng trong đơn hàng bằng cách nhân quantity với price_at_time_of_order, rồi tính giá trị trung bình cho mỗi tháng.
- **WHERE EXTRACT(YEAR FROM o.order_date) = EXTRACT(YEAR FROM CURRENT_DATE)**: Lọc ra các đơn hàng trong năm hiện tại.
- **GROUP BY EXTRACT(MONTH FROM o.order_date)**: Nhóm các đơn hàng theo tháng để tính giá trị trung bình theo từng tháng.
- **ORDER BY month**: Sắp xếp kết quả theo tháng để dễ dàng đọc.

Kết quả:

	month numeric	average_order_value numeric
1	5	980000.000000000000

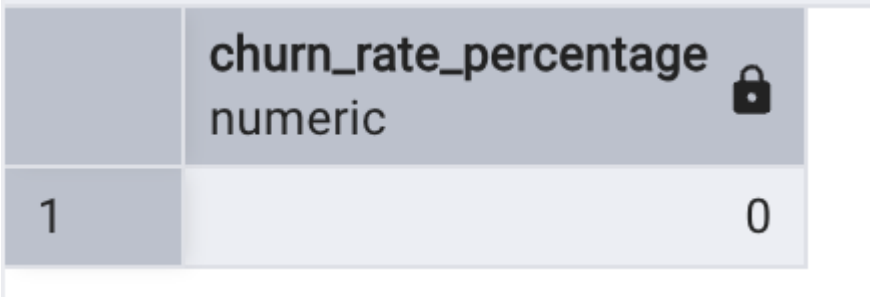
d) Write a query to calculate the churn rate of customers. The churn rate is defined as the percentage of customers who did not make a purchase in the last 6 months but had made a purchase in the 6 months prior to that.

```sql
WITH recent_purchases AS (
    -- Lấy khách hàng đã thực hiện mua hàng trong 12 tháng qua
    SELECT DISTINCT o.user_id
    FROM orders o
    WHERE o.order_date > CURRENT_DATE - INTERVAL '12 months'
),
last_6_months AS (
    -- Lấy khách hàng đã thực hiện mua hàng trong 6 tháng qua
    SELECT DISTINCT o.user_id
    FROM orders o
    WHERE o.order_date > CURRENT_DATE - INTERVAL '6 months'
),
previous_6_months AS (
    -- Lấy khách hàng đã thực hiện mua hàng trong 6 đến 12 tháng trước
    SELECT DISTINCT o.user_id
    FROM orders o
    WHERE o.order_date BETWEEN CURRENT_DATE - INTERVAL '12 months' AND CURRENT_DATE - INTERVAL '6 months'
)
-- Tính toán tỷ lệ khách hàng rời bỏ
SELECT
  CASE
        WHEN COUNT(DISTINCT rp.user_id) = 0 THEN 0
        ELSE ROUND(COUNT(DISTINCT psm.user_id) * 100.0 / COUNT(DISTINCT rp.user_id), 2)
    END AS churn_rate_percentage
FROM previous_6_months psm
JOIN recent_purchases rp ON psm.user_id = rp.user_id
LEFT JOIN last_6_months lsm ON psm.user_id = lsm.user_id
WHERE lsm.user_id IS NULL;
```

Giải thích:

- **recent_purchases**: Lấy danh sách khách hàng đã thực hiện mua hàng trong 12 tháng qua.
- **last_6_months**: Lấy danh sách khách hàng đã thực hiện mua hàng trong 6 tháng qua.
- **previous_6_months**: Lấy danh sách khách hàng đã thực hiện mua hàng trong khoảng thời gian từ 6 đến 12 tháng trước.
- Kết hợp **previous_6_months** với **recent_purchases** để đảm bảo rằng khách hàng đã thực hiện mua hàng trong 12 tháng qua. Sử dụng LEFT JOIN với last_6_months và lọc ra các khách hàng không thực hiện mua hàng trong 6 tháng qua (lsm.user_id IS NULL).
- Tính toán tỷ lệ churn bằng cách lấy số khách hàng trong nhóm **previous_6_months** không có trong **last_6_months**, chia cho tổng số khách hàng đã mua hàng trong 12 tháng qua.

Kết quả:



	churn_rate_percentage numeric
1	0

e) Write API specifications, API according to RESTful standards to:
i) Fetches a list of all product categories available in the e-commerce platform.
->
Endpoint: GET /api/categories
Mô tả: Trả về danh sách tất cả các danh mục sản phẩm.
Response (200):

```
[
  {
    "id": 1,
    "name": "Sneakers"
  },
 ]
```

ii) Fetches a list of products that belong to a specific category.
Endpoint: GET /api/categories/:categoryName/products
Params:

- categoryName: tên của danh mục cần lấy sản phẩm.

Response (200):

```
{
   "category": "sneak",
   "total": 1,
   "products": [
      {
         "id": 1,
         "name": "KAPPA Women's Sneakers",
         "category_id": 1,
         "price": 980000
      }
   ]
}
```

iii) Allows users to search (full-text search) for products using various filters and search terms.

Endpoint: GET /api/products/search
Query Parameters:

- q: từ khóa tìm kiếm
- category_id: lọc theo danh mục
- min_price, max_price: lọc theo khoảng giá

Ví dụ:
GET /api/products/search?q=sneakers&category_id=1&min_price=500000
Response (200):

```
[
  {
    "id": 1,
    "name": "KAPPA Women's Sneakers",
    "category_id": 1,
    "price": 980000
  }
]
```

iv) Creates a new order and processes payment.

Endpoint: POST /api/orders

Request Body:

```
{
 "userId": 1,
 "addressId": 1,
 "totalAmount": 980000,
 "items": [
  {
    "variantId": 1,
    "quantity": 1,
    "price": 980000
  }
 ]
}
```

Response (201):

```
{
```

"order_id": 6,
"status": "processing",
"total_amount": 980000,
"message": "Order created successfully"
}

v) Send order confirmation email to user (processed asynchronously with order creation flow).

Cơ chế: Sau khi đơn hàng được tạo thành công: hệ thống sẽ gởi email tự động đến email của khách hàng đã được đăng kí trên hệ thống
ví dụ từ jop payload:

{
"order_id": 1,
"user_email": "truong.caonguyenvan@hcmut.edu.vn
}

hệ thống sẽ gởi email với thông tin:
Subject: xác nhận đơn hàng - #123
Xin chào {user_name }, đơn hàng của bạn đặt hàng thành công và đơn giá là 980000đ.

f) Implement the above APIs with the language/framework you choose.

1. Backend Overview

- **Ngôn ngữ:** JavaScript (Node.js)
- **Framework:** Express.js (Dùng cho routing và middleware)
- **Cơ sở dữ liệu:** PostgreSQL (Sử dụng pg - thư viện node-postgres để kết nối và tương tác với cơ sở dữ liệu)
- **API:** RESTful APIs để thực hiện các chức năng như lấy danh mục sản phẩm, tìm kiếm sản phẩm theo danh mục, tìm kiếm sản phẩm theo tên, giá, và tạo đơn hàng. Các API này hỗ trợ lọc, phân trang và sắp xếp kết quả tìm kiếm.

Công cụ và thư viện sử dụng:

- **Twilio SendGrid**: Dùng để gửi email xác nhận đơn hàng cho người dùng sau khi đơn hàng được tạo thành công.
- **pg (node-postgres)**: Dùng để kết nối và tương tác với cơ sở dữ liệu PostgreSQL.
- **Express.js**: Cung cấp routing và middleware để xử lý các yêu cầu HTTP.

Chi tiết các API:

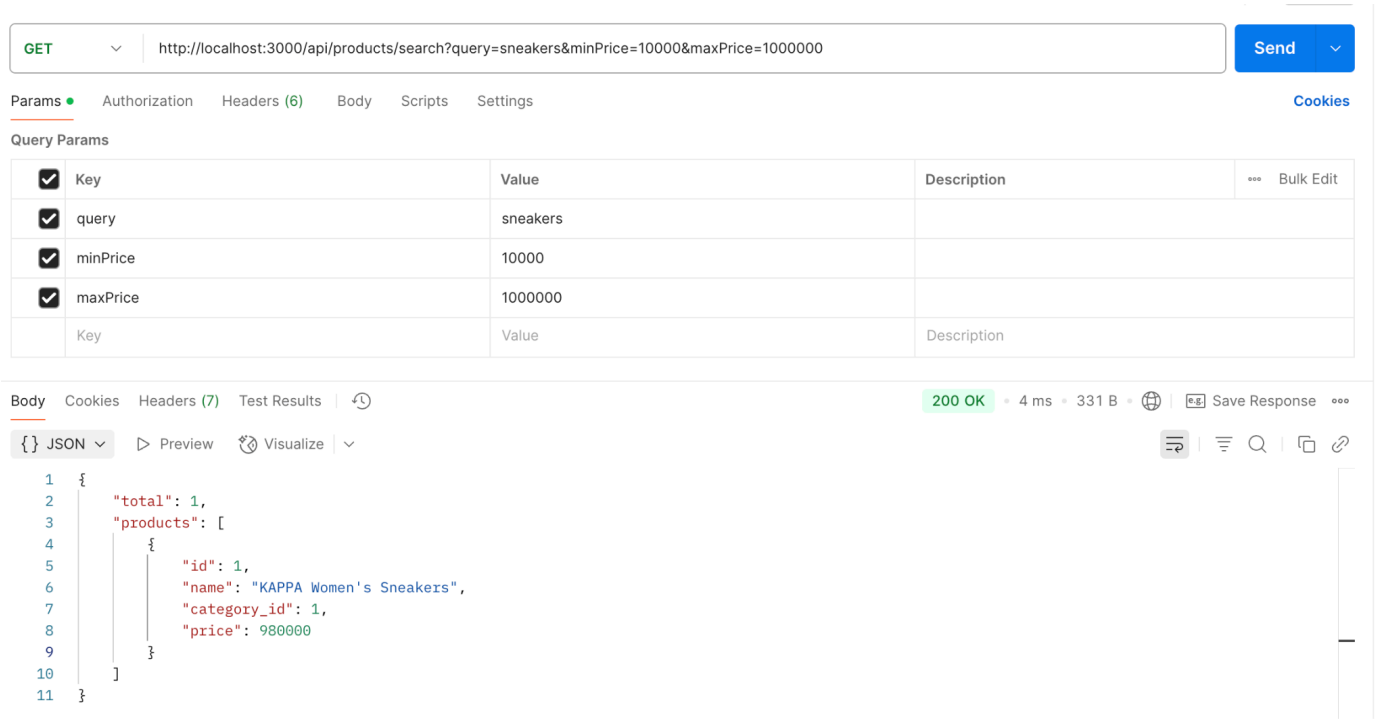
1. **GET /api/products/categories**: Lấy tất cả danh mục sản phẩm từ cơ sở dữ liệu.
2. **GET/api/products/categories/Sneakers/products:** Lấy sản phẩm theo tên danh mục, cho phép tìm kiếm sản phẩm dựa trên danh mục.
3. **GET /api/products/search?query=sneakers&minPrice=10000&maxPrice=1000000**: Tìm kiếm sản phẩm theo tên, giá, và sắp xếp kết quả tìm kiếm.
4. **POST /api/orders**: Tạo đơn hàng mới, bao gồm các mặt hàng trong đơn, tính toán tổng tiền và gửi email xác nhận đơn hàng.

Tính năng chính:

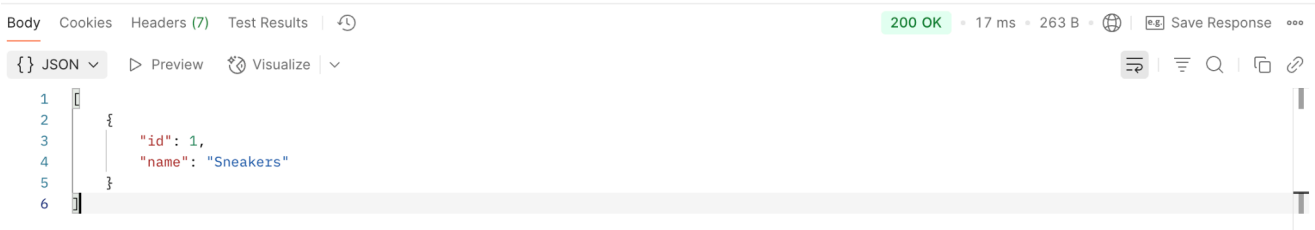
- **Lấy dữ liệu** từ cơ sở dữ liệu PostgreSQL qua các truy vấn SQL động.
- **Quản lý đơn hàng**: Tạo đơn hàng mới với các thông tin về sản phẩm, số lượng, giá và tính toán tổng giá trị đơn hàng.
- **Gửi email** xác nhận đơn hàng khi đơn hàng được tạo thành công.
- **Lọc và sắp xếp sản phẩm**: Cho phép tìm kiếm sản phẩm theo tên, giá và danh mục, với khả năng phân trang và sắp xếp theo các tiêu chí người dùng mong muốn.

Dưới đây là kết quả demo của các API thông qua postman:

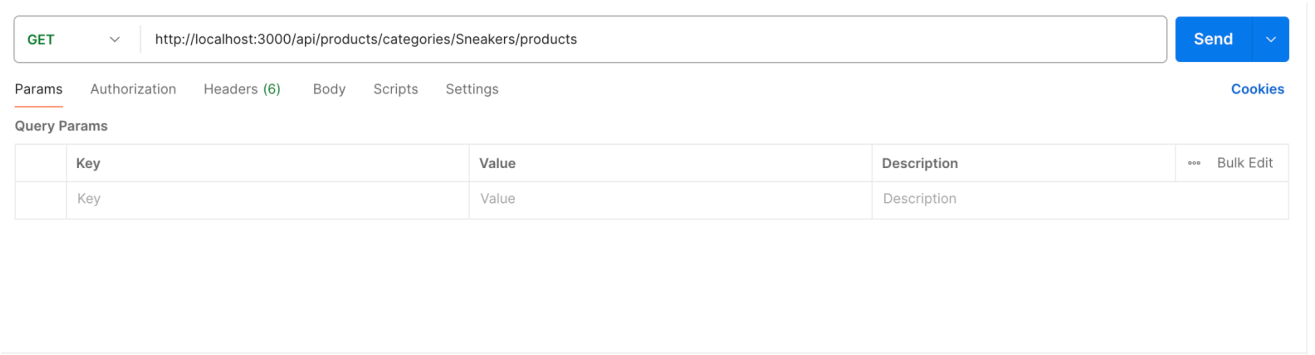
- API tìm kiếm sản phẩm theo tên(sneakers) và theo giá trị (nếu có).



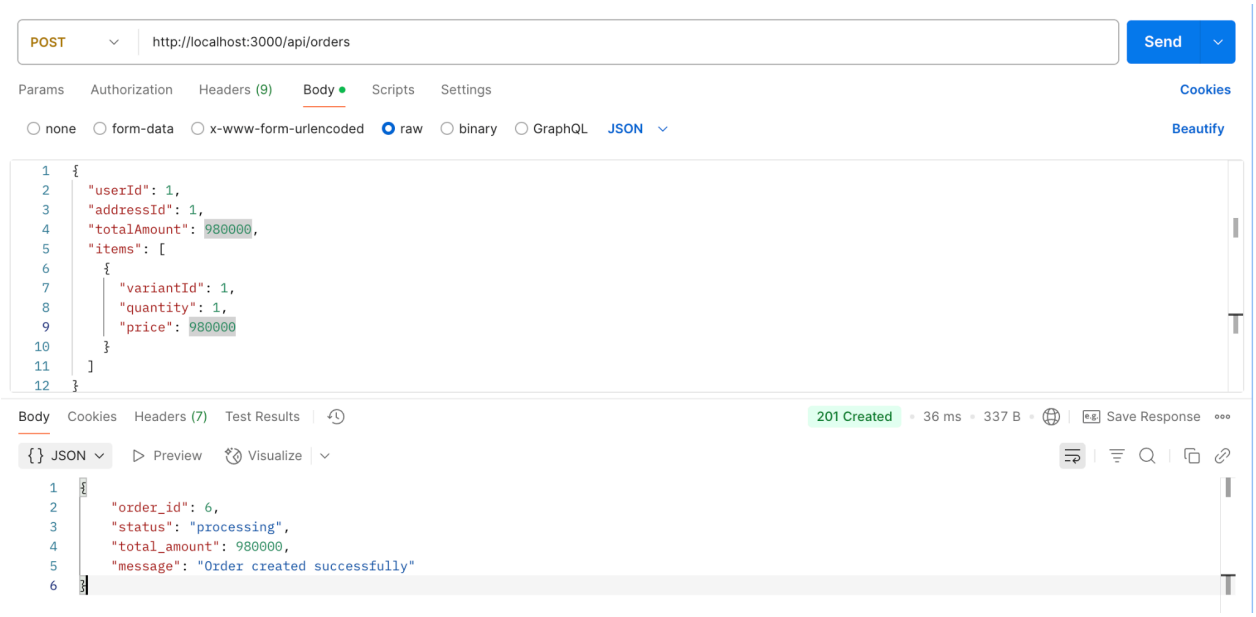
- Lấy tất cả danh mục sản phẩm từ cơ sở dữ liệu.



- Lấy sản phẩm theo tên danh mục (danh mục sneakers)



- Tạo đơn hàng mới và gửi email xác nhận đơn hàng.



Đây là email được gởi về