1. Question 1

1.1 DNS (Hệ thống phân giải tên miền)

- **Vai trò:** Chuyển đổi tên miền thành địa chỉ IP.
- **Chức năng:**
  - Giúp người dùng truy cập dịch vụ qua tên miền thay vì phải nhớ địa chỉ IP.
  - Tự động chuyển hướng người dùng đến máy chủ gần nhất, tối ưu hiệu suất và giảm độ trễ khi kết nối.
- **Tác động:** Đảm bảo kết nối giữa người dùng và máy chủ được nhanh chóng, chính xác và ít độ trễ, giúp cải thiện trải nghiệm người dùng.

1.2 Load Balancer (Bộ cân bằng tải)

- **Vai trò:** Phân phối lưu lượng truy cập đến nhiều máy chủ để đảm bảo hệ thống không bị quá tải.
- **Chức năng:**
  - Phân phối yêu cầu người dùng một cách đồng đều giữa các máy chủ để tránh một máy chủ bị quá tải.
  - Hỗ trợ khả năng mở rộng hệ thống và cung cấp dự phòng khi có máy chủ gặp sự cố.
- **Tác động:** Đảm bảo hệ thống vẫn hoạt động ổn định dưới tải cao và có thể xử lý lượng người dùng đồng thời mà không làm giảm hiệu suất.

1.3 Web app servers (Các máy chủ ứng dụng web)

- **Vai trò:** Xử lý các yêu cầu từ người dùng, thực hiện logic ứng dụng và trả về dữ liệu.
- **Chức năng:**
  - Tiếp nhận và xử lý các yêu cầu HTTP/HTTPS từ người dùng.
  - Liên lạc với các thành phần khác của hệ thống như cơ sở dữ liệu, bộ nhớ đệm, hoặc các dịch vụ khác.
- **Tác động:** Việc sử dụng nhiều máy chủ ứng dụng web hoạt động cùng một lúc giúp tăng khả năng xử lý và giúp hệ thống đáp ứng nhanh chóng các yêu cầu từ người dùng.

1.4 Database (Cơ sở dữ liệu)

- **Vai trò:** Lưu trữ và quản lý dữ liệu ứng dụng.
- **Chức năng:**
  - Lưu trữ thông tin người dùng, giao dịch, và các dữ liệu ứng dụng khác.
  - Hỗ trợ các truy vấn phức tạp từ máy chủ ứng dụng web để lấy và lưu trữ dữ liệu.
- **Tác động:** Cung cấp khả năng truy xuất dữ liệu nhanh chóng và chính xác, đảm bảo hệ thống luôn có dữ liệu mới nhất và đáp ứng yêu cầu người dùng. Giảm thiểu dư thừa dữ liệu và phục vụ cho số lượng dữ liệu lớn hơn so với cách tiếp cận file.

1.5 Caching Server (Dịch vụ bộ nhớ đệm)

- **Vai trò:** Lưu trữ tạm thời dữ liệu được truy cập thường xuyên để giảm tải cho cơ sở dữ liệu.
- **Chức năng:**
  - Giúp tăng tốc độ truy xuất dữ liệu bằng cách lưu trữ tạm thời trong bộ nhớ, tránh phải truy vấn cơ sở dữ liệu mỗi lần.
  - Dữ liệu được lưu trữ tạm thời có thể là kết quả của các truy vấn hoặc dữ liệu tĩnh.
- **Tác động:** Giảm thời gian phản hồi, giúp hệ thống phản hồi nhanh hơn cho người dùng.

1.6 Job queue and job servers (Hàng đợi công việc và máy chủ công việc)

- **Vai trò:** Xử lý các tác vụ không đồng bộ, giảm tải cho máy chủ ứng dụng web.
- **Chức năng:**
  - **Hàng đợi công việc:** Chứa các tác vụ cần thực hiện nhưng không cần phải xử lý ngay lập tức.
  - **Máy chủ công việc:** Xử lý các tác vụ không đồng bộ, chẳng hạn như gửi email, tạo báo cáo hoặc xử lý thông tin lâu dài.
- **Tác động:** Giảm tải cho máy chủ ứng dụng web và cải thiện hiệu suất của hệ thống, giúp hệ thống duy trì khả năng đáp ứng cao.

1.7 Full text search serice (Dịch vụ tìm kiếm toàn văn)

- **Vai trò:** Cung cấp khả năng tìm kiếm toàn văn trong hệ thống.
- **Chức năng:**
  - Cho phép người dùng tìm kiếm nhanh chóng các sản phẩm, bài viết hoặc tài liệu dựa trên các từ khóa.

- **Tác động:** Giảm độ trễ khi người dùng tìm kiếm và cải thiện trải nghiệm người dùng bằng cách cung cấp kết quả tìm kiếm nhanh chóng và chính xác.

1.8 Services (Các dịch vụ bổ trợ)

- **Vai trò:** Cung cấp các dịch vụ hỗ trợ như xử lý thanh toán hoặc xác thực người dùng.
- **Tác động:** Giúp hệ thống trở nên linh hoạt và dễ dàng mở rộng, cho phép thêm các chức năng bổ sung mà không làm phức tạp hệ thống chính.

1.9 Data Firehose

- **Vai trò:**
  - **Luồng dữ liệu:** Thu thập và xử lý dữ liệu thời gian thực từ ứng dụng.
  - **Chuyển đổi dữ liệu:** Xử lý cơ bản như nén, mã hóa, hoặc chuyển đổi dữ liệu trước khi lưu trữ.
  - **Truyền tải dữ liệu:** Gửi dữ liệu đến các đích như bộ nhớ đám mây (Cloud) hoặc kho dữ liệu (Data warehouse) để phân tích hoặc lưu trữ.
- **Tác động:**
  - Hỗ trợ xử lý và truyền tải dữ liệu theo thời gian thực, giúp các hệ thống phản hồi nhanh chóng trước các sự kiện.
  - Giảm tải công việc xử lý cho các hệ thống lưu trữ và phân tích.

1.10 Data Warehouse

- **Vai trò:**
  - **Lưu trữ dữ liệu lịch sử:** Tập hợp và quản lý dữ liệu từ nhiều nguồn để phục vụ phân tích và báo cáo.
  - **Tối ưu cho phân tích:** Tạo môi trường lý tưởng cho các truy vấn dữ liệu phức tạp và phân tích dài hạn.
  - **Tích hợp công cụ BI:** Hỗ trợ kết nối với các công cụ kinh doanh thông minh (BI) như Tableau hoặc Power BI.
- **Tác động:**
  - Cung cấp dữ liệu có tổ chức để phân tích, giúp đưa ra quyết định chiến lược hiệu quả hơn.
  - Giảm tải cho các hệ thống giao dịch, tập trung vào lưu trữ dữ liệu và phân tích chuyên sâu.

1.11 Cloud Storage (Lưu trữ đám mây)

- **Vai trò:**
  - Lưu trữ dữ liệu phi cấu trúc và ít thay đổi, chẳng hạn như tệp, hình ảnh, video, hoặc sao lưu dữ liệu quan trọng, trên các hệ thống đám mây có khả năng mở rộng.
  - Tích hợp với các dịch vụ mạng phân phối nội dung (CDN) để cung cấp dữ liệu hiệu quả hơn thông qua bộ nhớ cache toàn cầu.
- **Tác động:**
  - Giảm chi phí vận hành và bảo trì hệ thống lưu trữ nội bộ, tăng khả năng truy cập và chia sẻ dữ liệu linh hoạt từ bất kỳ đâu, đồng thời đảm bảo an toàn và sao lưu dữ liệu.
  - Cung cấp dữ liệu nhanh chóng và ổn định trên toàn cầu thông qua CDN, giảm độ trễ khi truy cập nội dung.

1.12 CDN (Mạng phân phối nội dung)

- **Vai trò:**
  - Phân phối các nội dung tĩnh (hình ảnh, video) từ Cloud Storage đến các máy chủ biên gần người dùng hơn.
  - Tối ưu hóa quá trình tải dữ liệu từ Cloud Storage bằng cách sử dụng bộ nhớ đệm tại các điểm phân phối CDN.
- **Tác động:**
  - Tăng tốc độ tải trang và giảm độ trễ bằng cách phục vụ nội dung từ các máy chủ gần người dùng nhất.
  - Giảm tải cho hệ thống Cloud Storage chính nhờ việc sử dụng bộ nhớ đệm (cache) tại các điểm phân phối CDN.
  - Đảm bảo khả năng mở rộng và hiệu suất cao khi có lượng truy cập lớn từ nhiều khu vực khác nhau.

1.13 Hiệu năng hệ thống

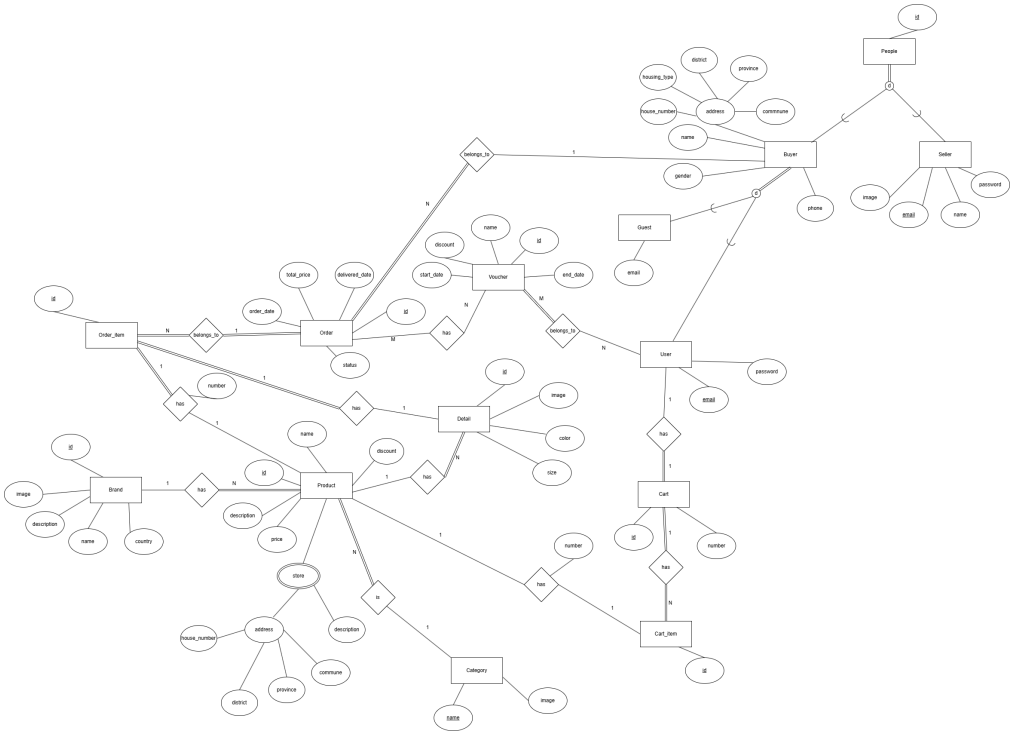
- **Độ trễ:** $<$ 1 giây (P99) đạt được nhờ CDN, bộ nhớ đệm và cân bằng tải.
- **Thông lượng:** 70–80 yêu cầu mỗi giây được hỗ trợ nhờ hạ tầng mở rộng.
- **Dung lượng:** Xử lý $>$ 1,000 người dùng tổng và $>$ 100 người dùng đồng thời một cách hiệu quả.

2. Question 2

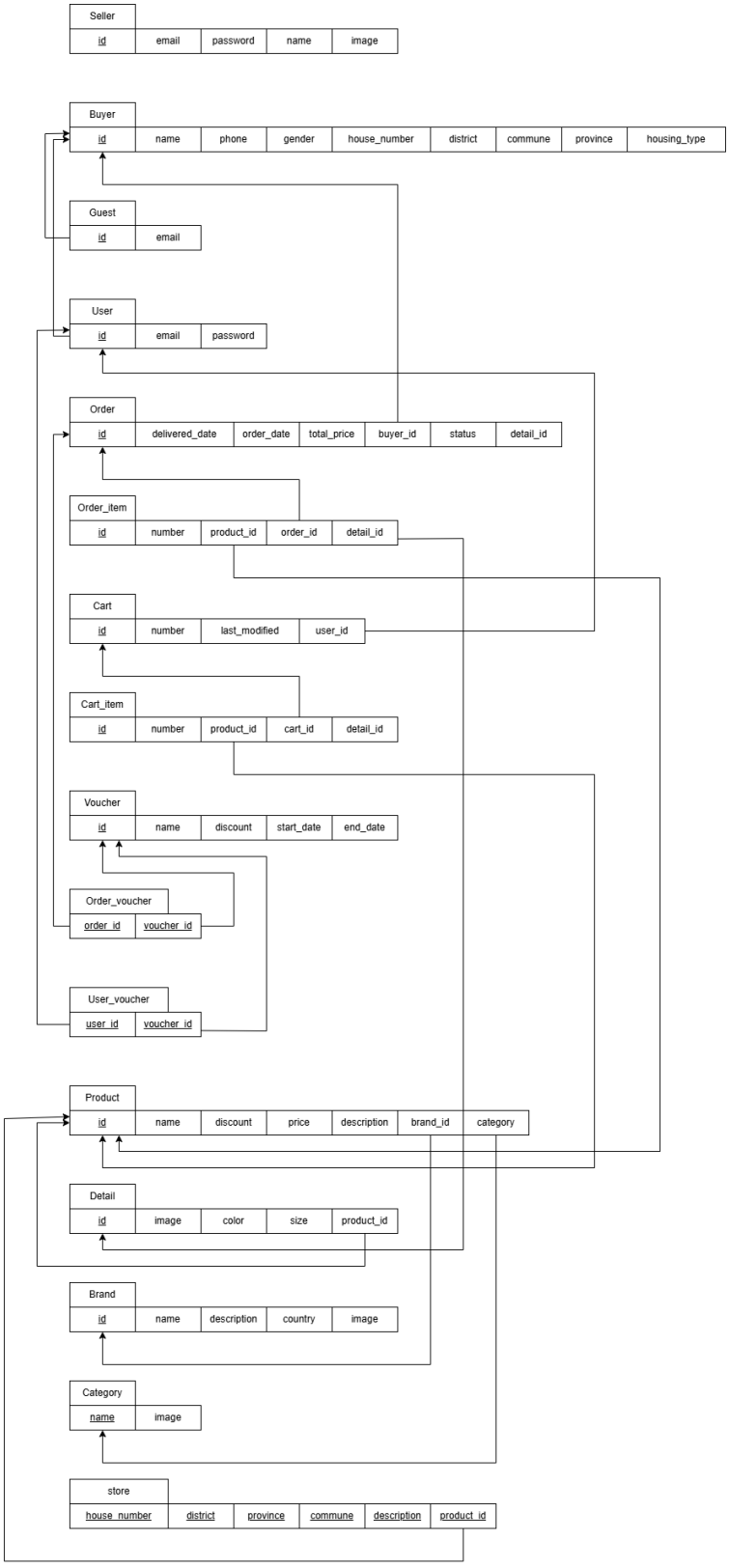
2.1 Xây dựng cơ sở dữ liệu

2.1.1 Các lược đồ ER và lược đồ quan hệ

Vì hình khá hơn và bị mờ khi thu nhỏ nên em xin phép được để 2 hình ở trong thư mục database (ERD và relational Schema) ạ. Các câu lệnh tạo bảng được thêm vào thư mục database/Create Table.sql.



Hình 1: ERD



Hình 2: Relational Schema

2.1.2 Bảng seller

- **id**: Mã định danh người bán, tự tăng.
- **email**: Email của người bán, không được trùng lặp.
- **password**: Mật khẩu của người bán.
- **name**: Tên của người bán.
- **image**: Hình ảnh của người bán.

2.1.3 Bảng buyer

- **id**: Mã định danh người mua, tự tăng.
- **name**: Tên của người mua.
- **phone**: Số điện thoại của người mua.
- **gender**: Giới tính của người mua.
- **house_number**: Số nhà của người mua.
- **district**: Quận/huyện của người mua.
- **commune**: Xã/phường của người mua.
- **province**: Tỉnh/thành phố của người mua.
- **housing_type**: Loại hình nhà ở của người mua.

2.1.4 Bảng guest

- **id**: Mã định danh khách mời, khóa chính.
- **email**: Email của khách mời.
- **fk_guest_buyer**: Khóa ngoại liên kết với buyer(id).

2.1.5 Bảng user

- **id**: Mã định danh người dùng, khóa chính.
- **email**: Email của người dùng, không được trùng lặp.
- **password**: Mật khẩu của người dùng.
- **fk_user_buyer**: Khóa ngoại liên kết với buyer(id).

2.1.6 Bảng brand

- **id**: Mã định danh thương hiệu, tự tăng.
- **name**: Tên thương hiệu, không được trùng lặp.
- **description**: Mô tả về thương hiệu.
- **country**: Quốc gia của thương hiệu.
- **image**: Hình ảnh của thương hiệu.

2.1.7 Bảng category

- **name**: Tên danh mục, khóa chính.
- **image**: Hình ảnh của danh mục.

2.1.8 Bảng product

- **id**: Mã định danh sản phẩm, tự tăng.
- **name**: Tên sản phẩm.
- **discount**: Mức giảm giá cho sản phẩm.
- **price**: Giá sản phẩm.
- **description**: Mô tả sản phẩm.
- **brand_id**: Khóa ngoại liên kết với brand(id).
- **category**: Khóa ngoại liên kết với category(name).

2.1.9 Bảng detail

- **id**: Mã định danh thông tin chi tiết sản phẩm, khóa chính.
- **product_id**: Mã định danh sản phẩm.
- **image**: Hình ảnh của sản phẩm.
- **color**: Màu sắc của sản phẩm.
- **size**: Kích thước của sản phẩm.

2.1.10 Bảng store

- **house_number**: Số nhà của cửa hàng.
- **district**: Quận/huyện của cửa hàng.
- **province**: Tỉnh/thành phố của cửa hàng.
- **commune**: Xã/phường của cửa hàng.

- **description**: Mô tả về cửa hàng.
- **product_id**: Khóa ngoại liên kết với product(id).

2.1.11 Bảng order

- **id**: Mã định danh đơn hàng, tự tăng.
- **delivered_date**: Ngày giao hàng.
- **order_date**: Ngày đặt hàng.
- **total_price**: Tổng giá trị đơn hàng.
- **buyer_id**: Khóa ngoại liên kết với buyer(id).
- **status**: Trạng thái đơn hàng.

2.1.12 Bảng order_item

- **id**: Mã định danh sản phẩm trong đơn hàng.
- **number**: Số lượng sản phẩm.
- **product_id**: Khóa ngoại liên kết với product(id).
- **order_id**: Khóa ngoại liên kết với order(id).
- **detail_id**: Khóa ngoại liên kết với detail(id).

2.1.13 Bảng cart

- **id**: Mã định danh giỏ hàng.
- **number**: Số lượng sản phẩm trong giỏ.
- **last_modified**: Thời gian chỉnh sửa cuối cùng.
- **user_id**: Khóa ngoại liên kết với user(id).

2.1.14 Bảng cart_item

- **id**: Mã định danh sản phẩm trong giỏ hàng.
- **number**: Số lượng sản phẩm trong giỏ.
- **product_id**: Khóa ngoại liên kết với product(id).
- **cart_id**: Khóa ngoại liên kết với cart(id).
- **detail_id**: Khóa ngoại liên kết với detail(id).

2.1.15 Bảng voucher

- **id**: Mã định danh mã giảm giá.
- **name**: Tên mã giảm giá.
- **discount**: Mức giảm giá.
- **start_date**: Ngày bắt đầu mã giảm giá.
- **end_date**: Ngày hết hạn mã giảm giá.

2.1.16 Bảng order_voucher

- **order_id**: Khóa ngoại liên kết với order(id).
- **voucher_id**: Khóa ngoại liên kết với voucher(id).

2.1.17 Bảng user_voucher

- **user_id**: Khóa ngoại liên kết với user(id).
- **voucher_id**: Khóa ngoại liên kết với voucher(id).

2.1.18 Dạng chuẩn hóa

Ở đây, em đã thiết kế cơ sở dữ liệu theo dạng chuẩn hóa BCNF với các đặc điểm sau:

- Mỗi bảng có các hàng duy nhất và không có nhóm giá trị lặp lại.
- Tất cả các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính.
- Tất cả các thuộc tính không khóa chỉ phụ thuộc vào khóa chính.
- Tất cả các phần tử ở vế trái của phụ thuộc hàm đều là khóa chính.

Chỉ duy nhất bảng detail chỉ được chuẩn hóa theo dạng 2NF do image có thể xác định được các hàm khác

2.2 Viết câu lệnh insert để tạo một order

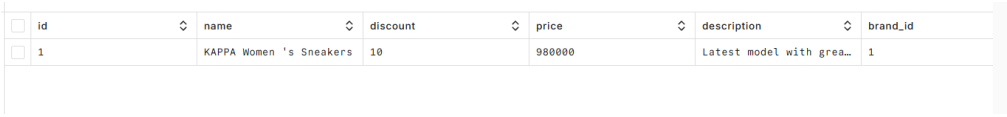
Câu lệnh được viết trong thư mục database/test.sql. Ở đây, em viết câu lệnh để thêm dữ liệu khi chưa có dữ liệu khác được thêm vào.

id	name	phone	gender	house_number	district
1	assessment	328355333	Male	73 tân hoà 2	Ba Bé

Hình 3: Thêm người mua

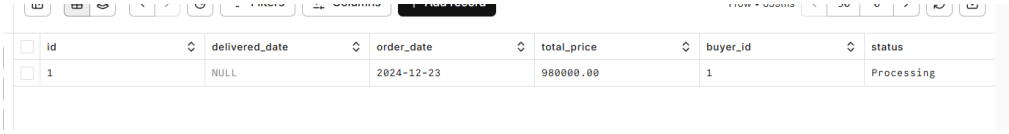
2.3 Viết một câu truy vấn để tính giá trị trung bình của đơn hàng (tổng giá trị các mặt hàng trong đơn hàng) cho từng tháng trong năm hiện tại

Câu lệnh được viết trong thư mục database/test.sql



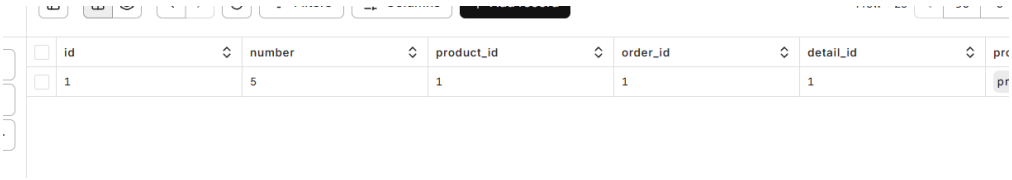
id	name	discount	price	description	brand_id
1	KAPPA Women 's Sneakers	10	980000	Latest model with grea…	1

Hình 4: Thêm sản phẩm



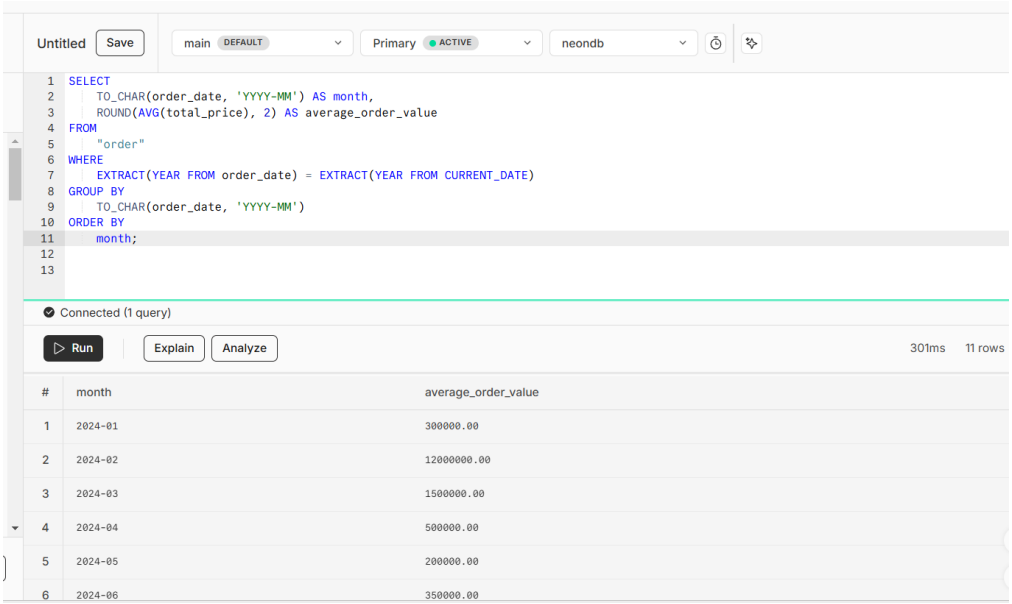
id	delivered_date	order_date	total_price	buyer_id	status
1	NULL	2024-12-23	980000.00	1	Processing

Hình 5: Thêm đơn hàng



id	number	product_id	order_id	detail_id	pr
1	5	1	1	1	

Hình 6: Thêm sản phẩm trong đơn hàng



Untitled Save main DEFAULT Primary ACTIVE neondb

```
SELECT
    TO_CHAR(order_date, 'YYYY-MM') AS month,
    ROUND(AVG(total_price), 2) AS average_order_value
FROM
    "order"
WHERE
    EXTRACT(YEAR FROM order_date) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY
    TO_CHAR(order_date, 'YYYY-MM')
ORDER BY
    month;
```

Connected (1 query)

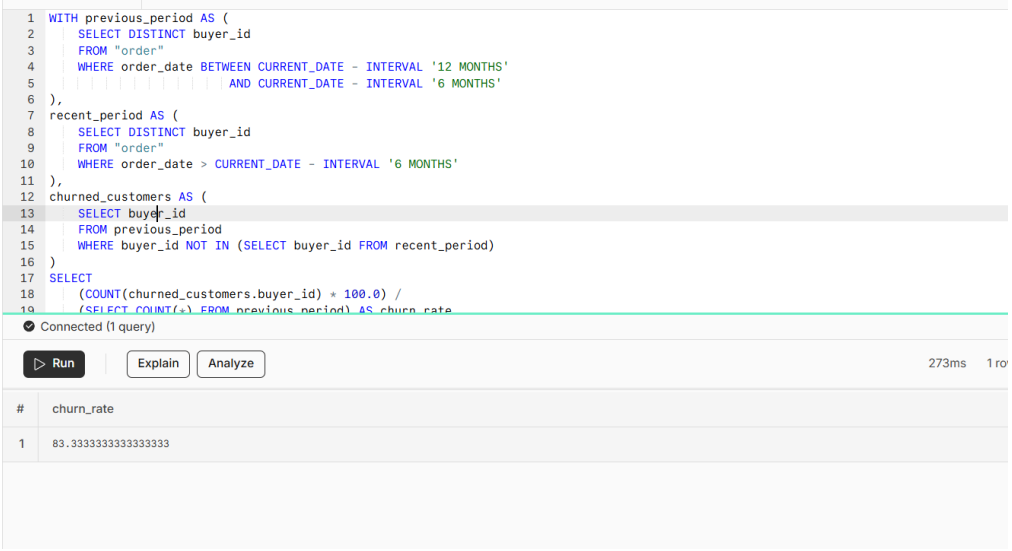
Run Explain Analyze 301ms 11 rows

#	month	average_order_value
1	2024-01	300000.00
2	2024-02	12000000.00
3	2024-03	1500000.00
4	2024-04	500000.00
5	2024-05	200000.00
6	2024-06	350000.00

Hình 7: Trung bình

2.4 Viết một câu truy vấn SQL để tính tỷ lệ churn rate của khách hàng, theo định nghĩa là tỷ lệ phần trăm khách hàng không thực hiện mua hàng trong 6 tháng gần nhất nhưng đã thực hiện mua hàng trong 6 tháng trước đó

Câu lệnh được viết trong thư mục database/test.sql.



Hình 8: Churn rate

2.5 Mô tả các RESTFUL api

2.5.1 API: Lấy tất cả sản phẩm

- **Phương thức**: `GET`
- **Endpoint**: `http://localhost:3900/api/v1/product/getAllProducts`
- **Chức năng**: API này được sử dụng để lấy danh sách tất cả sản phẩm từ cơ sở dữ liệu.
- **Đầu vào**: Không yêu cầu tham số đầu vào.
- **Đầu ra**:
  - **Trường hợp thành công (HTTP 200 OK)**:

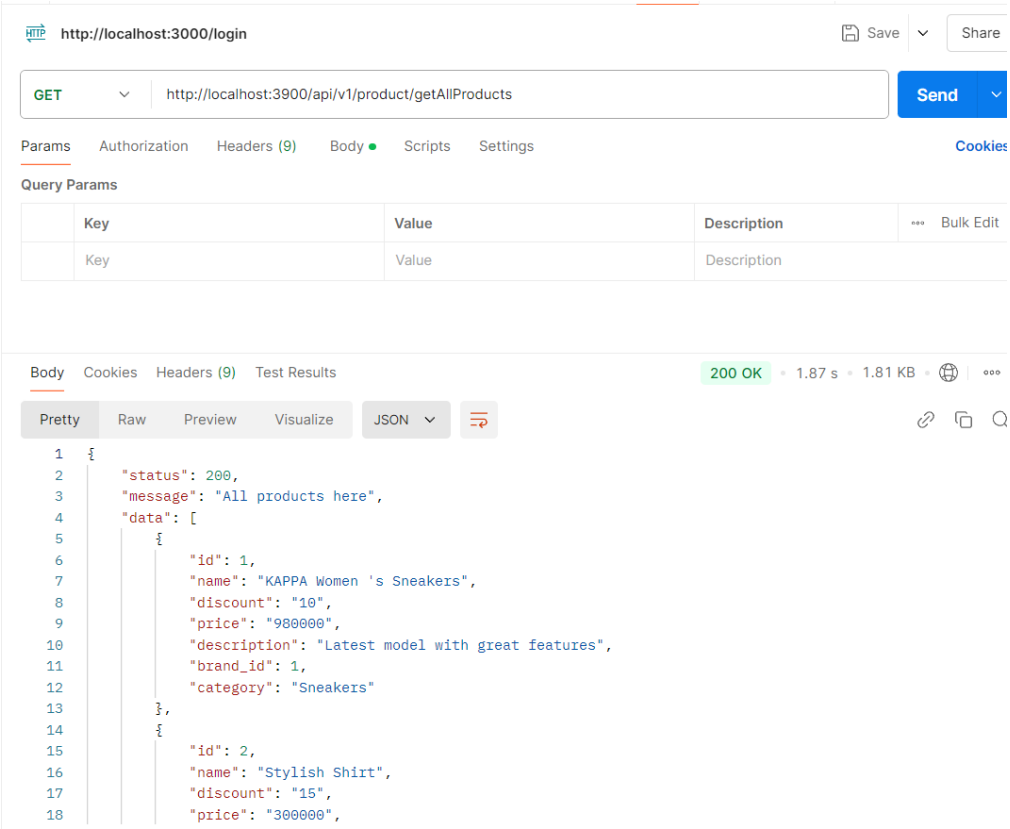
```
{
    "message": "Products fetched successfully",
    "data": [ /* Mảng các sản phẩm */ ]
}
```

    * `message`: Chuỗi thông báo thành công.
    * `data`: Một mảng chứa danh sách các sản phẩm.
  - **Trường hợp thất bại do lỗi logic ứng dụng (HTTP 500 Internal Server Error)**:

```
{
    "message": "Mô tả lỗi từ result.message",
    "error": "Thông tin chi tiết lỗi từ result.error"
}
```

  - **Trường hợp xảy ra lỗi bất ngờ (HTTP 500 Internal Server Error)**:

```
{
    "message": "Unexpected error occurred",
    "error": "Chi tiết lỗi"
}
```



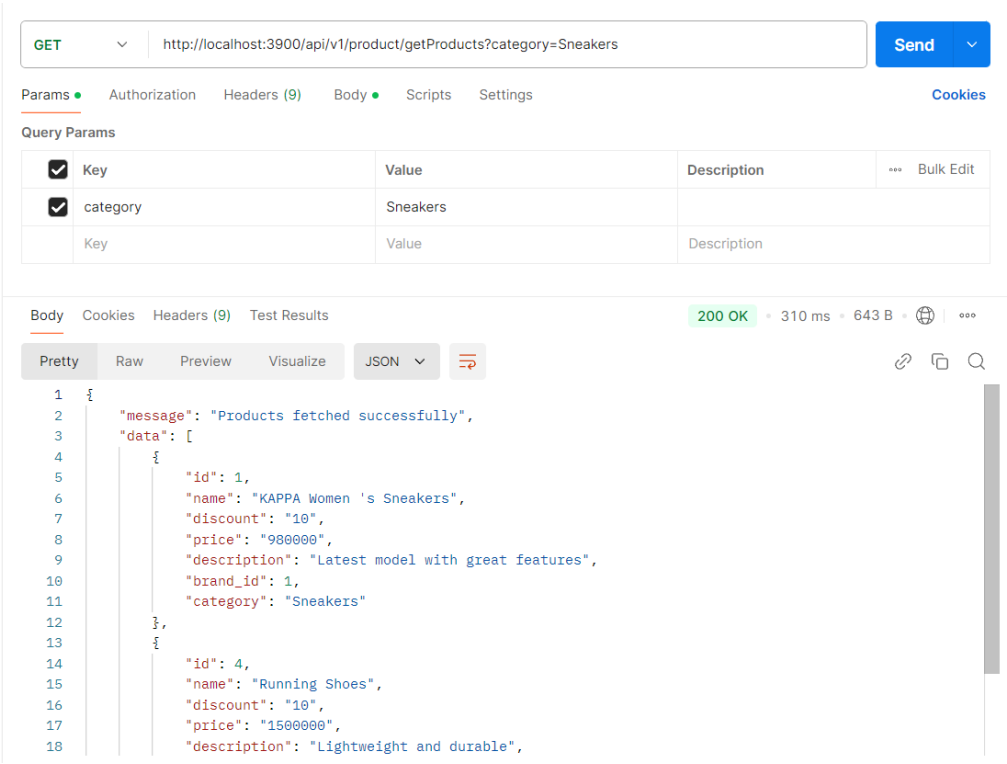
Hình 9: Get all Products

2.5.2 API: Lấy sản phẩm theo danh mục

- **Phương thức**: `GET`
- **Endpoint:** `http://localhost:3900/api/v1/product/getProducts?category=:category`
- **Mục đích:** Lấy danh sách sản phẩm theo danh mục được chỉ định.
- **Tham số yêu cầu:**
  - `category`: Tham số bắt buộc, kiểu chuỗi. Chỉ định danh mục sản phẩm.
- **Luồng xử lý:**
  1. Lấy tham số `category` từ `req.query`.
  2. Kiểm tra nếu `category` không tồn tại:
     - Trả về mã trạng thái `400 Bad Request` với thông báo: *"Category is required"*.
  3. Gọi phương thức `productModel.getProducts(category)` từ tầng model.

4. Kiểm tra kết quả từ tầng model:
   - Nếu thành công, trả về danh sách sản phẩm với mã trạng thái `200 OK`.
   - Nếu xảy ra lỗi, trả về mã trạng thái `400 Bad Request` cùng thông báo lỗi từ model.
5. Nếu có lỗi không mong muốn, trả về mã trạng thái `500 Internal Server Error`.

- **Phản hồi:**
  - `200 OK`: Trả về danh sách sản phẩm.
  - `400 Bad Request`: Thông báo lỗi nếu tham số không hợp lệ hoặc thiếu.
  - `500 Internal Server Error`: Thông báo lỗi nếu xảy ra lỗi không mong muốn.



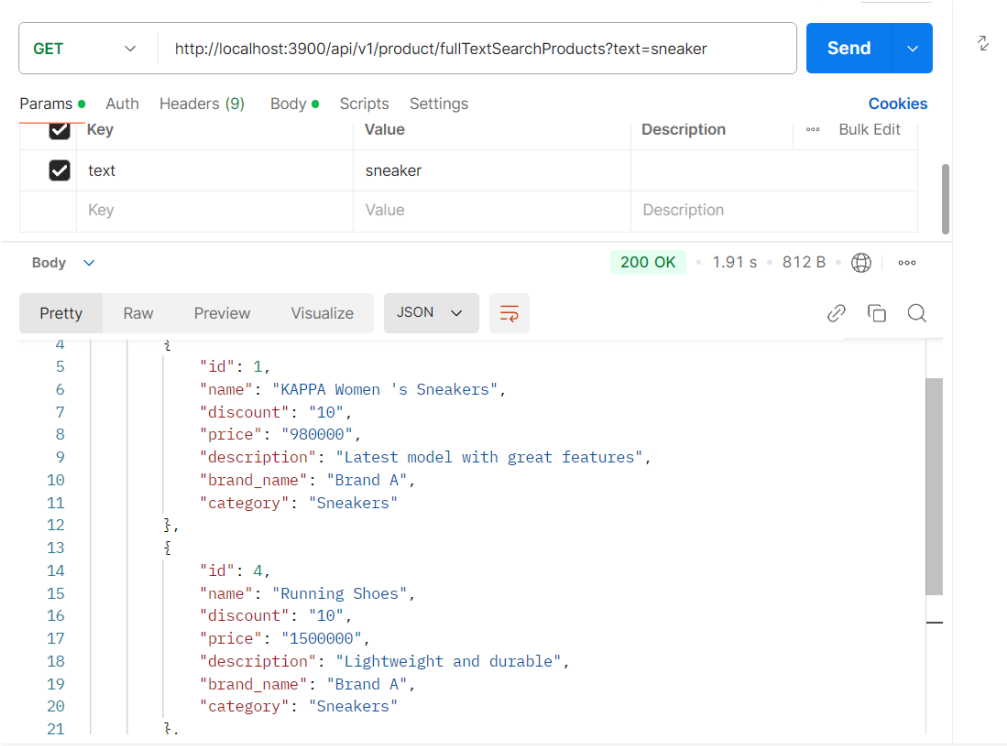
Hình 10: Get Products by category

2.5.3 API: `Tìm kiếm toàn văn cho sản phẩm`

- **HTTP Method:** GET
- **Endpoint:** `http://localhost:3900/api/v1/product/fullTextSearch?text=:text`
- **Mô tả:** API này nhận tham số `text` từ yêu cầu HTTP và trả về danh sách sản phẩm khớp với nội dung tìm kiếm.
- **Tham số:**
  - `text` (**bắt buộc**): Chuỗi văn bản cần tìm kiếm.
- **Quy trình xử lý:**

1. Kiểm tra sự tồn tại của tham số `text`.
   - Nếu không có tham số `text`, trả về mã lỗi 400 cùng thông báo: *"Search text is required"*.
2. Gọi phương thức `productModel.fullTextSearchProduct(text)` để thực hiện truy vấn.
   - Nếu truy vấn thành công, trả về mã thành công 200 cùng dữ liệu sản phẩm.
   - Nếu truy vấn thất bại, trả về mã lỗi 400 cùng thông báo lỗi chi tiết.
3. Xử lý lỗi nếu xảy ra trong quá trình thực thi:
   - Trả về mã lỗi 500 cùng thông báo: *"Unable to perform full-text search"*.

- **Mã trả về:**
  - `200 OK`: Tìm kiếm thành công, trả về danh sách sản phẩm.
  - `400 Bad Request`: Không có tham số `text` hoặc lỗi từ mô hình.
  - `500 Internal Server Error`: Lỗi không mong muốn trong hệ thống.



Hình 11: Full Text Search

2.5.4 API: `Tạo một order mới`

- **PHương thức**: POST
- **Endpoint:** `http://localhost:3900/api/v1/order/create`
- **Tham số:**

  - **Body của Request**: Các thông tin mà client gửi lên API.
  - Các thông số trong `req.body`:
    * **guest**: Thông tin khách hàng.
      · `name`: Tên khách hàng.
      · `phone`: Số điện thoại của khách hàng.
      · `gender`: Giới tính của khách hàng.
      · `house_number`: Số nhà của khách hàng.
      · `district`: Quận/huyện của khách hàng.
      · `commune`: Xã/phường của khách hàng.
      · `province`: Tỉnh/thành phố của khách hàng.
      · `housing_type`: Loại nhà của khách hàng (có thể là nhà ở, căn hộ, v.v.).
      · `email`: Địa chỉ email của khách hàng.
    * **order**: Thông tin về đơn hàng.
      · `totalPrice`: Tổng giá trị đơn hàng.
    * **orderItems**: Danh sách các sản phẩm trong đơn hàng.
      · `itemId`: ID của sản phẩm.
      · `quantity`: Số lượng của sản phẩm.
      · `price`: Giá của sản phẩm.

- **Thêm thông tin người mua (Buyer)**:
  - Dữ liệu người mua được lấy từ `req.body.guest`.
  - Phương thức `BuyerModel.addBuyer` được gọi để thêm người mua vào cơ sở dữ liệu.
  - Nếu việc thêm người mua thất bại (trả về `buyer.success` là `false`), API sẽ trả về lỗi với mã 500.

- **Thêm thông tin khách (Guest)**:
  - Sau khi tạo người mua thành công, thông tin khách (email) sẽ được thêm vào cơ sở dữ liệu bằng phương thức `BuyerModel.addGuest`.
  - Liên kết khách hàng với `buyerId` từ bước trước.
  - Nếu việc thêm khách thất bại, API sẽ trả về lỗi với mã 500.

- **Thêm thông tin đơn hàng (Order)**:
  - Sau khi khách hàng và người mua đã được thêm vào, đơn hàng sẽ được tạo thông qua phương thức `OrderModel.createOrder`, sử dụng giá trị `order.totalPrice` và `buyerId`.
  - Nếu đơn hàng không thể được tạo, API sẽ trả về lỗi với mã 500.

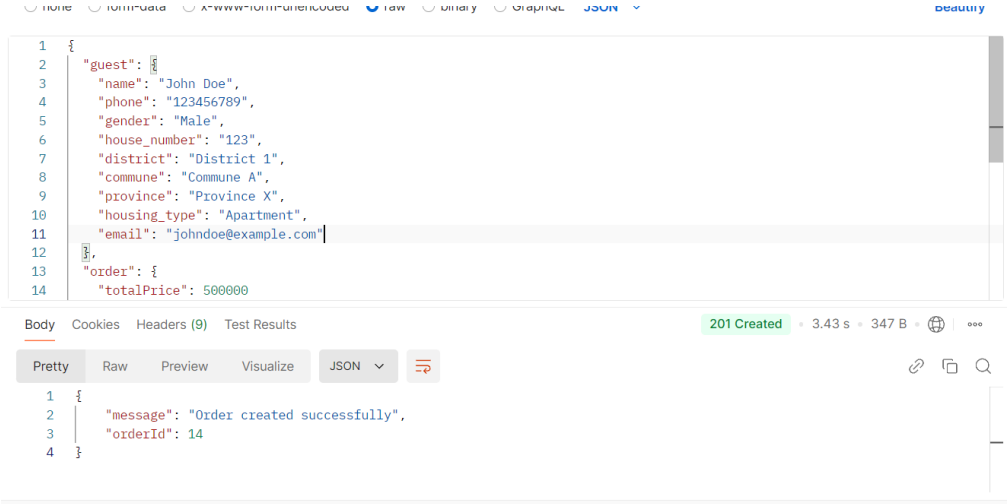
- **Thêm các sản phẩm vào đơn hàng (Order Items)**:
  - Đối với mỗi sản phẩm trong `orderItems`, API sẽ gọi phương thức `OrderModel.createOrderIte`
    để thêm sản phẩm vào đơn hàng vừa tạo.

  - Nếu có lỗi trong việc thêm sản phẩm, API sẽ trả về mã lỗi và thông báo lỗi tương ứng.

- **Gửi email xác nhận đơn hàng**:
  - Sau khi đơn hàng được tạo và các sản phẩm đã được thêm vào, API sẽ gửi email xác nhận đến khách hàng.
  - Các thông tin gửi email bao gồm:
    * `To`: Email của khách hàng.
    * `Subject`: Tiêu đề email chứa ID đơn hàng.
    * `Text Content`: Nội dung email xác nhận đơn hàng với ID và tổng số tiền.
    * `HTML Content`: Nội dung email xác nhận đơn hàng dưới dạng HTML.

- **Phản hồi API**:
  - Nếu tất cả các bước trước đều thành công, API sẽ trả về mã 201 (Created) cùng với thông báo thành công và `orderId` của đơn hàng.

- **Xử lý lỗi**:
  - Trong trường hợp có bất kỳ lỗi nào xảy ra trong quá trình thực thi (từ bước thêm người mua, khách, đơn hàng, sản phẩm, cho đến gửi email), API sẽ bắt lỗi trong khối `catch` và trả về mã lỗi 500 cùng với thông báo lỗi chi tiết.

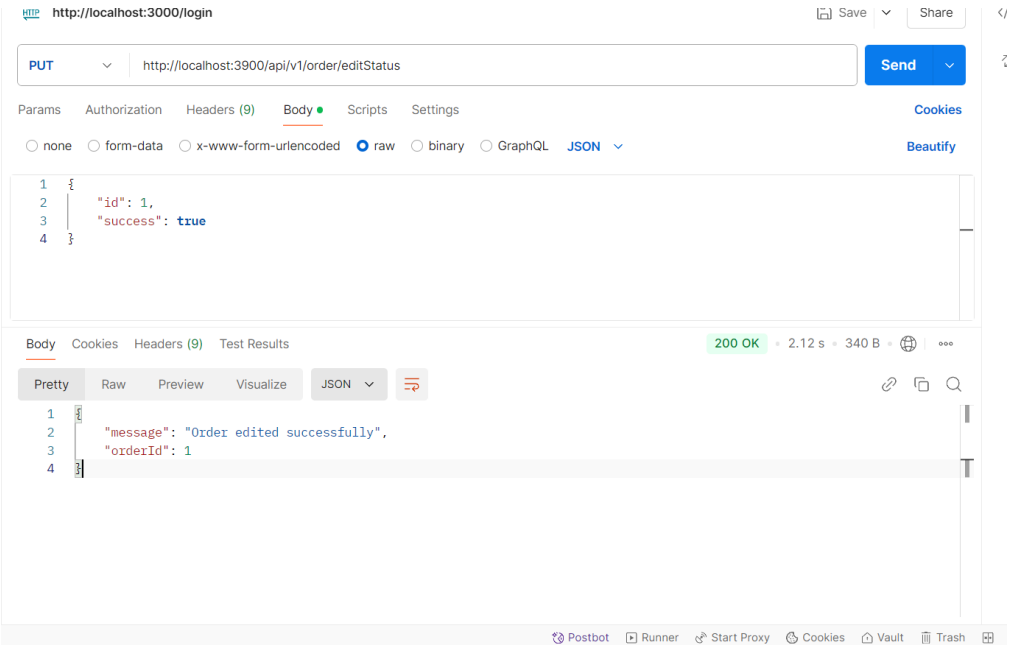


Hình 12: Create Order

2.5.5 API: `api/v1/order/editStatus`

- **HTTP Method:** PUT
- **Endpoint:** `http://localhost:3900/api/v1/order/editStatus`
- **Mô tả:** API này nhận tham số `id` và `success` từ yêu cầu HTTP và cập nhật trạng thái đơn hàng tương ứng. Dùng để xác nhận thanh toán thành công cho shipper khi giao hàng.

- **Tham số:**
  - `id` (**bắt buộc**): ID của đơn hàng cần cập nhật trạng thái.
  - `success` (**bắt buộc**): Trạng thái mới của đơn hàng (true/false).
- **Quy trình xử lý:**
  1. Kiểm tra sự tồn tại của tham số `id` và `success`.
  2. Gọi phương thức `OrderModel.editOrderStatus(id, success)` để cập nhật trạng thái đơn hàng.
     - Nếu truy vấn thành công, trả về mã thành công 200 cùng thông báo và `orderId`.
     - Nếu thất bại, trả về mã lỗi 400 với thông báo lỗi chi tiết.
  3. Xử lý lỗi nếu xảy ra trong quá trình thực thi:
     - Trả về mã lỗi 500 cùng thông báo: *"Failed to edit order"*.
- **Mã trả về:**
  - `200 OK`: Cập nhật trạng thái đơn hàng thành công.
  - `400 Bad Request`: Thất bại trong việc cập nhật trạng thái (ví dụ: không tìm thấy đơn hàng).
  - `500 Internal Server Error`: Lỗi không mong muốn trong hệ thống.



Hình 13: Edit Order