

HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



FINAL PROJECT REPORT

IT3290 – DATABASE LAB

Course information

Course ID

IT3290

Course title

DATABASE LAB

Class ID

147775

Student information

Student's full name	Class	Student ID
Mai Minh Quân	Việt Nhật 03	20225661
Vũ Minh Quân	Việt Nhật 04	20225910
Nguyễn Mạnh Hùng	Việt Nhật 03	20225630

Instructor: MSc. Vũ Tuyết Trinh

Mục lục

HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY	1
I. Mô tả bài toán	4
II. Thiết kế cơ sở dữ liệu	4
1. Sơ đồ thực thể liên kết.....	4
2. Thực thể liên kết.....	4
a. Tài khoản (accounts)	4
b. Khu vực (areas)	5
c. Bàn ăn (tables)	5
d. Món ăn (foods)	5
e. Combo (combos).....	6
f. Mặt hàng (items)	6
g. Hóa đơn (bills).....	7
3. Quan hệ.....	8
a. Món ăn – combo	8
b. Đặt món	8
III. Các chức năng của cơ sở dữ liệu	8
1. Tạo tài khoản cho khách hàng	8
2. Đặt bàn.....	9
3. Quản lý các bàn ăn	9
4. Quản lý thực đơn, món ăn	9
5. Gọi món.....	10
6. Thanh toán	10
7. Đánh giá	11
IV. Demo các chức năng của hệ CSDL	11
1. Các hàm FUNCTIONS	11
a. Hàm tạo tài khoản	11
b. Hàm đặt trước bàn	13
c. Hàm cập nhật time_check_in cho khách hàng đã đặt trước.....	13
d. Hàm tạo hóa đơn cho khách hàng chưa đặt trước.....	15
e. Hàm kiểm tra khách hàng có đến đúng giờ hay không cho khách hàng đã đặt trước	16
f. Hàm gọi món.....	18
g. Hàm tính hóa đơn gốc	20
h. Hàm sử dụng discount_point.....	22
i. Hàm gửi đánh giá của khách hàng	24
j. Hàm xác nhận hoàn thành thanh toán (bill_status chuyển từ 0 → 1)	25
2. Các TRIGGERS	26
a. Trigger khi insert food_id to items	26
b. Trigger khi delete food_id to items	26
c. Trigger khi insert combo_id to items	27
d. Trigger khi delete combo_id to items	27

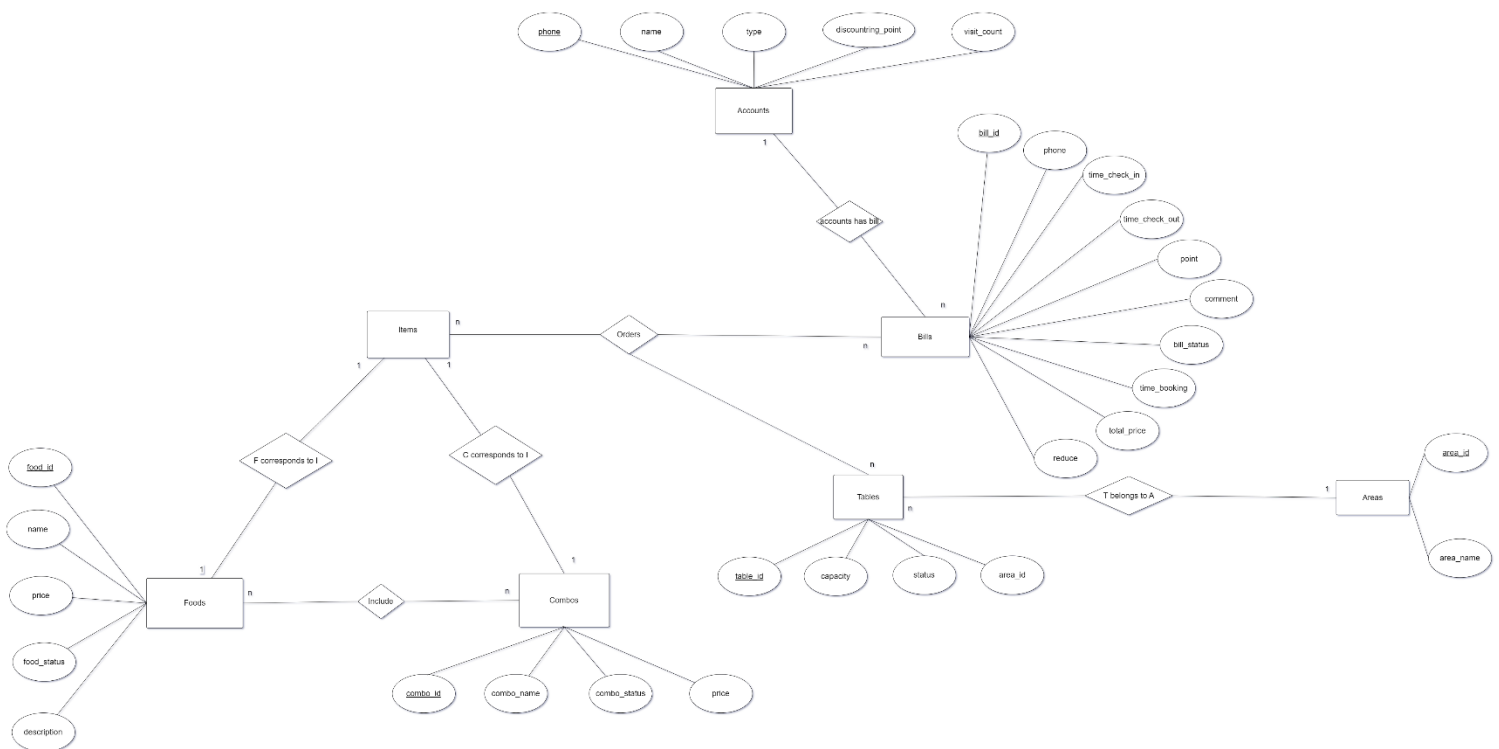
e.	Trigger khi update food_status to items	28
f.	Trigger khi update combo_status to items	28
g.	Trigger thay đổi table_status khi update orders	29
h.	Trigger tính điểm dựa vào visit_count.....	29
i.	Trigger thi thanh toán hóa đơn	31
3.	Các câu truy vấn.....	32
V.	Tối ưu câu truy vấn bằng INDEX	38
1.	Tạo index cho thuộc tính bill_id trong bảng orders.....	38
2.	Tạo index cho thuộc tính total_price trong bảng bills	40
VI.	Kết luận	41

I. Mô tả bài toán

Hệ thống đặt món ăn trong nhà hàng được thiết kế nhằm cung cấp một giải pháp toàn diện cho các quy trình từ đặt bàn, gọi món, thanh toán đến đánh giá chất lượng nhà hàng. Hệ thống sẽ giúp cải thiện hiệu quả hoạt động của nhà hàng, nâng cao trải nghiệm của khách hàng và cung cấp các dữ liệu phân tích hữu ích cho nhà quản lý.

II. Thiết kế cơ sở dữ liệu

1. Sơ đồ thực thể liên kết




2. Thực thể liên kết

a. Tài khoản (accounts)

- Mô tả: Tạo tài khoản (nếu chưa có), đặt bàn và thanh toán và tích điểm thưởng
- Thuộc tính

- *phone (prKey)*: số điện thoại khách hàng
- *name*: tên khách hàng
- *type*: loại tài khoản
- *discount_point*: điểm tích lũy
- *visit_count*: số lần đến nhà hàng

accounts	
phone 	int
name	varchar
type	varchar
discount_point	float
visit_count	int



b. Khu vực (areas)

- Mô tả: chia nhà hàng thành 2 khu vực là vip và regular
- Thuộc tính:
 - *area_id (prKey)*: id khu vực
 - *area_name*: tên khu vực

areas	
area_id 	int
area_name	varchar

c. Bàn ăn (tables)

- Mô tả: lưu trữ các thông tin liên quan đến trạng thái của bàn, sức chứa và khu vực hiện tại của bàn
- Thuộc tính:
 - *table_id (prKey)*: id của bàn ăn
 - *capacity*: sức chứa của bàn
 - *status*: trạng thái hiện tại của bàn (có người, còn trống)
 - *area_id*: id của khu vực chứa bàn ăn

tables 	
table_id 	int
capacity	int
status	int
area_id	int

d. Món ăn (foods)

- Mô tả: lưu trữ các thông tin liên quan đến các món ăn mà nhà hàng phục vụ

- Thuộc tính:

- *food_id* (*prKey*): id của món ăn
- *name*: tên của món ăn
- *price*: giá của món ăn
- *food_status*: trạng thái của món ăn (sẵn sàng phục vụ, hết hàng)
- *description*: mô tả món ăn

foods	
food_id 🔗	varchar
name	varchar
price	int
food_status	varchar
description	varchar

e. Combo (combos)

- Mô tả: lưu trữ các thông tin liên quan đến các combo mà nhà hàng phục vụ

- Thuộc tính:

- *combo_id* (*prKey*): id của combo
- *combo_name*: tên của combo
- *combo_status*: trạng thái của combo (sẵn sàng phục vụ, hết hàng)
- *price*: giá của combo

combos	
combo_id 🔗	varchar
combo_name	varchar
combo_status	varchar
price	int

f. Mặt hàng (items)

- Mô tả: lưu trữ id của các món ăn và combo nhà hàng phục vụ

- Thuộc tính:

- *item_id*: mã của các món ăn và combo trong menu mà nhà hàng đang phục vụ
- *food_id*: id của các món ăn trong nhà hàng
- *combo_id*: id của các combo trong nhà hàng

items	
item_id 🔗	varchar
food_id	varchar
combo_id	varchar

g. Hóa đơn (bills)

- Mô tả: lưu trữ các thông tin liên quan đến hóa đơn của khách hàng
- Thuộc tính:
 - *bill_id* (*prKey*): id của hóa đơn
 - *phone*: số điện thoại của khách hàng
 - *number_of_guest*: số lượng khách hàng
 - *time_check_in*: thời gian check_in của khách hàng
 - *time_check_out*: thời gian check_out của khách hàng
 - *point*: điểm đánh giá của khách hàng
 - *comment*: nhận xét của khách hàng
 - *bill_status*: trạng thái của hóa đơn (chưa thanh toán, đã thanh toán)
 - *time_booking*: thời gian khách hàng đặt trước bàn
 - *reduce*: lượng điểm mà khách hàng sử dụng để giảm vào giá trị hóa đơn
 - *total_price*: giá trị hóa đơn.

bills	
bill_id 🔗	int
phone	varchar
number_of_guest	int
time_check_in	timestamp
time_check_out	timestamp
point	int
comment	text
bill_status	int
time_booking	timestamp
reduce	int
total_price	float

3. Quan hệ

a. Món ăn – combo

- Mô tả: liên kết các món ăn có trong nhà hàng để tạo thành combo
- Thuộc tính:
 - o *food_id*: mã id của các món ăn
 - o *combo_id*: mã id của các combo

join_foods_combos	
food_id	varchar
combo_id	varchar

b. Đặt món

- Mô tả: xác định các món ăn, combo và số lượng mà khách hàng đặt trong hóa đơn
- Thuộc tính:
 - o *order_id* (*prKey*):
 - o *bill_id*: mã id hóa đơn của khách hàng
 - o *item_id*: mã id của món ăn (combo)
 - o *table_id*: id bàn ăn mà khách hàng gọi món
 - o *quantity*: số lượng các món ăn (combo) mà khách hàng gọi

orders	
order_id 🔗	int
bill_id	int
item_id	int
table_id	int
quantity	int

III. Các chức năng của cơ sở dữ liệu

1. Tạo tài khoản cho khách hàng

- Mô tả: hệ CSDL sẽ tạo tài khoản cho khách hàng dựa trên tên và số điện thoại mà khách hàng cung cấp
- Quy trình:
 - o Nếu số điện thoại mà khách hàng cung cấp chưa tồn tại trong CSDL thì hệ thống sẽ tạo ra một tài khoản mới với *discount_point* và *visit_count* có giá trị mặc định bằng 0 và trả về thông báo “Tao tai khoan thanh cong!”
 - o Nếu số điện thoại mà khách hàng cung cấp đã tồn tại trong CSDL thì hệ thống sẽ không tạo tài khoản mới và trả về thông báo “Tai khoan da ton tai trong he thong!”

2. Đặt bàn

- Mô tả: tạo hóa đơn khi khách hàng đặt bàn trước với các thuộc tính phone, number_of_guest, time_booking, bill_status
- Quy trình:
 - Khi đặt bàn trước, khách hàng phải cung cấp các thông tin như số điện thoại, số lượng người, thời gian dự định đến quán.
 - Hệ thống sẽ kiểm tra số điện thoại mà khách hàng cung cấp đã được dùng để tạo tài khoản chưa. Nếu chưa tồn tại thì sẽ tạo tài khoản mới cho khách hàng và liên kết hóa đơn với tài khoản bằng thuộc tính phone.
 - Nếu khách hàng đến quán mà chưa đặt trước, hệ thống sẽ tạo ra hóa đơn dựa trên số điện thoại và thời gian check in của khách hàng

3. Quản lý các bàn ăn

- Mô tả:
 - Khách hàng chưa đặt bàn trước sẽ chọn bàn trên hệ thống. Hệ thống sẽ hiện ra những thông tin về bàn như: bàn còn trống không? bàn dành cho bao nhiêu người? Bàn trong khu vip hay regular?
 - Nhà hàng sẽ được chia thành 2 khu vực: khu vực vip và regular
 - Trong đó, mỗi bàn trong cơ sở dữ liệu sẽ lưu trữ các thông tin:
 - Khu vực
 - Sức chứa
 - Trạng thái (có người, trống)
 - Khách hàng có thể đặt nhiều bàn cùng một lúc để phục vụ nhóm đông người.
 - Khi khách hàng bắt đầu gọi món, bàn của khách hàng đang ngồi sẽ chuyển sang trạng thái có người.
 - Sau khi thanh toán hóa đơn bàn của khách hàng sẽ chuyển sang trạng thái trống.

4. Quản lý thực đơn, món ăn

- Mô tả:
 - Dựa vào hệ CSDL, ta có thể biết được các thông tin cơ bản của món ăn, combo như tên, giá tiền, mô tả, có sẵn hay hết hàng.
 - Hệ thống thực đơn trong CSDL sẽ tổng hợp các món ăn, combo mà nhà hàng đang sẵn sàng phục vụ.

- Khách hàng có thể gọi theo những combo khuyến mãi của nhà hàng hoặc có thể gọi món riêng lẻ theo các hạng mục: món khai vị, món chính, món tráng miệng, đồ uống hoặc kết hợp cả 2 hình thức trên.
- Các món ăn của nhà hàng sẽ được quản lý theo món riêng lẻ và các combo.
 - Đối với các món ăn riêng lẻ, khách hàng có thể xem qua những thông tin cơ bản của món ăn như: tên món ăn, mô tả món ăn, giá tiền, trạng thái (hết hàng, sẵn sàng phục vụ, ...).
 - Đối với combo, khách hàng có thể xem tên combo, các món ăn bao gồm trong combo, giá tiền, trạng thái (hết hàng, sẵn sàng phục vụ, ...).

5. Gọi món

- Mô tả:

- Dựa vào thực đơn trong CSDL, khách hàng có thể chọn các món ăn, combo, trong đó bảng “orders” sẽ lưu các thông tin như mã id của món ăn, combo, số lượng, mã id bàn ăn của khách hàng. Sau đó, tất cả “orders” sẽ được liên kết với hóa đơn của khách hàng để có thể tính tổng giá trị hóa đơn
- Trong trường hợp, khách ngồi nhiều bàn, thông tin gọi món vẫn chỉ lưu vào 1 hóa đơn duy nhất ứng với số điện thoại của người gọi món.

6. Thanh toán

- Mô tả:

- Khi thanh toán, hệ thống sẽ tính tổng giá trị gốc của hóa đơn. Tiếp theo, khách hàng có thể lựa chọn sử dụng điểm tích lũy trong tài khoản của mình, và chọn lượng điểm để giảm vào hóa đơn (lượng điểm đó cũng sẽ trừ vào tài khoản của khách hàng).
- Sau đó, hệ thống sẽ kiểm tra xem khách hàng có đặt trước bàn và đến đúng hẹn không? Nếu có sẽ trừ đi 1% giá trị hóa đơn.
- Cuối cùng, hệ thống sẽ tính giá trị hóa đơn sau khi khách hàng sử dụng các hình thức giảm giá.
- Khi thanh toán thành công, tài khoản của khách hàng sẽ được cộng 2% giá trị hóa đơn cuối cùng. Đồng thời, bàn của khách hàng sẽ chuyển sang trạng thái “trống”.
- Đồng thời tăng số lần đến nhà hàng của khách hàng thêm 1.

- Khi số lần đến cửa hàng của khách hàng (N) đạt đến mức nhất định sẽ được cộng điểm vào discount_point trong tài khoản theo cơ chế sau:
 - $N = 5$ thì cộng thêm 4 điểm vào discount_point
 - $N > 5$ và $N \% 5 = 0$ thì công thức cộng điểm là: $\left(\frac{N}{5} - 1\right) * 10$

7. Đánh giá

- Mô tả:
 - Khách hàng có thể đánh giá trên thang 5 sao, bình luận về chất lượng dịch vụ của nhà hàng (giúp cho nhà hàng nhận những đóng góp của khách hàng để cải thiện chất lượng món ăn).

IV. Demo các chức năng của hệ CSDL

1. Các hàm FUNCTIONS

a. Hàm tạo tài khoản

- Mã chương trình

```
-- Hàm tạo tài khoản
CREATE OR REPLACE FUNCTION F_Create_Account(p_phone VARCHAR, p_name VARCHAR)
RETURNS VARCHAR AS $$
DECLARE
    v_username VARCHAR;
BEGIN
    SELECT name INTO v_username
    FROM accounts WHERE phone = p_phone;

    IF v_username IS NULL THEN
        BEGIN
            INSERT INTO accounts (phone, name, type, discount_point, visit_count)
            VALUES (p_phone, p_name, 'customer' , 0, 0);
            RETURN 'tao tai khoan thanh cong';
        EXCEPTION WHEN OTHERS THEN
            RETURN 'co loi xay ra khi tao tai khoan';
        END;
    ELSE
        RETURN 'tai khoan voi sdt nay da ton tai';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

- Kết quả:

```
11 SELECT f_create_account('0123456789', 'Mai Minh Quan');
```

Data Output Messages Notifications



	f_create_account character varying	🔒
1	Tao tai khoan thanh cong!	

```
11 SELECT f_create_account('0123456789', 'Mai Minh Quan');  
12 SELECT * FROM accounts
```

Data Output Messages Notifications



	phone [PK] character varying	name character varying	type character varying	discount_point double precision	visit_count integer
1	0123456789	Mai Minh Quan	customer	0	0

b. Hàm đặt trước bàn

- Mã chương trình

```
-- Hàm đặt trước bàn
CREATE OR REPLACE FUNCTION F_Create_Booking(
    p_phone VARCHAR,
    p_time_booking timestamp,
    p_number_of_guest int
)
RETURNS VARCHAR AS $$
BEGIN
    -- kiểm tra sdt hợp lệ
    IF NOT EXISTS(SELECT 1 FROM accounts WHERE phone = p_phone) THEN
        RETURN 'sdt ko ton tai';
    END IF;

    INSERT INTO bills(phone, time_booking, bill_status, number_of_guest)
    VALUES (p_phone, p_time_booking, 0, p_number_of_guest);

    RETURN 'tao booking thanh cong';
END;
$$ LANGUAGE plpgsql;
```

- Kết quả

```
SELECT F_Create_Booking('0123456789', '2024-06-11 18:00:00', 3);
SELECT * FROM bills
```

a Output Messages Notifications

bill_id	time_check_in	time_check_out	point	comment	bill_status	time_booking	total_price	phone	reduce	number_of_g
[PK] integer	timestamp with time zone	timestamp without time zone	integer	text	integer	timestamp without time zone	double precision	character varying(20)	double precision	integer
3	[null]	[null]	[null]	[null]	0	2024-06-11 18:00:00	[null]	0123456789	[null]	3

c. Hàm cập nhật time_check_in cho khách hàng đã đặt trước

- Mã chương trình

```

-- Hàm update time check in cho khách hàng đã đặt trước
CREATE OR REPLACE FUNCTION F_Update_Time_Check_In(p_phone varchar, p_bill_id INT)
RETURNS VARCHAR AS $$
DECLARE
BEGIN
    UPDATE bills
    SET time_check_in = date_trunc('second',CURRENT_TIMESTAMP)
    WHERE phone = p_phone
        AND time_booking IS NOT NULL
        AND time_check_in IS NULL
        AND p_bill_id = bill_id;
    RETURN 'da cap nhat time_check_in THANH CONG!';
END;
$$ LANGUAGE plpgsql;

```

- Kết quả

```

2 select f_update_time_check_in('0123456789', 11);
3 select * from bills;

```

Data Output Messages Notifications



	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number_of_guest integer
1	11	2024-06-11 21:44:57	[null]	[null]	[null]	0	2024-06-11 21:45:00	[null]	0123456789	[null]	

d. Hàm tạo hóa đơn cho khách hàng chưa đặt trước

- Mã chương trình

```
-- Hàm tạo hóa đơn cho khách hàng chưa đặt trước
CREATE OR REPLACE FUNCTION F_Create_Bill(
p_phone VARCHAR,
p_number_of_guest INT)
RETURNS VARCHAR AS $$
BEGIN
    -- kiểm tra số điện thoại hợp lệ
    IF NOT EXISTS(SELECT 1 FROM accounts WHERE phone = p_phone) THEN
        RETURN 'sdt ko ton tai';
    END IF;

    -- Thêm một bản ghi mới vào bảng bills với thời gian hiện tại đã loại bỏ phần mili giây
    INSERT INTO bills (phone, time_check_in, bill_status, number_of_guest)
    VALUES (p_phone, date_trunc('second', clock_timestamp()), 0, p_number_of_guest);

    RETURN 'tao bill thanh cong';
END;
$$ LANGUAGE plpgsql;
```

- Kết quả

```
1 select f_create_bill('0123456789', 2);
2 select * from bills;
```

Data Output Messages Notifications



	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number_of_guest integer
1	9	2024-06-11 21:22:32	[null]	[null]	[null]	0	[null]	[null]	0123456789	[null]	

e. Hàm kiểm tra khách hàng có đến đúng giờ hay không cho khách hàng đã đặt trước

- Mã chương trình

```
CREATE OR REPLACE FUNCTION F_On_Time(p_bill_id INT)
RETURNS INT AS $$
DECLARE
    v_time_booking TIMESTAMP;
    v_time_check_in TIMESTAMP;
BEGIN
    SELECT time_booking, time_check_in
    INTO v_time_booking, v_time_check_in
    FROM bills
    WHERE bill_id = p_bill_id;

    IF v_time_booking IS NULL THEN
        RETURN 0;
    ELSIF ABS (EXTRACT(EPOCH FROM (v_time_check_in - v_time_booking)) / 60) <= 15 THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

- Kết quả:

o Thông tin hóa đơn:

```
1 -- SELECT F_Create_Booking('0123456789', '2024-06-11 21:45:00', 3);
2 select f_update_time_check_in('0123456789', 11);
3 select * from bills;
4
```

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number_ integer
1	11	2024-06-11 21:44:57	[null]	[null]	[null]	0	2024-06-11 21:45:00	[null]	0123456789	[null]	[null]

o Kết quả trả về 1 khi time_check_in sớm hơn hoặc muộn không quá 15 phút so với time_booking


```
1  -- SELECT F_Create_Booking('0123456789', '2024-06-11 21:45:00', 3);
2  select f_update_time_check_in('0123456789', 11);
3  select * from bills;
4
5  select f_on_time(11);
```

Data Output Messages Notifications



	f_on_time integer	🔒
1		1

f. Hàm gọi món

- Mã chương trình

```
CREATE OR REPLACE FUNCTION F_InsertOrder (  
    IN p_bill_id INT,  
    IN p_item_id VARCHAR,  
    IN p_table_id INT,  
    IN p_quantity INT  
)  
RETURNS VOID AS $$  
DECLARE  
    v_order_id INT;  
    v_item_count INT;  
    v_new_count INT;  
BEGIN  
    -- Initialize variables  
    v_order_id := 0;  
    v_item_count := 0;  
    v_new_count := 0;  
  
    -- Check if the order already exists in BillInfo  
    SELECT order_id, quantity  
    INTO v_order_id, v_item_count  
    FROM orders  
    WHERE bill_id = p_bill_id AND item_id = p_item_id;
```

```

-- If order exists, update quantity
IF v_order_id > 0 THEN
    v_new_count := v_item_count + p_quantity;
    IF v_new_count > 0 THEN
        -- Update the existing order
        UPDATE orders
        SET quantity = v_new_count
        WHERE order_id = v_order_id;
    END IF;
ELSE
    -- Otherwise, insert a new order
    INSERT INTO orders (bill_id, item_id, table_id, quantity)
    VALUES (p_bill_id, p_item_id, p_table_id, p_quantity);
END IF;
END;
$$ LANGUAGE plpgsql;

```

- Kết quả:

1

select f_insertorder(9, 'F1', 1, 3);

2

select * from orders;

3

Data Output

Messages

Notifications

☰

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	order_id [PK] integer	bill_id integer	item_id character varying	table_id integer	quantity integer
1	4	9	F1	1	6

g. Hàm tính hóa đơn gốc

- Mã chương trình

```
CREATE OR REPLACE FUNCTION F_BillSum(p_bill_id INT)
RETURNS VOID AS $$
DECLARE
    v_total_sum_food INT := 0;
    v_total_sum_combo INT := 0;
BEGIN
    -- Tính tổng giá trị các món ăn
    SELECT COALESCE(SUM(f.price * o.quantity), 0)
    INTO v_total_sum_food
    FROM orders AS o
    JOIN foods AS f ON f.food_id = o.item_id
    WHERE o.bill_id = p_bill_id;

    -- Tính tổng giá trị các combo
    SELECT COALESCE(SUM(c.price * o.quantity), 0)
    INTO v_total_sum_combo
    FROM orders AS o
    JOIN combos AS c ON c.combo_id = o.item_id
    WHERE o.bill_id = p_bill_id;

    -- Trả về tổng giá trị
    update bills set total_price = v_total_sum_food + v_total_sum_combo
    where bill_id = p_bill_id ;
END;|
$$ LANGUAGE plpgsql;
```

- Kết quả:
 - o Thông tin các món ăn được order:

```

1 select f_insertorder(11, 'F1', 1, 3); -- F1: 14$
2 select f_insertorder(11, 'F2', 1, 1); -- F2: 12$
3 select f_insertorder(11, 'C2', 1, 1); -- C2: 65$
4 select * from orders;
5
6
7

```

Data Output Messages Notifications

	order_id [PK] integer	bill_id integer	item_id character varying	table_id integer	quantity integer
1	15	11	F1	1	3
2	16	11	F2	1	1
3	17	11	C2	1	1

○ Tính tiền hóa đơn gốc

```

6 select f_billsum(11); -- 14 * 3 + 12 * 1 + 65 * 1 = 119
7 select * from bills;
8

```

Data Output Messages Notifications

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number_ integer
1	11	2024-06-11 21:44:57	[null]	[null]	[null]	0	2024-06-11 21:45:00	119	0123456789	[null]	

h. Hàm sử dụng discount_point

- Mã chương trình

```
CREATE OR REPLACE FUNCTION F_UsingDiscountpoint(
    IN p_bill_id INT,
    IN p_reduce INT
)
RETURNS VOID AS $$
DECLARE
    v_user_point INT;
    v_total_price INT;
    v_phone VARCHAR;
BEGIN
    -- Lấy điểm giảm giá hiện tại của khách hàng
    SELECT phone INTO v_phone
    FROM bills WHERE bill_id = p_bill_id;

    SELECT discount_point INTO v_user_point
    FROM accounts
    WHERE phone = v_phone;

    SELECT total_price INTO v_total_price
    FROM bills
    WHERE bill_id = p_bill_id;

    -- Kiểm tra điều kiện để sử dụng điểm giảm giá
    IF p_reduce > 0 AND p_reduce % 5 = 0 AND p_reduce <= v_user_point AND v_total_price >= p_reduce THEN
        -- Thực hiện thêm dòng vào bảng Pays
        UPDATE bills SET reduce = p_reduce WHERE bill_id = p_bill_id;

        -- Cập nhật điểm giảm giá của khách hàng
        UPDATE accounts
        SET discount_point = v_user_point - p_reduce
        WHERE phone = v_phone;

        UPDATE bills
        SET total_price = v_total_price - p_reduce
        WHERE bill_id = p_bill_id;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

- Kết quả:

o Điểm discount ban đầu của khách:

	phone [PK] character varying	name character varying	type character varying	discount_point double precision	visit_count integer
1	0123456789	mai minh quan	customer	20	

o Giá trị hóa đơn gốc ban đầu

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number integer
1	11	2024-06-11 21:44:57	[null]	[null]	[null]	0	2024-06-11 21:45:00	119	0123456789	[null]	

- Sau khi sử dụng 20 điểm discount:

1 select f_usingdiscountpoint(11, 20);

2 select * from bills;

Data Output

Messages

Notifications

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number_c integer
1	11	2024-06-11 21:44:57	[null]	[null]	[null]	0	2024-06-11 21:45:00	99	0123456789	20	

1
2
3
4

```
select f_usingdiscountpoint(11, 20);
select * from bills;
select * from accounts;
```

Data Output

Messages

Notifications

	phone [PK] character varying	name character varying	type character varying	discount_point double precision	visit_count integer
1	0123456789	mai minh quan	customer	0	0

i. Hàm gửi đánh giá của khách hàng

- Mã chương trình

```
CREATE OR REPLACE FUNCTION submit_review(  
  p_bill_id INT,  
  p_point INT,  
  p_comment varchar  
)  
RETURNS VOID AS $$  
BEGIN  
  IF p_point < 1 OR p_point > 5 THEN  
    RAISE EXCEPTION 'diem danh gia phai tu 1 den 5';  
  END IF;  
  
  UPDATE bills  
  SET point = p_point, comment = p_comment  
  WHERE bill_id = p_bill_id;  
END;  
$$ LANGUAGE plpgsql;
```

- Kết quả:

o Bill trước khi gửi đánh giá:

1

select * from bills

Data Output

Messages

Notifications

	zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varying	reduce double precision	number_o integer
1		[null]	[null]	[null]	0	2024-06-11 21:45:00	99	0123456789	20	

o Bill sau khi gửi đánh giá

1

select submit_review(11, 4, 'quan phuc vu nhanh');

2

select * from bills;

Data Output

Messages

Notifications

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character varyin
1	11	2024-06-11 21:44:57	[null]	4	quan phuc vu nhanh	0	2024-06-11 21:45:00	99	0123456789

j. Hàm xác nhận hoàn thành thanh toán (bill_status chuyển từ 0 → 1)

- Mã chương trình

```
CREATE OR REPLACE FUNCTION submit_review(  
    p_bill_id INT,  
    p_point INT,  
    p_comment varchar  
)  
RETURNS VOID AS $$  
BEGIN  
    IF p_point < 1 OR p_point > 5 THEN  
        RAISE EXCEPTION 'diem danh gia phai tu 1 den 5';  
    END IF;  
  
    UPDATE bills  
    SET point = p_point, comment = p_comment  
    WHERE bill_id = p_bill_id;  
END;  
$$ LANGUAGE plpgsql;
```

- Kết quả:

o Bill trước khi xác nhận:

1

select * from bills;

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character
1	11	2024-06-11 21:44:57	[null]	4	quan phuc vu nhanh	0	2024-06-11 21:45:00	99	01234567

o Bill sau khi gửi đánh giá:

1

select f_purchase(11);

2

select * from bills;

	bill_id [PK] integer	time_check_in timestamp without time zone	time_check_out timestamp without time zone	point integer	comment text	bill_status integer	time_booking timestamp without time zone	total_price double precision	phone character
1	11	2024-06-11 21:44:57	[null]	4	quan phuc vu nhanh	1	2024-06-11 21:45:00	98.01	012345678

o Điểm discount của khách được cộng 2% hóa đơn cuối đồng thời tăng visit_count lên 1:

```

1 select f_purchase(11);
2 select * from bills;
3 select * from accounts;

```

Data Output Messages Notifications

	phone [PK] character varying	name character varying	type character varying	discount_point double precision	visit_count integer
1	0123456789	mai minh quan	customer	19.6020000000000004	1

2. Các TRIGGERS

a. Trigger khi insert food_id to items

```
-- Trigger add_food_to_item() --
```

```

CREATE OR REPLACE FUNCTION add_food_to_item()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.food_status = 'available' THEN
        INSERT INTO items (item_id, food_id, combo_id) VALUES (NEW.food_id, new.food_id, 'NC');
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER after_food_insert
AFTER INSERT ON foods
FOR EACH ROW
EXECUTE FUNCTION add_food_to_item();

```

b. Trigger khi delete food_id to items

```
-- Trigger remove_food_to_item() --
```

```

CREATE OR REPLACE FUNCTION remove_food_from_item()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM items WHERE item_id = OLD.food_id;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER after_food_delete
AFTER DELETE ON foods
FOR EACH ROW
EXECUTE FUNCTION remove_food_from_item();

```

c. Trigger khi insert combo_id to items

```
-- Trigger add_combo_to_item() --

CREATE OR REPLACE FUNCTION add_combo_to_item()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.combo_status = 'available' THEN
        INSERT INTO items (item_id, food_id, combo_id) VALUES (NEW.combo_id, 'NF', NEW.combo_id);
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_combo_insert
AFTER INSERT ON combos
FOR EACH ROW
EXECUTE FUNCTION add_combo_to_item();
```

d. Trigger khi delete combo_id to items

```
-- Trigger remove_combo_to_item() --

CREATE OR REPLACE FUNCTION remove_combo_from_item()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM items WHERE item_id = OLD.combo_id;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_combo_delete
AFTER DELETE ON combos
FOR EACH ROW
EXECUTE FUNCTION remove_combo_from_item();
```

e. Trigger khi update food_status to items

```
-- Trigger update_food_to_item() --

CREATE OR REPLACE FUNCTION update_food_status_to_item()
RETURNS TRIGGER AS $$
BEGIN
    IF OLD.food_status = 'available' AND NEW.food_status = 'unavailable' THEN
        DELETE FROM items WHERE item_id = NEW.food_id;
    ELSEIF OLD.food_status = 'unavailable' AND NEW.food_status = 'available' THEN
        INSERT INTO items(item_id, food_id, combo_id)
            VALUES(NEW.food_id, NEW.food_id, 'NC') ;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_food_status_to_item
AFTER UPDATE OF food_status ON foods
FOR EACH ROW
WHEN (OLD.food_status IS DISTINCT FROM NEW.food_status)
EXECUTE FUNCTION update_food_status_to_item();
```

f. Trigger khi update combo_status to items

```
-- Trigger update_combo_to_item() --

CREATE OR REPLACE FUNCTION update_combo_status_to_item()
RETURNS TRIGGER AS $$
BEGIN
    IF OLD.combo_status = 'available' AND NEW.combo_status = 'unavailable' THEN
        DELETE FROM items WHERE item_id = NEW.combo_id;
    ELSEIF OLD.combo_status = 'unavailable' AND NEW.combo_status = 'available' THEN
        INSERT INTO items(item_id, food_id, combo_id)
            VALUES(NEW.combo_id, 'NF', NEW.combo_id) ;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_combo_status_to_item
AFTER UPDATE OF combo_status ON combos
FOR EACH ROW
WHEN (OLD.combo_status IS DISTINCT FROM NEW.combo_status)
EXECUTE FUNCTION update_combo_status_to_item();
```

g. Trigger thay đổi table_status khi update orders

```
-- Trigger UpdateTableStatusWhenOrder() --

CREATE OR REPLACE FUNCTION UpdateTableStatusWhenOrder()
RETURNS TRIGGER AS $$
DECLARE
    v_table_id INT;
    v_table_status INT;
BEGIN
    -- Get table_id from the inserted or updated row
    v_table_id := NEW.table_id;

    -- Check if table_id is not null
    IF v_table_id IS NOT NULL THEN
        -- Get current status of the table
        SELECT status INTO v_table_status
        FROM tables
        WHERE table_id = v_table_id;

        -- Check if table status is 0 and update if necessary
        IF v_table_status = 0 THEN
            UPDATE tables
            SET status = 1
            WHERE table_id = v_table_id;
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_table_status_when_order_trigger
AFTER INSERT OR UPDATE ON orders
FOR EACH ROW
EXECUTE FUNCTION UpdateTableStatusWhenOrder();
```

h. Trigger tính điểm dựa vào visit_count

```

-- Trigger update_discount_points() --

CREATE OR REPLACE FUNCTION update_discount_points()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.visit_count = 5 THEN
        UPDATE accounts SET discount_point = NEW.discount_point + 4 WHERE phone = NEW.phone;
    ELSIF NEW.visit_count > 5 AND NEW.visit_count % 5 = 0 THEN
        UPDATE accounts SET discount_point = NEW.discount_point + ((NEW.visit_count / 5) - 1) * 10 WHERE phone = NEW.phone;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_discount_points
AFTER UPDATE OF visit_count ON accounts
FOR EACH ROW
WHEN (OLD.visit_count IS DISTINCT FROM NEW.visit_count)
EXECUTE FUNCTION update_discount_points();

```

i. Trigger thi thanh toán hóa đơn

```
-- Trigger UpdateBillWhenPaid() --

CREATE OR REPLACE FUNCTION UpdateBillWhenPaid()
RETURNS TRIGGER AS $$
DECLARE
    v_on_time INT;
    v_having_visit INT;
    v_phone VARCHAR;
    v_having_point FLOAT;
    v_total_present FLOAT;
BEGIN
    -- Update bàn thành trống sau khi thanh toán
    UPDATE tables
    SET status = 0
    WHERE table_id IN (
        SELECT table_id
        FROM orders
        WHERE bill_id = NEW.bill_id
    );

    -- Giảm 1% nếu đặt trước và đến đúng giờ
    SELECT f_on_Time(NEW.bill_id) INTO v_on_time;
    IF v_on_time = 1 THEN
        UPDATE bills SET total_price = NEW.total_price * 0.99
        WHERE bill_id = NEW.bill_id;
    END IF;

    -- Tăng visit_count
    SELECT phone INTO v_phone
    FROM bills WHERE bill_id = NEW.bill_id;

    SELECT visit_count INTO v_having_visit
    FROM accounts
    WHERE phone = v_phone;

    UPDATE accounts SET visit_count = v_having_visit + 1
    WHERE phone = v_phone;
```

```

-- Cộng 2% gtri hóa đơn vào discount_point của KH
SELECT discount_point INTO v_having_point
FROM accounts WHERE phone = v_phone;
-- Lấy total_price đã giảm 1%
SELECT total_price INTO v_total_present
FROM bills WHERE bill_id = NEW.bill_id;
-- Update lại discount_point
UPDATE accounts SET discount_point = v_having_point + ((v_total_present)* 0.2);
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER utg_updatebillwhenpaid_trigger
AFTER UPDATE OF bill_status ON bills
FOR EACH ROW
WHEN (OLD.bill_status IS DISTINCT FROM NEW.bill_status)
EXECUTE FUNCTION UpdateBillWhenPaid();

```

3. Các câu truy vấn

a. Đưa ra tên 5 khách hàng đến quán nhiều lần nhất

```












1 select name from accounts
2 order by visit_count desc
3 limit 5
4

```

Data Output		Messages	Notifications
<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>			
	name		
	character varying		
1	Shelley Barrett		
2	Brandon Mann		
3	Miranda Mora		
4	Michael Long		
5	John York		











b. Đưa ra số điện thoại của các khách hàng có số lần đến quán > 6

```
1 select phone, visit_count from accounts
2 where visit_count > 6
3
```

Data Output Messages Notifications		
        		
	phone [PK] character varying 	visit_count integer 
1	0215378789	12
2	0028516327	10
3	0833557252	7
4	0766268422	10
5	0613373561	12
6	0673471635	11

c. Đưa ra tên 5 khách hàng có điểm tích lũy cao nhất hiện tại












```
1 select name from accounts
2 order by discount_point desc
3 limit 5
4
```

Data Output Messages Notifications	
        	
	name character varying 
1	Matthew Cooper
2	Teresa Walter
3	John York
4	Shelley Barrett
5	Mary Clark

d. Đưa ra sdt của các khách hàng đến quán trong tháng 2024/05/01/ - 2024/05/10

```
1 SELECT a.phone, time_check_in
2 FROM accounts a
3 LEFT JOIN bills b ON a.phone = b.phone
4 WHERE b.time_check_in >= '2024-05-01'
5       AND b.time_check_in < '2024-05-10'
6
```


Data Output Messages Notifications

		
		
		
	phone character varying 	time_check_in timestamp without time zone 
1	0849057603	2024-05-01 12:10:00
2	0507948154	2024-05-03 12:10:00
3	0876552577	2024-05-06 12:10:00
4	0114142649	2024-05-07 12:10:00
5	0420279354	2024-05-08 12:10:00
6	0920695768	2024-05-01 12:10:00
7	0563787306	2024-05-09 12:10:00
8	0837040548	2024-05-06 12:10:00
9	0668073070	2024-05-01 12:10:00
10	0588616144	2024-05-08 12:10:00
11	0158655341	2024-05-09 12:10:00

e. Đưa ra 5 khách hàng đã trả nhiều tiền nhất

```
1 SELECT a.name
2 FROM accounts a
3 INNER JOIN bills b ON a.phone = b.phone
4 group by a.phone
5 order by SUM(b.total_price - b.reduce) DESC
6 limit 5
7
```


Data Output Messages Notifications

	name character varying 
1	Charles Suarez
2	Miranda Mora
3	Randy Williams
4	John Jordan II
5	John Terry

f. Đưa ra số khách hàng đến quán trong ngày “29-05-2024”

```
1 select COUNT(DISTINCT phone) from bills
2 where DATE(time_check_in) = '2024-05-29'
3
```

Data Output Messages Notifications

	count bigint 
1	12

- g. Đưa ra số lượng hóa đơn đã đặt hàng trước và không đến quán trong ngày “29-05-2024”

```
1 select COUNT(bill_id)
2 from bills
3 where DATE(time_check_in) != '2024-05-29' OR time_check_in IS NULL
4     AND DATE(time_booking) < '2024-05-29';
5
```

Data Output		Messages	Notifications
	count bigint		
1	4988		

- h. Đưa ra số lượng hóa đơn có giá trị >50\$ trong ngày “29-05-2024”

```
1 select count(bill_id)
2 from bills
3 where total_price > 50
4 and DATE(time_check_in) = '2024-05-29'
5
```

Data Output		Messages	Notifications
	count bigint		
1	12		

- i. Đưa ra đầy đủ thông tin các món ăn trong combo có giá từ 50\$ – 70\$

```

1 select f.food_id, f.name, f.price, f.food_status, f.description, c.combo_id
2 from foods f
3 inner join join_food_combos jfc on jfc.food_id = f.food_id
4 inner join combos c on jfc.combo_id = c.combo_id
5 where c.price between 50 and 70
6 group by c.combo_id, f.food_id;
7

```

Data Output Messages Notifications

	food_id character varying	name character varying	price integer	food_status character varying	description character varying	combo_id character varying
1	F1	Bangers and mash	14	available	Good	C2
2	F14	Pizza Tofful	15	available	Best Seller	C2
3	F15	Salad	5	available	Special	C2
4	F16	Sushi Japan	10	available	Good	C2
5	F17	Taco	8	available	Good	C2
6	F20	Appetizer	10	available	Good	C2
7	F5	Steak	15	available	Good	C2
8	F2	Fish and chips	12	available	Good	C3
9	F20	Appetizer	10	available	Good	C3
10	F3	Sunday roast	18	available	Good	C3
11	F5	Steak	15	available	Good	C3
12	F7	Afternoon tea	10	available	Good	C3

j. Đưa ra thông tin các món ăn, combo có status là “hết hàng”

```

1 SELECT f.food_id, f.name, f.price, f.food_status, 'Food' AS item_type
2 FROM foods f
3 WHERE f.food_status = 'unavailable'
4 UNION
5 SELECT c.combo_id, c.combo_name, c.price, c.combo_status, 'Combo' AS item_type
6 FROM combos c
7 WHERE c.combo_status = 'unavailable'
8

```

Data Output Messages Notifications

	food_id character varying	name character varying	price integer	food_status character varying	item_type text
1	F6	Full breakfast	12	unavailable	Food
2	C1	Happy Meal	45	unavailable	Combo
3	F13	Burger	8	unavailable	Food
4	F9	Ploughman's lunch	16	unavailable	Food

k. Đưa ra top 5 các món ăn được gọi nhiều nhất theo từng ngày từ 20/05 – 26/05

```
1 SELECT
2     DATE(time_check_in) AS ngay,
3     f.name AS mon_an,
4     SUM(o.quantity) AS so_luong
5 FROM bills b
6 INNER JOIN orders o ON b.bill_id = o.bill_id
7 INNER JOIN items i ON o.item_id = i.item_id
8 INNER JOIN foods f ON i.food_id = f.food_id
9 WHERE
10     DATE(time_check_in) = '2024-05-25'
11     AND f.name != 'null'
12 GROUP BY
13     DATE(time_check_in), f.food_id
14 ORDER BY
15     ngay, so_luong DESC
16 LIMIT 5;
```

Data Output Messages Notifications

	ngay date	mon_an character varying	so_luong bigint
1	2024-05-25	Karen Cardenas	5
2	2024-05-25	Mary Cummings	5
3	2024-05-25	Heather Adams	5
4	2024-05-25	Christine Martinez MD	5
5	2024-05-25	Gregory Evans	5

V. Tối ưu câu truy vấn bằng INDEX

1. Tạo index cho thuộc tính bill_id trong bảng orders
 - Trước khi tạo index

```

1  EXPLAIN ANALYZE
2  SELECT f.name AS item_name
3  FROM bills b
4  INNER JOIN orders o ON b.bill_id = o.bill_id
5  INNER JOIN menus m ON o.item_id = m.item_id
6  INNER JOIN foods f ON m.food_id = f.food_id
7  WHERE DATE(b.time_check_in) = '2024-05-29'

```

Data Output Messages Notifications



	QUERY PLAN	
	text	🔒
1	Nested Loop (cost=184.88..744.53 rows=125 width=14) (actual time=0.841..3.498 rows=60 loops=1)	
2	-> Nested Loop (cost=184.59..698.53 rows=125 width=5) (actual time=0.833..3.276 rows=60 loops=1)	
3	-> Hash Join (cost=184.31..659.99 rows=125 width=4) (actual time=0.809..3.022 rows=60 loops=1)	
4	Hash Cond: (o.bill_id = b.bill_id)	
5	-> Seq Scan on orders o (cost=0.00..410.00 rows=25000 width=8) (actual time=0.010..0.989 rows=25000 loops=1)	
6	-> Hash (cost=184.00..184.00 rows=25 width=4) (actual time=0.753..0.753 rows=12 loops=1)	
7	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
8	-> Seq Scan on bills b (cost=0.00..184.00 rows=25 width=4) (actual time=0.051..0.749 rows=12 loops=1)	
9	Filter: (date(time_check_in) = '2024-05-29'::date)	
10	Rows Removed by Filter: 4988	
11	-> Index Scan using menus_pkey on menus m (cost=0.28..0.31 rows=1 width=10) (actual time=0.004..0.004 rows=1 loops=1)	
12	Index Cond: ((item_id)::text = (o.item_id)::text)	
13	-> Index Scan using foods_pkey on foods f (cost=0.28..0.37 rows=1 width=19) (actual time=0.003..0.003 rows=1 loops=60)	
14	Index Cond: (((food_id)::text = (m.food_id)::text)	
15	Planning Time: 0.550 ms	
16	Execution Time: 3.534 ms	

- Sau khi tạo index:

```

1 CREATE INDEX idx_bill_id ON orders(bill_id);
2 EXPLAIN ANALYZE
3 SELECT f.name AS item_name
4 FROM bills b
5 INNER JOIN orders o ON b.bill_id = o.bill_id
6 INNER JOIN menus m ON o.item_id = m.item_id
7 INNER JOIN foods f ON m.food_id = f.food_id
8 WHERE DATE(b.time_check_in) = '2024-05-29'
9

```

Data Output Messages Notifications

	QUERY PLAN	
	text	🔒
1	Nested Loop (cost=0.85..451.17 rows=125 width=14) (actual time=0.132..1.440 rows=60 loops=1)	
2	-> Nested Loop (cost=0.57..405.17 rows=125 width=5) (actual time=0.120..1.129 rows=60 loops=1)	
3	-> Nested Loop (cost=0.29..366.62 rows=125 width=4) (actual time=0.105..0.798 rows=60 loops=1)	
4	-> Seq Scan on bills b (cost=0.00..184.00 rows=25 width=4) (actual time=0.095..0.650 rows=12 loops=1)	
5	Filter: (date(time_check_in) = '2024-05-29'::date)	
6	Rows Removed by Filter: 4988	
7	-> Index Scan using idx_bill_id on orders o (cost=0.29..7.25 rows=5 width=8) (actual time=0.010..0.011 rows=5 loops=1)	
8	Index Cond: (bill_id = b.bill_id)	
9	-> Index Scan using menus_pkey on menus m (cost=0.28..0.31 rows=1 width=10) (actual time=0.005..0.005 rows=1 loops=1)	
10	Index Cond: ((item_id)::text = (o.item_id)::text)	
11	-> Index Scan using foods_pkey on foods f (cost=0.28..0.37 rows=1 width=19) (actual time=0.005..0.005 rows=1 loops=60)	
12	Index Cond: ((food_id)::text = (m.food_id)::text)	
13	Planning Time: 1.920 ms	
14	Execution Time: 1.489 ms	

2. Tạo index cho thuộc tính total_price trong bảng bills

- Trước khi tạo index:

```

1 explain analyze SELECT * FROM bills WHERE total_price > 100000;
2

```

Data Output Messages Notifications

	QUERY PLAN	
	text	🔒
1	Seq Scan on bills (cost=0.00..171.50 rows=1 width=99) (actual time=0.590..0.590 rows=0 loops=1)	
2	Filter: (total_price > '100000'::double precision)	
3	Rows Removed by Filter: 5000	
4	Planning Time: 0.745 ms	
5	Execution Time: 0.633 ms	

- Sau khi tạo index:

1	CREATE INDEX idx_total_price ON bills (total_price);
2	explain analyze SELECT * FROM bills WHERE total_price > 100000;
3	

Data Output	Messages	Notifications
-------------	----------	---------------

+									
---	--	--	--	--	--	--	--	--	--

	QUERY PLAN	
	text	
1	Index Scan using idx_total_price on bills (cost=0.28..8.30 rows=1 width=99) (actual time=0.003..0.003 rows=0 loop...	
2	Index Cond: (total_price > '100000':double precision)	
3	Planning Time: 1.669 ms	
4	Execution Time: 0.020 ms	

VI. Kết luận

Hệ thống cơ sở dữ liệu cho nhà hàng với các chức năng tạo tài khoản khách hàng, đặt bàn, quản lý bàn ăn, quản lý thực đơn, gọi món, thanh toán và đánh giá đã được thiết kế nhằm tối ưu hóa các quy trình hoạt động và nâng cao trải nghiệm của khách hàng. Các chức năng này bao gồm:

1. Tạo tài khoản cho khách hàng:

- Hệ thống giúp quản lý thông tin khách hàng một cách hiệu quả, đảm bảo không có sự trùng lặp và cung cấp các ưu đãi cho khách hàng thường xuyên.

2. Đặt bàn:

- Quản lý đặt bàn trước và tại chỗ một cách linh hoạt, đảm bảo khách hàng luôn có chỗ ngồi phù hợp và nâng cao khả năng phục vụ của nhà hàng.

3. Quản lý các bàn ăn:

- Hệ thống cho phép theo dõi tình trạng bàn ăn (trống hoặc có người), sức chứa và vị trí bàn, giúp tối ưu hóa việc sử dụng không gian và nâng cao sự thuận tiện cho khách hàng.

4. Quản lý thực đơn, món ăn:

- Cơ sở dữ liệu cung cấp thông tin chi tiết về từng món ăn và combo, giúp khách hàng dễ dàng lựa chọn và nhà hàng dễ dàng cập nhật, quản lý thực đơn.

5. Gọi món:

- Hệ thống cho phép khách hàng gọi món trực tiếp từ thực đơn và liên kết các đơn hàng với hóa đơn, giúp quá trình phục vụ nhanh chóng và chính xác hơn.

6. Thanh toán:

- Quy trình thanh toán được tối ưu hóa với các tùy chọn giảm giá và điểm thưởng, khuyến khích khách hàng quay lại và tạo điều kiện thuận lợi cho việc thanh toán.

7. Đánh giá:

- Khách hàng có thể đánh giá và bình luận về món ăn, giúp nhà hàng nhận được phản hồi quý giá để cải thiện chất lượng dịch vụ và sản phẩm.

Lợi ích của hệ thống

- **Nâng cao trải nghiệm khách hàng:** Khách hàng được phục vụ nhanh chóng, tiện lợi từ khâu đặt bàn đến thanh toán và đánh giá.
- **Tăng cường hiệu quả quản lý:** Nhà hàng có thể quản lý thông tin khách hàng, bàn ăn và món ăn một cách hiệu quả, tối ưu hóa quy trình hoạt động.
- **Cải thiện chất lượng dịch vụ:** Phản hồi từ khách hàng giúp nhà hàng nhận diện điểm mạnh và yếu, từ đó nâng cao chất lượng dịch vụ và món ăn.
- **Khuyến khích khách hàng quay lại:** Chương trình tích điểm và giảm giá khuyến khích khách hàng quay lại nhà hàng, tăng cường sự trung thành của khách hàng.

Hệ thống cơ sở dữ liệu này không chỉ giúp nhà hàng vận hành trơn tru hơn mà còn tạo ra nhiều giá trị gia tăng cho khách hàng và nhà quản lý, góp phần vào sự phát triển bền vững của nhà hàng trong tương lai.