

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO PROJECT MÔN HỌC
Hệ thống quản lý trung tâm Toán học

Học phần: Thực hành cơ sở dữ liệu
Mã học phần: IT3290

Giảng viên hướng dẫn: TS. Nguyễn Thị Oanh

Sinh viên thực hiện: Nguyễn Thành Bách - 20235660 (Trưởng nhóm)
Hoàng Đức Anh - 20235640
Vũ Anh - 20235657

Mã lớp học: 156777

Lời mở đầu

Trong những năm gần đây, hoạt động dạy thêm - học thêm ở Việt Nam đã và đang diễn ra phổ biến trong môi trường sư phạm, đáp ứng nhu cầu nâng cao kiến thức, bồi dưỡng năng lực cá nhân. Tuy nhiên, hoạt động dạy thêm – học thêm ở Việt Nam gặp phải nhiều bất cập lớn. Trước hết là công tác quản lý “thả nổi” , một số giáo viên và nhà trường lợi dụng vị trí “chủ nhiệm, giảng dạy chính khóa” để ép học sinh đi học thêm dưới đủ hình thức, thậm chí tận dụng giờ chính khóa để “chạy” nội dung thêm, ép học sinh đăng ký học thêm dưới nhiều hình thức “tự nguyện” mờ ám, dẫn tới học sinh cảng mình với lịch học nối dài từ sáng tới tối, một số nơi còn sử dụng đòn bẩy như dùng ví dụ, bài tập trong giờ học thêm để kiểm tra chính thức. Hơn nữa, công tác quản lý dạy thêm – học thêm cũng thiếu minh bạch, các lớp ngoài giờ thường không minh bạch về giảng viên, học phí và nội dung, chưa yêu cầu đăng ký kinh doanh hay công khai thông tin, khiến phụ huynh và học sinh dễ nhầm lẫn giữa lớp chính quy và trung tâm giáo dục. Bên cạnh đó, người dân và các cơ quan chức năng chưa được trao đủ quyền, phương tiện để giám sát hiệu quả, dẫn đến việc kiểm tra, giám sát chưa đủ hiệu quả, sát sao.

Trước bối cảnh đó, sự ra đời của **Thông tư 29/2024/TT-BGDĐT** (hiệu lực từ 14/2/2025) đã định hình một khuôn khổ pháp lý mới. Thông tư khẳng định dạy thêm là một hoạt động cần thiết trong bối cảnh xã hội phát triển, nhưng đòi hỏi phải được tổ chức và quản lý một cách chuyên nghiệp, minh bạch và tuân thủ pháp luật. Đây vừa là thách thức, vừa là cơ hội để các trung tâm giáo dục chuẩn hóa hoạt động của mình, đáp ứng nhu cầu xã hội trong khi vẫn đảm bảo chất lượng và sự tin cậy. Do đó nhu cầu xây dựng một hệ thống quản lý trung tâm dạy học chuyên nghiệp trở nên cấp thiết hơn bao giờ hết.

Và đó cũng chính là lí do chúng em lựa chọn đề tài **Hệ thống quản lý trung tâm Toán học**. Hệ thống được xây dựng trên nền tảng Web nhằm số hóa và chuẩn hóa toàn bộ quy trình hoạt động của trung tâm. Toàn bộ dữ liệu được quản lý tập trung trên một **cơ sở dữ liệu** duy nhất, đảm bảo tính nhất quán, an toàn và cho phép truy xuất các báo cáo cần thiết và nhanh chóng. Đồng thời, dữ liệu về quá trình dạy học và kết quả kiểm tra định kỳ, đánh giá về lớp học và giáo viên được cập nhật liên tục vào cơ sở dữ liệu. Điều này tạo nền tảng vững chắc cho công tác kiểm định chất lượng đào tạo trung tâm, đánh giá năng lực học viên một cách minh bạch, chính xác từ đó có thể giúp trung tâm cải thiện ngày một tốt hơn.

Để có thể hoàn thành báo cáo đề tài “**Hệ thống quản lý trung tâm Toán học**”, nhóm 1 chúng em xin gửi lời cảm ơn chân thành nhất tới cô **Nguyễn Thị Oanh** - người đã truyền đạt, giảng dạy cho chúng em những kiến thức, những kinh nghiệm quý giá trong thời gian học tập môn này, tận tình chữa lỗi và giải đáp thắc mắc cho chúng em về Project môn học này.

Bằng nỗ lực chung của cả nhóm và sự hướng dẫn tận tình của cô thì chúng em đã hoàn thiện được báo cáo này. Song với kinh nghiệm và kiến thức còn hạn chế, báo cáo khó tránh khỏi những sai sót. Chúng em luôn sẵn sàng tiếp thu mọi ý kiến đóng góp xây dựng từ thầy cô và các bạn .Chúng em xin chân thành cảm ơn .

Mục lục

Lời mở đầu	2
1 Mô tả dự án	5
1.1 Tổng quan	5
1.2 Mô tả nghiệp vụ	5
1.2.1 Cơ cấu và Tổ chức	5
1.2.2 Quy trình nhập học	6
1.2.3 Thanh toán học phí	7
1.2.4 Quy trình giảng dạy	7
1.3 Mô tả chức năng	8
2 Thiết kế cơ sở dữ liệu	10
2.1 Sơ đồ thực thể liên kết	10
2.2 Sơ đồ quan hệ	11
2.3 Chi tiết bảng	11
3 Trigger	15
3.1 Trigger tự động update số lớp	15
3.2 Trigger đảm bảo 1 lớp có tối đa 30 học viên	16
3.3 Trigger tránh trùng lịch	17
4 SQL Query	19
4.1 Câu 1-10: Nguyễn Thành Bách	19
4.2 Câu 11-20: Hoàng Đức Anh	36
4.3 Câu 21-30: Vũ Anh	52
5 Xây dựng chương trình	68
5.1 Công nghệ sử dụng	68
5.2 Chức năng và giao diện chương trình	68
6 Tổng kết	75
6.1 Khó khăn gặp phải và cách khắc phục	75
6.2 Đánh giá kết quả đã đạt được	75
6.3 Phân công công việc	76

Phần 1

Mô tả dự án

1.1 Tổng quan

- Hệ thống Quản Lý Trung Tâm Toán Học nhằm cung cấp một hệ thống quản lý thông tin toàn diện cho các trung tâm toán học luyện thi đại học và có tiềm năng mở rộng ra các môn khác.
- Hệ thống này giúp quản lý và theo dõi các nhân viên, học viên, giáo viên, lớp học, điểm số, và các báo cáo tổng quan về trung tâm.
- Việc quản lý thông tin sẽ được tự động hóa và dễ dàng truy cập thông qua một giao diện Web đơn giản và dễ sử dụng với người dùng.

1.2 Mô tả nghiệp vụ

1.2.1 Cơ cấu và Tổ chức

Đối tượng khách hàng

- Trung tâm hướng đến đối tượng học viên muốn cải thiện kiến thức toán học, nâng cao kỹ năng giải quyết vấn đề và chuẩn bị kiến thức cho các kỳ thi như kỳ thi tuyển sinh đại học, TSA, HSA.

Mô hình Lớp học và Nhân sự

- Trung tâm sẽ chuyên tâm cho luyện thi đại học, sẽ có nhiều loại lớp khác nhau cho khách hàng lựa chọn.
- Trung tâm chuyên tâm cho luyện thi đại học, cung cấp nhiều loại lớp khác nhau cho học viên lựa chọn, hiện tại gồm có 4 loại I, M, O, E.
- Đội ngũ giáo viên có trình độ chuyên môn cao gồm các cấp bậc Cử nhân, Thạc sĩ, và Tiến sĩ.
- Một lớp sẽ do một nhân viên và giáo viên quản lý. Nhân viên sẽ phụ trách quản lý thông tin lớp học và hỗ trợ giáo viên nhập điểm. Giáo viên sẽ phụ trách giảng dạy và chấm điểm các bài kiểm tra của học viên.

Mô tả các loại lớp hiện có của trung tâm

- ◊ **LỚP I (Intro) — Nền tảng:** Xây chắc kiến thức Toán 12 từ cơ bản đến nâng cao.
- ◊ **LỚP M (Mastery) — Tăng tốc:** Chuyên sâu các chuyên đề VD-VDC Toán 12.
- ◊ **LỚP O (Optimization) — Luyện đề:** Thi thử thực chiến, luyện các bộ đề nâng cao điểm số.
- ◊ **LỚP E (Excellent) — Tổng ôn:** Rà soát toàn bộ kiến thức trọng yếu Toán 10, 11, 12.

Tuyển sinh

- Định kỳ mỗi tháng, trung tâm sẽ mở ra nhiều lớp học offline trực tiếp tại trung tâm. Số lượng lớp mở sẽ căn cứ vào số lượng giáo viên, cơ sở vật chất và kế hoạch đào tạo của trung tâm.
- Trước mỗi đợt khai giảng thì sẽ có các bài marketing trên các nền tảng mạng xã hội để bắt đầu nhận học viên mới.
- Học viên có thể học theo lộ trình IMOЕ hoặc chọn loại lớp mình cần. Mỗi loại ứng với một nội dung học khác nhau.

1.2.2 Quy trình nhập học

Quy trình

1. Học viên sẽ đến trực tiếp trung tâm để nghe tư vấn và đăng ký hoặc là có thể trao đổi online trước rồi đến đăng ký.
2. Khi đến trung tâm đăng ký thì đầu tiên học viên sẽ điền các thông tin cá nhân, mục tiêu học tập cho nhân viên trung tâm và trao đổi qua về các lớp học mà mình quan tâm.
3. Sau khi nhân viên tiếp nhận yêu cầu thì sẽ cho khách hàng làm 1 bài thi thử offline ngay tại trung tâm. Độ khó bài test sẽ tùy thuộc vào lớp học mà học viên muốn đăng ký.
4. Căn cứ vào điểm bài test mà nhân viên trung tâm sẽ đưa ra danh sách lớp gợi ý phù hợp với học viên.
5. Học viên sẽ chọn lớp mình muốn đăng ký và thanh toán trực tiếp với kế toán tại quầy. Học viên không thể đổi lớp một khi đã hoàn thiện đăng ký.
6. Nhân viên xác nhận đăng ký và cập nhật thông tin học viên lên hệ thống.

7. Học viên nhận các tài liệu cần thiết và đợi đến lúc khai giảng sẽ bắt đầu quá trình học.

1.2.3 Thanh toán học phí

- Phương thức thanh toán:** Trung tâm hiện tại áp dụng hình thức thanh toán học phí trực tiếp tại quầy giao dịch. Trung tâm không triển khai việc lưu trữ dữ liệu giao dịch trên hệ thống Web mà chỉ lưu trên giấy tờ.
- Quy định về học phí:** Học phí cần được hoàn tất một lần ngay tại thời điểm đăng ký lớp học. Trung tâm không áp dụng chính sách ghi nợ học phí hay trả góp để đảm bảo công tác tài chính được minh bạch và ổn định.
- Xác nhận giao dịch:** Sau khi học viên hoàn tất thanh toán, nhân viên trung tâm sẽ tiến hành ghi nhận thông tin giao dịch và cung cấp hóa đơn (bản giấy) để xác nhận rồi sau đó thêm học viên vô hệ thống.

1.2.4 Quy trình giảng dạy

- Học viên sẽ đi học hàng tuần theo lịch đã được thông báo từ giáo viên/nhân viên .
- Giáo viên ngẫu nhiên điểm danh rồi giảng dạy theo giáo trình đã soạn sẵn. Học viên có thể nhắn tin xin phép trước để được miễn vắng nếu có lý do phù hợp.
- Một lớp sẽ chỉ có 1 giáo viên và 1 nhân viên để quản lý. Mỗi lớp sẽ học trong 12 tuần. Mỗi tuần 3 buổi. Mỗi buổi 2 tiếng. Mỗi lớp có tối đa 30 học viên.
- Xuyên suốt quá trình học tại trung tâm thì học viên sẽ trải qua tổng cộng 6 bài kiểm tra:
 - Định kì mỗi 2 tuần sẽ có 1 bài minitest. Tổng có 4 bài (ứng với tuần 2,4,8,10).
 - Bài kiểm tra giữa khóa (tuần 6).
 - Bài kiểm tra cuối khóa (tuần 12).
- Sau mỗi bài minitest hoặc giữa khóa / cuối khóa thì giáo viên sẽ chấm và gửi nhân viên để nhập điểm lên hệ thống.
- Học viên sẽ nhận lại bài kiểm tra và được giáo viên nhận xét và chia lỗi sai tại lớp.
- Học viên sẽ nhận bằng khen nếu đủ điều kiện lúc tổng kết khóa (điểm tổng kết ≥ 8 , số buổi vắng ≤ 3).

★ Công thức tính điểm tổng kết:

$$\text{Điểm tổng kết} = \frac{\text{Tổng 4 bài minitest}}{4} \times 40\% + \text{GK} \times 30\% + \text{CK} \times 30\%$$

8. Sau khi tổng kết mỗi lớp thì học viên đều sẽ được làm 1 form khảo sát đánh giá cho điểm giáo viên, lớp học.
9. Điểm đánh giá sẽ được đưa lên hệ thống. Sẽ có thống kê điểm trung bình đánh giá của từng giáo viên, lớp học.
10. Trung tâm sẽ dựa vào đánh giá mà có thể thay đổi và cải thiện chiến lược, chất lượng giảng dạy hơn.

1.3 Mô tả chức năng

- Hệ thống được thiết kế là một công cụ quản lý online toàn diện của trung tâm với các chức năng cụ thể như sau:

◊ **Xem thống kê tổng quan:** Admin có thể xem các báo cáo tổng quan về tài chính, nhân sự của cả trung tâm, điểm đánh giá của giáo viên, lớp học hoặc là bảng xếp hạng học viên .

◊ **Quản lý Học viên:**

- ▷ Thêm học viên mới, sửa đổi thông tin, và xóa hồ sơ học viên khỏi hệ thống.
- ▷ Xem danh sách chi tiết các học viên đã đăng ký vào từng khóa học.

◊ **Quản lý Giáo viên:**

- ▷ Thêm, sửa, và xóa hồ sơ thông tin của giáo viên.
- ▷ Xem danh sách chi tiết các nhân viên.

◊ **Quản lý Nhân viên:**

- ▷ Thêm nhân viên mới, sửa đổi thông tin, và xóa hồ sơ nhân viên khỏi hệ thống.
- ▷ Xem danh sách chi tiết các nhân viên.

◊ **Quản lý Lớp học:**

- ▷ Thêm lớp học , loại lớp học mới, chỉnh sửa thông tin chi tiết của lớp (giáo viên phụ trách, lịch học, phòng học).
- ▷ Cập nhật số lượng học viên bằng cách thêm hoặc xóa học viên khỏi một lớp cụ thể.

◊ Quản lý Feedback:

- ▷ Thêm đánh giá mới bằng cách nhập thủ công hoặc nhập bằng dữ liệu từ File Excel.
- ▷ Xem danh sách đánh giá và chi tiết từng đánh giá.

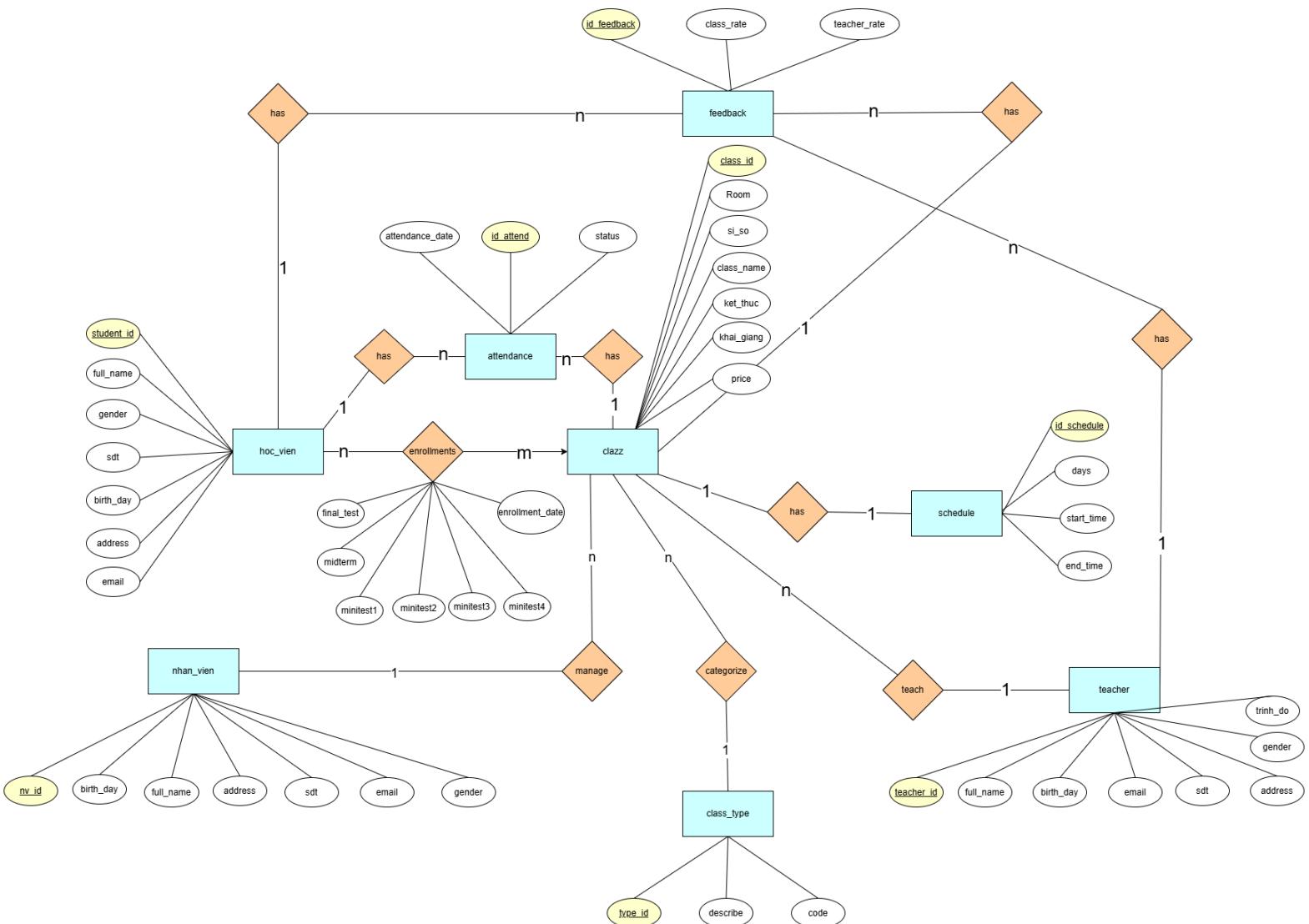
Bảng 1.1: Mô tả chức năng

Chức năng	Input	Output
Đăng nhập	Tài khoản , mật khẩu	Đăng nhập thành công rồi chuyển tới Dashboard
Đăng xuất		Người dùng thoát khỏi hệ thống
Dashboard thống kê		Thống kê doanh thu , bảng xếp hạng , ...
Thêm học viên mới	Thông tin học viên	Danh sách học viên được cập nhật
Tìm kiếm học viên	Tên, ID , email , SĐT ,..	Danh sách học viên
Xóa/Sửa học viên	Học viên cần xóa/sửa	Học viên bị xóa/sửa
Thêm giáo viên mới	Thông tin giáo viên	Danh sách giáo viên được cập nhật
Tìm kiếm giáo viên	Tên, ID , email , SĐT ,..	Danh sách giáo viên
Xóa/Sửa giáo viên	Giáo viên cần xóa/sửa	Giáo viên bị xóa/sửa
Thêm nhân viên mới	Thông tin nhân viên	Danh sách nhân viên được cập nhật
Tìm kiếm nhân viên	Tên, ID , email , SĐT ,..	Danh sách nhân viên
Xóa/Sửa nhân viên	Nhân viên cần xóa/sửa	Nhân viên bị xóa/sửa
Thêm lớp học/loại lớp học mới	Thông tin cần thiết	Danh sách lớp học/loại lớp học được cập nhật
Thêm học viên vô lớp	Thông tin học viên	Học viên được thêm vô lớp , sĩ số được +1
Xóa học viên khỏi lớp	Thông tin học viên	Học viên được xóa khỏi lớp , sĩ số lớp - 1
Thêm đánh giá	Thông tin đánh giá	Danh sách đánh giá được cập nhật

Phần 2

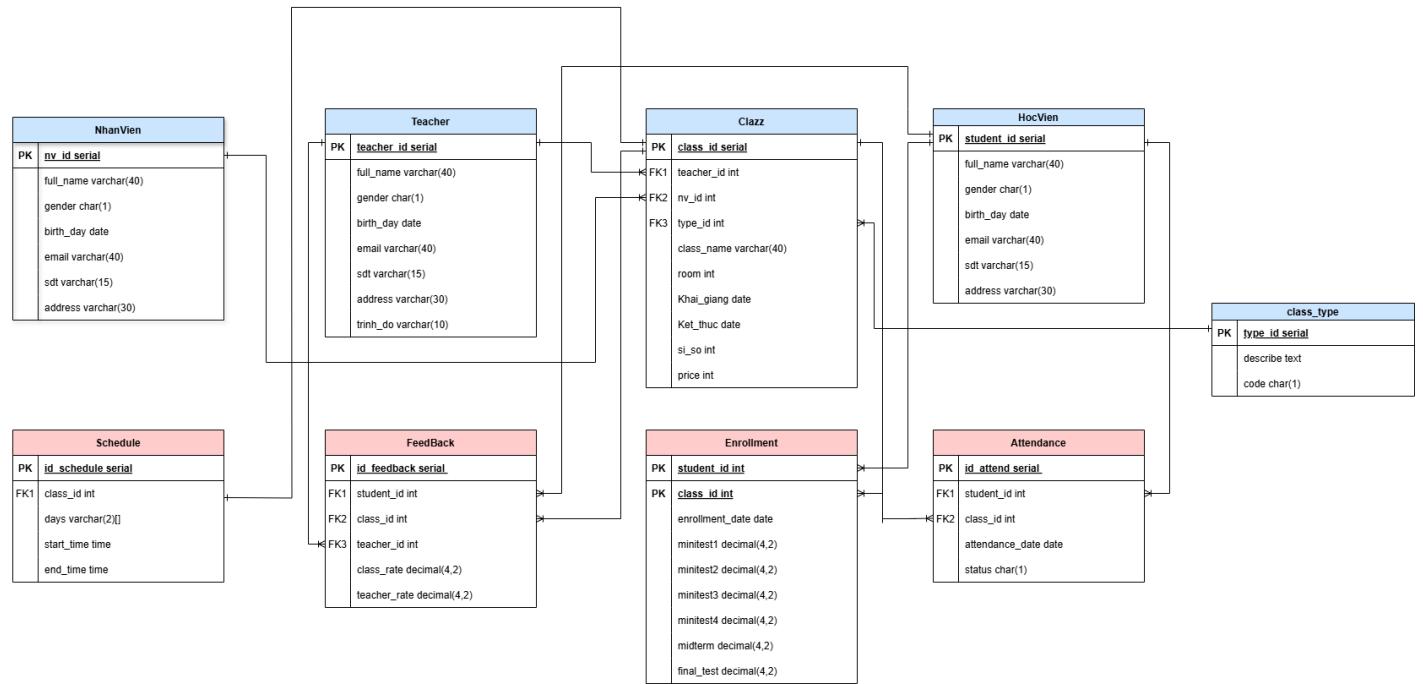
Thiết kế cơ sở dữ liệu

2.1 Sơ đồ thực thể liên kết



Hình 2.1: Sơ đồ thực thể-liên kết

2.2 Sơ đồ quan hệ



Hình 2.2: Sơ đồ quan hệ

2.3 Chi tiết bảng

Bảng nhan_vien

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
nv_id	SERIAL	PK	ID nhân viên
full_name	VARCHAR(40)	NOT NULL	Họ và tên
gender	CHAR(1)	CHECK ('M','F')	Giới tính
birth_day	DATE	NOT NULL	Ngày sinh
email	VARCHAR(40)	NOT NULL	Email
sdt	VARCHAR(15)	NOT NULL	Số điện thoại
address	VARCHAR(30)	NOT NULL	Địa chỉ

Bảng teacher

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
teacher_id	SERIAL	PK	ID giáo viên
full_name	VARCHAR(40)	NOT NULL	Họ và tên
gender	CHAR(1)	CHECK ('M','F')	Giới tính
birth_day	DATE	NOT NULL	Ngày sinh
email	VARCHAR(40)	NOT NULL	Email
sdt	VARCHAR(15)	NOT NULL	Số điện thoại
address	VARCHAR(30)	NOT NULL	Địa chỉ
trinh_do	VARCHAR(10)	CHECK('Cử nhân', 'Thạc Sĩ', 'Tiến Sĩ')	Trình độ học vấn

Bảng hoc_vien

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
student_id	SERIAL	PK	ID học viên
full_name	VARCHAR(40)	NOT NULL	Họ và tên
gender	CHAR(1)	CHECK ('M','F')	Giới tính
birth_day	DATE	NOT NULL	Ngày sinh
email	VARCHAR(40)	NOT NULL	Email
sdt	VARCHAR(15)	NOT NULL	Số điện thoại
address	VARCHAR(30)	NOT NULL	Địa chỉ

Bảng class_type

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
type_id	SERIAL	PK	ID loại lớp
describe	TEXT	NOT NULL	Mô tả chi tiết
code	CHAR(1)	NOT NULL	Mã loại lớp

Bảng clazz

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
class_id	SERIAL	PK	ID lớp học
nv_id	INTEGER	FK → nhan_vien	ID nhân viên quản lý
teacher_id	INTEGER	FK → teacher	ID giáo viên phụ trách
type_id	INTEGER	FK → class_type	ID loại lớp học
class_name	VARCHAR(40)	NOT NULL	Tên lớp học
room	INT	NOT NULL	Phòng học
khai_giang	DATE	NOT NULL	Ngày khai giảng
ket_thuc	DATE	NOT NULL, CHECK	Ngày kết thúc (3 tháng)
si_so	INTEGER	NOT NULL, DEFAULT 0	Sĩ số
price	INTEGER	NOT NULL	Học phí

Bảng schedule

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
id_schedule	SERIAL	PK	ID lịch học
class_id	INTEGER	FK → clazz	ID lớp học
days	VARCHAR(2) []	(‘2’,’3’,’4’,’5’,’6’,’7’,’CN’)	Mảng 3 ngày học
start_time	TIME	NOT NULL	Giờ bắt đầu
end_time	TIME	NOT NULL, CHECK	Giờ kết thúc (2h)

Bảng enrollments

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
student_id	INTEGER	PK, NN, FK → hoc_vien	ID học viên
class_id	INTEGER	PK, NN, FK → clazz	ID lớp học
enrollment_date	DATE	NOT NULL	Ngày đăng ký
minitest1	DECIMAL(4,2)	CHECK [0-10]	Điểm minitest
minitest2	DECIMAL(4,2)	CHECK [0-10]	Điểm minitest
minitest3	DECIMAL(4,2)	CHECK [0-10]	Điểm minitest
minitest4	DECIMAL(4,2)	CHECK [0-10]	Điểm minitest
midterm	DECIMAL(4,2)	CHECK [0-10]	Điểm giữa kỳ
final	DECIMAL(4,2)	CHECK [0-10]	Điểm cuối kỳ

Bảng attendance

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
id_attend	SERIAL	PK	ID điểm danh
student_id	INTEGER	FK → hoc_vien	ID học viên
class_id	INTEGER	FK → clazz	ID lớp học
attendance_date	DATE	NOT NULL	Ngày điểm danh
status	CHAR(1)	CHECK ('0','1')	Trạng thái (vắng/có)

Bảng feedback

Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
id_feedback	SERIAL	PK	ID phản hồi
class_rate	DECIMAL(4,2)	NN, [0-10]	Điểm đánh giá lớp
teacher_rate	DECIMAL(4,2)	NN, CHECK [0-10]	Điểm đánh giá GV
student_id	INTEGER	FK → hoc_vien	ID học viên phản hồi
class_id	INTEGER	FK → clazz	ID lớp được phản hồi
teacher_id	INTEGER	FK → teacher	ID GV được phản hồi

Phần 3

Trigger

Trong quá trình thiết kế cơ sở dữ liệu , các ràng buộc đã giúp đảm bảo tính toàn vẹn dữ liệu ở mức độ cơ bản. Tuy nhiên, trong thực tế vận hành, có rất nhiều quy tắc nghiệp vụ phức tạp mà các ràng buộc này không thể xử lý, ví dụ như tự động cập nhật số lượng học viên mới, hay kiểm tra các điều kiện logic liên quan đến nhiều bảng khác nhau. Khi đó ta sẽ cần tới Trigger.

3.1 Trigger tự động update số lượng lớp

```
CREATE OR REPLACE FUNCTION update_class_size()
RETURNS TRIGGER AS
$$
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE clazz
        SET si_so = si_so + 1
        WHERE class_id = NEW.class_id;
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE clazz
        SET si_so = si_so - 1
        WHERE class_id = OLD.class_id;
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER update_class_size_trigger
AFTER INSERT OR DELETE ON enrollments
FOR EACH ROW
EXECUTE FUNCTION update_class_size();
```

★ Trigger trên sẽ tự động update cột *si_so* của bảng *clazz* khi có thêm một học viên được thêm/xóa khỏi lớp.

3.2 Trigger đảm bảo 1 lớp có tối đa 30 học viên

```
CREATE OR REPLACE FUNCTION check_max_si_so()
RETURNS TRIGGER AS
$$
DECLARE current_size INTEGER;
BEGIN
    SELECT si_so INTO current_size
    FROM clazz
    WHERE class_id = NEW.class_id;

    IF current_size >= 30 THEN
        RAISE EXCEPTION 'Class (ID= % ) is full (30) !! Can not add more
student', NEW.class_id;
    END IF;

    RETURN NEW;
END ;
$$
LANGUAGE plpgsql;

CREATE TRIGGER tg_check_max_si_so
BEFORE INSERT ON enrollments
FOR EACH ROW
EXECUTE FUNCTION check_max_si_so();
```

★ Trigger trên sẽ đảm bảo số lượng học viên trong lớp không vượt quá 30. Nếu số lượng đã đầy thì sẽ không cho phép thêm học viên.

3.3 Trigger tránh trùng lịch

```

CREATE OR REPLACE FUNCTION validate_schedule()
RETURNS TRIGGER AS $$

DECLARE
    v_new_khai_giang DATE;
    v_new_ket_thuc DATE;
    v_new_room INTEGER;
    v_teacher_id INTEGER;
    v_conflict_exists BOOLEAN;

BEGIN
    -- New schedule
    SELECT c.khai_giang, c.ket_thuc, c.room, c.teacher_id
    INTO v_new_khai_giang, v_new_ket_thuc, v_new_room, v_teacher_id
    FROM clazz c
    WHERE c.class_id = NEW.class_id;

    -- Check room
    SELECT EXISTS (
        SELECT 1
        FROM schedule s JOIN clazz c ON s.class_id = c.class_id
        WHERE c.room = v_new_room
        AND s.class_id <> NEW.class_id
        AND s.days && NEW.days
        AND (NEW.start_time, NEW.end_time) OVERLAPS (s.start_time, s.end_time)
        AND (v_new_khai_giang, v_new_ket_thuc) OVERLAPS (c.khai_giang, c.ket_thuc)
    ) INTO v_conflict_exists;

    IF v_conflict_exists THEN
        RAISE EXCEPTION 'Duplicate classrooms: Room % used in this time.'
        , v_new_room;
    END IF;

    -- Check teacher schedule
    SELECT EXISTS (
        SELECT 1
        FROM schedule s JOIN clazz c ON s.class_id = c.class_id
        WHERE c.teacher_id = v_teacher_id
        AND s.class_id <> NEW.class_id
        AND s.days && NEW.days
        AND (NEW.start_time, NEW.end_time) OVERLAPS (s.start_time, s.end_time)
        AND (v_new_khai_giang, v_new_ket_thuc) OVERLAPS (c.khai_giang,
    
```

```
c.ket_thuc)
) INTO v_conflict_exists;

IF v_conflict_exists THEN
    RAISE EXCEPTION 'Duplicate teacher schedule (ID: %)', 
v_teacher_id;
END IF;

RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER tg_validate_schedule
BEFORE INSERT OR UPDATE ON schedule
FOR EACH ROW
EXECUTE FUNCTION validate_schedule();
```

★Trigger này sẽ đảm bảo lịch giữa các lớp không bị trùng , giáo viên không bị trùng tiết.

Phần 4

SQL Query

4.1 Câu 1-10: Nguyễn Thành Bách

Danh sách các câu truy vấn

1. Danh sách học viên dùng sim Vinaphone
 2. Tính doanh thu của trung tâm trong tháng 9/2023.
 3. Danh sách các lớp khai giảng trong tháng 8/2024.
 4. Danh sách học viên đã đăng ký các lớp loại I và O.
 5. Danh sách học viên có ít nhất 1 bài kiểm tra 0 điểm.
 6. Danh sách các lớp có giá cao nhất.
 7. View hiển thị top 10 học viên điểm tổng kết cao nhất(bằng nhau thì xếp theo tên)
 8. Danh sách các lớp chưa có Feedback nào.
 9. Danh sách các học viên tên "Giang"
 10. Danh sách các lớp tại phòng 201 kết thúc trong 3 tháng đầu năm 2025
-

CÂU 1: Danh sách học viên dùng sđt Vinaphone

Cách 1:Dùng OR

```
SELECT student_id, full_name, sdt
FROM hoc_vien
WHERE sdt LIKE '091%' OR sdt LIKE '094%' OR sdt LIKE '088%' OR
      sdt LIKE '081%' OR sdt LIKE '082%' OR sdt LIKE '083%' OR
      sdt LIKE '084%' OR sdt LIKE '085%';
```

Phân tích hiệu năng: Câu này có chi phí là **1141.0** , ta không thể dùng index ở câu này vì có toán tử OR ở trong WHERE.

QUERY PLAN
text
Seq Scan on hoc_vien (cost=0.00..1141.00 rows=3361 width=32) (actual time=0.081..10.631 rows=3523 loops=1)
Filter: (((sdt)::text ~~ '091%')::text) OR (((sdt)::text ~~ '094%')::text) OR (((sdt)::text ~~ '088%')::text) OR (((sdt)::text ~~ '081%')::text) OR (((sdt)::text ~~ '082%')::text) OR (((sdt)::text ~~ '083%')::text) OR (((sdt)::text ~~ '084%')::text) OR (((sdt)::text ~~ '085%')::text)
Rows Removed by Filter: 16477
Planning Time: 0.179 ms
Execution Time: 10.821 ms

Hình 4.1: Query Plan cách 1

Cách 2:Dùng Regular Expression

```
SELECT student_id,full_name,sdt
FROM hoc_vien
WHERE sdt ~ '^(091|094|088|081|082|083|084|085) ';
```

Phân tích hiệu năng: Câu này có chi phí là **791.0** - tối ưu hơn cách 1.Tương tự cách 1 ta không thể dùng index ở câu này vì có điều kiện OR ở trong WHERE.

QUERY PLAN
text
Seq Scan on hoc_vien (cost=0.00..791.00 rows=3636 width=32) (actual time=0.051..10.367 rows=3523 loops=...)
Filter: (((sdt)::text ~ '^^(091 094 088 081 082 083 084 085)'::text)
Rows Removed by Filter: 16477
Planning Time: 0.121 ms
Execution Time: 10.470 ms

Hình 4.2: Query Plan cách 2

Cách 3:Dùng hàm SUBSTRING

```
SELECT student_id ,full_name ,sdt
FROM hoc_vien
WHERE SUBSTRING(sdt , 1 , 3) IN ('091' , '094' , '088' , '081' , '082' ,
'083' , '084' , '085');
```

Phân tích hiệu năng: Câu này có chi phí là **991.0**. Khác với 2 câu trên thì do câu này dùng IN nên index sẽ có hiệu quả trong câu này. Ta sẽ dùng Functional Index:

```
CREATE INDEX idx_hocvien_sdt_prefix
ON hoc_vien( (SUBSTRING(sdt , 1 , 3)) );
```

QUERY PLAN	
text	Seq Scan on hoc_vien (cost=0.00..991.00 rows=800 width=32) (actual time=0.116..10.604 rows=3523 loops=...
	Filter: ("substring"((sdt)::text, 1, 3) = ANY ('{091,094,088,081,082,083,084,085}'::text[]))
	Rows Removed by Filter: 16477
	Planning Time: 0.118 ms
	Execution Time: 10.756 ms

Hình 4.3: Trước khi dùng index

QUERY PLAN	
text	Bitmap Heap Scan on hoc_vien (cost=32.20..617.69 rows=800 width=32) (actual time=0.290..1.298 rows=3523 loops=1)
	Recheck Cond: ("substring"((sdt)::text, 1, 3) = ANY ('{091,094,088,081,082,083,084,085}'::text[]))
	Heap Blocks: exact=274
	-> Bitmap Index Scan on idx_hocvien_sdt_prefix (cost=0.00..32.00 rows=800 width=0) (actual time=0.235..0.235 rows=3523 loop...
	Index Cond: ("substring"((sdt)::text, 1, 3) = ANY ('{091,094,088,081,082,083,084,085}'::text[]))
	Planning Time: 0.127 ms
	Execution Time: 1.479 ms

Hình 4.4: Sau khi dùng index

★ Ta có thể thấy sau khi dùng index thì chi phí đã giảm thiểu còn **617.69** - Tối ưu nhất trong 3 cách.

Câu 2: Tính doanh thu của trung tâm trong tháng 9/2023

Cách 1: Dùng bảng tạm

```
WITH RevenuePerClass AS (
    SELECT c.class_id, c.price * c.si_so AS class_revenue
    FROM clazz c
    JOIN enrollments e USING(class_id)
    WHERE enrollment_date >= '2023-09-01'
        AND enrollment_date < '2023-10-01'
    GROUP BY c.class_id
)
SELECT SUM(class_revenue) AS total_revenue
FROM RevenuePerClass;
```

Phân tích hiệu năng: Câu truy vấn dùng 1 bảng tạm để tính doanh thu từng lớp trong T9/2023 rồi tính tổng lại. Câu truy vấn có chi phí là **615.64**. Trong câu lệnh WHERE có dùng AND nên sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **420.94**.

```
CREATE INDEX idx_enrollment_date ON enrollments(enrollment_date);
```

QUERY PLAN	
text	
Aggregate (cost=615.63..615.64 rows=1 width=8) (actual time=1.298..1.299 rows=1 loops=1)	
-> HashAggregate (cost=601.31..608.47 rows=573 width=8) (actual time=1.275..1.294 rows=49 loops=1)	
Group Key: c.class_id	
Batches: 1 Memory Usage: 49kB	
-> Hash Join (cost=203.50..599.88 rows=573 width=12) (actual time=0.358..1.206 rows=567 loops=1)	
Hash Cond: (e.class_id = c.class_id)	
-> Seq Scan on enrollments e (cost=0.00..394.86 rows=573 width=4) (actual time=0.068..0.832 rows=...)	
Filter: ((enrollment_date >= '2023-09-01'::date) AND (enrollment_date < '2023-10-01'::date))	
Rows Removed By Filter: 14157	
-> Hash (cost=191.00..191.00 rows=1000 width=12) (actual time=0.285..0.285 rows=1000 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 51kB	
-> Seq Scan on clazz c (cost=0.00..191.00 rows=1000 width=12) (actual time=0.003..0.213 rows=...)	
Planning Time: 0.345 ms	
Execution Time: 1.348 ms	

a) Trước khi có index

QUERY PLAN	
text	
Aggregate (cost=413.53..413.54 rows=1 width=8) (actual time=1.033..1.035 rows=1 loops=1)	
-> HashAggregate (cost=399.20..406.36 rows=573 width=8) (actual time=1.014..1.027 rows=49 loops=1)	
Group Key: c.class_id	
Batches: 1 Memory Usage: 49kB	
-> Hash Join (cost=213.66..397.77 rows=573 width=12) (actual time=0.608..0.897 rows=567 loops=1)	
Hash Cond: (e.class_id = c.class_id)	
-> Bitmap Heap Scan on enrollments e (cost=10.16..192.75 rows=573 width=4) (actual time=0.119..0.271 rows=567 loops=1)	
Recheck Cond: ((enrollment_date >= '2023-09-01'::date) AND (enrollment_date < '2023-10-01'::date))	
Heap Blocks: exact=47	
-> Bitmap Index Scan on idx_enrollment_date (cost=0.00..10.02 rows=573 width=0) (actual time=0.100..0.100 rows=567)	
Index Cond: ((enrollment_date >= '2023-09-01'::date) AND (enrollment_date < '2023-10-01'::date))	
-> Hash (cost=191.00..191.00 rows=1000 width=12) (actual time=0.481..0.481 rows=1000 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 51kB	
-> Seq Scan on clazz c (cost=0.00..191.00 rows=1000 width=12) (actual time=0.010..0.332 rows=1000 loops=1)	
Planning Time: 2.907 ms	
Execution Time: 1.106 ms	

b) Sau khi có index

Cách 2:Dùng IN với Subquery

```
SELECT SUM(c.price * c.si_so) AS total_revenue
FROM clazz c
WHERE c.class_id IN (
    SELECT e.class_id
    FROM enrollments e
    WHERE enrollment_date >= '2023-09-01' AND enrollment_date < '
2023-10-01'
);
```

Phân tích hiệu năng: Câu truy vấn này sử dụng 1 Subquery để tạo ra danh sách các lớp có học viên đăng ký trong T9/2023, sau đó dùng toán tử IN để lọc bảng clazz. Câu truy vấn có chi phí là **609.04** khi chưa có index. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **406.94**(Tối ưu hơn cách 1).

```
CREATE INDEX idx_enrollment_date ON enrollments(enrollment_date);
```

QUERY PLAN
text

```
Aggregate (cost=609.03..609.04 rows=1 width=8) (actual time=1.705..1.707 rows=1 loops=1)
-> Hash Join (cost=406.17..606.17 rows=573 width=8) (actual time=1.446..1.699 rows=49 loops=1)
  Hash Cond: (c.class_id = e.class_id)
  -> Seq Scan on clazz c (cost=0.00..191.00 rows=1000 width=12) (actual time=0.010..0.177 rows=1000 loop...
  -> Hash (cost=400.68..400.68 rows=439 width=4) (actual time=1.401..1.402 rows=49 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 10kB
    -> HashAggregate (cost=396.29..400.68 rows=439 width=4) (actual time=1.386..1.394 rows=49 loops=1)
      Group Key: e.class_id
      Batches: 1 Memory Usage: 37kB
      -> Seq Scan on enrollments e (cost=0.00..394.86 rows=573 width=4) (actual time=0.065..1.289 rows...
        Filter: ((enrollment_date >= '2023-09-01'::date) AND (enrollment_date < '2023-10-01'::date))
        Rows Removed by Filter: 14157
Planning Time: 1.814 ms
Execution Time: 1.751 ms
```

a) Trước khi có index

QUERY PLAN
text

```
Aggregate (cost=406.93..406.94 rows=1 width=8) (actual time=0.728..0.729 rows=1 loops=1)
-> Hash Join (cost=204.06..404.06 rows=573 width=8) (actual time=0.426..0.720 rows=49 loops=1)
  Hash Cond: (c.class_id = e.class_id)
  -> Seq Scan on clazz c (cost=0.00..191.00 rows=1000 width=12) (actual time=0.015..0.212 rows=1000 loops=1)
  -> Hash (cost=198.58..198.58 rows=439 width=4) (actual time=0.375..0.376 rows=49 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 10kB
    -> HashAggregate (cost=194.19..198.58 rows=439 width=4) (actual time=0.352..0.361 rows=49 loops=1)
      Group Key: e.class_id
      Batches: 1 Memory Usage: 37kB
      -> Bitmap Heap Scan on enrollments e (cost=10.16..192.75 rows=573 width=4) (actual time=0.076..0.260 rows=56...
        Recheck Cond: ((enrollment_date >= '2023-09-01'::date) AND (enrollment_date < '2023-10-01'::date))
        Heap Blocks: exact=47
        -> Bitmap Index Scan on idx_enrollment_date (cost=0.00..10.02 rows=573 width=0) (actual time=0.058..0.058 r...
          Index Cond: ((enrollment_date >= '2023-09-01'::date) AND (enrollment_date < '2023-10-01'::date))
Planning Time: 2.702 ms
Execution Time: 0.776 ms
```

b) Sau khi có index

Câu 3: Danh sách các lớp khai giảng trong tháng 1/2025

Cách 1: Dùng hàm EXTRACT

```
SELECT c.class_name AS ten_lop_hoc ,
       c.khai_giang AS ngay_khai_giang ,
       t.full_name AS giao_vien_phu_trach ,
       c.price AS hoc_phi ,
       c.si_so
  FROM clazz c
 JOIN teacher t USING (teacher_id)
 WHERE EXTRACT(YEAR FROM c.khai_giang) = 2025
   AND EXTRACT(MONTH FROM c.khai_giang) = 1;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm EXTRACT để trích xuất năm và tháng thỏa mãn điều kiện .Câu truy vấn có chi phí là **209.32**. Sau khi thêm index, chi phí đã giảm còn **89.68**.

★ Cần lưu ý về thứ tự cột khi dùng Composite index

```
CREATE INDEX idx_extract_year_month_khai_giang
ON clazz(EXTRACT(YEAR FROM khai_giang),EXTRACT(MONTH FROM khai_giang)
)
```

QUERY PLAN
text
Nested Loop (cost=0.27..209.32 rows=1 width=54) (actual time=0.064..0.569 rows=39 loops=1)
-> Seq Scan on clazz c (cost=0.00..201.00 rows=1 width=41) (actual time=0.050..0.480 rows=39 loops=1)
Filter: ((EXTRACT(year FROM khai_giang) = '2025'::numeric) AND (EXTRACT(month FROM khai_giang) = '1'::numeric))
Rows Removed by Filter: 961
-> Index Scan using teacher_pkey on teacher t (cost=0.27..8.29 rows=1 width=21) (actual time=0.002..0.002 rows=1 loops=1)
Index Cond: (teacher_id = c.teacher_id)
Planning Time: 0.505 ms
Execution Time: 0.601 ms

a) Trước khi có index

QUERY PLAN
text
Hash Join (cost=31.72..89.68 rows=19 width=54) (actual time=0.173..0.221 rows=39 loops=1)
Hash Cond: (c.teacher_id = t.teacher_id)
-> Bitmap Heap Scan on clazz c (cost=4.47..62.38 rows=19 width=41) (actual time=0.033..0.068 rows=39 loops=1)
Recheck Cond: ((EXTRACT(year FROM khai_giang) = '2025'::numeric) AND (EXTRACT(month FROM khai_giang) = '1'::numeric))
Heap Blocks: exact=34
-> Bitmap Index Scan on idx_extract_year_month_khai_giang (cost=0.00..4.47 rows=19 width=0) (actual time=0.018..0.018 rows=39 loops=1)
Index Cond: ((EXTRACT(year FROM khai_giang) = '2025'::numeric) AND (EXTRACT(month FROM khai_giang) = '1'::numeric))
-> Hash (cost=21.00..21.00 rows=500 width=21) (actual time=0.133..0.133 rows=500 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 35kB
-> Seq Scan on teacher t (cost=0.00..21.00 rows=500 width=21) (actual time=0.008..0.068 rows=500 loops=1)
Planning Time: 0.331 ms
Execution Time: 0.249 ms

b) Sau khi có index

Cách 2: Xét khoảng

```

SELECT c.class_name AS ten_lop_hoc ,
       c.khai_giang AS ngay_khai_giang ,
       t.full_name AS giao_vien_phu_trach ,
       c.price AS hoc_phi ,
       c.si_so
FROM clazz c
JOIN teacher t USING (teacher_id)
WHERE c.khai_giang >= '2025-01-01' AND c.khai_giang < '2025-02-01';
    
```

Phân tích hiệu năng: Câu truy vấn này xét giá trị cột *khai_giang* trên 1 khoảng giá trị. Câu truy vấn có chi phí là **223.35**. Sau khi thêm index đã giảm còn **128.45**. Cách này không tối ưu bằng cách trên nhưng linh hoạt hơn vì có thể tìm trên khoảng cụ thể (ví dụ tìm các lớp khai giảng từ 22/8/2023 -> 15/9/2024)

```
CREATE INDEX idx_khai_giang ON clazz(khai_giang)
```

QUERY PLAN text	
1	CTE Scan on calculatedgrades cg (cost=455.26..455.57 rows=1 width=134) (actual time=1.228..1.229 rows=1 loops=1)
2	Filter: (calculated_final_grade = (InitPlan 2).col1)
3	Rows Removed by Filter: 25
4	CTE calculatedgrades
5	-> Nested Loop (cost=0.29..454.93 rows=14 width=57) (actual time=1.013..1.125 rows=26 loops=1)
6	-> Seq Scan on enrollments e (cost=0.00..342.31 rows=14 width=44) (actual time=0.985..0.988 rows=26 loops=1)
7	Filter: (class_id = 1000)
8	Rows Removed by Filter: 14959
9	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..8.02 rows=1 width=21) (actual time=0.004..0.004 rows=1 loops=1)
10	Index Cond: (student_id = e.student_id)
11	InitPlan 2
12	-> Aggregate (cost=0.32..0.33 rows=1 width=32) (actual time=0.207..0.207 rows=1 loops=1)
13	-> CTE Scan on calculatedgrades (cost=0.00..0.28 rows=14 width=32) (actual time=0.001..0.118 rows=26 loops=1)
14	Planning Time: 1.235 ms
15	Execution Time: 1.653 ms

a) Trước khi có index

QUERY PLAN text	
1	CTE Scan on calculatedgrades cg (cost=121.48..121.79 rows=1 width=134) (actual time=0.242..0.243 rows=1 loops=1)
2	Filter: (calculated_final_grade = (InitPlan 2).col1)
3	Rows Removed by Filter: 25
4	CTE calculatedgrades
5	-> Nested Loop (cost=0.57..121.15 rows=14 width=57) (actual time=0.042..0.210 rows=26 loops=1)
6	-> Index Scan using idx_enrollments_class_id on enrollments e (cost=0.29..8.53 rows=14 width=44) (actual time=0.020..0.028 rows=26 loops=1)
7	Index Cond: (class_id = 1000)
8	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..8.02 rows=1 width=21) (actual time=0.005..0.005 rows=1 loops=26)
9	Index Cond: (student_id = e.student_id)
10	InitPlan 2
11	-> Aggregate (cost=0.32..0.33 rows=1 width=32) (actual time=0.189..0.189 rows=1 loops=1)
12	-> CTE Scan on calculatedgrades (cost=0.00..0.28 rows=14 width=32) (actual time=0.000..0.179 rows=26 loops=1)
13	Planning Time: 0.693 ms
14	Execution Time: 0.720 ms

b) Sau khi có index

CÂU 4: Danh sách mã học viên đã đăng ký các lớp loại I và O

Cách 1: JOIN các bảng rồi kiểm tra điều kiện

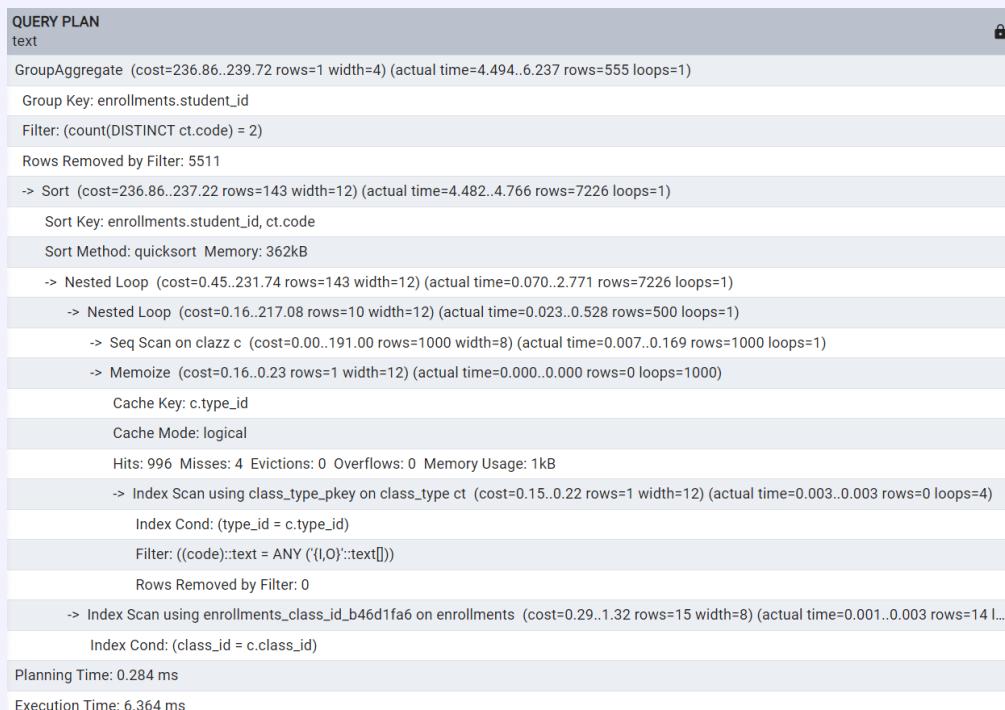
```
SELECT student_id
FROM enrollments
JOIN clazz c USING (class_id)
JOIN class_type ct USING (type_id)
WHERE ct.code IN ('I', 'O')
GROUP BY student_id
HAVING COUNT (DISTINCT ct.code)=2;
```

Phân tích hiệu năng: Cách này phải JOIN khá nhiều bảng rồi sau đó mới kiểm tra điều kiện bằng WHERE và HAVING. Câu truy vấn trải qua khá là nhiều bước và có chi phí là **239.72**. Ta nhận thấy là DBMS đã thực hiện Seq Scan lên bảng clazz để thực hiện nối với class_type.

→ Ta có thể thử thêm index lên bảng clazz.

```
CREATE INDEX idxClazz_type_id ON clazz(type_id);
```

Tuy nhiên ta nhận thấy sau khi thêm index thì DBMS vẫn thực hiện Seq Scan trên bảng clazz. Điều này có thể hiểu được vì type_id chỉ có đúng 4 giá trị khác nhau nên việc tạo index không hề giúp cải thiện hiệu năng.



a) Query Plan trước và sau index

Cách 2: Dùng INTERSECT

```

SELECT student_id
FROM enrollments
JOIN clazz c USING (class_id)
JOIN class_type ct USING (type_id)
WHERE ct.code = 'I'
INTERSECT
select student_id
FROM enrollments
JOIN clazz c USING (class_id)
JOIN class_type ct USING (type_id)
WHERE ct.code = 'O';

```

Phân tích hiệu năng: Câu truy vấn này dùng INTERSECT và có chi phí là **451.56** (không tối ưu bằng cách trên do phép JOIN được dùng nhiều lần). Cũng như cách trên thì thêm index không hề cải thiện được hiệu năng.Có lẽ do lượng dữ liệu nhỏ nên chi phí Index Scan(bao gồm chi phí truy nhập chỉ số và tìm kiếm thực tế) cao hơn so với quét tuần tự toàn bộ bảng nên DBMS vẫn chọn .

```
CREATE INDEX idxClazz_type_id ON clazz(type_id);
```

QUERY PLAN
text
HashSetOp Intersect (cost=0.45..451.56 rows=78 width=8) (actual time=4.937..5.000 rows=555 loops=1)
-> Append (cost=0.45..451.17 rows=156 width=8) (actual time=0.060..3.966 rows=7226 loops=1)
-> Subquery Scan on *SELECT* 1* (cost=0.45..225.19 rows=78 width=8) (actual time=0.060..1.781 rows=3621 loops=1)
-> Nested Loop (cost=0.45..224.41 rows=78 width=4) (actual time=0.060..1.536 rows=3621 loops=1)
-> Nested Loop (cost=0.16..217.09 rows=5 width=4) (actual time=0.027..0.481 rows=250 loops=1)
-> Seq Scan on clazz c (cost=0.00..191.00 rows=1000 width=8) (actual time=0.006..0.160 rows=1000 loops=1)
-> Memoize (cost=0.16..0.23 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=1000)
Cache Key: c.type_id
Cache Mode: logical
Hits: 996 Misses: 4 Evictions: 0 Overflows: 0 Memory Usage: 1kB
-> Index Scan using class_type_pkey on class_type ct (cost=0.15..0.22 rows=1 width=4) (actual time=0.003..0.003 rows=0 loops=4)
Index Cond: (type_id = c.type_id)
Filter: ((code)::text = 'I)::text
Rows Removed by Filter: 1
-> Index Scan using enrollments_class_id_b46d1fa6 on enrollments (cost=0.29..1.32 rows=15 width=8) (actual time=0.001..0.003 rows=14 loops=250)
Index Cond: (class_id = c.class_id)
-> Subquery Scan on *SELECT* 2* (cost=0.45..225.19 rows=78 width=8) (actual time=0.058..1.802 rows=3605 loops=1)
-> Nested Loop (cost=0.45..224.41 rows=78 width=4) (actual time=0.057..1.557 rows=3605 loops=1)
-> Nested Loop (cost=0.16..217.09 rows=5 width=4) (actual time=0.024..0.459 rows=250 loops=1)
-> Seq Scan on clazz c_1 (cost=0.00..191.00 rows=1000 width=8) (actual time=0.013..0.147 rows=1000 loops=1)
-> Memoize (cost=0.16..0.23 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=1000)
Cache Key: c_1.type_id
Cache Mode: logical
Hits: 996 Misses: 4 Evictions: 0 Overflows: 0 Memory Usage: 1kB
-> Index Scan using class_type_pkey on class_type ct_1 (cost=0.15..0.22 rows=1 width=4) (actual time=0.002..0.002 rows=0 loops=4)
Index Cond: (type_id = c_1.type_id)
Filter: ((code)::text = 'O)::text
Rows Removed by Filter: 1
-> Index Scan using enrollments_class_id_b46d1fa6 on enrollments enrollments_1 (cost=0.29..1.32 rows=15 width=8) (actual time=0.002..0.005 rows=14 loops=250)
Index Cond: (class_id = c_1.class_id)

Planning Time: 0.605 ms

Execution Time: 8.647 ms

a) Query Plan trước và sau khi có index

CÂU 5: Danh sách học viên có ít nhất 1 bài kiểm tra 0 điểm

Cách 1:Dùng OR

```
SELECT h.student_id, h.full_name
FROM hoc_vien h
JOIN enrollments USING (student_id)
WHERE minitest1 = 0 OR minitest2 = 0 OR minitest3=0 OR minitest4 = 0
      OR midterm = 0 OR final_test = 0
GROUP BY h.student_id;
```

Phân tích hiệu năng: Câu này phải dùng toán tử OR để check khá nhiều điều kiện trên các cột điểm khác nhau trong bảng enrollments do đó hiệu suất không cao . Chi phí = **1693.66**

QUERY PLAN	text	🔒
HashAggregate (cost=1689.50..1693.66 rows=416 width=21) (actual time=7.021..7.052 rows=413 loops=1)		
Group Key: h.student_id		
Batches: 1 Memory Usage: 61kB		
-> Hash Join (cost=718.30..1688.46 rows=416 width=21) (actual time=4.814..6.946 rows=415 loops=1)		
Hash Cond: (h.student_id = enrollments.student_id)		
-> Seq Scan on hoc_vien h (cost=0.00..741.00 rows=20000 width=21) (actual time=0.009..0.929 rows=20000 loops=1)		
-> Hash (cost=713.10..713.10 rows=416 width=4) (actual time=4.776..4.776 rows=415 loops=1)		
Buckets: 1024 Batches: 1 Memory Usage: 23kB		
-> Seq Scan on enrollments (cost=0.00..713.10 rows=416 width=4) (actual time=0.109..4.734 rows=415 loops=1)		
Filter: ((minitest1 = '0'::numeric) OR (minitest2 = '0'::numeric) OR (minitest3 = '0'::numeric) OR (minitest4 = '0'::numeric) OR (midterm = '0'::numeric) OR (final_test = '0'::numeric))		
Rows Removed by Filter: 14309		
Planning Time: 0.351 ms		
Execution Time: 7.104 ms		

a) Query Plan

Ta cũng không thể cải thiện hiệu năng bằng index vì câu truy vấn dùng toán tử OR, càng không thể chuyển sang UNION vì sẽ phải duyệt lại 1 bảng tận 6 lần. ⇒ Ta phải nghĩ đến việc viết bằng cách khác để cải thiện hiệu năng.

Cách 2: Biến thành các cột điểm thành 1 mảng

```
SELECT h.student_id, h.full_name
FROM hoc_vien h
JOIN enrollments USING (student_id)
WHERE ARRAY[minitest1, minitest2, minitest3, minitest4, midterm,
final_test] @> ARRAY[0.00]
GROUP BY h.student_id;
```

Phân tích hiệu năng: Cách này sẽ gộp 6 cột điểm thành chung 1 mảng rồi dùng toán tử @> để check điều kiện. Việc dùng mảng và toán tử @> tất cả là vì để có thể dùng index GIN .Đây là inverted index , mỗi index lưu trữ (key , list) với key là giá trị của cột (điểm) còn list là tập ID (vị trí) mà xuất hiện key ấy .Việc dùng index đã giảm thiểu chi phí từ **1094.29** xuống còn **758.76**

```
CREATE INDEX idx_gin_enrollments_grades ON enrollments
USING GIN((ARRAY[minitest1, minitest2, minitest3, minitest4, midterm,
final_test]));
```

QUERY PLAN	
text	
Group	(cost=1093.92..1094.29 rows=74 width=21) (actual time=9.087..9.175 rows=413 loops=1)
Group Key:	h.student_id
-> Sort	(cost=1093.92..1094.10 rows=74 width=21) (actual time=9.086..9.106 rows=415 loops=1)
Sort Key:	h.student_id
Sort Method:	quicksort Memory: 41kB
-> Nested Loop	(cost=0.29..1091.62 rows=74 width=21) (actual time=0.171..8.968 rows=415 loops=1)
-> Seq Scan on enrollments	(cost=0.00..529.05 rows=74 width=4) (actual time=0.162..8.007 rows=415 loops=1)
Filter:	(ARRAY[minitest1, minitest2, minitest3, minitest4, midterm, final_test] @> '{0.00}':numeric[])
Rows Removed by Filter:	14309
-> Index Scan using hoc_vien_pkey on hoc_vien h	(cost=0.29..7.60 rows=1 width=21) (actual time=0.002..0.002 rows=1 loops=415)
Index Cond:	(student_id = enrollments.student_id)
Planning Time:	1.345 ms
Execution Time:	9.220 ms

a) Trước khi có index

QUERY PLAN	
text	
Group	(cost=758.39..758.76 rows=74 width=21) (actual time=0.883..0.943 rows=413 loops=1)
Group Key:	h.student_id
-> Sort	(cost=758.39..758.57 rows=74 width=21) (actual time=0.881..0.896 rows=415 loops=1)
Sort Key:	h.student_id
Sort Method:	quicksort Memory: 41kB
-> Nested Loop	(cost=13.46..756.09 rows=74 width=21) (actual time=0.070..0.808 rows=415 loops=1)
-> Bitmap Heap Scan on enrollments	(cost=13.17..193.52 rows=74 width=4) (actual time=0.065..0.217 rows=415 loops=1)
Recheck Cond:	(ARRAY[minitest1, minitest2, minitest3, minitest4, midterm, final_test] @> '{0.00}':numeric[])
Heap Blocks:	exact=156
-> Bitmap Index Scan on idx_gin_enrollments_grades	(cost=0.00..13.16 rows=74 width=0) (actual time=0.043..0.043 rows=413)
Index Cond:	(ARRAY[minitest1, minitest2, minitest3, minitest4, midterm, final_test] @> '{0.00}':numeric[])
-> Index Scan using hoc_vien_pkey on hoc_vien h	(cost=0.29..7.60 rows=1 width=21) (actual time=0.001..0.001 rows=1 loops=413)
Index Cond:	(student_id = enrollments.student_id)
Planning Time:	0.270 ms
Execution Time:	0.980 ms

b) Sau khi có index

CÂU 6:Danh sách các lớp có giá cao nhất

Cách 1: Dùng ALL

```
SELECT class_id, class_name, price
FROM clazz
WHERE price >= ALL (SELECT price FROM clazz);
```

Phân tích hiệu năng: Câu này dùng 1 subquery để tìm giá của các lớp rồi kiểm tra điều kiện \geq ALL nhằm tìm ra price cao nhất. Do câu truy vấn sẽ so sánh price với mọi price từ subquery nên chi phí của cách này cực kì cao : **99443.50**

QUERY PLAN	
text	
Seq Scan on clazz (cost=0.00..99443.50 rows=500 width=33) (actual time=0.664..5.652 rows=27 loops=1)	
Filter: (ALL (price >= (SubPlan 1).col1))	
Rows Removed by Filter: 973	
SubPlan 1	
-> Materialize (cost=0.00..196.00 rows=1000 width=4) (actual time=0.000..0.002 rows=33 loops=1000)	
-> Seq Scan on clazz clazz_1 (cost=0.00..191.00 rows=1000 width=4) (actual time=0.005..0.327 rows=1000 loops=1)	
Planning Time: 0.137 ms	
Execution Time: 5.697 ms	

a) Query Plan

→ Ta nên dùng cách khác sao cho subquery không chạy nhiều lần

Cách 2: Dùng MAX

```
SELECT class_id, class_name, price
FROM clazz
WHERE price = (SELECT MAX(price) FROM clazz);
```

Phân tích hiệu năng: Cách này chỉ cần so sánh price với 1 giá trị có được từ câu truy vấn con nên chi phí đã tối ưu hơn cách 1 rất nhiều (**387.01**). Nhờ chuyển từ dùng ALL sang dùng MAX ta cũng có thể tạo index để tối ưu hơn nữa (**96.87**).

```
CREATE INDEX idx_price ON clazz(price)
```

QUERY PLAN	
text	
Seq Scan on clazz (cost=193.51..387.01 rows=37 width=33) (actual time=0.413..0.547 rows=27 loops=1)	
Filter: (price = (InitPlan 1).col1)	
Rows Removed by Filter: 973	
InitPlan 1	
-> Aggregate (cost=193.50..193.51 rows=1 width=4) (actual time=0.391..0.391 rows=1 loops=1)	
-> Seq Scan on clazz clazz_1 (cost=0.00..191.00 rows=1000 width=4) (actual time=0.003..0.230 rows=1000 loops=1)	
Planning Time: 0.164 ms	
Execution Time: 0.572 ms	
Bitmap Heap Scan on clazz (cost=4.62..96.87 rows=37 width=33) (actual time=0.037..0.056 rows=27 loops=1)	
Recheck Cond: (price = (InitPlan 2).col1)	
Heap Blocks: exact=27	
InitPlan 2	
-> Result (cost=0.17..0.18 rows=1 width=4) (actual time=0.015..0.016 rows=1 loops=1)	
InitPlan 1	
-> Limit (cost=0.15..0.17 rows=1 width=4) (actual time=0.014..0.014 rows=1 loops=1)	
-> Index Only Scan Backward using idx_price on clazz clazz_1 (cost=0.15..23.15 rows=1000 width=4) (actual time=0.013..0.013 rows=1)	
Heap Fetches: 0	
-> Bitmap Index Scan on idx_price (cost=0.00..4.43 rows=37 width=0) (actual time=0.028..0.028 rows=27 loops=1)	
Index Cond: (price = (InitPlan 2).col1)	
Planning Time: 0.138 ms	
Execution Time: 0.077 ms	

a) Trước khi có index

b) Sau khi có index

CÂU 7: View hiển thị top 10 học viên điểm tổng kết cao nhất (bằng nhau thì xếp theo tên)

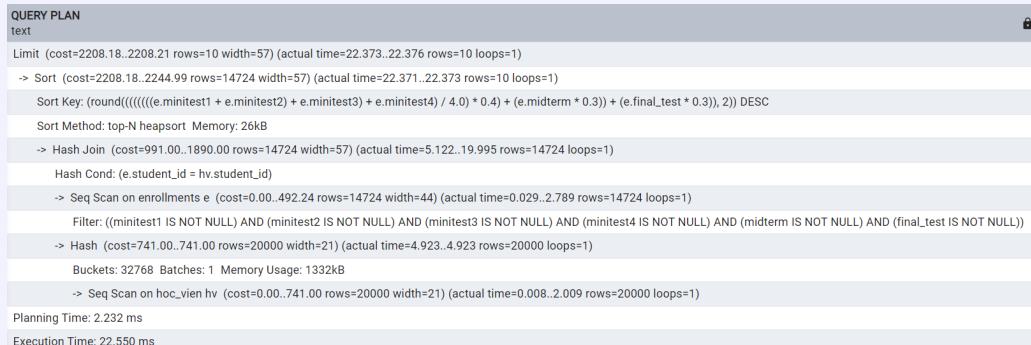
```

CREATE VIEW top_10_student_final_scores AS
SELECT hv.full_name AS student_name, hv.student_id, e.class_id,
       ROUND(
           (e.minitest1 + e.minitest2 + e.minitest3 + e.minitest4) / 4.0
           * 0.4 +
           e.midterm * 0.3 + e.final_test * 0.3
           , 2) AS final_score
FROM hoc_vien hv
JOIN enrollments e USING(student_id)
WHERE (e.minitest1, e.minitest2, e.minitest3, e.minitest4, e.midterm,
       e.final_test) IS NOT NULL
ORDER BY final_score DESC, student_name ASC
LIMIT 10;

```

Phân tích hiệu năng: Đây là 1 view nhằm mục đích thống kê bảng xếp hạng top 10 học viên có điểm tổng kết cao nhất và sẽ được hiển thị ở phần **Dashboard** bên phía ứng dụng. Câu lệnh bên trong View có chi phí là **2208.21**.

Ta không thể cải thiện câu này bằng index được nữa vì điều kiện WHERE dùng toán tử NOT nên index sẽ không được sử dụng.



a) Query Plan câu lệnh trong View

CÂU 8:Danh sách các lớp chưa có feedback nào (đã kết thúc)

Cách 1:Dùng NOT EXIST

```
SELECT c.class_id ,c.class_name ,c.ket_thuc
FROM clazz c
WHERE c.ket_thuc < CURRENT_DATE AND
      NOT EXISTS (
          SELECT 1
          FROM feedback f
          WHERE f.class_id = c.class_id
      );
```

Phân tích hiệu năng: Câu này dùng cách rất quen thuộc đó là Subquery bên trong NOT EXISTS , câu truy vấn có chi phí là **599.29** .

QUERY PLAN	
text	
Hash Anti Join (cost=396.17..599.29 rows=35 width=33) (actual time=4.754..5.359 rows=33 loops=1)	
Hash Cond: (c.class_id = f.class_id)	
-> Seq Scan on clazz c (cost=0.00..196.00 rows=925 width=33) (actual time=0.033..0.443 rows=926 loops=1)	
Filter: (ket_thuc < CURRENT_DATE)	
Rows Removed by Filter: 74	
-> Hash (cost=230.52..230.52 rows=13252 width=4) (actual time=4.370..4.370 rows=13252 loops=1)	
Buckets: 16384 Batches: 1 Memory Usage: 594kB	
-> Seq Scan on feedback f (cost=0.00..230.52 rows=13252 width=4) (actual time=0.017..1.994 rows=13252 loops=1)	
Planning Time: 1.801 ms	
Execution Time: 5.616 ms	

Hình 4.16: Query Plan

Tuy nhiên ta có thể dùng cách khác để tránh phải có Subquery chạy nhiều lần ảnh hưởng hiệu năng.

Cách 2:Dùng LEFT JOIN

```
SELECT c.class_id ,c.class_name ,c.ket_thuc
FROM clazz c
LEFT JOIN feedback f ON c.class_id = f.class_id
WHERE c.ket_thuc < CURRENT_DATE AND f.id_feedback IS NULL;
```

Phân tích hiệu năng: Cách này dùng LEFT JOIN kết hợp với check điều kiện NULL và có chi phí là **473.01**. Cách này đã tối ưu hơn cách 1 vì tránh phải chạy Subquery nhiều lần.

QUERY PLAN	
text	🔒
Hash Right Join (cost=207.56..473.01 rows=1 width=33) (actual time=5.311..5.349 rows=33 loops=1)	
Hash Cond: (f.class_id = c.class_id)	
Filter: (f.id_feedback IS NULL)	
Rows Removed by Filter: 12233	
-> Seq Scan on feedback f (cost=0.00..230.52 rows=13252 width=8) (actual time=0.010..1.349 rows=132...)	
-> Hash (cost=196.00..196.00 rows=925 width=33) (actual time=0.626..0.628 rows=926 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 69kB	
-> Seq Scan on clazz c (cost=0.00..196.00 rows=925 width=33) (actual time=0.014..0.453 rows=926 l...)	
Filter: (ket_thuc < CURRENT_DATE)	
Rows Removed by Filter: 74	
Planning Time: 0.368 ms	
Execution Time: 5.391 ms	

Hình 4.17: Query Plan

CÂU 9:Danh sách các sinh viên tên "Giang"

Cách 1:Dùng các Array Function

```
SELECT * FROM hoc_vien
WHERE split_part(full_name, ' ', array_length(string_to_array(
    full_name, ' '), 1)) = 'Giang';
```

Phân tích hiệu năng: Câu này dùng 1 loạt các array function do Postgre cung cấp để tách từ cuối ở chuỗi full_name ra để truy vấn được tên.Câu truy vấn có chi phí là **941.0**.Tuy nhiên dùng nhiều hàm như vậy thì sẽ không được tối ưu
→Ta nên dùng cách khác

QUERY PLAN	
text	Seq Scan on hoc_vien (cost=0.00..941.00 rows=100 width=76) (actual time=0.241..20.493 rows=342 loops=1)
	Filter: (split_part((full_name)::text, ' '::text, array_length(string_to_array((full_name)::text, ' '::text), 1)) = 'Giang'::te...)
	Rows Removed by Filter: 19658
Planning Time:	0.180 ms
Execution Time:	20.599 ms

Cách 2:Dùng LIKE

```
SELECT * FROM hoc_vien
WHERE full_name LIKE '%Giang';
```

Phân tích hiệu năng: Câu này dùng 1 cách rất quen thuộc đó là dùng toán tử LIKE và có chi phí là **791.0** - tối ưu hơn cách 1.Ta không thể dùng index Btree để cải thiện hiệu năng cách này vì biểu thức trong LIKE bắt đầu = %.Tuy nhiên qua tìm hiểu document của Postgre thì em có tìm thấy 1 loại index rất mạnh cho bài toán search text đó chính là index GIN with Trigrams .

★Ta có thể thấy sau khi thêm index thì chi phí đã giảm xuống còn **615.27**

```
CREATE INDEX idx_hv_full_name_trgm
ON hoc_vien USING GIN (full_name gin_trgm_ops);
```

QUERY PLAN	
text	Seq Scan on hoc_vien (cost=0.00..791.00 rows=598 width=76) (actual time=0.092..4.205 rows=342 loops=1)
Filter: ((full_name)::text ~~ '%Giang'::text)	Recheck Cond: ((full_name)::text ~~ '%Giang'::text)
Rows Removed by Filter: 19658	Heap Blocks: exact=196
Planning Time: 0.162 ms	-> Bitmap Index Scan on idx_hoc_vien_full_name_trgm (cost=0.00..41.79 rows=598 width=0) (actual time=0.290..0.290 rows=342 loop...
Execution Time: 4.239 ms	Index Cond: ((full_name)::text ~~ '%Giang'::text)

a) Trước khi có index

QUERY PLAN	
text	Bitmap Heap Scan on hoc_vien (cost=41.94..615.27 rows=598 width=76) (actual time=0.342..0.664 rows=342 loops=1)
	Recheck Cond: ((full_name)::text ~~ '%Giang'::text)
	Heap Blocks: exact=196
	-> Bitmap Index Scan on idx_hoc_vien_full_name_trgm (cost=0.00..41.79 rows=598 width=0) (actual time=0.290..0.290 rows=342 loop...
	Index Cond: ((full_name)::text ~~ '%Giang'::text)
Planning Time: 0.262 ms	Planning Time: 0.262 ms
Execution Time: 0.714 ms	Execution Time: 0.714 ms

b) Sau khi có index

CÂU 10:Danh sách các lớp tại phòng 201 kết thúc trong 3 tháng đầu năm 2025

```
SELECT class_id , class_name , ket_thuc
FROM clazz
WHERE room = 201
AND ket_thuc BETWEEN '2025-01-31' AND '2025-03-31';
```

Phân tích hiệu năng: Câu này có chi phí là **198.5**, ta nhận thấy ở điều kiện WHERE có dùng AND nên ta có thể nghĩ đến việc cải thiện bằng cách thêm index vô bảng **clazz**.

QUERY PLAN
text
Seq Scan on clazz (cost=0.00..198.50 rows=1 width=33) (actual time=0.050..0.336 rows=4 loops=1)
Filter: ((ket_thuc >= '2025-01-31'::date) AND (ket_thuc <= '2025-03-31'::date) AND (room = 201))
Rows Removed by Filter: 996
Planning Time: 0.247 ms
Execution Time: 0.374 ms

Hình 4.19: Trước khi dùng index

Index trên cột của bảng clazz	Cost
(room)	59.57
(ket_thuc)	152.21
(room) , (ket_thuc)	13.64
(ket_thuc , end)	9.30
(room, ket_thuc)	8.30

Từ thống kê ở trên ta có thể kết luận là đánh index (**room,ket_thuc**) trên bảng clazz sẽ tối ưu hiệu năng nhất và ta còn rút ra là khi đánh multicolumn index thì nên ưu tiên các cột có điều kiện = trước (room) sau đó tới các cột điều kiện khoảng.

4.2 Câu 11-20: Hoàng Đức Anh

Danh sách các câu truy vấn

11. Tính phần trăm tham gia lớp học (điểm danh) của học sinh có student_id là 18282
 12. Học sinh có điểm tổng kết cao nhất trong lớp có class_id = 1000
 13. Danh sách lớp 'O' có giờ học vào buổi tối (từ 18h)
 14. Danh sách các học viên không công 1 đồng cho tư bản (không có lớp)
 15. Danh sách lớp có ít học sinh nhất
 16. Tạo view hiển thị thông tin giáo viên cùng đánh giá trung bình về giáo viên
 17. Tính tổng học phí của học sinh có student_id = 855
 18. Tính phần trăm tham gia lớp học (điểm danh) của học sinh có student_id là 18282
 19. Danh sách học sinh có sự tiến bộ trong học tập tại các lớp năm 2024
 20. Danh sách các giáo viên có tên “Anh” và tổng số lớp có tiết Chủ Nhật mà họ dạy trong năm 2023
-

Câu 11: Tính phần trăm tham gia lớp học (điểm danh) của học sinh có student_id là 18282

Dùng hàm COUNT và FILTER

```
SELECT
    a.student_id,
    COUNT(*) FILTER (WHERE a.status = '1') AS di_hoc,
    COUNT(a.attendance_date) AS tong_so_buoi,
    ROUND(
        (COUNT(*) FILTER (WHERE a.status = '1'))::DECIMAL / COUNT(a.
        id_attend)::DECIMAL * 100),
        2
    ) AS phan_tram_tham_gia
FROM attendance AS a
WHERE a.student_id = 18282
GROUP BY student_id;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm COUNT(*) và FILTER để đếm có điều kiện (được tối ưu hóa trong PostgreSQL). Nó đếm tất cả các hàng (*) nhưng chỉ những hàng nào thỏa mãn điều kiện trong mệnh đề FILTER (WHERE a.status = '1') mới được tính vào tổng. Câu truy vấn có chi phí là 2822.06, execution time = 13.805 ms trước khi thêm index. Sau khi thêm index bên dưới, chi phí giảm xuống ít hơn nhiều chỉ còn 55.62, execution time = 0.594 ms.

```
CREATE INDEX idx_attendance_student_id ON attendance (student_id);
```

QUERY PLAN text	
1	GroupAggregate (cost=0.00..2822.06 rows=1 width=52) (actual time=13.769..13.770 rows=1 loops=1)
2	-> Seq Scan on attendance a (cost=0.00..2821.93 rows=14 width=10) (actual time=0.018..13.747 rows=22 loop...
3	Filter: (student_id = 18282)
4	Rows Removed by Filter: 149492
5	Planning Time: 0.136 ms
6	Execution Time: 13.805 ms

a) Trước khi có index

QUERY PLAN text	
1	GroupAggregate (cost=4.40..55.62 rows=1 width=52) (actual time=0.267..0.268 rows=1 loops=1)
2	-> Bitmap Heap Scan on attendance a (cost=4.40..55.49 rows=14 width=10) (actual time=0.244..0.248 rows=22 loops=1)
3	Recheck Cond: (student_id = 18282)
4	Heap Blocks: exact=2
5	-> Bitmap Index Scan on idx_attendance_student_id (cost=0.00..4.40 rows=14 width=0) (actual time=0.216..0.216 rows=22 l...
6	Index Cond: (student_id = 18282)
7	Planning Time: 0.929 ms
8	Execution Time: 0.594 ms

b) Sau khi có index

Câu 12: Học sinh có điểm tổng kết cao nhất trong lớp có class_id = 1000

Sử dụng bảng CTE để tính điểm trước, sau đó dùng MAX

```
WITH final_grade AS (
    SELECT e.student_id, hv.full_name, e.class_id,
    ROUND(
        (e.minitest1 + e.minitest2 + e.minitest3 + e.minitest4) /
    4 * 0.40
        + e.midterm * 0.30
        + e.final * 0.30
    ,2) AS calculated_final_grade
    FROM enrollments e
    JOIN hoc_vien hv ON e.student_id = hv.student_id
    WHERE e.class_id = 1000
)
SELECT
    cg.student_id,
    cg.full_name,
    cg.calculated_final_grade
FROM final_grade cg
WHERE cg.calculated_final_grade = (
    SELECT MAX(calculated_final_grade)
    FROM final_grade
);
```

Phân tích hiệu năng: Cách này sẽ tính toán điểm tổng kết cho mỗi học sinh trong lớp 1000 trong một CTE, sau đó tiến hành select và lọc ra học sinh từ CTE. Ta đưa ra điều kiện calculated_final_grade lớn nhất dùng MAX. Câu truy vấn ban đầu có chi phí là 455.57, execution time là 1.653 ms. Sau khi áp dụng index ở cột class_id của bảng enrollments, truy vấn được cải thiện đáng kể khi chi phí là 121.79 và execution time là 0.720 ms.

```
CREATE INDEX idx_enrollments_class_id ON enrollments (class_id);
```

QUERY PLAN	
1	CTE Scan on calculatedgrades cg (cost=455.26..455.57 rows=1 width=134) (actual time=1.228..1.229 rows=1 loops=1)
2	Filter: (calculated_final_grade = (InitPlan 2).col1)
3	Rows Removed by Filter: 25
4	CTE calculatedgrades
5	-> Nested Loop (cost=0.29..454.93 rows=14 width=57) (actual time=1.013..1.125 rows=26 loops=1)
6	-> Seq Scan on enrollments e (cost=0.00..342.31 rows=14 width=44) (actual time=0.985..0.988 rows=26 loops=1)
7	Filter: (class_id = 1000)
8	Rows Removed by Filter: 14959
9	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..8.02 rows=1 width=21) (actual time=0.004..0.004 rows=1 loops=1)
10	Index Cond: (student_id = e.student_id)
11	InitPlan 2
12	-> Aggregate (cost=0.32..0.33 rows=1 width=32) (actual time=0.207..0.207 rows=1 loops=1)
13	-> CTE Scan on calculatedgrades (cost=0.00..0.28 rows=14 width=32) (actual time=0.001..0.118 rows=26 loops=1)
14	Planning Time: 1.235 ms
15	Execution Time: 1.653 ms

a) Trước khi có index

QUERY PLAN	
1	CTE Scan on calculatedgrades cg (cost=121.48..121.79 rows=1 width=134) (actual time=0.242..0.243 rows=1 loops=1)
2	Filter: (calculated_final_grade = (InitPlan 2).col1)
3	Rows Removed by Filter: 25
4	CTE calculatedgrades
5	-> Nested Loop (cost=0.57..121.15 rows=14 width=57) (actual time=0.042..0.210 rows=26 loops=1)
6	-> Index Scan using idx_enrollments_class_id on enrollments e (cost=0.29..8.53 rows=14 width=44) (actual time=0.020..0.028 rows=26 loops=1)
7	Index Cond: (class_id = 1000)
8	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..8.02 rows=1 width=21) (actual time=0.005..0.005 rows=1 loops=26)
9	Index Cond: (student_id = e.student_id)
10	InitPlan 2
11	-> Aggregate (cost=0.32..0.33 rows=1 width=32) (actual time=0.189..0.189 rows=1 loops=1)
12	-> CTE Scan on calculatedgrades (cost=0.00..0.28 rows=14 width=32) (actual time=0.000..0.179 rows=26 loops=1)
13	Planning Time: 0.693 ms
14	Execution Time: 0.720 ms

b) Sau khi có index

Câu 13: Danh sách các lớp 'O' có giờ học buổi tối (từ 18h)

Cách 1 : Sử dụng NOT IN và AND

```
SELECT c.class_name, s.start_time, s.end_time, s.days, c.room
FROM schedule AS s
JOIN clazz AS c ON s.class_id = c.class_id
JOIN class_type AS ct ON c.type_id = ct.type_id
WHERE ct.code NOT IN ('M', 'I', 'E') AND s.start_time >= '18:00:00'
ORDER BY s.start_time, c.class_name;
```

Phân tích hiệu năng: Cách đầu tiên sẽ lấy start_time và end_time từ bảng schedule sau đó join với bảng clazz (lấy class_name) và bảng class_type (lọc ra lớp mã 'O'). Ở bước lọc, do số mã lớp của trung tâm chỉ có 4, ta tiến hành dùng NOT IN với thuộc tính ct.code, chọn ra các mã lớp không phải mã 'O', sau đó dùng NOT IN để lọc được mã 'O'. Cuối cùng là thêm điều kiện start_time >= '18:00:00'. Truy vấn trên chưa thực sự tối ưu với chi phí 250.39, sử dụng Seq Scan và execution time = 1.261 ms.

QUERY PLAN text	
1	Sort (cost=249.91..250.39 rows=193 width=99) (actual time=1.200..1.205 rows=51 loops=1)
2	Sort Key: s.start_time, c.class_name
3	Sort Method: quicksort Memory: 31kB
4	-> Nested Loop (cost=28.11..242.59 rows=193 width=99) (actual time=0.284..1.011 rows=51 loops=1)
5	-> Hash Join (cost=27.95..236.59 rows=196 width=103) (actual time=0.265..0.900 rows=196 loops=1)
6	Hash Cond: (c.class_id = s.class_id)
7	-> Seq Scan on clazz c (cost=0.00..206.00 rows=1000 width=44) (actual time=0.017..0.359 rows=1000 loops=1)
8	-> Hash (cost=25.50..25.50 rows=196 width=67) (actual time=0.234..0.234 rows=196 loops=1)
9	Buckets: 1024 Batches: 1 Memory Usage: 28kB
10	-> Seq Scan on schedule s (cost=0.00..25.50 rows=196 width=67) (actual time=0.012..0.183 rows=196 loops=1)
11	Filter: (start_time >= '18:00:00':time without time zone)
12	Rows Removed by Filter: 804
13	-> Memoize (cost=0.16..0.23 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=196)
14	Cache Key: c.type_id
15	Cache Mode: logical
16	Hits: 192 Misses: 4 Evictions: 0 Overflows: 0 Memory Usage: 1kB
17	-> Index Scan using class_type_pkey on class_type ct (cost=0.15..0.22 rows=1 width=4) (actual time=0.005..0.005 rows=1)
18	Index Cond: (type_id = c.type_id)
19	Filter: (code <> ALL ('M','I','E'))
20	Rows Removed by Filter: 1
21	Planning Time: 0.523 ms
22	Execution Time: 1.261 ms

a) Query plan

Cách 2: Sử dụng Index và không dùng NOT IN

```
SELECT c.class_name, s.start_time, s.end_time, s.days, c.room
FROM schedule AS s
JOIN clazz AS c ON s.class_id = c.class_id
JOIN class_type AS ct ON c.type_id = ct.type_id
WHERE s.start_time >= '18:00:00'
AND ct.code = 'O'
ORDER BY s.start_time, c.class_name;
```

Giải thích: Do ở cách đầu tiên, query plan sử dụng Seq Scan ở bảng schedule. Hơn nữa, cách 1 dùng NOT operator nên index sẽ không thực sự có hiệu quả. Vì vậy ở cách 2, ta sẽ không sử dụng NOT IN nữa, sử dụng logic trực tiếp(ct.code = O) để có thể sử dụng index phát huy hiệu quả nhất. Kết hợp tìm kiếm trong khoảng \geq để lọc dữ liệu, sử dụng index cho schedule tại start_time là hợp lý.

```
CREATE INDEX idx_schedule_start_time ON schedule (start_time);
```

Phân tích hiệu năng: Cách 2 tối ưu hơn do giảm được chi phí xuống 233.94. Tuy nhiên ta thấy việc dùng index tại start_time không thực sự phát huy hiệu quả (query plan không hề sử dụng) do số bản ghi khác nhau của start_time khá nhỏ : 16.

```
central_management_db=# select distinct start_time from schedule;
start_time
-----
16:30:00
13:30:00
16:00:00
09:30:00
17:00:00
17:30:00
07:00:00
19:00:00
09:00:00
15:00:00
18:00:00
08:00:00
07:30:00
14:00:00
19:30:00
14:30:00
(16 rows)
```

a) start_time

QUERY PLAN	
	text
1	Sort (cost=233.93..233.94 rows=1 width=99) (actual time=1.234..1.237 rows=51 loops=1)
2	Sort Key: s.start_time, c.class_name
3	Sort Method: quicksort Memory: 31kB
4	-> Nested Loop (cost=0.44..233.92 rows=1 width=99) (actual time=0.107..1.073 rows=51 loops=1)
5	-> Nested Loop (cost=0.16..232.09 rows=5 width=40) (actual time=0.067..0.694 rows=250 loops=1)
6	-> Seq Scan on clazz c (cost=0.00..206.00 rows=1000 width=44) (actual time=0.023..0.276 rows=1000 loops=1)
7	-> Memoize (cost=0.16..0.23 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=1000)
8	Cache Key: c.type_id
9	Cache Mode: logical
10	Hits: 996 Misses: 4 Evictions: 0 Overflows: 0 Memory Usage: 1kB
11	-> Index Scan using class_type_pkey on class_type ct (cost=0.15..0.22 rows=1 width=4) (actual time=0.010..0.010 rows=1)
12	Index Cond: (type_id = c.type_id)
13	Filter: (code = 'O':bpchar)
14	Rows Removed by Filter: 1
15	-> Index Scan using schedule_class_id_key on schedule s (cost=0.28..0.37 rows=1 width=67) (actual time=0.001..0.001 rows=1)
16	Index Cond: (class_id = c.class_id)
17	Filter: (start_time >= '18:00:00':time without time zone)
18	Rows Removed by Filter: 1
19	Planning Time: 1.045 ms
20	Execution Time: 1.284 ms

b) Sau khi có index

Câu 14: Danh sách học sinh không đăng ký lớp học nào

Cách 1: Sử dụng NOT EXIST và SUB QUERRY

```
SELECT hv.student_id, hv.full_name
FROM hoc_vien AS hv
WHERE NOT EXISTS (
    SELECT e.student_id
    FROM enrollments AS e
    WHERE e.student_id = hv.student_id
);
```

Phân tích hiệu năng: Để thực hiện, ta viết một truy vấn con chọn ra student_id từ bảng enrollments với điều kiện nó tham chiếu tới student_id trên bảng hoc_vien ở truy vấn ngoài. Như vậy, tất cả bản ghi chung của cả 2 bảng được chọn, truy vấn con sẽ kiểm tra sự tồn tại của một bản ghi đăng ký cho từng học sinh. Ở truy vấn ngoài, sử dụng WHERE NOT EXIST để lọc các bản ghi có lớp học để từ đó lấy được danh sách các học sinh chưa đăng ký lớp nào. Truy vấn có chi phí 1153.65

	QUERY PLAN text
1	Hash Right Anti Join (cost=721.00..1153.65 rows=9387 width=21) (actual time=25.609..29.326 rows=9387 loops=1)
2	Hash Cond: (e.student_id = hv.student_id)
3	-> Seq Scan on enrollments e (cost=0.00..304.85 rows=14985 width=4) (actual time=0.039..2.714 rows=14985 loop...
4	-> Hash (cost=471.00..471.00 rows=20000 width=21) (actual time=12.614..12.616 rows=20000 loops=1)
5	Buckets: 32768 Batches: 1 Memory Usage: 1301kB
6	-> Seq Scan on hoc_vien hv (cost=0.00..471.00 rows=20000 width=21) (actual time=0.090..5.362 rows=20000 l...
7	Planning Time: 3.840 ms
8	Execution Time: 30.742 ms

a) Querry plan

Cách 2: Sử dụng NOT IN và SUB QUERRY

```
SELECT hv.student_id, hv.full_name
FROM hoc_vien AS hv
WHERE hv.student_id NOT IN (
    SELECT student_id
    FROM enrollments
);
```

Phân tích hiệu năng: Cách 2 sử dụng truy vấn con không tương quan, thực hiện trước khi truy vấn chính bắt đầu, do student_id và điểm số ở bảng enrollment là NOT NULL nên tất cả bản ghi ở bảng trên đều là các học sinh đã đăng ký lớp. Vì vậy ở truy vấn ngoài chỉ cần thêm điều kiện hv.student_id NOT IN đối với truy vấn con ta sẽ lọc được tất cả các student_id ở truy vấn con (các bản ghi đã có lớp đăng ký). Có thể thấy cách này tối ưu hơn do chi phí thực thi đã giảm đi đáng kể xuống 863.31. Tuy nhiên cả 2 cách đều sử dụng NOT operator, nên SQL engine sẽ chạy chương trình không sử dụng index nữa nên không thể cải thiện bằng index.

QUERY PLAN text	
1	Seq Scan on hoc_vien hv (cost=342.31..863.31 rows=10000 width=21) (actual time=12.645..27.288 rows=9387 loo...
2	Filter: (NOT (ANY (student_id = (hashed SubPlan 1).col1)))
3	Rows Removed by Filter: 10613
4	SubPlan 1
5	-> Seq Scan on enrollments (cost=0.00..304.85 rows=14985 width=4) (actual time=0.021..3.838 rows=14985 loo...
6	Planning Time: 0.217 ms
7	Execution Time: 28.608 ms

a) Querry plan

Câu 15: Danh sách lớp có ít học sinh nhất

Cách 1: Sử dụng <= ALL

```
SELECT e.class_id, COUNT(DISTINCT e.student_id) AS total_students
FROM enrollments AS e
GROUP BY e.class_id
HAVING
    COUNT(DISTINCT e.student_id) <= ALL (
        SELECT COUNT(DISTINCT student_id)
        FROM enrollments
        GROUP BY class_id
    );
```

Phân tích hiệu năng: Truy vấn sử dụng count distinct và sub querry để đếm số bản ghi khác nhau để tổng hợp tổng số học sinh trong 1 lớp, điều kiện \leq ALL , điều đó có nghĩa total_students phải nhỏ hơn hoặc bằng giá trị nhỏ nhất (MIN) trong tập hợp kết quả của sub querry. Do truy vấn còn sử dụng seq scan trên enrollment \Rightarrow ta đề xuất tạo một index kết hợp trên enrollments để cải thiện hiệu suất ở cách 2.

```
CREATE INDEX idx_enrollments_class_student ON enrollments (class_id,
    student_id);
```

	QUERY PLAN
1	text
1	GroupAggregate (cost=2688.30..65773.25 rows=486 width=12) (actual time=30.693..36.665 rows=35 loops=1)
2	Group Key: e.class_id
3	Filter: (ALL (count(DISTINCT e.student_id) <= (SubPlan 1).col1))
4	Rows Removed by Filter: 938
5	-> Sort (cost=1344.15..1381.61 rows=14985 width=8) (actual time=20.627..21.185 rows=14985 loops=1)
6	Sort Key: e.class_id, e.student_id
7	Sort Method: quicksort Memory: 736kB
8	-> Seq Scan on enrollments e (cost=0.00..304.85 rows=14985 width=8) (actual time=0.793..15.205 rows=1498...
9	SubPlan 1
10	-> Materialize (cost=1344.15..1471.13 rows=973 width=12) (actual time=0.008..0.012 rows=40 loops=973)
11	-> GroupAggregate (cost=1344.15..1466.27 rows=973 width=12) (actual time=7.449..9.712 rows=973 loops=1)
12	Group Key: enrollments.class_id
13	-> Sort (cost=1344.15..1381.61 rows=14985 width=8) (actual time=7.430..7.971 rows=14985 loops=1)
14	Sort Key: enrollments.class_id, enrollments.student_id
15	Sort Method: quicksort Memory: 736kB
16	-> Seq Scan on enrollments (cost=0.00..304.85 rows=14985 width=8) (actual time=0.039..2.785 rows=...
17	Planning Time: 4.078 ms
18	Execution Time: 39.179 ms

a) Trước khi có index

Cách 2: Sử dụng MIN và SUB QUERRY

```

SELECT e.class_id, COUNT(DISTINCT e.student_id) AS total_students
FROM enrollments AS e
GROUP BY e.class_id
HAVING
    COUNT(DISTINCT e.student_id) = (
        SELECT MIN(student_count)
        FROM (
            SELECT COUNT(DISTINCT student_id) AS student_count
            FROM enrollments
            GROUP BY class_id
        ) AS subquery_counts
    );

```

Phân tích hiệu năng: Cách 2 sử dụng MIN kèm index kết hợp đã cải thiện hiệu suất đáng kể cho truy vấn. Truy vấn con trong FROM (inner subquery subquery_counts): Tính số lượng học sinh cho mỗi khóa học. Truy vấn con thứ hai (SELECT MIN(student_count)): Tìm ra giá trị số học sinh ít nhất từ kết quả của truy vấn con đầu tiên. Cách này trực tiếp và đơn giản hơn về mặt tư duy. Chi phí cũng được giảm đi đáng kể chỉ còn 978.03

QUERY PLAN text	
1	GroupAggregate (cost=494.17..978.03 rows=5 width=12) (actual time=5.360..7.203 rows=35 loops=1)
2	Group Key: e.class_id
3	Filter: (count(DISTINCT e.student_id) = (InitPlan 1).col1)
4	Rows Removed by Filter: 938
5	InitPlan 1
6	-> Aggregate (cost=493.88..493.89 rows=1 width=8) (actual time=5.135..5.136 rows=1 loops=1)
7	-> GroupAggregate (cost=0.29..481.71 rows=973 width=12) (actual time=0.026..5.087 rows=973 loops=1)
8	Group Key: enrollments.class_id
9	-> Index Only Scan using idx_enrollments_class_student on enrollments (cost=0.29..397.06 rows=14985 width=8) (actual time=0.019..3.824 rows=149...
10	Heap Fetches: 0
11	-> Index Only Scan using idx_enrollments_class_student on enrollments e (cost=0.29..397.06 rows=14985 width=8) (actual time=0.171..1.058 rows=14985 loo...
12	Heap Fetches: 0
13	Planning Time: 2.976 ms
14	Execution Time: 7.607 ms

a) Sau khi có index

Câu 16: Tạo view hiển thị thông tin giáo viên cùng đánh giá trung bình về giáo viên

```
CREATE VIEW teacher_rating AS
SELECT f.teacher_id, t.full_name,
       ROUND(AVG(f.teacher_rate), 2) AS average_teacher_rate
FROM feedback AS f
JOIN teacher t
USING (teacher_id)
GROUP BY f.teacher_id, t.full_name
ORDER BY AVG(f.teacher_rate) DESC ;
```

Phân tích hiệu năng: truy vấn này tạo một view để hiển thị danh sách các giáo viên cùng với điểm đánh giá trung bình của họ. Nó nối bảng feedback với bảng teacher qua teacher_id để lấy tên giáo viên. Sau đó, nhóm dữ liệu theo từng giáo viên và tính điểm trung bình (AVG) từ cột teacher_rate trong bảng feedback, đồng thời làm tròn 2 chữ số sau dấu phẩy. Cuối cùng, kết quả được sắp xếp theo điểm trung bình giảm dần.

Câu 17: Tính tổng học phí của học sinh có student_id = 855

Sử dụng SUM, JOIN và WHERE

```
SELECT hv.student_id, hv.full_name, SUM(c.price) AS tuition_fees
FROM hoc_vien AS hv
JOIN enrollments AS e ON hv.student_id = e.student_id
JOIN clazz AS c ON e.class_id = c.class_id
WHERE hv.student_id = 855
GROUP BY hv.student_id, hv.full_name;
```

Phân tích hiệu năng: Kế hoạch truy vấn cho thấy PostgreSQL tận dụng tối đa các khóa chính hiện có của các bảng hoc_vien, enrollments, và clazz để thực hiện các phép tìm kiếm và JOIN rất nhanh chóng thông qua các Index Scan và Nested Loop Join thay vì SEQ SCAN. Do đó, truy vấn này đã đạt được mức tối ưu và không cần thêm index nào khác để cải thiện hiệu suất. Cụ thể, chi phí cần thiết cho truy vấn chỉ ở mức 58.55 và execution time = 0.130 ms.

QUERY PLAN	
text	
1	GroupAggregate (cost=0.85..58.55 rows=1 width=29) (actual time=0.068..0.069 rows=1 loops=1)
2	-> Nested Loop (cost=0.85..58.52 rows=6 width=25) (actual time=0.038..0.062 rows=6 loops=1)
3	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..8.30 rows=1 width=21) (actual time=0.016..0.017 rows=1 loops=1)
4	Index Cond: (student_id = 855)
5	-> Nested Loop (cost=0.56..50.16 rows=6 width=8) (actual time=0.020..0.043 rows=6 loops=1)
6	-> Index Only Scan using enrollments_pkey on enrollments e (cost=0.29..4.39 rows=6 width=8) (actual time=0.011..0.012 rows=6 loops=1)
7	Index Cond: (student_id = 855)
8	Heap Fetches: 0
9	-> Index Scan using clazz_pkey on clazz c (cost=0.28..7.63 rows=1 width=8) (actual time=0.004..0.004 rows=1 loops=6)
10	Index Cond: (class_id = e.class_id)
11	Planning Time: 1.970 ms
12	Execution Time: 0.130 ms

a) Querry plan

Câu 18: Tính doanh thu mà các giáo viên mang lại cho trung tâm trong năm 2023

Dùng CTE - bảng tạm

```

WITH stu_per_class AS (
    SELECT e.class_id, COUNT(DISTINCT e.student_id) AS num_students
    FROM enrollments e
    GROUP BY e.class_id
),
ClassIncome AS (
    SELECT
        c.teacher_id,
        (c.price * spc.num_students) AS class_total_income
    FROM clazz c
    JOIN stu_per_class spc using(class_id)
    WHERE
        EXTRACT(YEAR FROM c.khai_giang) = 2023
        AND EXTRACT(YEAR FROM c.ket_thuc) = 2023
)
SELECT t.teacher_id, t.full_name, SUM(CI.class_total_income) AS
    total_income
FROM teacher t
JOIN ClassIncome CI using(teacher_id)
GROUP BY t.full_name, t.teacher_id
ORDER BY total_income DESC;

```

Phân tích hiệu năng: Cách đầu tiên dùng bảng tạm để tính số học viên trong 1 lớp, sau đó tạo classIncome để tính tổng tiền học từ 1 lớp = giá tiền lớp đó * số học viên 1 lớp. Cuối cùng dùng SUM để gộp hết tiền học và group by teacher_id sẽ ra tiền thu nhập của giáo viên trong năm 2023. Cách này khá phức tạp và dài, chi phí lên tới 718.28

QUERY PLAN text	
1	Sort (cost=718.27..718.28 rows=1 width=53) (actual time=3.845..3.851 rows=191 loops=1)
2	Sort Key: (sum((c.price * (count(DISTINCT e.student_id)))))) DESC
3	Sort Method: quicksort Memory: 33kB
4	-> GroupAggregate (cost=718.24..718.26 rows=1 width=53) (actual time=3.684..3.776 rows=191 loops=1)
5	Group Key: t.teacher_id
6	-> Sort (cost=718.24..718.24 rows=1 width=33) (actual time=3.676..3.688 rows=246 loops=1)
7	Sort Key: t.teacher_id
8	Sort Method: quicksort Memory: 37kB
9	-> Nested Loop (cost=216.57..718.23 rows=1 width=33) (actual time=0.947..3.619 rows=246 loops=1)
10	-> Merge Join (cost=216.29..709.90 rows=1 width=16) (actual time=0.934..3.276 rows=246 loops=1)
11	Merge Cond: (c.class_id = e.class_id)
12	-> Sort (cost=216.01..216.01 rows=1 width=12) (actual time=0.885..0.896 rows=252 loops=1)
13	Sort Key: c.class_id
14	Sort Method: quicksort Memory: 32kB
15	-> Seq Scan on clazz c (cost=0.00..216.00 rows=1 width=12) (actual time=0.028..0.830 rows=252 loops=1)
16	Filter: ((EXTRACT(year FROM khai_giang) = '2023':numeric) AND (EXTRACT(year FROM ket_thuc) = '2023':numeric))
17	Rows Removed by Filter: 748
18	-> GroupAggregate (cost=0.29..481.71 rows=973 width=12) (actual time=0.038..2.283 rows=972 loops=1)

a) Querry

Dùng hàm SUM

```

SELECT t.teacher_id, t.full_name, SUM(c.price) AS total_income
FROM teacher t
JOIN clazz c ON t.teacher_id = c.teacher_id
JOIN enrollments e ON c.class_id = e.class_id
WHERE EXTRACT(YEAR FROM c.khai_giang) = 2023
    AND EXTRACT(YEAR FROM c.ket_thuc) = 2023
GROUP BY t.full_name, t.teacher_id
ORDER BY total_income DESC;

```

Phân tích hiệu năng: Câu truy vấn tính tiền mà giáo viên mang lại cho trung tâm, vì vậy ta sử dụng hàm sum. Tuy nhiên, nếu chỉ join 2 bảng teacher và clazz sẽ ra tổng thu nhập giáo viên mang lại cho trung tâm theo định giá từng lớp. Mà mỗi lớp có nhiều học sinh nên để đúng yêu cầu truy vấn ta cần join thêm bảng enrollment rồi tính SUM(cprice) theo từng teacherid. Câu truy vấn có chi phí là 229.91. Cách này đơn giản hơn, tối ưu hơn.

QUERY PLAN	
text	
1	Sort (cost=229.87..229.91 rows=15 width=25) (actual time=8.707..8.716 rows=176 loops=1)
2	Sort Key: (sum(c.price)) DESC
3	Sort Method: quicksort Memory: 32kB
4	-> GroupAggregate (cost=229.31..229.58 rows=15 width=25) (actual time=7.794..8.660 rows=176 loops=1)
5	Group Key: t.full_name
6	-> Sort (cost=229.31..229.35 rows=15 width=21) (actual time=7.777..8.062 rows=3910 loops=1)
7	Sort Key: t.full_name
8	Sort Method: quicksort Memory: 253kB
9	-> Nested Loop (cost=0.56..229.02 rows=15 width=21) (actual time=0.123..2.349 rows=3910 loops=1)
10	-> Nested Loop (cost=0.27..224.32 rows=1 width=25) (actual time=0.037..1.122 rows=252 loops=1)
11	-> Seq Scan on clazz c (cost=0.00..216.00 rows=1 width=12) (actual time=0.024..0.660 rows=252 loops=1)
12	Filter: ((EXTRACT(year FROM khai_giang) = '2023'::numeric) AND (EXTRACT(year FROM ket_thuc) = '2023'::numeric))
13	Rows Removed by Filter: 748
14	-> Index Scan using teacher_pkey on teacher t (cost=0.27..8.29 rows=1 width=21) (actual time=0.001..0.001 rows=1 loops=252)
15	Index Cond: (teacher_id = c.teacher_id)
16	-> Index Only Scan using idx_enrollments_class_student on enrollments e (cost=0.29..4.55 rows=15 width=4) (actual time=0.002..0.003 rows=15 loops=252)
17	Index Cond: (class_id = c.class_id)
18	Heap Fetches: 0
19	Planning Time: 0.529 ms
20	Execution Time: 8.827 ms

a) Querry

Câu 19: Danh sách học sinh có sự tiến bộ trong học tập tại các lớp năm 2024

Cách 1: Sử dụng phép so sánh trực tiếp

```

SELECT hv.student_id, hv.full_name, c.class_name
FROM hoc_vien AS hv
JOIN enrollments AS e ON hv.student_id = e.student_id
JOIN clazz AS c ON e.class_id = c.class_id
WHERE
    khai_giang >= '2024-01-01'
    AND ket_thuc <= '2024-12-31'
    AND e.final > e.midterm
    AND e.minitest1 <= e.minitest2
    AND e.minitest2 <= e.minitest3
    AND e.minitest3 <= e.minitest4;

```

Phân tích hiệu năng: Điều kiện xét sự tiến bộ : các minitest có điểm tăng dần và điểm cuối kì cao hơn điểm giữa kì. Để áp dụng điều kiện trên ta sử dụng AND để ghép nhiều điều kiện với nhau. Đồng thời kết hợp xét khoảng để tìm các lớp học sinh đó học trong năm 2024. Truy vấn cho ra kết quả có chi phí 923.16, theo đánh giá thì đây là chi phí hơi lớn.

	QUERY PLAN text
1	Nested Loop (cost=215.07..923.16 rows=56 width=53) (actual time=0.543..4.591 rows=81 loops=1)
2	-> Hash Join (cost=214.79..669.97 rows=56 width=36) (actual time=0.533..4.236 rows=81 loops=1)
3	Hash Cond: (e.class_id = c.class_id)
4	-> Seq Scan on enrollments e (cost=0.00..454.70 rows=185 width=8) (actual time=0.027..3.677 rows=322 loops=1)
5	Filter: ((final > midterm) AND (minitest1 <= minitest2) AND (minitest2 <= minitest3) AND (minitest3 <= minitest4))
6	Rows Removed by Filter: 14663
7	-> Hash (cost=211.00..211.00 rows=303 width=36) (actual time=0.479..0.479 rows=245 loops=1)
8	Buckets: 1024 Batches: 1 Memory Usage: 25kB
9	-> Seq Scan on clazz c (cost=0.00..211.00 rows=303 width=36) (actual time=0.008..0.437 rows=245 loops=1)
10	Filter: ((khai_giang >= '2024-01-01'::date) AND (ket_thuc <= '2024-12-31'::date))
11	Rows Removed by Filter: 755
12	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..4.52 rows=1 width=21) (actual time=0.004..0.004 rows=1 loops=1)
13	Index Cond: (student_id = e.student_id)
14	Planning Time: 0.828 ms
15	Execution Time: 4.643 ms

a) Query plan

Cách 2: Sử dụng EXTRACT

```

SELECT hv.student_id, hv.full_name, c.class_name
FROM hoc_vien AS hv
JOIN enrollments AS e ON hv.student_id = e.student_id
JOIN clazz AS c ON e.class_id = c.class_id
WHERE
    EXTRACT(YEAR FROM c.khai_giang) = 2024
    AND EXTRACT(YEAR FROM c.ket_thuc) = 2024
    AND e.final > e.midterm
    AND e.minitest1 <= e.minitest2
    AND e.minitest2 <= e.minitest3
    AND e.minitest3 <= e.minitest4;

```

Phân tích hiệu năng: Truy vấn sử dụng extract để trích xuất 1 phần cụ thể, cụ thể là năm 2024 khỏi một giá trị thời gian từ clazz. Truy vấn này đơn giản hơn và cũng hiệu quả hơn trong trường hợp này, với chi phí 253.73, thấp hơn so với cách thứ nhất.

QUERY PLAN text	
1	Nested Loop (cost=0.57..253.73 rows=1 width=53) (actual time=0.614..4.813 rows=81 loops=1)
2	-> Nested Loop (cost=0.29..249.21 rows=1 width=36) (actual time=0.598..4.426 rows=81 loops=1)
3	-> Seq Scan on clazz c (cost=0.00..216.00 rows=1 width=36) (actual time=0.026..0.431 rows=245 loops=1)
4	Filter: ((EXTRACT(year FROM khai_giang) = '2024'::numeric) AND (EXTRACT(year FROM ket_thuc) = '2024'::numeric))
5	Rows Removed by Filter: 755
6	-> Index Scan using idx_enrollments_class_student on enrollments e (cost=0.29..33.20 rows=1 width=8) (actual time=0.015..0.016 rows=0 loops=1)
7	Index Cond: (class_id = c.class_id)
8	Filter: ((final > midterm) AND (minitest1 <= minitest2) AND (minitest2 <= minitest3) AND (minitest3 <= minitest4))
9	Rows Removed by Filter: 15
10	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..4.52 rows=1 width=21) (actual time=0.004..0.004 rows=1 loops=81)
11	Index Cond: (student_id = e.student_id)
12	Planning Time: 2.883 ms
13	Execution Time: 5.160 ms

a) Query plan

Câu 20:Danh sách các giáo viên có tên “Anh” và tổng số lớp có tiết Chủ Nhật mà họ dạy trong năm 2023

Sử dụng ANY với mảng và COUNT DISTINCT

```

SELECT t.teacher_id, t.full_name, COUNT(DISTINCT c.class_id) AS
    total_Sundayclasses
FROM teacher t
JOIN clazz c ON t.teacher_id = c.teacher_id
JOIN schedule AS s ON c.class_id = s.class_id
WHERE
    EXTRACT(YEAR FROM c.khai_giang) = 2023
    AND EXTRACT(YEAR FROM c.ket_thuc) = 2023
    AND 'CN' = ANY(s.days)
    AND t.full_name LIKE '%Anh'
GROUP BY
    t.teacher_id, t.full_name
ORDER BY
    total_Sundayclasses DESC;

```

Phân tích hiệu năng: Vì kiểu dữ liệu của sdays là varchar(2) - một mảng có kích thước 3 ký tự nên ta sử dụng ANY để tìm và trả về một ký tự trong một mảng, cụ thể là CN. Nhóm theo giáo viên: GROUP BY t.teacher_id, t.full_name đảm bảo mỗi giáo viên xuất hiện một lần trong kết quả. Đếm lớp duy nhất: COUNT(DISTINCT class_id) loại bỏ các bản ghi trùng lặp của cùng một lớp (do join với schedule tạo ra bởi schedule có khóa chính là id_schedule không phải class_id), đếm đúng số lớp khác nhau. Cuối cùng, sử dụng t.full_name like "%Anh" để lọc ra giáo viên có tên kết thúc là Anh

	QUERY PLAN text	🔒
1	Sort (cost=233.09..233.09 rows=1 width=29) (actual time=14.773..14.774 rows=6 loops=1)	
2	Sort Key: (count(DISTINCT c.class_id)) DESC	
3	Sort Method: quicksort Memory: 25kB	
4	-> GroupAggregate (cost=233.06..233.08 rows=1 width=29) (actual time=14.451..14.454 rows=6 loops=1)	
5	Group Key: t.teacher_id	
6	-> Sort (cost=233.06..233.06 rows=1 width=25) (actual time=14.443..14.444 rows=7 loops=1)	
7	Sort Key: t.teacher_id, c.class_id	
8	Sort Method: quicksort Memory: 25kB	
9	-> Nested Loop (cost=0.55..233.05 rows=1 width=25) (actual time=4.827..14.364 rows=7 loops=1)	
10	-> Nested Loop (cost=0.27..224.71 rows=1 width=25) (actual time=2.066..13.027 rows=21 loops=1)	
11	-> Seq Scan on clazz c (cost=0.00..216.00 rows=1 width=8) (actual time=0.968..11.425 rows=252 loops=1)	
12	Filter: ((EXTRACT(year FROM khai_giang) = '2023'::numeric) AND (EXTRACT(year FROM ket_thuc) = '2023'::numeric))	
13	Rows Removed by Filter: 748	
14	-> Index Scan using teacher_pkey on teacher t (cost=0.27..8.29 rows=1 width=21) (actual time=0.006..0.006 rows=0 loops=1)	
15	Index Cond: (teacher_id = c.teacher_id)	
16	Filter: ((full_name)::text ~~ '%Anh'::text)	
17	Rows Removed by Filter: 1	
18	-> Index Scan using schedule_class_id_key on schedule s (cost=0.28..8.30 rows=1 width=4) (actual time=0.063..0.063 rows=1)	
19	Index Cond: (class_id = c.class_id)	
20	Filter: ('CN'::text = ANY ((days)::text[]))	51
21	Rows Removed by Filter: 1	

4.3 Câu 21-30: Vũ Anh

Danh sách các câu truy vấn

21. Tính tổng doanh thu theo loại lớp trong 6 tháng qua
 22. Tính trung bình điểm midterm của học viên theo từng giáo viên
 23. Danh sách sinh viên có điểm tổng kết dưới trung bình
 24. Trung bình điểm midterm trong quý 4 năm 2024 theo loại lớp
 25. Tổng số học viên vắng mặt trong quý 4 năm 2024
 26. Danh sách lớp có sĩ số thay đổi trong quý 3/2023
 27. View hiển thị các lớp đầy trong quý 2/2023
 28. View hiển thị các học sinh vắng mặt trong tháng 6/2024
 29. Thứ hạng điểm tổng kết của học viên trong mỗi lớp quý 2 năm 2024
 30. Thông tin chi tiết về lịch học của từng lớp trong tháng 05/2024
-

Câu 21: Tính tổng doanh thu theo loại lớp trong 6 tháng qua

Cách 1: Dùng SUM and JOIN

```
SELECT ct.describe, SUM(c.price * c.si_so) AS total_revenue
FROM class_type ct
JOIN clazz c ON ct.type_id = c.type_id
JOIN enrollments e ON c.class_id = e.class_id
WHERE e.enrollment_date >= '2024-12-13'
GROUP BY ct.describe;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm SUM để tính tổng doanh thu theo loại lớp trong 6 tháng qua. Chi phí là **730.38**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **428.39**.

Vì sao sử dụng index: Index được sử dụng để tăng hiệu quả việc lọc dữ liệu của hàm WHERE. Bằng phương pháp Bitmap Index Scan, index có thể giảm thiểu số hàng cần được xử lý trước khi JOIN.

```
CREATE INDEX idx_enrollments_enrollment_date ON enrollments(
    enrollment_date);
```

QUERY PLAN text	
1	GroupAggregate (cost=729.45..730.38 rows=4 width=40) (actual time=2.436..2.444 rows=4 loops=1)
2	Group Key: ct.describe
3	-> Sort (cost=729.45..729.67 rows=89 width=40) (actual time=2.424..2.428 rows=49 loops=1)
4	Sort Key: ct.describe
5	Sort Method: quicksort Memory: 30kB
6	-> Hash Join (cost=222.59..726.56 rows=89 width=40) (actual time=0.951..1.898 rows=49 loops=1)
7	Hash Cond: (c.type_id = ct.type_id)
8	-> Hash Join (cost=221.50..725.00 rows=89 width=12) (actual time=0.838..1.776 rows=49 loops=1)
9	Hash Cond: (e.class_id = c.class_id)
10	-> Seq Scan on enrollments e (cost=0.00..503.26 rows=89 width=4) (actual time=0.239..1.166 rows=49 loops=1)
11	Filter: (enrollment_date >= '2024-12-13':date)
12	Rows Removed by Filter: 15172
13	-> Hash (cost=209.00..209.00 rows=1000 width=16) (actual time=0.483..0.483 rows=1000 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 55kB
15	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=16) (actual time=0.014..0.341 rows=1000 loops=1)
16	-> Hash (cost=1.04..1.04 rows=4 width=36) (actual time=0.027..0.027 rows=4 loops=1)
17	Buckets: 1024 Batches: 1 Memory Usage: 9kB
18	-> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.020..0.021 rows=4 loops=1)
19	Planning Time: 1.531 ms
20	Execution Time: 2.533 ms

QUERY PLAN text	
1	GroupAggregate (cost=427.46..428.39 rows=4 width=40) (actual time=0.636..0.641 rows=4 loops=1)
2	Group Key: ct.describe
3	-> Sort (cost=427.46..427.68 rows=89 width=40) (actual time=0.628..0.630 rows=49 loops=1)
4	Sort Key: ct.describe
5	Sort Method: quicksort Memory: 30kB
6	-> Hash Join (cost=227.56..424.58 rows=89 width=40) (actual time=0.565..0.589 rows=49 loops=1)
7	Hash Cond: (c.type_id = ct.type_id)
8	-> Hash Join (cost=226.47..423.01 rows=89 width=12) (actual time=0.452..0.470 rows=49 loops=1)
9	Hash Cond: (e.class_id = c.class_id)
10	-> Bitmap Heap Scan on enrollments e (cost=4.97..201.27 rows=89 width=4) (actual time=0.022..0.030 rows=49 loops=1)
11	Recheck Cond: (enrollment_date >= 2024-12-13:date)
12	Heap Blocks: exact=6
13	-> Bitmap Index Scan on idx_enrollments_enrollment_date (cost=0.00..4.95 rows=89 width=0) (actual time=0.011..0.011 rows=49 loops=1)
14	Index Cond: (enrollment_date >= 2024-12-13:date)
15	-> Hash (cost=209.00..209.00 rows=1000 width=16) (actual time=0.407..0.407 rows=1000 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 55kB
17	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=16) (actual time=0.011..0.298 rows=1000 loops=1)
18	-> Hash (cost=1.04..1.04 rows=4 width=36) (actual time=0.062..0.062 rows=4 loops=1)
19	Buckets: 1024 Batches: 1 Memory Usage: 9kB
20	-> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.036..0.037 rows=4 loops=1)
21	Planning Time: 2.465 ms
22	Execution Time: 0.714 ms

a) Trước khi có index

b) Sau khi có index

Cách 2:Dùng CTE

```

WITH class_revenue AS (
    SELECT ct.describe, c.price * c.si_so AS revenue
    FROM class_type ct
    JOIN clazz c ON ct.type_id = c.type_id
    JOIN enrollments e ON c.class_id = e.class_id
    WHERE e.enrollment_date >= CURRENT_DATE - INTERVAL '6 months'
)
SELECT describe, SUM(revenue) AS total_revenue
FROM class_revenue
GROUP BY describe;

```

Phân tích hiệu năng: Truy vấn sử dụng 1 bảng CTE tên là class_revenue để tính doanh thu theo loại lớp trong 6 tháng qua. Truy vấn sử dụng cách tính thời gian linh hoạt hơn so với cách 1 với CURRENT_DATE - INTERVAL '6 months'. Chi phí là **699.58** khi chưa có index. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **186.94**(hiệu quả hơn nhiều so với cách 1).

Vì sao sử dụng index: Tương tự như cách 1, nhưng index enrollment_date ở đây sẽ giúp duyệt theo range scan, thay vì phải full table scan cả bảng enrollment để tìm thời gian thỏa mãn.

```

CREATE INDEX idx_enrollments_enrollment_date ON enrollments(
    enrollment_date);

```

QUERY PLAN	
	text
1	GroupAggregate (cost=699.35..699.58 rows=4 width=40) (actual time=2.737..2.739 rows=1 loops=1)
2	Group Key: ct.describe
3	-> Sort (cost=699.35..699.40 rows=19 width=40) (actual time=2.730..2.732 rows=2 loops=1)
4	Sort Key: ct.describe
5	Sort Method: quicksort Memory: 25kB
6	-> Nested Loop (cost=0.28..698.95 rows=19 width=40) (actual time=0.842..2.724 rows=2 loops=1)
7	Join Filter: (ct.type_id = c.type_id)
8	Rows Removed by Filter: 6
9	-> Nested Loop (cost=0.28..696.94 rows=19 width=12) (actual time=0.822..2.703 rows=2 loops=1)
10	-> Seq Scan on enrollments e (cost=0.00..579.37 rows=19 width=4) (actual time=0.802..2.679 rows=2 loops=1)
11	Filter: (enrollment_date >= (CURRENT_DATE - '6 mons':interval))
12	Rows Removed by Filter: 15219
13	-> Index Scan using clazz_pkey on clazz c (cost=0.28..6.19 rows=1 width=16) (actual time=0.009..0.009 rows=1 loops=1)
14	Index Cond: (class_id = e.class_id)
15	-> Materialize (cost=0.00..1.06 rows=4 width=36) (actual time=0.009..0.009 rows=4 loops=1)
16	-> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.014..0.014 rows=4 loops=1)
17	Planning Time: 0.270 ms
18	Execution Time: 2.796 ms

a) Trước khi có index

QUERY PLAN	
	text
1	GroupAggregate (cost=186.71..186.94 rows=4 width=40) (actual time=0.073..0.073 rows=1 loops=1)
2	Group Key: ct.describe
3	-> Sort (cost=186.71..186.76 rows=19 width=40) (actual time=0.068..0.069 rows=2 loops=1)
4	Sort Key: ct.describe
5	Sort Method: quicksort Memory: 25kB
6	-> Nested Loop (cost=0.28..186.31 rows=19 width=40) (actual time=0.053..0.057 rows=2 loops=1)
7	Join Filter: (ct.type_id = c.type_id)
8	Rows Removed by Filter: 6
9	-> Nested Loop (cost=4.71..184.30 rows=19 width=12) (actual time=0.034..0.037 rows=2 loops=1)
10	-> Bitmap Heap Scan on enrollments e (cost=4.44..66.73 rows=19 width=4) (actual time=0.022..0.023 rows=2 loops=1)
11	Recheck Cond: (enrollment_date >= (CURRENT_DATE - '6 mons':interval))
12	Heap Blocks: exact=1
13	-> Bitmap Index Scan on idx_enrollments_enrollment_date (cost=0.00..4.43 rows=19 width=0) (actual time=0.005..0.006 rows=2 loops=1)
14	Index Cond: (enrollment_date >= (CURRENT_DATE - '6 mons':interval))
15	-> Index Scan using clazz_pkey on clazz c (cost=0.28..6.19 rows=1 width=16) (actual time=0.006..0.006 rows=2 loops=1)
16	Index Cond: (class_id = e.class_id)
17	-> Materialize (cost=0.00..1.06 rows=4 width=36) (actual time=0.008..0.009 rows=4 loops=1)
18	-> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.010..0.011 rows=4 loops=1)
19	Planning Time: 0.377 ms
20	Execution Time: 0.120 ms

b) Sau khi có index

Câu 22: Tính trung bình điểm midterm của học viên theo từng giáo viên

Cách 1: Dùng AVG and JOIN

```
SELECT t.full_name , AVG(e.midterm) AS avg_midterm
FROM teacher t
JOIN clazz c ON t.teacher_id = c.teacher_id
JOIN enrollments e ON c.class_id = e.class_id
GROUP BY t.full_name;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm AVG để tính trung bình điểm midterm của học viên theo từng giáo viên, kết quả được gộp theo teacher.full_name. Chi phí là **867.19**. Câu truy vấn này đã tối ưu và không cần index do không có điều kiện WHERE, cũng như hàm AVG cần duyệt tất cả dữ liệu của bảng midterm nên không cần index để giảm dữ liệu.

Cách 2:Dùng subquery

```
SELECT t.full_name , avg_midterm
FROM teacher t
JOIN (
    SELECT c.teacher_id , AVG(e.midterm) AS avg_midterm
    FROM clazz c
    JOIN enrollments e ON c.class_id = e.class_id
    GROUP BY c.teacher_id
) e ON t.teacher_id = e.teacher_id;
```

Phân tích hiệu năng: Truy vấn sử dụng subquery để tính trung bình điểm midterm của học viên theo từng giáo viên, kết quả được gộp theo teacher.full_name. Chi phí là **831.93**(hiệu quả hơn so với cách 1). Tương tự như cách 1, truy vấn không cần index.

Câu 23: Tổng số học viên đăng ký lớp quý 2 năm 2023

Cách 1: Dùng COUNT and JOIN

```
SELECT c.class_name, COUNT(e.student_id) AS enrolled_count
FROM clazz c
JOIN enrollments e ON c.class_id = e.class_id
WHERE e.enrollment_date BETWEEN '2023-04-01' AND '2023-06-15'
GROUP BY c.class_name;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm COUNT để tính số học viên đăng ký 1 lớp quý 2 năm 2023, gộp theo clazz.class_name. Chi phí là **779.07**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **584.09**.

Vì sao sử dụng index: Index được sử dụng để hỗ trợ giảm bớt dữ liệu cần lọc trong cột enrollment_date. Bằng phương pháp Bitmap Index Scan, index giảm số hàng cần được xử lý trước hàm JOIN.

```
CREATE INDEX idx_enrollments_date ON enrollments (enrollment_date);
```

QUERY PLAN text	
1	HashAggregate (cost=770.39..779.07 rows=868 width=40) (actual time=4.356..4.380 rows=74 loops=1)
2	Group Key: c.class_name
3	Batches: 1 Memory Usage: 49kB
4	-> Hash Join (cost=221.50..765.43 rows=992 width=36) (actual time=1.174..3.933 rows=997 loops=1)
5	Hash Cond: (e.class_id = c.class_id)
6	-> Seq Scan on enrollments e (cost=0.00..541.32 rows=992 width=8) (actual time=0.355..2.779 rows=99...
7	Filter: ((enrollment_date >= '2023-04-01':date) AND (enrollment_date <= '2023-06-15':date))
8	Rows Removed by Filter: 14224
9	-> Hash (cost=209.00..209.00 rows=1000 width=36) (actual time=0.802..0.803 rows=1000 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 76kB
11	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.023..0.580 rows=100...
12	Planning Time: 0.856 ms
13	Execution Time: 4.478 ms

a) Trước khi có index

QUERY PLAN text	
1	HashAggregate (cost=575.41..584.09 rows=868 width=40) (actual time=1.986..2.010 rows=74 loops=1)
2	Group Key: c.class_name
3	Batches: 1 Memory Usage: 49kB
4	-> Hash Join (cost=239.95..570.45 rows=992 width=36) (actual time=1.026..1.609 rows=997 loops=1)
5	Hash Cond: (e.class_id = c.class_id)
6	-> Bitmap Heap Scan on enrollments e (cost=18.45..346.33 rows=992 width=8) (actual time=0.146..0.359 rows=997 loops=1)
7	Recheck Cond: ((enrollment_date >= '2023-04-01':date) AND (enrollment_date <= '2023-06-15':date))
8	Heap Blocks: exact=65
9	-> Bitmap Index Scan on idx_enrollments_date (cost=0.00..18.21 rows=992 width=0) (actual time=0.113..0.113 rows=99...
10	Index Cond: ((enrollment_date >= '2023-04-01':date) AND (enrollment_date <= '2023-06-15':date))
11	-> Hash (cost=209.00..209.00 rows=1000 width=36) (actual time=0.857..0.858 rows=1000 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 76kB
13	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.017..0.577 rows=1000 loops=1)
14	Planning Time: 5.244 ms
15	Execution Time: 2.109 ms

b) Sau khi có index

Cách 2:Dùng subquery

```

SELECT c.class_name , (
    SELECT COUNT(e.student_id)
    FROM enrollments e
    WHERE e.class_id = c.class_id
    AND e.enrollment_date BETWEEN '2023-04-01' AND '2023-06-30'
) AS total_enrolled
FROM clazz c
WHERE c.khai_giang BETWEEN '2023-04-01' AND '2023-06-30';

```

Phân tích hiệu năng: Câu truy vấn dùng subquery để tính số học viên đăng ký 1 lớp quý 2 năm 2023, gộp theo clazz.class_name. Chi phí là **43884.15**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **860.73**.

Vì sao sử dụng index: Index giúp thực hiện index scan trên class_id để giảm chi phí quét toàn bộ bảng enrollment. Đồng thời giúp tối ưu hóa JOIN nhờ giảm dữ liệu cần lọc trong range query của truy vấn. Index cũng giúp giảm số hàng cần JOIN trong c.khai_giang.

```

CREATE INDEX idx_enrollments_class_date ON enrollments (class_id ,
enrollment_date);
CREATE INDEX idx_clazz_khai_giang ON clazz (khai_giang);

```

QUERY PLAN text	
1	Seq Scan on clazz c (cost=0.00..43884.15 rows=84 width=40) (actual time=1.043..97.587 rows=82 loops=1)
2	Filter: ((khai_giang >= '2023-04-01':date) AND (khai_giang <= '2023-06-30':date))
3	Rows Removed by Filter: 918
4	SubPlan 1
5	-> Aggregate (cost=519.87..519.88 rows=1 width=8) (actual time=1.175..1.175 rows=1 loops=82)
6	-> Bitmap Heap Scan on enrollments e (cost=466.44..519.87 rows=1 width=4) (actual time=1.147..1.152 rows=14 loop...
7	Recheck Cond: (class_id = c.class_id)
8	Filter: ((enrollment_date >= '2023-04-01':date) AND (enrollment_date <= '2023-06-30':date))
9	Rows Removed by Filter: 1
10	Heap Blocks: exact=89
11	-> Bitmap Index Scan on enrollments_pkey (cost=0.00..466.44 rows=16 width=0) (actual time=1.138..1.138 rows=1...
12	Index Cond: (class_id = c.class_id)
13	Planning Time: 0.378 ms
14	Execution Time: 97.699 ms

a) Trước khi có index

QUERY PLAN text	
1	Bitmap Heap Scan on clazz c (cost=5.14..860.73 rows=84 width=40) (actual time=0.233..3.159 rows=82 loops=1)
2	Recheck Cond: ((khai_giang >= '2023-04-01':date) AND (khai_giang <= '2023-06-30':date))
3	Heap Blocks: exact=66
4	-> Bitmap Index Scan on idx_clazz_khai_giang (cost=0.00..5.12 rows=84 width=0) (actual time=0.066..0.066 rows=82 loops=1)
5	Index Cond: (khai_giang >= '2023-04-01':date) AND (khai_giang <= '2023-06-30':date))
6	SubPlan 1
7	-> Aggregate (cost=8.31..8.32 rows=1 width=8) (actual time=0.032..0.032 rows=1 loops=82)
8	-> Index Scan using idx_enrollments_class_date on enrollments e (cost=0.29..8.31 rows=1 width=4) (actual time=0.019..0.026 rows=14 l...
9	Index Cond: ((class_id = c.class_id) AND (enrollment_date >= '2023-04-01':date) AND (enrollment_date <= '2023-06-30':date))
10	Planning Time: 0.554 ms
11	Execution Time: 3.279 ms

b) Sau khi có index

Câu 24: Trung bình điểm midterm trong quý 4 năm 2024 theo loại lớp

Cách 1: Dùng AVG and JOIN

```
SELECT ct.describe, AVG(e.midterm) AS avg_midterm
FROM class_type ct
JOIN clazz c ON ct.type_id = c.type_id
JOIN enrollments e ON c.class_id = e.class_id
WHERE c.khai_giang BETWEEN '2024-10-01' AND '2024-12-31'
GROUP BY ct.describe;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm AVG để tính điểm trung bình trong quý 4 năm 2024, điểm trung bình được tính theo loại lớp. Chi phí là **732.10** với execution time là **2.975ms**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **445.98** với execution time là **1.712ms**.

Vì sao dùng index: Dùng index trên bảng enrollments bằng phương pháp Index-Only Scan để cải thiện hiệu suất tổng thể bằng cách giảm chi phí cho các bước JOIN và tính toán.

```
CREATE INDEX idx_enrollments_class_midterm ON enrollments (class_id,
midterm);
```

QUERY PLAN
text
1 HashAggregate (cost=732.05..732.10 rows=4 width=64) (actual time=2.934..2.938 rows=4 loops=1)
2 Group Key: ct.describe
3 Batches: 1 Memory Usage: 24kB
4 -> Hash Join (cost=215.94..726.87 rows=1035 width=38) (actual time=0.339..2.659 rows=897 loops=1)
5 Hash Cond: (c.type_id = ct.type_id)
6 -> Hash Join (cost=214.85..720.18 rows=1035 width=10) (actual time=0.314..2.487 rows=897 loops=1)
7 Hash Cond: (e.class_id = c.class_id)
8 -> Seq Scan on enrollments e (cost=0.00..465.21 rows=15221 width=10) (actual time=0.042..0.876 rows=1522..)
9 -> Hash (cost=214.00..214.00 rows=68 width=8) (actual time=0.225..0.226 rows=67 loops=1)
10 Buckets: 1024 Batches: 1 Memory Usage: 11kB
11 -> Seq Scan on clazz c (cost=0.00..214.00 rows=68 width=8) (actual time=0.008..0.217 rows=67 loops=1)
12 Filter: ((khai_giang >= '2024-10-01':date) AND (khai_giang <= '2024-12-31':date))
13 Rows Removed by Filter: 933
14 -> Hash (cost=1.04..1.04 rows=4 width=36) (actual time=0.020..0.021 rows=4 loops=1)
15 Buckets: 1024 Batches: 1 Memory Usage: 9kB
16 -> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.016..0.017 rows=4 loops=1)
17 Planning Time: 0.359 ms
18 Execution Time: 2.975 ms

a) Trước khi có index

QUERY PLAN
text
1 HashAggregate (cost=445.93..445.98 rows=4 width=64) (actual time=1.651..1.658 rows=4 loops=1)
2 Group Key: ct.describe
3 Batches: 1 Memory Usage: 24kB
4 -> Nested Loop (cost=1.38..440.76 rows=1035 width=38) (actual time=0.099..1.085 rows=897 loops=1)
5 -> Hash Join (cost=1.09..215.46 rows=68 width=36) (actual time=0.049..0.503 rows=67 loops=1)
6 Hash Cond: (c.type_id = ct.type_id)
7 -> Seq Scan on clazz c (cost=0.00..214.00 rows=68 width=8) (actual time=0.024..0.441 rows=67 loops=1)
8 Filter: ((khai_giang >= '2024-10-01':date) AND (khai_giang <= '2024-12-31':date))
9 Rows Removed by Filter: 933
10 -> Hash (cost=1.04..1.04 rows=4 width=36) (actual time=0.017..0.017 rows=4 loops=1)
11 Buckets: 1024 Batches: 1 Memory Usage: 9kB
12 -> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.011..0.012 rows=4 loops=1)
13 -> Index Only Scan using idx_enrollments_class_midterm on enrollments e (cost=0.29..3.15 rows=16 width=10) (actual time=0.004..0.006 rows=13 loops=1)
14 Index Cond: (class_id = c.class_id)
15 Heap Fetches: 0
16 Planning Time: 0.477 ms
17 Execution Time: 1.712 ms

b) Sau khi có index

Cách 2:Dùng subquery

```
SELECT ct.describe , avg_midterm
FROM class_type ct
JOIN (
    SELECT c.type_id , AVG(e.midterm) AS avg_midterm
    FROM clazz c
    JOIN enrollments e ON c.class_id = e.class_id
    WHERE c.khai_giang BETWEEN '2024-10-01' AND '2024-12-31'
    GROUP BY c.type_id
) e ON ct.type_id = e.type_id;
```

Phân tích hiệu năng: Truy vấn sử dụng subquery để tính điểm trung bình theo loại lớp trong quý 4 năm 2024. Chi phí là **726.56** khi chưa có index, với execution time là **2.907ms**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **445.68**, với execution time là **1.088ms** (hiệu quả hơn so với cách 1).

Vì sao sử dụng index: Index tối ưu hóa JOIN và tính toán của AVG trong subquery, bằng cách subquery tách index và logic để hỗ trợ hiệu quả việc truy cập dữ liệu.

```
CREATE INDEX idx_enrollments_class_midterm ON enrollments (class_id ,
midterm);
```

QUERY PLAN text	
1	Hash Join (cost=725.50..726.56 rows=4 width=64) (actual time=2.851..2.853 rows=4 loops=1)
2	Hash Cond: (ct.type_id = e.type_id)
3	-> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.017..0.017 rows=4 loops=1)
4	-> Hash (cost=725.45..725.45 rows=4 width=36) (actual time=2.825..2.827 rows=4 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 9kB
6	-> Subquery Scan on e (cost=725.36..725.45 rows=4 width=36) (actual time=2.813..2.817 rows=4 loops=1)
7	-> HashAggregate (cost=725.36..725.41 rows=4 width=36) (actual time=2.812..2.815 rows=4 loops=1)
8	Group Key: c.type_id
9	Batches: 1 Memory Usage: 24kB
10	-> Hash Join (cost=214.85..720.18 rows=1035 width=10) (actual time=0.742..2.645 rows=897 loops=1)
11	Hash Cond: (e_1.class_id = c.class_id)
12	-> Seq Scan on enrollments e_1 (cost=0.00..465.21 rows=15221 width=10) (actual time=0.064..0.834 rows=15221)
13	-> Hash (cost=214.00..214.00 rows=68 width=8) (actual time=0.578..0.578 rows=67 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 11kB
15	-> Seq Scan on clazz c (cost=0.00..214.00 rows=68 width=8) (actual time=0.012..0.546 rows=67 loops=1)
16	Filter: ((khai_giang >= '2024-10-01':date) AND (khai_giang <= '2024-12-31':date))
17	Rows Removed by Filter: 933
18	Planning Time: 1.044 ms
19	Execution Time: 2.907 ms

a) Trước khi có index

QUERY PLAN text	
1	Hash Join (cost=444.62..445.68 rows=4 width=64) (actual time=1.041..1.044 rows=4 loops=1)
2	Hash Cond: (ct.type_id = e.type_id)
3	-> Seq Scan on class_type ct (cost=0.00..1.04 rows=4 width=36) (actual time=0.013..0.013 rows=4 loops=1)
4	-> Hash (cost=444.56..444.56 rows=4 width=36) (actual time=0.990..0.991 rows=4 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 9kB
6	-> Subquery Scan on e (cost=444.47..444.56 rows=4 width=36) (actual time=0.978..0.981 rows=4 loops=1)
7	-> HashAggregate (cost=444.47..444.52 rows=4 width=36) (actual time=0.978..0.980 rows=4 loops=1)
8	Group Key: c.type_id
9	Batches: 1 Memory Usage: 24kB
10	-> Nested Loop (cost=0.29..439.30 rows=1035 width=10) (actual time=0.098..0.807 rows=897 loops=1)
11	-> Seq Scan on clazz c (cost=0.00..214.00 rows=68 width=8) (actual time=0.018..0.243 rows=67 loops=1)
12	Filter: ((khai_giang >= '2024-10-01':date) AND (khai_giang <= '2024-12-31':date))
13	Rows Removed by Filter: 933
14	-> Index Only Scan using idx_enrollments_class_midterm on enrollments e_1 (cost=0.29..3.15 rows=16 width=10) (actual time=0.006..0.007 rows=16)
15	Index Cond: (class_id = c.class_id)
16	Heap Fetches: 0
17	Planning Time: 1.799 ms
18	Execution Time: 1.088 ms

b) Sau khi có index

Câu 25: Tổng số học viên vắng mặt trong quý 4 năm 2024

Cách 1: Dùng COUNT and JOIN

```
SELECT c.class_name, COUNT(a.student_id) AS absent_count
FROM clazz c
JOIN attendance a ON c.class_id = a.class_id
WHERE a.attendance_date BETWEEN '2024-01-01' AND '2024-03-31',
AND a.status = '1'
GROUP BY c.class_name;
```

Phân tích hiệu năng: Câu truy vấn dùng hàm COUNT để tính số học viên vắng mặt trong quý 1 năm 2024, COUNT theo student_id. Chi phí là **3889.57** với execution time là **22.949ms**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **1575.43**, với execution time là **2.333ms**.

Vì sao sử dụng index: Index được sử dụng để hỗ trợ hiệu quả việc lọc dữ liệu trong cột attendance_date và status. Bằng phương pháp Bitmap Index Scan, index giảm số hàng cần được xử lý trước hàm JOIN.

```
CREATE INDEX idx_attendance_date_absent ON attendance (
    attendance_date, status);
```

QUERY PLAN text	
1	HashAggregate (cost=3880.89..3889.57 rows=868 width=40) (actual time=22.892..22.911 rows=136 loops=1)
2	Group Key: c.class_name
3	Batches: 1 Memory Usage: 57kB
4	-> Hash Join (cost=221.50..3847.69 rows=6640 width=36) (actual time=0.347..20.914 rows=6654 loops=1)
5	Hash Cond: (a.class_id = c.class_id)
6	-> Seq Scan on attendance a (cost=0.00..3608.69 rows=6640 width=9) (actual time=0.054..18.776 rows=6654 loops=1)
7	Filter: ((attendance_date >= '2024-01-01':date) AND (attendance_date <= '2024-03-31':date) AND (status = '1':bpchar))
8	Rows Removed by Filter: 144528
9	-> Hash (cost=209.00..209.00 rows=1000 width=36) (actual time=0.288..0.288 rows=1000 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 76kB
11	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.006..0.194 rows=1000 loops=1)
12	Planning Time: 0.259 ms
13	Execution Time: 22.949 ms

a) Trước khi có index

QUERY PLAN text	
1	HashAggregate (cost=1566.75..1575.43 rows=868 width=40) (actual time=9.129..9.149 rows=136 loops=1)
2	Group Key: c.class_name
3	Batches: 1 Memory Usage: 57kB
4	-> Hash Join (cost=436.84..1533.55 rows=6640 width=36) (actual time=5.607..7.308 rows=6654 loops=1)
5	Hash Cond: (a.class_id = c.class_id)
6	-> Bitmap Heap Scan on attendance a (cost=215.34..1294.54 rows=6640 width=8) (actual time=2.646..3.317 rows=6654 loops=1)
7	Recheck Cond: ((attendance_date >= '2024-01-01':date) AND (attendance_date <= '2024-03-31':date) AND (status = '1':bpchar))
8	Heap Blocks: exact=242
9	-> Bitmap Index Scan on idx_attendance_date_absent (cost=0.00..213.68 rows=6640 width=0) (actual time=2.602..2.602 rows=6654 loops=1)
10	Index Cond: ((attendance_date >= '2024-01-01':date) AND (attendance_date <= '2024-03-31':date) AND (status = '1':bpchar))
11	-> Hash (cost=209.00..209.00 rows=1000 width=36) (actual time=2.922..2.923 rows=1000 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 76kB
13	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.055..2.727 rows=1000 loops=1)
14	Planning Time: 3.610 ms
15	Execution Time: 11.094 ms

b) Sau khi có index

Cách 2:Dùng subquery

```
SELECT c.class_name, absent_count
FROM clazz c
JOIN (
    SELECT class_id, COUNT(*) AS absent_count
    FROM attendance
    WHERE attendance_date BETWEEN '2024-01-01' AND '2024-03-31'
    AND status = '1'
    GROUP BY class_id
) a ON c.class_id = a.class_id;
```

Phân tích hiệu năng: Truy vấn sử dụng subquery để tính số học viên vắng mặt trong quý 1 năm 2024. Chi phí là **3884.69** khi chưa có index, với execution time là **33.888ms**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **1570.55**, với execution time là **7.479ms** (hiệu quả hơn so với cách 1).

Vì sao sử dụng index: Tương tự như cách 1, nhưng subquery cho phép tách phần index và phần logic tính toán để linh hoạt hơn trong việc truy cập dữ liệu.

```
CREATE INDEX idx_attendance_date_absent ON attendance (
    attendance_date, status);
```

QUERY PLAN text	
1	Hash Join (cost=3673.05..3884.69 rows=959 width=40) (actual time=20.033..20.239 rows=140 loops=1)
2	Hash Cond: (c.class_id = a.class_id)
3	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.017..0.142 rows=1000 loops=1)
4	-> Hash (cost=3661.07..3661.07 rows=959 width=12) (actual time=19.995..19.996 rows=140 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 15kB
6	-> Subquery Scan on a (cost=3641.89..3661.07 rows=959 width=12) (actual time=19.950..19.978 rows=140 loops=1)
7	-> HashAggregate (cost=3641.89..3651.48 rows=959 width=12) (actual time=19.949..19.970 rows=140 loops=1)
8	Group Key: attendance.class_id
9	Batches: 1 Memory Usage: 81kB
10	-> Seq Scan on attendance (cost=0.00..3608.69 rows=6640 width=4) (actual time=0.118..18.477 rows=6640 loops=1)
11	Filter: ((attendance_date >= '2024-01-01':date) AND (attendance_date <= '2024-03-31':date) AND (status = '1'))
12	Rows Removed by Filter: 144528
13	Planning Time: 0.256 ms
14	Execution Time: 20.283 ms

a) Trước khi có index

QUERY PLAN text	
1	Hash Join (cost=1358.91..1570.55 rows=959 width=40) (actual time=1.959..2.270 rows=140 loops=1)
2	Hash Cond: (c.class_id = a.class_id)
3	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.014..0.198 rows=1000 loops=1)
4	-> Hash (cost=1346.92..1346.92 rows=959 width=12) (actual time=1.914..1.915 rows=140 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 15kB
6	-> Subquery Scan on a (cost=1327.74..1346.92 rows=959 width=12) (actual time=1.856..1.893 rows=140 loops=1)
7	-> HashAggregate (cost=1327.74..1337.33 rows=959 width=12) (actual time=1.855..1.876 rows=140 loops=1)
8	Group Key: attendance.class_id
9	Batches: 1 Memory Usage: 81kB
10	-> Bitmap Heap Scan on attendance (cost=215.34..1294.54 rows=6640 width=4) (actual time=0.361..1.043 rows=6654 loops=1)
11	Recheck Cond: ((attendance_date >= '2024-01-01':date) AND (attendance_date <= '2024-03-31':date) AND (status = '1'))
12	Heap Blocks: exact=242
13	-> Bitmap Index Scan on idx_attendance_date_absent (cost=0.00..213.68 rows=6640 width=0) (actual time=0.326..0.327 rows=6640 loops=1)
14	Index Cond: ((attendance_date >= '2024-01-01':date) AND (attendance_date <= '2024-03-31':date) AND (status = '1'))
15	Planning Time: 1.613 ms
16	Execution Time: 2.333 ms

b) Sau khi có index

Câu 26: Danh sách lớp có sĩ số thay đổi trong quý 3/2023

Cách 1: Dùng EXIST

```
SELECT c.class_name, c.si_so
FROM clazz c
WHERE EXISTS (
    SELECT 1 FROM enrollments e
    WHERE e.class_id = c.class_id
    AND e.enrollment_date BETWEEN '2023-07-01' AND '2023-09-30'
);
```

Phân tích hiệu năng: Câu truy vấn dùng hàm EXISTS để tính danh sách lớp có sĩ số thay đổi trong quý 3/2023, hàm EXISTS lọc các dòng chỉ chứa enrollment_date từ tháng 7 đến tháng 9. Chi phí là **783.28** với execution time là **9.385ms**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **596.05**, với execution time là **1.456ms**.

Vì sao sử dụng index: Index được sử dụng để hỗ trợ tăng tốc việc lọc dữ liệu trong cột enrollment_date bằng phương pháp Bitmap Index Scan.

```
CREATE INDEX idx_enrollments_enroll_date ON enrollments (
    enrollment_date);
```

QUERY PLAN text	
1	HashAggregate (cost=774.53..783.02 rows=849 width=36) (actual time=5.562..5.584 rows=83 loops=1)
2	Group Key: c.class_name, c.si_so
3	Batches: 1 Memory Usage: 49kB
4	-> Hash Join (cost=223.54..768.27 rows=1251 width=36) (actual time=1.762..4.876 rows=1303 loops=1)
5	Hash Cond: (e.class_id = c.class_id)
6	-> Seq Scan on enrollments e (cost=0.00..541.32 rows=1299 width=4) (actual time=0.411..2.995 rows=1303 loops=1)
7	Filter: ((enrollment_date >= '2023-07-01':date) AND (enrollment_date <= '2023-09-30':date))
8	Rows Removed by Filter: 13918
9	-> Hash (cost=211.50..211.50 rows=963 width=40) (actual time=1.331..1.331 rows=963 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 78kB
11	-> Seq Scan on clazz c (cost=0.00..211.50 rows=963 width=40) (actual time=0.082..0.922 rows=963 loops=1)
12	Filter: (si_so > 0)
13	Rows Removed by Filter: 37
14	Planning Time: 3.612 ms
15	Execution Time: 5.752 ms

a) Trước khi có index

QUERY PLAN text	
1	HashAggregate (cost=587.30..595.79 rows=849 width=36) (actual time=4.100..4.125 rows=83 loops=1)
2	Group Key: c.class_name, c.si_so
3	Batches: 1 Memory Usage: 49kB
4	-> Hash Join (cost=245.14..581.04 rows=1251 width=36) (actual time=2.052..3.264 rows=1303 loops=1)
5	Hash Cond: (e.class_id = c.class_id)
6	-> Bitmap Heap Scan on enrollments e (cost=21.60..354.08 rows=1299 width=4) (actual time=0.179..0.624 rows=1303 loops=1)
7	Recheck Cond: (enrollment_date >= '2023-07-01':date) AND (enrollment_date <= '2023-09-30':date)
8	Heap Blocks: exact=73
9	-> Bitmap Index Scan on idx_enrollments_enroll_date (cost=0.00..21.28 rows=1299 width=0) (actual time=0.122..0.122 rows=1303 loops=1)
10	Index Cond: (enrollment_date >= '2023-07-01':date) AND (enrollment_date <= '2023-09-30':date)
11	-> Hash (cost=211.50..211.50 rows=963 width=40) (actual time=1.842..1.842 rows=963 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 78kB
13	-> Seq Scan on clazz c (cost=0.00..211.50 rows=963 width=40) (actual time=0.067..1.241 rows=963 loops=1)
14	Filter: (si_so > 0)
15	Rows Removed by Filter: 37
16	Planning Time: 0.932 ms
17	Execution Time: 4.225 ms

b) Sau khi có index

Cách 2:Dùng HAVING và JOIN

```
SELECT c.class_name, c.si_so
FROM clazz c
JOIN enrollments e ON c.class_id = e.class_id
WHERE e.enrollment_date BETWEEN '2023-07-01' AND '2023-09-30'
GROUP BY c.class_name, c.si_so
HAVING c.si_so > 0;
```

Phân tích hiệu năng: Truy vấn sử dụng hàm HAVING và JOIN để tính danh sách lớp có số sinh viên thay đổi trong quý 3/2023. Chi phí là **783.02** khi chưa có index, với execution time là **5.752ms**. Sau khi thêm index, chi phí đã giảm đáng kể xuống chỉ còn **595.79**, với execution time là **4.225ms** (hiệu quả hơn so với cách 1).

Vì sao sử dụng index: Index sẽ lọc trước các hàng trong enrollments, từ đó giảm số hàng cần JOIN. GROUP BY và HAVING cũng giúp tối ưu hóa dữ liệu.

```
CREATE INDEX idx_enrollments_enroll_date ON enrollments (
    enrollment_date);
```

QUERY PLAN text	
1	Hash Join (cost=3673.05..3884.69 rows=959 width=40) (actual time=20.033..20.239 rows=140 loops=1)
2	Hash Cond: (c.class_id = a.class_id)
3	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.017..0.142 rows=1000 loops=1)
4	-> Hash (cost=3661.07..3661.07 rows=959 width=12) (actual time=19.995..19.996 rows=140 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 15kB
6	-> Subquery Scan on a (cost=3641.89..3661.07 rows=959 width=12) (actual time=19.950..19.978 rows=140 loops=1)
7	-> HashAggregate (cost=3641.89..3651.48 rows=959 width=12) (actual time=19.949..19.970 rows=140 loops=1)
8	Group Key: attendance.class_id
9	Batches: 1 Memory Usage: 81kB
10	-> Seq Scan on attendance (cost=0.00..3608.69 rows=6640 width=4) (actual time=0.118..18.477 rows=6...
11	Filter: ((attendance_date >= '2024-01-01'::date) AND (attendance_date <= '2024-03-31'::date) AND (status = '1'::bpch...)
12	Rows Removed by Filter: 144528
13	Planning Time: 0.256 ms
14	Execution Time: 20.283 ms

a) Trước khi có index

QUERY PLAN text	
1	Hash Join (cost=1358.91..1570.55 rows=959 width=40) (actual time=1.959..2.270 rows=140 loops=1)
2	Hash Cond: (c.class_id = a.class_id)
3	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.014..0.198 rows=1000 loops=1)
4	-> Hash (cost=1346.92..1346.92 rows=959 width=12) (actual time=1.914..1.915 rows=140 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 15kB
6	-> Subquery Scan on a (cost=1327.74..1346.92 rows=959 width=12) (actual time=1.856..1.893 rows=140 loops=1)
7	-> HashAggregate (cost=1327.74..1337.33 rows=959 width=12) (actual time=1.855..1.876 rows=140 loops=1)
8	Group Key: attendance.class_id
9	Batches: 1 Memory Usage: 81kB
10	-> Bitmap Heap Scan on attendance (cost=215.34..1294.54 rows=6640 width=4) (actual time=0.361..1.043 rows=6654 loops=1)
11	Recheck Cond: ((attendance_date >= '2024-01-01'::date) AND (attendance_date <= '2024-03-31'::date) AND (status = '1'::bpch...)
12	Heap Blocks: exact=242
13	-> Bitmap Index Scan on idx_attendance_date_absent (cost=0.00..213.68 rows=6640 width=0) (actual time=0.326..0.327 ro...
14	Index Cond: ((attendance_date >= '2024-01-01'::date) AND (attendance_date <= '2024-03-31'::date) AND (status = '1'::bpch...)
15	Planning Time: 1.613 ms
16	Execution Time: 2.333 ms

b) Sau khi có index

CÂU 27: View hiển thị các lớp đãi trong quý 2/2023

```

CREATE VIEW vw_full_classes AS
SELECT c.class_id, c.class_name, c.si_so, c.khai_giang
FROM clazz c
WHERE c.khai_giang BETWEEN '2023-04-01' AND '2023-06-15'
AND c.si_so = 30
WITH CHECK OPTION;

```

Phân tích hiệu năng: Đây là 1 view để hiển thị các lớp đãi (các lớp có số sinh viên là 30) trong quý 2 năm 2023. Câu lệnh bên trong View có chi phí là **216.50** và execution time là **2.634ms**.

QUERY PLAN text	
1	Seq Scan on clazz c (cost=0.00..216.50 rows=3 width=44) (actual time=0.304..0.694 rows=5 loops=1)
2	Filter: ((khai_giang >= '2023-04-01'::date) AND (khai_giang <= '2023-06-15'::date) AND (si_so = 30))
3	Rows Removed by Filter: 995
4	Planning Time: 0.267 ms
5	Execution Time: 2.634 ms

a) Query Plan của câu lệnh trong View

CÂU 28: View hiển thị các học sinh vắng mặt trong tháng 6/2024

```

CREATE VIEW vw_absent_students AS
SELECT hv.student_id, hv.full_name, c.class_name, a.attendance_date
FROM hoc_vien hv
JOIN attendance a ON hv.student_id = a.student_id
JOIN clazz c ON a.class_id = c.class_id
WHERE a.attendance_date BETWEEN '2024-06-01' AND '2024-06-30'
AND a.status = '1';

```

Phân tích hiệu năng: Đây là 1 view để hiển thị các học sinh vắng mặt trong tháng 6 năm 2024. Câu lệnh bên trong View có chi phí là **4563.61** và execution time là **68.995ms** . -

QUERY PLAN text	
1	Hash Join (cost=942.50..4563.61 rows=2363 width=57) (actual time=13.485..67.774 rows=2419 loops=1)
2	Hash Cond: (a.class_id = c.class_id)
3	-> Hash Join (cost=721.00..4335.89 rows=2363 width=29) (actual time=12.302..64.847 rows=2419 loops=1)
4	Hash Cond: (a.student_id = hv.student_id)
5	-> Seq Scan on attendance a (cost=0.00..3608.69 rows=2363 width=12) (actual time=0.641..50.908 rows=2419 ...)
6	Filter: ((attendance_date >= '2024-06-01'::date) AND (attendance_date <= '2024-06-30'::date) AND (status = '1'...)
7	Rows Removed by Filter: 148763
8	-> Hash (cost=471.00..471.00 rows=20000 width=21) (actual time=11.397..11.400 rows=20000 loops=1)
9	Buckets: 32768 Batches: 1 Memory Usage: 1302kB
10	-> Seq Scan on hoc_vien hv (cost=0.00..471.00 rows=20000 width=21) (actual time=0.043..4.618 rows=200...
11	-> Hash (cost=209.00..209.00 rows=1000 width=36) (actual time=1.171..1.173 rows=1000 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 76kB
13	-> Seq Scan on clazz c (cost=0.00..209.00 rows=1000 width=36) (actual time=0.025..0.755 rows=1000 loops=1)
14	Planning Time: 1.593 ms
15	Execution Time: 68.995 ms

a) Query Plan của câu lệnh trong View

Câu 29: Thứ hạng điểm tổng kết của học viên trong mỗi lớp quý 2 năm 2024

Sử dụng RANK

```

SELECT hv.full_name, c.class_name,
       ROUND(((e.minitest1 + e.minitest2 + e.minitest3 + e.minitest4)
* 0.4 +
       e.midterm * 0.3 + e.final_test * 0.3), 2) AS
diem_tong_ket,
       RANK() OVER (PARTITION BY c.class_id ORDER BY ROUND(((e.
minitest1 + e.minitest2 + e.minitest3 + e.minitest4) * 0.4 +
       e.midterm * 0.3 + e.final_test * 0.3), 2) DESC) AS rank
FROM hoc_vien hv
JOIN enrollments e ON hv.student_id = e.student_id
JOIN clazz c ON e.class_id = c.class_id
WHERE c.khai_giang BETWEEN '2024-04-01' AND '2024-06-24';

```

Phân tích hiệu năng: Truy vấn sử dụng hàm RANK để cho biết Thứ hạng điểm tổng kết của học viên trong mỗi lớp quý 2 năm 2024 dựa theo trọng số điểm. Chi phí là **1393.08**. Truy vấn tính điểm tổng kết của từng học viên sau đó sử dụng hàm ROUND để làm tròn điểm. Truy vấn sử dụng hàm PARTITION để chia dữ liệu về các nhóm, ở đây là xếp hạng riêng của từng lớp, chia theo class_id. Hàm RANK được truy vấn sử dụng để xếp hạng điểm của học viên trong từng lớp. Truy vấn không cần index do phạm vi điều kiện WHERE ngắn nên chi phí sẽ không thay đổi rõ rệt khi thêm index.

QUERY PLAN text	
1	WindowAgg (cost=1336.83..1393.08 rows=1324 width=93) (actual time=5.919..6.442 rows=1247 loops=1)
2	-> Sort (cost=1336.81..1340.12 rows=1324 width=85) (actual time=5.913..5.954 rows=1247 loops=1)
3	Sort Key: c.class_id, (round((((e.minitest1 + e.minitest2) + e.minitest3) + e.minitest4) * 0.4) + (e.midterm * 0.3)) + (e.final_test * 0.3), 2))...
4	Sort Method: quicksort Memory: 147kB
5	-> Nested Loop (cost=215.38..1268.15 rows=1324 width=85) (actual time=0.352..5.278 rows=1247 loops=1)
6	-> Hash Join (cost=215.09..720.42 rows=1324 width=76) (actual time=0.335..2.288 rows=1247 loops=1)
7	Hash Cond: (e.class_id = c.class_id)
8	-> Seq Scan on enrollments e (cost=0.00..465.21 rows=15221 width=44) (actual time=0.060..0.789 rows=15221 loops=1)
9	-> Hash (cost=214.00..214.00 rows=87 width=36) (actual time=0.211..0.211 rows=86 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 14kB
11	-> Seq Scan on clazz c (cost=0.00..214.00 rows=87 width=36) (actual time=0.014..0.192 rows=86 loops=1)
12	Filter: ((khai_giang >= '2024-04-01)::date) AND (khai_giang <= '2024-06-24)::date)
13	Rows Removed by Filter: 914
14	-> Index Scan using hoc_vien_pkey on hoc_vien hv (cost=0.29..0.39 rows=1 width=21) (actual time=0.001..0.001 rows=1 loops=1247)
15	Index Cond: (student_id = e.student_id)
16	Planning Time: 1.810 ms
17	Execution Time: 6.565 ms

a) Query Plan của truy vấn

CÂU 30: Thông tin chi tiết về lịch học của từng lớp trong tháng 05/2024

Sử dụng CROSS JOIN LATERAL

```
SELECT c.class_name, s.days, s.start_time, s.end_time
FROM clazz c
CROSS JOIN LATERAL (
    SELECT days, start_time, end_time
    FROM schedule s
    WHERE s.class_id = c.class_id
    AND c.khai_giang BETWEEN '2024-05-01' AND '2024-05-24'
) s;
```

Phân tích hiệu năng: Đây là 1 câu truy vấn sử dụng CROSS JOIN LATERAL để hiển thị thông tin chi tiết về lịch học của từng lớp trong tháng 05/2024. Câu truy vấn có chi phí là **240.01**. Truy vấn kết hợp các hàng từ clazz với 1 tập con của bảng schedule được tạo ra từ CROSS JOIN LATERAL. LATERAL cho phép truy vấn bên trong tham chiếu đến hàng hiện tại của bảng ngoài. Truy vấn bên trong LATERAL thực hiện hiển thị thông tin về các lớp học trong tháng 5/2024.

	QUERY PLAN text	锁
1	Hash Join (cost=214.38..240.01 rows=30 width=95) (actual time=0.505..0.645 rows=30 loops=1)	
2	Hash Cond: (s.class_id = c.class_id)	
3	-> Seq Scan on schedule s (cost=0.00..23.00 rows=1000 width=67) (actual time=0.016..0.082 rows=10...)	
4	-> Hash (cost=214.00..214.00 rows=30 width=36) (actual time=0.455..0.456 rows=30 loops=1)	
5	Buckets: 1024 Batches: 1 Memory Usage: 11kB	
6	-> Seq Scan on clazz c (cost=0.00..214.00 rows=30 width=36) (actual time=0.057..0.444 rows=30 l...)	
7	Filter: ((khai_giang >= '2024-05-01'::date) AND (khai_giang <= '2024-05-24'::date))	
8	Rows Removed by Filter: 970	
9	Planning Time: 1.857 ms	
10	Execution Time: 0.675 ms	

a) Query Plan của truy vấn

Phần 5

Xây dựng chương trình

5.1 Công nghệ sử dụng

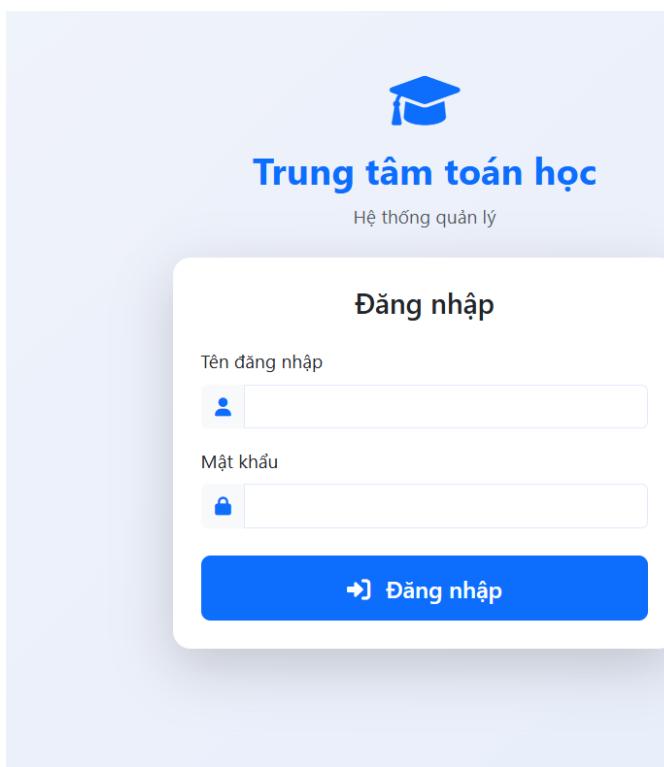
- Backend : Django
- Frontend: HTML + CSS
- DBMS : PostgreSQL

5.2 Chức năng và giao diện chương trình

Link Github: <https://github.com/NeoCyber05/Central-Management-Web>

Phần đăng nhập

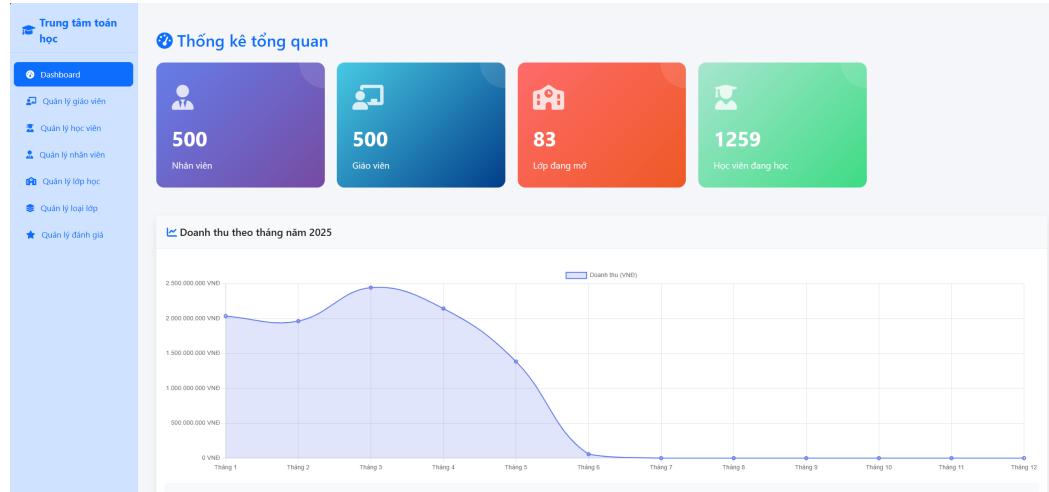
Khi vô Web thì người dùng sẽ đăng nhập với tài khoản được cấp để login vô.



Hình 5.1: Giao diện đăng nhập lúc vô Web

Phần Dashboard

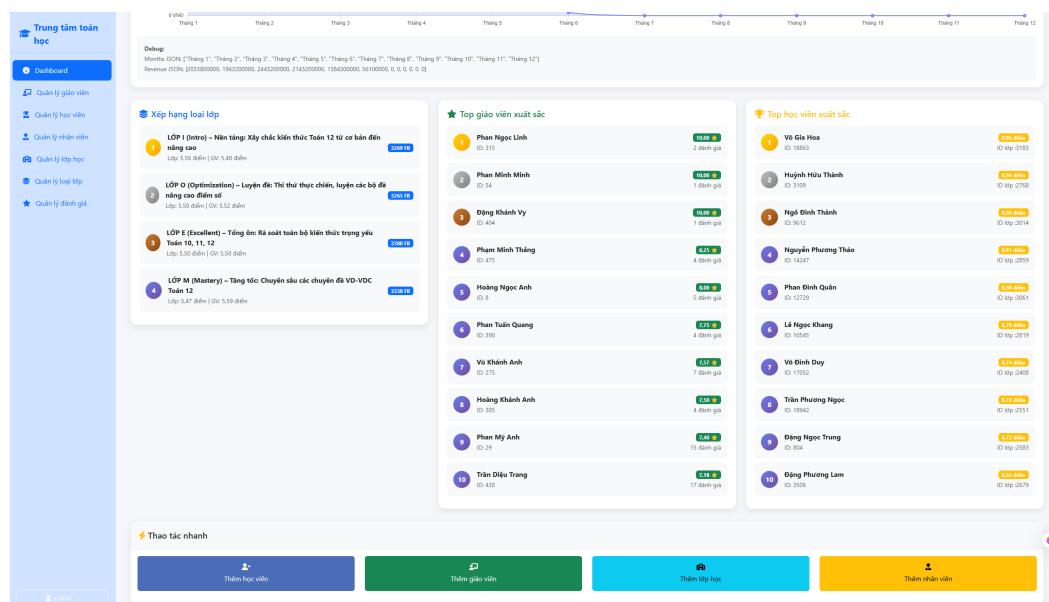
Phần Dashboard sẽ dành cho Quản lý trung tâm , tại đây quản lý có thể xem các thống kê tổng quan về trung tâm như số nhân viên , giáo viên , số lớp đang mở và số học viên đang học tại trung tâm,...



Hình 5.2: Giao diện Dashboard lúc Login vô

Phần tiếp theo của Dashboard (Hình 5.3) sẽ có các bảng đánh giá :

- Xếp hạng loại lớp
- Top 10 giáo viên xuất sắc
- Top 10 học viên xuất sắc



Hình 5.3: Giao diện phần dưới của Dashboard

Phần Quản lý nhân viên / giáo viên / học viên

Tại phần này thì người dùng có thể xem được danh sách và tìm kiếm theo tên/email/SDT/ID,. Đồng thời có thể thêm được nhân viên / giáo viên / học viên bằng cách điền các thông tin cần thiết của từng loại.

Mã GV	Họ tên	Giới tính	Ngày sinh	Email	SĐT	Địa chỉ	Trình độ	Hành động
1	Lê Mỹ Anh	Nữ	Ngày 16 tháng 7 năm 1994	anh.l1@giaoVien.edu.vn	0968775810	224 Xuân Thủy	Tiến Sĩ	Sửa Xóa
264	Hoàng Khánh Vy	Nữ	Ngày 12 tháng 11 năm 1988	vy.h264@giaoVien.edu.vn	0905355320	139 Kim Mã	Thạc Sĩ	Sửa Xóa
2	Phan Anh Tuấn	Nam	Ngày 04 tháng 3 năm 1996	tuấn.p2@giaoVien.edu.vn	0936718867	64 Kim Mã	Thạc Sĩ	Sửa Xóa
3	Nguyễn Mỹ Hương	Nữ	Ngày 16 tháng 12 năm 1993	huong.n3@giaoVien.edu.vn	0909826750	79 Kim Mã	Thạc Sĩ	Sửa Xóa
4	Trần Mỹ Vy	Nữ	Ngày 25 tháng 7 năm 1992	vy.t4@giaoVien.edu.vn	0939712797	220 Láng Hạ	Thạc Sĩ	Sửa Xóa
5	Võ Thành Sơn	Nam	Ngày 04 tháng 4 năm 1994	sơn.v5@giaoVien.edu.vn	0906600377	95 Mỹ Đình	Cử nhân	Sửa Xóa
6	Lê Minh Tuấn	Nam	Ngày 26 tháng 8 năm 1997	tuấn.l6@giaoVien.edu.vn	0964631130	133 Láng Hạ	Cử nhân	Sửa Xóa
7	Vũ Ngọc Linh	Nữ	Ngày 07 tháng 3 năm 1991	linh.v7@giaoVien.edu.vn	0969330506	136 Kim Mã	Thạc Sĩ	Sửa Xóa
8	Hoàng Ngọc Anh	Nữ	Ngày 27 tháng 5 năm 1992	anh.h8@giaoVien.edu.vn	0931056579	73 Giải Phóng	Cử nhân	Sửa Xóa
9	Hoàng Thị Huyền	Nữ	Ngày 29 tháng 11 năm 1993	huyen.h9@giaoVien.edu.vn	0861552885	177 Kim Mã	Thạc Sĩ	Sửa Xóa

Hình 5.4: Giao diện phần quản lý giáo viên

Hình 5.5: Giao diện phần thêm giáo viên

★ Giao diện và chức năng cho nhân viên và học viên tương tự.

Phần quản lý lớp học

Tại phần này thì người dùng có thể xem được danh sách và tìm kiếm theo lớp/mã lớp/phòng/ID,..

Đồng thời có thể thêm được lớp học bằng cách điền các thông tin cần thiết và có thể ấn vô tên lớp để xem thông tin chi tiết cụ thể lớp ấy (hình 5.7) hoặc có thể ấn Thêm lớp học (hình 5.8)

Mã lớp	Tên lớp	Giáo viên	Nhân viên quản lý	Class Type ID	Phòng	Khai giảng	Kết thúc	Số lượng	Học phí	Hành động
2247	Chuyên đề M-34 - Mâm Non	Nguyễn Đức Minh	Trần Hữu Hải	2	105	Ngày 05 tháng 12 năm 2024	Ngày 05 tháng 3 năm 2025	0	3800000 VND	Sửa Xóa
2299	Chuyên đề M-86 - Trời Non	Hoàng Quang Tuấn	Nguyễn Công Hải	2	201	Ngày 24 tháng 12 năm 2024	Ngày 24 tháng 3 năm 2025	0	3500000 VND	Sửa Xóa
2300	Nền tảng O-87 - RTX	Hoàng Anh Thắng	Nguyễn Thị An	3	102	Ngày 02 tháng 6 năm 2025	Ngày 02 tháng 9 năm 2025	0	2800000 VND	Sửa Xóa
2308	Nền tảng O-95 - A+	Võ Anh Dũng	Đỗ Minh An	3	103	Ngày 18 tháng 10 năm 2024	Ngày 18 tháng 1 năm 2025	0	2800000 VND	Sửa Xóa
2310	Luyện đề I-97 - Trời Non	Hoàng Tuấn Quang	Trần Ngọc Hương	1	302	Ngày 16 tháng 4 năm 2024	Ngày 16 tháng 7 năm 2024	0	3300000 VND	Sửa Xóa
2318	Luyện đề I-10 - Mâm Non	Lê Ngọc Huyền	Trần Minh Tùng	1	304	Ngày 18 tháng 2 năm 2025	Ngày 18 tháng 5 năm 2025	0	3400000 VND	Sửa Xóa
2333	VIP 1-1 E-12 - 12	Phan Quang Tháng	Nguyễn Hữu An	4	409	Ngày 01 tháng 9 năm 2023	Ngày 01 tháng 12 năm 2023	0	6000000 VND	Sửa Xóa
2363	Chuyên đề M-15 - 12	Nguyễn Mỹ Vy	Đỗ Văn Hải	2	103	Ngày 28 tháng 6 năm 2024	Ngày 28 tháng 9 năm 2024	0	3800000 VND	Sửa Xóa

Hình 5.6: Giao diện phần quản lý lớp học

Mã HV	Họ tên	Email	SDT	Ngày đăng ký	Hành động
15681	Hoàng Bảo Ngân	ngan.h15681@email.com	0927672424	Ngày 04 tháng 4 năm 2023	Chi tiết Xóa
8153	Huỳnh Đức Duy	duy.h8153@email.com	0333966143	Ngày 01 tháng 4 năm 2023	Chi tiết Xóa
3848	Trần Khanh Lan	lan.t3848@email.com	0986838222	Ngày 30 tháng 3 năm 2023	Chi tiết Xóa
15451	Ngô Khánh An	an.n15451@email.com	0378544707	Ngày 28 tháng 3 năm 2023	Chi tiết Xóa
19486	Hoàng Khánh Uyên	uyen.h19486@email.com	0722136823	Ngày 22 tháng 3 năm 2023	Chi tiết Xóa
6222	Lê Bảo Hiền	hienn.l6222@email.com	0573166192	Ngày 26 tháng 3 năm 2023	Chi tiết Xóa
18196	Lê Xuân Hiếu	hiieu.l18196@email.com	0562682366	Ngày 29 tháng 3 năm 2023	Chi tiết Xóa
10073	Hoàng Công Lân	lam.h10073@email.com	0387322476	Ngày 31 tháng 3 năm 2023	Chi tiết Xóa
3228	Phạm Phương Ngọc	ngoc.p3228@email.com	0834233444	Ngày 25 tháng 3 năm 2023	Chi tiết Xóa

Hình 5.7: Giao diện chi tiết từng lớp

+ Thêm lớp học

[← Quay lại danh sách](#)

Thông tin lớp học

Thông tin cơ bản

Tên lớp học *

Loại lớp * Chọn hoặc tìm kiếm...

Phân công nhân sự

Giáo viên * Chọn hoặc tìm kiếm...

Nhân viên quản lý * Chọn hoặc tìm kiếm...

Thời gian & địa điểm

Phòng học * Ngày khai giảng * Ngày kết thúc *

Thông tin học phí

Học phí * VND Nhập số tiền bằng số (VD: 1500000)

[Hủy bỏ](#) [Thêm lớp học](#)

Lưu ý quan trọng:

- Các trường có dấu (*) là bắt buộc
- Ngày kết thúc phải sau ngày khai giảng
- Giáo viên và nhân viên quản lý không được trùng nhau
- Phòng học không được để trống

Hình 5.8: Giao diện thêm lớp học

Phần quản lý loại lớp

Phần này chức năng cũng khá tương đồng với các phần trên.

Trung tâm toán học

[Dashboard](#) [Quản lý giáo viên](#) [Quản lý học viên](#) [Quản lý nhân viên](#) [Quản lý lớp học](#) [Quản lý loại lớp](#) [Quản lý danh sách](#)

Danh sách loại lớp

[+ Thêm loại lớp](#)

Mã code	Mô tả	Hành động
M	LỚP M (Mastery) – Tăng tốc: Chuyển sâu các chuyên đề VD-VDC Toán 12	Sửa Xóa
O	LỚP O (Optimization) – Luyện đề: Thị thử thực chiến, luyện các bộ đề nâng cao điểm số	Sửa Xóa
E	LỚP E (Excellent) – Tổng ôn: Rà soát toàn bộ kiến thức trọng yếu Toán 10, 11, 12	Sửa Xóa
I	LỚP I (Intro) – Nền tảng: Xây chắc kiến thức Toán 12 từ cơ bản đến nâng cao	Sửa Xóa

Hình 5.9: Giao diện loại lớp

Mô tả

Mã code

Quay lại Lưu

Hình 5.10: Giao diện thêm loại lớp

Quản lý đánh giá

Sau mỗi khi khóa học kết thúc thì học viên sẽ được làm 1 form đánh giá . Phần này sẽ hiển thị các đánh giá gần nhất và có thể thêm đánh giá bằng 2 cách :

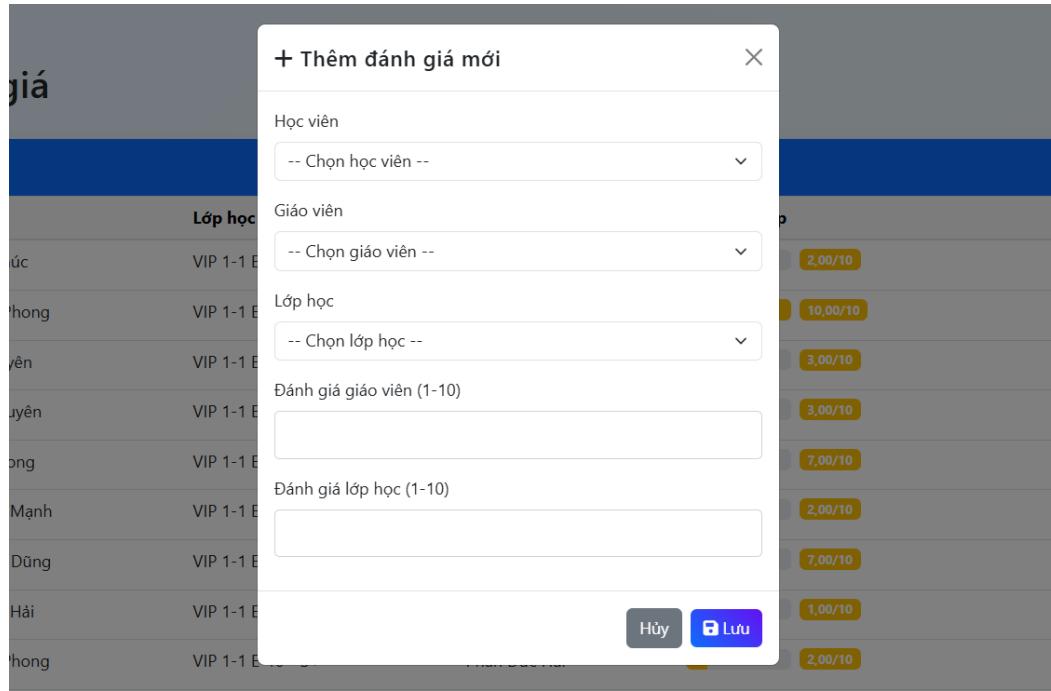
- Điền thủ công (hình 5.12)
- Nhập bằng file Excel (file này trích xuất từ Google Form) (hình 5.13)

★ Quản lý đánh giá

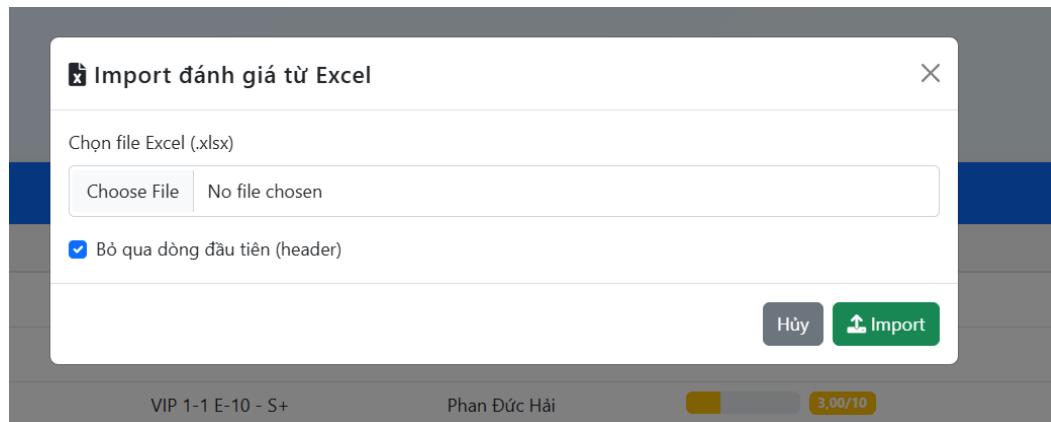
Import từ Excel + Thêm đánh giá

Mã đánh giá	Học viên	Lớp học	Giáo viên	Đánh giá lớp	Đánh giá giáo viên	Hành động
13252	Đỗ Hữu Phúc	VIP 1-1 E-10 - S+	Phan Đức Hải	2.00/10	4.00/10	<input type="button" value=""/>
13251	Bùi Ngọc Phong	VIP 1-1 E-10 - S+	Phan Đức Hải	10.00/10	4.00/10	<input type="button" value=""/>
13250	Ngô Mỹ Uyên	VIP 1-1 E-10 - S+	Phan Đức Hải	3.00/10	4.00/10	<input type="button" value=""/>
13249	Trần Mỹ Quyên	VIP 1-1 E-10 - S+	Phan Đức Hải	3.00/10	1.00/10	<input type="button" value=""/>
13248	Hồ Xuân Long	VIP 1-1 E-10 - S+	Phan Đức Hải	7.00/10	4.00/10	<input type="button" value=""/>
13247	Trần Công Mạnh	VIP 1-1 E-10 - S+	Phan Đức Hải	2.00/10	2.00/10	<input type="button" value=""/>
13246	Hồ Quang Dũng	VIP 1-1 E-10 - S+	Phan Đức Hải	7.00/10	1.00/10	<input type="button" value=""/>
13245	Trần Công Hải	VIP 1-1 E-10 - S+	Phan Đức Hải	1.00/10	2.00/10	<input type="button" value=""/>
13244	Bùi Công Phong	VIP 1-1 E-10 - S+	Phan Đức Hải	2.00/10	4.00/10	<input type="button" value=""/>
13243	Đỗ Ngọc Mai	VIP 1-1 E-10 - S+	Phan Đức Hải	10.00/10	10.00/10	<input type="button" value=""/>
13242	Hồ Mỹ Mai	VIP 1-1 E-10 - S+	Phan Đức Hải	3.00/10	4.00/10	<input type="button" value=""/>
13241	Hoàng Thị Ngân	VIP 1-1 E-10 - S+	Phan Đức Hải	3.00/10	4.00/10	<input type="button" value=""/>
13240	Đỗ Gia Anh	VIP 1-1 E-10 - S+	Phan Đức Hải	3.00/10	3.00/10	<input type="button" value=""/>
13239	Nguyễn Thúy Ly	VIP 1-1 E-10 - S+	Phan Đức Hải	4.00/10	4.00/10	<input type="button" value=""/>
13238	Võ Ngọc Toàn	VIP 1-1 E-10 - S+	Phan Đức Hải	4.00/10	4.00/10	<input type="button" value=""/>

Hình 5.11: Giao diện quản lý đánh giá



Hình 5.12: Giao diện thêm feedback thủ công



Hình 5.13: Giao diện thêm bằng Excel

Phần 6

Tổng kết

6.1 Khó khăn gặp phải và cách khắc phục

- Nhóm không phải ai cũng biết dùng Latex

⇒ Tạo một mẫu trình bày chung để thành viên nhóm dễ dàng trình bày được phần 10 câu SQL Query.

- Việc sinh dữ liệu theo từng dòng khá lâu và mất thời gian. Đặc biệt khi mà cần số lượng bản ghi lớn để index có hiệu quả cũng như đảm bảo không bị lỗi các khóa ngoài, ràng buộc.

⇒ Dùng vòng lặp kết hợp với hàm random để sinh bản ghi.

6.2 Đánh giá kết quả đã đạt được

◊ Ưu điểm:

- Xây dựng được 1 DATABASE hoàn chỉnh , đáp ứng nhu cầu quản lý lớp học, giáo viên, nhân viên ,... của 1 trung tâm trong thực tế .
- Số lượng bản ghi lớn với số lượng bản ghi từ 500 (bảng teacher) tới 146k (bảng attendance)
- Giao diện chương trình trực quan , thân thiện , dễ sử dụng.

◊ Nhược điểm:

- Web mới thiết kế cho quản lý.
- Chưa có chức năng điểm danh trên Web
- Web chưa được mượt mà đối với các dữ liệu có số lượng bản ghi lớn

6.3 Phân công công việc

Bảng 6.1: Phân chia công việc giữa các thành viên

Họ và tên	MSSV	Nhiệm vụ
Nguyễn Thành Bách	20235660	Thiết kế CSDL+Viết mô tả, 3c Trigger, 10c SQL+Thiết kế báo cáo+Code Web
Hoàng Đức Anh	20235640	Thiết kế sơ đồ thực thể - liên kết + Viết 10c SQL Query + Làm slide
Vũ Anh	20235657	Thiết kế sơ đồ quan hệ + Viết 10c SQL Query + Làm slide