Đề tài và mô tả yêu cầu BTL2

Nhóm xây dựng một số trang đơn giản, quản lí **hệ thống e-learning** đã được thiết kế CSDL ở BTL1. Các màn hình nhóm hiện thực bằng UI gồm:

1. Màn hình quản lí các khóa học đã đăng kí của một học sinh.
2. Màn hình quản lí các review khóa học đã được đăng kí của một học sinh.
3. Màn hình quản lí các khóa học của một giảng viên (instructor)

Các công việc cần làm:

- Tạo bảng và dữ liệu mẫu.
- Viết các thủ tục, trigger, hàm.
- Hiện thực ứng dụng.

Tạo bảng và dữ liệu mẫu

Yêu cầu:

- Viết các câu lệnh hiện thực các bảng dữ liệu đã thiết kế, trong đó có các ràng buộc khóa chính, khóa ngoại, các ràng buộc dữ liệu và các ràng buộc ngữ nghĩa nêu trong bài tập lớn 1 (sử dụng check hoặc trigger).
- Tạo dữ liệu mẫu có ý nghĩa ở tất cả các bảng (có thể nhập liệu bằng giao diện hoặc viết câu lệnh)

Kết quả:

- Tạo kiểu dữ liệu type trong PostgresSQL

```
-- CreateEnum
CREATE TYPE "CourseLabel" AS ENUM ('Bestseller', 'HotAndNew', 'New',
'HighestRated');

-- CreateEnum
CREATE TYPE "AudienceLabel" AS ENUM ('Beginner', 'Intermediate', 'Expert',
'AllLevels');

-- CreateEnum
CREATE TYPE "MaterialType" AS ENUM ('Video', 'Text');

-- CreateEnum
CREATE TYPE "AnswerOption" AS ENUM ('A', 'B', 'C', 'D');

-- CreateEnum
CREATE TYPE "PaymentMethod" AS ENUM ('BankTransfer', 'Cash');
```

- Tạo bảng với các trương dữ liệu, kiểu dữ liệu

```
-- CreateTable
CREATE TABLE "User" (
    "id" SERIAL NOT NULL,
    "email" VARCHAR(50) NOT NULL,
    "password" VARCHAR(255) NOT NULL,
    "firstName" VARCHAR(50) NOT NULL,
    "lastName" VARCHAR(50) NOT NULL,
    "avatarUrl" VARCHAR(255) NOT NULL,

    CONSTRAINT "User_pkey" PRIMARY KEY ("id")
```

```
);

-- CreateTable
CREATE TABLE "Student" (
    "userId" INTEGER NOT NULL,
    "target" VARCHAR(255) NOT NULL,

    CONSTRAINT "Student_pkey" PRIMARY KEY ("userId")
);

-- CreateTable
CREATE TABLE "Instructor" (
    "userId" INTEGER NOT NULL,
    "bankAccountNumber" VARCHAR(20) NOT NULL,
    "position" VARCHAR(100) NOT NULL,

    CONSTRAINT "Instructor_pkey" PRIMARY KEY ("userId")
);

-- CreateTable
CREATE TABLE "Course" (
    "id" SERIAL NOT NULL,
    "name" VARCHAR(100) NOT NULL,
    "description" VARCHAR(255),
    "courseLabel" "CourseLabel" NOT NULL,
    "audienceLabel" "AudienceLabel" NOT NULL,
    "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "totalDuration" INTEGER NOT NULL DEFAULT 0,
    "totalSections" INTEGER NOT NULL DEFAULT 0,
    "instructorId" INTEGER NOT NULL,

    CONSTRAINT "Course_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Certificate" (
    "id" SERIAL NOT NULL,
    "content" TEXT NOT NULL,
    "expirationDate" TIMESTAMP(3) NOT NULL,
    "courseId" INTEGER NOT NULL,

    CONSTRAINT "Certificate_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "FreeCourse" (
    "courseId" INTEGER NOT NULL,
    "sponsorName" VARCHAR(100) NOT NULL,

    CONSTRAINT "FreeCourse_pkey" PRIMARY KEY ("courseId")
);

-- CreateTable
CREATE TABLE "PaidCourse" (
```

```
    "courseId" INTEGER NOT NULL,
    "priceOriginal" INTEGER NOT NULL,
    "priceDiscounted" INTEGER NOT NULL,
    "discountPercentage" INTEGER NOT NULL,
    "promoEndDate" TIMESTAMP(3),
    "parentId" INTEGER,

    CONSTRAINT "PaidCourse_pkey" PRIMARY KEY ("courseId")
);

-- CreateTable
CREATE TABLE "Section" (
    "id" SERIAL NOT NULL,
    "name" VARCHAR(100) NOT NULL,
    "totalCompletionTime" INTEGER NOT NULL,
    "totalLectures" INTEGER NOT NULL,
    "courseId" INTEGER NOT NULL,

    CONSTRAINT "Section_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Lecture" (
    "id" SERIAL NOT NULL,
    "name" VARCHAR(100) NOT NULL,
    "description" VARCHAR(255),
    "duration" INTEGER NOT NULL,
    "sectionId" INTEGER NOT NULL,

    CONSTRAINT "Lecture_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Material" (
    "lectureId" INTEGER NOT NULL,
    "type" "MaterialType" NOT NULL,
    "name" VARCHAR(100) NOT NULL,
    "url" VARCHAR(255) NOT NULL,

    CONSTRAINT "Material_pkey" PRIMARY KEY ("lectureId")
);

-- CreateTable
CREATE TABLE "Quiz" (
    "lectureId" INTEGER NOT NULL,
    "totalQuestions" INTEGER NOT NULL,

    CONSTRAINT "Quiz_pkey" PRIMARY KEY ("lectureId")
);

-- CreateTable
CREATE TABLE "Question" (
    "id" SERIAL NOT NULL,
    "content" TEXT NOT NULL,
    "correctOption" "AnswerOption" NOT NULL,
```

```
    "quizId" INTEGER NOT NULL,

    CONSTRAINT "Question_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Order" (
    "id" SERIAL NOT NULL,
    "totalCost" INTEGER NOT NULL,
    "paymentMethod" "PaymentMethod" NOT NULL,
    "studentId" INTEGER NOT NULL,
    "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT "Order_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Category" (
    "name" VARCHAR(100) NOT NULL,
    "content" TEXT NOT NULL,
    "description" VARCHAR(255) NOT NULL,
    "courseId" INTEGER NOT NULL,

    CONSTRAINT "Category_pkey" PRIMARY KEY ("name","courseId")
);

-- CreateTable
CREATE TABLE "StudentReviewCourse" (
    "studentId" INTEGER NOT NULL,
    "courseId" INTEGER NOT NULL,
    "rating" INTEGER NOT NULL CHECK (rating >= 1 and rating <= 5),
    "content" TEXT NOT NULL,
    "createAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT "StudentReviewCourse_pkey" PRIMARY KEY
("studentId","courseId")
);

-- CreateTable
CREATE TABLE "StudentRegisterFreeCourse" (
    "studentId" INTEGER NOT NULL,
    "courseId" INTEGER NOT NULL,

    CONSTRAINT "StudentRegisterFreeCourse_pkey" PRIMARY KEY
("studentId","courseId")
);

-- CreateTable
CREATE TABLE "PaidCourseOrder" (
    "paidCourseId" INTEGER NOT NULL,
    "orderId" INTEGER NOT NULL,

    CONSTRAINT "PaidCourseOrder_pkey" PRIMARY KEY ("paidCourseId","orderId")
);
```

```
-- CreateTable
CREATE TABLE "Answer" (
    "questionId" INTEGER NOT NULL,
    "answerOption" "AnswerOption" NOT NULL,
    "content" TEXT NOT NULL,
    "isCorrect" BOOLEAN NOT NULL,
    "explanation" TEXT NOT NULL,

    CONSTRAINT "Answer_pkey" PRIMARY KEY ("questionId","answerOption")
);
```

- Tạo các mối quan hệ khóa chính, khóa ngoại giữa các bảng

```
-- AddForeignKey
ALTER TABLE "Student" ADD CONSTRAINT "Student_userId_fkey" FOREIGN KEY
("userId") REFERENCES "User"("id") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "Instructor" ADD CONSTRAINT "Instructor_userId_fkey" FOREIGN KEY
("userId") REFERENCES "User"("id") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "Course" ADD CONSTRAINT "Course_instructorId_fkey" FOREIGN KEY
("instructorId") REFERENCES "Instructor"("userId") ON DELETE CASCADE ON
UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "Certificate" ADD CONSTRAINT "Certificate_courseId_fkey" FOREIGN
KEY ("courseId") REFERENCES "Course"("id") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "FreeCourse" ADD CONSTRAINT "FreeCourse_courseId_fkey" FOREIGN
KEY ("courseId") REFERENCES "Course"("id") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "PaidCourse" ADD CONSTRAINT "PaidCourse_courseId_fkey" FOREIGN
KEY ("courseId") REFERENCES "Course"("id") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "PaidCourse" ADD CONSTRAINT "PaidCourse_parentId_fkey" FOREIGN
KEY ("parentId") REFERENCES "PaidCourse"("courseId") ON DELETE SET NULL ON
UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "Section" ADD CONSTRAINT "Section_courseId_fkey" FOREIGN KEY
("courseId") REFERENCES "Course"("id") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "Lecture" ADD CONSTRAINT "Lecture_sectionId_fkey" FOREIGN KEY
("sectionId") REFERENCES "Section"("id") ON DELETE CASCADE ON UPDATE
CASCADE;
```

```
-- AddForeignKey
ALTER TABLE "Material" ADD CONSTRAINT "Material_lectureId_fkey" FOREIGN KEY
("lectureId") REFERENCES "Lecture"("id") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "Quiz" ADD CONSTRAINT "Quiz_lectureId_fkey" FOREIGN KEY
("lectureId") REFERENCES "Lecture"("id") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "Question" ADD CONSTRAINT "Question_quizId_fkey" FOREIGN KEY
("quizId") REFERENCES "Quiz"("lectureId") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "Order" ADD CONSTRAINT "Order_studentId_fkey" FOREIGN KEY
("studentId") REFERENCES "Student"("userId") ON DELETE CASCADE ON UPDATE
CASCADE;

-- AddForeignKey
ALTER TABLE "Category" ADD CONSTRAINT "Category_courseId_fkey" FOREIGN KEY
("courseId") REFERENCES "Course"("id") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "StudentReviewCourse" ADD CONSTRAINT
"StudentReviewCourse_studentId_fkey" FOREIGN KEY ("studentId") REFERENCES
"Student"("userId") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "StudentReviewCourse" ADD CONSTRAINT
"StudentReviewCourse_courseId_fkey" FOREIGN KEY ("courseId") REFERENCES
"Course"("id") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "StudentRegisterFreeCourse" ADD CONSTRAINT
"StudentRegisterFreeCourse_studentId_fkey" FOREIGN KEY ("studentId")
REFERENCES "Student"("userId") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "StudentRegisterFreeCourse" ADD CONSTRAINT
"StudentRegisterFreeCourse_courseId_fkey" FOREIGN KEY ("courseId")
REFERENCES "Course"("id") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "PaidCourseOrder" ADD CONSTRAINT
"PaidCourseOrder_paidCourseId_fkey" FOREIGN KEY ("paidCourseId") REFERENCES
"PaidCourse"("courseId") ON DELETE CASCADE ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "PaidCourseOrder" ADD CONSTRAINT "PaidCourseOrder_orderId_fkey"
FOREIGN KEY ("orderId") REFERENCES "Order"("id") ON DELETE CASCADE ON UPDATE
CASCADE;
```

```
-- AddForeignKey
ALTER TABLE "Answer" ADD CONSTRAINT "Answer_questionId_fkey" FOREIGN KEY
("questionId") REFERENCES "Question"("id") ON DELETE CASCADE ON UPDATE
CASCADE;
```

- Ảnh dữ liệu trong các bảng

Bảng dữ liệu User

id [PK] integer	email character varying (50)	password character varying (255)	firstName character varying (50)	lastName character varying (50)	avatarUrl character varying (255)
1	hluong@gmail.com	$2b$10$0LtqsgPj1Oi8DKHpsMfyWuj5vhxGYfhm6lyAufS1XXZNzjTbGCd...	Hoàng	Lương	https://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d...
2	lvu@gmail.com	$2b$10$0LtqsgPj1Oi8DKHpsMfyWuj5vhxGYfhm6lyAufS1XXZNzjTbGCd...	Lâm	Vũ	https://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d...
3	nnphu@gmail.com	$2b$10$0LtqsgPj1Oi8DKHpsMfyWuj5vhxGYfhm6lyAufS1XXZNzjTbGCd...	Nguyễn Ngọc	Phú	https://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d...
4	ntduy@gmail.com	$2b$10$0LtqsgPj1Oi8DKHpsMfyWuj5vhxGYfhm6lyAufS1XXZNzjTbGCd...	Nguyễn Tuấn	Duy	https://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d...
5	tmthuany@gmail.com	$2b$10$0LtqsgPj1Oi8DKHpsMfyWuj5vhxGYfhm6lyAufS1XXZNzjTbGCd...	Trần Minh	Thuận	https://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d...

Bảng dữ liệu Student

	userId [PK] integer	target character varying (255)
1	1	Software Engineer
2	3	Database Administrator At Google
3	4	Fullstack Developer
4	5	Backend Developer

Bảng dữ liệu Instructor

	userId [PK] integer	bankAccountNumber character varying (20)	position character varying (100)
1	2	9374882123	Software Engineer At Google
2	3	9374882124	AI Researcher At OpenAI

Bảng dữ liệu Course

id [PK] integer	name character varying (100)	description character varying (255)	courseLabel "CourseLabel"	audienceLabel "AudienceLabel"	createdAt timestamp without time zone (3)	updatedAt timestamp without time zone (3)	totalDuration integer	totalSections integer	instructorId integer
1	HTML for Beginners	Learn HTML to become a fullstack developer	HighestRated	Beginner	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	2
2	SQL for Beginners	Learn SQL from zero to hero	Bestseller	Beginner	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	2200	2	2
3	PostgreSQL to in depth	Learn PostgreSQL to become a DBA	Bestseller	Intermediate	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	2
4	CSS for Beginners	Learn CSS to become a frontend developer	HighestRated	Beginner	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	2
5	JavaScript for Beginners	Learn JavaScript to become a fullstack developer	New	Intermediate	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	3
6	Python for Beginners	Learn Python to become a backend developer	HotAndNew	Beginner	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	2
7	Java for Beginners	Learn Java to become a backend developer	Bestseller	AllLevels	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	3
8	C# for Beginners	Learn C# to become a backend developer	HighestRated	Beginner	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	2
9	C++ for Beginners	Learn C++ to become a backend developer	New	AllLevels	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	2
10	PHP for Beginners	Learn PHP to become a backend developer	HighestRated	Expert	2023-12-14 11:19:45.321	2023-12-14 11:19:45.321	0	0	3

Bảng dữ liệu Certificate

id [PK] integer	content text	expirationDate timestamp without time zone (3)	courseId integer
1	Congratulation! You have completed SQL for Beginners course	2025-12-31 00:00:00	2
2	Congratulation! You have completed PostgreSQL to in depth cour...	2025-12-31 00:00:00	3

Bảng dữ liệu FreeCourse

courseId [PK] integer	sponsorName character varying (100)
1	Google
4	Google
7	Microsoft
10	Microsoft

Bảng dữ liệu PaidCourse

courseId [PK] integer	priceOriginal integer	priceDiscounted integer	discountPercentage integer	promoEndDate timestamp without time zone (3)	parentId integer
2	100000	50000	50	[null]	[null]
3	200000	160000	20	[null]	2
5	100000	50000	50	[null]	[null]
6	200000	160000	20	[null]	5
8	100000	50000	50	[null]	[null]
9	200000	160000	20	[null]	8

Bảng dữ liệu Section

id [PK] integer	name character varying (100)	totalCompletionTime integer	totalLectures integer	courseId integer
1	Data Definition Language (DDL)	1600	2	2
2	Data Manipulation Language (DML)	600	1	2

, height: 10% Bảng dữ liệu Lecture

id [PK] integer	name character varying (100)	description character varying (255)	duration integer	sectionId integer
1	Create Table	Learn how to create a table in SQL	1000	1
2	Alter Table	Learn how to alter a table in SQL	600	1
3	Quiz INSERT	Synthetic quiz for INSERT statement	600	2

Bảng dữ liệu Material

er	type "MaterialType"	name character varying (100)	url character varying (255)
1	Video	Video tutorial	https://www.youtube.com/watch?v=QnBp4NjUQPU
2	Text	PDF tutorial	https://www.tutorialspoint.com/sql/sql-alter-comm

Bảng dữ liệu Quiz

lectureId [PK] integer	totalQuestions integer
3	1

Bảng dữ liệu Order

id [PK] integer	totalCost integer	paymentMethod "PaymentMethod"	studentId integer	createdAt timestamp without time zone (3)
1	630000	Cash	1	2023-12-12 08:19:15.06

Bảng dữ liệu Category

name [PK] character varying (100)	content text	description character varying (255)	courseId [PK] integer
Database	How to become a database administrator	Database category	2
Database	How to become a database administrator	Database category	3
Database	How to become a database administrator	Database category	6
Database	How to become a database administrator	Database category	8
Database	How to become a database administrator	Database category	10
Web development	How to become a web developer	Web development category	1
Web development	How to become a web developer	Web development category	2
Web development	How to become a web developer	Web development category	4
Web development	How to become a web developer	Web development category	5
Web development	How to become a web developer	Web development category	7
Web development	How to become a web developer	Web development category	9

Bảng dữ liệu StudentReviewCourse

studentId [PK] integer	courseId [PK] integer	rating integer	content text	createAt timestamp without time zone (3)
1	1	3	It is so easy	2023-12-12 08:19:15.072
1	2	5	This course is aweso...	2023-12-12 08:19:15.072
1	3	4	This course is not bad	2023-12-12 08:19:15.072

Bảng dữ liệu StudentRegisterFreeCourse

studentId [PK] integer	courseId [PK] integer
1	1
1	4
1	7
1	10

Bảng dữ liệu PaidCourseOrder

paidCourseId [PK] integer	orderId [PK] integer
2	1
3	1
5	1
6	1
8	1
9	1

Bảng dữ liệu Answer

questionId [PK] integer	answerOption [PK] "AnswerOption"	content text	isCorrect boolean	explanation text
1	A	INSERT INTO table_name VALUES (value1, value2, value3, ...);	true	This is the correct syntax for INSERT statement
1	B	INSERT INTO table_name (column1, column2, column3, ...)	false	This is the incorrect syntax for INSERT stateme...
1	C	INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...	false	This is the incorrect syntax for INSERT stateme...
1	D	All of the above	false	This is the incorrect syntax for INSERT stateme...

Thủ tục - trigger - hàm

Viết thủ tục INSERT/UPDATE/DELETE

Yêu cầu:

Viết các thủ tục để thêm (insert), sửa (update), xóa (delete) dữ liệu vào MỘT bảng dữ liệu.

- Phải có thực hiện việc kiểm tra dữ liệu hợp lệ (validate) để đảm bảo các ràng buộc của bảng dữ liệu.
- Xuất ra thông báo lỗi có nghĩa, chỉ ra được lỗi sai cụ thể (không ghi chung chung là "Lỗi nhập dữ liệu!").

Kết quả:

- Học sinh thêm 1 review vào khóa học mà học sinh đó đã tham gia (hoặc đăng kí) - **INSERT**

```
        -- Insert
CREATE OR REPLACE FUNCTION validate_review_parameters(
            p_student_id INTEGER,
            p_course_id INTEGER,
            p_rating INTEGER,
            p_content TEXT
        ) RETURNS TEXT AS $$
        BEGIN
            IF p_student_id IS NULL THEN
                RETURN 'Học sinh không được để trống';
            ELSIF p_course_id IS NULL THEN
                RETURN 'Khóa học không được để trống';
            ELSIF p_rating IS NULL THEN
                RETURN 'Điểm đánh giá không được để trống';
            ELSIF p_content IS NULL THEN
                RETURN 'Nội dung đánh giá không được để trống';
            END IF;

            RETURN NULL;
        END;
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION is_exist_student_or_course(_student_id INTEGER,
_course_id INTEGER) RETURNS TEXT AS $$
        DECLARE
            student_exists BOOLEAN;
            course_exists BOOLEAN;
        BEGIN
            student_exists := EXISTS (
                SELECT 1
                FROM "Student"
                WHERE "userId" = _student_id
            );

            course_exists := EXISTS (
                SELECT 1
                FROM "Course"
                WHERE "id" = _course_id
            );

            IF NOT student_exists AND NOT course_exists THEN
                RETURN 'Không tồn tại học sinh và khóa học này';
            ELSIF NOT student_exists THEN
                RETURN 'Không tồn tại học sinh này';
            ELSIF NOT course_exists THEN
```

```
                RETURN 'Không tồn tại khóa học này';
            END IF;

            RETURN NULL;
        END;
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION is_course_registered(_student_id INTEGER,
_course_id INTEGER) RETURNS BOOLEAN AS $$
        BEGIN
            RETURN EXISTS (
                SELECT 1
                FROM "StudentRegisterFreeCourse"
                WHERE "studentId" = _student_id AND "courseId" = _course_id
            ) OR EXISTS (
                SELECT
                    1
                FROM
                    "PaidCourseOrder" AS pco
                    INNER JOIN "Order" AS o ON pco."orderId" = o."id"
                WHERE
                    pco."paidCourseId" = _course_id
                    AND o."studentId" = _student_id
            );
        END;
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION is_duplicate_pk(_student_id INTEGER, _course_id
INTEGER) RETURNS BOOLEAN AS $$
        BEGIN
            RETURN EXISTS (
                SELECT 1
                FROM "StudentReviewCourse"
                WHERE "studentId" = _student_id AND "courseId" = _course_id
            );
        END;
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION is_valid_rating(_rating INTEGER) RETURNS BOOLEAN
AS $$
        BEGIN
            RETURN _rating > 0 AND _rating <= 5;
        END;
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE insert_review(
            p_student_id INTEGER,
            p_course_id INTEGER,
            p_rating INTEGER,
            p_content TEXT
        ) AS $$
        DECLARE
            validation_error TEXT;
            exist_error TEXT;
```

```
        BEGIN
            validation_error := validate_review_parameters(p_student_id,
p_course_id, p_rating, p_content);
            IF validation_error IS NOT NULL THEN
                RAISE EXCEPTION '%', validation_error;
            END IF;

            exist_error := is_exist_student_or_course(p_student_id,
p_course_id);
            IF exist_error IS NOT NULL THEN
                RAISE EXCEPTION '%', exist_error;
            END IF;

            IF NOT is_course_registered(p_student_id, p_course_id) THEN
                RAISE EXCEPTION 'Học sinh chưa đăng ký khóa học này !';
            END IF;

            IF is_duplicate_pk(p_student_id, p_course_id) THEN
                RAISE EXCEPTION 'Học sinh đã review khóa học này !';
            END IF;

            IF NOT is_valid_rating(p_rating) THEN
                RAISE EXCEPTION 'Điểm rating không hợp lệ !';
            END IF;

            INSERT INTO
                "StudentReviewCourse"("studentId", "courseId", "rating",
"content")
            VALUES
                (p_student_id, p_course_id, p_rating, p_content);
        END;
        $$ LANGUAGE plpgsql;
```

Mô tả:

1. Hàm ***validate_review_parameters***: Hàm này được sử dụng để kiểm tra tính hợp lệ của các tham số đầu vào cho việc thêm đánh giá của học sinh. Nếu bất kỳ tham số nào là NULL, hàm sẽ trả về một thông báo lỗi tương ứng. Nếu không có lỗi, nó sẽ trả về NULL.
2. Hàm ***is_exist_student_or_course***: Hàm này kiểm tra xem một học sinh và một khóa học có tồn tại trong cơ sở dữ liệu không. Nếu cả hai không tồn tại, nó trả về thông báo lỗi "Không tồn tại học sinh và khóa học này". Nếu chỉ một trong hai không tồn tại, nó trả về thông báo lỗi tương ứng. Nếu cả hai tồn tại hoặc không có lỗi, nó trả về NULL.
3. Hàm ***is_course_registered***: Hàm này kiểm tra xem một học sinh đã đăng ký khóa học hay chưa. Nếu học sinh đã đăng ký (được xác định bằng việc tìm thấy dòng tương ứng trong các bảng "StudentRegisterFreeCourse" hoặc "PaidCourseOrder"), nó trả về TRUE; ngược lại, nó trả về FALSE.
4. Hàm ***is_duplicate_pk***: Hàm này kiểm tra xem một học sinh đã đánh giá khóa học này trước đó chưa, bằng cách kiểm tra sự tồn tại của một dòng trong bảng "StudentReviewCourse" có khóa chính tương ứng. Nếu đã tồn tại, nó trả về TRUE; ngược lại, nó trả về FALSE.

5. Hàm ***is_valid_rating***: Hàm này kiểm tra xem điểm đánh giá có hợp lệ hay không. Trong trường hợp này, điểm hợp lệ được xác định là điểm từ 1 đến 5. Nếu điểm hợp lệ, hàm trả về TRUE; ngược lại, nó trả về FALSE.
6. Procedure ***insert_review***: Procedure này thực hiện việc thêm một bản đánh giá mới vào bảng "StudentReviewCourse". Trước khi thêm, nó kiểm tra tính hợp lệ của các tham số đầu vào sử dụng các hàm đã được định nghĩa ở trên. Nếu có lỗi xảy ra (ví dụ: tham số NULL, học sinh chưa đăng ký khóa học, đã đánh giá trước đó, hoặc điểm đánh giá không hợp lệ), procedure sẽ ném ra một ngoại lệ với thông báo lỗi tương ứng. Nếu không có lỗi, nó thêm dòng đánh giá mới vào bảng "StudentReviewCourse".

- Thực update 1 review - **UPDATE**

```
CREATE OR REPLACE PROCEDURE update_review(
    p_student_id INTEGER,
    p_course_id INTEGER,
    p_rating INTEGER,
    p_content TEXT
) AS $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM "Student" WHERE "userId" = p_student_id)
THEN
        RAISE EXCEPTION 'Không tìm thấy học sinh !';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM "Course" WHERE "id" = p_course_id) THEN
        RAISE EXCEPTION 'Không tìm thấy khóa học !';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM "StudentReviewCourse" WHERE "studentId" =
p_student_id AND "courseId" = p_course_id) THEN
        RAISE EXCEPTION 'Không tìm thấy review của học sinh này trong khóa
học !';
    END IF;

    IF p_rating IS NOT NULL AND (p_rating <= 0 OR p_rating > 5) THEN
        RAISE EXCEPTION 'Điểm rating không hợp lệ !';
    END IF;

    UPDATE
        "StudentReviewCourse"
    SET
        "rating" = COALESCE(p_rating, "rating"),
        "content" = COALESCE(p_content, "content")
    WHERE
        "studentId" = p_student_id
        AND "courseId" = p_course_id;
END;
$$ LANGUAGE plpgsql;
```

Mô tả:

1. Procedure ***update_review*** nhận vào bốn tham số: ***p_student_id*** (ID của học sinh), ***p_course_id*** (ID của khóa học), ***p_rating*** (điểm đánh giá mới), và ***p_content*** (nội dung đánh giá mới).

2. Đầu tiên, procedure kiểm tra xem học sinh (có ID là ***p_student_id***) và khóa học (có ID là ***p_course_id***) có tồn tại trong cơ sở dữ liệu không. Nếu không tìm thấy học sinh hoặc khóa học tương ứng, procedure sẽ ném ra một ngoại lệ với thông báo lỗi tương ứng.
3. Tiếp theo, procedure kiểm tra xem có tồn tại một đánh giá của học sinh (***p_student_id***) cho khóa học (***p_course_id***) trong bảng "StudentReviewCourse" không. Nếu không tìm thấy, procedure cũng ném ra một ngoại lệ với thông báo lỗi "Không tìm thấy review của học sinh này trong khóa học !".
4. Procedure kiểm tra tính hợp lệ của ***p_rating*** (điểm đánh giá mới). Nếu ***p_rating*** không null và không nằm trong khoảng từ 1 đến 5, procedure sẽ ném ra một ngoại lệ với thông báo lỗi "Điểm rating không hợp lệ !".
5. Cuối cùng, nếu không có lỗi nào xảy ra trong quá trình kiểm tra, procedure sẽ thực hiện cập nhật bản đánh giá trong bảng "StudentReviewCourse". Nếu ***p_rating*** hoặc ***p_content*** là null, sẽ sử dụng giá trị hiện tại của cột tương ứng. Cập nhật sẽ được thực hiện dựa trên điều kiện "studentId" và "courseId" của bản đánh giá để đảm bảo rằng chỉ bản đánh giá của học sinh cụ thể cho khóa học cụ thể được cập nhật.

- Thực hiện xóa 1 review - **DELETE**

```
CREATE OR REPLACE PROCEDURE delete_review(
    p_student_id INTEGER,
    p_course_id INTEGER
) AS $$
BEGIN
IF NOT EXISTS (SELECT 1 FROM "Student" WHERE "userId" = p_student_id) THEN
    RAISE EXCEPTION 'Không tìm thấy học sinh !';
END IF;

IF NOT EXISTS (SELECT 1 FROM "Course" WHERE "id" = p_course_id) THEN
    RAISE EXCEPTION 'Không tìm thấy khóa học !';
END IF;

IF NOT EXISTS (SELECT 1 FROM "StudentReviewCourse" WHERE "studentId" =
p_student_id AND "courseId" = p_course_id) THEN
    RAISE EXCEPTION 'Không tìm thấy review của học sinh này trong khóa
học !';
END IF;

DELETE
    FROM "StudentReviewCourse"
WHERE
    "studentId" = p_student_id
    AND "courseId" = p_course_id;
END;
$$ LANGUAGE plpgsql;
```

Mô tả:

1. Procedure ***delete_review*** nhận vào hai tham số: ***p_student_id*** (ID của học sinh) và ***p_course_id*** (ID của khóa học) mà bạn muốn xóa đánh giá của học sinh đó.

2. Đầu tiên, procedure kiểm tra xem học sinh (có ID là ***p_student_id***) và khóa học (có ID là ***p_course_id***) có tồn tại trong cơ sở dữ liệu không. Nếu không tìm thấy học sinh hoặc khóa học tương ứng, procedure sẽ ném ra một ngoại lệ với thông báo lỗi tương ứng.
3. Sau đó, procedure kiểm tra xem có tồn tại một đánh giá của học sinh (***p_student_id***) cho khóa học (***p_course_id***) trong bảng "StudentReviewCourse" không. Nếu không tìm thấy, procedure cũng ném ra một ngoại lệ với thông báo lỗi "Không tìm thấy review của học sinh này trong khóa học !".
4. Nếu không có lỗi nào xảy ra trong quá trình kiểm tra, procedure sẽ thực hiện lệnh DELETE để xóa bản đánh giá cụ thể trong bảng "StudentReviewCourse". Xóa sẽ được thực hiện dựa trên điều kiện "studentId" và "courseId" của bản đánh giá để đảm bảo rằng chỉ bản đánh giá của học sinh cụ thể cho khóa học cụ thể được xóa.

- Lấy review từ học sinh - **GET**

```
CREATE OR REPLACE FUNCTION get_review_by_student(p_student_id INTEGER)
RETURNS TABLE(
        "courseId" INTEGER,
        "studentId" INTEGER,
        "courseName" VARCHAR(100),
        "categoryName" VARCHAR(100)[],
        rating INTEGER,
        content TEXT,
        "createdAt" TIMESTAMP(3)
    ) AS $$
    BEGIN
        RETURN QUERY
        SELECT
            src."courseId",
            src."studentId",
            cr."name" AS "courseName",
            array_agg(cat."name") AS "categoryName",
            src."rating",
            src."content",
            src."createAt"
        FROM
            "StudentReviewCourse" AS src
            INNER JOIN "Course" AS cr ON src."courseId" = cr."id"
            LEFT JOIN "Category" AS cat ON src."courseId" = cat."courseId"
        WHERE
            src."studentId" = p_student_id
        GROUP BY
            cr."name",
            src."rating",
            src."content",
            src."createAt",
            src."courseId",
            src."studentId";
    END;
$$ LANGUAGE plpgsql;
```

Mô tả:

1. Hàm ***get_review_by_student*** nhận một tham số là ***p_student_id***, là ID của học sinh mà bạn muốn truy vấn đánh giá của.
2. Hàm trả về một bảng kết quả với các cột sau:

"courseId": ID của khóa học mà đánh giá liên quan đến. "studentId": ID của học sinh. "courseName": Tên của khóa học được đánh giá. "categoryName": Mảng chứa tên các danh mục liên quan đến khóa học. "rating": Điểm đánh giá. "content": Nội dung đánh giá. "createdAt": Thời gian tạo đánh giá.

3. Hàm sử dụng lệnh SQL để truy vấn dữ liệu từ các bảng "StudentReviewCourse," "Course," và "Category." Hàm thực hiện các thao tác sau:
    - Kết nối bảng "StudentReviewCourse" với bảng "Course" dựa trên trường "courseId."
    - Thực hiện LEFT JOIN với bảng "Category" dựa trên trường "courseId" để lấy danh sách các danh mục liên quan đến khóa học.
    - Lọc kết quả để chỉ lấy các đánh giá mà có "studentId" trùng với tham số đầu vào p_student_id.
    - Sử dụng GROUP BY để tổng hợp kết quả theo "courseName," "rating," "content," "createdAt," "courseId," và "studentId."
4. Cuối cùng, hàm trả về kết quả của truy vấn dưới dạng một bảng chứa thông tin về các đánh giá của học sinh cho các khóa học tương ứng.

Viết trigger

- **Yêu cầu**

  Viết 2 trigger để kiểm soát các hành động INSERT, UPDATE, DELETE trên một số bảng đã tạo thỏa mãn yêu cầu sau:

  - Có ít nhất 1 trigger có tính toán cập nhật dữ liệu trên bảng dữ liệu khác bảng đang được thiết lập trigger. (Trigger liên quan đến việc tính toán thuộc tính dẫn xuất)

  - Chuẩn bị câu lệnh và dữ liệu minh họa cho việc kiểm tra trigger khi báo cáo.

- **Kết quả**

  Trigger 1: Thuộc tính dẫn xuất trong cùng một bảng

  Mô tả:
  - Giá của một khóa học có phí sẽ được tự động cập nhật bằng giá gốc trừ giá khuyến mãi.
  - Trigger này sẽ được kích hoạt khi thêm mới hoặc có sự cập nhật của giá gốc hoặc phần trăm khuyến mãi hoặc ngày hết hạn khuyến mãi của một khóa học có phí (PaidCourse)
  - Nếu hết thời gian khuyến mãi vượt quá hạn thì tự động cập nhật tỉ lệ giảm giá là 0%

  Code:

```
-- Create trigger function
CREATE OR REPLACE FUNCTION update_paid_course_price() RETURNS
TRIGGER AS $$
    BEGIN
    IF NEW."promoEndDate" IS NOT NULL AND NEW."promoEndDate" < NOW()
THEN
```

```
        NEW."discountPercentage" := 0;
    END IF;
    NEW."priceDiscounted" := NEW."priceOriginal" * (100-
NEW."discountPercentage")/100;
    RETURN NEW;
    END;
    $$ LANGUAGE PLPGSQL;

-- Create trigger
CREATE OR REPLACE TRIGGER paid_course_price_update
    BEFORE
    INSERT
    OR
    UPDATE OF "priceOriginal",
            "discountPercentage",
            "promoEndDate" ON "PaidCourse"
    FOR EACH ROW EXECUTE FUNCTION update_paid_course_price();
```

Trigger 2: Thuộc tính dẫn xuất khác bảng

Mô tả:

- Tổng số tiền của một đơn hàng sẽ được tự động cập nhật bằng tổng giá bản của các khóa học có phí đã được thêm vào đơn hàng
- Trigger này sẽ được kích hoạt khi thêm hoặc xóa một khóa học có phí khỏi một đơn hàng

Code:

```
CREATE OR REPLACE FUNCTION update_total_cost() RETURNS TRIGGER AS $
$
    DECLARE
    order_id INT;
    BEGIN
        IF TG_OP = 'DELETE' THEN
            order_id := OLD."orderId";
        ELSE
            order_id := NEW."orderId";
        END IF;

        UPDATE "Order" o
        SET "totalCost" = (
            SELECT COALESCE(SUM("priceDiscounted"), 0)
            FROM "PaidCourse" pc
            JOIN "PaidCourseOrder" pco ON pc."courseId" =
pco."paidCourseId"
            WHERE pco."orderId" = order_id
        )
        WHERE o.id = order_id;
        RETURN NULL;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER total_cost_update AFTER
INSERT
```

```
OR
DELETE ON "PaidCourseOrder"
FOR EACH ROW EXECUTE FUNCTION update_total_cost();
```

Trigger 3->8: trigger cho các thuộc tính dẫn xuất khác

- Tạo các trigger cho các thuộc tính dẫn xuất(2 trigger demo cho phần 1.2.2 sẽ không show ở phần này)

Mô tả:

Trigger 3 và hàm update_correct_answer:

- Trigger **correct_answer_update** được kích hoạt sau khi có sự thay đổi (INSERT hoặc UPDATE) trên bảng "Answer" và liên quan đến cột "isCorrect".
- Hàm **update_correct_answer** được gọi bởi trigger này và được sử dụng để cập nhật cột "correctOption" trong bảng "Question".
- Trigger này giúp đảm bảo rằng "correctOption" trong bảng "Question" sẽ luôn thể hiện câu trả lời đúng (được lấy từ bảng "Answer").

Trigger 4 và hàm update_total_course_duration:

- Trigger **total_duration_course_update** được kích hoạt sau khi có sự thay đổi (INSERT, UPDATE, hoặc DELETE) trên bảng "Section" và liên quan đến cột "totalCompletionTime".
- Hàm **update_total_course_duration** được gọi bởi trigger này và được sử dụng để cập nhật cột "totalDuration" trong bảng "Course".
- Trigger này cập nhật tổng thời lượng của một khóa học dựa trên tổng thời lượng các phần ("Section") trong khóa học.

Trigger 5 và hàm update_total_duration_section:

- Trigger **total_duration_section_update** được kích hoạt sau khi có sự thay đổi (INSERT, UPDATE, hoặc DELETE) trên bảng "Lecture" và liên quan đến cột "duration".
- Hàm **update_total_duration_section** được gọi bởi trigger này và được sử dụng để cập nhật cột "totalCompletionTime" trong bảng "Section".
- Trigger này cập nhật tổng thời lượng của một phần ("Section") dựa trên tổng thời lượng các bài giảng ("Lecture") trong phần đó.

Trigger 6 và hàm update_total_section:

- Trigger **total_section_update** được kích hoạt sau khi có sự thay đổi (INSERT hoặc DELETE) trên bảng "Section".
- Hàm **update_total_section** được gọi bởi trigger này và được sử dụng để cập nhật cột "totalSections" trong bảng "Course".
- Trigger này cập nhật tổng số các phần trong khóa học.

Trigger 7 và hàm update_total_lecture:

- Trigger **total_lecture_update** được kích hoạt sau khi có sự thay đổi (INSERT hoặc DELETE) trên bảng "Lecture".
- Hàm **update_total_lecture** được gọi bởi trigger này và được sử dụng để cập nhật cột "totalLectures" trong bảng "Section".
- Trigger này cập nhật tổng số bài giảng trong một phần.

Trigger 8 và hàm update_total_question:

Trigger **total_question_update** được kích hoạt sau khi có sự thay đổi (INSERT hoặc DELETE) trên bảng "Question". Hàm **update_total_question** được gọi bởi trigger này và được sử dụng để cập nhật cột "totalQuestions" trong bảng "Quiz". Trigger này cập nhật tổng số câu hỏi trong một bài kiểm tra ("Quiz").

Code:

```
-- Trigger update correct answer
CREATE OR REPLACE FUNCTION update_correct_answer() RETURNS TRIGGER
AS $$
    BEGIN
    UPDATE "Question"
    SET "correctOption" = COALESCE((SELECT "answerOption" FROM
"Answer" WHERE "questionId" = NEW."questionId" AND "isCorrect" =
TRUE LIMIT 1),'A')
    WHERE id = NEW."questionId";
    RETURN NULL;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER correct_answer_update AFTER
INSERT
OR
UPDATE OF "isCorrect" ON "Answer"
FOR EACH ROW EXECUTE FUNCTION update_correct_answer();

-- Trigger update duration of course
CREATE OR REPLACE FUNCTION update_total_course_duration() RETURNS
TRIGGER AS $$
    DECLARE
    course_id INT;
    BEGIN
    IF TG_OP = 'DELETE' THEN
        course_id := OLD."courseId";
    ELSE
        course_id := NEW."courseId";
    END IF;

    UPDATE "Course"
    SET "totalDuration" = (
        SELECT SUM("totalCompletionTime")
        FROM "Section"
        WHERE "courseId" = course_id
    )
    WHERE id = course_id;
    RETURN NULL;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER total_duration_course_update AFTER
INSERT
OR
UPDATE OF "totalCompletionTime"
```

```
OR
DELETE ON "Section"
FOR EACH ROW EXECUTE FUNCTION update_total_course_duration();

-- Trigger update duration of section
CREATE OR REPLACE FUNCTION update_total_duration_section() RETURNS
TRIGGER AS $$
    DECLARE
    section_id INT;
    BEGIN
    IF TG_OP = 'DELETE' THEN
        section_id := OLD."sectionId";
    ELSE
        section_id := NEW."sectionId";
    END IF;

    UPDATE "Section"
    SET "totalCompletionTime" = (
        SELECT SUM("duration")
        FROM "Lecture"
        WHERE "sectionId" = section_id
    )
    WHERE id = section_id;
    RETURN NULL;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER total_duration_section_update AFTER
INSERT
OR
UPDATE OF duration
OR
DELETE ON "Lecture"
FOR EACH ROW EXECUTE FUNCTION update_total_duration_section();

--Trigger update total section of course
CREATE OR REPLACE FUNCTION update_total_section() RETURNS TRIGGER
AS $$
    DECLARE
    course_id INT;
    BEGIN
    IF TG_OP = 'DELETE' THEN
        course_id := OLD."courseId";
    ELSE
        course_id := NEW."courseId";
    END IF;
    UPDATE "Course"
    SET "totalSections" = (
        SELECT COUNT(*)
        FROM "Section"
        WHERE "courseId" = course_id
    )
    WHERE id = course_id;
    RETURN NULL;
    END;
```

```
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER total_section_update AFTER
INSERT
OR
DELETE ON "Section"
FOR EACH ROW EXECUTE FUNCTION update_total_section();

--Trigger update total lucture of section
CREATE OR REPLACE FUNCTION update_total_lecture() RETURNS TRIGGER
AS $$
    DECLARE
    section_id INT;
    BEGIN
    IF TG_OP = 'DELETE' THEN
    section_id := OLD."sectionId";
    ELSE
    section_id := NEW."sectionId";
    END IF;
    UPDATE "Section"
    SET "totalLectures" = (SELECT COUNT(*) FROM "Lecture" WHERE
"sectionId" = section_id)
    WHERE id = section_id;
    RETURN NULL;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER total_lecture_update AFTER
INSERT
OR
DELETE ON "Lecture"
FOR EACH ROW EXECUTE FUNCTION update_total_lecture();

-- Trigger update total question of quiz
CREATE OR REPLACE FUNCTION update_total_question() RETURNS TRIGGER
AS $$
    DECLARE
    quiz_id INT;
    BEGIN

    IF TG_OP = 'DELETE' THEN
        quiz_id := OLD."quizId";
    ELSE
        quiz_id := NEW."quizId";
    END IF;

    UPDATE "Quiz"
    SET "totalQuestions" = (SELECT COUNT(*) FROM "Question" WHERE
"quizId" = quiz_id)
    WHERE "lectureId" = quiz_id;
    RETURN NULL;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER total_question_update AFTER
```

```
INSERT
OR
DELETE ON "Question"
FOR EACH ROW EXECUTE FUNCTION update_total_question();
```

Viết hàm

- **Yêu cầu:**

Viết 2 hàm thỏa yêu cầu sau:

- Chứa câu lệnh IF và/hoặc LOOP để tính toán dữ liệu được lưu trữ.
- Chứa câu lệnh truy vấn dữ liệu, lấy dữ liệu từ câu truy vấn để kiểm tra tính toán.
- Có tham số đầu vào và kiểm tra tham số đầu vào.
- Chuẩn bị các câu lệnh và dữ liệu để minh họa việc gọi hàm khi báo cáo.

- **Kết quả:**

  - **Hàm 1:** Tên khóa học có điểm rating trung bình cao nhất cùng với điểm rating đó theo từng loại đối tượng khóa học.
    - Input: là nhãn đối tượng người học (Beginner, Intermediate, Expert, AllLevels)
    - Output:
      - Tên khóa học mà có điểm rating cao nhất và điểm rating đó.
      - Exception 1: Thông báo lỗi Nhãn đối tượng người học không được NULL
      - Exception 2: Thông báo lỗi Nhãn đối tượng người học phải thuộc 1 trong 4 nhãn ở trên
      - Exception 3: Thông báo lỗi Không có khóa học nào với nhãn đối tượng trên
      - Exception 4: Thông báo lỗi Không có bất kì review nào ở tất cả các khóa học với nhãn đối tượng trên
    - Code:

```
CREATE OR REPLACE FUNCTION get_highest_rating_course(audience_label
"AudienceLabel") RETURNS TABLE("name" VARCHAR, "averageRating"
NUMERIC(10, 2)) AS $$
    DECLARE
        cou RECORD;
        max_course_name VARCHAR;
        max_average_rating NUMERIC(10,2) = 0;
    BEGIN
        -- check if audience label is NULL and raise exception
        IF audience_label IS NULL THEN
            RAISE EXCEPTION 'Audience label is null!';
        END IF;

        -- get all courses by audience label, if not found raise
exception
        IF NOT EXISTS (SELECT * FROM "Course" WHERE "audienceLabel" =
audience_label) THEN
            RAISE EXCEPTION 'Course with audience label % not found!',
audience_label;
        END IF;

        -- in StudentReviewCourse, group by courseId and calculate
```

```
average rating
        -- add a new column "averageRating" to result
        FOR cou IN
            SELECT "Course".id , "Course".name,
AVG("StudentReviewCourse"."rating")::NUMERIC(10,2) AS "averageRating"
            FROM "Course"
            JOIN "StudentReviewCourse" ON "Course".id =
"StudentReviewCourse"."courseId"
            WHERE "Course"."audienceLabel" = audience_label
            GROUP BY "Course".id
        LOOP
            IF cou."averageRating" > max_average_rating THEN
                max_course_name := cou.name;
                max_average_rating := cou."averageRating";
            END IF;
        END LOOP;

        -- return the course with the max rating or raise an exception
if not found
        IF max_average_rating = 0 THEN
            RAISE EXCEPTION 'Course with audience label % not found
review!', audience_label;
        END IF;
        RETURN QUERY SELECT max_course_name, max_average_rating;
    END;
    $$ LANGUAGE PLPGSQL;
```

- **Hàm 2:** Tính doanh thu của các khóa học có phí đã bán được trong một năm nhất định.
  - Input:là năm để tính doanh thu, phải là số nguyên dương
  - Output:
    - Doanh thu của từng khóa học trong một năm
    - Exception1 : Thông báo lỗi Invalid input khi input NULL hoặc không phải là số.
  - Code:

```
CREATE OR REPLACE FUNCTION get_revenue(YEAR INT) RETURNS TABLE(name
VARCHAR, revenue INT) AS $$
    DECLARE
        pai_cou RECORD;
        total_revenue INT;
    BEGIN
        IF year IS NULL THEN
        RAISE EXCEPTION 'Year is null!';
        END IF;

        IF year < 0 THEN
        RAISE EXCEPTION 'Year is invalid!';
        END IF;

        FOR pai_cou IN
        SELECT "Course".id, "Course".name,
```

```
"PaidCourse"."priceDiscounted"
        FROM "PaidCourse"
        LEFT JOIN "Course" ON "Course".id = "PaidCourse"."courseId"
        LOOP
        SELECT COALESCE(SUM("PaidCourse"."priceDiscounted"),0) INTO
total_revenue
        FROM "PaidCourse"
        JOIN "PaidCourseOrder" ON "PaidCourseOrder"."paidCourseId" =
"PaidCourse"."courseId"
        JOIN "Order" ON "Order".id = "PaidCourseOrder"."orderId"
        WHERE "PaidCourse"."courseId" = pai_cou.id AND EXTRACT(YEAR
FROM "Order"."createdAt") = year;

        name := pai_cou.name;
        revenue := total_revenue;
        RETURN NEXT;
        END LOOP;
    END;
    $$ LANGUAGE PLPGSQL;
```

Hiện thực ứng dụng

Phụ lục

ERD

Bảng phân công nhiệm vụ