Database Design and Implementation

College Enrollment System Case-Study

**Table of Content**  
  
1. [Introduction](#_Introduction)

2. [Mission and Objectives](#_Mission)

2.1 [Mission](#_Mission)

2.2 [Objectives](#_Objectives)

3. [Database Design](#_Database_Design)

3.1 [Entities](#_Entities)  
3.2 [Entity Relationship diagram (ER diagram)](#_Entity-Relationship_Diagram_(ERD))  
3.3 [Relationships](#_Relationships)  
3.4 [Data Dictionary](#_Data_Dictionary)  
3.5 [Table Creation](#_Queries_to_create)

4. [Advanced SQL Queries for College Enrollment System.](#_Advanced_SQL_Queries)

5. [Views](#_Views)

4.1 [Student Enrollment summary view](#_STUDENT_ENROLLMENT_SUMMARY)  
 4.2 [Courses in program view](#_COURSES_IN_PROGRAM)  
 4.3 [Student performance report](#_STUDENT_PERFORMANCE_REPORT)

6. [Conclusion](#_Conclusion)

# Introduction

The **College Enrollment System** is designed to simplify the academic management process by efficiently handling student enrollment, faculty assignments, course scheduling, and student performance in exams. This case study presents the database design and implementation, covering table structures, relationships, complex queries, and views.  
  
Mission and Objectives

## Mission

To simplify enrollment and academic management while ensuring accurate data handling.

## Objectives

* Maintain accurate records of student details, course schedules, and faculty assignments.
* Enable payments and fee management.
* Monitor performance and generate detailed reports.

# **Database Design**

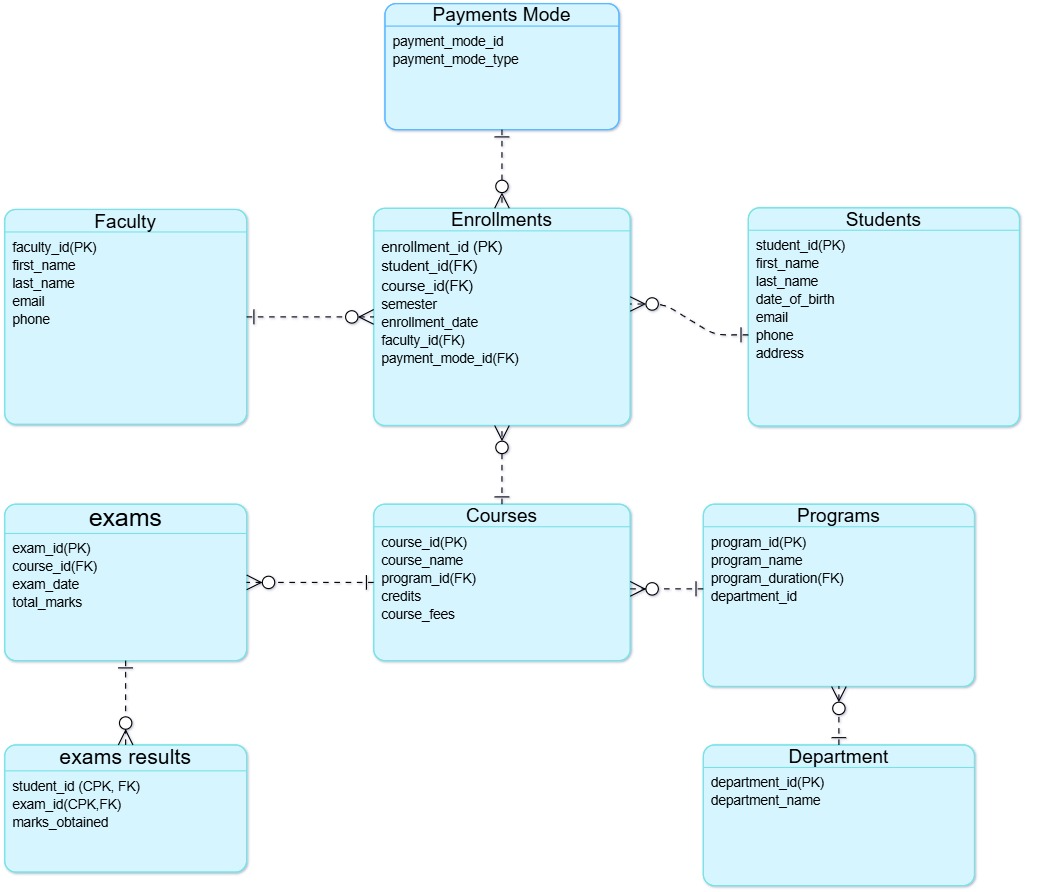
## Entities

Entities used in this database are as follows:

* **Students** – Stores student details such as name, contact information, and address.
* **Payment mode** – Records distinct payment modes available for payment.
* **Department** - Stores the details about the various academic departments in the college.
* **Programs** – Defines the academic programs offered by the college (e.g., BSc, BTech).
* **Faculty** – Stores the details of the faculty members who teach courses in the college.
* **Courses** – Stores information about courses within different programs.
* **Exams**– Stores information about the exams associated with each course.
* **Exam Results** - Stores the results of students' exams.
* **Enrollments** – Stores information about the courses each student is enrolled in.

## **Entity-Relationship Diagram (ERD)**

The **ER Diagram** represents the key entities of the system and their relationships.



## Relationships

### **1. Students and Enrollments (1:M)**

* **Relationship:** One student can have multiple enrollments, but each enrollment belongs to only one student.
* **Reason:** A student may enroll in multiple courses across different semesters.

### **2. Courses and Enrollments (1:M)**

* **Relationship:** One course can have multiple enrollments, but each enrollment is for only one course.
* **Reason:** Many students can enroll in the same course.

### **3. Faculty and Enrollments (1:M)**

* **Relationship:** One faculty member can oversee multiple enrollments, but each enrollment is handled by one faculty member.
* **Reason:** Faculty members may be assigned to multiple students in various courses.

### **4. Payments Mode and Enrollments (1:M)**

* **Relationship:** One payment mode can be used in multiple enrollments, but each enrollment has only one payment mode.
* **Reason:** Students may pay through different payment methods such as credit card, cash, or online transactions.

### **5. Programs and Courses (1:M)**

* **Relationship:** One program can include multiple courses, but each course belongs to only one program.
* **Reason:** Programs (like Computer Science or Business) consist of multiple courses.

### **6. Students and Exam Results (1:M)**

* **Relationship:** One student can have multiple exam results, but each exam result belongs to only one student.
* **Reason:** A student appears for multiple exams in different courses.

### **7. Exams and Exam Results (1:M)**

* **Relationship:** One exam can generate multiple results, but each result is linked to a single exam.
* **Reason:** Many students attempt the same exam, and each has a separate result.

### **8. Courses and Exams (1:M)**

* **Relationship:** One course can have multiple exams, but each exam belongs to only one course.
* **Reason:** Each course can have multiple exams throughout the semester.

### **9. Department and Programs (1:M)**

* **Relationship:** One department can have multiple programs, but each program belongs to only one department.
* **Reason:** A department (such as Engineering or Science) can offer multiple academic programs.

## Data Dictionary

The **relational model** of the College Enrollment System consists of multiple tables, each designed to store structured information and maintain referential integrity. Below is a description of the **key tables** and their relationships:

#### **Students Table**

Stores student details, including personal information.

This table holds all essential student information, ensuring that every student has a unique Student\_ID.

* **Primary Key:** Student\_ID
* **Not null Constraint :** first\_name, last\_name, dob, email, phone, address
* **Unique Constraint :** email

A close-up of a list

Description automatically generated

#### **2. Payment mode Table**

Stores information about students who are enrolled in the college.

* **Primary Key:** Payment\_ID
* **Not null Constraint :** payment\_mode\_type
* **Unique Constraint :** payment\_mode\_type

A close-up of a document

Description automatically generated

#### **3. Department Table**

Stores the details about the various academic departments in the college.

* **Primary Key:** department\_id
* **Not null Constraint :** department\_name
* **Unique Constraint :** department\_name

A close-up of a document

Description automatically generated

#### **4.** Programs **Table**

Stores information about the academic programs (e.g., BSc, BTech) offered by the college.

* **Primary Key:** program\_id
* **Foreign Key:** department\_id references *Department Table* department\_id
* **Not null Constraint :** program\_name
* **Unique Constraint :** program\_name

A close-up of a document

Description automatically generated

#### **5. Faculty Table**

Stores information about faculty members who teach courses.

* **Primary Key:** Faculty\_ID
* **Not null Constraint :** first\_name, last\_name, email, phone
* **Unique Constraint :** email, phone

A close-up of a list of words

Description automatically generated

#### **6. Courses Table**

Stores details about the courses offered by each program in the college.

Each course belongs to a program (e.g., BSc Computer Science), ensuring a structured academic hierarchy.

* **Primary Key:** Course\_ID
* **Foreign Key:** Program\_ID references *Programs table* Programs\_id
* **Not null Constraint :** course\_name
* **Unique Constraint :** course\_name
* **Default :** *True* for credit column

A close-up of a text

Description automatically generated

#### **7.** Exam Table

Stores information about the exams associated with each course.

* **Primary Key:** exam\_id
* **Foreign Key:** course\_id references *Course Table* course\_id
* **Not null Constraint:** course\_id

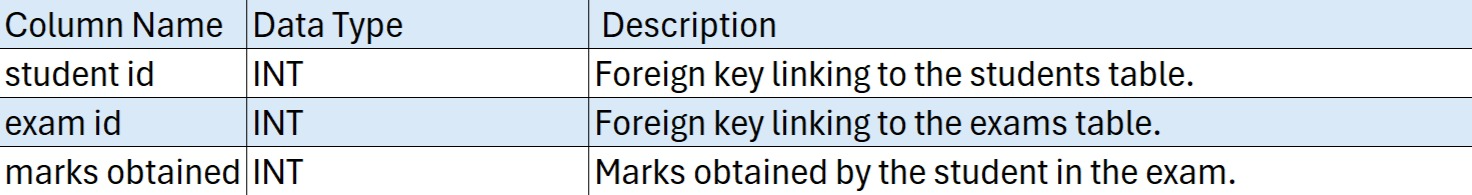
A close-up of a text

Description automatically generated

#### **8.** ExamResult Table

Stores the results of students' exams.

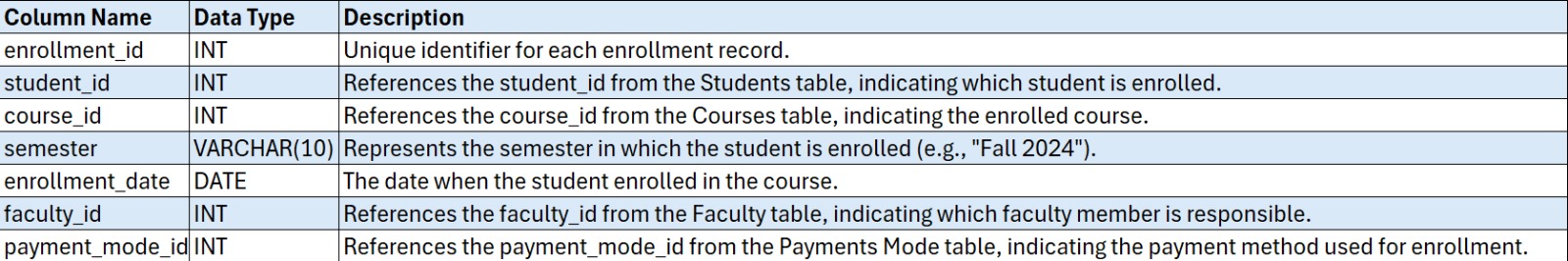
* **Composite Primary Key:** student\_id and exam\_id
* **Foreign Key:** student\_id references *Student Table* student\_id.   
   exam\_id references *Student Table* exam\_id



#### **9.** Enrollment Table (transaction table)

Stores information about the courses each student is enrolled in.

* **Composite Primary Key:** student\_id and exam\_id
* **Foreign Key:** student\_id references *Student Table* student\_id.   
  course\_id references *Course Table* course\_id  
  faculty\_id references *Faculty Table* faculty\_id  
  payment\_mode\_id references *Payment\_mode Table* payment\_mode\_id
* **Not null Constraint :** student\_id, course\_id
* **Default :** *curdate() function* for enrollment\_date column



## Queries to create Tables

Queries to create tables are as follows :

-- students table :

create table students (

student\_id int(10) primary key,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

dob date not null,

email varchar(120) not null unique,

phone varchar(17) not null,

address varchar(250) not null

);

-- payment mode table :

create table payment\_mode (

payment\_mode\_id int(2) primary key auto\_increment,

payment\_mode\_type varchar(30) unique not null

);

-- department table :

create table department (

department\_id int(2) primary key auto\_increment,

department\_name varchar(50) not null unique);

-- programs table :

create table programs (

program\_id int(4) primary key,

program\_name varchar(150) not null unique,

program\_duration\_year int(1) ,

department\_id int(2) ,

foreign key (department\_id) REFERENCES department(department\_id)

);

-- faculty table :

create table faculty(

faculty\_id int(10) primary key auto\_increment,

first\_name varchar(50) not null ,

last\_name varchar(50) not null,

email varchar(100) not null unique,

phone varchar(17) not null unique );

-- courses table :

create table courses (

course\_id int(10) primary key ,

course\_name varchar(150) not null unique,

program\_id int(3) ,

credit boolean default(true),

course\_fee int(5) ,

foreign key (program\_id) references programs(program\_id)

);

-- exams table :

create table exams(

exam\_id int(15) primary key auto\_increment ,

course\_id int(10) not null,

exam\_date date ,

total\_marks int(3),

foreign key (course\_id) references courses (course\_id)

);

-- exam result table :

create table exam\_result(

student\_id int(10) ,

exam\_id int(15),

marks\_obtained int(3),

primary key(student\_id , exam\_id), -- composite primary key

foreign key (student\_id) references students(student\_id),

foreign key (exam\_id) references exams(exam\_id)

);

-- Enrollment table(transaction table) :

create table enrollments(

enrollment\_id int(15) primary key auto\_increment,

student\_id int(10) not null,

course\_id int(10) not null,

semester int(2) ,

enrollment\_date date default(CURDATE()),

faculty\_id int(10) ,

payment\_mode\_id int(2),

UNIQUE (student\_id, course\_id),

foreign key (student\_id) references students(student\_id),

foreign key (course\_id) references courses (course\_id),

foreign key (faculty\_id) references faculty(faculty\_id),

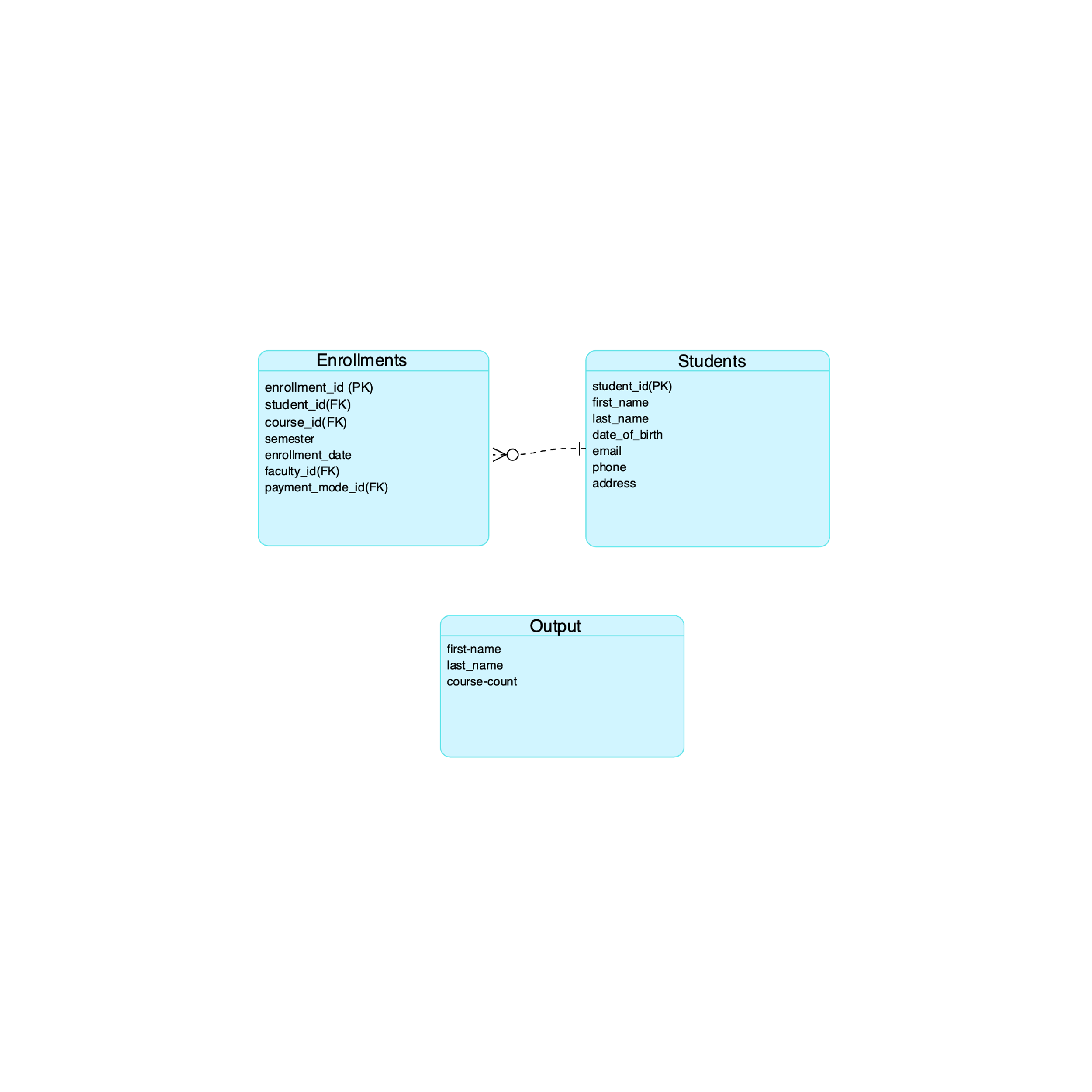
foreign key (payment\_mode\_id) references payment\_mode(payment\_mode\_id)

);

# Advanced SQL Queries for College Enrollment System

### Query to list students who have enrolled in multiple courses:

**Table used:** Enrollments, students



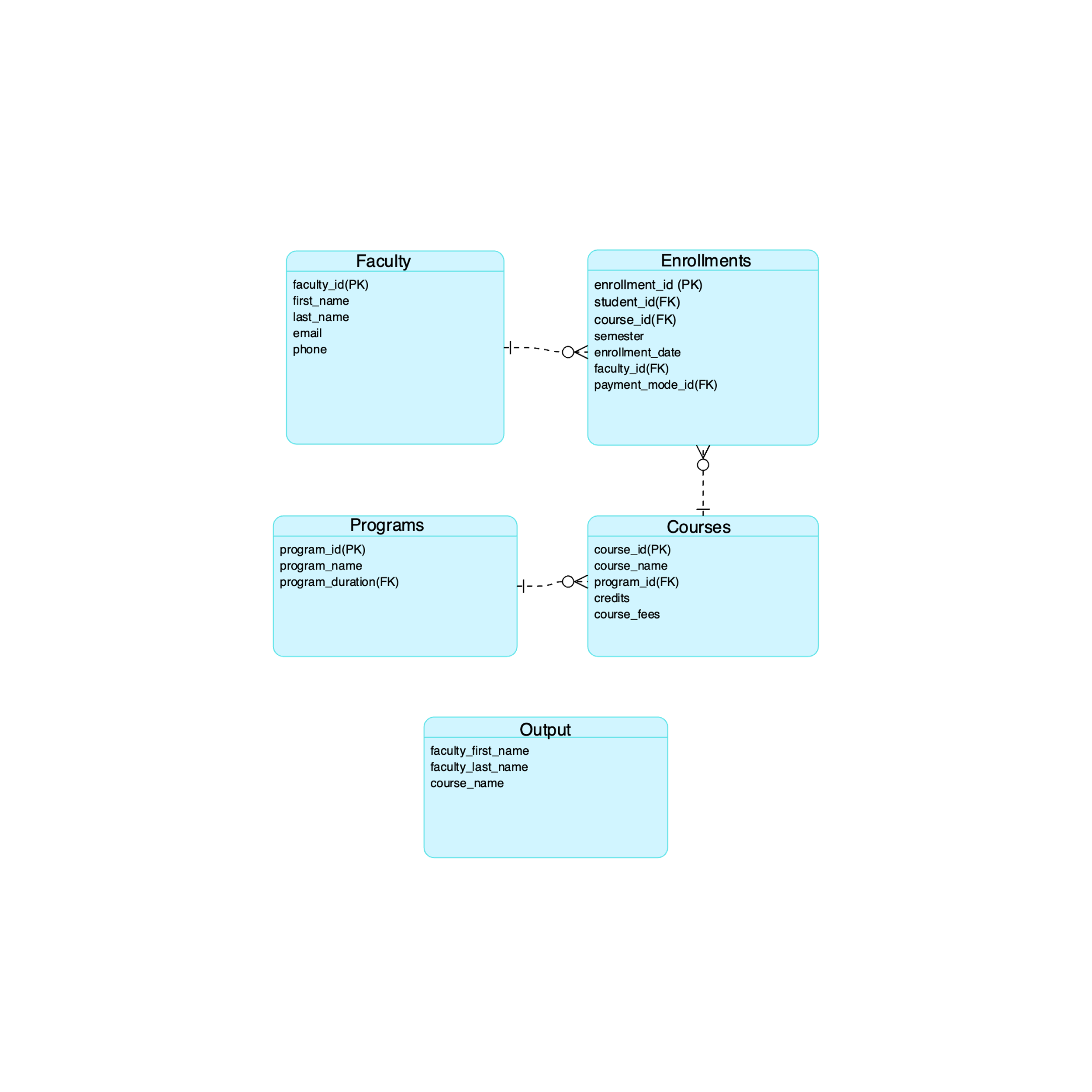
The following query retrieves students who are enrolled in **more than one course**, displaying their first name, last name, and total course count. It joins the **Students** and **Enrollments** tables, groups by student\_id, and filters results using the HAVING clause to show only students with multiple enrollments.

A screenshot of a computer

Description automatically generated

### Query to get faculty information for courses in a specific program

* + - **Table used:** Enrollments, students



This query retrieves the **first and last names of faculty members** along with the **course names** they teach in the **BSc Computer Science** program.

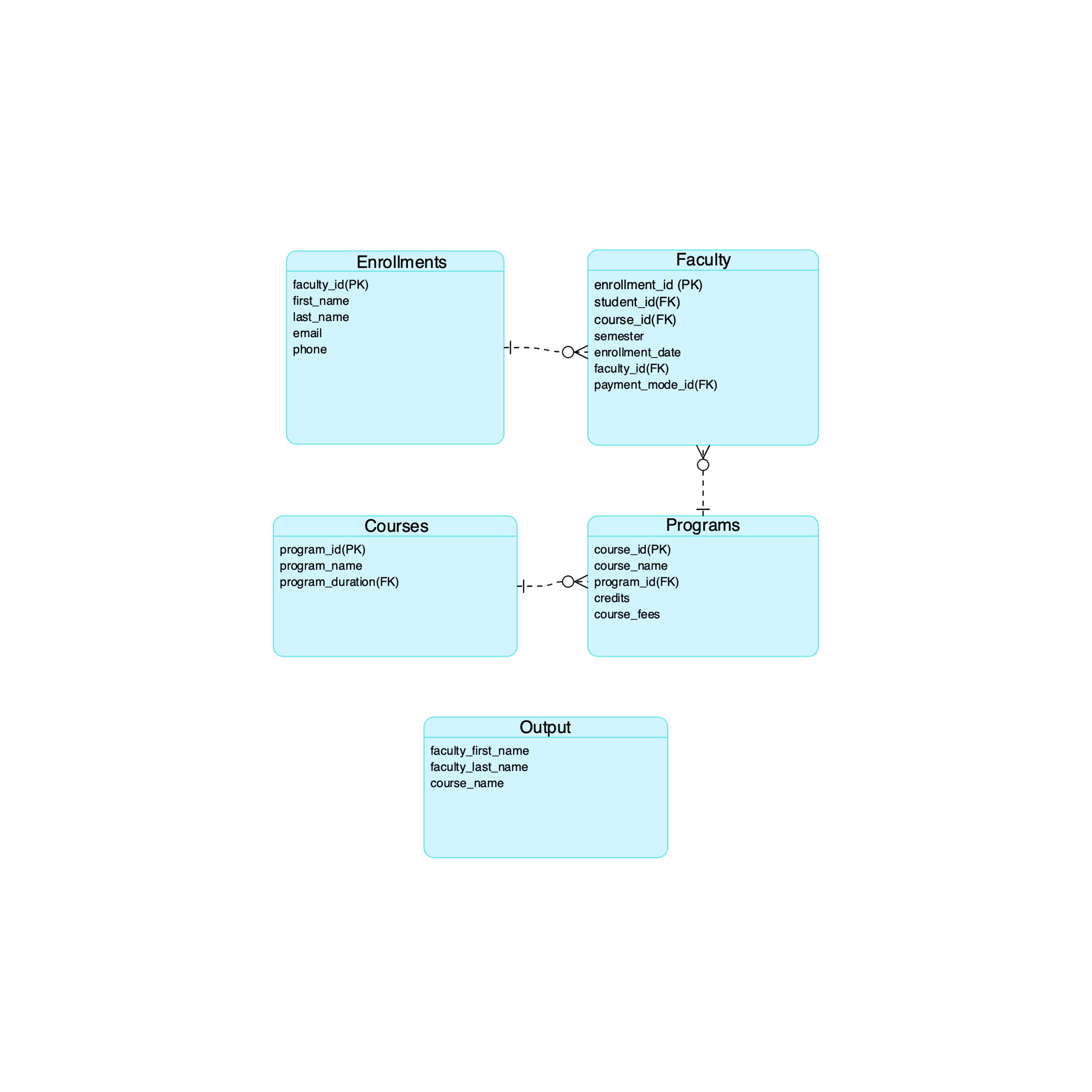
* It joins the **Faculty**, **Enrollments**, **Courses**, and **Programs** tables based on their relationships.
* The WHERE clause filters results to include only courses from the **BSc Computer Science** program.

A screenshot of a computer

Description automatically generated

### Query to find the total fees collected for each program

* + - **Table used:** Enrollments, students



This query calculates the **total fees collected** for each academic program.

* It joins the **Enrollments**, **Courses**, and **Programs** tables to link students' enrollments with their respective courses and programs.
* The SUM(c.course\_fee) function computes the total fees collected per program.
* The GROUP BY p.program\_name ensures the total fees are grouped by each program.

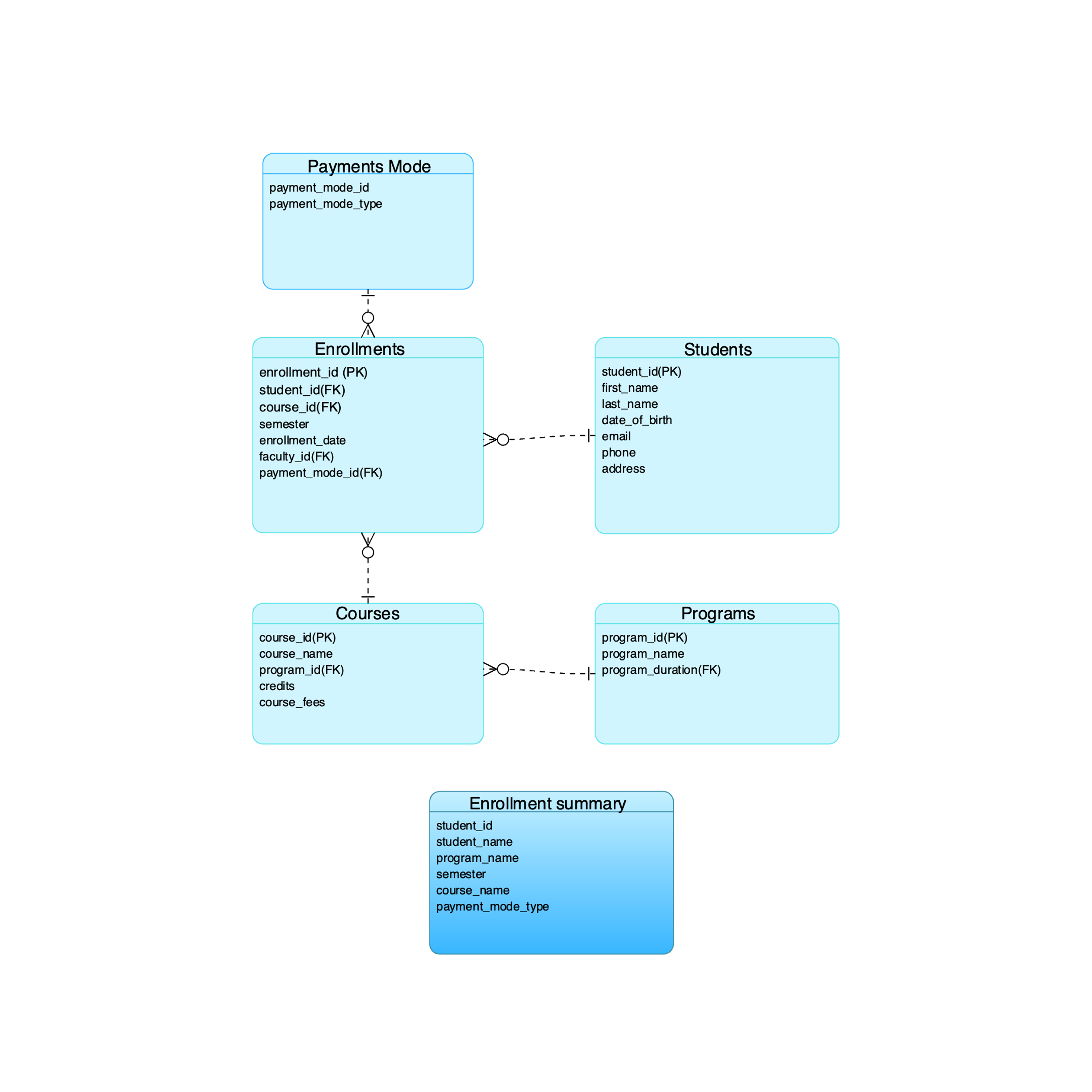
A screenshot of a computer

Description automatically generated

# Views

### STUDENT ENROLLMENT SUMMARY VIEW

Tables used :



**Scenario**: Track student enrollments by course, program, semester and payment mode.

**Purpose**: Monitor students enrollment and specify reporting

**Syntax :**

CREATE VIEW enrollment\_summary\_view AS

SELECT

s.student\_id,

CONCAT(s.first\_name, ' ', s.last\_name) AS student\_name,

p.program\_name,

e.semester,

c.course\_name,

pm.payment\_mode\_type

FROM enrollments e

JOIN students s ON e.student\_id = s.student\_id

JOIN courses c ON e.course\_id = c.course\_id

JOIN programs p ON c.program\_id = p.program\_id

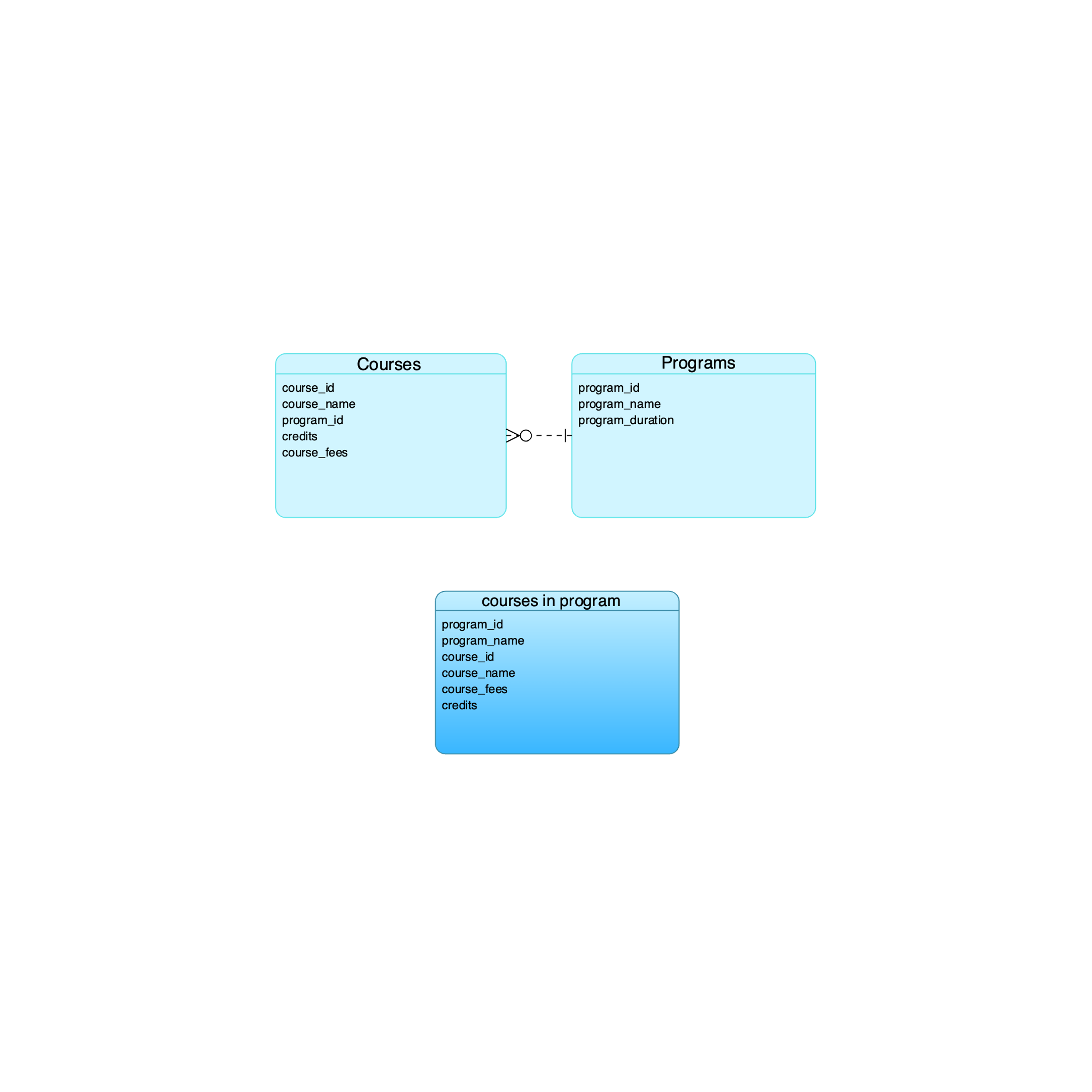
JOIN payments\_mode pm ON e.payment\_mode\_id = pm.payment\_mode\_id;

**Explanation:**

* **Joins multiple tables** to gather all necessary information.
* **Concatenates first name and last name** for student\_name.
* **Uses a view** to simplify future queries without needing complex joins

### COURSES IN PROGRAM VIEW

Tables used :



**Scenario**: Display all courses under a specific program.

**Purpose**: Help students & faculty to view available course in program.

**Syntax:**

CREATE VIEW courses\_in\_program AS

SELECT

p.program\_id,

p.program\_name,

c.course\_id,

c.course\_name,

c.course\_fees,

c.credits

FROM courses c

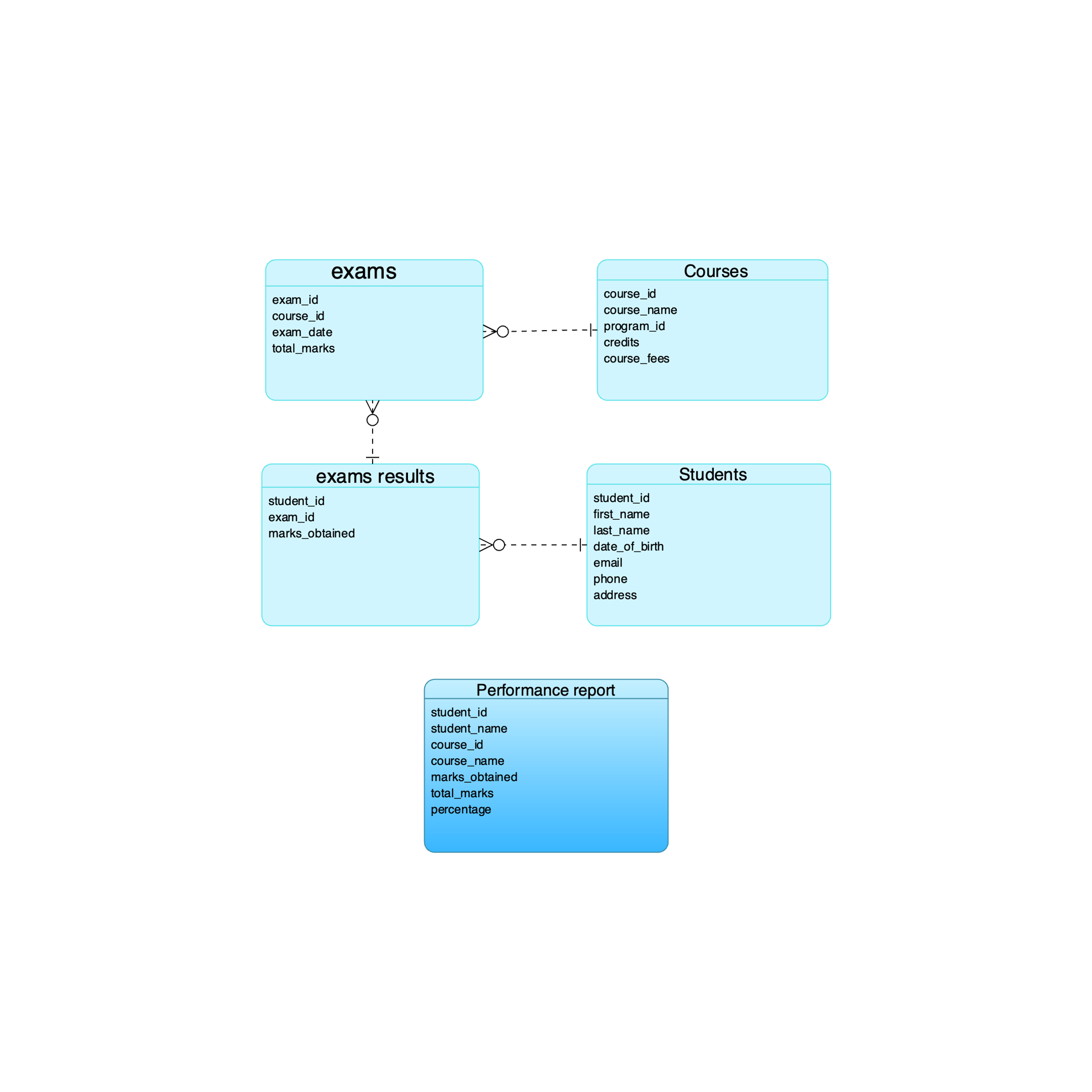
JOIN programs p ON c.program\_id = p.program\_id;

**Explanation:**

* **Joins courses and programs** using program\_id to fetch related data.
* **Includes relevant columns** like program name, course details, fees, and credits.
* **Simplifies queries** for retrieving program-wise course details.

### STUDENT PERFORMANCE REPORT

Tables used:



**Scenario**: Display students exam result along with course details

**Purpose:** Track student performance in exams Identify top & struggling students Provide insights for academic improvements

**Syntax:**

CREATE VIEW performance\_report AS

SELECT

s.student\_id,

CONCAT(s.first\_name, ' ', s.last\_name) AS student\_name,

c.course\_id,

c.course\_name,

er.marks\_obtained,

e.total\_marks,

ROUND((er.marks\_obtained / e.total\_marks) \* 100, 2) AS percentage

FROM students s

JOIN exams\_results er ON s.student\_id = er.student\_id

JOIN exams e ON er.exam\_id = e.exam\_id

JOIN courses c ON e.course\_id = c.course\_id;

**Explanation:**

 **Joins the students, exams\_results, exams, and courses tables** to collect all necessary details.

 **Uses CONCAT(s.first\_name, ' ', s.last\_name)** to display the student's full name.

 **Calculates the percentage** using ROUND((marks\_obtained / total\_marks) \* 100, 2).

 **Includes all key details** such as student ID, name, course details, obtained marks, total marks, and percentage.

# Conclusion

The College Enrollment System implements a well-structured database design comprising nine essential tables that effectively manage student enrollments, course offerings, faculty assignments, and examination records. Through carefully defined relationships and constraints, the system maintains data integrity while handling core academic operations. The implementation of advanced SQL queries and specialized views - including enrollment summaries, course listings, and performance reports - demonstrates the system's capability to provide meaningful insights for administrative decision-making. The database successfully achieves its objectives of maintaining accurate records, managing fees, and monitoring student performance, serving as an efficient solution for academic institution management.