College Enrollment System

A system which simplifies student admissions, course registration, and record management efficiently.

Author: Nikhil Mohan

Institution: Southern Alberta Institute of Technology

Date: Feb 03, 2025

Index

Tables

Introduction

In today's educational environment, institutions are constantly looking for ways to optimize their administrative tasks. Student enrollment is a crucial component of any academic institution, and a College Enrollment System aims to automate and simplify this process. By reducing paperwork, minimizing errors, and improving efficiency. It facilitates quick and accurate admissions, course registrations, and data management. Using technology, colleges can greatly enhance their operational effectiveness and offer a more seamless enrollment experience.

Mission

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

Objective

- Maintain accurate records of student details, course schedules, and faculty assignments.

- Enable payments and fee management.

- Monitor performance and generate detailed reports.

Design

Student table - Stores information about students who are enrolled in the college.

Payment Mode Table – Stores information about students who are enrolled in the college.

Department Table – Stores the details about the various academic departments in the college.

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

Faculty Table – Stores the details of the faculty members who teach courses in the college.

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

Exam Table – Stores information about the exams associated with each course.

Exam result Table - Stores the results of students' exams.

Enrolment Table - Stores information about the courses each student is enrolled in.

Relationship

1. Students → Enrollments (1:M)
    - One Student can have multiple enrollments.
    - The Enrollments table contains **student_id** as a foreign key (FK) to link each enrollment to a specific student.
    - Explanation: A student can enroll in multiple courses over different semesters, but each enrollment record belongs to only one student.

2. Enrollments → Courses (M:1)
    - Multiple Enrollments can be linked to one Course.
    - The Enrollments table contains **course_id** as an FK.
    - Explanation: Many students can enroll in the same course, but each enrollment refers to only one specific course.

3. Enrollments → Faculty (M:1)
    - Many enrollments can be associated with one faculty member.
    - The Enrollments table contains **faculty_id** as an FK.
    - Explanation: A faculty member oversees multiple enrollments for their course, but each enrollment record is managed by only one faculty member.

4. Enrollments → Payments Mode (M:1)
   - Many enrollments can have the same payment mode.
   - The Enrollments table contains **payment_mode_id** as an FK.
   - Explanation: Students can choose different payment methods, but each enrollment transaction is linked to only one payment mode.

5. Courses → Programs (M:1)
   - Many courses belong to one program.
   - The Courses table contains **program_id** as an FK.
   - Explanation: A program (e.g., Computer Science) consists of multiple courses (e.g., Database Systems, Algorithms), but each course belongs to only one program.

6. Programs → Department (M:1)
   - Many programs are associated with one department.
   - The Programs table contains **department_id** as an FK.
   - Explanation: A department (e.g., Engineering) has multiple programs (e.g., Computer Science, Electrical Engineering), but each program belongs to only one department.

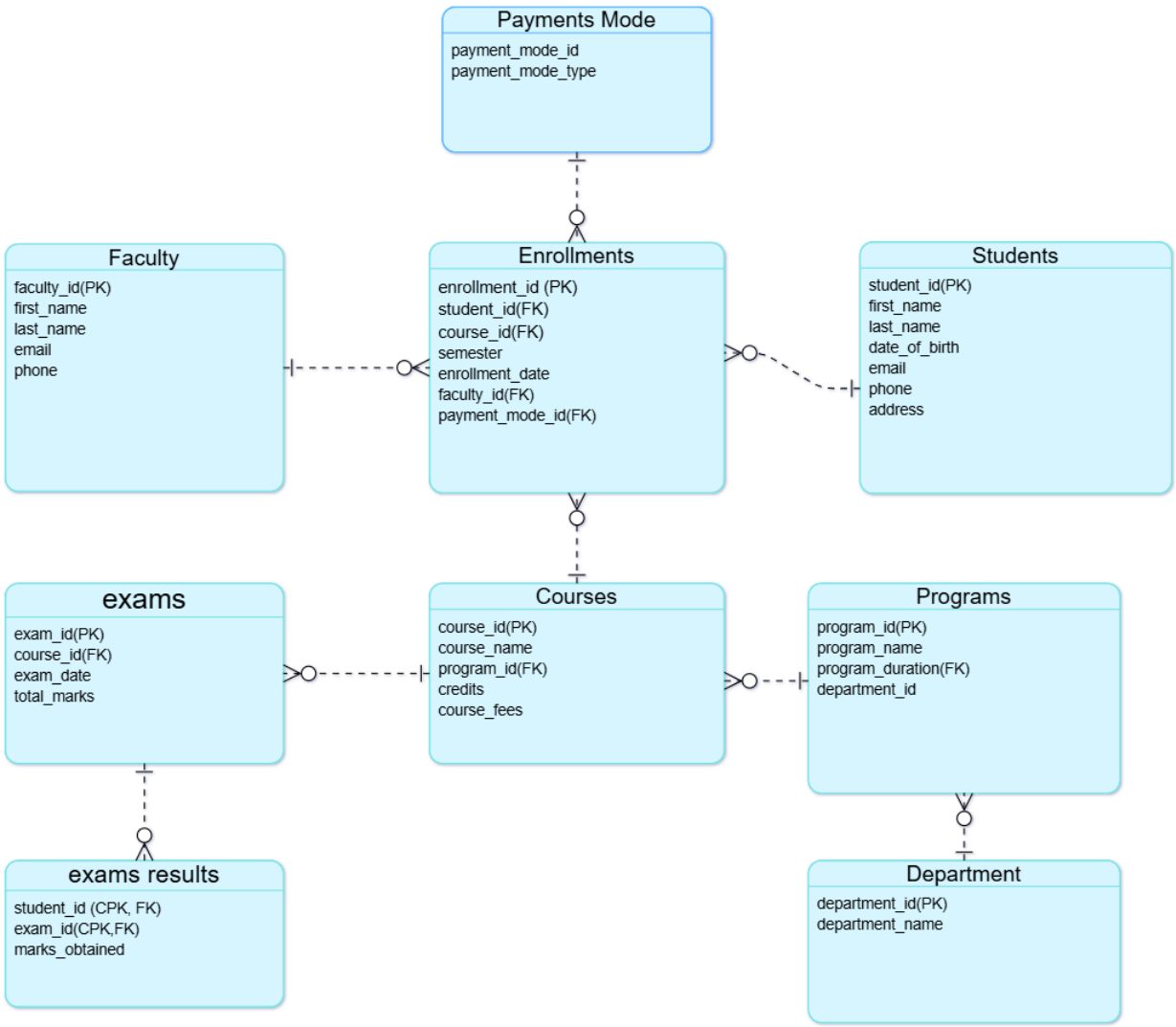
7. Courses → Exams (1:M)
   - One course can have multiple exams.
   - The Exams table contains **course_id** as an FK.
   - Explanation: A course (e.g., Mathematics) may have multiple exams (e.g., Midterm, Final), but each exam belongs to only one course.

8. Exams → Exam Results (1:M)
   - One exam can have multiple exam results.
   - The Exam Results table contains **exam_id** as an FK.
   - Explanation: Multiple students take an exam, and their scores are recorded in the exam results table. Each result belongs to only one exam.

9. Students → Exam Results (1:M)
   - One student can have multiple exam results.
   - The Exam Results table contains **student_id** as an FK.
   - Explanation: A student appears for multiple exams, and their results are recorded separately.

Entity Relation Diagram:

In the **College Management System**, the ER diagram visually represents how different entities, such as **Students, Faculty, Courses, Exams, and Departments**, interact with one another. This diagram defines relationships like **one-to-many (1:M)** and **many-to-one (M:1)** among various entities to ensure efficient data organization.



Summary of Relationship

Entity A	Entity B	Type	Explanation
Students	**Enrollments**	**1:M**	One student can enroll in multiple courses.
Enrollments	**Courses**	**M:1**	Many enrollments belong to a single course.
Enrollments	**Faculty**	**M:1**	Many enrollments are managed by one faculty.
Enrollments	**Payments Mode**	**M:1**	Many enrollments use the same payment mode.
Courses	**Programs**	**M:1**	Many courses belong to one program.
Programs	**Department**	**M:1**	Many programs belong to one department.
Courses	**Exams**	**1:M**	One course can have multiple exams.
Exams	**Exam Results**	**1:M**	One exam has multiple exam results.
Students	**Exam Results**	**1:M**	One student has multiple exam results.

Database Design

1. Student Table

Stores information about students who are enrolled in the college.

Column Name	Data Type	Description
student_id	INT	Unique identifier for each student. It is the primary key.
first_name	VARCHAR	Student's first name.
last_name	VARCHAR	Student's last name.
dob	DATE	Date of birth of the student.
email	VARCHAR	Student's email address.
phone	VARCHAR	Student's phone number.
address	TEXT	Student's residential address.

QUERY TO CREATE STUDENT 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
);
```

2. Payment Mode Table

It stores information about the various payment methods available for students to use when paying for their courses, fees

Column Name	Data Type	Description
payment_mode_id	INT	Unique identifier for each payment mode. It is the primary key.
payment_mode_type	VARCHAR	Type of payment (e.g., Credit Card, PayPal, Bank Transfer).

QUERY TO CREATE PAYMENT MODE TABLE –

```
create table payment_mode (
payment_mode_id int(2) primary key auto_increment,
payment_mode_type varchar(30) unique not null
);
```

3. Department Table

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

Column Name	Data Type	Description
department_id	INT	Unique identifier for each department. It is the primary key.
department_name	VARCHAR	Name of the department (e.g., Computer Science, Mathematics).

QUERY TO CREATE DEPARTMENT TABLE –

```
create table department (
department_id int(2) primary key auto_increment,
department_name varchar(50) not null unique
);
```

4. Program Table

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

Column Name	Data Type	Description
program_id	INT	Unique identifier for each program. It is the primary key.
program_name	VARCHAR	Name of the program (e.g., BSc Computer Science, BTech Electrical Engineering).
program_duration_year	INT	Duration of the program in years (e.g., 3 years, 4 years).
department_id	INT	Foreign key linking to the `department` table.

QUERY TO CREATE PROGRAM 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)
);
```

5. Faculty Table

Stores the details of the faculty members who teach courses in the college.

Column Name	Data Type	Description
faculty_id	INT	Unique identifier for each faculty member. It is the primary key.
first_name	VARCHAR	Faculty member's first name.
last_name	VARCHAR	Faculty member's last name.
email	VARCHAR	Faculty member's email address.
phone	VARCHAR	Faculty member's phone number.

QUERY TO CREATE 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 );
```

6. Course Table

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

Column Name	Data Type	Description
course_id	INT	Unique identifier for each course. It is the primary key.
course_name	VARCHAR	Name of the course (e.g., Data Structures, Calculus).
program_id	INT	Foreign key linking to the `programs` table.
credit	BOOLEAN	Indicates whether the course carries credit (TRUE/FALSE).
course_fee	DECIMAL	Fee for the course.

QUERY TO CREATE COURSE 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)
);
```

7. Exam Table

Stores information about the exams associated with each course.

Column Name	Data Type	Description
exam_id	INT	Unique identifier for each exam. It is the primary key.
course_id	INT	Foreign key linking to the `courses` table.
exam_date	DATE	Date of the exam.
total_marks	INT	Total marks for the exam.

QUERY TO CREATE COURSE MODE 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) );
```

8. Exam Result Table

Stores the results of students' exams.

Column Name	Data Type	Description
student id	INT	Foreign key linking to the students table.
exam id	INT	Foreign key linking to the exams table.
marks obtained	INT	Marks obtained by the student in the exam.

QUERY TO CREATE COURSE MODE 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) );
```

9. Enrollment Table

Stores information about the courses each student is enrolled in.

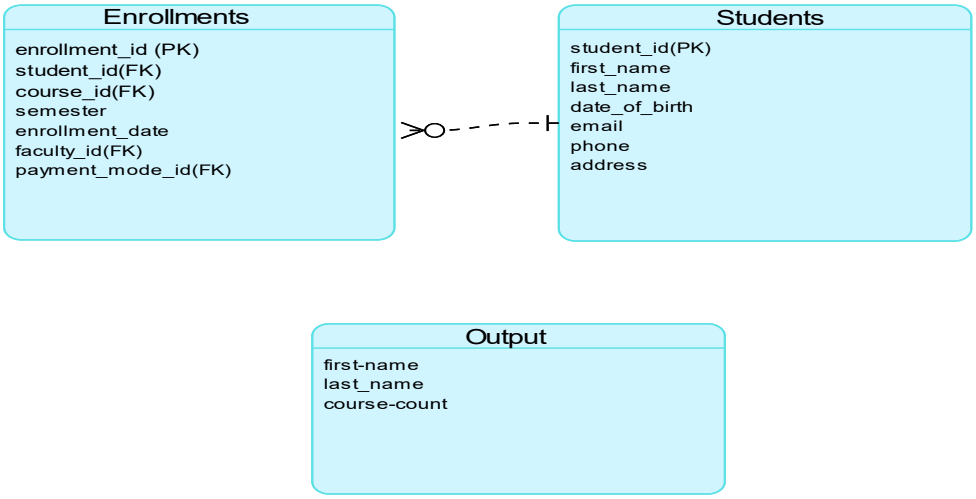
Column Name	Data Type	Description
enrollment_id	INT	Unique identifier for each enrollment record.
student_id	INT	References the student_id from the Students table, indicating which student is enrolled.
course_id	INT	References the course_id from the Courses table, indicating the enrolled course.
semester	VARCHAR(10)	Represents the semester in which the student is enrolled (e.g., "Fall 2024").
enrollment_date	DATE	The date when the student enrolled in the course.
faculty_id	INT	References the faculty_id from the Faculty table, indicating which faculty member is responsible.
payment_mode_id	INT	References the payment_mode_id from the Payments Mode table, indicating the payment method used for enrollment.

QUERY TO CREATE ENROLLMENT MODE 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

1. Query to list students who have enrolled in multiple courses:



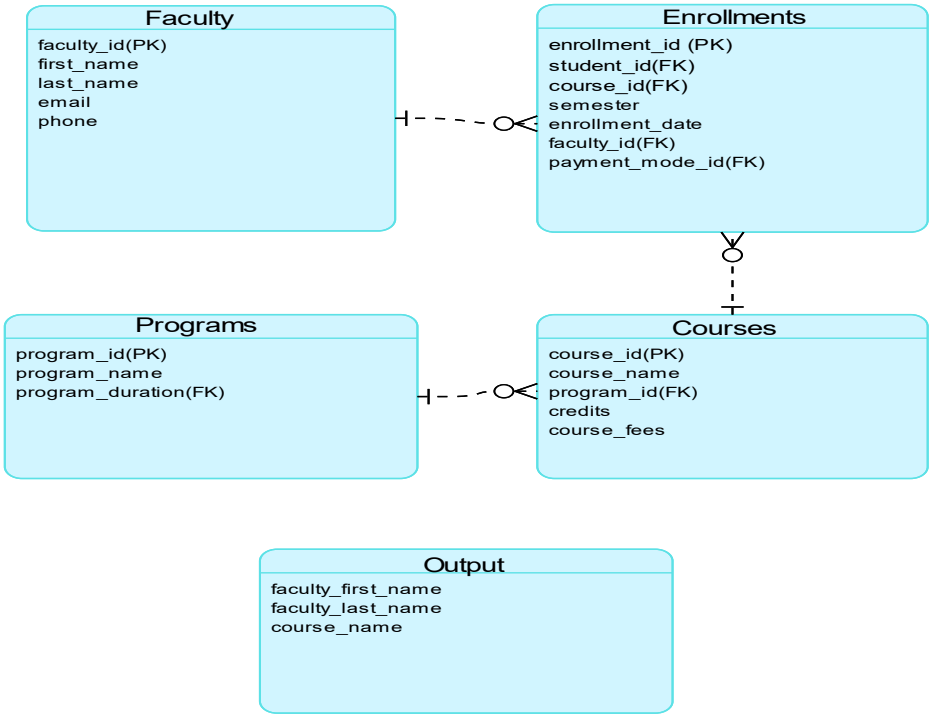
This query selects students enrolled in more than one course, showing their first name, last name, and the total number of courses they are registered for. It combines data from the Students and Enrollments tables, groups the results by student ID, and uses the HAVING clause to filter and display only those students with multiple course enrollments.

```
212    --Query to List Students Who Have Enrolled in Multiple Courses
213    SELECT s.first_name, s.last_name, COUNT(e.course_id) AS course_count
214    FROM enrollments e
215    JOIN student s ON e.student_id = s.student_id
216    GROUP BY s.student_id
217    HAVING COUNT(e.course_id) > 1;
218
219    --Query to Get Faculty Information for Courses in a Specific Program
```

Results Messages

	first_name	last_name	course_count
1	Nikhil	Choudhary	4
2	Mohan	Smith	3
3	Sharanpreet	Kaur	3
4	Jaspreet	Singh	4
5	Akshar	Sharma	2
6	Vishal	Wilson	3
7	Harpreet	Kaur	3
8	Sashi	Sharma	2

Query to get faculty information for courses in a specific program



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

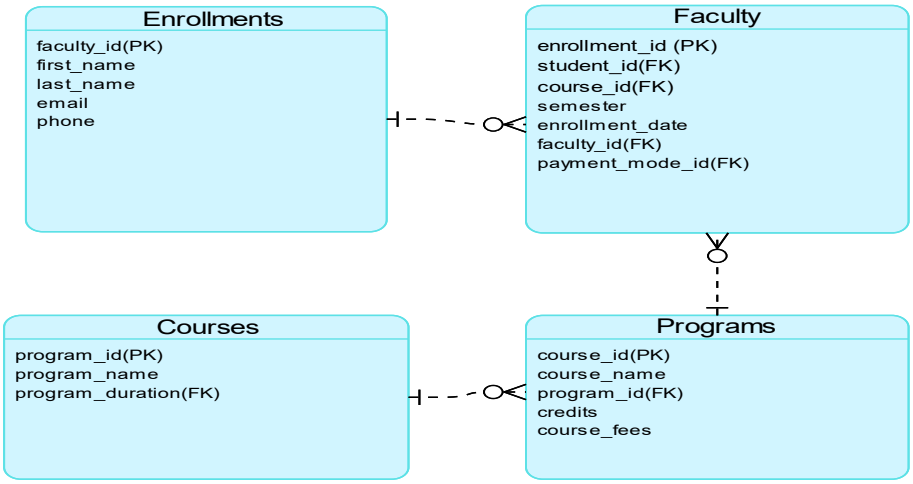
- It connects the Faculty, Enrollments, Courses, and Programs tables based on their relationships.
- The WHERE clause ensures that only courses from the BSc Computer Science program are included in the results.

```
219  --Query to Get Faculty Information for Courses in a Specific Program
220  SELECT DISTINCT f.first_name AS faculty_first_name, f.last_name AS faculty_last_name, c.course_name
221  FROM faculty f
222  JOIN enrollments en ON f.faculty_id = en.faculty_id
223  JOIN courses c ON en.course_id = c.course_id
224  JOIN programs p ON c.program_id = p.program_id
225  WHERE p.program_name = 'BSc Computer Science';
226
227  -- Query to Find the Total Fees Collected for Each Program
```

Results | Messages

	faculty_first_name	faculty_last_name	course_name
1	Junaid	Qazi	Data Structures and Algorithms
2	Priyanka	Miller	Introduction to Python

Query to find the total fees collected for each program



This query determines the total fees collected for each academic program.

- It links the Enrollments, Courses, and Programs tables to associate student enrollments with their corresponding courses and programs.
- The SUM(c.course_fee) function calculates the total fees collected for each program.
- The GROUP BY p.program_name groups the total fees by individual programs.

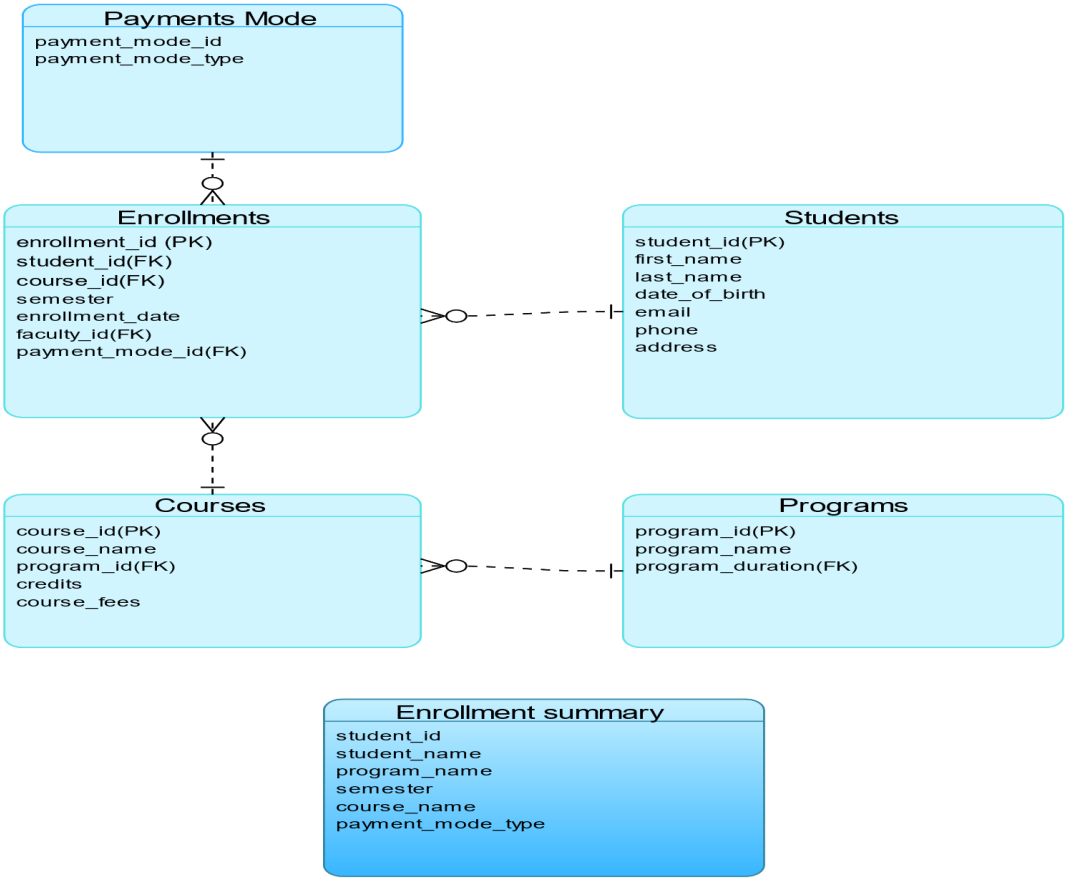
```
226
227    -- Query to Find the Total Fees Collected for Each Program
228    SELECT p.program_name, SUM(c.course_fee) AS total_fees_collected
229    FROM enrollments e
230    JOIN courses c ON e.course_id = c.course_id
231    JOIN programs p ON c.program_id = p.program_id
232    GROUP BY p.program_name;
```

Results Messages

	program_name	total_fees_collected
1	BSc Biology	15600
2	BSc Chemistry	19200
3	BSc Computer Science	15100
4	BSc Mathematics	9200
5	BSc Physics	22000
6	BTech Artificial Intelligence	15300
7	BTech Astrophysics	15900
8	BTech BioMechanical Engineering	10000

Views

STUDENT ENROLLMENT SUMMARY VIEW



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;

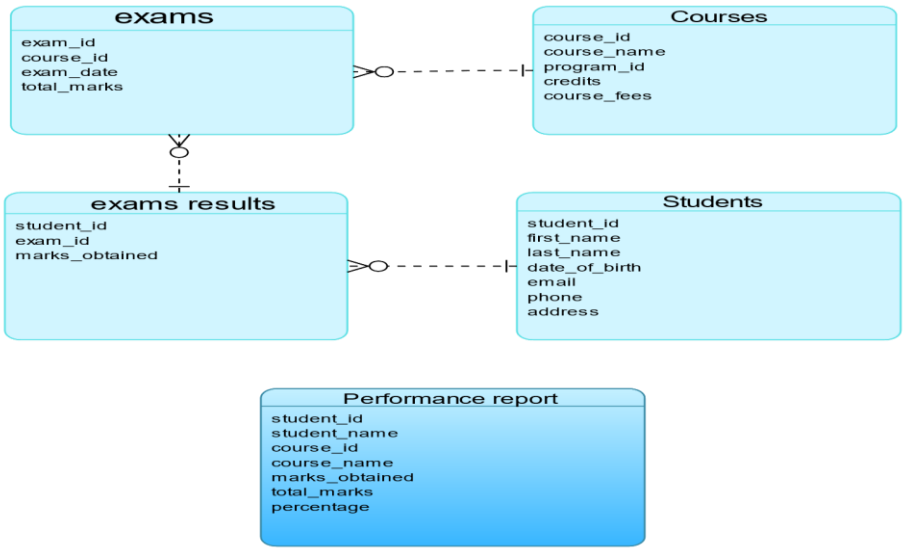
Explanation:

- **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

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 simplifies student admissions, course registration, and academic record management, making administrative processes more efficient. By combining these key functions into one platform, it enhances accessibility, minimizes manual work, and ensures accurate student data handling. Additionally, it improves data accuracy and security, ensuring that student records are well-organized and easily accessible when needed.