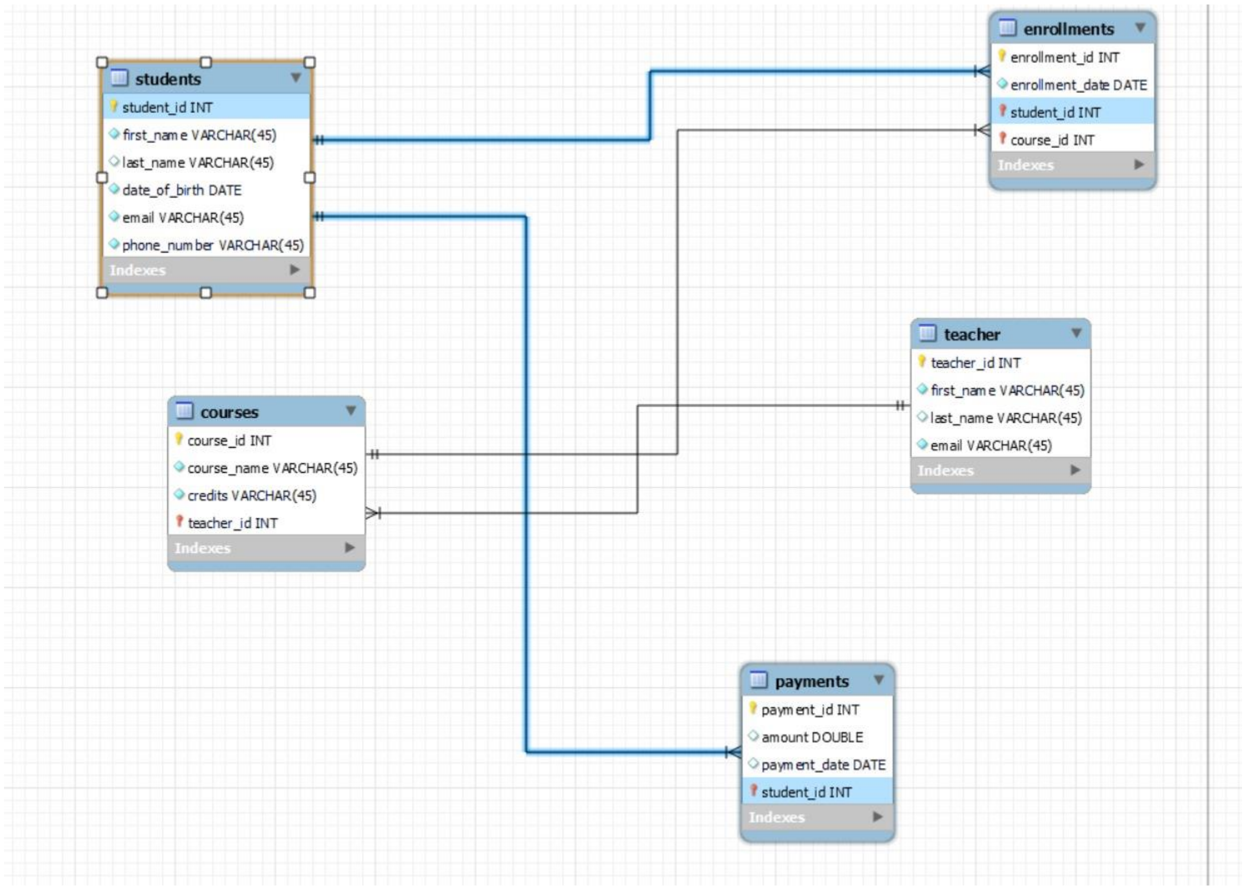
STUDENT MANAGEMENT SYSTEM

ER DIAGRAM



use Stud;

show databases;

show tables;

insert into students(first_name,last_name,date_of_birth,email,phone_number) values

('Harry','Potter','2000-01-06','harry@gmail.com',9876543455),

('Ronald','Weasley','2001-05-02','ronald@gmail.com',9876098455),

('Voldermot',null,'2001-02-04','nams@gmail.com',8866543455),

('Genny',null,'2001-02-03','genn@gmail.com',8876543468),

('Ronaldo','Trumph','2002-04-10','trumphh@gmail.com',8886545455),

```sql
('Cho','Chang','2000-01-01','cho@gmail.com',9876540010),

('Crabe',null,'2000-05-12','crabe@gmail.com',9878878455),

('Vincent','Creevy','2002-02-04','creev@gmail.com',8566643455),

('Dirk',null,'2000-02-01','dirk@gmail.com',8876540468),

('Barty','Crouch','2001-04-04','barty@gmail.com',8880045455);

select * from students;

insert into teacher(first_name,last_name,email)
values('Seema','Raj','see3@gmail.com'),('Jack',null,'jack@gmail.com'),

('Rose','Kevin','rose@gmail.com'),('Surya','Kumar','surr5@gmail.com'),('Rajini',null,'rajni3@gmail.com'),

('Raj','Kumar','raj@gmail.com'),('Ram','Gopal','ram@gmail.com'),('Latha','Gopal','latha@gmail.com'),('Gayu
','Chan','gayu@gmail.com'),

('Manoj','Kumar','mano@gmail.com');

select * from teacher;

insert into courses(course_name,credits,teacher_id)values

('Embbeded',10,1),('Java',8,2),('Python',9,3),('C',7,4),('C#',10,5),('C++',8,6),('Digital EC',9,5),

('Power BI',8,2),('ML',10,3),('Data Structures',9,10);

select * from courses;

insert into payments(amount,payment_date,student_id)values (2500,2023-01-01,1),

(3000,2023-02-01,2),(5000,2023-04-02,3),(2500,2023-03-06,4),(3000,2023-09-09,5),(3000,2023-09-09,5),

(3000,2023-05-01,6),(2000,2023-10-11,5),(15000,2023-12-09,3),(4500,2023-07-08,9);

select * from payments;

insert into enrollments(enrollment_date,student_id,course_id)values ('2023-01-20',1,2),('2023-01-29',3,2),

('2023-02-09',1,3),('2023-03-08',2,6),('2023-03-05',3,9),('2023-04-10',1,8),('2023-05-10',4,6),('2023-06-
06',8,3),('2023-08-13',10,9),

('2023-10-27',6,8);

select * from enrollments;

-- Tasks 2

-- 1. Write an SQL query to insert a new student into the "Students" table with the following details:

-- a. First Name: John  -- b. Last Name: Doe  -- c. Date of Birth: 1995-08-15  -- d. Email: john.doe@example.com
-- e. Phone Number: 1234567890

insert into students(first_name,last_name,date_of_birth,email,phone_number) values

('Viki','Joseph','2003-08-15','john.doe@example.com',8234567890);

-- 2. Write an SQL query to enroll a student in a course. Choose an existing student and course and
```

-- insert a record into the "Enrollments" table with the enrollment date.

-- Assuming you have existing students and courses, and you know the IDs

-- Assuming you have the student_id, course_id, and enrollment_date

insert into enrollments (student_id, course_id, enrollment_date) VALUES (1,10,'2024-09-10');

-- 3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and -- modify their email address.

update teacher set email='kumar@gmail.com' where teacher_id=10;

select * from teacher; -- To check the updated teacher table

-- 4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select

-- an enrollment record based on the student and course.

delete from enrollments WHERE student_id = 1 AND course_id = 8;

-- 5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.

update courses SET teacher_id = 1 WHERE course_id = 4;

-- 6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table.

-- Be sure to maintain referential integrity.

delete from enrollments where student_id=1;

delete from students where student_id = 1;

-- 7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.

update payments set amount = '6000' where payment_id = 4;

-- Task 3.

-- 1. Write an SQL query to calculate the total payments made by a specific student.

select s.student_id,s.first_name,sum(amount) from students s join payments p on s.student_id=p.student_id where s.student_id=1;

-- 2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course.

select c.course_name,count(e.student_id) as count_of_students_enrollments from courses c join enrollments e on c.course_id=e.course_id group by

c.course_name;

-- 3.Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments

select s.student_id,s.first_name from students s left join enrollments e on

s.student_id=e.student_id where e.student_id=null;

```sql
-- 4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are
enrolled in. Use JOIN operations between the "Students" table and the

-- "Enrollments" and "Courses" tables.

select s.first_name,s.last_name,c.course_name from students s join enrollments e on s.student_id=e.student_id
join courses c on c.course_id=

e.course_id;

-- 5. Create a query to list the names of teachers and the courses they are assigned to.

select t.teacher_id,t.first_name,t.last_name,c.course_name from teacher t join courses c on
t.teacher_id=c.teacher_id;

-- 6. Retrieve a list of students and their enrollment dates for a specific course.

select s.student_id,s.first_name,e.enrollment_date,c.course_name from students s join enrollments e on
s.student_id=e.student_id

join courses c on e.course_id=c.course_id where c.course_name='Java';

-- 7. Find the names of students who have not made any payments. Use a LEFT JOIN between the  -- "Students"
table and the "Payments" table and filter for students with NULL payment records.

select concat('s.first_name',' ','s.last_name') as student_name from students s left join payments p on
s.student_id=p.student_id

where p.student_id=null;

-- 8. Write a query to identify courses that have no enrollments.

select c.course_id,c.course_name from courses c join enrollments e on c.course_id=e.course_id where
e.course_id=null;

-- 9. Identify students who are enrolled in more than one course.

select s.student_id,s.first_name,count(e.course_id) as count from students s left join enrollments e on
s.student_id=e.student_id

group by s.student_id,s.first_name having count(e.course_id)>1;

-- 10. Find teachers who are not assigned to any courses.

select t.teacher_id,t.first_name from teacher t join courses c on c.teacher_id=t.teacher_id where
c.teacher_id=null;

-- Task 4.

-- 1. Write an SQL query to calculate the average number of students enrolled in each course.

select c.course_id,c.course_name,avg(e.student_id) as avg_num_students_enrolled from

courses c join enrollments e on c.course_id=e.course_id group by c.course_id,c.course_name;

select course_id,course_name from courses where course_id in(select avg(student_id) as

avg_num_of_stud from enrollments);

-- 2. Identify the student(s) who made the highest payment.
```

```
select student_id from payments where amount=(select max(amount) from payments);

-- 3. Retrieve a list of courses with the highest number of enrollments.

select course_id,course_name from courses where course_id in(select max(course_id) from enrollments);

-- 4. Calculate the total payments made to courses taught by each teacher.

-- 5. Identify students who are enrolled in all available courses. Use subqueries to compare a

-- student's enrollments with the total number of courses.

select student_id,first_name from students where student_id in(select course_id from enrollments);

-- 6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to

-- find teachers with no course assignments.

select teacher_id,first_name from teacher where teacher_id not in(select course_id from courses);

-- 7. Calculate the average age of all students.

select avg(DATEDIFF(CURRENT_DATE, date_of_birth)) AS average_age FROM students;

-- 8. Identify courses with no enrollments. Use subqueries to find courses without enrollment

-- records.

select course_id,course_name FROM courses where course_id not in (select course_id from enrollments);

-- 9. Calculate the total payments made by each student for each course they are enrolled in. Use

-- subqueries and aggregate functions to sum payments.

-- 10. Identify students who have made more than one payment. Use subqueries and aggregate

-- functions to count payments per student and filter for those with counts greater than one.

select student_id from payments where amount=(select count(amount)>1 from payments);

-- 11. Write an SQL query to calculate the total payments made by each student. Join the "Students"

-- table with the "Payments" table and use GROUP BY to calculate the sum of payments for each

-- student.

select s.student_id,s.first_name,sum(amount) from students s join payments p on

s.student_id=p.student_id group by s.student_id,s.first_name;

-- 12. Retrieve a list of course names along with the count of students enrolled in each course. Use

-- JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to

-- count enrollments.

select c.course_name,count(e.student_id) as count_of_students_enrolled from courses c join enrollments e on
c.course_id=e.course_id

group by c.course_name;

-- 13. Calculate the average payment amount made by students. Use JOIN operations between the
```

```sql
-- "Students" table and the "Payments" table and GROUP BY to calculate the average.
select s.student_id,s.first_name,avg(p.amount) as avg_payment from students s join payments p on s.student_id=
p.student_id group by s.student_id,s.first_name;
```