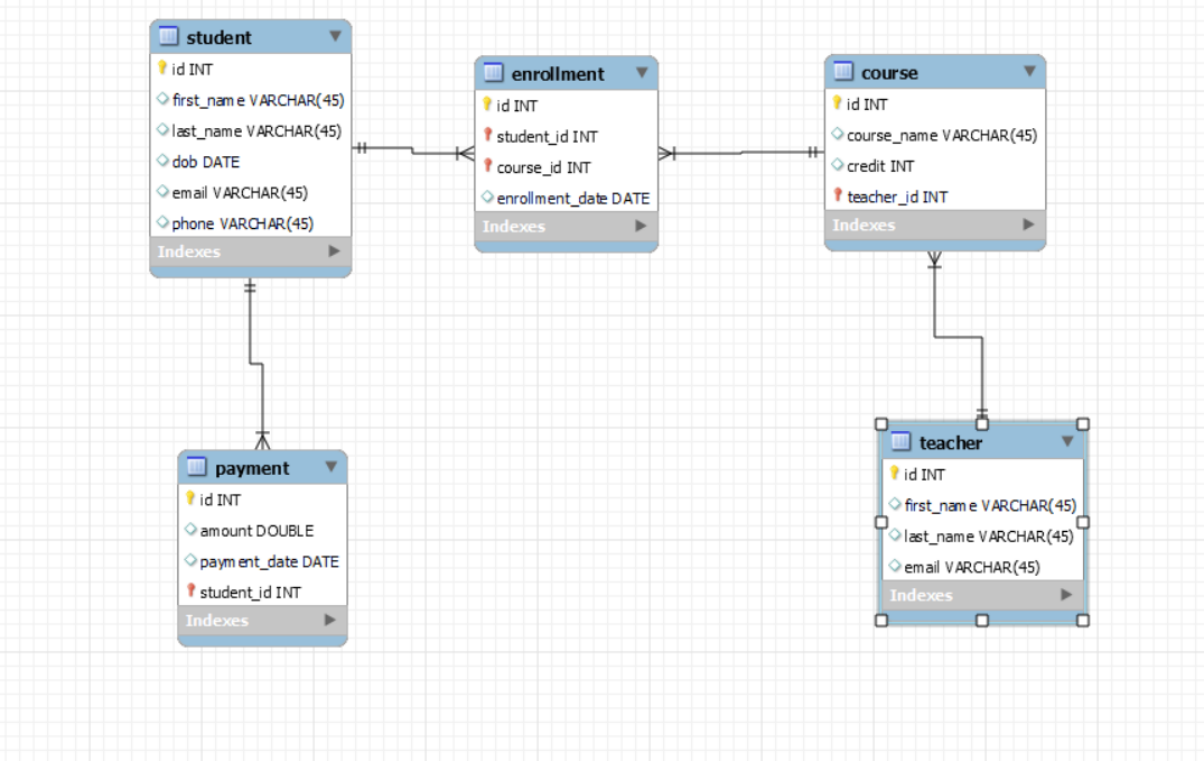
Student-Information-System-Assignment

Task-1:



Task-2:

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

-- a. First Name: David

-- b. Last Name: Doe

-- c. Date of Birth: 1995-08-15

-- d. Email: john.doe@example.com

-- e. Phone Number: 1234567890

```
insert into student(first_name,last_name,dob,email,phone)
values
('David', 'Doe', '1995-08-15', 'david.doe@example.com', '1234567890');
```

-- 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.

```
insert into enrollment (student_id,course_id,enrollment_date)
values
(5,5,'2023-12-30');
```

-- 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="garcia@example.com"
where id=3;
```

-- 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 enrollment
where student_id=1 and course_id=2;
```

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

```
update course
set teacher_id=3
where id=2;
```

-- 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 enrollment

where student_id=2;


delete from student

where id=2;
```

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

```
update payment

set amount=10000

where id=1;
```

Task-3:

-- 1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

```
select s.first_name,p.amount

from student s join payment p on s.id=p.student_id;
```

-- 2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

```
select c.id,count(c.id)

from course c join enrollment e on c.id=e.course_id
```

group by e.course_id;

-- 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.first_name

from student s left join enrollment e on s.id=e.student_id

where e.id is null;

-- 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 student s join enrollment e on s.id=e.student_id join course c on c.id=e.course_id ;

-- 5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.

select t.first_name,c.course_name

from teacher t join course c on t.id=c.teacher_id;

-- 6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

select s.first_name,c.course_name,e.enrollment_date

from student s join enrollment e on s.id=e.student_id join course c on c.id=e.course_id;

-- 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 s.first_name
from student s left join payment p on s.id=p.student_id
where p.id is null;
```

-- 8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.

```
select c.course_name
from course c left join enrollment e on c.id=e.course_id
where e.id is null;
```

-- 9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

```
select s.first_name ,s.dob,s.email,count(s.id) as no_of_course
from student s join enrollment e on s.id=e.student_id
group by e.student_id
having no_of_course >1;
```

-- 10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.

```
select t.first_name
from teacher t left join course c on t.id=c.teacher_id
where c.teacher_id is null;
```

Task-4:

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

-- aggregate functions and subqueries to achieve this.

```
select c.id,count(e.id)/(select count(id) from student) as Avg_no_of_students
from course c join enrollment e on c.id=e.course_id
group by e.course_id;
```

-- 2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.

```
select s.first_name from student s
where s.id in (select student_id from payment p where amount=(select max(amount) from payment));
```

-- 3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.

```
select c.id ,count(c.id) as Total_enrollment
from course c join enrollment e on c.id=e.course_id
group by e.course_id
order by Total_enrollment desc;
```

-- 4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.

```
select t.first_name,sum(p.amount) as total_payment

from student s join payment p on s.id=p.student_id join enrollment e on s.id=e.student_id join course c on c.id=e.course_id join teacher t on t.id=c.teacher_id

group by t.first_name;
```

-- 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 s.first_name ,count(e.id)

from student s join enrollment e on s.id=e.student_id join course c on c.id=e.course_id

group by e.student_id

having count(e.id) = (select distinct count(id) from course);
```

-- 6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

```
select first_name,last_name from teacher

where id not in (select teacher_id from course);
```

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

```
select * from course

where id not in (select course_id from enrollment);
```

-- 9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments. // doubt

```
select student_id,sum(amount) as total_payment
from payment
group by student_id;
```

-- 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 id,first_name,last_name from student
where id in (select student_id from payment group by student_id having count(id)>1);
```

-- 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 student_id,sum(amount) as total_payments from payment
group by student_id;
```

-- 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.id, count(e.id) from
enrollment e join course c on e.course_id=c.id
group by e.course_id;
```

-- 13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

```
select s.id,s.first_name,s.last_name,avg(amount) as average_payment from payment p

join student s on p.student_id=s.id

group by s.id;
```