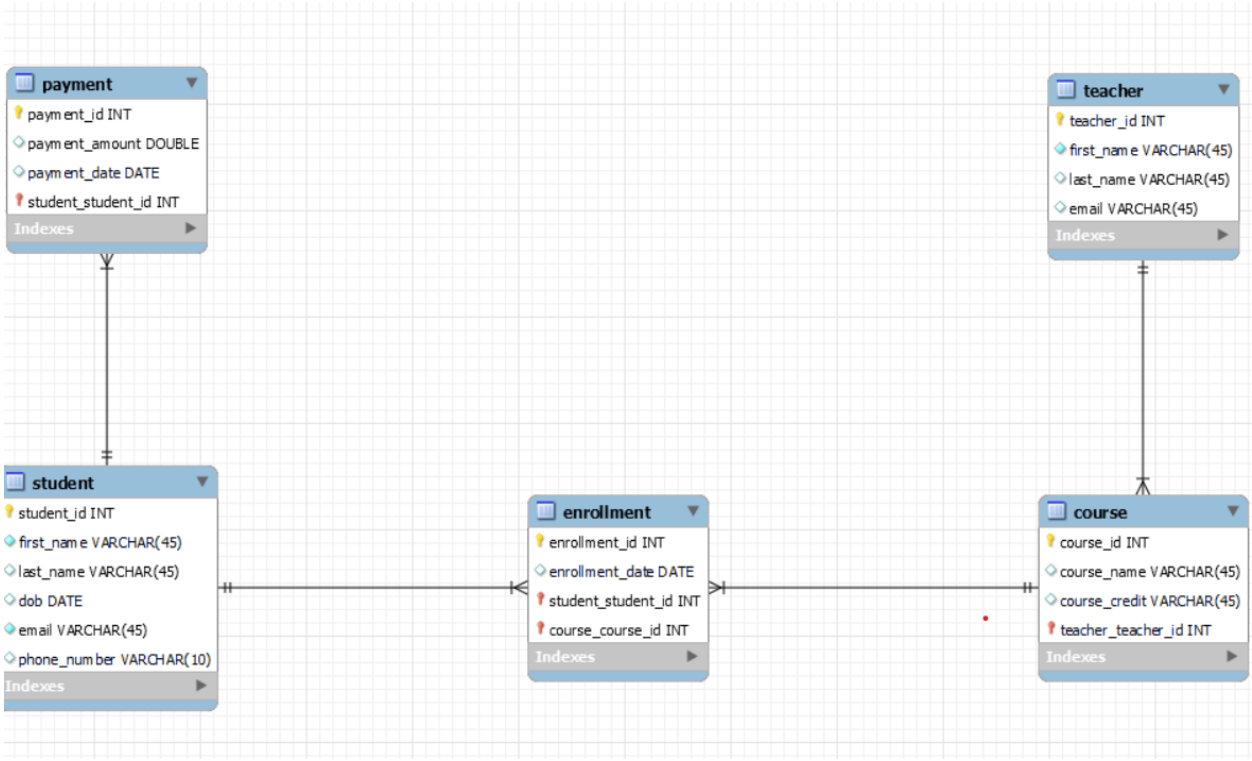
ASSIGNMENT 3

STUDENT INFORMATION SYSTEM

ER DIAGRAM:



QUERIES:

```
#02.02.2024
#MY OWN QUERIES
show databases;
create database hexfeb2024;
use hexfeb2024;
show tables;
describe teacher;
#TASK 1:
 insert into student(first_name,last_name,dob,email,phone_number)
 values('aadhya','parvin','2000-12-16','aadhaya@gmail.com','4563527');
 select * from student;
 insert into student(first_name,last_name,dob,email,phone_number)
 values('irani',null,'1999-01-29','irani@gmail.com','33625798'),
 ('pankaj','aayush','1999-02-28','pankg@gmail.com','33625798'),
 ('aarav','konda','1999-11-19','konda@gmail.com','33625798'),
('zoya','parvin','1999-07-18','zoya@gmail.com','33625798'),
('rakul','singh','2000-05-09','singh2gmail.com','33625798'),
```

```
 ('louis',null,'2000-03-06','abc@gmail.com','33625798'),
 ('yash','yadav','1999-05-27','yash@gmail.com','33625798'),
 ('goyal','kundre','1999-04-13','goyal@gmail.com','33625798'),
('akriti','pandey','2000-10-08','akriti@gmail.com','33625798');
 describe course;
 insert into teacher(first_name,last_name,email)
 values('nadhiya','gowda','abac@gmail.com');
  insert into teacher(first_name,last_name,email)
 values('rekha','nair','abac@gmail.com'),
 ('rachel',null,'abac@gmail.com'),
 ('hardik','pandey','abac@gmail.com');
 select* from enrollment;
 insert into course(course_name,course_credit,coursecol,teacher_teacher_id)
 values('java','9','A',2);
insert into course(course_name,course_credit,coursecol,teacher_teacher_id)
 values('cloud','9','A+',4),
 ('sql','10','O',3 ),
 ('python','10','O',1),
 ('js','9','A+',2),
 ('advanced java','10','A',3);
 insert into enrollment(enrollment_date,student_student_id,course_course_id)
 values('2023-01-25',1,2);
 insert into enrollment(enrollment_date,student_student_id,course_course_id)
 values('2023-02-20',2,2),
 ('2023-04-18',3,1),
 ('2023-03-03',4,3),
 ('2023-01-17',5,4);
 select * from payment;
 insert into payment(payment_id,payment_amount,payment_date,student_student_id)
 values(1,900000,'2023-01-25',1),
 (2,900000,'2023-02-20',2),
 (3,1000000,'2023-04-19',3),
 (4,150000,'2023-03-03',4),
 (5,80000,'2023-01-17',5);

 #TASK 2: Select, Where, Between, AND, LIKE:
/*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 student(first_name,last_name,dob,email,phone_number)
 values('john','doe','1999-08-15','doe@gmail.com','1234567890');
 select * from student;

 /*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(enrollment_date,student_student_id,course_course_id)
 values('2023-03-20',6,6);
 select * from enrollment;
/*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='rekha@gmail.com' where teacher_id=2;
select *from teacher;

/*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_student_id=2 and course_course_id=2;
select *from enrollment;

/*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_teacher_id=3 where course_id=1;
select *from course;

/*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 s,e from student s join enrollment e
 on s.student_id=e.student_student_id
 where s.student_id=2;


/*7. Update the payment amount for a specific payment record in the
"Payments" table. Choose any payment record and modify the payment amount.*/
update payment set payment_amount=100000 where payment_id=3;
select *from payment;
```

```
/*Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:
 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.student_id,sum(p.payment_amount) as amt
from student s,payment p
where s.student_id=p.student_student_id
group by s.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.course_name,count(e.student_student_id) as course
from course c,enrollment e
where c.course_id=e.course_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.*/#(doubt)
select s.first_name
from student s left join enrollment e
on s.student_id=e.student_student_id
where e.student_student_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,course c,enrollment e
 where c.course_id=e.course_course_id and s.student_id=e.student_student_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,t.last_name,c.course_name
from teacher t,course c
where t.teacher_id=c.teacher_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,s.last_name,e.enrollment_date,c.course_name
from student s,course c,enrollment e
where s.student_id=e.student_student_id and c.course_id=e.course_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,p.payment_amount
 from student s,payment p
 where s.student_id=p.student_student_id and p.payment_amount=0;
 update payment set payment_amount=0
 where student_student_id=5;

/*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_id,c.course_name
from course c left join enrollment e
on c.course_id=e.course_course_id where e.course_course_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 student_student_id,count(student_student_id) as cnt
from enrollment
group by student_student_id
having cnt>1;

update enrollment set student_student_id=4
where enrollment_id=5;

/*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.teacher_id,t.first_name
from teacher t left join course c
on t.teacher_id=c.teacher_teacher_id
where c.teacher_teacher_id is null;

-- Task 4. Subquery and its type:
```

```
/*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 s.student_id,avg(e.enrollment_id)
from student s,enrollment e
group by(enrollment_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 student_student_id,payment_amount
from payment
where(select max(payment_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.*/

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

-- 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 student
where student_id in(select student_student_id from enrollment);
/*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
from teacher
where teacher_id not in(select teacher_teacher_id
from course);
/*7. Calculate the average age of all students. Use subqueries to calculate the age of each student
based on their date of birth.*/

/*8. Identify courses with no enrollments. Use subqueries to find courses without enrollment
records.*/
select course_id,course_name
from course
where course_id not in(select course_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.*/
select s.student_id,(select sum(p.payment_amount) as pay from payment p
```

```
where p.student_student_id=s.student_id) as pay
from payment p,student s
group by s.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 student_student_id,count(*) as pay
from payment
group by(student_student_id)
having count(*)>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 s.student_id,sum(p.payment_amount),s.first_name
from student s join payment p
on s.student_id=p.student_student_id
group by s.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.course_id, c.course_name,count(e.student_.student_id)
from course c join enrollment e
on  c.course_id=e.course_course_id
group by c.course_id;

/*13. Calculate the average payment amount made by students. Use JOIN operations
the "Students" table and the "Payments" table and GROUP BY to calculate the average.*/
select avg(payment_amount) from payment
where student_student_id in(select student_id from student);
#or
select avg(payment_amount) as pay
from payment p join student s
on s.student_id=p.student_student_id;
```