Assignment - 3

```
use SISDB;

show tables;


describe courses;

describe students;

describe payment;

describe enrollments;

describe teacher;


-- task 1 - 5th ----------------------------------------------------------------


-- Students
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES
('Taylor', 'Swift', '1995-08-15', 'taylor.swift@gmail.com', '1234567890'),
('Nick', 'Jonas', '1998-03-20', 'nick.jonas@gmail.com', '9876543210'),
('Michael', 'Jackson', '1997-11-10', 'michael.jackson@gmail.com', '5551234567'),
('Emma', 'Watson', '1996-09-25', 'emma.watson@gmail.com', '4567890123'),
('lady', 'gaga', '1999-02-28', 'lady.gaga@gmail.com', '3216549870');
-- Courses
INSERT INTO Courses (course_name, credits, teacher_id)
VALUES
('Mathematics', 3, 1),
('English Literature', 4, 2),
('Computer Science', 3, 1),
('Art', 2, 2),
('Music', 2, 3);


-- Enrollments
INSERT INTO Enrollments (student_id, course_id, enrollment_data)
```

```
VALUES
(1, 1, '2024-01-17'),
(2, 2, '2024-02-11'),
(3, 3, '2024-03-25'),
(4, 4, '2024-01-02'),
(5, 5, '2024-02-05');

-- Teacher
INSERT INTO Teacher (first_name, last_name, email)
VALUES
('Kim', 'Taehyung', 'kim.tae@example.com'),
('Jeon', 'Jungkook', 'jeon.jung@example.com'),
('Rohit', 'Sharma', 'ro.sharma@example.com'),
('Sachin', 'Tendulkar', 'sachin.ten@example.com'),
('Alia', 'Bhaat', 'alia.rk@example.com');

-- Payments
INSERT INTO Payment (student_id, amount, payment_date)
VALUES
(1, 5000, '2024-01-17'),
(2, 1000, '2024-02-11'),
(3, 4500, '2024-03-25'),
(3, 5500, '2024-01-02'),
(1, 2500, '2024-03-05');

select * from students;
select* from Courses;
select * from payment;
select * from teacher;
select * from enrollments;
```

/*

----------Tasks 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 Students (first_name, last_name, date_of_birth, email, phone_number)

VALUES ('John', 'Doe', '1995-08-15', 'john.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 Enrollments (student_id, course_id, enrollment_data)

VALUES (2, 5, '2024-04-09');
```

-- 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 = 'newemail@example.com' WHERE teacher_id = 5;
```

-- 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 = 2 AND course_id = 5;
```

-- 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 = 2 WHERE course_id = 1;


-- 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;


-- 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 amount = 1200 WHERE payment_id = 2;


-- ----------task 3 ----------


-- 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.amount) from Students s , Payment p where s.student_id=p.student_id group by s.student_id;

/*

1	7500

2	1200

3	10000

*/


-- 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(*) from Students s join Enrollments e on s.student_id = e.student_id join courses c on e.course_id=c.course_id group by c.course_name;
```

/*

Art 1

Computer Science 1

English Literature 1

Mathematics 1

Music 1

*/

-- 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 * from Students s left join Enrollments e on s.student_id=e.student_id;

select concat(s.first_name,s.last_name) as full_name from Students s where s.student_id not in(select s.student_id from Students s join Enrollments e on s.student_id=e.student_id) ;

/*

JohnDoe

*/

-- 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 e.course_id = c.course_id ;

/*

Taylor Swift Mathematics

Nick Jonas English Literature

MichaelJackson Computer Science

Emma Watson Art

lady gaga Music

*/

-- 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 concat(t.first_name,t.last_name) as full_name , c.course_name from Courses c join Teacher t on c.teacher_id = t.teacher_id ;
```

```
/*
JeonJungkook        Mathematics
JeonJungkook        English Literature
KimTaehyung            Computer Science
JeonJungkook        Art
RohitSharma            Music
*/
```

-- 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 concat(s.first_name,s.last_name)as student_name , e.enrollment_data, 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 ='Music' ;
```

```
/*
ladygaga        2024-02-05      Music
*/
```

-- 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_with_no_payment from Students s left join Payment p on s.student_id = p.student_id where p.amount is null;
```

```
/*
EmmaWatson
ladygaga
JohnDoe
*/
```

-- 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 , count(*) from Courses c left join Enrollments e on c.course_id = e.course_id where e.enrollment_id is null ;
```

```
/*
                    0
*/
```

-- 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.last_name, COUNT(e.course_id) AS enrolled_courses FROM Students s  JOIN Enrollments e ON s.student_id = e.student_id GROUP BY s.student_id HAVING COUNT(e.course_id) > 1;
```

-- no students took more than 1 course according to given data

-- 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 concat(t.first_name,t.last_name) as teacher_name from Teacher t left join Courses c on c.teacher_id = t.teacher_id where c.course_id is null;
```

```
/*
SachinTendulkar
AliaBhaat
*/
```

-- ----------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 avg(s.student_id) , 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 group by c.course_name;


/*
4.0000  Art
3.0000  Computer Science
2.0000  English Literature
1.0000  Mathematics
5.0000  Music
*/


-- 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 ,s.last_name , sum(p.amount) from Students s , Payment p where
s.student_id=p.student_id group by s.first_name order by sum(p.amount) desc limit 1 ;


/*
MichaelJackson 10000
*/
-- 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.course_name , count(*) as enroll_count from Enrollments e join Courses c on e.course_id =
c.course_id group by c.course_name ;
select c.course_name , count(*) as enroll_count from Enrollments e join Courses c on e.course_id =
c.course_id group by c.course_name order by count(*) desc limit 1 ;


/*
Art                 1
Computer Science        1
English Literature          1
```

Mathematics 1

Music 1

*/

/*

Mathematics 1

*/

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

```
select sum(p.amount),t.first_name,t.last_name  from Payment p

join Students s on s.student_id = p.student_id

join Enrollments e on s.student_id = e.student_id

join Courses c on e.course_id = c.course_id

join Teacher t on t.teacher_id = c.teacher_id

group by t.teacher_id;
```

/*

8700 Jeon Jungkook

10000 Kim Taehyung

*/

-- 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 concat(s.first_name,s.last_name) as student_name , 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 concat(s.first_name,s.last_name)   = all(select c.course_name from Courses c);
```

/*

null output as no student is enrolled in all available courses

*/

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

select concat(t.first_name,t.last_name) as teacher_name from Teacher t left join Courses c on c.teacher_id = t.teacher_id where c.course_id is null;

/*

SachinTendulkar

AliaBhaat

*/

-- 7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.

select avg(DATEDIFF(NOW(), date_of_birth) / 365) as average_age from Students;

/*

27.08858447

*/

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

select c.course_name,count(*) from Enrollments e left join Courses c on e.course_id=c.course_id where e.enrollment_id is null group by c.course_name;

/*

all courses have atleast 1 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 sum(p.amount) as total_payment,s.student_id,c.course_name,e.enrollment_id from Payment
p JOIN Students s on p.student_id=s.student_id

JOIN Enrollments e on s.student_id=e.student_id

JOIN Courses c on e.course_id=c.course_id

group by s.student_id,c.course_name;

/*

7500    1       Mathematics     1

1200    2       English Literature      2

10000   3       Computer Science        3

*/


-- 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 Payment group by student_id having COUNT(*) > 1;


/*

1

3

*/


-- 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_payment from Payment group by student_id;


/*

1       7500

2       1200

3       10000
```

*/

-- 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 enrolled_students FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id GROUP BY c.course_id;

/*

Mathematics 1

English Literature 1

Computer Science 1

Art 1

Music 1

*/

-- 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 avg(amount) as average_payment_amount from Payment;

/*

3740.0000

*/