```
create database SISDB;
use sisdb;
show tables;


insert into students
( first_name, last_name, date_of_birth, email, phone_number) values
('Mani', 'Raushan', '2001-10-10', 'mani@gmail.com', '123'),
('Vishal', 'pandey', '2000-05-01', 'vishal@09.com', '456'),
('Rahul', 'Kumar', '2001-04-01', 'rahul@gmail.com', '789'),
('piyush', 'kumar', '2002-01-01', 'ps@gmail.com',   '980');

select * from students;

 insert into payments(student_id,amount,payment_date)
 values
 ('1','100000', '2024-06-02'),
 ('4','40000', '2024-04-09'),
 ('2','98000', '2024-02-12'),
 ('1','75000', '2024-05-06');

 select * from payments;

  insert into teacher(first_name, last_name, email)
  values
  ('robert','singh','rob@gmail.com'),
  ('william','kumar','william@gmail.com'),
   ('ronald', 'weasley','ron@gmail.com');

   select * from teacher;
```

```
insert into courses(course_name,credits,teacher_id)
values
('Java Programming', 120,2),
('Python Programming', 80,2),
('Selenium Admin', 60,3),
('Network Admin', 70,1);

select * from enrollments;

insert into enrollments(student_id,course_id,enrollment_date)
values(1,2,'2024-01-01'),
    (2,3,'2024-02-01'),
    (3,4,'2024-01-02');
```

/* 1 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_date)
values
( '1', '4','2024-01-07');
```

-- 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 = 'rob1@gmail.com'
where teacher_id = 1;
```

-- 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 = 3 and course_id = 4;
```

-- 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= 3;
select * from courses;
```

-- 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 = 3;

delete
from students
where student_id = 3;
```

-- 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 = '200000'
where payment_id = 1;
```

-- 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,sum(amount)
from students s join payments p
ON s.student_id = p.student_id
Group by s.student_id;

/*
 output

 +------------+-------------+
| first_name | sum(amount) |
+------------+-------------+
| Mani       |      275000 |
```

| piyush | 40000 |

| Vishal | 98000 |

+------------+-------------+

*/

/* 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_id)
from courses c JOIN enrollments e
ON c.course_id = e.course_id
group by course_name;
```

/*

output

+--------------------+---------------------+

| course_name | count(e.student_id) |

+--------------------+---------------------+

| Python Programming | 1 |

| Selenium Admin | 1 |

| Network Admin | 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
where e.student_id is null;        -- use is null because '==' use to compare boolean value only.



/*
 out[ut
  | student_id | first_name | last_name | date_of_birth | email              | phone_number | enrollment_id | student_id | course_id | enrollment_date |
+------------+------------+-----------+---------------+---------------------+--------------+---------------+------------+-----------+----------------+
|          4 | piyush     | kumar     | 2002-01-01    | ps@gmail.com        |          980 |          NULL |       NULL |      NULL | NULL           |
|          5 | john       | Doe       | 1995-08-15    | john.doe@example.com |   1234567890 |          NULL |       NULL |      NULL | 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 students s JOIN enrollments e
ON s.student_id = e.student_id
JOIN courses c ON e.course_id = c.course_id;



/* output  +------------+-----------+--------------------+
```

| first_name | last_name | course_name |

+------------+-----------+--------------------+

| Mani | Raushan | Python Programming |

| Vishal | pandey | Selenium Admin |

| Mani | Raushan | Network Admin |

+------------+-----------+--------------------+

*/

-- 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 JOIN courses c

ON t.teacher_id = c.teacher_id;

/* output

robert singh Selenium Admin

robert singh Network Admin

william kumar Java Programming

william kumar Python Programming

*/

-- 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

from students s JOIN enrollments e

ON s.student_id = e.student_id;

/* output

Mani Raushan 2024-01-01

Vishal pandey 2024-02-01

Mani Raushan 2024-01-07

*/

/* 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,s.last_name
from students s left join payments p
ON s.student_id = p.student_id
where amount is null;
```

-- output john Doe

/* 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 *
from courses c left join enrollments e
ON e.course_id = c.course_id
where enrollment_id is null;
```

-- output 1 Java Programming 120 2

-- 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, count(e.enrollment_id) as courses_enrolled
from students s join enrollments e
ON s.student_id = e.student_id
group by s.first_name
having courses_enrolled >1;
```

-- output Mani 2

-- 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 *
from teacher t left Join courses c
ON t.teacher_id = c.teacher_id
where c.course_id is Null;
```

-- output-- 3 ronald weasleyron@gmail.com

-- 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.

```
/* projection : enrollment
criteria : course
*/

select c.course_name,avg(e.enrollment_id)
from enrollments e join courses c
ON e.course_id = c.course_id
group by c.course_id ;

/* output

Python Programming     1.0000
Selenium Admin         2.0000
Network Admin 1.0000

*/
```

-- 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.

```
/* projection : student
criteria : payment */

select s.student_id,s.first_name,s.last_name,p.amount
```

```
from students s JOIN payments p
ON s.student_id = p.student_id
where p.amount = ( select max(amount)
            from payments);



/* output

1          Mani     Raushan           200000

*/
```

-- 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_id,c.course_name
from courses c JOIN enrollments e
ON c.course_id = e.course_id
group by c.course_id
having count(e.enrollment_id);
```

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

```
select  t.*, sum(distinct p.amount)
from teacher t join courses c
ON t.teacher_id = c.teacher_id
JOIN enrollments e ON c.course_id = e.course_id
JOIN students s ON e.student_id = s.student_id
JOIN payments p ON s.student_id = p.student_id
group by t.teacher_id;
```

```
/*
1       robert   singh    rob1@gmail.com        373000
2       william  kumar   william@gmail.com      275000
*/

 /*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, last_name
from Students
where (select count(distinct course_id)from Enrollments)
= (select count(distinct course_id)from 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 first_name, last_name
from Teacher
where teacher_id not in(select distinct teacher_id from Courses);

-- ronald          weasley

-- - 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 Courses
where course_id not in (select distinct course_id from Enrollments
);

/*
1	Java Programming

*/

-- 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 e.student_id, e.course_id, SUM(p.amount) AS total_payments
from Enrollments e
left join Payments p on e.student_id = p.student_id
group by e.student_id, e.course_id;

/*
1	2	275000
2	3	98000
1	4	275000
*/

/* 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
```

```
group by student_id
having count(*)>1;

--  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(amount) as total_payment
from students s join payments p
ON s.student_id=p.student_id
group by s.student_id ;

/*
1	275000
4	40000
2	98000
*/

/* 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 enrollment_count
from Courses c
left join Enrollments e on c.course_id = e.course_id
group by c.course_name;
```

```
/*
Java Programming        0
Python Programming      1
Selenium Admin          1
Network Admin 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 s.student_id,avg(amount)as avg_amount
from students s join payments p
ON s.student_id=p.student_id
group by s.student_id

/*
1       137500.0000
4       40000.0000
2       98000.0000
*/
```