```
use mydb_sms;

show tables;

describe courses;
describe enrollments;
describe payments;
describe students;
describe teacher;

insert into students(stu_first_name, stu_last_name, stu_dob, stu_email, stu_phone_number) values
('Ankit', 'Singh', '2001-04-13', 'ank@gmail.com', '626399');

insert into students(stu_first_name, stu_last_name, stu_dob, stu_email, stu_phone_number) values
('harry', 'potter', '2000-04-13', 'harry@gmail.com', '626399'),
('hermione', 'granger', '2001-04-13', 'granger@gmail.com', '9116399'),
('nattu', 'kakka', '1947-04-13', 'nattu@gmail.com', '516399'),
('jetha', 'lal', '2000-04-13', 'jetha@gmail.com', '626399');

select * from students;


insert into teacher(teacher_first_name, teacher_last_name, teacher_email) values
('Mr.Java', 'Singh', 'j@gmail.com'),
('Mrs.Python', 'Singh', 'p@gmail.com'),
('Droid', 'Singh', 'd@gmail.com');

select * from teacher;


insert into courses(course_id, course_name, course_credits, teacher_id) values
(1, 'Java Programming', 5, 1),
(2, 'Python Programming', 4, 2),
(3, 'Android Development', 10, 3);

select * from courses;

insert into enrollments(stu_id, course_id, enrollment_date) values
(1, 1, '2024-04-08'),
(2, 2, '2024-04-08'),
(3, 3, '2024-04-08');
insert into enrollments(stu_id, course_id, enrollment_date) values
(1, 2, '2024-04-09');
select * from enrollments;

insert into payments(payment_amount, payment_date, stu_id) values
(10000, '2024-04-10', 1),
(9000, '2024-04-10', 1),
(7000, '2024-04-10', 2),
(3000, '2024-04-10', 3);
```

```
select * from payments;



-- 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. dob: '1995-08-15'
-- d. Email: john.doe@example.com
insert into students(stu_first_name, stu_last_name, stu_dob, stu_email, stu_phone_number) values
('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');

select * from students;
/* Output
'1','Ankit','Singh','2001-04-13','ank@gmail.com','626399'
'2','harry','potter','2000-04-13','harry@gmail.com','626399'
'3','hermione','granger','2001-04-13','granger@gmail.com','9116399'
'4','nattu','kakka','1947-04-13','nattu@gmail.com','516399'
'5','jetha','lal','2000-04-13','jetha@gmail.com','626399'
'6','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(stu_id, course_id, enrollment_date) values
(2, 2, '2024-04-08');
select * from enrollments;
/* output
'1','1','1','2024-04-08'
'2','2','2','2024-04-08'
'3','3','3','2024-04-08'
'1','2','4','2024-04-09'
'2','2','5','2024-04-08'
*/



-- 3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their
email address.
select * from teacher;
update teacher
set teacher_email = 'droid@gmail.com'
where teacher_id = 3;
/* output
'1','Mr.Java','Singh','j@gmail.com'
'2','Mrs.Python','Singh','p@gmail.com'
'3','Droid','Singh','droid@gmail.com'
*/
```

```
-- 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 stu_id = 2
and course_id = 2;
/* output
'1','1','1','2024-04-08'
'3','3','3','2024-04-08'
'1','2','4','2024-04-09'
*/


-- 5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the
respective tables.
select * from teacher;
select * from courses;
update courses
set teacher_id = 3
where course_id = 3;
/* output
'1','Java Programming','5','1'
'2','Python Programming','4','2'
'3','Android Development','10','3'
*/


-- 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 students
where stu_id = 6;
select * from students;
/* output
'1','Ankit','Singh','2001-04-13','ank@gmail.com','626399'
'2','harry','potter','2000-04-13','harry@gmail.com','626399'
'3','hermione','granger','2001-04-13','granger@gmail.com','9116399'
'4','nattu','kakka','1947-04-13','nattu@gmail.com','516399'
'5','jetha','lal','2000-04-13','jetha@gmail.com','626399'
*/


-- 7. Update the payment amount for a specific payment record in the "Payments" table.
-- Choose any payment record and modify the payment amount.
select * from payments;
update payments
set payment_amount = 9900
where payment_id = 2;
/* output
```

'1','10000','2024-04-10','1'
'2','9900','2024-04-10','1'
'3','7000','2024-04-10','2'
'4','3000','2024-04-10','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.stu_id, sum(payment_amount) as total_payment
from students s join payments p on s.stu_id = p.stu_id
group by stu_id;
/* output
'1','19900'
'2','7000'
'3','3000'
*/

-- 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(*) as stu_count
from students s
join enrollments e on s.stu_id = e.stu_id
join courses c on e.course_id = c.course_id
group by c.course_name;
/* output
'Java Programming','1'
'Android Development','1'
'Python Programming','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 s.*
from students s left join enrollments e on s.stu_id = e.stu_id
where e.stu_id is null;
/* output
'2','harry','potter','2000-04-13','harry@gmail.com','626399'
'4','nattu','kakka','1947-04-13','nattu@gmail.com','516399'
'5','jetha','lal','2000-04-13','jetha@gmail.com','626399'
*/

-- 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.stu_first_name, s.stu_last_name, c.course_name
from students s
left join enrollments e on s.stu_id = e.stu_id
left join courses c on e.course_id = c.course_id;
/* output
'Ankit','Singh','Java Programming'
'Ankit','Singh','Python Programming'
'harry','potter',NULL
'hermione','granger','Android Development'
'nattu','kakka',NULL
'jetha','lal',NULL
*/


-- 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.teacher_first_name, t.teacher_last_name, c.course_name
from teacher t left join courses c on t.teacher_id = c.teacher_id;
/* output
'Mr.Java','Singh','Java Programming'
'Mrs.Python','Singh','Python Programming'
'Droid','Singh','Android Development'
*/


-- 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.stu_id, s.stu_first_name, s.stu_last_name, e.enrollment_date
from students s
left join enrollments e on s.stu_id = e.stu_id
left join courses c on e.course_id = c.course_id
where course_name = 'Java Programming';
/* output
'1','Ankit','Singh','2024-04-08'
*/

-- 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 *
from students s
left join payments p on s.stu_id = p.stu_id
where payment_id is null;
/* output
'4','nattu','kakka','1947-04-13','nattu@gmail.com','516399',NULL,NULL,NULL,NULL
'5','jetha','lal','2000-04-13','jetha@gmail.com','626399',NULL,NULL,NULL,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_id
from courses c
left join enrollments e on e.course_id = c.course_id
where e.course_id is null;
/* output
No such course
*/


-- 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.stu_id
from students s
join enrollments e on s.stu_id = e.stu_id
join courses c on e.course_id = c.course_id
group by stu_id
having count(*) > 1;
/* output
'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.*
from teacher t
left join courses c on t.teacher_id = c.teacher_id
where c.teacher_id is null;
/* output
nothing - Every teacher is assigned with a course
*/



-- 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 c.course_name, avg(e.stu_id) as average_stu
from enrollments e
join courses c on e.course_id = c.course_id
group by c.course_name;
/* output
'Java Programming','1.0000'
'Android Development','3.0000'
'Python Programming','1.0000'
*/


-- 2. Identify the students who made the highest payment.
```

```
-- Use a subquery to find the maximum payment amount and then retrieve the students) associated with that
amount.
select *
from students s join payments p on p.stu_id = s.stu_id
where p.payment_amount = (select max(payment_amount)
                                                    from payments);
/*
'1','Ankit','Singh','2001-04-13','ank@gmail.com','626399','1','10000','2024-04-10','1'
*/


-- 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 enrollment_count
from courses c
join enrollments e on e.course_id = c.course_id
group by c.course_name
order by enrollment_count desc
limit 1;
/* output
'Java Programming','1'
*/


-- 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(payment_amount) as total_payment
from payments p
join students s on p.stu_id = s.stu_id
join enrollments e on s.stu_id = e.stu_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;
/* output
'1','Mr.Java','Singh','j@gmail.com','19900'
'3','Droid','Singh','droid@gmail.com','3000'
'2','Mrs.Python','Singh','p@gmail.com','19900'
*/


-- 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 e.stu_id
from students s
join enrollments e on s.stu_id = e.stu_id
join courses c on e.course_id = c.course_id
group by s.stu_id
having count(distinct e.course_id) = (select count(*)
                                                    from courses);
/* output
```

```
no such student
*/

-- 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_first_name, teacher_last_name
from teacher
where teacher_id not in(select distinct teacher_id
                                                    from courses);
/* output
Blank - every teacher is been assigned to a course
*/


-- 7. Calculate the average age of all students.
-- not possible without age column



-- 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
);
/* output
null
*/


-- 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.stu_id, c.course_name, sum(p.payment_amount)
from payments p
join students s on p.stu_id = s.stu_id
join enrollments e on s.stu_id = e.stu_id
right join courses c on e.course_id = c.course_id
group by s.stu_id, c.course_name;
/* output
'1','Java Programming','19900'
'1','Python Programming','19900'
'3','Android Development','3000'
*/

-- 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 s.stu_id, count(*) as payment_count
from students s
```

```
join payments p on p.stu_id = s.stu_id
group by s.stu_id;
/* output
'1','2'
'2','1'
'3','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.stu_id, sum(payment_amount) as total_amount
from students s
left join payments p on p.stu_id = s.stu_id
group by s.stu_id;
/* output
'1','19900'
'2','7000'
'3','3000'
*/


-- 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(s.stu_id) as stu_count
from students s
join enrollments e on s.stu_id = e.stu_id
join courses c on e.course_id = c.course_id
group by c.course_name;
/* output
'Python Programming','1'
'Java Programming','1'
'Android Development','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.stu_id, avg(payment_amount) as avg_payment
from students s
left join payments p on p.stu_id = s.stu_id
group by s.stu_id;
/* output
'1','9950.0000'
'2','7000.0000'
'3','3000.0000'
'4',NULL
'5',NULL
```

*/