Rakshitha Devi J
superset id: 5369940
Saveetha engineering college

Student Information System (SIS)

Task 1. database design:

1.database:

```
create database sisdb;
use sisdb;
```

2.tables:

```
create table students (
    student_id int auto_increment primary key,
    first_name varchar(50),
    last_name varchar(50),
    dob date,
    email varchar(100) unique,
    phone_no varchar(15) unique
);

create table teacher (
    teacher_id int auto_increment primary key,
    first_name varchar(50),
    last_name varchar(50),
    email varchar(100) unique
);

create table courses (
    course_id int auto_increment primary key,
    course_name varchar(100),
    credits int,
    teacher_id int,
    foreign key (teacher_id) references teacher(teacher_id)
);

create table enrollments (
    enrollment_id int auto_increment primary key,
    student_id int,
    course_id int,
    enrollment_date date,
    foreign key (student_id) references students(student_id),
```

```
    foreign key (course_id) references courses(course_id)
);
```

```
create table payments (
    payment_id int auto_increment primary key,
    student_id int,
    amount decimal(10,2),
    payment_date date,
    foreign key (student_id) references students(student_id)
);
```

3.information:
student detail information:

```
insert into students (first_name, last_name, dob, email, phone_no) values
('amit', 'sharma', '1995-08-21', 'amitsharma@gmail.com', '9876543210'),
('priya', 'verma', '1998-02-15', 'priyaverma@gmail.com', '9812345678'),
('rohit', 'singh', '1997-06-30', 'rohitsingh@gmail.com', '9845678901'),
('neha', 'gupta', '1996-12-10', 'nehagupta@gmail.com', '9934567890'),
('vikram', 'rao', '1999-07-25', 'vikramrao@gmail.com', '9786543210'),
('anjali', 'mishra', '2000-03-18', 'anjalimishra@gmail.com', '9754321098'),
('suresh', 'yadav', '1994-09-05', 'sureshyadav@gmail.com', '9898765432'),
('kavita', 'desai', '1993-11-28', 'kavitadesai@gmail.com', '9923456781'),
('arjun', 'iyer', '1995-05-14', 'arjuniyer@gmail.com', '9988776655'),
('meera', 'nair', '1996-01-07', 'meeranair@gmail.com', '9876123450');
```

student_id	first_name	last_name	dob	email	phone_no
2	Priya	Verma	1998-02-15	priyaverma@gmail.com	9812345678
3	Rohit	Singh	1997-06-30	rohitsingh@gmail.com	9845678901
4	Neha	Gupta	1996-12-10	nehagupta@gmail.com	9934567890
5	Vikram	Rao	1999-07-25	vikramrao@gmail.com	9786543210
6	Anjali	Mishra	2000-03-18	anjalimishra@gmail.com	9754321098
7	Suresh	Yadav	1994-09-05	sureshyadav@gmail.com	9898765432
8	Kavita	Desai	1993-11-28	kavitadesai@gmail.com	9923456781
9	Arjun	Iyer	1995-05-14	arjuniyer@gmail.com	9988776655
10	Meera	Nair	1996-01-07	meeranair@gmail.com	9876123450
NULL	NULL	NULL	NULL	NULL	NULL

teacher detail:

insert into teacher (first_name, last_name, email) values

('dr. rajesh', 'sharma', 'rajeshsharma@example.com'),
('prof. ananya', 'iyer', 'ananyaiyer@example.com'),
('dr. vikram', 'verma', 'vikramverma@example.com'),
('prof. priya', 'nair', 'priyanair@example.com'),
('dr. suresh', 'patel', 'sureshpatel@example.com'),
('prof. kavita', 'reddy', 'kavitareddy@example.com'),
('dr. arjun', 'malhotra', 'arjunmalhotra@example.com'),
('prof. meera', 'joshi', 'meerajoshi@example.com'),
('dr. sanjay', 'desai', 'sanjaydesai@example.com'),
('prof. neha', 'choudhary', 'nehachoudhary@example.com');

teacher_id	first_name	last_name	email
1	Dr. Rajesh	Sharma	rajeshsharma@example.com
2	Prof. Ananya	Iyer	ananyaiyer@example.com
3	Dr. Vikram	Verma	vikramverma@example.com
4	Prof. Priya	Nair	priyanair@example.com
5	Dr. Suresh	Patel	sureshpatel@example.com
6	Prof. Kavita	Reddy	kavitareddy@example.com
7	Dr. Arjun	Malhotra	arjunmalhotra@example.com
8	Prof. Meera	Joshi	meerajoshi@example.com
9	Dr. Sanjay	Desai	sanjaydesai@example.com
10	Prof. Neha	Choudhary	nehachoudhary@example.com
NULL	NULL	NULL	NULL

course details:

insert into courses (course_name, credits, teacher_id) values
('database systems', 4, 1),
('computer networks', 3, 2),
('operating systems', 4, 3),
('software engineering', 3, 4),
('artificial intelligence', 4, 5),
('data structures', 3, 6),
('machine learning', 4, 7),
('cyber security', 3, 8),
('cloud computing', 4, 9),
('big data analytics', 3, 10);

course_id	course_name	credits	teacher_id
1	Database Systems	4	1
2	Computer Networks	3	2
3	Operating Systems	4	3
4	Software Engineering	3	4
5	Artificial Intelligence	4	5
6	Data Structures	3	6
7	Machine Learning	4	7
8	Cyber Security	3	8
9	Cloud Computing	4	9
10	Big Data Analytics	3	10
NULL	NULL	NULL	NULL

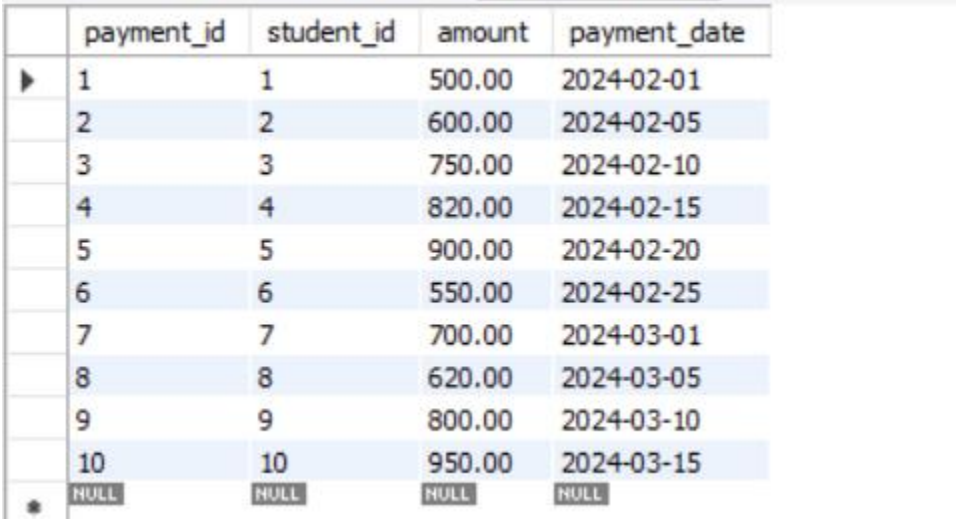
enrollment details:

```
insert into enrollments (student_id, course_id, enrollment_date) values
(1, 1, '2024-01-10'),
(2, 2, '2024-01-15'),
(3, 3, '2024-02-05'),
(4, 4, '2024-02-12'),
(5, 5, '2024-03-01'),
(6, 6, '2024-03-15'),
(7, 7, '2024-04-05'),
(8, 8, '2024-04-20'),
(9, 9, '2024-05-01'),
(10, 10, '2024-05-10');
```

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-15
3	3	3	2024-02-05
4	4	4	2024-02-12
5	5	5	2024-03-01
6	6	6	2024-03-15
7	7	7	2024-04-05
8	8	8	2024-04-20
9	9	9	2024-05-01
10	10	10	2024-05-10
NULL	NULL	NULL	NULL

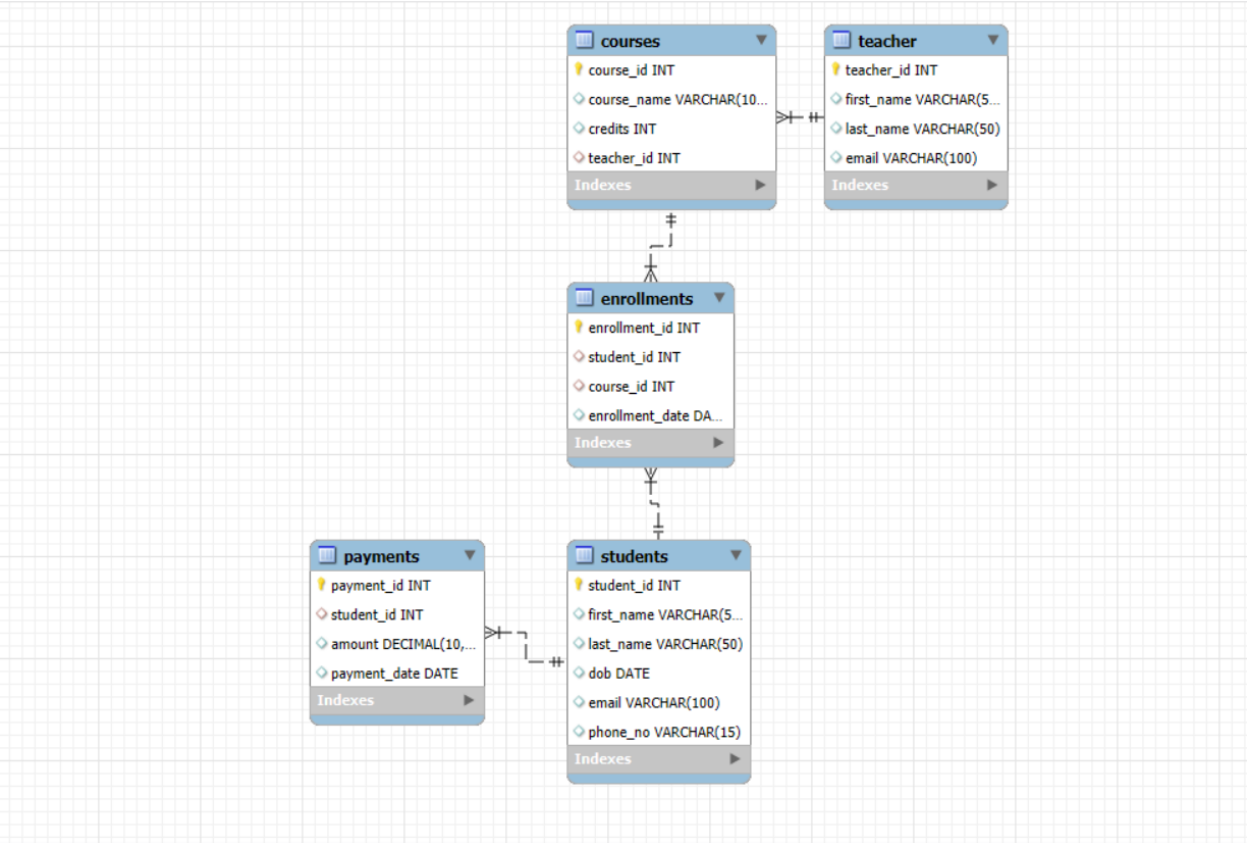
payment details:

```
insert into payments (student_id, amount, payment_date) values
(1, 500.00, '2024-02-01'),
(2, 600.00, '2024-02-05'),
(3, 750.00, '2024-02-10'),
(4, 820.00, '2024-02-15'),
(5, 900.00, '2024-02-20'),
(6, 550.00, '2024-02-25'),
(7, 700.00, '2024-03-01'),
(8, 620.00, '2024-03-05'),
(9, 800.00, '2024-03-10'),
(10,950.00, '2024-03-15');
```



payment_id	student_id	amount	payment_date
1	1	500.00	2024-02-01
2	2	600.00	2024-02-05
3	3	750.00	2024-02-10
4	4	820.00	2024-02-15
5	5	900.00	2024-02-20
6	6	550.00	2024-02-25
7	7	700.00	2024-03-01
8	8	620.00	2024-03-05
9	9	800.00	2024-03-10
10	10	950.00	2024-03-15
NULL	NULL	NULL	NULL

4.er diagram:



Tasks 2: select, where, between, and, like:

data manipulation queries

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, dob, email, phone_no)
values ('john', 'doe', '1995-08-15', 'john.doe@example.com', '1234567890');
```

student_id	first_name	last_name	dob	email	ph
1	Amit	Sharma	1995-08-21	amitsharma@gmail.com	987
2	Priya	Verma	1998-02-15	priyaverma@gmail.com	981
3	Rohit	Singh	1997-06-30	rohitsingh@gmail.com	984
4	Neha	Gupta	1996-12-10	nehagupta@gmail.com	993
5	Vikram	Rao	1999-07-25	vikramrao@gmail.com	978
6	Anjali	Mishra	2000-03-18	anjalimishra@gmail.com	975
7	Suresh	Yadav	1994-09-05	sureshyadav@gmail.com	989
8	Kavita	Desai	1993-11-28	kavitadesai@gmail.com	992
9	Arjun	Iyer	1995-05-14	arjuniyer@gmail.com	998
10	Meera	Nair	1996-01-07	meeranair@gmail.com	987
11	John	Doe	1995-08-15	john.doe@example.com	123
NULL	NULL	NULL	NULL	NULL	NULL

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, 2, curdate());

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-15
3	3	3	2024-02-05
4	4	4	2024-02-12
5	5	5	2024-03-01
6	6	6	2024-03-15
7	7	7	2024-04-05
8	8	8	2024-04-20
9	9	9	2024-05-01
10	10	10	2024-05-10
11	1	2	2025-03-19
NULL	NULL	NULL	NULL

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=1;

teacher_id	first_name	last_name	email
1	Dr. Rajesh	Sharma	newemail@example.com
2	Prof. Ananya	Iyer	ananyaiyer@example.com
3	Dr. Vikram	Verma	vikramverma@example.com
4	Prof. Priya	Nair	priyanair@example.com
5	Dr. Suresh	Patel	sureshpatel@example.com
6	Prof. Kavita	Reddy	kavitareddy@example.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 student_id=1 and course_id=2;

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-15
3	3	3	2024-02-05
4	4	4	2024-02-12
5	5	5	2024-03-01
6	6	6	2024-03-15
7	7	7	2024-04-05
8	8	8	2024-04-20
9	9	9	2024-05-01
10	10	10	2024-05-10
NULL	NULL	NULL	NULL

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;

course_id	course_name	credits	teacher_id
1	Database Systems	4	2
2	Computer Networks	3	2
3	Operating Systems	4	3
4	Software Engineering	3	4
5	Artificial Intelligence	4	5
6	Data Structures	3	6
7	Machine Learning	4	7
8	Cyber Security	3	8
9	Cloud Computing	4	9
10	Big Data Analytics	3	10
NULL	NULL	NULL	NULL

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=2;

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
3	3	3	2024-02-05
4	4	4	2024-02-12
5	5	5	2024-03-01
6	6	6	2024-03-15
7	7	7	2024-04-05
8	8	8	2024-04-20
9	9	9	2024-05-01
10	10	10	2024-05-10
NULL	NULL	NULL	NULL

Enrollments 10

delete from students where student_id=2;

0	**19**	**15:45:2**	**delete from students where student_id=2**	**error code: 1451. cannot delete or update a parent row: a foreign key constraint fails (`sisdb`.`payments`, constraint `payments_ibfk_1` foreign key**

(`student_id`) references `students`
(`student_id`))

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=750.00 where payment_id=1;

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.first_name, s.last_name, sum(p.amount) as total_paid
from students s
join payments p on s.student_id = p.student_id
group by s.student_id;
```

first_name	last_name	total_paid
Amit	Sharma	500.00
Priya	Verma	600.00
Rohit	Singh	750.00
Neha	Gupta	820.00
Vikram	Rao	900.00
Anjali	Mishra	550.00
Suresh	Yadav	700.00
Kavita	Desai	620.00
Arjun	Iyer	800.00

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) as enrolled_students
from courses c
left join enrollments e on c.course_id = e.course_id
group by c.course_id;
```

course_name	enrolled_students
Computer Networks	1
Operating Systems	1
Software Engineering	1
Artificial Intelligence	1
Data Structures	1
Machine Learning	1
Cyber Security	1
Cloud Computing	1
Big Data Analytics	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.first_name,s.last_name
from students s
left join enrollments e on s.student_id = e.student_id
where e.student_id is null;
```

first_name	last_name
John	Doe

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;
```

first_name	last_name	course_name
Amit	Sharma	Database Systems
Priya	Verma	Computer Networks
Rohit	Singh	Operating Systems
Neha	Gupta	Software Engineering
Vikram	Rao	Artificial Intelligence
Anjali	Mishra	Data Structures
Suresh	Yadav	Machine Learning
Kavita	Desai	Cyber Security
Arjun	Iyer	Cloud Computing

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
left join courses c on t.teacher_id = c.teacher_id;
```

first_name	last_name	course_name
Dr. Rajesh	Sharma	NULL
Prof. Ananya	Iyer	Database Systems
Prof. Ananya	Joshi	Computer Networks
Dr. Vikram	[illegible]	Operating Systems
Prof. Priya	Nair	Software Engineering
Dr. Suresh	Patel	Artificial Intelligence
Prof. Kavita	Reddy	Data Structures
Dr. Arjun	Malhotra	Machine Learning
Prof. Meera	Joshi	Cyber Security
Dr. Sanjay	Desai	Cloud Computing

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, c.course_name, e.enrollment_date
from students s
join enrollments e on s.student_id = e.student_id
join courses c on e.course_id = c.course_id;
order by c.course_name;
```

	first_name	last_name	course_name	enrollment_date
▶	Vikram	Rao	Artificial Intelligence	2024-03-01
	Meera	Nair	Big Data Analytics	2024-05-10
	Arjun	Iyer	Cloud Computing	2024-05-01
	Priya	Verma	Computer Networks	2024-01-15
	Kavita	Desai	Cyber Security	2024-04-20
	Anjali	Mishra	Data Structures	2024-03-15
	Amit	Sharma	Database Systems	2024-02-12
	Suresh	Yadav	Machine Learning	2024-04-05
	Rohit	Singh	Operating Systems	2024-02-05
	Neha	Gupta	Software Engineering	2024-02-12

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 p.payment_id is null;
```

	first_name	last_name
▶	John	Doe

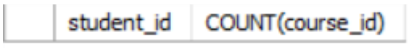
8. write a query to identify courses that have no enrollments. you'll need to use a left join between the "courses" table and the "enrollments" table and filter for courses with null enrollment records.

```
select course_name from courses
where course_id not in (select distinct course_id from enrollments);
```

	course_name

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.

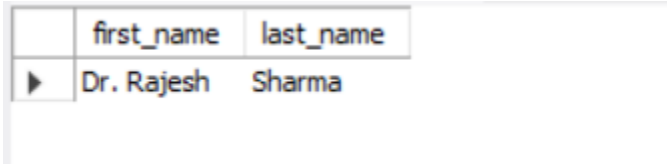
```
select student_id, count(course_id) from enrollments
group by student_id having count(course_id) > 1;
```



student_id	COUNT(course_id)

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 first_name, last_name from teacher
where teacher_id not in (select distinct teacher_id from courses);
```

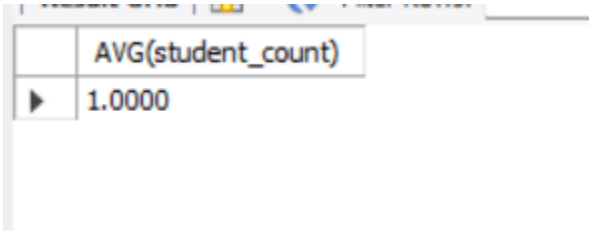


first_name	last_name
Dr. Rajesh	Sharma

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.

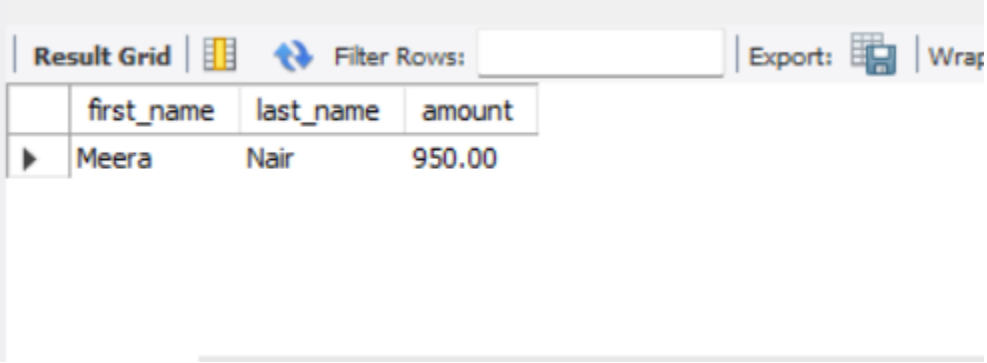
```
select avg(student_count) from (
select course_id, count(student_id) as student_count from enrollments group by course_id
) as subquery;
```



AVG(student_count)
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.

```
select s.first_name, s.last_name, p.amount as amount
from students s
join payments p on s.student_id = p.student_id
where p.amount = (select max(amount) from payments);
```



first_name	last_name	amount
Meera	Nair	950.00

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(e.student_id) as enrollment_count
from courses c
join enrollments e on c.course_id = e.course_id
group by c.course_id
having count(e.student_id) = (
select max(enrollment_count)
from (select count(student_id) as enrollment_count from enrollments group by course_id) as
subquery
);
```

course_name	enrollment_count
Database Systems	1
Operating Systems	1
Software Engineering	1
Artificial Intelligence	1
Data Structures	1
Machine Learning	1
Cyber Security	1
Cloud Computing	1
Big Data Analytics	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.first_name,t.last_name,(select sum(p.amount) from payments p
 where p.student_id in (select e.student_id from enrollments e where e.course_id in (select
c.course_id from courses c where c.teacher_id=t.teacher_id)))as total_payments from teacher t;
```

	first_name	last_name	total_payments
▶	Prof. Ananya	Iyer	500.00
	Dr. Vikram	Verma	750.00
	Prof. Priya	Nair	820.00
	Dr. Suresh	Patel	900.00
	Prof. Kavita	Reddy	550.00
	Dr. Arjun	Malhotra	700.00
	Prof. Meera	Joshi	620.00
	Dr. Sanjay	Desai	800.00
	Prof. Neha	Choudhary	950.00

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, s.last_name
from students s
where (select count(distinct course_id) from enrollments where student_id = s.student_id) =
(select count(course_id) from courses);
```

	first_name	last_name

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

```
select t.first_name, t.last_name
from teacher t
where not exists (select 1 from courses c where c.teacher_id = t.teacher_id);
```

	first_name	last_name
▶	Dr. Rajesh	Sharma

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(age) as avg_age
from (
select year(curdate()) - year(dob) as age
from students
) as subquery;
```

	avg_age
▶	28.8182

8. identify courses with no enrollments. use subqueries to find courses without enrollment records.

```
select c.course_name
from courses c
where not exists (select 1 from enrollments e where e.course_id = c.course_id);
```

	course_name
▶	Computer Networks

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.first_name, s.last_name,c.course_name,
```

```
(select sum(p.amount)
from payments p
where p.student_id = e.student_id) as total_paid
from enrollments e
join students s on e.student_id = s.student_id
join courses c on e.course_id = c.course_id;
```

first_name	last_name	course_name	total_paid
Amit	Sharma	Database Systems	500.00
Rohit	Singh	Operating Systems	750.00
Neha	Gupta	Software Engineering	820.00
Vikram	Rao	Artificial Intelligence	900.00
Anjali	Mishra	Data Structures	550.00
Suresh	Yadav	Machine Learning	700.00
Kavita	Desai	Cyber Security	620.00
Arjun	Iyer	Cloud Computing	800.00
Meera	Nair	Big Data Analytics	950.00

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.first_name, s.last_name,
(select count(*)
 from payments p
where p.student_id = s.student_id) as payment_count
from students s
where (select count(*)
from payments p
where p.student_id = s.student_id) > 1;
```

first_name	last_name	payment_count

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, first_name, last_name,
(select coalesce(sum(amount), 0)
```

```
from payments
  where payments.student_id = students.student_id) as total_paid
  from students;
```

.

first_name	last_name	total_paid
Amit	Sharma	500.00
Priya	Verma	600.00
Rohit	Singh	750.00
Neha	Gupta	820.00
Vikram	Rao	900.00
Anjali	Mishra	550.00
Suresh	Yadav	700.00
Kavita	Desai	620.00
Arjun	Iyer	800.00
Meera	Nair	950.00

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 course_name,
(select count(*)
 from enrollments
 where enrollments.course_id = courses.course_id) as enrolled_students
 from courses;
```

course_name	enrolled_students
Database Systems	1
Computer Networks	0
Operating Systems	1
Software Engineering	1
Artificial Intelligence	1
Data Structures	1
Machine Learning	1
Cyber Security	1
Cloud Computing	1
Big Data Analytics	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.first_name, s.last_name,
(select avg(p.amount)
from payments p
where p.student_id = s.student_id) as avg_payment
from students s
join payments p on s.student_id = p.student_id
group by s.student_id;
```

first_name	last_name	avg_payment
Amit	Sharma	500.000000
Priya	Verma	600.000000
Rohit	Singh	750.000000
Neha	Gupta	820.000000
Vikram	Rao	900.000000
Anjali	Mishra	550.000000
Suresh	Yadav	700.000000
Kavita	Desai	620.000000
Arjun	Iyer	800.000000
Meera	Nair	950.000000