Assignment SQL - Student Information System
Meenakshi M - Hexaware Trainee

Task 1: Database Design

1. Create the database named "SISDB"

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

2	14:33:44	create database sisdb	
3	14:33:58	use sisdb	

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

a. Students

```
create table Students(student_id int primary key auto_increment,
first_name varchar(40) not null,
last_name varchar(40),
date_of_birth date not null,
email varchar(100) unique not null,
phone_number varchar(20) not null unique);
```

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(40)	NO		NULL	
last_name	varchar(40)	YES		NULL	
date_of_birth	date	NO		NULL	
email	varchar(100)	NO	UNI	NULL	
phone_number	varchar(20)	NO	UNI	NULL	

b. Courses

```
create table Courses(course_id int primary key auto_increment,
course_name varchar(100) not null,
credits int not null,
teacher_id int,
foreign key (teacher_id) references Teacher(teacher_id) );
```

	Field	Type	Null	Key	Default	Extra
▶	course_id	int	NO	PRI	NULL	auto_increment
	course_name	varchar(100)	NO		NULL	
	credits	int	NO		NULL	
	teacher_id	int	YES	MUL	NULL	

c. Enrollments

```
create table Enrollments(enrollment_id int primary key auto_increment,
student_id int not null,
course_id int not null,
enrollment_date date not null,
foreign key (student_id) references Students (student_id) on delete cascade,
foreign key (course_id) references Courses(course_id));
```

	Field	Type	Null	Key	Default	Extra
▶	enrollment_id	int	NO	PRI	NULL	auto_increment
	student_id	int	NO	MUL	NULL	
	course_id	int	NO	MUL	NULL	
	enrollment_date	date	NO		NULL	

d. Teacher

```
create table Teacher(teacher_id int primary key auto_increment,
first_name varchar(40) not null,
last_name varchar(40),
email varchar(100) unique not null);
```

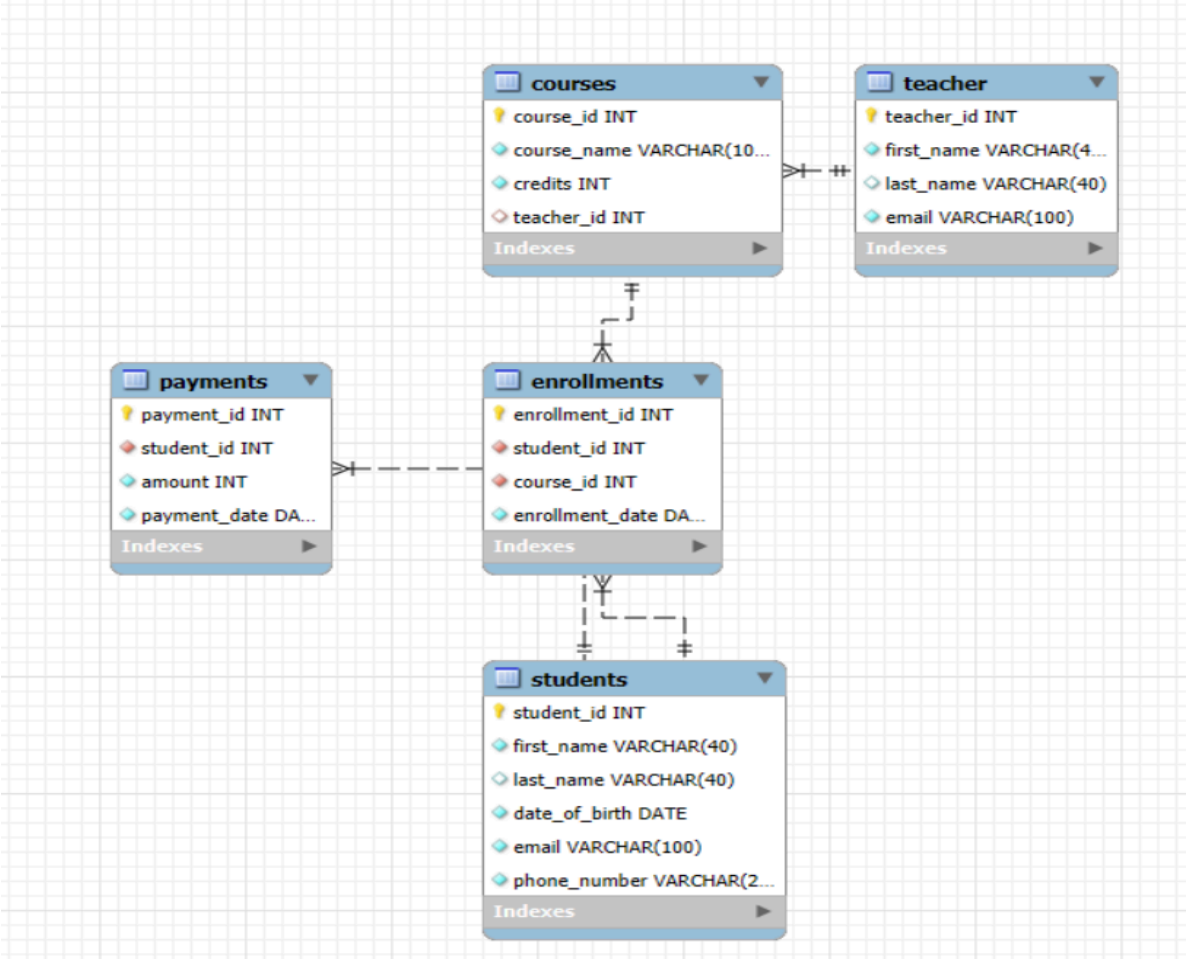
	Field	Type	Null	Key	Default	Extra
▶	teacher_id	int	NO	PRI	NULL	auto_increment
	first_name	varchar(40)	NO		NULL	
	last_name	varchar(40)	YES		NULL	
	email	varchar(100)	NO	UNI	NULL	

e. Payments

```
create table Payments(payment_id int primary key auto_increment,
student_id int not null,
foreign key(student_id) references Students(student_id),
amount int not null check (amount >100),
payment_date date not null);
```

	Field	Type	Null	Key	Default	Extra
▶	payment_id	int	NO	PRI	NULL	auto_increment
	student_id	int	NO	MUL	NULL	
	amount	int	NO		NULL	
	payment_date	date	NO		NULL	

3. Create an ERD (Entity Relationship Diagram) for the database.



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Refer Task 2

5. Insert at least 10 sample records into each of the following tables.
i. Students

```
insert into Students(first_name,last_name,date_of_birth,email,phone_number)
values('David','Miller','1996-06-23','david.miller@gmail.com',4585167612),
('Lina','Smith','1996-01-08','lina.smith@gmail.com',3256748934),
```

('Lana','Rey','1995-04-20','lana_rey@gmail.com',91546386451),
('Serena','Joe','1997-03-10','ser.joe@gmail.com',9563215410),
('Hailey','Baldwin','1995-06-30','call_me_hailey@gmail.com',8315426470),
('Santa','Clause','1997-12-25','santa.clause.town@gmail.com',763521944),
('Emily','Wilson','1996-08-22','emily@gamil.com',8367289131),
('Frank','Moore','1995-09-01','be_frank@gmail.com',91457535416),
('Charlie','Davis','1996-10-06','davis_day@gamil.com',7361470377),
('Hank','Andreson','1996-11-27','hank_and@gmail.com',893561531);

student_id	first_name	last_name	date_of_birth	email	phone_number
1	David	Miller	1996-06-23	david.miller@gmail.com	4585167612
2	Lina	Smith	1996-01-08	lina.smith@gmail.com	3256748934
3	Lana	Rey	1995-04-20	lana_rey@gmail.com	91546386451
4	Serena	Joe	1997-03-10	ser.joe@gmail.com	9563215410
5	Hailey	Baldwin	1995-06-30	call_me_hailey@gmail.com	8315426470
6	Santa	Clause	1997-12-25	santa.clause.town@gmail.com	763521944
7	Emily	Wilson	1996-08-22	emily@gamil.com	8367289131
8	Frank	Moore	1995-09-01	be_frank@gmail.com	91457535416
9	Charlie	Davis	1996-10-06	davis_day@gamil.com	7361470377
10	Hank	Andreson	1996-11-27	hank_and@gmail.com	893561531
NULL	NULL	NULL	NULL	NULL	NULL

ii. Courses

insert into Courses(course_name,credits,teacher_id) values
('Maths',5,1),
('Physics',3,2),
('Computer',4,3),
('Chemistry',3,4),
('Human Resource Management',3,5),
('Software Engineering',4,6),
('English',3,7),
('Game programming',4,8),
('Psychology',3,9),
('History',2,10);

course_id	course_name	credits	teacher_id
1	Maths	5	1
2	Physics	3	2
3	Computer	4	3
4	Chemisty	3	4
5	Human Resource Management	3	5
6	Software Engineering	4	6
7	English	3	7
8	Game programming	4	8
9	Psychology	3	9
10	History	2	10
NULL	NULL	NULL	NULL

iii. Enrollments

```
insert into Enrollments(student_id,course_id,enrollment_date)
values
(1,1,'2000-01-15'),
(2,2,'2000-01-16'),
(3,3,'2000-01-17'),
(4,4,'2000-01-18'),
(5,5,'2000-01-19'),
(6,6,'2000-01-20'),
(7,7,'2000-01-21'),
(8,8,'2000-01-22'),
(9,9,'2000-01-23'),
(10,10,'2000-01-24');
```

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2000-01-15
2	2	2	2000-01-16
3	3	3	2000-01-17
4	4	4	2000-01-18
5	5	5	2000-01-19
6	6	6	2000-01-20
7	7	7	2000-01-21
8	8	8	2000-01-22
9	9	9	2000-01-23
10	10	10	2000-01-24
NULL	NULL	NULL	NULL

iv. Teacher

```
insert into Teacher(first_name,last_name,email) values
('Dr.Robert','White','robert_white@gmail.com'),
('Dr.Remus','Lupin','lupin_moon@gmail.com'),
('Dr.Barbara','Sprout','barbara@gmail.com'),
('Dr.Michael','Sandel','michael_sandel@gmail.com'),
('Dr.Steven','Pinker','pinker@gmail.com'),
('Dr.Joanna','Aizenberg','aizenberg_jo@gmail.com'),
('Dr.Henry','Gates','henrygates@gmail.com'),
('Dr.Jennifer','Doudna','jenny_dou@gmail.com'),
('Dr.Catherine','Ramirez','catherine@gmail.com'),
('Dr.Olivia','Graeva','olivia_g@gmail.com');
```

teacher_id	first_name	last_name	email
1	Dr.Robert	White	robert_white@gmail.com
2	Dr.Remus	Lupin	lupin_moon@gmail.com
3	Dr.Barbara	Sprout	barbara@gmail.com
4	Dr.Michael	Sandel	michael_sandel@gmail.com
5	Dr.Steven	Pinker	pinker@gmail.com
6	Dr.Joanna	Aizenberg	aizenberg_jo@gmail.com
7	Dr.Henry	Gates	henrygates@gmail.com
8	Dr.Jennifer	Doudna	jenny_dou@gmail.com
9	Dr.Catherine	Ramirez	catherine@gmail.com
10	Dr.Olivia	Graeva	olivia_g@gmail.com
NULL	NULL	NULL	NULL

v. Payments

```
insert into Payments(student_id,amount,payment_date)
values
(1,600,'2000-01-02'),
(2,400,'2000-01-03'),
(3,500,'2000-01-04'),
(4,400,'2000-01-05'),
(5,300,'2000-01-06'),
(6,500,'2000-01-07'),
(7,300,'2000-01-08'),
(8,600,'2000-01-09'),
(9,300,'2000-01-10'),
(10,200,'2000-01-11');
```

payment_id	student_id	amount	payment_date
1	1	600	2000-01-02
2	2	400	2000-01-03
3	3	500	2000-01-04
4	4	400	2000-01-05
5	5	300	2000-01-06
6	6	500	2000-01-07
7	7	300	2000-01-08
8	8	600	2000-01-09
9	9	300	2000-01-10
10	10	200	2000-01-11
NULL	NULL	NULL	NULL

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@gmail.com",1234567890);
select * from Students where student_id = 11;
```

student_id	first_name	last_name	date_of_birth	email	phone_number
11	John	Doe	1995-08-15	john.doe@gmail.com	1234567890
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 (5,3,"2000-02-02");
select * from Enrollments where student_id = 5;
```

	enrollment_id	student_id	course_id	enrollment_date
▶	5	5	5	2000-01-19
	11	5	3	2000-02-02
*	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 = "lupin_full_moon@gmail.com" where teacher_id = 2;
select * from Teacher where teacher_id = 2;
```

	teacher_id	first_name	last_name	email
▶	2	Dr.Remus	Lupin	lupin_full_moon@gmail.com
*	NULL	NULL	NULL	NULL

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 = 3;
select * from Enrollments;
```

	enrollment_id	student_id	course_id	enrollment_date
▶	1	1	1	2000-01-15
	2	2	2	2000-01-16
	4	4	4	2000-01-18
	5	5	5	2000-01-19
	6	6	6	2000-01-20
	7	7	7	2000-01-21
	8	8	8	2000-01-22
	9	9	9	2000-01-23
	10	10	10	2000-01-24
	11	5	3	2000-02-02
*	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 = 3 where course_id = 5;
select * from Courses where course_id = 5;
```

course_id	course_name	credits	teacher_id
5	Human Resource Management	3	3
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 Students where student_id = 5;
```

Error:

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 = 250 where payment_id = 10;
select * from payments;
```

payment_id	student_id	amount	payment_date
1	1	600	2000-01-02
2	2	400	2000-01-03
3	3	500	2000-01-04
4	4	400	2000-01-05
5	5	300	2000-01-06
6	6	500	2000-01-07
7	7	300	2000-01-08
8	8	600	2000-01-09
9	9	300	2000-01-10
10	10	250	2000-01-11
NULL	NULL	NULL	NULL

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,s.first_name,s.last_name, sum(p.amount) from Students s join payments p on
s.student_id = p.student_id
where s.student_id = 1 group by s.first_name;
```

student_id	first_name	last_name	sum(p.amount)
1	David	Miller	600

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_id, c.course_name, count(e.student_id) as count from Courses c join Enrollments e on
c.course_id = e.course_id
group by c.course_id;
```

course_id	course_name	count
1	Maths	1
2	Physics	1
4	Chemisty	1
5	Human Resource Management	1
6	Software Engineering	1
7	English	1
8	Game programming	1
9	Psychology	1
10	History	1
3	Computer	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.student_id, s.first_name, s.last_name from Students s left join enrollments e on s.student_id =
e.student_id
where e.enrollment_id is null group by s.student_id;
```

student_id	first_name	last_name
3	Lana	Rey
11	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
c.course_id = e.course_id;
```

first_name	last_name	course_name
David	Miller	Maths
Lina	Smith	Physics
Serena	Joe	Chemisty
Hailey	Baldwin	Human Resource Management
Santa	Clause	Software Engineering
Emily	Wilson	English
Frank	Moore	Game programming
Charlie	Davis	Psychology
Hank	Andreson	History
Hailey	Baldwin	Computer

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 order by t.teacher_id;
```

first_name	last_name	course_name
Dr.Robert	White	Maths
Dr.Remus	Lupin	Physics
Dr.Barbara	Sprout	Computer
Dr.Barbara	Sprout	Human Resource Management
Dr.Michael	Sandel	Chemisty
Dr.Joanna	Aizenberg	Software Engineering
Dr.Henry	Gates	English
Dr.Jennifer	Doudna	Game programming
Dr.Catherine	Ramirez	Psychology
Dr.Olivia	Graeva	History

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 join Courses c on
c.course_id = e.course_id where c.course_name = "Computer";
```

first_name	last_name	enrollment_date
Hailey	Baldwin	2000-02-02

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.student_id, 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;
```

	student_id	first_name	last_name
▶	11	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 c.course_name from Courses c left join Enrollments e on c.course_id = e.course_id where
e.enrollment_id is null
order by c.course_id;
```

	course_name

→ Empty because there is no courses without enrollment

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 e1.student_id, e1.course_id,concat(s.first_name," ",s.last_name) as name from Enrollments e1
join Enrollments e2
on e1.student_id = e2.student_id and e1.course_id <> e2.course_id join Students s
on e1.student_id = s.student_id;
```

	student_id	course_id	name
▶	5	5	Hailey Baldwin
	5	3	Hailey Baldwin

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, t.last_name from Teacher t left join Courses c on t.teacher_id =
c.teacher_id
where c.course_id is null order by t.teacher_id;
```

	teacher_id	first_name	last_name
▶	5	Dr.Steven	Pinker

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(count_val) from (select course_id, count(student_id) as count_val from enrollments group by course_id) as total_enrollment;
```

	avg(count_val)
▶	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 concat(s.first_name," ",s.last_name) as name, p.amount from Students s join Payments p on p.student_id = s.student_id where
p.amount = (select max(amount) from Payments);
```

	name	amount
▶	David Miller	600
	Frank Moore	600

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, count(e.student_id) 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(count_of_std) from(select course_id, count(student_id) as count_of_std from enrollments group by course_id) as max_val));
```

	course_id	course_name	count(e.student_id)
▶	1	Maths	1
	2	Physics	1
	4	Chemisty	1
	5	Human Resource Management	1
	6	Software Engineering	1
	7	English	1
	8	Game programming	1
	9	Psychology	1
	10	History	1
	3	Computer	1

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

```
select concat(t.first_name," ",t.last_name) as teacher_name, (select sum(p.amount) from payments p join enrollments e on p.student_id = e.student_id join courses c on e.course_id = c.course_id where
```

```
c.teacher_id = t.teacher_id) as total_amt_of_each_course from Teacher t;
```

teacher_name	total_amt_of_each_course
Dr.Robert White	600
Dr.Remus Lupin	400
Dr.Barbara Sprout	600
Dr.Michael Sandel	400
Dr.Steven Pinker	NULL
Dr.Joanna Aizenberg	500
Dr.Henry Gates	300
Dr.Jennifer Doudna	600
Dr.Catherine Ramirez	300
Dr.Olivia Graeva	250

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_with_all_enrollment from Students s join
enrollments e on s.student_id = e.student_id
group by s.student_id having count(e.course_id) = (select count(*) from Courses);
```

student_name_with_all_enrollment

→ No students have enrolled all 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(first_name,last_name) as teacher_name from teacher where teacher_id not in (select
teacher_id from Courses);
```

teacher_name
Dr.StevenPinker

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(current_age_students) from (select year(curdate()) - year(date_of_birth) as
current_age_students from Students) as age;
```

avg(current_age_students)
29.1818

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

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

course_name

→ All courses are enrolled

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) as amount_of_each_student from Payments p where p.student_id = s.student_id)
as total_amt 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	total_amt
David	Miller	Maths	600
Lina	Smith	Physics	400
Serena	Joe	Chemisty	400
Hailey	Baldwin	Human Resource Management	300
Santa	Clause	Software Engineering	500
Emily	Wilson	English	300
Frank	Moore	Game programming	600
Charlie	Davis	Psychology	300
Hank	Andreson	History	250
Hailey	Baldwin	Computer	300

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 first_name, last_name from Students where student_id in
(select student_id from payments group by student_id having count(amount) > 1);
```

first_name	last_name

→ Students have not paid more than once.

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.first_name, s.last_name,
(select sum(p.amount) as total_amt from Payments p where p.student_id = s.student_id)
from Students s group by s.student_id;
```

	first_name	last_name	(select sum(p.amount) as total_amt from Payments p where p.student_id = s.student_id)
▶	David	Miller	600
	Lina	Smith	400
	Lana	Rey	500
	Serena	Joe	400
	Hailey	Baldwin	300
	Santa	Clause	500
	Emily	Wilson	300
	Frank	Moore	600
	Charlie	Davis	300
	Hank	Andreson	250
	John	Doe	NULL

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, (select count(e.student_id) from enrollments e where e.course_id = c.course_id)
as num_students
from courses c group by c.course_id;
```

	course_name	num_students
▶	Maths	1
	Physics	1
	Computer	1
	Chemisty	1
	Human Resource Management	1
	Software Engineering	1
	English	1
	Game programming	1
	Psychology	1
	History	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 s.student_id = p.student_id) as
avg_amt from Students s group by s.student_id;
```

first_name	last_name	avg_amt
David	Miller	600.0000
Lina	Smith	400.0000
Lana	Rey	500.0000
Serena	Joe	400.0000
Hailey	Baldwin	300.0000
Santa	Clause	500.0000
Emily	Wilson	300.0000
Frank	Moore	600.0000
Charlie	Davis	300.0000
Hank	Andreson	250.0000
John	Doe	NULL
