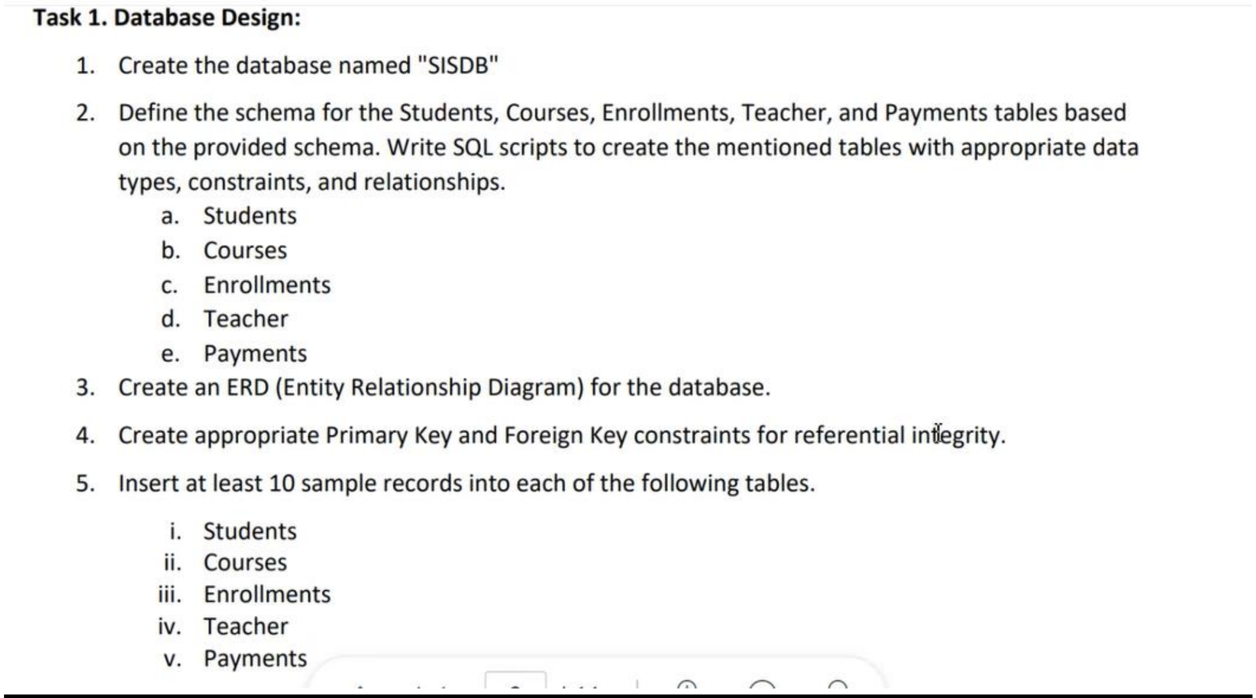
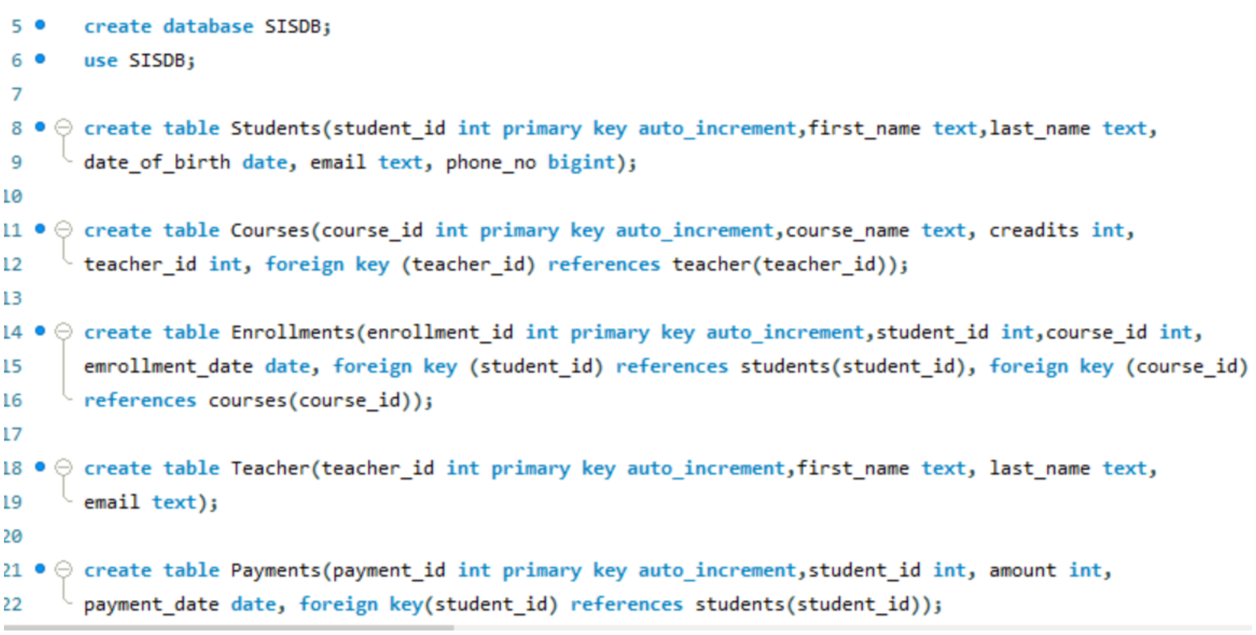
ASSIGNMENT-2(HEXA)

Task-1: By:Satyendra Singh Rathore

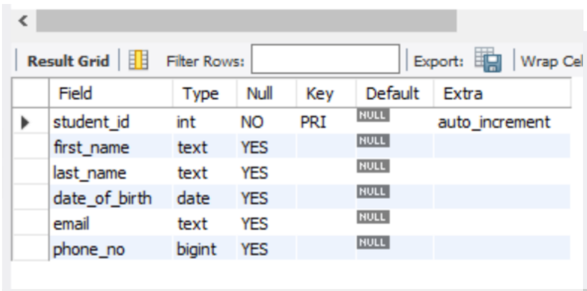


Task 1. Database Design:

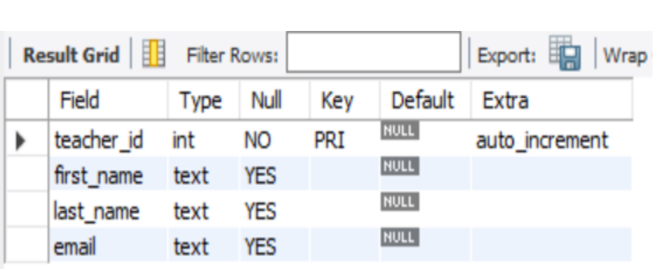
1. Create the database named "SISDB"
2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
    a. Students
    b. Courses
    c. Enrollments
    d. Teacher
    e. Payments
3. Create an ERD (Entity Relationship Diagram) for the database.
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.
5. Insert at least 10 sample records into each of the following tables.
    i. Students
    ii. Courses
    iii. Enrollments
    iv. Teacher
    v. Payments



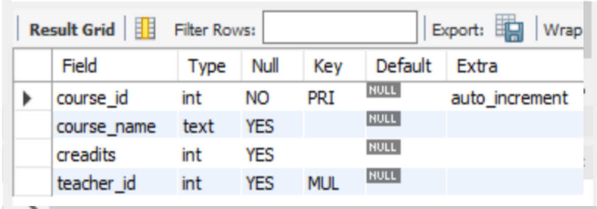
```sql
create database SISDB;
use SISDB;

create table Students(student_id int primary key auto_increment,first_name text,last_name text,
date_of_birth date, email text, phone_no bigint);

create table Courses(course_id int primary key auto_increment,course_name text, creadits int,
teacher_id int, foreign key (teacher_id) references teacher(teacher_id));

create table Enrollments(enrollment_id int primary key auto_increment,student_id int,course_id int,
emrollment_date date, foreign key (student_id) references students(student_id), foreign key (course_id)
references courses(course_id));

create table Teacher(teacher_id int primary key auto_increment,first_name text, last_name text,
email text);

create table Payments(payment_id int primary key auto_increment,student_id int, amount int,
payment_date date, foreign key(student_id) references students(student_id));
```



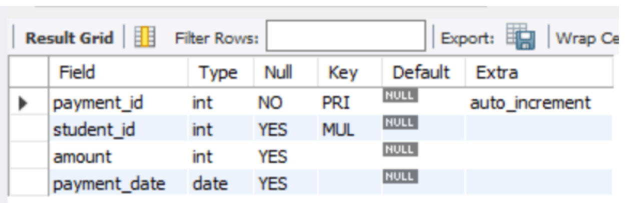
Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	auto_increment
first_name	text	YES		NULL	
last_name	text	YES		NULL	
date_of_birth	date	YES		NULL	
email	text	YES		NULL	
phone_no	bigint	YES		NULL	



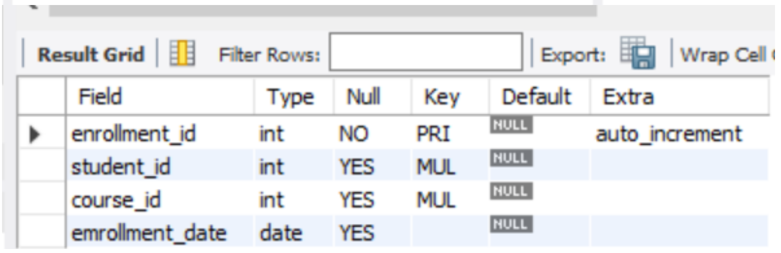
Field	Type	Null	Key	Default	Extra
teacher_id	int	NO	PRI	NULL	auto_increment
first_name	text	YES		NULL	
last_name	text	YES		NULL	
email	text	YES		NULL	



Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI	NULL	auto_increment
course_name	text	YES		NULL	
creadits	int	YES		NULL	
teacher_id	int	YES	MUL	NULL	

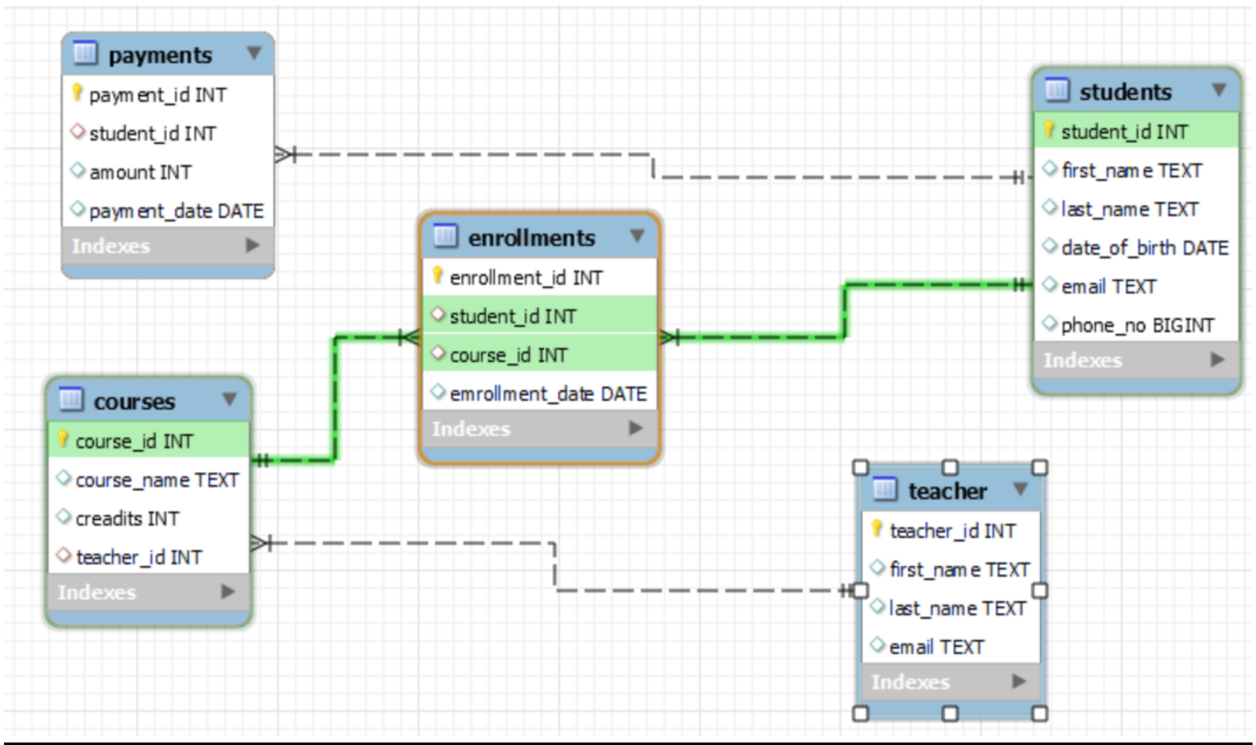


Field	Type	Null	Key	Default	Extra
payment_id	int	NO	PRI	NULL	auto_increment
student_id	int	YES	MUL	NULL	
amount	int	YES		NULL	
payment_date	date	YES		NULL	

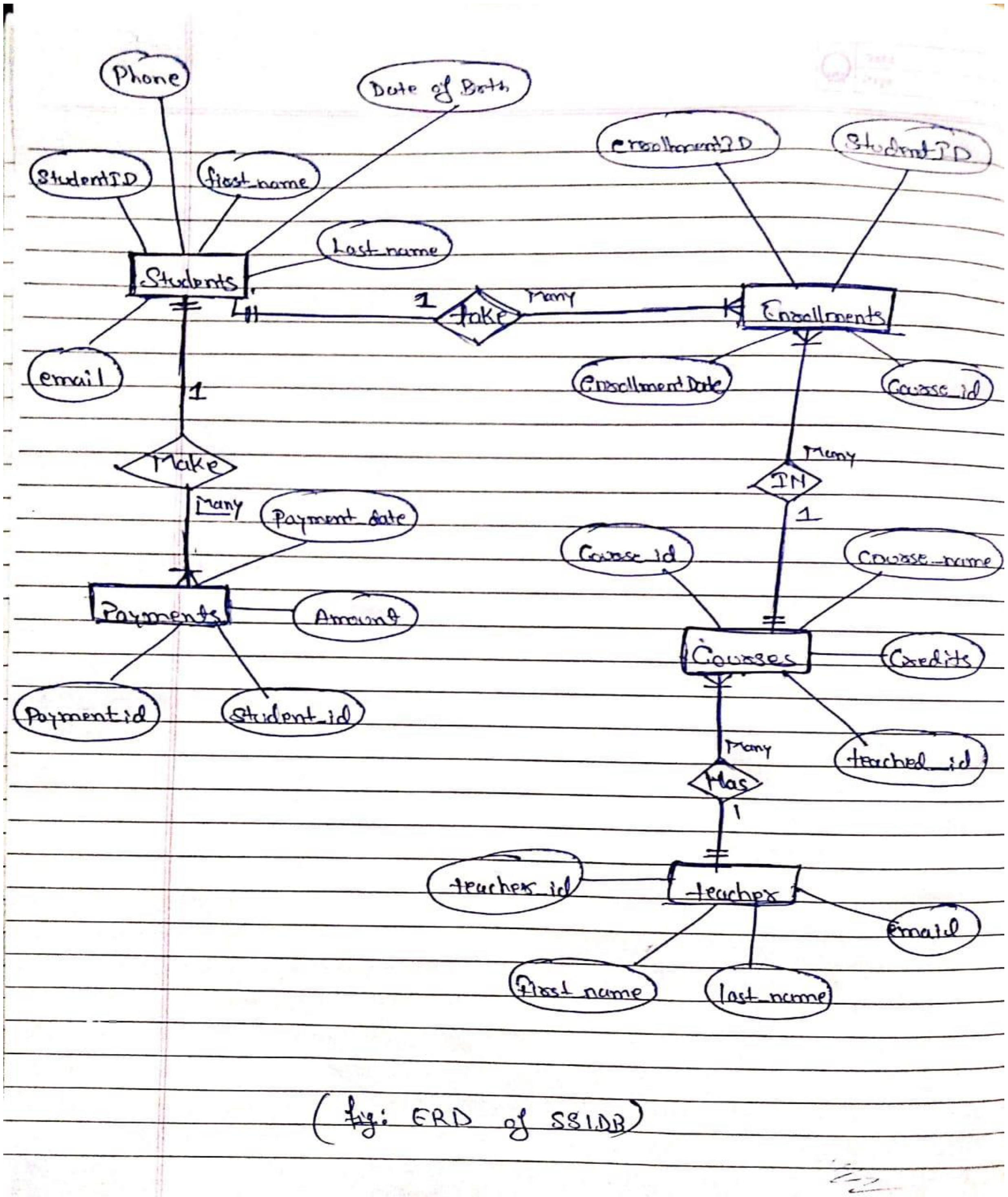


Field	Type	Null	Key	Default	Extra
enrollment_id	int	NO	PRI	NULL	auto_increment
student_id	int	YES	MUL	NULL	
course_id	int	YES	MUL	NULL	
emrollment_date	date	YES		NULL	

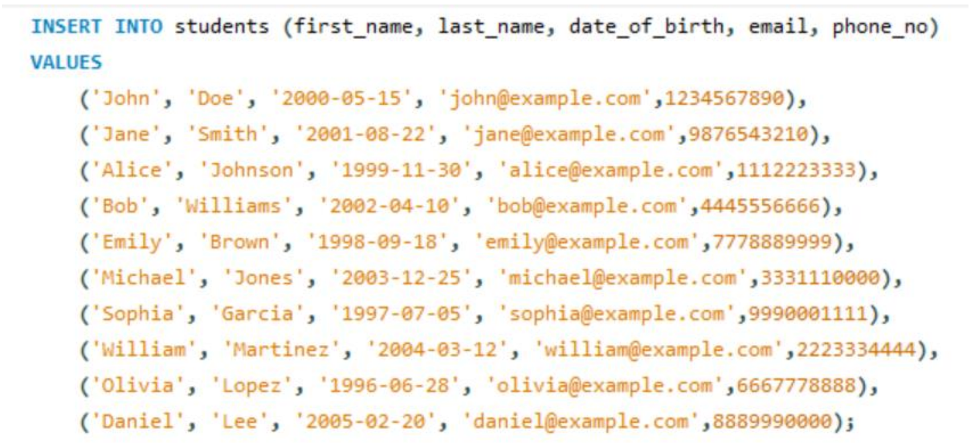
Relationship Model:



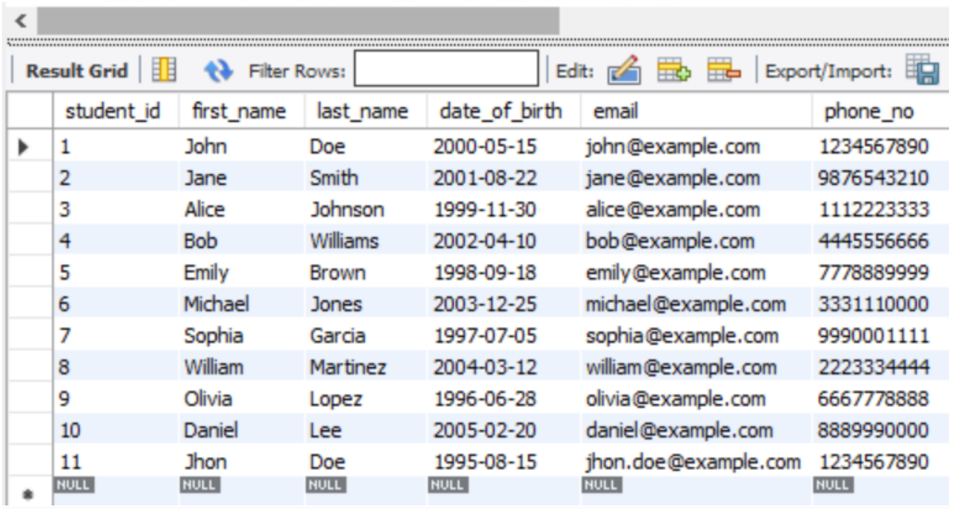
ER Diagram:



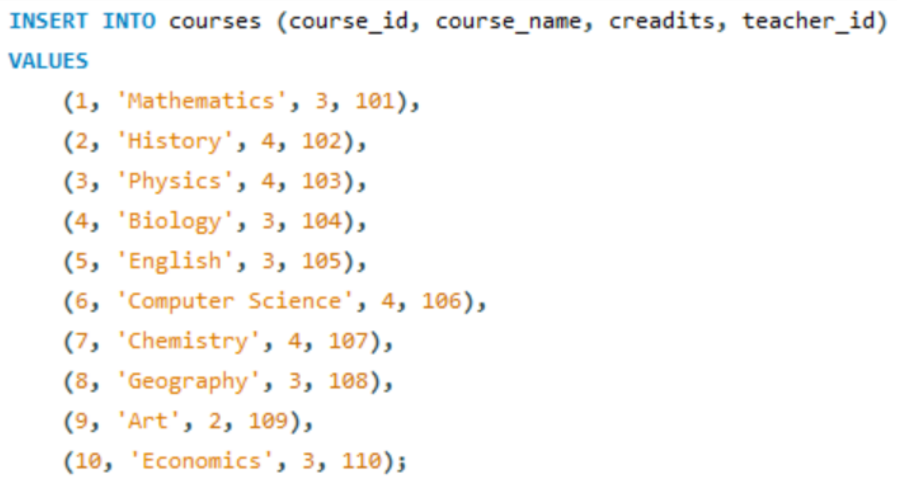
(fig: ERD of SSIDB)



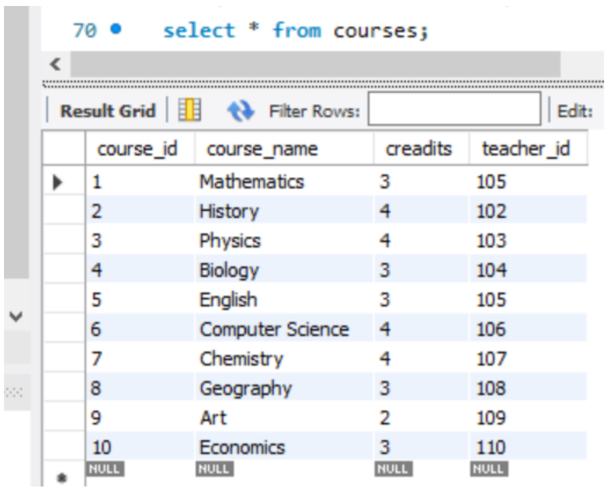
```sql
INSERT INTO students (first_name, last_name, date_of_birth, email, phone_no)
VALUES
    ('John', 'Doe', '2000-05-15', 'john@example.com',1234567890),
    ('Jane', 'Smith', '2001-08-22', 'jane@example.com',9876543210),
    ('Alice', 'Johnson', '1999-11-30', 'alice@example.com',1112223333),
    ('Bob', 'Williams', '2002-04-10', 'bob@example.com',4445556666),
    ('Emily', 'Brown', '1998-09-18', 'emily@example.com',7778889999),
    ('Michael', 'Jones', '2003-12-25', 'michael@example.com',3331110000),
    ('Sophia', 'Garcia', '1997-07-05', 'sophia@example.com',9990001111),
    ('William', 'Martinez', '2004-03-12', 'william@example.com',2223334444),
    ('Olivia', 'Lopez', '1996-06-28', 'olivia@example.com',6667778888),
    ('Daniel', 'Lee', '2005-02-20', 'daniel@example.com',8889990000);
```



student_id	first_name	last_name	date_of_birth	email	phone_no
1	John	Doe	2000-05-15	john@example.com	1234567890
2	Jane	Smith	2001-08-22	jane@example.com	9876543210
3	Alice	Johnson	1999-11-30	alice@example.com	1112223333
4	Bob	Williams	2002-04-10	bob@example.com	4445556666
5	Emily	Brown	1998-09-18	emily@example.com	7778889999
6	Michael	Jones	2003-12-25	michael@example.com	3331110000
7	Sophia	Garcia	1997-07-05	sophia@example.com	9990001111
8	William	Martinez	2004-03-12	william@example.com	2223334444
9	Olivia	Lopez	1996-06-28	olivia@example.com	6667778888
10	Daniel	Lee	2005-02-20	daniel@example.com	8889990000
11	Jhon	Doe	1995-08-15	jhon.doe@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

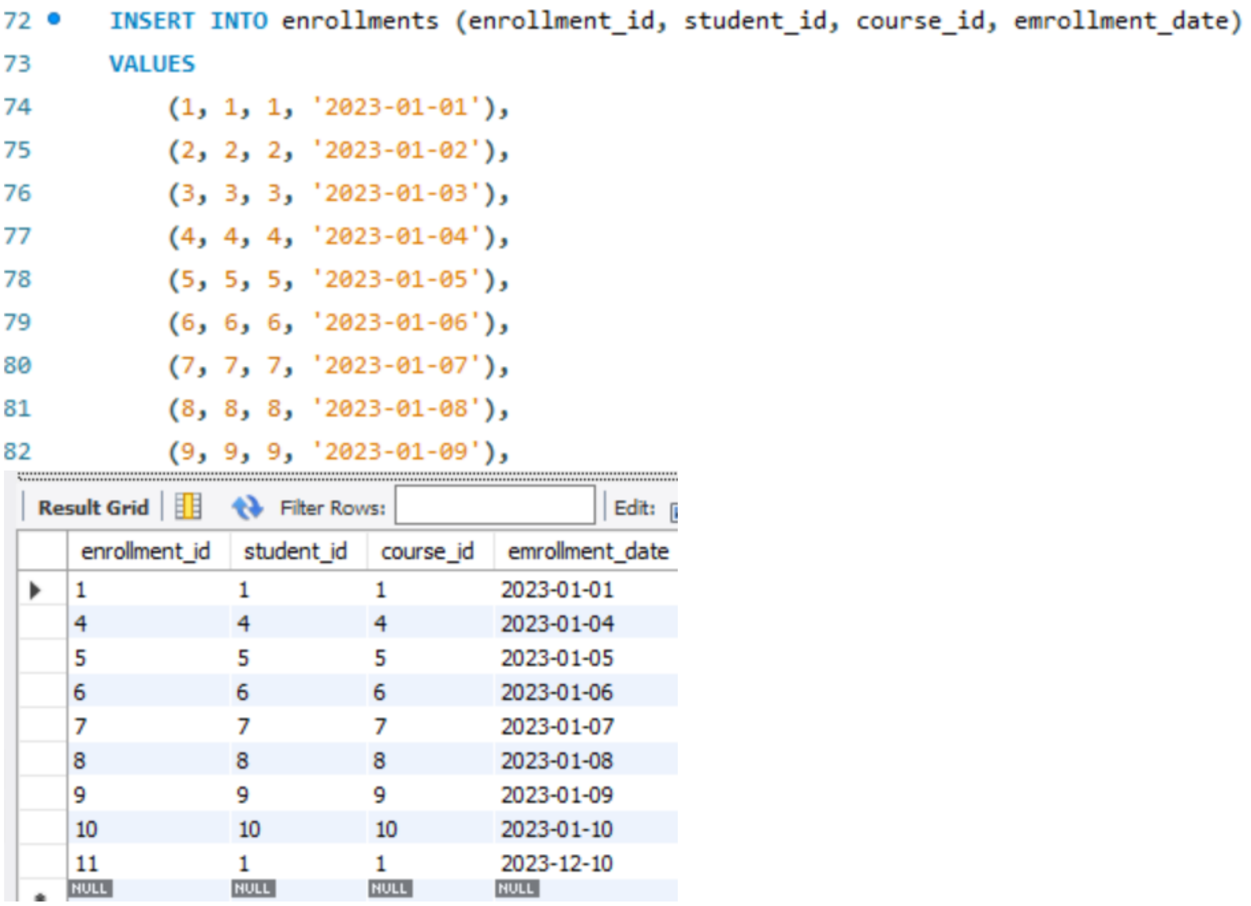


```sql
INSERT INTO courses (course_id, course_name, creadits, teacher_id)
VALUES
    (1, 'Mathematics', 3, 101),
    (2, 'History', 4, 102),
    (3, 'Physics', 4, 103),
    (4, 'Biology', 3, 104),
    (5, 'English', 3, 105),
    (6, 'Computer Science', 4, 106),
    (7, 'Chemistry', 4, 107),
    (8, 'Geography', 3, 108),
    (9, 'Art', 2, 109),
    (10, 'Economics', 3, 110);
```



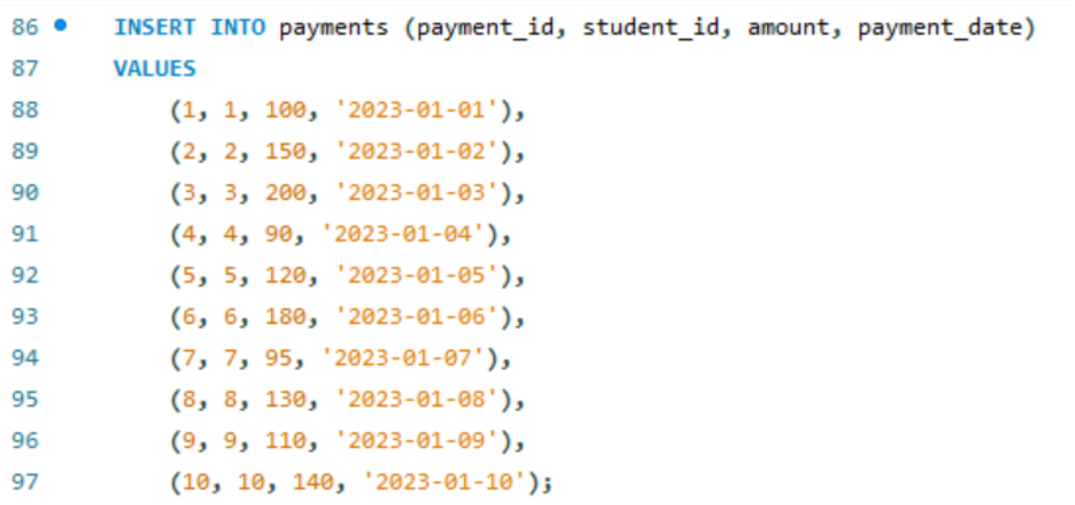
```sql
select * from courses;
```

course_id	course_name	creadits	teacher_id
1	Mathematics	3	105
2	History	4	102
3	Physics	4	103
4	Biology	3	104
5	English	3	105
6	Computer Science	4	106
7	Chemistry	4	107
8	Geography	3	108
9	Art	2	109
10	Economics	3	110
NULL	NULL	NULL	NULL

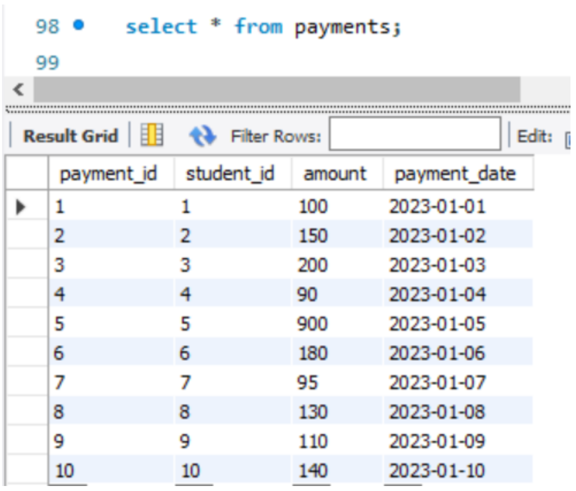


```sql
INSERT INTO enrollments (enrollment_id, student_id, course_id, emrollment_date)
VALUES
    (1, 1, 1, '2023-01-01'),
    (2, 2, 2, '2023-01-02'),
    (3, 3, 3, '2023-01-03'),
    (4, 4, 4, '2023-01-04'),
    (5, 5, 5, '2023-01-05'),
    (6, 6, 6, '2023-01-06'),
    (7, 7, 7, '2023-01-07'),
    (8, 8, 8, '2023-01-08'),
    (9, 9, 9, '2023-01-09'),
```

enrollment_id	student_id	course_id	emrollment_date
1	1	1	2023-01-01
4	4	4	2023-01-04
5	5	5	2023-01-05
6	6	6	2023-01-06
7	7	7	2023-01-07
8	8	8	2023-01-08
9	9	9	2023-01-09
10	10	10	2023-01-10
11	1	1	2023-12-10
NULL	NULL	NULL	NULL



```sql
INSERT INTO payments (payment_id, student_id, amount, payment_date)
VALUES
    (1, 1, 100, '2023-01-01'),
    (2, 2, 150, '2023-01-02'),
    (3, 3, 200, '2023-01-03'),
    (4, 4, 90, '2023-01-04'),
    (5, 5, 120, '2023-01-05'),
    (6, 6, 180, '2023-01-06'),
    (7, 7, 95, '2023-01-07'),
    (8, 8, 130, '2023-01-08'),
    (9, 9, 110, '2023-01-09'),
    (10, 10, 140, '2023-01-10');
```

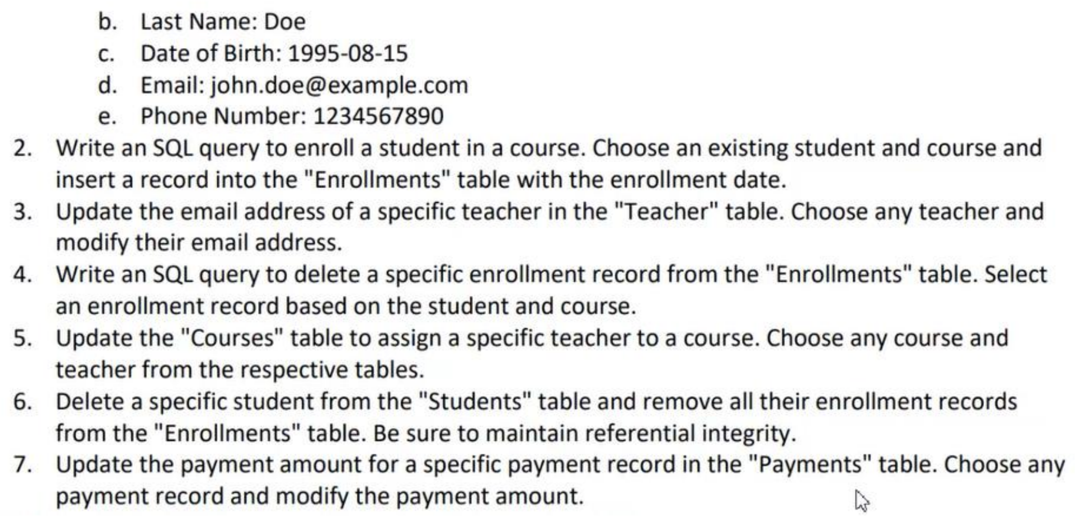


```sql
select * from payments;
```

Result Grid | Filter Rows: | Edit:

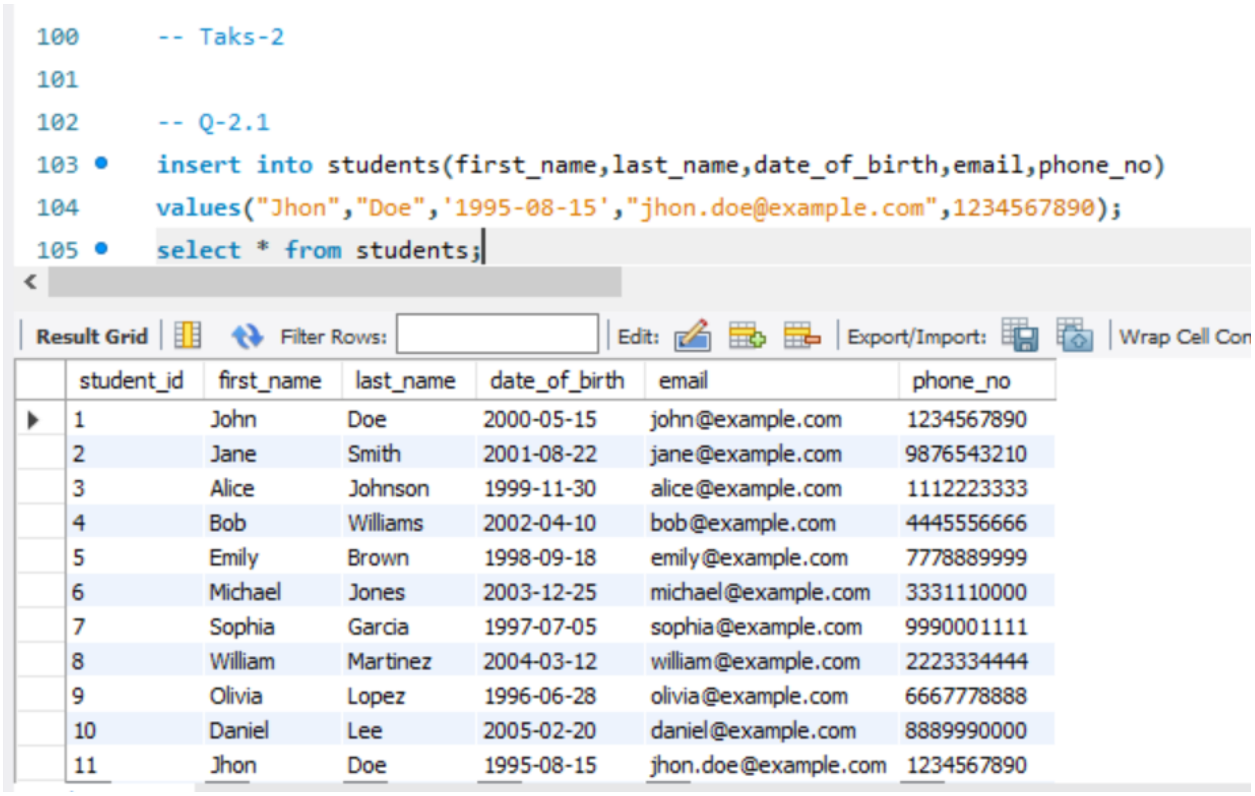
	payment_id	student_id	amount	payment_date
▶	1	1	100	2023-01-01
	2	2	150	2023-01-02
	3	3	200	2023-01-03
	4	4	90	2023-01-04
	5	5	900	2023-01-05
	6	6	180	2023-01-06
	7	7	95	2023-01-07
	8	8	130	2023-01-08
	9	9	110	2023-01-09
	10	10	140	2023-01-10

<u>Task-2:</u>



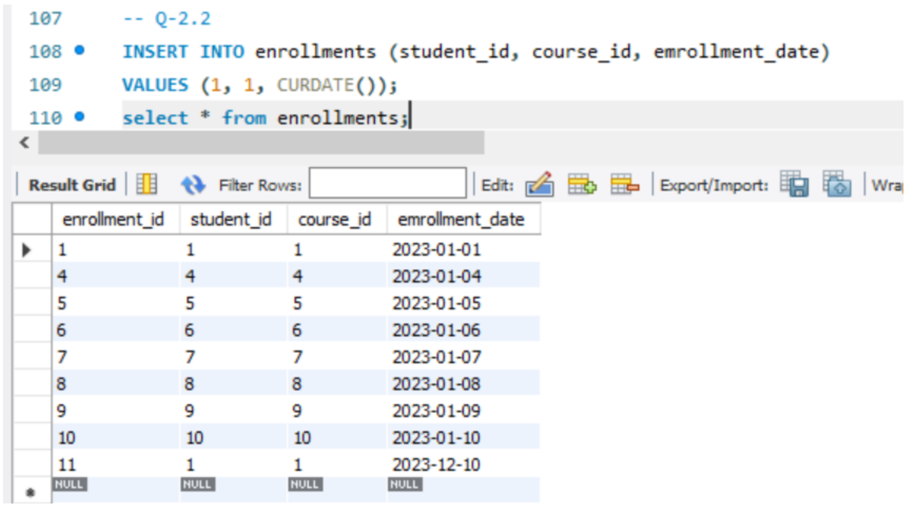
b. Last Name: Doe
c. Date of Birth: 1995-08-15
d. Email: john.doe@example.com
e. Phone Number: 1234567890

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.
3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.
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.
5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.
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.
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



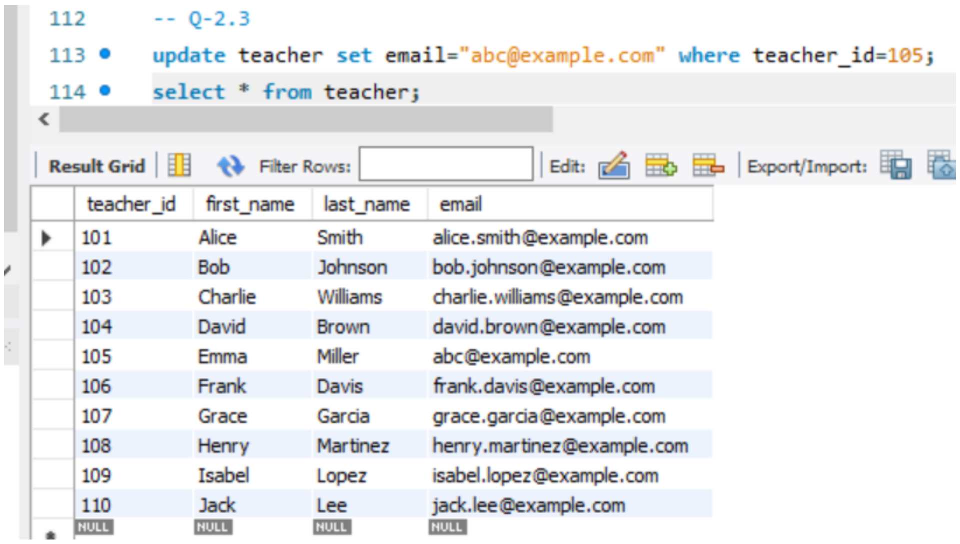
```
-- Taks-2

-- Q-2.1
insert into students(first_name,last_name,date_of_birth,email,phone_no)
values("Jhon","Doe",'1995-08-15',"jhon.doe@example.com",1234567890);
select * from students;
```

student_id	first_name	last_name	date_of_birth	email	phone_no
1	John	Doe	2000-05-15	john@example.com	1234567890
2	Jane	Smith	2001-08-22	jane@example.com	9876543210
3	Alice	Johnson	1999-11-30	alice@example.com	1112223333
4	Bob	Williams	2002-04-10	bob@example.com	4445556666
5	Emily	Brown	1998-09-18	emily@example.com	7778889999
6	Michael	Jones	2003-12-25	michael@example.com	3331110000
7	Sophia	Garcia	1997-07-05	sophia@example.com	9990001111
8	William	Martinez	2004-03-12	william@example.com	2223334444
9	Olivia	Lopez	1996-06-28	olivia@example.com	6667778888
10	Daniel	Lee	2005-02-20	daniel@example.com	8889990000
11	Jhon	Doe	1995-08-15	jhon.doe@example.com	1234567890



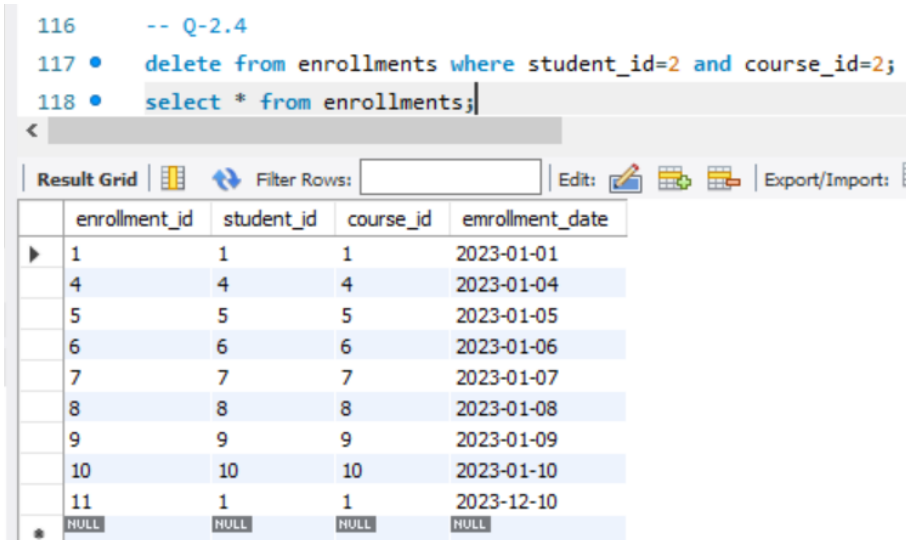
```
-- Q-2.2
INSERT INTO enrollments (student_id, course_id, emrollment_date)
VALUES (1, 1, CURDATE());
select * from enrollments;
```

enrollment_id	student_id	course_id	emrollment_date
1	1	1	2023-01-01
4	4	4	2023-01-04
5	5	5	2023-01-05
6	6	6	2023-01-06
7	7	7	2023-01-07
8	8	8	2023-01-08
9	9	9	2023-01-09
10	10	10	2023-01-10
11	1	1	2023-12-10
NULL	NULL	NULL	NULL



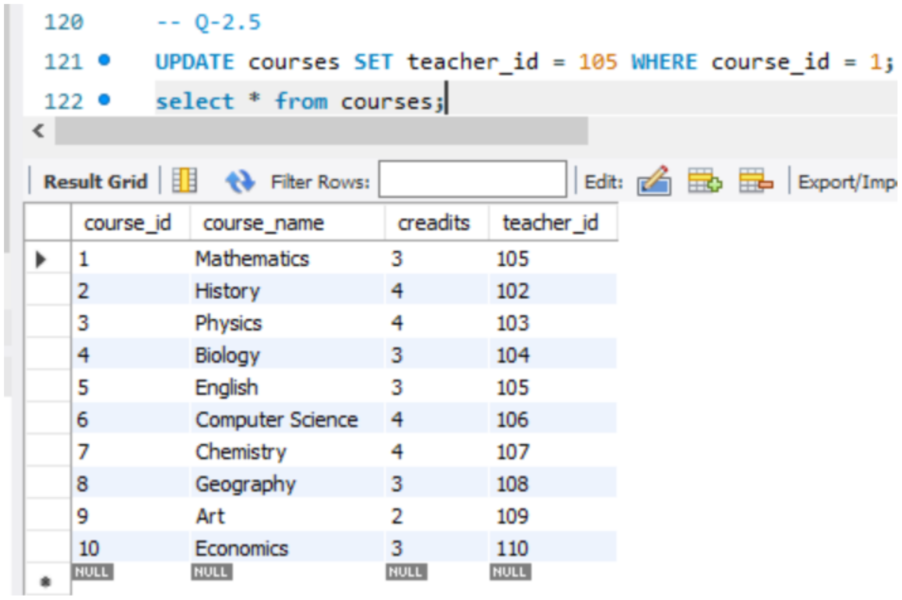
```sql
-- Q-2.3
update teacher set email="abc@example.com" where teacher_id=105;
select * from teacher;
```

teacher_id	first_name	last_name	email
101	Alice	Smith	alice.smith@example.com
102	Bob	Johnson	bob.johnson@example.com
103	Charlie	Williams	charlie.williams@example.com
104	David	Brown	david.brown@example.com
105	Emma	Miller	abc@example.com
106	Frank	Davis	frank.davis@example.com
107	Grace	Garcia	grace.garcia@example.com
108	Henry	Martinez	henry.martinez@example.com
109	Isabel	Lopez	isabel.lopez@example.com
110	Jack	Lee	jack.lee@example.com
NULL	NULL	NULL	NULL



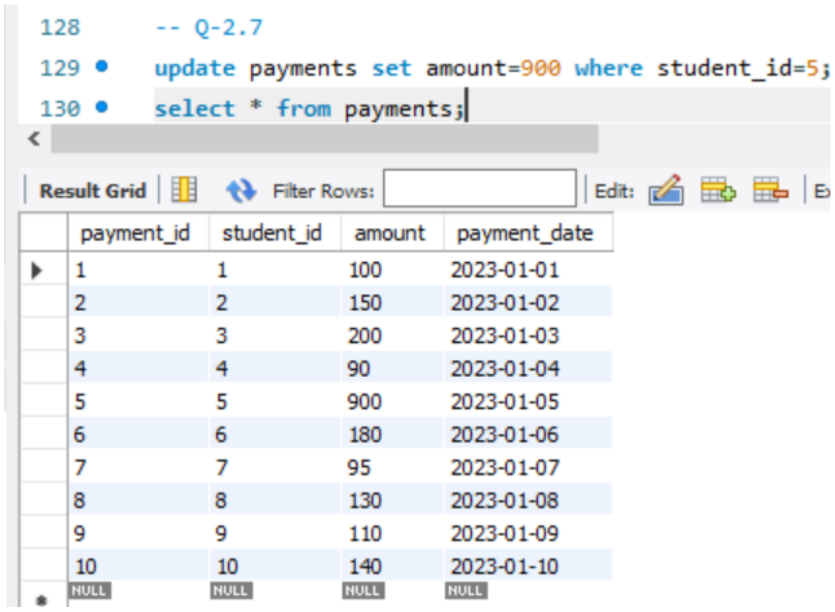
```sql
-- Q-2.4
delete from enrollments where student_id=2 and course_id=2;
select * from enrollments;
```

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2023-01-01
4	4	4	2023-01-04
5	5	5	2023-01-05
6	6	6	2023-01-06
7	7	7	2023-01-07
8	8	8	2023-01-08
9	9	9	2023-01-09
10	10	10	2023-01-10
11	1	1	2023-12-10
NULL	NULL	NULL	NULL



```sql
-- Q-2.5
UPDATE courses SET teacher_id = 105 WHERE course_id = 1;
select * from courses;
```

course_id	course_name	credits	teacher_id
1	Mathematics	3	105
2	History	4	102
3	Physics	4	103
4	Biology	3	104
5	English	3	105
6	Computer Science	4	106
7	Chemistry	4	107
8	Geography	3	108
9	Art	2	109
10	Economics	3	110
NULL	NULL	NULL	NULL



```
-- Q-2.7
update payments set amount=900 where student_id=5;
select * from payments;
```

Result Grid | Filter Rows: | Edit:

payment_id	student_id	amount	payment_date
1	1	100	2023-01-01
2	2	150	2023-01-02
3	3	200	2023-01-03
4	4	90	2023-01-04
5	5	900	2023-01-05
6	6	180	2023-01-06
7	7	95	2023-01-07
8	8	130	2023-01-08
9	9	110	2023-01-09
10	10	140	2023-01-10
NULL	NULL	NULL	NULL

Task-3:

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.

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.

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.

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.

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.

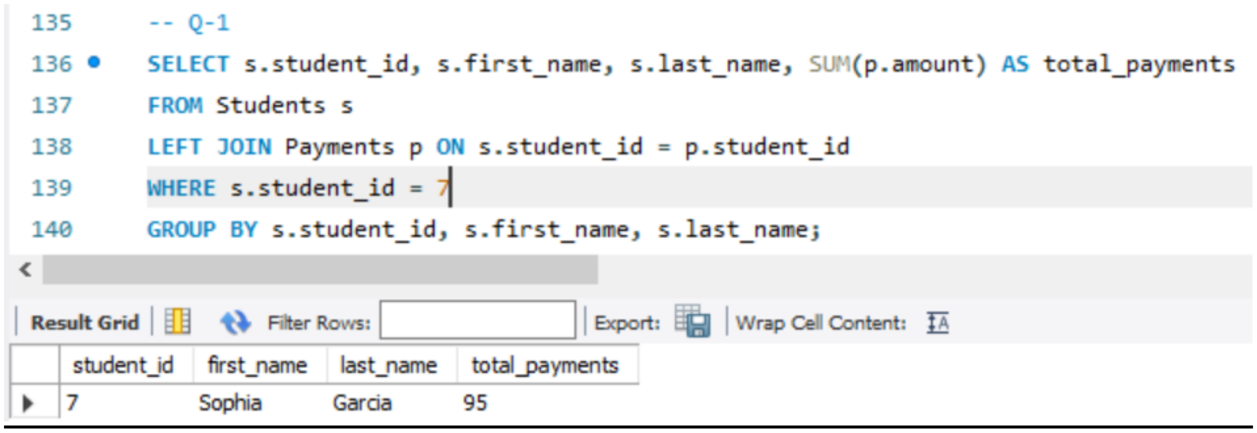
6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

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.

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.

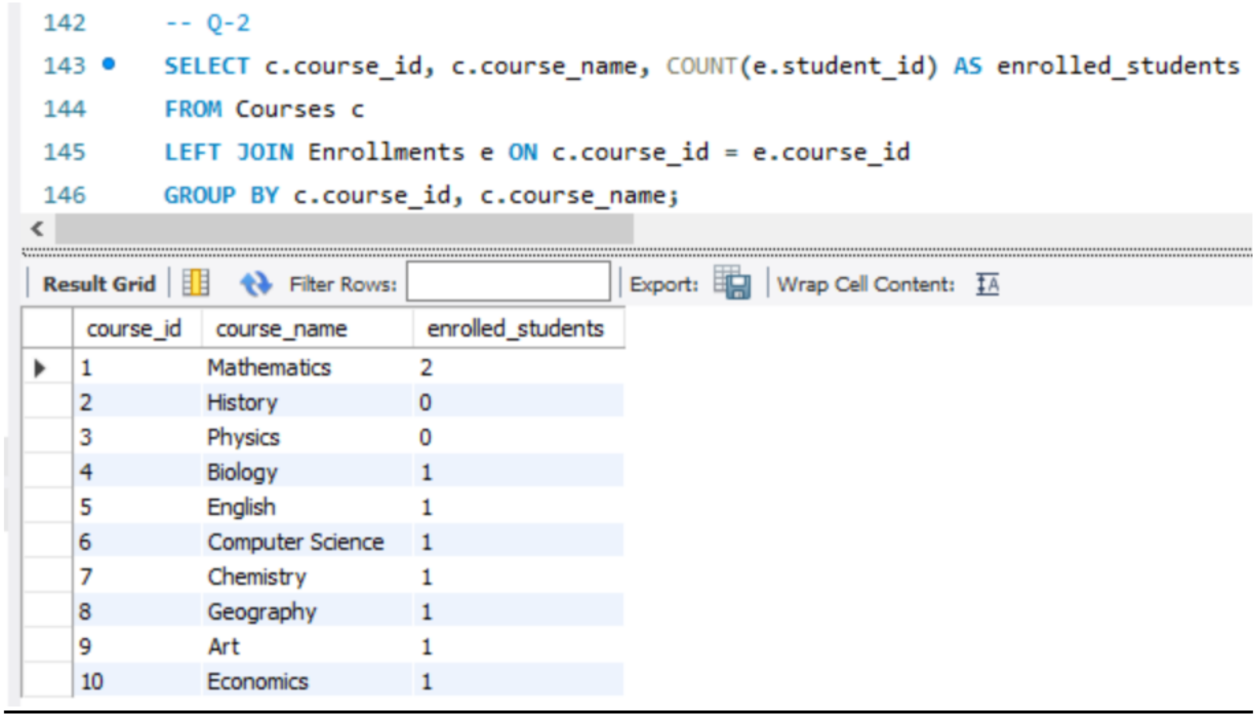
9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

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



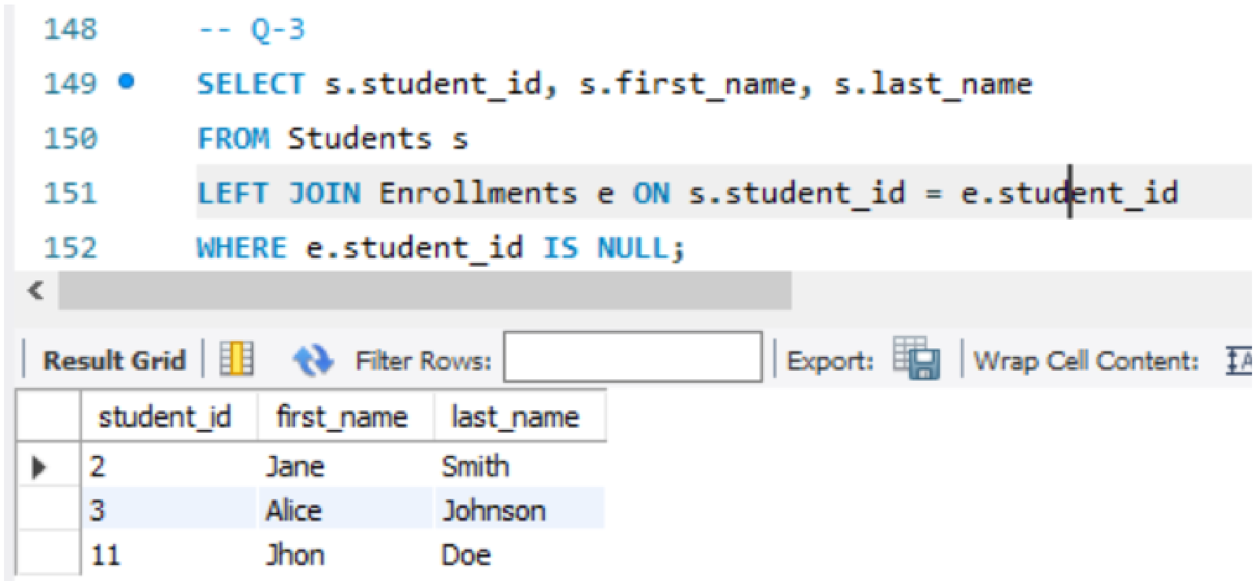
```
-- Q-1
SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
WHERE s.student_id = 7
GROUP BY s.student_id, s.first_name, s.last_name;
```

student_id	first_name	last_name	total_payments
7	Sophia	Garcia	95



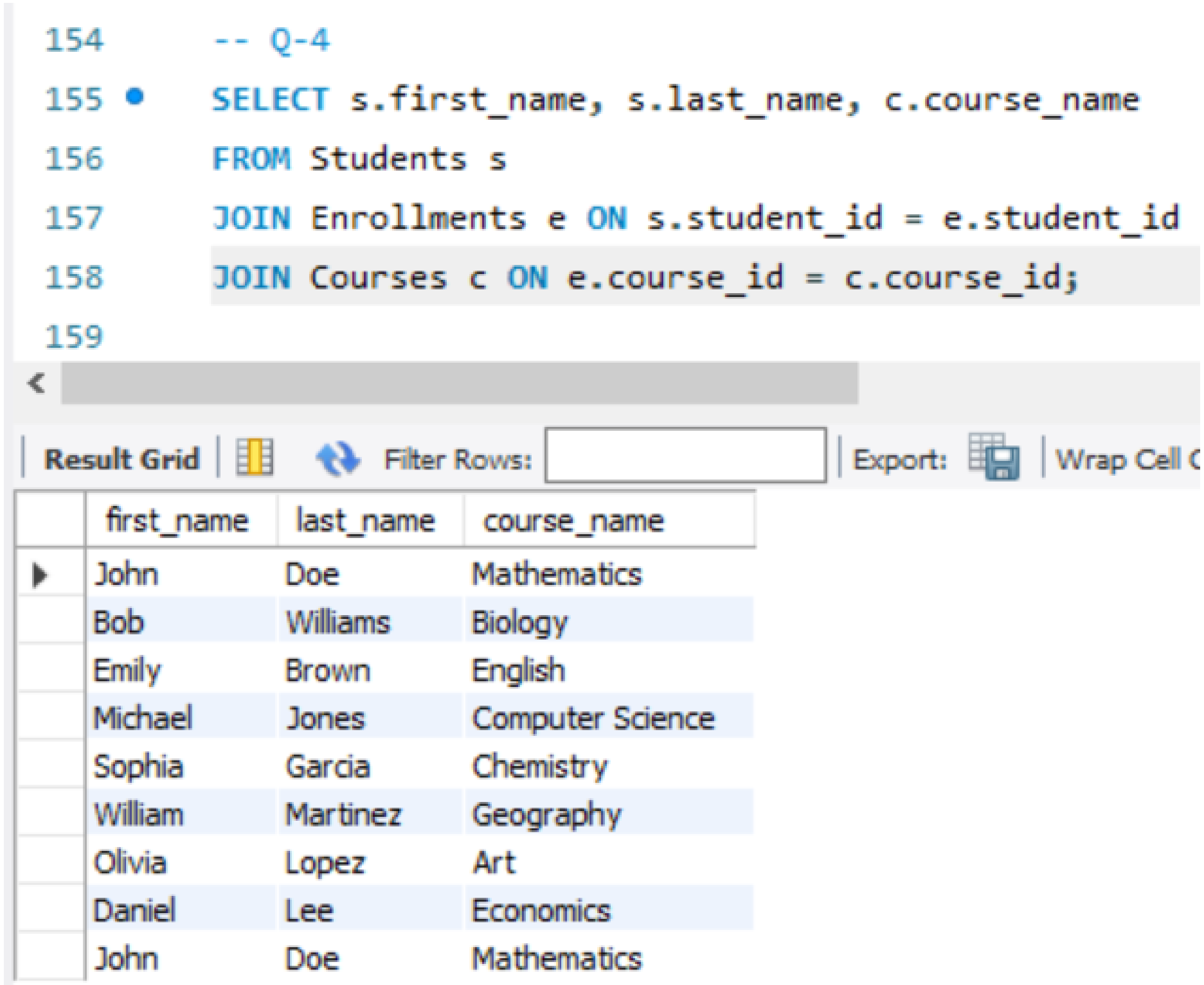
```
-- Q-2
SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name;
```

course_id	course_name	enrolled_students
1	Mathematics	2
2	History	0
3	Physics	0
4	Biology	1
5	English	1
6	Computer Science	1
7	Chemistry	1
8	Geography	1
9	Art	1
10	Economics	1



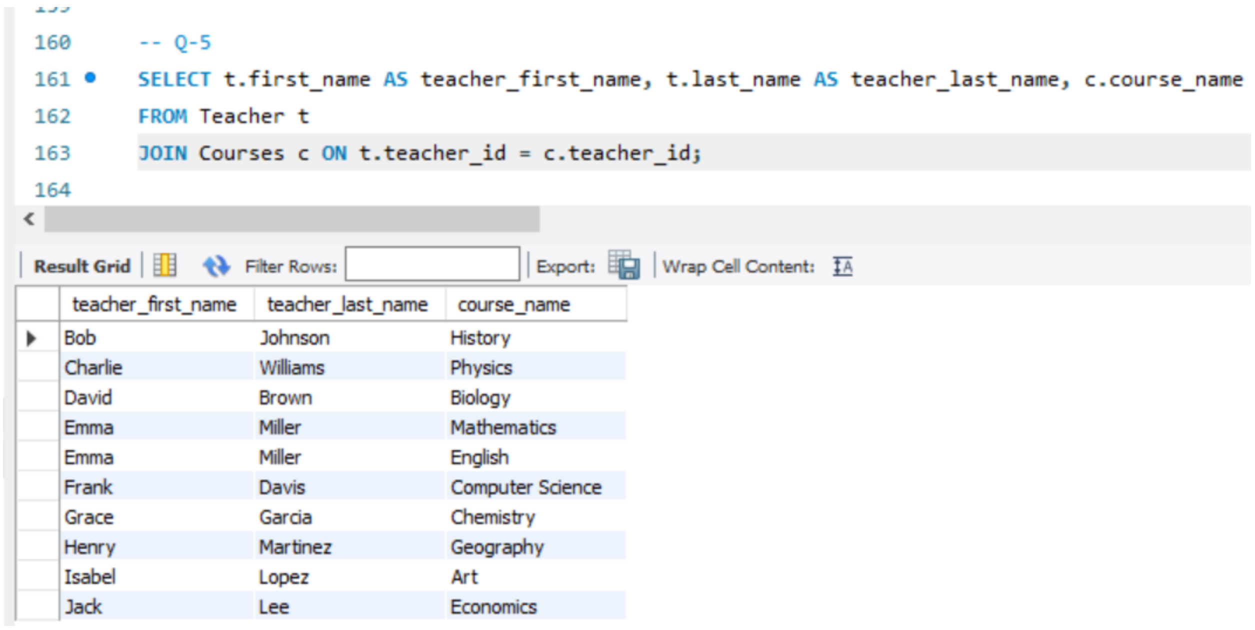
```sql
-- Q-3
SELECT s.student_id, s.first_name, s.last_name
FROM Students s
LEFT JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.student_id IS NULL;
```

student_id	first_name	last_name
2	Jane	Smith
3	Alice	Johnson
11	Jhon	Doe



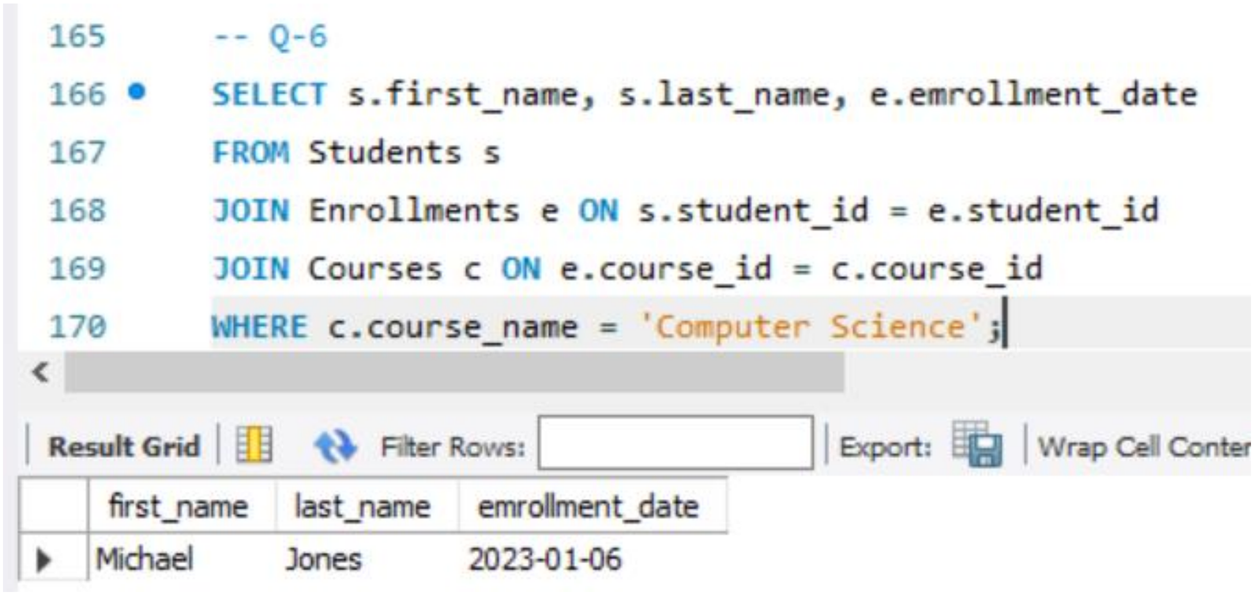
```sql
-- Q-4
SELECT s.first_name, s.last_name, c.course_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;
```

first_name	last_name	course_name
John	Doe	Mathematics
Bob	Williams	Biology
Emily	Brown	English
Michael	Jones	Computer Science
Sophia	Garcia	Chemistry
William	Martinez	Geography
Olivia	Lopez	Art
Daniel	Lee	Economics
John	Doe	Mathematics



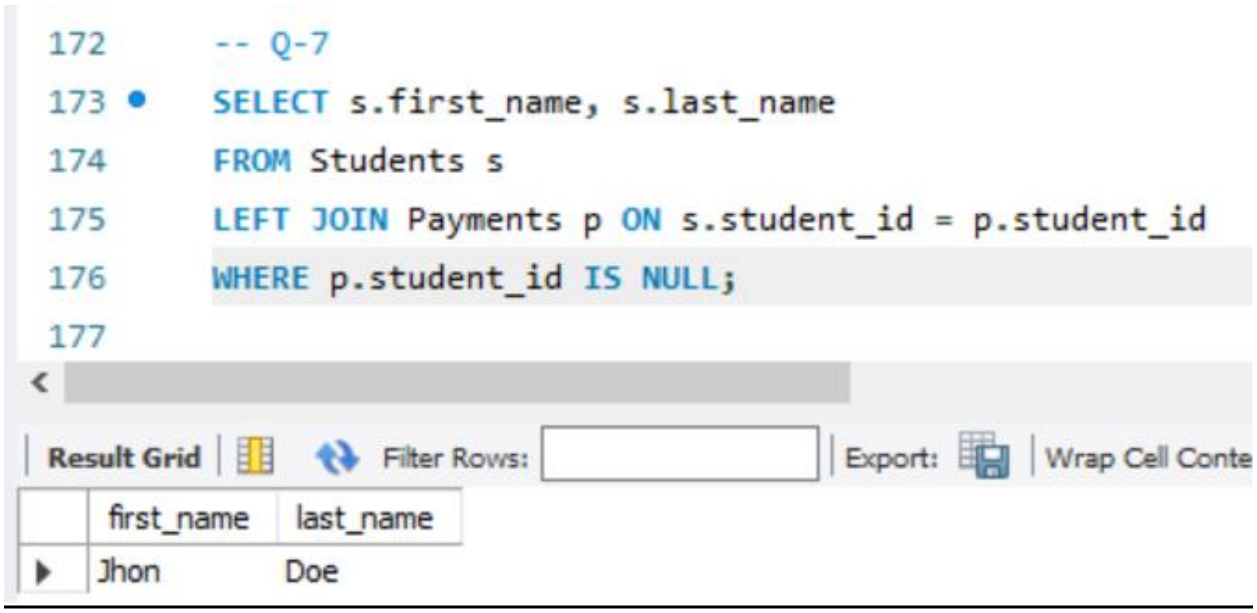
```sql
-- Q-5
SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id;
```

teacher_first_name	teacher_last_name	course_name
Bob	Johnson	History
Charlie	Williams	Physics
David	Brown	Biology
Emma	Miller	Mathematics
Emma	Miller	English
Frank	Davis	Computer Science
Grace	Garcia	Chemistry
Henry	Martinez	Geography
Isabel	Lopez	Art
Jack	Lee	Economics



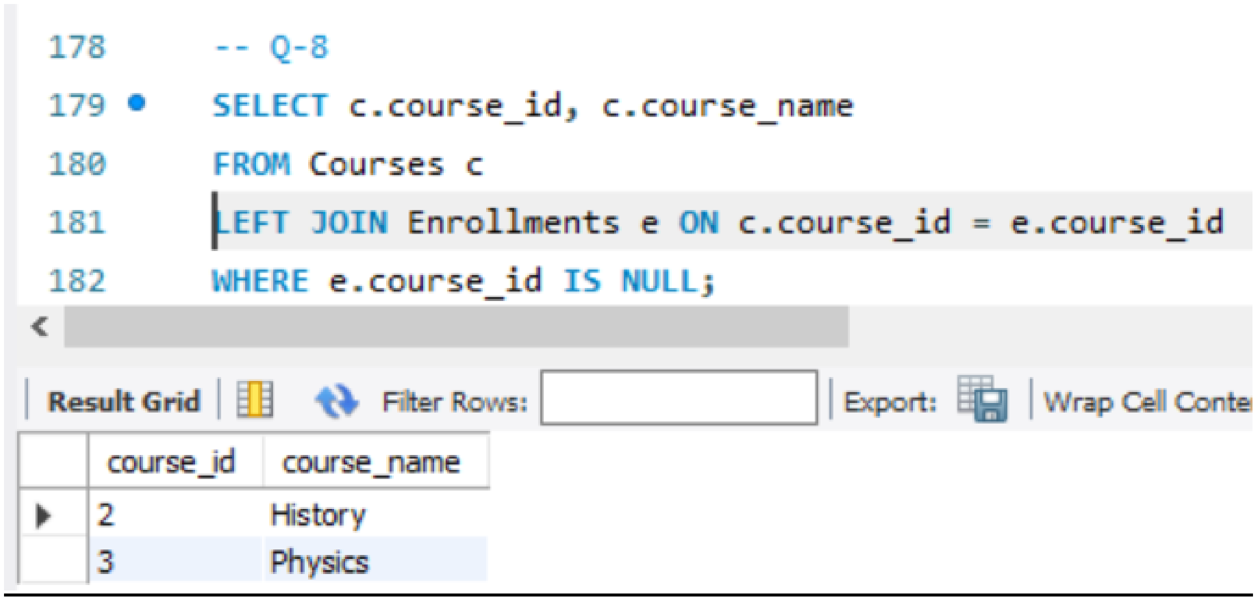
```sql
-- Q-6
SELECT s.first_name, s.last_name, e.emrollment_date
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE c.course_name = 'Computer Science';
```

first_name	last_name	emrollment_date
Michael	Jones	2023-01-06



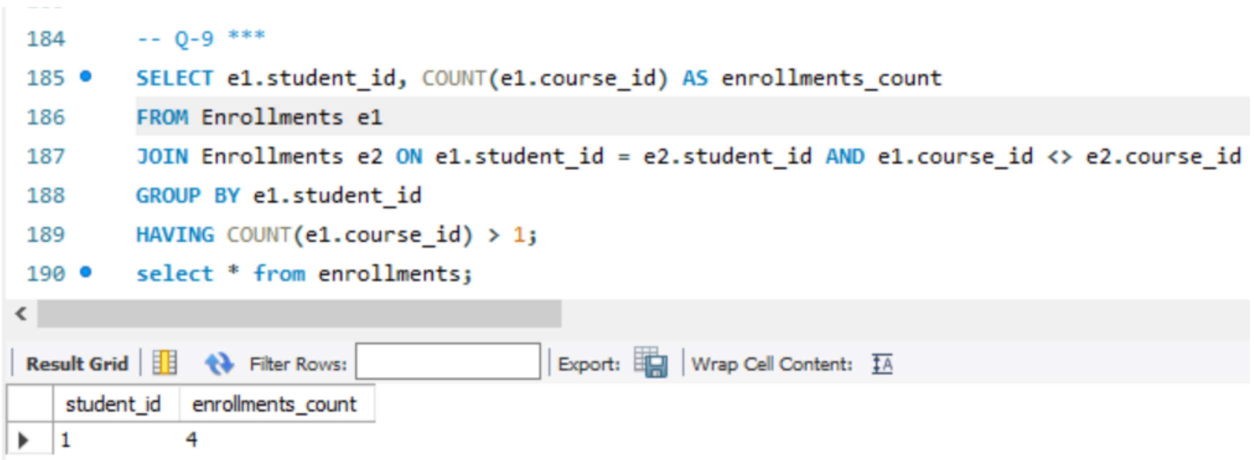
```
-- Q-7
SELECT s.first_name, s.last_name
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
WHERE p.student_id IS NULL;
```

first_name	last_name
Jhon	Doe



```
-- Q-8
SELECT c.course_id, c.course_name
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
WHERE e.course_id IS NULL;
```

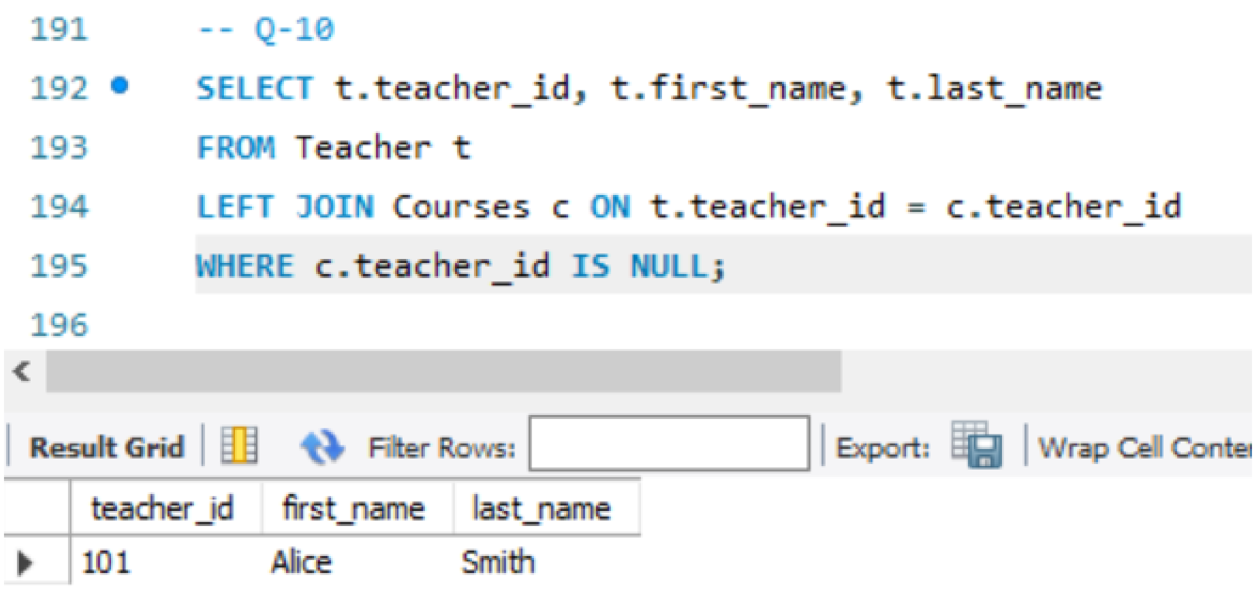
course_id	course_name
2	History
3	Physics



```
-- Q-9 ***
SELECT e1.student_id, COUNT(e1.course_id) AS enrollments_count
FROM Enrollments e1
JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id <> e2.course_id
GROUP BY e1.student_id
HAVING COUNT(e1.course_id) > 1;
select * from enrollments;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	enrollments_count
1	4



```
-- Q-10
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.teacher_id IS NULL;
```

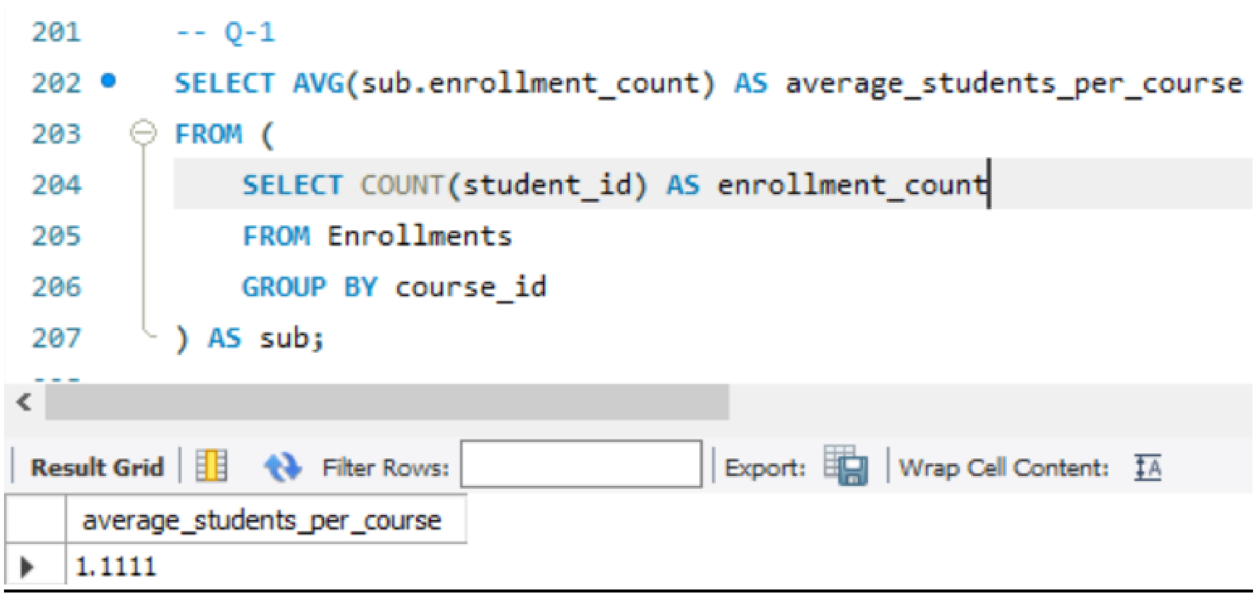
Result Grid | Filter Rows: | Export: | Wrap Cell Conte

teacher_id	first_name	last_name
101	Alice	Smith

<u>Task-4:</u>

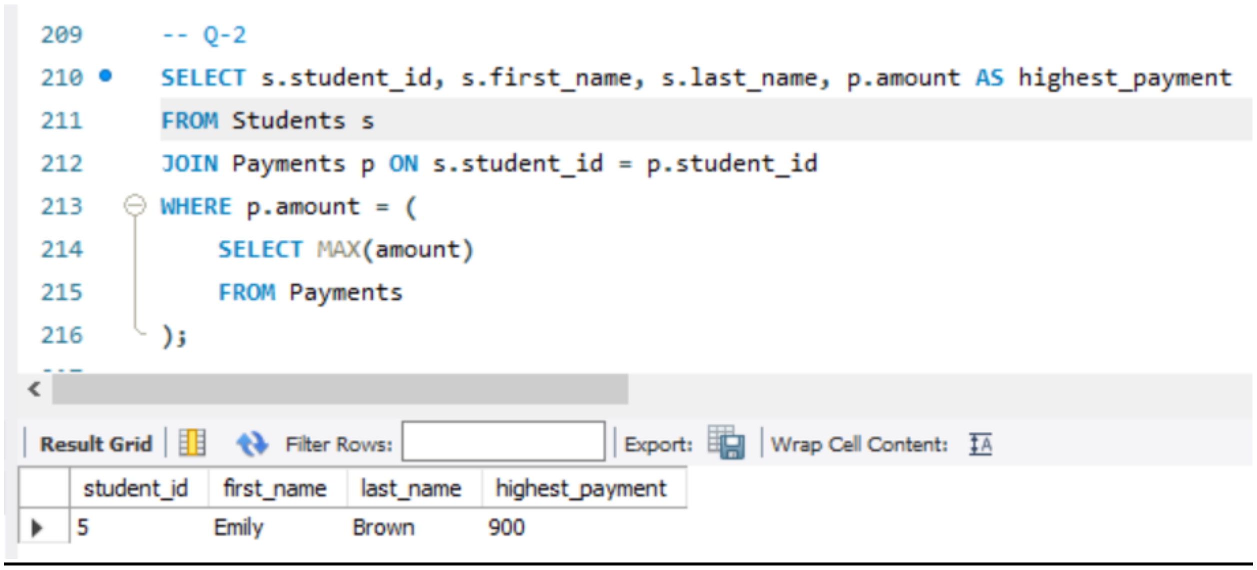
1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.
2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.
3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.
4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.
5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.
6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

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.
9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.
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.
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.
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.
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.



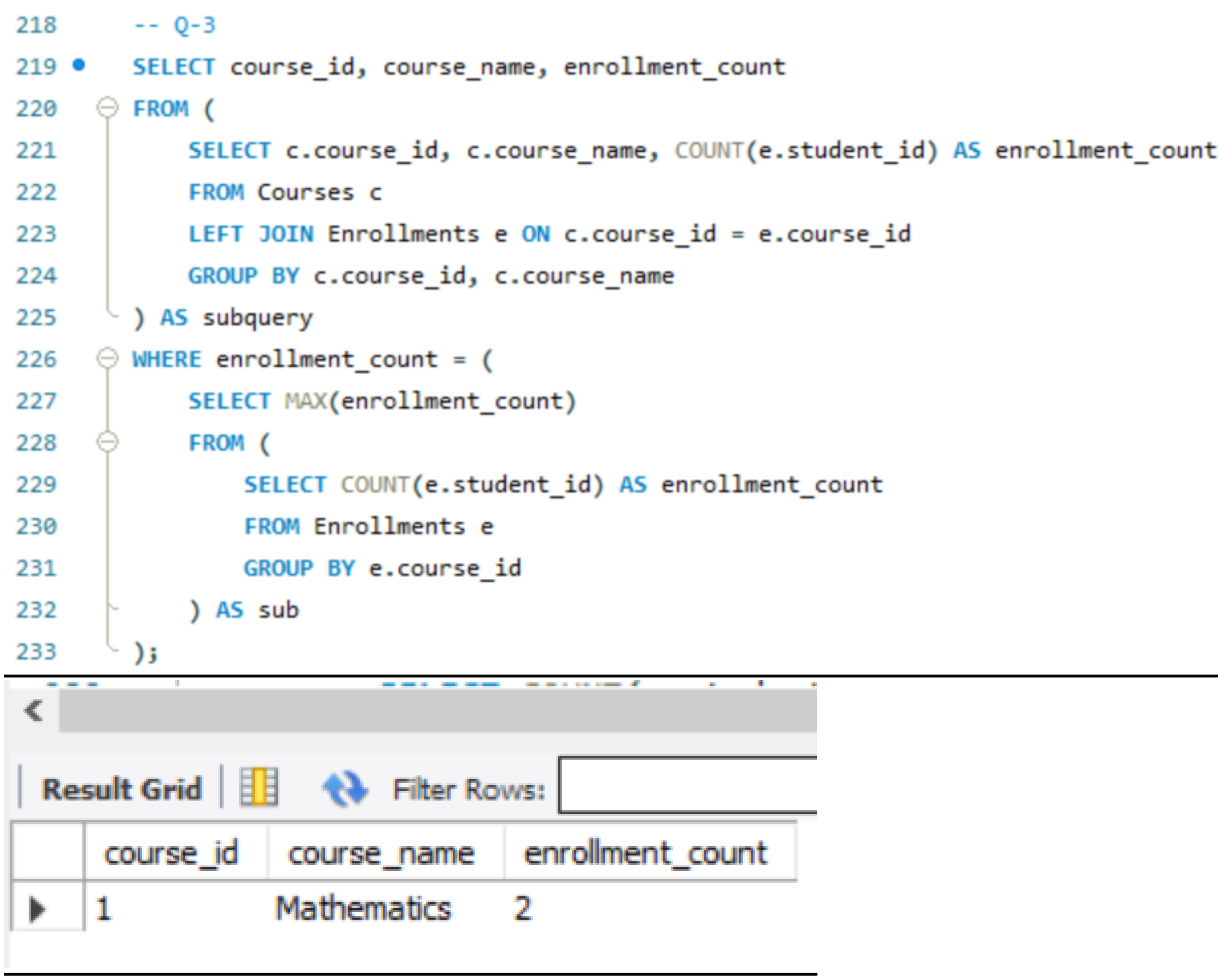
```
-- Q-1
SELECT AVG(sub.enrollment_count) AS average_students_per_course
FROM (
    SELECT COUNT(student_id) AS enrollment_count
    FROM Enrollments
    GROUP BY course_id
) AS sub;
```

```
average_students_per_course
1.1111
```



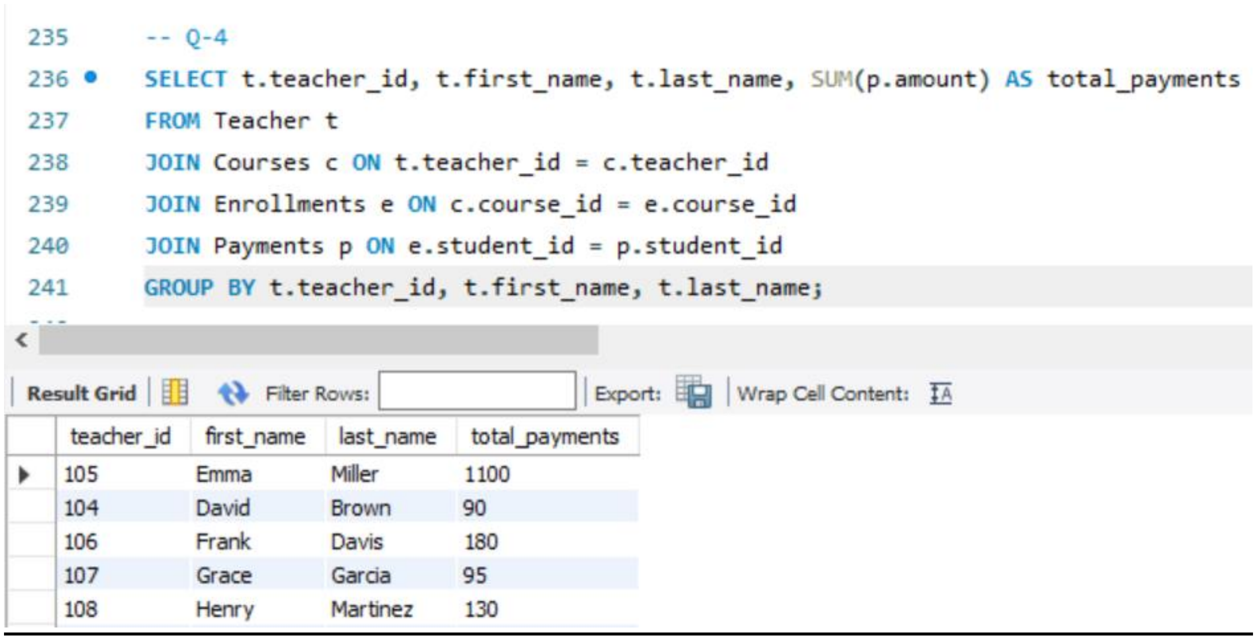
```sql
-- Q-2
SELECT s.student_id, s.first_name, s.last_name, p.amount AS highest_payment
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
WHERE p.amount = (
    SELECT MAX(amount)
    FROM Payments
);
```

student_id	first_name	last_name	highest_payment
5	Emily	Brown	900



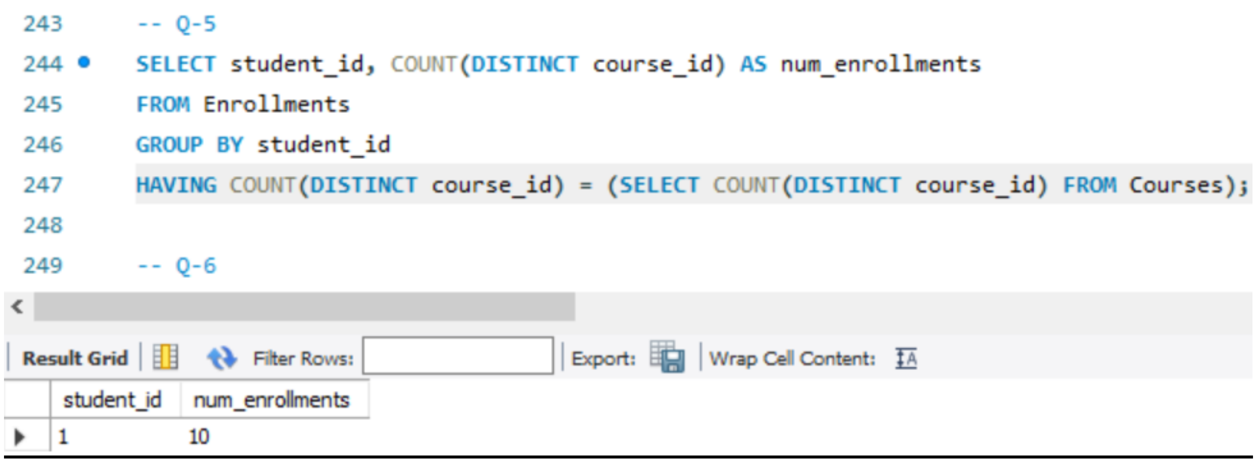
```sql
-- Q-3
SELECT course_id, course_name, enrollment_count
FROM (
    SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrollment_count
    FROM Courses c
    LEFT JOIN Enrollments e ON c.course_id = e.course_id
    GROUP BY c.course_id, c.course_name
) AS subquery
WHERE enrollment_count = (
    SELECT MAX(enrollment_count)
    FROM (
        SELECT COUNT(e.student_id) AS enrollment_count
        FROM Enrollments e
        GROUP BY e.course_id
    ) AS sub
);
```

course_id	course_name	enrollment_count
1	Mathematics	2



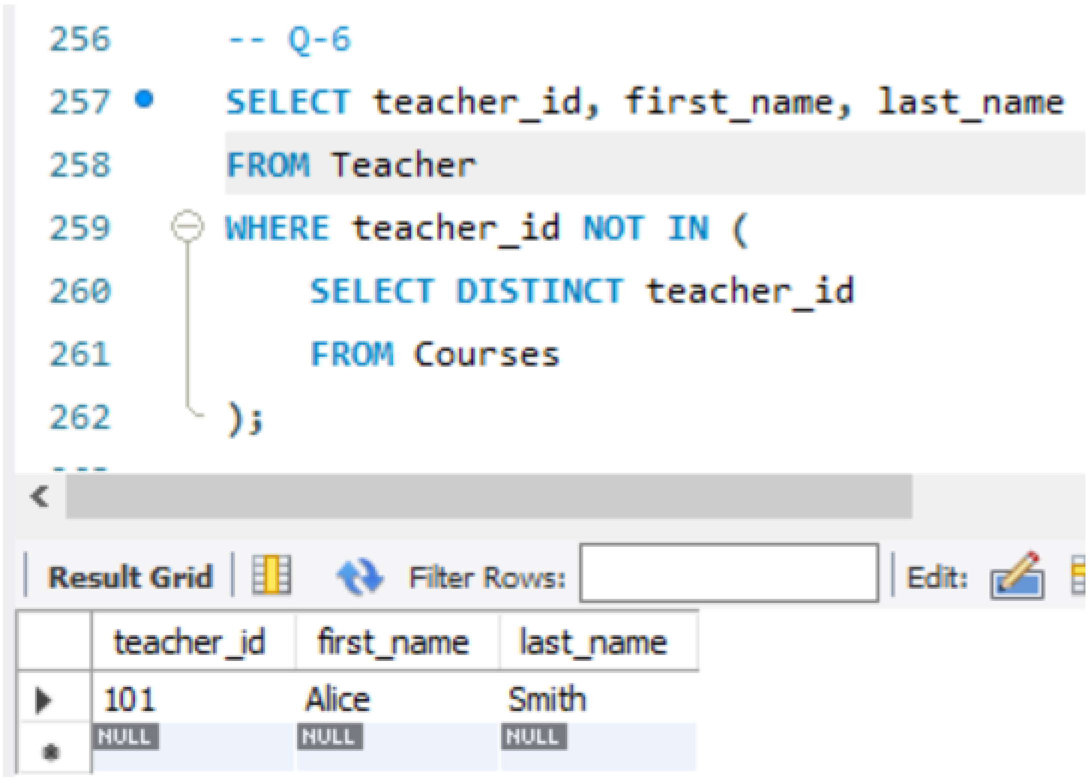
```
-- Q-4
SELECT t.teacher_id, t.first_name, t.last_name, SUM(p.amount) AS total_payments
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id
JOIN Enrollments e ON c.course_id = e.course_id
JOIN Payments p ON e.student_id = p.student_id
GROUP BY t.teacher_id, t.first_name, t.last_name;
```

teacher_id	first_name	last_name	total_payments
105	Emma	Miller	1100
104	David	Brown	90
106	Frank	Davis	180
107	Grace	Garcia	95
108	Henry	Martinez	130



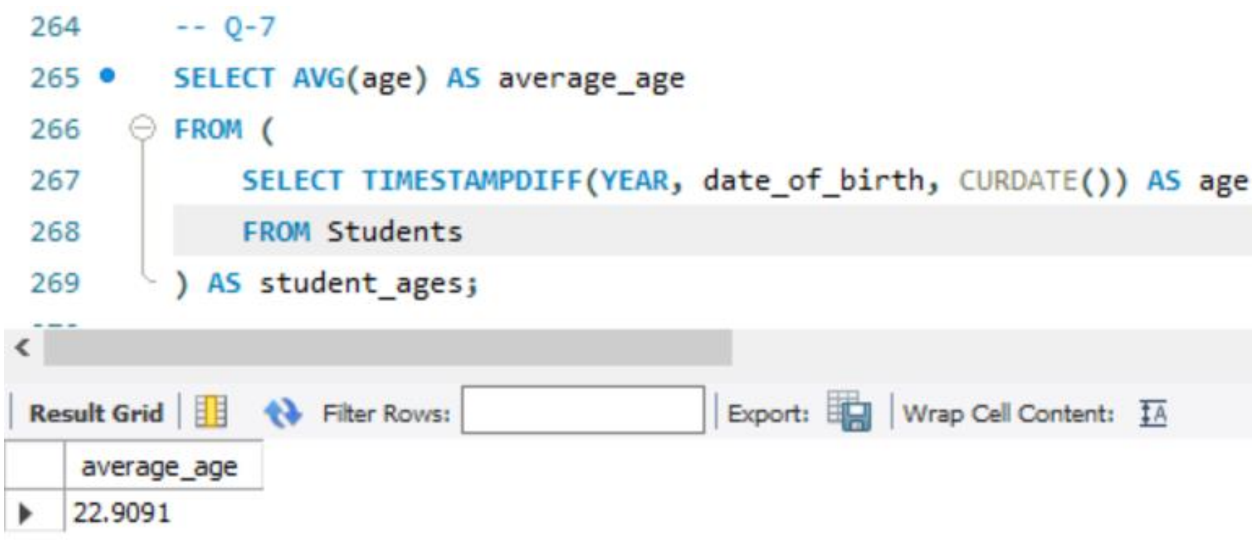
```
-- Q-5
SELECT student_id, COUNT(DISTINCT course_id) AS num_enrollments
FROM Enrollments
GROUP BY student_id
HAVING COUNT(DISTINCT course_id) = (SELECT COUNT(DISTINCT course_id) FROM Courses);

-- Q-6
```

student_id	num_enrollments
1	10



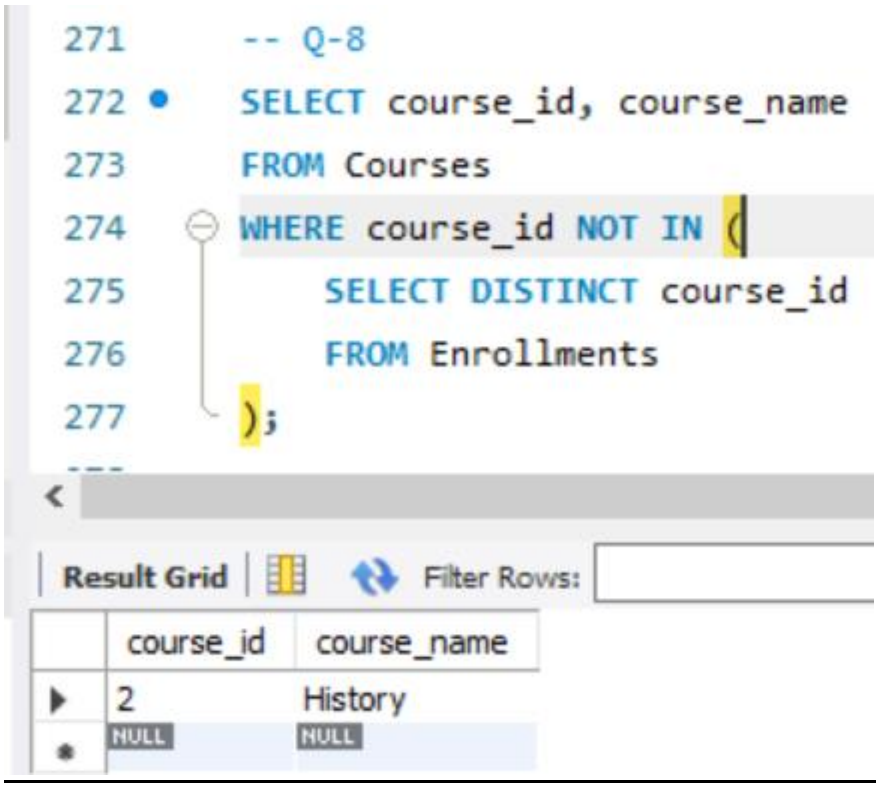
```
-- Q-6
SELECT teacher_id, first_name, last_name
FROM Teacher
WHERE teacher_id NOT IN (
    SELECT DISTINCT teacher_id
    FROM Courses
);
```

teacher_id	first_name	last_name
101	Alice	Smith
NULL	NULL	NULL



```
-- Q-7
SELECT AVG(age) AS average_age
FROM (
    SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    FROM Students
) AS student_ages;
```

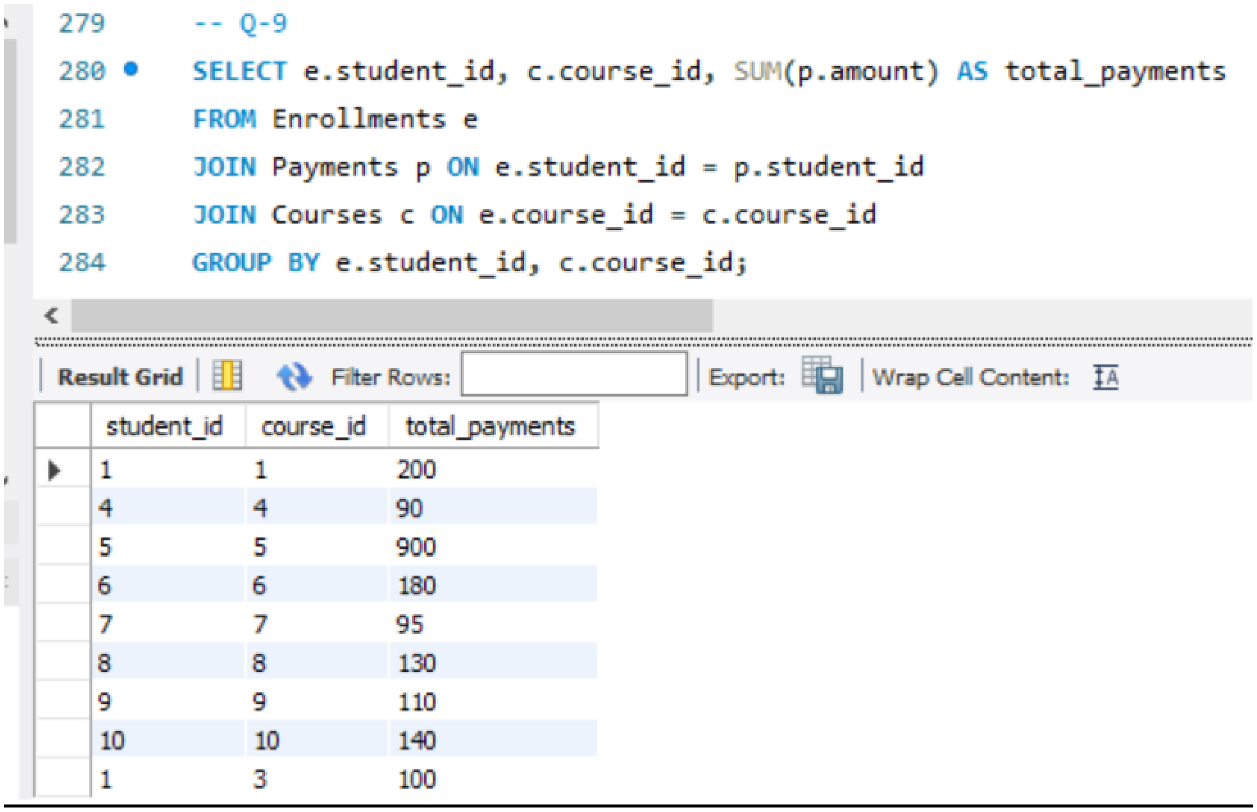
average_age
22.9091



```
-- Q-8
SELECT course_id, course_name
FROM Courses
WHERE course_id NOT IN (
    SELECT DISTINCT course_id
    FROM Enrollments
);
```

Result Grid | Filter Rows:

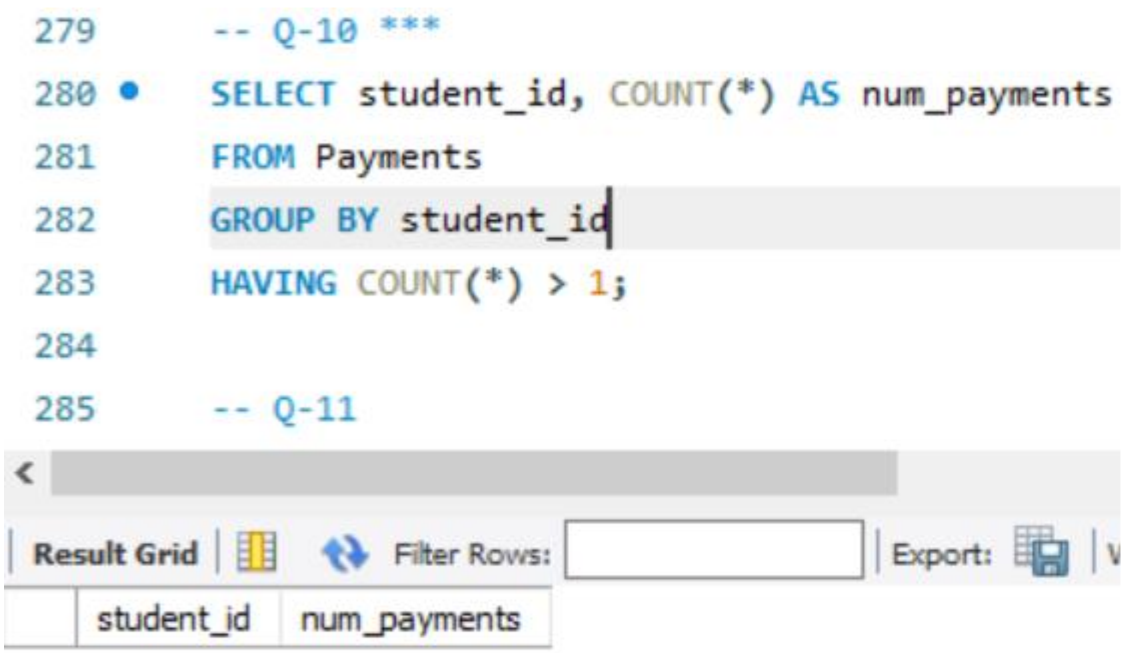
	course_id	course_name
▶	2	History
*	NULL	NULL



```
-- Q-9
SELECT e.student_id, c.course_id, SUM(p.amount) AS total_payments
FROM Enrollments e
JOIN Payments p ON e.student_id = p.student_id
JOIN Courses c ON e.course_id = c.course_id
GROUP BY e.student_id, c.course_id;
```

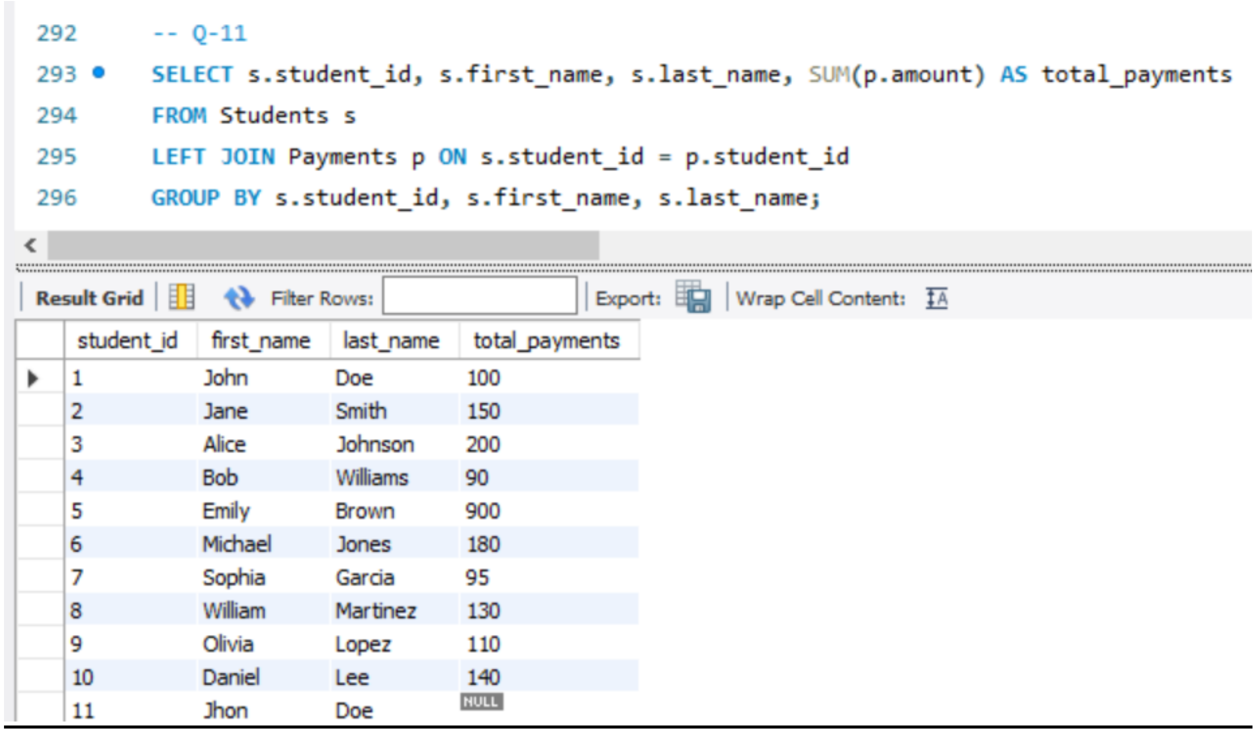
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	student_id	course_id	total_payments
▶	1	1	200
	4	4	90
	5	5	900
	6	6	180
	7	7	95
	8	8	130
	9	9	110
	10	10	140
	1	3	100



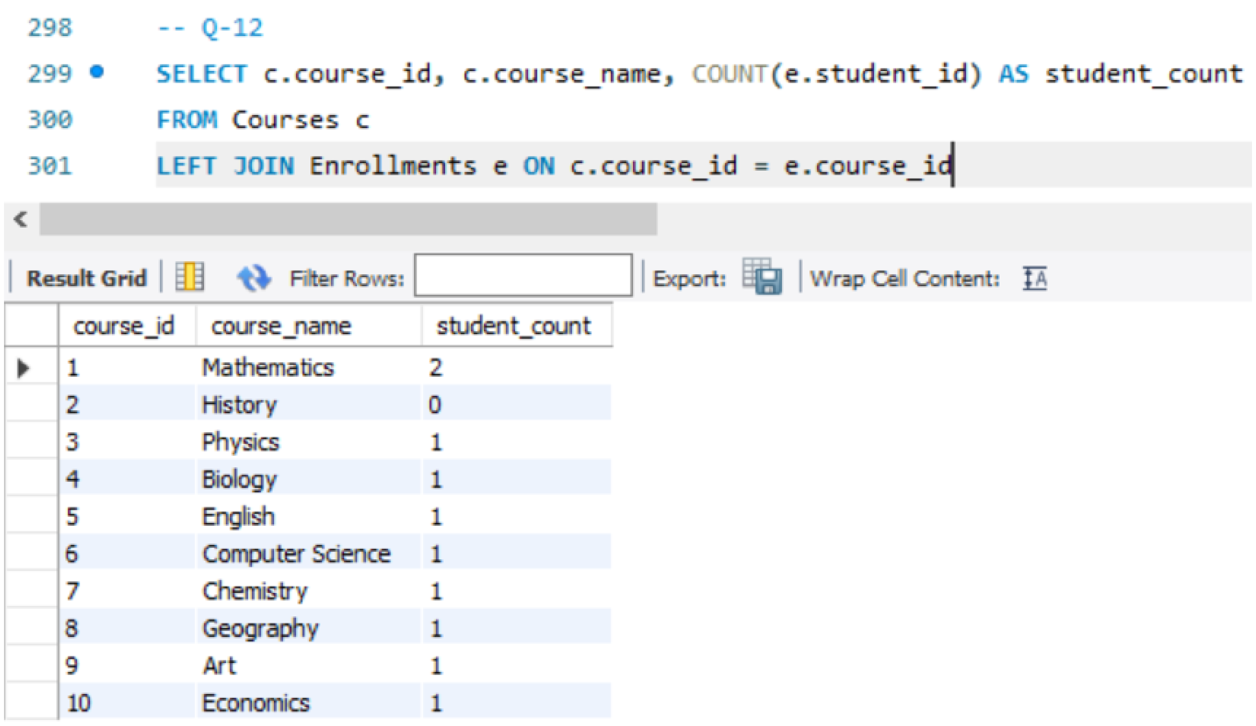
```
-- Q-10 ***
SELECT student_id, COUNT(*) AS num_payments
FROM Payments
GROUP BY student_id
HAVING COUNT(*) > 1;

-- Q-11
```

student_id	num_payments



```
-- Q-11
SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name, s.last_name;
```

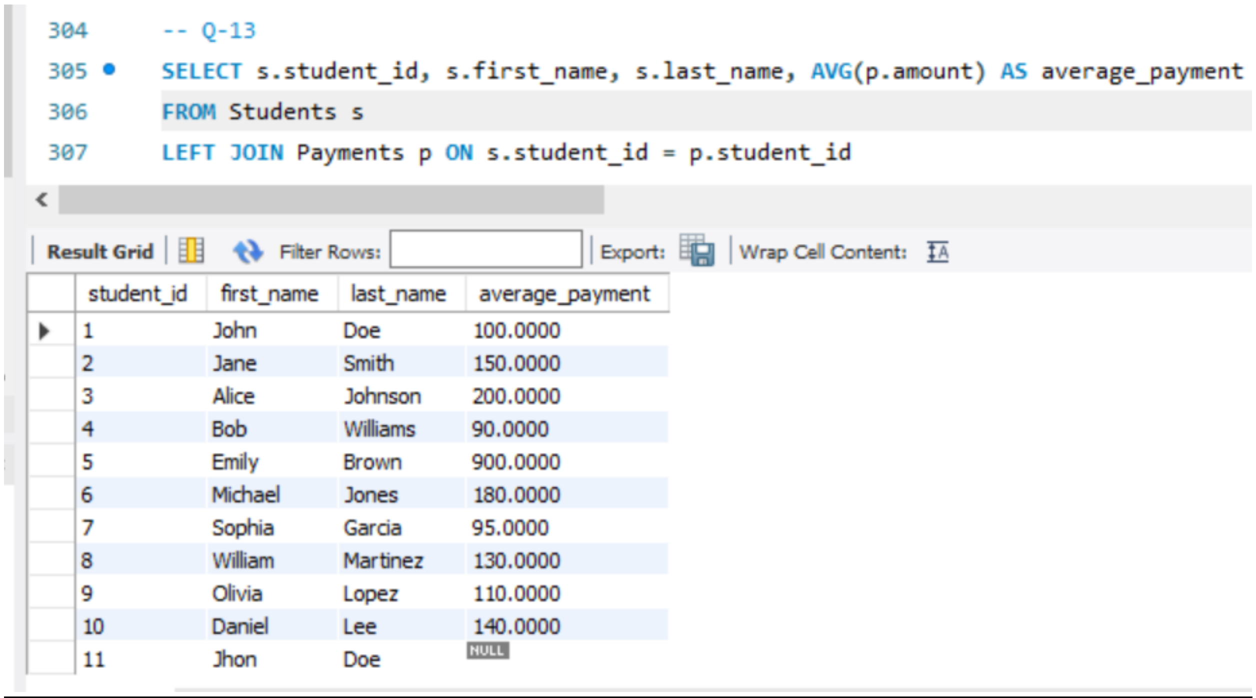
student_id	first_name	last_name	total_payments
1	John	Doe	100
2	Jane	Smith	150
3	Alice	Johnson	200
4	Bob	Williams	90
5	Emily	Brown	900
6	Michael	Jones	180
7	Sophia	Garcia	95
8	William	Martinez	130
9	Olivia	Lopez	110
10	Daniel	Lee	140
11	Jhon	Doe	NULL



```sql
-- Q-12
SELECT c.course_id, c.course_name, COUNT(e.student_id) AS student_count
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_id	course_name	student_count
1	Mathematics	2
2	History	0
3	Physics	1
4	Biology	1
5	English	1
6	Computer Science	1
7	Chemistry	1
8	Geography	1
9	Art	1
10	Economics	1



```sql
-- Q-13
SELECT s.student_id, s.first_name, s.last_name, AVG(p.amount) AS average_payment
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	average_payment
1	John	Doe	100.0000
2	Jane	Smith	150.0000
3	Alice	Johnson	200.0000
4	Bob	Williams	90.0000
5	Emily	Brown	900.0000
6	Michael	Jones	180.0000
7	Sophia	Garcia	95.0000
8	William	Martinez	130.0000
9	Olivia	Lopez	110.0000
10	Daniel	Lee	140.0000
11	Jhon	Doe	NULL