Hexaware Foundation Training (MSSql & Python)
SQL Assignment (Student Information System)

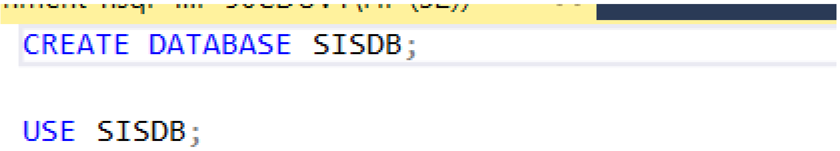
Name: Aathirainathan P

Date: 18-09-2024

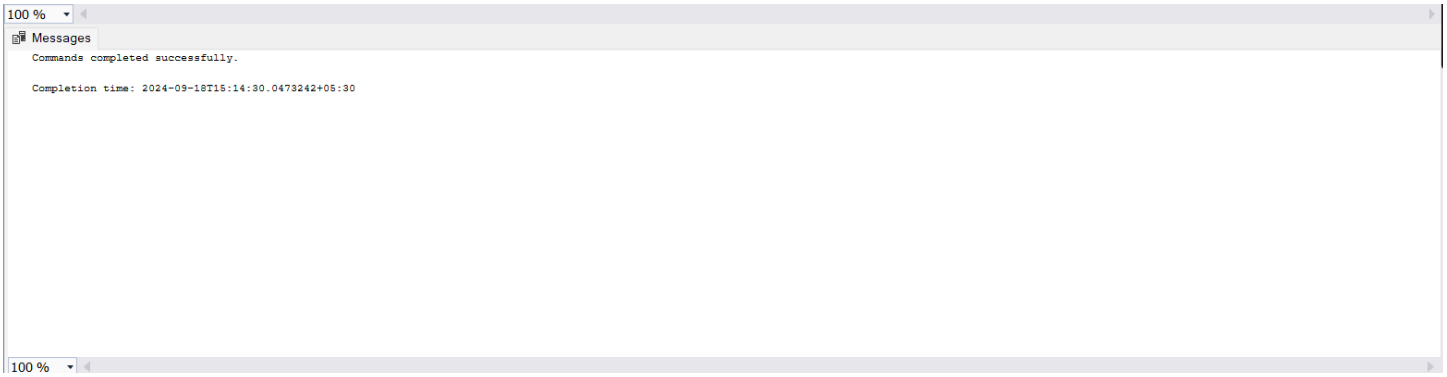
Task 1. Database Design

1. Create the database named "SISDB"

Query:



```
CREATE DATABASE SISDB;

USE SISDB;
```

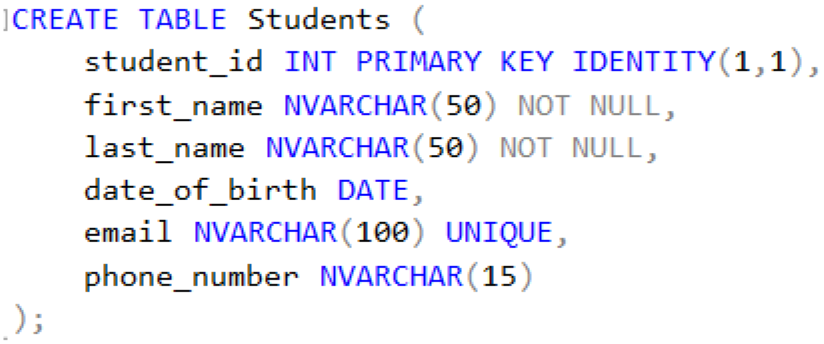


```
Commands completed successfully.

Completion time: 2024-09-18T15:14:30.0473242+05:30
```

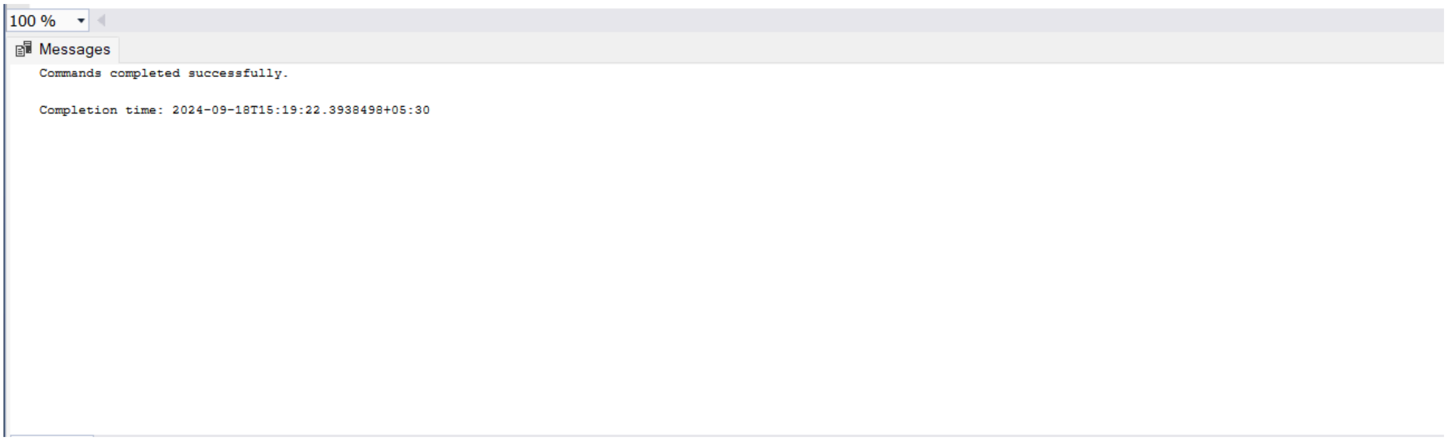
2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

a. Students

Query:



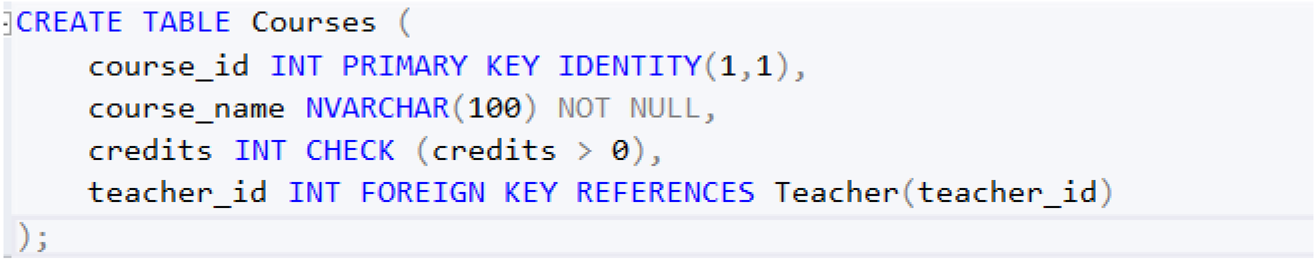
```
CREATE TABLE Students (
    student_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    date_of_birth DATE,
    email NVARCHAR(100) UNIQUE,
    phone_number NVARCHAR(15)
);
```



```
Commands completed successfully.

Completion time: 2024-09-18T15:19:22.3938498+05:30
```

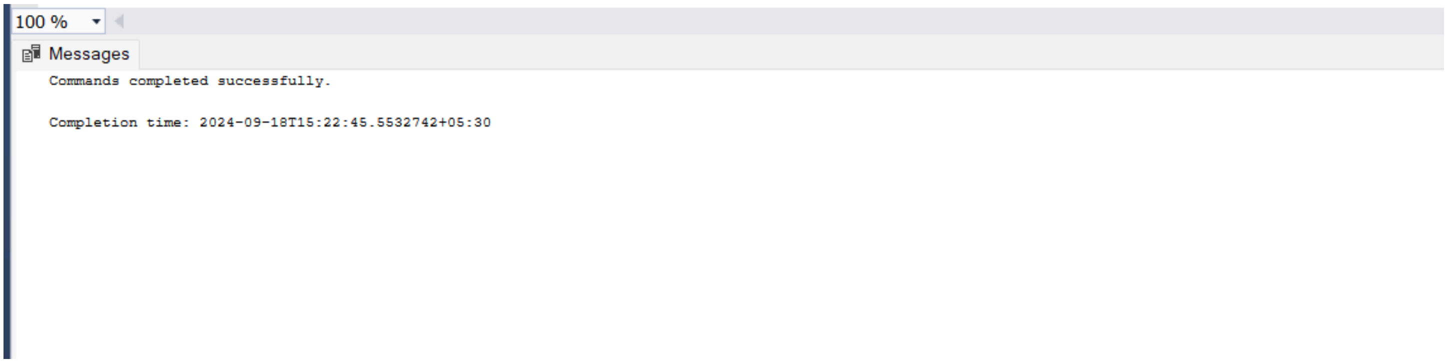
b. Courses

Query:



```
CREATE TABLE Courses (
    course_id INT PRIMARY KEY IDENTITY(1,1),
    course_name NVARCHAR(100) NOT NULL,
    credits INT CHECK (credits > 0),
    teacher_id INT FOREIGN KEY REFERENCES Teacher(teacher_id)
);
```

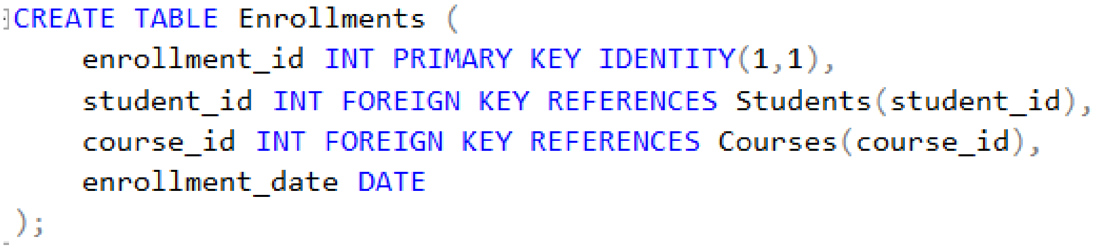
);



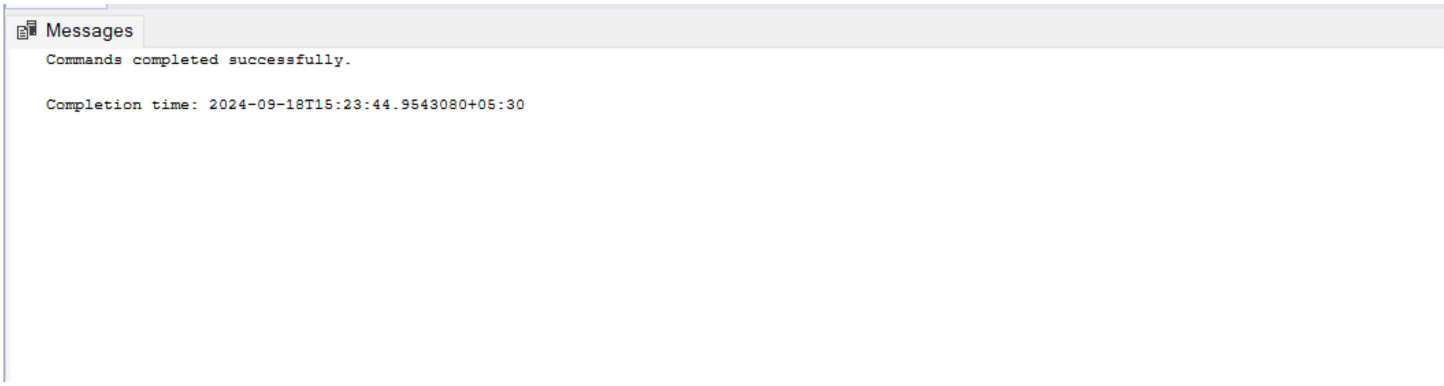
```
Commands completed successfully.

Completion time: 2024-09-18T15:22:45.5532742+05:30
```

c. Enrollments

Query:



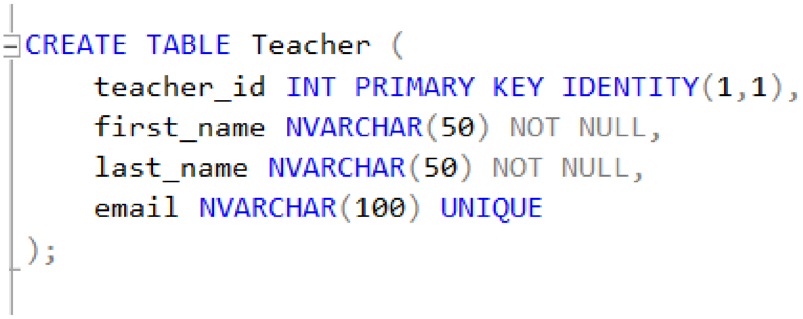
```
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT FOREIGN KEY REFERENCES Students(student_id),
    course_id INT FOREIGN KEY REFERENCES Courses(course_id),
    enrollment_date DATE
);
```



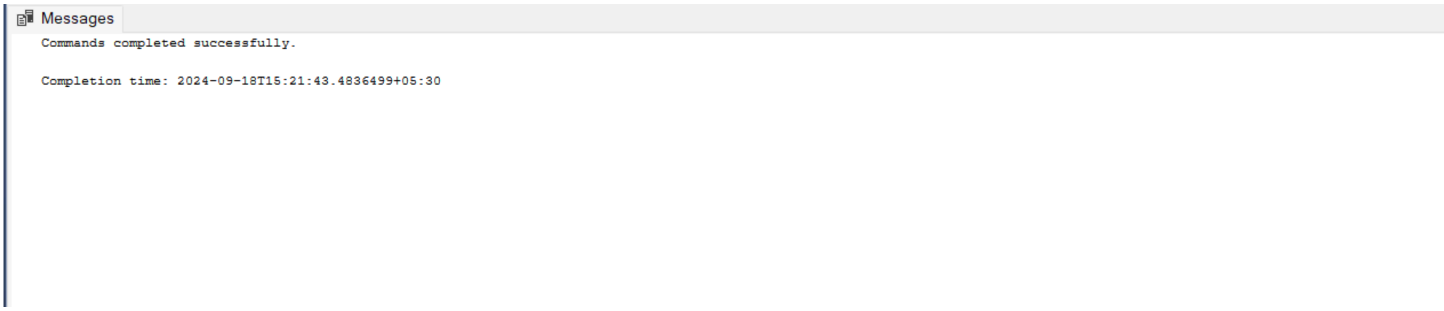
```
Commands completed successfully.

Completion time: 2024-09-18T15:23:44.9543080+05:30
```

d. Teacher

Query:



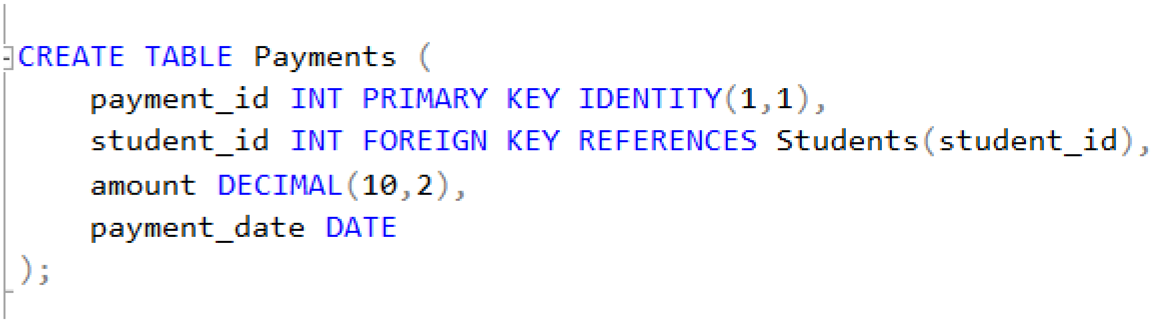
```
CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    email NVARCHAR(100) UNIQUE
);
```



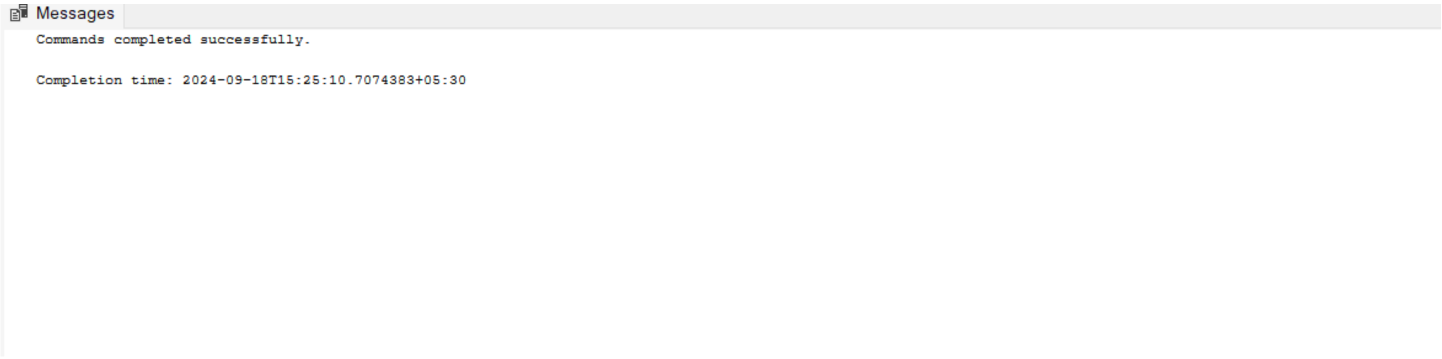
```
Commands completed successfully.

Completion time: 2024-09-18T15:21:43.4836499+05:30
```

e. Payments

Query:

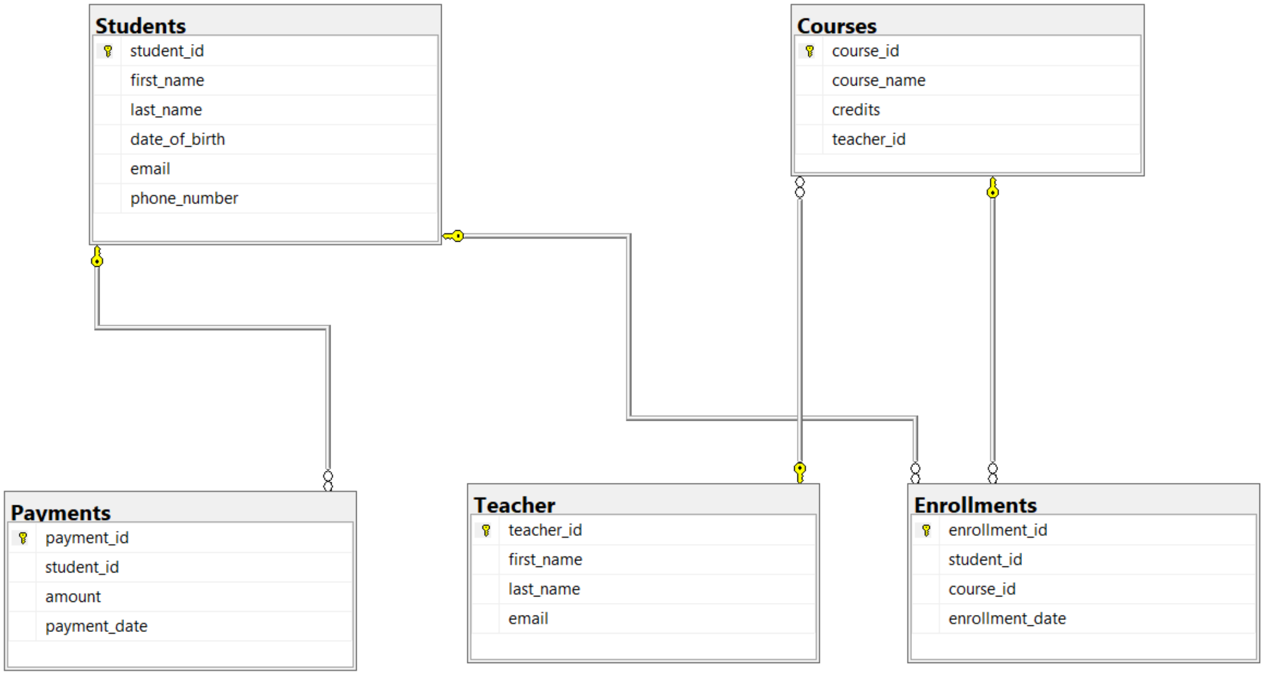


```
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT FOREIGN KEY REFERENCES Students(student_id),
    amount DECIMAL(10,2),
    payment_date DATE
);
```

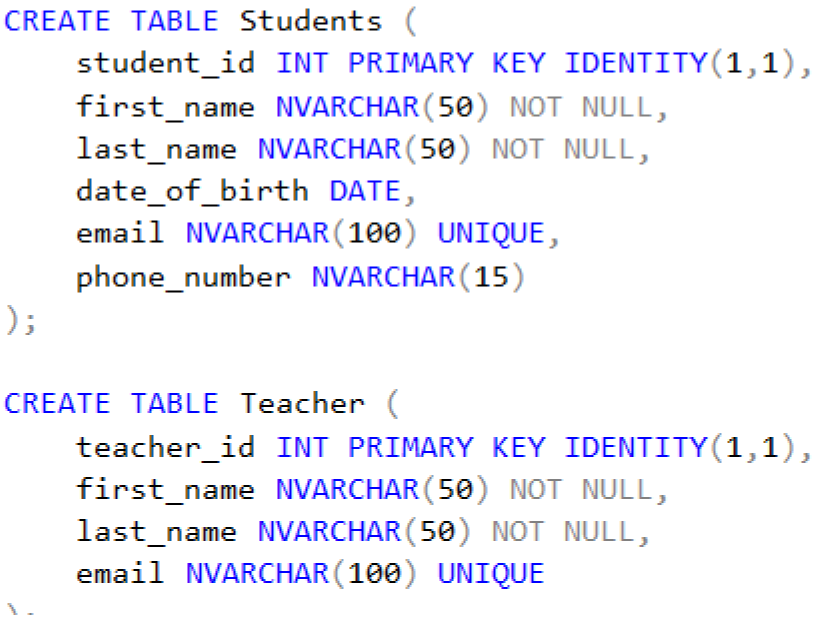


```
Commands completed successfully.

Completion time: 2024-09-18T15:25:10.7074383+05:30
```

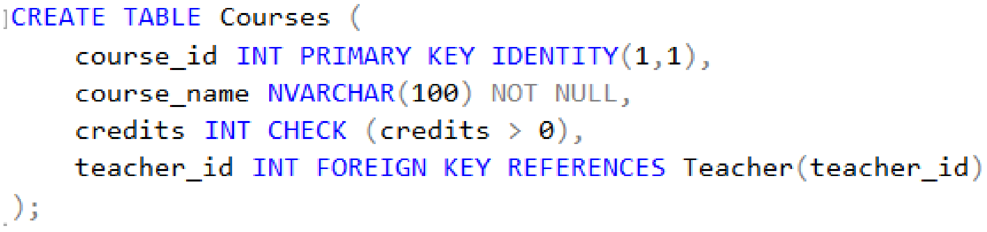
3. Create an ERD (Entity Relationship Diagram) for the database:



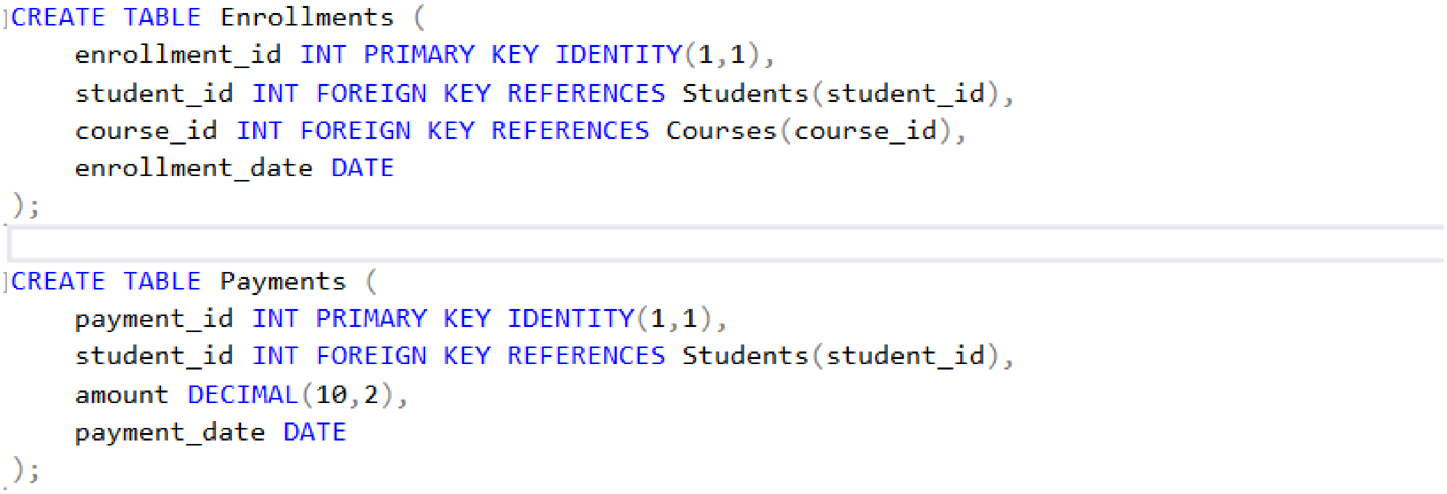
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



```
CREATE TABLE Students (
    student_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    date_of_birth DATE,
    email NVARCHAR(100) UNIQUE,
    phone_number NVARCHAR(15)
);

CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    email NVARCHAR(100) UNIQUE
);
```



```
CREATE TABLE Courses (
    course_id INT PRIMARY KEY IDENTITY(1,1),
    course_name NVARCHAR(100) NOT NULL,
    credits INT CHECK (credits > 0),
    teacher_id INT FOREIGN KEY REFERENCES Teacher(teacher_id)
);
```



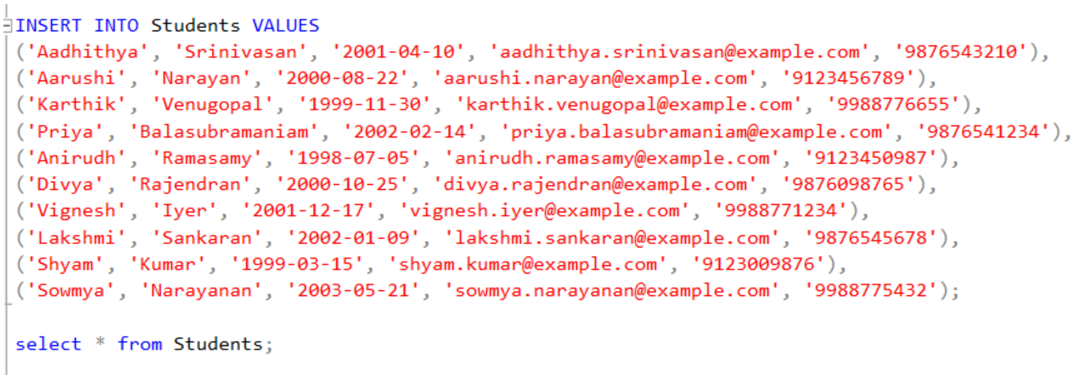
```
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT FOREIGN KEY REFERENCES Students(student_id),
    course_id INT FOREIGN KEY REFERENCES Courses(course_id),
    enrollment_date DATE
);

CREATE TABLE Payments (
    payment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT FOREIGN KEY REFERENCES Students(student_id),
    amount DECIMAL(10,2),
    payment_date DATE
);
```

All the necessary primary keys and foreign keys were added when tables were created.

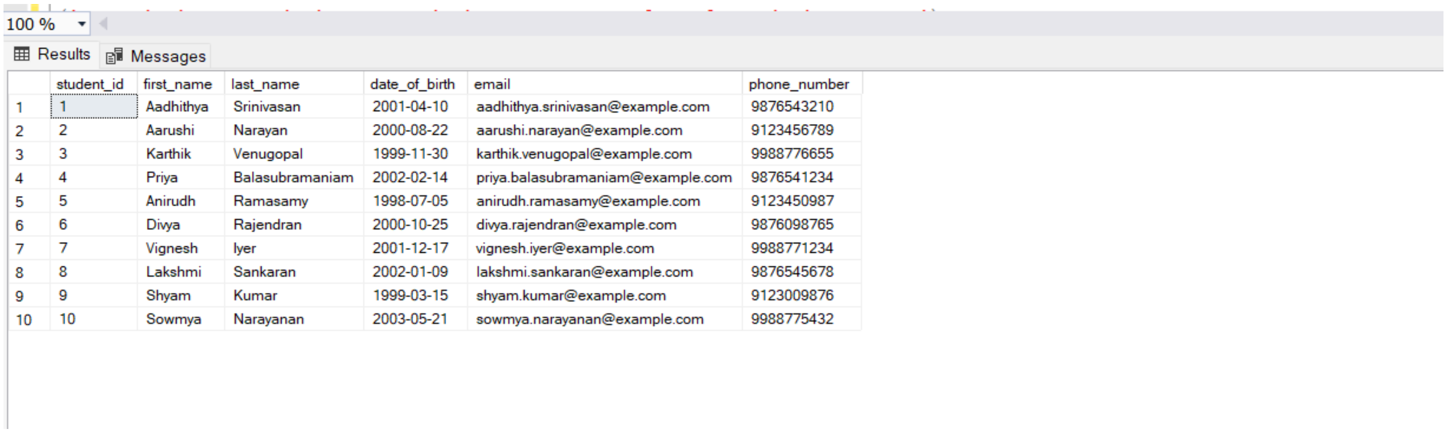
5. Insert at least 10 sample records into each of the following tables.

i. Students

Query:



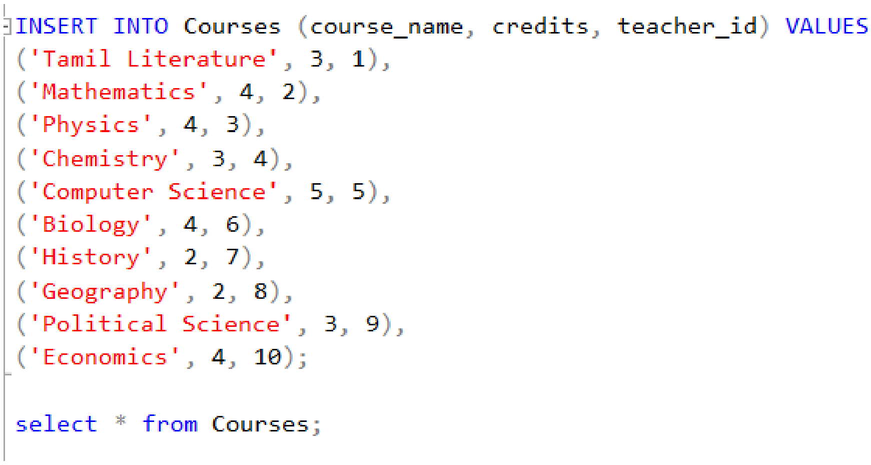
```
INSERT INTO Students VALUES
('Aadhithya', 'Srinivasan', '2001-04-10', 'aadhithya.srinivasan@example.com', '9876543210'),
('Aarushi', 'Narayan', '2000-08-22', 'aarushi.narayan@example.com', '9123456789'),
('Karthik', 'Venugopal', '1999-11-30', 'karthik.venugopal@example.com', '9988776655'),
('Priya', 'Balasubramaniam', '2002-02-14', 'priya.balasubramaniam@example.com', '9876541234'),
('Anirudh', 'Ramasamy', '1998-07-05', 'anirudh.ramasamy@example.com', '9123450987'),
('Divya', 'Rajendran', '2000-10-25', 'divya.rajendran@example.com', '9876098765'),
('Vignesh', 'Iyer', '2001-12-17', 'vignesh.iyer@example.com', '9988771234'),
('Lakshmi', 'Sankaran', '2002-01-09', 'lakshmi.sankaran@example.com', '9876545678'),
('Shyam', 'Kumar', '1999-03-15', 'shyam.kumar@example.com', '9123009876'),
('Sowmya', 'Narayanan', '2003-05-21', 'sowmya.narayanan@example.com', '9988775432');

select * from Students;
```



	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	Aadhithya	Srinivasan	2001-04-10	aadhithya.srinivasan@example.com	9876543210
2	2	Aarushi	Narayan	2000-08-22	aarushi.narayan@example.com	9123456789
3	3	Karthik	Venugopal	1999-11-30	karthik.venugopal@example.com	9988776655
4	4	Priya	Balasubramaniam	2002-02-14	priya.balasubramaniam@example.com	9876541234
5	5	Anirudh	Ramasamy	1998-07-05	anirudh.ramasamy@example.com	9123450987
6	6	Divya	Rajendran	2000-10-25	divya.rajendran@example.com	9876098765
7	7	Vignesh	Iyer	2001-12-17	vignesh.iyer@example.com	9988771234
8	8	Lakshmi	Sankaran	2002-01-09	lakshmi.sankaran@example.com	9876545678
9	9	Shyam	Kumar	1999-03-15	shyam.kumar@example.com	9123009876
10	10	Sowmya	Narayanan	2003-05-21	sowmya.narayanan@example.com	9988775432

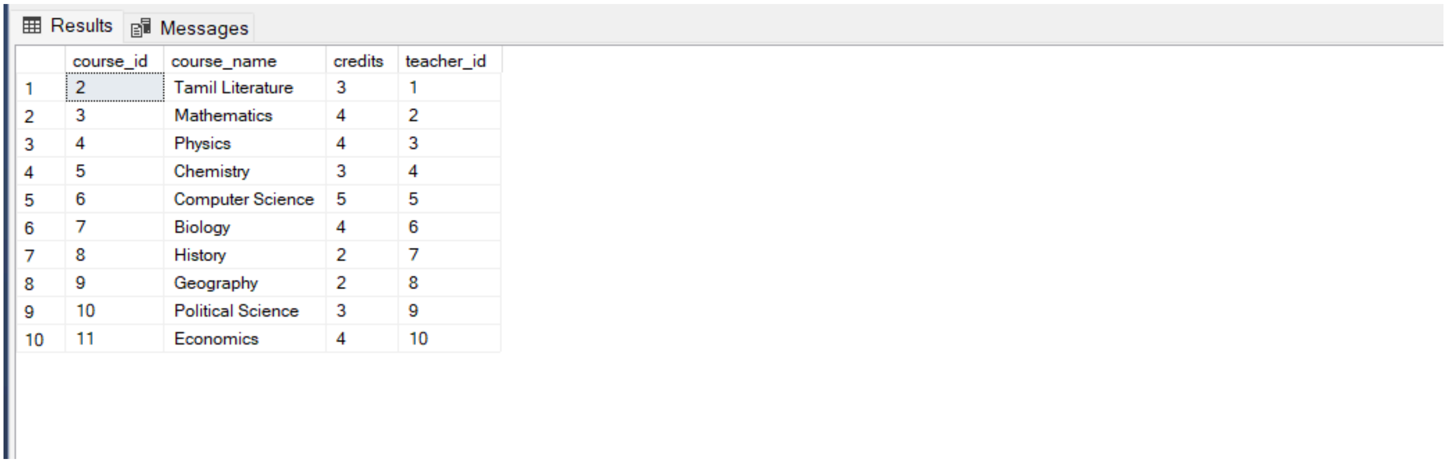
ii. Courses

Query:



```
INSERT INTO Courses (course_name, credits, teacher_id) VALUES
('Tamil Literature', 3, 1),
('Mathematics', 4, 2),
('Physics', 4, 3),
('Chemistry', 3, 4),
('Computer Science', 5, 5),
('Biology', 4, 6),
('History', 2, 7),
('Geography', 2, 8),
('Political Science', 3, 9),
('Economics', 4, 10);

select * from Courses;
```

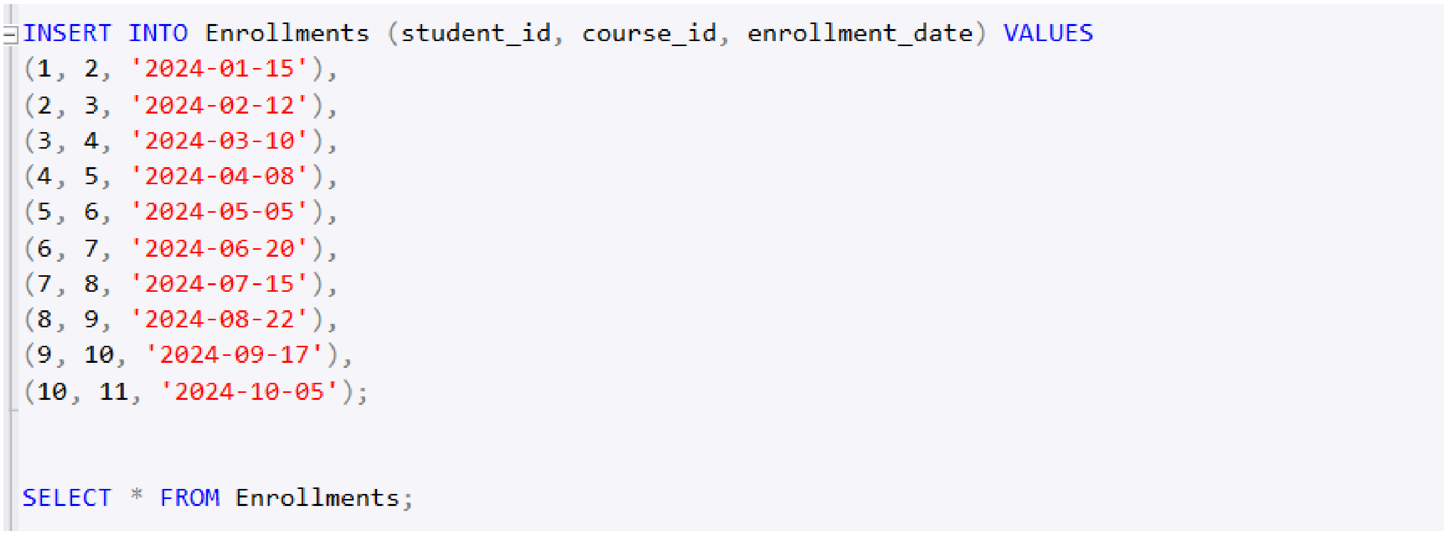
select * from Courses;



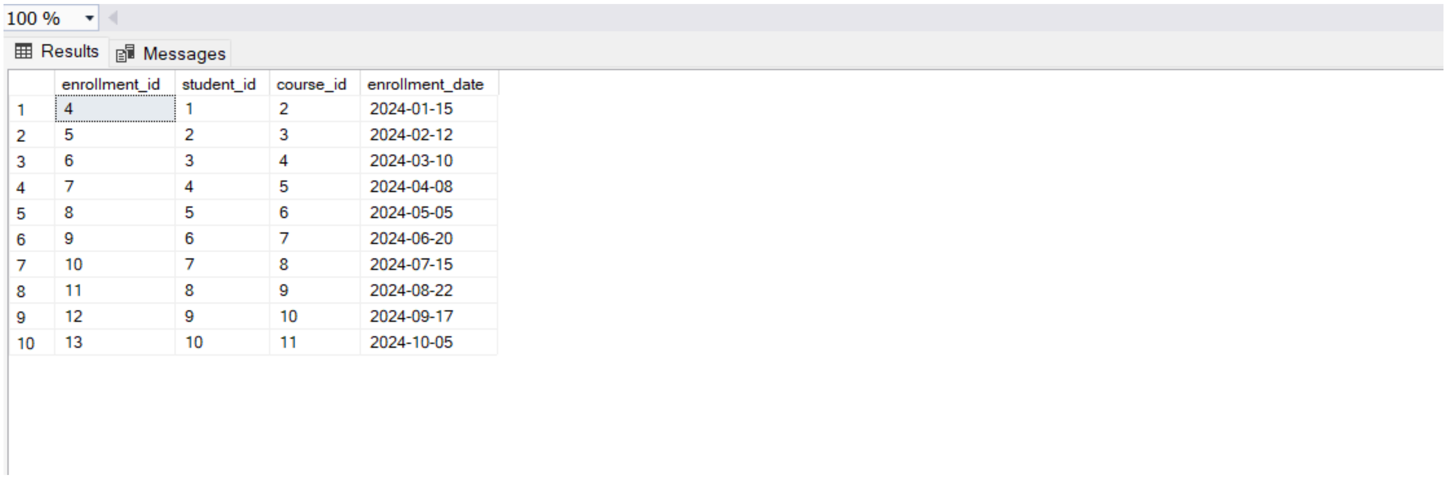
	course_id	course_name	credits	teacher_id
1	2	Tamil Literature	3	1
2	3	Mathematics	4	2
3	4	Physics	4	3
4	5	Chemistry	3	4
5	6	Computer Science	5	5
6	7	Biology	4	6
7	8	History	2	7
8	9	Geography	2	8
9	10	Political Science	3	9
10	11	Economics	4	10

iii. Enrollments

Query:



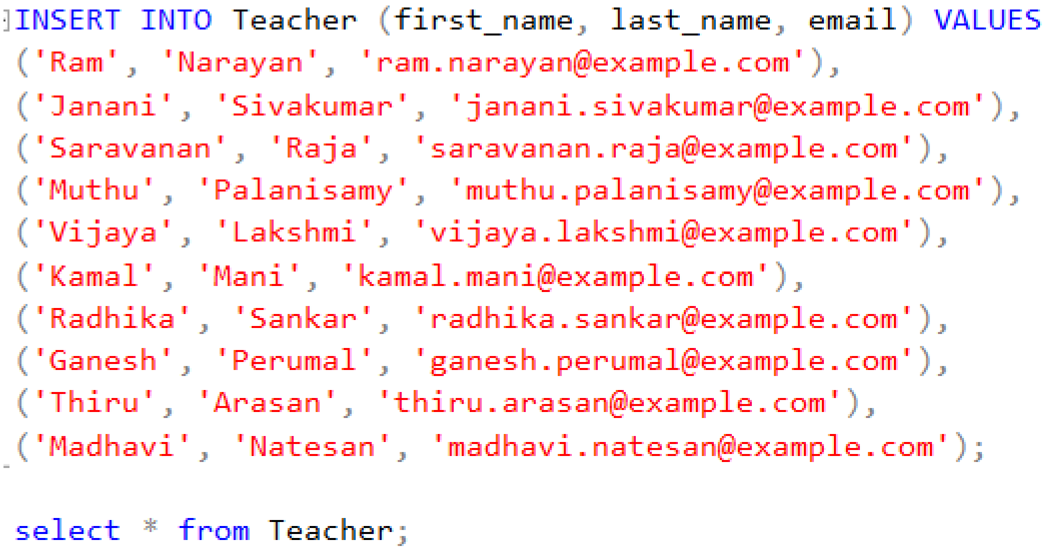
```
INSERT INTO Enrollments (student_id, course_id, enrollment_date) VALUES
(1, 2, '2024-01-15'),
(2, 3, '2024-02-12'),
(3, 4, '2024-03-10'),
(4, 5, '2024-04-08'),
(5, 6, '2024-05-05'),
(6, 7, '2024-06-20'),
(7, 8, '2024-07-15'),
(8, 9, '2024-08-22'),
(9, 10, '2024-09-17'),
(10, 11, '2024-10-05');


SELECT * FROM Enrollments;
```



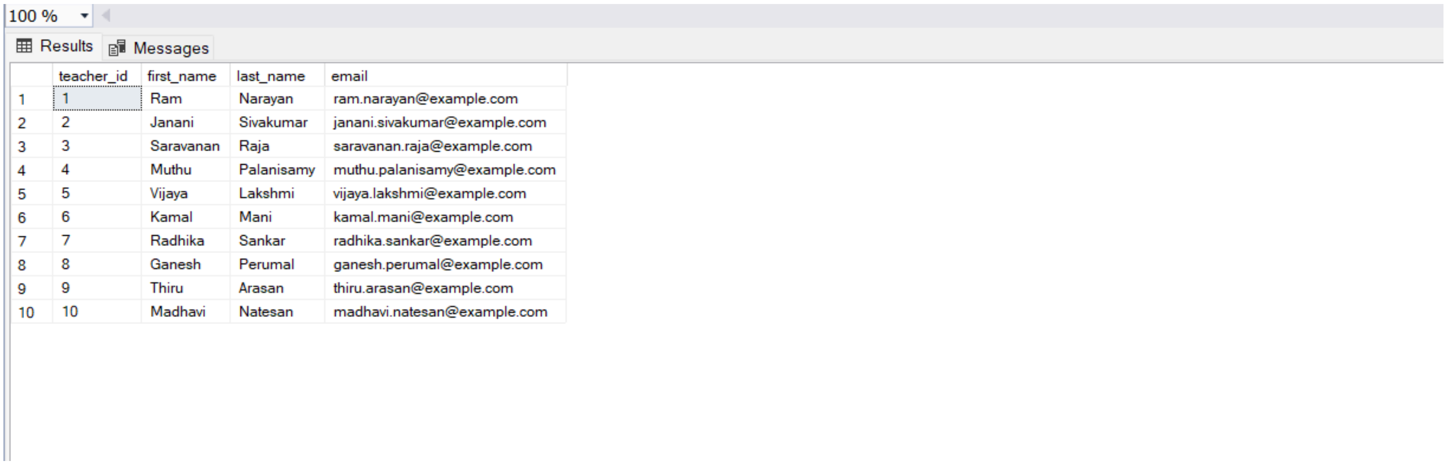
	enrollment_id	student_id	course_id	enrollment_date
1	4	1	2	2024-01-15
2	5	2	3	2024-02-12
3	6	3	4	2024-03-10
4	7	4	5	2024-04-08
5	8	5	6	2024-05-05
6	9	6	7	2024-06-20
7	10	7	8	2024-07-15
8	11	8	9	2024-08-22
9	12	9	10	2024-09-17
10	13	10	11	2024-10-05

iv. Teacher

Query:



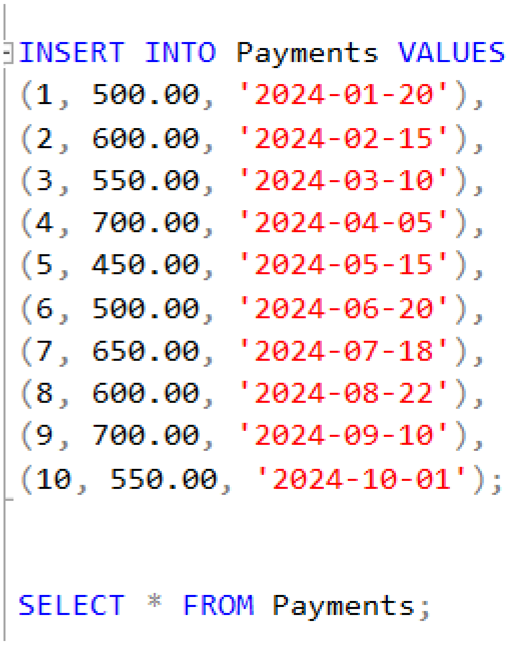
```
INSERT INTO Teacher (first_name, last_name, email) VALUES
('Ram', 'Narayan', 'ram.narayan@example.com'),
('Janani', 'Sivakumar', 'janani.sivakumar@example.com'),
('Saravanan', 'Raja', 'saravanan.raja@example.com'),
('Muthu', 'Palanisamy', 'muthu.palanisamy@example.com'),
('Vijaya', 'Lakshmi', 'vijaya.lakshmi@example.com'),
('Kamal', 'Mani', 'kamal.mani@example.com'),
('Radhika', 'Sankar', 'radhika.sankar@example.com'),
('Ganesh', 'Perumal', 'ganesh.perumal@example.com'),
('Thiru', 'Arasan', 'thiru.arasan@example.com'),
('Madhavi', 'Natesan', 'madhavi.natesan@example.com');

select * from Teacher;
```



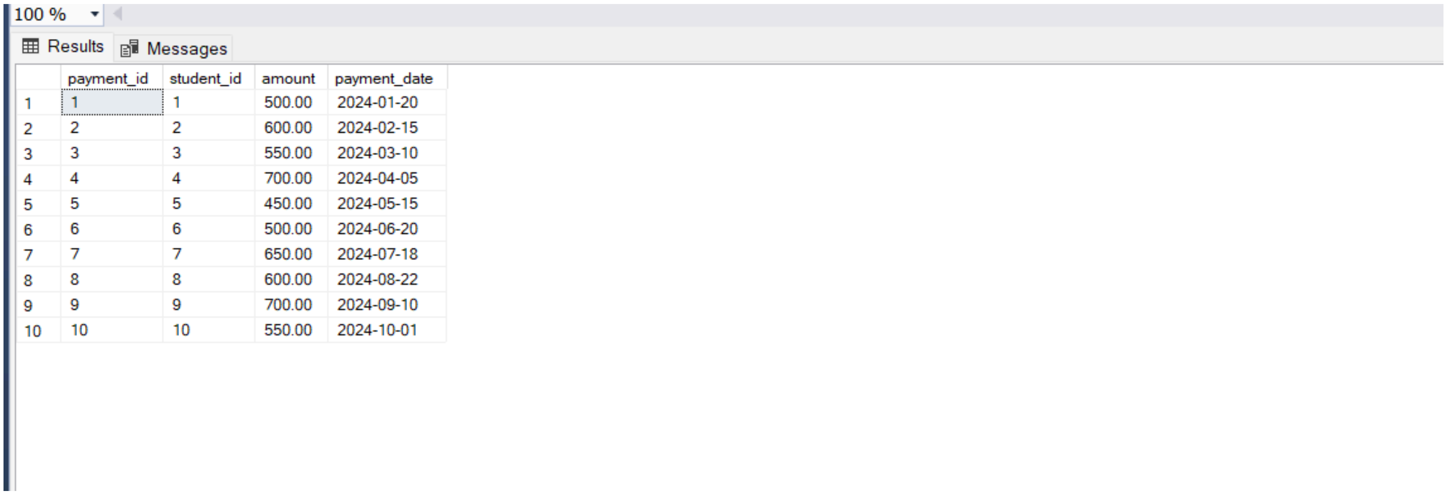
	teacher_id	first_name	last_name	email
1	1	Ram	Narayan	ram.narayan@example.com
2	2	Janani	Sivakumar	janani.sivakumar@example.com
3	3	Saravanan	Raja	saravanan.raja@example.com
4	4	Muthu	Palanisamy	muthu.palanisamy@example.com
5	5	Vijaya	Lakshmi	vijaya.lakshmi@example.com
6	6	Kamal	Mani	kamal.mani@example.com
7	7	Radhika	Sankar	radhika.sankar@example.com
8	8	Ganesh	Perumal	ganesh.perumal@example.com
9	9	Thiru	Arasan	thiru.arasan@example.com
10	10	Madhavi	Natesan	madhavi.natesan@example.com

v. Payments

Query:



```
INSERT INTO Payments VALUES
(1, 500.00, '2024-01-20'),
(2, 600.00, '2024-02-15'),
(3, 550.00, '2024-03-10'),
(4, 700.00, '2024-04-05'),
(5, 450.00, '2024-05-15'),
(6, 500.00, '2024-06-20'),
(7, 650.00, '2024-07-18'),
(8, 600.00, '2024-08-22'),
(9, 700.00, '2024-09-10'),
(10, 550.00, '2024-10-01');


SELECT * FROM Payments;
```



	payment_id	student_id	amount	payment_date
1	1	1	500.00	2024-01-20
2	2	2	600.00	2024-02-15
3	3	3	550.00	2024-03-10
4	4	4	700.00	2024-04-05
5	5	5	450.00	2024-05-15
6	6	6	500.00	2024-06-20
7	7	7	650.00	2024-07-18
8	8	8	600.00	2024-08-22
9	9	9	700.00	2024-09-10
10	10	10	550.00	2024-10-01

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

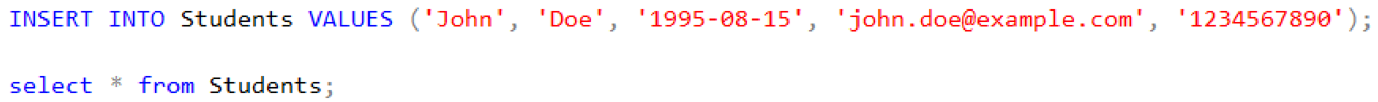
b. Last Name: Doe

c. Date of Birth: 1995-08-15

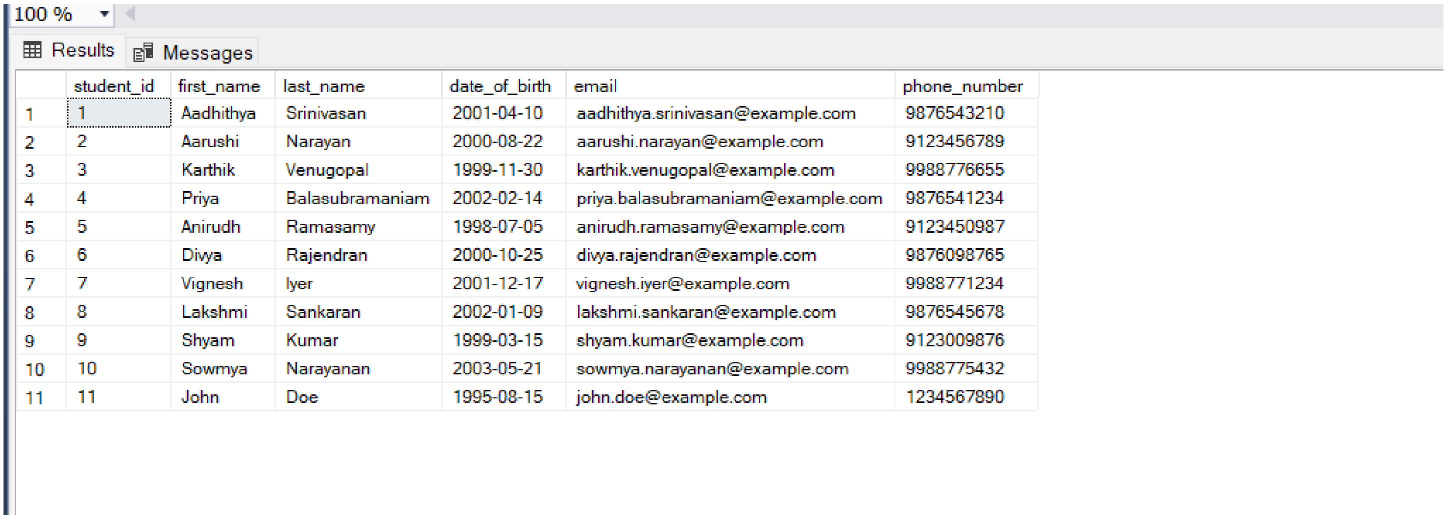
d. Email: john.doe@example.com

e. Phone Number: 1234567890

Query:



```
INSERT INTO Students VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');

select * from Students;
```

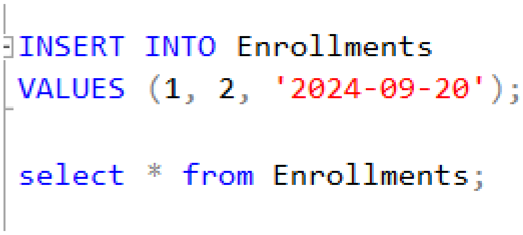


	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	Aadhithya	Srinivasan	2001-04-10	aadhithya.srinivasan@example.com	9876543210
2	2	Aarushi	Narayan	2000-08-22	aarushi.narayan@example.com	9123456789
3	3	Karthik	Venugopal	1999-11-30	karthik.venugopal@example.com	9988776655
4	4	Priya	Balasubramaniam	2002-02-14	priya.balasubramaniam@example.com	9876541234
5	5	Anirudh	Ramasamy	1998-07-05	anirudh.ramasamy@example.com	9123450987
6	6	Divya	Rajendran	2000-10-25	divya.rajendran@example.com	9876098765
7	7	Vignesh	Iyer	2001-12-17	vignesh.iyer@example.com	9988771234
8	8	Lakshmi	Sankaran	2002-01-09	lakshmi.sankaran@example.com	9876545678
9	9	Shyam	Kumar	1999-03-15	shyam.kumar@example.com	9123009876
10	10	Sowmya	Narayanan	2003-05-21	sowmya.narayanan@example.com	9988775432
11	11	John	Doe	1995-08-15	john.doe@example.com	1234567890

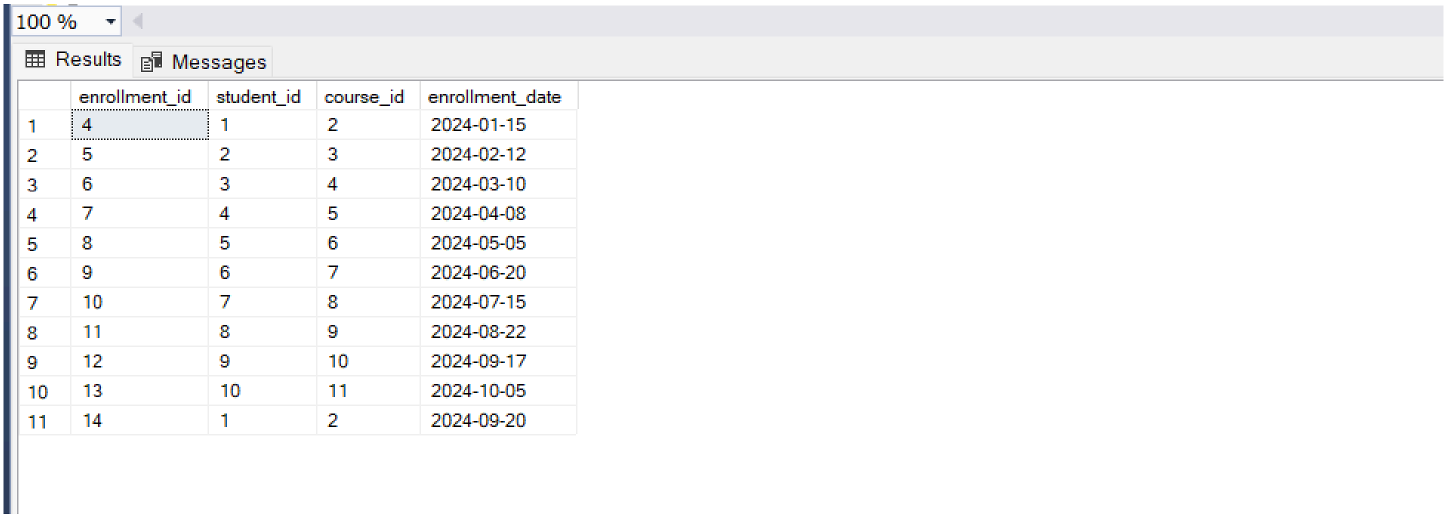
As a 11 th student, the given details were added to the table.

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.

Query:



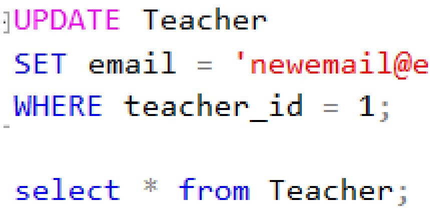
```
INSERT INTO Enrollments
VALUES (1, 2, '2024-09-20');

select * from Enrollments;
```



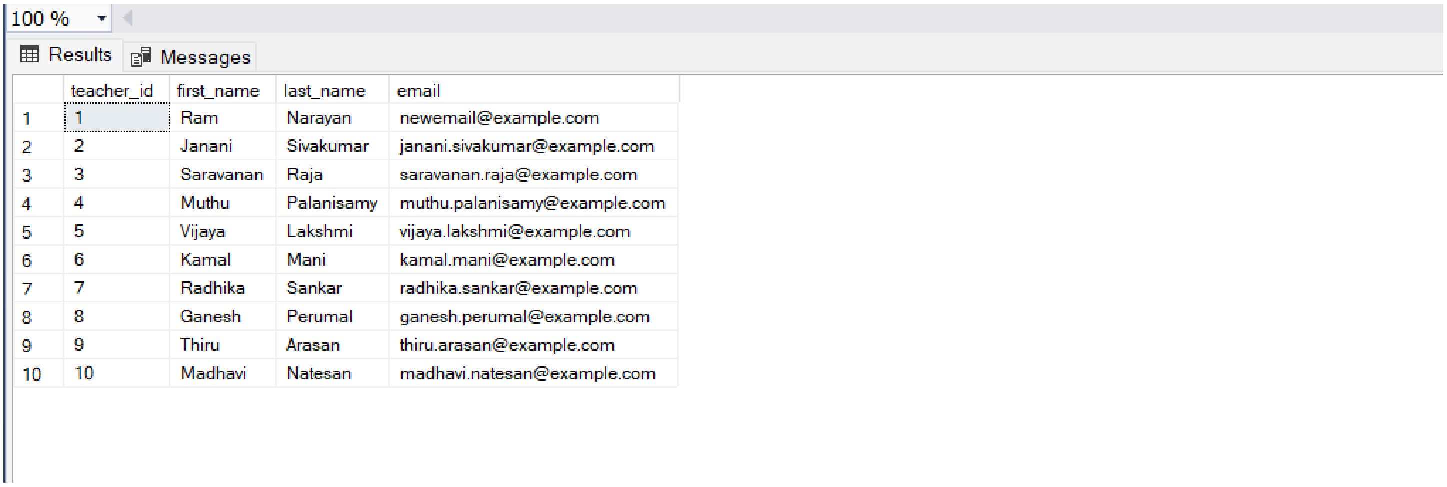
	enrollment_id	student_id	course_id	enrollment_date
1	4	1	2	2024-01-15
2	5	2	3	2024-02-12
3	6	3	4	2024-03-10
4	7	4	5	2024-04-08
5	8	5	6	2024-05-05
6	9	6	7	2024-06-20
7	10	7	8	2024-07-15
8	11	8	9	2024-08-22
9	12	9	10	2024-09-17
10	13	10	11	2024-10-05
11	14	1	2	2024-09-20

3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.

Query:

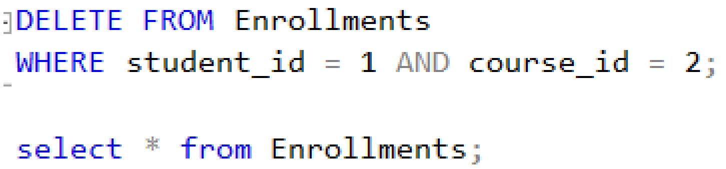


```
UPDATE Teacher
SET email = 'newemail@example.com'
WHERE teacher_id = 1;

select * from Teacher;
```

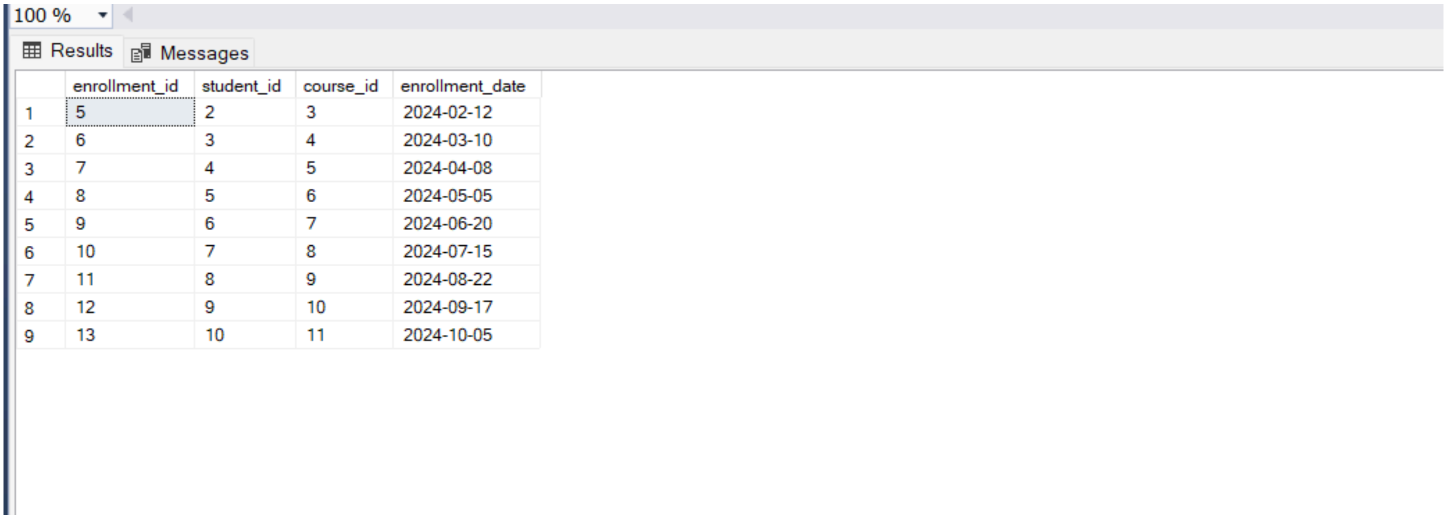


	teacher_id	first_name	last_name	email
1	1	Ram	Narayan	newemail@example.com
2	2	Janani	Sivakumar	janani.sivakumar@example.com
3	3	Saravanan	Raja	saravanan.raja@example.com
4	4	Muthu	Palanisamy	muthu.palanisamy@example.com
5	5	Vijaya	Lakshmi	vijaya.lakshmi@example.com
6	6	Kamal	Mani	kamal.mani@example.com
7	7	Radhika	Sankar	radhika.sankar@example.com
8	8	Ganesh	Perumal	ganesh.perumal@example.com
9	9	Thiru	Arasan	thiru.arasan@example.com
10	10	Madhavi	Natesan	madhavi.natesan@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.
Query:



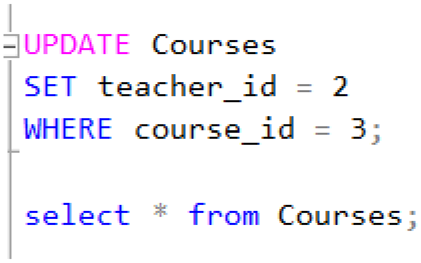
```
DELETE FROM Enrollments
WHERE student_id = 1 AND course_id = 2;

select * from Enrollments;
```



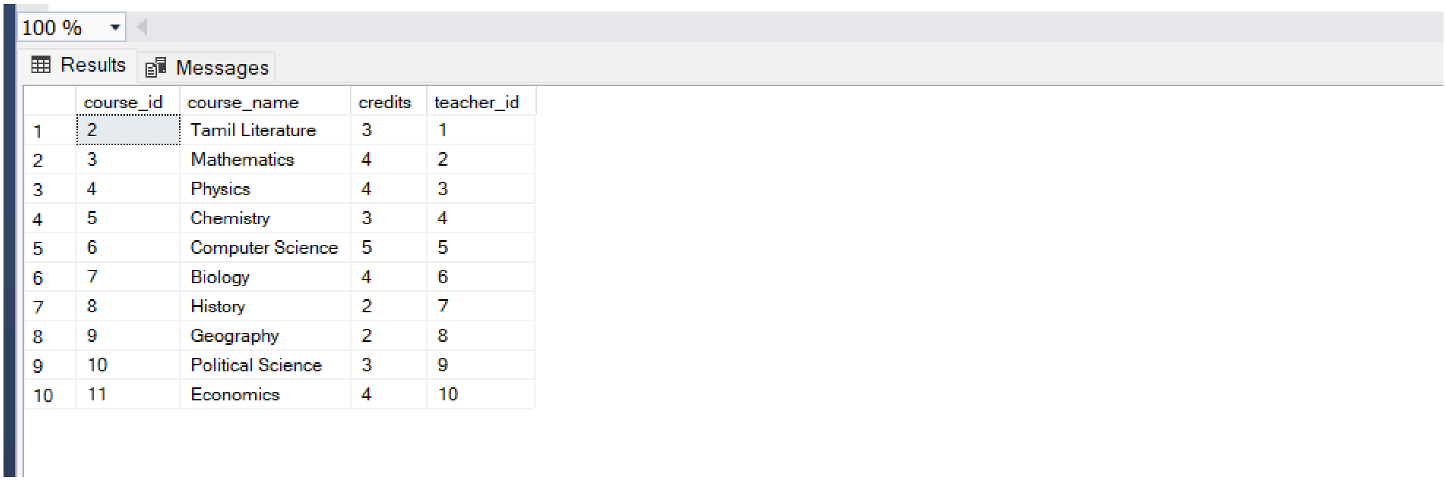
	enrollment_id	student_id	course_id	enrollment_date
1	5	2	3	2024-02-12
2	6	3	4	2024-03-10
3	7	4	5	2024-04-08
4	8	5	6	2024-05-05
5	9	6	7	2024-06-20
6	10	7	8	2024-07-15
7	11	8	9	2024-08-22
8	12	9	10	2024-09-17
9	13	10	11	2024-10-05

5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.

Query:



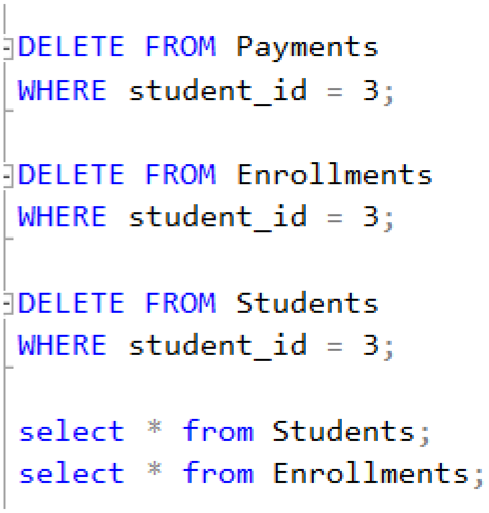
```
UPDATE Courses
SET teacher_id = 2
WHERE course_id = 3;

select * from Courses;
```



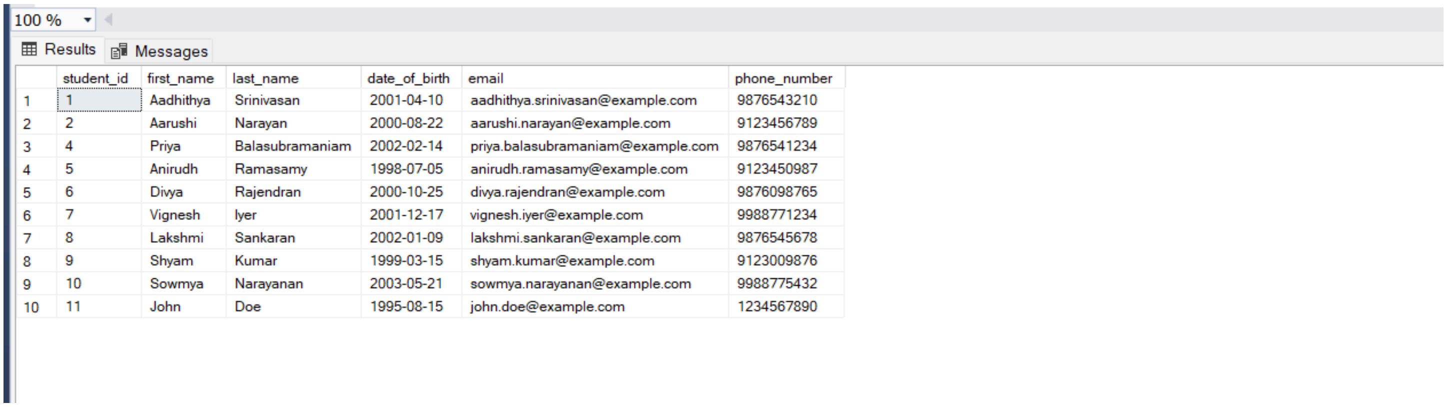
	course_id	course_name	credits	teacher_id
1	2	Tamil Literature	3	1
2	3	Mathematics	4	2
3	4	Physics	4	3
4	5	Chemistry	3	4
5	6	Computer Science	5	5
6	7	Biology	4	6
7	8	History	2	7
8	9	Geography	2	8
9	10	Political Science	3	9
10	11	Economics	4	10

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.

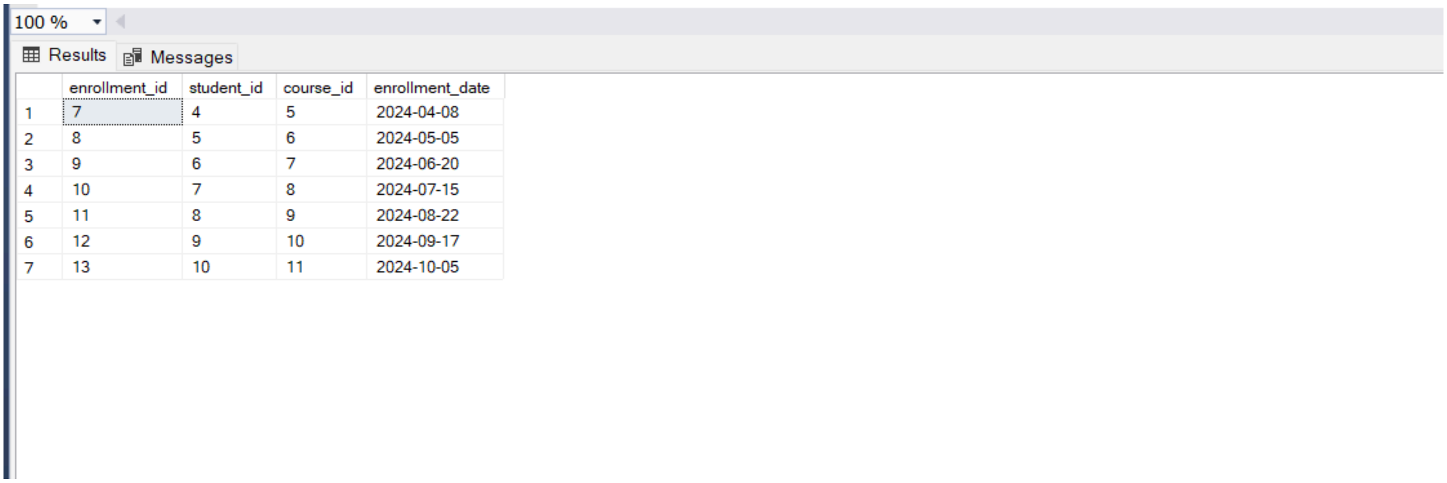
Query:



```
DELETE FROM Payments
WHERE student_id = 3;

DELETE FROM Enrollments
WHERE student_id = 3;

DELETE FROM Students
WHERE student_id = 3;

select * from Students;
select * from Enrollments;
```



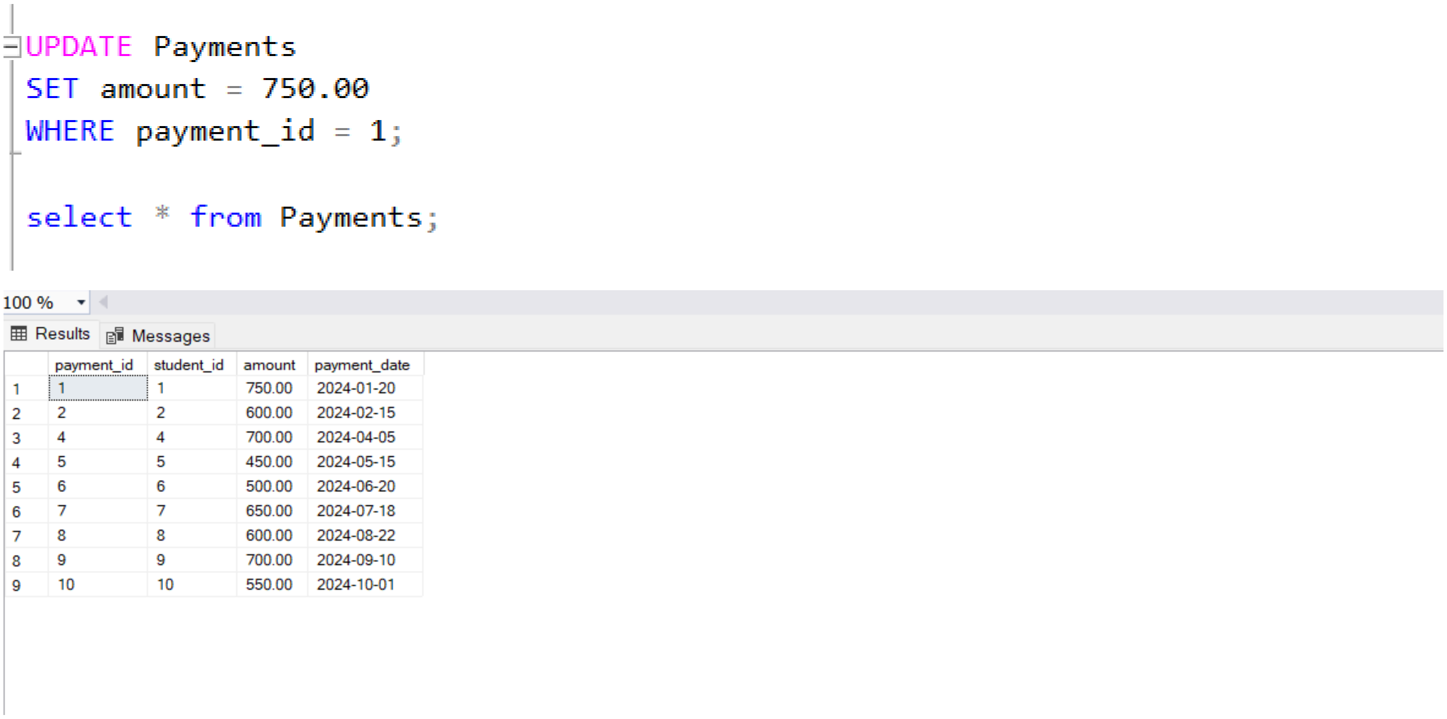
	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	Aadhithya	Srinivasan	2001-04-10	aadhithya.srinivasan@example.com	9876543210
2	2	Aarushi	Narayan	2000-08-22	aarushi.narayan@example.com	9123456789
3	4	Priya	Balasubramaniam	2002-02-14	priya.balasubramaniam@example.com	9876541234
4	5	Anirudh	Ramasamy	1998-07-05	anirudh.ramasamy@example.com	9123450987
5	6	Divya	Rajendran	2000-10-25	divya.rajendran@example.com	9876098765
6	7	Vignesh	Iyer	2001-12-17	vignesh.iyer@example.com	9988771234
7	8	Lakshmi	Sankaran	2002-01-09	lakshmi.sankaran@example.com	9876545678
8	9	Shyam	Kumar	1999-03-15	shyam.kumar@example.com	9123009876
9	10	Sowmya	Narayanan	2003-05-21	sowmya.narayanan@example.com	9988775432
10	11	John	Doe	1995-08-15	john.doe@example.com	1234567890



	enrollment_id	student_id	course_id	enrollment_date
1	7	4	5	2024-04-08
2	8	5	6	2024-05-05
3	9	6	7	2024-06-20
4	10	7	8	2024-07-15
5	11	8	9	2024-08-22
6	12	9	10	2024-09-17
7	13	10	11	2024-10-05

7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount

Query:



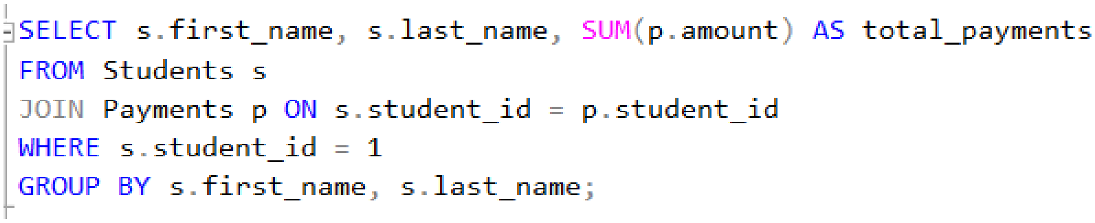
```
UPDATE Payments
SET amount = 750.00
WHERE payment_id = 1;

select * from Payments;
```

	payment_id	student_id	amount	payment_date
1	1	1	750.00	2024-01-20
2	2	2	600.00	2024-02-15
3	4	4	700.00	2024-04-05
4	5	5	450.00	2024-05-15
5	6	6	500.00	2024-06-20
6	7	7	650.00	2024-07-18
7	8	8	600.00	2024-08-22
8	9	9	700.00	2024-09-10
9	10	10	550.00	2024-10-01

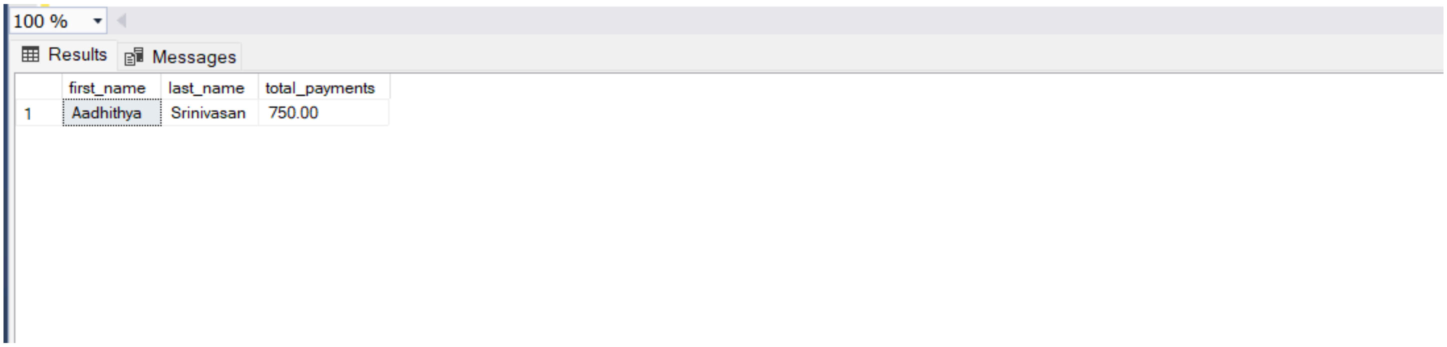
Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

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.

Query:



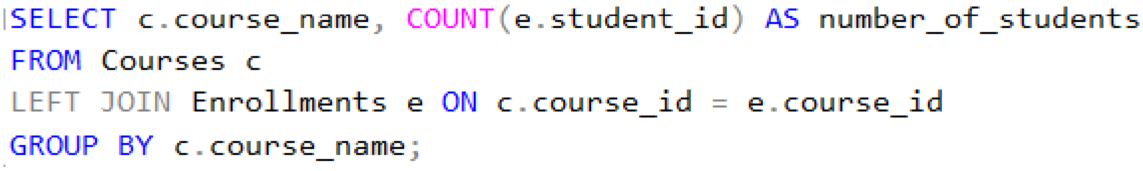
```
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
WHERE s.student_id = 1
GROUP BY s.first_name, s.last_name;
```



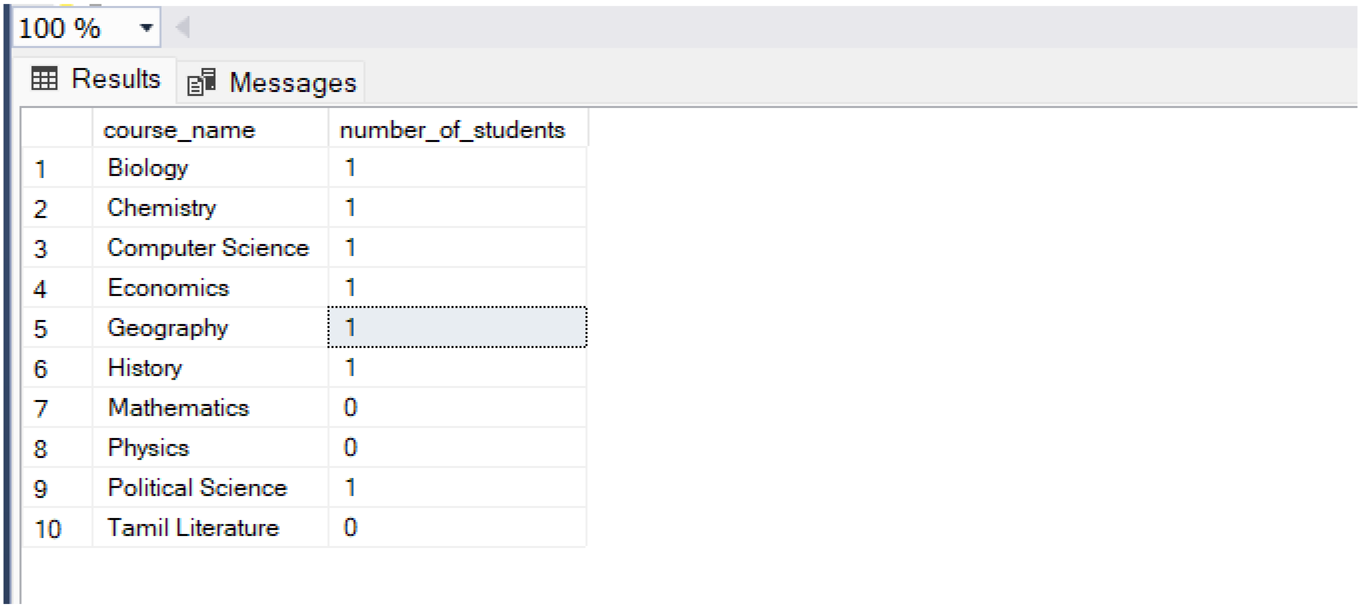
```
   first_name   last_name    total_payments
1  Aadhithya    Srinivasan   750.00
```

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.

Query:



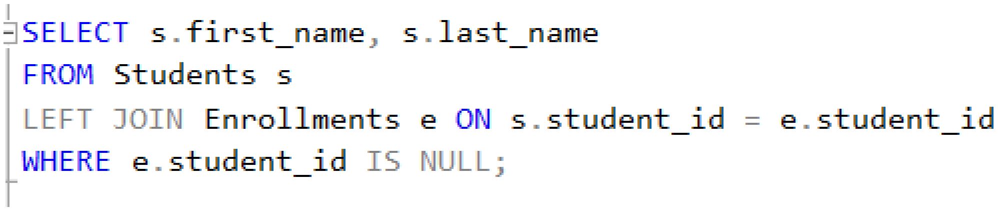
```
SELECT c.course_name, COUNT(e.student_id) AS number_of_students
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_name;
```



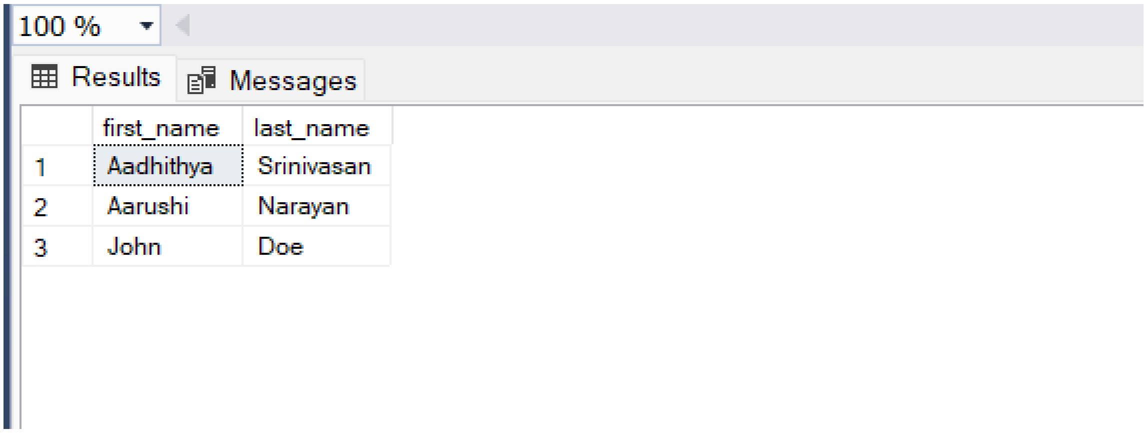
	course_name	number_of_students
1	Biology	1
2	Chemistry	1
3	Computer Science	1
4	Economics	1
5	Geography	1
6	History	1
7	Mathematics	0
8	Physics	0
9	Political Science	1
10	Tamil Literature	0

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.

Query:



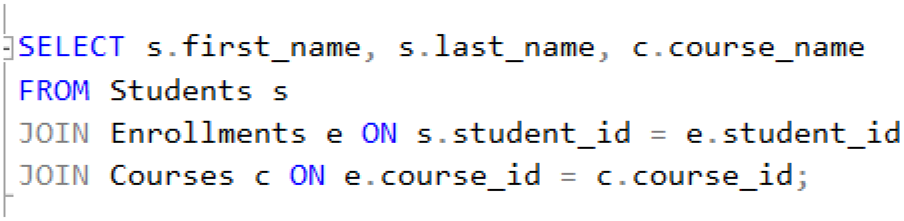
```
SELECT s.first_name, s.last_name
FROM Students s
LEFT JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.student_id IS NULL;
```



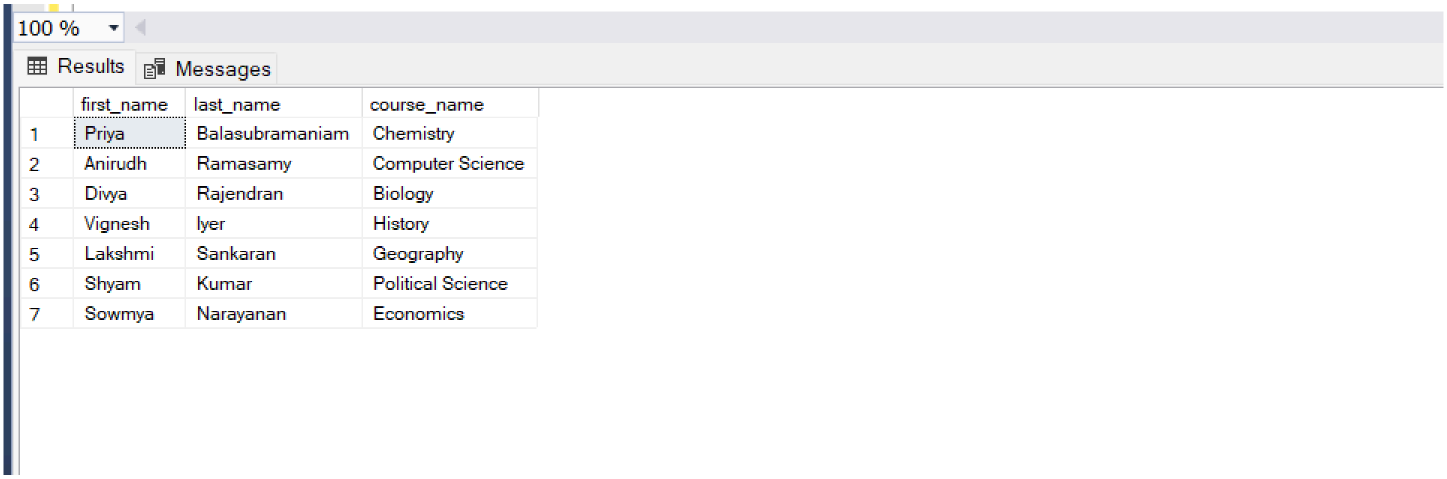
	first_name	last_name
1	Aadhithya	Srinivasan
2	Aarushi	Narayan
3	John	Doe

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.

Query:



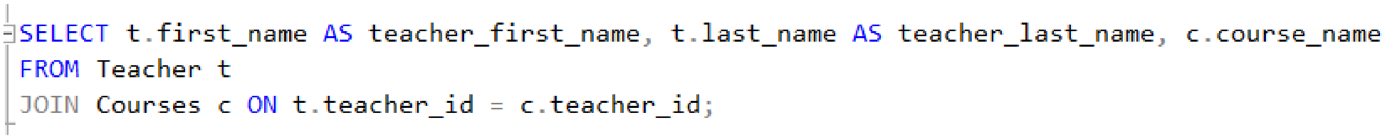
```
SELECT s.first_name, s.last_name, c.course_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;
```



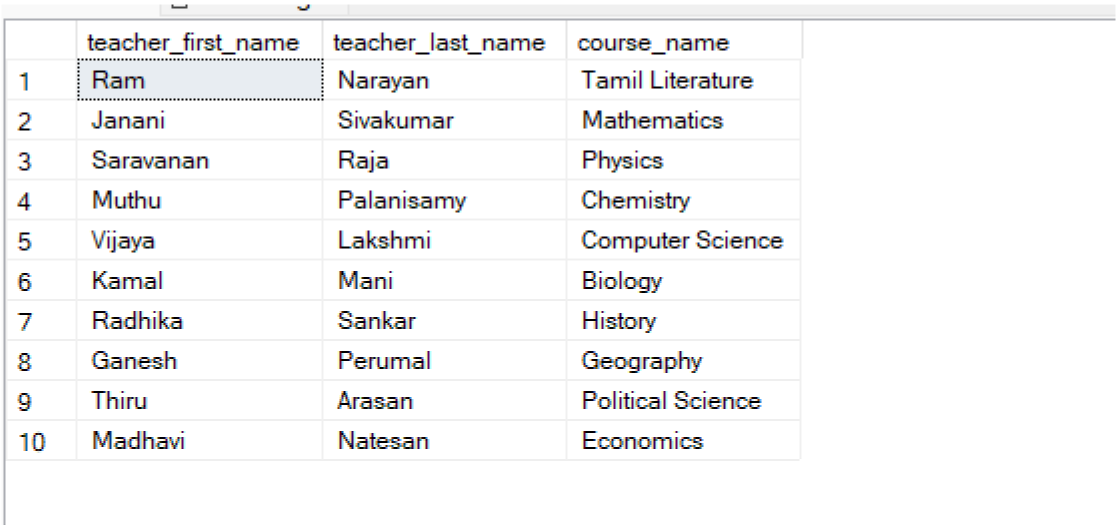
	first_name	last_name	course_name
1	Priya	Balasubramaniam	Chemistry
2	Anirudh	Ramasamy	Computer Science
3	Divya	Rajendran	Biology
4	Vignesh	Iyer	History
5	Lakshmi	Sankaran	Geography
6	Shyam	Kumar	Political Science
7	Sowmya	Narayanan	Economics

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.

Query:



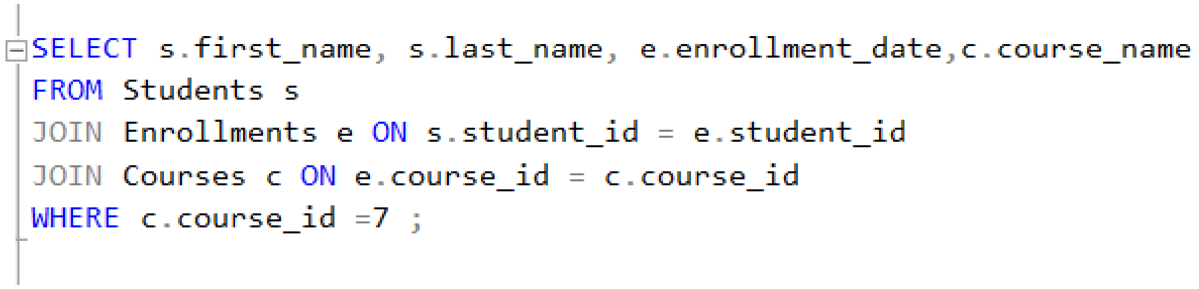
```
SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id;
```



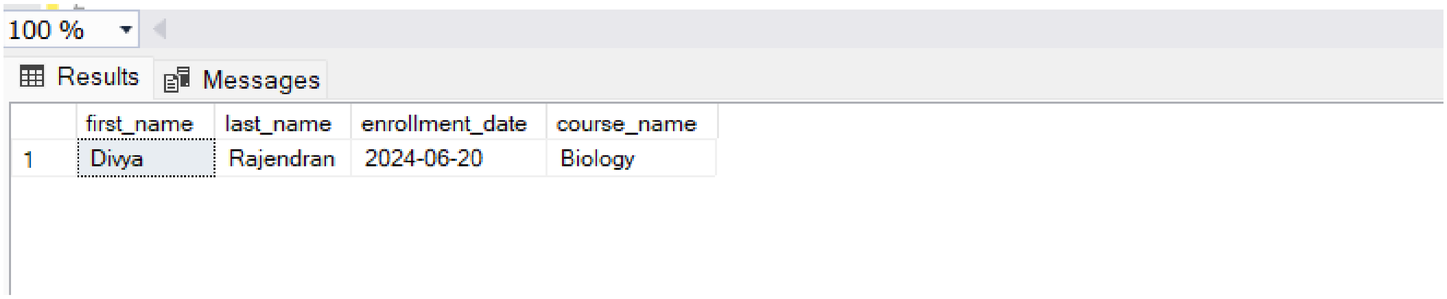
	teacher_first_name	teacher_last_name	course_name
1	Ram	Narayan	Tamil Literature
2	Janani	Sivakumar	Mathematics
3	Saravanan	Raja	Physics
4	Muthu	Palanisamy	Chemistry
5	Vijaya	Lakshmi	Computer Science
6	Kamal	Mani	Biology
7	Radhika	Sankar	History
8	Ganesh	Perumal	Geography
9	Thiru	Arasan	Political Science
10	Madhavi	Natesan	Economics

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.

Query:



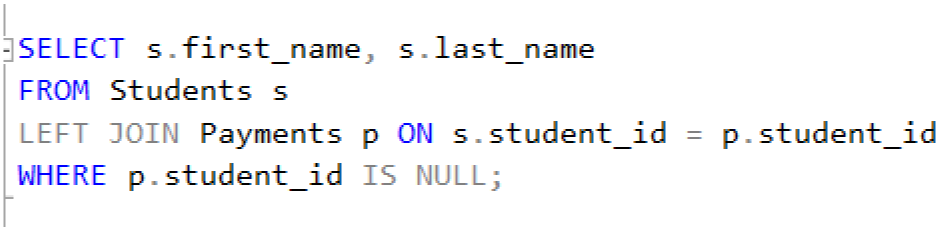
```
SELECT s.first_name, s.last_name, e.enrollment_date,c.course_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE c.course_id =7 ;
```



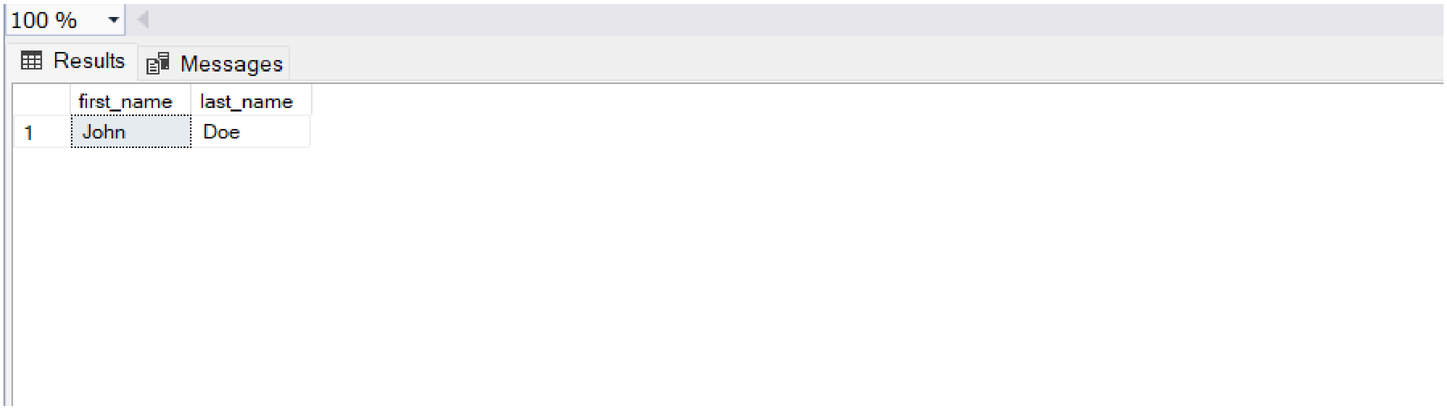
	first_name	last_name	enrollment_date	course_name
1	Divya	Rajendran	2024-06-20	Biology

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.

Query:



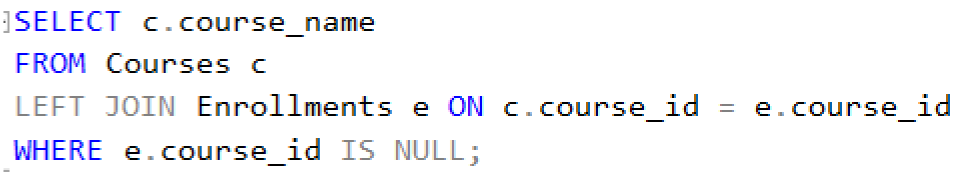
```
SELECT s.first_name, s.last_name
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
WHERE p.student_id IS NULL;
```



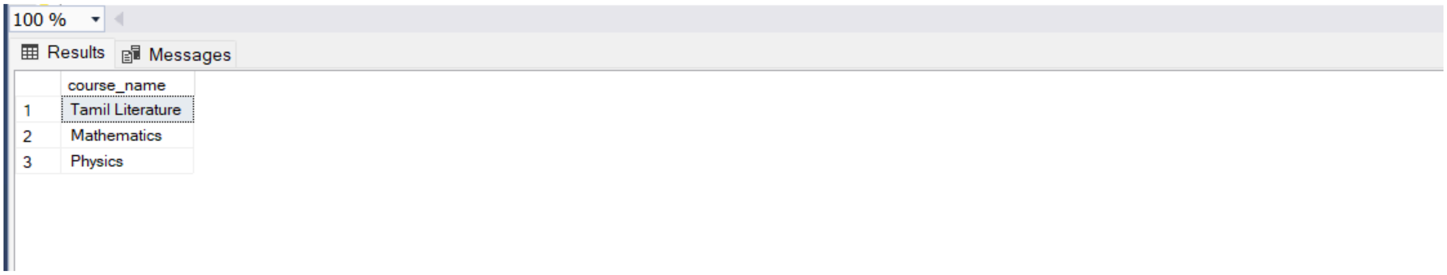
	first_name	last_name
1	John	Doe

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

Query:



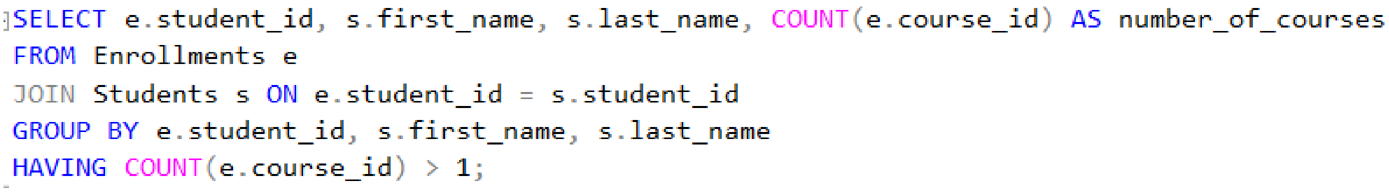
```
SELECT c.course_name
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
WHERE e.course_id IS NULL;
```



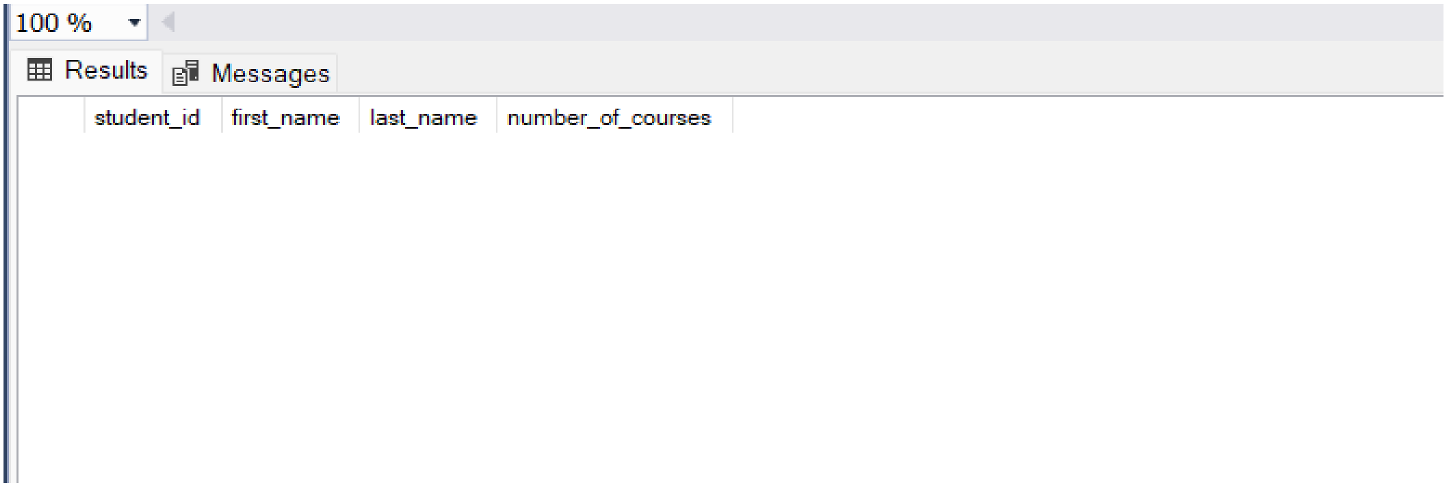
	course_name
1	Tamil Literature
2	Mathematics
3	Physics

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.

Query:



```
SELECT e.student_id, s.first_name, s.last_name, COUNT(e.course_id) AS number_of_courses
FROM Enrollments e
JOIN Students s ON e.student_id = s.student_id
GROUP BY e.student_id, s.first_name, s.last_name
HAVING COUNT(e.course_id) > 1;
```

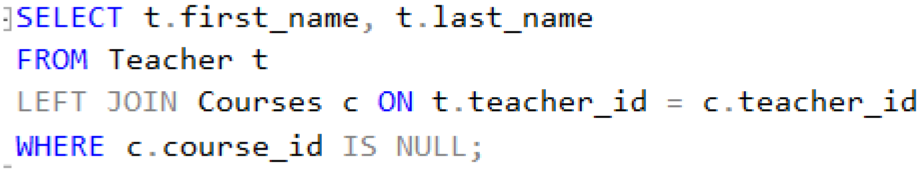


student_id	first_name	last_name	number_of_courses

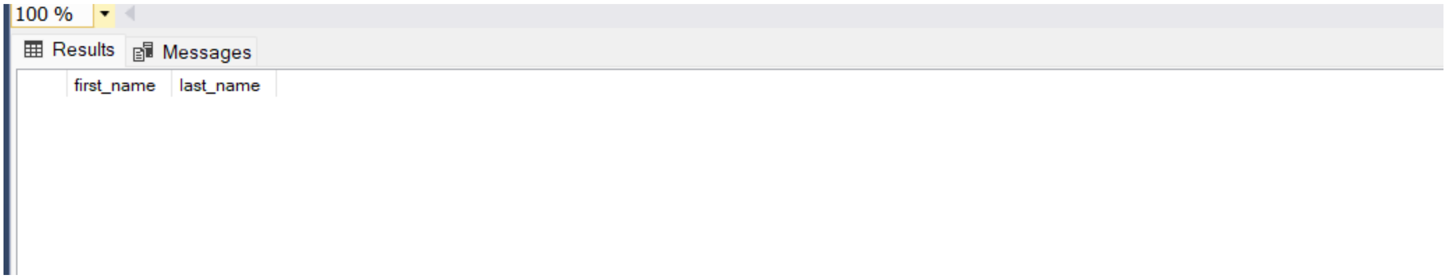
This indicates that no student is enrolled in more than 1 course.

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.

Query:



```
SELECT 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;
```



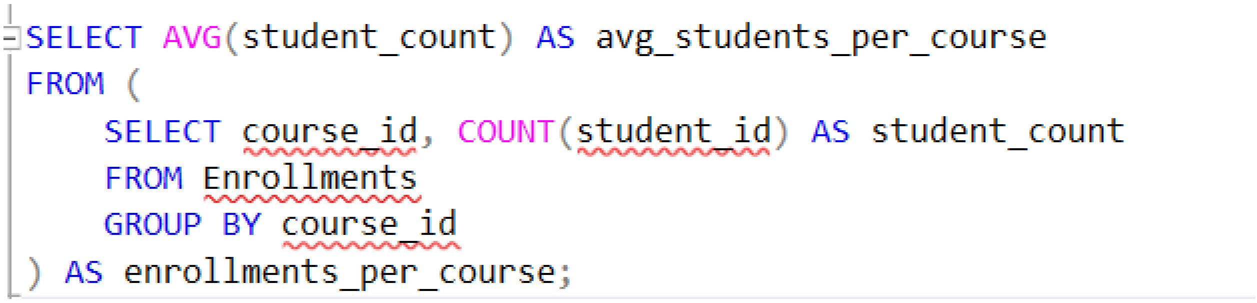
first_name	last_name

This indicates that all teachers have been assigned a course.

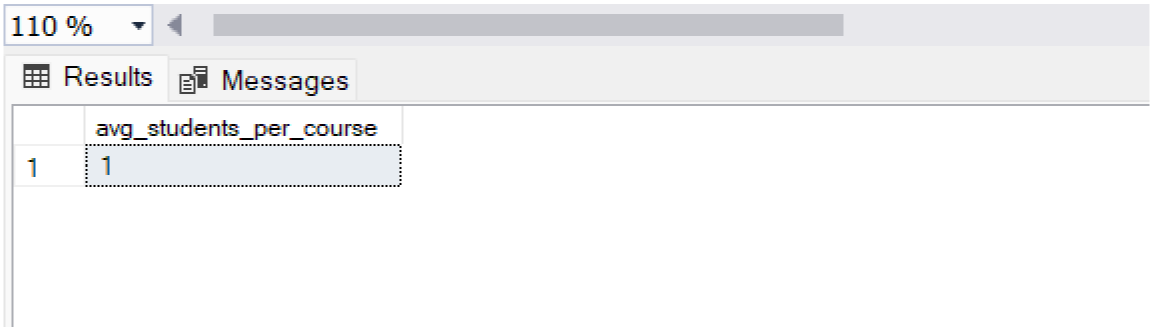
Task 4. Subquery and its type:

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

Query:



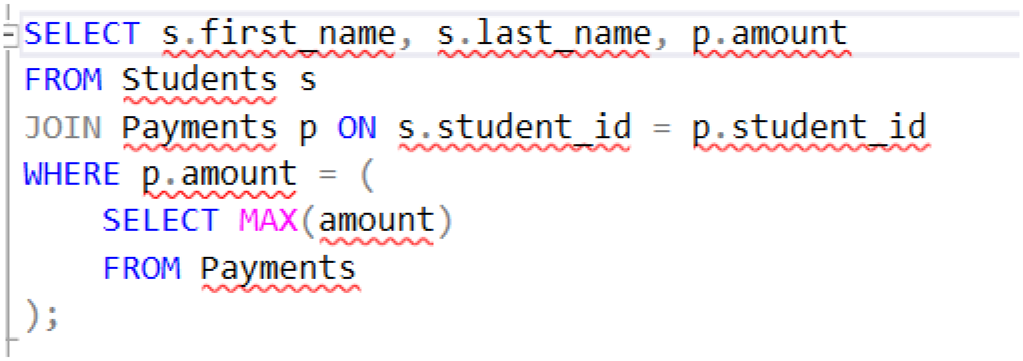
```
SELECT AVG(student_count) AS avg_students_per_course
FROM (
    SELECT course_id, COUNT(student_id) AS student_count
    FROM Enrollments
    GROUP BY course_id
) AS enrollments_per_course;
```



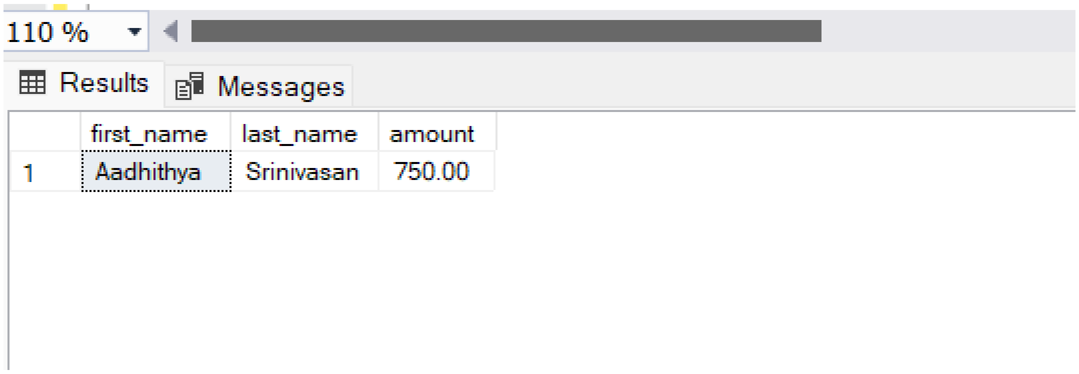
```
   avg_students_per_course
1  1
```

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.

Query:



```
SELECT 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
);
```

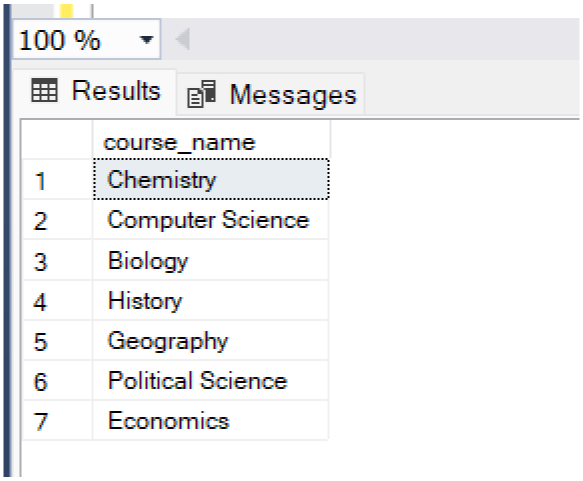


	first_name	last_name	amount
1	Aadhithya	Srinivasan	750.00

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.

Query:

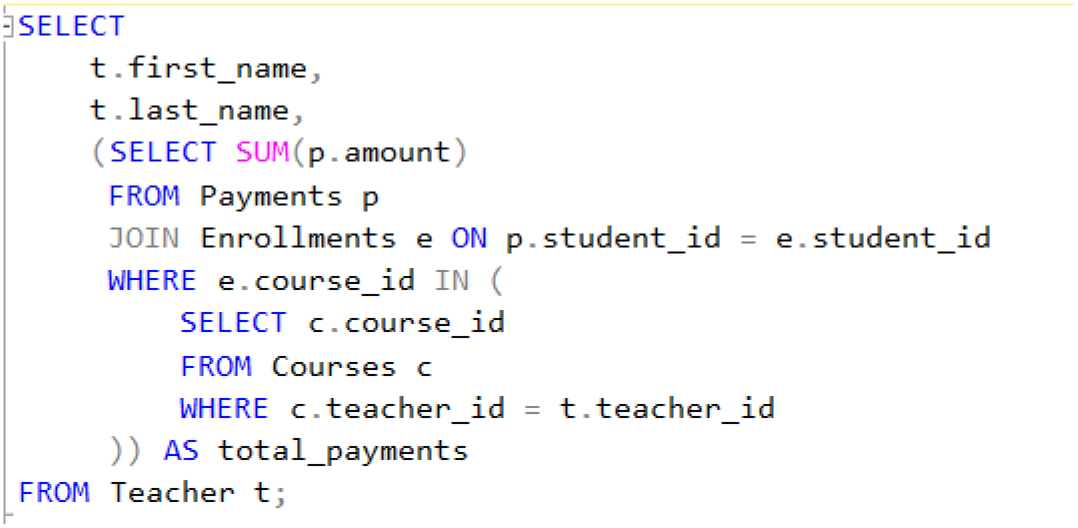
```
SELECT c.course_name
FROM Courses c
WHERE c.course_id IN (
    SELECT course_id
    FROM Enrollments
    GROUP BY course_id
    HAVING COUNT(student_id) = (
        SELECT MAX(enrollment_count)
        FROM (
            SELECT COUNT(student_id) AS enrollment_count
            FROM Enrollments
            GROUP BY course_id
        ) AS counts
    )
);
```



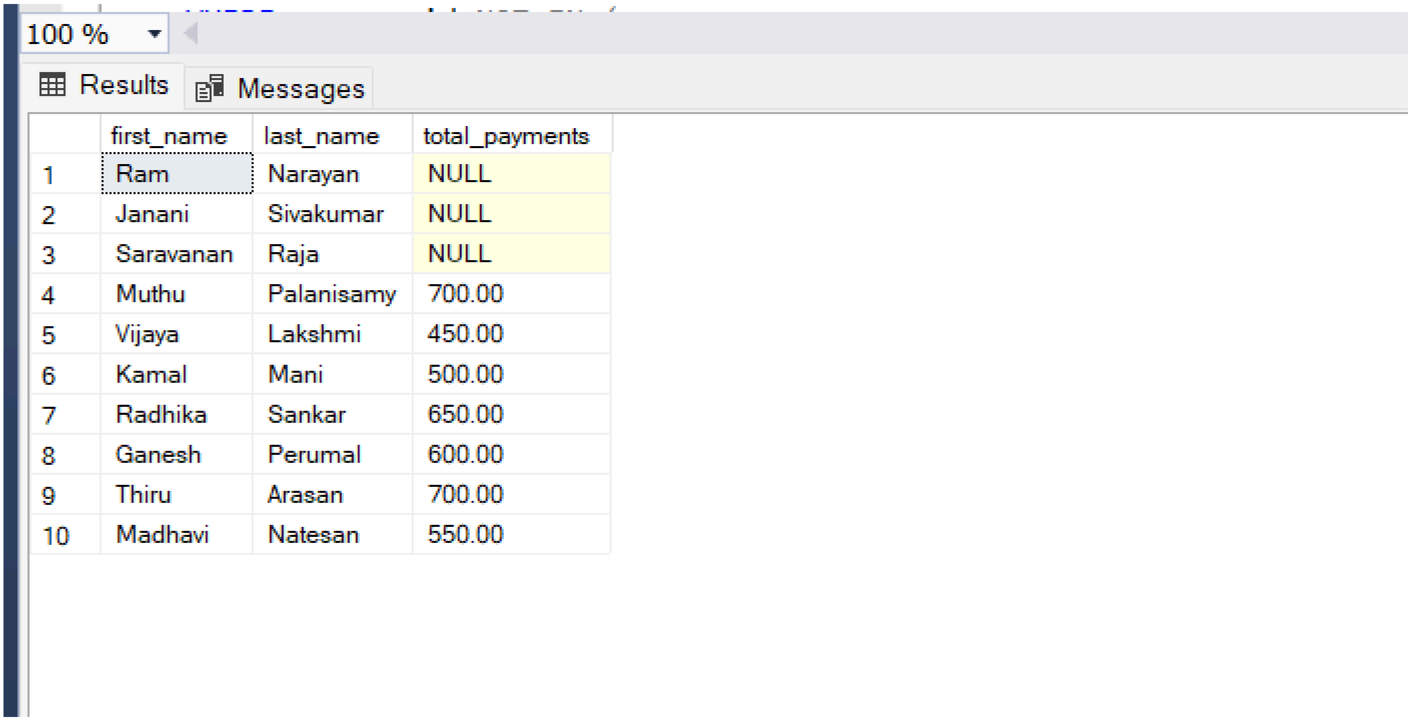
	course_name
1	Chemistry
2	Computer Science
3	Biology
4	History
5	Geography
6	Political Science
7	Economics

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

Query:



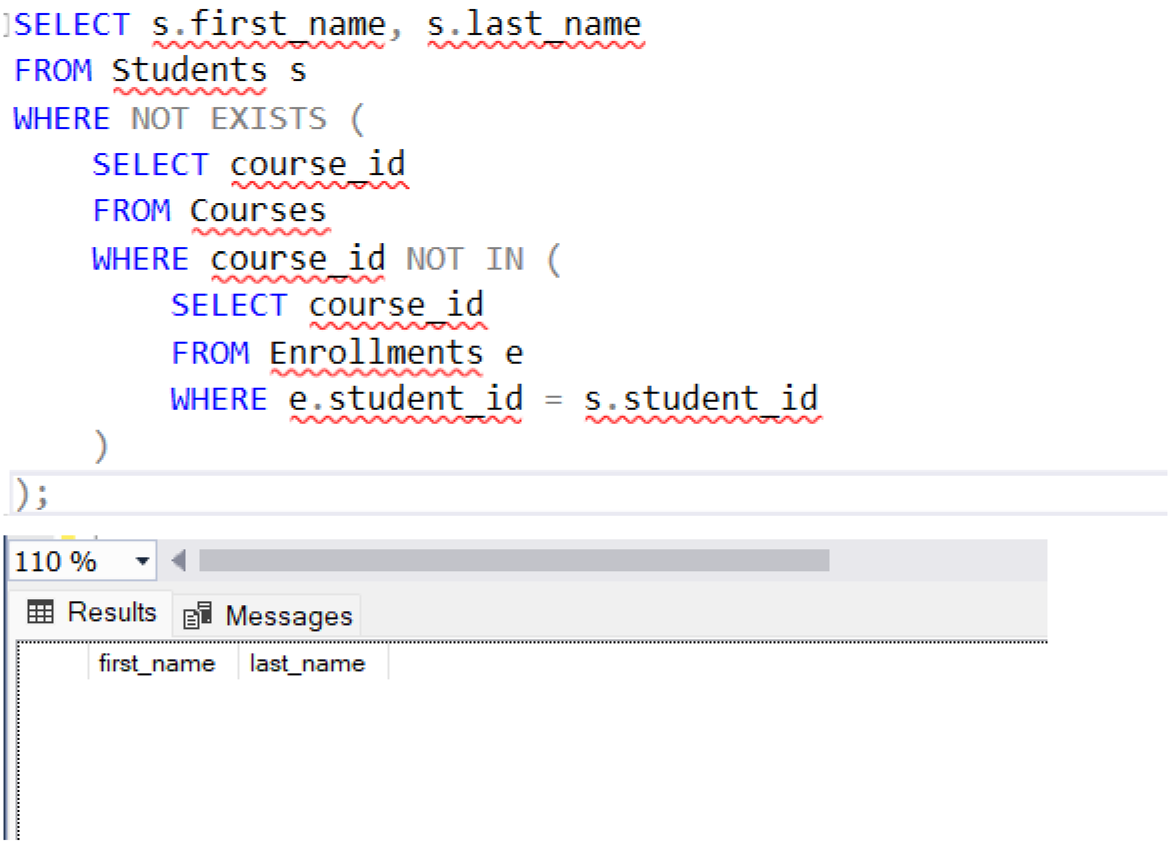
```sql
SELECT
    t.first_name,
    t.last_name,
    (SELECT SUM(p.amount)
     FROM Payments p
     JOIN Enrollments e ON p.student_id = e.student_id
     WHERE e.course_id IN (
         SELECT c.course_id
         FROM Courses c
         WHERE c.teacher_id = t.teacher_id
     )) AS total_payments
FROM Teacher t;
```



	first_name	last_name	total_payments
1	Ram	Narayan	NULL
2	Janani	Sivakumar	NULL
3	Saravanan	Raja	NULL
4	Muthu	Palanisamy	700.00
5	Vijaya	Lakshmi	450.00
6	Kamal	Mani	500.00
7	Radhika	Sankar	650.00
8	Ganesh	Perumal	600.00
9	Thiru	Arasan	700.00
10	Madhavi	Natesan	550.00

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

Query:



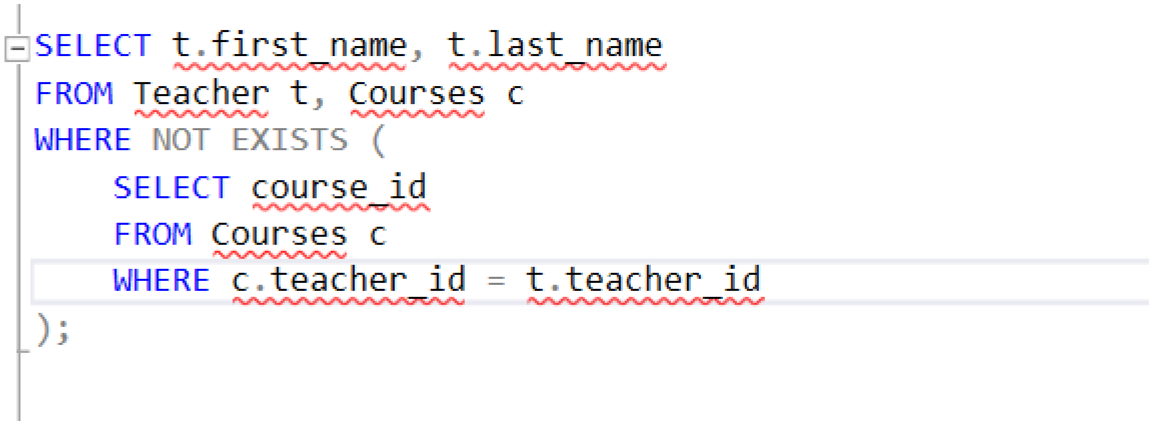
```
SELECT s.first_name, s.last_name
FROM Students s
WHERE NOT EXISTS (
    SELECT course_id
    FROM Courses
    WHERE course_id NOT IN (
        SELECT course_id
        FROM Enrollments e
        WHERE e.student_id = s.student_id
    )
);
```

```
first_name  last_name
```

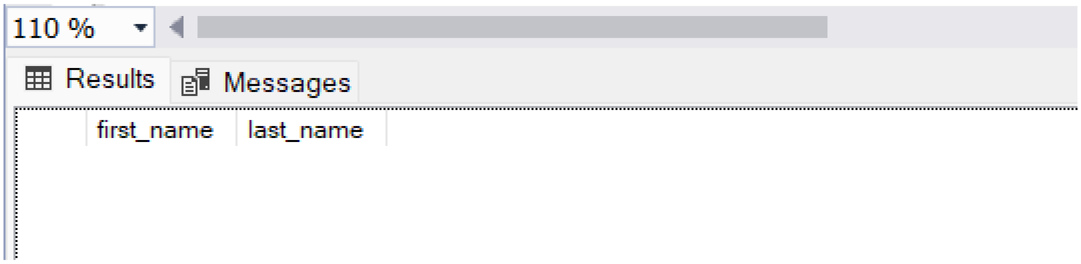
This shows that there is no student who is enrolled in all the courses.

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

Query:



```
SELECT t.first_name, t.last_name
FROM Teacher t, Courses c
WHERE NOT EXISTS (
    SELECT course_id
    FROM Courses c
    WHERE c.teacher_id = t.teacher_id
);
```

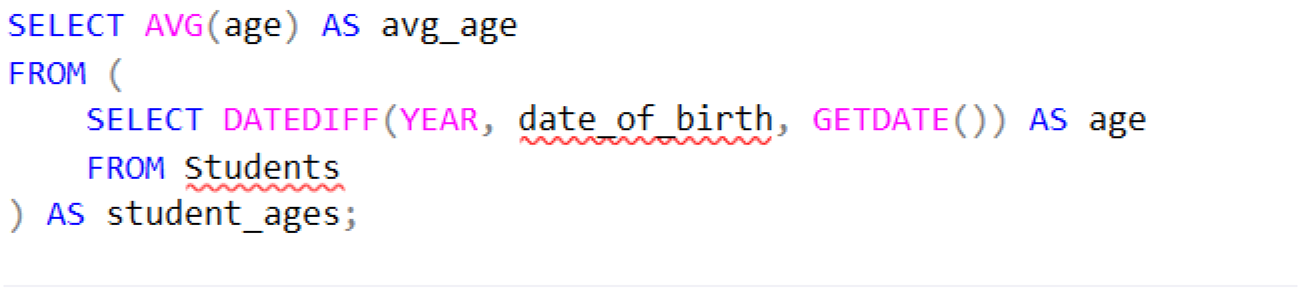


```
first_name   last_name
```

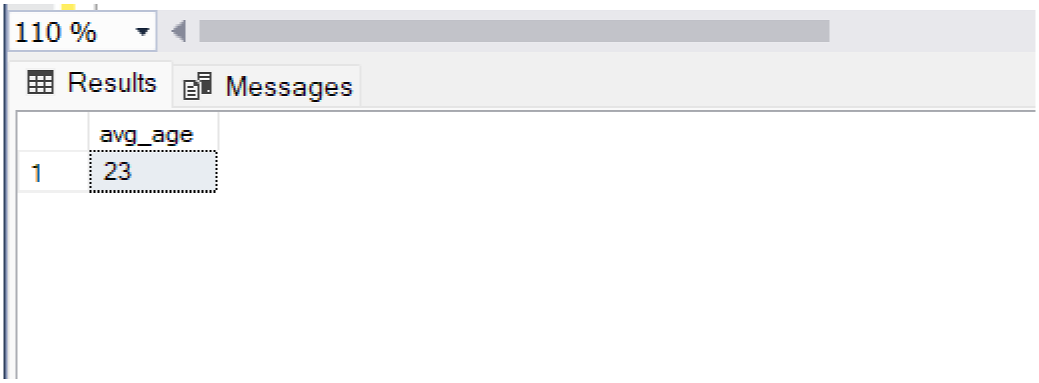
This indicates that all teachers have been assigned a course.

7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.

Query:



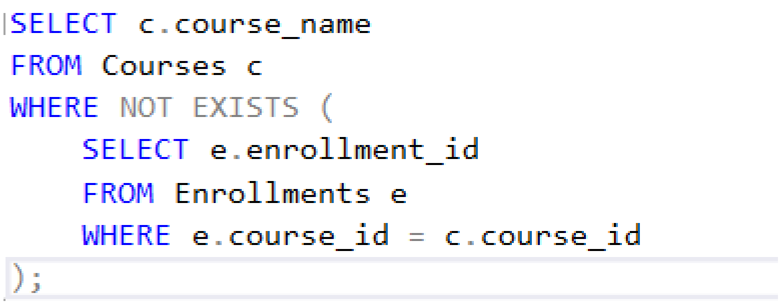
```
SELECT AVG(age) AS avg_age
FROM (
    SELECT DATEDIFF(YEAR, date_of_birth, GETDATE()) AS age
    FROM Students
) AS student_ages;
```



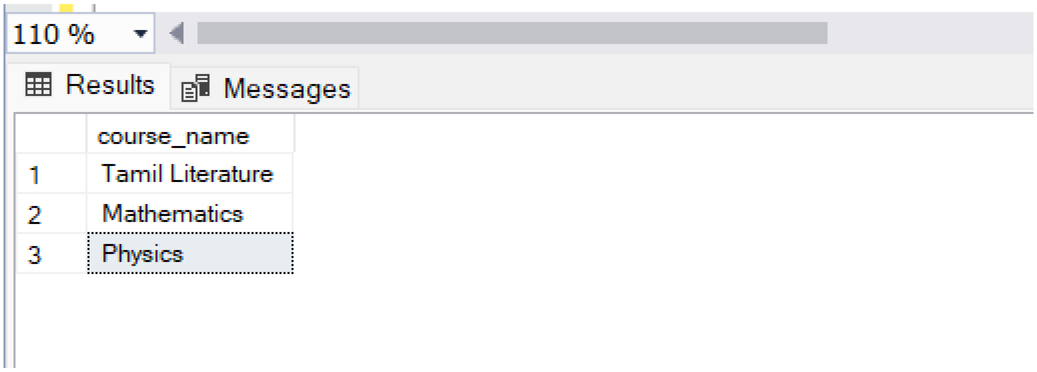
```
     avg_age
1    23
```

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

Query:



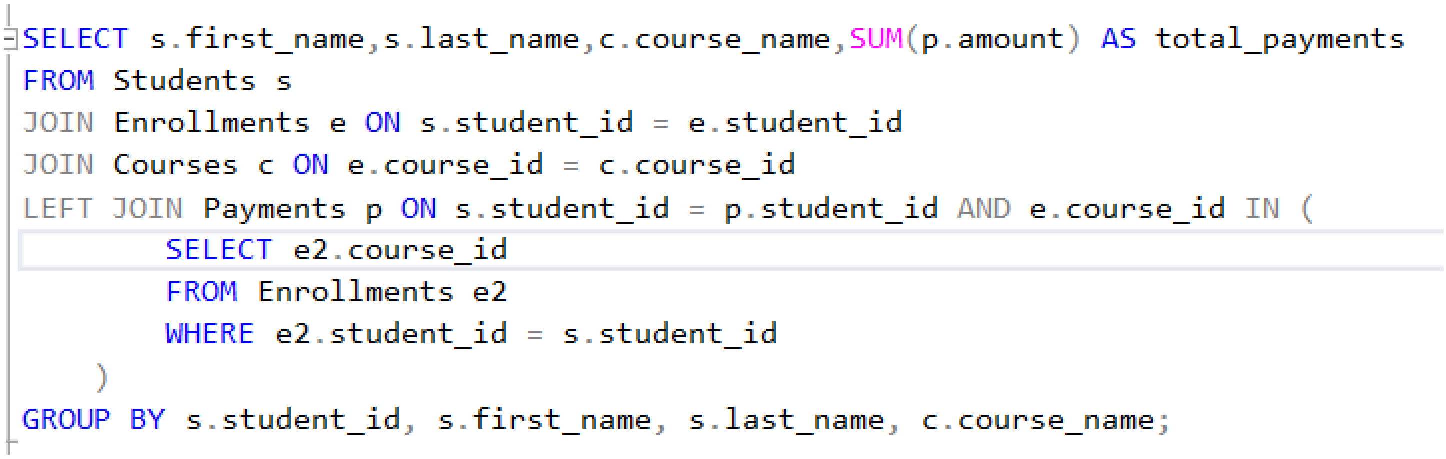
```
SELECT c.course_name
FROM Courses c
WHERE NOT EXISTS (
    SELECT e.enrollment_id
    FROM Enrollments e
    WHERE e.course_id = c.course_id
);
```



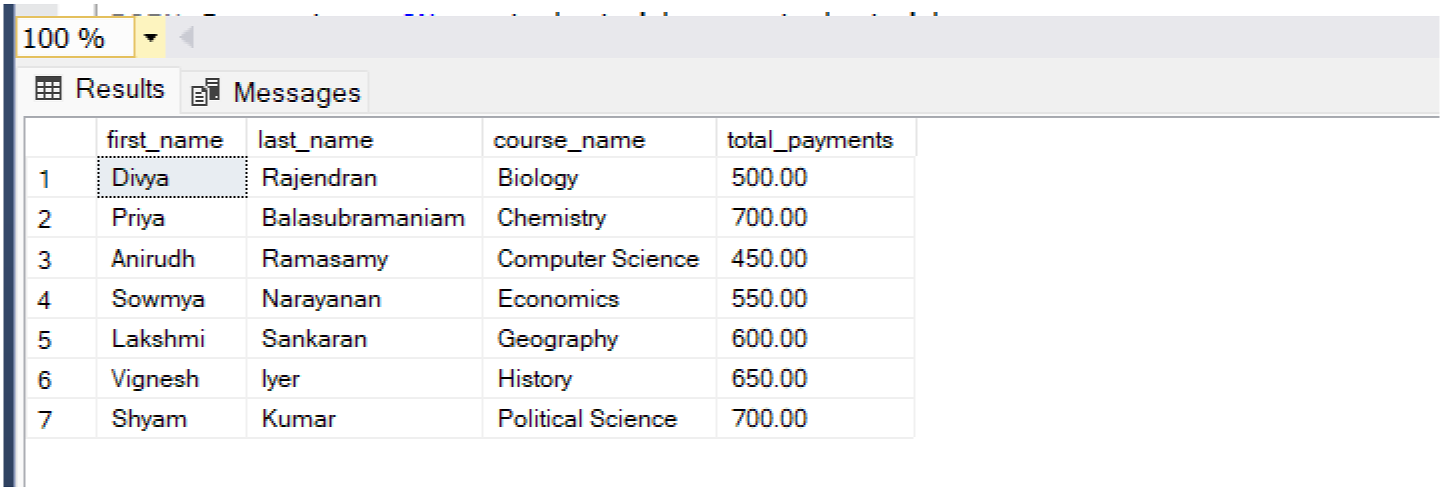
	course_name
1	Tamil Literature
2	Mathematics
3	Physics

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

Query:



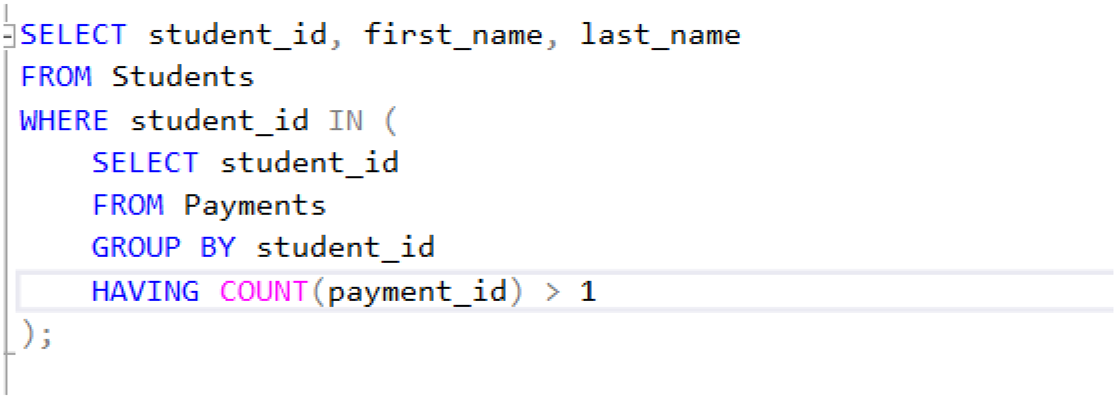
```
SELECT s.first_name,s.last_name,c.course_name,SUM(p.amount) AS total_payments
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
LEFT JOIN Payments p ON s.student_id = p.student_id AND e.course_id IN (
        SELECT e2.course_id
        FROM Enrollments e2
        WHERE e2.student_id = s.student_id
    )
GROUP BY s.student_id, s.first_name, s.last_name, c.course_name;
```



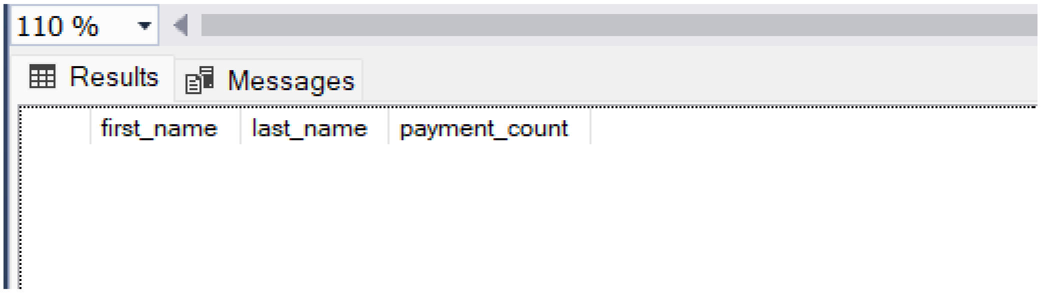
	first_name	last_name	course_name	total_payments
1	Divya	Rajendran	Biology	500.00
2	Priya	Balasubramaniam	Chemistry	700.00
3	Anirudh	Ramasamy	Computer Science	450.00
4	Sowmya	Narayanan	Economics	550.00
5	Lakshmi	Sankaran	Geography	600.00
6	Vignesh	Iyer	History	650.00
7	Shyam	Kumar	Political Science	700.00

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.

Query:



```sql
SELECT student_id, first_name, last_name
FROM Students
WHERE student_id IN (
    SELECT student_id
    FROM Payments
    GROUP BY student_id
    HAVING COUNT(payment_id) > 1
);
```

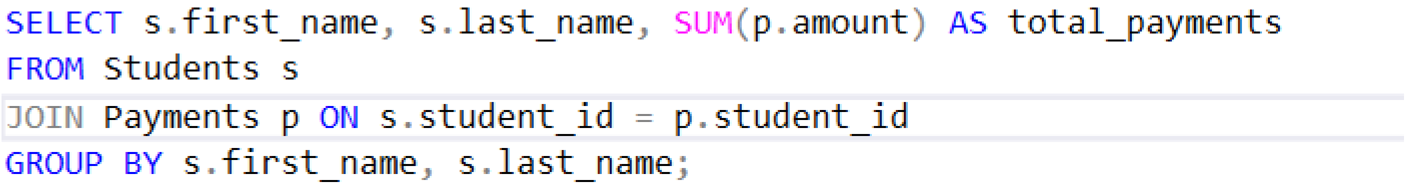


	first_name	last_name	payment_count

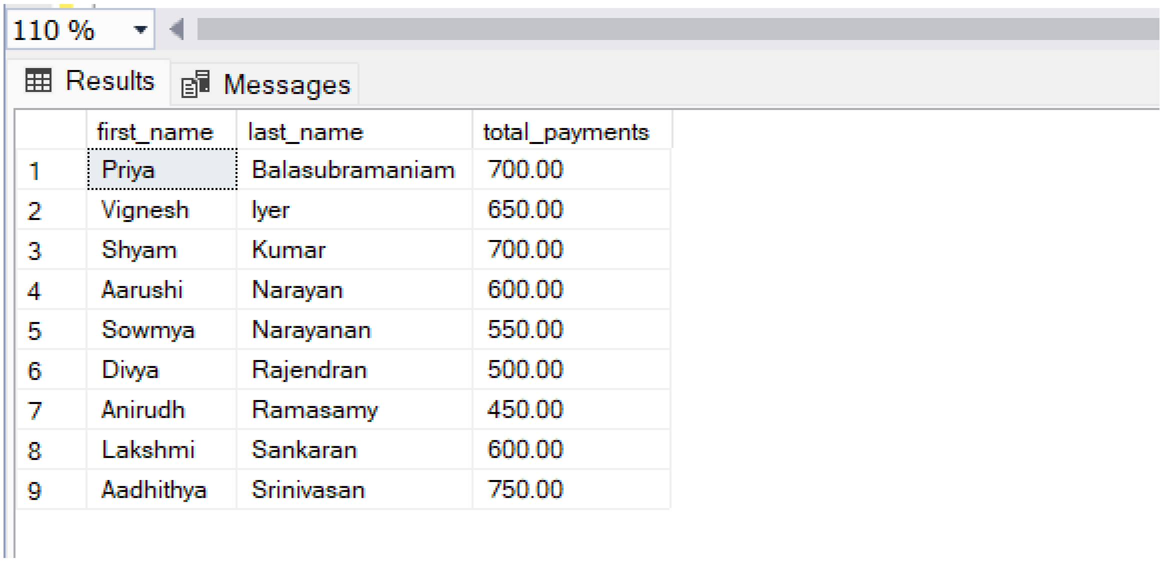
No student has made a payment more than once.

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.

Query:



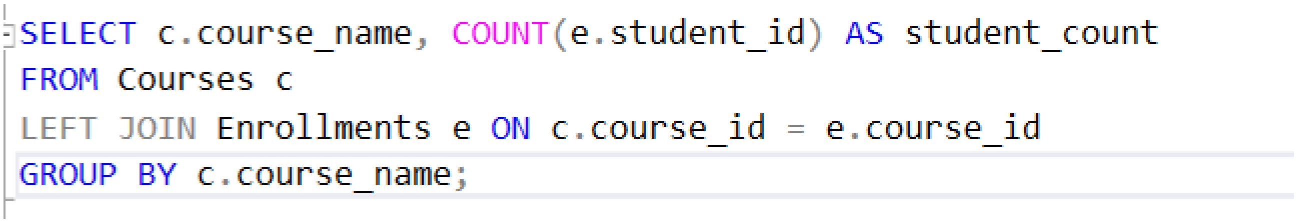
```sql
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name;
```



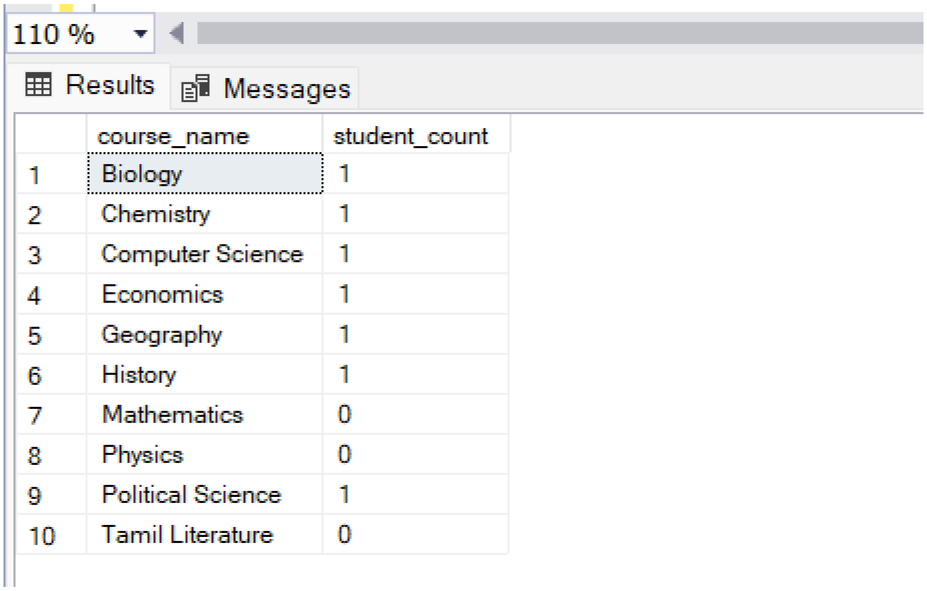
	first_name	last_name	total_payments
1	Priya	Balasubramaniam	700.00
2	Vignesh	Iyer	650.00
3	Shyam	Kumar	700.00
4	Aarushi	Narayan	600.00
5	Sowmya	Narayanan	550.00
6	Divya	Rajendran	500.00
7	Anirudh	Ramasamy	450.00
8	Lakshmi	Sankaran	600.00
9	Aadhithya	Srinivasan	750.00

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.

Query:



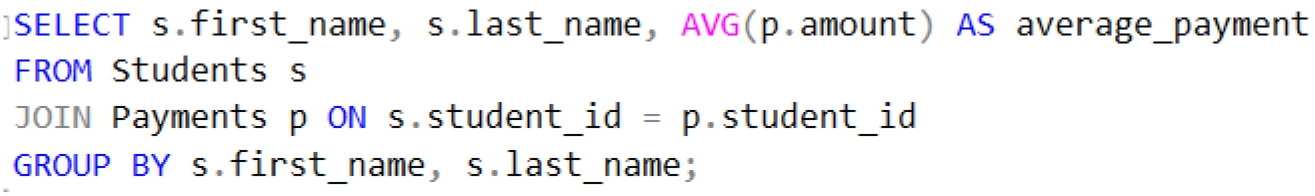
```
SELECT c.course_name, COUNT(e.student_id) AS student_count
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_name;
```



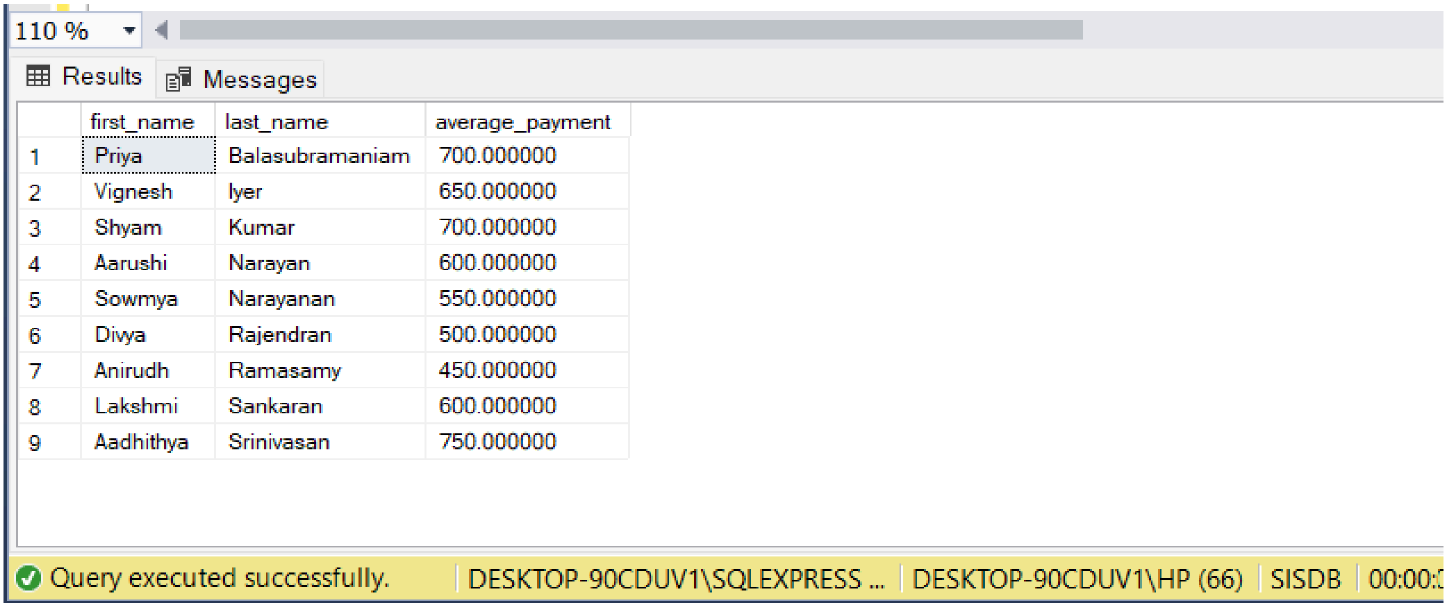
	course_name	student_count
1	Biology	1
2	Chemistry	1
3	Computer Science	1
4	Economics	1
5	Geography	1
6	History	1
7	Mathematics	0
8	Physics	0
9	Political Science	1
10	Tamil Literature	0

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.

Query:



```sql
SELECT s.first_name, s.last_name, AVG(p.amount) AS average_payment
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name;
```



	first_name	last_name	average_payment
1	Priya	Balasubramaniam	700.000000
2	Vignesh	Iyer	650.000000
3	Shyam	Kumar	700.000000
4	Aarushi	Narayan	600.000000
5	Sowmya	Narayanan	550.000000
6	Divya	Rajendran	500.000000
7	Anirudh	Ramasamy	450.000000
8	Lakshmi	Sankaran	600.000000
9	Aadhithya	Srinivasan	750.000000

✓ Query executed successfully. | DESKTOP-90CDUV1\SQLEXPRESS ... | DESKTOP-90CDUV1\HP (66) | SISDB | 00:00: