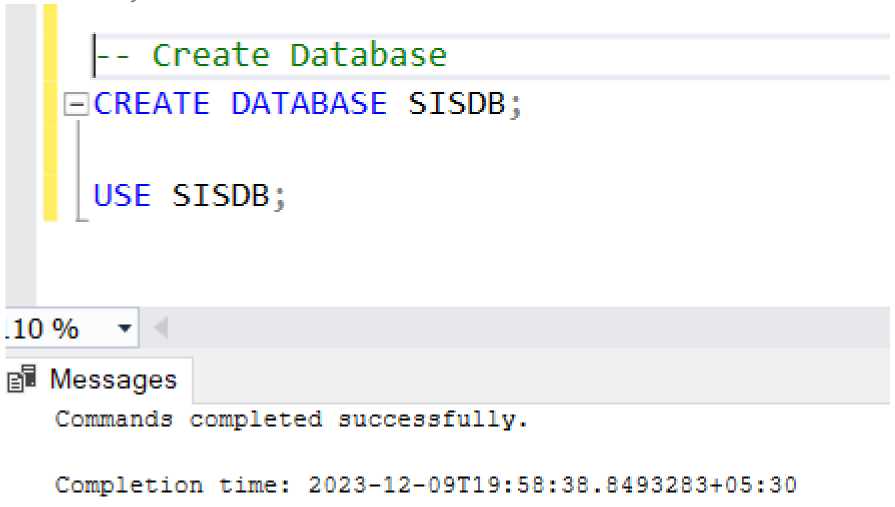
Student Information System (SIS)

Task 1. Database Design:

1. Create the database named "SISDB"

Ans)
```
CREATE DATABASE SISDB;

USE SISDB;
```



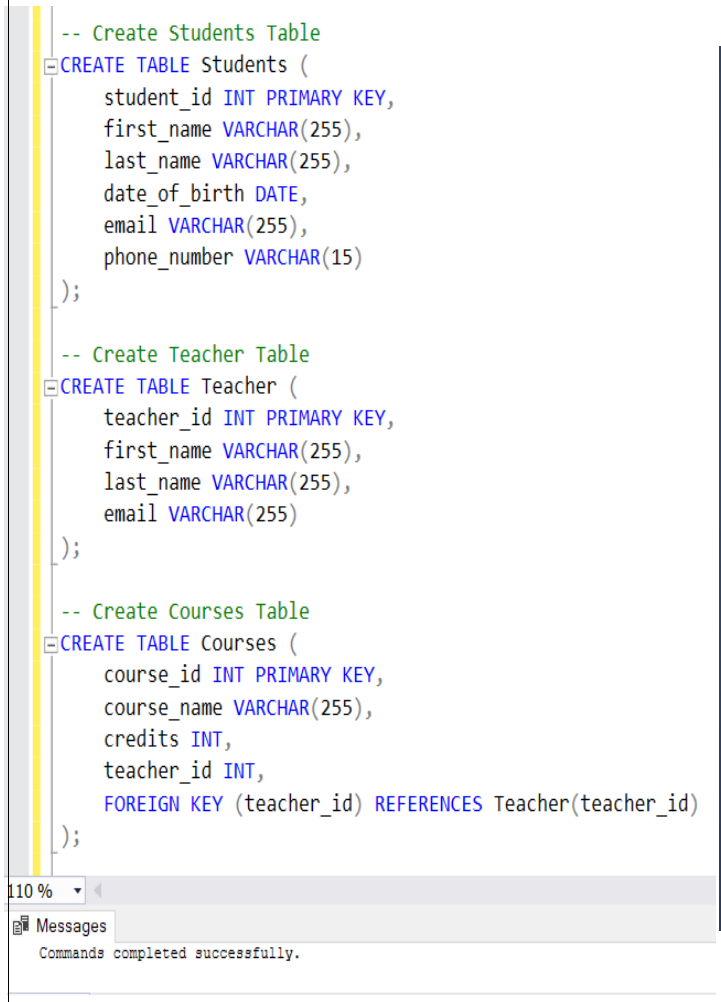
```
-- Create Database
CREATE DATABASE SISDB;

USE SISDB;
```

```
Commands completed successfully.

Completion time: 2023-12-09T19:58:38.8493283+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
   b. Courses
   c. Enrollments
   d. Teacher
   e. Payments

Ans)
```
-- Create Students Table
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    date_of_birth DATE,
    email VARCHAR(255),
    phone_number VARCHAR(15)
);

-- Create Teacher Table
CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255)
);

-- Create Courses Table
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(255),
```

```sql
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);

-- Create Enrollments Table
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);


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

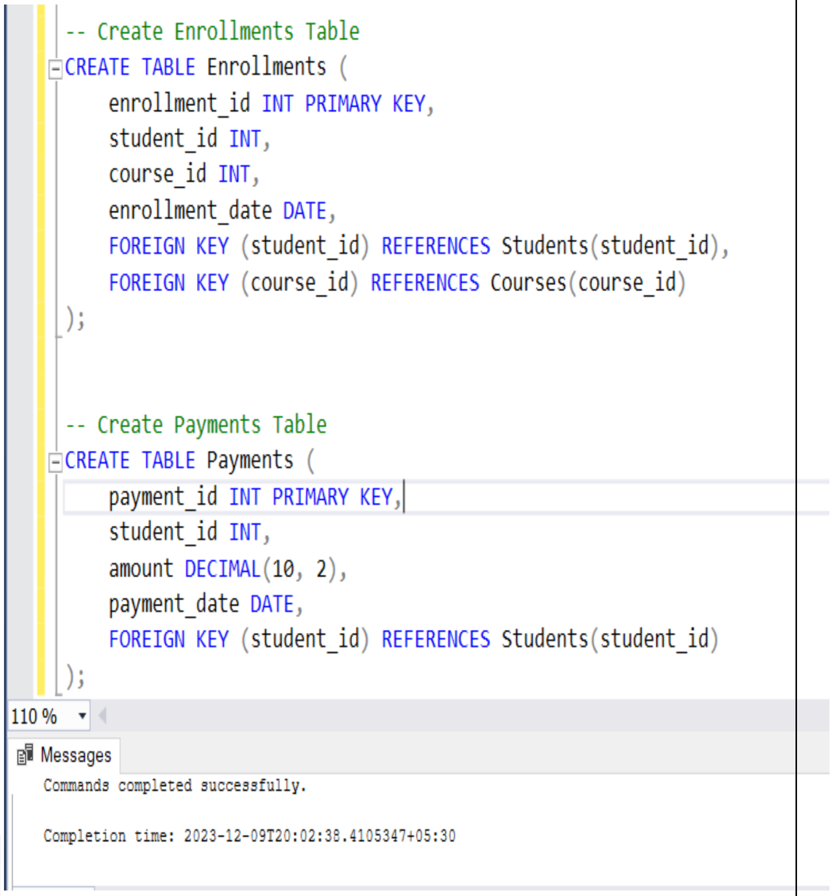


```sql
-- Create Students Table
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    date_of_birth DATE,
    email VARCHAR(255),
    phone_number VARCHAR(15)
);

-- Create Teacher Table
CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255)
);

-- Create Courses Table
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(255),
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);
```

110 %

Messages

```
Commands completed successfully.
```



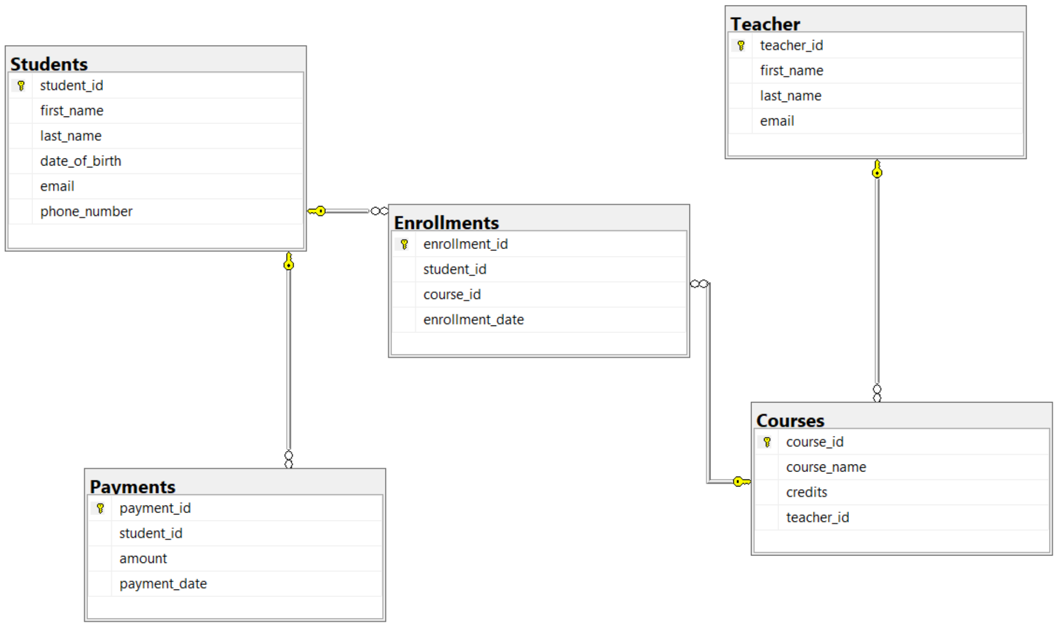
```sql
-- Create Enrollments Table
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);


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

110 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-09T20:02:38.4105347+05:30
```

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



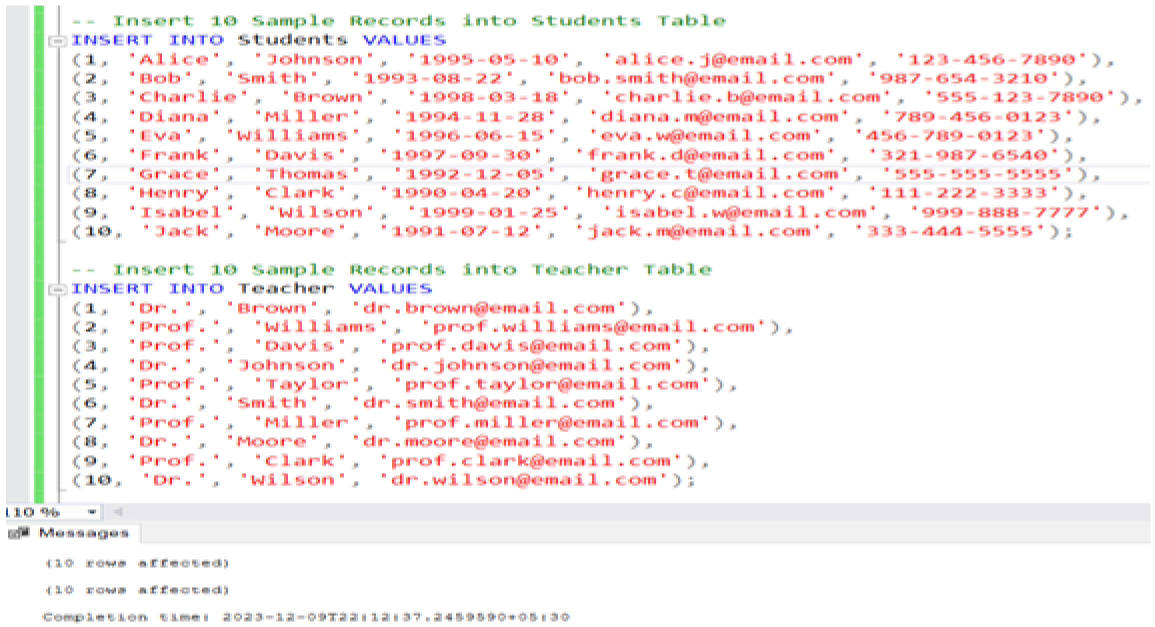
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.
   Ans) Already did in question 2.

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

   i. Students
   ii. Courses
   iii. Enrollments
   iv. Teacher
   v. Payments

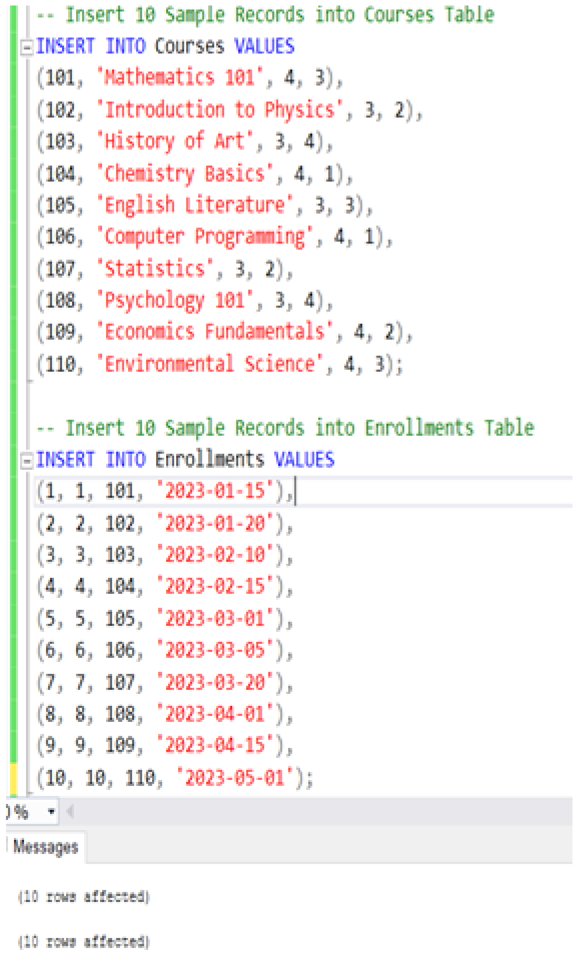
   Ans)

```sql
-- Insert 10 Sample Records into Students Table
INSERT INTO Students VALUES
(1, 'Alice', 'Johnson', '1995-05-10', 'alice.j@email.com', '123-456-7890'),
(2, 'Bob', 'Smith', '1993-08-22', 'bob.smith@email.com', '987-654-3210'),
(3, 'Charlie', 'Brown', '1998-03-18', 'charlie.b@email.com', '555-123-7890'),
(4, 'Diana', 'Miller', '1994-11-28', 'diana.m@email.com', '789-456-0123'),
(5, 'Eva', 'Williams', '1996-06-15', 'eva.w@email.com', '456-789-0123'),
(6, 'Frank', 'Davis', '1997-09-30', 'frank.d@email.com', '321-987-6540'),
(7, 'Grace', 'Thomas', '1992-12-05', 'grace.t@email.com', '555-555-5555'),
(8, 'Henry', 'Clark', '1990-04-20', 'henry.c@email.com', '111-222-3333'),
(9, 'Isabel', 'Wilson', '1999-01-25', 'isabel.w@email.com', '999-888-7777'),
(10, 'Jack', 'Moore', '1991-07-12', 'jack.m@email.com', '333-444-5555');

-- Insert 10 Sample Records into Teacher Table
INSERT INTO Teacher VALUES
(1, 'Dr.', 'Brown', 'dr.brown@email.com'),
(2, 'Prof.', 'Williams', 'prof.williams@email.com'),
(3, 'Prof.', 'Davis', 'prof.davis@email.com'),
(4, 'Dr.', 'Johnson', 'dr.johnson@email.com'),
(5, 'Prof.', 'Taylor', 'prof.taylor@email.com'),
(6, 'Dr.', 'Smith', 'dr.smith@email.com'),
(7, 'Prof.', 'Miller', 'prof.miller@email.com'),
(8, 'Dr.', 'Moore', 'dr.moore@email.com'),
(9, 'Prof.', 'Clark', 'prof.clark@email.com'),
(10, 'Dr.', 'Wilson', 'dr.wilson@email.com');

-- Insert 10 Sample Records into Courses Table
INSERT INTO Courses VALUES
(101, 'Mathematics 101', 4, 3),
(102, 'Introduction to Physics', 3, 2),
(103, 'History of Art', 3, 4),
(104, 'Chemistry Basics', 4, 1),
(105, 'English Literature', 3, 3),
(106, 'Computer Programming', 4, 1),
(107, 'Statistics', 3, 2),
(108, 'Psychology 101', 3, 4),
(109, 'Economics Fundamentals', 4, 2),
(110, 'Environmental Science', 4, 3);

-- Insert 10 Sample Records into Enrollments Table
INSERT INTO Enrollments VALUES
(1, 1, 101, '2023-01-15'),
(2, 2, 102, '2023-01-20'),
(3, 3, 103, '2023-02-10'),
(4, 4, 104, '2023-02-15'),
(5, 5, 105, '2023-03-01'),
(6, 6, 106, '2023-03-05'),
(7, 7, 107, '2023-03-20'),
(8, 8, 108, '2023-04-01'),
(9, 9, 109, '2023-04-15'),
(10, 10, 110, '2023-05-01');

-- Insert 10 Sample Records into Payments Table
INSERT INTO Payments VALUES
(1, 1, 250.00, '2023-02-01'),
(2, 2, 300.00, '2023-02-05'),
(3, 3, 150.00, '2023-03-01'),
(4, 4, 400.00, '2023-03-05'),
(5, 5, 200.00, '2023-04-01'),
(6, 6, 350.00, '2023-04-05'),
(7, 7, 180.00, '2023-05-01'),
(8, 8, 420.00, '2023-05-05'),
(9, 9, 300.00, '2023-06-01'),
(10, 10, 280.00, '2023-06-05');
```



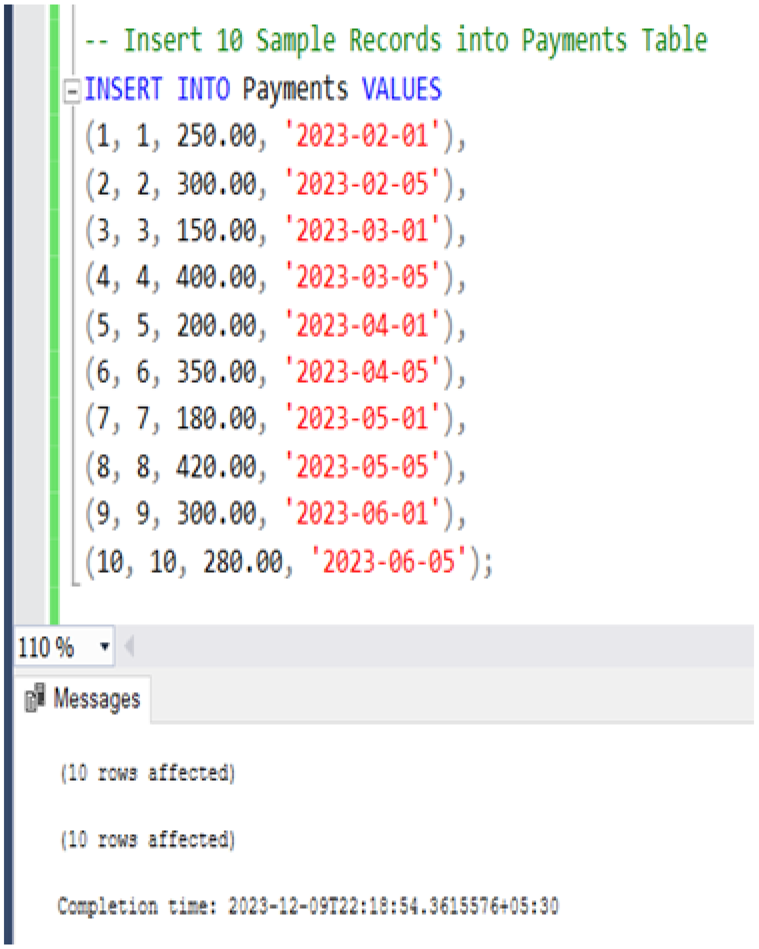
```sql
-- Insert 10 Sample Records into Students Table
INSERT INTO Students VALUES
(1, 'Alice', 'Johnson', '1995-05-10', 'alice.j@email.com', '123-456-7890'),
(2, 'Bob', 'Smith', '1993-08-22', 'bob.smith@email.com', '987-654-3210'),
(3, 'Charlie', 'Brown', '1998-03-18', 'charlie.b@email.com', '555-123-7890'),
(4, 'Diana', 'Miller', '1994-11-28', 'diana.m@email.com', '789-456-0123'),
(5, 'Eva', 'Williams', '1996-06-15', 'eva.w@email.com', '456-789-0123'),
(6, 'Frank', 'Davis', '1997-09-30', 'frank.d@email.com', '321-987-6540'),
(7, 'Grace', 'Thomas', '1992-12-05', 'grace.t@email.com', '555-555-5555'),
(8, 'Henry', 'Clark', '1990-04-20', 'henry.c@email.com', '111-222-3333'),
(9, 'Isabel', 'Wilson', '1999-01-25', 'isabel.w@email.com', '999-888-7777'),
(10, 'Jack', 'Moore', '1991-07-12', 'jack.m@email.com', '333-444-5555');

-- Insert 10 Sample Records into Teacher Table
INSERT INTO Teacher VALUES
(1, 'Dr.', 'Brown', 'dr.brown@email.com'),
(2, 'Prof.', 'Williams', 'prof.williams@email.com'),
(3, 'Prof.', 'Davis', 'prof.davis@email.com'),
(4, 'Dr.', 'Johnson', 'dr.johnson@email.com'),
(5, 'Prof.', 'Taylor', 'prof.taylor@email.com'),
(6, 'Dr.', 'Smith', 'dr.smith@email.com'),
(7, 'Prof.', 'Miller', 'prof.miller@email.com'),
(8, 'Dr.', 'Moore', 'dr.moore@email.com'),
(9, 'Prof.', 'Clark', 'prof.clark@email.com'),
(10, 'Dr.', 'Wilson', 'dr.wilson@email.com');
```

```
(10 rows affected)

(10 rows affected)

Completion time: 2023-12-09T22:12:37.2459590+05:30
```



```sql
-- Insert 10 Sample Records into Courses Table
INSERT INTO Courses VALUES
(101, 'Mathematics 101', 4, 3),
(102, 'Introduction to Physics', 3, 2),
(103, 'History of Art', 3, 4),
(104, 'Chemistry Basics', 4, 1),
(105, 'English Literature', 3, 3),
(106, 'Computer Programming', 4, 1),
(107, 'Statistics', 3, 2),
(108, 'Psychology 101', 3, 4),
(109, 'Economics Fundamentals', 4, 2),
(110, 'Environmental Science', 4, 3);

-- Insert 10 Sample Records into Enrollments Table
INSERT INTO Enrollments VALUES
(1, 1, 101, '2023-01-15'),
(2, 2, 102, '2023-01-20'),
(3, 3, 103, '2023-02-10'),
(4, 4, 104, '2023-02-15'),
(5, 5, 105, '2023-03-01'),
(6, 6, 106, '2023-03-05'),
(7, 7, 107, '2023-03-20'),
(8, 8, 108, '2023-04-01'),
(9, 9, 109, '2023-04-15'),
(10, 10, 110, '2023-05-01');
```

```
(10 rows affected)

(10 rows affected)
```



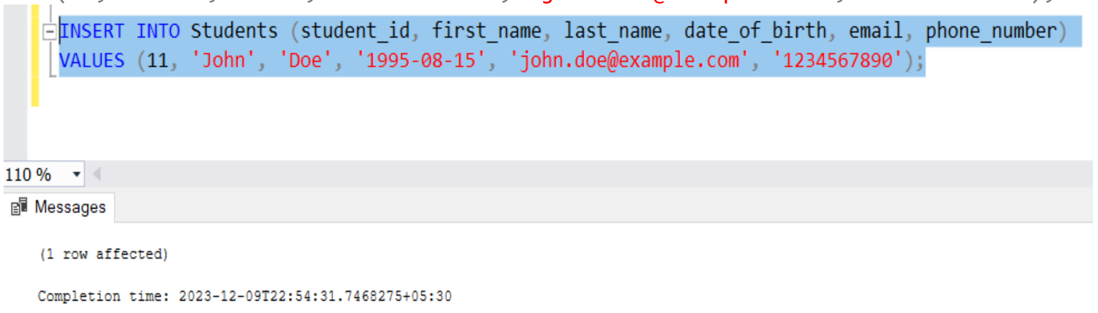
```sql
-- Insert 10 Sample Records into Payments Table
INSERT INTO Payments VALUES
(1, 1, 250.00, '2023-02-01'),
(2, 2, 300.00, '2023-02-05'),
(3, 3, 150.00, '2023-03-01'),
(4, 4, 400.00, '2023-03-05'),
(5, 5, 200.00, '2023-04-01'),
(6, 6, 350.00, '2023-04-05'),
(7, 7, 180.00, '2023-05-01'),
(8, 8, 420.00, '2023-05-05'),
(9, 9, 300.00, '2023-06-01'),
(10, 10, 280.00, '2023-06-05');
```

```
(10 rows affected)

(10 rows affected)

Completion time: 2023-12-09T22:18:54.3615576+05:30
```

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

Ans)
```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
VALUES (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
```

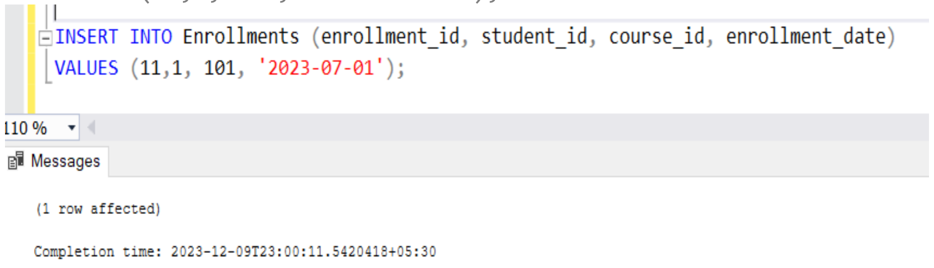


```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
VALUES (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
```

```
(1 row affected)

Completion time: 2023-12-09T22:54:31.7468275+05:30
```

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.

   Ans)
```
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES (11,1, 101, '2023-07-01');
```

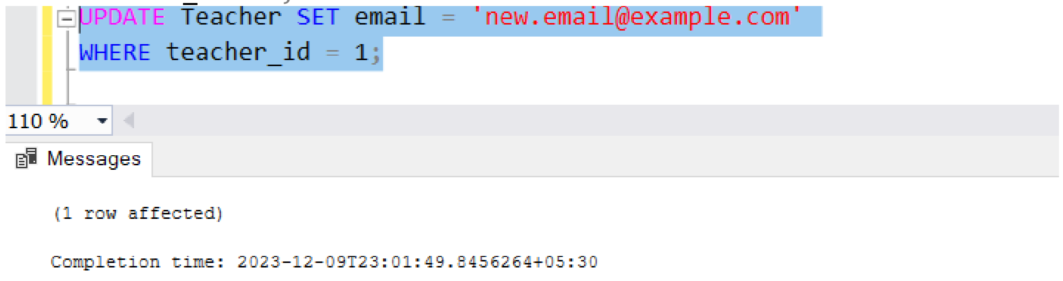


```
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES (11,1, 101, '2023-07-01');
```

```
(1 row affected)

Completion time: 2023-12-09T23:00:11.5420418+05:30
```

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

Ans)
```
UPDATE Teacher SET email = 'new.email@example.com'
WHERE teacher_id = 1;
```



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

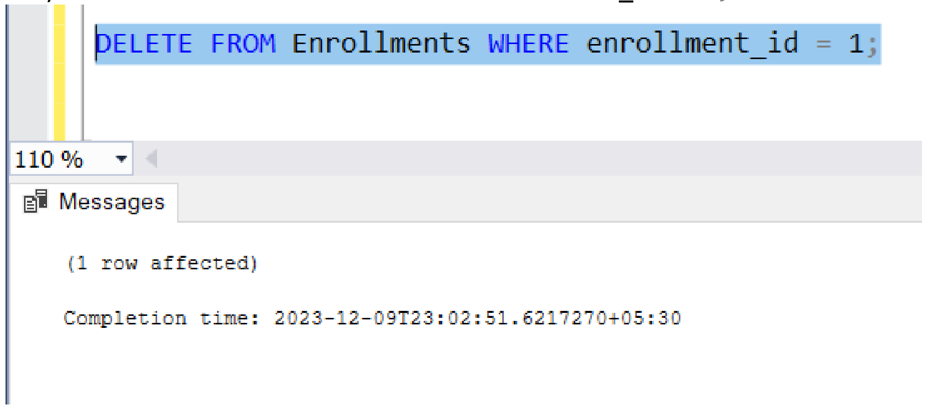
110 %

Messages

```
(1 row affected)

Completion time: 2023-12-09T23:01:49.8456264+05:30
```

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.

Ans)
```
DELETE FROM Enrollments WHERE enrollment_id = 1;
```



```
DELETE FROM Enrollments WHERE enrollment_id = 1;
```

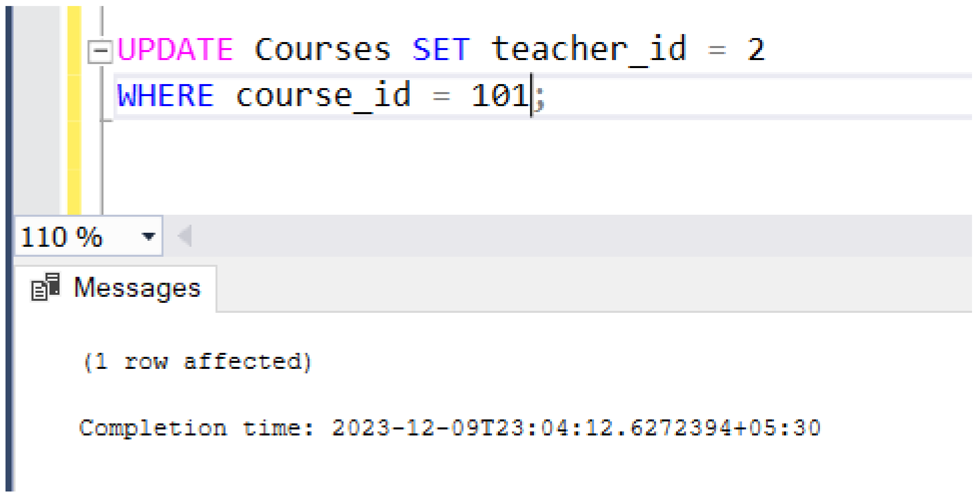
110 %

Messages

```
(1 row affected)

Completion time: 2023-12-09T23:02:51.6217270+05:30
```

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

Ans)
```
UPDATE Courses SET teacher_id = 2 WHERE course_id = 101;
```



```
UPDATE Courses SET teacher_id = 2
WHERE course_id = 101;
```

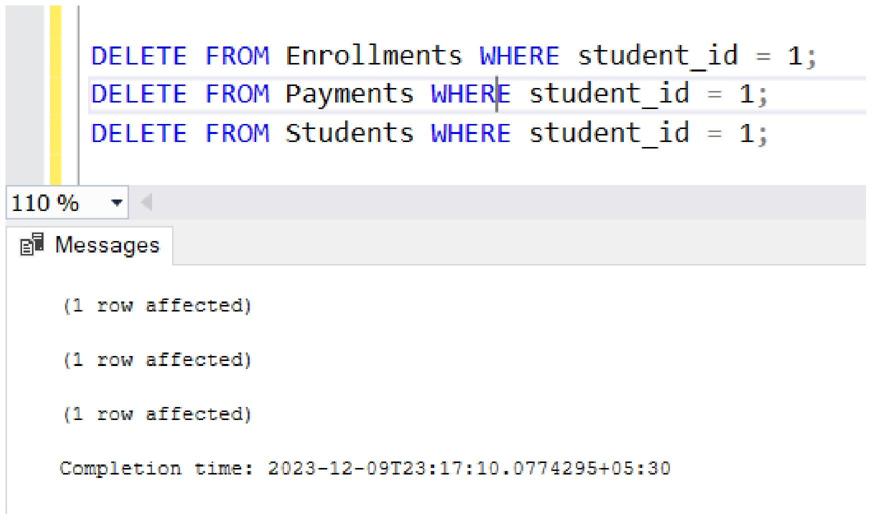
110 %

Messages

```
(1 row affected)

Completion time: 2023-12-09T23:04:12.6272394+05:30
```

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.

Ans)
```
DELETE FROM Enrollments WHERE student_id = 1;
DELETE FROM Payments WHERE student_id = 1;
DELETE FROM Students WHERE student_id = 1;
```

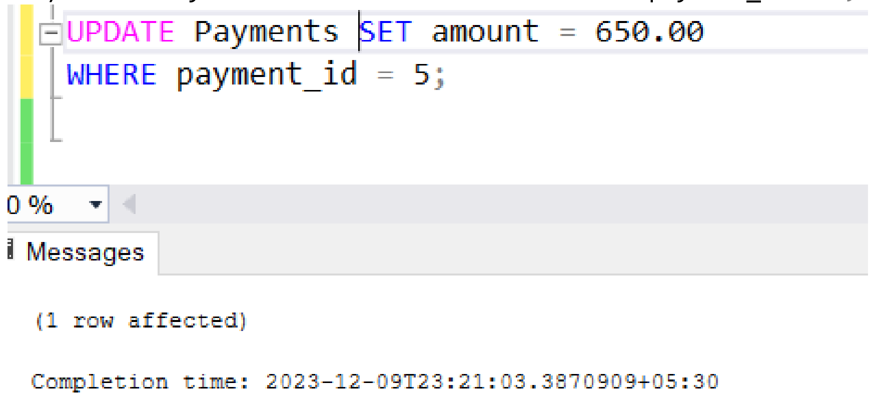


```
DELETE FROM Enrollments WHERE student_id = 1;
DELETE FROM Payments WHERE student_id = 1;
DELETE FROM Students WHERE student_id = 1;
```

```
(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2023-12-09T23:17:10.0774295+05:30
```

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

   Ans) `UPDATE Payments SET amount = 650.00 WHERE payment_id = 5;`



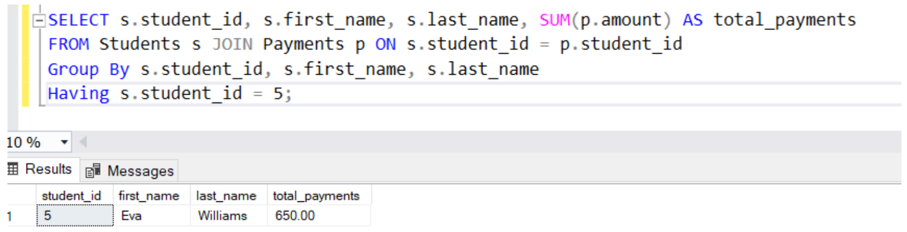
```
UPDATE Payments SET amount = 650.00
WHERE payment_id = 5;
```

```
(1 row affected)

Completion time: 2023-12-09T23:21:03.3870909+05:30
```

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.

   Ans)
```
SELECT s.student_id, 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.student_id, s.first_name, s.last_name
Having s.student_id = 5;
```

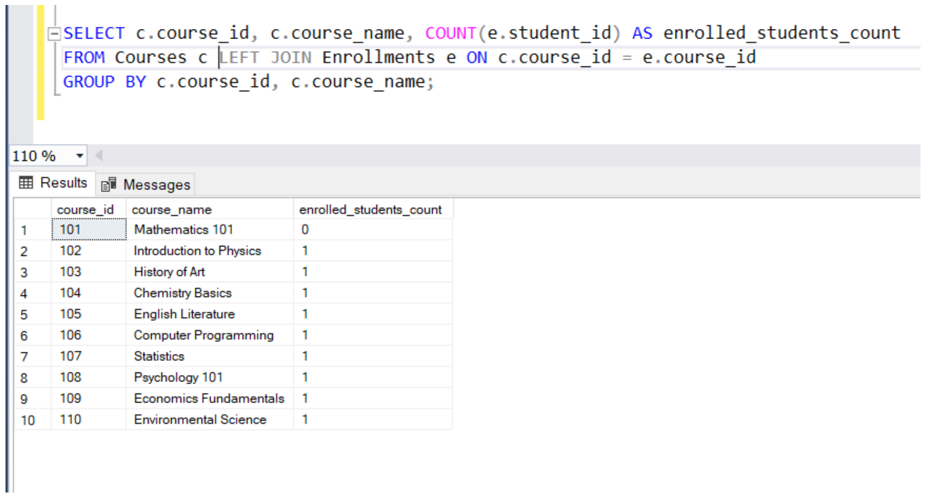


```
SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM Students s JOIN Payments p ON s.student_id = p.student_id
Group By s.student_id, s.first_name, s.last_name
Having s.student_id = 5;
```

	student_id	first_name	last_name	total_payments
1	5	Eva	Williams	650.00

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.

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



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

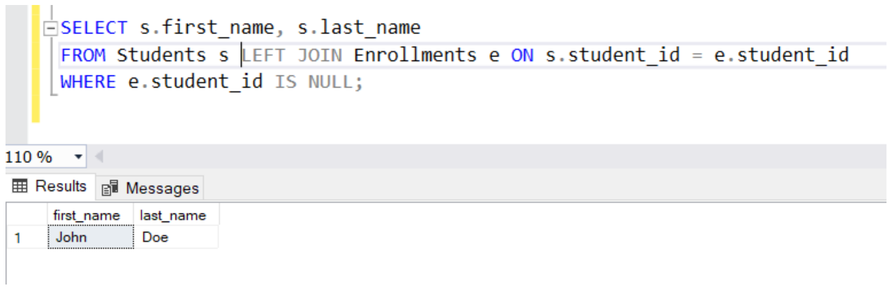
110 %

Results | Messages

	course_id	course_name	enrolled_students_count
1	101	Mathematics 101	0
2	102	Introduction to Physics	1
3	103	History of Art	1
4	104	Chemistry Basics	1
5	105	English Literature	1
6	106	Computer Programming	1
7	107	Statistics	1
8	108	Psychology 101	1
9	109	Economics Fundamentals	1
10	110	Environmental Science	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.

```
Ans) 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;
```



```
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;
```

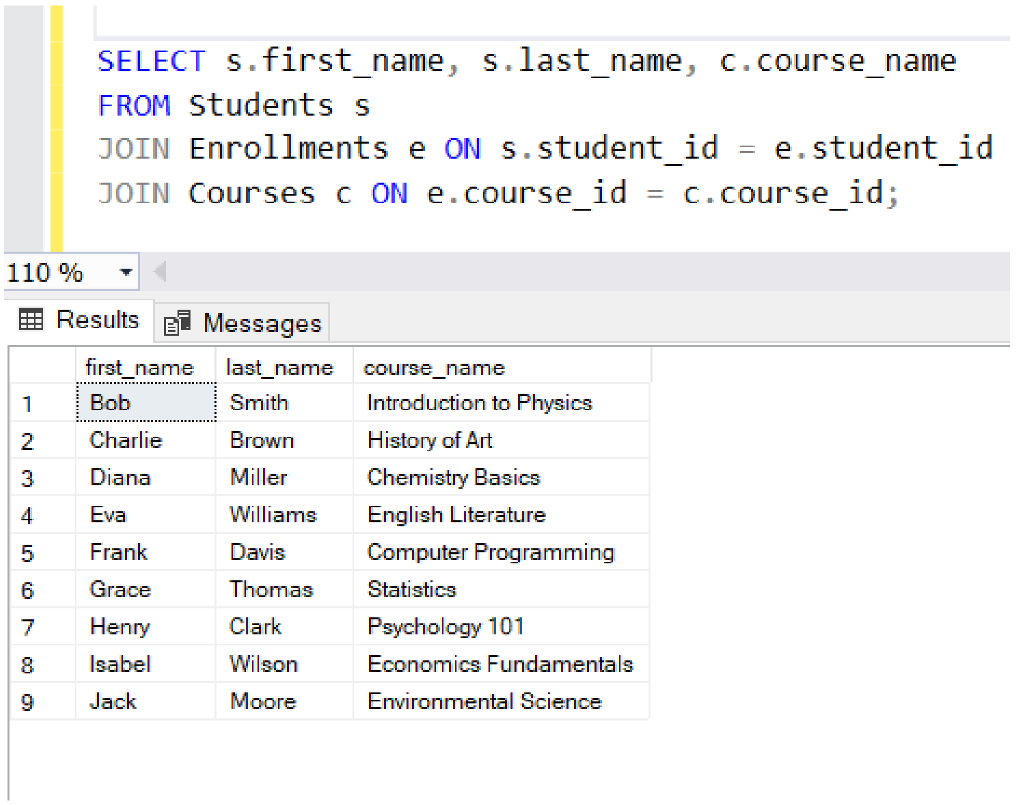
110 %

Results | Messages

	first_name	last_name
1	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.

```
Ans) 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;
```



```
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;
```

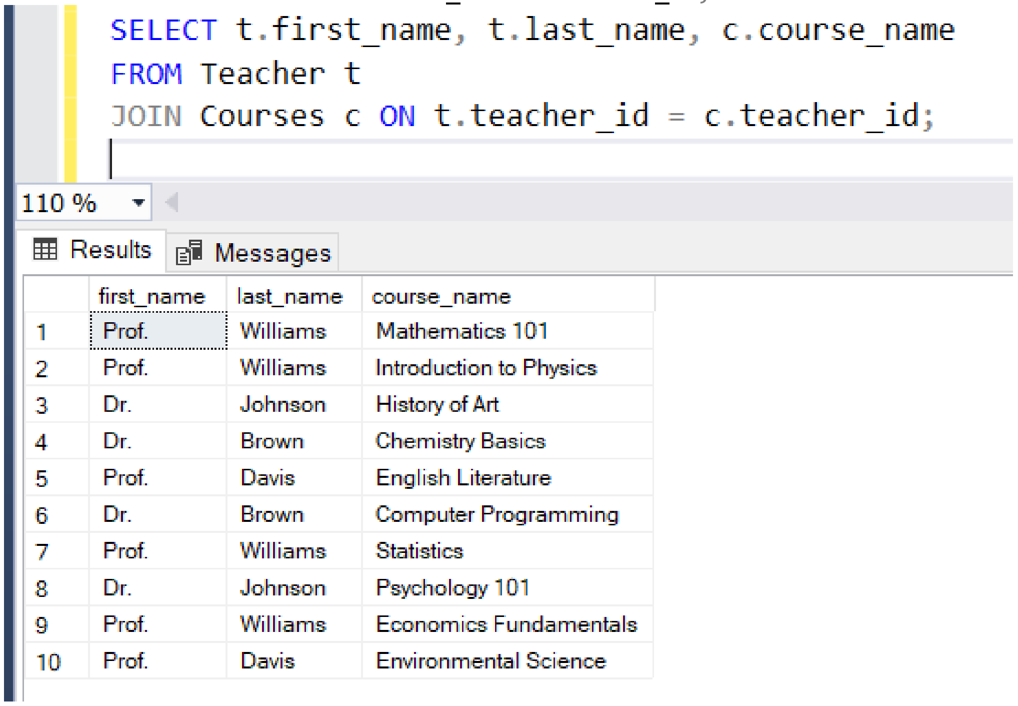
110 %

Results | Messages

	first_name	last_name	course_name
1	Bob	Smith	Introduction to Physics
2	Charlie	Brown	History of Art
3	Diana	Miller	Chemistry Basics
4	Eva	Williams	English Literature
5	Frank	Davis	Computer Programming
6	Grace	Thomas	Statistics
7	Henry	Clark	Psychology 101
8	Isabel	Wilson	Economics Fundamentals
9	Jack	Moore	Environmental Science

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.

Ans)
```
SELECT t.first_name, t.last_name, c.course_name
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id;
```



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

110 %

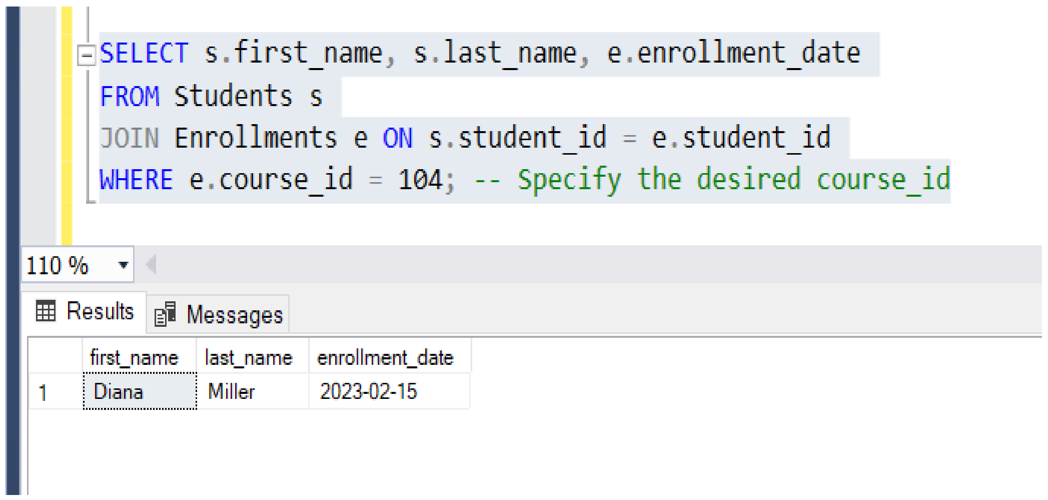
Results | Messages

	first_name	last_name	course_name
1	Prof.	Williams	Mathematics 101
2	Prof.	Williams	Introduction to Physics
3	Dr.	Johnson	History of Art
4	Dr.	Brown	Chemistry Basics
5	Prof.	Davis	English Literature
6	Dr.	Brown	Computer Programming
7	Prof.	Williams	Statistics
8	Dr.	Johnson	Psychology 101
9	Prof.	Williams	Economics Fundamentals
10	Prof.	Davis	Environmental Science

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.

Ans)
```
SELECT s.first_name, s.last_name, e.enrollment_date
FROM Students s
```

```sql
JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.course_id = 104; -- Specify the desired course_id
```



```sql
SELECT s.first_name, s.last_name, e.enrollment_date
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.course_id = 104; -- Specify the desired course_id
```

110 %

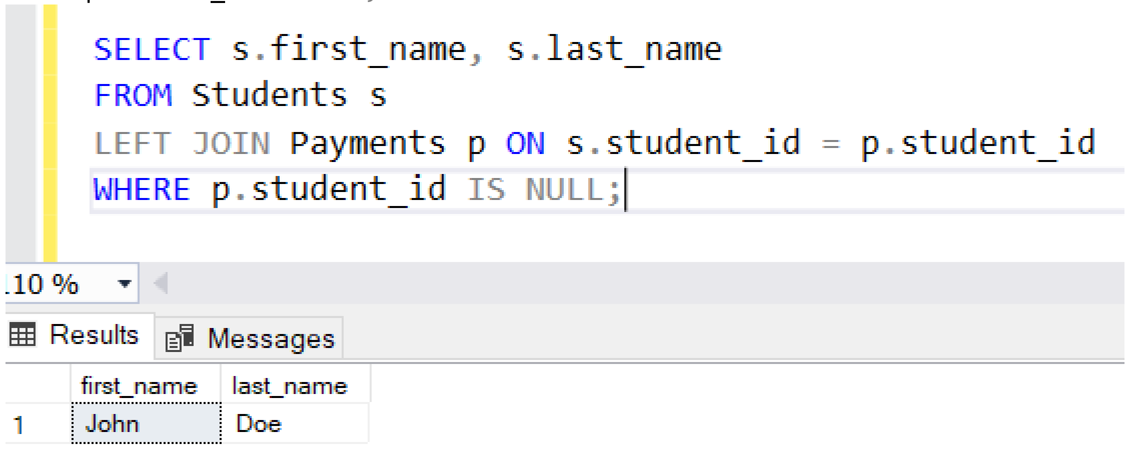
Results | Messages

```
   first_name  last_name  enrollment_date
1  Diana       Miller     2023-02-15
```

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.

Ans) `SELECT s.first_name, s.last_name`

```sql
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
WHERE p.student_id IS NULL;
```



```sql
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;
```

.10 %

Results | Messages

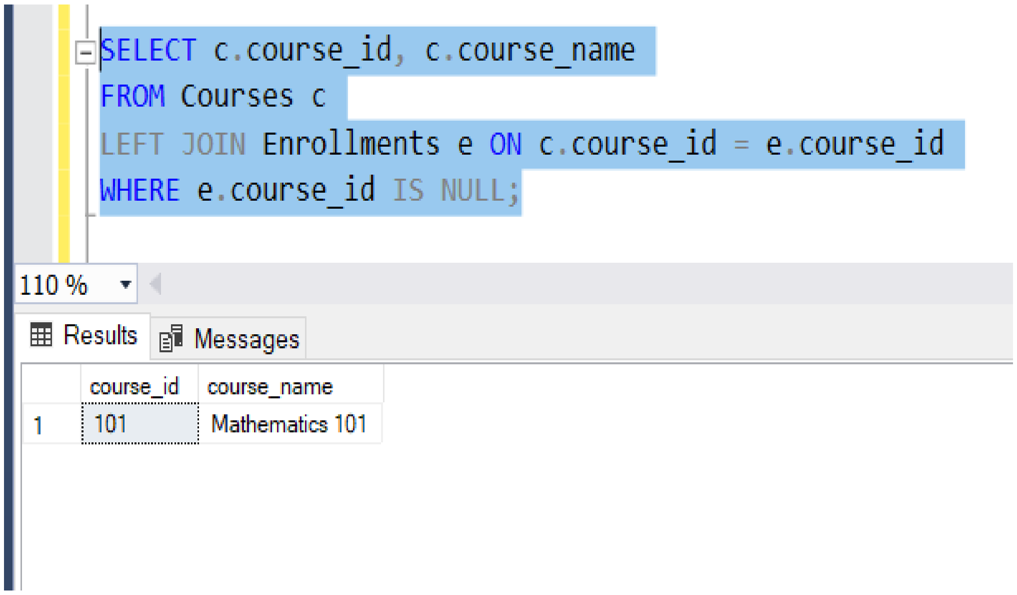
```
   first_name  last_name
1  John        Doe
```

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.

Ans) `SELECT c.course_id, c.course_name`

```sql
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
```

```
WHERE e.course_id IS NULL;
```



```
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;
```

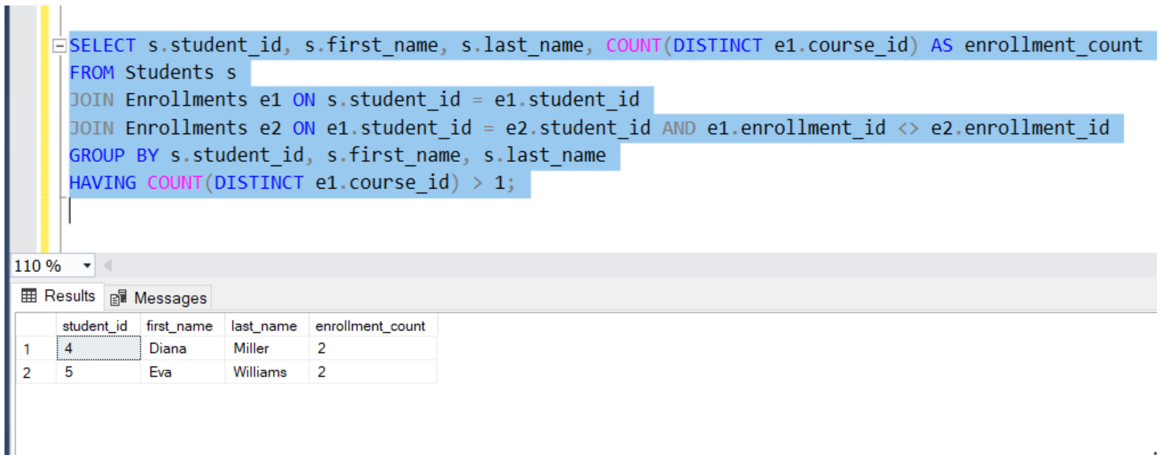
110 %

Results | Messages

	course_id	course_name
1	101	Mathematics 101

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.

Ans)
```
SELECT s.student_id, s.first_name, s.last_name, COUNT(DISTINCT e1.course_id) AS
enrollment_count FROM Students s
JOIN Enrollments e1 ON s.student_id = e1.student_id
JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.enrollment_id <> e2.enrollment_id
GROUP BY s.student_id, s.first_name, s.last_name
HAVING COUNT(DISTINCT e1.course_id) > 1;
```

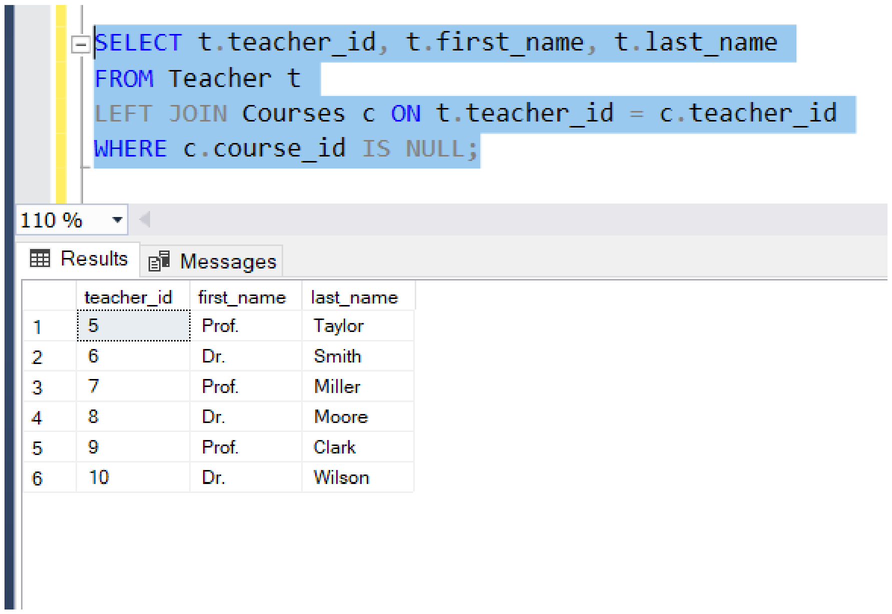


```
SELECT s.student_id, s.first_name, s.last_name, COUNT(DISTINCT e1.course_id) AS enrollment_count
FROM Students s
JOIN Enrollments e1 ON s.student_id = e1.student_id
JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.enrollment_id <> e2.enrollment_id
GROUP BY s.student_id, s.first_name, s.last_name
HAVING COUNT(DISTINCT e1.course_id) > 1;
```

	student_id	first_name	last_name	enrollment_count
1	4	Diana	Miller	2
2	5	Eva	Williams	2

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.

Ans)
```
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;
```



```
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;
```

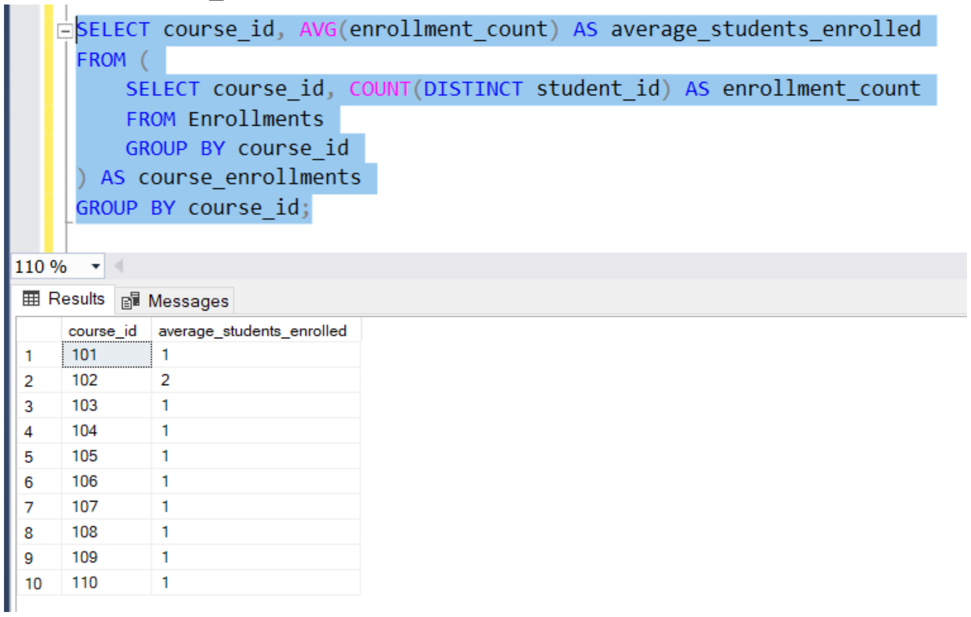
	teacher_id	first_name	last_name
1	5	Prof.	Taylor
2	6	Dr.	Smith
3	7	Prof.	Miller
4	8	Dr.	Moore
5	9	Prof.	Clark
6	10	Dr.	Wilson

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.

Ans)
```
SELECT course_id, AVG(enrollment_count) AS average_students_enrolled
```

```
FROM (
    SELECT course_id, COUNT(DISTINCT student_id) AS enrollment_count
    FROM Enrollments
    GROUP BY course_id
) AS course_enrollments
GROUP BY course_id;
```

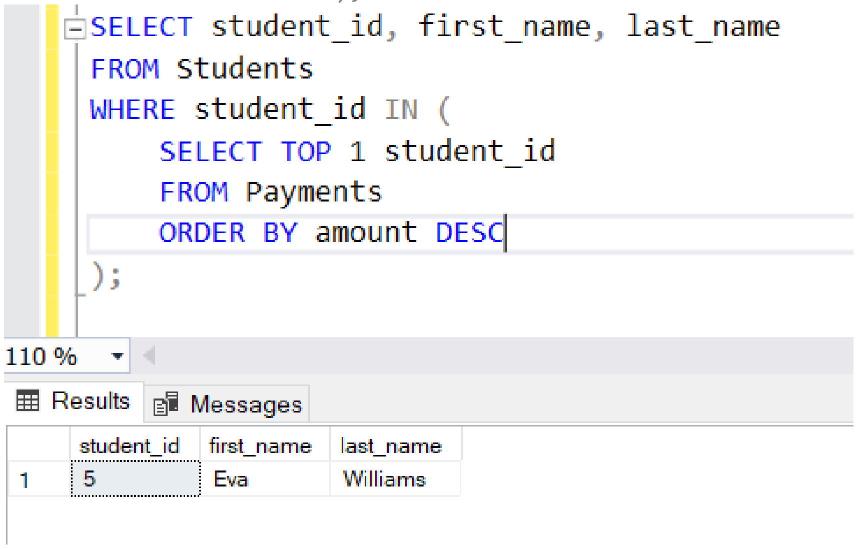


```
SELECT course_id, AVG(enrollment_count) AS average_students_enrolled
FROM (
    SELECT course_id, COUNT(DISTINCT student_id) AS enrollment_count
    FROM Enrollments
    GROUP BY course_id
) AS course_enrollments
GROUP BY course_id;
```

	course_id	average_students_enrolled
1	101	1
2	102	2
3	103	1
4	104	1
5	105	1
6	106	1
7	107	1
8	108	1
9	109	1
10	110	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.

Ans)
```
SELECT student_id, first_name, last_name FROM Students
WHERE student_id IN (
    SELECT TOP 1 student_id
    FROM Payments
    ORDER BY amount DESC);
```

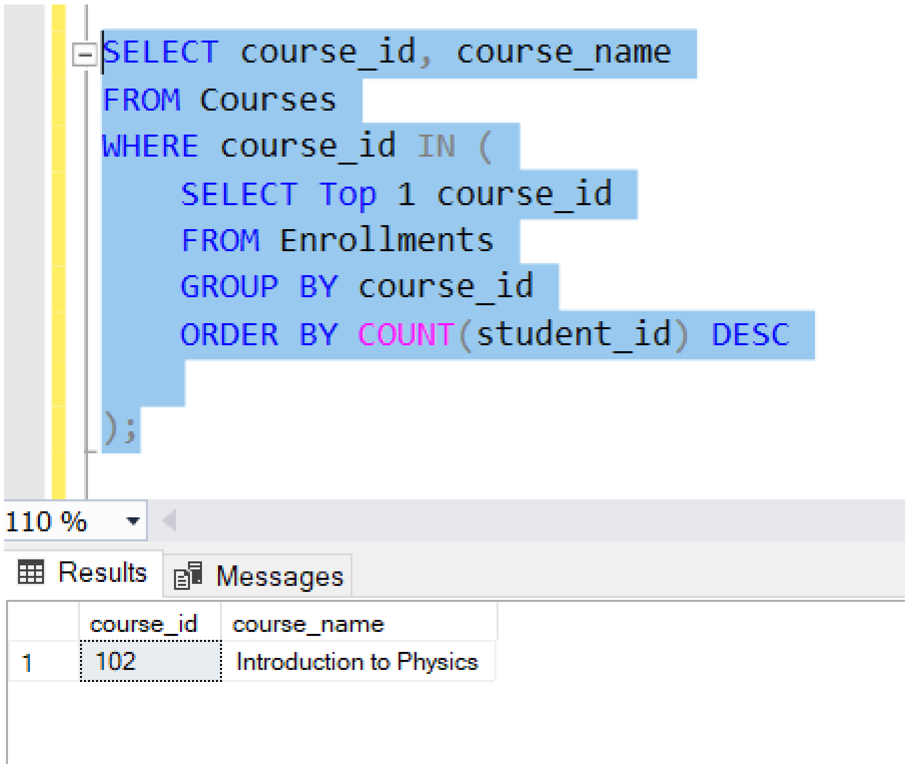


```
SELECT student_id, first_name, last_name
FROM Students
WHERE student_id IN (
    SELECT TOP 1 student_id
    FROM Payments
    ORDER BY amount DESC
);
```

	student_id	first_name	last_name
1	5	Eva	Williams

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

Ans)
```
SELECT course_id, course_name FROM Courses
WHERE course_id IN (
    SELECT Top 1 course_id
    FROM Enrollments
    GROUP BY course_id
    ORDER BY COUNT(student_id) DESC);
```



```
SELECT course_id, course_name
FROM Courses
WHERE course_id IN (
    SELECT Top 1 course_id
    FROM Enrollments
    GROUP BY course_id
    ORDER BY COUNT(student_id) DESC

);
```

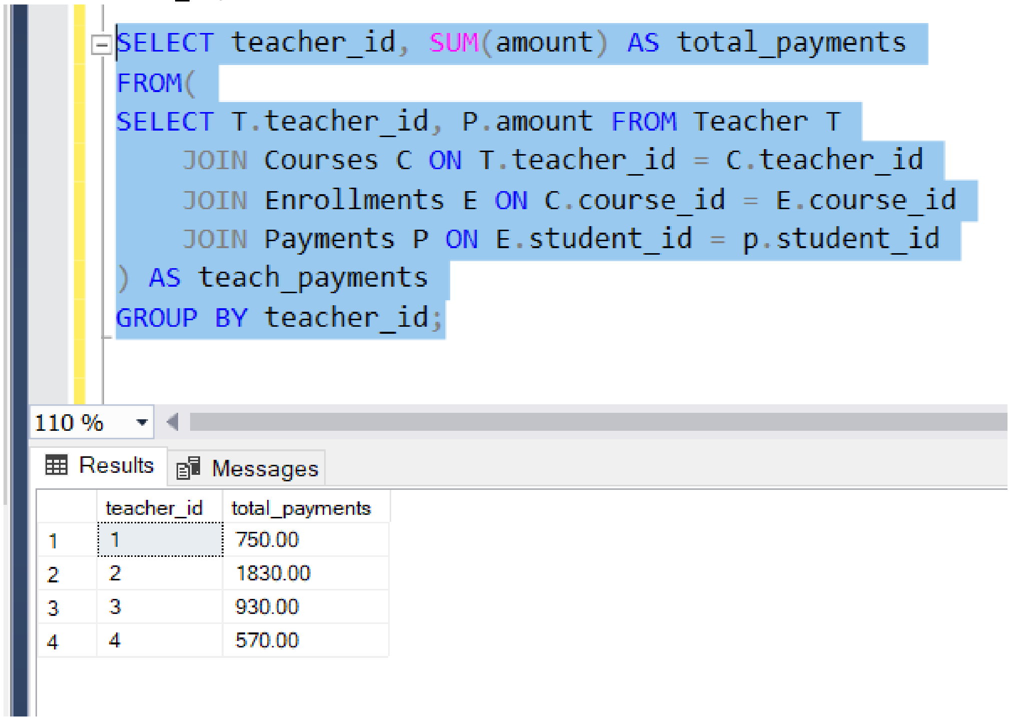
110 %

Results | Messages

	course_id	course_name
1	102	Introduction to Physics

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

Ans)
```
SELECT teacher_id, SUM(amount) AS total_payments
FROM(
SELECT T.teacher_id, P.amount 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
) AS teach_payments
GROUP BY teacher_id;
```



```
SELECT teacher_id, SUM(amount) AS total_payments
FROM(
SELECT T.teacher_id, P.amount 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
) AS teach_payments
GROUP BY teacher_id;
```

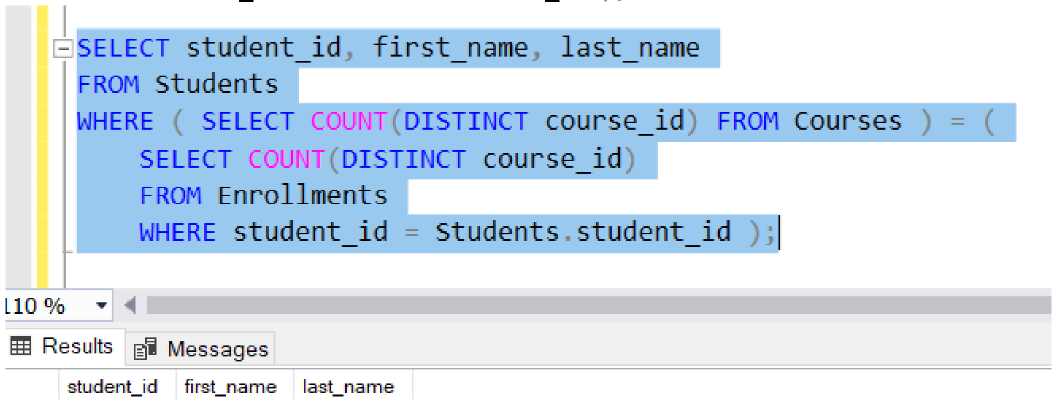
110 %

Results | Messages

	teacher_id	total_payments
1	1	750.00
2	2	1830.00
3	3	930.00
4	4	570.00

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

Ans)
```
SELECT student_id, first_name, last_name
FROM Students
WHERE ( SELECT COUNT(DISTINCT course_id) FROM Courses ) = (
    SELECT COUNT(DISTINCT course_id)
    FROM Enrollments
    WHERE student_id = Students.student_id );
```

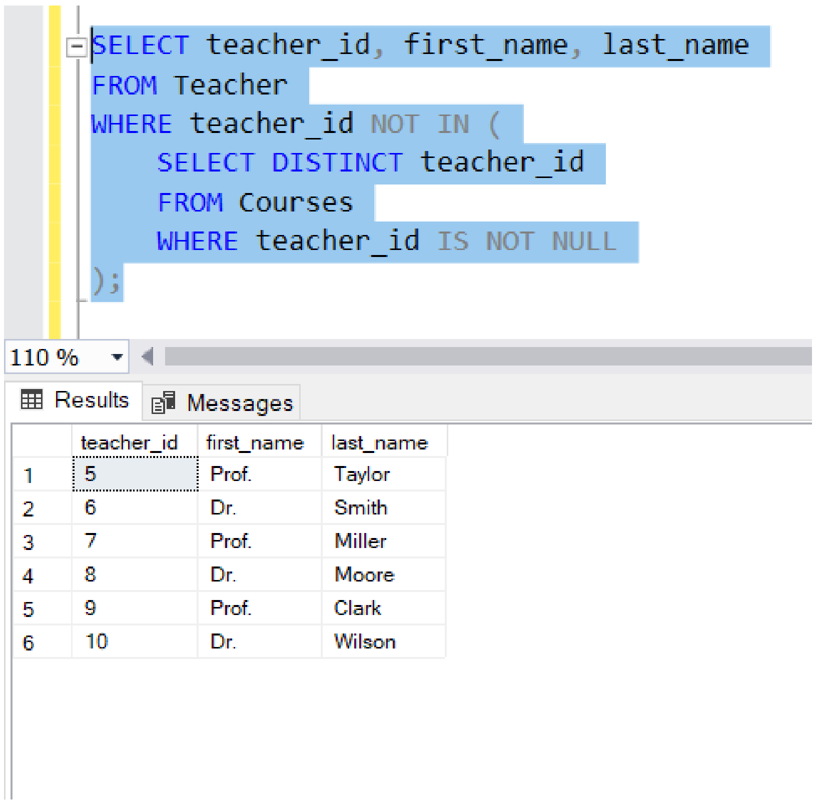


```
SELECT student_id, first_name, last_name
FROM Students
WHERE ( SELECT COUNT(DISTINCT course_id) FROM Courses ) = (
    SELECT COUNT(DISTINCT course_id)
    FROM Enrollments
    WHERE student_id = Students.student_id );
```

```
student_id  first_name  last_name
```

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

Ans)
```
SELECT teacher_id, first_name, last_name
FROM Teacher
WHERE teacher_id NOT IN (
    SELECT DISTINCT teacher_id
    FROM Courses
    WHERE teacher_id IS NOT NULL);
```

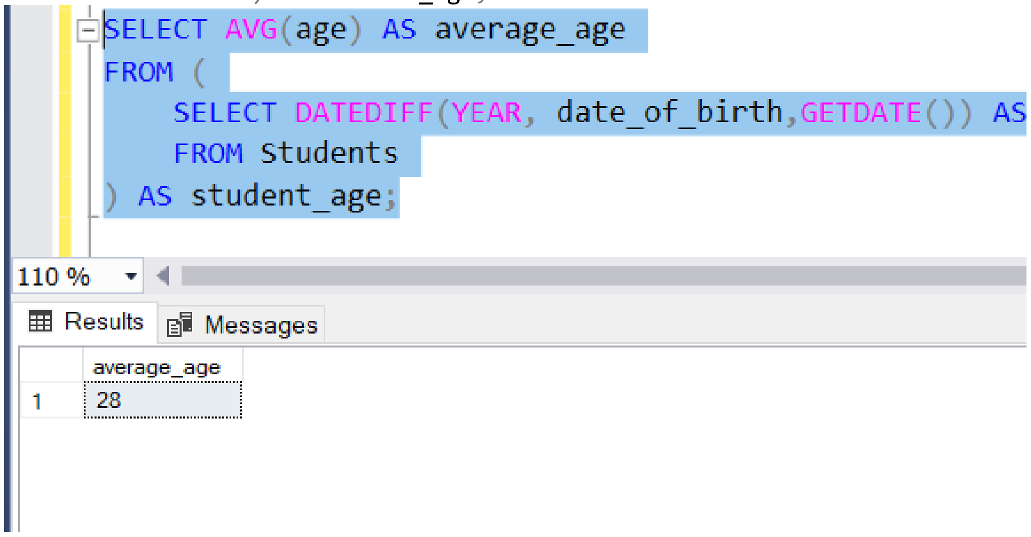


```
SELECT teacher_id, first_name, last_name
FROM Teacher
WHERE teacher_id NOT IN (
    SELECT DISTINCT teacher_id
    FROM Courses
    WHERE teacher_id IS NOT NULL
);
```

	teacher_id	first_name	last_name
1	5	Prof.	Taylor
2	6	Dr.	Smith
3	7	Prof.	Miller
4	8	Dr.	Moore
5	9	Prof.	Clark
6	10	Dr.	Wilson

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

Ans)
```
SELECT AVG(age) AS average_age
FROM (
    SELECT DATEDIFF(YEAR, date_of_birth,GETDATE()) AS age
    FROM Students ) AS student_age;
```

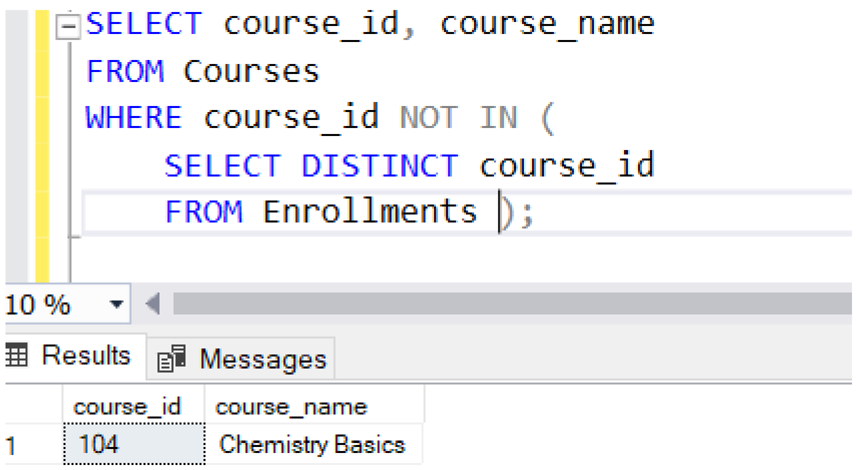


```
SELECT AVG(age) AS average_age
FROM (
    SELECT DATEDIFF(YEAR, date_of_birth,GETDATE()) AS
    FROM Students
) AS student_age;
```

	average_age
1	28

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

Ans)
```
SELECT course_id, course_name
FROM Courses
WHERE course_id NOT IN (
    SELECT DISTINCT course_id
    FROM Enrollments );
```



```sql
SELECT course_id, course_name
FROM Courses
WHERE course_id NOT IN (
    SELECT DISTINCT course_id
    FROM Enrollments );
```

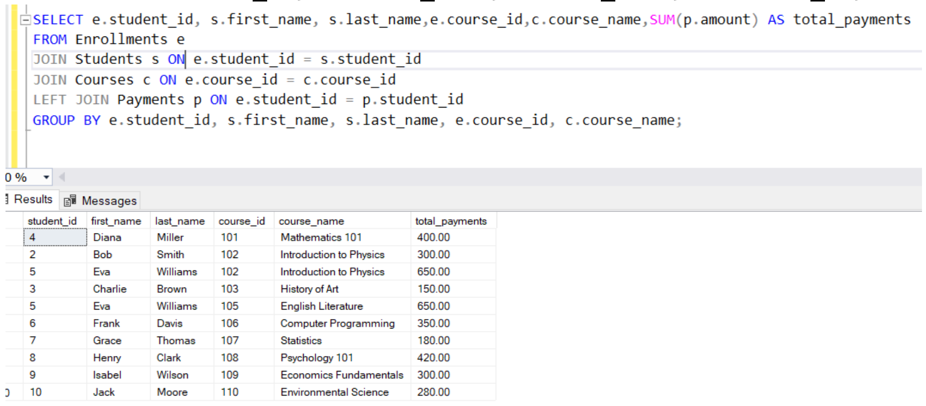
10 %

Results | Messages

	course_id	course_name
1	104	Chemistry Basics

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

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



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

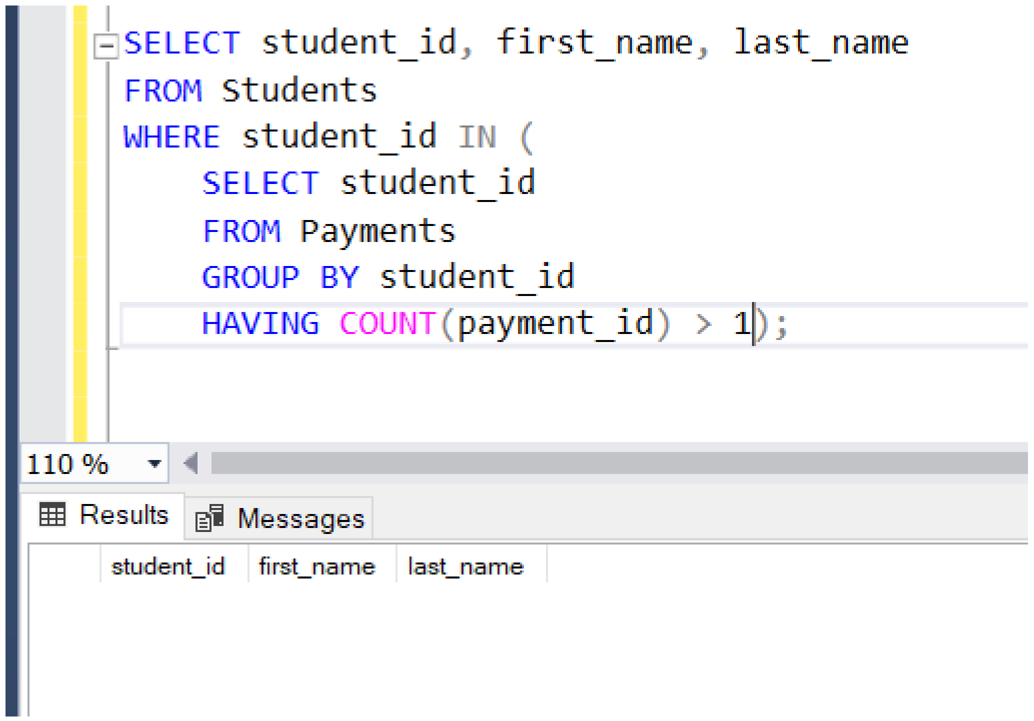
0 %

Results | Messages

student_id	first_name	last_name	course_id	course_name	total_payments
4	Diana	Miller	101	Mathematics 101	400.00
2	Bob	Smith	102	Introduction to Physics	300.00
5	Eva	Williams	102	Introduction to Physics	650.00
3	Charlie	Brown	103	History of Art	150.00
5	Eva	Williams	105	English Literature	650.00
6	Frank	Davis	106	Computer Programming	350.00
7	Grace	Thomas	107	Statistics	180.00
8	Henry	Clark	108	Psychology 101	420.00
9	Isabel	Wilson	109	Economics Fundamentals	300.00
10	Jack	Moore	110	Environmental Science	280.00

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.

Ans)
```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);
```



```
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);
```

110 %

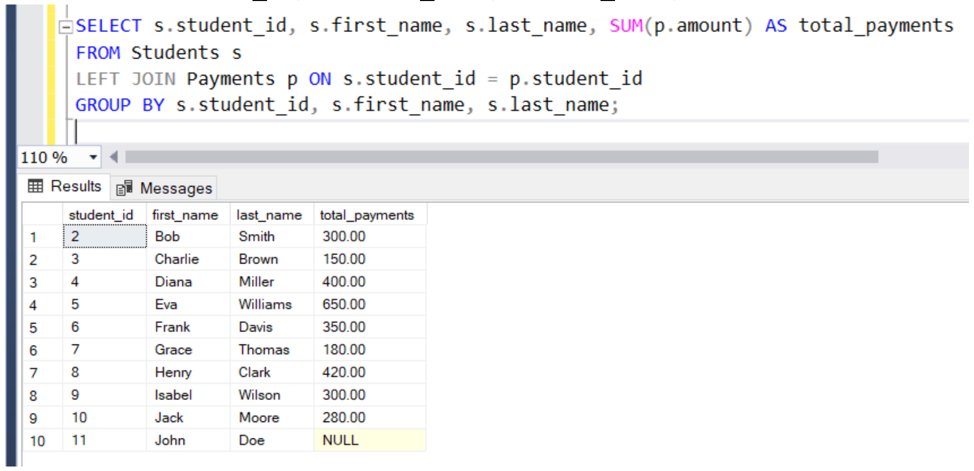
Results | Messages

```
student_id  first_name  last_name
```

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.

Ans) `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;
```



```
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;
```

110 %

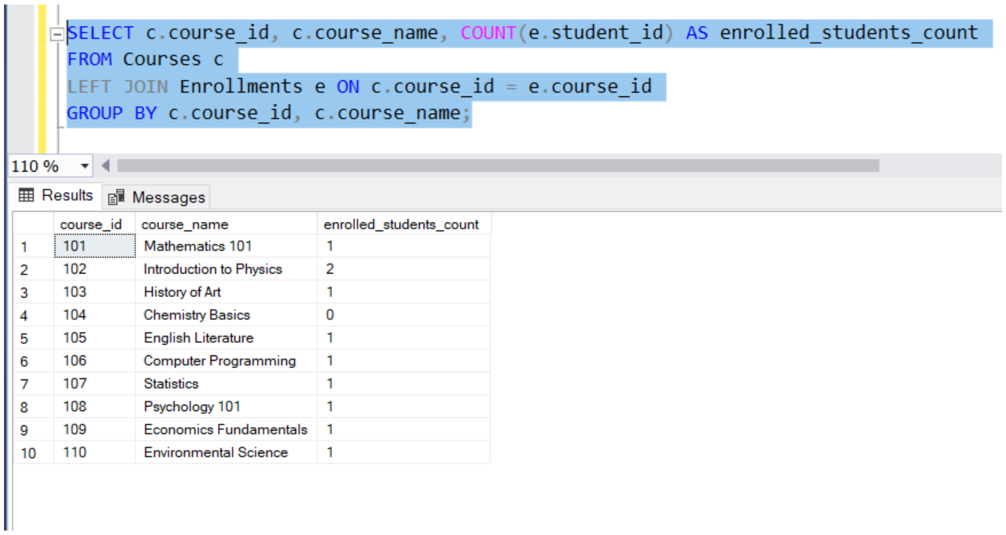
Results | Messages

```
    student_id  first_name  last_name  total_payments
1   2           Bob         Smith      300.00
2   3           Charlie     Brown      150.00
3   4           Diana       Miller     400.00
4   5           Eva         Williams   650.00
5   6           Frank       Davis      350.00
6   7           Grace       Thomas     180.00
7   8           Henry       Clark      420.00
8   9           Isabel      Wilson     300.00
9   10          Jack        Moore      280.00
10  11          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.

Ans) `SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students_count`

```
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name;
```



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

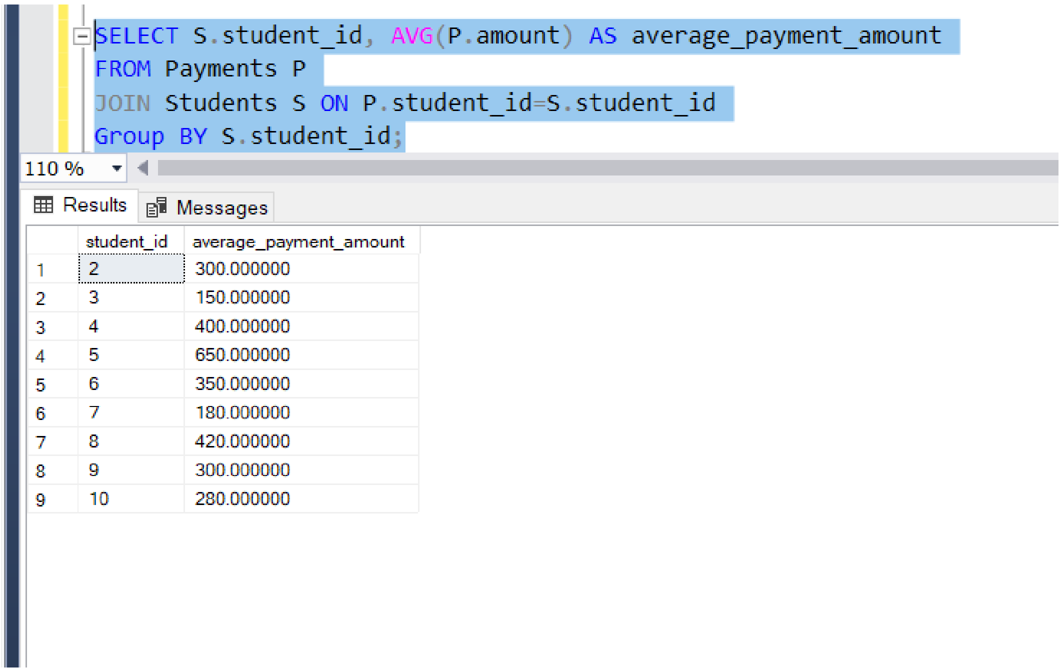
110 %

Results | Messages

	course_id	course_name	enrolled_students_count
1	101	Mathematics 101	1
2	102	Introduction to Physics	2
3	103	History of Art	1
4	104	Chemistry Basics	0
5	105	English Literature	1
6	106	Computer Programming	1
7	107	Statistics	1
8	108	Psychology 101	1
9	109	Economics Fundamentals	1
10	110	Environmental Science	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.

Ans)
```sql
SELECT S.student_id, AVG(P.amount) AS average_payment_amount
FROM Payments P
JOIN Students S ON P.student_id=S.student_id
Group BY S.student_id;
```



```sql
SELECT S.student_id, AVG(P.amount) AS average_payment_amount
FROM Payments P
JOIN Students S ON P.student_id=S.student_id
Group BY S.student_id;
```

110 %

Results | Messages

	student_id	average_payment_amount
1	2	300.000000
2	3	150.000000
3	4	400.000000
4	5	650.000000
5	6	350.000000
6	7	180.000000
7	8	420.000000
8	9	300.000000
9	10	280.000000