ASSIGNMENT

NAME: Aniroop Gupta

ASSIGNMENT: Student Information System

TASK-1 Database Design:

--1. Create the database named "SISDB"

```
Create Database SISDB;

Use SISDB;
```

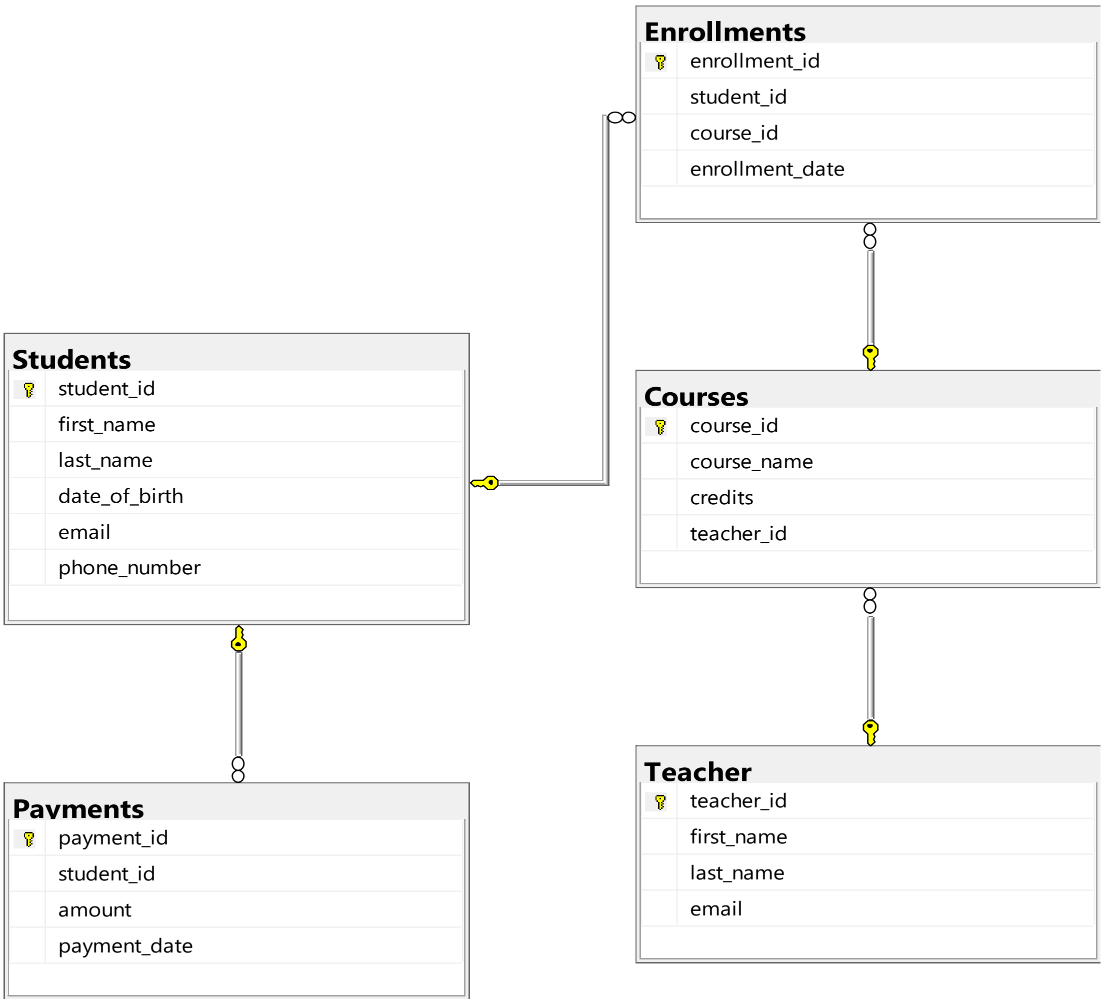
--2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based

--on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data

--types, constraints, and relationships. a. Students b. Courses c. Enrollments d. Teacher e. Payments

```
Create Table Students(
student_id INT PRIMARY KEY,
first_name VARCHAR(255),
last_name VARCHAR(255),
date_of_birth DATE,
email VARCHAR(100),
phone_number VARCHAR(20));

Create Table Courses(
course_id INT PRIMARY KEY,
course_name VARCHAR(255),
credits INT,
teacher_id INT);

CREATE TABLE Enrollments (
enrollment_id INT IDENTITY(1,1) PRIMARY KEY,
student_id INT,
course_id INT,
enrollment_date DATE);


CREATE Table Teacher(

teacher_id INT IDENTITY(1,1) PRIMARY KEY,

first_name VARCHAR(255),

last_name VARCHAR(255),

email VARCHAR(100));


CREATE TABLE Payments (

payment_id INT IDENTITY(1,1) PRIMARY KEY,

student_id INT,

amount BIGINT,

payment_date DATE);
```

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



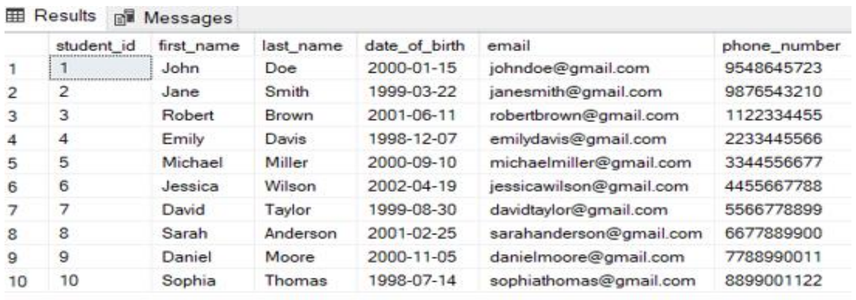
--4. Create appropriate Primary Key and Foreign Key constraints for referential integrity

```
ALTER TABLE Courses

ADD CONSTRAINT FK_Courses_Teacher FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id);

ALTER TABLE Enrollments

ADD CONSTRAINT FK_Enrollments_Students FOREIGN KEY (student_id) REFERENCES
Students(student_id);

ALTER TABLE Enrollments

ADD CONSTRAINT FK_Enrollments_Courses FOREIGN KEY (course_id) REFERENCES
Courses(course_id);

ALTER TABLE Payments

ADD CONSTRAINT FK_Payments_Students FOREIGN KEY (student_id) REFERENCES
Students(student_id);
```

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

--iii. Enrollments iv. Teacher v. Payments

```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)

VALUES

(1, 'John', 'Doe', '2000-01-15', 'johndoe@gmail.com', 9548645723),

(2, 'Jane', 'Smith', '1999-03-22', 'janesmith@gmail.com', 9876543210),

(3, 'Robert', 'Brown', '2001-06-11', 'robertbrown@gmail.com', 1122334455),

(4, 'Emily', 'Davis', '1998-12-07', 'emilydavis@gmail.com', 2233445566),

(5, 'Michael', 'Miller', '2000-09-10', 'michaelmiller@gmail.com', 3344556677),

(6, 'Jessica', 'Wilson', '2002-04-19', 'jessicawilson@gmail.com', 4455667788),

(7, 'David', 'Taylor', '1999-08-30', 'davidtaylor@gmail.com', 5566778899),

(8, 'Sarah', 'Anderson', '2001-02-25', 'sarahanderson@gmail.com', 6677889900),

(9, 'Daniel', 'Moore', '2000-11-05', 'danielmoore@gmail.com', 7788990011),

(10, 'Sophia', 'Thomas', '1998-07-14', 'sophiathomas@gmail.com', 8899001122);

Select * from Students;
```



	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	John	Doe	2000-01-15	johndoe@gmail.com	9548645723
2	2	Jane	Smith	1999-03-22	janesmith@gmail.com	9876543210
3	3	Robert	Brown	2001-06-11	robertbrown@gmail.com	1122334455
4	4	Emily	Davis	1998-12-07	emilydavis@gmail.com	2233445566
5	5	Michael	Miller	2000-09-10	michaelmiller@gmail.com	3344556677
6	6	Jessica	Wilson	2002-04-19	jessicawilson@gmail.com	4455667788
7	7	David	Taylor	1999-08-30	davidtaylor@gmail.com	5566778899
8	8	Sarah	Anderson	2001-02-25	sarahanderson@gmail.com	6677889900
9	9	Daniel	Moore	2000-11-05	danielmoore@gmail.com	7788990011
10	10	Sophia	Thomas	1998-07-14	sophiathomas@gmail.com	8899001122

INSERT INTO Teacher (first_name, last_name, email) VALUES

('Suresh', 'Verma', 'suresh.verma@example.com'),

('Anita', 'Mishra', 'anita.mishra@example.com'),

('Ramesh', 'Patel', 'ramesh.patel@example.com'),

('Sunita', 'Gupta', 'sunita.gupta@example.com'),

('Rajesh', 'Kumar', 'rajesh.kumar@example.com'),

('Rekha', 'Sharma', 'rekha.sharma@example.com'),

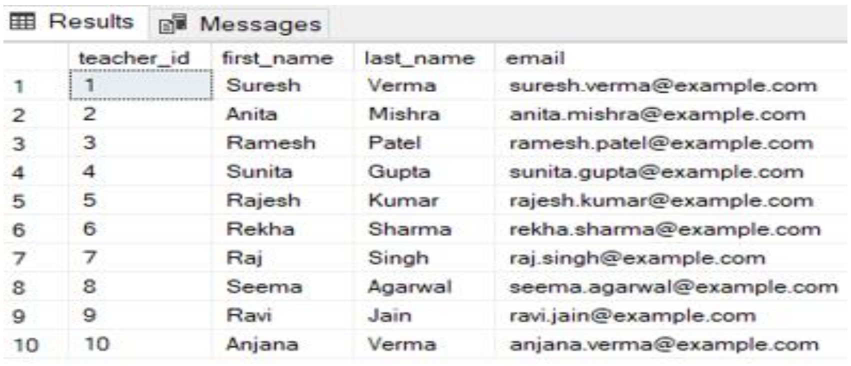
('Raj', 'Singh', 'raj.singh@example.com'),

('Seema', 'Agarwal', 'seema.agarwal@example.com'),

('Ravi', 'Jain', 'ravi.jain@example.com'),

('Anjana', 'Verma', 'anjana.verma@example.com');

Select * from Teacher;



```
    teacher_id  first_name  last_name  email
1   1           Suresh      Verma      suresh.verma@example.com
2   2           Anita       Mishra     anita.mishra@example.com
3   3           Ramesh      Patel      ramesh.patel@example.com
4   4           Sunita      Gupta      sunita.gupta@example.com
5   5           Rajesh      Kumar      rajesh.kumar@example.com
6   6           Rekha       Sharma     rekha.sharma@example.com
7   7           Raj         Singh      raj.singh@example.com
8   8           Seema       Agarwal    seema.agarwal@example.com
9   9           Ravi        Jain       ravi.jain@example.com
10  10          Anjana      Verma      anjana.verma@example.com
```

INSERT INTO Courses (course_id, course_name, credits, teacher_id)

VALUES

(101, 'Mathematics', 3, 1),

(102, 'Physics', 4, 2),

(103, 'Chemistry', 4, 3),

(104, 'Biology', 3, 4),

(105, 'Computer Science', 5, 5),

(106, 'History', 3, 6),

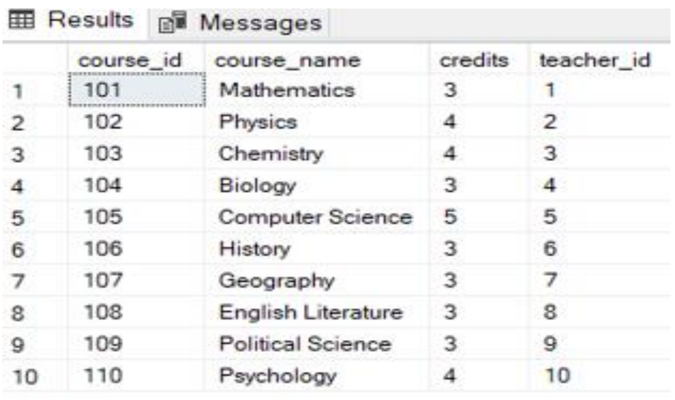
(107, 'Geography', 3, 7),

(108, 'English Literature', 3, 8),

(109, 'Political Science', 3, 9),

(110, 'Psychology', 4, 10);

Select * from Courses;



	course_id	course_name	credits	teacher_id
1	101	Mathematics	3	1
2	102	Physics	4	2
3	103	Chemistry	4	3
4	104	Biology	3	4
5	105	Computer Science	5	5
6	106	History	3	6
7	107	Geography	3	7
8	108	English Literature	3	8
9	109	Political Science	3	9
10	110	Psychology	4	10

INSERT INTO Enrollments (student_id, course_id, enrollment_date)

VALUES

(1, 101, '2023-08-10'),

(2, 102, '2023-08-11'),

(3, 103, '2023-08-12'),

(4, 104, '2023-08-13'),

(5, 105, '2023-08-14'),

(6, 106, '2023-08-15'),

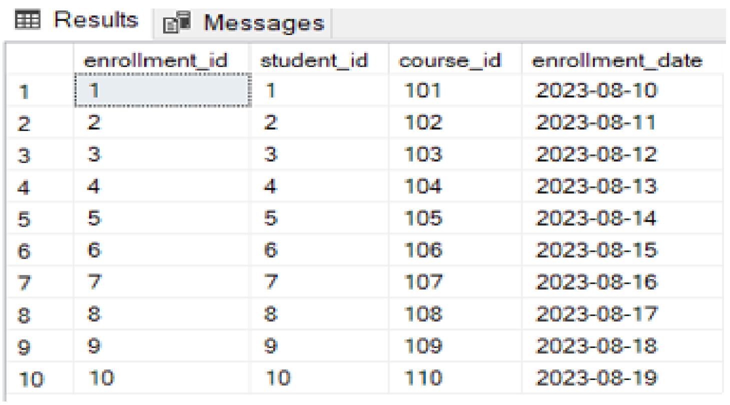
(7, 107, '2023-08-16'),

(8, 108, '2023-08-17'),

(9, 109, '2023-08-18'),

(10, 110, '2023-08-19');

Select * from Enrollments;



	enrollment_id	student_id	course_id	enrollment_date
1	1	1	101	2023-08-10
2	2	2	102	2023-08-11
3	3	3	103	2023-08-12
4	4	4	104	2023-08-13
5	5	5	105	2023-08-14
6	6	6	106	2023-08-15
7	7	7	107	2023-08-16
8	8	8	108	2023-08-17
9	9	9	109	2023-08-18
10	10	10	110	2023-08-19

INSERT INTO Payments (student_id, amount, payment_date)

VALUES

(1, 1500, '2023-09-01'),

(2, 2000, '2023-09-02'),

(3, 1800, '2023-09-03'),

(4, 2500, '2023-09-04'),

(5, 2200, '2023-09-05'),

(6, 2300, '2023-09-06'),

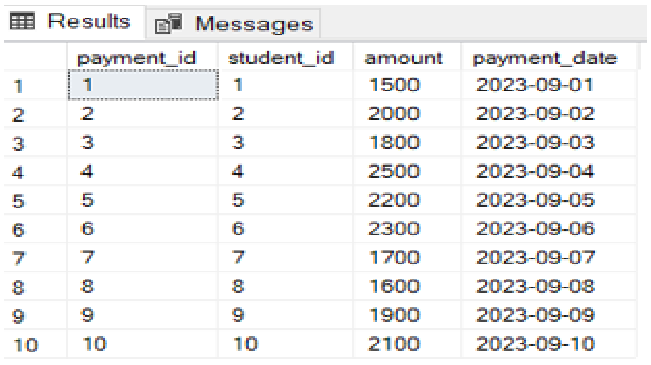
(7, 1700, '2023-09-07'),

(8, 1600, '2023-09-08'),

(9, 1900, '2023-09-09'),

(10, 2100, '2023-09-10');

Select * from Payments;



	payment_id	student_id	amount	payment_date
1	1	1	1500	2023-09-01
2	2	2	2000	2023-09-02
3	3	3	1800	2023-09-03
4	4	4	2500	2023-09-04
5	5	5	2200	2023-09-05
6	6	6	2300	2023-09-06
7	7	7	1700	2023-09-07
8	8	8	1600	2023-09-08
9	9	9	1900	2023-09-09
10	10	10	2100	2023-09-10

TASK-2 Select, where, between, AND, Like:

--1. Write an SQL query to insert a new student into the "Students" table with the following details:

--a. First Name: John

--b. Last Name: Doe

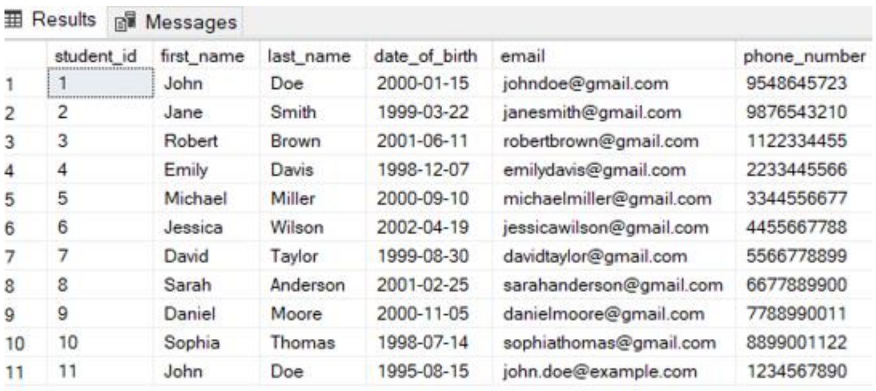
--c. Date of Birth: 1995-08-15

--d. Email: john.doe@example.com

--e. Phone Number: 1234567890

Insert into Students Values (11, '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	John	Doe	2000-01-15	johndoe@gmail.com	9548645723
2	2	Jane	Smith	1999-03-22	janesmith@gmail.com	9876543210
3	3	Robert	Brown	2001-06-11	robertbrown@gmail.com	1122334455
4	4	Emily	Davis	1998-12-07	emilydavis@gmail.com	2233445566
5	5	Michael	Miller	2000-09-10	michaelmiller@gmail.com	3344556677
6	6	Jessica	Wilson	2002-04-19	jessicawilson@gmail.com	4455667788
7	7	David	Taylor	1999-08-30	davidtaylor@gmail.com	5566778899
8	8	Sarah	Anderson	2001-02-25	sarahanderson@gmail.com	6677889900
9	9	Daniel	Moore	2000-11-05	danielmoore@gmail.com	7788990011
10	10	Sophia	Thomas	1998-07-14	sophiathomas@gmail.com	8899001122
11	11	John	Doe	1995-08-15	john.doe@example.com	1234567890

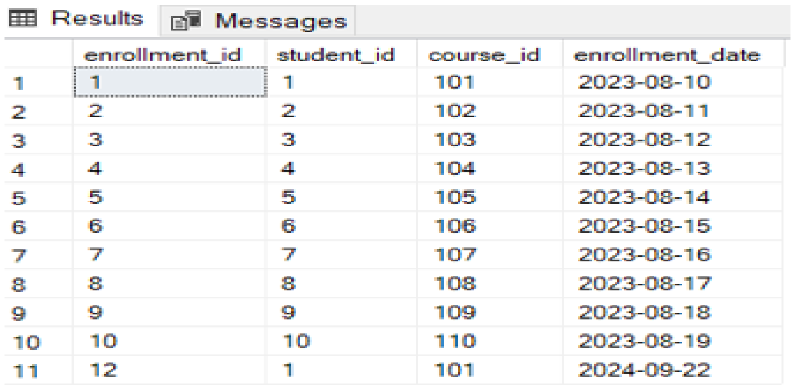
--2. Write an SQL query to enroll a student in a course. Choose an existing student and course and

--insert a record into the "Enrollments" table with the enrollment date.

INSERT INTO Enrollments(student_id, course_id, enrollment_date)

VALUES (1, 101, GETDATE());

Select * from Enrollments;



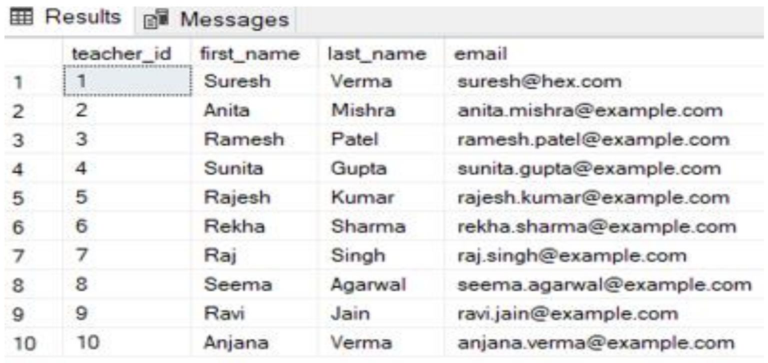
	enrollment_id	student_id	course_id	enrollment_date
1	1	1	101	2023-08-10
2	2	2	102	2023-08-11
3	3	3	103	2023-08-12
4	4	4	104	2023-08-13
5	5	5	105	2023-08-14
6	6	6	106	2023-08-15
7	7	7	107	2023-08-16
8	8	8	108	2023-08-17
9	9	9	109	2023-08-18
10	10	10	110	2023-08-19
11	12	1	101	2024-09-22

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

--modify their email address.

Update Teacher SET Email = 'suresh@hex.com' where teacher_id = 1;

Select * from Teacher;

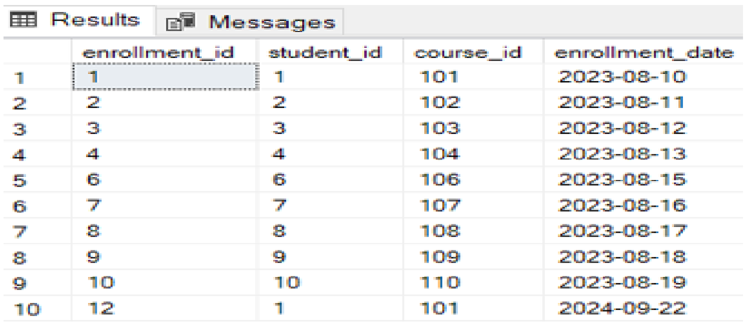


	teacher_id	first_name	last_name	email
1	1	Suresh	Verma	suresh@hex.com
2	2	Anita	Mishra	anita.mishra@example.com
3	3	Ramesh	Patel	ramesh.patel@example.com
4	4	Sunita	Gupta	sunita.gupta@example.com
5	5	Rajesh	Kumar	rajesh.kumar@example.com
6	6	Rekha	Sharma	rekha.sharma@example.com
7	7	Raj	Singh	raj.singh@example.com
8	8	Seema	Agarwal	seema.agarwal@example.com
9	9	Ravi	Jain	ravi.jain@example.com
10	10	Anjana	Verma	anjana.verma@example.com

--4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select --an enrollment record based on student and course.

Delete from Enrollments where student_id = 5 and course_id = 105;

select * from Enrollments;

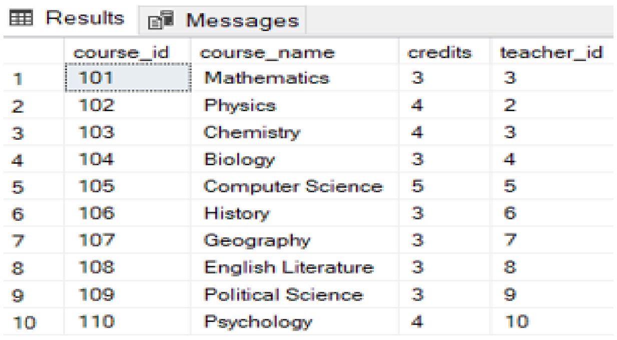


	enrollment_id	student_id	course_id	enrollment_date
1	1	1	101	2023-08-10
2	2	2	102	2023-08-11
3	3	3	103	2023-08-12
4	4	4	104	2023-08-13
5	6	6	106	2023-08-15
6	7	7	107	2023-08-16
7	8	8	108	2023-08-17
8	9	9	109	2023-08-18
9	10	10	110	2023-08-19
10	12	1	101	2024-09-22

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

UPDATE Courses SET teacher_id = 3 WHERE course_id = 101;

Select * from Courses;

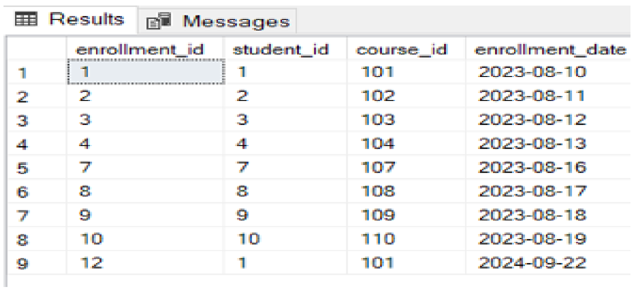


	course_id	course_name	credits	teacher_id
1	101	Mathematics	3	3
2	102	Physics	4	2
3	103	Chemistry	4	3
4	104	Biology	3	4
5	105	Computer Science	5	5
6	106	History	3	6
7	107	Geography	3	7
8	108	English Literature	3	8
9	109	Political Science	3	9
10	110	Psychology	4	10

--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.

Delete from Enrollments where student_id = 6;

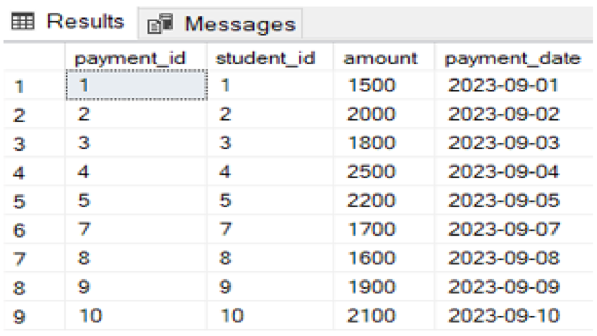
Select * from Enrollments;



	enrollment_id	student_id	course_id	enrollment_date
1	1	1	101	2023-08-10
2	2	2	102	2023-08-11
3	3	3	103	2023-08-12
4	4	4	104	2023-08-13
5	7	7	107	2023-08-16
6	8	8	108	2023-08-17
7	9	9	109	2023-08-18
8	10	10	110	2023-08-19
9	12	1	101	2024-09-22

DELETE FROM Payments WHERE student_id = 6;

Select * from Payments;

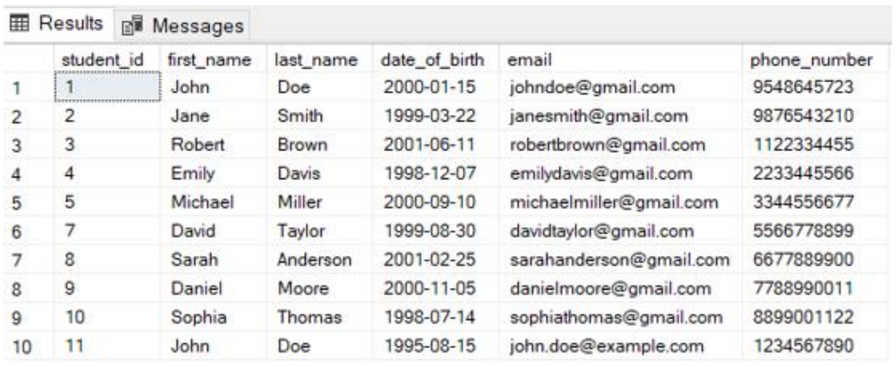


Results | Messages

	payment_id	student_id	amount	payment_date
1	1	1	1500	2023-09-01
2	2	2	2000	2023-09-02
3	3	3	1800	2023-09-03
4	4	4	2500	2023-09-04
5	5	5	2200	2023-09-05
6	7	7	1700	2023-09-07
7	8	8	1600	2023-09-08
8	9	9	1900	2023-09-09
9	10	10	2100	2023-09-10

Delete from Students where student_id = 6;

Select * from Students;



Results | Messages

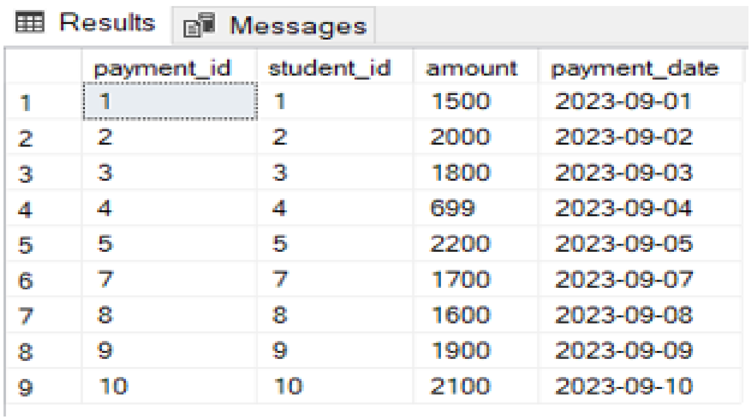
	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	John	Doe	2000-01-15	johndoe@gmail.com	9548645723
2	2	Jane	Smith	1999-03-22	janesmith@gmail.com	9876543210
3	3	Robert	Brown	2001-06-11	robertbrown@gmail.com	1122334455
4	4	Emily	Davis	1998-12-07	emilydavis@gmail.com	2233445566
5	5	Michael	Miller	2000-09-10	michaelmiller@gmail.com	3344556677
6	7	David	Taylor	1999-08-30	davidtaylor@gmail.com	5566778899
7	8	Sarah	Anderson	2001-02-25	sarahanderson@gmail.com	6677889900
8	9	Daniel	Moore	2000-11-05	danielmoore@gmail.com	7788990011
9	10	Sophia	Thomas	1998-07-14	sophiathomas@gmail.com	8899001122
10	11	John	Doe	1995-08-15	john.doe@example.com	1234567890

--7. Update the payment amount for specific payment record in the "Payments" table. Choose any

--payment record and modify the payment amount.

Update Payments SET amount = 699 where student_id = 4;

Select * from Payments;



Results | Messages

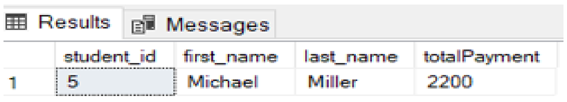
	payment_id	student_id	amount	payment_date
1	1	1	1500	2023-09-01
2	2	2	2000	2023-09-02
3	3	3	1800	2023-09-03
4	4	4	699	2023-09-04
5	5	5	2200	2023-09-05
6	7	7	1700	2023-09-07
7	8	8	1600	2023-09-08
8	9	9	1900	2023-09-09
9	10	10	2100	2023-09-10

TASK-3 Aggregate functions, Having, Order By, GroupBy and Joins:

--1. Write an SQL query to calculate the total payments made by a specific student. You will need

--to join the "Payments" table with the "Students" table based on the student's ID.

```
Select S.student_id, S.first_name, S.last_name, SUM(P.amount) AS totalPayment

from Students S JOIN Payments P

ON S.student_id = P.student_id

where S.student_id = 5

group by S.student_id, S.first_name, S.last_name;
```

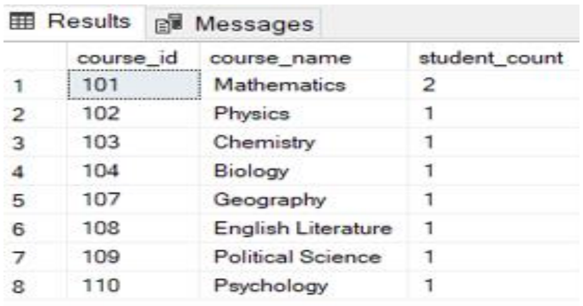


	student_id	first_name	last_name	totalPayment
1	5	Michael	Miller	2200

--2. Write an SQL query to retrieve list of courses along with the count of students enrolled in each

--course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

```
SELECT C.course_id, C.course_name, COUNT(E.student_id) AS student_count

FROM Courses C JOIN Enrollments E

ON C.course_id = E.course_id

GROUP BY C.course_id, C.course_name;
```



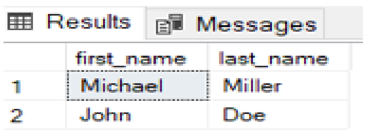
	course_id	course_name	student_count
1	101	Mathematics	2
2	102	Physics	1
3	103	Chemistry	1
4	104	Biology	1
5	107	Geography	1
6	108	English Literature	1
7	109	Political Science	1
8	110	Psychology	1

--3. Write an SQL query to find the names of students who have not enrolled in any course. Use a

--LEFT JOIN between the "Students" table and the "Enrollments" table to identify students

--without enrollments.

```
Select S.first_name, S.last_name

from Students S LEFT JOIN Enrollments E

ON S.student_id = E.student_id

WHERE E.student_id IS NULL;
```



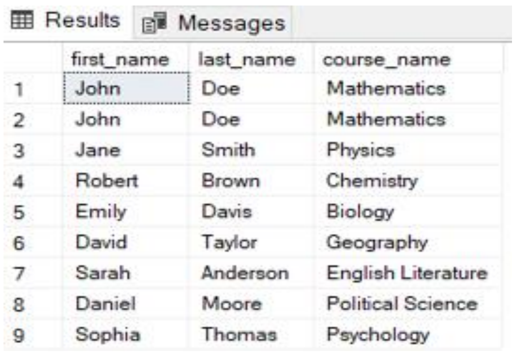
	first_name	last_name
1	Michael	Miller
2	John	Doe

--4. Write an SQL query to retrieve the first name, last name of students, and the names of the

--courses they are enrolled in. Use JOIN operations between the "Students" table and the

--"Enrollments" and "Courses" tables.

```
Select S.first_name, S.last_name, C.course_name

from Students S JOIN Enrollments E

ON S.student_id = E.student_id

JOIN Courses C

ON E.course_id = C.course_id;
```

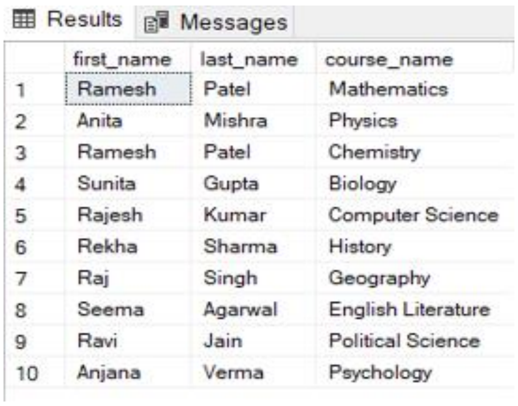


	first_name	last_name	course_name
1	John	Doe	Mathematics
2	John	Doe	Mathematics
3	Jane	Smith	Physics
4	Robert	Brown	Chemistry
5	Emily	Davis	Biology
6	David	Taylor	Geography
7	Sarah	Anderson	English Literature
8	Daniel	Moore	Political Science
9	Sophia	Thomas	Psychology

--5. Create a query to list the names of teachers and the courses they are assigned to. Join the

--"Teacher" table with the "Courses" table.

```
Select T.first_name, T.last_name, C.course_name

from Teacher T JOIN Courses C

ON T.teacher_id = C.teacher_id;
```

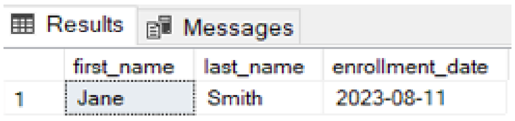


	first_name	last_name	course_name
1	Ramesh	Patel	Mathematics
2	Anita	Mishra	Physics
3	Ramesh	Patel	Chemistry
4	Sunita	Gupta	Biology
5	Rajesh	Kumar	Computer Science
6	Rekha	Sharma	History
7	Raj	Singh	Geography
8	Seema	Agarwal	English Literature
9	Ravi	Jain	Political Science
10	Anjana	Verma	Psychology

--6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join

--the "Students" table with the "Enrollments" and "Courses" tables.

```
SELECT S.first_name, S.last_name, E.enrollment_date

FROM Students S

JOIN Enrollments E ON S.student_id = E.student_id

JOIN Courses C ON E.course_id = C.course_id

WHERE C.course_id = 102;
```

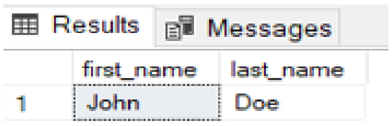

Results | Messages

	first_name	last_name	enrollment_date
1	Jane	Smith	2023-08-11

--7. Find the names of students who have not made any payments. Use a LEFT JOIN between the

--"Students" table and the "Payments" table and filter for students with NULL payment records.

```
Select S.first_name, S.last_name

from Students S LEFT JOIN Payments P

ON S.student_id = P.student_id

where P.student_id IS NULL;
```


Results | Messages

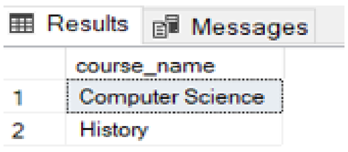
	first_name	last_name
1	John	Doe

--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.

```
SELECT C.course_name

FROM Courses C LEFT JOIN Enrollments E

ON C.course_id = E.course_id

WHERE E.course_id IS NULL;
```


Results | Messages

	course_name
1	Computer Science
2	History

--9. Identify students who are enrolled in more than one course. Use a self-join on the

--"Enrollments" table to find students with multiple enrollment records.

SELECT S.student_id, S.first_name, S.last_name, COUNT(E1.course_id) AS course_count

FROM Students S JOIN Enrollments E1

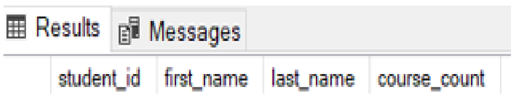
ON S.student_id = E1.student_id

JOIN Enrollments E2

ON E1.student_id = E2.student_id AND E1.course_id != E2.course_id

GROUP BY S.student_id, S.first_name, S.last_name

HAVING COUNT(E1.course_id) > 1;



student_id	first_name	last_name	course_count

--10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher"

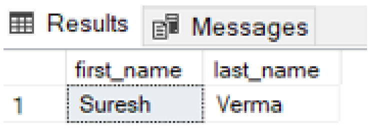
--table and the "Courses" table and filter for teachers with NULL course assignments.

SELECT T.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;



	first_name	last_name
1	Suresh	Verma

TASK-4 Subquery and its type:

--1. Write an SQL query to calculate the average number of students enrolled in each course. Use

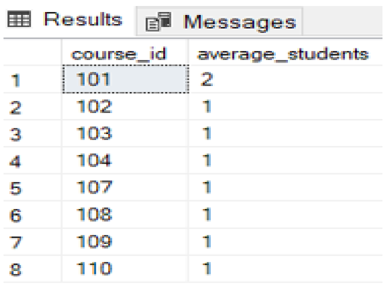
--aggregate functions and subqueries to achieve this.

SELECT course_id, AVG(student_count) as average_students

FROM (SELECT course_id, COUNT(student_id) as student_count

FROM Enrollments GROUP BY course_id) AS SUB

GROUP BY course_id;


Results Messages

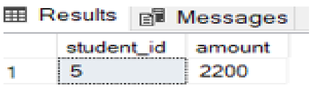
	course_id	average_students
1	101	2
2	102	1
3	103	1
4	104	1
5	107	1
6	108	1
7	109	1
8	110	1

--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.

Select student_id, amount from Payments

where amount = (Select MAX(amount) from Payments);


Results Messages

	student_id	amount
1	5	2200

--3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the

--course(s) with the maximum enrollment count.

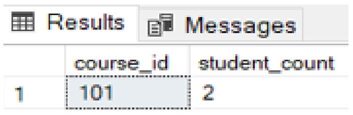
SELECT course_id, COUNT(*) as student_count

FROM Enrollments GROUP BY course_id

HAVING COUNT(*) = (SELECT MAX(student_count)

FROM(SELECT course_id, COUNT(*) as student_count

FROM Enrollments GROUP BY course_id) AS subquery);


Results Messages

	course_id	student_count
1	101	2

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

--payments for each teacher's courses.

SELECT T.teacher_id, SUM(P.amount) as total_payments

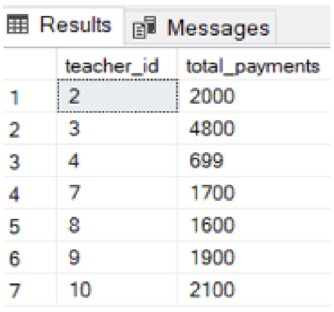
FROM Teacher T JOIN Courses C

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;

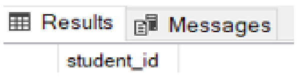

Results | Messages

	teacher_id	total_payments
1	2	2000
2	3	4800
3	4	699
4	7	1700
5	8	1600
6	9	1900
7	10	2100

--5. Identify students who are enrolled in all available courses. Use subqueries to compare a

--student's enrollments with the total number of courses.

```
SELECT student_id FROM Enrollments

GROUP BY student_id

HAVING COUNT(course_id) = (SELECT COUNT(*) FROM Courses);
```

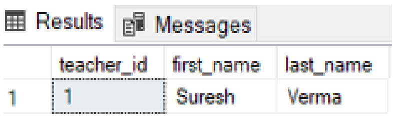

Results | Messages

	student_id

--6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to

--find teachers with no course assignments.

```
SELECT teacher_id, first_name, last_name

FROM Teacher

WHERE teacher_id NOT IN (

SELECT teacher_id FROM Courses);
```

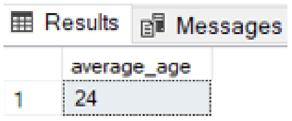

Results | Messages

	teacher_id	first_name	last_name
1	1	Suresh	Verma

--7. Calculate the average age of all students. Use subqueries to calculate the age of each student

--based on their date of birth.

```
SELECT AVG(age) as average_age from

(SELECT DATEDIFF(YEAR, date_of_birth, GETDATE()) as age

FROM Students) AS subquery;
```


Results | Messages

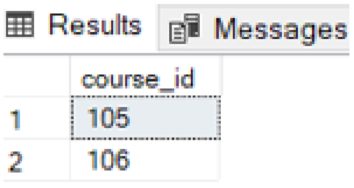
	average_age
1	24

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

--records.

Select course_id from Courses

where course_id not in (select course_id from Enrollments);



Results Messages

	course_id
1	105
2	106

--9. Calculate the total payments made by each student for each course they are enrolled in. Use

--subqueries and aggregate functions to sum payments.

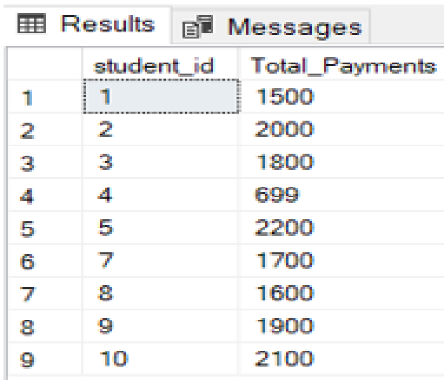
SELECT student_id, SUM(amount) AS Total_Payments

FROM Payments WHERE EXISTS

(SELECT 1 FROM Students

WHERE Students.student_id = Payments.student_id)

GROUP BY student_id;



Results Messages

	student_id	Total_Payments
1	1	1500
2	2	2000
3	3	1800
4	4	699
5	5	2200
6	7	1700
7	8	1600
8	9	1900
9	10	2100

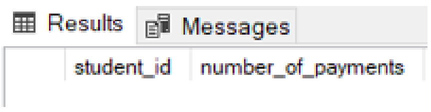
--10. Identify students who have made more than one payment. Use subqueries and aggregate

--functions to count payments per student and filter for those with counts greater than one.

Select P.student_id, count(*) as number_of_payments from Payments P

group by P.student_id

having count(*) > 1;


Results | Messages

student_id	number_of_payments

--11. Write an SQL query to calculate the total payments made by each student. Join the

--"Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments

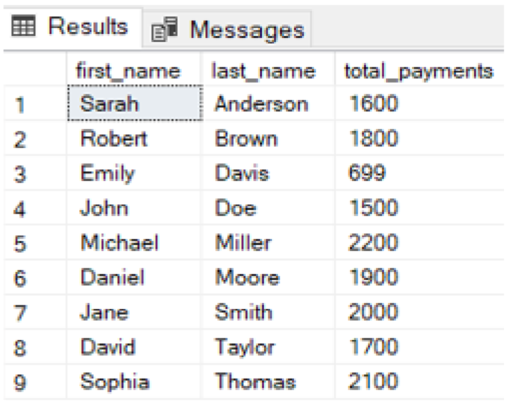
--for each student.

SELECT S.first_name, S.last_name, 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;


Results | Messages

	first_name	last_name	total_payments
1	Sarah	Anderson	1600
2	Robert	Brown	1800
3	Emily	Davis	699
4	John	Doe	1500
5	Michael	Miller	2200
6	Daniel	Moore	1900
7	Jane	Smith	2000
8	David	Taylor	1700
9	Sophia	Thomas	2100

--12. Retrieve a list of course names along with the count of students enrolled in each course. Use

--JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to

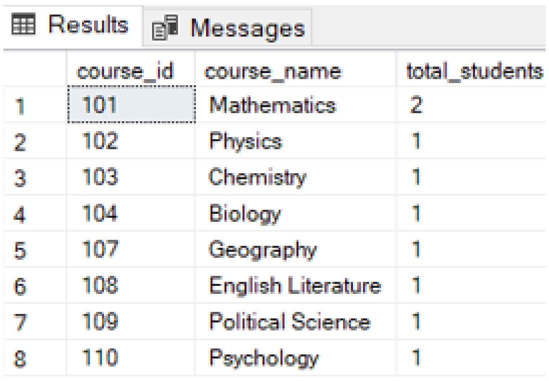
--count enrollments.

Select C.course_id, C.course_name, Count(E.student_id) as total_students

from Courses C JOIN Enrollments E

ON C.course_id = E.course_id

group by C.course_id, C.course_name;

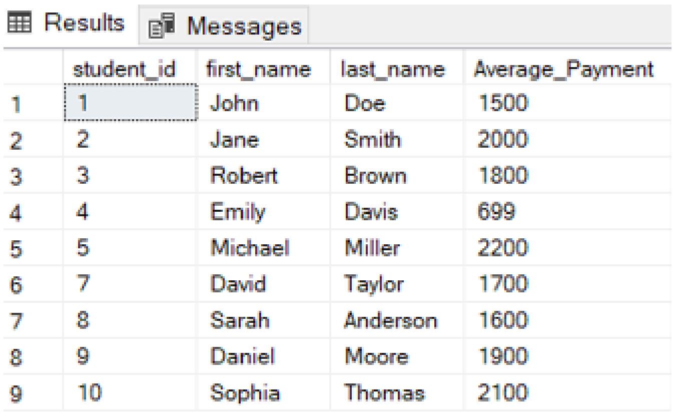

Results | Messages

	course_id	course_name	total_students
1	101	Mathematics	2
2	102	Physics	1
3	103	Chemistry	1
4	104	Biology	1
5	107	Geography	1
6	108	English Literature	1
7	109	Political Science	1
8	110	Psychology	1

--13. Calculate the average payment amount made by students. Use JOIN operations between the

--"Students" table and the "Payments" table and GROUP BY to calculate the average.

```
Select S.student_id, 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.student_id, S.first_name, S.last_name;
```



	student_id	first_name	last_name	Average_Payment
1	1	John	Doe	1500
2	2	Jane	Smith	2000
3	3	Robert	Brown	1800
4	4	Emily	Davis	699
5	5	Michael	Miller	2200
6	7	David	Taylor	1700
7	8	Sarah	Anderson	1600
8	9	Daniel	Moore	1900
9	10	Sophia	Thomas	2100