STUDENT INFORMATION SYSTEM (SIS)
-AAYUSHI GUPTA

TASK 1 : Database Design :

1. Create the database named "SISDB".

Ans.

```
CREATE DATABASE SISDB;
USE SISDB;
```

```
Messages
Commands completed successfully.

Completion time: 2024-09-20T16:29:36.8331316+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 TABLE Students(
 student_id VARCHAR(10) PRIMARY KEY,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   date_of_birth DATE,
   email VARCHAR(100),
   phone_number VARCHAR(15)
 );

 CREATE TABLE Courses(
 course_id VARCHAR(10) PRIMARY KEY,
 course_name VARCHAR(100),
 credits INT,
 teacher_id VARCHAR(10) FOREIGN KEY REFERENCES Teacher(teacher_id)
 );

 CREATE TABLE Enrollments (
   enrollment_id VARCHAR(10) PRIMARY KEY,
   student_id VARCHAR(10) FOREIGN KEY REFERENCES Students(student_id),
   course_id VARCHAR(10) FOREIGN KEY REFERENCES Courses(course_id),
   enrollment_date DATE
);
```

```sql
CREATE TABLE Teacher (
    teacher_id VARCHAR(10) PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100)
);

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

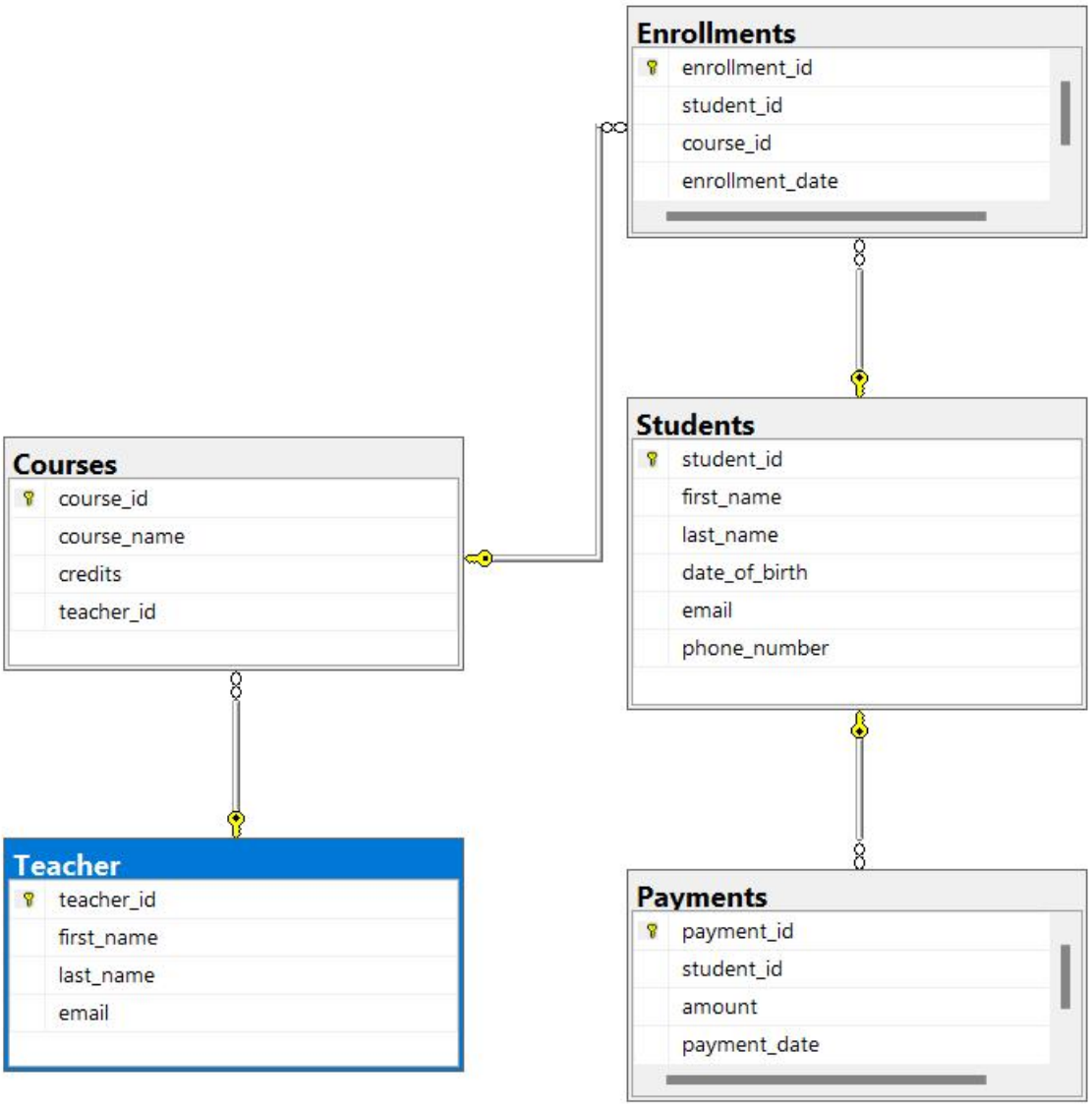
Messages

```
Commands completed successfully.

Completion time: 2024-09-20T16:47:42.0449828+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. Yes , I have already defined primary and foreign key constraints in the SQL script above.

5. Insert at least 10 sample records into each of the following tables.
   i. Students
   ii. Courses
   iii. Enrollments
   iv. Teacher
   v. Payments

Ans. (i)Students

```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number) VALUES
('S101', 'Aayushi', 'Gupta', '2001-02-18', 'aayushi@gmail.com', '9686129348'),
('S102', 'Alice', 'Smith', '1996-07-20', 'alice.smith@example.com', '9876543210'),
('S103', 'Bob', 'Johnson', '1997-03-12', 'bob.johnson@example.com', '1122334455'),
('S104', 'Charlie', 'Brown', '1995-11-22', 'charlie.brown@example.com', '5566778899'),
('S105', 'Diana', 'Miller', '1996-02-28', 'diana.miller@example.com', '6677889900'),
('S106', 'Eva', 'Wilson', '1997-09-05', 'eva.wilson@example.com', '2233445566'),
('S107', 'Frank', 'Garcia', '1995-12-12', 'frank.garcia@example.com', '7788990011'),
('S108', 'Grace', 'Martinez', '1996-04-04', 'grace.martinez@example.com', '8899001122'),
('S109', 'Henry', 'Davis', '1997-06-18', 'henry.davis@example.com', '3344556677'),
('S110', 'Ivy', 'Rodriguez', '1995-10-10', 'ivy.rodriguez@example.com', '9988776655');
```

```
Messages

(10 rows affected)

Completion time: 2024-09-21T11:20:29.1751501+05:30
```

(ii)Courses

```
INSERT INTO Courses (course_id, course_name, credits, teacher_id) VALUES
('C101', 'Introduction to Programming', 4, 'T101'),
('C102', 'Mathematics 101', 3, 'T102'),
('C103', 'Physics 101', 4, 'T103'),
('C104', 'Chemistry 101', 4, 'T104'),
('C105', 'English Literature', 2, 'T105'),
('C106', 'History of Art', 3, 'T106'),
('C107', 'Advanced Calculus', 4, 'T102'),
('C108', 'Database Management', 4, 'T107'),
('C109', 'Operating Systems', 4, 'T101'),
('C110', 'Network Security', 3, 'T108');
```

Messages

```
(10 rows affected)

Completion time: 2024-09-21T11:20:29.1751501+05:30
```

(iii)Enrollments

```
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
('E101', 'S101', 'C101', '2023-01-15'),
('E102', 'S102', 'C102', '2023-02-20'),
('E103', 'S103', 'C103', '2023-03-12'),
('E104', 'S104', 'C104', '2023-04-25'),
('E105', 'S105', 'C105', '2023-05-10'),
('E106', 'S106', 'C106', '2023-06-22'),
('E107', 'S107', 'C107', '2023-07-30'),
('E108', 'S108', 'C108', '2023-08-05'),
('E109', 'S109', 'C109', '2023-09-15'),
('E110', 'S110', 'C110', '2023-10-20');
```

Messages

```
(10 rows affected)

Completion time: 2024-09-21T11:20:29.1751501+05:30
```

(iv)Teacher

```
INSERT INTO Teacher (teacher_id, first_name, last_name, email) VALUES
('T101', 'Sarah', 'Johnson', 'sarah.johnson@example.com'),
('T102', 'Mark', 'Smith', 'mark.smith@example.com'),
('T103', 'Emily', 'Davis', 'emily.davis@example.com'),
('T104', 'James', 'Brown', 'james.brown@example.com'),
('T105', 'Laura', 'Garcia', 'laura.garcia@example.com'),
('T106', 'Daniel', 'Martinez', 'daniel.martinez@example.com'),
('T107', 'Susan', 'Wilson', 'susan.wilson@example.com'),
('T108', 'Michael', 'Rodriguez', 'michael.rodriguez@example.com'),
('T109', 'Aditi', 'Singh', 'aditi.singh@example.com'),
('T110', 'Shreyash', 'Shukla', 'shreyash.shukla@example.com');
```

Messages

```
(10 rows affected)

Completion time: 2024-09-21T11:20:29.1751501+05:30
```

(v)Payments

```
INSERT INTO Payments (payment_id, student_id, amount, payment_date) VALUES
('P101', 'S101', 500.00, '2023-01-20'),
('P102', 'S102', 300.00, '2023-02-25'),
('P103', 'S103', 450.00, '2023-03-18'),
('P104', 'S104', 600.00, '2023-04-30'),
('P105', 'S105', 350.00, '2023-05-12'),
('P106', 'S106', 400.00, '2023-06-24'),
('P107', 'S107', 550.00, '2023-07-31'),
('P108', 'S108', 650.00, '2023-08-10'),
('P109', 'S109', 475.00, '2023-09-18'),
('P110', 'S110', 525.00, '2023-10-22');
```

Messages

```
(10 rows affected)

Completion time: 2024-09-21T11:20:29.1751501+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 VALUES ('S111','John','Doe','1995-08-15','john.doe@example.com','1234567890');
```

Messages

```
(1 row affected)

Completion time: 2024-09-21T11:37:12.9500875+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
VALUES('E111','S101','C101','2024-09-10');
```

Messages

```
(1 row affected)

Completion time: 2024-09-21T11:43:47.4187913+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='marki@gmail.com'
where teacher_id='T102' ;
```

```
Messages

(1 row affected)

Completion time: 2024-09-21T11:44:45.6234767+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 student_id='S102' and course_id='C102';
```

```
Messages

(1 row affected)

Completion time: 2024-09-21T11:45:41.9052768+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='T101'
where course_id='C106';
```

```
Messages

(1 row affected)

Completion time: 2024-09-21T11:46:24.3146102+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. In order to maintain referential integrity following steps are to be followed

- -Delete the enrollments for the student.
- - Delete any payments associated with the student.
- - Delete the student.

```
DELETE FROM Enrollments
WHERE student_id = 'S103';
```

Messages

```
(1 row affected)

Completion time: 2024-09-21T11:51:01.7732858+05:30
```

```
DELETE FROM Payments
WHERE student_id = 'S103';
```

Messages

```
(1 row affected)

Completion time: 2024-09-21T11:54:09.6555344+05:30
```

```
DELETE FROM Students
WHERE student_id = 'S103';
```

Messages

```
(1 row affected)

Completion time: 2024-09-21T11:54:37.1665889+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 = 600.00
WHERE payment_id = 'P101';
```

Messages

```
(1 row affected)

Completion time: 2024-09-21T11:59:44.4723073+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.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 = 'S107'
GROUP BY s.first_name, s.last_name, s.student_id;
```

Results | Messages

	first_name	last_name	total_payments
1	Frank	Garcia	550.00

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_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_name;
```

Results | Messages

	course_name	student_count
1	Advanced Calculus	1
2	Chemistry 101	1
3	Database Management	1
4	English Literature	1
5	History of Art	1
6	Introduction to Programming	2
7	Network Security	1
8	Operating Systems	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;
```

Results | Messages

	first_name	last_name
1	Alice	Smith
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.

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;
```

Results | Messages

	first_name	last_name	course_name
1	Aayushi	Gupta	Introduction to Programming
2	Aayushi	Gupta	Introduction to Programming
3	Charlie	Brown	Chemistry 101
4	Diana	Miller	English Literature
5	Eva	Wilson	History of Art
6	Frank	Garcia	Advanced Calculus
7	Grace	Martinez	Database Management
8	Henry	Davis	Operating Systems
9	Ivy	Rodriguez	Network Security

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;
```

Results | Messages

	first_name	last_name	course_name
1	Sarah	Johnson	Introduction to Programming
2	Mark	Smith	Mathematics 101
3	Emily	Davis	Physics 101
4	James	Brown	Chemistry 101
5	Laura	Garcia	English Literature
6	Sarah	Johnson	History of Art
7	Mark	Smith	Advanced Calculus
8	Susan	Wilson	Database Management
9	Sarah	Johnson	Operating Systems
10	Michael	Rodriguez	Network Security

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
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE c.course_name = 'English Literature';
```

Results | Messages

	first_name	last_name	enrollment_date
1	Diana	Miller	2023-05-10

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
FROM Students s
LEFT JOIN Payments p
 ON s.student_id = p.student_id
WHERE p.payment_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.

Ans.

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

Results | Messages

	course_name
1	Mathematics 101
2	Physics 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.first_name, s.last_name, COUNT(e.course_id) AS course_count
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY s.first_name, s.last_name
HAVING COUNT(e.course_id) > 1;
```

Results | Messages

	first_name	last_name	course_count
1	Aayushi	Gupta	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.first_name, t.last_name
```

Results | Messages

	first_name	last_name
1	Daniel	Martinez
2	Aditi	Singh
3	Shreyash	Shukla

```
FROM Teacher t
LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
WHERE c.course_id IS NULL;
```

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 AVG(enrol_count) AS Average_students_per_course
FROM (
   SELECT COUNT(e.student_id) AS enrol_count
   FROM Enrollments e
   GROUP BY e.course_id
) AS course_enrol;
```

Results | Messages

	Average_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.

Ans.

```
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);
```

Results | Messages

	first_name	last_name	amount
1	Grace	Martinez	650.00

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

Ans.

```
SELECT course_name
```

```
FROM Courses c
WHERE c.course_id = (SELECT TOP 1 e.course_id
                     FROM Enrollments e
                     GROUP BY e.course_id
                     ORDER BY COUNT(e.student_id) DESC);
```

Results | Messages

	course_name
1	Introduction to Programming

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

Ans.

```
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 course_id
             FROM Courses
             WHERE teacher_id = t.teacher_id)) AS total_payments
FROM Teacher t;
```

Results | Messages

	first_name	last_name	total_payments
1	Sarah	Johnson	2075.00
2	Mark	Smith	550.00
3	Emily	Davis	NULL
4	James	Brown	600.00
5	Laura	Garcia	350.00
6	Daniel	Martinez	NULL
7	Susan	Wilson	650.00
8	Michael	Rodriguez	525.00
9	Aditi	Singh	NULL
10	Shreyash	Shukla	NULL

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 S.student_id, S.first_name, S.last_name
FROM Students S
JOIN Enrollments E ON S.student_id = E.student_id
```

```
GROUP BY S.student_id, S.first_name, S.last_name
HAVING COUNT(DISTINCT E.course_id) = (SELECT COUNT(*) FROM Courses);
```

Results | Messages

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 t.first_name, t.last_name
FROM Teacher t
WHERE NOT EXISTS (SELECT 1
                  FROM Courses c
                  WHERE c.teacher_id = t.teacher_id);
```

Results | Messages

	first_name	last_name
1	Daniel	Martinez
2	Aditi	Singh
3	Shreyash	Shukla

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(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age
FROM Students;
```

Results | Messages

	average_age
1	27

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

Ans.

```
SELECT course_name
FROM Courses c
WHERE NOT EXISTS (SELECT 1
                  FROM Enrollments e
                  WHERE e.course_id = c.course_id);
```

Results | Messages

	course_name
1	Mathematics 101
2	Physics 101

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.

```
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
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name, c.course_name;
```

Results | Messages

	first_name	last_name	course_name	total_payments
1	Aayushi	Gupta	Introduction to Programming	1200.00
2	Charlie	Brown	Chemistry 101	600.00
3	Diana	Miller	English Literature	350.00
4	Eva	Wilson	History of Art	400.00
5	Frank	Garcia	Advanced Calculus	550.00
6	Grace	Martinez	Database Management	650.00
7	Henry	Davis	Operating Systems	475.00
8	Ivy	Rodriguez	Network Security	525.00

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.

```
SELECT s.first_name, s.last_name, COUNT(p.payment_id) AS payment_count
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name
HAVING COUNT(p.payment_id) > 1;
```

Results | Messages

	first_name	last_name	payment_count

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.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	Charlie	Brown	600.00
2	Henry	Davis	475.00
3	Frank	Garcia	550.00
4	Aayushi	Gupta	600.00
5	Grace	Martinez	650.00
6	Diana	Miller	350.00
7	Ivy	Rodriguez	525.00
8	Alice	Smith	300.00
9	Eva	Wilson	400.00

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.

Ans.

```
SELECT 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_name;
```

Results | Messages

	course_name	student_count
1	Advanced Calculus	1
2	Chemistry 101	1
3	Database Management	1
4	English Literature	1
5	History of Art	1
6	Introduction to Programming	2
7	Network Security	1
8	Operating Systems	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.

```
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;
```

Results | Messages

	student_id	first_name	last_name	average_payment
1	S101	Aayushi	Gupta	600.000000
2	S102	Alice	Smith	300.000000
3	S104	Charlie	Brown	600.000000
4	S105	Diana	Miller	350.000000
5	S106	Eva	Wilson	400.000000
6	S107	Frank	Garcia	550.000000
7	S108	Grace	Martinez	650.000000
8	S109	Henry	Davis	475.000000
9	S110	Ivy	Rodriguez	525.000000