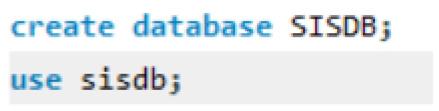
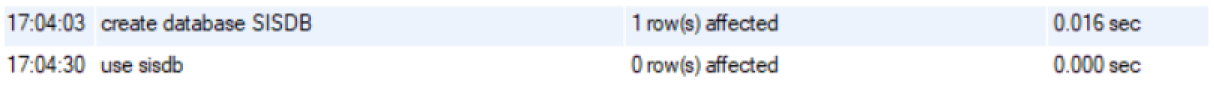
Assignment – 2 Student Information System (SQL)

Task 1:

1. Create the database named "SISDB"



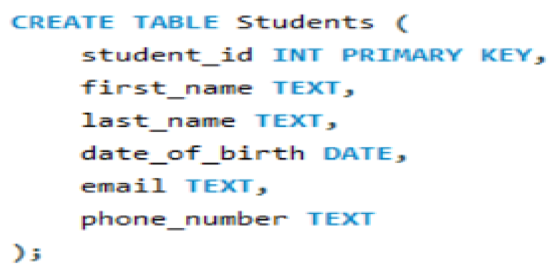
```sql
create database SISDB;
use sisdb;
```



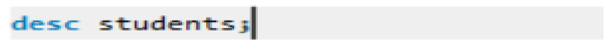
```
17:04:03  create database SISDB    1 row(s) affected    0.016 sec
17:04:30  use sisdb                0 row(s) affected    0.000 sec
```

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.

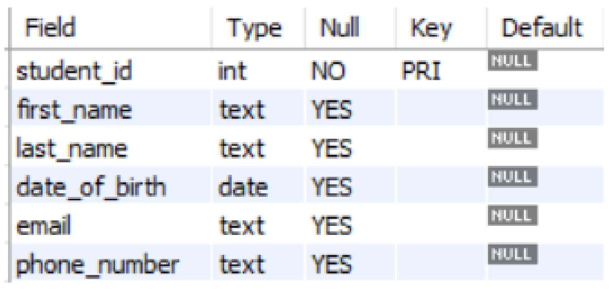
a. Students



```sql
CREATE TABLE Students (
     student_id INT PRIMARY KEY,
     first_name TEXT,
     last_name TEXT,
     date_of_birth DATE,
     email TEXT,
     phone_number TEXT
);
```

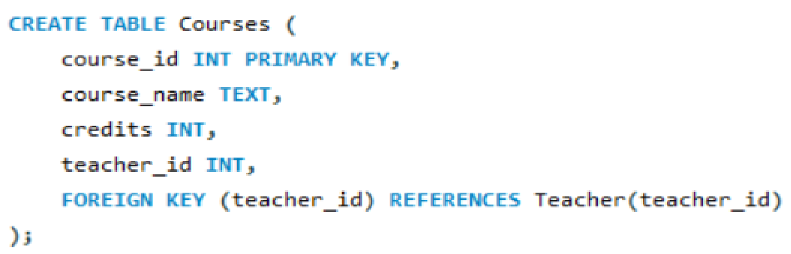


```sql
desc students;
```

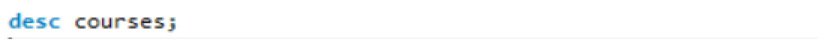


Field	Type	Null	Key	Default
student_id	int	NO	PRI	NULL
first_name	text	YES		NULL
last_name	text	YES		NULL
date_of_birth	date	YES		NULL
email	text	YES		NULL
phone_number	text	YES		NULL

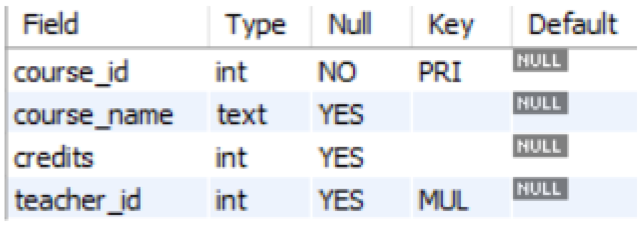
b. Courses



```sql
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name TEXT,
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);
```

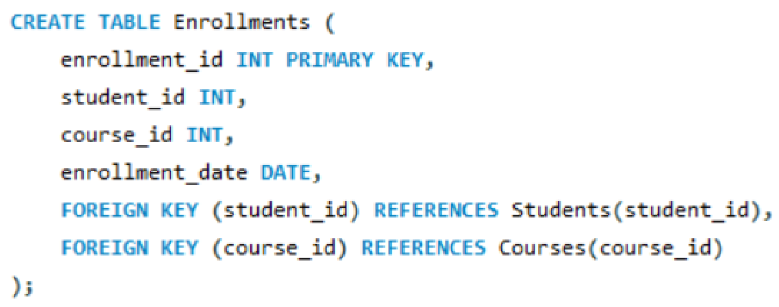


```sql
desc courses;
```

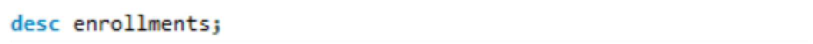


Field	Type	Null	Key	Default
course_id	int	NO	PRI	NULL
course_name	text	YES		NULL
credits	int	YES		NULL
teacher_id	int	YES	MUL	NULL

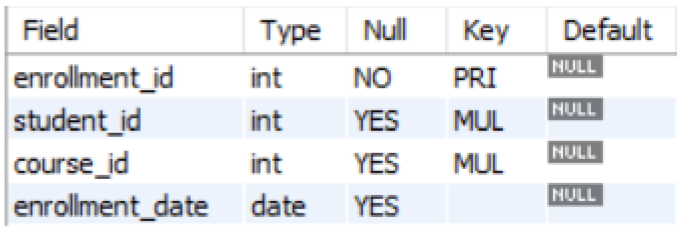
c. Enrollments



```
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)
);
```

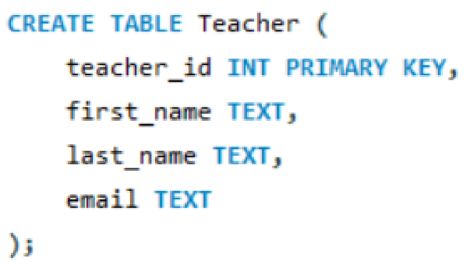


```
desc enrollments;
```

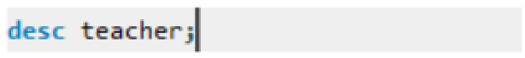


Field	Type	Null	Key	Default
enrollment_id	int	NO	PRI	NULL
student_id	int	YES	MUL	NULL
course_id	int	YES	MUL	NULL
enrollment_date	date	YES		NULL

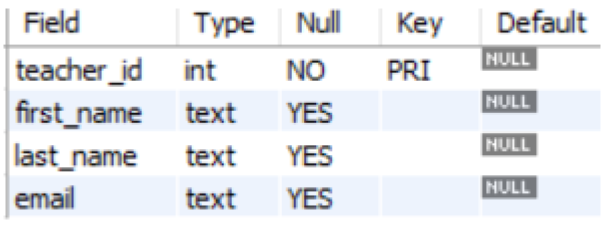
d. Teacher



```
CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY,
    first_name TEXT,
    last_name TEXT,
    email TEXT
);
```



```
desc teacher;
```

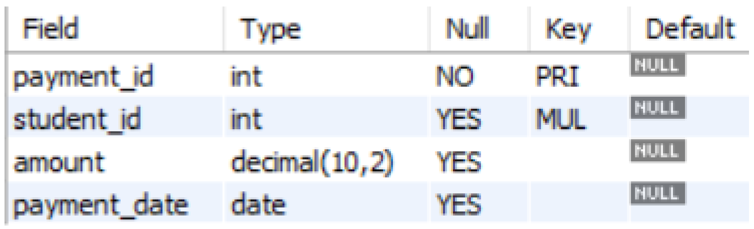


Field	Type	Null	Key	Default
teacher_id	int	NO	PRI	NULL
first_name	text	YES		NULL
last_name	text	YES		NULL
email	text	YES		NULL

e. Payments

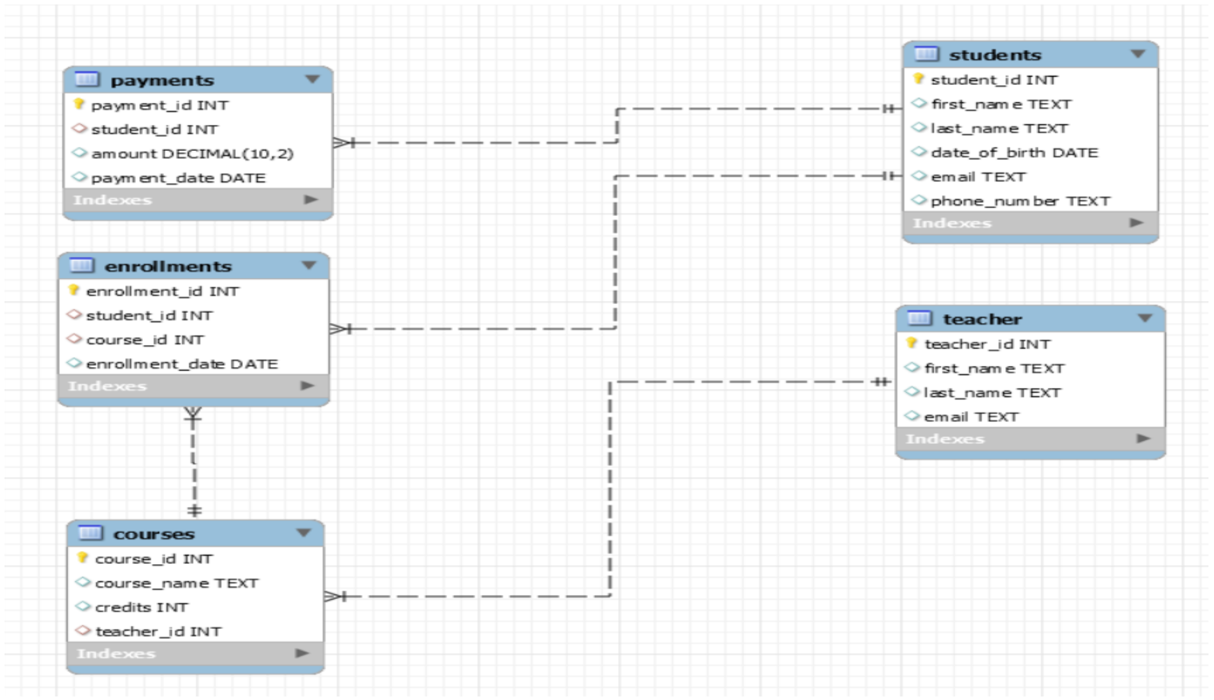
```
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY,
    student_id INT,
    amount DECIMAL(10, 2),
    payment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

```
desc payments;
```

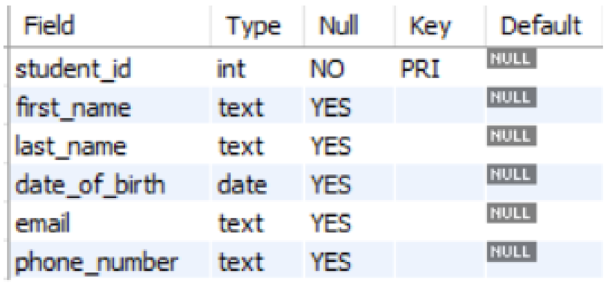


Field	Type	Null	Key	Default
payment_id	int	NO	PRI	NULL
student_id	int	YES	MUL	NULL
amount	decimal(10,2)	YES		NULL
payment_date	date	YES		NULL

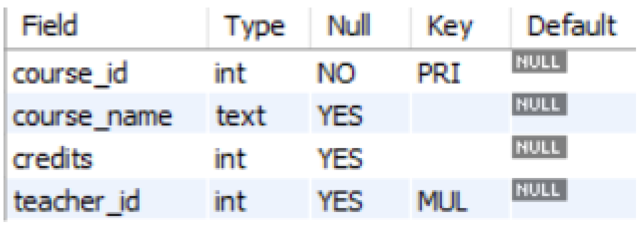
3. Create an ERD (Entity Relationship Diagram) for the database.



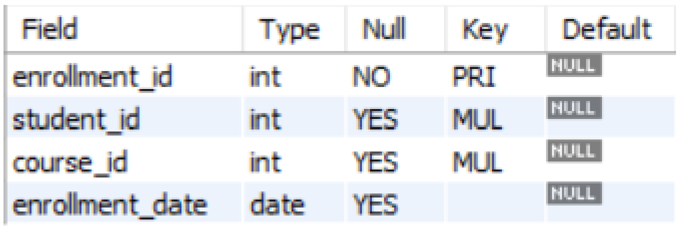
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



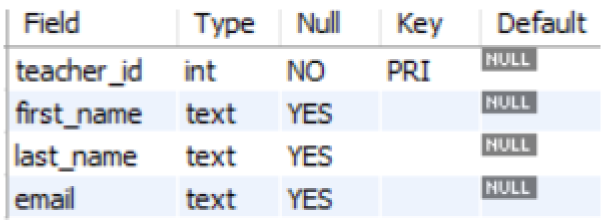
Field	Type	Null	Key	Default
student_id	int	NO	PRI	NULL
first_name	text	YES		NULL
last_name	text	YES		NULL
date_of_birth	date	YES		NULL
email	text	YES		NULL
phone_number	text	YES		NULL



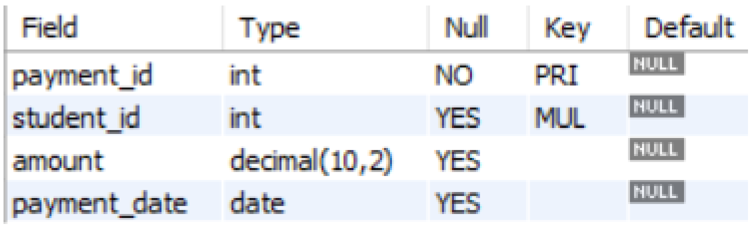
Field	Type	Null	Key	Default
course_id	int	NO	PRI	NULL
course_name	text	YES		NULL
credits	int	YES		NULL
teacher_id	int	YES	MUL	NULL



Field	Type	Null	Key	Default
enrollment_id	int	NO	PRI	NULL
student_id	int	YES	MUL	NULL
course_id	int	YES	MUL	NULL
enrollment_date	date	YES		NULL



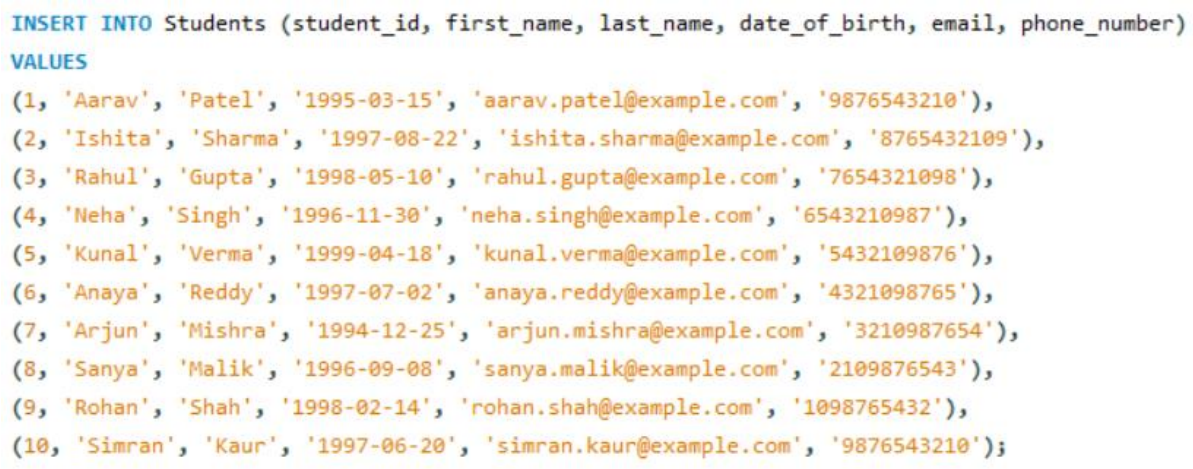
Field	Type	Null	Key	Default
teacher_id	int	NO	PRI	NULL
first_name	text	YES		NULL
last_name	text	YES		NULL
email	text	YES		NULL



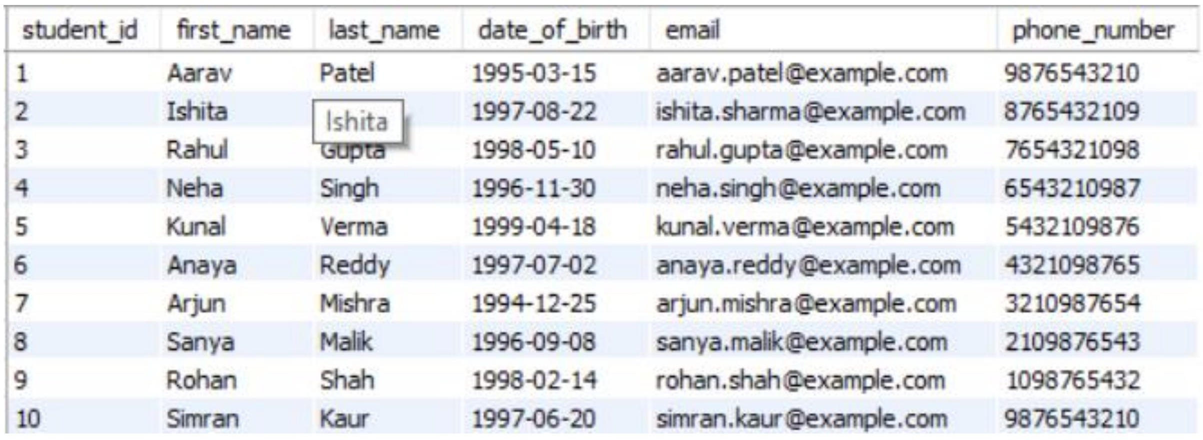
Field	Type	Null	Key	Default
payment_id	int	NO	PRI	NULL
student_id	int	YES	MUL	NULL
amount	decimal(10,2)	YES		NULL
payment_date	date	YES		NULL

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

i. Students

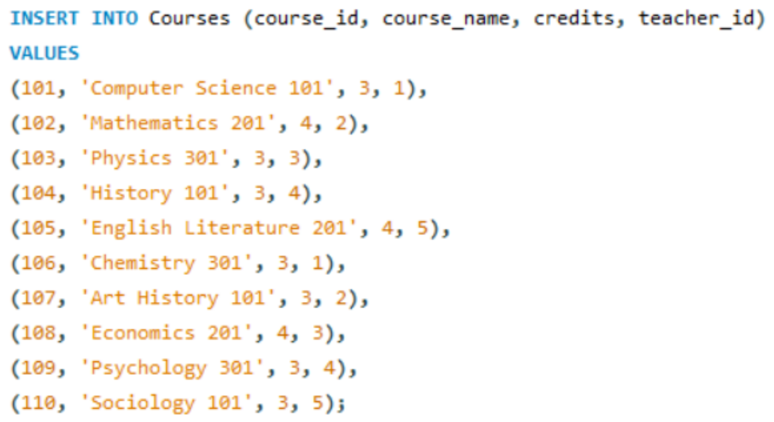


```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
VALUES
(1, 'Aarav', 'Patel', '1995-03-15', 'aarav.patel@example.com', '9876543210'),
(2, 'Ishita', 'Sharma', '1997-08-22', 'ishita.sharma@example.com', '8765432109'),
(3, 'Rahul', 'Gupta', '1998-05-10', 'rahul.gupta@example.com', '7654321098'),
(4, 'Neha', 'Singh', '1996-11-30', 'neha.singh@example.com', '6543210987'),
(5, 'Kunal', 'Verma', '1999-04-18', 'kunal.verma@example.com', '5432109876'),
(6, 'Anaya', 'Reddy', '1997-07-02', 'anaya.reddy@example.com', '4321098765'),
(7, 'Arjun', 'Mishra', '1994-12-25', 'arjun.mishra@example.com', '3210987654'),
(8, 'Sanya', 'Malik', '1996-09-08', 'sanya.malik@example.com', '2109876543'),
(9, 'Rohan', 'Shah', '1998-02-14', 'rohan.shah@example.com', '1098765432'),
(10, 'Simran', 'Kaur', '1997-06-20', 'simran.kaur@example.com', '9876543210');
```

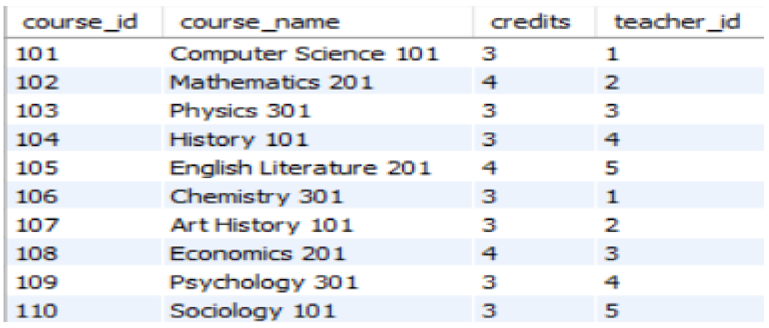


student_id	first_name	last_name	date_of_birth	email	phone_number
1	Aarav	Patel	1995-03-15	aarav.patel@example.com	9876543210
2	Ishita	Ishita	1997-08-22	ishita.sharma@example.com	8765432109
3	Rahul	Gupta	1998-05-10	rahul.gupta@example.com	7654321098
4	Neha	Singh	1996-11-30	neha.singh@example.com	6543210987
5	Kunal	Verma	1999-04-18	kunal.verma@example.com	5432109876
6	Anaya	Reddy	1997-07-02	anaya.reddy@example.com	4321098765
7	Arjun	Mishra	1994-12-25	arjun.mishra@example.com	3210987654
8	Sanya	Malik	1996-09-08	sanya.malik@example.com	2109876543
9	Rohan	Shah	1998-02-14	rohan.shah@example.com	1098765432
10	Simran	Kaur	1997-06-20	simran.kaur@example.com	9876543210

ii. Courses

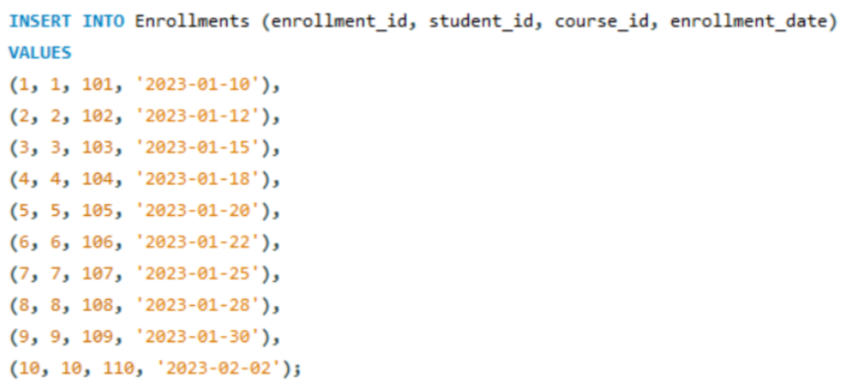


```
INSERT INTO Courses (course_id, course_name, credits, teacher_id)
VALUES
(101, 'Computer Science 101', 3, 1),
(102, 'Mathematics 201', 4, 2),
(103, 'Physics 301', 3, 3),
(104, 'History 101', 3, 4),
(105, 'English Literature 201', 4, 5),
(106, 'Chemistry 301', 3, 1),
(107, 'Art History 101', 3, 2),
(108, 'Economics 201', 4, 3),
(109, 'Psychology 301', 3, 4),
(110, 'Sociology 101', 3, 5);
```

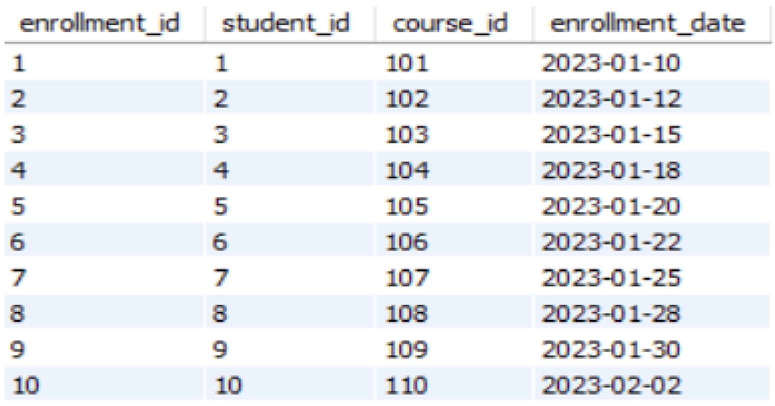


course_id	course_name	credits	teacher_id
101	Computer Science 101	3	1
102	Mathematics 201	4	2
103	Physics 301	3	3
104	History 101	3	4
105	English Literature 201	4	5
106	Chemistry 301	3	1
107	Art History 101	3	2
108	Economics 201	4	3
109	Psychology 301	3	4
110	Sociology 101	3	5

iii. Enrollments

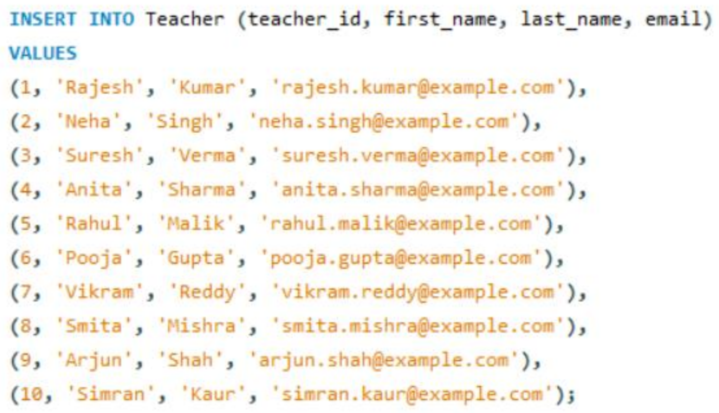


```sql
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
(1, 1, 101, '2023-01-10'),
(2, 2, 102, '2023-01-12'),
(3, 3, 103, '2023-01-15'),
(4, 4, 104, '2023-01-18'),
(5, 5, 105, '2023-01-20'),
(6, 6, 106, '2023-01-22'),
(7, 7, 107, '2023-01-25'),
(8, 8, 108, '2023-01-28'),
(9, 9, 109, '2023-01-30'),
(10, 10, 110, '2023-02-02');
```

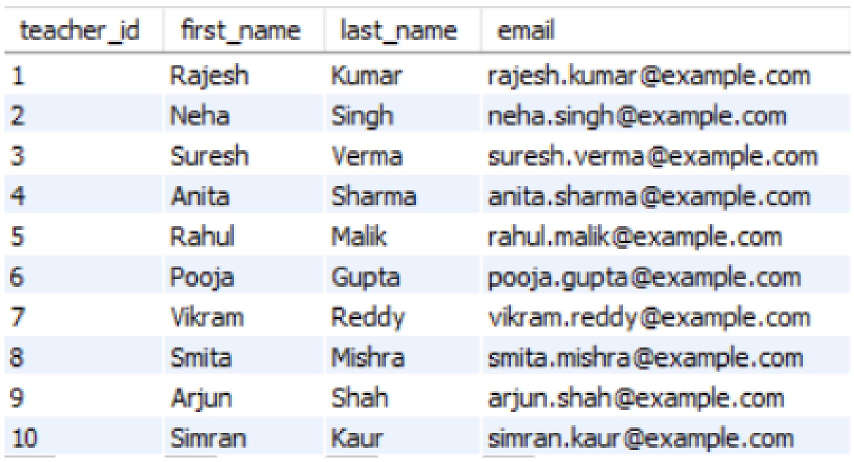


enrollment_id	student_id	course_id	enrollment_date
1	1	101	2023-01-10
2	2	102	2023-01-12
3	3	103	2023-01-15
4	4	104	2023-01-18
5	5	105	2023-01-20
6	6	106	2023-01-22
7	7	107	2023-01-25
8	8	108	2023-01-28
9	9	109	2023-01-30
10	10	110	2023-02-02

iv. Teacher

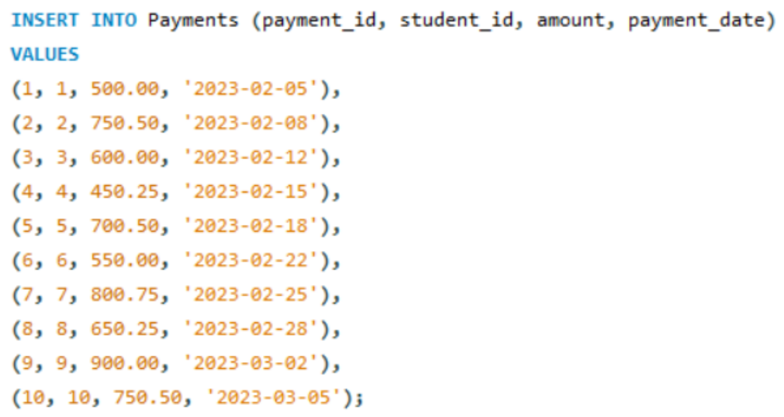


```sql
INSERT INTO Teacher (teacher_id, first_name, last_name, email)
VALUES
(1, 'Rajesh', 'Kumar', 'rajesh.kumar@example.com'),
(2, 'Neha', 'Singh', 'neha.singh@example.com'),
(3, 'Suresh', 'Verma', 'suresh.verma@example.com'),
(4, 'Anita', 'Sharma', 'anita.sharma@example.com'),
(5, 'Rahul', 'Malik', 'rahul.malik@example.com'),
(6, 'Pooja', 'Gupta', 'pooja.gupta@example.com'),
(7, 'Vikram', 'Reddy', 'vikram.reddy@example.com'),
(8, 'Smita', 'Mishra', 'smita.mishra@example.com'),
(9, 'Arjun', 'Shah', 'arjun.shah@example.com'),
(10, 'Simran', 'Kaur', 'simran.kaur@example.com');
```

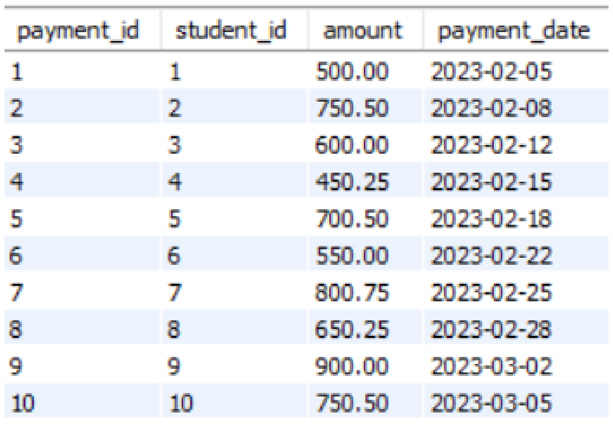


teacher_id	first_name	last_name	email
1	Rajesh	Kumar	rajesh.kumar@example.com
2	Neha	Singh	neha.singh@example.com
3	Suresh	Verma	suresh.verma@example.com
4	Anita	Sharma	anita.sharma@example.com
5	Rahul	Malik	rahul.malik@example.com
6	Pooja	Gupta	pooja.gupta@example.com
7	Vikram	Reddy	vikram.reddy@example.com
8	Smita	Mishra	smita.mishra@example.com
9	Arjun	Shah	arjun.shah@example.com
10	Simran	Kaur	simran.kaur@example.com

v. Payments



```
INSERT INTO Payments (payment_id, student_id, amount, payment_date)
VALUES
(1, 1, 500.00, '2023-02-05'),
(2, 2, 750.50, '2023-02-08'),
(3, 3, 600.00, '2023-02-12'),
(4, 4, 450.25, '2023-02-15'),
(5, 5, 700.50, '2023-02-18'),
(6, 6, 550.00, '2023-02-22'),
(7, 7, 800.75, '2023-02-25'),
(8, 8, 650.25, '2023-02-28'),
(9, 9, 900.00, '2023-03-02'),
(10, 10, 750.50, '2023-03-05');
```



payment_id	student_id	amount	payment_date
1	1	500.00	2023-02-05
2	2	750.50	2023-02-08
3	3	600.00	2023-02-12
4	4	450.25	2023-02-15
5	5	700.50	2023-02-18
6	6	550.00	2023-02-22
7	7	800.75	2023-02-25
8	8	650.25	2023-02-28
9	9	900.00	2023-03-02
10	10	750.50	2023-03-05

Task 2:

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

a. First Name: John

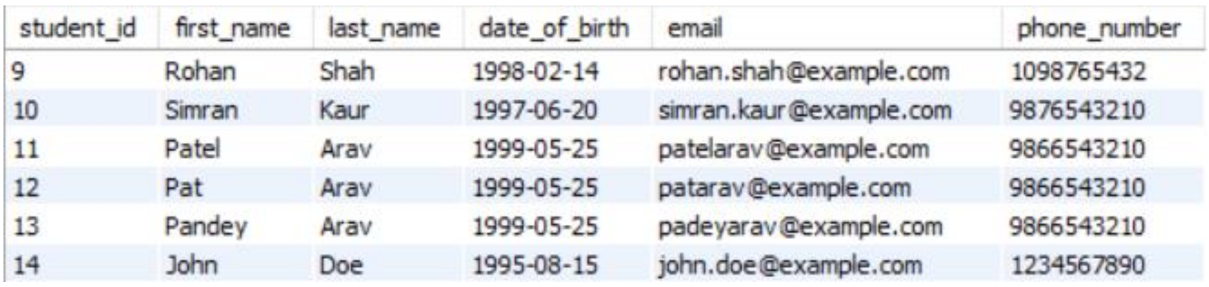
b. Last Name: Doe

c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

e. Phone Number: 1234567890

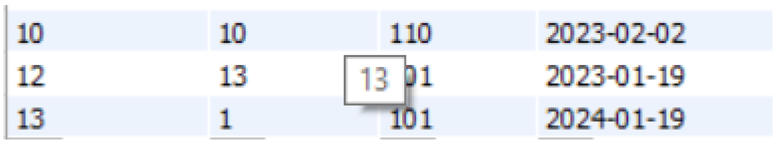
```
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
```



student_id	first_name	last_name	date_of_birth	email	phone_number
9	Rohan	Shah	1998-02-14	rohan.shah@example.com	1098765432
10	Simran	Kaur	1997-06-20	simran.kaur@example.com	9876543210
11	Patel	Arav	1999-05-25	patelarav@example.com	9866543210
12	Pat	Arav	1999-05-25	patarav@example.com	9866543210
13	Pandey	Arav	1999-05-25	padeyarav@example.com	9866543210
14	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.

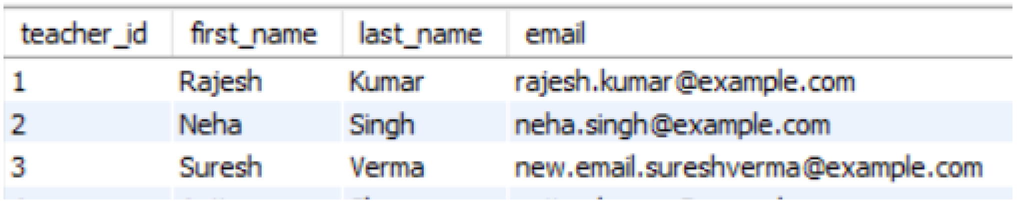
```
INSERT INTO Enrollments (student_id, course_id, enrollment_date)
VALUES (1, 101, CURRENT_DATE);
```



10	10	110	2023-02-02
12	13	13 01	2023-01-19
13	1	101	2024-01-19

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

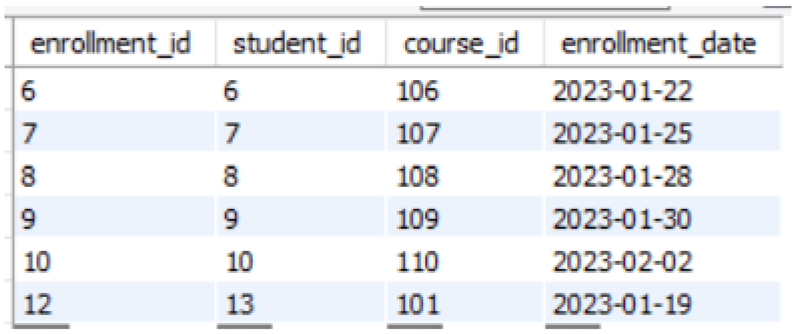
```
UPDATE Teacher
SET email = 'new.email.sureshverma@example.com'
WHERE teacher_id = 3;
```



teacher_id	first_name	last_name	email
1	Rajesh	Kumar	rajesh.kumar@example.com
2	Neha	Singh	neha.singh@example.com
3	Suresh	Verma	new.email.sureshverma@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.

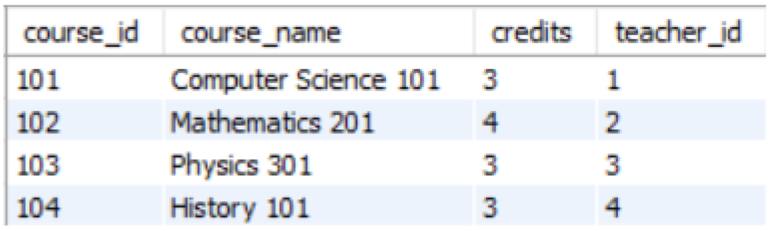
```
DELETE FROM Enrollments
WHERE student_id = 1 AND course_id = 101;
```



enrollment_id	student_id	course_id	enrollment_date
6	6	106	2023-01-22
7	7	107	2023-01-25
8	8	108	2023-01-28
9	9	109	2023-01-30
10	10	110	2023-02-02
12	13	101	2023-01-19

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

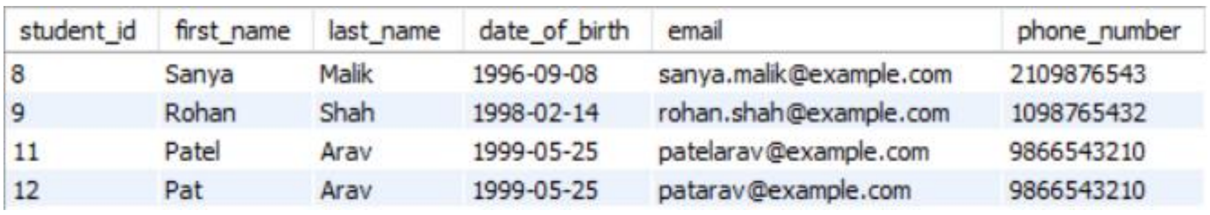
```
UPDATE Courses SET teacher_id = 2 WHERE course_id = 102;
```



course_id	course_name	credits	teacher_id
101	Computer Science 101	3	1
102	Mathematics 201	4	2
103	Physics 301	3	3
104	History 101	3	4

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.

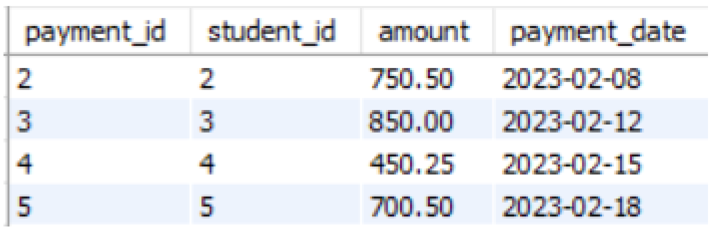
```
DELETE FROM Students WHERE student_id = 10;
```



student_id	first_name	last_name	date_of_birth	email	phone_number
8	Sanya	Malik	1996-09-08	sanya.malik@example.com	2109876543
9	Rohan	Shah	1998-02-14	rohan.shah@example.com	1098765432
11	Patel	Arav	1999-05-25	patelarav@example.com	9866543210
12	Pat	Arav	1999-05-25	patarav@example.com	9866543210

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

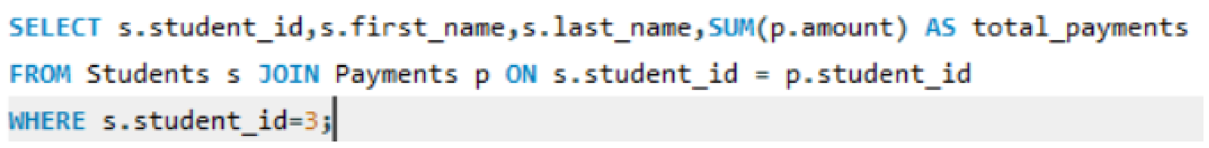
```
UPDATE Payments SET amount = 850.00 WHERE payment_id = 3;
```



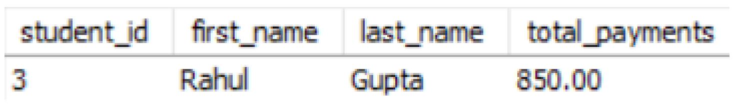
payment_id	student_id	amount	payment_date
2	2	750.50	2023-02-08
3	3	850.00	2023-02-12
4	4	450.25	2023-02-15
5	5	700.50	2023-02-18

Task 3:

1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

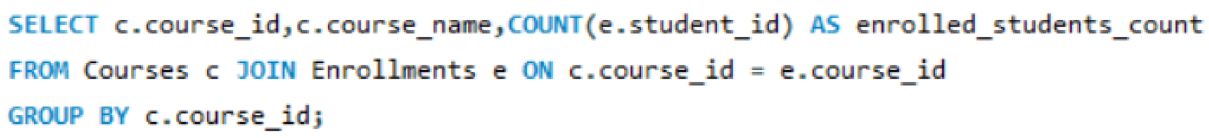


```sql
SELECT s.student_id,s.first_name,s.last_name,SUM(p.amount) AS total_payments
FROM Students s JOIN Payments p ON s.student_id = p.student_id
WHERE s.student_id=3;
```

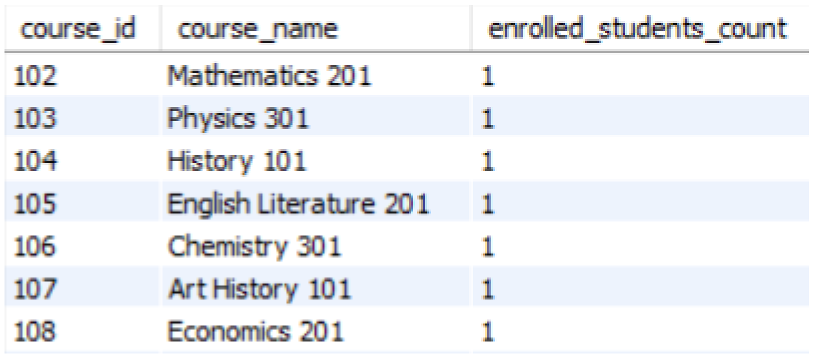


student_id	first_name	last_name	total_payments
3	Rahul	Gupta	850.00

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.

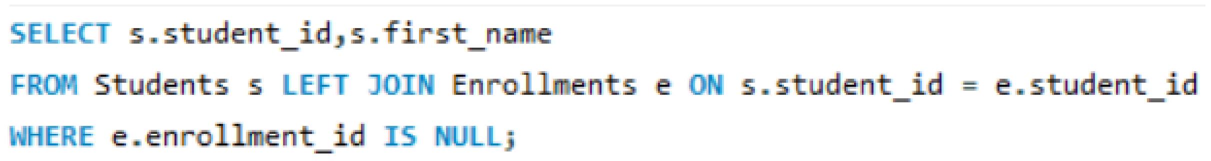


```sql
SELECT c.course_id,c.course_name,COUNT(e.student_id) AS enrolled_students_count
FROM Courses c JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id;
```

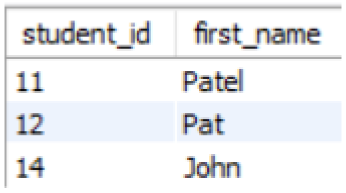


course_id	course_name	enrolled_students_count
102	Mathematics 201	1
103	Physics 301	1
104	History 101	1
105	English Literature 201	1
106	Chemistry 301	1
107	Art History 101	1
108	Economics 201	1

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.

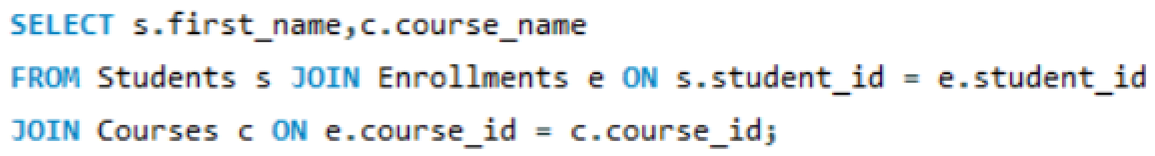


```sql
SELECT s.student_id,s.first_name
FROM Students s LEFT JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.enrollment_id IS NULL;
```

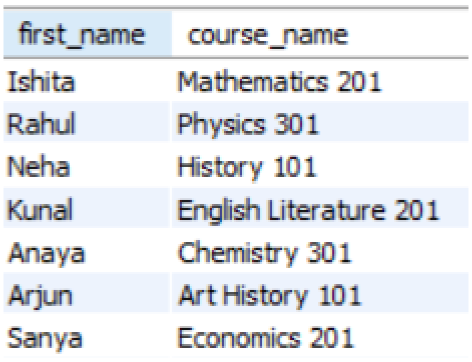


student_id	first_name
11	Patel
12	Pat
14	John

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.

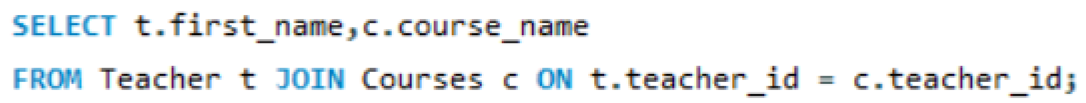


```sql
SELECT s.first_name,c.course_name
FROM Students s JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;
```

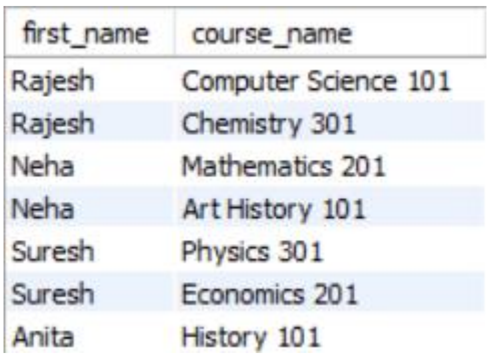


first_name	course_name
Ishita	Mathematics 201
Rahul	Physics 301
Neha	History 101
Kunal	English Literature 201
Anaya	Chemistry 301
Arjun	Art History 101
Sanya	Economics 201

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.

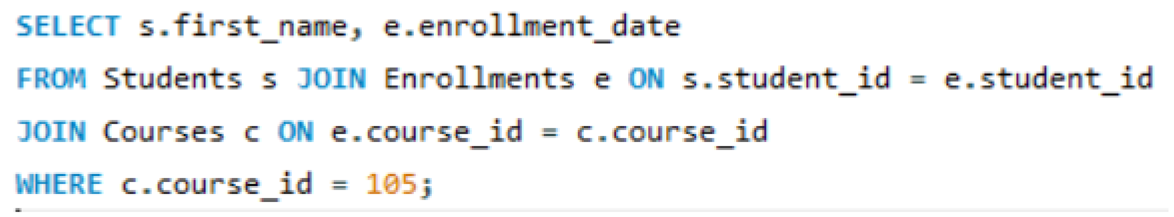


```
SELECT t.first_name,c.course_name
FROM Teacher t JOIN Courses c ON t.teacher_id = c.teacher_id;
```

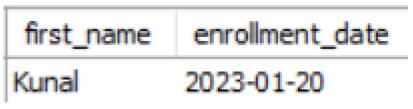


first_name	course_name
Rajesh	Computer Science 101
Rajesh	Chemistry 301
Neha	Mathematics 201
Neha	Art History 101
Suresh	Physics 301
Suresh	Economics 201
Anita	History 101

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.

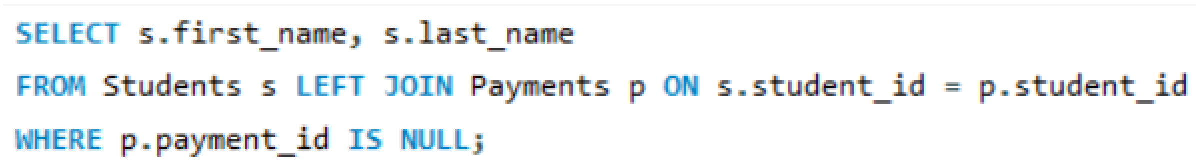


```
SELECT s.first_name, e.enrollment_date
FROM Students s JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE c.course_id = 105;
```

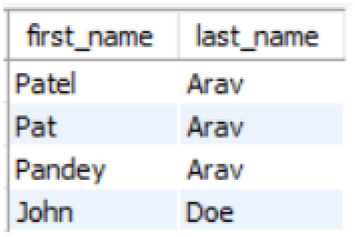


first_name	enrollment_date
Kunal	2023-01-20

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



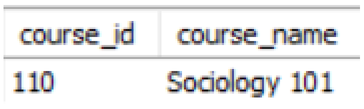
```
SELECT s.first_name, s.last_name
FROM Students s LEFT JOIN Payments p ON s.student_id = p.student_id
WHERE p.payment_id IS NULL;
```



first_name	last_name
Patel	Arav
Pat	Arav
Pandey	Arav
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.

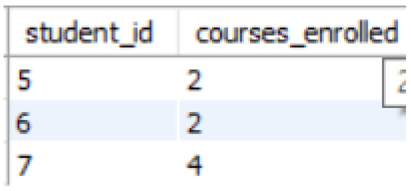
```
SELECT c.course_id,c.course_name
FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id
WHERE e.enrollment_id IS NULL;
```



course_id	course_name
110	Sociology 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.

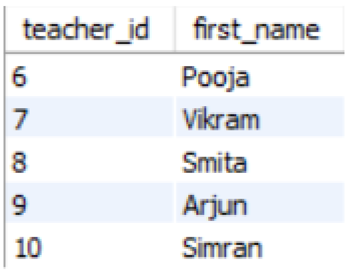
```
SELECT e1.student_id, COUNT(DISTINCT e1.course_id) AS courses_enrolled
FROM Enrollments e1 JOIN Enrollments e2 ON e1.student_id = e2.student_id
WHERE e1.course_id <> e2.course_id
GROUP BY e1.student_id;
```



student_id	courses_enrolled
5	2
6	2
7	4

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.

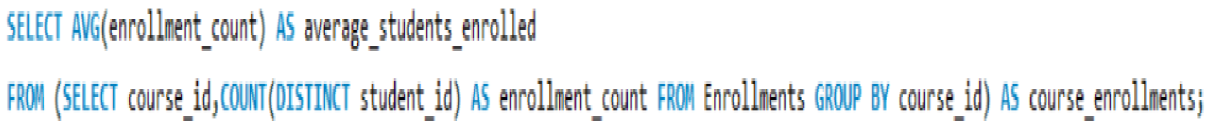
```
SELECT t.teacher_id,t.first_name
FROM Teacher t LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
WHERE c.course_id IS NULL;
```



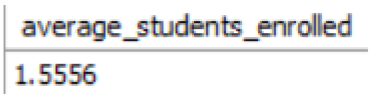
teacher_id	first_name
6	Pooja
7	Vikram
8	Smita
9	Arjun
10	Simran

Task 4:

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

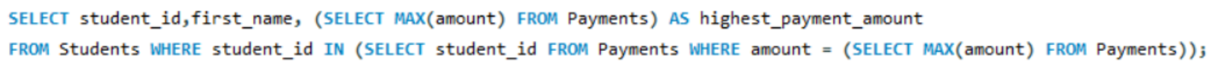


```
SELECT AVG(enrollment_count) AS average_students_enrolled

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

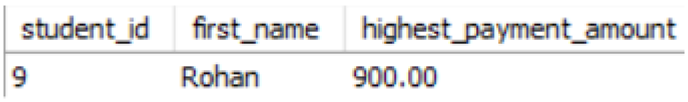


average_students_enrolled
1.5556

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.

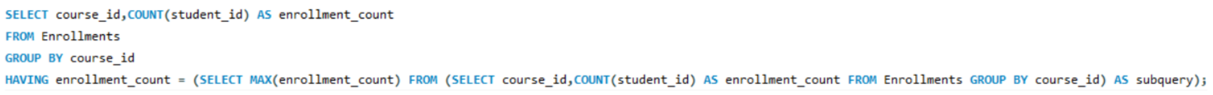


```
SELECT student_id,first_name, (SELECT MAX(amount) FROM Payments) AS highest_payment_amount
FROM Students WHERE student_id IN (SELECT student_id FROM Payments WHERE amount = (SELECT MAX(amount) FROM Payments));
```

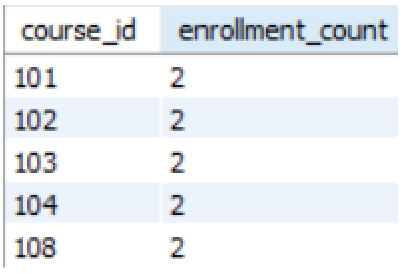


student_id	first_name	highest_payment_amount
9	Rohan	900.00

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

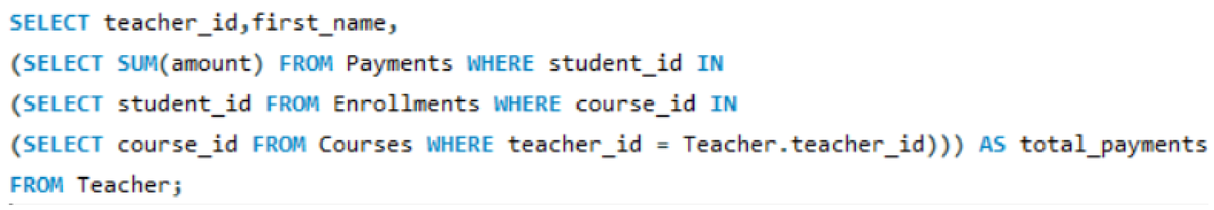


```
SELECT course_id,COUNT(student_id) AS enrollment_count
FROM Enrollments
GROUP BY course_id
HAVING enrollment_count = (SELECT MAX(enrollment_count) FROM (SELECT course_id,COUNT(student_id) AS enrollment_count FROM Enrollments GROUP BY course_id) AS subquery);
```

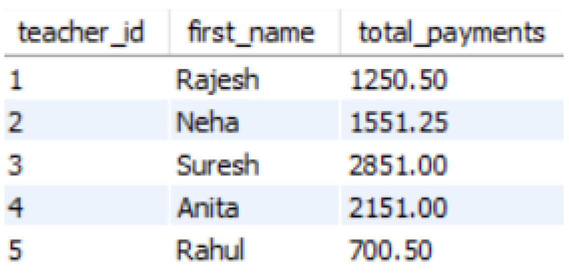


course_id	enrollment_count
101	2
102	2
103	2
104	2
108	2

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

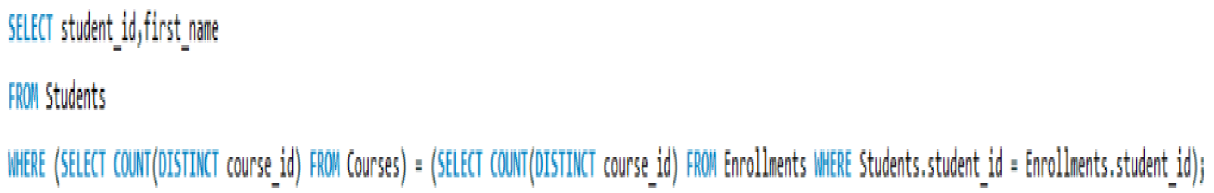


```
SELECT teacher_id,first_name,
(SELECT SUM(amount) FROM Payments WHERE student_id IN
(SELECT student_id FROM Enrollments WHERE course_id IN
(SELECT course_id FROM Courses WHERE teacher_id = Teacher.teacher_id))) AS total_payments
FROM Teacher;
```



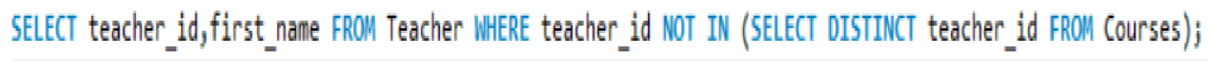
teacher_id	first_name	total_payments
1	Rajesh	1250.50
2	Neha	1551.25
3	Suresh	2851.00
4	Anita	2151.00
5	Rahul	700.50

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

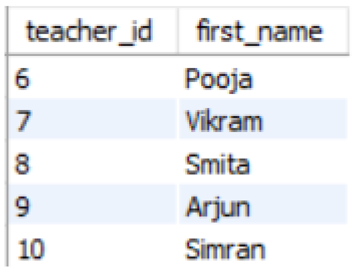


```
SELECT student_id,first_name

FROM Students

WHERE (SELECT COUNT(DISTINCT course_id) FROM Courses) = (SELECT COUNT(DISTINCT course_id) FROM Enrollments WHERE Students.student_id = Enrollments.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.

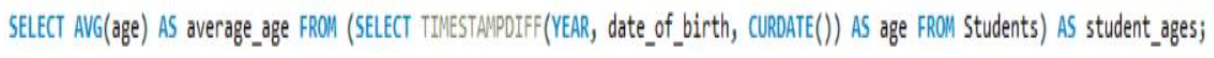


```
SELECT teacher_id,first_name FROM Teacher WHERE teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses);
```

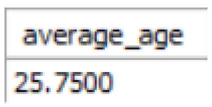


teacher_id	first_name
6	Pooja
7	Vikram
8	Smita
9	Arjun
10	Simran

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

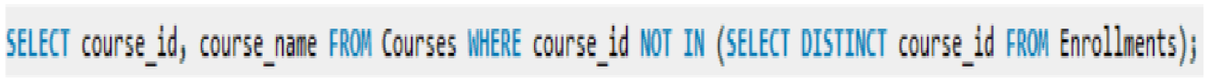


```
SELECT AVG(age) AS average_age FROM (SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age FROM Students) AS student_ages;
```

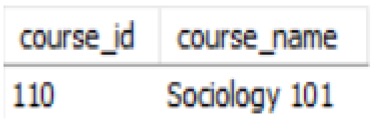


average_age
25.7500

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

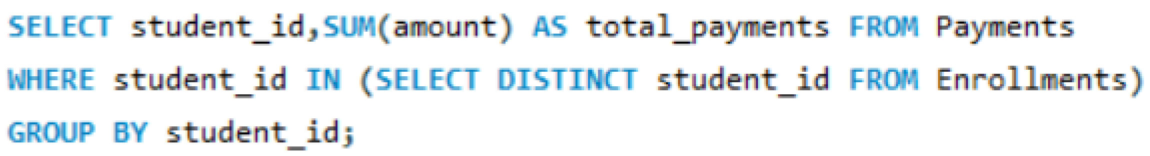


```sql
SELECT course_id, course_name FROM Courses WHERE course_id NOT IN (SELECT DISTINCT course_id FROM Enrollments);
```

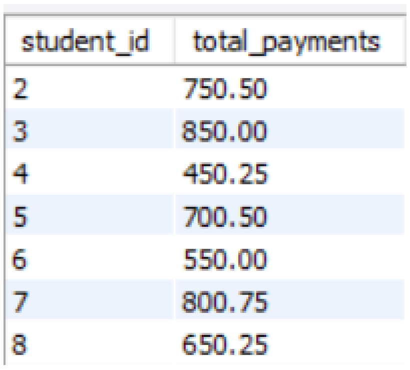


course_id	course_name
110	Sociology 101

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

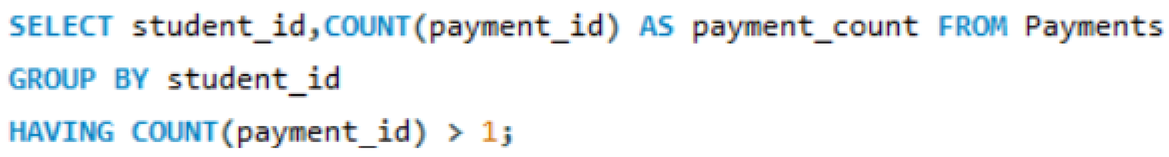


```sql
SELECT student_id,SUM(amount) AS total_payments FROM Payments
WHERE student_id IN (SELECT DISTINCT student_id FROM Enrollments)
GROUP BY student_id;
```

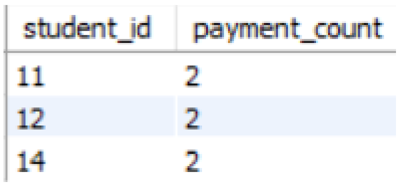


student_id	total_payments
2	750.50
3	850.00
4	450.25
5	700.50
6	550.00
7	800.75
8	650.25

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.

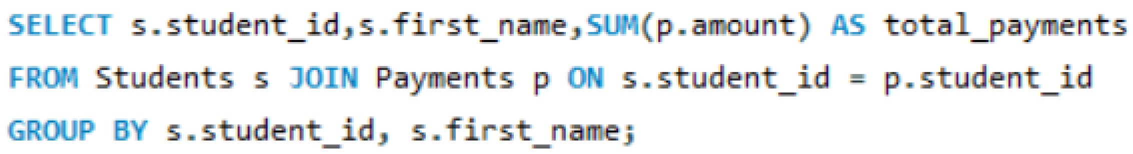


```sql
SELECT student_id,COUNT(payment_id) AS payment_count FROM Payments
GROUP BY student_id
HAVING COUNT(payment_id) > 1;
```

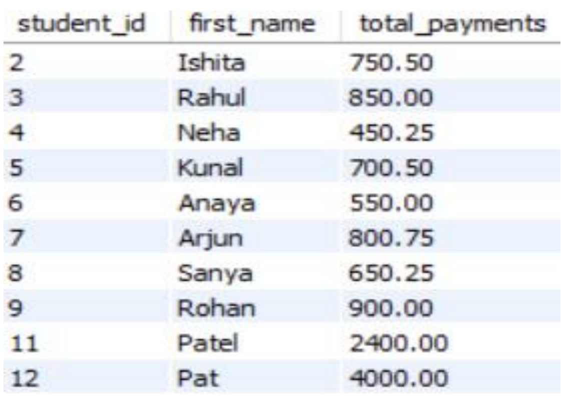


student_id	payment_count
11	2
12	2
14	2

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.

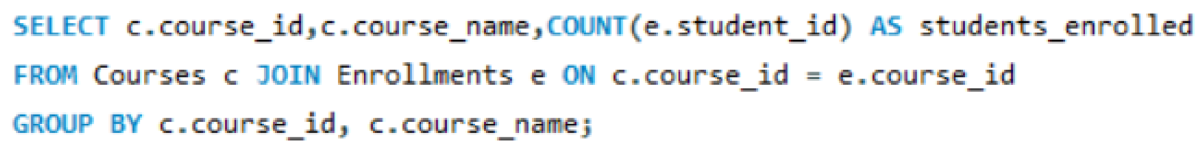


```sql
SELECT s.student_id,s.first_name,SUM(p.amount) AS total_payments
FROM Students s JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name;
```

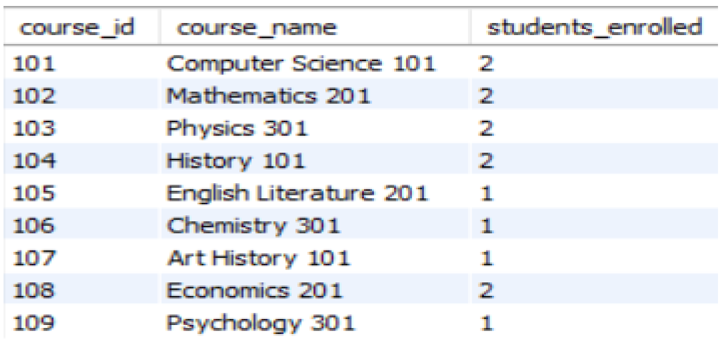


student_id	first_name	total_payments
2	Ishita	750.50
3	Rahul	850.00
4	Neha	450.25
5	Kunal	700.50
6	Anaya	550.00
7	Arjun	800.75
8	Sanya	650.25
9	Rohan	900.00
11	Patel	2400.00
12	Pat	4000.00

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.

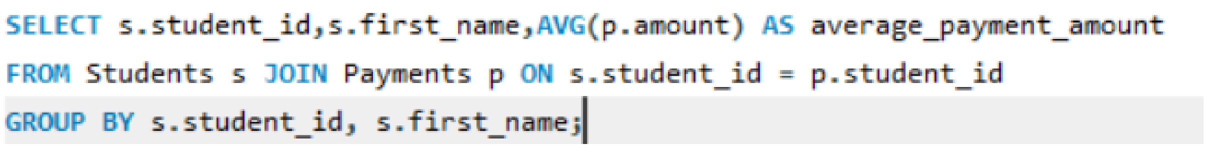


```
SELECT c.course_id,c.course_name,COUNT(e.student_id) AS students_enrolled
FROM Courses c JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name;
```

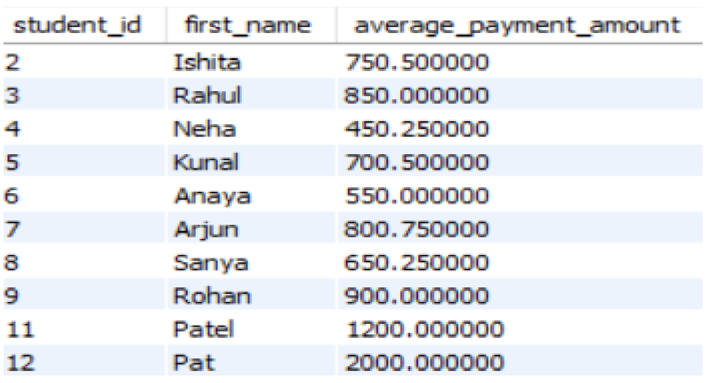


course_id	course_name	students_enrolled
101	Computer Science 101	2
102	Mathematics 201	2
103	Physics 301	2
104	History 101	2
105	English Literature 201	1
106	Chemistry 301	1
107	Art History 101	1
108	Economics 201	2
109	Psychology 301	1

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



student_id	first_name	average_payment_amount
2	Ishita	750.500000
3	Rahul	850.000000
4	Neha	450.250000
5	Kunal	700.500000
6	Anaya	550.000000
7	Arjun	800.750000
8	Sanya	650.250000
9	Rohan	900.000000
11	Patel	1200.000000
12	Pat	2000.000000