TASK 1

1. Create the database named "SISDB"

```
CREATE DATABASE IF NOT EXISTS SISDB;
SHOW DATABASES
```

Result Grid | Filter Rows: | Export: | W

Database
information_schema
mysql
performance_schema
sakila
sis
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

```
USE SISDB;

-- Students Table
CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    date_of_birth DATE,
    email VARCHAR(100),
    phone_number VARCHAR(15)
);

-- Teacher Table
CREATE TABLE Teachers (
    teacher_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100)
);
```

14	15:38:27	USE SISDB	0 row(s) affected	0.000 sec
15	15:38:27	CREATE TABLE Students (student_id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(...	0 row(s) affected	0.047 sec
16	15:40:07	CREATE TABLE Teachers (teacher_id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR...	0 row(s) affected	0.015 sec

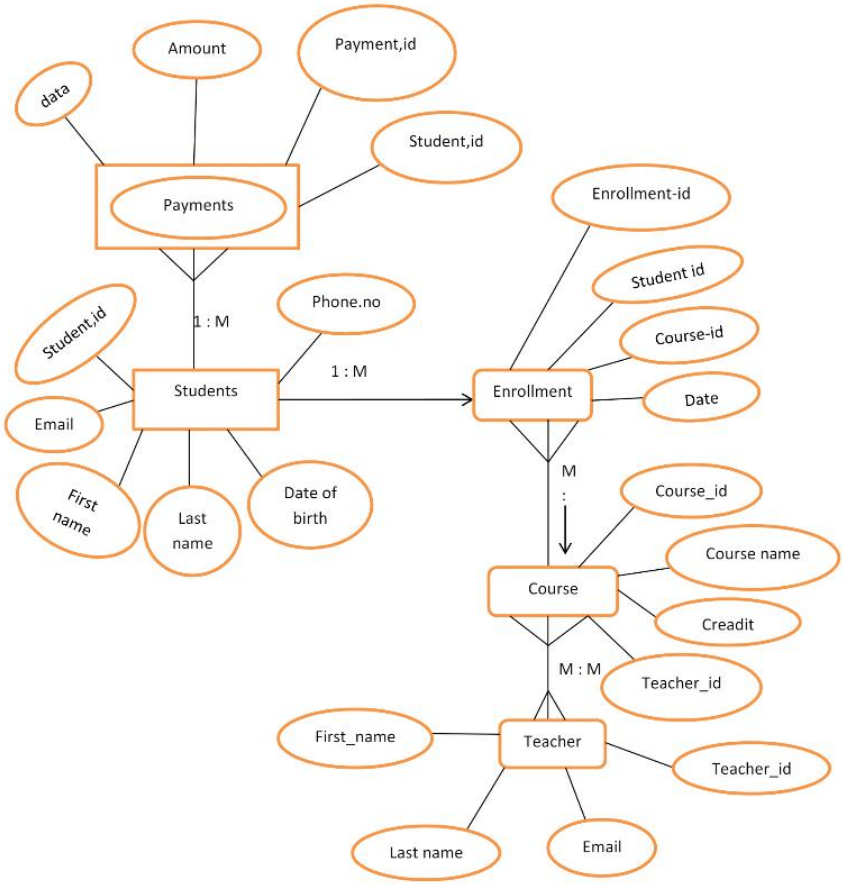
```sql
CREATE TABLE Courses (
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    course_name VARCHAR(100) NOT NULL,
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);

-- Enrollments Table
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    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)
);

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

17	15:40:07	CREATE TABLE Courses (course_id INT PRIMARY KEY AUTO_INCREMENT, course_name VARCHA...	0 row(s) affected	0.031 sec
18	15:40:07	CREATE TABLE Enrollments (enrollment_id INT PRIMARY KEY AUTO_INCREMENT, student_id INT, ...	0 row(s) affected	0.047 sec
19	15:40:07	CREATE TABLE Payments (payment_id INT PRIMARY KEY AUTO_INCREMENT, student_id INT, a...	0 row(s) affected	0.047 sec

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_sisdb
courses
enrollments
payments
students
teachers

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



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

i. Students

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

```sql
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES
    ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890'),
    ('Alice', 'Smith', '1998-04-22', 'alice.smith@example.com', '9876543210'),
    ('Amik', 'ghos', '1998-04-23', 'amik.gosh@example.com', '9876543211'),
    ('Asick', 'jay', '1998-04-24', 'asick.jay@example.com', '9871523210'),
    ('Peter', 'park', '1999-04-20', 'peter.park@example.com', '8976543210'),
    ('Anay', 'Astil', '1998-05-29', 'anay.astil@example.com', '9886543210'),
    ('Salman', 'Sheik', '1995-07-22', 'salman.sheik@example.com', '9876543409'),
    ('Ayush', 'Mohanty', '2001-04-24', 'ayush.mohanty@example.com', '9876544657'),
    ('Rujesh', 'rathior', '1900-09-22', 'r.r@example.com', '9876553210'),
    ('Michael', 'Smith', '2005-11-22', 'mike.smith@example.com', '9867854321');


SELECT * FROM Students;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	first_name	last_name	date_of_birth	email	phone_number
1	John	Doe	1995-08-15	john.doe@example.com	1234567890
2	Alice	Smith	1998-04-22	alice.smith@example.com	9876543210
3	Amik	ghos	1998-04-23	amik.gosh@example.com	9876543211
4	Asick	jay	1998-04-24	asick.jay@example.com	9871523210
5	Peter	park	1999-04-20	peter.park@example.com	8976543210
6	Anay	Astil	1998-05-29	anay.astil@example.com	9886543210
7	Salman	Sheik	1995-07-22	salman.sheik@example.com	9876543409
8	Ayush	Mohanty	2001-04-24	ayush.mohanty@example.com	9876544657
9	Rujesh	rathior	1900-09-22	r.r@example.com	9876553210
10	Michael	Smith	2005-11-22	mike.smith@example.com	9867854321
NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types

Limit to 1000 rows

```sql
INSERT INTO Teachers (first_name, last_name, email)
VALUES
    ('Michael', 'Johnson', 'michael.johnson@example.com'),
    ('Emily', 'Smith', 'emily.smith@example.com'),
    ('Robert', 'Williams', 'robert.williams@example.com'),
    ('Jessica', 'Davis', 'jessica.davis@example.com'),
    ('Daniel', 'Brown', 'daniel.brown@example.com'),
    ('Megan', 'Jones', 'megan.jones@example.com'),
    ('Christopher', 'Taylor', 'christopher.taylor@example.com'),
    ('Sophia', 'Anderson', 'sophia.anderson@example.com'),
    ('Andrew', 'Thomas', 'andrew.thomas@example.com'),
    ('Olivia', 'Martinez', 'olivia.martinez@example.com');


SELECT * FROM Teachers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

teacher_id	first_name	last_name	email
1	Michael	Johnson	michael.johnson@example.com
2	Emily	Smith	emily.smith@example.com
3	Robert	Williams	robert.williams@example.com
4	Jessica	Davis	jessica.davis@example.com
5	Daniel	Brown	daniel.brown@example.com
6	Megan	Jones	megan.jones@example.com
7	Christopher	Taylor	christopher.taylor@example.com
8	Sophia	Anderson	sophia.anderson@example.com
9	Andrew	Thomas	andrew.thomas@example.com
10	Olivia	Martinez	olivia.martinez@example.com
NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types

```sql
INSERT INTO Courses (course_name, credits, teacher_id)
VALUES
    ('Mathematics 101', 4, 1),
    ('Computer Science Fundamentals', 3, 2),
    ('History of Art', 3, 3),
    ('English Composition', 4, 4),
    ('Physics Basics', 4, 5),
    ('Introduction to Psychology', 3, 6),
    ('Web Development Advanced', 4, 7),
    ('Chemistry Principles', 4, 8),
    ('Spanish Language', 3, 9),
    ('Music Theory', 3, 10);

SELECT * FROM Courses;
```

Result Grid

course_id	course_name	credits	teacher_id
1	Mathematics 101	4	1
2	Computer Science Fundamentals	3	2
3	History of Art	3	3
4	English Composition	4	4
5	Physics Basics	4	5
6	Introduction to Psychology	3	6
7	Web Development Advanced	4	7
8	Chemistry Principles	4	8
9	Spanish Language	3	9
10	Music Theory	3	10
NULL	NULL	NULL	NULL

Courses 7

```sql
INSERT INTO Enrollments (student_id, course_id, enrollment_date)
VALUES
    (1, 1, '2022-01-15'),
    (2, 1, '2022-01-16'),
    (3, 2, '2022-01-17'),
    (4, 3, '2022-01-18'),
    (5, 4, '2022-01-19'),
    (6, 5, '2022-01-20'),
    (7, 6, '2022-01-21'),
    (8, 7, '2022-01-22'),
    (9, 8, '2022-01-23'),
    (10, 9, '2022-01-24');


SELECT * FROM Enrollments;
```

Result Grid

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2022-01-15
2	2	1	2022-01-16
3	3	2	2022-01-17
4	4	3	2022-01-18
5	5	4	2022-01-19
6	6	5	2022-01-20
7	7	6	2022-01-21
8	8	7	2022-01-22
9	9	8	2022-01-23
10	10	9	2022-01-24
NULL	NULL	NULL	NULL

Enrollments 8

```sql
INSERT INTO Payments (student_id, amount, payment_date)
VALUES
    (1, 500, '2022-01-20'),
    (2, 600, '2022-01-22'),
    (3, 450, '2022-01-24'),
    (4, 550, '2022-01-26'),
    (5, 700, '2022-01-28'),
    (6, 400, '2022-01-30'),
    (7, 750, '2022-02-01'),
    (8, 480, '2022-02-03'),
    (9, 600, '2022-02-05'),
    (10, 520, '2022-02-07');

SELECT * FROM Payments;
```

Result Grid

payment_id	student_id	amount	payment_date
1	1	500.00	2022-01-20
2	2	600.00	2022-01-22
3	3	450.00	2022-01-24
4	4	550.00	2022-01-26
5	5	700.00	2022-01-28
6	6	400.00	2022-01-30
7	7	750.00	2022-02-01
8	8	480.00	2022-02-03
9	9	600.00	2022-02-05
10	10	520.00	2022-02-07
NULL	NULL	NULL	NULL

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

```sql
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES ('Armir', 'Dele', '1993-08-15', 'armir.dele@example.com', '1234567890');


Select * from Students
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	first_name	last_name	date_of_birth	email	phone_number
1	John	Doe	1995-08-15	john.doe@example.com	1234567890
2	Alice	Smith	1998-04-22	alice.smith@example.com	9876543210
3	Amik	ghos	1998-04-23	amik.gosh@example.com	9876543211
4	Asick	jay	1998-04-24	asick.jay@example.com	9871523210
5	Peter	park	1999-04-20	peter.park@example.com	8976543210
6	Anay	Astil	1998-05-29	anay.astil@example.com	9886543210
7	Salman	Sheik	1995-07-22	salman.sheik@example.com	9876543409
8	Ayush	Mohanty	2001-04-24	ayush.mohanty@example.com	9876544657
9	Rujesh	rathior	1900-09-22	r.r@example.com	9876553210
10	Michael	Smith	2005-11-22	mike.smith@example.com	9867854321
11	Armir	Dele	1993-08-15	armir.dele@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.

```sql
INSERT INTO Enrollments (student_id, course_id, enrollment_date)
VALUES ('1', '2', '2022-01-10');


Select * from Enrollments
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2022-01-15
2	2	1	2022-01-16
3	3	2	2022-01-17
4	4	3	2022-01-18
5	5	4	2022-01-19
6	6	5	2022-01-20
7	7	6	2022-01-21
8	8	7	2022-01-22
9	9	8	2022-01-23
10	10	9	2022-01-24
11	1	2	2022-01-10
NULL	NULL	NULL	NULL

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

```sql
UPDATE Teachers
SET email = 'olivia.mart@gmail.com'
WHERE teacher_id = 10;

Select * from Teachers
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

teacher_id	first_name	last_name	email
1	Michael	Johnson	michael.johnson@example.com
2	Emily	Smith	emily.smith@example.com
3	Robert	Williams	robert.williams@example.com
4	Jessica	Davis	jessica.davis@example.com
5	Daniel	Brown	daniel.brown@example.com
6	Megan	Jones	megan.jones@example.com
7	Christopher	Taylor	christopher.taylor@example.com
8	Sophia	Anderson	sophia.anderson@example.com
9	Andrew	Thomas	andrew.thomas@example.com
10	Olivia	Martinez	olivia.mart@gmail.com
NULL	NULL	NULL	NULL

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.

Limit to 1000 rows

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


Select * from enrollments
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2022-01-15
2	2	1	2022-01-16
3	3	2	2022-01-17
4	4	3	2022-01-18
5	5	4	2022-01-19
6	6	5	2022-01-20
7	7	6	2022-01-21
8	8	7	2022-01-22
9	9	8	2022-01-23
10	10	9	2022-01-24
NULL	NULL	NULL	NULL

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 = '4'
WHERE course_id = '10';

Select * from courses
```

course_id	course_name	credits	teacher_id
1	Mathematics 101	4	1
2	Computer Science Fundamentals	3	2
3	History of Art	3	3
4	English Composition	4	4
5	Physics Basics	4	5
6	Introduction to Psychology	3	6
7	Web Development Advanced	4	7
8	Chemistry Principles	4	8
9	Spanish Language	3	9
10	Music Theory	3	4
NULL	NULL	NULL	NULL

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 = '11';


Select * from students
```

student_id	first_name	last_name	date_of_birth	email	phone_number
1	John	Doe	1995-08-15	john.doe@example.com	1234567890
2	Alice	Smith	1998-04-22	alice.smith@example.com	9876543210
3	Amik	ghos	1998-04-23	amik.gosh@example.com	9876543211
4	Asick	jay	1998-04-24	asick.jay@example.com	9871523210
5	Peter	park	1999-04-20	peter.park@example.com	8976543210
6	Anay	Astil	1998-05-29	anay.astil@example.com	9886543210
7	Salman	Sheik	1995-07-22	salman.sheik@example.com	9876543409
8	Ayush	Mohanty	2001-04-24	ayush.mohanty@example.com	9876544657
9	Rujesh	rathior	1900-09-22	r.r@example.com	9876553210
10	Michael	Smith	2005-11-22	mike.smith@example.com	9867854321
NULL	NULL	NULL	NULL	NULL	NULL

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

```sql
UPDATE Payments
SET amount = '820'
WHERE payment_id = '10';

Select * from payments
```

payment_id	student_id	amount	payment_date
1	1	500.00	2022-01-20
2	2	600.00	2022-01-22
3	3	450.00	2022-01-24
4	4	550.00	2022-01-26
5	5	700.00	2022-01-28
6	6	400.00	2022-01-30
7	7	750.00	2022-02-01
8	8	480.00	2022-02-03
9	9	600.00	2022-02-05
10	10	820.00	2022-02-07
NULL	NULL	NULL	NULL

2022-01-22

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.

```sql
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
WHERE s.student_id = 2
GROUP BY s.first_name, s.last_name;
```

first_name	last_name	total_payments
Alice	Smith	600.00

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

course_id	course_name	enrolled_students
1	Mathematics 101	2
2	Computer Science Fundamentals	1
3	History of Art	1
4	English Composition	1
5	Physics Basics	1
6	Introduction to Psychology	1
7	Web Development Advanced	1
8	Chemistry Principles	1
9	Spanish Language	1
10	Music Theory	0

3. Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students

without enrollments.

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

first_name	last_name

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, 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
John	Doe	Mathematics 101
Alice	Smith	Mathematics 101
Amik	ghos	Computer Science Fundamentals
Asick	jay	History of Art
Peter	park	English Composition
Anay	Astil	Physics Basics
Salman	Sheik	Introduction to Psychology
Ayush	Mohanty	Web Development Advanced
Rujesh	rathior	Chemistry Principles
Michael	Smith	Spanish Language

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.

```sql
SELECT t.first_name AS teachers_first_name, t.last_name AS teachers_last_name, c.course_name
FROM Teachers t
JOIN Courses c ON t.teacher_id = c.teacher_id;
```

teachers_first_name	teachers_last_name	course_name
Michael	Johnson	Mathematics 101
Emily	Smith	Computer Science Fundamentals
Robert	Williams	History of Art
Jessica	Davis	English Composition
Jessica	Davis	Music Theory
Daniel	Brown	Physics Basics
Megan	Jones	Introduction to Psychology
Christopher	Taylor	Web Development Advanced
Sophia	Anderson	Chemistry Principles
Andrew	Thomas	Spanish Language

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.

```sql
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 = 2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	enrollment_date
Amik	ghos	2022-01-17

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.

```sql
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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name

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.

```sql
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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_id	course_name
10	Music Theory

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.

```sql
SELECT s.student_id, s.first_name, s.last_name, COUNT(e.enrollment_id) AS num_enrollments
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(e.enrollment_id) > 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	num_enrollments

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.

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

teacher_id	first_name	last_name
10	Olivia	Martinez

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.

```sql
SELECT course_id, AVG(num_students) AS average_students
FROM (
 SELECT course_id, COUNT(DISTINCT student_id) AS num_students
 FROM Enrollments
 GROUP BY course_id
) AS CourseEnrollments
GROUP BY course id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_id	average_students
1	2.0000
2	1.0000
3	1.0000
4	1.0000
5	1.0000
6	1.0000
7	1.0000
8	1.0000
9	1.0000

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.

```sql
SELECT student_id, amount, payment_date
FROM Payments
WHERE amount = (SELECT MAX(amount) FROM Payments);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	amount	payment_date
10	820.00	2022-02-07

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

```sql
SELECT c.course_id, c.course_name, COUNT(e.enrollment_id) AS enrollment_count
FROM Courses c
JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name
HAVING COUNT(e.enrollment_id) = (
 SELECT MAX(enrollment_count)
 FROM (
 SELECT COUNT(enrollment_id) AS enrollment_count
 FROM Enrollments
 GROUP BY course_id
 ) AS max_enrollments);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_id	course_name	enrollment_count
1	Mathematics 101	2

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

```sql
SELECT t.teacher_id, t.first_name, t.last_name, SUM(p.amount) AS total_payments
FROM Teachers 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, t.first_name, t.last_name;
```

teacher_id	first_name	last_name	total_payments
1	Michael	Johnson	1100.00
2	Emily	Smith	450.00
3	Robert	Williams	550.00
4	Jessica	Davis	700.00
5	Daniel	Brown	400.00
6	Megan	Jones	750.00
7	Christopher	Taylor	480.00
8	Sophia	Anderson	600.00
9	Andrew	Thomas	820.00

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

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

student_id	first_name	last_name
NULL	NULL	NULL

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

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

teacher_id	first_name	last_name
10	Olivia	Martinez
NULL	NULL	NULL

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

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

average_age
34.3000

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
10	Music Theory
NULL	NULL

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
 s.student_id,
 s.first_name,
 s.last_name,
 c.course_id,
 c.course_name,
 COALESCE(SUM(p.amount), 0) AS total_payments
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
LEFT JOIN Payments p ON s.student_id = p.student_id
GROUP BY
 s.student_id,
 s.first_name,
 s.last_name,
 c.course_id,
 c.course_name;
```

student_id	first_name	last_name	course_id	course_name	total_payments
1	John	Doe	1	Mathematics 101	500.00
2	Alice	Smith	1	Mathematics 101	600.00
3	Amik	ghos	2	Computer Science Fundamentals	450.00
4	Asick	jay	3	History of Art	550.00
5	Peter	park	4	English Composition	700.00
6	Anay	Astil	5	Physics Basics	400.00
7	Salman	Sheik	6	Introduction to Psychology	750.00
8	Ayush	Mohanty	7	Web Development Advanced	480.00
9	Rujesh	rathior	8	Chemistry Principles	600.00
10	Michael	Smith	9	Spanish Language	820.00

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.

```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
);
```

student_id	first_name	last_name
NULL	NULL	NULL

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,
 s.last_name,
 COALESCE(SUM(p.amount), 0) 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;
```

student_id	first_name	last_name	total_payments
1	John	Doe	500.00
2	Alice	Smith	600.00
3	Amik	ghos	450.00
4	Asick	jay	550.00
5	Peter	park	700.00
6	Anay	Astil	400.00
7	Salman	Sheik	750.00
8	Ayush	Mohanty	480.00
9	Rujesh	rathior	600.00
10	Michael	Smith	820.00

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.

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

course_id	course_name	enrolled_students
1	Mathematics 101	2
2	Computer Science Fundamentals	1
3	History of Art	1
4	English Composition	1
5	Physics Basics	1
6	Introduction to Psychology	1
7	Web Development Advanced	1
8	Chemistry Principles	1
9	Spanish Language	1
10	Music Theory	0

13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

```sql
SELECT
 s.student_id,
 s.first_name,
 s.last_name,
 COALESCE(AVG(p.amount), 0) AS average_payment_amount
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;
```

student_id	first_name	last_name	average_payment_amount
1	John	Doe	500.000000
2	Alice	Smith	600.000000
3	Amik	ghos	450.000000
4	Asick	jay	550.000000
5	Peter	park	700.000000
6	Anay	Astil	400.000000
7	Salman	Sheik	750.000000
8	Ayush	Mohanty	480.000000
9	Rujesh	rathior	600.000000
10	Michael	Smith	820.000000