

Task 1. Database Design:

1. Create the database named "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
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```
1 • CREATE DATABASE SISDB;
2 • USE SISDB;
3 • CREATE TABLE Students (
4     student_id INT PRIMARY KEY AUTO_INCREMENT,
5     first_name VARCHAR(255),
6     last_name VARCHAR(255),
7     date_of_birth DATE,
8     email VARCHAR(255),
9     phone_number VARCHAR(20)
10 );
11
12 • CREATE TABLE Teacher (
13     teacher_id INT PRIMARY KEY AUTO_INCREMENT,
14     first_name VARCHAR(255),
15     last_name VARCHAR(255),
16     email VARCHAR(255)
17 );
18
19 • CREATE TABLE Courses (
20     course_id INT PRIMARY KEY AUTO_INCREMENT,
21     course_name VARCHAR(255),
22     credits INT,
23     teacher_id INT,
24     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
25 );
26 • CREATE TABLE Enrollments (
27     enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
28     student_id INT NOT NULL,
29     course_id INT NOT NULL,
30     enrollment_date DATE NOT NULL,
31     FOREIGN KEY (student_id) REFERENCES Students(student_id),
32     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
33 );
```

```

35 • CREATE TABLE Payments (
36     payment_id INT PRIMARY KEY AUTO_INCREMENT,
37     student_id INT NOT NULL,
38     amount DECIMAL(10, 2),
39     payment_date DATE,
40     FOREIGN KEY (student_id) REFERENCES Students(student_id)
41 );

```

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

i. Students

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

```

43 • INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
44 VALUES
45 (1, 'Krishna', 'Patle', '2001-08-12', 'krishnapatle@gmail.com', '9325654953'),
46 (2, 'Kashyap', 'Punyawar', '2001-03-04', 'kashyappunyawar@gmail.com', '9325655453'),
47 (3, 'Harshal', 'Meshram', '2002-05-27', 'harshalmeshram@gmail.com', '9125654953'),
48 (4, 'Nitin', 'Turkar', '2000-12-01', 'nitinturkar@gmail.com', '8698454798'),
49 (5, 'Vikas', 'Nagpure', '2001-04-23', 'vikasnagpure@gmail.com', '9124524953'),
50 (6, 'Shivam', 'Kale', '2001-03-16', 'shivamkale@gmail.com', '7825654953'),
51 (7, 'Ruchika', 'Chafekar', '2001-06-30', 'ruchikachafekar@gmail.com', '7447654906'),
52 (8, 'Neha', 'Patle', '2003-09-03', 'nehapatle@gmail.com', '7825654901'),
53 (9, 'Pratiksha', 'Katre', '2000-05-12', 'pratikshakatre@gmail.com', '9125654953'),
54 (10, 'Shruti', 'Kolhe', '2001-03-06', 'shrutikolhe@gmail.com', '6725654900');
55 • SELECT * FROM Students;

```

Result Grid						
Filter Rows:						
Edit: Export/Import: Wrap Cell Content: I A						
	student_id	first_name	last_name	date_of_birth	email	phone_number
▶	1	Krishna	Patle	2001-08-12	krishnapatle@gmail.com	9325654953
	2	Kashyap	Punyawar	2001-03-04	kashyappunyawar@gmail.com	9325655453
	3	Harshal	Meshram	2002-05-27	harshalmeshram@gmail.com	9125654953
	4	Nitin	Turkar	2000-12-01	nitinturkar@gmail.com	8698454798
	5	Vikas	Nagpure	2001-04-23	vikasnagpure@gmail.com	9124524953
	6	Shivam	Kale	2001-03-16	shivamkale@gmail.com	7825654953
	7	Ruchika	Chafekar	2001-06-30	ruchikachafekar@gmail.com	7447654906
	8	Neha	Patle	2003-09-03	nehapatle@gmail.com	7825654901
	9	Pratiksha	Katre	2000-05-12	pratikshakatre@gmail.com	9125654953
	10	Shruti	Kolhe	2001-03-06	shrutikolhe@gmail.com	6725654900
•	NULL	NULL	NULL	NULL	NULL	NULL

```

57 • INSERT INTO Teacher (first_name, last_name, email) VALUES
58   ("Mr. Smith", "Taylor", "smith@email.com"),
59   ("Ms. Jones", "Swift", "jones@email.com"),
60   ("Mr. Kailas", "Shekhar", "kailash@gmail.com"),
61   ("Mr. Pruthvi", "Chaudhary", "pruthvi@gmail.com"),
62   ("Mr. Rajat", "Patidar", "rajat@gmail.com"),
63   ("Ms. Yogita", "Lanjewar", "yogita@gmail.com"),
64   ("Ms. Ragini", "Yadav", "ragini@gmail.com"),
65   ("Mr. Saurabh", "Gedekar", "saurabh@gmail.com"),
66   ("Ms. Neha", "Patle", "nehapatle@gmail.com"),
67   ("Mr. Harshal", "Meshram", "harshal@gmail.com");
68
69 • SELECT * FROM Teacher;

```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:				
Export/Import:				
Wrap Cell Content:				
	teacher_id	first_name	last_name	email
▶	1	Mr. Smith	Taylor	smith@email.com
	2	Ms. Jones	Swift	jones@email.com
	3	Mr. Kailas	Shekhar	kailash@gmail.com
	4	Mr. Pruthvi	Chaudhary	pruthvi@gmail.com
	5	Mr. Rajat	Patidar	rajat@gmail.com
	6	Ms. Yogita	Lanjewar	yogita@gmail.com
	7	Ms. Ragini	Yadav	ragini@gmail.com
	8	Mr. Saurabh	Gedekar	saurabh@gmail.com
	9	Ms. Neha	Patle	nehapatle@gmail.com
	10	Mr. Harshal	Meshram	harshal@gmail.com
•	NULL	NULL	NULL	NULL

```

71 • INSERT INTO Courses (course_name, credits, teacher_id) VALUES
72   ("Mathematics", 3, 1),
73   ("English", 3, 2),
74   ("History", 2, 3),
75   ("Geography", 3, 4),
76   ("C++", 4, 5),
77   ("Python", 4, 6),
78   ("Economy", 2, 7),
79   ("Science", 4, 8),
80   ("Social Science", 2, 9),
81   ("Java", 4, 10);
82
83 • SELECT * FROM Courses;

```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:				
Export/Import:				
Wrap Cell Content:				
	course_id	course_name	credits	teacher_id
▶	1	Mathematics	3	1
	2	English	3	2
	3	History	2	3
	4	Geography	3	4
	5	C++	4	5
	6	Python	4	6
	7	Economy	2	7
	8	Science	4	8
	9	Social Science	2	9
	10	Java	4	10

```
85 • INSERT INTO Enrollments (student_id, course_id, enrollment_date) VALUES
86     (1, 1, "2023-10-01"),
87     (2, 2, "2023-10-02"),
88     (3,5,"2022-3-4"),
89     (4,3,"2019-8-12"),
90     (5,4,"2023-4-23"),
91     (6,7,"2022-09-8"),
92     (7,8,"2022-7-1"),
93     (8,9,"2023-10-10"),
94     (9,10,"2022-09-5"),
95     (10,6,"2022-12-30");
96
97 • SELECT * FROM Enrollments;
```

Result Grid				
Filter Rows:		Edit:		
Export/Import:		Wrap Cell Content:		
	enrollment_id	student_id	course_id	enrollment_date
▶	1	1	1	2023-10-01
	2	2	2	2023-10-02
	3	3	5	2022-03-04
	4	4	3	2019-08-12
	5	5	4	2023-04-23
	6	6	7	2022-09-08
	7	7	8	2022-07-01
	8	8	9	2023-10-10
	9	9	10	2022-09-05
	10	10	6	2022-12-30
*	NULL	NULL	NULL	NULL

```
99 • INSERT INTO Payments (student_id, amount, payment_date) VALUES
100 ( 1, 500.00, '2023-01-20'),
101 ( 2, 750.00, '2023-02-25'),
102 (3,1000.00,"2022-3-4"),
103 (4,800.00,"2019-8-12"),
104 (5,900.00,"2023-4-23"),
105 (6,750.00,"2022-09-8"),
106 (7,650.00,"2022-7-1"),
107 (8,700.00,"2023-10-10"),
108 (9,550.00,"2022-09-5"),
109 (10,850.00,"2022-12-30");
110
111 • SELECT * FROM Payments;
```

Result Grid				
Filter Rows:				
Edit: Export/Import: Wrap Cell Content:				
	payment_id	student_id	amount	payment_date
▶	1	1	500.00	2023-01-20
	2	2	750.00	2023-02-25
	3	3	1000.00	2022-03-04
	4	4	800.00	2019-08-12
	5	5	900.00	2023-04-23
	6	6	750.00	2022-09-08
	7	7	650.00	2022-07-01
	8	8	700.00	2023-10-10
	9	9	550.00	2022-09-05
	10	10	850.00	2022-12-30
*	NULL	NULL	NULL	NULL

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


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

Result Grid						
Filter Rows:						
Edit: Export/Import: Wrap Cell Content: I A						
student_id	first_name	last_name	date_of_birth	email	phone_number	
1	Krishna	Patle	2001-08-12	krishnapatle@gmail.com	9325654953	
2	Kashyap	Punyawar	2001-03-04	kashyappunyawar@gmail.com	9325655453	
3	Harshal	Meshram	2002-05-27	harshalmeshram@gmail.com	9125654953	
4	Nitin	Turkar	2000-12-01	nitinturkar@gmail.com	8698454798	
5	Vikas	Nagpure	2001-04-23	vikasnagpure@gmail.com	9124524953	
6	Shivam	Kale	2001-03-16	shivamkale@gmail.com	7825654953	
7	Ruchika	Chafekar	2001-06-30	ruchikachafekar@gmail.com	7447654906	
8	Neha	Patle	2003-09-03	nehapatle@gmail.com	7825654901	
9	Pratiksha	Katre	2000-05-12	pratikshakatre@gmail.com	9125654953	
10	Shruti	Kolhe	2001-03-06	shrutikolhe@gmail.com	6725654900	
11	John	Doe	1995-08-15	john.doe@example.com	1234567890	
*	NULL	NULL	NULL	NULL	NULL	

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.

```
117 • INSERT INTO Enrollments (student_id, course_id, enrollment_date)
118   VALUES (1, 2, '2023-10-26');
119 • SELECT * FROM Enrollments;
```

Result Grid				
Filter Rows:				
Edit: Export/Import: Wrap Cell Content: I A				
enrollment_id	student_id	course_id	enrollment_date	
1	1	1	2023-10-01	
2	2	2	2023-10-02	
3	3	5	2022-03-04	
4	4	3	2019-08-12	
5	5	4	2023-04-23	
6	6	7	2022-09-08	
7	7	8	2022-07-01	
8	8	9	2023-10-10	
9	9	10	2022-09-05	
10	10	6	2022-12-30	
11	1	2	2023-10-26	
*	NULL	NULL	NULL	

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

```
121 • UPDATE Teacher SET email = 'kailash12@example.com' WHERE teacher_id = 3;
122 • SELECT * FROM Teacher;
```

Result Grid				
Filter Rows:				
Edit: Export/Import: Wrap Cell Content: I A				
teacher_id	first_name	last_name	email	
1	Mr. Smith	Taylor	smith@email.com	
2	Ms. Jones	Swift	jones@email.com	
3	Mr. Kailas	Shekhar	kailash12@example.com	
4	Mr. Pruthvi	Chaudhary	pruthvi@gmail.com	
5	Mr. Rajat	Patidar	rajat@gmail.com	
6	Ms. Yogita	Lanjewar	yogita@gmail.com	
7	Ms. Ragini	Yadav	ragini@gmail.com	
8	Mr. Saurabh	Gedekar	saurabh@gmail.com	
9	Ms. Neha	Patle	nehapatle@gmail.com	
10	Mr. Harshal	Meshram	harshal@gmail.com	
*	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.

```
124 • DELETE FROM Enrollments WHERE student_id = 1 AND course_id = 1;
125 • SELECT * FROM Enrollments;
```

Result Grid				
Filter Rows:				
Edit:				
Export/Import:				
Wrap Cell Content:				
	enrollment_id	student_id	course_id	enrollment_date
▶	2	2	2	2023-10-02
	3	3	5	2022-03-04
	4	4	3	2019-08-12
	5	5	4	2023-04-23
	6	6	7	2022-09-08
	7	7	8	2022-07-01
	8	8	9	2023-10-10
	9	9	10	2022-09-05
	10	10	6	2022-12-30
*	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.

```
128 • UPDATE Courses SET teacher_id = 5 WHERE course_id = 4;
129 • SELECT * FROM Courses;
```

Result Grid				
Filter Rows:				
Edit:				
Export/Import:				
Wrap Cell Content:				
	course_id	course_name	credits	teacher_id
▶	1	Mathematics	3	1
	2	English	3	2
	3	History	2	3
	4	Geograhpy	3	5
	5	C++	4	5
	6	Python	4	6
	7	Economy	2	7
	8	Science	4	8
	9	Social Science	2	9
	10	Java	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.

```
131 • DELETE FROM Enrollments WHERE student_id = 6;
132 • DELETE FROM Students WHERE student_id = 6;
133 • SELECT *FROM Enrollments;
```

Result Grid				
Filter Rows:				
Edit:				
Export/Import:				
Wrap Cell Content:				
	enrollment_id	student_id	course_id	enrollment_date
▶	2	2	2	2023-10-02
	3	3	5	2022-03-04
	4	4	3	2019-08-12
	5	5	4	2023-04-23
	7	7	8	2022-07-01
	8	8	9	2023-10-10
	9	9	10	2022-09-05
	10	10	6	2022-12-30
*	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.

```
135 • UPDATE Payments SET amount = 200 WHERE payment_id = 7;
136 • SELECT *FROM Payments;
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
payment_id	student_id	amount	payment_date	
1	1	500.00	2023-01-20	
2	2	750.00	2023-02-25	
3	3	1000.00	2022-03-04	
4	4	800.00	2019-08-12	
5	5	900.00	2023-04-23	
6	6	750.00	2022-09-08	
7	7	200.00	2022-07-01	
8	8	700.00	2023-10-10	
9	9	550.00	2022-09-05	
10	10	850.00	2022-12-30	
* NULL	NULL	NULL	NULL	

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.

```
138 • SELECT Students.first_name, Students.last_name, SUM(Payments.amount) AS total_payments
139 FROM Students
140 JOIN Payments ON Students.student_id = Payments.student_id
141 GROUP BY Students.student_id, Students.first_name, Students.last_name;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
first_name	last_name	total_payments	
Krishna	Patle	500.00	
Kashyap	Punyawar	750.00	
Harshal	Meshram	1000.00	
Nitin	Turkar	800.00	
Vikas	Nagpure	900.00	
Shivam	Kale	750.00	
Ruchika	Chafekar	200.00	
Neha	Patle	700.00	
Pratiksha	Katre	550.00	
Shruti	Kolhe	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.

```
143 • SELECT Courses.course_name, COUNT(Enrollments.student_id) AS enrolled_students
144 FROM Courses
145 LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
146 GROUP BY Courses.course_id, Courses.course_name;
147
```

Result Grid		
	Filter Rows:	Export: Wrap Cell Content:
course_name	enrolled_students	
Mathematics	0	
English	1	
History	1	
Geograhpy	1	
C++	1	
Python	1	
Economy	0	
Science	1	
Social Science	1	
Java	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.

```
148 • SELECT Students.first_name, Students.last_name
149 FROM Students
150 LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id
151 WHERE Enrollments.student_id IS NULL;
152
```


Result Grid		
	Filter Rows:	Export: Wrap Cell Content:
first_name	last_name	
Krishna	Patle	
Shivam	Kale	
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.

```

153 • SELECT Students.first_name, Students.last_name, Courses.course_name, Enrollments.enrollment_date
154 FROM Students
155 JOIN Enrollments ON Students.student_id = Enrollments.student_id
156 JOIN Courses ON Enrollments.course_id = Courses.course_id;
157

```

Result Grid  Filter Rows: Export:  Wrap Cell Content: 

	first_name	last_name	course_name	enrollment_date
▶	Kashyap	Punyawar	English	2023-10-02
	Harshal	Meshram	C++	2022-03-04
	Nitin	Turkar	History	2019-08-12
	Vikas	Nagpure	Geography	2023-04-23
	Ruchika	Chafekar	Science	2022-07-01
	Neha	Patle	Social Science	2023-10-10
	Pratiksha	Katre	Java	2022-09-05
	Shruti	Kolhe	Python	2022-12-30

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.

```

153 • SELECT Students.first_name, Students.last_name, Courses.course_name, Enrollments.enrollment_date
154 FROM Students
155 JOIN Enrollments ON Students.student_id = Enrollments.student_id

```

Result Grid  Filter Rows: Export:  Wrap Cell Content: 




	first_name	last_name	course_name
▶	Mr. Smith	Taylor	Mathematics
	Mr. Smith	Taylor	Mathematics
	Ms. Jones	Swift	English
	Ms. Jones	Swift	English
	Mr. Kailas	Shekhar	History
	Mr. Kailas	Shekhar	History
	Mr. Pruthvi	Chaudhary	Geography
	Mr. Rajat	Patidar	Geography
	Mr. Rajat	Patidar	C++
	Mr. Rajat	Patidar	C++
	Ms. Yogita	Lanjewar	Python
	Ms. Yogita	Lanjewar	Python
	Ms. Ragini	Yadav	Economy
	Ms. Ragini	Yadav	Economy
	Mr. Saurabh	Gedekar	Science
	Mr. Saurabh	Gedekar	Science
	Ms. Neha	Patle	Social Science
	Ms. Neha	Patle	Social Science
	Mr. Harshal	Meshram	Java
	Mr. Harshal	Meshram	Java

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.

```

162 • SELECT Students.first_name, Students.last_name, Enrollments.enrollment_date
163 FROM Students
164 JOIN Enrollments ON Students.student_id = Enrollments.student_id
165 JOIN Courses ON Enrollments.course_id = Courses.course_id
166 WHERE Courses.course_name = 'Science';




```

Result Grid  Filter Rows: Export:  Wrap Cell Content: 

	first_name	last_name	enrollment_date
▶	Ruchika	Chafekar	2022-07-01

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.

```
168 • SELECT Students.first_name, Students.last_name
169 FROM Students
170 LEFT JOIN Payments ON Students.student_id = Payments.student_id
171 WHERE Payments.payment_id IS NULL;
172
```

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

	first_name	last_name
▶	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

```
173 • SELECT Courses.course_name
174 FROM Courses
175 LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
176 WHERE Enrollments.enrollment_id IS NULL;
177
```

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

	course_name
▶	Mathematics
	Economy
	Mathematics
	English
	History
	Geograhpy
	C++
	Python
	Economy
	Science
	Social Science
	Java

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.

```

178 • SELECT DISTINCT E1.student_id, S.first_name, S.last_name
179 FROM Enrollments E1
180 JOIN Enrollments E2 ON E1.student_id = E2.student_id AND E1.enrollment_id <> E2.enrollment_id

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
student_id	first_name	last_name	

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

```

183 • SELECT Teacher.first_name, Teacher.last_name
184 FROM Teacher
185 LEFT JOIN Courses ON Teacher.teacher_id = Courses.teacher_id
186 WHERE Courses.course_id IS NULL;
187

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
first_name	last_name		

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 .

```

459 • SELECT
460     course_id,
461     AVG(enrollment_count) AS average_students_enrolled
462 FROM (
463     SELECT
464         course_id,
465         COUNT(student_id) AS enrollment_count
466     FROM enrollments
467     GROUP BY course_id
468 ) AS course_enrollments
469 GROUP BY course_id;

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
course_id	average_students_enrolled		
2	1.0000		
3	1.0000		
4	1.0000		
5	1.0000		
6	1.0000		
8	1.0000		
9	1.0000		
10	1.0000		

- 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.

```
471 • SELECT student_id
472 FROM payments
473 WHERE amount = (SELECT MAX(amount) FROM payments);
474
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
student_id				
▶	3			

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

```
475 • SELECT
476     course_id,
477     COUNT(student_id) AS enrollment_count
478 FROM enrollments
479 GROUP BY course_id
480 ORDER BY enrollment_count DESC
481 LIMIT 1;
482
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
course_id		enrollment_count			
▶	2	1			

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


```

483 • SELECT
484     teacher_id,
485     SUM(amount) AS total_payments
486 FROM (
487     SELECT
488         e.course_id,
489         p.amount,
490         c.teacher_id
491     FROM enrollments e
492     JOIN payments p ON e.student_id = p.student_id AND e.course_id = e.course_id
493     JOIN courses c ON e.course_id = c.course_id
494 ) AS course_payments
495 GROUP BY teacher_id;

```

Result Grid		
Filter Rows:		
Export: Wrap Cell Content:		
teacher_id	total_payments	
2	750.00	
5	1900.00	
3	800.00	
8	200.00	
9	700.00	
10	550.00	
6	850.00	

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

```

497 • SELECT student_id
498 FROM enrollments
499 GROUP BY student_id
500 HAVING COUNT(DISTINCT course_id) = (SELECT COUNT(*) FROM courses);
501

```

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

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

```

502 • SELECT teacher_id
503 FROM teacher
504 WHERE teacher_id NOT IN (SELECT DISTINCT teacher_id FROM courses);

```

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

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

```

506 • SELECT AVG(age) AS average_age
507 FROM (
508     SELECT student_id, TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
509     FROM students
510 ) AS student_age;
511

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	average_age			
▶	22.4545			

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

```




513 • SELECT course_id
514 FROM courses
515 WHERE course_id NOT IN (SELECT DISTINCT course_id FROM enrollments);
516
517

```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	course_id				
▶	1				
	11				
	12				
	13				
	14				
	15				
	16				
	7				
	17				
	18				
	19				
	20				
▶	NULL				

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

```
517 • SELECT
518     student_id,
519     SUM(amount) AS total_payments
520 FROM payments
521 GROUP BY student_id;
522
```

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 		
	student_id	total_payments
▶	1	500.00
	2	750.00
	3	1000.00
	4	800.00
	5	900.00
	6	750.00
	7	200.00
	8	700.00
	9	550.00
	10	850.00




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.

```
523 • SELECT student_id
524 FROM payments
525 GROUP BY student_id
526 HAVING COUNT(payment_id) > 1;
527
528
```

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 		
	student_id	

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.

```
528 • SELECT
529     s.student_id,
530     s.first_name,
531     s.last_name,
532     COALESCE(SUM(p.amount), 0) AS total_payments
533 FROM students s
534 LEFT JOIN payments p ON s.student_id = p.student_id
535 GROUP BY s.student_id, s.first_name, s.last_name;
536
537
```

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 				
	student_id	first_name	last_name	total_payments
▶	1	Krishna	Patle	500.00
	2	Kashyap	Punyawar	750.00
	3	Harshal	Meshram	1000.00
	4	Nitin	Turkar	800.00
	5	Vikas	Nagpure	900.00
	6	Shivam	Kale	750.00
	7	Ruchika	Chafekar	200.00
	8	Neha	Patle	700.00
	9	Pratiksha	Katre	550.00
	10	Shruti	Kolhe	850.00
	11	John	Doe	0.00

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.

```

537 • SELECT
538     c.course_name,
539     COUNT(e.student_id) AS student_count
540 FROM courses c
541 LEFT JOIN enrollments e ON c.course_id = e.course_id
542 GROUP BY c.course_id, c.course_name;
543
544

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	course_name	student_count			
▶	Mathematics	0			
	English	1			
	History	1			
	Geograhpy	1			
	C++	1			
	Python	1			
	Economy	0			
	Science	1			
	Social Science	1			
	Java	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.

```

544 • SELECT AVG(amount) AS average_payment_amount
545 FROM payments;
546
547

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	average_payment_amount				
▶	700.000000				