ASSIGNMENT 2:STUDENT INFORMATION SYSTEM

SUBMITTED BY :SRINIDHI.V

TASK 1: DATABASE DESIGN

1. Create the database named "SISDB"

```
DROP DATABASE IF EXISTS SISDB;
CREATE DATABASE SISDB;
USE SISDB;
```

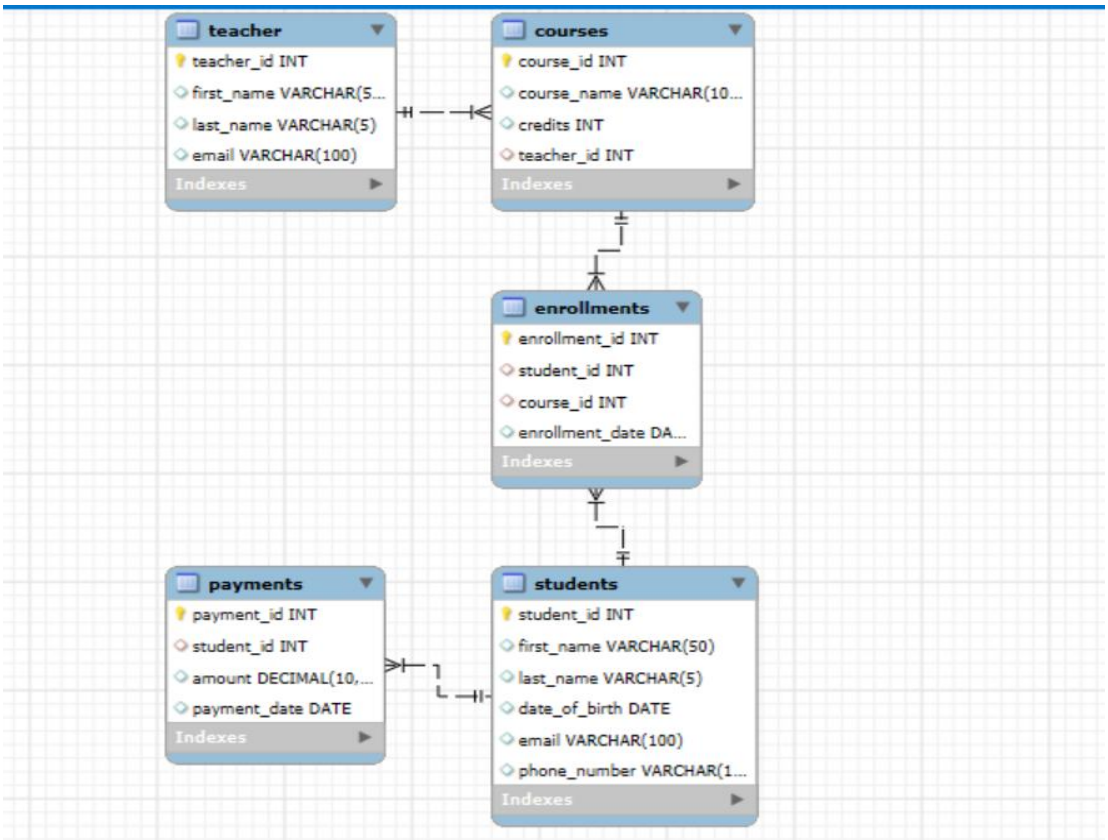
2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

a. Students
b. Courses
c. Enrollments
d. Teacher
e. Payments

```
CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(5),
    date_of_birth DATE,
    email VARCHAR(100) UNIQUE,
    phone_number VARCHAR(15)
);

CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(5),
    email VARCHAR(100) UNIQUE
);

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

```
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)
);

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)
);
```

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



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

4.1 PRIMARY KEY

```sql
CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(5),
    date_of_birth DATE,
    email VARCHAR(100) UNIQUE,
    phone_number VARCHAR(15)
);
```

4.2 FOREIGN KEY

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

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)
);
```

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

INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES
('Sri', 'V', '2000-01-15', 'sri.r@example.com', '9876500010'),
('Mickey', 'R', '1999-05-21', 'mickey.r@example.com', '9876500020'),
('Jenny', 'M', '2001-09-12', 'jenny.m@example.com', '9876500030'),
('Charls', 'D', '2000-07-08', 'charls.d@example.com', '9876500040'),
('Sujana', 'N', '1998-03-30', 'sujana.n@example.com', '9876500050'),
('Sai', 'V', '2002-11-05', 'sai.v@example.com', '9876500060'),
('Jack', 'D', '2001-12-17', 'jack.d@example.com', '9876500070'),
('Srinidhi', 'V', '2003-12-02', 'srinidhi.r@example.com', '9876500080'),

('Aakash', 'K', '2003-12-13', 'aakash.m@example.com', '9876500090'),
('Benita', 'J', '2001-10-14', 'benita.j@example.com', '9876500100');

INSERT INTO Teacher (first_name, last_name, email) VALUES
('Madhu', 'K', 'madhu.k@school.com'),
('Raj', 'S', 'raj.s@school.com'),
('Sign', 'P', 'sign.p@school.com');

INSERT INTO Courses (course_name, credits, teacher_id) VALUES
('Mathematics', 4, 1),
('Physics', 3, 2),
('Chemistry', 4, 3),
('Biology', 3, 1),
('English', 2, 2),
('Computer Science', 4, 3),
('History', 3, 1),
('Geography', 3, 2),
('Economics', 3, 3),
('Tamil', 2, 1);

INSERT INTO Enrollments (student_id, course_id, enrollment_date) VALUES
(1, 1, '2024-01-10'),
(2, 2, '2024-01-11'),
(3, 3, '2024-01-12'),
(4, 1, '2024-01-13'),
(5, 4, '2024-01-14'),
(6, 5, '2024-01-15'),
(7, 6, '2024-01-16'),
(8, 7, '2024-01-17'),
(9, 8, '2024-01-18'),
(10, 9, '2024-01-19');

INSERT INTO Payments (student_id, amount, payment_date) VALUES
(1, 1000.00, '2024-02-01'),
(2, 1200.00, '2024-02-02'),
(3, 1100.00, '2024-02-03'),
(4, 1300.00, '2024-02-04'),
(5, 1250.00, '2024-02-05'),
(6, 1400.00, '2024-02-06'),
(7, 1150.00, '2024-02-07'),
(8, 1500.00, '2024-02-08'),
(9, 1600.00, '2024-02-09'),
(10, 1700.00, '2024-02-10');

TASK 2:Select, Where, Between, AND, LIKE:

1. Write an SQL query to insert a new student into the "Students" table with the following details:
a. First Name: John
b. Last Name: Doe
c. Date of Birth: 1995-08-15
d. Email: john.doe@example.com
e. Phone Number: 1234567890

```
99 •  INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
100   VALUES ('John', 'D', '1995-08-15', 'john.doe@example.com', '1234567890');
101 •  SELECT * FROM Students;
102
103
```

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

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Sri	V	2000-01-15	sri.r@example.com	9876500010
2	Mickey	R	1999-05-21	mickey.r@example.com	9876500020
3	Jenny	M	2001-09-12	jenny.m@example.com	9876500030
4	Charls	D	2000-07-08	charls.d@example.com	9876500040
5	Sujana	N	1998-03-30	sujana.n@example.com	9876500050
6	Sai	V	2002-11-05	sai.v@example.com	9876500060
7	Jack	D	2001-12-17	jack.d@example.com	9876500070
8	Srinidhi	V	2003-12-02	srinidhi.r@example.com	9876500080
9	Aakash	K	2003-12-13	aakash.m@example.com	9876500090
10	Benita	J	2001-10-14	benita.j@example.com	9876500100
11	John	D	1995-08-15	john.doe@example.com	1234567890
NULL	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.

```
103 •  INSERT INTO Enrollments (student_id, course_id, enrollment_date)
104    VALUES (1, 1, '2024-03-01');
105 •  SELECT * FROM Enrollments;
106
107
```

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

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-11
3	3	3	2024-01-12
4	4	1	2024-01-13
5	5	4	2024-01-14
6	6	5	2024-01-15
7	7	6	2024-01-16
8	8	7	2024-01-17
9	9	8	2024-01-18
10	10	9	2024-01-19
11	1	1	2024-03-01
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.

```
107
108 •  UPDATE Teacher
109    SET email = 'raj.updated@school.com'
110    WHERE teacher_id = 2;
111
```

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

teacher_id	first_name	last_name	email
1	Madhu	K	madhu.k@school.com
2	Raj	S	raj.updated@school.com
3	Sign	P	sign.p@school.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.

```
114 •  DELETE FROM Enrollments
115    WHERE student_id = 1 AND course_id = 2;
116
117 •  SELECT* FROM Enrollments;
```

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

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-11
3	3	3	2024-01-12
4	4	1	2024-01-13
5	5	4	2024-01-14
6	6	5	2024-01-15
7	7	6	2024-01-16
8	8	7	2024-01-17
9	9	8	2024-01-18
10	10	9	2024-01-19
11	1	1	2024-03-01
NULL	NULL	NULL	NULL

Students 27 | Enrollments 28 | Teacher 29 | Enrollments 30 | Apply

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

```
-- Assigning Madhu (teacher_id = 1) to course_id = 5 (English)
UPDATE Courses
SET teacher_id = 1
WHERE course_id = 5;
SELECT *FROM Courses;
```

5	English	2	1

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.

```
124     -- Delete student_id = 10 (Benita)
125 •   DELETE FROM Enrollments WHERE student_id = 10;
126 •   DELETE FROM Payments WHERE student_id = 10;
127 •   DELETE FROM Students WHERE student_id = 10;
128 •   SELECT * FROM Students;
```

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

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Sri	V	2000-01-15	sri.r@example.com	9876500010
2	Mickey	R	1999-05-21	mickey.r@example.com	9876500020
3	Jenny	M	2001-09-12	jenny.m@example.com	9876500030
4	Charls	D	2000-07-08	charls.d@example.com	9876500040
5	Sujana	N	1998-03-30	sujana.n@example.com	9876500050
6	Sai	V	2002-11-05	sai.v@example.com	9876500060
7	Jack	D	2001-12-17	jack.d@example.com	9876500070
8	Srinidhi	V	2003-12-02	srinidhi.r@example.com	9876500080
9	Aakash	K	2003-12-13	aakash.m@example.com	9876500090
11	John	D	1995-08-15	john.doe@example.com	1234567890
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.

```
131     -- Update payment for payment_id = 1
132 •   UPDATE Payments
133     SET amount = 1500.00
134     WHERE payment_id = 1;
135 •   SELECT* FROM Payments;
```

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

payment_id	student_id	amount	payment_date
1	1	1500.00	2024-02-01
2	2	1200.00	2024-02-02
3	3	1100.00	2024-02-03
4	4	1300.00	2024-02-04
5	5	1250.00	2024-02-05
6	6	1400.00	2024-02-06
7	7	1150.00	2024-02-07
8	8	1500.00	2024-02-08
9	9	1600.00	2024-02-09
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.

```
141     -- TASK 3
142 •   SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
143     FROM Students s
144     JOIN Payments p ON s.student_id = p.student_id
145     WHERE s.student_id = 1
146     GROUP BY s.student_id;
147
148
```

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

first_name	last_name	total_payment
Sri	V	1500.00

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.

Limit to 1000 rows

```
142 •   SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
143     FROM Students s
144     JOIN Payments p ON s.student_id = p.student_id
145     WHERE s.student_id = 1
146     GROUP BY s.student_id;
```

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

course_name	enrolled_students
Mathematics	3
Physics	1
Chemistry	1
Biology	1
English	1
Computer Science	1
History	1
Geography	1
Economics	0
Tamil	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.

```
153 •  SELECT s.first_name, s.last_name
154    FROM Students s
155    LEFT JOIN Enrollments e ON s.student_id = e.student_id
156    WHERE e.enrollment_id IS NULL;
```

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

first_name	last_name
John	D

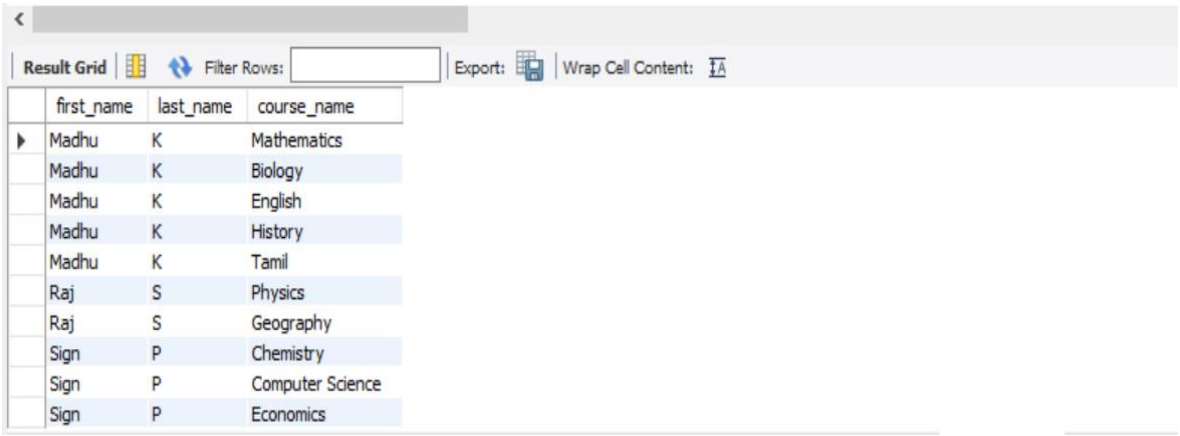
4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.

```
158 •  SELECT s.first_name, s.last_name, c.course_name
159    FROM Students s
160    JOIN Enrollments e ON s.student_id = e.student_id
161    JOIN Courses c ON e.course_id = c.course_id;
162
```

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

first_name	last_name	course_name
Sri	V	Mathematics
Sri	V	Mathematics
Mickey	R	Physics
Jenny	M	Chemistry
Charls	D	Mathematics
Sujana	N	Biology
Sai	V	English
Jack	D	Computer Science
Srinidhi	V	History
Aakash	K	Geography

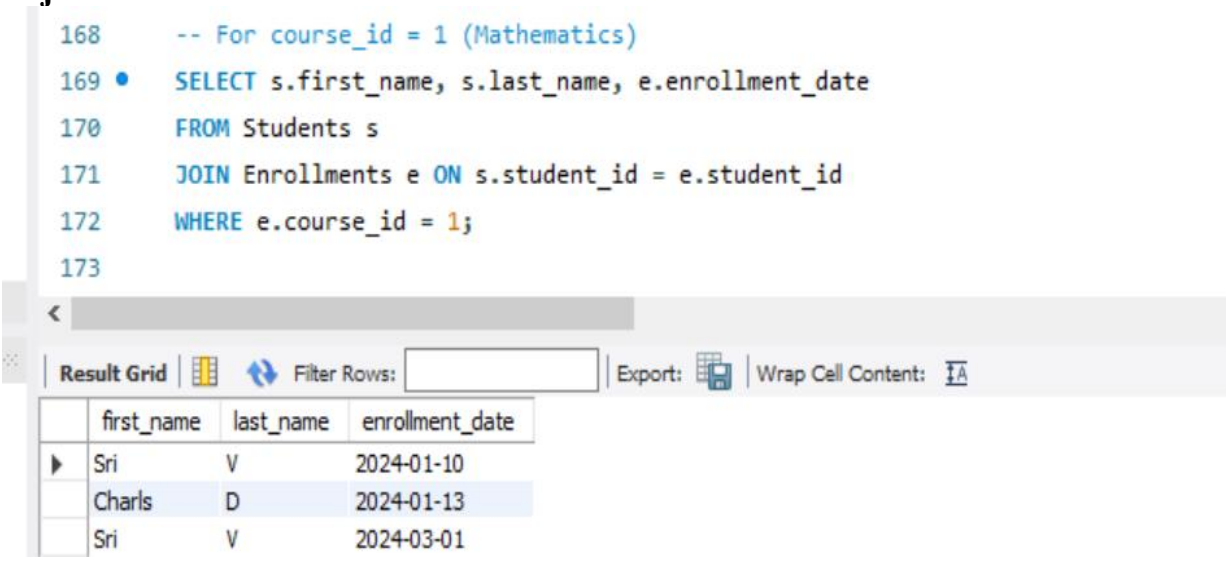
5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.



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

first_name	last_name	course_name
Madhu	K	Mathematics
Madhu	K	Biology
Madhu	K	English
Madhu	K	History
Madhu	K	Tamil
Raj	S	Physics
Raj	S	Geography
Sign	P	Chemistry
Sign	P	Computer Science
Sign	P	Economics

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.

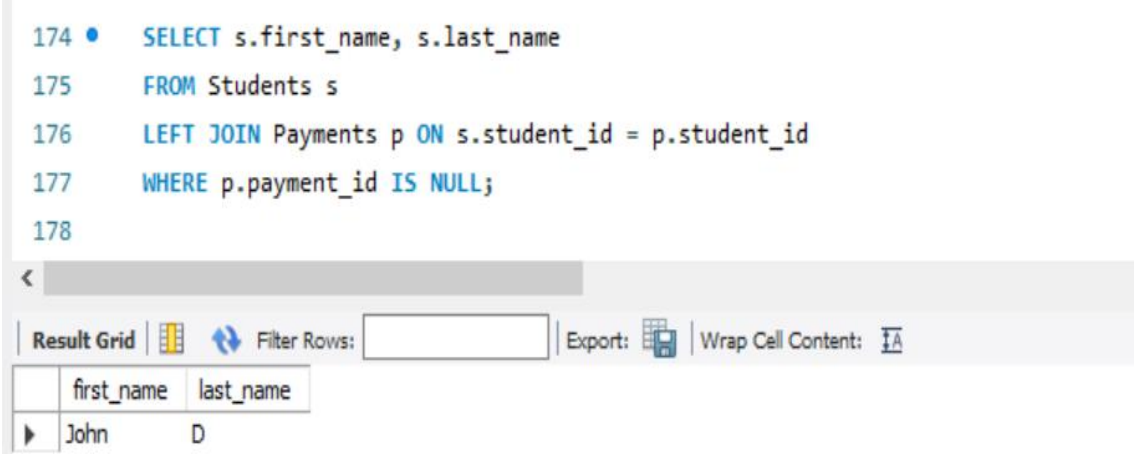


```
168    -- For course_id = 1 (Mathematics)
169 •  SELECT s.first_name, s.last_name, e.enrollment_date
170    FROM Students s
171    JOIN Enrollments e ON s.student_id = e.student_id
172    WHERE e.course_id = 1;
173
```

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

first_name	last_name	enrollment_date
Sri	V	2024-01-10
Charls	D	2024-01-13
Sri	V	2024-03-01

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.



```
174 •  SELECT s.first_name, s.last_name
175    FROM Students s
176    LEFT JOIN Payments p ON s.student_id = p.student_id
177    WHERE p.payment_id IS NULL;
178
```

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

first_name	last_name
John	D

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.

```
179 •   SELECT c.course_name
180     FROM Courses c
181     LEFT JOIN Enrollments e ON c.course_id = e.course_id
182     WHERE e.enrollment_id IS NULL;
183
184
```

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

course_name
Economics
Tamil

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.

```
184 •   SELECT s.first_name, s.last_name, COUNT(e.course_id) AS course_count
185     FROM Students s
186     JOIN Enrollments e ON s.student_id = e.student_id
187     GROUP BY s.student_id
188     HAVING COUNT(e.course_id) > 1;
189
```

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

first_name	last_name	course_count
Sri	V	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

```
190 •   SELECT t.first_name, t.last_name
191     FROM Teacher t
192     LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
193     WHERE c.course_id IS NULL;
194
```

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

first_name	last_name

TASK 4: SUBQUERY AND ITS TYPES

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

```
195     -- 1. Average number of students enrolled in each course
196 •   SELECT AVG(student_count) AS avg_students_per_course
197     FROM (
198         SELECT course_id, COUNT(*) AS student_count
199         FROM Enrollments
200         GROUP BY course_id
201     ) AS course_enrollments;
```

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

avg_students_per_course
1.2500

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.

```
203     -- 2. Student(s) who made the highest payment
204 •   SELECT s.first_name, s.last_name, p.amount
205     FROM Payments p
206     JOIN Students s ON p.student_id = s.student_id
207     WHERE p.amount = (
208         SELECT MAX(amount) FROM Payments
209     );
210
```

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

first_name	last_name	amount
Aakash	K	1600.00

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

```
211     -- 3. Courses with the highest number of enrollments
212 •   SELECT c.course_name, COUNT(e.enrollment_id) AS enrollment_count
213     FROM Courses c
214     JOIN Enrollments e ON c.course_id = e.course_id
215     GROUP BY c.course_id
216     HAVING COUNT(e.enrollment_id) = (
217         SELECT MAX(enrollment_count)
218         FROM (
219             SELECT course_id, COUNT(*) AS enrollment_count
220             FROM Enrollments
221             GROUP BY course_id
222         ) AS sub
223     );
```

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

course_name	enrollment_count
Mathematics	3

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

```
225     -- 4. Total payments made to courses taught by each teacher
226 •   SELECT t.first_name, t.last_name, SUM(p.amount) AS total_payments
227     FROM Teacher t
228     JOIN Courses c ON t.teacher_id = c.teacher_id
229     JOIN Enrollments e ON c.course_id = e.course_id
230     JOIN Payments p ON e.student_id = p.student_id
231     GROUP BY t.teacher_id;
```

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

first_name	last_name	total_payments
Madhu	K	8450.00
Raj	S	2800.00
Sign	P	2250.00

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

```
233     -- 5. Students enrolled in ALL available courses
234 •   SELECT s.first_name, s.last_name
235     FROM Students s
236     WHERE NOT EXISTS (
237         SELECT c.course_id FROM Courses c
238         WHERE NOT EXISTS (
239             SELECT * FROM Enrollments e
240             WHERE e.course_id = c.course_id AND e.student_id = s.student_id
241         )
242     );
```

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

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.

```
244     -- 6. Teachers not assigned to any courses
245 •   SELECT first_name, last_name
246     FROM Teacher
247     WHERE teacher_id NOT IN (
248         SELECT DISTINCT teacher_id FROM Courses WHERE teacher_id IS NOT NULL
249     );
```

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

first_name	last_name

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

```
251     -- 7. Average age of all students
252 •   SELECT AVG(TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE())) AS average_age
253     FROM Students;
254
```

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

average_age
24.0000

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

```
255     -- 8. Courses with no enrollments
256 •   SELECT course_name
257     FROM Courses
258     WHERE course_id NOT IN (
259         SELECT DISTINCT course_id FROM Enrollments
260     );
261
```

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

course_name
Economics
Tamil

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

```
262     -- 9. Total payments made by each student for each course
263 •   SELECT s.first_name, s.last_name, c.course_name, SUM(p.amount) AS total_payment
264     FROM Students s
265     JOIN Enrollments e ON s.student_id = e.student_id
266     JOIN Courses c ON e.course_id = c.course_id
267     JOIN Payments p ON p.student_id = s.student_id
268     GROUP BY s.student_id, c.course_id;
```

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

first_name	last_name	course_name	total_payment
Sri	V	Mathematics	3000.00
Mickey	R	Physics	1200.00
Jenny	M	Chemistry	1100.00
Charls	D	Mathematics	1300.00
Sujana	N	Biology	1250.00
Sai	V	English	1400.00
Jack	D	Computer Science	1150.00
Srinidhi	V	History	1500.00
Aakash	K	Geography	1600.00

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.

```
270     -- 10. Students who made more than one payment
271 •   SELECT s.first_name, s.last_name, COUNT(p.payment_id) AS payment_count
272     FROM Students s
273     JOIN Payments p ON s.student_id = p.student_id
274     GROUP BY s.student_id
275     HAVING COUNT(p.payment_id) > 1;
276
```

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

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.

```
277     -- 11. Total payments made by each student
278 •   SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
279     FROM Students s
280     JOIN Payments p ON s.student_id = p.student_id
281     GROUP BY s.student_id;
```

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

first_name	last_name	total_payment
Sri	V	1500.00
Mickey	R	1200.00
Jenny	M	1100.00
Charls	D	1300.00
Sujana	N	1250.00
Sai	V	1400.00
Jack	D	1150.00
Srinidhi	V	1500.00
Aakash	K	1600.00

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.

```
283     -- 12. Course names and count of students enrolled in each
284 •   SELECT c.course_name, COUNT(e.enrollment_id) AS num_students
285     FROM Courses c
286     LEFT JOIN Enrollments e ON c.course_id = e.course_id
287     GROUP BY c.course_id;
288
```

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

course_name	num_students
Mathematics	3
Physics	1
Chemistry	1
Biology	1
English	1
Computer Science	1
History	1
Geography	1
Economics	0
Tamil	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.

```
289     -- 13. Average payment amount made by students
290 •   SELECT s.first_name, s.last_name, AVG(p.amount) AS avg_payment
291     FROM Students s
292     JOIN Payments p ON s.student_id = p.student_id
293     GROUP BY s.student_id;
294
```

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

first_name	last_name	avg_payment
Sri	V	1500.000000
Mickey	R	1200.000000
Jenny	M	1100.000000
Charls	D	1300.000000
Sujana	N	1250.000000
Sai	V	1400.000000
Jack	D	1150.000000
Srinidhi	V	1500.000000
Aakash	K	1600.000000