

Assignment:2

Student Information System (SIS)

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 if not exists SISDB;
2 • use SISDB;
3
4 • create table if not exists Students (
5     student_id int PRIMARY KEY,
6     first_name varchar(50),
7     last_name varchar(50),
8     date_of_birth date,
9     email varchar(50),
10    phone_no varchar(12)
11 );
12 • create table if not exists Teacher (
13     teacher_id int PRIMARY KEY,
14     first_name varchar(50),
15     last_name varchar(50),
16     email varchar(25)
17 );
18
19 • create table if not exists Courses (
20     course_id int PRIMARY KEY,
21     course_name varchar(100),
22     credits int,
23     teacher_id int, FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
24 );
25
26 • create table if not exists Enrollments (
27     enrollment_id int PRIMARY KEY,
28     student_id int, FOREIGN KEY (student_id) REFERENCES Students(student_id),
29     course_id int, FOREIGN KEY (course_id) REFERENCES Courses(course_id),
30     enrollment_date DATE
31 );
32 • create table if not exists Payments (
33     payment_id int PRIMARY KEY,
34     student_id int, FOREIGN KEY (student_id) REFERENCES Students(student_id),
35     amount decimal(10, 2),
36     payment_date DATE
37 );
```

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

i. Students

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

The screenshot shows the SQL Developer interface with a script editor on the left and a result grid on the right. The script editor contains the following SQL code:

```
39 • insert into Students values(1,'Daniel', 'Brown', '1998-04-17', 'daniel.brown@yahoo.com', '8839347856'),
40 (2,'Sophia', 'Wilson', '1989-10-10', 'sophia.wilson@yahoo.com', '6378236475'),
41 (3,'Olivia', 'Anderson', '1990-12-05', 'olivia.anderson@yahoo.com', '7823127639'),
42 (4,'Michael', 'Taylor', '1995-07-04', 'michael.taylor@yahoo.com', '6739292034'),
43 (5,'Emily', 'Davis', '1993-02-19', 'emily.davis@yahoo.com', '9937214358'),
44 (6,'David', 'Miller', '1996-09-03', 'david.miller@yahoo.com', '9793123423'),
45 (7,'Alice', 'Williams', '1993-06-08', 'alice.williams@yahoo.com', '9453234358'),
46 (8,'John', 'Doe', '1997-08-10', 'john.doe@yahoo.com', '9876543210'),
47 (9,'Jane', 'Smith', '1991-03-12', 'jane.smith@yahoo.com', '9862377443'),
48 (10,'Bob', 'Johnson', '1999-11-15', 'bob.johnson@yahoo.com', '7843892356');
49 • select * from Students
```

The result grid displays the following data:

student_id	first_name	last_name	date_of_birth	email	phone_no
1	Daniel	Brown	1998-04-17	daniel.brown@yahoo.com	8839347856
2	Sophia	Wilson	1989-10-10	sophia.wilson@yahoo.com	6378236475
3	Olivia	Anderson	1990-12-05	olivia.anderson@yahoo.com	7823127639
4	Michael	Taylor	1995-07-04	michael.taylor@yahoo.com	6739292034
5	Emily	Davis	1993-02-19	emily.davis@yahoo.com	9937214358
6	David	Miller	1996-09-03	david.miller@yahoo.com	9793123423
7	Alice	Williams	1993-06-08	alice.williams@yahoo.com	9453234358
8	John	Doe	1997-08-10	john.doe@yahoo.com	9876543210
9	Jane	Smith	1991-03-12	jane.smith@yahoo.com	9862377443
10	Bob	Johnson	1999-11-15	bob.johnson@yahoo.com	7843892356

The screenshot shows the SQL Developer interface with a script editor on the left and a result grid on the right. The script editor contains the following SQL code:

```
56 • insert into Teacher values (1, 'Jane', 'Smith', 'jane.smith@yahoo.com'),
57 (2, 'John', 'Doe', 'john.doe@yahoo.com'),
58 (3, 'Emily', 'Jones', 'emily.jones@yahoo.com'),
59 (4, 'Michael', 'Johnson', 'michael.johnson@yahoo.com'),
60 (5, 'Sophia', 'Williams', 'sophia.williams@yahoo.com'),
61 (6, 'David', 'Brown', 'david.brown@yahoo.com'),
62 (7, 'Olivia', 'Taylor', 'olivia.taylor@yahoo.com'),
63 (8, 'Daniel', 'Miller', 'daniel.miller@yahoo.com'),
64 (9, 'Alice', 'Anderson', 'alice.anderson@yahoo.com'),
65 (10, 'Bob', 'Wilson', 'bob.wilson@yahoo.com');
66 • Select * from Teacher;
```

The result grid displays the following data:

teacher_id	first_name	last_name	email
1	Jane	Smith	jane.smith@yahoo.com
2	John	Doe	john.doe@yahoo.com
3	Emily	Jones	emily.jones@yahoo.com
4	Michael	Johnson	michael.johnson@yahoo.com
5	Sophia	Williams	sophia.williams@yahoo.com
6	David	Brown	david.brown@yahoo.com
7	Olivia	Taylor	olivia.taylor@yahoo.com
8	Daniel	Miller	daniel.miller@yahoo.com
9	Alice	Anderson	alice.anderson@yahoo.com
10	Bob	Wilson	bob.wilson@yahoo.com

db1
sys
Tables
Views
Stored...
Functionio...
TechShop
Tables
Views
Stored...
Functionio...

```
69 • insert into Courses values(1, 'Physical Education', 3, 1),
70 (2, 'Art', 3, 2),
71 (3, 'English Literature', 5, 3),
72 (4, 'Music', 3, 4),
73 (5, 'Computer Science', 4, 5),
74 (6, 'History', 2, 6),
75 (7, 'Chemistry', 4, 7),
76 (8, 'Physics', 4, 8),
77 (9, 'Biology', 5, 9),
78 (10, 'Mathematics', 5, 10);
79 • select * from Courses;
```

100% 23:79 1 error found

Result Grid Filter Rows: Search Edit: Export/Import:

	course_id	course_name	credits	teacher_id
1	1	Physical Education	3	1
2	2	Art	3	2
3	3	English Literature	5	3
4	4	Music	3	4
5	5	Computer Science	4	5
6	6	History	2	6
7	7	Chemistry	4	7
8	8	Physics	4	8
9	9	Biology	5	9
10	10	Mathematics	5	10
	NULL	NULL	NULL	NULL

Courses 5 Apply

Action Output

	Time	Action	Response	Duration / Fetch Time
240	00:34:09	select * from Courses LIMIT 0, 1000	10 row(s) returned	0.00060 sec / 0.000...

Query Completed

db1
sys
Tables
Views
Stored...
Functionio...
TechShop
Tables
Views
Stored...
Functionio...

```
81 • insert into Enrollments values
82 (1, 1, 1, '2023-02-01'),
83 (2, 2, 3, '2023-01-12'),
84 (3, 3, 2, '2023-04-23'),
85 (4, 4, 1, '2023-03-22'),
86 (5, 5, 4, '2023-04-20'),
87 (6, 6, 3, '2023-05-10'),
88 (7, 7, 2, '2023-06-05'),
89 (8, 8, 1, '2023-10-20'),
90 (9, 9, 4, '2023-09-25'),
91 (10, 10, 3, '2023-08-05');
92 • Select * from Enrollments
```

100% 26:92 2 errors found

Result Grid Filter Rows: Search Edit: Export/Import:

	enrollment_id	student_id	course_id	enrollment_da...
1	1	1	1	2023-02-01
2	2	2	3	2023-01-12
3	3	3	2	2023-04-23
4	4	4	1	2023-03-22
5	5	5	4	2023-04-20
6	6	6	3	2023-05-10
7	7	7	2	2023-06-05
8	8	8	1	2023-10-20
9	9	9	4	2023-09-25
10	10	10	3	2023-08-05

db1
sys
Tables
Views
Stored...
Functionio...
TechShop
Tables
Views
Stored...
Functionio...

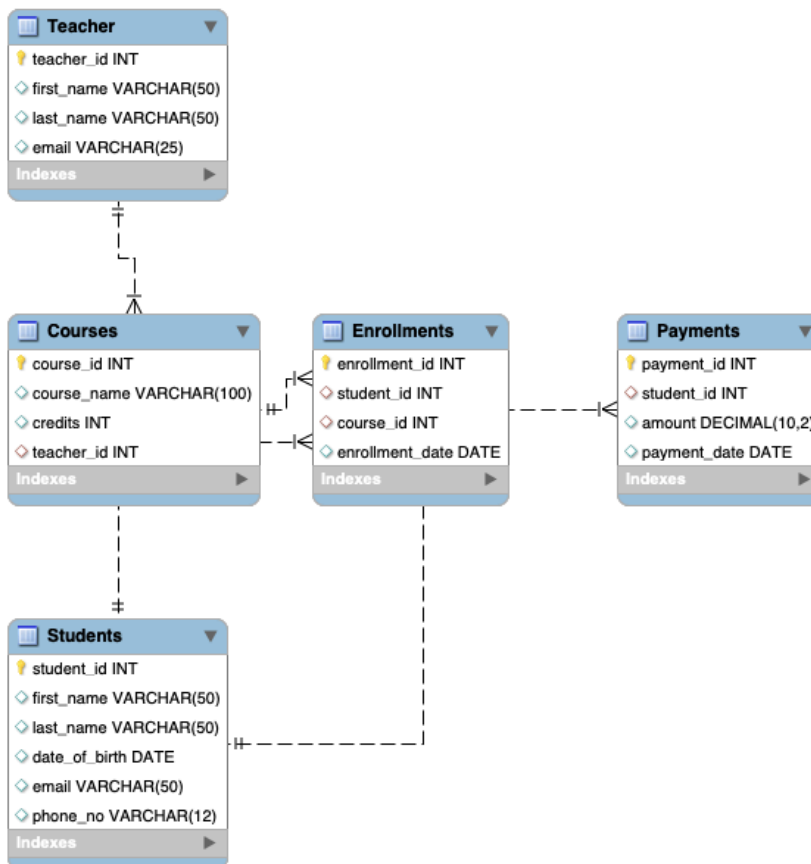
```
94 • insert into Payments values(1,1,750.00,'2023-02-01'),
95 (2,2, 500.00, '2023-01-12'),
96 (3, 3,850.00, '2023-04-23'),
97 (4,4, 1000.00, '2023-03-22'),
98 (5, 5,500.00, '2023-04-20'),
99 (6, 6,900.00, '2023-05-10'),
100 (7, 7,750.00, '2023-06-05'),
101 (8, 8,600.00, '2023-10-20'),
102 (9, 9,500.00, '2023-09-25'),
103 (10, 10,750.00, '2023-08-05');
104 • select * from Payments
105
```

100% 1:105 3 errors found

Result Grid Filter Rows: Search Edit: Export/Import:

	payment_id	student_id	amount	payment_date
1	1	1	750.00	2023-02-01
2	2	2	500.00	2023-01-12
3	3	3	850.00	2023-04-23
4	4	4	1000.00	2023-03-22
5	5	5	500.00	2023-04-20
6	6	6	900.00	2023-05-10
7	7	7	750.00	2023-06-05
8	8	8	600.00	2023-10-20
9	9	9	500.00	2023-09-25
10	10	10	750.00	2023-08-05

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



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:

- First Name: John
- Last Name: Doe
- Date of Birth: 1995-08-15
- Email: john.doe@example.com
- Phone Number: 1234567890

107 `insert into Students (student_id,first_name, last_name, date_of_birth, email, phone_no)`
 108 `VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');`
 109 `select * from Students;`

110 24:109 3 errors found

student_id	first_name	last_name	date_of_birth	email	phone_no
1	Daniel	Brown	1996-04-17	daniel.brown@yahoo.com	8839347856
2	Sophia	Wilson	1989-10-10	sophia.wilson@yahoo.com	6378236475
3	Olivia	Anderson	1990-12-05	olivia.anderson@yahoo.com	7623127639
4	Michael	Taylor	1995-07-04	michael.taylor@yahoo.com	6739292034
5	Emily	Davis	1993-02-19	emily.davis@yahoo.com	9937214358
6	David	Miller	1996-09-03	david.miller@yahoo.com	9793123423
7	Alice	Williams	1993-06-08	alice.williams@yahoo.com	9453234358
8	John	Doe	1997-08-10	john.doe@yahoo.com	9876543210
9	Jane	Smith	1991-03-12	jane.smith@yahoo.com	9862377443
10	Bob	Johnson	1999-11-15	bob.johnson@yahoo.com	7843892356
11	John	Doe	1995-08-15	john.doe@example.com	1234567890

Students 10

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.

The screenshot shows a database management interface with a sidebar on the left containing 'Tables', 'Views', 'Stored...', and 'Function...'. The main area displays SQL queries:

```
115 • insert into Enrollments (enrollment_id, student_id, course_id, enrollment_date)
116 • VALUES (11, 1, 2, '2023-08-01');
117 • select * from Enrollments;
```

Below the queries, a 'Result Grid' shows the 'Enrollments' table with 11 rows. The columns are 'enrollment_id', 'student_id', 'course_id', and 'enrollment_date'. The data is as follows:

enrollment_id	student_id	course_id	enrollment_date
2	2	3	2023-01-12
3	3	2	2023-04-23
4	4	1	2023-03-22
5	5	4	2023-04-20
6	6	3	2023-05-10
7	7	2	2023-06-05
8	8	1	2023-10-20
9	9	4	2023-09-25
10	10	3	2023-08-05
11	1	2	2023-08-01
NULL	NULL	NULL	NULL

The bottom of the interface shows 'Enrollments 11' and an 'Apply' button.

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

The screenshot shows a database management interface with a sidebar on the left containing 'Views', 'Stored...', and 'Function...'. The main area displays SQL queries:

```
119 • UPDATE Teacher Set email='emiley.jones12@gmail.com' where teacher_id=3;
120 • select * from Teacher;
```

Below the queries, a 'Result Grid' shows the 'Teacher' table with 10 rows. The columns are 'teacher_id', 'first_name', 'last_name', and 'email'. The data is as follows:

teacher_id	first_name	last_name	email
1	Jane	Smith	jane.smith@yahoo.com
2	John	Doe	john.doe@yahoo.com
3	Emily	Jones	emiley.jones12@gmail.com
4	Michael	Johnson	michael.johnson@yahoo.com
5	Sophia	Williams	sophia.williams@yahoo.com
6	David	Brown	david.brown@yahoo.com
7	Olivia	Taylor	olivia.taylor@yahoo.com
8	Daniel	Miller	daniel.miller@yahoo.com
9	Alice	Anderson	alice.anderson@yahoo.com
10	Bob	Wilson	bob.wilson@yahoo.com
NULL	NULL	NULL	NULL

The bottom of the interface shows 'Teacher 13' and an 'Apply' button.

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.

The screenshot shows a database management interface with a sidebar on the left containing 'Views', 'Stored...', and 'Function...'. The main area displays SQL queries:

```
122 • DELETE from Enrollments Where student_id=1 and course_id=1;
123 • select * from Enrollments;
```

Below the queries, a 'Result Grid' shows the 'Enrollments' table with 11 rows. The columns are 'enrollment_id', 'student_id', 'course_id', and 'enrollment_date'. The data is as follows:

enrollment_id	student_id	course_id	enrollment_date
2	2	3	2023-01-12
3	3	2	2023-04-23
4	4	1	2023-03-22
5	5	4	2023-04-20
6	6	3	2023-05-10
7	7	2	2023-06-05
8	8	1	2023-10-20
9	9	4	2023-09-25
10	10	3	2023-08-05
11	1	2	2023-08-01
NULL	NULL	NULL	NULL

The bottom of the interface shows 'Enrollments 14' and an 'Apply' button.

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

The screenshot shows a database management interface with a sidebar on the left containing 'Tables', 'Views', 'Stored...', and 'Funcio...'. The main area displays SQL queries:

```
124
125 • UPDATE Courses SET teacher_id=5 WHERE course_id=4;
126 • select * from Courses;
```

Below the queries is a 'Result Grid' showing the contents of the 'Courses' table:

course_id	course_name	credits	teacher_id
1	Physical Education	3	1
2	Art	3	2
3	English Literature	5	3
4	Music	3	5
5	Computer Science	4	5
6	History	2	6
7	Chemistry	4	7
8	Physics	4	8
9	Biology	5	9
10	Mathematics	5	10
NULL	NULL	NULL	NULL

The interface also shows 'Object Info' and 'Session' tabs, with 'No object selected' displayed. The bottom status bar indicates 'Courses 16' and 'Apply'.

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.

The screenshot shows a database management interface with a sidebar on the left containing 'Tables', 'Views', 'Stored...', and 'Funcio...'. The main area displays SQL queries:

```
128 • DELETE from Enrollments WHERE student_id=6;
129 • DELETE from Students WHERE student_id=6;
130 • select * from Enrollments;
```

Below the queries is a 'Result Grid' showing the contents of the 'Enrollments' table:

enrollment_id	student_id	course_id	enrollment_date
2	2	3	2023-01-12
3	3	2	2023-04-23
4	4	1	2023-03-22
5	5	4	2023-04-20
7	7	2	2023-06-05
8	8	1	2023-10-20
9	9	4	2023-09-25
10	10	3	2023-08-05
11	1	2	2023-08-01
NULL	NULL	NULL	NULL

The interface also shows 'Object Info' and 'Session' tabs, with 'No object selected' displayed. The bottom status bar indicates 'Enrollments 17' and 'Apply'.

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

The screenshot shows a database management interface with a sidebar on the left containing 'Tables', 'Views', 'Stored...', and 'Funcio...'. The main area displays SQL queries:

```
132 • UPDATE Payments SET amount=200 Where payment_id=7;
133 • select * from Payments;
```

Below the queries is a 'Result Grid' showing the contents of the 'Payments' table:

payment_id	student_id	amount	payment_date
1	1	750.00	2023-02-01
2	2	500.00	2023-01-12
3	3	850.00	2023-04-23
4	4	1000.00	2023-03-22
5	5	500.00	2023-04-20
6	6	900.00	2023-05-10
7	7	200.00	2023-06-05
8	8	600.00	2023-10-20
9	9	500.00	2023-09-25
10	10	750.00	2023-08-05
NULL	NULL	NULL	NULL

The interface also shows 'Object Info' and 'Session' tabs, with 'No object selected' displayed. The bottom status bar indicates 'Payments 24:133' and 'Apply'.

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.

```
135 • Select Students.first_name,Students.last_name, SUM(Payments.amount) AS total_payments
136 from Students
137 JOIN Payments ON Students.student_id= Payments.student_id
138 Group by Students.student_id,Students.first_name,Students.last_name;
```

100% 69:138 3 errors found

Result Grid			
Filter Rows: Search			
Export:			
first_name	last_name	total_payme...	
Daniel	Brown	750.00	
Sophia	Wilson	500.00	
Olivia	Anderson	850.00	
Michael	Taylor	1000.00	
Emily	Davis	500.00	
David	Miller	900.00	
Alice	Williams	200.00	
John	Doe	600.00	
Jane	Smith	500.00	
Bob	Johnson	750.00	

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.

```
140 • Select Courses.course_name,COUNT(Enrollments.student_id) AS enrolled_students
141 FROM Courses
142 LEFT JOIN Enrollments ON Courses.course_id=Enrollments.course_id
143 GROUP BY Courses.course_id, Courses.course_name;
144
```

100% 1:139 3 errors found

Result Grid		
Filter Rows: Search		
Export:		
course_name	enrolled_stude...	
Physical Education	2	
Art	3	
English Literature	2	
Music	2	
Computer Science	0	
History	0	
Chemistry	0	
Physics	0	
Biology	0	
Mathematics	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.

```
145 • SELECT Students.first_name, Students.last_name
146 FROM Students
147 LEFT JOIN Enrollments ON Students.student_id=Enrollments.student_id
148 WHERE Enrollments.student_id IS NULL;
149
```

100% 38:148 3 errors found

Result Grid Filter Rows: Search Export:

first_name	last_name
David	Miller
John	Doe

Result Grid Form Editor

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.

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

100% 1:150 3 errors found

Result Grid Filter Rows: Search Export:

first_name	last_name	course_name	enrollment_da...
Sophia	Wilson	English Literature	2023-01-12
Olivia	Anderson	Art	2023-04-23
Michael	Taylor	Physical Education	2023-03-22
Emily	Davis	Music	2023-04-20
Alice	Williams	Art	2023-06-05
John	Doe	Physical Education	2023-10-20
Jane	Smith	Music	2023-09-25
Bob	Johnson	English Literature	2023-08-05
Daniel	Brown	Art	2023-08-01

Result Grid Form Editor Field Types

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.

```
164 • SELECT Students.first_name, Students.last_name, Enrollments.enrollment_date
165 FROM Students
166 JOIN Enrollments ON Students.student_id=Enrollments.student_id
167 JOIN Courses ON Enrollments.course_id=Courses.course_id
168 WHERE Courses.course_name='Music';
```

100% 35:168 3 errors found

Result Grid Filter Rows: Search Export:

first_name	last_name	enrollment_da...
Emily	Davis	2023-04-20
Jane	Smith	2023-09-25

Result Grid Form Editor Field Types

Result 26 Read Only

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.

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

00% 35:174 3 errors found

Result Grid Filter Rows: Search Export:

first_name	last_name
John	Doe

Result Grid Form Editor Field Types

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.

```
176 • SELECT Courses.course_name
177 FROM Courses
178 LEFT JOIN Enrollments ON Courses.course_id=Enrollments.course_id
179 WHERE Enrollments.enrollment_id is NULL;
```

100%

41:179

3 errors found

Result Grid

Filter Rows:

Search

Export:

	course_name
	Computer Science
	History
	Chemistry
	Physics
	Biology
	Mathematics

Result Grid

Form Editor

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.

```
183 • SELECT DISTINCT e1.student_id, s.first_name, s.last_name
184 FROM Enrollments e1
185 JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.enrollment_id <> e2.enrollment_id
186 JOIN Students s ON e1.student_id = s.student_id
187 ORDER BY e1.student_id;
```

100% 24:187 3 errors found

Result Grid Filter Rows: Search Export:

student_id	first_name	last_name
------------	------------	-----------

Result Grid

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.*
191 FROM Teacher t
192 LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
193 WHERE c.teacher_id IS NULL;
194
195

```

100% 1:194 3 errors found

Result Grid Filter Rows: Search Export:

	teacher_id	first_name	last_name	email
▶ 4		Michael	Johnson	michael.johnson@yahoo.com

Result Grid

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.

```

196 • SELECT
197     c.course_id,
198     c.course_name,
199     AVG(enrollment_count) AS average_students_enrolled
200 FROM Courses c
201 LEFT JOIN (
202     SELECT course_id, COUNT(DISTINCT student_id) AS enrollment_count
203     FROM Enrollments
204     GROUP BY course_id
205 ) e ON c.course_id = e.course_id
206 GROUP BY c.course_id, c.course_name;
207

```

100% 1:196 3 errors found

Result Grid Filter Rows: Search Export:

	course_id	course_name	average_students_enroll...
▶ 1		Physical Education	2.0000
2		Art	3.0000
3		English Literature	2.0000
4		Music	2.0000
5		Computer Science	NULL
6		History	NULL
7		Chemistry	NULL
8		Physics	NULL
9		Biology	NULL
10		Mathematics	NULL

Result Grid Form Editor Field Types

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.

```

208 • SELECT s.student_id, s.first_name, s.last_name, p.amount AS highest_payment
209 FROM Students s
210 JOIN Payments p ON s.student_id = p.student_id
211 WHERE p.amount = (
212     SELECT MAX(amount)
213     FROM Payments
214 );
215
216

```

100% 1:215 3 errors found

Result Grid Filter Rows: Search Export:

	student_...	first_name	last_name	highest_paym...
▶ 4		Michael	Taylor	1000.00

Result Grid

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

```

215 • SELECT c.course_id, c.course_name, COUNT(e.enrollment_id) AS enrollment_count
216 FROM Courses c
217 LEFT JOIN Enrollments e ON c.course_id = e.course_id
218 GROUP BY c.course_id, c.course_name
219 HAVING COUNT(e.enrollment_id) = (
220     SELECT MAX(enrollment_count)
221     FROM (
222         SELECT course_id, COUNT(enrollment_id) AS enrollment_count
223         FROM Enrollments
224         GROUP BY course_id
225     ) AS max_enrollments
226 );

```

100% 3:226 3 errors found

Result Grid Filter Rows: Search Export:

	course_id	course_name	enrollment_co...
▶ 2	Art		3

Result Grid

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

```

228 • SELECT
229     t.teacher_id,
230     t.first_name AS teacher_first_name,
231     t.last_name AS teacher_last_name,
232     COALESCE(SUM(p.amount), 0) AS total_payments
233 FROM Teacher t
234 LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
235 LEFT JOIN Enrollments e ON c.course_id = e.course_id
236 LEFT JOIN Payments p ON e.student_id = p.student_id
237 GROUP BY t.teacher_id, t.first_name, t.last_name;

```

100% 50:237 3 errors found

Result Grid Filter Rows: Search Export:

	teacher_id	teacher_first_na...	teacher_last_na...	total_payme...
▶ 1	Jane	Smith	1600.00	
▶ 2	John	Doe	1800.00	
▶ 3	Emily	Jones	1250.00	
▶ 4	Michael	Johnson	0.00	
▶ 5	Sophia	Williams	1000.00	
▶ 6	David	Brown	0.00	
▶ 7	Olivia	Taylor	0.00	
▶ 8	Daniel	Miller	0.00	
▶ 9	Alice	Anderson	0.00	
▶ 10	Bob	Wilson	0.00	

Result Grid

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

```

239 • SELECT
240     s.student_id,
241     s.first_name,
242     s.last_name
243 FROM Students s
244 WHERE NOT EXISTS (
245     SELECT c.course_id
246     FROM Courses c
247     WHERE NOT EXISTS (
248         SELECT e.course_id
249         FROM Enrollments e
250         WHERE e.student_id = s.student_id
251               AND e.course_id = c.course_id
252     )
253 );
254

```

100% 1:254 3 errors found

Result Grid Filter Rows: Search Edit: Export/Import:

student_id	first_name	last_name
NULL	NULL	NULL

Result Grid

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

```

255 • SELECT
256     t.teacher_id,
257     t.first_name,
258     t.last_name
259 FROM Teacher t
260 WHERE NOT EXISTS (
261     SELECT 1
262     FROM Courses c
263     WHERE c.teacher_id = t.teacher_id
264 );
265

```

100% 3:264 3 errors found

Result Grid Filter Rows: Search Edit: Export/Import:

teacher_id	first_name	last_name
4	Michael	Johnson

Result Grid

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

```

266 • SELECT
267     AVG(age) AS average_age
268 FROM (
269     SELECT
270         student_id,
271         first_name,
272         last_name,
273         FLOOR(DATEDIFF(CURDATE(), date_of_birth) / 365) AS age
274     FROM Students
275 ) AS student_age;
276

```

100% 18:275 3 errors found

Result Grid Filter Rows: Search Export:

average_age
28.8182

Result Grid

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

```

277 • SELECT
278     c.course_id,
279     c.course_name
280 FROM Courses c
281 WHERE NOT EXISTS (
282     SELECT 1
283     FROM Enrollments e
284     WHERE e.course_id = c.course_id
285 );

```

100% 1:287 3 errors found

Result Grid Filter Rows: Search Edit: Export/Import:

course_id	course_name
5	Computer Science
6	History
7	Chemistry
8	Physics
9	Biology
10	Mathematics

Result Grid Form Editor

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

```

287 • SELECT
288     e.student_id,
289     s.first_name,
290     s.last_name,
291     e.course_id,
292     c.course_name,
293     COALESCE(SUM(p.amount), 0) AS total_payments
294 FROM Enrollments e
295 JOIN Students s ON e.student_id = s.student_id
296 JOIN Courses c ON e.course_id = c.course_id
297 LEFT JOIN Payments p ON e.student_id = p.student_id
298 GROUP BY e.student_id, s.first_name, s.last_name, e.course_id, c.course_name;
299
300

```

100% 78:298 3 errors found

Result Grid Filter Rows: Search Export:

student_id	first_name	last_name	course_id	course_name	total_payments
2	Sophia	Wilson	3	English Literature	500.00
3	Olivia	Anderson	2	Art	850.00
4	Michael	Taylor	1	Physical Education	1000.00
5	Emily	Davis	4	Music	500.00
7	Alice	Williams	2	Art	200.00
8	John	Doe	1	Physical Education	600.00
9	Jane	Smith	4	Music	500.00
10	Bob	Johnson	3	English Literature	750.00
1	Daniel	Brown	2	Art	750.00

Result Grid Form Editor

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.

```

301 • SELECT
302     p.student_id,
303     s.first_name,
304     s.last_name,
305     COUNT(p.payment_id) AS payment_count
306 FROM Payments p
307 JOIN Students s ON p.student_id = s.student_id
308 GROUP BY p.student_id, s.first_name, s.last_name
309 HAVING COUNT(p.payment_id) > 1;
310

```

100% 1:310 3 errors found

Result Grid Filter Rows: Search Export:

student_id	first_name	last_name	payment_count
------------	------------	-----------	---------------

Result Grid

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.

```
312 • SELECT
313     s.student_id,
314     s.first_name,
315     s.last_name,
316     COALESCE(SUM(p.amount), 0) AS total_payments
317 FROM Students s
318 LEFT JOIN Payments p ON s.student_id = p.student_id
319 GROUP BY s.student_id;
320
```

100% 1:321 3 errors found

Result Grid Filter Rows: Search Export:

	student_id	first_name	last_name	total_payments
1	Daniel	Brown	750.00	
2	Sophia	Wilson	500.00	
3	Olivia	Anderson	850.00	
4	Michael	Taylor	1000.00	
5	Emily	Davis	500.00	
6	David	Miller	900.00	
7	Alice	Williams	200.00	
8	John	Doe	600.00	
9	Jane	Smith	500.00	
10	Bob	Johnson	750.00	
11	John	Doe	0.00	

Result Grid Form Editor Field Types

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.

```
322 • SELECT
323     c.course_id,
324     c.course_name,
325     COUNT(e.enrollment_id) AS enrolled_students_count
326 FROM Courses c
327 LEFT JOIN Enrollments e ON c.course_id = e.course_id
328 GROUP BY c.course_id, c.course_name;
329
```

100% 1:329 3 errors found

Result Grid Filter Rows: Search Export:

	course_id	course_name	enrolled_students_count
1	Physical Education	2	
2	Art	3	
3	English Literature	2	
4	Music	2	
5	Computer Science	0	
6	History	0	
7	Chemistry	0	
8	Physics	0	
9	Biology	0	
10	Mathematics	0	

Result Grid Form Editor Field Types

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.

```
330 • SELECT
331     s.student_id,
332     s.first_name,
333     s.last_name,
334     AVG(p.amount) AS average_payment_amount
335 FROM Students s
336 LEFT JOIN Payments p ON s.student_id = p.student_id
337 GROUP BY s.student_id, s.first_name, s.last_name;
338
```

100% 1:338 3 errors found

Result Grid Filter Rows: Search Export:

	student_id	first_name	last_name	average_payment_amount
▶ 1	Daniel	Brown	750.000000	
2	Sophia	Wilson	500.000000	
3	Olivia	Anderson	850.000000	
4	Michael	Taylor	1000.000000	
5	Emily	Davis	500.000000	
6	David	Miller	900.000000	
7	Alice	Williams	200.000000	
8	John	Doe	600.000000	
9	Jane	Smith	500.000000	
10	Bob	Johnson	750.000000	
11	John	Doe	NULL	

Result Grid Form Editor Field Types