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.

```sql
create database if not exists SISDB;
use SISDB;

create table if not exists Students (
student_id int PRIMARY KEY,
first_name varchar(50),
last_name varchar(50),
date_of_birth date,
email varchar(50),
phone_no varchar(12)
);
create table if not exists Teacher (
    teacher_id int PRIMARY KEY,
    first_name varchar(50),
    last_name varchar(50),
    email varchar(25)
);

create table if not exists Courses (
course_id int PRIMARY KEY,
course_name varchar(100),
credits int,
teacher_id int,FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);
```

```sql
create table if not exists Enrollments (
    enrollment_id int PRIMARY KEY,
    student_id int,FOREIGN KEY (student_id) REFERENCES Students(student_id),
    course_id int,FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    enrollment_date DATE
);
create table if not exists Payments (
    payment_id INT PRIMARY KEY,
    student_id INT,FOREIGN KEY (student_id) REFERENCES Students(student_id),
    amount DECIMAL(10, 2),
    payment_date DATE
);
```

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

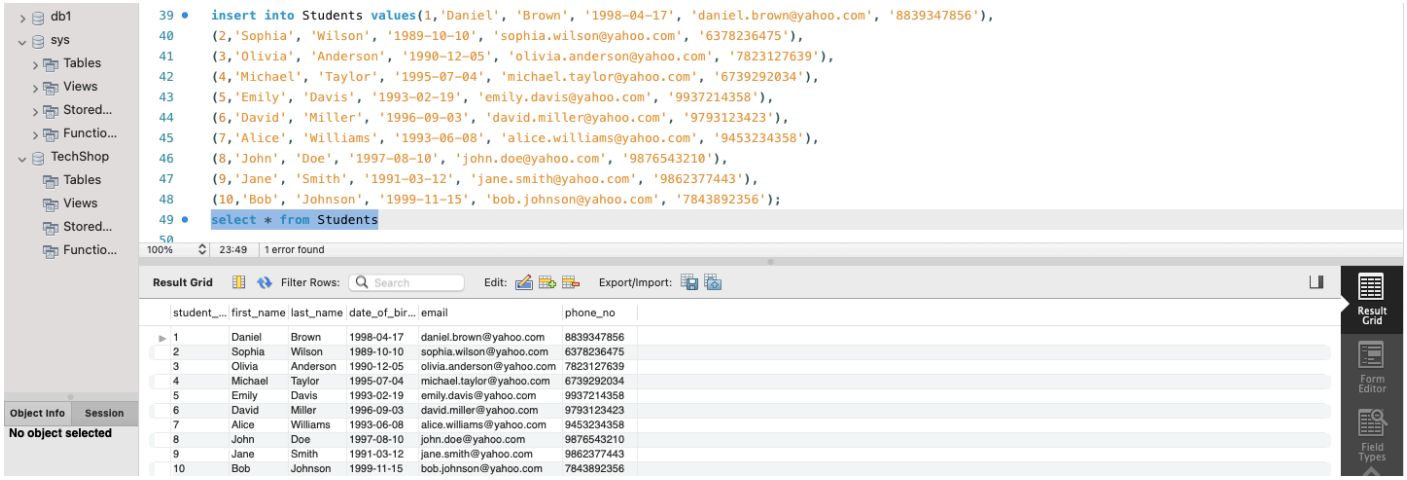
i. Students

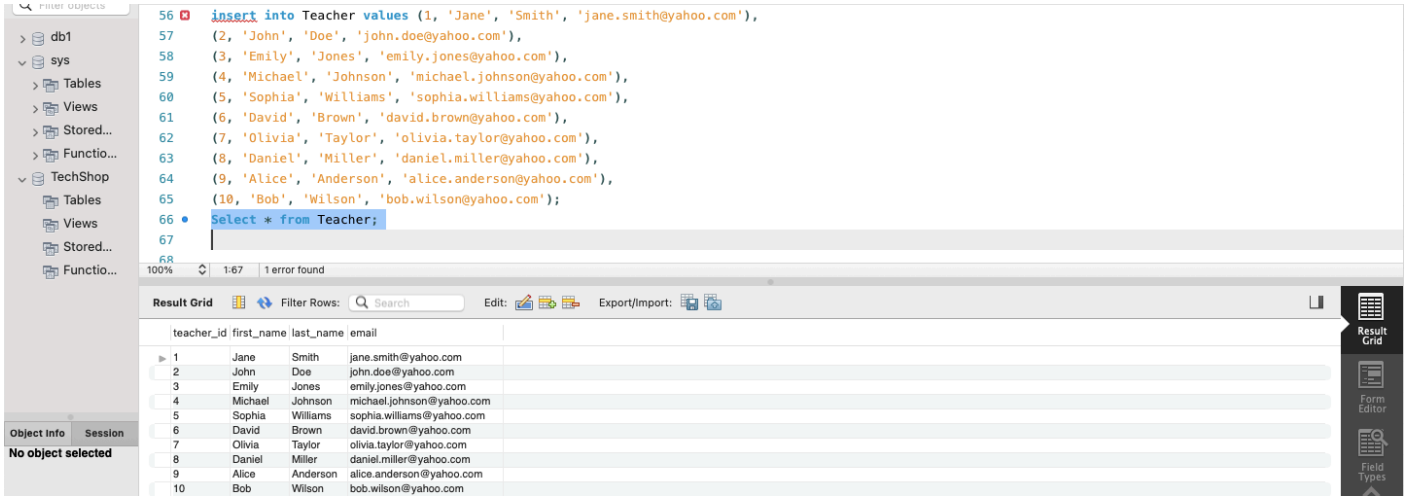
ii. Courses

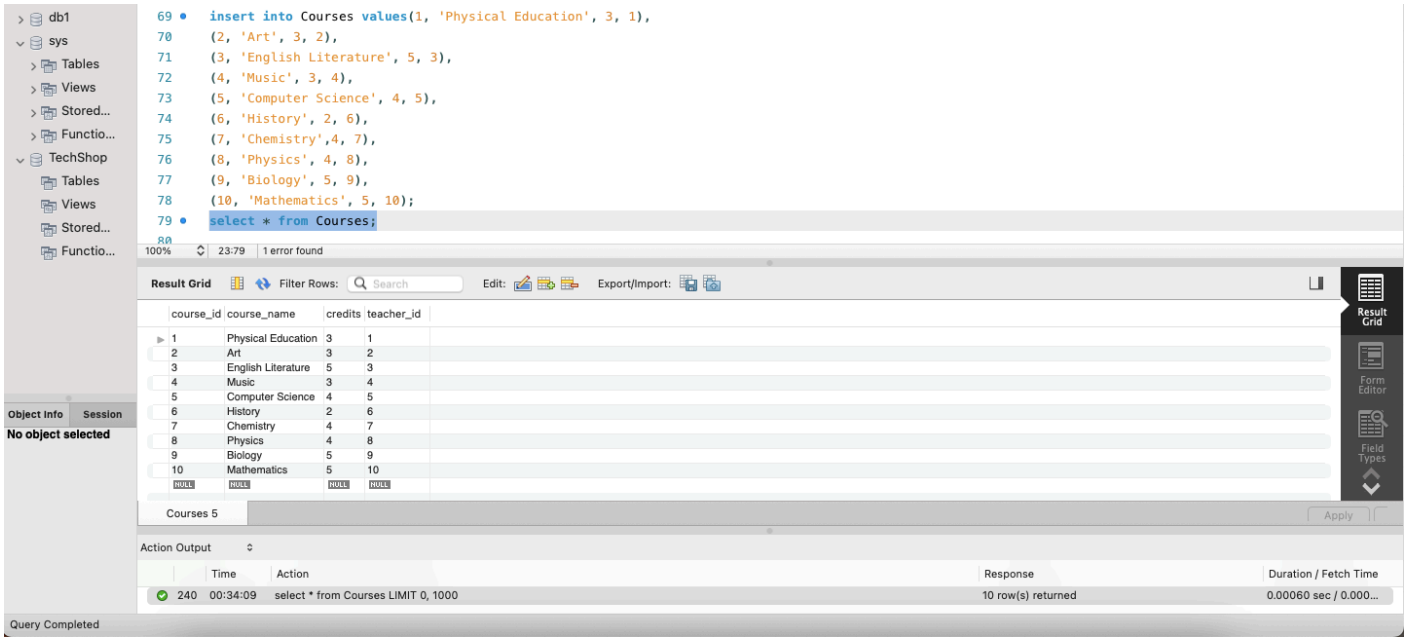
iii. Enrollments

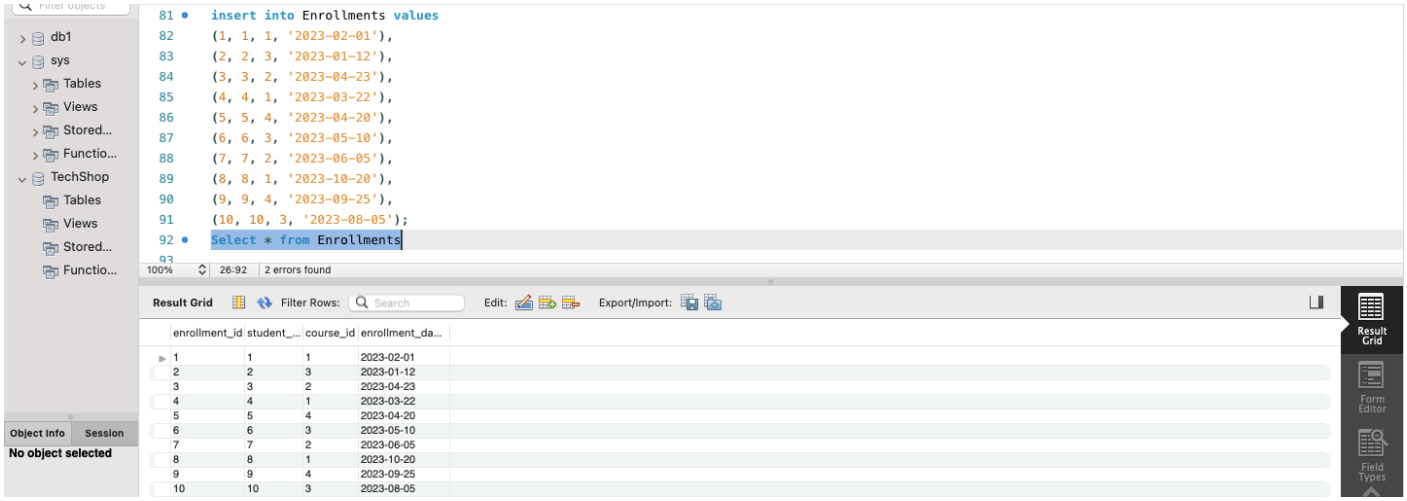
iv. Teacher

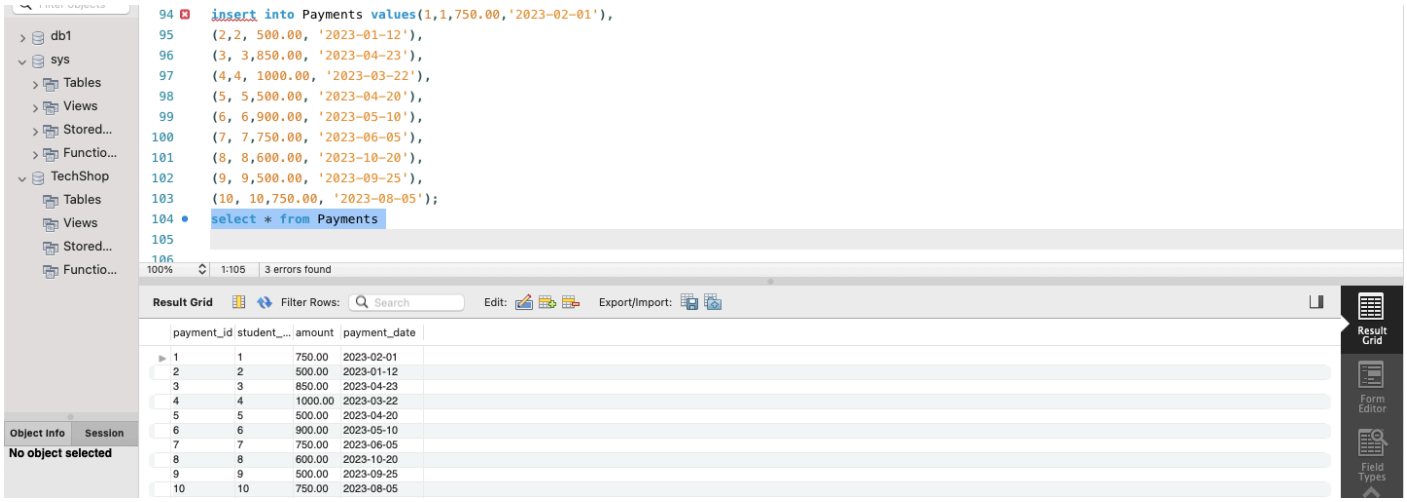
v. Payments



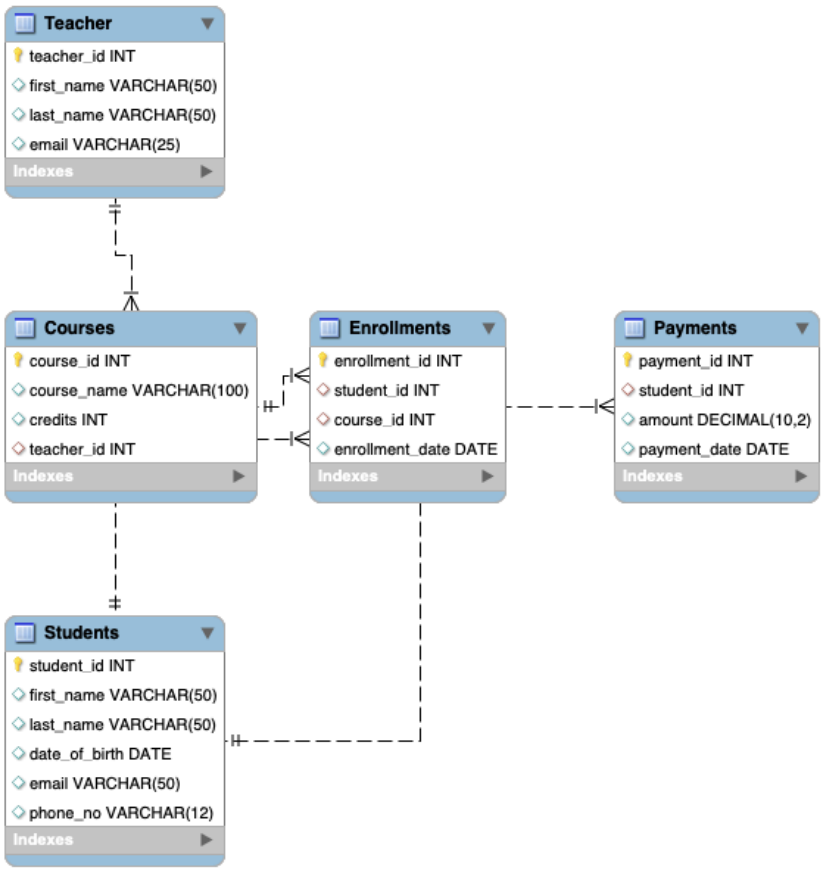



db1
sys
Tables
Views
Stored...
Functio...
TechShop
Tables
Views
Stored...
Functio...
Object Info
Session
No object selected
69 insert into Courses values(1, 'Physical Education', 3, 1),
70 (2, 'Art', 3, 2),
71 (3, 'English Literature', 5, 3),
72 (4, 'Music', 3, 4),
73 (5, 'Computer Science', 4, 5),
74 (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 Physical Education 3 1
2 Art 3 2
3 English Literature 5 3
4 Music 3 4
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
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...
Functio...
TechShop
Tables
Views
Stored...
Functio...
Object Info
Session
No object selected
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_... course_id enrollment_da...
1 1 1 2023-02-01
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


db1
sys
Tables
Views
Stored...
Functio...
TechShop
Tables
Views
Stored...
Functio...
Object Info
Session
No object selected
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_... 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 7 750.00 2023-06-05
8 8 600.00 2023-10-20
9 9 500.00 2023-09-25
10 10 750.00 2023-08-05

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:

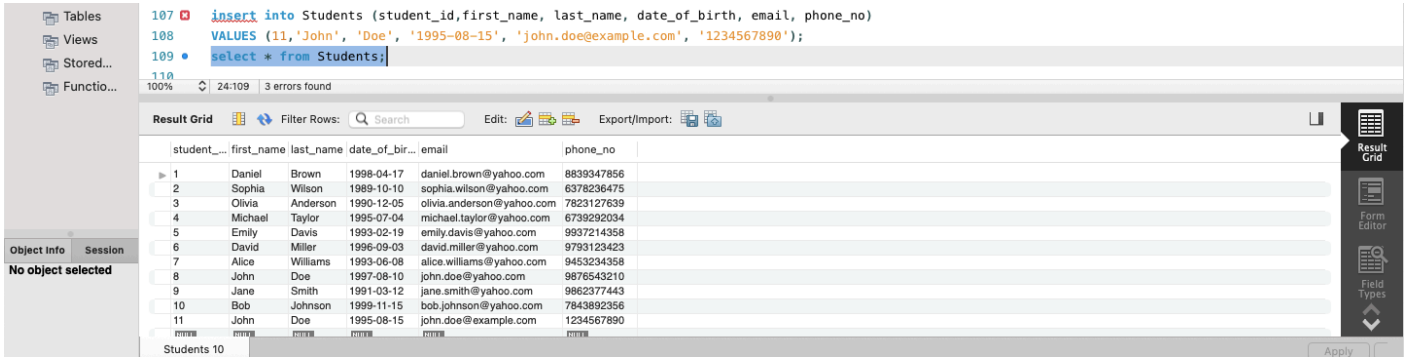
a. First Name: John

b. Last Name: Doe

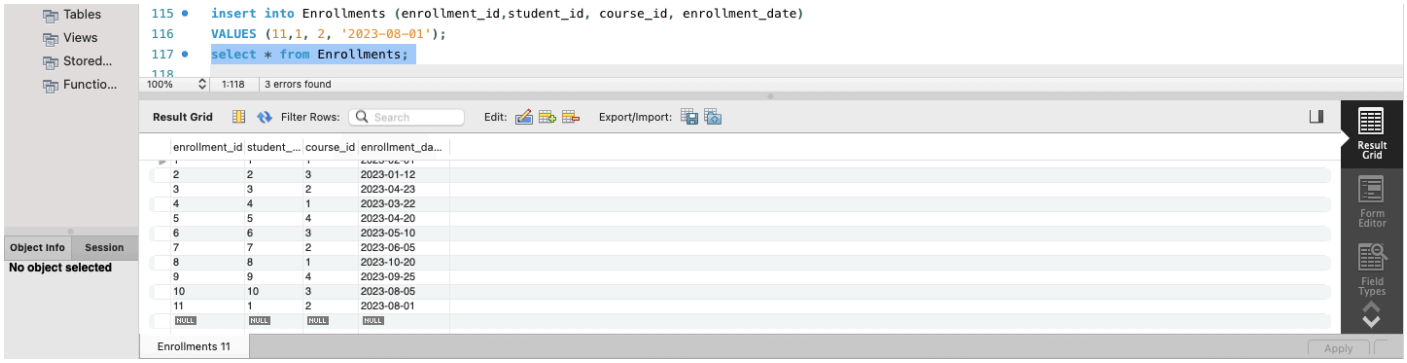
c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

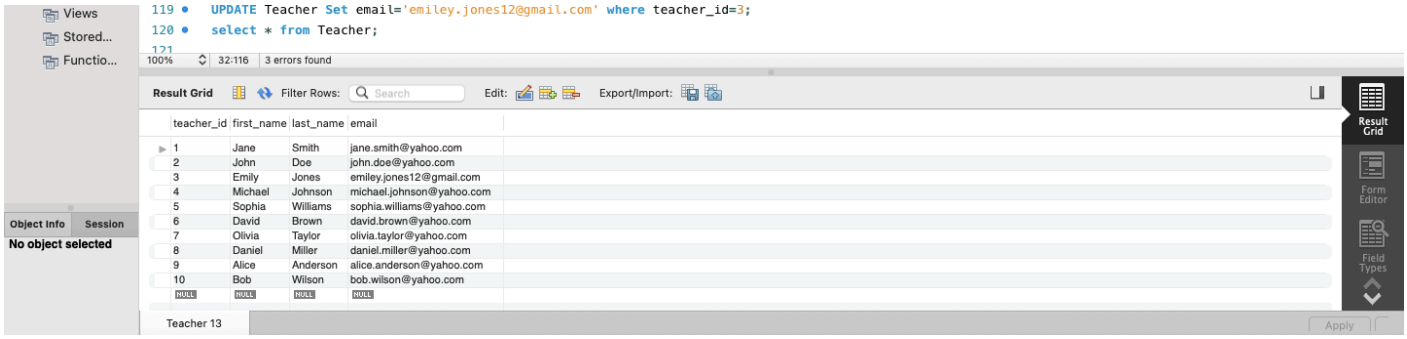
e. Phone Number: 1234567890



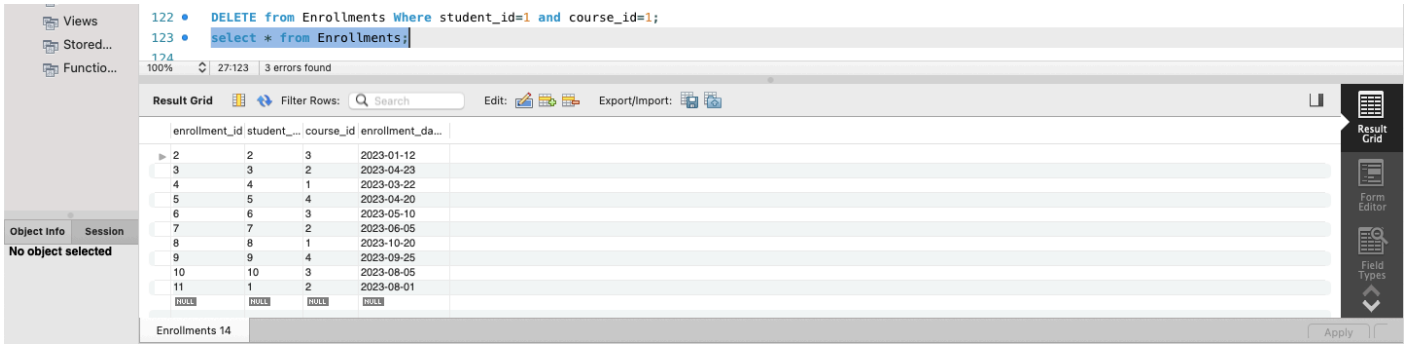
2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.



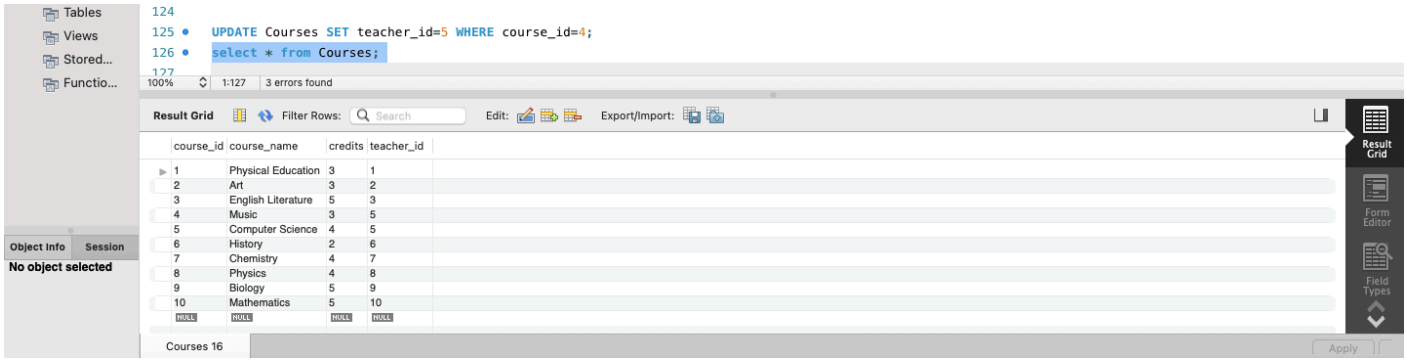
3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.



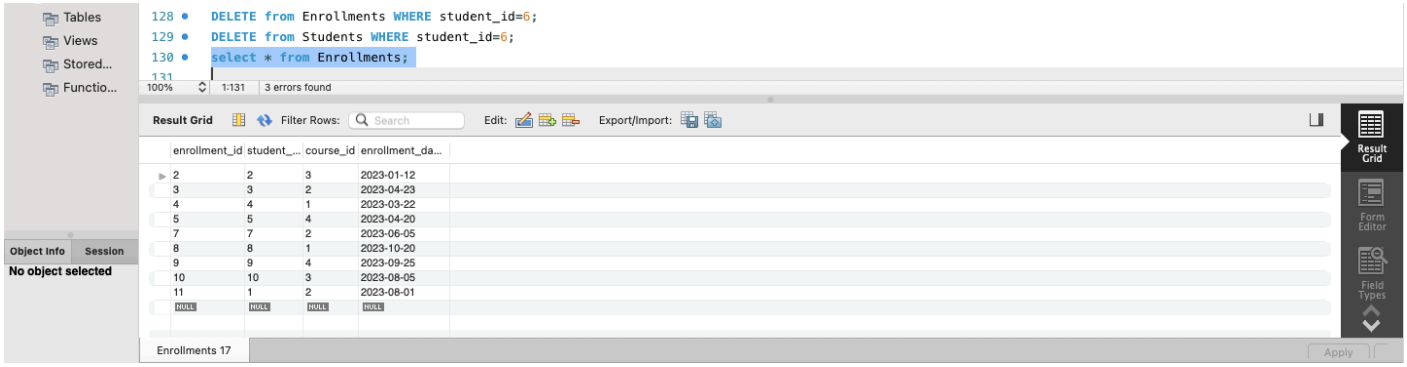
4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.



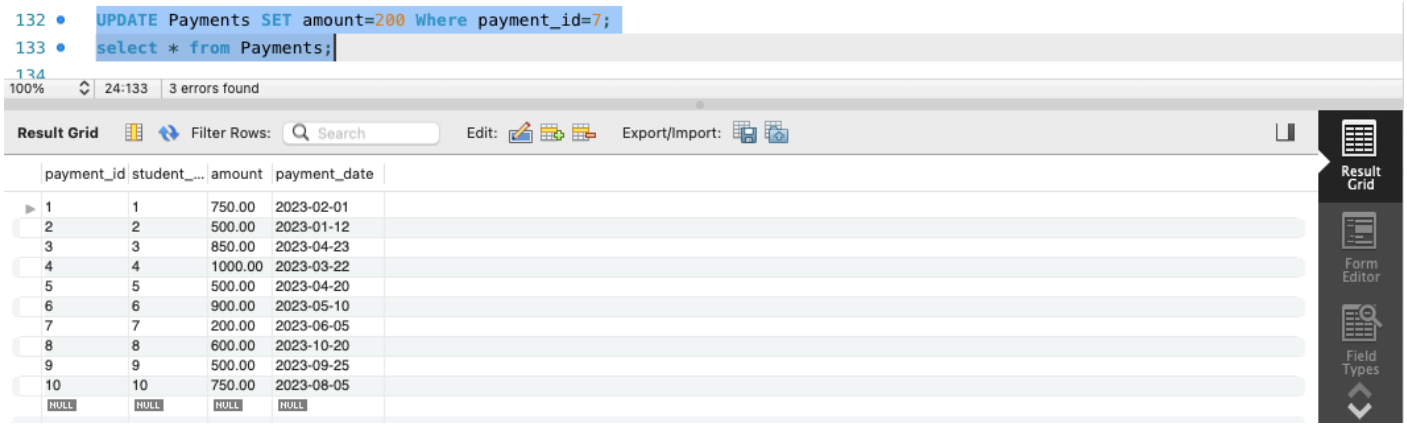
5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.



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.



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



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;
188
```

100% 24:187 3 errors found

Result Grid Filter Rows: Search Export:

student_...	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

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

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

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

```
SELECT c.course_id, c.course_name, COUNT(e.enrollment_id) AS enrollment_count
FROM Courses c
LEFT 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 course_id, COUNT(enrollment_id) AS enrollment_count
        FROM Enrollments
        GROUP BY course_id
    ) AS max_enrollments
);
```

100% 3:226 3 errors found

Result Grid Filter Rows: Search Export:

course_id	course_name	enrollment_co...
2	Art	3

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

```
SELECT
    t.teacher_id,
    t.first_name AS teacher_first_name,
    t.last_name AS teacher_last_name,
    COALESCE(SUM(p.amount), 0) AS total_payments
FROM Teacher t
LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
LEFT JOIN Enrollments e ON c.course_id = e.course_id
LEFT JOIN Payments p ON e.student_id = p.student_id
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

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
    s.student_id,
    s.first_name,
    s.last_name
FROM Students s
WHERE NOT EXISTS (
    SELECT c.course_id
    FROM Courses c
    WHERE NOT EXISTS (
        SELECT e.course_id
        FROM Enrollments e
        WHERE e.student_id = s.student_id
        AND e.course_id = c.course_id
    )
);
```

100% 1:254 3 errors found

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

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

100% 3:264 3 errors found

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

teacher_id	first_name	last_name
4	Michael	Johnson

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(age) AS average_age
FROM (
    SELECT
        student_id,
        first_name,
        last_name,
        FLOOR(DATEDIFF(CURDATE(), date_of_birth) / 365) AS age
    FROM Students
) AS student_age;
```

100% 18:275 3 errors found

Result Grid Filter Rows: Search Export:

average_age
28.8182

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

```sql
SELECT
    c.course_id,
    c.course_name
FROM Courses c
WHERE NOT EXISTS (
    SELECT 1
    FROM Enrollments e
    WHERE e.course_id = c.course_id
);
```

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

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

100% 78:298 3 errors found

Result Grid | Filter Rows: Search | Export:

student_...	first_name	last_name	course_id	course_name	total_payme...
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

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
    p.student_id,
    s.first_name,
    s.last_name,
    COUNT(p.payment_id) AS payment_count
FROM Payments p
JOIN Students s ON p.student_id = s.student_id
GROUP BY p.student_id, s.first_name, s.last_name
HAVING COUNT(p.payment_id) > 1;
```

100% 1:310 3 errors found

Result Grid | Filter Rows: Search | Export:

student_...	first_name	last_name	payment_cou...

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;
```

100% 1:321 3 errors found

Result Grid Filter Rows: Search Export:

student_...	first_name	last_name	total_payme...
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

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

100% 1:329 3 errors found

Result Grid Filter Rows: Search Export:

course_id	course_name	enrolled_students_co...
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

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_...	first_name	last_name	average_payment_amo...
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