

## **TASK 1: Database Design:**

## **Q1. Create the database named "SISDB"**

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation panels for MANAGEMENT (Server Status, Client Connection, Users and Privileges, Status and System, Data Export, Data Import/Res), INSTANCE (Startup / Shutdown, Server Logs, Options File), and PERFORMANCE (Dashboard, Performance Rep, Performance Sch). The main area has tabs for Navigator, Query 1 (selected), and SQLAdditions. The Query 1 tab displays the following SQL code:

```
267
268 • select avg(TotalAmount) from Orders;
269
270 • use codingchallenges;
271
272
273
274 • create database SISDB;
275
276 • create table Students(
277     student_id int primary key,
278     first_name text,
279     last_name text,
280     d_o_b date,
281     email text,
282     phone bigint
283 );
284
```

The code at line 274 is highlighted. Below the code, the Output pane shows the execution results:

Action	Output			
Action Output				
#	Time	Action	Message	Duration / Fetch
1	14:34:07	create database SISDB	1 row(s) affected	0.000 sec

The bottom status bar shows system icons and the date/time: 12/10/2023 2:34 PM.

**Q2. 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. Teachers
  - e. Payments

The screenshot shows the MySQL Workbench interface with the following details:

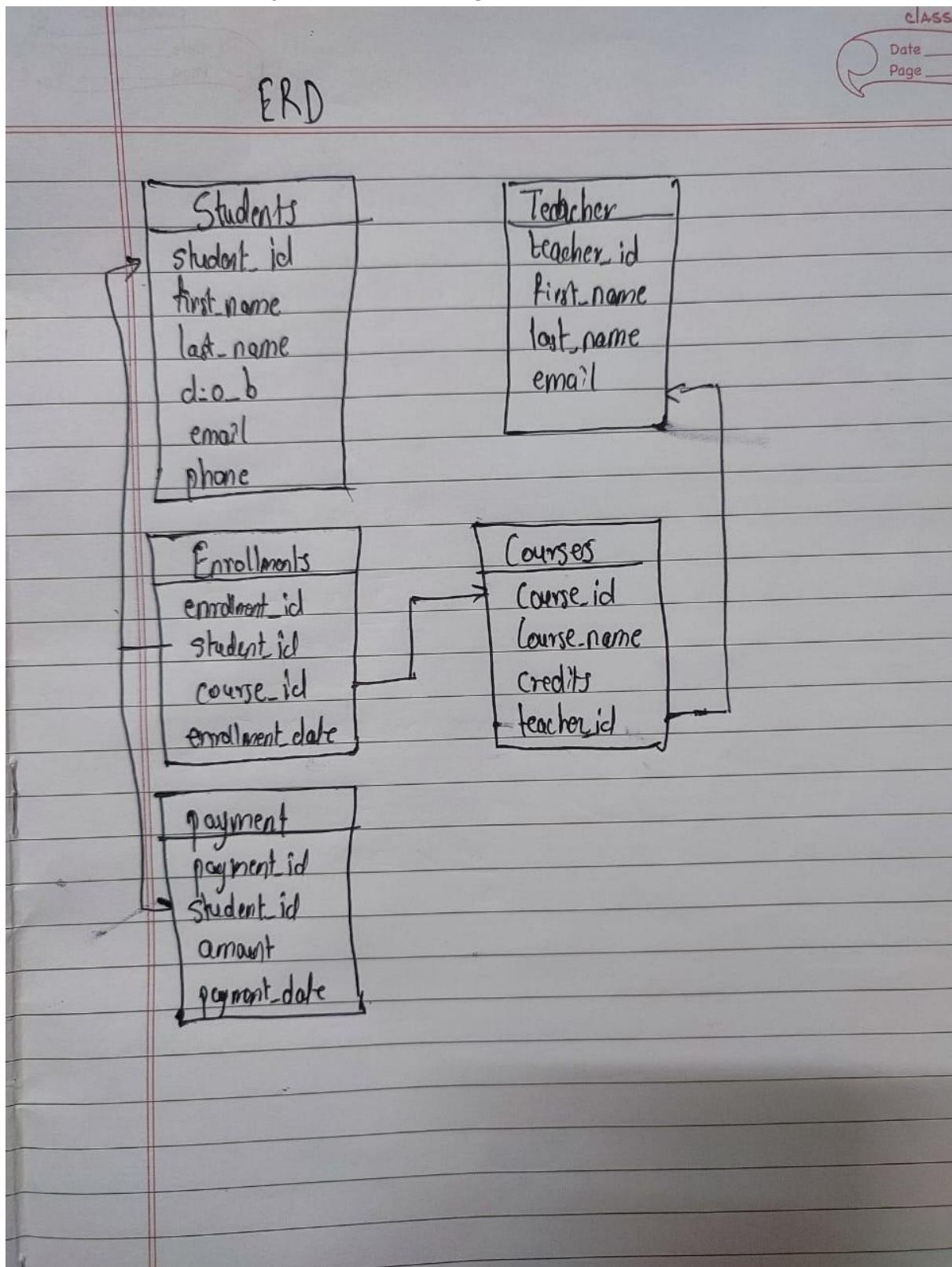
- File** | Local instance MySQL80 | **Edit** | **View** | **Query** | **Database** | **Server** | **Tools** | **Scripting** | **Help**
- Navigator**: MANAGEMENT (Server Status, Client Connectivity, Users and Privileges, Status and System, Data Export, Data Import/Resync), INSTANCE (Startup / Shutdown, Server Logs, Options File), PERFORMANCE (Dashboard, Performance Report, Performance Schema).
- SQL Editor (Query 1)**: Displays the SQL code for creating three tables:

```
enrollment_id int primary key,
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,
student_id int,
amount double,
payment_date date,
foreign key (student_id) references Students(student_id)
);
```
- Information**: Action Output table showing the results of the create statements:

#	Time	Action	Message	Duration / Fetch
4	14:35:36	create table Students(student_id int primary key, first_name text, last_name text, d_o_b date, email text, ph...)	0 row(s) affected	0.015 sec
5	14:35:36	create table Teacher(teacher_id int primary key, first_name text, last_name text, email text)	0 row(s) affected	0.016 sec
6	14:35:36	create table Courses(course_id int primary key, course_name text, credits int, teacher_id int, foreign key (teach...	0 row(s) affected	0.031 sec
7	14:35:36	create table Enrolments(enrollment_id int primary key, student_id int, course_id int, enrollment_date date, fore...	0 row(s) affected	0.031 sec
8	14:35:36	create table Payments(payment_id int primary key, student_id int, amount double, payment_date date, foreign...	0 row(s) affected	0.016 sec
- Object Info**: Shows the selected object is "No object selected".
- SQLAdditions**: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Q3. Create an ERD (Entity Relationship Diagram) for the database.



**Q4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** MANAGEMENT (Server Status, Client Connectivity, Users and Privileges, Status and System, Data Export, Data Import/Resync), INSTANCE (Startup / Shutdown, Server Logs, Options File), PERFORMANCE (Dashboard, Performance Report, Performance Schema).
- Query Editor:** Query 1 (SQL) containing the following code:

```
enrollment_id int primary key,
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,
student_id int,
amount double,
payment_date date,
foreign key (student_id) references Students(student_id)
);
```
- Output:** Action Output table showing the execution results of the queries:

#	Time	Action	Message	Duration / Fetch
4	14:35:36	create table Students(student_id int primary key, first_name text, last_name text, d_o_b date, email text, ph...)	0 row(s) affected	0.015 sec
5	14:35:36	create table Teacher(teacher_id int primary key, first_name text, last_name text, email text)	0 rows affected	0.015 sec
6	14:35:36	create table Courses(course_id int primary key, course_name text, credits int, teacher_id int, foreign key (teac...	0 row(s) affected	0.031 sec
7	14:35:36	create table Enrolments(enrollment_id int primary key, student_id int, course_id int, enrollment_date date, fore...	0 row(s) affected	0.031 sec
8	14:35:36	create table Payments(payment_id int primary key, student_id int, amount double, payment_date date, foreign...	0 row(s) affected	0.016 sec
- Bottom Bar:** ENG US, 2:35 PM, 12/10/2023.

**Q5. Insert at least 10 sample records into each of the following tables.**

## i. Students    ii. Courses    iii. Enrollments    iv. Teacher    v. Payments

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

Q1. 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](mailto:john.doe@example.com) e. Phone Number: 1234567890

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Shows the SQL command:

```
INSERT INTO Students (student_id, first_name, last_name, d_o_b, email, phone)
VALUES (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);
```
- Output Window:** Displays the execution message:

```
1 15:10:56 INSERT INTO Students (student_id, first_name, last_name, d_o_b, email, phone) VALUES (11, 'John', 'Doe', '1995-08-15... 1 row(s) affected
```
- System Tray:** Shows the Windows taskbar with various application icons.

Q2. 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 the MySQL Workbench interface with the following details:

- Query Editor:** Shows two SQL commands:

```
INSERT INTO Students (student_id, first_name, last_name, d_o_b, email, phone)
VALUES (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);

INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES (11, 1, 1, '2023-06-27');
```
- Output Window:** Displays the execution messages:

```
1 15:17:49 INSERT INTO Students (student_id, first_name, last_name, d_o_b, email, phone) VALUES (11, 'John', 'Doe', '1995-08-15... 1 row(s) affected
```

```
1 15:17:49 INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES (11, 1, 1, '2023-06-27') 1 row(s) affected
```
- System Tray:** Shows the Windows taskbar with various application icons.

**Q3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.**

The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1'. The code entered is:

```
384 • INSERT INTO Students (student_id, first_name, last_name, d_o_b, email, phone)
VALUES
386 (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);
387
388 • INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
390 (11, 1, 1, '2023-06-27');
391
392 • UPDATE Teacher
393 SET email = 'mrbrown23@example.com'
394 WHERE teacher_id = 4;
395
```

The output pane shows the execution results:

Action Output	Time	Action	Message	Duration / Fetch
	1 15:20:11	UPDATE Teacher SET email = 'mrbrown23@example.com' WHERE teacher_id = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.047 sec

**Q4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.**

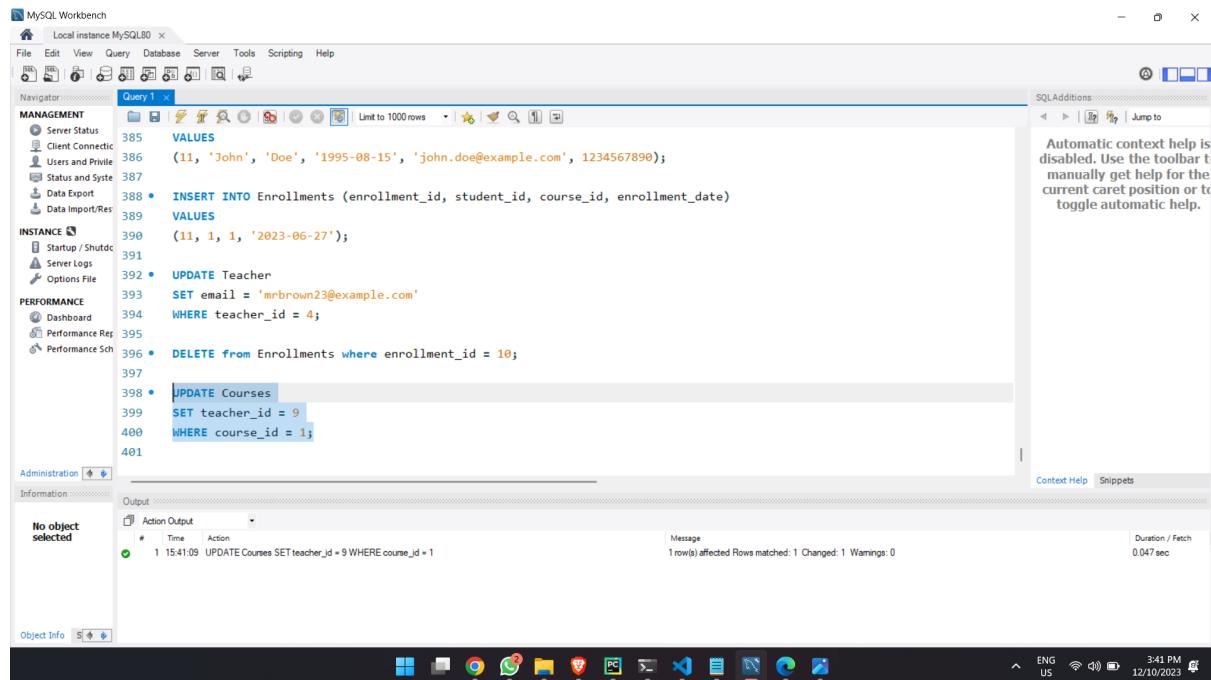
The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1'. The code entered is:

```
382
383
384 • INSERT INTO Students (student_id, first_name, last_name, d_o_b, email, phone)
VALUES
386 (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);
387
388 • INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
390 (11, 1, 1, '2023-06-27');
391
392 • UPDATE Teacher
393 SET email = 'mrbrown23@example.com'
394 WHERE teacher_id = 4;
395
396 • DELETE from Enrollments where enrollment_id = 10;
397
398
```

The output pane shows the execution results:

Action Output	Time	Action	Message	Duration / Fetch
	1 15:24:18	DELETE from Enrollment where enrollment_id = 10	Error Code: 1146. Table 'sdb.enrollment' doesn't exist	0.000 sec
	2 15:24:43	DELETE from Enrollments where enrollment_id = 10	1 row(s) affected	0.047 sec

**Q5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.**



The screenshot shows the MySQL Workbench interface with the following details:

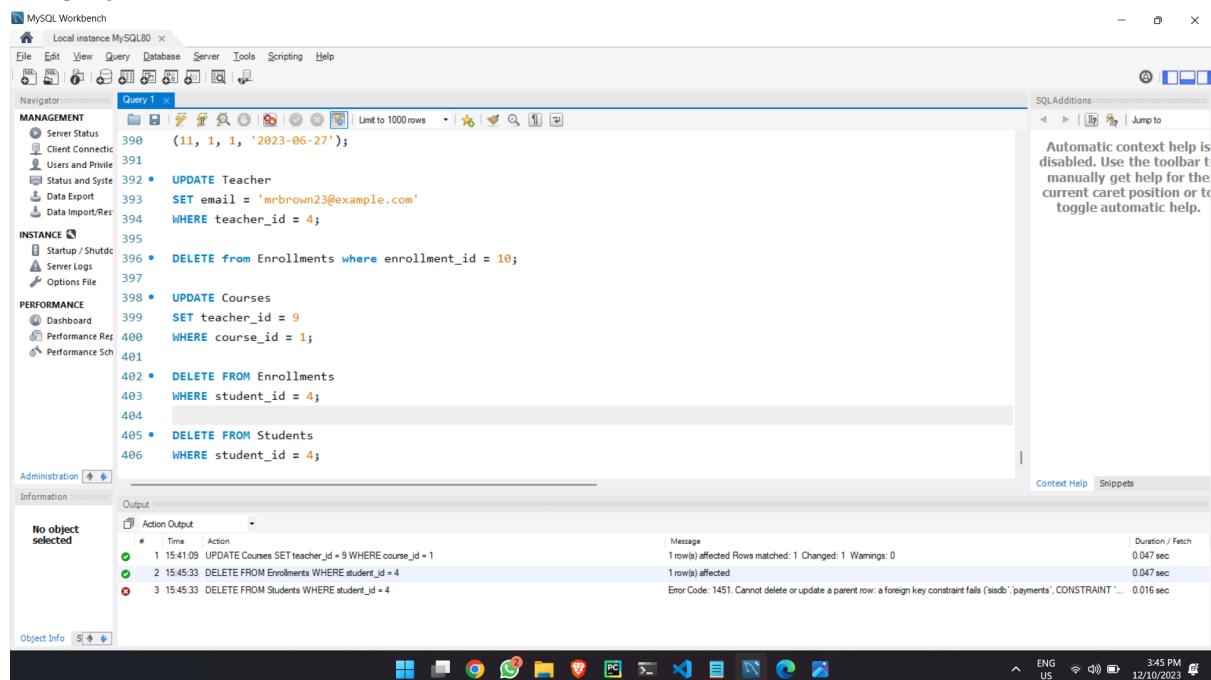
- Navigator:** Shows the database structure with sections like MANAGEMENT, INSTANCE, and PERFORMANCE.
- Query Editor (Query 1):** Contains the following SQL code:
 

```

385     VALUES
386     (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);
387
388 • INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
389     VALUES
390     (11, 1, 1, '2023-06-27');
392 • UPDATE Teacher
393     SET email = 'mbrown23@example.com'
394     WHERE teacher_id = 4;
396 • DELETE FROM Enrollments WHERE enrollment_id = 10;
397
398 • UPDATE Courses
399     SET teacher_id = 9
400     WHERE course_id = 1;
401
      
```
- Output:** Displays the results of the UPDATE query:
 

Action Output	Time	Action	Message	Duration / Fetch
1	15:41:09	UPDATE Courses SET teacher_id = 9 WHERE course_id = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.047 sec
- System Bar:** Shows the taskbar with various application icons and system status indicators.

**Q6. 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 the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure with sections like MANAGEMENT, INSTANCE, and PERFORMANCE.
- Query Editor (Query 1):** Contains the following SQL code:
 

```

390     (11, 1, 1, '2023-06-27');
391
392 • UPDATE Teacher
393     SET email = 'mbrown23@example.com'
394     WHERE teacher_id = 4;
395
396 • DELETE FROM Enrollments WHERE enrollment_id = 10;
397
398 • UPDATE Courses
399     SET teacher_id = 9
400     WHERE course_id = 1;
401
402 • DELETE FROM Enrollments
403     WHERE student_id = 4;
404
405 • DELETE FROM Students
406     WHERE student_id = 4;
      
```
- Output:** Displays the results of the queries, including a foreign key constraint error for the last delete:
 

Action Output	Time	Action	Message	Duration / Fetch
1	15:41:09	UPDATE Courses SET teacher_id = 9 WHERE course_id = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.047 sec
2	15:45:33	DELETE FROM Enrollments WHERE student_id = 4	1 row(s) affected	0.047 sec
3	15:45:33	DELETE FROM Students WHERE student_id = 4	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('isdb'.'payments', CONSTRAINT ...)	0.016 sec
- System Bar:** Shows the taskbar with various application icons and system status indicators.

**Q7. 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 the MySQL Workbench interface. In the Query Editor (Query 1), a series of SQL statements are run. The relevant statement is:

```
408 • UPDATE Payments  
409 SET amount = amount + 80.00  
410 WHERE student_id = 9;
```

The output window shows the execution details:

#	Time	Action
1	15:54:23	UPDATE Payments SET amount = amount + 80.00 WHERE student_id = 9

Message: 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 Duration / Fetch: 0.000 sec

The status bar at the bottom right indicates: ENG US 3:54 PM 12/10/2023

### Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

**Q1. 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.

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:

```
459
460 • use SISDB;
461
462 • SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS TotalPayments
463   FROM Students s
464   JOIN Payments p ON s.student_id = p.student_id
465   WHERE s.student_id = 5
466   GROUP BY s.student_id, s.first_name, s.last_name;
```
- Result Grid:** Displays the result of the query:

student_id	first_name	last_name	TotalPayments
5	Michael	Brown	550
- Output Window:** Shows the execution log:

Action	Time	Message	Duration / Fetch
1. 09:13:33 use SISDB		0 row(s) affected	0.015 sec
2. 09:13:38 SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS TotalPayments FROM Students s JOIN Payments p ... 1 row(s) returned			0.016 sec / 0.000 sec

**Q2. 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.**

The screenshot shows the MySQL Workbench interface with the following details:

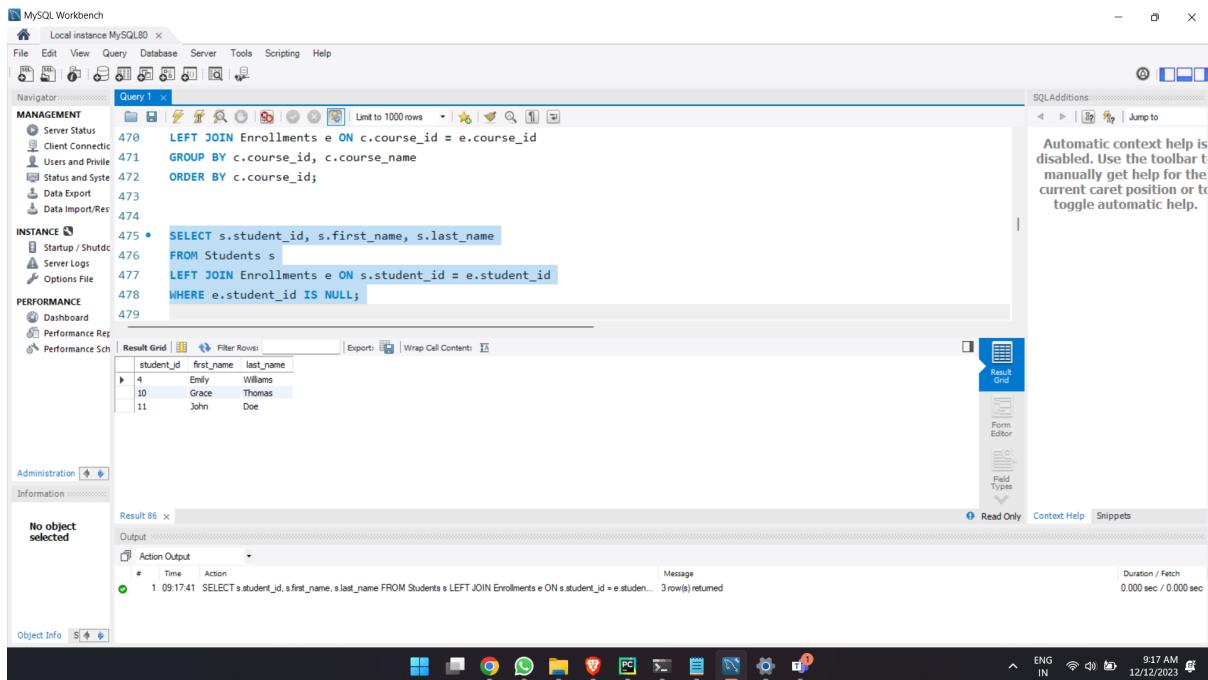
- Query Editor:** Contains the following SQL code:

```
465 WHERE s.student_id = 5
466 GROUP BY s.student_id, s.first_name, s.last_name;
467
468 • SELECT c.course_id, c.course_name, COUNT(e.student_id) AS StudentCount
469   FROM Courses c
470   LEFT JOIN Enrollments e ON c.course_id = e.course_id
471   GROUP BY c.course_id, c.course_name
472   ORDER BY c.course_id;
```
- Result Grid:** Displays the result of the query:

course_id	course_name	StudentCount
1	Computer Science 101	2
2	Mathematics 202	1
3	History 101	1
4	Physics 301	0
5	English 201	1
6	Chemistry 201	1
7	Economics 101	1
8	Biology 301	1
9	Psychology 201	1
10	Art 101	0
- Output Window:** Shows the execution log:

Action	Time	Message	Duration / Fetch
1. 09:13:33 use SISDB		0 row(s) affected	0.015 sec
2. 09:13:38 SELECT c.course_id, c.course_name, COUNT(e.student_id) AS StudentCount FROM Courses c LEFT JOIN Enrollments ... 10 row(s) returned			0.016 sec / 0.000 sec
3. 09:16:10 SELECT c.course_id, c.course_name, COUNT(e.student_id) AS StudentCount FROM Courses c LEFT JOIN Enrollments ... 10 row(s) returned			0.016 sec / 0.000 sec

**Q3. 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.**



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80 x
Navigator: Query 1 x
MANAGEMENT
    Server Status
    Client Connect
    Users and Privileges
    Status and System
    Data Export
    Data Import/Reimport
INSTANCE
    Startup / Shutdown
    Server Logs
    Options File
PERFORMANCE
    Dashboard
    Performance Schema
Administration
Information
No object selected
Result 86 x
Output
Action Output
# Time Action
1 09:17:41 SELECT s.student_id, s.first_name, s.last_name FROM Students s LEFT JOIN Enrollments e ON s.student_id = e.student_id WHERE e.student_id IS NULL;
Duration / Fetch
0.000 sec / 0.000 sec
Object Info S
9:17 AM 12/12/2023

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1'. The query is:

```

470 LEFT JOIN Enrollments e ON c.course_id = e.course_id
471 GROUP BY c.course_id, c.course_name
472 ORDER BY c.course_id;
473
474
475 • SELECT s.student_id, s.first_name, s.last_name
476 FROM Students s
477 LEFT JOIN Enrollments e ON s.student_id = e.student_id
478 WHERE e.student_id IS NULL;
479

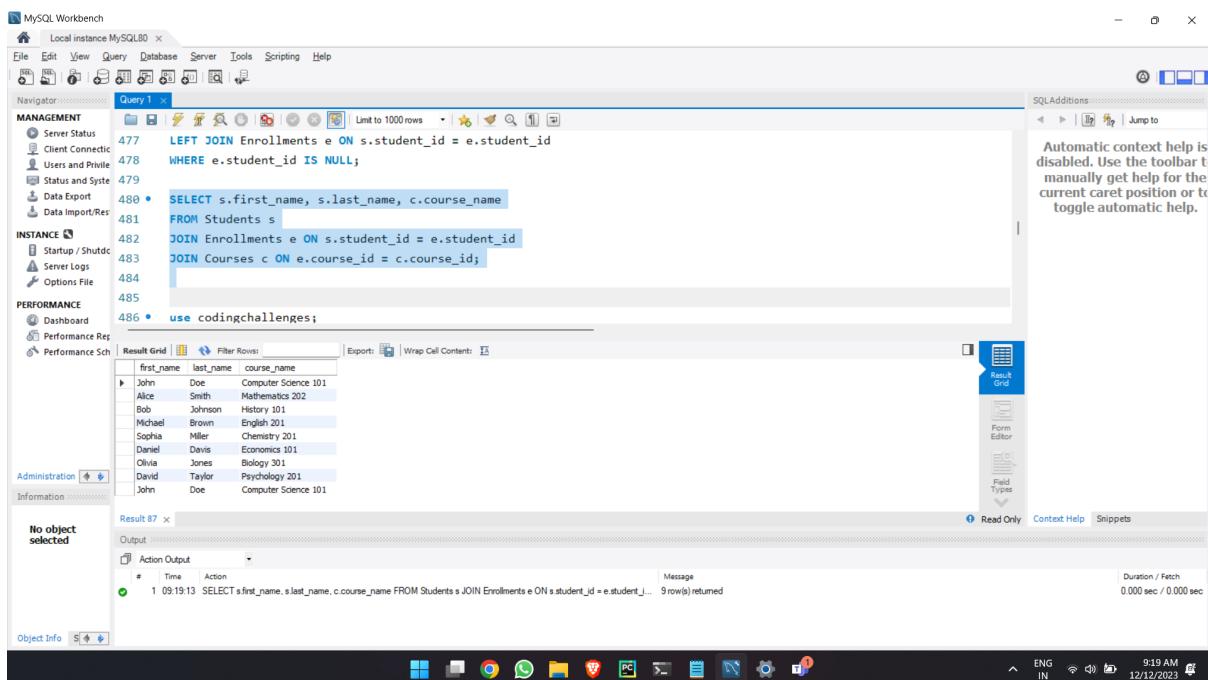
```

The result grid displays three rows of student data:

student_id	first_name	last_name
4	Emily	Williams
10	Grace	Thomas
11	John	Doe

The status bar at the bottom right shows the time as 9:17 AM and the date as 12/12/2023.

**Q4. 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.**



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80 x
Navigator: Query 1 x
MANAGEMENT
    Server Status
    Client Connect
    Users and Privileges
    Status and System
    Data Export
    Data Import/Reimport
INSTANCE
    Startup / Shutdown
    Server Logs
    Options File
PERFORMANCE
    Dashboard
    Performance Schema
Administration
Information
No object selected
Result 87 x
Output
Action Output
# Time Action
1 09:19:13 SELECT s.first_name, s.last_name, c.course_name FROM Students s JOIN Enrollments e ON s.student_id = e.student_id JOIN Courses c ON e.course_id = c.course_id;
Duration / Fetch
0.000 sec / 0.000 sec
Object Info S
9:19 AM 12/12/2023

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1'. The query is:

```

477 LEFT JOIN Enrollments e ON s.student_id = e.student_id;
479 WHERE e.student_id IS NULL;
480 • SELECT s.first_name, s.last_name, c.course_name
481 FROM Students s
482 JOIN Enrollments e ON s.student_id = e.student_id;
483 JOIN Courses c ON e.course_id = c.course_id;
484
485
486 • use codingchallenges;
487

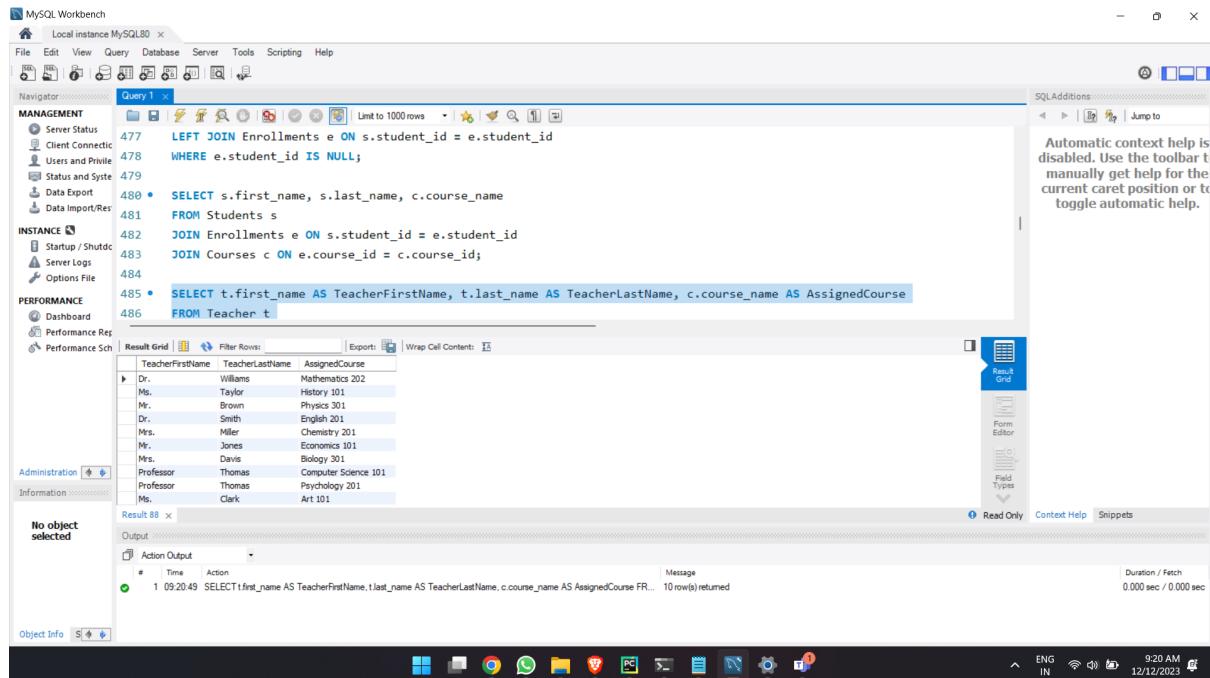
```

The result grid displays nine rows of student-course data:

first_name	last_name	course_name
John	Doe	Computer Science 101
Alice	Smith	Mathematics 202
Bob	Johnson	History 101
Michael	Brown	English 201
Sophia	Miller	Chemistry 201
Daniel	Davis	Computer Science 101
Olivia	Jones	Biology 301
David	Taylor	Psychology 201
John	Doe	Computer Science 101

The status bar at the bottom right shows the time as 9:19 AM and the date as 12/12/2023.

**Q5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.**



The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```

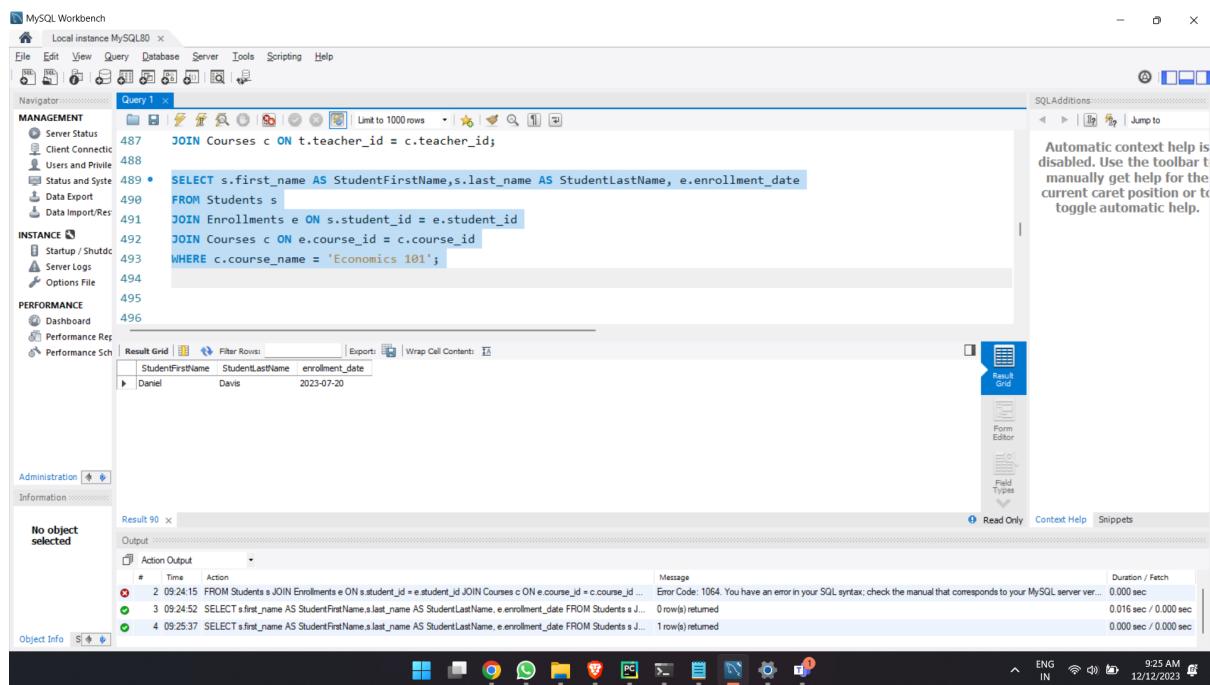
477 LEFT JOIN Enrollments e ON s.student_id = e.student_id
478 WHERE e.student_id IS NULL;
479
480 • SELECT s.first_name, s.last_name, c.course_name
481   FROM Students s
482   JOIN Enrollments e ON s.student_id = e.student_id
483   JOIN Courses c ON e.course_id = c.course_id;
484
485 • SELECT t.first_name AS TeacherFirstName, t.last_name AS TeacherLastName, c.course_name AS AssignedCourse
486   FROM Teacher t
    
```

The results grid displays the following data:

TeacherFirstName	TeacherLastName	AssignedCourse
Dr.	Williams	Mathematics 202
Ms.	Taylor	History 101
Mr.	Brown	Physics 301
Dr.	Smith	English 201
Mrs.	Miller	Chemistry 201
Mr.	Jones	Economics 101
Mrs.	Davis	Biology 301
Professor	Thomas	Computer Science 101
Professor	Thomas	Psychology 201
Ms.	Clark	Art 101

The status bar at the bottom right shows the date and time as 12/12/2023 9:20 AM.

**Q6. 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.**



The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```

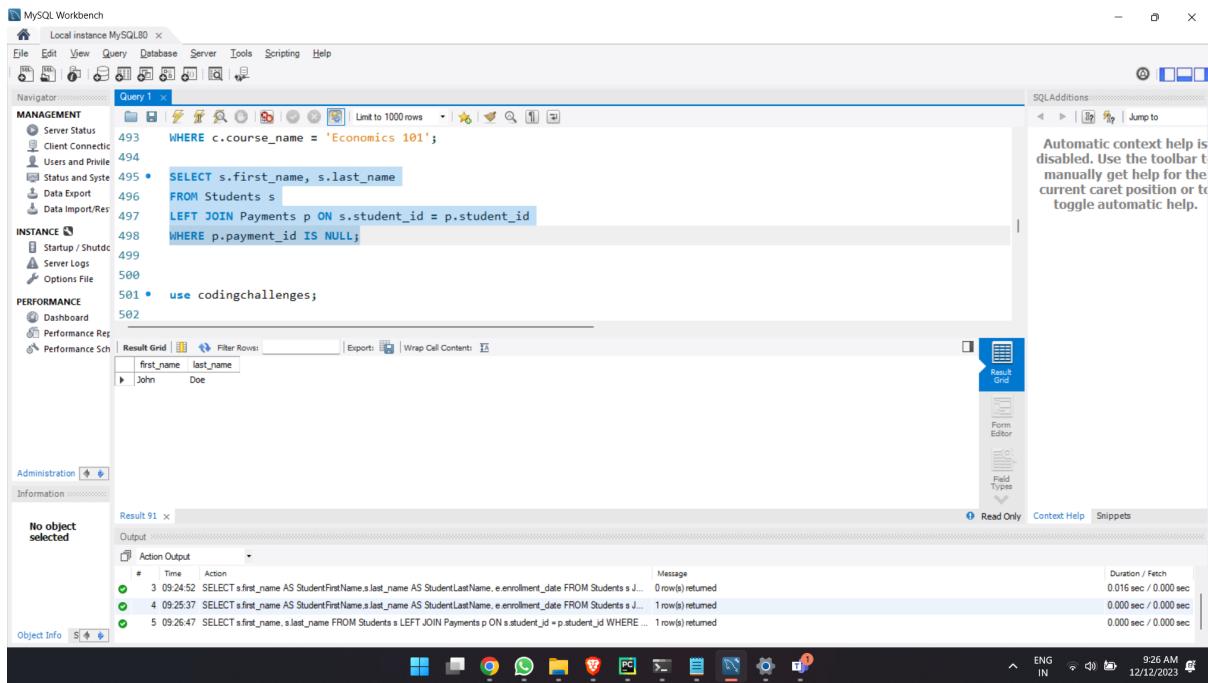
487 JOIN Courses c ON t.teacher_id = c.teacher_id;
488
489 • SELECT s.first_name AS StudentFirstName, s.last_name AS StudentLastName, e.enrollment_date
490   FROM Students s
491   JOIN Enrollments e ON s.student_id = e.student_id
492   JOIN Courses c ON e.course_id = c.course_id
493   WHERE c.course_name = 'Economics 101';
494
495
496
    
```

The results grid displays the following data:

StudentFirstName	StudentLastName	enrollment_date
Daniel	Davis	2023-07-20

The status bar at the bottom right shows the date and time as 12/13/2023 9:25 AM.

**Q7. 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.**



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:
 

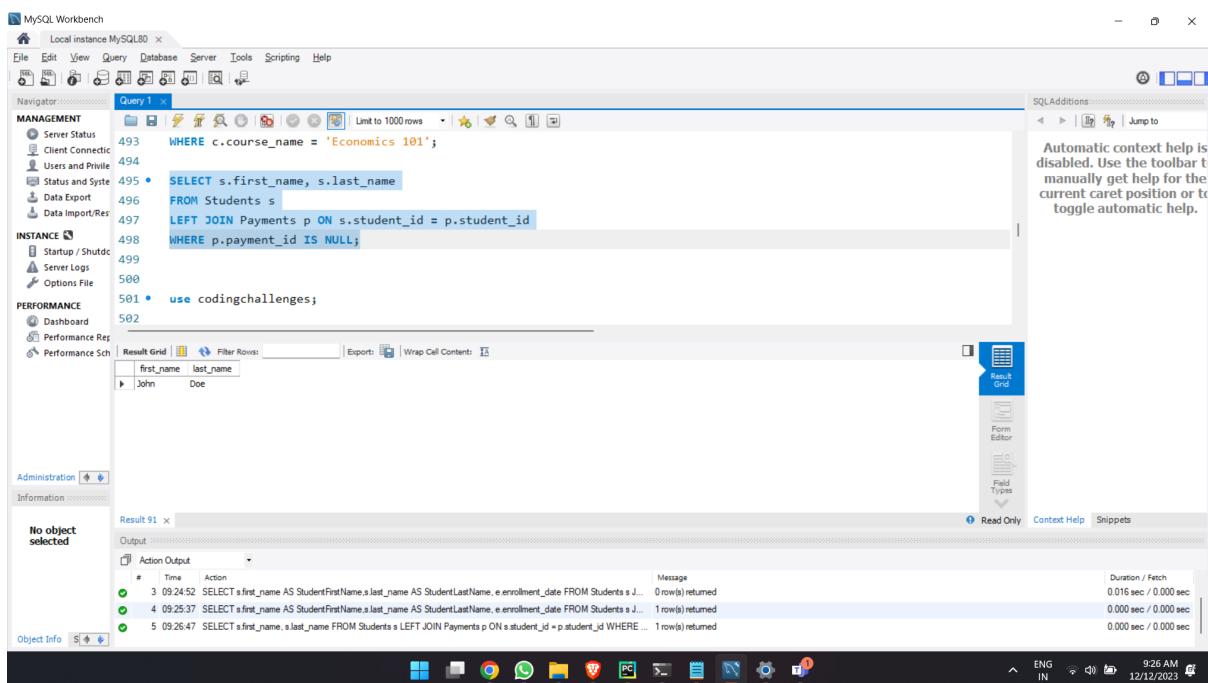
```

493 WHERE c.course_name = 'Economics 101';
494
495 • SELECT s.first_name, s.last_name
496 FROM Students s
497 LEFT JOIN Payments p ON s.student_id = p.student_id
498 WHERE p.payment_id IS NULL;
499
500
501 • use codingchallenges;
502
      
```
- Result Grid:** Displays the output of the query, showing one row:
 

first_name	last_name
John	Doe
- Output Window:** Shows the execution log with three entries:
 

#	Time	Action	Message	Duration / Fetch
3	09:24:52	SELECT s.first_name AS StudentFirstName,s.last_name AS StudentLastName, e.enrollment_date FROM Students s...	0 row(s) returned	0.016 sec / 0.000 sec
4	09:25:37	SELECT s.first_name AS StudentFirstName,s.last_name AS StudentLastName, e.enrollment_date FROM Students s...	1 row(s) returned	0.000 sec / 0.000 sec
5	09:26:47	SELECT s.first_name, s.last_name FROM Students s LEFT JOIN Payments p ON s.student_id = p.student_id WHERE ...	1 row(s) returned	0.000 sec / 0.000 sec

**Q8. 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.**



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:
 

```

493 WHERE c.course_name = 'Economics 101';
494
495 • SELECT s.first_name, s.last_name
496 FROM Students s
497 LEFT JOIN Payments p ON s.student_id = p.student_id
498 WHERE p.payment_id IS NULL;
499
500
501 • use codingchallenges;
502
      
```
- Result Grid:** Displays the output of the query, showing one row:
 

first_name	last_name
John	Doe
- Output Window:** Shows the execution log with three entries:
 

#	Time	Action	Message	Duration / Fetch
3	09:24:52	SELECT s.first_name AS StudentFirstName,s.last_name AS StudentLastName, e.enrollment_date FROM Students s...	0 row(s) returned	0.016 sec / 0.000 sec
4	09:25:37	SELECT s.first_name AS StudentFirstName,s.last_name AS StudentLastName, e.enrollment_date FROM Students s...	1 row(s) returned	0.000 sec / 0.000 sec
5	09:26:47	SELECT s.first_name, s.last_name FROM Students s LEFT JOIN Payments p ON s.student_id = p.student_id WHERE ...	1 row(s) returned	0.000 sec / 0.000 sec

## TASK 4: Subquery and its type:

Q1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is as follows:

```
498 WHERE p.payment_id IS NULL;
499
500 • SELECT
501     AVG(StudentCount) AS AverageStudentsPerCourse
502     FROM (SELECT c.course_id, COUNT(e.student_id) AS StudentCount
503             FROM Courses c
504             LEFT JOIN Enrollments e ON c.course_id = e.course_id
505             GROUP BY c.course_id)
506     AS CourseStudentCounts;
507
```

The result grid shows a single row with the value 0.9000 under the column "AverageStudentsPerCourse". The status bar at the bottom right indicates the time as 9:38 AM and the date as 12/12/2023.

Q2. 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.

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is as follows:

```
501     AVG(StudentCount) AS AverageStudentsPerCourse
502     FROM (SELECT c.course_id, COUNT(e.student_id) AS StudentCount
503             FROM Courses c
504             LEFT JOIN Enrollments e ON c.course_id = e.course_id
505             GROUP BY c.course_id)
506     AS CourseStudentCounts;
507
508 • SELECT s.first_name, s.last_name, p.amount AS MaximumPayment
509     FROM Students s
510
```

The result grid shows a single row with the values Daniel, Davis, and 900 under the columns "first\_name", "last\_name", and "MaximumPayment". The status bar at the bottom right indicates the time as 9:50 AM and the date as 12/12/2023.

**Q3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.**

```

MySQL Workbench
Local instance MySQL80 x
File Edit View Query Database Server Tools Scripting Help
Navigator: Query1 x
MANAGEMENT
    Server Status
    Client Connectic
    Users and Privile
    Status and Syste
    Data Export
    Data Import/Res
INSTANCE
    Startup / Shuttin
    Server Logs
    Options File
PERFORMANCE
    Dashboard
    Performance Rep
    Performance Sch
Administration
Information
No object selected
Result 94 x
Output
Action Output
# Time Action
1 09:52:33 SELECT c.course_id, c.course_name, COUNT(e.student_id) AS EnrollmentCount FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id GROUP BY c.course_id, c.course_name
Object Info
Result Grid | Filter Rows | Export: | Wrap Cell Content: |
course_id course_name EnrollmentCount
1 Computer Science 101 2
Result 94 x
Output
Action Output
# Time Action
1 09:52:33 SELECT c.course_id, c.course_name, COUNT(e.student_id) AS EnrollmentCount FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id GROUP BY c.course_id, c.course_name
Object Info
Result Grid | Filter Rows | Export: | Wrap Cell Content: |
course_id course_name EnrollmentCount
1 Computer Science 101 2

```

The screenshot shows the MySQL Workbench interface. The main window displays a query in the SQL editor:

```

508 • SELECT s.first_name, s.last_name, p.amount AS MaximumPayment
      FROM Students s
      JOIN Payments p ON s.student_id = p.student_id
     WHERE p.amount = (SELECT MAX(amount) FROM Payments);
513
514 • SELECT c.course_id, c.course_name, COUNT(e.student_id) AS EnrollmentCount
      FROM Courses c
      LEFT JOIN Enrollments e ON c.course_id = e.course_id
     GROUP BY c.course_id, c.course_name;

```

The results grid shows one row:

course_id	course_name	EnrollmentCount
1	Computer Science 101	2

The status bar at the bottom right indicates the time as 9:52 AM and the date as 12/12/2023.

**Q4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.**

```

MySQL Workbench
Local instance MySQL80 x
File Edit View Query Database Server Tools Scripting Help
Navigator: Query1 x
MANAGEMENT
    Server Status
    Client Connectic
    Users and Privile
    Status and Syste
    Data Export
    Data Import/Res
INSTANCE
    Startup / Shuttin
    Server Logs
    Options File
PERFORMANCE
    Dashboard
    Performance Rep
    Performance Sch
Administration
Information
No object selected
Result 95 x
Output
Action Output
# Time Action
1 09:54:31 SELECT t.teacher_id, t.first_name AS TeacherFirstName, t.last_name AS TeacherLastName, COALESCE(SUM(p.amount), 0) AS TotalPayments
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;
Object Info
Result Grid | Filter Rows | Export: | Wrap Cell Content: |
teacher_id TeacherFirstName TeacherLastName TotalPayments
1 Professor Johnson 0
2 Dr. Williams 750
3 Ms. Taylor 600
4 Mr. Brown 0
5 Dr. Smith 550
6 Mrs. Miller 700
7 Mr. Jones 900
8 Mrs. Davis 650
9 Professor Thomas 1880
10 Ms. Clark 0
Result 95 x
Output
Action Output
# Time Action
1 09:54:31 SELECT t.teacher_id, t.first_name AS TeacherFirstName, t.last_name AS TeacherLastName, COALESCE(SUM(p.amount), 0) AS TotalPayments
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;
Object Info
Result Grid | Filter Rows | Export: | Wrap Cell Content: |
teacher_id TeacherFirstName TeacherLastName TotalPayments
1 Professor Johnson 0
2 Dr. Williams 750
3 Ms. Taylor 600
4 Mr. Brown 0
5 Dr. Smith 550
6 Mrs. Miller 700
7 Mr. Jones 900
8 Mrs. Davis 650
9 Professor Thomas 1880
10 Ms. Clark 0

```

The screenshot shows the MySQL Workbench interface. The main window displays a query in the SQL editor:

```

517 GROUP BY c.course_id, c.course_name
      HAVING COUNT(e.student_id) = (SELECT MAX(EnrollmentCount) FROM (SELECT course_id, COUNT(student_id) AS EnrollmentCount
      FROM Courses c
      LEFT JOIN Enrollments e ON c.course_id = e.course_id
      GROUP BY c.course_id, c.course_name) AS Subquery)
518
519
520 • SELECT t.teacher_id, t.first_name AS TeacherFirstName, t.last_name AS TeacherLastName, COALESCE(SUM(p.amount), 0) AS TotalPayments
      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;

```

The results grid shows ten rows:

teacher_id	TeacherFirstName	TeacherLastName	TotalPayments
1	Professor	Johnson	0
2	Dr.	Williams	750
3	Ms.	Taylor	600
4	Mr.	Brown	0
5	Dr.	Smith	550
6	Mrs.	Miller	700
7	Mr.	Jones	900
8	Mrs.	Davis	650
9	Professor	Thomas	1880
10	Ms.	Clark	0

The status bar at the bottom right indicates the time as 9:54 AM and the date as 12/12/2023.

**Q5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.**

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```
523     LEFT JOIN Enrollments e ON c.course_id = e.course_id
524     LEFT JOIN Payments p ON e.student_id = p.student_id
525     GROUP BY t.teacher_id, t.first_name, t.last_name;
526
527 • SELECT s.student_id, s.first_name, s.last_name
528   FROM Students s
529   WHERE NOT EXISTS (SELECT
530                      c.course_id
531                      FROM
532                          Courses c
```

The result grid shows one row with all columns set to NULL:

student_id	first_name	last_name
NULL	NULL	NULL

The status bar at the bottom right indicates the duration of 0.015 sec / 0.000 sec.

**Q6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.**

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```
542     );
543   );
544
545 • SELECT t.teacher_id, t.first_name AS TeacherFirstName, t.last_name AS TeacherLastName
546   FROM Teacher t
547   WHERE NOT EXISTS (
548     SELECT
549       c.course_id
550       FROM
551           Courses c
```

The result grid shows one row with teacher\_id 1, TeacherFirstName Professor, and TeacherLastName Johnson:

teacher_id	TeacherFirstName	TeacherLastName
1	Professor	Johnson

The status bar at the bottom right indicates the duration of 0.016 sec / 0.000 sec.

**Q7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.**

```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Management
  Server Status
  Client Connectic
  Users and Privile
  Status and Syste
  Data Export
  Data Import/Res
  Startup / Shutt
  Server Logs
  Options File
  Performance
  Dashboard
  Performance Rep
  Performance Sch
  Administration
    Information
      No object selected
  Result Grid
    Filter Rows
    Export
    Wrap Cell Content
    AverageAge
    > 27.25753425
  SQLAdditions
    Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Query 1 x
  553   c.teacher_id = t.teacher_id
  554   );
  555 •   SELECT AVG(Age) AS AverageAge
  556   FROM (
  557     SELECT
  558       DATEDIFF(CURDATE(), d_o_b) / 365 AS Age
  559     FROM
  560       Students
  561   ) AS StudentAges;
  562

  Result 98 x
  Output
    Action Output
    # Time Action
    1 10:01:30 SELECT AVG(Age) AS AverageAge FROM ( SELECT DATEDIFF(CURDATE(),d_o_b) / 365 AS Age FROM ... 1 row(s) returned
  Duration / Fetch
  0.000 sec / 0.000 sec
  Object Info
  ENG IN 10:01 AM 12/12/2023

```

**Q8. Identify courses with no enrollments. Use subqueries to find courses without enrollment records.**

```

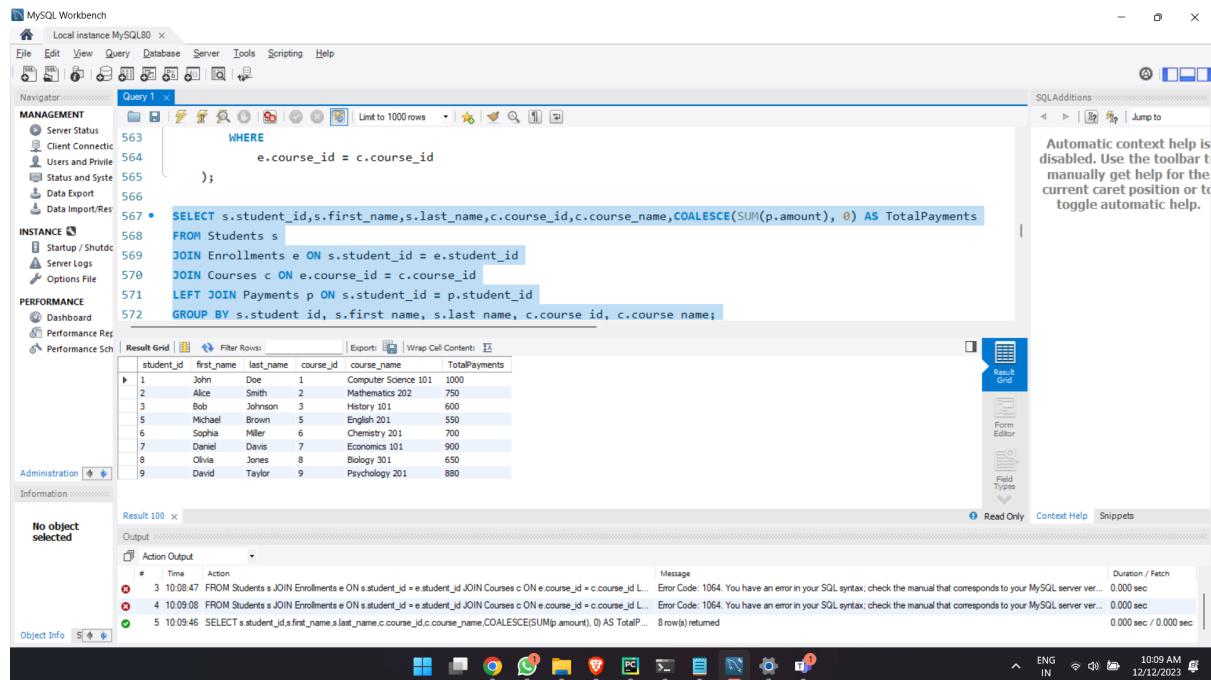
MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Management
  Server Status
  Client Connectic
  Users and Privile
  Status and Syste
  Data Export
  Data Import/Res
  Startup / Shutt
  Server Logs
  Options File
  Performance
  Dashboard
  Performance Rep
  Performance Sch
  Administration
    Information
      No object selected
  Result Grid
    Filter Rows
    Edit
    Export/Import
    Wrap Cell Content
    course_id course_name
    > 4 Physics 301
    10 Art 101
  SQLAdditions
    Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Query 1 x
  555   Execute the selected portion of the script or everything, if there is no selection
  556 •   SELECT c.course_id,c.course_name
  557   FROM Courses c
  558   WHERE NOT EXISTS (
  559     SELECT
  560       e.course_id
  561     FROM
  562       Enrollments e
  563     WHERE
  564       e.course_id = c.course_id
  )
  Result 99 x
  Output
    Action Output
    # Time Action
    1 10:01:30 SELECT AVG(Age) AS AverageAge FROM ( SELECT DATEDIFF(CURDATE(),d_o_b) / 365 AS Age FROM ... 1 row(s) returned
    2 10:06:31 SELECT c.course_id,c.course_name FROM Courses c WHERE NOT EXISTS ( SELECT e.course_id ... 2 row(s) returned
  Duration / Fetch
  0.000 sec / 0.000 sec
  0.000 sec / 0.000 sec
  Object Info
  ENG IN 10:06 AM 12/12/2023

```

**Q9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.**



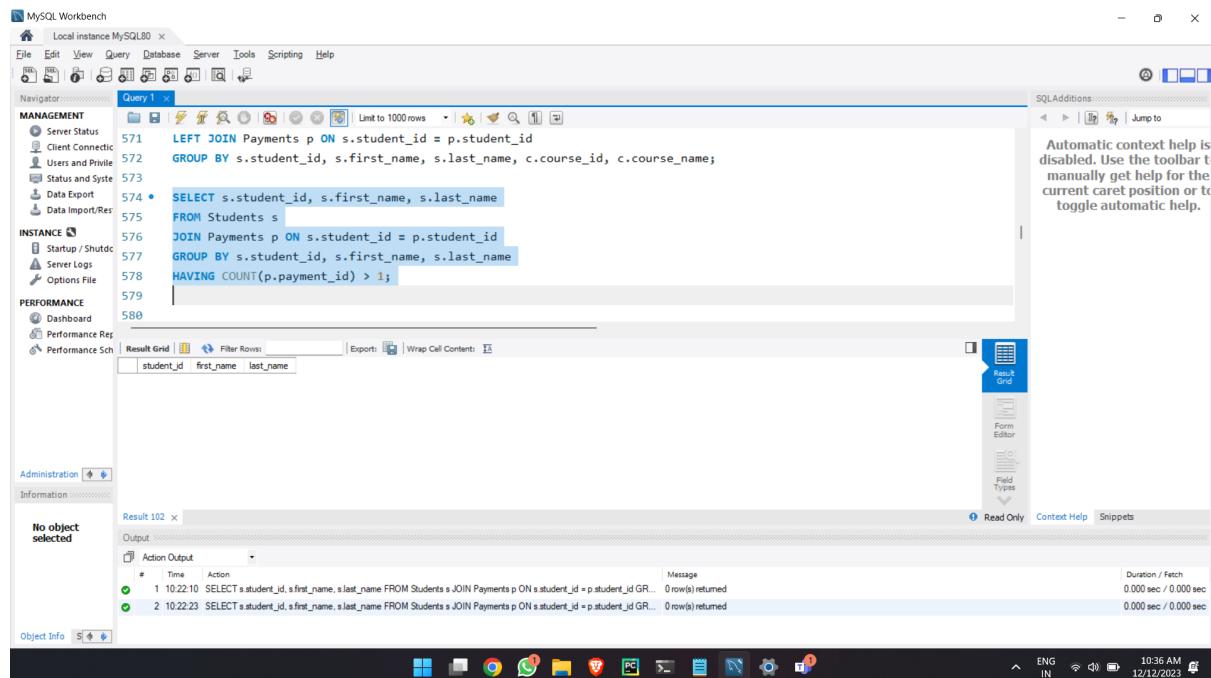
```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Navigator: MANAGEMENT, INSTANCE, PERFORMANCE, Administration, Information
Query 1 x
WHERE
e.course_id = c.course_id
);
567 • SELECT s.student_id, s.first_name, s.last_name, c.course_id, c.course_name, COALESCE(SUM(p.amount), 0) AS TotalPayments
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
LEFT JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name, s.last_name, c.course_id, c.course_name;
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
student_id first_name last_name course_id course_name TotalPayments
1 John Doe 1 Computer Science 101 1000
2 Alice Smith 2 Mathematics 202 750
3 Bob Johnson 3 History 101 600
4 Michael Brown 5 English 201 550
5 Sophia Miller 6 Chemistry 201 700
6 Daniel Davis 7 Economics 101 900
7 Olivia Jones 8 Biology 301 650
8 David Taylor 9 Psychology 201 880
Result 100 x
Output
Action Output
# Time Action Message Duration / Fetch
3 10:08:47 FROM Students s JOIN Enrollments e ON s.student_id = e.student_id JOIN Courses c ON e.course_id = c.course_id L... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server ver... 0.000 sec
4 10:09:08 FROM Students s JOIN Enrollments e ON s.student_id = e.student_id JOIN Courses c ON e.course_id = c.course_id L... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server ver... 0.000 sec
5 10:09:46 SELECT s.student_id, s.first_name, s.last_name, c.course_id, c.course_name, COALESCE(SUM(p.amount), 0) AS TotalP... 8 row(s) returned 0.000 sec / 0.000 sec
Object Info S
Read Only Context Help Snippets
ENG IN 10:09 AM 12/12/2023

```

**Q10. 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.**



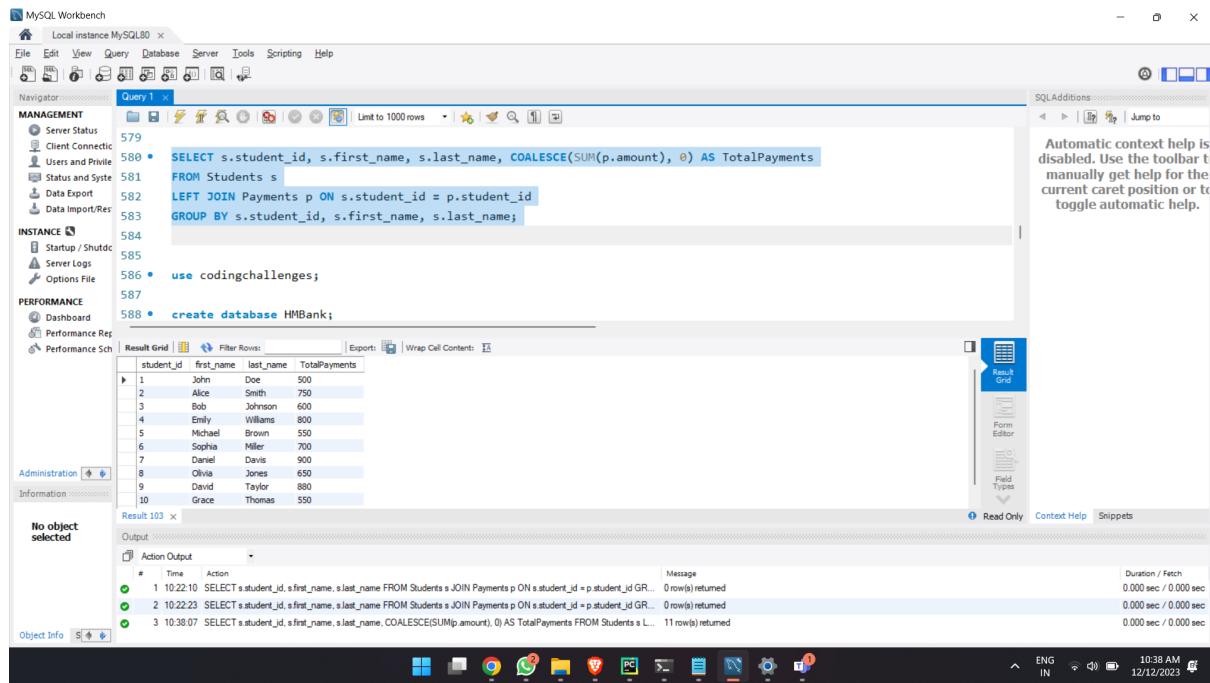
```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Navigator: MANAGEMENT, INSTANCE, PERFORMANCE, Administration, Information
Query 1 x
LEFT JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name, s.last_name, c.course_id, c.course_name;
571
572
573
574 • SELECT s.student_id, s.first_name, s.last_name
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name, s.last_name
HAVING COUNT(p.payment_id) > 1;
575
576
577
578
579
580
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
student_id first_name last_name
Result 102 x
Output
Action Output
# Time Action Message Duration / Fetch
1 10:22:10 SELECT s.student_id, s.first_name, s.last_name FROM Students s JOIN Payments p ON s.student_id = p.student_id GR... 0 row(s) returned 0.000 sec / 0.000 sec
2 10:22:23 SELECT s.student_id, s.first_name, s.last_name FROM Students s JOIN Payments p ON s.student_id = p.student_id GR... 0 row(s) returned 0.000 sec / 0.000 sec
Object Info S
Read Only Context Help Snippets
ENG IN 10:36 AM 12/12/2023

```

**Q11. 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.**



The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1'. The code entered is:

```

579
580 • SELECT s.student_id, s.first_name, s.last_name, COALESCE(SUM(p.amount), 0) AS TotalPayments
581   FROM Students s
582   LEFT JOIN Payments p ON s.student_id = p.student_id
583   GROUP BY s.student_id, s.first_name, s.last_name;
584
585
586 • use codingchallenges;
587
588 • create database HMBank;

```

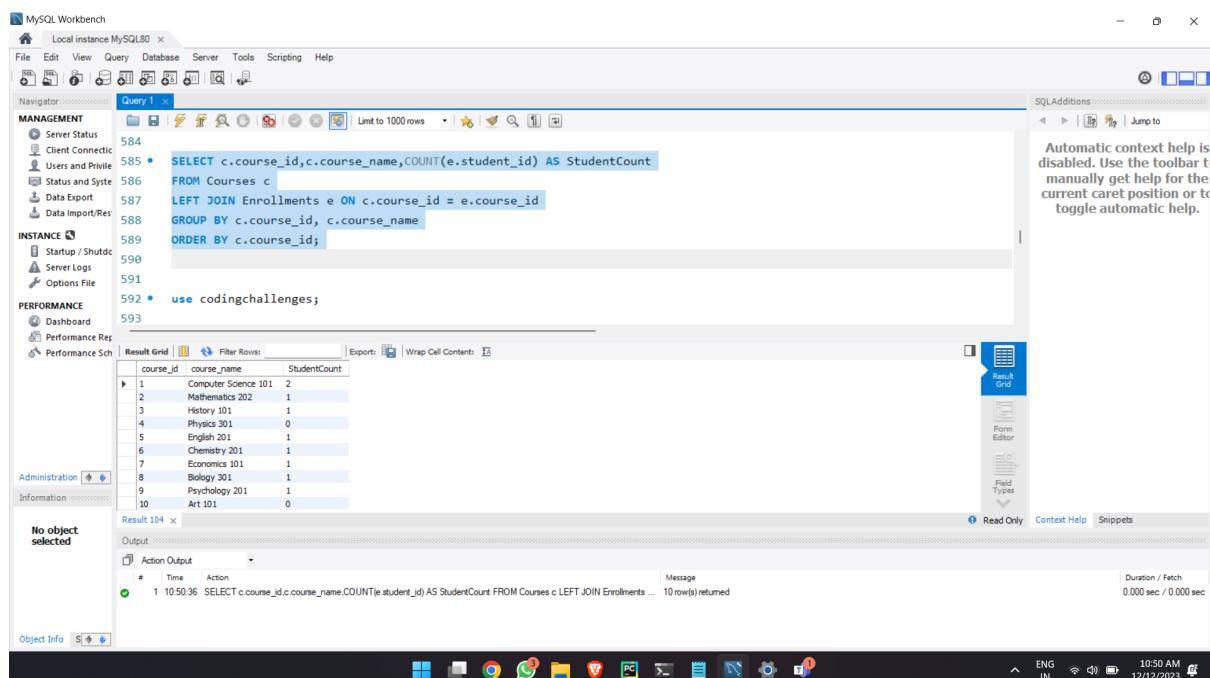
The results grid displays the following data:

student_id	first_name	last_name	TotalPayments
1	John	Doe	500
2	Alice	Smith	750
3	Bob	Johnson	600
4	Emily	Williams	800
5	Michael	Brown	550
6	Sophia	Miller	700
7	Daniel	Davis	900
8	Olivia	Jones	650
9	David	Taylor	880
10	Grace	Thomas	550

The output pane shows the execution log:

- 1 10:22:10 SELECT s.student\_id, s.first\_name, s.last\_name FROM Students s JOIN Payments p ON s.student\_id = p.student\_id GROUP BY s.student\_id, s.first\_name, s.last\_name; 0 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec
- 2 10:22:23 SELECT s.student\_id, s.first\_name, s.last\_name FROM Students s JOIN Payments p ON s.student\_id = p.student\_id GROUP BY s.student\_id, s.first\_name, s.last\_name; 0 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec
- 3 10:38:07 SELECT s.student\_id, s.first\_name, s.last\_name, COALESCE(SUM(p.amount), 0) AS TotalPayments FROM Students s LEFT JOIN Payments p ON s.student\_id = p.student\_id GROUP BY s.student\_id, s.first\_name, s.last\_name; 11 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec

**Q12. 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.**



The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1'. The code entered is:

```

584
585 • SELECT c.course_id, c.course_name,COUNT(e.student_id) AS StudentCount
586   FROM Courses c
587   LEFT JOIN Enrollments e ON c.course_id = e.course_id
588   GROUP BY c.course_id, c.course_name
589
590
591
592 • use codingchallenges;
593

```

The results grid displays the following data:

course_id	course_name	StudentCount
1	Computer Science 101	2
2	Mathematics 202	1
3	History 101	1
4	Physics 301	0
5	English 201	1
6	Chemistry 201	1
7	Economics 101	1
8	Biology 301	1
9	Psychology 201	1
10	Art 101	0

The output pane shows the execution log:

- 1 10:50:36 SELECT c.course\_id,c.course\_name,COUNT(e.student\_id) AS StudentCount FROM Courses c LEFT JOIN Enrollments ... 10 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec

**Q13. 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.**

The screenshot shows the MySQL Workbench interface. In the top-left corner, the title bar reads "MySQL Workbench Local instance MySQL80". The main area contains a SQL editor window titled "Query 1" with the following code:

```
587 LEFT JOIN Enrollments e ON c.course_id = e.course_id
588 GROUP BY c.course_id, c.course_name
589 ORDER BY c.course_id;
590
591 • SELECT s.student_id, s.first_name, s.last_name, COALESCE(AVG(p.amount), 0) AS AveragePaymentAmount
592 FROM Students s
593 LEFT JOIN Payments p ON s.student_id = p.student_id
594 GROUP BY s.student_id, s.first_name, s.last_name;
595
596
```

Below the SQL editor is a "Result Grid" window displaying the query results:

student_id	first_name	last_name	AveragePaymentAmount
1	John	Doe	500
2	Alice	Smith	750
3	Bob	Johnson	600
4	Emily	Williams	800
5	Michael	Brown	550
6	Sophia	Miller	700
7	Daniel	Davis	900
8	Olivia	Jones	650
9	David	Taylor	880
10	Grace	Thomas	550

At the bottom of the interface, the status bar shows "Result 105" and the system date and time as "12/12/2023 10:57 AM".