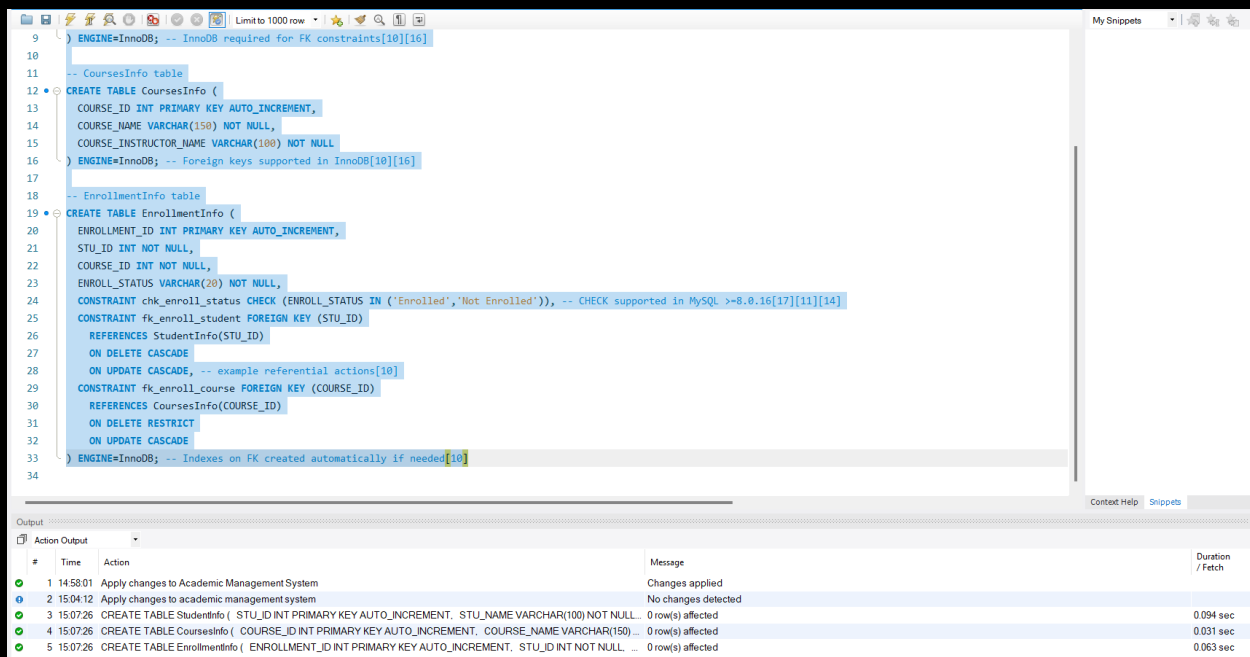


## [ Task 1 ]Project Title: Academic Management System (using SQL)

Project Description : Design and develop an Academic Management System using SQL. The project should involve three tables: 1. StudentInfo 2. CoursesInfo 3. EnrollmentInfo. The aim is to create a system that allows for managing student information and course enrollment. The project will include the following tasks:

### 1) Database Creation:

- Create the StudentInfo table with columns STU\_ID, STU\_NAME, DOB, PHONE\_NO, EMAIL\_ID, ADDRESS.
- Create the CoursesInfo table with columns COURSE\_ID, COURSE\_NAME, COURSE\_INSTRUCTOR\_NAME.
- Create the EnrollmentInfo table with columns ENROLLMENT\_ID, STU\_ID, COURSE\_ID, ENROLL\_STATUS (Enrolled/Not Enrolled). The FOREIGN KEY constraint in the EnrollmentInfo table references the STU\_ID column in the StudentInfo table and the COURSE\_ID column in the CoursesInfo table.



```
9  ) ENGINE=InnoDB; -- InnoDB required for FK constraints[10][16]
10
11  -- CoursesInfo table
12  CREATE TABLE CoursesInfo (
13    COURSE_ID INT PRIMARY KEY AUTO_INCREMENT,
14    COURSE_NAME VARCHAR(150) NOT NULL,
15    COURSE_INSTRUCTOR_NAME VARCHAR(100) NOT NULL
16  ) ENGINE=InnoDB; -- Foreign keys supported in InnoDB[10][16]
17
18  -- EnrollmentInfo table
19  CREATE TABLE EnrollmentInfo (
20    ENROLLMENT_ID INT PRIMARY KEY AUTO_INCREMENT,
21    STU_ID INT NOT NULL,
22    COURSE_ID INT NOT NULL,
23    ENROLL_STATUS VARCHAR(20) NOT NULL,
24    CONSTRAINT chk_enroll_status CHECK (ENROLL_STATUS IN ('Enrolled','Not Enrolled')), -- CHECK supported in MySQL >=8.0.16[17][11][14]
25    CONSTRAINT fk_enroll_student FOREIGN KEY (STU_ID)
26      REFERENCES StudentInfo(STU_ID)
27      ON DELETE CASCADE
28      ON UPDATE CASCADE, -- example referential actions[10]
29    CONSTRAINT fk_enroll_course FOREIGN KEY (COURSE_ID)
30      REFERENCES CoursesInfo(COURSE_ID)
31      ON DELETE RESTRICT
32      ON UPDATE CASCADE
33  ) ENGINE=InnoDB; -- Indexes on FK created automatically if needed[10]
34
```

#	Time	Action	Message	Duration / Fetch
1	14:58:01	Apply changes to Academic Management System	Changes applied	
2	15:04:12	Apply changes to academic management system	No changes detected	
3	15:07:26	CREATE TABLE StudentInfo ( STU_ID INT PRIMARY KEY AUTO_INCREMENT, STU_NAME VARCHAR(100) NOT NULL, ...	0 row(s) affected	0.094 sec
4	15:07:26	CREATE TABLE CoursesInfo ( COURSE_ID INT PRIMARY KEY AUTO_INCREMENT, COURSE_NAME VARCHAR(150) ...	0 row(s) affected	0.031 sec
5	15:07:26	CREATE TABLE EnrollmentInfo ( ENROLLMENT_ID INT PRIMARY KEY AUTO_INCREMENT, STU_ID INT NOT NULL, ...	0 row(s) affected	0.063 sec

### 2) Data Creation:

Insert some sample data for StudentInfo table, CoursesInfo table, and EnrollmentInfo with respective fields.

```

35 • INSERT INTO StudentInfo (STU_NAME, DOB, PHONE_NO, EMAIL_ID, ADDRESS) VALUES
36 ('Alice Johnson', '2003-04-15', '555-1111', 'alice@example.com', '12 Oak St'),
37 ('Bob Smith', '2002-11-30', '555-2222', 'bob@example.com', '34 Pine Ave'),
38 ('Charlie Kim', '2001-06-08', '555-3333', 'charlie@example.com', '56 Maple Rd'),
39 ('Diana Patel', '2003-02-20', '555-4444', 'diana@example.com', '78 Cedar Ln'),
40 ('Ethan Brown', '2002-09-05', '555-5555', 'ethan@example.com', '90 Birch Dr');
41
42 • INSERT INTO CoursesInfo (COURSE_NAME, COURSE_INSTRUCTOR_NAME) VALUES
43 ('Database Systems', 'Dr. Green'),
44 ('Operating Systems', 'Dr. White'),
45 ('Algorithms', 'Dr. Black'),
46 ('Data Structures', 'Dr. Gray');
47
48 -- Enrollments: include Enrolled and Not Enrolled for testing filters
49 • INSERT INTO EnrollmentInfo (STU_ID, COURSE_ID, ENROLL_STATUS) VALUES
50 (1, 1, 'Enrolled'),
51 (1, 2, 'Enrolled'),
52 (2, 1, 'Not Enrolled'),
53 (2, 3, 'Enrolled'),
54 (3, 1, 'Enrolled'),
55 (3, 4, 'Enrolled'),
56 (4, 2, 'Enrolled'),
57 (4, 3, 'Not Enrolled'),
58 (5, 3, 'Enrolled');
59
60

```

#	Time	Action	Message	Duration / Fetch
4	15:07:26	CREATE TABLE CoursesInfo ( COURSE_ID INT PRIMARY KEY AUTO_INCREMENT, COURSE_NAME VARCHAR(150) ...	0 row(s) affected	0.031 sec
5	15:07:26	CREATE TABLE EnrollmentInfo ( ENROLLMENT_ID INT PRIMARY KEY AUTO_INCREMENT, STU_ID INT NOT NULL ...	0 row(s) affected	0.063 sec
6	15:10:40	INSERT INTO StudentInfo (STU_NAME, DOB, PHONE_NO, EMAIL_ID, ADDRESS) VALUES ('Alice Johnson', '2003-04-15'...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
7	15:10:40	INSERT INTO CoursesInfo (COURSE_NAME, COURSE_INSTRUCTOR_NAME) VALUES ('Database Systems', 'Dr. Gree...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
8	15:10:40	INSERT INTO EnrollmentInfo (STU_ID, COURSE_ID, ENROLL_STATUS) VALUES (1, 1, 'Enrolled'), (1, 2, 'Enrolled'), (2...	9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0	0.016 sec

### 3) Retrieve the Student Information:

a) Write a query to retrieve student details, such as student name, contact information, and enrollment status.

```

46 ('Data Structures', 'Dr. Gray');
47
48 -- Enrollments: include Enrolled and Not Enrolled for testing filters
49 • INSERT INTO EnrollmentInfo (STU_ID, COURSE_ID, ENROLL_STATUS) VALUES
50 (1, 1, 'Enrolled'),
51 (1, 2, 'Enrolled'),
52 (2, 1, 'Not Enrolled'),
53 (2, 3, 'Enrolled'),
54 (3, 1, 'Enrolled'),
55 (3, 4, 'Enrolled'),
56 (4, 2, 'Enrolled'),
57 (4, 3, 'Not Enrolled'),
58 (5, 3, 'Enrolled');
59
60 • SELECT
61 s.STU_ID,
62 s.STU_NAME,
63 s.PHONE_NO,
64 s.EMAIL_ID,

```

STU_ID	STU_NAME	PHONE_NO	EMAIL_ID	ENROLL_STATUS	COURSE_ID
1	Alice Johnson	555-1111	alice@example.com	Enrolled	1
1	Alice Johnson	555-1111	alice@example.com	Enrolled	2
2	Bob Smith	555-2222	bob@example.com	Not Enrolled	1
2	Bob Smith	555-2222	bob@example.com	Enrolled	3
3	Charlie Kim	555-3333	charlie@example.com	Enrolled	1
3	Charlie Kim	555-3333	charlie@example.com	Enrolled	4

#	Time	Action	Message	Duration / Fetch
5	15:07:26	CREATE TABLE EnrollmentInfo ( ENROLLMENT_ID INT PRIMARY KEY AUTO_INCREMENT, STU_ID INT NOT N...	0 row(s) affected	0.063 sec
6	15:10:40	INSERT INTO StudentInfo (STU_NAME, DOB, PHONE_NO, EMAIL_ID, ADDRESS) VALUES ('Alice Johnson', '2003-0...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
7	15:10:40	INSERT INTO CoursesInfo (COURSE_NAME, COURSE_INSTRUCTOR_NAME) VALUES ('Database Systems', 'Dr. ...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
8	15:10:40	INSERT INTO EnrollmentInfo (STU_ID, COURSE_ID, ENROLL_STATUS) VALUES (1, 1, 'Enrolled'), (1, 2, 'Enrolled'), (2...	9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0	0.016 sec
9	15:11:47	SELECT s.STU_ID, s.STU_NAME, s.PHONE_NO, s.EMAIL_ID, e.ENROLL_STATUS, e.COURSE_ID FROM Stu...	9 row(s) returned	0.000 sec / 0.000 sec

b) Write a query to retrieve a list of courses in which a specific student is enrolled.

Limit to 1000 row

```

65 e.ENROLL_STATUS,
66 e.COURSE_ID
67 FROM StudentInfo s
68 LEFT JOIN EnrollmentInfo e
69 ON s.STU_ID = e.STU_ID
70 ORDER BY s.STU_ID, e.COURSE_ID;
71
72 -- Example: courses for STU_ID = 1 and status Enrolled
73 • SELECT
74   c.COURSE_ID,
75   c.COURSE_NAME,
76   c.COURSE_INSTRUCTOR_NAME,
77   e.ENROLL_STATUS
78 FROM EnrollmentInfo e
79 JOIN CoursesInfo c
80 ON e.COURSE_ID = c.COURSE_ID
81 WHERE e.STU_ID = 1
82 AND e.ENROLL_STATUS = 'Enrolled'
83 ORDER BY c.COURSE_NAME;

```

Result Grid

	COURSE_ID	COURSE_NAME	COURSE_INSTRUCTOR_NAME	ENROLL_STATUS
1	Database Systems	Dr. Green	Enrolled	
2	Operating Systems	Dr. White	Enrolled	

Output

#	Time	Action	Message	Duration / Fetch
6	15:10:40	INSERT INTO StudentInfo (STU_NAME, DOB, PHONE_NO, EMAIL_ID, ADDRESS) VALUES ('Alice Johnson', '2003-0...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
7	15:10:40	INSERT INTO CoursesInfo (COURSE_NAME, COURSE_INSTRUCTOR_NAME) VALUES ('Database Systems', 'Dr. ...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
8	15:10:40	INSERT INTO EnrollmentInfo (STU_ID, COURSE_ID, ENROLL_STATUS) VALUES (1, 1, 'Enrolled'), (1, 2, 'Enrolled'), (2...	9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0	0.016 sec
9	15:11:47	SELECT s.STU_ID, s.STU_NAME, s.PHONE_NO, s.EMAIL_ID, e.ENROLL_STATUS, e.COURSE_ID FROM Stu...	9 row(s) returned	0.000 sec / 0.000 sec
10	15:13:06	SELECT c.COURSE_ID, c.COURSE_NAME, c.COURSE_INSTRUCTOR_NAME, e.ENROLL_STATUS FROM Enr...	2 row(s) returned	0.000 sec / 0.000 sec

c) Write a query to retrieve course information, including course name and instructor information.

Limit to 1000 row

```

69 ON s.STU_ID = e.STU_ID
70 ORDER BY s.STU_ID, e.COURSE_ID;
71
72 -- Example: courses for STU_ID = 1 and status Enrolled
73 • SELECT
74   c.COURSE_ID,
75   c.COURSE_NAME,
76   c.COURSE_INSTRUCTOR_NAME,
77   e.ENROLL_STATUS
78 FROM EnrollmentInfo e
79 JOIN CoursesInfo c
80 ON e.COURSE_ID = c.COURSE_ID
81 WHERE e.STU_ID = 1
82 AND e.ENROLL_STATUS = 'Enrolled'
83 ORDER BY c.COURSE_NAME;
84
85 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
86 FROM CoursesInfo
87 ORDER BY COURSE_ID;

```

Result Grid

	COURSE_ID	COURSE_NAME	COURSE_INSTRUCTOR_NAME
1	Database Systems	Dr. Green	
2	Operating Systems	Dr. White	
3	Algorithms	Dr. Black	
4	Data Structures	Dr. Gray	

Output

#	Time	Action	Message	Duration / Fetch
7	15:10:40	INSERT INTO CoursesInfo (COURSE_NAME, COURSE_INSTRUCTOR_NAME) VALUES ('Database Systems', 'Dr. ...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
8	15:10:40	INSERT INTO EnrollmentInfo (STU_ID, COURSE_ID, ENROLL_STATUS) VALUES (1, 1, 'Enrolled'), (1, 2, 'Enrolled'), (2...	9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0	0.016 sec
9	15:11:47	SELECT s.STU_ID, s.STU_NAME, s.PHONE_NO, s.EMAIL_ID, e.ENROLL_STATUS, e.COURSE_ID FROM Stu...	9 row(s) returned	0.000 sec / 0.000 sec
10	15:13:06	SELECT c.COURSE_ID, c.COURSE_NAME, c.COURSE_INSTRUCTOR_NAME, e.ENROLL_STATUS FROM Enr...	2 row(s) returned	0.000 sec / 0.000 sec
11	15:13:45	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo ORDER BY COURSE...	4 row(s) returned	0.000 sec / 0.000 sec

d) Write a query to retrieve course information for a specific course (Algorithms).

```

81 WHERE e.STU_ID = 1
82 AND e.ENROLL_STATUS = 'Enrolled'
83 ORDER BY c.COURSE_NAME;
84
85 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
86 FROM CoursesInfo
87 ORDER BY COURSE_ID;
88
89 -- By id
90 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
91 FROM CoursesInfo
92 WHERE COURSE_ID = 3;
93
94 -- Or by name
95 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
96 FROM CoursesInfo
97 WHERE COURSE_NAME = 'Algorithms';
98
99

```

Result Grid

COURSE_ID	COURSE_NAME	COURSE_INSTRUCTOR_NAME
3	Algorithms	Dr. Black

CoursesInfo 7 x

Output

#	Time	Action	Message	Duration / Fetch
12	15:14:45	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	1 row(s) returned	0.000 sec / 0.000 sec
13	15:14:45	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	1 row(s) returned	0.000 sec / 0.000 sec
14	15:15:31	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	1 row(s) returned	0.000 sec / 0.000 sec
15	15:15:51	By id SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version [4.0.15.1]	0.000 sec
16	15:16:00	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	1 row(s) returned	0.000 sec / 0.000 sec

e) Write a query to retrieve course information for multiple courses.

```

85 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
86 FROM CoursesInfo
87 ORDER BY COURSE_ID;
88
89 -- By id
90 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
91 FROM CoursesInfo
92 WHERE COURSE_ID = 3;
93
94 -- Or by name
95 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
96 FROM CoursesInfo
97 WHERE COURSE_NAME = 'Algorithms';
98
99 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
100 FROM CoursesInfo
101 WHERE COURSE_ID IN (1,2,3,4)
102 ORDER BY COURSE_ID;
103

```

Result Grid

COURSE_ID	COURSE_NAME	COURSE_INSTRUCTOR_NAME
1	Database Systems	Dr. Green
2	Operating Systems	Dr. White
3	Algorithms	Dr. Black
4	Data Structures	Dr. Gray

CoursesInfo 10 x

Output

#	Time	Action	Message	Duration / Fetch
16	15:16:00	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	1 row(s) returned	0.000 sec / 0.000 sec
17	15:18:52	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	3 row(s) returned	0.000 sec / 0.000 sec
18	15:20:21	INSERT INTO CoursesInfo (COURSE_NAME, COURSE_INSTRUCTOR_NAME) VALUES ('Database Systems', 'Dr. Green')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
19	15:20:49	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	3 row(s) returned	0.000 sec / 0.000 sec
20	15:21:06	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID = 3	4 row(s) returned	0.000 sec / 0.000 sec

f) Test the queries to ensure accurate retrieval of student information (execute the queries and verify the results against the expected output). [the data has been verified]

4) Reporting and Analytics (Using joining queries):

a) Write a query to retrieve the number of students enrolled in each course.

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```

95 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
96 FROM CoursesInfo
97 WHERE COURSE_NAME = 'Algorithms';
98
99 • SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME
100 FROM CoursesInfo
101 WHERE COURSE_ID IN (1,2,3,4)
102 ORDER BY COURSE_ID;
103
104
105 • SELECT
106     c.COURSE_ID,
107     c.COURSE_NAME,
108     COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END) AS enrolled_count
109 FROM CoursesInfo c
110 LEFT JOIN EnrollmentInfo e
111 ON c.COURSE_ID = e.COURSE_ID
112 GROUP BY c.COURSE_ID, c.COURSE_NAME
113 ORDER BY enrolled_count DESC, c.COURSE_ID;
114

```

The results grid shows the following data:

COURSE_ID	COURSE_NAME	enrolled_count
1	Database Systems	2
2	Operating Systems	2
3	Algorithms	2
4	Data Structures	1
5	Database Systems	0

The output pane shows the execution of the query, with a message indicating that 8 row(s) were returned.

b) Write a query to retrieve the list of students enrolled in a specific course.

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```

107 c.COURSE_NAME,
108 COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END) AS enrolled_count
109 FROM CoursesInfo c
110 LEFT JOIN EnrollmentInfo e
111 ON c.COURSE_ID = e.COURSE_ID
112 GROUP BY c.COURSE_ID, c.COURSE_NAME
113 ORDER BY enrolled_count DESC, c.COURSE_ID;
114
115 -- Example: COURSE_ID = 1
116 • SELECT
117     s.STU_ID,
118     s.STU_NAME,
119     c.COURSE_ID,
120     c.COURSE_NAME
121 FROM EnrollmentInfo e
122 JOIN StudentInfo s ON e.STU_ID = s.STU_ID
123 JOIN CoursesInfo c ON e.COURSE_ID = c.COURSE_ID
124 WHERE e.COURSE_ID = 1
125 AND e.ENROLL_STATUS = 'Enrolled'
126 ORDER BY s.STU_NAME;

```

The results grid shows the following data:

STU_ID	STU_NAME	COURSE_ID	COURSE_NAME
1	Alice Johnson	1	Database Systems
3	Charlie Kim	1	Database Systems

The output pane shows the execution of the query, with a message indicating that 2 row(s) were returned.

c) Write a query to retrieve the count of enrolled students for each instructor.

The screenshot shows a SQL IDE with two queries. The first query (lines 116-127) selects student information (STU\_ID, STU\_NAME, COURSE\_ID, COURSE\_NAME) from EnrollmentInfo, joined with StudentInfo and CoursesInfo, where the enrollment status is 'Enrolled'. The second query (lines 128-135) selects the instructor name and the count of enrolled students for each course from CoursesInfo, joined with EnrollmentInfo, grouped by instructor name and ordered by the count in descending order.

**Result Grid:**

COURSE_INSTRUCTOR_NAME	enrolled_count
Dr. Black	2
Dr. Green	2
Dr. White	2
Dr. Gray	1

**Output:**

#	Time	Action	Message	Duration / Fetch
19	15:20:49	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID ...	3 row(s) returned	0.000 sec / 0.000 sec
20	15:21:06	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID ...	4 row(s) returned	0.000 sec / 0.000 sec
21	15:25:18	SELECT c.COURSE_ID, c.COURSE_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END ...	8 row(s) returned	0.000 sec / 0.000 sec
22	15:27:17	SELECT s.STU_ID, s.STU_NAME, c.COURSE_ID, c.COURSE_NAME FROM EnrollmentInfo e JOIN StudentInfo s ...	2 row(s) returned	0.000 sec / 0.000 sec
23	15:28:09	SELECT c.COURSE_INSTRUCTOR_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END ...	4 row(s) returned	0.000 sec / 0.000 sec

d) Write a query to retrieve the list of students who are enrolled in multiple courses.

The screenshot shows a SQL IDE with two queries. The first query (lines 128-135) is the same as the one in the previous screenshot, selecting instructor names and enrollment counts. The second query (lines 137-146) selects student information (STU\_ID, STU\_NAME) from EnrollmentInfo, joined with StudentInfo, where the enrollment status is 'Enrolled' and the student is enrolled in more than one course (HAVING COUNT(\*) >= 2), ordered by the number of courses in descending order.

**Result Grid:**

STU_ID	STU_NAME	enrolled_courses
1	Alice Johnson	2
3	Charlie Kim	2

**Output:**

#	Time	Action	Message	Duration / Fetch
20	15:21:06	SELECT COURSE_ID, COURSE_NAME, COURSE_INSTRUCTOR_NAME FROM CoursesInfo WHERE COURSE_ID ...	4 row(s) returned	0.000 sec / 0.000 sec
21	15:25:18	SELECT c.COURSE_ID, c.COURSE_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END ...	8 row(s) returned	0.000 sec / 0.000 sec
22	15:27:17	SELECT s.STU_ID, s.STU_NAME, c.COURSE_ID, c.COURSE_NAME FROM EnrollmentInfo e JOIN StudentInfo s ...	2 row(s) returned	0.000 sec / 0.000 sec
23	15:28:09	SELECT c.COURSE_INSTRUCTOR_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END ...	4 row(s) returned	0.000 sec / 0.000 sec
24	15:28:43	SELECT s.STU_ID, s.STU_NAME, COUNT(*) AS enrolled_courses FROM EnrollmentInfo e JOIN StudentInfo s ON ...	2 row(s) returned	0.000 sec / 0.000 sec

e) Write a query to retrieve the courses that have the highest number of enrolled students (arranging from highest to lowest).

Limit to 1000 row

My Snippets

```
137 • SELECT
138     s.STU_ID,
139     s.STU_NAME,
140     COUNT(*) AS enrolled_courses
141 FROM EnrollmentInfo e
142 JOIN StudentInfo s ON e.STU_ID = s.STU_ID
143 WHERE e.ENROLL_STATUS = 'Enrolled'
144 GROUP BY s.STU_ID, s.STU_NAME
145 HAVING COUNT(*) >= 2
146 ORDER BY enrolled_courses DESC, s.STU_NAME;
147
148 • SELECT
149     c.COURSE_ID,
150     c.COURSE_NAME,
151     COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END) AS enrolled_count
152 FROM CoursesInfo c
153 LEFT JOIN EnrollmentInfo e
154     ON c.COURSE_ID = e.COURSE_ID
155 GROUP BY c.COURSE_ID, c.COURSE_NAME
156 ORDER BY enrolled_count DESC, c.COURSE_NAME;
```

Result Grid

Filter Rows

Export

Wrap Cell Contents

	COURSE_ID	COURSE_NAME	enrolled_count
▶	3	Algorithms	2
▶	1	Database Systems	2
▶	2	Operating Systems	2
▶	4	Data Structures	1
▶	7	Algorithms	0

Result 15 x

Output

Read Only

Context Help

Snippets

#	Time	Action	Message	Duration / Fetch
21	15:25:18	SELECT c.COURSE_ID, c.COURSE_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END)	8 row(s) returned	0.000 sec / 0.000 sec
22	15:27:17	SELECT s.STU_ID, s.STU_NAME, c.COURSE_ID, c.COURSE_NAME FROM EnrollmentInfo e JOIN StudentInfo s	2 row(s) returned	0.000 sec / 0.000 sec
23	15:28:09	SELECT c.COURSE_ID, c.COURSE_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END)	4 row(s) returned	0.000 sec / 0.000 sec
24	15:28:43	SELECT s.STU_ID, s.STU_NAME, COUNT(*) AS enrolled_courses FROM EnrollmentInfo e JOIN StudentInfo s ON	2 row(s) returned	0.000 sec / 0.000 sec
25	15:29:18	SELECT c.COURSE_ID, c.COURSE_NAME, COUNT(CASE WHEN e.ENROLL_STATUS = 'Enrolled' THEN 1 END)	8 row(s) returned	0.000 sec / 0.000 sec