

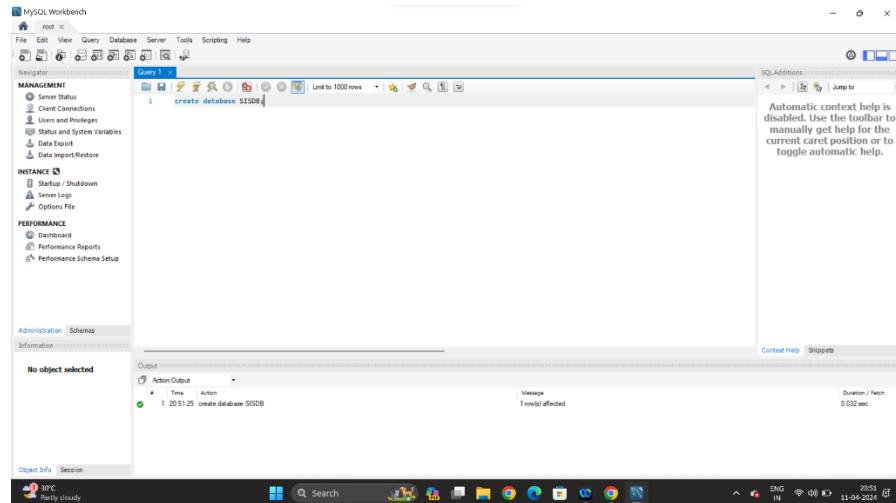
Assignment SQL - Student Information System

Name: Thilak Raaj R A

Batch: Python batch 2

TASK 1 – DATABASE DESIGN:

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

Table-Students:

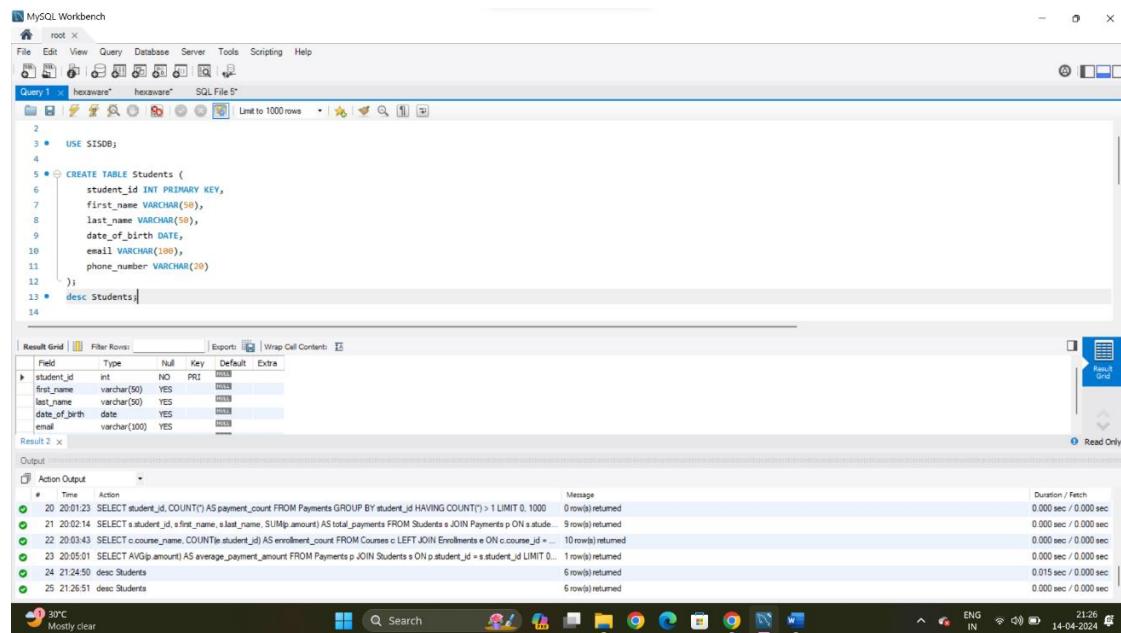


Table-Courses:

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5*
10     email VARCHAR(100),
11     phone_number VARCHAR(20)
12   );
13 • desc Students;
14
15 • CREATE TABLE Courses (
16     course_id INT PRIMARY KEY,
17     course_name VARCHAR(100),
18     credits INT,
19     teacher_id INT,
20     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
21   );
22 • desc Courses;

```

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

Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI		
course_name	varchar(100)	YES			
credits	int	YES			
teacher_id	int	YES	MUL		

Result 3 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
21	20:02:14	SELECT t.student_id, t.first_name, t.last_name, SUM(p.amount) AS total_payments FROM Students t JOIN Payments p ON t.student_id = p.student_id GROUP BY t.student_id	9 row(s) returned	0.000 sec / 0.000 sec
22	20:03:43	SELECT c.course_name, COUNT(e.student_id) AS enrollment_count FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id GROUP BY c.course_id	10 row(s) returned	0.000 sec / 0.000 sec
23	20:05:01	SELECT AVG(p.amount) AS average_payment_amount FROM Payments p JOIN Students s ON p.student_id = s.student_id LIMIT 0, 1	1 row(s) returned	0.000 sec / 0.000 sec
24	21:24:50	desc Students	6 row(s) returned	0.015 sec / 0.000 sec
25	21:26:51	desc Courses	6 row(s) returned	0.000 sec / 0.000 sec
26	21:36:26	desc Courses	4 row(s) returned	0.000 sec / 0.000 sec

30°C Mostly clear 21:36 ENG IN 14-04-2024

Table-Enrollments:

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5*
20     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id);
21   );
22 • desc Courses;
23
24 • CREATE TABLE Enrollments (
25     enrollment_id INT PRIMARY KEY,
26     student_id INT,
27     course_id INT,
28     enrollment_date DATE,
29     FOREIGN KEY (student_id) REFERENCES Students(student_id),
30     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
31   );
32 • desc Enrollments;

```

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

Field	Type	Null	Key	Default	Extra
enrollment_id	int	NO	PRI		
student_id	int	YES	MUL		
course_id	int	YES	MUL		
enrollment_date	date	YES			

Result 4 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
22	20:03:43	SELECT c.course_name, COUNT(e.student_id) AS enrollment_count FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id GROUP BY c.course_id	10 row(s) returned	0.000 sec / 0.000 sec
23	20:05:01	SELECT AVG(p.amount) AS average_payment_amount FROM Payments p JOIN Students s ON p.student_id = s.student_id LIMIT 0, 1	1 row(s) returned	0.000 sec / 0.000 sec
24	21:24:50	desc Students	6 row(s) returned	0.015 sec / 0.000 sec
25	21:26:51	desc Courses	6 row(s) returned	0.000 sec / 0.000 sec
26	21:36:26	desc Courses	4 row(s) returned	0.000 sec / 0.000 sec
27	21:37:29	desc Enrollments	4 row(s) returned	0.000 sec / 0.000 sec

30°C Mostly clear 21:37 ENG IN 14-04-2024

Table-Teachers:

The screenshot shows the MySQL Workbench interface with a query editor containing SQL code. The code creates two tables: 'Teacher' and 'Enrollments'. The 'Teacher' table has columns: teacher_id (INT, primary key), first_name (VARCHAR(50)), last_name (VARCHAR(50)), and email (VARCHAR(100)). The 'Enrollments' table has columns: enrollment_id (INT, primary key), course_id (INT), and student_id (INT). Foreign keys are defined: 'student_id' references 'Students(student_id)' and 'course_id' references 'Courses(course_id)'. The 'Enrollments' table also has an index on the 'course_id' column.

```

29     FOREIGN KEY (student_id) REFERENCES Students(student_id),
30     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
31   );
32 •   desc Enrollments;
33
34 • CREATE TABLE Teacher (
35   teacher_id INT PRIMARY KEY,
36   first_name VARCHAR(50),
37   last_name VARCHAR(50),
38   email VARCHAR(100)
39 );
40 • desc Teacher;
41

```

Field	Type	Null	Key	Default	Extra
teacher_id	int	NO	PRI	NULL	
first_name	varchar(50)	YES		NULL	
last_name	varchar(50)	YES		NULL	
email	varchar(100)	YES		NULL	

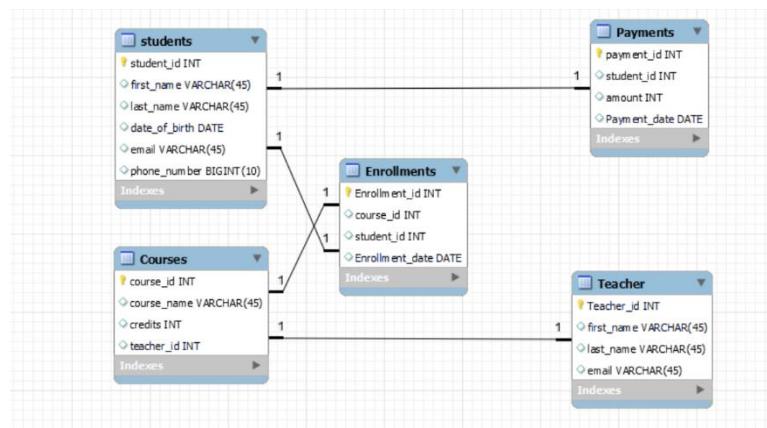
Action Output

- 23 20:05:01 Action SELECT AVG(p.amount) AS average_payment_amount FROM Payments p JOIN Students s ON p.student_id = s.student_id LIMIT 1; 1 row(s) returned 0.000 sec / 0.000 sec
- 24 21:24:50 desc Students 6 row(s) returned 0.015 sec / 0.000 sec
- 25 21:26:51 desc Students 6 row(s) returned 0.000 sec / 0.000 sec
- 26 21:36:26 desc Courses 4 row(s) returned 0.000 sec / 0.000 sec
- 27 21:37:29 desc Enrollments 4 row(s) returned 0.000 sec / 0.000 sec
- 28 21:46:28 desc Teacher 4 row(s) returned 0.000 sec / 0.000 sec

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

Table-Payments:

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



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

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Print, and Database management. The main area has tabs for 'Query 1' (hexaware) and 'hexaware*' (SQL File 5*). A search bar at the top right contains the placeholder 'Search'.

The SQL editor displays a query for inserting data into the 'Students' table:

```
1 • INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number) VALUES
2   (1, 'Rahul', 'Gupta', '2000-01-01', 'rahul.gupta@gmail.com', '123-456-7890'),
3   (2, 'Priya', 'Sharma', '2001-02-02', 'priya.sharma@gmail.com', '987-654-3210'),
4   (3, 'Amit', 'Patel', '1999-03-03', 'amit.patel@gmail.com', '456-789-0123'),
5   (4, 'Sneha', 'Singh', '2000-04-04', 'sneha.singh@gmail.com', '789-012-3456'),
6   (5, 'Neha', 'Yadav', '2001-05-05', 'nehayadav@gmail.com', '012-345-6789'),
7   (6, 'Akash', 'Joshi', '1999-06-06', 'akash.joshi@gmail.com', '234-567-8901'),
8   (7, 'Pooja', 'Gandhi', '2000-07-07', 'pooja.gandhi@gmail.com', '567-890-1234'),
9   (8, 'Rajesh', 'Kumar', '2001-08-08', 'rajesh.kumar@gmail.com', '890-123-4567'),
10  (9, 'Anjali', 'Verma', '1999-09-09', 'anjali.verma@gmail.com', '345-678-9012'),
11  (10, 'Vikram', 'Rao', '2000-10-10', 'vikram.rao@gmail.com', '678-901-2345');
12 • desc Students
```

The results grid shows the structure of the 'Students' table:

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	1	
first_name	varchar(50)	YES			
last_name	varchar(50)	YES			
date_of_birth	date	YES			
email	varchar(100)	YES			

The 'Output' section at the bottom lists recent actions:

- 25 21:26:51 desc Students
- 26 21:36:26 desc Courses
- 27 21:37:29 desc Enrollments
- 28 21:45:28 desc Teacher
- 29 21:47:35 desc Payments
- 30 21:49:15 desc Students

On the far right, there is a vertical sidebar labeled 'Result Grid' with a 'Read Only' button.

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

- MySQL Workbench** window title.
- File Edit View Query Database Server Tools Scripting Help** menu bar.
- Query 1 hexaware hexaware SQL File 5*** tab.
- SQL Editor** content:

```
13
14 • INSERT INTO Courses (course_id, course_name, credits, teacher_id) VALUES
15     (1, 'Mathematics', 3, 1),
16     (2, 'English Literature', 3, 2),
17     (3, 'Computer Science', 4, 3),
18     (4, 'History', 3, 4),
19     (5, 'Physics', 4, 5),
20     (6, 'Biology', 4, 6),
21     (7, 'Chemistry', 4, 7),
22     (8, 'Art', 2, 8),
23     (9, 'Music', 2, 9),
24     (10, 'Physical Education', 2, 10);
25 • desc Courses;
```
- Result Grid** pane showing the structure of the **Courses** table:

Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI	NULL	
course_name	varchar(100)	YES	MUL		
credits	int	YES	MUL		
teacher_id	int	YES	MUL		
- Result 2** pane showing the execution history of the `desc Courses` command.
- Action Output** pane showing the results of the `desc Courses` command.
- System Status** at the bottom left: 29°C, Air: Moderate.
- Taskbar** icons: Start, Search, File Explorer, Task View, Edge, Google Chrome, File Manager, and others.
- System Tray** icons: Network, Battery, Volume, and Date/Time (21:51, 14-04-2024).

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

i. Students:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

```

1 • INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number) VALUES
2     (1, 'Rahul', 'Gupta', '2000-01-01', 'rahul.gupt@gmail.com', '123-456-7890'),
3     (2, 'Priya', 'Sharma', '2001-02-02', 'priya.sharma@gmail.com', '987-654-3210'),
4     (3, 'Amit', 'Patel', '1999-03-03', 'amit.patel@gmail.com', '456-789-0123'),
5     (4, 'Sneha', 'Singh', '2000-04-04', 'sneha.singh@gmail.com', '789-012-3456'),
6     (5, 'Neha', 'Yadav', '2001-05-05', 'neha.yadav@gmail.com', '012-345-6789'),
7     (6, 'Akash', 'Joshi', '1999-06-06', 'akash.joshi@gmail.com', '234-567-8901'),
8     (7, 'Pooja', 'Gandhi', '2000-07-07', 'pooja.gandhi@gmail.com', '567-890-1234'),
9     (8, 'Rajesh', 'Kumar', '2001-08-08', 'rajesh.kumar@gmail.com', '890-123-4567'),
10    (9, 'Anjali', 'Verma', '1999-09-09', 'anjali.verma@gmail.com', '345-678-9012'),
11    (10, 'Vikram', 'Rao', '2000-10-10', 'vikram.rao@gmail.com', '678-901-2345');
12 • select * from Students;
13

```

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

student_id	first_name	last_name	date_of_birth	email	phone_number
2	Priya	Sharma	2001-02-02	priya.sharma@gmail.com	987-654-3210
3	Amit	Patel	1999-03-03	amit.patel@gmail.com	456-789-0123
4	Sneha	Singh	2000-04-04	sneha.singh@gmail.com	789-012-3456
5	Neha	Yadav	2001-05-05	neha.yadav@gmail.com	012-345-6789
6	Akash	Joshi	1999-06-06	akash.joshi@gmail.com	234-567-8901

Students 3 x

Output:

Action Output

Time	Action	Message	Duration / Fetch
27 21:37:29	desc Enrollments	4 row(s) returned	0.000 sec / 0.000 sec
28 21:46:28	desc Teacher	4 row(s) returned	0.000 sec / 0.000 sec
29 21:47:35	desc Payments	4 row(s) returned	0.000 sec / 0.000 sec
30 21:49:15	desc Students	6 row(s) returned	0.000 sec / 0.000 sec
31 21:51:23	desc Courses	4 row(s) returned	0.000 sec / 0.000 sec
32 21:53:01	select * from Students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

28°C Mostly clear | Search | ENG IN 21:53 14-04-2024

ii.Courses:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

```

16     (2, 'English Literature', 3, 2),
17     (3, 'Computer Science', 4, 3),
18     (4, 'History', 3, 4),
19     (5, 'Physics', 4, 5),
20     (6, 'Biology', 4, 6),
21     (7, 'Chemistry', 4, 7),
22     (8, 'Art', 2, 8),
23     (9, 'Music', 2, 9),
24     (10, 'Physical Education', 2, 10);
25 • select * from Courses;
26
27 • INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES
28     (1, 1, 1, '2024-01-01'),

```

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

course_id	course_name	credits	teacher_id
1	Mathematics	3	1
2	English Literature	3	2
3	Computer Science	4	3
4	History	3	4
5	Physics	4	5

Courses 4 x

Output:

Action Output

Time	Action	Message	Duration / Fetch
28 21:46:28	desc Teacher	4 row(s) returned	0.000 sec / 0.000 sec
29 21:47:35	desc Payments	4 row(s) returned	0.000 sec / 0.000 sec
30 21:49:15	desc Students	6 row(s) returned	0.000 sec / 0.000 sec
31 21:51:23	desc Courses	4 row(s) returned	0.000 sec / 0.000 sec
32 21:53:01	select * from Students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
33 21:55:35	select * from Courses LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

29°C Mostly clear | Search | ENG IN 21:55 14-04-2024

iii.Teacher:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5* Limit to 1000 rows

```

38 • INSERT INTO Teacher (teacher_id, first_name, last_name, email) VALUES
39   (1, 'Suresh', 'Sharma', 'suresh.sharma@gmail.com'),
40   (2, 'Neha', 'Patil', 'neha.patil@gmail.com'),
41   (3, 'Rahul', 'Singh', 'rahul.singh@gmail.com'),
42   (4, 'Priya', 'Gupta', 'priya.gupta@gmail.com'),
43   (5, 'Amit', 'Yadav', 'amit.yadav@gmail.com'),
44   (6, 'Sneha', 'Kumar', 'sneha.kumar@gmail.com'),
45   (7, 'Neha', 'Verma', 'neha.verma@gmail.com'),
46   (8, 'Akash', 'Rao', 'akash.rao@gmail.com'),
47   (9, 'Pooja', 'Joshi', 'pooja.joshi@gmail.com'),
48   (10, 'Rajesh', 'Gandhi', 'rajesh.gandhi@gmail.com');
49 • select * from Teacher;

```

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

teacher_id	first_name	last_name	email
1	Suresh	Sharma	suresh.sharma@gmail.com
2	Neha	Patil	neha.patil@gmail.com
3	Rahul	Singh	rahul.singh@gmail.com
4	Priya	Gupta	priya.gupta@gmail.com
5	Amit	Yadav	amit.yadav@gmail.com

Teacher 5 ×

Action Output

- Time Action Message Duration / Fetch
 - 30 21:49:15 desc Students 6 row(s) returned 0.000 sec / 0.000 sec
 - 31 21:51:23 desc Courses 4 row(s) returned 0.000 sec / 0.000 sec
 - 32 21:53:01 select * from Students LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
 - 33 21:55:35 select * from Courses LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
 - 34 21:56:16 select * from Teachers LIMIT 0, 1000 Error Code: 1146: Table 'ssdb.teachers' doesn't exist 0.000 sec / 0.000 sec
 - 35 21:56:33 select * from Teacher LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec

29°C Mostly clear 21:56 ENG IN 14-04-2024

iv. Enrollments:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5* Limit to 1000 rows

```

26
27 • INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES
28   (1, 1, 1, '2024-01-01'),
29   (2, 2, 2, '2024-01-01'),
30   (3, 3, 3, '2024-01-01'),
31   (4, 4, 4, '2024-01-01'),
32   (5, 5, 5, '2024-01-01'),
33   (6, 6, 6, '2024-01-01'),
34   (7, 7, 7, '2024-01-01'),
35   (8, 8, 8, '2024-01-01'),
36   (9, 9, 9, '2024-01-01'),
37   (10, 10, 10, '2024-01-01');
38 • select * from Enrollments;

```

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

enrollment_id	student_id	course_id	enrollment_date
2	2	2	2024-01-01
3	3	3	2024-01-01
4	4	4	2024-01-01
5	5	5	2024-01-01
6	6	6	2024-01-01

Enrollments 6 ×

Action Output

- Time Action Message Duration / Fetch
 - 31 21:51:23 desc Courses 4 row(s) returned 0.000 sec / 0.000 sec
 - 32 21:53:01 select * from Students LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
 - 33 21:55:35 select * from Courses LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
 - 34 21:56:16 select * from Teachers LIMIT 0, 1000 Error Code: 1146: Table 'ssdb.teachers' doesn't exist 0.000 sec / 0.000 sec
 - 35 21:56:33 select * from Teacher LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
 - 36 21:57:32 select * from Enrollments LIMIT 0, 1000 9 row(s) returned 0.000 sec / 0.000 sec

29°C Mostly clear 21:57 ENG IN 14-04-2024

v. Payments:

The screenshot shows the MySQL Workbench interface. In the Query Editor, there are two queries:

```

56 (3, 3, 200.00, '2024-01-17'),
57 (4, 4, 100.00, '2024-01-18'),
58 (5, 5, 150.00, '2024-01-19'),
59 (6, 6, 200.00, '2024-01-20'),
60 (7, 7, 100.00, '2024-01-21'),
61 (8, 8, 150.00, '2024-01-22'),
62 (9, 9, 200.00, '2024-01-23'),
63 (10, 10, 100.00, '2024-01-24');
64 • select * from payments;
65
66 • INSERT INTO Students (student_id,first_name, last_name, date_of_birth, email, phone_number) VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');

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


| payment_id | student_id | amount | payment_date |
|------------|------------|--------|--------------|
| 2          | 2          | 150.00 | 2024-01-16   |
| 3          | 3          | 200.00 | 2024-01-17   |
| 4          | 4          | 100.00 | 2024-01-18   |
| 5          | 5          | 150.00 | 2024-01-19   |
| 6          | 6          | 200.00 | 2024-01-20   |


```

The Result Grid displays the data from the 'payments' table. Below the editor, the Action Output pane shows the execution history of the session.

TASKS 2: SELECT, WHERE, BETWEEN, AND, LIKE:

- 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

The screenshot shows the MySQL Workbench interface. In the Query Editor, there are two queries:

```

56 (3, 3, 200.00, '2024-01-17'),
57 (4, 4, 100.00, '2024-01-18'),
58 (5, 5, 150.00, '2024-01-19'),
59 (6, 6, 200.00, '2024-01-20'),
60 (7, 7, 100.00, '2024-01-21'),
61 (8, 8, 150.00, '2024-01-22'),
62 (9, 9, 200.00, '2024-01-23'),
63 (10, 10, 100.00, '2024-01-24');
64 • select * from payments;
65
66 • INSERT INTO Students (student_id,first_name, last_name, date_of_birth, email, phone_number) VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
67 • select * from Students;

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


| student_id | first_name | last_name | date_of_birth | email                  | phone_number |
|------------|------------|-----------|---------------|------------------------|--------------|
| 8          | Rajesh     | Kumar     | 2001-08-08    | rajesh.kumar@gmail.com | 890-123-4567 |
| 9          | Anjali     | Varma     | 1999-09-09    | anjali.verma@gmail.com | 345-678-9012 |
| 10         | Vikram     | Rao       | 2000-10-10    | vikram.rao@gmail.com   | 678-901-2345 |
| 11         | John       | Doe       | 1995-08-15    | john.doe@example.com   | 1234567890   |


```

The Result Grid displays the data from the 'Students' table. Below the editor, the Action Output pane shows the execution history of the session.

- 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.

MySQL Workbench

```

65
66 • INSERT INTO Students (student_id,first_name, last_name, date_of_birth, email, phone_number) VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
67 • select * from Students;
68
69
70 • INSERT INTO Enrollments (enrollment_id,student_id, course_id, enrollment_date)
VALUES (11, 1, 1, CURDATE());
71 • select * from Enrollments;
72
73
74 • UPDATE Teacher
SET email = 'rahul.singhraj@gmail.com'
WHERE teacher_id = 3;
75
76
77

```

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

enrollment_id	student_id	course_id	enrollment_date
6	6	6	2024-01-01
7	7	7	2024-01-01
8	8	8	2024-01-01
9	9	9	2024-01-01
10	10	10	2024-01-01

Enrollments 9 x

Action Output

#	Time	Action	Message	Duration / Fetch
34	21:56:16	select * from Teachers LIMIT 0, 1000	Error Code: 1146. Table 'ssdb.teachers' doesn't exist	0.000 sec
35	21:56:33	select * from Teacher LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
36	21:57:32	select * from Enrollments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
37	21:58:16	select * from payments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
38	21:59:44	select * from Students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
39	22:00:41	select * from Enrollments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec

Output

29°C Mostly clear

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

MySQL Workbench

```

65
66 • INSERT INTO Students (student_id,first_name, last_name, date_of_birth, email, phone_number) VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
67 • select * from Students;
68
69
70 • INSERT INTO Enrollments (enrollment_id,student_id, course_id, enrollment_date)
VALUES (11, 1, 1, CURDATE());
71 • select * from Enrollments;
72 • select * from Teachers;
73
74 • UPDATE Teacher
SET email = 'rahul.singhraj@gmail.com'
WHERE teacher_id = 3;
75
76
77 • select * from Teacher;

```

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

teacher_id	first_name	last_name	email
1	Suresh	Sharma	suresh.sharma@gmail.com
2	Neha	Patil	neha.patil@gmail.com
3	Rahul	Singh	rahul.singhraj@gmail.com
4	Priya	Gupta	priya.gupta@gmail.com
5	Amit	Yadav	amit.yadav@gmail.com

Teacher 10 x

Action Output

#	Time	Action	Message	Duration / Fetch
35	21:56:33	select * from Teacher LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
36	21:57:32	select * from Enrollments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
37	21:58:16	select * from payments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
38	21:59:44	select * from Students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
39	22:00:41	select * from Enrollments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
40	22:02:11	select * from Teacher LIMIT 0, 1000	10 row(s) returned	0.015 sec / 0.000 sec

Output

29°C Mostly clear

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.

MySQL Workbench

root

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

65
66 • INSERT INTO Students (student_id,first_name, last_name, date_of_birth, email, phone_number) VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
67 • select * from Students;
68
69
70 • INSERT INTO Enrollments (enrollment_id,student_id, course_id, enrollment_date)
71 VALUES (11, 1, 1, CURDATE());
72 • select * from Enrollments;
73
74 • UPDATE Teacher
75 SET email = 'rahul.singhraj@gmail.com'
76 WHERE teacher_id = 3;
77 • select * from Teacher;
78
79 • DELETE FROM Enrollments
80 WHERE student_id = 1
81 AND course_id = 1;
82 • select * from Enrollments;
83
84 • SHOW CREATE TABLE

Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Content | Result Grid | Form Editor | Revert

enrollment_id	student_id	course_id	enrollment_date
2	2	2	2024-01-01
3	3	3	2024-01-01
4	4	4	2024-01-01
5	5	5	2024-01-01
6	6	6	2024-01-01
7	7	7	2024-01-01
8	8	8	2024-01-01
9	9	9	2024-01-01
10	10	10	2024-01-01

Enrollments 11 x

29°C Mostly clear

Search

22:03 14-04-2024

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

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

1 Limit to 1000 rows

```
71 VALUES (11, 1, 1, CURDATE());
72 • select * from Enrollments;
73
74 • UPDATE Teacher
75     SET email = 'rahul.singhraj@gmail.com'
76     WHERE teacher_id = 3;
77 • select * from Teacher;
78
79 • DELETE FROM Enrollments
80     WHERE student_id = 1
81     AND course_id = 1;
82 • select * from Enrollments;
83
84 • UPDATE Courses
85     SET teacher_id = 1
86     WHERE course_id = 1;
87 • select * from Courses;
88
89 • delete from Enrollments
90 where student_id = 1;
```

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

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

Courses 12 x

29°C Mostly clear

Search

22:04 14-04-2024

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.

MySQL Workbench

root X

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

Limit to 1000 rows

```
85 SET teacher_id = 1;
86 WHERE course_id = 1;
87 • select * from Courses;
88
89 • delete from Enrollments
90 where student_id = 1;
91 • delete from Students
92 where student_id = 1;
93 • delete from Payments
94 where student_id = 1;
95
96 • select * from Students;
97
98 • update Payments
99 set amount = 150.00
100 where payment_id = 1;
101
102
103
```

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

student_id	first_name	last_name	date_of_birth	email	phone_number
4	Sneha	Singh	2000-04-04	sneha.singh@gmail.com	789-012-3456
5	Neha	Yadav	2001-05-05	neha.yadav@gmail.com	012-345-6789
6	Akash	Joshi	1999-06-06	akash.joshi@gmail.com	234-567-8901
7	Pooja	Gandhi	2000-07-07	pooja.gandhi@gmail.com	567-890-1234
8	Rajesh	Kumar	2001-08-08	rajesh.kumar@gmail.com	890-123-4567
9	Anjali	Verma	1999-09-09	anjali.verma@gmail.com	345-678-9012
10	Vikram	Rao	2000-10-10	vikram.rao@gmail.com	678-901-2345
11	John	Doe	1995-08-15	john.doe@example.com	1234567890

Students 13 X

Apply

28°C Haze ENG IN 22:39 14-04-2024

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

Limit to 1000 rows

```
85 SET teacher_id = 1;
86 WHERE course_id = 1;
87 • select * from Courses;
88
89 • delete from Enrollments
90 where student_id = 1;
91 • delete from Students
92 where student_id = 1;
93 • delete from Payments
94 where student_id = 1;
95
96 • select * from Students;
97
98 • update Payments
99 set amount = 150.00
100 where payment_id = 1;
101
102
103
```

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

student_id	first_name	last_name	date_of_birth	email	phone_number
4	Sneha	Singh	2000-04-04	sneha.singh@gmail.com	789-012-3456
5	Neha	Yadav	2001-05-05	neha.yadav@gmail.com	012-345-6789
6	Akash	Joshi	1999-06-06	akash.joshi@gmail.com	234-567-8901
7	Pooja	Gandhi	2000-07-07	pooja.gandhi@gmail.com	567-890-1234
8	Rajesh	Kumar	2001-08-08	rajesh.kumar@gmail.com	890-123-4567
9	Anjali	Verma	1999-09-09	anjali.verma@gmail.com	345-678-9012
10	Vikram	Rao	2000-10-10	vikram.rao@gmail.com	678-901-2345
11	John	Doe	1995-08-15	john.doe@example.com	1234567890

Students 13 X

Apply

28°C Haze ENG IN 22:39 14-04-2024

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 the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The main window has a toolbar with various icons for file operations, database management, and search. A query editor window titled "hexaware" is open, displaying the following SQL script:

```
88
89 • delete from Enrollments
90 where student_id = 1;
91 • delete from Students
92 where student_id = 1;
93 • delete from Payments
94 where student_id = 1;
95
96 • select * from Students;
97
98 • update Payments
99 set amount = 150.00
100 where payment_id = 1;
101 • select * from Payments;]
```

The result grid below shows the data for the Payments table:

payment_id	student_id	amount	payment_date
2	2	150.00	2024-01-16
3	3	200.00	2024-01-17
4	4	100.00	2024-01-18
5	5	150.00	2024-01-19
6	6	200.00	2024-01-20
7	7	100.00	2024-01-21
8	8	150.00	2024-01-22
9	9	200.00	2024-01-23
10	10	100.00	2024-01-24

A sidebar on the right contains tabs for Result Grid, Form Editor, and Result Grid. The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray.

TASK 3 . AGGREGATE FUNCTIONS, HAVING, ORDER BY, GROUP BY 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.

The screenshot shows the MySQL Workbench interface. In the Query Editor (Query 1), the following SQL code is written:

```

1 • SELECT SUM(amount) AS total_payments
2   FROM Payments
3  WHERE student_id = 2;
4

```

The Result Grid shows the output of the query:

total_payments
150.00

Below the result grid, the Query History (Result 2) displays the following log entries:

Action	Time	Action	Message	Duration / Fetch
31	12:28:42	delete from Payments where student_id = 1	1 row(s) affected	0.015 sec
32	12:28:45	delete from Students where student_id = 1	1 row(s) affected	0.000 sec
33	12:28:54	select * from Students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
34	12:30:56	update Payments set amount = 150.00 where payment_id = 1	0 row(s) affected Rows matched: 0	0.015 sec
35	12:35:49	SELECT SUM(amount) AS total_payments FROM Payments WHERE student_id = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
36	12:35:58	SELECT SUM(amount) AS total_payments FROM Payments WHERE student_id = 2 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

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.

The screenshot shows the MySQL Workbench interface. In the Query Editor (Query 1), the following SQL code is written:

```

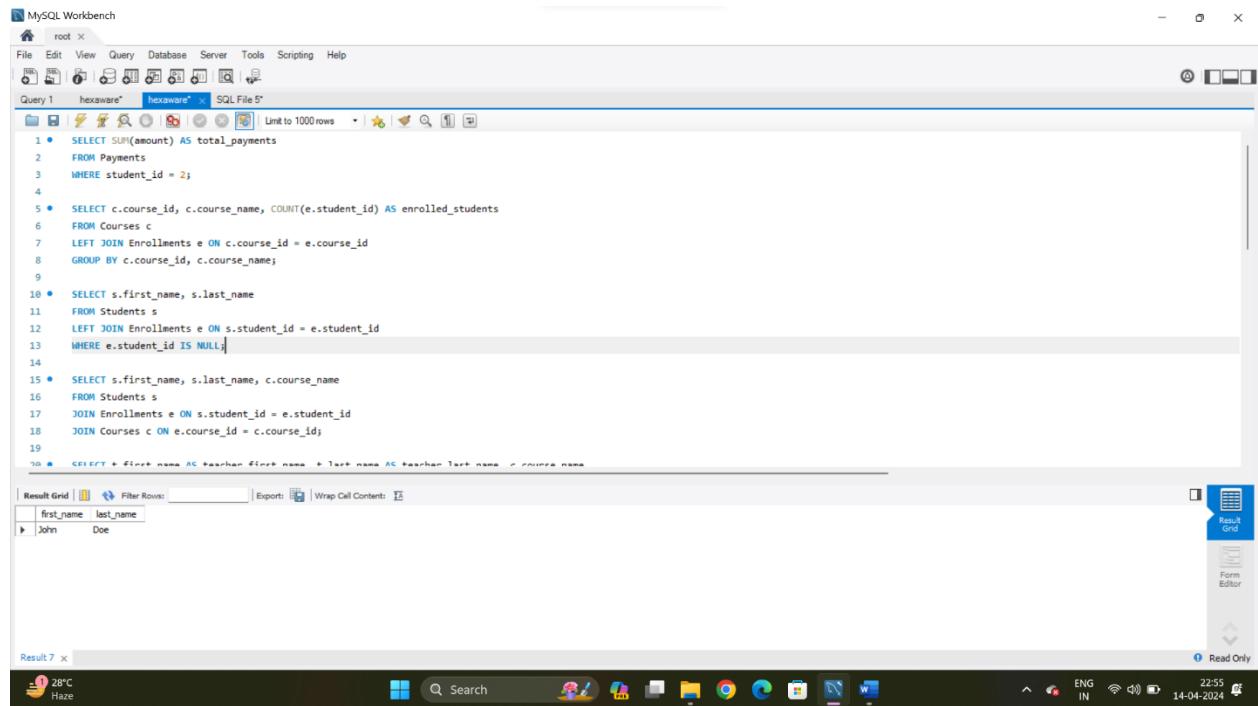
1 • SELECT SUM(amount) AS total_payments
2   FROM Payments
3  WHERE student_id = 2;
4
5 • SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students
6   FROM Courses c
7  LEFT JOIN Enrollments e ON c.course_id = e.course_id
8  GROUP BY c.course_id, c.course_name;
9
10 • SELECT s.first_name, s.last_name
11   FROM Students s
12  LEFT JOIN Enrollments e ON s.student_id = e.student_id
13 WHERE e.student_id IS NULL;
14
15 • SELECT s.first_name, s.last_name, c.course_name
16   FROM Students s
17  JOIN Enrollments e ON s.student_id = e.student_id
18  JOIN Courses c ON e.course_id = c.course_id;
19
20 • SELECT * first_name AS teacher_first_name + last_name AS teacher_last_name, c.course_name

```

The Result Grid shows the output of the query:

course_id	course_name	enrolled_students
1	Mathematics	0
2	English Literature	1
3	Computer Science	1
4	History	1
5	Physics	1
6	Biology	1
7	Chemistry	1
8	Arts	1
9	Music	1

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.



```

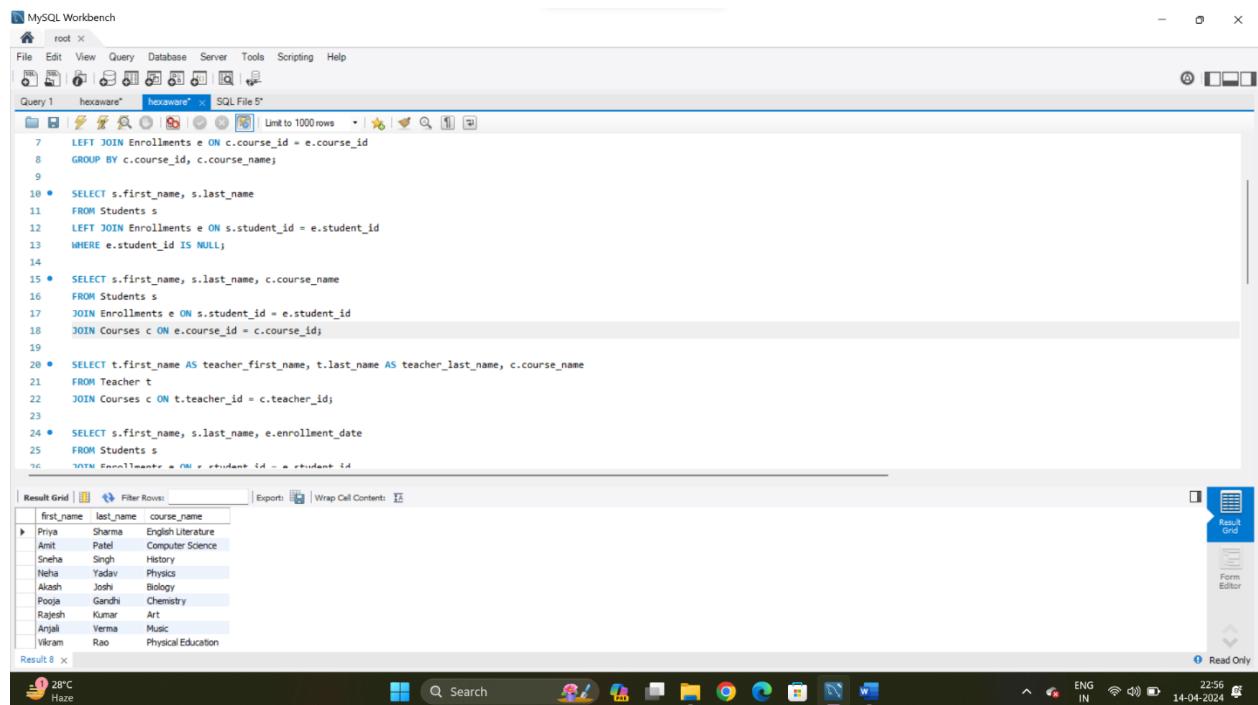
MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5*
1 • SELECT SUM(amount) AS total_payments
2 FROM Payments
3 WHERE student_id = 2;
4
5 • SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students
6 FROM Courses c
7 LEFT JOIN Enrollments e ON c.course_id = e.course_id
8 GROUP BY c.course_id, c.course_name;
9
10 • SELECT s.first_name, s.last_name
11 FROM Students s
12 LEFT JOIN Enrollments e ON s.student_id = e.student_id
13 WHERE e.student_id IS NULL;
14
15 • SELECT s.first_name, s.last_name, c.course_name
16 FROM Students s
17 JOIN Enrollments e ON s.student_id = e.student_id
18 JOIN Courses c ON e.course_id = c.course_id;
19
20 • SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name
21 FROM Teacher t
22 JOIN Courses c ON t.teacher_id = c.teacher_id;
23
24 • SELECT s.first_name, s.last_name, e.enrollment_date
25 FROM Students s
26 WHERE Enrollments.e.student_id = s.student_id

```

The screenshot shows the MySQL Workbench interface with a query editor containing the provided SQL code. The results pane displays a single row of data:

first_name	last_name
John	Doe

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.



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5*
7 LEFT JOIN Enrollments e ON c.course_id = e.course_id
8 GROUP BY c.course_id, c.course_name;
9
10 • SELECT s.first_name, s.last_name
11 FROM Students s
12 LEFT JOIN Enrollments e ON s.student_id = e.student_id
13 WHERE e.student_id IS NULL;
14
15 • SELECT s.first_name, s.last_name, c.course_name
16 FROM Students s
17 JOIN Enrollments e ON s.student_id = e.student_id
18 JOIN Courses c ON e.course_id = c.course_id;
19
20 • SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name
21 FROM Teacher t
22 JOIN Courses c ON t.teacher_id = c.teacher_id;
23
24 • SELECT s.first_name, s.last_name, e.enrollment_date
25 FROM Students s
26 WHERE Enrollments.e.student_id = s.student_id

```

The screenshot shows the MySQL Workbench interface with a query editor containing the provided SQL code. The results pane displays a table of student names and their enrolled courses:

first_name	last_name	course_name
Priva	Sharma	English Literature
Amit	Patel	Computer Science
Sneha	Singh	History
Neha	Yadav	Physics
Akash	Joshi	Biology
Pooja	Gandhi	Chemistry
Rajesh	Kumar	Art
Anjali	Verma	Music
Vikram	Rao	Physical Education

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

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

```
13 WHERE e.student_id IS NULL;
14
15 • SELECT s.first_name, s.last_name, c.course_name
16   FROM Students s
17  JOIN Enrollments e ON s.student_id = e.student_id
18  JOIN Courses c ON e.course_id = c.course_id;
19
20 • SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name
21   FROM Teacher t
22  JOIN Courses c ON t.teacher_id = c.teacher_id;
23
24 • SELECT s.first_name, s.last_name, e.enrollment_date
25   FROM Students s
26  JOIN Enrollments e ON s.student_id = e.student_id
27  JOIN Courses c ON e.course_id = c.course_id
28 WHERE c.course_name = 'Art';
29
30 • SELECT s.first_name, s.last_name
31   FROM Students s
32  INSERT INTO Payments n ON e.student_id = n.student_id;
```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result 9 ×

teacher_first_name	teacher_last_name	course_name
Suresh	Sharma	Mathematics
Neha	Patil	English Literature
Rahul	Singh	Computer Science
Priya	Gupta	History
Amit	Yadav	Physics
Sneha	Kumar	Biology
Neha	Verma	Chemistry
Akash	Rao	Art
Pooja	Joshi	Music

Result 9 ×

28°C Haze ENG IN 22:58 14-04-2024

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.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Query 1 hexaware* hexaware* SQL File 5*

13 WHERE e.student_id IS NULL;

14

15 • SELECT s.first_name, s.last_name, c.course_name

16 FROM Students s

17 JOIN Enrollments e ON s.student_id = e.student_id

18 JOIN Courses c ON e.course_id = c.course_id;

19

20 • SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name

21 FROM Teacher t

22 JOIN Courses c ON t.teacher_id = c.teacher_id;

23

24 • SELECT s.first_name, s.last_name, e.enrollment_date

25 FROM Students s

26 JOIN Enrollments e ON s.student_id = e.student_id

27 JOIN Courses c ON e.course_id = c.course_id

28 WHERE c.course_name = 'Art';

29

30 • SELECT s.first_name, s.last_name

31 FROM Students s

32 INSERT INTO Enrollments (student_id, course_id)

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

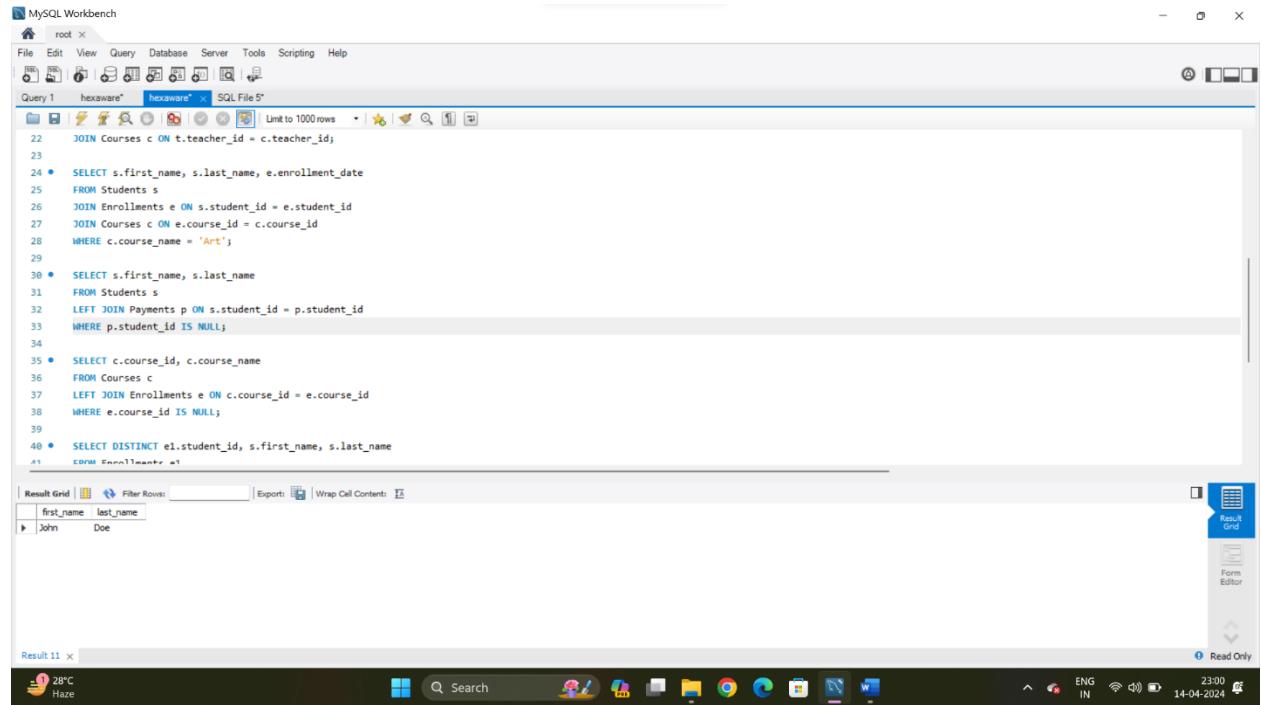
first_name	last_name	enrollment_date
Rajesh	Kumar	2024-01-01

Result 10 x

28°C Haze ENG IN 22:59 14-04-2024

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.



The screenshot shows the MySQL Workbench interface. The query editor window contains the following SQL code:

```

22 JOIN Courses c ON t.teacher_id = c.teacher_id;
23
24 •   SELECT s.first_name, s.last_name, e.enrollment_date
25     FROM Students s
26   JOIN Enrollments e ON s.student_id = e.student_id
27   JOIN Courses c ON e.course_id = c.course_id
28 WHERE c.course_name = 'Art';
29
30 •   SELECT s.first_name, s.last_name
31     FROM Students s
32   LEFT JOIN Payments p ON s.student_id = p.student_id
33 WHERE p.student_id IS NULL;
34
35 •   SELECT c.course_id, c.course_name
36     FROM Courses c
37   LEFT JOIN Enrollments e ON c.course_id = e.course_id
38 WHERE e.course_id IS NULL;
39
40 •   SELECT DISTINCT el.student_id, s.first_name, s.last_name
41   FROM Enrollments el

```

The results grid shows the output of the query, which is a single row:

first_name	last_name
John	Doe

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.

MySQL Workbench Screenshot:

```

22 JOIN Courses c ON t.teacher_id = c.teacher_id;
23
24 •   SELECT s.first_name, s.last_name, e.enrollment_date
25     FROM Students s
26   JOIN Enrollments e ON s.student_id = e.student_id
27   JOIN Courses c ON e.course_id = c.course_id
28   WHERE c.course_name = 'Art';
29
30 •   SELECT s.first_name, s.last_name
31     FROM Students s
32   LEFT JOIN Payments p ON s.student_id = p.student_id
33   WHERE p.student_id IS NULL;
34
35 •   SELECT c.course_id, c.course_name
36     FROM Courses c
37   LEFT JOIN Enrollments e ON c.course_id = e.course_id
38   WHERE e.course_id IS NULL;
39
40 •   SELECT DISTINCT el.student_id, s.first_name, s.last_name
41   FROM Enrollment e1

```

The Result Grid shows one row:

course_id	course_name
1	Mathematics

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.

MySQL Workbench Screenshot:

```

27 JOIN Courses c ON e.course_id = c.course_id
28 WHERE c.course_name = 'Art';
29
30 •   SELECT s.first_name, s.last_name
31     FROM Students s
32   LEFT JOIN Payments p ON s.student_id = p.student_id
33   WHERE p.student_id IS NULL;
34
35 •   SELECT c.course_id, c.course_name
36     FROM Courses c
37   LEFT JOIN Enrollments e ON c.course_id = e.course_id
38   WHERE e.course_id IS NULL;
39
40 •   SELECT DISTINCT e1.student_id, s.first_name, s.last_name
41     FROM Enrollments e1
42   JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id <> e2.course_id
43   JOIN Students s ON e1.student_id = s.student_id;
44
45 •   SELECT t.teacher_id, t.first_name, t.last_name
46   FROM Teacher t

```

The Result Grid shows one row:

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

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.

```

30 •   SELECT s.first_name, s.last_name
31     FROM Students s
32   LEFT JOIN Payments p ON s.student_id = p.student_id
33 WHERE p.student_id IS NULL;
34
35 •   SELECT c.course_id, c.course_name
36     FROM Courses c
37   LEFT JOIN Enrollments e ON c.course_id = e.course_id
38 WHERE e.course_id IS NULL;
39
40 •   SELECT DISTINCT e1.student_id, s.first_name, s.last_name
41     FROM Enrollments e1
42   JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id <> e2.course_id
43   JOIN Students s ON e1.student_id = s.student_id;
44
45 •   SELECT t.teacher_id, t.first_name, t.last_name
46     FROM Teacher t
47   LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
48 WHERE c.teacher_id IS NULL;
49

```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Contents:

teacher_id	first_name	last_name

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.

```

1 •   SELECT AVG(enrollment_count) AS average_enrollment_count
2
3   FROM (
4     SELECT COUNT(student_id) AS enrollment_count
5       FROM Enrollments
6      GROUP BY course_id
7   ) AS course_enrollment_counts;
8
9 •   SELECT s.student_id, s.first_name, s.last_name, p.amount AS highest_payment
10    FROM Students s
11   JOIN Payments p ON s.student_id = p.student_id
12 WHERE p.amount = (
13   SELECT MAX(amount)
14     FROM Payments
15 );
16
17 •   SELECT c.course_id, c.course_name, COUNT(*) AS enrollment_count
18     FROM Courses c
19   JOIN Enrollments e ON c.course_id = e.course_id
20   GROUP BY c.course_id, c.course_name
21 HAVING COUNT(*) = 1;

```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Contents:

average_enrollment_count
1.0000

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.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5* x
1 • SELECT AVG(enrollment_count) AS average_enrollment_count
2   FROM (
3     SELECT COUNT(student_id) AS enrollment_count
4       FROM Enrollments
5      GROUP BY course_id
6   ) AS course_enrollment_counts;
7
8 • SELECT s.student_id, s.first_name, s.last_name, p.amount AS highest_payment
9   FROM Students s
10  JOIN Payments p ON s.student_id = p.student_id
11 WHERE p.amount = (
12   SELECT MAX(amount)
13     FROM Payments
14 )
15
16 • SELECT c.course_id, c.course_name, COUNT(*) AS enrollment_count
17   FROM Courses c
18  JOIN Enrollments e ON c.course_id = e.course_id
19  GROUP BY c.course_id, c.course_name
20 HAVING COUNT(*) = /
21
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
student_id first_name last_name highest_payment
3 Amt Patel 200.00
6 Akash Joshi 200.00
9 Anjali Verma 200.00

```

Result 14 x Read Only

28°C Haze 23:09 14-04-2024

3. 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
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5* x
13   FROM Payments
14 );
15
16 • SELECT c.course_id, c.course_name, COUNT(*) AS enrollment_count
17   FROM Courses c
18  JOIN Enrollments e ON c.course_id = e.course_id
19  GROUP BY c.course_id, c.course_name
20 HAVING COUNT(*) = (
21   SELECT MAX(enrollment_count)
22     FROM (
23     SELECT COUNT(*) AS enrollment_count
24       FROM Enrollments
25      GROUP BY course_id
26   ) AS max_enrollment_count
27 )
28
29 • SELECT s.student_id, s.first_name, s.last_name
30   FROM Students s
31 WHERE (
32   SELECT COUNT(DISTINCT course_id)
33
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
course_id course_name enrollment_count
2 English Literature 1
3 Computer Science 1
4 History 1
5 Physics 1
6 Biology 1
7 Chemistry 1
8 Art 1
9 Music 1
10 Physical Education 1

```

Result 15 x Read Only

28°C Haze 23:09 14-04-2024

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

MySQL Workbench - hexaware - hexaware - SQL File 5

```

22 FROM (
23     SELECT COUNT(*) AS enrollment_count
24     FROM Enrollments
25     GROUP BY course_id
26 ) AS max_enrollment_count
27 );
28
29 •   SELECT s.student_id, s.first_name, s.last_name
30   FROM Students s
31 WHERE (
32     SELECT COUNT(DISTINCT course_id)
33     FROM Enrollments
34 ) =
35     (SELECT COUNT(DISTINCT course_id)
36     FROM Enrollments e
37     WHERE e.student_id = s.student_id
38 );
39
40 •   SELECT s.student_id, s.first_name, s.last_name
41   FROM Students s

```

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

student_id	first_name	last_name

Students 16 x

28°C Haze 23:11 ENG IN 14-04-2024

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

MySQL Workbench - hexaware - hexaware - SQL File 5

```

37 WHERE e.student_id = s.student_id
38 );
39
40 •   SELECT s.student_id, s.first_name, s.last_name
41   FROM Students s
42 WHERE (
43     SELECT COUNT(DISTINCT course_id)
44     FROM Enrollments
45 ) =
46     (SELECT COUNT(DISTINCT course_id)
47     FROM Enrollments e
48     WHERE e.student_id = s.student_id
49 );
50
51 •   SELECT AVG(age) AS average_age
52   FROM (
53     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
54     FROM Students
55   ) AS student_ages;
56

```

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

student_id	first_name	last_name

Students 17 x

28°C Haze 23:11 ENG IN 14-04-2024

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

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5* x
46   SELECT COUNT(DISTINCT course_id)
47     FROM Enrollments e
48    WHERE e.student_id = s.student_id
49  }
50
51 •   SELECT AVG(age) AS average_age
52   FROM (
53     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
54       FROM Students
55   ) AS student_ages;
56
57 •   SELECT c.course_id, c.course_name
58     FROM Courses c
59   WHERE NOT EXISTS (
60     SELECT 1
61       FROM Enrollments e
62      WHERE e.course_id = c.course_id
63   }
64
65 •   SELECT e.student_id, e.course_id, COUNT(e.student_id) AS total_enrolments
66
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
average_age
▶ 23.8000
Result 18 x
28°C Haze

```

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

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Query 1 hexaware* hexaware* SQL File 5* x
46   SELECT COUNT(DISTINCT course_id)
47     FROM Enrollments e
48    WHERE e.student_id = s.student_id
49  }
50
51 •   SELECT AVG(age) AS average_age
52   FROM (
53     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
54       FROM Students
55   ) AS student_ages;
56
57 •   SELECT c.course_id, c.course_name
58     FROM Courses c
59   WHERE NOT EXISTS (
60     SELECT 1
61       FROM Enrollments e
62      WHERE e.course_id = c.course_id
63   }
64
65 •   SELECT e.student_id, e.course_id, COUNT(e.student_id) AS total_enrolments
66
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |
course_id course_name
▶ 1 Mathematics
Courses 19 x
28°C Haze

```

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

MySQL Workbench

```

52   FROM (
53     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
54     FROM Students
55   ) AS student_ages;
56
57 •   SELECT c.course_id, c.course_name
58     FROM Courses c
59   WHERE NOT EXISTS (
60     SELECT 1
61     FROM Enrollments e
62     WHERE e.course_id = c.course_id
63   );
64
65 •   SELECT e.student_id, e.course_id, SUM(p.amount) AS total_payments
66     FROM Enrollments e
67     JOIN Payments p ON e.student_id = p.student_id
68     GROUP BY e.student_id, e.course_id;
69
70 •   SELECT student_id, COUNT(*) AS payment_count
71   FROM Enrollments

```

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

student_id	course_id	total_payments
2	2	150.00
3	3	200.00
4	4	100.00
5	5	150.00
6	6	200.00
7	7	100.00
8	8	150.00
9	9	200.00
10	10	100.00

Result 20 x

28°C Haze 23:14 ENG IN 14-04-2024

9. 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

```

61   FROM Enrollments e
62   WHERE e.course_id = c.course_id
63 );
64
65 •   SELECT e.student_id, e.course_id, SUM(p.amount) AS total_payments
66     FROM Enrollments e
67     JOIN Payments p ON e.student_id = p.student_id
68     GROUP BY e.student_id, e.course_id;
69
70 •   SELECT student_id, COUNT(*) AS payment_count
71     FROM Payments
72     GROUP BY student_id
73     HAVING COUNT(*) > 1;
74
75 •   SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
76     FROM Students s
77     JOIN Payments p ON s.student_id = p.student_id
78     GROUP BY s.student_id, s.first_name, s.last_name;
79
80 •   SELECT s.student_id, COUNT(*) AS enrollment_count
81     FROM Students s
82     JOIN Enrollments e ON s.student_id = e.student_id
83     GROUP BY s.student_id
84     HAVING COUNT(*) > 1;

```

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

student_id	payment_count
1	2
2	2
3	2
4	2
5	2
6	2
7	2
8	2
9	2
10	2

Result 21 x

28°C Haze 23:16 ENG IN 14-04-2024

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.

MySQL Workbench - root

```

61     FROM Enrollments e
62     WHERE e.course_id = c.course_id
63   );
64
65 •  SELECT e.student_id, e.course_id, SUM(p.amount) AS total_payments
66   FROM Enrollments e
67   JOIN Payments p ON e.student_id = p.student_id
68   GROUP BY e.student_id, e.course_id;
69
70 •  SELECT student_id, COUNT(*) AS payment_count
71   FROM Payments
72   GROUP BY student_id
73   HAVING COUNT(*) > 1;
74
75 •  SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
76   FROM Students s
77   JOIN Payments p ON s.student_id = p.student_id
78   GROUP BY s.student_id, s.first_name, s.last_name;
79
80 •  SELECT c.course_name, COUNT(e.student_id) AS enrollment_count

```

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

student_id	first_name	last_name	total_payments
2	Priya	Sharma	150.00
3	Amit	Patel	200.00
4	Sneha	Singh	100.00
5	Neha	Yadav	150.00
6	Akash	Joshi	200.00
7	Pooja	Gandhi	100.00
8	Rajesh	Kumar	150.00
9	Anjali	Verma	200.00
10	Vikram	Rao	100.00

Result 22 x

EUR/INR -0.51% 23:17 ENG IN 14-04-2024

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.

MySQL Workbench - root

```

66     FROM Enrollments e
67   JOIN Payments p ON e.student_id = p.student_id
68   GROUP BY e.student_id, e.course_id;
69
70 •  SELECT student_id, COUNT(*) AS payment_count
71   FROM Payments
72   GROUP BY student_id
73   HAVING COUNT(*) > 1;
74
75 •  SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
76   FROM Students s
77   JOIN Payments p ON s.student_id = p.student_id
78   GROUP BY s.student_id, s.first_name, s.last_name;
79
80 •  SELECT c.course_name, COUNT(e.student_id) AS enrollment_count
81   FROM Courses c
82   LEFT JOIN Enrollments e ON c.course_id = e.course_id
83   GROUP BY c.course_name;
84
85 •  SELECT AVG(amount) AS average_payment_amount

```

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

course_name	enrollment_count
Mathematics	0
English Literature	1
Computer Science	1
History	1
Physics	1
Biology	1
Chemistry	1
Art	1
Music	1

Result 23 x

EUR/INR -0.51% 23:18 ENG IN 14-04-2024

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.

MySQL Workbench

```

66 FROM Enrollments e
67 JOIN Payments p ON e.student_id = p.student_id
68 GROUP BY e.student_id, e.course_id;
69
70 • SELECT student_id, COUNT(*) AS payment_count
71 FROM Payments
72 GROUP BY student_id
73 HAVING COUNT(*) > 1;
74
75 • SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
76 FROM Students s
77 JOIN Payments p ON s.student_id = p.student_id
78 GROUP BY s.student_id, s.first_name, s.last_name;
79
80 • SELECT c.course_name, COUNT(e.student_id) AS enrollment_count
81 FROM Courses c
82 LEFT JOIN Enrollments e ON c.course_id = e.course_id
83 GROUP BY c.course_name;
84
85 • SELECT AVG(p.amount) AS average_payment_amount
86 FROM Payments p
87 JOIN Students s ON p.student_id = s.student_id;
88

```

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

course_name	enrollment_count
Mathematics	0
English Literature	1
Computer Science	1
History	1
Physics	1
Biology	1
Chemistry	1
Art	1
Music	1

Result 23 x

EUR/INR -0.51% 23:18 ENG IN 14-04-2024

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.

MySQL Workbench

```

66
67 • SELECT student_id, COUNT(*) AS payment_count
68 FROM Payments
69 GROUP BY student_id
70 HAVING COUNT(*) > 1;
71
72 • SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
73 FROM Students s
74 JOIN Payments p ON s.student_id = p.student_id
75 GROUP BY s.student_id, s.first_name, s.last_name;
76
77 • SELECT c.course_name, COUNT(e.student_id) AS enrollment_count
78 FROM Courses c
79 LEFT JOIN Enrollments e ON c.course_id = e.course_id
80 GROUP BY c.course_name;
81
82 • SELECT AVG(p.amount) AS average_payment_amount
83 FROM Payments p
84 JOIN Students s ON p.student_id = s.student_id;
85

```

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

average_payment_amount
150.000000

Result 24 x

EUR/INR -0.51% 23:19 ENG IN 14-04-2024