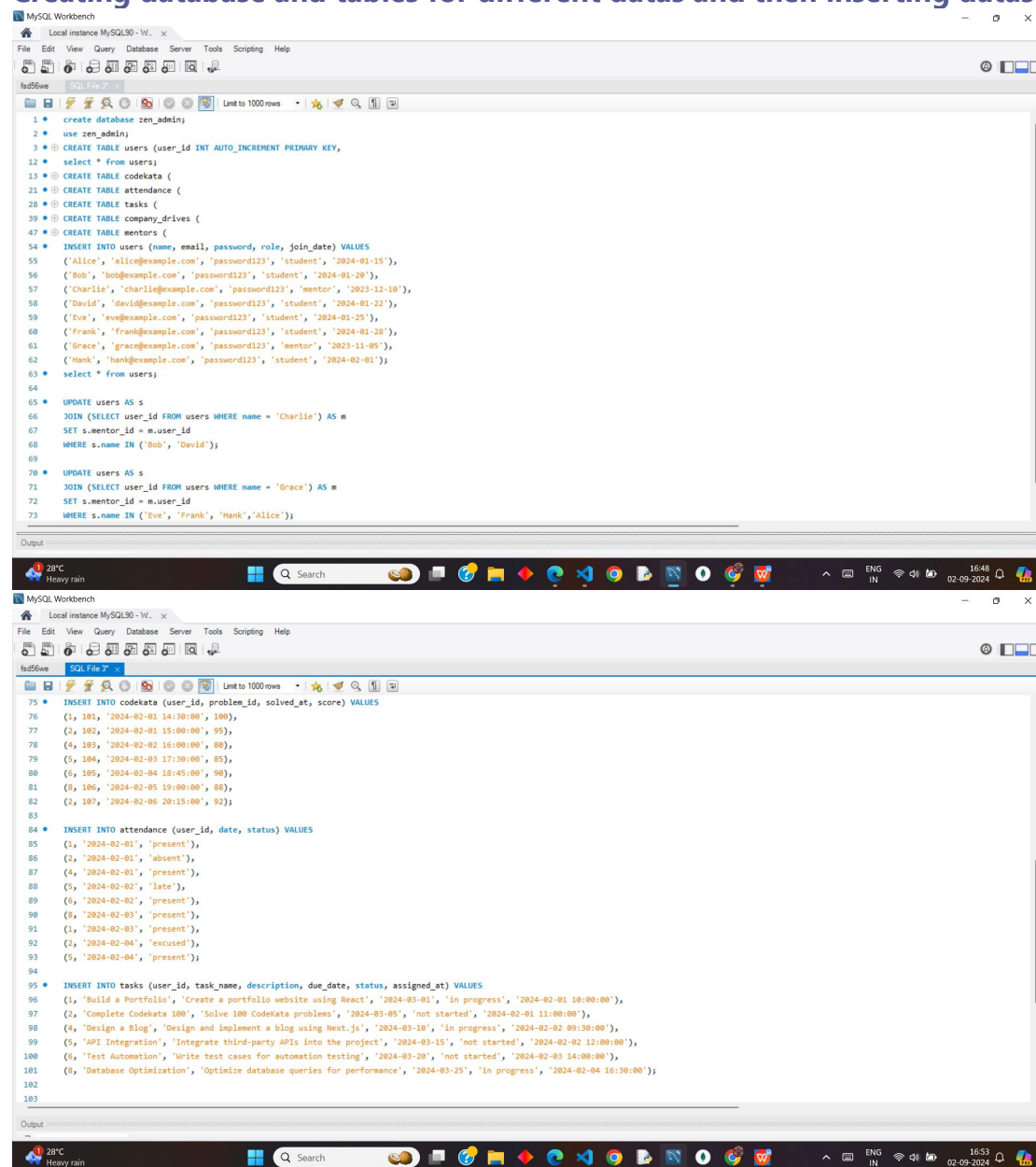


# Design DB model for Guvi Zen class

## Relationships:

- **Users to Codekata:** One-to-many relationship, as each user (student) can solve many CodeKata problems.
- **Users to Attendance:** One-to-many relationship, as each user has multiple attendance records.
- **Users to Tasks:** One-to-many relationship, as each user can be assigned multiple tasks.
- **Users to Mentors:** Self-referencing one-to-many relationship, as a mentor can have multiple mentees, and a student can have one mentor.
- **Company Drives:** This table is independent but can be linked to users if you want to track which students participated.

## Creating database and tables for different datas and then inserting datas



The screenshot displays the MySQL Workbench interface with a SQL script for creating a database and tables, and inserting data. The script is as follows:

```
1 • create database zen_admin;
2 • use zen_admin;
3 • CREATE TABLE users (user_id INT AUTO_INCREMENT PRIMARY KEY,
12 • select * from users;
13 • CREATE TABLE codekata (
21 • CREATE TABLE attendance (
28 • CREATE TABLE tasks (
39 • CREATE TABLE company_drives (
47 • CREATE TABLE mentors (
54 • INSERT INTO users (name, email, password, role, join_date) VALUES
55 ('Alice', 'alice@example.com', 'password123', 'student', '2024-01-15'),
56 ('Bob', 'bob@example.com', 'password123', 'student', '2024-01-20'),
57 ('Charlie', 'charlie@example.com', 'password123', 'mentor', '2023-12-10'),
58 ('David', 'david@example.com', 'password123', 'student', '2024-01-22'),
59 ('Eve', 'eve@example.com', 'password123', 'student', '2024-01-25'),
60 ('Frank', 'frank@example.com', 'password123', 'student', '2024-02-10'),
61 ('Grace', 'grace@example.com', 'password123', 'mentor', '2023-11-05'),
62 ('Hank', 'hank@example.com', 'password123', 'student', '2024-02-01');
63 • select * from users;
64
65 • UPDATE users AS s
66 JOIN (SELECT user_id FROM users WHERE name = 'Charlie') AS m
67 SET s.mentor_id = m.user_id
68 WHERE s.name IN ('Bob', 'David');
69
70 • UPDATE users AS s
71 JOIN (SELECT user_id FROM users WHERE name = 'Grace') AS m
72 SET s.mentor_id = m.user_id
73 WHERE s.name IN ('Eve', 'Frank', 'Hank', 'Alice');
```

The second screenshot shows the continuation of the SQL script, inserting data into the codekata, attendance, and tasks tables:

```
75 • INSERT INTO codekata (user_id, problem_id, solved_at, score) VALUES
76 (1, 101, '2024-02-01 14:30:00', 100),
77 (2, 102, '2024-02-01 15:00:00', 95),
78 (4, 103, '2024-02-02 16:00:00', 80),
79 (5, 104, '2024-02-03 17:30:00', 85),
80 (6, 105, '2024-02-04 18:45:00', 90),
81 (8, 106, '2024-02-05 19:00:00', 88),
82 (2, 107, '2024-02-06 20:15:00', 92);
83
84 • INSERT INTO attendance (user_id, date, status) VALUES
85 (1, '2024-02-01', 'present'),
86 (2, '2024-02-01', 'absent'),
87 (4, '2024-02-01', 'present'),
88 (5, '2024-02-02', 'late'),
89 (6, '2024-02-02', 'present'),
90 (6, '2024-02-03', 'present'),
91 (1, '2024-02-03', 'present'),
92 (2, '2024-02-04', 'excused'),
93 (5, '2024-02-04', 'present');
94
95 • INSERT INTO tasks (user_id, task_name, description, due_date, status, assigned_at) VALUES
96 (1, 'Build a Portfolio', 'Create a portfolio website using React', '2024-03-01', 'in progress', '2024-02-01 10:00:00'),
97 (2, 'Complete Codekata 100', 'Solve 100 Codekata problems', '2024-03-05', 'not started', '2024-02-01 11:00:00'),
98 (4, 'Design a Blog', 'Design and implement a blog using Next.js', '2024-03-10', 'in progress', '2024-02-02 09:30:00'),
99 (5, 'API Integration', 'Integrate third-party APIs into the project', '2024-03-15', 'not started', '2024-02-02 12:00:00'),
100 (6, 'Text Automation', 'Write test cases for automation testing', '2024-03-20', 'not started', '2024-02-03 14:00:00'),
101 (8, 'Database Optimization', 'Optimize database queries for performance', '2024-03-25', 'in progress', '2024-02-04 16:30:00');
102
103
```

MySQL Workbench

Local instance MySQL80 - W...

File Edit View Query Database Server Tools Scripting Help

SQL File 3

```
92 (2, '2024-02-04', 'excused'),
93 (5, '2024-02-04', 'present'));
94
95 • INSERT INTO tasks (user_id, task_name, description, due_date, status, assigned_at) VALUES
96 ((1, 'Build a Portfolio', 'Create a portfolio website using React', '2024-03-01', 'in progress', '2024-02-01 10:00:00'),
97 (2, 'Complete Codekata 100', 'Solve 100 Codekata problems', '2024-03-05', 'not started', '2024-02-01 11:00:00'),
98 (3, 'Design a Blog', 'Design and Implement a blog using Next.js', '2024-03-10', 'in progress', '2024-02-02 09:30:00'),
99 (5, 'API Integration', 'Integrate third-party APIs into the project', '2024-03-15', 'not started', '2024-02-02 12:00:00'),
100 (6, 'Test Automation', 'Write test cases for automation testing', '2024-03-20', 'not started', '2024-02-03 14:00:00'),
101 (8, 'Database Optimization', 'Optimize database queries for performance', '2024-03-25', 'in progress', '2024-02-04 16:30:00'));
102
103
104 • INSERT INTO company_drives (company_name, drive_date, location, eligibility_criteria) VALUES
105 ('TechCorp', '2024-04-15', 'Bangalore', 'B.Tech with 70%+ in aggregate'),
106 ('Innovatech', '2024-05-10', 'Hyderabad', 'Any degree with 60%+ in aggregate'),
107 ('AlphaTech', '2024-05-25', 'Pune', 'B.Tech/M.Tech with 75%+ in aggregate'),
108 ('BetaSoft', '2024-06-05', 'Chennai', 'Any degree with 65%+ in aggregate'),
109 ('GammaSolutions', '2024-06-15', 'Delhi', 'B.Sc/BCA with 70%+ in aggregate'),
110 ('DeltaCorp', '2024-07-01', 'Mumbai', 'MBA with 65%+ in aggregate');
111
112 • INSERT INTO mentors (mentor_id, mentee_id) VALUES
113 ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'Bob')),
114 ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'David')),
115 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Alice')),
116 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Eve')),
117 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Frank')),
118 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Hank'));
119
120 select * from mentors;
```

Output

28°C Heavy rain

Search

ENG IN

16:53 02-09-2024

## Basic Queries

### a. Retrieve all students' details:

MySQL Workbench

Local instance MySQL80 - W...

File Edit View Query Database Server Tools Scripting Help

SQL File 3

```
105 ('TechCorp', '2024-04-15', 'Bangalore', 'B.Tech with 70%+ in aggregate'),
106 ('Innovatech', '2024-05-10', 'Hyderabad', 'Any degree with 60%+ in aggregate'),
107 ('AlphaTech', '2024-05-25', 'Pune', 'B.Tech/M.Tech with 75%+ in aggregate'),
108 ('BetaSoft', '2024-06-05', 'Chennai', 'Any degree with 65%+ in aggregate'),
109 ('GammaSolutions', '2024-06-15', 'Delhi', 'B.Sc/BCA with 70%+ in aggregate'),
110 ('DeltaCorp', '2024-07-01', 'Mumbai', 'MBA with 65%+ in aggregate');
111
112 • INSERT INTO mentors (mentor_id, mentee_id) VALUES
113 ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'Bob')),
114 ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'David')),
115 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Alice')),
116 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Eve')),
117 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Frank')),
118 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Hank'));
119
120 select * from mentors;
121
122 • SELECT * FROM users WHERE role = 'student';
123
```

Result Grid

|   | user_id | name              | email       | password | role       | join_date | mentor_id |
|---|---------|-------------------|-------------|----------|------------|-----------|-----------|
| 1 | Alice   | alice@example.com | password123 | student  | 2024-01-15 | 7         |           |
| 2 | Bob     | bob@example.com   | password123 | student  | 2024-01-20 | 3         |           |
| 4 | David   | david@example.com | password123 | student  | 2024-01-22 | 3         |           |
| 5 | Eve     | eve@example.com   | password123 | student  | 2024-01-25 | 7         |           |
| 6 | Frank   | frank@example.com | password123 | student  | 2024-01-28 | 7         |           |
| 8 | Hank    | hank@example.com  | password123 | student  | 2024-02-01 | 7         |           |

users 4 x

Apply Revert

Output

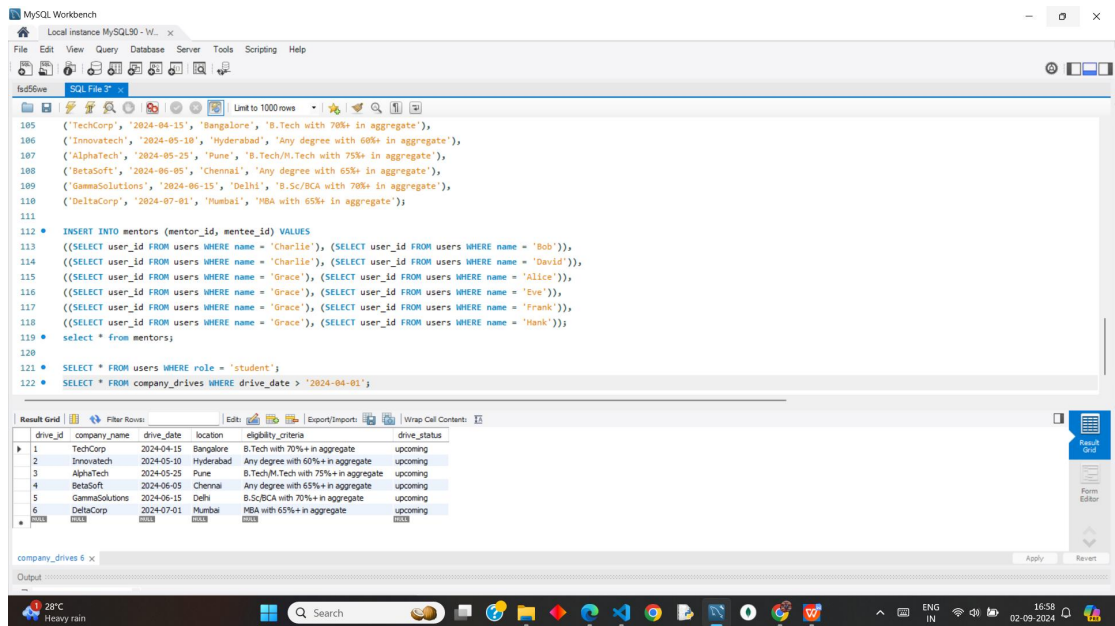
28°C Heavy rain

Search

ENG IN

16:57 02-09-2024

## b. List all company drives happening after a certain date:



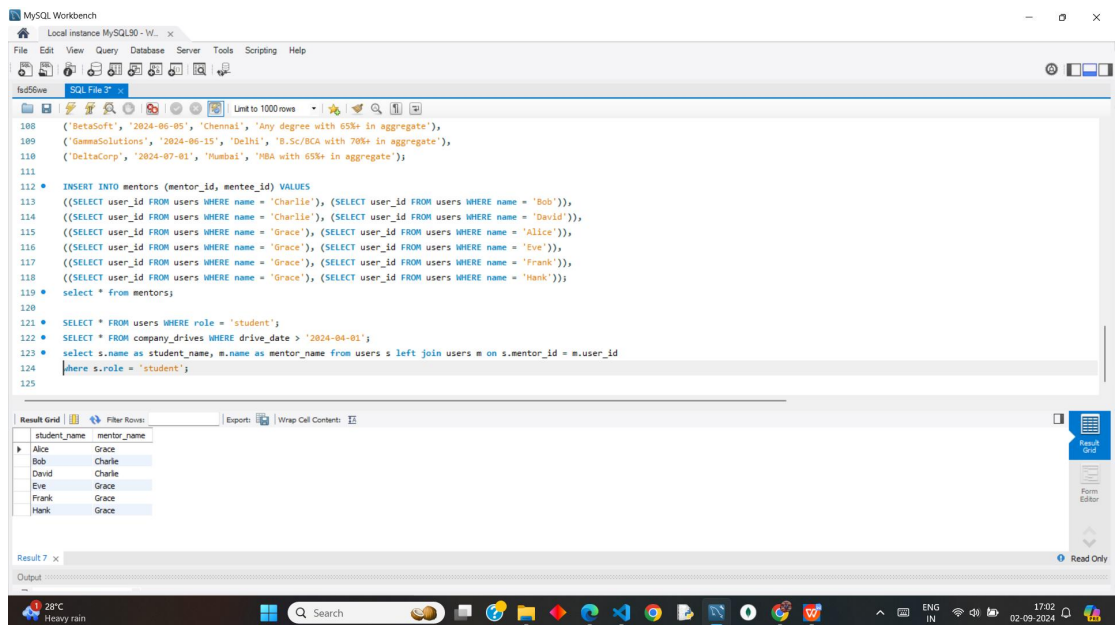
The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that inserts data into the 'mentors' table and then selects all records from it. The 'company\_drives' table is also visible in the 'Result Grid'.

```
112 • INSERT INTO mentors (mentor_id, mentee_id) VALUES
113 • ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'Bob')),
114 • ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'David')),
115 • ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Alice')),
116 • ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Eve')),
117 • ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Frank')),
118 • ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Hank'));
119 • select * from mentors;
120
121 • SELECT * FROM users WHERE role = 'student';
122 • SELECT * FROM company_drives WHERE drive_date > '2024-04-01';
```

| drive_id | company_name   | drive_date | location  | eligibility_criteria                 | drive_status |
|----------|----------------|------------|-----------|--------------------------------------|--------------|
| 1        | TechCorp       | 2024-04-15 | Bangalore | B.Tech with 70%+ in aggregate        | upcoming     |
| 2        | Innovatech     | 2024-05-10 | Hyderabad | Any degree with 60%+ in aggregate    | upcoming     |
| 3        | AlphaTech      | 2024-05-25 | Pune      | B.Tech/M.Tech with 75%+ in aggregate | upcoming     |
| 4        | BetaSoft       | 2024-06-05 | Chennai   | Any degree with 65%+ in aggregate    | upcoming     |
| 5        | GammaSolutions | 2024-06-15 | Delhi     | B.Sc/BCA with 70%+ in aggregate      | upcoming     |
| 6        | DeltaCorp      | 2024-07-01 | Mumbai    | MBA with 65%+ in aggregate           | upcoming     |

## Joins

### a. Retrieve all students along with their mentor's name:



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that selects all students and their mentors using a left join. The 'Result Grid' displays the output of the query.

```
120
121 • SELECT * FROM users WHERE role = 'student';
122 • SELECT * FROM company_drives WHERE drive_date > '2024-04-01';
123 • select s.name as student_name, m.name as mentor_name from users s left join users m on s.mentor_id = m.user_id
124 • where s.role = 'student';
125
```

| student_name | mentor_name |
|--------------|-------------|
| Alice        | Grace       |
| Bob          | Charlie     |
| David        | Charlie     |
| Eve          | Grace       |
| Frank        | Grace       |
| Hank         | Grace       |

## b. List students with their attendance status for a specific date:

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is as follows:

```
111
112 • INSERT INTO mentors (mentor_id, mentee_id) VALUES
113 ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'Bob')),
114 ((SELECT user_id FROM users WHERE name = 'Charlie'), (SELECT user_id FROM users WHERE name = 'David')),
115 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Alice')),
116 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Eve')),
117 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Frank')),
118 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Hank'));
119 • select * from mentors;
120
121 • SELECT * FROM users WHERE role = 'student';
122 • SELECT * FROM company_drives WHERE drive_date > '2024-04-01';
123 • select s.name as student_name, m.name as mentor_name from users s left join users m on s.mentor_id = m.user_id
124 where s.role = 'student';
125 • SELECT u.name, a.date, a.status
126 FROM users u
127 JOIN attendance a ON u.user_id = a.user_id
128 WHERE a.date = '2024-02-01' AND u.role = 'student';
```

The result grid shows the following data:

| name  | date       | status  |
|-------|------------|---------|
| Alice | 2024-02-01 | present |
| Bob   | 2024-02-01 | absent  |
| David | 2024-02-01 | present |

## Aggregate Functions

### a. Count the number of students assigned to each mentor:

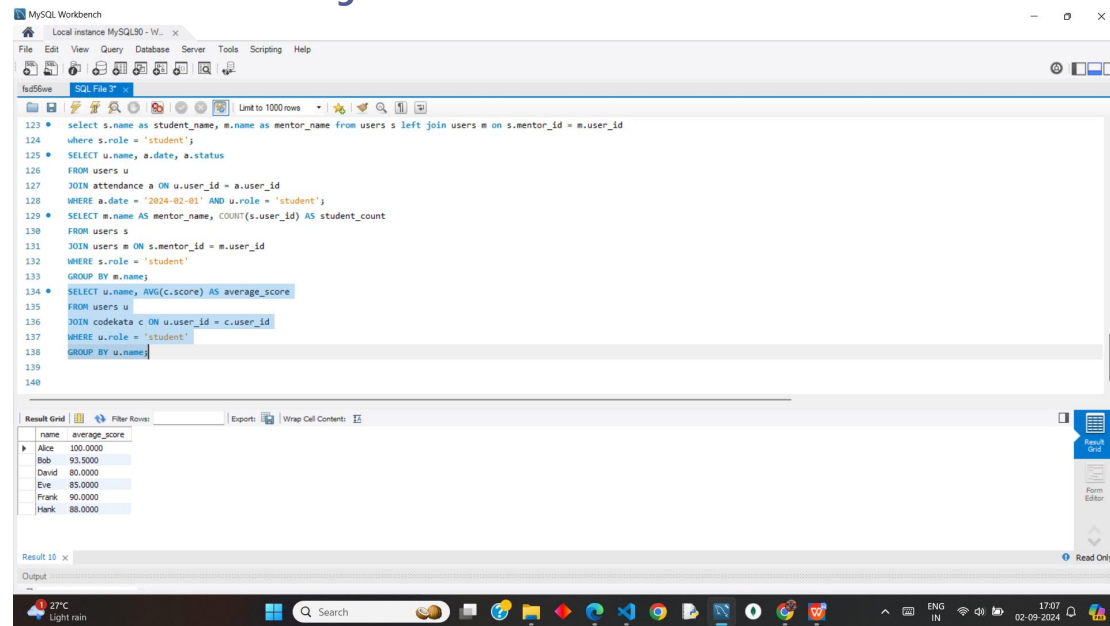
The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is as follows:

```
117 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Frank')),
118 ((SELECT user_id FROM users WHERE name = 'Grace'), (SELECT user_id FROM users WHERE name = 'Hank'));
119 • select * from mentors;
120
121 • SELECT * FROM users WHERE role = 'student';
122 • SELECT * FROM company_drives WHERE drive_date > '2024-04-01';
123 • select s.name as student_name, m.name as mentor_name from users s left join users m on s.mentor_id = m.user_id
124 where s.role = 'student';
125 • SELECT u.name, a.date, a.status
126 FROM users u
127 JOIN attendance a ON u.user_id = a.user_id
128 WHERE a.date = '2024-02-01' AND u.role = 'student';
129 • SELECT m.name AS mentor_name, COUNT(s.user_id) AS student_count
130 FROM users s
131 JOIN users m ON s.mentor_id = m.user_id
132 WHERE s.role = 'student'
133 GROUP BY m.name;
```

The result grid shows the following data:

| mentor_name | student_count |
|-------------|---------------|
| Grace       | 4             |
| Charlie     | 2             |

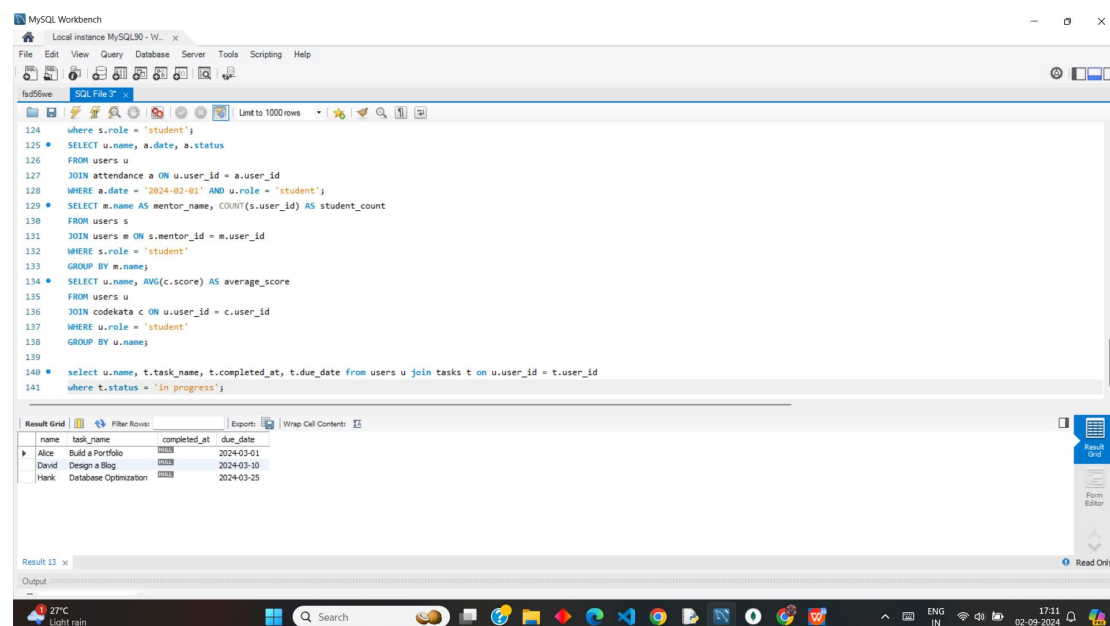
## b. Calculate the average Codekata score for each student:



```
123 • select s.name as student_name, m.name as mentor_name from users s left join users m on s.mentor_id = m.user_id
124 where s.role = 'student';
125 • SELECT u.name, a.date, a.status
126 FROM users u
127 JOIN attendance a ON u.user_id = a.user_id
128 WHERE a.date = '2024-02-01' AND u.role = 'student';
129 • SELECT m.name AS mentor_name, COUNT(s.user_id) AS student_count
130 FROM users s
131 JOIN users m ON s.mentor_id = m.user_id
132 WHERE s.role = 'student'
133 GROUP BY m.name;
134 • SELECT u.name, AVG(c.score) AS average_score
135 FROM users u
136 JOIN codekata c ON u.user_id = c.user_id
137 WHERE u.role = 'student'
138 GROUP BY u.name;
```

| name  | average_score |
|-------|---------------|
| Alice | 100.0000      |
| Bob   | 95.0000       |
| David | 80.0000       |
| Eve   | 85.0000       |
| Frank | 90.0000       |
| Hank  | 88.0000       |

## Get the details of students who have status of 'in progress' in their tasks:

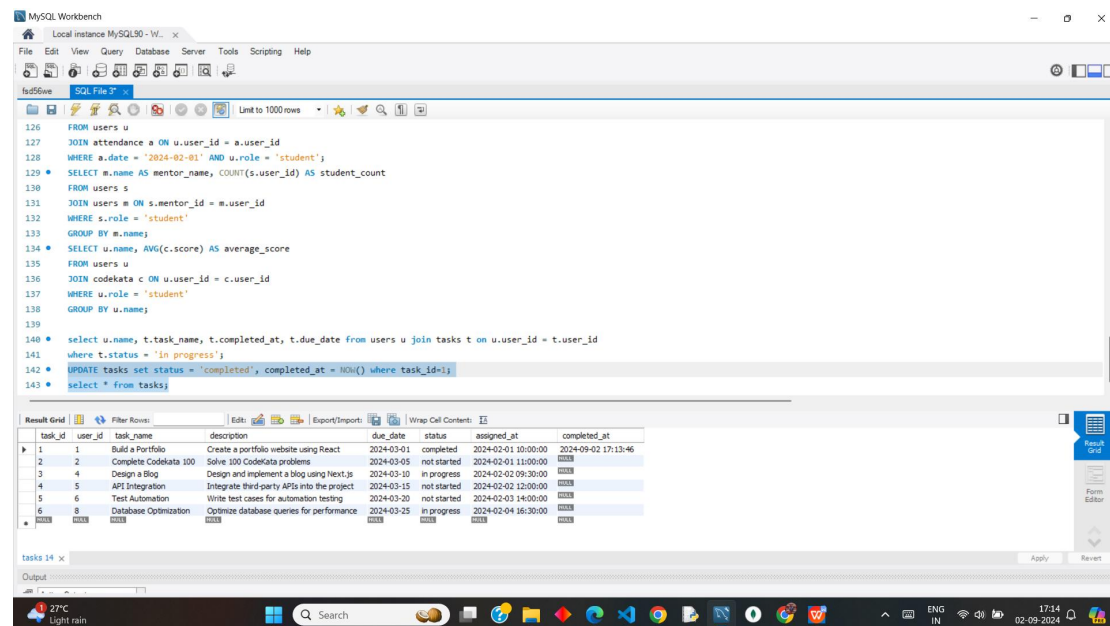


```
124 where s.role = 'student';
125 • SELECT u.name, a.date, a.status
126 FROM users u
127 JOIN attendance a ON u.user_id = a.user_id
128 WHERE a.date = '2024-02-01' AND u.role = 'student';
129 • SELECT m.name AS mentor_name, COUNT(s.user_id) AS student_count
130 FROM users s
131 JOIN users m ON s.mentor_id = m.user_id
132 WHERE s.role = 'student'
133 GROUP BY m.name;
134 • SELECT u.name, AVG(c.score) AS average_score
135 FROM users u
136 JOIN codekata c ON u.user_id = c.user_id
137 WHERE u.role = 'student'
138 GROUP BY u.name;
139
140 • select u.name, t.task_name, t.completed_at, t.due_date from users u join tasks t on u.user_id = t.user_id
141 where t.status = 'in progress';
```

| name  | task_name             | completed_at | due_date   |
|-------|-----------------------|--------------|------------|
| Alice | Build a Portfolio     | 2024-03-01   | 2024-03-01 |
| David | Design a Blog         | 2024-03-10   | 2024-03-10 |
| Hank  | Database Optimization | 2024-03-25   | 2024-03-25 |

## Updating and Deleting Data

### a. Update the status of a task when it's completed:



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that updates the status of tasks to 'completed' and sets the completed\_at timestamp to NOW(). The query is as follows:

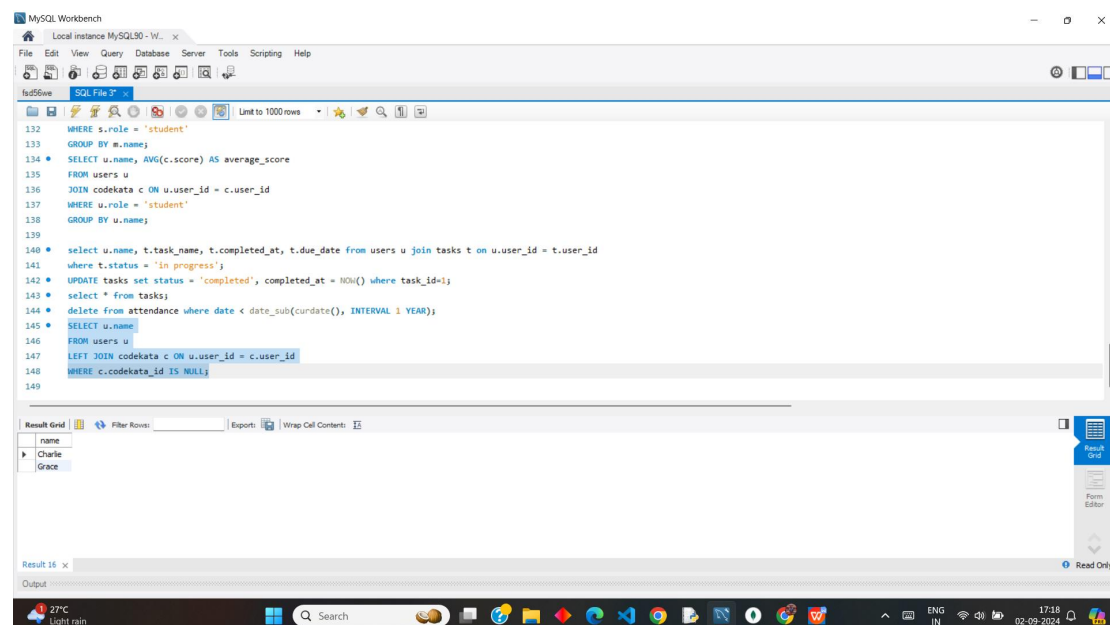
```
126 FROM users u
127 JOIN attendance a ON u.user_id = a.user_id
128 WHERE a.date = '2024-02-01' AND u.role = 'student';
129 • SELECT m.name AS mentor_name, COUNT(s.user_id) AS student_count
130 FROM users s
131 JOIN users m ON s.mentor_id = m.user_id
132 WHERE s.role = 'student'
133 GROUP BY m.name;
134 • SELECT u.name, AVG(c.score) AS average_score
135 FROM users u
136 JOIN codekata c ON u.user_id = c.user_id
137 WHERE u.role = 'student'
138 GROUP BY u.name;
139
140 • select u.name, t.task_name, t.completed_at, t.due_date from users u join tasks t on u.user_id = t.user_id
141 where t.status = 'in progress';
142 • UPDATE tasks set status = 'completed', completed_at = NOW() where task_id=i;
143 • select * from tasks;
```

The Result Grid shows the following data:

| task_id | user_id | task_name             | description                                 | due_date   | status      | assigned_at         | completed_at        |
|---------|---------|-----------------------|---|------------|-------------|---------------------|---------------------|
| 1       | 1       | Build a Portfolio     | Create a portfolio website using React      | 2024-03-01 | completed   | 2024-02-01 10:00:00 | 2024-09-02 17:13:46 |
| 2       | 2       | Complete Codekata 100 | Solve 100 Codekata problems                 | 2024-03-05 | not started | 2024-02-01 11:00:00 |                     |
| 3       | 4       | Design a Blog         | Design and implement a blog using Next.js   | 2024-03-10 | in progress | 2024-02-02 09:30:00 |                     |
| 4       | 5       | API Integration       | Integrate third-party APIs into the project | 2024-03-15 | not started | 2024-02-02 12:00:00 |                     |
| 5       | 6       | Test Automation       | Write test cases for automation testing     | 2024-03-20 | not started | 2024-02-03 14:00:00 |                     |
| 6       | 8       | Database Optimization | Optimize database queries for performance   | 2024-03-25 | in progress | 2024-02-04 16:30:00 |                     |

## Advanced Joins

### a. Find data who haven't solved any Codekata problems:



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that finds users who haven't solved any Codekata problems using a LEFT JOIN and a WHERE clause to filter for NULL values.

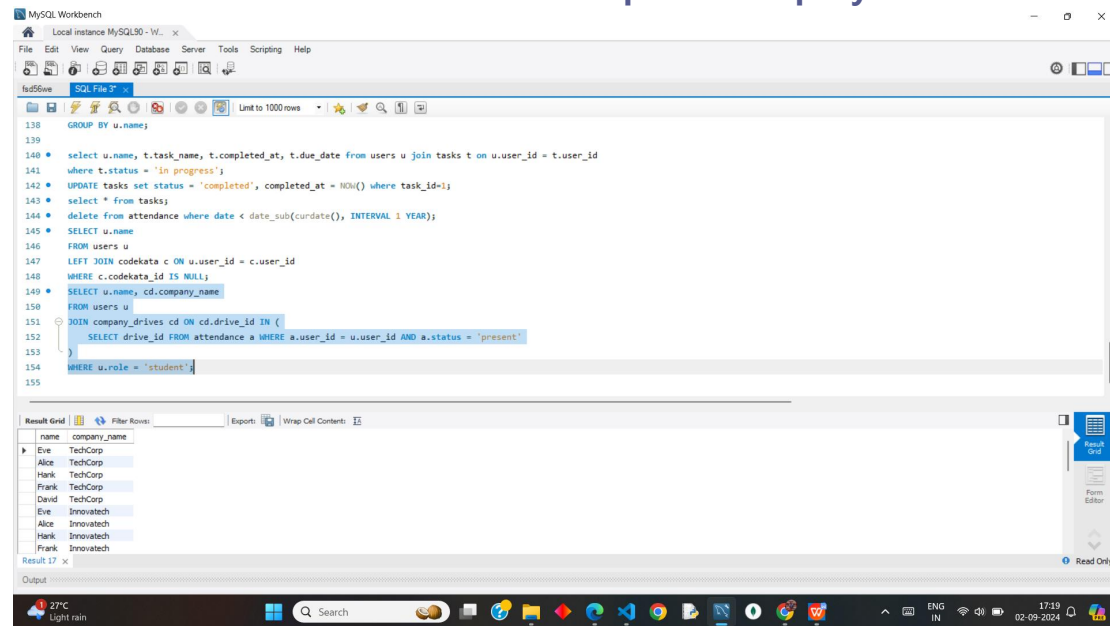
```
132 WHERE s.role = 'student'
133 GROUP BY m.name;
134 • SELECT u.name, AVG(c.score) AS average_score
135 FROM users u
136 JOIN codekata c ON u.user_id = c.user_id
137 WHERE u.role = 'student'
138 GROUP BY u.name;
139
140 • select u.name, t.task_name, t.completed_at, t.due_date from users u join tasks t on u.user_id = t.user_id
141 where t.status = 'in progress';
142 • UPDATE tasks set status = 'completed', completed_at = NOW() where task_id=i;
143 • select * from tasks;
144 • delete from attendance where date < date_sub(curdate(), INTERVAL 1 YEAR);
145 • SELECT u.name
146 FROM users u
147 LEFT JOIN codekata c ON u.user_id = c.user_id
148 WHERE c.codekata_id IS NULL;
149
```

The Result Grid shows the following data:

| name    |
|---------|
| Charlie |
| Grace   |



## b. Retrieve all students who attended a specific company drive:



The screenshot shows the MySQL Workbench interface. The SQL Editor contains a query that filters for students who attended a specific company drive. The query is as follows:

```
138 GROUP BY u.name;
139
140 • select u.name, t.task_name, t.completed_at, t.due_date from users u join tasks t on u.user_id = t.user_id
141 where t.status = 'in progress';
142 • UPDATE tasks set status = 'completed', completed_at = NOW() where task_id=1;
143 • select * from tasks;
144 • delete from attendance where date < date_sub(curdate(), INTERVAL 1 YEAR);
145 • SELECT u.name
146 FROM users u
147 LEFT JOIN codekata c ON u.user_id = c.user_id
148 WHERE c.codekata_id IS NULL;
149 • SELECT u.name, cd.company_name
150 FROM users u
151 JOIN company_drives cd ON cd.drive_id IN (
152 SELECT drive_id FROM attendance a WHERE a.user_id = u.user_id AND a.status = 'present'
153 )
154 WHERE u.role = 'student';
155
```

The Results Grid shows the following data:

| name  | company_name |
|-------|--------------|
| Eve   | TechCorp     |
| Alice | TechCorp     |
| Hank  | TechCorp     |
| Frank | TechCorp     |
| David | TechCorp     |
| Eve   | Innovatech   |
| Alice | Innovatech   |
| Hank  | Innovatech   |
| Frank | Innovatech   |

The bottom status bar shows the system temperature as 27°C, light rain, and the date as 02-09-2024.