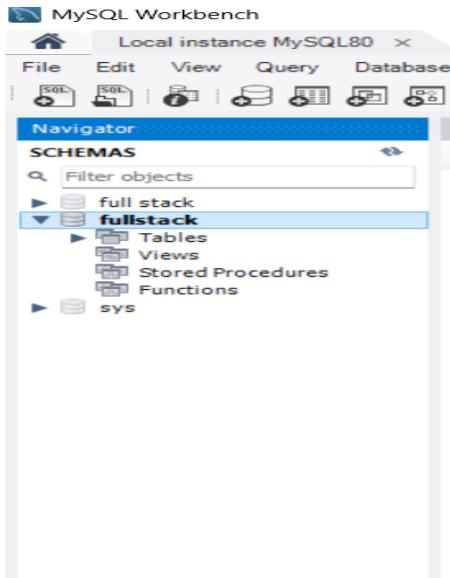


1. Create a database.

```
CREATE DATABASE fullstack;
```

```
USE fullstack;
```



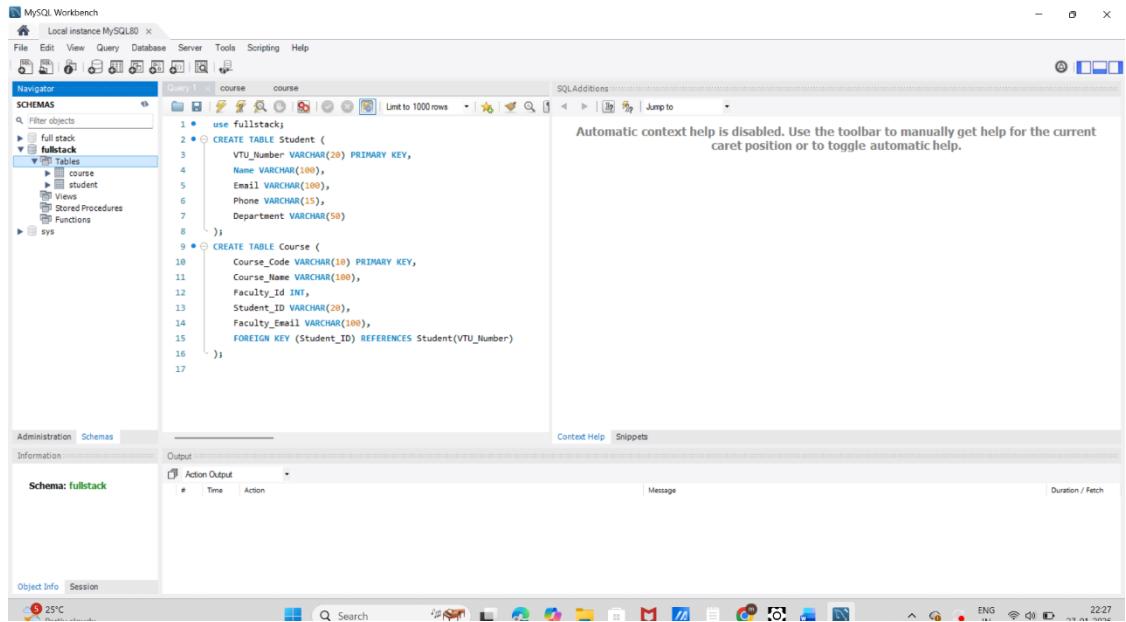
Create at least two tables:

Table 1: Student (VTU Number, Name, Email, Phone, Department)

```
CREATE TABLE Student (
    VTU_Number VARCHAR(20) PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    Department VARCHAR(50)
);
```

Table 2: Course (Course Code, Course Name, Faculty Id, Student ID, Faculty Email)

```
CREATE TABLE Course (
    Course_Code VARCHAR(10) PRIMARY KEY,
    Course_Name VARCHAR(100),
    Faculty_Id INT,
    Student_ID VARCHAR(20),
    Faculty_Email VARCHAR(100),
    FOREIGN KEY (Student_ID) REFERENCES Student(VTU_Number)
);
```



3. Insert minimum 5 records into each table.

#### -- Inserting into Student Table

#### INSERT INTO Student VALUES

```

('1MS23CS001', 'Arjun Kumar', 'arjun@mail.com', '9845011223', 'CSE'),
('1MS23CS002', 'Sneha Rao', 'sneha@mail.com', '9845022334', 'IT'),
('1MS23CS003', 'Rahul Nair', 'rahul@mail.com', '9845033445', 'CSE'),
('1MS23CS004', 'Priya Das', 'priya@mail.com', '9845044556', 'ECE'),
('1MS23CS005', 'Amit Singh', 'amit@mail.com', '9845055667', 'IT');

```

#### -- Inserting into Course Table

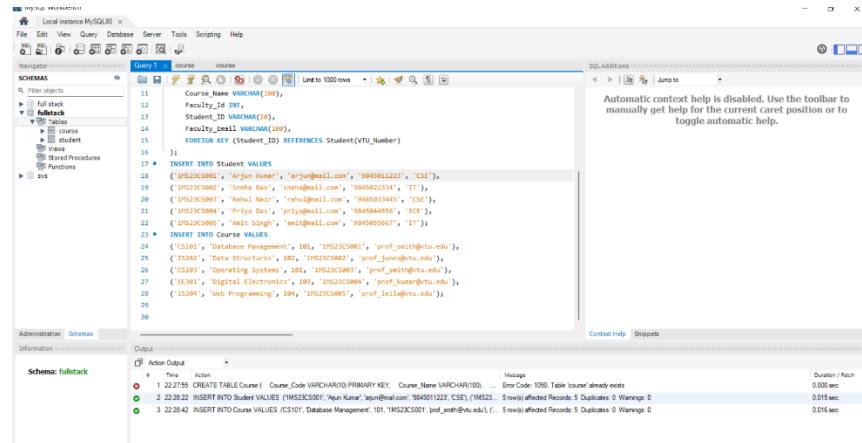
#### INSERT INTO Course VALUES

```

('CS101', 'Database Management', 101, '1MS23CS001',
'prof_smith@vtu.edu'),
('IS202', 'Data Structures', 102, '1MS23CS002', 'prof_jones@vtu.edu'),
('CS103', 'Operating Systems', 101, '1MS23CS003',
'prof_smith@vtu.edu'),

```

('EC301', 'Digital Electronics', 103, '1MS23CS004',  
 'prof\_kumar@vtu.edu'),  
 ('IS204', 'Web Programming', 104, '1MS23CS005', 'prof\_leila@vtu.edu');



```

CREATE TABLE Course (
  Course_Code VARCHAR(10) PRIMARY KEY,
  Course_Name VARCHAR(100),
  Faculty_Id INT,
  Student_ID VARCHAR(20),
  Faculty_Email VARCHAR(100),
  FOREIGN KEY (Student_ID) REFERENCES Student(VTU_Number)
);

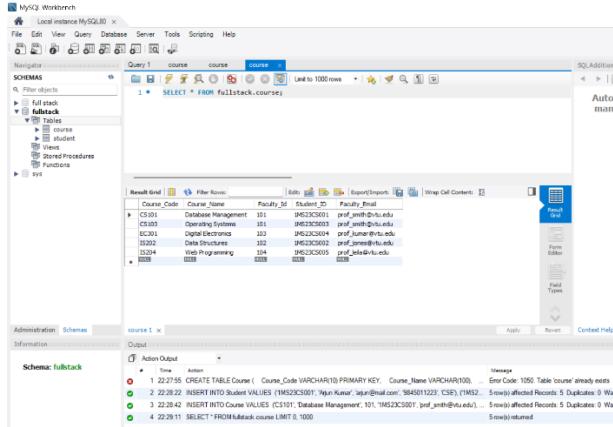
INSERT INTO Student VALUES
('1MS23CS001', 'Arijit Kumar', 'ajrj@gmail.com', '9845011232', 'CSE'),
('1MS23CS002', 'Sneha Rani', 'sneha@gmail.com', '9845023345', 'IT'),
('1MS23CS003', 'Renu Nali', 'renu@gmail.com', '9845033456', 'CSE'),
('1MS23CS004', 'Priya Das', 'priyadas@gmail.com', '9845044556', 'IT'),
('1MS23CS005', 'Anita Singh', 'anita@gmail.com', '9845055667', 'IT');

INSERT INTO Course VALUES
('CS101', 'Computer Organization', 101, '1MS23CS001', 'prof_sank@vtu.edu'),
('CS102', 'Data Structures', 102, '1MS23CS002', 'prof_jone@vtu.edu'),
('CS103', 'Operating Systems', 103, '1MS23CS003', 'prof_swami@vtu.edu'),
('EC301', 'Digital Electronics', 103, '1MS23CS004', 'prof_kumar@vtu.edu'),
('IS204', 'Web Programming', 104, '1MS23CS005', 'prof_leila@vtu.edu');
  
```

#### 4. Select records using different CLAUSE.

##### i. WHERE clause:

SELECT \* FROM Student WHERE Department = 'CSE';



Course_Code	Course_Name	Faculty_Id	Student_ID	Faculty_Email
CS101	Computer Organization	101	1MS23CS001	prof_sank@vtu.edu
CS102	Data Structures	102	1MS23CS002	prof_jone@vtu.edu
EC301	Digital Electronics	103	1MS23CS004	prof_kumar@vtu.edu
IS204	Web Programming	104	1MS23CS005	prof_leila@vtu.edu

##### ii. ORDER BY Clause (Sorting):

SELECT \* FROM Course ORDER BY Course\_Name ASC;

##### iii. LIKE Clause (Pattern Matching):

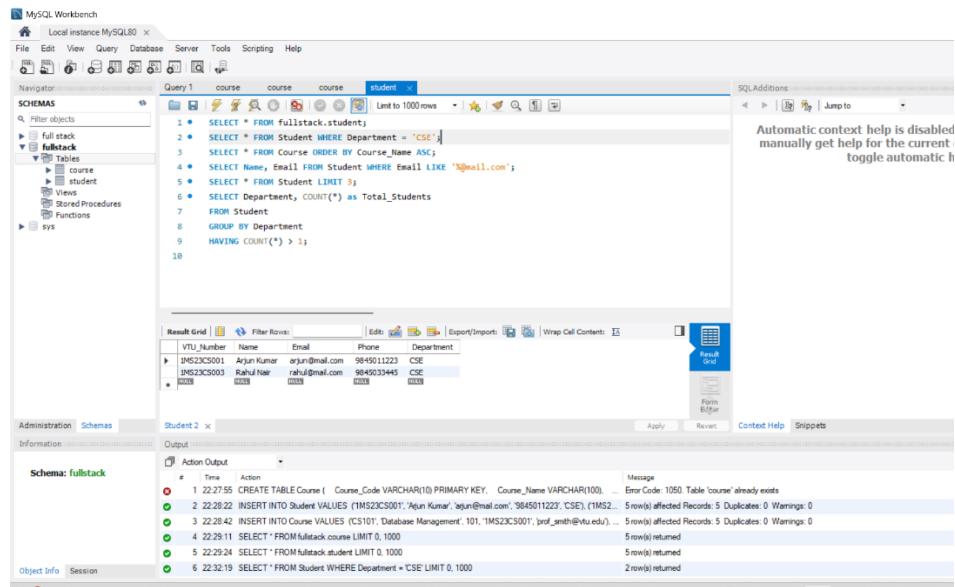
SELECT Name, Email FROM Student WHERE Email LIKE '%@mail.com';

##### iv. LIMIT Clause (Restricting Results):

SELECT \* FROM Student LIMIT 3;

##### v. GROUP BY & HAVING Clause (Aggregating):

```
SELECT Department, COUNT(*) as Total_Students
FROM Student
GROUP BY Department
HAVING COUNT(*) > 1;
```



## Session 3:

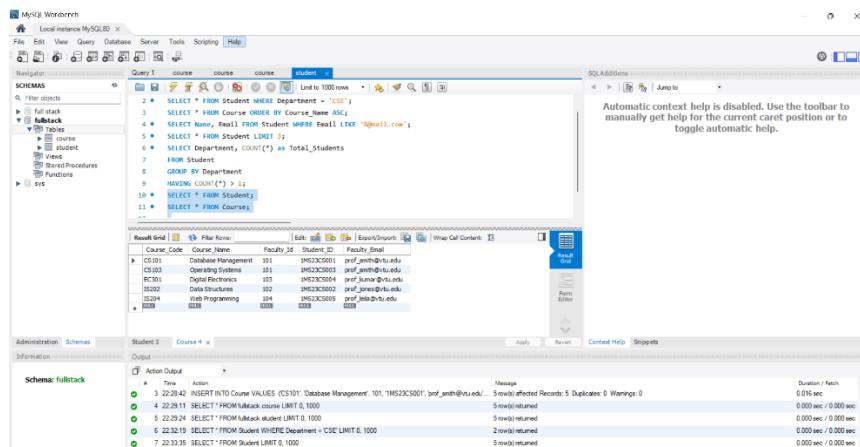
1. Write SELECT queries to display all records.

-- Display all students

**SELECT \* FROM Student;**

-- Display all courses

**SELECT \* FROM Course;**



- ## 2. Write queries using aggregate functions.

-- Count total number of students

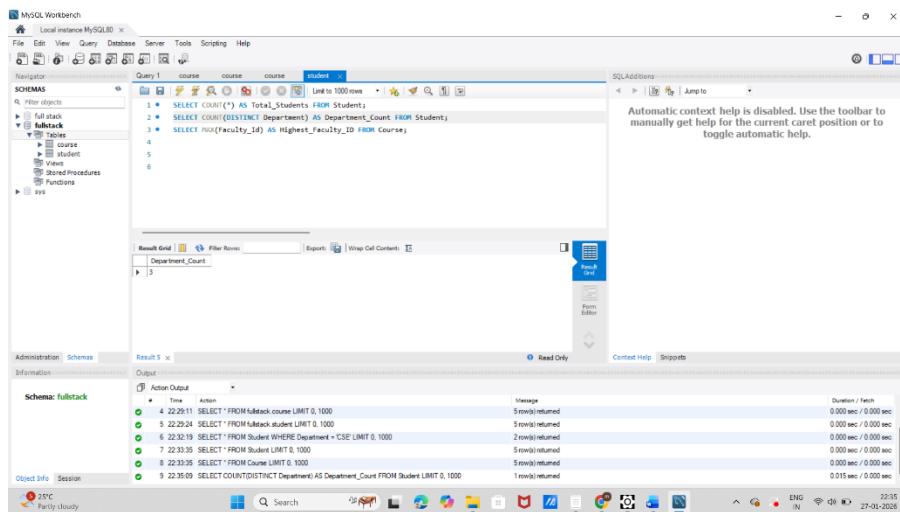
```
SELECT COUNT(*) AS Total_Students FROM Student;
```

-- Count unique departments

```
SELECT COUNT(DISTINCT Department) AS Department_Count FROM Student;
```

-- Find the highest Faculty ID assigned

```
SELECT MAX(Faculty_Id) AS Highest_Faculty_ID FROM Course;
```



3. Sort data and display based in ascending/descending order of the VTU number.

-- Sort - Ascending order (Default)

```
SELECT * FROM Student
```

```
ORDER BY VTU_Number ASC;
```

-- Sort - in Descending order

```
SELECT * FROM Student
```

```
ORDER BY VTU_Number DESC;
```

4. Display student records belonging to a particular department.

```
SELECT * FROM Student
```

```
WHERE Department = 'CSE';
```

The screenshot shows the MySQL Workbench interface. The left pane displays the 'Schemas' tree with 'student' selected. The 'Query' tab contains two queries:

```

1 • SELECT * FROM student ORDER BY VTU_Number ASC;
2 • SELECT Execute the statement under the keyboard cursor.
3
4
5

```

The results grid shows the following data:

VTU_Number	Name	Email	Phone	Department
102323C5001	Aryun Kumar	aryun@mail.com	9845012233	CSE
102323C5002	Sneha Rao	sneha@mail.com	9845022334	IT
102323C5003	Praveen	praveen@mail.com	9845034567	CSE
102323C5004	Priya Das	priya@mail.com	9845044556	ECE
102323C5005	Ankit Singh	ankit@mail.com	9845055667	IT
102323C5006	Udit	udit@mail.com	9845066778	IT

### Inner Join:

**SELECT**

```

s.VTU_Number,
s.Name AS Student_Name,
c.Course_Name,
c.Faculty_Id,
c.Faculty_Email

```

**FROM Student s**

**INNER JOIN Course c ON s.VTU\_Number = c.Student\_ID;**

### Left Join:

**SELECT**

```

s.VTU_Number,
s.Name,
c.Course_Name

```

**FROM Student s**

**LEFT JOIN Course c ON s.VTU\_Number = c.Student\_ID;**

### Right Join:

**SELECT**

```

s.Name,
c.Course_Code,
c.Course_Name

```

**FROM Student s**

**RIGHT JOIN Course c ON s.VTU\_Number = c.Student\_ID;**

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

- File Menu:** File, Edit, View, Query, Database, Server, Tool, Scripting, Help.
- Toolbar:** Standard database management tools like New, Open, Save, etc.
- Navigator:** Schemas, showing the current database is "fullstack".
- Query Editor:** Contains a multi-line text area with the following SQL query:

```
6  --> 6. SELECT * FROM student WHERE Department = 'CSE' LIMIT 0, 1000
7  FROM STUDENT s
8  INNER JOIN Course c ON s.VTU_Number = c.Student_ID;
9  *
10 SELECT
11   s.VTU_Number,
12   s.Name,
13   c.Course_Name
14  FROM Student s
15  LEFT JOIN Course c ON s.VTU_Number = c.Student_ID;
16
17 --
```
- Result Grid:** Displays the results of the query:

VTU_Number	Name	Course_Name
1M623CS001	Arjun Kumar	Database Management
1M623CS002	Sneha Rao	Data Structures
1M623CS003	Ranu Nar	Operating Systems
1M623CS004	Priya Das	Digital Electronics
1M623CS005	Ant Singh	Web Programming
- Right Panel:** Shows "SQLAdditions" and "Automatic complete manually get" buttons.
- Bottom Navigation:** Administration, Schemas, Information, Schema: fullstack, Action Output, Read Only, Context Help, Snippets.

## **FULL JOIN (Full Outer Join):**

```
SELECT s.Name, c.Course_Name FROM Student s
LEFT JOIN Course c ON s.VTU_Number = c.Student_ID
UNION
SELECT s.Name, c.Course_Name FROM Student s
RIGHT JOIN Course c ON s.VTU_Number = c.Student_ID
```

## Cross Join(Cartesian Product):

```
SELECT s.Name, c.Course_Name  
FROM Student s  
CROSS JOIN Course c;
```

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

- Query Editor:** Displays a complex SQL query involving multiple tables (Student, Course, Department, Faculty) and joins (INNER JOIN, LEFT JOIN, RIGHT JOIN, UNION). The query retrieves student names, course names, and department names.
- Results Grid:** Shows the output of the query, listing student names, course codes, and course names.
- Action History:** A history of recent actions, including SELECT statements and a COUNT(DISTINCT) query.
- Information:** A panel showing the schema of the current database (fullstack).
- Object Info:** A panel showing the session information.