```
CREATE DATABASE StudentInformationDB;
USE StudentInformationDB;

CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50) NOT NULL,
    DepartmentHead VARCHAR(50)
);

INSERT INTO Department VALUES
(1, 'Computer Science', 'Dr. Smith'),
(2, 'Mechanical Engineering', 'Dr. Johnson'),
(3, 'Electrical Engineering', 'Dr. Williams'),
(4, 'Civil Engineering', 'Dr. Brown'),
(5, 'Mathematics', 'Dr. Davis');
select * from department;

# Instructor Table
CREATE TABLE Instructor (
    InstructorID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(50),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

INSERT INTO Instructor VALUES
(101, 'Alice', 'Taylor', 'alice.taylor@uni.com', 1),
(102, 'Bob', 'Miller', 'bob.miller@uni.com', 2),
(103, 'Charlie', 'Wilson', 'charlie.wilson@uni.com', 3),
(104, 'Diana', 'Moore', 'diana.moore@uni.com', 4),
(105, 'Ethan', 'Clark', 'ethan.clark@uni.com', 5);

 # Course Table
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(50) NOT NULL,
    CourseDuration VARCHAR(20),
    CourseFee DECIMAL(10,2),
    InstructorID INT,
    DepartmentID INT,
    FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

INSERT INTO Course VALUES
(201, 'Data Structures', '4 Months', 2000.00, 101, 1),
(202, 'Thermodynamics', '5 Months', 2500.00, 102, 2),
(203, 'Circuit Analysis', '3 Months', 2200.00, 103, 3),
(204, 'Structural Design', '6 Months', 3000.00, 104, 4),
(205, 'Calculus', '4 Months', 1800.00, 105, 5);

 # Student Table
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
```

```
    LastName VARCHAR(50) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    Email VARCHAR(50)
);
INSERT INTO Student VALUES
(301, 'John', 'Doe', '2001-05-15', 'Male', 'john.doe@email.com'),
(302, 'Jane', 'Smith', '2000-08-22', 'Female', 'jane.smith@email.com'),
(303, 'Michael', 'Johnson', '2002-03-10', 'Male',
'michael.johnson@email.com'),
(304, 'Emily', 'Davis', '2001-12-05', 'Female',
'emily.davis@email.com'),
(305, 'David', 'Brown', '2000-11-30', 'Male', 'david.brown@email.com');

 # Enrollment Table
CREATE TABLE Enrollment (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    EnrollmentDate DATE,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
INSERT INTO Enrollment VALUES
(401, 301, 201, '2025-01-10'),
(402, 302, 202, '2025-01-12'),
(403, 303, 203, '2025-01-15'),
(404, 304, 204, '2025-01-18'),
(405, 305, 205, '2025-01-20');




# Exam Table
CREATE TABLE Exam (
    ExamID INT PRIMARY KEY,
    CourseID INT,
    ExamDate DATE,
    TotalMarks INT,
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
INSERT INTO Exam VALUES
(501, 201, '2025-06-10', 65),
(502, 202, '2025-06-12', 80),
(503, 203, '2025-06-15', 75),
(504, 204, '2025-06-18', 84),
(505, 205, '2025-06-20', 90);

select * from Student;
select * from course;
select * from Instructor;
select * from Department;
select * from exam;
select * from Enrollment;
#BASED ON WHERE CLUASE AND LOGICAL OPERATORS AND SPECIAL OPERATORS
SELECT
    FirstName, LastName,
    DATEDIFF(YEAR, DateOfBirth, GETDATE()) AS Age
```

```
FROM Student;
SELECT
    SUM(CourseFee) AS TotalFee
FROM Course
WHERE CourseID IN (201, 202);

SELECT
    StudentID,
    FirstName + ' ' + LastName AS FullName
FROM Student;
select * from student;
select * from student where studentId =301 and firstname='john';
SELECT FirstName, LastName, Gender
FROM Student
WHERE FirstName LIKE 'J%' AND Gender = 'Male';

SELECT *
FROM Student
WHERE FirstName LIKE 'm%' AND Gender = 'Male';

SELECT ExamID, CourseID, TotalMarks
FROM Exam
WHERE TotalMarks BETWEEN 80 AND 100;

SELECT CourseID, CourseName
FROM Course
WHERE InstructorID <> 102; # not equal

SELECT CourseName, CONCAT('Course: ', CourseName) AS FullCourseName,
       CourseFee, CourseFee * 0.9 AS DiscountedFee
FROM Course
WHERE CourseFee > 2000;

SELECT ExamID, CourseID, ExamDate
FROM Exam
WHERE ExamDate BETWEEN '2025-06-20' AND '2025-07-30';

SELECT InstructorID, FirstName, LastName, DepartmentID
FROM Instructor
WHERE (DepartmentID = 1 OR DepartmentID = 2) AND DepartmentID <> 3;

SELECT StudentID, FirstName, LastName, Email
FROM Student
WHERE Email IS NULL OR Email LIKE '%@email.com';

 # BASED ON SINGLE ROW FUNCTION
SELECT
    CourseID,
    UPPER(CourseName) AS CourseNameUpper
FROM Course;

SELECT
    CourseID,
    LOWER(CourseName) AS CourseNameUpper
FROM Course;

SELECT
    StudentID,
```

```
    FirstName,
    LastName,
    DATEDIFF(YEAR, DateOfBirth, GETDATE()) AS Age
FROM Student;

SELECT
    CourseID,
    CourseName,
    ROUND(CourseFee, 0) AS RoundedFee
FROM Course;

SELECT
    ExamID,
    CourseID,
    ExamDate,
    MONTH(ExamDate) AS ExamMonth
FROM Exam;

#BASE ON MULTI ROW FUNCTION

SELECT COUNT(*) AS TotalStudents
FROM Student;

SELECT MAX(CourseFee) AS MaxCourseFee
FROM Course;

SELECT MIN(CourseFee) AS MaxCourseFee
FROM Course;

SELECT MIN(TotalMarks) AS MinExamMarks
FROM Exam;

SELECT AVG(CourseFee) AS AvgCourseFee
FROM Course;

SELECT SUM(CourseFee) AS total_fees from course;

# based on group clause

SELECT
    Gender,
    COUNT(*) AS TotalStudents
FROM Student
GROUP BY Gender;

SELECT
    CourseID,
    COUNT(StudentID) AS TotalStudents
FROM Enrollment
GROUP BY CourseID;

SELECT
    DepartmentID,
    MAX(CourseFee) AS MaxCourseFee
FROM Course
GROUP BY DepartmentID;

SELECT
```

```
    CourseID,
    AVG(TotalMarks) AS AvgMarks
FROM Exam
GROUP BY CourseID;

SELECT
    DepartmentID,
    COUNT(InstructorID) AS TotalInstructors
FROM Instructor
GROUP BY DepartmentID;

SELECT
    EnrollmentDate,
    COUNT(EnrollmentID) AS TotalEnrollments
FROM Enrollment
GROUP BY EnrollmentDate;

 # BASED ON HAVING CALUSE

 SELECT
    CourseID,
    COUNT(StudentID) AS TotalStudents
FROM Enrollment
GROUP BY CourseID
HAVING COUNT(StudentID) > 0;
SELECT
    DepartmentID,
    COUNT(InstructorID) AS TotalInstructors
FROM Instructor
GROUP BY DepartmentID
HAVING COUNT(InstructorID) >0;

SELECT
    DepartmentID,
    SUM(CourseFee) AS TotalFees
FROM Course
GROUP BY DepartmentID
HAVING SUM(CourseFee) > 2500;

select * from course where coursefee > 2500;

SELECT
    CourseID,
    AVG(TotalMarks) AS AvgMarks
FROM Exam
GROUP BY CourseID
HAVING AVG(TotalMarks) > 75;

SELECT
    EnrollmentDate,
    COUNT(EnrollmentID) AS TotalEnrollments
FROM Enrollment
GROUP BY EnrollmentDate
HAVING COUNT(EnrollmentID) > 0;

SELECT
    DepartmentID,
    MIN(CourseFee) AS MinFee
```

```
FROM Course
GROUP BY DepartmentID
HAVING MIN(CourseFee) < 2000;

# BASED ON SUBQUERY

SELECT StudentID, FirstName, LastName
FROM Student
WHERE StudentID IN (
    SELECT StudentID
    FROM Enrollment
    WHERE CourseID = (
        SELECT CourseID
        FROM Course
        WHERE CourseFee = (SELECT MAX(CourseFee) FROM Course)
    )
);

SELECT CourseName
FROM Course
WHERE CourseID IN (
    SELECT DISTINCT CourseID
    FROM Enrollment
);

SELECT CourseName, CourseFee
FROM Course
WHERE CourseFee = (SELECT MAX(CourseFee) FROM Course);

SELECT StudentID, FirstName, LastName, DateOfBirth
FROM Student
WHERE DateOfBirth < (
    SELECT AVG(DateOfBirth) FROM Student
);

SELECT ExamID, CourseID, ExamDate
FROM Exam
WHERE ExamDate = (SELECT MAX(ExamDate) FROM Exam);

SELECT DepartmentID, DepartmentName
FROM Department
WHERE DepartmentID = (
    SELECT DepartmentID
    FROM Course
    WHERE CourseFee = (SELECT MIN(CourseFee) FROM Course)
);

SELECT StudentID, FirstName, LastName
FROM Student
WHERE StudentID IN (
    SELECT StudentID
    FROM Enrollment
    WHERE EnrollmentDate = (SELECT MAX(EnrollmentDate) FROM Enrollment)
);

# BASED ON MULTI SUB_QUERY

SELECT StudentID, FirstName, LastName
```

```
FROM Student
WHERE StudentID IN (
    SELECT StudentID
    FROM Enrollment
    WHERE CourseID = (
        SELECT CourseID
        FROM Exam
        WHERE ExamDate = (SELECT MAX(ExamDate) FROM Exam)
    )
);
select * from Instructor where InstructorID=( SELECT INSTRUCTORID FROM
COURSE WHERE COURSEFEE =(SELECT MAX(COURSEFEE) FROM COURSE));

SELECT * FROM DEPARTMENT WHERE DEPARTMENTID IN (SELECT DEPARTMENTID
FROM COURSE WHERE COURSEID IN (SELECT COURSEID FROM Enrollment));

SELECT DepartmentID, DepartmentName
FROM Department
WHERE DepartmentID IN (
    SELECT DepartmentID
    FROM Course
    WHERE CourseID IN (
        SELECT CourseID FROM Enrollment
    )
);


 SELECT StudentID, FirstName, LastName
FROM Student
WHERE StudentID IN (
    SELECT StudentID
    FROM Enrollment
    WHERE CourseID IN (
        SELECT CourseID
        FROM Course
        WHERE DepartmentID = (
            SELECT DepartmentID
            FROM Department
            WHERE DepartmentName = 'Computer Science'
        )
    )
);

SELECT ExamID, ExamDate, TotalMarks
FROM Exam
WHERE CourseID IN (
    SELECT CourseID
    FROM Course
    WHERE DepartmentID = (
        SELECT DepartmentID
        FROM Course
        GROUP BY DepartmentID
        ORDER BY COUNT(CourseID) DESC
        LIMIT 1
    )
);
```

```
# BASED ON MULTI ROW SUB QUERY USING SPECIAL OPERATORS (ANY,ALL)

SELECT CourseName, CourseFee
FROM Course
WHERE CourseFee > ANY (
    SELECT CourseFee
    FROM Course
    WHERE DepartmentID = 1
);

SELECT CourseName, CourseFee
FROM Course
WHERE CourseFee < ALL (
    SELECT CourseFee
    FROM Course
    WHERE DepartmentID = 2
);

SELECT StudentID, FirstName, LastName
FROM Student
WHERE StudentID IN (
    SELECT StudentID
    FROM Enrollment
    WHERE CourseID = ANY (
        SELECT CourseID
        FROM Course
        WHERE CourseFee > ANY (
            SELECT CourseFee
            FROM Course
            WHERE DepartmentID = 3
        )
    )
);

SELECT InstructorID, FirstName, LastName
FROM Instructor
WHERE DepartmentID = ANY (
    SELECT DepartmentID
    FROM Course
    WHERE CourseFee > ALL (
        SELECT CourseFee
        FROM Course
        WHERE DepartmentID = 1
    )
);

SELECT CourseID, CourseName, CourseFee
FROM Course
WHERE CourseFee = ANY (
    SELECT TotalMarks
    FROM Exam
);

SELECT ExamID, CourseID, ExamDate, TotalMarks
FROM Exam
WHERE TotalMarks > ALL (
    SELECT CourseFee
    FROM Course
```

```
);

# based on order by clause and limit clause

SELECT CourseID, CourseName, CourseFee
FROM Course
ORDER BY CourseFee DESC
LIMIT 4;

SELECT StudentID, FirstName, LastName, DateOfBirth
FROM Student
ORDER BY DateOfBirth DESC
LIMIT 5;

SELECT ExamID, CourseID, ExamDate, TotalMarks
FROM Exam
ORDER BY ExamDate DESC
LIMIT 2;

SELECT InstructorID, FirstName, LastName
FROM Instructor
ORDER BY LastName ASC
LIMIT 3;

SELECT DepartmentID, COUNT(CourseID) AS TotalCourses
FROM Course
GROUP BY DepartmentID
ORDER BY TotalCourses DESC
LIMIT 1;

# based on offset clause

SELECT CourseID, CourseName, CourseFee
FROM Course
ORDER BY CourseFee DESC
LIMIT 1 OFFSET 1;

SELECT StudentID, FirstName, LastName, DateOfBirth
FROM Student
ORDER BY DateOfBirth DESC
LIMIT 3 OFFSET 2;

SELECT ExamID, CourseID, ExamDate, TotalMarks
FROM Exam
ORDER BY ExamDate DESC
LIMIT 1 OFFSET 1;

SELECT InstructorID, FirstName, LastName
FROM Instructor
ORDER BY LastName ASC
LIMIT 3 OFFSET 2;

SELECT DepartmentID, COUNT(CourseID) AS TotalCourses
FROM Course
GROUP BY DepartmentID
ORDER BY TotalCourses DESC
LIMIT 1 OFFSET 1;
```

```
# based on joins (inner joins)

SELECT s.FirstName, s.LastName, c.CourseName
FROM Student s
INNER JOIN Enrollment e ON s.StudentID = e.StudentID
INNER JOIN Course c ON e.CourseID = c.CourseID;

# Students with their courses
select s.firstname,s.lastname, c.coursename
from student s
inner join Enrollment e on s.studentid=e.studentid
inner join course c on e.course= c.courseid;

# Courses with instructor names
SELECT c.CourseName, i.FirstName, i.LastName
FROM Course c
INNER JOIN Instructor i ON c.InstructorID = i.InstructorID;

# Courses with department names
SELECT c.CourseName, d.DepartmentName
FROM Course c
INNER JOIN Department d ON c.DepartmentID = d.DepartmentID;

# List students with their exam details
SELECT s.FirstName, s.LastName, ex.ExamDate, ex.TotalMarks
FROM Student s
INNER JOIN Enrollment e ON s.StudentID = e.StudentID
INNER JOIN Exam ex ON e.CourseID = ex.CourseID;

 # List courses with instructor and department
SELECT c.CourseName, i.FirstName AS InstructorFirstName, i.LastName AS
InstructorLastName, d.DepartmentName
FROM Course c
INNER JOIN Instructor i ON c.InstructorID = i.InstructorID
INNER JOIN Department d ON c.DepartmentID = d.DepartmentID;

# List exams with course name and instructor
SELECT ex.ExamID, ex.ExamDate, c.CourseName, i.FirstName, i.LastName
FROM Exam ex
INNER JOIN Course c ON ex.CourseID = c.CourseID
INNER JOIN Instructor i ON c.InstructorID = i.InstructorID;

# Count students in each course (using INNER JOIN + GROUP BY)
SELECT c.CourseName, COUNT(s.StudentID) AS TotalStudents
FROM Student s
INNER JOIN Enrollment e ON s.StudentID = e.StudentID
INNER JOIN Course c ON e.CourseID = c.CourseID
GROUP BY c.CourseName;

# outer join
# List all students with their enrolled courses (LEFT OUTER JOIN)
SELECT s.StudentID, s.FirstName, s.LastName, c.CourseName
FROM Student s
LEFT OUTER JOIN Enrollment e ON s.StudentID = e.StudentID
LEFT OUTER JOIN Course c ON e.CourseID = c.CourseID;

# List all courses with their instructors (LEFT OUTER JOIN)
SELECT c.CourseID, c.CourseName, i.FirstName, i.LastName
```

```
FROM Course c
LEFT OUTER JOIN Instructor i ON c.InstructorID = i.InstructorID;

# List all courses with department names (RIGHT OUTER JOIN)
SELECT c.CourseName, d.DepartmentName
FROM Course c
RIGHT OUTER JOIN Department d ON c.DepartmentID = d.DepartmentID;

 # List all exams and course names (FULL OUTER JOIN)
SELECT ex.ExamID, ex.ExamDate, c.CourseName
FROM Exam ex
FULL OUTER JOIN Course c ON ex.CourseID = c.CourseID;


 # List all exams and course names (FULL OUTER JOIN)
SELECT ex.ExamID, ex.ExamDate, c.CourseName
FROM Exam ex
FULL OUTER JOIN Course c ON ex.CourseID = c.CourseID;


SELECT ex.ExamID, ex.ExamDate, c.CourseName
FROM Exam ex
LEFT JOIN Course c ON ex.CourseID = c.CourseID

UNION

-- Step 2: Get all courses with matching exams
SELECT ex.ExamID, ex.ExamDate, c.CourseName
FROM Exam ex
RIGHT JOIN Course c ON ex.CourseID = c.CourseID;

# List all courses with department names (RIGHT OUTER JOIN)
SELECT c.CourseName, d.DepartmentName
FROM Course c
RIGHT OUTER JOIN Department d ON c.DepartmentID = d.DepartmentID;


# self join
# Students in the same course
SELECT s1.FirstName AS Student1, e1.CourseID
FROM Student s1
JOIN Enrollment e1 ON s1.StudentID = e1.StudentID;

# Instructors in the same department
SELECT i1.FirstName AS Instructor1, i2.FirstName AS Instructor2,
i1.DepartmentID
FROM Instructor i1
JOIN Instructor i2 ON i1.DepartmentID = i2.DepartmentID AND
i1.InstructorID < i2.InstructorID;

 # Courses in the same department
SELECT c1.CourseName AS Course1, c2.CourseName AS Course2,
c1.DepartmentID
FROM Course c1
JOIN Course c2 ON c1.DepartmentID = c2.DepartmentID AND c1.CourseID <
c2.CourseID;

# Exams for the same course
```

```
SELECT ex1.ExamID AS Exam1, ex2.ExamID AS Exam2, ex1.CourseID
FROM Exam ex1
JOIN Exam ex2 ON ex1.CourseID = ex2.CourseID AND ex1.ExamID
ex2.ExamID;


call GetStudentByID('302');
call GetCourseByID('201');
call GetInstructorByID('104');
CALL GetDepartmentByID('1');
CALL GetExamByID('501');

# Trigger: Set default total marks for new exams


DELIMITER //

CREATE TRIGGER trg_DefaultTotalMarks
BEFORE INSERT ON Exam
FOR EACH ROW
BEGIN
    IF NEW.TotalMarks IS NULL THEN
        SET NEW.TotalMarks = 100;
    END IF;
END;
//

DELIMITER ;

DROP TRIGGER IF EXISTS trg_DefaultTotalMarks;

INSERT INTO Exam (ExamID,  ExamDate)
VALUES (1,  '2025-09-22');

SELECT * FROM Exam;

DELIMITER //

CREATE TRIGGER trg_SetEnrollmentDate
BEFORE INSERT ON Enrollment
FOR EACH ROW
BEGIN
    IF NEW.EnrollmentDate IS NULL THEN
        SET NEW.EnrollmentDate = CURDATE();
    END IF;
END;
//

DELIMITER ;

INSERT INTO Enrollment (EnrollmentID, StudentID, CourseID)
VALUES (1, 1, 2);

INSERT INTO Student (StudentID, FirstName, LastName, DateOfBirth,
Gender, Email)
VALUES (1, 'Manish', 'Walekar', '2000-01-01', 'M',
'manish@example.com');
```

```
INSERT INTO Course (CourseID, CourseName, CourseDuration, CourseFee,
InstructorID, DepartmentID)
VALUES (2, 'Computer Networks', '4 Months', 1000, 1, 1);


SELECT * FROM Enrollment;
SELECT * FROM STUDENT;
```