# University Database Management System

## Objective:

Develop SQL queries to manage a university database, covering the basics of RDBMS, table creation, data manipulation, ordering data, and using functions.

### SQL Query to Display Schema for Each Table:

```
-- Describe Students table
DESCRIBE Students;

-- Describe Courses table
DESCRIBE Courses;

-- Describe Enrollments table
DESCRIBE Enrollments;
```

## 2. SQL Create Table, Update, Insert, and Delete

### Create Tables:

```
-- Create Students table
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE,
    dob DATE
);

-- Create Courses table
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    department VARCHAR(50)
);

-- Create Enrollments table
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
```

```
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

#### Insert Data:

```
-- Insert data into Students table
INSERT INTO Students (student_id, first_name, last_name, email, dob) VALUES
(1, 'Amol', 'Patole', 'amol@example.com', '2000-01-01'),
(2, 'Akshay', 'Patole', 'akshay@example.com', '1999-02-02');

-- Insert data into Courses table
INSERT INTO Courses (course_id, course_name, department) VALUES
(1, 'Database Systems', 'Computer Science'),
(2, 'Operating Systems', 'Computer Science');

-- Insert data into Enrollments table
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES
(1, 1, 1, '2024-01-15'),
(2, 1, 2, '2024-01-20');
```

#### Update Data:

```
-- Update email address for a specific student
UPDATE Students
SET email = 'new.email@example.com'
WHERE student_id = 1;
```

#### Delete Data:

```
-- Delete a specific enrollment record
DELETE FROM Enrollments
WHERE enrollment_id = 1;
```

## 3. OrderBy

#### Query to Display All Students Ordered by Last Name:

```
SELECT * FROM Students
ORDER BY last_name;
```

### Query to Display All Courses Ordered by Course Name in Descending Order:

```
SELECT * FROM Courses
ORDER BY course_name DESC;
```

### Query to Display All Enrollments Ordered by Enrollment Date:

```
SELECT * FROM Enrollments
ORDER BY enrollment_date;
```

## 4. Functions and Inbuilt Functions

### Aggregate Functions:

```
-- Query to count the number of students
SELECT COUNT(*) AS number_of_students FROM Students;
```

### String Functions:

```
-- Query to display the first and last name of each student in uppercase
SELECT UPPER(first_name) AS first_name, UPPER(last_name) AS last_name FROM Students;
```

### Date Functions:

```
-- Query to display the age of each student based on their date of birth
SELECT first_name, last_name,
    FLOOR(DATEDIFF(CURDATE(), dob) / 365) AS age
FROM Students;
```

### Mathematical Functions:

```
-- Query to find the average length of course names
SELECT AVG(LENGTH(course_name)) AS avg_course_name_length FROM Courses;
```

### Conditional Functions:

```
-- Query to display students who have enrolled in more than one course
SELECT s.student_id, s.first_name, s.last_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY s.student_id
HAVING COUNT(e.course_id) > 1;
```