

# Database Management System

(Assignment No 01)



Session (2022-2026)

Program/Class

**BS-Computer Science / 5<sup>th</sup> Section-A**

Submitted By:

Student Name: Shaheer Ali

Roll Number: 301-221044

Supervised By:

Ms. Muneeba Darwaish

Lecturer

CS& IT Department

---

**Hazara University, Mansehra**

# Type Of Database To Create

You are managing a **student management system** database that contains the following tables:

## 1. Students

student_id	first_name	last_name	enrollment_year	major	gpa
1	John	Doe	2022	CS	3.5
2	Jane	Smith	2021	IT	3.7
3	Alice	Johnson	2023	CS	3.2
4	Bob	Lee	2021	IT	2.8
5	Charlie	Brown	2022	CS	3.9

## 2. Courses

course_id	course_name	credit_hours
101	Database Systems	3
102	Computer Networks	4
103	Software Engineering	3
104	Cybersecurity Basics	2

## 3. Enrollments

enrollment_id	student_id	course_id	semester	grade
1	1	101	Spring	A
2	2	102	Fall	B
3	3	101	Fall	C
4	4	104	Spring	B
5	1	103	Fall	A

---

# Creating Database StudentManagement

```
CREATE DATABASE StudentManagement;  
USE StudentManagement;
```

```
CREATE TABLE Students (  
    student_id INT PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    enrollment_year INT,  
    major VARCHAR(50),  
    gpa DECIMAL(3, 2)  
);
```

```
INSERT INTO Students (student_id, first_name, last_name, enrollment_year, major, gpa)  
VALUES  
(1, 'John', 'Doe', 2022, 'CS', 3.5),  
(2, 'Jane', 'Smith', 2021, 'IT', 3.7),  
(3, 'Alice', 'Johnson', 2023, 'CS', 3.2),  
(4, 'Bob', 'Lee', 2021, 'IT', 2.8),  
(5, 'Charlie', 'Brown', 2022, 'CS', 3.9);
```

```
CREATE TABLE Courses (  
    course_id INT PRIMARY KEY,  
    course_name VARCHAR(100),  
    credit_hours INT  
);
```

```
INSERT INTO Courses (course_id, course_name, credit_hours) VALUES  
(101, 'Database Systems', 3),  
(102, 'Computer Networks', 4),  
(103, 'Software Engineering', 3),  
(104, 'Cybersecurity Basics', 2);
```

```
CREATE TABLE Enrollments (  
    enrollment_id INT PRIMARY KEY,  
    student_id INT,  
    course_id INT,  
    semester VARCHAR(20),  
    grade CHAR(1),  
    FOREIGN KEY (student_id) REFERENCES Students(student_id),  
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)  
);
```

```
INSERT INTO Enrollments (enrollment_id, student_id, course_id, semester, grade) VALUES  
(1, 1, 101, 'Spring', 'A'),  
(2, 2, 102, 'Fall', 'B'),  
(3, 3, 101, 'Fall', 'C'),  
(4, 4, 104, 'Spring', 'B'),  
(5, 1, 103, 'Fall', 'A');
```

## Write SQL queries to answer the following questions:

1. **Retrieve all students** who are enrolled in a **CS major** and have a **GPA greater than 3.0**.

### Query:

```
SELECT student_id, first_name, last_name, enrollment_year, major, gpa
```

```
FROM Students
```

```
WHERE major = 'CS' AND gpa > 3.0;
```

### Screenshot:

The screenshot shows a SQL query editor with the following query:

```
49 • SELECT student_id, first_name, last_name, enrollment_year, major, gpa
50 FROM Students
51 WHERE major = 'CS' AND gpa > 3.0;
52
```

Below the query editor is a "Result Grid" showing the results of the query. The grid has columns for student\_id, first\_name, last\_name, enrollment\_year, major, and gpa. The results are as follows:

	student_id	first_name	last_name	enrollment_year	major	gpa
▶	1	John	Doe	2022	CS	3.50
	3	Alice	Johnson	2023	CS	3.20
	5	Charlie	Brown	2022	CS	3.90
*	NULL	NULL	NULL	NULL	NULL	NULL

The interface includes a "Filter Rows" section, an "Edit" button, and an "Export/Import" button. The "Result Grid" button is also visible on the right side.

- Find the **average GPA** of students enrolled in the **IT major**.

**Query:**

```
SELECT AVG(gpa) AS avg_gpa
```

```
FROM Students
```

```
WHERE major = 'IT';
```

**Screenshot:**

The screenshot shows a SQL query editor with the following text:

```
52  
53 • SELECT AVG(gpa) AS avg_gpa  
54 FROM Students  
55 WHERE major = 'IT';  
56
```

Below the editor is a toolbar with the following options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The Result Grid is active, showing the following data:

avg_gpa
3.250000

At the bottom of the window, there is a tab labeled "Result 2" and a "Read Only" indicator.

3. List all courses along with the **number of students enrolled** in each course.

**Query:**

```
SELECT Courses.course_id, Courses.course_name, COUNT(Enrollments.student_id) AS  
num_students
```

```
FROM Courses
```

```
LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
```

```
GROUP BY Courses.course_id, Courses.course_name;
```

**Screenshot:**

The screenshot shows a database query editor with a SQL query and a result grid below it. The query is as follows:

```
56  
57 • SELECT Courses.course_id, Courses.course_name, COUNT(Enrollments.student_id) AS num_students  
58 FROM Courses  
59 LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id  
60 GROUP BY Courses.course_id, Courses.course_name;  
61  
62
```

The result grid displays the following data:

course_id	course_name	num_students
101	Database Systems	2
102	Computer Networks	1
103	Software Engineering	1
104	Cybersecurity Basics	1

4. Identify students who are taking the **Database Systems** course in any semester.

**Query:**

```
SELECT Students.student_id, Students.first_name, Students.last_name  
  
FROM Students  
  
JOIN Enrollments ON Students.student_id = Enrollments.student_id  
  
JOIN Courses ON Enrollments.course_id = Courses.course_id  
  
WHERE Courses.course_name = 'Database Systems';
```

**Screenshot:**

The screenshot displays a SQL query editor with the following code:

```
62 • SELECT Students.student_id, Students.first_name, Students.last_name  
63 FROM Students  
64 JOIN Enrollments ON Students.student_id = Enrollments.student_id  
65 JOIN Courses ON Enrollments.course_id = Courses.course_id  
66 WHERE Courses.course_name = 'Database Systems';
```

Below the query editor is a toolbar with options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The Result Grid is active, showing the following data:

	student_id	first_name	last_name
▶	1	John	Doe
	3	Alice	Johnson

At the bottom, there is a tab labeled "Result 5" and a "Read Only" status indicator.

5. Display students who received a **grade of A in at least two courses**.

**Query:**

```
SELECT Students.student_id, Students.first_name, Students.last_name  
  
FROM Students  
  
JOIN Enrollments ON Students.student_id = Enrollments.student_id  
  
WHERE Enrollments.grade = 'A'  
  
GROUP BY Students.student_id, Students.first_name, Students.last_name  
  
HAVING COUNT(Enrollments.course_id) >= 2;
```

**Screenshot:**

The screenshot shows a SQL query editor interface. The query is as follows:

```
68 SELECT Students.student_id, Students.first_name, Students.last_name  
69 FROM Students  
70 JOIN Enrollments ON Students.student_id = Enrollments.student_id  
71 WHERE Enrollments.grade = 'A'  
72 GROUP BY Students.student_id, Students.first_name, Students.last_name  
73 HAVING COUNT(Enrollments.course_id) >= 2;  
74
```

Below the query editor, the 'Result Grid' is displayed. It shows a table with the following data:

	student_id	first_name	last_name
▶	1	John	Doe

The interface includes a toolbar at the top with various icons and a 'Limit to 100 rows' dropdown. The bottom right corner features a 'Result Grid' button and a scroll bar.