Assignment On

Course Title: Database Management System

Course Code: CSEL-2204

[Assignment 1- University Database]

Submitted To,

Professor **Dr. Md. Manowarul Islam**

Department of CSE

Jagannath University, Dhaka

Submitted By,

Atik Jawad

ID: B220305043

Department of Computer Science & Engineering

Jagannath University, Dhaka

Date of Submission: 5/11/2025

Table of Contents

Introduction

This assignment demonstrates SQL table creation, data insertion, and query execution using basic, aggregate, and join operations in DBMS. The objective is to understand relational database design, normalization, and retrieval techniques using SQL queries.

Part 1: Table Creation & Data Entry

1. Create all four tables with proper primary key and foreign key constraints.

    1. Department Table:

```sql

CREATE TABLE Department (
  dept_id INT PRIMARY KEY,
  dept_name VARCHAR(50),
  location VARCHAR(50)
);
```

    2. Student Table:

```sql
CREATE TABLE Student (
  student_id INT PRIMARY KEY,
  student_name VARCHAR(50),
  dept_id INT,
  gender CHAR(1),
  cgpa DECIMAL(3,2),
  FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);
```

    3. Course Table:

```sql
CREATE TABLE Course (
  course_id INT PRIMARY KEY,
  course_name VARCHAR(50),
  credit INT,
  dept_id INT,
  FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);
```

4. **Enrollment Table:**

```sql
CREATE TABLE Enrollment (
  enroll_id INT PRIMARY KEY,
  student_id INT,
  course_id INT,
  semester VARCHAR(20),
  grade CHAR(2),
  FOREIGN KEY (student_id) REFERENCES Student(student_id),
  FOREIGN KEY (course_id) REFERENCES Course(course_id)
);
```

2. Insert at least five records into each table.

1. **INSERT INTO Department:**

```sql
INSERT INTO Department VALUES
(1, 'CSE',   'Block A'),
(2, 'EEE',   'Block B'),
(3, 'BBA',   'Block C'),
(4, 'CIVIL', 'Block D'),
(5, 'ENG',   'Block E');
```

2. **INSERT INTO Student:**

```sql
INSERT INTO Student VALUES
(101, 'Rafiul Islam',      1, 'M', 3.75),
(102, 'Jannatul Ferdous',  1, 'F', 3.92),
(103, 'Salman Rahman',     2, 'M', 3.50),
(104, 'Nusrat Nahar',      3, 'F', 3.20),
(105, 'Aisha Khan',        1, 'F', 3.60);
```

3. **INSERT INTO Course:**

```sql
INSERT INTO Course VALUES
(501, 'Database Systems', 3, 1),
(502, 'Data Structures',  3, 1),
(503, 'Circuit Analysis', 3, 2),
(504, 'Marketing 101',    3, 3),
(505, 'Algorithms',       3, 1),
(506, 'Signals',          3, 2);
```

4. **INSERT INTO Enrollment:**

```sql
INSERT INTO Enrollment VALUES
(1, 101, 501, 'Spring2025', 'A'),
(2, 101, 502, 'Spring2025', 'A-'),
(3, 102, 501, 'Spring2025', 'A'),
(4, 103, 503, 'Spring2025', 'B+'),
(5, 104, 504, 'Spring2025', 'A-'),
(6, 105, 505, 'Spring2025', 'B'),
(7, 102, 505, 'Spring2025', 'A-');  -- extra enrollment
```

3. Display all student details along with their department names.

```sql
SELECT s.student_id, s.student_name, s.dept_id, d.dept_name, s.gender, s.cgpa
FROM Student s
JOIN Department d ON s.dept_id = d.dept_id;
```

Result (rows):

student_id	student_name	dept_id	dept_name	gender	cgpa
101	Rafiul Islam	1	CSE	M	3.75
102	Jannatul Ferdous	1	CSE	F	3.92
103	Salman Rahman	2	EEE	M	3.50
104	Nusrat Nahar	3	BBA	F	3.20
105	Aisha Khan	1	CSE	F	3.60

Explanation: simple INNER JOIN between Student and Department on dept_id to show department names.

4. List all courses offered by the CSE department.

```sql
SELECT course_id, course_name, credit
FROM Course
WHERE dept_id = (SELECT dept_id FROM Department WHERE dept_name = 'CSE');
```

Result:

course_id	course_name	credit
501	Database Systems	3
502	Data Structures	3
505	Algorithms	3

Explanation: we filtered Course for dept_id of CSE. (Alternatively JOIN Department and filter on dept_name='CSE'.)

5. **Show all students who received a grade 'A' in Spring2025.**

```sql
SELECT DISTINCT s.student_id, s.student_name, e.course_id, e.grade
FROM Enrollment e
JOIN Student s ON e.student_id = s.student_id
WHERE e.semester = 'Spring2025' AND e.grade = 'A';
```

Result:

student_id	student_name	course_id	grade
101	Rafiul Islam	501	A
102	Jannatul Ferdous	501	A

Explanation: finds enrollments in Spring2025 with grade 'A' and shows the students (distinct so student appears once per course — here both got A in course 501).

Part 2: Aggregation & Grouping

6. Find the average CGPA of students in each department.

```sql
SELECT d.dept_id, d.dept_name, ROUND(AVG(s.cgpa), 2) AS avg_cgpa
FROM Department d
LEFT JOIN Student s ON d.dept_id = s.dept_id
GROUP BY d.dept_id, d.dept_name
ORDER BY d.dept_id;
```

Result:

dept_id ▲ 1	dept_name	avg_cgpa
1	CSE	3.76
2	EEE	3.50
3	BBA	3.20
4	CIVIL	*NULL*
5	ENG	*NULL*

Explanation: LEFT JOIN used to show departments without students (NULL mean no students). AVG rounded to 2 decimals.

7. Count the number of students in each department.

```sql
SELECT d.dept_id, d.dept_name, COUNT(s.student_id) AS num_students
FROM Department d
LEFT JOIN Student s ON d.dept_id = s.dept_id
GROUP BY d.dept_id, d.dept_name
ORDER BY d.dept_id;
```

Results:

dept_id ▲ 1	dept_name	num_students
1	CSE	3
2	EEE	1
3	BBA	1
4	CIVIL	0
5	ENG	0

Explanation: COUNT over joined rows; LEFT JOIN shows zeros.

8. Show the total credit hours each student has enrolled in.

```
SELECT s.student_id, s.student_name, COALESCE(SUM(c.credit),0) AS
total_credits
FROM Student s
LEFT JOIN Enrollment e ON s.student_id = e.student_id
LEFT JOIN Course c ON e.course_id = c.course_id
GROUP BY s.student_id, s.student_name
ORDER BY s.student_id;
```

Result:

student_id ▲ 1	student_name	total_credits
101	Rafiul Islam	6
102	Jannatul Ferdous	6
103	Salman Rahman	3
104	Nusrat Nahar	3
105	Aisha Khan	3

Explanation: sum of credits via joins; COALESCE handles students with zero enrollments.

9. Display the highest CGPA student in each department.

```
SELECT d.dept_id, d.dept_name, s.student_id, s.student_name, s.cgpa
FROM Department d
LEFT JOIN (
  SELECT student_id, student_name, dept_id, cgpa
  FROM Student
) s ON d.dept_id = s.dept_id
WHERE (s.cgpa IS NOT NULL AND s.cgpa = (
    SELECT MAX(s2.cgpa) FROM Student s2 WHERE s2.dept_id = d.dept_id
)) OR (s.cgpa IS NULL)
ORDER BY d.dept_id;
```

Result:

dept_id ▲ 1	dept_name	student_id	student_name	cgpa
1	CSE	102	Jannatul Ferdous	3.92
2	EEE	103	Salman Rahman	3.50
3	BBA	104	Nusrat Nahar	3.20
4	CIVIL	*NULL*	*NULL*	*NULL*
5	ENG	*NULL*	*NULL*	*NULL*

Explanation: For each department pick student(s) with the max CGPA. (If multiple students tie, the query returns ties.)

10. Find students who have enrolled in more than one course.

```sql
SELECT s.student_id, s.student_name, COUNT(e.course_id) AS courses_enrolled
FROM Student s
JOIN Enrollment e ON s.student_id = e.student_id
GROUP BY s.student_id, s.student_name
HAVING COUNT(e.course_id) > 1;
```

Result:

student_id	student_name	courses_enrolled
101	Rafiul Islam	2
102	Jannatul Ferdous	2

Explanation: GROUP BY student and HAVING count > 1 finds students enrolled multiple times.

Part 3: Subqueries & Conditional Queries

11. Display students whose CGPA is above the average CGPA of all students.

```sql
SELECT student_id, student_name, cgpa
FROM Student
WHERE cgpa > (SELECT AVG(cgpa) FROM Student);
```

Result:

student_id	student_name	cgpa
101	Rafiul Islam	3.75
102	Jannatul Ferdous	3.92
105	Aisha Khan	3.60

Explanation: simple scalar subquery returning overall average, then filter students above it.

12. List all courses not enrolled by any student.

```sql
SELECT c.course_id, c.course_name
FROM Course c
LEFT JOIN Enrollment e ON c.course_id = e.course_id
WHERE e.enroll_id IS NULL;
```

Result:

course_id	course_name
506	Signals

Explanation: LEFT JOIN + IS NULL finds courses with no matching enrollment.

13. Show students who have not enrolled in any course yet.

```sql
SELECT s.student_id, s.student_name
FROM Student s
LEFT JOIN Enrollment e ON s.student_id = e.student_id
WHERE e.enroll_id IS NULL;
```

Result:

No rows (empty set) — using our sample data every student (101..105) has at least one enrollment.

Explanation: This query would return students not present in Enrollment. In our dataset all 5 students are enrolled, so result is empty.

14. Find students who scored below 'A' (A-, B+, etc.) in Spring2025.

```sql
SELECT DISTINCT s.student_id, s.student_name, e.grade
FROM Enrollment e
JOIN Student s ON e.student_id = s.student_id
WHERE e.semester = 'Spring2025' AND e.grade <> 'A';
```

Result:

student_id	student_name	grade
101	Rafiul Islam	A-
103	Salman Rahman	B+
104	Nusrat Nahar	A-
105	Aisha Khan	B
102	Jannatul Ferdous	A-

Explanation: If you wanted only students whose *all* grades are below A, you'd need a different query. Here we list anyone who has any below-A grade.

15. Display the department name, course name, and total number of enrolled students using joins.

```sql
SELECT d.dept_name, c.course_name, COALESCE(COUNT(e.student_id),0) AS
enrolled_count
FROM Course c
JOIN Department d ON c.dept_id = d.dept_id
LEFT JOIN Enrollment e ON c.course_id = e.course_id
GROUP BY d.dept_name, c.course_name
ORDER BY d.dept_name, c.course_name;
```

Result:

dept_name ▲ 1	course_name	enrolled_count
BBA	Marketing 101	1
CSE	Algorithms	2
CSE	Data Structures	1
CSE	Database Systems	2
EEE	Circuit Analysis	1
EEE	Signals	0

Explanation: LEFT JOIN ensures courses with zero enrollments still appear with count 0.

Part 4: JOIN Operations

16. **Display each student's name and department name using a JOIN.**

```sql
SELECT s.student_name, d.dept_name
FROM Student s
JOIN Department d ON s.dept_id = d.dept_id;
```

Result:

student_name	dept_name
Rafiul Islam	CSE
Jannatul Ferdous	CSE
Salman Rahman	EEE
Nusrat Nahar	BBA
Aisha Khan	CSE

Explanation: basic INNER JOIN. same as Part1.1 (student + department pairs for all students).

17. **Show student names, course names, and grades for all enrollments.**

```sql
SELECT s.student_name, c.course_name, e.grade, e.semester
FROM Enrollment e
JOIN Student s ON e.student_id = s.student_id
JOIN Course c ON e.course_id = c.course_id
ORDER BY e.enroll_id;
```

Explanation: joins Enrollment → Student and Course to list the enrollment details.

Result:

student_name	course_name	grade	semester
Rafiul Islam	Database Systems	A	Spring2025
Rafiul Islam	Data Structures	A-	Spring2025
Jannatul Ferdous	Database Systems	A	Spring2025
Salman Rahman	Circuit Analysis	B+	Spring2025
Nusrat Nahar	Marketing 101	A-	Spring2025
Aisha Khan	Algorithms	B	Spring2025
Jannatul Ferdous	Algorithms	A-	Spring2025

18. Display department name, course name, and number of enrolled students (using LEFT JOIN).

```sql
SELECT d.dept_name, c.course_name, COUNT(e.enroll_id) AS num_enrolled
FROM Department d
JOIN Course c ON d.dept_id = c.dept_id
LEFT JOIN Enrollment e ON c.course_id = e.course_id
GROUP BY d.dept_name, c.course_name
ORDER BY d.dept_name, c.course_name;
```

Result:

dept_name ▲ 1	course_name	num_enrolled
BBA	Marketing 101	1
CSE	Algorithms	2
CSE	Data Structures	1
CSE	Database Systems	2
EEE	Circuit Analysis	1
EEE	Signals	0

Explanation: demonstrates JOIN + LEFT JOIN to include zero-enrollment courses.

19. **Show student name, department, and course for all enrollments in Spring2025.**

```sql
SELECT s.student_name, d.dept_name, c.course_name, e.grade
FROM Enrollment e
JOIN Student s ON e.student_id = s.student_id
JOIN Course c ON e.course_id = c.course_id
JOIN Department d ON s.dept_id = d.dept_id
WHERE e.semester = 'Spring2025';
```

Result:

student_name	dept_name	course_name	grade
Rafiul Islam	CSE	Database Systems	A
Rafiul Islam	CSE	Data Structures	A-
Jannatul Ferdous	CSE	Database Systems	A
Salman Rahman	EEE	Circuit Analysis	B+
Nusrat Nahar	BBA	Marketing 101	A-
Aisha Khan	CSE	Algorithms	B
Jannatul Ferdous	CSE	Algorithms	A-

Explanation: Filtering by semester after joining. same rows as 4.2 but includes department column; list per enrollment.

20. **List all students along with their enrolled courses, including those who haven't enrolled in any (using LEFT JOIN).**

```sql
SELECT s.student_id, s.student_name, c.course_name, e.grade
FROM Student s
LEFT JOIN Enrollment e ON s.student_id = e.student_id
LEFT JOIN Course c ON e.course_id = c.course_id
ORDER BY s.student_id;
```

Result:

student_id ▲ 1	student_name	course_name	grade
101	Rafiul Islam	Database Systems	A
101	Rafiul Islam	Data Structures	A-
102	Jannatul Ferdous	Database Systems	A
102	Jannatul Ferdous	Algorithms	A-
103	Salman Rahman	Circuit Analysis	B+
104	Nusrat Nahar	Marketing 101	A-
105	Aisha Khan	Algorithms	B

Explanation: LEFT JOIN ensures any student without enrollments would still appear with NULL course/grade. (In our dataset none have NULLs because everyone is enrolled.)

Part 5: Data Updates & Modifications

21. Update the CGPA of student ID = 104 to 3.35.

```
UPDATE Student
SET cgpa = 3.35
WHERE student_id = 104;
SELECT student_id, student_name, cgpa FROM Student WHERE student_id = 104;
```

Result:

student_id	student_name	cgpa
104	Nusrat Nahar	3.35

Explanation: simple UPDATE changing a single row.

22. Delete all enrollments where grade = 'C'.

```
1 DELETE FROM Enrollment WHERE grade = 'C';
2 SELECT * FROM Enrollment WHERE grade = 'C';
```

Result: In our sample data there are **no** enrollments with grade 'C', so **0 rows deleted**.

Explanation: safe delete; none matched.

23. Add a new column email to the Student table.

```sql
ALTER TABLE Student
ADD COLUMN email VARCHAR(100);
UPDATE Student SET email = CONCAT(LOWER(REPLACE(student_name,' ','.')),
'@university.edu');
```

Result:

student_id	student_name	dept_id	gender	cgpa	email
101	Rafiul Islam	1	M	3.75	rafiul.islam@university.edu
102	Jannatul Ferdous	1	F	3.92	jannatul.ferdous@university.edu
103	Salman Rahman	2	M	3.50	salman.rahman@university.edu
104	Nusrat Nahar	3	F	3.35	nusrat.nahar@university.edu
105	Aisha Khan	1	F	3.60	aisha.khan@university.edu

Explanation: adding column and example population step.

24. Change the location of the EEE department to 'Block D'.

```
UPDATE Department
SET location = 'Block D'
WHERE dept_name = 'EEE';
SELECT dept_id, dept_name, location FROM Department WHERE dept_name = 'EEE';
```

Result:

dept_id	dept_name	location
2	EEE	Block D

Explanation: straightforward update of a department row.

25. Create a view named CSE_Students showing only students from the CSE department.

```
CREATE VIEW CSE_Students AS
SELECT s.student_id, s.student_name, s.cgpa, s.gender
FROM Student s
JOIN Department d ON s.dept_id = d.dept_id
WHERE d.dept_name = 'CSE';
```

Result:

student_id	student_name	cgpa	gender
101	Rafiul Islam	3.75	M
102	Jannatul Ferdous	3.92	F
105	Aisha Khan	3.60	F

Explanation: the view simplifies frequent queries fetching CSE students.

Conclusion

Through this assignment, I learned how to create relational tables, insert records, and execute various SQL queries including joins, grouping, subqueries, and views.

Atik Jawad
ID: B220305043
(Thank You)