Experiment No. 1 - Database Management System (DBMS)

Topic: SQL Queries using DDL, DML, Operators, Functions, and SQL Objects

```
============================================================
■■ DATABASE CREATION
============================================================
-- Create a new database and use it
CREATE DATABASE student_db;
USE student_db;


============================================================
■ TABLE CREATION WITH CONSTRAINTS
============================================================
-- Create the student table with appropriate constraints
CREATE TABLE student (
    roll_no INT PRIMARY KEY,
    student_name VARCHAR(50) NOT NULL,
    student_department VARCHAR(50) NOT NULL,
    student_address VARCHAR(50),
    student_marks INT CHECK (student_marks BETWEEN 0 AND 100)
);


============================================================
■ DATA INSERTION
============================================================
-- Insert sample records into the student table
INSERT INTO student VALUES (25, 'Kiran', 'Civil', 'Kolhapur', 31);
INSERT INTO student VALUES (20, 'Prathamesh', 'AIDS', 'Pune', 30);
INSERT INTO student VALUES (29, 'Rahul', 'IT', 'Thane', 30);
INSERT INTO student VALUES (23, 'Om', 'ENTC', 'Pune', 29);
INSERT INTO student VALUES (21, 'Rohit', 'CSE', 'Mumbai', 28);
INSERT INTO student VALUES (24, 'Priya', 'Mechanical', 'Nashik', 27);
INSERT INTO student VALUES (26, 'Anjali', 'Electrical', 'Aurangabad', 26);
INSERT INTO student VALUES (22, 'Snehal', 'IT', 'Nagpur', 25);
INSERT INTO student VALUES (28, 'Neha', 'CSE', 'Pune', 24);


============================================================
■ SQL QUERIES WITH COMMENTS
============================================================

-- 1. Display all student records
SELECT * FROM student;

-- 2. Display students from Pune
SELECT * FROM student WHERE student_address = 'Pune';

-- 3. Display students having marks greater than 28
SELECT student_name, student_marks FROM student WHERE student_marks > 28;

-- 4. Display students sorted by marks in descending order
SELECT * FROM student ORDER BY student_marks DESC;

-- 5. Count number of students per department
SELECT student_department, COUNT(*) AS total_students FROM student GROUP BY student_department;

-- 6. Find average marks of students in each department
SELECT student_department, AVG(student_marks) AS avg_marks FROM student GROUP BY student_department;

-- 7. Find students having marks between 25 and 30
SELECT * FROM student WHERE student_marks BETWEEN 25 AND 30;

-- 8. Update marks of student “Neha” by +5
UPDATE student SET student_marks = student_marks + 5 WHERE student_name = 'Neha';

-- 9. Delete record of student “Anjali”
```

```sql
DELETE FROM student WHERE student_name = 'Anjali';

-- 10. Create a view to show only CSE students
CREATE VIEW cse_students AS
SELECT roll_no, student_name, student_marks FROM student WHERE student_department = 'CSE';

-- 11. Create an index for faster search by department
CREATE INDEX idx_department ON student(student_department);

-- 12. Create table using Auto Increment (sequence-like behavior)
CREATE TABLE student_seq (
    roll_no INT AUTO_INCREMENT PRIMARY KEY,
    student_name VARCHAR(50),
    student_department VARCHAR(50)
);

-- Insert values into auto increment table
INSERT INTO student_seq (student_name, student_department)
VALUES ('Kiran', 'Civil'), ('Rohit', 'CSE');

-- 13. Create synonym-like view (as MySQL alternative)
CREATE VIEW stud AS SELECT * FROM student;

-- Display data from synonym view
SELECT * FROM stud;

-- 14. Calculate total marks of all students
SELECT SUM(student_marks) AS total_marks FROM student;

-- 15. Display the student(s) with the highest marks
SELECT student_name, student_marks FROM student
WHERE student_marks = (SELECT MAX(student_marks) FROM student);
```

==
■ CONCLUSION
==

We successfully implemented DDL and DML commands using the student table.
All SQL queries, constraints, views, indexes, and aggregate functions were executed correctly.
This experiment demonstrates the ability to write and understand SQL queries efficiently.