Experiment_No: 2

```
-- Syntax:1. Create a table
Create table TY (Rl int not null primary key, sname varchar(25) not null);
-- Syntax:2. Alter a Table
ALTER TABLE TY rename column Rl to Roll_No;
ALTER TABLE TY ADD (rl_no int not null primary key);
ALTER TABLE TY ADD (marks int not null );
-- Syntax:3.Drop
ALTER TABLE TY Drop column Roll_No;
select * from TY;
-- Syntax:4.Insert data into table
INSERT INTO TY values("Ram",12,25);    -- single values
INSERT INTO TY VALUES ("Wali",9,26),("Jay",13,19),("AP",5,28),("Jay",15,29);   -- Multiple values
-- Syntax: 5 Update the table
UPDATE TY SET sname = "JOY"
WHERE rl_no = 13;
-- Syntax: 6 Delete a table data
DELETE FROM TY WHERE rl_no = 13;

-- Aggrigate Function
-- AVG(), COUNT(), MIN(), MAX()
SELECT * FROM TY;

SELECT AVG(marks) FROM TY;
SELECT MIN(marks) FROM TY;
SELECT MAX(marks) FROM TY;
SELECT COUNT(marks) FROM TY;
```

Experiment_No: 3

```
-- JOIN, AGGRIGATE FUNCTION, VIEWS

Create table employee (emp_id int not null primary key,emp_name varchar(35) not null,dept_id int);
Insert into employee values(101,"John Doe",1),(102,"Jane Smith",2),(103,"Alice Brown",NULL);

create table department(dept_id int not null primary key,dept_name varchar(20) not null);
insert into department values(1,"Sales"),(2,"HR"),(3,"IT");

-- Inner JOIN:
SELECT *,department.dept_name
FROM employee
INNER JOIN department
ON employee.dept_id = department.dept_id;

-- LEFT JOIN:
SELECT *, department.dept_name
FROM employee
LEFT JOIN department
ON employee.dept_id = department.dept_id;

-- RIGHT JOIN:
SELECT *, department.dept_name
FROM employee
RIGHT JOIN department
ON employee.dept_id = department.dept_id;

-- FULL OUTER JOIN:
SELECT e.emp_id, e.emp_name, e.dept_id, d.dept_name
FROM employee e
LEFT JOIN department d
```

```
ON e.dept_id = d.dept_id

UNION

SELECT e.emp_id, e.emp_name, e.dept_id, d.dept_name
FROM employee e
RIGHT JOIN department d
ON e.dept_id = d.dept_id;

-- Views
-- Syntax:
-- CREATE VIEW view_name AS
-- SELECT column1, column2, ...
-- FROM table_name
-- WHERE condition;
-- Example:
CREATE VIEW employees_without_department AS
SELECT emp_id, emp_name, dept_id
FROM employee
WHERE dept_id IS NULL;

SELECT * FROM employees_without_department;
```

Experiment_No: 4

```
 -- Indexes and Stored Procedures


 -- How to Create an Index
CREATE INDEX idx_dept_id ON employee(dept_id);
CREATE INDEX idx_dept_name ON department(dept_name);
-- How to Call or Use an Index
SELECT emp_name
FROM employee
WHERE dept_id = 1;
-- NOTE:- The index idx_dept_id will be used to quickly locate rows where dept_id = 1.
-- Checking if an Index is Used
EXPLAIN SELECT emp_name FROM employee WHERE dept_id = 1;


-- Stored Procedure
/*
A stored procedure is a set of SQL statements stored in the database that can
be executed as a single unit.
It simplifies repetitive tasks, promotes code reuse,
and enhances security by controlling access to data.
*/
DELIMITER //

CREATE PROCEDURE GetEmployeeDepartment()
BEGIN
   SELECT
       e.emp_id,
       e.emp_name,
       e.dept_id,
       d.dept_name
   FROM
```

```
        employee e
    LEFT JOIN
        department d
    ON
        e.dept_id = d.dept_id;
END //

DELIMITER ;
-- execute the stored procedure or call it
CALL GetEmployeeDepartment();

-- $$$$$ With Parameter
DELIMITER //

CREATE PROCEDURE UpdateEmployeeSalary(IN emp_id_param INT, IN increment DECIMAL(10, 2))
BEGIN
    UPDATE employee
    SET salary = salary + increment
    WHERE emp_id = emp_id_param;
END //

DELIMITER ;

ALTER TABLE employee ADD COLUMN salary DECIMAL(10, 2);

UPDATE employee SET salary = 0245
WHERE emp_id = 102;
-- execute the stored procedure or call it
CALL UpdateEmployeeSalary(102, 5000);
```

Experiment_No: 5

-- Write a code to implement User defined Functions on DB.

```
-- Without Parameter
DELIMITER //

CREATE FUNCTION GetTotalEmployees()
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE total INT;

   -- Count the total number of employees
   SELECT COUNT(*) INTO total
   FROM employee;

   RETURN total;
END //

DELIMITER ;
-- Call
SELECT GetTotalEmployees() AS TotalEmployees;

-- With Parameter
DELIMITER //

CREATE FUNCTION GetTotalSalaryByDept(dept_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE total_salary DECIMAL(10,2);
```

```
    -- Calculate the total salary for the given department ID
    SELECT SUM(salary) INTO total_salary
    FROM employee
    WHERE dept_id = dept_id;

    -- Handle case where no employees exist in the department
    IF total_salary IS NULL THEN
        RETURN 0; -- Default value if no salaries are found
    END IF;

    RETURN total_salary;
END //

DELIMITER ;
-- Call
SELECT GetTotalSalaryByDept(1) AS TotalSalary;
```