1) Create a table called Employee & execute the following. Employee(EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)

1. Create a user and grant all permissions to the user.

2. Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.

3. Add primary key constraint and not null constraint to the employee table.

4. Insert null values to the employee table and verify the result.

```sql
create database company;
use company;

create table employee(
empno int primary key,
ename varchar(255),
job varchar(255),
manager_no int,
sal decimal(10,2),
commision decimal(10,2)
);

create user anil @'localhost' identified by 'password';
grant all privileges on your_database_name.*to 'anil'@'localhost';

flush privileges;

insert into employee values(101,'Alice','Manager',null,75000,null);
insert into employee values(102,'Bob','Analyst',101,50000,5000);
insert into employee values(103,'Charlie','Clerk',102,30000,null);

select * from employee;
rollback;
select * from employee;

alter table employee
modify empno int primary key;

alter table employee
modify ename varchar(50) not null;

insert into employee
(empno,ename,job,manager_no,sal,commision)
values(104,'David','Salesman',null,null,null);
```

2) Create a table called Employee that contain attributes EMPNO,ENAME,JOB, MGR,SAL & execute the following.

1. Add a column commission with domain to the Employee table.
2. Insert any five records into the table.
3. Update the column details of job
4. Rename the column of Employ table using alter command.
5. Delete the employee whose Empno is 105.

```sql
create database company;
use company;

CREATE TABLE Employee (
EMPNO INT,
ENAME VARCHAR(50),
JOB VARCHAR(50),
MGR INT,
SAL DECIMAL(10, 2)
);

ALTER TABLE Employee
ADD COLUMN COMMISSION DECIMAL(10, 2);

INSERT INTO Employee (EMPNO, ENAME, JOB, MGR, SAL, COMMISSION) VALUES
(101, 'John Doe', 'Manager', NULL, 5000.00, NULL),
(102, 'Jane Smith', 'Developer', 101, 4000.00, NULL),
(103, 'Alice Johnson', 'Designer', 101, 4500.00, NULL),
(104, 'Bob Brown', 'Analyst', 101, 4200.00, NULL),
(105, 'Emma Wilson', 'Tester', 102, 3800.00, NULL);

UPDATE Employee
SET JOB = 'Senior Salesman'
WHERE EMPNO = 104;

ALTER TABLE Employee
RENAME COLUMN ENAME TO EMP_NAME;

DELETE FROM Employee
WHERE EMPNO = 105;
```

3) Queries using aggregate functions (COUNT,AVG,MIN,MAX,SUM),Group by, Order by.
Employee(E_id, E_name, Age, Salary)

1. Create Employee table containing all Records E_id, E_name, Age, Salary.

2. Count number of employee names from employee table

3. Find the Maximum age from employee table.

4. Find the Minimum age from employee table.

5. Find salaries of employee in Ascending Order.

6. Find grouped salaries of employees.

```sql
CREATE TABLE Employee (
E_id INT PRIMARY KEY,
E_name VARCHAR(50),
Age INT,
Salary DECIMAL(10, 2)
);

INSERT INTO Employee (E_id, E_name, Age, Salary) VALUES
(1, 'Alice', 30, 50000.00),
(2, 'Bob', 45, 60000.00),
(3, 'Charlie', 28, 45000.00),
(4, 'Diana', 35, 70000.00),
(5, 'Evan', 30, 50000.00);

SELECT COUNT(E_name) AS num_employees
FROM Employee;

SELECT MAX(Age) AS max_age
FROM Employee;

SELECT MIN(Age) AS min_age
FROM Employee;

SELECT E_name, Salary
FROM Employee
ORDER BY Salary ASC;

SELECT Salary, COUNT(*) AS num_employees
FROM Employee
GROUP BY Salary;
```

4) Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary. CUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)

```
create database dbms;
use dbms;

CREATE TABLE customers (
ID INT AUTO_INCREMENT PRIMARY KEY,
NAME VARCHAR(100),
AGE INT,
ADDRESS VARCHAR(255),
SALARY DECIMAL(10, 2)
);

 INSERT INTO customers (NAME, AGE, ADDRESS, SALARY) VALUES
('John Doe', 30, '123 Main St, Anytown, USA', 50000.00),
('Jane Smith', 25, '456 Elm St, Othertown, USA', 60000.00),
('Alice Johnson', 35, '789 Oak St, Anycity, USA', 70000.00),
('Bob Brown', 40, '321 Pine St, Anothercity, USA', 55000.00),
('Emily Davis', 28, '654 Maple St, Somewhere, USA', 65000.00),
('Michael Wilson', 45, '987 Cedar St, Nowhere, USA', 75000.00),
('Sarah Miller', 32, '159 Birch St, Anyplace, USA', 62000.00),
('David Jones', 38, '852 Walnut St, Elsewhere, USA', 68000.00),
('Jennifer Taylor', 27, '753 Cherry St, Elsewhere, USA', 59000.00),
('William Anderson', 33, '369 Oak St, Somewhere, USA', 71000.00);

 select *from customers;

DELIMITER $$

CREATE TRIGGER display_salary_changes_insert
BEFORE INSERT ON customers
FOR EACH ROW
BEGIN
    INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
     VALUES (NEW.ID, NULL, NEW.salary, NOW());
END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER display_salary_changes_update
BEFORE UPDATE ON customers
FOR EACH ROW
BEGIN
    INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
    VALUES (NEW.ID, OLD.salary, NEW.salary, NOW());
END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER display_salary_changes_delete
BEFORE DELETE ON customers
FOR EACH ROW
BEGIN
    INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
    VALUES (OLD.ID, OLD.salary, NULL, NOW());
END$$

DELIMITER ;

SHOW TRIGGERS;
```

5) Create cursor for Employee table & extract the values from the table. Declare the variables, Open the cursor & extract the values from the cursor. Close the cursor. Employee(E_id, E_name, Age, Salary).

```
CREATE DATABASE employee;
USE employee;

CREATE TABLE employee (
    E_id INT AUTO_INCREMENT PRIMARY KEY,
    E_name VARCHAR(100),
    Age INT,
    Salary DECIMAL(10, 2)
);

INSERT INTO employee (E_name, Age, Salary) VALUES
('John Doe', 30, 50000.00),
('Alice Smith', 25, 60000.00),
('Bob Johnson', 35, 70000.00);

DELIMITER //

CREATE PROCEDURE employee_cursor_proc()
BEGIN
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE emp_id INT;
    DECLARE emp_name VARCHAR(100);
    DECLARE emp_age INT;
    DECLARE emp_salary DECIMAL(10, 2);

    DECLARE emp_cursor CURSOR FOR
        SELECT E_id, E_name, Age, Salary FROM employee;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN emp_cursor;

    emp_loop: LOOP
        FETCH emp_cursor INTO emp_id, emp_name, emp_age, emp_salary;

        IF done THEN
            LEAVE emp_loop;
        END IF;

        SELECT CONCAT('Employee ID: ', emp_id, ', Name: ', emp_name, ', Age: ', emp_age, ', Salary: ', emp_salary) AS EmployeeInfo;

    END LOOP;

    CLOSE emp_cursor;
END//

DELIMITER ;

CALL employee_cursor_proc();
```

6) Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

```
CREATE DATABASE Employee;
USE Employee;
CREATE TABLE Employee (
    E_id INT AUTO_INCREMENT PRIMARY KEY,
    E_name VARCHAR(100),
    Age INT,
    Salary DECIMAL(10, 2)
);

INSERT INTO Employee (E_name, Age, Salary) VALUES
('John Doe', 30, 50000.00),
('Jane Smith', 28, 60000.00),
('Michael Johnson', 35, 75000.00);

DELIMITER //

CREATE PROCEDURE merge_data()
BEGIN
    DECLARE v_n_rollcall_id INT;
    DECLARE v_n_rollcall_data VARCHAR(255);
    DECLARE v_exists INT;
    DECLARE done BOOLEAN DEFAULT FALSE;

    DECLARE n_rollcall_cursor CURSOR FOR
        SELECT ID, Data FROM N_RollCall;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    OPEN n_rollcall_cursor;

    n_rollcall_loop: LOOP
        FETCH n_rollcall_cursor INTO v_n_rollcall_id, v_n_rollcall_data;

        IF done THEN
            LEAVE n_rollcall_loop;
        END IF;

        SELECT COUNT(*) INTO v_exists
        FROM O_RollCall
        WHERE Data = v_n_rollcall_data;

        IF v_exists = 0 THEN
            INSERT INTO O_RollCall (ID, Data)
            VALUES (v_n_rollcall_id, v_n_rollcall_data);
        ELSE
            SELECT CONCAT('Data with ID ', v_n_rollcall_id, ' already exists in O_RollCall. Skipping...') AS Message;
        END IF;
    END LOOP;

    CLOSE n_rollcall_cursor;
END //

DELIMITER ;
```

7) Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations