RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Recognized by GOK, Approved by AICTE, New Delhi
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098
Ph:(080)28611880,28611881 URL: www.rnsit.ac.in

DEPARTMENT OF AI & ML

DATABASE MANAGEMENT SYSTEMS LAB MANUAL

(BCSL403)

(As per Visvesvaraya Technological University Course type- PCCL)

DEPARTMENT OF CSE(AI & ML)

R N S Institute of Technology

Bengaluru-98

Name: _____

USN: _____

RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Recognized by GOK, Approved by AICTE, New Delhi
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098
Ph:(080)28611880,28611881 URL: www.rnsit.ac.in

DEPARTMENT OF AI & ML

Vision of the Department

Empowering AI & ML Engineers to seamlessly integrate society and technology.

Mission of the Department

The Department of AI&ML will make every effort to promote an intellectual and ethical environment by

- To Inculcate, strong mathematical foundations as applied to AIML domain.
- To Equip AIML graduates with skills to meet Industrial and Societal challenges.
- To Foster ethical values & engineering norms and standards in AIML graduates.

Disclaimer

Trademark

Estd : 2001

Edition: 2023- 24

Document Owner

The primary contact for questions regarding this document is:

Author(s):	1. Prof. POOJA M 2. Prof. KAVYASHREE H L
Department:	**CSE (AI & ML)**
Contact email ids :	pooja.m@rnsit.ac.in kavyashree.hl@rnsit.ac.in

PROGRAM LIST AND CONDUCTION PLAN

Sl. NO.	*Date Week*	*Program Description*	*Page No.*
1	27/04/2024 29/04/2024	***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 theuser.*** ***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.***	*10*
2	4/05/2024 6/05/2024	***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 Employeetable.*** ***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***	*12*
3	11/05/2024 13/05/2024	**Queries using *aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Orderby.*** ***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 employeetable*** ***3. Find the Maximum age from employee table.*** ***4. Find the Minimum age from employeetable.*** ***5. Find salaries of employee in Ascending Order.*** ***6. Find grouped salaries of employees.***	*14*

4	25/05/2024 27/05/2024	***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)***	*18*
5	25/06/2024 27/06/2024	***Create cursor for Employee table & extract the values from the table. Declare the variables ,Open the cursor & extrct the values from the cursor. Close the cursor. Employee(E_id, E_name, Age, Salary)***	*21*
6	25/06/2024 27/06/2024	***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 skippe***	
7	25/06/2024 27/06/2024	***Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.***	

Course objectives:

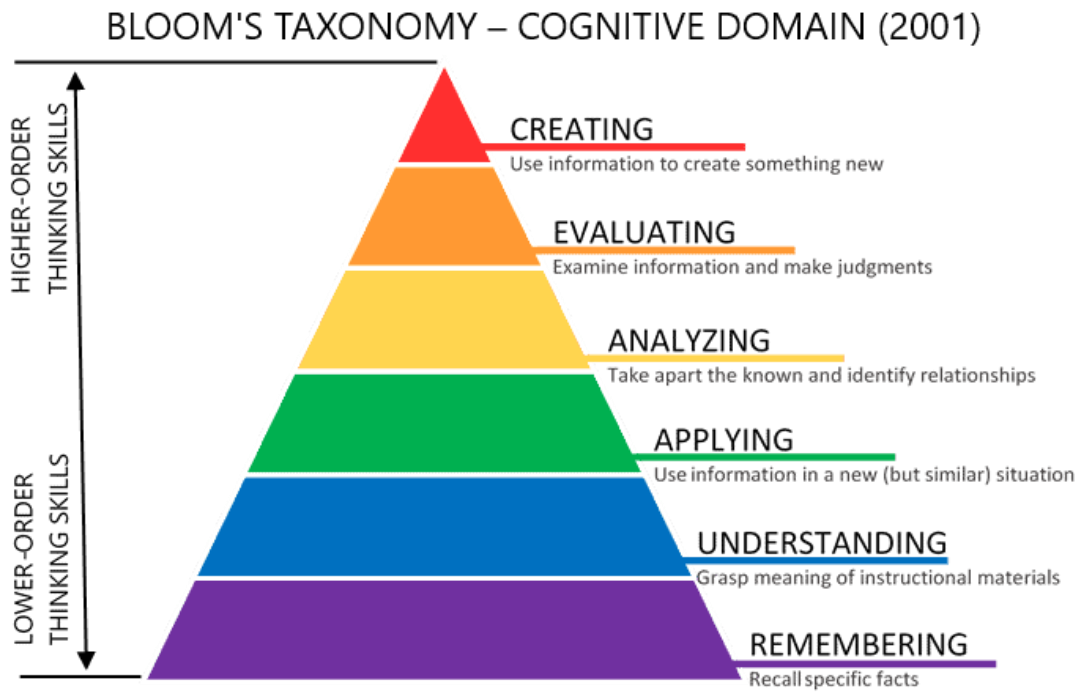
- To Provide a strong foundation in database concepts, technology, and practice.
- To Practice SQL programming through a variety of database problems.
- To Understand the relational database design principles.
- To Demonstrate the use of concurrency and transactions in database.
- To Design and build database applications for real world problems.
- To become familiar with database storage structures and access techniques

Course outcomes (Course Skill Set): At the end of the course, the student will be able to:

- Describe the basic elements of a relational database management system
- Design entity relationship for the given scenario.
- Apply various Structured Query Language (SQL) statements for database manipulation.

- Analyse various normalization forms for the given application.
- Develop database applications for the given real world problem.
- Understand the concepts related to NoSQL databases

REVISED BLOOMS TAXONOMY (RBT)



Program 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.

Create a user and grant all permissions to the user

```
show databases;

use gg;

show tables;

SELECT host, user, authentication_string AS password FROM mysql.user;

create user 'po'@'localhost' identified by 'root';

SELECT host, user, authentication_string AS password FROM mysql.user;

CREATE USER 'po'@'localhost' IDENTIFIED BY 'root';

GRANT ALL PRIVILEGES ON mysql.* TO 'po'@'localhost';
```

-- Step 1: Creating the Employee table

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

-- Step 2: Creating a user and granting permissions

```
CREATE USER the_user IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON Employee TO the_user;
```

-- Step 3: Inserting three records into the Employee table

INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER_NO, SAL, COMMISSION) VALUES

(1, 'John Doe', 'Manager', NULL, 50000.00, 1000.00),

(2, 'Jane Smith', 'Assistant', 1, 40000.00, 800.00),

(3, 'Michael Johnson', 'Clerk', 2, 30000.00, 600.00);

-- Starting a transaction

START TRANSACTION;

-- Deleting a record from the Employee table

DELETE FROM Employee WHERE EMPNO = 1;

-- Verify that the row is deleted

SELECT * FROM Employee;

```
mysql> DELETE FROM Employee WHERE EMPNO = 1;
Query OK, 1 row affected (0.00 sec)

mysql> select * from Employee;
+-------+----------------+-----------+------------+---------+------------+
| EMPNO | ENAME          | JOB       | MANAGER_NO | SAL     | COMMISSION |
+-------+----------------+-----------+------------+---------+------------+
|     2 | Jane Smith     | Assistant |          1 | 40000.00 |     800.00 |
|     3 | Michael Johnson | Clerk     |          2 | 30000.00 |     600.00 |
+-------+----------------+-----------+------------+---------+------------+
2 rows in set (0.00 sec)
```

-- Rolling back the changes

ROLLBACK;

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from Employee;
+-------+----------------+-----------+------------+---------+------------+
| EMPNO | ENAME          | JOB       | MANAGER_NO | SAL     | COMMISSION |
+-------+----------------+-----------+------------+---------+------------+
|     1 | John Doe       | Manager   |       NULL | 50000.00 |    1000.00 |
|     2 | Jane Smith     | Assistant |          1 | 40000.00 |     800.00 |
|     3 | Michael Johnson | Clerk     |          2 | 30000.00 |     600.00 |
+-------+----------------+-----------+------------+---------+------------+
3 rows in set (0.00 sec)
```

-- Verify that the row is restored

SELECT * FROM Employee;

-- Step 4: Adding primary key constraint and not null constraint

```
ALTER TABLE Employee
ADD CONSTRAINT pk_Employee PRIMARY KEY (EMPNO),
MODIFY EMPNO INT NOT NULL,
MODIFY ENAME VARCHAR(50) NOT NULL,
MODIFY JOB VARCHAR(50) NOT NULL,
MODIFY SAL DECIMAL(10, 2) NOT NULL;
desc employee;
```

```
mysql> ALTER TABLE Employee
    -> ADD CONSTRAINT pk_Employee PRIMARY KEY (EMPNO),
    -> MODIFY EMPNO INT NOT NULL,
    -> MODIFY ENAME VARCHAR(50) NOT NULL,
    -> MODIFY JOB VARCHAR(50) NOT NULL,
    -> MODIFY SAL DECIMAL(10, 2) NOT NULL;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc employee;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| EMPNO      | int           | NO   | PRI | NULL    |       |
| ENAME      | varchar(50)   | NO   |     | NULL    |       |
| JOB        | varchar(50)   | NO   |     | NULL    |       |
| MANAGER_NO | int           | YES  |     | NULL    |       |
| SAL        | decimal(10,2) | NO   |     | NULL    |       |
| COMMISSION | decimal(10,2) | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

-- Inserting null values to the Employee table

```
INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER_NO, SAL, COMMISSION) VALUES
(NULL, NULL, NULL, NULL, NULL, NULL);
```

-- Verifying the result

```
SELECT * FROM Employee;
```

Program 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 Employeetable. 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.

```
-- 1. Create the Employee table with the given attributes
CREATE TABLE Employee (
  EMPNO INT,
  ENAME VARCHAR(50),
  JOB VARCHAR(50),
  MANAGER_ID INT,
  SAL DECIMAL(10,2),
  COMM DECIMAL(10,2)
);

-- 2. Add a column 'COMM' for commission with the appropriate domain
ALTER TABLE Employee
ADD COLUMN COMM DECIMAL(10,2);

-- 3. Insert five records into the Employee table
INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER_ID, SAL, COMM)
VALUES
  (101, 'John Doe', 'Manager', NULL, 5000.00, 500.00),
  (102, 'Jane Smith', 'Developer', 101, 4500.00, 400.00),
  (103, 'Michael Johnson', 'Analyst', 101, 4000.00, 300.00),
  (104, 'Emily Brown', 'Designer', 102, 3800.00, 250.00),
  (105, 'David Lee', 'Intern', 103, 2500.00, 150.00);
```

```
-- 4. Update the job details
UPDATE Employee
SET JOB = 'Senior Manager'
WHERE EMPNO = 101;

UPDATE Employee
SET JOB = 'Senior Developer'
WHERE EMPNO = 102;

UPDATE Employee
SET JOB = 'Senior Analyst'
WHERE EMPNO = 103;

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

UPDATE Employee
SET JOB = 'Associate Intern'
WHERE EMPNO = 105;

-- 5. Rename the column 'EMPNO' to 'Employee_ID'
ALTER TABLE Employee
RENAME COLUMN EMPNO TO Employee_ID;

-- 6. Delete the employee with Employee_ID 105
DELETE FROM Employee
WHERE Employee_ID = 105;

-- Display the final result
```

SELECT * FROM Employee;

OUTPUT

+--------------+------------+--------------+------------+----------+------+

| Employee_ID | ENAME | JOB | MANAGER_ID | SAL | COMM |

+--------------+------------+--------------+------------+----------+------+

| 101 | John Doe | Senior Manager| NULL | 5000.00 | 500.00|

| 102 | Jane Smith | Senior Developer| 101 | 4500.00 | 400.00|

| 103 | Michael Johnson| Senior Analyst| 101 | 4000.00 | 300.00|

| 104 | Emily Brown | Senior Designer| 102 | 3800.00 | 250.00|


```
mysql> select * from employee;
+-------+-----------------+------------------+------------+---------+--------+
| EMPNO | ENAME           | JOB              | MANAGER_ID | SAL     | COMM   |
+-------+-----------------+------------------+------------+---------+--------+
|   101 | John Doe        | Senior Manager   |       NULL | 5000.00 | 500.00 |
|   102 | Jane Smith      | Senior Developer |        101 | 4500.00 | 400.00 |
|   103 | Michael Johnson | Senior Analyst   |        101 | 4000.00 | 300.00 |
|   104 | Emily Brown     | Senior Designer  |        102 | 3800.00 | 250.00 |
+-------+-----------------+------------------+------------+---------+--------+
4 rows in set (0.00 sec)
```

Program 3

Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Orderby. 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.

```
-- 1. Create the Employee table with the given attributes
CREATE TABLE Employee (
  E_id INT,
  E_name VARCHAR(50),
  Age INT,
  Salary DECIMAL(10,2)
);

-- Insert sample data into the Employee table
INSERT INTO Employee (E_id, E_name, Age, Salary)
VALUES
  (101, 'John Doe', 35, 5000.00),
  (102, 'Jane Smith', 28, 4500.00),
  (103, 'Michael Johnson', 42, 4000.00),
  (104, 'Emily Brown', 31, 3800.00),
  (105, 'David Lee', 25, 2500.00),
  (106, 'Sarah Williams', 38, 4200.00),
  (107, 'Robert Davis', 29, 3900.00);

-- 2. Count the number of employee names from the Employee table
SELECT COUNT(E_name) AS TotalEmployees
FROM Employee;

-- 3. Find the Maximum age from the Employee table
SELECT MAX(Age) AS MaxAge
FROM Employee;
```

-- 4. Find the Minimum age from the Employee table

SELECT MIN(Age) AS MinAge

FROM Employee;

-- 5. Find salaries of employees in Ascending Order

SELECT Salary

FROM Employee

ORDER BY Salary ASC;

-- 6. Find grouped salaries of employees

SELECT Salary, COUNT(*) AS EmployeeCount

FROM Employee

GROUP BY Salary

ORDER BY Salary;

Output:

```
mysql> CREATE TABLE Employee1 (
    ->   E_id INT,
    ->   E_name VARCHAR(50),
    ->   Age INT,
    ->   Salary DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO Employee1 (E_id, E_name, Age, Salary)
    -> VALUES
    ->   (101, 'John Doe', 35, 5000.00),
    ->   (102, 'Jane Smith', 28, 4500.00),
    ->   (103, 'Michael Johnson', 42, 4000.00),
    ->   (104, 'Emily Brown', 31, 3800.00),
    ->   (105, 'David Lee', 25, 2500.00),
    ->   (106, 'Sarah Williams', 38, 4200.00),
    ->   (107, 'Robert Davis', 29, 3900.00);
Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> SELECT COUNT(E_name) AS TotalEmployees
    -> FROM Employee;
ERROR 1054 (42S22): Unknown column 'E_name' in 'field list'
mysql> SELECT COUNT(E_name) AS TotalEmployees
    -> FROM Employee1;
+----------------+
| TotalEmployees |
+----------------+
|              7 |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT MAX(Age) AS MaxAge
    -> FROM Employee1;
+--------+
| MaxAge |
+--------+
|     42 |
+--------+
1 row in set (0.00 sec)

mysql> SELECT MIN(Age) AS MINAge
    -> FROM Employee1;
+--------+
| MINAge |
+--------+
|     25 |
+--------+
1 row in set (0.00 sec)

mysql> SELECT Salary
    -> FROM Employee1
    -> ORDER BY Salary ASC;
+---------+
| Salary  |
+---------+
| 2500.00 |
| 3800.00 |
| 3900.00 |
| 4000.00 |
| 4200.00 |
| 4500.00 |
| 5000.00 |
+---------+
7 rows in set (0.00 sec)
```

```
mysql> SELECT Salary, COUNT(*) AS EmployeeCount
    -> FROM Employee1
    -> GROUP BY Salary
    -> ORDER BY Salary;
+---------+---------------+
| Salary  | EmployeeCount |
+---------+---------------+
| 2500.00 |             1 |
| 3800.00 |             1 |
| 3900.00 |             1 |
| 4000.00 |             1 |
| 4200.00 |             1 |
| 4500.00 |             1 |
| 5000.00 |             1 |
+---------+---------------+
7 rows in set (0.00 sec)
```

Program 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 TABLE CUSTOMERS2 (
  ID INT,
  NAME VARCHAR(50),
  AGE INT,
  ADDRESS VARCHAR(100),
  SALARY DECIMAL(10,2)
);

INSERT INTO CUSTOMERS2 VALUES (1, 'Ramesh', 23, 'Allahabad', 20000.00);
INSERT INTO CUSTOMERS2 VALUES (2, 'Suresh', 22, 'Kanpur', 22000.00);
INSERT INTO CUSTOMERS2 VALUES (3, 'Mahesh', 24, 'Ghaziabad', 24000.00);
INSERT INTO CUSTOMERS2 VALUES (4, 'Chandan', 25, 'Noida', 26000.00);
INSERT INTO CUSTOMERS2 VALUES (5, 'Alex', 21, 'Paris', 28000.00);
INSERT INTO CUSTOMERS2 VALUES (6, 'Sunita', 20, 'Delhi', 30000.00);

SELECT * FROM CUSTOMERS2;

DELIMITER //
CREATE TRIGGER display_salary_changes
BEFORE UPDATE ON CUSTOMERS
FOR EACH ROW
BEGIN
  DECLARE sal_diff DECIMAL(10,2);
  SET sal_diff = NEW.SALARY - OLD.SALARY;
  SELECT CONCAT('Old salary: ', OLD.SALARY),
         CONCAT('New salary: ', NEW.SALARY),
         CONCAT('Salary difference: ', sal_diff)
  INTO @old_salary, @new_salary, @sal_diff;
END//
DELIMITER ;

DECLARE @old_salary VARCHAR(50), @new_salary VARCHAR(50), @sal_diff
VARCHAR(50);
UPDATE CUSTOMERS
SET SALARY = SALARY + 5000;
SELECT @old_salary, @new_salary, @sal_diff;


-- Check the salary difference by procedure
BEGIN;
UPDATE CUSTOMERS
SET SALARY = SALARY + 5000.00;
COMMIT;
```

```
mysql> SELECT @old_salary, @new_salary, @sal_diff;
+----------------------+----------------------+---------------------------+
| @old_salary          | @new_salary          | @sal_diff                 |
+----------------------+----------------------+---------------------------+
| Old salary: 40000.00 | New salary: 45000.00 | Salary difference: 5000.00 |
+----------------------+----------------------+---------------------------+
1 row in set (0.00 sec)

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE CUSTOMERS
    -> SET SALARY = SALARY + 5000.00;
Query OK, 6 rows affected (0.00 sec)
Rows matched: 6  Changed: 6  Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>  SELECT @old_salary, @new_salary, @sal_diff;
+----------------------+----------------------+---------------------------+
| @old_salary          | @new_salary          | @sal_diff                 |
+----------------------+----------------------+---------------------------+
| Old salary: 45000.00 | New salary: 50000.00 | Salary difference: 5000.00 |
+----------------------+----------------------+---------------------------+
1 row in set (0.00 sec)
```

Program 5

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

```
CREATE TABLE employees (
    employee_id INTEGER,
    first_name VARCHAR(25),
    last_name VARCHAR(25),
    email VARCHAR(25),
    phone_number VARCHAR(15),
    hire_date DATE,
    job_id VARCHAR(25),
    salary INTEGER,
    commission_pct DECIMAL(5,2),
    manager_id INTEGER,
    department_id INTEGER
);
```

- **Insert values**

```
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, commission_pct, manager_id, department_id)

VALUES

    (1, 'John', 'Doe', 'john.doe@example.com', '1234567890', '2022-01-01', 'IT_PROG', 5000,
0.05, NULL, 10),

    (2, 'Jane', 'Smith', 'jane.smith@example.com', '9876543210', '2022-02-01', 'HR_REP',
6000, 0.03, 1, 20),

    -- Add more rows as needed

    (100, 'Max', 'Johnson', 'max.johnson@example.com', '5555555555', '2022-03-01',
'SA_REP', 8000, 0.08, 2, 30);


DELIMITER //


CREATE PROCEDURE retrieve_employee_data()

BEGIN

    DECLARE done INT DEFAULT FALSE;

    DECLARE emp_id INT;

    DECLARE emp_first_name VARCHAR(25);
```

```
DECLARE emp_last_name VARCHAR(25);
DECLARE emp_email VARCHAR(25);
DECLARE emp_phone_number VARCHAR(15);
DECLARE emp_hire_date DATE;
DECLARE emp_job_id VARCHAR(25);
DECLARE emp_salary INT;
DECLARE emp_commission_pct DECIMAL(5,2);
DECLARE emp_manager_id INT;
DECLARE emp_department_id INT;

-- Declare cursor
DECLARE emp_cursor CURSOR FOR
  SELECT employee_id, first_name, last_name, email, phone_number,
      hire_date, job_id, salary, commission_pct, manager_id, department_id
  FROM employees;
-- Declare continue handler
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

-- Open cursor
OPEN emp_cursor;

-- Fetch data from cursor and display
read_loop: LOOP
  FETCH emp_cursor INTO
    emp_id, emp_first_name, emp_last_name, emp_email, emp_phone_number,
    emp_hire_date, emp_job_id, emp_salary, emp_commission_pct,
    emp_manager_id, emp_department_id;

  IF done THEN
    LEAVE read_loop;
  END IF;
```

```
    -- Display employee information
    SELECT CONCAT('Employee ID: ', emp_id);
    SELECT CONCAT('Employee Name: ', CONCAT(emp_first_name, ' ', emp_last_name));
    SELECT CONCAT('Email: ', emp_email);
    SELECT CONCAT('Phone Number: ', emp_phone_number);
    SELECT CONCAT('Hire Date: ', DATE_FORMAT(emp_hire_date, '%d-%b-%Y'));
    SELECT CONCAT('Job ID: ', emp_job_id);
    SELECT CONCAT('Salary: ', emp_salary);
    SELECT CONCAT('Commission Pct: ', emp_commission_pct);
    SELECT CONCAT('Manager ID: ', emp_manager_id);
    SELECT CONCAT('Department ID: ', emp_department_id);
    SELECT '------------------------';
  END LOOP;

  -- Close cursor
  CLOSE emp_cursor;

END//

DELIMITER ;
```

To call this stored procedure and see the output, you can use the following SQL command:

```
CALL retrieve_employee_data();
```

OUTPUT

```
mysql> CALL retrieve_employee_data();
+---------------------------------+
| CONCAT('Employee ID: ', emp_id) |
+---------------------------------+
| Employee ID: 1                  |
+---------------------------------+
1 row in set (0.00 sec)

+-----------------------------------------------------------------------+
| CONCAT('Employee Name: ', CONCAT(emp_first_name, ' ', emp_last_name)) |
+-----------------------------------------------------------------------+
| Employee Name: John Doe                                               |
+-----------------------------------------------------------------------+
1 row in set (0.01 sec)

+-----------------------------+
| CONCAT('Email: ', emp_email) |
+-----------------------------+
| Email: john.doe@example.com  |
+-----------------------------+
1 row in set (0.01 sec)

+-------------------------------------------+
| CONCAT('Phone Number: ', emp_phone_number) |
+-------------------------------------------+
| Phone Number: 1234567890                  |
+-------------------------------------------+
1 row in set (0.01 sec)

+-------------------------------------------------------------+
| CONCAT('Hire Date: ', DATE_FORMAT(emp_hire_date, '%d-%b-%Y')) |
+-------------------------------------------------------------+
| Hire Date: 01-Jan-2022                                      |
+-------------------------------------------------------------+
1 row in set (0.01 sec)

+---------------------------------+
| CONCAT('Job ID: ', emp_job_id) |
+---------------------------------+
```

PROGRAM 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 assi8;
use assi8;
show tables;
create table old_roll(roll int,name varchar(10));
create table new_roll(roll int,name varchar(10));
insert into old_roll values(4,'d');
insert into old_roll values(3,'bcd');
insert into old_roll values(1,'bc');
insert into old_roll values(5,'bch');
insert into new_roll values(2,'b');
insert into new_roll values(5,'bch');
insert into new_roll values(1,'bc');

select * from old_roll;
select * from new_roll;
delimiter $
create procedure roll_list()
begin
declare oldrollnumber int;
declare oldname varchar(10);
declare newrollnumber int;
declare newname varchar(10);
declare done int default false;
declare c1 cursor for select roll,name from old_roll;
declare c2 cursor for select roll,name from new_roll;
declare continue handler for not found set done=true;
open c1;
```

```
loop1:loop
fetch c1 into oldrollnumber,oldname;
if done then
leave loop1;
end if;
open c2;

loop2:loop
fetch c2 into newrollnumber,newname;
if done then
insert into new_roll values(oldrollnumber,oldname);
set done=false;
close c2;
leave loop2;
end if;
if oldrollnumber=newrollnumber then
leave loop2;
end if;
end loop;
end loop;
close c1;
end $
delimiter ;
call roll_list();
select * from new_roll;
```

Explanation:

- The procedure `roll_list()` is created within the `assi7` database.
- It declares variables for storing roll numbers and names from both tables.
- Cursors `c1` and `c2` are declared to fetch records from `old_roll` and `new_roll`, respectively.

- A handler is set to manage the situation when no more records are found in a cursor.
- The procedure opens cursor `c1` and iterates through each record in `old_roll`.
- For each record in `old_roll`, it opens cursor `c2` and compares the roll numbers with records in `new_roll`.
- If the record doesn't exist in `new_roll`, it's inserted.
- After processing all records, the cursors are closed.

```
mysql> select * from new_roll;
+------+------+
| roll | name |
+------+------+
|    2 | b    |
|    5 | bch  |
|    1 | bc   |
|    4 | d    |
|    3 | bcd  |
|    2 | b    |
|    5 | bch  |
|    1 | bc   |
|    2 | b    |
|    5 | bch  |
|    1 | bc   |
+------+------+
11 rows in set (0.00 sec)
```

Program 7

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