ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ, ಬೆಳಗಾವಿ

VISVESVARAYA TECHNOLOGICAL UNIVERSITY - BELAGAVI

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. ROAD, V.V PURAM, BENGALURU – 560 004

INFORMATION SCIENCE AND ENGINEERING

(AFFILIATED TO VTU, BELAGAVI)

CODE: BCS403

DATABASE MANAGEMENT SYSTEM LAB

As per Choice Based Credit System Scheme (CBCS) FOR

IV SEMESTER CSE/ISE AS PRESCRIBED BY VTU

Academic year 2023-2024

Prepared By:

Prof. Chethana M
Assistant Professor
Dept of ISE, BIT

Prof. Pavithra N
Assistant Professor
Dept of ISE, BIT

BANGALORE INSTITUTE OF TECHNOLOGY

VISION:

Establish and develop the Institute as the Centre of higher learning, ever abreast with expanding horizon of knowledge in the field of Engineering and Technology with entrepreneurial thinking, leadership excellence for life-long success and solve societal problems.

MISSION:

- Provide high quality education in the Engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.
- Develop the Institute as a leader in Science, Engineering, Technology, Management and Research and apply knowledge for the benefit of society.
- Establish mutual beneficial partnerships with Industry, Alumni, Local, State and Central Governments by Public Service Assistance and Collaborative Research.
- Inculcate personality development through sports, cultural and extracurricular activities and engage in social, economic and professional challenges.

Bangalore Institute of Technology

K. R. Road, V. V. Pura, Bengaluru- 560004

Department of Information Science and Engineering

VISION:

Empower every student to be innovative, creative and productive in the field of Information Technology by imparting quality technical education, developing skills andinculcating human values.

MISSION:

M1	To evolve continually as a Centre of Excellence in offering quality Information Technology **Education**.
M2	To nurture the students to meet the global competency in industry for **Employment**.
M3	To promote collaboration with industry and academia for constructive interaction to empower **Entrepreneurship.**
M4	To provide reliable, contemporary and integrated technology to support and facilitate **Teaching, Learning, Research and Service.**

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-1	Uplift the students through Information Technology **Education.**
PEO-2	Provide exposure to emerging technologies and train them to **Employable** in multi-disciplinary industries.
PEO-3	Motivate them to become good professional Engineers and **Entrepreneur.**
PEO-4	Inspire them to prepare for **Higher Learning and Research.**

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1	To provide our graduates **with Core Competence in Information Processing and Management.**
PSO-2	To provide our graduates with Higher Learning in **Computing Skills.**

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Bangalore Institute of Technology

K. R. Road, VV Pura, Bangalore 560004

Department of Information Science and Engineering

DBMS Laboratory (BCS403)

Pre-requisites:

Discrete Mathematics, Programming Language, Good understanding of computerc oncepts such as primary memory, secondary memory and data structures.

Software Used:

Oracle, MySQL, MS SQL Server or any other DBMS under LINUX/Windows environment.

Course Objectives:

This course will enable students to:

. 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 (CO):

The students are able to:

Describe the basic elements of a relational database management system

1. Describe the basic elements of a relational database management system.
2. Design entity relationship for the given scenario.
3. Apply various Structured Query Language (SQL) statements for database manipulation
4. Analyze various normalization forms for the given application.
5. Develop database applications for the given real-world problem
6. Understand the concepts related to NoSQL databases.

e. Exit

Support the program with appropriate functions for each of the above operations

Mapping of COs-POs and COs-PSOs

DBMS Laboratory (BCS403)

CO-PO and CO-PSO Mapping of DBMS Laboratory (BCS403)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1										1		
CO2	2									1		
CO3	3	3									2	2
CO4	2											
CO5	3	3	1		1				2	2		2
CO6	2									1		2

		PSO1	PSO2
DBMS Laboratory (BCS403)	CO1	2	2
	CO2	3	3
	CO3	2	2
	CO4	3	3
	CO5	3	3
	CO6	2	2

DBMS LABORATORY
[As per Choice Based Credit System (CBCS) scheme](Effective from the academic year 2023 -2024) SEMESTER – IV

Subject Code	BCS403	CIE Marks	25
Teaching Hours/Week (L:T:P: S)	0:0:2:0		
Total Hours of Pedagogy	20	Total Marks	25
Credits	01	Exam Hours	02

Course Learning 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.

1

SQL Programming (Max. Exam Marks. 25)

Design, develop, and implement the specified queries for the following problems using Oracle, MySQL, MS SQL Server, or any other DBMS under LINUX/Windows environment.Create Schema and insert at least 5 records for each table. Add appropriate database constraints.

Aim: Demonstrating creation of tables, applying the primary key constraints and NOT NULL constraints.

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 contain 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.

2	Aim: Discuss the update operations. 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.
3	Aim: Demonstrate the concepts of aggregate functions. Queries using aggregate functions (COUNT, AVG, MIN, MAX, SUM), Group byOrderby. **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.
4	Aim: Introduce concepts of trigger. 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)**
5	Aim: Demonstrate the core concepts on cursor 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).**
6	Aim: Demonstrate the core concepts on PL/SQL Write a PL/SQL block of code using parameterized Cursor, that will merge the data availablein 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.

7	Aim: Demonstrate the Installation of MongoDB Install an Open Source NoSQL Data base MangoDB & perform basic CRUD (Create,Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.

Pedagogy: For the above experiments the following pedagogy can be considered. Problem based learning, Active learning, MOOC, Chalk & Talk.

CIE for the practical component of the IPCC

- **15 marks** for the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voceand marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to **15 marks**.
- The laboratory test **(duration 02/03 hours)** after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks.**
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

1. Create a table called Employee & execute the following.

   Employee(EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)

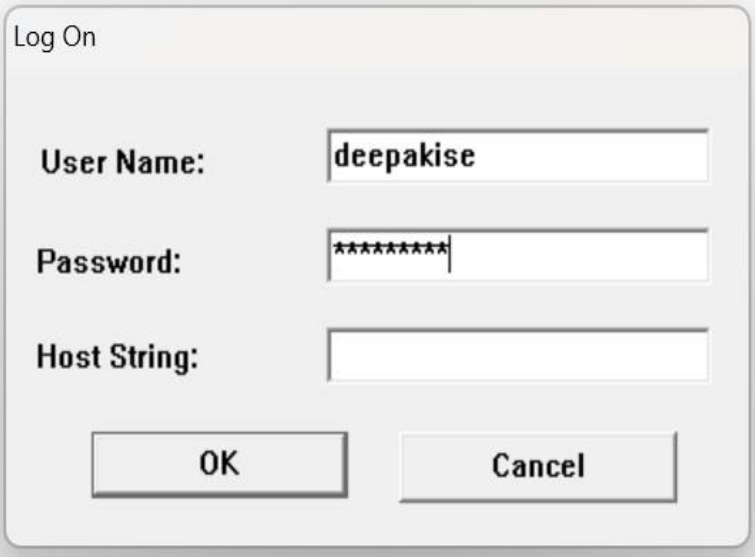
   1. Create a user and grant all permissions to theuser.
   2. Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.
   3. Add primary key constraint and not null constraint to the employee table.
   4. Insert null values to the employee table and verify the result.

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

```
Connect sys as sysdba;

Enter password: ise123

create user deepakise identified by deepak123;

GRANT CONNECT, RESOURCE TO deepakise;

GRANT SELECT, INSERT, UPDATE, DELETE ON Employee To deepakise;

GRANT CREATE TABLE, CREATE SEQUENCE To deepakise;

grant all privileges to deepakise identified by deepak123;
```

```
CREATE TABLE Employee1 (
EMPNO NUMBER(5),
ENAME VARCHAR2(50),
JOB VARCHAR2(50),
MANAGER_NO NUMBER(5),
SAL NUMBER(10, 2),
COMMISSION NUMBER(10, 2)
);
commit;

Disconnect;
```

```
SQL*Plus: Release 10.2.0.3.0 - Production on Wed May 29 09:44:23 2024

Copyright (c) 1982, 2006, Oracle.  All Rights Reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> Connect sys as sysdba;
Enter password: ******
Connected.


SQL> create user deepakise identified by deepak123;

User created.

SQL> GRANT CONNECT, RESOURCE TO deepakise;

Grant succeeded.

SQL> GRANT SELECT, INSERT, UPDATE, DELETE ON Employee To deepakise;

Grant succeeded.

SQL> GRANT CREATE TABLE, CREATE SEQUENCE To deepakise;

Grant succeeded.

SQL> grant all privileges to deepakise identified by deepak123;

Grant succeeded.

SQL> |
```



```
SQL> CREATE TABLE Employee1 (
  2       EMPNO NUMBER(5),
  3       ENAME VARCHAR2(50),
  4     JOB VARCHAR2(50),
  5    MANAGER_NO NUMBER(5),
  6    SAL NUMBER(10, 2),
  7    COMMISSION NUMBER(10, 2)
  8    );

Table created.
```

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.
      Delete the employee whose Empno is 105.

1. Creating the Employee table

```
CREATE TABLE Employee2 (
EMPNO INT PRIMARY KEY,
ENAME VARCHAR(50),
JOB VARCHAR(50),
MGR INT,
SAL DECIMAL(10, 2)
);
```

2. Adding a column commission to the Employee table

```
ALTER TABLE Employee2
ADD commission DECIMAL (10, 2);
```

3. Inserting five records into the table

```
INSERT INTO Employee3 (EMPNO, ENAME, JOB, MGR, SAL, commission)
VALUES (101, 'John Doe', 'Manager', NULL, 50000.00, 1000.00);
INSERT INTO Employee3 (EMPNO, ENAME, JOB, MGR, SAL, commission)
VALUES (102, 'Jane Smith', 'Developer', 101, 40000.00, 800.00);
INSERT INTO Employee3 (EMPNO, ENAME, JOB, MGR, SAL, commission)
VALUES (103, 'Mike Johnson', 'Analyst', 101, 35000.00, 700.00);
INSERT INTO Employee3 (EMPNO, ENAME, JOB, MGR, SAL, commission)
VALUES (104, 'Emily Brown', 'Designer', 102, 38000.00, 750.00);
INSERT INTO Employee3 (EMPNO, ENAME, JOB, MGR, SAL, commission)
VALUES (105, 'David Lee', 'Tester', 103, 32000.00, 600.00);
```

4. Updating the column details of job.

```
-- For example, changing 'Manager' to 'Project Manager'
UPDATE Employee2
SET JOB = 'Project Manager'
WHERE JOB = 'Manager';
```

5. Renaming the column of Employee table using ALTER command

```
ALTER TABLE Employee2
RENAME   COLUMN MGR TO MANAGER_ID;
```

6. Deleting the employee whose Empno is 105

```
DELETE FROM Employee2
WHERE EMPNO = 105;
```

Result:

```
SQL> CREATE TABLE Employee2 (
  2      EMPNO INT PRIMARY KEY,
  3      ENAME VARCHAR(50),
  4      JOB VARCHAR(50),
  5      MGR INT,
  6      SAL DECIMAL(10, 2)
  7  );

Table created.

SQL> desc EMPLOYEE2;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(38)
 ENAME                                              VARCHAR2(50)
 JOB                                                VARCHAR2(50)
 MGR                                                NUMBER(38)
 SAL                                                NUMBER(10,2)
```

```
SQL> ALTER TABLE Employee2
  2  ADD commission DECIMAL(10, 2);

Table altered.

SQL> INSERT INTO Employee2 (EMPNO, ENAME, JOB, MGR, SAL, commission)
  2  VALUES (101, 'John Doe', 'Manager', NULL, 50000.00, 1000.00);

1 row created.

SQL> INSERT INTO Employee2 (EMPNO, ENAME, JOB, MGR, SAL, commission)
  2  VALUES (102, 'Jane Smith', 'Developer', 101, 40000.00, 800.00);

1 row created.

SQL> INSERT INTO Employee2 (EMPNO, ENAME, JOB, MGR, SAL, commission)
  2  VALUES (103, 'Mike Johnson', 'Analyst', 101, 35000.00, 700.00);

1 row created.

SQL> INSERT INTO Employee2 (EMPNO, ENAME, JOB, MGR, SAL, commission)
  2  VALUES (105, 'David Lee', 'Tester', 103, 32000.00, 600.00);

1 row created.

SQL> SELECT * FROM EMPLOYEE2;
```

```
SQL> select * from Employee2;

     EMPNO ENAME                                          JOB                                                 MGR        SAL COMMISSION
---------- ---------------------------------------------- ------------------------------------
       101 John Doe                                       Manager                                                      50000       1000
       102 Jane Smith                                     Developer                                           101      40000        800
       103 Mike Johnson                                   Analyst                                             101      35000        700
       105 David Lee                                      Tester                                              103      32000        600

SQL> UPDATE Employee2
  2  SET JOB = 'Project Manager'
  3  WHERE JOB = 'Manager';

1 row updated.

SQL> select * from employee2;

     EMPNO ENAME                                          JOB                                                 MGR        SAL COMMISSION
---------- ---------------------------------------------- ------------------------------------
       101 John Doe                                       Project Manager                                              50000       1000
       102 Jane Smith                                     Developer                                           101      40000        800
       103 Mike Johnson                                   Analyst                                             101      35000        700
       105 David Lee                                      Tester                                              103      32000        600

SQL> ALTER TABLE Employee2
  2  RENAME   COLUMN MGR TO MANAGER_ID;

Table altered.

SQL> select * from Employee2;

     EMPNO ENAME                                          JOB                                           MANAGER_ID        SAL COMMISSION
---------- ---------------------------------------------- ------------------------------------
       101 John Doe                                       Project Manager                                              50000       1000
       102 Jane Smith                                     Developer                                           101      40000        800
       103 Mike Johnson                                   Analyst                                             101      35000        700
       105 David Lee                                      Tester                                              103      32000        600

SQL> DELETE FROM Employee2
  2  WHERE EMPNO = 105;

1 row deleted.
```

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 employeetable
3. Find the Maximum age from employee table.
4. Find the Minimum age from employeetable.
5. Find salaries of employee in Ascending Order.

Find grouped salaries of employees.

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

```
CREATE TABLE Employee3 (
 E_id INT PRIMARY KEY,
 E_name VARCHAR(20),
Age INT,
Salary DECIMAL(10, 2)
  );
INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
VALUES (101, 'John Doe', 30, 50000.00);
INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
VALUES (102, 'Jane Smith', 32, 40000.00);
INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
VALUES  (103, 'Mike Johnson', 33, 35000.00);
INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
VALUES (104, 'Emily Brown', 32, 38000.00);
```

2. Count number of employee names from employee table

```
SELECT COUNT(E_name) AS TotalEmployees FROM Employee3;
```

3. Find the Maximum age from employee table.

```
SELECT MAX(Age) AS MaxAge FROM Employee3;
```

4. Find the Minimum age from employee table

```
SELECT MIN(Age) AS MinAge FROM Employee3;
```

5. Find salaries of employee in Ascending Order.

SELECT Salary FROM Employee ORDER BY Salary ASC;

6. Find grouped salaries of employees.

SELECT Salary, COUNT(*) AS NumEmployees FROM Employee3 GROUP BY Salary;

SELECT E_name, Age, Age + 5 AS AgeAfter5Years FROM Employee30;

SELECT E_name, Salary, Salary * 0.1 AS SalaryIncrease FROM Employee3;

SELECT AVG(Salary) AS AverageSalary FROM Employee3;

Result:

```
SQL> CREATE TABLE Employee3 (
  2      E_id INT PRIMARY KEY,
  3      E_name VARCHAR(20),
  4      Age INT,
  5      Salary DECIMAL(10, 2)
  6  );

Table created.

SQL> desc employee3;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 E_ID                                      NOT NULL NUMBER(38)
 E_NAME                                             VARCHAR2(20)
 AGE                                                NUMBER(38)
 SALARY                                             NUMBER(10,2)

SQL> INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
  2  VALUES (101, 'John Doe', 30, 50000.00);

1 row created.

SQL> INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
  2  VALUES (102, 'Jane Smith', 32, 40000.00);

1 row created.

SQL>  INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
  2  VALUES  (103, 'Mike Johnson', 33, 35000.00);

1 row created.

SQL>    INSERT INTO Employee3 (E_ID, E_NAME, Age, Salary )
  2  VALUES (104, 'Emily Brown', 32, 38000.00);

1 row created.
```

```
SQL> select * from Employee3;

      E_ID E_NAME                      AGE     SALARY
---------- -------------------- ---------- ----------
       101 John Doe                     30      50000
       102 Jane Smith                   32      40000
       103 Mike Johnson                 33      35000
       104 Emily Brown                  32      38000

SQL> SELECT COUNT(E_name) AS TotalEmployees FROM Employee3;

TOTALEMPLOYEES
--------------
             4

SQL> SELECT MAX(Age) AS MaxAge FROM Employee3;

    MAXAGE
----------
        33

SQL> SELECT MIN(Age) AS MinAge FROM Employee3;

    MINAGE
----------
        30

SQL> SELECT Salary FROM Employee3 ORDER BY Salary ASC;

    SALARY
----------
     35000
     38000
     40000
     50000


SQL> SELECT COUNT(E_name) AS TotalEmployees FROM Employee3;

TOTALEMPLOYEES
--------------
             4

SQL> SELECT MAX(Age) AS MaxAge FROM Employee3;

    MAXAGE
----------
        33

SQL> SELECT MIN(Age) AS MinAge FROM Employee3;

    MINAGE
----------
        30

SQL> SELECT Salary FROM Employee3 ORDER BY Salary ASC;

    SALARY
----------
     35000
     38000
     40000
     50000

SQL> SELECT Salary, COUNT(*) AS NumEmployees FROM Employee3 GROUP BY Salary;

    SALARY NUMEMPLOYEES
---------- ------------
     38000            1
     50000            1
     40000            1
     35000            1
```

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)

```
SET SERVEROUTPUT ON;

CREATE TABLE CUSTOMER (
 ID INT PRIMARY KEY,
 NAME VARCHAR(10),
 AGE INT,
 ADDRESS VARCHAR(10),
 SALARY number(10, 2)
);

INSERT INTO CUSTOMER values(1,’Ramesh’,25,’Mysore’,200000);
INSERT INTO CUSTOMER values(2,’Komal’,35,’Bangalore’,800000);
INSERT INTO CUSTOMER values(3,’Mala’,45,’Mangalore’,56000);

CREATE OR REPLACE TRIGGER sal_difference_trigger
BEFORE INSERT OR UPDATE OR DELETE ON CUSTOMER
FOR EACH ROW
DECLARE
old_salary NUMBER;
new_salary NUMBER;
BEGIN
  IF INSERTING OR UPDATING THEN
old_salary := NVL(:OLD.SALARY, 0);
new_salary := NVL(:NEW.SALARY, 0);
    DBMS_OUTPUT.PUT_LINE('Salary difference: ' || (new_salary - old_salary));
  ELSIF DELETING THEN
old_salary := NVL(:OLD.SALARY, 0);
    DBMS_OUTPUT.PUT_LINE('Salary before deletion: ' || old_salary);
  END IF;
END;
```

RESULT:

```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE TABLE CUSTOMER (
  2      ID INT PRIMARY KEY,
  3      NAME VARCHAR(10),
  4      AGE INT,
  5      ADDRESS VARCHAR(10),
  6      SALARY number(10, 2)
  7  );

Table created.

SQL> INSERT INTO CUSTOMER values(1,'Ramesh',25,'Mysore',200000);

1 row created.

SQL> INSERT INTO CUSTOMER values(2,'Komal',35,'Bangalore',800000);

1 row created.

SQL> INSERT INTO CUSTOMER values(3,'Mala',45,'Mangalore',56000);

1 row created.

SQL>
SQL> select * from customer;

        ID NAME             AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore        200000
         2 Komal              35 Bangalore     800000
         3 Mala               45 Mangalore      56000
```

```
SQL> CREATE OR REPLACE TRIGGER sal_difference_trigger
  2  BEFORE INSERT OR UPDATE OR DELETE ON CUSTOMER
  3  FOR EACH ROW
  4  DECLARE
  5  old_salary NUMBER;
  6  new_salary NUMBER;
  7  BEGIN
  8      IF INSERTING OR UPDATING THEN
  9  old_salary := NVL(:OLD.SALARY, 0);
 10  new_salary := NVL(:NEW.SALARY, 0);
 11          DBMS_OUTPUT.PUT_LINE('Salary difference: ' || (new_salary - old_salary));
 12      ELSIF DELETING THEN
 13  old_salary := NVL(:OLD.SALARY, 0);
 14          DBMS_OUTPUT.PUT_LINE('Salary before deletion: ' || old_salary);
 15      END IF;
 16  END;
 17  /

Trigger created.

SQL> select * from customer;

        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      800000
         3 Mala               45 Mangalore       56000

SQL> INSERT INTO CUSTOMER values(6,'Jamal',30,'Mumbai',70000);
Salary difference: 70000

1 row created.

SQL> select * from customer;

        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      800000
         3 Mala               45 Mangalore       56000
         6 Jamal              30 Mumbai          70000

SQL> UPDATE customer SET salary = salary + 5000   WHERE id = 2;
Salary difference: 5000

SQL> select * from customer;

        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 Ramesh             25 Mysore         200000
         2 Komal              35 Bangalore      805000
         3 Mala               45 Mangalore       56000
         6 Jamal              30 Mumbai          70000

SQL> delete from customer where id=2;
Salary before deletion: 805000

1 row deleted.
```

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 TABLE EMPLOYEE5
(
E_ID INT PRIMARY KEY,
E_NAME VARCHAR (15),
AGE INT,
SALARY DECIMAL (10, 2)
);


INSERT INTO EMPLOYEE5 VALUES (1, 'Ramesh', 32, 2000.00);
INSERT INTO EMPLOYEE5 VALUES (2, 'Khilan', 25,1500.00);
INSERT INTO EMPLOYEE5 VALUES (3, 'Kaushik', 23,2000.00);
INSERT INTO EMPLOYEE5 VALUES (4, 'Chaitali', 25,6500.00);

DECLARE
E_id Employee5.E_id%TYPE;
E_name Employee5.E_name%TYPE;
Age Employee5.Age%TYPE;
Salary Employee5.Salary%TYPE;

-- Declare cursor
CURSOR employee_cursor
SELECT E_id, E_name, Age, Salary
FROM Employee5;

-- Open the cursor
BEGIN
OPEN employee_cursor;
-- Fetch data from cursor
LOOP
FETCH employee_cursor INTO E_id, E_name, Age, Salary;
EXIT WHEN employee_cursor%NOTFOUND;
```

```
 -- Output or use the fetched values
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || E_id || ', Name: ' || E_name || ', Age: ' || Age || ',
Salary: ' || Salary);
END LOOP;

CLOSE employee_cursor;
END;
```

Result:

```
SQL> CREATE TABLE EMPLOYEE5
  2   (
  3  E_ID INT PRIMARY KEY,
  4  E_NAME VARCHAR (15),
  5   AGE INT,
  6   SALARY DECIMAL (10, 2)
  7  );

Table created.

SQL>
SQL> INSERT INTO EMPLOYEE5 VALUES (1, 'Ramesh', 32, 2000.00);

1 row created.

SQL> INSERT INTO EMPLOYEE5 VALUES (2, 'Khilan', 25,1500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE5VALUES (3, 'Kaushik', 23,2000.00);
INSERT INTO EMPLOYEE5VALUES (3, 'Kaushik', 23,2000.00)
                            *
ERROR at line 1:
ORA-00928: missing SELECT keyword


SQL> INSERT INTO EMPLOYEE5 VALUES (4, 'Chaitali', 25,6500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE5 VALUES (3, 'Kaushik', 23,2000.00);

1 row created.
```

```
SQL> select * from employee5;

      E_ID E_NAME                    AGE     SALARY
---------- --------------- ---------- ----------
         1 Ramesh                   32       2000
         2 Khilan                   25       1500
         4 Chaitali                 25       6500
         3 Kaushik                  23       2000
```

```
SQL> DECLARE
  2  E_id Employee5.E_id%TYPE;
  3  E_name Employee5.E_name%TYPE;
  4      Age Employee5.Age%TYPE;
  5      Salary Employee5.Salary%TYPE;
  6
  7  -- Declare cursor
  8  CURSOR employee_cursor IS
  9      SELECT E_id, E_name, Age, Salary
 10      FROM Employee5;
 11
 12  -- Open the cursor
 13  BEGIN
 14      OPEN employee_cursor;
 15
 16      -- Fetch data from cursor
 17      LOOP
 18          FETCH employee_cursor INTO E_id, E_name, Age, Salary;
 19          EXIT WHEN employee_cursor%NOTFOUND;
 20
 21          -- Output or use the fetched values
 22          DBMS_OUTPUT.PUT_LINE('Employee ID: ' || E_id || ', Name: ' || E_name || ', Age: ' || Ag
e || ', Salary: ' || Salary);
 23      END LOOP;
 24
 25      -- Close the cursor
 26      CLOSE employee_cursor;
 27  END;
 28  /
Employee ID: 1, Name: Ramesh, Age: 32, Salary: 2000
Employee ID: 2, Name: Khilan, Age: 25, Salary: 1500
Employee ID: 4, Name: Chaitali, Age: 25, Salary: 6500
Employee ID: 3, Name: Kaushik, Age: 23, Salary: 2000

PL/SQL procedure successfully completed.
```

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 table O_RollCall (roll int,name varchar(10));
create table N_RollCall (roll int,name varchar(10));

insert into O_RollCall values(1,'bc');
insert into O_RollCall values(3,'bcd');
insert into O_RollCall values(4,'d');
insert into O_RollCall values(5,'bch');

insert into N_RollCall values(1,'bc');
insert into N_RollCall values(2,'b');
insert into N_RollCall values(5,'bch');

DECLARE
v_count NUMBER;
   CURSOR c_new_rollcall IS
      SELECT roll, name
      FROM N_RollCall;
BEGIN
   FOR new_rec IN c_new_rollcall LOOP
      -- Check if the record already exists in O_RollCall
      SELECT COUNT(*)
      INTO   v_count
      FROM   O_RollCall
WHERE  roll = new_rec. roll;

      -- If record doesn't exist, insert it
      IF v_count = 0 THEN
         INSERT INTO O_RollCall (roll, name)
         VALUES (new_rec. roll, new_rec.name);
         DBMS_OUTPUT.PUT_LINE('Record inserted: ' || new_rec. roll);
```

```
      ELSE
         DBMS_OUTPUT.PUT_LINE('Record skipped: ' || new_rec. roll);
      END IF;
   END LOOP;
   COMMIT;
END;
select * from N_RollCall;
select * from O_RollCall;
```

RESULT:

```
SQL>  select * from O_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         3 bcd
         4 d
         5 bch

SQL> select * from N_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         2 b
         5 bch
```

```
SQL> DECLARE
  2  v_count NUMBER;
  3      CURSOR c_new_rollcall IS
  4          SELECT roll, name
  5          FROM N_RollCall;
  6  BEGIN
  7      FOR new_rec IN c_new_rollcall LOOP
  8          -- Check if the record already exists in O_RollCall
  9          SELECT COUNT(*)
 10          INTO   v_count
 11          FROM   O_RollCall
 12  WHERE  roll = new_rec. roll;
 13
 14          -- If record doesn't exist, insert it
 15          IF v_count = 0 THEN
 16              INSERT INTO O_RollCall (roll, name)
 17              VALUES (new_rec. roll, new_rec.name);
 18              DBMS_OUTPUT.PUT_LINE('Record inserted: ' || new_rec. roll);
 19          ELSE
 20              DBMS_OUTPUT.PUT_LINE('Record skipped: ' || new_rec. roll);
 21          END IF;
 22      END LOOP;
 23      COMMIT;
 24  END;
 25  /
Record skipped: 1
Record inserted: 2
Record skipped: 5

PL/SQL procedure successfully completed.
```

```
SQL> select * from N_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         2 b
         5 bch

SQL> select * from O_RollCall;

      ROLL NAME
---------- ----------
         1 bc
         3 bcd
         4 d
         5 bch
         2 b
```

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

https://www.mongodb.com/try/download/community

https://www.mongodb.com/try/download/shell

Set Environmental Path

Mongod –version
Mongosh
Use dbname
db.createCollection("collectionname")
db.sana.insert({ "name": "Alice", "age": 30 })

db.sana.find()
db.sana.update({ "name": "Alice" }, { $set: { "age": 31 } })