Task-5:- Writing Join queries; Equivalent; AND/OR
Recursive Queries

Aim:- To implement and execute Join queries equivident query and recursive queries using a university database scenario

Procedure:-

The SQL Joins Calues used to records two or more tables in database means of the actually Performed by the 'where' Calues which Contains rows and of tables

* Create the database and tables
* insert Sample data
* implement a recursive query
* display results and verify Corrections

Step:1 :-

Syntax:-

1. Types of Joins
   1. Simple Join
   2. Self Join
   3 Outer Join

Simple Join:-

It is most Common type of Join it retrives the row from 2 tables

Equi-Join:-

A Join, which is based on evuliation Called equi-Join

Example:-

Select * from item cost where item id=cost id.

1. Db Queries:-

```
CREATE TABLE (

  CREATE TABLE department (
      dep Id. PRIMARY KEY,
      deptname VARCHAR (50)
      );
  CREATE TABLE students (
      student ID INT Primary key;
      student Name VARCHAR(50);
        Dept INT;
    FORGEIN key (deptId) REFERENCES
  departments (deptID);

CREATETABLE courses (
    Course ID VARCHAR(10) PRIMARY KEY;
    Course name VARCHAR (50)
  dept INT;
  FORGEIN KEY (student ID) REFERENCES
          students (student ID);
FORGEIN KEY (CourseID) Reference Courses
          (course CID));

  CREATE TABLE Prerequisites (
      CourseID VARCHAR(10);
      Pre reqID VARCHAR(10)
      );
```

2) INSERT SAMPLE DATA :-

```
INSERT INTO department VALUES
(101, Computer Science);
(102, Electrical engg);
(103, Mechanical Engg);
INSERT INTO students VALUES;
(1, 'Alice' 101);
(2 'Bob' 102);
(3 'Charlie' 103);
(4 'David' 104);
(5 'Emma' 104);
INSERT INTO Course VALUES
('C1, database systems, 101);
('C2, operating systems, 101);
('C3, 'circuits' 102);
('C4, Thermodynamics, 103);
INSERT INTO Prerequisites values
('C2', 'C1') - OS requires DB
('C3', 'C5') - Circuits requires OS.
```

3. JOIN QUERIES (ALL TYPES)

a) INNER JOIN

```
SELECT s.student Name ; d.Dept Nam
FROM students.
LEFT JOIN departments don dept ID = ddept Id
```

(b) LEFT JOIN:-

```
SELECT s. student Name . ddept Name
FROM students.
LEFT JOIN departments don dept ID = ddept ID
```

(c) RIGHT JOIN

```
SELECT s student Name ddept Name
FROM student s
RIGHT JOIN departments don sdept ID = ddept ID
```

(d) FULL OUTER JOIN

```
SELETs.students Name, Course name
FROM students
FULL OUTER JOIN departments don dept ID = dept Id
```

(e) CROSS JOIN

```
SELECT s. student name . Course Name
FROM students.
CROSS JOIN courses;
```

(F) SELF JOIN

```
SELECT S1. Student name As Student 1
       S2 student name As Student 2 S1. dept ID
FROM students S1
JOIN students S2 ON dept ID = S2 dept ID
WHERE S1 . student ID < S2 . student ID
```

(4) EQUIVALENT QUERIES:-

Using JOIN

```
SELECT S.StudentName, d.deptName
FROM Students s
JOIN departments d ON deptID = d.deptID
```

- using subquery

```
SELECT StudentName,
    (SELECT deptName FROM department d
                    WHERE d.deptID
    = s.deptID) AS deptName
FROM students s
```

(5) RECURSIVE QUERY:-

```
WITH RECURSIVE CourseHierarchy AS (
SELECT Prerequisites
 UNION
SELECT P.CourseID, C.PrereqID
FROM Prerequisites P
FROM
JOIN CourseHierarchy CON PPrereqID = e.CourseID
)
SELECT * FROM CourseHierarchy
```

Result:- The implementation of SQL Commands using Joins and recursive queries are executed successfully

8/9/24

VEL TECH - CSE	
EX NO.	05
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	4
RECORD (5)	-
TOTAL (20)	14
SIGN WITH DATE	