Task :- Writing Join queries, Equalent, AND/OR Recursive Queries

Aim :- To Implement and encute JOIN queries equivalent quires and recursive quires

procedure :-

* Create the database and tables
* Insert sample dates.
* write SQL quires using differenti'ent types of joins.
* write equivalent quires
* Implement a recursive query (using) with Requrisve.
* display results and writy condenses.

different types of SQL JOINS

* (Inner Join):
  tables select column-name (s) FROM table INNER JOIN table 2 ON table1. Colum-name
* Left counter Join:
  SELECT column-name (s) FROM table 1 LEFT JOIN table 2 ON table 1. column -name = table 2 column-name
* Right (counter) JOIN:
  SELECT colum-number (s) FROM table 1 Right JOIN table 2 ON table . column-name = table 2. colum-name;

* FULL (  ) JOIN ;-

FULL OUTER JOIN table 2 ON table 1.
column_name = table 2. column_name;

1. JOIN Queire (All types)

CREATE TABLE Department (

DEPT ID INT PRIMARY KEY,
DEPTNAME VARCHAR(5)

));

CREATE TABLE patients (
patID INT PRIMARY KEY
pat Name VARCHAR(5);
DeptID INT,
FORENGN KEY (Dept ID) REFERENCES Department (DptID)

);

create TABLE Prerequiser (

CourseID VARCHAR (10);
Prerequiser VARCHAR (10);

));

2) INSERT SAMPLE DATA ;-

INSERT. INTO department VALUES

(101; computer science);
(102; Electrical Engg);
(103; Mechanical Engg);

INSERT INTO student VALUES;

(1. Alice, 101);
(2. 'Bob' 102);
(3. 'charlie' 103);
(4. david' 104);

(5, 'Emma', 104);

INSERT INTO Courses VALUES;

("C1, database systems; 101);

("C2, operating systems; 101);

("C3; circuits " 102);

("C4; Thermodynamics; 103);

INSERT INTO prerequisits values

("C2, C1") - os requires DB

("C3, C2") - circuits require os.

3) JOIN queries (All types)

a) Inner JOIN

select - sstudent Name; dept Nam

FROM students

left JOIN departments don dept ID = dept ID

b) left JOIN :

SELECT s. students Name. ddept name

FROM students.

LEFT JOIN departments ON dept ID = ddept ID

c) RIGHT JOIN

SELECT s student Name dept name

FROM students

RIGHT JOIN departments don sdept ID = ddept ID

d) FULL OUTER JOIN

```
SELECT s.student_name, c.course_name FROM
students
FULL OUTER JOIN departments dept_ID
= dept_ID
```

e) CROSS JOIN

```
SELECT s.student_name, c.course_name
FROM students
CROSS courses c;
```

f) SELF JOIN

```
SELECT s1.student_name AS student
       s2.student_name AS student2
FROM students s1
JOIN students s2 ON dept_ID = s2.dept_ID
WHERE s1.student_ID < s2.student_ID.
```

f) EQUIVALENT QUERIES :-

using JOIN

```
SELECT s.student_name, d.dept_name
FROM students s.
JOIN departments d ON dept_ID = d.dept_ID
```

- using subquery

```
SELECT student Name;
(SELECT dept Name FROM department
    = student ID) As dept Name
FROM students.s
```

5) RECURSIVE QUERY :-

```
WITH . RECURSIVE Course Hierarchy AS(
SELECT Prerequites
UNION
SELECT P.course ID, C.Prepare ID
FROM Prerequinstes P.
FROM
JOIN Course Hearphy (JOIN prepreq ID
                    = (course ID
)
SELECT * FROM Course Hierachy;
```

8/9/25

Result :- The Implementation of SQL commands using JOINS and recursive queries are executed sucessfully.

| VEL TECH - CSE | |
|---|---|
| EX NO. | 05 |
| PERFORMANCE (5) | 5 |
| RESULT AND ANALYSIS (5) | 5 |
| VIVA VOCE (5) | 4 |
| RECORD (5) | - |
| TOTAL (20) | 15 |
| SIGN WITH DATE | |