Task No: 5 Date: 09/09/25	Writing Join Queries, Equivalent, AND/OR Recursive Quries.

Aim: To implement and execute Join queries, equivalent queries, and recursive queries.

Types of Joins in SQL :-

1. Inner Join:- Returns records that have matching values in both tables.

Syntax:- Select column_name(s) From table1 INNER JOIN table2 ON table1.column_name = table2.column_name;

2. Left Outer Join:- Returns all records from the left table, and the matched records from the right table.

Syntax: Select column_name(s) From table1 LEFT JOIN table2 ON table1.column_name = table2.column_name;

3. Right outer Join:- Return all records from the right table, and the matched records from the left table.

Syntax:- Select column_name(s) From table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name.

4. Full outer Join: Returns all records when there is a match in either left or right table. ~~select~~ ~~column~~

Syntax: Select column_name(s) From table1 Full outer Join table2 ON table1.column_name = table2.column_name;

1. JOIN QUERIES

Create Tables

```
Create table customer(
    CustomerID int primary key,
    name Varchar(50),
    address Varchar(100), referred By ID INT NULL,
  );  Foreign key (referredByID) Rerferences customer(customerID).

create Table bank_account(
    account_number int Primary key;
    ~~CustomerID int~~ CustomerID int,
    balance int,
    category Varchar(50),
    Foreign key (customerID) references customer (customerID)
  );

Create table branch(
    branchID int primary key,
    branchName varchar(50),
);
```

2. Insert Sample data

```
insert into customer(customerID, name, address) values
    (101, 'Ram kumar', 'chennai');
insert into customer (customerID, name, address) values
    (102, 'vijay Rao', 'Hyderabad');
insert into Customer (customerID, name, address) values
    (103, 'Vasu Reddy', 'vizag');
insert into customer (customerID, name, address) values
    (104, 'vinay kumar', 'chennai');
insert into customer(customerID, name, address) values
    (105, 'Rohit', 'Delhi');
insert into bank_account (account-number, customerID,
balance, category) values (1001, 101, 15000, 'savings');
insert into bank_account (account_number, customerID,
balance, category) values (1002, 102, 0, 'current');
```

insert into bank_account (account_number, customerID, balance, category) values (1003, 103, 5000, 'savings');

insert into bank_account (account_number, customerID, balance, Category) values (1004, 105, 2000, 'current');

insert into branch (branchID, branchName) values (1, 'chennai Branch');

insert into branch (branchID, branchName) values (2, 'Hyderabad Branch');

insert into branch (branchID, branchName) values (3, 'vizag Branch');

3. Join Queries.

(a) <u>Inner Join</u>:-

Query:- Select c.name, b.account_number from customer C inner Join bank_account b ON c.customerID=b.customerID;

Output:-

name	account_number
Ram kumar	1001
vijay Rao	1002
vasu Reddy	1003
vinay kumar	1004.

(b) Left Join:-

Query:- Select c.name, b.account_number from Customer c

left join bank_account b ON c.customerID = b.customerID;

output:-

name	account_number
Ramkumar	1001
vijay Rao	1002
Vasu Reddy	1003
Vinay kumar	1004
Rohit Sharma	NULL

(c) Right Join:-

Query:- Select c.name, b.account_number from Customer c

Right Join bank_account b ON c.customerID = b.customerID;

Output:-

name	account_number
Ramkumar	1001
vijay Rao	1002
vasu Reddy	1003
vinay kumar	1004

(d) Full outer Join:-

Query:- Select c.name, b.account_number from Customer c

Full outer join bank_account b ON c.customerID = b.customerID;

name	account_number
Ram kumar	1001
Vijay Rao	1002
vasu Reddy	1003
vinay kumar	1004
Rohit sharma	NULL.

Equivalent Query

(a) using Join

Query:- Select c.name AS CustomerName, b.account_number AS Accountnumber From Customer c Join bank_account b on c.customerID = b.customerID;

output :-

customerName	AccountNumber
Ram kumar	1001
Vijay RaO	1002
Vasu Reddy	1003
Vinay kumar	1004

(b) using subquery

Query :- Select c.name AS CustomerName, (select b.account_numb
From bank_account b where b.customerID = c.customer ID
Limit 1) AS AccountNumber From customer c;

output :-

customer Name	AccountNumber
Ram kumar	1001
Vijay Rao	1002
Vasu Reddy	1003
Vinay kumar	1004
Rohit sharma	~~NU~~ NULL

5. Recursive Query :-

Query :- with Recursive ReferralHierarchy AS (select
customerID, referredByID From customer where
referredByID is NOT NULL UNION .
Select c.customerID, c.referredByID From customer c
Join Referral Hierarchy rh on c.referred ByID = rh.CustomerID
) select * from ReferralHierarchy;

output :-

CustomerID	referredByID
102	101
103	102
104	103

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	-
TOTAL (20)	14
SIGN WITH DATE	[illegible] 9/9/25

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