Date 29/9/25.

Task-No 35 Writing Join Queries, Equivalent, AND/OR Recursive Queries.

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 table 1 INNER JOIN table2 ON table 1_ column_name = table 2. 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 table 1 (LEFT JOIN table 2 on table 1. column_name = table 2. 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 table 1 RIGHT JOIN table 2 on table 1. Column-name = table 2. column_name.

4. Full outer join: Returns all records when there is a match in either left or right table.

syntax: select column_name (5) from table 1 Full outer join table 2 ON table 1. Column_name = table 2. Column-name,

1. Join Queries

Create Tables:

```
create table customer (.
  customer ID int primary key,
  name varchar (50),
  address varchar (100) reference by ID INT NULL,
  Foreign Key (reference ID) Reference customer (customer ID).
);
```

```
Create table bank-account(
    account-number int primary key;
    customer ID int;
    balance int,
    category Varchar (50),
    foreign key (customer ID) reference customer (customer ID)
);

Create table branch (
    branch ID int primary key,
    branch name Varchar (50),
);
```

2. Insert Sample data

```
insert into customer (customer ID, name, address) values
(101, 'Ram Kumar', 'chennai');
insert into customer (customer ID, name, address) values
(102, 'Vijay Rao', 'Hyderabad');
insert into customer (customer ID, name, address) values.
(103, 'Vasu Reddy', 'Vizag');
insert into customer (customer ID, name, address) values
(104, 'Vinay Kumar', 'chennai');
insert into customer (customer ID, name, address) values
(105, 'Rohit', 'Delhi');
insert into bank-account (account-number customer ID,
balance, category) values (1001, 101, 15000, 'saving');
insert into bank-account (account-number, customer ID,
balance, category) values (1002, 102, 0, 'current');
insert into bank-account (account-number, customer ID,
balance, category) values (1003, 103, 5000, 'savings');
insert into bank-account (account-number, customer ID,
balance, category) values (1004, 105, 2000, 'current');
```

insert into branch (branch ID, branchname) values
(1, 'chennai Branch');
insert into branch (branch ID, branch name) values
(2, 'Hyderabad Branch');
insert into branch (branch ID, branch name) values.
(3, 'vizag Branch');

3. Join Queries:

a) Inner Join:

Query :- select c.name.b. account-number from customer c.
inner join bank-account b ON c. customer ID=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. customer ID = b. customer ID;

Output:

Name	account-number
Ramkumar	1001
vijay Rao.	1002
Vasu Reddy.	1003
Vinay Kumar	1004
Rohit sharma	1005.

c) Right Join:-

Query:- select c.name, b. account-number from customer c
Right join bank-account b ON c. customer ID = b. customerID;

output:

Name	account-number
Ram Kumar	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. customer ID=b.customerID,

Name	account-number
Ram Kumar	1001
Vijay Rao.	1002
Vasu Reddy	1003
Vinay Kumar	1004
Rohit sharma	1005

~~Insert into bank-account (account-number, customer ID, balance, category) values (1002, 102 0, 'current');~~
~~Insert into bank-account (account-number, customerID, balance, category) values (1003,~~

Equivalent Query.

a) Using Join.

Query:- select c. name As customer Name, b.account-number As Account number From customer c. Join bank-account b on c. customer ID = b. customer ID;

Output:

customer name	Account number
Ram Kumar	1001
Vijay Rao.	1002
Vasu Reddy.	1003
Vinay Kumar	1004.

b) Using sub Query.

Query: select c.name AS customer Name, (select b. account-number From bank-account b where B.customer ID = c.customer ID limit 1) AS Account number from customer c;

output :-

customer name	Account number.
Ram Kumar	1001
Vijay Rao.	1002
Vasu Reddy	1003
Vinay Kumar	1004
Rohit sharma	NULL.

5. Recursive Query:

Query: with Recursive Referral iterachy AS (select customer ID, reference By ID from customer where. By ID is NOT NULL UNION.

select c. customer ID, c. reference by ID from customer c. Join Referral Hierarchy on c. referred by ID = rh.customer ID) select * from Referral Hierarchy;

Output:-

customer ID	referred by ID.
102	101
103	102
104.	103.

VEL TECH

EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIG WITH DATE	[illegible] 15/9

<u>Result</u>: the implementation of SQL commands using Joins and recursive Queries are executed successfully.