Task 5: Writing Join Queries, Equivalent, AND/OR Recursive Queries.

09-25

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 table.

SYNTAX: Select column_name(s) From table1 INNER JOIN table2 ON table1.Column_name = table2.colu-mn_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.

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)
);
```

```
Create Table bank_account(
    account_number int primary key,
    CustomerID int,
    balance int,
    Category varchar(50)
    Foreign key (CustomerID) references customer(customerID)
);
Create table branch(
    branchID int primary key,
    branch Name 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, 'vijayRao', 'Hyderabad');
insert into customer(customerID, name, address) va-
-lues (103, 'vasu', 'vizag');
insert into Customer(customerID, name, address)
values(104, 'vinay', 'chennai');
insert into customer(customerID, name, address)
values (105, 'Rohit', 'Delhi');
insert into bank_account(account_number, customer
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');

Q. Join Queries

(a) Inner Join:-

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	1002
~~srinag~~ vasu	1003
vinay	1004

b) Left Join :-

Query:- Select C. name, b.account_number from Customer C left join bank_account b ON C. Customer ID = b.CustomerID;

Output :-

Name	Account_number
Ramkumar	1001
Vijay	1002
Valu	1003
Vinay	1004
Rohit	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
Ram kumar	1001
vijay	1002
Valu	1003
Vinay	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	1002
Valu	1003
Vinay	1004
Rohit	NULL

Equivalent Query

(a) Using Join

Query :- Select c.name As customerName, b. account-number As Account number From customer c Join bank_account b on c. customerID = b. customerID;

Output:

Customer Name	Account Number
Ramkumar	1001
Vijay	1002
vasu	1003
vinay	1004

(b) Using SubQuery

Query:- Using c.name As customerName, (select b.account-number From bank_account b where b.customerID = c. CustomerID limit 1 As Account Number From customer c;

output:

Customerrlame	AccountNumber
Ramkumar	1001
Vijay	1002
vasu	1003
vinay	1004
Rohith	NULL

5) Recursive Query :-

With Recursive Referral Hierarchy As (select CustomerID, referredByID From customer where refferedByID is NOT NULL UNION Select c. customerID, c. referredByID From customer c Join Referral hierarchy rh on c. referredByID = rh. customerID) select * from Referral Hierarchy;

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)	5
RECORD (5)	5
TOTAL (20)	15
SIGN WITH DATE	

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