Task NO-5    Writing Join Queries, Equivalent, AND/OR Recursive Queries.

Date :- 09-09-25

Aim :- To implement and execute join queries, equivalen 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 table 2 ON table 1 - column name = table 2. coloumn-Name;

2. left outer join :- Returns all records from the left table, and the matched records from the right table.

syntax :- select column-names(s)-from table1 LEFT JOIN table 2 on table 1. column-name = table2. column; Name;

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

syntax :- select column-names(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(s) from table1 full outer join table 2 ON table 1- column-Name = table 2, column-name;

1. Join Queries

Create tables :-

```
Create table customer(
  customerID int primary key,
   name varchar(50),
   address varchar(100)reference by ID INT NULL,
  foreign key (referenceID)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 (customerID, name, address)values.
  (101, 'Ram kumar', 'chennai');
insert into customer(customerID, name, address) values
  (102, 'vijay Rao', 'Hyderabad');
insert into customer (customer ID, name, address)value,
   (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, customerID,
balance, category) values (1001, 101, 15000, 'Savings');
```

```
insert into bank-account (account-number, customer ID,
balance, category) values (1002, 102, 0, 'curren');
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, branch Name) 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. customer_ID;

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 |
|---|---|
| Ram kumar | 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.customerID = 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.customerID = b.customerID;

| name | account-number |
|---|---|
| Ram kumar | 1001 |
| vijay Rao | 1002 |
| vasu Reddy | 1003 |
| vinay kumar | 1004 |
| Rohit sharma | 1005 |

Equivalent Query:

a) using Join

Query:- Select c.name As customer Name, b.account-Number As Accountnumber from customer c Join bank-account b on c.customerID=b.customerID;

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.customerID-c.customerID limit 1) As account number from customer;

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 customerID, Reference By ID from customer where ByID is NOT NULL UNION

Select c .customerID, c.reference byID from customer c
Join Refernal Hirerarchy on c.refered by ID=
rh.customer ID) Select * from Referal Hierachy;

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) | 4 |
| RECORD (5) | |
| TOTAL (20) | 14 |
| SIGN WITH DATE | [illegible] 9/7/25 |

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