Lab 9
Theory & Concept

Objective: - To implement the concept of Joins

Joint Multiple Table (Equi Join): Sometimes we require to treat more than one table as though manipulate data from all the tables as though the tables were not separate object but one single entity. To achieve this, we have to join tables. Tables are joined on column that have same data type and data with in tables.

The tables that must be joined are specified in the FROM clause and thejoining attributes in the WHERE clause.

Key Concepts of an Equi Join

1. **Tables to be Joined**: Multiple tables are specified in the **FROM** clause.
2. **Joining Condition**: The condition to join the tables is specified in the **WHERE** clause, where you define how the columns in the tables relate to each other. Specifically, you look for rows where the values of certain columns match (equally).

Example

Let's assume we have two tables:

- **employees**
  - **Columns:** emp_id, emp_name, dept_id
- **departments**
  - **Columns:** dept_id, dept_name

The dept_id in the employees table is a foreign key that references the dept_id in the departments table. We want to find out the names of employees along with the names of their respective departments.

Equi Join Syntax

```
SELECT e.emp_name, d.dept_name

FROM employees e, departments d

WHERE e.dept_id = d.dept_id;
```

Explanation:

1. **FROM clause**: Specifies the two tables (employees and departments) that you want to join.
2. **WHERE clause**: Specifies the join condition (e.dept_id = d.dept_id). This condition ensures that only rows where the dept_id from the employees table match with the dept_id in the departments table will be included in the result set.
3. **SELECT clause**: Specifies the columns to be displayed in the result set (emp_name and dept_name).

Example Data:

`employees` table:

emp_id	**emp_name**	**dept_id**
1	Alice	101
2	Bob	102
3	Charlie	103

`departments` table:

dept_id	**dept_name**
101	HR
102	Engineering
103	Marketing
104	Sales

Result of Equi Join:

emp_name	**dept_name**
Alice	HR
Bob	Engineering
Charlie	Marketing

As shown in the result, the employees are listed along with the corresponding department name where their `dept_id` matches.

Cartesian Product (Without Join Condition)

Before applying the join condition in the **WHERE** clause, SQL will first generate a **Cartesian Product** (also known as a cross product) of the two tables, which means every row in the first table is combined with every row in the second table. This results in a table with all possible combinations of rows, which is generally not useful until the **WHERE** condition is applied to filter the results.

For instance, if the employees table has **3 rows** and the departments table has **4 rows**, the Cartesian product will result in **3 * 4 = 12 rows**. However, the equi join ensures that only the rows where the **dept_id** matches between the two tables are returned.

Variants of Joins:

1. Inner Join (Equi Join):

An **inner join** is essentially an equi join where you only return the rows where the join condition is met. It's the default type of join used in most cases when you specify the **JOIN** keyword.

Example SQL Query:

SELECT e.emp_name, d.dept_name

FROM employees e

INNER JOIN departments d **ON** e.dept_id = d.dept_id;

Example Data:

`employees` table:

emp_id	emp_name	dept_id
1	Alice	101
2	Bob	102
3	Charlie	103
4	David	101

`departments` table:

dept_id	dept_name
101	HR
102	Engineering
103	Marketing

Result of Inner Join:

emp_name	dept_name
Alice	HR
Bob	Engineering
Charlie	Marketing
David	HR

2. Left Outer Join:

A **Left Outer Join** includes all rows from the left table **(`employees`)** and the matched rows from the right table **(`departments`).** If there is no match, the result will contain NULL values for columns from the right table.

Example SQL Query:

SELECT e.emp_name, d.dept_name

FROM employees e

LEFT JOIN departments d **ON** e.dept_id = d.dept_id;

Example Data:

`employees` table:

emp_id	emp_name	dept_id
1	Alice	101
2	Bob	102
3	Charlie	103

emp_id	emp_name	dept_id
4	David	NULL

`departments` table:

dept_id	dept_name
101	HR
102	Engineering
103	Marketing

Result of Left Outer Join:

emp_name	dept_name
Alice	HR
Bob	Engineering
Charlie	Marketing
David	NULL

3. Right Outer Join:

A **Right Outer Join** includes all rows from the right table (`departments`) and the matched rows from the left table **(`employees`).** If there is no match, NULL values are returned for columns from the left table.

Example SQL Query:

SELECT e.emp_name, d.dept_name

FROM employees e

RIGHT JOIN departments d **ON** e.dept_id = d.dept_id;

Example Data:

`employees` table:

emp_id	**emp_name**	**dept_id**
1	Alice	101
2	Bob	102
3	Charlie	103

`departments` table:

dept_id	**dept_name**
101	HR
102	Engineering
103	Marketing
104	Sales

Result of Right Outer Join:

emp_name	**dept_name**
Alice	HR
Bob	Engineering
Charlie	Marketing
NULL	Sales

4. Full Outer Join:

A **Full Outer Join** returns all rows when there is a match in either left or right table. It returns NULL for unmatched rows from both tables.

Example SQL Query:

SELECT e.emp_name, d.dept_name

FROM employees e

FULL OUTER JOIN departments d **ON** e.dept_id = d.dept_id;

Example Data:

`employees` table:

emp_id	**emp_name**	**dept_id**
1	Alice	101
2	Bob	102
3	Charlie	103
4	David	NULL

`departments` table:

dept_id	**dept_name**
101	HR
102	Engineering
103	Marketing
104	Sales

Result of Full Outer Join:

emp_name	**dept_name**
Alice	HR
Bob	Engineering
Charlie	Marketing
David	NULL
NULL	Sales

LAB TASK

OBJECTIVE: Answer the following Queries:

1. Find out which product has been sold to 'Ivan.'
2. Find out the product and their quantities that are in the process of being delivered.
3. Find the product_no and description of products that have 10 orders.
4. Find out the names of clients who have purchased 'CD DRIVE'.
5. List the product_no and s_order_no of clients having qty ordered less than 5 from the sales_order_details table for the product "1.44 floppies".
6. Find the products and their quantities for the orders placed by 'Vandan' and 'Ivan'.
7. Find the products and their quantities for the orders placed by client_no "0004" and "0005".
8. Display the s_order_date in the format "dd-mm-yy" e.g. "12-Feb-96".
9. Find the date, 15 days after the given date.