Experiment Number 4

AIM: To implement different types of Joins in SQL

In SQL, joins are used to combine rows from two or more tables based on a related column between them. The different types of joins are designed to retrieve data from multiple tables in a relational database, and each type of join specifies a different way of combining data. Here are the main types of SQL joins:
Before going to types of joins, consider two tables, employees and departments, with the following structures:

```
-- Creating Employees Table
CREATE TABLE Employees (
    employee_id NUMBER PRIMARY KEY,
    name VARCHAR2(50) NOT NULL,
    department_id NUMBER
);

-- Creating Departments Table
CREATE TABLE Departments (
    department_id NUMBER PRIMARY KEY,
    department_name VARCHAR2(50) NOT NULL
);

-- Inserting data into Employees table
INSERT INTO Employees VALUES (1, 'Alice', 101);
INSERT INTO Employees VALUES (2, 'Bob', 102);
INSERT INTO Employees VALUES (3, 'Charlie', 101);
INSERT INTO Employees VALUES (4, 'Adam', 104);


-- Inserting data into Departments table
INSERT INTO Departments VALUES (101, 'HR');
INSERT INTO Departments VALUES (102, 'IT');
INSERT INTO Departments VALUES (103, 'Finance');
```

employees:

employee_id	name	department_id
1	Alice	101
2	Bob	102
3	Charlie	101
4	Adam	104

departments:

department_id	department_name
101	HR
102	IT
103	Finance

1. INNER JOIN

The **INNER JOIN** keyword selects records that have matching values in both tables. It returns rows when there is a match in both tables.

Syntax:

```
SELECT columns FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

Example:

```
SELECT employees.name, departments.department_name FROM employees
INNER JOIN departments
ON employees.department_id = departments.department_id;
```

name	department_name
Alice	HR
Bob	IT
Charlie	HR

2. LEFT JOIN (or LEFT OUTER JOIN)

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side if there is no match.

Syntax:

```
SELECT columns FROM table1
LEFT JOIN table2
ON table1.common_column = table2.common_column;
```

Example:

```
SELECT employees.name, departments.department_name FROM employees
LEFT JOIN departments
ON employees.department_id = departments.department_id;
```

name	department_name
Alice	HR
Bob	IT
Charlie	HR
Adam	NULL

3. RIGHT JOIN (or RIGHT OUTER JOIN)

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side if there is no match.

Syntax:

SELECT columns FROM table1
RIGHT JOIN table2
ON table1.common_column = table2.common_column;
Example:
SELECT employees.name, departments.department_name FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.department_id;

name	department_name
Alice	HR
Charlie	HR
Bob	IT
NULL	Finance

4. FULL JOIN (or FULL OUTER JOIN)

The FULL JOIN keyword returns all records when there is a match in either left (table1) or right (table2) table records. It returns NULL for records with no match in the other table.

Syntax:

SELECT columns FROM table1
FULL JOIN table2
ON table1.common_column = table2.common_column;
Example:
SELECT employees.name, departments.department_name FROM employees
FULL JOIN departments
ON employees.department_id = departments.department_id;

name	department_name
Alice	HR
Charlie	HR
Bob	IT
Adam	NULL
NULL	Finance

5. CROSS JOIN

The CROSS JOIN keyword returns the Cartesian product of the two tables. It combines all rows from the first table with all rows from the second table.

Syntax:

SELECT columns FROM table1
CROSS JOIN table2;

Example:

SELECT employees.name, departments.department_name FROM employees
CROSS JOIN departments;

name	department_name
Alice	HR
Alice	IT
Alice	Finance
Bob	HR
Bob	IT
Bob	Finance
Charlie	HR
Charlie	IT
Charlie	Finance
Adam	HR
Adam	IT
Adam	Finance

6. SELF JOIN

A SELF JOIN is a regular join but the table is joined with itself.

Syntax:

```
SELECT a.columns, b.columns
FROM table a JOIN table b
ON a.column_id=b.column_id;
```

Example:

```
SELECT e1.name AS Employee1, e2.name AS Employee2 FROM employees e1
JOIN employees e2 ON e1.manager_id = e2.employee_id;
```

Table (employees) with manager_id:

employee_id	name	department_id	manager_id
1	Alice	101	3
2	Bob	102	3
3	Charlie	101	NULL
4	Adam	104	1

Employee1	Employee2
Alice	Charlie
Bob	Charlie
Adam	Alice

7. NATURAL JOIN

The NATURAL JOIN keyword is based on all columns in the two tables that have the same name and same datatype and selects rows with equal values in the relevant columns.

Syntax:

```
SELECT columns
FROM table1
NATURAL JOIN table2;
```

Example:

```
SELECT * FROM employees
NATURAL JOIN departments;
```

employee_id	name	department_id	department_name
1	Alice	101	HR
3	Charlie	101	HR
2	Bob	102	IT

Summary

- **INNER JOIN**: Selects records with matching values in both tables.
- **LEFT JOIN**: Selects all records from the left table and matched records from the right table.
- **RIGHT JOIN**: Selects all records from the right table and matched records from the left table.
- **FULL JOIN**: Selects all records when there is a match in either table.
- **CROSS JOIN**: Returns the Cartesian product of both tables.
- **SELF JOIN**: Joins a table with itself.
- **NATURAL JOIN**: Joins tables based on columns with the same name and data type.