JOIN IN SQL

OBJECTIVES:

- To know about the SQL joins such as inner join, natural join, outer join.

OVERVIEW:

SQL JOIN clause is used to query and access data from multiple tables by establishing logical relationships between them. It can access data from multiple tables simultaneously using common key values shared across different tables.

We can use SQL JOIN with multiple tables. It can also be paired with other clauses, the most popular use will be using JOIN with WHERE clause to filter data retrieval.

Types of SQL joins:

a) SQL INNER JOIN:

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

The syntax for SQL INNER JOIN is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
INNER JOIN table2
ON  table1.matching_column = table2.matching_column;
```

Here,

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

b) SQL Natural join:

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables. Both table must have at least one common column with same column name and same data type. The two table are joined using Cross join. DBMS will look for a common column with same name and data type Tuples having exactly same values in common columns are kept in result.

Syntax:

The syntax for SQL NATURAL JOIN is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
NATURAL JOIN table2
```

c) LEFT JOIN

LEFT JOIN returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax

The syntax of LEFT JOIN in SQL is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

d) SQL RIGHT JOIN

RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join.It is very similar to LEFT JOIN For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

The syntax of RIGHT JOIN in SQL is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

e) SQL FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from

both tables. For the rows for which there is no matching, the result-set will contain NULL values.

Syntax

The syntax of SQL FULL JOIN is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

LAB WORK

a. Create two tables Departments and Employees .

Query:

To Create Tables:

For Departments:

```
CREATE TABLE Departments(
        dept_id int NOT NULL PRIMARY KEY,
        dept_name varchar(50),
         location varchar(50)
    );
```

OUTPUT

```
CREATE TABLE Departments(
        dept_id int NOT NULL PRIMARY KEY,
        dept_name varchar(50),
         location varchar(50)
    );
SELECT* from Departments;
```

esults | Messages

dept_id	dept_name	location

For Employees:

```
CREATE TABLE Employees(
        emp_id int NOT NULL PRIMARY KEY,
        emp_name varchar(50),
        dept_id int,
          age int,
```

salary int

FOREIGN KEY (dept_id) REFERENCES Departments(dept_id),

);

OUTPUT

```
CREATE TABLE Departments(
        dept_id int NOT NULL PRIMARY KEY,
        dept_name varchar(50),
         location varchar(50)
    );
SELECT* from Departments;
```

esults Messages

dept_id	dept_name	location

b. Insert the values into the tables.

INSERT INTO Departments VALUES
(1,'HR','New York'),
(2,'Finance','San Francisco'),
(3,'Engineering','Borton'),
(4,'Sales','Chicago'),
(5, ‘Marketing’ , ‘Texas);

```
INSERT INTO Departments VALUES
(1,'HR','New York'),
(2,'Finance','San Francisco'),
(3,'Engineering','Borton'),
(4,'Sales','Chicago'),
(5, 'Marketing' , 'Texas');
```

%

Results Messages

dept_id	dept_name	location
1	HR	New York
2	Finance	San Francisco
3	Engineering	Borton
4	Sales	Chicago
5	Marketing	Texas

INSERT INTO Employees VALUES
(1,'Anchal',1,28,60000),
(2,'Nisha',2,32,75000),

(3,'Ayush',3,26,80000),
(4,'Arpan',3,29,82000),
(5,'Roshan',4,27,55000),
(6,'Virat',4,30,58000),
(7,'Rohit',1,31,82000);

OUTPUT

```
INSERT INTO Employees VALUES
(1,'Anchal',1,28,60000),
(2,'Nisha',2,32,75000),
(3,'Ayush',3,26,80000),
(4,'Arpan',3,29,82000),
(5,'Roshan',4,27,55000),
(6,'Virat',4,30,58000),
(7,'Rohit',1,31,82000);
SELECT * FROM Employees;
```

0 %

Results | Messages

emp_id	emp_name	dept_id	age	salary
1	Anchal	1	28	60000
2	Nisha	2	32	75000
3	Ayush	3	26	80000
4	Arpan	3	29	82000
5	Roshan	4	27	55000
6	Virat	4	30	58000
7	Rohit	1	31	82000

c. Write a Query to retrieve a list of employees along with their departments name.

Query:

```
SELECT*FROM Employees;
SELECT E.emp_name,D.dept_name  FROM Departments as D
INNER JOIN Employees AS E
ON d.dept_id=E.dept_id;
```

OUTPUT

```
SELECT E.emp_name,D.dept_name  FROM Departments as D
INNER JOIN Employees AS E
ON d.dept_id=E.dept_id;
```

0 %

Results | Messages

emp_name	dept_name
Anchal	HR
Nisha	Finance
Ayush	Engineering
Arpan	Engineering
Roshan	Sales
Virat	Sales
Rohit	HR

d. Write a Query to retrieve a list of employees and their departments Showing employees even if not assigned to any departments .

Query:

```
SELECT E.emp_name,D.dept_name  FROM Departments as D
LEFT JOIN Employees AS E
ON d.dept_id=E.dept_id;
```

OUTPUT

```
SELECT E.emp_name,D.dept_name  FROM Departments as D
LEFT JOIN Employees AS E
ON d.dept_id=E.dept_id;
```

%

Results | Messages

emp_name	dept_name
Anchal	HR
Rohit	HR
Nisha	Finance
Ayush	Engineering
Arpan	Engineering
Roshan	Sales
Virat	Sales
NULL	Marketing

e. Retrieve a list of all departments and their employees include departments even if they have no employees.

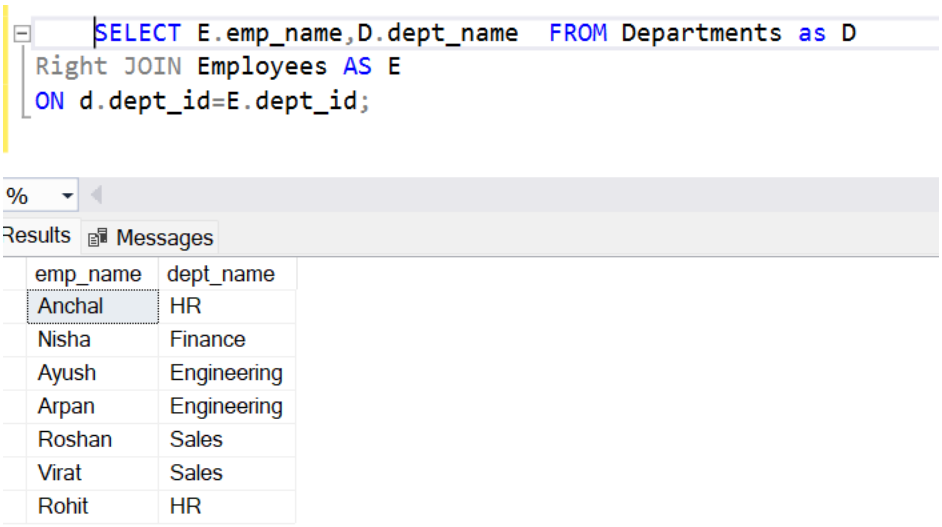
Query:

```
SELECT E.emp_name,D.dept_name  FROM Departments as D

Right JOIN Employees AS E

ON d.dept_id=E.dept_id;
```

OUTPUT



```
    SELECT E.emp_name,D.dept_name  FROM Departments as D
Right JOIN Employees AS E
ON d.dept_id=E.dept_id;
```

%

Results | Messages

emp_name	dept_name
Anchal	HR
Nisha	Finance
Ayush	Engineering
Arpan	Engineering
Roshan	Sales
Virat	Sales
Rohit	HR

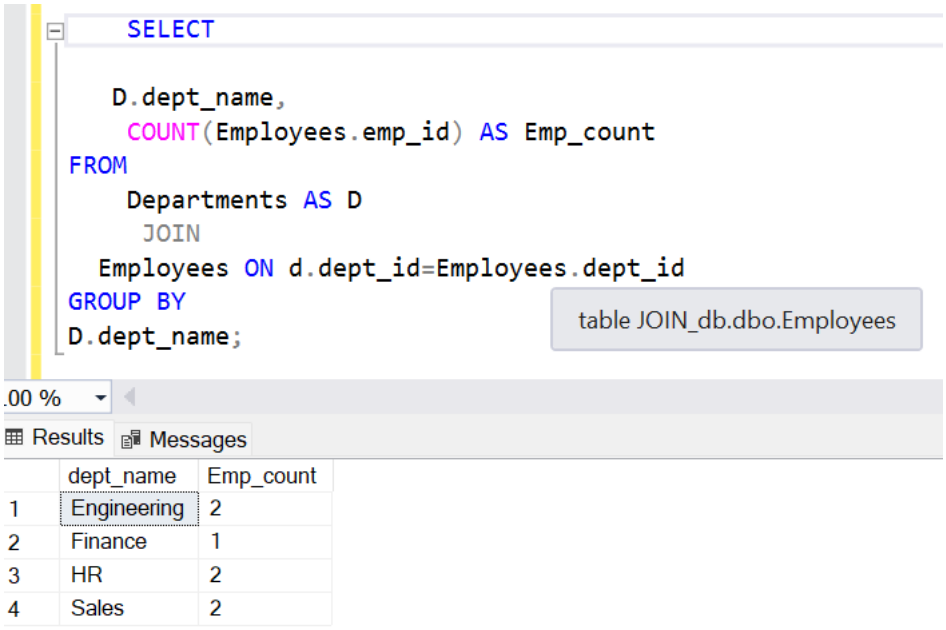
e. Count the number of employees in each department.

Query:

```
SELECT
    D.dept_name,
  COUNT(Employees.emp_id) AS Emp_count
FROM
   Departments AS D
 JOIN
  Employees ON d.dept_id=Employees.dept_id
GROUP BY
```

D.dept_name;

OUTPUT



f. Find the name of employees and their departments where the salary id greater than 60000.

Query:

SELECT

e. SELECT

Employees.emp_name,

Departments.dept_name,

Employees.salary

FROM Employees

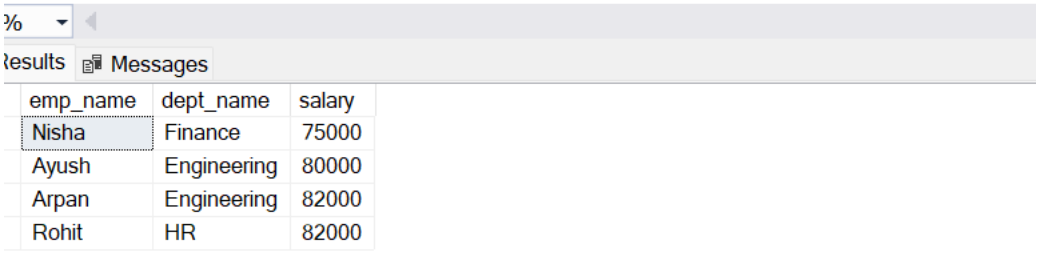
JOIN

Departments ON Employees.dept_id = Departments.dept_id

WHERE

Employees.salary > 60000;OUTPUT

```
    SELECT
    Employees.emp_name,
    Departments.dept_name,
    Employees.salary
FROM Employees
JOIN
    Departments ON Employees.dept_id = Departments.dept_id
WHERE
    Employees.salary > 60000;
```



%

Results Messages

emp_name	dept_name	salary
Nisha	Finance	75000
Ayush	Engineering	80000
Arpan	Engineering	82000
Rohit	HR	82000

g. Find the highest paid employee in each department.

Query:

SELECT

SELECT

e.emp_name,

d.dept_name,

e.salary

FROM

Employees e

JOIN

Departments d ON e.dept_id = d.dept_id

WHERE

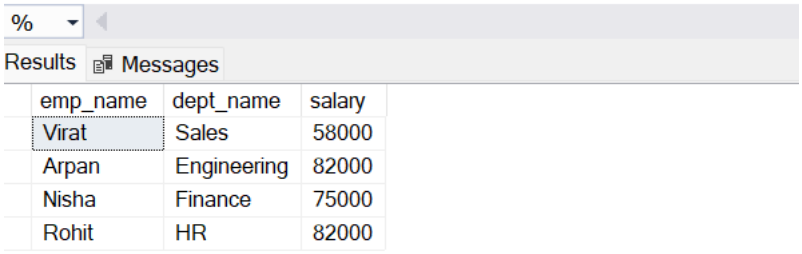
e.salary = (SELECT MAX(salary)

FROM Employees

WHERE dept_id = e.dept_id);

OUTPUT

```sql
SELECT
    e.emp_name,
    d.dept_name,
    e.salary
FROM
    Employees e
JOIN
    Departments d ON e.dept_id = d.dept_id
WHERE
    e.salary = (SELECT MAX(salary)
                FROM Employees
                WHERE dept_id = e.dept_id);
```



%

Results Messages

emp_name	dept_name	salary
Virat	Sales	58000
Arpan	Engineering	82000
Nisha	Finance	75000
Rohit	HR	82000

CONCLUSION

In conclusion, understanding SQL joins is crucial for efficient database management, enabling the combination of data from multiple tables based on related columns. Mastery of inner, outer, left, and right joins enhances data querying capabilities, providing comprehensive insights. Proper use of joins optimizes data retrieval and supports robust, scalable database applications.