SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

DBMS Complex SQL Queries

Course Name : 23CAT603 - DATA BASE MANAGEMENT SYSTEM

Class : I Year / II Semester

Unit II – Complex SQL Queries

Agenda

SQL Joins (Inner, Left, Right and Full Join)

SQL Joins (Inner, Left, Right and Full Join)

SQL Join operation combines data or rows from two or more tables based on a common field between them.

We will learn about **Joins in SQL,** covering JOIN types, syntax, and examples.

SQL JOIN

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.

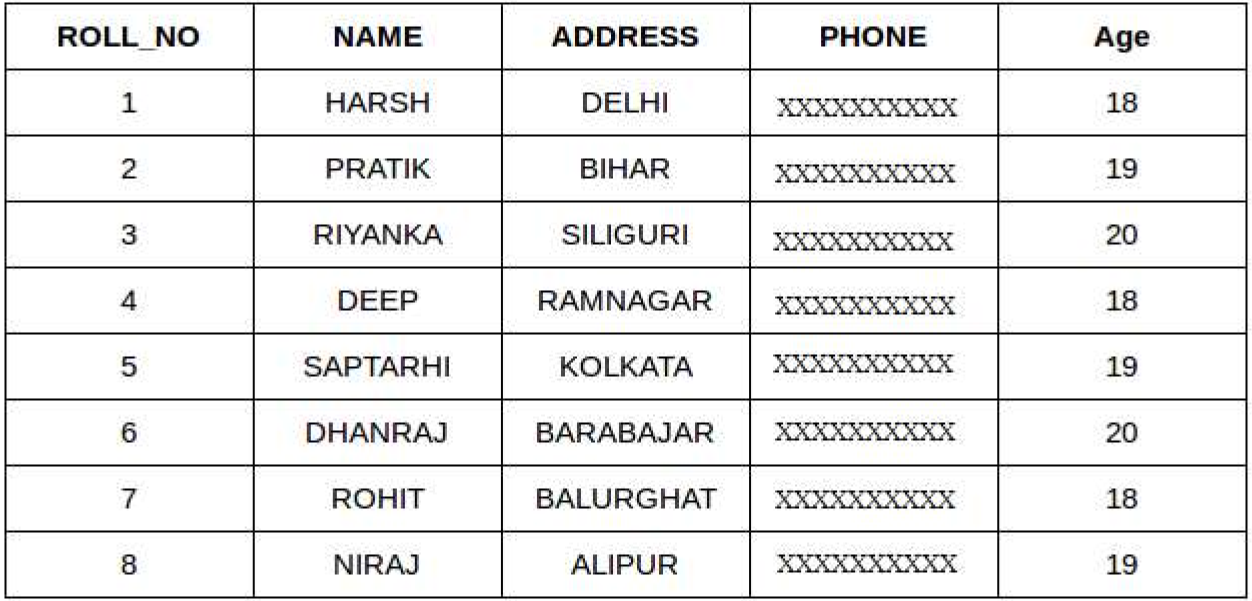
We can use SQL JOIN with multiple tables. It can also be paired with other clauses, the most popular use will be using JOIN with **WHERE clause** to filter data retrieval.

SQL JOIN Example

Consider the two tables below as follows:

SQL Joins (Inner, Left, Right and Full Join)

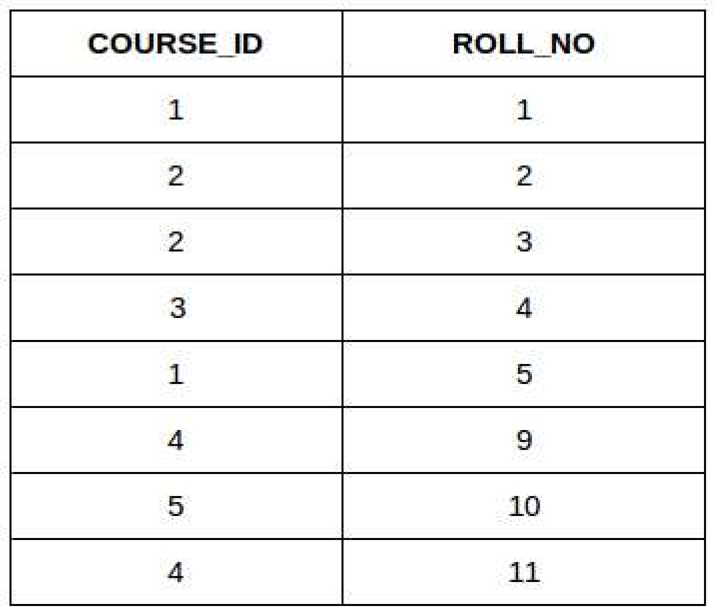
Student:



ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

SQL Joins (Inner, Left, Right and Full Join)

StudentCourse :



COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

SQL Joins (Inner, Left, Right and Full Join)

Both these tables are connected by one common key (column) i.e **ROLL_NO**.

We can perform a JOIN operation using the given SQL query:

```
SELECT s.roll_no, s.name, s.address,
s.phone, s.age, sc.course_id
FROM Student s
JOIN StudentCourse sc ON s.roll_no =
sc.roll_no;
```

Types of JOIN in SQL

Types of JOIN in SQL

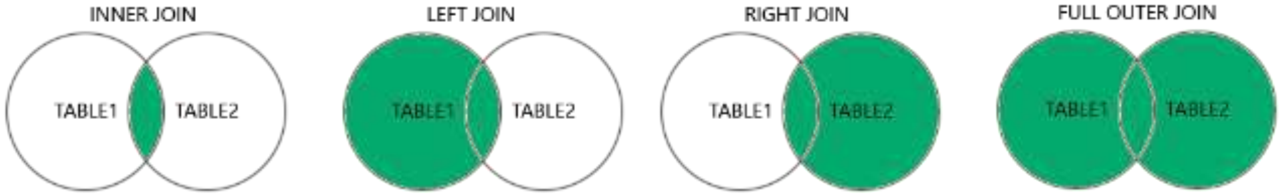
There are many types of Joins in SQL. Depending on the use case, you can use different type of SQL JOIN clause.

Different Types of SQL JOINs

Here are the frequently used SQL JOIN types:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- Natural join



SQL INNER JOIN

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.

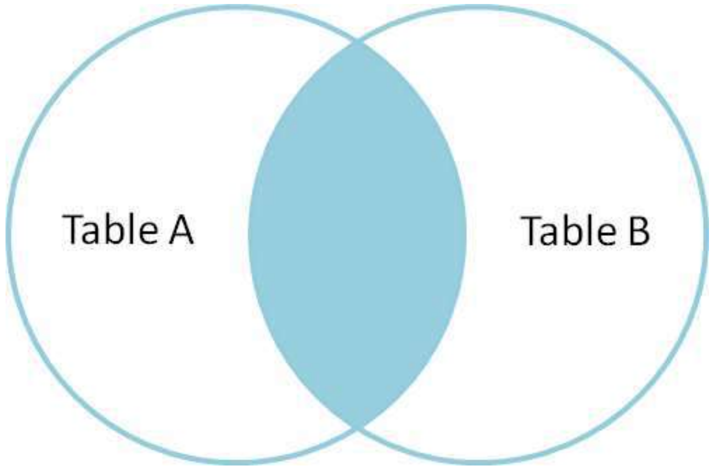
Note*: We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.*

SQL INNER JOIN

INNER JOIN Example

Let's look at the example of INNER JOIN clause, and understand it's working.

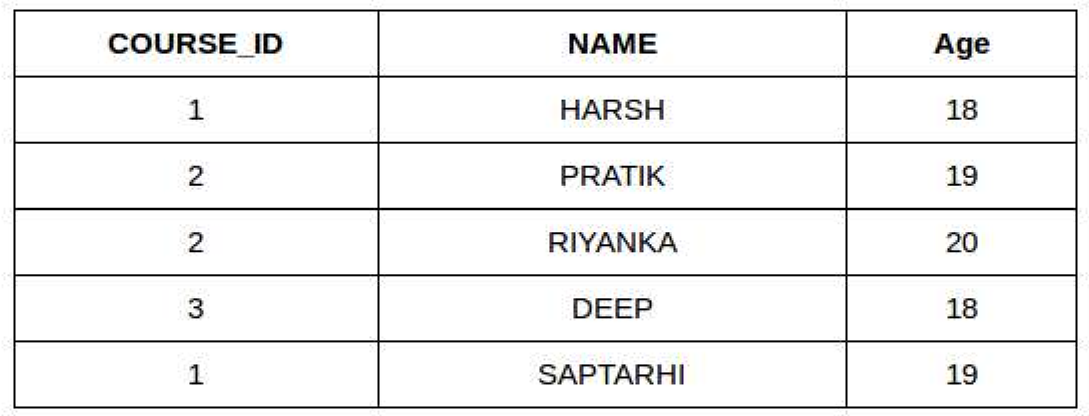
This query will show the names and age of students enrolled in different courses.



SQL INNER JOIN

```
SELECT StudentCourse.COURSE_ID, Student.NAME,
Student.AGE FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Output:



COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

SQL 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.

<u>Syntax:</u>

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;
```

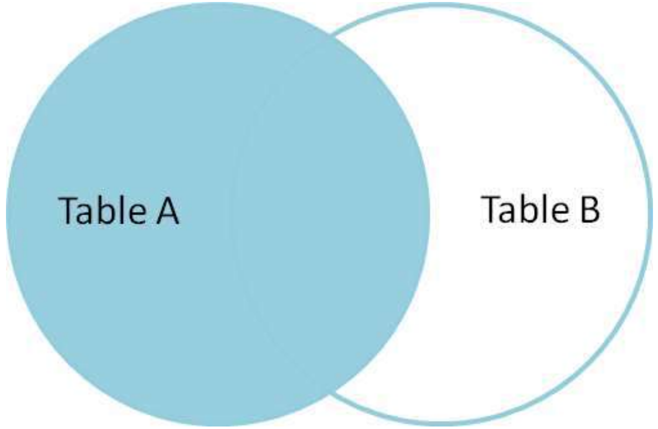
Here,

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

Note: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.



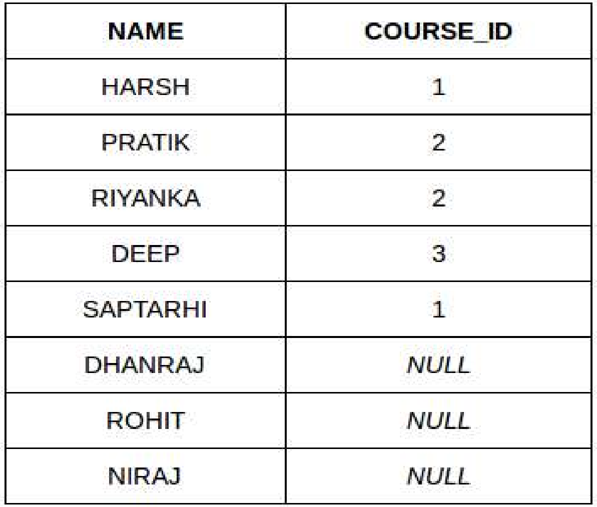
SQL LEFT JOIN

LEFT JOIN Example

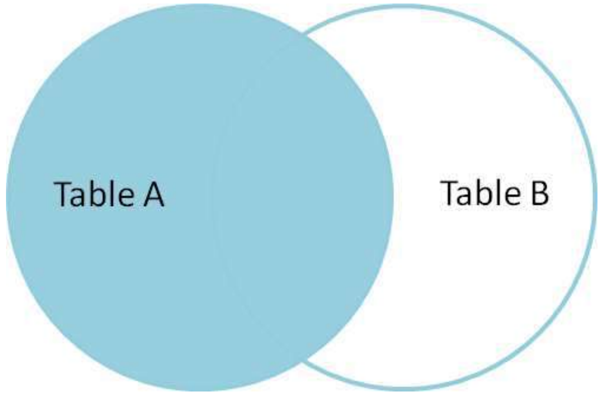
Let’s look at the example of LEFT JOIN clause, and understand it’s working

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*



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.

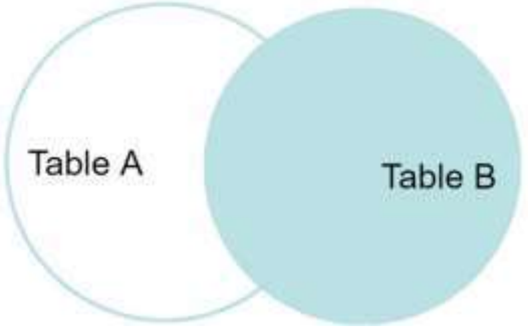
Syntax:

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;

Here,

table1: First table.
table2: Second table
matching_column: Column common to both the tables.
Note: We can also use RIGHT OUTER JOIN instead of RIGHT JOIN, both are the same.

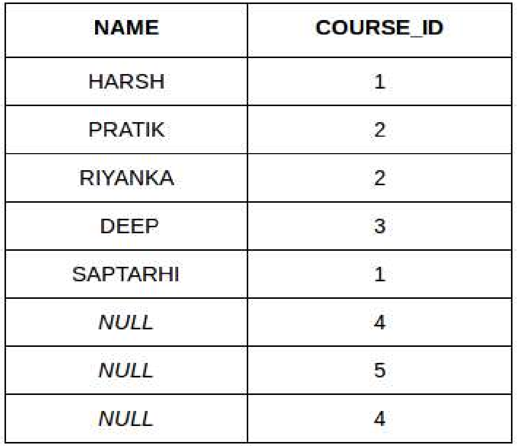


SQL RIGHT JOIN

RIGHT JOIN Example:
Let’s look at the example of RIGHT JOIN clause, and understand it’s working

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

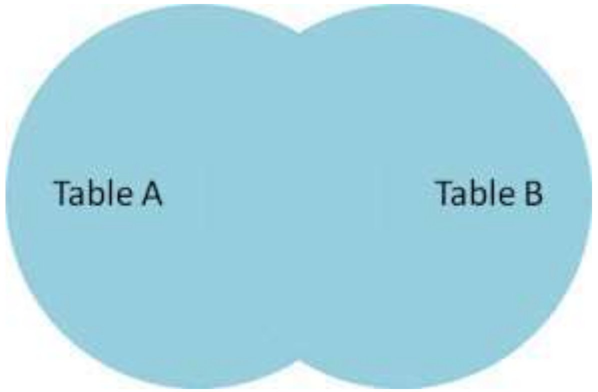
Output:



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

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.



SQL FULL JOIN

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;

Here,

- **table1**: First table.
- **table2**: Second table
- **matching_column**: Column common to both the tables.

FULL JOIN Example

Let's look at the example of FULL JOIN clause, and understand it's working

SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	NULL
ROHIT	NULL
NIRAJ	NULL
NULL	4
NULL	5
NULL	4

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.

Natural join Example:

Look at the two tables below- Employee and Department

SQL Natural join (?)

Natural join Example:

Look at the two tables below- Employee and Department

Employee		
Emp_id	Emp_name	Dept_id
1	Ram	10
2	Jon	30
3	Bob	50

Department	
Dept_id	Dept_name
10	IT
30	HR
40	TIS

Problem: Find all Employees and their respective departments.

Solution Query: (Employee) ? (Department)

Emp_id	Emp_name	Dept_id	Dept_id	Dept_name
1	Ram	10	10	IT
2	Jon	30	30	HR
Employee data			Department data	

SQL Outer Join

In a relational DBMS, we follow the principles of normalization that allows us to minimize the large tables into small tables. By using a select statement in Joins, we can retrieve the big table back. Outer joins are of following three types.

1.Left outer join

2.Right outer join

3.Full outer join

Creating a database : Run the following command to create a database.

```
Create database testdb;
```

Using the database : Run the following command to use a database.

```
use testdb;
```

SQL Outer Join

Adding table to the database :

Run the following command to add tables to a database.

```
CREATE TABLE Students (
  StudentID int,
  LastName varchar(255),
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
);
```

SQL Outer Join

Inserting rows into database :

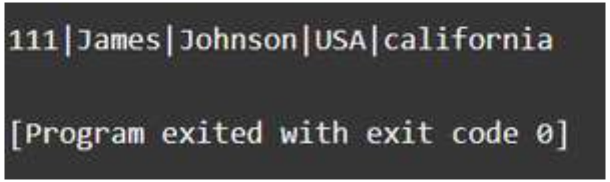
```
INSERT INTO students (
StudentID,
LastName,
FirstName,
Address,
City
)
VALUES (
111,
'James',
 'Johnson',
 'USA',
 california
);
```

SQL Outer Join

Output of database :

Type the following command to get output.

SELECT * FROM students;

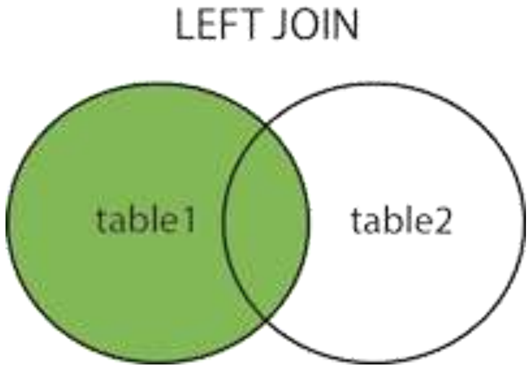


```
111|James|Johnson|USA|california

[Program exited with exit code 0]
```

Types of outer join

1.**Left Outer Join** : The left join operation returns all record from left table and matching records from the right table. On a matching element not found in right table, NULL is represented in that case.

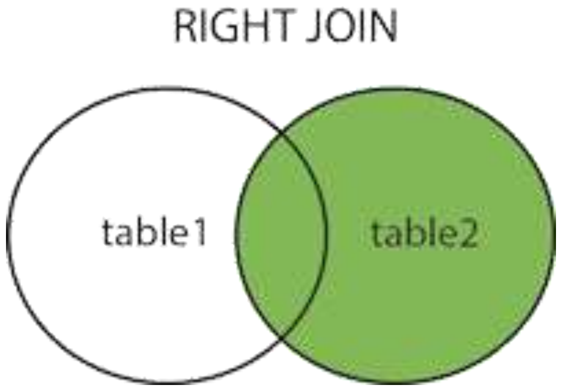


Syntax :

SELECT column_name(s) FROM table1 LEFT JOIN Table2 ON Table1.Column_Name=table2.column_name;

Types of outer join

2. Right Outer Join : The right join operation returns all record from right table and matching records from the left table. On a matching element not found in left table, NULL is represented in that case.

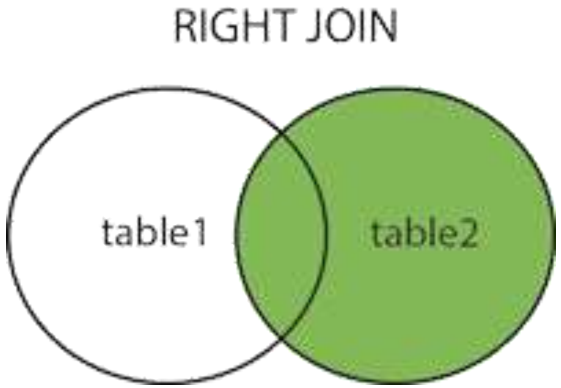


Syntax :

SELECT column_name(s) FROM table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name;

Types of outer join

2. Right Outer Join : The right join operation returns all record from right table and matching records from the left table. On a matching element not found in left table, NULL is represented in that case.

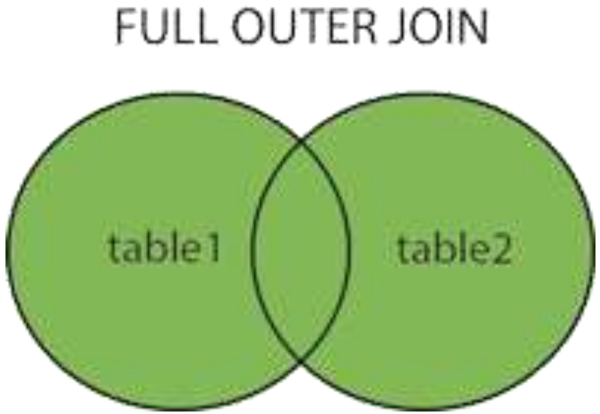


Syntax :

SELECT column_name(s) FROM table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name;

Types of outer join

3. Full Outer Join : The full outer Join keyword returns all records when there is a match in left or right table records.



Syntax: SELECT column_name FROM table1 FULL OUTER JOIN table2 ON table1.columnName = table2.columnName WHERE condition;

Types of outer join

Example :

Creating 1st Sample table students.

```
CREATE TABLE students (
 id INTEGER,
 name TEXT NOT NULL,
 gender TEXT NOT NULL
);
-- insert some values
INSERT INTO students VALUES (1, 'Ryan', 'M');
INSERT INTO students VALUES (2, 'Joanna', 'F');
INSERT INTO students Values (3, 'Moana', 'F');
```

Types of outer join

Creating 2nd sample table college.

```
CREATE TABLE college (
 id INTEGER,
 classTeacher TEXT NOT NULL,
 Strength TEXT NOT NULL
);
-- insert some values
INSERT INTO college VALUES (1, 'Alpha', '50');
INSERT INTO college VALUES (2, 'Romeo', '60');
INSERT INTO college Values (3, 'Charlie', '55');
```

Types of outer join

Performing outer join on above two tables.

```
SELECT College.classTeacher, students.id
FROM College
FULL OUTER JOIN College ON College.id=students.id
ORDER BY College.classTeacher;
```

The above code will perform a full outer join on tables students and college and will return the output that matches the id of college with id of students. The output will be class Teacher from college table and id from students table. The table will be ordered by class Teacher from college table.

Class Teacher	Id
Alpha	1
Romeo	2
Charlie	3

SQL LEFT JOIN

SQL LEFT JOIN command returns all records from the left table and matching records from the right table. If there is no matching record in the right table, the right table records will contain **NULL values**.

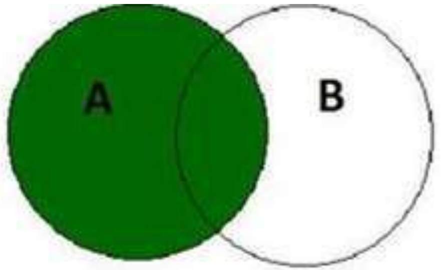
LEFT JOIN in SQL

LEFT JOIN in SQL is used to combine rows from two or more tables, based on a related column between them. It returns all rows from the left table and matching records from the right table.

If a certain row is present in the left table but not in the right table, the result will include this row, but with a NULL value in each column from the right table. If a record from the right table is not on the left, it will not be included in the result.

SQL LEFT JOIN Venn Diagram

This VENN diagram shows how a LEFT JOIN works in SQL.



SQL LEFT JOIN

Syntax

The LEFT JOIN syntax is:

```
SELECT column_name(s)
FROM tableA
LEFT JOIN tableB ON tableA.column_name = tableB.column_name;
```

SQL LEFT JOIN Example

Let’s look at an example of LEFT JOIN in SQL to understand it better.

Let’s consider two tables Emp containing details of the Employee working in the particular department, and department table containing the details of the department

Employee Table

SQL LEFT JOIN

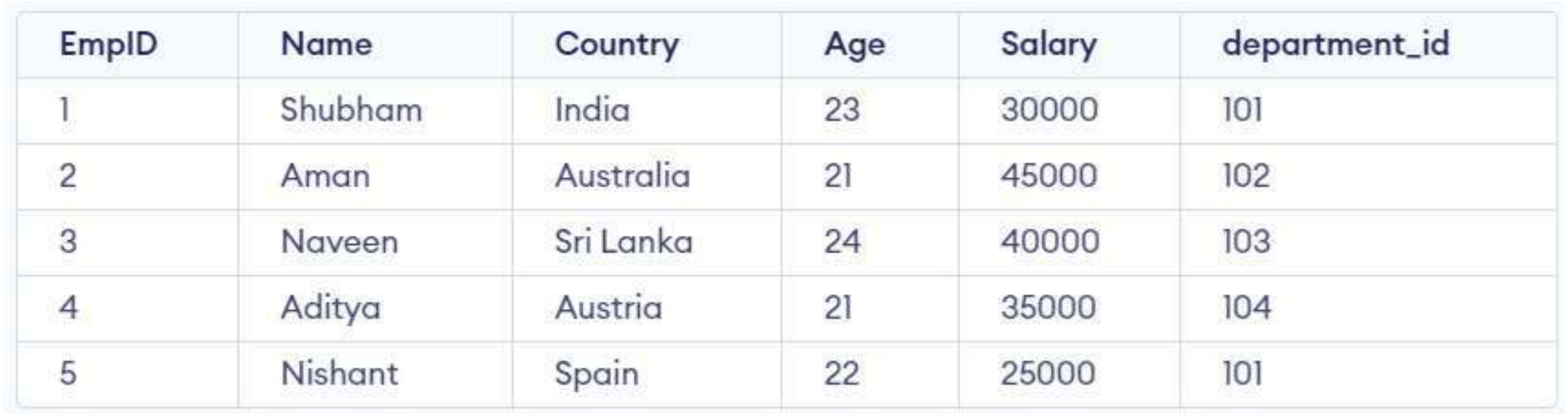
Employee Table

Query:

```
CREATE TABLE Emp (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Country VARCHAR(50),
    Age INT,
    Salary INT,
    department_id INT
);
INSERT INTO Emp (EmpID, Name, Country, Age, Salary, department_id)
VALUES (1, 'Shubham', 'India', 23, 30000, 101),
       (2, 'Aman', 'Australia', 21, 45000, 102),
       (3, 'Naveen', 'Sri Lanka', 24, 40000, 103),
       (4, 'Aditya', 'Austria', 21, 35000, 104),
       (5, 'Nishant', 'Spain', 22, 25000, 101);
```

SQL LEFT JOIN

Output



EmpID	Name	Country	Age	Salary	department_id
1	Shubham	India	23	30000	101
2	Aman	Australia	21	45000	102
3	Naveen	Sri Lanka	24	40000	103
4	Aditya	Austria	21	35000	104
5	Nishant	Spain	22	25000	101

SQL LEFT JOIN

Department Table

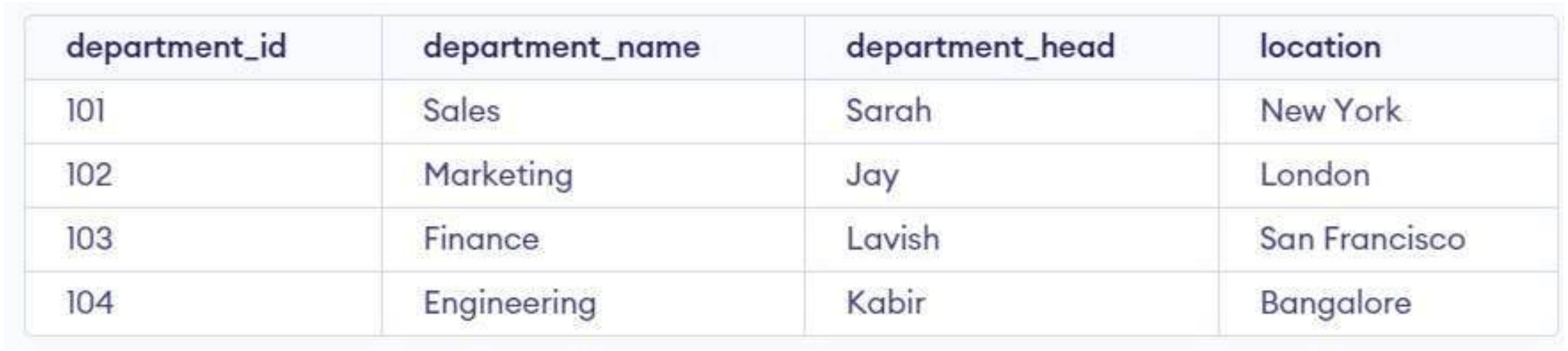
Query:

```
  CREATE TABLE department (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50),
    department_head VARCHAR(50),
    location VARCHAR(50)
);

INSERT INTO department (department_id, department_name, department_head, location)
VALUES (101, 'Sales', 'Sarah', 'New York'),
       (102, 'Marketing', 'Jay', 'London'),
       (103, 'Finance', 'Lavish', 'San Francisco'),
       (104, 'Engineering', 'Kabir', 'Bangalore');
SELECT * FROM department;
```

SQL LEFT JOIN

Output



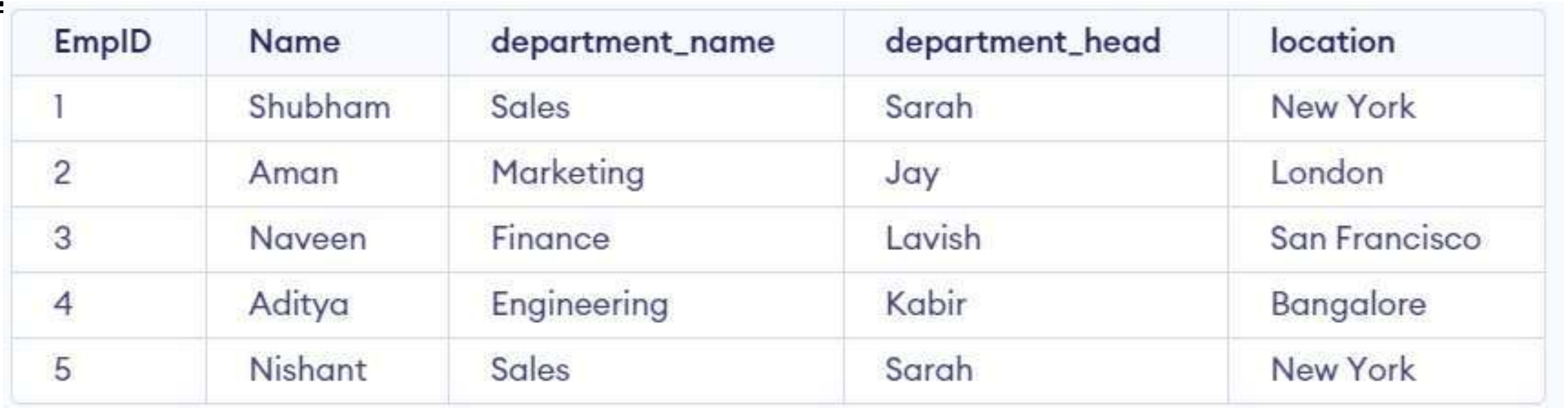
department_id	department_name	department_head	location
101	Sales	Sarah	New York
102	Marketing	Jay	London
103	Finance	Lavish	San Francisco
104	Engineering	Kabir	Bangalore

SQL LEFT JOIN

To perform left-join on these two tables we will use the following SQL query:

```
SELECT Emp.EmpID, Emp.Name, department.
department_name, department.department_head,
department.location
FROM Emp
LEFT JOIN department ON Emp.department_id = department.department_id;
```

Output:



EmpID	Name	department_name	department_head	location
1	Shubham	Sales	Sarah	New York
2	Aman	Marketing	Jay	London
3	Naveen	Finance	Lavish	San Francisco
4	Aditya	Engineering	Kabir	Bangalore
5	Nishant	Sales	Sarah	New York

As left join gives the matching rows and the rows that are present in the left table but not in the right table. Here in this example, we see that the employees that do not work in a particular department, i.e., having dept no values as [NULL], contain [NULL] values of dept name and location after the left join.

SQL LEFT JOIN

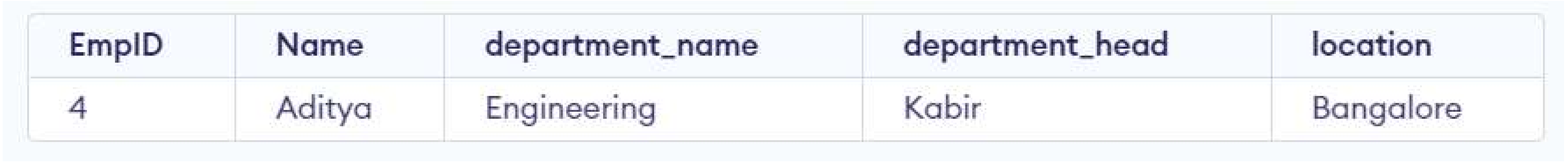
SQL LEFT JOIN with WHERE Clause Example

In this example, we will add a WHERE clause that specifies to only return results where the “location” column in the department table equals ‘Bangalore’. This will filter the results to only show employees who belong to a department located in Bangalore, and departments that have no employees will not be returned in the results.

Query:

```
SELECT e.EmpID, e.Name, d.department_name,
d.department_head, d.location
FROM Emp e
LEFT JOIN department d ON e.department_id
 = d.department_id
WHERE d.location = 'Bangalore';
```

Output:



EmpID	Name	department_name	department_head	location
4	Aditya	Engineering	Kabir	Bangalore

SQL LEFT JOIN

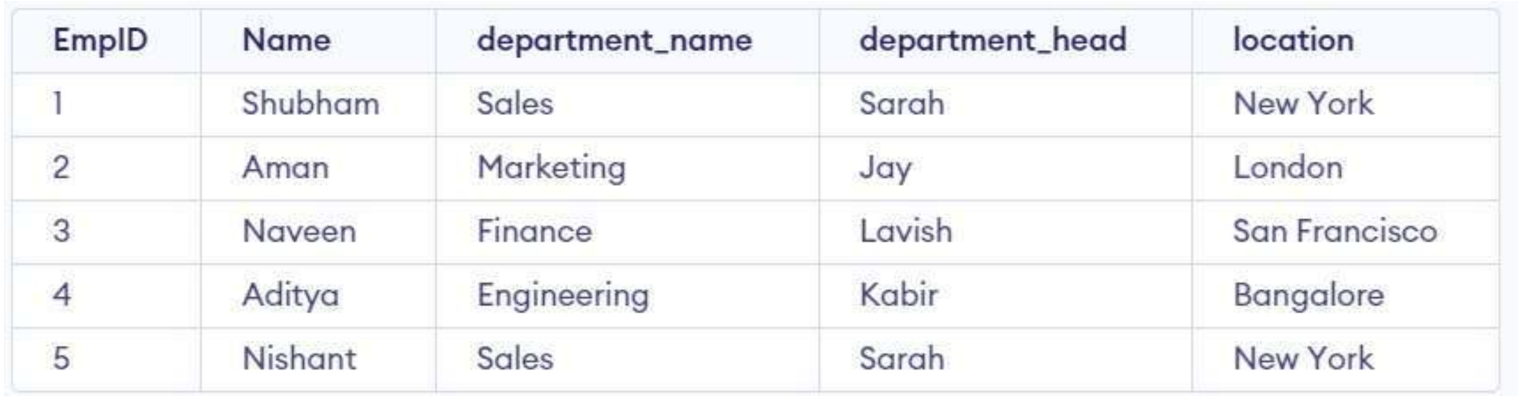
SQL LEFT JOIN as Aliases Example

In this query, we'll use aliases "e" for the Emp table and "d" for the department table. The SELECT statement references these aliases for each column, making the query easier to read and type. Aliases simplify code and improve readability, especially with long or complex table names.

Query:

```
SELECT e.EmpID, e.Name, d.department_name,
d.department_head, d.location
FROM Emp e
LEFT JOIN department d ON
e.department_id = d.department_id;
```

Output:



EmpID	Name	department_name	department_head	location
1	Shubham	Sales	Sarah	New York
2	Aman	Marketing	Jay	London
3	Naveen	Finance	Lavish	San Francisco
4	Aditya	Engineering	Kabir	Bangalore
5	Nishant	Sales	Sarah	New York

SQL LEFT JOIN

Important Points About SQL LEFT JOIN

- LEFT JOIN in SQL returns all records from the left table and matching records from the right table.
- NULL values are included for unmatched records on the right side.
- LEFT JOIN is used to combine data based on related columns.
- Aliases can simplify queries with long table names.
- LEFT JOIN with WHERE clause is used for filtering records.

SQL RIGHT JOIN

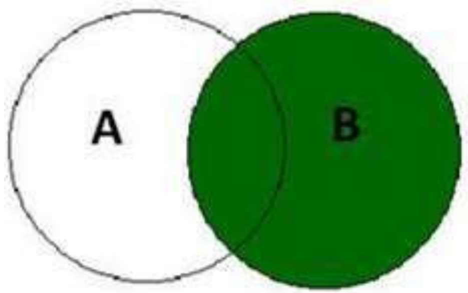
The **SQL RIGHT JOIN Keyword** is a powerful tool used to combine records from two tables. SQL RIGHT JOIN returns all records from the right table, and the matching records from the left table in the results set.

SQL RIGHT JOIN Keyword

The **RIGHT JOIN in SQL** returns a table that contains all the records from the right table and only matching records from the left table.

In simpler terms, if a row is present in the right table but not in the left table, the result will include this row with NULL values for columns from the left table. Conversely, if a record from the left table is not in the right table, it will not be included in the result.

The **Visual Representation of RIGHT JOIN** is shown below in the **Venn Diagram**.



SQL RIGHT JOIN

Syntax

```
SELECT column_name(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.column_name = tableB.column_name;
```

SQL RIGHT JOIN Examples

In this example, we will consider two tables employee table containing details of the employees working in the particular department the and department table containing the details of the department

Employee Table

emp_no	emp_name	dept_no
E1	Varun Singhal	D1
E2	Amrita Aggarwal	D2
E3	Ravi Anand	D3

SQL RIGHT JOIN

```
CREATE TABLE employee (
  emp_no CHAR(3) PRIMARY KEY,  -- Adjust length if needed for employee numbers
  emp_name VARCHAR(50) NOT NULL,
  dept_no CHAR(2)
);
INSERT INTO employee (emp_no, emp_name, dept_no)
VALUES ('E1', 'Varun Singhal', 'D1'),
       ('E2', 'Amrita Aggarwal', 'D2'),
       ('E3', 'Ravi Anand', 'D3');
```

Department Table:

dept_no	d_name	location
D1	IT	Delhi
D2	HR	Hyderabad
D3	Finance	Pune
D4	Testing	Noida
D5	Marketing	Mathura

SQL Query to Create department table

```
CREATE TABLE department (
  dept_no CHAR(2) PRIMARY KEY,
  d_name VARCHAR(20) NOT NULL,
  location VARCHAR(50)
);

INSERT INTO department (dept_no, d_name, location)
VALUES ('D1', 'IT', 'Delhi'),
       ('D2', 'HR', 'Hyderabad'),
       ('D3', 'Finance', 'Pune'),
       ('D4', 'Testing', 'Noida'),
       ('D5', 'Marketing', 'Mathura');
```

SQL Query to Create department table

Now, we will perform SQL RIGHT JOIN on these two tables.

Query:

SELECT emp_no , emp_name ,d_name, location
FROM employee
RIGHT JOIN dept
ON employee.dept_no = department.dept_no;

Explanation: As right join gives the matching row
right table but not in the left table. Here in this
that contains no employee contains **[NULL] val**
performing the right join.

emp_no	emp_name	d_name	location
E1	Varun Singhal	IT	Delhi
E2	Amrita Aggarwal	HR	Hyderabad
E3	Ravi Anand	Finance	Pune
[NULL]	[NULL]	Testing	Noida
[NULL]	[NULL]	Marketing	Mathura

Applications of SQL RIGHT JOIN

Merging Data: Allows to merge data from different tables in database.

Ensuring Completeness: A RIGHT JOIN ensures that all records from the right table are included in the result, even if there are no corresponding matches in the left table

Handling Missing Values: Allows to look for missing values in one of the table. For example, combining customer and orders table allows to look at customers and their orders.

Analyzing Relationships: Useful in finding patterns and relations between data.

Important Points About SQL RIGHT JOIN

Right JOIN allows to join two table, keeping all the data or right table and only matching data of left table.

Right JOIN is a type of outer join in SQL.

It allows us to deal with missing values in database and also helps in analyzing relationships between data.

Simplifies queries by eliminating the need for complex conditional logic using CASE statements.

SQL FULL JOIN

In SQL, the FULL JOIN (or FULL OUTER JOIN) is a powerful technique used to combine records from two or more tables. Unlike an INNER JOIN, which only returns rows where there are matches in both tables, a FULL JOIN retrieves all rows from both tables, filling in NULL values where matches do not exist. In this article, We will learn about SQL FULL JOIN by understanding various examples in detail.

SQL FULL JOIN

The FULL JOIN or FULL OUTER JOIN in SQL is used to retrieve all rows from both tables involved in the join, regardless of whether there is a match between the rows in the two tables.

It combines the results of both a LEFT JOIN and a RIGHT JOIN.

When there is no match, the result will include NULLs for the columns of the table that does not have a matching row.

Tip: We can use FULL JOIN to combine multiple tables, by sequentially performing FULL JOIN on two tables at a time.

SQL FULL JOIN

Syntax

```
SELECT columns
FROM table1
FULL JOIN table2
ON table1.column = table2.column;
```

Explanation:

SELECT columns: Specifies the columns to retrieve.

FROM table1: The first table to be joined.

FULL JOIN table2: Specifies the second table to join with the first table using a FULL JOIN.

ON table1.column = table2.column: Defines the condition to match rows between the two tables.

This query retrieves all records from both table1 and table2, returning NULL where there are no matches.

SQL FULL JOIN

Examples of SQL FULL JOIN

Let’s look at some examples of the FULL JOIN in SQL and understand it’s working.

First, let’s create a demo database and two tables on which we will perform the **JOIN**.

Table 1- Students

ID	NAME	BRANCH	NUMBER
1	SURYANSH JOHARI	CS	984012
2	AMAN SHARMA	IT	771346
3	DEV VERMA	ME	638587
4	JOY SMITH	CE	876691
5	CHARLES GATTO	EE	997679

Table 2- Library

BOOK_ID	BOOK_NAME	ISSUED_ ON	DUE_DATE
1	RD SHARMA	2023-01-01	2023-01-08
2	GATE CRACKER	2023-02-02	2023-02-09
3	MORRIS MANO	2023-03-03	2023-03-10
4	NK PUBLICATION S	2023-04-04	2023-04-11
5	BIG BANG THEORY	2023-05-05	2023-05-12

SQL FULL JOIN

Example 1: Joining Multiple Tables with Full Join

We want to demonstrate how to use FULL JOIN to combine two or more tables based on common columns. In this case, we will create two additional tables: Authors and Publishers, and join them with the Books table using a FULL JOIN.

```
SELECT
   b.BOOK_ID,
   b.BOOK_NAME,
   a.AUTHOR_NAME,
   p.PUBLISHER_NAME
FROM
   Books b
FULL JOIN Authors a ON b.BOOK_ID = a.AUTHOR_ID
FULL JOIN Publishers p ON b.BOOK_ID = p.PUBLISHER_ID;
```

SQL FULL JOIN

Explanation:

In this query, we used FULL JOIN to join three tables: Books, Authors, and Publishers. The FULL JOIN ensures that all records from both tables are included, whether or not they match. Here, each book has a corresponding author and publisher. If any book didn't have an author or publisher, the result would still include that row with NULL in the respective columns.

BOOK_ID	BOOK_NAME	AUTHOR_NAME	PUBLISHER_NAME
1	RD SHARMA	Ram Kumar	Pearson
2	GATE CRACKER	Shyam Sunder	Wiley
3	MORRIS MANO	Sita Singh	McGraw-Hill
4	NK PUBLICATIONS	Mohan Gupta	Springer
5	BIG BANG THEORY	Raj Kapoor	Elsevier

SQL FULL JOIN

Example 2: Full Join with WHERE Clause

Now, we want to filter the results from the above join based on a specific condition. We will select only books that have “Sharma” in the book name.

Query:

```
SELECT
   b.BOOK_ID,
   b.BOOK_NAME,
   a.AUTHOR_NAME,
   p.PUBLISHER_NAME
FROM
   Books b
FULL JOIN Authors a ON b.BOOK_ID = a.AUTHOR_ID
FULL JOIN Publishers p ON b.BOOK_ID = p.PUBLISHER_ID
WHERE
   b.BOOK_NAME LIKE '%Sharma%';
```

Output:

BOOK_ID	BOOK_NAME	AUTHOR_NAME	PUBLISHER_NAME
1	RD SHARMA	Ram Kumar	Pearson

Explanation:

In this example, the **WHERE** clause filters out all books that do not contain the word “Sharma” in their name. After applying the filter, only the record for “RD SHARMA” remains.

SQL CROSS JOIN

SQL CROSS JOIN returns all the records from the left and right tables. CROSS JOIN returns a combination of each row in the left table paired with each row in the right table.

CROSS JOIN in SQL

Cross Join in SQL produces a result set that contains the cartesian product of two or more tables. Cross join is also called a **Cartesian Join**.

When CROSS JOIN is used with a **WHERE clause**, it behaves like **INNER JOIN**, filtering the results based on specific conditions.

CROSS JOIN is the best choice when we need to match each row of one table to every other row of another table. It is helpful in many applications where we need to obtain paired combinations of records.

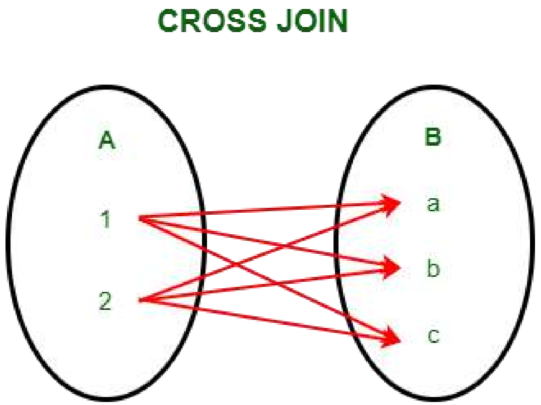
Syntax

```
SELECT * FROM table1

CROSS JOIN table2;
```

SQL CROSS JOIN Example

Let's look at some examples of CROSS JOIN statement in SQL to understand it's working.



SQL CROSS JOIN

Demo SQL Database

In this CROSS JOIN tutorial, we will use the following two tables in examples:

Table 1- Customer

ID	NAME	AGE	PHONE
1	AMIT JAIN	21	98474
2	JATIN VERMA	47	63996

Table 2- Orders

ORDER_ID	AMOUNT	PLACED_ON
101	999	2023-04-19
102	4999	2023-04-20

SQL CROSS JOIN

```
CREATE DATABASE db;
USE db;
CREATE TABLE CUSTOMER(
   ID INT,
   NAME VARCHAR(20),
   AGE INT,
   PHONE INT);
CREATE TABLE ORDERS(
   ORDER_ID INT,
   AMOUNT INT,
   PLACED_ON DATE);

INSERT INTO CUSTOMER VALUES(1,'AMIT JAIN',21,98474);
INSERT INTO CUSTOMER VALUES(2,'JATIN VERMA',47,63996);
INSERT INTO ORDERS VALUES(101,999,'2023-04-19');
INSERT INTO ORDERS VALUES(102,4999,'2023-04-20');
```

SQL CROSS JOIN

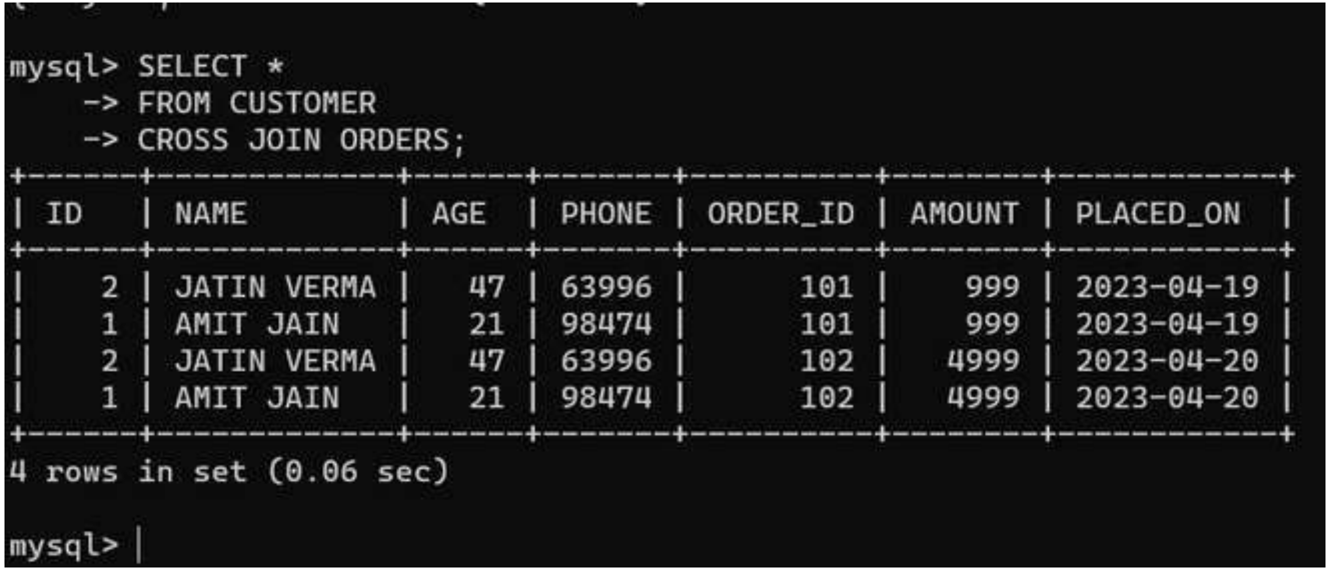
CROSS JOIN Example

In this example, we will use the CROSS JOIN command to match the data of the Customer and Orders table.

Query

SELECT *
FROM CUSTOMER
CROSS JOIN ORDERS;

Output



```
mysql> SELECT *
    -> FROM CUSTOMER
    -> CROSS JOIN ORDERS;
+------+-------------+------+-------+----------+--------+------------+
| ID   | NAME        | AGE  | PHONE | ORDER_ID | AMOUNT | PLACED_ON  |
+------+-------------+------+-------+----------+--------+------------+
|    2 | JATIN VERMA |   47 | 63996 |      101 |    999 | 2023-04-19 |
|    1 | AMIT JAIN   |   21 | 98474 |      101 |    999 | 2023-04-19 |
|    2 | JATIN VERMA |   47 | 63996 |      102 |   4999 | 2023-04-20 |
|    1 | AMIT JAIN   |   21 | 98474 |      102 |   4999 | 2023-04-20 |
+------+-------------+------+-------+----------+--------+------------+
4 rows in set (0.06 sec)

mysql> |
```

As we can see, whether the other table matches or not, the CROSS JOIN keyword returns all similar records from both tables. Therefore, if there are rows in “Customers” or “Orders” that do not match any entries in either table, those rows will also be listed.

SQL CROSS JOIN

Important Points About CROSS JOIN

CROSS JOIN performs the cross-product of records from two or more joined tables.

It is used when we want every possible combination of rows to be present in a database’s tables.

SQL CROSS JOIN with condition of WHERE Clause operates as an INNER JOIN; when used without one, it produces the cartesian product of all the rows from all the tables provided in the SQL query.

CROSS JOIN is different from other join types like INNER JOIN, **LEFT JOIN**, and **RIGHT JOIN**, as it does not require a matching condition between the tables.

SQL Self Join

Joins in SQL, a self join is a regular join that is used to join a table with itself. It basically allows us to combine the rows from the same table based on some specific conditions. It is very useful and easy to work with, and it allows us to retrieve data or information which involves comparing records within the same table.

Syntax:

SELECT columns

FROM table AS alias1

JOIN table AS alias2 ON alias1.column = alias2.column;

Explnation:

SELECT columns: With the help of this we specify the columns you want to retrieve from the self-joined table.
FROM table AS alias1: With the help of this we specify the name of the table you want to join with itself.
JOIN table AS alias2: In this we use the JOIN keyword to show that we are performing a self join on the same table.

SQL Self Join

Example:

Let’s use an illustration to further understand how the self-join functions. Assume that we have a table called “GFGemployees” with the columns employee_id, employee_name, and manager_id. Each employee in the company is assigned a manager, and using the manager-ids, we can identify each employee. We need to extract the list of employees along with the names of their managers because the manager_id column contains the manager ID for each employee

Step 1: First, we need to create the “GFGemployees” table with following query.

```
CREATE TABLE GFGemployees(employee_id
INT PRIMARY KEY, employee_name VARCHAR(50), manager_id INT);
```

Step 2: Now we will add data into the ‘GFGemployees’ table using INSERT INTO statement:

```
INSERT INTO GFGemployees (employee_id, employee_name, manager_id)
VALUES  (1, 'Zaid', 3),  (2, 'Rahul', 3),  (3, 'Raman', 4),
(4, 'Kamran', NULL),  (5, 'Farhan', 4);
```

SQL Self Join

Output

employee_id	employee_name	manager_id
1	Zaid	3
2	Rahul	3
3	Raman	4
4	Kamran	NULL
5	Farhan	4

SQL Self Join

Step 3: Explanation and implementation of Self Join

Now, we need to perform self join on the table we created i.e.”GFGemployees” in order to retrieve the list of employees and their corresponding managers name and for that we need to write a query, where we will create two different aliases for the “GFGemployees” table as “e” which will represent the GFG employee’s information and “m” will represent the manager’s information. This way by joining the table with itself using the manager_id and employee_id columns, we can generate relationship between employees and their managers.

Step 4: Query for Self-join

```
SELECT e.employee_name AS employee,
m.employee_name AS managerFROM
GFGemployees AS e JOIN GFGemployees
AS m ON e.manager_id = m.employee_id;
```

Output:

The resultant table after performing self join will be as follows:

employee	manager
Zaid	Raman
Rahul	Raman
Raman	Kamran
Farhan	Kamran

SQL | UPDATE with JOIN

SQL UPDATE JOIN could be used to update one table using another table and join condition.

Syntax –

UPDATE tablename
INNER JOIN tablename
ON tablename.columnname = tablename.columnname
SET tablenmae.columnnmae = tablenmae.columnname;
Use multiple tables in SQL UPDATE with JOIN statement.

Let us assume we have two tables – G1 and G2. To check the content in the table –
SELECT * FROM G1;
SELECT * FROM G2;

col1	col2	col3
1	11	FIRST
11	12	SECOND
21	13	THIRD
31	14	FOURTH

col1	col2	col3
1	21	TWO-ONE
11	22	TWO-TWO
21	23	TWO-THREE
31	24	TWO-FOUR

SQL | UPDATE with JOIN

Example –

We have table G2 which has two rows where Col 1 is 21 & 31 and we want to update the value from table G2 to table G1 for the rows where Col 1 is 21 and 31. Also, we want to update the values of Col 2 and Col 3 only.

```
UPDATE G1
SET col2 = Geeks2.col2,
col3 = Geeks2.col3
FROM G1
INNER JOIN G2 ON Geeks1.col1 = Geeks2.col1
WHERE Geeks1.col1 IN (21, 31);
```

Output –

```
(2 row(s) affected)
SELECT *  FROM G1;
SELECT *  FROM G2;
```

col1	col2	col3
1	11	FIRST
11	12	SECOND
21	23	TWO-THREE
31	24	TWO-FOUR

col1	col2	col3
1	21	TWO-ONE
11	22	TWO-TWO
21	23	TWO-THREE
31	24	TWO-FOUR

SQL DELETE JOIN

DELETE JOIN in SQL lets you delete rows of a table, based on conditions involving another table. We can use the DELETE statement with the JOIN operation to perform DELETE JOIN.

We use JOIN to combine data from multiple tables., to delete the same rows or related rows from the table at that time we use delete join.

In this article let us see how to delete multiple data using DELETE using JOIN by using MSSQL as a server.

Syntax

```
DELETE table1

FROM table1 JOIN table2

ON table1.attribute_name = table2.attribute_name

WHERE condition
```

Demo SQL Database

For this DELETE JOIN tutorial, we will use the following two tables in examples:

Table 1- Student

SQL DELETE JOIN

Table 1- Student

student_id	student_name	student_branch
1001	PRADEEP	E.C.E
1002	KIRAN	E.C.E
1003	PRANAV	E.C.E
2001	PADMA	C.S.E
2002	SRUTHI	C.S.E
2003	HARSITHA	C.S.E
3001	SAI	I.T
3002	HARSH	I.T
3003	HARSHINI	I.T

Table 2- Library books

lib_id	book_taken
1001	2
1002	3
1003	4
2001	2
3001	3

SQL DELETE JOIN

```
CREATE DATABASE db1;
USE db1
CREATE TABLE student (
student_id VARCHAR(8),
student_name VARCHAR(20),
student_branch VARCHAR(20)
)

CREATE TABLE library_books(
lib_id VARCHAR(20),
book_taken INT
)

INSERT INTO students
VALUES( '1001','PRADEEP','E.C.E'),
( '1002','KIRAN','E.C.E'),
( '1003','PRANAV','E.C.E'),
( '2001','PADMA','C.S.E'),
( '2002','SRUTHI','C.S.E'),
( '2003','HARSITHA','C.S.E'),
( '3001','SAI','I.T'),
( '3002','HARSH','I.T'),
( '3003','HARSHINI','I.T')

INSERT INTO library_books
VALUES( '1001',2),
( '1002',3),
( '1003',4),
( '2001',2),
( '3001',3)
```

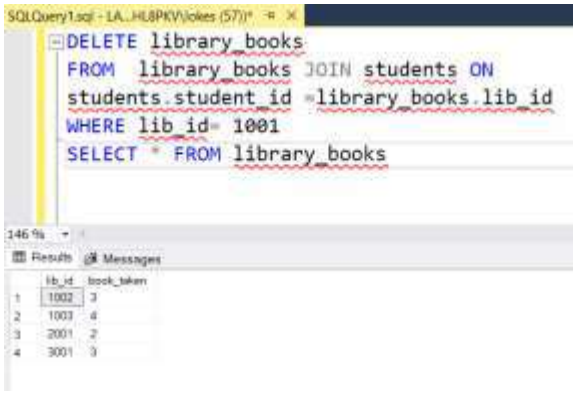
SQL DELETE JOIN

SQL DELETE JOIN Example

Query to delete library entry for id 1001 using JOIN

Query:

DELETE library_books
FROM library_books JOIN students ON
students.student_id =library_books.lib_id
WHERE lib_id= 1001
SELECT * FROM library_books



SQLQuery1.sql - LA...HL8PKV\lokes (57))*

```
DELETE library_books
FROM  library_books JOIN students ON
students.student_id =library_books.lib_id
WHERE lib_id= 1001
SELECT * FROM library_books
```

146 %

Results | Messages

	lib_id	book_taken
1	1002	3
2	1003	4
3	2001	2
4	3001	3

Key Takeaways about DELETE JOIN

DELETE JOIN allows to delete rows from a table based on condition involving another table.

We can use DELETE with JOIN to delete multiple rows from two or more tables.

Using WHERE clause with JOIN allows to specify condition for deleting rows.

If a record is deleted from a table, related records in other table will be deleted too

Recursive Join in SQL

A recursive query is a powerful feature that allows us to query hierarchical data which are used in relational databases. They are a compound operation that helps in accumulating records every time until a repetition makes no change to the result.

Recursive queries are useful to build a hierarchy tree, best in working with hierarchical data such as org charts for the bill of materials traverse graphs or to generate arbitrary row sets. This involves joining a set with itself an arbitrary number of times. A recursive query is usually defined by the anchor part and the recursive part.

Recursive joins are sometimes also called “fixed-point joins”. They are used to obtain the parent-child data. In SQL Recursive joins are implemented with recursive common table expressions. Recursive common table expression (CTEs) is a way to reference a query over and over again.

Now we understand the Recursive Join in SQL by using an example.

Step 1: First we create a database of employees, Where Common Table Expression of the company for its Employee Id, Employee name, Employee age.

Recursive Join in SQL

Query:

```
CREATE TABLE employees (
 id serial,
 name varchar(20),
  age int
);
```

Step 2: In this step insert values into an employee table.

Query:

```
INSERT INTO employees VALUES (1, 'Ankit', 32);
INSERT INTO employees VALUES (2, 'Ayush', 31);
INSERT INTO employees VALUES (3, 'Piyush', 42);
INSERT INTO employees VALUES (4, 'Ramesh', 31);
INSERT INTO employees VALUES (5, 'Rohan', 29);
INSERT INTO employees VALUES (6, 'Harry', 28);
INSERT INTO employees VALUES (7, 'Rohit', 32);
INSERT INTO employees VALUES (8, 'Gogi', 32);
INSERT INTO employees VALUES (9, 'Tapu', 33);
INSERT INTO employees VALUES (10, 'Sonu', 40);
```

Recursive Join in SQL

Step 3: A statement that gives all the reports that roll up into a certain organization within the company. A CTE is defined using a WITH statement, followed by a table expression definition. The AS command is used to rename a column or table with an alias. A recursive CTE must contain a UNION statement and be recursive.

Query:

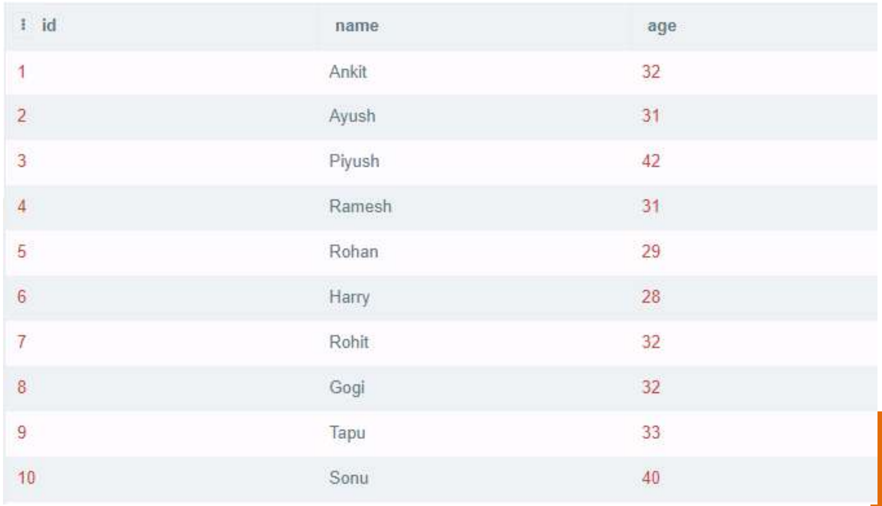
```
WITH RECURSIVE managertree AS (
 SELECT id, name, age
 FROM employees
 WHERE id = 1
 UNION ALL
 SELECT e.id, e.name, e.age
 FROM employees e
 INNER JOIN managertree mtree ON mtree.id = e.id
)
```

Step 4: To check the recursive join data we use the following query.

Query:

```
SELECT * FROM managertree;
```

Output:



id	name	age
1	Ankit	32
2	Ayush	31
3	Piyush	42
4	Ramesh	31
5	Rohan	29
6	Harry	28
7	Rohit	32
8	Gogi	32
9	Tapu	33
10	Sonu	40

References

1. https://www.geeksforgeeks.org/sql-left-join/?ref=lbp
2. https://www.w3schools.com/sql/default.asp
3. https://www.freecodecamp.org/news/sql-and-databases-full-course/