MENTORNESS ARTICLE

TASK 1

## MIP-DA-04 BATCH

# UNDERSTANDING SQL JOINS

By SUVODEEP CHAKRABORTY

## INTRODUCTION

Let's start with an overview of what a database is. A database is a collection of different tables storing different types of information. The SQL JOIN is a command clause that combines records from two or more tables in a database. It is a means of combining data in fields from two tables by using values common to each table. The JOIN clause is used when retrieving data from related tables in a database. The SQL JOIN clause is more complex than a simple query that retrieves data from a single table because it retrieves data from multiple tables.

## DIFFERENT TYPES OF SQL JOINS

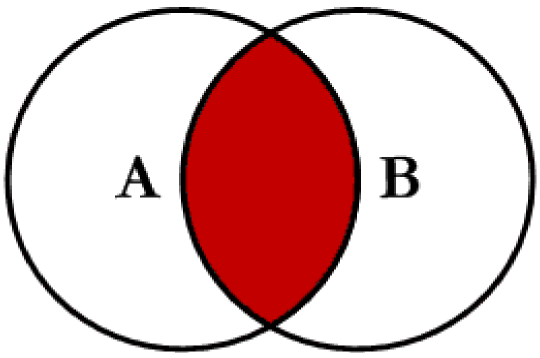
In this section we will discuss about several types of SQL Joins –

1. Inner Join
2. Outer Join
    - Left Join
    - Right Join
    - Full Join
3. Cross Join
4. Semi-join and Anti-join

## Inner Join:

Inner JOINs combine two tables based on a shared key. For example, if you had a table with a column called "user id" and each user id was unique to a user, you could join that table to another table with a "user id" column to find the information associated with each user. This example shows how to use an Inner JOIN clause to join two tables:

**Syntax:** *SELECT * FROM table1 INNER JOIN table2 ON table1.id = table2.id;*



**Example:**

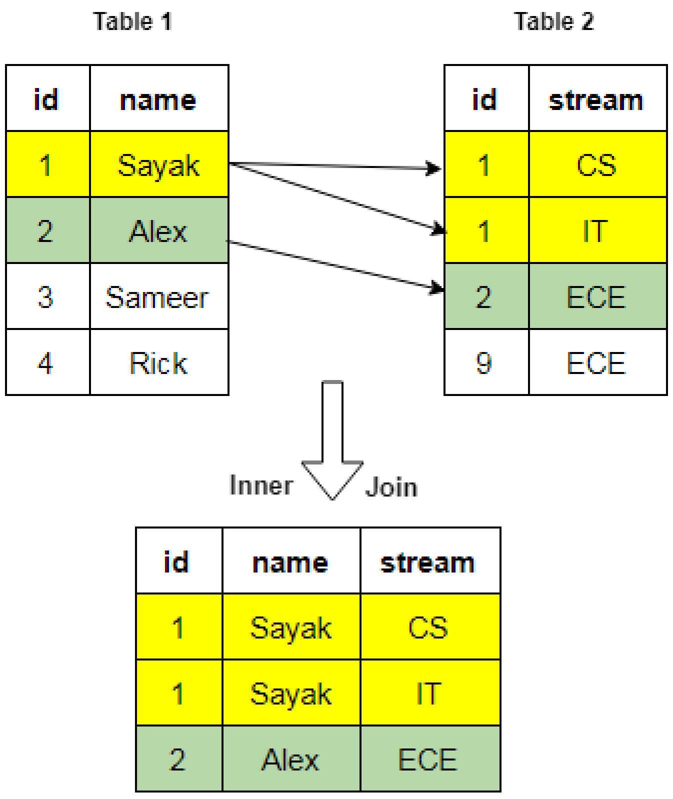


Table 1

| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |
| 3 | Sameer |
| 4 | Rick |

Table 2

| id | stream |
|---|---|
| 1 | CS |
| 1 | IT |
| 2 | ECE |
| 9 | ECE |

| id | name | stream |
|---|---|---|
| 1 | Sayak | CS |
| 1 | Sayak | IT |
| 2 | Alex | ECE |

In the above example, the column under consideration is the id column. INNER JOIN will ignore rest of the columns for which the values are not common in both the tables.

**Syntax:**

```
SELECT s1.id, s1.name, s2.stream
FROM student_name AS s1
INNER JOIN student_stream AS s2
ON s1.id = s2.id;
```

## Outer Join:

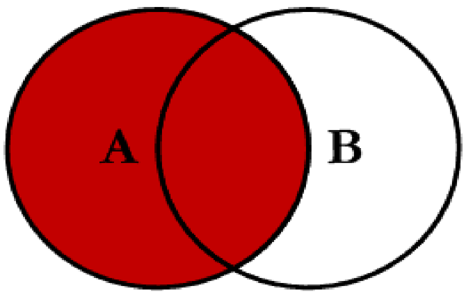
Whereas INNER JOIN zips together two tables that share a common index column, The Outer Joins returns rows that satisfy the operation specified. Outer Join can further be divided into three types –

## Left Join:

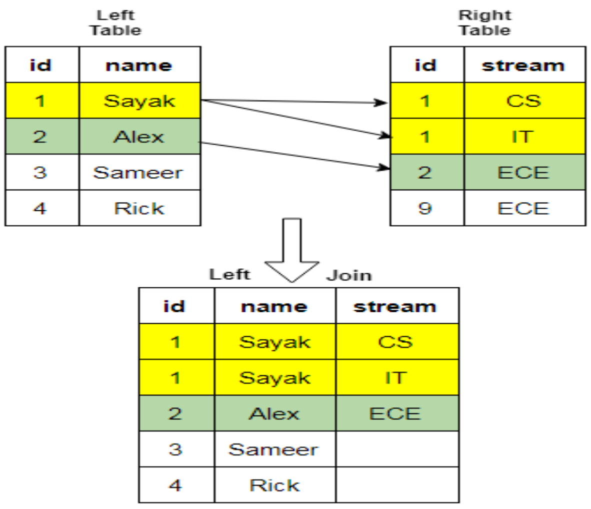
Left JOINs return all rows from the first table and only the rows in the second table that match. This example shows how to use a Left Outer JOIN clause to join two tables:

### Syntax:

*SELECT * FROM table1 LEFT OUTER JOIN table2  ON table1.id = table2.user_id;*



## Example:


Left Table
Right Table
Left Join

Left Table

| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |
| 3 | Sameer |
| 4 | Rick |

Right Table

| id | stream |
|---|---|
| 1 | CS |
| 1 | IT |
| 2 | ECE |
| 9 | ECE |

Left Join

| id | name | stream |
|---|---|---|
| 1 | Sayak | CS |
| 1 | Sayak | IT |
| 2 | Alex | ECE |
| 3 | Sameer | |
| 4 | Rick | |

Unlike INNER JOIN, LEFT JOIN fetches you the records from the left table for which there was not any matching entry in the right table. This tells you that Sameer and Rick have not enrolled in any streams.

**Syntax:**

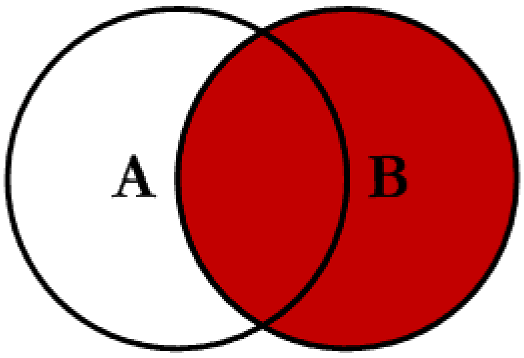
```
SELECT s1.id, s1.name, s2.stream
FROM student_name AS s1
LEFT JOIN student_stream AS s2
ON s1.id = s2.id;
```

## Right Join:

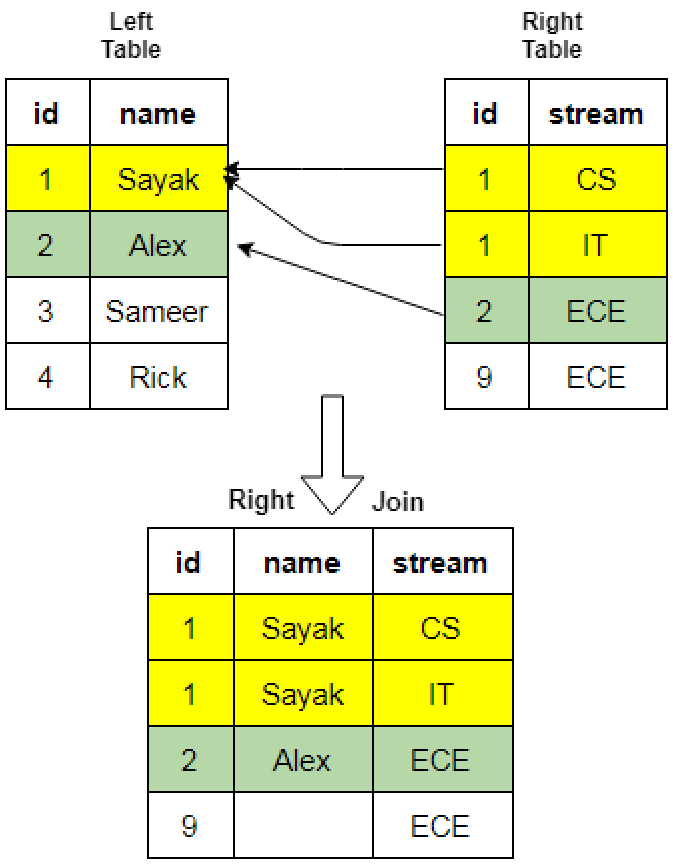
Right JOINs are logically the opposite of Left JOINs—they return all rows from the second table, and only the rows in the first table that match. This example shows how to use a Right Outer JOIN clause to join two tables:

### Syntax:

*SELECT * FROM table1  RIGHT OUTER JOIN table2 ON table1.id = table2.user_id;*



## Example:



| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |
| 3 | Sameer |
| 4 | Rick |

| id | stream |
|---|---|
| 1 | CS |
| 1 | IT |
| 2 | ECE |
| 9 | ECE |

| id | name | stream |
|---|---|---|
| 1 | Sayak | CS |
| 1 | Sayak | IT |
| 2 | Alex | ECE |
| 9 |  | ECE |

RIGHT JOIN can help you in finding the streams for which no student has enrolled.

## Syntax:

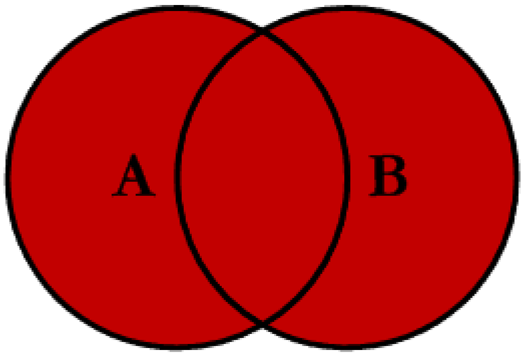
```
SELECT s1.id, s1.name, s2.stream
FROM student_name AS s1
RIGHT JOIN student_stream AS s2
ON s1.id = s2.id;
```

## Full Join:

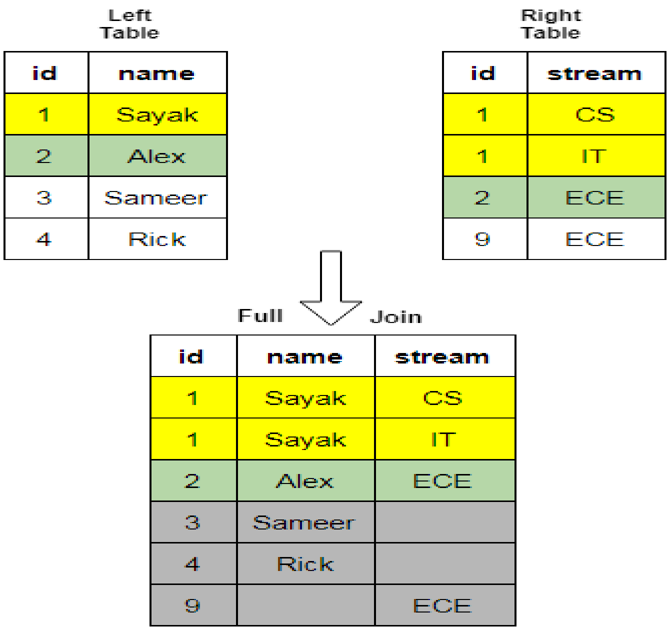
Full JOINs combine both left and right joins by returning all rows from both tables, as long as there is at least one match between them. This example shows how to use a Full Outer JOIN clause to join two tables:

### Syntax:

*SELECT * FROM table1  FULL OUTER JOIN table2 ON table1.id = table2.user_id;*



### Example:



Left Table

| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |
| 3 | Sameer |
| 4 | Rick |

Right Table

| id | stream |
|---|---|
| 1 | CS |
| 1 | IT |
| 2 | ECE |
| 9 | ECE |

Full Join

| id | name | stream |
|---|---|---|
| 1 | Sayak | CS |
| 1 | Sayak | IT |
| 2 | Alex | ECE |
| 3 | Sameer | |
| 4 | Rick | |
| 9 | | ECE |

FULL JOIN lets you combine both LEFT JOIN and RIGHT JOIN into a single compilation.

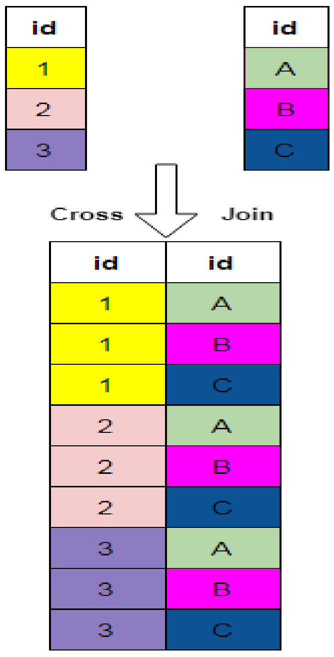
### Syntax:

```
SELECT s1.id, s1.name, s2.stream
FROM student_name AS s1
FULL JOIN student_stream AS s2
ON s1.id = s2.id;
```

## Cross Join:

CROSS JOIN is essentially the cartesian product between two elements expressed using SQL. Suppose, you need to have all the combinations possible in between two tables or even in a single table. You will need CROSS JOIN for doing this.

### Example:



| id |
|---|
| 1 |
| 2 |
| 3 |

| id |
|---|
| A |
| B |
| C |

| id | id |
|---|---|
| 1 | A |
| 1 | B |
| 1 | C |
| 2 | A |
| 2 | B |
| 2 | C |
| 3 | A |
| 3 | B |
| 3 | C |

In order to have all possible combinations between the id columns of the student_name and student_stream tables, you can execute the following query -

**Syntax:**

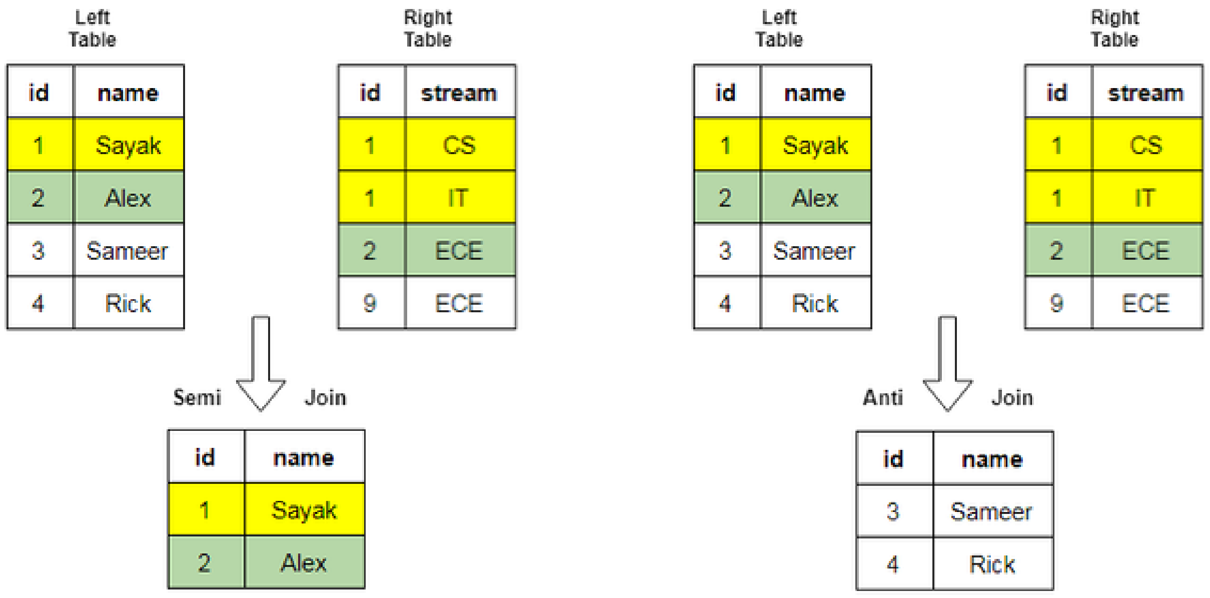
```
SELECT s1.id, s2.id
FROM student_name AS s1
CROSS JOIN student_stream AS s2;
```

## Semi-join and Anti-join:

Semi-join queries are generally executed in the form of subqueries where rows are picked up from the first table with respect to a condition that is matched in the second table. Let's assume the left table is the first table, and the right table is the second table.

Anti-join queries are the exact opposite. In Anti-join rows are picked up from the first table with respect to a condition that is not matched in the second table.

**Example:**



Left Table

| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |
| 3 | Sameer |
| 4 | Rick |

Right Table

| id | stream |
|---|---|
| 1 | CS |
| 1 | IT |
| 2 | ECE |
| 9 | ECE |

Semi Join

| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |

Left Table

| id | name |
|---|---|
| 1 | Sayak |
| 2 | Alex |
| 3 | Sameer |
| 4 | Rick |

Right Table

| id | stream |
|---|---|
| 1 | CS |
| 1 | IT |
| 2 | ECE |
| 9 | ECE |

Anti Join

| id | name |
|---|---|
| 3 | Sameer |
| 4 | Rick |

**Syntax:**

The query for realizing the Semi-join would be –

```
select id, name
from student_name
where id IN
(select id from student_stream where stream
IN ('CS', 'IT', 'ECE'));
```

Similarly, the query that realizes Anti-join, in this case, would be –

```
select id, name
from student_name
where id NOT IN
(select id from student_stream where stream
IN ('CS', 'IT', 'ECE'));
```

## CONCLUSION

Mastering SQL joins is an essential skill for any database user. It helps in enhancing query performance, extracting relevant data, and delivering insights. In this article, we covered the most commonly used queries for extracting data between two tables. Depending on the specific conditions, the queries can be modified further to filter out irrelevant data according to your needs. The usage and understanding of joins will help to easily create complex queries and assist in data retrieval and data analytics. Incorrect usage of joins could also make your query slower. Always check the query performance using EXPLAIN to make optimizations and subsequent modifications in the query.