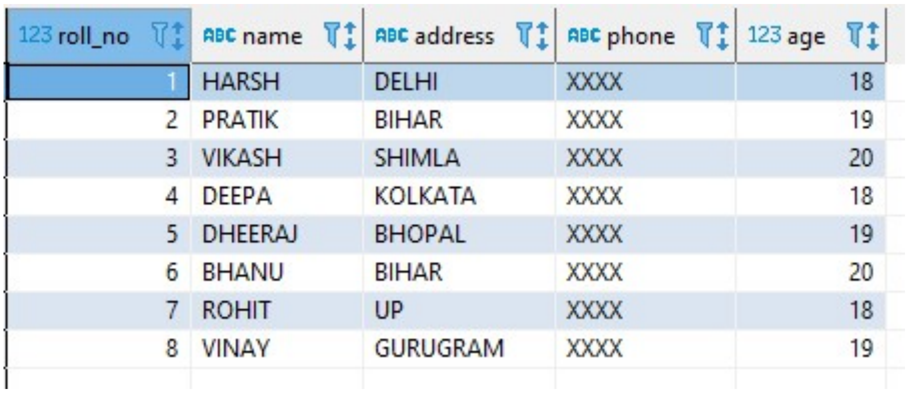
# Joins and Unions

## SQL Joins

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

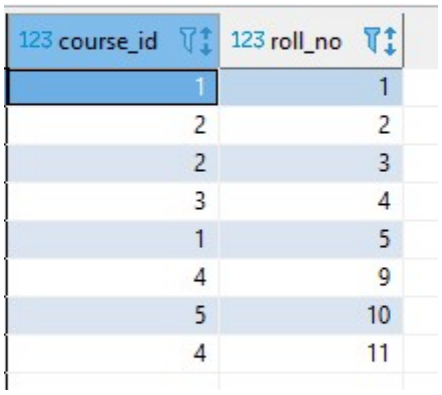
Consider the following two tables,

(a) **Student** table is as follows:



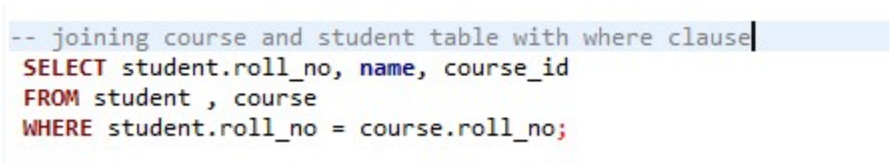
| roll_no | name | address | phone | age |
|---|---|---|---|---|
| 1 | HARSH | DELHI | XXXX | 18 |
| 2 | PRATIK | BIHAR | XXXX | 19 |
| 3 | VIKASH | SHIMLA | XXXX | 20 |
| 4 | DEEPA | KOLKATA | XXXX | 18 |
| 5 | DHEERAJ | BHOPAL | XXXX | 19 |
| 6 | BHANU | BIHAR | XXXX | 20 |
| 7 | ROHIT | UP | XXXX | 18 |
| 8 | VINAY | GURUGRAM | XXXX | 19 |

(b) Another table is **course** as follows:



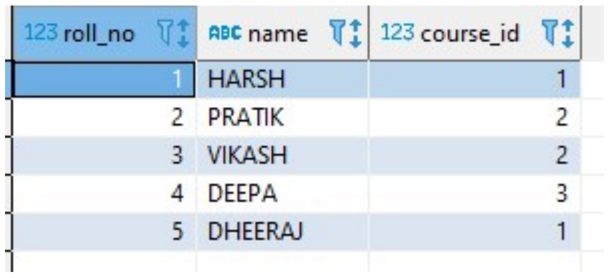
| course_id | roll_no |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 2 | 3 |
| 3 | 4 |
| 1 | 5 |
| 4 | 9 |
| 5 | 10 |
| 4 | 11 |

Now, let us join these two tables in our SELECT statement as follows:



```
-- joining course and student table with where clause
 SELECT student.roll_no, name, course_id
 FROM student , course
 WHERE student.roll_no = course.roll_no;
```

This would produce the following result:



| roll_no | name | course_id |
|---|---|---|
| 1 | HARSH | 1 |
| 2 | PRATIK | 2 |
| 3 | VIKASH | 2 |
| 4 | DEEPA | 3 |
| 5 | DHEERAJ | 1 |

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal symbol.

## Types of Joins in SQL:

There are different types of joins available in SQL. They are as follows:

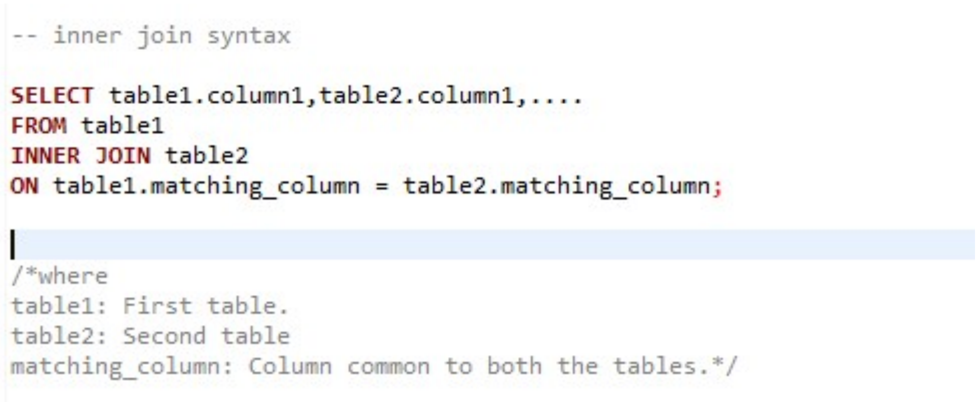
- **INNER JOIN**: returns rows when there is a match in both tables.
- **LEFT JOIN**: returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN**: returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN**: returns rows when there is a match in one of the tables.
- **SELF JOIN**: is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- **CARTESIAN JOIN**: returns the Cartesian product of the sets of records from the two or more joined tables.

The simplest Join is INNER JOIN.

### INNER JOIN:

The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. 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 same.

**Syntax**:



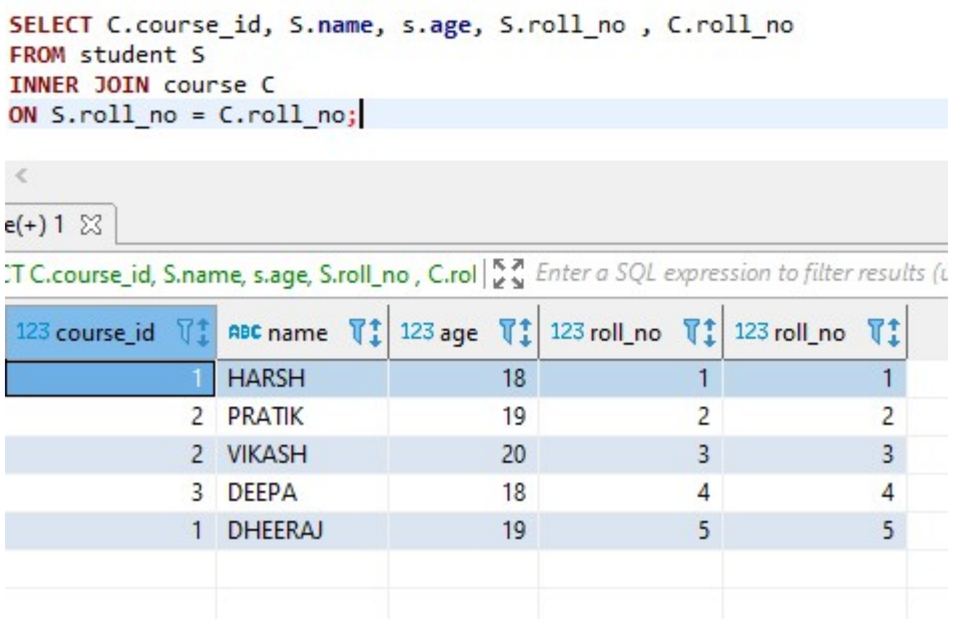
```
-- inner join syntax

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

/*where
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.*

e.g. Let's apply inner join on Student and course table.



```
SELECT C.course_id, S.name, s.age, S.roll_no , C.roll_no
FROM student S
INNER JOIN course C
ON S.roll_no = C.roll_no;
```

e(+) 1
T C.course_id, S.name, s.age, S.roll_no , C.rol
Enter a SQL expression to filter results (

| course_id | name | age | roll_no | roll_no |
|---|---|---|---|---|
| 1 | HARSH | 18 | 1 | 1 |
| 2 | PRATIK | 19 | 2 | 2 |
| 2 | VIKASH | 20 | 3 | 3 |
| 3 | DEEPA | 18 | 4 | 4 |
| 1 | DHEERAJ | 19 | 5 | 5 |

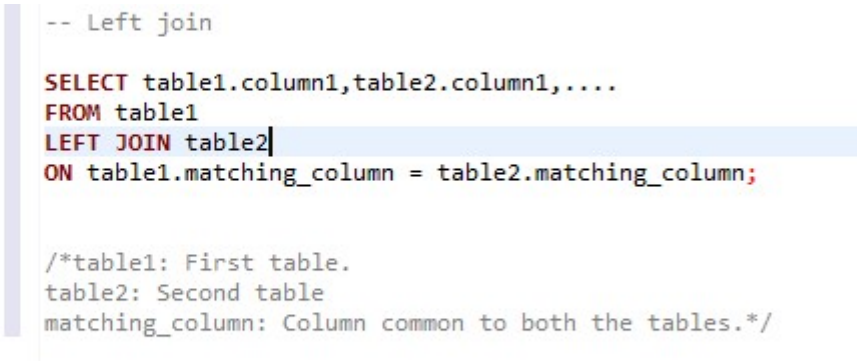
Above query shows names and age of students enrolled in different courses along with their roll numbers from both the tables.

## LEFT JOIN

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table. This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

**Syntax**:

The basic syntax of LEFT JOIN is as follows:



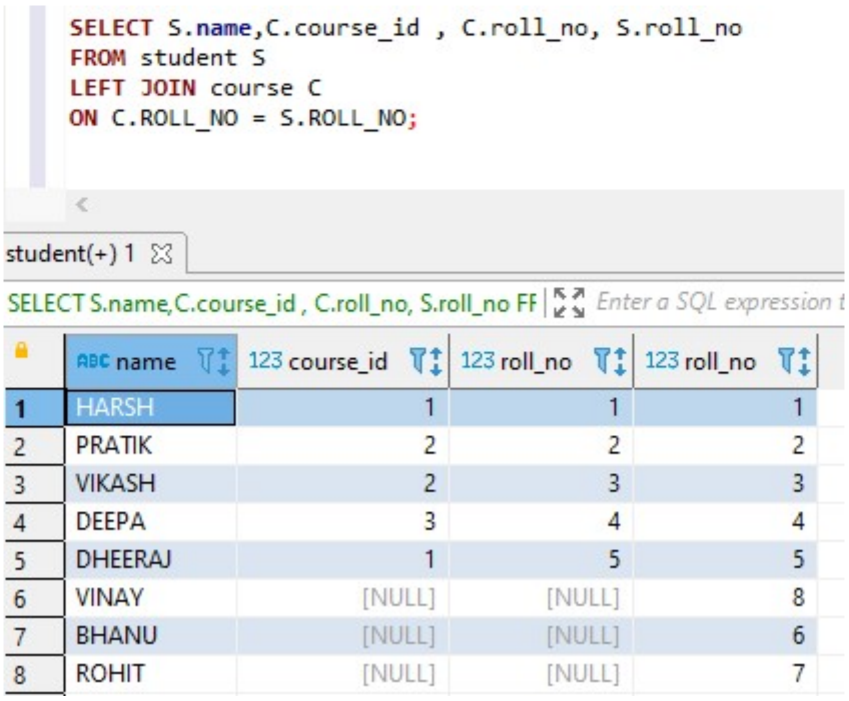
```
-- Left join

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


/*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 same*

Now, let us join Students and course tables using LEFT JOIN as follows:



```
SELECT S.name,C.course_id , C.roll_no, S.roll_no
FROM student S
LEFT JOIN course C
ON C.ROLL_NO = S.ROLL_NO;
```

```
SELECT S.name,C.course_id , C.roll_no, S.roll_no FF
```

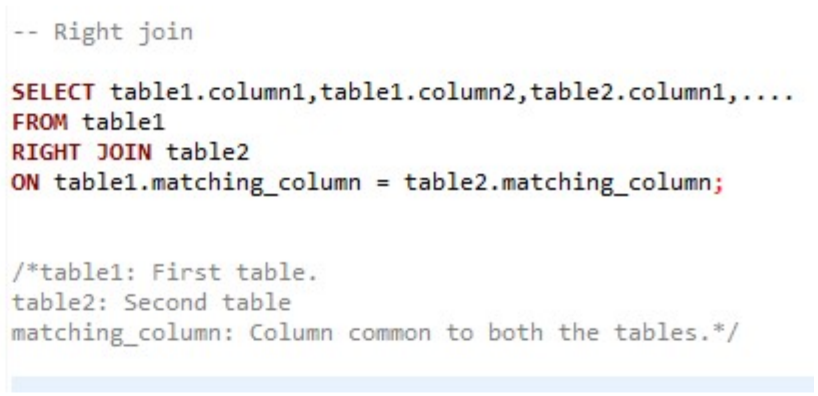
|   | name | course_id | roll_no | roll_no |
|---|---|---|---|---|
| 1 | HARSH | 1 | 1 | 1 |
| 2 | PRATIK | 2 | 2 | 2 |
| 3 | VIKASH | 2 | 3 | 3 |
| 4 | DEEPA | 3 | 4 | 4 |
| 5 | DHEERAJ | 1 | 5 | 5 |
| 6 | VINAY | [NULL] | [NULL] | 8 |
| 7 | BHANU | [NULL] | [NULL] | 6 |
| 8 | ROHIT | [NULL] | [NULL] | 7 |

## RIGHT JOIN

The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the join will still return a row in the result, but with NULL in each column from left table. This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

**Syntax**:

The basic syntax of RIGHT JOIN is as follows:



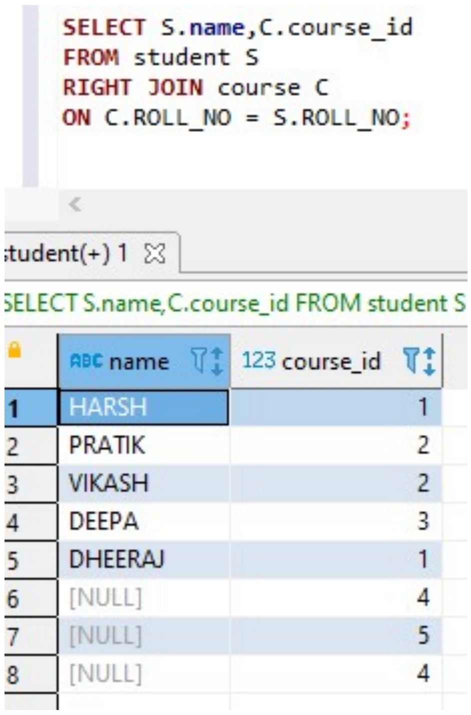
```
-- Right join

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


/*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 same.*

Now, let us join Students and course tables using RIGHT JOIN as follows:



```
SELECT S.name,C.course_id
FROM student S
RIGHT JOIN course C
ON C.ROLL_NO = S.ROLL_NO;
```

tudent(+) 1

SELECT S.name,C.course_id FROM student S

|   | name | course_id |
|---|---|---|
| 1 | HARSH | 1 |
| 2 | PRATIK | 2 |
| 3 | VIKASH | 2 |
| 4 | DEEPA | 3 |
| 5 | DHEERAJ | 1 |
| 6 | [NULL] | 4 |
| 7 | [NULL] | 5 |
| 8 | [NULL] | 4 |

## FULL JOIN

The SQL FULL JOIN combines the results of both left and right outer joins. The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

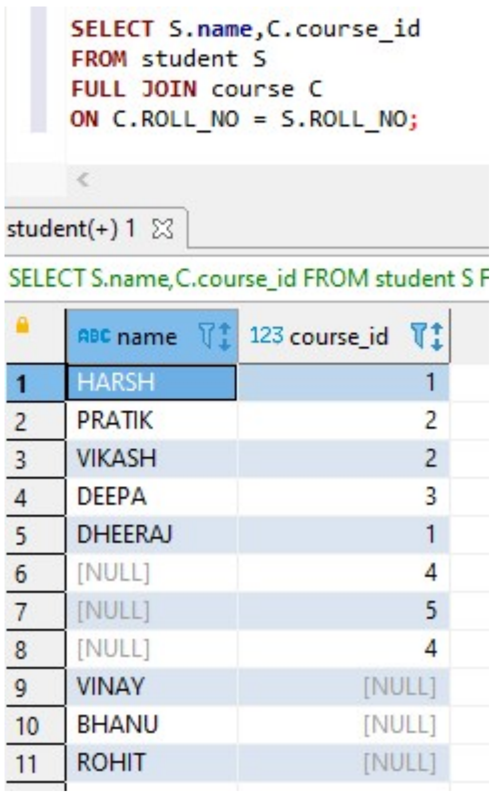
**Syntax:**

The basic syntax of FULL JOIN is as follows:

```
-- FULL JOIN
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_filed = table2.common_field;

/*where
table1: First table.
table2: Second table
common_filed: Column common to both the tables.*/
```

Now, let us join Students and course tables using RIGHT JOIN as follows:



```
SELECT S.name,C.course_id
FROM student S
FULL JOIN course C
ON C.ROLL_NO = S.ROLL_NO;
```

student(+) 1

SELECT S.name,C.course_id FROM student S F

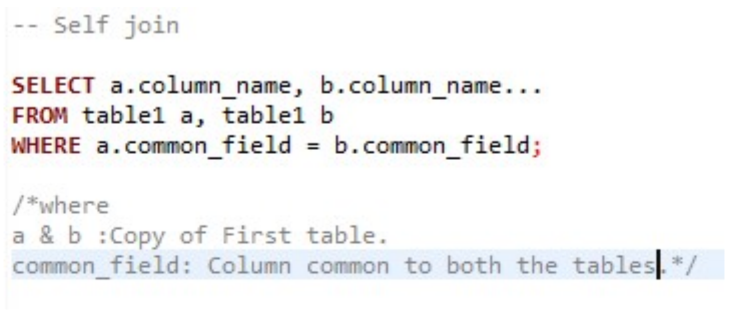
| | name | course_id |
|---|---|---|
| 1 | HARSH | 1 |
| 2 | PRATIK | 2 |
| 3 | VIKASH | 2 |
| 4 | DEEPA | 3 |
| 5 | DHEERAJ | 1 |
| 6 | [NULL] | 4 |
| 7 | [NULL] | 5 |
| 8 | [NULL] | 4 |
| 9 | VINAY | [NULL] |
| 10 | BHANU | [NULL] |
| 11 | ROHIT | [NULL] |

## SELF JOIN

The SQL SELF JOIN is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

**Syntax:**

The basic syntax of SELF JOIN is as follows:

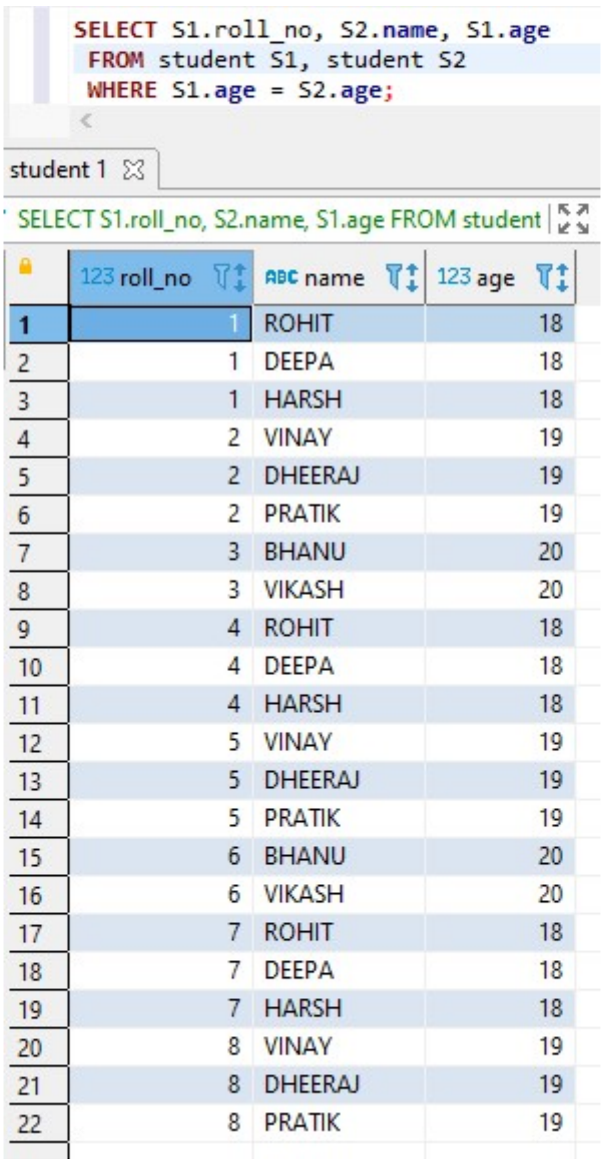


```
-- Self join

SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;

/*where
a & b :Copy of First table.
common_field: Column common to both the tables.*/
```

Let's apply self join on Student table.



```
SELECT S1.roll_no, S2.name, S1.age
 FROM student S1, student S2
 WHERE S1.age = S2.age;
```

student 1

SELECT S1.roll_no, S2.name, S1.age FROM student

| | roll_no | name | age |
|---|---|---|---|
| 1 | 1 | ROHIT | 18 |
| 2 | 1 | DEEPA | 18 |
| 3 | 1 | HARSH | 18 |
| 4 | 2 | VINAY | 19 |
| 5 | 2 | DHEERAJ | 19 |
| 6 | 2 | PRATIK | 19 |
| 7 | 3 | BHANU | 20 |
| 8 | 3 | VIKASH | 20 |
| 9 | 4 | ROHIT | 18 |
| 10 | 4 | DEEPA | 18 |
| 11 | 4 | HARSH | 18 |
| 12 | 5 | VINAY | 19 |
| 13 | 5 | DHEERAJ | 19 |
| 14 | 5 | PRATIK | 19 |
| 15 | 6 | BHANU | 20 |
| 16 | 6 | VIKASH | 20 |
| 17 | 7 | ROHIT | 18 |
| 18 | 7 | DEEPA | 18 |
| 19 | 7 | HARSH | 18 |
| 20 | 8 | VINAY | 19 |
| 21 | 8 | DHEERAJ | 19 |
| 22 | 8 | PRATIK | 19 |

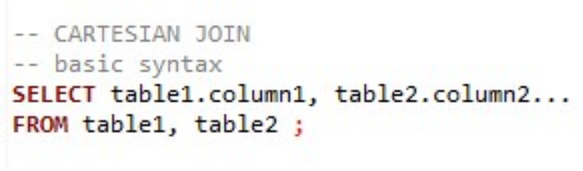
*Note - It is useful when we want to correlate pairs of rows from the same table.*

## CARTESIAN JOIN

The CARTESIAN JOIN or CROSS JOIN returns the cartesian product of the sets of records from the two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to True or where the join condition is absent from the statement.

**Syntax**:

The basic syntax of INNER JOIN is as follows:



```
-- CARTESIAN JOIN
-- basic syntax
SELECT table1.column1, table2.column2...
FROM table1, table2 ;
```

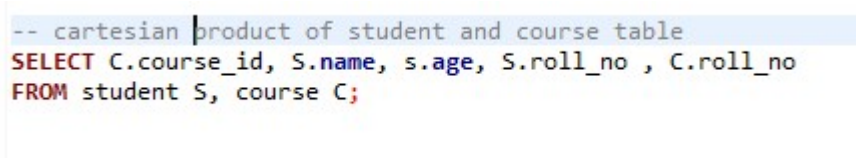
Suppose table1 has m records And

table2 has n records.

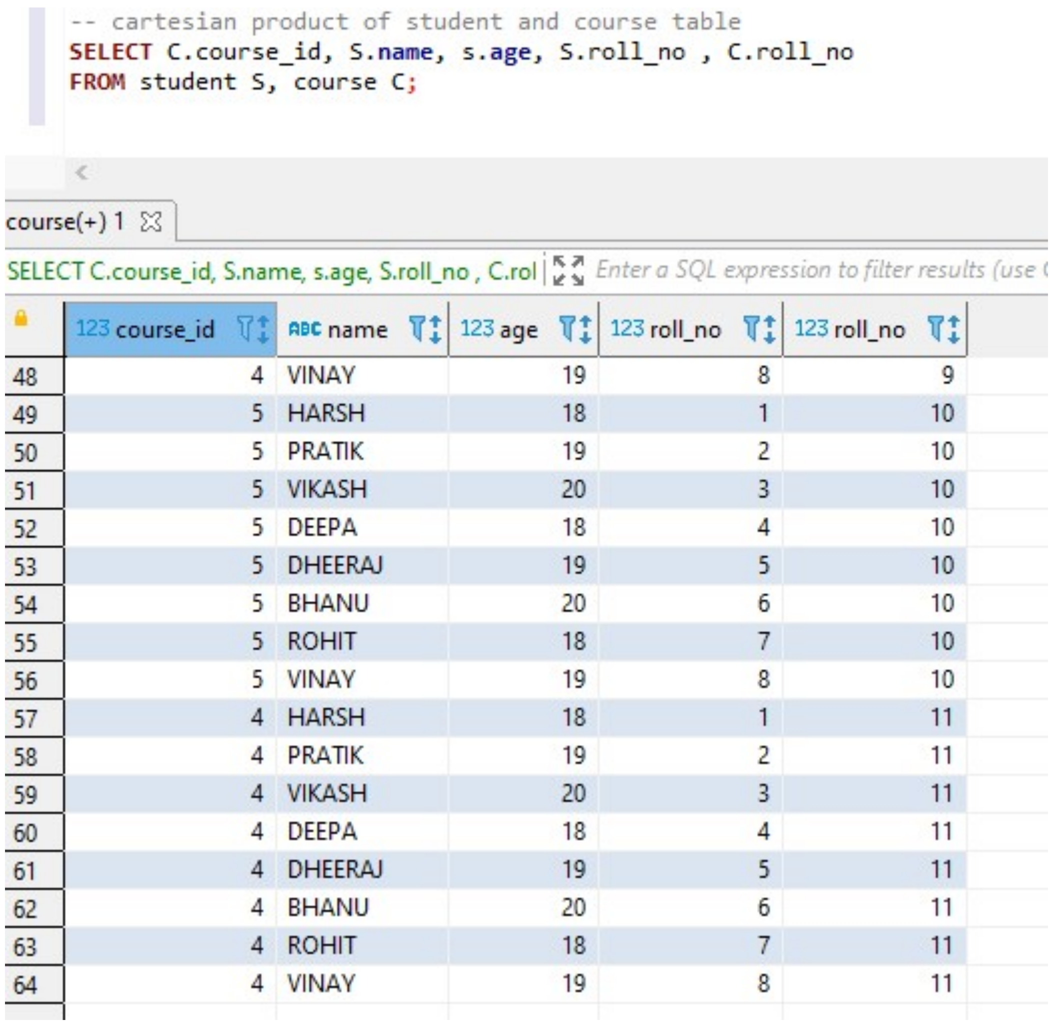
Then,

On applying CARTESIAN JOIN or CROSS join on two tables we get a total of  (m x n) records on the new table.

e.g. Apply Cartesian join on student and course table:



```
-- cartesian product of student and course table
SELECT C.course_id, S.name, s.age, S.roll_no , C.roll_no
FROM student S, course C;
```



```
-- cartesian product of student and course table
SELECT C.course_id, S.name, s.age, S.roll_no , C.roll_no
FROM student S, course C;
```

course(+) 1
SELECT C.course_id, S.name, s.age, S.roll_no , C.rol
Enter a SQL expression to filter results (use

| | course_id | name | age | roll_no | roll_no |
|---|---|---|---|---|---|
| 48 | 4 | VINAY | 19 | 8 | 9 |
| 49 | 5 | HARSH | 18 | 1 | 10 |
| 50 | 5 | PRATIK | 19 | 2 | 10 |
| 51 | 5 | VIKASH | 20 | 3 | 10 |
| 52 | 5 | DEEPA | 18 | 4 | 10 |
| 53 | 5 | DHEERAJ | 19 | 5 | 10 |
| 54 | 5 | BHANU | 20 | 6 | 10 |
| 55 | 5 | ROHIT | 18 | 7 | 10 |
| 56 | 5 | VINAY | 19 | 8 | 10 |
| 57 | 4 | HARSH | 18 | 1 | 11 |
| 58 | 4 | PRATIK | 19 | 2 | 11 |
| 59 | 4 | VIKASH | 20 | 3 | 11 |
| 60 | 4 | DEEPA | 18 | 4 | 11 |
| 61 | 4 | DHEERAJ | 19 | 5 | 11 |
| 62 | 4 | BHANU | 20 | 6 | 11 |
| 63 | 4 | ROHIT | 18 | 7 | 11 |
| 64 | 4 | VINAY | 19 | 8 | 11 |

Here, we have got a total of 64 records which is 8*8.

## SQL Unions

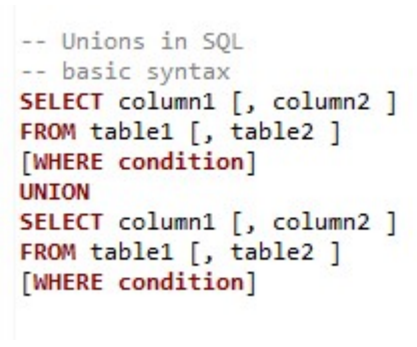
### The SQL UNION

The SQL union clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

To use UNION, each SELECT must have the same number of columns selected, the same number of column expressions, the same data type, and have them in the same order, but they do not have to be the same length.

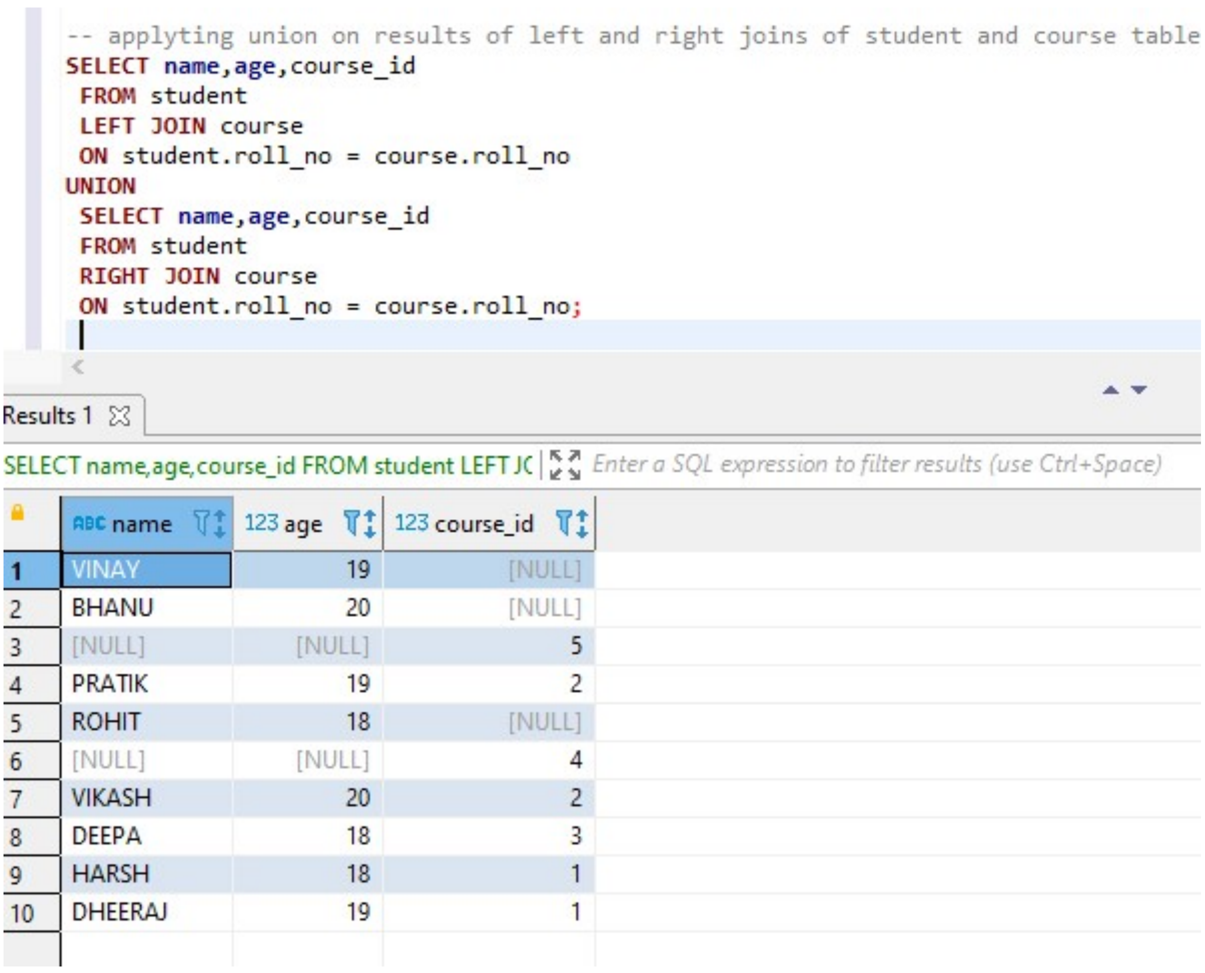
Syntax:

The basic syntax of UNION is as follows:



```
-- Unions in SQL
-- basic syntax
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

On applying union on results of left and right joins of student and course tables we get following result:



```
-- applyting union on results of left and right joins of student and course table
SELECT name,age,course_id
 FROM student
 LEFT JOIN course
 ON student.roll_no = course.roll_no
UNION
 SELECT name,age,course_id
 FROM student
 RIGHT JOIN course
 ON student.roll_no = course.roll_no;
```

Results 1

SELECT name,age,course_id FROM student LEFT JC | Enter a SQL expression to filter results (use Ctrl+Space)

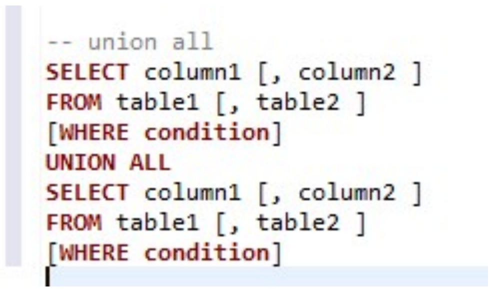
| | name | age | course_id |
|---|---|---|---|
| 1 | VINAY | 19 | [NULL] |
| 2 | BHANU | 20 | [NULL] |
| 3 | [NULL] | [NULL] | 5 |
| 4 | PRATIK | 19 | 2 |
| 5 | ROHIT | 18 | [NULL] |
| 6 | [NULL] | [NULL] | 4 |
| 7 | VIKASH | 20 | 2 |
| 8 | DEEPA | 18 | 3 |
| 9 | HARSH | 18 | 1 |
| 10 | DHEERAJ | 19 | 1 |

## The UNION ALL Clause:

The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows. The same rules that apply to UNION apply to the UNION ALL operator. The only difference is duplicates are allowed in this.

**Syntax**:

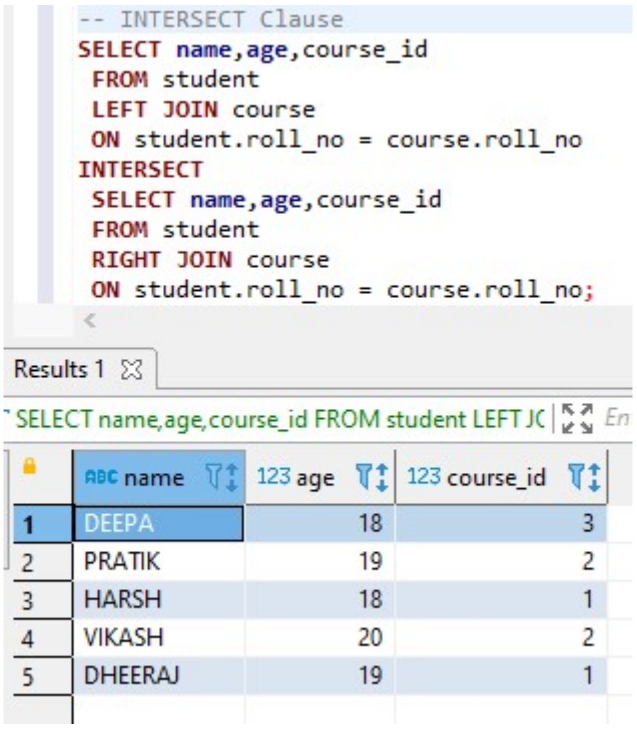
The basic syntax of UNION ALL is as follows:



```
-- union all
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION ALL
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Apart from UNION & UNION ALL there are two other clauses similar to these. They are:

- SQL INTERSECT Clause: It is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement.
- SQL EXCEPT Clause : combines two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement.

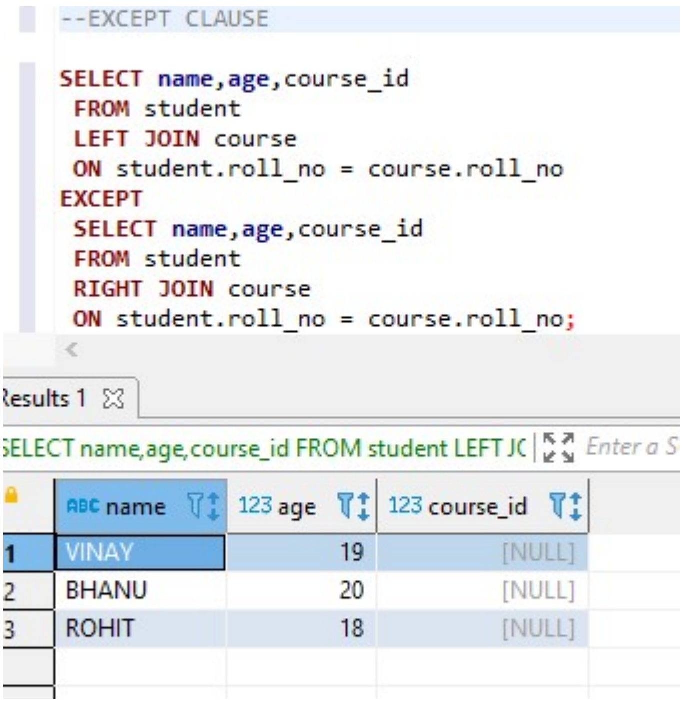


```
-- INTERSECT Clause
SELECT name,age,course_id
 FROM student
 LEFT JOIN course
 ON student.roll_no = course.roll_no
INTERSECT
 SELECT name,age,course_id
 FROM student
 RIGHT JOIN course
 ON student.roll_no = course.roll_no;
```

Results 1

SELECT name,age,course_id FROM student LEFT JC

| | name | age | course_id |
|---|---|---|---|
| 1 | DEEPA | 18 | 3 |
| 2 | PRATIK | 19 | 2 |
| 3 | HARSH | 18 | 1 |
| 4 | VIKASH | 20 | 2 |
| 5 | DHEERAJ | 19 | 1 |



```
--EXCEPT CLAUSE

SELECT name,age,course_id
 FROM student
 LEFT JOIN course
 ON student.roll_no = course.roll_no
EXCEPT
 SELECT name,age,course_id
 FROM student
 RIGHT JOIN course
 ON student.roll_no = course.roll_no;
```

Results 1

SELECT name,age,course_id FROM student LEFT JC

| | name | age | course_id |
|---|---|---|---|
| 1 | VINAY | 19 | [NULL] |
| 2 | BHANU | 20 | [NULL] |
| 3 | ROHIT | 18 | [NULL] |