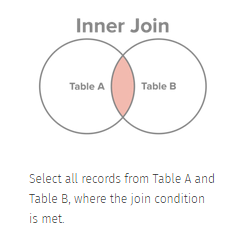
**Joins:**

To combine data from 2 tables.

JOINS have better performance compared to sub queries

**Inner Join:**

Select all rows from both tables get matched.



**Table 1: Student**

[](https://www.geeksforgeeks.org/wp-content/uploads/table1-3.png)

**Table 2: Student Course**

[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/table5.png)

**Syntax:**

**SELECT <table1.column1>,<table1.column2>, <table2.column1>,....**

**FROM <table1>**

**INNER JOIN <table2>**

**ON <table1.matching\_column> = <table2.matching\_column>;**

**Ex:**

SELECT StudentCourse.COURSE\_ID, Student.NAME, Student.AGE FROM Student

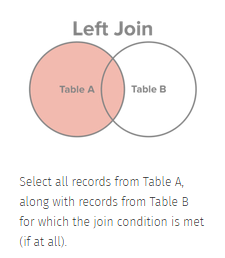
INNER JOIN StudentCourse

ON Student.ROLL\_NO = StudentCourse.ROLL\_NO;

**Output**:  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/table22.png)

**LEFT JOIN:**

**Select records from left table that matching with right table**



**Syntax:**

**SELECT <table1.column1>,<table1.column2>, <table2.column1>,....**

**FROM <table1>**

**LEFT JOIN <table2>**

**ON <table1.matching\_column> = <table2.matching\_column>;**

**Ex:**

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

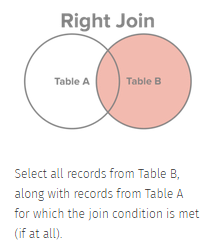
LEFT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**Output**:  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/table31.png)

**Right JOIN:**

**Select records from right table that matching with left table**



**Syntax:**

**SELECT <table1.column1>,<table1.column2>, <table2.column1>,....**

**FROM <table1>**

**RIGHT JOIN <table2>**

**ON <table1.matching\_column> = <table2.matching\_column>;**

**Ex:**

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

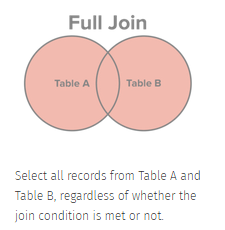
RIGHT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**Output**:  
  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/table6.png)

**FULL JOIN:**

**Matched and unmatched record in both the tables**

****

**Syntax:**

**SELECT <table1.column1>,<table1.column2>, <table2.column1>,....**

**FROM <table1>**

**FULL JOIN <table2>**

**ON <table1.matching\_column> = <table2.matching\_column>;**

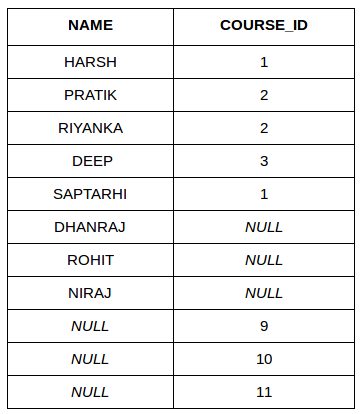
**Ex:**

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

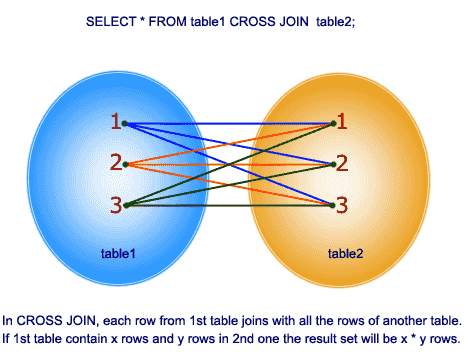
FULL JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**Output:**  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/table7.png)

**Cross join:**

Number of rows in the first table multiplied by the number of rows in the second table

****

**Syntax:**

**SELECT <table1.column1>,<table1.column2>, <table2.column1>,....**

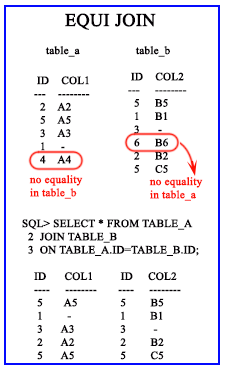
**FROM <table1>**

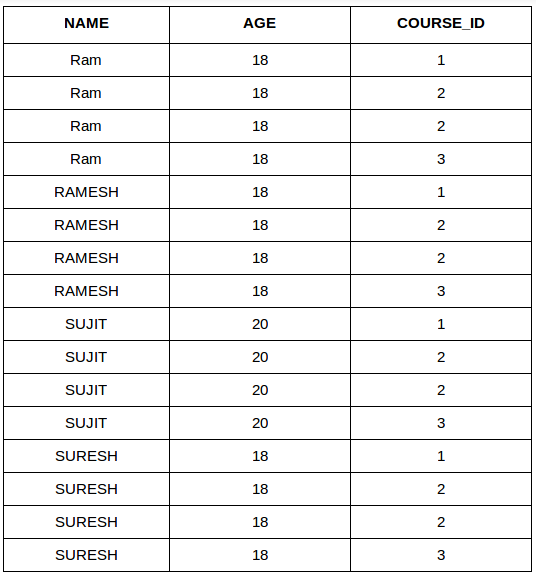
**CROSS JOIN <table2>**

**ON <table1.matching\_column> = <table2.matching\_column>;**

**Equi join:**

An equijoin returns only the rows that have equivalent values for the specified columns.

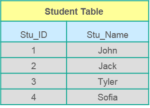
****

**Output**:  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/table_final.png)

**Basic Queries:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Name** | **Description** | **Syntax** |
| 1 | CREATE Database |  | CREATE DATABASE DatabaseName; |
| 2 | Create table |  | CREATE TABLE table\_name(  column1 datatype,  column2 datatype,  column3 datatype,  .....  columnN datatype,  PRIMARY KEY( one or more columns ) ); |
| 3 | Insert query |  | INSERT INTO TABLE\_NAME (column1, column2, column3,...columnN)  VALUES (value1, value2, value3,...valueN); |
| 4 | Select Query |  | SELECT <column name> FROM table\_name; |
| 5 | Update |  | UPDATE table\_name SET column1 = value1, column2 = value2...., columnN = valueN WHERE [condition]; |
| 6 | Delete | used to delete a row in a table,can rollback | DELETE FROM table\_name WHERE [condition]; |
| 7 | Distinct |  | SELECT DISTINCT <columnname> FROM table\_name WHERE [condition] |
| 8 | Order By | Sort the column name by asc or desc | SELECT <column-name> FROM table\_name  [WHERE condition]  [ORDER BY <columnname>] [ASC | DESC]; |
| 9 | Group by |  | SELECT <column-name> FROM table\_name WHERE [ conditions ] GROUP BY <columnname> ORDER BY <columnname> |
| 10 | union | is used to combine two separate select statements Not allowed duplicates | SELECT <column-name> FROM <table1> UNION SELECT <column-name> FROM <table2>; |
| 11 | Union All | is used to combine two separate select statements  allowed duplicates | SELECT <column-name> FROM <table1> UNION ALL SELECT <column-name> FROM <table2>; |
| 12 | In | The IN operator is a shorthand for multiple OR conditions | SELECT <column-names> FROM table\_name WHERE <column-name> IN (value1, value2, ...); |
| 13 | Between | selects values within a given range. The values can be numbers, text, or dates | SELECT <column-names> FROM <table\_name> WHERE <column-name > BETWEEN value1 AND value2; |
| 14 | Aliases | used to give a table, or a column in a table, a temporary name | SELECT <column\_name> AS <alias\_name> FROM <table\_name>; |
| 15 | AND | displays a record if all the conditions separated by AND are TRUE. | SELECT <columnNAME> FROM <table\_name> WHERE condition1 AND condition2 AND condition3 ...; |
| 16 | OR | displays a record if any of the conditions separated by OR is TRUE. | SELECT <columnNAME> FROM <table\_name> WHERE condition1 OR condition2 OR condition3 ...; |
| 17 | COUNT() | returns the number of rows that matches a specified criteria | SELECT COUNT(column\_name) FROM table\_name WHERE condition; |
| 18 | AVG() | returns the average value of a numeric column | SELECT AVG(column\_name) FROM table\_name WHERE condition; |
| 19 | SUM() | returns the total sum of a numeric column. | SELECT SUM(column\_name) FROM table\_name WHERE condition; |
| 20 | Like | is used in a WHERE clause to search for a specified pattern in a column  % - The percent sign represents zero, one, or multiple characters  \_ - The underscore represents a single character | WHERE CustomerName LIKE 'a%'==== Finds any values that start with "a" |
| 21 | WHERE CustomerName LIKE '%a'====Finds any values that end with "a" |
| 22 | WHERE CustomerName LIKE '%or%'====Finds any values that have "or" in any position |
| 23 | WHERE CustomerName LIKE '\_r%'=======Finds any values that have "r" in the second position |
| 24 | WHERE CustomerName LIKE 'a\_%\_%'=====Finds any values that start with "a" and are at least 3 characters in length |
| 25 | WHERE ContactName LIKE 'a%o'====Finds any values that start with "a" and ends with "o" |
| 26 | Exists | used to test for the existence of any record in a subquer | SELECT column\_name(s) FROM table\_name WHERE EXISTS (SELECT column\_name FROM table\_name WHERE condition); |
| 27 | Select into | copies data from one table into a new table. creates a new table located in the default filegroup | Select <column name> INTO <table 1> from <table2> where cnd='' |
| 28 | min | returns the minimum value for a column. | Select Min<columnname> from<tablename> |
| 29 | Max | returns the maximum value for a column | Select Max<columnname> from<tablename> |
| 30 | Top | used to fetch limited number of rows from a database. | Select Top value<columnname> from <tablename> |
| 31 | LIMIT | Used to fetch the records with limit based | Select <columnname> from <tablename> LIMIT<number> |
| 32 | Except | Fetch the except value from 1st table  Doesnot alllow duplicates | SELECT column\_name(s) FROM table\_name Except (SELECT column\_name FROM table\_name WHERE condition); |
| 33 | Except All | Fetch the exceptall value from 1st table  alllowed duplicates | SELECT column\_name(s) FROM table\_name ExceptALL (SELECT column\_name FROM table\_name WHERE condition); |
| 34 | Intersect | result contains all the rows which are common to both the SELECT statements. | SELECT column\_name(s) FROM table\_name INTERSECT (SELECT column\_name FROM table\_name WHERE condition); |
| 35 | Having | [used with the GROUP BY clause to specify a filter condition for a group or an aggregate](https://www.tutorialrepublic.com/sql-tutorial/sql-group-by-clause.php) | Select <columnanme> from <tablename> groupby <columnname> Having <columnname> |
| 36 | Drop table | drops all tables in the database and deletes the database | Drop <tablename> |
| 37 | Truncate | used to remove all records from a table,cant rollback | TRUNCATE TABLE table\_name; |

## **What is a Primary key?**

* APrimary keyis a column (or collection of columns) or a set of columns that uniquely identifies each row in the table.
* Uniquely identifies a single row in the table
* Null values not allowed

Example- In the Student table, Stu\_ID is the primary key.

## ****What is a Foreign key?****

* Foreign key maintains referential integrity by enforcing a link between the data in two tables.
* The foreign key in the child table references the primary key in the parent table.
* The foreign key constraint prevents actions that would destroy links between the child and parent tables.

**Difference between TRUNCATE, DELETE and DROP commands?**  
**Ans.** DELETE removes some or all rows from a table based on the condition. It can be rolled back.

* TRUNCATE removes ALL rows from a table by de-allocating the memory pages. The operation cannot be rolled back
* DROP command removes a table from the database completely.