# What Are Joins In SQL

## ****Introduction to Joins In SQL****

Whenever we want to fetch data from two or more database tables based on common field (column) we use Joins.

**Definition:  
Technically Joins are SQL operations which help us in retrieving data from two or more tables that share a common field.**

## Types of Joins

There several Types of Joins such as:

* Inner Join
* Outer Join
* Cross Join
* Self-Join

Outer Join can further be divided into 3 more categories:

* Right Outer Join
* Left Outer Join
* Full Outer Join

Apart from types of joins we also have two join conditions. These are:

* Equi Joins
* Non-Equi Joins

**Equi Joins:** This type of Join looks for common records in two tables on the basis of equality condition and then combines them. Equi-Join is constructed with the help of equality operator (=) where the values of Primary key and Foreign key are compared. Hence the common or matching records from the two tables are presented in the result.

**Non Equi Joins:** Here the equality operator is not used. Instead operators such as <, >, BETWEEN etc. are employed. Therefore Non Equi-Join is the opposite of Equi-Join and uses joining conditions excluding equal operator. For example, in a non Equi-Join condition you can use !=, <=,>=,<,> or BETWEEN etc. operators can be used for joining. For implementation you can read about Inner-Join.

**Outer Join:** Outer joins are the SQL operations which definitely return all the rows from Source table no matter whether there is a matching join condition hit or not. At the same time it returns only those rows of the target table that fulfills the matching join condition otherwise it just shows ‘null’ in the rows.

**Natural Joins**

**Definition of Natural Joins:**  
A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns of the two tables that are being joined. Common columns are columns that have the same name in both the tables.  
A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join. However the INNER join is the default one.

***Source table:*** The table which comes after FROM clause in the select statement.

***Target table:*** All the tables that come after JOIN clause in the query.

When the join query is executed then the oracle starts matching the data from the source table to the target table. If there is a hit for a matching source table data in the target table then the value is returned.

## Examples of Natural Joins in Oracle Database

### Scenario 1: When there is only one identical name column between source and target tables.

Let’s use Departments table and locations table for the demonstration.

Department table has 4 columns – DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID, LOCATION\_ID

And Locations table has 6 columns – LOCATION\_ID, STREET ADDRESS, POSTAL CODE, CITY, STATE PROVINCE AND COUNTRY\_ID

Upon minutely observing you will notice that location id is a common column between our source and target tables. It is a foreign key and thus has the same data type and column width. Let’s write the query for natural join.

Say you want to find the City for all your Departments.

***SELECT   department\_name, city   FROM   departments   NATURAL JOIN   locations;***

In this query we are selecting department name from departments table and city from locations table. You can select other columns too.

Departments is our source table as it’s coming right after FROM keyword and locations is our Target table as it’s coming after JOIN clause.

The best part of using Natural join is that you do not need to specify the join column because column with the same name in source and target tables are automatically associated with each other.

On executing this query oracle returns all the rows from department column of departments table and from city column of locations table, where location id of departments table is equal to the location id of location table.

### Scenario 2: What if when our source and target tables have more than one identical name column.

To demonstrate this scenario we will use employees and departments table. These two tables share two common columns which are the department id and the manager id.

Here if we put Natural join on these two tables then oracle engine will use these two common columns to return the result.

Say we want to see the name of an employee and the name of the department in which he or she works. For that we will first select a name from the employee table which is our source table and the department’s name from departments table.

***SELECT   first\_name, department\_name   FROM   employees   NATURAL JOIN   departments;***

As I have already mentioned that when we use natural join, there is no need to specify join columns explicitly. All the columns which are common in both the source and the target table get associated automatically.

The queries in which all the common columns of source and target table get associated automatically are known as Pure Natural Join.

This means that in this particular query, oracle has automatically associated our common columns which are department id and manager id respectively.

To help you understand more clearly let’s write the same query but this time using ON clause.

***SELECT   first\_name,department\_name   FROM   employees  JOIN    departments   ON   (employees.manager\_id = departments.manager\_id   AND   employees.department\_id = departments. department\_id);***

Here is a slight change in the natural join syntax as we are using ON clause this time. So instead of writing natural join we will only write JOIN.

Both these queries produce the same results as both are the same queries just written in a different way.

In this query we emphasize natural join using ON clause. As we are using ON clause thus it becomes mandatory to specify the columns over which we want to perform the join.

### Scenario 3: Natural join with USING clause

Here Once again we will use our natural join query.

***SELECT   first\_name, department\_name   FROM   employees    NATURAL JOIN   departments;***

As we can see here in this query that when we have more than one common column then the oracle engine uses all of them to produce the result.

Now, here in this query we are using employees and department tables which have 2 common columns department id and manager id.

### What if you want to use only department id in your natural join instead of both manager id and department id?

In this situation we use natural join with USING clause.

Say we want to select all the first names and department names from the source table employees and target table departments from where we have equal values of manager ids in both the employees table and the departments table.

Let’s write the query

***SELECT   first\_name,department\_name   FROM   employees   JOIN   departments   USING(manager\_id);***

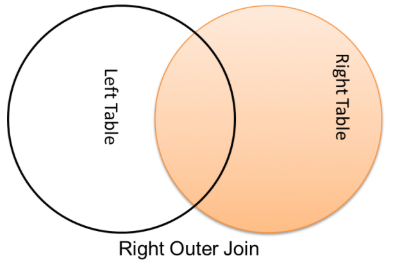
# What Is Right Outer Join In SQL

As the name suggests Right Outer Join is a form of Outer Join that returns each and every record from the source table and returns only those values from the target table that fulfil the Join condition.

\*Note: The Source table is the one situated on the right side of the Right Outer Join Clause whereas The Target table is the one on the left side of this clause.

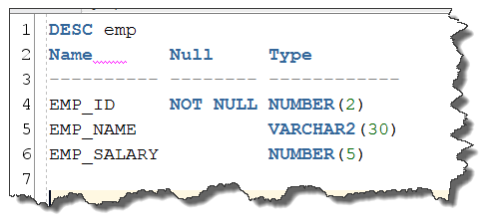
Syntax :

***SELECT   
    columns   
  FROM   
  target\_table RIGHT OUTER JOIN source\_table   
  ON(source\_table.column = target\_table.column)  
  WHERE condition  
  ORDER BY column\_names;***

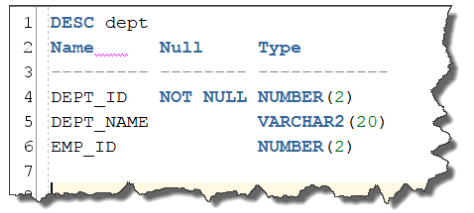


Examples:  
To demonstrate the working of Right Outer Join, I have created two tables by the name of emp and dept and have also inserted some data in them.

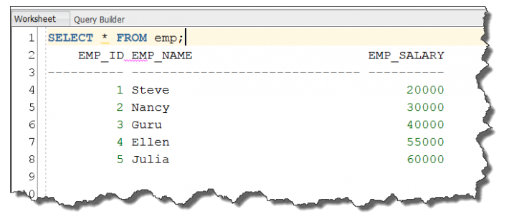
Structure of emp table



Structure of dept table



Data in emp table



Have you noticed that in dept table, IT and marketing department don’t have any employees.

Query 1: Right Outer Join with ‘ON’ clause

Now when should we use ON Join Condition in SQL Joins you ask?  
When columns which are participating in ON join condition have Different name and Same Data type or

When columns which are participating in ON join condition have SAME NAME and SAME Data-type.

Let’s write a very simple Right outer join query.

***SELECT   
  emp\_name, dept\_name  
FROM  dept  RIGHT OUTER JOIN  emp  
ON (emp.EMP\_ID = dept.EMP\_ID);***

In the above query we are selecting emp\_name from emp table and dept\_name from dept table. Emp table is the source table as it’s on the right hand side of join clause thus automatically making dept table as the target table.  
Column emp\_id is a primary key in emp table and foreign key in dept table thus column emp\_id is the column which is establishing a relationship in between these two tables. As column emp\_id is common in both tables thus it’s a best fit for Join condition (ON clause).

If you execute this query you will get all the rows of emp\_name column of emp table as emp is the source table and only those rows of dept\_name column of dept table which satisfy the join condition (condition in ON clause).

Query 2: Right Outer Join with WHERE clause

You can use WHERE clause with any type of JOIN and limit or Filter the result. In case of JOINS, WHERE clause always comes after Join Condition (ON or USING). Say you want to see employees name and department name whose salary is less than 50,000.

***SELECT   emp\_name, dept\_name   
FROM  dept  RIGHT OUTER JOIN  emp   
ON  (emp.EMP\_ID = dept.EMP\_ID) WHERE  emp\_salary < 50000;***

Query 3: Right Outer Join with USING clause

Using clause can be used in the place of ON clause when

We are performing Equi Join.

We want to put join on that column of source and target table which are common i.e. have same name and data type.

Most of the joins you’ll perform will be equijoins, and if you always use the same name as the primary key for your foreign keys, then:

***SELECT  emp\_name,dept\_name  FROM  dept  RIGHT OUTER JOIN  emp  USING (emp\_id);***

# Full Outer Joins In SQL

Before moving ahead just want to say that in order to better understand the concept of full outer Join please read my last two SQL tutorial article on [right](http://localhost/RebellionRider/right-outer-join/) and [left outer join](http://localhost/RebellionRider/left-outer-join/).

Here once again we will be using the same tables which we have used so far in this Outer Join series. These are the emp and the dept tables.

Let’s have a quick look of the structure and the data of these two tables.

Now data of dept table

**SELECT \* FROM dept;**

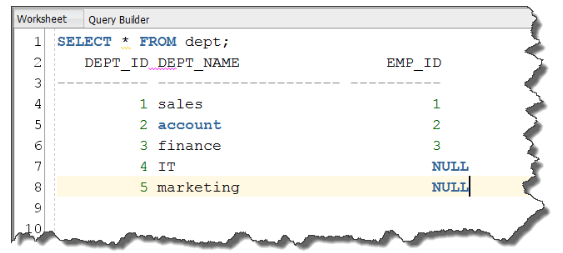


Table Dept also has 5 columns. Column one hold dept id where column two has dept name and column three emp id which is the foreign key has only 3 records corresponding to department name sales account and finance.   
We can interpret this data easily. For example the employee with employee id 1 is named Steve in our emp table and works in department of Sales. Similarly employee with employee id 2 is Nancy who works in Accounts department and employee with employee id 3 is Guru who works in finance. Also we have not assigned any employee id for department of IT and Marketing which signifies that no one works in these departments.

Let’s jump over to full outer join.   
**Full outer join is kind of a combination of both right outer join and left outer join because it returns all the rows from the left as well as the right side table.**  
Let’s have a look at FULL OUTER JOIN syntax.

Syntax

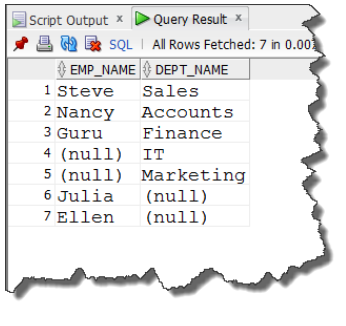
**SELECT column names FROM   table1 FULL OUTER JOIN / FULL JOIN table 2  ON(expression) or USING(column\_name)   WHERE(expression) ORDER BY column names;**

Syntax is pretty similar to our left or right join as you can see. We have our SELECT statement where you can specify the name of the columns from both the participating tables followed by FROM clause. And our JOIN clause which is full outer join. Here you can either write full outer join or just outer join since both are permissible and perform the same task. And then we have our Join condition ON and USING followed by WHERE and ORDER BY clause.

## ****Query 1 : Full Outer Join With ON clause****

**SELECT   emp\_name, dept\_name   FROM   emp   FULL OUTER JOIN   dept   ON  (emp.emp\_id  =  dept.emp\_id);**

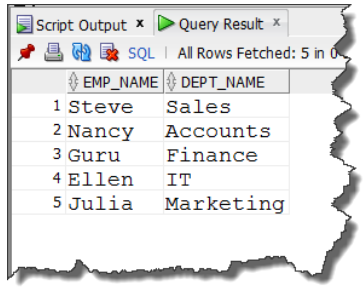
Here in this query we are selecting emp name column from emp table and dept name column from dept table. In our full outer join clause we have emp table on left side and dept table on right side and then we have Our ON clause where we are comparing the values of emp id columns from the both the tables.  
On executing this query the result will be.



If you will observe minutely then you can see that the result till row 5 is similar to that of right outer join, as all the records from right side table is here and only those records from left side table which satisfy the join condition are here in the result. Followed by all the remaining records from the tables thus the last row 6 and 7 contains the remaining emp names.  
Here in this ON join clause we used columns which have same name and data type. Now let’s use columns which have different name and same data type for example column  
dept id.

**SELECT   emp\_name, dept\_name   FROM   emp   FULL OUTER JOIN   dept   ON  (emp.emp\_id  =  dept.dept\_id);**

and the result of this query is



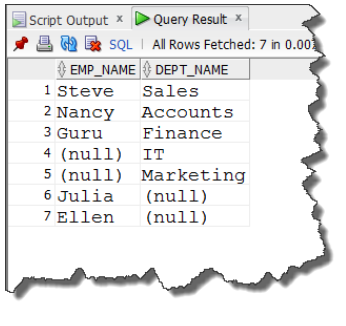
## ****Query 2 : Full Outer Join With USING clause****

We use USING join condition when:

The column in join condition share the same name and same data type and are compared only using = comparison operator and no other comparison operator such as greater than, less than etc.   
As you can see, in the join condition (ON clause) of Query 1 we have used emp id column of both the tables. This column shares the same name and same data type hence we can easily replace this ON clause with USING clause. Let’s do it.

**SELECT   emp\_name, dept\_name   FROM   emp   FULL OUTER JOIN   dept   USING  (emp\_id);**

The result of this query is

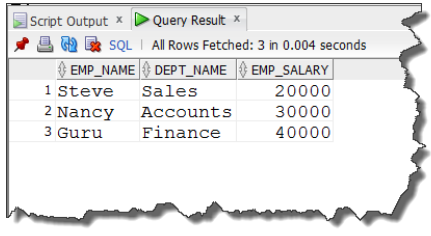


## ****Query 3 : Full Outer Join With WHERE clause****

You can also limit the result by using WHERE clause. Say you want to see the name of only those employees and their departments who have a salary of less than 50000  
For that we can modify our query 2 and add a where clause to it.

**SELECT   emp\_name, dept\_name, emp\_salary   FROM   emp   FULL OUTER JOIN   dept   USING  (emp\_id) WHERE emp.emp\_salary < 50000;**

On executing this query you will get only those employees and their departments who have salary less than 50,000

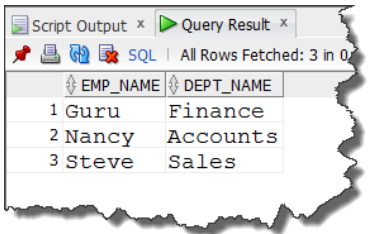


## ****Query 4 : Full Outer Join With ORDER BY clause****

Similarly you can sort the result using ORDER BY clause. Say you want to sort the result of query 3 in ascending order according to the employee name (emp\_name column) For that just add the ORDER BY clause followed by the column name which is emp\_name in our case.

**SELECT   emp\_name, dept\_name   FROM   emp   FULL OUTER JOIN   dept   USING  (emp\_id)  WHERE emp.emp\_salary < 50000  ORDER BY emp\_name;**

If you execute this query then by default the result will be sorted in ascending order.



If you write DESC right after the column, your result will be sorted in descending order. For Example

***SELECT   emp\_name, dept\_name   FROM   emp   FULL OUTER JOIN   dept   USING  (emp\_id)    
WHERE emp.emp\_salary < 50000   ORDER BY emp\_name DESC;***

# Inner Joins In SQL

And in this tutorial we will cover the concepts of INNER Join.

**Definition  
  Inner Join is the join which returns all those rows from both the participating tables that satisfy the Join condition or for that matter expression of ON/USING clause.**

**Syntax  SELECT column names FROM   table1 INNER JOIN / JOIN table 2  ON(expression) or USING(column\_name)  WHERE(expression) ORDER BY column names;**

In the first line of our syntax we have our SELECT statement where you specify all those columns from both the participating tables (table1 and table2) whose data you want to fetch in your result set. The SELECT statement is followed by FROM keyword.

In the second line of our syntax we have our JOIN clause which is INNER join as obvious. You can either write INNER JOIN or simple JOIN as both are permissible and perform the same task. On both sides of our Join clause we have our tables which are TABLE 1 and TABLE 2.

Followed by our join clause we have our Join condition. Basically we have two types of join conditions that we can use one at a time but never together. These two join conditions are ON and USING. Usage of each join condition depends on certain conditional parameters.

And at the end of the syntax we have our WHERE and ORDER BY clause.

***\*Note here if you are using ORDER BY clause then it must be the last statement of your query.***

Let’s do some practical exercises. I’ll be using the same table which we have been using in all our JOIN tutorials – emp and dept.  
(for tables you can refer to [Table for SQL Joins](http://localhost/RebellionRider/sql-join-table-structure/) tutorial)

## ****QUERY 1: INNER JOIN with ON clause****

**SELECT emp\_name,dept\_name FROM emp INNER JOIN dept ON(emp.EMP\_ID = dept.EMP\_ID);**

Here in this query we are selecting emp name from emp table and dept name from dept table while in JOIN condition which in this case is the ON clause. We are comparing emp\_id column of both the tables.  
As here in this query we are using ON join condition thus we can use any column as expression of ON clause as long as columns share same data types. The name of the column doesn’t matter here.  
For example we have dept\_id column in dept table which shares same data type as of emp\_id column of emp table which is a NUMBER thus here in our ON clause we can use this dept\_id column too.

**SELECT emp\_name,dept\_name FROM emp INNER JOIN dept ON(emp.EMP\_ID = dept.DEPT\_ID);**

## ****QUERY 2: INNER JOIN with USING clause.****

We already know that we can use USING clause when the columns in join condition share same name and same data type and are compared only using equal to ( = ) comparison operator and no other comparison operator such as greater than, less than etc.

**SELECT emp\_name,dept\_name FROM emp INNER JOIN dept USING(EMP\_ID);**

Here we used emp\_id column in USING clause because it’s common in both the tables (foreign key relation in emp and dept table) and shares same name as well as data type. We can easily compare its value using equal to operator in both the tables as we did in our first query where we used emp\_id column of both the tables in the expression of ON clause.

## ****QUERY 3: INNER JOIN with WHERE clause.****

We use WHERE clause to limit the result of a Query, similarly you can use WHERE clause here with Inner join to do the same. Say you want to see the name and departments of only those employees who have a salary of less than 50000. For that you just have to add the WHERE Clause right after the JOIN condition in the query.

For example

**SELECT emp\_name,dept\_name FROM emp INNER JOIN dept USING(EMP\_ID) WHERE emp.emp\_salary < 50000;**

## ****QUERY 4: INNER JOIN with ORDER BY clause.****

Feel free to use ORDER BY clause if you want to sort the result returned by your query in Ascending or Descending order.  
**NOTE here ORDER BY clause must be the last statement of a QUERY.**ORDER BY clause by default sorts the result in ascending order. But if you want to arrange the result in Descending order then you have to specify it by using DESCENDING or DESC keyword with the ORDER BY clause.

For example   
Sort the result in ascending order according to emp\_name column

**SELECT emp\_name,dept\_name FROM emp INNER JOIN dept USING(EMP\_ID) ORDER BY emp\_name;**

Sort the result in descending order according to emp\_name column.

**SELECT emp\_name,dept\_name FROM emp INNER JOIN dept USING(EMP\_ID) ORDER BY emp\_name DESC;**

# Cross Join In SQL

**Definition  
  Cross Join produces [Cartesian product](http://en.wikipedia.org/wiki/Cartesian_product) of the tables which are participating in the Join queries that’s why it’s also known by the name of Cartesian Join.**

Generally we use CROSS JOIN when there is no relationship between participating tables.

Syntax

***SELECT  column names  FROM  table1  CROSS JOIN  table 2  WHERE (expression)  ORDER BY  column names;***

Or

***SELECT  column names  FROM  table1,  table 2  WHERE (expression)  ORDER BY  column names***

In the first line we have our Select statement where you can specify the list of columns from both the participating tables.

In the second line we have our tables and the CROSS JOIN clause. Here you can write either Cross Join or just put a comma in between the names of both tables.

Also with cross join we do not have any ON or USING join condition. But you may, however, specify a WHERE and ORDER BY clause.

Note if you are using ORDER BY clause then make sure it must be the last statement of your SQL query.

**Example:**

I’ll be using the same table which we have been using in all our JOIN tutorials so far. These are the emp and dept tables.  
(For tables you can refer to [Table for SQL Joins](http://localhost/RebellionRider/sql-join-table-structure/) tutorial)

**Query 1: CROSS JOIN**

***SELECT  emp\_name, dept\_name  FROM  emp  CROSS JOIN  dept;***

This query is fairly simple; we are selecting emp name and dept name from emp and dept tables respectively.   
On executing this query, the first record of emp name column of emp table that is – Steve, gets paired with all the rows of the second table dept. Similarly second record Nancy gets paired with all the rows of dept table and so on.   
Since this cross join produces a Cartesian product therefore the total number of rows in the result will be equal to total number of rows in table 1 multiplied by total number of rows in table 2.   
Since in our case we have total 5 rows in each table thus total number of rows in our result is 25.

**Query 2: CROSS JOIN with WHERE clause.**

Say you want to see only those records where dept name is IT.

***SELECT  emp\_name,dept\_name  FROM  emp  CROSS JOIN  dept  WHERE dept\_name = ‘IT’;***

On execution this query will return all the emp name which are corresponding to Department name IT

**Query 3 : Cross Join with Order By clause**

Similarly you can use ORDER BY clause if you want to sort the result returned by your query in Ascending or Descending order.

***NOTE: Here ORDER BY clause must be the last statement of a QUERY.***

ORDER BY clause by default sorts the results in ascending order. But if you want to arrange the result in Descending order then you have to specify it by using DESCENDING or DESC keyword with the ORDER BY clause.

***SELECT  emp\_name,dept\_name  FROM  emp  CROSS JOIN  dept  WHERE dept\_name = ‘IT’  ORDER BY emp\_name;***

And if you want to reverse the order meaning if you want to sort the result in Descending order, then you just need to add DESC or DESCENDING keyword after the column name of ORDER BY clause

***SELECT  emp\_name,dept\_name  FROM  emp  CROSS JOIN  dept   WHERE dept\_name = ‘IT’  ORDER BY emp\_name  DESC;***