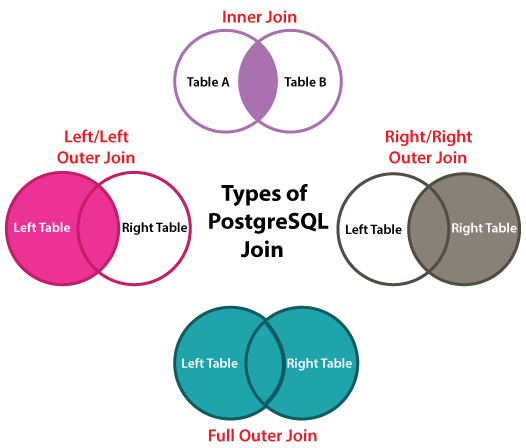
**JOIN in PostgreSQL**

PostgreSQL JOINS are used with [SELECT command](https://www.javatpoint.com/postgresql-select), which helps us to retrieve data from various tables. And we can merge the **Select and Joins** statements together into a single command. Whenever we want to get records from two or more tables, we will execute the joins commands. It is used to merge columns from one or more tables according to the data of the standard columns between connected tables. Usually, the **standard columns** of the **first table** are **primary key columns** and the **second table** columns are **foreign key** columns.

In PostgreSQL, we have various types of joins which are as follows:

* **Inner join**
* **Left join**
* **Right join**
* **Full outer join**
* **Cross join**

The below image displays most importantly used PostgreSQL joins, which we are going to explain in this section of the PostgreSQL tutorial.



**Example of PostgreSQL Joins**

Let us see some examples of different types of PostgreSQL joins:

Here, we will be creating and inserting the **two different tables** where we perform actions on several types of joins:

In the below example, we will use the Create command to create a ***Luxury\_cars*** table.

CREATE TABLE Luxury\_cars (

L\_ID INT PRIMARY KEY,

luxury\_car\_names VARCHAR (100) NOT NULL);

Once we execute the above command, we will get the below message, which displays that the ***Luxury\_cars*** table has been created successfully.

Here again, we will use the Create command to create a ***Sports\_cars*** table as follows:

CREATE TABLE Sports\_cars (

S\_ID INT PRIMARY KEY,

sports\_car\_names VARCHAR (100) NOT NULL);

Once we executed the above command, we will get the below message, which displays that the ***Sports\_cars*** table has been created successfully.

After that, we will insert some values in the ***Luxury\_cars*** table by using the INSERT command:

INSERT INTO Luxury\_cars (L\_ID, luxury\_car\_names)

VALUES

(1, 'Chevrolet Corvette'),

(2, 'Mercedes Benz SL Class'),

(3, 'Audi A7'),

(4, 'Genesis G90'),

(5,'Lincoln Continental');

After executing the above command, we will get the below message that the values have been inserted successfully into the ***Luxury\_cars*** table.

Just like we inserted the value in the ***Luxury\_cars*** table, we will insert the values into the ***Sports\_cars*** table as well with the help of Insert command:

INSERT INTO Sports\_cars (S\_ID, sports\_car\_names)

VALUES

 (1, 'BMW Z4'),

 (2, 'Nissan 370Z'),

 (3, 'Chevrolet Corvette'),

 (4, 'Mercedes Benz SL Class'),

(5,'Subaru BRZ');

After executing the above command, we will get the below message that the values have been inserted successfully into the ***Sports\_cars*** table.

The above tables have some similar cars, for example, **Chevrolet Corvette** and **Mercedes Benz SL Class**.

Now we will use the **SELECT** command to get the following data from the ***Luxury\_cars*** table:

Select \* from Luxury\_cars;

**Output**

day2=# Select \* from Luxury\_cars;

l\_id | luxury\_car\_names

------+------------------------

1 | Chevrolet Corvette

2 | Mercedes Benz SL Class

3 | Audi A7

4 | Genesis G90

5 | Lincoln Continental

(5 rows)

We will get the following data from the ***Sports\_cars*** table by using the SELECT command:

Select \* from Sports\_cars;

**Output**

day2=# Select \* from Sports\_cars;

s\_id | sports\_car\_names

------+------------------------

1 | BMW Z4

2 | Nissan 370Z

3 | Chevrolet Corvette

4 | Mercedes Benz SL Class

5 | Subaru BRZ

(5 rows)

After executing the above command, we will get the below result:

Now, let us see the working of different types of PostgreSQL Joins in real-time:

**PostgreSQL Inner join**

The PostgreSQL INNER JOIN is used to return all rows from various tables where the join condition is fulfilled.

**Syntax of PostgreSQL Inner Join**

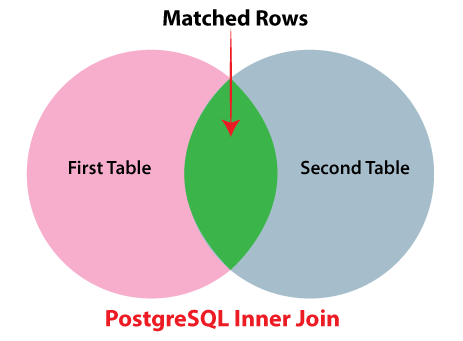
SELECT columns

FROM table1

INNER JOIN table2

ON table1.column = table2.column;

The below visual representation shows the working of PostgreSQL inner join:



**For Example**

We will take the above tables (**Luxury\_cars and Sports\_cars**) to understand the PostgreSQL inner join.

The below command will join the **first table (*Luxury\_cars***) with the **second table (Sports\_carsv)** by matching the values in the **luxury\_car\_name** and **sports\_car\_name** columns:

SELECT L\_ID, luxury\_car\_names, S\_ID, sports\_car\_names

FROM Luxury\_cars

INNER JOIN Sports\_cars

ON luxury\_car\_names= sports\_car\_names;

**Output**

**A black screen with white text

Description automatically generated**

Once we implemented the above command, we will get the below result where we can see the matched rows data from ***Luxury\_cars* and *Sports\_cars*** tables.

* The inner join is used to analyze each row in **Table A (*Luxury\_cars***).
* It equates the records in the **luxury\_car\_name** column with records in the **sports\_car\_name** column of each row in **Table B (*Sports\_cars***).
* If these records are similar, then the **inner join** creates a new row containing columns from both tables and enhances the particular row into the output.

**PostgreSQL Left join**

The PostgreSQL LEFT JOIN is used to return all rows from the left table, which can define in the **ON condition** and only those rows from the other table where the join condition is satisfied.

**Syntax of PostgreSQL Left join**

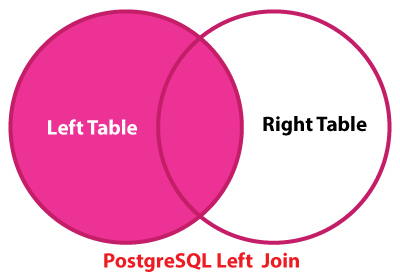
SELECT columns

FROM table1

LEFT JOIN table2

ON table1.column = table2.column;

The below visual representation displays the working of PostgreSQL Left join:



**For Example**

In the below command, we are going to use the Left Join condition to join the ***Luxury\_cars table*** with the **Sports\_cars** table.

In the Left join clause, **Table A or the first table** is known as the ***Left table***, and **Table B or the second table** is known as ***the Right table***.

SELECT L\_ID, luxury\_car\_names, S\_ID, sports\_car\_names

FROM Luxury\_cars

LEFT JOIN Sports\_cars

ON luxury\_car\_names= sports\_car\_names;

**Output**

Once we implemented the above command, we will get the below result.

**A black screen with white text

Description automatically generated**

**Working of PostgreSQL Left Join**

* In the above screenshot, the left join condition selects the records from the **left table ( Luxury\_cars)**, and it equates the values in the **luxury\_car\_names** column with the values in the **sports\_car\_names** column from the ***Sports\_cars*** table.
* If these records are similar, then the **left join** creates a new row, which is having the columns of both tables and adds the particular row to the result as we can see the **Row1 and Row2** in the above output.
* Suppose, if the values are not similar, then the left join also generates a new row, which involves the columns from both tables and adds it to the outcome.
* However, it fills the columns of the right table (***Sports\_cars***) with null as we can see the **Row3, Row4, and Row5** in the output.

**PostgreSQL RIGHT JOIN**

The **PostgreSQL RIGHT JOIN** is used to return all rows from the **Right table**, which can define in the **ON condition** and only those rows from another table where the join condition is fulfilled.

The **RIGHT JOIN** will get the data from the right table as it is the opposite of the **LEFT JOIN**.

**Syntax of PostgreSQL Right Join**

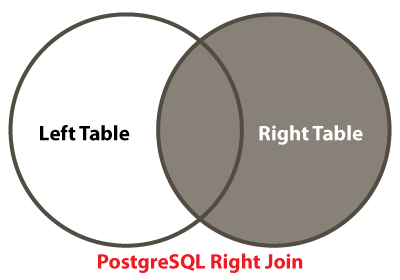
SELECT columns

FROM table1

RIGHT JOIN table2

ON table1.column = table2.column;

The below Venn diagram displays the working of PostgreSQL Right join:



**For Example**

The below command is used to represent the working of **Right join** where we join the **Luxury\_cars** table with the **Sports\_cars** table:

SELECT L\_ID, luxury\_car\_names, S\_ID, sports\_car\_names

FROM Luxury\_cars

RIGHT JOIN Sports\_cars

ON luxury\_car\_names= sports\_car\_names;

**Output**

A screen shot of a computer

Description automatically generated

After executing the above command, we will get the below output:

**Working of PostgreSQL Right join**

* In the above image, the **RIGHT JOIN** equates each value in the **Sports\_car\_names** column of every row in the ***Sports\_cars*** table with each value in the **luxury\_cars\_name** column of all row in the ***Luxury\_cars*** table.
* If these values are similar, then the **right join** generates a new row, which carries the columns from both tables (**Luxury\_cars and Sports\_cars**).
* Suppose, if the values are not similar, then the right join also produced a new row, which involves the columns from both tables and enhances it to the output.

**PostgreSQL Full Outer Join**

The **FULL OUTER JOIN** is used to return all records when there is a match in the **left table** or **right table** records.

**Syntax of PostgreSQL Full Outer Join**

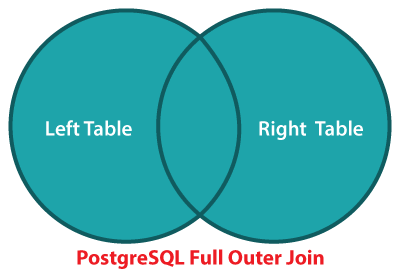
SELECT columns

FROM table1

FULL OUTER JOIN table2

ON table1.column = table2.column;

The below Venn diagram displays the working of PostgreSQL Full Outer join:



**For Example**

The below command is used to represent the working of the **Full Outer join** to join the **Luxury\_cars** table with the **Sports\_cars** table.

SELECT L\_ID, luxury\_car\_names, S\_ID, sports\_car\_names

FROM Luxury\_cars

FULL OUTER JOIN Sports\_cars

ON luxury\_car\_names= sports\_car\_names;

**Output**

A screen shot of a computer

Description automatically generated

After executing the above command, we will get the below result:

**PostgreSQL Cross Join**

Select \* from Luxury\_cars CROSS JOIN Sports\_cars;

A screenshot of a computer

Description automatically generated