

# **Lecture 7**

## **JOINS.**

**Inner Join. Full Join. Left Join.  
Right Join. Cross Join. Natural Join.  
Self Join.**

# **PostgreSQL JOINS.**

**PART I.** PostgreSQL JOINS. Inner Join. Full Join. Left Join. Right Join.

**PART II.** Cross Join. Natural Join. Self Join.

# **PART I. PostgreSQL JOINS. Inner Join. Full Join. Left Join. Right Join.**

# PostgreSQL Joins

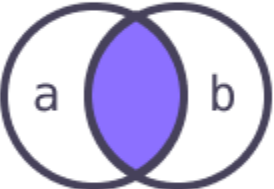


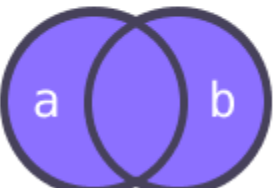
The **SQL Join** clause is used to combine data from two or more tables in a database. When the related data is stored across multiple tables, joins help you to retrieve records combining the fields from these tables using their foreign keys.

Following is the basic syntax of a the **SQL JOIN CLAUSE**:

```
SELECT column_name(s)
```

```
FROM table1
```

```
JOIN table2;
```

Join type	Visually	Example usage
Inner join		a <b>JOIN</b> b <b>ON</b> a.id = b.id
Left join		a <b>LEFT JOIN</b> b <b>ON</b> a.id = b.id
Right join		a <b>RIGHT JOIN</b> b <b>ON</b> a.id = b.id
Full outer join		a <b>FULL OUTER JOIN</b> b <b>ON</b> a.id = b.id

# Example

**SELECT**

first\_name,  
last\_name,  
booking\_id

**FROM**

passengers

**JOIN**

booking **ON** passengers.passenger\_id = booking.passenger\_id;

	Data Output	Explain	Messages	Notifications
	<b>first_name</b> character varying (50) 🔒	<b>last_name</b> character varying (50) 🔒	<b>booking_id</b> integer 🔒	
1	Stacee	Scud	1	
2	Ignacio	Manville	2	
3	Reilly	Scourgie	4	
4	Trista	Passion	5	
5	Gabriella	Beidebeke	6	

# Example

**SELECT**

`p.first_name,`  
`p.last_name,`  
`b.booking_id`

**FROM**

`passengers p`

**JOIN**

`booking b ON p.passenger_id = b.passenger_id;`

	Data Output	Explain	Messages	Notifications
	<b>first_name</b> character varying (50) 🔒	<b>last_name</b> character varying (50) 🔒	<b>booking_id</b> integer 🔒	
1	Stacee	Scud	1	
2	Ignacio	Manville	2	
3	Reilly	Scourgie	4	
4	Trista	Passion	5	
5	Gabriella	Beidebeke	6	

# Explanation

- SELECT**: Specifies the columns you want to retrieve.
- FROM**: Indicates the primary table (passengers) you are querying from.
- JOIN**: Combines rows from passengers and booking based on the matching condition.
- ON**: Specifies the condition for the join (matching passenger\_id in both tables).



# The SQL Inner Join

The **SQL Inner Join** is a type of join that combines multiple tables by retrieving records that have matching values in both tables (in the common column).

Following is the basic syntax of SQL Inner Join:

```
SELECT column_name(s)
```

```
FROM table1
```

```
INNER JOIN table2
```

```
ON table1.column_name = table2.column_name;
```

# Explanation of Inner Join

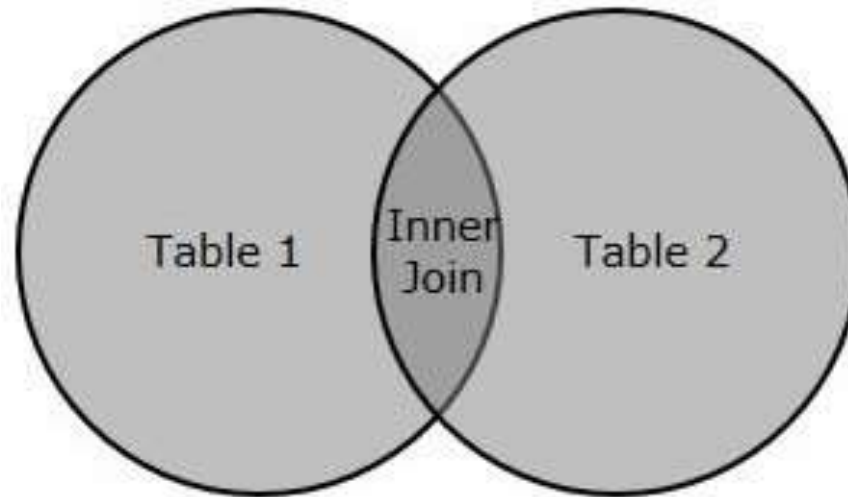
EmpDetails			MaritalStatus		
ID	Name	Salary	ID	Name	Status
1	John	40000	1	John	Married
2	Alex	25000	3	Simon	Married
3	Simon	43000	4	Stella	Unmarried

+

=

ID	Name	Salary	Status
1	John	40000	Married
3	Simon	43000	Married

**The following Venn diagram illustrates the inner join:**



# Example

```
SELECT p.passenger_id, p.first_name, p.last_name, b.booking_platform,  
       b.booking_id,  
       b.created_at  
FROM  
       passengers p  
INNER JOIN  
       booking b ON p.passenger_id = b.passenger_id  
WHERE  
       b.booking_platform = 'Johns Inc';
```

passenger_id integer	first_name character varying (50)	last_name character varying (50)	booking_platform character varying (50)	booking_id integer	created_at timestamp without time zone
139	Kermy	Graves	Johns Inc	40	2023-08-01 13:53:27

# Joining Multiple Tables Using Inner Join

It is possible to join as many tables as possible, using Inner Join, by specifying the condition :

```
SELECT column1, column2, column3...
```

```
FROM table1
```

```
INNER JOIN
```

```
table2
```

```
ON condition_1
```

```
INNER JOIN
```

```
table3 ON
```

```
condition_2
```

```
....
```

```
INNER JOIN
```

```
tableN ON
```

# Example

```
SELECT p.passenger_id, p.first_name, p.last_name,  
       b.booking_id,  
       f.update_at,  
       b.created_at  
FROM  
       passengers p  
INNER JOIN  
       booking b ON p.passenger_id = b.passenger_id  
INNER JOIN  
       booking_flight f ON b.booking_id = f.booking_id;
```

Data Output		Explain	Messages	Notifications		
	<b>passenger_id</b> integer	<b>first_name</b> character varying (50)	<b>last_name</b> character varying (50)	<b>booking_id</b> integer	<b>update_at</b> timestamp without time zone	<b>created_at</b> timestamp without time zone
1	68	Augustine	Bellanger	184	2023-09-21 15:12:45	2023-12-04 21:24:37
2	76	Murray	Aston	444	2024-02-29 20:10:56	2023-04-29 22:46:03
3	58	Harv	Kilshall	478	2023-07-09 12:17:19	2023-03-19 23:48:17
4	195	Uta	Rzehor	275	2024-02-09 03:08:59	2023-03-18 08:24:24
5	142	Ignacius	McMeyler	309	2023-05-04 07:17:59	2023-09-15 08:54:16

# The SQL OUTER Join

An **Outer Join** retrieves all the records in two tables even if there is no counterpart row of one table in another table, unlike Inner Join.

Following are the different types of outer Joins:

- 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.

# The SQL Left Join

**Left Join** or **Left Outer Join** in SQL combines two or more tables, where the first table is returned wholly; but, only the matching record(s) are retrieved from the consequent tables.

Following is the basic syntax of Left Join in SQL:

```
SELECT column_name(s)
```

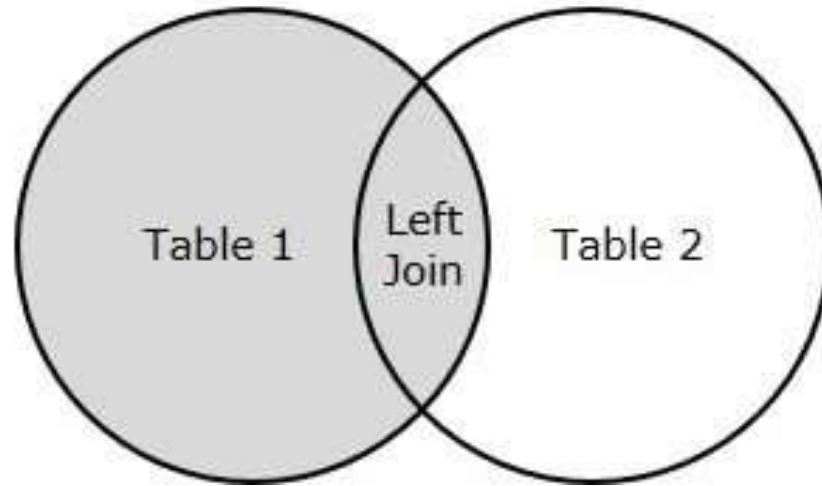
```
FROM table1
```

```
LEFT JOIN table2
```

```
ON table1.column_name = table2.column_name;
```



**The following Venn diagram illustrates the left join:**



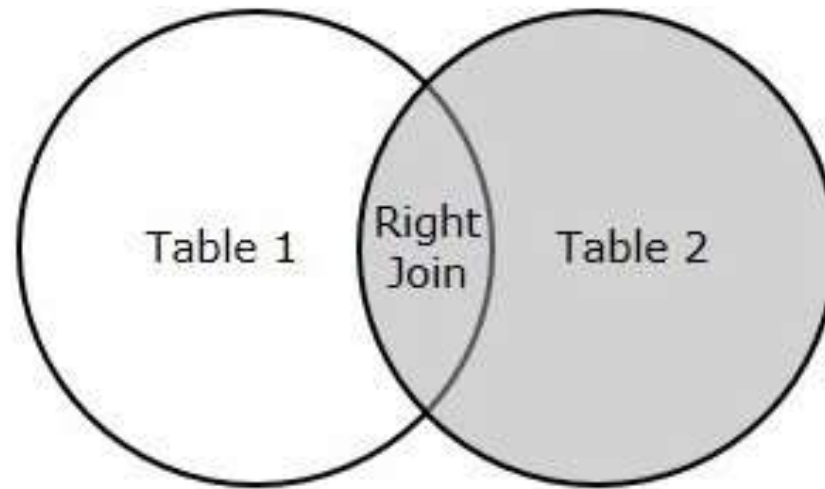
# The SQL Right Join

The **Right Join** or **Right Outer Join** query in SQL returns all rows from the right table, even if there are no matches in the left table.

Following is the basic syntax of Right Join in SQL:

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

**The following Venn diagram illustrates the right join:**



# Example

```
SELECT
    first_name,
    last_name,
    booking_id
FROM
    passengers
RIGHT JOIN
    booking ON passengers.passenger_id = booking.passenger_id;
```

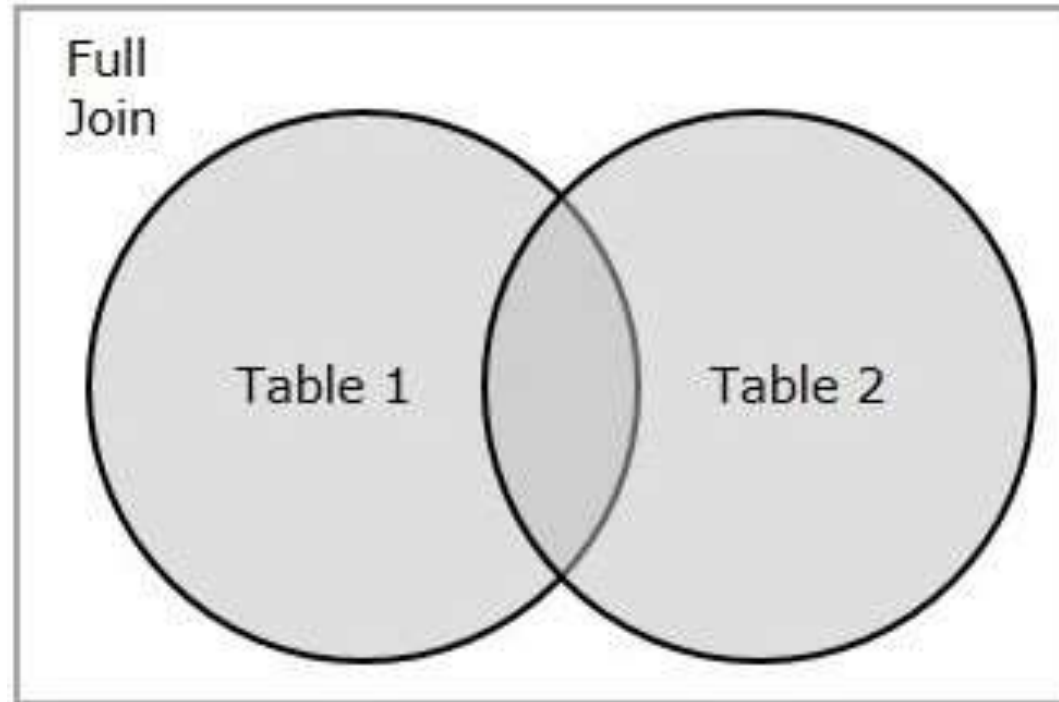
# The SQL Full Join

**SQL Full Join** creates a new table by joining two tables as a whole. The joined table contains all records from both the tables and fills NULL values for missing matches on either side.

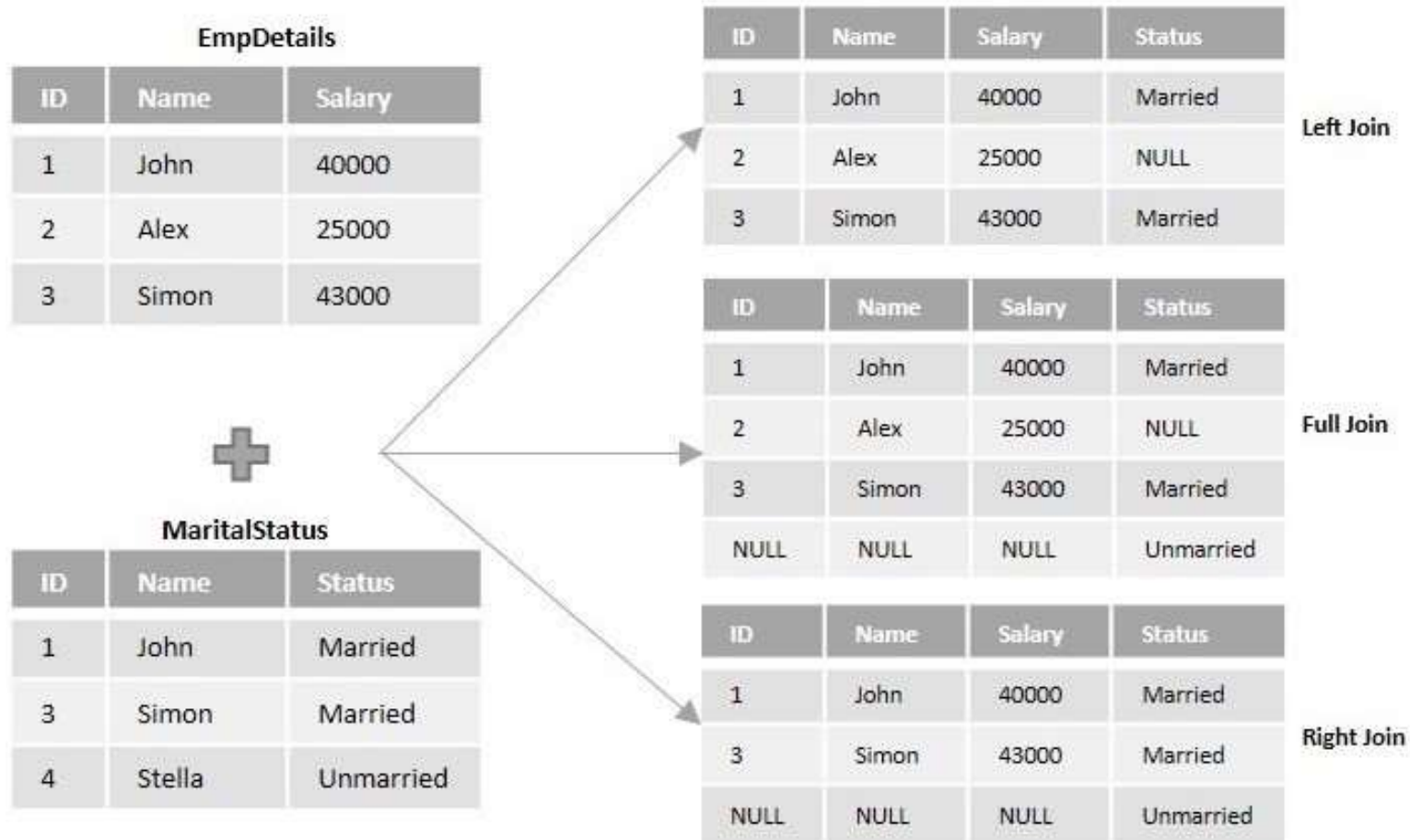
Following is the basic syntax of Full Join in SQL:

```
SELECT column_name(s)
FROM table1
FULL JOIN table2
ON table1.column_name = table2.column_name;
```

**The following Venn diagram illustrates the full outer join:**



# Outer join explanation



# Example

**SELECT**

first\_name,  
last\_name,  
booking\_id

**FROM**

passengers

**LEFT JOIN**

booking **ON** passengers.passenger\_id = booking.passenger\_id;

Data Output

Explain

Messages

Notifications

	first_name character varying (50)	last_name character varying (50)	booking_id integer
498	Roungo	Feanyougn	499
499	Shani	Hooks	500
500	Lizabeth	Summersby	3
501	Killy	Davidman	[null]
502	Harwilll	Salan	[null]
503	Legra	Robard	[null]



## **PART II. Cross Join. Natural Join. Self Join.**

# SQL NATURAL JOIN

A **natural join** is a join that creates an implicit join based on the same column names in the joined tables.

The following shows the syntax of the PostgreSQL natural join:

```
SELECT select_list
```

```
FROM T1
```

```
NATURAL [INNER, LEFT, RIGHT] JOIN T2;
```

# SQL NATURAL JOIN examples

---

```
SELECT *  
FROM passengers  
NATURAL JOIN booking;
```

# Differences

Join Type	Returns Rows Where Match Exists	Returns Unmatched Rows from Left Table	Returns Unmatched Rows from Right Table
NATURAL JOIN	Yes (based on common columns)	No	No
INNER JOIN	Yes	No	No
LEFT JOIN	Yes	Yes	No
RIGHT JOIN	Yes	No	Yes
FULL OUTER JOIN	Yes	Yes	Yes

# Differences

- **NATURAL JOIN** is convenient for combining tables with multiple common columns, but it can lead to unexpected results if not managed carefully.
- **INNER JOIN** is precise for matching records in both tables.
- **LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN** provide more flexibility for including records even when matches don't exist, with different orientations (left or right) depending on the focus of the query.

# SQL Self-Join

A **self-join** is a regular join that joins a table to itself. In practice, you typically use a self-join to query hierarchical data or to compare rows within the same table.

The following query uses an INNER JOIN that joins the table to itself:

```
SELECT select_list  
  
FROM table_name t1  
INNER JOIN table_name t2 on join_predicate;
```

# Example

---

**SELECT**

```
p1.passenger_id AS Passenger_ID,  
p1.first_name AS Passenger_Name,  
p2.first_name AS Manager_Name
```

**FROM**

```
passengers p1
```

**LEFT JOIN**

```
passengers p2 ON p1.passenger_id = p2.passenger_id;
```

# SQL Cross Join

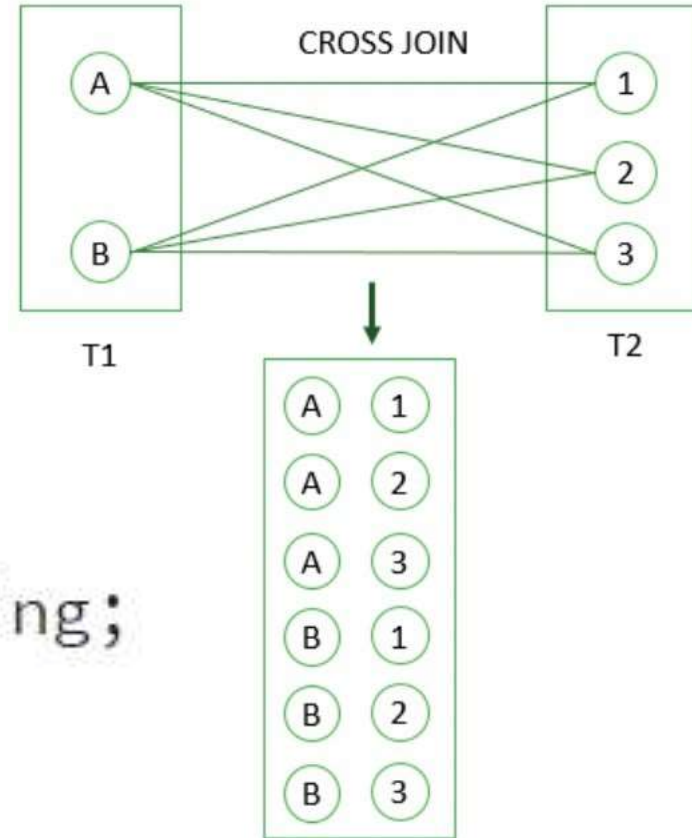
A **CROSS JOIN** clause allows you to produce a Cartesian Product of rows in two or more tables.

The following illustrates the syntax of the CROSS JOIN syntax:

```
SELECT select_list  
FROM T1  
CROSS JOIN T2;
```




# SQL CROSS JOIN example




```
SELECT *
```

```
FROM passengers CROSS JOIN booking;
```

Table 1 

1		
2		

Table 2 

1		
3		
4		

Outer Join 

1				
2				
3				
4				

Inner Join 

1				

Left Join 

1				
2				

Union  + 

1		
2		
1		
3		
4		

Cross Join 

1			1	
1			3	
1			4	
2			1	
2			3	
2			4	