Joins

What are SQL joins?

→ In SQL, a joins is a way to combined data from two or more database tables based on a related column between them.

Joins are used when we want to query information that is distributed across multiple tables in a database, and the information we need is not contained in a single table. By joining tables together, we can create a virtual table that contains all of the information we need for our query.

But why have data in multiple tables?

userid	name	State	city
1	bharat	Gujrat	Ahemdabad
2	Pearl	Maharashtra	Pune
3	Jahan	MP	Bhopal
4	Divsha	Rajasthan	Jaipur
5	Kasheen	WB	Kolkata
6	Hazel	Karnatka	Bangalore
7	Sonakshi	J&K	Kashmir

Order-id	userid	date
B-25601	1	-
B-26011	1	-
B-26074	1	-
B-25602	2	-
B-25114	2	-
B-25614	2	-
B-22541	3	-

common column

Order-id	name
B-25601	bharat
-	bharat
-	-
-	-
-	-
-	-
-	-
-	-
-	-

* why not store two different tables in single large table?

→ Bcz

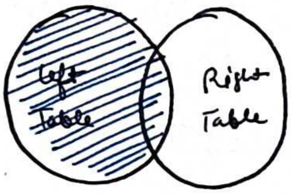
user_id	name	State	City
–	–	Guj	Ahem
–	–	Guj	Ahem
–		–	Ah
–		–	

] — change city, State

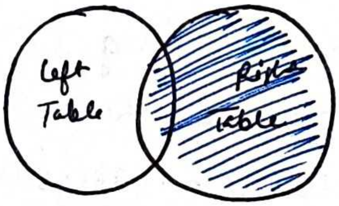
we have to change all city and state, if one-two miss than create error in future and take large storage.

Types of JOINS

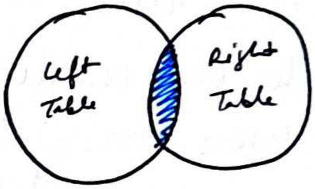
→ left JOIN



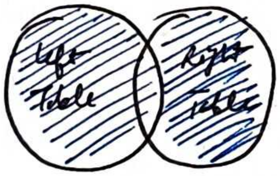
→ Right Join



→ Inner Join



→ Full Join



→ Cross Join

→ Self join

Cross joins → Cartesian Products

In SQL, a cross join(also known as a cartesian product) is a type of join that return the cartesian product of the two tables being joined. In other words, it return all possible combinations of rows from the two tables.

Cross joins are not commonly used in practice, but they can be useful in certain scenarios, such as generating test data or exploring all possible combination of items in a product catalogue. However, it's important to be cautious when using cross joins with large tables, as they can generate a very large result set, which can be resource-intensive and slow to process.

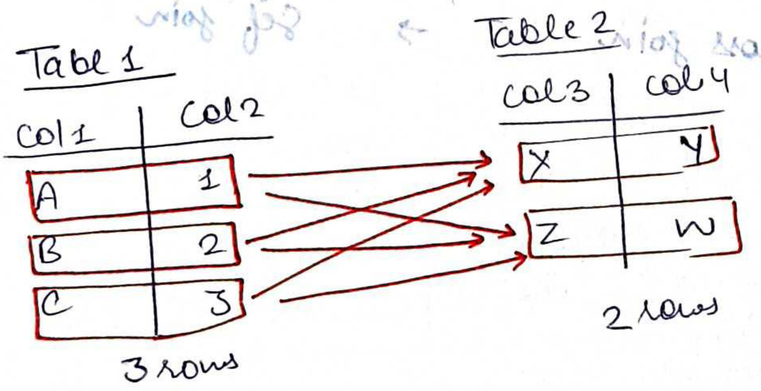


Table 1

col1	col2
A	1
B	2
C	3

3 rows

Table 2

col3	col4
X	Y
Z	W

2 rows

Result

col1	col2	col3	col4
A	1	X	Y
A	1	Z	W
B	2	X	Y
B	2	Z	W
C	3	X	Y
C	3	Z	W

3×2 = 6 rows

Inner Joins

In SQL, an inner join is a type of joins operation that combines data from two or more tables based on a specified condition. The inner join returns only the rows from both table that Satisfy the specified condition, i.e the matching rows

When you perform an inner join on two tables, the result set will only contains rows where there is a match between the joining columns in both tables. If there is no match, then the row will not be included in the result set.

Table 1]

Employee_ID	Name	Department ID	Salary
1	John	1	100k
2	Jane	2	50k
3	Tim	3	75k
4	Bob	1	90k
5	Lisa	2	120k
6	Mike	4	60k
7	Sarah	NULL	80k

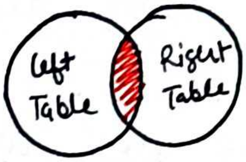


Table 2]

Department_ID	Department_name
1	Engineering
2	Sales
3	Finance
4	Marketing
5	Operations

Employee ID	Name	Department.ID	Salary	DepartmentID	Department
1	John	1	100k	1	Engineer
4	Bob	1	90k	1	Engineer
2	Jane	2	50k	2	Sales
5	Lisa	2	120k	2	Sales
3	Tim	3	75k	3	Finance
6	Mike	4	60k	4	Marketing

Left Join

A left join, also known as a left outer join, is a type of SQL join operation that returns all the rows from the left table (also known as the "first" table) and matching rows from the right table (also known as "second" table). If there are no matching rows in the right table, the result will contain NULL values in the columns that come from the right table.

In other words, a left join combines the row from both tables based on a common column, but it also includes all the row from the left table, even if there are no matches in the right table. This is useful when you wan to include all the records from the first table, but only some records from the second tables.

Employee-ID	Name	Department ID	Salary
1	John	1	100k
2	Jane	2	50k
3	Bob	3	75k
4	Lisa	1	90k
5	Mike	2	120k
6	Tim	4	60k
7	Sarah	5	80k
8	Mark	2	95k

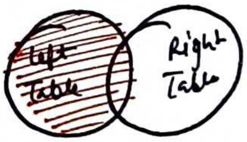


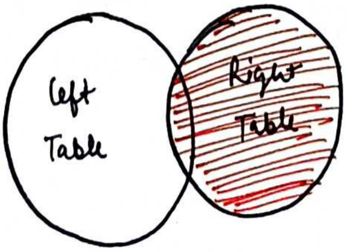
Table 2

Department_ID	Department Name
1	Engineering
2	Sales
3	Finance

Employee ID	Name	Department_ID	Salary	Department ID	Department
1	John	1	100k	1	Engineering
2	Jane	2	50k	2	Sales
3	Bob	3	75k	3	Finance
4	Lisa	1	90k	1	Engineering
5	Mike	2	120k	2	Sales
6	Tim	4	60k	NULL	NULL
7	Sarah	5	80k	NULL	NULL
8	Mark	2	95k	2	Sales

Right Join

A right join, also known as a right outer join, is type of join operations in SQL that returns all the rows from the right table and matching rows from the left tables. If there are no matches in the left table, the result will still contain all the rows from the right table, with Null values for the columns from the left table.



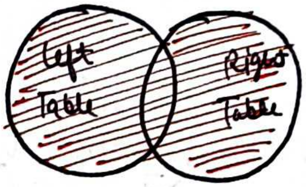
Employee ID	Name	Department ID	Salary
1	John	1	100k
2	Jane	2	50k
3	Bob	3	75k
4	Lisa	1	90k
5	Mike	2	120k
7	Sarah	NULL	80k
8	Mark	2	95k
8			

Department ID	Department Name
1	Engineering
2	Sales
3	Financial
4	Marketing
5	HR

Employee ID	Name	Department ID	Salary	Department Name
1	John	1	100k	Engineering
4	Lisa	1	90k	Engineering
2	Jane	2	50k	Sales
5	Mike	2	120k	Sales
8	Mark	2	95k	Sales
3	Bob	3	75k	Finance
NULL	NULL	4	NULL	Marketing
NULL	NULL	5	NULL	HR

Full Outer Join

A full outer join, sometimes called a full join, is a type of join operation in SQL that return all matching rows from both the left and right tables, as well as any non-matching rows from either table. In other words, a full outer join return all the rows from both tables and matches rows with common values in the specified column, and fills in NULL values for columns where there is no match.



emp id	emp name	dept id
1	Alice	1
2	bob	1
3	Charlie	2
4	Dave	null
5	Eve	3

dept id	dept name
1	Sales
2	Marketing
3	Finance
4	IT
5	HR

emp id	emp name	dept id	dept name	dept id
1	Alice	1	Sales	1
2	bob	1	Sales	1
3	Charlie	2	Marketing	2
4	Dave	null	null	null
5	Eve	3	Finance	3
null	null	null	IT	4
null	null	null	HR	5

CROSS JOIN

database name　　table name

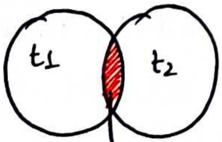
SELECT * FROM dvdb1 . user t1 → nickname of table

CROSS JOIN dvdb1 . groups t2 → nick name of table

database name　　table name

user ↓ 4 rows　　groups ↓ 6 rows　　CROSS → JOIN　　New Table ↓ 24 rows

INNER TABLE →



→ show common Table

database name　　table name

SELECT * FROM dvdb1. membership t1 → nick name of table

INNER JOIN dvdb1. user t2
table name

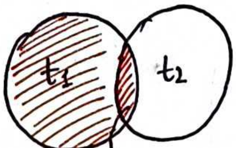
ON t1. user_id = t2. user_id

↳ condition

column name which is common of both table

* default → JOIN is inner Join

Left Join



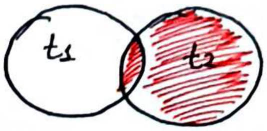
→ left table and common row will show

SELECT * FROM dvdb1. membership t1

LEFT JOIN dvdb1. user t2

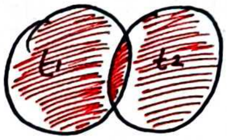
ON t1. user_id = t2. userid

Right Join →



```
SELECT * FROM dvdb1. membership t1
RIGHT JOIN dvdb1. users t2
ON t1. user_id = t2. user_id
```

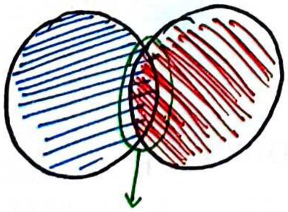
Full Outer JOIN →



```
SELECT * FROM dvdb1. membership t1
FULL OUTER JOIN dvdb1. user t2
ON t1. user_id = t2. userid
```

→ This query not working

```
SELET * FROM dvdb1. membership t1
LEFT JOIN dvdb1. user t2
ON t1. userd_id = t2. user_id
[UNION]
SELET * FROM dvdb1. membership t1
RIHGT JOIN dvdb1. user t2
ON t1. userid = t2. user_id
```



because left join mein common row agya and Right join mein bhi common row agya. 2 baar common row agya to single ban karne ke liye UNION.

SQL Set Operation

1. **Union:** The UNION Operator is used to combine the results of two or more SELECT Statements into a single result set. The UNION Operator removes duplicate rows between the various SELECT statements.

Person 1

id	Name
1	Alice
2	Bob
3	Charlie

Person 2

id	Name
3	Charlie
4	David
5	Emily

Union

id	name
1	Alice
2	Bob
3	Charlie
4	David
5	Emily

2. **Union All:** The union All operator is similar to the UNION operator, but it does not remove duplicate rows from the result set.

Person 1

id	name
1	Alice
2	Bob
3	charlie

Person 2

id	Name
3	charlie
4	David
5	Emily

union All

id	Name
1	Alice
2	Bob
3	charlie
3	charlie
4	David
5	emily

3. **INTERSECT** : The intersect operator returns only the rows that appear in both result sets of two SELECT statements

Person 1

id	name
1	Alice
2	Bob
3	Charlie

Person2

id	Name
3	charlie
4	David
5	Emily

Intersect

id	Name
3	charlie

4. **Except**: The EXCEPT or MINUS operator return only the distinct row that appear in the first result set but not in the second result set of two SELECT statements.

Person1

id	name
1	Alice
2	Bob
3	Charlie

Person2

id	Name
3	Charlie
4	David
5	Emily

Except

id	name
1	Alice
2	bob

Union

```
SELECT * FROM dvdb1. person1
UNION
SELECT * FROM dvdb1. person2
```

Union All

```
SELECT * FROM dvdb1. person1
UNION ALL
SELECT * FROM dvdb1. person2
```

Intersect

```
SELECT * FROM dvdb1. person1
INTERSECT / EXCEPT
SELECT * FROM dvdb1. person2
```

Self Joins

A self join is a type of join in which a table is joined with itself. This means that the table is treated as two separate tables, with each row in the table being compared to every other row in the same table.

Self joins are used when you want to compare the values of two different rows within the same table. For example you miget use a self join to compare the salaries of two employee who work in the same department, or to find all pairs of customers who have the same billing address.

Table 1

user-id	name	age	emergency-contact
1	Nitish	34	11 → 11 number user-id
2	Ankit	32	1 → 1 number user id
3	Neha	23	1 → " " "
4	Radhika	34	3 → 3 num. userd-id
8	Abhinav	31	11 → 11 " "
11	Rahul	29	8 → 8 " "

Table 2

user-id	name	age	emergency-contact
1	Nitish	34	11
2	Ankit	32	1
3	Neha	23	1
4	Radhika	34	3
8	Abhinav	31	11
11	Rahul	29	8

Self Join of Table1 and Table2

expected output:-

name	emergency contact for
Nitish	Rahul → 11
Ankit	Nitish → 1
-	-
-	-
-	-
-	-

Joining on more than one cols

Student

Std-id	first-name	last-name	class-id	enroll-year
1	John	Smith	1	2021
2	Jane	Doe	2	2020
3	Bob	Johnson	1	2021
4	Sally	Brown	3	2022
5	Tom	Williams	2	2022
6	Alice	Davis	4	2020

→ class-id common bet^n both table but every teacher change. So, we have to match enrollyear and class year

Class

Class-id	Class-name	teacher	Class-year
1	Math 101	Mr. smith	2021
2	English 1	Mrs. Johnson	2021
3	Science 1	Dr. Lee	2022
4	History	Mrs. Williams	2022

```
SELECT * FROM dudb1. students t1
JOIN dudb1. class t2
ON t1. class_id = t2. class_id
AND t1. enrollment_year = t2. class_year
```

use another joins → Right / Left- - -

Joining more than 2 tables

```
SELECT * FROM flipcart. order_detail t1
JOIN flipcart. orders t2
ON t1. order_id = t2. order_id
JOIN flipcart. users t3
ON t2. user_id = t3. user_id
```

Filtering columns

```
SELECT
t1. order_id,
t1. amount,
t1. profit,
t3. name
FROM flipcart. order_detail t1
JOIN flipcart . orders t2
ON t1. order_id = t2. order_id
JOIN flipcart . users t3
ON t2. user_id = t3. user_id
```

select specific columns