

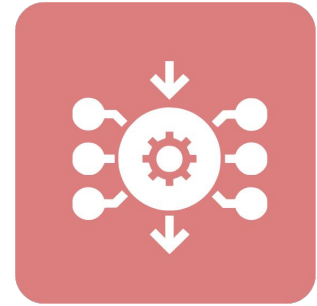
JOINS - II and Union

Relevel
by Unacademy



What is Self JOIN?

A self-join is a Structured Query Language (SQL) statement that joins a table with itself. An inner join is performed on a single table, especially when comparing records from the same table to determine a relationship.



Structure of a Self Join Query

```
SELECT a.coulmn1 , b.column2
```

```
FROM table_name a, table_name b
```

```
WHERE some_condition;
```

How SQL SELF JOIN work?

A self-join joins a table with itself based on a predefined condition. Assume we have a group of students, and one of them has a best friend relationship with another student at the same table, and we want to know the names of the student and his friend.

Id	Student_name	Friend_id
1	JAY	3
2	RAY	1
3	MAY	2

How SQL SELF JOIN work?

Now, to obtain the name of each student and their friend, we can use a self-join, which will join the table in the following manner if friend id equals student id.

id	Student_name	Friend_id
1	JAY	3
2	RAY	1
3	MAY	2

id	Student_name	Friend_id
1	JAY	3
2	RAY	1
3	MAY	2

How SQL SELF JOIN work?

The resultant table will look something like this :

Id	Student_name	Friend
1	JAY	MAY
2	RAY	JAY
3	MAY	RAY

Understanding Self Join with Demonstrations

Let's have a look at the records in the customer's table. We will use this table in next few examples:

ID	Customer	City	Country	Items_purchased	Amount_paid
1	Peter King	Manchester	England	Books	120
2	Priya Krishna	New Delhi	India	pen	50
3	Jim Halpert	Manchester	England	pencil	43
4	Michael Scott	New York	USA	Books	250
5	Harvey Spector	Birmingham	England	pencil	100
6	Deepa Kamat	Mumbai	India	Books	370
7	Anita Desai	London	England	pencil	50
8	Rachel Zane	Michigan	USA	pen	70
9	Pretoria John	Canberra	Australia	pen	190
10	John L	Budapest	Hungary	Books	540
11	Justin Green	Ottawa City	Canada	pen	65
12	Babita Ghosh	Kolkata	India	pencil	75
13	Krish Pratt	London	England	eraser	30
14	Elizabeth Blunt	London	England	pencil	340
15	Nina Debrov	Amsterdam	Netherlands	Books	452

Self Join – Example 1

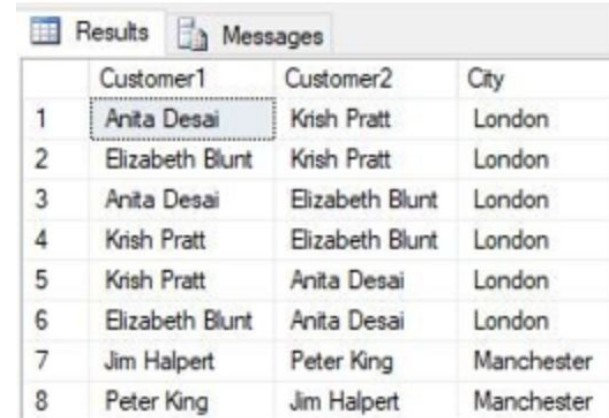
Find the pairs of customers who belong to the same city.

```
SELECT t1.Customer AS Customer1, t2.Customer AS Customer2, t1.City
```

```
FROM Customers t1, Customers t2
```

```
WHERE t1.ID <> t2.ID AND t1.City = t2.City
```

```
ORDER BY t1.City;
```



	Customer1	Customer2	City
1	Anita Desai	Krish Pratt	London
2	Elizabeth Blunt	Krish Pratt	London
3	Anita Desai	Elizabeth Blunt	London
4	Krish Pratt	Elizabeth Blunt	London
5	Krish Pratt	Anita Desai	London
6	Elizabeth Blunt	Anita Desai	London
7	Jim Halpert	Peter King	Manchester
8	Peter King	Jim Halpert	Manchester

Self Join – Example 1

	Customer1	Customer2	City
1	Anita Desai	Krish Pratt	London
2	Elizabeth Blunt	Krish Pratt	London
3	Anita Desai	Elizabeth Blunt	London
4	Krish Pratt	Elizabeth Blunt	London
5	Krish Pratt	Anita Desai	London
6	Elizabeth Blunt	Anita Desai	London
7	Jim Halpert	Peter King	Manchester
8	Peter King	Jim Halpert	Manchester

Self Join – Example 2

Find customers who belong to the same city and have shopped for pencils.

```
SELECT t1.Customer AS Customer1, t2.Customer AS Customer2, t1.City  
FROM Customers t1, Customers t2  
WHERE t1.ID <> t2.ID AND t1.City = t2.City AND t1.Items_purchased = 'pencil'  
AND t2.Items_purchased = 'pencil'  
ORDER BY t1.City;
```

	Customer1	Customer2	City
1	Anita Desai	Elizabeth Blunt	London
2	Elizabeth Blunt	Anita Desai	London

Self Join – Example 3

Find the customers who belong to the same country and have shopped for less than 100.

```
SELECT t1.Customer AS Customer1, t2.Customer AS Customer2, t1.Country, t1.Amount_paid , t2.
```

```
Amount_paid
```

```
FROM Customers t1, Customers t2
```

```
WHERE t1.ID <> t2.ID AND t1.Country = t2.Country AND t1.Amount_paid < 100 AND
```

```
t2.Amount_paid < 100
```

```
ORDER BY Country;
```

	Customer1	Customer2	Country	Amount_paid	Amount_paid
1	Anita Desai	Krish Pratt	England	50	30
2	Krish Pratt	Anita Desai	England	30	50

Self Join – Example 4

Find the pair of customers who have spent more than 300 on Books.

```
SELECT t1.customer, t1.Amount_paid, t2.customer, t2.Amount_paid  
FROM customers as t1 , customers as t2  
WHERE t1.Items_purchased = t2.Items_purchased  
AND t1.Items_purchased = 'Books' AND t2.Items_purchased = 'Books'  
AND t1.Amount_paid > 300 and t2.Amount_paid > 300 AND t1.ID <> t2.ID  
Order by t1.Customer;
```



Self Join – Example 4

Results		Messages		
	customer	Amount_paid	customer	Amount_paid
1	Deepa Kamat	370	John L	540
2	Deepa Kamat	370	Nina Debrov	452
3	John L	540	Deepa Kamat	370
4	John L	540	Nina Debrov	452
5	Nina Debrov	452	Deepa Kamat	370
6	Nina Debrov	452	John L	540

Instructions for practice questions



- Create account on
 - <https://www.stratascratch.com/>
- Refer to the url provided in the practice questions or look at the provided snippet of the dataset

Practice Question - 1

<https://platform.stratascratch.com/coding/9894-employee-and-manager-salaries?python=>

Solution - 1

```
SELECT
    a.first_name AS employee_name,
    a.salary AS employee_salary
FROM
    employee AS a
JOIN
    employee AS b
ON a.manager_id = b.id
WHERE a.salary > b.salary;
```

Practice Question - 2

Write a query to return the first name and last name of superior of each employee.

Employee

id	first_name	last_name	salary	manager_id
1	John	Watson	7550	NULL
2	Anne	Brown	3500	1
3	James	Black	3000	1
4	Scarlett	Miller	2500	3
5	Ethan	Davis	1200	3
6	Jacob	Smith	2000	3

Solution - 2

```
SELECT
    e.id,
    e.first_name,
    e.last_name,
    e.salary,
    m.first_name AS fname_boss,
    m.last_name AS lname_boss
FROM
    employee e
JOIN
    employee m
ON e.manager_id = m.id
```

Solution - 2

Output

id	first_name	last_name	salary	fname_boss	lname_boss
2	Anne	Brown	3500	John	Watson
3	James	Black	3000	John	Watson
4	Scarlett	Miller	2500	James	Black
5	Ethan	Davis	1200	James	Black
6	Jacob	Smith	2000	James	Black

Practice Question - 3

Write a query to return the name of manager, employee id, and employee salary.

Employee

Id	Name	Salary	ManagerId
1	Greg	100000	1
2	George	150000	1
3	Helen	130000	1
4	Tom	120000	2
5	Kevin	110000	2
6	David	120000	3
7	Geek	110000	3
8	Lucy	150000	3
9	Tesla	120000	3

Solution - 3

```
SELECT
    m.Name AS [Manager Name],
    e.Id AS [Employee ID],
    e.salary AS [Employee Salary]
FROM
    Employees e
JOIN
    Employees m
ON e.ManagerId = m.Id
```

Solution - 3

Output

	Manager Name	Employee ID	Employee Salary
1	Greg	1	100000
2	Greg	2	150000
3	Greg	3	130000
4	George	4	120000
5	George	5	110000
6	Helen	6	120000
7	Helen	7	110000
8	Helen	8	150000
9	Helen	9	120000

Practice Question - 4

Write a query to return the name of manager, number of team members, and sum of total salary of the team.

Employee

Id	Name	Salary	ManagerId
1	Greg	100000	1
2	George	150000	1
3	Helen	130000	1
4	Tom	120000	2
5	Kevin	110000	2
6	David	120000	3
7	Geek	110000	3
8	Lucy	150000	3
9	Tesla	120000	3

Solution - 4

```
SELECT
    m.Name AS [Manager Name],
    count(m.Id) AS [team amount],
    sum(e.salary) AS [Total Salary]
FROM
    Employees e
JOIN
    Employees m
ON e.ManagerId = m.Id
GROUP BY m.name
```

Solution - 4

Output

	Manager Name	team amount	Total Salary
1	George	2	230000
2	Greg	3	380000
3	Helen	3	350000

Practice Question - 5

Consider a dataset film. Find all pairs of the movies that have the same length

Dataset Description

film
* film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
rating
last_update
special_features
fulltext

Solution - 5

```
SELECT
    f1.title,
    f2.title,
    f1.length
FROM
    film f1
INNER JOIN film f2
    ON f1.film_id <> f2.film_id AND
    f1.length = f2.length;
```

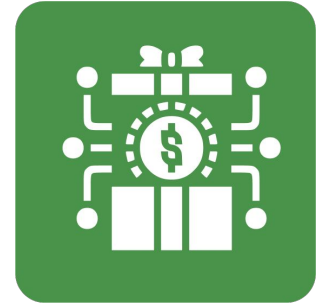
Solution - 5

Output

	title character varying (255)	title character varying (255)	length smallint
1	Chamber Italian	Resurrection Silverado	117
2	Chamber Italian	Magic Mallrats	117
3	Chamber Italian	Graffiti Love	117
4	Chamber Italian	Affair Prejudice	117
5	Grosse Wonderful	Hurricane Affair	49
6	Grosse Wonderful	Hook Chariots	49
7	Grosse Wonderful	Heavenly Gun	49
8	Grosse Wonderful	Doors President	49
9	Airport Pollock	Sense Greek	54
10	Airport Pollock	October Submarine	54
11	Airport Pollock	Kill Brotherhood	54

What is CARTESIAN/CROSS JOIN?

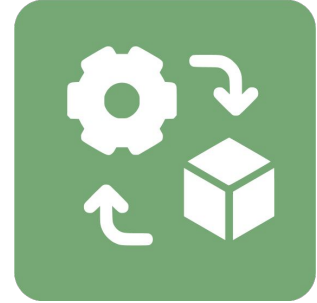
The CARTESIAN JOIN is also referred to as the CROSS JOIN. Each row of one table is joined to every row of another table in a CARTESIAN JOIN. This usually occurs when no matching column or WHERE condition is specified.



- In the absence of a WHERE condition, the CARTESIAN JOIN behaves similarly to the CARTESIAN PRODUCT. In other words, the number of rows in the result-set is the product of the two tables' rows.
- When the WHERE condition is present, this JOIN will behave like an INNER JOIN.
- In general, a cross join is similar to an inner join in that the join-condition continually evaluates to True.

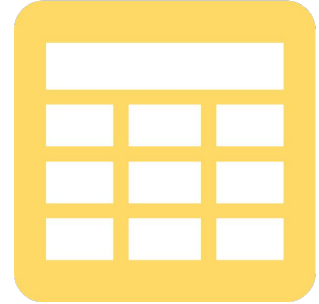
How does CROSS JOIN work?

- It is typically used to make combinations of each row from two or more tables. Cross Product Join refers to this result table of all row pairs.
- The SQL Cross Join is useful for obtaining all the combinations of items from two different records of two other collections of tables, such as colors or sizes, etc., at the business level or where an extensive database of this type is present in the industries.



How does CROSS JOIN work?

- As a result, when a Cross Join in a database does not use a WHERE clause, the rows in the first table are multiplied by the rows in the second table, we get a Cartesian product.
- Assume we have two tables X and Y with 'n' and rows, respectively, and when we perform a Cross join or Cartesian of these two tables, each row of tables is paired to the other, and the result set contains $n*m$ rows as the result rows of the $X*Y$ Cartesian product.



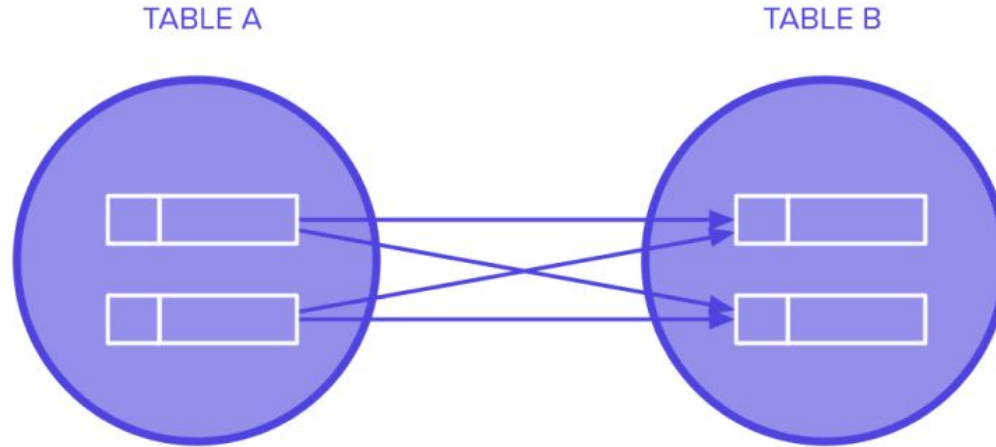
Structure of a Cross Join Query

Structure of a CROSS Join Query

```
SELECT table1.column1 , table1.column2, table2.column1...  
  
FROM table1  
  
CROSS JOIN table2;
```



CROSS JOIN – A visual Demonstration



CROSS JOIN – Example 1

Question - Retrieve colors allotted to each id and with their sizes

T-shirt Table

tshirt	
id	size
1	S
2	M
3	L
4	XL

Colour Table

id	name
1	yellow
2	green
3	pink

Query
SELECT *
FROM tshirt
CROSS JOIN color;

Output

id	size	id	name
1	S	1	yellow
2	M	1	yellow
3	L	1	yellow
4	XL	1	yellow
1	S	2	green
2	M	2	green
3	L	2	green
4	XL	2	green
1	S	3	pink
2	M	3	pink
3	L	3	pink
4	XL	3	pink

CROSS JOIN – Example 2

Question - Join Match score table and department table


MatchScore Table

Player	Department_id	Goals
Franklin	1	2
Alan	1	3
Priyanka	2	2
Rajesh	3	5

Departments Table

Department_id	Department_name
1	IT
2	HR
3	Marketing

Query
SELECT *
FROM MatchScore
CROSS JOIN Departments;



Output

Player	Department_id	Goals	Department_id	Department_name
Franklin	1	2	1	IT
Alan	1	3	1	IT
Priyanka	2	2	1	IT
Rajesh	3	5	1	IT
Franklin	1	2	2	HR
Alan	1	3	2	HR
Priyanka	2	2	2	HR
Rajesh	3	5	2	HR
Franklin	1	2	3	Marketing
Alan	1	3	3	Marketing
Priyanka	2	2	3	Marketing
Rajesh	3	5	3	Marketing

CROSS JOIN – Example 3

Question - Find t-shirt size,color and fabric for each id

Color Table

id	name
1	blue
3	pink

tshirt Table

id	size
1	S
2	M
3	L

Fabric Table

id	name
1	cotton
2	linen

Query

```
SELECT tshirt.size as size, color.name AS color, fabric.name as fabric  
FROM tshirt  
CROSS JOIN fabric  
CROSS JOIN color ;
```

CROSS JOIN – Example 3

Output

size	color	fabric
S	blue	cotton
M	blue	cotton
L	blue	cotton
S	pink	cotton
M	pink	cotton
L	pink	cotton
S	blue	linen
M	blue	linen
L	blue	linen
S	pink	linen
M	pink	linen
L	pink	linen

Union

The UNION operator combines the results of two or more SELECT statements while also removing duplicate rows.

DISTINCT is applied to the result set before the outcome of the UNION operation, which removes duplicate rows and displays only the relevant data to the user.

Structure of a Union Query

```
SELECT * FROM table A
```

```
UNION
```

```
SELECT * FROM table B;
```

- The number of columns should be equal in both table A and table B.

Union ALL

The UNION ALL operator combines the results of two or more SELECT statements, but it does not perform any additional operations on the resultset.

Instead, it returns the entire table by concatenating the results of SELECT statements and displaying them to the user as is. It is slightly faster than the UNION operator because no extra overhead of removing duplicates is required.

Structure of a Union ALL Query

```
SELECT * FROM table A
```

```
UNION ALL
```

```
SELECT * FROM table B;
```

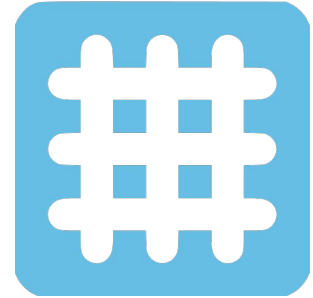
- The number of columns should be equal in both table A and table B.

Conditions for performing Union and UNION ALL

- The following conditions must be met to perform the UNION and UNION ALL operations for combining the tables:
- Each SELECT statement to be combined should return the same number of columns.
- Each SELECT statements' columns must be of the same data type.
- Columns returned by each SELECT statement must be returned in the same order.



Union Vs UNION ALL



1. The UNION and UNION ALL operators join two tables by combining the rows returned by two or more SELECT statements. The only distinction is that UNION does not replace or remove duplicate rows.
2. The UNION operator removes duplicate rows from the resultset by performing the DISTINCT operation. As a result, when there is a large volume of data in the tables, the UNION operator is considered slower than the UNION ALL operator because it slows down the overall speed by performing the additional overhead of the DISTINCT operation.

Example of Union

Question - Write a query to fetch names and id of all employees from different departments.

IT Department

emp_id	emp_name
E_001	Arush
E_002	Akash
E_003	Amitansh
E_004	Gourang
E_005	Manish

Operations Department

emp_id	emp_name
E_006	Akansha
E_007	Atul
E_008	Amrish
E_004	Gourang
E_009	Akshat

```
SELECT * from IT Department  
UNION  
SELECT * from Operations Department;
```

emp_id	emp_name
E_006	Akansha
E_007	Atul
E_008	Amrish
E_009	Akshat
E_002	Akash
E_004	Gourang
E_001	Arush
E_003	Amitansh
E_005	Manish

Example of Union

Question- Write a query to fetch names of all employees and id systematically.

IT Department

emp_id	emp_name
E_001	Arush
E_002	Akash
E_003	Amitansh
E_004	Gourang
E_005	Manish

Operations Department

emp_id	emp_name
E_006	Akansha
E_007	Atul
E_008	Amrisha
E_004	Gourang
E_009	Akshat

```
SELECT * from IT Department  
UNION ALL  
SELECT * from Operations Department;
```



emp_id	emp_name
E_001	Arush
E_002	Akash
E_003	Amitansh
E_004	Gourang
E_005	Manish
E_006	Akansha
E_007	Atul
E_008	Amrisha
E_004	Gourang
E_009	Akshat

Practice Question - 6

Write a query to combine the suppliers table(where supplier_id is greater than 2000) with companies table(where company id is greater than 1000).

Supplier

supplier_id	supplier_name
1000	Microsoft
2000	Oracle
3000	Apple
4000	Samsung

Supplier

company_id	company_name
1000	Microsoft
3000	Apple
7000	Sony
8000	IBM

Solution - 6

```
SELECT
    supplier_id AS ID_Value,
    supplier_name AS Name_Value
FROM
    suppliers
WHERE
    supplier_id > 2000
```

UNION

```
SELECT
    company_id AS ID_Value,
    company_name AS Name_Value
FROM
    companies
WHERE
    company_id > 1000
ORDER BY 1;
```

Note :

Here, order by 1 implies first column of suppliers table.

Solution - 6

Output

ID_Value	Name_Value
3000	Apple
4000	Samsung
7000	Sony
8000	IBM

Conclusion

In the next class we will study:



Problem Solving on Joins