- Join Clause (Retrieving Data From multiple tables)

. In SQL the JOIN Clause is used to combine rows from two or more tables based on a related column between them

. There are several types of joins in SQL, including inner join, left join, right join, and full outer join and more.

↳ Inner Join

An inner join return only the rows that have matching values in both tables being joined. It uses the matching column(s) to join two table together.

example:

Suppose we have two tables:

employees [employee_id, name, department_id]

departments [department_id, department_name]

We want to join these tables on the department ID column to get a list of all employees and their corresponding department names.

```
SELECT e.name,
       d.department_name
FROM employees e
JOIN departments d
    ON e.department_id = d.department_id
```

=> output: name (from emp table), department_name (from depart table)

- An inner join between two tables will return only records where a joining field such as Key, find a match in both tables.

left_table

id	left_val
1	L1
2	L2
3	L3
4	L4

right_table

id	right_val
1	R1
4	R4
5	R5
6	R6

result after INNER JOIN

id	left_val	right_val
1	L1	R1
4	L4	R4

-- we can also join tables in different databases.

- For example if we want to join the "product" table located in "sql_inventory" database, with the "order_item" table located in "sql_store" database

```
USE sql_store
FROM SELECT *
FROM   sql_inventory.products p
JOIN   order_items           oi
    ON   oi.product_id       p.product_id
```

- in this case we only have to prefix tables that are not in the current (selected) database.

-- The Using Clause

Is used in a JOIN statement to specify the columns used to join 2 or more tables. It's an alternative to the ON clause.

With the USING clause, columns used to join tables must have the same name and data type in both tables.

Self Join

. A self join is a special type of join where a table is joined with itself.

This can be useful when you have a table that cointain hierarchial data or when you want to compare data within the same table.

example:

Suppose we have a table "employees":

Employees [employee_id, name, manager_id]

The manager_id column contains the ID of the employee's manager.

we want to join the "employees" table with itself to get a list of all employees and their corresponding manager name.

```
SELECT e.name AS employee_name,
       m.name AS employee_manager
FROM employees e
JOIN employees m
    ON m.employee_id = e.manager_id
```

=> output: employee_name, manager.

-- joining multiple table (more than 2)

. we can also join more than 2 tables.

For example in the sql_store database we have the tables: Orders [order_id, customer_id, date, status]

Customers [customer_id, name, birth_date]

Order_Statuses [order_status_id, name]

Suppose we want to join these 3 table such that we get a report that cOntain:

[Order_id, customer_name, order_name, status]

```
USE sql_store
SELECT o.order_id ,
       c.name AS customer_name,
       o.date AS order_date,
       os.name AS status
FROM orders o
JOIN customers c USING (customer_id)
JOIN order_statuses os
     ON o.status = os.order_status_id.
```

-- Compound Join Conditions

Compound Join conditions refer to using more than one condition to join two or more tables together
It can be used to join a table that have composed primary Key

Example:

Suppos we have two tables:

Orders [order_id , order_date , customer_id]

Order_Items [order_id, item_id , item_name , item_price]

Order_item_notes [note_id , order_id , product_id , note]

Left Join / Right Join (Outer Join)

A left join returns all the rows from the left table and the matching rows from the right table. If there no matching rows in the right table, the result will show NULL value for the right table columns.

left table

id	left_val
1	L1
2	L2
3	L3
4	L4

right table

id	Right_val
1	R1
4	R2
5	R3
6	R4

result after LEFT JOIN

id	left_val	right_val
1	L1	R1
2	L2	null
3	L3	null
4	L4	R2

result after RIGHT JOIN

id	left_val	right_val
1	L1	R1
4	L4	R2
5	null	R3
6	null	R4

A right join returns all the rows from the right table and the matching rows from the left table. If there no matching rows in the left table, the result will show NULL value for the left table columns.

Examples

-- left join:

Suppose we have two tables:

Employees [employee_id, name, department_id]

Departments [department_id, department_name]

Suppose we want to get a list of all employees and their department name even if they don't have a department.

```
SELECT e.name, d.department_name
FROM employees e
LEFT JOIN departments d
  USING (department_id)
```

-- Right join:

Continuing with our "employees" and "departments" tables. Suppose we want to get a list of all department and employees in this department, even a department dosn't have employees

```
SELECT e.name, d.department_name
FROM employees e
RIGHT JOIN departments d
    USING (department_id);
```

↳ **Full outer Join**

A full outer join returns all the rows from both tables, including matching and non-matching rows If there are no matching rows in either table, the result will show NULL value for the columns from that table.

Example:

Suppose we want to get a list of all employees and their corresponding department name, including any department with no matching employees

```
SELECT e.name, d.department_name
FROM employees e
FULL OUTER JOIN departments d
ON e.department_id = d.department_id;
```

LEFT TABLE

id	left_val
1	L1
2	L2
3	L3
4	L4

RIGHT TABLE

id	right_val
1	R1
4	R2
5	R3
6	R4

result after FULL OUTER Join

id	left_val	Right_val
1	L1	R1
2	L2	null
3	L3	null
4	L4	R2
5	null	R3
6	null	R4

↳ Cross Join (Cartesian Product)

A cross join, is a type of join in SQL where each row from one table is joined with every row from another table. (creates all possible combination)

Example:

Suppose we have two tables "color" and sizes", and we want to create a new table that list every possible combination of colors and sizes.

```
SELECT c. color_name , s. size_name
FROM  colors c
CROSS JOIN sizes s;
```

id1	id2
1	A
1	B
1	C
2	A
2	B
2	C