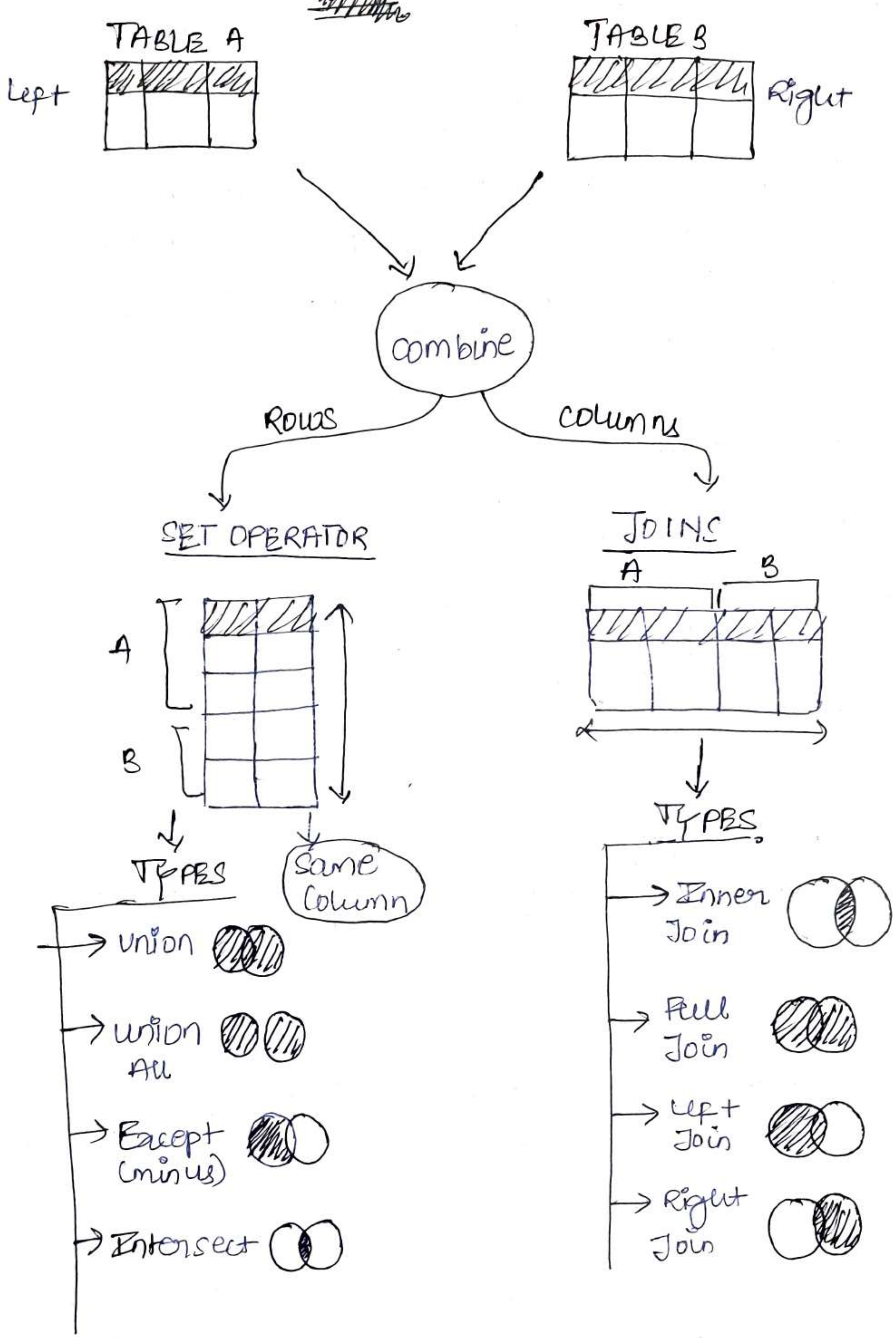


TABLE A
Left
TABLE B
Right
combine
Rows
columns
SET OPERATOR
A
B
Types
Same Column
Union
Union All
Except (minus)
Intersect
JOINS
A
B
Types
Inner Join
Full Join
Left Join
Right Join

<u>what is SQL Joins</u>?

JOIN in SQL is used to combine Rows from two or more tables based on a related column b/w them. Helps in Retrieving Meaningful information from Relational databases by connecting data stored in separate tables.

<u>when to use SQL</u>?

1) <u>Recombine</u> <u>Data</u>.

Customers　　　　Address

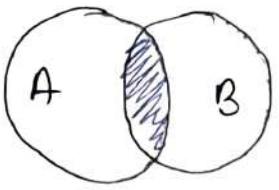
$$\left.\begin{array}{l}\\ \\ \end{array}\right\} \rightarrow \underline{\text{combine}}$$

Reviews　　　　orders

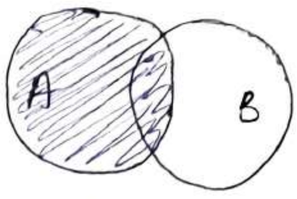
2) <u>Data</u> <u>Enrichment</u> "Getting Extra Data"

3) <u>Check</u> for <u>Existence</u> "Filtering"

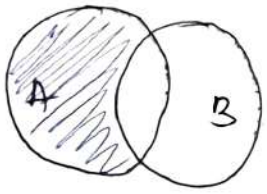
JOIN TYPES



Matching Data



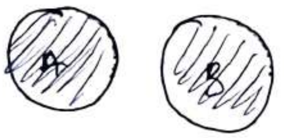
All Data



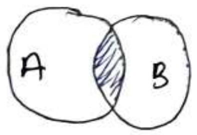
unMatching Data

X —— X —— X —— X —— X —— X —

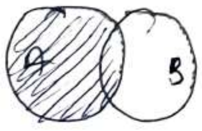
Basics



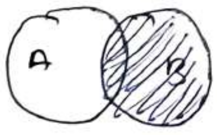
NO Join



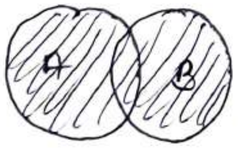
Inner Join



Left Join

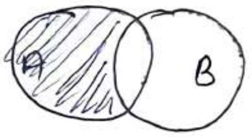


Right Join

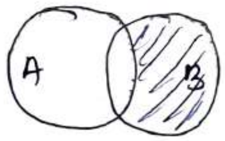


Full Join

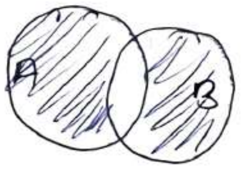
Advanced



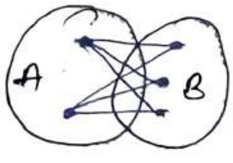
Left anti join



Right anti join



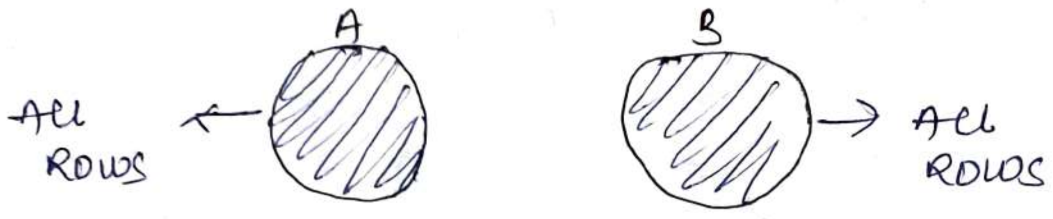
Full anti join



cross join

NO JOINS

Return Data from Tables without combining Them:



Two Results No Need to combine.

Syntax

```
SELECT *
  FROM A


SELECT *
    FROM B.
```

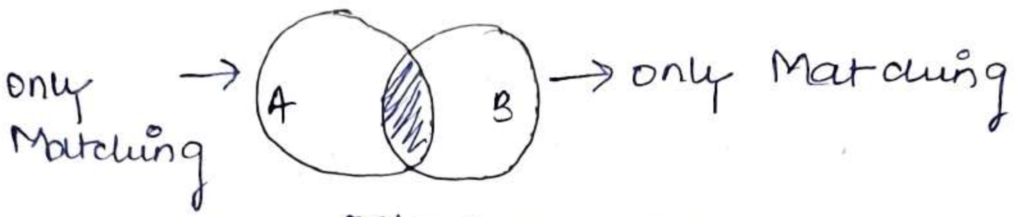
Ex

```
SELECT *
   FROM customers;

SELECT *
    FROM orders;
```

INNER JOIN

Returns only Matching Rows from Both Tables.



only common Data.

Syntax

```
SELECT *
  FROM A            INNER
~~[illegible]~~ [TYPE] JOIN B
       ON <condition>
```

Default inner join

$\underbrace{\text{condition}}_{\text{A.key = B.key}}$

• Order of Table Doesn't Matters.

Ex:- Get all customer along with their order, but only for customer who have placed an order.

```
SELECT *
  FROM customers
  INNER JOIN orders
  ON id = customer_id
```

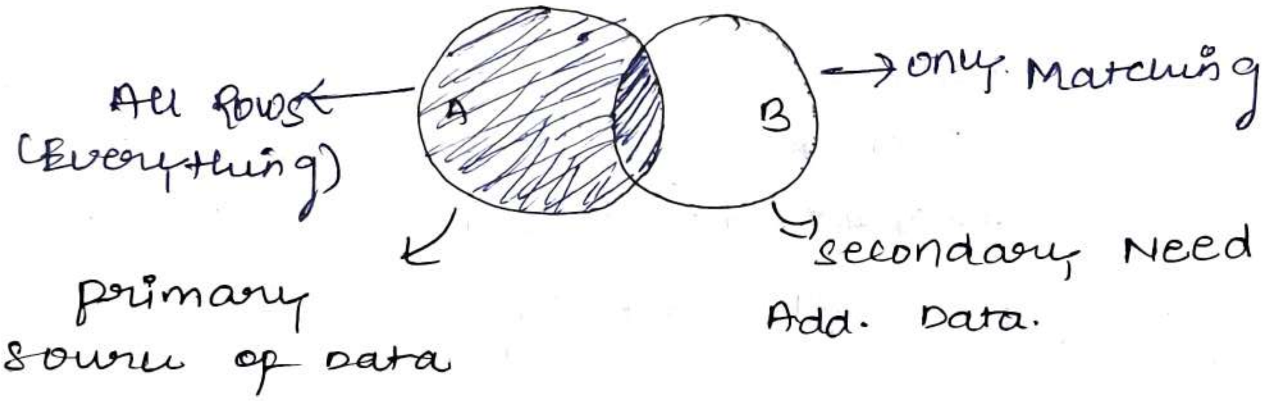
If we have same columns in both Table we use.

AS c	AS O
c. customer	o. id
c. id	o. sales

INNER JOIN uses for → Recombine data
→ filtering / check Existance.

#. LEFT JOIN.

Returns All rows from left and only Matching from Right.



```
SELECT *
FROM A          ← Left
LEFT JOIN B     → Right.
ON A.Key = B.KEY.
```

*order of Tables is important.

Ex: Get all customers along with their orders, including those without order

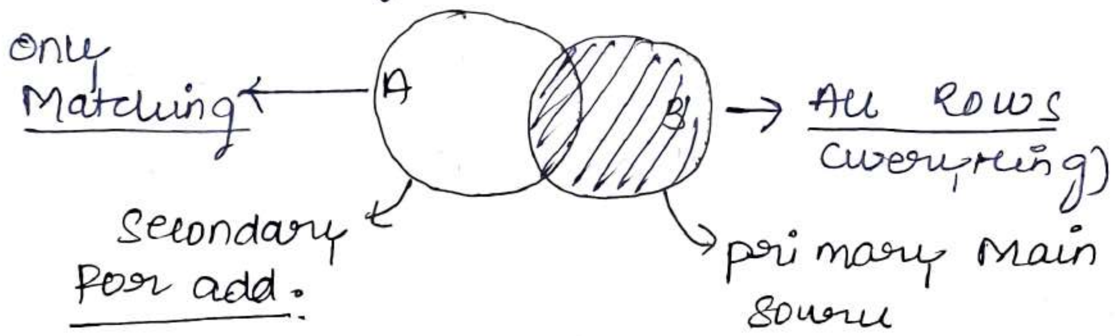
```
SELECT
        C.Id,
        C.first_name,
        O.order_id,
        O.sales
```

```
FROM customers AS C
LEFT JOIN ~~AS~~ orders AS O
ON c.id = o.customer_id.
```

Left join use { Recombine Data
              { Data Enrichment.

RIGT JOIN

Returns All Rows from Right and only Matching from left.



```
SELECT *
  FROM A
RIGT JOIN B
ON A.key = B.key
```

* ORDER of Tables in important.

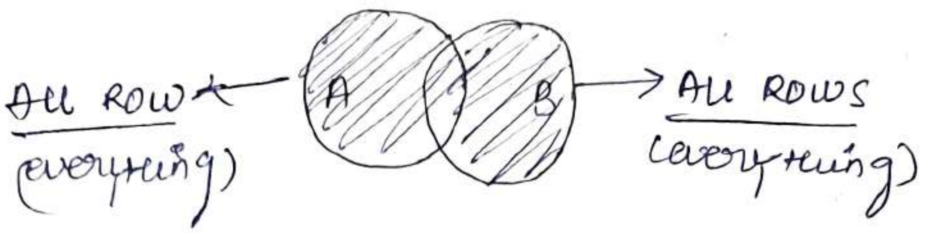
Ex: Get all customers along with their order including order without Matching customer.

```
SELECT
        c.id,
        c.first_name,
        o.order_id,
        o.sales.
    FROM customer AS c
    RIGHT JOIN orders AS O
    ON c.id = o.customer_id.
```

⇓ Alternative

```
SELECT
        c.id,
        c.first_name,
        o.order_id,
        o.sales
    FROM orders AS O
    LEFT JOIN customers AS c
    ON c.id = o.customer_id.
```

FULL JOIN



- Returns All Rows From Both Tables.

```
SELECT *
  FROM A
FULL JOIN B
ON A.Key = B.Key
```

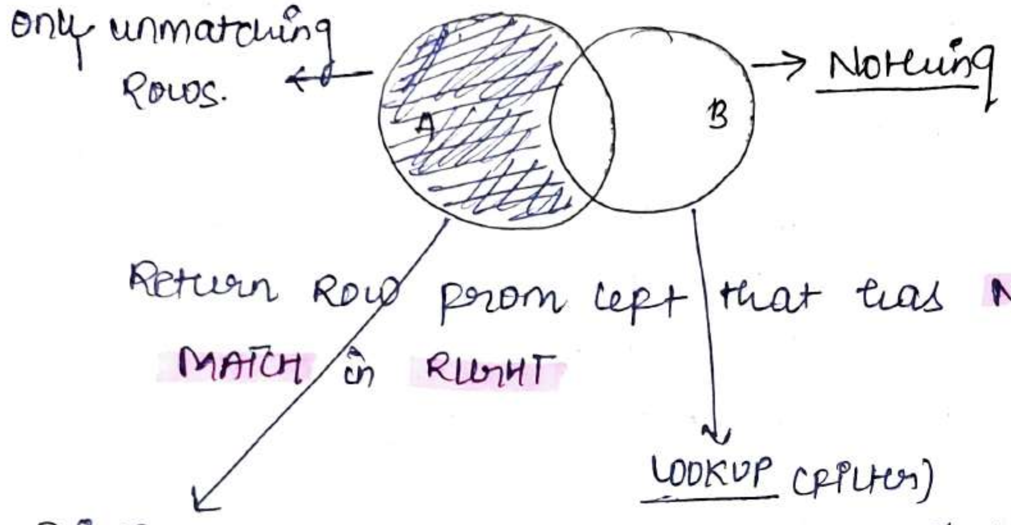
- Order of Tables does not Matter.

Ex: Crete all customer and order, even if there is no match.

Full Join + Where use { Recombine / Filter check existence.

ADVANCED JOIN.

Left Anti Join



Return Row from left that has NO MATCH in RIGHT

Primary
Main Source of Data.

LOOKUP (Filter)
Not for Data, Just for Filter.

Syntax.

The order of Tables Is important.

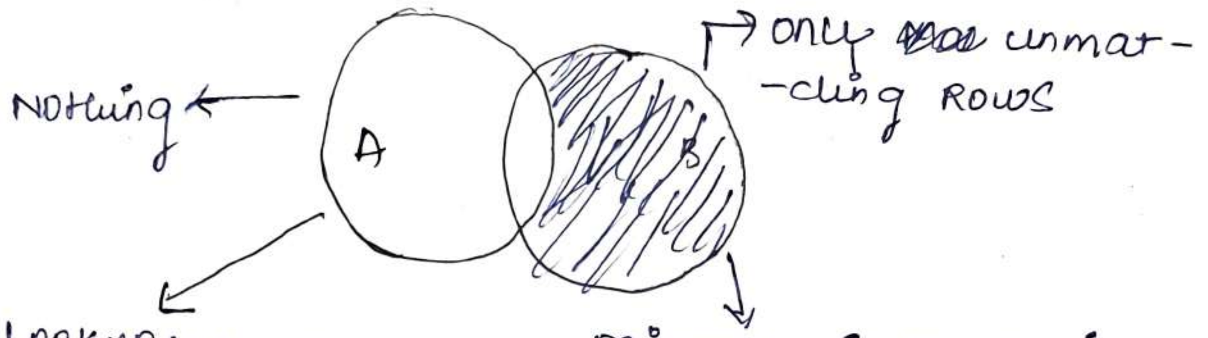
```
SELECT *
    FROM A
    LEFT JOIN B
   ON A.Key = B.KEY
   WHERE B.Key Is NULL
```

Ex-
```
SELECT *
 FROM customers AS C
LEFT JOIN orders AS O
ON c.id = O.customer_id
WHERE O.customer_id IS NULL.
```

LEFT anti join use to check existance.

RIGHT ANTI JOIN

Returns Rows From Right that has NO MATCH in left.



LOOKUP (Filter)
NO Data only For filter.

primary source(Main) of Data.

Syntax.

```
SELECT *
 FROM A
RIGHT JOIN B
ON A.Key = B.Key
WHERE A.key IS NULL.
```

→ A is left/ NO Data

• Order of Tables Must important.

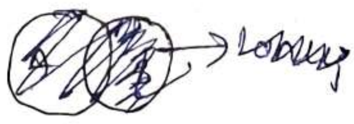
Ex.

```
SELECT *
     FROM customers AS C
RIGHT JOIN orders AS O
ON    C.id = O.customer_id
WHERE C.id IS NULL.
```

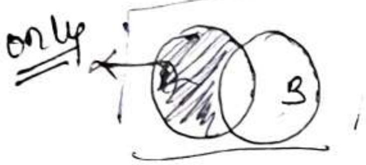
(customers = A, orders = B)

⇓ alternative way. (LEFT JOIN)

LEFT JOIN (Method)

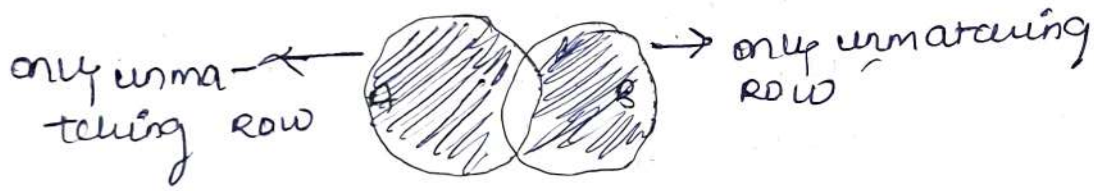


```
SELECT *
          A
     FROM orders AS O
                  B
LEFT JOIN customers AS C
ON   O.customer_id = C.id
WHERE ~~o.customer_id~~ IS NULL.
              C.id
```



FULL ANTI JOIN

- Returns only ROWS that Don't Match in Either Tables.



only unmatching Row.

Syntax.

```
SELECT *
 FROM A
FULL JOIN B
ON A.Key = B.Key
WHERE
     B.Key IS NULL
     OR
     A.Key IS NULL
```

Note: Tables of order does not Matter

Eg:- [C.id Means order hai but without customer wala order].

```
SELECT *
  FROM customers AS C
  FULL JOIN orders AS O
  ON C.id = O.customer_id
  WHERE C.id IS NULL OR O.customer_id
    IS NULL
```

[C.id IS NULL = order without customer.
O.customer_id IS NULL = cust. without orders]

Another Ques.

· Get all customer along with their orders, but only for customers who have placed an order without using (INNER JOIN)

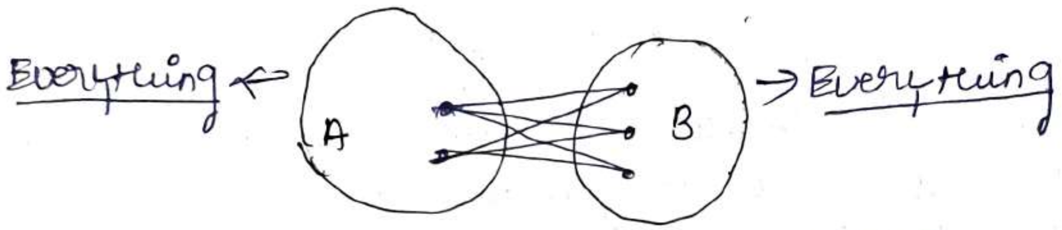
```
SELECT *
FROM customes AS C
LEFT JOIN orders AS O
ON C.id = O.customer_id
WHERE O.customer_id IS NOT NULL
```

Sirf wahi row chahiye jaha orders side me customer_id NULL nahi hai

CROSS JOIN

Combines Every Row from left with Every Row from Right.

All possible combinations - cartesian Join.



2x3 = 6 → Total Rows.

Syntax.

```
SELECT *
FROM A
CROSS JOIN B
```

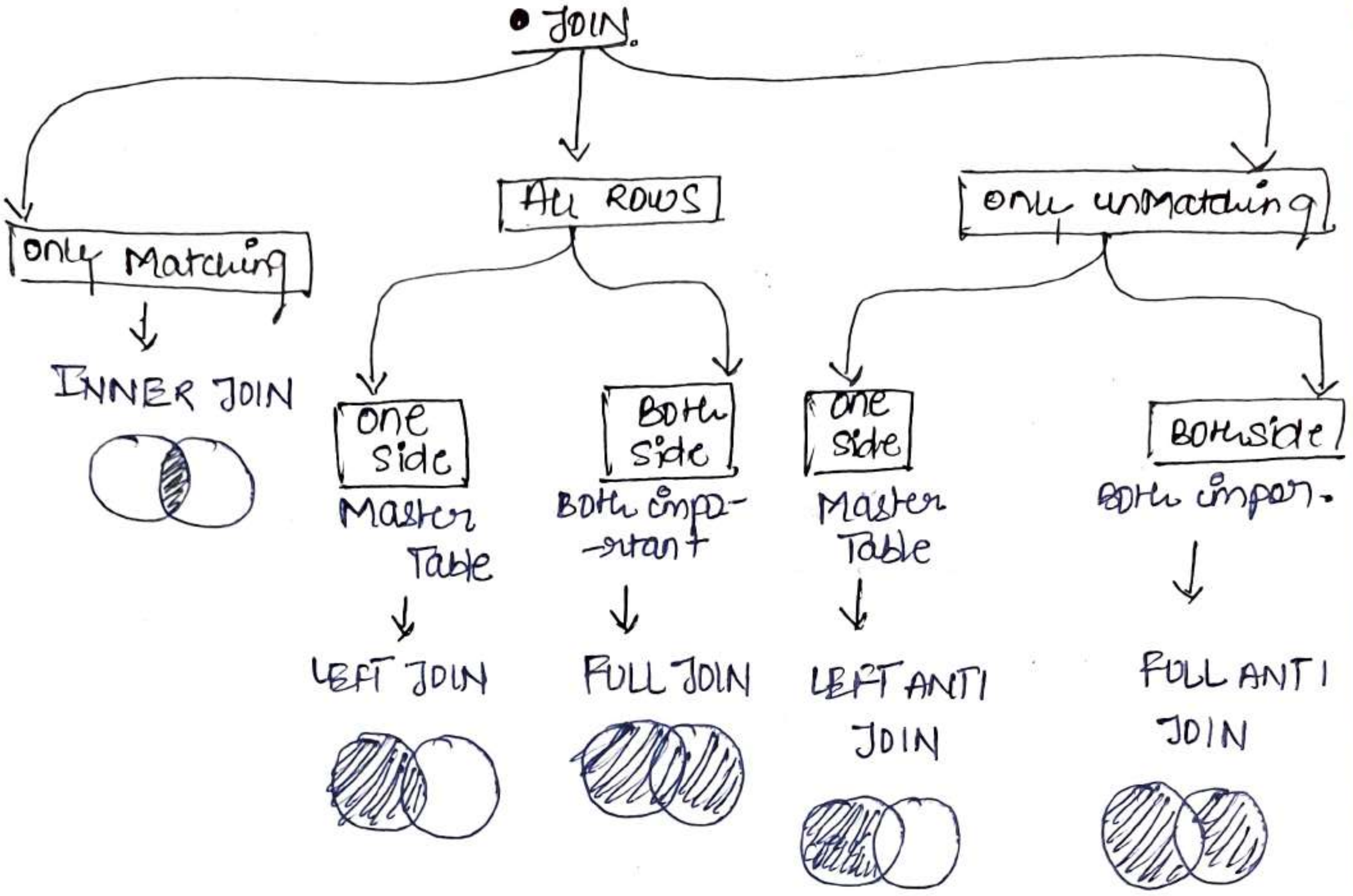
↙

No condition Needed.

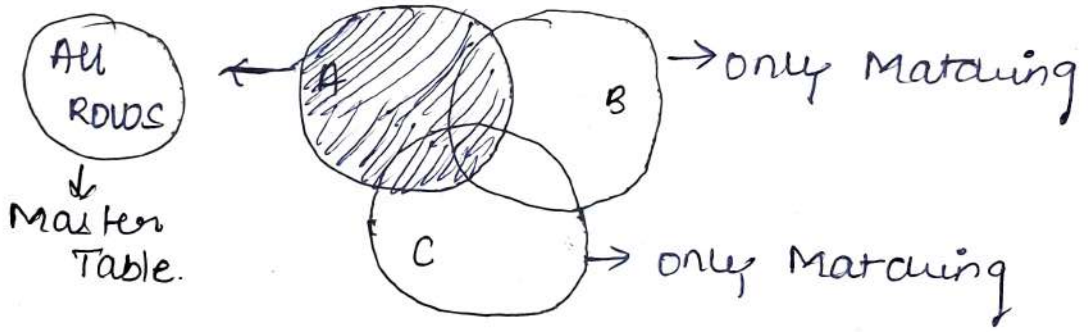
Rule: Order of Table does not Matter.

Eg: Generate all possible combination of customers and orders

```
SELECT *
FROM A
CROSS JOIN B
```

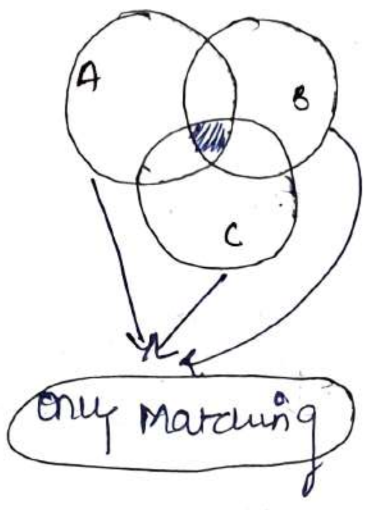


Multiple Join Table advance.



```
SELECT *
  FROM A
 LEFT B ON ...
 LEFT C ON ...
 WHERE (control what
                to keep.)
```

#overlapping Data.



Syntax

```
SELECT *
  FROM A
INNER B --
INNER C ---.
```

E.g) using sales DB, Retrieve list of all order, along with the Related customer, product and employee details.

sol For each order, Display.

- order ID.
- customer name.
- product name.
- Sales amount.
- product price.
- Sales person's name.

```
SELECT
        o. orderID,
        o. sales * p. price AS sales Amount,
    c. ~~e~~firstname AS customerfirstname,
    c. Lastname AS customerLastname,
    P. product AS productName,
    P. price AS productprice,
    e. firstname AS Employeefirstname,
    e. Lastname AS Employeelastname,
FROM sales.orders AS o
LEFT JOIN sales. customers AS c
ON o. customerID = c. customerID
LEFT JOIN sales. product AS P
ON o. productID = p. productID
LEFT JOIN sales. employees AS e
ON o. salespersonID = e. employeeID.
```