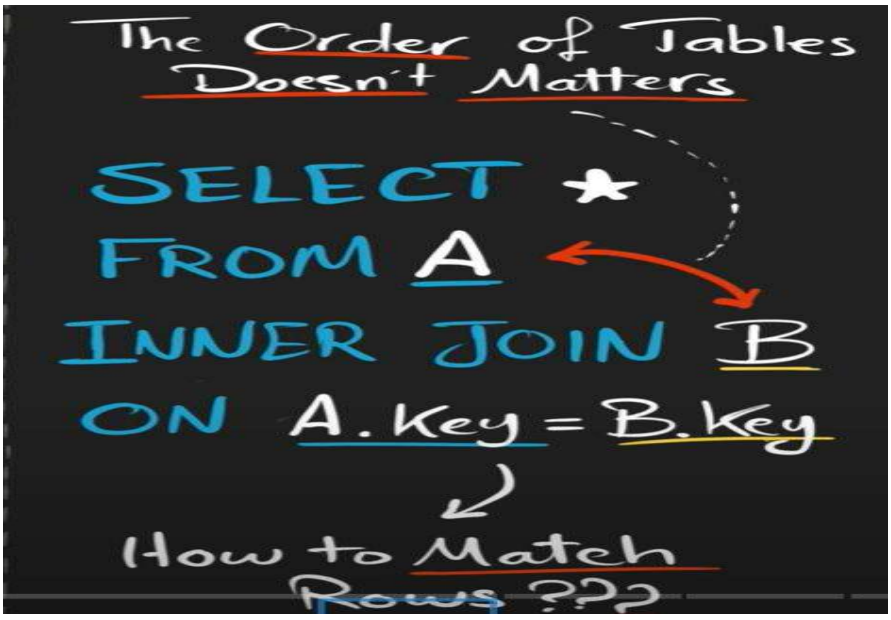
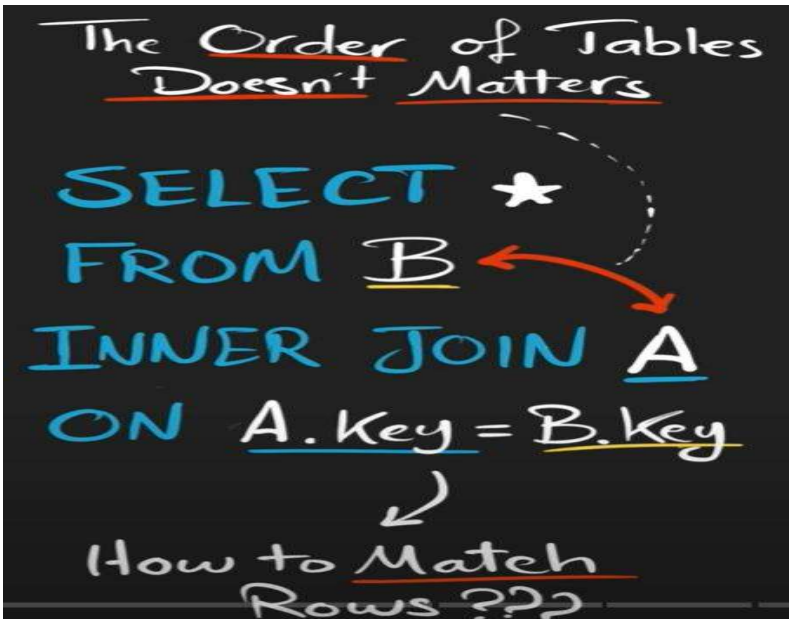
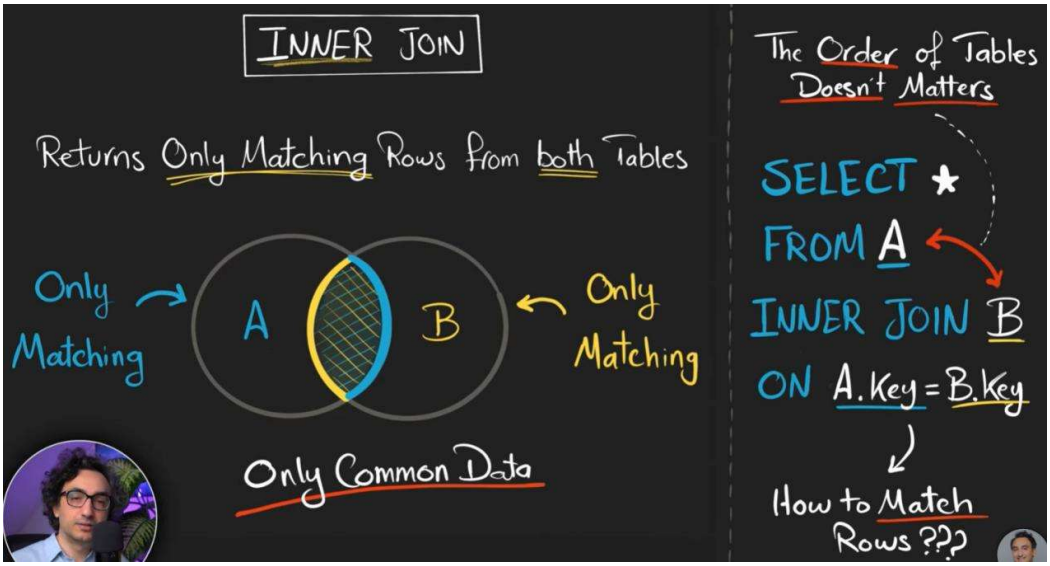
How to deal with rows between two tables sql know by using condition Inner Join.
ON Keyword and Join condition tells how to combine the tables.
In order to join two tables we have to find out common column in order to match the data and usually in sql there are the keys and ID's so condition can be like that the key from the table a must be equal to the key from the table b so this is join condition using the join condition sql can start matching the data from the left table and the right table.

One more important thing while you are joining the table you have to undersatand the order of the table in your query.

But in inner join the order of the join doesn't really matter.
So wheather you starts from A or starts from B it does not matter because you will get the same result both of the table has same priority and it does not matter from which table you start.wheather we say A join B or we can say B join A

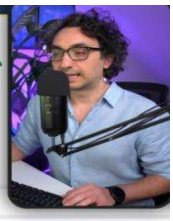






This syntax works only in Microsoft DB but it is a wrong practice may cause error in MYSQL

```sql
/* Get all customers along with their orders,
but only for customers who have placed an order

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



Results Messages

	id	first_name	country	score	order_id	customer_id	order_date	sales
1	1	Maria	Germany	350	1001	1	2021-01-11	35
2	2	John	USA	900	1002	2	2021-04-05	15
3	3	Georg	UK	750	1003	3	2021-06-18	20

Error may Occur:

-- MySQL sees CustomerID and gets confused:
-- "Which table's CustomerID are you talking about?"
-- That’s what “ambiguous” means — it’s unclear which column you mean.

Correct way to write it

```sql
select * from customers c
Inner join  Orders o On c.customerid = o.OrderID;
```

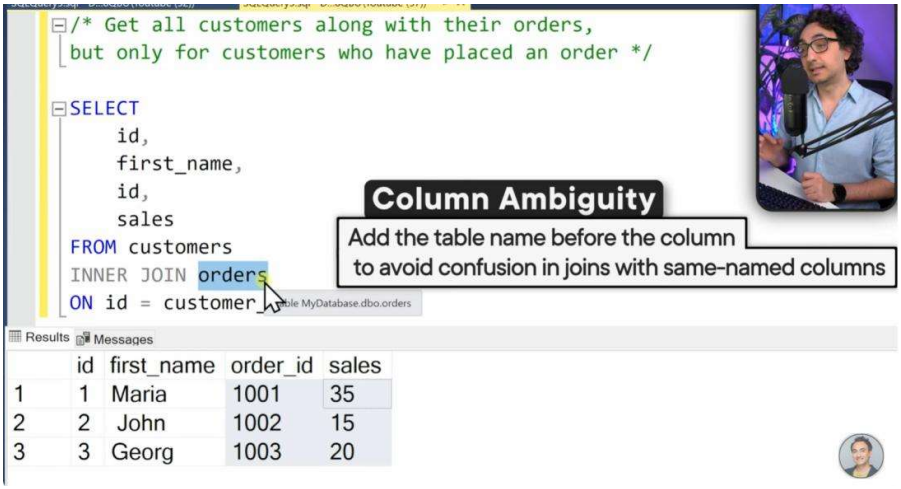
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

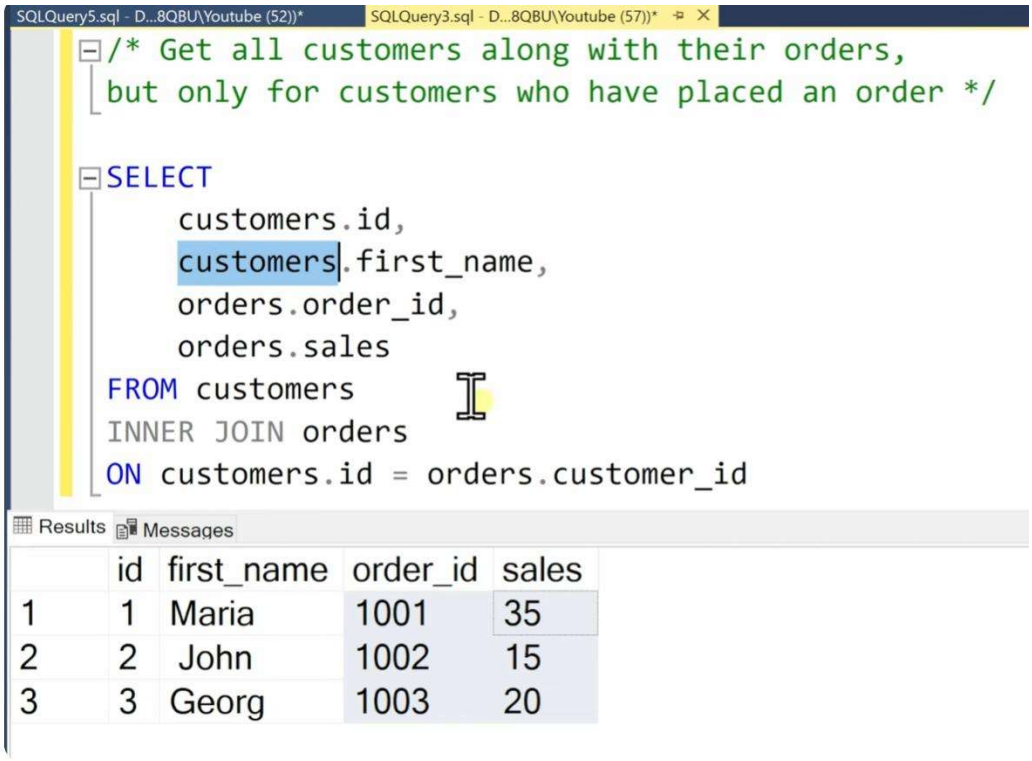
CustomerID	FirstName	LastName	Country	Score	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate	Ship
1	Shiya	Goldberg	Chhatarpur	350	1	101	2	3	2025-01-01	2025
2	Kevin	Brown	USA	900	2	102	3	3	2025-01-05	2025
3	Shivam	Chaurasia	India	450	3	101	1	5	2025-01-10	2025
4	Alice	Smith	USA	720	4	105	1	3	2025-01-20	2025
5	Bob	Johnson	Canada	680	5	104	2	5	2025-02-01	2025
6	Charlie	Lee	UK	750	6	104	3	5	2025-02-05	2025
7	priya	kapoor	NULL	NULL	7	102	1	1	2025-02-15	2025
8	kiya	singh	NULL	NULL	8	101	4	3	2025-02-18	2025

```sql
/* Get all customers along with their orders,
but only for customers who have placed an order */

SELECT
    id,
    first_name,
    order_id,
    sales
FROM customers
INNER JOIN orders
ON id = customer_id
```

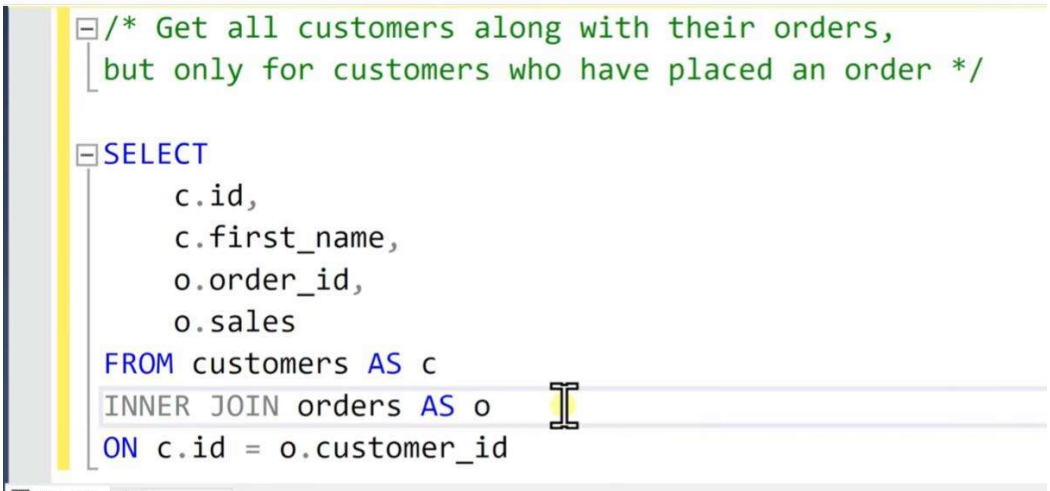
Results | Messages

	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20





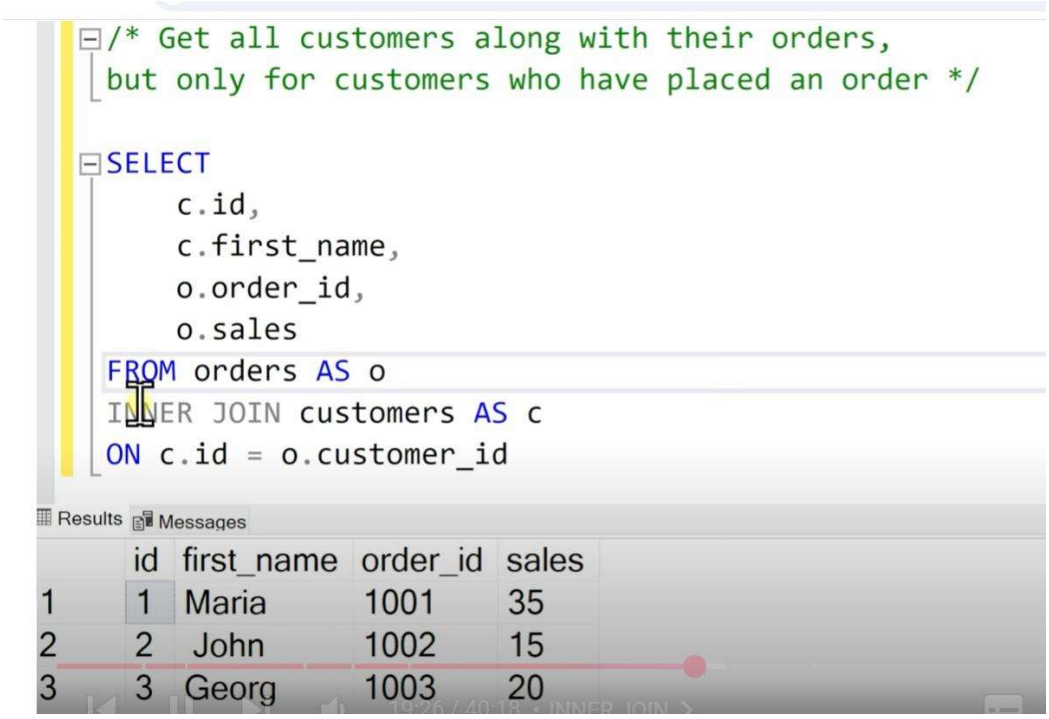
	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20



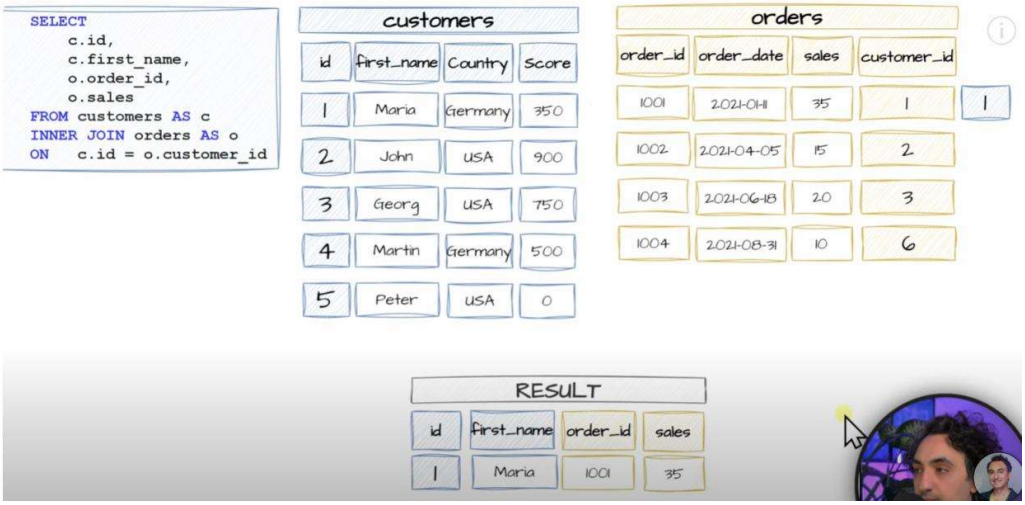
	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20

As due to inner join No matter order of table so we can keep any order as in above example or in below example.

If there is a match then data add into the result.



How the query being executed
SQL go and check each id in all rows wheather customer id of customer table is matching to the customer id of order table



No match in any row of cutomer id -1 except first row sql check in each row for each id match

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

customers

id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

orders

order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

RESULT

id	first_name	order_id	sales
1	Maria	1001	35


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

customers

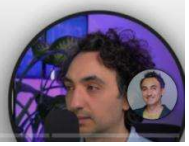
id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

orders

order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

RESULT

id	first_name	order_id	sales
1	Maria	1001	35




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

customers

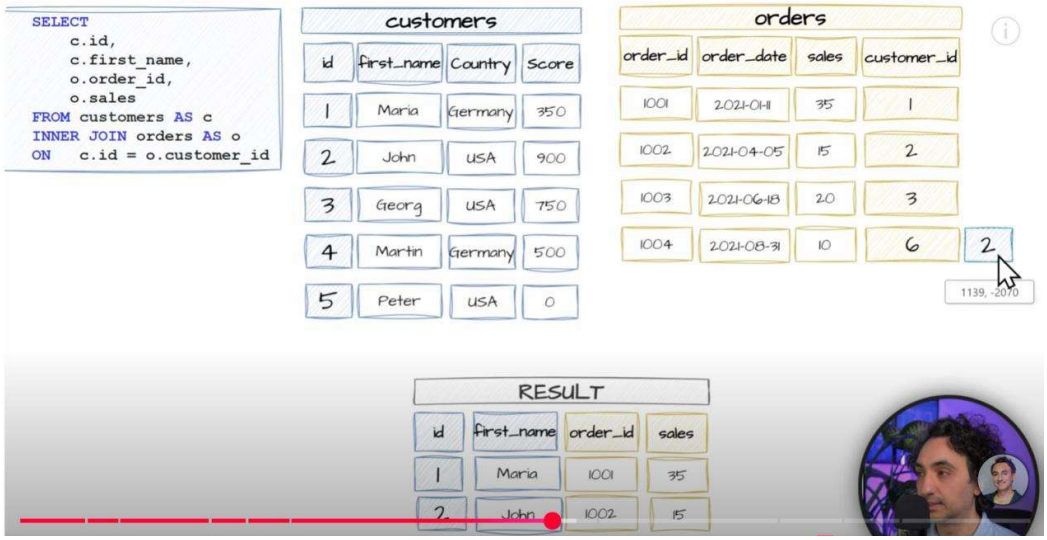
id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

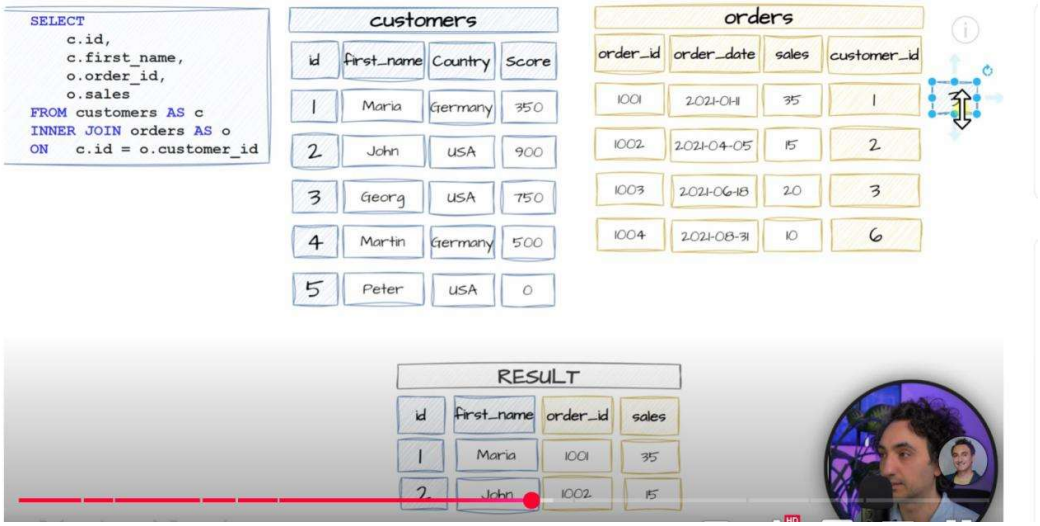
orders

order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

RESULT

id	first_name	order_id	sales
1	Maria	1001	35


SELECT
c.id,
c.first_name,
o.order_id,
o.sales
FROM customers AS c
INNER JOIN orders AS o
ON c.id = o.customer_id
customers
id | first_name | Country | Score
1 | Maria | Germany | 350
2 | John | USA | 900
3 | Georg | USA | 750
4 | Martin | Germany | 500
5 | Peter | USA | 0
orders
order_id | order_date | sales | customer_id
1001 | 2021-01-11 | 35 | 1
1002 | 2021-04-05 | 15 | 2
1003 | 2021-06-18 | 20 | 3
1004 | 2021-08-31 | 10 | 6
2
1139, -2070
RESULT
id | first_name | order_id | sales
1 | Maria | 1001 | 35
2 | John | 1002 | 15


SELECT
c.id,
c.first_name,
o.order_id,
o.sales
FROM customers AS c
INNER JOIN orders AS o
ON c.id = o.customer_id
customers
id | first_name | Country | Score
1 | Maria | Germany | 350
2 | John | USA | 900
3 | Georg | USA | 750
4 | Martin | Germany | 500
5 | Peter | USA | 0
orders
order_id | order_date | sales | customer_id
1001 | 2021-01-11 | 35 | 1
1002 | 2021-04-05 | 15 | 2
1003 | 2021-06-18 | 20 | 3
1004 | 2021-08-31 | 10 | 6
RESULT
id | first_name | order_id | sales
1 | Maria | 1001 | 35
2 | John | 1002 | 15

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

customers

id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

orders

order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

RESULT

id	first_name	order_id	sales
1	Maria	1001	35
2	John	1002	15
3	Georg	1003	20


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

customers

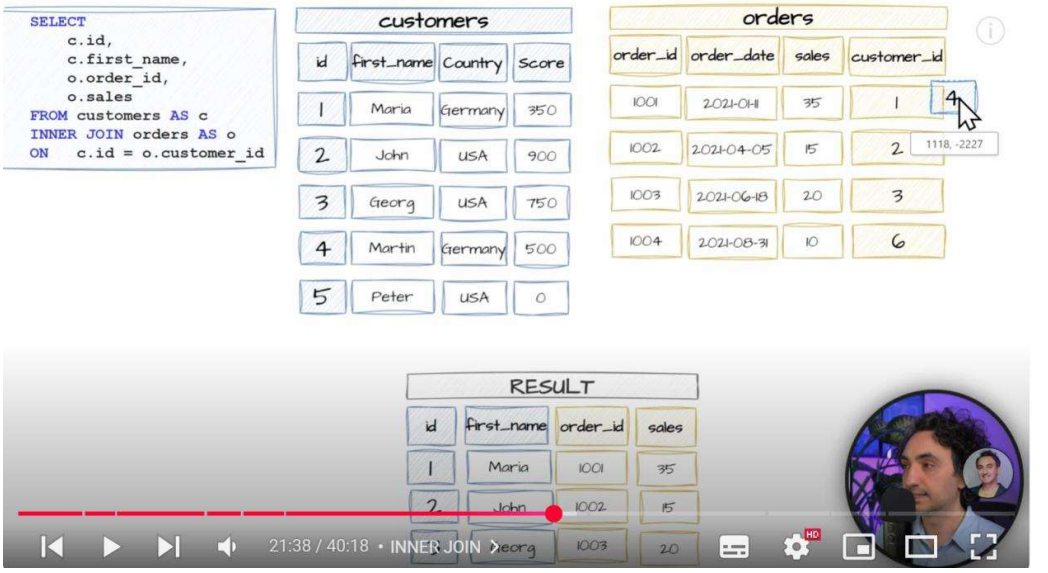
id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

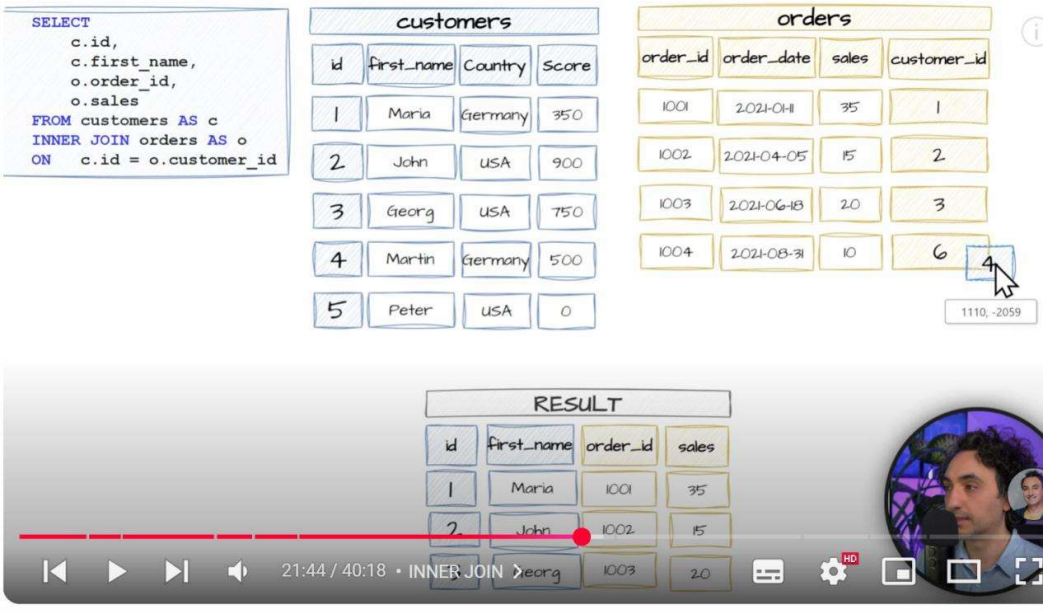
orders

order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

RESULT

id	first_name	order_id	sales
1	Maria	1001	35
2	John	1002	15
3	Georg	1003	20





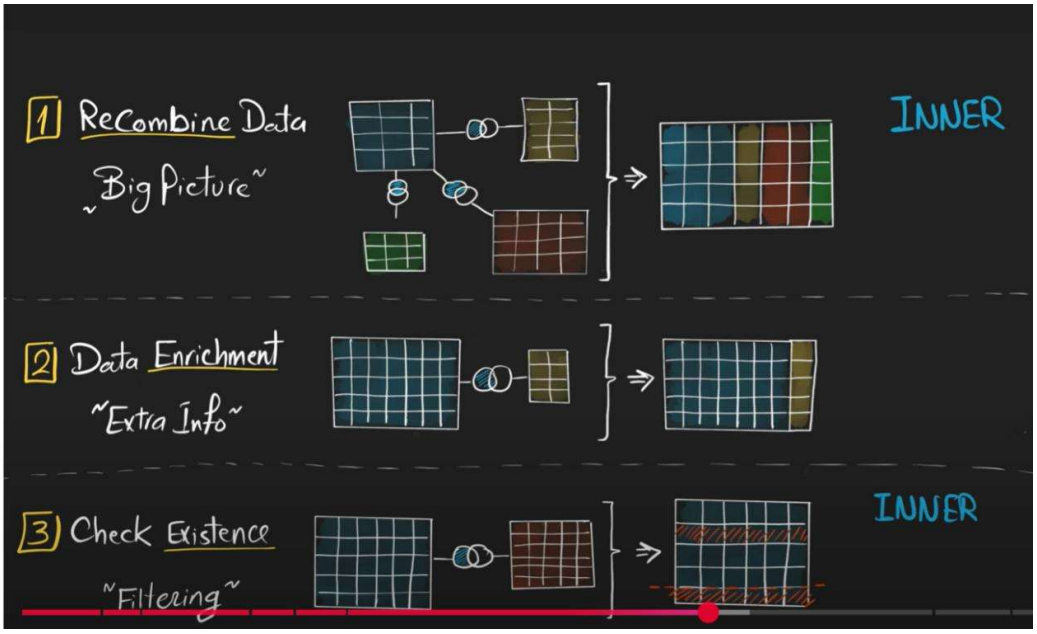
No match for customer 4 in order table - so the data of customer 4 will not be add in result because inner join does not allow to add unmatched data. Sql will ignore those cutomers having no match found


SELECT
c.id,
c.first_name,
o.order_id,
o.sales
FROM customers AS c
INNER JOIN orders AS o
ON c.id = o.customer_id
customers
id | first_name | Country | Score
1 | Maria | Germany | 350
2 | John | USA | 900
3 | Georg | USA | 750
4 | Martin | Germany | 500
5 | Peter | USA | 0
orders
order_id | order_date | sales | customer_id
1001 | 2021-01-11 | 35 | 1
1002 | 2021-04-05 | 15 | 2
1003 | 2021-06-18 | 20 | 3
1004 | 2021-08-31 | 10 | 6
5
1139, -2227
RESULT
id | first_name | order_id | sales
1 | Maria | 1001 | 35
2 | John | 1002 | 15
3 | Georg | 1003 | 20
22:01 / 40:18 • INNER JOIN


SELECT
c.id,
c.first_name,
o.order_id,
o.sales
FROM customers AS c
INNER JOIN orders AS o
ON c.id = o.customer_id
customers
id | first_name | Country | Score
1 | Maria | Germany | 350
2 | John | USA | 900
3 | Georg | USA | 750
4 | Martin | Germany | 500
5 | Peter | USA | 0
orders
order_id | order_date | sales | customer_id
1001 | 2021-01-11 | 35 | 1
1002 | 2021-04-05 | 15 | 2
1003 | 2021-06-18 | 20 | 3
1004 | 2021-08-31 | 10 | 6
5
1128, -2072
RESULT
id | first_name | order_id | sales
1 | Maria | 1001 | 35
2 | John | 1002 | 15
3 | Georg | 1003 | 20
22:04 / 40:18 • INNER JOIN

Sql starts on the left side data amd match it with right side data table

Inner Join --- used to recombine the data into a big picture or one table
It is used to filter the matching data - filtering the data we are checking the existance of records into another table.

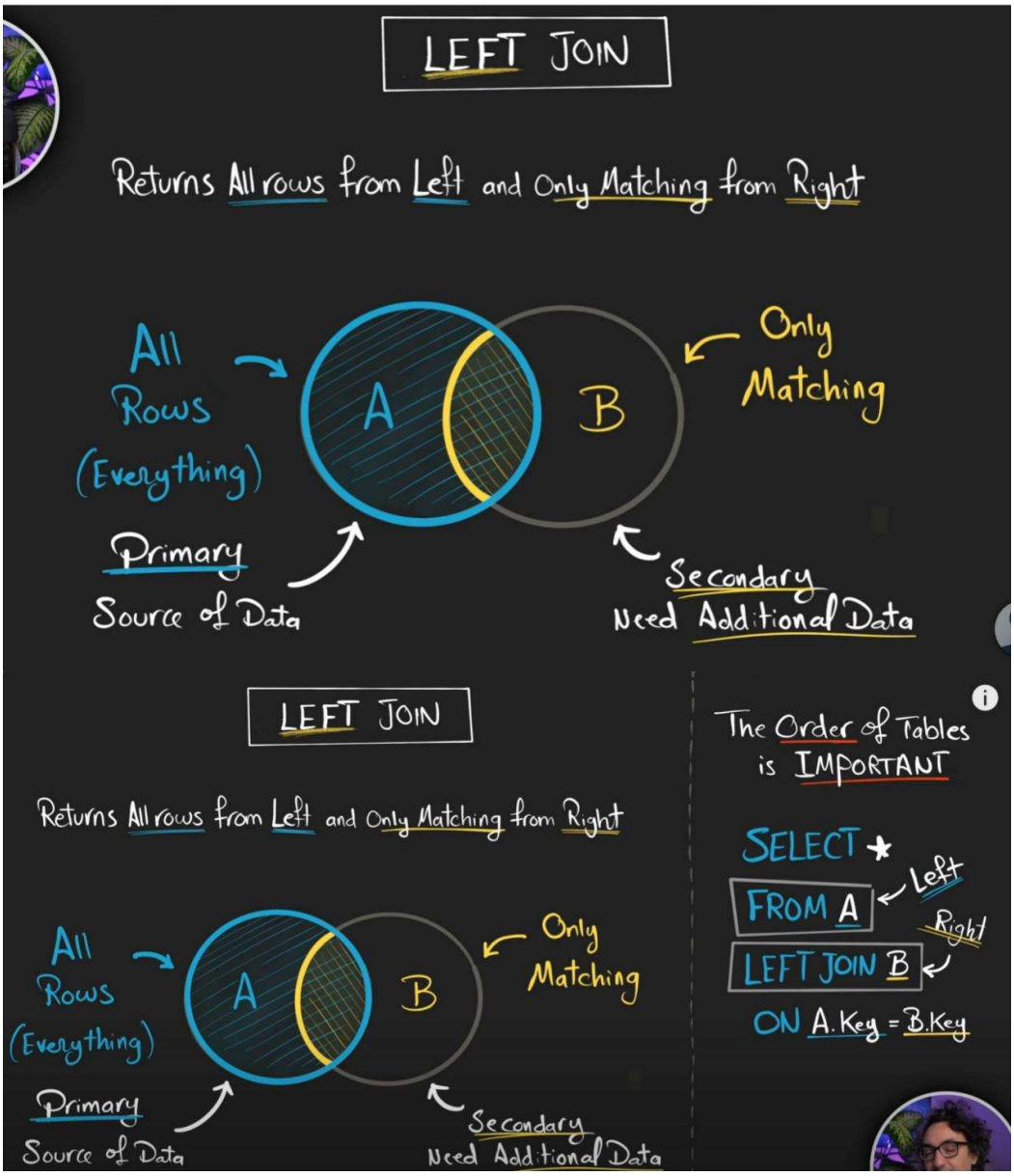


Left Join:

In Left join order of table is very important because we want all the data records from left table so no data can be missed from left table and

matching data from right table so here order of Joining a table matter you can not join table B to Table A Here you will not get require result

You will join Table A to Table B only to get an accurate result that's why order of table is very important.



```sql
/* Get all customers along with their orders,
   including those without orders. */

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

Results Messages

id	first_name	order_id	sales
1	Maria	1001	35
2	John	1002	15
3	Georg	1003	20
4	Martin	NULL	NULL
5	Peter	NULL	NULL

As order of Table is very important in left join As We are not getting accurate result.by chaging order of table.

```sql
/* Get all customers along with their orders,
   including those without orders. */

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
```

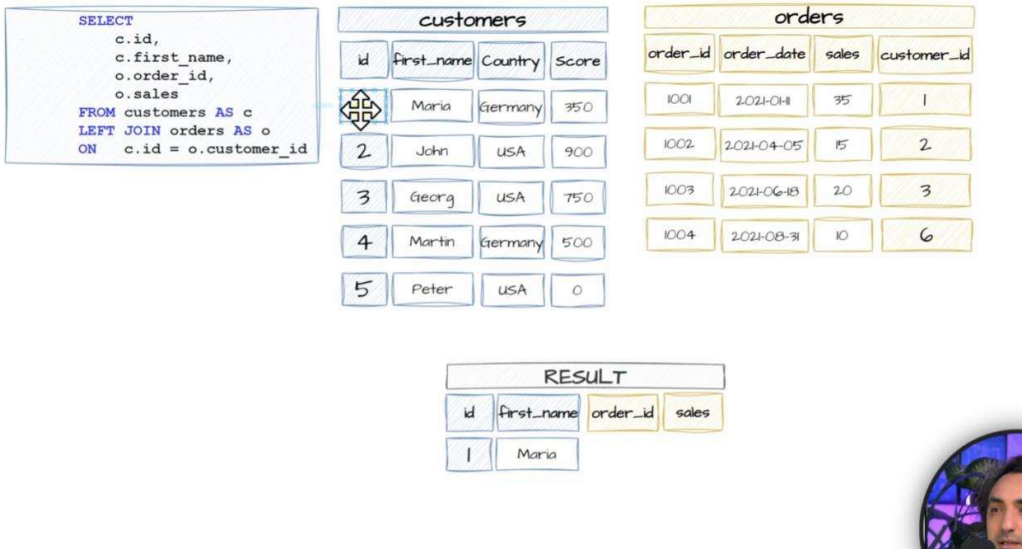
Results Messages

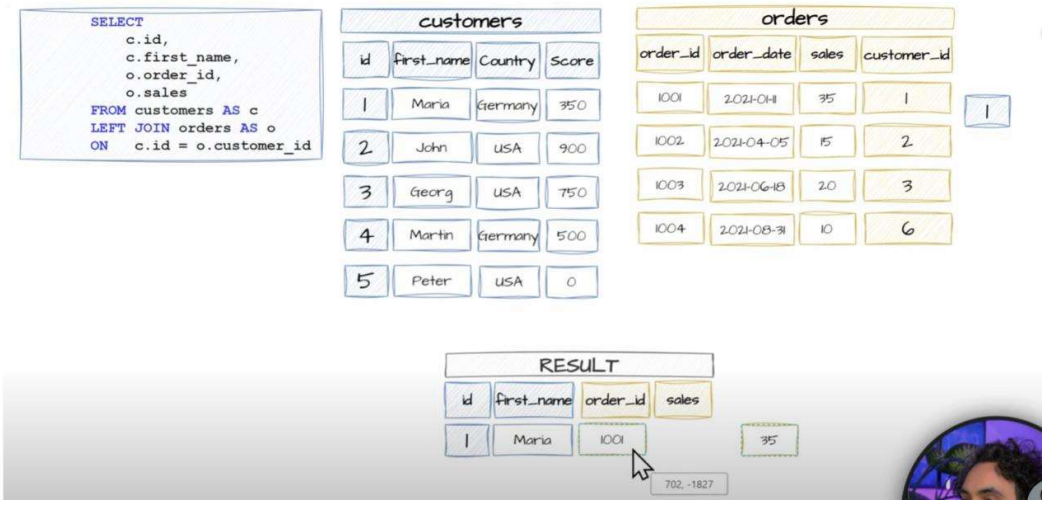
	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20
4	NULL	NULL	1004	10

Execution Order of Left Join:

For left table this add the result without check whether order exist for this customer or not
For getting order detail of that customer it matched the customer id and found the detail of order of that customer then add in table else add null

values in-place of order details of that customer and keep the customer into the result it does not ignore the customer or not keep out customer from result.





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

customers

id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

orders

order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

RESULT

id	first_name	order_id	sales
1	Maria	1001	35
2	John		

546, -1782


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

customers

id	first_name	Country	Score
1	Maria	Germany	350
2	John	USA	900
3	Georg	USA	750
4	Martin	Germany	500
5	Peter	USA	0

orders

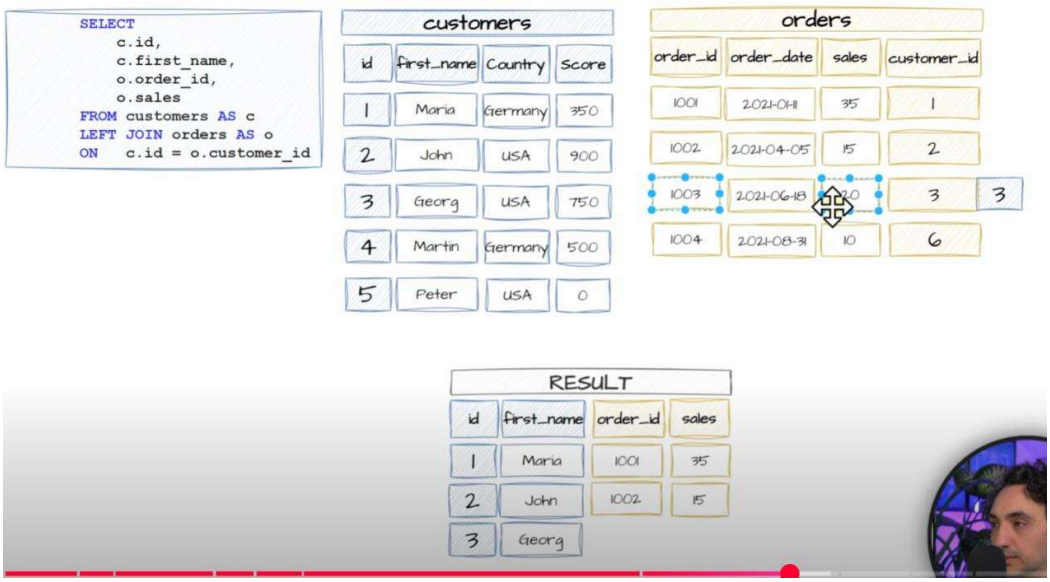
order_id	order_date	sales	customer_id
1001	2021-01-11	35	1
1002	2021-04-05	15	2
1003	2021-06-18	20	3
1004	2021-08-31	10	6

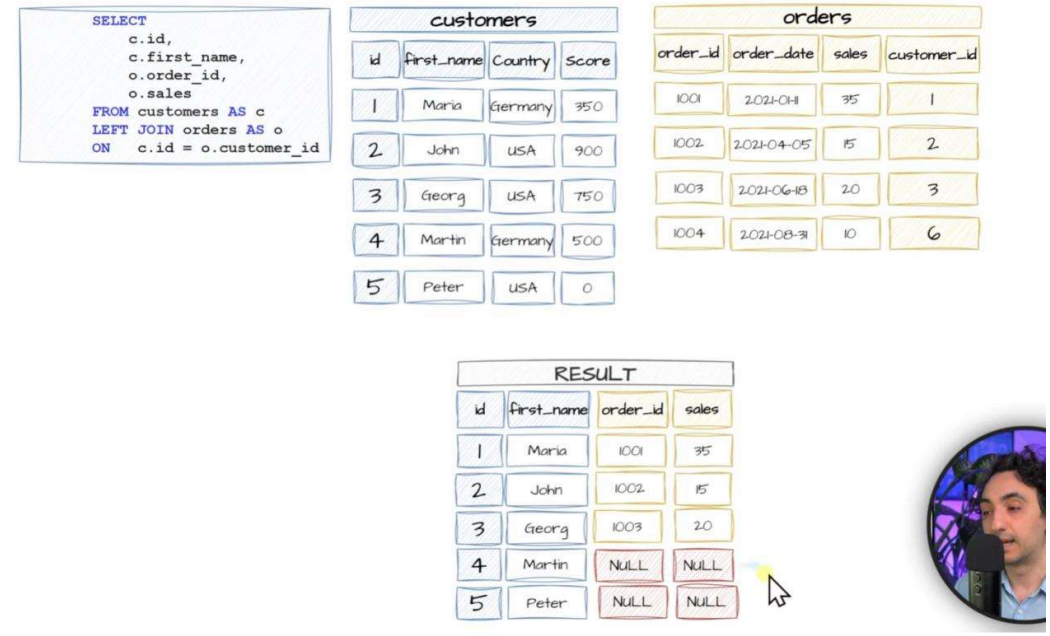
2

RESULT

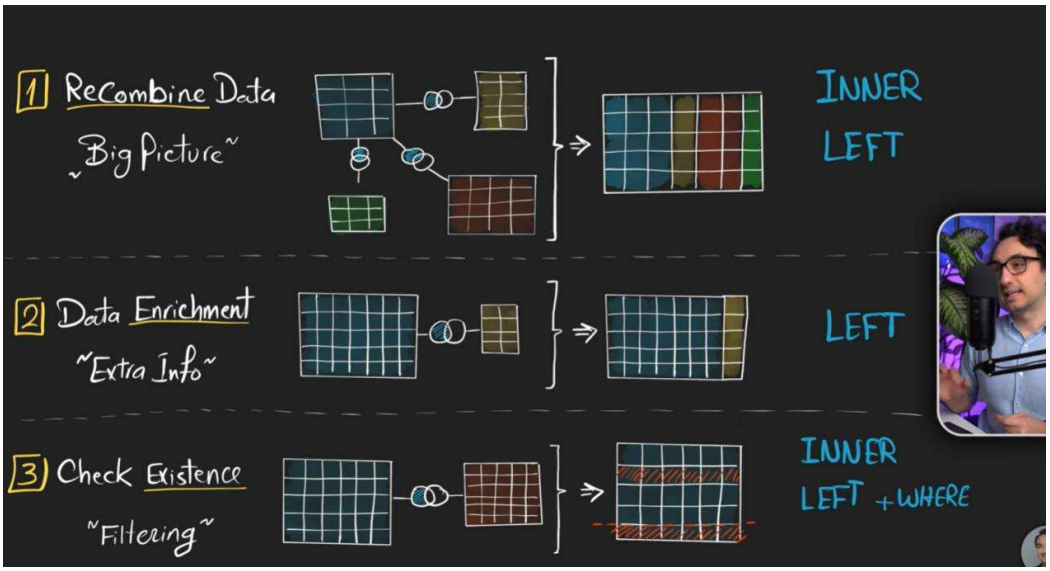
id	first_name	order_id	sales
1	Maria	1001	35
2	John	1001	

15

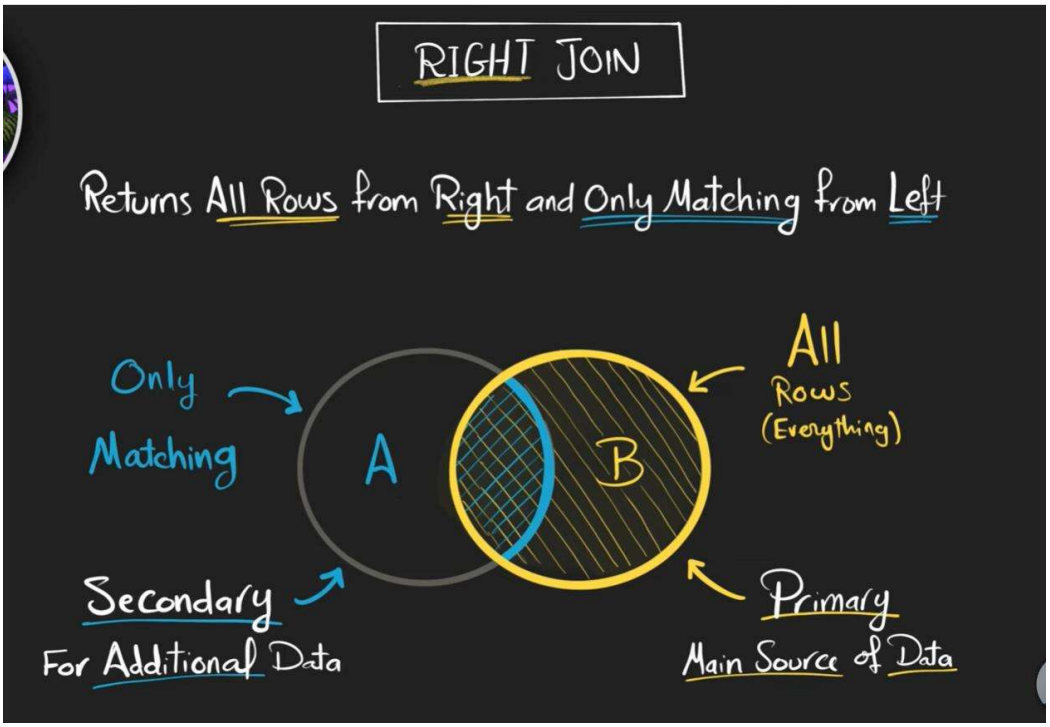


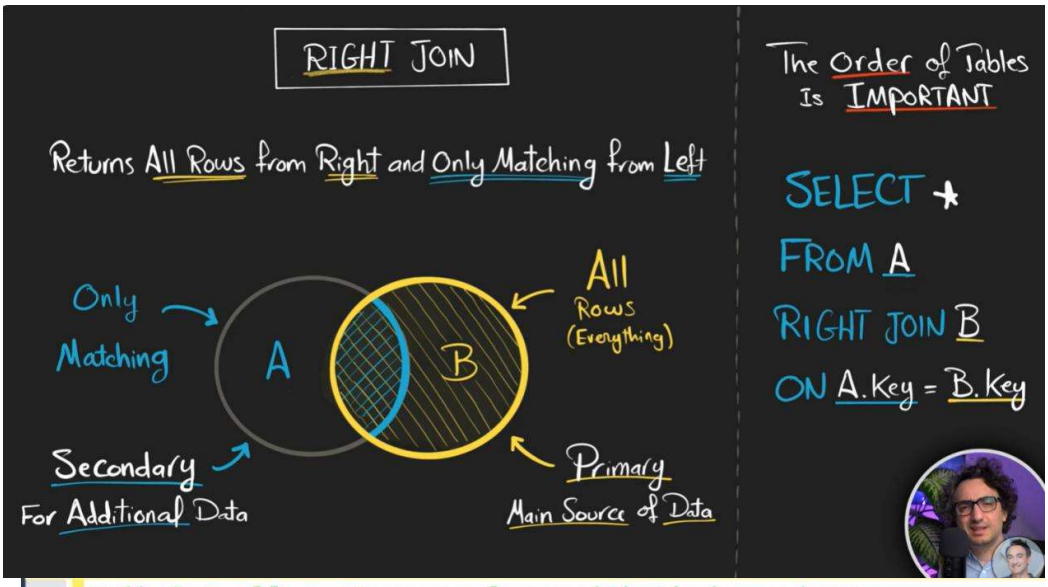


Left Join- used to recombine the data
Used to get extra information of data from another table
Used for filtering data with where clause


1 Recombine Data
~Big Picture~
INNER
LEFT
2 Data Enrichment
~Extra Info~
LEFT
3 Check Existence
~Filtering~
INNER
LEFT +WHERE

Right Join:


RIGHT JOIN
Returns All Rows from Right and Only Matching from Left
Only
Matching
A
B
All
Rows
(Everything)
Secondary
For Additional Data
Primary
Main Source of Data




```sql
/* Get all customers along with their orders,
   including orders without matching customers. */

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

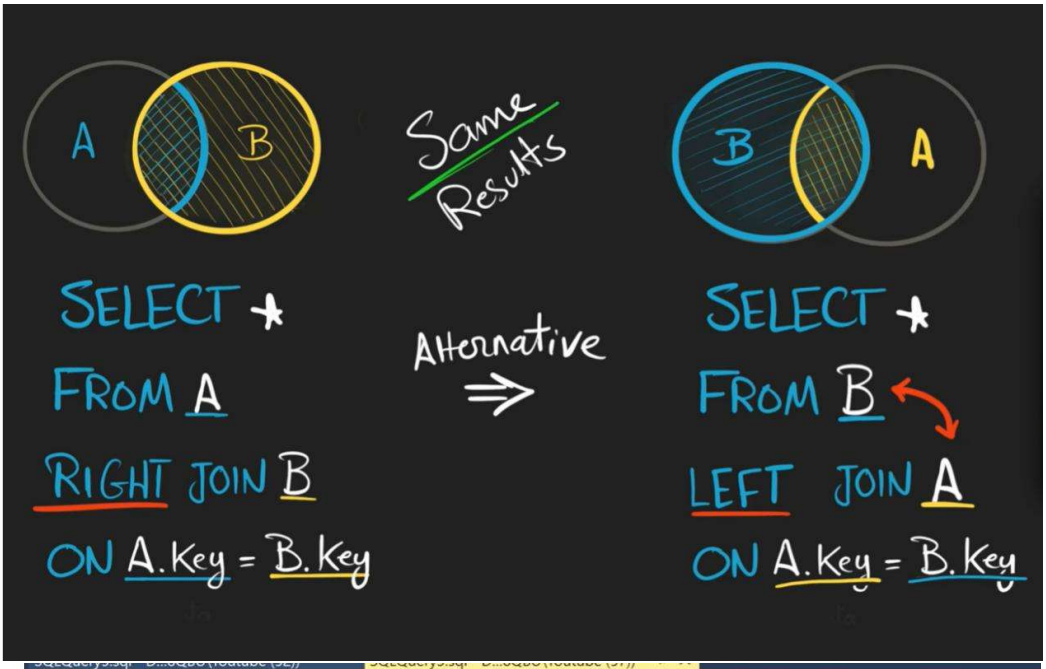
Results Messages

	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20
4	NULL	NULL	1004	10

Messages

id	first_name	order_id	sales
	Maria	1001	35
	John	1002	15

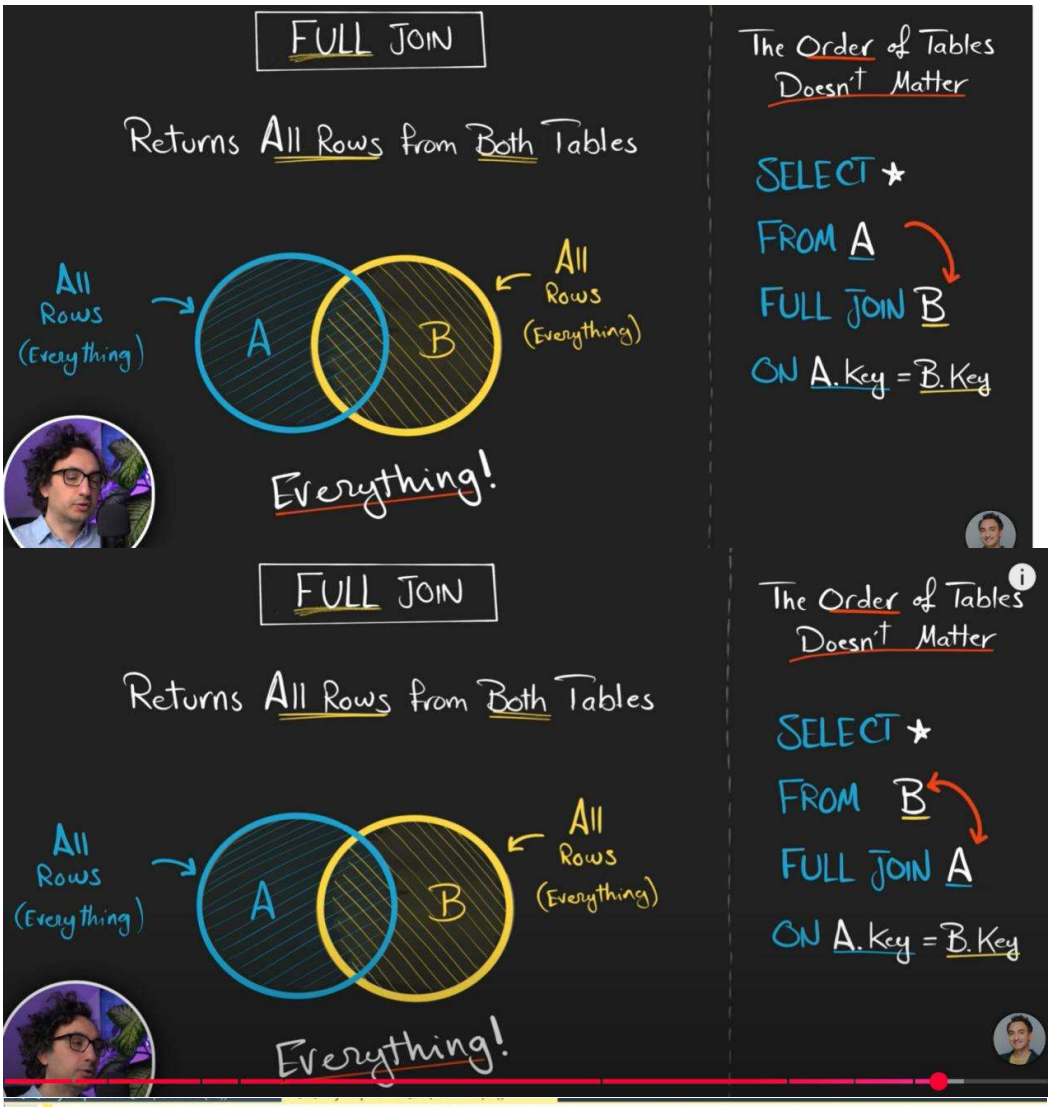
Solve the same task using LEFT JOIN




```sql
/* Get all customers along with their orders,
   including orders without matching customers. */

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
```

Results Messages

	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20
4	NULL	NULL	1004	10




```sql
/* Get all customers and all orders, even if there's no match. */

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

Results Messages

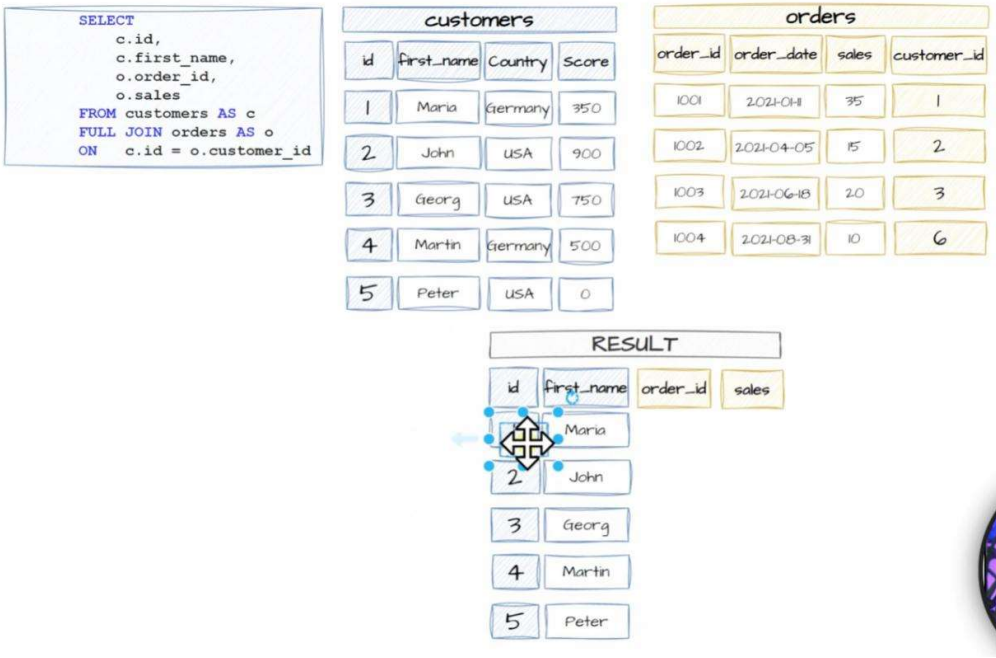
	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20
4	4	Martin	NULL	NULL
5	5	Peter	NULL	NULL
6	NULL	NULL	1004	10

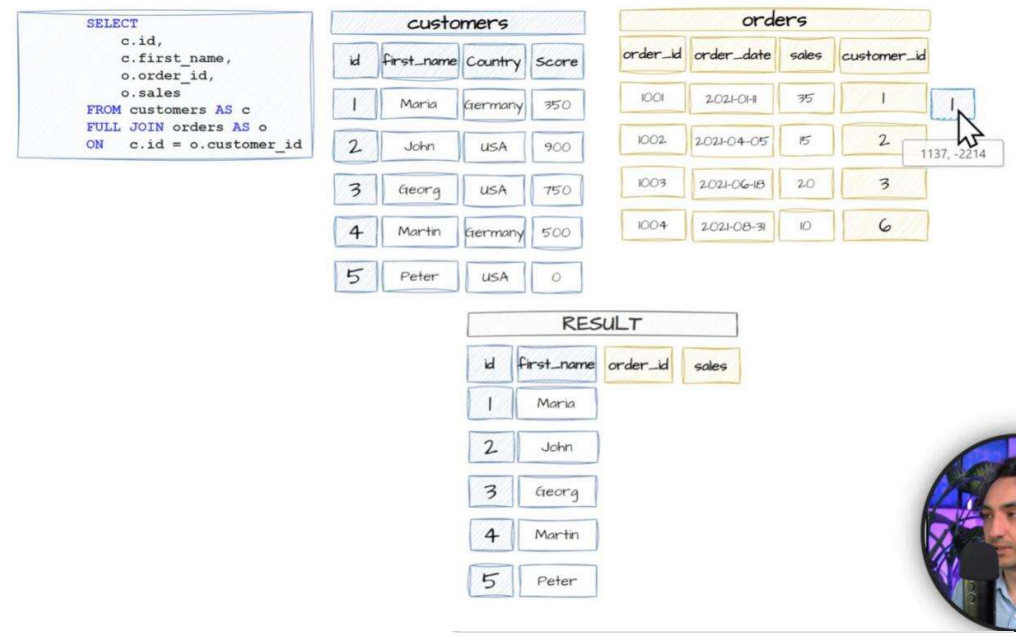
Order not matter here so same result by interchanging the order

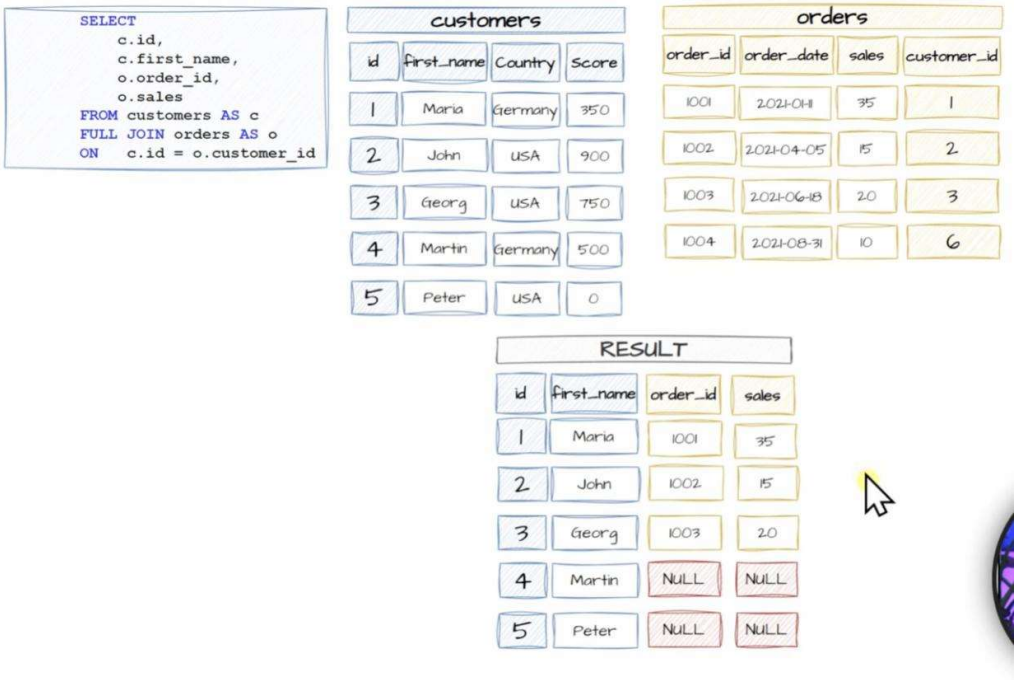
```sql
/* Get all customers and all orders, even if there's no match. */

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

Results | Messages

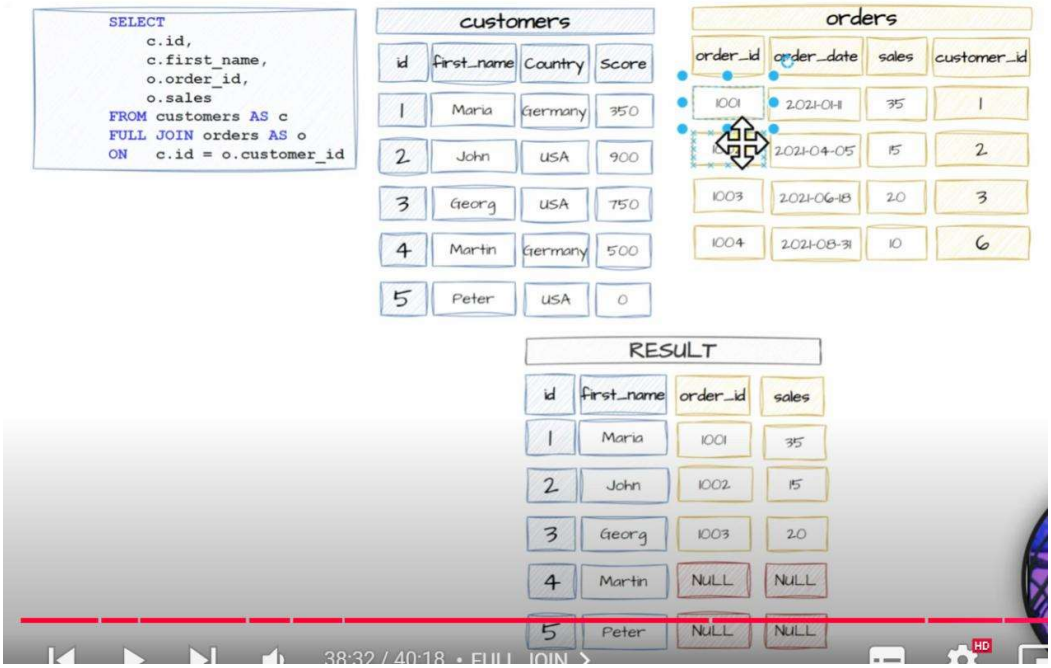
	id	first_name	order_id	sales
1	1	Maria	1001	35
2	2	John	1002	15
3	3	Georg	1003	20
4	NULL	NULL	1004	10
5	4	Martin	NULL	NULL
6	5	Peter	NULL	NULL

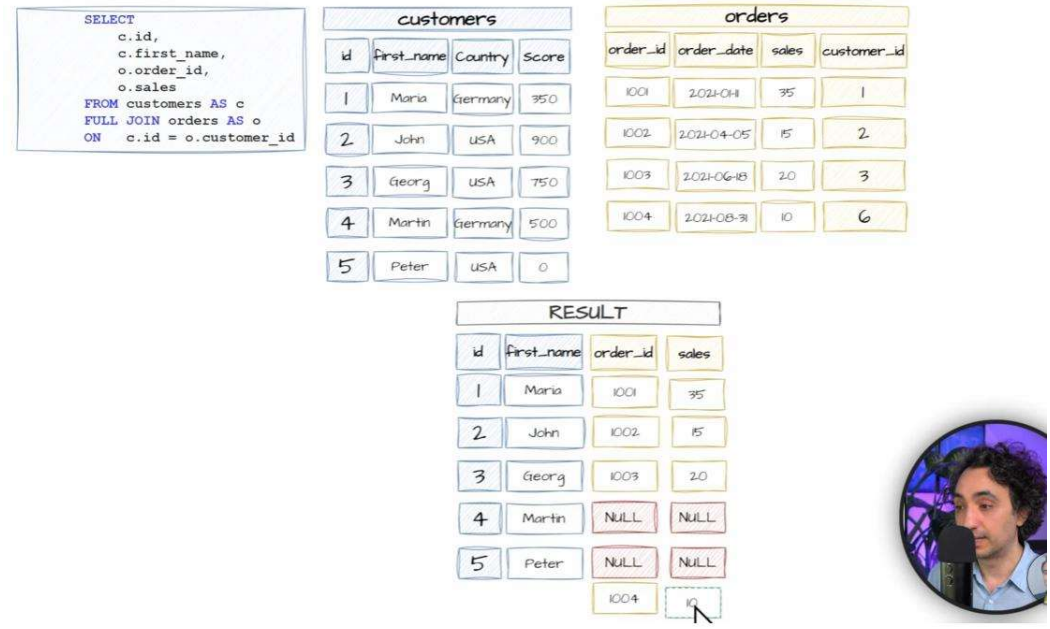




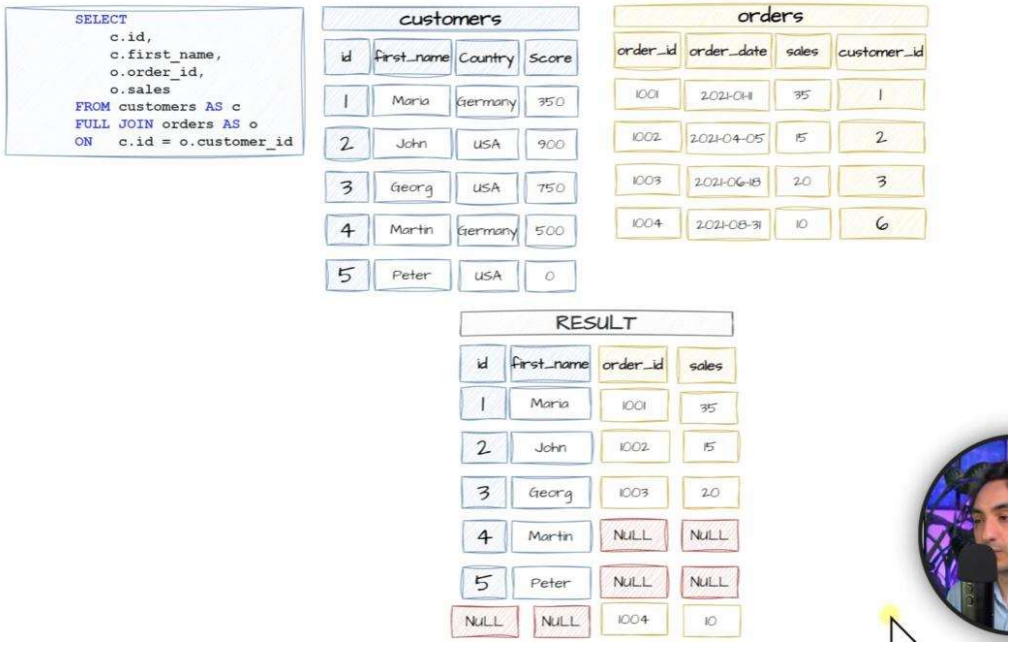


It check lastly all orders is added into the final output result

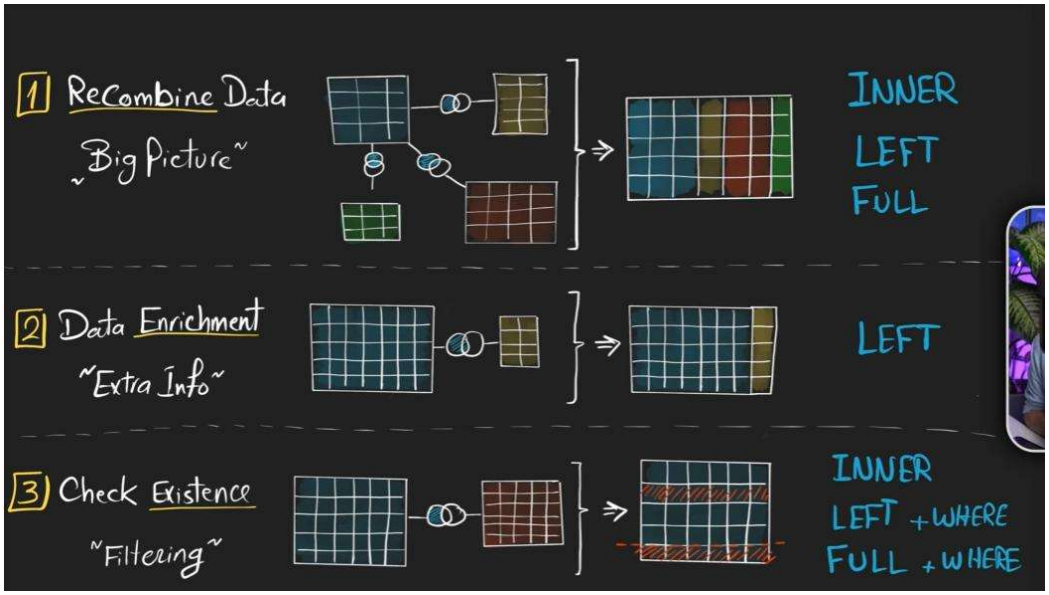




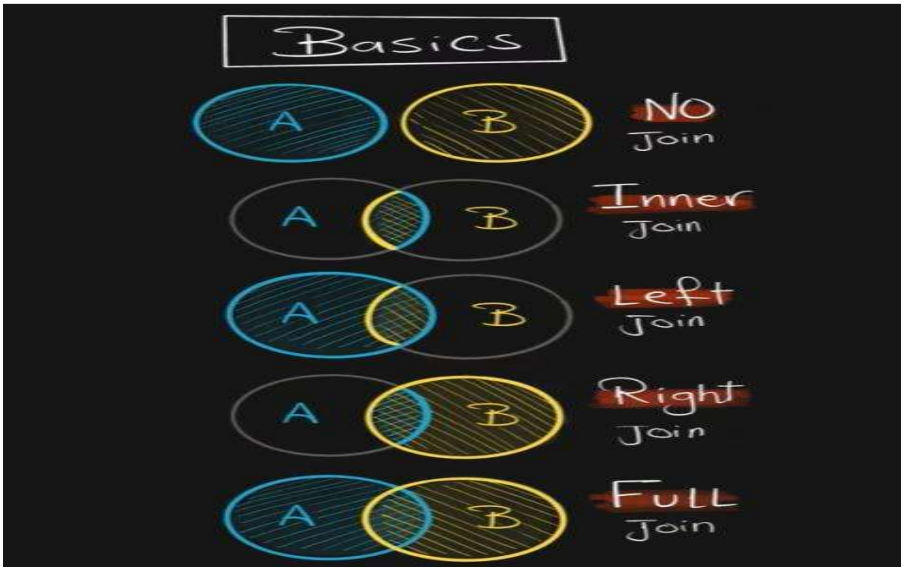
Above check and put all orders of right side and add null to left side if record not exist



Uses of Full Join

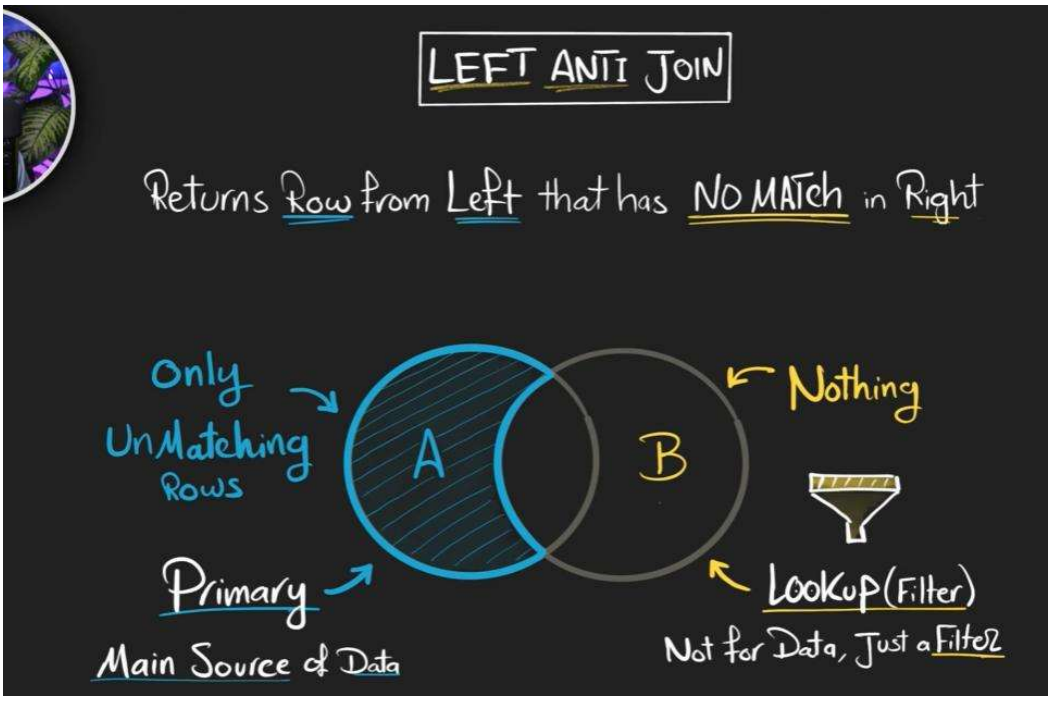


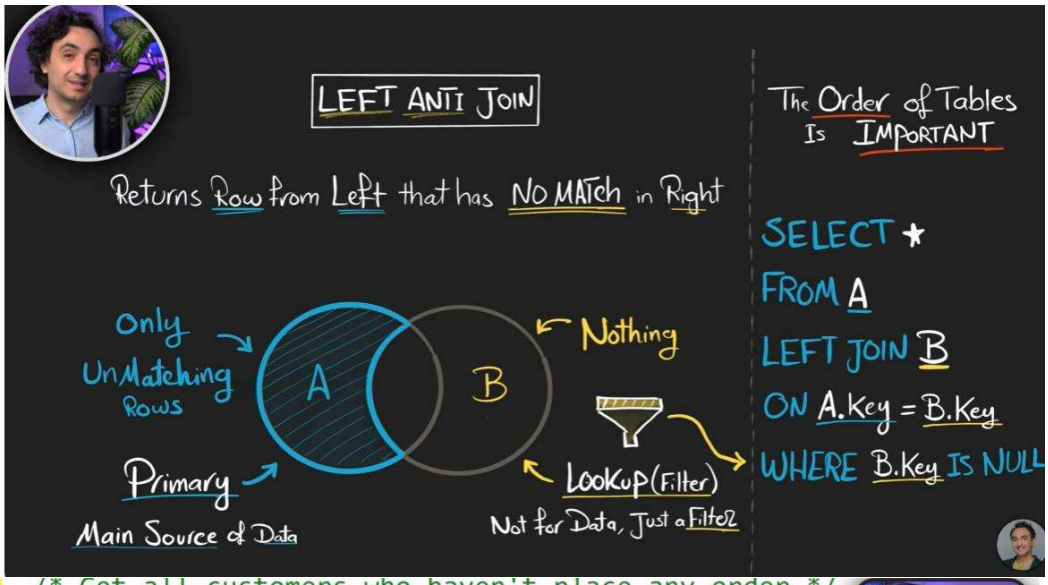
All Basics Joins covered



Advanced sql Joins :

1. Left Anti Joins






```sql
/* Get all customers who haven't place any order */

SELECT *
FROM customers AS c
LEFT JOIN orders AS o
ON c.id = o.customer_id
```

Results Messages

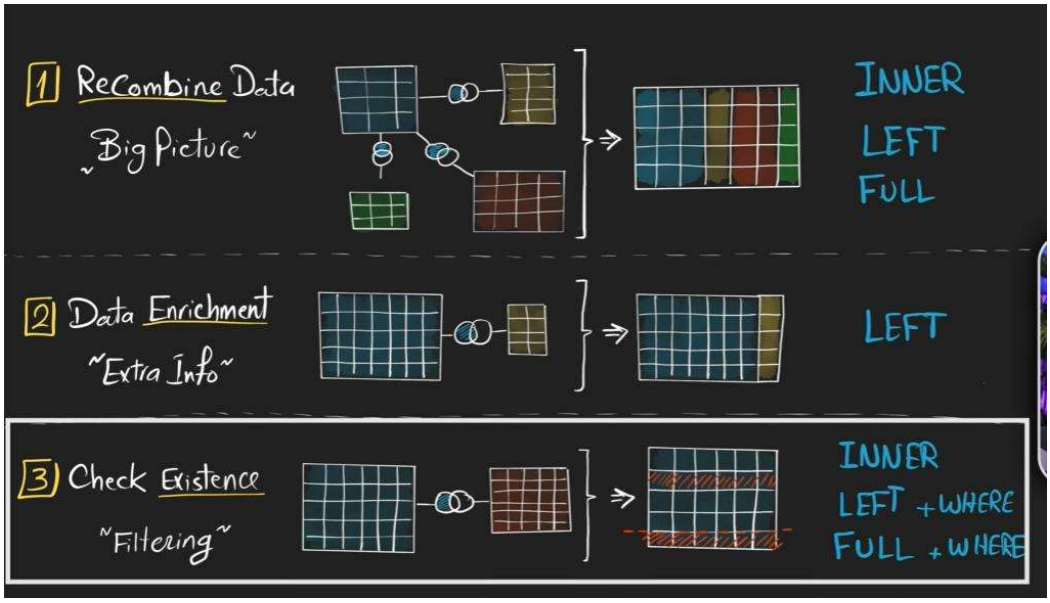
	id	first_name	country	score	order_id	customer_id	order_date	sales
1	1	Maria	Germany	350	1001	1	2021-01-11	35
2	2	John	USA	900	1002	2	2021-04-05	15
3	3	Georg	UK	750	1003	3	2021-06-18	20
4	4	Martin	Germany	500	NULL	NULL	NULL	NULL
5	5	Peter	USA	0	NULL	NULL	NULL	NULL

Currect Output -

```sql
/* Get all customers who haven't place any order */

SELECT *
FROM customers AS c
LEFT JOIN orders AS o
ON c.id = o.customer_id
WHERE o.customer_id IS NULL
```

Results Messages

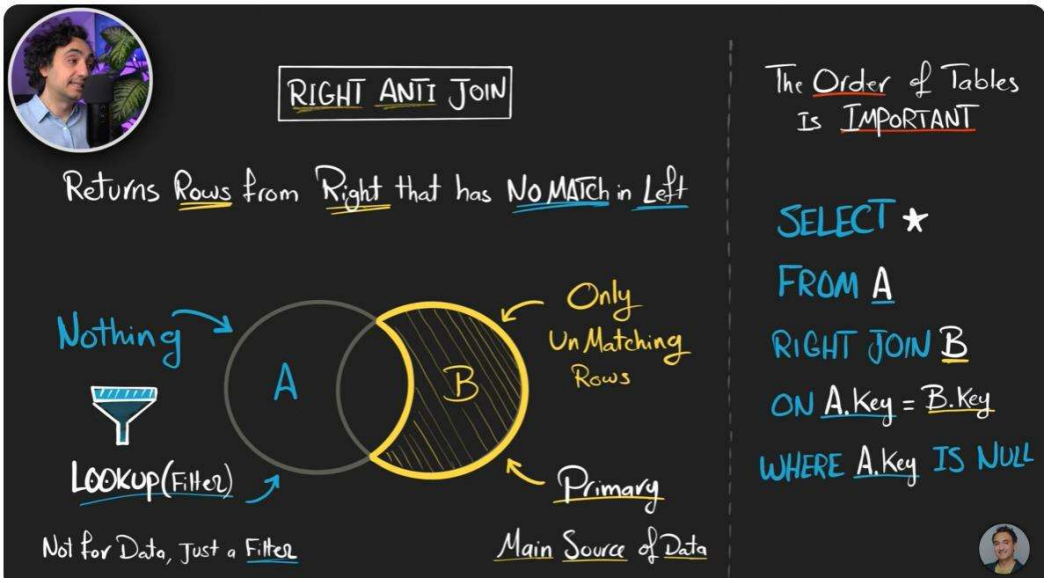
	id	first_name	country	score	order_id	customer_id	order_date	sales
1	4	Martin	Germany	500	NULL	NULL	NULL	NULL
2	5	Peter	USA	0	NULL	NULL	NULL	NULL



Used to check not exiting of data in other table

Right Anti Join:

Rows from right table that has no match in left table



Ex- get all orders withou matching customers
Business order without valid customers

```sql
/* Get all orders without matching customers */

SELECT *
FROM customers AS c
RIGHT JOIN orders AS o
ON c.id = o.customer_id
--WHERE o.customer_id IS NULL
```

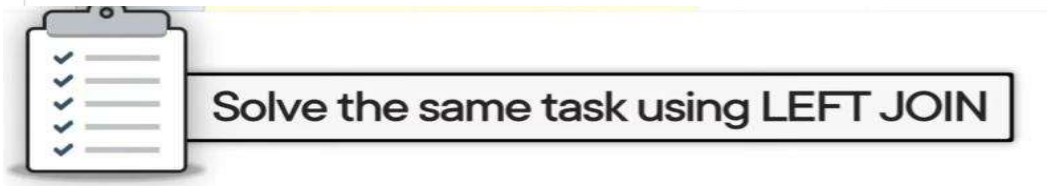
Results Messages

	id	first_name	country	score	order_id	customer_id	order_date	sale
1	1	Maria	Germany	350	1001	1	2021-01-11	35
2	2	John	USA	900	1002	2	2021-04-05	15
3	3	Georg	UK	750	1003	3	2021-06-18	20
4	NULL	NULL	NULL	NULL	1004	6	2021-08-31	10

```sql
/* Get all orders without matching customers */

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

Results Messages

	id	first_name	country	score	order_id	customer_id	order_date	sales
1	NULL	NULL	NULL	NULL	1004	6	2021-08-31	10

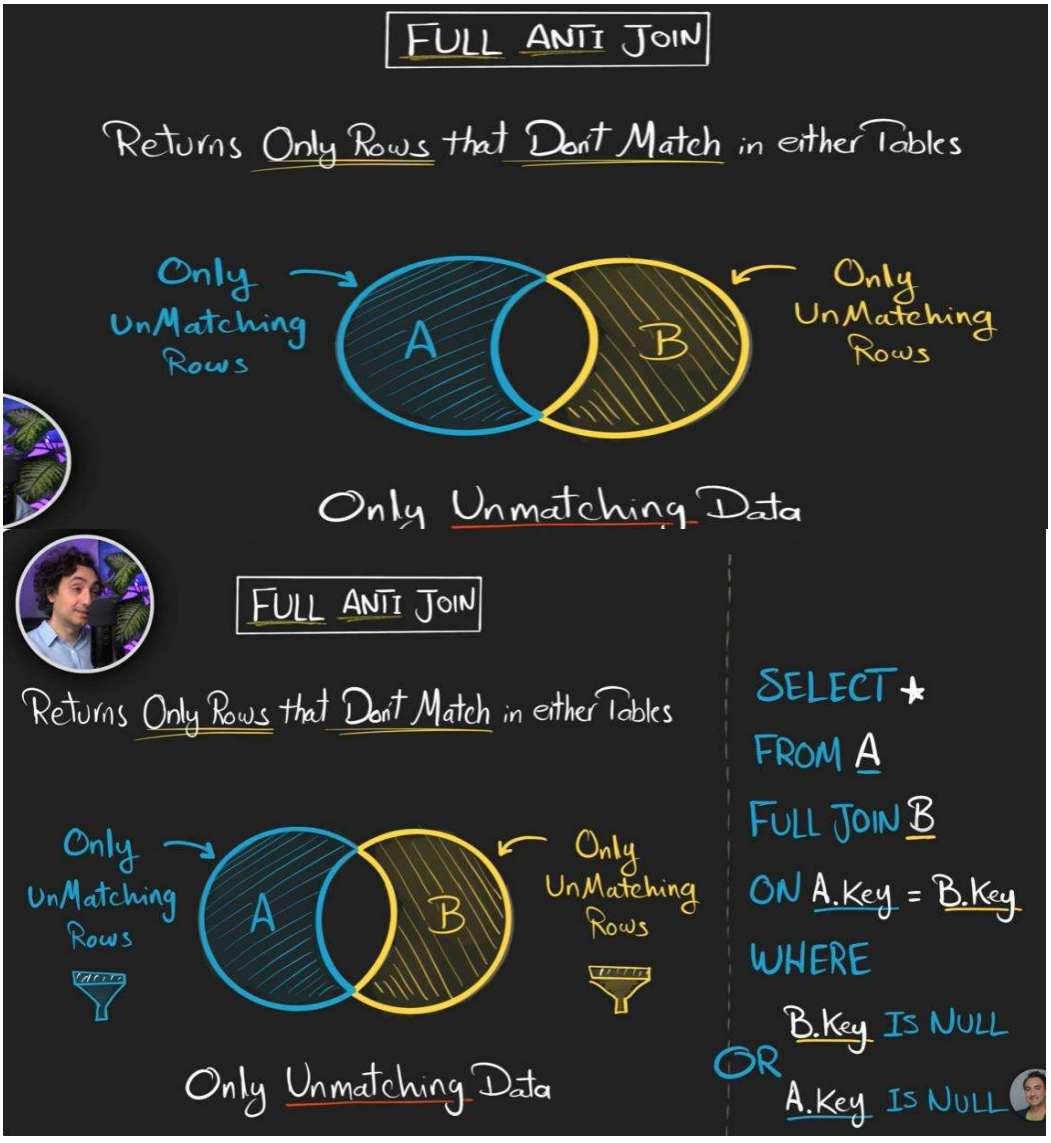


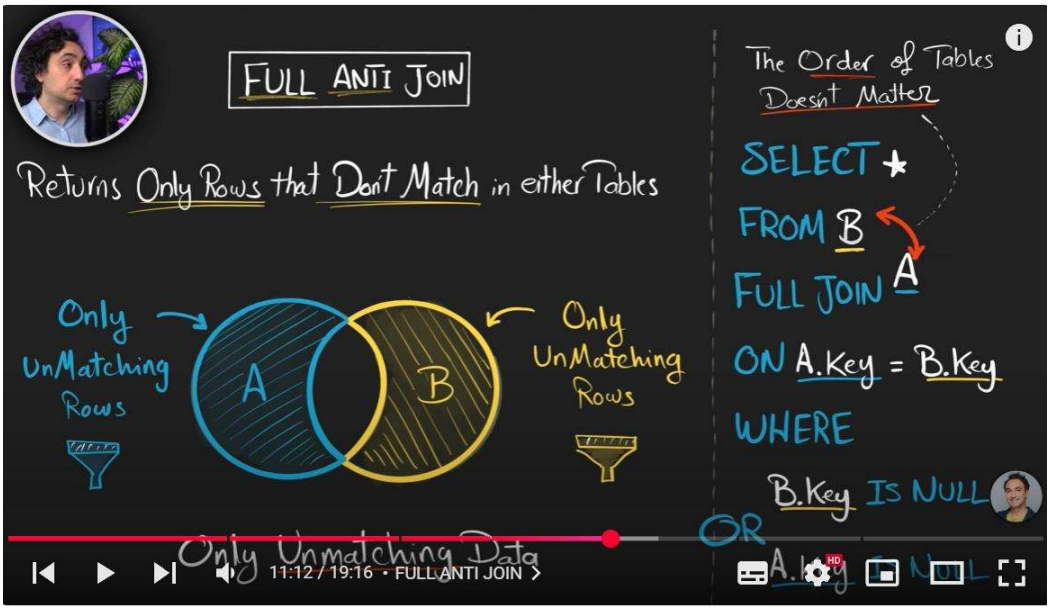
```sql
/* Get all orders without matching customers */

SELECT *
FROM orders AS o
LEFT JOIN customers AS c
ON c.id = o.customer_id
WHERE c.id IS NULL
```

Results Messages

	order_id	customer_id	order_date	sales	id	first_name	country	score
1	1004	6	2021-08-31	10	NULL	NULL	NULL	NULL

Full Anti Join in SQL:
Returns rows that do not match in either table.





Find customers without orders and orders without customers

```sql
/* Find customers without orders and orders without customers */

SELECT *
FROM orders AS o
FULL JOIN customers AS c
ON c.id = o.customer_id
-- WHERE c.id IS NULL
```

Results Messages

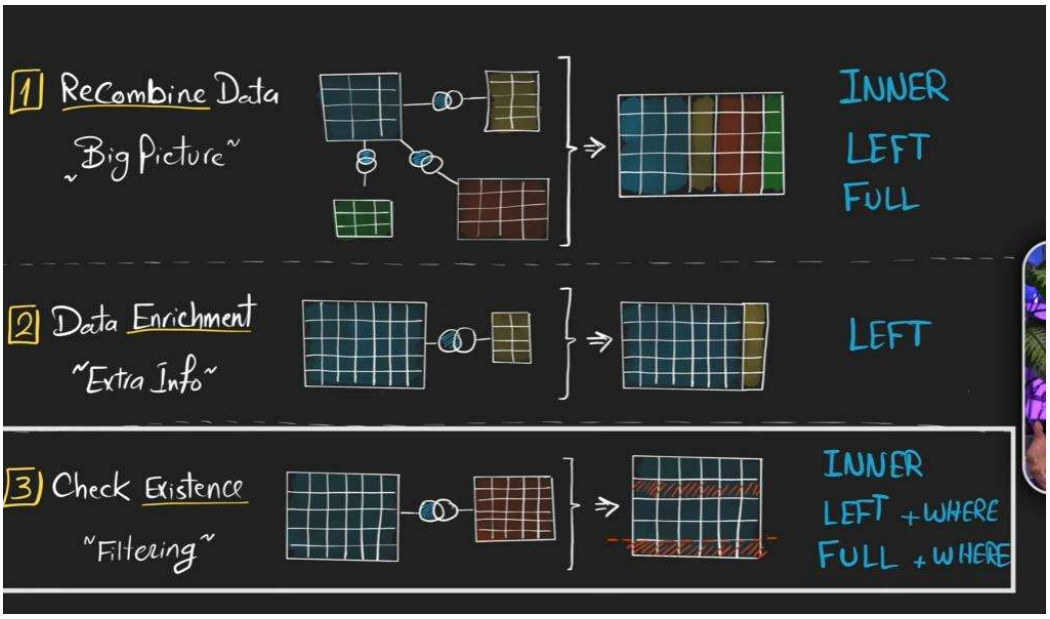
	order_id	customer_id	order_date	sales	id	first_name	country	score
1	1001	1	2021-01-11	35	1	Maria	Germany	350
2	1002	2	2021-04-05	15	2	John	USA	900
3	1003	3	2021-06-18	20	3	Georg	UK	750
4	1004	6	2021-08-31	10	NULL	NULL	NULL	NULL
5	NULL	NULL	NULL	NULL	4	Martin	Germany	500
6	NULL	NULL	NULL	NULL	5	Peter	USA	0

```sql
/* Find customers without orders and orders without customers */

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

Results Messages

	order_id	customer_id	order_date	sales	id	first_name	country	score
1	1004	6	2021-08-31	10	NULL	NULL	NULL	NULL
2	NULL	NULL	NULL	NULL	4	Martin	Germany	500
3	NULL	NULL	NULL	NULL	5	Peter	USA	0

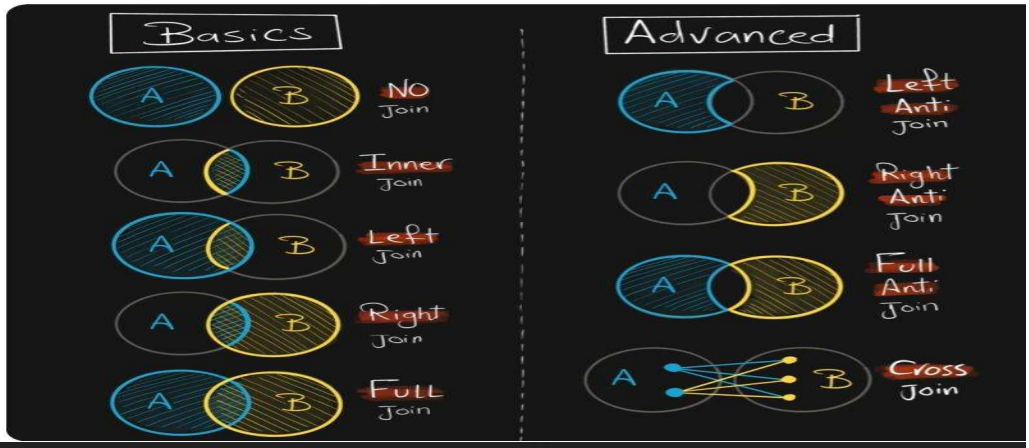
To check not existance of your data -- used Anti full join

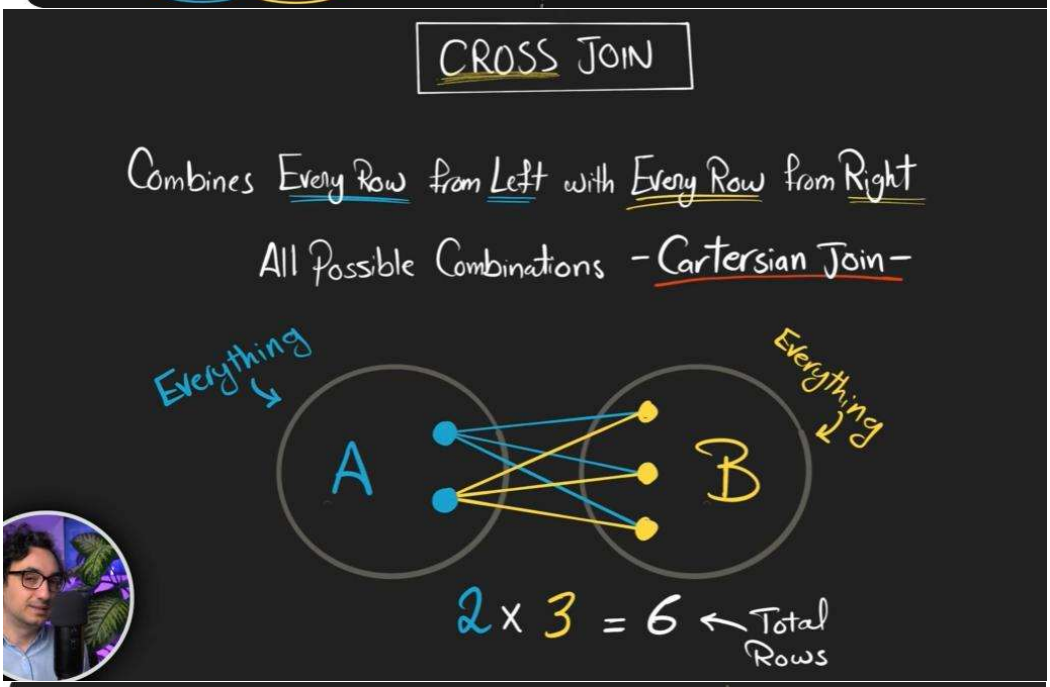


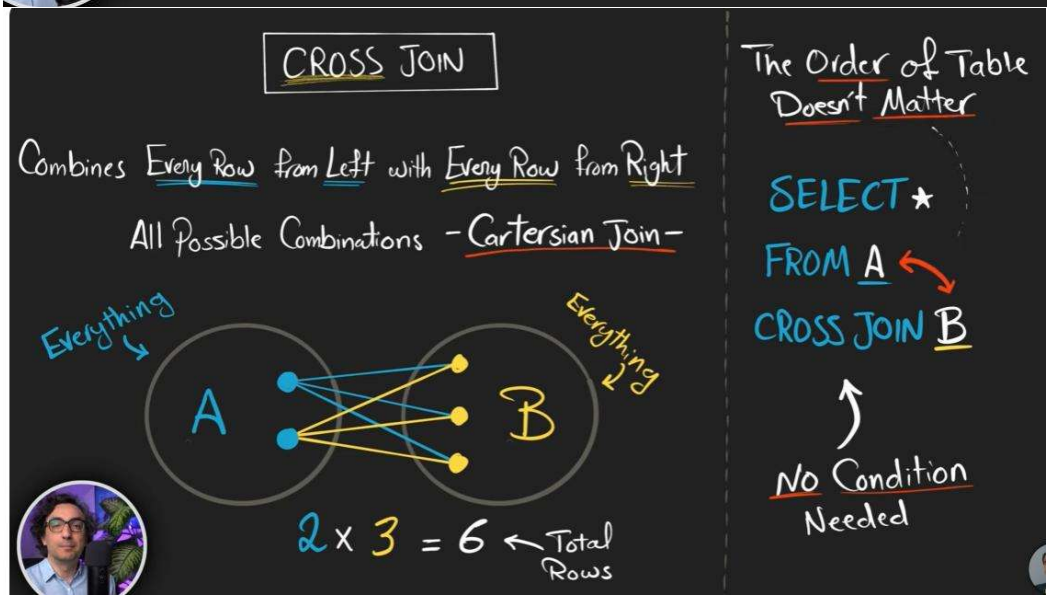

```sql
/* Get all customers along with their orders,
but only for customers who have placed an order
Without using INNER JOIN!! */

SELECT *
FROM customers AS c
LEFT JOIN orders AS o
ON c.id = o.customer_id
WHERE o.customer_id IS NOT NULL
```

Results Messages

	id	first_name	country	score	order_id	customer_id	order_date	sales
1	1	Maria	Germany	350	1001	1	2021-01-11	35
2	2	John	USA	900	1002	2	2021-04-05	15
3	3	Georg	UK	750	1003	3	2021-06-18	20


Basics
A
B
NO Join
Inner Join
Left Join
Right Join
FULL Join
Advanced
Left Anti Join
Right Anti Join
Full Anti Join
Cross Join

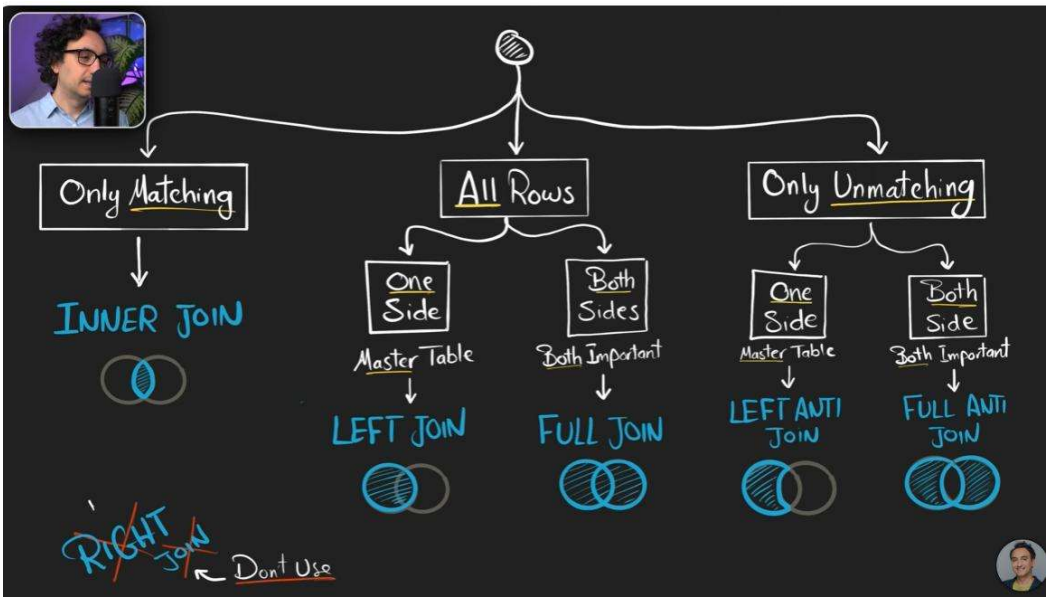

CROSS JOIN
Combines Every Row from Left with Every Row from Right
All Possible Combinations -Cartersian Join-
Everything
Everything
A
B
2 x 3 = 6 ← Total Rows


CROSS JOIN
Combines Every Row from Left with Every Row from Right
All Possible Combinations -Cartersian Join-
Everything
Everything
A
B
2 x 3 = 6 ← Total Rows
The Order of Table Doesn't Matter
SELECT *
FROM A
CROSS JOIN B
No Condition Needed

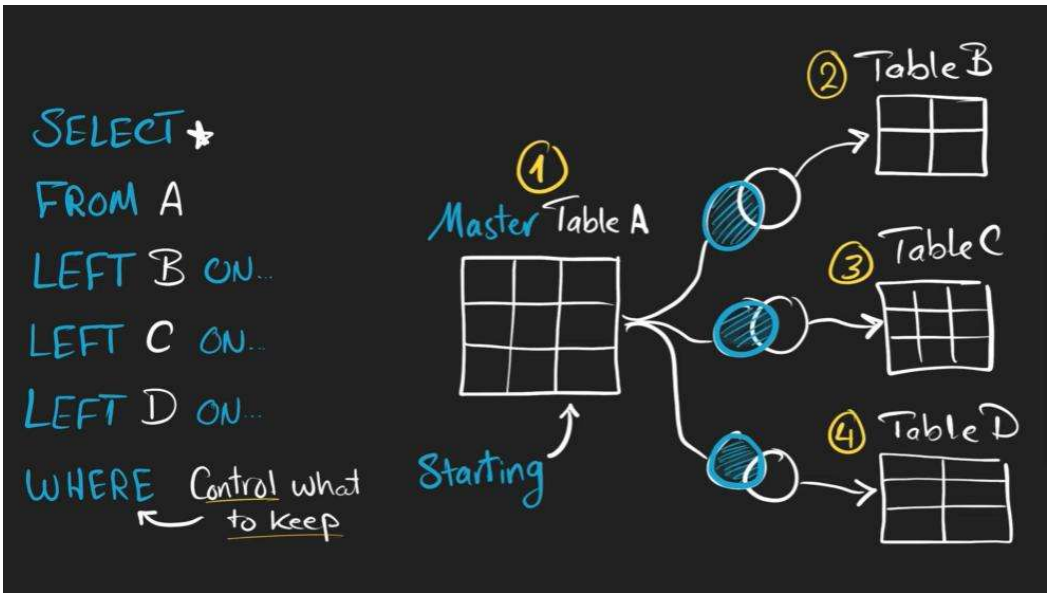
```sql
/* Generate all possible combinations of customers and orders. */

SELECT *
FROM customers
CROSS JOIN orders
```

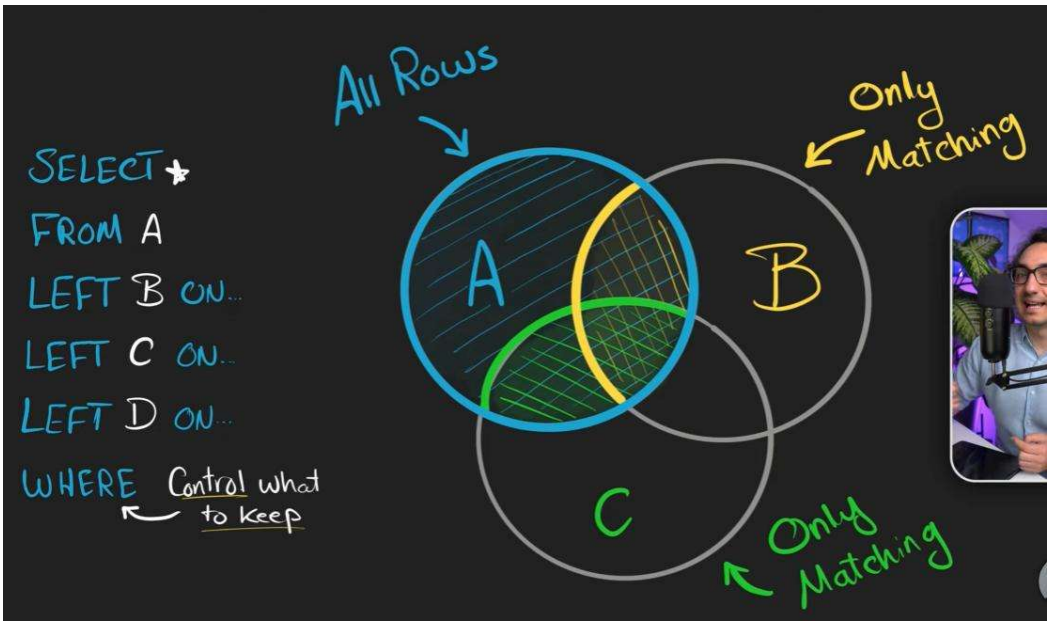
Results Messages

	id	first_name	country	score	order_id	customer_id	order_date	sales
1	1	Maria	Germany	350	1001	1	2021-01-11	35
2	2	John	USA	900	1001	1	2021-01-11	35
3	3	Georg	UK	750	1001	1	2021-01-11	35
4	4	Martin	Germany	500	1001	1	2021-01-11	35
5	5	Peter	USA	0	1001	1	2021-01-11	35
6	1	Maria	Germany	350	1002	2	2021-04-05	15
7	2	John	USA	900	1002	2	2021-04-05	15
8	3	Georg	UK	750	1002	2	2021-04-05	15
9	4	Martin	Germany	500	1002	2	2021-04-05	15
10	5	Peter	USA	0	1002	2	2021-04-05	15
11	1	Maria	Germany	350	1003	3	2021-06-18	20
12	2	John	USA	900	1003	3	2021-06-18	20



Using left join and left join with where clause we can control any condition and make result as we want.


SELECT *
FROM A
LEFT B ON...
LEFT C ON...
LEFT D ON...
WHERE Control what to keep
1
Master Table A
Starting
2 Table B
3 Table C
4 Table D


SELECT *
FROM A
LEFT B ON...
LEFT C ON...
LEFT D ON...
WHERE Control what to keep
All Rows
Only Matching
A
B
C
Only Matching