**INNER JOIN**

// intersection

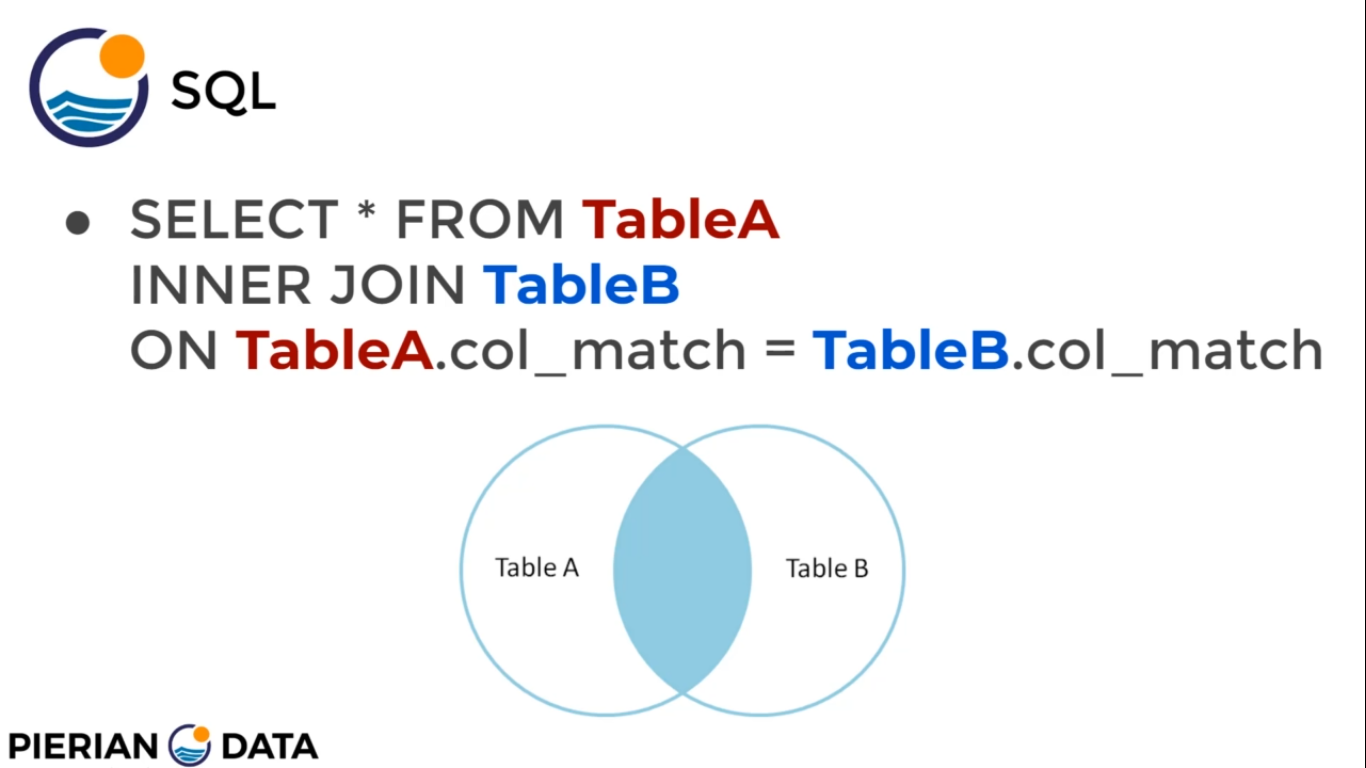
// order of table A nd B does not matter

// INNER JOIN ON

// final result has all the columns of A + B

// avoid dupliacy

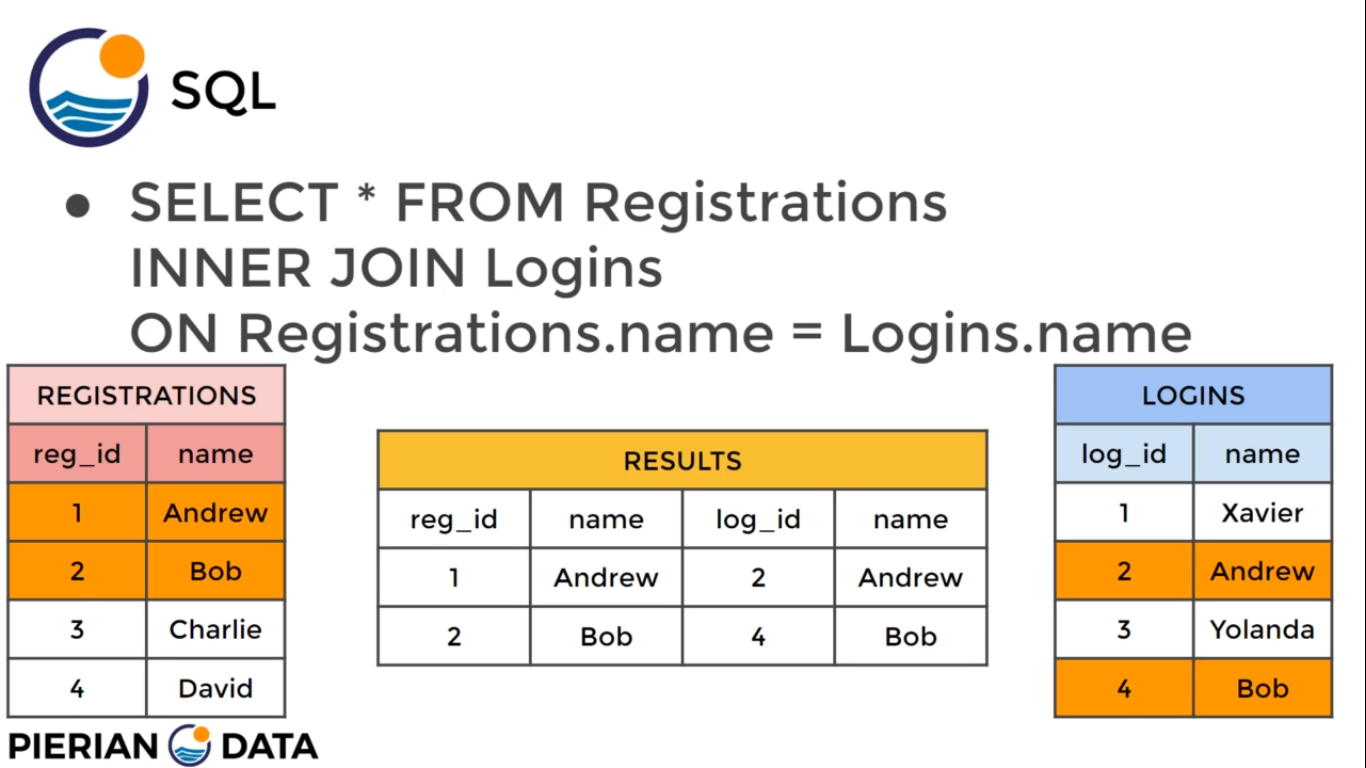
// JOIN is treated as INNER JOIN in Postgresql



Situation - find customer that are in both table

**SELECT \* FROM payment INNER JOIN customer ON payment.customer\_id = payment.customer\_id**

**SELECT email, payment.customer\_id ,amount FROM payment INNER JOIN customer ON payment.customer\_id = payment.customer\_id**



**OUTER JOIN**

// union

// with condition removes the intersection part of venn diagram

// order of table A nd B does not matter

// FULL OUTER JOIN ON

// final result has all the columns of A + B

// for a match in A and B values are filled in columns

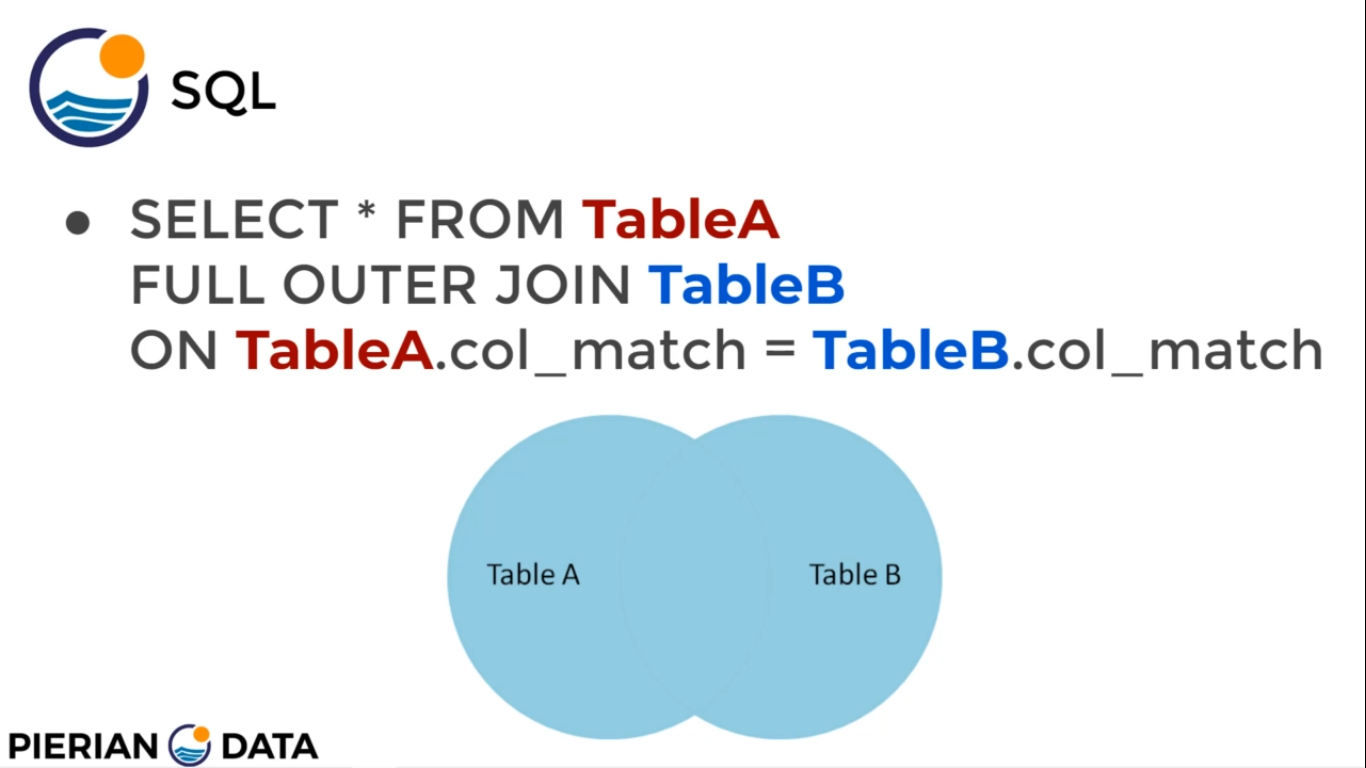
// for no match NULL is filled

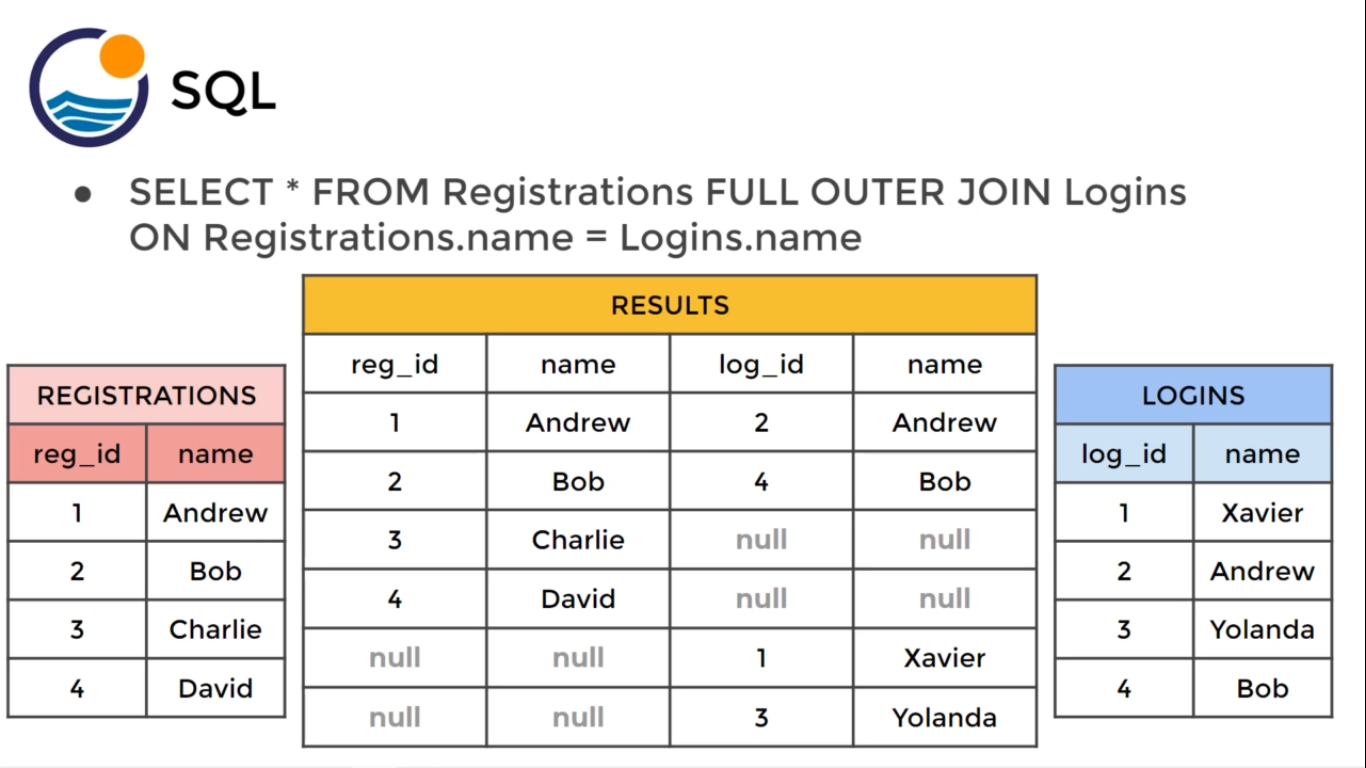
// can be used to get unique items to A nd B

// FULL OUTER JOIN contains INNER JOIN

situation- find customer that are not in both table(present in either of the table)

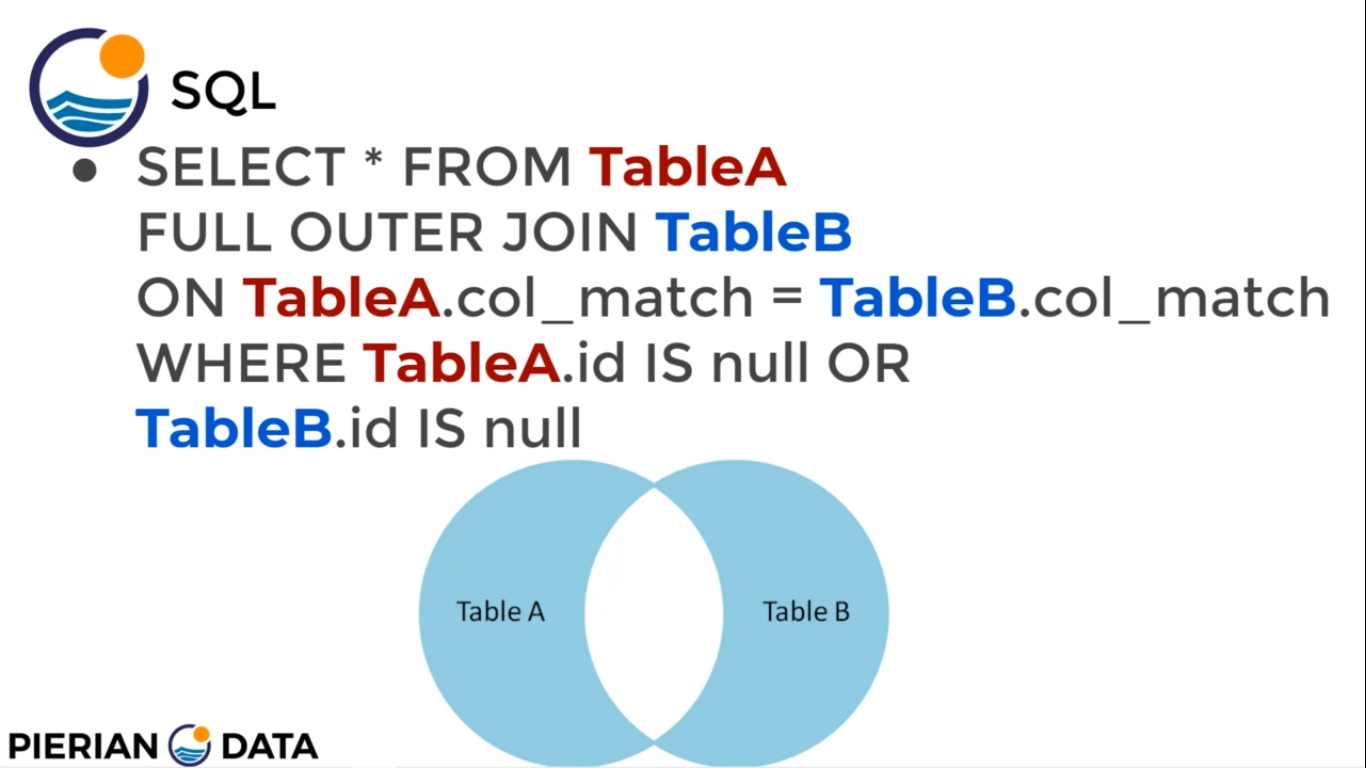
**SELECT \* FROM customer FULL OUTER JOIN payment ON customer.customer\_id = payment.customer\_id**





**SELECT \* FROM customer FULL OUTER JOIN payment ON customer.customer\_id = payment.customer\_id**

**WHERE customer.customer\_id IS null AND payment.customer\_id IS null**



**LEFT JOIN**

// considers only LEFT table A

// with condition removes the intersection part of venn diagram

// order of table A nd B does matter

// LEFT OUTER JOIN ON

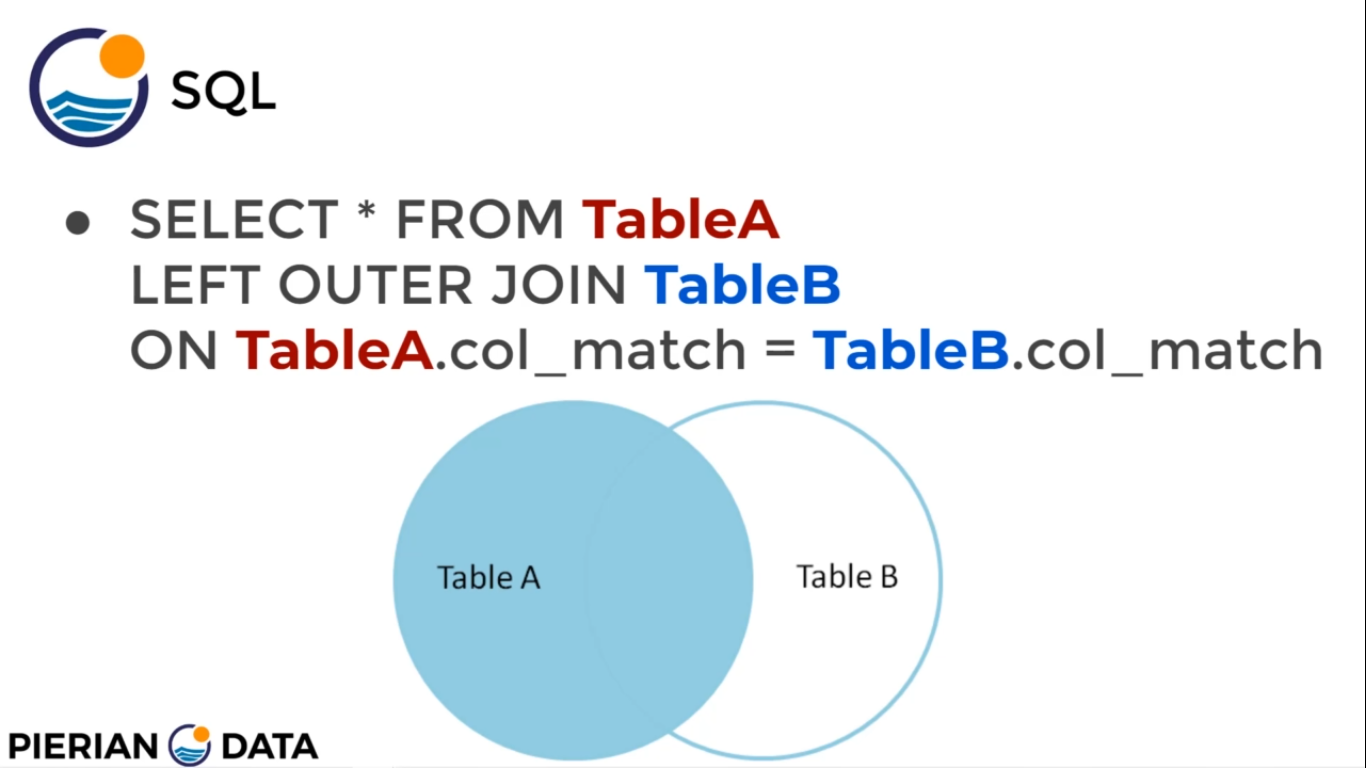
// final result has all the columns of A + B

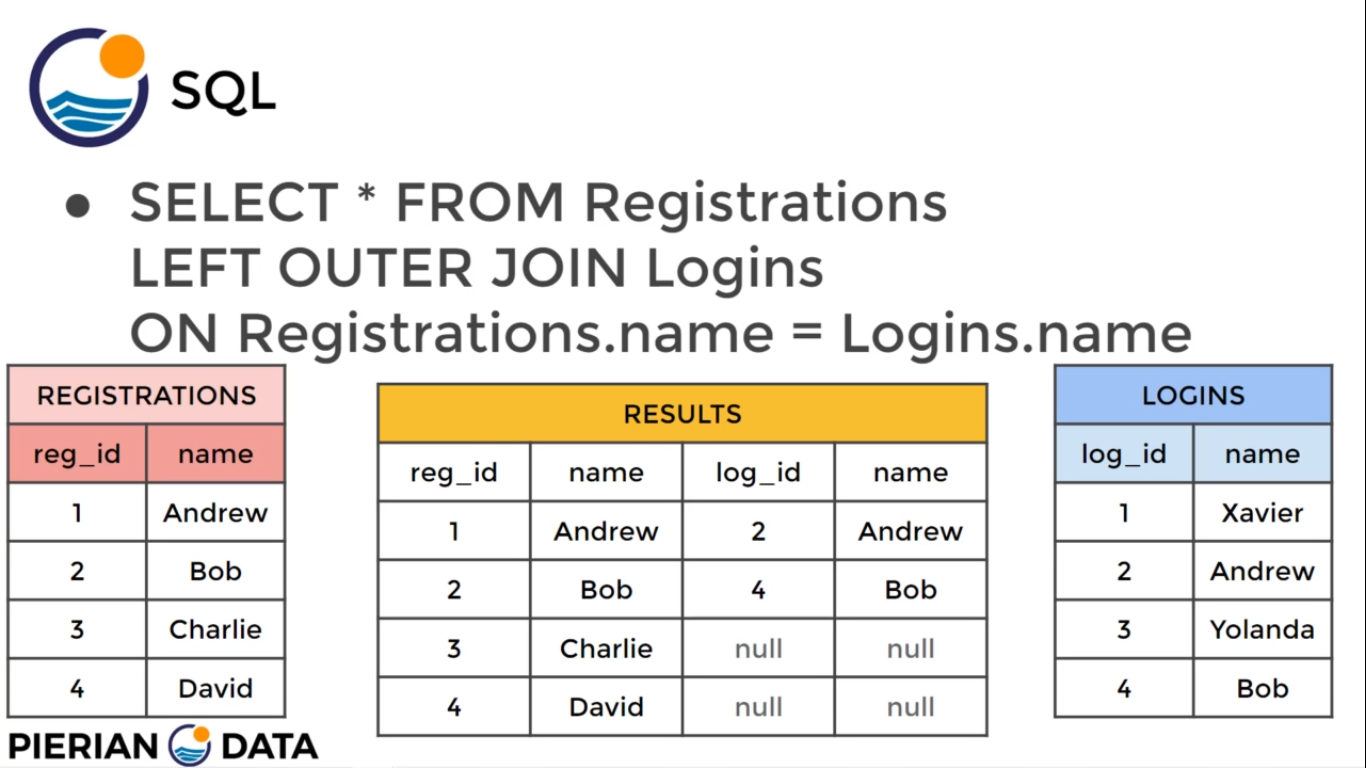
// has all the rows of A

// for a match in A and B values are filled in columns

// for no match in B null is filled

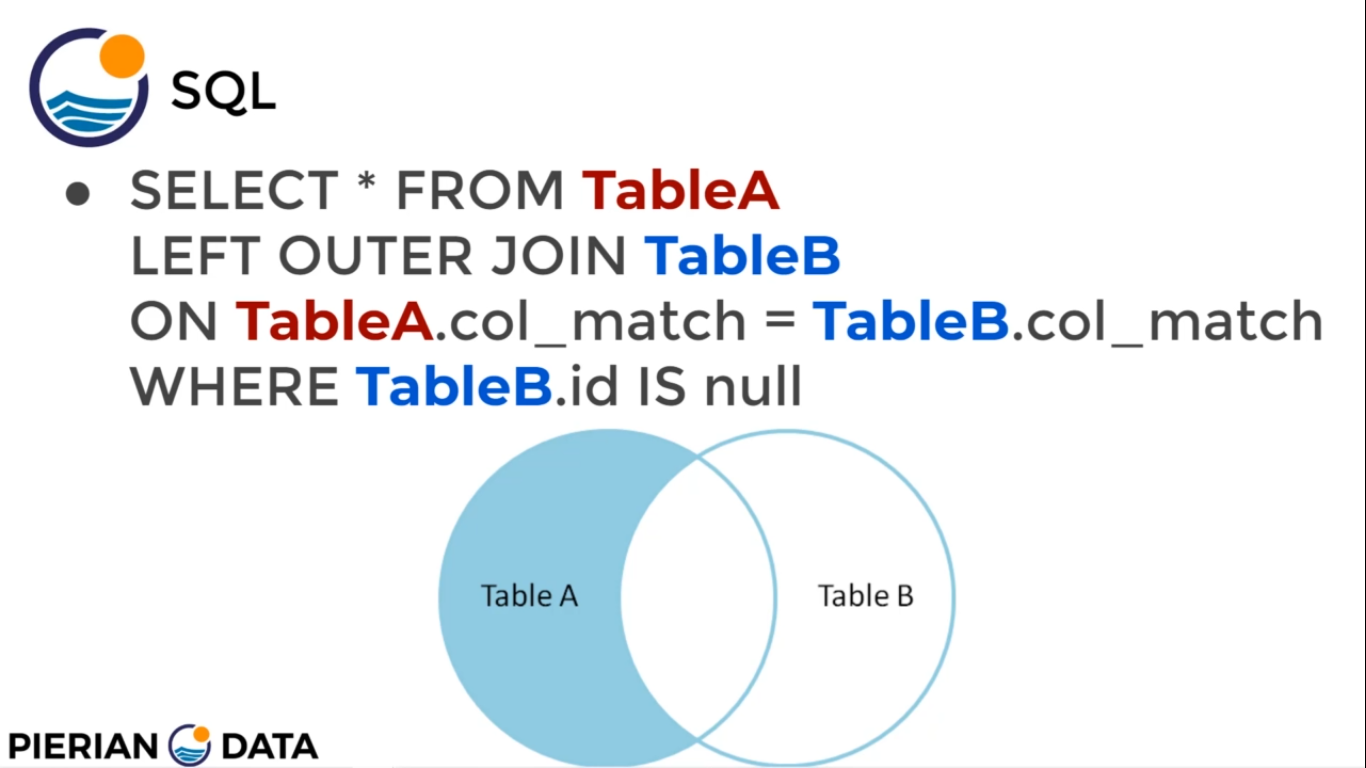
// can be used to get unique items to A nd B





**SELECT \* FROM film LEFT OUTER JOIN inventory**

**ON film.film\_id = inventory.film\_id WHERE inventory.film\_id IS null**



**RIGHT JOIN**

// considers only RIGHT table B

// with condition removes the intersection part of venn diagram

// order of table A nd B does matter

// RIGHT OUTER JOIN ON

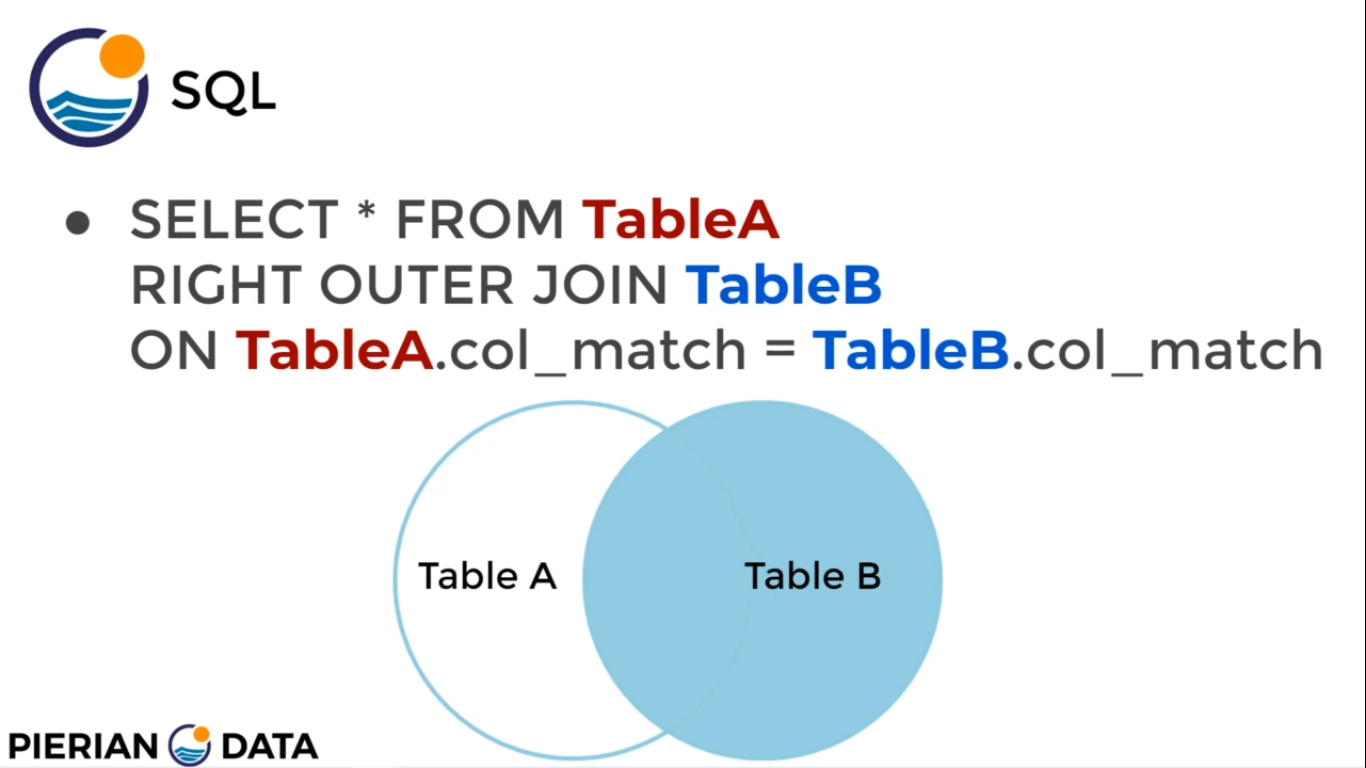
// final result has all the columns of A + B

// has all the rows of B

// for a match in A and B values are filled in columns

// for no match in A null is filled

// can be used to get unique items to A nd B



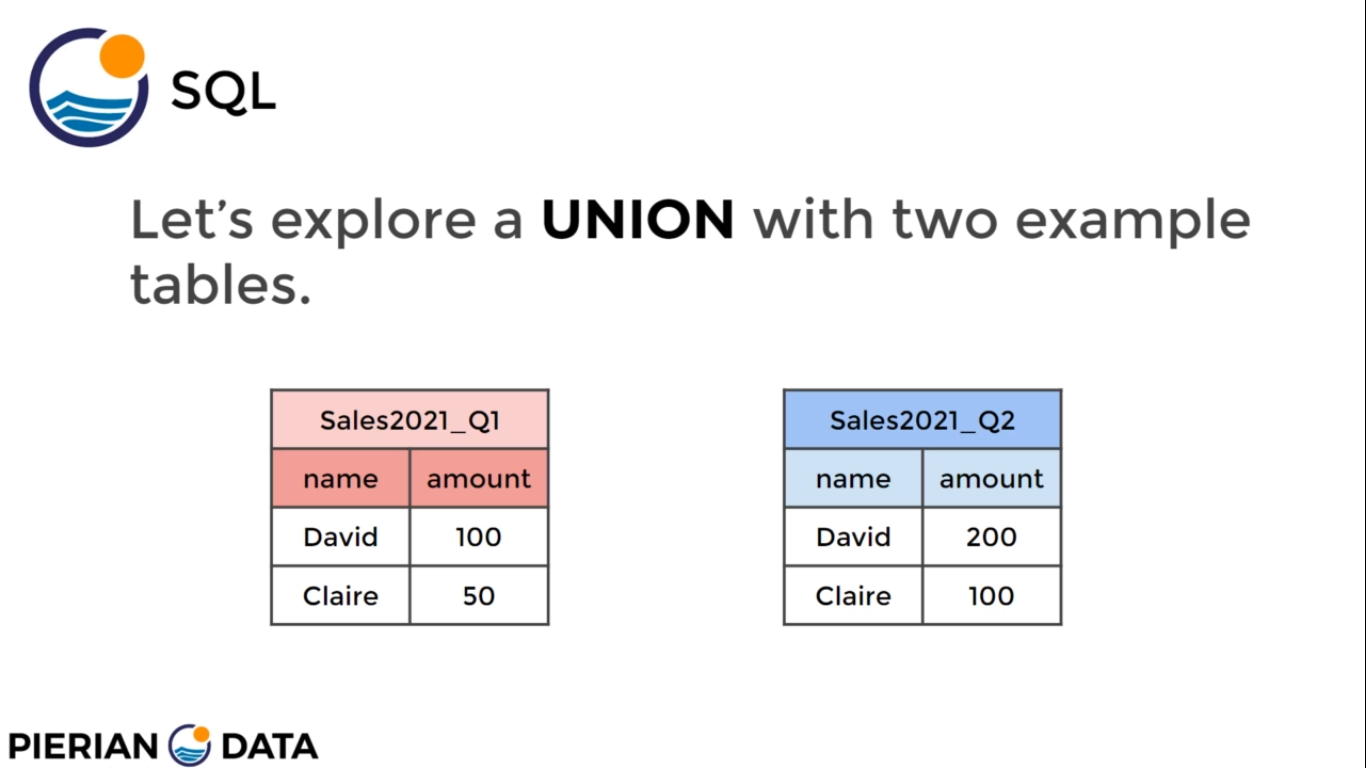
**SELECT \* FROM film RIGHT OUTER JOIN inventory**

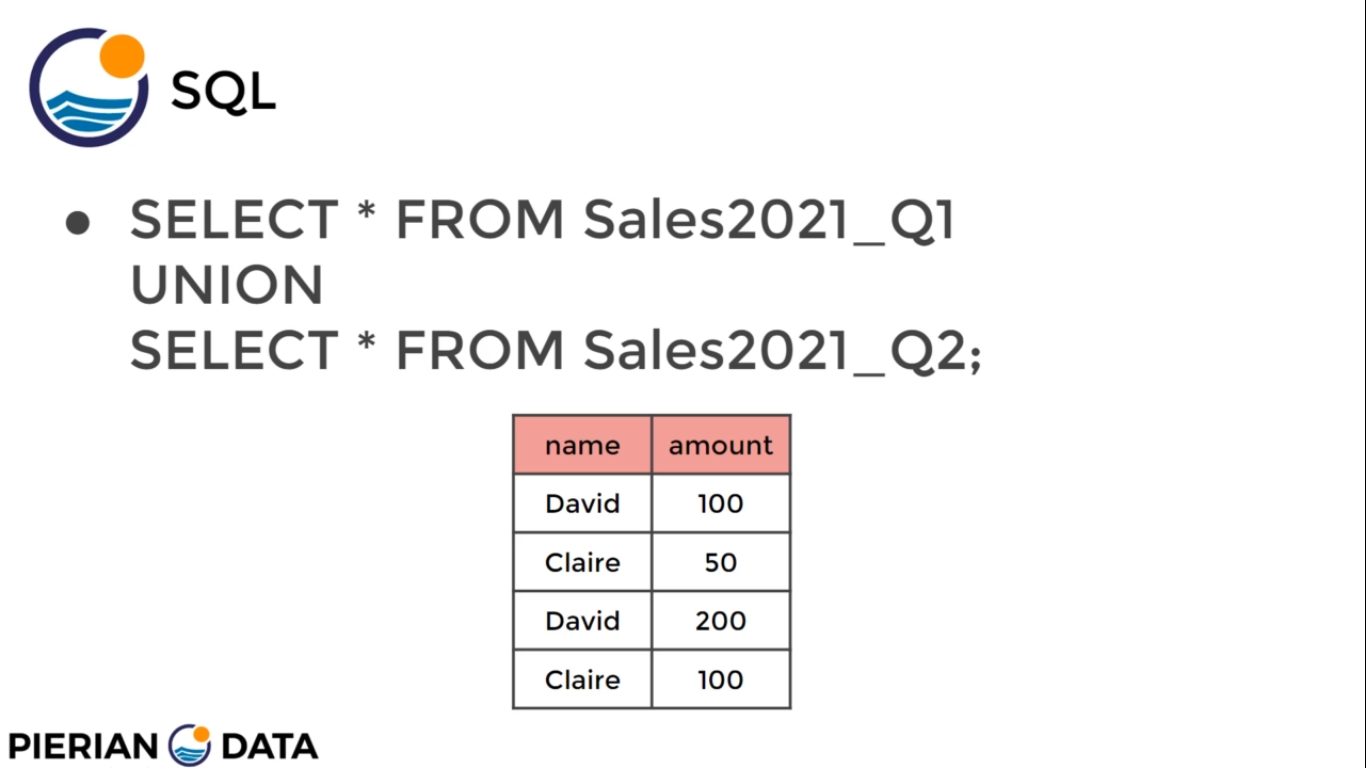
**ON film.film\_id = inventory.film\_id WHERE film.film\_id IS null**

**UNION**

// has 2 SELECT statment

// simple pasting on top of other





//CHALLANGE

SELECT email FROM address INNER JOIN customer

ON address.address\_id = customer.address\_id

WHERE address.district = 'California'

SELECT title,first\_name,last\_name FROM actor AS act INNER JOIN film\_actor AS fa

ON act.actor\_id = fa.actor\_id

INNER JOIN film

ON fa.film\_id = film.film\_id

WHERE first\_name = 'Nick' AND last\_name LIKE 'W%'

**SELF JOIN**

// join within the same table

// when all the info is in same table

// use two alias for same table

situation- make pairs of film with same length

**SELECT f1.title,f2.title,f1.length FROM film AS f1 INNER JOIN film AS f2**

**ON f1.film\_id != f2.film\_id AND f1.length = f2.length**

**ORDER BY f1.length**