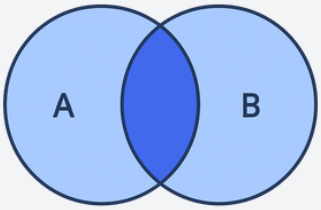
SQL Joins

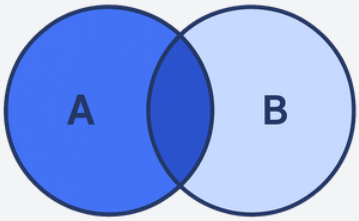
From Data with Dylan

What is a SQL Join?

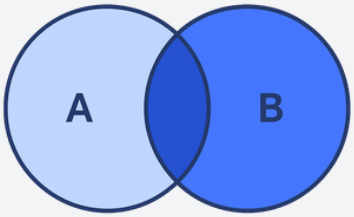
Combine data from different tables using a shared column



INNER JOIN



LEFT JOIN



RIGHT JOIN

EXAMPLE

Table: Customers

customer_id	customer_name	city
1	Alice Kim	Seattle
2	Brandon Lee	Portland
3	Chloe Nguyen	Seattle
4	Daniel Smith	Spokane

Table: Orders

order_id	customer_id	order_date	total_amount ($)
101	1	2024-01-12	89.99
102	2	2024-01-15	120.50
103	1	2024-02-05	42.00
104	3	2024-02-20	15.75
105	5	2024-02-21	60.00

These tables will be used for the practice problems below! The data is entirely fictional.

(All solutions will be provided in the last page!)

INNER JOIN

Definition: Returns rows where both tables match on a shared column

```
SELECT A.column_name, B.another_column_name
FROM A
INNER JOIN B
  ON A.shared_column = B.shared_column;
```

Returns only the rows where A and B have matching values in the shared column

Question 1: Write a SQL query to list each customer who has placed at least one order, along with the order ID and order total.

Hint: Return 3 columns and GROUP BY is not needed

Question 2: Retrieve the customers who have made orders that are greater than $50.00, and sort the results so the highest-paying customer appears first. Please also return the order IDs and the payments.

Hint: Filter using WHERE after joining, and sort using ORDER BY

LEFT JOIN

Definition: Returns all rows from the left table and matches from the right on a shared column

```
SELECT A.column_name, B.another_column_name
FROM A
LEFT JOIN B
  ON A.shared_column = B.shared_column;
```

This will return all rows from A, and the matching rows from B (with NULLs where no match exists)

Question 1: Retrieve all customers and their corresponding order IDs, including customers who did not place an order.

Hint: Order of how you specify the tables matters

Question 2: Retrieve all customers and the total amount of each order they made (return the order IDs as well). For customers with no orders, show NULL for the total.

RIGHT JOIN

Definition: Returns all rows from the right table and matches from the left on a shared column

```
SELECT A.column_name, B.another_column_name
FROM A
RIGHT JOIN B
  ON A.shared_column = B.shared_column;
```

This will return all rows from B, and the matching rows from A (with NULLs where no match exists)

Retrieve all orders, including those that do not have a matching customer, and show each order's ID alongside the customer name.

Hint: Customers is the left table

SUMMARY

INNER JOIN = Only matching rows
LEFT JOIN = All left rows + matches
RIGHT JOIN = All right rows + matches

Joins are used to combine data from multiple tables by matching rows through a related column

Matching occurs only on the columns that you specify on the ON clause

Order on how you specify the tables matter for LEFT JOIN and RIGHT JOIN

! In my opinion, **INNER JOIN** and **LEFT JOIN** are more commonly used than RIGHT JOINS!

RESOURCES

W3Schools is a great place to learn the basics!

If you want something more advanced, check out this real SQL interview question I solved.

SOLUTIONS

NOTE: The solutions I am providing are not the only correct answers. As long as you get the same output as me, then that's what matters!

INNER JOIN

1)

```
SELECT Customers.customer_name, Orders.order_id, Orders.total_amount
FROM Customers
INNER JOIN Orders
  ON Customers.customer_id = Orders.customer_id;
```

customer_name	order_id	total_amount
Alice Kim	101	89.99
Brandon Lee	102	120.50
Alice Kim	103	42.00
Chloe Nguyen	104	15.75

Answer Explanation: We must return three columns "customer_name" from the Customers table, as well as "order_id" and "total_amount" from the Orders table. Because we only want the customers who placed at least one order, we use the INNER JOIN, which will eliminate any customer who did not make an order. We join on the "customer_id" column because it exists in both tables and will form a relationship between them.

2)

```
SELECT Customers.customer_name, Orders.order_id
FROM Customers
INNER JOIN Orders
  ON Customers.customer_id = Orders.customer_id
WHERE Orders.total_amount > 50
ORDER BY Orders.total_amount DESC;
```

customer_name	order_id	total_amount
Brandon Lee	102	120.50
Alice Kim	101	89.99

Answer Explanation: This is a tricky problem! We must use an INNER JOIN because we only want to retrieve the customers who place an order. Because we want the customers who made an order more than $50, we use the WHERE clause to handle the condition. Finally, we use the ORDER BY clause to sort the results.

LEFT JOIN

1)

```
SELECT Customers.customer_name, Orders.order_id
FROM Customers
LEFT JOIN Orders
  ON Customers.customer_id = Orders.customer_id;
```

customer_name	order_id
Alice Kim	101
Alice Kim	103
Brandon Lee	102
Chloe Nguyen	104
Daniel Smith	NULL

Answer Explanation: The customer Daniel Smith has a customer_id of 4, but that customer_id does not exist in the Orders table. This means that Daniel Smith did not place an order, meaning that all the columns returned from the right table that correspond to his customer_id will be equal to NULL (in this case, it's "order_id"). All the other customers are returned with their corresponding order_ids because their customer_ids exist in the Orders table.

2)

```
SELECT Customers.customer_name, Orders.total_amount
FROM Customers
LEFT JOIN Orders
  ON Customers.customer_id = Orders.customer_id;
```

customer_name	order_id	total_amount
Alice Kim	103	42.00
Alice Kim	101	89.99
Brandon Lee	102	120.50
Chloe Nguyen	104	15.75
Daniel Smith	NULL	NULL

Answer Explanation: This is essentially the same as the previous LEFT JOIN problem, except that we're also returning the "total_amount" column from the right table. Since the customer Daniel Smith has a customer_id that does not exist in the Orders table, all of its columns from the right table will be NULL.

RIGHT JOIN

```
SELECT Customers.customer_name, Orders.order_id
FROM Customers
RIGHT JOIN Orders
  ON Customers.customer_id = Orders.customer_id;
```

customer_name	order_id
Alice Kim	101
Brandon Lee	102
Alice Kim	103
Chloe Nguyen	104
NULL	105

Answer Explanation: This is the exact opposite as our LEFT JOIN problem. Instead of returning all the customers, including the ones who have not placed an order, we are returning all the orders, including the ones that do not have customers associated with them. In this case, the order_id of 105 has a customer_name of NULL because its customer_id of 5 does not exist in the Customers table.

Scan to see me solve

real SQL interview questions!