# SQL Joins Mastery: Projects-Only Course

This course is focused entirely on hands-on projects and real-world SQL JOIN scenarios. You'll use SQL to combine, analyze, and extract insights from multiple tables. All projects are beginner-friendly and use simple e-commerce-style datasets.

## Sample Tables for Projects

Below are two sample tables: `customers` and `orders`. Use these for all the join operations in the examples and projects.

### SQL Setup: Create and Populate Tables

Paste the code below into your SQL compiler to create the working dataset:

-- Table: customers  
CREATE TABLE customers (  
 customer\_id INT PRIMARY KEY,  
 name VARCHAR(100),  
 city VARCHAR(50)  
);  
  
INSERT INTO customers (customer\_id, name, city) VALUES  
(1, 'Alice', 'New York'),  
(2, 'Bob', 'Los Angeles'),  
(3, 'Charlie', 'Chicago'),  
(4, 'Diana', 'Miami');  
  
-- Table: orders  
CREATE TABLE orders (  
 order\_id INT PRIMARY KEY,  
 customer\_id INT,  
 product VARCHAR(100),  
 amount DECIMAL(10, 2)  
);  
  
INSERT INTO orders (order\_id, customer\_id, product, amount) VALUES  
(101, 1, 'Laptop', 1200.00),  
(102, 2, 'Phone', 800.00),  
(103, 2, 'Tablet', 400.00),  
(104, 5, 'Monitor', 300.00);

## INNER JOIN - Examples

* List customers who have made at least one order

SELECT c.customer\_id, c.name, o.order\_id, o.product, o.amount  
FROM customers c  
INNER JOIN orders o ON c.customer\_id = o.customer\_id;

* Get order details with customer names only for matched records

SELECT o.order\_id, o.product, o.amount, c.name  
FROM orders o  
INNER JOIN customers c ON o.customer\_id = c.customer\_id;

* Show customers and their total amount spent

SELECT c.name, SUM(o.amount) AS total\_spent  
FROM customers c  
INNER JOIN orders o ON c.customer\_id = o.customer\_id  
GROUP BY c.name;

## LEFT JOIN - Examples

* Show all customers and their orders if they exist

SELECT c.customer\_id, c.name, o.order\_id, o.product  
FROM customers c  
LEFT JOIN orders o ON c.customer\_id = o.customer\_id;

* Find customers who haven't made any orders

SELECT c.name  
FROM customers c  
LEFT JOIN orders o ON c.customer\_id = o.customer\_id  
WHERE o.order\_id IS NULL;

* List all customers and count of orders (0 if none)

SELECT c.name, COUNT(o.order\_id) AS total\_orders  
FROM customers c  
LEFT JOIN orders o ON c.customer\_id = o.customer\_id  
GROUP BY c.name;

## RIGHT JOIN - Examples

* Show all orders and the customer who made them

SELECT o.order\_id, o.product, c.name  
FROM customers c  
RIGHT JOIN orders o ON c.customer\_id = o.customer\_id;

* Find unmatched orders (orders made by unknown customers)

SELECT o.order\_id, o.product  
FROM customers c  
RIGHT JOIN orders o ON c.customer\_id = o.customer\_id  
WHERE c.customer\_id IS NULL;

* List all orders with customers and show 'Unknown' if not found

SELECT o.order\_id, o.product, COALESCE(c.name, 'Unknown') AS customer\_name  
FROM customers c  
RIGHT JOIN orders o ON c.customer\_id = o.customer\_id;

## FULL OUTER JOIN - Examples

* Show all customers and all orders whether matched or not

SELECT c.name, o.product, o.amount  
FROM customers c  
FULL OUTER JOIN orders o ON c.customer\_id = o.customer\_id;

* Find customers without orders and orders without customers

SELECT c.name, o.product  
FROM customers c  
FULL OUTER JOIN orders o ON c.customer\_id = o.customer\_id  
WHERE c.customer\_id IS NULL OR o.customer\_id IS NULL;

* Summarize total spending per customer including unmatched orders

SELECT COALESCE(c.name, 'Unknown') AS customer\_name, SUM(o.amount) AS total\_spent  
FROM customers c  
FULL OUTER JOIN orders o ON c.customer\_id = o.customer\_id  
GROUP BY c.name;

## Final Project: Join Analytics Challenge

Use the `customers` and `orders` tables to answer the following:

1. List all customers along with the number of orders they’ve placed.
2. Find customers who haven’t placed any orders.
3. Find orders that were placed by unknown customers (those not in the customer list).
4. Calculate the total amount spent per customer.
5. Show a combined list of all customers and all orders (even unmatched ones).
6. Provide a summary: number of matched, unmatched, and total records when joining the tables.