🎬 [YouTube Video Script – Joins in MySQL]

📌 Title: MySQL Joins Explained Simply | INNER, LEFT, RIGHT, FULL, CROSS & SELF JOIN

👋 [Intro – 0:00 to 0:25]

🎙️ "Hey everyone, welcome back!
Today's topic is one of the most important concepts in SQL—**JOINS**.
We'll learn how to combine data from multiple tables using different types of joins, and when to use each one."

🔗 [What Are Joins? – 0:25 to 0:50]

🎙️ "In relational databases, data is split into multiple tables.
JOINs help us pull related data together—like combining customer info with their orders.
Let's break down the types of JOINs with easy examples."

🔵 [1. INNER JOIN – 0:50 to 1:30]

🎙️ "An **INNER JOIN** returns only the rows that exist in both tables.

📌 Analogy: Like finding people who are both in your phone contacts and Facebook friends.

```
SELECT c.name, o.order_date, o.total_amount
FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id;
```

✅ **Result**: Only customers who have placed orders."

🔙 [2. LEFT JOIN – 1:30 to 2:00]

🎙️ "A **LEFT JOIN** shows **all rows from the left table**, and matches from the right table.

📌 Analogy: A class roster with test scores—you see all students, even if some missed the test.

```
SELECT c.name, o.order_date, o.total_amount
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

✅ **Result**: All customers—even those without any orders (they'll have NULL in order columns)."

➡ [3. RIGHT JOIN – 2:00 to 2:30]

🎙 "A **RIGHT JOIN** is the opposite—**all records from the right table**, with matches from the left.

📌 Analogy: Showing all products and their sales, even if some products haven't sold.

```
SELECT c.name, o.order_date, o.total_amount
FROM customers c
RIGHT JOIN orders o ON c.customer_id = o.customer_id;
```

✅ **Result**: All orders—even ones without matching customer details."

🌐 [4. FULL OUTER JOIN – 2:30 to 3:00]

🎙 "A **FULL OUTER JOIN** shows **all records from both tables**, with matches where possible.

```
SELECT *
FROM table1
FULL OUTER JOIN table2
ON table1.id = table2.id;
```

✅ **Result**: All data from both tables—even unmatched rows.

🔔 Note: MySQL doesn't support FULL OUTER JOIN directly—you'll simulate it using UNION."

✖ [5. CROSS JOIN – 3:00 to 3:30]

🎙 "A **CROSS JOIN** creates the **Cartesian product** of two tables—every combination of records.

📌 Analogy: All possible combinations of pizza sizes and toppings.

```
SELECT p.product_name, s.size_name
FROM products p
CROSS JOIN sizes s;
```

⚠**Warning**: This can explode quickly! 10 × 10 = 100 rows."

♻ [6. SELF JOIN – 3:30 to 4:00]

🎙 "A **SELF JOIN** is when a table is joined with itself.

📌 Analogy: Matching employees to their managers from the same employees table.

```
SELECT e1.name AS employee_name, e2.name AS manager_name
FROM employees e1
LEFT JOIN employees e2 ON e1.manager_id = e2.employee_id;
```

Perfect for **hierarchical data** or comparing rows in the same table."

🧱 [Join Patterns – One-to-Many – 4:00 to 4:30]

🎙 "Here's a common **one-to-many** example:

One customer → many orders.

```
SELECT c.name, COUNT(o.order_id) as order_count
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

🧱 **[Join Patterns – Many-to-Many – 4:30 to 5:00]**

🎙 "Now for a **many-to-many** pattern:

Students enroll in many courses, and courses have many students.

```
SELECT s.student_name, c.course_name
FROM students s
INNER JOIN enrollments e ON s.student_id = e.student_id
INNER JOIN courses c ON e.course_id = c.course_id;
```

📣 **[Outro – 5:00 to 5:30]**

🎙 "That wraps up the **types of JOINs in MySQL**!
Understanding joins unlocks your ability to work with real-world data across tables.

If this helped you, drop a like, subscribe for more SQL tutorials, and tell us in the comments which JOIN you use the most!"