

→ Understanding how changing join conditions can help solve complex problems by altering the structure of tables and the resulting outcomes.

# Creation of Schema (Customer log) & Table – Customers -DDL commands

```
3 • use customerlog;
4
5 • CREATE TABLE Customers (
6     customer_id INT PRIMARY KEY,      -- Primary key
7     first_name VARCHAR(50),
8     last_name VARCHAR(50),
9     email VARCHAR(100)
10 );
```

# Creation of orders table

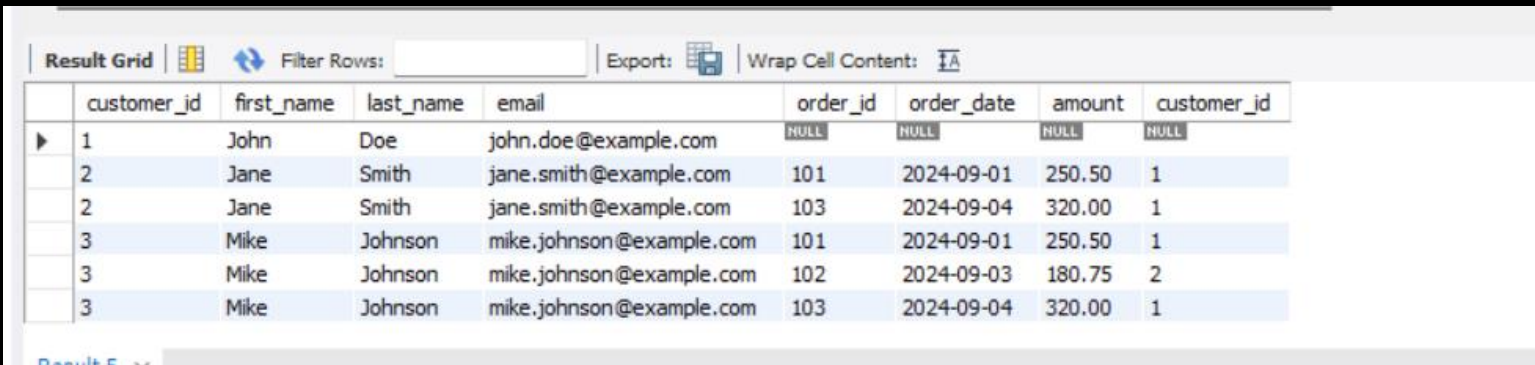
```
• CREATE TABLE Orders (  
    order_id INT PRIMARY KEY,          -- Primary key  
    order_date DATE,  
    amount DECIMAL(10, 2),  
    customer_id INT,                  -- Foreign key to Customers table  
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)  
);
```

# Insertion of values

```
20 • INSERT INTO Customers (customer_id, first_name, last_name, email)
21 VALUES
22 (1, 'John', 'Doe', 'john.doe@example.com'),
23 (2, 'Jane', 'Smith', 'jane.smith@example.com'),
24 (3, 'Mike', 'Johnson', 'mike.johnson@example.com');
25
26 • INSERT INTO Orders (order_id, order_date, amount, customer_id)
27 VALUES
28 (101, '2024-09-01', 250.50, 1), -- John Doe's order
29 (102, '2024-09-03', 180.75, 2), -- Jane Smith's order
30 (103, '2024-09-04', 320.00, 1), -- Another order for John Doe
31 (104, '2024-09-05', 450.00, 3); -- Mike Johnson's order
32
```

# Use joins by changing conditions $\lt\gt$ case1:

```
select * from Customers c left join orders o on c.customer_id>o.customer_id;
```




The screenshot shows a database query result grid. The grid has a toolbar at the top with options like 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with 9 columns: customer\_id, first\_name, last\_name, email, order\_id, order\_date, amount, and customer\_id. The table contains 6 rows of data. The first row shows a customer with id 1 and no orders. The subsequent rows show customers with orders where the customer\_id is greater than the order's customer\_id.

	customer_id	first_name	last_name	email	order_id	order_date	amount	customer_id
▶	1	John	Doe	john.doe@example.com	NULL	NULL	NULL	NULL
	2	Jane	Smith	jane.smith@example.com	101	2024-09-01	250.50	1
	2	Jane	Smith	jane.smith@example.com	103	2024-09-04	320.00	1
	3	Mike	Johnson	mike.johnson@example.com	101	2024-09-01	250.50	1
	3	Mike	Johnson	mike.johnson@example.com	102	2024-09-03	180.75	2
	3	Mike	Johnson	mike.johnson@example.com	103	2024-09-04	320.00	1

In this case, when  $c.customer\_id > o.customer\_id$  is used with a left join, the count of customer\_id from the left table remains intact. For rows where the condition is not satisfied, the right table's corresponding entries are filled with null values

## Case 2:

```
select * from Customers c left join orders o on c.customer_id=o.customer_id
where c.customer_id>o.customer_id
```



The screenshot shows a MySQL query result interface. At the top, there are tabs for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below these is a table with the following columns: customer\_id, first\_name, last\_name, email, order\_id, order\_date, amount, and customer\_id. The table is currently empty, indicating that no rows were returned by the query. At the bottom left, it says 'Result 8 x' and at the bottom right, there is a small icon and the letter 'R'.

customer_id	first_name	last_name	email	order_id	order_date	amount	customer_id
-------------	------------	-----------	-------	----------	------------	--------	-------------

In this case, when `c.customer_id = o.customer_id` is used with a left join, the count of `customer_id` from the left table remains intact. For rows where the condition is not satisfied, the right table's corresponding entries are filled with null values. Using the logic of order of execution where clause will do filtering and in output We will receive empty table.



- Thanks For Your time !!
- Happy SQLing 😊

