

Examining the order\_id of 2, we see it has a customer\_id of 2. So we go to the “customers” table where we see customer\_id 2 corresponds to ‘Jane Doe’.

The act of matching shared columns through different tables is called **joining**.

**SELECT \***

**FROM orders**

**JOIN customers**

**ON orders.customer\_id = customers.customer\_id**

**WHERE description = ‘Fashion Magazine’;**

1. Selects all columns from the combined table.
2. Specifies first table we want to look in, orders.
3. Specifies second table we with to join. So we want to JOIN orders and customers.
4. Match orders’ customer\_id column with customers’ customer\_id column.
5. Selects rows where description value = ‘Fashion Magazine’.

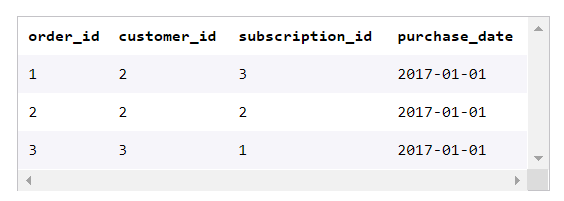
**SELECT \***

**FROM table1**

**LEFT JOIN table2**

**ON table1.c2 = table2.c2;**

Joins table1 and table2 by table1.c2 and table2.c2 columns. However, this keeps unmatched values from column1, regardless of EMPTY or not from column2.



**order\_id**, **customer\_id**, **subscription\_id** are all PRIMARY KEYS for their respective tables. They cannot be NULL, must be unique, and cannot have more than one primary key column.

When **customer\_id** and **subscription\_id** both appear in ‘order’ table, they are considered FOREIGN KEYS. When working with multiple FOREIGN KEYS (usually joining two tables by a common primary key), they have more descriptive names. Otherwise they are just called ‘id’.

**SELECT shirts.shirt\_colour, pants.pant\_colour**

**FROM shirts**

**CROSS JOIN pants;**

Displays shirt colours, pant colours columns next to each other. From shirts table, we CROSS JOIN pants table where each each shirt\_color is matched to each pant\_color, basically creating a table of all the possible shirt/pant colour combinations.

**SELECT \***

**FROM table1**

**UNION**

**SELECT \***

**FROM table2**

Stacks one table’s data onto another, literally. MUST HAVE SAME NUMBER OF COLUMNS. MUST HAVE SAME COLUMN DATA TYPES.

**WITH <new\_table> AS (**

**SELECT …**

**…**

**…**

**)**

**SELECT \***

**FROM <new\_table>**

**JOIN customers**

**ON customers.id = customers.name;**

WITH allows us to perform a separate query (such as aggregating customer’s subs) stored in a temporary table. We can later reference the temporary table.

