SQL CRASH COURSE PART 2
Query the database joining tables

Every time an order is placed, a new line gets created in the orders table. That table contains the total paid in the order. Additionally, for each unique product inside of the order, a new line gets created in the line_item table, storing the amount of products bought and their unitary price.

Another table, products, stores extra information about each individual product, including their brand, category and price.

In the following exercises we will query different tables in the database to understand better why the price for each product is stored twice, both in the line_item table and in the products table.

- **Query 1.** Our first query should return the "sku", "product_quantity", "date" and "unit_price" from the line_item table together with the "name" and the "price" of each product from the "products" table. We want only products present in both tables.

- **Query 2.** You might notice that the *unit_price* from the *line_item* table and the price from the *product* table is not the same. Let's investigate that! Extend your previous query by adding a column with the difference in price. Name that column *price_difference*.

- **Query 3.** Build a query that outputs the price difference that you just calculated, grouping products by category. Round the result.

- **Query 4.** Create the same query as before (calculating the price difference between the *line_item* and the *products* tables, but now grouping by brands instead of categories.

- **Query 5.** Let's focus on the brands with a big price difference: run the same query as before, but now limiting the results to only brands with an avg_price_dif of more than 50000. Order the results by *avg_price_dif* (bigger to smaller).

- Feel free to continue exploring this mysterious price difference on your own!

Let's explore Cancelled orders now. We will try to find relationships between the state of the orders and the category & brand of the products in those orders.

- **Query 6.** Query 6. We want to know the sku, product_quantity and date of all the orders, ordered by SKU. If it takes too long and/or the connection gets lost try first selecting only the first 50 results and then ordering by sku.

- **Query 7.** Add to the previous information about "*brand*" and "*manual_categories*" fields from the products table.

- **Query 8.** We want to know which brand and which categories are most frequent in Cancelled orders. Let's keep working on the same query: now we want to keep only Cancelled orders. Modify this query to group the results from the previous query, first by category and then by brand, adding in both cases a count and ordering by descending count. If it takes too long and/or the connection gets lost try putting a LIMIT command.