The CASE command

This lesson covers the CASE command, which is used to create an if-then-else logical condition in SQL.

Say you have a large dataset. You want to create four new categories and then assign each row of data to one of those categories. You can't go through the data and manually assign a category to each row because that would take too long! But you can use the SQL CASE command instead. CASE is useful whenever you need a logical statement with multiple conditions, each with its own resulting value if the condition is met.

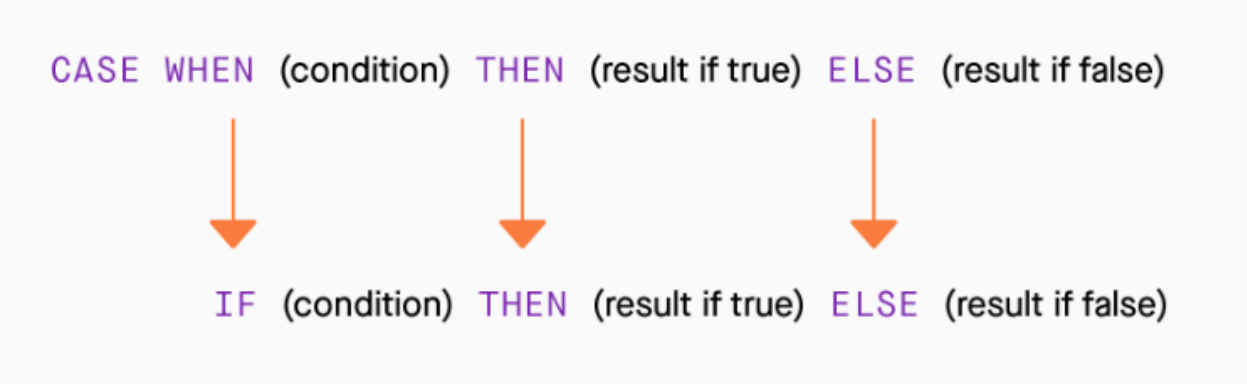
If-then-else logic

The CASE command in SQL follows if-then-else logic. If-then-else logic is something that you already use in everyday decision-making; for example, you might think "if the item is cheap, then I'll buy it, else it's expensive and I won't buy it," or "if I have time tonight, then I'll run the errand, else I won't have time and I'll do it tomorrow."

This logic can also be applied to data when you want to assign values to things based on certain conditions. Recall using nested IF functions in Excel. As you learned, those are used to create multiple conditions in a formula and then assign or return a value based on if those conditions are met. CASE is SQL's version of nested IF functions.

CASE

The CASE command is paired with WHEN, THEN, and ELSE and resembles if-then-else grammar.



The CASE command uses the following structure:

```
SELECT column_name(s),
CASE
WHEN condition1 THEN result1
WHEN condition2 THEN result2
WHEN condition3 THEN result3
ELSE result4
END
FROM table_name;
```