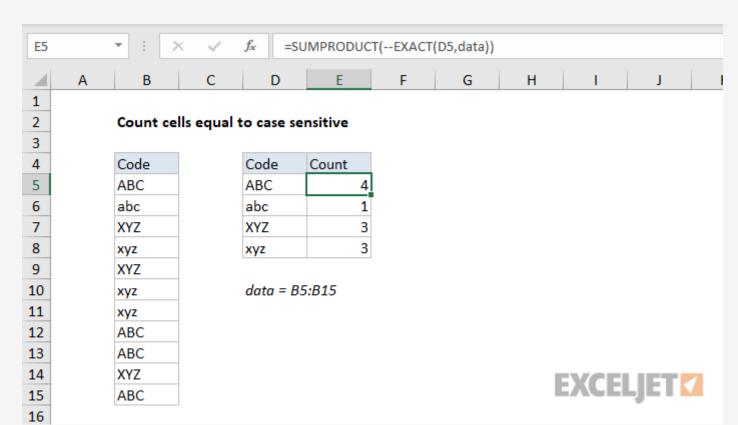
## Count cells equal to case sensitive



## Generic formula

```
= SUMPRODUCT( -- EXACT(value, range))
```

## Summary

To count cells that contain specific text, taking into account upper and lower case, you can use a formula based on the <u>EXACT function</u> together with the <u>SUMPRODUCT function</u>. In the example shown, E5 contains this formula, copied down:

```
= SUMPRODUCT( -- EXACT(D5, data))
```

Where "data" is the <u>named range</u> B5:B15. The result is a case-sensitive count of each code listed in column D.

## Explanation

In this example, the goal is to count codes in a case-sensitive way. The functions COUNTIF and COUNTIFS are both good options for counting text values, but neither is case-sensitive, so they can't be used to solve this problem. The solution is to use the <a href="EXACT function">EXACT function</a> to compare codes and the <a href="SUMPRODUCT function">SUMPRODUCT function</a> to add up the results.

The EXACT function takes two arguments: *text1* and *text2*. When *text1* and *text2* match exactly (considering upper and lower case), EXACT returns TRUE. Otherwise, EXACT returns FALSE:

```
= EXACT("abc", "abc") // returns TRUE
= EXACT("abc", "ABC") // returns FALSE
= EXACT("abc", "Abc") // returns FALSE
```

In the example shown, we have four codes in column D and some duplicated codes in B5:B15, the <u>named range</u> data. We want to count how many times each code in D5:D8 appears in B5:B15, and this count needs to be case-sensitive.

The formula in E5, copied down, is:

```
= SUMPRODUCT(( -- EXACT(D5, data)))

Working from the inside-out, we are using the <u>EXACT function</u> to compare the codes:
```

```
-- EXACT (D5, data)

EXACT compares the value in D5 ("ABC") to all values in B5:B15. Because we are giving EXACT
```

multiple values in the second argument, it returns multiple results. In total, EXACT returns 11 values (one for each code in B5:B15) in an <u>array</u> like this:

```
-- {TRUE; FALSE; FALSE; FALSE; FALSE; FALSE; TRUE; TRUE; FALSE; TRUE}
```

Each TRUE represents an exact match of "ABC" in B5:B15. Each FALSE represents a value

in B5:B15 that does not match "ABC". Because we want to *count* results, we use a <u>double-negative</u> (--) to convert TRUE and FALSE values into 1's and 0's. The resulting array looks like this:

```
{1;0;0;0;0;0;1;1;0;1} // 11 results
```

Using the double-negative like this is an example of <u>Boolean logic</u>, a technique for handling TRUE

```
and FALSE values like 1's and 0's. This array is delivered directly to the SUMPRODUCT function:
```

= SUMPRODUCT({1;0;0;0;0;0;1;1;0;1}) // returns 4

With just one array to process, SUMPRODUCT sums all numbers in the array and returns the final

result: 4.

Note: Because SUMPRODUCT can handle arrays natively, it's not necessary to use

Control+Shift+Enter to enter this formula.