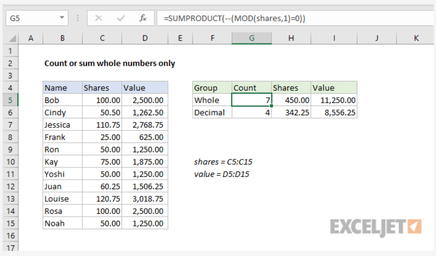
Count or sum whole numbers only



Generic formula

```
=SUMPRODUCT(--(MOD(range,1)=0))
```

Summary

To count or sum values that are whole numbers, you can use a formula based on the SUMPRODUCT and MOD functions. In the example shown, the formula in G5 is:

```
=SUMPRODUCT(--(MOD(shares,1)=0))
```

where **shares** is the named range C5:C15.

Explanation

In this example, the goal is to get a count of people that hold shares in whole numbers. For example, Bob holds 100 shares (even), so he *should* be included in the whole number count, while Cindy holds 50.5 shares, so she *should not* be included in the whole number count.

The first problem is how to determine whole numbers. This can be done with the INT, TRUNC, or MOD functions as explained in detail here. In this example shown above, we are using the MOD function option:

```
=MOD(A1,1)=0 // TRUE for whole numbers
```

Now that we know how to test for a whole number, how can we use this approach to get a *count* of whole numbers? You might at first be tempted to use the COUNTIF or COUNTIFS functions. However, these functions won't let you use an array* in place of the *range* argument, so COUNTIF won't work:

```
=COUNTIF(MOD(shares,1),0) // won't work!
```

**MOD(shares,1) is technically an array operation that returns an array of values. See this article for more information about limitations in COUNTIF, SUMIF, etc.*

Instead, we need a way to work with the array directly with Boolean logic. Boolean logic is a technique for building formulas that take advantage of the fact that TRUE = 1, and FALSE = 0 in math operations. In the example shown, this is what the formula in G5 does:

```
=SUMPRODUCT(--(MOD(shares,1)=0))
```

Working from the inside out, we first run all values through the MOD test shown above:

```
=MOD(shares,1)=0 // test all values
```

Because there are eleven values in **shares** (C5:C15), we get eleven results in an array like this:

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

In this array, TRUE values represent a whole number, and FALSE values represent a decimal number. Next, we need to convert the TRUE and FALSE values to 1s and 0s. To do this, we use a double-negative (--):

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

This operation returns a numeric array composed only of 1s and 0s:

```
{1;0;0;1;1;1;1;0;0;1;1}
```

This is exactly what we need to count whole numbers. This array is returned directly to the SUMPRODUCT function:

```
=SUMPRODUCT({1;0;0;1;1;1;1;0;0;1;1}) // returns 7
```

With just one array to process, SUMPRODUCT returns the sum of all items in the array, 7, which is the count of whole numbers in the range C5:C15.

Count decimal values

To change the formula to count numbers with decimal values, we only need to change the logical operator in the MOD snippet from an equal sign (=) to the not equal to (<>) operator. The formula in G6:

```
=SUMPRODUCT(--(MOD(shares,1)<>0)) // returns 4
```

Note the only change to the formula is the logical operator.

Sum whole number shares

To sum whole numbers only, we need to extend the formula a bit by multiplying the Boolean array explained above by the values in the named range **shares.** The formula in H5 calculates the total number of shares in the whole number group:

```
=SUMPRODUCT(--(MOD(shares,1)=0)*shares)
```

Notice the formula is *almost* the same as above. The result is that the zero values effectively cancel out the shares in the decimal group:

```
=SUMPRODUCT(--(MOD(shares,1)=0)*shares)
=SUMPRODUCT({1;0;0;1;1;1;1;0;0;1;1}*
{100;50.5;110.75;25;50;75;50;60.25;120.75;100;50})
=SUMPRODUCT({100;0;0;25;50;75;50;0;0;100;50})
=450
```

Sum whole number share values

To sum the values associated with whole number shares, we need to adjust the formula again. This time instead of multiplying the Boolean array by **shares**, we multiply by **value**. The formula in I5 is:

```
=SUMPRODUCT(--(MOD(shares,1)=0)*value)
```

The formula is solved in exactly the same way:

```
=SUMPRODUCT(--(MOD(shares,1)=0)*value)
=SUMPRODUCT({1;0;0;1;1;1;1;0;0;1;1}*
{2500;1262.5;2768.75;625;1250;1875;1250;1506.25;3018.75;2500;1250})
=SUMPRODUCT({2500;0;0;625;1250;1875;1250;0;0;2500;1250})
=11250
```

As above, the zero values in the Boolean array cancel out values for non-whole number shares, and the final result returned by SUMPRODUCT is 11250.