# Multi-column SUMPRODUCT with LAMBDA

# Suppose we have some sales data

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | Product | Date | Quantity | Price | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ 5.22 | |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ 7.47 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ 6.54 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ 7.19 | |
| 7 | | Pears | 2024-01-18 | 10 | $ 5.23 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ 6.91 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ 5.40 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ 5.66 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ 7.98 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ 8.40 | |
| 13 | | | | | | |

# The simple way to calculate the total amount is to multiply the quantity by the price on each row, then sum the new column

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | Product | Date | Quantity | Price | Amount | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ 5.22 | $ 57.42 | =D3*E3 |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ 7.47 | $ 14.94 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ 6.54 | $ 78.48 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ 7.19 | $ 93.47 | |
| 7 | | Pears | 2024-01-18 | 10 | $ 5.23 | $ 52.30 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ 6.91 | $ 69.10 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ 5.40 | $ 75.60 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ 5.66 | $ 84.90 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ 7.98 | $ 23.94 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ 8.40 | $ 126.00 | |
| 13 | | | | | Total Amount $ | $ 676.15 | =SUM(F3:F12) |
| 14 | | | | | | | |

# This can also be slightly simplified with an array formula to create the new column. The sum now refers to the spilled range using F3#

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | Product | Date | Quantity | Price | Amount | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ 5.22 | $ 57.42 | =D3:D12*E3:E12 |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ 7.47 | $ 14.94 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ 6.54 | $ 78.48 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ 7.19 | $ 93.47 | |
| 7 | | Pears | 2024-01-18 | 10 | $ 5.23 | $ 52.30 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ 6.91 | $ 69.10 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ 5.40 | $ 75.60 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ 5.66 | $ 84.90 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ 7.98 | $ 23.94 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ 8.40 | $ 126.00 | |
| 13 | | | | | Total Amount $ | $ 676.15 | =SUM(F3#) |
| 14 | | | | | | | |

# We can also skip calculating the new column and calculate the total amount directly using the SUMPRODUCT function

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | Product | Date | Quantity | Price | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ 5.22 | |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ 7.47 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ 6.54 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ 7.19 | |
| 7 | | Pears | 2024-01-18 | 10 | $ 5.23 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ 6.91 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ 5.40 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ 5.66 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ 7.98 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ 8.40 | |
| 13 | | | | | | |
| 14 | | | | Total Amount | $ 676.15 | =SUMPRODUCT(D3:D12,E3:E12) |
| 15 | | | | | | |

We can pass 1 or more arrays (ranges) to SUMPRODUCT, separated by commas. SUMPRODUCT multiplies each element by the corresponding elements in the other arrays, then sums the result.

# However, if we want to just pass one multiple-column array, it will not calculate the products row-wise

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | Product | Date | Quantity | Price | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ 5.22 | |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ 7.47 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ 6.54 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ 7.19 | |
| 7 | | Pears | 2024-01-18 | 10 | $ 5.23 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ 6.91 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ 5.40 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ 5.66 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ 7.98 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ 8.40 | |
| 13 | | | | | | |
| 14 | | | | Total Amount | $ 171.00 | =SUMPRODUCT(D3:E12) |
| 15 | | | | | | |
| 16 | | | | Comparison | 171 | =SUM(D3:E12) |
| 17 | | | | | | |

Because we only passed 1 argument to SUMPRODUCT, no multiplication happens. It simply takes the SUM of all the elements in the array. This is identical to using the SUM function.

# If we want to mimic SUMPRODUCT behavior on a 2D array, we can wrap PRODUCT with BYROW, and wrap the result in SUM

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | Product | Date | Quantity | Price | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ 5.22 | |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ 7.47 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ 6.54 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ 7.19 | |
| 7 | | Pears | 2024-01-18 | 10 | $ 5.23 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ 6.91 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ 5.40 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ 5.66 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ 7.98 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ 8.40 | |
| 13 | | | | | | |
| 14 | | | | Total Amount | $ 676.15 | =SUM(BYROW(D3:E12,PRODUCT)) |
| 15 | | | | | | |
| 16 | | | | Comparison | $ 676.15 | =SUMPRODUCT(D3:D12,E3:E12) |

PRODUCT within BYROW multiplies the values on each row. SUM then sums the result

# If we want to use this 2D array syntax frequently, we can create a LAMBDA

```
SUMPRODUCT2 = LAMBDA(array,
    SUM(BYROW(array, PRODUCT))
);
```

| | A | B | C | D | | | |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | Product | Date | Quantity | Price | | |
| 3 | | Gooseberries | 2023-06-26 | 11 | $ | 5.22 | |
| 4 | | Gooseberries | 2024-05-09 | 2 | $ | 7.47 | |
| 5 | | Blackberries | 2024-03-23 | 12 | $ | 6.54 | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ | 7.19 | |
| 7 | | Pears | 2024-01-18 | 10 | $ | 5.23 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ | 6.91 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ | 5.40 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ | 5.66 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ | 7.98 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ | 8.40 | |
| 13 | | | | | | | |
| 14 | | | | Total Amount | $ | 676.15 | =SUMPRODUCT2(D3:E12) |
| 15 | | | | | | | |
| 16 | | | | Comparison | $ | 676.15 | =SUMPRODUCT(D3:D12,E3:E12) |
| 17 | | | | | | | |

# In future, we might want to perform the multiplication BYCOL, so we can extend the LAMBDA

```
SUMPRODUCT2 = LAMBDA(array, axis,
    IF(axis = 0,
        SUM(BYROW(array, PRODUCT)),
        SUM(BYCOL(array, PRODUCT))
    )
);
```

| | A | B | C | D | | E | |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | Product | Date | Quantit | | | |
| 3 | | Gooseberries | 2023-06-26 | | | | |
| 4 | | Gooseberries | 2024-05-09 | | | | |
| 5 | | Blackberries | 2024-03-23 | | | | |
| 6 | | Blackberries | 2024-09-30 | 13 | $ | 7.19 | |
| 7 | | Pears | 2024-01-18 | 10 | $ | 5.23 | |
| 8 | | Blackberries | 2024-04-27 | 10 | $ | 6.91 | |
| 9 | | Boysenberries | 2024-09-24 | 14 | $ | 5.40 | |
| 10 | | Boysenberries | 2024-03-21 | 15 | $ | 5.66 | |
| 11 | | Snozzberries | 2024-01-06 | 3 | $ | 7.98 | |
| 12 | | Raspberries | 2024-01-30 | 15 | $ | 8.40 | |
| 13 | | | | | | | |
| 14 | | | | Total Amount | $ | 676.15 | =SUMPRODUCT2(D3:E12,0) |
| 15 | | | | | | | |
| 16 | | | | Comparison | $ | 676.15 | =SUMPRODUCT(D3:D12,E3:E12) |
| 17 | | | | | | | |

# Most use of this function will be BYROW, so we can make the axis argument optional

```
IFOMITTED = LAMBDA(arg,then,IF(ISOMITTED(arg),then,arg));
SUMPRODUCT2 = LAMBDA(array, [axis],
    LET(
        _axis, IFOMITTED(axis,0),
        IF(_axis=0,
            SUM(BYROW(array, PRODUCT)),
            SUM(BYCOL(array, PRODUCT))
        )
    )
);
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 10 | Boysenberries | 2024-03-21 | 15 | $ | 5.66 | |
| 11 | Snozzberries | 2024-01-06 | 3 | $ | 7.98 | |
| 12 | Raspberries | 2024-01-30 | 15 | $ | 8.40 | |
| 13 | | | | | | |
| 14 | | | Total Amount | $ | 676.15 | =SUMPRODUCT2(D3:E12) |
| 15 | | | | | | |
| 16 | | | Comparison | $ | 676.15 | =SUMPRODUCT(D3:D12,E3:E12) |
| 17 | | | | | | |

The axis argument is made optional by wrapping it in square brackets.

Now, if we omit the axis argument, it will default to BYROW.

# But this can still be simplified

From this

```
IFOMITTED = LAMBDA(arg,then,IF(ISOMITTED(arg),then,arg));
SUMPRODUCT2 = LAMBDA(array, [axis],
    LET(
        _axis, IFOMITTED(axis,0),
        IF(_axis=0,
           SUM(BYROW(array, PRODUCT)),
           SUM(BYCOL(array, PRODUCT))
        )
    )
);
```

To this

```
IFOMITTED = LAMBDA(arg,then,IF(ISOMITTED(arg),then,arg));
SUMPRODUCT2 = LAMBDA(array, [axis],
    LET(
        _axis, IFOMITTED(axis,0),
        _direction, IF(_axis=0, BYROW, BYCOL),
        SUM(_direction(array, PRODUCT))
    )
);
```

The _direction variable is now a function – either BYROW or BYCOL, depending on the value of _axis.

# Which can be further condensed

From this

```
IFOMITTED = LAMBDA(arg,then,IF(ISOMITTED(arg),then,arg));
SUMPRODUCT2 = LAMBDA(array, [axis],
    LET(
        _axis, IFOMITTED(axis,0),
        _direction, IF(_axis=0, BYROW, BYCOL),
        SUM(_direction(array, PRODUCT))
    )
);
```

To this

```
IFOMITTED = LAMBDA(arg,then,IF(ISOMITTED(arg),then,arg));
SUMPRODUCT2 = LAMBDA(array, [axis],
    SUM(IF(IFOMITTED(axis,0)=0, BYROW, BYCOL)(array, PRODUCT))
);
```

Using a variable's name is the same as using the calculation for that variable!

# Takeaways:

1. If an Excel function doesn't do something we would like it to do, we can create a LAMBDA with the new behavior
2. Functions can be assigned to LET variables
3. Anywhere we use a LET variable, we can use the calculation for that LET variable. Including in place of function calls