MS 365 Excel Basics #9

Data Analysis & Single Cell Reporting Formulas: GROUPBY & PIVOTBY Dynamic Spilled Array Functions

Table of Contents

Dynamic Spilled Array Functions seen in this video 2

Characteristics of Dynamic Spilled Array Formulas (DSAF) 2

SEQUENCE Function to generate sequences of Numbers 3

Format Dynamic Spilled Arrays with Conditional Formatting 3

History Of Reporting with Single Cell Reporting Formulas 6

PIVOTBY Array Function to Create Reports 7

GROUPBY Array Function to Create Reports 8

When to use PivotTable & When to Use PIVOTBY & GROUPBY 9

Example of Instant Update in the GROUPBY Function 9

Array Functions to Building Single Cell Array Formulas: A Revolution in Excel 10

Example 1: GROUPBY Function to Create a Frequency Distribution 11

Note about Excel Table Formula Nomenclature (Excel Table Formulas) 13

Eta-LAMBDA Functions 14

field_relationship argument in GROUPBY 14

Example 2: GROUPBY Function & ARRAYTOTEXT Function 15

Example 3: GROUPBY Function & MEDIAN Function 16

Example 4: XLOOKUP Helper Column in *values* argument of GROUPBY 17

Example 5: Use HSTACK to create side-by-side calculations in a GROUPBY Report 18

Example 6: Payroll Helper Column Formula in values argument of GROUPBY 19

Example 7: PIVOTBY function to create a % of Parent Row Total Report 20

Example 8: PIVOTBY Function to Create a 3-in-1 Report with % Of Column Total, % Of Row Total & % Of Grand Total Reports 21

Example 9: GROUPBY to Help Create an X-Y Scatter Chart 22

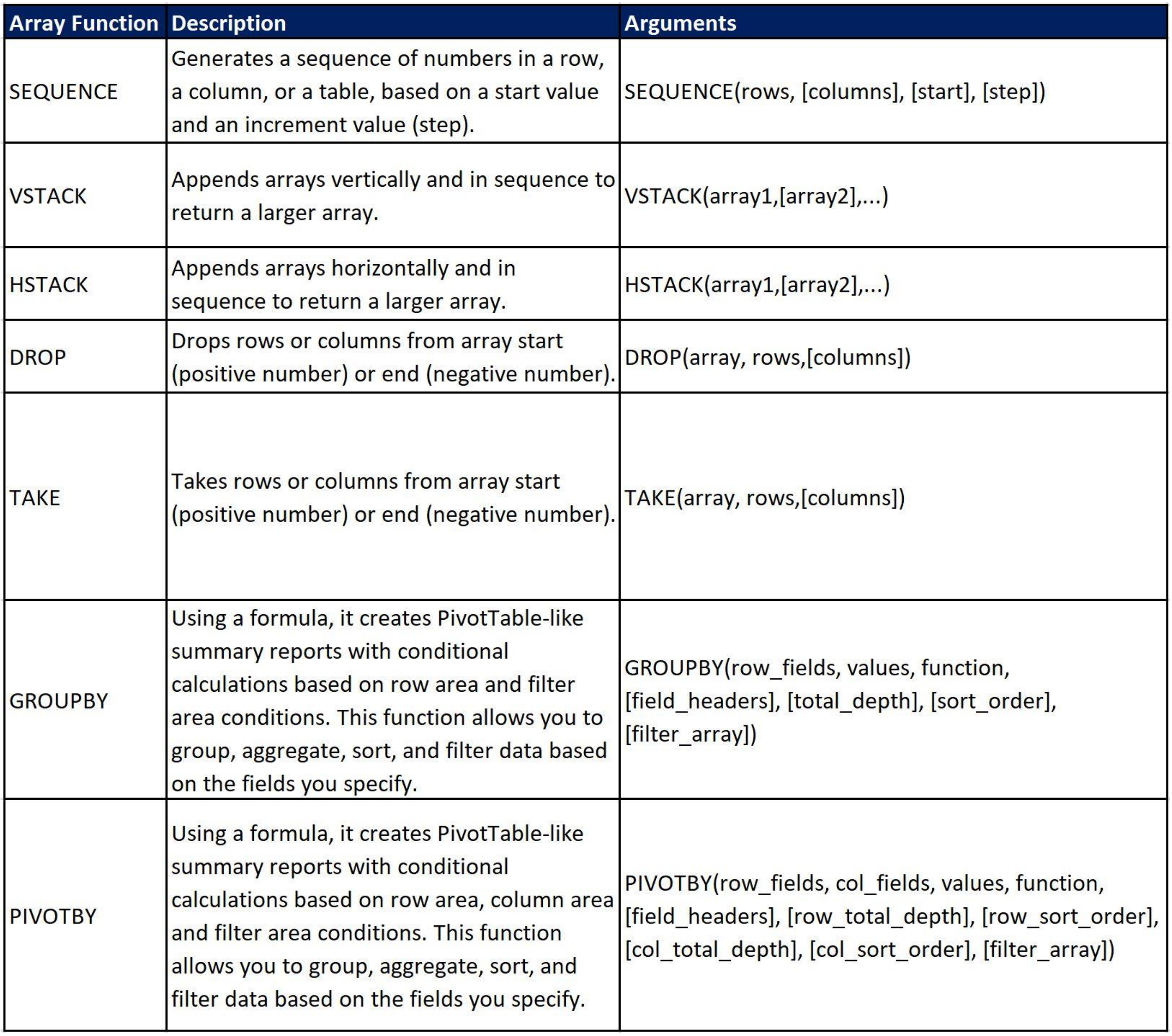
Example 10: GROUPBY worked quickly over 990,000 rows of data and the data did not have to be stored in PivotTable Cache 23

Bonus: Logical Formulas to Apply Conditional Formatting to a Single Cell Report 24

LET function to Define Variables (not covered in class) 25

LAMBDA Function (not covered in this video) 26

Dynamic Spilled Array Functions seen in this video



Array Function	Description	Arguments
SEQUENCE	Generates a sequence of numbers in a row, a column, or a table, based on a start value and an increment value (step).	SEQUENCE(rows, [columns], [start], [step])
VSTACK	Appends arrays vertically and in sequence to return a larger array.	VSTACK(array1,[array2],...)
HSTACK	Appends arrays horizontally and in sequence to return a larger array.	HSTACK(array1,[array2],...)
DROP	Drops rows or columns from array start (positive number) or end (negative number).	DROP(array, rows,[columns])
TAKE	Takes rows or columns from array start (positive number) or end (negative number).	TAKE(array, rows,[columns])
GROUPBY	Using a formula, it creates PivotTable-like summary reports with conditional calculations based on row area and filter area conditions. This function allows you to group, aggregate, sort, and filter data based on the fields you specify.	GROUPBY(row_fields, values, function, [field_headers], [total_depth], [sort_order], [filter_array])
PIVOTBY	Using a formula, it creates PivotTable-like summary reports with conditional calculations based on row area, column area and filter area conditions. This function allows you to group, aggregate, sort, and filter data based on the fields you specify.	PIVOTBY(row_fields, col_fields, values, function, [field_headers], [row_total_depth], [row_sort_order], [col_total_depth], [col_sort_order], [filter_array])

Characteristics of Dynamic Spilled Array Formulas (DSAF)

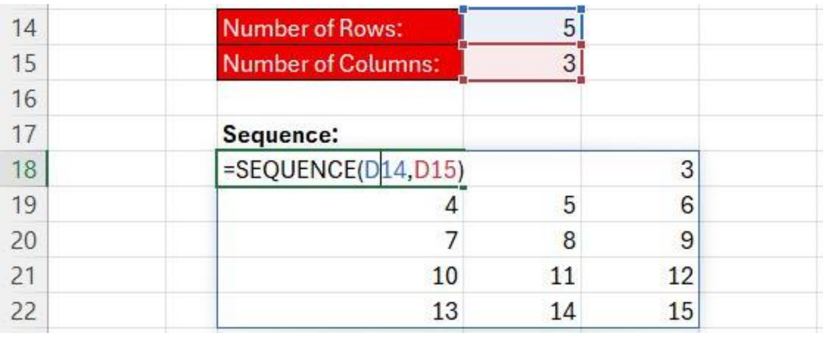
Characteristics of Dynamic Spilled Array Formula:

1. The formula lives in the top cell.
2. Spilled values spill down and to the right.
3. To edit a dynamic spilled array formula, you edit the formula in the top left cell.
4. Cells below the top cell do not contain values. All values emanate from the top cell.
5. Even though the values below the top cell do not live in the cell, you can refer to a value in any of the spilled range with a cell reference.
6. If a cell in the path of the spilled values contains a value, you will get a #SPILL! error.
7. You refer to a spilled range of values using the top cell address and the spilled range operator: # symbol, like E5#.
8. The most amazing characteristic of dynamic spilled array formulas is that when the source data changes and the resultant array expands (or contracts), the spilled range dynamically updates.
9. Not all worksheet functions can spill results. Aggregate functions like SUM, AVERAGE, AND, OR, and SUMPRODUCT cannot deliver spilled arrays.
10. Spilled array formulas are not allowed in Excel Tables.

11. Some function arguments do not allow function argument array operations, such as:
    a. Range argument of the functions SUMIF, COUNTIF, and AVERAGEIF.
    b. Criteria_range argument of the functions SUMIFS, COUNTIFS, AVERAGEIFS, MINIFS, & MAXIFS.
    c. First argument of the functions SUMIFS, AVERAGEIFS, MINIFS, and MAXIFS COUNTIFS.
    d. lookup_value argument in VLOOKUP and HLOOKUP.
12. Almost all array operations involve operations on multiple formula inputs, such as C5:C7*D9, where a column of values is multiplied by a single value. The exceptions are array functions like SEQUENCE, RANDARRAY, and MUNIT, which are each programmed to generate an array of answers from a single input (for example, =SEQUENCE(3) = {1;2;3}).
13. Different than a PivotTable, which has automatic dynamic formatting that follows the report when expands or contracts, Dynamic Spilled Array Formulas do not have automatic formatting.
    a. Use Conditional Formatting: either built-in, or logical formula driven conditional formatting.

SEQUENCE Function to generate sequences of Numbers

Use SEQUENCE to generate a rectangle sequence of numbers:

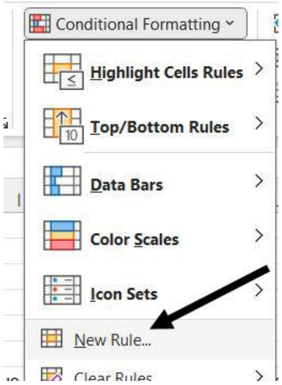


14		Number of Rows:	5	
15		Number of Columns:	3	
16				
17		Sequence:		
18		=SEQUENCE(D14,D15)		3
19		4	5	6
20		7	8	9
21		10	11	12
22		13	14	15

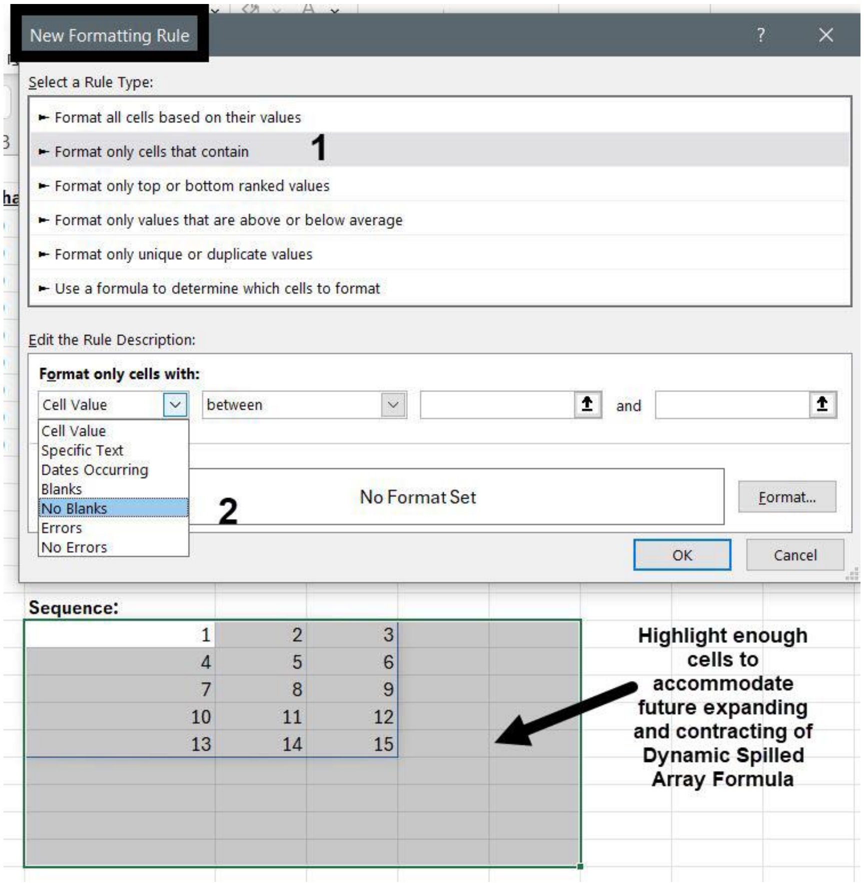
Format Dynamic Spilled Arrays with Conditional Formatting

Whereas, where a PivotTable has automatic formatting that adjusts when you pivot your report, Dynamic Spilled Array Formula have no formatting abilities. If the Dynamic Spilled Array Formula will not expand or contract, then you can just use style formatting like fill color, font color, bold and borders. If you want the report formatting to automatically adjust to future expanding or contacting, you use Conditional Formatting with either the built-in options or the option to build your own logical formula. Before you add Conditional Formatting to a Dynamic Spilled Array Formula, you must always select a range that will accommodate any anticipated expanding or contacting of the formula, then you apply the Conditional Formatting. In this example, you want to apply a border format if the cell contains content. To accomplish this, you can use a built-in option, as shown in the following six steps.

Step 1: Highlight a range large enough to accommodate any future expanding or contracting of the Dynamic Spilled Array Formula. Then in the Home Ribbon tab, Styles group, click the Conditional Formatting dropdown arrow, then click New Rule.



Step 2: In the New Formatting Rule dialog box: 1) in the Select a Rule Type area, click Format only cells that contain, 2) in the Edit the Rule Description area, click the Format only cells with dropdown arrow and then select No Blanks. “No Blanks” is a logical test that evaluates to TRUE when a cell has content and evaluates to FALSE when the cell is empty. When the logical test evaluates to TRUE, the conditional formatting will be applied.



New Formatting Rule

Select a Rule Type:

- Format all cells based on their values
- Format only cells that contain 1
- Format only top or bottom ranked values
- Format only values that are above or below average
- Format only unique or duplicate values
- Use a formula to determine which cells to format

Edit the Rule Description:

Format only cells with:

Cell Value | between | | and |

- Cell Value
- Specific Text
- Dates Occurring
- Blanks
- No Blanks
- Errors
- No Errors

2

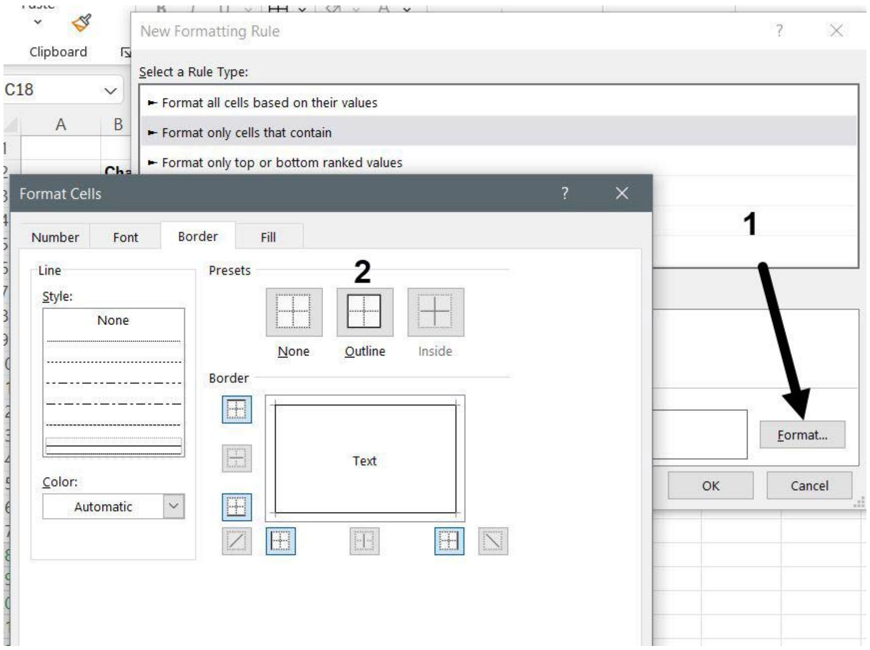
No Format Set

Sequence:

1	2	3		
4	5	6		
7	8	9		
10	11	12		
13	14	15		

Highlight enough cells to accommodate future expanding and contracting of Dynamic Spilled Array Formula

Step 3: In the New Formatting Rule dialog box: 1) click the Format button to determine what formatting should be applied, 2) In the Format Cells dialog box click the Border tab and then in the Preset area select Outline. This cell outline will be applied when the logical test evaluates to TRUE.



New Formatting Rule

Select a Rule Type:

- Format all cells based on their values
- Format only cells that contain
- Format only top or bottom ranked values

1

Format Cells

Number | Font | Border | Fill

Line

Style: None

Color: Automatic

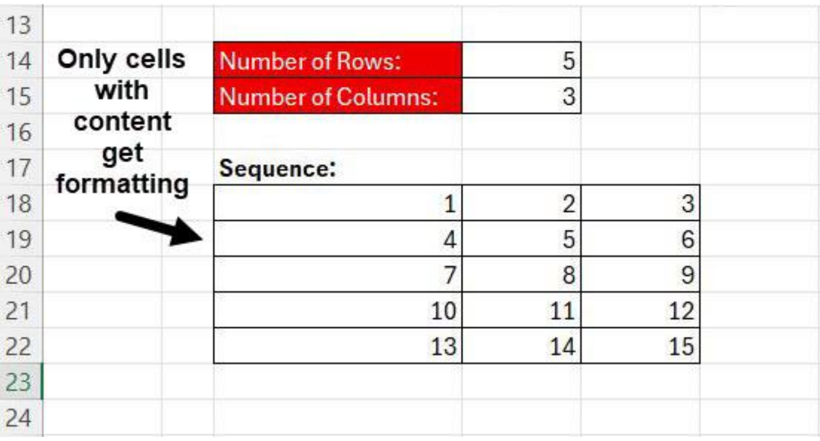
Presets 2

None | Outline | Inside

Border

Text

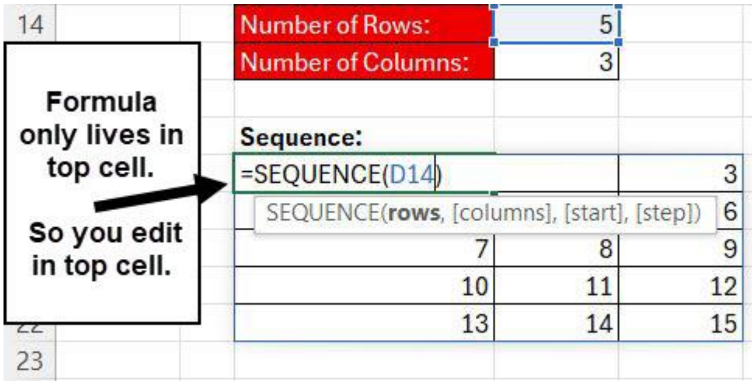
Step 4: After you click OK on the Format Cells and the New Formatting Rule dialog boxes, only the cells with content get the border formatting.



Only cells with content get formatting

13			
14	Number of Rows:	5	
15	Number of Columns:	3	
16			
17	Sequence:		
18	1	2	3
19	4	5	6
20	7	8	9
21	10	11	12
22	13	14	15
23			
24			

Step 5: Edit the SEQUENCE function in the top cell of the array so that the function only delivers the row numbers 1 to 5.

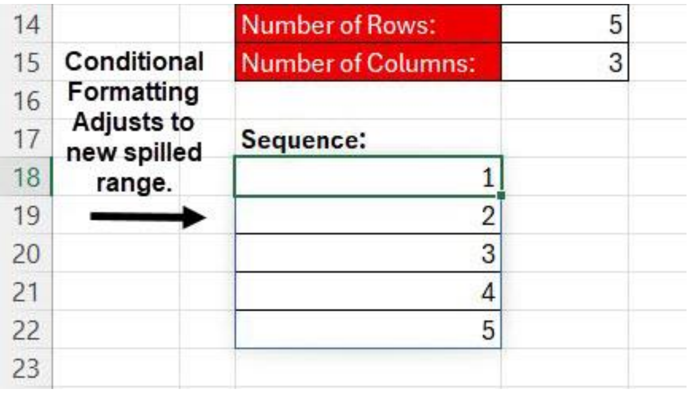


Formula only lives in top cell.

So you edit in top cell.

14	Number of Rows:	5	
	Number of Columns:	3	
	Sequence:		
	=SEQUENCE(D14)		3
			6
	7	8	9
	10	11	12
	13	14	15
23			

Step 6: When the Dynamic Spilled Array Formula spills into a smaller range, the conditional formatting adjust to format just the range with the spilled results.



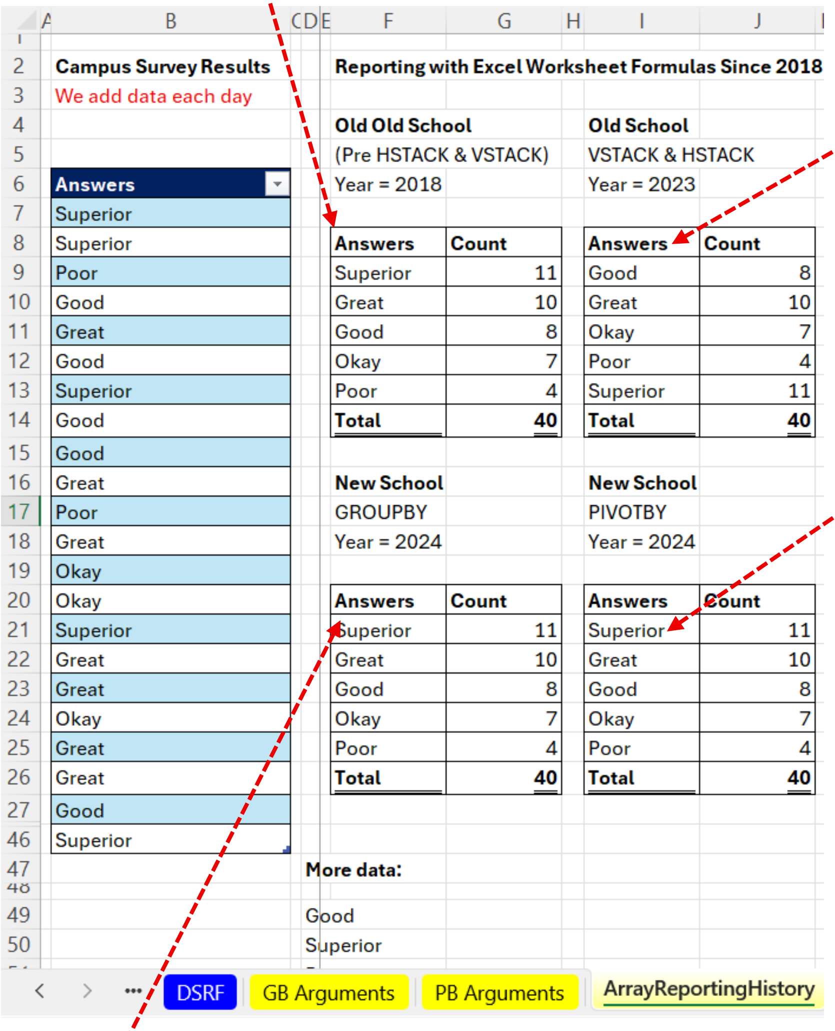
Conditional Formatting Adjusts to new spilled range.

14	Number of Rows:	5
15	Number of Columns:	3
16		
17	Sequence:	
18	1	
19	2	
20	3	
21	4	
22	5	
23		

History Of Reporting with Single Cell Reporting Formulas

2018

```
=LET(
Data,B7:B46,
UniqueList,UNIQUE(Data),
Counts,COUNTIFS(Data,UniqueList),
ReportHeaders,CHOOSE({1,2},B6,"Count"),
ReportMid,SORT(CHOOSE({1,2},UniqueList,Counts),2,-1),
ReportTotalRow,CHOOSE({1,2},"Total",COUNTA(Data)),
NumRowsInReport,COUNTA(UniqueList)+2,
SeqRowNums,SEQUENCE(NumRowsInReport),
SWITCH(SeqRowNums,
     1,ReportHeaders,
     NumRowsInReport,ReportTotalRow,
     INDEX(ReportMid,SeqRowNums-1,{1,2})))
```



Campus Survey Results

We add data each day

Row	Answers
7	Superior
8	Superior
9	Poor
10	Good
11	Great
12	Good
13	Superior
14	Good
15	Good
16	Great
17	Poor
18	Great
19	Okay
20	Okay
21	Superior
22	Great
23	Great
24	Okay
25	Great
26	Great
27	Good
46	Superior

Reporting with Excel Worksheet Formulas Since 2018

Old Old School
(Pre HSTACK & VSTACK)
Year = 2018

Answers	Count
Superior	11
Great	10
Good	8
Okay	7
Poor	4
Total	**40**

Old School
VSTACK & HSTACK
Year = 2023

Answers	Count
Good	8
Great	10
Okay	7
Poor	4
Superior	11
Total	**40**

New School
GROUPBY
Year = 2024

Answers	Count
Superior	11
Great	10
Good	8
Okay	7
Poor	4
Total	**40**

New School
PIVOTBY
Year = 2024

Answers	Count
Superior	11
Great	10
Good	8
Okay	7
Poor	4
Total	**40**

More data:

Good
Superior

2023

```
=LET(
d,B7:B46,
su,SORT(UNIQUE(d)),
c,COUNTIFS(d,su),
rh,{"Answers","Count"},
VSTACK(rh,HSTACK(su,c),HSTACK("Total",COUNTA(d))))
```

2024

```
=PIVOTBY(S[Answers],,S[Answers],COUNTA,,,-2)
```

2024

```
=GROUPBY(S[Answers],S[Answers],COUNTA,,,-2)
```

PIVOTBY Array Function to Create Reports

PIVOTBY Array Function:
Using a formula, it creates PivotTable-like summary reports with conditional calculations based on row area, column area and filter area conditions. This function allows you to group, aggregate, sort, and filter data based on the fields you specify.

**PIVOTBY(row_fields , col_fields , values , function ,
[field_headers] , [row_total_depth] , [row_sort_order] , [col_total_depth] , [col_sort_order] , [filter_array] , [relative_to])**

Argument	Description
row_fields (required)	A column-oriented array or range that contains the values which are used to group rows and generate row headers. The array or range may contain multiple columns. If so, the output will have multiple row group levels.
col_fields (required)	A column-oriented array or range that contains the values which are used to group columns and generate column headers. The array or range may contain multiple columns. If so, the output will have multiple column group levels.
values (required)	A column-oriented array or range of the data to aggregate. The array or range may contain multiple columns. If so, the output will have multiple aggregations.
function (required)	An explicit or eta reduced lambda (SUM, PERCENTOF, AVERAGE, COUNT, etc) that is used to aggregate values. A vector of lambdas can be provided (with HSTACK or VSTACK). If so, the output will have multiple aggregations. The orientation of the vector will determine whether they are laid out row- or column-wise.
field_headers	A number that specifies whether the **row_fields**, **col_fields**, and **values** have headers and whether field headers should be returned in the results. The possible values are: **Missing**: Automatic. **0**: No **1**: Yes and don't show **2**: No but generate **3**: Yes and show **Note**: Automatic assumes the data contains headers based on the values argument. If the 1st value is text and the 2nd value is a number, then the data is assumed to have headers. Fields headers are shown if there are multiple row or column group levels.
row_total_depth	Determines whether the row headers should contain totals. The possible values are: **Missing**: Automatic: Grand totals and, where possible, subtotals. **0**: No Totals **1**: Grand Totals **2**: Grand and Subtotals **-1**: Grand Totals at Top **-2**: Grand and Subtotals at Top **Note**: For subtotals, **row_fields** must have at least 2 columns. Numbers greater than 2 are supported provided **row_fields** has sufficient columns.
row_sort_order	A number indicating how rows should be sorted. Numbers correspond with columns in **row_fields** followed by the columns in **values**. If the number is negative, the rows are sorted in descending/reverse order. A vector of numbers can be provided when sorting based on only **row_fields**.
col_total_depth	Determines whether the row headers should contain totals. The possible values are: **Missing**: Automatic: Grand totals and, where possible, subtotals. **0**: No Totals **1**: Grand Totals **2**: Grand and Subtotals **-1**: Grand Totals at Top **-2**: Grand and Subtotals at Top **Note**: For subtotals, **col_fields** must have at least 2 columns. Numbers greater than 2 are supported provided **col_fields** has sufficient columns.
col_sort_order	A number indicating how rows should be sorted. Numbers correspond with columns in **col_fields** followed by the columns in **values**. If the number is negative, the rows are sorted in descending/reverse order. A vector of numbers can be provided when sorting based on only **col_fields**.

GROUPBY Array Function to Create Reports

The GROUPBY Array Function

Using a formula, the GROUPBY function creates PivotTable-like summary reports with conditional calculations based on row area and filter area conditions. This function allows you to group, aggregate, sort, and filter data based on the fields you specify

GROUPBY(row_fields , values , function ,
[field_headers] , [total_depth] , [sort_order] , [filter_array] , [field_relationship])

Argument	Description
row_fields (required)	A column-oriented array or range that contains the values which are used to group rows and generate row headers. The array or range may contain multiple columns. If so, the output will have multiple row group levels.
values (required)	A column-oriented array or range of the data to aggregate. The array or range may contain multiple columns. If so, the output will have multiple aggregations.
function (required)	An explicit or eta reduced lambda (SUM, PERCENTOF, AVERAGE, COUNT, etc.) that is used to aggregate values. A vector of lambdas can be provided (with HSTACK or VSTACK). If so, the output will have multiple aggregations. The orientation of the vector will determine whether they are laid out row- or column-wise.
field_headers	A number that specifies whether the **row_fields** and **values** have headers and whether field headers should be returned in the results. The possible values are: **Missing**: Automatic. **0**: No **1**: Yes and don't show **2**: No but generate **3**: Yes and show **Note**: Automatic assumes the data contains headers based on the values argument. If the 1st value is text and the 2nd value is a number, then the data is assumed to have headers. Fields headers are shown if there are multiple row or column group levels.
total_depth	Determines whether the row headers should contain totals. The possible values are: **Missing**: Automatic: Grand totals and, where possible, subtotals. **0**: No Totals **1**: Grand Totals **2**: Grand and Subtotals **-1**: Grand Totals at Top **-2**: Grand and Subtotals at Top **Note**: For subtotals, fields must have at least 2 columns. Numbers greater than 2 are supported provided field has sufficient columns.
sort_order	A number indicating how rows should be sorted. Numbers correspond with columns in **row_fields** followed by the columns in **values**. If the number is negative, the rows are sorted in descending/reverse order. A vector of numbers can be provided when sorting based on only **row_fields**.
filter_array	A column-oriented 1D array of Booleans that indicate whether the corresponding row of data should be considered. **Note**: The length of the array must match the length of those provided to **row_fields**.
field_relationship	Specifies the relationship fields when multiple columns are provided to **row_fields**. The possible values are: **0**: Hierarchy (default) **1**: Table With a Hierarchy field relationship (0), sorting of later field columns takes into account the hierarchy of earlier columns. This means if you sort the second column, it is sorted within the first column, a different sort for each group from the first column. With a Table field relationship (1), sorting of each field column is done independently. If you sort a column, the records remain intact, just as the would in a table. Subtotals are not supported as they rely on the data having a hierarchy.

When to use PivotTable & When to Use PIVOTBY & GROUPBY

The main advantages of the PivotTable feature are:

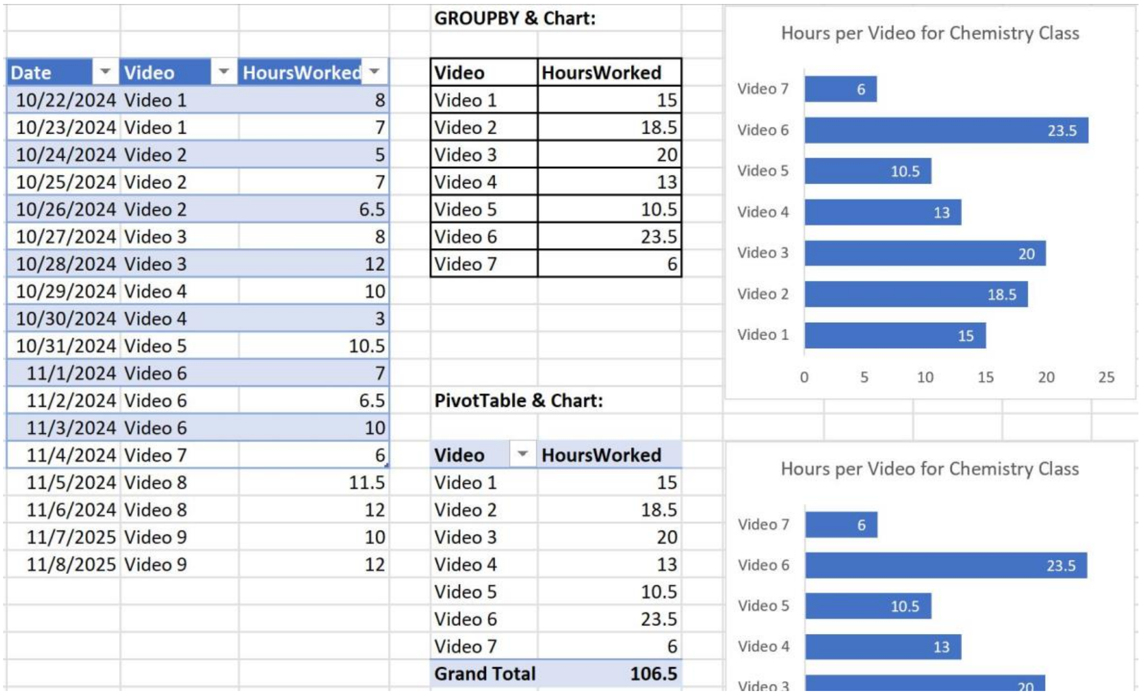
- The most common reporting calculations are adding, counting and creating percentages. The PivotTable allows you to create these calculations quickly with the features: Summarize Values By (11 aggregate functions) and Show Values As (14 additional calculations).
- PivotTables have automatic formatting that follows the report as you pivot it.
- The PivotTable Grouping Feature is unparalleled in how easy it is to group dates into months, standard quarters and years, times into hours and numbers into grouped categories with an upper and lower limit.
- Pivoting a report is easier than with formulas.

The main advantages of the GROUPBY & PIVOTBY functions are:

- The GROUPBY & PIVOTBY functions update instantly when source data changes.
- In the GROUPBY & PIVOTBY *function* argument, there are more aggregate functions available than in the PivotTable, such as: MEDIAN, ARRAYTOTEXT, CONCAT, SINGLE, SQRT, SUMSQ, and many more.
- You can avoid adding helper columns to source data tables by adding the helper column formula directly to the *values* argument in the GROUPBY & PIVOTBY functions (similar to DAX Functions in Power Pivot & Power BI like SUMX and AVERAGEX). For example, you can add a price lookup formula, a hourly time calculation formula, a price increase formula directly to the *values* argument.
- In the GROUPBY & PIVOTBY *function* argument, you can use the LAMBDA function to define your own custom function. The possibilities are limitless. This ability is not taught in this video. It will be taught in a different video.
- Anecdotal benefits:
  - You can use the report result from the GROUPBY & PIVOTBY functions as the source data for an X-Y Scatter chart, whereas you cannot create an X-Y Scatter chart from a PivotTable.
  - You can use the 1 – Table option in the field_relationship argument of the GROUPBY function to ignore the hierarchical relationship between two columns and sort a report independent of the hierarchical relationship (sort records like you would in a table).

Example of Instant Update in the GROUPBY Function

GROUPBY function and PivotTable reports BEFORE new data is added to Excel Table:



Date	Video	HoursWorked
10/22/2024	Video 1	8
10/23/2024	Video 1	7
10/24/2024	Video 2	5
10/25/2024	Video 2	7
10/26/2024	Video 2	6.5
10/27/2024	Video 3	8
10/28/2024	Video 3	12
10/29/2024	Video 4	10
10/30/2024	Video 4	3
10/31/2024	Video 5	10.5
11/1/2024	Video 6	7
11/2/2024	Video 6	6.5
11/3/2024	Video 6	10
11/4/2024	Video 7	6
11/5/2024	Video 8	11.5
11/6/2024	Video 8	12
11/7/2025	Video 9	10
11/8/2025	Video 9	12

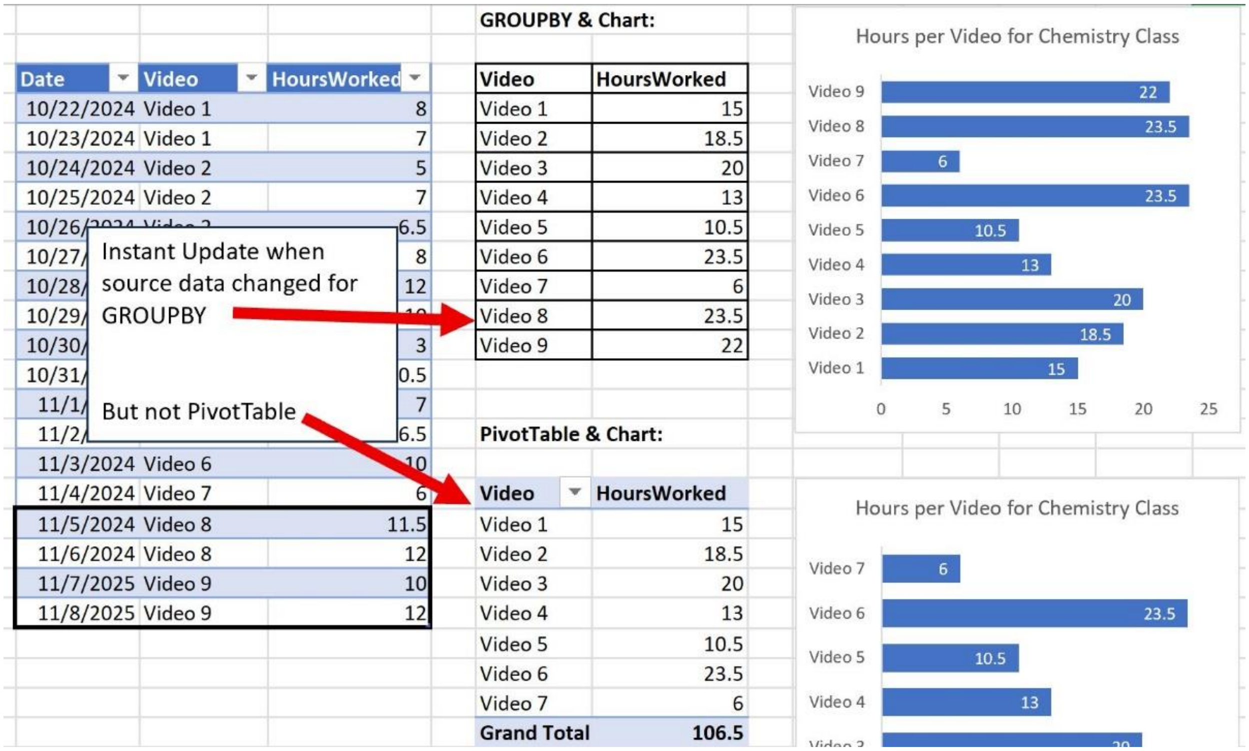
GROUPBY & Chart:

Video	HoursWorked
Video 1	15
Video 2	18.5
Video 3	20
Video 4	13
Video 5	10.5
Video 6	23.5
Video 7	6

PivotTable & Chart:

Video	HoursWorked
Video 1	15
Video 2	18.5
Video 3	20
Video 4	13
Video 5	10.5
Video 6	23.5
Video 7	6
Grand Total	**106.5**

GROUPBY function and PivotTable reports AFTER new data is added to Excel Table:



Date	Video	HoursWorked
10/22/2024	Video 1	8
10/23/2024	Video 1	7
10/24/2024	Video 2	5
10/25/2024	Video 2	7
10/26/2024	Video 3	6.5
10/27/		8
10/28/		12
10/29/		10
10/30/		3
10/31/		0.5
11/1/		7
11/2/		6.5
11/3/2024	Video 6	10
11/4/2024	Video 7	6
11/5/2024	Video 8	11.5
11/6/2024	Video 8	12
11/7/2025	Video 9	10
11/8/2025	Video 9	12

Instant Update when source data changed for GROUPBY

But not PivotTable

GROUPBY & Chart:

Video	HoursWorked
Video 1	15
Video 2	18.5
Video 3	20
Video 4	13
Video 5	10.5
Video 6	23.5
Video 7	6
Video 8	23.5
Video 9	22

PivotTable & Chart:

Video	HoursWorked
Video 1	15
Video 2	18.5
Video 3	20
Video 4	13
Video 5	10.5
Video 6	23.5
Video 7	6
Grand Total	106.5

Array Functions to Building Single Cell Array Formulas: A Revolution in Excel

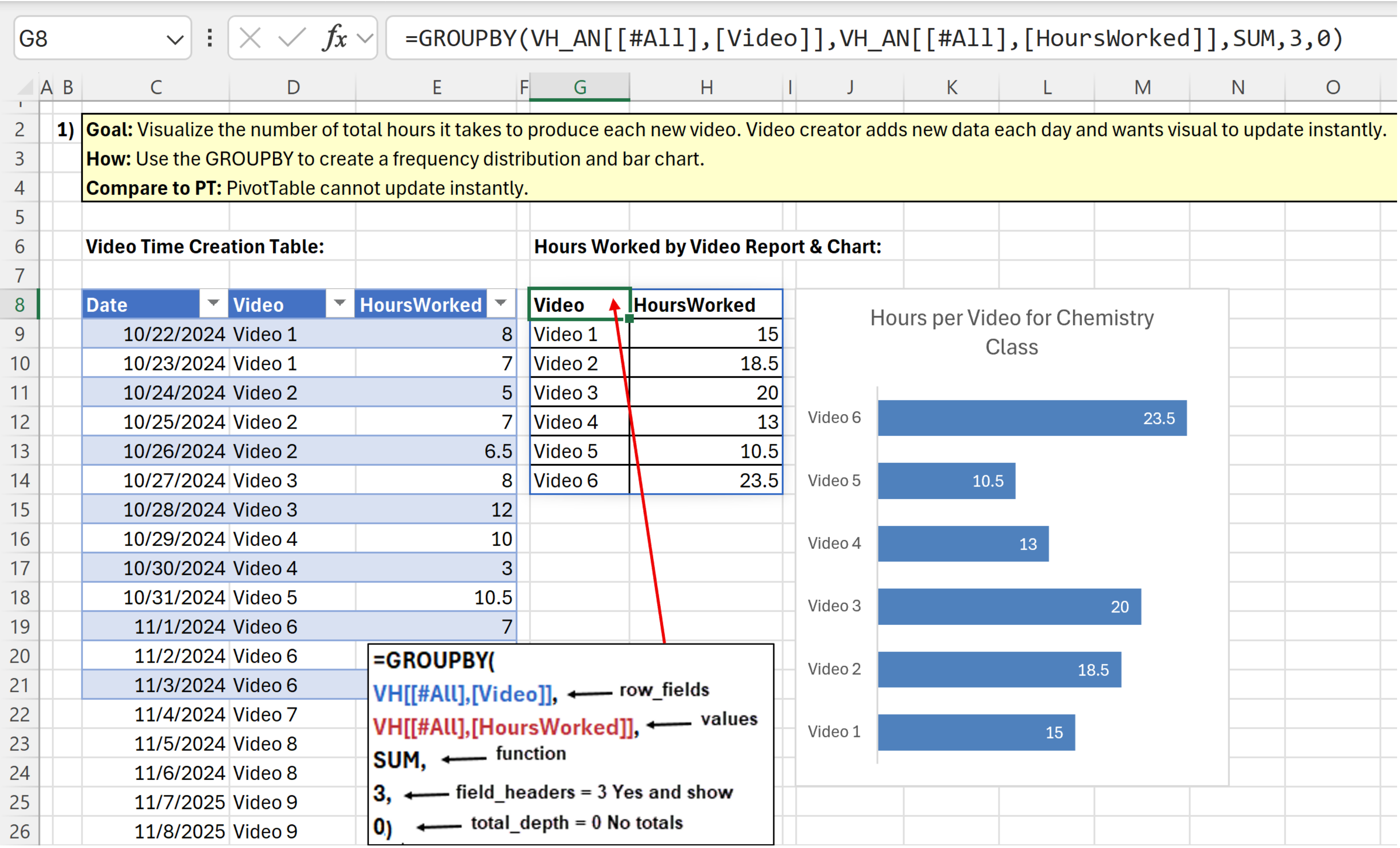
The GROUOPBY, PIVOTBY, VSTACK, HSTACK and other functions have revolutionized how reporting can be done with single cell formulas. The nine examples we will see in this video are as follows:

1) Use GROUPBY to create a frequency distribution and chart to show a video creator's total time by video that will update instantly when daily data is added to an Excel Table.
2) Use the ARRAYTOTEXT function in the *function* argument of the GROUPBY function to create a unique list of videos with a list of hours worked next to each video title.
3) Use GROUPBY to find the median price for the Baltic birch product for each supplier.
4) Use the GROUPBY and in the *values* argument, avoid using a helper column in the source data table and instead create a sales transaction amount column using the XLOOKUP function.
5) In the *function* argument of the GROUPBY function, use the HSTACK function to horizontally stack the two functions: SUM and PERCENTOF.
6) Use GROUPBY to create the time to hour conversion formula in the *values* argument and avoid a helper column in the source data table.
7) Use the PIVOTBY function and the *related_to* argument with the Parent Row Total option to convert the PERCENTOF function default, % of column total calculation, to a % of parent row calculation.
8) In the *function* argument of PIVOTBY, use the PERCENTOF function to create a three-in-one report that can either show % of grand total, % of column total or % or row total. In the *related_to* argument of the PIVOTBY function, use a formula input that allows you to switch between the different reports.
9) Because we cannot summarize data in a PivotTable and use it as the source data for an X-Y Scatter chart, in this example, you can use GROUPBY function to summarize monthly advertising dollars spent (X) and monthly sales (Y) and then use it as the source data to create an X-Y Scatter chart to see if there is a relationship between the amount spent on advertising and sales.
10) We will use a source data table with 990,000 rows of data and we will see that GROUPBY & PIVOTBY can handle this amount of data quickly, and it will not have to be added to a PivotTable cache, like if you use a PivotTable.

Example 1: GROUPBY Function to Create a Frequency Distribution

In this example you will use GROUPBY to create a frequency distribution and chart to show a video creator's total time by video that will update instantly when daily data is added to an Excel Table. The picture below shows the result:

Single cell report in cell G4 and Excel Bar Chart before new data is added:



G8 | =GROUPBY(VH_AN[[#All],[Video]],VH_AN[[#All],[HoursWorked]],SUM,3,0)

1) **Goal:** Visualize the number of total hours it takes to produce each new video. Video creator adds new data each day and wants visual to update instantly.
How: Use the GROUPBY to create a frequency distribution and bar chart.
Compare to PT: PivotTable cannot update instantly.

Video Time Creation Table:

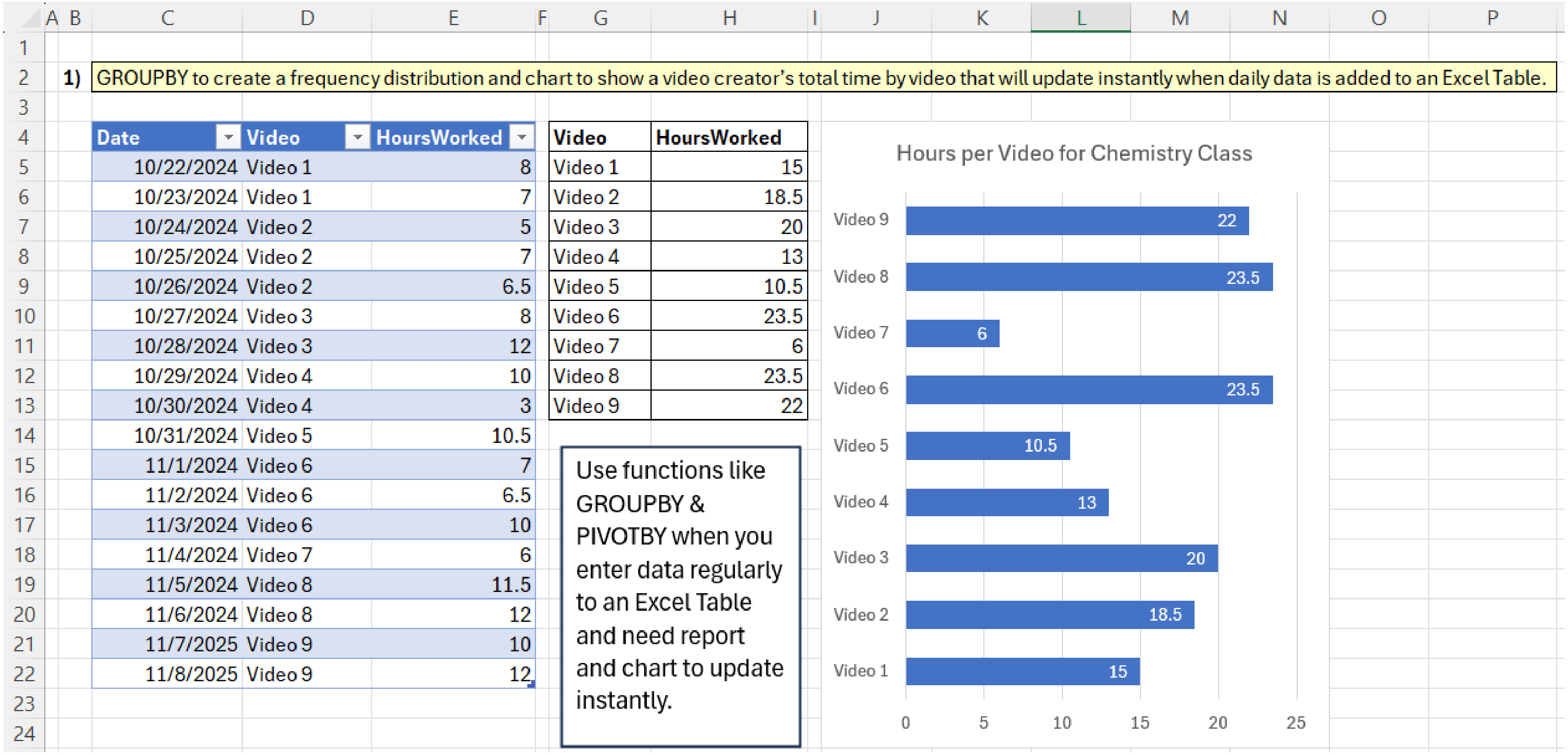
Date	Video	HoursWorked
10/22/2024	Video 1	8
10/23/2024	Video 1	7
10/24/2024	Video 2	5
10/25/2024	Video 2	7
10/26/2024	Video 2	6.5
10/27/2024	Video 3	8
10/28/2024	Video 3	12
10/29/2024	Video 4	10
10/30/2024	Video 4	3
10/31/2024	Video 5	10.5
11/1/2024	Video 6	7
11/2/2024	Video 6	
11/3/2024	Video 6	
11/4/2024	Video 7	
11/5/2024	Video 8	
11/6/2024	Video 8	
11/7/2025	Video 9	
11/8/2025	Video 9	

Hours Worked by Video Report & Chart:

Video	HoursWorked
Video 1	15
Video 2	18.5
Video 3	20
Video 4	13
Video 5	10.5
Video 6	23.5

=GROUPBY(
VH[[#All],[Video]], ← row_fields
VH[[#All],[HoursWorked]], ← values
SUM, ← function
3, ← field_headers = 3 Yes and show
0) ← total_depth = 0 No totals

Single cell report Excel Bar Chart AFTER new data is added:



1) GROUPBY to create a frequency distribution and chart to show a video creator's total time by video that will update instantly when daily data is added to an Excel Table.

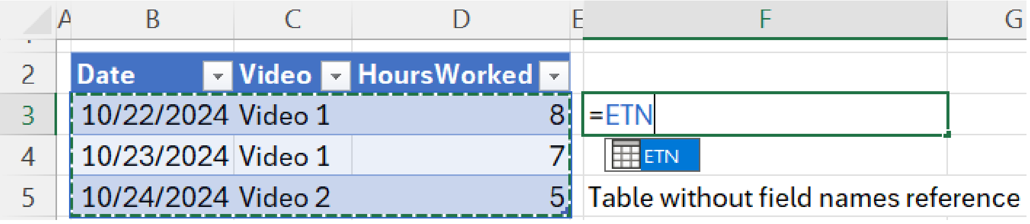
Date	Video	HoursWorked
10/22/2024	Video 1	8
10/23/2024	Video 1	7
10/24/2024	Video 2	5
10/25/2024	Video 2	7
10/26/2024	Video 2	6.5
10/27/2024	Video 3	8
10/28/2024	Video 3	12
10/29/2024	Video 4	10
10/30/2024	Video 4	3
10/31/2024	Video 5	10.5
11/1/2024	Video 6	7
11/2/2024	Video 6	6.5
11/3/2024	Video 6	10
11/4/2024	Video 7	6
11/5/2024	Video 8	11.5
11/6/2024	Video 8	12
11/7/2025	Video 9	10
11/8/2025	Video 9	12

Video	HoursWorked
Video 1	15
Video 2	18.5
Video 3	20
Video 4	13
Video 5	10.5
Video 6	23.5
Video 7	6
Video 8	23.5
Video 9	22

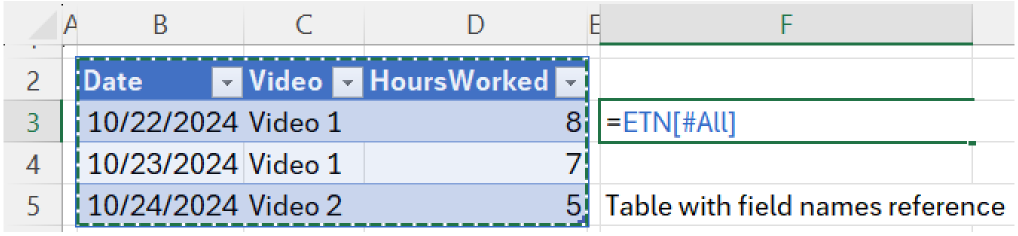
Use functions like GROUPBY & PIVOTBY when you enter data regularly to an Excel Table and need report and chart to update instantly.

Note about Excel Table Formula Nomenclature (Excel Table Formulas)

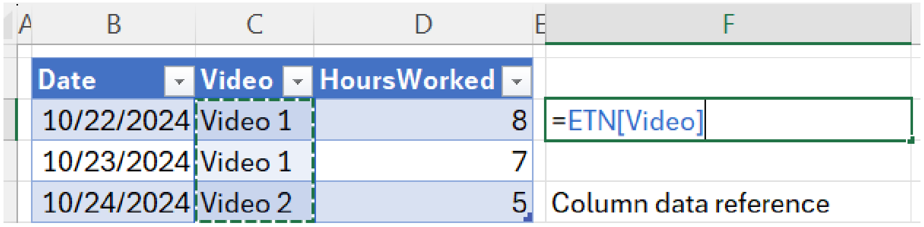
Below are pictures of the most important types of references that we can make to an Excel Table.



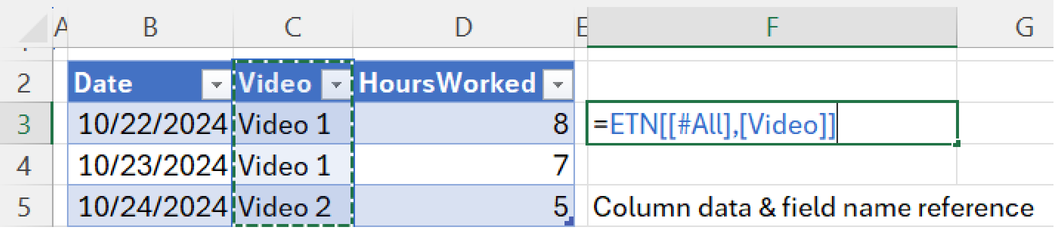
	A	B	C	D	E	F	G
2		Date	Video	HoursWorked			
3		10/22/2024	Video 1	8		=ETN	
4		10/23/2024	Video 1	7		ETN	
5		10/24/2024	Video 2	5		Table without field names reference	



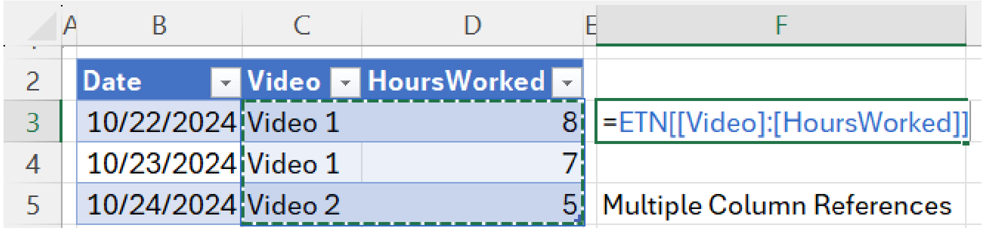
	A	B	C	D	E	F
2		Date	Video	HoursWorked		
3		10/22/2024	Video 1	8		=ETN[#All]
4		10/23/2024	Video 1	7		
5		10/24/2024	Video 2	5		Table with field names reference



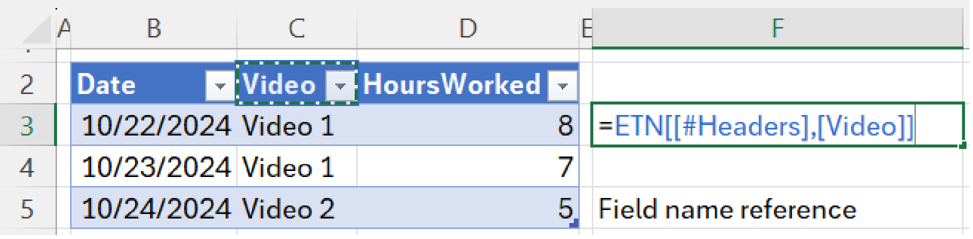
A	B	C	D	E	F
	Date	Video	HoursWorked		
	10/22/2024	Video 1	8		=ETN[Video]
	10/23/2024	Video 1	7		
	10/24/2024	Video 2	5		Column data reference



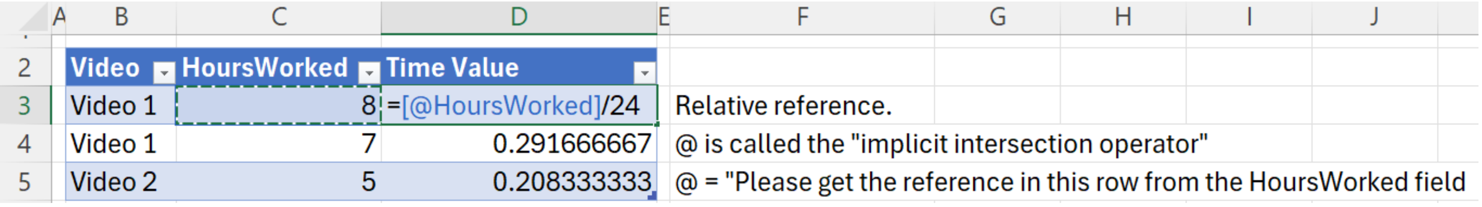
	A	B	C	D	E	F	G
2		Date	Video	HoursWorked			
3		10/22/2024	Video 1	8		=ETN[[#All],[Video]]	
4		10/23/2024	Video 1	7			
5		10/24/2024	Video 2	5		Column data & field name reference	



	A	B	C	D	E	F
2		Date	Video	HoursWorked		
3		10/22/2024	Video 1	8		=ETN[[Video]:[HoursWorked]]
4		10/23/2024	Video 1	7		
5		10/24/2024	Video 2	5		Multiple Column References



	A	B	C	D	E	F
2		Date	Video	HoursWorked		
3		10/22/2024	Video 1	8		=ETN[[#Headers],[Video]]
4		10/23/2024	Video 1	7		
5		10/24/2024	Video 2	5		Field name reference



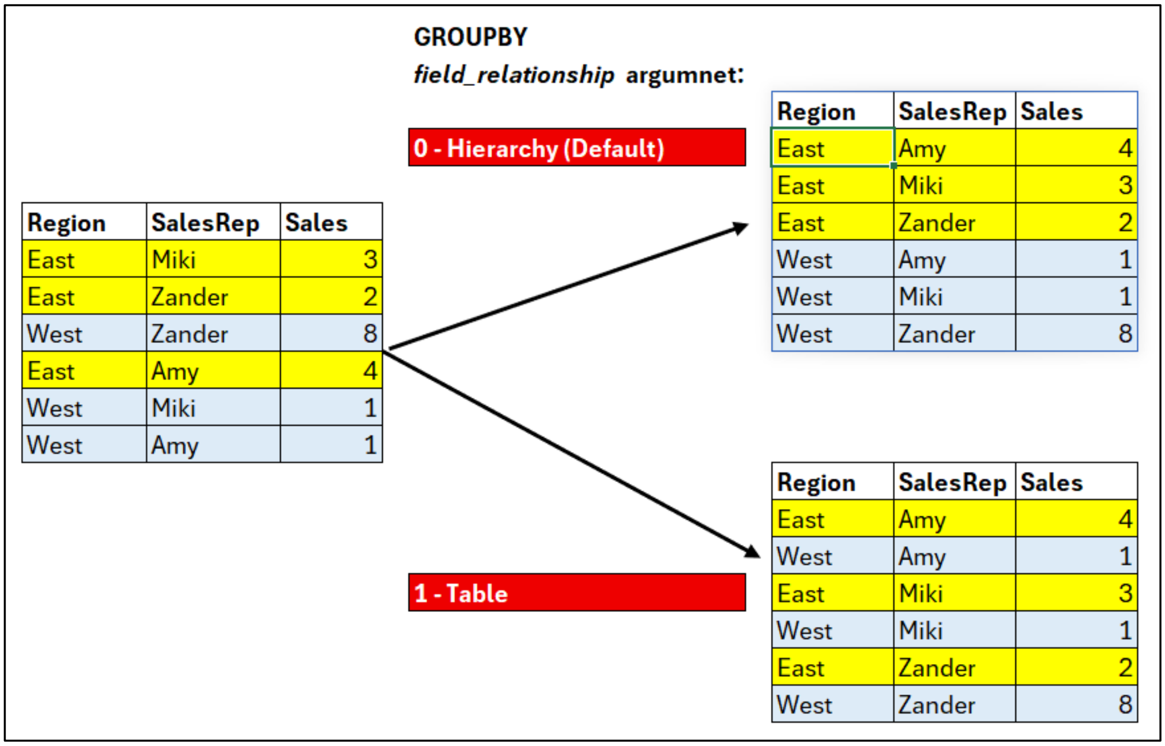
	A	B	C	D	E	F
2		Video	HoursWorked	Time Value		
3		Video 1	8	=[@HoursWorked]/24		Relative reference.
4		Video 1	7	0.291666667		@ is called the "implicit intersection operator"
5		Video 2	5	0.208333333		@ = "Please get the reference in this row from the HoursWorked field

Eta-LAMBDA Functions

Eta LAMBDA functions are built-in aggregate functions that can be used in the function argument of the GROUPBY and PIVOTBY functions (Other Dynamic Spilled Array Functions with this function argument, such as BYROW, BYCOL, SCAN and others (not seen in this class) can use Eta-LAMBDAs also. In the PIVOTBY and GROUPBY functions, Eta LAMBDAs can make an aggregate calculations for each condition in the row or column sections of the generated report. A list of some of the possible Eta LAMBDA functions are shown here:

- ARRAYTOTEXT
- AVERAGE
- CONCAT
- COUNT
- COUNTA
- MAX
- MEDIAN
- MIN
- MODE.SNGL
- PERCENTOF
- PRODUCT
- SINGLE
- SQRT
- STDEV.P
- STDEV.S
- SUMSQ
- VAR.P
- VAR.S

field_relationship argument in GROUPBY



GROUPBY

***field_relationship* argumnet:**

Region	SalesRep	Sales
East	Miki	3
East	Zander	2
West	Zander	8
East	Amy	4
West	Miki	1
West	Amy	1

0 - Hierarchy (Default)

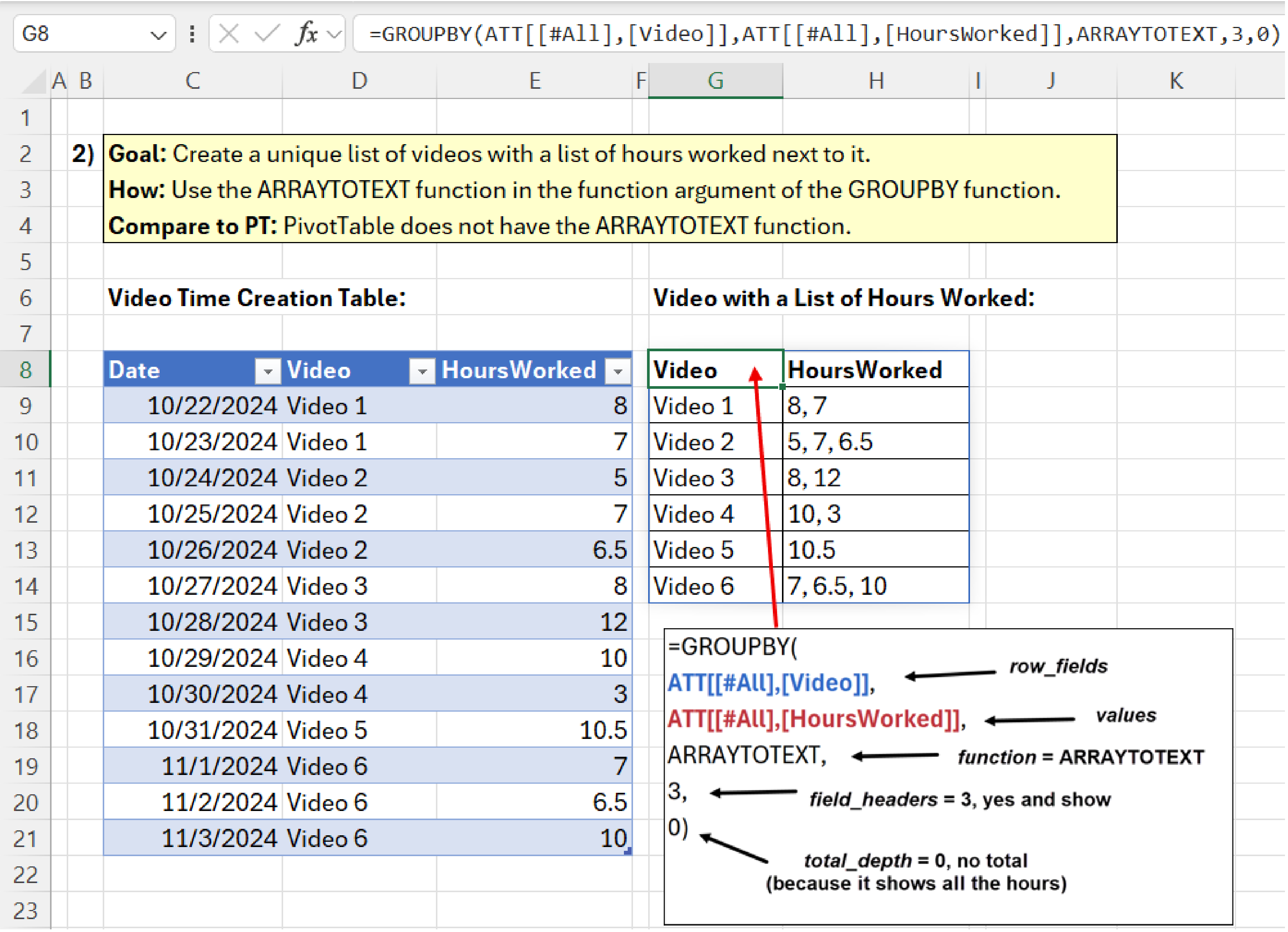
Region	SalesRep	Sales
East	Amy	4
East	Miki	3
East	Zander	2
West	Amy	1
West	Miki	1
West	Zander	8

1 - Table

Region	SalesRep	Sales
East	Amy	4
West	Amy	1
East	Miki	3
West	Miki	1
East	Zander	2
West	Zander	8

Example 2: GROUPBY Function & ARRAYTOTEXT Function

In this example, use the ARRAYTOTEXT function in the function argument of the GROUPBY function to create a unique list of videos with a list of hours worked next to each video title. The picture below shows the result:



```
=GROUPBY(ATT[[#All],[Video]],ATT[[#All],[HoursWorked]],ARRAYTOTEXT,3,0)
```

2) **Goal:** Create a unique list of videos with a list of hours worked next to it.
How: Use the ARRAYTOTEXT function in the function argument of the GROUPBY function.
Compare to PT: PivotTable does not have the ARRAYTOTEXT function.

Video Time Creation Table:

Date	Video	HoursWorked
10/22/2024	Video 1	8
10/23/2024	Video 1	7
10/24/2024	Video 2	5
10/25/2024	Video 2	7
10/26/2024	Video 2	6.5
10/27/2024	Video 3	8
10/28/2024	Video 3	12
10/29/2024	Video 4	10
10/30/2024	Video 4	3
10/31/2024	Video 5	10.5
11/1/2024	Video 6	7
11/2/2024	Video 6	6.5
11/3/2024	Video 6	10

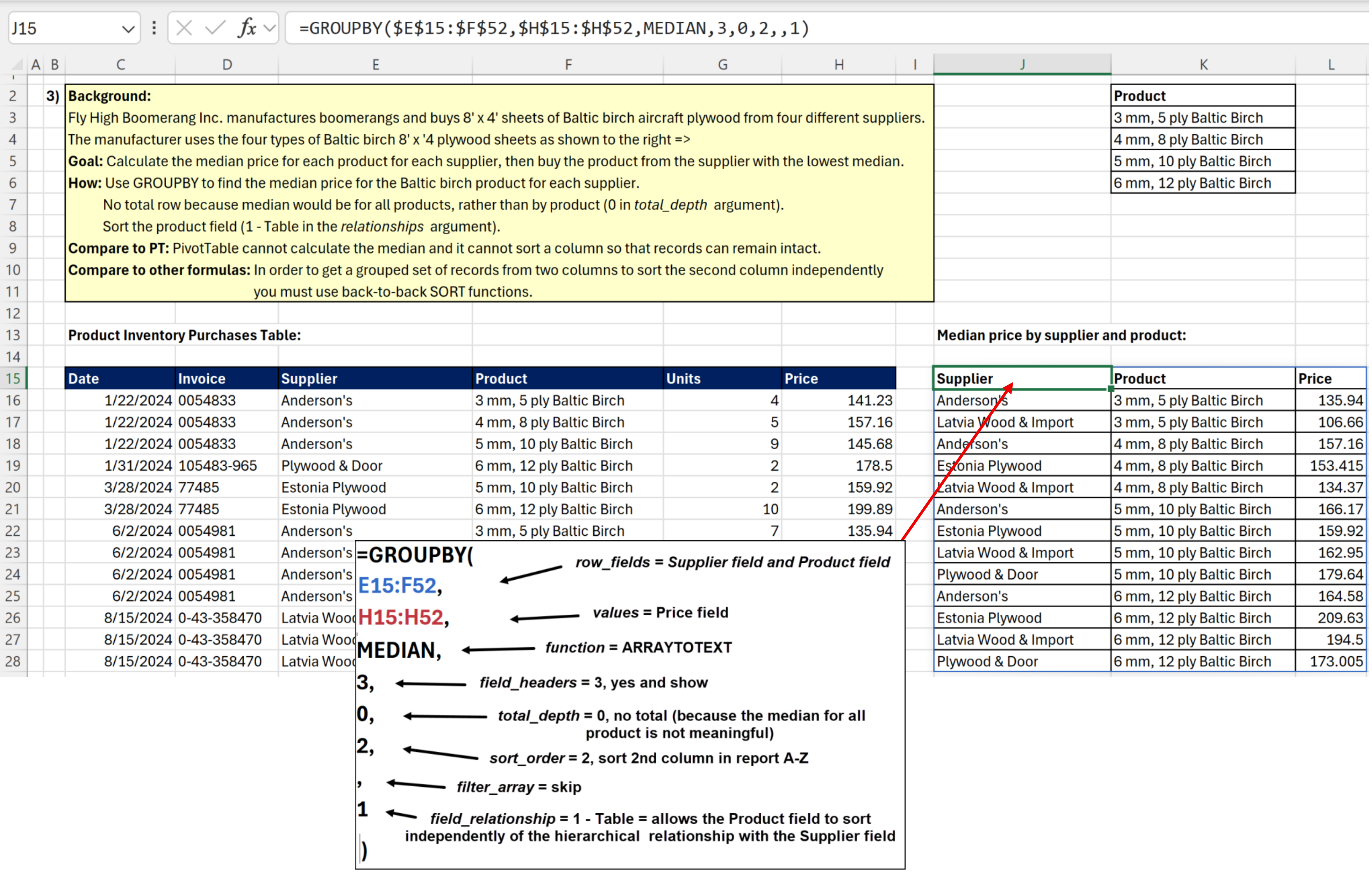
Video with a List of Hours Worked:

Video	HoursWorked
Video 1	8, 7
Video 2	5, 7, 6.5
Video 3	8, 12
Video 4	10, 3
Video 5	10.5
Video 6	7, 6.5, 10

```
=GROUPBY(
ATT[[#All],[Video]],         ← row_fields
ATT[[#All],[HoursWorked]],   ← values
ARRAYTOTEXT,                 ← function = ARRAYTOTEXT
3,                           ← field_headers = 3, yes and show
0)                           ← total_depth = 0, no total
                               (because it shows all the hours)
```

Example 3: GROUPBY Function & MEDIAN Function

Use GROUPBY to find the median price for the Baltic birch product for each supplier. The picture below shows the result:



J15

```
=GROUPBY($E$15:$F$52,$H$15:$H$52,MEDIAN,3,0,2,,1)
```

3) **Background:**
Fly High Boomerang Inc. manufactures boomerangs and buys 8' x 4' sheets of Baltic birch aircraft plywood from four different suppliers.
The manufacturer uses the four types of Baltic birch 8' x '4 plywood sheets as shown to the right =>
Goal: Calculate the median price for each product for each supplier, then buy the product from the supplier with the lowest median.
How: Use GROUPBY to find the median price for the Baltic birch product for each supplier.
No total row because median would be for all products, rather than by product (0 in *total_depth* argument).
Sort the product field (1 - Table in the *relationships* argument).
Compare to PT: PivotTable cannot calculate the median and it cannot sort a column so that records can remain intact.
Compare to other formulas: In order to get a grouped set of records from two columns to sort the second column independently
you must use back-to-back SORT functions.

Product
3 mm, 5 ply Baltic Birch
4 mm, 8 ply Baltic Birch
5 mm, 10 ply Baltic Birch
6 mm, 12 ply Baltic Birch

Product Inventory Purchases Table:

Date	Invoice	Supplier	Product	Units	Price
1/22/2024	0054833	Anderson's	3 mm, 5 ply Baltic Birch	4	141.23
1/22/2024	0054833	Anderson's	4 mm, 8 ply Baltic Birch	5	157.16
1/22/2024	0054833	Anderson's	5 mm, 10 ply Baltic Birch	9	145.68
1/31/2024	105483-965	Plywood & Door	6 mm, 12 ply Baltic Birch	2	178.5
3/28/2024	77485	Estonia Plywood	5 mm, 10 ply Baltic Birch	2	159.92
3/28/2024	77485	Estonia Plywood	6 mm, 12 ply Baltic Birch	10	199.89
6/2/2024	0054981	Anderson's	3 mm, 5 ply Baltic Birch	7	135.94
6/2/2024	0054981	Anderson's			
6/2/2024	0054981	Anderson's			
6/2/2024	0054981	Anderson's			
8/15/2024	0-43-358470	Latvia Woo			
8/15/2024	0-43-358470	Latvia Woo			
8/15/2024	0-43-358470	Latvia Woo			

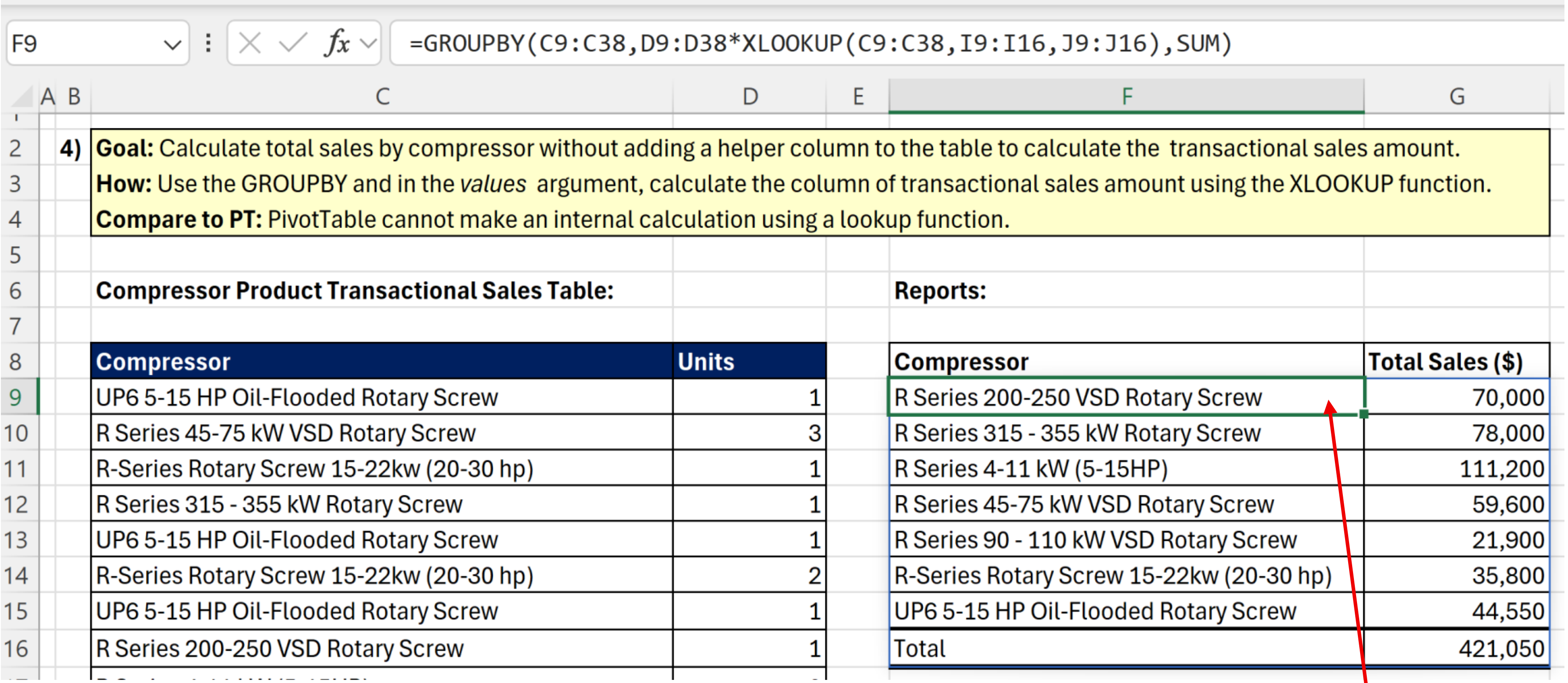
Median price by supplier and product:

Supplier	Product	Price
Anderson's	3 mm, 5 ply Baltic Birch	135.94
Latvia Wood & Import	3 mm, 5 ply Baltic Birch	106.66
Anderson's	4 mm, 8 ply Baltic Birch	157.16
Estonia Plywood	4 mm, 8 ply Baltic Birch	153.415
Latvia Wood & Import	4 mm, 8 ply Baltic Birch	134.37
Anderson's	5 mm, 10 ply Baltic Birch	166.17
Estonia Plywood	5 mm, 10 ply Baltic Birch	159.92
Latvia Wood & Import	5 mm, 10 ply Baltic Birch	162.95
Plywood & Door	5 mm, 10 ply Baltic Birch	179.64
Anderson's	6 mm, 12 ply Baltic Birch	164.58
Estonia Plywood	6 mm, 12 ply Baltic Birch	209.63
Latvia Wood & Import	6 mm, 12 ply Baltic Birch	194.5
Plywood & Door	6 mm, 12 ply Baltic Birch	173.005

=GROUPBY(
E15:F52, ← *row_fields* = Supplier field and Product field
H15:H52, ← *values* = Price field
MEDIAN, ← *function* = ARRAYTOTEXT
3, ← *field_headers* = 3, yes and show
0, ← *total_depth* = 0, no total (because the median for all product is not meaningful)
2, ← *sort_order* = 2, sort 2nd column in report A-Z
, ← *filter_array* = skip
1 ← *field_relationship* = 1 - Table = allows the Product field to sort independently of the hierarchical relationship with the Supplier field
)

Example 4: XLOOKUP Helper Column in *values* argument of GROUPBY

Use the GROUPBY and in the values argument, avoid using a helper column in the source data table and instead create a sales transaction amount column using the XLOOKUP function. The picture below shows the result:



=GROUPBY(C9:C38,D9:D38*XLOOKUP(C9:C38,I9:I16,J9:J16),SUM)

4) Goal: Calculate total sales by compressor without adding a helper column to the table to calculate the transactional sales amount.
How: Use the GROUPBY and in the *values* argument, calculate the column of transactional sales amount using the XLOOKUP function.
Compare to PT: PivotTable cannot make an internal calculation using a lookup function.

Compressor Product Transactional Sales Table:

Compressor	Units
UP6 5-15 HP Oil-Flooded Rotary Screw	1
R Series 45-75 kW VSD Rotary Screw	3
R-Series Rotary Screw 15-22kw (20-30 hp)	1
R Series 315 - 355 kW Rotary Screw	1
UP6 5-15 HP Oil-Flooded Rotary Screw	1
R-Series Rotary Screw 15-22kw (20-30 hp)	2
UP6 5-15 HP Oil-Flooded Rotary Screw	1
R Series 200-250 VSD Rotary Screw	1

Reports:

Compressor	Total Sales ($)
R Series 200-250 VSD Rotary Screw	70,000
R Series 315 - 355 kW Rotary Screw	78,000
R Series 4-11 kW (5-15HP)	111,200
R Series 45-75 kW VSD Rotary Screw	59,600
R Series 90 - 110 kW VSD Rotary Screw	21,900
R-Series Rotary Screw 15-22kw (20-30 hp)	35,800
UP6 5-15 HP Oil-Flooded Rotary Screw	44,550
Total	421,050

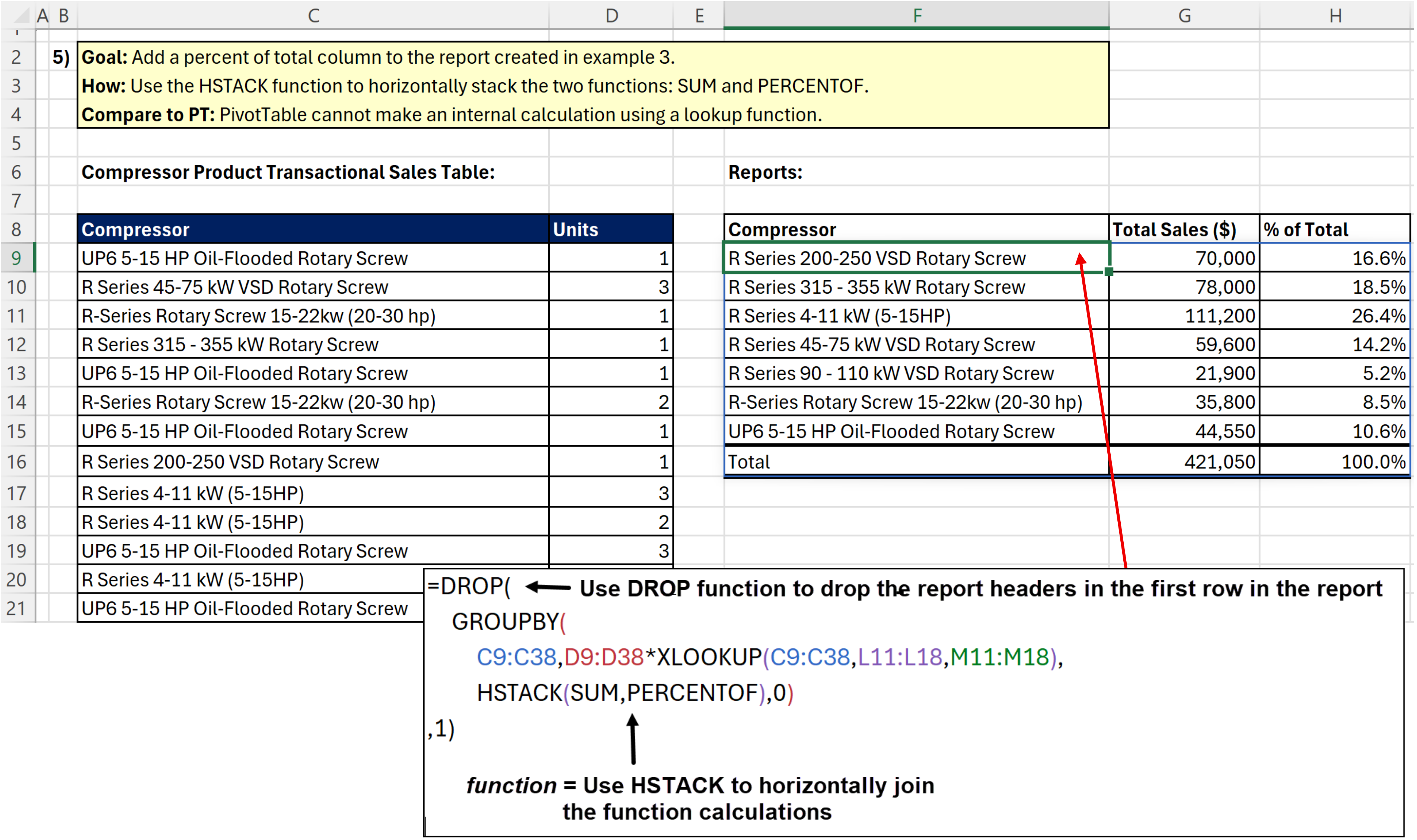
=GROUPBY(
C9:C38, ← ***row_fields* = Compressor field**
D9:D38*XLOOKUP(C9:C38,I9:I16,J9:J16), ← ***values* = XLOOKUP Helper Column**
SUM) ← ***function* = SUM**

Alternative to vertically stack the report headers and the report.

=VSTACK(
{"Compressor","Total Sales ($)"},
GROUPBY(C9:C38,D9:D38*XLOOKUP(C9:C38,I9:I16,J9:J16),SUM))

Example 5: Use HSTACK to create side-by-side calculations in a GROUPBY Report

In the function argument of the GROUPBY function, use the HSTACK function to horizontally stack the two functions: SUM and PERCENTOF. The picture below shows the result:



5) **Goal:** Add a percent of total column to the report created in example 3.
How: Use the HSTACK function to horizontally stack the two functions: SUM and PERCENTOF.
Compare to PT: PivotTable cannot make an internal calculation using a lookup function.

Compressor Product Transactional Sales Table:

Compressor	Units
UP6 5-15 HP Oil-Flooded Rotary Screw	1
R Series 45-75 kW VSD Rotary Screw	3
R-Series Rotary Screw 15-22kw (20-30 hp)	1
R Series 315 - 355 kW Rotary Screw	1
UP6 5-15 HP Oil-Flooded Rotary Screw	1
R-Series Rotary Screw 15-22kw (20-30 hp)	2
UP6 5-15 HP Oil-Flooded Rotary Screw	1
R Series 200-250 VSD Rotary Screw	1
R Series 4-11 kW (5-15HP)	3
R Series 4-11 kW (5-15HP)	2
UP6 5-15 HP Oil-Flooded Rotary Screw	3
R Series 4-11 kW (5-15HP)	
UP6 5-15 HP Oil-Flooded Rotary Screw	

Reports:

Compressor	Total Sales ($)	% of Total
R Series 200-250 VSD Rotary Screw	70,000	16.6%
R Series 315 - 355 kW Rotary Screw	78,000	18.5%
R Series 4-11 kW (5-15HP)	111,200	26.4%
R Series 45-75 kW VSD Rotary Screw	59,600	14.2%
R Series 90 - 110 kW VSD Rotary Screw	21,900	5.2%
R-Series Rotary Screw 15-22kw (20-30 hp)	35,800	8.5%
UP6 5-15 HP Oil-Flooded Rotary Screw	44,550	10.6%
Total	421,050	100.0%

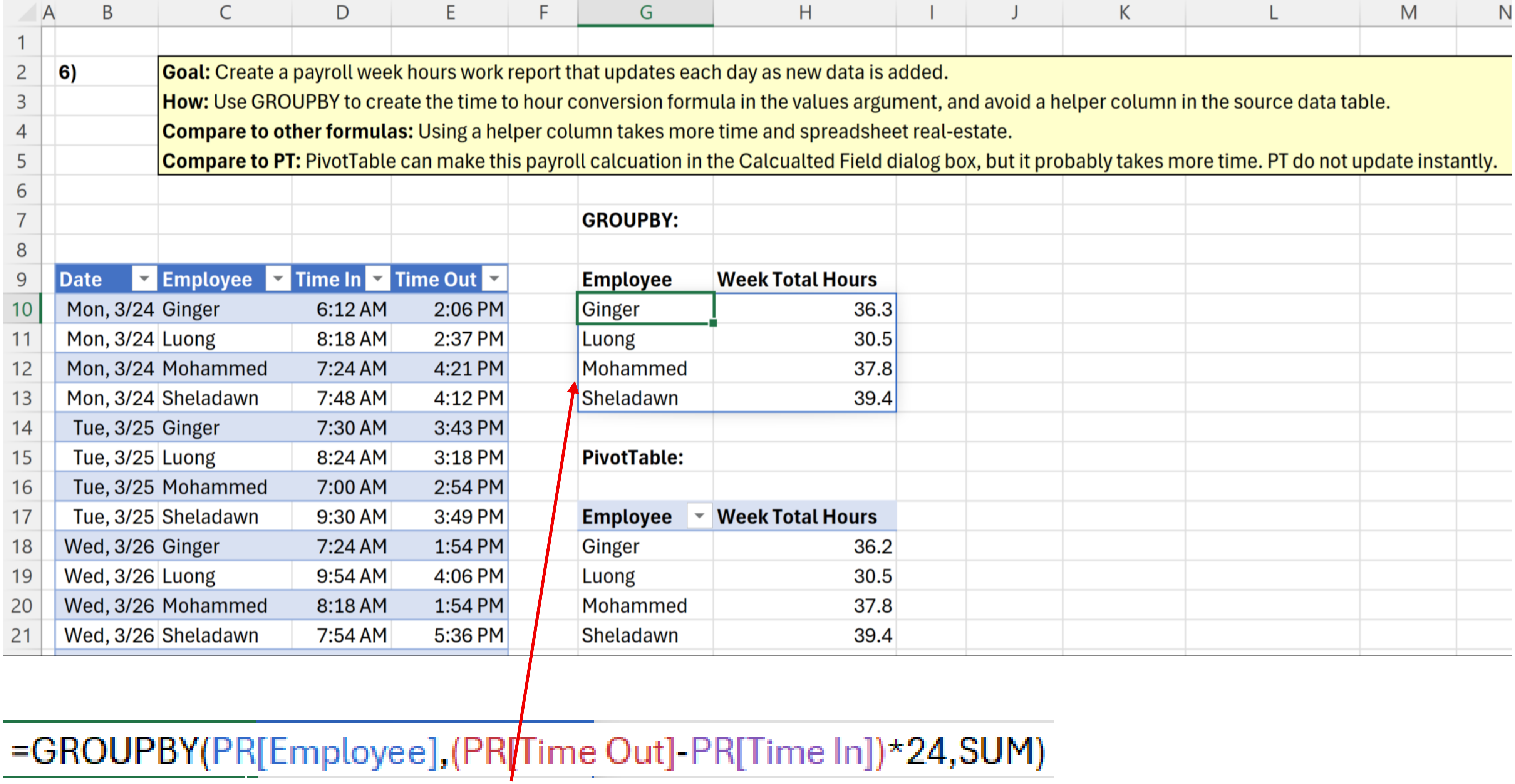
```
=DROP(
  GROUPBY(
    C9:C38,D9:D38*XLOOKUP(C9:C38,L11:L18,M11:M18),
    HSTACK(SUM,PERCENTOF),0)
,1)
```

Use DROP function to drop the report headers in the first row in the report

function **= Use HSTACK to horizontally join the function calculations**

Example 6: Payroll Helper Column Formula in values argument of GROUPBY

Use GROUPBY to create the time to hour conversion formula in the values argument and avoid a helper column in the source data table. The picture below shows the result:



6)

Goal: Create a payroll week hours work report that updates each day as new data is added.
How: Use GROUPBY to create the time to hour conversion formula in the values argument, and avoid a helper column in the source data table.
Compare to other formulas: Using a helper column takes more time and spreadsheet real-estate.
Compare to PT: PivotTable can make this payroll calcuation in the Calcualted Field dialog box, but it probably takes more time. PT do not update instantly.

Date	Employee	Time In	Time Out
Mon, 3/24	Ginger	6:12 AM	2:06 PM
Mon, 3/24	Luong	8:18 AM	2:37 PM
Mon, 3/24	Mohammed	7:24 AM	4:21 PM
Mon, 3/24	Sheladawn	7:48 AM	4:12 PM
Tue, 3/25	Ginger	7:30 AM	3:43 PM
Tue, 3/25	Luong	8:24 AM	3:18 PM
Tue, 3/25	Mohammed	7:00 AM	2:54 PM
Tue, 3/25	Sheladawn	9:30 AM	3:49 PM
Wed, 3/26	Ginger	7:24 AM	1:54 PM
Wed, 3/26	Luong	9:54 AM	4:06 PM
Wed, 3/26	Mohammed	8:18 AM	1:54 PM
Wed, 3/26	Sheladawn	7:54 AM	5:36 PM

GROUPBY:

Employee	Week Total Hours
Ginger	36.3
Luong	30.5
Mohammed	37.8
Sheladawn	39.4

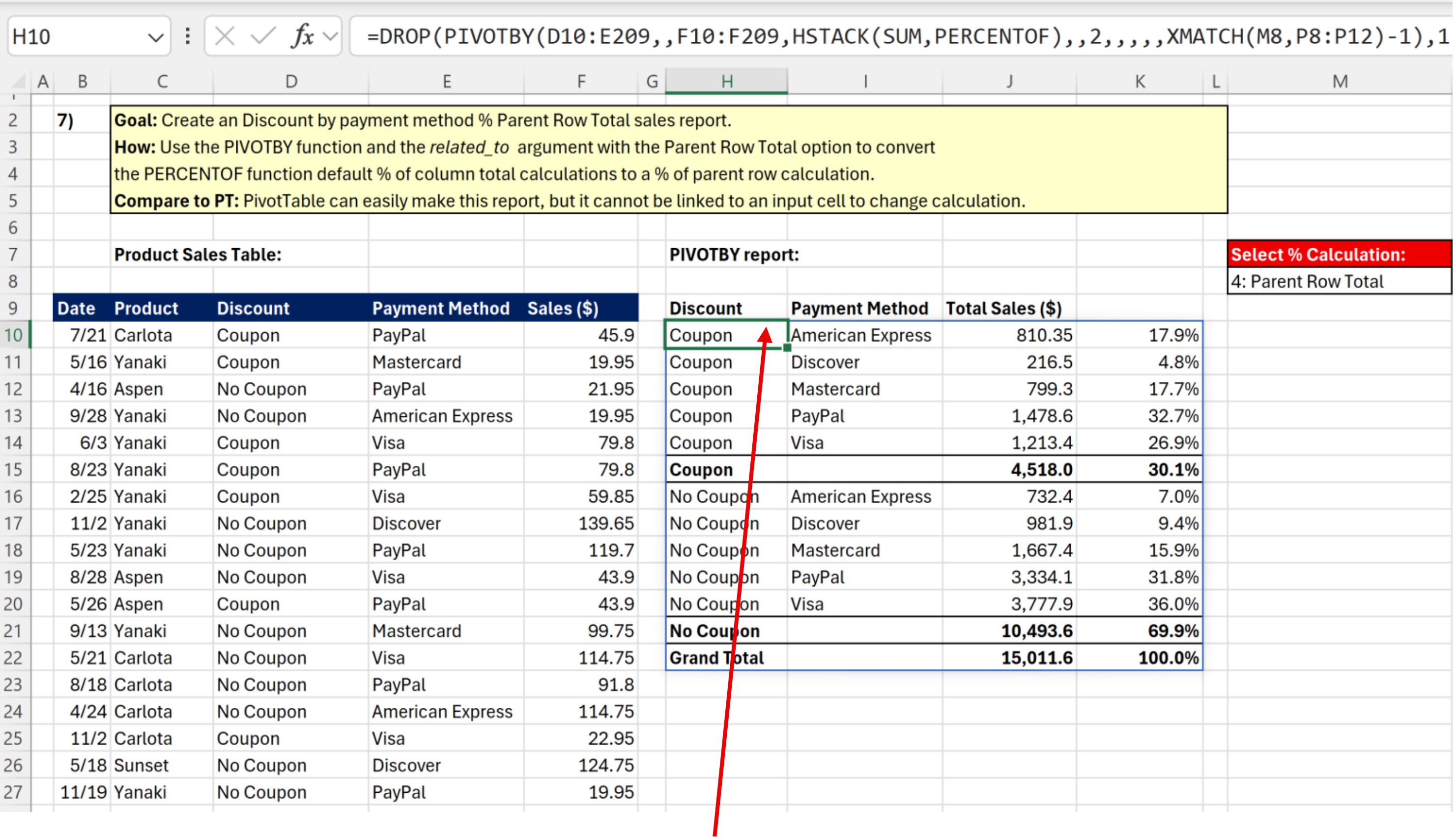
PivotTable:

Employee	Week Total Hours
Ginger	36.2
Luong	30.5
Mohammed	37.8
Sheladawn	39.4

```
=GROUPBY(PR[Employee],(PR[Time Out]-PR[Time In])*24,SUM)
```

Example 7: PIVOTBY function to create a % of Parent Row Total Report

Use the PIVOTBY function and the related_to argument with the Parent Row Total option to convert the PERCENTOF function default, % of column total calculation, to a % of parent row calculation. The picture below shows the result:



H10

```
=DROP(PIVOTBY(D10:E209,,F10:F209,HSTACK(SUM,PERCENTOF),,2,,,,,XMATCH(M8,P8:P12)-1),1
```

7)

Goal: Create an Discount by payment method % Parent Row Total sales report.
How: Use the PIVOTBY function and the *related_to* argument with the Parent Row Total option to convert the PERCENTOF function default % of column total calculations to a % of parent row calculation.
Compare to PT: PivotTable can easily make this report, but it cannot be linked to an input cell to change calculation.

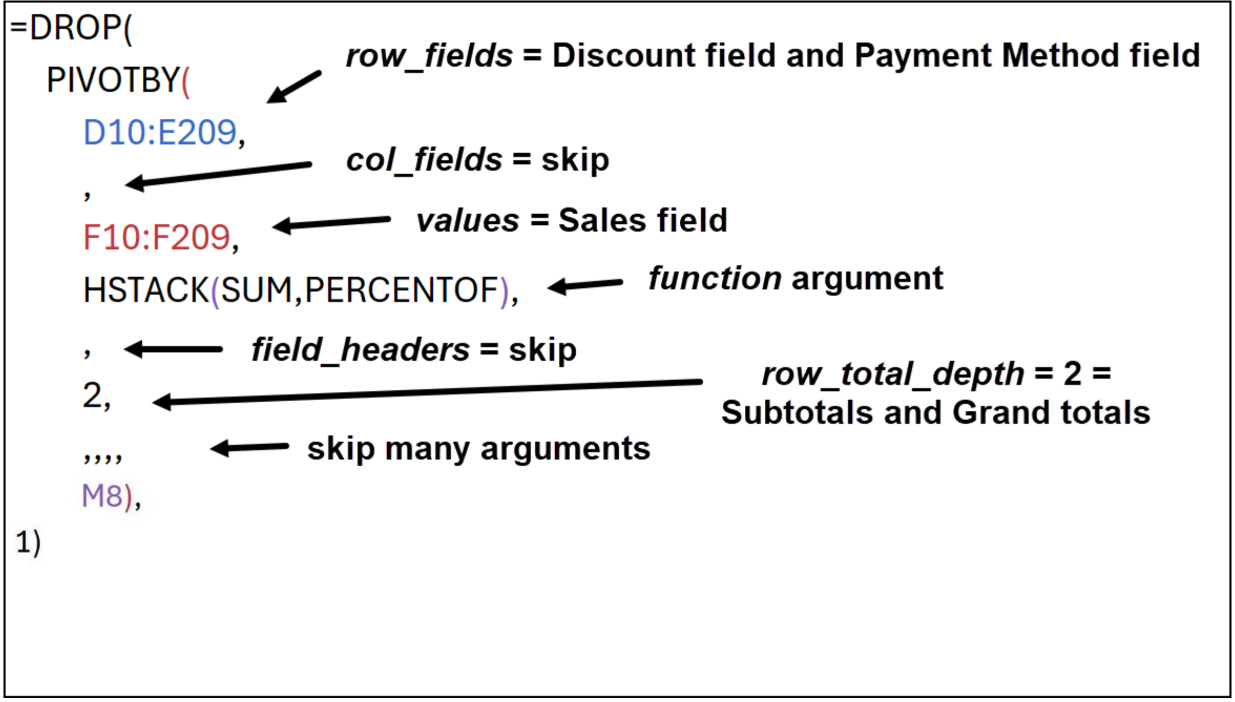
Product Sales Table:

Date	Product	Discount	Payment Method	Sales ($)
7/21	Carlota	Coupon	PayPal	45.9
5/16	Yanaki	Coupon	Mastercard	19.95
4/16	Aspen	No Coupon	PayPal	21.95
9/28	Yanaki	No Coupon	American Express	19.95
6/3	Yanaki	Coupon	Visa	79.8
8/23	Yanaki	Coupon	PayPal	79.8
2/25	Yanaki	Coupon	Visa	59.85
11/2	Yanaki	No Coupon	Discover	139.65
5/23	Yanaki	No Coupon	PayPal	119.7
8/28	Aspen	No Coupon	Visa	43.9
5/26	Aspen	Coupon	PayPal	43.9
9/13	Yanaki	No Coupon	Mastercard	99.75
5/21	Carlota	No Coupon	Visa	114.75
8/18	Carlota	No Coupon	PayPal	91.8
4/24	Carlota	No Coupon	American Express	114.75
11/2	Carlota	Coupon	Visa	22.95
5/18	Sunset	No Coupon	Discover	124.75
11/19	Yanaki	No Coupon	PayPal	19.95

PIVOTBY report:

Discount	Payment Method	Total Sales ($)	
Coupon	American Express	810.35	17.9%
Coupon	Discover	216.5	4.8%
Coupon	Mastercard	799.3	17.7%
Coupon	PayPal	1,478.6	32.7%
Coupon	Visa	1,213.4	26.9%
Coupon		**4,518.0**	**30.1%**
No Coupon	American Express	732.4	7.0%
No Coupon	Discover	981.9	9.4%
No Coupon	Mastercard	1,667.4	15.9%
No Coupon	PayPal	3,334.1	31.8%
No Coupon	Visa	3,777.9	36.0%
No Coupon		**10,493.6**	**69.9%**
Grand Total		**15,011.6**	**100.0%**

Select % Calculation:
4: Parent Row Total



```
=DROP(
  PIVOTBY(
    D10:E209,
    ,
    F10:F209,
    HSTACK(SUM,PERCENTOF),
    ,
    2,
    ,,,,
    M8),
1)
```

row_fields = Discount field and Payment Method field

col_fields = skip

values = Sales field

function argument

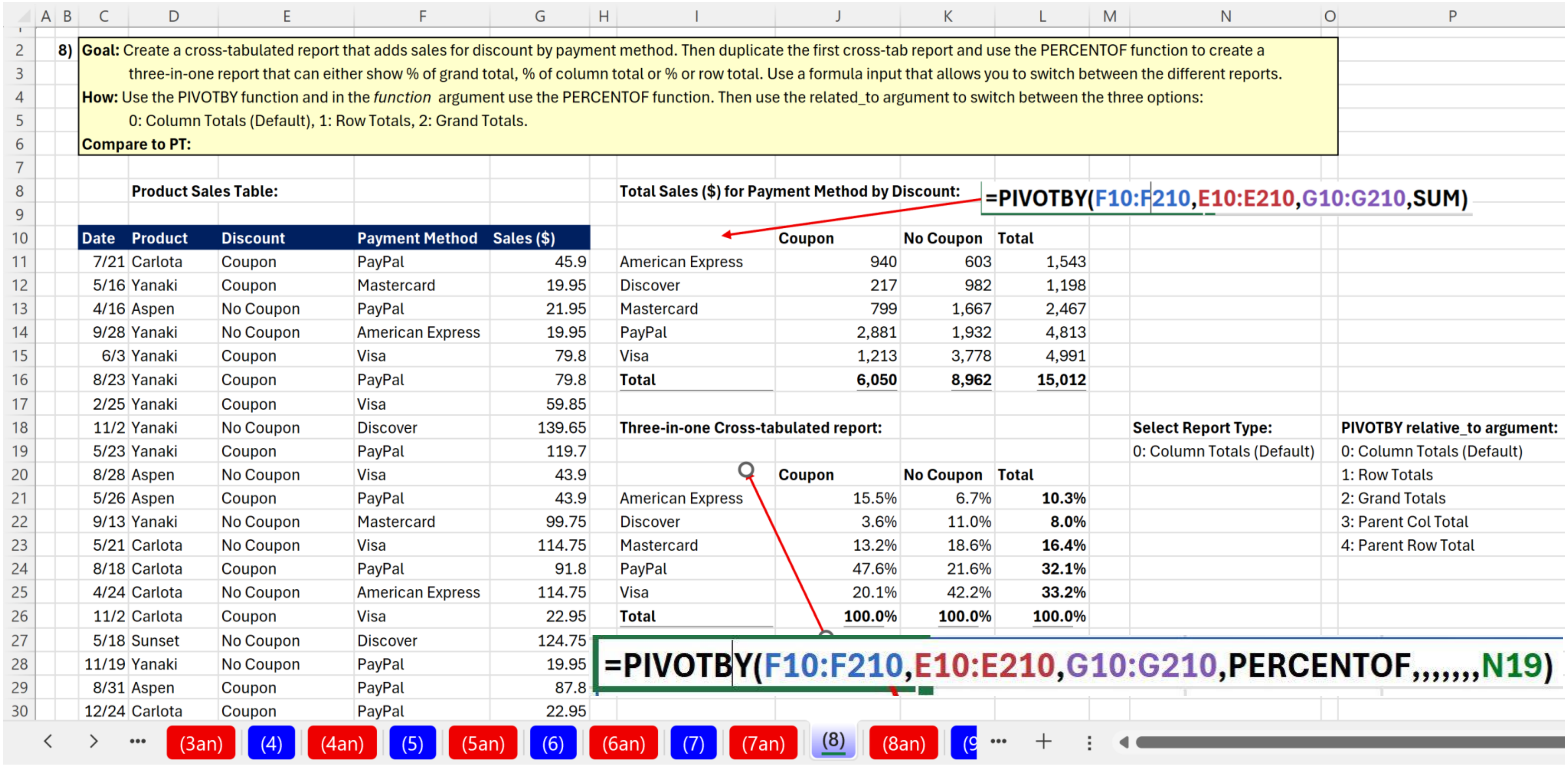
field_headers = skip

row_total_depth = 2 = Subtotals and Grand totals

skip many arguments

Example 8: PIVOTBY Function to Create a 3-in-1 Report with % Of Column Total, % Of Row Total & % Of Grand Total Reports

In the function argument of PIVOTBY, use the PERCENTOF function to create a three-in-one report that can either show % of grand total, % of column total or % or row total. In the related_to argument of the PIVOTBY function, use a formula input that allows you to switch between the different reports. The picture below shows the result:



8) **Goal:** Create a cross-tabulated report that adds sales for discount by payment method. Then duplicate the first cross-tab report and use the PERCENTOF function to create a three-in-one report that can either show % of grand total, % of column total or % or row total. Use a formula input that allows you to switch between the different reports.
How: Use the PIVOTBY function and in the *function* argument use the PERCENTOF function. Then use the related_to argument to switch between the three options:
0: Column Totals (Default), 1: Row Totals, 2: Grand Totals.
Compare to PT:

Product Sales Table:

Date	Product	Discount	Payment Method	Sales ($)
7/21	Carlota	Coupon	PayPal	45.9
5/16	Yanaki	Coupon	Mastercard	19.95
4/16	Aspen	No Coupon	PayPal	21.95
9/28	Yanaki	No Coupon	American Express	19.95
6/3	Yanaki	Coupon	Visa	79.8
8/23	Yanaki	Coupon	PayPal	79.8
2/25	Yanaki	Coupon	Visa	59.85
11/2	Yanaki	No Coupon	Discover	139.65
5/23	Yanaki	Coupon	PayPal	119.7
8/28	Aspen	No Coupon	Visa	43.9
5/26	Aspen	Coupon	PayPal	43.9
9/13	Yanaki	No Coupon	Mastercard	99.75
5/21	Carlota	No Coupon	Visa	114.75
8/18	Carlota	Coupon	PayPal	91.8
4/24	Carlota	No Coupon	American Express	114.75
11/2	Carlota	Coupon	Visa	22.95
5/18	Sunset	No Coupon	Discover	124.75
11/19	Yanaki	No Coupon	PayPal	19.95
8/31	Aspen	Coupon	PayPal	87.8
12/24	Carlota	Coupon	PayPal	22.95

Total Sales ($) for Payment Method by Discount:

=PIVOTBY(F10:F210,E10:E210,G10:G210,SUM)

	Coupon	No Coupon	Total
American Express	940	603	1,543
Discover	217	982	1,198
Mastercard	799	1,667	2,467
PayPal	2,881	1,932	4,813
Visa	1,213	3,778	4,991
Total	**6,050**	**8,962**	**15,012**

Three-in-one Cross-tabulated report:

	Coupon	No Coupon	Total
American Express	15.5%	6.7%	**10.3%**
Discover	3.6%	11.0%	**8.0%**
Mastercard	13.2%	18.6%	**16.4%**
PayPal	47.6%	21.6%	**32.1%**
Visa	20.1%	42.2%	**33.2%**
Total	**100.0%**	**100.0%**	**100.0%**

=PIVOTBY(F10:F210,E10:E210,G10:G210,PERCENTOF,,,,,,,N19)

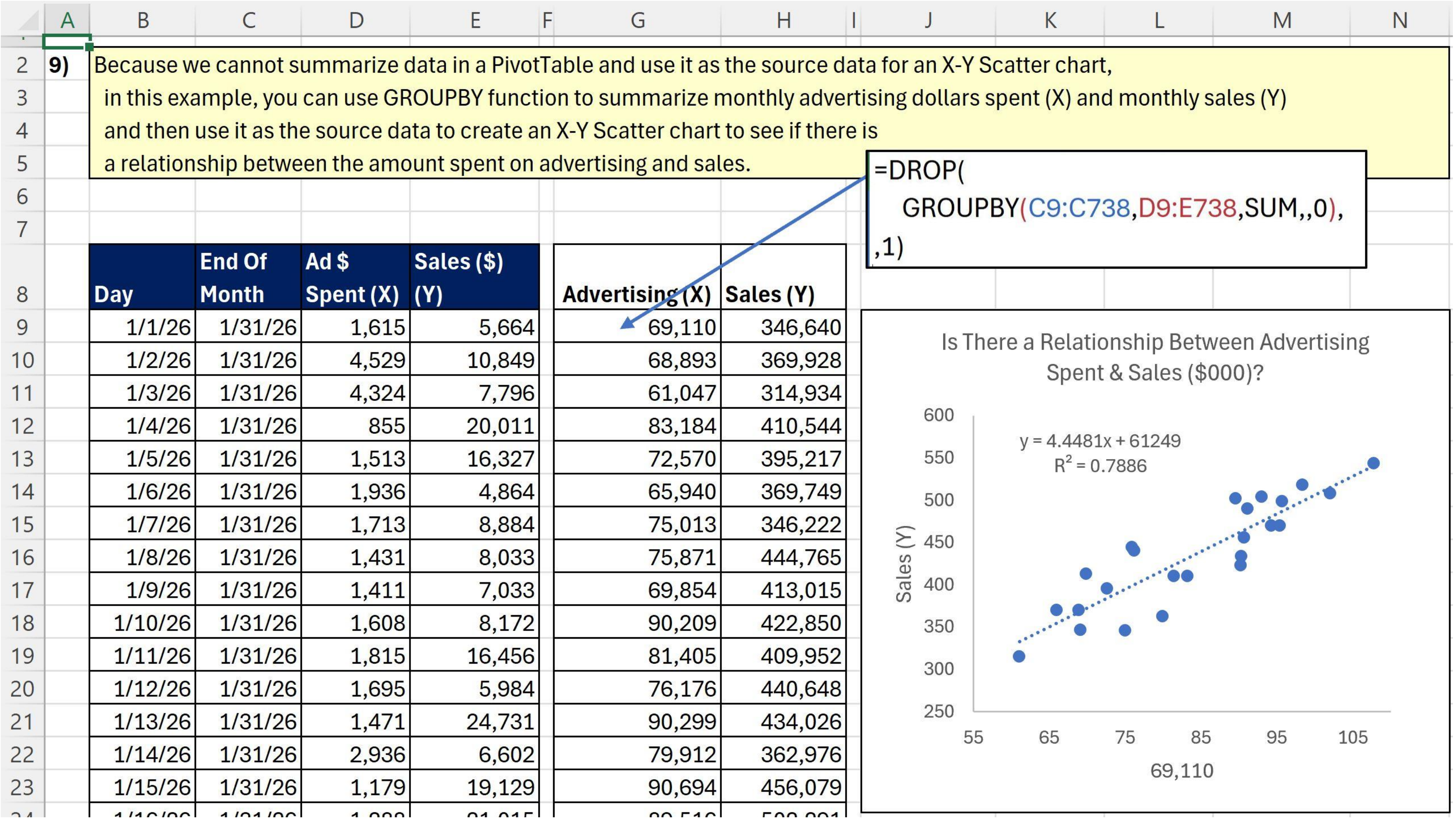
Select Report Type:
0: Column Totals (Default)

PIVOTBY relative_to argument:
0: Column Totals (Default)
1: Row Totals
2: Grand Totals
3: Parent Col Total
4: Parent Row Total

Example 9: GROUPBY to Help Create an X-Y Scatter Chart

Because we cannot summarize data in a PivotTable and use it as the source data for an X-Y Scatter chart, in this example, you can use GROUPBY function to summarize monthly advertising dollars spent (X) and monthly sales (Y) and then use it as the source data to create an X-Y Scatter chart to see if there is a relationship between the amount spent on advertising and sales. The solution is shown here:

The picture below shows the result:



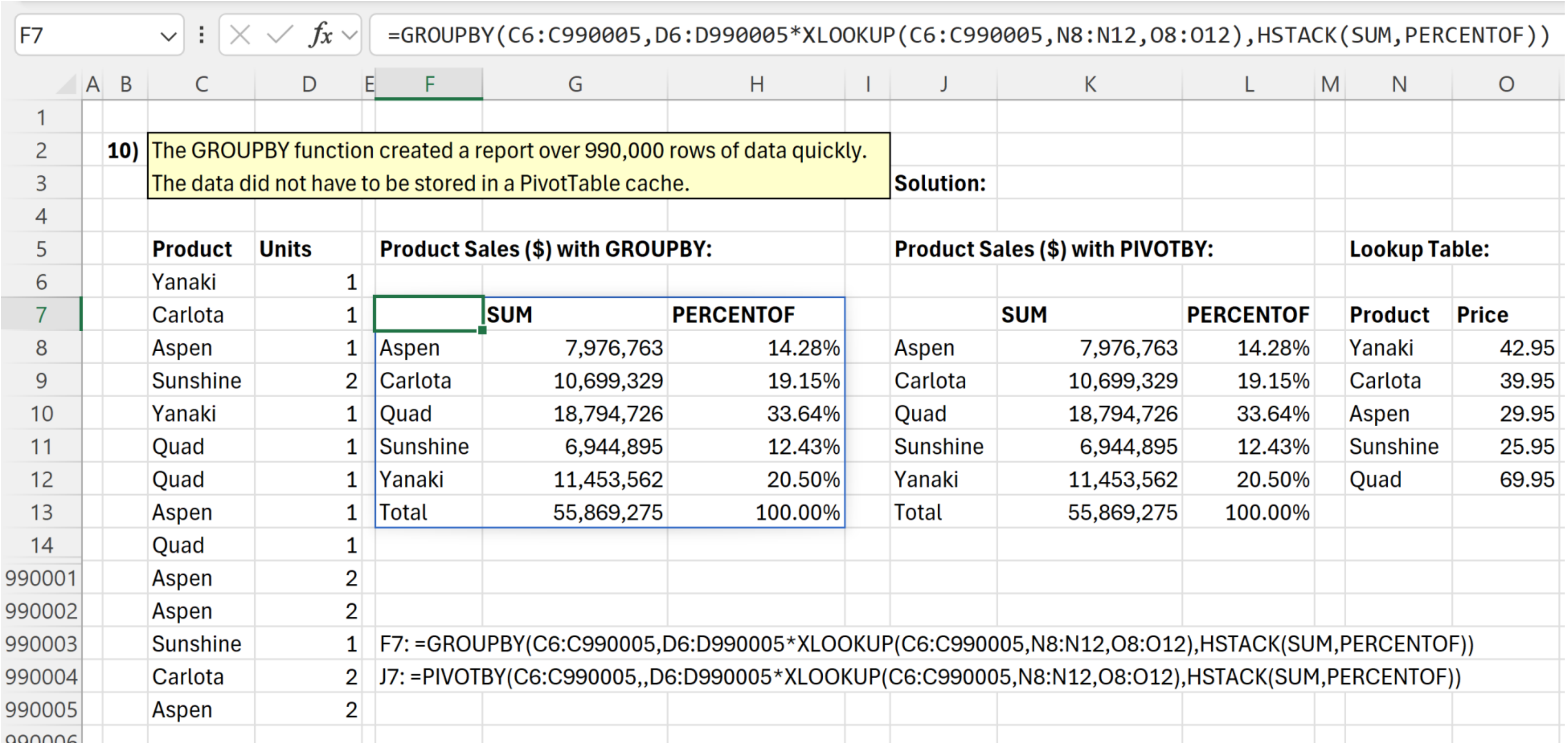
9) Because we cannot summarize data in a PivotTable and use it as the source data for an X-Y Scatter chart, in this example, you can use GROUPBY function to summarize monthly advertising dollars spent (X) and monthly sales (Y) and then use it as the source data to create an X-Y Scatter chart to see if there is a relationship between the amount spent on advertising and sales.

```
=DROP(
   GROUPBY(C9:C738,D9:E738,SUM,,0),
,1)
```

Day	End Of Month	Ad $ Spent (X)	Sales ($) (Y)	Advertising (X)	Sales (Y)
1/1/26	1/31/26	1,615	5,664	69,110	346,640
1/2/26	1/31/26	4,529	10,849	68,893	369,928
1/3/26	1/31/26	4,324	7,796	61,047	314,934
1/4/26	1/31/26	855	20,011	83,184	410,544
1/5/26	1/31/26	1,513	16,327	72,570	395,217
1/6/26	1/31/26	1,936	4,864	65,940	369,749
1/7/26	1/31/26	1,713	8,884	75,013	346,222
1/8/26	1/31/26	1,431	8,033	75,871	444,765
1/9/26	1/31/26	1,411	7,033	69,854	413,015
1/10/26	1/31/26	1,608	8,172	90,209	422,850
1/11/26	1/31/26	1,815	16,456	81,405	409,952
1/12/26	1/31/26	1,695	5,984	76,176	440,648
1/13/26	1/31/26	1,471	24,731	90,299	434,026
1/14/26	1/31/26	2,936	6,602	79,912	362,976
1/15/26	1/31/26	1,179	19,129	90,694	456,079

Is There a Relationship Between Advertising Spent & Sales ($000)?

Example 10: GROUPBY worked quickly over 990,000 rows of data and the data did not have to be stored in PivotTable Cache



F7 =GROUPBY(C6:C990005,D6:D990005*XLOOKUP(C6:C990005,N8:N12,O8:O12),HSTACK(SUM,PERCENTOF))

10) The GROUPBY function created a report over 990,000 rows of data quickly.
The data did not have to be stored in a PivotTable cache.

Solution:

Row	Product	Units
6	Yanaki	1
7	Carlota	1
8	Aspen	1
9	Sunshine	2
10	Yanaki	1
11	Quad	1
12	Quad	1
13	Aspen	1
14	Quad	1
990001	Aspen	2
990002	Aspen	2
990003	Sunshine	1
990004	Carlota	2
990005	Aspen	2

Product Sales ($) with GROUPBY:

	SUM	PERCENTOF
Aspen	7,976,763	14.28%
Carlota	10,699,329	19.15%
Quad	18,794,726	33.64%
Sunshine	6,944,895	12.43%
Yanaki	11,453,562	20.50%
Total	55,869,275	100.00%

Product Sales ($) with PIVOTBY:

	SUM	PERCENTOF
Aspen	7,976,763	14.28%
Carlota	10,699,329	19.15%
Quad	18,794,726	33.64%
Sunshine	6,944,895	12.43%
Yanaki	11,453,562	20.50%
Total	55,869,275	100.00%

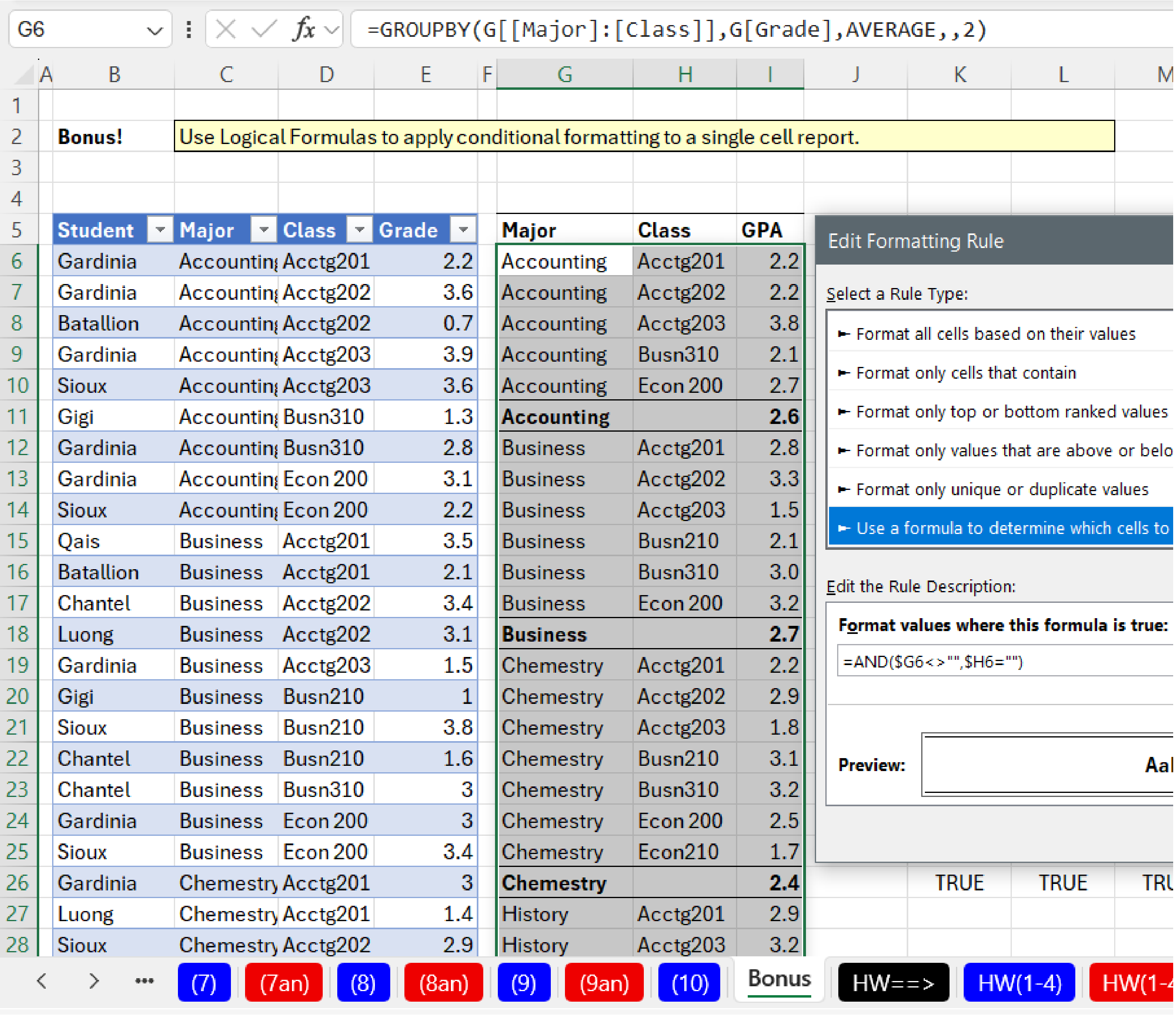
Lookup Table:

Product	Price
Yanaki	42.95
Carlota	39.95
Aspen	29.95
Sunshine	25.95
Quad	69.95

F7: =GROUPBY(C6:C990005,D6:D990005*XLOOKUP(C6:C990005,N8:N12,O8:O12),HSTACK(SUM,PERCENTOF))

J7: =PIVOTBY(C6:C990005,,D6:D990005*XLOOKUP(C6:C990005,N8:N12,O8:O12),HSTACK(SUM,PERCENTOF))

Bonus: Logical Formulas to Apply Conditional Formatting to a Single Cell Report



G6 | =GROUPBY(G[[Major]:[Class]],G[Grade],AVERAGE,,2)

	B	C	D	E	G	H	I	J	K	L	M
1											
2	**Bonus!**	Use Logical Formulas to apply conditional formatting to a single cell report.									
3											
4											
5	**Student**	**Major**	**Class**	**Grade**	**Major**	**Class**	**GPA**				
6	Gardinia	Accountin	Acctg201	2.2	Accounting	Acctg201	2.2				
7	Gardinia	Accountin	Acctg202	3.6	Accounting	Acctg202	2.2				
8	Batallion	Accountin	Acctg202	0.7	Accounting	Acctg203	3.8				
9	Gardinia	Accountin	Acctg203	3.9	Accounting	Busn310	2.1				
10	Sioux	Accountin	Acctg203	3.6	Accounting	Econ 200	2.7				
11	Gigi	Accountin	Busn310	1.3	**Accounting**		**2.6**				
12	Gardinia	Accountin	Busn310	2.8	Business	Acctg201	2.8				
13	Gardinia	Accountin	Econ 200	3.1	Business	Acctg202	3.3				
14	Sioux	Accountin	Econ 200	2.2	Business	Acctg203	1.5				
15	Qais	Business	Acctg201	3.5	Business	Busn210	2.1				
16	Batallion	Business	Acctg201	2.1	Business	Busn310	3.0				
17	Chantel	Business	Acctg202	3.4	Business	Econ 200	3.2				
18	Luong	Business	Acctg202	3.1	**Business**		**2.7**				
19	Gardinia	Business	Acctg203	1.5	Chemestry	Acctg201	2.2				
20	Gigi	Business	Busn210	1	Chemestry	Acctg202	2.9				
21	Sioux	Business	Busn210	3.8	Chemestry	Acctg203	1.8				
22	Chantel	Business	Busn210	1.6	Chemestry	Busn210	3.1				
23	Chantel	Business	Busn310	3	Chemestry	Busn310	3.2				
24	Gardinia	Business	Econ 200	3	Chemestry	Econ 200	2.5				
25	Sioux	Business	Econ 200	3.4	Chemestry	Econ210	1.7				
26	Gardinia	Chemestry	Acctg201	3	**Chemestry**		**2.4**		TRUE	TRUE	TRU
27	Luong	Chemestry	Acctg201	1.4	History	Acctg201	2.9				
28	Sioux	Chemestry	Acctg202	2.9	History	Acctg203	3.2				

Edit Formatting Rule

Select a Rule Type:

- Format all cells based on their values
- Format only cells that contain
- Format only top or bottom ranked values
- Format only values that are above or belo
- Format only unique or duplicate values
- Use a formula to determine which cells to

Edit the Rule Description:

Format values where this formula is true:

=AND($G6<>"",$H6="")

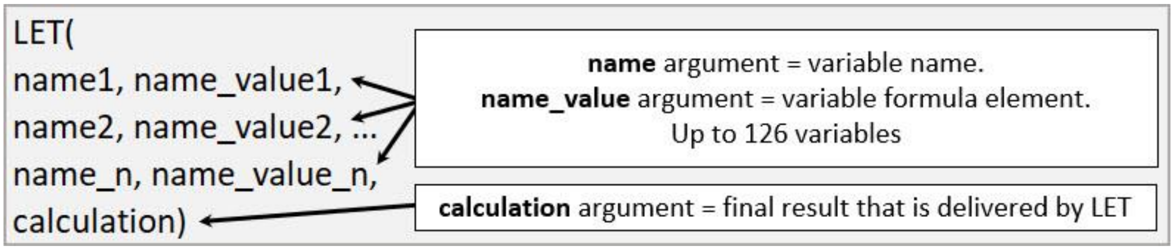
Preview: AaI

LET function to Define Variables (not covered in class)

The LET worksheet function allows you to define variables within the function itself and use the variables throughout the LET function to create a final calculation that is delivered to the worksheet or internally in other formulas. The advantages of using the LET function to create worksheet solutions are:

1. A variable is evaluated a single time, with the result stored in memory so that it can be used throughout the formula. For formulas with repeating formula elements, this can reduce overall calculation time by avoiding duplicate evaluation procedures.
2. Formulas with repeating elements are easier to edit because you have only one location to edit.
3. Complex formulas can be visually easier to read because each element is given a name and can be placed on a different line by using the keyboard for a line feed, Alt + Enter.
4. You can condense reports made up of multiple formulas into a single cell formula that spills the complete report into the worksheet.

The arguments for the LET function are shown in Figure 16.1.



LET(
name1, name_value1,
name2, name_value2, ...
name_n, name_value_n,
calculation)

name argument = variable name.
name_value argument = variable formula element.
Up to 126 variables

calculation argument = final result that is delivered by LET

Figure 16.1 Arguments in the LET worksheet function.

Full details about the LET function’s arguments are listed here:

- **name1** argument = Name of variable 1. Just as with Defined Names and Excel Table Names, you are not allowed to use spaces and other characters like * / + - ( ) ^ < > + & % ~ ` |] [} { @ ” ; : , ’ $ # !.
- **name_value1** argument = Worksheet elements such as references, numbers, functions and so on.
- **name2** argument = name of variable 2.
- **name_value2** argument = Worksheet elements or any previously define variable. Previously define variable will appear in a dropdown list along with defined names, table names and worksheet function names as you type the first few letters of the variable name in your formula.
- **name_n and name_value_n** arguments = You can list up to 126 variables.
- **calculation** argument = Formula that can use worksheet elements and any previously defined variables. This is the final result that is delivered by LET.

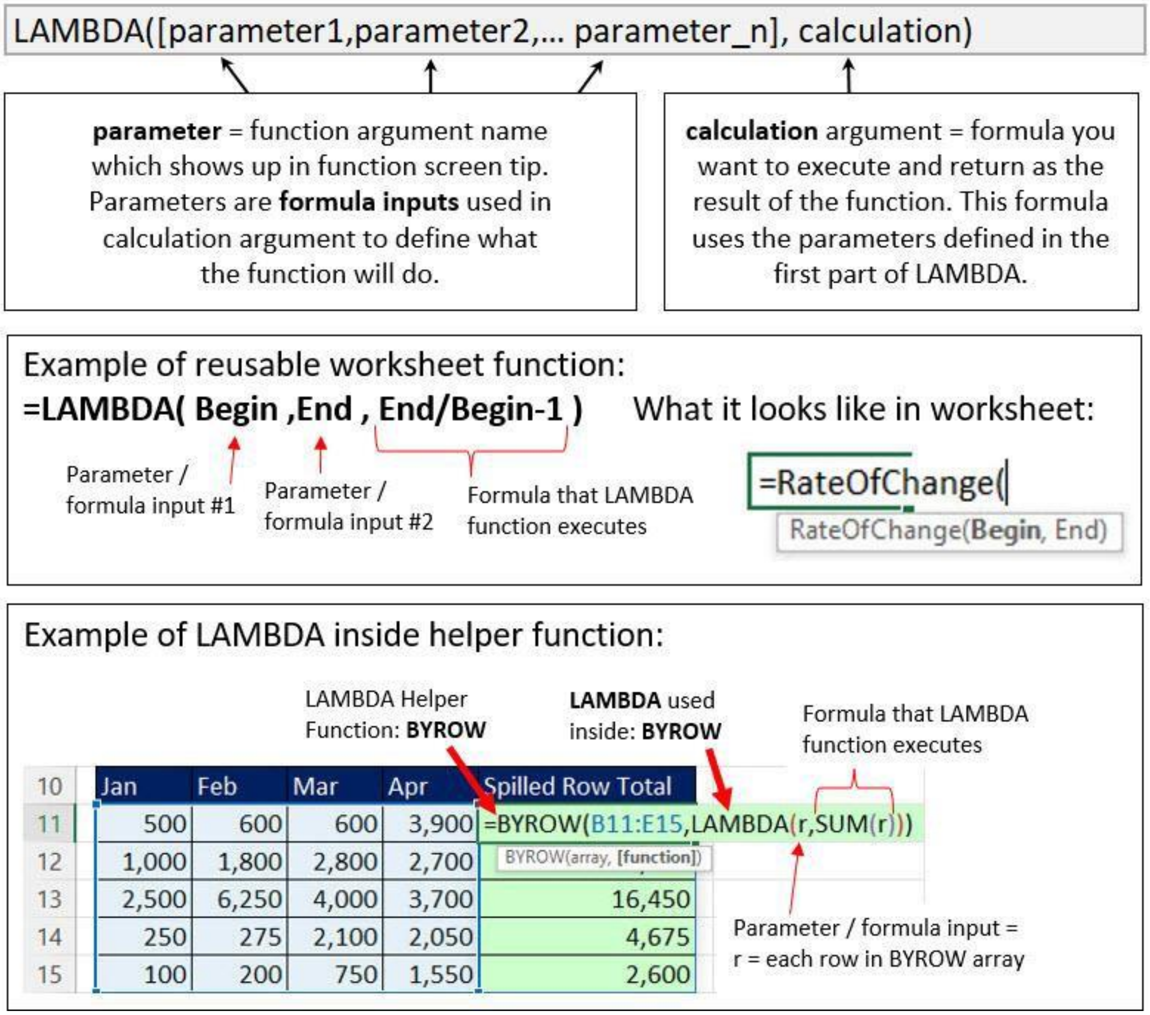
LAMBDA Function (not covered in this video)

Define LAMBDA function

- The LAMBDA function allows to create a custom function value, which: 1] Can be stored in a Defined Name to create a reusable function

  or

  2] Can be used in one of six LAMBDA Helper Functions for specific tasks such as spilling an aggregate calculation down a set of rows. When you use LAMBDA in a helper function you can use the formula directly in the worksheet or you can store it in a Defined Name to create a reusable function.
- The arguments for the LAMBDA function are shown here:



LAMBDA([parameter1,parameter2,... parameter_n], calculation)

parameter = function argument name which shows up in function screen tip. Parameters are **formula inputs** used in calculation argument to define what the function will do.

calculation argument = formula you want to execute and return as the result of the function. This formula uses the parameters defined in the first part of LAMBDA.

Example of reusable worksheet function:

=LAMBDA(Begin ,End , End/Begin-1)

What it looks like in worksheet:

=RateOfChange(

RateOfChange(**Begin**, End)

Parameter / formula input #1
Parameter / formula input #2
Formula that LAMBDA function executes

Example of LAMBDA inside helper function:

LAMBDA Helper Function: **BYROW**
LAMBDA used inside: **BYROW**
Formula that LAMBDA function executes
Parameter / formula input = r = each row in BYROW array
BYROW(array, [function])

	Jan	Feb	Mar	Apr	Spilled Row Total
10	Jan	Feb	Mar	Apr	Spilled Row Total
11	500	600	600	3,900	=BYROW(B11:E15,LAMBDA(r,SUM(r)))
12	1,000	1,800	2,800	2,700	
13	2,500	6,250	4,000	3,700	16,450
14	250	275	2,100	2,050	4,675
15	100	200	750	1,550	2,600