No.	Link to Sheet:	Topics:
1	AFNotes(1)	Array Formula Terms, Definition, and Types
2	ArrayConfig(1)	Array Formula Configurations
3	Fun	Fundamentals of Array Formulas
4	MakingThingsEasy	Array Formulas Make Things Easy
5	CompactSolutions	Array Formulas Help Make Compact Solutions
6	SUMPRODUCT	SUMPRODUCT function
7	ArrayFunctions	List of Array Functions
8		Array Function Examples:
9	Re-Orient	Re-Orient Data. Function examples: CHOOSECOLS, CHOOSEROWS, DROP, TAKE, VSTACK, HSTACK, SORT, TOCOL, TOROW, TRANSPOSE, WRAPCOLS, WRAPROWS, SORTBY, UNIQUE, EXPAND, FILTER
10	Create No. Array	Create Arrays of Numbers. Function examples: RANDARRAY, SEQUENCE
11	Stats	Statistics & Matrix Algebra. Function examples: FREQUENCY, LINEST, MODE.MULT, TREND
12	Text	Text. Function example: TEXTSPLIT

Array Formula Terms, Definition, and Types by excelisfun

1) An **array** is a collection of two or more items.

Sale ($)	Tax ($)
123.87	94.76
52.14	39.89
3,158.49	2,416.24

array

C6: =ROUND(B6:B8 * B11,2)

Tax Rate	Add Top 2 ($)
0.765	3,282.36

array

C11: =SUM(LARGE(B6:B8, {1;2}))

2) **Items allowed in a worksheet array are:**

	Items:
Numbers:	43
Text:	Excel
Boolean:	TRUE
Errors:	#DIV/0!
Empty Cells:	

Add ($)
43

E18: =AGGREGATE(9,6, C18:C22)

array contains all 5 items

3) **Array constant** = vales typed into formula (hard coded in) using array syntax.

Array Syntax:

Curly Brackets house the array: { }
Comma means column ,
Semi-colon means row ;

Formula typed into E32: ={1,2,3}

Array constant of column values in a row:

1	2	3

Array constant of row values in a column:

1
2
3

Formula typed into E34: ={1;2;3}

Formula typed into E38: ={1,"Jan";2,"Feb";3,"Mar"}

Array constant of table values:

1	Jan
2	Feb
3	Mar

4) A **worksheet array formula** is a formula that contains at least one operation that delivers an array of answers rather than a single answer.

Divide by	2

Numbers	Divide by 2
10	5
8	4

C46: =B46:B47/F44 = {10;8}/2 = {5;4}

math operation delivers an array of answers and therefore this is an array formula

List	Count Unique:
Quad	2
Aspen	
Quad	

C50: =ROWS(UNIQUE(B50:B52)) = ROWS({"Quad";"Aspen"}) = 2

UNIQUE function delivers an array of answers into ROWS function and therefore this is an array formula

5) An **aggregate operation** operates on an array of items, but delivers a single answer.

Numbers
10
8

Add Numbers
18

D58: =SUM(B58:B59) = 18

NOT an array formula!!

The SUM aggregate operation delivers a single answer.
SUM and all other aggregate functions can not deliver an array of answers.

6) The **final answer from an array formula** can either be a *range of spilled values* or a *single value* .

When the last operation in an array formula is an array operation, the formula delivers a *range of spilled values* into the worksheet and is called a **dynamic spilled array formula**.

Numbers	Divide by 2
10	5
8	4

C69: =B69:B70/G68

Divide by	2

dynamic spilled array formula

When the last operation in an array formula is an aggregate operation, the formula delivers a *single value* and is called a **scalar array formula**.

Total
9

C76: =SUM(B69:B70/G68)

array operation makes it an array formula

scalar array formula

SUM to deliver one answer makes it a scalar array formula

7) When a formula operation delivers an array of answers, the operation is called an **array operation**. The array of answers generated by an array operation is called a **resultant array**.

resultant array

C46: =B46:B47/F44 = {10;8}*2 = {5;4}

math array operation

C50: =ROWS(UNIQUE(B50:B52)) = ROWS({"Quad";"Aspen"}) = 2

array function operation

8) **Types** of worksheet arrays:

1. Range reference
2. Array constant
3. Resultant array
4. Dynamic spilled array

Sale ($)
123.87
52.14
3,158.49

Tax Rate
0.765

Top Two ($)
2416.24
94.76

1. range reference **2. array constant**

D99: =LARGE(ROUND(B96:B98 * D96,2), {1;2})

Calculation steps:

LARGE({94.76;39.89;2416.24} ,{1;2})

3. resultant array from LARGE → {2416.24;94.76}

3. resultant array from ROUND

Day	Fiscal Quarter
4/12/23	Q4
5/9/23	Q4
11/13/23	Q2
2/8/23	Q3
9/8/23	Q1

4. dynamic spilled array

3. array constant (full table)

C108: =LOOKUP(MONTH(B108:B112), {1,"Q3";4,"Q4";7,"Q1";10,"Q2"})

9) **Array operations that are possible in a worksheet are:**

1] Direct array operation: math, comparative, join operator works directly against an array of values, such as:

Quad Cost	Aspen Cost
10	8

Quad Cost*2	Aspen Cost*2	Quad Cost>=2	Aspen Cost>=2	Quad Cost&2	Aspen Cost&2
20	16	TRUE	TRUE	102	82

Math:
C10: =C7:D7*2
Calculation steps:
{10,8}*2
{10*2,8*2}
{20,16}

Comparative:
E10: =C7:D7>=2
Calculation steps:
{10,8}>=2
{10>=2,8>=2}
{TRUE,TRUE}

Join:
G10: =C7:D7&2
Calculation steps:
{10,8}&2
{10&2,8&2}
{"102","82"}

2] Function argument array operation: function argument expects a single value, and you give it an array of values. A function argument array operation forces the function to deliver an array of answers, such as:

Tax Rate	10.5%

Employee	Gross Pay ($)	Tax ($)
Chantel	2,379.09	249.8
Kamala	2,630.57	276.21
Sioux	1,884.75	197.9

ROUND:
E25: =ROUND(D25:D27*D22,2) =
Calculation steps:
ROUND({249.804;276.209;197.898} ,2)
{249.80;276.21;197.90}

List 1	List 2	Item in List 2,
Gigi	Chantel	#N/A
Ty	Gigi	1
Sioux	Ty	2
	Miki	#N/A

XMATCH:
E31: =XMATCH(D31:D34,C31:C33)
Calculation steps:
{#N/A;1;2;#N/A}

3] Array function operation: functions such as UNIQUE and are programmed to deliver an array of answers.

No.	Products
1	Quad
2	Carlota
3	Quad
4	Quad

Unique List:
Quad
Carlota

UNIQUE:
F39: =UNIQUE(D39:D42)
Calculation steps:
{"Quad";"Carlota"}

SEQUENCE:
C39: =SEQUENCE(4)
Calculation steps:
{1;2;3;4}

10) **Dynamic Spilled Array Formula Notes:**

When you create a 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:
    * Range argument of the functions SUMIF, COUNTIF, and AVERAGEIF.
    * Criteria_range argument of the functions SUMIFS, COUNTIFS, AVERAGEIFS, MINIFS, & MAXIFS.
    * First argument of the functions SUMIFS, AVERAGEIFS, MINIFS, and MAXIFS COUNTIFS.
    * 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}).

11) Array Operation Configurations, the Evaluation Process, and the Size of the Resultant Array

Multiple	2

Sales	Tax ($)
10	20
20	40

Cell * Column:
D7: =D4*C7:C8
=2*{10;20} = {2*10;2*20} = {20;40}
=Cell (1R x 1C) * Column (2R x 1C) = Column (2R x 1C)

Price ($)	Units	Sales
2	5	10
5	4	20

Column * Column (same number of rows):
D12: =B12:B13*C12:C13
={2;5}*{5;4} = {2*5;5*4} = {10;20}
=Column (2R x 1C) * Column (2R x 1C) = Column (2R x 1C)

Price ($)	Units	Sales
2	5	10
5	4	20
	2	#N/A

Column * Column (different number of rows):
D17: =B17:B18*C17:C19
={2;5}*{5;4;2} = {2*5;5*4;???*4} = {10;20;#N/A}
=Column (2R x 1C) * Column (3R x 1C) = Error because columns do not contain same number of rows.

Sales	10	20
Tax Paid	20	40

Cell * Row:
C22: =D24*C21:D21
=2*{10,20} = {2*10,2*20} = {20,40}
=Cell (1R x 1C) * Row (1R x 2C) = Row (1R x 2C)

Multiple	2

Price ($)	2	5
Units	5	4
Sales	10	20

Row * Row (same number of columns):
C28: =C26:D26*C27:D27
={2,5}*{5,4} = {2*5,5*4} = {10,20}
=Row (1R x 2C) * Row (1R x 2C) = Row (1R x 2C)

Price ($)	2	5	
Units	5	4	2
Sales	10	20	#N/A

Row * Row (different number of columns):
C33: =C31:D31*C32:E32
={2,5}*{5,4,2} = {2*5;5*4;???*4} = {10,20,#N/A}
=Row (1R x 2C) * Row (1R x 3C) = Error because rows do not contain same number of columns

	B	C	D
		10	**20**
	2	20	40
	1	10	20
	3	30	60

Array Formula Process for Column * Row:

C3: =B3:B5*C2:D2

={2;1;3}*{10,20} = {2*10,2*20;1*10,1*20;3*10,3*20} = {20,40;10,20;30,60}

=Column (3R x 1C) * Row (1R x 2C) = Table (3R x 2C)

	B	C	D	E
		2	**1**	**3**
	10	20	10	30
	20	40	20	60

Array Formula Process for Row * Column:

C8: =C7:E7*B8:B9

={2,1,3}*{10;20} = {2*10,1*10,3*10;2*20,1*20,3*20} = {20,10,30;40,20,60}

=Column (3R x 1C) * Row (1R x 2C) = Table (3R x 2C)

	B	C
	14	10
	4	15
	3	24

Array Formula Process for Table * Table:

B12: =B16:C18*B20:C22

={2,5;4,3;1,6}*{7,2;1,5;3,4} = {2*7,5*2;4*1,3*5;1*3,6*4} = {14,10;4,15;3,24}

Table (3R x 2C) * Table (3R x 2C) = Table (3R x 2C)

	B	C
	2	**5**
	4	**3**
	1	**6**

	B	C
	7	**2**
	1	**5**
	3	**4**

	B	C
	1	12
	10	8
	1	8

Array Formula Process for Column * Row * Table:

B24: =B30:B32*C29:D29*C30:D32

={1;2;1}*{1,4}*{1,3;5,1;1,2} = {1*1,1*4;2*1,2*4;1*1,1*4}*{1,3;5,1;1,2} =

{1,4;2,8;1,4}*{1,3;5,1;1,2} = {1*1,4*3;2*5,8*1;1*1,4*2] = {1,12;10,8;1,8}

=Column (3R x 1C) * Row (1R x 2C) * Table (3R x 2C) = Table (3R x 2C)

	B	C	D
	Row → Column ↓	**1**	**4**
	1	1	3
	2	5	1
	1	1	2

Total Sales

Product/SalesRep	Sioux	Ty
Aspen	20	40
Quad	20	20

Product	SalesRep	Sales
Quad	Sioux	10
Aspen	Ty	20
Quad	Ty	20
Aspen	Ty	20
Aspen	Sioux	20
Quad	Sioux	10

Function Argument Array Operation in SUMIFS

C4: =SUMIFS(D8:D13,B8:B13, B4:B5,C8:C13, C3:D3)

=SUMIFS(D3:D8,B3:B8, {"Aspen";"Quad"} ,C3:C8, {"Sioux","Ty"})

Two function argument array operations:

{"Aspen";"Quad"} = Column (2R x 1C)

{"Sioux","Ty"} = Row (1R x 2C)

SUMIFS delivers a cross tabulated table (2R x 2C) like this:

{20,40;20,20}

Contains Criteria
Carlota
Quad

Product
Quad
Aspen
Quad
Aspen
Aspen
Quad

Contains?
#VALUE!
#VALUE!
#N/A
#N/A
#N/A
#N/A

Function Argument Array Operation in SEARCH

D16: =SEARCH(B16:B17,B20:B25)

=SEARCH({"Carlota";"Quad"} ,B20:B25)

Function argument array operation:

{"Carlota";"Quad"} = Column (2R x 1C),

Because B20:B25 is 6 rows and {"Carlota";"Quad"} is 2 rows,

SEARCH delivers a column of errors:

{#VALUE!;#VALUE!;#N/A;#N/A;#N/A;#N/A}

When more than one row is involved in array operation,

they must have same number of rows.

When they do not, we get an error.

Commission Rate
0.0575

Date	Employee	Sales($)	Commission $
10/26/22	Chantel Mimms	1,211.71	69.67
10/26/22	Nanjala Gwenevere	1,269.93	73.02
10/26/22	Sioux Radcoolinator	879.40	50.57
10/26/22	Timmy Smith	957.74	55.07
10/27/22	Chantel Mimms	1,212.54	69.72
10/27/22	Nanjala Gwenevere	652.94	37.54
10/27/22	Sioux Radcoolinator	891.45	51.26
10/27/22	Timmy Smith	128.30	7.38
10/28/22	Chantel Mimms	890.25	51.19
10/28/22	Nanjala Gwenevere	719.90	41.39
10/28/22	Sioux Radcoolinator	724.66	41.67
10/28/22	Timmy Smith	1,213.85	69.8
10/29/22	Chantel Mimms	11,222.47	645.29
10/29/22	Nanjala Gwenevere	12,990.78	746.97
10/29/22	Sioux Radcoolinator	2,957.70	170.07
10/29/22	Abigail Sinclaire	3,786.09	217.7
10/30/22	Chantel Mimms	5,516.06	317.17
10/30/22	Nanjala Gwenevere	5,047.96	290.26
10/30/22	Sioux Radcoolinator	14,338.83	824.48
10/30/22	Abigail Sinclaire	10,887.73	626.04

New Records:

10/29/22	Chantel Mimms	11222.47
10/29/22	Nanjala Gwenevere	12990.78
10/29/22	Sioux Radcoolinator	2957.7
10/29/22	Abigail Sinclaire	3786.09
10/30/22	Chantel Mimms	5516.06
10/30/22	Nanjala Gwenevere	5047.96
10/30/22	Sioux Radcoolinator	14338.83
10/30/22	Abigail Sinclaire	10887.73

Total Sales
77,500.29

Total Comm$
4,456.26

Unique Dates
10/26/22
10/27/22
10/28/22
10/29/22
10/30/22

Employee	Total Sales
Abigail Sinclaire	14673.82
Chantel Mimms	20053.03
Nanjala Gwenevere	20681.51
Sioux Radcoolinator	19792.04
Timmy Smith	2299.89

2 Biggest Sales	Sales Rep?
14,338.83	Sioux Radcoolinator
12,990.78	Nanjala Gwenevere

G5: =SUM(Sales016[Sales($)])
I5: =UNIQUE(Sales016[Date])
K5: =SORT(UNIQUE(Sales016[Employee]))
L5: =SUMIFS(Sales016[Sales($)],Sales016[Employee],K5#)
N5: =LARGE(Sales016[Sales($)],{1;2})
O5: =XLOOKUP(N5#,Sales016[Sales($)],Sales016[Employee])
G9: =SUM(ROUND(Sales016[Sales($)]*E2,2))
K12: =ShowFormulas(G4:O41)
G23: =Sales016[Employee]&" comm = "&ROUND(Sales016[Sales($)]*E2,2)

Join:
Chantel Mimms comm = 69.67
Nanjala Gwenevere comm = 73.02
Sioux Radcoolinator comm = 50.57
Timmy Smith comm = 55.07
Chantel Mimms comm = 69.72
Nanjala Gwenevere comm = 37.54
Sioux Radcoolinator comm = 51.26
Timmy Smith comm = 7.38
Chantel Mimms comm = 51.19
Nanjala Gwenevere comm = 41.39
Sioux Radcoolinator comm = 41.67
Timmy Smith comm = 69.8
Chantel Mimms comm = 645.29
Nanjala Gwenevere comm = 746.97
Sioux Radcoolinator comm = 170.07
Abigail Sinclaire comm = 217.7
Chantel Mimms comm = 317.17
Nanjala Gwenevere comm = 290.26
Sioux Radcoolinator comm = 824.48
Abigail Sinclaire comm = 626.04

Hurdle:
60

Boolean:
TRUE
TRUE
FALSE
FALSE
TRUE
FALSE

Array = 2 or more items
Aggregate Operation = operates on an array of items, but delivers a single answer
Array Operation = formula operation that delivers an array of answers
Types of Operations =
Direct
Math
Join
Logical
Function argument
Function
Resultant Array = array created by array operation
Types of Arrays =
Range
Resultant Array
Dynamic Spilled Array
Array Constant: {} House Array. , = Column. ; = Row.
Define Worksheet Array Formula = formula that contains at least one operation that deliver an array of answers, rather than a single answer
Types of Array Formulas =
Dynamic Spilled Array Formulas
Scalar Array Formula
Dynamic Array Formula Characteristics:
1. Dynamic spilled array formula lives in top cell
2. Formula spills down from and to right of the top cell
3. Edit formula in top cell
4. Cells below top cell do not contains values, but can be referenced
5. If cell content is in way of spilled results, you get a #SPILL! error
6. Spilled Range Operator is: #. Refer to dynamic spilled array with top cell and #, like: I5#
7. Spilled arrays not allowed in Excel Tables

Why Array Formulas are useful:

1) Make Worksheet Formulas Easier Than Before with Dynamic Spilled Array Formulas

Dynamic Spilled Array Formulas:

Don't have to lock cell references

Don't have to manually copy formula

Editing is done in top cell with no re-copy

Goal 1: Create Profit Budget

Expenses as % of Sales	%
COGS Expense	0.375
Research Expense	0.125
Selling & Marketing Ex.	0.0975
Interest Expense	0.05
Income Tax Expense	0.175
Other Expense	0.015

Budget Income Statement by Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
Revenue	5,000	4,500	4,375	8,540	9,700	1,452	1,500	1,125	9,050	1,375	1,950	2,275	50,842
COGS Expense	1,875	1,688	1,641	3,203	3,638	545	563	422	3,394	516	731	853	19,066
Research Expense	625	563	547	1,068	1,213	182	188	141	1,131	172	244	284	6,355
Selling & Marketing Ex.	488	439	427	833	946	142	146	110	882	134	190	222	4,957
Interest Expense	250	225	219	427	485	73	75	56	453	69	98	114	2,542
Income Tax Expense	875	788	766	1,495	1,698	254	263	197	1,584	241	341	398	8,897
Other Expense	75	68	66	128	146	22	23	17	136	21	29	34	763
Total Expenses	4,188	3,769	3,664	7,152	8,124	1,216	1,256	942	7,579	1,152	1,633	1,905	42,580
Net Income (Profit)	813	731	711	1,388	1,576	236	244	183	1,471	223	317	370	8,262

F12: =ROUND(F11:Q11*C12:C17,2)

Goal 2: Create Schedule of Future Value Retirement Amounts

Retirement Assumptions:	
Start Years to invest	10
Number Year Examples	8
Year Increment	5
Start Rate of Return	2.0%
Number Rate Examples	6
Rate Increment	2.0%
Yearly Deposit ($)	10,000
Initial Deposit ($)	5,000

Year/Rate of Return	0.02	0.04	0.06	0.08	0.1	0.12
10	115,592	127,462	140,762	155,660	172,343	191,017
15	179,664	209,241	244,742	287,382	338,611	400,165
20	250,403	308,736	383,892	480,924	606,387	768,756
25	328,506	429,788	570,104	765,302	1,037,644	1,418,339
30	414,738	577,066	819,299	1,183,145	1,732,187	2,563,126
35	509,944	756,253	1,152,778	1,797,095	2,850,756	4,580,633
40	615,060	974,260	1,599,048	2,699,188	4,652,222	8,136,169
45	731,116	1,239,500	2,196,258	4,024,658	7,553,501	14,402,238

F26: =FV(F25#,E26#,-C32,-C33)

Why Array Formulas are useful:

2) Help Build More Compact Solutions

Scarlar Array Formulas can eliminate unnecessay intermediate steps in the worksheet

Goal 1: Estimate returns for stock A & B in a portfolio of stocks.

Weight of Stock in Portfolio:		60.00%	40.00%
Probability of Economic State		**Stock A Estimated Return**	**Stock B Estimated Return**
Bad	35.00%	2.50%	-7.50%
OK	25.00%	8.50%	12.50%
Good	40.00%	14.50%	22.00%
		Estimated Portfolio Returns:	9.00%

Stock A Contribution to Portfolio	**Stock B Contribution to Portfolio**
0.00525	-0.0105
0.01275	0.0125
0.0348	0.0352
Estimated Portfolio Returns:	9.00%

G11: =C11:C13*D9:E9*D11:E13

H14: =SUM(G11#)

Example of More Compact Solution that

E14: =SUM(D11:E13*C11:C13*D9:E9)

Goal 2: Estimate the Overall Profit for Next Year

Expenses as % of Sales	%
COGS Expense	0.375
Research Expense	0.125
Selling & Marketing Ex.	0.0975
Interest Expense	0.05
Income Tax Expense	0.175
Other Expense	0.015
Revenue	$
Jan	5,000
Feb	4,500
Mar	4,375
Apr	8,540
May	9,700
Jun	1,452
Jul	1,500
Aug	1,125
Sep	9,050
Oct	1,375
Nov	1,950
Dec	2,275

Net Year Estimated Profit:
8261.74

G21: =SUM(-ROUND(TOROW(E28:E39)*E21:E26,2),E28:E39)

Example of More Compact Solution that

Goal: Calculate Standard deviation for the portfolio that contains stock A & B.

Weight of Stock in Portfolio:		0.6	0.4
Probability of Economic State		Stock A Full Estimated Return	Stock B Full Estimated Return
Bad	0.5	0.0	-0.15
OK	0.4	0.05	0.05
Great	0.1	0.1	0.2

1) Weight * Return		2) Add for Econ State	3. Subtract Mean & Square	4) * Prob of Econ
0	-0.06	-0.06	0.004096	0.002048
0.03	0.02	0.05	0.002116	0.0008464
0.06	0.08	0.14	0.018496	0.0018496

5) Add = Var	0.004744
6) SQRT = SD	0.068876701
Expected Portfolio Returns:	0.0040

Example of More Compact Solution that avoids all worksheet cell steps that are listed above:

Standard Deviation of Ex. Portfolio Returns:	0.068876701
Standard Deviation of Ex. Portfolio Returns:	0.068876701

M16: =SQRT(SUM(BYROW(D6:E8*D4:E4,LAMBDA(r,(SUM(r)-M12)^2))*C6:C8))

M17: =SQRT(SUM((MMULT(D6:E8*D4:E4,SEQUENCE(COLUMNS(D4:E4),1,1,0))-M12)^2*C6:C8))

Question:

Executive at HMO wanted to check Patient Record Data accuracy. An independent and random sample of medical records was taken for each of the four years.

Year	SurveyAnswer
2011	Yes
2012	Yes
2013	No
2014	No
2011	No
2012	No
2013	No
2014	No
2011	No
2012	No
2013	No
2014	No
2011	No
2012	No
2013	No
2014	No
2011	Yes
2012	No
2013	No
2014	No
2011	Yes
2012	No
2013	Yes
2014	No
2011	No
2012	No
2013	No
2014	No
2011	No
2012	No
2013	No
2014	No
2011	Yes

Count	Year				
SurveyAnswer	2011	2012	2013	2014	Grand Total
Yes	39	43	45	41	168
No	304	223	271	341	1139
Grand Total	343	266	316	382	1307

% Column Total	Year				
SurveyAnswer	2011	2012	2013	2014	Grand Total
Yes	11.37%	16.17%	14.24%	10.73%	12.85%
No	88.63%	83.83%	85.76%	89.27%	87.15%
Grand Total	100.00%	100.00%	100.00%	100.00%	100.00%

Expected Value	Year				
SurveyAnswer	2011	2012	2013	2014	Grand Total
Yes	44.08875	34.19128	40.61821	49.10176	168
No	298.9112	231.8087	275.3818	332.8982	1139
Grand Total	343	266	316	382	1307

Deviation^2	Year			
SurveyAnswer	2011	2012	2013	2014
Yes	25.89541	77.59359	19.20009	65.63851
No	25.89541	77.59359	19.20009	65.63851

Deviations Squared/Expected Frequencies:				
SurveyAnswer	2011	2012	2013	2014
Yes	0.587347	2.269397	0.472697	1.336785
No	0.086632	0.334731	0.069722	0.197173

Test Statistic Chi-Square = χ^2 =	5.354484

Example of More Compact Solution that avoids all worksheet cell steps that are listed above:

Test Statistic Chi-Square = χ^2 =	5.354484

G37: =SUM((G9:J9*K7:K8/K9-G7:J8)^2/(G9:J9*K7:K8/K9))

SUMPRODUCT function is an aggregate function that is programmed to make array calculations, but deliver an aggregate answer.
SUMPRODUCT can multiple 2 or more same size arrays, then add the resultant result to get an aggregate sum total
Arguments: SUMPRODUCT(array1,[array2] …)

Goal 1: Calculate total sales (price * units) in single cell.

Product	Price	Units
c	49.05	6
q	13.31	126
a	26.79	1
c	37.92	167
q	49.92	78
a	40.37	87
c	25.83	32
q	44.77	105
a	18.52	66
c	46.21	163
q	10.7	142
a	30.82	141
c	36.35	31
q	44.79	137
a	16.24	21
c	10.64	39
q	35.68	147
a	38.54	157
c	13.18	78
q	30.97	32
a	46.33	109
c	36.31	4
q	47.16	1
a	47.95	40
c	42.54	146
q	26.3	186
c	46.75	10

Total Sales ($)
284,443,532.22

Total Sales ($)
284,443,532.22

F9: =SUM(C9:C100560*D9:D100560)
F12: =SUMPRODUCT(C9:C100560,D9:D100560)

Time in milliseconds:

SUM Times:	SUMPRODUCT times:
7.3	6.6
7.3	6.5
7.5	6.6
7.4	6.6
7.38	6.58

% Decrease:	-10.85%

Goal 2: Multiply each test score by each test weight and then add to get student total score.
Example: Sioux's Total Score = 91*20% + 94*25.00% + 91*15.00% + 87.5*40.00% = 18.2 + 23.5 + 13.65 + 35 = 90.35

Weights:	20%	25%	15%	40%

Name	Test 1	Test 2	Test 3	Test 4	Weighted Ave.	Weighted Ave.	Weighted Ave.	Weighted Ave.
Sioux	91	94	91	87.5	90.35	90.35	90.35	90.35
Chin	98	87.5	79	86	87.73	87.73	87.73	87.725
Ty	73	36	56	78	63.20	63.20	63.20	63.2
Mo	65	70	72	84	74.90	74.90	74.90	74.9

Test	Weights
Test 1	20%
Test 2	25%
Test 3	15%
Test 4	40%

K24: =SUMPRODUCT(G24:J24,$G$21:$J$21)
L24: =SUMPRODUCT(G24:J24,TOROW($G$30:$G$33))
M24: =G24*$G$21+H24*$H$21+I24*$I$21+J24*$J$21
N24: =MMULT(G24:J24,$G$30:$G$33)

No.	Array Function	Description	Arguments
		Re-Orient Data Functions:	
1	CHOOSECOLS	Returns the specified column/s from an array with column position numbers.	CHOOSECOLS(array,col_num1,[col_num2],…)
2	CHOOSEROWS	Returns the specified row/s from an array with row position numbers.	CHOOSEROWS(array,row_num1,[row_num2],…)
3	DROP	Drops rows or columns from array start (positive number) or end (negative number).	DROP(array, rows,[columns])
4	TAKE	Takes rows or columns from array start (positive number) or end (negative number).	TAKE(array, rows,[columns])
5	VSTACK	Appends arrays vertically and in sequence to return a larger array.	VSTACK(array1,[array2],...)
6	HSTACK	Appends arrays horizontally and in sequence to return a larger array.	HSTACK(array1,[array2],...)
7	TOCOL	Returns the array as a single column. Stacks row values by default, unless you use 3rd argument.	TOCOL(array, [ignore], [scan_by_column])
8	TOROW	Returns the array as a single row. Stacks row values by default, unless you use 3rd argument.	TOROW(array, [ignore], [scan_by_column])
9	WRAPCOLS	Wraps the provided row or column (one-dimension) of values by columns after a specified number of elements to form a new array. Use 3rd argument to pad when there are not enough values to wrap full array.	WRAPCOLS(vector, wrap_count, [pad_with])
10	WRAPROWS	Wraps the provided row or column (one-dimension) of values by rows after a specified number of elements to form a new array. Use 3rd argument to pad when there are not enough values to wrap full array.	WRAPROWS(vector, wrap_count, [pad_with])
11	EXPAND	Expands or pads an array to specified row and column dimensions, where row or column number must be >= the number of rows or columns, respectively, in the original array. Use 4th argument to pad when there are not enough values to wrap full array.	EXPAND(array, rows, [columns], [pad_with])
12	TRANSPOSE	Converts a vertical array into a horizontal array or vice versa.	TRANSPOSE(array)
13	SORT	Sorts a row, a column, or a table in ascending or descending order. 2nd argument says which column in table to sort. Default sort order is A-Z.	SORT(array, [sort_index], [sort_order], [by_col])
14	SORTBY	Sorts an array by the values in one or more corresponding arrays. Default sort order is A-Z.	SORTBY(array, by_array, [sort_order], [array/order], ...)
15	UNIQUE	Creates a unique list of values or records. By default it yields a unique set of row records from a column or table. 2nd argument allows you to get a unique list by columns. 3rd argument allows you to extract a list of items only listed one time in original list.	UNIQUE(array, [by_col], [exactly_once])
16	FILTER	Filters a dataset based on conditions and criteria that you specify in a Boolean array logical test in the allow argument.	FILTER(array, include, [if_empty])

No.	Array Function	Description	Arguments
		Create Arrays of Numbers:	
17	RANDARRAY	Creates an array of random numbers based on a min and max number using a uniform distribution. If you enter just a row number, it generates an array of numbers that span from zero to one, each with 15 digits.	RANDARRAY([rows],[columns],[min],[max], [whole_number])
18	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])
		Statistics and Matrix Algebra:	
19	FREQUENCY	Counts how many values are in each category, given the upper limit for each category, and returns the counts in a vertical array that is one more row than there are upper limits. The extra row is an extra category that includes any values that might be greater than the last upper limit.	FREQUENCY(data_array, bins_array)
20	GROWTH	Calculates predicted exponential growth by using existing data. GROWTH returns the y-values for a series of new x-values that you specify by using existing x-values and y-values. You can also use the GROWTH worksheet function to fit an exponential curve to existing x-values and y-values.	GROWTH(known_y's, [known_x's], [new_x's], [const])
21	LINEST	Returns a set of statistics for single or multiple regression, using the least-squares method for best fitting data to a straight line.	LINEST(known_y's, [known_x's], [const], [stats])
22	LOGEST	In regression analysis, calculates an exponential curve that fits the data and returns an array of values that describes the curve.	LOGEST(known_y's, [known_x's], [const], [stats])
23	MMULT	Returns the matrix product of two arrays. Number columns of first array must equal number of rows in second array. The resultant array is the number of rows in first array by the number of columns in the second array.	MMULT(array1, array2)
24	MODE.MULT	Calculates the mode for a set of numbers, where the mode is the number that occurs most frequently. If there are multiple modes, MODE.MULT lists all modes in a vertical array.	MODE.MULT((number1,[number2],...)
25	MUNIT	Returns the unit matrix, given a single number.	MUNIT(dimension)
26	TREND	Using the least-squares method for best-fitting data to a straight line, returns an array of y-values, given these formula inputs: known y-values, known x-values, and an array of x-values used to estimate the array of y-values.	TREND(known_y's, [known_x's], [new_x's], [const])
		Text:	
27	TEXTSPLIT	Splits text by using column and/or row delimiters. 5th argument determines case sensitiveness. Use 6th argument to specify value to show when no delimiters are found.	TEXTSPLIT(text,col_delimiter,[row_delimiter], [ignore_empty], [match_mode], [pad_with])
28	FILTERXML	The FILTERXML function returns specific data from XML content by using the specified xpath.	FILTERXML(xml, xpath)

No.	LAMBDA Helper Array Function	Description	Arguments
The function argument in LAMBDA Helper Array Functions can use the LAMBDA function (to define your own calculation) or one of the 16 Eta-Lambda aggregate functions: SUM, PERCENTOF, AVERAGE, MEDIAN, COUNT, COUNTA, MAX, MIN, PRODUCT, ARRAYTOTEXT, CONCAT, STDEV.S, STDEV.P, VAR.S, VAR.P, MODE.SNGL			
29	BYCOL	Iterates a function defined by the LAMBDA function over an specified array, making a function calculation for each column and returning an array of the results.	BYCOL (array, function)
30	BYROW	Iterates a function defined by the LAMBDA function over an specified array, making a function calculation for each row and returning an array of the results.	BYROW(array, function)
31	MAKEARRAY	Returns a calculated array of a specified row and column size, by applying a function defined by the LAMBDA function.	MAKEARRAY(rows, cols, function)
32	MAP	Iterates a function defined by the LAMBDA function over an specified set of arrays (1 or more), making a function calculation for each cell in the corresponding arrays and then returning an array of the results. Last argument can contain another array to be mapped.	MAP (array1, lambda_or_array<#>)
33	SCAN	Scans an array by applying a LAMBDA to each value and returns an array that has each intermediate value. For example, it can create a spilled running total.	SCAN (initial_value, array, function)
No.	**LAMBDA Helper Non-Array Function**	**Description**	**Arguments**
	ISOMITTED	Checks whether the value in a LAMBDA is missing and returns TRUE or FALSE.	ISOMITTED(argument)
No.	**Array Functions that deliver an aggregate answer**	**Description**	**Arguments**
34	REDUCE	Reduces an array to an accumulated value by applying a LAMBDA to each value and returning the last value in the accumulator array.	REDUCE([initial_value], array, function)
35	SUMPRODUCT	SUMPRODUCT function is an aggregate function that is programmed to make array calculations, but deliver an aggregate answer.	SUMPRODUCT(array1,[array2] ...)
No.	**Reporting Array Functions**	**Description**	**Arguments**
36	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])
37	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])

Date	Student	Quarter 1 Test	Score
1/6/23	Sioux	Test 1	87
2/10/23	Sioux	Test 2	86
2/22/23	Sioux	Test 3	70
1/6/23	Chantel	Test 1	96
2/10/23	Chantel	Test 2	90
2/22/23	Chantel	Test 3	90
1/6/23	Mo	Test 1	89
2/10/23	Mo	Test 2	45
2/22/23	Mo	Test 3	93
1/6/23	Ty	Test 1	99
2/10/23	Ty	Test 2	92
2/22/23	Ty	Test 3	92

Date	Student	Quarter 2 Test	Score
3/1/23	Sioux	Test 4	87
3/15/23	Sioux	Test 5	86
4/3/23	Sioux	Test 6	70
3/1/23	Chantel	Test 4	87
3/15/23	Chantel	Test 5	90
4/3/23	Chantel	Test 6	90
3/1/23	Mo	Test 4	89
3/15/23	Mo	Test 5	90
4/3/23	Mo	Test 6	93
3/1/23	Ty	Test 4	99
3/15/23	Ty	Test 5	90
4/3/23	Ty	Test 6	92

Re-Orient Data Functions
CHOOSECOLS
CHOOSEROWS
DROP
TAKE
VSTACK
HSTACK
SORT
TOCOL
TOROW
TRANSPOSE
WRAPCOLS
WRAPROWS
SORTBY
UNIQUE
EXPAND
FILTER

VSTACK
CHOOSECOLS
SORT
TAKE

Goal:
Top Five Student Sorted Scores from both tables

Top:	6

Student	ALL Scores
Ty	99
Ty	99
Chantel	96
Mo	93
Mo	93
Ty	92

Top 4 include ties:

Top	4
Value	93

Student	ALL Scores
Ty	99
Ty	99
Chantel	96
Mo	93
Mo	93

N13: =TAKE(SORT(CHOOSECOLS(VSTACK(C3:E14,H3:J14),1,3),2,-1),O10)

C22: =CHOOSECOLS(SORT(FILTER(VSTACK(C3:E14,H3:J14),VSTACK(E3:E14,J3:J14)>=D19),3,-1),1,3)

TOROW

TRANSPOSE

Goal:

Orient test scores from second table horizontally

Tests:	Test 4, 87, Test 5, 86, Test 6, 70, Test 4, 87, Test 5, 90, Test 6, 90, Test 4, 89, Test 5, 90, Test 6, 93, Test 4, 99, Test 5, 90, Test 6, 92												
Test:	Test 4	Test 5	Test 6	Test 4	Test 5	Test 6	Test 4	Test 5	Test 6	Test 4	Test 5	Test 6	
Score:	87	86	70	87	90	90	89	90	93	99	90	92	

R7: =TEXTJOIN(", ",,TOROW(I3:J14))

R9: =TRANSPOSE(I3:J14)

WRAPCOLS

Goal:
Create Rectangle of Test Scores in Cells

Test 4 = 87	Test 4 = 87	Test 4 = 89	Test 4 = 99
Test 5 = 86	Test 5 = 90	Test 5 = 90	Test 5 = 90
Test 6 = 70	Test 6 = 90	Test 6 = 93	Test 6 = 92

AF6: =WRAPCOLS(I3:I14,3)&" = "&WRAPCOLS(J3:J14,3)

SORTBY

UNIQUE

Goal:
Extract unique list of test scores sorted by Test, then by Score

Test	Score
Test 4	99
Test 4	89
Test 4	87
Test 5	90
Test 5	86
Test 6	93
Test 6	92
Test 6	90
Test 6	70

AK9: =UNIQUE(SORTBY(I3:J14,I3:I14,1,J3:J14,-1))

AK30: =UNIQUE(SORT(I3:J14,{1,2},{1,-1}))

Test	Score
Test 4	99
Test 4	89
Test 4	87
Test 5	90
Test 5	86
Test 6	93
Test 6	92
Test 6	90
Test 6	70

FILTER

EXPAND

Goal:
Show tests after March 15, with word Last in cell to left

Score	Label
70	Last
90	Last
93	Last
92	Last

AO9: =EXPAND(FILTER(J3:J14,G3:G14>AQ2),,2,"Last")

Score	Label
70	Last
90	Last
93	Last
92	Last

AO15: =HSTACK(FILTER(J3:J14,G3:G14>AQ2),{"Last";"Last";"Last";"Last"})

Date

3/15/23

Product	Sales
Quad	439.43
Carlota	231.05
Quad	120.01
Aspen	550.51
Aspen	400.1
Carlota	397.84

New Data:

Yanaki	129.27
Carlota	388.49
Yanaki	43.69

VSTACK

SORT

UNIQUE

DROP

Goal:

Create Dynamic Product Sales Report

Product	Sales
Aspen	950.61
Carlota	628.89
Quad	559.44
Total	**2138.94**

AW10: =VSTACK(SORT(UNIQUE(ProdSales[Product])),"Total")

AX10: =VSTACK(DROP(SUMIFS(ProdSales[Sales],ProdSales[Product],AW10#),-1),SUM(ProdSales[Sales]))

Prod/SR	Joe	Sioux
Quad	12	24
Aspen	11	14
Carlota	34	22

SEQUENCE

TOCOL

EXPAND

Goal: Convert Cross Tabulated Report into Proper Data Set with Formula

Row C.	3
Column C.	2

Prod	Prod	SR	SR	Sales
Quad	Quad	Joe	Joe	12
Quad	Quad	Sioux	Sioux	24
Aspen	Aspen	Joe	Joe	11
Aspen	Aspen	Sioux	Sioux	14
Carlota	Carlota	Joe	Joe	34
Carlota	Carlota	Sioux	Sioux	22

BF13: =INDEX(BF3:BF5,ROUNDUP(SEQUENCE($BG$9*$BG$10)/BG10,0))
BG13: =TOCOL(IFNA(EXPAND(BF3:BF5,,BG10),BF3:BF5))
BH13: =INDEX(BG2:BH2,MOD(SEQUENCE($BG$9*$BG$10,,0),BG10)+1)
BI13: =TOCOL(IFNA(EXPAND(BG2:BH2,BG9),BG2:BH2))
BJ13: =TOCOL(BG3:BH5)

	Time1	Time2	Time3	Time4	Time5
	0.101	0.177	0.103	0.108	0.163
	0.106	0.103	0.112	0.174	0.105
	0.1	0.104	0.104	0.104	0.0999
	0.106	0.105	0.108	0.106	0.111
	0.134	0.131	0.117	0.106	0.1
	0.106	0.115	0.097	0.108	0.085
Ave	0.1088333	0.1225	0.106833	0.117667	0.11065

Create Arrays of Numbers

RANDARRAY

SEQUENCE

Rows	10
Columns	2
Min	1
Max	5

RANDARRAY

0 to 1	1 to 5	Rounded	Text	Col1	Col2
0.020353096	3	2.04	Yanaki	0.09443854	0.29889143
0.020060188	3	4.09	Bellen	0.25791647	0.27118213
0.452945726	2	2.56	Carlota	0.3728837	0.04736142
0.128392687	4	4.67	Yanaki	0.8431947	0.78732642
0.854260646	2	4.71	Bellen	0.54913444	0.30608632
0.548742503	2	2.76	Yanaki	0.52959386	0.1060084
0.618090179	5	2.23	Carlota	0.03146846	0.50966053
0.314188881	2	4.11	Carlota	0.08098736	0.68743436
0.726513975	4	3.04	Quad	0.29443964	0.41247996
0.034440246	3	3.29	Quad	0.08242327	0.67828537

D11: =RANDARRAY(E1)
E11: =RANDARRAY(E1,,E3,E4,1)
F11: =ROUND(RANDARRAY(E1,,E3,E4),2)
G11: =INDEX(G2:G5,RANDARRAY(E1,,1,ROWS(G2:G5),1))
H11: =RANDARRAY(E1,E2)
K11: =SEQUENCE(L1)
L11: =SEQUENCE(L1,,L3,L4)
P11: =INDEX("Sioux",SEQUENCE(ROWS(O11:O18),,1,0))
S11: =SEQUENCE(T8+1,,0)
T11: =BINOM.DIST.RANGE(T8,T7,S11#)
V11: =SEQUENCE(W8-W7+1,,W7)
W11: =NORM.DIST(V11#,W5,W6,0)
N25: =MID(N22,SEQUENCE(LEN(N22)),1)
O25: =CODE(N25#)

Products
Quad
Yanaki
Carlota
Bellen

Rows	10
Columns	2
Start	5
Step	5

SEQUENCE

Seq1	Seq5
1	5
2	10
3	15
4	20
5	25
6	30
7	35
8	40
9	45
10	50

Goal: Spill Repeated Word

Date	Sales	SalesRep
11/19/22	775.17	Sioux
11/20/22	631.59	Sioux
11/21/22	1929.39	Sioux
11/22/22	688.83	Sioux
11/23/22	167.37	Sioux
11/24/22	765.01	Sioux
11/25/22	936.83	Sioux
11/26/22	930.98	Sioux

Goal: Investigate characters in text string:

The dog is rad

Characters:	Code:
T	84
h	104
e	101
	32
d	100
o	111
g	103
	160
i	105
s	115
	32
r	114
a	97
d	100

Insurance Agent has 6 appointments.
It is known from past data that she makes a sale 20.0% of the time.

What is probability that she will make 3 sales?

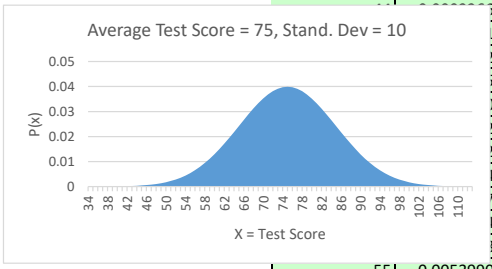
Binomial Distribution

P	0.2
n	6

X	P(x)
0	0.262144
1	0.393216
2	0.24576
3	0.08192
4	0.01536
5	0.001536
6	6.4E-05

Average Test Sc	75
Stand. Dev	10
Start	34
End	114

X = Test Score	P(x)
34	8.92617E-06
35	1.3383E-05
36	1.98655E-05
37	2.91947E-05
38	4.2478E-05
39	6.11902E-05
40	8.72683E-05
41	0.000123222
42	0.000172257
43	0.000238409
55	0.005399097
56	0.006561581
57	0.007895016
58	0.009404908
59	0.011092083
60	0.01295176
61	0.014972747



Statistics & Matrix Algebra
FREQUENCY
GROWTH
LINEST
LOGEST
MMULT
MODE.MULT
MUNIT
TREND

Cola	Sales ($)
Coke	250
Pepsi	504
Bloxy	331
Bloxy	1047
c	275
RC	1005
RC	616
Bloxy	1043
RC	1054
RC	275
RC	271
c	604
RC	923
Bloxy	613
Pepsi	474
Pepsi	923
RC	678
Pepsi	169

FREQUENCY

Upper Limit for Sales ($)	Frequency
250	2
450	4
750	6
1050	5
	1

Categories for Upper Limits
Sales <= 250
250 < Sales <=450
450 < Sales <=750
750 < Sales <=1050
Sales >1050

MODE.MULT

Mode = Number That Occurs Most Frequently

Modes:
275
923

H4: =FREQUENCY(E4:E21,G4:G7)
H16: =MODE.MULT(E4:E21)

Sales Calls Made (X)	Sales (Y)
156	96,408
39	22,737
121	95,832
68	35,496
30	20,190
48	37,968
157	140,358
103	74,572
74	43,734
87	60,030
165	112,200
63	58,212
148	141,192
64	36,416
51	40,851
68	39,916

TREND Linear Regression to estimate based on X & Y.

Day	Call Estimates (X)	Sales Estimates (Y)	Sales Estimates (Y)
Day 1	52	34,657	34,657
Day 2	67	46,992	46,992
Day 3	133	101,262	101,262
Day 4	160	123,464	123,464
Day 5	111	83,172	83,172
Day 6	26	13,278	13,278
Day 7	136	103,729	103,729
Day 8	126	95,506	95,506
Day 9	159	122,642	122,642
Day 10	70	49,459	49,459
Day 11	163	125,931	125,931

Slope
822.285112
Y-intercept
-8101.4457

Q4: =TREND(M4:M19,L4:L19,P4:P14)
R4: =P4:P14*P23+Q23

LINEST
Calculates Linear Regression Statistics.

Slope	Y-intercept
822.2851123	-8101.445746

P23: =LINEST(M4:M19,L4:L19)
P30: =LINEST(M4:M19,L4:L19,1,1)

LINEST

Slope m	822.2851123	-8101.445746	Intercept b
Standard Error m	78.06068356	7831.591503	Standard Error b
Coefficient of Determination	0.887967288	13763.44004	Standard Error y
F	110.963502	14	df
SS Regression	21020069392	2652051946	SS Residual

const: = 1 = TRUE = b calculated normally. 0 = FALSE = b set to zero.
stats: 1 = TRUE = calculate all statistics. 0 = FALSE = calculate just m and b.

TEXTSPLIT
TEXTSPLIT
FILTERXML

Text	TEXTSPLIT - row delimiter	FILTERXML
2,23,43,6,90,690	2	2
	23	23
	43	43
	6	6
	90	90
	690	690

Text	TEXTSPLIT - row & column delimiter					
2,23;43,6;90,690	2	23				
	43	6				
	90	690				

Text	TEXTSPLIT - 2 delimters for row					
2,23;43,6;90,690	2	23	43	6	90	690

E3: =TEXTSPLIT(D3,,",")

F3: =FILTERXML("<AllText><Num>"&SUBSTITUTE(D3,",","</Num><Num>")&"</Num></AllText>","//Num")

E11: =TEXTSPLIT(D11,",",";")

E16: =TEXTSPLIT(D16,{",",";"})