

# Lesson 10

Sébastien Mathier

[www.excel-pratique.com/en](http://www.excel-pratique.com/en)

Arrays are "variables" that allow many values to be stored. We have already covered this topic in Lesson 3, but now we will go into greater depth ...

Why use arrays? :

Imagine that you are trying to write a procedure in which you will need to store 500 values. If you had to create 500 individual variables to do this, it would be extremely difficult. With an array, storing and working with these values will be much easier.

A second reason to use arrays is their "speed". It takes much less time to retrieve data from arrays than from "tables" (made up of cells) on an Excel worksheet ...

So here is an example that will make this all a little clearer ...

On the first worksheet ("DS") you will find a 5000 row by 3 column data set :

1	Date	Client number	Payment
2	03.11.2026	24	YES
3	10.23.2018	30	YES
4	03.21.2021	6	YES
5	03.08.2023	24	YES
6	04.05.2016	29	YES
7	03.22.2018	10	YES
8	07.27.2021	30	YES
9	09.25.2025	5	YES
10	02.04.2016	23	NO
11	09.11.2019	20	NO

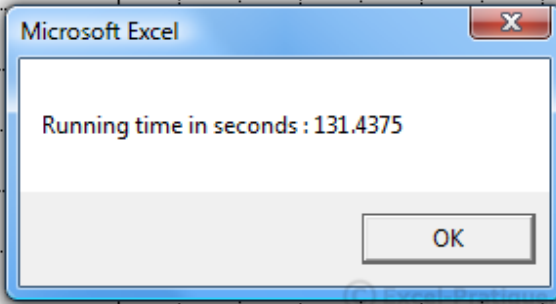
On the second worksheet you will find a summary table which accounts for all the "YES" responses by year and by client :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Client no.																															
2	Years	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
3	2011																															
4	2012																															
5	2013																															
6	2014																															
7	2015																															
8	2016																															
9	2017																															
10	2018																															
11	2019																															
12	2020																															
13	2021																															
14	2022																															
15	2023																															
16	2024																															
17	2025																															
18	2026																															
19	Calculate :																															
20																																
21																																
22																																
23																																

In this case, the procedure will use a loop to process the data set and record the number of "YES" responses for each year and each client number, then enter this data into the corresponding cells.

Without using arrays, it would take Excel **131.44 seconds** to execute this procedure :

	A	B	C	D	E	F	G	H	I
1	Client no.								
1	Years	1	2	3	4	5	6	7	8
2	2011	11	5	9	10	4	8	11	10
3	2012	9	6	6	9	10	9	12	9
4	2013	8	10	7	8	15	5	6	11
5	2014	9	8	8	5	7	8	7	9
6	2015	7	12	12	8	9	8	10	3
7	2016	15	9	12	7	11	9	5	12
8	2017	5	5	12	7	8	10	9	13
9	2018	4	8	3	5	7	8	13	7



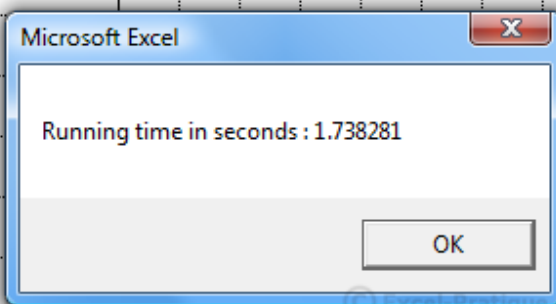
Microsoft Excel

Running time in seconds : 131.4375

OK

But by first storing the data (from worksheet "DS") in an array and then carrying out the same calculations (using the array instead of the data set from worksheet "DS"), it will only take **1.74 seconds** for the procedure to execute :

6	2015	7	12	12	8	9	8	10	3
7	2016	15	9	12	7	11	9	5	12
8	2017	5	5	12	7	8	10	9	13
9	2018	4	8	3	5	7	8	13	7



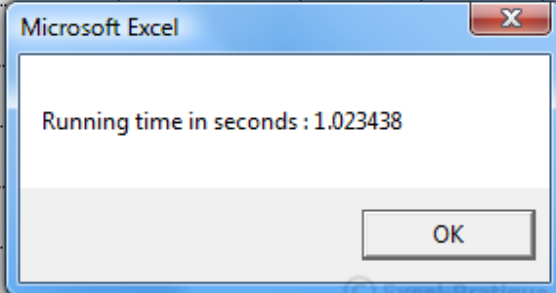
Microsoft Excel

Running time in seconds : 1.738281

OK

And if we decided to optimize the procedure by storing only the data containing "YES" responses in the array (which is about 3/4 of the data), it would take only **1.02 seconds** :

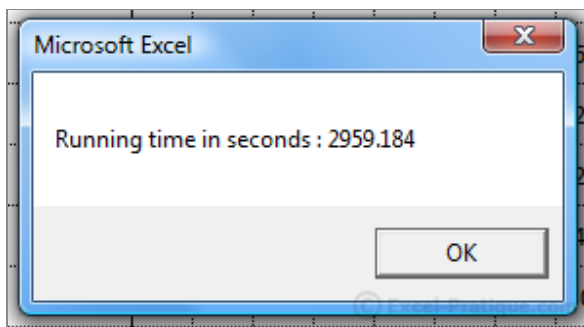
6	2015	7	12	12	8	9	8	10	3
7	2016	15	9	12	7	11	9	5	12
8	2017	5	5	12	7	8	10	9	13
9	2018	4	8	3	5	7	8	13	7
10									
11									
12									
13									
14									



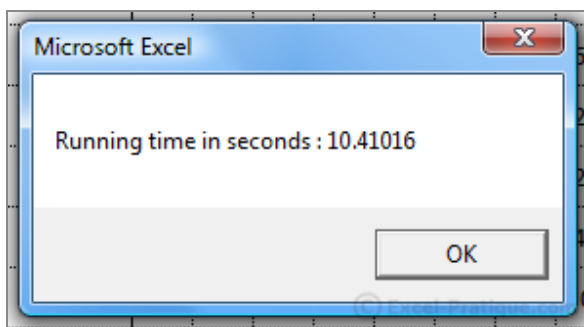
This is a good example of how using an array makes it possible to execute a procedure about **128x faster**. This improvement would be even greater if we were working with multiple data sets at once.

Here is another example in which a procedure uses two data sets, effectively the equivalent of a "long loop" executed 78,240,000 times.

Without using arrays, it would take **more than 49 minutes** for Excel to complete the execution of this immense procedure :



Simply by using an array for the data set (and without any other modifications), it hardly takes more than **10 seconds** to execute the same procedure :



### Declaring an array :

Here are a few examples of array declarations (if the first 2 don't make sense to you, reread Lesson 3) :

```
'Sample declaration of a 1 dimensional array
Dim array1(4)

'Sample declaration of a 2 dimensional array
Dim array2(6, 1)

'Sample declaration of a dynamic array
Dim array3()
```

If you can't enter fixed values when you declare an array (because they will depend on the size of the data set, for example), leave the parentheses empty.

You don't have to declare a type (string, long, etc.), although in many cases this will slow down the execution of your procedure ...

### Storing data in an array :

Let's begin by storing some data in an array :

	A
1	Date
2	03.11.2026
3	10.23.2018
4	03.21.2021
5	03.08.2023
6	04.05.2016
7	03.22.2018
8	07.27.2021
9	09.25.2025
10	02.04.2016
11	09.11.2019
12	01.25.2020
13	

We want to store 11 x 1 values in this case, so we need to create a 1 dimensional array :

```
'Declaration
Dim array_example(10)
```

Don't forget that **array element numbering begins with 0** (this is standard in programming, so it's a good idea to get in the habit of working this way, even though it is actually possible to change this convention in VBA).

Each element in the array will now receive its value :

```
'Storing values in the array
array_example(0) = Range("A2")
array_example(1) = Range("A3")
array_example(2) = Range("A4")
array_example(3) = Range("A5")
array_example(4) = Range("A6")
array_example(5) = Range("A7")
array_example(6) = Range("A8")
array_example(7) = Range("A9")
array_example(8) = Range("A10")
array_example(9) = Range("A11")
array_example(10) = Range("A12")
```

You can work with or modify each element of the array as though it were a variable.

Here is an example in which we use **array\_example(8)** :

```
Sub example()
  'Declaration
  Dim array_example(10)

  'Storing values in the array
  array_example(0) = Range("A2")
  array_example(1) = Range("A3")
  array_example(2) = Range("A4")
  array_example(3) = Range("A5")
  array_example(4) = Range("A6")
  array_example(5) = Range("A7")
  array_example(6) = Range("A8")
  array_example(7) = Range("A9")
  array_example(8) = Range("A10")
  array_example(9) = Range("A11")
  array_example(10) = Range("A12")

  'Test 1
  MsgBox array_example(8) '=> returns : 02.04.2016

  'Changing one of the values
  array_example(8) = Year(array_example(8))

  'Test 2
  MsgBox array_example(8) '=> returns : 2016
End Sub
```

A **For** loop would be an effective way to stock the array faster :

```
'Declaration
Dim array_example(10)

'Storing values in the array
For i = 0 To 10
  array_example(i) = Range("A" & i + 2)
Next
```

The two dimensional array :

To store more than one column of data, we will need another dimension in our array. Here is an example :

	A	B	C
1	Date	Client number	Payment
2	03.11.2026	24	YES
3	10.23.2018	30	YES
4	03.21.2021	6	YES
5	03.08.2023	24	YES
6	04.05.2016	29	YES
7	03.22.2018	10	YES
8	07.27.2021	30	YES
9	09.25.2025	5	YES
10	02.04.2016	23	NO
11	09.11.2019	20	NO
12	01.25.2020	28	YES
13			

Storing data in a 2 dimensional array :

```
'Declaration
Dim array_example(10, 2) '11 x 3 "element" array

'Storing values in the array
For i = 0 To 10
    array_example(i, 0) = Range("A" & i + 2)
    array_example(i, 1) = Range("B" & i + 2)
    array_example(i, 2) = Range("C" & i + 2)
Next
```

And here are a few examples of working with these values :

```
MsgBox array_example(0, 0) '=> returns : 03.11.2026
MsgBox array_example(0, 1) '=> returns : 24
MsgBox array_example(9, 2) '=> returns : NO
MsgBox array_example(10, 2) '=> returns : YES
```

## The dynamic array :

Just imagine for a moment that this same data set was going to be updated regularly and therefore we couldn't set fixed values when we declare it ...

	A	B	C
1	Date	Client number	Payment
2	03.11.2026	24	YES
3	10.23.2018	30	YES
4	03.21.2021	6	YES
5	03.08.2023	24	YES
6	04.05.2016	29	YES
7	03.22.2018	10	YES
8	07.27.2021	30	YES
9	09.25.2025	5	YES
10	02.04.2016	23	NO
11	09.11.2019	20	NO

To find out the row number of the last cell in a series of non-empty cells, or in other words, the last row in our database, we'll use the following formula :

```
last_row = Range("A1").End(xlDown).Row
```

Excel does not accept variables in declarations.

Instead, declare a dynamic array (using empty parentheses), then define its dimensions using **ReDim** :

```
Dim array_example()  
ReDim array_example(last_row - 2, 2)
```

Using the following procedure, you can store all the rows in the data set in your array :

```
Sub example()  
    last_row = Range("A1").End(xlDown).Row 'Last row of the data set  
  
    Dim array_example()  
    ReDim array_example(last_row - 2, 2)  
  
    'Storing values in the array  
    For i = 0 To last_row - 2  
        array_example(i, 0) = Range("A" & i + 2)  
        array_example(i, 1) = Range("B" & i + 2)  
        array_example(i, 2) = Range("C" & i + 2)  
    Next  
End Sub
```



Ubound :

In the preceding example, the last number in our array was **last\_row - 2** :

```
For i = 0 To last_row - 2
```

Another way to determine the last number in the array would be to use **Ubound** :

```
For i = 0 To UBound(array_example)
```

This function returns the highest number in the array for the chosen dimension (the first dimension is the default).

Here are a few examples that will make this clearer :

```
Sub example()  
    Dim array_example(10, 2)  
  
    MsgBox UBound(array_example) '=> returns : 10  
    MsgBox UBound(array_example, 1) '=> returns : 10  
    MsgBox UBound(array_example, 2) '=> returns : 2  
End Sub
```

## Storing data in a range of array elements :

It's possible to populate an array with the data from a range of worksheet cells without even using a loop.

```
'Declaration
Dim array_example(10, 2) '11 x 3 "element" array

'Storing values in the array
For i = 0 To 10
    array_example(i, 0) = Range("A" & i + 2)
    array_example(i, 1) = Range("B" & i + 2)
    array_example(i, 2) = Range("C" & i + 2)
Next
```

The preceding code can effectively be replaced with :

```
'Declaration
Dim array_example()

'Storing values in the array
array_example = Range("A2:C12").Value
```

But if this second method seems attractive at first, be warned that in many cases it can cost you more time than the first one ...

If you store data in your array in this way, the first number will be 1 rather than 0, which can cause confusion ... Further along in the development process, if you decide to save only data that correspond to certain search criteria in your array (or to carry out an entirely different operation), you will have to entirely rewrite the code using another loop function ...

But this second method is quite useful if you need to store the entire contents of a large data set, because it's faster than a loop (saving about 0.2 seconds for every 15,000 entries).

Array :

But if you need to create an array that has "fixed" contents.

One solution would be to set the contents line by line :

```
Dim en(5)

en(0) = "IF"
en(1) = "VLOOKUP"
en(2) = "SUM"
en(3) = "COUNT"
en(4) = "ISNUMBER"
en(5) = "MID"
```

Luckily, you can simplify this code by using **Array** :

```
en = Array("IF", "VLOOKUP", "SUM", "COUNT", "ISNUMBER", "MID")
```

Here is a demonstration of the use of the **Replace** function (this will help you understand the example that follows) :

```
Sub replace_example()
    Dim var_translate As String

    'A string for this example
    var_translate = "Hello World !"

    'Replacement of "World" with "you" in the character string
    var_translate = Replace(var_translate, "World", "you")

    'The string after replacement
    MsgBox var_translate '> returns "Hello you !"
End Sub
```

Now if we want to replace a series of values with another series, using arrays and the (**Array**) function will be extremely helpful :

```
Sub translate() 'Simplified example of EN-FR translation for formulas
    Dim var_translate As String

    'A string for this example
    var_translate = "Formula to translate : SUM(IF(ISNUMBER(A1:E1),A1:E1,0))"

    'The two series of values
    en = Array("IF", "VLOOKUP", "SUM", "COUNT", "ISNUMBER", "MID")
    fr = Array("SI", "RECHERCHEV", "SOMME", "NB", "ESTNUM", "STXT")

    'Replacing "SI" with "IF", and "RECHERVEV" with "VLOOKUP", etc.
    For i = 0 To UBound(en)
        var_translate = Replace(var_translate, en(i), fr(i))
    Next

    'The string after the replacements
    MsgBox var_translate '> returns "Formula to translate : SOMME(SI(ESTNUM(A1:E1),A1:E1,0))"
End Sub
```

## Split :

The **Split** function allows us to convert a character string into an array.

To convert the string into an array, do the following :

```
variable = "IF/VLOOKUP/SUM/COUNT/ISNUMBER/MID"
```

Use the **Split** function and define the separator :

```
en = Split(variable, "/")
```

The array **en** will return the following values :

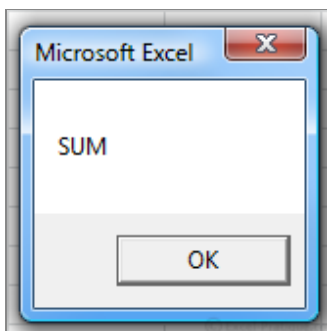
```
MsgBox en(0) '=> returns : IF  
MsgBox en(1) '=> returns : VLOOKUP  
MsgBox en(2) '=> returns : SUM  
MsgBox en(3) '=> returns : COUNT  
MsgBox en(4) '=> returns : ISNUMBER  
MsgBox en(5) '=> returns : MID
```

The following 3 arrays will also return the same values :

```
en = Array("IF", "VLOOKUP", "SUM", "COUNT", "ISNUMBER", "MID")  
en = Split("IF,VLOOKUP,SUM,COUNT,ISNUMBER,MID", ",")  
en = Split("IF VLOOKUP SUM COUNT ISNUMBER MID", " ")
```

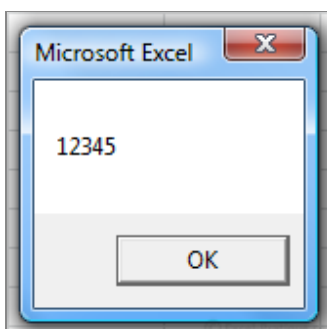
The following example returns the 3<sup>rd</sup> value in the string :

```
MsgBox Split("IF,VLOOKUP,SUM,COUNT,ISNUMBER,MID", ",")(2) '=> returns : SUM
```



The opposite of the **Split** function is **Join**. This function assembles the values of an array into a string.

```
MsgBox Join(Array(1, 2, 3, 4, 5), "") '=> returns : 12345
```



Exercise :

To practice using arrays, you will create your own version of the macro that we used to demonstrate the speed advantages of arrays, step by step ...

Here is the starting point of this exercise (you will see that the data set has been reduced to 1000 rows) :

Source file : **arrays\_exercise.xls**

	A	B	C
1	Date	Client number	Payment
2	03.11.2026	24	YES
3	10.23.2018	30	YES
4	03.21.2021	6	YES
5	03.08.2023	24	YES
6	04.05.2016	29	YES
7	03.22.2018	10	YES
8	07.27.2021	30	YES
9	09.25.2025	5	YES
10	02.04.2016	23	NO
11	09.11.2019	20	NO
12	01.25.2020	28	YES

Goal of the exercise : the procedure should process the data in the data set using a loop and count the number of "YES" or "NO" responses for each year and for each client number (either "YES" or "NO", depending on user selection) and enter this count in a specified cell on the worksheet.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF				
1	Client no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30					
2	2011																																			
3	2012																																			
4	2013																																			
5	2014																																			
6	2015																																			
7	2016																																			
8	2017																																			
9	2018																																			
10	2019																																			
11	2020																																			
12	2021																																			
13	2022																																			
14	2023																																			
15	2024																																			
16	2025																																			
17	2026																																			
18																																				
19	Calculate :	<input checked="" type="radio"/> YES number (paid)	<i>(by year and by client number)</i>																																	
20		<input type="radio"/> NO number (unpaid)																																		
21																																				
22		<input type="button" value="Update"/>																																		
23																																				

Complete the following macro to store the data set from worksheet "DS" in an array :

```
Sub actualize()  
    Dim last_row As Integer  
  
    'Last row of the data set  
    '...  
  
    'Storing the data set in a dynamic array  
    Dim array_db()  
    '...  
  
End Sub
```

.....



.  
. .  
.

Here is one solution :

```
Sub actualize()  
  Dim last_row As Integer, search_value As String, insert_row As Integer, value_yes_no As  
  String, rows_number As Integer  
  
  'Last row of the data set  
  last_row = Sheets("DS").Range("A1").End(xlDown).Row  
  
  'Search value (YES or NO)  
  If Sheets("RES").OptionButton_yes.Value = True Then  
    search_value = "YES"  
  Else  
    search_value = "NO"  
  End If  
  
  'Number of YES or NO responses  
  rows_number = WorksheetFunction.CountIf(Sheets("DS").Range("C2:C" & last_row),  
  search_value)  
  
  'Storing the data set in the array  
  Dim array_db()  
  ReDim array_db(rows_number - 1, 1)  
  
  insert_row = 0  
  
  For row_number = 2 To last_row  
    value_yes_no = Sheets("DS").Range("C" & row_number)  
    If value_yes_no = search_value Then  
      array_db(insert_row, 0) = Sheets("DS").Range("A" & row_number)  
      array_db(insert_row, 1) = Sheets("DS").Range("B" & row_number)  
      insert_row = insert_row + 1  
    End If  
  Next  
End Sub
```

The user's search choice is determined at the beginning of the procedure by the following code :

```
'Search value (YES or NO)  
If Sheets("RES").OptionButton_yes.Value = True Then  
  search_value = "YES"  
Else  
  search_value = "NO"  
End If
```

We are using the CountIF function to determine the number of YES or NO responses :

```
'Number of YES or NO responses  
rows_number = WorksheetFunction.CountIf(Sheets("DS").Range("C2:C" & last_row), search_value)
```

The size of the array has been adjusted to fit the number of YES or NO responses and reduced to two columns :

```
ReDim array_db(rows_number - 1, 1)
```



The data will now be stored in the array when its 3<sup>rd</sup> column corresponds to the user's search choice :

```
'Insertion number in the array
insert_row = 0

'Processing the data set
For row_number = 2 To last_row
    'Value of column C (YES or NO)
    value_yes_no = Sheets("DS").Range("C" & row_number)
    'If the value corresponds to the user's search choice, the row is stored in the array
    If value_yes_no = search_value Then
        'Storing the value of column A
        array_db(insert_row, 0) = Sheets("DS").Range("A" & row_number)
        'Storing the value of column B
        array_db(insert_row, 1) = Sheets("DS").Range("B" & row_number)
        'One row has been stored => the insertion number in the array is incremented by 1
        insert_row = insert_row + 1
    End If
Next
```

The array contains only the data that we are interested in.

All that we have left to do is to :

- Process each element of the table on the "RES" worksheet using 2 loops (this is the same idea as the checkerboard exercise)
- And insert the total number of entries on this worksheet for a year by client number for each cell

.....

.  
. .  
.

Here is a solution :

```
'Count of "YES"/"NO" responses
For no_years = 2011 To 2026
  For no_client = 1 To 30
    counter = 0
    For i = 0 To UBound(array_db)
      If Year(array_db(i, 0)) = no_years And array_db(i, 1) = no_client Then
        counter = counter + 1
      End If
    Next
    Cells(no_years - 2009, no_client + 1) = counter
  Next
Next
```

The solution with commentaries that explain it in detail :

```
'A loop for each row
For no_years = 2011 To 2026
  'A loop for each column
  For no_client = 1 To 30
    'Counter re-initialized
    counter = 0
    'Processing the array
    For i = 0 To UBound(array_db)
      'Verify that the row in the table match the year and client number
      If Year(array_db(i, 0)) = no_years And array_db(i, 1) = no_client Then
        'If year and client number match, the counter is incremented by 1
        counter = counter + 1
      End If
    Next
    'After processing the array, the total is entered in the appropriate cell
    Cells(no_years - 2009, no_client + 1) = counter
  Next
Next
```

And finally, the code of the entire macro :

```
Sub actualize()  
    Dim last_row As Integer, search_value As String, insert_row As Integer, value_yes_no As  
String, rows_number As Integer, counter As Integer  
  
    'Deleting contents  
    Range("B2:AE17").ClearContents  
  
    'Last row in the data set  
    last_row = Sheets("DS").Range("A1").End(xlDown).Row  
  
    'Search value (YES ou NO)  
    If Sheets("RES").OptionButton_yes.Value = True Then  
        search_value = "YES"  
    Else  
        search_value = "NO"  
    End If  
  
    'Number of YES or NO responses  
    rows_number = WorksheetFunction.CountIf(Sheets("DS").Range("C2:C" & last_row),  
search_value)  
  
    'Storing the data set in the array  
    Dim array_db()  
    ReDim array_db(rows_number - 1, 1)  
  
    insert_row = 0  
  
    For row_number = 2 To last_row  
        value_yes_no = Sheets("DS").Range("C" & row_number)  
        If value_yes_no = search_value Then  
            array_db(insert_row, 0) = Sheets("DS").Range("A" & row_number)  
            array_db(insert_row, 1) = Sheets("DS").Range("B" & row_number)  
            insert_row = insert_row + 1  
        End If  
    Next  
  
    'Count of YES or NO responses  
    For no_years = 2011 To 2026  
        For no_client = 1 To 30  
            counter = 0  
            For i = 0 To UBound(array_db)  
                If Year(array_db(i, 0)) = no_years And array_db(i, 1) = no_client Then  
                    counter = counter + 1  
                End If  
            Next  
            Cells(no_years - 2009, no_client + 1) = counter  
        Next  
    Next  
End Sub
```

Source file : [arrays\\_exercise\\_completed.xls](#)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF		
1	Client no.																																	
2	Years	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30			
3	2011	4	0	1	3	0	2	2	1	2	0	0	4	0	2	1	1	1	1	2	4	2	3	1	0	2	1	1	2	1	4			
4	2012	1	2	1	0	3	1	2	2	1	1	0	1	2	2	0	0	2	1	2	3	0	5	2	4	1	4	1	2	1	3			
5	2013	2	1	1	0	4	1	1	4	2	1	5	0	5	1	1	3	1	3	2	0	2	2	0	0	1	0	3	2	1	0			
6	2014	1	2	3	1	3	0	1	5	1	1	2	3	5	2	1	0	3	1	2	4	0	2	0	2	0	1	3	0	2	1			
7	2015	1	1	1	4	2	3	1	0	2	3	0	2	1	3	1	3	2	1	0	1	2	0	2	0	1	1	1	3	2	1			
8	2016	2	1	2	3	3	2	0	4	1	1	1	0	2	3	3	1	0	0	2	1	2	1	3	0	4	2	0	1	3	0			
9	2017	2	0	1	1	4	1	3	3	0	3	2	3	2	3	0	1	3	3	0	2	2	1	3	0	3	4	2	3	0	2			
10	2018	1	1	0	0	2	2	4	3	2	1	4	1	1	1	2	3	2	2	2	0	2	1	2	1	0	1	1	2	3	1			
11	2019	2	1	4	3	3	0	1	2	2	3	1	2	1	1	0	2	1	4	1	1	2	1	2	1	3	0	0	1	1	2			
12	2020	2	0	4	5	1	0	0	0	1	1	3	2	2	1	4	1	1	4	4	3	1	3	7	1	3	1	3	3	3	4			
13	2021	1	3	0	0	2	2	3	1	2	1	3	1	3	1	4	0	2	0	2	1	2	1	1	0	4	1	1	0	4	2			
14	2022	2	2	1	2	2	2	1	2	2	2	1	5	1	2	3	0	1	2	2	2	3	5	3	2	1	0	2	0	0	0			
15	2023	1	1	1	2	1	3	3	0	0	0	2	4	1	2	2	0	2	1	2	1	0	6	1	4	0	2	2	0	4	3			
16	2024	3	3	2	1	3	2	0	4	4	0	0	1	1	4	1	0	3	1	2	1	3	4	2	4	1	2	4	0	2	2			
17	2025	2	0	1	2	3	4	1	1	5	1	1	1	3	0	0	1	3	2	0	2	3	3	1	1	0	0	2	2	0	0			
18	2026	0	0	1	0	3	0	0	1	0	2	3	1	0	2	2	0	0	1	0	4	0	0	1	1	0	0	1	0	3	1			
19	Calculate :	<input checked="" type="radio"/> YES number (paid) <span style="margin-left: 20px;">(By year and by client number)</span> <input type="radio"/> NO number (unpaid)																																
20																																		
21																																		
22																																		
23																																		

