IDENTIFYING UNIQUE VALUES IN AN ARRAY OR RANGE (VBA)

Have you ever had to work with just the unique items in a range? If your data is in the form of a database, you can use the **Advanced Filter** command to extract the unique items from a single column. But if your data spans multiple columns, **Advanced Filter** won't work. And the **Advanced Filter** won't do you any good if your data is in a **VBA** array.

In this document I present a **VBA** function that accepts either a worksheet range object or a **VBA** array. The function returns either:

- A variant array that consists of just the unique elements in the input array or range (or)
- A **single value**: the number of unique elements in the input array or range.

Here's the syntax for the **UniqueItems** function (which is listed at the end of this document):

```
UniqueItems(ArrayIn, Count)
```

- **ArrayIn:** A range object, or an array
- **Count:** (Optional) If **True** or omitted, the function returns a single value – the number of unique items in **ArrayIn**. If False, the function returns an array that consists of the unique items in **ArrayIn**.

EXAMPLE 1

The subroutine below demonstrates **UniqueItems**. The routine generates ***100 random integers*** and stores them in an array. This array is then passed to the **UniqueItems** function and a message box displays the number of unique integers in the array.

The number will vary each time you run the subroutine.

```
Sub Test1()

 Dim z(1 To 100)

 For i = 1 To 100

 z(i) = Int(Rnd() * 100)

 Next i
```

```
 MsgBox UniqueItems(z, True)

End Sub
```

EXAMPLE 2

The subroutine below counts the number of common elements in two worksheet ranges. It creates two arrays. ***Array1*** consists of the unique items in **A1:A16**; ***Array2*** consists of the unique items in **B1:B16**. A nested loop counts the number of items that are in both ranges.

```
Sub Test2()

 Set Range1 = Sheets("Sheet1").Range("A1:A16")

 Set Range2 = Sheets("Sheet1").Range("B1:B16")

 Array1 = UniqueItems(Range1, False)

 Array2 = UniqueItems(Range2, False)

 CommonCount = 0

 For i = LBound(Array1) To UBound(Array1)

 For j = LBound(Array2) To UBound(Array2)

 If Array1(i) = Array2(j) Then _

 CommonCount = CommonCount + 1

 Next j

 Next i

 MsgBox CommonCount

End Sub
```

EXAMPLE 3

The **UniqueItems** function can also be used in worksheet formulas. The formula below returns the number of unique items in a range:

```
=UniqueItems(A1:D21)
```

EXAMPLE 4

To display the unique items in a range, you must array-enter the formula into a range of cells (use **Ctrl+Shift+Enter**). The result of the **UniqueItems** function is a horizontal array. If you would like to display the unique values in a column, you can use the **TRANSPOSE** function. The formula below (which is array-entered into a vertical range) returns the unique items in**A1:D21**.

```
=TRANSPOSE(UniqueItems(A1:D21,FALSE))
```

THE CODE

```
Option Base 1

Function UniqueItems(ArrayIn, Optional Count As Variant) As Variant

' Accepts an array or range as input ' If Count = True or is missing, the function
returns the number of unique elements ' If Count = False, the function returns a variant
array of unique elements

 Dim Unique() As Variant ' array that holds the unique items

 Dim Element As Variant

 Dim i As Integer

 Dim FoundMatch As Boolean

' If 2nd argument is missing, assign default value

 If IsMissing(Count) Then Count = True
```

```
' Counter for number of unique elements

 NumUnique = 0

' Loop thru the input array

 For Each Element In ArrayIn

 FoundMatch = False

' Has item been added yet?

 For i = 1 To NumUnique

 If Element = Unique(i) Then

 FoundMatch = True

 Exit For '(exit loop)

 End If

 Next i

AddItem:

' If not in list, add the item to unique list

 If Not FoundMatch And Not IsEmpty(Element) Then

 NumUnique = NumUnique + 1

 ReDim Preserve Unique(NumUnique)

 Unique(NumUnique) = Element

 End If
```

```
 Next Element

' Assign a value to the function

 If Count Then UniqueItems = NumUnique Else UniqueItems = Unique

End Function
```