Working with Microsoft Excel using powershell

[Leave a Comment](https://www.myfaqbase.com/b/working-with-microsoft-excel-using-powershell-part-1/#respond) / [Office VBA](https://www.myfaqbase.com/b/category/all/software/microsoft-office/office-vba/), [Powershell](https://www.myfaqbase.com/b/category/all/scripting/powershell/) / 2021-03-15 / [Microsoft Excel](https://www.myfaqbase.com/b/tag/microsoft-excel/), [Office VBA](https://www.myfaqbase.com/b/tag/office-vba/), [Powershell](https://www.myfaqbase.com/b/tag/powershell/)

Task #1 – Open Excel, select the 1st worksheet, set AutoFilter and then filter by specific column that equals a particular string

$Filename = "c:\temp\mysheet.xlsx"

$objExcel = New-Object -ComObject Excel.Application

$objExcel.DisplayAlerts = $False

$objWrkBk = $objExcel.Workbooks.Open($Filename)

$objExcel.Visible = $True

# one way of selecting a sheet e.g. 3rd worksheet

# $objWrkBk.Sheets(3).Select()

# another way of selecting a sheet e.g. 1st worksheet

# $objWrkBk.Worksheets.item(1).Select()

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

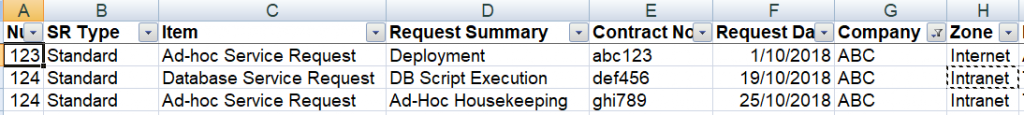
$objWrkSht1.Select()

# To filter on 7th column, with string matching 'ABC'

$objWrkSht1.Range('$A$nn:$P$mm').AutoFilter(7,'=ABC')

# Note: $A$nn => range first cell position, $P$mm = range last cell position

Output:



 Task #2 – Continuing with the same spreadsheet, read data from particular cell. In this case column 7, row 1

Code:

$objWrkSht1.Cells.Item(7,1).Value2

Output:

Company

 Task #3 – Read data from a cell range, in this case G2:G4

Code:

$objWrkSht1.Range("G2:G4").Value2

Output:

ABC

ABC

ABC

 Task #4 – Read data from a cell range into an array

Code:

#read into an array

$companies = @($objWrkSht1.Range("G2:G21").Value2)

# iterate on all array items, method 1

for ($i=0;$i -lt $companies.Length; $i++)

{

$companies[$i]

}

# iterate on all array items, method 2

foreach ($coy in $companies) { $coy }

Output:

CA04

CA04

CA03

CY02

CY02

CY03

RA01

RA01

ABC01

ABC01

DEF02

DEF02

DEF02

DEF02

DEF02

VA02

VA02

VA02

VA02

VA02

 Task #5 – Select all used cells in a worksheet

Code:

# Code

# select all used cells

$objExcel.Cells.Select()

Result:

A screenshot of a computer

Description automatically generated

 Task 6# – Sorting on 2 columns with headers. In our example sort on columns H & G with 1st row as column headers and column G sorted in ascending order, column H sorted in descending order

Code:

$xlSortOnValues = $xlSortNormal = 0

$xlTopToBottom = $xlSummaryBelow = 1

$xlAscending = 1

$xlDescending = 2

$xlNo = 2

$xlYes = 1

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

$objWrkSht1.Select()

$objRange = $objWrkSht1.UsedRange

$objRange1 = $objWrkSht1.range("G1")

$objRange2 = $objWrkSht1.range("H1")

$objWrkSht1.Sort.SortFields.Clear()

[void] $objWrkSht1.Sort.SortFields.Add($objRange1,$xlSortOnValues,$xlAscending,$xlSortNormal)

[void] $objWrkSht1.Sort.SortFields.Add($objRange2,$xlSortOnValues,$xlDescending,$xlSortNormal)

$objWrkSht1.sort.setRange($objRange) # define the range to sort

$objWrkSht1.sort.header = $xlYes # range has a header

$objWrkSht1.sort.orientation = $xlTopToBottom

$objWrkSht1.sort.apply()

Before Sort:

A screenshot of a computer

Description automatically generated

Output After Sort

A screenshot of a computer

Description automatically generated

 Task 7# – Finding the last row and last column of the worksheet that is in use

Code:

$Filename = "c:\temp\mysheet.xlsx"

$objExcel = New-Object -ComObject Excel.Application

$objExcel.DisplayAlerts = $False

$objWrkBk = $objExcel.Workbooks.Open($Filename)

$objExcel.Visible = $True

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

$objWrkSht1.Select()

$objRange = $objWrkSht1.UsedRange

$xlCellTypeLastCell = 11

$objLastcell = $objRange.SpecialCells($xlCellTypeLastCell)

$lastrow = $objLastcell.row

$lastcolumn = $objLastcell.column

$lastrow,$lastcolumn

Output:

1175

11

Task 8# – Set autofilter on a column by multiple criteria

Code:

$Filename = "c:\temp\mysheet.xlsx"

$objExcel = New-Object -ComObject Excel.Application

$objExcel.DisplayAlerts = $False

$objWrkBk = $objExcel.Workbooks.Open($Filename)

$objExcel.Visible = $True

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

$objWrkSht1.Select()

$objRange = $objWrkSht1.UsedRange

$CriteriaArray = @('CA03','CY02','DEF02',"VA02","=")

# select blank with "=" OR ""

#$CriteriaArray = @('CA03','CY02','DEF02',"VA02")

$xlFilterValues = 7

$objRange.AutoFilter(7,$CriteriaArray,$xlFilterValues)

Output:

A screenshot of a computer

Description automatically generated

Task 9# – Below are more autofilter examples. See the comments (prefixed with #) above the lines of code for the explanation.

# More complex autofilters

$Filename = "c:\temp\mysheet.xlsx"

$objExcel = New-Object -ComObject Excel.Application

$objExcel.DisplayAlerts = $False

$objWrkBk = $objExcel.Workbooks.Open($Filename)

$objExcel.Visible = $True

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

$objWrkSht1.Select()

$objRange = $objWrkSht1.UsedRange

$xlFilterValues = 7

$xlAnd = 1

$xlOr = 2

# single criteria

$Criteria = "<>ABC01" # Not Equal, use with xlAnd

$Criteria = "=ABC\*" # Begins with, use with xlAnd

$Criteria = "=\*02" # Ends with, use with xlAnd

$Criteria = "=\*BC\*" # Contains, use with xlAnd

$Criteria = "<>\*BC\*" # Does Not Contain, use with xlAnd

$objRange.AutoFilter(7,$Criteria,$xlAnd)

# 2 Criteria with logical And

$Criteria1 = "=A\*"

$Criteria2 = "=\*1"

$objRange.AutoFilter(7,$Criteria1,$xlAnd,$Criteria2)

# 2 Criteria with logical Or

$objRange.AutoFilter(7,$Criteria1,$xlOr,$Criteria2)

# TO CLEAR EXISTING FILTERS

$objRange.AutoFilter()

Task 10# – Select cells that were auto-filtered

Code:

# Code

# apply autofilter and then select the active cells that are visible from top left to bottom right

$Filename = "c:\temp\mysheet.xlsx"

$objExcel = New-Object -ComObject Excel.Application

$objExcel.DisplayAlerts = $False

$objWrkBk = $objExcel.Workbooks.Open($Filename)

$objExcel.Visible = $True

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

$objWrkSht1.Select()

$objRange = $objWrkSht1.UsedRange

$xlCellTypeLastCell = 11

$objLastcell = $objRange.SpecialCells($xlCellTypeLastCell)

$lastrow = $objLastcell.row

$lastcolumn = $objLastcell.column

$xlFilterValues = 7

$xlAnd = 1

$xlOr = 2

# single criteria

$Criteria = "=DEF02"

$objRange.AutoFilter(7,$Criteria)

$xlLastCell = 11

#$objExcel.Cells.Select()

#$objExcel.Selection.AutoFilter(7,$Criteria)

$objExcel.ActiveCell.SpecialCells($xlLastCell).Select()

$objExcel.Range($objExcel.Selection,$objExcel.Cells(1)).Select()

Output:

A screenshot of a computer

Description automatically generated

Task 11# – Select auto-filtered cells, copy and past into new workbook, save as new file name and close.

Code:

$Filename = "c:\temp\mysheet.xlsx"

$objExcel = New-Object -ComObject Excel.Application

$objExcel.DisplayAlerts = $False

$objWrkBk = $objExcel.Workbooks.Open($Filename)

$objExcel.Visible = $True

$objWrkSht1 = $objWrkBk.Worksheets.item(1)

$objWrkSht1.Select()

$objRange = $objWrkSht1.UsedRange

# define constants

$xlCellTypeLastCell = 11

$xlOpenXMLWorkbook = 51

$xlFilterValues = 7

$xlAnd = 1

$xlOr = 2

$xlLastCell = 11

# find last cell

$objLastcell = $objRange.SpecialCells($xlCellTypeLastCell)

$lastrow = $objLastcell.row

$lastcolumn = $objLastcell.column

# Set autofilter

$Criteria = "=DEF02"

$objRange.AutoFilter(7,$Criteria)

# Select filtered cells

$objExcel.ActiveCell.SpecialCells($xlLastCell).Select()

$objExcel.Range($objExcel.Selection,$objExcel.Cells(1)).Select()

# copy and paste to new workbook

$objExcel.Selection.Copy()

$objExcel.Workbooks.Add()

$objExcel.ActiveSheet.Paste()

# set nowrap and column autofit

$objExcel.Selection.WrapText = $False

$objExcel.Selection.Columns.AutoFit()

$objExcel.Range("A2").Select()

# save as new excel workbook

$objExcel.ActiveWorkbook.SaveAs("c:\temp\DEF02-SR-Oct-2018.xlsx",$xlOpenXMLWorkbook)

#

$objExcel.Workbooks.Close()

$objExcel.Quit()

Initial workbook:

A screenshot of a computer

Description automatically generated

Output after applying the autofilter and saving as new workbook:

A screenshot of a computer

Description automatically generated

Task 12# – Import data from CSV file into a table-like custom object.

The Import-Csv cmdlet creates table-like custom objects from the items in CSV files. Each column in the CSV file becomes a property of the custom object and the items in rows become the property values.

 Code:

# Code

# This is native to powershell and not really part of Excel

$mycontacts = import-csv "c:\temp\contacts.csv"

# Creates table-like custom objects from the items in a CSV file

# An array of row items, each column accessible by .ColumnName

$mycontacts[0].name,$mycontacts[0].address,$mycontacts[0].phone

'------------'

foreach ($contact in $mycontacts) { $contact.name,$contact.address,$contact.phone }

Using this source .csv file for our example:

A screenshot of a computer

Description automatically generated

Output :

tom

1 main street

12345678

------------

tom

1 main street

12345678

dick

2 side avenue

23456781

mary

3 highland drive

34567812