How to use Export-CSV in PowerShell

lazyadmin.nl/powershell/export-csv-in-powershell

September 11, 2021

With PowerShell, you can extract all kinds of information from services like Active Directory or Microsoft 365. But sometimes you need to process this information further in Excel or another system. To do this we can use the Export-CSV function in PowerShell.

The Export-CSV function converts PowerShell objects into a CSV string and saves them into a CSV file. If you only need a CSV string, then you can also use the ConvertTo-CSV function in PowerShell.

In this article, we are going to take a look at how to use the Export-CSV function, how to prevent common mistakes, and what different options there are that you can use.

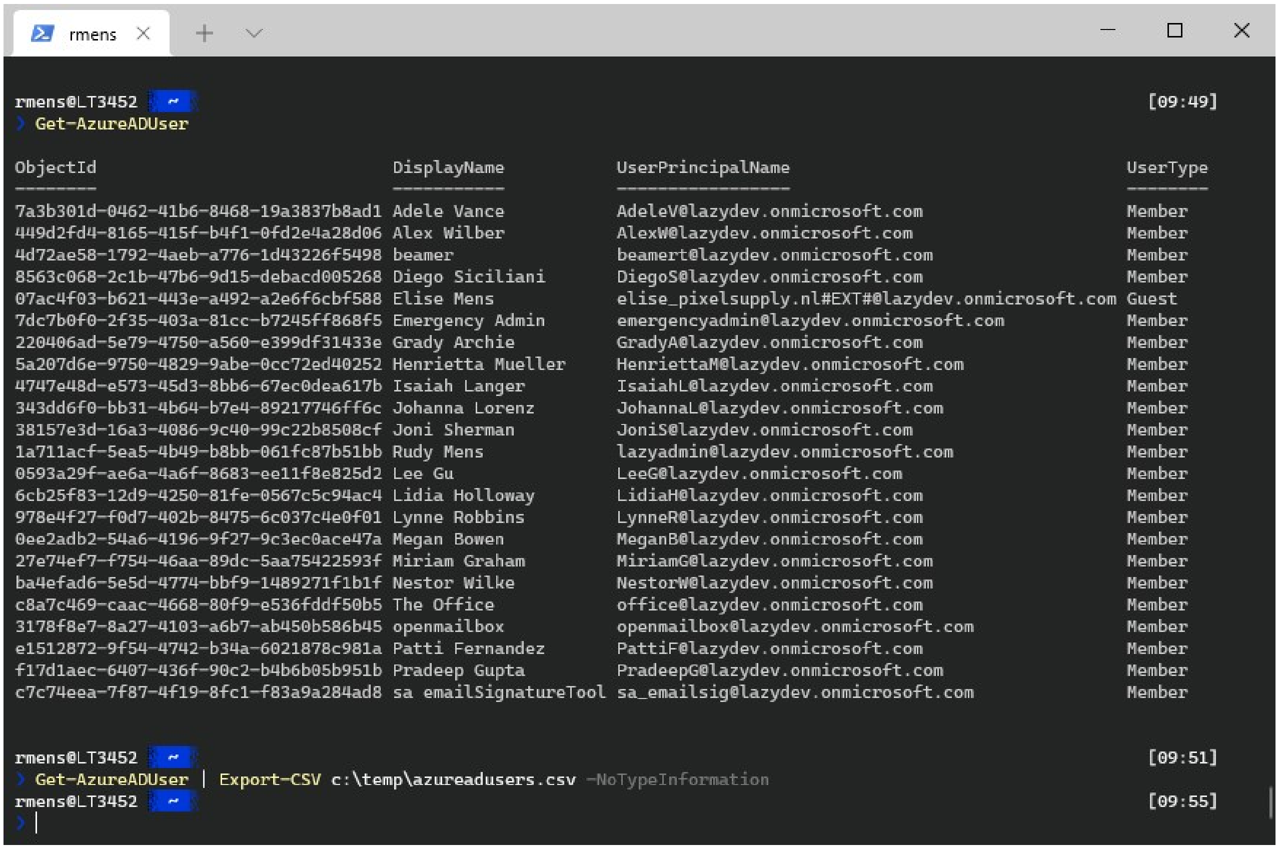
PowerShell Export-CSV

The Export-CSV cmdlet is pretty straightforward and only has a few properties that are useful:

- **Path** – (Required) Location of the CSV file
- **NoTypeInformation** – Removes the Type information header from the output. Not needed any more in PowerShell 6
- **Delimiter** – Default is comma, but you can change it
- **Append** – Append to an existing CSV file
- **Force** – Useful in combination with Append
- **NoClobber** – Don’t overwrite existing files
- **UseQuotes** – (PowerShell 7 only) wrap values in quotes or not

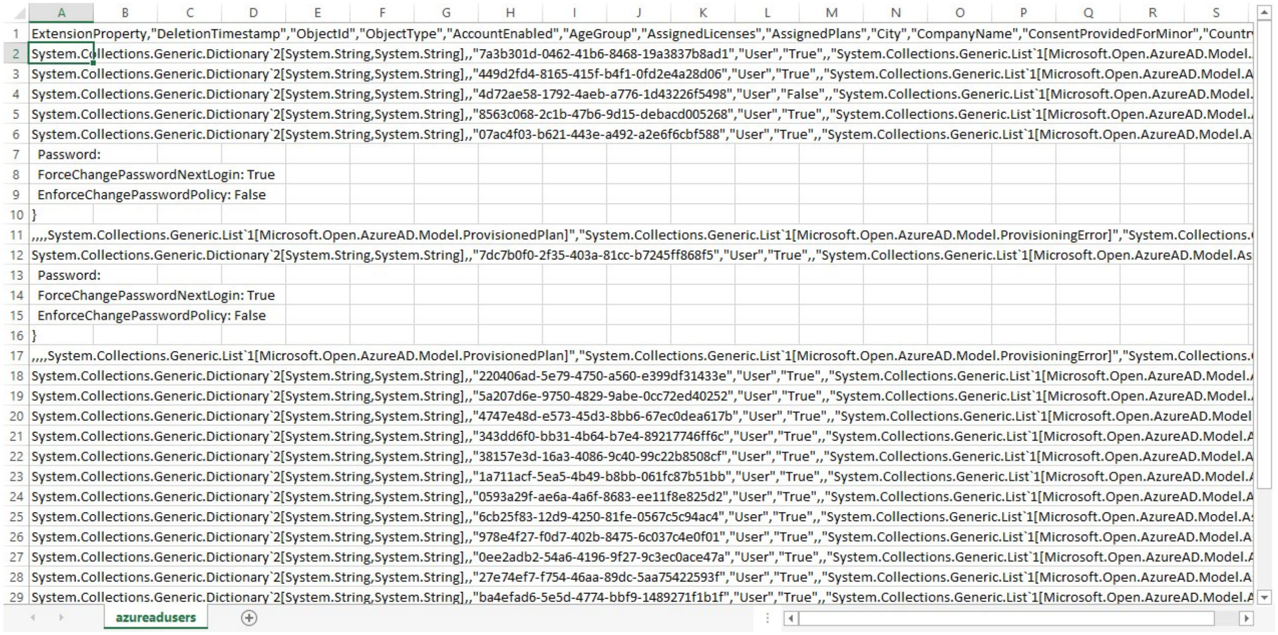
We are going to start with something simple, exporting our Microsoft 365 users to a CSV file. I am going to use Azure AD throughout the examples here, so if you want to follow along, make sure you connect the Azure AD first.

The `Get-AzureADUser` cmdlet returns all the users in your Microsoft 365 tenant, as you can see in the screenshot below. What we are going to do is to export this output to a CSV file. To do this we can simply pipe the `Export-CSV` cmdlet behind it:



```
rmens@LT3452  ~                                                                                [09:49]
❯ Get-AzureADUser

ObjectId                             DisplayName            UserPrincipalName                                  UserType
--------                             -----------            -----------------                                  --------
7a3b301d-0462-41b6-8468-19a3837b8ad1 Adele Vance            AdeleV@lazydev.onmicrosoft.com                     Member
449d2fd4-8165-415f-b4f1-0fd2e4a28d06 Alex Wilber            AlexW@lazydev.onmicrosoft.com                      Member
4d72ae58-1792-4aeb-a776-1d43226f5498 beamer                 beamert@lazydev.onmicrosoft.com                    Member
8563c068-2c1b-47b6-9d15-debacd005268 Diego Siciliani        DiegoS@lazydev.onmicrosoft.com                     Member
07ac4f03-b621-443e-a492-a2e6f6cbf588 Elise Mens             elise_pixelsupply.nl#EXT#@lazydev.onmicrosoft.com Guest
7dc7b0f0-2f35-403a-81cc-b7245ff868f5 Emergency Admin        emergencyadmin@lazydev.onmicrosoft.com             Member
220406ad-5e79-4750-a560-e399df31433e Grady Archie           GradyA@lazydev.onmicrosoft.com                     Member
5a207d6e-9750-4829-9abe-0cc72ed40252 Henrietta Mueller      HenriettaM@lazydev.onmicrosoft.com                 Member
4747e48d-e573-45d3-8bb6-67ec0dea617b Isaiah Langer          IsaiahL@lazydev.onmicrosoft.com                    Member
343dd6f0-bb31-4b64-b7e4-89217746ff6c Johanna Lorenz         JohannaL@lazydev.onmicrosoft.com                   Member
38157e3d-16a3-4086-9c40-99c22b8508cf Joni Sherman           JoniS@lazydev.onmicrosoft.com                      Member
1a711acf-5ea5-4b49-b8bb-061fc87b51bb Rudy Mens              lazyadmin@lazydev.onmicrosoft.com                  Member
0593a29f-ae6a-4a6f-8683-ee11f8e825d2 Lee Gu                 LeeG@lazydev.onmicrosoft.com                       Member
6cb25f83-12d9-4250-81fe-0567c5c94ac4 Lidia Holloway         LidiaH@lazydev.onmicrosoft.com                     Member
978e4f27-f0d7-402b-8475-6c037c4e0f01 Lynne Robbins          LynneR@lazydev.onmicrosoft.com                     Member
0ee2adb2-54a6-4196-9f27-9c3ec0ace47a Megan Bowen            MeganB@lazydev.onmicrosoft.com                     Member
27e74ef7-f754-46aa-89dc-5aa75422593f Miriam Graham          MiriamG@lazydev.onmicrosoft.com                    Member
ba4efad6-5e5d-4774-bbf9-1489271f1b1f Nestor Wilke           NestorW@lazydev.onmicrosoft.com                    Member
c8a7c469-caac-4668-80f9-e536fddf50b5 The Office             office@lazydev.onmicrosoft.com                     Member
3178f8e7-8a27-4103-a6b7-ab450b586b45 openmailbox            openmailbox@lazydev.onmicrosoft.com                Member
e1512872-9f54-4742-b34a-6021878c981a Patti Fernandez        PattiF@lazydev.onmicrosoft.com                     Member
f17d1aec-6407-436f-90c2-b4b6b05b951b Pradeep Gupta          PradeepG@lazydev.onmicrosoft.com                   Member
c7c74eea-7f87-4f19-8fc1-f83a9a284ad8 sa emailSignatureTool  sa_emailsig@lazydev.onmicrosoft.com                Member


rmens@LT3452  ~                                                                                [09:51]
❯ Get-AzureADUser | Export-CSV c:\temp\azureadusers.csv -NoTypeInformation
rmens@LT3452  ~                                                                                [09:55]
❯ |
```

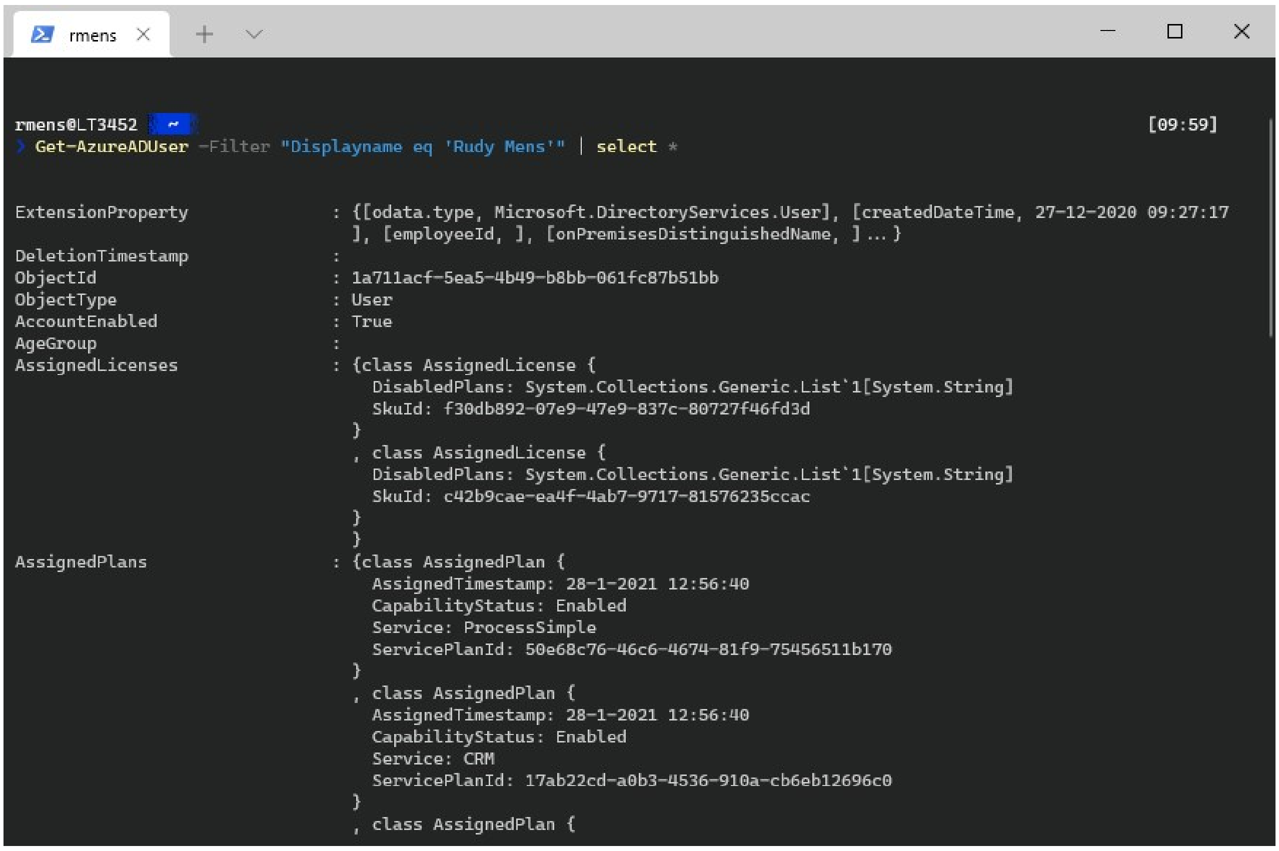
Get-AzureADUser | Export-Csv c:\temp\azureadusers.csv -NoTypeInformation
Sounds easy, right? Well if you open the CSV file you will notice that we got a bit more than we needed and not the nice list that we had seen in PowerShell before.



```
ExtensionProperty,"DeletionTimestamp","ObjectId","ObjectType","AccountEnabled","AgeGroup","AssignedLicenses","AssignedPlans","City","CompanyName","ConsentProvidedForMinor","Countr
System.Collections.Generic.Dictionary`2[System.String,System.String],,"7a3b301d-0462-41b6-8468-19a3837b8ad1","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"449d2fd4-8165-415f-b4f1-0fd2e4a28d06","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"4d72ae58-1792-4aeb-a776-1d43226f5498","User","False",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"8563c068-2c1b-47b6-9d15-debacd005268","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"07ac4f03-b621-443e-a492-a2e6f6cbf588","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
 Password:
 ForceChangePasswordNextLogin: True
 EnforceChangePasswordPolicy: False
}
,,,,System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.ProvisionedPlan]","System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.ProvisioningError]","System.Collections.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"7dc7b0f0-2f35-403a-81cc-b7245ff868f5","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.As
 Password:
 ForceChangePasswordNextLogin: True
 EnforceChangePasswordPolicy: False
}
,,,,System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.ProvisionedPlan]","System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.ProvisioningError]","System.Collections.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"220406ad-5e79-4750-a560-e399df31433e","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"5a207d6e-9750-4829-9abe-0cc72ed40252","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"4747e48d-e573-45d3-8bb6-67ec0dea617b","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model
System.Collections.Generic.Dictionary`2[System.String,System.String],,"343dd6f0-bb31-4b64-b7e4-89217746ff6c","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"38157e3d-16a3-4086-9c40-99c22b8508cf","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"1a711acf-5ea5-4b49-b8bb-061fc87b51bb","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"0593a29f-ae6a-4a6f-8683-ee11f8e825d2","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"6cb25f83-12d9-4250-81fe-0567c5c94ac4","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"978e4f27-f0d7-402b-8475-6c037c4e0f01","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"0ee2adb2-54a6-4196-9f27-9c3ec0ace47a","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
System.Collections.Generic.Dictionary`2[System.String,System.String],,"27e74ef7-f754-46aa-89dc-5aa75422593f","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.A
System.Collections.Generic.Dictionary`2[System.String,System.String],,"ba4efad6-5e5d-4774-bbf9-1489271f1b1f","User","True",,"System.Collections.Generic.List`1[Microsoft.Open.AzureAD.Model.
```

So why is this happening? The Export-CSV cmdlet exports **all the individual objects** from the Get-AzureADUser cmdlet. If we look up a single Azure AD user then you can see all the data that is returned from a single user object:

Get-AzureADUser -Filter "Displayname eq 'Rudy Mens'" | select *



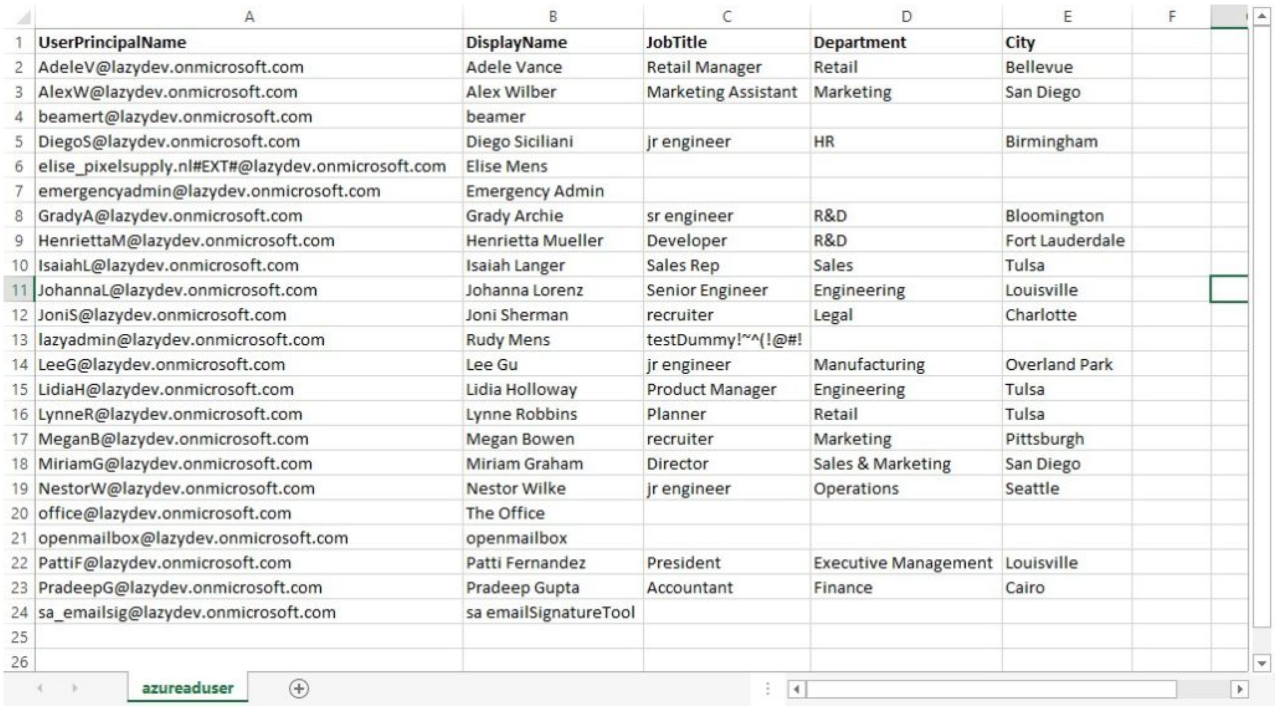
```
rmens@LT3452  ~                                                                        [09:59]
> Get-AzureADUser -Filter "Displayname eq 'Rudy Mens'" | select *

ExtensionProperty             : {[odata.type, Microsoft.DirectoryServices.User], [createdDateTime, 27-12-2020 09:27:17
                                ], [employeeId, ], [onPremisesDistinguishedName, ]…}
DeletionTimestamp             :
ObjectId                      : 1a711acf-5ea5-4b49-b8bb-061fc87b51bb
ObjectType                    : User
AccountEnabled                : True
AgeGroup                      :
AssignedLicenses              : {class AssignedLicense {
                                  DisabledPlans: System.Collections.Generic.List`1[System.String]
                                  SkuId: f30db892-07e9-47e9-837c-80727f46fd3d
                                }
                                , class AssignedLicense {
                                  DisabledPlans: System.Collections.Generic.List`1[System.String]
                                  SkuId: c42b9cae-ea4f-4ab7-9717-81576235ccac
                                }
                                }
AssignedPlans                 : {class AssignedPlan {
                                  AssignedTimestamp: 28-1-2021 12:56:40
                                  CapabilityStatus: Enabled
                                  Service: ProcessSimple
                                  ServicePlanId: 50e68c76-46c6-4674-81f9-75456511b170
                                }
                                , class AssignedPlan {
                                  AssignedTimestamp: 28-1-2021 12:56:40
                                  CapabilityStatus: Enabled
                                  Service: CRM
                                  ServicePlanId: 17ab22cd-a0b3-4536-910a-cb6eb12696c0
                                }
                                , class AssignedPlan {
```

How to Export the correct information with Export-CSV

What we need to do is first select the correct information (properties) that we need before we export the user objects to a CSV file.

Get-AzureADUser | select userprincipalname, displayname, jobtitle, department, city | Export-CSV c:\temp\azureaduser.csv -NoTypeInformation

This will return the selected fields from each user in a nice CSV file that we can actually use:



	A	B	C	D	E	F
1	UserPrincipalName	DisplayName	JobTitle	Department	City	
2	AdeleV@lazydev.onmicrosoft.com	Adele Vance	Retail Manager	Retail	Bellevue	
3	AlexW@lazydev.onmicrosoft.com	Alex Wilber	Marketing Assistant	Marketing	San Diego	
4	beamert@lazydev.onmicrosoft.com	beamer				
5	DiegoS@lazydev.onmicrosoft.com	Diego Siciliani	jr engineer	HR	Birmingham	
6	elise_pixelsupply.nl#EXT#@lazydev.onmicrosoft.com	Elise Mens				
7	emergencyadmin@lazydev.onmicrosoft.com	Emergency Admin				
8	GradyA@lazydev.onmicrosoft.com	Grady Archie	sr engineer	R&D	Bloomington	
9	HenriettaM@lazydev.onmicrosoft.com	Henrietta Mueller	Developer	R&D	Fort Lauderdale	
10	IsaiahL@lazydev.onmicrosoft.com	Isaiah Langer	Sales Rep	Sales	Tulsa	
11	JohannaL@lazydev.onmicrosoft.com	Johanna Lorenz	Senior Engineer	Engineering	Louisville	
12	JoniS@lazydev.onmicrosoft.com	Joni Sherman	recruiter	Legal	Charlotte	
13	lazyadmin@lazydev.onmicrosoft.com	Rudy Mens	testDummy!~^(!@#!			
14	LeeG@lazydev.onmicrosoft.com	Lee Gu	jr engineer	Manufacturing	Overland Park	
15	LidiaH@lazydev.onmicrosoft.com	Lidia Holloway	Product Manager	Engineering	Tulsa	
16	LynneR@lazydev.onmicrosoft.com	Lynne Robbins	Planner	Retail	Tulsa	
17	MeganB@lazydev.onmicrosoft.com	Megan Bowen	recruiter	Marketing	Pittsburgh	
18	MiriamG@lazydev.onmicrosoft.com	Miriam Graham	Director	Sales & Marketing	San Diego	
19	NestorW@lazydev.onmicrosoft.com	Nestor Wilke	jr engineer	Operations	Seattle	
20	office@lazydev.onmicrosoft.com	The Office				
21	openmailbox@lazydev.onmicrosoft.com	openmailbox				
22	PattiF@lazydev.onmicrosoft.com	Patti Fernandez	President	Executive Management	Louisville	
23	PradeepG@lazydev.onmicrosoft.com	Pradeep Gupta	Accountant	Finance	Cairo	
24	sa_emailsig@lazydev.onmicrosoft.com	sa emailSignatureTool				
25						
26						

Appending to CSV file with PowerShell

On some occasions, you may want to append information to a CSV file with PowerShell. Let's say we want to add the manager of each user to the list. There is a more efficient way to do this, but in this case, we are going to loop through all the users and get the manager from Azure AD.

> Tip
>
> *Learn more about creating PowerShell Scripts in this complete guide*

For this example, we create a custom PowerShell object that will hold the user information and the manager. With the `-append` switch we can add the user to the CSV file.

```
$users = Get-AzureADUser
$users | ForEach-Object {
$Manager = Get-AzureADUserManager -ObjectId $_.ObjectId
$user =[pscustomobject]@{
'Displayname' = $_.displayname
'Jobtitle' = $_.jobtitle
'Department' = $_.Department
'Manager' = $manager.displayname
}
$user | Export-CSV c:\temp\usermanager.csv -Append -NoTypeInformation -Force
}
```

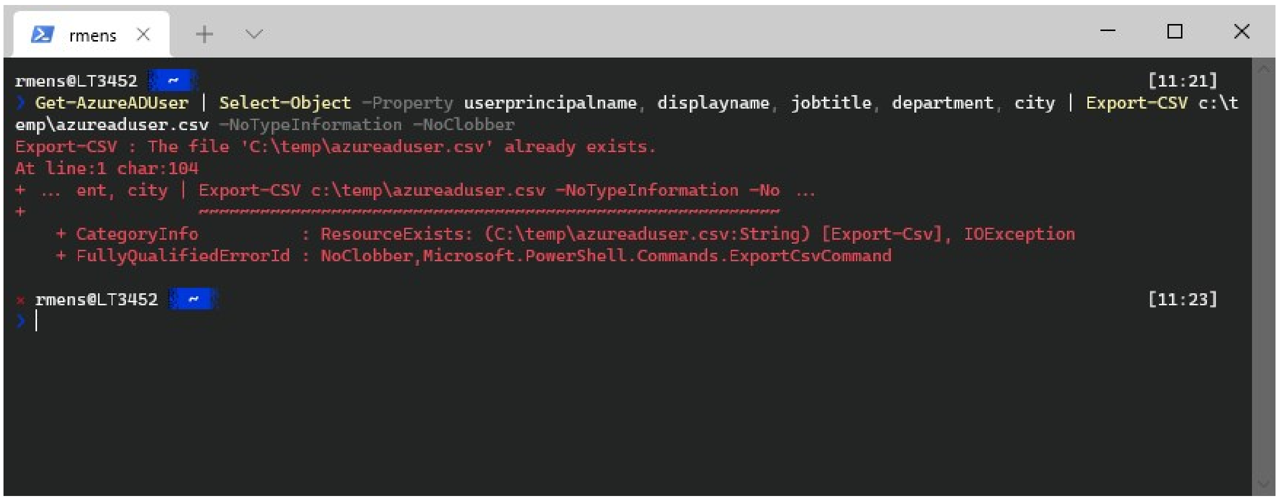
I have added the -Force switch as well. This way the CSV file will be created if it doesn't exist and objects that have mismatched properties can still be written to the CSV file. It will only write the properties that match though, other properties are discarded.

Other useful Export-CSV parameters in PowerShell

There are a couple of other parameters that are useful when you are using the Export-CSV cmdlet in PowerShell.

NoClobber

By default, the Export-CSV cmdlet will overwrite any existing file when used. If you don't want this then you can add the `-NoClobber` parameter. This way, if the file already exists, you will get an error, preventing you from overwriting the file.



```
rmens@LT3452  ~                                                          [11:21]
❯ Get-AzureADUser | Select-Object -Property userprincipalname, displayname, jobtitle, department, city | Export-CSV c:\t
emp\azureaduser.csv -NoTypeInformation -NoClobber
Export-CSV : The file 'C:\temp\azureaduser.csv' already exists.
At line:1 char:104
+ ... ent, city | Export-CSV c:\temp\azureaduser.csv -NoTypeInformation -No ...
+               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ResourceExists: (C:\temp\azureaduser.csv:String) [Export-Csv], IOException
    + FullyQualifiedErrorId : NoClobber,Microsoft.PowerShell.Commands.ExportCsvCommand

× rmens@LT3452  ~                                                        [11:23]
❯ |
```

Delimiter

The values in a Comma Separated Values File (CSV) are by default separated with a comma. Not all applications follow this standard. So when you need to import the data into another application that required a semicolon (;) for example, then you can change the delimiter.

Make sure though that you enclose the new delimiter in quotation marks:

Get-AzureADUser | Select-Object -Property userprincipalname, displayname, jobtitle, department, city | Export-CSV c:\temp\azureaduser.csv -NoTypeInformation -Delimiter ';'

UseQuotes

When you export objects in PowerShell 5 or 6 then all values are wrapped in quotation marks. On some occasions, you might don't want that. If you are using PowerShell 7 then you can use the -UseQuotes parameter.

This parameter has a couple of options:

- AsNeeded
- Always (default)
- Never

Get-Process | export-csv -Path c:\temp\process.csv -UseQuotes AsNeeded

Wrapping Up

The Export-CSV cmdlet is a PowerShell function that I use quite a lot. It's really useful to pull data out of Microsoft 365 quickly for reports or other tasks. Make sure you select first the correct properties when exporting objects. This is the most commonly made mistake when it comes to exporting PowerShell data to Excel.

If you have any questions, then just drop a comment below.

Did you **Liked** this **Article**?