SQLShack

SQL Server training Español

Script SQL Server objects using DBATools

June 20, 2019 by Rajendra Gupta

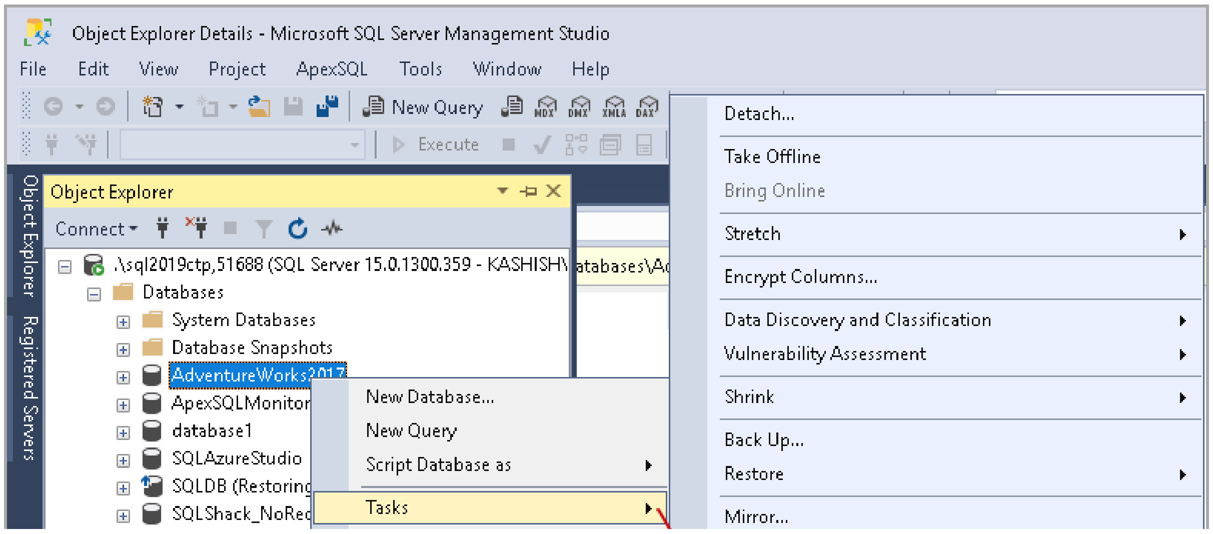
This article gives an overview to generate scripts for SQL Server objects with Windows PowerShell tool DBATools.

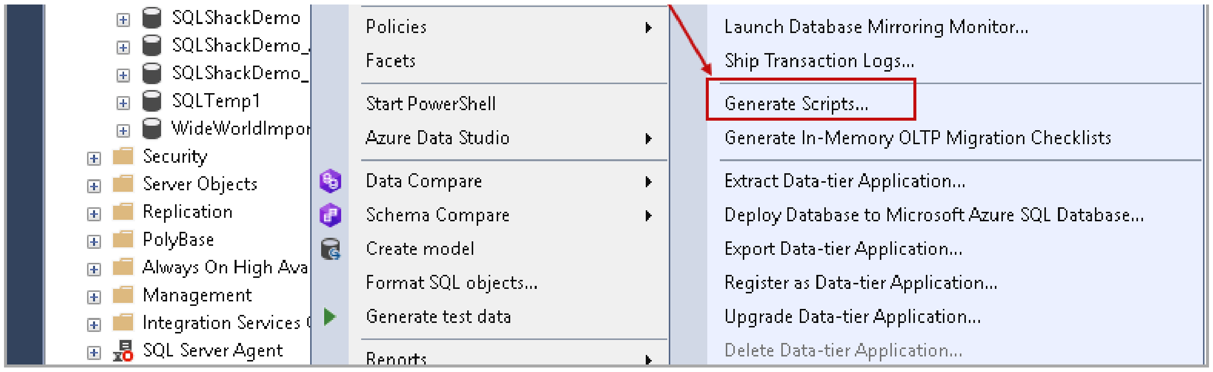
Database administrators or developers require generating scripts for SQL Server objects. We might need scripts to store a copy of object script before object change, create specific objects into other database environments such as development, UAT or non-prod environment. It is an excellent practice to keep a copy of the object before making a change to it. We can easily refer to the old script and roll back if required. Usually, we use SSMS Generate Scripts wizard to get these scripts.

Generate Script Wizard in SSMS

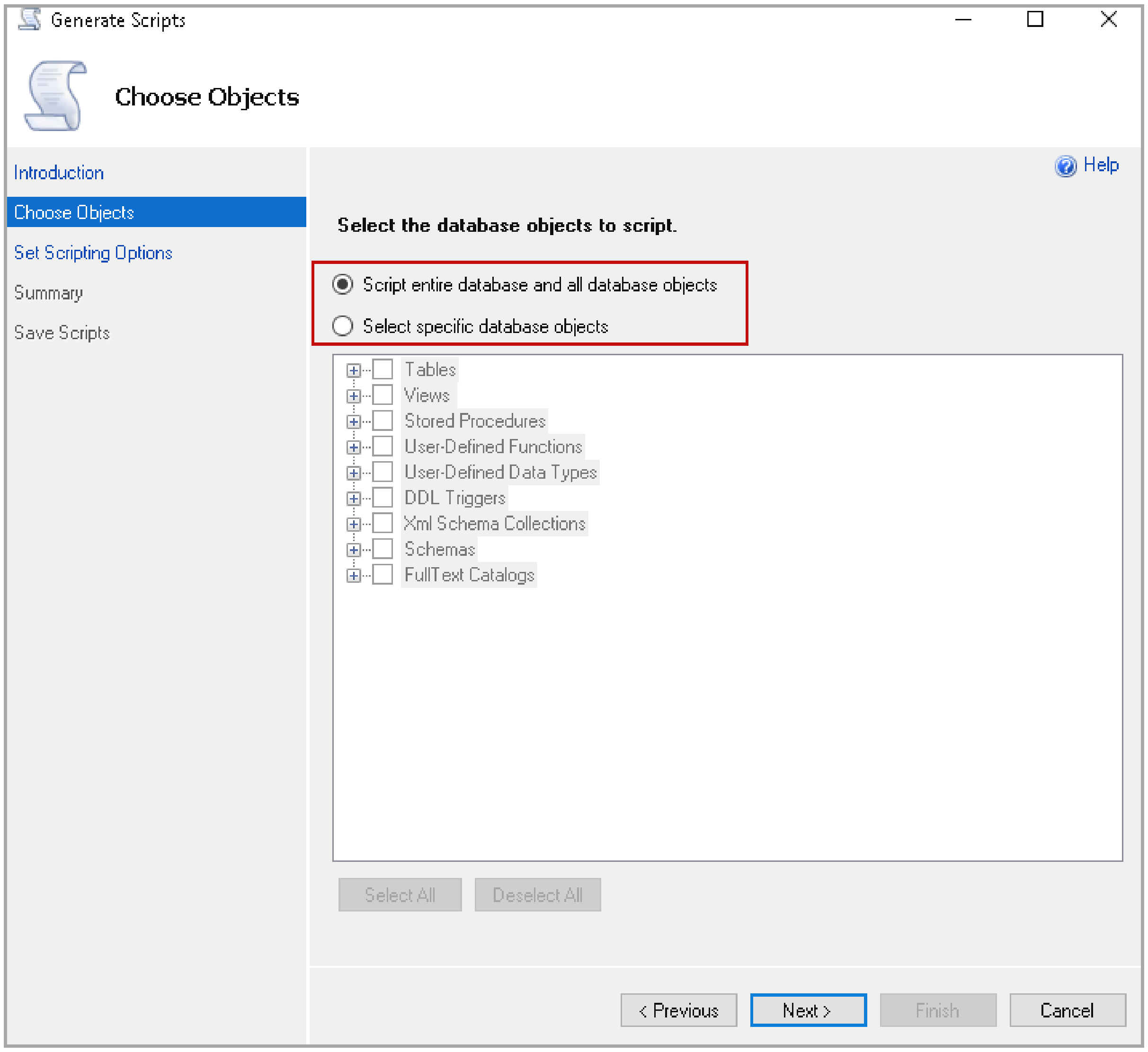
Let's have a quick review of Generate Scripts Wizard in SSMS.

Right click on a database and go to Tasks and Generate Scripts.





It gives the option to script the entire database or specific database object.

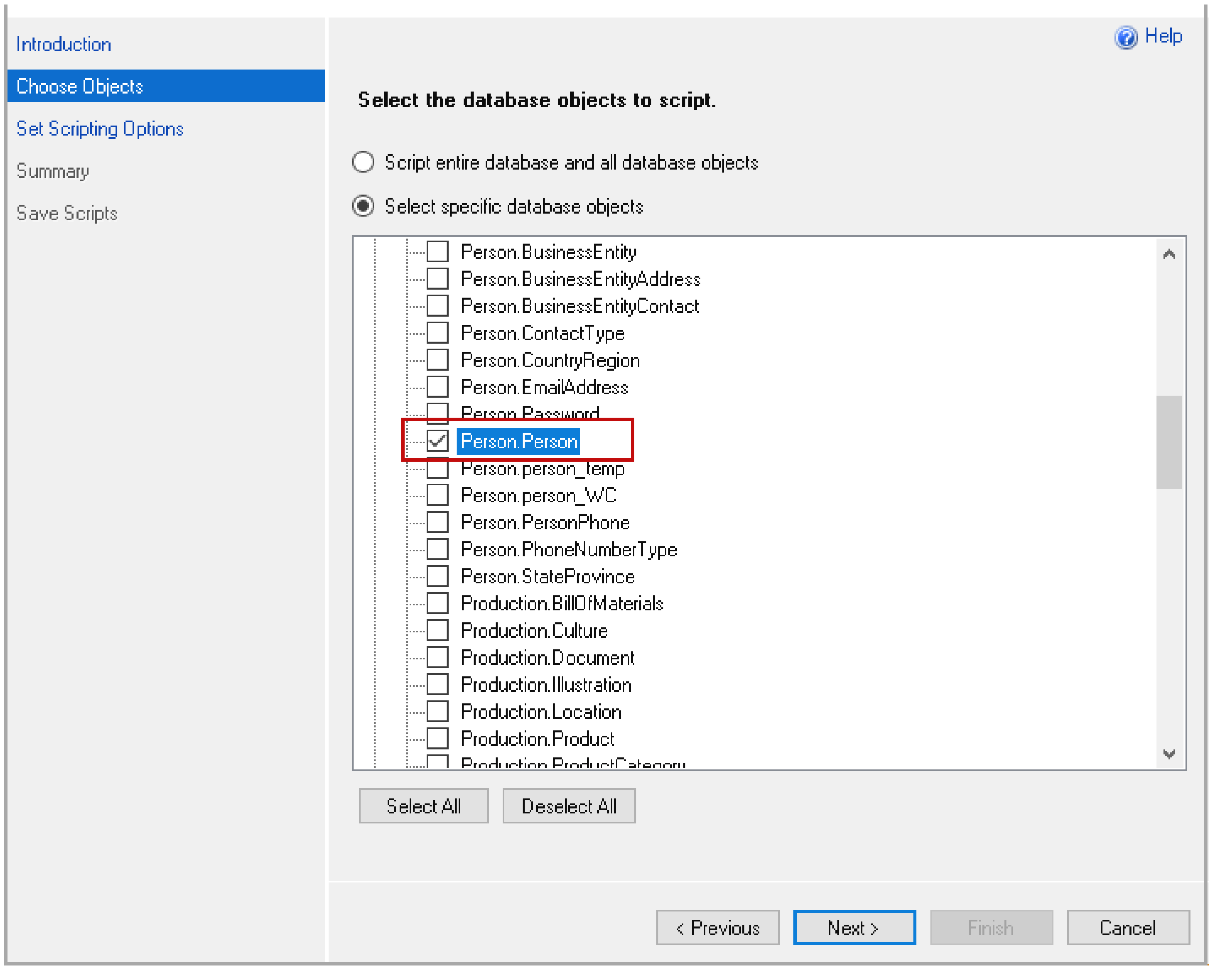


Select the database objects to script.

● Script entire database and all database objects

○ Select specific database objects

Select specific database objects to script out and click on Next. We might select multiple objects as well to script out together.

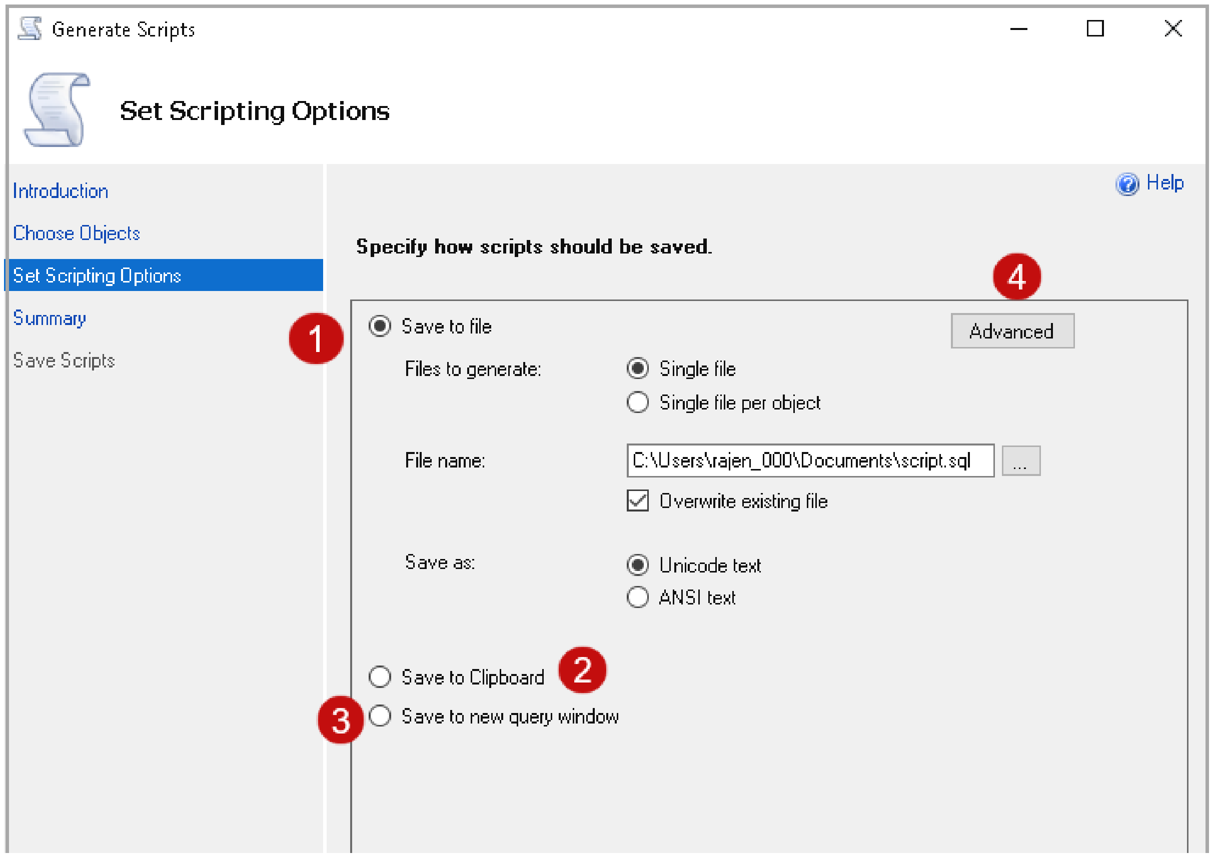


Select the database objects to script.

○ Script entire database and all database objects
● Select specific database objects

- ☐ Person.BusinessEntity
- ☐ Person.BusinessEntityAddress
- ☐ Person.BusinessEntityContact
- ☐ Person.ContactType
- ☐ Person.CountryRegion
- ☐ Person.EmailAddress
- ☐ Person.Password
- ☑ Person.Person
- ☐ Person.person_temp
- ☐ Person.person_WC
- ☐ Person.PersonPhone
- ☐ Person.PhoneNumberType
- ☐ Person.StateProvince
- ☐ Production.BillOfMaterials
- ☐ Production.Culture
- ☐ Production.Document
- ☐ Production.Illustration
- ☐ Production.Location
- ☐ Production.Product
- ☐ Production.ProductCategory

In the next page, it gives us the scripting options. We get the following options.



Set Scripting Options

Specify how scripts should be saved.

● Save to file

Files to generate: ● Single file ○ Single file per object

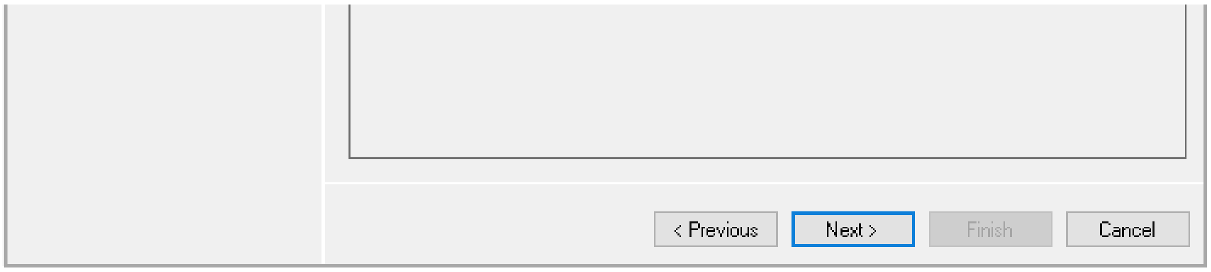
File name: C:\Users\rajen_000\Documents\script.sql

☑ Overwrite existing file

Save as: ● Unicode text ○ ANSI text

○ Save to Clipboard

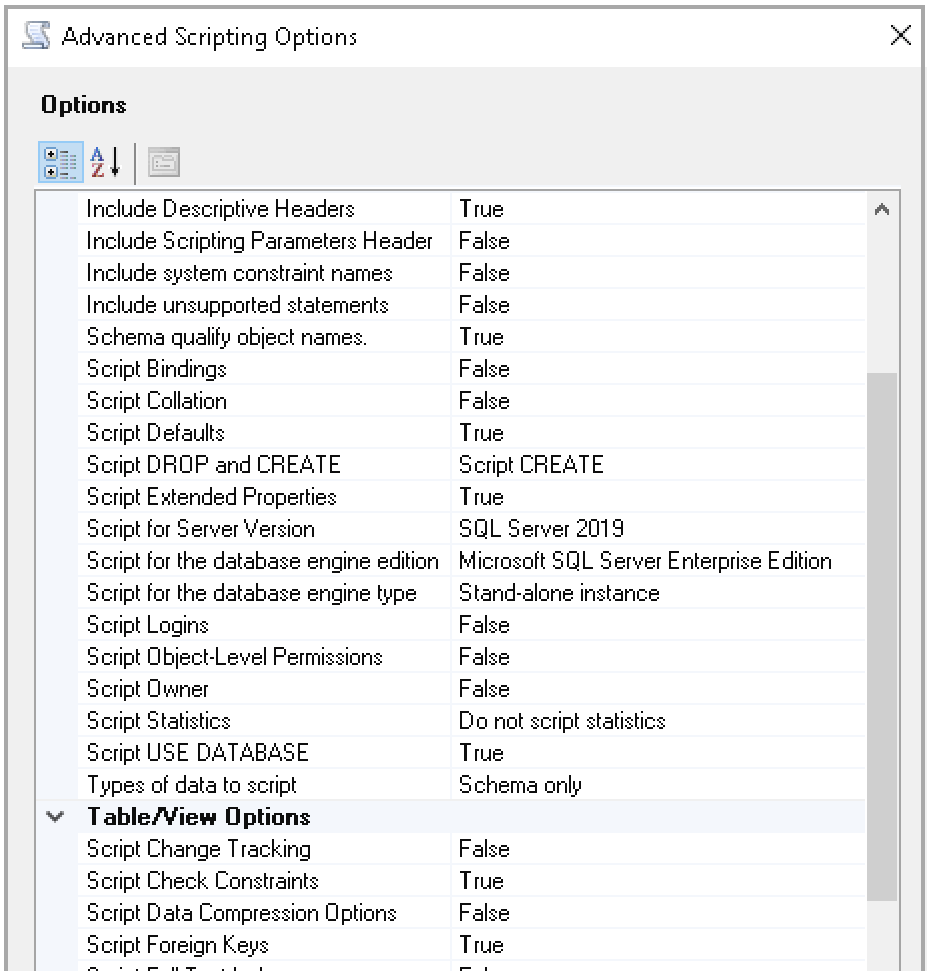
○ Save to new query window



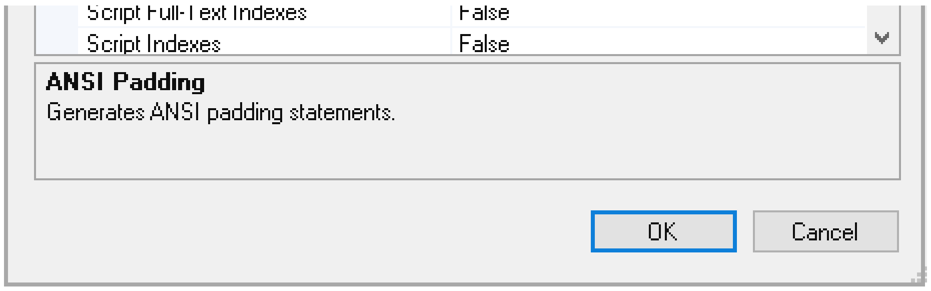
1. Save to file: Select this option to save the script as a file. If we want to generate scripts for multiple objects together, it generates all scripts in a single SQL file. We can use option **Single file per object** to generate all scripts in different files
2. Save to clipboard: we can save the generated script to the clipboard using this option
3. Save to New query window: It generates the script and opens it in a new query window of SSMS

Click on **Advanced** to set advanced scripting options. On this page, you can set various options to generate scripts. A few relevant options are as follows.

- Script for the server version
- Script primary and foreign keys
- Script change tracking
- Script primary, unique keys
- Types of data to script – Schema only, Data Only and Schema with Data



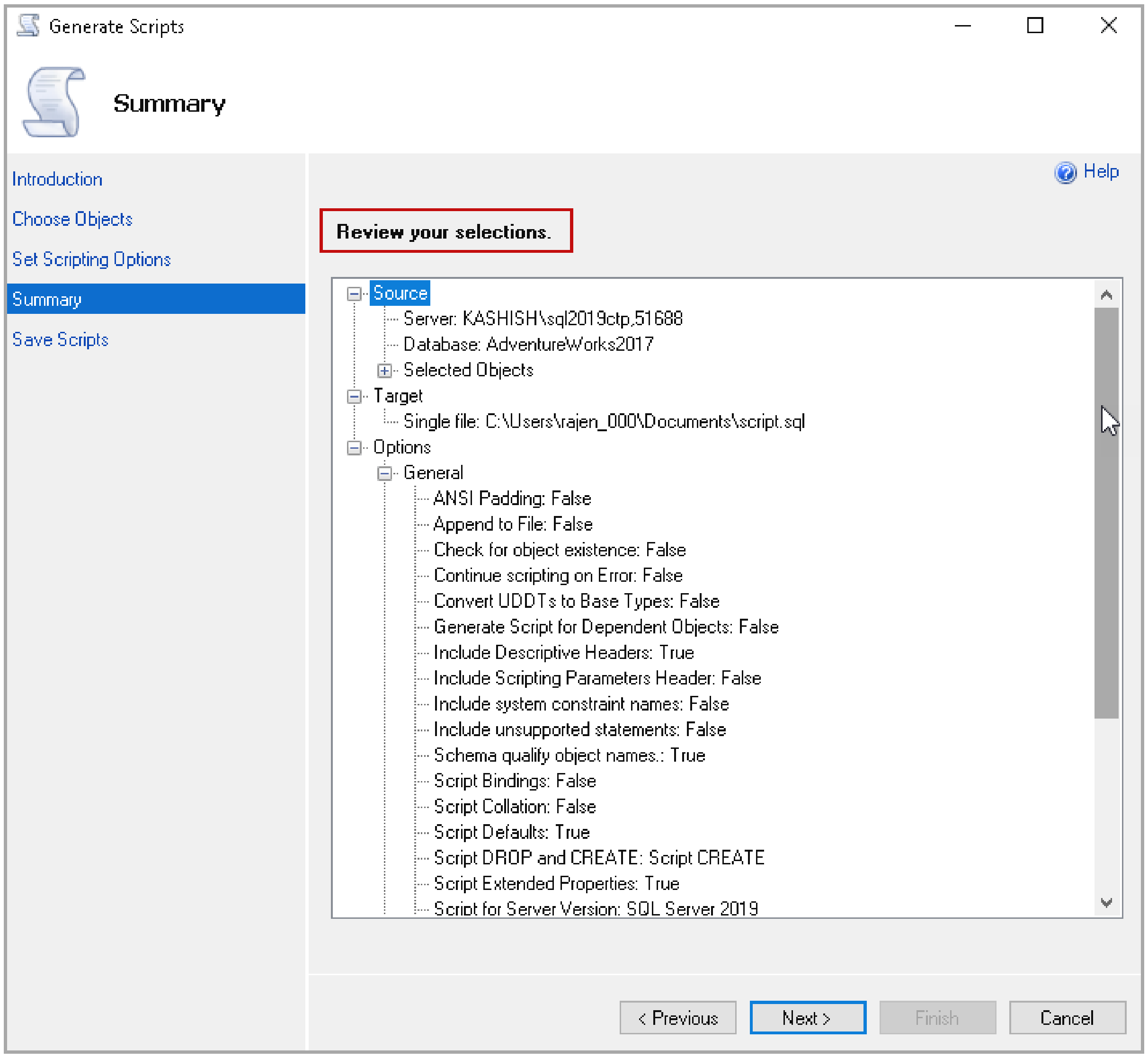
Include Descriptive Headers	True
Include Scripting Parameters Header	False
Include system constraint names	False
Include unsupported statements	False
Schema qualify object names.	True
Script Bindings	False
Script Collation	False
Script Defaults	True
Script DROP and CREATE	Script CREATE
Script Extended Properties	True
Script for Server Version	SQL Server 2019
Script for the database engine edition	Microsoft SQL Server Enterprise Edition
Script for the database engine type	Stand-alone instance
Script Logins	False
Script Object-Level Permissions	False
Script Owner	False
Script Statistics	Do not script statistics
Script USE DATABASE	True
Types of data to script	Schema only
Table/View Options	
Script Change Tracking	False
Script Check Constraints	True
Script Data Compression Options	False
Script Foreign Keys	True



Script Full-Text Indexes False
Script Indexes False

ANSI Padding
Generates ANSI padding statements.

On the next page, we can review the configuration and finish to generate object scripts.



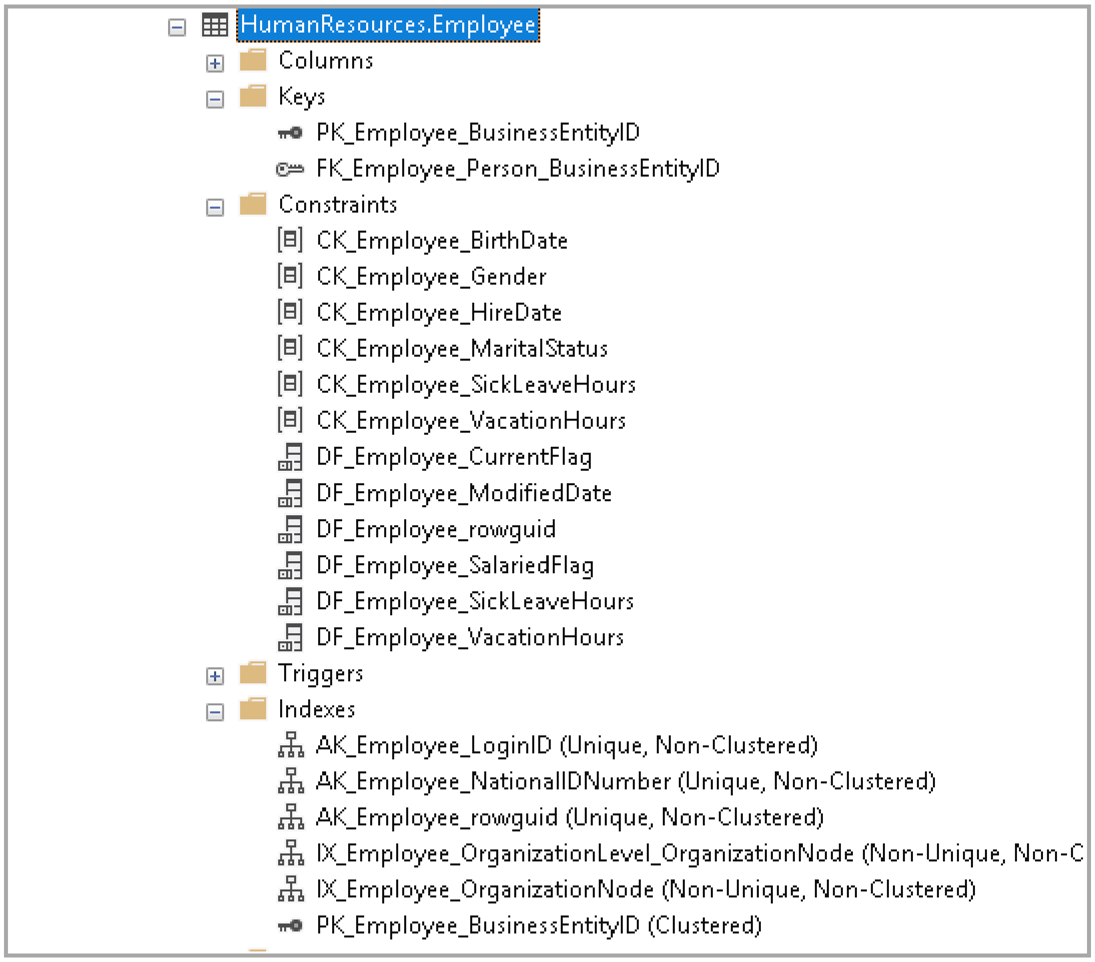
Review your selections.

```
Source
    Server: KASHISH\sql2019ctp,51688
    Database: AdventureWorks2017
    Selected Objects
Target
    Single file: C:\Users\rajen_000\Documents\script.sql
Options
    General
        ANSI Padding: False
        Append to File: False
        Check for object existence: False
        Continue scripting on Error: False
        Convert UDDTs to Base Types: False
        Generate Script for Dependent Objects: False
        Include Descriptive Headers: True
        Include Scripting Parameters Header: False
        Include system constraint names: False
        Include unsupported statements: False
        Schema qualify object names.: True
        Script Bindings: False
        Script Collation: False
        Script Defaults: True
        Script DROP and CREATE: Script CREATE
        Script Extended Properties: True
        Script for Server Version: SQL Server 2019
```

We need to repeat the same process depending upon the requirements of objects script. We might need to set options different for few objects. We need to follow this wizard for a specific object in this case. It might be a time-consuming process to do it.

This approach also works on instance level only. We need to do this task only for each SQL Server instance. We cannot use this task with multiple instances altogether.

In this article, I will use the AdventureWorks2017 database and HumanResources.Employee table. We can see that this table contains Primary, foreign keys, clustered, non-clustered index and triggers.



```
HumanResources.Employee
  Columns
  Keys
    PK_Employee_BusinessEntityID
    FK_Employee_Person_BusinessEntityID
  Constraints
    CK_Employee_BirthDate
    CK_Employee_Gender
    CK_Employee_HireDate
    CK_Employee_MaritalStatus
    CK_Employee_SickLeaveHours
    CK_Employee_VacationHours
    DF_Employee_CurrentFlag
    DF_Employee_ModifiedDate
    DF_Employee_rowguid
    DF_Employee_SalariedFlag
    DF_Employee_SickLeaveHours
    DF_Employee_VacationHours
  Triggers
  Indexes
    AK_Employee_LoginID (Unique, Non-Clustered)
    AK_Employee_NationalIDNumber (Unique, Non-Clustered)
    AK_Employee_rowguid (Unique, Non-Clustered)
    IX_Employee_OrganizationLevel_OrganizationNode (Non-Unique, Non-C
    IX_Employee_OrganizationNode (Non-Unique, Non-Clustered)
    PK_Employee_BusinessEntityID (Clustered)
```

In this case, we can use PowerShell open-source module DBATools to do this task for us.

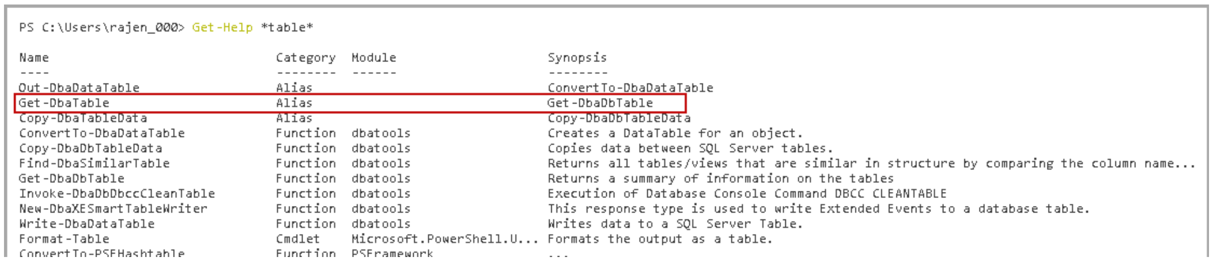
DBATools to generate object scripts

In my previous articles on DBAtools, we explored a few essential commands to do tasks in SQL Server. We need to use a combination of commands to generate object scripts using DBATools.

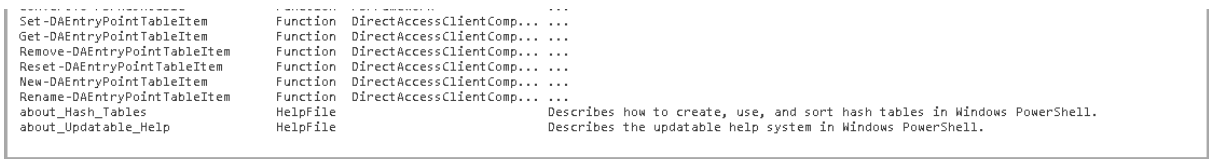
- Get-DbaTable

We use Get-Help command in DBATools to get search commands containing the keyword.

```
> Get-Help *table*
```

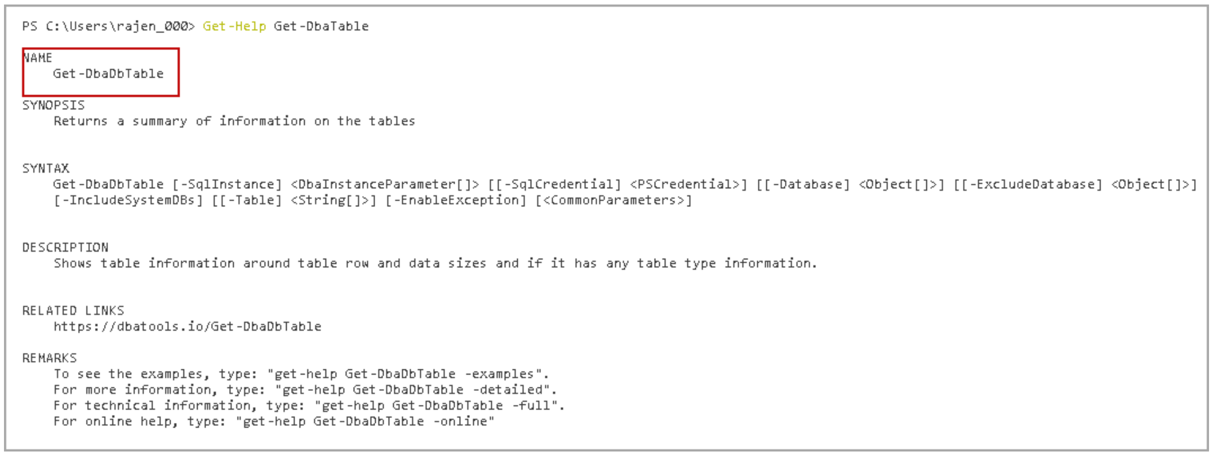


```
PS C:\Users\rajen_000> Get-Help *table*

Name                             Category  Module                    Synopsis
----                             --------  ------                    --------
Out-DbaDataTable                 Alias                               ConvertTo-DbaDataTable
Get-DbaTable                     Alias                               Get-DbaDbTable
Copy-DbaTableData                Alias                               Copy-DbaDbTableData
ConvertTo-DbaDataTable           Function  dbatools                  Creates a DataTable for an object.
Copy-DbaDbTableData              Function  dbatools                  Copies data between SQL Server tables.
Find-DbaSimilarTable             Function  dbatools                  Returns all tables/views that are similar in structure by comparing the column name...
Get-DbaDbTable                   Function  dbatools                  Returns a summary of information on the tables
Invoke-DbaDbDbccCleanTable       Function  dbatools                  Execution of Database Console Command DBCC CLEANTABLE
New-DbaXESmartTableWriter        Function  dbatools                  This response type is used to write Extended Events to a database table.
Write-DbaDataTable               Function  dbatools                  Writes data to a SQL Server Table.
Format-Table                     Cmdlet    Microsoft.PowerShell.U... Formats the output as a table.
ConvertTo-PSFHashtable           Function  PSFramework               ...
```



```
Set-DAEntryPointTableItem        Function  DirectAccessClientComp... ...
Get-DAEntryPointTableItem        Function  DirectAccessClientComp... ...
Remove-DAEntryPointTableItem     Function  DirectAccessClientComp... ...
Reset-DAEntryPointTableItem      Function  DirectAccessClientComp... ...
New-DAEntryPointTableItem        Function  DirectAccessClientComp... ...
Rename-DAEntryPointTableItem     Function  DirectAccessClientComp... ...
about_Hash_Tables                HelpFile                            Describes how to create, use, and sort hash tables in Windows PowerShell.
about_Updatable_Help             HelpFile                            Describes the updatable help system in Windows PowerShell.
```

You can find the synopsis and syntax of the Get-DbaTable command in DBATools.



```
PS C:\Users\rajen_000> Get-Help Get-DbaTable

NAME
    Get-DbaDbTable

SYNOPSIS
    Returns a summary of information on the tables


SYNTAX
    Get-DbaDbTable [-SqlInstance] <DbaInstanceParameter[]> [[-SqlCredential] <PSCredential>] [[-Database] <Object[]>] [[-ExcludeDatabase] <Object[]>]
    [-IncludeSystemDBs] [[-Table] <String[]>] [-EnableException] [<CommonParameters>]


DESCRIPTION
    Shows table information around table row and data sizes and if it has any table type information.


RELATED LINKS
    https://dbatools.io/Get-DbaDbTable

REMARKS
    To see the examples, type: "get-help Get-DbaDbTable -examples".
    For more information, type: "get-help Get-DbaDbTable -detailed".
    For technical information, type: "get-help Get-DbaDbTable -full".
    For online help, type: "get-help Get-DbaDbTable -online"
```

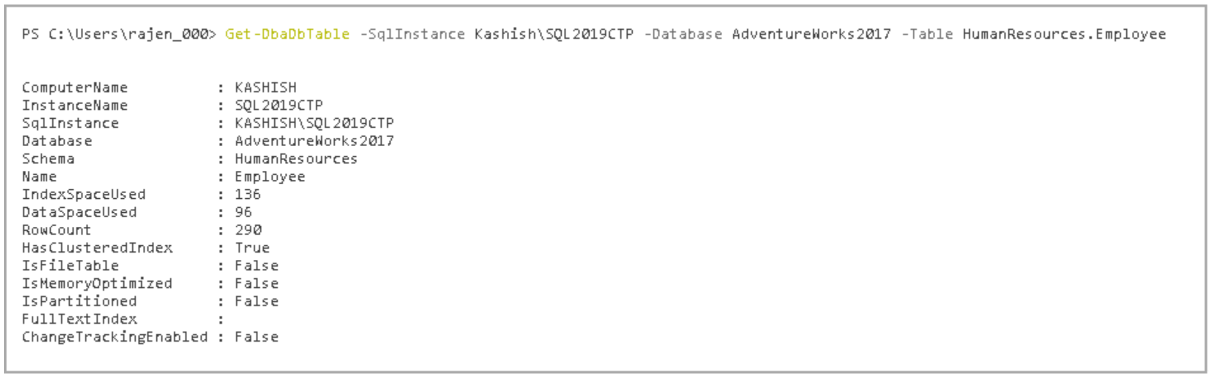
Let's run this command to get information about the *HumanResources.Employee* table.

In the following query, we use the following parameters.

- SqlInstance: We specify the SQL instance using this parameter
- Database: We can specify the database name in this parameter
- Table: Specify table for which we want to generate a script

```
> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Tab
le HumanResources.Employee
```

In the output, we can see that we get a piece of information about the index and data space, row count along with the table properties information such as FILETABLE, memory optimized, partition table, change tracking.



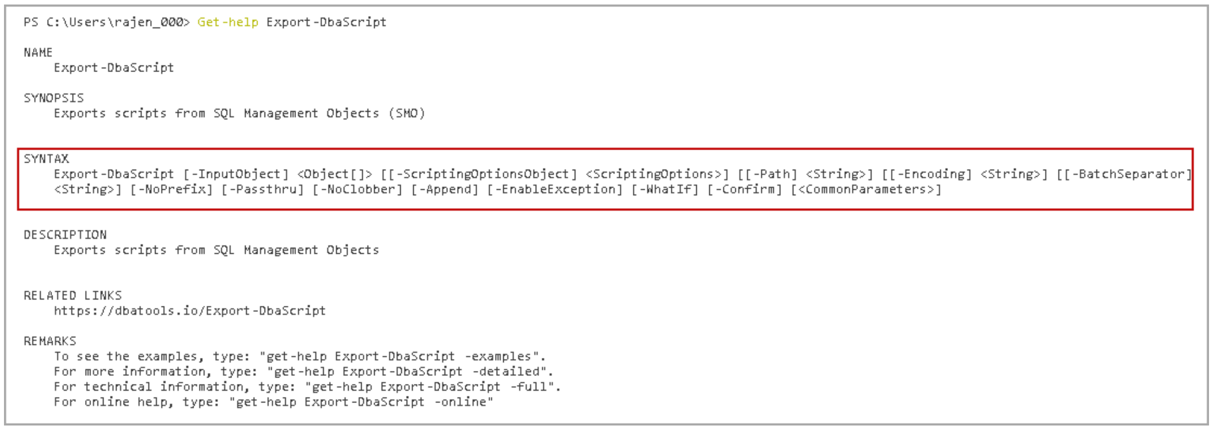
```
PS C:\Users\rajen_000> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Table HumanResources.Employee


ComputerName          : KASHISH
InstanceName          : SQL2019CTP
SqlInstance           : KASHISH\SQL2019CTP
Database              : AdventureWorks2017
Schema                : HumanResources
Name                  : Employee
IndexSpaceUsed        : 136
DataSpaceUsed         : 96
RowCount              : 290
HasClusteredIndex     : True
IsFileTable           : False
IsMemoryOptimized     : False
IsPartitioned         : False
FullTextIndex         :
ChangeTrackingEnabled : False
```

We require generating scripts for the object. We need to use Get-DbaDbTable with the **Export-DbaScript** command to generate a script for the object.

DBATools command **Export-DbaScript** allows exports of SQL Server objects from SQL Management Objects (SMO).

Let's check the synopsis and syntax of **Export-DbaScript** with the following command.

```
> Get-help Export-DbaScript
```



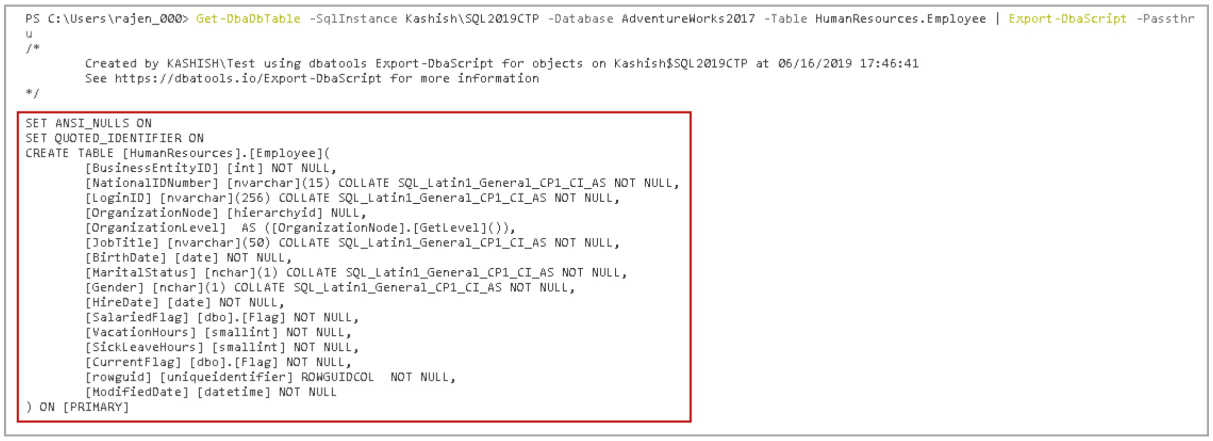
```
PS C:\Users\rajen_000> Get-help Export-DbaScript

NAME
    Export-DbaScript

SYNOPSIS
    Exports scripts from SQL Management Objects (SMO)

SYNTAX
    Export-DbaScript [-InputObject] <Object[]> [[-ScriptingOptionsObject] <ScriptingOptions>] [[-Path] <String>] [[-Encoding] <String>] [[-BatchSeparator]
    <String>] [-NoPrefix] [-Passthru] [-NoClobber] [-Append] [-EnableException] [-WhatIf] [-Confirm] [<CommonParameters>]

DESCRIPTION
    Exports scripts from SQL Management Objects

RELATED LINKS
    https://dbatools.io/Export-DbaScript

REMARKS
    To see the examples, type: "get-help Export-DbaScript -examples".
    For more information, type: "get-help Export-DbaScript -detailed".
    For technical information, type: "get-help Export-DbaScript -full".
    For online help, type: "get-help Export-DbaScript -online"
```

Let's execute DBATools commands **Get-DbaDbTable** and **Export-DbaScript** to generate a script for the object.

```
> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Tab
le HumanResources.Employee | Export-DbaScript -Passthru
```

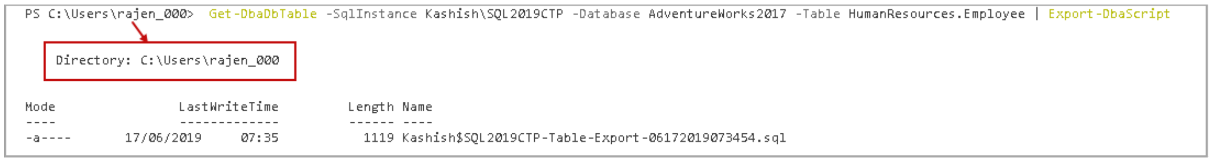
In this script, we use **-Passthru** parameter to display script in the window itself.

It generates the object script; however, we did not get scripts for keys, constraints, indexes. This script might not be useful for us because it does not replicate the source objects and gives only basis object creation script.



```
PS C:\Users\rajen_000> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Table HumanResources.Employee | Export-DbaScript -Passthr
u
/*
        Created by KASHISH\Test using dbatools Export-DbaScript for objects on Kashish$SQL2019CTP at 06/16/2019 17:46:41
        See https://dbatools.io/Export-DbaScript for more information
*/
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL
) ON [PRIMARY]
```

If we execute the above command without **-Passthru** parameter, it saves the script in the current user context. You can go to the directory and open the script in SSMS to go through it.



```
PS C:\Users\rajen_000>  Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Table HumanResources.Employee | Export-DbaScript

    Directory: C:\Users\rajen_000

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       17/06/2019     07:35           1119 Kashish$SQL2019CTP-Table-Export-06172019073454.sql
```

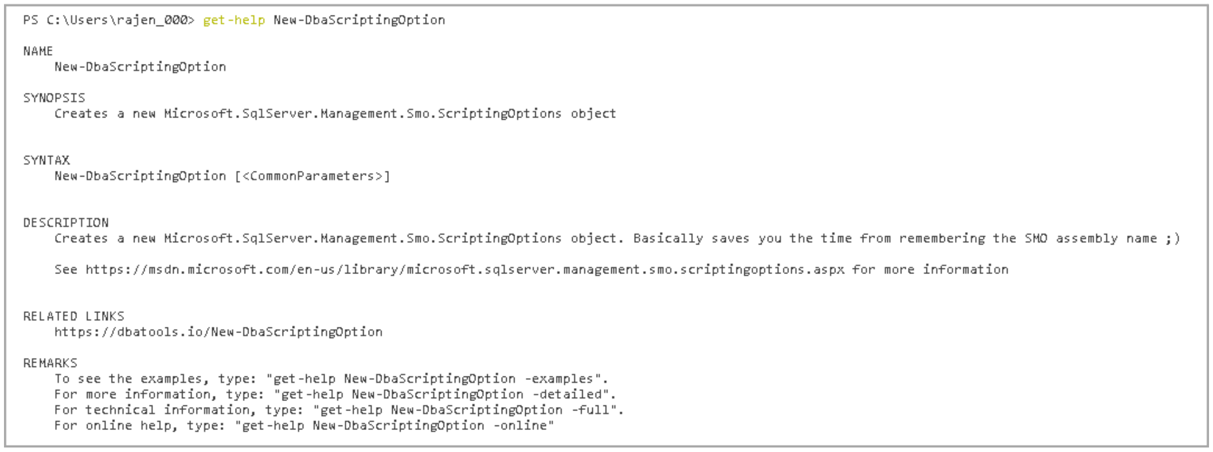
In the SSMS Generate Script Wizard, we set the scripting options under the **Advanced** section. In the DBATools also, we can set the scripting options using the new command **New-DbaScriptingOption.**

Let's explore this command to set scripting options and generate the desired script.

New-DbaScriptingOption command DBATools

In the following screenshot, we can check the details about the New-DbaScriptingOption with the following the command

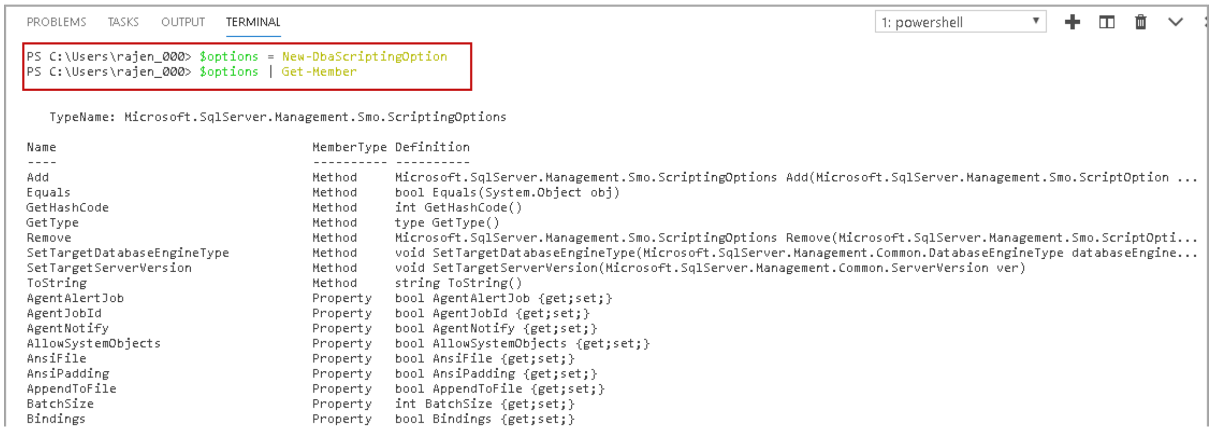
```
>Get-help New-DbaScriptingOption -examples
```



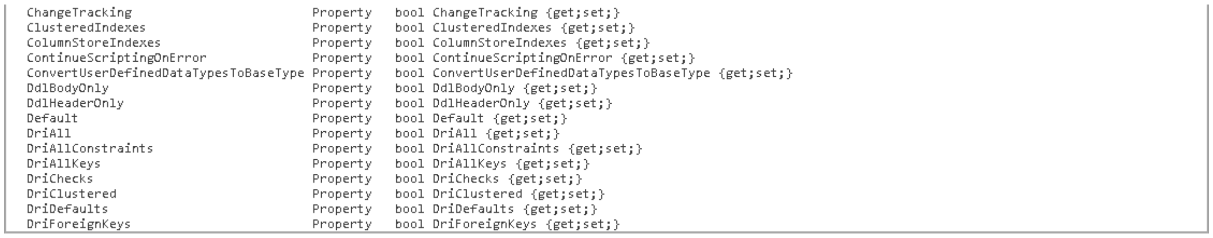
```
PS C:\Users\rajen_000> get-help New-DbaScriptingOption

NAME
    New-DbaScriptingOption

SYNOPSIS
    Creates a new Microsoft.SqlServer.Management.Smo.ScriptingOptions object


SYNTAX
    New-DbaScriptingOption [<CommonParameters>]


DESCRIPTION
    Creates a new Microsoft.SqlServer.Management.Smo.ScriptingOptions object. Basically saves you the time from remembering the SMO assembly name ;)

    See https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.smo.scriptingoptions.aspx for more information


RELATED LINKS
    https://dbatools.io/New-DbaScriptingOption

REMARKS
    To see the examples, type: "get-help New-DbaScriptingOption -examples".
    For more information, type: "get-help New-DbaScriptingOption -detailed".
    For technical information, type: "get-help New-DbaScriptingOption -full".
    For online help, type: "get-help New-DbaScriptingOption -online"
```

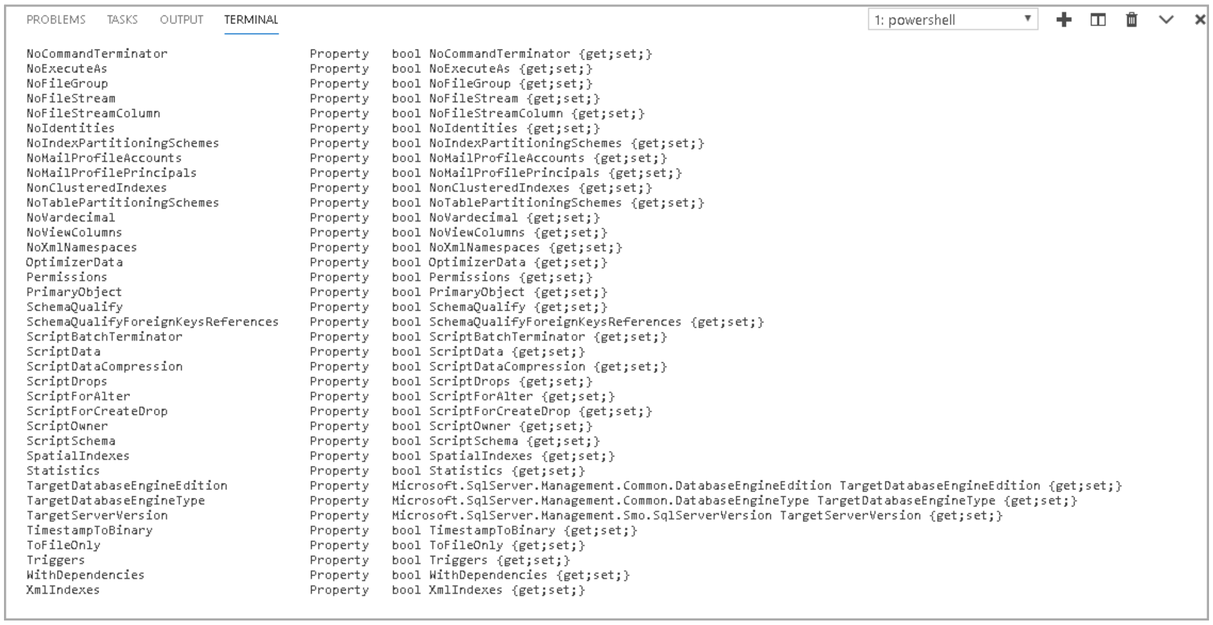
We can check the available scripting options using the Get-Member command. Execute the following command to get a list of available properties along with their definitions.

```
> $options = New-DbaScriptingOption
>$options | Get-Member
```



```
PS C:\Users\rajen_000> $options = New-DbaScriptingOption
PS C:\Users\rajen_000> $options | Get-Member


   TypeName: Microsoft.SqlServer.Management.Smo.ScriptingOptions

Name                                  MemberType Definition
----                                  ---------- ----------
Add                                   Method     Microsoft.SqlServer.Management.Smo.ScriptingOptions Add(Microsoft.SqlServer.Management.Smo.ScriptOption ...
Equals                                Method     bool Equals(System.Object obj)
GetHashCode                           Method     int GetHashCode()
GetType                               Method     type GetType()
Remove                                Method     Microsoft.SqlServer.Management.Smo.ScriptingOptions Remove(Microsoft.SqlServer.Management.Smo.ScriptOpti...
SetTargetDatabaseEngineType           Method     void SetTargetDatabaseEngineType(Microsoft.SqlServer.Management.Common.DatabaseEngineType databaseEngine...
SetTargetServerVersion                Method     void SetTargetServerVersion(Microsoft.SqlServer.Management.Common.ServerVersion ver)
ToString                              Method     string ToString()
AgentAlertJob                         Property   bool AgentAlertJob {get;set;}
AgentJobId                            Property   bool AgentJobId {get;set;}
AgentNotify                           Property   bool AgentNotify {get;set;}
AllowSystemObjects                    Property   bool AllowSystemObjects {get;set;}
AnsiFile                              Property   bool AnsiFile {get;set;}
AnsiPadding                           Property   bool AnsiPadding {get;set;}
AppendToFile                          Property   bool AppendToFile {get;set;}
BatchSize                             Property   int BatchSize {get;set;}
Bindings                              Property   bool Bindings {get;set;}
```


```
ChangeTracking                      Property   bool ChangeTracking {get;set;}
ClusteredIndexes                    Property   bool ClusteredIndexes {get;set;}
ColumnStoreIndexes                  Property   bool ColumnStoreIndexes {get;set;}
ContinueScriptingOnError            Property   bool ContinueScriptingOnError {get;set;}
ConvertUserDefinedDataTypesToBaseType Property bool ConvertUserDefinedDataTypesToBaseType {get;set;}
DdlBodyOnly                         Property   bool DdlBodyOnly {get;set;}
DdlHeaderOnly                       Property   bool DdlHeaderOnly {get;set;}
Default                             Property   bool Default {get;set;}
DriAll                              Property   bool DriAll {get;set;}
DriAllConstraints                   Property   bool DriAllConstraints {get;set;}
DriAllKeys                          Property   bool DriAllKeys {get;set;}
DriChecks                           Property   bool DriChecks {get;set;}
DriClustered                        Property   bool DriClustered {get;set;}
DriDefaults                         Property   bool DriDefaults {get;set;}
DriForeignKeys                      Property   bool DriForeignKeys {get;set;}
```

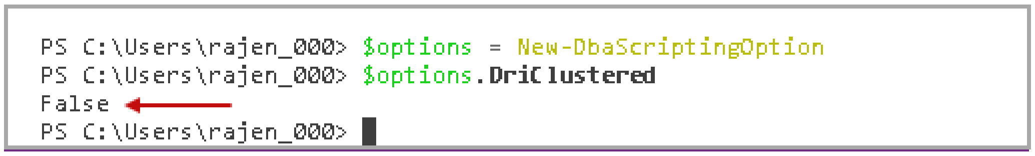


```
NoCommandTerminator                 Property   bool NoCommandTerminator {get;set;}
NoExecuteAs                         Property   bool NoExecuteAs {get;set;}
NoFileGroup                         Property   bool NoFileGroup {get;set;}
NoFileStream                        Property   bool NoFileStream {get;set;}
NoFileStreamColumn                  Property   bool NoFileStreamColumn {get;set;}
NoIdentities                        Property   bool NoIdentities {get;set;}
NoIndexPartitioningSchemes          Property   bool NoIndexPartitioningSchemes {get;set;}
NoMailProfileAccounts               Property   bool NoMailProfileAccounts {get;set;}
NoMailProfilePrincipals             Property   bool NoMailProfilePrincipals {get;set;}
NonClusteredIndexes                 Property   bool NonClusteredIndexes {get;set;}
NoTablePartitioningSchemes          Property   bool NoTablePartitioningSchemes {get;set;}
NoVardecimal                        Property   bool NoVardecimal {get;set;}
NoViewColumns                       Property   bool NoViewColumns {get;set;}
NoXmlNamespaces                     Property   bool NoXmlNamespaces {get;set;}
OptimizerData                       Property   bool OptimizerData {get;set;}
Permissions                         Property   bool Permissions {get;set;}
PrimaryObject                       Property   bool PrimaryObject {get;set;}
SchemaQualify                       Property   bool SchemaQualify {get;set;}
SchemaQualifyForeignKeysReferences  Property   bool SchemaQualifyForeignKeysReferences {get;set;}
ScriptBatchTerminator               Property   bool ScriptBatchTerminator {get;set;}
ScriptData                          Property   bool ScriptData {get;set;}
ScriptDataCompression               Property   bool ScriptDataCompression {get;set;}
ScriptDrops                         Property   bool ScriptDrops {get;set;}
ScriptForAlter                      Property   bool ScriptForAlter {get;set;}
ScriptForCreateDrop                 Property   bool ScriptForCreateDrop {get;set;}
ScriptOwner                         Property   bool ScriptOwner {get;set;}
ScriptSchema                        Property   bool ScriptSchema {get;set;}
SpatialIndexes                      Property   bool SpatialIndexes {get;set;}
Statistics                          Property   bool Statistics {get;set;}
TargetDatabaseEngineEdition         Property   Microsoft.SqlServer.Management.Common.DatabaseEngineEdition TargetDatabaseEngineEdition {get;set;}
TargetDatabaseEngineType            Property   Microsoft.SqlServer.Management.Common.DatabaseEngineType TargetDatabaseEngineType {get;set;}
TargetServerVersion                 Property   Microsoft.SqlServer.Management.Smo.SqlServerVersion TargetServerVersion {get;set;}
TimestampToBinary                   Property   bool TimestampToBinary {get;set;}
ToFileOnly                          Property   bool ToFileOnly {get;set;}
Triggers                            Property   bool Triggers {get;set;}
WithDependencies                    Property   bool WithDependencies {get;set;}
XmlIndexes                          Property   bool XmlIndexes {get;set;}
```

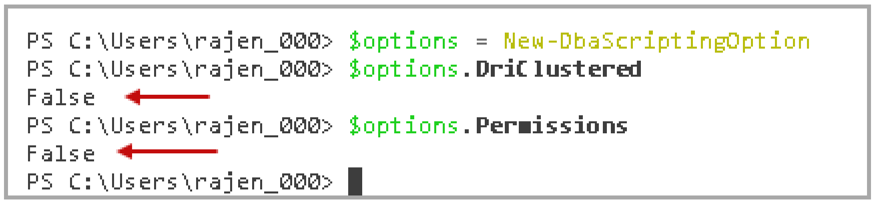
We can check the value of individual property as well. For example, let's check the value of the property DriClustered.

```
> $options = New-DbaScriptingOption
> $options.DriClustered
```

We get the return value **False** for the **DriClustered** parameter. It is the reason that the generated script using DBATools does not contain the clustered index information.


```
PS C:\Users\rajen_000> $options = New-DbaScriptingOption
PS C:\Users\rajen_000> $options.DriClustered
False
PS C:\Users\rajen_000>
```

Similarly, we can check value for other properties.


```
PS C:\Users\rajen_000> $options = New-DbaScriptingOption
PS C:\Users\rajen_000> $options.DriClustered
False
PS C:\Users\rajen_000> $options.Permissions
False
PS C:\Users\rajen_000>
```

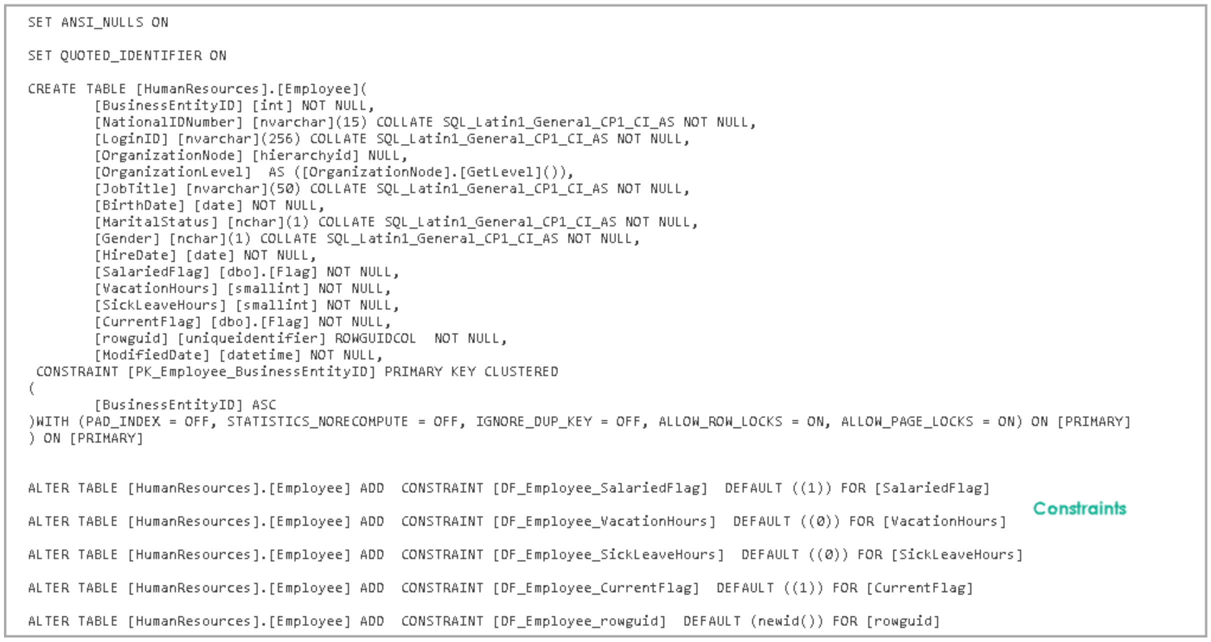
We can change the value of required properties to TRUE and use $options object along with the parameter –**ScriptingOptionsObject** to generate the script with these objects.

Add constraints in object creation script using DBATools

Suppose we want to add all constraints in the object script. Execute the following script to change the value to TRUE for DriAllConstraints and generate the script.

```
> $options = New-DbaScriptingOption
> $options.DriAllConstraints =$true
> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Tab
le HumanResources.Employee | Export-DbaScript -Passthru -ScriptingOptionsObject $o
ptions
```

We can see constraints as well in the script for the specified object.



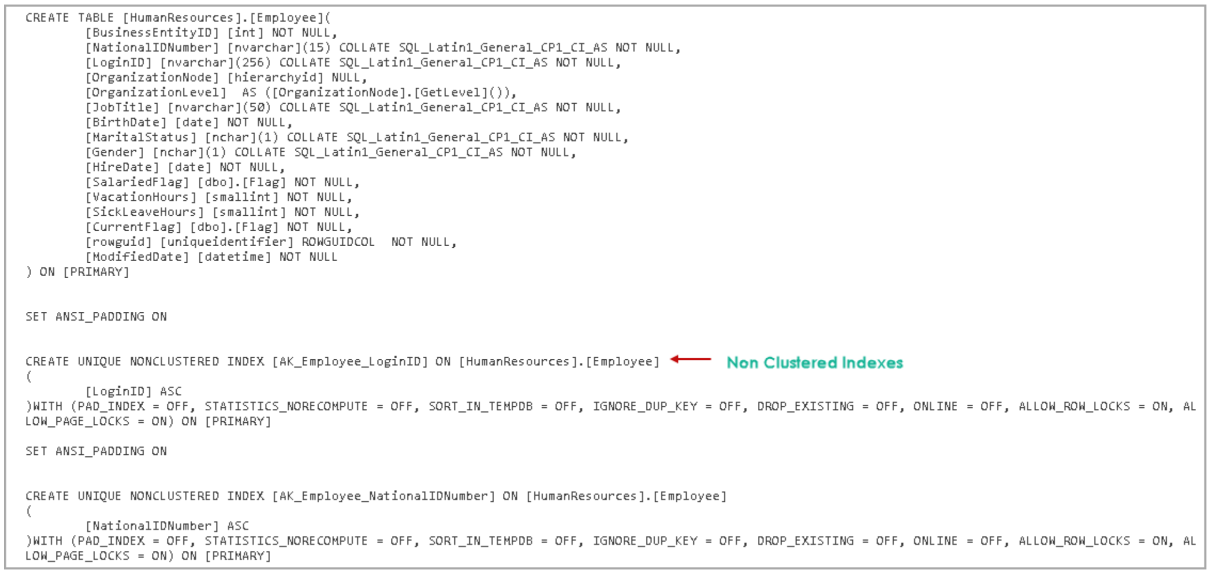
```
SET ANSI_NULLS ON

SET QUOTED_IDENTIFIER ON

CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL,
 CONSTRAINT [PK_Employee_BusinessEntityID] PRIMARY KEY CLUSTERED
(
        [BusinessEntityID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]


ALTER TABLE [HumanResources].[Employee] ADD  CONSTRAINT [DF_Employee_SalariedFlag]  DEFAULT ((1)) FOR [SalariedFlag]

ALTER TABLE [HumanResources].[Employee] ADD  CONSTRAINT [DF_Employee_VacationHours]  DEFAULT ((0)) FOR [VacationHours]

ALTER TABLE [HumanResources].[Employee] ADD  CONSTRAINT [DF_Employee_SickLeaveHours]  DEFAULT ((0)) FOR [SickLeaveHours]

ALTER TABLE [HumanResources].[Employee] ADD  CONSTRAINT [DF_Employee_CurrentFlag]  DEFAULT ((1)) FOR [CurrentFlag]

ALTER TABLE [HumanResources].[Employee] ADD  CONSTRAINT [DF_Employee_rowguid]  DEFAULT (newid()) FOR [rowguid]
```

Add Non clustered indexes in object creation script using DBATools

Let’s do the following things for this example.

- We do not want to add constraints in the script, therefore, change constraint **DriAllConstraints** property to false
- We want to add all non-clustered indexes in the script, therefore change constraint **NonClusteredIndexes** property to true

```
> $options = New-DbaScriptingOptionPS
> $options.DriAllConstraints =$false
> $options.NonClusteredIndexes  =$true
>  Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Ta
ble HumanResources.Employee | Export-DbaScript -Passthru -ScriptingOptionsObject
$options
```



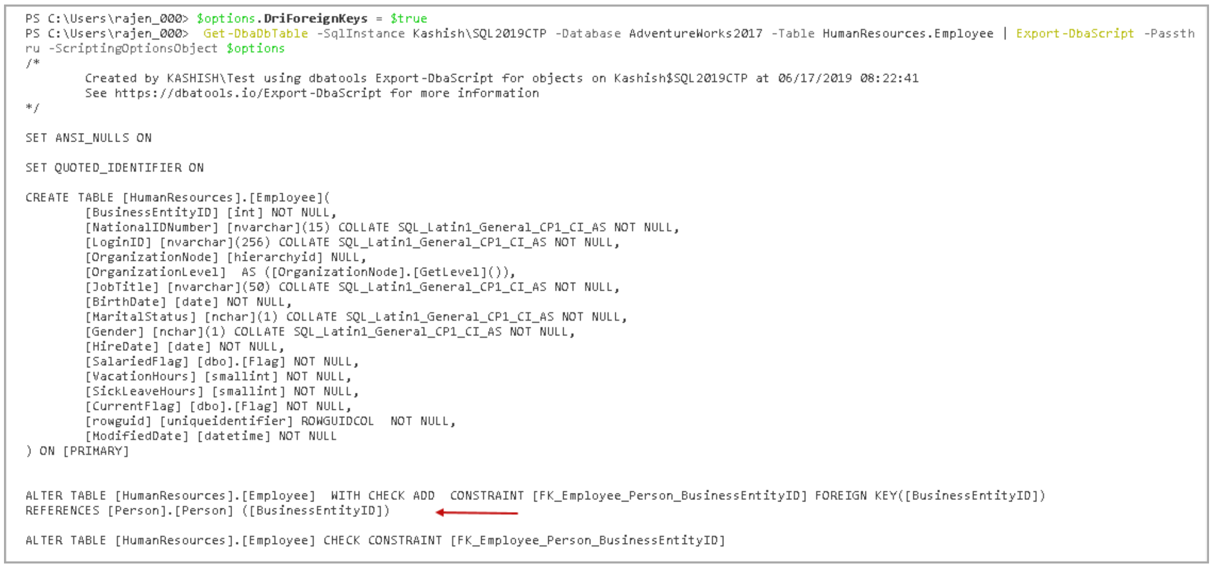
```
CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL
) ON [PRIMARY]

SET ANSI_PADDING ON

CREATE UNIQUE NONCLUSTERED INDEX [AK_Employee_LoginID] ON [HumanResources].[Employee]
(
        [LoginID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, AL
LOW_PAGE_LOCKS = ON) ON [PRIMARY]

SET ANSI_PADDING ON

CREATE UNIQUE NONCLUSTERED INDEX [AK_Employee_NationalIDNumber] ON [HumanResources].[Employee]
(
        [NationalIDNumber] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, AL
LOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

Add Foreign key in object creation script using DBATools

Let's try with a few other interesting options. Suppose we want foreign keys in the object scripts. We need to enable DriForeignKeys parameter and execute the script as follows

```
> $options.DriForeignKeys = $true
> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Tab
le HumanResources.Employee | Export-DbaScript -Passthru -ScriptingOptionsObject $o
ptions
```

In the output, we can see foreign key constraints along with the object creating script.



```
PS C:\Users\rajen_000> $options.DriForeignKeys = $true
PS C:\Users\rajen_000>  Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Table HumanResources.Employee | Export-DbaScript -Passth
ru -ScriptingOptionsObject $options
/*
        Created by KASHISH\Test using dbatools Export-DbaScript for objects on Kashish$SQL2019CTP at 06/17/2019 08:22:41
        See https://dbatools.io/Export-DbaScript for more information
*/

SET ANSI_NULLS ON

SET QUOTED_IDENTIFIER ON

CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL
) ON [PRIMARY]

ALTER TABLE [HumanResources].[Employee]  WITH CHECK ADD  CONSTRAINT [FK_Employee_Person_BusinessEntityID] FOREIGN KEY([BusinessEntityID])
REFERENCES [Person].[Person] ([BusinessEntityID])

ALTER TABLE [HumanResources].[Employee] CHECK CONSTRAINT [FK_Employee_Person_BusinessEntityID]
```

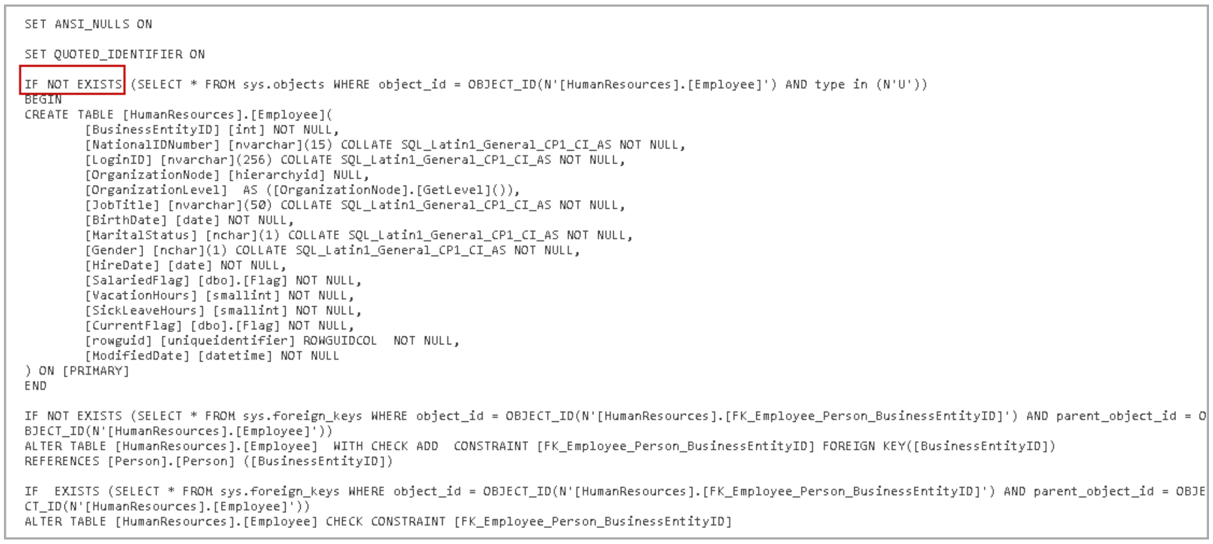
Add If Not Exists in object creation script using DBATools

It is a good practice to check whether the object exists or not before we create an object. We might

It is a good practice to check whether the object exists or not before we create an object. We might have another object with a similar name. We use SQL Exists operator to test the existence of an object in the SQL Server database.

Our script should include Not Exists operator, and we should create an object if it does not exists. We need to enable parameter IncludeIfNotExists to the **true** and generated script will contain the IF EXISTS clause.

```
>$options.IncludeIfNotExists = $true
> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Tab
le HumanResources.Employee | Export-DbaScript -Passthru -ScriptingOptionsObject $o
ptions
```



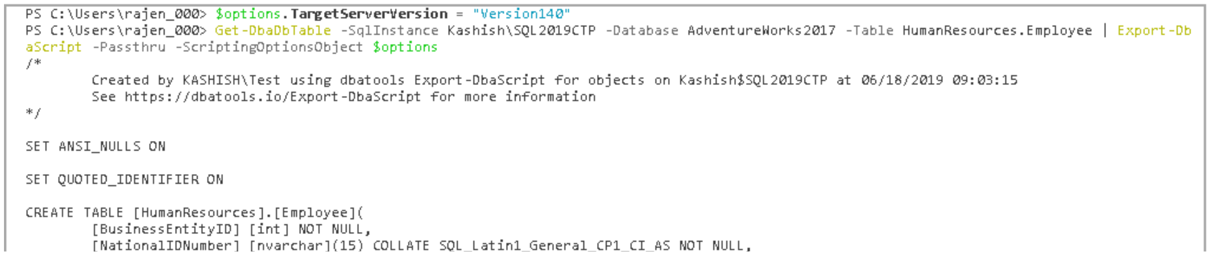
```
SET ANSI_NULLS ON

SET QUOTED_IDENTIFIER ON

IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[HumanResources].[Employee]') AND type in (N'U'))
BEGIN
CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL
) ON [PRIMARY]
END

IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = OBJECT_ID(N'[HumanResources].[FK_Employee_Person_BusinessEntityID]') AND parent_object_id = O
BJECT_ID(N'[HumanResources].[Employee]'))
ALTER TABLE [HumanResources].[Employee]  WITH CHECK ADD  CONSTRAINT [FK_Employee_Person_BusinessEntityID] FOREIGN KEY([BusinessEntityID])
REFERENCES [Person].[Person] ([BusinessEntityID])

IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = OBJECT_ID(N'[HumanResources].[FK_Employee_Person_BusinessEntityID]') AND parent_object_id = OBJE
CT_ID(N'[HumanResources].[Employee]'))
ALTER TABLE [HumanResources].[Employee] CHECK CONSTRAINT [FK_Employee_Person_BusinessEntityID]
```

Specify a target version to generate a script using DBATools

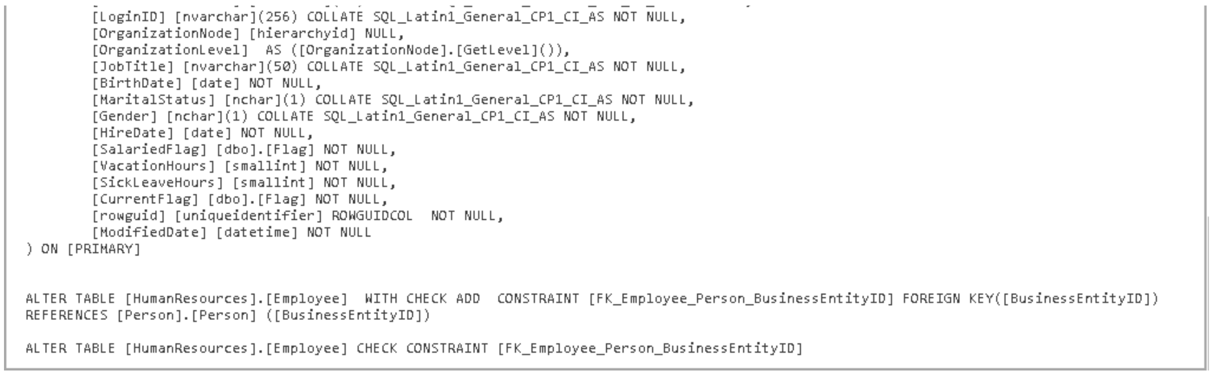
We might have a different version of the destination SQL Server for which we want to generate a script. For example, I want to generate a script for SQL Server Compatibility level 140, whereas the source compatibility level is 150.

We want to set the TargetServerVersion parameter for the SQL Server compatibility level we want to generate the script.

```
>$options.TargetServerVersion = "Version140"
>Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Tabl
e HumanResources.Employee | Export-DbaScript -Passthru -ScriptingOptionsObject $op
tions
```



```
PS C:\Users\rajen_000> $options.TargetServerVersion = "Version140"
PS C:\Users\rajen_000> Get-DbaDbTable -SqlInstance Kashish\SQL2019CTP -Database AdventureWorks2017 -Table HumanResources.Employee | Export-Db
aScript -Passthru -ScriptingOptionsObject $options
/*
        Created by KASHISH\Test using dbatools Export-DbaScript for objects on Kashish$SQL2019CTP at 06/18/2019 09:03:15
        See https://dbatools.io/Export-DbaScript for more information
*/

SET ANSI_NULLS ON

SET QUOTED_IDENTIFIER ON

CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
```



```
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL
) ON [PRIMARY]

ALTER TABLE [HumanResources].[Employee]  WITH CHECK ADD  CONSTRAINT [FK_Employee_Person_BusinessEntityID] FOREIGN KEY([BusinessEntityID])
REFERENCES [Person].[Person] ([BusinessEntityID])

ALTER TABLE [HumanResources].[Employee] CHECK CONSTRAINT [FK_Employee_Person_BusinessEntityID]
```

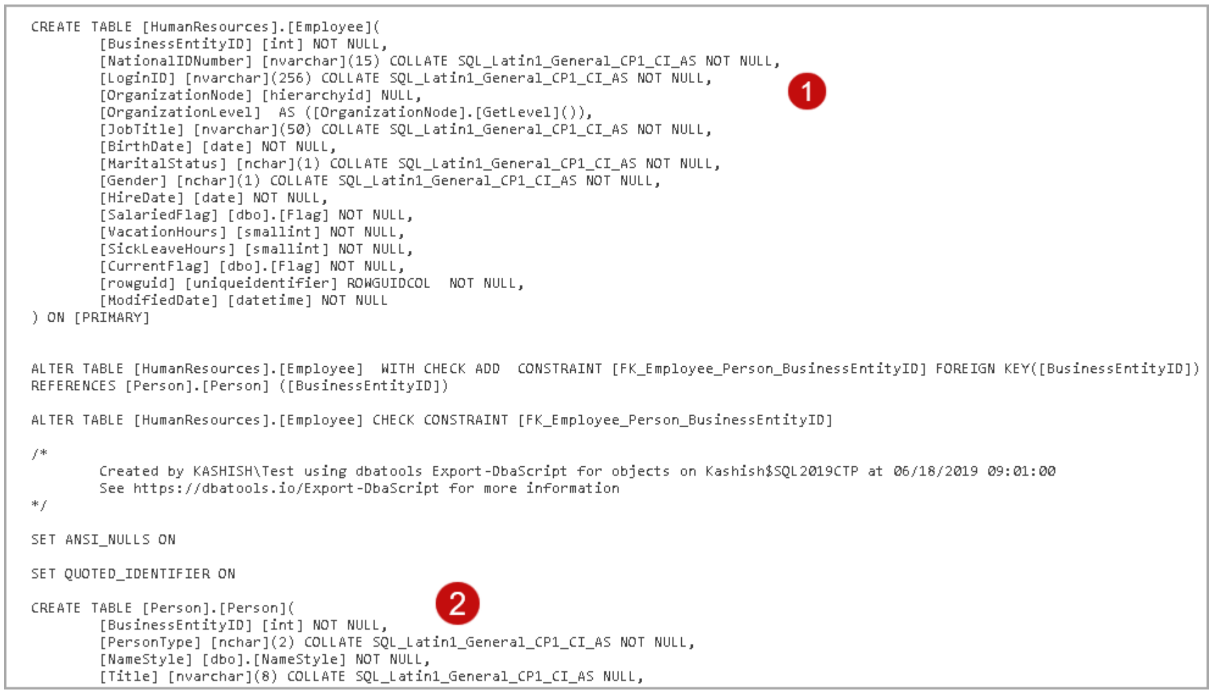
Generate scripts for multiple objects together using DBATools

In previous examples, we generated a script for an object using DBATools. We might want to generate scripts for multiple objects. We can select multiple objects in the Generate Scripts wizard of SSMS. In the DBATools also we can do it using variables.

In the following query, we defined a $Tablename variable, and it includes two table names. We use Foreach-object loop to go through each table and generate the required script. We can consider Foreach-object loop similar to a loop in SQL Server.

```
> $TableName = "HumanResources.Employee", 'Person.Person';
> $options = New-DbaScriptingOption
> $options.DriForeignKeys = $true
> $TableName | Foreach-Object
>> Get-DbaDbTable -ServerInstance Kashish\SQL2019CTP -Database AdventureWorks2017
$SourceDB -Table $_ | Export-DbaScript -ScriptingOptionsObject $options -Passthru;
>>}
```

In the following screenshot, we can see it generated scripts for both the objects.



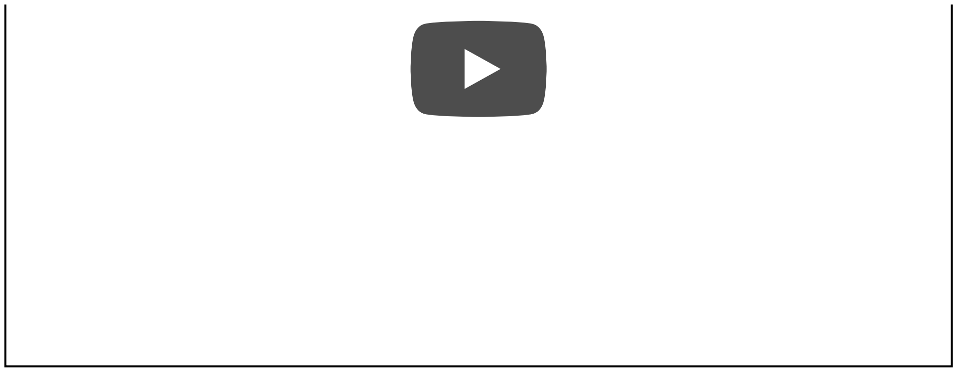
```
CREATE TABLE [HumanResources].[Employee](
        [BusinessEntityID] [int] NOT NULL,
        [NationalIDNumber] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [LoginID] [nvarchar](256) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [OrganizationNode] [hierarchyid] NULL,
        [OrganizationLevel]  AS ([OrganizationNode].[GetLevel]()),
        [JobTitle] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [BirthDate] [date] NOT NULL,
        [MaritalStatus] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [Gender] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [HireDate] [date] NOT NULL,
        [SalariedFlag] [dbo].[Flag] NOT NULL,
        [VacationHours] [smallint] NOT NULL,
        [SickLeaveHours] [smallint] NOT NULL,
        [CurrentFlag] [dbo].[Flag] NOT NULL,
        [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
        [ModifiedDate] [datetime] NOT NULL
) ON [PRIMARY]

ALTER TABLE [HumanResources].[Employee]  WITH CHECK ADD  CONSTRAINT [FK_Employee_Person_BusinessEntityID] FOREIGN KEY([BusinessEntityID])
REFERENCES [Person].[Person] ([BusinessEntityID])

ALTER TABLE [HumanResources].[Employee] CHECK CONSTRAINT [FK_Employee_Person_BusinessEntityID]

/*
        Created by KASHISH\Test using dbatools Export-DbaScript for objects on Kashish$SQL2019CTP at 06/18/2019 09:01:00
        See https://dbatools.io/Export-DbaScript for more information
*/

SET ANSI_NULLS ON

SET QUOTED_IDENTIFIER ON

CREATE TABLE [Person].[Person](
        [BusinessEntityID] [int] NOT NULL,
        [PersonType] [nchar](2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [NameStyle] [dbo].[NameStyle] NOT NULL,
        [Title] [nvarchar](8) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
```

Conclusion

In this article, we explored to generate a script using DBATools Windows PowerShell commands. We can use DBATools to automate these scripts and run as per our requirements. I would suggest reviewing them as per your environment. If you have any comments or questions, feel free to leave them in the comments below.

Table of contents

See more



Rajendra Gupta

Rajendra has 8+ years of experience in database administration having a passion for database performance optimization, monitoring, and high availability and disaster recovery technologies, learning new things, new features.

While working as a Senior consultant DBA for big customers and having certified with MCSA SQL 2012, he likes to share knowledge on various blogs.
He can be reached at rajendra.gupta16@gmail.com

View all posts by Rajendra Gupta

Related Posts:

1. **Fix Orphan users in SQL Server using DBATools PowerShell**
2. **DBATools PowerShell SQL Server Database Backups commands**
3. **SQL Restore Database using DBATools**
4. **Validate backups with SQL restore database operations using DBATools**
5. **IDENTITY columns threshold using PowerShell SQL Server DBATools**

DBAtools, PowerShell

2,472 Views

ALSO ON **SQL SHACK**

onths ago · 1 comment
QL While loop: understanding …

2 months ago · 1 comment
SQL Server Linux installation for

2 r
S Data Extra

Comments Community Privacy Policy 1 Login

Recommend 1 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Martin Gibson • 8 months ago

Thanks Rajendra - that's very useful

1 • Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd