Generic generators

SQL Data Generator provides the following generators in the Generic category for you to customize:

- CSV
- File Import
- File List
- Python script
- Regexp
- Simple expression
- SQL Statement
- Text Shuffler
- Weighted List

Information about each of these generators is provided below.

For information about how to customize the generators, see Customizing existing generators.

CSV generator

Use the CSV generator when you want to import data from a CSV file into a single column. (If you want to import data from a CSV file into an entire table or multiple columns in a table, you can use the **Use existing data source** table generation setting instead; for details, see Mapping CSV files.)

Click **Browse** to select the CSV file you want to use; you then specify the delimiters to be used when importing the data, and select the column in the CSV file that you want to import.

File Import generator

Use the File Import generator to import the contents of files in a specified folder.

For example, if you specify a folder containing a number of images, each image is imported into a new row. You can specify a search string to identify the files within the specified folder you want to use.

If you specify large files, or if you specify a large number of files, performance will be reduced.

File List generator

Use the File List generator to import values from a text file.

You must first create a text file containing the list of values, with each value on a new line. The values will be imported from the list in a random order. You can then browse to this file when you select the File List generator.

If you have a very long list of values, you may want to consider creating a CSV file with the list of values and then importing the values using the CSV generator to import the values.

Python script generator

Use the Python script generator to define the generated data using an IronPython script.

main(config) function

A script must define a `main()` function that takes a single `config` argument. `config` is a dictionary that contains values you can use in your scripts. Values are specified for the following keys:

Key	Description	Example value
`column_name`	The name of a column	*First Name*
`column_type`	The column's data type	*varchar*
`column_size`	The column's size (for char columns)	*40*
`n_rows`	The number of rows to generate	*1000*
`seed`	The generator's seed value	*10*
`is_unique`	The uniqueness of a column	*True*
`connection_string`	The connection string for the database	*Data Source=(local);initial catalog="ADatabase";Integrated security=SSPI;*

`generator_path`	The path to the installed Generators folder	*C:\Program Files (x86)\Red Gate\SQL Data Generator 3\Generators*
`config_path`	The path to the installed Config (External files) folder	*C:\Program Files (x86)\Red Gate\SQL Data Generator 3\Config*
`user_config_path`	The path to the user-defined generators folder	*C:\Users\an.other\AppData\Roaming\Red Gate\SQL Data Generator\Config*

The `main()` function can return an iterable object. The object's elements will be used to populate the column. For example, it could return a list of strings, or a tuple of integers. You can use the `yield` keyword to return values lazily.

If the `main()` function does not return an iterable, it will be run once for each row. The values of other columns in this row can be referenced by their names.

For information on how to write Python scripts, see the Python tutorial (Python documentation).

For a list of example scripts you can use with SQL Data Generator, see Example Python scripts.

Regexp generator

Use the Regexp generator to define the generated data using a regular expression.

In the basic syntax, most characters are treated as literals (for example, a generates "a"). Below is a list of syntax elements.

Syntax	Example	Generates
`ordinary chars`	`bob`	`bob`
`[chars]character set`	`[A-Z0- 9]`	`eg. 5 or G`
`individual chars`	`[FM]`	`F or M`
`initial ] in char set`	`[]]`	`]`
`[x-y] range`	`[0-9]`	`eg. 3 or 9`
`complement`	`[^abc]`	`eg. d or #`
`* zero or more`	`abc*`	`eg. abccccccc or ab`
`+ 1 or more`	`abc+`	`eg. abcccc or abc`
`? Include or not`	`abc?`	`ab or abc`
`(regexp) grouping`	`(abc)*d`	`eg., abcabcd or d`
`{num} repeat`	`a{4}`	`aaaa`
`{min,max} repeat`	`a{2,3}`	`aa or aaa`
`{min,} at least min repeats`	`a{3,}`	`eg. aaa or aaaaaaaaaa`
`() empty string`	`()`	
`| alternatives`	`Yes|No`	`Yes or No`
`Empty Alternative`	`(|some- |often-)time`	`eg. some-time or often-time`

Escapes

Syntax	Generates
`\xdd hex char (8- bit)`	`eg. \x21 generates !`
`\udddd hex unicode`	`eg. \u0021 generates !`
`\\`	`\`
`\.`	`.`
`\^`	`^`
`\$`	`$`

\{	{
\[	[
\]	\]
\(	(
|	|
\)	)
*	*
\+	+
\?	?
\a	alarm character
\b	backspace
\d	digit
\e	escape
\f	formfeed
\n	newline
\t	tab
\r	carriage return
\s	space
\v	vertical tab
\w	[A-Za- z_0-9]

Use `$"file_name"` to import a text file containing a list of values.

Use `$[column_name]` to import the values from another column in the same table.

When you import values from another column:

- for repeating values, you must specify a fixed range of repetitions using `{1,10}`; you cannot use * when you import values from a column
- changing the **Seed** value in the RegexpGenerator settings changes only the position of any null values; to shuffle the order of the values, use the settings on the referenced column
- you shouldn't select **Set unique** in the RegexpGenerator settings, because only one row will be generated

You can use the buttons below the **Regular Expression** box to add commonly-used components for a regular expression, file lists, and table columns.

Simple expression generator

With the Simple expression generator, you can write a one line expression to generate data. An expression can use:

- data from other columns in the table
- Python functions
- a set of C# DateTime methods

If you want to write an expression that's more than one line, we recommend you use the Python script generator.

Python Functions

You can use Python functions with the Simple expression generator, including:

Boolean

and, or

Random

randint, choice, uniform, random

String

lower, upper

Math

factorial, log, pow, sqrt, sin, cos, tan

C# methods

You can also use the following C# DateTime methods with the Simple expression generator:

DateTime

Now, AddHours, AddDays, AddMonths, TotalDays

Example expressions

Eligibile to vote

This expression uses a table's *BirthDate* column to determine if a person is eligible to vote:

```
(DateTime.Now - BirthDate).TotalDays/365 > 18
```

The expression generates the values *True* or *False*.

Add days to date

This expression adds 100 days to values in a *Date* column:

```
Date.AddDays(100)
```

Exam pass or fail

This expression uses a table's *ExamScore* column to determine if a student has passed or failed an exam:

```
"Pass" if ExamScore <= 75 else "Fail"
```

The expression generates the values *Pass* or *Fail.*

Offset time using fixed intervals

This expression offsets values from a table's *BookingTime* column by intervals of 20 minutes between 6am and 9am:

```
BookingTime.Date.AddHours(random.choice([6,7,8,9])).AddMinutes(random.choice([0, 20, 40]))
```

SQL Statement generator

Use the SQL Statement generator to define data to import from an external database using a SQL statement. If you want to import data from an external database into a an entire table or multiple columns in a table, you can use the **Use existing data source** table generation setting instead; for details, see M apping SQL tables or views.

Click **Edit** to specify the SQL Server and database, and then click **Next** to enter the SQL statement. The SQL statement must select a single column of values which are of the correct data type.

If the column you are populating has a unique constraint, make sure the SQL statement returns unique values; if it does not, the data generation will fail.

When you select **Shuffle data**, changing the **Seed** value in the SQL Statement settings changes only the position of any null values.

Text Shuffler generator

Use the Text Shuffler generator when you want to create values that contain words randomly selected from a pre-defined list. For example, you may want to use this to check the performance of a full text index.

You can type the text, or you can import it from a text file. You are recommended to use text that contains a high variety of words that are similar to your real data.

SQL Data Generator shuffles the text using the spaces as delimiters for the words. Therefore, if there is punctuation in the text, this will be included in the values.

Weighted List generator

Use this generator when you want to specify the percentage for the number of occurrences of each value in the column. For example, you may want to use this to check that your indexing strategy will work on the table.